Universidade Federal do ABC
Centro de Engenharia, Modelagem e Ciências Sociais Aplicadas
Trabalho de Graduação em Engenharia de Informação

Desenvolvimento de um aplicativo móvel baseado em visão computacional para telemonitoramento de dados fisiológicos envolvendo o cuidado híbrido

Fabiano Fernandes de Avelar

Santo André - SP
2025

Fabiano Fernandes de Avelar

Desenvolvimento de um aplicativo móvel baseado em visão computacional para telemonitoramento de dados fisiológicos envolvendo o cuidado híbrido

Trabalho de Graduação apresentado ao curso de Engenharia de Informação da Universidade Federal do ABC, como parte dos requisitos necessários para a obtenção do grau de Bacharel em Engenharia de Informação.

Orientador: Prof. Dr. André Kazuo Takahata

Santo André - SP

2025

Resumo

Com o avanço da telemedicina e a crescente adoção de soluções digitais para o acompanhamento remoto de pacientes, torna-se essencial o desenvolvimento de métodos eficazes para validação e processamento de dados clínicos coletados por dispositivos móveis. Nesse cenário, o cuidado híbrido que integra o atendimento presencial com o monitoramento remoto emerge como uma abordagem promissora para ampliar o acesso à saúde e otimizar a continuidade do cuidado. Neste contexto, o presente trabalho propõe o desenvolvimento de um protótipo de aplicativo móvel voltado para aplicações de biotelemetria em ambientes de mHealth. O objetivo central é permitir a aquisição e reconhecimento de valores de dados fisiológicos a partir de imagens capturadas pela câmera do telefone celular do paciente, reduzindo erros na análise de dados por parte dos profissionais de saúde e promovendo maior confiabilidade no acompanhamento remoto. O aplicativo foi projetado para capturar imagens de dispositivos médicos de uso doméstico, como balanças digitais, oxímetros, medidores de pulso e de pressão arterial. As imagens capturadas são processadas localmente com o uso da biblioteca ML Kit, fornecida pelo Google, utilizando o módulo de OCR (Reconhecimento Óptico de Caracteres) para extrair os dados exibidos nos visores dos dispositivos. Além disso, foi treinado e integrado ao sistema um modelo customizado, desenvolvido com TensorFlow, capaz de reconhecer caracteres em displays de sete segmentos com acurácia de 99,16%, atuando como solução de *fallback* em casos de falha do OCR padrão. A solução proposta visa oferecer uma alternativa de baixo custo para monitoramento remoto de pacientes, ampliando o acesso à saúde digital, ao mesmo tempo em que proporciona maior conforto e autonomia aos usuários que necessitam de acompanhamento contínuo. O sistema reforça a importância da biotelemetria como ferramenta complementar no cenário da saúde digital e dos cuidados híbridos, promovendo agilidade, segurança e eficiência no processo de coleta e análise de dados clínicos à distância.

Palavras-chaves: Processamento de imagem; Biotelemetria; mHealth; Saúde digital; Display de medidores; Display de sete segmentos.

Abstract

As telemedicine advances and digital solutions for remote patient monitoring are increasingly adopted, it becomes essential to develop effective methods for validating and processing clinical data collected by mobile devices. In this scenario, hybrid care models, which integrate in-person consultations with remote monitoring, emerge as a promising approach to expand access to healthcare and enhance continuity of care. In this context, this work proposes the development of a prototype mobile application aimed at biotelemetry applications in mHealth environments. The main goal is to enable automatic acquisition and validation of images captured by the patient's smartphone camera, reducing errors in data analysis by healthcare professionals and promoting greater reliability in remote monitoring.

The application was designed to capture images of household medical devices, such as digital scales, oximeters, pulse and blood pressure monitors. The captured images are processed locally using the ML Kit library provided by Google, employing the OCR module (Optical Character Recognition) to extract the data displayed on the device screens. In addition, a customized model developed with TensorFlow was trained and integrated into the system, which is capable of recognizing characters on seven-segment displays with greater accuracy, acting as a fallback solution in cases of failure of the standard OCR.

The proposed solution aims to provide a low-cost alternative for remote patient monitoring, expanding access to digital health while providing greater comfort and autonomy to users who require continuous follow-up. The system reinforces the importance of biotelemetry as a complementary tool in connected health and hybrid care scenario, promoting agility, security, and efficiency in the process of collecting and analyzing clinical data remotely.

Keywords: Image processing; Biotelemetry; mHealth; Digital health; Medical device display; Seven-segment display.

Lista de ilustrações

Lista de tabelas

Sumário

1 Introdução

A pandemia de COVID-19 representou um marco nas transformações em diversas áreas da sociedade, especialmente na saúde. Durante esse período, a necessidade de distanciamento social e a alta demanda por serviços médicos impulsionaram a adoção de soluções tecnológicas, destacando-se a telemedicina (FREIRE et al., 2023), que é a aplicação das tecnologias da informação e comunicação para viabilizar a prestação de serviços médicos à distância, especialmente em situações em que a localização geográfica representa uma barreira ao atendimento presencial (MALDONADO; MARQUES; CRUZ, 2016), antes considerada uma alternativa complementar, tornou-se um pilar importante para garantir o acesso aos cuidados de saúde (FREIRE et al., 2023), permitindo consultas e monitoramento remoto de pacientes por meio de plataformas digitais.

