,

Universidade de Brasília – UnB
Campus Gama – FGA
Engenharia Eletrônica

Análise de Configurações de Mão em Linguagem Brasileira de Sinais com base em Aprendizagem de Máquina sobre Sequências de Imagens com Subamostragem e Reescalonamento Temporal

Bruno Carvalho Faria dos Santos

Orientador: Cristiano Jacques Miosso

UnB – Universidade de Brasília

FGA – Faculdade Gama

Engenharia Eletrônica

Análise de Configurações de Mão em Linguagem Brasileira de Sinais com Base em Aprendizagem de Máquina sobre Sequências de Imagens com Subamostragem e Reescalonamento Temporal

Bruno Carvalho Faria dos Santos

Orientador: Cristiano Jacques Miosso

Trabalho de Conclusão de Curso
Engenharia Eletrônica

Brasília/DF, Maio de 2021

UnB – Universidade de Brasília

FGA – Faculdade Gama

Engenharia Eletrônica

Análise de Configurações de Mão em Linguagem Brasileira de Sinais com Base em Aprendizagem de Máquina sobre Sequências de Imagens com Subamostragem e Reescalonamento Temporal

Bruno Carvalho Faria dos Santos

Trabalho de Conclusão de Curso submetido à Faculdade UnB Gama da Universidade de Brasília, como parte dos requisitos necessários para a obtenção do grau de bacharel em engenharia eletrônica

Aprovada por:

Cristiano Jacques Miosso, PhD

(Orientador)

Fernando Vinícius Gonçalvez de Souza, MSc

Filipe Emídio Tôrres, MSc

Brasília/DF, Maio de 2021

Ficha Catalográfica

dos Santos, Bruno Carvalho Faria
Análise de Configurações de Mão em Linguagem Brasileira de Sinais
com Base em Aprendizagem de Máquina sobre Sequências
de Imagens com Subamostragem e Reescalonamento Temporal
[Distrito Federal], 2021.
100p., 210 × 297 mm (FGA/UnB Gama, Bacharelado em Engenharia Eletrônica, 2021).
Trabalho de Conclusão de Curso, Faculdade UnB Gama, Engenharia Eletrônica

1. Aprendizagem de Máquina
2. Processamento de Imagens
3. *LIBRAS*
4. Deficiência Auditiva

I. FGA UnB/UnB.

Referência

dos Santos, Bruno Carvalho Faria (2021). Análise de Configurações de Mão em Linguagem Brasileira de Sinais com Base em Aprendizagem de Máquina sobre Sequências de Imagens com Subamostragem e Reescalonamento Temporal. Trabalho de Conclusão de Curso, Engenharia Eletrônica, Faculdade UnB Gama, Universidade de Brasília, Brasília, DF, 100p.

Cessão de Direitos

---

brunofga1@gmail.com

Brasília, DF – Brasil

Epígrafe
Autor da epígrafe

Dedico esse trabalho primeiramente ao meu esforço, dedicação e perseverança, aos anos que passei na FGA que foram de bastante aprendizado acadêmico, científico, intrapessoal e interpessoal.

Um agradecimento especial à minha família, meu pai Edvalso Faria dos Santos, minha mãe Domingas Carvalho Faria, e meu irmão Tiago Carvalho Faria dos Santos, além de todos meus parentes de ambas famílias Carvalho e Faria, que estiveram comigo durante todo esse tempo, apoiando meus estudos, respeitando e incentivando meu crescimento dentro do espaço de tempo necessário.

Agradeço ao professor Doutor Cristiano Jacques Miosso por ter aceitado essa missão de me orientar neste trabalho, o seu auxílio foi imensurável, todas reuniões que tivemos ao longo destes dois semestres letivos sempre possibilitaram meu crescimento, adquiri tantos novos conhecimentos neste curto período de tempo, dicas de programação, de comportamento em entrevistas de emprego, uma orientação para a vida, sempre disposto a ajudar e tirar dúvidas, mesmo com minhas dificuldades e os desafios que encontramos no meio do caminho, finalizamos esta etapa de uma maneira satisfatória para mim, e espero que para o senhor também, e quem sabe no futuro nossa trajetória não se cruze de novo para novos projetos.

Agradecimentos

Gostaria de agradecer a todo corpo docente da Faculdade do Gama da Universidade de Brasília, tanto os professores que lecionaram nas turmas das disciplinas que fiz, quanto das outras turmas, dos outros cursos de engenharia da FGA e da UnB, aos técnicos do laboratório NEI por serem sempre tão atentos e solicitos para auxiliar os alunos, e a todos servidores e terceirizados que indiretamente também fizeram parte dessa jornada mantendo a faculdade sempre funcionando para que nós alunos pudéssemos dar o nosso melhor, e não menos importante a todos os cidadãos brasileiros que indiretamente financiam a educação, que infelizmente nem sempre os alcança como deveria.

Um voto de agradecimento a todos que juntas fizeram, fazem e farão da UnB parte da sua história, nestes anos pude perceber o quão privilegiado fui de ter acesso a uma educação de qualidade, pública e gratuita e espero que em um futuro não muito distante esse privilégio torne-se uma possibilidade para todos, fui capaz de refletir meu local na sociedade, como aluno, cidadão, trabalhador, não apenas me formando como um engenheiro, e sim como um humano e com isso capaz de dizer o quanto a educação é libertadora, um viva à educação.

Finalmente um agradecimento aos meus amigos, Vanessa, Ana Paula, Sannya, Breno Amadeus, Julie Delchova, Pedro Helias, Elpídio Bisneto, Danny, Lipe, Matteus Nascimento, Caio Cardoso, Victor Hugo Ciurlini, Guilherme Richard, Hebert Max e toda a galera da graminha, Carlos Eduardo "Cadu", Igor Mickael, Joaquim Vieira, Tiago Rodrigues, Nithielle Lanay, Wesley Sales, Jefferson Soares, Mateus Torquato, Thiago Alves, Gustavo Rodrigues, amigos do ambiente acadêmico e fora dele, que por muitas vezes eram meu refúgio para falar do meu dia, comentar minhas frustrações, alegrias, que entendem e são parte muito importante de toda minha caminhada acadêmica, e muitos outros amigos que dariam um outro TCC, obrigado a todos que estiveram ao meu lado, e que estão me vendo aqui, concluindo esta nova etapa, espero que sigamos crescendo juntos nos novos ciclos que chegarão.

Resumo

A área de Inteligência Artificial(IA) teve seus primeiros paradigmas apresentados nas décadas de 50 e 60, essa nova área foi apresentada como uma poderosa ferramenta na busca de otimizar processos de identificação de padrões, e um campo com aplicações variadas, o campo teve uma expansão no fim da década de 90 e início dos anos 2000 com criação de novas metodologias que resolviam lacunas preexistentes na área, e sua ascenção na década de 2010 com uma ampla participação da comunidade acadêmica, científica, curiosos e simpatizantes de tecnologias.

A busca pela identificação de padrões é a natureza do trabalho proposto, a Língua Brasileira de Sinais é a segunda língua oficial do Brasil, e é a principal forma de comunicação de pessoas surdas-mudas ou com algum grau de deficiência auditiva que tenham sido alfabetizadas em escolas de ensino especial ou bilíngue, mesmo sendo oficial, não é difundida em toda sociedade, logo, uma barreira comunicativa existe entre a comunidade gesticulante e os não-gesticulantes que constituem a maior parte da população.

A área de aprendizagem de máquina possui inúmeros classificadores que atendem as mais diversas problemáticas, a CNN e a SVM são dois classificadores de amplo conhecimento da comunidade científica para reconhecimento de padrões, neste trabalho foram realizados testes que atestaram a CNN como o classificador mais adequado ao problema.

A base de dados usada é composta de vídeos com as 61 configurações de mão que compõe a LIBRAS, os vídeos foram pré-processados sendo submetidos à aplicação de algoritmos de data augmentation, subamostragem das configurações dadas as características distintas, e reescalonamento temporal da quantidade de frames, de modo que a aprendizagem de máquina gerasse um classificador com boa capacidade de generalização.

Os vídeos resultantes da base pré-processada foram convertidos em um arquivo de dados binários para se adequar ao recurso computacional disponível, a determinação da quantidade de camadas do classificador e os valores dos hiperparâmetros, foi realizada por testes e adequação ao sistema, uma vez que a aplicação é específica e não foram encontrados valores base no estudo bibliográfico.

Os resultados das métricas obtidas nos treinamentos realizados sugerem que a abordagem proposta pode vir futuramente a resolver o problema da classificação automática das configurações de mão, pela: (i) classificação com bom desempenho (acurácia, precisão, sensibilidade da ordem de 100%, 100%, 100%) no caso de imagens estáticas mesmo que as configurações de mão sejam distintas, e que o classificador tivesse sido treinado de modo a generalizar bem os casos, acredita-se que possa ter ocorrido overfitting, e pelo desempenho ainda baixo mas indicando a possibilidade de classificação no caso de poucos *frames* tempo-

rais utilizando vídeos ao invés das imagens estáticas (acurácia, precisão e sensibilidade da ordem de 53.29%, 52.42%, 69.24%) como no exemplo com redução da quantidade e dimensão do *frames* para 3 e 60x80 pixels respesctivamente.

Por outro lado, há indícios de que a quantidade de exemplos disponíveis na base de dados utilizada não é suficiente para o treinamento de uma rede CNN (ainda que comparativamente pequena) para resolver o problema de classificação de configurações de mão a partir de vários *frames*. De fato, mesmo um aumento de dados da ordem de 152 vezes não permitiu obter resultados de classificação próximos aos obtidos com a classificação de imagens estáticas.

Palavras-chave: Configurações de mão, LIBRAS, Aprendizado de Máquina, Sequência Temporal de Imagens, Subamostragem e eescalonamento temporal.

Abstract

The area of Artificial Intelligence (AI) had its first paradigms presented in the 50s and 60s, this new area was presented as a powerful tool in the search to optimize pattern identification processes and a field with varied applications, the field had an expansion in the late 90s and early 2000s with the creation of new methodologies that solved preexisting gaps in the area, and its rise in the 2010s with a wide participation of the academic, scientific, curious and technology sympathetic community.

The search for pattern identification is the nature of the proposed work, the Brazilian Sign Language is the second official language of Brazil, and is the main form of communication for deaf-mute people or people with some degree of hearing impairment who have been literate in a bilingual school, even though it is official, is not widespread in all society, therefore, a communicative barrier exists between the gesticulating community and the non-gesticulating community that make up the majority of the population.

The machine learning area has numerous classifiers that address the most diverse problems, CNN and SVM are two classifiers with a wide knowledge of the scientific community for pattern recognition. In this work, tests were carried out to certify CNN as the most suitable classifier for problem.

The database used is composed of videos with the 61 hand configurations that make up LIBRAS, the videos were pre-processed and submitted to the application of data augmentation algorithms, subsampling of the configurations given the distinct characteristics, and temporal rescheduling of the amount of frames, so that machine learning would generate a classifier with good generalizability.

The videos resulting from the pre-processed base were converted into a binary data file to suit the available computational resource, the determination of the number of layers of the classifier and the values of the hyperparameters, was carried out by tests and adaptation to the system, since the application is specific and no base values were found in the bibliographic study.

The results of the metrics obtained in the training conducted suggest that the proposed approach may come to solve the problem of automatic classification of hand configurations in the future, by: (i) classification with good performance (accuracy, precision, sensitivity of the order of 100%, 100 %, 100%) in the case of static images even though the hand configurations are different, and the classifier had been trained in order to generalize the cases well, it is believed that overfitting may have occurred, and due to the

still low performance, but indicating the possibility of classification in the case of a few temporal frames using videos instead of static images (accuracy, precision and sensitivity of the order of 53.29%, 52.42%, 69.24%) as in the example in which the number and size of the frames were reduced to 3 and 60x80 pixels respectively.

On the other hand, there are indications that the number of examples available in the database used is not sufficient for training a CNN network (even though comparatively smaller) to solve the problem of classifying hand configurations from several frames. In fact, even an increase in data of the order of 152 times did not allow to obtain classification results close to those obtained with the classification of still images.

Keywords: Hand configurations, LIBRAS, Machine Learning, Image Time Sequence, Subsampling and time Reschedule.

Sumário

Lista de Tabelas

Lista de Figuras

Lista de Símbolos, Nomenclaturas e Abreviações

LIBRAS	–	Língua Brasileira de Sinais
PCD	–	Pessoas com Deficiência
IBGE	–	Instituto Brasileiro de Geografia e Estatística
PNS	–	Pesquisa Nacional de Saúde
SUS	–	Sistema Único de Saúde
INES	–	Instituto Nacional de Surdos-Mudos
CM	–	Configurações de Mão
IA	–	Inteligência Artificial
RNC	–	Rede Neural Convolucional
API	–	*Application Programming Interface*
RGBD	–	*Red,Green,Blue and Depht*
SVM	–	*Support Machine Vector*
CNN	–	*Convolutional Neural Network*
FCN	–	*Fully-Conected Convolutional Neural Network*
ML	–	*Machine Learning*

1 Introdução

1.1 Os desafios para facilitação da comunicação via LIBRAS

A LIBRAS (Lingua Brasileira de Sinais) é a segunda língua oficial do Brasil, oficializada pelo sancionamento da lei nº 10.436/2002. É o principal meio de comunicação de pessoas com tipos de deficiência auditiva que não permitem a utilização da lingua portuguesa [8], possuindo uma estrutura morfológica e sintática própria.

Engana-se quem pensa que a língua de sinais é uma tradução gestual da falada. É um sistema linguístico complexo que permite a interação da pessoa surda-muda com o mundo externo por meio de movimentos e posicionamento das mãos no espaço, e uso de expressões faciais. Possui regionalismo e vários aspectos de uma linguagem. o que lhe concede a natureza de lingua natural [9].

Porém a inserção da pessoa com deficiência auditiva nos espaços de convívio social como escola, mercado de trabalho, acesso aos serviços saúde, é dificultada pela comunicação entre quem utiliza a língua portuguesa e que não sabe gesticular LIBRAS, representando a maior parte da população, e as pessoas com deficiência, que conseguem assimiliar apenas 20% do que é expresso por meio de leitura labial, quando possuem essa habilidade [9].

O reconhecimento de sinais compostos, que é o caso da LIBRAS e suas estruturas sematosemas e morfemas que serão abordados no próximo capítulo, está relacionado ao processamento de linguagem natural, o campo de aprendizagem de máquina mostra-se uma ferramenta científica capaz de lidar com a complexidade deste problema visto que, segundo [4], possui algoritmos capazes de reconhecer padrões de sinais complexos multidimensionais e não-lineares.

Neste trabalho a busca pelo desenvolvimento de um sistema capaz de romper a barreira de comunicação entre um sujeito surdo-mudo e um apenas falante, com uma abordagem tecnológica, é incentivado pela leitura da obra *Homens, Engenharias e rumos sociais* em que Gilberto Freyre [10] trás uma reflexão sobre o campo das engenharias e a

contribuição social e humana que esta área pode implicar.

1.2 Trabalhos Correlatos

Diante da problemática do reconhecimento da LIBRAS, [11] utilizou uma abordagem de tradução automática por meio de visão computacional, para indentificar os números de 0 a 8 usando extração de características geométricas da mão, e classificação pelo algoritmo SVM [11].

Giordano Bruno [8], criou um banco de imagens usando a *Application Programming Interface* (API) do sensor *Kinect*, para obtenção de imagens das configurações de mão usando marcadores visuais em luvas, e estratégia de tratamento das imagens.

Porfírio [7], atua quanto ao reconhecimento da configuração por meio de uma base de dados adquiridas usando o sensor *Kinect*, e usando a componente de profundidade das imagens captadas dos vídeos, e extração de características para gerar malhas tridimensionais que classifique as 61 configurações de mão (CM).

1.3 Definição do Problema Científico acerca da classificação de sinais de LIBRAS

O desafio em criar um sistema automatizado que traduza LIBRAS é que, por se tratar de um problema de sinais compostos de uma lingua natural, envolve a necessidade da disponibilidade de uma grande base de dados para que o processo de aprendizagem de máquina possa gerar um resultado satisfatório.

A LIBRAS é a língua de sinais usada no Brasil, embora tenha suas raízes na francesa é única, logo, diferente de bases de dados de símbolos alfanuméricos, animais, objetos ou quaisquer conjunto de informações utilizados no aprendizado de máquina clássico, é restrito a informações da comunidade acadêmica e científica disponíveis.

As bases de dados disponíveis em mecanismos de busca são quase em sua maioria de produção do autor da pesquisa, logo, a métrica de desenvolvimento utilizada não segue um padrão, o que faz com que os sinais advindos de fontes diferentes não possuam homogeneidade de coleta, e consecutivamente acarreta ruídos na informação.

O uso do sensor *Kinect* é amplamente utilizado nos trabalhos que envolvem coleta de sinais de LIBRAS, por autores de trabalhos acadêmicos como, Oikonomodis, Kryiazis

e Argyros, Santos al [12], Chen, Yeo [11] devido a sua capacidade de coletar as imagens em formato R,G,B e profundidade, este último sendo uma alternativa de uso dos dados, porém dado o tempo dos vídeos gravados, distância da pessoa que está executando o sinal em relação ao sensor, e problemas técnicos e humanos associados as coletas, faz-se necessário o pré-processamento dos sinais para haver o menor erro possível no algoritmo de classificação relacionado à construção da base.

Para bases de dados em que o tamanho dos vídeos dos sinais de mão não sejam iguais, a quantidade de frames do vídeo também não será igual, logo, a sequência de imagens que formam um sinal terão dimensões diferentes uma das outras, e para LIBRAS que possui 61 configurações de mão, que são a base para a formação de palavras e expressões, acarretará um heterogeneidade acumulativa.

Esta dificuldade em buscar uma base de dados foi o incentivo dos trabalhos correlatos apresentados como [7], que obteve as imagens para compor a base, e teve o trabalho concentrado em desenvolver um mecanismo que reconheça as configurações de mão, [12] foi um pouco adiante e utilizou o conceito de sematosemas e morfemas para criar um classificador capaz de traduzir as formações de palavras da língua falada escolhidas para o estudo de caso.

Um estudo sobre reconhecimento de gestos dinâmicos de mão utilizando redes convolucionais, ou em inglês , Convolutional Networks (ConvNet), é apresentado em [13], em que os vídeos dos gestos são segmentados em um número fixo de *frames*.

Em uma abordagem a partir das sequências de imagens extraídas dos vídeos, a busca por um fator de escalonamento da janela temporal para fixar o número de imagens dos *framess*, embora possa acarretar uma alteração da resolução temporal, é uma alternativa de pré-processamento que é aplicada em alguns algoritmos de classificação e que será abordado neste trabalho.

1.4 Objetivos

1.4.1 Objetivo Geral

O objetivo deste trabalho é desenvolver e avaliar uma solução para classificação automatizada de configurações de mão que compõe a LIBRAS, por meio de técnicas avançadas de aprendizagem de máquina com subamostagem e reescalonamento temporal [13].

1.4.2 Objetivos Específicos

O desenvolvimento do sistema de classificação automatizado dos gestos de mão da linguagem brasileira de sinais, será implementado a partir das seguintes etapas intermediáras:

- Levantamento de bases de dados que contenham vídeos das configurações de mão em LIBRAS;
- Aplicação de algoritmo de extração dos *frames* dos vídeos e análise da sequência temporal das imagens;
- Busca de trabalhos correlatos sobre uso de algoritmos de aprendizagem de máquina para reconhecimento de sinais de LIBRAS;
- Determinação do algoritmo para executar a classificação;
- Classificação dos sinais e análise das métricas de avaliação de algoritmos;
- Determinar estudos de caso para realizar testes do modelo de predição;

2 Fundamentação teórica

2.1 Aprendizagem de Máquina

O uso de computadores para desenvolver soluções automatizadas de problemas relacionados ao reconhecimento de padrões em imagens, possui os primeiros indícios de desenvolvimento na década de 1950, pelo surgimento dos primeiros paradigmas de Inteligência Artifical (IA), que propunham simular o comportamento da capacidade cognitiva humana, a fim de desenvolver um sistema que apresente inteligência própria.

O reconhecimento de padrões por máquinas envolve 4 etapas fundamentais de acordo com [14]:

1. Aquisição dos dados que pretende-se utilizar para o treinamento do classificador.
2. Pré-processamento das informações, os dados coletados precisam passar por processos como redução, ampliação da dimensão dos dados que compõe o conjunto, tratamento de cores, mineração de casos que foram indevidamente coletados e causam ruídos às informações etc, de modo a homogeneizar a base que será usada no classificador.
3. A extração de características, processo de enfatizar apenas informações importantes em relação à base de dados, deve ocorrer de acordo com o modelo que deseja-se treinar, logo, o conhecimento acerca do trabalho desenvolvido já deve estar consolidado.
4. E a classificação, que é a etapa em que a partir da natureza das características da base de dados, e da escolha do classificador e parâmetros de treinamento, é gerado o modelo aprendiz.

2.2 Classificadores

Árvore de decisão, rede neural, máquina de vetor de suporte, K-vizinho mais próximo, Análise de discriminante linear são alguns dos mais conhecidos algoritmos de apredizagem responsáveis por gerar modelos, sejam eles de predição ou estruturação, dos dados de entrada [3].

Segundo Haykin três paradigmas podem ser utilizados na geração do classificador: supervisionado, não-supervisionado e por reforço, a escolha do paradigma está relacionado com a natureza dos dados e o modo como ocorrerá a aprendizagem [2].

Devido ao objetivo da aprendizagem ser a generalização de características de um conjunto de dados, é necessária atenção a certos mecanismos do processo para que o erro de predição seja o menor possível, para isso faz-se necessário que durante o processo o conjunto dos dados seja separado em dois subconjuntos, de treinamento e de teste, onde os rótulos das informações são conhecidos, isso é necessário para garantir que durante o processo de validação do algoritmo não ocorra sobreamostragem da taxa de erro de predição [3].

O processo de criar um modelo que permita a generalização de quaisquer dados de entrada, da mesma natureza do modelo gerado, ensinando o algoritmo a identificar esses dados, é o que o processo que foi chamado de *Machine Learning* ou *Aprendizagem de Máquina* efetivamente realiza.

2.2.1 emphSupport Vectors Machine (SVM's)

SVM é um classificador linear de aprendizagem para reconhecimento de padrões que utiliza uma abordagem geométrica. No qual x_i são os pontos dos dados de entrada representados em um espaço euclidiano $\mathbb{R}^n$, onde são representados por elementos positivos e negativos resultantes do aprendizado [3].

As características que pressupõem o uso da SVM são [2]:

- Boa capacidade de generalização: a SVM é capaz de gerar alta taxa de eficiência de predição em relação a dados que não fazem parte do conjunto de treinamento.
- Robustez em grandes dimensões: A SVM possui boas taxas em relação a objetos de grandes dimensões, enquanto outros classificadores tendem a ser menos eficientes pela ocorrência de *overfitting*.

- Base sólida: A teoria por trás da SVM é bem consolidada tanto na área de matemática quanto de estatística.

A separação dos dados do conjunto na SVM ocorre por meio de um hiperplano, e por isso é considerado um classificador linear de Equação 2.1,

$$w \cdot x + b = 0, \tag{2.1}$$

onde W e b são obtidos durante o treinamento, e representam a divisão do espaço da Figura 2.1 nas regiões positivas e negativas como indicado na Equação 2.2,

$$\begin{Bmatrix} Se & W \cdot X_i + b > 0 & y_i = +1 \\ Se & X \cdot X_i + b < 0 & y_i = -1 \end{Bmatrix} \tag{2.2}$$

que representam as classes, para $W \cdot X + b = 0$ y_i é indefinido [2].

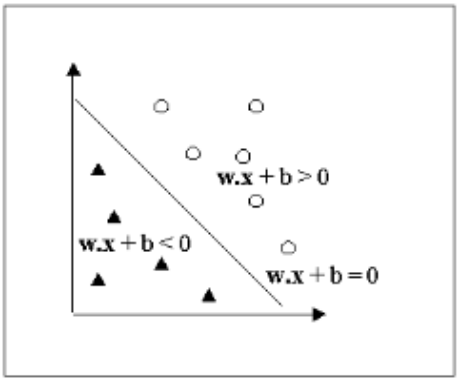


Figura 2.1. Conjunto de classes separado por um hiperplano com distribuição linear. Fonte: [2].

A distância euclidiana γ da Equação 2.3,

$$\gamma = \frac{|W \cdot X_i + b|}{||W||} \tag{2.3}$$

de um ponto X_i é sempre maior do que os hiperplanos $W \cdot X + b = 0$ e $|W \cdot X + b| = 1$ e por isso é chamada de margem rídiga [2].

Considerando a margem γ, a distância mínima entre os hiperplanos $W \cdot X + b = 0$ e $|W \cdot X + b| \geq 1$ é $||W||^{-1}$, logo, a distância entre os hiperplanos $W \cdot X + b = 1$ e $W \cdot X + b = -1$ é $2 \cdot ||W||^{-1}$ como pode ser evidenciado na Figura 2.2, e os pontos de X_i que estão localizados sob estes hiperplanos são os chamados vetores de suporte.

Determinar o hiperplano ótimo para separar as classes é um problema de otimização que leva em consideração os valores de w e b que satisfazem a margem rígida e em que a norma $||w||$ é mínima. Esta otimização pode ser determinada por meio da equação primária de otimização Equação 2.4,

$$(W^*, b^*) = argmin_{W,b,\xi i} + C \sum_{i=1}^{m} \xi_i \tag{2.4}$$

$$y_i(w \cdot X_i + b) \geq 1 - \xi_i \tag{2.5}$$

$$\xi_i \geq 0 \tag{2.6}$$

como apresentado por [3].

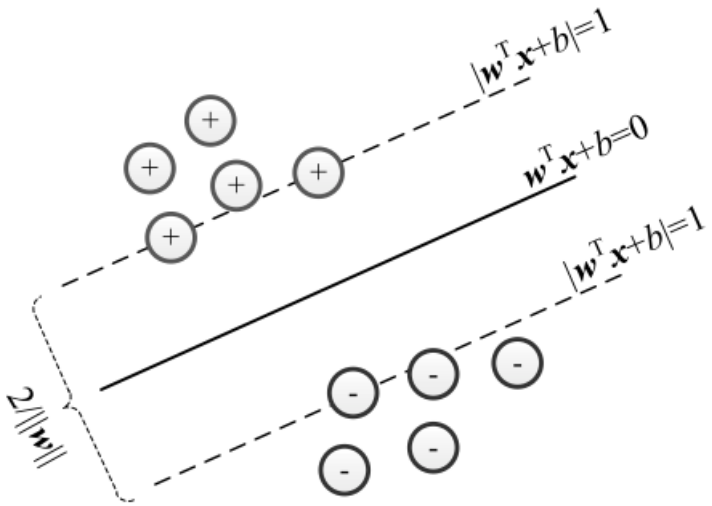


Figura 2.2. Representação do funcionamento de uma SVM. Fonte: [2].

