SISTEMA PARA TRADUÇÃO AUTOMÁTICA ENTRE TEXTO E BRAILLE UTILIZANDO RECONHECIMENTO ÓPTICO

SYSTEM FOR AUTOMATIC TRANSLATION BETWEEN TEXT AND BRAILLE USING OPTICAL RECOGNITION

Alexsandro Costa da Silva*
Alexandro Lima Damasceno**

RESUMO

O acesso limitado a materiais em braille afeta milhões de pessoas com deficiência visual que dependem dessa forma de leitura. Este trabalho propõe um sistema que, por meio de documentos ou imagens, realize traduções entre texto alfabético e braille e vice-versa. Para isso, emprega-se o YOLOv11 na detecção automática de células braille em imagens e o Tesseract OCR na extração de texto impresso. A arquitetura do sistema é baseada em um backend desenvolvido em Django, responsável pelo processamento das imagens e pela conversão de formatos. O modelo de detecção de braille alcançou 97,8% de mAP@50, optando-se pelo uso de um modelo sem classes numéricas para eliminar ambiguidades e garantir a assertividade da tradução alfabética. O reconhecimento de texto impresso atingiu uma taxa de erro de palavra de 7,1% e uma taxa de erro de caractere de apenas 2,0% após o pré-processamento. Como contribuição prática, o sistema possibilita a rápida geração de arquivos BRF, PDF e DOCX, facilitando a produção de materiais acessíveis em ambientes educacionais e reduzindo custos e tempo de tradução manual.

Palavras-chave: Braille. Acessibilidade. Reconhecimento de Imagem. Reconhecimento Óptico de Caracteres. Tradução.

ABSTRACT

Limited access to braille materials affects millions of visually impaired people who depend on this form of reading. This work proposes a system that, through documents or images, performs translations between alphabetic text and braille and vice versa. To this end, YOLOv11 is used for the automatic detection of braille cells in images, and Tesseract OCR is used for the extraction of printed text. The system architecture is based on a backend developed in Django, responsible for

* Graduando em Ciência da Computação, Instituto Federal de Educação, Ciência e Tecnologia do Ceará, Aracati, CE, Brasil. E-mail: alexsandro.costasv@gmail.com
** Mestre em Ciência da Computação, docente do Instituto Federal de Educação, Ciência e Tecnologia do Ceará, Aracati, CE, Brasil. E-mail: alexandro.lima@ifce.edu.br

image processing and format conversion. The braille detection model achieved 97.8% mAP@50, opting for the use of a model without numerical classes to eliminate ambiguities and ensure the accuracy of the alphabetic translation. Printed text recognition achieved a word error rate of 7.1% and a character error rate of only 2.0% after preprocessing. As a practical contribution, the system enables the rapid generation of BRF, PDF, and DOCX files, facilitating the production of accessible materials in educational environments and reducing the costs and time of manual translation.

Keywords: Braille. Accessibility. Image Recognition. Optical Character Recognition. Translation.

1 INTRODUÇÃO

De acordo com projeções do IBGE (2025), a população brasileira é de aproximadamente 212 milhões de pessoas, das quais cerca de 6,5 milhões possuem deficiência visual, com diferentes níveis de severidade, desde a cegueira total até dificuldades severas para enxergar. À medida que a visão de um indivíduo se agrava, ele passa a depender cada vez mais de outros sentidos, como o tato, para realizar atividades cotidianas, inclusive a leitura de textos. Para isso, existe o sistema braille, o que permite que pessoas cegas ou com baixa visão leiam e escrevam por meio de pontos em relevo.

Diante desse cenário demográfico e da crescente demanda por acessibilidade, os avanços tecnológicos recentes têm proporcionado aos cegos diversas novas possibilidades de acesso e compartilhamento de informações. No entanto, a leitura e a escrita em braille impresso ainda representam uma forma essencial de comunicação para essas pessoas. O braille também é amplamente utilizado na interação entre cegos e não cegos, sendo especialmente comum em contextos educacionais, como quando professores não cegos ensinam alunos cegos e precisam manusear livros didáticos ou tarefas produzidas em braille (OVODOV, 2020).

Contudo, a disponibilidade de materiais acessíveis ainda é um grande desafio. Menos de 10% dos livros publicados mundialmente estão disponíveis em formatos acessíveis, segundo Jubeh, Dard e Zayed (2020). Além disso, muitos documentos antigos em braille, encontrados em bibliotecas, estão se tornando obsoletos e não podem ser reproduzidos, pois não existem arquivos originais desses documentos em forma de texto alfabético para que possam ser reimpressos. A única alternativa seria a tradução manual, documento por documento (YAMIN; KUSUMA; TASRIPAN, 2022), processo que pode levar de 1 a 5 dias, mesmo para documentos pequenos, e pode custar, em média, até R$ 30 por página (RAJ, 2022).

Em virtude do que foi apresentado, é proposto o desenvolvimento de um sistema capaz de realizar a tradução bidirecional entre texto alfabético e braille a partir de imagens e documentos digitais. Para isso, os objetivos específicos consistem em identificar automaticamente pontos em braille por meio de técnicas de processamento de imagens e de uma rede neural treinada

para reconhecer relevos, converter braille para texto alfabético de forma precisa, aplicar OCR para transformar textos impressos em sua representação braille e gerar arquivos compatíveis com leitores de tela e impressoras braille, garantindo acessibilidade, preservação de acervos e ampliação do acesso à informação por pessoas com deficiência visual.

O restante do artigo está organizado da seguinte forma: a Seção 2 apresenta o referencial teórico, abordando o sistema braille, YOLOv11, o Tesseract OCR e o framework Django; a Seção 3 discute os trabalhos relacionados e compara as soluções existentes; a Seção 4 descreve a metodologia de criação do dataset, treinamento e arquitetura do sistema; a Seção 5 detalha os resultados obtidos, analisando o desempenho dos modelos e a validação prática da ferramenta; e, por fim, a Seção 6 apresenta as conclusões e perspectivas para trabalhos futuros.

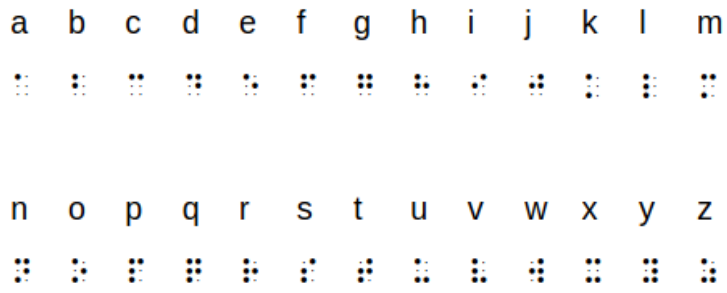
2 REFERENCIAL TEÓRICO

Esta seção aborda os conceitos e tecnologias essenciais para este trabalho, com foco em métodos de captura, reconhecimento e conversão de padrões táteis em imagens e texto, incluindo o sistema braille, técnicas de visão computacional, reconhecimento óptico de caracteres e a arquitetura de servidor do sistema.