Entretanto, a expansão da telemedicina também deu origem a um novo paradigma conhecido como cuidado híbrido, que combina o atendimento presencial com o acompanhamento remoto de pacientes. Essa abordagem busca integrar as vantagens do contato físico direto essencial para determinados diagnósticos e procedimentos com os benefícios do monitoramento digital e do suporte contínuo proporcionado pelas tecnologias de saúde conectada. O cuidado híbrido favorece a continuidade do tratamento, a redução de internações desnecessárias e uma maior autonomia do paciente, ao mesmo tempo em que fortalecem a comunicação entre profissionais e usuários do sistema de saúde (NETTO, 2020).

A telemedicina, no entanto, é um componente fundamental de um conceito mais abrangente: a eHealth (saúde eletrônica). A eHealth engloba o uso de tecnologias da informação e comunicação (TICs) para melhorar a saúde e o bem-estar dos indivíduos em sentido amplo. Isso inclui não apenas a prestação de serviços de saúde à distância (como a telemedicina), mas também a digitalização de registros médicos (prontuários eletrônicos), o gerenciamento de informações de saúde, a educação em saúde online, a pesquisa médica e até mesmo sistemas de suporte à decisão clínica. Em essência, a eHealth busca otimizar a coleta, armazenamento, recuperação e uso de informações em saúde, com o objetivo de aprimorar a qualidade dos serviços, a segurança do paciente, a eficiência operacional e a sustentabilidade dos sistemas de saúde.

Dentro do vasto escopo da eHealth, um subdomínio que ganhou proeminência é a mHealth (saúde móvel). A mHealth refere-se especificamente à prática da medicina e da saúde pública suportada por dispositivos móveis, como *smartphones*, *tablets*, *smartwatches* e outros *wearables* (dispositivos vestíveis) (ROCHA et al., 2016). Esta modalidade capitaliza a onipresença e a capacidade de processamento dos dispositivos móveis para oferecer

uma gama de serviços e funcionalidades. Isso inclui desde aplicativos para gerenciamento de doenças crônicas, acompanhamento de sinais vitais, lembretes de medicação, educação para a saúde, até plataformas para teleconsultas e monitoramento remoto de pacientes (AMIRI; NADRI; BAHAADINBEIGY, 2023). A mHealth se destaca por sua portabilidade, acessibilidade e capacidade de integrar-se ao cotidiano do usuário, promovendo o autocuidado e a adesão ao tratamento de forma mais engajada e personalizada (ALKHUZAIMI et al., 2025).

Fora do contexto da pandemia, tanto a eHealth quanto a mHealth, com a telemedicina e o cuidado híbrido, consolidam-se como ferramentas cruciais para ampliar o acesso à saúde, especialmente em regiões remotas ou com escassez de profissionais especializados. Além de reduzir barreiras geográficas, essas abordagens promovem maior eficiência nos atendimentos, diminuindo o tempo de espera e os custos associados a deslocamentos (BASHSHUR et al., 2014). A integração da saúde digital ao cotidiano dos sistemas de saúde possibilita a realização de diagnósticos precoces, o acompanhamento de condições crônicas e o fortalecimento da relação entre pacientes e profissionais, tornando-a uma estratégia sustentável e acessível.

No cenário brasileiro, a pertinência dessas características é ainda mais acentuada devido às vastas dimensões territoriais e à profunda desigualdade regional na distribuição de recursos e profissionais de saúde (ALMEIDA; PROTASIO; REIS, 2021). Assim, a eHealth e a mHealth se apresentam como instrumentos fundamentais para a consolidação do princípio da universalidade do Sistema Único de Saúde (SUS). Isso se deve à sua capacidade de reduzir as barreiras geográficas o que é crítico em um país onde populações ribeirinhas, indígenas e de áreas rurais enfrentam grandes desafios para acessar consultas especializadas. O uso da telemedicina permite que um especialista, alocado em um grande centro urbano, realize diagnósticos, telediagnósticos e monitoramentos a distância, sendo estratégico na mitigação da carência de profissionais em regiões interioranas e remotas e em regiões mais remotas (NICHIATA; PASSARO, 2023). Além disso, a otimização dos processos através de plataformas digitais contribui para a eficiência e economicidade do SUS, viabilizando o acompanhamento de condições crônicas e a medicina preventiva de forma mais capilarizada e custo-efetiva (OLIVEIRA et al., 2023), alinhada com as necessidades de um sistema público de grande escala.

Dentro da telemedicina, o telemonitoramento se destaca como uma subárea voltada ao acompanhamento remoto de pacientes, com destaque para aqueles com doenças crônicas, em suas residências. Essa abordagem reduz custos associados à hospitalização, garante reação rápida a emergências e promove conforto e independência para os pacientes.

Os sistemas de telemonitoramento utilizam tecnologias que permitem a observação, medição e avaliação contínua dos sinais vitais e condições de saúde à distância. Essas ferramentas, muitas vezes associadas à biotelemetria, que é definida como a transmissão

de sinais biológicos e fisiológicos de um local remoto para outro com capacidade de análise dos dados, contribuem significativamente para uma gestão mais proativa e integrada da saúde (NETTO; TATEYAMA, 2018).

Além disso, o telemonitoramento possibilita a coleta contínua de dados, o envio de alertas personalizados e o apoio ao autocuidado, promovendo maior adesão aos tratamentos, facilitando a comunicação entre pacientes e profissionais de saúde, e resultando em uma melhora na qualidade de vida.

