UNIVERSIDADE FEDERAL DO TOCANTINS

CÂMPUS UNIVERSITÁRIO DE PALMAS

CURSO DE CIÊNCIA DA COMPUTAÇÃO

USO DO RECONHECIMENTO FACIAL PARA BUSCA DE FRAUDES EM REDES SOCIAIS

JÉSSICA ROMERO

PALMAS (TO)

2021

JÉSSICA ROMERO

USO DO RECONHECIMENTO FACIAL PARA BUSCA DE FRAUDES EM REDES SOCIAIS

Trabalho de Conclusão de Curso II apresentado à Universidade Federal do Tocantins para obtenção do título de Bacharel em Ciência da Computação, sob a orientação do(a) Prof.(a) Dr. Eduardo Ribeiro.

Orientador: Dr. Eduardo Ribeiro

PALMAS (TO)

2021

Dados Internacionais de Catalogação na Publicação (CIP)
Sistema de Bibliotecas da Universidade Federal do Tocantins

R763u Romero, Jéssica .
USO DO RECONHECIMENTO FACIAL PARA BUSCA DE FRAUDES EM REDES SOCIAIS. / Jéssica Romero. – Palmas, TO, 2021.
95 f.

Monografia Graduação - Universidade Federal do Tocantins – Câmpus Universitário de Palmas - Curso de Ciências da Computação, 2021.
Orientador: Prof. Dr. Eduardo Ribeiro

1. Reconhecimento facial. 2. Detecção de faces. 3. Algoritmo KNN. 4. Web Crawler. I. Título

CDD 004

Elaborado pelo sistema de geração automática de ficha catalográfica da UFT com os dados fornecidos pelo(a) autor(a).

JÉSSICA ROMERO

USO DO RECONHECIMENTO FACIAL PARA BUSCA DE FRAUDES EM REDES SOCIAIS

Trabalho de Conclusão de Curso II apresentado à UFT – Universidade Federal do Tocantins – Câmpus Universitário de Palmas, Curso de Ciência da Computação foi avaliado para a obtenção do título de Bacharel e aprovada em sua forma final pelo Orientador e pela Banca Examinadora.

Data de aprovação: 22 / 12 / 2021

Banca Examinadora:

Prof. DSc. Marcelo Lisboa Rocha

Profa. Dra. Glenda Michele Botelho

A Deus por ter me dado força
para continuar, a minha família,
amigos e por todos aqueles que
acreditaram em mim.

AGRADECIMENTOS

Gostaria de agradecer primeiramente a Deus que me deu força todos os dias continuar e chegar até aqui, aos meus pais por toda paciência comigo, a meus amigos que me apoiaram e não deixaram eu desistir. Agradeço a todos aqueles que que contribuíram para meu crescimento, em especial, um agradecimento ao meu orientador por não ter desistido de mim.

RESUMO

Diante da atual era digital onde quase tudo está sendo sistematizado, pode-se encontrar criminosos atuando nessa área também. Tem se tornado crescente o número de roubos de contas de redes sociais em troca de dinheiro para devolver ou para vender um produto que não é da pessoa, fraude com perfis falsos se passando por outras pessoas ou estabelecimento e de alguma maneira causando danos aqueles que são inocentes. Mediante a esses fatores, nesse trabalho propõe-se a criação de um algoritmo capaz de coletar fotos através de *hashtags* no *instagram*, fazer o reconhecimento facial e a classificação se alguma dessas fotografias contém o indivíduo que está interessado em saber se existem possíveis fotos que possam estar lhe causando algum tipo de dano ou não. No decorrer deste trabalho é observado que para realizar a implementação está sendo usado os coletadores de dados da internet (*Web Crawler* e *Web Scraping*) junto com detecção e reconhecimento facial. Os algoritmos que auxiliaram o desenvolvimento é o CNN (Rede Neural Convolucional) e o HOG (Histograma de Gradientes Orientados) para a detecção de faces, além de se fazer uma comparação entre eles, em conjunto com o algoritmo KNN (K-vizinhos mais próximos) para o reconhecimento facial deles.

Palavra-chave: Reconhecimento facial. Detecção de faces. Fraudes. Crimes. *Web Crawler*. *Web Scraping*. Algoritmo KNN.

ABSTRACT

Faced with the current digital age where almost everything is being systematized, one can find criminals working in this area as well. There has been an increasing number of thefts of social media accounts in exchange for money to return or sell a product that does not belong to the person, fraud with false profiles impersonating other people or establishments and somehow causing harm to those who are innocent. Based on these factors, this work proposes the creation of an algorithm capable of collecting photos through *hashtags* in *instagram*, performing facial recognition and classifying if any of these photographs contain the individual who is interested in knowing if there are possible photos that could be causing you some kind of damage or not. In the course of this work, it is observed that to carry out the implementation, internet data collectors (*Web Crawler* and *Web Scraping*) are being used together with detection and facial recognition. The algorithms that helped the development are the CNN (Convolutional Neural Network) and the HOG (Oriented Gradient Histogram) for the detection of faces, in addition to making a comparison between them, together with the KNN (K-nearest neighbors) algorithm.) for their facial recognition.

Keywords: Facial recognition. Face detection. Frauds. Crimes. *Web Crawler*. *Web Scraping*. kNN Algorithm.

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

A origem da internet se deu durante a Guerra Fria, em outubro de 1969 (UOL, 2021), para fins militares, essa aparição trouxe um nova era conhecida como era digital. Desde então, ao longo de todos esses anos muitas coisas evoluíram, outras foram criadas e outras estão em constante transformação e adaptação. Durante esses anos pode-se destacar a criação das redes sociais, que se tornaram um meio de comunicação aberta para todo planeta. Assim, uma pessoa do Brasil pode se comunicar instantaneamente com alguém da china. A primeira rede social, dentro dos moldes que se espera, foi criada em 1995, chamada *Classmates.com*, entretanto, a primeira que tornou-se massivamente popular foi o *Orkut*, com milhões de usuários (GOGONI, 2019b), sendo substituída anos mais tarde pelo *Facebook* em primeiro lugar e depois outras redes sociais como *Instagram*, *Twitter*, *WhatsApp*, *LinkedIn*, *TikTok* entre outros.

Essas malhas de comunicação têm se tornado cada vez mais popular por disponibilizar a opção de se fazer novas amizades, podendo até virar relacionamentos sérios e em alguns casos levando a casamentos. Outra alternativa tem sido a opção de vários empreendimentos de vendas *on-line*. Além disso, os usuários utilizam deste meio para expor suas opiniões e sentimentos a respeito de algo ou alguém, fora a exibição de fotos e vídeos de diversos conteúdos (ATHENIENSE, 2010).

Além das formas de comunicação, muitas empresas têm expandido seus serviços para serem acessados na plataforma *on-line* com a finalidade de facilitar o acesso para os seus clientes. No final do ano de 2019 surgiu um novo vírus na China, conhecido como novo Coronavírus ou *COVID-19*, com sua alta taxa de transmissão se alastrou por todo planeta em alguns meses, devido a isso foi adotado métodos para conter o avanço por conta de sua letalidade e por desconhecimento de como reage em cada pessoa. Dessa maneira, uma das medidas foi o isolamento social, com isso muitos comércios tiveram de ser fechados por determinado tempo ou adotado o sistema de vendas *on-line* com entrega por *deliveries*. Por consequência disso, houve um aumento em acessos a aplicativos e *sites* para fazer compras.

O *Instagram* vem se popularizando cada vez mais, com números de usuários cada vez maiores a cada ano que se passa. Com esse crescente número de acesso, empresas estão aplicando táticas como promoções e publicando nessa rede social e em outras. Esse método já existia antes da pandemia, porém tem se usado cada vez mais. Entretanto, existem duas vertentes, uma no qual se usa a internet e tudo que tem disponível nela para o bem e a outra que se usa para o mal. Com o decorrer dos dias os ataques cibernéticos e as *fake news* tornam-se mais frequentes. Os ataques as empresas, aos órgãos públicos e privados com roubo de dados e informações confidenciais e importantes, onde aqueles que roubaram pedem dinheiro em permuta da devolução das informações e não postagem dos

dados na internet (BA, 2021).

Criminosos têm se passado de empregados de certas empresas, criando uma promoção falsa e/ou enviando uma mensagem dizendo que um cliente ganhou uma promoção que o estabelecimento estava disponibilizando, o indivíduo que tornou-se o felizardo da vez para receber o prêmio teria que passar os dados pessoais, incluindo CPF (*Cadastro de Pessoas Físicas*), registro de identidade (*RG*) e endereço para mandar o produto, depois pedem para que enviem um código que o cidadão irá receber, eles informam que essa chave será necessária para a validação da promoção, entretanto os números recebidos são para clonar o *WhatsApp* do mesmo e pedir dinheiro aos números de contatos encontrado por lá. Isso gera um transtorno enorme para todos aqueles que são ludibriados e acabam ficando no prejuízo de alguma maneira (CLIENTES, 2021).

Muitos artistas e pessoas anônimas fazem uso da rede social *Instagram*, do *Twitter*, entre outras postando coisas sobre seu cotidiano, fotos de trabalhos ou momentos de lazer e com isso adquirem muitos seguidores que gostam ou se interessam pelas publicações. Outras pessoas não gostam de se expor, criando assim perfis falsos, lembrando-se que a maneira que esse perfil é conduzido dirá se é crime ou não, ou seja, pode-se usar imagens de pessoas não existentes, personagens, animais, mas se forem de pessoas existentes poderá apenas estar infringindo algumas regras de Termos de Serviços do *site*. Caso haja a denúncia, o dono do perfil falso poderá ter apenas o perfil excluído. Entretanto, se o infrator fizer mau uso da imagem de alguém, de forma que venha a causar danos a pessoa, será julgado e caso tenham as provas e seja comprovado o indivíduo prejudicado poderá obter uma indenização e o violador uma punição (UOL, 2019; BARBOSA; TEIXEIRA, 2020).

Portanto, seja criando um perfil falso de uma pessoa real, viva ou morta, o ser a quem o criou pode cometer o crime de falsidade ideológica, caso venha a causar algum tipo de danos a vítima. Ou seja, o sujeito está passando-se por uma outra pessoa e inserindo declarações ou imagens que venha a prejudicar a outrem. Em caso de utilizar as imagens de terceiros, só torna-se um ato legal se possuir o consentimento da pessoa com autorização por escrito. Caso o contrário, o transgressor estará violando o Artigo 5º da Constituição federal que protege o direito de personalidade, como honra, intimidade, vida privada e imagem e o Artigo 307º do Código penal que diz, respetivamente:

> Art. 5º - ″Todos são iguais perante a lei, sem distinção de qualquer natureza, garantindo-se aos brasileiros e aos estrangeiros residentes no País a inviolabilidade do direito à vida, à liberdade, à igualdade, à segurança e à propriedade, nos termos seguintes:
> V – é assegurado o direito de resposta, proporcional ao agravo, além da indenização por dano material, moral ou à imagem;
> X – são invioláveis a intimidade, a vida privada, a honra e a imagem das

> pessoas, assegurado o direito a indenização pelo dano material ou moral decorrente de sua violação (PLANALTO.GOV.BR, 2016);″

O artigo 307º do Código penal se trata da falsa identidade e a pena que pode-se pegar em caso de comprovar-se o crime desse porte cometido pelo infrator, desse modo diz o seguinte:

> Art. 307 - ″Atribuir-se ou atribuir a terceiro falsa identidade para obter vantagem, em proveito próprio ou alheio, ou para causar dano a outrem: Pena - detenção, de três meses a um ano, ou multa, se o fato não constitui elemento de crime mais grave (PLANALTO.GOV.BR, 2016).″

Por conseguinte, diante desses fatos e dos transtornos que geram, propõe-se uma aplicação onde faz o uso de algumas aplicações desenvolvidas nesses anos da era digital como o reconhecimento facial em uma foto ou vídeo de uma pessoa, com a captura dos detalhes do rosto, o *Web Scraping* e *web crawler* que fará uma varredura em *sites* da internet na busca de saber se o rosto individuo está sendo usado em outro lugar de forma ilegal e que venha a causar danos ao cidadão. Os algoritmos que foram usados como auxilio são os: CNN (Rede Neural Convolucional), o HOG (Histograma de Gradientes Orientados), ambos para a detecção de faces e em conjunto com o KNN (K-vizinhos mais próximos) para o reconhecimento facial. Na intenção de atuar no combate a fraudes *on-line*. Leia a seguir sobre cada uma dessas aplicações citadas (FEITOSA, 2020).

1.1 Objetivo Geral

Realizar um estudo de algoritmo de busca, detecção e reconhecimento facial com aplicação na plataforma do *instagram* que fará o reconhecimento facial e classificação do rosto que se busca com o objetivo de averiguar possíveis fotos, através de *hashtagas*, que possam se categorizar como fraudes e/ou uso indevido.

1.2 Objetivos Específicos

Para organizar o processo de desenvolvimento do trabalho, são definidos os seguintes objetivos específicos:

1. Determinar o método de processamento digital de imagens usado para detectar pontos de interesse na imagem e selecione o método mais relevante para resolver os problema levantado de acordo com a literatura.

2. Identificar e selecionar bibliotecas de processamento de imagens e reconhecimento de padrões.

3. Avaliar os métodos de reconhecimento de padrões.

4. Analisar, determinar e avaliar uma abordagem para a tarefa de classificação das imagens extraídas.

5. Analisar, projetar e desenvolver um componente responsável pela detecção do pontos de interesse, reconhecimento facial e classificação das imagens.

1.3 Organização do Trabalho

A estrutura deste trabalho é detalhada a seguir.

- Capítulo 2 - é visto toda fundamentação teórica para compreensão do que é usado para a obtenção dos resultados esperados.
- Capítulo 3 - é visto o que é utilizado para se fazer a implementação.
- Capítulo 4 - é visto os resultados obtidos com o algoritmo.
- Capítulo - 5 é visto as considerações finais de todo o trabalho e ideias para dar continuidade ao trabalho futuramente.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 Princípio da Recuperação de Informação

As ferramentas de visualização de dados, bem como a análise de dados e extração de conhecimento, requerem principalmente que os dados sejam organizados de uma determinada maneira para que possam ser acessados de forma rápida e eficiente. A análise manual de grandes bancos de dados armazenados em papel impresso ou inventário torna-se muito caro e pode levar a tomar decisões incorretas. Portanto, neste caso, o armazenamento correto e automático dos dados e a recuperação eficiente da informação tornam-se cruciais.

De acordo com o método de armazenamento de dados, os dados podem ser divididos em três categorias: dados estruturados, semiestruturados e não estruturados. Dados estruturados são aqueles dados armazenados de forma rígida, geralmente definidos em uma tabela organizada de acordo com um índice específico. Normalmente, a base de dados tradicional é preenchida com este tipo de dados, conforme mostrado na figura 1. (RIBEIRO, 2021)

Figura 1 – Tipos de dados

Como exemplo pode-se citar a situação em uma área de Recursos Humanos de uma determinada empresa, onde todos os funcionários deverão ter descrições, como por exemplo nome, carga horária, idade, CPF etc. que estes devem estar armazenados em um banco da dados estruturado.

Com o advento da Internet, o banco de dados se expandiu para outros níveis. O formato de armazenamento teve de ser reconsiderado para que outros dados além dos armazenados na tabela pudessem ser organizados. Com isso, veio a surgir o banco de dados não estruturado. Conforme o nome já indica, dados não estruturados são aqueles que não possuem estrutura ou organização predefinida (RIBEIRO, 2021). Alguns exemplos são documentos em formato de texto e/ou *PDFs*, *sites* da Internet e redes sociais são todos bancos de dados que armazenam dados não estruturados, conforme mostrado na figura 1.

Também existe um tipo de dado que não possui a estrutura rígida como de um banco de dado estruturado, mas não é tão flexível e "livre" quanto os dados não estruturados. Esse tipo de dado é chamado de dados semiestruturados. Um exemplo explicado na figura 1 são os dados de linguagens de programação da Internet, como XML (*Extensible Markup Language*, que significa Linguagem de marcação em português). Esses dados são muito específicos e geralmente são processados individualmente por especialistas na área (RIBEIRO, 2021).

Em comparação com bancos de dados e arquivos tradicionais, as principais vantagens de armazenar dados em bancos de dados digitais são muitos e, atualmente, é quase obrigatório para organizações que lidam com grandes quantidades de informação. A substituição de antigos documentos e arquivos em papel por sistemas digitais possibilita um acesso mais rápido e preciso aos dados, além de sua atualização e disponibilização dos dados a qualquer momento, em tempo real e com mais segurança.

Devido à complexidade e escala de alguns bancos de dados de algumas organizações, seja porque eles foram criados e mesclados ao longo dos anos, seja por problemas no design lógico dos mesmos, a redundância de dados é natural e até aceitável. Em alguns casos, é até importante ter essa redundância para fins de segurança, como no caso de perda de dados. O que não deve acontecer é a chamada inconsistência de dados, ou seja, dados redundantes, onde os mesmos dados estão armazenados em locais diferentes e contém valores diferentes.

Um exemplo é o *backup* que nada mais é que uma cópia total ou parcial dos dados do banco de dados, armazenados separadamente em outro local (por motivos de segurança física). Com isso, pode-se garantir que ele (ou pelo menos a maior parte dele) será restaurado em caso de perda de dados. E a pessoa responsável por monitorar isso é o Administrador do Banco de Dados (DBA).

A modelagem de banco de dados é uma parte fundamental do projeto de sistemas de informação. Um sistema de informação é um mecanismo desenhado com a finalidade de coletar, processar, armazenar e transmitir dados a fim de utilizá-los como informação para usuários interessados e propor soluções para problemas (RIBEIRO, 2021).

A Internet está repleta de formas naturais de dados, formato de texto em páginas da *web* ou texto em formato de documento .pdf ou .doc. Para organizar esses dados, existem métodos chamados de tecnologias de recuperação de informação, que são a base dos motores de busca mais populares da atualidade (como *Google* e *Bing*).

Na tentativa de organizar todos esses documentos, algumas empresas desenvolveram sistemas de recuperação de informação com uma estrutura de índice. A Figura 2 mostra as três etapas principais na tentativa de organizar esse conjunto de dados não estruturados. A primeira coisa a fazer é digitalizar toda a *Web* e copiar todo o conteúdo encontrado. É nesta fase que muitos detalhes de cada página e documento são armazenados localmente, e esses detalhes serão usados posteriormente para saber quais páginas

devem aparecer primeiro nos resultados da consulta. Os índices são então construídos para que os motores de busca possam acessá-los (RIBEIRO, 2021).

Figura 2 – Funcionamento de uma buscar na *Web*

Essa verificação inicial de arquivos não estruturados é feita por algoritmos chamados de robôs ou "*crawlers*". Esses algoritmos começam com os *sites* mais visitados (chamados de *torrents* ou origem) e partem dos *hiperlinks* gerados por esses *sites* para outros *sites* e assim por diante, conforme mostrado na Figura 2. Esses documentos são então indexados e organizados para que a pesquisa possa ser realizada de maneira organizada. De modo geral, um sistema responsável pela busca de um conjunto de documentos (seja na *Web* ou não) consiste basicamente em três partes: coleta e armazenamento de documentos e um índice para processamento da mesma consulta (RIBEIRO, 2021).

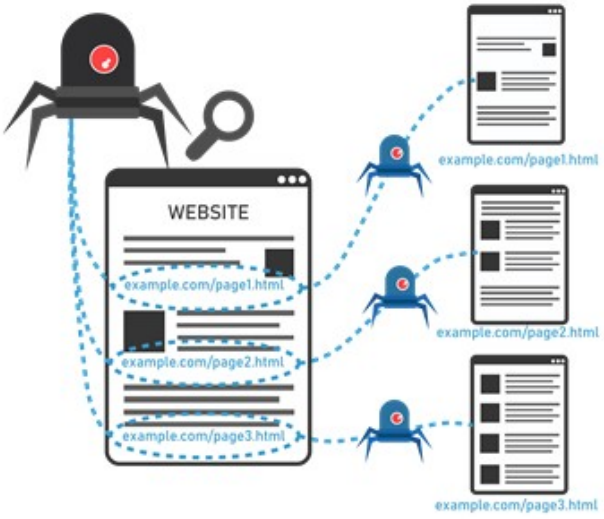


Figura 3 – Exemplo de busca indexada

O princípio utilizado pelo modelo de RI (Recuperação de Informação) mais tradicional é que cada documento pode ser representado por um conjunto de palavras que

melhor o descrevem, e esse conjunto de palavras está contido no próprio documento/*site*. Para construir o índice é muito simples. Crie um dicionário de palavras contendo todo o conjunto de palavras existentes na rede. Essas palavras serão colocadas em uma tabela. Cada linha da tabela representa uma palavra ou uma generalização da mesma palavra (como plurais, conjugados, variantes etc.) Cada coluna da tabela representará o mesmo documento, portanto, para cada arquivo, a palavra será contada, e para cada coluna da tabela, o valor da palavra encontrada aumentará. A Figura 4 mostra um exemplo de construção de um índice invertido para dois documentos (RIBEIRO, 2021).

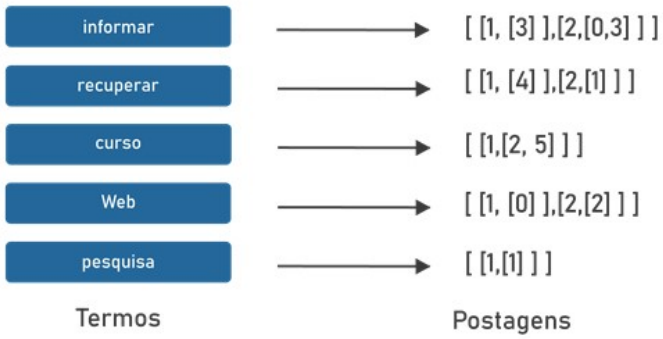


Figura 4 – Exemplo de índice.

Depois que todos os arquivos são indexados, o dicionário de pesquisa está pronto para uso. Nas linhas gerais usadas para consultas, para cada documento (linha da tabela), aquela palavra que alcançar o valor mais alto será a palavra que representa esse documento.

Exemplo, um texto onde a palavra Michael Jackson aparece 30 vezes, a probabilidade desse documento e/ou *site* ser sobre o rei do *pop* é alto. Porém se aparecer 40 vezes Madona, a probabilidade maior é sobre a rainha do *pop*.

No processo de pesquisa, o usuário define uma consulta, ou seja, um conjunto de palavras a serem pesquisadas. Em seguida, envia a consulta e usa essa para gerar novas linhas na tabela de índice para indexação. O sistema compara a linha da consulta (vetor de manuseio) com todos os outros vetores (linhas que representam o arquivo) e retorna o vetor mais semelhante ao vetor pesquisada por meio do consulta de classificação.

É válido dizer que a tarefa mais importante do mecanismo RI é poder classificar os documentos como relevantes ou irrelevantes de acordo com a consulta do usuário. Essa decisão geralmente é feita por um algoritmo de classificação, que estabelece uma ordem entre os objetos retornados. Os objetos no topo são considerados mais relevantes. Esse algoritmo opera de acordo com um conjunto de premissas que descrevem o conceito de relevância adotado pelo sistema (RIBEIRO, 2021).

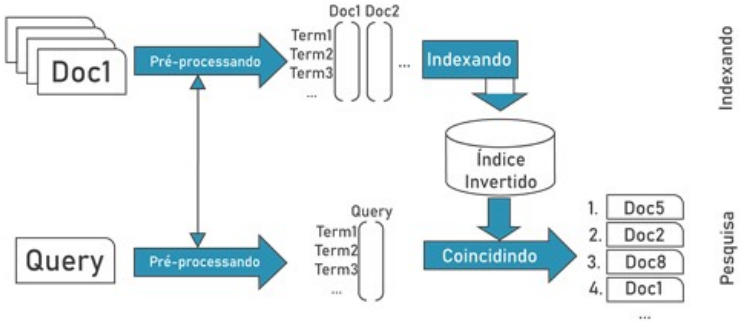


Figura 5 – Exemplo de índexação de documentos.

A Figura 5 explicita o processo de RI. No começo, antes mesmo que o processo de pesquisa possa começar, o trabalho do mecanismo é gerar uma estrutura de índice por meio do pré-processamento. A partir do momento em que é feita a indexação do documento, pode-se iniciar o processo de recuperação. Antes de tudo, o usuário define uma consulta, que é submetida ao mecanismo de recuperação por meio da linguagem de consulta fornecida pelo sistema. A consulta está sujeita às mesmas ações (mesmo pré-processamento) realizadas nos documentos indexados. Em seguida, a consulta é avaliada por um mecanismo de busca, que irá recuperar os documentos relevantes com base na consulta enviada. Antes que de serem exibidos para os usuários esses objetos passam por um processo de classificação. O processo de classificação é o que será responsável por classificar os documentos de acordo com sua significância. Depois de classificar os objetos, os usuário poderão ver em forma de resultados (RIBEIRO, 2021).

