Utilização de Reconhecimento Facial para Auxílio de Deficientes Visuais

Mateus H. Ittner[1], Cristiano Bertolini[1]

[1]Curso de Bacharelado em Sistemas de Informação

Departamento de Tecnologia da Informação (UFSM)

Universidade Federal de Santa Maria (UFSM) - Campus Frederico Westphalen - Linha 7 de Setembro, s/n, CEP: 98400-000, BR 386 Km 40- Frederico Westphalen - RS

matittner@gmail.com, cbertolini@gmail.com;

Resumo. *Pessoas com deficiências visuais possuem dificuldades em identificar outras pessoas ao seu redor. Neste contexto, este trabalho busca desenvolver um aplicativo mobile utilizando ferramentas open source de deep learning e machine learning, pré-treinados para reconhecimento facial. O aplicativo utiliza um servidor remoto para realizar o processamento de imagens obtidas por um smartphone, a fim de efetuar o reconhecimento facial por meio de um banco de dados formado por fotos de funcionários da UFSM, com o intuito de auxiliar deficientes visuais em seu dia a dia na universidade. Os resultados obtidos mostram o potencial da tecnologia, alcançando bons resultados no reconhecimento sem a necessidade de ajuda de acessórios externos.*

Palavras-Chave: *Deficiência Visual, Reconhecimento Facial, Aprendizado de Máquina, Tecnologias Assistivas*

Abstract. *People with visual impairments have great difficulties in identifying persons around them, using advances in the machine learning area, this paper seeks to develop a mobile application using pre-trained open source deep learning and machine learning tools for facial recognition, using a remote server to process images obtained from a smartphone, in order to perform facial recognition through a database formed by photos from UFSM workers, seeking to help people with visual difficulties in their day to day life at the university. The results achieved shows the potential of this technology, obtaining good recognition results without the need from external tools.*

Keywords: *Visual Impairments, Facial Recognition, Machine learning, Assistive Technologies*

1. Introdução

Deficientes visuais possuem dificuldades em identificar outras pessoas ao seu redor, utilizando da voz dessas pessoas para a identificação. Assim, nem sempre o deficiente visual lembra da voz de uma pessoa, especialmente se é alguém com quem não tem muito contato ou não conhece [1]. Neste contexto, este trabalho apresentado o desenvolvimento de um aplicativo de reconhecimento facial de funcionários e servidores da UFSM, visando a oferecer uma melhora na qualidade de vida de deficientes visuais que frequentam a instituição, propiciando autonomia e confiança adquiridas pela possibilidade de identificar pessoas por meio de um *smartphone*, sem a necessidade de perguntar com quem se está falando ou a possibilidade de engano e o constrangimento social resultante.

Atualmente, devido ao avanço na área de Inteligência Artificial e a facilidade de acesso a poder computacional, várias APIs (*application programming interface*) são desenvolvidas para reconhecimento facial. A utilização delas vão desde a área de segurança até jogos digitais e hoje estão presentes em toda a parte, como por exemplo, em propagandas inteligentes que julgam reações de quem as assiste [2].

Devido à crescente ubiquidade dessa emergente tecnologia, várias soluções de código aberto se encontram disponíveis e com a universalidade alcançada por *smartphones*, buscamos o desenvolvimento de um aplicativo na área de acessibilidade, a fim de ajudar pessoas combinando as tecnologias citadas, com o objetivo de melhorar a qualidade de vida, aumentando a segurança e acesso a informações corretas por meio da correta identificação de servidores relacionados à UFSM.

Sendo assim, este trabalho apresenta desenvolvimento de um aplicativo para *smartphones,* capaz de realizar o reconhecimento facial dos servidores da UFSM, utilizando ferramentas de código aberto (*open source*) e apostando no processamento remoto de dados, a fim de oferecer uma solução de maior acessibilidade para o usuário. O resultado compreende um aplicativo gratuito, de fácil utilização e de alta performance, independente do *hardware* utilizado.

Para dar conta desta proposta, o artigo está estruturado da seguinte forma: a seção 2 apresenta o referencial teórico, conceituando os principais temas e tecnologias relacionadas ao trabalho. A seção 3 apresenta trabalhos relacionados. A seção 4 elabora a solução desenvolvida, mostrando o aplicativo implementado. Concluindo o artigo, são apresentadas as considerações finais na seção 5 e as referências empregadas.

2. Referencial Teórico

Esta seção apresenta um breve referencial teórico sobre as áreas envolvidas neste trabalho, focando na Deficiência Visual, Reconhecimento Facial, Aprendizado de Máquina, Aprendizado Profundo e Tecnologias Assistivas.

2.1 Deficiência Visual

Deficiência visual é a perda ou diminuição permanente da capacidade visual, não corrigível por lentes ou cirurgias. A expressão ‘deficiência visual’ se refere ao espectro que vai da cegueira até a visão subnormal (ou baixa visão) [3]. Números da OMS (Organização Mundial da Saúde) de 2019 mostram que 1 bilhão de pessoas no mundo são classificadas como deficientes visuais [4], com dados do IBGE (Instituto Brasileiro de Geografia e Estatística) de 2015 apontando entre 7-8 milhões no Brasil [5].

As causas da deficiência visual podem ser congênitas ou adquiridas, sendo sua classificação baseada em um padrão de eficiência visual, que é de certo modo abstrato. Tem sido empregada, cada vez mais, uma definição funcional que enfatiza os efeitos da limitação visual sobre a habilidade crítica da leitura. O instrumento padrão é a Escala de *Snellen*, que consiste em fileiras de letras de tamanhos decrescentes que devem ser lidas a uma distância de 20 pés. Os escores são baseados na exatidão com que a pessoa com deficiência visual foi capaz de identificar as fileiras de letras utilizando um olho de cada vez.

Seguindo essa definição pela capacidade de leitura, uma pessoa cega é aquela que possui perda total ou resíduo mínimo de visão, necessitando do método *Braille* como meio de leitura e escrita ou outros métodos, recursos didáticos e equipamentos especiais para a leitura e uma pessoa com baixa visão é aquela que possui resíduos visuais em grau que permitam ler textos impressos à tinta, desde que se empreguem recursos didáticos e equipamentos especiais, excluindo as deficiências facilmente corrigidas pelo uso adequado de lentes. [6].

Sofrimento psicológico, dificuldades nas atividades da vida diária e baixa qualidade de vida relacionada à saúde têm sido consistentemente relatadas em pessoas com deficiências visuais, sofrimentos comparáveis a problemas muito mais sérios, como acidentes vascular cerebrais (AVCs), quando avaliados quesitos como mobilidade, capacidade de cuidado pessoal, atividades diárias (trabalho, estudo, tarefas domésticas, etc.), desconforto, depressão, entre outros [7,8].

2.2 Reconhecimento Facial

O reconhecimento fácil é uma tecnologia capaz de identificar uma pessoa por meio de uma imagem ou vídeo. Há diversos meios de reconhecimento facial, o mais utilizado é a comparação com imagens de um banco de dados ou com informações obtidas em cima dele. Esses algoritmos de reconhecimento facial usam detalhes do rosto de uma pessoa, como a distância entre os olhos, formato do rosto, entre outros. Esses detalhes são convertidos em uma representação matemática e comparados com o banco de dados [1]. Outros meios de reconhecimento seriam métodos de análise de textura da pele (similarmente a técnicas de biometria com impressões digitais) [9], uso de câmeras infravermelhas [10] e métodos de construção de modelos faciais 3D [11].

A maioria dos sistemas de reconhecimento facial usam probabilidade para reconhecer alguém, trazendo uma porcentagem e não uma simples resposta “Sim” ou “Não”. Esses sistemas são notavelmente fracos em identificar pessoas em situações de baixa luminosidade, baixa resolução e ângulos ruins, sendo esses problemas mitigados com uma grande base de dados para comparação e tecnologias de suporte, como iluminadores infravermelhos.

Sistemas de reconhecimento facial inicialmente utilizavam simples comparações de imagens de banco de dados para realizar o reconhecimento. Hoje, geralmente, utilizam técnicas de *Machine Learning* e *Deep Learning* para criar vetores matemáticos representando pessoas utilizando de suas características faciais [12].

Atualmente, a maior aplicação de sistemas de reconhecimento facial é em sistemas de segurança utilizados pelo governo ou grandes empresas, como, por exemplo, verificação de passaportes, segurança bancária, verificação de identidade em votações eleitorais, entre outros. Mas, devido a avanços na tecnologia, maior acesso às câmeras pessoais e poder de processamento, essa tecnologia está começando a se popularizar e ser utilizada em aplicações mais cotidianas, como a buscada por este trabalho.