Onde ξ_i é um fator para lidar com dados que não sejam perfeitamente separáveis, para descrever esse problema de otimização em relação aos pontos X_i a equação é dado pela forma dual Equação 2.7 [3].

$$\alpha^* = argmax_\alpha \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle \tag{2.7}$$

$$\sum_{i=1}^{m} \alpha_i y_i \tag{2.8}$$

$$\alpha_i \geq 0 \tag{2.9}$$

Da equação da forma dual é possível determinar um novo valor para o termo w^* da forma primária por meio da Equação 2.10,

$$w^* = \sum_{i=1}^{m} \alpha_i^* y_i x_i, \tag{2.10}$$

e com este novo valor é possível determinar o produto interno entre w^* e qualquer ponto x por meio da Equação 2.11 [3],

$$\langle w^*, x \rangle = \sum_{i=i}^{m} \alpha_i^* y_i \langle x_i, x \rangle \tag{2.11}$$

O problema deste tipo de classificador é que foi pensado para conjuntos de dados linearmente separáveis, o que no contexto dos dados utilizados no dia-a-dia são situações bem raras de ocorrer, para isso este tipo de classificador possui uma classe de funções que visa driblar esse problema, as funções *kernel*, que implementam uma transformação de espaço onde neste novo espaço, chamado de Espaço de Hilbert de reprodução de *Kernel* onde o conjunto de dados é mapeado para um novo espaço de maior dimensão e é representado pela Equação 2.12,

$$K(X_i, X_j) = \langle \phi(X_i), \phi(X_j) \rangle, \tag{2.12}$$

onde este novo formato equivale os produtos da forma dual da equação de otimização [3].

2.2.2 Redes Neurais Convolucionais (RNC's)

Redes Neurais, em inglês *Neural Network* são um subconjunto do campo de Inteligência Artificial (IA), o tipo de rede neural usado com mais frequência para solucionar problemas relacionados ao processamento de sinais multidimensionais é a *Convolutional Neural Network* (CNN's ou ConvNets), esta rede neural é a abordagem amplamente utilizada para tarefas complexas de reconhecimento de padrões de imagens.

O modelo cognitivo da unidade sensorial básica, *perceptrons*,da rede neural de inteligência artificial apresentado por F.Rosenblatt em *Principles of Neurodynamics: Perceptrons and the theory of the brain mechanisms* [15], indicou a prova matemática que as unidades convergem para uma solução no fim da camada que constituem, em um número finito de passos, por meio de pesos ajustáveis, quando treinado com conjuntos de dados que sejam linearmente separáveis.

O campo de aprendizagem de máquina viria à se expandir no fim da década de 80 e início dos 90's, quando a solução a restrição da capacidade de solucionar problemas práticos das redes neurais baseadas em *perceptrons* foi apresentada por Rummerlhart, por meio do algoritmo de *backpropagation*, que utilizou de mecanismos de retropropagação dos erros nos pesos das camadas intermediárias da rede neural para solucionar um problema de sinais 2-D em 1989, em um contexto do campo que seria conhecido como Redes Neurais Convolucionais Profundas [14].

O modelo de neurônio mais famoso é chamado modelo *McCulloch-Pitts*, apresentado na Figura 2.3, onde os padrões de características lineares de entrada x_i são multiplicados com pesos de conexão w_i, o sinal resultante é comparado com um valor limite denominado *bias* w_{n+1}, como mostrado na Equação 2.13,

$$z = \sum_{i=1}^{m} w_i x_i + w_{n+1} \tag{2.13}$$

se o novo sinal for maior que o limite, o neurônio é ativado e por meio de uma função de ativação a saída é gerada como um valor de ativação apresentada na Equação 2.14,

$$a_{x,y} = h(z_{x,y}) \tag{2.14}$$

e continua pela rede [3].

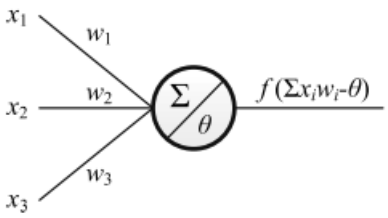


Figura 2.3. Modelo do neurônio de *McCulloch-Pitts*. Fonte: [3]

Redes neurais convolucionais (ConvNets), Figura 2.4, são construídas para lidar com dados em formato de múltiplos vetores, como vetores 2D da intensidade dos pixels dos canais de cores de imagens coloridas, dados podem vir em variados formatos de vetores, dependendo se lidam com sinais e sequências, imagens ou vídeos [16].

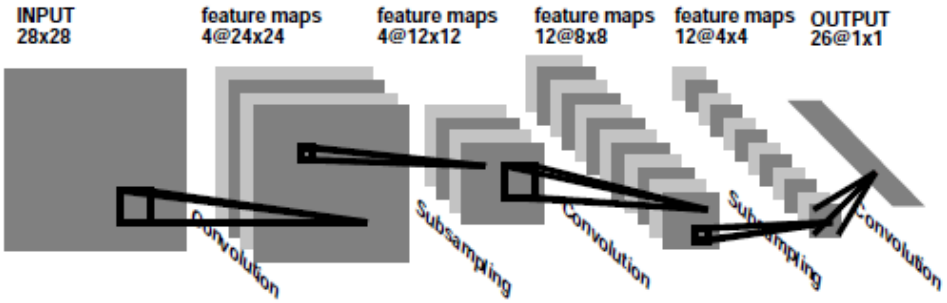


Figura 2.4. Exemplificação do funcionamento de uma rede neural convolucional com 4 camadas. Fonte: [4]

Há quatro idéias principais que regem o funcionamento das redes convolucionais segundo [16], essas ideias estão relacionadas às propriedades intrísecas de sinais: conexões locais, pesos compartilhados, agrupamento de dados por subamostragem espacial ou temporal e o uso de muitas camadas, e essas propriedades são a base da modelagem dos estágios da CNN, a combinação dessas ideias arquiteturais garantem certo grau de invariância de deslocamento e distorção [4].

A entrada da rede para reconhecimento de imagens recebe dados que costumam já estar normalizados e centrados [4]. As primeiras camadas comuns na arquitetura da maior parte das ConvNets são a convolucional, as unidades que a compõe estão organizadas em mapas de características, e a camada de agrupamento, cada uma das unidades da camada convolucional estão conectadas em um arranjo as unidades da camada anterior por meio de um conjunto de pesos denominado banco de filtros [16].

Essa soma ponderada entre as unidades das camadas é passado por uma não-linearidade

como as funções *Rectified Linear Unity*(ReLU), Tangente hiperbólica e Sigmoide, dadas pelas Equações (2.15), (2.16) e (2.17) respectivamente.

$$f(z) = max(0, z) \tag{2.15}$$

$$f(z) = tanh(z) \tag{2.16}$$

$$h(z) = \frac{1}{1 + e^{-z}}, \tag{2.17}$$

todas as unidades que compõe um mapa de características passam pelo mesmo banco de filtros, e os outros mapas de características que compõe uma camada possuem bancos de filtros diferentes, essa arquitetura é justificada pelas características dos dados, em vetores de dados como imagens, valores locais pela camada tendem a ser altamente correlacionados formando assim conteúdos locais facilmente detectáveis, e os resultados locais de imagens são invariantes a localização, logo, se um conteúdo aparece em uma parte da imagem pode aparecer em qualquer local, devido a essa característica ocorre a ideia de unidades em diferentes localizações compartilhando os mesmos pesos e detectando o mesmo padrão em diferentes partes do vetor [16].

Matematicamente a operação de filtragem em um mapa de características é uma convolução discreta, por isso é denominada camada convolucional, mesmo que o papel de uma camada convolucional seja identificar condições locais de cacaracterísticas das camadas anteriores, a função da camada de agrupamento é fundir características semânticamente semelhantes em apenas uma [16].

Uma unidade de agrupamento típica, avalia o máximo de um conteúdo local de unidades em um mapa de características, o agrupamento das unidades são formadas pelo deslocamento em uma linha ou coluna em relação à anterior, assim ocasionando uma redução da representação, e criando uma característica de invariância à pequenos deslocamentos e distorção das unidades resultantes.

Os estágios de convolução, não-linearidade e agrupamento (quando necessário) são empilhados, e seguidos de mais camadas convolucionais e inteiramente conectadas, que

resultam em uma rede neural convolucional como a mostrada na Figura 2.4. Os gradientes de *Backpropagation* em rede neural convolucional permitem o treinamento dos valores dos pesos de conexão em todos os bancos de filtros [16].

ConvNets têm obtido grande sucesso na aplicação de segmentação e reconhecimento de objtos e regiões em imagens, umas vez que há muitos dados já rotulados sobre esses tópicos, imagens podem ser rotuladas a nível pixelar, aplicação muito utilizada em tecnologias de carros autônomos, e tem encontrada aplicações importantes no entendimento de linguagem natural e reconhecimento de fala [16].

Em 2012 em uma competição *ImageNet* o uso de redes neurais profundas marcou a consagração de ConvNets para aplicação em quase todo tipo de tarefa de reconhecimento e detecção, o uso eficientes de GPUs, ReLUs, *Dropout*, e técnicas de *Data Augmentation* fizeram com que uma base de dados com aproximadamente 1 milhão de imagens, de 1000 classes, obtivem-se resultados excelentes, com taxas de erro não obtidas até então para problemas tão grandes [16].

2.2.3 Métricas para avaliação do desempenho dos algoritmos de aprendizagem

O resultado do processo de treinamento é o modelo de classificação, em que os pesos e bias associados às informações obtidas do dos dados de treinamento da base, são utilizados para predizer a classe de dados externos aos que foram utilizados pelo processo, porém uma etapa necessária para avaliar a qualidade do processo são as métricas de avaliação dos algoritmos de classificação.

Em termos da métrica de avaliação, exatidão é a porcentagem dos dados da base de teste que são classificados corretamente, porém devido à problemas como *overfitting*, não é um critério suficiente para avaliar a qualidade de um algoritmo de classificação, outras métricas muito utilizadas são sensibilidade, precisão e F-Measure [17].

Para a aplicação destas métricas, 4 conceitos devem ser levados em consideração [17]:

- **Verdadeiro Positivo (VP)** representa os dados que são positivos, e que foram classificados como positivos.
- **Falso Positivo (FP)** representa os dados que são negativos, e que foram classificados como positivas, logo, foram classificadas incorretamente.
- **Verdadeiro Negativo (VN)** representa os dados que são negativos, e foram classificados como negativos.

- **Falso Negativo (FN)** representa os dados que são positivos, e foram classificados como negativos logo, foram classificadas incorretamente.

As métricas são avaliadas em relação a estes conceitos, exatidão apresentada na Equação 2.18,

$$E = \frac{VP + VN}{VP + VN + FP + FN}, \tag{2.18}$$

Precisão é o percentual dos dados da base de teste que foi corretamente classificadas como positivos em relação a todos os dados que foram classificados como positivos apresentada na Equação 2.19,

$$P = \frac{VP}{VP + FP}, \tag{2.19}$$

Sensibilidade é o percentual dos dados da base de teste que foi corretamente classificadas como positivos em relação a todos os dados que são verdadeiramente positivos apresentada na Equação 2.20,

$$S = \frac{VP}{VP + FN} \tag{2.20}$$

E f-measure derivada da medida de de eficiência de Rijsbergen(1979), é a média harmônica ponderada entre a precisão e a sensibilidade apresentada na Equação 2.21 [**?**], [17],

$$F1 = 2 \cdot \frac{precisao \cdot sensibilidade}{precisao + sensibilidade} \tag{2.21}$$

2.3 Linguagem python e problemas de classificação em *Machine Learning*

2.3.1 Python e Machine Learning

Aplicações computacionais científicas como *Machine Learning* costumam utilizar operações de álgebra linear em matrizes multidimensionais devido a estrutura dos dados computacionais, que são representados por vetores, matrizes, tensores. A linguagem python tem uma biblioteca de rotinas de operação de álgebra linear chamada *NumPy* [18].

Mesmo com *deep learning* aumenta sua popularidade ao longo dos últimos 10 anos, as aplicações com aprendizagem de máquina (AM) clássico ainda são bastante comuns no meio acadêmico e industrial, os classificadores clássicos de AM foram pensados para dados estruturados, enquanto *deep learning* mostra-se uma alternativa mais adequada para casos de grandes bases de dados não-estruturados, para a aplicação clássica de machine learning, python possui uma biblioteca de grande popularidade para modelar conjuntos de dados de dimensões medianas (entre 1000 - 100.000 exemplos), *Scikit-learn*, que é pioneira no uso de modelo de API *"fit()\predict()"* devido a grande colaboração da comunidade aberta [18].

Desenvolvido pelo Google em 2016, *Tensorflow* é um *framework* que permite a definição de redes convolucionais profundas aplicadas diretamente em relação ao tempo de execução do programa por meio de uma API de fácil implementação, permite aplicar otimização de código, exportação do modelo de classificação o que torna uma ferramenta atraente para uso [18].

PyTorch(2017) é outro tipo de *framework* para aplicação de rede neural profunda, é uma aplicação que permite interação direta com o computador, ao invés de definir um grafo estático, utiliza um estilo de programação imperativa [18].

As bibliotecas de programação de redes profundas usam a linguagem de modo direto, para permitir o maior acesso a praticantes de programação são usadas *Application Programming Interface* (API's) que tem como função melhorar a experiência entre a ação e personalização do código, porém ainda assim podem ter o uso confuso para novos praticantes, para isso existem outros tipos de ferramentas que é as bibliotecas *wrapper* que abstraem a complexidade do funcionamento das API's, um *wrapper* famoso é o *keras* que oferece suporte ao *Tensorflow*, na versão 2.0 do *Tensorflow* em 2019 foi implementado pelo submódulo *tensorflow.keras* como API de usuário oficial da biblioteca [18].

2.4 *Deficiência Auditiva*

Pessoas com deficiência (PcD), nomeclatura indicada pela Portaria da Presidência da República – Secretaria de Direitos Humanos, Nº 2.344, de 3 de novembro de 2010, vêm sendo tratadas de formas diferente ao longo da história, da época dos gregos e romanos, onde habilidades físicas e intelectuais eram veneradas, seja a crença de que as deficiências são punições, ou sinal de semidivindade,a estrutura da sociedade tende a tratá-las de modo marginalizado, sendo inabilitadas e vistas como objeto de caridade perante a comunidade e a própria família, o que dificulta a inserção destas pessoas na sociedade, e cria impactos no acesso aos direitos básicos, como educação,saúde e mercado de trabalho [9],[19].

A surdez é caracterizada como a redução ou ausência da capacidade de ouvir determinados sons, uma forma de indicar este fenômeno é pela caracteração dos tipos e causas, os tipos de surdez segundo o [20] são:

- Ligeira, no qual a pessoa não possui atrasos no desenvolvimento e obtenção da linguagem, apenas alguns elementos fonéticos são despercebidos ocasionando dificuldade em ouvir uma conversa;
- Média, neste, ocorre dificuldade em adquirir a linguagem e distúbio na articulação de palavras, há necessidade de meios visuais, como leitural labial para entendimento;
- Severa, na qual a aquisição de linguagem e articulação de palavras é comprometido e conversas em tons normais são quase despercebidos sem uso intenso de elementos visuais;
- Profunda, na qual não há sensibilidade auditiva, dificuldade extrema na aquisição da linguagem oral, e articulação das palavras tornando necessário o uso de linguagem gestual;
- Cofose, na qual há ausência completa de percepção do som, e impossibilidade de articulação das palavras e aquisição de linguagem oral.

As causas podem ser classificada em duas [21], [20]:

- Perda auditiva condutiva, ocorre por bloqueio da orelha externa, Figura 2.5), causada pelo acúmulo de cera de ouvida, infecções (otite) de repetição, imobilização de um ou mais ossos que constituem o ouvido, rompimento do tímpano, etc. Logo, dificulta a passagem das ondas sonoras pelos canais que compõe o sistema auditivo e o estímulo elétrico necessário para interpretação destes sinais pelo cérebro.

- Perda auditiva neurossensorial, que compreende danos nas células ciliadas da cóclea, ou nervo auditivo, Figura 2.5, pode ser desencadeada por rubéola, sífilis ou toxoplasmose gestacional, predispoção ou herança genética, meningites, presbiacusia, etc. Logo, causa danos diretos a parte do sistema auditivo responsável por gerar os estímulos elétricos.

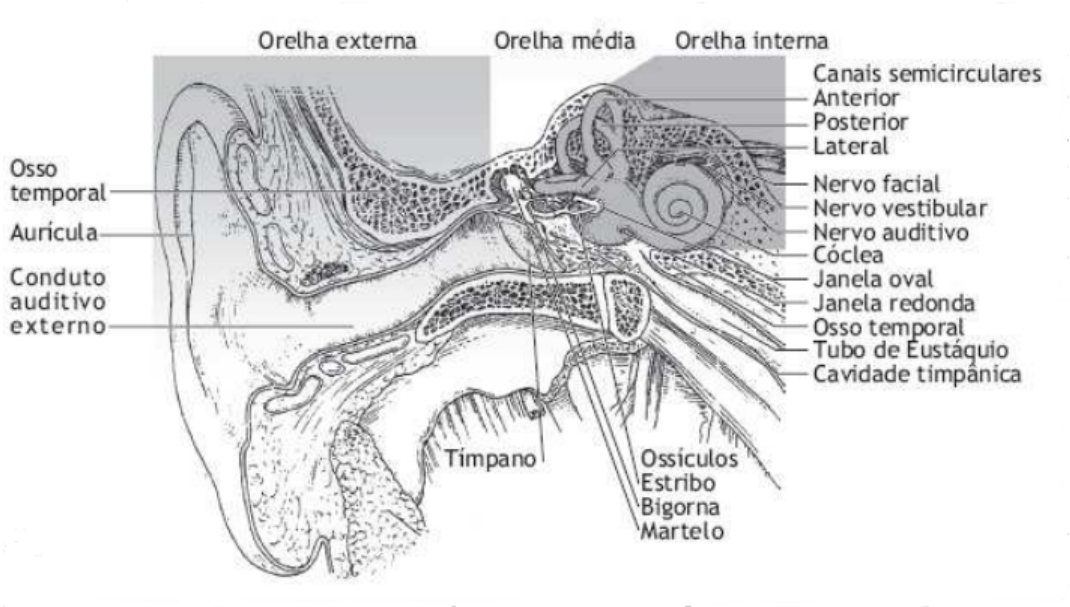


Figura 2.5. Anatomia da orelha humana. Fonte: [5].

O Instituto Brasileiro de Geografia e Estatística (IBGE), desenvolve uma pesquisa que visa coletar dados e desenlvover uma pesquisa em relação à saúde, a última edição da Pesquisa Nacional de Saúde (PNS), realizada em 2013 e divulgada em 2015, apresentou no volume temático: Ciclos de vida, o plano de amostragem e indicadores de saúde específico para alguns grupos que compõe a população brasileira, com destaque para Pessoas com Deficiência. A PNS estimou que 6,2% da população, de 200,6 milhões de pessoas residentes em domicílios particulares permanentes em 2013, possuem algum tipo das 4 tipos de deficiência: intelectual, física, auditiva e visual.

A PNS classificou a pessoa com deficiência auditiva, como aquela que tinha surdez nos dois ouvidos,surdez em um ouvido e audição reduzida no outro, ou ainda audição reduzida de ambos os ouvidos, para coleta dos dados, e estimou-se que representam 1,1% da população [6].

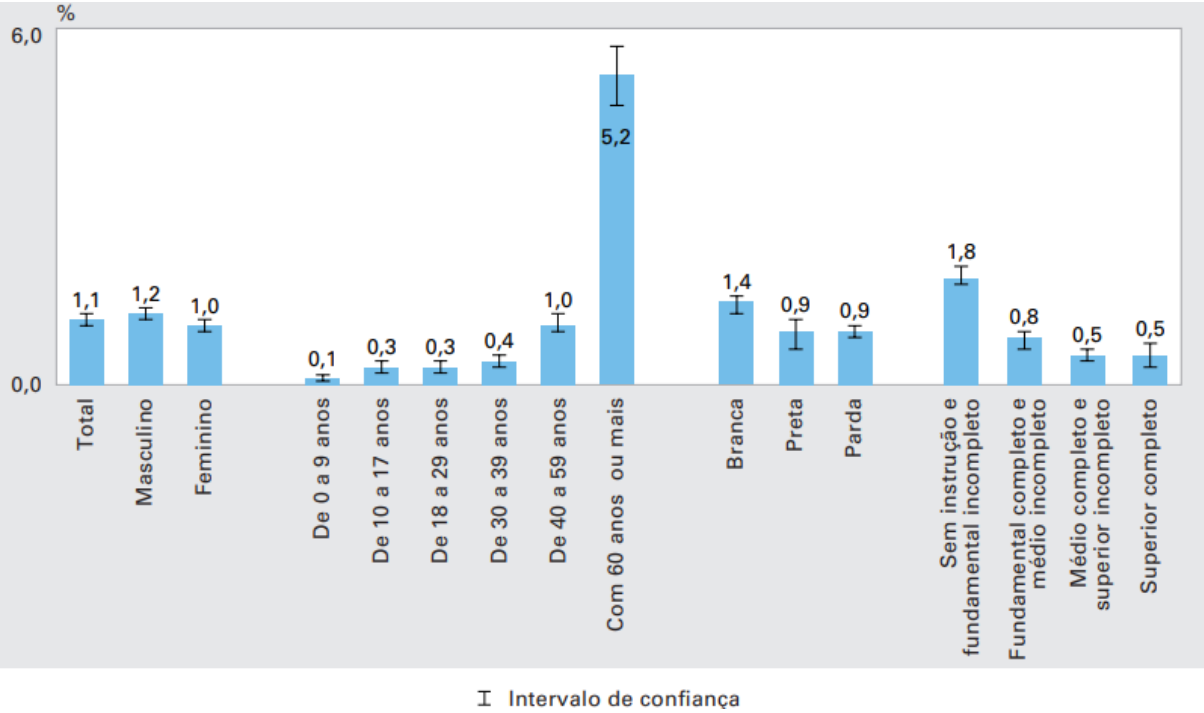


Figura 2.6. Gráfico da proporção de pessoas com deficiência auditiva, na população total, com indicação do intervalo de confiança de 95%, segundo o sexo, os grupos de idade, a cor, ou raça e o nível de instrução - Brasil - 2013. Fonte: [6].

2.4.1 Surdez e Saúde

A comunidade surda representa uma minoria sociolinguística e cultural em relação a comunidade que utiliza linguagem oral, logo o processo de comunicação entre estes indivíduos passa por barreiras, adversidade de integração que promove desafios no atendimento nas unidades de saúde para o sujeito surdo e aliado ao déficit de humanização na relação profissional da saúde - paciente deficiente auditivo, [22], onde as diferenças lexicais e semânticas entre a língua portuguesa, usada por pessoas com capacidade de articulação de palavras e capacidade de aquisição de linguagem oral, e a LIBRAS, linguagem visual-gestual, primeira língua do sujeito surdo, tornam-se um obstáculo para ambos os lados uma vez que a abordagem deve ser feita com termos expressões mais simples e de fácil compreensão na conversão da lingua falada para a língua oral.

A acessibilidade para pacientes surdos ou com deficiência auditiva ao atendimento no Sistema Único de Saúde (SUS) é garantido pelo Decreto Lei nº 5.626 de 2005, porém ainda conta com problemas como falta de Aparelhos de Amplificação Sonora Individuais (AASI), falta de capacitação dos profissionais de saúde para este tipo de atendimento, adaptação escassa para promover o acolhimento, que é estabelecido por meio das relações solidárias e com o ambiente [22],[23], ocasionando busca por serviços de saúde menos

frequentes entre pacientes surdos e deficientes físicos em relação a pacientes ouvintes.

Um estudo realizado entre Março de 2011 e Julho de 2012 pelo Grupo de Estudos e Pesquisa da Universidade Estadual da Paraíba [1] aborda o impacto da surdez na periodicidade de busca a serviços de saúde. O estudo foi composto por 36 pessoas surdas e mostrou que 44.4% dos pacientes não procuram serviços de saúde por não ter companhia de uma pessoa ouvinte gesticulante, Tabela 2.1,

Tabela 2.1. Frequência de acesso aos serviços de saúdes pelos participantes da pesquisa. Fonte: [1].

Variáveis de Acesso		Número	%
Acesso ao serviço médico	Sim	27	75,0
	Não	9	25,0
Acesso ao serviço dentário	Sim	32	88,9
	Não	4	11,1
Tipo de serviço acessado	Serviço Público	17	53,1
	Plano de Saúde	8	25,0
	Particular	7	21,8
	Não	4	11,1
Serviços usados (mais de 2 vezes no ano)	Consulta Médica	9	25,0
	Atendimento Dentário	8	22,2
	Exame Clínico	5	13,8
	Fisioterapia	1	2,7
	Hospitalização	1	2,7
Por que não acessar	Sem necessidade	8	88,8
	Sem acompanhante	4	44,4
	Dificuldade Geográfica	3	33,3
	Dificuldade de locomoção	2	22,2
	Dificuldade de transporte	2	22,2
	Problemas Financeiros	2	22,2
Faz uso de medicamentos	Sim	12	33,3
	Não	23	63,8
Dificuldade de acesso aos medicamentos	Sim	8	66,6
	Não	4	33,3

e que 100% dos pacientes encontram dificuldades em comunicação com os profissionais , Tabela 2.2,

Tabela 2.2. Indicação dos fatores de comunicação no acesso aos serviços de saúde. Fonte: [1].

Variáveis de Comunicação		Número	%
Dificuldade de Comunicação	Sim	36	100,0
Forma de Expressão	Assistência de Familiar	31	86.1
	Escrita	10	27.7
	Mímica	9	25,0
	Leitura Labial	8	22,2
	Assistência de Intérprete	5	13,8
	Imagens	2	5,5
	Desenho	1	2,7
	Libras	1	2,7

o que vai de encontro com o levantamento de [22] que por meio de levantamento de artigos utilizando descritores indexados envolvendo temas relacionados a: surdez, pessoas com deficiência auditiva, saúde, e serviços de saúde, determinou que 100% dos artigos abordam a barreira comunicacional entre paciente surdo e profissional de saúde.

2.4.2 Surdez e Educação

Pessoas surdas fazem parte da constituição da sociedade há séculos, porém sua integração à parte majoritária da população, que é ouvinte e capaz de oralizar a comunicação, encontrou barreiras de ambas as partes, fazendo com que a população surda-muda se segregasse a uma comunidade com língua, cultura e identidade próprias [24].

O problema da comunicação entre ouvintes e falantes e não-ouvintes e não-falantes proporcionou a busca de soluções, que tem as primeiras evidências registradas datadas do século XVI, na Europa, marcada pelo desenvolvimento do manual: A escrita e a oralização, de Ponce de Leon (1520-1584), que constituiu a base para que outros países interessados na educação dos surdos criassem seus próprios manuais [24], [25].

