UNIVERSIDADE FEDERAL DO MARANHÃO

Curso de Ciência da Computação

Paulo Victor Borges Oliveira Lima

Detecção Automática de Emoções no Aprendizado de Algoritmos

São Luís

2023

Paulo Victor Borges Oliveira Lima

Detecção Automática de Emoções no Aprendizado de Algoritmos

Monografia apresentada ao curso de Ciência da Computação da Universidade Federal do Maranhão, como parte dos requisitos necessários para obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Prof. Dr. Carlos de Salles Soares Neto

São Luís

2023

Ficha gerada por meio do SIGAA/Biblioteca com dados fornecidos pelo(a) autor(a).
Diretoria Integrada de Bibliotecas/UFMA

Victor Borges Oliveira Lima, Paulo.
Detecção Automática de Emoções no Aprendizado de Algoritmos / Paulo Victor Borges Oliveira Lima. - 2023.
33 p.

Orientador(a): Carlos de Salles Soares Neto.
Curso de Ciência da Computação, Universidade Federal do Maranhão, São Luís, 2023.

1. Aprendizado Baseado em Problemas. 2. Deep Learning. 3. Detecção de Emoções. I. de Salles Soares Neto, Carlos. II. Título.

Paulo Victor Borges Oliveira Lima

Detecção Automática de Emoções no Aprendizado de Algoritmos

Monografia apresentada ao curso de Ciência da Computação da Universidade Federal do Maranhão, como parte dos requisitos necessários para obtenção do grau de Bacharel em Ciência da Computação.

Prof. Dr. Carlos de Salles Soares Neto

Prof. Dr. Alexandre Cesar Muniz De Oliveira

Profa. Dra. Li-Chang Shuen Cristina Silva Sousa

Prof. Dr. Tiago Bonini Borchartt

São Luís
2023

Agradecimentos

Primeiramente, quero expressar minha mais profunda gratidão a Deus, por me conceder saúde, força e sabedoria para percorrer essa jornada acadêmica. Sua presença constante foi minha fonte de inspiração e motivação durante os momentos de desafio e incerteza.

Gostaria de expressar minha profunda gratidão ao meu orientador, cuja expertise, compreensão e paciência adicionaram muito à minha experiência acadêmica. Sua orientação e apoio constantes foram inestimáveis para mim, não apenas durante a realização deste trabalho, mas também ao longo de todo o meu percurso acadêmico. Agradeço por ter me desafiado a pensar de maneira crítica e aprofundada sobre o meu trabalho, por ter me incentivado a buscar sempre a excelência e por ter me ensinado a transformar obstáculos em oportunidades de aprendizado. Sua paixão pela pesquisa e pelo conhecimento foi uma fonte de inspiração constante. Agradeço por ter dedicado seu tempo e esforço para me ajudar a desenvolver minhas habilidades e aprimorar meu trabalho. Sua orientação foi essencial para a realização deste projeto e seu impacto em minha formação acadêmica e profissional é imensurável.

Agradeço à minha família, meu porto seguro, por seu amor incondicional e apoio incansável. Aos meus pais, que sempre acreditaram em mim e me incentivaram a seguir meus sonhos, mesmo quando pareciam distantes e inatingíveis. Aos meus irmãos, por sua compreensão e paciência durante os momentos em que tive que me ausentar para me dedicar aos estudos. A cada um de vocês, minha eterna gratidão.

Aos meus amigos, que se tornaram uma segunda família para mim, agradeço por estarem ao meu lado em cada passo desta jornada. Vocês tornaram os momentos difíceis mais suportáveis e os momentos felizes ainda mais memoráveis. Suas palavras de encorajamento, risos compartilhados e ombros para chorar foram essenciais para a conclusão deste trabalho.

A todos que, de alguma forma, contribuíram para a realização deste trabalho, meu sincero agradecimento. Cada um de vocês desempenhou um papel importante nesta conquista e, por isso, compartilho com vocês a alegria deste momento.

Por fim, agradeço a todos que acreditaram em mim. Este trabalho é o resultado de muita dedicação, esforço e perseverança, mas também é o resultado do amor, apoio e fé que recebi de todos vocês.

Obrigado.

"Não fiques em terreno plano.
Não subas muito alto.
O mais belo olhar sobre o mundo
Está a meia encosta."

Friedrich Nietzsche, em *"A Gaia Ciência"*

Resumo

O aprendizado baseado em problemas é uma metodologia vista recorrentemente na literatura para o ensino de algoritmos e estruturas de dados. Há inúmeras plataformas de juízes online disponíveis, onde o aluno resolve problemas computacionais e seus códigos são corrigidos automaticamente comparando se a resposta de saída condiz com o gabarito esperado para aquele problema. Apesar da popularidade, há poucos trabalhos que associam o emprego de técnicas de detecção de emoções como ferramenta de diagnóstico para avaliar a qualidade desses problemas, conduzindo a uma análise mais específica sobre os sucessos, frustrações e alegrias de alunos nessa tarefa. Neste artigo é apresentada uma técnica que define uma linha temporal descrevendo o passo-a-passo de um aluno ao resolver um problema computacional, criando anotações sobre os instantes de tempo em que sentimentos foram percebidos em todas essas etapas, desde a leitura do problema, até a codificação e envio da resposta do juiz online. Nesse contexto, uma contribuição deste artigo é o desenvolvimento de uma API para detecção de emoções em expressões faciais, utilizando técnicas de Deep Learning e redes neurais convolucionais. A API é acessível e escalável via endpoints HTTP e permite a integração fácil em aplicações de terceiros. Os primeiros resultados mostram que a detecção de emoções em tempo real pode ser uma ferramenta valiosa para melhorar a experiência do usuário em plataformas de juízes online. A técnica permite estratificar o diagnóstico dos problemas, analisando se há presença ou predominância de sentimentos indesejáveis em qualquer uma das etapas, seja ela de leitura, codificação ou submissão da solução.

Palavras-chave: Inteligência Artificial, Deep Learning, Detecção de Emoções.

Abstract

Problem-based learning is a methodology seen repeatedly in the literature for teaching algorithms and data structures. There are numerous online judge platforms available, where the student solves computational problems and their codes are automatically corrected by comparing whether the output response matches the expected feedback for that problem. Despite its popularity, there are few works that associate the use of emotion detection techniques as a diagnostic tool to assess the quality of these problems, leading to a more specific analysis of the successes, frustrations and joys of students in this task. This article presents a technique that defines a timeline describing the step-by-step of a student when solving a computational problem, creating notes about the instants of time in which feelings were perceived in all these stages, from reading the problem, to coding and sending the judge's response online. In this context, a contribution of this article is the development of an API for detecting emotions in facial expressions, using Deep Learning techniques and convolutional neural networks. The API is accessible and scalable via HTTP endpoints and allows for easy integration into third-party applications. The first results show that real-time emotion detection can be a valuable tool to improve the user experience on online judge platforms. The technique makes it possible to stratify the diagnosis of problems, analyzing whether there is a presence or predominance of undesirable feelings in any of the stages, be it reading, coding or submission of the solution.

Keywords: Artificial Intelligence, Deep Learning, Emotion Detection.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

CC	*Ciência da Computação*
UFMA	*Universidade Federal do Maranhão*

Sumário

1 Introdução

A dificuldade de aprender algoritmos por meio da resolução de questões reside na necessidade de compreender e aplicar conceitos teóricos de forma prática (FALCKEMBACH; ARAUJO, 2006). É preciso desenvolver habilidades de análise, síntese, abstração e pensamento computacional para traduzir problemas complexos em soluções estruturadas passo a passo. Além disso, o domínio da sintaxe e da lógica de programação também é fundamental. Portanto, a prática consistente e o apoio de plataformas e metodologias educacionais adequadas são essenciais para superar essa dificuldade e aprimorar a capacidade de resolver questões algorítmicas.

