PONTIFÍCIA UNIVERSIDADE CATÓLICA DE GOIÁS
ESCOLA POLITÉCNICA E DE ARTES
GRADUAÇÃO EM ENGENHARIA DE COMPUTAÇÃO

RECONHECIMENTO FACIAL PARA SEGURANÇA RESIDENCIAL: ABORDAGENS COM REDES NEURAIS CONVOLUCIONAIS

AUGUSTO VAZ RODRIGUES

GOIÂNIA
2024

AUGUSTO VAZ RODRIGUES

RECONHECIMENTO FACIAL PARA SEGURANÇA RESIDENCIAL: ABORDAGENS COM REDES NEURAIS CONVOLUCIONAIS

Trabalho de Conclusão de Curso apresentado à Escola Politécnica e de Artes, da Pontifícia Universidade Católica de Goiás, como parte dos requisitos para a obtenção do título de Bacharel em Engenharia de Computação.

Orientador: Prof. Me. Rafael Leal Martins.

GOIÂNIA
2024

AUGUSTO VAZ RODRIGUES

RECONHECIMENTO FACIAL PARA SEGURANÇA RESIDENCIAL: ABORDAGENS COM REDES NEURAIS CONVOLUCIONAIS

Trabalho de Conclusão de Curso aprovado em sua forma final pela Escola Politécnica e de Artes, da Pontifícia Universidade Católica de Goiás, para obtenção do título de Bacharel em Engenharia de Computação, em _____/_____/_____.

_____
Orientador: Prof. Me. Rafael Leal Martins.

_____
Prof.ª Me. Ana Flavia Marinho De Lima Garrote.

_____
Prof. Me. Gustavo Siqueira Vinhal.

GOIÂNIA
2024

AGRADECIMENTOS

Gostaria de expressar meus sinceros agradecimentos a todas as pessoas que contribuíram para a realização deste trabalho. Em primeiro lugar, agradeço à minha família pelo amor, apoio incondicional e compreensão durante os momentos de dedicação ao TCC. Agradeço também ao meu orientador/professor Rafael, pela orientação e insights valiosos ao longo deste processo. Aos meus amigos e colegas de curso, que compartilharam seu encorajamento, meu profundo agradecimento. Por fim, expresso minha gratidão a todas as fontes de conhecimento, autores e pesquisadores cujo trabalho e contribuições foram essenciais para a construção deste estudo. Este trabalho não teria sido possível sem o apoio e colaboração de todos vocês. Muito obrigado.

"Não podemos permitir que o medo nos impeça de avançar, não podemos deixar que o medo impeça que a tecnologia seja usada para o bem."

Edward Snowden

RESUMO

O presente estudo aborda o uso de Redes Neurais Convolucionais (CNNs) no contexto do reconhecimento facial aplicado à segurança residencial. A pesquisa propõe a implementação e análise de sistemas de reconhecimento facial baseados em CNNs, visando otimizar a identificação e autenticação de residentes e não residentes em ambientes residenciais. Utilizando técnicas de visão computacional, o trabalho explora como as CNNs podem ser empregadas para processar imagens de segurança, reconhecendo faces com maior rapidez e precisão. Além disso, o estudo considera a integração dessas redes neurais como um meio de aprimorar a eficiência dos sistemas de segurança residencial. Por meio de experimentação e análise de dados obtidos em um ambiente controlado, são investigados os benefícios e desafios do uso das CNNs nesse contexto, destacando a relevância e as possibilidades desses sistemas para aprimorar a segurança residencial.

Palavras-chave: Reconhecimento Facial. Redes Neurais Convolucionais. Visão Computacional. Inteligência Artificial. Segurança.

ABSTRACT

The present study addresses the use of Convolutional Neural Networks (CNNs) in the context of facial recognition applied to residential security. The research proposes the implementation and analysis of CNN-based facial recognition systems, aiming to optimize the identification and authentication of residents and non-residents in residential environments. Using computer vision techniques, the work explores how CNNs can be used to process security images, recognizing faces more quickly and accurately. Additionally, the study considers the integration of these neural networks as a means of improving the efficiency of residential security systems. Through experimentation and data analysis obtained in a controlled environment, the benefits, and challenges of using CNNs in this context are investigated, highlighting the relevance and possibilities of these systems to improve residential security.

Key words: Facial recognition. Convolutional Neural Networks. Computer vision. Artificial intelligence. Security.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

ANN	*Artificial Neural Networks*
API	*Application Programming Interface*
AWS	*Amazon Web Services*
CNN	*Convolutional Neural Network*
Colab	*Google Colaboratory*
DL	*Deep Learning*
FC	*Fully Connected*
IA	Inteligência Artificial
ILSVRC	*ImageNet Large Scale Visual Recognition Challenge*
IMB	*International Business Machines Corporation*
LR	*Learning Rate*
ML	*Machine Learning*
PIN	*Personal Identification Number*
SGD	*Stochastic Gradient Descent*
USPS	*United States Postal Service*
VSCode	*Visual Studio Code*

1. INTRODUÇÃO

A visão computacional é uma tecnologia que permite que máquinas identifiquem e descrevam imagens automaticamente, com precisão e eficiência. Atualmente, sistemas de computador têm amplo acesso a uma vasta quantidade de imagens e dados de vídeo provenientes de *smartphones*, câmeras de trânsito, sistemas de segurança e outros dispositivos. A visão computacional, impulsionada pela inteligência artificial e *machine learning* (IA/ML), processa esses dados com precisão, viabilizando a identificação de objetos, reconhecimento facial, classificação, recomendação, monitoramento e detecção (AWS, 2023).

O reconhecimento facial, uma aplicação da visão computacional, desempenha um papel fundamental na análise e identificação de indivíduos. Essa tecnologia serve como base para a identificação, agrupamento, classificação e verificação de rostos. Sua relevância se estende a áreas como segurança cibernética, detecção de fraudes, controle de fronteiras e serviços bancários, onde a identificação precisa e confirmação de identidades são essenciais.

Um *software* de análise facial opera ao identificar ou confirmar a identidade de uma pessoa através do rosto. Essa análise envolve a identificação e medição de componentes faciais presentes em uma imagem. O reconhecimento facial é capaz de identificar rostos humanos em imagens ou vídeos, determinar se dois rostos em diferentes imagens pertencem ao mesmo indivíduo e até mesmo buscar um rosto em um vasto conjunto de imagens existentes. Sistemas de segurança biométrica fazem uso do reconhecimento facial para uma identificação exclusiva durante integrações ou logins de usuários, bem como para fortalecer a atividade de autenticação (AWS, 2023).

A incorporação das Redes Neurais Convolucionais (CNNs) ao reconhecimento facial potencializa ainda mais a compreensão na identificação de pessoas, mesmo diante de imagens com ruídos ou adversidades. As redes convolucionais, variantes de redes neurais, empregam operações matemáticas de convolução ao invés das tradicionais multiplicações gerais de matrizes. A convolução incorpora ideias essenciais para aprimorar sistemas de aprendizado

de máquina, incluindo interações esparsas, compartilhamento de parâmetros e representações equivariantes. Além disso, a convolução possibilita a manipulação de entradas com tamanhos variáveis (Goodfellow; Courville; Bengio, 2015).

As redes convolucionais, em contraste com camadas de redes neurais tradicionais, frequentemente adotam interações esparsas, também conhecidas como conectividade esparsa ou pesos esparsos (Goodfellow; Courville; Bengio, 2015).

A importância da conectividade esparsa nas CNNs reside na sua capacidade de reduzir significativamente a quantidade de parâmetros e cálculos necessários, facilitando o treinamento de modelos em grandes conjuntos de dados. Essa propriedade permite que as CNNs capturem características locais e hierárquicas dos dados de entrada, promovendo uma maior eficiência computacional e melhorando a capacidade de generalização. Além disso, a conectividade esparsa diminui o risco de *overfitting*, pois força a rede a aprender filtros mais robustos e relevantes, ao invés de memorizar os dados de treinamento.

O reconhecimento facial, além de eficiente, tornou-se uma forma ágil de verificação. Comparado a outras tecnologias biométricas, como impressões digitais ou reconhecimento de retina, o reconhecimento facial é mais rápido e conveniente. Além disso, sua aplicação envolve menos pontos de contato, quando comparada à inserção de senhas ou *PINs* (AWS, 2023).

Empresas adotam a tecnologia de reconhecimento facial como alternativa às senhas, fortalecendo assim as medidas de segurança cibernética. A autenticação através do reconhecimento facial é um desafio para acesso não autorizado, uma vez que alterações físicas no rosto são impossíveis. Os *softwares* de reconhecimento facial também se estabelecem como uma ferramenta de segurança conveniente e altamente precisa para desbloquear *smartphones* e outros dispositivos pessoais (AWS, 2023).

Diante deste contexto, esse projeto visa responder a seguinte questão de pesquisa: - **Como as Redes Neurais Convolucionais podem ser usadas para otimizar o reconhecimento facial, auxiliando na identificação e no**

reconhecimento de pessoas em filmagens de segurança com maior rapidez e eficiência, levando em consideração sistemas de segurança residencial?

1.1. Objetivo Geral

- Desenvolver um sistema de segurança residencial baseado em uma Convolutional Neural Network (CNN) para processar e reconhecer rostos, autenticando os moradores previamente cadastrados no sistema.

1.2. Objetivos Específicos

- Utilizar a CNN Inception_V3 pré-treinada para reconhecimento facial.
- Treinar um modelo para identificação das pessoas cadastradas no sistema.
- Autenticar os rostos presentes nas filmagens de segurança com os dados no sistema.
- Disparar notificação de alertas via Telegram com a detecção de possível violação de segurança a residência.

1.3. Justificativa

O trabalho foi motivado pela crescente demanda por segurança residencial, que segundo Flávia Albuquerque, repórter da Agência Brasil, vem crescendo cerca de 40% no Brasil nos últimos cinco anos. Essa demanda é impulsionada por diversos fatores, como a crescente violência urbana, a maior conscientização sobre a importância da segurança residencial e o desenvolvimento de tecnologias mais sofisticadas e acessíveis.

Nesse contexto, o reconhecimento facial é uma tecnologia promissora com o potencial de melhorar a segurança residencial de diversas maneiras. Por exemplo, o reconhecimento facial pode ser usado para controlar o acesso a condomínios, casas e outros espaços residenciais, detectar intrusos em propriedades residenciais e identificar vítimas de desastres ou emergências.