Esses avanços também trazem preocupações com a privacidade, já que sua utilização não necessita de permissão ou sequer de conhecimento do participante.

Plataformas como *Facebook* e *Instagram* fornecem um enorme banco de dados público, que, em conjunto com dados e infraestrutura dos governos, torna possível o rastreamento e a observação de pessoas em qualquer lugar, possibilitando um nível de controle comparado por muitos ao famoso *Big Brother* e ao livro *1984* de George Orwell. [13,14]

2.3 Tecnologias Assistivas

Tecnologia Assistiva é um termo utilizado para identificar recursos e serviços que contribuem para proporcionar, ampliar e reabilitar habilidades funcionais de pessoas com deficiência e, com isso, promover qualidade de vida, inclusão social e independência pessoal [15]. Exemplos dessas tecnologias são cadeiras de rodas, *Braille*, ferramentas *text-to-speech*, entre outros [16].

O termo tecnologia assistiva tem sua origem ligada à criação de uma lei nos Estados Unidos em 1988, que define tecnologias assistivas como qualquer item, equipamento ou produto, adquirido comercialmente pronto para uso, modificado ou personalizado, usado para aumentar, manter ou melhorar o funcionamento capacidades de uma criança com deficiência [17].

Estudos também apontam que o uso de tecnologias pode trazer um estigma para seus usuários, devido à expectativa e limitações por parte das pessoas para com a tecnologia, a aparência dos dispositivos, medo de dependência por parte de pais e familiares, aceitação social e exposição pública, entre outros aspectos [18].

2.4 Aprendizado de Máquina (*Machine Learning*)

A Inteligência Artificial (IA) pode ser definida como a ideia de criar máquinas pensantes, máquinas capazes de imitar a capacidade humana de pensar, perceber o mundo e identificar objetos à nossa volta e até mesmo falar e compreender nossa linguagem [19]

A IA é uma das ciências mais recentes, tendo início após a Segunda Guerra Mundial e, atualmente, abrange uma enorme variedade de subcampos, desde áreas de uso geral, como aprendizado e percepção, até tarefas específicas como jogos de xadrez, demonstração de teoremas matemáticos, criação de poesia e diagnóstico de doenças. A IA sistematiza e automatiza tarefas intelectuais e, portanto, é potencialmente relevante para qualquer esfera da atividade intelectual humana [20].

Machine learning é uma das áreas de estudo da IA, compreendendo algoritmos que, simplificadamente, se aprimoram automaticamente por meio da experiência. Esses algoritmos criam um modelo matemático baseado em dados para fazer predições ou decisões sem serem especificamente programados para isso [21]. Desenvolvedores que se utilizam dessa metodologia buscam o desenvolvimento de algoritmos que modificam e adaptam suas ações para que as mesmas fiquem mais precisas, utilizando da precisão alcançada pela experiência a fim de resolver problemas de características similares. [22]

O método usado para a criação desses modelos e a tomada de decisões é a indução, uma forma de inferência lógica que permite obter conclusões genéricas sobre um conjunto particular de exemplos. Ela generaliza um conceito específico, criando

uma hipótese que é aplicada na resolução de problemas similares. Como essas hipóteses são criadas por meio de exemplos, os dados fornecidos para a criação dessas hipóteses são de grande importância. Uma pequena quantidade de dados, dados incorretos ou com algum tipo de viés podem resultar em uma hipótese de pouco valor, que não é capaz de alcançar um nível de precisão satisfatório. [23]

Segundo Marsland *et al.* (2015), o processo de desenvolvimento e aplicação de um sistema de *Machine Learning* pode ser definido em 6 etapas:

- Coleta e preparação de dados: como mencionado anteriormente, os dados utilizados para o treinamento de um algoritmo são de grande importância para a criação de uma hipótese correta. Sendo assim, a coleta e a preparação de dados precisam de muita atenção e cuidado, buscando dados corretos, sem erros, em grande quantidade, sem viés e representativos do problema a ser resolvido;
- Seleção de características: definição das características dos exemplos coletados relevantes para o processo de indução do algoritmo. Nem sempre essas características são conhecidas;
- Escolha do algoritmo: existem diferentes técnicas e algoritmos para a aplicação do conceito de *machine learning*, como algoritmos supervisionados ou não supervisionados, a escolha do algoritmo correto para o problema é crucial;
- Escolha dos parâmetros e modelos: alguns algoritmos dependem de parâmetros manuais para seu funcionamento, requerendo experimentação para encontrar os valores adequados;
- Treinamento: dados o algoritmo, parâmetros e o conjunto de dados, o treinamento é a aplicação dos mesmos para obter um modelo ou hipótese para ser aplicados a novos dados e problemas;
- Avaliação: antes da aplicação do sistema, sua precisão deve ser avaliada utilizando dados com os quais não foi treinado.

2.5 Aprendizado Profundo (*Deep Learning*)

Deep learning é uma classe de algoritmos de *machine learning*, inspiradas no cérebro humano, compreendendo redes neurais que usam diversas camadas de processamento para progressivamente extrair propriedades de dados brutos. A Figura 1 mostra a hierarquia e relação do *deep learning* dentro da IA. Por exemplo, em processamento de imagens, camadas inferiores identificam bordas, enquanto camadas superiores identificam conceitos relevantes a humanos, como letras ou faces. Cada camada sucessiva usa a saída da camada anterior como entrada. [24]

Figura 1: Hierarquia das áreas de IA, *Machine Learning* e *Deep Learning*

(Adaptado de: www.medium.com/data-hackers "*Deep Learning*: do Conceito às Aplicações")

Uma rede neural, geralmente, contém apenas uma ou duas camadas escondidas, enquanto uma rede profunda pode contar com um número muito maior de camadas, dependendo de sua arquitetura e aplicação. A primeira camada de uma rede neural é conhecida como a camada de entrada. Cada nó nessa camada recebe um valor de entrada e, então, passa sua saída como a entrada para cada nó na seguinte camada. Não existem conexões entre nós da mesma camada, e a última camada produz a saída. As camadas intermediárias são conhecidas como *hidden layers* (camadas escondidas) [25], como mostra a Figura 2.

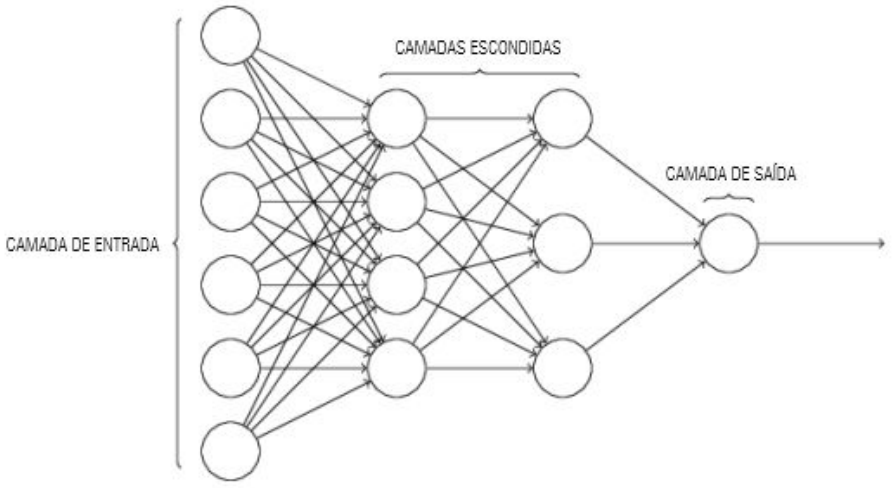

Figura 2: Rede neural com duas camadas escondidas.

(Adaptado de: Nielsen, 2015 [25])

3. Trabalhos Relacionados

Nessa seção apresentam-se alguns trabalhos relacionados ao aplicativo apresentado nesse artigo, apresentando as estratégias utilizadas e resultados alcançados pelos mesmos.

3.1 *Facial and expression recognition for the blind using computer vision*

O trabalho de Astler *et al.* [26] buscou criar uma solução completa para o reconhecimento de pessoas para deficientes visuais, desenvolvendo um protótipo consistindo de uma bengala com uma câmera embutida conectada a um laptop ou notebook carregados em uma mochila.

Os autores buscaram desenvolver mais funcionalidades que o simples reconhecimento e identificação de pessoas, sendo desenvolvido um sistema em tempo real, capaz de identificar também as expressões faciais de pessoas e de armazenar imagens de pessoas com as quais os usuários têm um maior número de interações, buscando melhorar a precisão do sistema.