Uma técnica utilizada é o Aprendizado Baseado em Problemas (Problem-Based Learning - PBL) (SCHMIDT; ROTGANS; YEW, 2011), que é uma metodologia pedagógica que tem sido amplamente utilizada no ensino de algoritmos. No PBL, os alunos aprendem por meio da resolução de problemas complexos e reais, o que promove o desenvolvimento de habilidades de pensamento crítico e a aplicação prática de conceitos de algoritmos.

Com a crescente disponibilidade de plataformas de juízes online, iniciantes em programação possuem acesso a um grande número de problemas computacionais para resolver. Embora isso possa oferecer muitas oportunidades para a prática e o aprendizado, pode ser algo "assustador"para alguns alunos, que podem se perder em meio à quantidade de problemas disponíveis e ter dificuldade em escolher os mais adequados para o seu nível de habilidade.

Além disso, avaliar a qualidade desses problemas computacionais é uma tarefa complexa. Embora seja possível ter dados sobre o percentual de acertos em relação a erros, que ajuda a identificar se um problema é fácil ou difícil, avaliar se um problema é bem recebido pelos alunos é mais desafiador. Uma maneira de abordar isso é usando a detecção e análise de expressões faciais para obter informações sobre a experiência emocional dos alunos ao resolver os problemas.

A detecção e análise de expressões faciais é uma área de pesquisa muito ampla, com aplicações que vão desde a interação humano-computador até a educação. Normalmente, envolve duas etapas principais: a detecção de rosto e o reconhecimento de emoção. A primeira consiste na identificação e localização de rostos humanos em imagens ou vídeos, fazendo-se uso de técnicas de visão computacional, como classificadores em cascata baseados em recursos de Haar (CHO et al., 2009).

Uma vez detectado o rosto, a próxima etapa é reconhecimento da emoção, que envolve a classificação da expressão facial em uma das várias categorias emocionais. Com

frequência, isso é realizado usando-se técnicas de aprendizado profundo, como redes neurais convolucionais (CNNs) (BHARATI; PRAMANIK, 2020).

Nesse cenário, vislumbrou-se a oportunidade de se desenvolver uma API (ASSIS,) capaz de processar imagens em sequência e detectar emoções em imagens que possa ser facilmente acoplada a plataformas adaptadas ao contexto do aprendizado baseado em problemas. Isso permite que a detecção de emoções seja integrada em várias aplicações e plataformas, sem interferir no funcionamento principal dessas aplicações.

No entanto, a implementação de tal API apresenta seus próprios desafios, principalmente em termos de escalabilidade. Com um grande número de alunos potencialmente usando a plataforma ao mesmo tempo, a API precisa ser capaz de processar um grande volume de solicitações de forma eficiente.

Neste artigo, é apresentada uma técnica que combina a detecção de emoções com a análise de linha do tempo para avaliar a experiência emocional dos alunos ao resolver problemas de programação. Em outros termos, o conceito central é modelar o juiz online tradicional como um sistema multimídia que sincroniza as diferentes etapas da resolução das questões com anotações sobre os sentimentos percebidos nos usuários. Essa abordagem pode fornecer discussões valiosas sobre o processo de aprendizado e ajudar a melhorar a eficácia do ensino de algoritmos.

2 Fundamentação Teórica

A pesquisa intitulada "Facial Emotion Recognition Using Transfer Learning in the Deep CNN"(AKHAND et al., 2021) apresentou uma estratégia de aprendizado transferencial que emprega Redes Neurais Convolucionais (CNNs) profundas para identificar emoções faciais. O estudo utilizou a base de dados FER2013, composta por aproximadamente 35.000 imagens de rostos categorizados em sete emoções distintas: raiva, repulsa, medo, felicidade, tristeza, surpresa e neutro. A abordagem de aprendizado transferencial envolveu a utilização de uma CNN pré-treinada na base de dados ImageNet para a extração de características, seguida do treinamento de um classificador para reconhecer as emoções faciais presentes na FER2013.

Adicionalmente, o artigo (ZATARAIN-CABADA et al., 2017) discute o desenvolvimento de um sistema de reconhecimento de expressões faciais para um Sistema de Tutoria Inteligente (STI). A proposta do estudo é a criação de um banco de dados de expressões faciais para ser utilizado em um STI capaz de detectar as emoções do estudante e fornecer um feedback personalizado. Para atingir esse objetivo, a equipe de pesquisa capturou as expressões faciais de 26 voluntários enquanto assistiam a vídeos educativos, utilizando uma câmera. Posteriormente, desenvolveram uma rede neural capaz de identificar seis emoções básicas - alegria, tristeza, raiva, medo, surpresa e repulsa - com base nas expressões faciais registradas.

Além disso, o ensino de algoritmos pode ser aprimorado mediante a aplicação da abordagem *Problem-Based Learning* (PBL) (ALLEN; DONHAM; BERNHARDT, 2011). O PBL é uma abordagem pedagógica que envolve a apresentação de problemas complexos e reais aos alunos, incentivando-os a buscar soluções de forma autônoma e colaborativa. No contexto do ensino de algoritmos, o PBL pode ser particularmente eficaz, pois os problemas de programação são inerentemente orientados para a solução de problemas. Além disso, o PBL pode ajudar a desenvolver habilidades de pensamento crítico e criativo, bem como a capacidade de trabalhar efetivamente em equipe, todas as quais são habilidades valiosas para os programadores.

Portanto, o presente trabalho foi inspirado pela abordagem de aprendizado transferencial para o reconhecimento de emoções faciais apresentada em "*Facial Emotion Recognition Using Transfer Learning in the Deep CNN*"(AKHAND et al., 2021). A estratégia de utilizar uma CNN pré-treinada para a extração de características e posteriormente treinar um classificador para reconhecer emoções específicas foi fundamental para o treinamento da rede neural e o aumento da sua acurácia.

Além disso, foram incorporadas técnicas do estudo de Zarata-Cabada et al. (ZATARAIN-CABADA et al., 2017), que desenvolveu um sistema de reconhecimento de expressões faciais para um Sistema de Tutoria Inteligente. A ideia de capturar e analisar expressões faciais para fornecer feedback personalizado foi uma influência significativa na presente abordagem.

3 Metodologia

A metodologia desse trabalho envolve duas vertentes principais: o desenvolvimento da API especializada na detecção de emoções e a aplicação da técnica de linha do tempo para mapear a jornada emocional dos alunos ao resolverem problemas computacionais.

3.0.1 Desenvolvimento da API

Para tornar essa rede neural acessível e fácil de usar, foi desenvolvida uma API que expõe seus métodos por meio de *endpoints HTTP*. Um desses *endpoints* é responsável por receber uma imagem codificada em *base64* e retornar a emoção predominante da face na imagem, caso haja uma face presente, conforme descrito na seção 3.0.1.1.

A API foi desenvolvida de modo a ser facilmente acoplada a aplicações de terceiros, permitindo que seja integrada a diferentes contextos e aplicações. Nesse presente estudo, a API é integrada a uma plataforma de ensino de programação, que contém um sistema de correção automática de questões.