No entanto, os sistemas de reconhecimento facial tradicionais, baseados em biometria manual ou senhas, podem ser lentos e ineficientes, especialmente em emergências. As redes neurais, por outro lado, são um tipo de inteligência

artificial que pode ser treinada para identificar padrões complexos em dados. Elas têm sido usadas com sucesso para melhorar o desempenho de sistemas de reconhecimento facial em diversos cenários, incluindo a segurança residencial.

2. REFERENCIAL TEÓRICO

Esse capítulo traz os principais conceitos a respeito de Reconhecimento Facial, Redes Neurais convolucionais, métodos e técnicas utilizadas para esta prática. Além disso, traz também trabalhos relevantes relacionados a este tema.

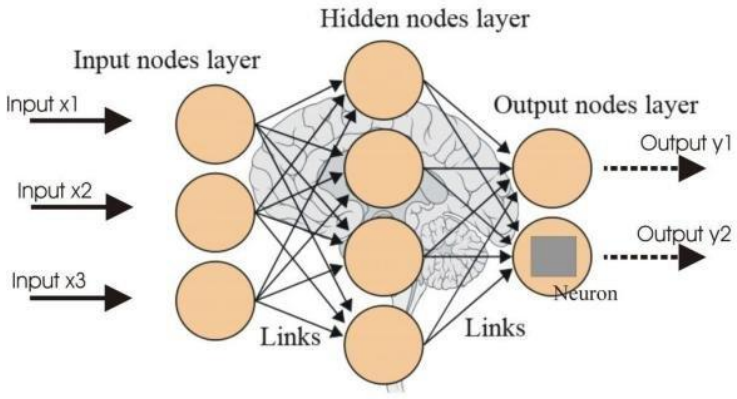
2.1. Redes Neurais Artificiais

As redes neurais artificiais (RNAs) ou *Artificial Neural Networks* (ANNs), representam um método fundamental na inteligência artificial que se inspira na estrutura e funcionamento do cérebro humano. Elas constituem um subcampo essencial do aprendizado de máquina, conhecido como aprendizado profundo, e são capazes de processar e interpretar dados complexos de maneira semelhante à cognição humana.

De acordo com a AWS (2023), uma rede neural consiste em uma rede de neurônios interconectados organizados em camadas, assemelhando-se à organização neural do cérebro humano. Essa arquitetura única permite que a rede neural crie um sistema adaptativo, no qual os computadores podem aprender com os erros e aprimorar continuamente seu desempenho. As redes neurais artificiais são empregadas na resolução de problemas desafiadores, como resumir documentos ou realizar o reconhecimento facial com alta precisão.

A Figura 1 ilustra uma RNA com camadas escondidas que são modelos inspirados no cérebro humano, compostos por camadas interconectadas de neurônios artificiais. As camadas escondidas permitem aprender relações complexas nos dados. O número de camadas e neurônios deve ser balanceado para evitar *overfitting*. RNAs com camadas escondidas são usadas em áreas como reconhecimento de imagens, processamento de linguagem natural e detecção de anomalias.

Figura 1 - Rede neural artificial genérica



Fonte: Rajabi, 2019

Inicialmente, o objetivo da abordagem de rede neural era criar sistemas computacionais capazes de emular a resolução de problemas de forma semelhante ao cérebro humano. No entanto, ao longo do tempo, os pesquisadores expandiram suas aplicações e afastaram-se de uma abordagem estritamente baseada na biologia. Como resultado, as redes neurais passaram a ser utilizadas em diversas tarefas, oferecendo suporte à solução de problemas em uma variedade de domínios (SAS Insights, 2023).

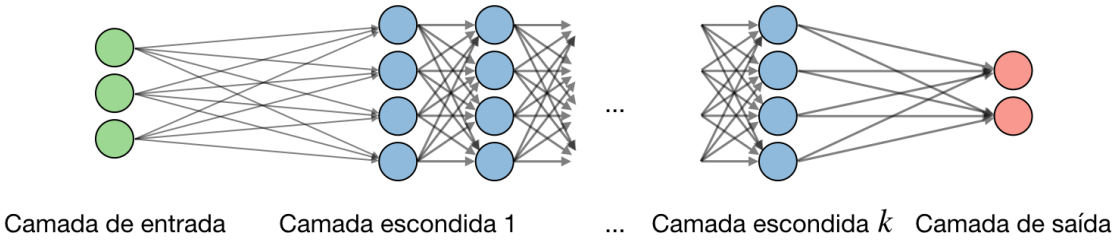
Segundo Weiss (2021), as redes neurais são frequentemente caracterizadas por três camadas essenciais: a camada de entrada, a camada oculta e a camada de saída. Quando uma rede neural possui mais de uma camada oculta, ela é denominada "profunda", daí o termo *Deep Learning* (DL). Nas redes de monocamada, uma única camada de neurônios está diretamente ligada aos nós de entrada, fornecendo diretamente os dados de saída. Já nas redes de multicamadas, existem uma ou mais camadas intermediárias de neurônios na camada oculta, possibilitando uma maior interação e complexidade entre os neurônios.

Uma característica importante das redes neurais é que os neurônios podem estar conectados de forma total ou parcial. Conexões totais ocorrem em camadas densas, onde cada neurônio está conectado a todos os neurônios da camada seguinte, permitindo a captura de interações complexas. Conexões parciais ocorrem onde cada neurônio se conecta apenas a uma região local da

camada anterior, detectando padrões locais e reduzindo a complexidade computacional.

A Figura 2 apresenta um diagrama de uma RNA com duas camadas escondidas, ilustrando a complexa rede de conexões que permite que a RNA aprenda relações abstratas entre os dados.

Figura 2 - Arquitetura de uma rede neural de aprendizado profundo



Fonte: Amidi; Amidi, 2018

Um neurônio artificial é uma unidade computacional projetada para emular as propriedades fundamentais de um neurônio biológico. Cada neurônio artificial em uma rede desempenha um papel simplificado, mas ao serem combinados em grande quantidade e conectados de maneira complexa, exibem um comportamento notavelmente poderoso e versátil. A estrutura de um neurônio artificial pode ser decomposta em quatro componentes principais: entradas (*inputs*), pesos (*weights*), viés (*bias*) e uma função de ativação (Ashade, 2024).

As redes neurais são treinadas com múltiplos dados de treinamento, o que lhes permite aprender e aprimorar sua precisão ao longo do tempo. À medida que esses algoritmos de aprendizagem são ajustados para maximizar a precisão, eles se tornam ferramentas poderosas nas áreas de ciência da computação e inteligência artificial. Eles são capazes de classificar e agrupar dados a uma velocidade notável, tornando possível a execução de tarefas como reconhecimento de fala e reconhecimento de imagem em questão de minutos, em contraste com a análise manual que requeria horas de especialistas humanos (IBM, 2023).

2.1.1. Redes Neurais Convolucionais

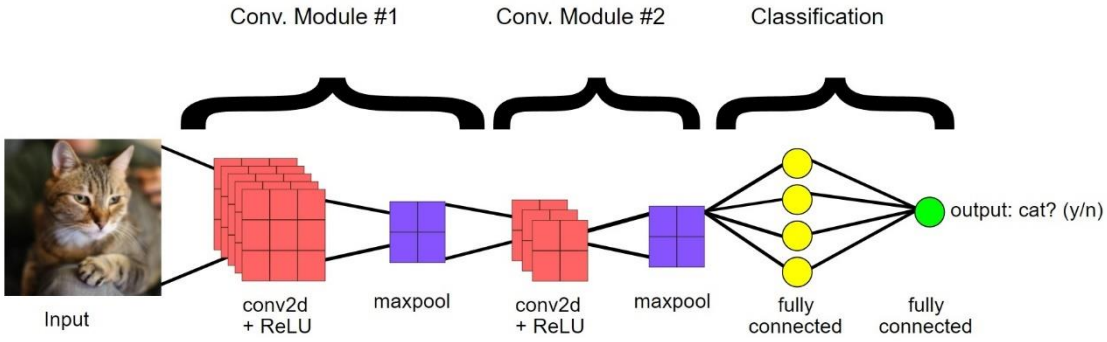
Segundo Martin (2018), nós, humanos, somos capazes de observar um Dogue Alemão e reconhecer que, apesar de seu grande porte, ele ainda é um

cachorro. Em contrapartida, os computadores identificam apenas números. Então, como eles conseguem diferenciar imagens de um Dogue Alemão das imagens de um cavalo? Essa diferenciação é possível graças ao processamento da representação numérica dos pixels por meio de várias camadas de uma Rede Neural Convolucional (CNN). Desse modo, diversas características do Dogue Alemão são identificadas, permitindo que o sistema conclua que se trata de um cão.

As CNNs consistem em um sistema de neurônios interconectados com a diferença de que cada neurônio possui um peso e uma tendência de aprendizado. Outra diferença que define esta rede é o fato de utilizar operações matemáticas lineares, chamadas convoluções, ao invés da tradicional multiplicação geral de matrizes em pelo menos uma de suas camadas. A camada oculta é composta da camada convolucional, camada de *pooling*, e uma camada totalmente conectada (*Fully Connected* - FC), além das camadas de normalização. Enquanto a camada de convolução tem a finalidade de mesclar dois conjuntos de informações, a camada de *pooling* é usada para reduzir a dimensionalidade, associando a saída do agrupamento de neurônios em uma camada de neurônio único. Por fim, a camada FC conecta todos os neurônios de uma camada a todos de outra camada (Weiss, 2021).

A Figura 3 ilustra uma CNN para a classificação de imagens, que processa uma imagem de um gato através de dois módulos convolucionais, cada um com uma camada de convolução seguida por ReLU e *Maxpooling*, para extrair e reduzir características importantes. Em seguida, essas características são passadas por camadas totalmente conectadas para gerar uma saída que indica se a imagem contém um gato ou não.

Figura 3 - Arquitetura de uma rede neural convolucional



Fonte: Google for Developers, 2022

Essas camadas de operações matemáticas auxiliam os computadores na análise minuciosa de imagens, processando pequenas partes de cada vez, com intuito de identificar objetos específicos, animais ou qualquer outro alvo desejado. No entanto, podem cometer erros, principalmente no início do seu processo de treinamento.

Em uma CNN, a camada de entrada é a primeira camada da rede. Ela é projetada para acomodar as imagens de entrada. As imagens são representadas como matrizes tridimensionais, geralmente com as dimensões altura x largura x profundidade (canais de cores).

