Ferramenta para controle de presenças de alunos por reconhecimento facial

Vinícius Assunção Marins[1], Vilson Francisco Maziero[2]

[1]Discente do curso superior Bacharelado em Ciência da Computação – Instituto Federal de Educação Ciência e Tecnologia de São Paulo, Campus Presidente Epitácio

[2]Docente do curso superior Bacharelado em Ciência da Computação – Instituto Federal de Educação Ciência e Tecnologia de São Paulo, Campus Presidente Epitácio

Rua José Ramos Júnior, 27-50 - 19470-000 Presidente Epitácio, SP - Brasil

v.marins@aluno.ifsp.edu.br[1], vilson.maziero@ifsp.edu.br[2]

Resumo. *Este trabalho propõe a aplicação de redes neurais convolucionais (CNN) para o tratamento e utilização de reconhecimento facial em âmbito escolar, visto que as CNNs apresentam resultados mais precisos se comparado a outros modelos. O presente projeto apresenta o desenvolvimento de uma ferramenta de fácil uso para o controle de presenças de alunos por reconhecimento facial, contendo funcionalidades como treinamento do modelo matemático, reconhecimento facial e lançamento de presenças para os alunos. Além disso, visa o estudo e abordagem de assuntos como visão computacional, processamento de imagens, aprendizado de máquina e outras áreas da Inteligência Artificial.*

Abstract. *This work proposes the application of convolutional neural networks (CNN) for the treatment and use of facial recognition in schools, as CNNs present more accurate results compared to other models. This project presents the development of an easy-to-use tool for controlling student attendance using facial recognition, containing features such as training the mathematical model, facial recognition and launching attendance for students. Furthermore, it aims to study and approach subjects such as computer vision, image processing, machine learning and other areas of Artificial Intelligence.*

1. Introdução

Atualmente grande parte das instituições possuem um sistema integrado eletrônico para realizar a gestão dos alunos, onde é possível consultar seus dados acadêmicos, presenças, matérias, dados pessoais entre outras informações. Porém para realizar o controle de presenças desses alunos, independente do meio a ser feito, é necessária uma chamada realizada manualmente pelos professores. O controle de presenças de alunos é algo que pode ser facilitado para os professores de uma forma mais prática ou automatizada, para alunos do ensino médio ou superior.

O reconhecimento facial é uma tecnologia de biometria que permite a identificação de indivíduos por características físicas, como as feições do rosto. Essa tecnologia vem sendo cada vez mais utilizada em diferentes áreas, como segurança, comércio e educação. É possível afirmar que, no contexto educacional, o reconhecimento facial pode ser usado para controlar a presença dos alunos em salas de aulas. Com essas tecnologias, é possível verificar a presença dos alunos de forma automatizada, sem a

necessidade de chamadas manuais, o que pode ser mais rápido e não gerando uma interrupção na aula com essa atividade.

Destaca-se a utilização de redes neurais artificiais, sendo o método mais utilizado para o reconhecimento facial (Krizhevsky, Sutskever e Hinton, 2012). As redes neurais artificiais (RNA) são sistemas computacionais inspirados na estrutura e funcionamento do cérebro humano. Elas são compostas por diversas camadas de neurônios artificiais interconectados, capazes de aprender e generalizar padrões em dados de entrada. No reconhecimento facial utilizando redes neurais, geralmente as informações das características faciais são inseridas como dados de entrada para a rede neural, que será treinada para identificar padrões nessas características faciais e associá-las a uma identidade específica. Dentre os modelos de redes neurais, o modelo mais utilizado para o reconhecimento facial são as redes neurais convolucionais (CNN), por ser o modelo mais eficiente e apresentarem melhores resultados (Krizhevsky, Sutskever e Hinton, 2012).

Este trabalho é fundamentado no uso de aprendizado de máquina, principalmente, pela aplicação de redes neurais convolucionais. A utilização do aprendizado de máquina é altamente coerente para abordar o problema em questão, que consiste em automatizar o controle de presença de alunos de ensino médio ou superior. A tarefa de reconhecimento facial engloba precisamente a extração de padrões e o treinamento de um algoritmo para executar uma tarefa que normalmente é desempenhada por seres humanos.

Neste trabalho o processamento de imagens será efetivamente utilizado durante as etapas de pré-processamento das imagens da rede neural, como reconhecimento de rostos em imagens e comparação com um banco de dados de rostos conhecidos. Os processos e métodos de conversão de escala de cores e manipulação das imagens digitais será um fator de extrema relevância para o desenvolvimento do projeto, como será apresentado detalhadamente na seção de desenvolvimento. Além disso, será utilizada técnicas e recursos oferecidos pela visão computacional. Para a implementação dos métodos e algoritmos de visão computacional serão utilizadas algumas bibliotecas Python de código aberto como OpenCV e Dlib. A combinação dessas bibliotecas, juntamente com os conceitos de visão computacional, permitirá a implementação da ferramenta proposta.

Portanto, o objetivo principal do trabalho é o desenvolvimento de uma ferramenta de controle de presença de alunos utilizando reconhecimento facial através de redes neurais convolucionais (CNN). A ferramenta proposta visa fornecer uma solução automatizada para controlar a presença dos alunos em ambientes educacionais, reduzindo a necessidade de registro manual e realização de chamadas.

2. Trabalhos Relacionados

Trabalhos relacionados são descritos a seguir, possuindo tipos de abordagens similares, porém alternativas, para o desenvolvimento de um sistema de reconhecimento facial. Alguns dos trabalhos citados utilizam modelos prontos para a realização de estudos, testes e análises, em contrapartida este trabalho tem o objetivo de encontrar uma outra solução para a resolução do problema em questão de uma maneira diferente, definindo e treinando um novo modelo.