A comunicação entre a API e a aplicação externa escolhida funciona da seguinte forma:

- Cliente (React): Este é o ponto de partida. O cliente envia uma requisição POST para o webservice FastAPI com uma imagem codificada em base64.
- Webservice (FastAPI): Este container recebe a requisição do cliente e processa a informação. Após processar a requisição, ele responde com um JSON com a linha do tempo daquela submissão.
- Cliente (React): O cliente recebe a resposta do webservice e envia essa informação para o backend.
- Backend (Node): O backend recebe a informação do cliente e processa os dados conforme necessário. Ele então armazena os dados no banco de dados.
- Banco de dados (MySQL): Este é o destino final dos dados. O backend armazena os dados aqui para uso futuro.

Nesse sentido, a arquitetura do software segue o diagrama exposto na figura 1:

3.0.1.1 Função de detecção

Para prever as emoções das faces detectadas na imagem, a API utiliza o modelo de rede neural pré-treinado apresentado na Seção 3.1.5. Além disso, ele carrega o classificador

Haar cascade, que é um método de detecção de objetos usado para identificar objetos em imagens ou fluxos de vídeo, utilizado, nesse caso, para detecção de faces.

Ao receber a requisição, a API decodifica a imagem, converte o quadro para escala de cinza, detecta faces na imagem usando o método *face cascade* e, em seguida, recorta e redimensiona cada face para 48x48 *pixels*. Este último passo é essencial, uma vez que o modelo é treinado para trabalhar com esse tamanho específico de imagem. Em seguida, são normalizados os valores dos *pixels* e é feita uma previsão de emoção usando o modelo pré treinado.

A API, então, armazena as emoções reconhecidas para futuramente serem retornadas ao fim da resolução da questão, sendo assim, retorna uma linha do tempo das emoções registradas naquela submissão.

3.0.2 Análise de Emoções

A captura da imagem do usuário se inicia a partir do momento em que o mesmo abre uma questão, perdurando até o momento em que é fechada a questão. Alguns momentos

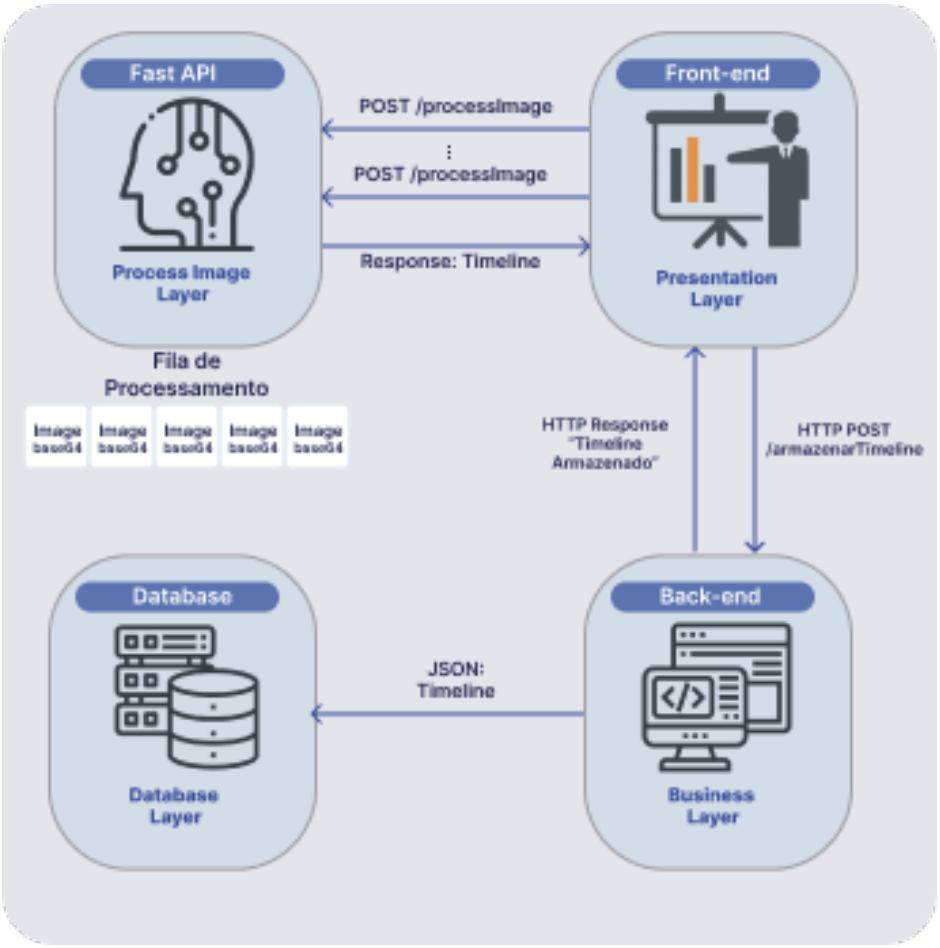


Figura 1 – Imagem representando a comunicação entre as camadas

são mais significativos durante o processo, sendo eles: o início da leitura da questão; o momento em que o aluno finaliza a leitura e começa a escrever o código; o momento de submissão da resolução e resultado do processamento da questão.

Dessa forma, é possível avaliar as emoções dos usuários em diferentes momentos-chave durante a resolução das questões.

Após a análise das emoções do usuário, e de acordo com os possíveis resultados, algumas medidas podem ser tomadas pela plataforma, visando maior imersão, sendo elas:

- felicidade: quando um aluno acerta uma questão difícil, ele pode se sentir feliz e animado. Essa emoção pode ser detectada pela API e usada para fornecer *feedback* positivo ao aluno, como uma mensagem parabenizando-o pela conquista;
- tristeza ou raiva: quando um aluno está tendo dificuldade para resolver uma questão, ele pode ficar frustrado e desanimado. Essa emoção também pode ser detectada pela API e usada para fornecer *feedback* construtivo ao aluno, como sugestões de como abordar a questão de forma diferente ou dicas para superar a dificuldade; e
- surpresa: Quando um aluno é surpreendido por uma questão inesperada ou por um resultado inesperado, ele pode ficar surpreso e curioso. Essa emoção pode ser detectada pela API e usada para fornecer *feedback* interessante ou motivador ao aluno, como uma explicação do porquê da questão ser importante ou de como ela pode ser aplicada na prática.

3.0.3 Análise de Linha do Tempo

Paralelamente ao desenvolvimento da API, também foi implementada uma técnica de linha do tempo para capturar a jornada emocional dos alunos. Esta técnica envolveu o acompanhamento dos alunos ao longo de várias etapas: leitura do problema, codificação da solução e submissão da solução a um juíz *online*. (KARAM, 1994)

O uso desse tipo de técnica visa entender aspectos diferentes sobre a questão, isto é, quais emoções podem ser evocadas durante cada ponto da resolução de um usuário, abstrair e discutir o porquê de tais emoções existirem durante a leitura da questão, talvez por um enunciado confuso ou por um enunciado bem descrito.

Em cada etapa, a API de detecção de emoções foi utilizada para identificar a emoção predominante do aluno. Estes dados foram então anotados em uma estrutura de linha de tempo, proporcionando uma visão detalhada do fluxo emocional do aluno ao longo do processo de resolução do problema.

Dessa forma, essa combinação da tecnologia de detecção de emoções com a análise de linha do tempo pode fornecer *insights* valiosos acerca da experiência do aluno ao resolver

problemas computacionais e, potencialmente, orientar melhorias na forma como esses problemas são apresentados e estruturados.