As dimensões espaciais de uma imagem são representadas pela sua altura e largura, o que equivale à resolução da imagem em termos de pixels. A profundidade, por sua vez, se refere ao número de canais de cores presentes na imagem. No caso de imagens coloridas no padrão RGB *(Red, Green, Blue)*, a profundidade é de 3, pois cada pixel é composto por três valores que representam as intensidades das cores vermelha, verde e azul. Antes de serem inseridas na CNN, é comum realizar procedimentos de pré-processamento, que podem incluir o redimensionamento para um tamanho específico, normalização das intensidades dos pixels e aumento de dados para ampliar a variabilidade dos dados de treinamento, entre outros processos.

As camadas ocultas nas redes neurais convolucionais desempenham funções matemáticas cruciais, conhecidas como convoluções. Essas camadas desempenham um papel vital na classificação de imagens, uma vez que são capazes de extrair características relevantes, fundamentais para o

reconhecimento e classificação eficazes. Esse formato reformulado facilita o processamento sem sacrificar recursos essenciais, que desempenham um papel crucial na obtenção de previsões precisas. Cada camada oculta é especializada na extração e processamento de diferentes aspectos das imagens, como bordas, cores e profundidade (AWS, 2023).

A convolução (I) é uma operação matemática que descreve a operação sobre duas funções para produzirem uma terceira função expressando como uma é modificada pela outra. Como analogia pode-se imaginar estar entre dois palcos durante um festival de música ao se mover em direção a um ou outro palco, uma das músicas será ouvida melhor que a outra, no meio do caminho, os sinais irão se sobrepor e gerar ruídos.

$$H(x,y) = \sum_{a=0}^{n-1} \sum_{b=0}^{n-1} E(x-a, y-b) K(a,b) \tag{I}$$

A equação para o cálculo da convolução, dada uma entrada *E* e um kernel *K* de dimensão *n* × *n*, consiste na soma dos produtos entre cada elemento do kernel e a área correspondente da entrada a partir das posições *x* e *y*. O resultado desse cálculo é uma resposta de ativação.

Em linhas gerais, a convolução é uma operação que, a partir de duas funções reais, resulta em uma terceira função que mede a soma do produto dessas funções ao longo da região subentendida por sua superposição, considerando o deslocamento entre elas.

Na área de análise funcional e processamento de sinais, a função resultante da convolução mede a soma do produto das funções ao longo da região sobreposta devido ao deslocamento entre elas. Este conceito de convolução está intrinsecamente relacionado a várias áreas, como óptica de Fourier, teoria das vibrações com a integral de Duhamel, estudo de sistemas lineares invariantes no tempo através do Teorema de Borel e até mesmo em estatística e processamento de sinais com funções de correlação e autocorrelação. Além disso, é fundamental em aplicações de análise de

imagens, incluindo digitalização, alisamento, embaçamento e correção de aberração cromática.

A convolução espacial discreta, amplamente utilizada em visão computacional, emprega imagens, que são representadas como matrizes, em vez de funções contínuas ou sinais. Esse processo de convolução não exige mais uma abordagem contínua, em vez disso, ele envolve a sobreposição de imagens, a multiplicação de valores de pixel a pixel e a soma resultante, fornecendo o valor da convolução para cada ponto. A natureza das convoluções realizadas pode variar dependendo do tamanho das imagens, com operações de convolução direta quando as imagens têm o mesmo tamanho e o deslocamento e cálculo adequados das operações de convolução quando as imagens têm tamanhos diferentes (Rangel, 2020).

Nas CNNs as camadas convolucionais são responsáveis pela detecção de características locais em uma imagem. Cada camada convolucional consiste em um conjunto de filtros ou *kernels* que deslizam pela imagem de entrada, realizando operações de convolução. Essas operações aplicam pesos aos pixels da imagem, realçando regiões com características relevantes, como bordas, texturas e padrões. Dessa forma, a rede neural convolucional consegue aprender e extrair características fundamentais das imagens de treinamento.

Uma das vantagens da convolução é que ela compartilha parâmetros entre diferentes regiões da imagem. Isso significa que o mesmo filtro é aplicado a várias partes da imagem, permitindo que a rede neural convolucional aprenda a detectar características em diferentes posições. Essa capacidade de compartilhamento de parâmetros torna as CNNs eficientes em termos de memória e cálculo, além de capturar informações espaciais invariantes (Awari, 2023).

Após cada operação de convolução, uma CNN aplica uma transformação com a função de Unidade Linear Retificada (ReLU) ao mapa de características, introduzindo não linearidade ao modelo. A função ReLU retorna zero para todos os valores negativos e o próprio valor para valores positivos. Essa operação ajuda a evitar o problema de desvanecimento do gradiente, promovendo a

aprendizagem de representações mais complexas e permitindo que a rede capture melhor as relações não lineares nos dados.

Finalizadas as operações de convolução, a camada de *pooling* é executada na CNN desempenhando um papel importante na redução da dimensionalidade dos mapas de características, para tornar o processamento mais eficiente e ajudar a evitar o *overfitting* que ocorre quando o modelo se ajusta tão bem aos dados de treinamento que perde a capacidade de generalizar para novos dados, resultando em desempenho inferior em dados de teste. A camada faz isso selecionando regiões nos mapas de características e aplicando operações de agregação, como *Max Pooling* ou *Average Pooling* (Máximo ou Média), para criar um mapa de características com uma resolução espacial menor, mas com informações relevantes preservadas. Essa camada também ajuda a tornar as características mais invariantes a pequenas translações na imagem de entrada.

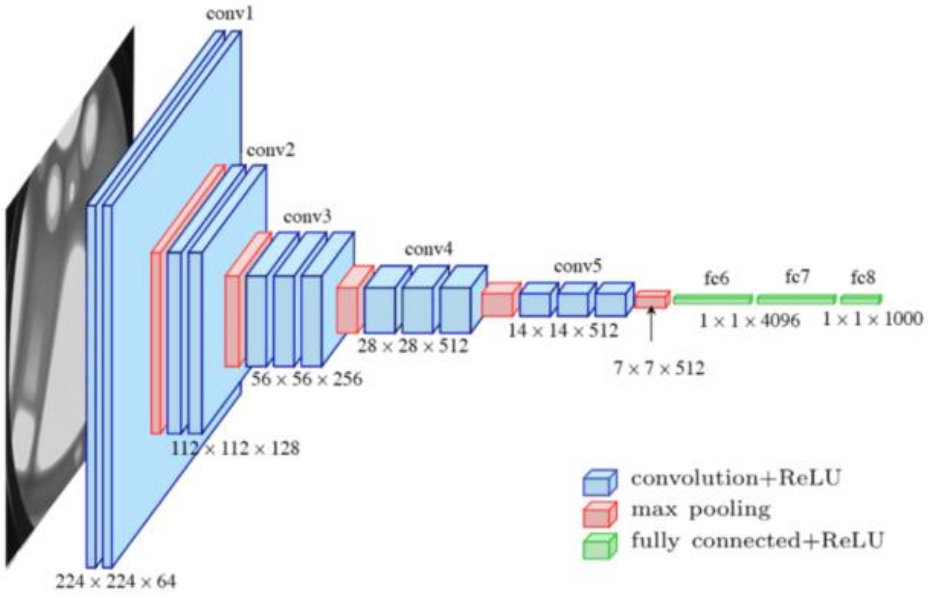
Max Pooling também funciona como um supressor de ruído. Ele descarta completamente as ativações ruidosas e realiza eliminação de ruído junto com redução de dimensionalidade. Por outro lado, o *Average Pooling* simplesmente realiza a redução da dimensionalidade como um mecanismo de supressão de ruído. Portanto, é possível dizer que o *Max Pooling* tem um desempenho superior ao *Average Pooling* (Saha, 2018).

Assim como na convolução, na camada de *pooling* é escolhida uma unidade de área, por exemplo 2x2, para transitar por toda a saída da camada anterior. A unidade é responsável por resumir a informação daquela área em um único valor. Se a saída da camada anterior for 24x24, a saída do *pooling* será 12x12. Além disso, é preciso escolher como será feita a sumarização. O método mais utilizado é o *Max Pooling*, no qual apenas o maior número da unidade é passado para a saída. Essa sumarização de dados serve para diminuir a quantidade de pesos a serem aprendidos e para evitar *overfitting* (Alves, 2018).

A camada totalmente conectada em uma CNN desempenha o papel combinatório das informações das camadas de convolução e *pooling* para produzir a saída final da rede. Ela faz isso através de conexões densas, pesos ajustáveis, funções de ativação e treinamento iterativo para aprender a

representação dos dados e realizar a tarefa desejada, como a classificação de imagens. A Figura 4 ilustra o funcionamento de uma CNN como explicado no texto acima.

Figura 4 - Funcionamento de uma CNN

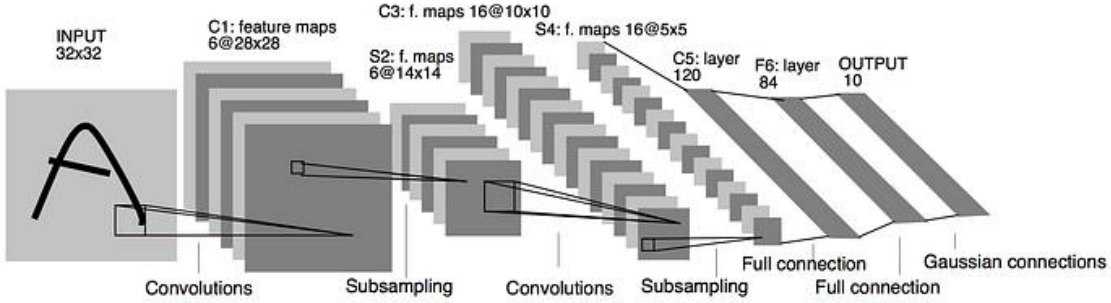


Fonte: Kumar, 2023

As redes neurais convolucionais, foram criadas por um grupo de pesquisadores da Universidade de Toronto, liderados por Yann LeCun, em 1998. O primeiro modelo de CNN proposto por esse grupo foi a *LeNet*, que foi usado para reconhecer dígitos manuscritos.