O trabalho apresenta a metodologia de desenvolvimento, a seleção de algoritmos para o reconhecimento facial e de reconhecimento de expressões, o design físico do dispositivo e os resultados obtidos, além de sua avaliação por parte dos usuários que testaram o protótipo.

Devido ao requerimento de que o sistema fosse capaz de reconhecer pessoas e expressões em tempo real, a performance foi um ponto muito importante no projeto e por isso foram testados diversos algoritmos, buscando um que oferecesse a performance necessária para o funcionamento adequado do sistema. O resultado foi a utilização de duas ferramentas proprietárias, FaceSDK para o reconhecimento facial e FaceAPI para o reconhecimento de expressões. Com esse foco em performance, o sistema consegue realizar o reconhecimento a cada 30 frames, um número menor que o esperado pelos

autores, mas o suficiente para ser considerado tempo real. O sistema se comunica com o usuário por meio de comandos de voz através de um headset bluetooth com microfone. O usuário precisa perguntar para o sistema, usando o microfone, quem está sendo reconhecido naquele instante e recebe o feedback por meio de vozes ou de alertas sonoros.

Como resultado, os autores reconheceram o problema de a performance ser menor que a esperada. Também realçaram o resultado das entrevistas com usuários que indicaram que o tamanho do dispositivo e a necessidade de um notebook ou tablet para o processamento atrapalha muito sua utilização. Também indicaram que a discrição é um fator muito importante para os usuários e reconhecendo que o método de interação do usuário com o sistema, pela voz, impossibilita sua utilização em diversas situações.

3.2 *A face recognition application for people with visual impairments: Understanding use beyond the lab.*

O segundo trabalho a ser detalhado é um sistema desenvolvido por Zhao *et al.* [27] em 2018, pela Universidade de Cornell nos Estados Unidos, em conjunto com o Facebook.

O sistema foi desenvolvido em forma de um bot para o Facebook Messenger, o aplicativo de mensagens do Facebook. O bot usa as informações da conta do Facebook do usuário para localizar seus amigos e identificá-los em fotos tiradas pelo usuário. O bot só identifica uma pessoa se a mesma está na lista de amigos do usuário no Facebook.

Para realizar o reconhecimento, o usuário precisa ativar o bot, que vai utilizar a câmera do smartphone para fazer a detecção de faces e, com um comando do usuário (clicando na tela 2 vezes), ele tira uma foto que é enviada para um servidor remoto para o processamento e reconhecimento, retornando os resultados por meio de uma lista que é então lida para o usuário utilizando text-to-speech. O bot também é capaz de reconhecer expressões, posições (esquerda ou direita) e características faciais como cor do cabelo, presença de óculos ou barba.

O reconhecimento é feito por algoritmos proprietários do Facebook, realizado remotamente, dependendo de uma conexão com a Internet para a identificação.

Para a avaliação do bot, os autores forneceram o bot para 6 pessoas, que o utilizaram por uma semana e responderam perguntas sobre a utilização do mesmo. Os resultados alcançados indicaram uma baixa precisão, ligada principalmente às condições das fotos tiradas pelos usuários. Fotos em situações de baixa luminosidade, fora de foco ou distantes influenciaram muito na precisão do algoritmo e a utilização das fotos dos perfis públicos de usuários de Facebook também trouxe problemas, pois muitas vezes as pessoas são bem diferentes na realidade e usam fotos em situações bem diferentes das usuais em seus perfis.

3.3 *A wearable face recognition system for individuals with visual impairments*

Krishna *et al.* [28] são nomes que se destacam no desenvolvimento de tecnologias que utilizam de visão computacional para alcançar seus objetivos, entre elas, vários

dispositivos de tecnologia assistiva para deficientes visuais, como, por exemplo, cintos que utilizam vibração para alertar usuários sobre a proximidade de pessoas ao seu redor.

O trabalho aqui analisado buscou desenvolver um sistema de reconhecimento facial utilizando óculos com uma câmera embutida, em conjunto com um tablet para o processamento das imagens. O foco principal do trabalho é na comparação e no teste de algoritmos utilizados para o reconhecimento, avaliando o impacto da iluminação e do ângulo da imagem na precisão do reconhecimento e o tempo de resposta do algoritmo.

O desenvolvimento de algoritmos para reconhecimento facial avançou bastante desde a época do trabalho, mas os algoritmos analisados ainda são utilizados e funcionam como base para muitos algoritmos modernos, então ainda são relevantes na atualidade.

O sistema desenvolvido captura uma sequência de imagens, não em uma frequência suficiente para ser considerado um vídeo em tempo real, e realiza o processamento em um tablet. Após o reconhecimento de 5 imagens em sequência da mesma pessoa, o sistema informa ao usuário o resultado obtido, utilizando-se de *text-to-speech* por meio de um fone de ouvido.

Como esse trabalho foi mais voltado para a performance, não foram realizados testes ou entrevistas com usuários. Os autores destacam que os resultados adquiridos demonstram que o algoritmo de Principle Component Analysis (PCA), ou análise de componentes principais, mostrou o melhor resultado, considerando precisão, complexidade do algoritmo e performance.

3.4. Estudo Comparativo

O quadro 1 a seguir apresenta algumas características que permitem comparar os trabalhos apresentados com a solução proposta.

Características	**Astler, 2012**	**Zhao, 2018**	**Krishna, 2005**	**Solução Proposta**
Dispositivo Utilizado	Bengala com câmera	Smartphone	Óculos com câmera	Smartphone
Algoritmo utilizado	*FaceSDK*	API proprietário do Facebook	*PCA*	*SVM* (*Support Vector Machine*)
Forma de processamento	Tablet	Servidor Remoto	Tablet	Servidor Remoto
Banco de Dados	Próprio	Facebook	Próprio	Próprio
Forma de coleta de dados	Vídeo em tempo real	Imagem	Híbrido	Imagem

Reconhecimento de expressões faciais	Sim	Sim	Não	Não
Open Source	Não	Não	Sim	Sim
Atualização remota do sistema	Não	Limitada	Não	Sim

Quadro 1 – Estudo Comparativo dos

trabalhos apresentados e o aplicativo desenvolvido

(Fonte: do autores, 2021)

Um problema comum em todos os trabalhos estudados e apresentados no Quadro 1, é a baixa precisão dos algoritmos de reconhecimento facial em situações de baixa luminosidade, um problema que, até o momento, não foi possível de ser solucionado sem a utilização de ferramentas mais complexas dedicadas a esse fim, fora do escopo dos trabalhos que buscam uma solução de baixo custo a fim de proporcionar acesso a todos a essa nova ferramenta.

Os trabalhos selecionados para a comparação foram escolhidos baseados em seus diferentes métodos para resolução do problema, para serem comparados com o aplicativo apresentado neste artigo.

Astler *et al.* (2012) e Krishna *et al.* (2005) apostaram no desenvolvimento de uma ferramenta dedicada, adicionando complexidade e custo ao projeto, devido à necessidade de aquisição de câmeras e notebooks/tablets para o processamento. Isso, porém, possibilitou uma maior capacidade de customização de seu sistema e, por meio do uso de notebooks e tablets, uma capacidade de processamento maior que a oferecida por Zhao *et al.* (2018) e este artigo, possibilitando uma performance melhor e permitindo a avaliação de vários frames para cada reconhecimento, trazendo resultados muito mais precisos, especialmente no caso de Astler et al. (2012), por ter acesso a tecnologias mais modernas que Krishna *et al.* (2005). Vale destacar que Astler *et al.* (2012) só alcançaram uma performance melhor por utilizarem softwares proprietários de reconhecimento e detecção facial, adicionando um custo ainda maior ao projeto.

Outro sistema que usa tecnologia proprietária é o trabalho de Zhao *et al.* (2018), que realizaram um trabalho conjunto com funcionários do *Facebook,* tendo à API de reconhecimento facial proprietária da empresa. Da mesma forma que a solução proposta por esse artigo, a qual será descrita na próxima seção, Zhao *et al*. (2018) optaram por realizar o processamento remoto das imagens obtidas.

O aplicativo aqui apresentado busca se diferenciar dos trabalhos citados, trazendo uma solução que se utiliza totalmente de softwares *open source*, utilizando o *smartphone* como dispositivo para obtenção das imagens a serem processadas.