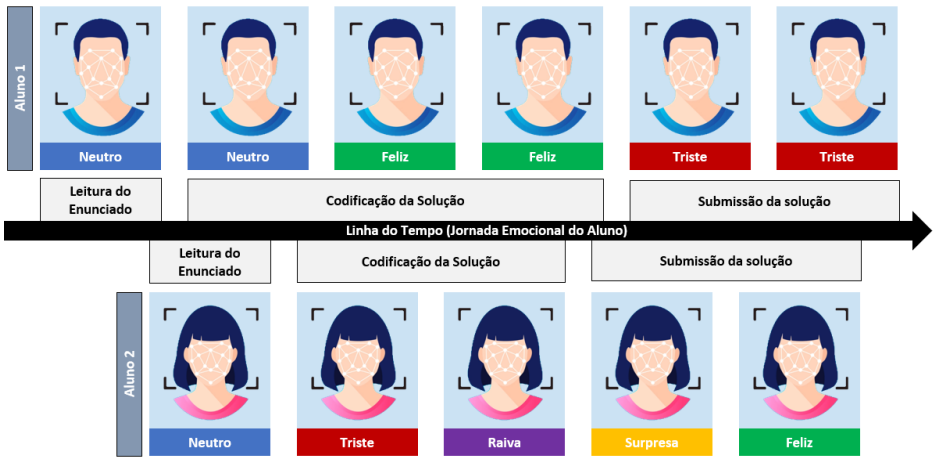


Figura 2 – Exemplo de timeline

3.0.4 Escalabilidade

A arquitetura e fluxo da aplicação podem ser fatores essenciais para garantir o desempenho, escalabilidade e fluidez do processo de adquirir a imagem ou vídeo, processar a imagem, retornar a principal emoção, caso haja uma face na imagem em questão, e armazenar os dados desejados.

É importante notar que o desempenho do processo de detecção de faces pode ser afetado por fatores como iluminação, expressões faciais e posição da cabeça.

A API de detecção de emoções foi desenvolvida usando Python e FastAPI (LATHKAR, 2023), um moderno e rápido (alta performance) *framework web* para construção de APIs, baseado em padrões Python para APIs assíncronas. A escolha do *FastAPI* foi motivada por sua alta velocidade e facilidade de uso, bem como por suas poderosas funcionalidades, como a validação automática de requisições e respostas, a documentação interativa da API e a suporte nativo para concorrência assíncrona.

Um dos principais desafios enfrentados durante o desenvolvimento da API foi garantir que ela não se tornasse um gargalo para a aplicação. Isso foi particularmente desafiador devido à natureza intensiva de dados do processamento de detecção de emoções. Para superar esse desafio, foram implementadas várias otimizações de desempenho, como a utilização de operações assíncronas para evitar bloqueios e a implementação de um sistema de filas para armazenar temporariamente as imagens recebidas por meio das requisições, reduzindo a necessidade de processamento simultâneo.

A implementação de operações assíncronas permitiu que a API processasse várias requisições simultaneamente sem bloquear a execução de outras tarefas. Isso foi

particularmente útil para lidar com a natureza intensiva de dados do processamento de detecção de emoções, pois permitiu que a API processasse uma imagem enquanto aguardava a chegada de novas requisições.

O sistema de filas foi usado para gerenciar o fluxo de imagens recebidas pela API. Quando uma nova requisição chega, a imagem associada é adicionada a uma fila de espera em vez de ser processada imediatamente. Um conjunto de *workers* assíncronos é então responsável por retirar imagens da fila e processá-las em segundo plano. Isso permite que a API seja capaz de lidar com picos de carga, pois as imagens podem ser armazenadas na fila e processadas em um ritmo controlado, em vez de sobrecarregar o sistema com um grande volume de processamento simultâneo.

O Código em Python que descreve esse processamento:

```
def process_image_from_queue():
    while True:
        #Pega uma imagem da Fila
        request_json = image_queue.get()

        #Processa a Imagem
        base64_image = request_json['image']
        img = readb64(base64_image)
        response = classify_emotion(img, model, face_cascade)

        #Imprime a Resposta do Processamento
        emotion = response[0]
        print(emotion)

        #Sinaliza o Fim da Tarefa
        image_queue.task_done()

#Inicia o worker threads
for _ in range(5):
    executor.submit(process_image_from_queue)

@app.post("/enqueueImage")
async def enqueue_image(request: Request):

    request_json = await request.json()
    image_queue.put(request_json)
```

```
return {"status": "Image added to queue"}
```

Além disso, para garantir que a API pudesse lidar com uma grande quantidade de requisições simultâneas sem degradar o desempenho, foram realizados extensos testes de carga e estresse. Esses testes envolveram o envio de um grande volume de requisições, os valores de teste foram de 10 mil requisições simultâneas à API e a avaliação de sua capacidade de processar essas requisições de forma eficiente e sem erros.

Essas otimizações de desempenho permitiram que a API de detecção de emoções funcionasse de maneira eficiente e escalável, mesmo sob condições de alta carga. Isso foi crucial para garantir que a API não se tornasse um gargalo para a aplicação e pudesse fornecer resultados de detecção de emoções em tempo real.

Diante disso, a API retornará, caso a imagem possua um rosto, uma das sete emoções previamente citadas.

3.1 Implementação da Rede Neural

A detecção de emoções por meio de expressões faciais é uma atividade complexa (DHALL et al., 2014) e desafiadora na área de visão computacional. No entanto, os avanços recentes em *Deep Learning* e redes neurais convolucionais possibilitaram a construção de modelos de detecção de emoções precisos e confiáveis, capazes de entender relações complexas e não lineares, assim como as emoções humanas baseadas em expressões faciais.

3.1.1 Aquisição do *dataset* de treinamento

Para realizar o treinamento da rede neural, foi utilizado o *dataset* FER+, que consiste em uma extensão do conjunto de dados FER2013 (GOODFELLOW et al., 2013). Esse conjunto de dados contém imagens faciais rotuladas com seis emoções diferentes: alegria, tristeza, nojo, raiva, surpresa e medo. O FER+ também inclui anotações adicionais que classificam as expressões faciais em termos de sua intensidade, fornecendo informações mais detalhadas sobre o nível de emoção presente em cada imagem.

3.1.2 Pré-processamento dos dados

Foram utilizadas técnicas de pré-processamento, sendo elas *Data Augmentation* (SHORTEN; KHOSHGOFTAAR, 2019), aplicando transformações como rotação, espelhamento horizontal e deslocamento vertical e horizontal às imagens de treinamento; e normalização e limpeza.

3.1.3 Definição da Arquitetura da Rede Neural

A rede neural definida neste trabalho utiliza como base a técnica de transferência de aprendizado (*transfer-learning*) (WEISS; KHOSHGOFTAAR; WANG, 2016), que consiste em aproveitar o conhecimento prévio de uma rede neural treinada em uma tarefa relacionada e reutilizar seus pesos como ponto de partida para uma nova tarefa. Para tanto, uma rede VGG16 (KUSUMA; JONATHAN; LIM, 2020) pré-treinada em um conjunto de dados de classificação de imagens é utilizada como ponto de partida para a detecção de emoções.

Ao utilizar a VGG16 como base, a rede descrita já possui uma série de filtros e características aprendidas, que podem ser transferidas para a nova tarefa de detecção de emoções. Isso permite que a rede seja treinada com menos dados de treinamento, resultando em um treinamento mais rápido e com melhor desempenho em relação a uma rede neural treinada desde o início.

Este modelo é uma Rede Neural utilizada para processamento de imagens, que consiste em diversas camadas convolucionais e de pooling para extrair características relevantes das imagens.