A *LeNet* foi um avanço significativo em relação aos métodos de reconhecimento de imagens existentes na época, e foi um dos fatores que contribuíram para o ressurgimento do interesse em redes neurais artificiais nos anos 2000. Projetada originalmente para tarefas de reconhecimento de caracteres escritos à mão e foi usada pelo *United States Postal Service* (USPS) para automatizar a classificação de códigos postais em envelopes. A arquitetura da *LeNet* incluía camadas de convolução, camadas de *pooling* e camadas totalmente conectadas. Ilustrado na Figura 5 é possível ver a arquitetura da rede LeNet de forma dinâmica.

Figura 5 - Arquitetura da rede LeNet



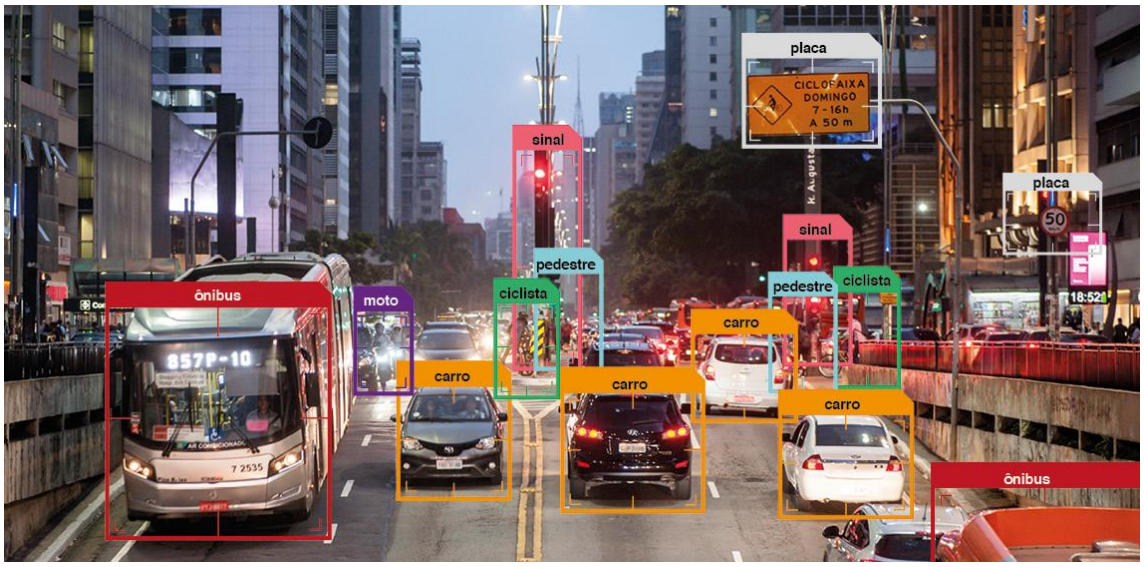
Fonte: More, 2020

A importância das CNNs foi posteriormente ampliada, em 2012, uma equipe liderada por Alex Krizhevsky participou do desafio *ImageNet Large Scale Visual Recognition Challenge* (ILSVRC) e ganhou com uma CNN chamada "*AlexNet*". Isso marcou um grande avanço no campo da visão computacional e estimulou o interesse e o desenvolvimento contínuo de CNNs para uma ampla gama de aplicações, incluindo reconhecimento de objetos em imagens, segmentação de imagens, diagnóstico médico por imagem e muito mais.

2.2. Visão Computacional

A visão computacional é uma ciência que capacita as máquinas a perceberem o mundo ao seu redor, decodificando informações significativas a partir de imagens capturadas por câmeras de vídeo, sensores, *scanners* e outros dispositivos similares. Como exemplificado na Figura 6. Essas informações possibilitam a identificação, manipulação e interpretação dos objetos que compõem uma imagem (Milano; Barrozo, 2014).

Figura 6 - Visão do mundo externo por uma máquina



Fonte: Andrade, 2019

Em contraste com os seres humanos, as máquinas não experimentam fadiga. Elas podem ser treinadas para analisar milhares de ativos ou produtos em questão de minutos, automatizando a detecção de defeitos que seriam praticamente imperceptíveis ao olho humano.

A eficácia da visão computacional depende amplamente de um vasto banco de dados. As soluções de visão computacional examinam repetidamente informações até que todas as percepções necessárias para sua tarefa designada sejam obtidas. Por exemplo, para que um computador seja capaz de reconhecer culturas saudáveis, ele deve ter acesso a milhares de imagens de referência visual de culturas, terras agrícolas, animais e outros objetos relacionados. Somente após esse treinamento extenso, ele será capaz de distinguir eficazmente diferentes tipos de culturas saudáveis, identificar culturas doentes, avaliar a qualidade das terras agrícolas e detectar pragas e outros elementos indesejados entre as culturas.

Embora a tecnologia de processamento de informações visuais já existisse há algum tempo, muitos de seus processos exigiam intervenção humana, eram demorados e suscetíveis a erros. Por exemplo, no passado, a implementação de um sistema de reconhecimento facial exigia que desenvolvedores marcassem manualmente milhares de imagens com pontos de dados cruciais, como a largura da ponte nasal e a distância entre os olhos. Automatizar essas tarefas requer uma grande capacidade computacional, uma

vez que os dados de imagem são não estruturados e sua organização por computadores é uma tarefa complexa (AWS, 2023).

A partir da primeira década dos anos 2000, o avanço tecnológico acelerou rapidamente, tornando possível o uso da visão computacional em aplicações surpreendentes, como o notório *"Deep Fake"*. Popularizado em 2020, essa técnica permite substituir a imagem de uma pessoa por outra ou até mesmo criar imagens de pessoas fictícias em gravações de vídeo (PixForce, 2022).

Devido a esses avanços, hoje, a Visão Computacional está presente em uma ampla gama de aplicações, desde sistemas bancários, análise de imagens médicas até sistemas de segurança, revolucionando a forma como interagimos com o mundo e como as máquinas compreendem nosso ambiente.

2.2.1. Reconhecimento Facial

Os humanos sempre tiveram a capacidade inata de reconhecer e distinguir rostos, mas só recentemente os computadores demonstraram a mesma capacidade. Em meados da década de 1960, os cientistas começaram a trabalhar no uso do computador para reconhecer rostos humanos (Bonsor; Johnson, 2001).

As máquinas conseguem realizar tal processo através da visão computacional, para identificar pessoas, lugares e coisas em imagens com precisão humana (ou superior) com muito mais rapidez e eficiência. Com tecnologias de IA complexa, a visão computacional automatiza a extração, análise, classificação e compreensão de informações úteis de dados de imagem. Os dados de imagem assumem muitas formas, como: imagens simples, sequências de vídeo, visualizações de várias câmeras, dados tridimensionais (AWS, 2023).

No passado, o *software* de reconhecimento facial dependia de uma imagem 2D para comparar ou identificar outra imagem 2D do banco de dados. Para ser eficaz e precisa, a imagem capturada precisava ser de um rosto olhando quase diretamente para a câmera, com pouca variação de luz ou expressão facial da imagem no banco de dados (Bonsor; Johnson, 2001).

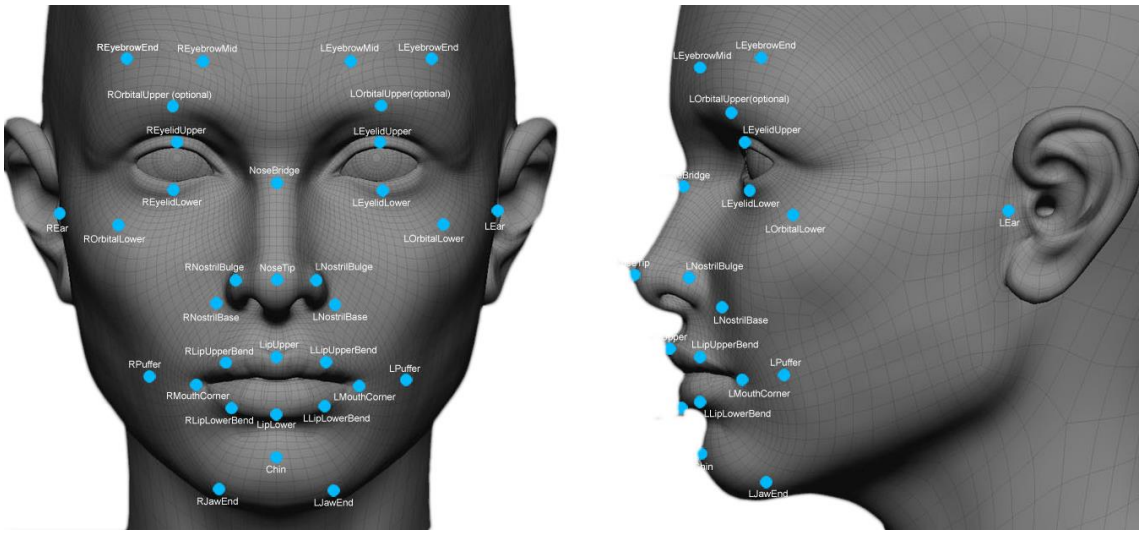
Os seres humanos usam seus olhos para observar o ambiente e obter contexto para diferenciar entre pessoas, medir distâncias, calcular velocidades e detectar erros. O reconhecimento facial permite que as máquinas alimentadas por IA realizem esses mesmos processos usando uma combinação de câmeras, algoritmos e dados. Eles podem ser treinados para analisar milhares de rostos em minutos, automatizando a detecção de pessoas suspeitas, através de pontuações de confiança.

O reconhecimento facial apesar de ser dito de forma generalizada, é composto por algumas etapas de detecção facial e autenticação facial, que juntos permitem a máquina identificar uma pessoa já conhecida.

Ambos os sistemas de detecção e comparação facial podem fornecer uma estimativa do nível de confiança da previsão na forma de uma probabilidade ou pontuação de confiança. Por exemplo, um sistema de detecção facial pode prever que uma região da imagem é uma face com uma pontuação de confiança de 90%, e outra região da imagem é uma face com uma pontuação de confiança de 60%. A região com a maior pontuação de confiança deve ter uma maior probabilidade de conter uma face. [...] Da mesma forma, um sistema de comparação facial pode não estabelecer a correspondência entre duas faces pertencentes à mesma pessoa (detecção perdida/falso negativo) ou pode prever, incorretamente, que duas faces de pessoas diferentes são a mesma pessoa (alarme falso/falso positivo) (AWS, 2023).

Cada rosto possui características distintas chamadas de pontos nodais ou marcos, que são essenciais para a identificação. Geralmente, um rosto humano é composto por cerca de 80 pontos nodais, distância entre os olhos, largura do nariz, profundidade das órbitas oculares, forma das maçãs do rosto e comprimento da linha da mandíbula são alguns deles. O *software* de reconhecimento facial examina esses pontos nodais, considerando fatores do rosto para identificar e diferenciar os indivíduos. Como ilustrado na Figura 7, a seguir.