Para explicar o assunto de indexação e visualização de dados, o primeiro passo é obviamente organizar as imagens em um banco de dados ou banco de imagens.

Uma biblioteca de imagens nada mais é do que uma coleção de imagens na estrutura de dados de um banco de dados especificado ou de uma biblioteca de armazenamento. Algumas aplicações de banco de imagens já podem ser encontradas como banco de dados de imagens médicas por varreduras de raio-x e CT, arquivos de imagem de registros, arquivos digitais, sensoriamento remoto, dados geográficos, redes sociais de imagens de álbuns, entre outros. Em particular, aplicativos de banco de dados para aplicações médicas e imagens de satélite têm atraído muita atenção de projetos de pesquisa.

Uma aplicação de grande importância para a biblioteca de imagens e tecnologia de recuperação é a rede mundial de computadores (*World Wide Web*) com bilhões de imagens. Em comparação com os motores de busca de texto, os motores de busca de imagens ainda estão numa fase inicial. Esses mecanismos de texto são ferramentas poderosas para pesquisar informações em páginas de texto *WWW* (principalmente páginas construídas com linguagens de marcação, como *HTML*). Nestes casos, a ferramenta de busca de imagens realizará a recuperação do texto de uma forma clássica com base em termos que descrevem o título da imagem e até mesmo o texto em torno da imagem.

Existe um sistema denominado recuperação de imagens baseada em conteúdo (RIBC), que usa o conteúdo visual das imagens para pesquisar e recuperar as imagens mais semelhantes em um determinado banco de dados. O ponto-chave do sistema é obter dados discriminantes, que corresponderão aos possíveis julgamentos dos humanos sobre a semelhança entre a imagem pesquisada e outras imagens pertencentes à biblioteca de imagens. No sistema RIBC, as imagens são indexadas por recursos derivados diretamente de seu conteúdo visual. As informações desses atributos também são chamadas de informações de recursos de baixo nível por muitos autores, como cor, textura, forma e relacionamento espacial da imagem (RIBEIRO, 2021).

As pesquisas de imagens usando a aproximação RIBC geralmente são realizadas usando imagens de amostra ou imagens. A tarefa do sistema é recuperar a imagem mais semelhante à imagem de referência. Esse método relativamente antigo costuma ser chamado de Consulta por meio de exemplo, conforme mostrado na Figura 6.

Quando o sistema apresenta a seleção inicial da imagem de referência por meio da interface, a restauração é iniciada. O sistema pode apresentar algumas imagens representativas ou selecioná-las aleatoriamente. O usuário então seleciona uma ou mais imagens relacionadas e as imagens selecionadas são usadas para recuperar as imagens mais semelhantes a elas no banco de dados.

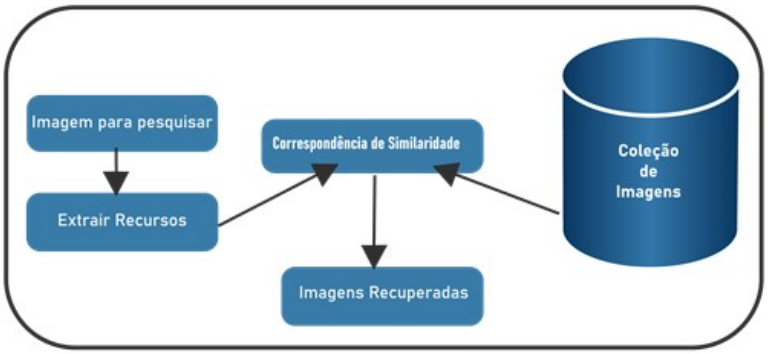


Figura 6 – Exemplo de busca e recuperação de imagens.

Essa imagem também pode ser chamada de imagem de consulta. Em seguida, o sistema faz a aceitação da seleção e categoriza a imagem para formar um vetor de características, que é armazenado em um banco de dados onde contém as características extraídas e a própria imagem. A biblioteca de imagens pode ser a *World Wide Web (WWW)* ou mais conhecida como rede mundial de computadores, neste caso em que o sistema deve saber o endereço (índice da imagem) onde a imagem está localizada para poder restaurá-la. Após a caracterização das (essas) imagens, os vetores de características existentes no banco de dados são comparados com os vetores de consulta por meio de cálculos de similaridade, que pode ser realizada de diversas maneiras utilizando a distância entre os vetores.

Com isso, as imagens mais semelhantes formam uma classificação (a primeira mais semelhante, a segunda mais semelhante, etc.) e apresentada ao usuário por meio da interface do sistema.

2.2 Rastreadores e Coletores de Páginas *Web*

Mais e mais *sites* estão sendo modernizados e tentando permanecer no topo dos resultados de pesquisa. Porém, para isso, é necessário investimento em tecnologia para alcançar um melhor posicionamento. Devido ao aumento massivo de materiais disponíveis na Internet, sua existência deve ser confirmada para se manter competitiva. Os *sites* que se classificam nos motores de busca certamente se beneficiarão.

Para isso, um mecanismo de busca é definido como um banco de dados que pode encontrar resultados com base em palavras ou termos escritos pelos usuários. Existe uma classificação do motor de busca baseada em robôs (*spiders* ou *crawlers*).

2.2.1 *Web Crawler*

É uma ferramenta bastante utilizada por sistema de busca, pode ser chamada também por *spiders* (aranha) ou rastreadores da Internet e são um tipo de robô(*bot*), geralmente operado por mecanismos de busca como *Google* e *Bing* (citado no tópico anterior). Seu objetivo é indexar o conteúdo de *sites* em toda a internet para que esses *sites* possam aparecer nos resultados dos buscadores.

Ou seja, essa ferramenta baixa o conteúdo de várias partes da internet e indexa. O objetivo de tais robôs é detectar o conteúdo de (quase) todas as páginas da internet para que as informações possam ser recuperadas quando necessário. Em outras palavras, o *crawler* faz uma varredura nas páginas da internet baixa os dados e pode armazenar de duas maneiras: o documento original compactado e indexado, catalogando-os entre mais importantes (os mais acessados) e menos importante (menos acessados). A tarefa a ser realizada é a de manter todos esses dados baixados atualizados (ROBERTO; BLUMENAU, 2006; VERZBICKAS et al., 2013).

Além disso, ele pode servir também para tarefas de manutenção de página, como verificação de *links*, validação de código *HTML* e, frequentemente, também podem ser usados para armazenar algumas informações, como endereços de *e-mail*. (Normalmente usado para *spam*).

Este software geralmente requer uma semente. A semente é uma lista de endereços a serem acessados. Quando o rastreador visitar esses endereços, todos os *hiperlinks* serão identificados e adicionados à lista de endereços a serem visitados. Sabemos que estamos lidando com uma grande quantidade de dados e a taxa de alteração desses dados também é alta. Portanto, o programa deve dar prioridade ao seu *download*, pois é provável que

seja atualizado no momento em que a página for armazenada. O *URL* armazenado deve ser acessado de acordo com um conjunto de políticas. O comportamento do *Web Crawler* é o resultado de uma combinação de estratégias, sendo elas (IME.USP.BR, 2021):

- *Política de seleção*: tomará a decisão de quais páginas será realizado o *download*;
- *Política de revisitação* : fará a verificação das mudanças ocorridas nas páginas;
- *Política de cordialidade*: terá a missão de evitar o *overloading* de *websites*;
- *Política de paralelização*: é responsável por coordenar os *bots* distribuídos.

2.2.1.1 **Política de seleção**

Devido à grande quantidade de dados na rede, armazenar todo o seu conteúdo é uma tarefa impossível. Como os rastreadores da *web* armazenam apenas uma pequena parte das informações, espera-se que essas informações sejam relevantes e não aleatórias. Esta é uma tarefa muito difícil porque deve processar parte das informações e o conjunto de páginas é desconhecido durante o processo de rastreamento.

2.2.1.2 **Política de revisitação**

A natureza da rede é muito dinâmica e o rastreamento em uma pequena parte da rede pode levar muito tempo. Quando o rastreador da *web* conclui sua tarefa, muitas coisas podem ter acontecido, como a criação de novas páginas, atualização e exclusão. Este não é um bom negócio, porque os motores de busca não serão valorizados por falta de resultados ou dados desatualizados. Do ponto de vista dos buscadores, é chamado de custo relevante. Portanto, o objetivo dos rastreadores da *web* é reduzir ao máximo a idade média das páginas armazenadas. Os dois tipos de estratégias de revisitação são (IME.USP.BR, 2021):

1. Estratégia unificada - comprometida em revisitar todas as páginas com a mesma frequência.
2. Política de proporção - dedicada a revisitar as páginas atualizadas com mais frequência.

2.2.1.3 **Política de cordialidade**

Podemos ver que os rastreadores da *web* são muito úteis para muitas tarefas. Mas seu uso tem um preço. Por exemplo, alto consumo de banda larga e sobrecarga do servidor por um longo período de tempo - especialmente quando a frequência de acesso é muito alta. Parte da solução para esse problema é o protocolo de exclusão do robô, também conhecido como *robots.txt-protocol*, que é um padrão pelo qual os administradores indicam

quais partes de seus servidores não devem ser acessadas pelos rastreadores. Esta é a maneira mais eficaz de evitar sobrecargas de servidor hoje.

2.2.1.4 **Política de paralelização**

Um rastreador pode executar vários processos em paralelo. Isso visa maximizar a velocidade de *download* e minimizar os custos de paralelização. Para evitar o *download* repetido de *URLs*, existem as seguintes estratégias (IME.USP.BR, 2021):

1. Alocação dinâmica - O servidor central aloca endereços dinamicamente para rastreadores. Permite que o sistema controle a carga de cada rastreador, e também permite a exclusão dinâmica e adição de *downloads*.

2. Alocação estática - as regras são fixadas desde o início. Como existem *links* externos que podem acessar o *site* desde o primeiro processo até o *site* atribuído ao segundo processo, deve ocorrer uma troca de *URL*.

Verazmente, o rastreador da internet fará a visita de um determinado número de páginas, pausando ao atingir o seu máximo. Para fazer a atualização, o sistema faz um novo processo de busca e quando termina faz a substituição, apagando a que existia antes e adicionando a nova. Existem métricas e métodos que podem ser utilizados para tornar mais eficientes as buscas e atualizações do banco de dados, sendo eles os algoritmos:

- *First-In-First-Out (FIFO)*: esse algoritmo faz uma busca e uma coleta mais vasta, tendo em vista atingir o número maior de *sites*;

- *Last-In-First-Out (LIFO)*: esse é uma busca e coleta mais concentrada, procura mais páginas por *site*;

- *Backlink count (Conectividade)*: essa busca o *site* mais popular, ou seja, quando mais páginas estiverem *links* que levam a ela, maior será a chance de ser a primeira a ser indexada.

Além disso, o *crawler* também pode manter a sua coleção de dados atualizadas de outras duas maneiras:

- *Crawler incremental* – sua atualização de páginas é de forma constante, ou seja, após atingir seu número máximo, ele começa a substituir aquelas menos frequentadas pelas mais frequentadas. Esse método faz com que ele visite menos aquelas de pouca importância.

- *Crawler periódico* – a cada determinado número de dias, a coleção é atualizada, fazendo assim a exclusão do anterior pelo novo. Já citado acima.

Observe a figura 7 que é um exemplo em gráfico:

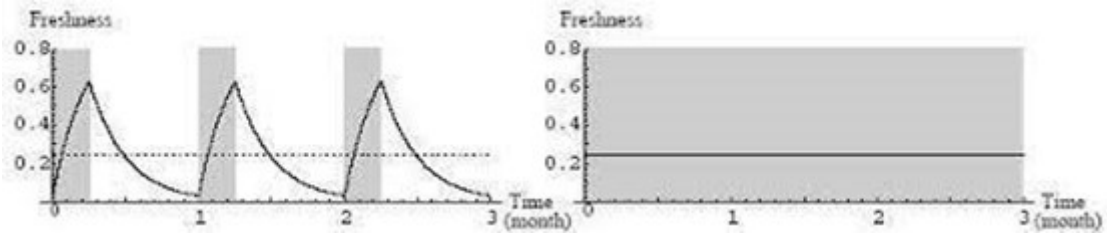


Figura 7 – Exemplo de como funciona um reconhecimento facial. Fonte: (FACERECOGNITIONSOLUTION.COM, 2014)

Nota-se que no primeiro gráfico é um caso de *crawler* periódico, onde apenas nos dias que se faz a atualização dos dados é que atinge o pico tornando a cair logo em seguida até a próxima atualização. Já no segundo, trata-se do incremental, pois mantém um padrão, uma constância de atualizações.

2.2.2 *Web Scraping*

No competitivo mundo *on-line* de hoje, todos procuram maneiras de inovar e usar novas tecnologias. A *web scraping* é uma solução para quem deseja acessar dados estruturados da *web* de maneira automatizada. O *web scraping* pode ser muito útil quando o *site* do qual você deseja obter dados não tem uma *API* ou se houver uma *API* que forneça acesso limitado aos dados.

web scraping é o processo de coleta de dados estruturados da *web* de maneira automatizada. Também é chamado de extração de dados da *web* ou coleta de dados. Alguns dos principais casos de uso de *web scraping* incluem rastreamento de preços, inteligência de preços, rastreamento de notícias, geração de *leads*, análise de mercado e no caso estudado será para busca por fraudes e/ou uso indevido ou não de fotos.

Sendo assim, *web Scraping* ou raspagem de dados é um método de extração de dados que se utiliza, como o prórpio nome já diz, para extrair dados de um determinado *site*. Com implementações e processos automatizados, se treina um robô (*bot*), esse *bot* acessa o *site* em específico e faz a "raspagem" ou coleta de dados(informações) criando assim uma cópia de todo o conteúdo onde se armazena em um banco de dados ou planilha local, para fazer-se a recuperação ou a análise futuramente.

Se você já copiou e colou informações de um *site*, sua função é a mesma de qualquer rastreador da *web*, apenas na faixa micro e manual. Ao contrário do processo de extração manual de dados, o *web scraping* usa automação inteligente para recuperar centenas de milhões ou até bilhões de pontos de dados das fronteiras aparentemente infinitas da Internet.

Essa ferramenta é muito importante para profissionais do *marketing*, por exemplo, pois eles podem fazer a coleta de dados das informações que querem, depois faz uma análise comparando com outros e facilita uma tomada de decisão melhor. Entretanto,

tem se tornado uma prática maliciosa, onde criminosos usam para roubar informações e aplicar golpes.

Embora a implementação possa ser complicada, o processo de *web scraping* é muito simples. Esse processo é dividido em três etapas:

1. No primeiro passo, o *snippet* (fragmento) de código usado para extrair as informações (o chamamos de *bot* rastreador) envia uma solicitação *HTTP GET* para um *site* específico.

2. Após a resposta do *site*, o raspador verifica o documento *HTML* na procura de encontrar um padrão específico de dados.

3. Por fim, após a extração dos dados, eles são transformados para o formato especifico o qual o autor do *bot* tenha programado.

Os *bots* raspadores podem ser arquitetados para diversas finalidades, entre elas estão: raspagem de conteúdo, raspagem de preços, raspagem de contatos (usados grande parte por golpistas), raspagem de fotos etc. O último em específico será abordado nesse trabalho, com a finalidade de combater os golpes e outros crimes relacionados (HIMAWAN; PRIADANA; MURDIYANTO, 2020), (ZHAO, 2017), (UNIVEM.EDU.BR, 2021), (CLOUDFLARE, 2021b).

2.2.3 **Diferenças entre *Web Scraping* e *Web Crawler***

O rastreamento automático é um processo executado pelos principais mecanismos de pesquisa (por exemplo, *Google*) quando enviam seus robôs (por exemplo, *Googlebot*) para indexar conteúdo da internet. Já a raspagem geralmente é construída especificamente para extrair dados de um *site* específico.

Abaixo, são listados três métodos que um robô raspador pode usar, que são diferentes do comportamento dos *spiders*:

1. O *bot Scraper* finge ser um navegador da *web*, e o *bot* mostra sua finalidade e não tenta enganar o *site* fazendo-o pensar que não é.

2. Às vezes, o raspador trabalha de forma avançada, realizando os preenchimentos ou desempenhando determinadas ações para chegar à parte específica do *site*. Já os *crawlers* não farão isso.

3. Os *scrapings* geralmente não consideram o arquivo *robots.txt*, que é um arquivo de texto que contém informações projetadas especificamente para informar aos rastreadores da Internet quais dados devem ser analisados e para evitar a visita a áreas do *site*. Como eles são projetados para extrair conteúdo específico, eles podem extrair conteúdo claramente marcado como ignorado.

Portanto, enquanto um tem a finalidade de indexar os *sites* e páginas mais importantes (*Web Crawler*), o outro tem a intenção de baixar os dados contidos em um *site* específico e de que tem relevância (*Web Scraping*).(CLOUDFLARE, 2021a).

Pode-se observar no exemplo, um código simples em *Python* que faz a coleta de dados do *site Wikipedia*.

```
from bs4 import BeautifulSoup
import requests

pages_crawled = []


def crawler(url):
    page = requests.get(url)
    soup = BeautifulSoup(page.text, 'html.parser')
    links = soup.find_all('a')

    for link in links:
        if 'href' in link.attrs:
            if link['href'].startswith('/wiki') and ':' not in link['
                href']:
                if link['href'] not in pages_crawled:
                    new_link = f"https://en.wikipedia.org{link['href']}"
                    pages_crawled.append(link['href'])
                    try:
                        with open('data.csv', 'a') as file:
                            file.write(f'{soup.title.text}; {soup.h1.
                                text}; {link["href"]}\n')
                        crawler(new_link)
                    except:
                        continue


crawler('https://en.wikipedia.org')
```

2.3 Princípios de Processamento de Imagens

As imagens são produzidas por diversos equipamentos físicos, como câmeras e câmeras de vídeo, equipamentos radiográficos, microscópios eletrônicos, magnéticos e de força atômica, radar, equipamento ultrassônico, etc. A produção e o uso de imagens podem ter diferentes finalidades, desde o puro entretenimento até aplicações militares, médicas ou técnicas. Seja um observador humano ou uma máquina, o objetivo da análise de imagens é extrair informações úteis e relevantes para cada aplicação necessária.

Geralmente, a imagem pura recém-adquirida pelo dispositivo de captura precisa ser transformada e aprimorada para torná-la mais adequada para que o conteúdo de informações necessário possa ser extraído de forma mais eficaz. O processamento digital de imagens (PDI) é um campo da teoria da eletrônica/sinal em que uma imagem é convertida em uma matriz de inteiros, e cada elemento dessa matriz é composto de um elemento básico: *pixel* (abreviação de *picture element*). A partir dessa matriz de *pixels* que representa a imagem, diversos tipos de processamento digital podem ser implementados por meio de algoritmos computacionais. A aplicação desses algoritmos irá realizar as transformações necessárias, por exemplo, para obter uma imagem com os pontos brilhantes desejados ou para extrair atributos ou informações relacionadas. Neste trabalho, o PDI será considerado uma combinação de processamento de imagem e visão computacional (GOGONI, 2019a).

2.3.1 ***Pixel***

O termo *pixel* é uma combinação das duas palavras "*picture*" e "*element*", que significa "elemento de imagem". Independentemente da fonte, é a menor unidade que constitui uma imagem digital. Ao maximizar a imagem, podemos ver claramente os *pixels* na composição da imagem (GOGONI, 2019a).

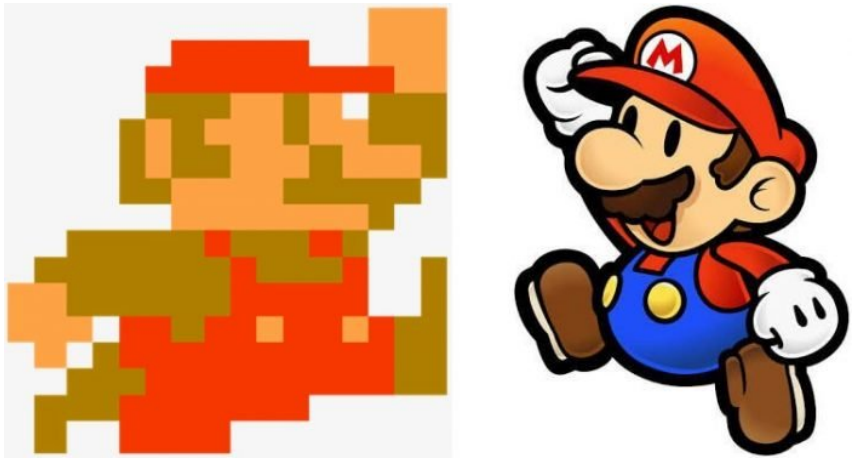


Figura 8 – Cada quadradinho do Mario à esquerda é um pixel; como o Mario à direita possui mais pixels (maior resolução), eles não são visíveis a olho nu.

Além de telas, os *pixels* também existem em fotos, animações, TVs, sensores de câmera e *smartphones*. Cada *pixel* é baseado nas três cores básicas do sistema RGB: vermelho, verde e azul. Cada uma dessas cores tem 256 tons diferentes e pode formar até 16 milhões de combinações.

São os *pixels* que juntos formam a imagem, a foto e o quadro do vídeo, e seu número implica diretamente na resolução, ou seja, na qualidade da imagem. Ou seja, quanto mais *pixel* tiver, melhor será a resolução da imagem(GOGONI, 2019a).

2.3.2 ***RGB***

RGB esse modelo é a abreviatura das iniciais em inglês *Red*, *Green*, *Blue*. O modelo de cores RGB é baseado na teoria das três cores primárias, nas quais a luz é usada para formar as cores.

A luz tem as três cores principais mencionadas acima: vermelho, verde e azul. Este modo RGB de geração de cores, quando dividido em 255 níveis, pode gerar mais de 16 milhões de cores(LEOCáDIO, 2016).

Exemplo das cores aditivas primárias:

- *Branco*: R= 255, G=255, B=255;
- *Azul*: R=0, G=0, B=255;
- *Vermelho*: R=255, G=0, B=0;
- *Verde*: R=0, G=255, B=0.

Ou seja, para que as cores se formem no sistema RGB, deve-se adicionar luz com a gradação necessária. Portanto, chamamos o sistema RGB de sistema de cores aditivas. Observe a figura 9:

Figura 9 – RGB

Exemplo das cores aditivas secundárias:

- *Amarelo*: R=255, G=255, B=0;
- *Magenta*: R=255, G=0, B=255;
- *Ciano*: R=0, G=255, B=255.

Todas as cores RGB juntas formam luz branca com a mesma intensidade e a combinação dessas duas cores forma CMY (ciano, magenta e amarelo). Este é um sistema de cores geralmente usado para *displays* e produtos eletrônicos. Desta forma, o padrão de cores RGB é adequado para produção digital. Por exemplo: *banners*, arte de *site* ou rede social, etc (LEOCáDIO, 2016).

2.3.3 ***CMYK***

Desta maneira, o CMYK é a abreviatura de Ciano, Magenta, Amarelo e Preto, como dito anteriormente. O sistema não se baseia na emissão de luz, mas na pigmentação. Este sistema é considerado um sistema de cores subtrativas e suas cores são consideradas cores subtrativas das cores primárias. Isso ocorre porque cada um representa dois aditivos principais, um dos quais foi subtraído pela luz branca. Basicamente, o objeto refletirá a cor do sistema não absorvido. Compreendendo a figura 10 (LEOCáDIO, 2016):

Figura 10 – CMYK

2.3.4 **Formação de uma imagem**

Geralmente, o arranjo das imagens na forma de uma matriz de *pixels* é feito em simetria quadrada. Isso se deve à facilidade de implementação eletrônica, seja um sistema de aquisição ou de visualização de imagens. Esse tipo de organização leva a dois problemas importantes na tecnologia de processamento. Primeiro, as propriedades de um *pixel* em todas as direções não são as mesmas, ou seja, é anisotrópico. Esta propriedade resulta em um *pixel* com quatro vizinhos de borda e quatro vizinhos diagonais, conforme mostrado na Figura 11. Este atributo exige que você defina o tipo de conexão que será usada, B4 (apenas os vizinhos da borda são considerados) ou B8 (os vizinhos da borda e diagonais são considerados). O segundo problema é uma consequência direta do primeiro problema, ou seja, para qualquer tipo de vizinho, a distância entre um ponto e seu vizi-

nho não é a mesma. O vizinho do lado é igual a 1 e o vizinho da diagonal é igual a $\sqrt{2}$ (PROCESSAMENTO, 2021).

