Transcrição Automatizada de Algoritmos Manuscritos: Reduzindo Esforços com Inteligência Artificial

Lucas Gabriel de Resende José[1], Ângelo Magno de Jesus[1]

[1] Instituto Federal de Minas Gerais, Campus Ouro Branco (IFMG)

lucasgresende12@gmail.com, angelo.jesus@ifmg.edu.br

Abstract. *Technology has transformed the educational context, making life easier for young people, civil servants, and teachers. When it comes to learning, writing enhances information retention and improves memorization. However, in the area of information technology, programming requires codes to be typed and tested in compilers and translators, which can compromise the efficiency of students when transposing handwritten algorithms to the digital environment. In this context, it is worth mentioning activities that require manual writing of codes or drafts, whether due to preference or limitations of computational resources, which are challenging for teachers, who need to correct them manually. These activities require significant cognitive effort or testing of their drafted solutions. Therefore, to solve this problem, we present this study as a computational tool based on Machine Learning aimed at facilitating and automating the process of transcribing handwritten algorithms into executable code. This tool facilitates the correction of activities, reduces the effort of correcting tests by teachers, and promotes the practical exploration of ideas in the development of algorithms.*

Resumo. *A tecnologia tem transformado o contexto educacional, facilitando a vida de jovens, servidores e professores. No que diz respeito à aprendizagem, a escrita potencializa a retenção de informações e aprimora a memorização. Entretanto, na área de tecnologia da informação, a programação exige que os códigos sejam digitados e testados em compiladores e tradutores, o que pode comprometer a eficiência de estudantes ao transpor algoritmos escritos à mão para o ambiente digital. Neste contexto, vale exemplificar as atividades que exigem escrita manual de códigos ou rascunhos, seja por questões preferenciais ou limitações de recursos computacionais, são desafiadoras para os docentes, que necessitam corrigi-las manualmente. Estas atividades demandam esforço cognitivo significativo ou testando suas soluções rascunhadas, assim, para solucionar este problema, apresentamos este estudo como uma ferramenta computacional baseada em Aprendizado de Máquina visando facilitar e automatizar o processo de transcrição de algoritmos manuscritos em código executável. Esta ferramenta facilita a correção de atividades, reduz o esforço de correção de provas por parte dos professores e promove a exploração prática de ideias no desenvolvimento de algoritmos.*

1. Introdução

A revolução digital reinventou o mundo, inovando-o, reduzindo consideravelmente o uso de papel, transformando a escrita manual em escrita digital, encurtando o tempo necessário para comunicação e ampliando a circulação de informações em altas velocidades.

No contexto educacional, alunos, servidores, professores obtêm acesso a informação de maneira rápida e eficaz, impulsionando seu aprendizado disponível na internet. Entretanto, na área de tecnologia da informação o desenvolvimento de aplicações ocorre digitalmente, das quais o código precisa ser compilado e traduzido, obrigando o usuário a transpor códigos escritos à mão para o ambiente digital.

Estudos apontam que a escrita manual de códigos de programação tem um impacto maior no aprendizado. “Pesquisas publicadas na revistana Frontiers in Psychology, mostraram que dessa forma os alunos aprendem mais, além de melhorar a precisão ortográfica e a memória”(Belém, 2024).

Atividades ou rascunhos que exigem a escrita manual são limitadores para os docentes, já que estas precisam ser corrigidas manualmente, o que demanda um alto esforço e tempo.

Nesse sentido,o desenvolvimento de uma ferramenta baseado em aprendizagem de máquina, capaz de reconhecer e compilar o código manuscrito, poderia otimizar esse processo.

2. Desenvolvimento

O *Machine Learning (*Aprendizado de Máquina) se relaciona com Inteligência Artificial, mas são termos distintos. A ideia de Inteligência Artificial foi popularizada por *Alan Turing,* com a pergunta “Máquinas podem pensar?” em seu artigo *Computing Machinery And Intelligence*, 1950, que descreve que uma máquina poderia enganar um ser humano, ficando conhecido como *Teste de Turing*.