No Século XVII e XVIII com avanços no desenvolvimento da educação para surdos, temos o francês L'Epée e a fundação do Instituto Nacional de Surdos-mudos (1750), ocorrendo paralelamente o desenvolvimento de material que apresenta a língua de sinais como uma língua universal, e capaz de expressar a comunicação igualmente a língua falada para ouvintes, e o desenvolvimento de metologias pedagógicas pelos jesuítas afim de unificar a coletânea que usavam para instruir os jesuítas docentes [25].

Durante essa era inicial do desenvolvimento de metodologias para ensino de surdos, houve divergências, que segundo Massone foram ocasionadas pelos dois diferentes modelos de compreensão sobre a surdez, o modelo clínico-terapêutico, que leva em consideração a surdez em relação ao distúrbio da deficiência auditiva, com a proposta do ensino oralista de Heinick, que foi recusado por parte da comunidade cientifica da época, e o que os sinais de mão propostos por L'Epée segue, modelo socioantropológico, que acredita na comunicação gestual ao invés da oralização [24], e permitiu que a Lingua de Sinais Francesa fosse difundida em parte da Europa e mundo, sendo a base das linguas de sinais americana e brasileira [25].

A história da educação dos surdos no Brasil tem como principal personagem Dom Pedro II, e com o ano marcante para traçar a evolução histórica do desenvolvimento, 1857, com a fundação do Imperial Instituto de Surdos-Mudos, atualmente denominado Instituto Nacional de Surdos-Mudos (INES), uma escola para surdos inaugurada pelo professor surdo francês Ernest Huet, que chegou em 1855 a convite do monarca [24].

Em 1960, Dr.Stokoe relata pela primeira vez a percepção de que a língua de sinais possui aspectos linguísticos, tais como variações linguísticas, iconicidade e arbitrariedade, características que indicam uma língua natural e completa, logo não dependem, nem descendem da língua oral [24].

A apresentação de regionalidade na expressão linguistica-visual marca também uma característica derivada do movimento surdo-mudo, que diante o histórico de preconceito e marginalização social buscaram ocupar espaços comuns para reunir pessoas surdas que compartilham dos mesmos interesses, costumes, histórias [24].

O reconhecimento das linguas de sinais como não universais, e sim como entes próprios, com características que mudavam de país para país, fomentou a oficialização das mesmas, no Brasil, a lingua de sinais francesa lecionada por Huet misturou-se com aspectos da lingua portuguesa, logo, reafirmando o *status* de língua natural [24], [25].

Durante várias décadas no Brasil a luta dos educadores em relação à escola inclusiva e democrática, acrescido no início da década de 80 do uso da Comunicação Total, filosofia e movimento americano de inclusão educacional, base da formulação do modelo socioantropológico da surdez, resultou na criação de escolas especiais, onde alunos surdos e professores ouvintes iniciaram a introdução dos aspectos visuais já existentes na língua falada para a língua de sinais [24].

A constituição brasileira de 1988 trás artigos que citam os direitos das pessoas surdas, fomentando a oficialização das línguas de sinais, que no Brasil ocorreu em 2002 pela lei 10.436, que reconhece a LIBRAS como a primeira língua oficial do sujeito surdo-mudo

e a língua portuguesa na modalidade escrita como segunda língua, juntamente com a resolução Nº 5.626 de 2005 que regulamentou a lei e discorre sobre a difusão da Libras em todos os meios escolares, e tornou obrigatória a disciplina em todos os cursos de licenciatura e Fonoaudiologia [25], e garante o direito ao ensino de Libras para pessoas surdas-mudas.

A oficialização de LIBRAS como língua oficial culminou no desenvolvimento de um curso próprio para a área, que atende a necessidade do estudo mais aprofundado de uma língua, assim como todas as línguas necessitam, com a criação do curso de Letras/Libras(2006), garante o direito da pessoa surda-muda ao acesso à saúde e outras atividades governamentais por meio da obrigatoriedade e oficialização profissional de intérpretes, assim como tradutores, pela lei Nº 12.319 de 2010 [25].

O modelo oralista de Heinick baseado no entendimento clínico do problema da surdez, implementou um sistema oralista no congresso de milão em 1880, em que os países que adotaram o sistema proibiram o uso da lingua de sinais para as pessoas surdas, provado futuramente que a lingua gestual-visual do modelo socioantropológico conseguia atender melhor as necessidades de aprendizagem, por apresentar a surdez como uma diferença linguística de uma lingua natural, logo, entende que aspectos como convivência social podem promover alterações, crianças nascidas surdas-mudas, filhas de pais também surdos-mudos, com o modelo da língua de sinais conseguem apresentar um melhor desenvolvimento [25].

Com um olhar atual, a surdez deve ser entendida como uma patologia, e uma marca linguística de aproximação do sujeito surdo-mudo com a sociedade ouvinte, cujo não possui muitos espaços de convívio que permitem uma integração, logo, em relação a educação inclusiva ou bilingue, em que aprende-se a lingua portuguesa oral e escrita e a língua brasileira de sinais, deve-se levar em consideração a proposta de políticas públicas que vão permitir a maior inserção de surdos-mudos em espaços de convívio sociais e reduzir o preconceito contra essa comunidade [24].

A escolarização de pessoas com deficiência foi abordado por um documento aprovado em uma conferência mundial de educação especial, em 1994 na Espanha, sobre políticas e práticas educacionais para pessoa com deficiência, chamada Declaração de Salamanca, que propôs um novo olhar sobre a educação especial, orientando à nível nacional e internacional, de organização e políticas educacionais que leva em consideração a mudança da estrutura curricular, a profissionalização dos educadores, e indica grupos prioritários ao acesso dessa educação, que seriam crianças e jovens, com inserção da educação infantil inclusiva até o ensino básico [24].

Essa inclusão incluia políticas de transição escola-trabalho, em que as demandas de profissionalização, sociais e de comunicação que atendam às expectativas da vida adulta, a educação de adultos permite que essa parcela da população que não teve acesso a educação básica inclusiva possa se profissionalizar e inserir no mercado de trabalho, de modo a se tornarem economicamente ativos, e educação de meninas, uma vêz que a deficiência soma-se as desvantagens pré-existentes na vida de uma mulher [26].

A educação inclusiva é uma pauta que confronta-se com princípios e problemáticas das políticas públicas do ensino regular, que antecedem a introdução do ensino especial como aponta Angelucci(2002), que é o aspecto do acesso universal a educação básica, no Brasil apresentou-se um critério excludente, uma vez que a escola para todos não estava atendendo essa máxima democrática, independente dos alunos possuirem ou não alguma deficiência, o Plano Nacional de Educação (2014) que possui em sua meta 4 a introdução do ensino bilíngue na educação, pode ser um subsidio para que ocorra maior atenção do ministério da educação em melhorar as práticas de inclusão em escolas que possuam LIBRAS no currículo, o uso de ferramentas tecnológicas podem e devem ser icentivadas para fomentar este processo.

2.4.3 Surdez e Mercado de Trabalho

Abraham Maslow (1908-1970), conhecido pela fundamentação da Teoria das Necessidades, por meio do qual baseado em observações empíricas dado sua função de psicólogo, apresentou a proposta de divisão dos cinco níveis de fatores de satisfação do ser humano, em formato pirâmide, da necessidade de nível mais baixo (base) até a de nível alto (topo), leva em consideração que a necessidade, impulso e estados motivadores são concepções que implicam na motivação, no qual a necessidade é a restrição da alegria [27].

As necessidades segundo Maslow são definidas nas categorias fisiológica (base da pirêmide), segurança, social, estima e auto-realização (topo da pirâmide), e o ato de trabalhar pode ser avaliado de forma distinta levando em conta todos os níveis de necessidade, da base para o topo da pirâmide, no que tange ao espectro das necessidades [27].

As necessidades são traduzidas em busca de boas condições de trabalho, ordem e previsibilidade, dado pela busca de benefícios, atendimento de normas de segurança e padrões de desempenho, integração social por meio de relacionamentos interpessoais, busca de reconhecimento com base em mérito dada a competência que resulte em oportunidades de crescimento [28].

O "assumir responsabilidades"por Zarifian é um termo comparativo para definir com-

petência, dentro do contexto do mercado de trabalho, retrata a qualificação do sujeito devido à natureza do seu trabalho [29].

No que concerne a discussão sobre a surdez, os aspectos sociais e psicológicos da necessidade do trabalho, a discussão sobre o desenvolvimento da competência do sujeito surdo é abordado por [29] em relação as virtudes esperadas que o sistema educacional e questões psicológicas de formação proporcionem para que seja desenvolvido as virtudes intelectuais esperadas.

A integração, que na constituição de salamanca foi substituído por inclusão, de pessoas com deficiência é um tópico que procura normalizar as condições de vida das pessoas deficientes, é um equacionamento de problemas, busca de soluções e concretizar a equidade das oportunidades para todos de modo que haja participação das pessoas deficientes, e da socidade em geral [29].

Completando 30 anos (2021), a Lei de cotas para pessoa com deficiência, 8.213/91, que obriga empresas que se adequam à proporção de empregados, a reservar vagas para pessoas com deficiência. A proporção incia-se com empresas que possuam a partir de 100 funcionários como pode ser visto na Tabela 2.3,

Tabela 2.3. Percentual de cotas para pessoas com deficiência de acordo com a Lei 8.213/91

I-até 200 empregados	2%
II-de 201 a 500	3%
III-de 501 a 1000	4%
IV-de 1001 em diante	5%

de acordo com a Relação Anual de Informações Sociais (Rais), em 2018, o mercado de trabalho formal contabilizou a existência 456,7 mil vagas ocupadas por pessoas com defiência reabilitadas (termo usado para descrever que a natureza das vagas é referente às ações afirmativas) no Brasil [30]. Em 2017, mais de 79 mil trabalhadores surdos possuiam carteira assinada [31], dos quais a maioria das vagas foram preenchidas para auxiliar de escritório.

O direito ao trabalho está previsto no estatuto da pessoa com deficiência, instituído pela lei 13.146/2015. As associações legitimadas representativas da população surda buscam incentivar iniciativas de parceria com o setor produtivo para desenvolvimento de projetos de qualificação, assim como a luta pela politica de legalização e divulgação da LIBRAS [29].

Centros como a Federação Nacional de Educação e de Integração do Surdo (FENEIS), INES/RJ, associado ao governo federal, apoiam o ensino profissionalizante por

meio da Divisão de Qualificação e Encaminhamento Profissional (DIEPRO), dentro da área do trabalho autônomo, competitivo e em oficinas protegidas ou abrigadas, e estágio remunerado [29].

No Brasil, em 1989, foi iniciado o Programa de Emprego Apoiado(PEA), que em 1993 viria a ser nomeado Grupo de Emprego Apoiado(GEA) [29], que é uma iniciativa que busca oferecer apoio para obtenção de emprego para uma pessoa deficiente, independente da severidade da deficiência e possibilitando apoio individualizado.

> *Uma ação que se pretenda inclusiva deve atentar para o redimensionamento do tempo e do espaço de aprendizagem quando necessários; deve flexibilizar conteúdos, incluir valores culturais dos sujeitos; atentar para metodologias que auxiliem o domínio do papel profissional com o desenvolvimento das competências para o trabalho, tendo em vista as necessidades educativas especiais; implicar, nas estratégias pedagógicas, todos os atores, portadores ou não de alguma deficiência. Caso contrário, corre-se o risco de favorecer uma exclusão, ao incluir sem critérios, no Mundo do Trabalho, pessoas com deficiência; principalmente se esse trabalho é de natureza competitiva.* Fonte: [29],4º parágrafo, página:52.

O emprego apoiado em empresas é um processo sistemático que consiste em [32]:

- Perfil vocacional: descoberta dos pontos fortes, interesses e necessidades de apoio da pessoa.
- Desenvolvimento de emprego: Pesquisa de vagas de trabalho que encaixa-se no perfil vocacional.
- Acompanhamento pós-inclusão: Na qual o treinamento e inclusão social é verificado de modo a validar as estratégias empregadas estão sendo adequadas às necessidades da pessoa e da própria empresa.

O apoio pode ser realizado por meio de orientação, auxílio personalizado no treinamento, *feedbacks*,e assim como a pessoa com deficência, os outros funcionários devem passar por treinamento para que a inclusão possa efetivamente inserir o candidato naquele ambiente.

Empregar pessoas com deficiência no quadro de funcionários possibilita a transformação da imagem perante a sociedade devido a visão inclusiva sobre os princípios que regem a empresa [32].

2.5 LIBRAS

Na Lingua Brasileira de Sinais as relações gramaticais são especificadas através da manipulação espacial dos sinais, por ser uma lingua gestual-visual, o uso do espaço implica considerar o referencial, o mesmo sinal pode ser realizado em diferentes pontos de referencial em nível fonológico, implicando mudanças de significado no nível semântico, e indicar marcação de plural e flexão verbal em nível morfológico e estabelece relações gramaticais em nível sintático [33].

Os sinais da LIBRAS, sematosEmas, são a combinção dos elementos movimento das mãos e do braço, com um formato específico, posição espacial e movimentação, é constituída de 5 parâmetros, configurações de mão, ponto de articulação, movimentação, orientação e expressões corporais e faciais quando aplicável [8].

As linguas faladas possuem classificadores que apresentam a relação entre o significado, e função da palavra , além de ser possível combiná-los, de acordo com [33] em sete categorias de classificadores que costumam estar presentes na organização das línguas, são eles material, formato, consistência, tamanho, localização, arranjo e quantia.

Na LIBRAS os classificadores podem ser descritivos, especificadores, de plural, instrumental e de corpo [33], estes são apresentados por meio das combinações dos sematosEmas das 61 configurações de mão (CM) Figura 2.7 que descrevem o referente quanto à forma, tamanho, maneira como se comporta na ação e a classe que pertence.

Para pessoas que utilizam a língua portuguesa o conceito de sematosemas e morfemas podem ser confundidos com letra e palavra, respectivamente, porém o léxico da lingua de sinais mostra-se mais complexo uma vez que um sinal pode ser formado por dois morfemas como exemplifica [12] com o exemplo das palavras estudar e universidade, no qual estudar é um morfema composto por uma sequencia de sematoemas, e a palavra universidade é formada pela palavra estudar e um acréscimo de sematoemas.

Figura 2.7. Configurações de mão da LIBRAS. Fonte: [7].

3 Métodos para classificação de mão em LIBRAS

3.1 Informações dos recursos computacionais utilizados

Este trabalho foi realizado em um notebook da marca Dell Inspiron 5458, com processador com base x64 Intel(R) Core(TM) i5-5200U CPU e clock de 2.20GHz (4 núcleos), possuindo 8GB de memória RAM instalada, e sistema operacional Windows 10 Home Single Language de 64 bits. O disco local com capacidade de 464GB sendo 224GB livres para uso. A versão dos programas utilizados foram:

- Matlab R2018a v9.4.0.813654 win64
- Visual Studio Code v1.56.0 Windows_NT x64 10.0.19041

A versão das bibliotecas em Python 3.7.6 utilizadas são:

- Numpy v1.18.1
- Matplotlib v3.1.3
- opencv-python v4.2.0
- openni v2.3.0
- tensorflow v2.3.1
- walk 0.3.5
- keras v2.4.3
- path v13.1.0
- scipy 1.4.1
- argh v0.26.2

3.2 Experimento de implementação preliminar dos classificadores

Primeiramente foi realizado um experimento em linguagem de programação python, para análise das informações relacionadas aos classificadores mais indicados para lidar com problemas de reconhecimento de padrões da natureza da LIBRAS, que é o SVM e o CNN. Para isso foi proposto um exemplo simples, onde foram geradas duas classes de pontos por meio do programa do Apêndice 6.1,'+,red' e '.,blue', Figura 3.1, para analisar os resultados gerados por ambos métodos de classificação.

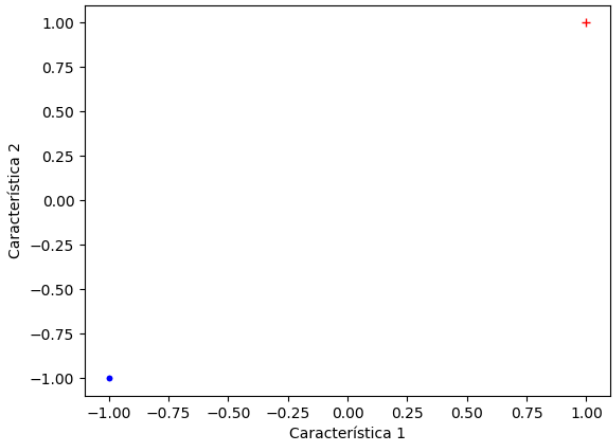


Figura 3.1. Geração de dois pontos que representam 2 classes distintas.

Para a classificação dos casos de distribuição linear dos pontos, Apêndice 6.2, que resultou na Figura 3.2,

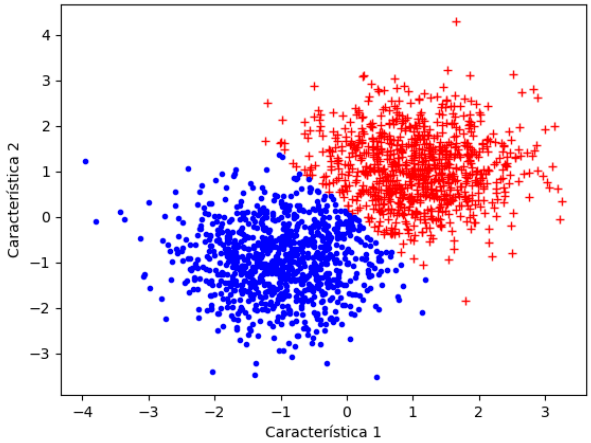


Figura 3.2. Conjunto de pontos das 2 classes com distribuição linear

e distribuição circular, Apêndice 6.3, que resultou na Figura 3.3,

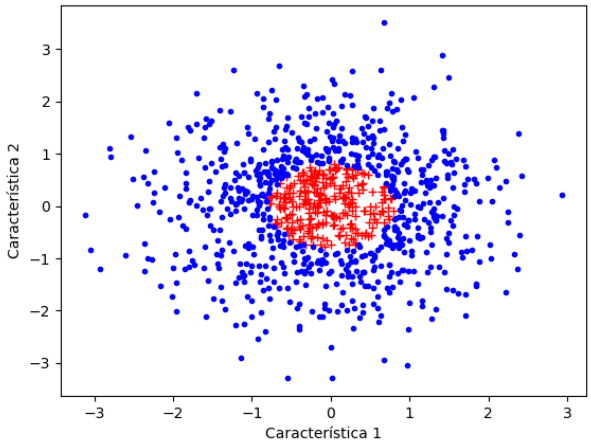


Figura 3.3. Conjunto de pontos das 2 classes com distribuição circular

foi utilizado um programa em que a chamada da *main* é indicado no Apêdice 6.7.

Foi gerado um conjunto de 1000 unidades para treinamento denominada M_training, 500 unidades para cada classe, e 400 unidades para teste denominada M_testing, 200 para cada classe, foi gerado uma nuvem onde foi indicado um espaço para cada classe por meio dos rótulos t_training, no caso do conjunto de treinamento e t_testing para o conjunto de teste, a distrbuição dos pontos ocorreu de modo randômico com uma taxa de afastamento associado a um desvio padrão.

No caso da SVM, apresentado no Apêndice 6.4 foi utilizada a classe SVC para gerar o modelo para classificação, é aplicado uma função kernel linear devido a natureza simples dos dados do exemplo, para realizar a predição a partir do conjunto de dados de teste, antes então, é preciso fazer o *fitting* do vetor de características e de rótulo do conjunto de dados que treina o modelo por meio de ajuste dos parâmetros, com o modelo treinado é possível predizer a resposta obtida da aplicação no conjunto de testes.

Para o classificador CNN apresentado no Apêndice 6.5 foi utilizado as bibliotecas *tensorflow*, *tensorflow.keras* e *tensorflow.keras.utils*, além do uso das classes *layers* e *models*, a primeira etapa é transformar a matrix do conjunto de dados de entrada em um vetor, tanto para os dados de treinamento quanto para os de teste e suas respectivos rótulos, usando a classe sequential do keras é criado as camadas do modelo de treinamento, a sequência usada para este exemplo foi:

1. Aplicação de filtro convolucional unidimensional, uma vez que os dados foram transformados em vetores, com os seguintes parâmetros: 8 filtros convolucional, kernel

unidimensional , função de ativação 'relu', e como foram utilizados 2 vetores unidimensionais na entrada, de treinamento e teste, o *input_shape* = [2,1].

2. A próxima etapa seria a mineração dos dados com o objetivo de reduzir o tamanho dos vetores unidimensionais, para isso foi utilizado a função *MaxPooling1D*.

3. É aplicado a função *Flatten* que transforma o formato do dado de entrada de uma matrix [x,y] para um vetor de dimensão [x*y]

4. Para determinar a probabilidade de que o dado de entrada pertença a uma das duas classes do exemplo, é utilizado a função *Dense*, que é uma camada de 2 nós com função de ativação *softmax* que retorna um array de 2 probabilidades em que cada nó contém a probabilidade que o ponto do conjunto de teste pertença a uma das 2 classes.

O teste do modelo é precedido de uma etapa de ajuda de configurações de compilação, para este exemplo as configurações foram as seguintes:

- Para determinar a precisão do modelo durante o processo de treinamento usou-se a função de perda *categorical_crossentropy*.
- Para que o modelo adapte-se ao processamento de treinamento devido o dado de entrada e os valores da função de perda é utilizada a função de otimização *sgd*.
- Para determinar a qualidade do processo de treinamento foi utilizada a métrica de exatidão.

Como difere-se do classificador SVM por realizar o processo de treinamento por vários ciclos, o processo *fitting* para o treinamento do modelo necessita de alguns parâmetros além dos dados do vetor de características e de rótulo do conjunto de dados de treinamento, essas características são:

- Os modelos do keras para treinamento usam um conjunto de dados de uma vez para fazer a predição, no caso do exemplo o *batch* foi de tamanho 100.
- A quantidade de ciclos de treinamento escolhido foi de 500.
- E o critério para visualização do progresso das métricas durante o processo de treinamento (diminuição do valor da função perda e aumento da acurácia do modelo treinado) foi indicado como 2.

E com o modelo treinado é possível predizer a resposta obtida da aplicação no conjunto de testes.

Devido a natureza dos problemas de classificação, o resultado é obtido por mais de um único valor, a avaliação da qualidade do modelo treinado é dado por um conjunto de parâmetros, apresentados em Apêndice 6.6.

3.3 Base de dados das configurações de mão de LIBRAS

A proposta inicial do desenvolvimento do trabalho consistia em utilizar uma base de dados pronta, e a partir de uma coleta de campo, construção de uma base de dados própria, e assim, aplicando as mesmas metodologias indicar os resultados e apresentar a discussão, porém devido à política de distanciamento promovida em função da pandemia de COVID-19, optou-se por utilizar apenas a base de dados encontrada.

Andres Jessé Porfírio [7], desenvolveu uma base de dados contendo as 61 configurações de mão de LIBRAS a partir de sensores RGBD (Kinect) Tabela 3.1,

Tabela 3.1. Configurações do sensor RGBD Kinect

Especificações do sensor Kinect	
Resolução do vídeo	640x480 pixels
Taxa de captura	30 Hz
Distância do objeto à câmera	de 0,8 a 3m
Abertura do sensor a 0.8m	87x63cm
Transmissão de dados	USB

usando software de aquisição especialmente desenvolvido baseado na biblioteca OpenNI4. Atores performaram a sequência pre-definida de movimentos, 2 vezes para cada sinal, uma com a área da silhueta de interesse destacada, e outra com exposição completa da área.

Resultando em 732 vídeos, 12 para cada classificação, além dos canais RGBD(Red,Blue, Green and Depth), o vídeo ainda contém o esqueleto dos atores, gerado durante a aquisição das imagens, permitindo localizar as juntas dos atores, cada vídeo tem duração de 5 a 10 s e resolução VGA (640 x 480 pixels).

3.3.1 Extração dos frames

A obtenção dos *frames*(quadros) dos vídeos foi realizado por meio do programa apresentado no Anexo 7.1, este programa utiliza as bibliotecas os, sys, cv2, openni, argparse e numpy, abrindo o diretório onde os vídeos .oni estão, extrai os frames gerando 3 versões de cada, colorida Figura 3.4,

Figura 3.4. Versão colorida do frame.

a versão do contorno do frame na Figura 3.5,

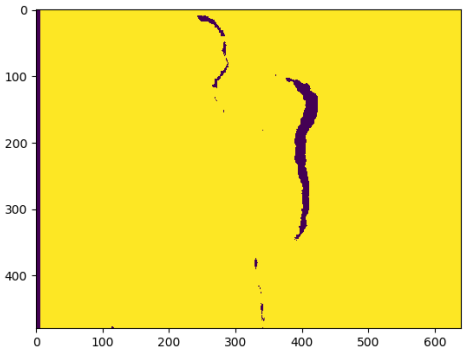


Figura 3.5. Versão do contorno do frame.

e a componente de profundidade Figura 3.6,

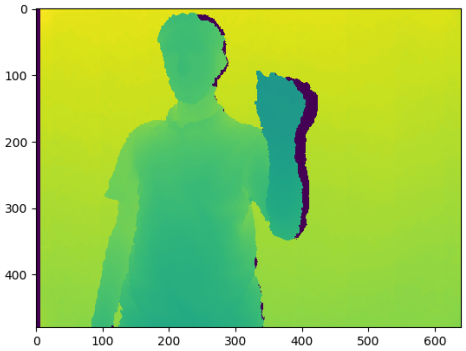


Figura 3.6. Versão da profundidade do frame.

e ao final cria um diretório novo para os frames de todos os 732 vídeos, como o intervalo de duração de cada vídeo difere, a quantidade de frames gerado também difere, e há 4 vídeos que por corrompimento do arquivo não geraram os frames, ao todo então temos 728 vídeos, 4 classificações de mão ficaram com 11 vídeos, ao invés de 12, logo causando heterogeneidade das informações.

3.3.2 Classificação para caso estático -configuração Open-Hand-Closed-hand

Para iniciar a classificação dos dados usados neste trabalho, foi usado um caso isolado para teste e verificação dos resultados obtidos, ao invés de utilizar o estudo baseado nas 61 configurações de mãos da LIBRAS, o estudo foi realizado em relação ao estudo das características mão aberta e mão fechada.

Devido a natureza da imagem, foi escolhido o classificador CNN uma vez que após extraído os frames dos videos, o objetivo é que a imagem toda fosse usada para o treinamento, para a SVM seria necessário uma etapa de pré-processamento para extração das características adequadas para tratar de estudos sobre classificação da LIBRAS.

O estudo foi feito utilizando as imagens da profundidade Figura 3.6 do frame do vídeo, foram utilizadas 2 configurações de mão, que para esse estudo não foram consideradas como um sematosEmas e sim apenas em relação as características mão aberta e fechada, duas versões utilizando o frame do vídeo em que o sinal foi realizado com a exposição completa do ator, e as duas versões que possuem a silheuta da área de interesse destaca, de um dos 5 atores.