Santos (2022) desenvolveu um sistema de reconhecimento facial com base em técnicas de aprendizado de máquina, que visa a utilização de modelos previamente

validados como *Facenet* e *OpenFace*, os quais possuem resultados científicos com alto índice de eficácia. Esses modelos de reconhecimento facial de última geração, são capazes de identificar ou verificar uma pessoa a partir de uma imagem digital ou um quadro de vídeo a partir de uma entrada, comparando características faciais selecionadas com rostos dentro de um banco de dados. Os procedimentos realizados por Santos (2022) foram aplicados e comparados com diferentes classificadores como *Support Vector Machine* (SVM), *k-Nearest Neighbors* (KNN) e *Convolutional Neural Network* (CNN), e em todos os casos os resultados obtidos foram de uma acurácia de aproximadamente 100%. Vale destacar que os classificadores são algoritmos de aprendizado de máquina que podem ser usados para resolver problemas de classificação, funcionando através do treinamento de algoritmos para aprender padrões e fazer previsões a partir de dados.

Outro trabalho bastante interessante foi proposto por Diniz (2016), que realizou um estudo empírico de um sistema de reconhecimento facial utilizando o classificador KNN. Ele analisou as taxas de acurácias resultantes de um sistema de reconhecimento facial baseado nas técnicas de *Eigenfaces* e *K-Nearest Neighbors*. Além disso, ele verificou a relevância de parâmetros para essas técnicas, obtendo resultados que comprovaram que imagens com dimensões 12x9 pixels produzem as melhores taxas de acurácias de reconhecimento facial, juntamente com a medida de distância euclidiana normalizada e um número de *EigenFaces* igual a vinte.

Utilizando redes neurais do tipo *Multilayer Perceptron* (MLP) treinada com o algoritmo Rprop, Martins (2011) apresentou uma abordagem para a resolução de reconhecimento facial. O objetivo deste trabalho foi demonstrar a eficiência de redes MLP aplicada ao reconhecimento de faces. Os testes foram realizados levando em consideração variações nas estruturas de camada da rede neural, onde foi avaliado o desempenho para cada variação, a fim de verificar qual apresenta melhores resultados. Os resultados de acurácia obtidos ficaram entre 68.5% e 75%.

3. Fundamentação Teórica

Para que seja possível a compreensão do trabalho proposto, esta seção apresenta uma introdução e abordagem de alguns conceitos que serão utilizados durante a etapa de desenvolvimento do projeto. Portanto serão abordados temas como Inteligência Artificial, *Machine Learning* e Redes Neurais.

3.1. Inteligência Artificial

A inteligência artificial é um campo interdisciplinar que combina a teoria da aplicação de algoritmos de aprendizado de máquina e outros métodos computacionais para permitir que sistemas de computadores tenham a capacidade de aprender e executar tarefas de forma autônoma, simulando o comportamento e modo de pensar de um ser humano (Russel e Norvig, 2022).

Com o avanço contínuo da tecnologia, a Inteligência Artificial foi tomando proporções cada vez maiores e sendo apta a atingir funcionalidades e habilidades que antes eram dadas como apenas questionamentos, se realmente as máquinas um dia iriam conseguir atingir esse nível de capacidade de processamento lógico. Atualmente a IA (Inteligência Artificial) possui diversos campos de estudo na ciência da computação, como aprendizado de máquina (*machine learning*), visão computacional, robótica,

processamento de linguagem natural, entre outros. Alguns desses campos serão evidenciados neste trabalho.

3.2. Machine Learning

Machine learning (ML) é uma subárea da Inteligência Artificial que tem como propósito o desenvolvimento de algoritmos que podem aprender padrões a partir de dados, algoritmos esses que melhoram seu desempenho em uma tarefa específica de acordo com o aumento de seu aprendizado (Murphy, 2012). A ideia central é que ao invés de explicitar instruções, o algoritmo de ML é treinado para identificar padrões nos dados e tomar decisões com base nessas informações.

O processo para treinar algoritmos de ML é padrão, de forma que é feita uma coleta de dados relevantes para o problema em questão. Logo após, o conjunto de dados de entrada é dividido em duas partes, uma para treinamento do algoritmo e outra para teste. Como o próprio nome já diz, o conjunto de treinamento serve para treinar o modelo, enquanto o conjunto de teste é usado para avaliar a precisão do modelo. Depois do modelo ser treinado, ele é usado para realizar previsões em novos dados de entrada, que são advindos do conjunto de testes (Murphy, 2012). O *machine learning* é um campo em constante evolução que está revolucionando a forma como indivíduos e empresas fazem uso de informações para automatizar processos e tomar decisões mais assertivas. Além disso, ele possui diversas aplicações, incluindo reconhecimento de voz, reconhecimento de imagens, diagnóstico médico, sistemas de recomendação e muitas outras.

3.3. Redes Neurais

Redes neurais são um dos principais métodos utilizados na IA para permitir que computadores consigam aprender a realizar tarefas complexas e estruturadas. As redes neurais são modelos matemáticos inspirados no funcionamento do cérebro humano, que são compostos por neurônios interligados distribuídos em diversas camadas que são capazes de aprender a partir de treinamentos exemplificados (Luz, 2019). Elas são um tipo de algoritmo de *machine learning*, uma subárea da IA já abordada neste trabalho, que permite com que computadores aprendam a partir de grandes conjuntos de dados e apresentem resultados superiores em tarefas de reconhecimento de padrões.