Uma questão muito importante hoje, realçada pelos resultados e *feedback* de

usuários nos trabalhos relacionados, relacionada ao reconhecimento facial é a privacidade e a possibilidade de abuso dessa tecnologia, por isso optamos por ter um banco de dados próprio, contendo somente servidores e professores da UFSM, sem o envolvimento de terceiros.

A solução apresentada neste artigo também se diferencia pela facilidade de modificação do sistema remotamente, sem alterações necessárias na parte *front end* do sistema. Por fazer o processamento remotamente e não depender de nenhuma ferramenta proprietária específica, como o trabalho de Zhao *et al.* (2018), ou dispositivos de *hardware* dedicados como ocorre nos trabalhos de Astler *et al.* (2012) e de Krishna *et al.* (2005), o aplicativo desenvolvido permite a alteração total do *back end* do sistema, incluindo algoritmo de reconhecimento e banco de dados, sem prejuízo ao usuário.

4. Solução Desenvolvida

A solução desenvolvida, apresentada neste artigo, compreende um sistema de reconhecimento facial, por meio de um aplicativo para *smartphones*.

Durante a construção dos protótipos do aplicativo, foi concluído que a performance dos *smartphones* atuais, principalmente os de uso em massa que não são estados da arte em performance como algoritmos governamentais ou proprietários, não é suficiente para a criação de um aplicativo de reconhecimento facial satisfatório em tempo real, devido a menor performance e acesso a dados se comparados a soluções de software aberto ou com *hardware* mais antigo. Não sendo possível a utilização em tempo real, a solução desenvolvida utiliza imagens para o reconhecimento facial, e devido a isso, foi decidido que o sistema desenvolvido fará o processamento remoto das imagens. Acredita-se que esse formato trará vantagens ao usuário, pois resulta em um aplicativo menor e de muito melhor performance, mais confiável e que funciona em qualquer aparelho da mesma forma, independente de capacidade de processamento.

O sistema se divide em duas partes: (1) o aplicativo, no *smartphone,* que é encarregado de obter as imagens a serem processadas e enviá-las ao (2) servidor remoto, que por sua vez é responsável pelo processamento das imagens e enviar a resposta resultante ao usuário, para ser apresentada ao mesmo pelo aplicativo, utilizando *text-to-speech*.

4.1 Aplicativo

O aplicativo para *smartphone* é a parte do sistema responsável pela aquisição das imagens e subsequente envio ao servidor remoto, Também é responsável por receber o resultado e informá-lo ao usuário.

Devido à inerente dificuldade de usuários com deficiência visual, a interface desenvolvida é a mais minimalista e simples possível, utilizando da interface já existente da câmera do dispositivo, contendo apenas um botão para tirar a foto. Os tradicionais botões para alternar entre a câmera frontal e traseira e a utilização do *flash* foram removidos para facilitar a utilização. Também foi incluída a opção de tirar a foto por meio dos botões de alteração de volume, levando em conta a possível dificuldade

em acertar o botão de captura de imagem para uma pessoa que não consegue enxergar o mesmo. Ambos os modos de captura fazem o telefone vibrar quando acionados. O *flash* está desativado por padrão para evitar constrangimentos e o foco para a foto é feito automaticamente pelo celular. A Figura 3 mostra a escolha de interface para o aplicativo, constando somente o botão para a foto.

Figura 3: Interface do aplicativo

(Fonte: do autores, 2021)

O aplicativo foi desenvolvido utilizando *React Native* [29] por meio do *framework* de desenvolvimento *Expo* [30]. O aplicativo funciona em plataformas *Android* com versão superior a 7.

No aplicativo o usuário é capaz somente de tirar fotos, as quais são automaticamente enviadas ao servidor remoto por meio da Internet e, quando é retornada a resposta, o mesmo informa ao usuário a resposta enviada pelo servidor, a partir do processamento do algoritmo de reconhecimento. A resposta do servidor é o nome das pessoas reconhecidas (se houver) e sua probabilidade em porcentagem. Também é informado, com distinção, se houve algum rosto detectado ou se o rosto detectado não foi reconhecido. Para ilustrar a sequência de interação entre usuário, servidor e aplicativo, foi desenvolvido o diagrama de sequência apresentado na Figura 4.

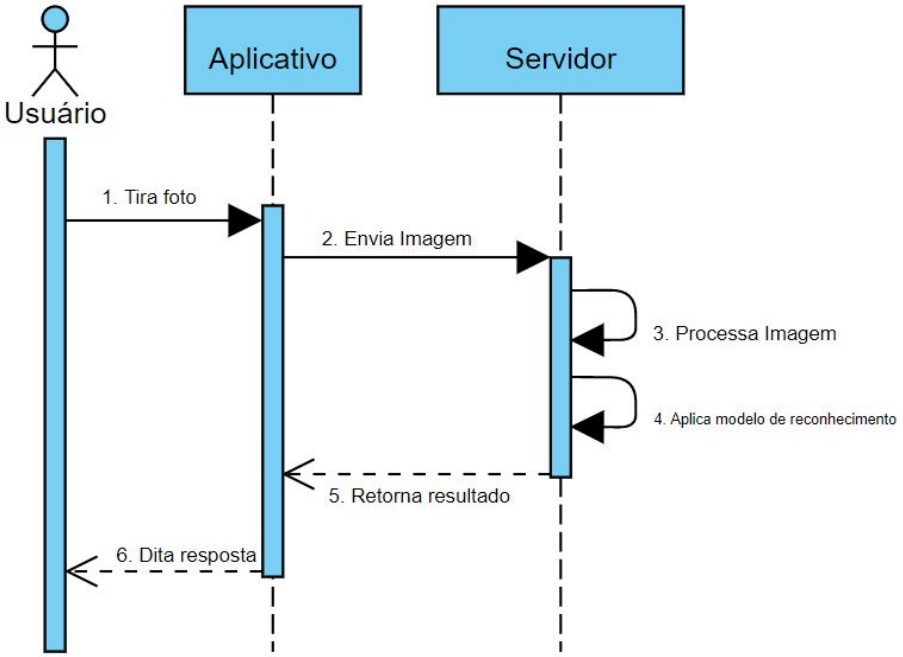

Figura 4: Diagrama de Sequência do Sistema

(Fonte: do autores, 2021)

4.1.2 Text-to-Speech

O resultado do processamento da imagem enviada pelo aplicativo ao servidor é transformado em voz por meio da API text-to-speech do *framework Expo*. Ele possui a capacidade de falar o resultado em português, porém a entonação de sobrenomes pode ficar confusa devido a limitações na tecnologia.

4.2 Servidor Remoto - *Webserver*

O servidor remoto foi desenvolvido utilizando-se a linguagem de programação *Python* e é o responsável por processar a imagem recebida pelo aplicativo e enviar para o mesmo uma resposta. Utiliza o *webserver Apache* com o padrão cliente - servidor por meio do protocolo HTTP *(Hypertext Transfer Protocol)* para fazer a comunicação entre o aplicativo e o servidor.

O servidor é hospedado na plataforma *Google App Engine,* que oferece a escalabilidade e confiança de uma solução em nuvem gerenciada pela *Google*.

Para possibilitar o reconhecimento facial, o sistema se utiliza de 4 algoritmos independentes, suportados pela biblioteca *open source OpenCV*, que serão explicados nas subseções seguintes. São eles: o algoritmo quantificador, responsável por criar os vetores de características das imagens de pessoas do banco de dados; o algoritmo para treinar o modelo de reconhecimento com esses vetores; o algoritmo de reconhecimento que aplica o modelo em uma imagem e o quarto algoritmo é o detector de faces, que é utilizado pelo algoritmo quantificador e o algoritmo de reconhecimento. Para melhor demonstrar como o sistema está organizado, foi desenvolvido o fluxograma apresentado

na Figura 5.

Todo o processo do sistema, do início com a quantificação das faces até a comunicação com o aplicativo *mobile* do usuário é administrado e gerenciado pela aplicação desenvolvida e hospedada no servidor.