Foram realizados 3 testes, 1 utilizando as imagens de mão fechada e mão aberta apenas com as imagens dos frames retirados do vídeo em que ocorre a exposição completa do

ator, como mostrado em Figura 3.4, um utilizando as imagens do vídeo em que a silheta está destacada, como mostrado na Figura 3.7, e o último foi utilizado uma combinação dos dois tipos de imagens, para ambos casos da mão aberta e fechada.

Figura 3.7. Versão da imagem em que o ator está com a silhueta destacada.

A programação envolvida nesta etapa é apresentada no Apêndice 6.8, como já explicado, após passar pela extração dos frames dos vídeos, o programa de extração cria pastas para cada um dos vídeos contendo os 3 tipos de imagens do mesmo frame, a primeira etapa para o processamento das imagens é extrair dessas pastas a imagem de profundidade dos frames.

O programa de extração nomeia cada tipo de imagem de modo específico, a imagem colorida é denominada (número_do_frame)_color, a de contorno é denominada (número_do _frame)_8bit e a de profundidade é (número_do_frame)_16bit, por meio dessa denominação, foi utilizado a parte "16bit"da nomeclatura do arquivo para extração das imagens de profundidade, e contagem da quantidade de imagens deste tipo de cada pasta.

Assim como foi realizado na etapa do experimento dos classificadores, as imagens foram separadas em quatro conjuntos, aqui a função foi chamada separate _training _teste, de características e rótulos de treinamento, e de características e rótulos de teste, logo, foram criados 8 conjuntos que chamam a função na main, um para as imagens de mão aberta e um para as imagens de mão fechada.

Foi indicado que 70% das imagens fossem utilizadas para que aleatoriamente os conjuntos de treinamento e teste fossem determinados, sendo que os de treinamento teriam a dimensão de 0 até um valor aleatório de (0.7· quantidade_de_imagens) assim como seus respectivos rótulos, e a dimensão dos conjuntos de teste seriam um valor aleatório de (0.7· quantidade_de_imagens) assim como seus respectivos rótulos.

Após a definição das dimensões dos conjuntos o próximo passo é a definição dos vetores de características e de rótulos de treinamento e os de teste das classes mão aberta e mão fechada, realizados pela função *join*.

O próximo passo é configurar o classificador, a lógica é a mesma seguida na do exemplo de aplicação dos classificadores porém devido aos dados serem mais complexos, as configurações possem algumas mudanças e acréscimos, usando a classe sequential do keras as camadas do modelo de treinamento possui a seguinte sequência:

1. Diferente do exemplo '+,red' e '.,blue' que possuia apenas uma camada de filtro convolucional unidimensional, para as classes mão aberta e mão fechada são usadas duas camadas de filtro convolucional bidimensional, uma vez que os dados são imagens, com os seguintes parâmetros: 8 filtros convolucionais, kernel [6,6] , função de ativação 'relu', e como foram utilizados matrizes bidimensionais na entrada, de treinamento e teste, o *input_shape* = [linhas, colunas,1], a segunda camada de filtro convolucional difere-se apenas pela quantidade de filtros que possui 6.

2. A etapa de mineração dos dados com o objetivo de reduzir o tamanho dos vetores é realizado após cada uma das camadas de filtro convolucional, para isso foi utilizado a função *MaxPooling2D* sendo a primeira com dimensão 8, e a segunda com dimensão 4.

3. É aplicado a função *Flatten* que transforma o formato do dado de entrada de uma matrix [x,y] para um vetor de dimensão [x·y]

4. Para determinar a probabilidade de que o dado de entrada pertença a uma das duas classes do exemplo, é utilizado a função *Dense*, que é uma camada de 2 nós com função de ativação *softmax* que retorna um array de 2 probabilidades em que cada nó contém a probabilidade que o ponto do conjunto de teste pertença a uma das 2 classes.

O teste do modelo é precedido de uma etapa de ajuda de configurações de compilação, como para a classe de dados utilizado as informações são mais complexas as configurações tiveram que ser melhor desenvolvidas para gerar um modelo mais robusto, as características determinadas foram:

- Para determinar a precisão do modelo durante o processo de treinamento usou-se a função de perda *categorical_crossentropy*.

- Para que o modelo adapte-se ao processamento de treinamento que agora possui caráter mais complexo devido o dado de entrada e os valores da função de perda é utilizada a função de otimização *sgd* que também precisou ser configurada utilizando alguns parâmetros: lr = 0,0100, decay de 1e-6, momentum de 0.9 enesterov = True, estes parâmetros estão associados ao *bias*, e significa que os pesos gerados no

treinamento serão atualizados com um fator do produto da taxa de aprendizagem em relação ao gradiente de decaimento de 1e-6 quando o momentum for 0,9.

- Para determinar a qualidade do processo de treinamento foi utilizada a métrica de exatidão.

O processo *fitting* para o treinamento do modelo necessita de alguns parâmetros além dos dados do vetor de características e de rótulo do conjunto de dados de treinamento, essas características são:

- A definição do conjunto de dados de uma vez para fazer a predição, *batch* foi de tamanho 200.
- A quantidade de ciclos de treinamento escolhido foi de 750.
- E o critério para visualização do progresso das métricas durante o processo de treinamento (diminuição do valor da função perda e aumento da acurácia do modelo treinado) foi indicado como 2.

O resultado do treinamento do modelo de predição segue o mesmo critério dos usados no exemplo das duas classes de experimento dos classificadores, as métricas são as mesmas, apresentados no Apêndice 6.6. E como neste exemplo o modelo foi treinado para que fosse criado um arquivo de treinamento (json) e um arquivo com os pesos(h5), foi necessário criar um outro programa para testar o modelo de predição.

Utilizando a função de definição dos rótulos, é necessário primeiro fazer a leitura do arquivo do modelo, carregar os pesos, e seguinte fazer a escolha das imagens de profundidade, de preferência escolhendo as imagens em que sua versão colorida é conhecida, para quando aplicar a predição verificar se o resultado do preditor coincidiu com as imagens esolhidas.

3.4 *Data Augmentation*, subamostragem e reescalonamento da base de dados

Dentro das etapas propostos no escopo deste projeto, estão o reescalonamento e subamostragem, que constituem etapas de pré-processamento dos dados de entrada do algoritmo de treinamento do modelo, e este, como já citado, será realizado por meio de um algoritmo de CNN, o que implica na necessidade de uma grande base de dados para que o sistema possa generalizar de modo correto as classes.

Ainda dentro das 2 classes *Open Hand* e *Closed Hand* para validação da metodologia, a primeira etapa realizada foi a aplicação de algoritmos de data augmentation para aumentar a quantidade de imagens da base de dados, porém é necessário atenção com a metodologia aplicada, como as informações deste trabalho incluem um característica dinâmica que é o movimento da mão e formato da configuração, a alteração deve ser feita de modo a não descaracterizar a imagem de modo que possa acabar sendo confundida com outra devido a alteração.

A base é formada por 12 diretórios, cada um com 61 diretórios que são as configurações de mão, logo, para cada configuração existem 12 exemplos, no caso das configurações 6, 33, 35 e 42 possuem apenas 11, pois 1 dos diretórios tiveram erro na extração dos frames.

A quantidade de frames diferem-se por configuração de mão, para solucionar esse problema propõe-se a equalização da quantidade de frames por meio da subamostragem das imagens, a equalização ocorreu convertendo as imagens que foram extraídas dos vídeos com extensão oni, novamente para vídeo, desta vez em extensões avi e mp4, usando uma função do OpenCV, o ffmpeg que permite a definição da quantidade de frames em um vídeo.

A primeira tentativa foi realizada com a aplicação em uma série de algoritmos na linguagem python de data augmentation de filtragem, ruído e mudança do mapa de cores de todas as imagens, depois utilizando o matlab foi criado uma função que fazia a conversão das imagens em um novo vídeo de quantidade de frames fixa, foram escolhidas a quantidade de 30 frames, uma vez que o Kinect possui taxa de 30Hz (quadros/segundo), reescalonou-se temporalmente para que todos os vídeos tivessem 1 segundo, os vídeos então eram processados por outro algoritmo que extraia os frames dos vídeos, desta forma, todos as classes teriam a mesma quantidade de frames.

A segunda tentativa foi realizada com apenas um algoritmo em python que por meio de funções baseadas na aplicação de alguns mecanismos de data augmentation usados na primeira foi realizado um arranjo combinatório entre as funções para ampliar a variabilidade da base, e gerava um vídeo a partir dessas imagens, com quantidade fixa de 30 frames como foi proposto na metodologia anterior, porém desta vez os frames não foram extraídos novamente, optou-se por usar os vídeos como dados de entrada para o classificador.

Os vídeos foram nomeados em relação a configuração de mão, o participante da base, o modo que o vídeo foi gravado (ocorre a alteração de vestimenta dos participantes em alguns, ou mudança da localização onde o vídeo foi gravado), e segue com a nomeclatura em relação as funções de data augmentation que serão aplicadas.

3.5 Classificação das configurações de mão a patir de um vídeo com quantidade fixa de 30 frames

O treinamento do modelo para as configurações de mão, difere-se do caso estático usando como estudo de caso, pois a Libras pressupõe uma característica dinâmica, a mão se move no espaço, não fica parada, logo o reconhecimento das mesmas pode ocorrer identificando a sequência das posições ao longo do tempo, a proposta indicada é que a entrada do classificador sejam os aquivos com os vídeos gerados com 30 frames, ao invés dos frames separadamente como foi realizado na classificação dos casos estáticos.

Usar o vídeo ao invés dos frames como entrada altera a dimensionalidade do problema do caso estático, de um vetor passa a ser um tensor, e a quantidade de frames deve ser levado em consideração dentro deste tensor, inicialmente mantendo as outras características utilizadas no caso estático para que possa ser feito uma comparação entre os valores obtidos em ambos os casos.

O conjunto de treinamento e teste foram obtidos por meio das configurações de mão 1 e 56, que representam as classes *Open Hand* e *Closed Hand* respectivamente, os participantes identificados como 1,2,3 e 6 para treinamento e os partipantes identificados como 4 e 5 para teste, por meio da nomeclatura do arquivos de vídeos ocorreu a separação dos tensores que contém os dados dos frames, e os rótulos dos vídeos.

Para o treinamento do classificador, a entrada serão os vídeos, logo, teremos um tensor, e pelo tamanho da quantidade de frames estar fixo, este valor pode ser especificado no dimensionamento do tensor, devido a quantidade de vídeos, o classificador vai demorar a rodar a cada novo experimento realizado, optou-se por salvar os dados dos vídeos em um arquivo binário de extensão mat, dessa forma, ao invés do programa ter que carregar todos os vídeos, terá que carregar apenas os dados já salvos.

A quantidade de camadas e hiperparâmetros da rede serão testados tanto com dimensões e valores semelhantes aos usados nos estudos de caso, quanto novos valores, uma vez que a dimensionalidade do problema alterou-se com a passagem da entrada apenas das imagens para os vídeos, logo, pretende-se analisar o impacto que essa alteração ocasionou no classificador implementado quando decidiu-se usar a CNN.

Os testes do modelo treinado ocorrerão de forma semelhante aos do caso *Open Hand* e *Closed Hand*, vídeos serão escolhidos aleatoriamentes e por meio dos arquivos de extensão json e h5 que guardarão as características do modelo gerado, e ocorrerá a predição, resultado este essencial para a avaliar a ampliação do uso da metodologia proposta para as outras configurações de mão.

4 Resultados

4.1 Resultado dos experimentos preliminares dos classificadores

Para ambos classificadores SVM e CNN, o programa foi implementado 4 vezes e como a distribuição das nuvens de pontos ocorre de modo aleatório, os valores obtidos foram diferentes e os resultados das métricas de avaliação dos exemplos de distribuição linear Tabela 4.1,

Tabela 4.1. Valor dos parâmetros resultantes da classificação de duas classes com distribuição linear de características usando SVM e CNN

Parâmetros	SVM	CNN
Exatidão	0,940, 0,942, 0,905, 0,927 média = 0,928±0,014	0.932, 0.942, 0,910, 0,932 média = 0,929±0,011
Sensibilidade	0.940, 0.925, 0.915, 0.910 média = 0,922±0,011	0,935, 0,930, 0,910, 0,915 média = 0,922±0,010
Precisão	0,940, 0,958, 0,897, 0,943 média = 0,934±0,022	0,930, 0,953, 0,910, 0,948 média = 0,932±0,016
F1-*measure*	0,940, 0,941, 0,905, 0,926 média = 0,927±0,014	0,932, 0,941, 0,910, 0,931 média = 0,928±0,011

e as apresentações da média e margem de erro no Gráfico 4.1,

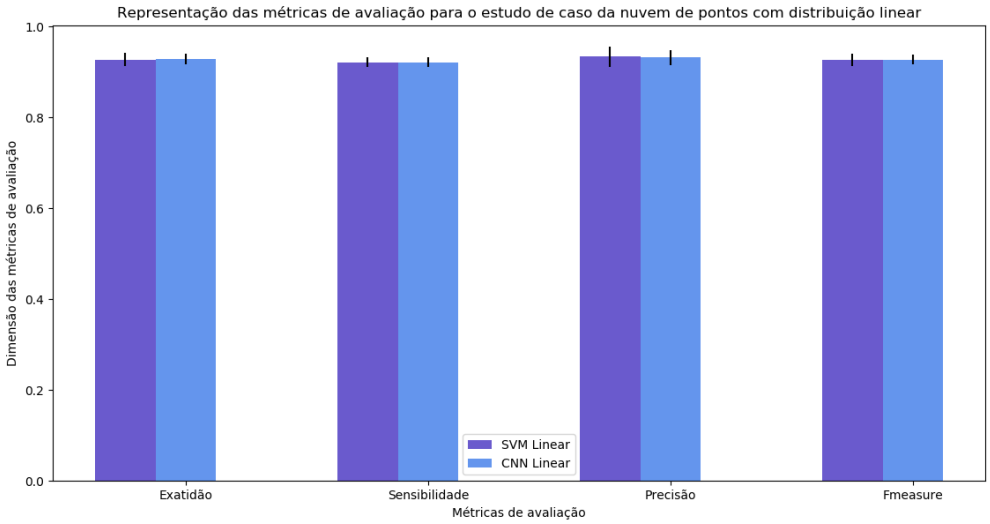


Figura 4.1. Gráfico da média e margem de erro dos parâmetros resultantes do caso de distribuição linear para o estudo de caso dos classificadores.

Os valores da distribuição circular na Tabela 4.2,

Tabela 4.2. Valor dos parâmetros resultantes da classificação de duas classes com distribuição cirular de características usando SVM e CNN

Parâmetros	SVM	CNN
Exatidão	0,995, 0,970, 0,997, 0,997 média = 0,987±0,011	0,982, 0,967, 0,967, 0,980 média = 0,974±0,007
Sensibilidade	1,000, 0,972, 1,000, 1,000 média = 0,993±0,121	0,938, 0,927, 0,886, 0,940 média = 0,922±0,217
Precisão	0,982, 0,922, 0,991, 0,990 média = 0,971±0,028	1,000, 0,953, 1,000, 0,979 média = 0,983±0,019
F1-*measure*	0,991, 0,946, 0,995, 0,995 média = 0,981±0,020	0,968, 0,940, 0,940, 0,959 média = 0,951±0,012

e as apresentações da média e margem de erro no Gráfico 4.2,

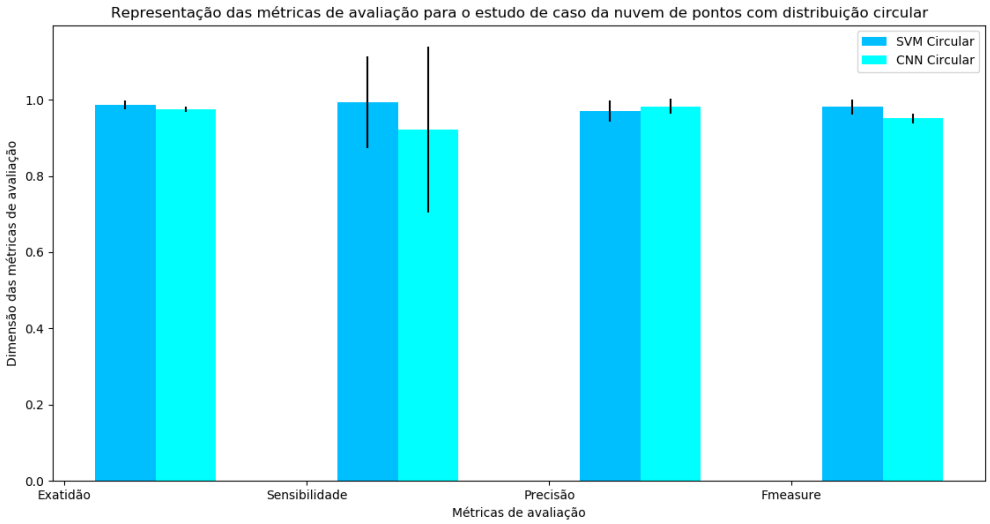


Figura 4.2. Gráfico da média e margem de erro dos parâmetros resultantes do caso de distribuição circular para o estudo de caso dos classificadores.

Como pode ser visto no Gráfico 4.3, para ambos casos do estudo, da distribuição linear e circular, os classificadores SVM e CNN, atingiram ótimos resultados, o objetivo desta etapa era determinar e validar o uso dos algortimos com relação a sinais distintos, no caso circular houve maiores variações nos valores do que no linear, o que era esperado uma vez que é um sinal mais complexo.

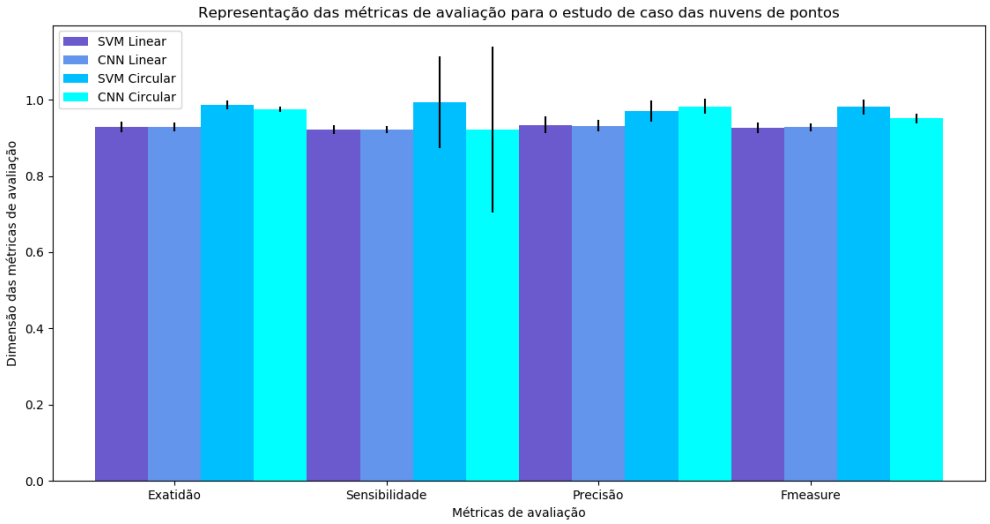


Figura 4.3. Gráfico da média e margem de erro dos parâmetros resultantes do caso de distribuição linear e circular para o estudo de caso dos classificadores.

4.2 Estudo de caso: imagens estáticas com *Open-hand* e *Closed-hand*

4.2.1 Imagem com a exposição completa do ator

Dada a quantidade de imagens dos diretórios de cada classe na entrada do algoritmo, apresentado na Tabela 4.3

Tabela 4.3. Quantidade de imagens para cada classe.

Classe	Quantidade de imagens de profundidade
Open-hand	142
Closed-hand	88

Após a conclusão do processo de treinamento, o resultado das métricas de avaliação podem ser avaliadas na Tabela 4.4,

Tabela 4.4. Resultado das métricas de avaliação. O valor elevado das métricas pode estar associado a grande diferença das duas classes testadas preliminarmente e pode ser um indício de *overfit*

Métricas de avaliação	Valor
Exatidão	1,0
Sensibilidade	1,0
Precisão	1,0
F1-*measure*	1,0

na próxima etapa do trabalho advindo deste procedimento os resultados passarão por uma avaliação de validação cruzada e reamostragem aleatória [3] para evitar que problemas de amostragem ocorram.

O teste de modelo para predição ocorreu com a escolha de 8 imagens aleatórias, 4 de cada uma das classes, como pode ser visto na Figura 4.4

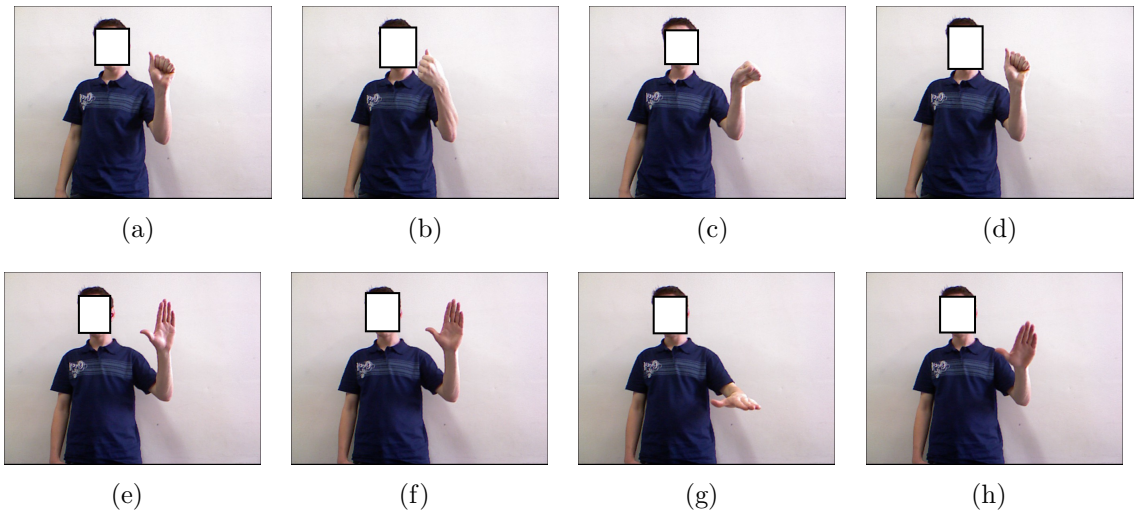


Figura 4.4. Imagens utilizadas para o teste do modelo de predição.

Logo, como apresentado na Tabela 4.5,

Tabela 4.5. Resultado da aplicação de imagens no modelo de predição

Figura	Classificação Correta	Predição
1	Fechada	Fechada
2	Fechada	Fechada
3	Fechada	Fechada
4	Fechada	Fechada
5	Aberta	Aberta
6	Aberta	Aberta
7	Aberta	Aberta
8	Aberta	Aberta

o modelo de predição conseguiu acertar todos os exemplos, este resultado será discutido na conclusão.

4.2.2 Imagens com a silhueta da área de interesse destacada

Dada a quantidade de imagens dos diretórios de cada classe na entrada do algoritmo, apresentado na Tabela 4.6

Tabela 4.6. Quantidade de imagens para cada classe.

Classe	Quantidade de imagens de profundidade
Open-hand	131
Closed-hand	133

Após a conclusão do processo de treinamento, o resultado das métricas de avaliação podem ser avaliadas na Tabela 4.7,

Tabela 4.7. Resultado das métricas de avaliação. O valor elevado das métricas pode estar associado a grande diferença das duas classes testadas preliminarmente e pode ser um indício de *overfit*

Métricas de avaliação	Valor
Exatidão	1,0
Sensibilidade	1,0
Precisão	1,0
F1-*measure*	1,0

na próxima etapa do trabalho advindo deste procedimento os resultados passarão por uma avaliação de validação cruzada e reamostragem aleatória [3] para evitar que problemas de amostragem ocorram.

O teste de modelo para predição ocorreu com a escolha de 8 imagens aleatórias, 4 de cada uma das classes, como pode ser visto na Figura 4.5.

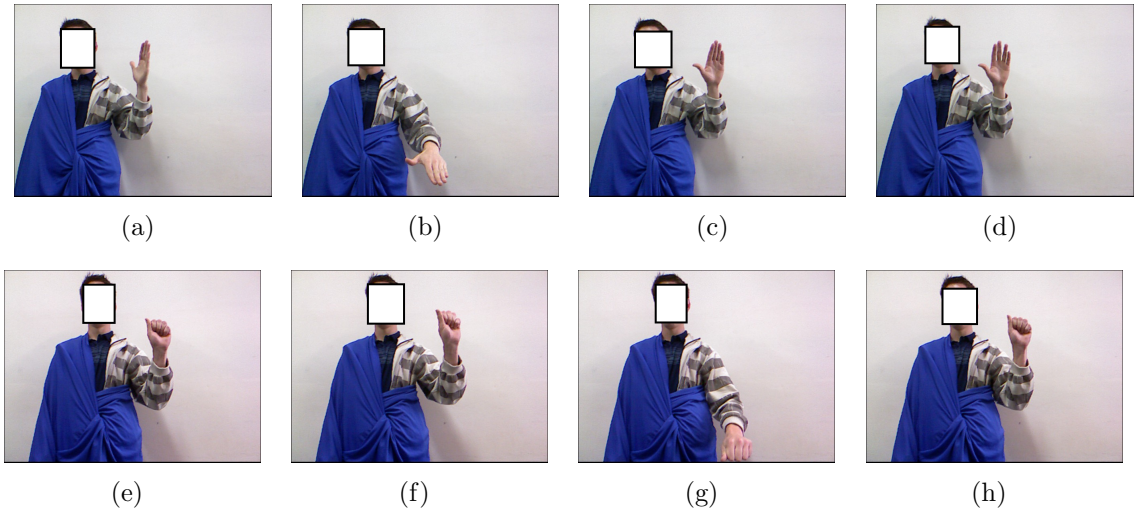


Figura 4.5. Imagens utilizadas para o teste do modelo de predição.

Logo, como apresentado na Tabela 4.8,

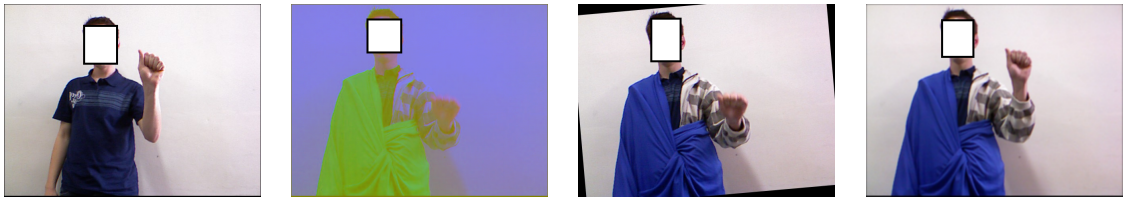
Tabela 4.8. Resultado da aplicação de imagens no modelo de predição

Figura	Classificação Correta	Predição
1	Aberta	Aberta
2	Aberta	Aberta
3	Aberta	Aberta
4	Aberta	Aberta
5	Fechada	Fechada
6	Fechada	Fechada
7	Fechada	Fechada
8	Fechada	Fechada

o modelo de predição conseguiu acertar todos os exemplos, este resultado será discutido na conclusão.