Assim como no cérebro humano, as redes neurais são compostas por várias camadas de neurônios interligados, que processam informações de entrada para gerar uma saída. Elas possuem uma camada de entrada que recebe os dados de entrada e repassa para as camadas ocultas, onde será feito o processamento desses dados a fim de repassar para a camada de saída, que entregará o resultado final (Luz, 2019).

É importante destacar que, há vários tipos de redes neurais, sendo elas *feedforward* (MLPs), convolucionais (CNNs) e recorrentes (RNNs), porém apenas as CNNs serão abordadas com profundidade neste trabalho, já que elas foram utilizadas e aplicadas no desenvolvimento do projeto.

3.3.1. Redes Neurais Convolucionais

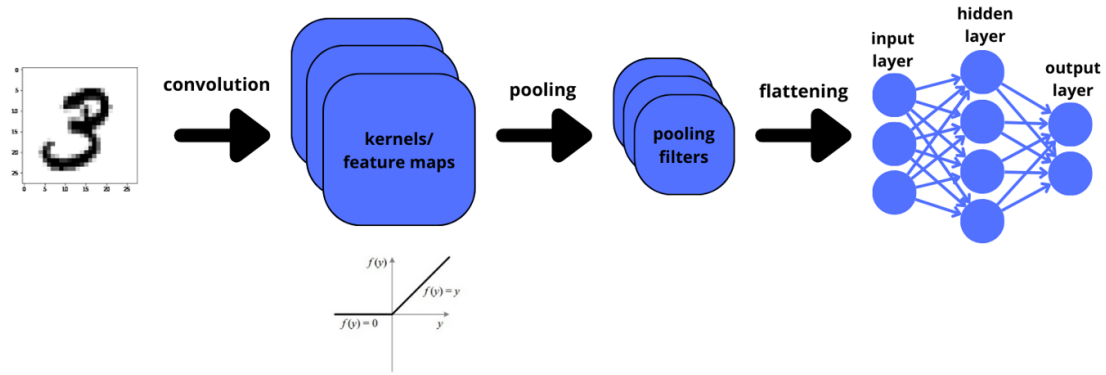
As redes neurais convolucionais são um tipo especial de rede neural que utiliza a operação matemática de convolução para extrair características relevantes dos dados de entrada, como imagens, vídeos e áudios. A principal diferença das CNNs é a etapa de pré-

processamento que ela realiza com os dados antes de efetivamente fornecer os mesmo como entrada para os neurônios da camada de entrada.

A convolução é uma operação matemática que combina os dados de entrada com um conjunto de filtros para gerar mapas de características, que destacam padrões importantes nestes dados. Essa etapa de pré-processamento dos dados é fundamental para que a rede consiga realizar com precisão tarefas como classificação, segmentação ou detecção de objetos em imagens, sendo esse o principal motivo pelo qual as CNNs são as mais utilizadas em reconhecimento facial (Goodfellow, Bengio e Courville, 2016).

A Figura 1 ilustra a arquitetura de uma rede neural convolucional, é possível visualizar o fluxo de processos e camadas que serão citados nesta seção. No exemplo ilustrado a entrada hipotética é a imagem de um número da base de dados MNIST.

Figura 1. Arquitetura de uma rede neural convolucional.



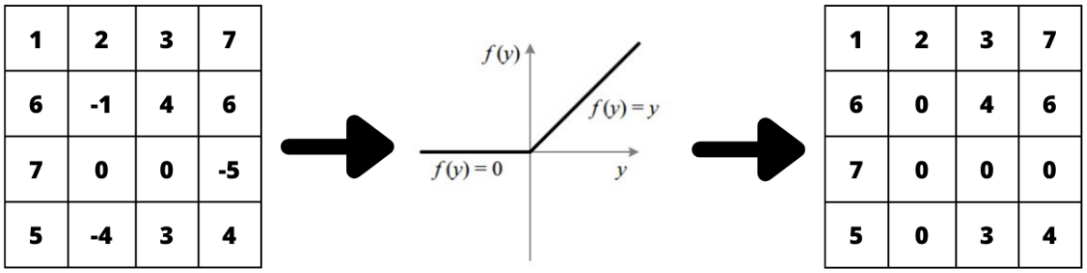
Fonte: Autores.

Esse tipo de rede neural possui três principais tipos de camadas, sendo elas, as camadas convolucionais, camadas de *pooling* e camadas totalmente conectadas. Para uma melhor compreensão do funcionamento dessas redes será feito uma explicação do processo que acontece em cada camada, ressaltando a importância da aplicação de cada método utilizado para melhorar a qualidade do resultado final. Os exemplos utilizados foram baseados em dados de entrada sendo imagens.

Nas camadas convolucionais ocorre o processo de convolução, onde são aplicados diversos filtros ou *kernels* em uma parte da imagem, esses *kernels* são matrizes utilizadas para gerar efeitos. Os *kernels* são aplicados com o intuito de extrair características importantes/relevantes, como bordas, texturas e padrões. Cada *kernel* aplicado gera um mapa de características que também é uma matriz, onde irá conter as representações numéricas das características da imagem (Goodfellow, Bengio e Courville, 2016).

Logo em seguida, uma função de ativação chamada ReLU (*Rectified Linear Unit*) é aplicada para cada valor do mapa de características resultante. Ela é definida matematicamente como $f(x) = max(0, x)$, o que significa que ela irá retornar o valor de x se x for positivo, e caso contrário irá retornar 0. Essa função é importante pois faz com que a rede neural se torne mais eficiente em âmbito computacional, ela ajuda a introduzir a não-linearidade na rede e evitar problemas de desaparecimento de gradiente. A Figura 2 ilustra a utilização da função ReLU em um mapa de características, resultando em um novo mapa.

