

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE COMPUTAÇÃO
CURSO DE ENGENHARIA DE COMPUTAÇÃO

ANDERSON MAKOTO SHINODA

**VISÃO COMPUTACIONAL APLICADA AO RECONHECIMENTO
DE ACORDES DE VIOLÃO**

TRABALHO DE CONCLUSÃO DE CURSO

CORNÉLIO PROCÓPIO
2021

ANDERSON MAKOTO SHINODA

VISÃO COMPUTACIONAL APLICADA AO RECONHECIMENTO DE ACORDES DE VIOLÃO

Trabalho de Conclusão de Curso apresentado ao Curso de Engenharia de Computação da Universidade Tecnológica Federal do Paraná, como requisito parcial para a obtenção do título de Bacharel em Engenharia de Computação.

Orientador: Claiton de Oliveira
Universidade Tecnológica Federal do Paraná

CORNÉLIO PROCÓPIO
2021



4.0 Internacional

Esta licença permite remixe, adaptação e criação a partir do trabalho para fins não comerciais, desde que sejam atribuídos créditos ao(s) autor(es) e que licenciem as novas criações sob termos idênticos. Conteúdos elaborados por terceiros, citados e referenciados nesta obra não são cobertos pela licença.



Ministério da Educação
Universidade Tecnológica Federal do Paraná
Câmpus Cornélio Procópio
Nome da Diretoria
Nome da Coordenação
Nome do Curso



TERMO DE APROVAÇÃO

Visão Computacional Aplicada ao Reconhecimento de Acordes de Violão

por

Anderson Makoto Shinoda

Este Trabalho de Conclusão de Curso de graduação foi julgado adequado para obtenção do Título de “Bacharel em Engenharia de Computação” e aprovado em sua forma final pelo Programa de Graduação em Engenharia de Computação da Universidade Tecnológica Federal do Paraná.

Cornélio Procópio, 03/05/2021

Prof. Dr. Claiton de Oliveira

Prof. Dr. Cléber Gimenez Corrêa

Prof. Dr. Silvio Ricardo Rodrigues Sanches

“A Folha de Aprovação assinada encontra-se na Coordenação do Curso”

Dedico este trabalho, primeiramente, a toda minha família e amigos, que me ajudaram por todo este tempo na Universidade, gostaria de agradecer também ao meu orientador, Clayton de Oliveira, por todo o auxílio durante o desenvolvimento deste projeto.

AGRADECIMENTOS

Eu gostaria de agradecer a todos amigos e professores que me acompanharam durante a faculdade, e principalmente à minha família que sempre me apoiou em minhas escolhas. Por fim, gostaria de agradecer à instituição UTFPR por fornecer toda a estrutura necessária para minha formação.

Eu denomino meu campo de Gestão do Conhecimento, mas você não pode gerenciar conhecimento. Ninguém pode. O que pode fazer - o que a empresa pode fazer - é gerenciar o ambiente que otimize o conhecimento. (PRUSAK, Laurence, 1997).

RESUMO

SHINODA, Anderson. Visão Computacional Aplicada ao Reconhecimento de Acordes de Violão. 2021. 29 f. Trabalho de Conclusão de Curso – Curso de Engenharia de Computação, Universidade Tecnológica Federal do Paraná. Cornélio Procópio, 2021.

A aplicação de métodos de visão computacional permite o reconhecimento de padrões presentes em imagens digitais através de técnicas de processamento de imagens, extração de características e classificação utilizando inteligência artificial. No caso deste trabalho, estes métodos foram utilizados para a implementação e aplicação de métodos de visão computacional para o reconhecimento de acordes de violão a partir de imagens. Para isso, foram implementadas técnicas de otimização de dados, segmentação automática e classificação com base em *deep learning*. Os resultados obtidos apresentaram 67% como sendo o maior valor de acurácia utilizando *deep learning*.

Palavras-chave: Visão Computacional. Aprendizado de Máquina. Acordes de Violão. *Deep Learning*.

ABSTRACT

SHINODA, Anderson. Computer Vision Applied to Recognition of Guitar Chords. 2021. [29](#)
f. Trabalho de Conclusão de Curso – Curso de Engenharia de Computação, Universidade
Tecnológica Federal do Paraná. Cornélio Procópio, 2021.

The application of computer vision methods allows the recognition of patterns present in digital images through image processing techniques, feature extraction and classification using artificial intelligence. In the case of this work, these methods were used for the implementation and application of computer vision methods for the recognition of guitar chords from images. For that, data optimization techniques, automatic segmentation techniques will be implemented and classified based on deep learning. The results obtained showed 67% as the highest accuracy level using deep learning.

Keywords: Image Processing. Machine Learning. Guitar Chords. Deep Learning.

LISTA DE FIGURAS

Figura 1 – Fluxograma da Visão Computacional	3
Figura 2 – Fluxograma do modelo <i>bagging</i>	9
Figura 3 – Representação do modelo <i>stacking</i>	9
Figura 4 – Estrutura da rede <i>U-net</i>	12
Figura 5 – Estrutura da rede <i>Linknet</i>	13
Figura 6 – Exemplo de imagem do acorde dó	16
Figura 7 – Fluxograma da Metodologia	17
Figura 8 – (a) RGB, (b) cinza, (c) <i>K-means</i>	18
Figura 9 – (a) Acurácia <i>U-net</i> 128x128 RGB (b) <i>Loss U-net</i> 256x256 RGB (c) Acurácia <i>U-net</i> 256x256 RGB (d) <i>Loss U-net</i> 256x256 RGB	20
Figura 10 – (a) Acurácia <i>U-net</i> 128x128 cinza (b) <i>Loss U-net</i> 256x256 cinza (c) Acurácia <i>U-net</i> 256x256 cinza (d) <i>Loss U-net</i> 256x256 cinza	21
Figura 11 – (a) Acurácia <i>U-net</i> 128x128 <i>K-means</i> (b) <i>Loss U-net</i> 256x256 <i>K-means</i> (c) Acurácia <i>U-net</i> 256x256 <i>K-means</i> (d) <i>Loss U-net</i> 256x256 <i>K-means</i>	21
Figura 12 – (a) Acurácia <i>Linknet</i> 128x128 RGB (b) <i>Loss Linknet</i> 256x256 RGB (c) Acurácia <i>Linknet</i> 256x256 RGB (d) <i>Loss Linknet</i> 256x256 RGB	22
Figura 13 – (a) Acurácia <i>Linknet</i> 128x128 cinza (b) <i>Loss Linknet</i> 256x256 cinza (c) Acurácia <i>Linknet</i> 256x256 cinza (d) <i>Loss Linknet</i> 256x256 cinza	23
Figura 14 – (a) Acurácia <i>Linknet</i> 128x128 <i>K-means</i> (b) <i>Loss Linknet</i> 256x256 <i>K-means</i> (c) Acurácia <i>Linknet</i> 256x256 <i>K-means</i> (d) <i>Loss Linknet</i> 256x256 <i>K-means</i>	23
Figura 15 – (a) Acorde dó original (b) Acorde dó segmentado	24

LISTA DE TABELAS

Tabela 1 – Tabela de acurácia em teste da rede <i>U-net</i>	22
Tabela 2 – Tabela de acurácia em teste da rede <i>Linknet</i>	24
Tabela 3 – Tabela de acurácia em teste do método <i>bagging</i>	24
Tabela 4 – Tabela de acurácia em teste do método <i>boosting</i>	25
Tabela 5 – Tabela de acurácia em teste do método <i>stacking</i>	25
Tabela 6 – Tabela de acurácia em teste do método <i>deep learning</i>	25
Tabela 7 – Métrica por rótulo do classificador <i>deep learning</i> em imagens segmentadas	25

LISTA DE ABREVIATURAS E SIGLAS

IA	Inteligencia Artificial
IC	Iniciação Científica
RGB	<i>Red Green Blue</i>
SFS	<i>Sequential Feature Selection</i>
SVM	<i>Support Vector Machines</i>
SFFS	<i>Sequential Forward Feature Selection</i>
TCC	Trabalho de Conclusão de Curso
ReLU	<i>Rectified Linear Unit</i>

LISTA DE SÍMBOLOS

Γ	Letra grega Gama
λ	Comprimento de onda
\in	Pertence
\sum	Somatório
Θ	Letra grega Theta
\oplus	“ou” exclusivo
σ	Letra grega Sigma
\prod	Produtório

SUMÁRIO

1 – INTRODUÇÃO	1
1.1 PROBLEMA	1
1.2 JUSTIFICATIVA	1
1.3 OBJETIVOS	1
1.3.1 Objetivo Geral	2
1.3.2 Objetivos Específicos	2
1.4 ORGANIZAÇÃO DO TEXTO	2
2 – FUNDAMENTAÇÃO TEÓRICA	3
2.1 VISÃO COMPUTACIONAL	3
2.1.1 Aquisição de Imagens	3
2.1.2 Pré-Processamento	3
2.1.3 Segmentação	4
2.1.4 Pós-Processamento	5
2.1.5 Seleção/Extração de Características	6
2.1.6 Classificação	6
2.1.7 Avaliação	7
2.2 <i>ENSEMBLE LEARNING</i>	8
2.3 OTIMIZAÇÃO DE DADOS	9
2.4 <i>DEEP LEARNING</i>	11
2.4.1 <i>U-net</i>	12
2.4.2 <i>Linknet</i>	13
2.5 REDES NEURAIS CONVOLUCIONAIS	13
2.6 TRABALHOS RELACIONADOS	14
2.6.1 Materiais e Métodos	14
2.6.2 Resultados	15
3 – MATERIAIS E MÉTODOS	16
3.1 MATERIAIS	16
3.2 METODOLOGIA	16
3.2.1 Pré-processamento	17
3.2.2 Segmentação	17
3.2.3 Extração de características	18
3.2.4 Processamento do <i>dataset</i>	18
3.2.5 Classificação	19
3.2.6 Avaliação dos classificadores	19

4 – RESULTADOS	20
4.1 Segmentação	20
4.2 Classificação	24
5 – CONSIDERAÇÕES FINAIS	27
Referências	28

1 INTRODUÇÃO

A grande vantagem de algoritmos computacionais é a velocidade com que os dados são processados, tornando a execução de tarefas muito mais rápida do que seria se fosse feita manualmente. Porém, uma das dificuldades é a incapacidade de adaptação do algoritmo à diferentes situações, ao menos até a criação da Inteligência Artificial (IA). Conceitualmente, a IA foi criada em 1956 por John McCarthy. Essa linha de estudo tem como objetivo fazer com que algoritmos possam aprender padrões para se adaptarem a diferentes cenários de um problema ([MCCORDUCK et al., 1977](#)).