Figura 7 - Exemplos de pontos nodais usados no mapeamento da face



Fonte: Projeto Draft, 2018

Depois de mapeados esses pontos, uma imagem tridimensional é gerada e transformada em uma sequência de números. Esses números, por sua vez, são armazenados e reconhecidos pelo sistema, gerando a partir dessas informações a identidade facial. Essa identidade facial é, posteriormente, comparada às informações armazenadas em bancos de dados, e é por meio da comparação desses dados que o sistema estabelece o nível de similaridade entre o rosto analisado e a imagem existente (Barros, 2023).

3. DESENVOLVIMENTO

Neste capítulo, serão tratados os principais métodos e técnicas utilizadas para a construção e execução prática do sistema de reconhecimento facial. Além disso, também incluirá os devidos testes e resultados obtidos durante todo o processo.

Segundo sua natureza, este trabalho é um resumo de assunto, pois reúne, analisa e discute conhecimentos e informações que já foram publicadas acerca do tema. Portanto, a pesquisa é classificada dessa forma, pois sistematiza uma área de conhecimento, indicando sua evolução histórica e estado (WAZLAWICK, 2014).

Quanto aos objetivos é uma pesquisa explicativa, esta é uma pesquisa mais completa pois visa analisar os dados observados, buscar suas causas e explicações, ou seja, fatores determinantes desses dados (WAZLAWICK, 2014).

No que diz respeito aos procedimentos técnicos adotados, este estudo assume uma abordagem experimental, uma vez que desenvolve e controla variáveis experimentais no contexto da segurança com inteligência artificial. Essa metodologia permite a análise das variáveis obtidas a partir da detecção e reconhecimento facial, viabilizando o agrupamento e análise dos fatores de fidelidade no reconhecimento das faces.

3.1. Implementação

O processo de desenvolvimento da aplicação teve início com a instalação dos programas essenciais para sua execução, bem como seus métodos necessários. Será discutida a utilização dos serviços OpenCV e InceptionV3, com a linguagem Python na versão 3.10, responsáveis por conduzir o processo exigente de detecção e identificação das imagens, viabilizado por meio da integração com suas bibliotecas.

Os ambientes de desenvolvimento utilizados para a execução do projeto foram o *Visual Studio Code* (VSCode) e o *Google Colaboratory* (Colab), juntamente com o sistema Windows. Estes foram empregados em conjunto com as seguintes bibliotecas, conforme ilustrado na Figura 8, cada uma desempenhando funções específicas.

Figura 8 – Bibliotecas utilizadas

```
1  import os
2  import cv2
3  import glob
4  import torch
5  import face_recognition
6  import api.telegram_bot as api
7  import torchvision.transforms as transforms
8  from datetime import datetime
9  from torchvision.models import inception_v3
```

Fonte: Autoria Própria, 2024

- Os – Fornece funções para interagir com o sistema operacional subjacente;
- Cv2 – Biblioteca de código aberto amplamente utilizada para processamento de imagens e visão computacional;

- Glob – Fornece uma função para encontrar todos os nomes de arquivos que correspondem a um padrão especificado;
- Torch - Framework de machine learning de código aberto, usado para criar e treinar modelos de aprendizado profundo;
- Face_recognition – Fornece métodos simples para detectar, identificar e manipular faces em imagens;
- Api.telegram_bot - Biblioteca proprietária para integrar e interagir com a *API* do Telegram;
- Torchvision.transforms – Módulo do PyTorch que fornece operações de transformação de imagem comumente usadas, como redimensionamento, recorte, normalização etc.;
- Datetime – Fornece classes para manipulação de datas e horas em Python;
- Inception_v3 – Modelo de CNN pré-treinado desenvolvido pela equipe do Google e é conhecido por sua arquitetura complexa e desempenho impressionante em várias tarefas de visão computacional;
- Torch.nn – Módulo do PyTorch que fornece classes e funções para construir e treinar redes neurais;
- Torch.optim – Módulo do PyTorch que fornece implementações de vários algoritmos de otimização;
- Matplotlib – Biblioteca para criação de gráficos e visualizações em Python;
- Datasts – Submódulo do torchvision que fornece classes para carregar e pré-processar conjuntos de dados usados em visão computacional e aprendizado profundo;

Inicialmente, foi realizada uma coleta de dados brutos, focando em imagens contendo rostos de pessoas, totalizando aproximadamente 1720 fotografias. Esta coleta foi efetuada em locais aleatórios, distintos do ambiente de testes abordado neste trabalho. As imagens coletadas passaram por um processo de seleção manual, no qual foram removidas aquelas que apresentavam desfoque, baixa resolução ou características irreconhecíveis, resultando em uma redução para 672 imagens. Esta filtragem é crucial, pois dados de baixa qualidade podem impactar negativamente o treinamento do modelo. O conjunto reduzido de imagens foi então preparado para análise

através de um *script*, conforme ilustrado na Figura 9. Este *script* tinha o objetivo de extrair e recortar apenas os rostos das pessoas, eliminando dados irrelevantes e desnecessários para o treinamento do modelo de reconhecimento facial.

Figura 9 – Função de recorte de imagens para o *dataset*

```
def cut_face(name, images_dir, output_dir):
    i = 0
    for filename in os.listdir(images_dir):
        if filename.endswith(('.jpg', '.pnp')):
            image_path = os.path.join(images_dir, filename)

            image_RGB = cv2.imread(image_path)
            image_gray = cv2.cvtColor(image_RGB, cv2.COLOR_BGR2GRAY)

            face_locations = face_recognition.face_locations(image_gray)

            for _, (top, right, bottom, left) in enumerate(face_locations):
                rosto_recortado = image_RGB[top:bottom, left:right]
                output_path = os.path.join(output_dir, f"{name}_{i}.jpg")
                cv2.imwrite(output_path, rosto_recortado)
                i+=1
```

Fonte: Autoria Própria, 2024

Após a montagem do *dataset* com as imagens tratadas, deu-se início ao processo de treinamento do modelo, utilizando a arquitetura da CNN InceptionV3. O modelo foi treinado para o reconhecimento facial por meio de 68 pontos nodais, que abrangem características como olhos, nariz, boca, entre outras proporções do rosto humano.

Na Figura 10, é possível observar a classe “LandmarkDetector” presente na biblioteca PyTorch. Essa classe encapsula o modelo de detecção de pontos nodais, o qual é uma versão modificada da arquitetura InceptionV3. Nessa modificação, a camada final é substituída por uma nova camada linear, ajustando-se assim ao problema específico de detecção de pontos nodais.

Figura 10 - Subclasse LandmarkDetector do modulo PyTorch

```
class LandmarkDetector(nn.Module):
    def __init__(self, num_landmarks):
        super(LandmarkDetector, self).__init__()
        self.base_model = inception_v3(pretrained=True)
        self.base_model.fc = nn.Linear(self.base_model.fc.in_features, num_landmarks * 2)

    def forward(self, x):
        return self.base_model(x)
```

Fonte: Autoria Própria, 2024

Para uma maior eficiência do treinamento, transformações de dados foram definidas para pré-processamento das imagens do *dataset*. Estas transformações incluem redimensionamento, corte, flip horizontal, rotação, ajuste de cor e normalização como demonstrado na Figura 11.

Figura 11 - Definição das transformações de dados

```
data_transforms = {
    'train': transforms.Compose([
        transforms.RandomResizedCrop(299),
        transforms.RandomHorizontalFlip(),
        transforms.RandomRotation(degrees=30),
        transforms.ColorJitter(brightness=0.2, contrast=0.2, saturation=0.2, hue=0.1),
        transforms.ToTensor(),
        transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])
    ]),
    'val': transforms.Compose([
        transforms.Resize(299),
        transforms.CenterCrop(299),
        transforms.ToTensor(),
        transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])
    ])
}
```

Fonte: Autoria Própria, 2024

Durante o treinamento, foram utilizados valores específicos para os hiperparâmetros (parâmetros ajustáveis que influenciam o aprendizado do modelo): 128 para o tamanho do lote (*batch*), 0.00001 para a taxa de aprendizado (LR - *Learning Rate*) e foram gastas 199 épocas para o número total de iterações sobre o conjunto de dados para finalizar o treinamento. Esses valores foram escolhidos e testados meticulosamente para garantir o melhor desempenho do treinamento do modelo com o *dataset* mencionado anteriormente. Foram realizados 85 testes para ajustar os hiperparâmetros e obter os resultados apresentados posteriormente neste trabalho.

Com o modelo devidamente treinado, é possível carregá-lo junto com a arquitetura da InceptionV3, substituindo suas camadas pelas do modelo treinado. Para usá-lo no sistema, a biblioteca OpenCV é empregada para carregar a câmera de monitoramento, cujas imagens são capturadas usando o comando "cv2.VideoCapture()". Os frames capturados são então processados pelo método “face_locations” da biblioteca “face_recognition”, que otimiza a detecção de pessoas na filmagem, evitando detecções ilusórias e gerando uma lista dessas pessoas presentes na filmagem.

Posteriormente, após o pré-processamento das imagens para transformá-las em um tensor que se adeque ao modelo Inception, é possível realizar o reconhecimento facial de cada pessoa através da função “recognize_face”. Esta função determina se a pessoa é conhecida ou não pelo sistema, e sua taxa de confiança nesse reconhecimento. Apenas taxas de confiança acima de 80% são aceitas como reconhecimento correto. Como pode ser visto na Figura 12.

Figura 12 – Processo para detecção facial

```
face_locations = face_recognition.face_locations(rgb_frame)

for _, (top, right, bottom, left) in enumerate(face_locations):
    # Recortar o rosto
    face = frame[top:bottom, left:right]

    # Realizar reconhecimento facial
    face_tensor = preprocess_image(face)
    predicted_id, confidence = recognize_face(face_tensor, model)

    # Obter o nome associado ao ID previsto
    label = "Desconhecido" if predicted_id not in id_to_name else ("Alerta" if confidence < 0.80 else id_to_name[predicted_id])

    # Desenhar um retângulo ao redor do rosto detectado
    cv2.rectangle(frame, (left, top), (right, bottom), (0, 255, 0), 2)

    # Exibir o nome e a confiança associados ao rosto identificado
    text = f"{label} ({confidence:.2f})"
    cv2.putText(frame, text, (left, top-10), cv2.FONT_HERSHEY_SIMPLEX, 0.9, (255, 0, 0), 2)
```

Fonte: Autoria Própria, 2024

O sistema é projetado para diferenciar entre pessoas cadastradas e desconhecidas. Ele compara a face capturada com as faces no banco de dados e, se a correspondência estiver abaixo de um limiar de confiança predeterminado, classifica a pessoa como desconhecida. Esse limiar é ajustado para minimizar erros, garantindo que uma baixa pontuação de confiança seja interpretada corretamente como uma pessoa não cadastrada.