Figura 2. Aplicação da função ReLU.



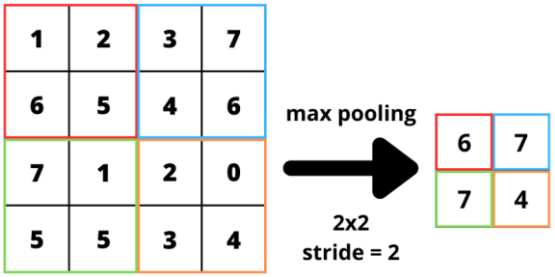
Fonte: Autores.

Em seguida, o mapa de características resultante da aplicação da função ReLU é utilizado como entrada para a camada de *pooling*, onde ocorre o processo de *pooling*. Esse processo consiste em dividir o mapa em regiões, a fim de extrair as características mais importantes de cada uma delas, reduzindo a resolução e dimensão espacial para obter um novo mapa reduzido.

A aplicação do processo ocorre por meio de uma janela deslizante que percorre o mapa de características em passos definidos pelo valor do *stride*, ou seja, se o valor do *stride* for igual a 2, a janela irá deslizar 2 pixels. Para cada região que a janela estiver fazendo-se uso da operação de *max pooling*, o pixel de maior relevância será mantido, ou seja, o pixel de maior valor terá prioridade na região. É importante ressaltar que as dimensões da matriz de *pooling* são um hiperparâmetro, que é definido na construção da rede neural convolucional. Um valor comum para o tamanho dessa matriz é 2x2, o que significa que a janela deslizante será de tamanho 2x2, essa escolha é de senso comum pois permite reduzir a dimensão do mapa de características pela metade em cada operação de *pooling* (Goodfellow, Bengio e Courville, 2016).

Note que o mapa resultante do processo de *pooling* preserva apenas as características mais importantes, isso ajuda a reduzir o número de parâmetros da rede além de torná-la mais eficiente no caso de variações de valores em uma determinada região, melhorando o processo de generalização. A Figura 3 ilustra o processo de *pooling* em um mapa de características, o valor do *stride* é igual a 2 e foi utilizado a operação de *max pooling* (pegar o maior valor).

Figura 3. Ilustração max pooling.

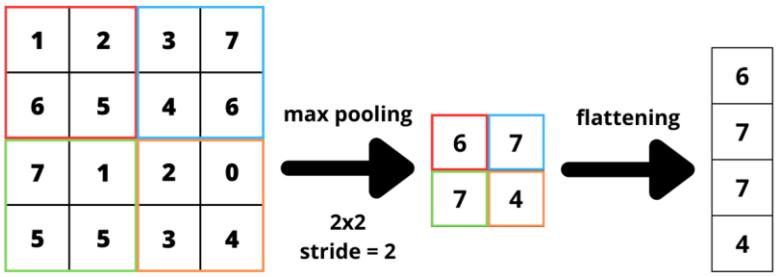


Fonte: Autores.

Por fim, para fornecer a entrada para a próxima camada da rede neural é aplicada a operação de *flattening*, que consiste em transformar o mapa de características reduzido em um vetor unidimensional. O processo de *flattening* de acordo com a Figura 4, permite

que a rede convolucional possa usar o vetor como entrada para as camadas totalmente conectadas, que fazem parte da arquitetura básica da rede.

Figura 4. Ilustração max pooling e flattening.



Fonte: Autores.

4. Materiais e Métodos

Esta seção apresenta os materiais, metodologias e métodos empregados durante o desenvolvimento do presente projeto. Este trabalho trata-se de uma pesquisa aplicada, utilizando procedimentos de cunho experimental e revisões bibliográficas. Para alcançar os objetivos propostos, foram utilizadas algumas etapas e ferramentas de desenvolvimento.

Algumas etapas foram definidas para a realização objetiva e clara do projeto. Primeiramente foi realizado um levantamento bibliográfico para a coleta de materiais de embasamento teórico, paralelamente foi feito um estudo profundo e revisão dos assuntos e temas abordados na fundamentação teórica, fazendo uso de artigos científicos, artigos acadêmicos, livros e publicações de periódicos online.

Em seguida, foi feito um estudo e apuramento de algoritmos e implementações de aprendizado de máquina e processamento de imagens, além da busca e escolha de melhores ferramentas a se utilizar e bibliotecas de código aberto. Posteriormente também foi realizada a escolha das bibliotecas e da ferramenta que seria utilizada para a implementação da interface gráfica do projeto. Por fim deu-se início a fase de desenvolvimento do projeto, que será abordada com mais detalhes na seção seguinte.

O projeto utiliza os seguintes materiais e ferramentas para o seu desenvolvimento:

- Computador de uso próprio com processador AMD, 16GB de memória RAM, 1 TB de HD;
- Webcam para captura de imagens digitais;
- Ambiente de desenvolvimento Visual Studio Code;
- Bibliotecas para processamento de imagens e visão computacional como OpenCV e Dlib;
- Bibliotecas de aprendizado de máquina como TensorFlow, Keras e Sklearn;
- Biblioteca de interface de usuário gráfica como PyQt5;
- Biblioteca de mapeamento objeto-relacional SQL como SQLAlchemy;
- Linguagem de Programação Python v3.8.10.