Uma das áreas em que a IA pode ser aplicada é a visão computacional, que consiste em classificar objetos presentes nas imagens a partir de etapas de processamento das imagens, extração de características e classificações ([BARELLI, 2018](#)).

1.1 PROBLEMA

A principal questão a ser abordada neste projeto é o desenvolvimento de um algoritmo de classificação de acordes que possa obter bons resultados em uso fora de um ambiente controlado. Sendo assim, um dos problema a ser resolvido é encontrar o conjunto de técnicas que permitam que o algoritmo possa generalizar suas classificações para ambientes com diferentes níveis de luz, tonalidades de pele, ângulos, e localizações, uma vez que, por se tratar de um classificador de acordes, levando em conta um uso prático para o algoritmo, poderia ser utilizado em vários ambientes. Além disso, um outro problema é encontrar os melhores parâmetros nas diferentes etapas do processamento que contribuam para a eficácia do classificador, como os parâmetros de *threshold* na etapa de segmentação, por exemplo.

1.2 JUSTIFICATIVA

Além da importância técnica em relação ao conhecimento agregado no tema de aprendizado de máquina, por se tratar de um estudo mais aprofundado de um trabalho já existente, e permitir o estudo comparativo de diferentes métodos nesta linha de pesquisa dentro do tema musical, existe também uma importância socio-cultural em relação ao uso deste tipo de trabalho, que possui o objetivo, se possível, de facilitar o aprendizado voltado à criação musical, uma vez que um algoritmo pode ser treinado para “entender” as composições, e, posteriormente, pode ser usado para facilitar o trabalho de aprendizado.

1.3 OBJETIVOS

Esta subseção está dividida entre os objetivos gerais da pesquisa e os específicos.

1.3.1 Objetivo Geral

Este trabalho tem como principal objetivo a obtenção de um classificador de acordes de violão, utilizando métodos de visão computacional com *deep learning* e *ensemble learning*.

1.3.2 Objetivos Específicos

Para alcançar o objetivo geral os seguintes objetivos específicos foram definidos:

- Implementar e aplicar métodos de segmentação de imagens, utilizando aprendizado de máquina para automatizar a escolha dos parâmetros ótimos.
- Realizar a normalização e estudo da dimensionalidade das características extraídas para tornar o processamento dos dados mais eficiente.
- Extrair características de cor e forma.
- Classificar as imagens a partir das características extraídas utilizando *deep learning* e *ensemble learning*.
- Analisar e comparar os resultados obtidos com resultados de um trabalho anterior ([SHINODA; OLIVEIRA, 2019](#)).

1.4 ORGANIZAÇÃO DO TEXTO

Este relatório está organizado da seguinte forma: o Capítulo 2 contém a fundamentação teórica. Os materiais e métodos são apresentados no Capítulo 3 e os resultados obtidos da metodologia no Capítulo 4. Por fim, no Capítulo 5 são apresentadas as considerações finais.

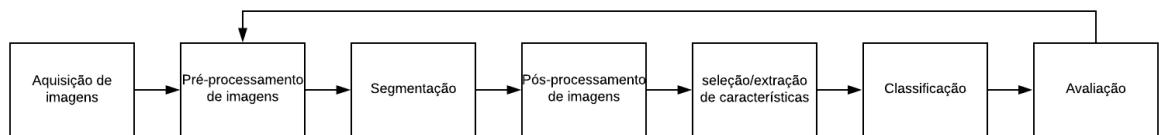
2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo, são abordados aspectos teóricos sobre visão computacional, além dos conceitos de *deep learning*, *ensemble learning*, métodos de segmentação automatizada e otimização de dados.

2.1 VISÃO COMPUTACIONAL

Tratando-se das etapas desenvolvidas na visão computacional, métodos de pré-processamento têm o objetivo de melhorar a qualidade da informação extraída da imagem. Por conta disso, neste processo, os parâmetros de cada uma dessas etapas são controlados, de forma que este sistema seja ótimo para problemas específicos, este conjunto de processos é chamado de processamento de imagens (ALBUQUERQUE; ALBUQUERQUE, 2000). A partir do processamento de imagens, aplica-se o reconhecimento de padrões, que consiste na etapa de gerar informações e usá-las para classificação dos dados (BARELLI, 2018). Dessa forma a visão computacional pode ser dividida nas seguintes etapas: Aquisição das imagens, pré-processamento, segmentação, pós-processamento, extração de características, classificação e avaliação, como pode ser visto na [Figura 6](#).

Figura 1 – Fluxograma da Visão Computacional



Fonte: Elaborado pelo autor

2.1.1 Aquisição de Imagens

A etapa de aquisição de imagens, como o próprio nome sugere, consiste em obter o banco de imagens, sejam elas criadas, utilizando-se de câmeras, por exemplo, ou usar um banco já pronto. A partir disso, quando digitalizada, a imagem passa a ser entendida pelo sistema como uma matriz de intensidade de cores na qual cada posição é referente a um *pixel* da imagem (BACKES; JUNIOR, 2019).

2.1.2 Pré-Processamento

Se na etapa de tratamento anterior à aquisição da imagem, há uma manipulação do ambiente físico para garantir a qualidade da imagem, na etapa de pré-processamento, é realizado o controle da qualidade de imagem no ambiente virtual, ou seja, são usados alguns

algoritmos computacionais nesta etapa para que seja suprida a necessidade de melhoria na qualidade da imagem proveniente da primeira etapa de tratamento. Algumas das técnicas usadas nesta etapa são:

- **Equalização de histograma**, neste método usa-se do histograma de imagens. O histograma mostra a distribuição estatística de cada tonalidade de cores em relação ao total de *pixels* presentes na imagem. Desta forma, ao aplicar a equalização, procura-se evidenciar as diferenças entre essas tonalidades aumentando a uniformidade dos tons, facilitando assim a detecção de detalhes na imagem (SOUZA; CORREIA, 2007).
- **Filtros**, este último método consiste em procurar e suavizar ruídos da imagem. Para isso usa-se uma máscara, que nada mais é do que uma matriz menor que a matriz correspondente à imagem. Esta matriz menor percorre a imagem e realiza um cálculo ponderando todos os *pixels* da vizinhança do *pixel* central da matriz e atualizando seu valor (SOUZA; CORREIA, 2007). Este cálculo pode ser visto na [Equação \(1\)](#).

$$P_C = \sum_{i=1}^t (v_i * p_i) \quad (1)$$

Sendo “ P_C ” o valor final do *pixel* central, “ t ” a quantidade de *pixels* na máscara, “ v ” o valor do *pixel* atual da vizinhança e “ p ” o valor do peso do *pixel* atual.

2.1.3 Segmentação

Na etapa de segmentação da imagem serão extraídos os objetos de interesse do fundo da imagem. Para isso a imagem é submetida a um processo inicial de conversão em escala de cinza. Este processo consiste em calcular a média dos canais de cores da imagem, por exemplo, caso o canal usado seja o RGB (*red, green, blue*), será somado o valor de cada um dos três canais e dividido por 3, resultando assim em um único canal na escala de cinza. Existe, também, a possibilidade de evidenciar uma certa cor atribuindo pesos nos canais (BARELLI, 2018). Este processo pode ser visto na [Equação \(2\)](#).

$$C = \frac{\sum_{i=1}^t v_i}{t} \quad (2)$$

Sendo que “ C ” é a tonalidade em cinza resultante, “ t ” é o total de canais e “ v ” é o valor do canal atual.

Após esta etapa, ocorre a limiarização, ou binarização, na qual de fato, será realizada a segmentação da imagem, uma vez convertida a imagem em tons de cinza, define-se um valor limite, ou valor de *threshold*, dentro do espectro destas tonalidades, na qual todos os *pixels* cujo os valores estejam abaixo deste limite, terão uma tonalidade única, normalmente preto ou branco, enquanto que os que estão acima deste limite, terão a tonalidade oposta.

Um dos problemas neste método de segmentação é que exige um trabalho muitas vezes manual para se encontrar o melhor valor de *threshold*, e com bons resultados somente em ambientes específicos. Por conta disso, um outro meio é utilizar um algoritmo de aprendizado

de máquina para automatizar a classificação do *pixel* em abaixo ou acima do limiar. Uma das técnicas mais disseminadas neste tipo de abordagem é utilizando classificadores não supervisionados, mais especificamente, o *k-means*. Nesta abordagem usa-se a semelhança de tonalidades dos *pixels* para organizá-los em grupos, fazendo com que os valores com maior semelhança tenham a tendência de se agruparem em um mesmo *cluster* (TAKAHASHI; BEDREGAL; LYRA, 2005).

Além de se usar métodos de segmentação baseados em aprendizado não supervisionado, existe, também, a possibilidade de segmentar utilizando métodos supervisionados, como por exemplo, usando redes neurais, com o mesmo intuito de se classificar o rótulo do *pixel* como objeto de interesse ou não (SIMÕES, 2000).

2.1.4 Pós-Processamento

Após realizada a segmentação da imagem, é possível que ainda existam ruídos na segmentação, por conta disso, existe a etapa de pós-processamento, na qual são aplicadas técnicas de realce ou eliminação de partes indesejadas na segmentação. Este processo consiste no uso de métodos de morfologia matemática, sendo eles: erosão, dilatação, abertura e fechamento.

- **Erosão** é o método que consiste em reduzir as bordas de um objeto segmentado, ou seja, ele é capaz de separar objetos em contato na segmentação, ou apagar pequenos ruídos na imagem (ISHIKAWA, 2008). Para isso, usa-se a seguinte a Equação (3).

$$A \ominus B = \{c | (B)_C \subseteq A\} \quad (3)$$

Sendo que “ $A \ominus B$ ” é a erosão de “ A ” pelo elemento estruturante(matriz) “ B ”, e “ C ” são todos os *pixels* na qual ocorrerão os translados de “ B ”.