A função “recognize_face”, conforme mostrada na Figura 13, desempenha um papel crucial na identificação precisa dos rostos, fazendo uso do modelo de classificação. Recebendo como entrada o tensor de imagem pré-processado que representa o rosto a ser analisado, a função utiliza o modelo para buscar correspondências. Por meio de uma série de operações computacionais, a função determina a classe prevista para o rosto e a confiança associada a essa previsão. Esses resultados são fundamentais para garantir a precisão e a confiabilidade do sistema de reconhecimento facial, proporcionando uma abordagem robusta e eficaz para identificação de indivíduos.

Figura 13 – Função para reconhecimento facial

```python
# Função para identificar o rosto usando o modelo de classificação
def recognize_face(face_tensor, model):
    with torch.no_grad():
        outputs = model(face_tensor.unsqueeze(0))
        predicted_label = torch.argmax(outputs)
        confidence = torch.softmax(outputs, dim=1)[0][predicted_label]

        return predicted_label.item(), confidence.item()
```

Fonte: Autoria Própria, 2024

No caso da identificação de pessoas desconhecidas ou com uma taxa de confiabilidade abaixo de 80%, o sistema automaticamente o caracteriza como desconhecido e registra a imagem da pessoa presente na filmagem e emite um alerta ao proprietário da casa (usuário do sistema), utilizando as funções “alert_warning” e “danger_warning”, como mostrado na Figura 14.

Figura 14 – Funções para notificação de possível violação de segurança

```python
def alert_warning(date_time, file):
    id = api.load_usuarios()[0]
    msg = f"Atenção possível falha de detecção, por favor averiguar as câmeras: {date_time}"
    api.send_message(id, msg, file)

def danger_warning(date_time, file):
    id = api.load_usuarios()[0]
    msg = f"Atenção identificamos uma pessoa não cadastrada, recomenda-se examinar as filmagens: {date_time}"
    api.send_message(id, msg, file)
```

Fonte: Autoria Própria, 2024

Estas funções enviam a data e hora exatas da detecção, juntamente com a imagem da pessoa, e uma mensagem de alerta diferente para cada caso, via telegram, por meio da função “send_message” do módulo “telegram_bot” presente na biblioteca “api”. Essa biblioteca foi desenvolvida juntamente ao projeto para permitir uma fácil integração do *bot* do Telegram ao sistema.

Figura 15 – Funções para disparo de mensagens via telegram

```python
# Função para disparar mensagens
def send_message(chat_id, mensagem, imagem=None):
    try:
        bot.send_message(chat_id, mensagem)
        if imagem:
            with open(imagem, "rb") as f:
                bot.send_photo(chat_id, f)
    except Exception as e:
        print(f"Error sending message or image: {e}")
```

Fonte: Autoria Própria, 2024

Conforme demonstrado na Figura 15, o funcionamento do *bot* é simples, ele é responsável apenas pelo envio da mensagem para o usuário através de seu *Chat-ID* único do Telegram, deixando assim o proprietário ciente do possível risco à sua residência.

Figura 16 – Função para monitoramento de horário durante execução

```
85  # Função para monitoramento de horário durante execução
86  def check_time(dir_alert, dir_unknown, alert_faces, unknown_faces):
87      global time_list
88      # Obter o horário atual
89      now = datetime.now()
90
91      # Verificar se é dia 1 e o horário atual é 00:00
92      if now.day == 1 and now.hour == 0 and now.minute == 0 and now.second == 0:
93          # Chamar o método de limpeza
94          clear_folder(dir_alert)
95          clear_folder(dir_unknown)
96      # Intervalo de tempo para evitar disparos infinitos de mensagens (Spam)
97      if alert_faces | unknown_faces is not None and time_list is None:
98          time_list = now
99      elif now - time_list >= timedelta(minutes=1):
100         time_list = None
101         return True
102     else:
103         return False
```

Fonte: Autoria Própria, 2024

A função “check_time”, presente na Figura 16, desempenha um papel crucial no gerenciamento das notificações de segurança, baseando-se no tempo decorrido desde a última notificação e na remoção das imagens armazenadas nos registros de desconhecidos e alertas. Isso é fundamental para evitar problemas relacionados a direitos autorais, decorrentes do armazenamento de imagens sem autorização prévia, bem como para prevenir o esgotamento do armazenamento do sistema.

Inicialmente, a função obtém o horário atual da chamada de função e verifica se é o dia 1 do mês e se é exatamente meia-noite (00:00:00). Se ambas as condições forem atendidas, a função chama o método “clear_folder” para limpar os diretórios “dir_alert” e “dir_unknown”, assegurando que os arquivos nessas pastas sejam removidos periodicamente.

Além disso, a função é responsável por controlar a periodicidade das notificações de segurança, evitando o envio excessivo de mensagens. Para isso, verifica se o tempo decorrido desde a última notificação é maior ou igual a 1

minuto. Se for, retorna *True*, indicando que o sistema deve enviar uma nova notificação de violação caso seja identificada uma pessoa desconhecida nas filmagens. Caso contrário, retorna *False*, permitindo que o sistema aguarde antes de realizar o próximo disparo de segurança.

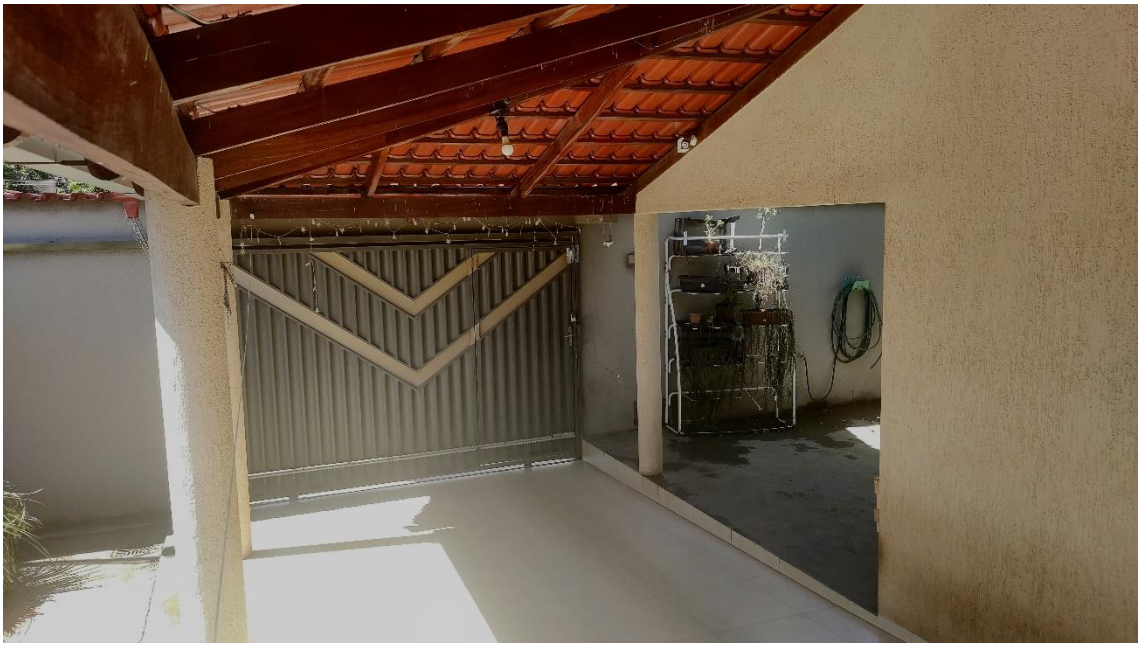
Essa função desempenha um papel fundamental para garantir que o usuário seja alertado novamente apenas se uma nova detecção ocorrer após um intervalo de tempo significativo. Além disso, o intervalo de tempo pode ser ajustado conforme a necessidade do proprietário, proporcionando flexibilidade ao sistema.

3.2. Resultados

É importante ressaltar que os resultados apresentados neste capítulo foram obtidos em condições controladas de teste e podem variar em um ambiente de uso real. No entanto, os testes realizados forneceram uma avaliação inicial do desempenho e da eficácia do sistema de reconhecimento facial implementado.

Na Figura 17, é apresentado um exemplo dos cenários de testes usados no trabalho, sendo em ambiente externo e mais próximo de uma aplicação real.

Figura 17 – Exemplos de ambientes usados para realização de testes



Fonte: Autoria Própria, 2024

3.2.1. Desempenho do Modelo de Reconhecimento Facial

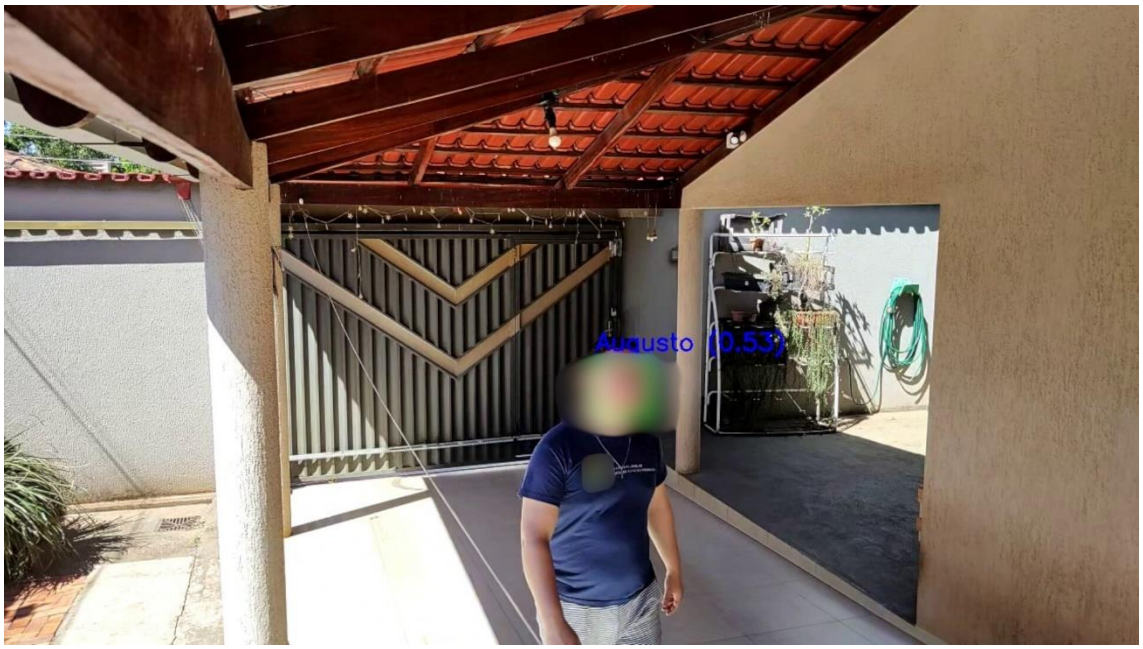
O desempenho do modelo de reconhecimento facial foi avaliado em um conjunto de testes realizados em um ambiente residencial localizado em uma zona urbana. A câmera foi estrategicamente posicionada em frente ao portão de entrada principal, proporcionando uma visão direta da entrada das pessoas na residência.