Apesar dos avanços na telemedicina e mHealth, a coleta regular de dados fisiológicos em ambiente doméstico ainda apresentam desafios. Muitos dispositivos médicos amplamente utilizados pelos pacientes no Brasil, como oxímetros de pulso, medidores de pressão arterial e glicosímetros, exibem as medições em displays digitais, mas, via de regra, não possuem qualquer tipo de conectividade wireless ou interface de comunicação integrada (FINNEGAN et al., 2019). Isso obriga pacientes ou cuidadores a realizar a transcrição manual das informações, um processo suscetível a erros de digitação, omissões e atrasos no registro, o que pode comprometer a precisão do acompanhamento e a agilidade na tomada de decisões clínicas. Nesse cenário, a simples digitação de valores pelo paciente também pode introduzir vieses ou imprecisões.

Diante desse contexto, este projeto propõe o desenvolvimento de um protótipo de aplicativo móvel capaz de capturar imagens de displays de medidores de sinais vitais (como oxímetros, medidores de pressão, etc.) e converter a medição em dados textuais digitais. A escolha da captura por imagem, em substituição à inserção manual, é crucial para a confiabilidade do sistema, pois reduz significativamente a possibilidade de erros humanos na transcrição dos dados. Ao automatizar a leitura visual, o aplicativo assegura que as informações transmitidas aos profissionais da saúde sejam um reflexo mais fiel das medições do dispositivo, aumentando a integridade e a credibilidade do telemonitoramento. Essa funcionalidade oferece uma solução segura e acessível para a digitalização e posterior envio desses dados aos profissionais da saúde, buscando promover uma melhoria significativa no acompanhamento de pacientes com doenças crônicas não transmissíveis.

2 Telemedicina

A telemedicina consiste na aplicação de tecnologias de comunicação e informação para viabilizar a troca remota de dados e informações médicas entre diferentes localidades. Seu principal objetivo é ampliar o acesso aos serviços de saúde, permitindo que regiões com infraestrutura médica limitada possam receber suporte de centros especializados. De acordo com a definição da Organização Mundial da Saúde (OMS), telemedicina refere-se à prestação de serviços relacionados ao cuidado com a saúde em contextos nos quais a distância representa um fator crítico. Esses serviços são realizados por profissionais da área médica, por meio do uso de recursos tecnológicos que possibilitam o intercâmbio de informações confiáveis para finalidades como diagnóstico, prevenção, tratamento de doenças, educação continuada dos profissionais de saúde, além de suporte à pesquisa e à avaliação de políticas e práticas médicas (WORD HEALTH ORGANIZATION, 2016).

O campo da telemedicina abrange diversas subáreas, cada uma com aplicações específicas no cuidado à saúde. Entre essas, destacam-se: telediagnóstico, que permite a emissão de laudos médicos a distância; telecirurgia, que envolve a realização de procedimentos cirúrgicos remotos com o auxílio de sistemas robóticos; teleconsulta, que viabiliza o atendimento médico em tempo real entre paciente e profissional de saúde; teleinformação, voltada à disseminação de conteúdos educativos e informativos; telereabilitação, focada no acompanhamento remoto de terapias físicas e cognitivas; teleassistência, que oferece suporte contínuo ao paciente fora do ambiente hospitalar; e telemonitoramento, utilizado para o acompanhamento remoto de sinais vitais e outras variáveis clínicas por meio de sensores e dispositivos conectados (NETTO; TATEYAMA, 2018). A Tabela 1, a seguir, mostra as subáreas da telemedicina segundo a Resolução do Conselho Federal de Medicina (CFM) nº 2.314/2022 e suas respectivas aplicações no contexto assistencial (CFM, 2022).

Tabela 1 – Subáreas da telemedicina segundo a Resolução CFM nº 2.314/2022

Subárea	**Descrição**	**Aplicação**
Telediagnóstico	Emissão de laudos ou pareceres médicos à distância, por médico com registro de especialista e com dados/exames digitais.	Diagnóstico em radiologia, cardiologia, patologia, entre outros, sem necessidade de presença física do especialista.
Telecirurgia	Realização de procedimento cirúrgico remoto, mediado por sistemas robóticos e TDICs.	Cirurgias assistidas à distância, com suporte remoto a cirurgião local em centros especializados.
Teleconsulta	Consulta médica não presencial, mediada por tecnologia, entre médico e paciente em locais distintos.	Avaliação clínica, acompanhamento de doenças crônicas, orientação médica em tempo real ou assíncrona.
Teleinformação	Disseminação de conteúdo educativo ou informativo de saúde por meio de TDICs.	Campanhas de saúde, educação contínua de pacientes e populações, divulgação de protocolos clínicos.
Telereabilitação	Acompanhamento remoto de terapias físicas e cognitivas, orientadas por profissionais.	Sessões de fisioterapia, fonoaudiologia e terapia ocupacional realizadas por videoconferência ou apps.
Teleassistência	Suporte contínuo ao paciente fora do ambiente hospitalar, com interação médica ou de equipe multidisciplinar.	Suporte ao paciente domiciliado ou no pós-operatório, com assistência e orientação remota.
Telemonitoramento	Vigilância remota de sinais vitais ou dados clínicos por meio de dispositivos conectados, sob supervisão médica.	Monitoramento de pressão, glicemia, sinais cardíacos em pacientes crônicos ou em acompanhamento domiciliar.