2.1 Braille

Louis Braille é reconhecido como o criador do sistema de escrita por tato que leva seu nome. Esta inovação revolucionária possibilita que indivíduos cegos tenham acesso à cultura escrita e pode, portanto, ser visto como um significativo progresso na melhoria da qualidade de vida das pessoas com cegueira (JIMéNEZ et al., 2009). O sistema braille é formado por células compostas por 6 pontos em alto-relevo, organizados em 3 linhas e 2 colunas (YAMIN; KUSUMA; TASRIPAN, 2022), como ilustrado na Figura 1, que apresenta o alfabeto braille. Cada ponto pode estar elevado ou não, o que gera $2^6 = 64$ combinações possíveis.

Figura 1 – Alfabeto braille



Fonte: (SANTOS, 2018)

Além de representar o alfabeto tradicional, o sistema braille também é versátil em outras áreas do conhecimento. Segundo Teixeira et al. (2016), o sistema braille não se limita à representação de símbolos literais, podendo também ser aplicado a símbolos matemáticos, químicos, fonéticos, informáticos, musicais, entre outros. Na língua portuguesa, a maioria dos

caracteres é mantida em sua forma original, com exceção de algumas vogais acentuadas, que possuem sinais específicos, conforme exemplificado na Figura 2

Figura 2 – Letras com diacríticos

Vogais	a		e		i		o		u	
Acento agudo	á		é		í		ó		ú	
Acento grave	à		–		–		–		–	
Acento circunflexo	â		ê		–		ô		–	
Til	ã		–		–		õ		–	

Consoante	c	
Cedilha	ç	

Fonte: (SANTOS, 2018)

Tendo estabelecido os fundamentos do sistema braille e suas representações táteis, impõe-se a investigação de métodos computacionais capazes de identificar esses padrões em imagens. Para atender a essa necessidade de detecção eficiente, este trabalho adota uma abordagem baseada em aprendizado profundo. A próxima seção detalha o YOLOv11, a arquitetura de rede neural escolhida para realizar o reconhecimento automático das células em relevo.

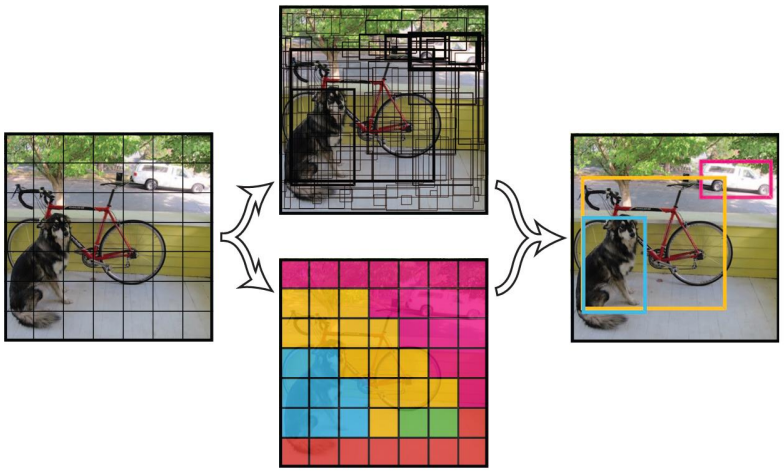
2.2 YOLOv11

You Only Look Once (YOLO) é uma família de modelos de redes neurais convolucionais de código aberto, desenvolvida para realizar detecção e classificação de objetos em tempo real (ALSULTAN; MOHAMMAD, 2023). Desde sua proposta inicial em 2015, as variantes YOLO evoluíram para equilibrar desempenho, precisão e eficiência computacional em dispositivos com recursos limitados, o que permite atender a aplicações que exigem alta taxa de quadros por segundo e detecção robusta (HUSSAIN, 2023).

Diferentemente de outros métodos que dividiam a imagem em várias regiões e aplicavam classificadores independentes em cada uma delas, o YOLO trata a detecção e a classificação como um único problema de regressão. A rede processa a imagem inteira em apenas uma passagem, gerando simultaneamente as caixas delimitadoras (*bounding boxes*) e as respectivas probabilidades de classe para cada ponto (GOMES, 2022). Durante a detecção, várias caixas delimitadoras podem ser previstas para uma mesma classe. Para eliminar sobreposições e selecionar apenas as predições mais confiáveis, aplica-se o algoritmo de supressão não máxima (NMS) (RIBEIRO et al., 2024), ilustrado na na Figura 3.

Em setembro de 2024, a Ultralytics lançou o YOLOv11, versão que otimiza o equilíbrio entre velocidade e precisão. O modelo introduz avanços na arquitetura de backbone e neck, melhorando a extração de características em cenários complexos. Esta versão apresenta uma maior Precisão Média (mAP) no dataset COCO utilizando 22% menos parâmetros que o seu antecessor, o que facilita a sua implementação em dispositivos de borda (edge computing) (ULTRALYTICS, 2025).

Figura 3 – Algoritmo NMS em ação após detectar várias caixas delimitadoras.



Fonte: (RIBEIRO et al., 2024)

Apesar dos modelos da YOLO desempenharem um papel fundamental na detecção e classificação de padrões visuais, nem todas as informações presentes nos documentos, que podem ser compostos integralmente por texto em relevo (braille) ou por texto impresso convencional, podem ser interpretadas apenas com base em suas estruturas visuais. Assim, enquanto a CNN lida com o reconhecimento dos padrões braille, o conteúdo em texto impresso exige técnicas específicas de reconhecimento óptico de caracteres (OCR). Para essa finalidade, este trabalho adota o Tesseract, cuja abordagem será detalhada na próxima seção.

2.3 Tesseract OCR

Optical Character Recognition (OCR), ou Reconhecimento Óptico de Caracteres, permite que computadores leiam e processem textos contidos em imagens manuscritas ou digitalizadas. Esse processo converte os padrões dos caracteres presentes na imagem em texto legível por máquina, tornando as informações acessíveis digitalmente (WANG, 2023). Entre as opções de OCR, o Tesseract se destaca por sua precisão e alto desempenho. Criado inicialmente pela HP e mantido atualmente pelo Google, o Tesseract é amplamente utilizado por sua capacidade de transformar imagens em texto editável (PATIENCE et al., 2024).

O Tesseract OCR utiliza redes neurais recorrentes baseadas em Memória de Longo Prazo (LSTM), técnicas comuns em processamento de linguagem natural, que contribuem para a redução de erros na identificação dos caracteres (GARAI et al., 2022).

Após o texto alfabético ser extraído por meio do Tesseract OCR, ele pode ser convertido para o sistema braille, utilizando o mapeamento de cada letra ou símbolo para uma célula de seis pontos. O Tesseract facilita a digitalização e extração do conteúdo textual de documentos, o que permite que outra ferramenta realize a transformação desse texto em braille e o torne acessível de maneira tátil para pessoas com deficiência visual. Para operacionalizar todo esse processo, a próxima seção apresenta o *framework* Django, que coordena o uso do OCR e da CNN para realizar a tradução entre texto impresso e braille.

2.4 Django

O Django é um *framework* de desenvolvimento *web* de alto nível, mantido pela *Django Software Foundation*, que visa promover o desenvolvimento ágil e seguro de aplicações. Escrito em Python, o Django segue o padrão arquitetural MTV (Model-Template-View) e oferece uma estrutura robusta com diversas funcionalidades integradas, como sistema de autenticação, painel administrativo, ORM (Object-Relational Mapping), entre outros recursos (DJANGO SOFTWARE FOUNDATION, 2025).