Durante os testes, foram utilizadas imagens de pessoas cadastradas e não cadastradas, além de vídeos e filmagens ao vivo. Os resultados revelaram uma taxa de acerto significativa na identificação das pessoas cadastradas, com uma taxa de confiança de reconhecimento superior a 80% na maioria dos casos.

Entretanto, observou-se uma leve perda de desempenho nos cenários de vídeos e filmagens ao vivo, o que era esperado devido aos possíveis ruídos presentes nessas imagens. Apesar disso, o modelo manteve uma taxa alta de acertos, o que indica sua eficácia em reconhecer as pessoas previamente registradas no sistema.

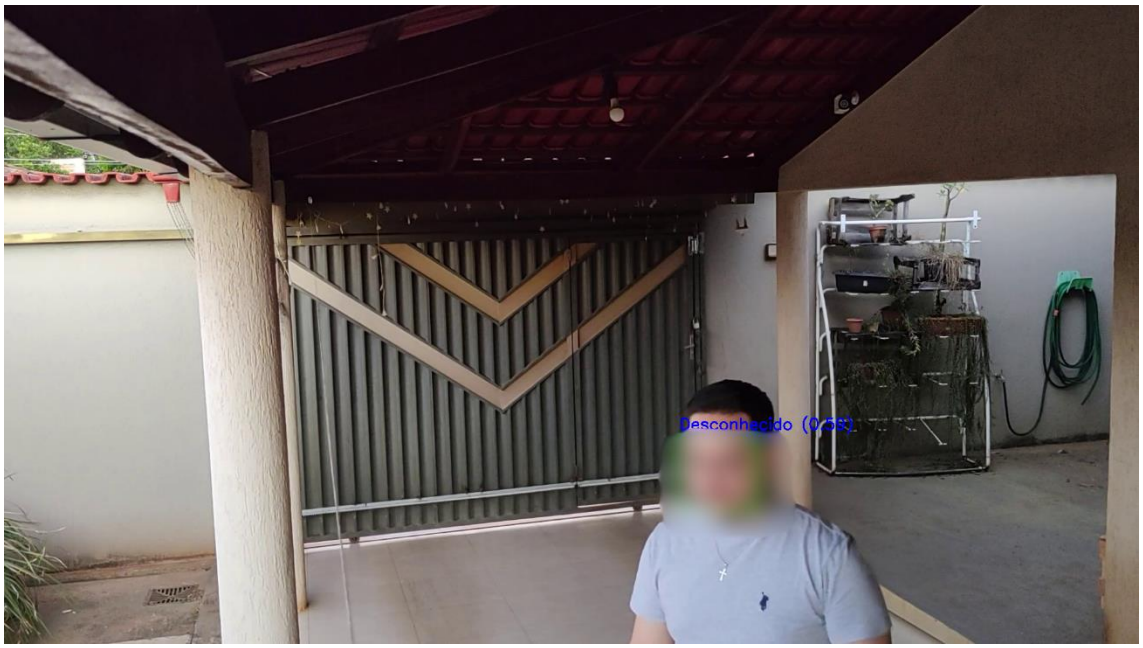
Nas Figuras 18 e 19 é possível ver exemplos dos testes realizados em filmagens ao vivo, durante o dia com boa iluminação e imagem Full HD 1080p.

Figura 18 – Detecção de pessoa cadastrada em filmagem ao vivo



Fonte: Autoria Própria, 2024

Figura 19 – Detecção de pessoa não cadastrada em filmagem ao vivo

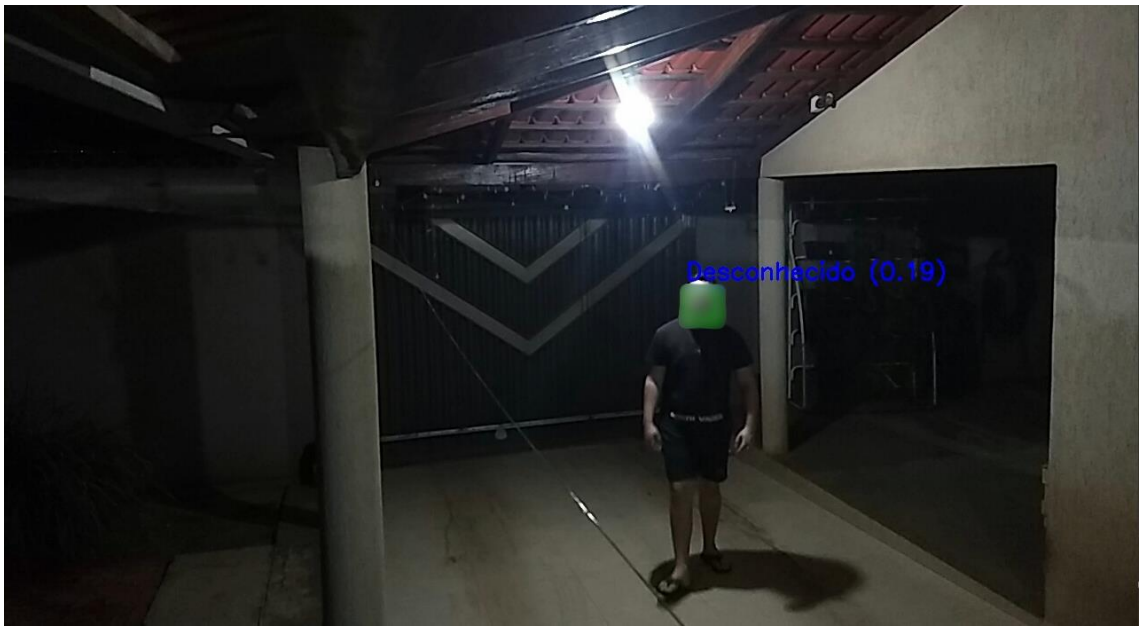


Fonte: Autoria Própria, 2024

Durante os testes, foi observado que a precisão do modelo foi influenciada por diversos fatores, incluindo a qualidade das imagens de entrada, as condições de iluminação e a posição das pessoas em relação à câmera de monitoramento. No entanto, mesmo diante desses desafios, o modelo demonstrou uma capacidade robusta de reconhecer os rostos das pessoas cadastradas na maioria das situações.

Na Figura 20 é apresentada imagem com falha de reconhecimento, ocasionado por uma iluminação instável e baixa qualidade de imagem SD 480p.

Figura 20 – Falha de reconhecimento devido à imagem de baixa qualidade



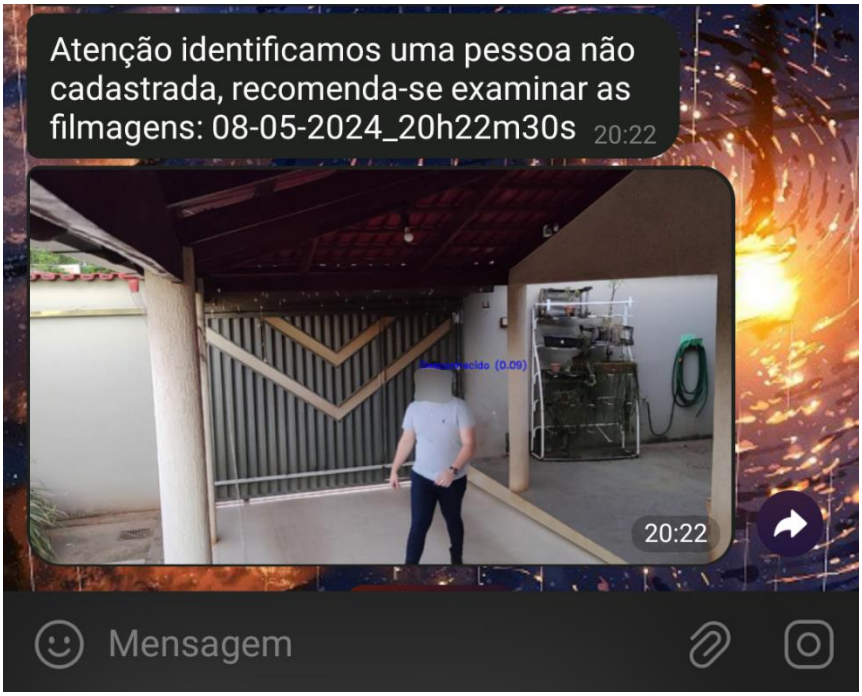
Fonte: Autoria Própria, 2024

3.2.2. Eficácia das Notificações de Segurança

As notificações de segurança foram avaliadas em termos de sua capacidade de alertar o proprietário da residência sobre possíveis violações de segurança. Durante os testes, o sistema foi capaz de detectar com sucesso pessoas não cadastradas ou com baixa taxa de confiabilidade de reconhecimento.

Quando uma pessoa desconhecida foi detectada, o sistema acionou as notificações de segurança conforme projetado, enviando ao proprietário da residência a data, hora e uma imagem da pessoa detectada, como demonstrado na Figura 21. Isso permitiu que o proprietário tomasse medidas imediatas para garantir a segurança de sua residência.