Fonte: Elaborado pelo autor com base na Resolução CFM nº 2.314/2022.

Entre os diversos recursos que integram a telemedicina, destaca-se a biotelemetria, uma área que combina os conceitos de biometria (medição de sinais vitais e parâmetros fisiológicos) com telemetria (transmissão remota de dados). A biotelemetria é definida como a transmissão de sinais biológicos e fisiológicos de uma localização remota para uma localidade com capacidade de receber e analisar os dados (GÜLER; ÜBEYLI, 2002). Essa tecnologia é uma ferramenta com potencial promissor para o acompanhamento de pacientes com doenças crônicas ou em recuperação, que requerem a medição periódica de parâmetros como frequência cardíaca, pressão arterial, temperatura corporal, níveis de oxigênio no sangue, atividade elétrica do coração (ECG), entre outros.

A utilização da biotelemetria possibilita o envio desses dados para sistemas informatizados ou diretamente para profissionais de saúde, com o objetivo de facilitar uma

resposta clínica mais eficaz diante de alterações no estado fisiológico do paciente e contribui para a otimização dos recursos hospitalares, reduzindo a necessidade de deslocamentos frequentes e permitindo um uso mais eficiente do tempo das equipes médicas.

2.1 mHealth

A Organização Mundial da Saúde (OMS), por meio de seu Observatório Global de eSaúde (Global Observatory for eHealth), define mHealth como a prática médica e de saúde pública apoiada por dispositivos móveis, incluindo telefones celulares, tablets, dispositivos de monitoramento de pacientes, assistentes digitais pessoais (Personal Digital Assistants - PDAs) e outros equipamentos sem fio (KAY; SANTOS; TAKANE, 2011). Esta vertente da saúde digital, inserida no escopo mais amplo da eHealth, tem se consolidado como uma estratégia promissora para enfrentar os desafios dos sistemas de saúde contemporâneos, especialmente em países em desenvolvimento.

O uso de tecnologias móveis na área da saúde tem promovido um redesenho significativo na forma como serviços médicos são oferecidos à população. Através de soluções mHealth, tornou-se viável a prestação de cuidados remotos à comunidades que historicamente enfrentam limitações de acesso a estruturas de saúde convencionais, como hospitais ou postos de atendimento básico. Essa realidade é particularmente relevante em regiões rurais, periféricas ou geograficamente isoladas, onde o déficit de infraestrutura e a escassez de profissionais da saúde dificultam a oferta contínua de cuidados. Intervenções baseadas em mHealth têm apresentado impactos positivos no acompanhamento de doenças crônicas e no fortalecimento da atenção básica em países com restrições econômicas e logísticas (BERATARRECHEA et al., 2014).

Esse avanço tem sido impulsionado pela expansão massiva da telefonia móvel. De acordo com o relatório Digital 2025 do DataReportal, estima-se que cerca de 5,81 bilhões de pessoas utilizem telefones celulares no mundo, o que representa aproximadamente 70,7% da população global (DATAREPORTAL, 2025). A disseminação desses dispositivos, aliada à popularização de conexões móveis de dados, criou uma infraestrutura tecnológica acessível e distribuída, capaz de sustentar aplicações de saúde em larga escala. Tal democratização do acesso aos dispositivos móveis tem sido fundamental para a consolidação do mHealth como ferramenta estratégica na promoção da saúde pública.

O avanço da mHealth tem sido impulsionado por uma combinação de fatores cruciais. Primeiramente, as restrições inerentes aos sistemas de saúde em países em desenvolvimento criam uma demanda por soluções inovadoras e acessíveis. Essas limitações, frequentemente ligadas à escassez de recursos, infraestrutura inadequada e barreiras geográficas, tornam a mHealth uma alternativa viável para expandir o acesso a serviços de saúde essenciais.

Em segundo lugar, a rápida e massiva popularização dos aparelhos celulares e

das redes móveis tem sido um catalisador fundamental. Com bilhões de assinantes em todo o mundo, e mais de 70% deles residindo em países de baixa e média renda, a conectividade móvel ultrapassou barreiras, alcançando, inclusive, áreas rurais. A ampla disponibilidade desses dispositivos móveis, por sua vez, abre um vasto potencial para a troca de informações e a prestação de cuidados de saúde de forma mais eficiente e a custos reduzidos, transformando a entrega de serviços de saúde globalmente (WORLD HEALTH ORGANIZATION, 2011).

A eficácia do mHealth não se restringe apenas ao seu potencial teórico; há exemplos concretos de sua aplicação bem-sucedida, com resultados verificáveis. A Organização Mundial da Saúde (OMS), por exemplo, endossou iniciativas como a *Mobile Alliance for Maternal Action* (MAMA) (MAMA, 2010–2016), que utilizou mensagens de texto para fornecer informações cruciais a gestantes e novas mães em países de baixa e média renda. Embora não haja uma única métrica de redução de mortalidade diretamente atribuível à MAMA, a iniciativa demonstrou um aumento significativo na adesão a práticas essenciais para a saúde materno-infantil, como a busca por partos assistidos por profissionais qualificados e o cuidado pós-parto, além de melhorar as taxas de amamentação exclusiva e o uso de mosquiteiros.