- **Dilatação**, este método, diferentemente da erosão, serve para aumentar os objetos segmentados, isso ocorre expandindo suas bordas. Este método serve para corrigir falhas ou *pixels* faltantes em objetos de interesse segmentados (ISHIKAWA, 2008). De acordo com a Equação (4).

$$A \oplus B = \left\{ x | [(B)_x \bigcap A] \subseteq A \right\} \quad (4)$$

Tem-se que “ $A \oplus B$ ” é a dilatação do segmento “ A ” no elemento estruturante “ B ” em que todos os *pixels* “ x ” são transladados por “ B ” e sobrepõem os elementos de “ A ” em que há pelo menos um elemento de valor não nulo.

- **Abertura** consiste em reduzir os ruídos em um processo semelhante à erosão, porém sem impactar de maneira radical os objetos de interesse segmentados. Para isso usa-se de uma erosão seguida de uma dilatação.
- **Fechamento**, por fim, a operação de fechamento, de maneira análoga à operação de abertura, consiste aumentar um objeto de interesse sem aumentar consideravelmente os ruídos, para isso, usa-se da operação de dilatação seguida pela erosão.

2.1.5 Seleção/Extração de Características

Nesta etapa do processamento de imagens é onde se obtêm, efetivamente, os dados das imagens. Para isso, são extraídas características pertinentes ao objeto de interesse para posterior treino e classificação destas imagens.

Existem duas abordagens diferentes para a escolha das características, sendo elas: a seleção das características e a extração das características.

A etapa de seleção de características consiste em buscar as informações diretas das imagens que são pertinentes para a classificação. Por exemplo, altura do objeto segmentado ou quantidade de *pixels* no perímetro. Enquanto que, a extração consiste em informações indiretas da imagem, este tipo de característica é obtida através das características diretas. Tomando como exemplo as características diretas altura e largura do objeto de interesse, pode se extrair a característica da razão entre a altura e largura (LEE, 2005), ou o *Chain-Code*, que consiste em analisar os *pixels* de borda e a direção dos seus *pixels* vizinhos de acordo com o tipo de conectividade, normalmente levando-se em conta a conectividade-8, e as características são a somatória de *pixels* vizinhos em cada uma das direções.

Além da etapa da obtenção das características, existe também a necessidade de processamento destas características obtidas, realizando uma otimização nestes dados. Sendo assim, alguns métodos podem ser aplicados, como normalização, verificação de correlação entre características e seleção das características mais importantes. Estes processos podem ser melhor vistos na [Seção 2.3 \(GUYON; ELISSEEFF, 2003\)](#).

2.1.6 Classificação

Na classificação, é a etapa em que efetivamente serão aplicados os métodos de aprendizado de máquina. Para isso, existem alguns métodos usados na classificação, são eles, o supervisionado, na qual os rótulos dos dados já são conhecidos pelo algoritmo, não-supervisionado, que diferentemente do supervisionado, o classificador não possui conhecimento sobre os rótulos, aprendizado semi-supervisionado, que consiste no algoritmo trabalhar com dados rotulados e não rotulados. E, por fim, aprendizado por reforço, que não possui uma base de dados para treino, diferentemente dos métodos anteriores, ele se aprimora a partir de "recompensas", ou seja, caso o algoritmo, que inicialmente toma decisões aleatórias, gera uma decisão correta, ele ganha uma recompensa positiva, enquanto que, se for uma decisão errada, o algoritmo é recompensado negativamente (MONARD; BARANAUSKAS, 2003).

Para este trabalho, por se tratar da área de processamento de imagens, será testado o desempenho do classificador baseado em *deep learning*. Como pode ser visto na [Seção 2.4](#), assim como conceitos importantes para este modelo de classificação como redes neurais convolucionais, que é descrito na [Seção 2.5](#).

Uma outra técnica que pode ser usada para melhorar o desempenho nesta etapa, é o uso de classificadores em conjunto, este método, chamado de *ensemble learning*, ou aprendizado

baseado em agrupamento, pode ser melhor explicado na [Seção 2.2](#).

2.1.7 Avaliação

Por fim, após a classificação, a etapa de avaliação do algoritmo, aqui são utilizadas métricas para saber se o algoritmo possui um desempenho aceitável para a proposta em que ele está incluído. Algumas dessas métricas são: acurácia, precisão, *recall* e *f1-score* ([MONARD; BARANAUSKAS, 2003](#)).

- **Acurácia**, esta métrica nos fornece o grau de acerto geral do algoritmo, como pode ser visto na [Equação \(5\)](#) ([MONARD; BARANAUSKAS, 2003](#)).

$$A = \frac{VP + VN}{T} \quad (5)$$

Sendo que “*VP*” é a quantidade de rótulos positivos que o algoritmo acertou, “*VN*” é a quantidade de rótulos negativos que o algoritmo acertou, e “*T*” é o total de instâncias do *dataset*.

- **Precisão** é a medida que retorna, dentre os dados que são de determinado rótulo, todos que o algoritmo classificou como sendo deste rótulo, de acordo com a [Equação \(6\)](#) ([MONARD; BARANAUSKAS, 2003](#)).

$$P = \frac{VP}{VP + FP} \quad (6)$$

Sendo que “*FP*” é a quantidade de instâncias que o algoritmo classificou como positivo, mas na verdade são negativos.

- **Recall**, pode ser entendido como, de todas as instâncias de um determinado rótulo, quantas o classificador acertou. A fórmula do *recall* pode ser vista na [Equação \(7\)](#) ([MONARD; BARANAUSKAS, 2003](#)).

$$P = \frac{VP}{VP + FN} \quad (7)$$

- **F1-score**, esta equação mostra o quanto o *recall* e a precisão estão平衡ados, sendo que, quanto mais próximo de 1, melhor. Esta métrica pode ser vista na [Equação \(8\)](#) ([MONARD; BARANAUSKAS, 2003](#)).

$$P = \frac{2 * precisao * recall}{precisao + recall} \quad (8)$$

- **Curva ROC**, esta métrica permite balancear a sensibilidade (índice de dados classificados como positivos dentre todos os que, de fato, são positivos) e a especificidade (índice de dados classificados como negativos dentre todos os que, de fato, são negativos), relacionando como o aumento ou diminuição de um dos índices afeta ao outro, como pode ser visto na [Equação \(6\)](#) e [Equação \(9\)](#) ([MONARD; BARANAUSKAS, 2003](#)).

$$E = \frac{VN}{VN + FN} \quad (9)$$

Sendo que “ E ” é a especificidade, “ FN ” são os índices de instância que foram classificadas como negativas, mas são positivas e, “ VN ” é o índice de instâncias classificadas como negativas que, de fato, são negativas.

Desta forma, pode ser visto que a curva ROC mostra o quanto classificar cada vez mais características como positivas vai diminuir o número de acertos das instâncias que são negativas, assim como classificar cada vez mais instâncias como negativas influenciará na queda de instâncias positivas. Sendo assim, é dada esta relação, para se ter uma ideia do balanceamento entre estas duas métricas, de acordo com a [Equação \(10\)](#).

$$ROC = \text{especificidade} * \text{sensibilidade} \quad (10)$$

Logo, a Curva ROC indica que, quanto mais próximo de 1 o valor do produto, melhores são os resultados do classificador.

- **Binary Cross Entropy** esta métrica utilizada em redes neurais convolucionais tem como objetivo calcular o total de erros (diferença do resultado do rótulo com o previsto) do algoritmo. O cálculo do *Binary Cross Entropy* pode ser visto na [Equação \(11\)](#).

$$L(vr, pr) = -vr * \log(pr) - (1 - vr) * \log(1 - pr) \quad (11)$$

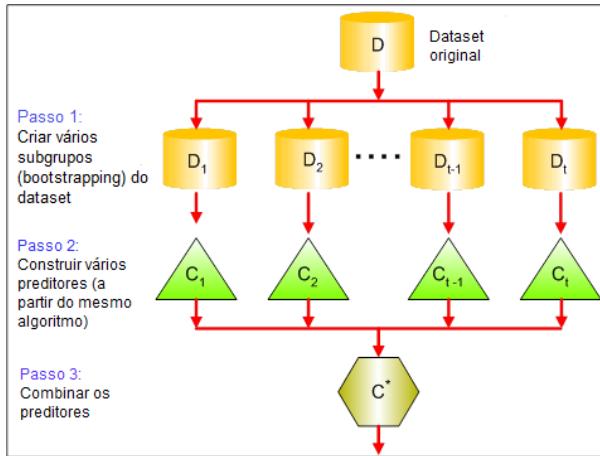
Sendo que “ vr ” é o valor real do rótulo e “ pr ” é a predição do algoritmo.

2.2 ENSEMBLE LEARNING

Uma alternativa para tentar melhorar o desempenho do algoritmo é a aplicação de técnicas de *ensemble learning*, aprendizado por agrupamento. Essa técnica consiste em usar vários classificadores juntos, também chamados de *weak learners*, no processo de classificação, criando um modelo de agrupamento mais robusto, *strong learner* ([ZHOU, 2009](#)).

Para a composição do agrupamento, pode-se usar diferentes modelos de *weak learners*, classificando um agrupamento heterogêneo, ou usar o mesmo modelo de classificadores, criando um agrupamento homogêneo, como é o caso das *Random Forests*, que utilizam somente de árvores de decisão para sua composição. Além disso, existem alguns métodos de agrupamento, sendo os mais usados pela literatura são: *bagging*, *boosting* e *stacking* ([ZHOU, 2009](#)).