As camadas convolucionais são responsáveis por realizar a convolução entre os filtros e os dados de entrada. O objetivo é detectar características relevantes durante o treinamento da rede, copmo bordas, texturas e padrões. As camadas de *pooling* têm responsabilidade de reduzir a dimensionalidade espacial da saída das camadas convolucionais, porém, mantendo as características mais importantes detectadas. O uso em conjunto dessas camadas permite que o modelo entenda detalhes locais nas camadas inicias e características mais globais nas camadas posteriores.

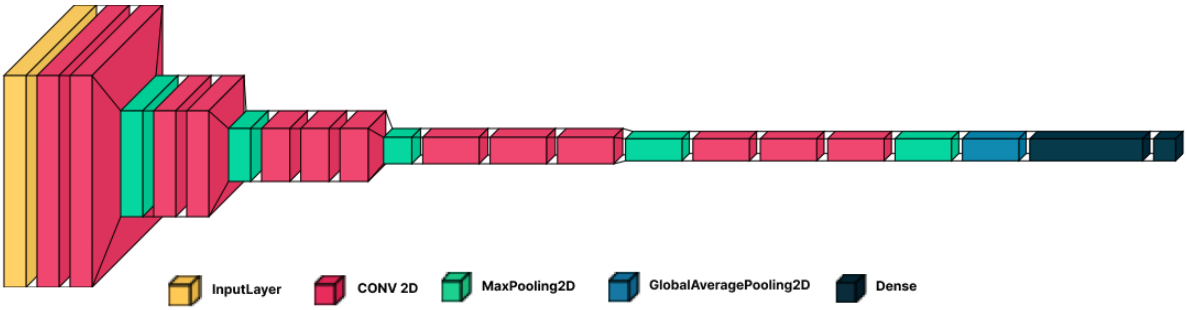


Figura 3 – Arquitetura da rede neural utilizada.

3.1.4 Definição dos hiperparâmetros

Após a definição da arquitetura da Rede Neural, foram realizadas alterações na VGG16 para melhoria do treinamento para a tarefa de detecção de emoções. A arquitetura final da rede utilizada está representada na figura 3

Neste caso, a arquitetura VGG16 está sendo utilizada sem as camadas completamente conectadas e adicionando suas próprias camadas no topo das camadas de convolução, sendo a base convolucional da rede. A saída das camadas de convolução é passada por uma camada *GlobalAveragePooling2D*, que reduz a dimensionalidade da saída e a torna mais gerenciável. A ideia de utilizar essa camada nesse ponto é expor uma representação global resumida dos recursos, tornando a rede menos sensível a variações locais na imagem.

A saída da camada *GlobalAveragePooling2D* é passada por uma camada *Dense* com 1024 neurônios, usando a função de ativação *ReLU*. Essa camada é usada para aprender uma representação mais complexa e abstrata das características extraídas pelas camadas de convolução.

Finalmente, a saída da camada *Dense* é passada por uma camada final *Dense* com 7 neurônios, usando a função de ativação *softmax*. Essa camada é usada para produzir as previsões finais da rede, com 7 neurônios correspondentes às 7 classes de emoções mapeadas.

Dentro das camadas convolucionais, o Kernel se move para a direita. Durante o caminho percorrido, são analisados os objetos da imagem até o fim da sua amplitude. Após finalizar uma linha, ele segue para a próxima linha, seguindo o mesmo fluxo, até que toda imagem seja coberta (ZHAO et al., 2017).

A arquitetura começa com uma camada convolucional. Nesse caso, a camada convolucional é configurada com 32 filtros. Cada filtro possui um tamanho de *kernel* de 3x3, o que significa que ele examina uma vizinhança de *pixels* 3x3 por vez.

Para garantir que a camada convolucional leve em consideração as bordas da imagem, é aplicado um preenchimento (*padding*) chamado *same*. O *same* preenche a imagem de entrada com zeros ao redor das bordas, para que o tamanho da saída da camada convolucional seja o mesmo que o tamanho da imagem de entrada.

Após a operação convolucional, é aplicada uma função de ativação chamada *LeakyReLU*. Essa função introduz não-linearidade ao modelo, permitindo que ele aprenda relações e padrões mais complexos nos dados. A *LeakyReLU* é uma variação da função de ativação *ReLU* (*Rectified Linear Unit*) e possui um parâmetro chamado taxa de inclinação (alpha). Nesse caso, a taxa de inclinação é definida como 0.1, o que significa que, para valores negativos, a função terá uma inclinação suave de 0.1, em vez de ser completamente linear.

Essa sequência de camadas - convolucional seguida por *LeakyReLU* - é comumente usada em CNNs para processar imagens e extrair características relevantes para a tarefa de classificação. Depois, há uma camada de *pooling* com tamanho de *pool* (2,2), que reduz as dimensões da saída da camada anterior pela metade. Esse processo é repetido mais

quatro vezes com camadas convolucionais de 64, 128, 256 e 512 filtros, respectivamente, e tamanhos de pool (2,2). Em seguida, há três camadas convolucionais com 1024 filtros, cada uma seguida por uma *LeakyReLU*.

Ao final das camadas convolucionais, as saídas são achatadas em um vetor unidimensional por uma camada *Flatten*, e então passadas por duas camadas densas com funções de ativação *LeakyReLU*, uma com 256 unidades e outra com 4096 unidades, respectivamente. Essas camadas densas são responsáveis por fazer a classificação das imagens, convertendo os recursos extraídos pelas camadas convolucionais em uma probabilidade de pertencer a cada uma das classes possíveis. A última camada usa uma ativação *softmax* para produzir a probabilidade de saída para cada classe.

3.1.5 Treinamento da Rede Neural

No processo de treinamento do modelo, foram utilizados parâmetros específicos para otimizar a aprendizagem. O número de épocas, que representa a quantidade de vezes que o conjunto de treinamento é passado através da rede, foi definido após tentativas de treinamentos com diferentes valores, o máximo de acurácia foi obtido com 80 épocas. Isso significa que o modelo teve a oportunidade de aprender e ajustar seus pesos internos 80 vezes para cada imagem no conjunto de treinamento.

Foram efetuadas várias tentativas para ajustar os parâmetros e encontrar a melhor configuração que maximizasse a precisão do modelo, alterando os valores de *batch*, *epochs*, otimizadores e taxa de perda.

O tamanho do lote, ou *batch size*, foi definido como 64. Este parâmetro determina o número de exemplos de treinamento considerados antes que o modelo faça uma atualização nos pesos. Um tamanho de lote menor pode levar a uma aprendizagem mais estável, enquanto um tamanho de lote maior pode acelerar o processo de treinamento.

O otimizador escolhido para este modelo foi o Adam (BAE; RYU; SHIN, 2019) com uma taxa de aprendizado de 0.0001. O otimizador Adam é uma escolha popular devido à sua eficiência computacional e à sua eficácia em problemas de aprendizado de máquina de grande escala. A taxa de aprendizado determina o tamanho dos passos que o otimizador dá para alcançar o mínimo global, ou seja, a melhor solução possível.

A função de perda utilizada foi a *binary crossentropy*. Esta função de perda calcula a diferença entre as previsões do modelo e os verdadeiros valores alvo.

Durante o treinamento, o modelo também foi validado em um conjunto de dados de validação. Isso permite monitorar o desempenho do modelo em dados não vistos durante o treinamento e ajustar os parâmetros, se necessário, para evitar o *overfitting* (JABBAR; KHAN, 2015).

Por fim, após o treinamento, o modelo foi avaliado em um conjunto de dados de teste para verificar seu desempenho em dados completamente novos. As previsões do modelo para o conjunto de teste foram salvas para análise posterior. Após o treinamento, algumas métricas foram obtidas, como acurácia, precisão, como mostra a Tabela 1.