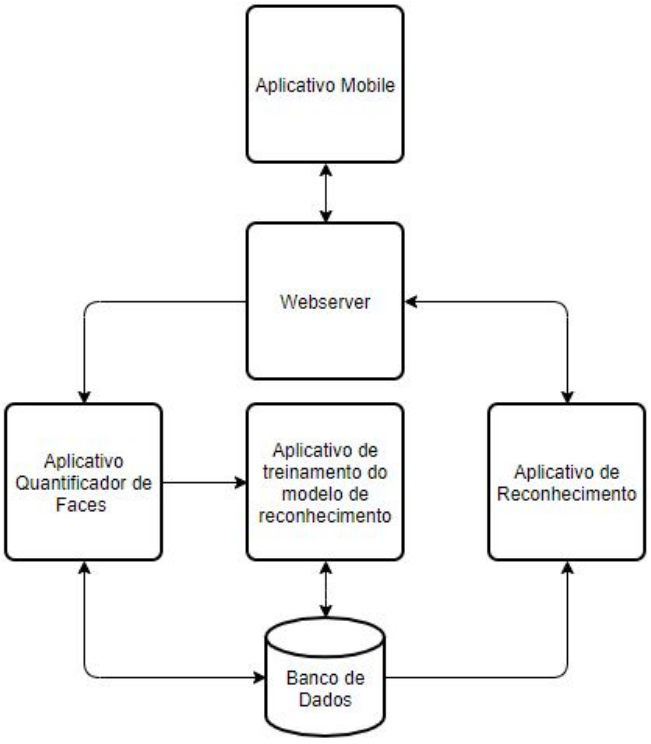

Figura 5: Fluxograma do funcionamento lógico do sistema

(Fonte: do autores, 2021)

O sistema desenvolvido emprega apenas algoritmos *open source*, utilizando os algoritmos e modelos disponíveis pelo *OpenCV* [31], *scikit-learn* [32] e *OpenFace* [33].

A sequência exibida na Figura 5 não precisa ser repetida toda vez que seja necessário realizar o reconhecimento já que, uma vez quantificadas as imagens do banco de dados e treinado o algoritmo, o algoritmo de reconhecimento só aplica os mesmos para efetivamente realizar o reconhecimento. O processo foi particionado dessa forma para possibilitar a modularidade desejada pela solução desenvolvida, permitindo a alteração de módulos e algoritmos do sistema sem a necessidade de reconfigurar os algoritmos não modificados.

As alterações de módulos (algoritmos) devem respeitar a lógica de *input* e *output* do sistema, peculiaridades de cada algoritmo, como, por exemplo, a necessidade de tratamento de imagem, conversão de formatos, formatação de *output*, entre outros, devem ser tratadas dentro do próprio módulo, evitando a necessidade de alteração no código do servidor ou dos outros algoritmos.

4.2.1 OpenCV

OpenCV, ou *Open Source Computer Vision Library*, é uma biblioteca para visão computacional e *machine learning* de código aberto. A biblioteca possui mais de 2500 algoritmos, clássicos e estado da arte, usados para detectar faces, identificar objetos, classificar ações, extrair modelos 3D de objetos, etc. Declara ter 47 mil membros na comunidade e mais de 18 milhões de downloads.

A biblioteca está disponível para *C++, Python, Java* e *MATLAB*, suportando os sistemas operacionais *Windows, Linux, Android* e *MacOS.*

A escolha do *OpenCV* se deu por conta de ser uma solução gratuita, de ampla documentação e utilizada em diversas aplicações com funcionalidades diferentes, fornecendo um alto nível de performance e confiabilidade, de fácil implementação e manutenção.

4.2.2 Detector de Faces

O algoritmo detector de faces utiliza o módulo de *deep learning* do *OpenCV*, realizando a detecção de faces por meio do algoritmo *Caffe* [34]. *Caffe* é um *framework* para *deep learning* desenvolvido inicialmente pela Universidade da Califórnia Berkeley e hoje continua como um projeto *open source* mantido no *GitHub*.

Caffe usa redes neurais e *deep learning* para encontrar as características para qual foi treinado em uma imagem ou vídeo. Para isso, é preciso enviar para o algoritmo dois arquivos, um arquivo *.protxt* que contém as camadas e um arquivo *.caffemodel* que contém o peso para as camadas. Ambos os arquivos são fornecidos pelo próprio OpenCV.

4.2.3 Quantificador de Faces

Utilizando o banco de dados com imagens das pessoas a serem reconhecidas, o algoritmo quantificador itera pelas imagens de cada pessoa e cria um vetor único, quantitativo das características dessas pessoas.

O algoritmo utilizado aqui é o algoritmo *FaceNet*, do projeto *OpenFace*, que usa uma rede neural convolucional (CNN) para representar a face em uma hiperesfera unitária de 128 dimensões. A representação é uma representação genérica para o rosto de qualquer pessoa. Ao contrário de outras representações de rosto, essa representação tem a propriedade de que uma distância maior entre duas representações de face significa que os rostos provavelmente não são da mesma pessoa. Essa propriedade facilita as tarefas de agrupamento, detecção de similaridade e classificação. O processo de criação de uma representação é apresentado na Figura 6. A detecção é feita pelo algoritmo *Caffe*, a transformação e tratamento da imagem pelo *OpenCV*, o algoritmo de quantificação usa uma rede neural profunda para criar a representação na hiperesfera de 128 dimensões. As representações resultantes são usadas no treinamento da SVM (*Support Vector Machine*), no próximo passo do sistema. A Figura 6 mostra a sequência para criação da representação quantificada de uma face, seguido da parte do código correspondente com a Figura 7.

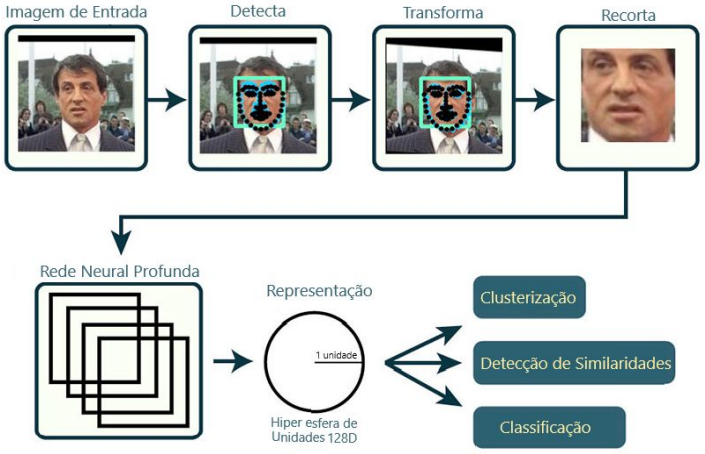

Figura 6: representação da sequência lógica para a criação de um *embedding*

(Adaptado de: https://cmusatyalab.github.io/openface/**)**

```
# loop sobre as pastas
for (i, imagePath) in enumerate(imagePaths):
    # extrai o nome da pessoa atraves do caminho da pasta
    print("[INFO] processando imagem {}/{}".format(i + 1,
        len(imagePaths)))
    name = imagePath.split(os.path.sep)[-2]

    # redimensiona para uma largura de 600 e pega as dimensoes
    image = cv2.imread(imagePath)
    image = imutils.resize(image, width=600)
    (h, w) = image.shape[:2]

    # constroi a blob
    imageBlob = cv2.dnn.blobFromImage(
        cv2.resize(image, (300, 300)), 1.0, (300, 300),
        (104.0, 177.0, 123.0), swapRB=False, crop=False)

    # chama o detector do opencv para procurar faces na imagem
    detector.setInput(imageBlob)
    detections = detector.forward()

    # se encontrou alguma
    if len(detections) > 0:
        # assumindo que cada imagem so tem uma face,
        # se tiver mais ele vai pegar a com a maior probabilidade
        i = np.argmax(detections[0, 0, :, 2])
        confidence = detections[0, 0, i, 2]

        # filtra a deteccao com base no arg de confianca minima informada
        if confidence > args["confidence"]:
            # busca a posicao da face na imagem
            box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
            (startX, startY, endX, endY) = box.astype("int")

            # extrai a ROI (regiao de interesse) da face e o seu tamanho
            face = image[startY:endY, startX:endX]
            (fH, fW) = face.shape[:2]

            # construi a blob para a ROI e a envia para o modelo
            # criar a quantificacao 128-d do rosto
            faceBlob = cv2.dnn.blobFromImage(face, 1.0 / 255,
                (96, 96), (0, 0, 0), swapRB=True, crop=False)
            embedder.setInput(faceBlob)
            vec = embedder.forward()

            # adiciona o embedding dessa imagem para a lista de embeddings da pessoa
            # para posterior utilizacao no algoritmo de classificacao
            knownNames.append(name)
            knownEmbeddings.append(vec.flatten())

            # aumenta o total de imagens processadas
            total += 1
```

Figura 7: Código principal de extração dos embeddings

(Fonte: do autores, 2021)

Para criar a representação (*embedding*), o algoritmo usa da função *triplet loss*, que utiliza 3 imagens: a imagem a ser utilizada para criar o *embedding* de uma pessoa A (âncora); uma imagem diferente também da pessoa A (positiva) e uma imagem de uma pessoa diferente B (negativa), ilustrado na Figura 8. O algoritmo calcula o *embedding* para cada face e modifica os pesos da rede utilizando da função *triplet loss*, tornando a imagem âncora e positiva mais próximas e ao mesmo tempo afastando a imagem negativa. Dessa maneira, o algoritmo é capaz de quantificar faces e retornar *embeddings* altamente discriminatórios e robustos para o reconhecimento facial.