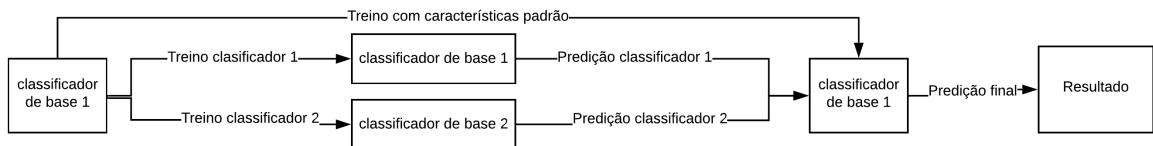
- **Bagging**, esta técnica consiste em subdividir o *dataset* em partes menores para poder ter um melhor controle sobre os valores de média e desvio padrão dos dados. A partir disso, é aplicado um mesmo *weak learner*, pois o *bagging* é um modelo de agrupamento homogêneo, em cada um dos subgrupos, tomando o resultado final como a média dos resultados individuais de cada um ([NEVES, 2019](#)). Pode-se ver o funcionamento do *bagging* na [Figura 2](#).
- **Boosting**, diferentemente do *bagging*, este modelo consiste em treinar os modelos de modo iterativo, ou seja, todos os modelos treinados são ponderados pelos resultados obtidos dos modelos anteriores, criando uma relação entre eles, assim, ao final do último

Figura 2 – Fluxograma do modelo *bagging*.

Fonte: ([PORUSNIUC et al., 2019](#))

classificador, o resultado é obtido colocando maior relevância nos classificadores que gerarem maiores acurácia ([NEVES, 2019](#)).

- **Stacking**, este modelo consiste em usar as previsões dos *weak learners* como características do modelo de previsão final, ou seja, a saída destes classificadores de base será inserida na entrada do classificador final junto às outras características padrão ([NEVES, 2019](#)). Um modelo do *stacking* pode ser visto na [Figura 3](#).

Figura 3 – Representação do modelo *stacking*.

Fonte: Elaborado pelo autor

2.3 OTIMIZAÇÃO DE DADOS

Na etapa de seleção de características, existe muitas vezes a necessidade de melhorar a qualidade do *dataset* para tornar a etapa de classificação menos custosa e obter uma maior acurácia. Por conta disso alguns métodos podem ser aplicados para verificar a qualidade dos dados.

Alguns dos problemas encontrados é em relação à redundância ou dados discrepantes. Por questões, muitas vezes, de equipamentos falhos na aquisição, um ambiente não propício para adquirir bons dados, e também problemas na segmentação das imagens.

No caso de redundância de dados, os casos mais perceptíveis são quando há dados repetidos no *dataset*, na qual a solução é a exclusão das instâncias iguais, mantendo-se somente

uma instância. Porém, em alguns casos, esta redundância não se mostra como valores idênticos, mas sim como dados que se comportam da mesma maneira, que acabam resultando em informações, também redundantes, para o classificador. Para este caso, calcula-se a correlação destas instâncias, uma das equações de correlação mais conhecidas é o cálculo de correlação de Pearson (GUYON; ELISSEEFF, 2003). Como pode ser visto na [Equação \(12\)](#).

$$r = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_i (x_i - \bar{x})^2 \sum_i (y_i - \bar{y})^2}} \quad (12)$$

Sendo que “ x ” e “ y ” representam as características de interesse para análise de correlação, “ i ” a instância atual do *dataset*, e “ r ” o resultado da correlação. Este cálculo mostra o quanto as características possuem o mesmo comportamento uma em relação à outra, sendo que, quanto mais próximo de 1 é o valor de “ r ”, mais as duas têm comportamento semelhante e em sentidos iguais, ou seja, quando os valores de uma característica aumentam, os da outra aumentam na mesma proporção. E quanto mais próximo de -1, maior é a relação do comportamento em sentidos opostos, o que significa que quando os valores de uma característica aumentam, os da outra diminuem proporcionalmente, sendo assim, quanto mais próximo de 0, menor é a correlação entre as características, portanto, menor a chance de redundância nos dados (GUYON; ELISSEEFF, 2003).

Além destes problemas, existe também a questão de características com valores discrepantes em relação a outras. Isso faz com que ao se treinar um classificador na qual no *dataset*, por exemplo, existe uma característica “ C_1 ”, cujo os valores variam entre 500 e 1000, e uma característica “ C_2 ”, em que os valores variam entre 0 e 1, a característica com maior valor gera uma dominância em relação à característica com menor valor, ou seja, o classificador atribui um menor peso na classificação desta segunda característica, fazendo com que a informação trazida por essa característica não seja devidamente considerada pelo algoritmo (KOTSIANTIS; KANELLOPOULOS; PINTELAS, 2006).

Para isso, o ideal é evitar que os valores das características tenham uma discrepância muito alta, um dos métodos que podem ser aplicados para isso é a normalização, que consiste em limitar o intervalo de valores entre -1 e 1, como pode ser visto na [Equação \(13\)](#).

$$x' = \frac{x - \mu}{\sigma^2} \quad (13)$$

Na [Equação \(13\)](#), “ x' ” representa o valor normalizado, “ x ”, o valor original, “ μ ”, a média dos valores desta característica, e, por fim, “ σ ”, que representa o desvio padrão. Com isso, essa fórmula permite que a distribuição esteja proporcionalmente menor, sem perder a qualidade dos valores (KOTSIANTIS; KANELLOPOULOS; PINTELAS, 2006).

Por fim, existe um conceito adotado ao se otimizar os dados. Esse conceito representa a relação entre a quantidade de características a serem processadas, e a quantidade de instâncias disponíveis pelo *dataset*. Essa relação indica a quantidade ideal de dados que se deve ter para uma boa classificação. Conhecido como “Maldição da Dimensionalidade”, esta análise, que

pode ser extraída por meio da aplicação da Equação (14) (KOTSIANTIS; KANELLOPOULOS; PINTELAS, 2006).

$$V_t = \prod_{i=1}^t V_i \quad (14)$$

Sendo que " V_t " é a quantidade de instâncias ideal, " t " é o total de características sendo usadas, e " V_i " é a quantidade de valores possíveis para a característica " i ". Naturalmente, para este cálculo, os valores devem ser discretos, e não contínuos.

Por conta disso, ao invés de conseguir o número de instâncias necessárias, uma outra maneira é diminuir a quantidade de características para não sofrer tanto dos efeitos da "Maldição da Dimensionalidade". Para isso, pode-se usar técnicas para seleção das características mais representativas, uma das soluções mais usadas, é aplicar uma busca heurística (PUDIL; NOVOVIČOVÁ; KITTLER, 1994).

Uma das possibilidades da busca heurística é a *Sequential Forward Selection* (SFS), que consiste em criar um conjunto, inicialmente vazio, de características. A partir disso, busca-se a melhor característica que, em conjunto com as já selecionadas, geram os melhores resultados. Sendo assim, em primeiro lugar, a melhor característica é selecionada, em seguida, a segunda é selecionada de modo que, em conjunto com a primeira, gerem os melhores resultados, e assim por diante (PUDIL; NOVOVIČOVÁ; KITTLER, 1994).

2.4 DEEP LEARNING

O aprendizado profundo é uma técnica de classificação supervisionada, ou seja, para o seu aprendizado, ela utiliza de um *dataset* previamente rotulado para poder balancear seus parâmetros de forma que se aproxime cada vez mais de uma acurácia ótima (NIELSEN, 2015).

Para isso, o aprendizado profundo utiliza de redes neurais para a classificação. Porém, diferentemente de uma rede neural convencional, esta técnica implementa uma rede massiva, na qual existem muitas camadas intermediárias entre a camada de entrada do *dataset* e a camada final (NIELSEN, 2015).

O funcionamento desta rede multicamadas ocorre a partir da hierarquização das características fornecidas. Sendo assim, as características de camadas superiores são formadas pelo conjunto de características de níveis subjacentes a ela na hierarquia. Por exemplo, nós pertencentes à camadas mais iniciais são responsáveis por detecção de características de borda, cantos ou retas, enquanto os nós das camadas mais profundas são responsáveis pela detecção de formas a partir das bordas, cantos e retas detectadas nas camadas iniciais (BEZERRA, 2016).

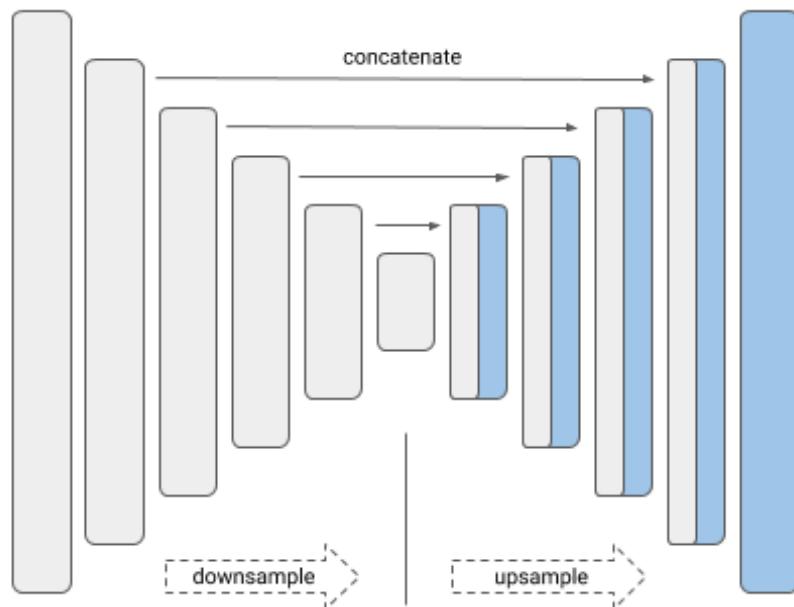
2.4.1 U-net

A rede *U-net* consiste em uma rede neural convolucional utilizada para segmentação de imagens. Como qualquer rede convolucional, ela gera vários filtros que são passados através de máscaras pela imagem gerando uma “nova imagem” filtrada da original. No caso da *U-net*, sua arquitetura consiste em duas estruturas: a *downsample* e *upsample* (RONNEBERGER; FISCHER; BROX, 2015).

Na etapa de *downsampling* a imagem passa por diferentes convoluções através de dois filtros 3x3. Após isso, é realizado o processo de *max pooling* que consiste em passar uma nova máscara na imagem e pegar o pixel com maior valor e inseri-lo na imagem filtrada, no caso da *U-net*, o *max pooling* usa uma máscara 2x2 que cria uma imagem filtrada com metade das dimensões do *input* da camada anterior (RONNEBERGER; FISCHER; BROX, 2015).