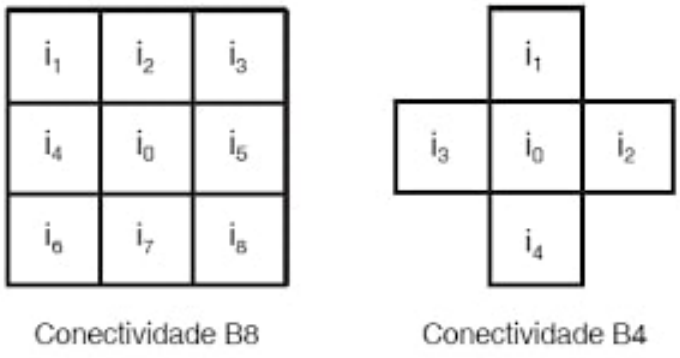


Figura 11 – Uma ilustração do tipo de conexão dos *pixels* adjacentes ao pixel central i0. A conectividade B8 tem 8 vizinhos, 4 arestas e 4 diagonais. Conectividade B4 exibe apenas *pixels* de limite.

Para fazer a captura da imagens, necessita de alguns dispositivos. O primeiro é um dispositivo físico, que deve ser sensível ao espectro de energia eletromagnética, como raio X, ultravioleta, luz visível ou espectro infravermelho. O dispositivo transdutor deve produzir um sinal elétrico proporcional ao nível de energia percebido em sua saída. O segundo tipo é chamado de digitalizador, que é um dispositivo que converte o sinal elétrico analógico gerado na saída do sensor em um sinal digital(GOGONI, 2019a).

Após a captura a imagem passa por um pré-processamento para que sua qualidade seja melhorada. Essas tecnologias envolvem duas grandes categorias: métodos que operam no domínio espacial e métodos que operam no domínio da frequência. A tecnologia de processamento de domínio espacial é baseada em um filtro que manipula o plano da imagem, enquanto a tecnologia de processamento de domínio de frequência é baseada em um filtro que atua no espectro da imagem. É bem comum fazer a junção de vários métodos para realçar um imagem. Pode-se observar a imagem 12 que passa por dois métodos de realce (GOGONI, 2019a).

Figura 12 – A primeiro figura mostra ruídos e a segunda com aplicação de métodos suavizou.

2.3.4.1 Histograma

O histograma da imagem digital é uma ferramenta muito útil no estágio de pré-processamento, pois fornece uma visão estatística da distribuição de *pixels*, contraste da imagem e nível de brilho. Além disso, os histogramas são amplamente utilizados na etapa de segmentação, principalmente em técnicas que utilizam similaridade entre *pixels*. Um histograma é frequentemente usado como uma distribuição estatística de *pixels* ("brilho") em uma imagem, por exemplo, no caso de técnicas que o utilizam para calcular a entropia da imagem.

O histograma de uma imagem digital com k níveis de cinza é definido por uma função discreta

$p(k) = \frac{n_k}{n}$

o parâmetro k representa o nível de brilho discreto, n_k representa o número de *pixels* com intensidade k na imagem e n é o número total de *pixels* na imagem, ou seja, n = M × N. Em termos simples, podemos dizer que o histograma de brilho da imagem representa a contagem dos níveis de cinza da imagem 13, o que pode informar a distribuição dos *pixels* dentro de k níveis possíveis. O histograma pode ser considerado uma função de distribuição de probabilidade, seguindo os axiomas e teoremas da teoria da probabilidade, *i.e.* que $\sum_k p(k) = 1$ (PROCESSAMENTO, 2021).

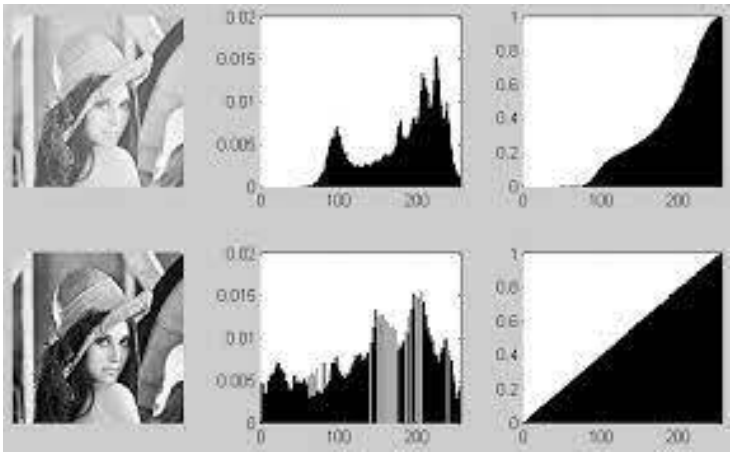


Figura 13 – Exemplo de imagens com histogramas diferenciados e com melhoramento de imagens.

2.3.4.2 **Segmentação**

Segmentar uma imagem significa que, de forma simplificada, a imagem como um todo se divide nas partes que a compõem, e são diferentes umas das outras. O ”objeto”da imagem geralmente é chamado de grupo de *pixels* de interesse ou fornece algumas informações para PDI (Processamento De Imagem). Da mesma forma, o nome da imagem *”background”* é usado para grupos de *pixels* que podem ser ignorados ou inúteis no PDI. Esses nomes ”objeto”e *”background”* têm significados muito subjetivos e podem se referir a grupos de *pixels* que formam certas áreas da imagem, em vez de representar literalmente objetos que existem na imagem processada.

Deve-se notar que não existe um modelo formal de segmentação de imagens. A segmentação é um processo empírico e adaptativo, buscando sempre se adaptar às características específicas de cada tipo de imagem e ao objetivo a ser alcançado. Embora existam muitos tipos de tecnologias de segmentação de imagem, as pessoas ainda têm um grande interesse na pesquisa e no desenvolvimento de novas tecnologias.

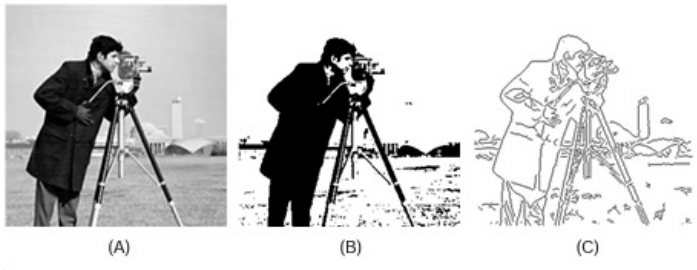


Figura 14 – Exemplo de duas abordagens para segmentação. (A) Imagem original em níveis de cinza. (B) Imagem segmentada através de uma binarização. (C) Imagem segmentada por detecção de bordas. FONTE: (PROCESSAMENTO, 2021)

De maneira geral, as técnicas de segmentação usam dois métodos principais: a semelhança entre os *pixels* e a descontinuidade entre eles. Sem dúvida, a técnica baseada

em similaridade mais comumente usada é chamada de binarização. Do ponto de vista computacional, a binarização de imagens ou limiarização de imagens é uma técnica simples e eficaz, por isso é amplamente utilizada em sistemas de visão por computador. Este tipo de segmentação é utilizado quando a magnitude do nível de cinza é suficiente para caracterizar o "objeto"presente na imagem. Na binarização, a escala de cinza é considerada o limite de separação entre os *pixels* que compõem o objeto e o fundo. Nessa tecnologia, uma imagem binária é obtida do sistema como saída, ou seja, uma imagem com apenas dois níveis de brilho: preto e branco. Determinar o limite para segmentação de imagem de maneira otimizada é o objetivo principal de vários métodos de binarização existentes.

Técnicas baseadas em descontinuidades entre *pixels* tentam determinar mudanças repentinas nos níveis de brilho entre *pixels* adjacentes. Geralmente, essas alterações permitem a detecção de grupos de *pixels* que definem os contornos ou bordas dos objetos na imagem. A técnica de segmentação mais comumente usada com base na descontinuidade é chamada de detecção de borda. A Figura 14 mostra dois exemplos de segmentação, binarização e detecção de borda. A imagem (A) é a imagem original em tons de cinza, a imagem (B) é a segmentação binarizada e a imagem (C) é a segmentação de detecção de borda.

2.3.4.3 **Pós-processamento**

O pós-processamento geralmente é uma etapa após a segmentação. É nesta fase que as principais falhas ou defeitos na segmentação são corrigidos corretamente. Geralmente, esses defeitos de segmentação são corrigidos por meio da aplicação sequencial de filtros morfológicos por meio de técnicas de morfologia matemática, que realizam análises quantitativas nos *pixels* da imagem.

2.3.4.4 **Operações Morfológicas Básicas**

Morfologia matemática (MM) é um modelo teórico de imagem digital baseado na teoria da rede e na estrutura topológica. É a base do processamento morfológico de imagens, e é baseado no operador invariante de deslocamento (invariante de translação) baseado principalmente na adição de *Minkowski* (WIKIMEDIA, 2005). Todos os métodos descritos por MM baseiam-se basicamente em duas linhas: os operadores *booleanos* de conjuntos (união, intersecção, complemento, etc.) e o conceito de formas básicas denominadas "elementos estruturais". As operações são sempre realizadas entre imagens e elementos estruturais. A forma do elemento estrutural depende do tratamento requerido e do tipo de conexão utilizada (B4 ou B8).

Dois operadores básicos são usados na maioria das técnicas de MM: corrosão e dilatação. Por exemplo, vamos pensar no objeto X como um conjunto de *pixels* x separados por linhas tracejadas, conforme mostrado na Figura 15. A operação de erosão envolve a

eliminação do *pixel* x do conjunto X em função do elemento estruturante e a dilatação envolve a dilatação do *pixel* x do conjunto X (PROCESSAMENTO, 2021).

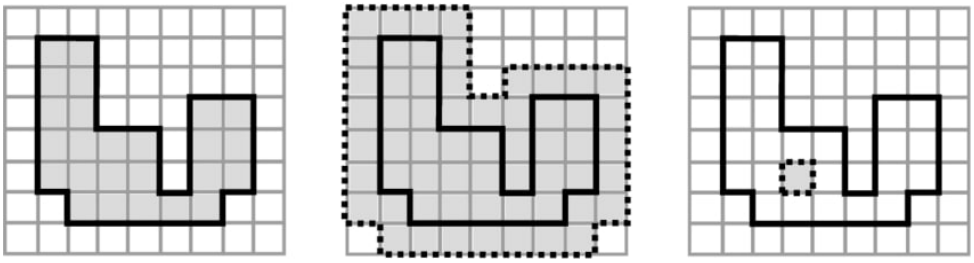


Figura 15 – Exemplo de dilação e erosão. Imagem original, imagem com dilação em relação a original e imagem com erosão em relação a original.

Esse modelo foi originalmente desenvolvido para imagens binárias, consideradas um subconjunto de Z^2 (ou Z^d, para qualquer dimensão d), e mais tarde foi estendido com sucesso para imagens em tons de cinza. Hoje, a morfologia matemática tem muitas aplicações e pode até ser usada para imagens coloridas (WIKIMEDIA, 2005).

2.3.4.5 **Extração de Atributos**

A última etapa do sistema de processamento de imagem é extrair informações úteis da imagem processada. Quando o objetivo do processamento é a obtenção de informações numéricas, os atributos da imagem são extraídos (PROCESSAMENTO, 2021).

2.3.4.6 **Rotulação ou Labelização**

A etapa chamada marcação ou rotulação é uma etapa intermediária na extração de atributos. Após a etapa de segmentação, obtém-se uma imagem na qual a área correspondente ao "objeto"é separada da área correspondente ao "fundo"da imagem. Nesse ponto do sistema de processamento, as áreas de interesse são agrupadas consecutivamente por *pixels* que se tocam. A próxima etapa é dar a cada grupo de *pixels* uma *label* (ou rótulo). Este reconhecimento permitirá a parametrização de objetos de segmentos, calculando parâmetros específicos para cada região de *pixel* contígua, como área ou perímetro. A Figura 16 mostra um exemplo dessa técnica para imagens compostas por células bem espaçadas. O processo de segmentação separa a área pertencente à célula da área (fundo) entre as células, criando um separador entre elas. A etapa de "marcação"cria uma etiqueta para identificar cada uma dessas áreas de modo que o seguinte processamento de informações seja concentrado em cada área onde a etiqueta é recebida.

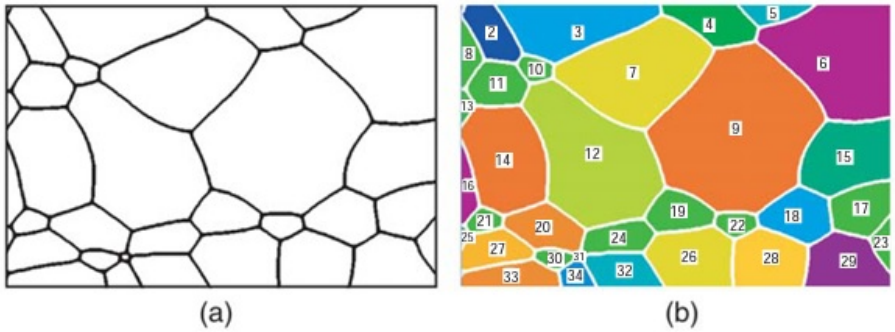


Figura 16 – Exemplo de rotulação. (a) Imagem original. (b) Imagem final após o processo de rotulação. FONTE: (PROCESSAMENTO, 2021)

2.3.4.7 **Atributos da Imagem**

Existem basicamente dois tipos de métricas: i) os atributos da imagem como um todo (características do campo), como o número de objetos, a área total dos objetos e assim por diante. ii) Referir-se de forma independente às características regionais do objeto, como área, perímetro, forma, etc. Os atributos de área podem ser muito complexos, permitindo que os objetos sejam separados em classes semelhantes com base em parâmetros de medição. A Figura 16 mostra os principais atributos da área que podem ser extraídos da imagem após as etapas de segmentação e pós-processamento (PROCESSAMENTO, 2021).

2.3.4.8 **Classificação e Reconhecimento**

O objetivo da identificação é realizar automaticamente o ”reconhecimento”dos objetos segmentados na imagem. O processo de classificação de formas tem duas etapas: aprendizagem e reconhecimento em si. Na maioria dos sistemas de reconhecimento de forma, os parâmetros da etapa de extração de atributo são usados para construir um espaço de medição N-dimensional. O sistema de aprendizagem irá definir uma função discriminante que pode efetivamente separar todas as formas representadas neste espaço de medição.

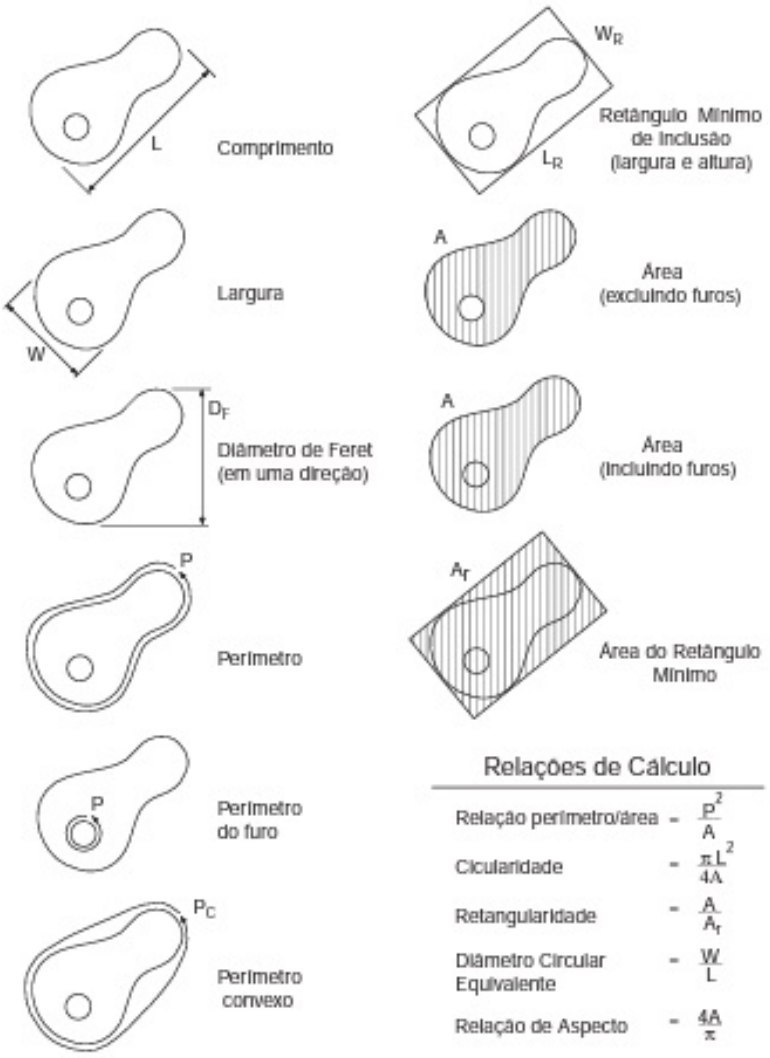


Figura 17 – Exemplo dos principais atributos de região, ou seja, dos objetos independentes presentes na imagem. FONTE: (PROCESSAMENTO, 2021)

Podemos dividir o processo de aprendizagem em duas categorias: métodos supervisionados e métodos não supervisionados. No método supervisionado, o classificador funciona sob a supervisão de outro sistema de reconhecimento (por exemplo, externo), que identificou previamente o objeto de teste e permitirá a construção correta de seu espaço de medição e de sua função discriminante. Nesse processo, devemos modificar os parâmetros que constituem o espaço de medição e permitir um melhor ajuste da função discriminante, sempre visando fazer com que o sistema execute o processo de classificação de forma mais eficaz. Finalmente, a função discriminante responsável por separar diferentes categorias pode ser determinada. Este processo pode ser lento e caro do ponto de vista computacional. Somente assim o objeto desconhecido pode ser fornecido ao classificador na fase de reconhecimento.

Sem classificação supervisionada, ou seja, não supervisionada, o classificador receberá objetos desconhecidos e tentará atribuí-los a diferentes classes com base na medição de diferentes parâmetros (atributos de objetos presentes na imagem). O reconhecimento de classe geralmente é realizado a partir do reconhecimento de grupo em *"clusters"* de objetos no espaço de medição.

Atualmente, existem vários métodos de reconhecimento de forma. Entre eles, podemos citar métodos baseados nas propriedades estatísticas dos objetos (classificadores *bayesianos*), métodos baseados na distância entre os objetos nas imagens e suas formas padrão (como as redes neurais artificiais), e até mesmo através de dicionários ou linguagens básicas (PROCESSAMENTO, 2021).

Podemos citar dois algoritmos, sendo:

- **Algoritmo SIFT** (*Scale-invariant feature transform*, em português significa Transformação de recurso invariável de escala) - é um algoritmo de visão computacional usado para detectar, descrever e combinar características locais em imagens. Foi inventado por David Lowe em 1999. As aplicações incluem reconhecimento de objetos, mapeamento e navegação de robôs, imagens de costura, modelagem 3D, reconhecimento de gestos, rastreamento de vídeo, reconhecimento individual de vida selvagem e correspondência móvel. Primeiramente, os pontos-chave do SIFT de um objeto são extraídos de um conjunto de imagens de referência e armazenados no banco de dados. Comparando cada característica da nova imagem com o banco de dados e procurando por características de correspondência candidatas de acordo com a distância euclidiana de seu vetor de característica, o objeto é identificado na nova imagem. A partir do conjunto completo de correspondência, identifica um subconjunto de pontos-chave consistentes com o objeto e sua posição, proporção e direção na nova imagem para filtrar boas correspondências. Uma implementação de tabela de *hash* eficiente usando a transformação de Hough generalizada pode determinar rapidamente *clusters* consistentes. Cada *cluster* de 3 ou mais recursos consistentes com o objeto e sua postura passa por uma verificação de modelo mais detalhada e, em seguida, os *outliers* são descartados. Finalmente, dada a precisão do ajuste e o número de possíveis correspondências falsas, calcula a probabilidade de que um determinado conjunto de características indique a existência do objeto. As correspondências de objeto que passam em todos esses testes podem ser identificadas como corretas com alta confiança.

- **Algoritmo SURF** (*Speeded up robust features*, em português Recursos Robustos Acelerados) - esse algoritmo tem uma abordagem muito similar ao SIFT, porém, encontra os pontos de interesse e os descreve computacionalmente mais rápido. Para detectar pontos de interesse, o SURF usa a aproximação inteira do determinante do detector de *blob de Hessian*, que pode ser calculado por 3 operações inteiras usando

uma imagem integral pré-calculada. Seu descritor de recurso é baseado na soma das respostas da *wavelet Haar* em torno do ponto de interesse. Eles também podem ser calculados com a ajuda de imagens integrais. Usa a tecnologia de pirâmide de resolução múltipla para converter a imagem em coordenadas para replicar a imagem original em formato de pirâmide *gaussiana* ou *laplaciana* para obter uma imagem do mesmo tamanho, mas com largura de banda reduzida. Isso atinge um efeito especial de desfoque na imagem original, denominado espaço de escala, e garante que o ponto de interesse seja invariável em escala.

2.4 Reconhecimento de Faces

Os humanos geralmente dependem de rostos para identificar indivíduos. Por sua vez, os avanços no poder da computação nas últimas décadas agora permitem um reconhecimento semelhante, mas automaticamente. O primeiro algoritmo de reconhecimento facial usava um modelo geométrico simples, mas o processo de reconhecimento agora atingiu um nível de maturidade que o permite se apresentar como uma ciência de representação matemática complexa e processos de comparação.

O primeiro sistema de reconhecimento facial foi semiautomático e seu desenvolvimento teve origem na década de 1960. O sistema exigia que seus responsáveis localizassem características em fotos, como os olhos, o nariz entre outros, antes de o sistema fazer o cálculo da distância e a proporção de pontos de referências comuns. Em seguida, comparava este ponto de referência com os dados já existentes.

No ano de 1970, *Goldstein*, *Harmon* e *Lesk* usaram 21 marcadores subjetivos específicos - incluindo a cor do cabelo e a espessura dos lábios - para identificá-los automaticamente. O problema com essas duas soluções iniciais é que as medidas e posições deveriam ser calculadas manualmente.

Já em 1988, *Kirby* e *Sirovich* aplicaram o princípio da análise de componentes (uma técnica de computação padrão) ao problema de reconhecimento facial. Isso é considerado um marco porque mostra que menos de cem valores são necessários para codificar com precisão uma imagem de face corretamente normalizada e alinhada.

Em 1991, *Turk* e *Pentland* fizeram a descoberta que, apesar do uso da tecnologia de *eigenface*, os erros residuais ainda poderiam ser usados para detectar rostos em imagens. Essa descoberta levou à criação de um sistema automatizado de reconhecimento facial em tempo real. Este método é limitado por fatores ambientais até certo ponto, mas acabou estimulando o interesse das pessoas no futuro desenvolvimento da tecnologia de reconhecimento automático de rosto.

A tecnologia atraiu a atenção do público devido à resposta da mídia à implementação experimental do *Super Bowl* em janeiro de 2001. O experimento capturou vídeo de vigilância e o comparou com um banco de dados de conteúdo digital. A apresentação

iniciou a análise necessária de como usar a tecnologia para atender às necessidades de determinados países, levando em consideração questões sociais e de privacidade.

O algoritmo de reconhecimento facial original usava um modelo geométrico simples, mas com tempo o processo de reconhecimento atingiu um nível maduro e pode ser apresentado como uma ciência de representações matemáticas complexas e processos de comparação. O tremendo progresso e as iniciativas registradas nos últimos anos destacaram a tecnologia de reconhecimento facial. O reconhecimento de rosto está atualmente disponível para fins de verificação e identificação, apesar disso, essa tecnologia está sendo usada para combater fraudes de passaportes, identificar crianças desaparecidas, aplicar leis, dentre outros.

No momento atual, a averiguação funciona da forma em que se usa um conjunto de sistemas que utilizam algoritmos e softwares para desenhar padrões em rostos humanos. Embora existam diferenças entre as pessoas, a composição básica do rosto não mudará, e o aplicativo os trata como pontos comuns, que mudam de acordo com a complexidade do sistema. Todos os sistemas de verificação têm o mesmo princípio: devem-se usar formas geométricas e formas de algoritmo para detectar faces e, em seguida, montá-las como um quebra-cabeça.

Mediante a isso, pode-se notar que existem dois métodos principais de reconhecimento facial: métodos geométricos (com base em características) e métodos fotométricos (com base em visualização). Como os pesquisadores continuam interessados no reconhecimento facial, vários algoritmos diferentes foram desenvolvidos, três dos quais são objeto de um grande estudo na literatura de reconhecimento facial. Eles são Análise de Componentes Principais (*PCA – Principal Components Analysis*), Análise Discriminante Linear (*LDA – Linear Discriminant Analysis*) e Correspondência de Padrão de Pacote Elástico (*EBGM – Elastic Bunch Graph Matching*).