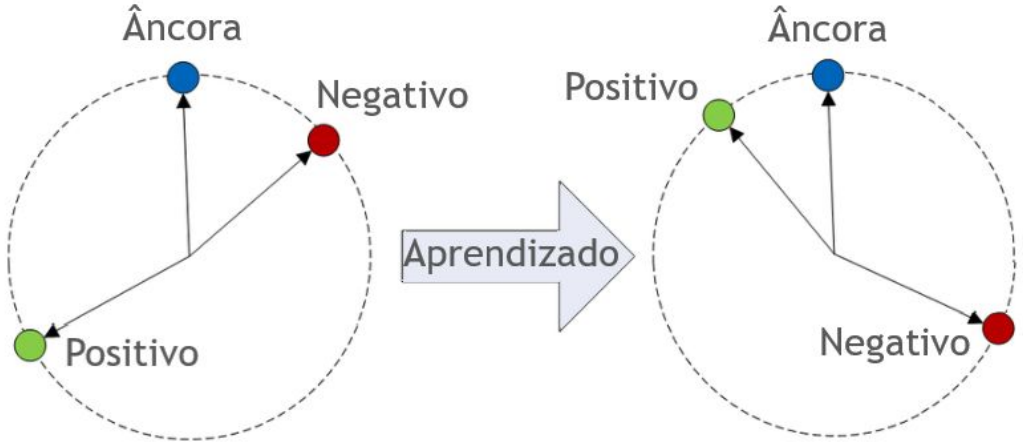

Figura 8: representação gráfica da função *Triplet Loss*

(**Adaptado de:** https://mc.ai/we-know-who-you-are-before-you-finish-saying-hello/)

4.2.4 *Support Vector Machine* (SVM) – Algoritmo de Treinamento

Uma SVM é um classificador discriminativo formalmente definido por um hiperplano de separação. Em outras palavras, informados dados de treinamento rotulados (aprendizado supervisionado), o algoritmo gera um hiperplano ideal que categoriza novos exemplos [35].

A Figura 9 mostra um problema simples de classificação de objetos de 2 classes, e demonstra o comportamento de uma SVM. A imagem da esquerda mostra possíveis soluções válidas usando hiperplanos, a imagem da direita mostra a solução encontrada por uma SVM, que maximiza a margem para cada classe e cria um hiperplano de separação entre elas. Vale lembrar que o exemplo utilizando planos cartesianos e 2 dimensões é extremamente simplificado, o modelo real utiliza vetores e hiperplanos em um espaço tridimensional.

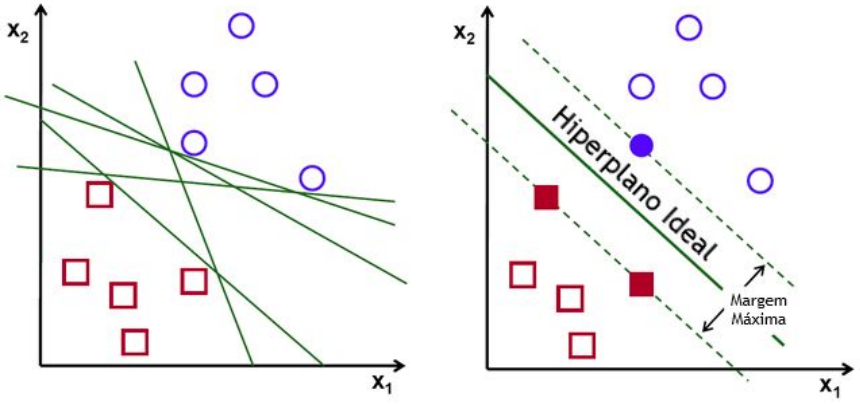

Figura 9: Exemplo de classificação utilizando SVM

(Adaptado de: https://docs.opencv.org/3.4/d1/d73/tutorial_introduction_to_svm.html)

O algoritmo de treinamento treina uma SVM utilizando os *embeddings* extraídos pelo algoritmo quantificador, criando um modelo para ser utilizado no algoritmo de reconhecimento.

4.2.5 Reconhecimento

O algoritmo de reconhecimento é o responsável por aplicar os resultados e algoritmos anteriores para realizar o reconhecimento. Primeiramente ele utiliza o algoritmo de detecção para encontrar uma face na imagem, que é filtrada e transformada para poder ser usada pelo aplicativo seguinte. O algoritmo de quantificação então obtém um *embedding*, que por sua vez, utilizando a SVM treinada pelo algoritmo anterior, é classificado baseado em sua probabilidade relacionada aos *embedding*s obtidos a partir do banco de dados. O resultado é então enviado para o aplicativo *mobile*, que o informa ao usuário do mesmo.

O tempo de processamento e reconhecimento da imagem fica entre 600 e 800 ms. Por meio dos testes realizados, verificou-se que não houve alteração significativa no tempo de resposta entre imagens com reconhecimento positivo, negativo ou com mais pessoas.

5. Validação do aplicativo

Devido à suspensão das atividades presenciais da universidade e da importância do distanciamento social postos pela crise sanitária relativa à pandemia de COVID-19, testes com usuários finais em situações de uso real nos campus da UFSM foram impossibilitados, o que não permitiu melhorias e ajustes baseados no *feedback* de usuários em situações não ideais e, reconhecendo que o funcionamento correto do aplicativo utilizando câmeras é dependente de situações ótimas de fatores como alta luminosidade, foco da câmera, entre outras características, não conseguimos medir os resultados obtidos em situações reais de utilização por usuários portadores de deficiências visuais.

A validação do aplicativo foi realizada com um banco de dados totalizando 13355 imagens e 5761 pessoas. As imagens utilizadas foram obtidas através do *dataset*

Labeled Faces in the Wild [36], desenvolvido pela Universidade de Massachusetts, nos Estados Unidos. Além do dataset, foram utilizadas imagens obtidas pelo autor.

Com essa quantidade de imagens, o algoritmo quantificador demorou aproximadamente 41 minutos para criar os *embeddings* e 11 minutos para o treinamento do modelo de reconhecimento. Lembrando que a extração dos *enbeddings* e o treinamento não são realizados toda vez que é realizado algum reconhecimento, eles são executados somente quando necessários, devido alguma alteração no banco de dados, mudanças nos parâmetros de treinamento ou decisão do administrador.