5. Desenvolvimento

O trabalho possui a finalidade de desenvolver uma ferramenta de controle de presença de alunos por reconhecimento facial. O projeto foi dividido em duas fases de desenvolvimento: realizar o reconhecimento facial de alunos por meio do

desenvolvimento de uma rede neural convolucional (CNN) e desenvolvimento da ferramenta interativa por meio de uma interface de usuário gráfica para controlar as presenças dos alunos, possuindo conexão com um banco de dados relacional para contabilização das mesmas.

5.1. Primeira Fase

Após todo o processo de revisão bibliográfica e aquisição do conhecimento necessário, deu-se início a primeira fase de desenvolvimento: realizar a implementação da rede neural convolucional para o reconhecimento de faces de alunos.

Utilizando a linguagem Python juntamente com a biblioteca OpenCV e algumas bibliotecas integradas, foi implementado um programa para capturar imagens de um aluno específico, e a partir dessas imagens ser possível identificar e recortar a face desse aluno, consequentemente gerando uma nova imagem em escala menor e redimensionada apenas com a face. O aluno precisa apenas ser identificado para obter suas amostras inclusas no conjunto de dados. A partir disso, foi gerado um *dataset* que posteriormente foi utilizado para treinar e testar a rede neural convolucional definida.

Inicialmente nessa primeira fase do projeto o programa foi desenvolvido sendo utilizado por meio de um console, portanto, as funcionalidades obtidas nesta primeira fase fazem parte de um protótipo inicial. Nas próximas subseções, será detalhado cada passo realizado para atingir o objetivo desta primeira fase.

5.1.1. Aquisição da base de dados

O primeiro passo foi realizar a aquisição do conjunto de dados que seria utilizado para o treinamento e teste da rede neural. Para isso, as imagens foram capturadas em dimensões de 640x480 pixels e o intuito era redimensiona-las para 300x300 pixels utilizando um método de interpolação de pixels disponível na OpenCV. Além disso, foi feito o cálculo de um *ratio* por meio da divisão da largura da imagem original pela altura da imagem original. Então a largura final de redimensionamento foi definida com base nesse valor. Portanto a dimensão final da foto foi definida como 400x300, conforme Figura 5.

Figura 5. Imagem redimensionada 400x300.

Fonte: Autores.

O trabalho de diminuir a dimensionalidade da imagem preservando as proporções é um passo importante, para quando posteriormente a face do aluno for detectada, não ocorrer um único redimensionamento muito brusco de proporções, que causaria uma perda maior de informações e detalhes da imagem. Após isso foi aplicado nas imagens redimensionadas um algoritmo de detecção de objetos da biblioteca Dlib chamado HOG (*Histogram of Oriented Gradients*), gerando uma nova imagem. Posteriormente, a nova

imagem obtida foi redimensionada para 100x100 pixels conforme a Figura 6, novamente utilizando o método de interpolação de pixels. Por fim essa imagem é armazenada na base de dados.

Figura 6. Imagem final da face detectada em 100x100.

Fonte: Autores.

5.1.2. Pré-processamento dos dados

Primeiramente foi definida uma proporção para realizar a divisão da base de dados total em duas partes: 15% da base separada para o conjunto de testes e 85% da base separada para o conjunto de treinamento. Foi realizada uma conversão de escala de cores das imagens adquiridas. As fotos faciais dos alunos foram previamente salvas em escala RGB (*Red Green Blue*), e posteriormente, convertidas para escala de cinza. Esse primeiro pré-processamento foi realizado através da biblioteca OpenCV que fornece um método COLOR_RGB2GRAY de conversão de escala de cores. Conforme pode ser visto na Figura 7.

Figura 7. Imagem convertida para escala de cinza.

Fonte: Autores.

Após isso, procedeu-se com a normalização dos dados e codificação dos nomes dos alunos. A codificação das *labels*, ou nesse caso dos nomes dos alunos, ao treinar uma rede neural é um método bastante utilizado, já que as redes neurais geralmente usam dados numéricos como entrada.

5.1.3. Definição da rede neural convolucional

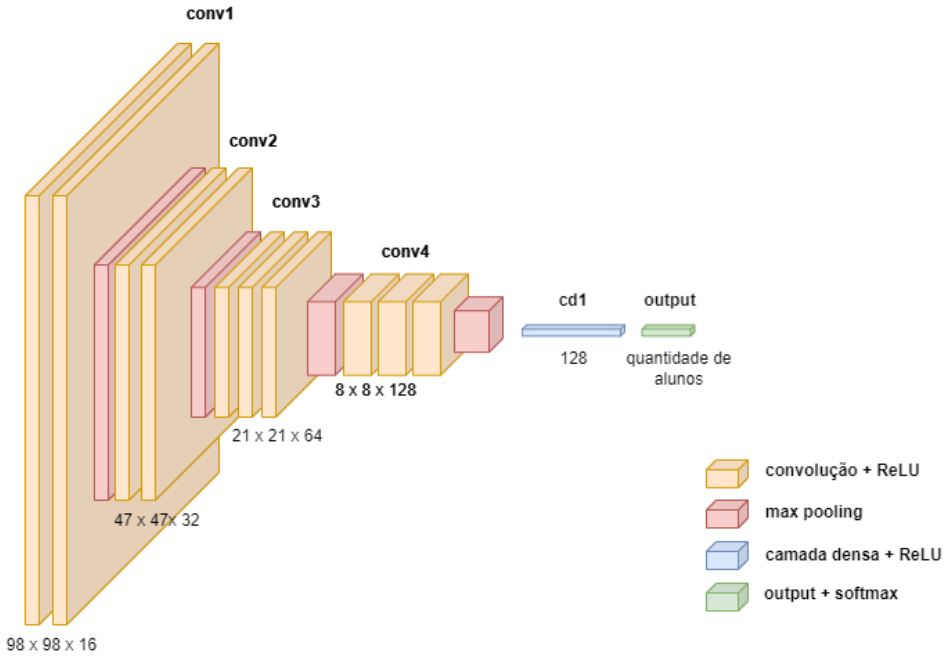
A rede neural convolucional foi implementada utilizando o módulo Keras da biblioteca TensorFlow, e a estrutura de camadas do modelo foi definida de forma empírica. O modelo é composto por quatro camadas de convolução, uma camada densa e uma camada de saída. É importante ressaltar que, todas as camadas de convolução foram definidas com um tamanho de kernel de 3x3 e utilizam a função de ativação ReLU. Além disso, foi aplicado um *Max Pooling* usando o tamanho de janela 2x2 com um *stride* de 1.