No contexto das doenças crônicas, o mHealth tem demonstrado um impacto quantificável e positivo, evidenciado por revisões sistemáticas e ensaios clínicos randomizados (VASLI et al., 2018). No tratamento do diabetes, por exemplo, intervenções baseadas em tecnologias móveis têm levado a uma redução estatisticamente significativa dos níveis de hemoglobina glicada (HbA1c), um biomarcador crucial para o controle glicêmico de longo prazo (ZOU et al., 2020). Esses estudos consistentemente apontam para diminuições adicionais de HbA1c em comparação com os cuidados convencionais, reforçando o valor do mHealth no controle da doença e na prevenção de complicações associadas.

Outra área de sucesso notável é a cessação do tabagismo, onde programas de mHealth que oferecem suporte via mensagens de texto resultaram em taxas de abandono do cigarro significativamente maiores do que as abordagens tradicionais. Um estudo pioneiro demonstrou um impressionante aumento de 10 pontos percentuais na taxa de cessação completa do tabagismo (YBARRA et al., 2013), destacando a habilidade do mHealth em influenciar comportamentos de saúde complexos e, consequentemente, reduzir fatores de risco importantes. A Tabela 2 sintetiza esses exemplos, detalhando o tipo de intervenção e os resultados alcançados.

Tabela 2 – Intervenções de mHealth e Seus Resultados

Exemplo de mHealth	**Tipo de Intervenção**	**Resultados Chave**
Mobile Alliance for Maternal Action (MAMA)	Mensagens de texto informativas para gestantes e puérperas	Aumento no uso de partos assistidos por profissionais de saúde qualificados; duplicação do cuidado pós-parto; melhoria nas taxas de amamentação exclusiva e uso de mosquiteiros.
Manejo do Diabetes Tipo 2	Aplicativos e plataformas móveis para monitoramento e suporte ao controle glicêmico	Redução média de 0,3% a 0,79% nos níveis de hemoglobina glicada (HbA1c).
Cessação do Tabagismo (Programa "*Stop My Smoking USA*")	Programas de apoio via mensagens de texto para jovens adultos	Aumento de 10 pontos percentuais na taxa de cessação completa do tabagismo em 4 semanas.

Fonte: Elaborado pelo autor.

3 Processamento Digital de Imagens

O processamento digital de imagens (PDI) é uma área da engenharia e da ciência da computação que se dedica à análise e manipulação de imagens digitais por meio de algoritmos computacionais. Uma imagem digital pode ser entendida como uma função bidimensional $f(x, y)$, onde cada par de coordenadas (x, y) representa um ponto da imagem e o valor da função corresponde à intensidade luminosa ou nível de cinza naquele ponto. A Figura 1 ilustra essa representação, mostrando como uma imagem contínua (a) é projetada em uma matriz discreta de sensores, resultando na imagem digital (b) após a amostragem e quantização. Nesta imagem, cada elemento da matriz (x, y) corresponde a um pixel, e o valor numérico associado a essa posição representa a intensidade luminosa ou nível de cinza naquele ponto específico. Quando essas intensidades são representadas por valores quantizados e discretos, temos uma imagem digital propriamente dita, composta por pixels, que são os menores elementos de uma imagem.

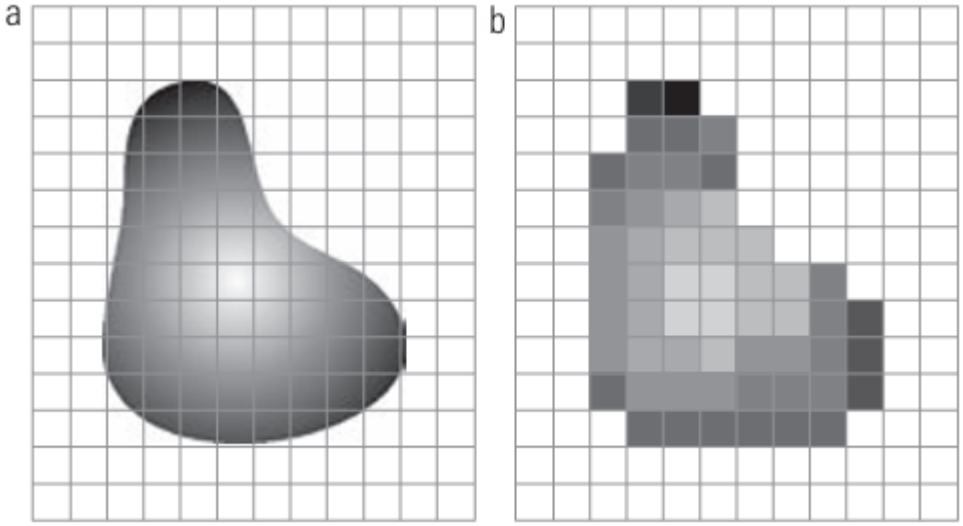

Figura 1 – (a) Imagem contínua projetada em uma matriz de sensores. (b) Resultado da amostragem e quantização da imagem. Fonte: Gonzalez e Woods (2010, p. [35]).

O PDI tem como objetivo melhorar a qualidade visual das imagens para interpretação humana ou extrair informações úteis para análise automática, sendo amplamente utilizado em áreas como medicina, astronomia, indústria, defesa, vigilância e reconhecimento automático.