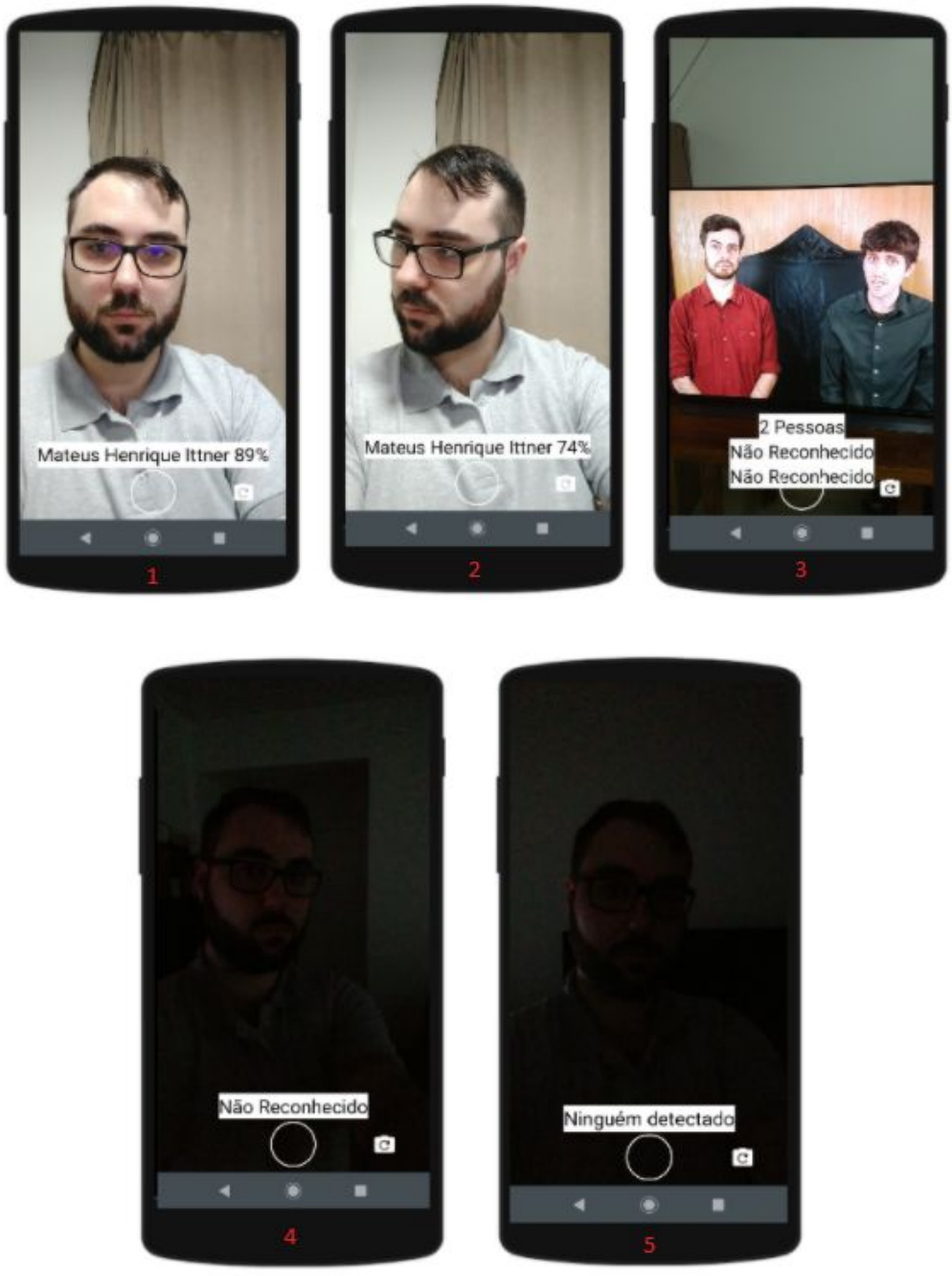

Figura 10: Utilização do Aplicativo

(Fonte: do autores, 2021)

A Figura 10 apresenta alguns resultados obtidos com a utilização do aplicativo. As fotos 1, 2 e 3 foram tiradas em situações normais de utilização e as fotos 4 e 5 em situações de baixa iluminação. Para possibilitar a demonstração dos resultados obtidos pelo aplicativo no artigo, o resultado ditado para o usuário foi convertido em texto e para os testes também foi adicionada a possibilidade de alteração entre câmera frontal e traseira, funcionalidades não presentes no aplicativo final. As fotos 1 e 2 demonstram as capacidades do aplicativo em situação ótima de utilização: fotos próximas da pessoa a ser reconhecida e com alta luminosidade; os resultados foram satisfatórios, vale ressaltar que a maior quantidade e qualidade de fotos utilizadas para o treinamento aumentam a confiança do algoritmo, assim como a configuração de confiança mínima necessária para o treinamento e posteriormente para o reconhecimento. A foto 3 foi obtida por meio da captura da foto de 2 pessoas em um aparelho de TV, apenas para demonstrar como o algoritmo responde se foram detectadas mais de uma pessoa. Ambas as pessoas da foto não estão no banco de dados e, portanto, não foram reconhecidas. As fotos 4 e 5 demonstram situações em que o algoritmo não consegue realizar o reconhecimento; situações de baixa luminosidade afetam muito a qualidade da imagem, causando o não reconhecimento e possivelmente até a completa não detecção de uma face na foto.

Figura 11: Testes de Validação

(Fonte: do autores, 2021)

A Figura 11 demonstra resultados obtidos para o reconhecimento de outras pessoas no banco de dados. Devido a impossibilidade de fotos em tempo real dessas pessoas, as imagens foram processadas diretamente pelo algoritmo de reconhecimento, sem o envio por parte do aplicativo. Devido a limitação do *dataset,* que tem poucas imagens por pessoas, o resultado para algumas não foi satisfatório e em geral as probabilidades encontradas para as pessoas reconhecidas não foram muito altas. No caso da segunda foto o ex-presidente Bill Clinton não foi reconhecido. Diminuindo a confiança necessária para uma resposta, verificamos que ele foi reconhecido como outro ex-presidente, George W. Bush, com uma probabilidade de 2.74%. Em uma situação de utilização real, o banco de dados vai conter mais imagens da pessoa a ser reconhecida, não somente uma, como no dataset utilizado para demonstrar o resultado, aumentando a qualidade do reconhecimento, pois os algoritmos utilizados para o treinamento dependem tanto da quantidade total de pessoas, quanto da quantidade de imagens por pessoa.

A versão final do aplicativo no *Google App Engine* oferece boa performance e

tempo de resposta adequado, mas idealmente o sistema seria implantado em servidores próprios da UFSM, melhorando drasticamente o tempo de resposta, reduzindo custos, oferecendo maior controle sobre o ambiente. Entretanto, ainda teriam que ser analisadas a performance e confiabilidade dos servidores da UFSM.

O tempo de resposta do aplicativo, do momento da foto até a leitura do resultado, fica, em média, de 2 a 6 segundos. A variação de tempo é causada pela qualidade da conexão de Internet do usuário. Testes realizados no computador pessoal dos pesquisadores, com conexão de *upload* de 100 *Mbps* (*megabits* por segundo), resultaram em um atraso de resposta em torno de 1800 ms (1.8s); testes realizados em um *smartphone* com conexão *wi-fi* 5GHz de 80 *Mbps* resultaram em média 3.3s (3300ms) e uma conexão de dados móveis HSPA+ (H+) de 0.4 *Mbps* demorou em média 5800 ms (5.8s). Como mencionado anteriormente, o tempo de processamento e reconhecimento da imagem fica entre 600 e 800 ms (0.6 - 0.8s), demonstrando que a maior parte do atraso se dá no envio da imagem ao servidor, que é muito dependente da qualidade da conexão do usuário. Em conexões mais lentas chega a alcançar 80% do tempo de atraso total.

6. Considerações Finais

As principais dificuldades encontradas foram relacionadas à definição final da arquitetura do sistema. Inicialmente foi desenvolvido um protótipo para *notebooks* utilizando a linguagem de programação *Python*, mas não foi possível fazer a portabilidade do mesmo para um ambiente *mobile*, devido à arquitetura do *OpenCV Python* criar conflitos com os *frameworks* de desenvolvimento *mobile* em *Python*. O segundo protótipo foi desenvolvido inteiramente *mobile*, utilizando *React Native* e *OpenCV* para *Android* (por meio da linguagem de programação Java), mas a performance era muito ruim em tempo real, impossibilitando o uso de vídeos em tempo real. Já que o reconhecimento em tempo real não seria possível, foi decidido em realizar o processamento remotamente, fazendo a modularização do sistema e permitindo a alteração de algoritmos de forma muito simples, sem nenhuma complexidade extra trazida em ambientes *mobiles*, como performance, memória, entre outros, permitindo algoritmos mais eficientes que trariam atrasos com o pouco poder de processamento oferecido por *smartphones*. O único custo de fazer o processamento remoto é a necessidade de uma conexão com a Internet, a qual não vemos com muita preocupação pois a finalidade do aplicativo é a utilização dentro da UFSM, que fornece *wi-fi* gratuito. Outro problema é o atraso causado pela necessidade de envio para um servidor remoto e subsequente retorno do mesmo, mas acreditamos que esse atraso é ainda menor do que se feito o reconhecimento inteiramente no *smartphone*.

Levando em consideração os trabalhos relacionados, acreditamos que as principais vantagens do aplicativo sejam a facilidade de utilização devido à simplicidade e a não necessidade de conhecimento prévio sobre a tecnologia ou treinamento para sua utilização; um grande alcance de usuários pois não precisa de nenhum acessório específico além de um *smartphone* e o fato de ser totalmente gratuito, estabilidade do sistema pois todo processamento é realizado em um servidor remoto e a facilidade de manutenção e alteração do código.