Vale salientar que *“Machine Learning* é a ciência do desenvolvimento de algoritmos e modelos estatísticos que os sistemas de computador usam para realizar tarefas sem instruções explícitas, confiando em padrões e inferências.” (AWS, 2025).

Para o desenvolvimento de um aprendizado de máquina é necessário a definição de uma estrutura em que o código funcione como um neurônio. Todo código passa por um processo de análise antes de qualquer execução, seja ela léxica, sintática ou semântica.

No caso do Python, uma linguagem interpretada, seu código é lido e executado *line by line* sem a necessidade de ter uma compilação prévia. Assim, o Python torna-se uma escolha essencial em demandas que necessitam de agilidade e fluidez.

Abaixo, pode-se notar a arquitetura modulada para desenvolvimento da ferramenta:

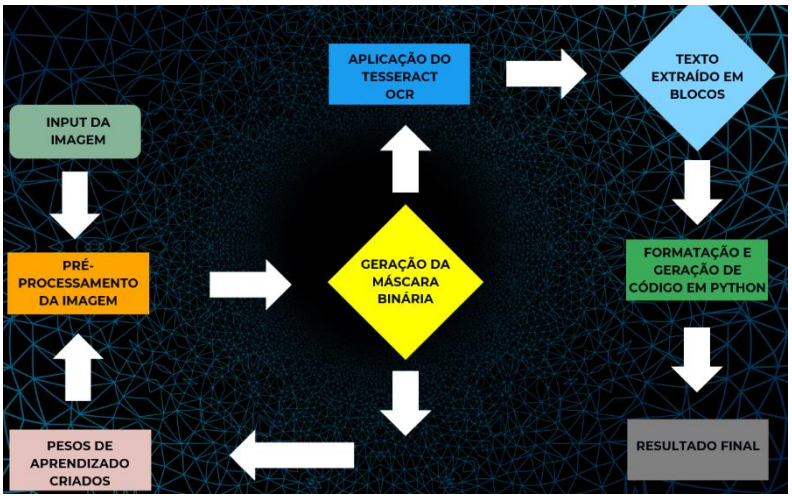

Figura 1. Arquitetura da ferramenta

Para devida solução do problema, utilizamos o **Modelo U-NET, ‘**uma arquitetura de rede neural convolucional desenvolvida para segmentação de imagens biomédicas’ [Ronneberger, Fischer e Brox, 2015]. Seu formato é com a letra “**U”**, pois usa duas partes principais: **encoder e decoder.**

O **encoder** é responsável por extrair as características da imagem, reduzindo sua dimensão progressivamente, técnica conhecida como ***downsampling.*** Ele irá usar duas convoluções 3 x 3, das quais a imagem passa por dois filtros detectando padrões, sendo que, a cada convolução, aplica-se uma função de ativação chamada **ReLU (Rectified Linear Unit)** para não trazer não-linearidade ao modelo. Logo em seguida, ocorre um **Max Pooling 2x2**, o que significa que a resolução é reduzida pela metade, mantendo apenas as características de interesse da imagem. A cada etapa, o número de filtros dobra, fazendo com que a rede aprenda sobre os padrões a cada nível.

O **decoder** é o caminho expansivo, responsável pela reconstrução da imagem segmentada, chamando em termos técnicos de ***upsampling.*** Nesse caso, ele irá passar por um processo de “**up-convolution”,** uma convolução 2x2 que aumenta a resolução e reduz o número de filtros pela metade. Aplica-se em seguida duas convoluções 3x3 + **ReLU** para refinar as características. É um processo importante, pois durante o processo, a imagem perde alguns

pixels em sua borda. No final, uma convolução 1x1 transforma os vetores de características em probabilidade para cada classe da segmentação.