O Django é empregado como *backend* do sistema, responsável por fornecer a API REST que recebe os dados, processa as imagens com OCR e CNN, realiza as conversões necessárias e retorna os resultados. Sua estrutura robusta e segura garante a comunicação eficiente dos módulos de reconhecimento e tradução, gerenciando todo o fluxo de dados recebidos pela API.

3 TRABALHOS RELACIONADOS

Nesta seção, são apresentados alguns trabalhos que exploram diversas abordagens para reconhecimento e conversão de braille a partir de imagens. Cada estudo contribui com uma perspectiva distinta. As abordagens variam desde métodos clássicos de processamento de imagem para segmentação de células em relevo, passando por soluções multifuncionais que integram OCR e tradução bidirecional texto–braille, até abordagens baseadas em redes neurais convolucionais que buscam robustez a variações de iluminação e perspectiva em dispositivos móveis.

Yamin, Kusuma e Tasripan (2022) propõem um sistema de cópia de documentos em braille como recurso adicional para a impressora braille do *Institut Teknologi Sepuluh Nopember* (ITS). A solução captura imagens de páginas em braille via *smartphone*, aplica processamento de imagem (binarização, segmentação e operações morfológicas) para identificar as células de 6 pontos e, em seguida, traduz cada padrão em caracteres alfabéticos. O texto reconhecido é então enviado, via conexão Wi-Fi, a um microcontrolador ESP32, que transmite os dados à impressora braille. Em experimentos com cerca de 1.200 imagens, o sistema alcançou 99% de acurácia no reconhecimento dos padrões em relevo, embora tenha sido testado em condições de iluminação controlada e não inclua extração de texto impresso via OCR nem operação em ambiente multiplataforma.

O trabalho de Raj (2022) apresenta um sistema multifuncional de conversão entre texto e braille, com funcionalidades como braille para texto, texto para braille, manuscrito para braille e geração de audiolivros. Utilizando o Tesseract OCR, o sistema é capaz de converter documentos em braille e texto manuscrito em formatos acessíveis para pessoas com deficiência visual. O objetivo principal é reduzir significativamente o tempo necessário para a conversão de documentos, que manualmente pode levar de 2 a 3 dias, para apenas alguns minutos. Apesar de sua abrangência, o projeto não especifica detalhes sobre a arquitetura utilizada, como o uso de redes neurais convolucionais (CNNs) para reconhecimento de padrões em braille.

O estudo de Hsu (2020) propõe um sistema de reconhecimento óptico de braille utilizando

redes neurais convolucionais (CNNs), alcançando 97,3% de acurácia em um *dataset* com 15.000 imagens de células braille. Os autores abordam desafios práticos como variações de iluminação e desgaste do material, aplicando técnicas de pré-processamento (normalização de brilho, filtragem de ruído) e aumento de dados (rotações de ±15°, ajustes de contraste). A arquitetura CNN de 8 camadas convolucionais mostrou-se superior a métodos tradicionais baseados em SVM, validando a eficácia de redes profundas para decodificação automática de braille em condições não ideais, um avanço crítico para aplicações móveis, onde a qualidade das imagens capturadas pode variar significativamente.

O Quradro 1 apresenta uma comparação entre os trabalhos citados e este trabalho, destacando os principais recursos utilizados por cada abordagem.

Quadro 1 – Comparação entre os trabalhos relacionados e a proposta atual

Recurso/Método	**Hsu (2020)**	**Raj (2022)**	**Yamin (2022)**	**Este Trabalho**
Braille para Texto	X	X	X	X
Texto para braille		X		X
Uso de OCR		X		X
Rede Neural (CNN)	X			X
Detecção braille em imagens	X		X	X

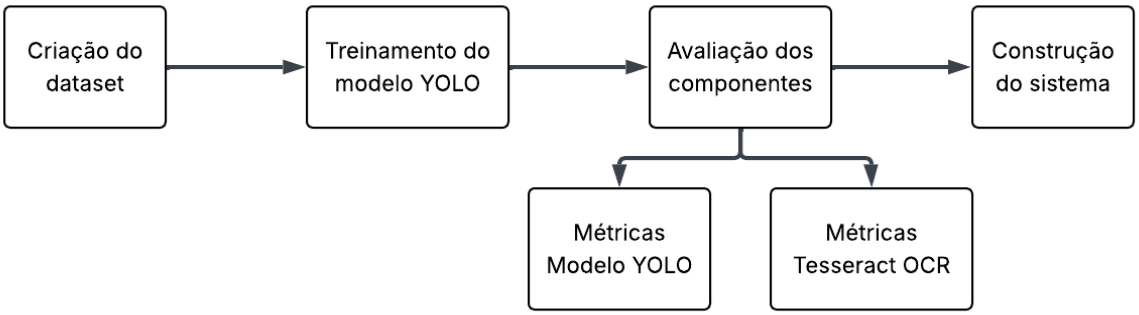
Fonte: Elaborado pelo autor.

Com base na análise dos trabalhos relacionados, observa-se que as soluções existentes focam em vias de tradução específicas ou carecem de integração tecnológica. Abordagens como a de Yamin, Kusuma e Tasripan (2022) e Hsu (2020) focam exclusivamente na via braille para texto, com Hsu (2020) já demonstra a robustez das CNNs para essa tarefa. Por outro lado, o sistema de Raj (2022) é bidirecional , mas não especifica o uso de redes convolucionais para o reconhecimento de braille a partir de imagens. Persiste, assim, a lacuna de uma arquitetura de sistema unificada que integre robustamente ambas as vias. Este trabalho propõe um fluxo de processamento híbrido, implementado como um *backend*, que utiliza CNN para a detecção de braille e OCR para a extração de texto, com o objetivo de oferecer um serviço de tradução bidirecional completo.

4 METODOLOGIA

Nesta seção são detalhadas as etapas percorridas para o desenvolvimento da solução proposta, conforme ilustra o fluxo metodológico na Figura 4. O processo inicia-se com a fase de preparação de dados e aprendizado de máquina, que envolve a criação do *dataset* e o treinamento do modelo YOLOv11. Na sequência, realiza-se a avaliação dos modelos, etapa em que são aplicadas as métricas de desempenho específicas para o modelo de detecção e para o Tesseract OCR. Por fim, descreve-se a construção do sistema, com o detalhamento da arquitetura da API e a integração lógica que implementa a conversão bidirecional.

Figura 4 – Fluxograma da Metodologia.



Fonte: Elaborado pelo autor.

4.1 Criação do Dataset

Para o treinamento do modelo YOLOv11, foi construído um *dataset* composto por 8.031 imagens de folhas com células braille, contendo a representação do alfabeto (A–Z), números (0–9) e diacríticos (vogais acentuadas e cedilha). As imagens foram obtidas a partir de fotografias de folhas nas quais as células foram produzidas manualmente com um reglete de escrita em braille, em papel de gramatura 120, e o conjunto foi complementado com imagens do conjunto de dados *Braille Dataset*[1] , originalmente do alfabeto cirílico, cujas marcações foram convertidas para o alfabeto português e as células que não representavam nenhum símbolo foram desmarcadas, e do *braille dataset 2*[2] , que contém representações no alfabeto A–Z e no qual foi necessário apenas criar as anotações.