Tabela 1 – Medidas de acurácia e precisão para diferentes emoções mapeadas.

Emoção	**Acurácia**	**Precisão**
Alegria	0.92	0.92
Tristeza	0.89	0.87
Nojo	0.60	0.63
Raiva	0.83	0.73
Surpresa	0.87	0.86
Medo	0.73	0.63
Neutro	0.92	0.87

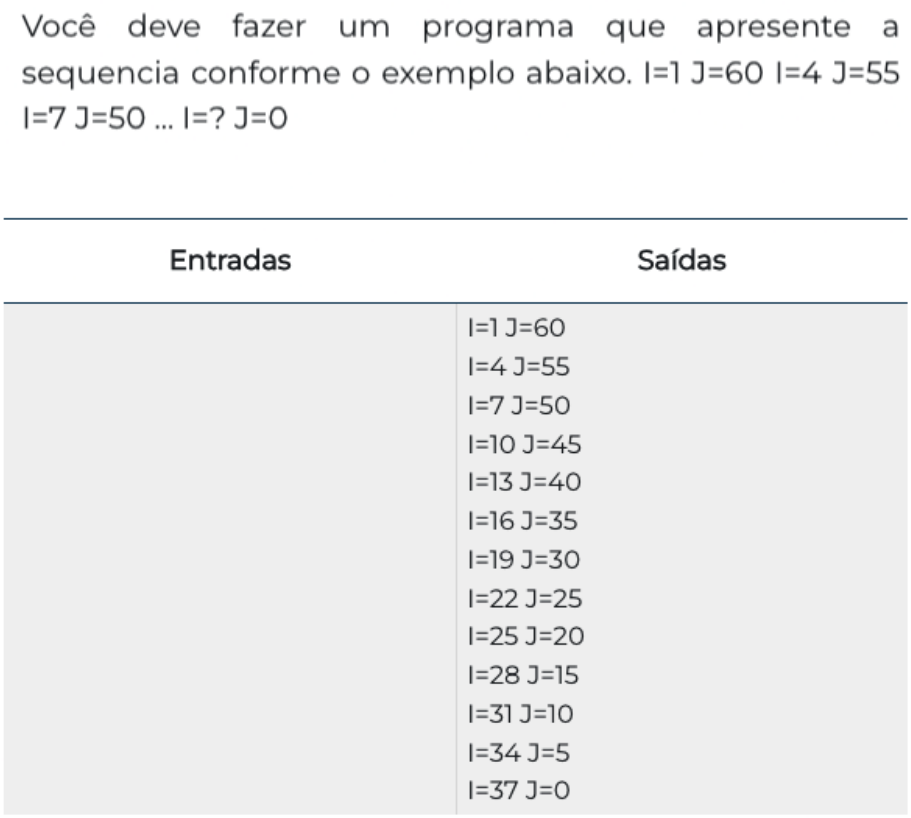
3.2 Estudo de Caso

Um estudo de caso foi realizado em uma plataforma de ensino com distribuição e correção automática de questões de algoritmos, com sistema de juiz online. Para tanto, selecionou-se um grupo de 5 alunos do curso de Ciência da Computação, da instituição (REMOVIDA PARA REVISÃO), com diversos níveis de experiência em programação.

Foram selecionadas 4 questões de lógica de programação com a temática "Laços de Repetição", sendo 1 de nível fácil, 2 de nível intermediário e 1 de nível difícil, exemplificado na figura 4 .

O objetivo do estudo de caso consistiu na avaliação da eficácia da API de detecção de emoções em tempo real. Especificamente, procurou-se compreender como as emoções dos alunos variam durante as diferentes etapas de resolução de um problema de programação, desde a leitura do problema, passando pela fase de codificação, até a submissão da solução. Com a análise de emoções, visou-se identificar padrões que possam contribuir para a melhoria da experiência do usuário na plataforma, bem como fornecer análises sobre como a dificuldade do problema e a experiência do aluno em programação podem influenciar suas emoções durante o processo de resolução de problemas. Os resultados obtidos estão nas tabelas 2, 3, 4 e 5.

Na primeira questão, a leitura mostrou predominância da emoção neutra, evidenciada por 73 ocorrências. As emoções de felicidade e raiva foram consideravelmente menos frequentes, ambas com apenas 5 ocorrências. Durante a etapa de codificação, a emoção neutra continuou predominante, com 197 ocorrências, enquanto a felicidade foi registrada 15 vezes. Na fase de submissão, a emoção neutra manteve-se como a mais frequente, com 20 ocorrências, seguida pela felicidade com 5 ocorrências.



Você deve fazer um programa que apresente a sequencia conforme o exemplo abaixo. I=1 J=60 I=4 J=55 I=7 J=50 ... I=? J=0

Entradas	Saídas
	I=1 J=60 I=4 J=55 I=7 J=50 I=10 J=45 I=13 J=40 I=16 J=35 I=19 J=30 I=22 J=25 I=25 J=20 I=28 J=15 I=31 J=10 I=34 J=5 I=37 J=0

Figura 4 – Exemplo de questão de Laço de Repetição de nível fácil

Tabela 2 – Frequência das emoções detectadas dos usuários na resolução da primeira questão.

Momento	**Emoção**	**Qtd. Emoções**
Leitura	Neutro	73
Leitura	Feliz	5
Leitura	Raiva	5
Codificação	Neutro	197
Codificação	Feliz	15
Submissão	Neutro	20
Submissão	Feliz	5

Tabela 3 – Frequência das emoções detectadas dos usuários na resolução da segunda questão.

Momento	**Emoção**	**Qtd. Emoções**
Leitura	Neutro	84
Leitura	Feliz	4
Leitura	Raiva	1
Leitura	Surpresa	1
Codificação	Neutro	123
Codificação	Feliz	32
Codificação	Raiva	9
Codificação	Surpresa	1
Submissão	Neutro	43

Tabela 4 – Frequência das emoções detectadas na resolução da terceira questão.

Momento	**Emoção**	**Qtd. Emoções**
Leitura	Neutro	40
Leitura	Feliz	66
Leitura	Raiva	4
Codificação	Neutro	80
Codificação	Raiva	8
Codificação	Surpresa	1
Codificação	Feliz	63
Submissão	Feliz	66
Submissão	Neutro	40
Submissão	Raiva	2

Tabela 5 – Frequência das emoções detectadas na resolução da quarta questão.

Emoção	**Momento**	**Qtd. Emoções**
Leitura	Neutro	155
Leitura	Surpresa	4
Leitura	Feliz	2
Codificação	Raiva	7
Codificação	Medo	2
Codificação	Neutro	74
Codificação	Feliz	6
Codificação	Nojo	1
Submissão	Neutro	128
Submissão	Tristeza	60
Submissão	Feliz	74

No caso da terceira questão, a leitura trouxe a emoção de felicidade como a mais frequente, registrando 66 ocorrências. A emoção neutra seguiu com 40 ocorrências e a raiva foi notada em 4 instâncias. A etapa de codificação mostrou novamente a emoção neutra como a mais prevalente, com 80 ocorrências. No entanto, a felicidade ainda se fez presente em 63 ocorrências, seguida pela raiva com 8 ocorrências e uma única instância de surpresa. Durante a submissão, a felicidade novamente se tornou a emoção mais comum, com 66 ocorrências, seguida pelas emoções neutra (40 ocorrências) e raiva (2 ocorrências).