Uma vez terminada as camadas de *downsampling*, a imagem é montada novamente através das camadas de *upsampling* na qual o processo inverso é realizado. Logo, é criada uma convolução com máscara 2x2 que une as imagens filtradas de camada em camada e posteriormente passa-se duas máscaras 3x3 para correção de problemas de borda que podem ser geradas da etapa de *downsampling*. A estrutura da rede *U-net* pode ser vista na Figura 4 (RONNEBERGER; FISCHER; BROX, 2015).

Figura 4 – Estrutura da rede *U-net*.



Fonte: (YAKUBOVSKIY, 2019)

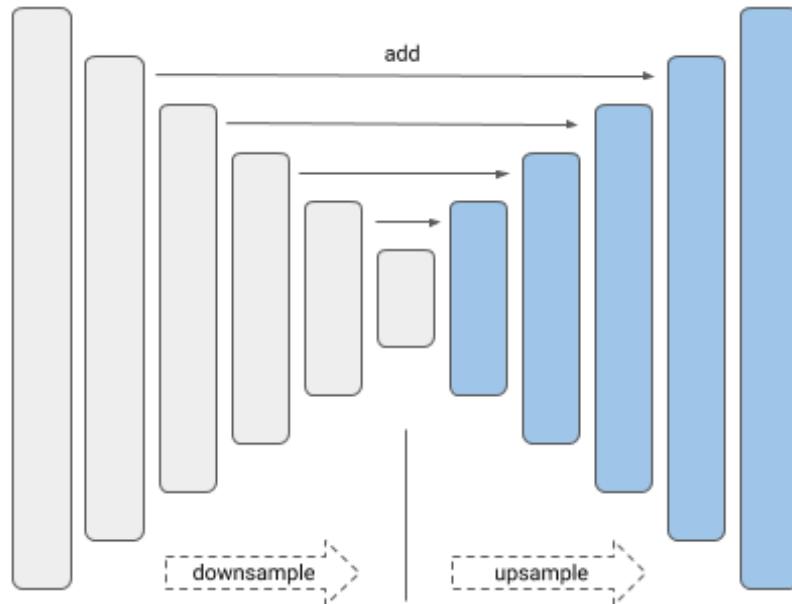
2.4.2 Linknet

A rede *Linknet* é uma rede neural convolucional cuja a estrutura é baseada em *encoding* e *decoding*, que são processos análogos ao *downsampling* e *upsampling*, respectivamente, da *U-net* ([CHAURASIA; CULURCIELLO, 2017](#)).

No caso do *encoding*, inicialmente, passa-se uma máscara 7x7 seguida de um *max pooling* em uma área de 3x3 no *input* da camada, da mesma forma que ocorre no *downsampling* ([CHAURASIA; CULURCIELLO, 2017](#)).

Já na parte do *decoding*, diferentemente do *upsampling*, além de realizar o processo oposto do *encoding* unindo as imagens filtradas, ele também adiciona no *output* das camadas, o *input* das respectivas camadas do *encoding*. Isso ocorre para preservar possíveis perdas que ocorreriam naturalmente no processo de *encoding*, já que nos *outputs* é diminuído a dimensão dos dados em relação aos *inputs*, fazendo com que percam informações espaciais da imagem ([CHAURASIA; CULURCIELLO, 2017](#)). A estrutura da rede *Linknet* pode ser vista na [Figura 5](#).

Figura 5 – Estrutura da rede *Linknet*.



Fonte: ([YAKUBOVSKIY, 2019](#))

2.5 REDES NEURAIS CONVOLUCIONAIS

Um dos principais problemas ao se utilizar redes consideravelmente massivas está no desempenho do algoritmo, pois, uma das características das redes neurais é a conectividade

global, na qual os nós de uma camada são conectados, cada um deles, com cada nó da camada adjacente a eles, fazendo com que o número de possibilidades de conexões cresça exponencialmente de acordo com a quantidade de nós de cada camada.

Por conta disso, foram criadas as redes neurais convolucionais, que diferentemente das ligações convencionais entre os nós das camadas, que são feitas de forma global, estas são realizadas de forma local, ou seja, são criadas regiões de conectividade. Desta forma, somente alguns nós de uma camada são conectados a um único nó da camada posterior, este nó é chamado de campo receptivo local ([PORUSNIUC et al., 2019](#)).

Em processamento de imagens, cada nó da primeira camada intermediária, ou seja, cada campo receptivo local desta camada pode classificar características básicas que compõem a imagem, tal como, bordas, segmentos e curvas ([BEZERRA, 2016](#)).

Como uma imagem é composta de várias destas características básicas, um campo responsável por esta característica pode ser usado em diferentes nós de camadas posteriores, sendo assim, pode-se ter nós, na camada de detecção de elementos básicos, ligados com um conjunto diferente de nós da camada anterior, porém estes detectores possuem os mesmos parâmetros de peso e viés, fazendo com que eles processem a mesma característica, mas atuando em diferentes regiões da imagem ([BEZERRA, 2016](#)).

A rede neural convolucional aplica também, como o nome sugere uma convolução. Para isso, o algoritmo aplica uma máscara dentro da imagem de modo que esta máscara a percorra inteira, em cada iteração em que a máscara muda de posição, ela aplica uma operação nos *pixels* em que ela está posicionada ([BEZERRA, 2016](#)).

Por fim, é aplicada uma técnica de subamostragem, na qual a imagem original é reduzida, em relação a sua resolução, sem perder muita informação. Este processo ocorre percorrendo uma máscara na imagem e reduzindo todos os *pixels* presentes nesta máscara em um único *pixel* ([BEZERRA, 2016](#)).

2.6 TRABALHOS RELACIONADOS

Nesta seção foi descrito o trabalho relacionado *Visual recognition of guitar chords using neural networks* que tem, também, como objetivo, a classificação de acordes de violão utilizando visão computacional com redes neurais ([COMA, 2020](#)).

2.6.1 Materiais e Métodos

Em relação aos materiais, foram utilizados 4 bases de imagens diferentes, sendo elas: *Hand Dataset* com 13 mil imagens de mãos, *SCUT-Ego-Gesture Dataset* com 59 mil imagens de mãos em 16 posições diferentes, 275 imagens livres pegas no Google, sendo 95 de pessoas tocando violão, mostrando-as de corpo inteiro, e as outras 180 de mãos tocando diferentes acordes e, por fim, foram tiradas mais 208 fotos de 3 diferentes pessoas tocando diferentes acordes. Já em relação ao *hardware*, foi utilizada uma GPU Geforce GTX 1080 Ti ([COMA](#),

2020).

Inicialmente, foi utilizada uma rede YOLO (*You Only Look Once*), que é uma rede neural que realiza suas predições através de *bounding boxes*. Nesta rede, foi realizado um pré-treino com a base *Hand Dataset* e as 303 imagens de corpo inteiro. Além disso, uma outra rede chamada *Hourglass Network*, que é uma rede utilizada para detecção de partes do corpo humano, passou também por um processo de pré-treinamento, na qual ela foi alimentada com a base *SCUT-Ego-Gesture Dataset*, nesta primeira etapa, a rede foi treinada para detectar somente a ponta dos dedos e suas coordenadas. Além disso, foram realizados processos de aumento de dados nas bases de imagens e o *input* nas redes foram com imagens na escala 416x416 pixels na rede YOLO e 300x300 pixels na *Hourglass Network*. Cada uma delas rodou por 100 épocas (COMA, 2020).

Para a classificação, na base de imagens de teste foi utilizada primeiramente a rede YOLO para gerar o *bounding box* da mão e isola-lá, e a partir disso foi usada a rede *Hourglass Network* para projetar em quais casas as cordas os dedos estão posicionados, classificando em 14 acordes, maiores e menores, diferentes (COMA, 2020).

2.6.2 Resultados

Para a rede YOLO, testando em 61 imagens de validação, foram obtidos 96% de precisão e 95% de *recall* (COMA, 2020).

Após isso, utilizando a rede *Hourglass Network* para a detecção de acordes de fato, foi obtido uma acurácia de 97% (COMA, 2020).

3 MATERIAIS E MÉTODOS

Nesta seção são abordados os materiais que foram utilizados, bem como a metodologia empregada no projeto.

3.1 MATERIAIS

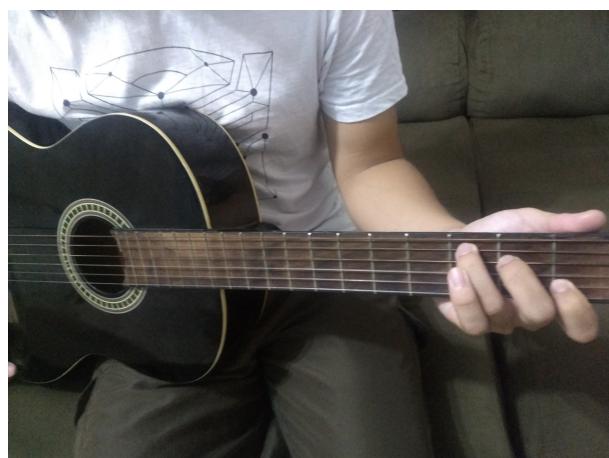
Este tópico descreve os materiais, sendo eles: *software*, *hardware* e conjunto de dados utilizados.

O computador utilizado é um Lenovo X240, com um processador Intel I5 4300U, com 8 *gigabytes* de memória RAM.

Em relação aos *software*, para o desenvolvimento do projeto, foi utilizada a linguagem Python 3.7, com as bibliotecas OpenCV 4.5.1.48 para o processamento das imagens, Scikit-Learn 0.24.1 para a implementação dos métodos de aprendizado de *ensemble learning* e a biblioteca Tensorflow 2.4.1 para desenvolvimento das técnicas de *deep learning*.

O banco de imagens usado foi o mesmo banco do projeto de iniciação científica ([SHINODA; OLIVEIRA, 2019](#)), sendo assim, são 520 imagens, com a resolução de 4032x3024, divididos entre os acordes de dó, ré, mi, fá, fá#, sol lá e si. Sendo 64 imagens do acorde dó, 70 imagens do ré, 62 do mi, 62 do fá, 69 do fá#, 63 do sol, 66 do si e 64 do lá.