Segundo Gonzalez e Woods (GONZALEZ; WOODS, 2010), as atividades desenvolvidas no PDI podem ser compreendidas em diferentes níveis de complexidade, que se relacionam diretamente com o tipo de operação realizada e com os resultados esperados. No nível mais elementar, o processamento atua diretamente sobre os pixels da imagem,

promovendo transformações básicas como o realce de contraste, a suavização, a remoção de ruídos e a aplicação de filtros espaciais ou no domínio da frequência. Essas operações têm como característica o fato de que tanto a entrada quanto a saída do processo são imagens, e seu principal objetivo é a melhoria da qualidade visual ou a preparação da imagem para etapas posteriores.

Em um estágio intermediário, o foco se desloca para a extração de atributos relevantes da imagem, tais como bordas, contornos, regiões homogêneas e características texturais. Essas informações, por sua vez, são utilizadas para representar objetos de forma estruturada, permitindo que sejam manipuladas por sistemas computacionais de forma mais eficiente. Esse nível de processamento é fundamental para que a imagem digital deixe de ser apenas uma matriz de intensidades e passe a representar entidades com significado computacional.

Já no nível mais elevado, o processamento visa interpretar os dados extraídos da imagem, promovendo o reconhecimento de padrões e objetos. Nessa etapa, o sistema busca extrair informações acerca do conteúdo da imagem, realizando tarefas como classificação, detecção de objetos e análise contextual. É nesse ponto que o PDI se torna útil para a criação de recursos de visão computacional e de inteligência artificial, integrando-se, assim, a sistemas capazes de realizar inferências e tomar decisões com base em informações visuais.

3.1 Reconhecimento de Objetos

O reconhecimento de objetos representa uma das etapas mais avançadas e desafiadoras dentro do PDI. Esse processo busca identificar automaticamente, em uma imagem, entidades distintas que pertencem a classes previamente definidas, atribuindo a elas rótulos semânticos com base em suas características visuais. A complexidade do reconhecimento decorre tanto da diversidade e variabilidade dos objetos presentes no mundo real quanto das condições sob as quais as imagens são adquiridas, como variações de iluminação, escala, oclusões, ruído e perspectivas geométricas distintas.

Conforme apresentado por Gonzalez e Woods (GONZALEZ; WOODS, 2010), o reconhecimento pode ser entendido como uma extensão natural das etapas anteriores de segmentação e descrição. Ou seja, antes de reconhecer um objeto, é necessário isolá-lo do fundo da imagem e extrair um conjunto significativo de atributos que capturem suas propriedades distintivas. Esses atributos podem ser estruturais, como contornos, formas e simetrias, ou estatísticos, como momentos invariantes, texturas e transformadas espectrais. O reconhecimento propriamente dito consiste então em comparar os padrões extraídos com modelos previamente conhecidos e classificados.

Ao longo das décadas, diversas abordagens teóricas foram propostas para o reco-

nhecimento de objetos. No início, predominavam métodos baseados em decisão estatística, como os classificadores bayesianos, que assumem uma distribuição de probabilidade dos dados para determinar a classe mais provável a partir das características extraídas. Com o avanço da computação e a maior disponibilidade de dados, técnicas de aprendizado supervisionado passaram a dominar o cenário, especialmente aquelas baseadas em redes neurais artificiais. Estas redes, especialmente as convolucionais (CNNs), têm se destacado por sua capacidade de aprender representações hierárquicas dos dados visuais, dispensando, em muitos casos, a necessidade de engenharia manual de atributos. Redes convolucionais são compostas por camadas que atuam como detectores de padrões progressivamente mais complexos, começando por bordas e texturas e culminando em formas específicas que caracterizam um objeto (LECUN; BENGIO; HINTON, 2015).

Nos últimos anos, com a consolidação do campo da visão computacional e da inteligência artificial, o reconhecimento de objetos deixou de ser uma tarefa isolada e passou a integrar sistemas completos de análise de cena, rastreamento e tomada de decisão. Isso é particularmente evidente em aplicações como veículos autônomos, nos quais o reconhecimento em tempo real de pedestres, placas, faixas e outros veículos é fundamental para a segurança e eficácia da navegação (VOULODIMOS et al., 2018). Também em ambientes industriais, sistemas de inspeção visual baseados em redes neurais são capazes de identificar peças defeituosas com precisão superior à humana, mesmo sob condições de alta velocidade ou baixa iluminação (YAN; LIU; YANG, 2021).

É importante destacar que o reconhecimento de objetos não se limita a identificar o que está presente em uma imagem. Muitas aplicações exigem também a sua localização precisa, o que deu origem a métodos como o object detection, que combinam reconhecimento e segmentação. Algoritmos como YOLO (*You Only Look Once*) (REDMON et al., 2016) e SSD (*Single Shot MultiBox Detector*) (LIU et al., 2016) exemplificam essa abordagem, realizando em tempo real a detecção de múltiplos objetos em imagens complexas.

Em síntese, o reconhecimento de objetos se destaca no processamento digital de imagens como a fronteira entre a percepção computacional e a compreensão semântica. Trata-se de uma área que não apenas sintetiza as etapas anteriores do processamento, como aquisição, pré-processamento, segmentação e descrição, mas também projeta essas informações para domínios mais amplos de decisão, interação e inteligência artificial.