- *Análise de Componentes Principais (PCA)*
  A análise de PCA, comumente é referida como baseada no uso de *eigenfaces*, é uma técnica proposta por *Kirby* e *Sirivich* em 1988. Com esse método, a imagem a ser pesquisada e a imagem do banco de dados devem ter o mesmo tamanho e precisam ser normalizadas com antecedência para alinhar os olhos e a boca na imagem. Portanto, o método PCA é usado para reduzir o tamanho dos dados comprimindo os dados, revelando a estrutura dimensional efetiva do padrão facial reduzido.

  Essa redução na dimensionalidade remove informações inúteis e decompõe a estrutura da face em componentes ortogonais (irrelevantes), chamados de *eigenfaces*. Cada imagem de face pode ser representada como a soma (vetor de recurso) de faces de recurso armazenadas em um conjunto unidimensional. Em seguida, compare a imagem dada a ser pesquisada com a galeria e meça a distância entre seus respectivos vetores de recursos.

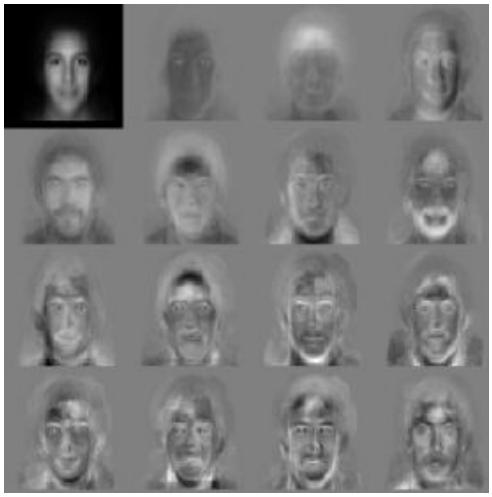


Figura 18 – Face de recurso padrão. Vetor de característica. É derivado usando *eigenfaces*.

O método PCA geralmente precisa apresentar toda a imagem frontal da face. Caso contrário, a imagem apresentará um desempenho ruim. A principal vantagem dessa tecnologia está relacionada à redução da possibilidade de identificação dos dados necessários para os indivíduos a partir de uma relação de 1/1000 (um milésimo) dos dados fornecidos.

- *Análise Discriminante linear(LDA)*:

  LDA é um método estatístico usado para classificar amostras de categorias desconhecidas com base em amostras preparadas para categorias conhecidas. Essa técnica visa maximizar a diferença entre as classes (ou seja, entre os usuários) e minimizar a diferença dentro de uma classe (ou seja, entre os usuários).

Figura 19 – Exemplo de seis classes utilizando a LDA.

Na Figura 19, cada bloco representa uma classe, as diferenças entre as classes são grandes e as diferenças dentro das classes são pequenas. Ao processar dados de faces humanas de alta dimensão, essa tecnologia enfrenta o problema do tamanho pequeno da amostra. Comparado com a dimensão do espaço da amostra, esse problema ocorre quando o número de amostras de preparação disponíveis é pequeno.

- *Correspondência de Padrão de Pacote Elástico (EBGM)*:

  EBGM é baseado na ideia de que imagens de rosto reais têm várias características não lineares que não são consideradas pelos métodos de análise linear - por exemplo, mudanças de iluminação (luz externa e luz interna), postura (postura mais rígida e luz interna). Postura mais relaxada), ou expressão (sorrindo para um rosto mais sério).

  A transformação *wavelet Gabor* cria uma arquitetura de conexão dinâmica para projetar rostos humanos em grades elásticas. *Jet Gabor* é um nó na grade elástica, marcado com um círculo na figura abaixo, que descreve o comportamento da imagem em torno de um determinado *pixel*. É o resultado da fusão de imagens com o filtro *Gabor*, que é usado para usar o processamento de imagens para detectar formas e extrair recursos. A confluência expressa o número de funções sobrepostas misturando-as. A identificação é baseada na similaridade da resposta do filtro *Gabor* de cada nó *Gabor*.

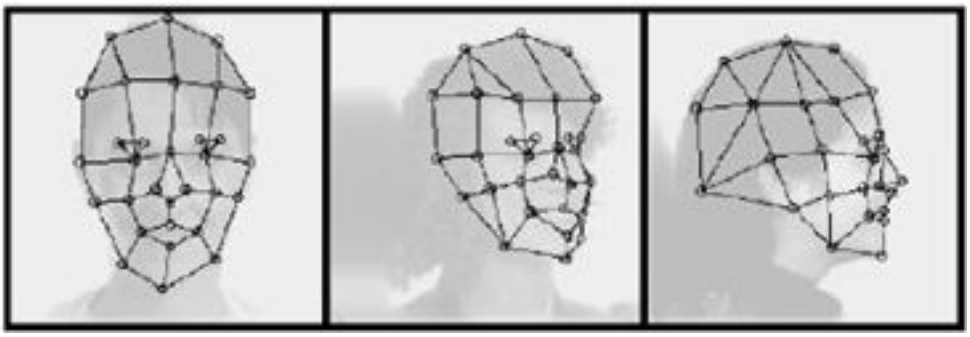


Figura 20 – *Elastic Bunch Map Graphing.*

  Este método de base biológica, usando filtros *Gabor*, é um processo realizado no córtex visual de mamíferos superiores. A dificuldade desse método reside na necessidade de localização precisa, o que às vezes pode ser alcançado combinando os métodos PCA e LDA.

De um modo geral, pode-se dar um exemplo em que com uma câmera, seja de celular, de segurança etc. usa para identificar todos ou quase todos esses pontos comuns, como os dois olhos e a distância entre eles, o nariz e seu comprimento, a boca, as bochechas e queixo, limitando assim a forma do rosto e o espaço que ocupa. Esses pontos são registrados e armazenados na forma de algoritmos em um banco de dados, que os reconhece por meio de cálculos. E então, após a busca há o retorno de encontrado ou não o rosto que se busca.

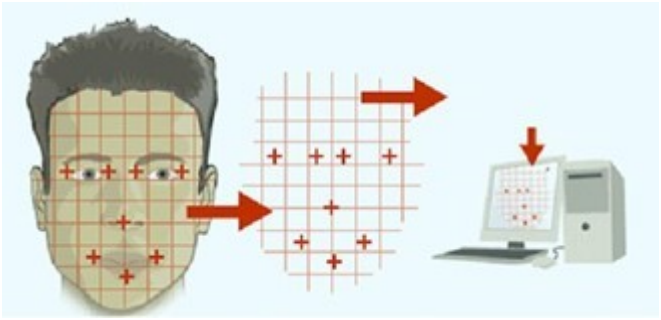


Figura 21 – Exemplo de como funciona um reconhecimento facial. Fonte: (FACERECOGNITIONSOLUTION.COM, 2014)

Com o tempo e as evoluções tecnológicas, o sistema de reconhecimento de rostos melhoraram e com as modificações de seus criadores, houve o aprendizado de que rostos mudam devido ao envelhecimento e outros motivos. Além do envelhecimento, outros fatores causariam erro ou falha na detecção como as mudanças repentinas, um exemplo seria os movimentos faciais, onde a máquina se espera que esteja voltado totalmente para ela e caso não esteja dentro do que foi treinada, seus resultados serão imprecisos. Obviamente, outras mudanças como algo simples (como na iluminação), ao usar acessórios como chapéus ou fazer caretas poderia dificultar o reconhecimento. Além disso, por exemplo, a proporção e os direitos de acesso do banco de dados tornaram-se cada vez mais poderosos, o que possibilitou "treinar" mais robôs com sucesso para que as pessoas possam se distinguir umas das outras com mais precisão. Hoje em dia, o que é capturado é o formato da cabeça do usuário, portanto, independentemente de seu ângulo em relação à câmera, seu rosto pode ser reconhecido (KLEINA, 2021).

Uma vez que, como todas as ferramentas começaram a apresentar falhas com o tempo e teve de se fazer melhoras e adaptações para tornar mais eficiente o trabalho.

Abaixo, pode-se analisar um exemplo em código da linguagem *python* com *openCV* (DEEPNOTE, 2021).

```
1  import cv2
2
3  imagem = cv2.imread("fotos/grupo.0.jpg")
4  classificador = cv2.CascadeClassifier("recursos/
       haarcascade_frontalface_default.xml")
5  imagemCinza = cv2.cvtColor(imagem, cv2.COLOR_BGR2GRAY)
6  facesDetectadas = classificador.detectMultiScale(imagemCinza,
       scaleFactor=1.2, minSize=(50,50))
7  print(facesDetectadas)
8  print("Faces detectadas: ", len(facesDetectadas))
9  for (x, y, l, a) in facesDetectadas:
10     cv2.rectangle(imagem, (x, y), (x + l, y + a), (0, 255, 0), 2)
```

```
11
12 cv2.imshow("Detector haar", imagem)
13 cv2.waitKey(0)
14
15 cv2.destroyAllWindows()
```

2.4.1 **Algoritmos de classificação**

- *Multilayer Perceptron* (Rede Neural Perceptron multicamadas) - Segundo RUSSELL e NORVIG (1995) Uma rede neural é um modelo matemático que replica a atividade mental do cérebro humano para gerar conhecimento. É formada por uma estrutura denominada neurônios artificiais, que é composta por conexões, entradas e saídas, funções de entrada e funções de saída(JHONATTAN; GHELLERE, 2015). Esse neurônio é disparado quando uma combinação linear de suas entradas excede algum limiar, ou seja, ele implementa um classificador linear. A organização das redes neurais é feita na camada de neurônios interconectados, sendo esta interconexão responsável por definir a arquitetura da rede.

- *FURIA* - A lógica *fuzzy* (ou lógica nebulosa) possui características muito diferentes da lógica tradicional, a partir dos valores possíveis que podem ser assumidos para o verdadeiro valor da preposição. Na lógica difusa, esses valores podem ser subconjuntos difusos de qualquer conjunto parcialmente ordenado. Além disso, esses valores podem ser expressos linguisticamente por meio de predicados difusos (como alto e baixo) (JHONATTAN; GHELLERE, 2015). A inferência é criada aplicando a regra se a então b. *Unordered Fuzzy Rule Induction Sorter* ou em português Classificador por indução de regras nebulosas não ordenadas (FURIA) é um método de classificação baseado em regras que aprende a separar cada classe de todas as outras classes, o que significa que as regras padrão não são usadas e a ordem das classes não importa, ou quando há classificador treinamento, não considere outras categorias.

- *Random Forest* (Floresta aleatória) - Eles são uma combinação de árvores de previsão, de forma que cada árvore depende do valor da amostra de um vetor aleatório independente e tem a mesma distribuição para todas as árvores da floresta. De um modo geral, a combinação de árvores está completa. Trabalha com a construção de cada árvore a partir das amostras do conjunto de treinamento, partindo da mesma construção, votando na escolha de cada árvore, de forma que a categoria final seja selecionada com base no maior número de votos(JHONATTAN; GHELLERE, 2015).

- *Support Vector Machines* (SVM, em português Máquina de vetores de suporte) - Tem como base a teoria estatística, e seu principal objetivo é encontrar a melhor função de classificação para distinguir dois tipos de exemplos. Isso é feito construindo

o hiperplano ótimo para separar as classes.

- *K-Means* - Este algoritmo particiona o conjunto de dados e um número específico de grupos (*clusters*) de maneira iterativa. Funciona da seguinte maneira (JHONATTAN; GHELLERE, 2015):

  1. Um certo número de *clusters* (grupos) é definido.
  2. Crie arbitrariamente (aleatoriamente) objetos k chamados centróides de *cluster*.
  3. Para cada elemento do conjunto de dados, o centróide mais próximo é atribuído de acordo com a distância.
  4. Quando cada elemento está associado a um dos k centróides, significa que n grupos serão formados.
  5. Os valores dos k centróides são recalculados como a média de todos os elementos (elementos do grupo) associados a eles.
  6. Repita esse processo até que o centróide não seja mais modificado.
  7. Finalmente, n grupos são formados.

2.4.2 **Algoritmo HOG (Histograma de Gradientes Orientados)**

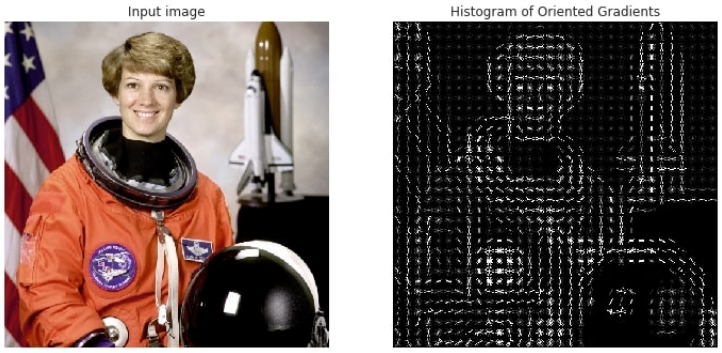


Figura 22 – Exemplo da aplicação do algoritmo HOG

A figura 22 é um exemplo da execução do algoritmo HOG em uma foto, fazendo a extração das partes interessadas.

O método Histograma de gradientes orientados é usado principalmente para detecção de faces e classificação de imagens, como você pode ver acima. Existem inúmeras aplicações nesta área, desde carros autônomos até vigilância mais inteligente e tecnologias de publicidade. Esse algoritmo passa por algumas etapas, sendo elas (ICHI.PRO, 2020):

1. Descritores de recursos
2. Pré-processamento

3. Calculando os Gradientes
4. Fazendo um histograma a partir desses gradientes
5. Normalização de bloco
6. Visualização de imagem

2.4.2.1 **Descritores de recursos**

O descritor de recursos representa apenas a representação da imagem, ele simplesmente extrai informações úteis da imagem, enquanto ignora informações desnecessárias. No caso do descritor de recursos HOG, também faz-se a conversão da imagem (largura x altura x canal) em um vetor de recursos de comprimento n selecionado pelo usuário. Embora essas imagens possam ser difíceis de visualizar, elas são adequadas para algoritmos de classificação de imagens, como SVM, para produzir bons resultados.

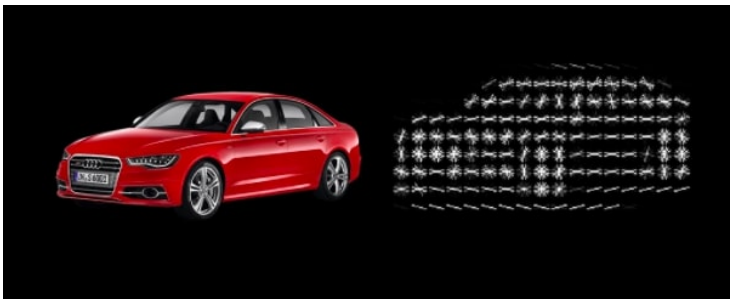


Figura 23 – Exemplo 2 da aplicação do algoritmo HOG

Para classificar as informações desnecessárias ele usa o histograma de gradientes que são usados como características de uma imagem. Os gradientes são muito importantes para verificar as bordas e cantos da imagem (por meio das áreas de mudança de intensidade), porque geralmente contêm mais informações do que áreas planas (ICHI.PRO, 2020).

2.4.2.2 **Pré-processamento**

Um erro importante que as pessoas frequentemente cometem ao detectar objetos HOG é que elas se esquecem de pré-processar a imagem para ter uma proporção fixa. Uma proporção comum (largura: altura) é 1: 2, então sua imagem pode ter 100 x 200, 500 x 1000 etc.

Para a imagem específica escolhida, é importante garantir que haja a determinação da parte necessária para que se ajuste à proporção da imagem corretamente e permita um acesso mais fácil a longo prazo.

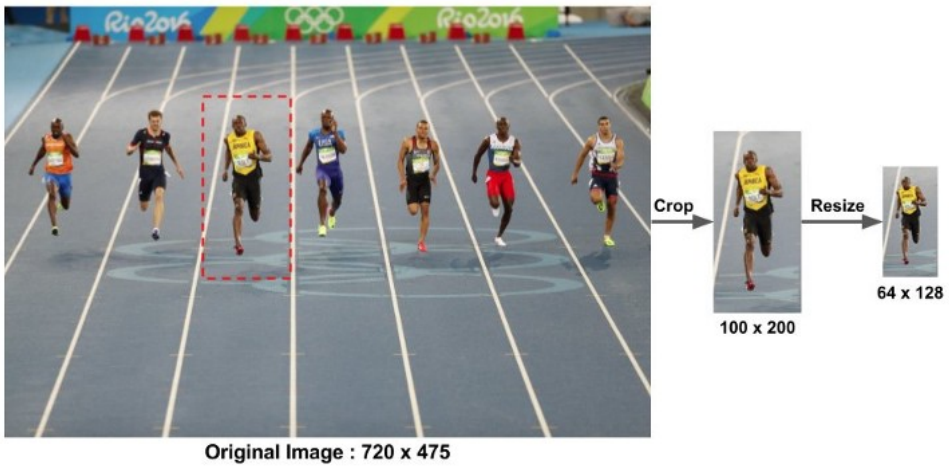


Figura 24 – Exemplo normalização da imagem

2.4.2.3 Calculando os Gradientes

Para fazer o descritor de recursos HOG discutido acima, precisamos calcular os respectivos gradientes horizontal e vertical para fornecer um histograma que possa ser usado por trás do algoritmo. Isso pode ser feito simplesmente filtrando a imagem por meio destes *kernels*:

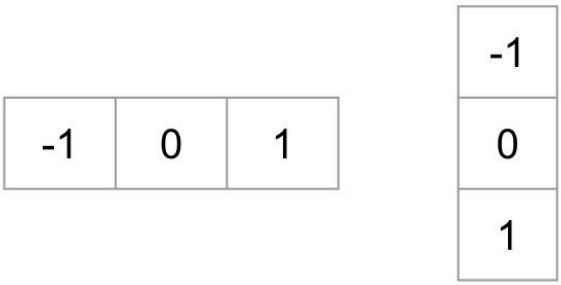


Figura 25 – *Kernels*

Kernels como esses são frequentemente usados para classificação de imagens, especialmente em redes neurais convolucionais, a fim de encontrar bordas e pontos importantes em uma imagem específica.

Portanto, você pode simplesmente encontrar o tamanho e a direção do gradiente usando a seguinte fórmula (observe que esta é apenas uma conversão de coordenadas cartesianas para coordenadas polares em certo sentido) (ICHI.PRO, 2020):

$$g = \sqrt{g_x^2 + g_y^2}$$
$$\theta = \arctan \frac{g_y}{g_x}$$

A principal lição que deve-se aprender com os gradientes é que sempre que a intensidade muda repentinamente, o tamanho do gradiente aumenta. A imagem abaixo destaca um exemplo de gradientes ideais, o que pode ser feito quando eles são emitidos ao redor das bordas da imagem através desses gradientes. As informações desnecessárias são excluídas como pano de fundo e apenas as partes necessárias são mantidas.

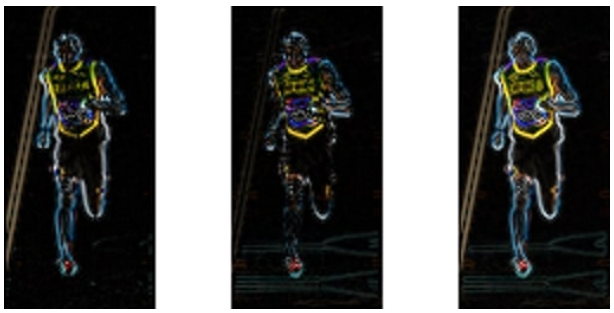


Figura 26 – Exemplo de magnitude de gradiente

Entende-se que o gradiente tem uma magnitude e direção, onde a magnitude é calculada com base na magnitude máxima do gradiente dos três canais de cores, e o ângulo é calculado com base no ângulo correspondente ao gradiente máximo dos três canais de avaliação. Eles podem gerar imagens como as acima, onde podem detectar informações importantes e ignorar partes desnecessárias.

2.4.2.4 **Fazendo um histograma a partir desses gradientes**

Para avançar para a próxima etapa do algoritmo HOG, deve certificar-se de que a imagem seja dividida em células para que o histograma de gradientes possa ser calculado para cada célula. Por exemplo, se tiver uma imagem 64x128, necessita fazer a divisão da imagem e transformá-la em células 8x8.

O descritor de recursos permitirá uma representação concisa de pontos específicos na imagem; tomando nosso exemplo acima como exemplo, 128 números (8x8x2, os 2 últimos dos quais são valores de magnitude e direção de gradiente) podem ser usados para explicar simplesmente células 8x8. Ao converter ainda mais esses números para calcular o histograma, habilitamos um bloco de imagem mais resistente a ruídos e mais compacto.

Para o histograma, certifique-se de dividi-lo em nove caixas separadas, cada uma correspondendo a um ângulo de 0 a 160 em incrementos de 20. A seguir está um exemplo de uma imagem com a respectiva amplitude e direção do gradiente (observe que a seta maior depende do tamanho) (ICHI.PRO, 2020).

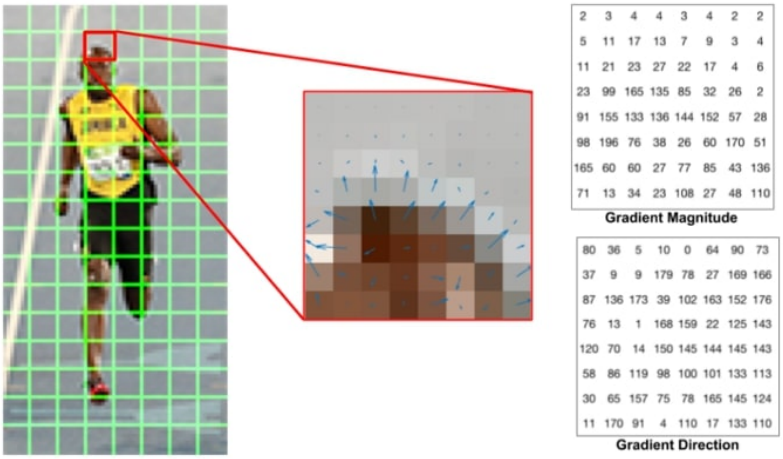


Figura 27 – Exemplo de histograma a partir de gradiente

Um compartimento é selecionado de acordo com a direção escolhida e o valor colocado no compartimento depende do tamanho. Observe que se um *pixel* estiver localizado no meio de duas caixas, ele será dividido de acordo com a distância de cada caixa. Depois de realizar esse processo, um histograma pode ser formado e caixas mais pesadas podem ser vistas facilmente.

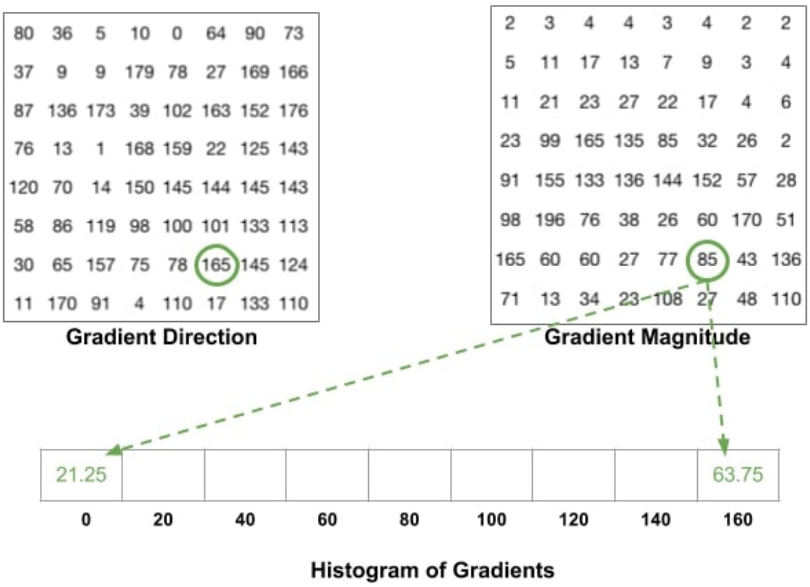


Figura 28 – Exemplo de histograma

2.4.2.5 Normalização de bloco

As mudanças de iluminação são outro fator importante, que afeta como esses gradientes são calculados. Por exemplo, se a imagem for 1/2 mais escura do que o brilho atual, a amplitude do gradiente e a amplitude do histograma subsequente serão reduzidas pela metade. Portanto, esperamos que nossos descritores não sejam afetados por mudanças na

iluminação para permanecerem justos e eficazes.

O processo de normalização é simplesmente usar o tamanho do vetor para calcular o comprimento do vetor e, em seguida, simplesmente dividir todos os elementos do vetor pelo comprimento. Por exemplo, se você tiver um vetor de [1,2,3], usando princípios matemáticos básicos, o comprimento do vetor será a raiz quadrada de 14. Dividindo o vetor por este comprimento, você pode obter um novo vetor normalizado [0,27, 0,53, 0,80].