Figura 6 – Exemplo de imagem do acorde dó

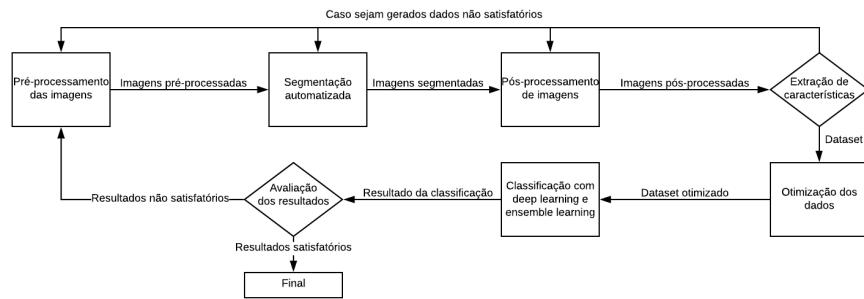


Fonte: Elaborado pelo autor

3.2 METODOLOGIA

A metodologia deste trabalho foi baseada na estrutura do trabalho realizado durante a iniciação científica ([SHINODA; OLIVEIRA, 2019](#)), adicionando melhorias nas etapas de segmentação, extração de características e classificação. Todo o processo pode ser visto na [Figura 7](#).

Figura 7 – Fluxograma da Metodologia



Fonte: Elaborado pelo autor

3.2.1 Pré-processamento

Inicialmente, na etapa de pré-processamento de dados, foi utilizado um filtro bilateral para melhoria da qualidade da imagem, este filtro tem como objetivo remover pequenos ruídos de sal e pimenta mantendo a definição das bordas, para isso, ele calcula a media gaussiana ponderada dos *pixels* vizinhos somente se estes *pixels* possuem um valor semelhante ao central. Para isto foi passado uma máscara 25x25 pela imagem para aplicação do filtro.

3.2.2 Segmentação

Na etapa de segmentação, o principal objetivo foi testar métodos de segmentação semântica que consiste em aplicar técnicas de aprendizado de máquina para a segmentação, sendo assim, não havendo nenhuma interferência humana na escolha dos parâmetros.

Em primeiro lugar, a imagem original foi cortada e mantida só a parte da mão, reduzindo a imagem original de 4032x3024 para 2015x1512, por conta de que, como os métodos de *deep learning* usam os próprios *pixels* como características, na imagem original, a quantidade de *pixels* referentes ao objeto de interesse são muito menores do que a quantidade de *pixels* de *background*, o que torna o *dataset* para segmentação muito desbalanceado.

Feito isso, foram testados diferentes formatos de imagem para treinamento e validação dos algoritmos de segmentação, sendo esses tipos: imagem RGB, em escalas de cinza, e com o espaço de cores agrupado. Cada um desses tipos foram testados em resoluções de 128x128 e 256x256.

Para o agrupamento de cores da imagem, foi utilizado o classificador *K-means* que agrupou todas as tonalidades em 20 centroides, este valor foi escolhido de maneira empírica. Dessa forma, os diferentes formatos podem ser vistos na Figura 8.

Para a segmentação propriamente dita, foram testados as redes *U-net* e *Linknet*, descritos na Seção 2.4. Cada rede treinou com cada formato de imagem descrito na etapa anterior, nas resoluções de 128x128 e 256x256, cada *dataset* foi separado em 1/3 para teste e 2/3 para treinamento, os algoritmos rodaram com um *batch size* de 16, ou seja, ele roda 16 instâncias do *dataset* antes de atualizar seus parâmetros e, também, foram testados em 50

Figura 8 – (a) RGB, (b) cinza, (c) *K-means*

Fonte: Elaborado pelo autor

épocas, o que significa que ele itera 50 vezes sobre o *dataset* de treino.

Feito isso, os resultados foram comparados, e o classificador com maior acurácia e menor índice de *loss* foi escolhido para realizar a segmentação. Os resultados podem ser vistos na [Seção 4.1](#).

3.2.3 Extração de características

Nesta etapa foram extraídas as características das imagens segmentadas, sendo elas: *Chain-Code*, que pode ser visto na [Subseção 2.1.5](#). A razão entre a largura e altura do objeto de interesse, de acordo com a equação [Equação \(15\)](#).

$$R = \frac{P_{acima} - P_{abaixo}}{P_{direita} - P_{esquerda}} \quad (15)$$

Sendo que " P_{acima} ", " P_{abaixo} ", " $P_{esquerda}$ " e " $P_{direita}$ " são as coordenadas dos *pixels* mais acima, mais abaixo, mais à esquerda e mais à direita pertencentes ao objeto de interesse.

A terceira característica extraída é a maior convexidade do objeto de interesse, então foi traçada uma reta de borda a borda da convexidade e, a partir disso, calculada a distância entre o ponto mais profundo da convexidade e a reta, e, por fim, foi pego como valor da característica a maior distância adquirida, dentre todas as convexidades calculadas. A última característica é a distância entre a pestana e a mão, para isso, foi medida a distância entre o *pixel* mais acima pertencente à mão, e o *pixel* mais acima pertencente à pestana.

3.2.4 Processamento do *dataset*

Extraídas as características, a próxima etapa é a de processamento do *dataset*, como descrito pela [Seção 2.3](#), sendo assim, inicialmente foi realizada uma etapa de normalização dos dados, uma vez que o intervalo de valores entre as características não são limitados, fazendo com que algumas destas características tenham um peso menor na classificação, resultando em um valor entre 0 e 1.

Por fim, pode ser que por conta da maldição da dimensionalidade, não existam dados suficientes na nuvem de características, resultando em muitos espaços sem valores, sendo assim,

foi aplicada a busca heurística SFS, para verificar qual é a melhor combinação de características que resulta nos melhores resultados.

3.2.5 Classificação

Feito a etapa de processamento, é a hora da classificação, nesta etapa, foram testados, os algoritmos de *deep learning* e *ensemble learning*, neste segundo, foram unidos diferentes métodos de aprendizado supervisionado existentes na biblioteca Scikit-learn, procurando sempre unir algoritmos com abordagens diferentes para criar um modelo final feito de classificadores de base bem diversificado e mais tolerante à diferentes situações. Para isso, foram testados os seguintes classificadores de base: *Random forest*, *multilayer perceptron*, *SVM* e *extra trees*.

No caso do algoritmo de *deep learning*, foi testado uma rede com 5 camada de convolução, na qual cada camada implementa a função de ativação ReLU.

O *dataset* foi dividido em 66% para treinar o algoritmo e o resto para testes, mantendo a estratificação dos dados, para garantir que todas as classes existam nas duas divisões do *dataset*. Cada classificador foi rodado 10 vezes para garantir que o resultado obtido seja confiável.

3.2.6 Avaliação dos classificadores

Por fim, na etapa de avaliação, foram aplicadas as métricas vistas na [Subseção 2.1.7](#) para todos os casos dos classificadores, analisando o desempenho e comparando-os com os resultados obtidos no trabalho de IC.

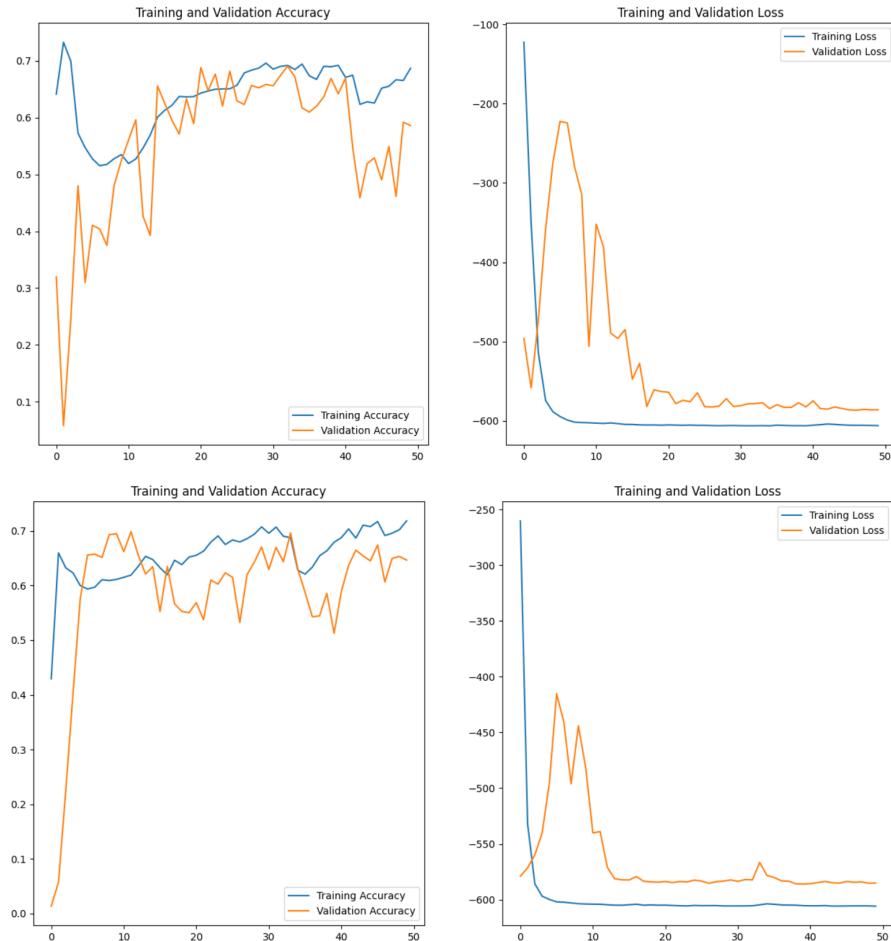
4 RESULTADOS

Neste capítulo são apresentados e discutidos os resultados obtidos separados em: resultados da segmentação e da classificação.