Na quarta questão, durante a leitura, a emoção neutra se destacou com 155 ocorrências, seguida por surpresa (4 ocorrências) e felicidade (2 ocorrências). Na fase de codificação, a emoção neutra, mais uma vez, foi a mais comum, com 74 ocorrências, porém acompanhada de felicidade (6 ocorrências), raiva (7 ocorrências), um pouco de medo (2 ocorrências) e uma instância de nojo. Na fase de submissão, a emoção neutra permaneceu como a mais prevalente (128 ocorrências), com a presença também de emoções como felicidade (74 ocorrências) e tristeza (60 ocorrências).

Em suma, durante as fases de leitura e codificação das questões, a emoção "neutra"parece ser a mais predominante. No entanto, a fase de submissão traz uma gama mais variada de respostas emocionais, com destaque para a felicidade. As emoções negativas, como raiva, medo e nojo, são menos frequentes, sugerindo que a resolução das questões é, em geral, uma experiência neutra ou positiva para os usuários. A presença de surpresa e tristeza durante a submissão pode indicar diferentes reações à avaliação dos resultados.

Durante o momento da leitura, foi possível observar a presença das emoções "Neutro", "Feliz"e "Raiva"em diferentes quantidades. Uma possível relação a ser identificada é a associação entre a emoção "Raiva"e o baixo número de ocorrências em comparação com as outras emoções.

Essa relação pode indicar que a leitura da questão geralmente não desperta sentimentos de raiva nos usuários, enquanto as emoções neutras e felizes são mais comuns nesse contexto. No entanto, é importante ter em mente que a interpretação precisa dos dados pode variar dependendo do contexto do experimento e das características específicas dos participantes.

Na fase de codificação, a emoção predominante é novamente "Neutro", mas também há uma presença significativa da emoção "Feliz". Isso pode indicar que alguns usuários estão mais engajados e satisfeitos durante a codificação da solução.

Na fase de submissão, a emoção predominante é "Feliz", seguida pela emoção "Surpreso"em um número menor de ocorrências. Isso sugere que muitos usuários experimentam uma sensação de satisfação e até surpresa ao submeterem sua solução.

Esses resultados destacam a importância de considerar a dimensão emocional no processo de resolução de problemas, fornecendo dados valiosos para aprimorar a experiência do aluno e a eficácia do ensino.

3.2.1 Privacidade

Para garantir a privacidade do usuário, nenhuma imagem é armazenada durante todo o uso da plataforma. As imagens recebidas por meio de requisições HTTP são enviadas com uma codificação base64 e as imagens só começam a ser capturadas após o consentimento do usuário para capturar a imagem no navegador.

Além disso, antes da sessão do experimento, os usuários responderam um formulário contendo perguntas relacionadas ao consentimento do uso de imagem e dos dados extraídos pela plataforma.

3.2.2 Definição dos checkpoints

A definição dos pontos em que as imagens devem ser coletadas são controladas pelo cliente da aplicação externa. Durante todas as etapas, as imagens do usuário são capturadas em um intervalo de 2 milissegundos.

Ao usuário entrar na página, é iniciada a coleta de emoções, definindo esse primeiro momento como "Leitura da questão". A partir do momento que o usuário começa a escrever o seu código, outra etapa é iniciada, sendo chamada de "Codificação". Com o envio do código para correção do juiz, outra etapa é iniciada, visando capturar as possíveis emoções após a resposta de sucesso ou fracasso.

Caso o código enviado esteja incorreto, o sistema retorna a etapa de codificação, caso contrário, o sistema finaliza a coleta de imagens do usuário até que o mesmo inicie outra questão.

Após a coleta de dados dos usuários, foi possível desenvolver uma linha do tempo de cada uma das submissões, visando entender qual momento durante a resolução evocou uma emoção diferente, durante os pontos chave da resolução.

Posteriormente, após o tratamento dos dados coletados, essa linha do tempo fica disponível para que sejam feitas análises das submissões, avaliando os efeitos da detecção de emoções para cada momento da resolução de cada aluno.

3.2.3 Escalabilidade da API

Para avaliar a escalabilidade da API desenvolvida para a detecção de emoções em expressões faciais, foi realizado um teste de carga abrangente (NETO; CLAUDIO, 2007). O objetivo do teste era verificar o desempenho da API quando submetida a um

alto volume de requisições simultâneas, simulando uma carga realista em um ambiente de produção.

Durante o teste, foram enviadas, gradualmente, até 10 mil requisições simultâneas à API, representando um cenário desafiador e intenso. Cada requisição consistia na submissão de uma expressão facial para detecção de emoções. A latência média registrada durante o teste foi de 50 milissegundos, demonstrando uma resposta rápida e eficiente mesmo sob uma carga significativa.

Esses resultados indicam que a API apresenta uma escalabilidade promissora, pois conseguiu lidar com um volume significativo de requisições sem comprometer a qualidade ou o tempo de resposta. A latência média observada de 50 milissegundos garante uma experiência de usuário suave e responsiva, mesmo durante períodos de pico de utilização.

Essa capacidade de escalabilidade é essencial para plataformas de juízes online, em que um grande número de alunos pode estar interagindo simultaneamente com a API ao resolver problemas computacionais. A capacidade de receber, processar e fornecer respostas rápidas, mesmo com milhares de requisições simultâneas, permite que a API seja amplamente utilizada sem interrupções ou atrasos perceptíveis.

Em suma, o teste de carga demonstrou a notável escalabilidade da API, fornecendo uma resposta rápida e estável mesmo sob altas demandas. Essa escalabilidade comprovada fortalece a confiabilidade da API e confirma sua capacidade de suportar um número substancial de usuários simultâneos, aprimorando assim a experiência de aprendizado dos alunos na resolução de problemas de programação.

4 Resultados

A partir dos resultados obtidos, é possível concluir que o modelo de Rede Neural proposto para a detecção de emoções apresentou um desempenho notável. A acurácia e precisão para a detecção de várias emoções, como mostrado na Tabela 1, foram bastante altas para a maioria das emoções, com destaque para a emoção de alegria, que alcançou uma acurácia e precisão de 92%.

No entanto, o modelo teve um desempenho mais baixo na detecção da emoção de nojo, com uma acurácia de 0.60 e precisão de 0.63. Isso sugere que essa emoção pode ser mais desafiadora para ser detectada, possivelmente devido à sua complexidade e à semelhança com outras emoções.

As Tabelas 5 e 4 mostram a frequência das emoções detectadas em diferentes momentos durante a resolução de questões. É interessante notar que a emoção neutra foi a mais frequente durante a leitura e a codificação, o que pode indicar que os usuários estavam concentrados nessas tarefas. Além disso, a emoção de felicidade foi observada com frequência durante a submissão das respostas, o que pode sugerir um sentimento de realização.

Em suma, os resultados discutidos nesse estudo de caso demonstram o potencial das Redes Neurais Convolucionais para a detecção de emoções em um contexto de aprendizado de algorítmos. Apesar disso, também mostram a necessidade de melhorar a detecção de emoções mais complexas e sutis. Acreditamos que este trabalho contribui para o campo de detecção de emoções e mostra um caminho para futuras pesquisas que podem explorar técnicas mais avançadas e aplicações práticas desta tecnologia em um experimento feito em uma amostra mais volumosa.

Além disso, resultados iniciais do caso de uso descrito foram positivos. Com os dados recebidos foi possível traçar uma trilha temporal em cada uma questões, podendo ter uma discussão sobre a eficácia de utilizar a mesma para o ensino de um tópico específico.