Este processo de normalização pode ser realizado de acordo com sua preferência (se você deseja rodar em blocos de 8x8 ou em blocos maiores de 16x16). Deve ser lembrado de converter esses blocos em vetores de elemento primeiro para que a normalização delineada acima possa ser realizada(ICHI.PRO, 2020).

2.4.2.6 **Visualização de imagem**

Em muitos casos, o descritor HOG é geralmente exibido com a imagem à direita para obter uma representação precisa da figura humana. Esta visualização é muito útil para entender a posição das mudanças de gradiente e entender a posição do objeto na imagem.

Como pode-se observar na figura a seguir, com a extração as informações, com o gradiente aparece um rosto (ICHI.PRO, 2020).

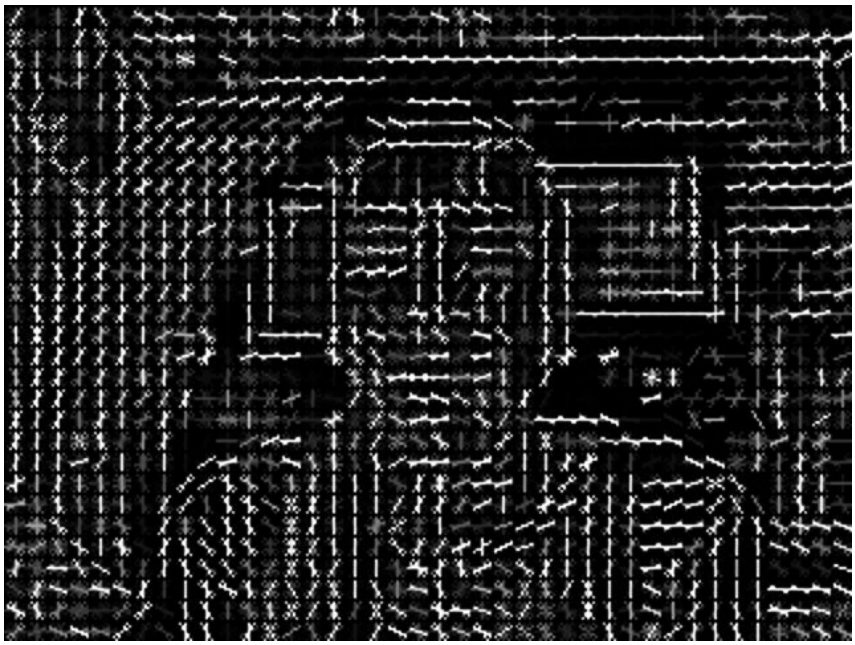


Figura 29 – Exemplo de visualização final de HOG

2.4.3 **Algoritmo CNN**

A Rede Neural Convolucional (CNN) é um algoritmo de aprendizado profundo que pode capturar imagens de entrada e atribuir importância (como pesos e tendências)

a vários aspectos e objetos da imagem, e pode distingui-los uns dos outros (ALIGER, 2019).

Em comparação com outros algoritmos de classificação, o pré-processamento necessário no CNN é muito menor. Embora no método original os filtros sejam feitos à mão, CNN pode aprender esses filtros ou recursos por conta própria.

A arquitetura do CNN é semelhante ao padrão de conexão dos neurônios do cérebro humano, que é inspirado na organização do córtex visual. Desta maneira, pode-se dizer que uma rede neural é um método de cálculo que aprende com a experiência (ALIGER, 2019). Cada rede neural possui nós ou neurônios e é dividida em camadas, camadas de entrada (entrada), saída (saída) e camadas ocultas (ocultas). Neurônios ocultos podem formar mais de uma camada (E-LIBRARY, 2018).

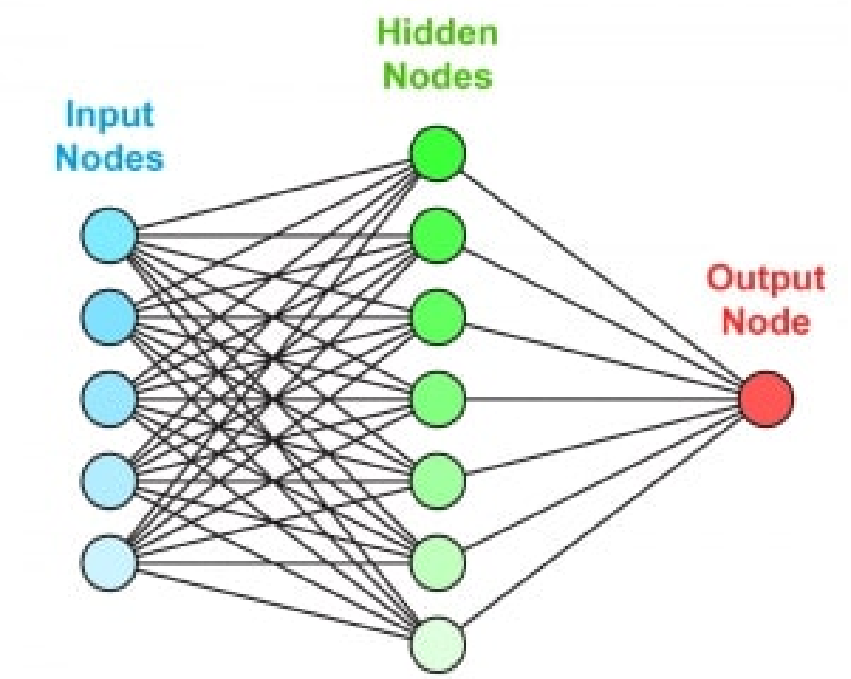


Figura 30 – Exemplo Rede Neural

O neurônio representa uma função de ativação, e a rede recebe informações no nó de entrada, que são processadas na camada oculta e produzem uma saída. Todos os nós de uma rede neural não treinada têm o mesmo peso. Quando ela aprende, algumas conexões passam a ter pesos maiores do que outras. Conexões com pesos maiores significam que eles estão no processo de aprendizagem e execução de tarefas da rede a influência é maior.

O exemplo abaixo é um modelo matemático de neurônios artificiais. Onde os "X" são as entradas, "W"os pesos, $\sum$ é a função de soma e o bloco f(a) é a função de ativação com o limitador t (E-LIBRARY, 2018).

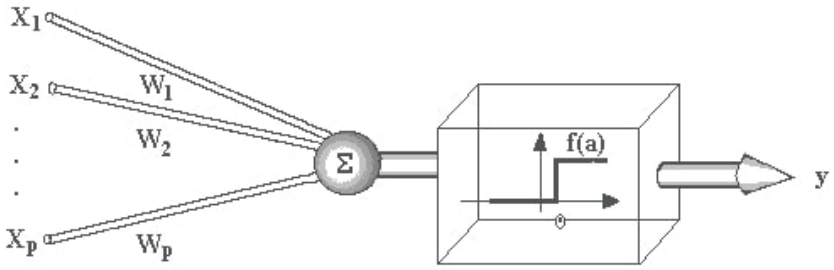


Figura 31 – Exemplo Neurônio Artificial

Uma das regras de aprendizado mais comuns é a regra delta, que é usada em redes neurais de retro-propagação. É uma regra de aprendizado supervisionado, ou seja, o agente externo determina a saída necessária, e a rede neural deve ajustar o peso da conexão para produzir a saída necessária. A regra delta do *Perceptron* atualiza os pesos com base em uma função de etapa, e a regra delta do Adaline atualiza os pesos com base na ativação linear.

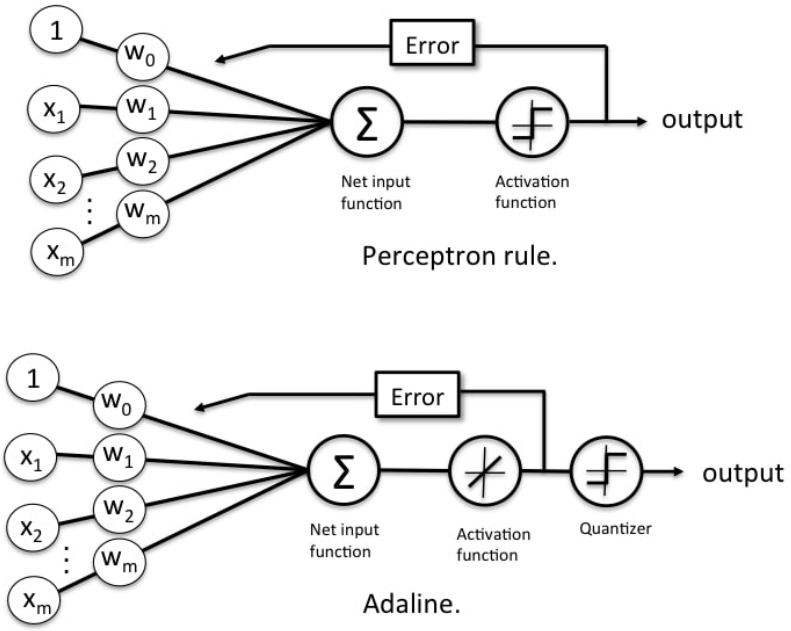


Figura 32 – Exemplo *Perceptron* e Adaline

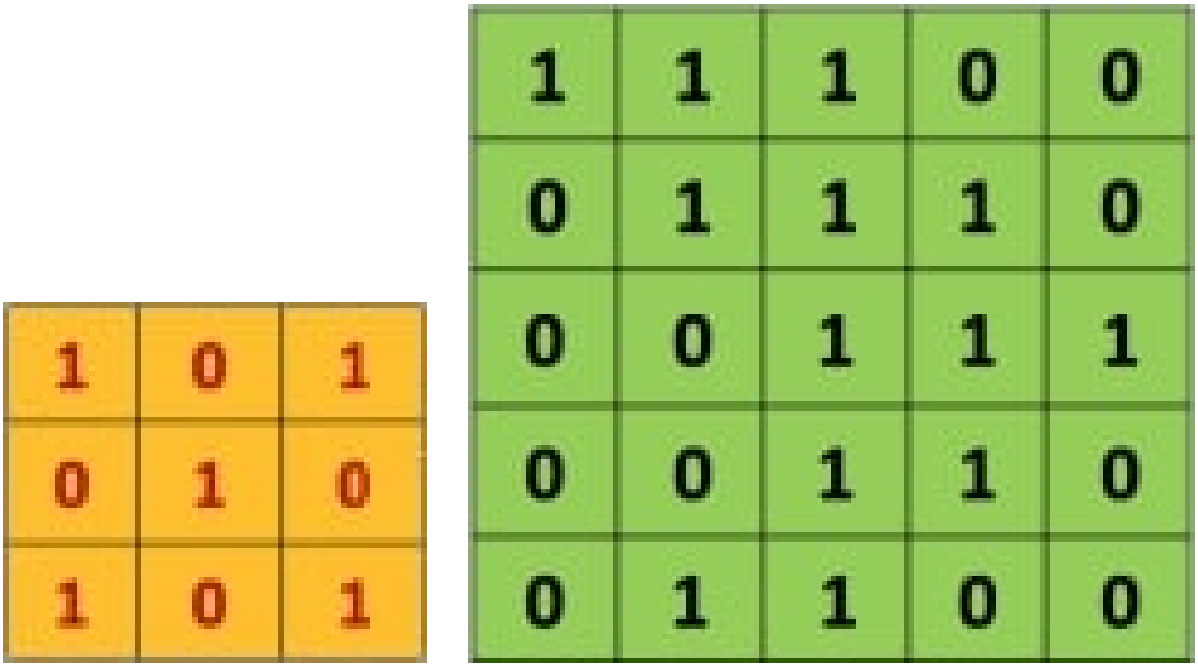
Um padrão é inicialmente exibido para a rede, o que produz uma saída que mede a distância entre o peso ideal e o peso real e ajusta o peso da conexão para reduzir o erro quadrático médio ao mínimo global. O hiperparabolóide representa a margem de erro do peso, que diminui antes de atingir o valor de erro mínimo aceitável. A regra delta cria um vetor gradiente em direção ao vetor de peso ideal (E-LIBRARY, 2018). Na aprendizagem por reforço, a saída na rede é avaliada e, se a resposta for sim, os pesos são ajustados para produzir essa saída. Existem casos em que uma grande quantidade de dados não pode ser rotulada, então um método de aprendizagem não supervisionado é implementado, no

qual os dados são divididos em grandes grupos chamados *clusters*.

As redes neurais tradicionais requerem muitas entradas e parâmetros para analisar pequenos padrões e muito poder de processamento do computador. Além disso, ela pode se tornar especialista nos dados em que é treinada e pode perder desempenho ao visualizar novos dados.

Observando sobre as operações de convolação, temos que a imagem é tratada como uma matriz tridimensional: a altura e a largura determinam o tamanho da imagem, e a profundidade representa o canal de cores RGB. As três cores primárias são: vermelho (R), verde (G) e azul (B), como já visto (E-LIBRARY, 2018).

A rede neural realiza operações de convolução de matriz. O *array* menor é usado como filtro, também chamado de *kernel*. O filtro lê todos os *pixels* e gera um *array* com uma dimensão menor do que o *array* de entrada, como mostra o exemplo abaixo.



(a) (b)

Figura 33 – Exemplo de uma operação de convolução

Na CNN existem filtros para tratar imagens, elas usadas para desfocagem, relevos, cores, manipular a imagem, detectar bordas e nitidez (E-LIBRARY, 2018).

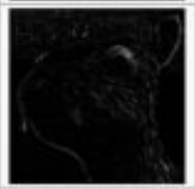
Operation	Filter	Convolved Image
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
Gaussian blur (approximation)	$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	

Figura 34 – Exemplo de filtros

Não-linearidade

Após a convolução, a função de ativação é usada para operações não lineares. Isso é necessário porque o mundo real é não linear e as redes neurais convolucionais devem reconhecer padrões não lineares. A função mais eficaz é ReLU.

f(x)=max(0,x)

Esta função substitui os valores de *pixels* negativos por zeros para evitar a saturação durante o treinamento.

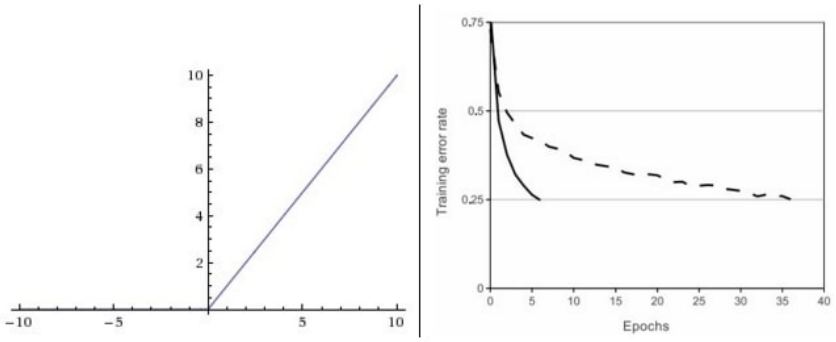


Figura 35 – Exemplo ReLU histograma

Imagens antes, esquerda e depois de ReLU, direita(E-LIBRARY, 2018).

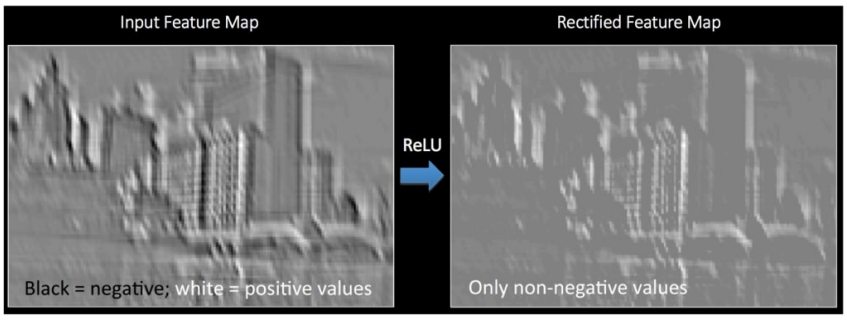


Figura 36 – Exemplo ReLU

Pooling

Ele reduz o tamanho da matriz por meio da amostragem e retém os recursos mais importantes. Os principais tipos são: máximo, soma e média. A seguir está um *pool* máximo.

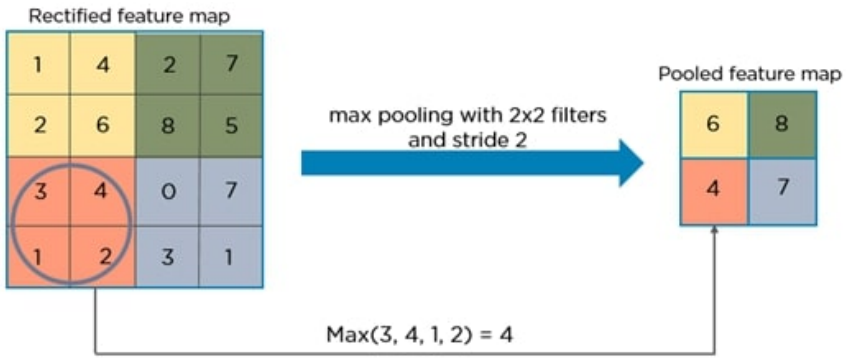


Figura 37 – Exemplo *pooling*

A aparência da imagem após o processo de agrupamento.

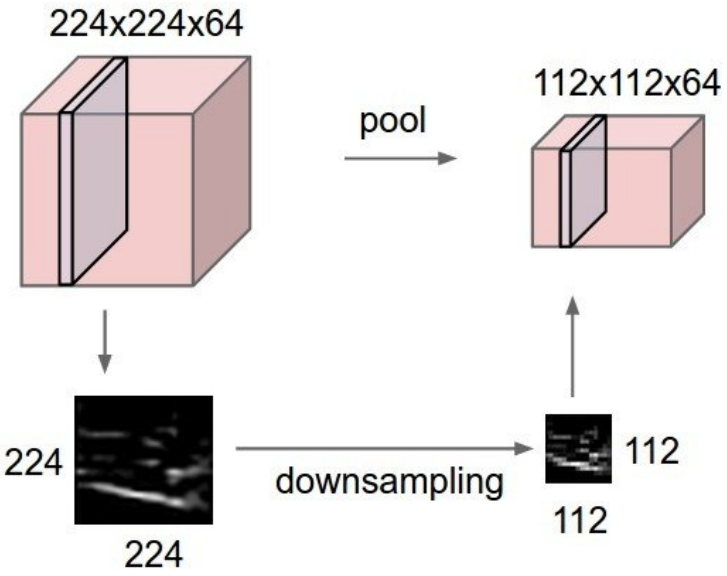


Figura 38 – Exemplo resultado do pooling

A função desse processo é simplificar e reduzir as informações dos neurônios da rede e reduzir o número de parâmetros.

Camada totalmente conectada

A última camada está totalmente conectada e é onde a rede neural *perceptron* multicamadas está localizada. Recebe itens processados para avaliação. Ele tem esse nome porque cada neurônio da camada superior está conectado a todos os neurônios da camada inferior.

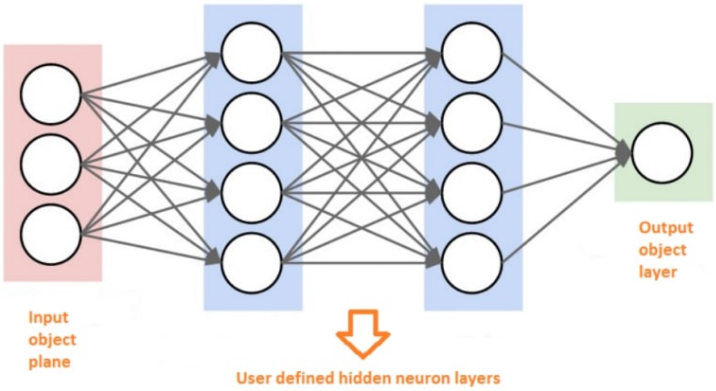


Figura 39 – Exemplo CNN

Arquitetura

Na arquitetura de rede neural convolucional, as posições das camadas são organizadas na seguinte ordem: entrada, convolução, não linearidade,*pooling* e rede neural totalmente conectada (E-LIBRARY, 2018).

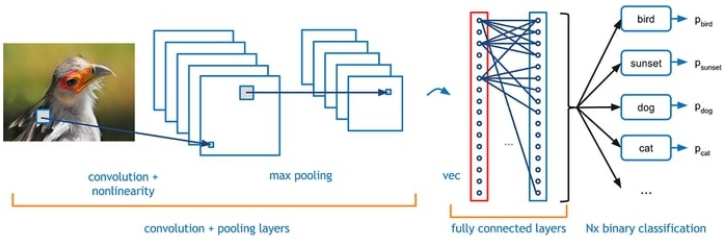


Figura 40 – Exemplo arquitetura CNN

Algumas redes neurais convolucionais têm 2 ou mais estágios de convolução, ReLU e *pooling*.

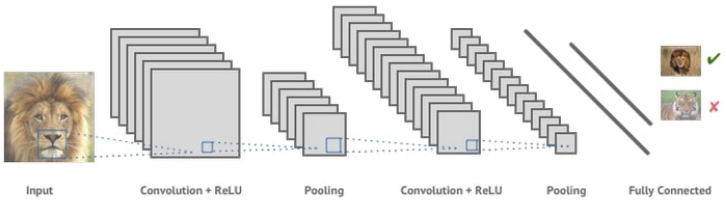


Figura 41 – Exemplo resultado CNN

Algumas aplicações

As redes neurais convolucionais podem reconhecer sinais de tráfego e serão usadas em veículos não tripulados. E também podem ser usadas para reconhecimento de voz, e os dados de áudio são representados visualmente em um espectrograma. Entre outras aplicações (E-LIBRARY, 2018).

2.4.4 Algoritmo KNN

No reconhecimento de padrões, K-Nearest Neighbors ou K-vizinhos mais próximo (KNN) é um método não paramétrico para classificação e regressão. Em ambos os casos, a entrada consiste nos k exemplos de treinamento mais próximos no espaço de recursos. A saída depende se KNN é usado para classificação ou regressão (MASTER, 2019):

Na regressão k-NN, a saída é o valor do atributo do objeto. Este valor é o valor médio dos k vizinhos mais próximos. Já na classificação KNN, a saída é uma associação de classes, ou seja, quando usamos o algoritmo KNN, um objeto é classificado de acordo com seus vizinhos. Geralmente, a classe atribuída é a classe mais comum entre seus k vizinhos mais próximos. Quando o usuário define k = 1, o algoritmo entenderá que o objeto pertence à mesma classe que o único vizinho mais próximo. Quando K é maior que 1, o valor de seus k vizinhos mais próximos é calculado, e então é possível classificar os eventos recém-surgidos. Observe um exemplo abaixo.

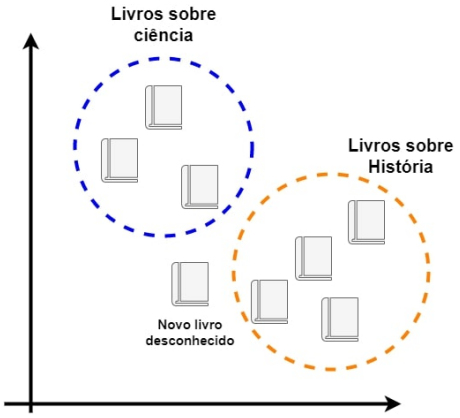


Figura 42 – Exemplo KNN

Na imagem acima pode-se ver claramente que existem dois grupos claramente definidos (livros de história e livros de ciências). No entanto, uma nova amostra foi inserida em nosso gráfico, mas é desconhecida. O algoritmo KNN pode classificar facilmente este novo livro com base apenas em seus vizinhos. Opta-se por escolher avaliar os 5 vizinhos mais próximos (MASTER, 2019):

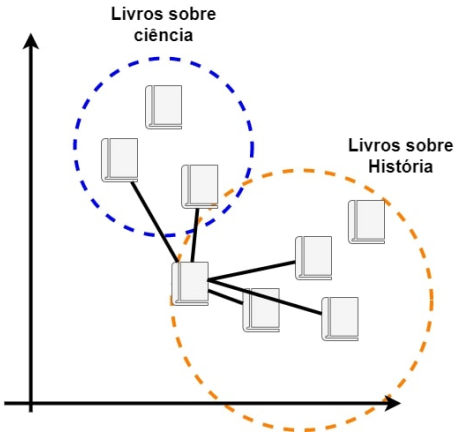


Figura 43 – Exemplo busca por vizinho mais próximo

Ao analisar os 5 vizinhos mais próximos, descobrimos que 3 deles são da história e apenas 2 são da ciência. Portanto, nosso novo exemplar é classificado como livro de história.

Ao falar sobre KNN, é estar se referindo a um tipo de aprendizagem baseada em instâncias ou aprendizagem preguiçosa, em que as funções são apenas aproximações locais e todos os cálculos são postergados para classificação (MASTER, 2019).