A marcação de todas as imagens foi realizada de forma manual no *Roboflow*[3], onde foram desenhadas caixas delimitadoras (*bounding boxes*) para cada célula. Foi observada uma diferença na frequência de ocorrência de certas células ao unir as imagens próprias às dos demais *datasets*. Para corrigir esse desbalanceamento, foi aplicado um aumento de dados seletivo no conjunto, gerando variações das imagens das classes menos frequentes. As transformações aplicadas incluíram conversão para escala de cinza, aumento de contraste para realçar as sombras dos relevos, *blur* gaussiano leve para simular variações de foco, adição de ruído gaussiano, aumento de saturação e redução de brilho para diferenciar condições de iluminação.

Por fim, o *dataset* final foi dividido em 80%, correspondendo a 6.411 imagens para treinamento, 10%, correspondendo a 810 imagens para validação, e 10%, também com 810 imagens, para teste. Nenhum outro aumento de dados foi aplicado após a divisão. Por fim, o conjunto foi exportado no formato YOLOv11 e disponibilizado no Kaggle[4].

[1] Disponível em: <https://www.kaggle.com/datasets/nagabushan/braille-dataset>
[2] Disponível em: <https://www.kaggle.com/datasets/shemescobal/braille-dataset-2>
[3] Disponível em: <https://roboflow.com/>
[4] Disponível em: <https://www.kaggle.com/datasets/alexsandrocostasilva/brailledataset>

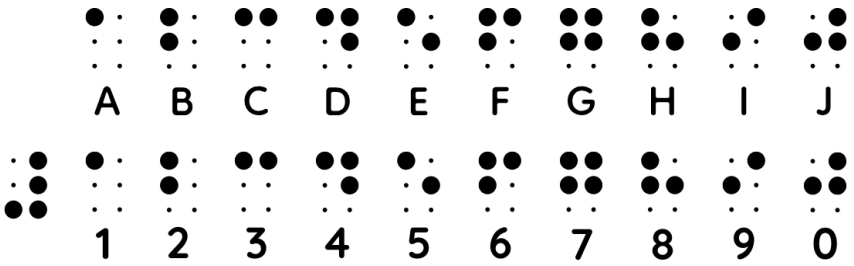
4.2 Treinamento do modelo YOLO

O treinamento teve como base o modelo pré-treinado *yolo11l.pt* , selecionado por oferecer boa capacidade de representação sem exceder os limites de memória da GPU disponíveis. O ambiente de execução utilizado foi o *Kaggle Notebooks*, que proporcionou acesso direto ao *dataset* e persistência dos arquivos de treinamento.

Os hiperparâmetros principais adotados foram 250 épocas, tamanho de imagem (imgsz) de 1280 e *batch size* igual a 4. O imgsz de 1280 foi escolhido para preservar os detalhes das pequenas células durante o pré-processamento e a inferência, enquanto o *batch size* reduzido permitiu adequar o consumo de memória de vídeo ao modelo selecionado. Utilizou-se ainda *patience* igual a 40 para controle de parada antecipada (*early stopping*), uma técnica que interrompe o treinamento caso o desempenho do modelo não melhore após um número definido de épocas. O treinamento foi executado com GPU T4.

Foi executado também outro treinamento com uma variação do *dataset* sem representações braille de números para evitar conflitos de classificação, visto que, no sistema Braille, os caracteres numéricos têm representações de pontos semelhantes das letras de 'a' a 'j' conforme ilustrado na Figura 5 e diferem apenas pelo uso do sinal de numeral, semelhante a um 'L' invertido.

Figura 5 – Padrões de pontos compartilhados entre letras e números no sistema braille



Fonte: Elaborado pelo autor.

4.3 Métricas de Avaliação dos Modelos

A avaliação do desempenho do YOLOv11 baseou-se em métricas padrão de detecção de objetos, conforme definido na documentação oficial da YOLO[5]. O critério central a *Intersection over Union* (IoU) para comparar predições com o *ground truth*.

Uma detecção foi considerada Verdadeiro Positivo (TP) quando apresentou IoU $>=$ 0,5 com a classe correta, enquanto predições com sobreposição inferior ou classe errada foram contadas como Falsos Positivos (FP), e células não detectadas como Falsos Negativos (FN). A partir dessas definições, calcularam-se a *precision* $(TP/(TP+FP))$, que indica a confiabilidade das detecções, e o *recall* $(TP/(TP+FN))$, que mede a capacidade do modelo em localizar todas as células presentes. Também foram aferidos o *F1-score* que é a média harmônica entre *precision*

[5] Disponível em: <https://docs.ultralytics.com/pt/guides/yolo-performance-metrics/>

e *recall* e a *mean Average Precision* (mAP) em duas configurações, mAP@50 limiar fixo de 0,5 e mAP@50-95 média de múltiplos limiares.

Paralelamente, para o reconhecimento de texto impresso, o sistema integrou o Tesseract OCR via biblioteca *pytesseract*. A validação deste componente utilizou um conjunto de 102 fotografias de páginas de livro. O experimento comparou o desempenho em imagens originais com o das imagens submetidas a pré-processamento que consistiu especificamente na conversão para escala de cinza e na utilização de filtro gaussiano. A eficácia foi medida pelas métricas de Taxa de Erro de Caractere (CER), que quantifica a proporção de inserções, deleções e substituições em relação ao total de caracteres e Taxa de Erro de Palavra (WER), que aplica a mesma lógica ao nível de palavras, calculadas via biblioteca *jiwer*.

4.4 Arquitetura do Sistema e Fluxo de Dados

A operacionalização dos modelos treinados ocorre por meio de uma arquitetura *backend* desenvolvida em Python, estruturada sobre o *framework* Django. O sistema foi projetado para funcionar como uma API RESTful, de modo a isolar a lógica de processamento pesado das interfaces de usuário. Para garantir a reprodutibilidade do ambiente e a escalabilidade horizontal, a aplicação foi containerizada via Docker. Essa estratégia isola as dependências complexas das bibliotecas de visão computacional como PyTorch e Tesseract do sistema operacional hospedeiro, o que facilita o *deploy* em diferentes servidores.

O fluxo de dados inicia-se na camada de aquisição, onde o sistema recebe as requisições via protocolo HTTP. Os *endpoints* da API foram estruturados para aceitar tanto imagens brutas para detecção de braille quanto documentos digitais como PDF e DOCX para extração de texto via OCR. O Quadro 2 detalha as principais rotas desenvolvidas para a interação com o sistema.

Quadro 2 – Descrição dos Endpoints da API

Método	Endpoint	Descrição
POST	/api/tradutor/braille-texto/	Recebe imagem com braille, executa o modelo YOLOv11 e retorna os links das imagens processadas e arquivos com a tradução.
POST	/api/tradutor/texto-braille/	Recebe documentos (PDF/DOCX/IMG) com texto impresso, executa OCR e retorna os links dos arquivos com a tradução.

Fonte: Elaborado pelo autor.