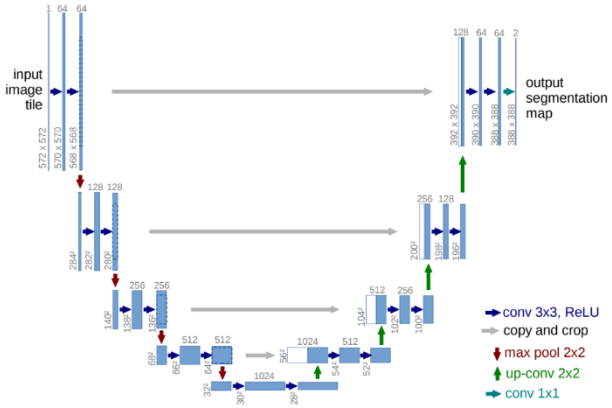

Figura 2. Modelo de Arquitetura U-NET

Nas figuras abaixo é possível verificar um exemplo prático de binarização da imagem ao utilizar a arquitetura U-Net:

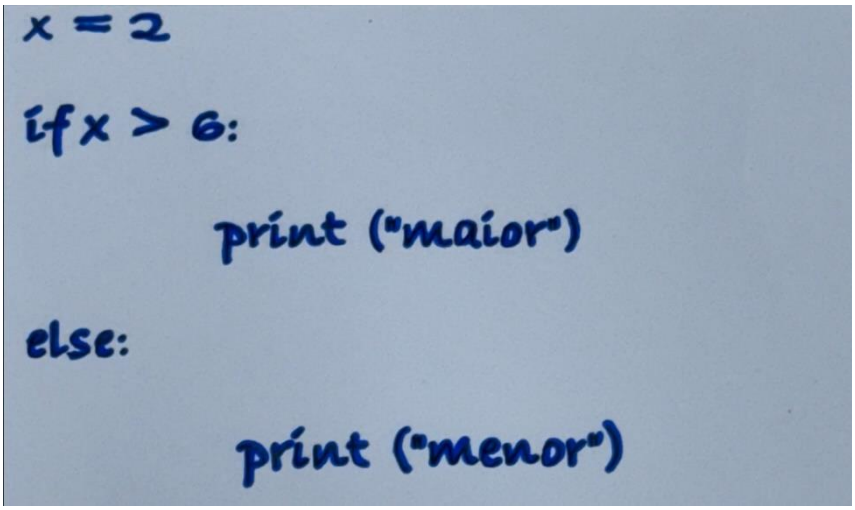

Figura 3. Foto de um trecho de código em Python

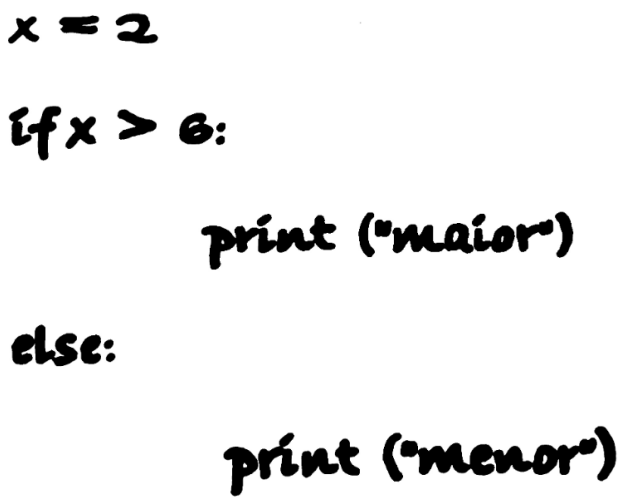

Figura 4. Imagem binarizada após processamento U-NET

O modelo de binarização da imagem, isto é, transformá-la em preto e branco nos permite uma extração com maior chance de um texto escrito de maneira manuscrita. Para isso, utilizamos uma técnica chamada **Tesseract OCR**, "um mecanismo de reconhecimento de texto baseado em Machine Learning, utilizado para extração de texto impresso em imagens, com suporte em diversos idiomas". (BISPO, 2024).