O algoritmo KNN é um dos algoritmos mais simples entre todos os algoritmos de aprendizado de máquina. Para classificação e regressão, uma técnica útil pode ser usada para ponderar as contribuições dos vizinhos de modo que vizinhos mais próximos contribuam mais para a média do que vizinhos distantes. Por exemplo, um esquema de ponderação comum é dar a cada vizinho um peso de 1/d, onde d é a distância ao

vizinho. Os vizinhos são obtidos de um conjunto de objetos com categorias conhecidas (para classificação KNN) ou valores de atributos de objetos (para regressão KNN). Isso pode ser considerado o conjunto de treinamento do algoritmo, embora nenhuma etapa de treinamento explícita seja necessária.

3 METODOLOGIA

Após visto a teoria e algoritmos na seção anterior, nessa é observado como é feita a implementação. Os algoritmos escolhidos para a realização do processo são:

- Algoritmo CNN em conjunto com HOG para a detecção de faces.
- Algoritmo KNN para reconhecimento facial e classificação.

A escolha desses dois algoritmos para a detecção, reconhecimento e classificação de faces é pelo fato de sua confiabilidade nos resultados, na seção 4, resultados, é visto um terceiro algoritmo, o HOG, apenas para mostar a comparação com o CNN e observar-se que o algoritmo escolhido tem resultados melhores que o outro comparado.

A figura 44 representa o fluxograma dos passos a realizar para a obtenção dos resultados e logo após a explicação de cada um.

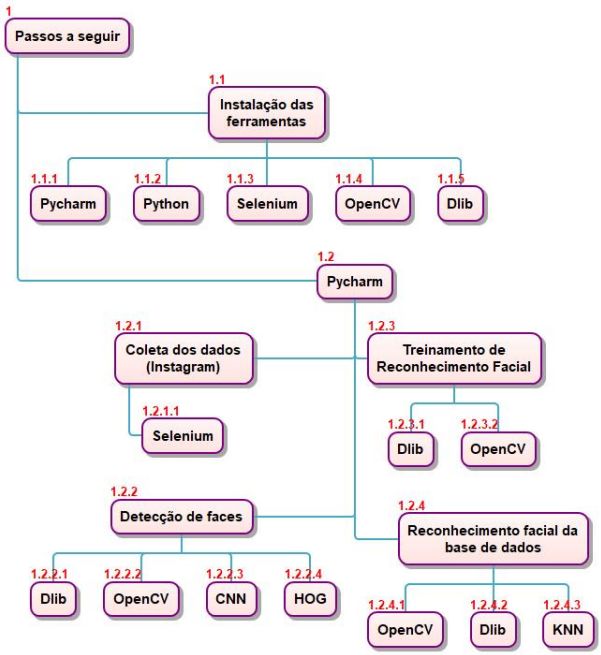


Figura 44 – Fluxograma dos passos necessários.

1.1 - Instalação das ferramentas que são utilizadas como *python*, *OpenCV*, *Dlib* e *Selenium*.

1.2 - Utilização da ferramenta Pycharm para o desenvolvimento de todo algoritmo.

1.2.1 - Coleta dos dados (*instagram* por meio da biblioteca *Selenium*).

1.2.2 - Detecção de faces usando o algoritmo HOG e comparando com o algoritmo CNN.

1.2.3 - Treinamento de Reconhecimento facial usando a biblioteca OpenCV e Dlib.

1.2.4 - Reconhecimento facial com a base de dados baixados do *instagram*.

3.1 *Hardware* Utilizado

O computador usado para a realização da implementação e teste é:

- *Notebook* da *ACER* ano 2017
- Processador: *Core i5 7th Gen*
- Memoria: 16GB
- *SSD*: 256GB
- Placa de vídeo: *NVIDEA GFORCE 940MX*
- Sistema Operacional: *Windows 10*

3.2 Linguagem escolhida: *Python*

A linguagem escolhida é o *Python* por sua baixa complexidade de implementação e por ter diversas bibliotecas que auxiliaram na implementação dos algoritmos.

Sobre o *Python* pode-se dizer que foi criada por Guido van Rossum no ano de 1991, ela é uma linguagem de programação de alto nível, imperativa, orientada a objeto, funcional, de tipagem dinâmica e forte e interpretada de *script*, além disso, *Python* é uma linguagem de programação de fácil acesso, e tem se tornado muito popular em áreas emergentes da indústria de tecnologia.

A filosofia de design da linguagem é enfatizar a importância do esforço do programador sobre o esforço computacional. Prioriza a legibilidade do código em relação à velocidade ou expressividade. Ele combina a sintaxe concisa e clara e as funções poderosas de sua biblioteca padrão e módulos e estruturas de terceiros.

Devido às suas características, ele é usado principalmente para processamento de texto, dados científicos e criação CGI(*Common Gateway Interface*, Interface Comum de Ligação) de páginas da *web* dinâmicas.

Como quase todas as boas ideias, essa ideia nasceu da necessidade de economizar tempo de desenvolvimento e melhorar a eficiência dos projetos desenvolvidos no instituto de pesquisa onde Guido é pesquisador.

Para completar esta melhoria de forma mais rápida e eficiente, Guido desenvolveu uma linguagem muito simples e flexível: *Python*, como dito anteriormente.

Uma vez que essa linguagem passou a permitir a criação de *scripts* muito simples para sistemas extremamente poderosos, profissionais de diversas áreas passaram a utilizá-la cada vez mais.

Hoje, além de desenvolvedores de *software*, também conta-se com biólogos, contadores, físicos e outros profissionais através dela para utilizar suas competências.

O *Python* tem bibliotecas que ajudam no desenvolvimento como na parte de Processamento de imagens, Inteligencia Artificial, Redes Neurais, entre outros. Ela vem com pacotes prontos com o intuito de facilitar o desenvolvimento para o programador. Diante disso, é usado três dessas bibliotecas para a realização desse projeto sendo eles:

- ***Selenium*** que auxilia na coleta de dados.
- ***Dlib*** que auxilia na detecção de faces e o reconhecimento facial.
- ***OpenCV*** que auxilia na detecção de faces.

Veja mais sobre cada nas próximas subseções.

3.2.1 *Selenium*

Selenium é um conjunto de ferramentas de *software* livre de plataforma cruzada para testar aplicativos da *Web* de maneira automatizada por meio de um navegador. Ele executa testes funcionais de aplicativos da *web* e testes de compatibilidade de navegador de plataforma cruzada. Esta bibloteca oferece suporte a várias linguagens de programação, como C#, *Java* e *Python*, bem como vários navegadores da *web*, como *Chrome* e *Firefox*.

O ecossistema *Selenium* é muito completo, incluindo:

- ***Selenium IDE*** - *Selenium* IDE é um ambiente de desenvolvimento integrado para *scripts* de teste automatizados, que permite aos usuários criar testes muito rapidamente. Esta ferramenta permite que o desenvolvedor grave o código, atue como um gravador e registre ações do usuário. O responsável parametriza e executa a cifra gerada conforme necessário. O *Selenium* IDE inclui o *Selenium Core*, permitindo que quem o utiliza grave e reproduza facilmente os testes no ambiente real onde eles serão executados. Esta IDE é voltado para testes e *feedback* rápidos.
- ***Selenium WebDriver*** - *Selenium WebDriver* usa seu próprio *driver* de navegador para automação. Atualmente é a forma de interação mais moderna, pois cada navegador possui seu próprio *driver*, permitindo a interação entre os *scripts* de teste e seus respectivos navegadores. Recomenda-se usar esta versão para testes mais complexos e usuários familiarizados com a ferramenta. Normalmente, você usa *Selenium* IDE para testes básicos, exporta o código e edita para realizar testes mais complexos.

- ***Selenium Grid*** - No *Selenium Grid*, o programador lança seu código sobre diferentes navegadores. O *Grid* é voltado para clusterização, permitindo ao desenvolvedor realizar os testes em máquinas diferentes de forma remota. Segundo o *site* oficial do *Selenium*, o *Selenium Grid* leva o *WebDriver* a outro nível, executando testes em muitas máquinas ao mesmo tempo, reduzindo o tempo necessário para testar em vários navegadores e sistemas operacionais. Utilizando este método você ganha na escala, onde executamos o mesmo teste em centenas de configurações diferentes.

O ***Selenium WebDriver*** é o escolhido para a contribuição da implementação deste trabalho.

3.2.2 *Dlib*

Dlib é um *kit* de ferramentas C++ moderno que contém algoritmos de aprendizado de máquina e ferramentas para a criação de *software* C++ complexo para resolver problemas práticos. É amplamente utilizado na indústria e na área acadêmica, incluindo robótica, dispositivos incorporados, telefones celulares e grandes ambientes de computação de alto desempenho. A licença de código aberto do *Dlib* permite que você o use gratuitamente em qualquer aplicativo. Algumas de suas principais características são:

- **Documentação**
  - Ao contrário de muitos projetos de código aberto, este projeto fornece documentação completa e precisa para cada classe e função. Existem também alguns modos de depuração para verificar os pré-requisitos de gravação da função. Quando habilitado, ele detectará a maioria dos erros causados por chamadas incorretas de funções ou uso incorreto de objetos.

- **Algoritmos de aprendizagem de máquina**
  - Aprendizagem Profunda.
  - Ferramentas SVM estruturais para rotulagem de sequência.
  - Ferramentas SVM estruturais para resolver problemas de atribuição.
  - Ferramentas SVM estruturais para detecção de objetos em imagens, bem como ferramentas de aprendizado profundo mais poderosas (porém mais lentas) para detecção de objetos.
  - Ferramentas SVM estruturais para rotular nós em gráficos.
  - Uma implementação SVM-Rank em grande escala.
  - Um algoritmo de regressão RLS de *kernel on-line*.
  - Um algoritmo de classificação SVM *on-line*.
  - Aprendizagem de métrica semi-definida.

- **Processamento de imagem**
  - Rotinas para ler e escrever formatos de imagem comuns.

- Conversão automática do espaço de cores entre vários tipos de *pixel*.
- Operações comuns de imagem, como localização de bordas e operações morfológicas.
- Implementações dos algoritmos de extração de recursos SURF, HOG e FHOG.
- Reconhecimento facial de alta qualidade.

- ***Threading***
  - A biblioteca fornece uma API de *thread* simples e portátil.
  - Um objeto temporizador capaz de gerar eventos regularmente espaçados no tempo.
  - Funções encadeadas.
  - Paralelo para *loops*.

- **Algoritmos Numéricos**
  - Um objeto de matriz rápida implementado usando a técnica de *templates* de expressão e capaz de usar as bibliotecas BLAS e LAPACK quando disponíveis.
  - Numerosas álgebra linear e operações matemáticas são definidas para o objeto de matriz, como decomposição de valor singular, transposição, funções trigonométricas, etc.
  - Finalidade geral é gerar algoritmos de optimização sem restrições não linear, utilizando o gradiente conjugado, BFGS(*Broyden-Fletcher-Goldfarb-Shanno*) , e L-BFGS(*Limited-memory Broyden-Fletcher-Goldfarb-Shanno*) técnicas.

3.2.3 *OpenCV*

OpenCV (*Open Source Computer Vision*) é uma biblioteca de programação de código aberto originalmente desenvolvida pela *Intel* para tornar mais fácil para os desenvolvedores e entusiastas usar a visão computacional. Atualmente possui mais de 500 funções e pode usar várias linguagens de programação (C++, *Python*, *Ruby*, *Java*) para diferentes tipos de análise de imagem e vídeo, como detecção, rastreamento e reconhecimento facial, edição de fotos e vídeo, Detecção e análise de texto, etc.

O projeto *OpenCV* foi lançado pela primeira vez em 1999 como uma proposta da *Intel Research* para melhorar os aplicativos de uso intensivo do processador e fazer parte de uma série de projetos, incluindo rastreamento de raios e monitores 3D. Os principais contribuintes para este projeto são da *Intel* Rússia, bem como da equipe de desempenho de bibliotecas da *Intel*. No início do projeto, o objetivo era definido como:

- Não apenas fornece código aberto, mas também otimiza para tarefas básicas de visão, para que o código seja fácil de ler e transmitir, avançando assim na pesquisa de visão computacional.
- Promover aplicativos baseados em visão computacional, fornecendo código otimizado portátil gratuitamente, e obtendo licenças que não requeiram aplicativos de código aberto.

3.2.4 ***Pycharm***

A plataforma de desenvolvimento *PyCharm* é um ambiente de desenvolvimento integrado (IDE) para programação de computadores, especificamente para a linguagem *Python*. Foi desenvolvido pela empresa checa *JetBrains* (anteriormente *IntelliJ*). Ele fornece análise de código, depurador gráfico, testador de unidade integrado e integração de sistema de controle de versão (VCS), e suporta o uso de *Django* para desenvolvimento *web* e Anaconda para ciência de dados.

PyCharm é multiplataforma, com versões *Windows*, *MacOS* e *Linux*. A edição da comunidade é lançada sob a licença *Apache*, e também há uma edição profissional com recursos adicionais lançada sob uma licença proprietária.

4 RESULTADOS

4.1 Coleta de dados com aplicação no *Instagram*

A coleta de dados é feita com *Python* e usando a biblioteca *Selenium WebDriver*. Houve a escolha de duas pessoas como teste para a obtenção do resultado, ou seja, se existe ou não fotos sendo usadas sem ser pelo dono das fotografias e sem ser de conhecimento do mesmo, sendo um anônimo, Jéssica Romero, um famoso, Neymar Junior. Para a obtenção dos dados, devido a politica do *Instagram*, é conseguido através de *hashtags*. No caso do Neymar para a coletar os dados poderia ser usado as *hashtags* em relação ao trabalho dele, futebol, a ex-namorada, Bruna Marquezine ou pelo nome direto do jogador que foi o utilizado. Já no caso da anônima, da mesma forma que com o jogador, a coleta de sua base de dados pode ser pelos gostos da mesma, lugares que ela frequenta, festas e/ou pelo nome, nesse caso foi usado um lugar onde ela frequenta bastante e tem algumas publicações, praia da graciosa que fica localizada na cidade de Palmas - Tocantins, lugar da qual ela reside. Abaixo encontra-se os passos que são realizados e a apresentação do código desenvolvido na linguagem escolhida.

1. O primeiro passo é a criação de um *login* e senha para entrar na plataforma do *Instagram*.

2. Após a criação, de maneira automática o *Selenium* entra na plataforma e vai até a barra de pesquisa e busca #neymarjr e #praiadagraciosa e faz o acesso ao primeiro que aparecer na busca, abrindo o conteúdo da *hashtag*.

3. Com a conclusão do passo dois, vem a três em que o *WebDriver* faz uma varredura pela página e retorna a quantidade de fotos obtida pelo resultado e faz o *download* das de 30 a 45 fotos iniciais e armazena na pasta direcionada.

4. Com a coleta dos dados concluída vai para o próximo passo que é a detecção de faces. Entretanto, antes de usar a base de dados coletado, necessita fazer o treinamento do algoritmo para a obtenção dos resultados.

O trecho do código para acessar a *hashtag* em *python* é

```
1 keyword = "#praiadagraciosa"
2 searchbox.send_keys(keyword)
```

O trecho do *script* acima mostra que a busca está sendo realizada pelo interesse da Jéssica, por esse motivo o *keyword* é igual a #praiadagraciosa, quando se passa para a extração da base de dados do Neymar o *keyword* ganha valor igual a #neymarjr, todo o restante do código permanece igual. Dando prosseguimento, a outra parte para fazer a varredura na página.

```
n_scrolls = 3
for j in range(0, n_scrolls):
    driver.execute_script("window.scrollTo(0, document.body.scrollHeight
        );")
    time.sleep(5)

    anchors = driver.find_elements_by_tag_name('a')
anchors = [a.get_attribute('href') for a in anchors]
```

Após feito a varredura, o trecho que faz a raspagem e salva as imagens em uma pasta é

```
os.mkdir(path)

path

counter = 0
for image in images:
    save_as = os.path.join(path, keyword[1:] + str(counter) + '.jpg')
    wget.download(image, save_as)
    counter += 1
```

Com os dados de interesse armazenados vamos para o passo seguinte.

4.2 Resultados do Crawling

Com o fim da execução do código acima, obtém-se os resultados da base de dados da qual se fará a busca pelo reconhecimento facial e a classificação em cada foto armazenada. A figura 45 representa os dados coletados com a #praiadagraciosa referente a Jéssica e a figura 46 representa os dados coletados com a #neymarjr referente ao jogador, Neymar Junior. Pode-se observar abaixo a visualização de ambas:

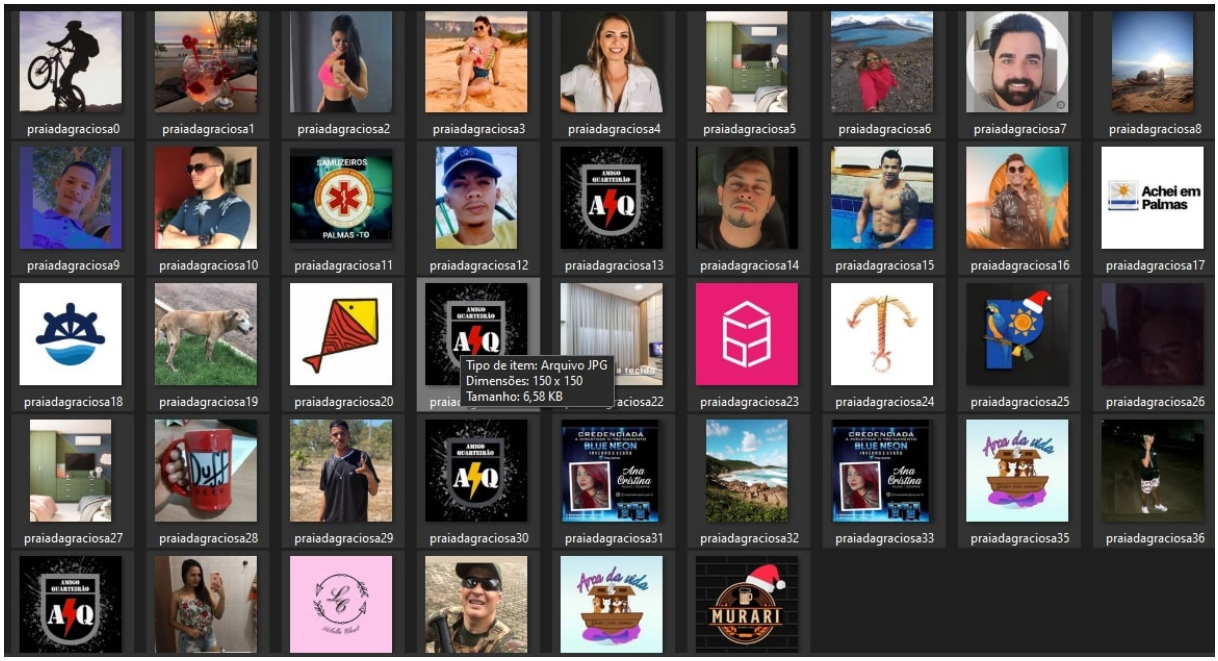


Figura 45 – A base de dados obtida do *Instagram* com a #praiadagraciosa

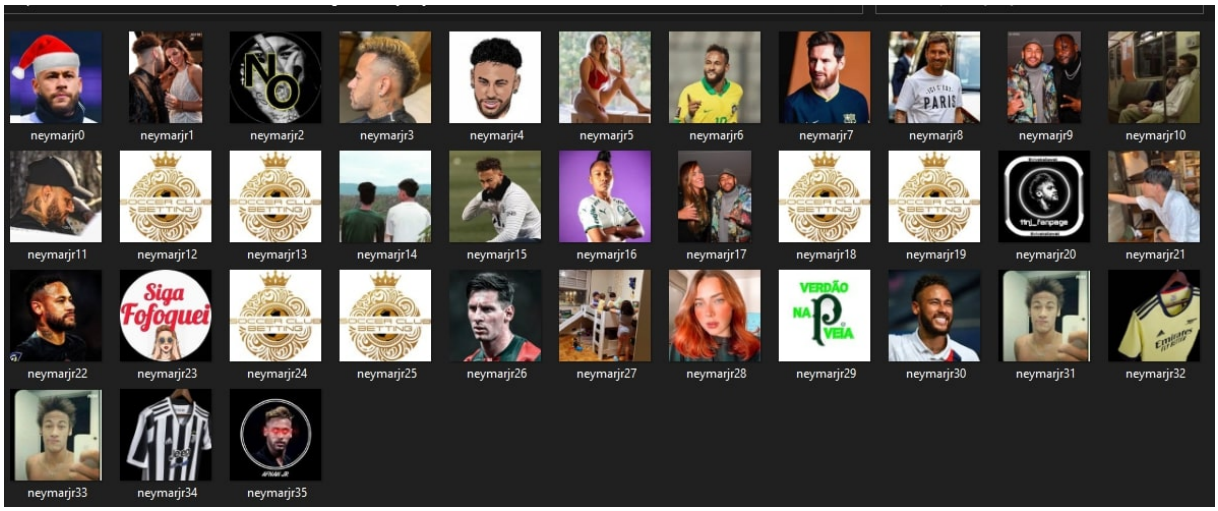


Figura 46 – A base de dados obtida do *Instagram* com a #neymarjr

4.3 Detecção de faces

Os 7 processos realizados:

1. Com a utilização da biblioteca *OpenCV* e *Dlib*, utilizando os algoritmos HOG e CNN, respectivamente, obtém o resultado da detecção de faces.

2. Detecção de faces: O primeiro passo para reconhecer rostos humanos é que deve-se inicialmente detectá-los. Para isso, é utilizado as funções disponíveis na biblioteca *DLib*.

3. Detecção de pontos faciais: Antes do reconhecimento facial, se é necessário a utili-

zação de uma tecnologia que nos permita detectar a região de interesse na imagem, pois é a detecção de rosto. Um exemplo de região de interesse pode ser a detecção de pontos faciais.

4. Utilização do CNN: faz-se o treinamento de um modelo de CNN pré-treinado para treinar as fotos a fim de se estar aptos para realizar o reconhecimento facial.

5. O próximo passo é o reconhecimento facial feito com a biblioteca *Dlib* e utilizando o algoritmo KNN. Desta maneira, é feito o treinamento e depois o teste.

6. Por fim, para saber se o algoritmo está funcionando de forma correta, obtém a classificação, ou seja, detectar dentre algumas fotos o rosto treinado e o seu grau de acurácia.

7. Obtenção dos resultados.

Para os passos 2 e 3, encontra-se disponibilizado na internet modelos prontos que facilitam o reconhecimento facial. Nesse caso usamos a biblioteca *OpenCV* e *Dlib*, respectivamente, uma carrega a imagem e a outra detecta a face.

```
1 imagem = cv2.imread("praiadagraciosas/treinamento/grupo06.jpg")
2 detector = dlib.get_frontal_face_detector()
3 facesDetectadas = detector(imagem, 1)
4 print(facesDetectadas)
```

Esse algoritmo é baseado no HOG (Histograma de Gradientes Orientados). Para a criação da caixa delimitadora é preciso passar os paramentros *left*, *top*, *right* e *bottom*.

```
1 for face in facesDetectadas:
2       e, t, d, b = (int(face.left()), int(face.top()), int(face.right())
            , int(face.bottom()))
3     cv2.rectangle(imagem, (e, t), (d, b), (0, 255, 255), 2)
```

Passando para o passo dois, é preciso fazer a extração da região de interesse. Para isso, é utilizado um código já treinado para reconhecer 68 pontos faciais e com ele, ajuda a delimitar a região de interesse.

```
1 detectorFace = dlib.get_frontal_face_detector()
2 detectorPontos = dlib.shape_predictor("recursos/
    shape_predictor_68_face_landmarks.dat")
```

```
1 def imprimeLinhas(imagem, pontosFaciais):
2     p68 = [[0, 16, False], # linha do queixo
3            [17, 21, False], # sombrancelha direita
4            [22, 26, False], # sombancelha esquerda
5            [27, 30, False], # ponte nasal
6            [30, 35, True], # nariz inferior
7            [36, 41, True], # olho esquerdo
```

```
8            [42, 47, True], # olho direito
9            [48, 59, True], # labio externo
10           [60, 67, True]] # labio interno
11    for k in range(0, len(p68)):
12        pontos = []
13        for i in range(p68[k][0], p68[k][1] + 1):
14            ponto = [pontosFaciais.part(i).x, pontosFaciais.part(i).y]
15            pontos.append(ponto)
16        pontos = np.array(pontos, dtype=np.int32)
17        cv2.polylines(imagem, [pontos], p68[k][2], (255, 0, 0), 2)
```

4.4 Resultado da detecção de face - Jéssica

- **Base de dados**

  Na seção 4.2 obtém-se a base de dados da plataforma da rede social em que se faz a aplicação. Entretanto, a escolha da não utilização da mesma nesse momento se deu pelo fato de que diante da pesquisa, observou-se que primeiro deve-se realizar testes e treinamentos com retrato dos humanos escolhidos primeiro sozinhos e logo após em grupo, desta maneira o algoritmo está sendo ensinado, para então passar para a utilização do banco de dados obtido pelo *web crawler*. Portanto, são utilizadas duas base de dados para se fazer a detecção de faces, uma com a pessoa sozinha e a outra com várias pessoas na foto, dessa maneira pode-se analisar os resultados do algoritmo usado. A figura 47, representa a base de dados dela individual para realizar o treinamento de detecção para a Jéssica.