4.1 Segmentação

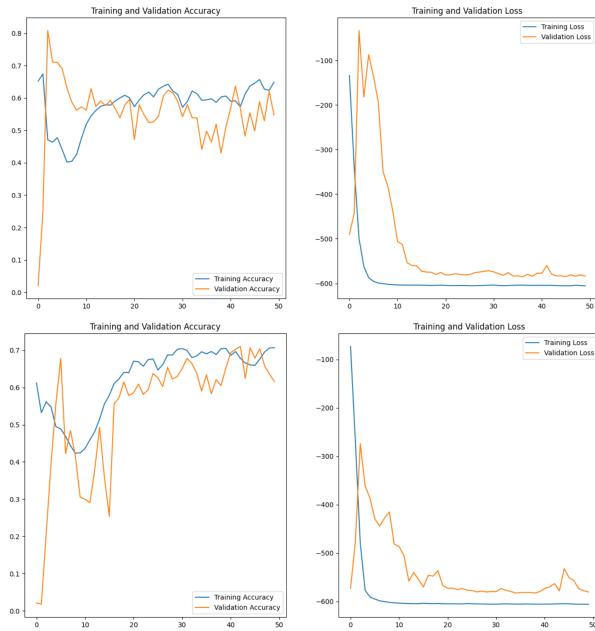
Inicialmente, em relação à etapa de segmentação das imagens utilizando a rede *U-net*, foram obtidos os resultados que podem ser vistos nas [Figura 9](#), [Figura 10](#) e [Figura 11](#).

Figura 9 – (a) Acurácia *U-net* 128x128 RGB (b) Loss *U-net* 256x256 RGB (c) Acurácia *U-net* 256x256 RGB (d) Loss *U-net* 256x256 RGB



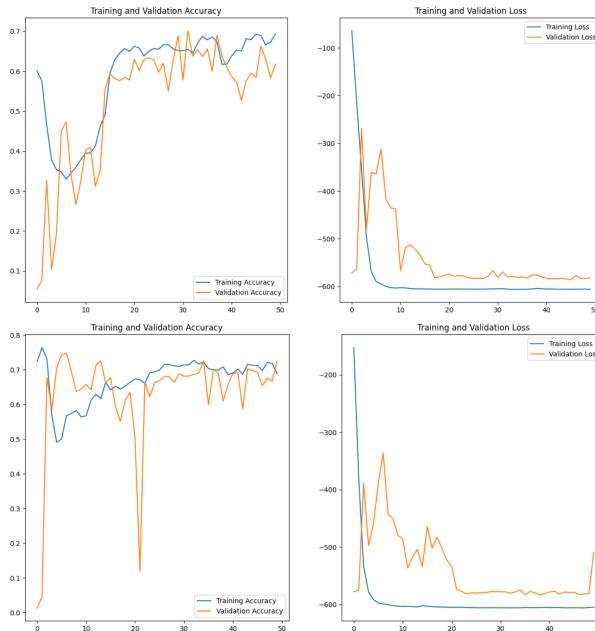
Fonte: Elaborado pelo autor

Figura 10 – (a) Acurácia *U-net* 128x128 cinza (b) *Loss* *U-net* 256x256 cinza (c) Acurácia *U-net* 256x256 cinza (d) *Loss* *U-net* 256x256 cinza



Fonte: Elaborado pelo autor

Figura 11 – (a) Acurácia *U-net* 128x128 *K-means* (b) *Loss* *U-net* 256x256 *K-means* (c) Acurácia *U-net* 256x256 *K-means* (d) *Loss* *U-net* 256x256 *K-means*



Fonte: Elaborado pelo autor

Os valores de acurárias em teste da rede *U-net* pode ser vista na [Tabela 1](#).

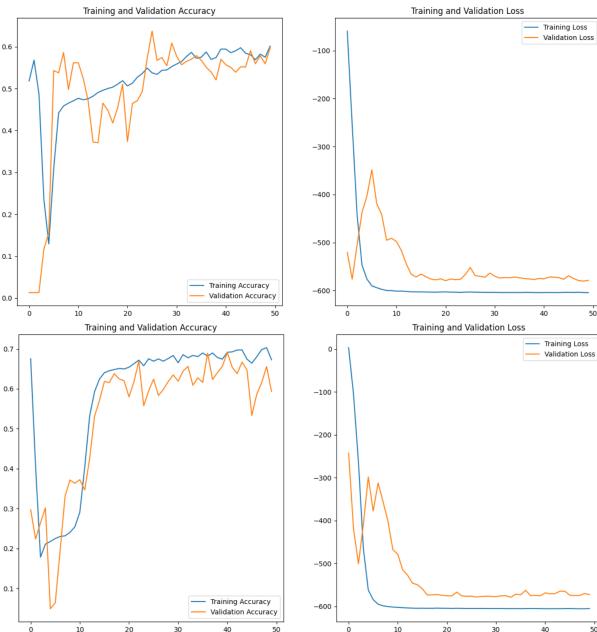
Tabela 1 – Tabela de acuráncias em teste da rede *U-net*.

	RGB	Cinza	K-means
128x128	57%	55%	62%
256x256	64%	62%	72%

Fonte: Elaborado pelo autor

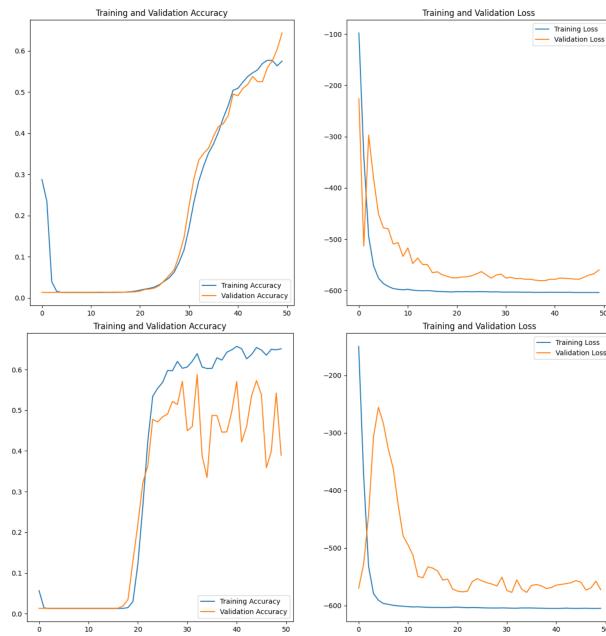
Percebe-se com a [Tabela 1](#) que os valores de acurácia ficaram, em média, próximos dos 62% para a rede *U-net*, existem alguns fatores a serem analisados para explicar este resultado, o primeiro deles é que de acordo com as [Figura 9](#), [Figura 10](#) e [Figura 11](#), pode-se ver que as acuráncias do conjunto de treinamento obtiveram um valor médio próximo dos 69%, o segundo, é que de acordo com os valores de acurácia ao longo das épocas, pode se perceber que elas tendem a convergir próximo da vigésima época. Ambos indícios apontam que a rede está, possivelmente, caindo em um *underfitting*, ou seja, não existem valores suficientes de dados no *dataset* para que o classificador possa obter resultados melhores.

Já no caso da rede *Linknet*, foram obtidos os resultados que podem ser vistos nas [Figura 12](#), [Figura 13](#) e [Figura 14](#).

Figura 12 – (a) Acurácia *Linknet* 128x128 RGB (b) Loss *Linknet* 256x256 RGB (c) Acurácia *Linknet* 256x256 RGB (d) Loss *Linknet* 256x256 RGB

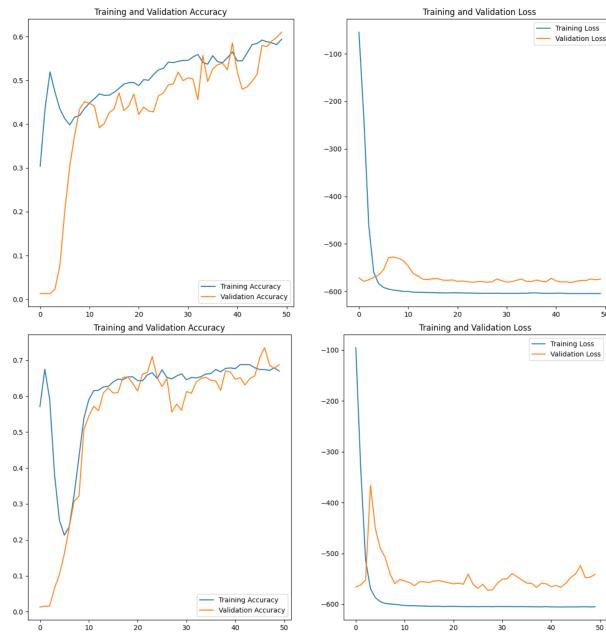
Fonte: Elaborado pelo autor

Figura 13 – (a) Acurácia *Linknet* 128x128 cinza (b) Loss *Linknet* 256x256 cinza (c) Acurácia *Linknet* 256x256 cinza (d) Loss *Linknet* 256x256 cinza



Fonte: Elaborado pelo autor

Figura 14 – (a) Acurácia *Linknet* 128x128 *K-means* (b) Loss *Linknet* 256x256 *K-means* (c) Acurácia *Linknet* 256x256 *K-means* (d) Loss *Linknet* 256x256 *K-means*



Fonte: Elaborado pelo autor

As acurárias obtida na rede *Linknet* podem ser vistas na [Tabela 2](#).

Tabela 2 – Tabela de acurácia em teste da rede *Linknet*.

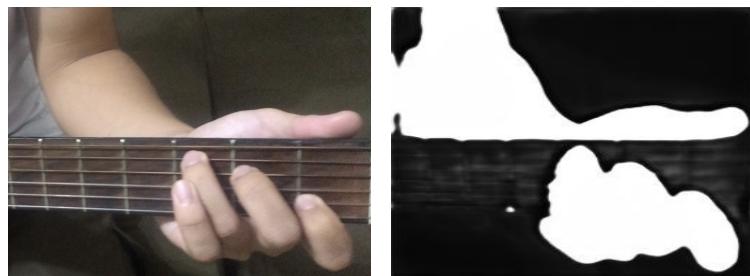
	RGB	Cinza	K-means
128x128	60%	64%	62%
256x256	59%	39%	69%

Fonte: Elaborado pelo autor

Analizando os valores da [Tabela 2](#), percebe-se que o valor máximo de acurácia foi de 69%, utilizando as tonalidades agrupadas com a resolução 256x256, e a média dos valores ficou em aproximadamente 59%, cerca de 10% abaixo da média na *U-net*. Nesta rede, também pode se perceber um comportamento semelhante ao ocorrido na rede *U-net*, na qual os valores começam a convergir perto das primeiras 20 épocas, o que é um indício que reforça que estes resultados são impactados pelo efeito do *underfitting*.