Além disso, a API pode ser utilizada como um serviço em diferentes casos de uso, sua eficiência e robustez pôde ser demonstrada após os testes efetuados e sua utilização em uma aplicação externa. Esses resultados sugerem que ela pode ser efetivamente utilizada em uma variedade de cenários e aplicações. Isso inclui, mas não se limita a ambientes de aprendizado virtuais, em que a detecção de emoções pode ser usada para melhorar a experiência do usuário e otimizar o processo de aprendizado. Esse serviço tem o potencial de ser integrado em uma variedade de outras aplicações, tais como jogos, assistentes virtuais, e sistemas de recomendação, onde a detecção de emoções pode ser usada para personalizar a experiência do usuário com base em seu estado emocional atual.

5 Conclusão

Este artigo apresenta uma abordagem inovadora que utiliza a detecção de emoções como ferramenta de diagnóstico em plataformas de juízes online no ensino de algoritmos. Através da análise temporal das emoções experimentadas pelos alunos, foi possível identificar padrões emocionais e compreender suas experiências durante cada etapa do processo de resolução de problemas.

Ao desenvolver uma API escalável para a detecção de emoções em expressões faciais, este estudo também contribuiu para a disponibilização de uma ferramenta acessível e eficiente. Os resultados dos testes de carga evidenciaram a capacidade da API de lidar com um alto volume de requisições simultâneas, garantindo uma experiência de usuário fluida.

Com a detecção de emoções em tempo real, torna-se possível personalizar o ambiente de aprendizado, oferecer suporte emocional adequado e aumentar o engajamento dos alunos. Essa abordagem promove um ensino mais adaptativo, direcionado às necessidades individuais de cada estudante.

No entanto, é importante destacar que este estudo apresenta algumas limitações e sugere possíveis direções para pesquisas futuras. É fundamental explorar outras técnicas de análise de emoções, como a detecção de emoções em áudio e linguagem natural, para uma compreensão mais abrangente das experiências emocionais dos alunos.

Na perspectiva de futuros trabalhos, é interessante a elaboração de um experimento com um valor expressivo de pessoas com diferentes níveis de experiência em programação, contendo pessoas de diferentes gêneros, idades e variedade de amostra, visando buscar a validação da rede neural.

Em suma, este trabalho proporciona uma nova perspectiva sobre o uso da detecção de emoções como ferramenta de diagnóstico no ensino de algoritmos em plataformas de juízes online. Espera-se que essa abordagem contribua para uma experiência de aprendizado mais engajado e eficaz, impulsionando o desenvolvimento de metodologias educacionais inovadoras e melhorando o sucesso dos alunos nessa importante área de estudo. Além disso, a proposta permite avaliar também as questões disponíveis, ao analisar os sentimentos evocados por elas tanto na leitura quanto submissão da mesma.

Referências

AKHAND, M.; ROY, S.; SIDDIQUE, N.; KAMAL, M. A. S.; SHIMAMURA, T. Facial emotion recognition using transfer learning in the deep cnn. *Electronics*, MDPI, v. 10, n. 9, p. 1036, 2021. Citado na página 13.

ALLEN, D. E.; DONHAM, R. S.; BERNHARDT, S. A. Problem-based learning. *New directions for teaching and learning*, Wiley Online Library, v. 2011, n. 128, p. 21–29, 2011. Citado na página 13.

ASSIS, D. P. D. Uma arquitetura de software baseada em web services. Citado na página 12.

BAE, K.; RYU, H.; SHIN, H. Does adam optimizer keep close to the optimal point? *arXiv preprint arXiv:1911.00289*, 2019. Citado na página 23.

BHARATI, P.; PRAMANIK, A. Deep learning techniques—r-cnn to mask r-cnn: a survey. *Computational Intelligence in Pattern Recognition: Proceedings of CIPR 2019*, Springer, p. 657–668, 2020. Citado na página 12.

CHO, J.; MIRZAEI, S.; OBERG, J.; KASTNER, R. Fpga-based face detection system using haar classifiers. In: *Proceedings of the ACM/SIGDA international symposium on Field programmable gate arrays*. [S.l.: s.n.], 2009. p. 103–112. Citado na página 11.

DHALL, A.; GOECKE, R.; JOSHI, J.; SIKKA, K.; GEDEON, T. Emotion recognition in the wild challenge 2014: Baseline, data and protocol. In: *Proceedings of the 16th international conference on multimodal interaction*. [S.l.: s.n.], 2014. p. 461–466. Citado na página 20.

FALCKEMBACH, G. A. M.; ARAUJO, F. V. de. Aprendizagem de algoritmos: dificuldades na resolução de problemas. *Anais Sulcomp*, v. 2, 2006. Citado na página 11.

GOODFELLOW, I. J.; ERHAN, D.; CARRIER, P. L.; COURVILLE, A.; MIRZA, M.; HAMNER, B.; CUKIERSKI, W.; TANG, Y.; THALER, D.; LEE, D.-H. et al. Challenges in representation learning: A report on three machine learning contests. In: SPRINGER. *Neural Information Processing: 20th International Conference, ICONIP 2013, Daegu, Korea, November 3-7, 2013. Proceedings, Part III 20*. [S.l.], 2013. p. 117–124. Citado na página 20.

JABBAR, H.; KHAN, R. Z. Methods to avoid over-fitting and under-fitting in supervised machine learning (comparative study). *Computer Science, Communication and Instrumentation Devices*, v. 70, p. 163–172, 2015. Citado na página 23.

KARAM, G. M. Visualization using timelines. In: *Proceedings of the 1994 ACM SIGSOFT international symposium on Software testing and analysis*. [S.l.: s.n.], 1994. p. 125–137. Citado na página 17.

KUSUMA, G. P.; JONATHAN, J.; LIM, A. Emotion recognition on fer-2013 face images using fine-tuned vgg-16. *Advances in Science, Technology and Engineering Systems Journal*, v. 5, n. 6, p. 315–322, 2020. Citado na página 21.

LATHKAR, M. Getting started with fastapi. In: *High-Performance Web Apps with FastAPI: The Asynchronous Web Framework Based on Modern Python*. [S.l.]: Springer, 2023. p. 29–64. Citado na página 18.

NETO, A.; CLAUDIO, D. Introdução a teste de software. *Engenharia de Software Magazine*, v. 1, p. 22, 2007. Citado na página 28.

SCHMIDT, H. G.; ROTGANS, J. I.; YEW, E. H. The process of problem-based learning: what works and why. *Medical education*, Wiley Online Library, v. 45, n. 8, p. 792–806, 2011. Citado na página 11.

SHORTEN, C.; KHOSHGOFTAAR, T. M. A survey on image data augmentation for deep learning. *Journal of big data*, SpringerOpen, v. 6, n. 1, p. 1–48, 2019. Citado na página 20.

WEISS, K.; KHOSHGOFTAAR, T. M.; WANG, D. A survey of transfer learning. *Journal of Big data*, SpringerOpen, v. 3, n. 1, p. 1–40, 2016. Citado na página 21.

ZATARAIN-CABADA, R.; BARRON-ESTRADA, M. L.; GONZALEZ-HERNANDEZ, F.; RODRIGUEZ-RANGEL, H. Building a face expression recognizer and a face expression database for an intelligent tutoring system. In: IEEE. *2017 IEEE 17th International Conference on Advanced Learning Technologies (ICALT)*. [S.l.], 2017. p. 391–393. Citado 2 vezes nas páginas 13 e 14.

ZHAO, R.; NIU, X.; WU, Y.; LUK, W.; LIU, Q. Optimizing cnn-based object detection algorithms on embedded fpga platforms. In: SPRINGER. *Applied Reconfigurable Computing: 13th International Symposium, ARC 2017, Delft, The Netherlands, April 3-7, 2017, Proceedings 13*. [S.l.], 2017. p. 255–267. Citado na página 22.