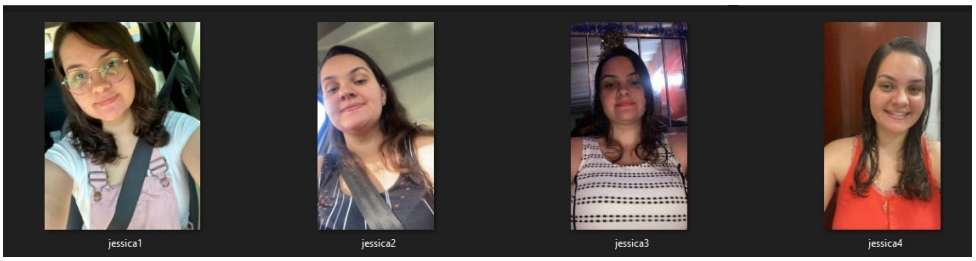


Figura 47 – Base de dados para treinamento Jéssica

  Já a figura 48 representa ela em grupo.

- **Resultado com o algoritmo HOG**

  Na seção 2.4.2 é visto como é a lógica do funcionamento do Algoritmo Histograma de Gradiente Orientado (HOG), já nesta seção, pode-se visualizar o seu resultado.

  Resultado com a base de dados Jéssica sozinha

  Observa-se na figura 49 que em todas as fotos houve a detecção dos rostos, seu resultado é 100%.

  Resultado com a base de dados Jéssica em grupo

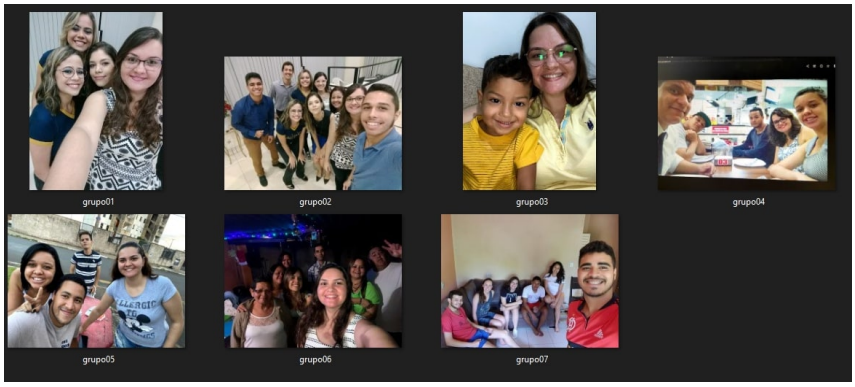
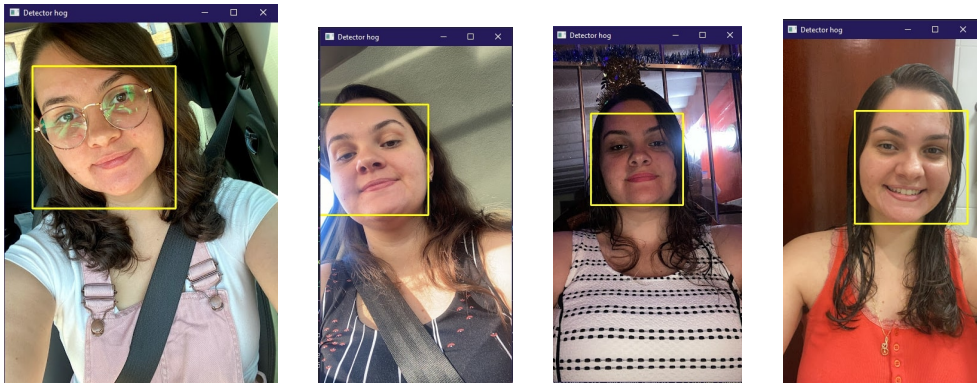
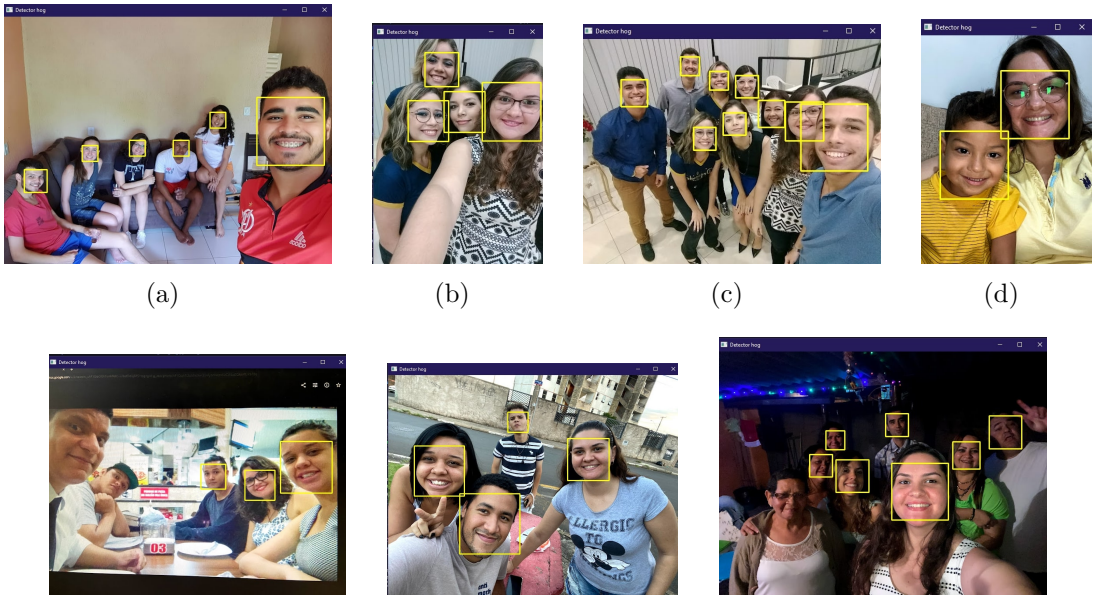


Figura 48 – Base de dados para treinamento em grupo - Jéssica



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

Figura 49 – Algoritmo detecção de face HOG para Jéssica sozinha



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

(e) (f) (g)

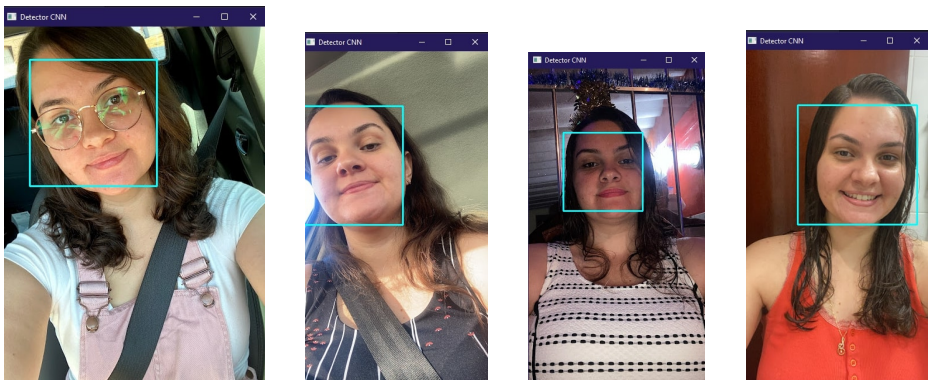
Figura 50 – Algoritmo detecção de face HOG para Jéssica em grupo

Ao analisar a figura 50, referindo a base de dados em grupo, pode-se observar que apenas no (c) e (g) o algoritmo não conseguiu fazer a detecção de todos os rosto, pode-se

levar em consideração a qualidade no (c) e a iluminação ruim no (g) e possivelmente por esses motivos não houve a localização dos rostos, portanto, seu resultado é 71,43% (calculo é feito pelo número de faces detectadas dividido pelo número total de faces e multiplica por 100 para encontrar a porcentagem).

- **Resultado com o algoritmo CNN**
  Como é visto anteriormente o funcionamento do Algoritmo Rede Neural Convolucional (CNN), nesta seção, pode-se visualizar o resultado.

  Resultado com a base de dados Jéssica sozinha

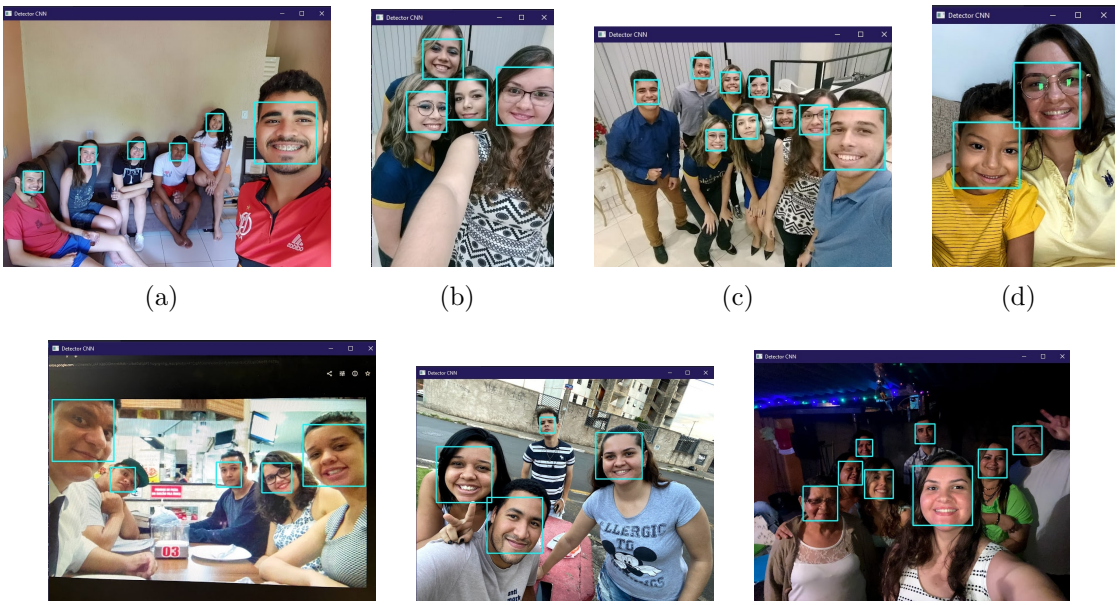


(a) (b) (c) (d)

Figura 51 – Algoritmo detecção de face CNN para Jéssica sozinha

Observa-se na figura 51 que em todas as fotos houve a detecção dos rostos, portanto, seu resultado é 100%.

Resultado com a base de dados Jéssica em grupo



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

(e) (f) (g)

Figura 52 – Algoritmo detecção de face CNN para Jéssica em grupo

Ao analisar a figura 52, referindo a base de dados em grupo, observa-se que todos os rostos são detectados, sendo assim, seu resultado é 100%.

- **Comparação dos resultados obtidos pelos algoritmos HOG e CNN**
  Comparando os algoritmos, nota-se que o CNN foi mais eficiente que o HOG, pois houve 100% de resultado em ambas base de dados, enquanto no HOG seu resultado na base de dados em grupo foi de 71,43%

4.5 Resultado da detecção de face - Neymar

O procedimento é o mesmo com o Neymar, usa-se duas base de dados também onde uma ele está sozinho e a outra em grupo.

- **Resultado com o algoritmo HOG**
  Nesta seção, pode-se visualizar o resultado em relação ao Neymar.
  Resultado com a base de dados Neymar sozinho

(a) (b) (c)

(d) (e) (f)

(g)

Figura 53 – Algoritmo detecção de face HOG para Neymar sozinho

Observa-se na figura 53 que em todas as fotos houve a detecção dos rostos, dando explicação para o (d), a imagem ficou distorcida devido ao seu tamanho, ou seja, a grande quantidade de *pixel* afeta na momento da compitalação e por exigir mais do processador, sua resposta pode ser demorada, embora isso, seu resultado é 100%.

Resultado com a base de dados Neymar em grupo

(a) (b) (c)

(d) (e) (f)

(g)

Figura 54 – Algoritmo detecção de face HOG para Neymar em grupo

Ao analisar a figura 54, referindo a base de dados em grupo, pode-se observar que apenas no (b) e (c) o algoritmo não conseguiu fazer a detecção dos rosto, e no (d) é feito a detecção do rosto no *banner* também, portanto, seu resultado é 71,43%.

- **Resultado com o algoritmo CNN**
  Nesta seção, pode-se visualizar o resultado.

  Resultado com a base de dados Neymar sozinho

(a) (b) (c)

(d) (e) (f)

(g)

Figura 55 – Algoritmo detecção de face CNN para Neymar sozinho

Pode-se notar na figura 55, que em todas as fotos houve a detecção dos rostos, de igual modo que na figura 56, nessa há uma grande quantidade de *pixels* na (d), embora isso, o seu resultado é 100%.

Resultado com a base de dados Neymar em grupo

(a) (b) (c)

(d) (e) (f)

(g)

Figura 56 – Algoritmo detecção de face CNN para Neymar em grupo

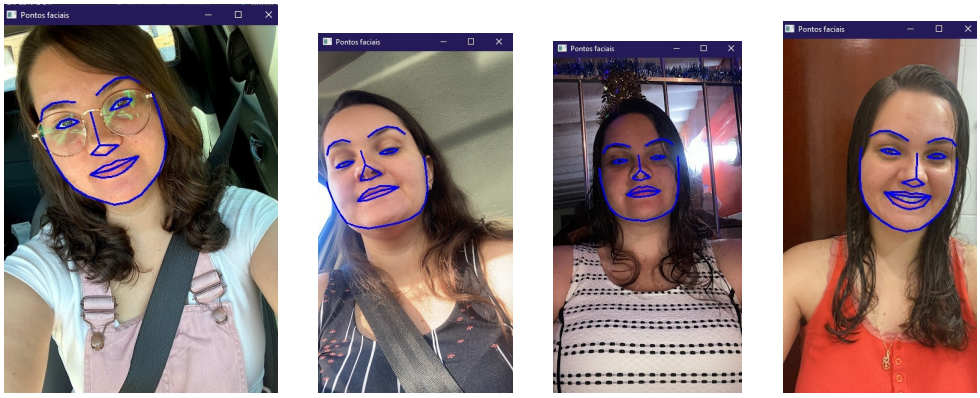
Consta-se na figura 56, referindo a base de dados em grupo, pode-se observar que no (b), (d) e (g) não tive todos as faces localizadas, sendo assim, seu resultado é 57,14%.

- **Comparação dos resultados obtidos pelos algoritmos HOG e CNN**
  Comparando os algoritmos, nota-se que levando em consideram as porcentagens o HOG foi mais eficiente que o CNN neste caso, pois no HOG sua porcentagem é de 71,43% enquanto no CNN é apenas 57,14%.

4.6 Detecção por pontos faciais

A detecção por pontos facial é um algoritmo onde se faz a extração do que tem relevância na imagem e atribuindo pontos na região. Vê-se os resultados dessa cifra.

- **Resultado com a base de dados Jéssica sozinha**

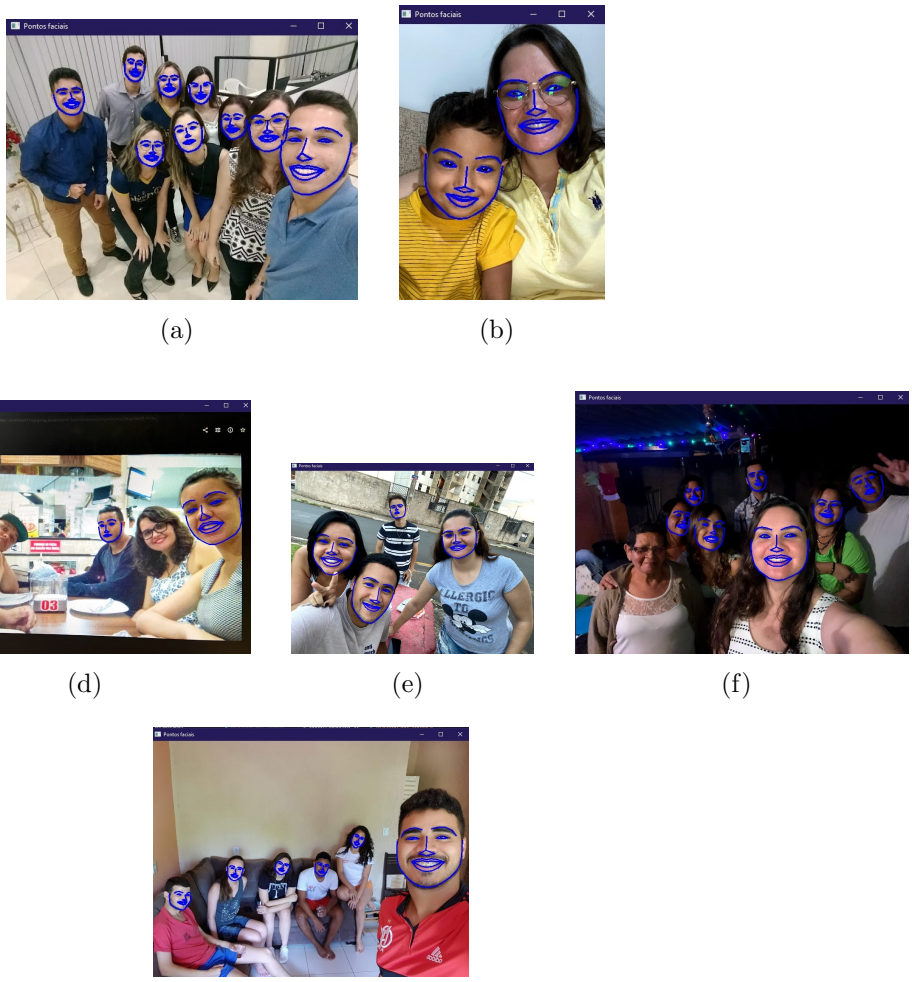


(a) (b) (c) (d)

Figura 57 – Algoritmo detecção de face Ponto Faciais para Jéssica sozinha

Pode-se observa-se na figura 57 que em todas as fotos houve a localização dos pontos faciais no rosto, portanto, seu resultado é 100%.

Resultado com a base de dados Jéssica em grupo



(a) (b)

(c) (d) (e) (f)

(g)

Figura 58 – Algoritmo detecção de face Ponto Faciais para Jéssica em grupo

Observa-se na figura 58, referindo a base de dados em grupo, nota-se que apenas no (d) e (f) o algoritmo não conseguiu fazer a detecção de todos os rosto, da mesma maneira podendo se levar em consideração a qualidade no (d) e a iluminação ruim no (f), já visto em resultados anteriores, e possivelmente por esses motivos não encontrou-se os rostos, portanto, seu resultado é 71,43%.

- **Resultado com a base de dados Neymar sozinho**

(a) (b) (c)

(d) (e) (f)

(g)

Figura 59 – Algoritmo detecção de face Pontos Faciais para Neymar sozinho

De maneira igual ao anterior, pode-se notar na figura 59 que em todas as fotos houve a localização dos pontos faciais no rosto, e o (d) já foi detalhado anteriormente, portanto, seu resultado é 100%.

Resultado com a base de dados Neymar em grupo

Ao que vê-se a figura 60, referindo a base de dados em grupo, averiguá-se que apenas no (b) e (c) o algoritmo não conseguiu encontrar todas as faces, da mesma maneira podendo se levar em consideração a qualidade no (d) em que o algoritmo faz o reconhecimento do Neymar no *banner* já visto em resultados anteriores, portanto, seu resultado é 71,43%.

(a) (b) (c) (d) (e) (f) (g)

Figura 60 – Algoritmo detecção de face Ponto Faciais para Neymar em grupo

4.7 Reconhecimento facial

Após fazer a detecção do rosto o próximo passo é efetuar o reconhecimento facial. Em um primeiro momento a base de dados a ser realizado os testes são os mesmos utilizados para detecção dos rostos, ou seja, as fotos de Jéssica e a do Neymar em grupo e individual As fotos de treinamento de rosto é a da base de dados de sozinho e em grupo para cada um.

Então, cria-se um *loop* para usar todas as fotos em formato jpg no diretório de praiadagraciosas/treinamento/treinamento para a Jéssica e neymarjrs/treinamento/treinamento para o Neymar. Depois disso, aplica-se outro *loop* à imagem onde o rosto é detectado, a partir daí detém-se a detecção de pontos de rosto, e também aplica-se o reconhecedor de rosto via método *compute-face-descriptor*(). Depois disso, é necessário se fazer algumas conversões para que o valor encontrado fique em um formato de *array numpy*, o que ajudará a comparar com a nova imagem em que se passa no teste. Por fim, há a iteração sobre cada índice associado à imagem treinada.

```
1 for arquivo in glob.glob(os.path.join("praiadagraciosas/treinamento/
    treinamento", "*.jpg")):
```

```
2       imagem = cv2.imread(arquivo)
3       facesDetectadas = detectorFace(imagem, 1)
4       numeroFacesDetectadas = len(facesDetectadas)
5
6       if numeroFacesDetectadas > 1:
7           print("Ha mais de uma face na imagem {}".format(arquivo))
8           exit(0)
9       elif numeroFacesDetectadas < 1:
10          print("Nenhuma face encontrada no arquivo {}".format(arquivo))
11          exit(0)
12
13      for face in facesDetectadas:
14          pontosFaciais = detectorPontos(imagem, face)
15          descritorFacial = reconhecimentoFacial.compute_face_descriptor(
                imagem, pontosFaciais)
16
17          listaDescritorFacial = [df for df in descritorFacial]
18
19          npArrayDescritorFacial = np.asarray(listaDescritorFacial, dtype=
                np.float64)
20
21          npArrayDescritorFacial = npArrayDescritorFacial[np.newaxis, :]
22
23          if descritoresFaciais is None:
24              descritoresFaciais = npArrayDescritorFacial
25          else:
26              descritoresFaciais = np.concatenate((descritoresFaciais,
                    npArrayDescritorFacial), axis=0)
27
28          indice[idx] = arquivo
29          idx += 1
30
31  np.save("recursos/descritores_jess.npy", descritoresFaciais)
32  with open("recursos/indices_jess.pickle", 'wb') as f:
33      cPickle.dump(indice, f)
```

A execução deste *script* irá eventualmente criar dois arquivos: *descriptores-jess.npy* e *indices-jess.pickle*, ressaltando que arquivo é para a Jéssica, para o do Neymar os arquivos se chamam *descriptores-ney.npy* e *indices-ney.pickle*.

4.8 Resultado Reconhecimento facial Jéssica

O algoritmo KNN é utilizado aqui em conjunto com o CNN, seu papel é procurar o vizinho mais próximo para conseguir categorizar a imagem, em outras palavras, com a entrada da imagem de treinamento, ou seja, as fotografias com faces únicas, o algoritmo é pré-treinado para aguardar as informações do rosto, então faz-se a entrada de uma nova

imagem da qual será feita a comparação com a imagem já treinada e buscar características próximas, o KNN faz cálculo da distância euclidiana e então retorna o valor final. Aqui é colocado uma limiar = 0,5, logo, todo valor dado do resultado da distância for de zero até 0,5 é categorizado como o rosto da Jéssica e os valores acima significa que não é a face da mesma. Quanto menor for a acurácia, mais próximo de ser a face é buscada.

- **Base de dados**
  A base de dados usada é a mesma que é feita a detecção de faces, uma com a pessoa sozinha e a outra com várias pessoas na foto, dessa maneira pode-se analisar os resultados do algoritmo usado. A figura 61, representa a base de dados com o treinamento de reconhecimento facial para a Jéssica realizado.