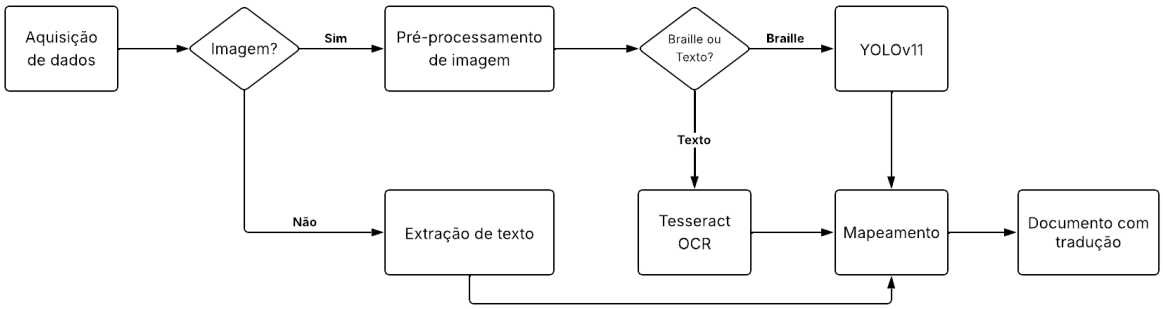
Ao receber um arquivo, o *controller* do Django valida o formato de entrada e encaminha o fluxo para o módulo de processamento correspondente. Diferentemente de execuções em *notebooks* experimentais, no ambiente de produção os modelos são carregados na memória na inicialização do serviço, o que reduz a latência por requisição.

A resposta do sistema é padronizada em formato JSON (*JavaScript Object Notation*). O objeto de retorno fornece as URLs diretas para download dos arquivos gerados com a tradução.

4.5 Fluxo de Processamento e Conversão

Após a aquisição e triagem dos dados pela API, inicia-se o núcleo de processamento lógico do sistema, cujo fluxo de execução está detalhado na Figura 6. Caso a entrada seja uma imagem, aplica-se primeiramente um pré-processamento que converte o arquivo para escala de cinza e aplica um filtro gaussiano, de modo a preparar a entrada tanto para o OCR quanto para a rede neural.

Figura 6 – Fluxograma do sistema proposto para tradução entre braille e texto alfabético.



Fonte: Elaborado pelo autor.

No módulo de tradução de braille para texto, o algoritmo de inferência utiliza o modelo YOLOv11. O processamento das predições segue uma sequência lógica onde o modelo retorna as classes e coordenadas das células identificadas, as quais são submetidas a uma ordenação espacial baseada nos eixos (x, y) para reconstruir o fluxo de leitura natural. Por fim, as classes ordenadas são decodificadas em caracteres alfabéticos por meio de um dicionário, o que gera a *string* de texto final.

No fluxo inverso, texto para braille, o conteúdo extraído via OCR ou enviado diretamente passa por uma normalização, que inclui a conversão para minúsculas, o tratamento de hífens de quebra de linha e a remoção de caracteres não imprimíveis. A tradução é executada pela biblioteca *Liblouis*[6], configurada com a tabela pt-pt-g1 (português grau 1), e gera tanto a representação visual das células braille quanto a codificação ASCII necessária para a comunicação com hardwares específicos.

Para finalizar o ciclo, o sistema encapsula o conteúdo traduzido em formatos de arquivo padronizados para distribuição e consumo. Para documentos digitais, como PDF e DOCX, as células braille são dispostas em conformidade com a formatação de 40 caracteres por linha e 20 linhas por página, com espaçamento interlinear de 2 mm, de modo a facilitar a leitura tátil caso o documento seja impresso em papel com relevo. Simultaneamente, o sistema gera arquivos BRF baseados na saída ASCII do *Liblouis*, compatíveis com a maioria das impressoras braille comerciais, além de arquivos TXT simples para correções manuais rápidas.

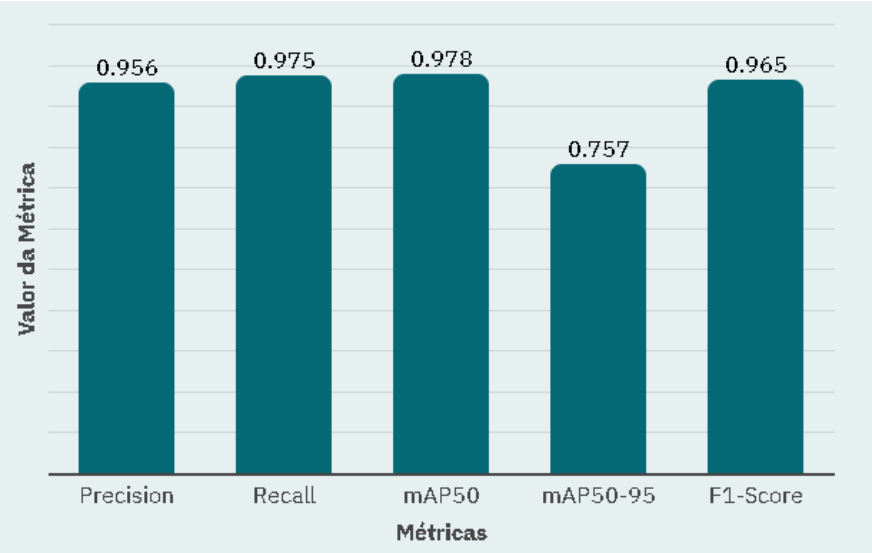
[6] Liblouis é um tradutor e retradutor de braille de código aberto que suporta múltiplos idiomas, incluindo braille literário, computacional e matemático, permitindo personalização por meio de tabelas baseadas em regras ou dicionários.

5 RESULTADOS

5.1 Desempenho do modelo YOLO

O modelo treinado para identificar e classificar células braille apresentou desempenho elevado nas métricas principais, com *precision* de 0,956, *recall* de 0,975, mAP@50 de 0,978, mAP@50-95 de 0,757 e *F1-score* de 0,965, a Figura 7 ilustra essas medidas. Em termos práticos, o alto valor de *precision* indica que a maioria das detecções efetuadas pelo modelo está correta, enquanto o alto *recall* indica que o modelo é capaz de localizar a maior parte das células braille presentes nas imagens.

Figura 7 – Gráfico de avaliação do modelo com todas as classes.



Fonte: Elaborado pelo autor.