4 Trabalhos Relacionados

Além do estudo de (FINNEGAN et al., 2019), diversos trabalhos na literatura têm proposto abordagens para a leitura automática de dígitos em displays de sete segmentos, especialmente em dispositivos médicos de uso domiciliar. Assim como discutido por (MOREIRA; NETO et al., 2022), esses métodos podem ser organizados em duas grandes categorias: (i) abordagens baseadas em técnicas tradicionais de processamento de imagens e (ii) métodos fundamentados em redes neurais profundas.

As abordagens clássicas incluem técnicas descritas por (KANAGARATHINAM; SEKAR, 2019), que utilizam etapas de binarização, operações morfológicas, detecção de contornos e regras heurísticas para determinar quais segmentos estão ativados. Embora apresentem baixo custo computacional, esses métodos são sensíveis a variações de iluminação, reflexos, variações de orientação na captura e inconsistências na geometria dos dígitos, limitações destacadas também por (FINNEGAN et al., 2019), em que é proposto um *pipeline* de pré-processamento elaborado para mitigar tais efeitos.

Com o advento das redes neurais convolucionais (CNNs), métodos baseados em aprendizado profundo tornaram-se predominantes. Em (FINNEGAN et al., 2019) são exploradas redes totalmente conectadas para classificar dígitos segmentados, obtendo resultados robustos, ainda que dependentes de segmentação prévia de alta qualidade. Trabalhos posteriores, como o de (PRAKRUTHI; SRINATH; RAMAKRISHNA, 2017), avançaram nesse sentido ao investigar a aplicação de CNNs diretamente em dispositivos móveis, demonstrando a viabilidade de inferência embarcada para leitura automatizada de equipamentos biomédicos.

Em (MOREIRA; NETO et al., 2022), essa linha é aprofundada ao se propor um sistema baseado em detecção de objetos com modelos *EfficientDet* e *EfficientDet-Lite*, otimizados para execução em *smartphones* por meio do TensorFlow Lite. Seus experimentos reportam acurácia superior a 98% na detecção e classificação de dígitos mesmo em condições reais, evidenciando a relevância de arquiteturas leves com capacidade de operação em tempo real, especialmente em cenários de *mHealth* e telemedicina.

Além disso, estudos recentes demonstram a viabilidade de modelos compactos para reconhecimento visual em dispositivos móveis. Em (ADI; CASSON, 2021) são analisadas estratégias de otimização de redes neurais executadas diretamente em *smartphones* por meio do TensorFlow Lite, destacando o equilíbrio entre acurácia, consumo energético e latência — parâmetros essenciais para aplicações de processamento local no ponto de cuidado. De forma complementar, em (LIU; CHEN; ZHANG, 2021) são investigadas arquiteturas de detecção em tempo real em dispositivos portáteis e com recursos restritos, reforçando que

modelos leves podem operar com eficiência mesmo em ambientes computacionalmente limitados.

À luz dessa literatura, a solução desenvolvida neste trabalho adota uma abordagem híbrida: utiliza o ML Kit como OCR de propósito geral e, paralelamente, um modelo compacto em TensorFlow Lite especializado na classificação de dígitos de sete segmentos, precedido por um *pipeline* de pré-processamento implementado com OpenCV. Essa arquitetura supera limitações de métodos puramente morfológicos e complementa as vantagens das abordagens profundas, proporcionando um sistema robusto, eficiente e totalmente executável em *smartphones*, em consonância com as tendências recentes da visão computacional aplicada à saúde digital.

5 Metodologia

Para validar a proposta desenvolvida neste trabalho, foi implementada uma aplicação móvel compatível com dispositivos que utilizam o sistema operacional Android. O principal objetivo do aplicativo é permitir a leitura automatizada de valores numéricos exibidos em dispositivos médicos, como medidores de pressão arterial, balanças e oxímetros.

A interface do aplicativo oferece ao usuário duas opções para aquisição da imagem: captura em tempo real por meio da câmera do dispositivo ou seleção de uma imagem previamente armazenada na galeria. Após a obtenção da imagem, o usuário é orientado a selecionar manualmente a Região de Interesse (ROI – *Region of Interest*), delimitando especificamente a área que contém os dígitos numéricos no visor do equipamento. Esse procedimento é necessário para o funcionamento adequado do sistema, pois restringe o processamento apenas à área relevante, reduzindo ruídos e aumentando a precisão do reconhecimento.

Com a ROI definida, a imagem é enviada para a API de reconhecimento óptico de caracteres (OCR - *Optical Character Recognition*) da biblioteca Google ML Kit, que realiza o processamento para extração do texto contido na imagem. Caso o reconhecimento seja bem-sucedido, isto é, se o processamento do ML Kit retornar um resultado válido e não uma exceção, os valores extraídos são exibidos na tela do aplicativo e armazenados em variáveis específicas para posterior utilização, como o envio a profissionais da saúde ou o armazenamento local.

Entretanto, em situações em que o ML Kit não consegue identificar corretamente os caracteres, especialmente quando se trata de dígitos apresentados em visores de sete segmentos, como os encontrados em muitos dispositivos médicos, o sistema emprega uma segunda abordagem de reconhecimento. Nessa alternativa, a imagem é processada por um modelo de rede neural artificial previamente treinado com a biblioteca TensorFlow, especificamente desenvolvido para lidar com a estrutura peculiar dos caracteres formados por sete segmentos. Antes de ser submetida ao modelo, a imagem passa por uma etapa de pré-processamento que envolve a conversão para tons de cinza, binarização e normalização, com o objetivo de reduzir ruídos e destacar os contornos dos dígitos, aumentando assim o desempenho do reconhecimento.