O algoritmo selecionado foi o *U-net* com o conjunto de imagens agrupadas pelo *K-means* na resolução 256x256 por ter obtido o maior valor de acurácia dentre os métodos testados. Um exemplo, com acorde dó, do resultado desta segmentação com pode ser visto na [Figura 15](#).

Figura 15 – (a) Acorde dó original (b) Acorde dó segmentado



Fonte: Elaborado pelo autor

4.2 Classificação

Em relação à classificação das imagens, foram testadas as técnicas de *boosting*, *stacking*, *bagging* e *deep learning*. Os resultados podem ser vistos nas [Tabela 3](#), [Tabela 4](#), [Tabela 5](#) e [Tabela 6](#).

Tabela 3 – Tabela de acurácia em teste do método *bagging*

	11 feat	9 feat	7 feat	5 feat
Random Forests	47%	48%	49%	52%
Extra Tree Classifier	48%	49%	50%	53%
Multilayer Perceptron	45%	47%	45%	47%
SVM	49%	49%	48%	48%

Fonte: Elaborado pelo autor

Tabela 4 – Tabela de acurárias em teste do método *boosting*

	11 feat	9 feat	7 feat	5 feat
Boosting	50%	53%	53%	51%

Fonte: Elaborado pelo autor

Tabela 5 – Tabela de acurárias em teste do método *stacking*

	11 feat	9 feat	7 feat	5 feat
Stacking	50%	47%	47%	45%

Fonte: Elaborado pelo autor

Tabela 6 – Tabela de acurárias em teste do método *deep learning*

	original	segmentada	K-means
Deep Learning	58%	67%	53%

Fonte: Elaborado pelo autor

Pode-se perceber, pela análise das acurárias que o método de *deep learning* utilizando a imagem segmentada obteve a maior acurácia, chegando aos 67%. Observando este classificador em relação à assertividade de cada rótulo individualmente, é possível se perceber que, de acordo com a Tabela 7, que o acorde de FA# teve maior resultado de *f1-score*, o que indica o melhor equilíbrio entre a precisão e *recall*.

Tabela 7 – Métrica por rótulo do classificador *deep learning* em imagens segmentadas

	precisão	recall	f1-score
DO	67%	74%	70%
FA	71%	43%	54%
FA#	88%	82%	85%
LA	50%	69%	58%
MI	71%	57%	63%
RE	81%	67%	73%
SI	71%	92%	80%
SOL	41%	64%	50%

Fonte: Elaborado pelo autor

Para efeitos de comparação, o maior valor de acurácia obtido no trabalho de IC foi de 97% em um algoritmo de *boosting*, enquanto que neste projeto, o maior valor alcançado pelo mesmo método foi de 53%. No caso do menor valor, no trabalho de IC. Esta diferença nos resultados é uma consequência do desempenho na etapa de segmentação da imagem, uma vez que, todas as características são extraídas dela.

O resultado alcançado na segmentação pode ser explicado pelo efeito da Maldição da Dimensionalidade, que descreve a relação entre a quantidade de instâncias no *dataset*, com a quantidade de características disponíveis, na qual, conforme mais características são usadas, a quantidade de dados no *dataset* deve aumentar exponencialmente para garantir uma boa representatividade dos dados. Esta característica é muito evidenciada no algoritmo de *deep learning*, pois, no caso deste tipo de classificador, a própria imagem é usada no *input* da rede, fazendo com que cada *pixel* seja analisado como uma característica por si só.

5 CONSIDERAÇÕES FINAIS

Pôde-se perceber com o desenvolvimento deste projeto, que os resultados obtidos acabaram alcançando valores menores dos que foram apresentados no trabalho de IC, como foi visto no [Capítulo 4](#), pode-se apontar como sendo um dos motivos a baixa quantidade de imagens disponíveis no *dataset* para que os algoritmos de *deep learning* pudessem ajustar seus pesos de uma forma mais assertiva.

Para se contornar este problema, existem algumas propostas que podem ser testadas, uma delas seria o próprio aumento no *dataset*, inserindo novos dados, lembrando-se de manter a diversidade para não permitir que os classificadores fiquem viciados no conjunto de treinamento e sofram de *overfitting*. Um possível segundo método seria testar o aumento de dados através de transformações nas imagens, como por exemplo, utilizando rotação no próprio eixo da imagem, aplicação de zoom ou de filtros, sempre tomando cuidado para não permitir que o algoritmo caia em *overfitting*.

Referências

- ALBUQUERQUE, M. P. de; ALBUQUERQUE, M. P. de. Processamento de imagens: métodos e análises. **Rio de Janeiro, Brasil**, v. 12, 2000. Citado na página 3.
- BACKES, A. R.; JUNIOR, J. J. d. M. S. **Introdução à visão computacional usando Matlab**. [S.I.]: Alta Books Editora, 2019. Citado na página 3.
- BARELLI, F. **Introdução à Visão Computacional: Uma abordagem prática com Python e OpenCV**. [S.I.]: Editora Casa do Código, 2018. Citado 3 vezes nas páginas 1, 3 e 4.
- BEZERRA, E. Introdução à aprendizagem profunda. **Artigo–31º Simpósio Brasileiro de Banco de Dados–SBBD2016–Salvador**, 2016. Citado 2 vezes nas páginas 11 e 14.
- CHAURASIA, A.; CULURCIELLO, E. Linknet: Exploiting encoder representations for efficient semantic segmentation. In: IEEE. **2017 IEEE Visual Communications and Image Processing (VCIP)**. [S.I.], 2017. p. 1–4. Citado na página 13.
- COMA, A. M. Visual recognition of guitar chords using neural networks. 2020. Citado 2 vezes nas páginas 14 e 15.
- GUYON, I.; ELISSEEFF, A. An introduction to variable and feature selection. **Journal of machine learning research**, v. 3, n. Mar, p. 1157–1182, 2003. Citado 2 vezes nas páginas 6 e 10.
- ISHIKAWA, A. S. Detecção de rodovias em imagens digitais de alta resolução com o uso da teoria de morfologia matemática. Universidade Estadual Paulista (UNESP), 2008. Citado na página 5.
- KOTSIANTIS, S.; KANELLOPOULOS, D.; PINTELAS, P. Data preprocessing for supervised learning. **International Journal of Computer Science**, Citeseer, v. 1, n. 2, p. 111–117, 2006. Citado 2 vezes nas páginas 10 e 11.
- LEE, H. D. **Seleção de atributos importantes para a extração de conhecimento de bases de dados**. Tese (Doutorado) — Universidade de São Paulo, 2005. Citado na página 6.
- MCCORDUCK, P. et al. History of artificial intelligence. In: **IJCAI**. [S.I.: s.n.], 1977. p. 951–954. Citado na página 1.
- MONARD, M. C.; BARANAUSKAS, J. A. Conceitos sobre aprendizado de máquina. **Sistemas inteligentes-Fundamentos e aplicações**, Manole Ltda, v. 1, n. 1, p. 32, 2003. Citado 2 vezes nas páginas 6 e 7.
- NEVES, B. C. E. C. **Modelos de Predição | Ensemble Learning - Turing Talks - Medium**. 2019. <<https://medium.com/turing-talks/turing-talks-24-modelos-de-pred%C3%A7%C3%A3o-ensemble-learning-aa02ce01afda>>. Acesso em: 24 de maio de 2020. Citado 2 vezes nas páginas 8 e 9.
- NIELSEN, M. A. **Neural networks and deep learning**. [S.I.]: Determination press San Francisco, CA, 2015. v. 2018. Citado na página 11.

- PORUSNIUC, G. et al. Convolutional neural networks architectures for facial expression recognition. In: . [S.I.: s.n.], 2019. p. 1–6. Citado 2 vezes nas páginas [9](#) e [14](#).
- PUDIL, P.; NOVOVIČOVÁ, J.; KITTNER, J. Floating search methods in feature selection. **Pattern recognition letters**, Elsevier, v. 15, n. 11, p. 1119–1125, 1994. Citado na página [11](#).
- RONNEBERGER, O.; FISCHER, P.; BROX, T. U-net: Convolutional networks for biomedical image segmentation. In: SPRINGER. **International Conference on Medical image computing and computer-assisted intervention**. [S.I.], 2015. p. 234–241. Citado na página [12](#).
- SHINODA, A. M.; OLIVEIRA, C. de. Reconhecimento da forma da mão em acordes de violão utilizando processamento de imagens. **XXIV Seminário de Iniciação Científica e Tecnológica da UTFPR (SICITE 2019)**, Pato Branco-PR, 2019. Citado 2 vezes nas páginas [2](#) e [16](#).
- SIMÕES, A. d. S. **Segmentação de imagens por classificação de cores: uma abordagem neural**. Tese (Doutorado) — Universidade de São Paulo, 2000. Citado na página [5](#).
- SOUZA, T.; CORREIA, S. Estudo de técnicas de realce de imagens digitais e suas aplicações. **João Pessoa. Paraíba**, 2007. Citado na página [4](#).
- TAKAHASHI, A.; BEDREGAL, B. R. C.; LYRA, A. Uma versão intervalar do método de segmentação de imagens utilizando o k-means. **TEMA-Tendências em Matemática Aplicada e Computacional**, v. 6, n. 2, p. 315–324, 2005. Citado na página [5](#).
- YAKUBOVSKIY, P. **Segmentation Models**. [S.I.]: GitHub, 2019. <https://github.com/qubvel/segmentation_models>. Citado 2 vezes nas páginas [12](#) e [13](#).
- ZHOU, Z.-H. Ensemble learning. **Encyclopedia of biometrics**, v. 1, p. 270–273, 2009. Citado na página [8](#).