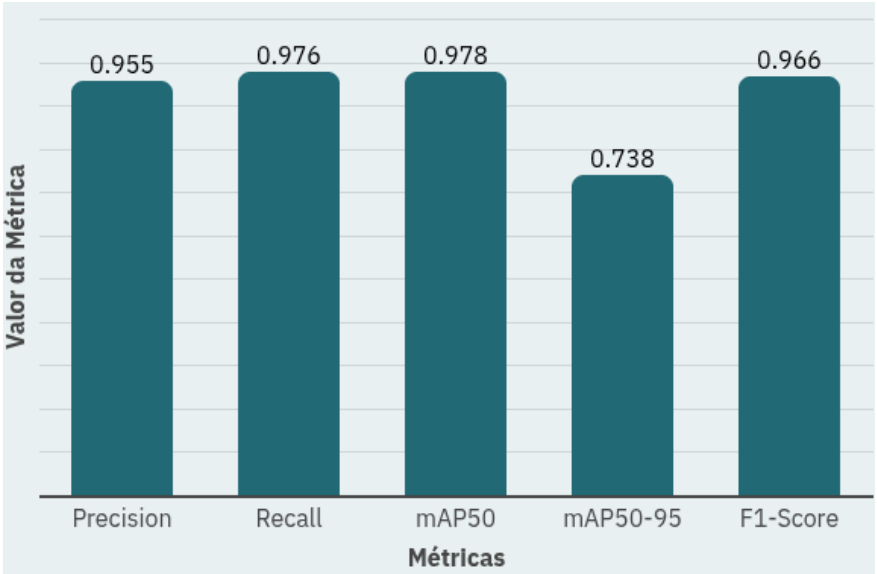
No entanto, o mAP cai quando avaliado em limiares de IoU mais estritos (mAP@50-95 de 0,757), o que indica que em alguns casos as caixas previstas não chegam a um alinhamento perfeito com as caixas de referência, isso pode dificultar a separação e a classificação de células muito próximas.

A Figura 8 apresenta as métricas obtidas pelo modelo treinado sem as classes numéricas, com alcance de *precision* de 0,955, *recall* de 0,976, mAP@50 de 0,978 e *F1-score* de 0,966. Embora o modelo treinado com o *dataset* completo apresente índices ligeiramente superiores em métricas de precisão estrita, como o mAP@50-95 de 0,757 contra 0,738, esses valores puramente numéricos não refletem a maior assertividade do modelo na prática

Para demonstrar essa diferença de comportamento na prática, a comparação das inferências realizadas em uma mesma amostra de teste é fundamental. A Figura 9 apresenta o resultado gerado pelo modelo treinado com o *dataset* completo, no qual o detector gera rótulos confusos na primeira linha e comete erros de classificação com trocas indevidas entre letras e números devido à sobreposição de padrões.

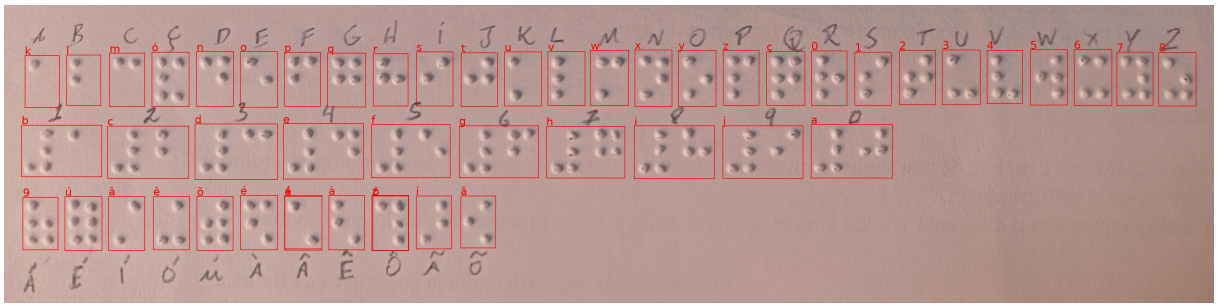
Por outro lado, a Figura 10 exibe o resultado do modelo treinado sem classes numéricas. A imagem mostra uma detecção limpa e correta dos caracteres alfabéticos na primeira e terceira

Figura 8 – Gráfico de avaliação do modelo sem classes numéricas.



Fonte: Elaborado pelo autor.

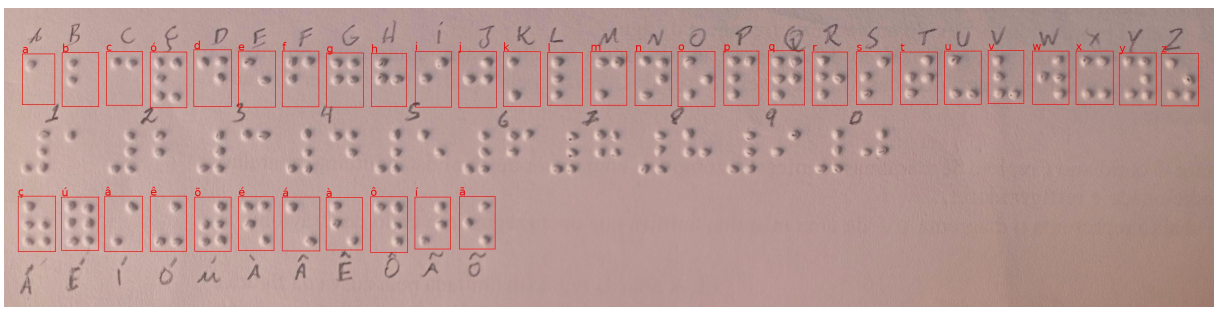
Figura 9 – Detecção com o modelo treinado com todas as classes.



Fonte: Elaborado pelo autor.

linhas. O modelo experimental ignorou completamente a linha de números, o que confirma sua capacidade de filtrar ambiguidades e focar exclusivamente nas classes alfabéticas com alta assertividade.

Figura 10 – Detecção com o modelo sem números.

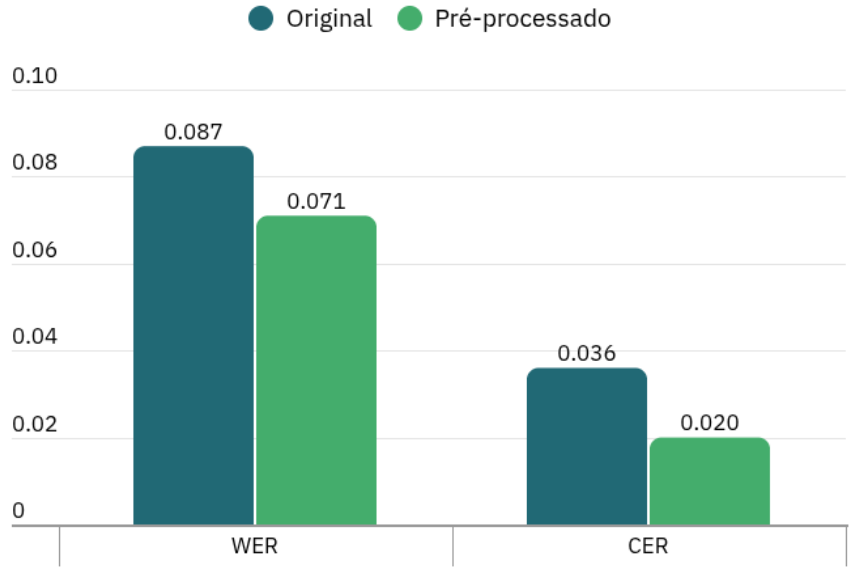


Fonte: Elaborado pelo autor.

5.2 Desempenho do Tesseract OCR

Os resultados da avaliação de desempenho do Tesseract OCR são mostrados na Figura 11. A análise comparativa demonstra claramente o impacto positivo das técnicas de pré-processamento aplicadas.

Figura 11 – Gráfico de avaliação do Tesseract OCR.



Fonte: Elaborado pelo autor.