Após receber a imagem binarizada o Tesseract 'processa o documento analisando as formas dos caracteres e convertendo-os em texto digital' [BISPO, 2024]. Esses textos são armazenados em blocos, sendo cada palavra com seu respectivo bloco. Para auxiliar na análise e ajuste do treinamento, utilizamos o **jTessBoxEditor**, ferramenta que nos permite analisar o armazenamento de texto em blocos**,** sendo responsável por auxiliar e ajustar treinamento para o Tesseract OCR.

Abaixo é possível observar como é realizado essa análise:

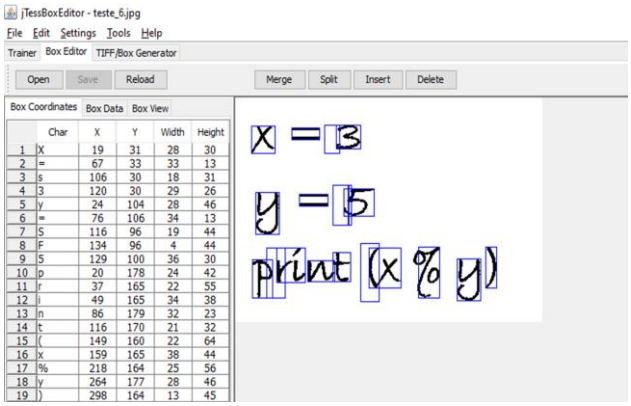

Figura 5. Armazenamento de texto em Blocos com jTessBoxEditor

Na sequência, para a extração do arquivo box com a aplicação da ferramenta Tesseract OCR, utilizou-se regras de formatação para a linguagem Python.

Neste sentido, é consabido que o Python exige uma endentação correta para compilação ser bem-sucedida, assim, desenvolveu-se técnicas específicas para estruturar o código extraído, de forma que o código seja executado normalmente, exibindo, destarte, o resultado da compilação.

3. Resultados

Após diversos treinamentos de binarização de imagem e aplicação correta de formatação para Python, logrou-se a extração do texto do código e o tornou executável.

Ademais, foram implementadas regras de negócio para que nem todo texto seja de fato um código em Python. A imagem abaixo torna possível visualizar o código extraído:

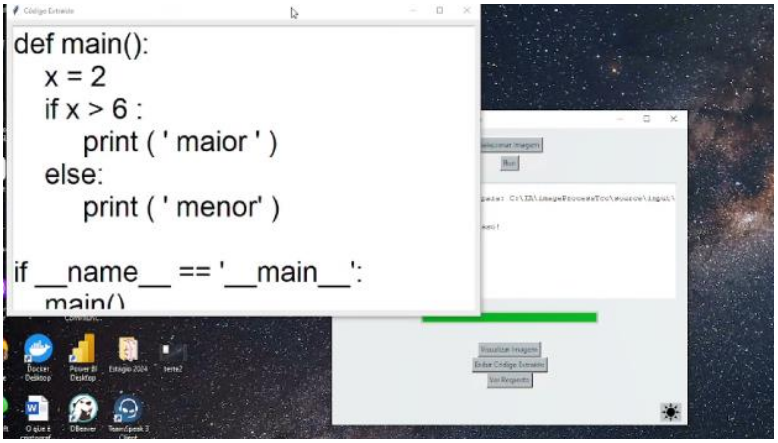

Figura 6. Código formatado após executado as técnicas de formatação

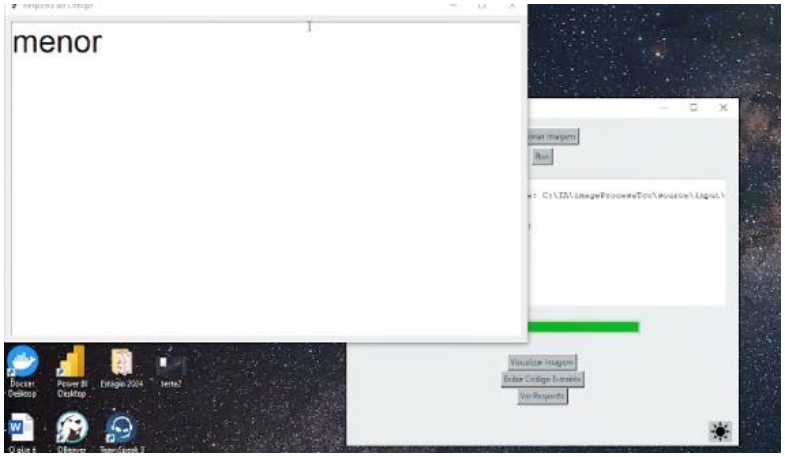

Figura 7. Resposta da compilação do código

4. Conclusão

Com essa ferramenta, é possível reduzir o desgaste ao analisar tarefas que requerem escritas em códigos, sendo que há múltiplas maneiras de se chegar um resultado. Além disso, ela otimiza o tempo ao permitir que ideias momentâneas sejam rapidamente testadas e estruturadas, transformando rascunhos abstratos em soluções concretas

Outro importante benefício é sua aplicabilidade no ensino e aprendizado em programação. Muitos alunos possuem hábitos de escrever código no papel antes de digitá-lo virtualmente, o que acaba sendo um processo demorado. Com essa ferramenta, suas ideias podem ser válidas de maneira instantâneas, reduzindo a frustação e aumentando a eficiência de aprendizado.

Ela também pode ser uma importante aliada para profissionais e desenvolvedores que desejam testar seus conceitos de maneira rápida, sem a necessidade de configurar um ambiente de desenvolvimento completo. Isso não só melhora a produtividade, mas incentiva a experimentação e inovação de ideias.

Por final, observou-se que a ferramenta contribui para um fluxo de trabalho mais dinâmico, colaborativo e eficiente, impactando de maneira positiva tanto na educação quanto no setor profissional de tecnologia.

5. Trabalhos Futuros

Os próximos passos para deixar a ferramenta mais intuitiva, otimizada e precisa envolve a integração com dispositivos moveis, reduzindo ainda mais o tempo de validação de código, suporte para mais linguagens, viso que cada aluno/docente tem uma linguagem preferida, além de melhor reconhecimento da caligrafia, permitindo um reconhecimento mais avançado e preciso, visto que cada pessoa tem uma forma de escrever.

6. Referência Bibliográfica

BELÉM, K. Escrever à mão faz bem para o cérebro e melhora a memória; Melhor do que digitar. Disponível em: <*https://www.sonoticiaboa.com.br/2024/01/29/escrever-mao-faz-bem-cerebro-melhora-memoria-melhor-que-digitar?utm_source=chatgpt.com*>. Acesso em: 5 fev. 2025.

O que é machine learning? – **Explicação sobre machine learning empresarial – AWS**. Disponível em: <*https://aws.amazon.com/pt/what-is/machine-learning/*>. Acesso em: 6 fev. 2025

RONNEBERGER, Olaf; FISCHER, Philipp; BROX, Thomas. **U-Net: Convolutional Networks for Biomedical Image Segmentation**. 2015. Disponível em: *https://arxiv.org/abs/1505.04597*. Acesso em: 5 fev. 2025**.**

PAPERS WITH CODE. **U-Net**. Disponível em: *https://paperswithcode.com/method/u-net*. Acesso em: 5 fev. 2025**.**

VIETOCR. **Training VietOCR**. Disponível em*: https://vietocr.sourceforge.net/training.html*. Acesso em: 8 fev. 2025**.**

LIMA, José Guilherme Pereira. **Attention Efficient Residual U-Net: uma Rede Neural para a Segmentação de Lesões de Pele**. 2021. 39 f. Monografia (Graduação em Ciência da Computação) – Universidade Federal do Maranhão, São Luís, 2021. Disponível em: *https://monografias.ufma.br/jspui/bitstream/123456789/6368/1/Jos%C3%A9GuilhermePereiraLima.pdf*. Acesso em: 8 fev. 2025**.**