A primeira camada de convolução possui 16 filtros. A segunda camada de convolução possui 32 filtros. A terceira camada de convolução possui 64 filtros. Por fim, a quarta e última camada de convolução possui 128 filtros. Note que, as camadas de convolução fazem parte da etapa de extração de características das imagens, extraindo informações e padrões desse conjunto de dados. Além disso, foi adicionado ao modelo

uma camada densa com 128 neurônios utilizando a função de ativação ReLU. Por fim foi adicionado a camada de saída da rede neural.

É válido destacar que, a quantidade de neurônios da camada de saída da rede neural é definida baseada na quantidade de classes obtidas em um dado problema. Neste projeto a quantidade de classes é a quantidade de alunos existentes na base de dados. A Figura 8 apresenta uma representação visual do modelo criado, a fim de se obter uma melhor visualização e compreensão do modelo implementado.

Figura 8. Representação visual do modelo da rede neural.



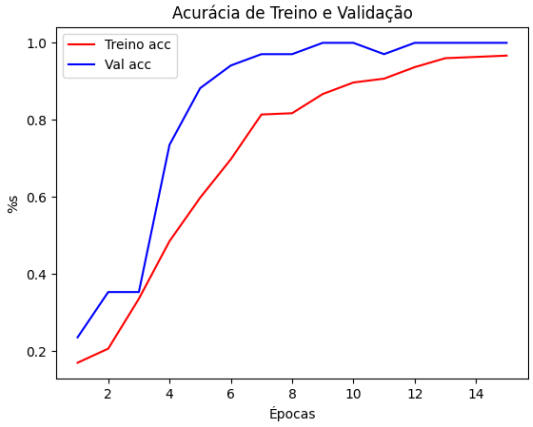
Fonte: Autores.

5.1.4. Treinamento da rede neural convolucional

O treinamento da rede neural tem como objetivo ajustar os pesos da rede com base nos dados de treinamento, a fim de minimizar a função de perda, aprender e aprimorar o desempenho da rede. Essa etapa é fundamental para permitir que a rede neural aprenda com os dados fornecidos e adquira a capacidade de generalizar para novos dados, obtendo um bom desempenho. O modelo foi treinado durante a quantidade de épocas definida como 15. As épocas referem-se a uma passagem completa pelos dados de treinamento, ou seja, todos os exemplos desse conjunto são apresentados ao modelo, e os pesos são atualizados com base nos erros cometidos na previsão.

Através da biblioteca Matplotlib juntamente com os dados do histórico do modelo gerados durante o treinamento, é possível gerar os gráficos de acurácia e *loss* da rede neural. Através da Figura 9, é possível visualizar o gráfico de acurácia do modelo durante as 15 épocas de treinamento. Nota-se uma melhora crescente da assertividade do modelo durante o treinamento. A linha vermelha refere-se à acurácia do modelo no conjunto de treinamento e a linha azul refere-se à acurácia do modelo no conjunto de validação (teste). Percebe-se que, aproximadamente a partir da época 12 o modelo se estabiliza a uma precisão de aproximadamente 96%.

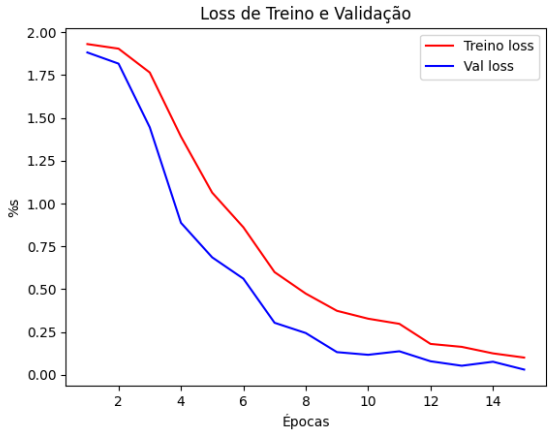
Figura 9. Gráfico de acurácia do modelo treinado.



Fonte: Autores.

Em contrapartida, de acordo com a Figura 10 também é possível visualizar o gráfico de perda de assertividade, tanto para o conjunto de treinamento quanto para o conjunto de validação.

Figura 10. Gráfico de loss do modelo treinado.



Fonte: Autores.