Para a métrica de WER, observou-se uma redução de 0,087 nas imagens originais para 0,071 nas imagens pré-processadas, o que representa uma melhoria de 18,4%. O ganho de precisão foi ainda mais expressivo ao se analisar o CER, que diminuiu de 0,036 para 0,020, uma redução de 44,4% nos erros.

Esses dados confirmam que o pré-processamento foi eficaz na otimização do reconhecimento de texto, aprimorando significativamente a clareza dos caracteres e a acurácia geral.

5.3 Integração e Resultados da API

Para demonstrar a aplicabilidade prática do sistema, foram realizados testes de integração completa nos *endpoints* desenvolvidos. A validação consistiu no envio de uma imagem real para os *endpoints* e na análise do retorno estruturado.

A Figura 12 apresenta o resultado de uma requisição realizada ao *endpoint* /api/tradutor/braille-texto/. Como observado, o sistema processou a imagem de entrada e retornou um objeto JSON contendo o status da operação e as URLs diretas para o download dos resultados.

Entre os recursos retornados, destaca-se a imagem processada, que oferece um *feedback* visual das inferências do modelo. A Figura 13 exibe a imagem original com as caixas delimitadoras sobrepostas, permitindo a verificação das células braille identificadas.

Por fim, a disponibilidade das URLs na resposta da API permite o acesso imediato ao conteúdo traduzido em múltiplos formatos, visando atender a diferentes necessidades de uso. A

Figura 12 – Exemplo de resposta da API.

Code | Details

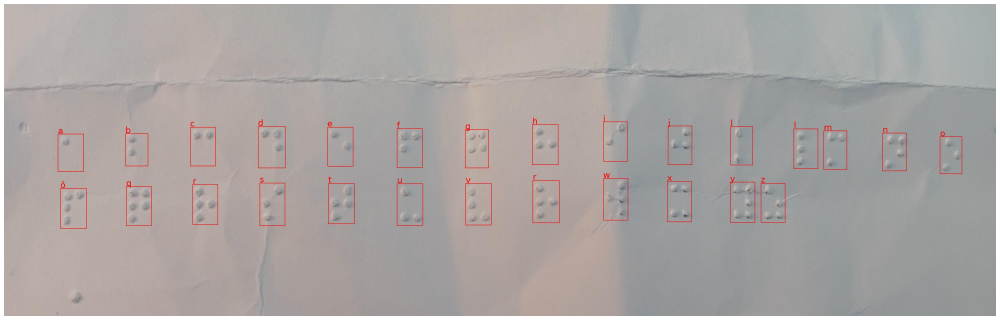
200

Response body


```
[
  {
    "imagem": "http://localhost:8000/media/imagens/processed_imagem.jpg",
    "pdf": "http://localhost:8000/media/exportacoes/imagem.jpg.pdf",
    "docx": "http://localhost:8000/media/exportacoes/imagem.jpg.docx",
    "txt": "http://localhost:8000/media/exportacoes/imagem.jpg.txt"
  }
]
```


Fonte: Elaborado pelo autor.

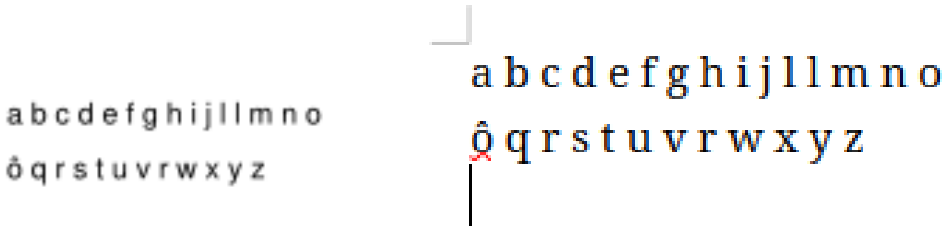
Figura 13 – Visualização das células braille detectadas na imagem processada.



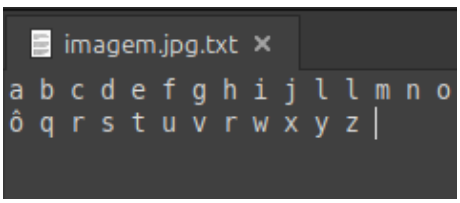
Fonte: Elaborado pelo autor.

Figura 14 demonstra a materialização desse resultado, exibindo os arquivos finais baixados e abertos. É possível observar a consistência do texto traduzido entre as versões geradas, o PDF e o DOCX (a e b), que preservam a formatação estrutural para leitura e impressão, e o arquivo TXT (c), que oferece o texto puro para edições rápidas.

Figura 14 – Resultados obtidos: (a) Visualização em PDF, (b) Arquivo editável DOCX e (c) Texto puro TXT.



(a) (b)



(c)

Fonte: Elaborado pelo autor.

Em seguida, a Figura 15 apresenta o resultado de uma requisição realizada ao *endpoint* /api/tradutor/texto-braille/. Como observado, o sistema processou o documento de entrada e

retornou um objeto JSON contendo o status da operação e as URLs diretas para o download dos resultados.

Figura 15 – Exemplo de resposta da API.

Code | Details

200

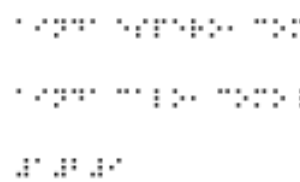
Response body


```
{
  "pdf": "http://localhost:8000/media/exportacoes/77353779_431973037491632_8834832044156467179_n.pdf",
  "docx": "http://localhost:8000/media/exportacoes/77353779_431973037491632_8834832044156467179_n.docx",
  "txt": "http://localhost:8000/media/exportacoes/77353779_431973037491632_8834832044156467179_n.txt",
  "brf": "http://localhost:8000/media/exportacoes/77353779_431973037491632_8834832044156467179_n.brf"
}
```

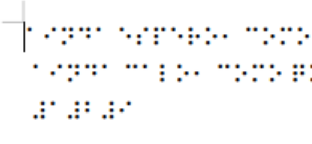

Fonte: Elaborado pelo autor.

Os links retornados permitem o acesso aos documentos convertidos em múltiplos formatos, garantindo flexibilidade para diferentes usos. A Figura 16 exibe os resultados baixados, o PDF (a), o DOCX (b) e o TXT (c) apresentam a diagramação visual dos pontos para conferência visual e o arquivo BRF (d) entrega o formato binário padrão pronto para ser enviado a impressoras braille.

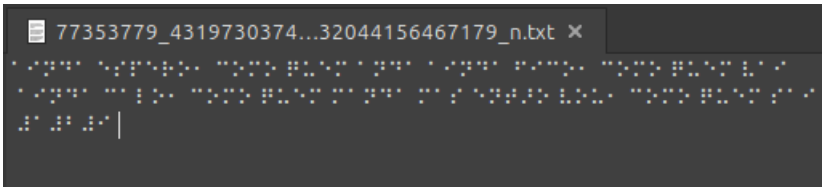
Figura 16 – Resultados obtidos: (a) Visualização em PDF, (b) Arquivo editável DOCX, (c) Texto puro TXT e (d) Arquivo BRF



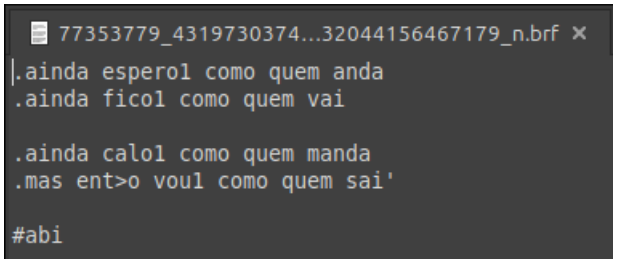
(a)



(b)



(c)



(d)

Fonte: Elaborado pelo autor.