Figura 21 – Notificação de segurança via Telegram em funcionamento



Fonte: Autoria Própria, 2024

4. CONCLUSÃO

O objetivo deste trabalho foi desenvolver um sistema capaz de identificar e reconhecer pessoas utilizando um modelo treinado com a arquitetura de IA baseada no InceptionV3, integrando-o a um sistema de segurança para gerenciamento e detecção de possíveis invasões, proporcionando notificações em tempo real sobre violações de segurança e melhorando a segurança residencial.

Embora um sistema de segurança por si só seja um elemento crucial de prevenção, ele não é totalmente eficaz sem a supervisão humana das imagens capturadas. A sua utilidade é limitada à mera gravação de possíveis atividades ilícitas.

Ao longo do tempo, o reconhecimento facial tem ganhado destaque crescente na área de segurança. Isso é evidenciado pela implementação de sistemas como catracas eletrônicas em empresas, que utilizam o cadastro facial como método de acesso. Essa tendência é impulsionada principalmente pelos avanços significativos em IA nos últimos anos, possibilitando um reconhecimento mais preciso e robusto das imagens, tanto em fotografias estáticas quanto em tempo real.

Um sistema de segurança gerenciado por IA pode amenizar perdas e até impedir incidentes piores, pois é altamente responsivo e não se cansa. O proprietário é avisado no momento exato do perigo. No entanto, como não existem sistemas perfeitos à prova de falhas, seus principais empecilhos podem ser a necessidade de equipamentos de alta qualidade, o que aumenta o custo do sistema de segurança, e a constante necessidade de conexão à internet para momentos de disparo de alertas.

Os objetivos primários deste trabalho foram plenamente alcançados, uma vez que o sistema demonstrou eficiência na detecção e no reconhecimento da base de testes, com uma alta taxa de confiabilidade. Todas as tecnologias propostas foram implementadas com sucesso, o treinamento do modelo mostrou-se eficiente e responsivo quando devidamente ajustado, e a escolha da linguagem Python foi certeira, devido à sua ampla gama de bibliotecas de código aberto voltadas para IA e análise de dados, bem como à participação ativa da comunidade, que constantemente compartilha novos conhecimentos.

O ambiente Windows mostrou-se prático devido às suas ferramentas nativas e otimizadas, como o VSCode, que foi utilizado no desenvolvimento do projeto. No entanto, sua configuração não é indispensável para a execução do sistema, podendo ser substituído por um ambiente Linux mais leve e bem configurado.

O processo de treinamento realizado no ambiente Google Colab foi essencial devido ao grande poder de processamento e ao tempo economizado durante a execução. Apesar dos contratempos mencionados a seguir, o Colab permitiu que o treinamento fosse realizado em paralelo com o desenvolvimento do sistema, otimizando assim o tempo de desenvolvimento do projeto.

Durante o desenvolvimento deste trabalho, foram enfrentadas dificuldades relacionadas à utilização do Google Colab para o treinamento do modelo. Na versão gratuita, o Colab disponibiliza apenas GPUs básicas, 12,7 GB de memória RAM e um tempo de execução limitado. Muitas vezes, esses recursos foram insuficientes para concluir o treinamento, resultando em necessidade de migrar para a versão paga, que oferece GPUs mais rápidas e maior quantidade de memória RAM, resultando em uma otimização significativa do tempo de treinamento.

Entretanto, mesmo com o serviço pago, durante a execução dos treinamentos, foi possível observar picos de utilização da memória das GPUs em casos de valores de *batch* elevados. Além disso, problemas como lentidão e quedas do serviço também foram enfrentados, o que resultou em perda de desempenho, dificuldades na execução e até mesmo interrupções totais do treinamento, exigindo reinicialização. Esses contratempos foram causados principalmente pelas operações serem realizadas via rede, o que demandou mais tempo do que o necessário para concluir o treinamento.

Como sugestão para trabalhos futuros, propõem-se: o aprimoramento do modelo de reconhecimento facial citado acima, a utilização de outras tecnologias para desenvolvimento do sistema de detecção e a comparação dos resultados obtidos e a implementação de outras arquiteturas de CNNs para o treinamento do modelo de reconhecimento facial.

REFERENCIAS

ALBUQUERQUE, F. **Setor de segurança tem alta de 40% na busca por tecnologia inteligente**. Disponível em: <https://agenciabrasil.ebc.com.br/economia/noticia/2020-07/setor-de-seguranca-tem-alta-de-40-na-busca-por-tecnologia-inteligente>. Acesso em: 12 nov. 2023.

ALVES, G. **Entendendo Redes convolucionais (CNNs)**. Disponível em: <https://medium.com/neuronio-br/entendendo-redes-convolucionais-cnns-d10359f21184>. Acesso em: 15 set. 2023.

AMIDI, A.; AMIDI, S. **CS 229 - Dicas de aprendizado profundo**. Disponível em: <https://stanford.edu/~shervine/l/pt/teaching/cs-229/dicas-aprendizado-profundo>. Acesso em: 18 nov. 2023.

ASHADE, A. **Neurônios e redes neurais: Os blocos de construção**. Disponível em: <https://itshow.com.br/neuronios-e-redes-neurais-os-blocos-de-construcao/?trk=article-ssr-frontend-pulse_little-text-block>. Acesso em: 21 jun. 2024.

AWARI. **Redes Neurais Convolucionais: Como elas funcionam**. Disponível em: <https://tinyurl.com/3zffska3>. Acesso em: 14 set. 2023.

AWS. **O que é reconhecimento facial?** Disponível em: <https://aws.amazon.com/pt/what-is/facial-recognition/>. Acesso em: 25 ago. 2023.

AWS. **O que é a visão computacional?** Disponível em: <https://aws.amazon.com/pt/what-is/computer-vision/>. Acesso em: 25 ago. 2023.

AWS. **O que é uma rede neural?** Disponível em: <https://aws.amazon.com/pt/what-is/neural-network/>. Acesso em: 31 ago. 2023.

AWS. **Visão geral da detecção facial e comparação facial**. Disponível em: <https://docs.aws.amazon.com/pt_br/rekognition/latest/dg/face-feature-differences.html>. Acesso em: 09 out. 2023.

BARROS, L. **Biometria Facial: como essa tecnologia otimiza o controle de ponto**. Disponível em: <https://tangerino.com.br/blog/biometria-facial/>. Acesso em: 26 nov. 2023.

BONSOR, K.; JOHNSON, R. **How facial recognition systems work**. Disponível em: <https://electronics.howstuffworks.com/gadgets/high-tech-gadgets/facial-recognition.htm>. Acesso em: 29 ago. 2023.

DE OLIVEIRA ANDRADE, R. **As máquinas que tudo veem**. Disponível em: <https://revistapesquisa.fapesp.br/as-maquinas-que-tudo-veem/>. Acesso em: 22 nov. 2023.

Face, body, vehicle, and license plate number recognition. Disponível em: <https://ntechlab.com/pt_br/blog/2017/04/17/redes-neurais-parte-2/>. Acesso em: 28 ago. 2023.

GIL, A. **Como Elaborar Projetos de Pesquisa. 6 ed**. [s.l.] Editora Atlas Ltda, 2017.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Learning**. [s.l.] Alanna Maldonado, 2023.

GOOGLE. **Prática de ML: classificação de imagens**. Disponível em: <https://developers.google.com/machine-learning/practica/image-classification/convolutional-neural-networks?hl=pt-br>. Acesso em: 18 nov. 2023.

IBM. **O que são Redes Neurais?** Disponível em: <https://www.ibm.com/br-pt/topics/neural-networks>. Acesso em: 20 ago. 2023.

KUMAR, A. ***Different types of CNN architectures explained: Examples***. Disponível em: <https://vitalflux.com/different-types-of-cnn-architectures-explained-examples/>. Acesso em: 15 nov. 2023.

MARTIN, S. **What's the difference between a CNN and an RNN?** Disponível em: <https://blogs.nvidia.com/blog/2018/09/05/whats-the-difference-between-a-cnn-and-an-rnn/>. Acesso em: 31 ago. 2023.

MENA, I. **Verbete Draft: o que é Reconhecimento Facial**. Disponível em: <https://www.projetodraft.com/verbete-draft-o-que-e-reconhecimento-facial/>. Acesso em: 15 nov. 2023.

MORE, O. ***LeNet-5: Covolution neural network architecture***. Disponível em: <https://medium.com/analytics-vidhya/lenet-5-covolution-neural-network-architecture-23b93f75fe01>. Acesso em: 20 nov. 2023.

NUNES, Á. L. P.; REIS, P. M. S. B. DOS. **Programa de reconhecimento facial**. Disponível em: <http://repositorio.unitau.br/jspui/handle/20.500.11874/4427>. Acesso em: 30 ago. 2023.

PIXFORCE. **Introdução à visão computacional: do passado ao futuro**. Disponível em: <https://www.pixforce.com.br/post/do-passado-ao-futuro-entenda-a-vis%C3%A3o-computacional>. Acesso em: 27 set. 2023.

RAJABI, N. **ResearchGate**. Disponível em: <https://www.researchgate.net/figure/A-Generic-Neural-Network-9-Feedforward-Neural-Networks-consist-of-three-major-stages_fig1_336616532>. Acesso em: 22 nov. 2023.

RANGEL, R. F. **Visão Computacional - O que é convolução?** Disponível em: <https://medium.com/turing-talks/vis%C3%A3o-computacional-o-que-%C3%A9-convolu%C3%A7%C3%A3o-ad709f7bd6b0>. Acesso em: 26 set. 2023.

SAHA, S. **A comprehensive guide to convolutional neural networks — the ELI5 way**. Disponível em: <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53>. Acesso em: 15 set. 2023.

SANCHEZ, T. **Visão computacional: descubra o que é e onde ela é aplicada**. Disponível em: <https://santodigital.com.br/o-que-e-visao-computacional-e-para-que-serve/>. Acesso em: 28 ago. 2023.

SAS. **Redes neurais - o que são e qual sua importância?** Disponível em: <https://www.sas.com/pt_br/insights/analytics/neural-networks.html>. Acesso em: 31 ago. 2023.

SZELISKI, R. **Computer Vision: Algorithms and Applications**. [s.l.] Springer, 2022.

WAZLAWICK, R. S. **Metodologia da Pesquisa para Ciência da Computação. 2ª**. [s.l.] Campus, 2014.

WEISS, V. A. **Arquitetura de Redes Neurais Artificiais**. **Ateliware.com**, 7 jun. 2021. Disponível em: <https://ateliware.com/blog/redes-neurais-artificiais>. Acesso em: 14 set. 2023