4.3 Avaliação da etapa de *Data Augmentation*, subamostragem e reescalonamento da base de dados

A implementação dos algoritmos de data augmentation ocorreu utilizando 20 classes para gerar as imagens, 3 delas podem ser vistas nas Figuras 4.6.b, onde ocorreu mudança no mapa de cores da imagem para Hue Saturation Value(HSV), 4.6.c, que a imagem foi rotacionado tomando o centro da imagem como ponto rotacional e 4.6.d, que foi aplicação de um filtro na imagem, a rotação foi aplicada na função variando 5 unidades, o valor foi deixado baixo pois a rotação poderia fazer com que a mão ficasse em uma posição que coincidiria com algumas das outras configurações.



(a) (b) (c) (d)

Figura 4.6. Imagens resultantes da aplicação dos algoritmos de data augmentation sobre a base de dados, sendo representada pelos casos *Open Hand* e *Closed Hand*. (a) Imagem original;(b) Imagem com a aplicação da mudança do mapa de cores; (c) Imagem com rotação usando o centro da imagem como centro, e (d) Imagem com aplicação de filtro blur.

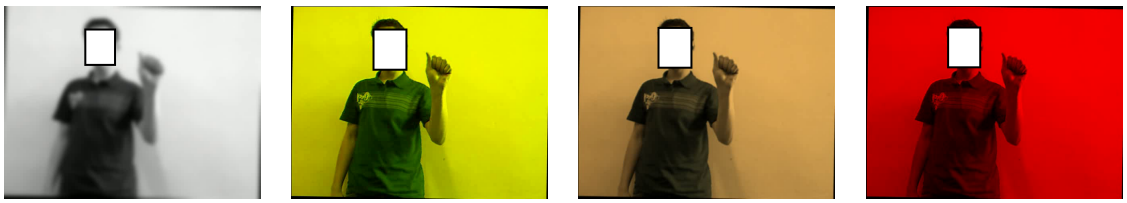
A primeira tentativa de lidar com o problema da subamostragem conseguiu obter os resultados esperados, porém foi uma metologia muito trabalhosa visto que foi feita a conversão manualmente das imagens de nova classe derivada da aplicação do data augmentation para vídeo, seguido da extração dos frames do novo vídeo, além de terem sido realizados em linguagens diferentes e programas diferentes, 1 dos 12 diretórios de apenas 1 das 61 configurações de mão após passar por todo processo resultou em um diretório com 4,4GB de imagens, para os outros 727 diretórios seria necessário a disponibilidade de 3,2TB de memória em disco, e como citado, o notebok em que o trabalho foi desenvolvido tinha pouco mais que 240GB disponíveis, logo, mostrou-se uma metologia com alto custo de memória e tempo, e foi descartada.

A segunda metodologia veio para tentar equilibrar o tempo gasto no desenvolvimento da primeira, desta vez foi utilizada apenas a linguagem python, o algoritmo varre todos os diretórios e todos os arquivos, ao invés de aplicar as mudanças nas imagens para depois convertê-las em vídeo, funções chamavam as imagens para aplicação das mudanças, Apêndice 6.11, depois outra função chamava as imagens e gerava o vídeo das mesmas, e devido a esse uso de funções, os algoritmos de data augmentation que na primeira metodologia foram aplicados um a um, puderam agora ser aplicados em várias combinações.

Utilizando a segunda metodologia aplicada aos casos de estudos *Open Hand* e *Closed Hand*, realizando 182 combinações dos 3 algoritmos implementados, obteve-se 1824 vídeos em formato mp4 de 30 frames para cada configuração, totalizando 3648 vídeos. Chamando na main do programa, Apêndice (6.10), os vídeos de 30 frames gerados por combinações como as apresentadas na lista (4.3).

- process_all_dirs('color', rotation = -1.0, translation= 0, blurlevel = 5.0)
- process_all_dirs('color', rotation = 2.0, translation= 1, blurlevel = 0.0)
- process_all_dirs('color', rotation = -2.0, translation= 1, blurlevel = 1.0)
- process_all_dirs('color', rotation = 0.0, translation= 2, blurlevel = 0.0)

Os arquivos são gerados com a nomeclatura como 'configuracao0_participante1_modo1 _frameshift0_rotation-1.0_translation0_blurlevel5.0.mp4 ' e em relação aos algoritmos que resultaram no arquivo, o programa gera um arranjo para todas as configurações, todos os participantes e todos os modos resultando em vídeos que tem frames como os exemplos que podem ser vistos na figura (4.7).



(a) (b) (c) (d)

Figura 4.7. Imagens resultantes de um frame obtido da aplicação dos algoritmos de data augmentation sobre a base de dados fazendo a combinação dos processos e geração do vídeo de 30 frames, sendo representada pelos casos *Open Hand* e *Closed Hand*. (a) Representa a chamada do primeiro item da lista (4.3); (b) Representa o segundo item; (c) Representa o terceiro e (d) Representa o quarto item da lista.

4.4 Classificação das configurações de mão a patir de um vídeo com quantidade fixa de 30 frames

A primeira etapa para o desenvolvimento do treinamento do classificador é separar os dados de treinamento e validação, isso é realizado por meio da chamada da função *extract_training_test_sets_from_videos* na main, que é composto das funções, *load_video* que carrega os vídeos gerados com o data augmentation, *list_all_files* que vai armazenar os nomes dos vídeos, um vez que estes foram gerados para especificar todas as características do vídeo, *participant_from_filename* e *class_from_filename* que são funções responsáveis por separar os participantes e a classe do vídeos em relação a configuração de mão que representa, todas essas funções são apresentadas no Apêndice 6.12.

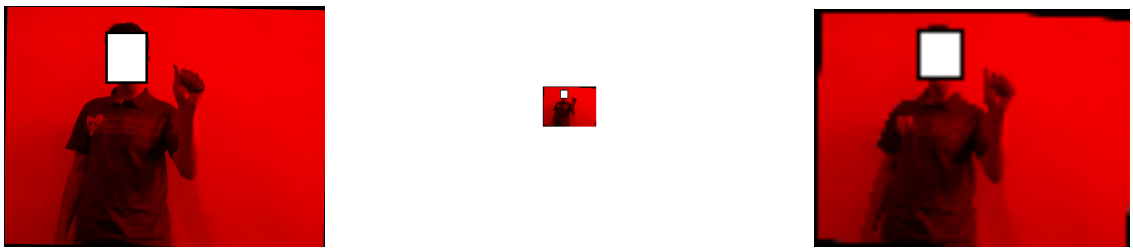
A etapa de salvar estes dados no arquivo binário de extensão mat por meio da função *save_training_testing_tensors* chamada na main do Apêndice 6.12, deve ser realizada separadamente da parte da classificação, uma vez que depois de ser gerada será utilizada pelo menos, depois deve ser comentada para que não seja executada novamente, e o início do treinamento ocorre carregando o arquivo gerado por meio da função *load_training_testing_tensors*, e inserindo na rotina de treinamento por meio da função *single_session_training_testing_cnn*, todos na main do Apêndice 6.12.

O treinamento do classificador após a separação dos dados de treinamento e validação ocorreu alterando os hiperparâmetros da função *single_session_training_testing_cnn* para a nova dimensão do tensor, que inclui o número de frames, porém o software indicou que não havia disponibilidade de memória Random Access Memory (RAM), o programa VSCode indicou que seriam necessários 240GB de memória RAM, e como já indicado, a máquina usada para realizar o projeto possui apenas 8GB de RAM disponível.

Diante deste problema, algumas soluções testadas para realizar a redução da quanti-

dade de memória RAM necessária foram, a redução da dimensão dos frames dos vídeos mantendo-se a quantidade de frames e quantidade de vídeos de cada classe para treinamento e teste, a redução da quantidade e dimensão dos frames do vídeo e mantendo a quantidade de vídeos, essa sendo possível devido as características distintas das configurações de mão usadas, e uma outra alternativa seria reduzir a quantidade de frames dos vídeos, a quantidade de vídeos, e manter a dimensão original dos frames, deste modo, os dados estariam mais parecidos com os usados para as imagens estáticas.

A redução da dimensão dos frames do vídeos de 480x640 pixels 4.8.a, foi escolhida de modo que a memória necessária fosse metade da quantidade máxima disponível, logo 4GB, foi reduzido em 64 vezes para a dimensão 60x80 pixels, como pode ser visto em 4.8.b, e ocasionou uma baixa resolução espacial 4.8.c, essa redução foi utilizada para todas as soluções que usaram os frames redimensionados.



(a) (b) (c)

Figura 4.8. Imagens resultantes do Reescalonamento dos frames para que fosse reduzido o consumo de memória RAM no processo de treinamento do modelo de classificação, (a) Imagem que compõe o vídeo de 30 frames, com dimensão 480x640; (b) Imagem reescalonada em 64 vezes, logo, com dimensão 60x80 e (c) Imagem reescalonada, aumentada para 480x640 com a nova resolução.

A primeira alternativa de solução para o problema da memória RAM, que consistia em manter a quantidade de frames e de vídeos, e apenas reduzir o dimensionamento dos frames não foi uma solução que pode ser muito explorada, pois mesmo que a redução utiliza-se apenas dos 4GB de RAM planejados, ainda assim tornou o notebook muito lento, não foi possível implementar de modo a alterar as camadas e os hiperparâmetros do classificador, e assim obter os resultados necessários para avaliação da eficácia da solução.

Para não reduzir mais a imagem e ocasionar a descaracterização, foi necessário usar as alternativas que combinavam o redimensionamento já aplicado com a redução da quantidade de frames dos vídeos e a redução da quantidade de vídeos de treinamento e teste, e a que combina a redução da quantidade de frames e vídeos, mantendo a dimensão original de 480x640 pixels.

Para todos os casos também foi gerada dois gráficos, um relacionando o crescimento da acurácia em relação ao número de épocas de treinamento, comparando o desempenho para os dados usados para treinamento e validação, e outro gráfico que compara os desempenhos dos dados de treinamento e validação que relaciona a função de perda e a quantidade de épocas de treinamento, para isso poder determinar a composição das alternativas que apresentou o melhor resultado.

4.4.1 Treinamento por meio de vídeos com redimensionamento e redução da quantidade de frames

Em termos da quantidade de frames do vídeo original, os valores foram escolhido como prova de teste, dado que as configurações de mão escolhidas são distintas, esperava-se encontrar uma tendência no treinamento que fosse capaz de gerar um classificador com bons resultados independente da quantidade de frames, com a baixa resolução espacial já sendo considerada para todos os casos, e mantendo a quantidade de vídeos da base.

As tentativas de configuração podem ser vistas na função *single_session_training_testing_cnn* no Apêndice 6.12.

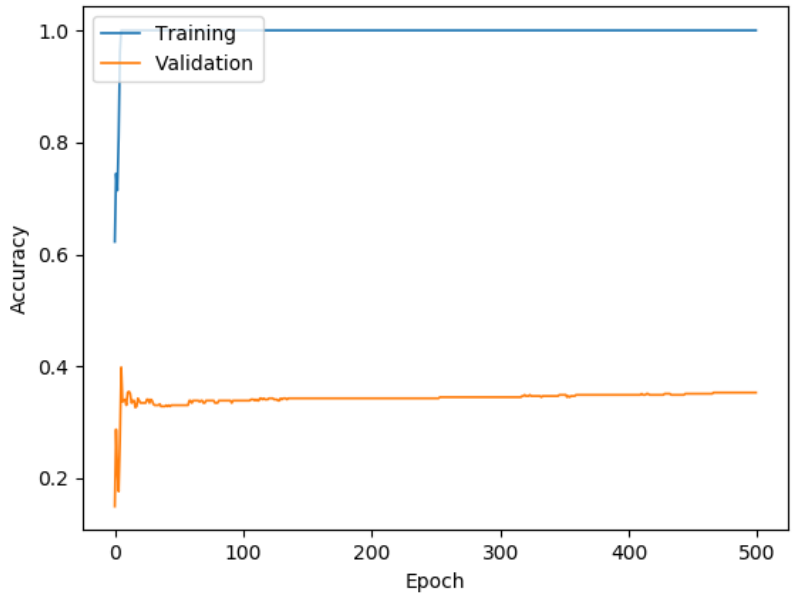
Em relação aos gráficos que relacionam a acurácia e função de ativação com a quantidade de épocas, para ambos casos de treinamento e validação, é esperado que as curvas tenham característica assintótica, indicando que o treinamento seguiu o formato adequado.

Para 1 frame

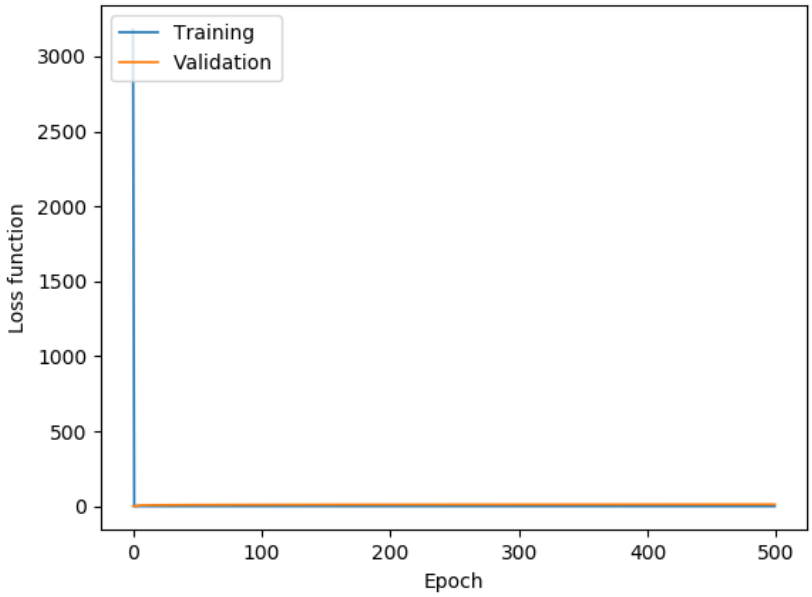
Foi realizado utilizando 4 camadas sendo destas uma camada convolucional, com as seguintes características:

- 13 filtros convolucionais de dimensão 3x3, configurado para 1 frame na dimensão do tensor;
- Pooling com dimensão 3x3;
- Dense = 2, uma vez que só há duas configurações;
- Otimizadores, lr = 0,00965, decay = $0{,}225e^{-6}$ e momentum = 0,115;
- Função de perda = categorical_crossentropy;
- Métrica = acurácia;
- Tamanho do Batch = 160, 500 épocas de treinamento, e 20% dos dados para validação.

Essas configurações dos hiperparâmetros resultaram em 12,976 parâmetros treináveis, o resultado do treinamento é apresentado nos gráficos 4.9.a e 4.9.b que relacionam os valores da acurácia e da função de perda respectivamente, em relação a quantidade de épocas de treinamento.



(a)



(b)

Figura 4.9. Gráficos resultantes da geração do classificador para o caso de 1 frame sendo usado para compor o vídeo, (a) Gráfico que mostra a relação da acurácia de treinamento em função da quantidade de épocas para os casos dos dados de treinamento e validação; (b) Gráfico que mostra a relação da função de perda de treinamento em função da quantidade de épocas para os casos dos dados de treinamento e validação.

Os resultados das métricas de avaliação podem ser vistas na tabela 4.9,

Tabela 4.9. Resultado das métricas de avaliação.

Métricas de avaliação	Valor
Exatidão	0,6168
Sensibilidade	0,7056
Precisão	0,5992
F1-*measure*	0,6480

e a representação gráfica dos valores das métricas no Gráfico 4.10

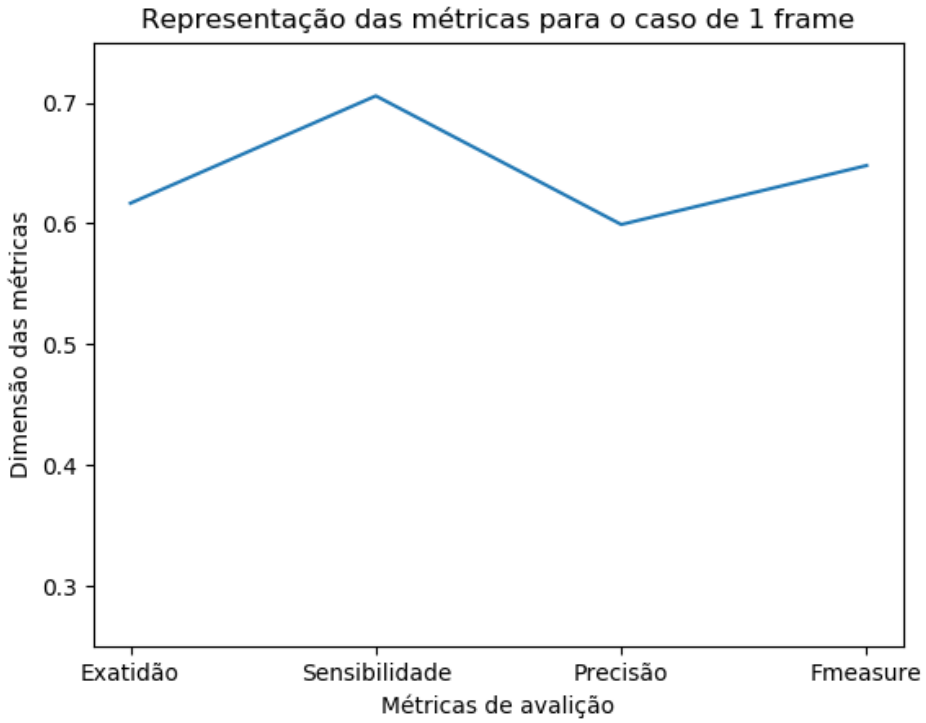


Figura 4.10. Gráfico que apresenta a tendência dos valores das métricas de avaliação para a redução da quantidade de frames de 30 para 1.

Para 2 frames

Foi realizado utilizando 4 camadas sendo destas uma camada convolucional, com as seguintes características:

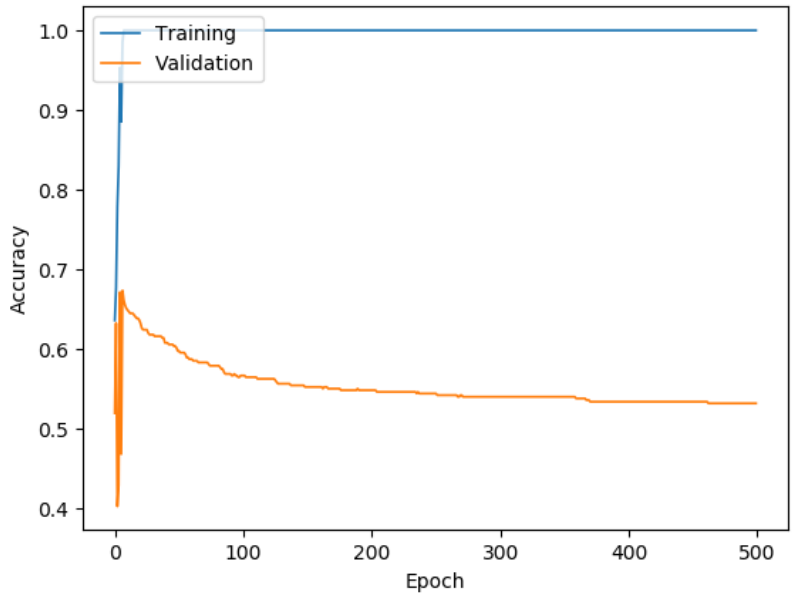
- 13 filtros convolucionais de dimensão 3x3 , configurado para 2 frames na dimensão do tensor;
- Pooling com dimensão 3x3;
- Dense = 2, uma vez que só há duas configurações;
- Otimizadores, lr = 0,00965, decay = $0{,}225e^{-6}$ e momentum = 0,115;
- Função de perda = categorical_crossentropy;
- Métrica = acurácia;
- Tamanho do Batch = 160, 500 épocas de treinamento, e 20% dos dados para validação.

Essas configurações dos hiperparâmetros resultaram em 2,752 parâmetros treináveis, o resultado do treinamento é apresentado nos gráficos 4.11.a e 4.11.b que relacionam os valores da acurácia e da função de perda respectivamente, em relação a quantidade de épocas de treinamento,

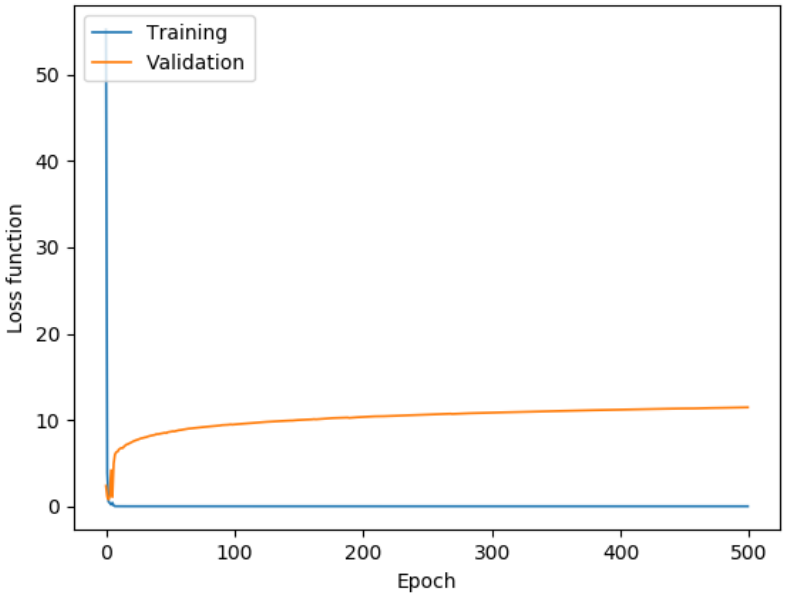
os resultados das métricas de avaliação podem ser vistas na tabela 4.10, e a representação gráfica dos valores das métricas no Gráfico 4.12.

Tabela 4.10. Resultado das métricas de avaliação.

Métricas de avaliação	Valor
Exatidão	0,6234
Sensibilidade	0,6318
Precisão	0,6214
F1-*measure*	0,6264



(a)



(b)

Figura 4.11. Gráficos resultantes da geração do classificador para o caso de 2 frames sendo usado para compor o vídeo, (a) Gráfico que mostra a relação da acurácia de treinamento em função da quantidade de épocas para os casos dos dados de treinamento e validação; (b) Gráfico que mostra a relação da função de perda de treinamento em função da quantidade de épocas para os casos dos dados de treinamento e validação.

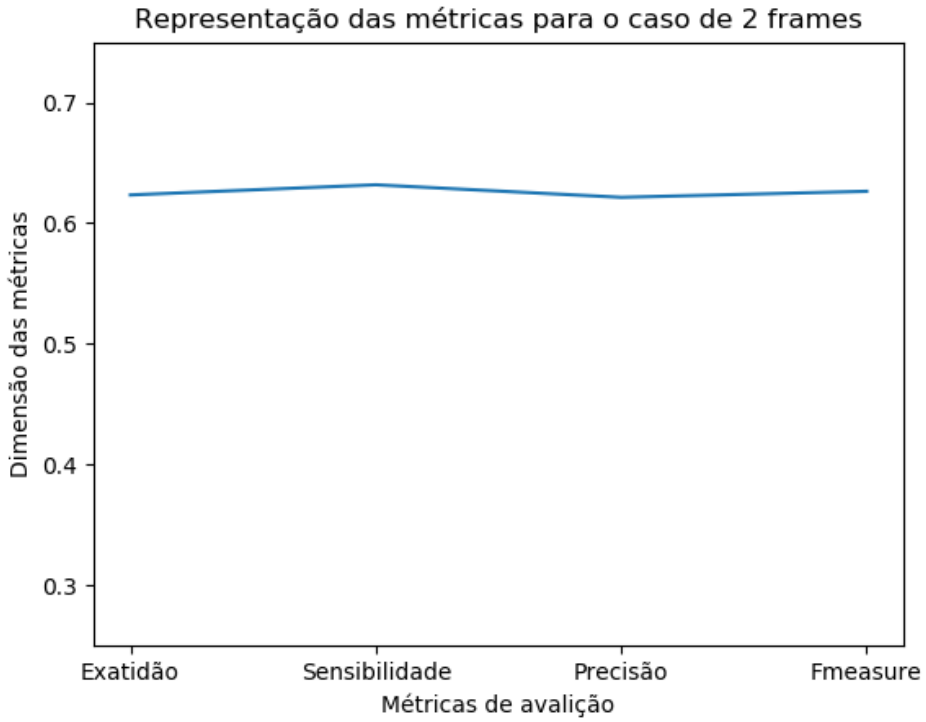


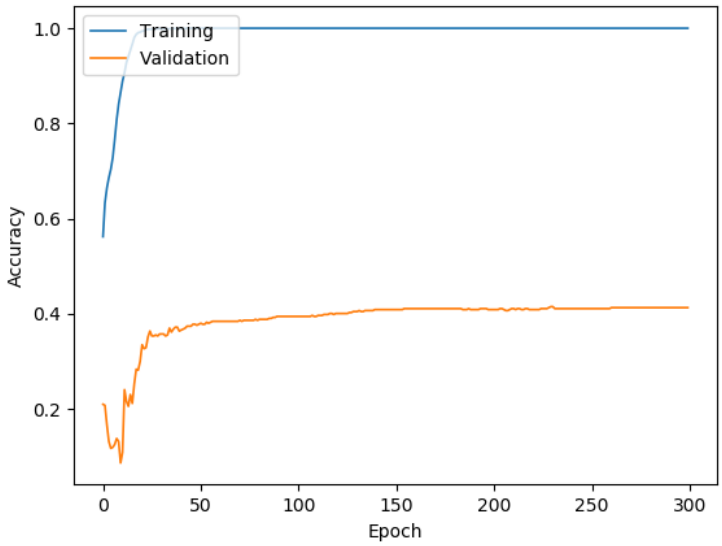
Figura 4.12. Gráfico que apresenta a tendência dos valores das métricas de avaliação para a redução da quantidade de frames de 30 para 2.