Como desvantagem, podemos citar um menor grau de precisão se comparados a

algoritmos proprietários (como o do *Facebook*), apesar de que, como não foi possível realizar testes extensivos, talvez nem se encontre essa diferença. Como todo o processamento é realizado por um servidor remoto, a necessidade de conexão à Internet para a utilização do aplicativo também é uma grande desvantagem, mas, como mencionado na seção anterior, o aplicativo foi desenvolvido com a utilização focada na UFSM, que oferece *wi-fi* gratuita, na teoria não causa muito impacto ao usuário.

A interface atual e as informações faladas por meio do *text-to-speech* foram desenvolvidas levando em consideração a avaliação de trabalhos relacionados, então acreditamos que seja uma vantagem, porém como não conseguimos testar o aplicativo com usuários reais, não sabemos como um usuário com deficiência visual vai se adaptar à utilização do aplicativo.

O tempo de resposta do aplicativo traz uma preocupação quanto à demora em casos de má qualidade de conexão com a Internet, possíveis pré-processamentos da imagem, como a realização da detecção facial no próprio *smartphone*, trariam uma diminuição no tempo de resposta, porém podem sacrificar parte da modularidade do sistema pois a foto já viria pré-processada e ainda é possível que esse pré-processamento demoraria mais tempo do que o envio da imagem original, não trazendo nenhum benefício quanto a demora e sacrificando vantagens trazidas pela realização do processamento remoto, como a estabilidade e compatibilidade do aplicativo com diversos aparelhos.

Possíveis trabalhos futuros são a implantação e testes de novos algoritmos para uma possível substituição de módulos do sistema, buscando melhor precisão, performance ou mais funcionalidades; criação de algoritmos de alinhamento e tratamento das imagens enviadas para reconhecimento, buscando aumento da precisão do mesmo por meio de melhorias na qualidade de imagens e possíveis correções em luminosidade, etc; algoritmos de reconhecimento de expressões faciais, características de pessoas e ambientes, buscando oferecer uma forma de transmitir a situação das pessoas no ambiente e as características dos mesmos. Como a crise do Coronavírus impossibilitou testes em situações de utilização real, também se indica como um trabalho futuro a avaliação e atualização do aplicativo com base nos *feedbacks* obtidos.

A utilização de somente um botão como interface também foi motivada pensando na utilização de outros dispositivos. Como o servidor só recebe uma imagem e retorna um resultado, qualquer *hardware* capaz de receber e transmitir dados pode ser adaptado para a utilização do sistema, possibilitando a utilização de acessórios diferentes como os utilizados em alguns trabalhos relacionados, como óculos adaptados, bengalas, etc.

Referências

[1] ASTLER, Douglas *et al*. Increased accessibility to nonverbal communication through facial and expression recognition technologies for blind/visually impaired subjects. In: The proceedings of the 13th international ACM SIGACCESS conference on Computers and accessibility. 2011. p. 259-260.

[2] BOTTOU, Léon et al. Counterfactual reasoning and learning systems: The example of computational advertising. The Journal of Machine Learning Research, v. 14, n. 1, p. 3207-3260, 2013.

[3] Deficiência visual / Marta Gil (org.). – Brasília : MEC. Secretaria de Educação a Distância, 2000. 80 p. : il. - (Cadernos da TV Escola. 1. ISSN 1518-4692)

[4] World Health Organization. World Report on Vision. Outubro 2019

[5] Instituto Brasileiro de Geografia e Estatística. Pesquisa Nacional de Saúde 2013. Julho 2016

[6] BRASIL. Ministério da Educação e do Desporto, Secretaria da Educação Especial. Subsídios para a formulação da política nacional de educação especial. Brasília, 1993.

[7] CHIA, Ee-Munn et al. Impact of bilateral visual impairment on health-related quality of life: the Blue Mountains Eye Study. Investigative ophthalmology & visual science, v. 45, n. 1, p. 71-76, 2004.

[8] LANGELAAN, Maaike et al. Impact of visual impairment on quality of life: a comparison with quality of life in the general population and with other chronic conditions. Ophthalmic epidemiology, v. 14, n. 3, p. 119-126, 2007.

[9] PIERRARD, Jean-Sébastien; VETTER, Thomas. Skin detail analysis for face recognition. In: 2007 IEEE conference on computer vision and pattern recognition. IEEE, 2007. p. 1-8.

[10] SOCOLINSKY, Diego A.; SELINGER, Andrea; NEUHEISEL, Joshua D. Face recognition with visible and thermal infrared imagery. Computer vision and image understanding, v. 91, n. 1-2, p. 72-114, 2003.

[11] BLANZ, Volker; VETTER, Thomas. Face recognition based on fitting a 3d morphable model. IEEE Transactions on pattern analysis and machine intelligence, v. 25, n. 9, p. 1063-1074, 2003.

[12] BONSOR, Kevin; JOHNSON, Ryan. How facial recognition systems work. HowStuffWorks. Com Np, v. 4, 2001.

[13] NAKER, Sharon; GREENBAUM, Dov. Now you see me: Now you still do: Facial recognition technology and the growing lack of privacy. BUJ Sci. & Tech. L., v. 23, p. 88, 2017.

[14] MCCOY, Susan. O'Big Brother where art thou?: The constitutional use of facial-recognition technology. Order, v. 20, 2002.

[15] SCHERER, Marcia J. Outcomes of assistive technology use on quality of life. Disability and rehabilitation, v. 18, n. 9, p. 439-448, 1996.

[16] RUSSELL, J. Neil et al. Trends and differential use of assistive technology devices: United States, 1994. US Department of Health and Human Services, Centers for Disease Control and Prevention, National Center for Health Statistics, 1997.

[17] EDYBURN, Dave L. Rethinking assistive technology. Special Education Technology Practice, v. 5, n. 4, p. 16-23, 2004.

[18] PARETTE, Phil; SCHERER, Marcia. Assistive technology use and stigma. Education and Training in Developmental Disabilities, p. 217-226, 2004.

[19] TEIXEIRA, João. O que é inteligência artificial. E-Galáxia, 2019.

[20] GOMES, D. dos S. Inteligência Artificial: conceitos e aplicações. Olhar Científico. v1, n. 2, p. 234-246, 2010.

[21] MITCHELL, Thomas M. et al. Machine learning. 1997.

[22] MARSLAND, Stephen. Machine learning: an algorithmic perspective. CRC press, 2015.

[23] MONARD, Maria Carolina; BARANAUSKAS, José Augusto. Conceitos sobre aprendizado de máquina. Sistemas inteligentes-Fundamentos e aplicações, v. 1, n. 1, p. 32, 2003.

[24] DENG, Li et al. Deep learning: methods and applications. Foundations and Trends® in Signal Processing, v. 7, n. 3–4, p. 197-387, 2014.

[25] NIELSEN, Michael A. Neural networks and deep learning. San Francisco, CA, USA: Determination press, 2015.

[26] ASTLER, Douglas et al. Facial and expression recognition for the blind using computer vision. 2012. Tese de Doutorado.

[27] ZHAO, Yuhang et al. A face recognition application for people with visual impairments: Understanding use beyond the lab. In: Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems. 2018. p. 1-14.

[28] KRISHNA, Sreekar et al. A wearable face recognition system for individuals with visual impairments. In: Proceedings of the 7th international ACM SIGACCESS conference on Computers and accessibility. 2005. p. 106-113.

[29] FACEBOOK. React Native. Versão 0.62, 26 de março de 2020. Disponível em: https://reactnative.dev/versions. Acesso em: 22 de junho de 2020.

[30] 650 INDUSTRIES INC. Expo. Versão SDK 37, 31 de março de 2020. Disponível em: https://expo.io/. Acesso em: 22 de junho de 2020.

[31] BRADSKI, Gary; KAEHLER, Adrian. OpenCV. Dr. Dobb's journal of software tools, v. 3, 2000.

[32] PEDREGOSA, Fabian et al. Scikit-learn: Machine learning in Python. the Journal of machine Learning research, v. 12, p. 2825-2830, 2011.

[33] AMOS, Brandon et al. OpenFace: A general-purpose face recognition library with mobile applications. CMU School of Computer Science, v. 6, n. 2, 2016.

[34] JIA, Yangqing et al. Caffe: Convolutional architecture for fast feature embedding. In: Proceedings of the 22nd ACM international conference on Multimedia. 2014. p. 675-678.

[35] HEARST, Marti A.. et al. Support vector machines. IEEE Intelligent Systems and their applications, v. 13, n. 4, p. 18-28, 1998.

[36] HUANG, G. et al. Labeled faces in the wild: A database for studying face recognition in unconstrained environments University of Massachusetts, Amherst. MA, Tech. Rep. 07-49, 2007.