6 CONCLUSÃO

Este trabalho propôs e implementou um sistema de tradução automática entre texto alfabético e braille e vice-versa com o objetivo de ampliar a oferta de materiais acessíveis a pessoas com deficiência visual. A arquitetura baseada em API centraliza o processamento e permite que diferentes aplicações consumam os módulos de tradução, o que facilita integração e escalabilidade.

A validação mostrou que a semelhança visual entre os números e as letras de 'a' a 'j' no braille dificulta a detecção automática. Embora o modelo treinado com todas as classes tenha apresentado métricas altas, a análise prática provou que a remoção das classes numéricas trouxe resultados superiores. O modelo experimental eliminou as confusões de classificação e garantiu uma tradução correta para as letras, o que valida a estratégia de priorizar a qualidade da tradução acima de pequenas diferenças nas estatísticas de precisão.

No módulo de reconhecimento de texto impresso, as técnicas de pré-processamento de imagem mostraram-se decisivas. A aplicação de filtros e conversão para escala de cinza reduziu a Taxa de Erro de Caractere em 44,4%. Isso resultou em um índice final de 2,0% e uma Taxa de Erro de Palavra de 7,1%, o que assegura a qualidade da entrada para a conversão em braille.

Como contribuição prática, o sistema gera saídas compatíveis com impressoras braille e leitores de tela, como BRF, TXT, PDF e DOCX. Para trabalhos futuros, destacam-se a reintegração das classes numéricas via pós-processamento contextual para resolver ambiguidades e a expansão do suporte ao braille grau 2. Além disso, propõe-se o desenvolvimento de interfaces gráficas que consumam a API. Assim, o trabalho entrega um sistema validado que equilibra precisão técnica e utilidade prática, consolidando-se como uma ferramenta eficaz para a produção de conteúdo acessível.

REFERÊNCIAS

ALSULTAN, O. K. T.; MOHAMMAD, M. T. A Deep Learning-Based Assistive System for the Visually Impaired Using YOLO-V7. **Revue d'Intelligence Artificielle**, v. 37, n. 4, p. 901–906, ago. 2023. ISSN 0992499X, 19585748. Disponível em: <https://iieta.org/journals/ria/paper/10.18280/ria.370409>.

DJANGO SOFTWARE FOUNDATION. **Django Overview**. 2025. Acesso em: 18 de abril de 2025. Disponível em: <https://www.djangoproject.com/start/overview/>.

GARAI, S. K. et al. A Novel Method for Image to Text Extraction Using Tesseract-OCR. **American Journal of Electronics & Communication**, v. 3, n. 2, p. 8–11, out. 2022. ISSN 2690-2087. Disponível em: <https://www.ingentaconnect.com/content/10.15864/ajec.3202>.

GOMES, J. V. E. Detecção de objetos com a arquitetura YOLO. 2022. Disponível em: <http://www.monografias.ufop.br/handle/35400000/4746>.

HSU, B.-M. Braille recognition for reducing asymmetric communication between the blind and non-blind. **Symmetry**, v. 12, n. 7, 2020. ISSN 2073-8994. Disponível em: <https://www.mdpi.com/2073-8994/12/7/1069>.

HUSSAIN, M. YOLO-v1 to YOLO-v8, the Rise of YOLO and Its Complementary Nature toward Digital Manufacturing and Industrial Defect Detection. **Machines**, v. 11, n. 7, p. 677, jun. 2023. ISSN 2075-1702. Disponível em: <https://www.mdpi.com/2075-1702/11/7/677>.

IBGE. **Projeção da População do Brasil**. 2025. <https://www.ibge.gov.br/apps/populacao/projecao/index.html>. Acesso em: 01 de maio de 2025.

JIMéNEZ, J. et al. Biography of Louis Braille and Invention of the Braille Alphabet. **Survey of Ophthalmology**, v. 54, n. 1, p. 142–149, jan. 2009. ISSN 00396257. Disponível em: <https://linkinghub.elsevier.com/retrieve/pii/S0039625708001860>.

JUBEH, K. A.; DARD, B.; ZAYED, Y. **Accessibility GO! A Guide to Action**. World Blind Union and CBM Global Disability Inclusion, 2020. Acesso em: 2 maio 2025. Disponível em: <https://worldblindunion.org/wp-content/uploads/2021/12/Accessibility-GO-A-Guide-to-Action-WBU-CBM-Global-Dec2021.pdf>.

OVODOV, I. G. **Optical Braille Recognition Using Object Detection CNN**. 2020. Disponível em: <https://arxiv.org/abs/2012.12412>.

PATIENCE, O. O. et al. Enhanced Text Recognition in Images Using Tesseract OCR within the Laravel Framework. **Asian Journal of Research in Computer Science**, v. 17, n. 9, p. 58–69, set. 2024. ISSN 2581-8260. Disponível em: <https://journalajrcos.com/index.php/AJRCOS/article/view/499>.

RAJ, S. S. **Braille Convertor**. 2022. <https://doi.org/10.7939/r3-x9mm-9q24>. Concordia University of Edmonton, Master of Science in Information Technology Project Report.

RIBEIRO, W. S. et al. Crack detection in buildings using the yolo v8 network. **Revista ALCONPAT**, v. 14, n. 3, p. 288–298, Sep. 2024. Disponível em: <https://revistaalconpat.org/index.php/RA/article/view/765>.

SANTOS, F. C. d. **Grafia braille para língua portuguesa**. [S.l.]: Sec. de Educ. Continuada, Alfabetiz., Diversid. e Inclusão, 2018. ISBN 978-85-7994-092-7.

TEIXEIRA, G. et al. Conversor braile: Um aporte ao processo de alfabetização de pessoas com deficiência visual. In: **Anais do 4º Seminário Nacional de Inclusão Digital**. Passo Fundo, RS, Brasil: [s.n.], 2016. Apresentado no 4º Seminário Nacional de Inclusão Digital.

ULTRALYTICS. **Ultralytics YOLOv11 Docs**. 2025. <http://docs.ultralytics.com/pt/models/yolo11/>. Acesso em: 15 de setembro de 2025.

WANG, J. A Study of The OCR Development History and Directions of Development. **Highlights in Science, Engineering and Technology**, v. 72, p. 409–415, dez. 2023. ISSN 2791-0210. Disponível em: <https://drpress.org/ojs/index.php/HSET/article/view/13285>.

YAMIN, R. B.; KUSUMA, H.; TASRIPAN. Image processing-based braille document copying system as additional feature for ITS's braille printer. **Procedia Computer Science**, v. 197, p. 230–237, 2022. ISSN 18770509. Disponível em: <https://linkinghub.elsevier.com/retrieve/pii/S1877050921023607>.