5.2. Segunda Fase

A biblioteca escolhida para desenvolvimento da interface gráfica do projeto foi a PyQt, juntamente com o recurso Qt Designer. O PyQt é uma biblioteca Python de código aberto que fornece uma interface de programação de aplicativos para a biblioteca Qt. A Qt é uma biblioteca multiplataforma para desenvolvimento de interfaces gráficas de usuário (GUIs), que é usada em uma ampla variedade de aplicações, incluindo software de desktop, web e mobile. O PyQt também é altamente flexível e pode ser usado para criar desde simples interfaces de linha de comando até interfaces mais complexas com muitos recursos. Essa ferramenta conta com alguns recursos como suporte a uma ampla variedade de *widgets*, incluindo botões, caixas de seleção, caixas de texto, eventos que permitem que as aplicações respondam a ações do usuário, animações, gráficos e multimídia.

Além disso, o Qt Designer é uma ferramenta de design gráfico que auxilia o desenvolvimento de GUIs usando a biblioteca Qt. Ele permite a criação de GUIs

arrastando e soltando *widgets*, o que torna o processo de desenvolvimento mais rápido e prático. É uma ferramenta que permite que as GUIs criadas com ele possam ser executadas em qualquer plataforma incluindo Windows, macOS, Linux, Android e iOS. Essas ferramentas foram escolhidas por se tornarem uma ótima opção de ferramenta para desenvolvimento de GUIs de maneira rápida, simples e fácil.

5.2.1. Adaptação das funcionalidades

O PyQt tem seu próprio loop de eventos, também conhecido como "loop principal do Qt". Este loop é responsável por monitorar eventos de entrada (pressionamento de teclas, cliques do mouse, entre outros) e executar *callbacks* associadas a esses eventos. Quando você inicia um aplicativo PyQt, o loop de eventos é iniciado automaticamente.

Compreender como o PyQt lida com o loop principal é importante para o desenvolvimento de interfaces gráficas responsivas. PyQt, como outras bibliotecas GUI em Python, usa um modelo de programação chamado “orientado a eventos”. Neste modelo, a execução do programa é controlada pela ocorrência de eventos como cliques do mouse e pressionamentos de teclas.

No entanto, executar operações de longa execução ou de bloqueio diretamente no código executado no mesmo thread que o loop principal pode fazer com que a UI (*User Interface*) congele e pare de responder. Isso ocorre porque o loop de eventos é bloqueado por essas operações demoradas e não pode processar outros eventos.

Anteriormente, o programa desenvolvido na primeira fase do projeto tinha funcionamento via console, as operações poderiam ser executadas sequencialmente, sem a necessidade de considerar a concorrência e responsividade que uma interface gráfica demanda. Ao migrar para uma ferramenta com interface gráfica utilizando o PyQt, foi necessário realizar adaptações no código. Isso se deve ao fato de que, ao executar o processamento de tarefas demoradas diretamente no thread principal, a interface do usuário ficava bloqueada, comprometendo a experiência do usuário.

Para contornar esse problema, foi essencial reestruturar o código para incorporar o uso de threads separadas para algumas tarefas como o treinamento da rede neural e barras de carregamento na UI. Essa abordagem permite que as operações demoradas sejam executadas em segundo plano, em um thread distinto, enquanto o loop principal permanece livre para processar eventos de entrada e manter a interface responsiva.

Além disso, nesta segunda fase, como chave de identificação e cadastro de um aluno na ferramenta, foi utilizado o prontuário ao invés do nome (chave definida no protótipo inicial da primeira fase).

5.2.2. Interface e Contabilização de Presenças

A ferramenta com interface gráfica de usuário desenvolvida na segunda fase possui todas as funcionalidades anteriormente implementadas na primeira fase, além de incluir funcionalidades adicionais relacionadas à contabilização de presenças dos alunos identificados nas imagens fornecidas como entrada para o modelo.

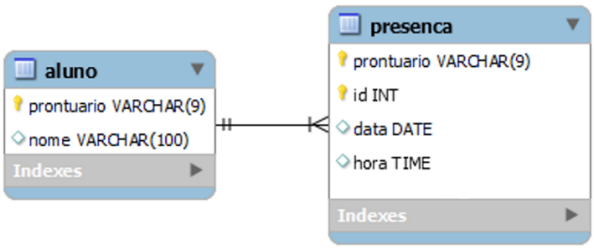
Através da Figura 11 é possível visualizar a tela principal da ferramenta desenvolvida.

Figura 11. Tela principal da ferramenta desenvolvida.

Fonte: Autores.

A funcionalidade de contabilizar presenças quando utilizada abre uma tela para iniciar um programa de captura de imagem, no momento em que é realizada uma captura o algoritmo implementado procura o rosto de alunos conhecidos previamente adicionados ao conjunto de dados, após a identificação dos rostos dos alunos uma conexão com um banco de dados é feita, relacionando o aluno identificado com uma tabela de presenças, realizando a contabilização de presenças para cada aluno. A Figura 12 representa a relação de aluno para presenças.

Figura 12. Modelo lógico do banco de dados utilizado.



Fonte: Autores.

6. Considerações Finais e Resultados

Este trabalho apresenta a implementação de uma ferramenta para controle de presenças de alunos por meio de um algoritmo para reconhecimento facial, baseado em redes neurais convolucionais. Durante o processo de desenvolvimento foram utilizados, principalmente, diversos conceitos importantes de inteligência artificial, *machine learning*, processamento de imagens e visão computacional. Além disso, foi de extrema relevância a utilização de bibliotecas como OpenCV, Dlib, PyQt e alguns métodos do módulo Keras da biblioteca TensorFlow, disponíveis em conjunto com a linguagem de programação Python.