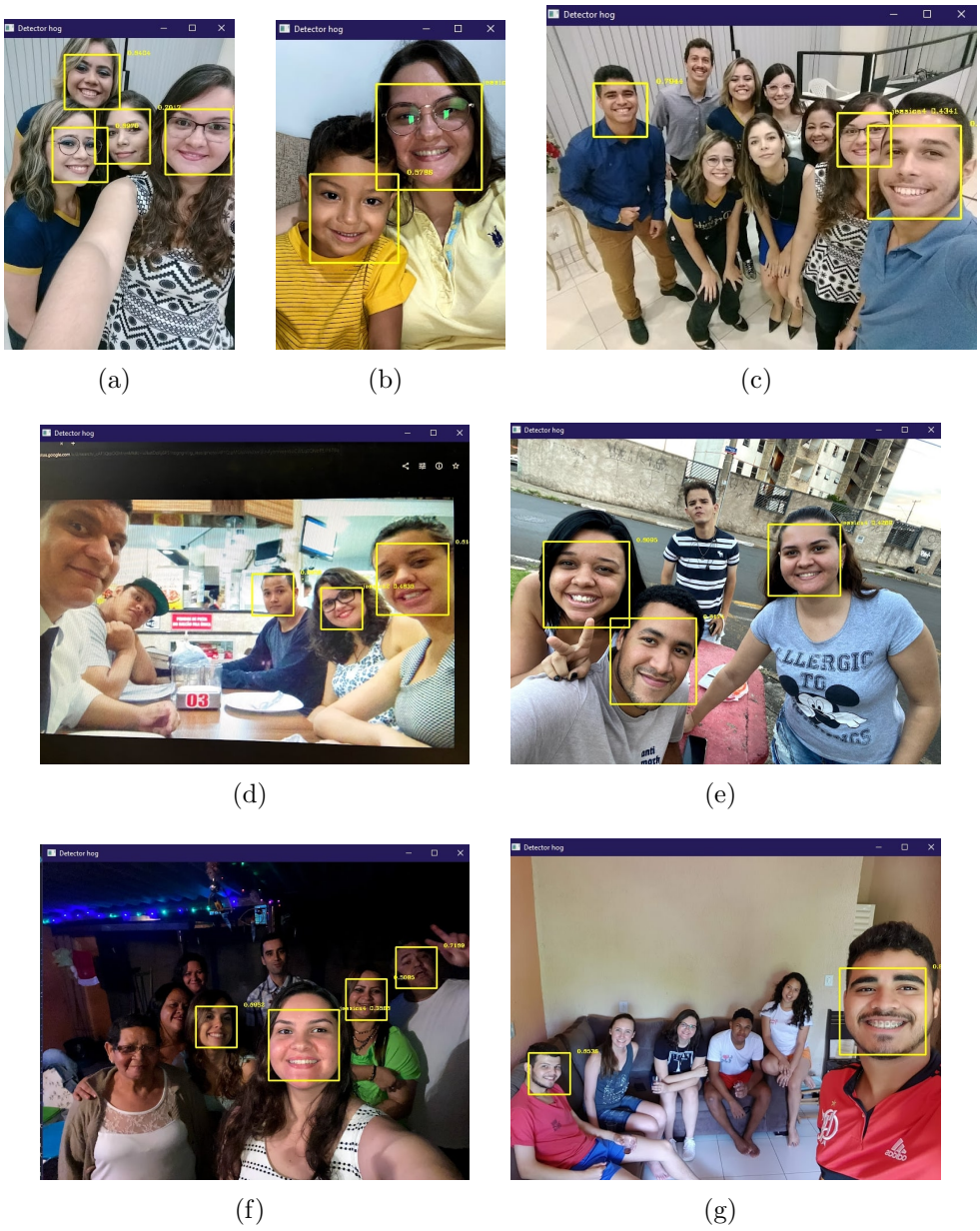


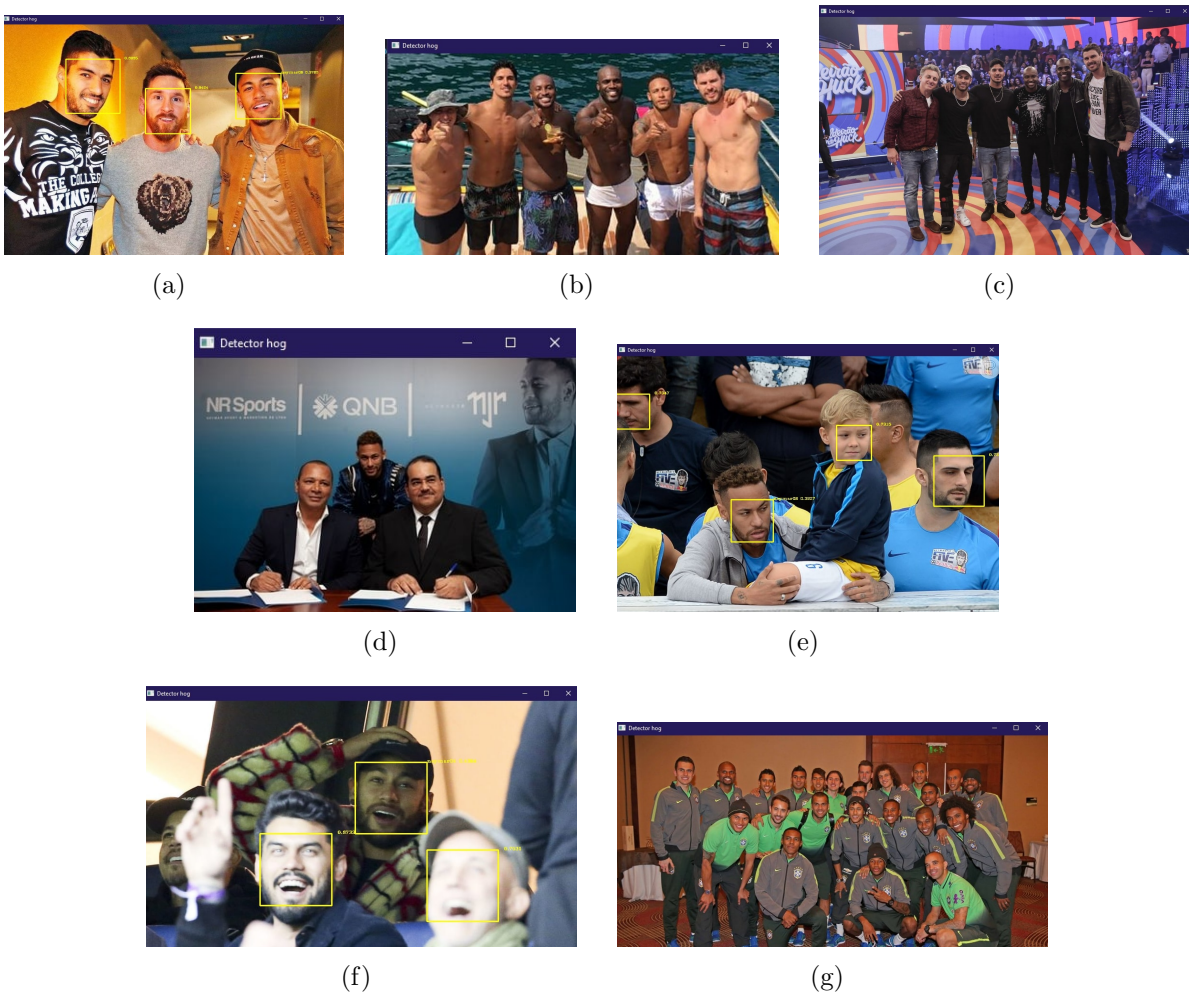
Figura 61 – Algoritmo reconhecimento facial para Jéssica

Constata-se na figura 61, que o algoritmo não conseguiu reconhecer a face da Jéssica no (g), outro detalha que se pode reparar é a não localização de outras faces em algumas imagens como no (c) e (f). Desde modo, o resultado foi que em 7 imagens o algoritmo conseguiu encontrar o rosto buscado em 6, logo seu resultado é 85%.

4.9 Resultado Reconhecimento facial Neymar

- **Base de dados**

  De maneira idêntica ao tópico anterior. A figura 62, representa a base de dados com o treinamento de reconhecimento facial para a Neymar feito.



(a) (b) (c)

(d) (e)

(f) (g)

Figura 62 – Algoritmo reconhecimento facial para Neymar

Repara-se na figura 62, que o algoritmo não conseguiu reconhecer a face da Neymar no (b), (c), (d) e (g), além disso, muitas faces não foram localizadas. Desde modo, o resultado é que dentre as 7 imagens contidas no banco de dados, apenas 3 dessas tiveram rostos localizados e classificado, logo seu resultado é 42,86%. Fazendo a analise, uma probabilidade de o resultado ter sido baixo pode ser a quantidade de fotos treinadas, ou seja, quanto mais fotos e de diversas posições e ângulo existirem na base de dados, maior é a quantidade do treinamento e classificação da inteligência artificial.

4.10 Utilizando a base extraída do *instagram*

Após os processos das seções anteriores, parte-se então para o elemento da base de dados, extraída na seção 4.2, que é o ponto chave deste trabalho. Visto os primeiro passos, houve uma coleta de dados extraída do *instagram*, armazenamos e é com ela que deve-se trabalhar, pois ela que irá dizer se tem ou não alguma foto de uma determinada pessoa espalhada na internet de maneira que ela não saiba. Dessa forma, é feita a verificação do treinamento e a análise de seu funcionamento e então, é feito o reconhecimento. Diante

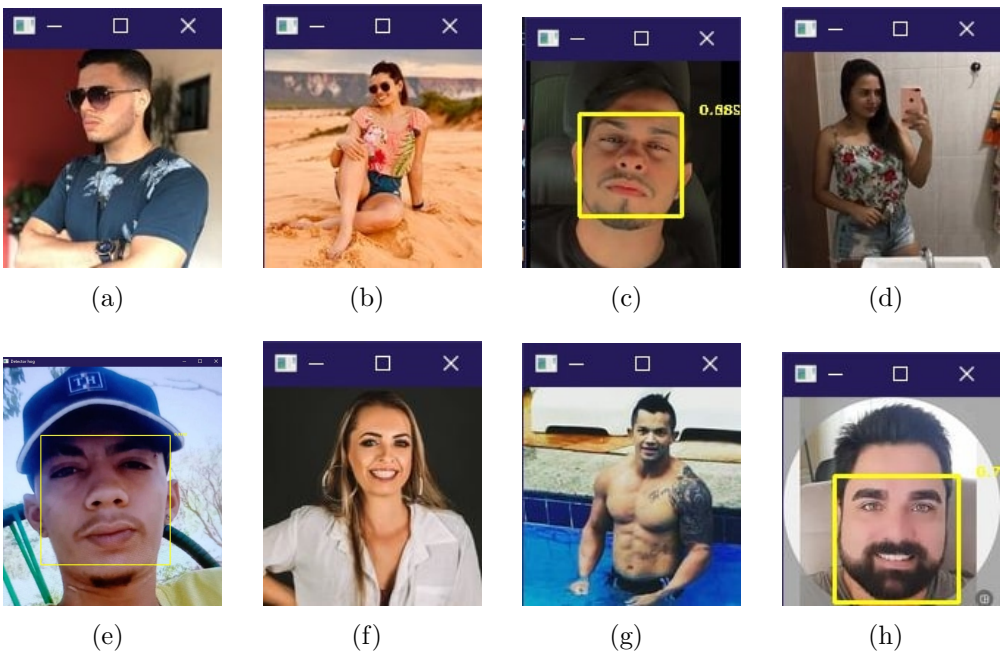
disso, tem-se o seguinte código:

```
limiar = 0.5
for arquivo in glob.glob(os.path.join("praiadagraciosas", "*.jpg")):
    imagem = cv2.imread(arquivo)
    facesDetectadas = detectorFace(imagem)
    for face in facesDetectadas:
        e, t, d, b = (int(face.left()), int(face.top()), int(face.right
            ()), int(face.bottom()))
        pontosFaciais = detectorPontos(imagem, face)
        descritorFacial = reconhecimentoFacial.compute_face_descriptor(
            imagem, pontosFaciais)
        listaDescritorFacial = [fd for fd in descritorFacial]
        npArrayDescritorFacial = np.asarray(listaDescritorFacial, dtype=
            np.float64)
        npArrayDescritorFacial = npArrayDescritorFacial[np.newaxis, :]

        distancias = np.linalg.norm(npArrayDescritorFacial -
            descritoresFaciais, axis=1)
        print("Distancias: {}".format(distancias))
        minimo = np.argmin(distancias)
        print(minimo)
        distanciaMinima = distancias[minimo]
        print(distanciaMinima)

        if distanciaMinima <= limiar:
            nome = os.path.split(indices[minimo])[1].split(".")[0]
        else:
            nome = ' '

        cv2.rectangle(imagem, (e, t), (d, b), (0, 255, 255), 2)
        texto = "{} {:.4f}".format(nome, distanciaMinima)
        cv2.putText(imagem, texto, (d, t), cv2.
            FONT_HERSHEY_COMPLEX_SMALL, 0.5, (0, 255, 255))

    cv2.imshow("Detector hog", imagem)
    cv2.waitKey(0)

cv2.destroyAllWindows()
```

Pode-se observar que este código é muito semelhante ao anterior. Porém, aqui encontra-se alguns detalhes: o primeiro é exportar os arquivos que é gravado durante o treinamento, a saber, índices e descritores. Também é definido um limite, porque agora é feito a comparação do valor de cada imagem de teste com o valor do descritor facial gerado a partir da imagem de treinamento. Nesse caso, a distância entre esses valores deve ser menor ou igual ao valor que foi definido inicialmente no limiar para que se possa provar que o rosto é a de Neymar e da Jéssica (trecho exemplificado na seção anterior).

4.10.1 **Resultado da classificação com a base de dados do *instagram* - Jéssica**

Com a base de dado já pré-treinada com o rosto dela, é inserido a banco de dado que é coletado do *instagram* e realiza a verificação comparando cada foto na busca de encontrar alguma foto parecida. Pela base de dados conter 42 imagens, nos resultados será mostrado 9 deles, aqueles que se aproximaram mais o resultado esperado. Diante disso, constata-se os resultados na figura 63 abaixo:



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

(e) (f) (g) (h)

Figura 63 – Resultado do reconhecimento facial com o banco de dados extraído do *instagram* - Jéssica

Das 42 imagens coletadas, apenas 3 delas é detectada a face e das 3 nenhuma é da Jéssica, portanto, mesmo mediante aos riscos pelas outras imagens não terem sido reconhecidas e classificadas, pode-se dizer que o resultado é 100%.

4.10.2 **Resultado da classificação com a base de dados do *instagram* - Neymar**

Na base extraída do *instagram* contém 33 fotos. Com o algoritmo pré-treinado para o reconhecimento da face do Neymar, obteve-se os resultados. Devido a grande quantidade de foto, é selecionado apenas 9 contendo os melhores resultados para apresentar, como pode-se averiguar na figura 64 abaixo:

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

(e) (f) (g) (h)

(i)

Figura 64 – Algoritmo reconhecimento facial para Jéssica

Das 33 fotos que obteve, o algoritmos detectou face em 13, senda 9 dessas classificada com o rosto do Neymar. Logo, diante do fato de o Neymar ser um jogar mundialmente conhecido e com diversos fãs espalhados pelo mundo, desta maneira o resultado é de 72,72% de aprovação.

5 CONCLUSÃO E TRABALHOS FUTUROS

O presente trabalho apresentou uma visão ampla dos conceitos e métodos de detecção e reconhecimento facial e explicações detalhadas sobre os algoritmos CNN (Rede Neural Convolucional), HOG (Histograma de Gradientes Orientados) e KNN (K-vizinhos mais próximos), incluindo a implementação de sistemas computacionais capazes de detectar e reconhecer faces e classificar utilizando esses algoritmos.

Para atingir o objetivo de fornecer base teórica para a implementação os sistemas de detecção e reconhecimento de rosto requerem extensa pesquisa bibliografia.

Foi possível mostrar a comparação dos algoritmos HOG e CNN utilizando a biblioteca *OpenCV* e *Dlib* e verificar a performance no processo de detecção da face. Foi mostrado os resultados obtidos em todas as etapas e que o algoritmo CNN para a detecção de face junto com o KNN para o reconhecimento da mesma e a sua classificação trouxe os resultados de 72,72% para aprovação do Neymar com a base de dados extraída do *instagram* e 100% de aprovação no da Jéssica, onde houve o reconhecimento de 9 e zero faces, respectivamente.

Neste presente trabalho, a complexidade do algoritmo foi baixa se fazendo o uso da parâmetros existentes na biblioteca para a realização da detecção e os parâmetros passados foram as imagens obtidas através da pesquisa feita no *Google* para os testes e as fotos obtidas no *Instagram* com a coleta de dados. Devido a esse fator, não pode-se dizer o numero de camadas, o formato dos filtros e a arquitetura.

Pode-se notar de forma clara que o reconhecimento facial ainda é um desafio no campo do processamento de imagens, sendo necessário superar diversas dificuldades causadas por fatores como iluminação, baixa resolução, alterações posturais, oclusão e expressões faciais.

5.1 Para trabalhos futuros pretende-se:

- Propõe-se buscar melhorias no quesito para extração dos dados do *instagram*.
- Utilização de um algoritmo mais complexo definindo a arquitetura, quantidade de camadas, filtros utilizados, camadas internas, etc.
- Reconhecimento por vídeo.
- Estudar uma forma de acrescentar busca por perfis através de fotos, ou seja, há a entrada de uma foto e o algoritmo irá fazer uma busca para encontrar perfis que o reconhecimento facial se aproxime da pessoa. Este estudo é complexo e querer muito tempo e testes, além de que a plataforma *instagram* tem politicas que podem ser violadas e isso caracterizaria crime.

- Estudar formas de na raspagem dos dados, para que as imagens coletadas não sejam repetidas.
- Uma base de dados maior para um melhor treinamento de algoritmo de detecção facial e reconhecimento facial.
- Construção de um aplicativo para acesso ao público em geral, onde o usuário possa escolher em qual categoria ele quer que a *hashtag* entre, ou seja, se é nas mais recentes, de mais relevância ou *reels*.

REFERÊNCIAS

ALIGER, E. **As Redes Neurais Convolucionais no Deep Learning | Aliger | A empresa da Inteligencia das coisas**. 2019. Disponível em: <https://www.aliger.com.br/blog/as-redes-neuronais-convolutivas-no-deep-learning/>.

ATHENIENSE, A. **Ter um perfil falso na internet é crime?** Jusbrasil, 2010. Disponível em: <https://alexandre-atheniense.jusbrasil.com.br/noticias/2122641/ter-um-perfil-falso-na-internet-e-crime>.

BA, n. **Golpistas criam perfis falsos em redes sociais para aplicar golpes em clientes de empresas na BA; especialista faz alerta**. 2021. Disponível em: <https://g1.globo.com/ba/bahia/noticia/2021/02/05/golpistas-criam-perfis-falsos-em-redes-sociais-para-aplicar-golpes-em-clientes-de-hamburgueria-e-pou
ghtml>.

BARBOSA, W. R.; TEIXEIRA, H. P. **Perfil falso e suas consequências**. Jus Navigandi, 2020. Disponível em: <aa>.

CLIENTES, C. criam perfis falsos nas redes sociais e se passam por empresas para dar golpe em. **Criminosos criam perfis falsos nas redes sociais e se passam por empresas para dar golpe em clientes**. 2021. Disponível em: <https://g1.globo.com/to/tocantins/noticia/2021/02/16/criminosos-criam-perfis-falsos-nas-redes-sociais-e-se-passam-por-empresas-para-dar-golpe-em-cliente
ghtml>.

CLOUDFLARE. **O que é raspagem de dados?** 2021. Disponível em: <https://www.cloudflare.com/pt-br/learning/bots/what-is-data-scraping/>.

CLOUDFLARE. **What is content scraping? | Web scraping**. 2021. Disponível em: <https://www.cloudflare.com/pt-br/learning/bots/what-is-content-scraping/>.

DEEPNOTE. **Face Recognition Demo**. 2021. Disponível em: <https://deepnote.com/project/Face-Recognition-Demo-PmD68NZqTJuZENRj4q1jnw/%2Fexample.ipynb/#00001-543e8415-0801-4538-a669-2eae5e7989ce>.

E-LIBRARY. **O que são redes neurais convolucionais? - Electrical e-Library.com**. 2018. Disponível em: <https://www.electricalelibrary.com/2018/11/20/o-que-sao-redes-neurais-convolucionais/>.

FACERECOGNITIONSOLUTION.COM. **Face Recognition System - Biometric Face Detection Verification System**. 2014. Disponível em: <http://www.facerecognitionsolution.com/index.html>.

FEITOSA, P. **Criar fake na internet é crime**. Jusbrasil, 2020. Disponível em: <https://patriciafeitosa323.jusbrasil.com.br/artigos/826820814/criar-fake-na-internet-e-crime?ref=feed>.

GOGONI, R. **O que é um pixel? – Tecnoblog**. 2019. Disponível em: <https://tecnoblog.net/295290/o-que-e-um-pixel/>.

GOGONI, R. **Qual foi a primeira rede social criada na internet? | Internet | Tecnoblog**. 2019. Disponível em: <https://tecnoblog.net/315992/qual-foi-a-primeira-rede-social-criada-na-internet/>.

HIMAWAN, A.; PRIADANA, A.; MURDIYANTO, A. Implementation of web scraping to build a web-based instagram account data downloader application. **IJID (International Journal on Informatics for Development)**, v. 9, p. 59–65, 12 2020. Disponível em: <http://ejournal.uin-suka.ac.id/saintek/ijid/article/view/09201>.

ICHI.PRO. **Uma introdução suave ao histograma de gradientes orientados**. 2020. Disponível em: <https://ichi.pro/pt/uma-introducao-suave-ao-histograma-de-gradientes-orientados-279201309061802>.

IME.USP.BR. 2021. Disponível em: <https://www.linux.ime.usp.br/~cef/mac499-06/monografias/andre/WebCrawler.html>.

JHONATTAN, S.; GHELLERE. **UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ -UTFPR DEPARTAMENTO ACADÊMICO DE COMPUTAÇÃO CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO**. 2015.

KLEINA, N. **Como funciona o reconhecimento facial**. TecMundo, 2021. Disponível em: <https://www.tecmundo.com.br/camera-digital/10347-como-funcionam-os-sistemas-de-reconhecimento-facial.htm>.

LEOCáDIO, R. **RGB e CMYK – Principal Diferença entre o Sistema RGB e CMYK**. 2016. Disponível em: <https://www.futuraexpress.com.br/blog/rgb-e-cmyk/>.

MASTER, C. S. **Como funciona o algoritmo KNN ■ Computer Science Master**. 2019. Disponível em: <https://www.computersciencemaster.com.br/como-funciona-o-algoritmo-knn/>.

PLANALTO.GOV.BR. **Constituição**. 2016. Disponível em: <http://www.planalto.gov.br/ccivil_03/constituicao/constituicao.htm>.

PROCESSAMENTO. 2021. Disponível em: <chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/viewer.html?pdfurl=http%3A%2F%2Fwww.cbpf.br%2Fcat%2Fpdsi%2Fpdf%2Fcap3webfinal.pdf&clen=335440&chunk=true>.

RIBEIRO, D. E. F. **Armazenamento e Recuperação de Informações**. [S.l.]: Editora, 2021. v. 1.

ROBERTO, T.; BLUMENAU, F. **UNIVERSIDADE REGIONAL DE BLUMENAU CENTRO DE CIÊNCIAS EXATAS E NATURAIS CURSO DE CIÊNCIAS DA COMPUTAÇÃO -BACHARELADO IMPLEMENTAÇÃO DE UM CRAWLER INCREMENTAL DISTRIBUÍDO: UM SISTEMA DE BUSCA NA WEB**. 2006. Disponível em: <http://campeche.inf.furb.br/tccs/2006-I/2006-1tiagorobertofischervf.pdf>.

UNIVEM.EDU.BR. **View of Plataforma de extração e recuperação de dados na Web no contexto de Big Data**. 2021. Disponível em: <https://revista.univem.edu.br/jadi/article/view/1038/389>.

UOL. **Criar fake na internet só é crime se for baseado em pessoa real**. UOL, 2019. Disponível em: <https://www.uol.com.br/tilt/noticias/redacao/2019/04/14/criar-fake-na-internet-so-e-crime-se-for-baseado-em-pessoa-real.htm>.

UOL. **Folha Online - Cotidiano - Internet foi criada em 1969 com o nome de "Arpanet"nos EUA - 12/08/2001**. 2021. Disponível em: <https://www1.folha.uol.com.br/folha/cotidiano/ult95u34809.shtml>.

VERZBICKAS, A. et al. **UNIVERSIDADE FEDERAL DE SANTA CATARINA CENTRO TECNOLÓGICO DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA CURSO DE SISTEMAS DE INFORMAÇÃO INE5644 DATA MINING RELATÓRIO WEB CRAWLERS**. 2013. Disponível em: <http://www.inf.ufsc.br/~luis.alvares/INE5644/G1_WebCrawlers.pdf>.

WIKIMEDIA, C. dos projetos da. **Morfologia matemática**. Fundação Wikimedia, Inc., 2005. Disponível em: <https://pt.wikipedia.org/wiki/Morfologia_matem%C3%A1tica>.

ZHAO, B. **Web Scraping**. unknown, 2017. Disponível em: <https://www.researchgate.net/publication/317177787_Web_Scraping>.