Para 3 frames

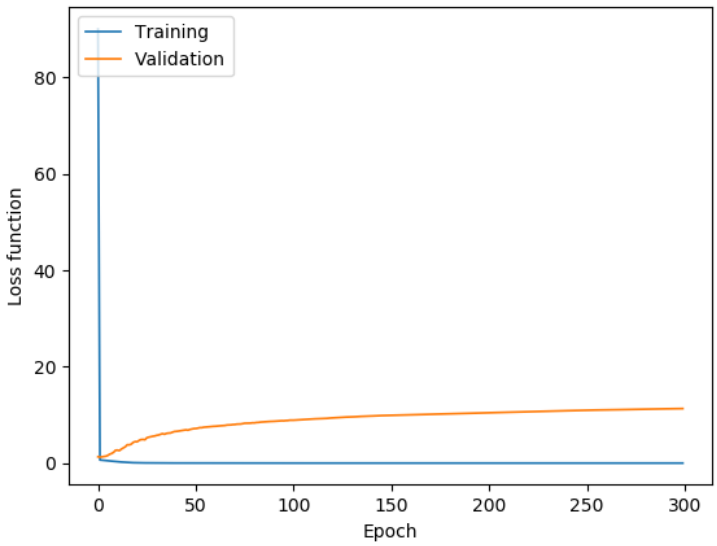
Foi realizado utilizando 4 camadas sendo destas uma camada convolucional, com as seguintes características:

- 8 filtros convolucionais de dimensão 3x3 , configurado para 3 frames na dimensão do tensor;
- Pooling com dimensão 3x3;
- Dense = 2, uma vez que só há duas configurações;
- Otimizadores, lr = 0,001, decay = $0{,}25e^{-6}$ e momentum = 0;
- Função de perda = categorical_crossentropy;
- Métrica = acurácia;
- Tamanho do Batch = 200, 300 épocas de treinamento, e 20% dos dados para validação.

Essas configurações dos hiperparâmetros resultaram em 13,210 parâmetros treináveis, o resultado do treinamento é apresentado nos gráficos 4.13.a e 4.13.b que relacionam os valores da acurácia e da função de perda respectivamente, em relação a quantidade de épocas de treinamento,



(a)



(b)

Figura 4.13. Gráficos resultantes da geração do classificador para o caso de 3 frames sendo usado para compor o vídeo, (a) Gráfico que mostra a relação da acurácia de treinamento em função da quantidade de épocas para os casos dos dados de treinamento e validação; (b) Gráfico que mostra a relação da função de perda de treinamento em função da quantidade de épocas para os casos dos dados de treinamento e validação.

os resultados das métricas de avaliação podem ser vistas na tabela 4.11,

Tabela 4.11. Resultado das métricas de avaliação.

Métricas de avaliação	Valor
Exatidão	0,5329
Sensibilidade	0,6924
Precisão	0,5249
F1-*measure*	0,5971

e a representação gráfica dos valores das métricas no Gráfico 4.14.

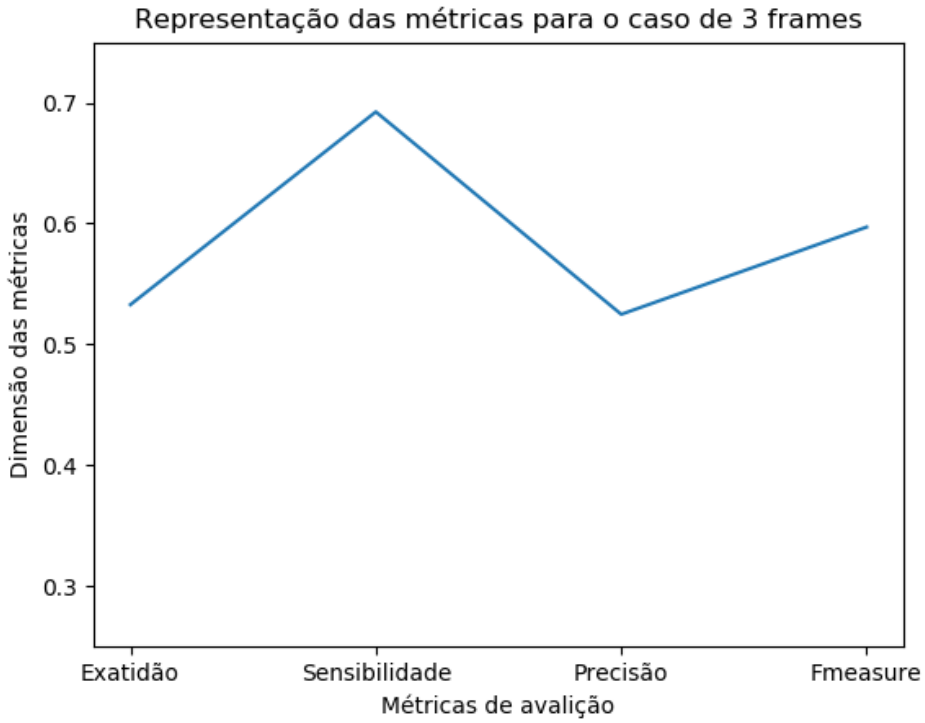


Figura 4.14. Gráfico que apresenta a tendência dos valores das métricas de avaliação para a redução da quantidade de frames de 30 para 3.

Para 15 frames

Foi realizado utilizando 4 camadas sendo destas uma camada convolucional, com as seguintes características:

- 16 filtros convolucionais de dimensão 3x3 , configurado para 3 frames na dimensão do tensor;

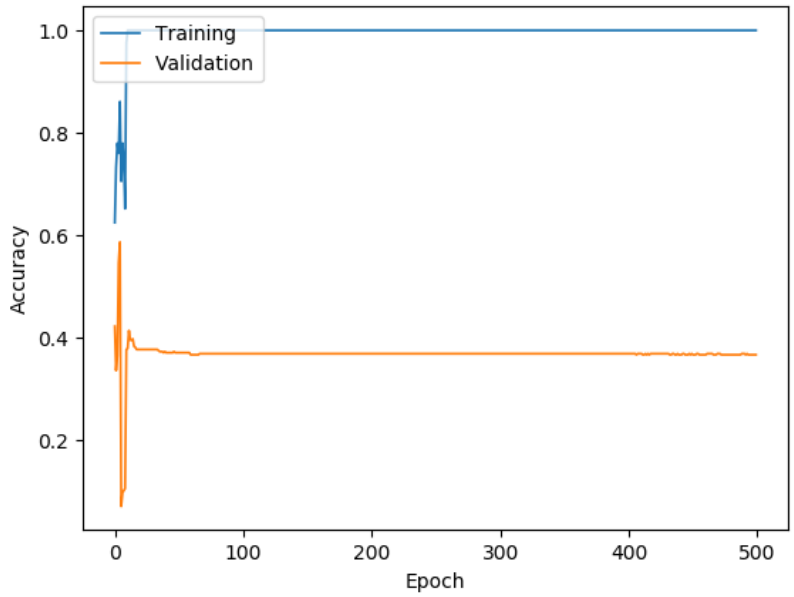
- Pooling com dimensão 3x3;
- Dense = 2, uma vez que só há duas configurações;
- Otimizadores, lr = 0,00445, decay = $0{,}315e^{-6}$ e momentum = 0,0105;
- Função de perda = categorical_crossentropy;
- Métrica = acurácia;
- Tamanho do Batch = 150, 450 épocas de treinamento, e 20% dos dados para validação.

Essas configurações dos hiperparâmetros resultaram em 17,842 parâmetros treináveis, o resultado do treinamento é apresentado nos gráficos 4.15.a e 4.15.b que relacionam os valores da acurácia e da função de perda respectivamente, em relação a quantidade de épocas de treinamento.

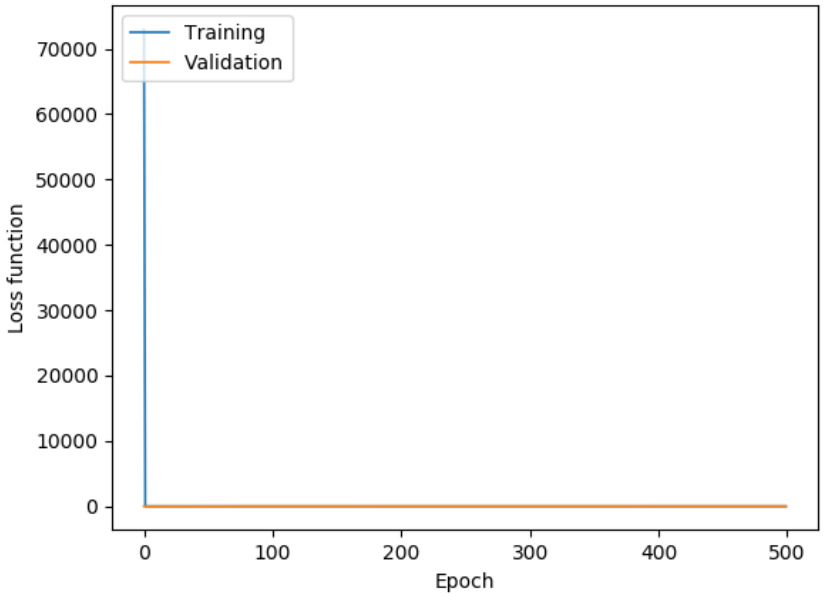
Os resultados das métricas de avaliação podem ser vistas na tabela 4.12, e a representação gráfica dos valores das métricas no Gráfico 4.16.

Tabela 4.12. Resultado das métricas de avaliação.

Métricas de avaliação	Valor
Exatidão	0,4655
Sensibilidade	0,2763
Precisão	0,4444
F1-*measure*	0,3408



(a)



(b)

Figura 4.15. Gráficos resultantes da geração do classificador para o caso de 15 frames sendo usado para compor o vídeo, (a) Gráfico que mostra a relação da acurácia de treinamento em função da quantidade de épocas para os casos dos dados de treinamento e validação; (b) Gráfico que mostra a relação da função de perda de treinamento em função da quantidade de épocas para os casos dos dados de treinamento e validação.

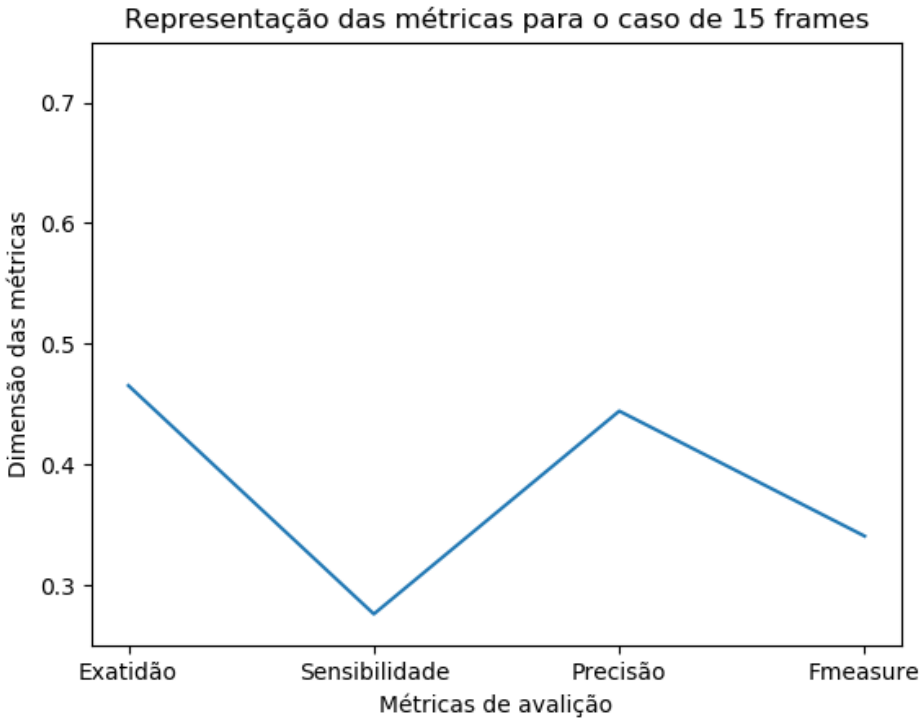
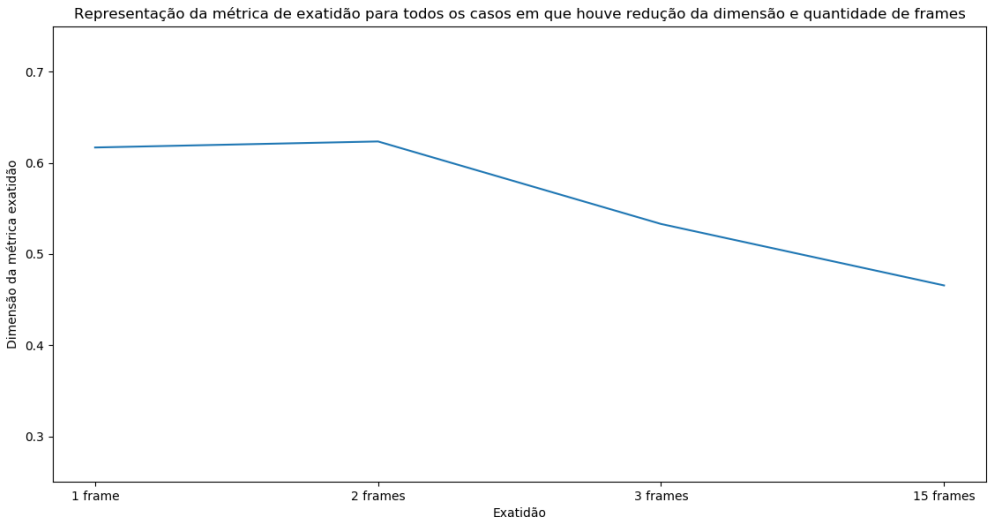
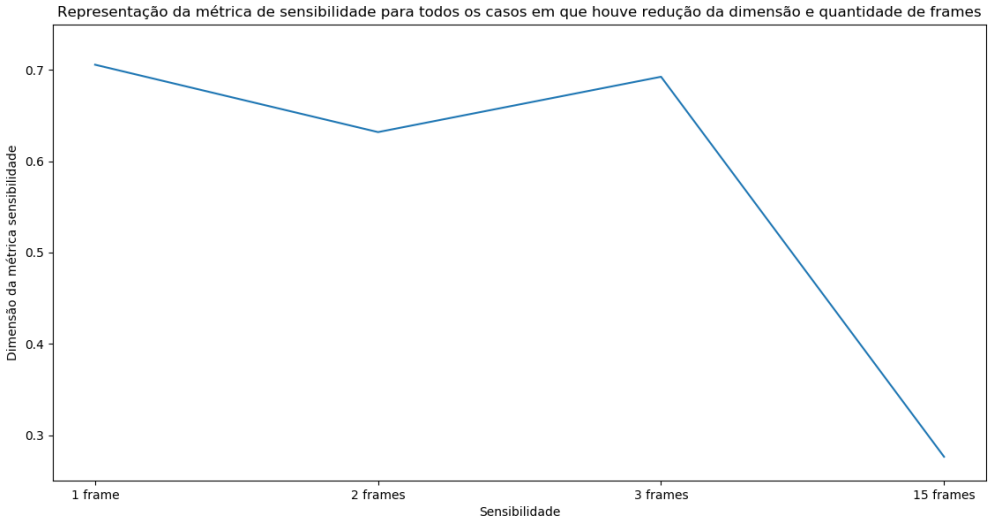


Figura 4.16. Gráfico que apresenta a tendência dos valores das métricas de avaliação para a redução da quantidade de frames de 30 para 15.

Como pode ser visto, nos casos de 1 e 2 frames apresentaram os melhores valores de métricas, principalmente exatidão e sensibilidade, como pode ser visto nas Figuras 4.17.a e 4.17.b respectivamente,



(a)



(b)

Figura 4.17. Gráficos das métricas (a) Exatidão e (b) Sensibilidade, para todos os casos apresentados da redução da dimensão e quantidade dos frames.

indicando, como suspeitava-se, que quanto mais frames, mais exemplos são necessários, embora parte do valor de 3 e 15 frames ocorreu devido a demora um pouco maior para os dados dos arquivos serem carregados e o processo de treinamento ocorrer, logo, não foi possível testar mais combinações que pudessem vir a resultar em melhores valores de métricas.

Dos gráficos obtidos, o caso de 3 frames apresentou a curva com as características mais perto do esperado como pode ser visto na Figura 4.18,

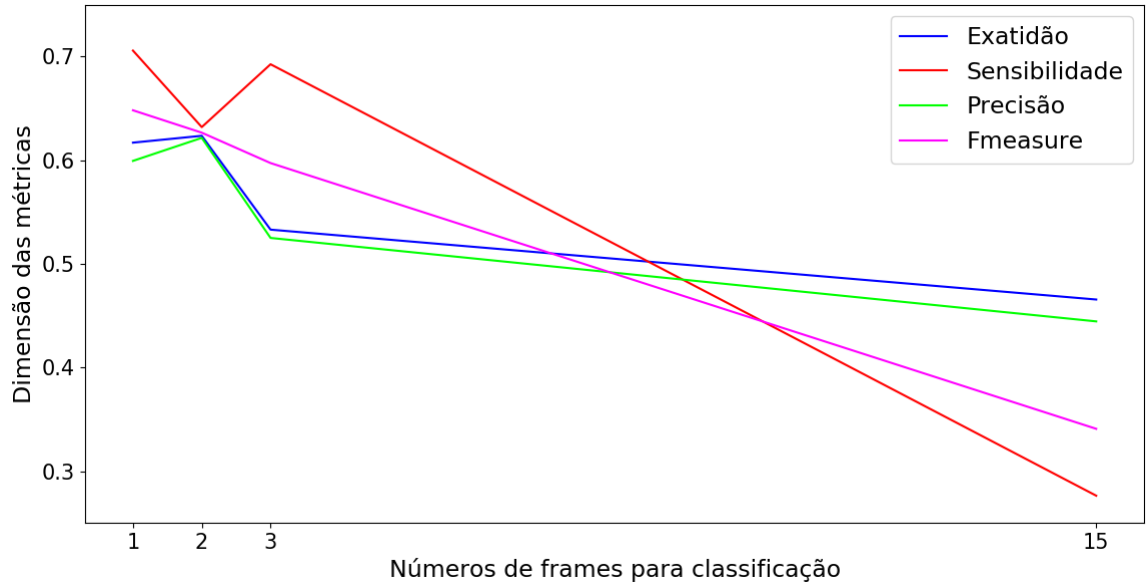


Figura 4.18. Gráfico que apresenta a tendência dos valores das métricas de avaliação, para todos os casos apresentados da redução da dimensão e quantidade dos frames.

entretanto apresentou valores muito baixos para indicar que a aplicação tenha sido bem sucedida no caso, acredita-se que a alteração de alguns dos hiperparâmetros poderia ajustar o gráfico para que obtivesse um formato mais suave, e com taxas mais altas.

Utilizando a mesma quantidade de camadas e hiperparâmetros em todos os casos, é perceptível que o classicador obtém resultados diferentes, e para isso foi necessário alterar os parâmetros em busca dos melhores resultados de métrica possível, porém ainda assim, para todos os casos em que ocorre a a baixa resolução espacial e da quantidade de frames dos vídeos, obteve valores abaixo dos obtidos no caso do treinamento da imagem estática.

Para todos os casos, as métricas não obtiveram um desempenho alto, sendo considerada alta valores de acurácia próxima ou superiores a 90%, porém também, pelos valores obtidos, acredita-se que não tenha ocorrido overfit ou underfit, então incialmente descarta-se esse problema como um dos motivos que pode ter ocasionado os treinamentos com desempenho aquém do esperado.

A abordagem do uso do classificador para o treinamento dos vídeos, seguiu uma linha metodológica análoga a usada no caso da imagem estática, logo esperava-se que os resultados das métricas fossem semelhantes, há indícios que podem ser vistos na Figura 4.18, que os resultados tem uma tendência a funcionar bem para o uso de mais frames, como os vídeos gerados com quantidade fixa de 30 frames e com o dimensionamento original de

480x640 pixels, mas para isso seria necessário aumentar a base de dados com mais vídeos para aumentar a quantidade de parâmetros treináveis.

É necessário a verificação dos algoritmos de data augmentation aplicados para avaliar se não ocasionaram a descaracterização da imagem, e aplicar novos algoritmos e mais complexos, para que assim o sistema possa ter mais parâmetros treináveis.

5 Conclusão

A proposta inicial deste trabalho visava criar um estudo de aprendizagem de máquina que pudesse contribuir para os avanços acadêmicos, tecnológicos e sociais na busca da melhoria da condição de vida das pessoas da comunidade surda-muda alfabetizada em LIBRAS ou bilíngues, e pessoas que possuam algum grau mais leve de deficiência auditiva e que também sejam gesticulantes de LIBRAS, por meio da facilitação da comunicação com pessoas não-gesticulantes, reduzindo ainda mais os contratempos ocasionados pela barreira comunicativa.

CNN é um classificador que lida muito bem com sinais da natureza do problema dos sinais da LIBRAS, sua capacidade de mapear um conjunto de características de dados de imagens 2-D em relação ao rótulo indicado para o treinamento do modelo, e capacidade ajuste dos parâmetros da rede para criar o modelo com as melhores métricas foram os aspectos usados para a escolha do mesmo, embora no estudo de caso do experimento dos classificadores a SVM tenha obtido melhores resultados nas métricas de avaliação, os sinais que compõe a base de dados não teriam o mesmo efeito.

Os valores das métricas e a capacidade de predição das imagens inseridas como teste aplicadas sobre o estudo de caso *Open Hand - Closed Hand* resultaram em ótimos valores, mesmo que o caso use imagens simples diante existe uma suspeita de ocorrência de *overfitting*,logo não significa que o modelo gerado terá os mesmos resultados de avaliação do algoritmo e predição dos casos de teste para quando o estudo for replicado para todos os sinais da base de dados, todos os *frames* já foram extraídos dos vídeos, ainda faz-se necessário fazer um estudo da melhor forma de reescalonar as sequências de imagens temporalmente para gerar sinais com número fixos de imagens para gerar o modelo que vai apresentar a melhor capacidade de predição.

O uso de vídeos faz que com que o classificador tenha mais parâmetros treináveis do que no caso das imagens estáticas, o que indica a necessidade de mais exemplos para compor a base, sendo necessário a aplicação de mais algoritmos de data augmentation, sem que seja necessário reduzir a quantidade de parâmetros treináveis, pois poderia ocasionar a perda da capacidade de identificação das características para o caso das configurações

de mão que não tem características distintas tais quais os casos *Open Hand* e *Closed Hand*.

O treinamento do classificador como implementado, necessita de aumento da base de dados, e do uso da quantidade fixa de frames resultantes do reescalonamento temporal feito na etapa de geração dos vídeos, a alteração das camadas e hiperparâmetros do classificador para testar várias configurações distintas como necessário, dependem dos recursos computacionais disponíveis.

A capacidade computacional do notebook utilizado impactou na aplicação da metodologia, é necessário maior espaço em disco maior para o aumento da base com os vídeos, maior memória RAM para que os vídeos usados fossem os com a quantidade fixa de 30 frames, são necessários para que ocorra a geração de mais exemplos treináveis para que os parâmetros de treinamento sejam atendidos.

Inicialmente planejou-se que ao fim do projeto, todas as 61 configurações de mão que constituem a LIBRAS seriam analisadas, usando a metodologia proposta de subamostragem e reescalonamento das imagens, ainda na primeira etapa que foi o estudo da Inteligência Artificial (IA) e suas vertentes, a busca pelo melhor classificador para o problema proposto e testes de um estudo de caso usando 2 das configurações, apresentaram resultados confiáveis que permitiram a escolha do classificador dado suas características distintas.

Várias hipóteses foram testadas para buscar os melhores resultados do classificador para o problema, e durante essas hipóteses, novas metodologias tiveram que ser implementadas para se adequar as adversidade encontradas, o aumento da base de dados por uso de algoritmos de data augmentation, o reescalonamento temporal para 1 segundo, que gerou os vídeos com tamanho fixo 30 frames, a redução da resolução espacial, porém dos casos propostos apenas o caso com redução da dimensão e quantidade dos frames foi executado e gerou resultados passíveis de análise, e até o momento, os resultados possuem valores das métricas mais baixos do que no caso das imagens estáticas, a impossibilidade de testar mais hipóteses deixa a dúvida se o erro foi ocasionado pelas alterações aplicadas, pelo baixo números de elementos treináveis devido a dimensão da base, e avaliar se possa ter ocorrido erro na programação do algoritmo do treinamento.

O potencial do projeto é muito alto, os resultados não atenderam o que foi planejado inicialmente, mas dado os desafios tecnológicos e de implementação encontrados, o uso do estudo de caso que acabou estendendo-se para a finalização do projeto, conseguiu gerar um resultado que mostra as possibilidades de expansão do que foi realizado para as outras configurações de mão, e o quanto ainda existe para ser feito, para assim finalmente lidar

com as palavras em LIBRAS.

A evolução deste projeto pressupõe incialmente o acesso a máquinas mais com espaço em disco e memória RAM altos o suficientes para lidar com o grande número de dados, aplicação de mais algoritmos de data augmentation e uma avaliação de quais melhores alterações que não prejudiquem a constituição da base, para gerar mais vídeos válidos e assim massificar ainda mais a base de dados, a otimização do algoritmo de treinamento com implementação de rotinas de tunelamento para a busca dos melhores valores de parâmetros, quantidades de camadas do classificador, e o treinamento de uma deep CNN para lidar com o problema da linguagem natural da LIBRAS já que a formação de palavras é de natureza complexa.

Referências Bibliográficas

[1] J.Aragão, I.Magalhães, e A.Coura et al. Access and communication of deaf adults: a voice silenced in health services. *Journal of Research Fundamental Care Online*, 6(1):1–7, 2014.

[2] A.C.Lorena e A. de Carvalho. As máquinas de vetores suporte (*Support Vector Machines)*. (1):1–58, 2007.

[3] Z.H Zhou. *Ensemble Methods: Foundations and Algorithms*. 2012.

[4] Y. Lecun e Y.Bengio. Convolutional networks for images, speech, and time-series. In *The handbook of brain theory and neural networks*. 1995.

[5] J.P.Donoso. Sistema Auditivo. FFI0210 Acústica Física. Instituto de Física de São Carlos - IFSC (USP), 2017.

[6] N.Barbosa, A.Menezes, e A.Carlos et al. *Pesquisa nacional de saúde : 2013*. IBGE, Coordenação de Trabalho e Rendimento, Rio de Janeiro, 2015.

[7] A.J.Porfirio. Reconhecimento das Configurações de mão da LIBRAS a Partir de Malhas 3D. *2013 IEEE International Conference on Systems, Man, and Cybernetics (SMC), 2013*, pages 1–56, 2013.

[8] G.B.M. Gois. Reconhecimento do alfabeto de Libras usando sensor Kinect e marcadores visuais. 2014.

[9] S.B.R. Duarte e et al. Aspectos históricos e socioculturais da população surda. *História, Ciências, Saúde – Manguinhos*, 20(4):1713–1734, 2013.

[10] G. Freyre. *Homens, Engenharias E Rumos Sociais*. 1987.

[11] R.A.A. Barros, A.V. Pontes, e J.D.S. Almeida. Reconhecimento de linguagem de sinais: aplicação em LIBRAS. *JIM,2014*, pages 1–10, 2014.