Diversos recursos presentes como captura de imagens, detecção facial, interface gráfica e definição/treinamento de redes neurais só foram possíveis devido a estes recursos. Foram realizadas algumas atividades como, aquisição de base dados, pré-processamento de imagens, definição e treinamento da rede neural convolucional, contabilização das presenças através do reconhecimento de rostos e desenvolvimento de uma interface de usuário gráfica. A ferramenta desenvolvida na segunda fase do projeto

é fácil de usar, entender e adaptar, sendo uma excelente solução para o controle de presenças dos alunos por reconhecimento facial.

Alguns desafios foram encontrados durante o desenvolvimento da primeira fase do projeto como tamanho da base de dados, processamento de imagens em escalas de cores diferentes e redimensionamento adequado de imagens. Foi possível perceber a importância de alimentar as redes neurais convolucionais com uma grande quantidade de dados, uma vez que sua efetividade e eficiência são significativamente reduzidas quando operam com poucos dados. Outro ponto importante é a atenção que deve ser dada ao processar imagens de diferentes escalas de cores, sendo crucial manter sempre uma coerência no uso de conversão dessas escalas. Quanto ao redimensionamento das imagens, é uma tarefa que requer cautela, pois a diminuição das dimensões pode resultar na perda de informações relevantes para alcançar resultados satisfatórios.

É importante salientar que o reconhecimento facial através do método escolhido utilizando os mapas de características das imagens de rostos como entrada para a CNN, apresentou diversos fatores de influência negativa na assertividade do modelo final, fatores esses como iluminação, distância dos indivíduos da câmera, mudanças visuais do indivíduo (com óculos/sem óculos) e qualidade da câmera utilizada. Devido a isso, foi possível notar que o método utilizado talvez não seja a melhor solução para o problema, mesmo tendo resultados consideravelmente satisfatórios.

O reconhecimento facial não é uma tarefa simples e elementos característicos de uma pessoa podem variar como cabelo, óculos e barba, o que dificulta o processo de reconhecimento facial, consequentemente ocorrendo uma diminuição na acurácia do modelo. Para um projeto futuro recomenda-se a utilização das características físicas/pontos faciais como entrada para a CNN.

Os resultados obtidos foram satisfatórios e mostram que é coerente a utilização de redes neurais convolucionais (CNNs) para a realização de reconhecimento facial. O modelo de CNN definido e treinado na primeira fase do projeto, mostrou uma taxa de acurácia de ≅ 96% e um *loss* de 0.14 quando testado com o conjunto de dados de teste. Quando testado com fotos selecionadas arbitrariamente obteve um desempenho mediano comparado ao conjunto de teste. Foram testadas 10 fotos selecionadas manualmente e nunca antes vistas pela rede neural, entre as 10 fotos fornecidas a rede neural obteve um aproveitamento (acurácia) ≅ 70%. De acordo com a Figura 13 é possível visualizar um exemplo de predição realizada pelo modelo desenvolvido.

Figura 13. Predição realizada pelo modelo.

Fonte: Autores.

Portanto pode-se concluir que, essa estrutura de rede neural (CNN) mostra uma grande capacidade e eficiência ao trabalhar com um conjunto de dados amplo e adequado,

além de oferecer recursos suficientes ao trabalhar com processamento de imagens, possibilitando a resolução de diversos problemas tais como a complexidade do reconhecimento facial.

Referências

DINIZ, Fabio; SILVA, Thiago; ALENCAR, Francisco. **Um estudo empírico de um sistema de reconhecimento facial utilizando o classificador KNN**. Revista Brasileira de Computação Aplicada, v. 8, n. 1, p. 50-63, 30 abr. 2016. Disponível em: <http://seer.upf.br/index.php/rbca/article/view/5227>. Acesso em 22 mar. 2023.

GOODFELLOW, Ian; BENGIO, Yoshua; COURVILLE, Aaron. **Deep Learning**. 1. ed. Cambridge, MA: MIT Press, 2016.

KRIZHEVKSY, Alex; SUTSKEVER, Ilya; HINTON, Geoffrey E. **ImageNet Classification with Deep Convolutional Neural Networks**. In: Proceedings of the 25th International Conference on Neural Information Processing Systems, Lake Tahoe, Nevada, USA, 2012.

LUZ, Gabriel. **Redes Neurais Artificiais**. 2019. Disponível em: <https://medium.com/gabriel-luz/redes-neurais-artificiais-f95c716a32f3>. Acesso em: 24 abril 2024.

MARTINS, L. F. M. **Reconhecimento facial utilizando redes neurais**. Orientador: Rodolfo Barros Chiaramonte. 2011. TCC (Graduação) - Curso de Bacharelado em Ciência da Computação, UNIVEM, MARÍLIA, SP 2011. Disponível em: <https://aberto.univem.edu.br/handle/11077/360>. Acesso em: 22 mar. 2023.

MURPHY, Kevin P. **Machine Learning: A Probabilistic Perspective**. 2. ed. Cambridge, MA: MIT Press, 2012.

RUSSEL, Stuart J.; NORVIG, Peter. **Inteligência artificial: uma abordagem moderna**. 4. ed. Rio de Janeiro: Elsevier, 2022. 1.144 p.

SANTOS, Vagner Vieira dos. **Sistema de reconhecimento facial com base em técnicas de aprendizado de máquina**. Orientador: Daniel da Conceição Pinheiro. 2022. [12] f. Trabalho de Conclusão de Curso (Bacharelado em Engenharia da Computação) – Campus Universitário de Tucuruí, Universidade Federal do Pará, Tucuruí, 2022. Disponível em: <https://bdm.ufpa.br:8443/jspui/handle/prefix/3983>. Acesso em: 17 mar. 2023.