[12] B.V.L. Silva e G.O.Koroishi. Reconhecimento de sinais da libras por visão computacional. pages 1–56, 2014.

[13] W. Zhang, J. Wang, e F. Lan. Dynamic hand gesture recognition based on short-term sampling neural networks. *IEEE/CAA Journal of Automatica Sinica*, 8(1):110–120, 2021.

[14] R.C. Gonzalez. Deep Convolutional Neural Networks [Lecture Notes]. *IEEE Signal Processing Magazine*, 35(6):79–87, 2018.

[15] A.A. Mullin e F.Rosenblatt. Principles of Neurodynamics. *The American Mathematical Monthly*, 70(5):586, 1963.

[16] Y. Lecun, Y. Bengio, e G. Hinton. Deep learning. *Nature*, 521, 2015.

[17] L.M.S. Guimarães, M.R.G. Meireles, e P.E.M. Almeida. Avaliação das etapas de pré- processamento e de treinamento em algoritmos de classificação de textos no contexto da recuperação da informação. *Perspectivas em Ciência da Informação*, 24(1):169–190, 2019.

[18] S.Raschka, J.Patterson, e C.Nolet. Machine learning in python: Main developments and technology trends in data science, machine learning, and artificial intelligence. *Information (Switzerland)*, 11(4), 2020.

[19] M.R.C. MACIEL. Portadores de deficiência: a questão da inclusão social. *São Paulo em Perspectiva*, 14(2):51–56, 2000.

[20] Brasil. Ministério da Saúde. *Surdez - Bilioteca Virtual em Saúde*, 2017.

[21] R.Monteiro, D.N.H. Silva, e C.Ratner. Surdez e Diagnóstico: narrativas de surdos adultos. *Psicologia: Teoria e Pesquisa*, 32(spe):1–7, 2016.

[22] M.F.N.S. de Souza e et al. Principais dificuldades e obstáculos enfrentados pela comunidade surda no acesso à saúde : uma revisão integrativa de literatura. (3):395–405, 2017.

[23] J.R.Tedesco e J.R.Junges. Challenges for receiving hearing-impaired individuals in primary healthcare services. *Comunicação Breve*, 29(8):1685–1689, 2013.

[24] S. Nunes, A.L. Saia, L.J. Silva, e S.D.A. Mimessi. Surdez e educação : escolas inclusivas e / ou bilíngues ? *Revista Quadrimestral da Associação Brasileira de Psicologia Escolar e Educacional*, 19(3):537–545, 2015.

[25] N.N.R. Mori e R.E. Sander. História da educação dos surdos no brasil. *Seminário de Pesquisa do PPE*, pages 1–16, 2015.

[26] E. Dussel. Declaração de Salamanca: Sobre princípios, Políticas e Práticas na Área das Necessidades Educativas Especiais. *De la "conquista"a la "colonización"del mundo de la vida (Lebenswelt)*, 1982(85):323, 1994.

[27] A. Ferreira, C. Demutti, e P. Gimenez. A Teoria das Necessidades de Maslow : A Influência do Nível Educacional Sobre a sua Percepção no Ambiente de Trabalho . *XIII SEMEAD Seminário em Administração*, pages 2–17, 2010.

[28] C. Paiva, L. Adas, F. Vendrame, e et al. Uma abordagem as teorias motivacionais. page 12, 2009.

[29] M.A.N. Araújo. A estruturação da linguagem e a formação de conceitos na qualificação de surdos para o trabalho. *Psicologia: Ciência e Profissão*, 25(2):240–251, 2005.

[30] M.M.F.D. Humanos. Inclusão no mercado de trabalho: Lei de cotas para pessoas com deficiência completa 29 anos, 2020.

[31] S.E.P. e Trabalho. Mais de 79 mil trabalhadores surdos têm carteira assinada no país, 2017.

[32] C.P.I.P com Deficiência. Emprego apoiado cria pontes e promove autonomia, 2020.

[33] A.L. Pizzio, A.R.S. Campelo, P.L.F. Rezende, e R.M. de Quadros. Língua Brasileira de Sinais II. pages 1–37, 2008.

[34] P. Rocco. Frame Extractor for openi2 recordins - pyOniExtractor. *GitHub repository*, 2020.

6 Apêndices

6.1 Gerador dos pontos '+,red' e '.,blue'

```
import numpy as np
import matplotlib
matplotlib.use('TkAgg') #renderiza o gr fico
from matplotlib import pyplot as plt

def cloud_plot(M, t, show_now = True):
  k =  ([t[i] == max(t) for i in range(0, len(t))])
  plt.figure()
  plt.plot(M[k, 0], M[k, 1], '+', color = 'red')
  k = np.where([t[i] == min(t) for i in range(0, len(t))])
  plt.plot(M[k, 0], M[k, 1], '.', color = 'blue')
  if show_now:
    plt.show()

if __name__ == "__main__":
  M = np.array([[-1, -1], [1, 1], [-1, -1]])
  cloud_plot(M, [0, 1, 0])
```

6.2 Gerador da nuvem de pontos com distribuição linear

```
# |--- Imported libraries ---|--------------------------{{{
import numpy as np
from cloud_plot import cloud_plot
#---}}}

# |--- generate_cloud_examples ---|--------------------{{{
def generate_cloud_examples(n):
  n1 = round(n / 2.0)
  n2 = n - n1
  M1 = (np.random.randn(n1, 2) * 0.9) + 1
  t1 = np.ones(shape = [n1,])
```

```
    M2 = (np.random.randn(n2, 2) * 0.9) - 1
    t2 = -np.ones(shape = [n2,])
    M = np.ndarray(shape = [n, 2])
    t = np.ndarray(shape = [n,])
    M[0 : n1, 0 : 2] = M1
    M[n1 : M.shape[0], 0 : 2] = M2
    t[0 : n1,] = t1
    t[n1 : t.shape[0],] = t2
    return M, t
#---}}}
```

6.3 Gerador da nuvem de pontos com distribuição circular

```
# |--- Imported libraries ---|-------------------------{{{
import numpy as np
from cloud_plot import cloud_plot
#---}}}

# |--- generate_cloud_examples ---|-------------------------{{{
def generate_cloud_examples(n, radius = 0.8):
    M = np.random.randn(n, 2)
    R = np.sqrt(M[:, 0] ** 2 + M[:, 1] ** 2)
    locations_positive_class = np.where(R <= radius)
    locations_positive_class = locations_positive_class[0]
    t = np.zeros(shape = (n, )) - 1
    t[locations_positive_class] = +1
    return M, t
#---}}}
```

6.4 Modelo do classificador SVM no exemplo das nuvens pontos

```
# |--- single_session_training_testing_svm ---|-------------------------{{{
def single_session_training_testing_svm(M_training, t_training, M_testing,
    t_testing):
    from sklearn import \texit{SVM}
    s = \texit{SVM}.SVC(kernel = 'rbf')
    s.fit(M_training, t_training)
    results = s.predict(M_testing)
```

6.5 Modelo do classificador CNN no exemplo das nuvens pontos

```
# |--- labels2activationmatrix ---|-------------------------{{{
def labels2activationmatrix(t, possible_labels):
    samples = t.shape[0]
    number_labels = len(possible_labels)
    y = np.zeros(shape = [samples, number_labels])
    for k in range(0, number_labels):
        L = possible_labels[k]
        v = [t[i] == L for i in range(0, len(t))]
        n = np.where(v)
        y[n[0], k] = 1
    return y
#---}}}

# |--- activationmatrix2labels ---|-------------------------{{{
def activationmatrix2labels(y, possible_labels):
    samples = y.shape[0]
    t = np.zeros(shape = [samples,])
    for k in range(0, t.shape[0]):
        M = max(y[k, :])
        v = [y[k, i] == M for i in range(0, y.shape[1])]
        v = np.where(v)
        v = v[0]
        try:
            v = v[0]
        except:
            pass
        t[k] = possible_labels[v]
    return t
#---}}}

# |--- single_session_training_testing_cnn ---|-------------------------{{{
def single_session_training_testing_cnn(M_training, t_training, M_testing,
    t_testing):
    import tensorflow as tf # Ver tamb m o pytorch
    from tensorflow.keras import layers
    from tensorflow.keras import models
    from tensorflow.keras.utils import to_categorical
    # from tensorflow.compat.v1 import ConfigProto
    # from tensorflow.compat.v1 import InteractiveSession
    # config = ConfigProto()
    # config.gpu_options.allow_growth = True
    # session = InteractiveSession(config=config)
    import logging
    logging.getLogger('tensorflow').disabled = True
    M_training_ = np.zeros(shape = [M_training.shape[0], 2, 1])
```

```
M_training_[0 : M_training.shape[0], 0 : 2, 0] = M_training
M_testing_ = np.zeros(shape = [M_testing.shape[0], 2, 1])
M_testing_[0 : M_testing.shape[0], 0 : 2, 0] = M_testing
c = models.Sequential()
c.add(layers.Conv1D(filters = 8, kernel_size = 1, activation='relu',
    input_shape = [2, 1]))
c.add(layers.MaxPooling1D(1))
c.add(layers.Flatten())
c.add(layers.Dense(2, activation='softmax'))
#c.summary()
c.compile(loss='categorical_crossentropy', optimizer='sgd', metrics=['
    accuracy'])
c.fit(M_training_, labels2activationmatrix(t_training, [-1, 1]), \
batch_size = 100, epochs = 500, verbose = 2)
results = c.predict(M_testing_)
results = activationmatrix2labels(results, [-1, 1])
```

6.6 Resultado da classificação das nuvens pontos SVM e CNN por meio das métricas de avaliação

```
k = [results[i] == max(t_training) and t_testing[i] == max(t_training)
    \
for i in range(0, len(t_testing))]
k = np.where(k)
true_positives = len(k[0])
k = [results[i] == min(t_training) and t_testing[i] == min(t_training)
    \
for i in range(0, len(t_testing))]
k = np.where(k)
true_negatives = len(k[0])
k = [results[i] == max(t_training) and t_testing[i] == min(t_training)
    \
for i in range(0, len(t_testing))]
k = np.where(k)
false_positives = len(k[0])
k = [results[i] == min(t_training) and t_testing[i] == max(t_training)
    \
for i in range(0, len(t_testing))]
k = np.where(k)
false_negatives = len(k[0])
k = [t_testing[i] == max(t_training) \
for i in range(0, len(t_testing))]
k = np.where(k)
```

```
    positives = len(k[0])
    k = [t_testing[i] == min(t_training) \
    for i in range(0, len(t_testing))]
    k = np.where(k)
    negatives = len(k[0])
    accuracy = (true_negatives + true_positives) / (positives + negatives)
    sensitivity = true_positives / positives
    if true_positives + false_positives == 0:
        precision = 0
        f1measure = 0
    else:
        precision = (true_positives) / (true_positives + false_positives)
        f1measure = 2.0 / (1.0 / precision + 1.0 / sensitivity)
    return results, accuracy, sensitivity, precision, f1measure
#---}}}

# |--- show_results ---|-------------------------{{{
def show_results(classifier_type, accuracy, sensitivity, precision,
    f1measure):
    print('-------------------------------------------------')
    print('Results using ' + classifier_type + ':')
    print('Accuracy: ' + str(accuracy))
    print('Sensitivity: ' + str(sensitivity))
    print('Precision: ' + str(precision))
    print('F1-score: ' + str(f1measure))
    print('-------------------------------------------------')
#---}}}
```

6.7 main dos programas de ambos classificadores SVM e CNN para o exemplo da nuvem de pontos

```
# |--- main ---|-------------------------{{{
if __name__ == '__main__':
    number_examples_training = 1000
    number_examples_testing = 400
    M_training, t_training = generate_cloud_examples(
        number_examples_training)
    M_testing, t_testing = generate_cloud_examples(number_examples_testing)
    cloud_plot(M_training, t_training, show_now = True)
    results, accuracy, sensitivity, precision, f1measure = \
    single_session_training_testing_svm(M_training, t_training, M_testing,
        t_testing)
    show_results('an \texit{SVM}', accuracy, sensitivity, precision,
```

```
        f1measure)
    results, accuracy, sensitivity, precision, f1measure = \
12      single_session_training_testing_cnn(M_training, t_training, M_testing,
        t_testing)
    show_results('a \texit{CNN}', accuracy, sensitivity, precision,
        f1measure)
14 #---}}}
```

6.8 Estudo de caso *Open Hand - Closed Hand*

```
import numpy as np
2 import matplotlib
matplotlib.use('TkAgg') #renderiza o gr fico
4 from matplotlib import pyplot as plt
from matplotlib.image import imread
6 import os

8 def extract_images(directory, label):
    downsample_factor = 3
10  all_images = os.listdir(directory)
    depth_images = []
12  for k in range(0, len(all_images)):
        filename = all_images[k]
14      k = filename.find('16bit')
        if k > -1:
16          depth_images.append(filename)
    number_images = len(depth_images)
18  x = plt.imread(directory + '/' + depth_images[0])
    x = x[0 : x.shape[0] :downsample_factor, 0 : x.shape[1] :
        downsample_factor]
20  rows    = x.shape[0]
    columns = x.shape[1]
22  M = np.zeros(shape = (number_images, rows, columns, 1))
    for k in range(0, len(depth_images)):
24      x = plt.imread(directory + '/' + depth_images[k])
        x = x[0 : x.shape[0] :downsample_factor, 0 : x.shape[1] :
            downsample_factor]
26      M[k, :, :, 0] = x
    t = np.zeros(shape = (number_images, )) + label
28  return M, t

30 def separate_training_test(M, t, training_proportion):
    number_images = M.shape[0]
32  number_training_images = int(training_proportion * number_images)
```

```
    r = np.random.permutation(number_images)
    M_training = M[r[0 : number_training_images], :, :, :]
    M_test = M[r[number_training_images :], :, :, :]
    t_training = t[r[0 : number_training_images],]
    t_test = t[r[number_training_images :],]
    return M_training, t_training, M_test, t_test

def join(M1, M2, t1, t2):
    M = np.zeros(shape = \
    (M1.shape[0] + M2.shape[0], M1.shape[1], M1. shape[2], M1.shape[3]))
    M[0 : M1.shape[0], :, :, :] = M1
    M[M1.shape[0] :, :, :, :] = M2
    t = np.zeros(shape = (t1.shape[0] + t2.shape[0], ))
    t[0 : t1.shape[0], ] = t1
    t[t1.shape[0] :, ] = t2
    return M, t

def show_example(M, image_number):
    rows = M.shape[1]
    columns = M.shape[2]
    I = np.zeros(shape = (rows, columns))
    I[:, :] = M[image_number, :, :, 0]
    plt.imshow(I)
    plt.show()

# |--- labels2activationmatrix ---|-------------------------{{{
def labels2activationmatrix(t, possible_labels):
    samples = t.shape[0]
    number_labels = len(possible_labels)
    y = np.zeros(shape = [samples, number_labels])
    for k in range(0, number_labels):
        L = possible_labels[k]
        v = [t[i] == L for i in range(0, len(t))]
        n = np.where(v)
        y[n[0], k] = 1
    return y
#---}}}

# |--- activationmatrix2labels ---|-------------------------{{{
def activationmatrix2labels(y, possible_labels):
    samples = y.shape[0]
    t = np.zeros(shape = [samples,])
    for k in range(0, t.shape[0]):
        M = max(y[k, :])
        v = [y[k, i] == M for i in range(0, y.shape[1])]
        v = np.where(v)
        v = v[0]
```

```
80          try:
                v = v[0]
82          except:
                pass
84          t[k] = possible_labels[v]
        return t
86  #---}}}

88  # |--- single_session_training_testing_cnn ---|--------------------------{{{
    def single_session_training_testing_cnn(M_training, t_training, M_testing,
        t_testing):
90      import tensorflow as tf # Ver tamb m o pytorch
        from tensorflow.keras import layers
92      from tensorflow.keras import models
        from tensorflow.keras.utils import to_categorical
94      from tensorflow import optimizers
        # from tensorflow.compat.v1 import ConfigProto
96      # from tensorflow.compat.v1 import InteractiveSession
        # config = ConfigProto()
98      # config.gpu_options.allow_growth = True
        # session = InteractiveSession(config=config)
100     import logging
        rows = M_training.shape[1]
102     columns = M_training.shape[2]
        logging.getLogger('tensorflow').disabled = True
104     c = models.Sequential()
        c.add(layers.Conv2D(filters = 8, kernel_size = [6, 6], activation='relu
            ', input_shape = [rows, columns, 1]))
106     c.add(layers.MaxPooling2D(8))
        c.add(layers.Conv2D(filters = 6, kernel_size = [6, 6], activation='relu
            '))
108     c.add(layers.MaxPooling2D(4))
        c.add(layers.Flatten())
110     c.add(layers.Dense(2, activation='softmax'))
        c.summary()
112     sgd = optimizers.SGD(lr = 0.0100, decay = 1e-6, momentum = 0.9,
            nesterov = True)
        c.compile(loss='categorical_crossentropy', optimizer=sgd, metrics=['
            accuracy'])
114     c.fit(M_training, labels2activationmatrix(t_training, [-1, 1]), \
        batch_size = 200, epochs = 750, verbose = 2)
116     # serialize model to JSON
        model_json = c.to_json()
118     with open("trained_model.json", "w") as json_file:
            json_file.write(model_json)
120     # serialize weights to HDF5
        c.save_weights("trained_model.h5")
```

```
122     print("Saved model to disk.")
        results = c.predict(M_testing)
124     results = activationmatrix2labels(results, [-1, 1])
        k = [results[i] == max(t_training) and t_testing[i] == max(t_training)
            \
126     for i in range(0, len(t_testing))]
        k = np.where(k)
128     true_positives = len(k[0])
        k = [results[i] == min(t_training) and t_testing[i] == min(t_training)
            \
130     for i in range(0, len(t_testing))]
        k = np.where(k)
132     true_negatives = len(k[0])
        k = [results[i] == max(t_training) and t_testing[i] == min(t_training)
            \
134     for i in range(0, len(t_testing))]
        k = np.where(k)
136     false_positives = len(k[0])
        k = [results[i] == min(t_training) and t_testing[i] == max(t_training)
            \
138     for i in range(0, len(t_testing))]
        k = np.where(k)
140     false_negatives = len(k[0])
        k = [t_testing[i] == max(t_training) \
142     for i in range(0, len(t_testing))]
        k = np.where(k)
144     positives = len(k[0])
        k = [t_testing[i] == min(t_training) \
146     for i in range(0, len(t_testing))]
        k = np.where(k)
148     negatives = len(k[0])
        accuracy = (true_negatives + true_positives) / (positives + negatives)
150     sensitivity = true_positives / positives
        precision = (true_positives) / (true_positives + false_positives)
152     f1measure = 2.0 / (1.0 / precision + 1.0 / sensitivity)
        return results, accuracy, sensitivity, precision, f1measure
154 #---}}}

156 if __name__ == '__main__':
        directory_open_hand    = "C:/Users/pc home jan 2020/OneDrive/Documentos
            /2020_1/TCC/bruno_processamento_sinais/DATABASE/LIBRAS_HC_RGBDS
            _2011/LIBRAS_HC_RGBDS_2011/C1_1/C1_1/TESTE_CLASSIFICACAO_OH_CH_TOTAL
            /open_hand"
158     directory_closed_hand  = "C:/Users/pc home jan 2020/OneDrive/Documentos
            /2020_1/TCC/bruno_processamento_sinais/DATABASE/LIBRAS_HC_RGBDS
            _2011/LIBRAS_HC_RGBDS_2011/C1_1/C1_1/TESTE_CLASSIFICACAO_OH_CH_TOTAL
            /closed_hand"
```

```
M_O, t_O = extract_images(directory_open_hand, -1)
M_C, t_C = extract_images(directory_closed_hand, +1)
M_O_training, t_O_training, M_O_test, t_O_test = \
separate_training_test(M_O, t_O, 0.70)
M_C_training, t_C_training, M_C_test, t_C_test = \
separate_training_test(M_C, t_C, 0.70)
examples_training = min((M_O_training.shape[0], M_C_training.shape[0]))
M_O_training = M_O_training[0 : examples_training, :, :, :]
t_O_training = t_O_training[0 : examples_training,]
M_C_training = M_C_training[0 : examples_training, :, :, :]
t_C_training = t_C_training[0 : examples_training,]
examples_test = min((M_O_test.shape[0], M_C_test.shape[0]))
M_O_test = M_O_test[0 : examples_test, :, :, :]
t_O_test = t_O_test[0 : examples_test,]
M_C_test = M_C_test[0 : examples_test, :, :, :]
t_C_test = t_C_test[0 : examples_test,]
M_training, t_training = \
join(M_O_training, M_C_training, t_O_training, t_C_training)
M_test, t_test = join(M_O_test, M_C_test, t_O_test, t_C_test)
# show_example(M_training, 0)
# show_example(M_training, M_training.shape[0] - 1)
results, accuracy, sensitivity, precision, f1measure = \
single_session_training_testing_cnn(M_training, t_training, M_test,
    t_test)
print('accuracy: ' + str(accuracy))
print('sensitivity: ' + str(sensitivity))
print('precision: ' + str(precision))
print('f1measure: ' + str(f1measure))
```

6.9 Teste do modelo do estudo de caso *Open Hand - Closed Hand*

```
from tensorflow.keras.models import model_from_json
import numpy as np
import matplotlib.pyplot as plt

# |--- activationmatrix2labels ---|-------------------------{{{
def activationmatrix2labels(y, possible_labels):
    samples = y.shape[0]
    t = np.zeros(shape = [samples,])
    for k in range(0, t.shape[0]):
        M = max(y[k, :])
        v = [y[k, i] == M for i in range(0, y.shape[1])]
```

```
            v = np.where(v)
            v = v[0]
            try:
                v = v[0]
            except:
                pass
            try:
                t[k] = possible_labels[v]
            except:
                t[k] = -1
        return t
#---}}}

def extract_image(filename):
    downsample_factor = 3
    x = plt.imread(filename)
    x = x[0 : x.shape[0] : downsample_factor, 0 : x.shape[1] :
        downsample_factor]
    y = np.zeros(shape = (1, x.shape[0], x.shape[1], 1))
    y[0, :, :, 0] = x
    return y

if __name__ == '__main__':
    json_file = open("C:/Users/pc home jan 2020/OneDrive/Documentos/2020_1/
        TCC/bruno_processamento_sinais/DATABASE/LIBRAS-HC-RGBDS-2011/LIBRAS-
        HC-RGBDS-2011/C1_1/C1_1/TESTE_CLASSIFICACAO_OH_CH_TOTAL/
        trained_model.json", 'r')
    loaded_model_json = json_file.read()
    json_file.close()
    c = model_from_json(loaded_model_json)
    c.load_weights("C:/Users/pc home jan 2020/OneDrive/Documentos/2020_1/
        TCC/bruno_processamento_sinais/DATABASE/LIBRAS-HC-RGBDS-2011/LIBRAS-
        HC-RGBDS-2011/C1_1/C1_1/TESTE_CLASSIFICACAO_OH_CH_TOTAL/
        trained_model.h5")
    x1 = extract_image("C:/Users/pc home jan 2020/OneDrive/Documentos/2020
        _1/TCC/bruno_processamento_sinais/DATABASE/LIBRAS-HC-RGBDS-2011/
        LIBRAS-HC-RGBDS-2011/C1_1/C1_1/TESTE_CLASSIFICACAO_OH_CH_TOTAL/
        example1.png")
    x2 = extract_image("C:/Users/pc home jan 2020/OneDrive/Documentos/2020
        _1/TCC/bruno_processamento_sinais/DATABASE/LIBRAS-HC-RGBDS-2011/
        LIBRAS-HC-RGBDS-2011/C1_1/C1_1/TESTE_CLASSIFICACAO_OH_CH_TOTAL/
        example2.png")
    x3 = extract_image("C:/Users/pc home jan 2020/OneDrive/Documentos/2020
        _1/TCC/bruno_processamento_sinais/DATABASE/LIBRAS-HC-RGBDS-2011/
        LIBRAS-HC-RGBDS-2011/C1_1/C1_1/TESTE_CLASSIFICACAO_OH_CH_TOTAL/
        example3.png")
    x4 = extract_image("C:/Users/pc home jan 2020/OneDrive/Documentos/2020
```

```
    _1/TCC/bruno_processamento_sinais/DATABASE/LIBRAS_HC_RGBDS_2011/
    LIBRAS_HC_RGBDS_2011/C1_1/C1_1/TESTE_CLASSIFICACAO_OH_CH_TOTAL/
    example4.png")
x5 = extract_image("C:/Users/pc home jan 2020/OneDrive/Documentos/2020
    _1/TCC/bruno_processamento_sinais/DATABASE/LIBRAS_HC_RGBDS_2011/
    LIBRAS_HC_RGBDS_2011/C1_1/C1_1/TESTE_CLASSIFICACAO_OH_CH_TOTAL/
    example5.png")
x6 = extract_image("C:/Users/pc home jan 2020/OneDrive/Documentos/2020
    _1/TCC/bruno_processamento_sinais/DATABASE/LIBRAS_HC_RGBDS_2011/
    LIBRAS_HC_RGBDS_2011/C1_1/C1_1/TESTE_CLASSIFICACAO_OH_CH_TOTAL/
    example6.png")
x7 = extract_image("C:/Users/pc home jan 2020/OneDrive/Documentos/2020
    _1/TCC/bruno_processamento_sinais/DATABASE/LIBRAS_HC_RGBDS_2011/
    LIBRAS_HC_RGBDS_2011/C1_1/C1_1/TESTE_CLASSIFICACAO_OH_CH_TOTAL/
    example7.png")
x8 = extract_image("C:/Users/pc home jan 2020/OneDrive/Documentos/2020
    _1/TCC/bruno_processamento_sinais/DATABASE/LIBRAS_HC_RGBDS_2011/
    LIBRAS_HC_RGBDS_2011/C1_1/C1_1/TESTE_CLASSIFICACAO_OH_CH_TOTAL/
    example8.png")

result1 = activationmatrix2labels(c.predict(x1), [-1, 1])
result2 = activationmatrix2labels(c.predict(x2), [-1, 1])
result3 = activationmatrix2labels(c.predict(x3), [-1, 1])
result4 = activationmatrix2labels(c.predict(x4), [-1, 1])
result5 = activationmatrix2labels(c.predict(x5), [-1, 1])
result6 = activationmatrix2labels(c.predict(x6), [-1, 1])
result7 = activationmatrix2labels(c.predict(x7), [-1, 1])
result8 = activationmatrix2labels(c.predict(x8), [-1, 1])

if result1 == -1:
    result1 = 'open hand'
else:
    result1 = 'closed hand'
if result2 == -1:
    result2 = 'open hand'
else:
    result2 = 'closed hand'
if result3 == -1:
    result3 = 'open hand'
else:
    result3 = 'closed hand'
if result4 == -1:
    result4 = 'open hand'
else:
    result4 = 'closed hand'
if result5 == -1:
    result5 = 'open hand'
```

```
    else:
        result5 = 'closed hand'
    if result6 == -1:
        result6 = 'open hand'
    else:
        result6 = 'closed hand'
    if result7 == -1:
        result7 = 'open hand'
    else:
        result7 = 'closed hand'
    if result8 == -1:
        result8 = 'open hand'
    else:
        result8 = 'closed hand'


    print('Result for image1: ' + result1)
    print('Result for image2: ' + result2)
    print('Result for image3: ' + result3)
    print('Result for image1: ' + result4)
    print('Result for image2: ' + result5)
    print('Result for image3: ' + result6)
    print('Result for image1: ' + result7)
    print('Result for image2: ' + result8)
```

6.10 Aplicação de Data Augmentation no dataset *Open Hand* e *Closed Hand* e geração de vídeos de 30 frames

```
# -----| Imported libraries
    |-------------------------------------------------------------------------{{{
from os import walk
import cv2
import numpy as np
from scipy import ndimage
# ---}}}

# -----| is_image_type
    |-------------------------------------------------------------------------{{{
def is_image_type(filename, desired_image_type):
    t = filename[::-1]
    k = t.find('.')
    t = t[k + 1 :]
    k = t.find('_')
```

```
14      t = t[: k]
        t = t[::-1]
16      if t == desired_image_type:
            i = True
18      else:
            i = False
20      return i
    # ---}}}
22
    # -----| extract_image_number
        |-------------------------------------------------------------------------{{{
24  def extract_image_number(filename):
        k = filename.find('_')
26      n = filename[: k]
        n = int(n)
28      return n
    # ---}}}
30
    # -----| list_ordered_files
        |-------------------------------------------------------------------------{{{
32  def list_ordered_files(root, files, desired_image_type):
        F = [''] * len(files)
34      added_images = 0
        for k in range(0, len(files)):
36          filename = files[k]
            if is_image_type(filename, desired_image_type):
38              n = extract_image_number(filename)
                F[n - 1] = root + '/' + files[k]
40              added_images += 1
        F = F[: added_images]
42      return F
    # ---}}}
44
    # -----| extract_data_info
        |-------------------------------------------------------------------------{{{
46  def extract_data_info(root):
        initial_part = 'dataset_original/'
48      d = root[len(initial_part) :]
        participant = int(d[1])
50      mode = int(d[3])
        d = d[10 :]
52      hand_configuration = int(d)
        return participant, mode, hand_configuration
54  # ---}}}

56  # -----| select_components
        |-------------------------------------------------------------------------{{{
```

```
def select_components(F, indices):
    G = []
    for k in range(0, len(indices)):
        i = int(indices[k])
        if i < 0:
            i = 0
        if i > len(F) - 1:
            i = len(F) - 1
        G.append(F[i])
    return G
# ---}}}

# -----| apply_rotation
    |-------------------------------------------------------------------------{{{
def apply_rotation(x, rotation):
    y = ndimage.rotate(x, rotation, reshape = False)
    return y
# ---}}}

# -----| apply_translation
    |-------------------------------------------------------------------------{{{
def apply_translation(x, translation):
    y = ndimage.shift(x, translation)
    return y
# ---}}}


# -----| apply_blur
    |-------------------------------------------------------------------------{{{
def apply_blur(x,blurlevel):
    y = ndimage.gaussian_filter(x,blurlevel)
    return y
# ---}}}

# -----| save_video
    |-------------------------------------------------------------------------{{{
import cv2
def save_video(filename, F, rotation, translation, blurlevel, number_frames
    = 30):
    video_duration_seconds = 4.0
    number_frames_per_second = number_frames / video_duration_seconds
    indices = np.linspace(0, len(F), number_frames)
    F = select_components(F, indices)
    img_array = []
    for k in range(0, number_frames):
        img = cv2.imread(F[k])
        height, width, layers = img.shape
```

```
            size = (width, height)
            img = apply_rotation(img, rotation)
            img = apply_translation(img, translation)
            img = apply_blur(img, blurlevel)
            img_array.append(img)
        out = cv2.VideoWriter(filename+str('.mp4'), cv2.VideoWriter_fourcc('D',
            'I','V','X'), number_frames_per_second, size)
        for i in range(len(img_array)):
            out.write(img_array[i])
        out.release()
# ---}}}

# -----| generate_video
    |-------------------------------------------------------------------------{{{
def generate_video(F, participant, mode, hand_configuration, rotation,
    translation, blurlevel):
    output_directory = 'dataset_organizado_com_data_augmentation/' + '
        configuracao' + str(hand_configuration) + '/'
    output_file = 'configuracao' + str(hand_configuration) + \
    '_participante' + str(participant) + \
    '_modo' + str(mode) + \
    '_rotation' + str(rotation) + \
    '_translation' + str(translation) + \
    '_blurlevel' + str(blurlevel) + \
    '.mp4'
    save_video(output_directory + output_file, F, rotation, translation,
        blurlevel)
# ---}}}

# -----| process_all_files
    |-------------------------------------------------------------------------{{{
def process_all_files(root, files, desired_image_type, rotation,
    translation, blurlevel):
    F = list_ordered_files(root, files, desired_image_type)
    participant, mode, hand_configuration = extract_data_info(root)
    generate_video(F, participant, mode, hand_configuration, rotation,
        translation, blurlevel)
# ---}}}

# -----| process_all_dirs
    |-------------------------------------------------------------------------{{{
def process_all_dirs(desired_image_type, rotation = 0.0, translation = 0,
    blurlevel = 0.0):
    rootdir = 'dataset_original'
    for root, subdirs, files in walk(rootdir):
        if len(files) > 1:
            process_all_files(root, files, desired_image_type, rotation,
```

```
                translation, blurlevel)
# ---}}}

# -----| main
    ├──────────────────────────────────────────────────────────────────────{{{
if __name__ == '__main__':
    process_all_dirs('color')
    process_all_dirs('color', rotation = 1.0)
    process_all_dirs('color', translation = 1)
    process_all_dirs('color', translation = 2)
    process_all_dirs('color', blurlevel = 1.0)
    process_all_dirs('color', rotation = 1.0, translation= 1)
    process_all_dirs('color', rotation = 5.0, translation= 1)
    process_all_dirs('color', rotation = 5.0, translation= 2)
    process_all_dirs('color', rotation= 1.0, blurlevel = 1.0)
    process_all_dirs('color', rotation= 1.0, blurlevel = 4.0)
    process_all_dirs('color', rotation= 2.0, blurlevel = 5.0)
    process_all_dirs('color', rotation = 1.0, translation= 1, blurlevel =
        4.0)
    process_all_dirs('color', rotation = 1.0, translation= 1, blurlevel =
        5.0)
    process_all_dirs('color', rotation = 1.0, translation= 1, blurlevel =
        1.0)
    process_all_dirs('color', rotation= -5.0, blurlevel = 1.0)
    process_all_dirs('color', rotation= -1.0, blurlevel = 2.0)
    process_all_dirs('color', rotation= -2.0, blurlevel = 2.0)
    process_all_dirs('color', rotation = -2.0, translation= 2, blurlevel =
        2.0)
    process_all_dirs('color', rotation = -5.0, translation= 1, blurlevel =
        1.0)
    process_all_dirs('color', rotation = -3.0, translation= 2, blurlevel =
        1.0)
    process_all_dirs('color', rotation = -3.0, translation= 2, blurlevel =
        2.0)
    process_all_dirs('color', rotation = -3.0, translation= 2, blurlevel =
        3.0)
    process_all_dirs('color', rotation = -3.0, translation= 2, blurlevel =
        4.0)
    process_all_dirs('color', rotation = -3.0, translation= 2, blurlevel =
        5.0)
    process_all_dirs('color', rotation= -1.0, translation= 2)
    process_all_dirs('color', rotation= -4.0, translation= 2)
    process_all_dirs('color', rotation= -5.0, translation= 2)
# ---}}}
```

6.11 Funções implementadas para o Data Augmentation

```
# -----| apply_rotation
    |-------------------------------------------------------------------------{{{
def apply_rotation(x, rotation):
    y = ndimage.rotate(x, rotation, reshape = False)
    return y
# ---}}}

# -----| apply_translation
    |-------------------------------------------------------------------------{{{
def apply_translation(x, translation):
    y = ndimage.shift(x, translation)
    return y
# ---}}}


# -----| apply_blur
    |-------------------------------------------------------------------------{{{
def apply_blur(x,blurlevel):
    y = ndimage.gaussian_filter(x,blurlevel)
    return y
# ---}}}
```

6.12 Treinamento do classificador de vídeos das configurações de mão

```
#!/usr/bin/env ipython

# -----| Imported libraries
    |-------------------------------------------------------------------------{{{
import numpy as np
from matplotlib import pyplot as plt
# import matplotlib
# matplotlib.use('Qt5Agg')
from scipy.io import savemat, loadmat
import cv2
import os
from os import walk
from os.path import exists as file_exists
# ---}}}

# -----| Global variables
```

```
    |-------------------------------------------------------------------------{{{
16 dataset_dir = 'dataset_organizado_com_data_augmentation/'
  # ---}}}
18
  # -----| join
    |-------------------------------------------------------------------------{{{
20 def join(M1, M2, t1, t2):
      M = np.zeros(shape = \
22     (M1.shape[0] + M2.shape[0], M1.shape[1], M1. shape[2], M1.shape[3]))
      M[0 : M1.shape[0], :, :, :] = M1
24    M[M1.shape[0] :, :, :, :] = M2
      t = np.zeros(shape = (t1.shape[0] + t2.shape[0], ))
26    t[0 : t1.shape[0], ] = t1
      t[t1.shape[0] :, ] = t2
28    return M, t
  # ---}}}
30
  # |--- labels2activationmatrix ---|-------------------------{{{
32 def labels2activationmatrix(t, possible_labels):
      samples = t.shape[0]
34    number_labels = len(possible_labels)
      y = np.zeros(shape = [samples, number_labels])
36    for k in range(0, number_labels):
          L = possible_labels[k]
38        v = [t[i] == L for i in range(0, len(t))]
          n = np.where(v)
40        y[n[0], k] = 1
      return y
42 #---}}}

44 # |--- activationmatrix2labels ---|-------------------------{{{
  def activationmatrix2labels(y, possible_labels):
46    samples = y.shape[0]
      t = np.zeros(shape = [samples,])
48    for k in range(0, t.shape[0]):
          M = max(y[k, :])
50        v = [y[k, i] == M for i in range(0, y.shape[1])]
          v = np.where(v)
52        v = v[0]
          try:
54            v = v[0]
          except:
56            pass
          t[k] = possible_labels[v]
58    return t
  #---}}}
60
```

```
# |--- single_session_training_testing_cnn ---|-------------------------{{{
def single_session_training_testing_cnn(M_training, t_training, M_testing,
    t_testing):
    import tensorflow as tf # Ver tamb m o pytorch
    from tensorflow.keras import layers
    from tensorflow.keras import models
    from tensorflow.keras.utils import to_categorical
    from tensorflow import optimizers
    # from tensorflow.compat.v1 import ConfigProto
    # from tensorflow.compat.v1 import InteractiveSession
    # config = ConfigProto()
    # config.gpu_options.allow_growth = True
    # session = InteractiveSession(config=config)
    import logging
    rows = M_training.shape[1]
    columns = M_training.shape[2]
    number_frames = M_training.shape[3]
    logging.getLogger('tensorflow').disabled = True
    c = models.Sequential()
    c.add(layers.Conv2D(filters = 4, kernel_size = [3, 3], activation='relu
        ', \
    input_shape = [rows, columns, number_frames]))
    c.add(layers.MaxPooling2D(2))
    c.add(layers.Conv2D(filters = 2, kernel_size = [3, 3], activation='relu
        '))
    c.add(layers.MaxPooling2D(2))
    c.add(layers.Flatten())
    c.add(layers.Dense(2, activation='softmax'))
    c.summary()
    sgd = optimizers.SGD(lr = 0.010, decay = 1e-6, momentum = 0.9, nesterov
        = True)
    c.compile(loss='categorical_crossentropy', optimizer=sgd, metrics=['
        accuracy'])
    history = c.fit(M_training, labels2activationmatrix(t_training, [0, 1])
        , \
    #batch_size = 200, epochs = 750, verbose = 2)
    batch_size = 100, epochs = 149, verbose = 2, validation_split  = 0.2)
    # ##########################################################################33
    # Training history visualization
    # plt.plot(history.history['accuracy'], label = 'Training', linewidth =
        1.2)
    # plt.plot(history.history['val_accuracy'], label = 'Validation',
        linewidth = 1.2)
    # plt.xlabel('Epoch')
    # plt.ylabel('Accuracy')
    # plt.legend(loc="upper left")
    # plt.show()
```

```
100     # plt.plot(history.history['loss'], label = 'Training', linewidth =
            1.2)
        # plt.plot(history.history['val_loss'], label = 'Validation', linewidth
            = 1.2)
102     # plt.xlabel('Epoch')
        # plt.ylabel('Loss function')
104     # plt.legend(loc="upper left")
        # plt.show()
106     # ##########################################################################33
        # serialize model to JSON
108     model_json = c.to_json()
        with open("trained_model.json", "w") as json_file:
110         json_file.write(model_json)
        # serialize weights to HDF5
112     c.save_weights("trained_model.h5")
        print("Saved model to disk.")
114     results = c.predict(M_testing)
        results = activationmatrix2labels(results, [0, 1])
116     k = [results[i] == max(t_training) and t_testing[i] == max(t_training)
            \
        for i in range(0, len(t_testing))]
118     k = np.where(k)
        true_positives = len(k[0])
120     k = [results[i] == min(t_training) and t_testing[i] == min(t_training)
            \
        for i in range(0, len(t_testing))]
122     k = np.where(k)
        true_negatives = len(k[0])
124     k = [results[i] == max(t_training) and t_testing[i] == min(t_training)
            \
        for i in range(0, len(t_testing))]
126     k = np.where(k)
        false_positives = len(k[0])
128     k = [results[i] == min(t_training) and t_testing[i] == max(t_training)
            \
        for i in range(0, len(t_testing))]
130     k = np.where(k)
        false_negatives = len(k[0])
132     k = [t_testing[i] == max(t_training) \
        for i in range(0, len(t_testing))]
134     k = np.where(k)
        positives = len(k[0])
136     k = [t_testing[i] == min(t_training) \
        for i in range(0, len(t_testing))]
138     k = np.where(k)
        negatives = len(k[0])
140     try:
```

```
            accuracy = (true_negatives + true_positives) / (positives +
                negatives)
142     except:
            accuracy = 0
144     try:
            sensitivity = true_positives / positives
146     except:
            sensitivity = 0
148     try:
            precision = (true_positives) / (true_positives + false_positives)
150     except:
            precision = 0
152     try:
            f1measure = 2.0 / (1.0 / precision + 1.0 / sensitivity)
154     except:
            f1measure = 0
156     return results, accuracy, sensitivity, precision, f1measure
#---}}}
158
# ----| load_video
    |-------------------------------------------------------------------------{{{
160 def load_video(filename): #, max_frames = 1000
    \texit{frames} = []
162     filename1 = filename
    filename2 = filename + '.mp4'
164     filename3 = filename[0 : len(filename) - 4]
    if not(file_exists(filename)):
166         filename = filename2
        if not(file_exists(filename)):
168             filename = filename3
    cap = cv2.VideoCapture(filename) # F is now in a VideoCapture data type
170     # Check if camera opened successfully
    if (cap.isOpened()== False):
172         print("Error opening video stream or file")
    # Read until video is completed
174     while(cap.isOpened() and len(frames)<30):
        # Capture \texit{frame}-by-frame
176         ret, \texit{frame} = cap.read()
        if ret == True:
178             \texit{frame} = \texit{frame}[::8,::8]
            \texit{frames}.append(frame)
180         # Press Q on keyboard to  exit
        #if cv2.waitKey(25) & 0xFF == ord('q'):
182         #    break
      # Break the loop
184         #else:
        #    break
```

```
186     # When everything done, release the video capture object
        cap.release()
188     rows = \texit{frames}[0].shape[0]
        columns = \texit{frames}[0].shape[1]
190     F = np.zeros(shape = (rows, columns, len(frames)))
        for k in range(0, len(frames)):
192         I = \texit{frames}[k]
            gray = cv2.cvtColor(I, cv2.COLOR_BGR2GRAY)
194         F[0 : rows, 0 : columns, k] = gray
        return F
196 # ---}}}

198 # -----| extract_extension()
    |-------------------------------------------------------------------------{{{
  def extract_extension(filename):
200     e = ''
        filename = filename[: : -1]
202     k = filename.find('.')
        if k > -1:
204         e = filename[: k]
            e = e[: : -1]
206     return e
  # ---}}}
208
  # -----| list_all_files
    |-------------------------------------------------------------------------{{{
210 def list_all_files(extension):
        directories = []
212     filenames = []
        for root, subdirs, files in walk(dataset_dir):
214         if len(files) > 1:
                for k in range(0, len(files)):
216                 if extract_extension(files[k]) == extension:
                        filenames.append(files[k])
218                     directories.append(root + '/')
        return filenames, directories
220 # ---}}}

222 # -----| participant_from_filename
    |-------------------------------------------------------------------------{{{
  def participant_from_filename(filename):
224     # configuracao0_participante1_modo1_frameshift0_rotation0.0
            _translation0_blurlevel0.0.mp4
        p = 0
226     string_identifier = '_participante'
        k = filename.find(string_identifier)
228     if k > -1:
```

```
            s = filename[k + len(string_identifier) :]
            k = s.find('_')
            if k > -1:
                p = s[: k]
                p = int(p)
    return p
# ---}}}

# -----| participant_from_filename
    |-------------------------------------------------------------------------{{{
def class_from_filename(filename, configurations):
    # configuracao0_participante1_modo1_frameshift0_rotation0.0
        _translation0_blurlevel0.0.mp4
    c = 0
    string_identifier = 'configuracao'
    k = filename.find(string_identifier)
    if k > -1:
        s = filename[k + len(string_identifier) :]
        k = s.find('_')
        if k > -1:
            c = s[: k]
            c = int(c)
            c = np.where(np.array(configurations) == c)[0][0]
    return c
# ---}}}

# -----| extract_training_test_sets_from_videos
    |-------------------------------------------------------------------------{{{
def extract_training_testing_sets_from_videos(configurations = [0, 56],
    training_participants = [1, 6, 2, 3], testing_participants = [4, 5]):
    x = load_video(dataset_dir + 'configuracao0/' + \
    'configuracao0_participante1_modo1_frameshift0_rotation0.0
        _translation0_blurlevel0.0.mp4')
    rows = x.shape[0]
    columns = x.shape[1]
    number_frames = x.shape[2]
    list_all_video_files, directories = list_all_files('mp4')
    M_training = np.zeros(shape = (len(list_all_video_files), rows, columns
        , number_frames))
    M_testing = np.zeros(shape = (len(list_all_video_files), rows, columns,
         number_frames))
    t_training = np.zeros(shape = (len(list_all_video_files), 1))
    t_testing = np.zeros(shape = (len(list_all_video_files), 1))
    N_training = 0
    N_testing = 0
    for k in range(0,len(list_all_video_files)):
        filename = list_all_video_files[k]
```

```
        x = load_video(directories[k] + list_all_video_files[k])
        if participant_from_filename(filename) in training_participants:
            M_training[N_training, 0 : rows, 0 : columns, 0 : number_frames
                ] = x
            t_training[N_training, 0] = class_from_filename(filename,
                configurations)
            N_training += 1
        if participant_from_filename(filename) in testing_participants:
            M_testing[N_testing, 0 : rows, 0 : columns, 0 : number_frames]
                = x
            t_testing[N_testing, 0] = class_from_filename(filename,
                configurations)
            N_testing += 1
        if np.mod(k, 100) == 0:
            print('Analyzed ' + str(k + 1) + ' video(s) of ' + str(len(
                list_all_video_files)) + '.')
    M_training = M_training[0 : N_training, :, :, :]
    M_testing = M_testing[0 : N_testing, :, :, :]
    t_training = t_training[0 : N_training, :]
    t_testing = t_testing[0 : N_testing, :]
    return M_training, t_training, M_testing, t_testing
# ---}}}

# -----| save_training_testing
    |-------------------------------------------------------------------------{{{
def save_training_testing_tensors(filename, M_training, t_training,
    M_testing, t_testing):
    d = {'M_training': M_training, 't_training' : t_training, 'M_testing' :
        M_testing, 't_testing' : t_testing}
    savemat(filename,d)
# ---}}}

# -----| load_training_testing
    |-------------------------------------------------------------------------{{{
def load_training_testing_tensors(filename):
    d = loadmat(filename)
    M_training = d['M_training']
    M_testing = d['M_testing']
    t_training = d['t_training']
    t_testing = d['t_testing']
    return M_training, t_training, M_testing, t_testing
# ---}}}


# -----| main
    |-------------------------------------------------------------------------{{{
if __name__ == '__main__':
```

```
306     # M_training, t_training, M_testing, t_testing =
            extract_training_testing_sets_from_videos()
        #save_training_testing_tensors('
            time_undersample4_training_test_new_set_1.mat', M_training,\
308     # t_training, M_testing, t_testing )
        M_training, t_training, M_testing, t_testing = \
310     load_training_testing_tensors('
            time_undersample4_training_test_new_set_1.mat')
        # Training and test session:
312     results, accuracy, sensitivity, precision, f1measure = \
        single_session_training_testing_cnn(M_training, t_training, M_testing,
            t_testing)
314     print('accuracy: ' + str(accuracy))
        print('sensitivity: ' + str(sensitivity))
316     print('precision: ' + str(precision))
        print('f1measure: ' + str(f1measure))
318 # ---}}}
```

7 Anexos

7.1 Extração dos frames dos vídeos

```
import os
import sys
import cv2
from openni import openni2
import argparse
import numpy as np

def openDevice(video_path):
    try:
        if sys.platform == "win32":
            libpath = "lib/Windows"
        else:
            libpath = "lib/Linux"
        openni2.initialize(libpath)
        dev = openni2.Device.open_file(video_path)
        pbs = openni2.PlaybackSupport(dev)

        pbs.set_repeat_enabled(True)
        pbs.set_speed(-1.0)

        return dev, pbs
    except Exception as ex:
        print(ex)
        raise Exception("Initialization Error")

def processDepth(dev,pbs,interval,dst):
    try:
        depth_stream = dev.create_depth_stream()
        depth_stream.start()
        depth_scale_factor = 255.0 / (650.0-520.0)
        depth_scale_beta_factor = -520.0*255.0/(650.0-520.0)
        nframes = depth_stream.get_number_of_frames()
        print("Depth \texit{frames}: " + str(nframes))
        with open('timestampsdepth.txt', 'w') as tfile:
```

```
            for i in range(1, nframes+1, interval):
36              try:
                    if i != nframes:
38                      pbs.seek(depth_stream, i)    #Seek hangs if last \
                            texit{frame} is empty, sometimes happens
                except Exception as ex:
40                  print("Error on depth stream seek ", ex)
                    continue
42              s = openni2.wait_for_any_stream([depth_stream], 2)
                if s != depth_stream:
44                  print("Error on depth stream, timeout reached reading \
                        texit{frame} n: ", str(i))
                    continue
46              \texit{frame}_depth = depth_stream.read_frame()
                \texit{frame}_depth_data = \texit{frame}_depth.
                    get_buffer_as_uint16()
48              depth_array = np.ndarray((frame_depth.height, \texit{frame}
                    _depth.width), dtype=np.uint16,
                                         buffer=frame_depth_data)
50              depth_uint8 = depth_array*depth_scale_factor+
                    depth_scale_beta_factor
                depth_uint8[depth_uint8 > 255] = 255
52              depth_uint8[depth_uint8 < 0] = 0
                depth_uint8 = depth_uint8.astype('uint8')
54              cv2.imwrite(dst + "/" + str(frame_depth.frameIndex) + "
                    _16bit.png", depth_array)
                cv2.imwrite(dst + "/" + str(frame_depth.frameIndex) + "
                    _8bit.png", depth_uint8)
56              tfile.write(str(frame_depth.frameIndex) + ';' + str(
                    frame_depth.timestamp) + '\n')
        depth_stream.close()
58      print("All depth \texit{frames} extracted")
    except Exception as ex:
60      print(ex)

62 def processColor(dev,pbs,interval,dst):
    try:
64      color_stream = dev.create_color_stream()
        color_stream.start()
66      nframes = color_stream.get_number_of_frames()
        print("Color \texit{frames}: " + str(nframes))
68      with open('timestampscolor.txt', 'w') as tfile:
            for i in range(1, nframes+1, interval):
70              try:
                    if i != nframes:
72                      pbs.seek(color_stream, i)  # Seek hangs if last \
                            texit{frame} is empty, sometimes happens
```

```
                except Exception as ex:
74                  print("Error on color stream seek ", ex)
                    continue
76              s = openni2.wait_for_any_stream([color_stream], 2)
                if s != color_stream:
78                  print("Error on color stream, timeout reached reading \
                        texit{frame} n: ", str(i))
                    continue
80              \texit{frame}_color = color_stream.read_frame()
                \texit{frame}_color_data = \texit{frame}_color.
                    get_buffer_as_uint8()
82              color_array = np.ndarray((frame_color.height, \texit{frame}
                    _color.width, 3), dtype=np.uint8,
                                         buffer=frame_color_data)
84              color_array = cv2.cvtColor(color_array, cv2.COLOR_BGR2RGB)
                cv2.imwrite(dst + "/" + str(frame_color.frameIndex) + "
                    _color.png", color_array)
86              tfile.write(str(frame_color.frameIndex) + ';' + str(
                    frame_color.timestamp) + '\n')
        color_stream.close()
88      print("All color \texit{frames} extracted")
    except Exception as ex:
90      print(ex)

92 def main():
    p = argparse.ArgumentParser(formatter_class=argparse.
        RawDescriptionHelpFormatter, description="")
94  p.add_argument('--v', dest='video_path', action='store', required=True,
        help='path Video')
    p.add_argument('--d', dest='dst', action='store', default='img', help='
        Destination Folder')
96  p.add_argument('--i', dest='interval', action='store', default=1, help=
        'Interval')
    args = p.parse_args()
98  interval = int(args.interval)
    dst = args.dst
100 if not os.path.exists(dst):
        os.mkdir(dst)
102     print("Directory ", dst, " Created ")
    try:
104     dev, pbs = openDevice(args.video_path.encode('utf-8'))
        if dev.has_sensor(openni2.SENSOR_COLOR):
106         print("Color Stream found")
            processColor(dev, pbs, interval, dst)
108     if dev.has_sensor(openni2.SENSOR_DEPTH):
            print("Depth Stream found")
110         processDepth(dev, pbs, interval, dst)
```

```
            print("Done!")
        except Exception as ex:
            print(ex)
        openni2.unload()

if __name__ == '__main__':
    main()
```

1

[1]Fonte: [34]