Ao final do processo, independentemente da biblioteca utilizada, os valores reconhecidos são apresentados ao usuário de forma estruturada na interface do aplicativo. A Figura 2 ilustra o fluxo completo do processo de reconhecimento e tomada de decisão implementado na aplicação.

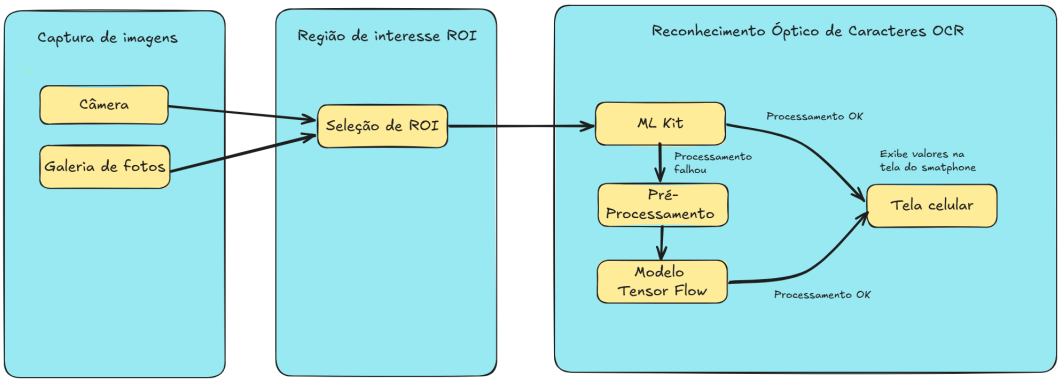

Figura 2 – Fluxo de processamento da aplicação móvel para reconhecimento de dígitos

A seleção das bibliotecas utilizadas no desenvolvimento do sistema foi guiada por critérios de desempenho, compatibilidade com dispositivos móveis e adequação às etapas específicas do *pipeline*. Embora o OpenCV não tenha sido utilizado como solução principal de visão computacional, sua integração no aplicativo desempenha um papel fundamental na preparação das imagens destinadas ao modelo TensorFlow Lite.

O *OpenCV para Android* foi empregado exclusivamente na etapa de pré-processamento das imagens que são enviadas ao modelo TFLite. Após o recorte da Região de Interesse (ROI), a imagem passa por operações como conversão para escala de cinza, normalização, redimensionamento e ajustes estruturais necessários para que o *input* fornecido ao modelo possua o formato e o padrão visual esperados pela rede neural. O uso do OpenCV possibilita a aplicação eficiente de manipulação de matrizes, filtros e transformações, garantindo maior controle sobre a qualidade do sinal visual utilizado na inferência.

Por outro lado, o *ML Kit On-Device OCR* recebe diretamente a imagem recortada da câmera ou da galeria, sem qualquer intervenção do OpenCV. Isso ocorre porque o ML Kit dispõe de um pipeline interno de pré-processamento otimizado para OCR, capaz de operar diretamente sobre bitmaps nativos do Android. Dessa forma, o OpenCV é utilizado apenas no caminho destinado ao processamento especializado do TensorFlow Lite e não interfere na etapa de OCR genérico conduzida pelo ML Kit.

O *TensorFlow Lite*, por sua vez, foi empregado como mecanismo de reconhecimento especializado para leitura de dígitos de sete segmentos, atuando nos casos em que o ML Kit não apresenta confiabilidade suficiente. A combinação entre o OpenCV, responsável por preparar adequadamente a imagem e o TFLite responsável pela inferência permite que o modelo receba dados padronizados, com menor ruído e maior consistência visual, elevando a acurácia do reconhecimento.

Assim, o pipeline de bibliotecas do sistema pode ser resumido da seguinte forma:

- **ML Kit**: realiza OCR direto a partir da imagem recortada, sem uso de OpenCV;

- **TensorFlow Lite**: executa o reconhecimento especializado, dependente do pré-processamento via OpenCV;
- **OpenCV**: utilizado exclusivamente para preparar a imagem destinada ao modelo TFLite, garantindo padronização e redução de ruído.

Essa arquitetura híbrida aproveita as vantagens específicas de cada biblioteca, resultando em uma solução eficiente, leve e adequada ao ambiente móvel, ao mesmo tempo em que garante alta precisão no reconhecimento de dígitos de sete segmentos.

5.1 Aplicativo Kotlin

Para implementar o aplicativo móvel (captura de imagens, seleção de região de interesse e exibição dos resultados), optou-se pela linguagem de programação Kotlin, conhecida por ser de propósito geral e tipagem estática (JETBRAINS, 2010) . Criada pela JetBrains (reconhecida por ferramentas como IntelliJ IDEA e Android Studio), Kotlin foi anunciada em 19 de julho de 2011 no JVM Language Summit (BELLA, 2021) . Inicialmente vista como alternativa moderna ao Java para a JVM (*Java Virtual Machine*), a linguagem rapidamente evoluiu para um verdadeiro ecossistema multiplataforma. Em 2021, por exemplo, a JetBrains destacou que "Kotlin evoluiu de uma alternativa ao Java para um ecossistema completo que permite escrever código para diferentes finalidades", incluindo aplicações para servidores, dispositivos móveis, web e projetos multiplataforma (JETBRAINS, 2010).

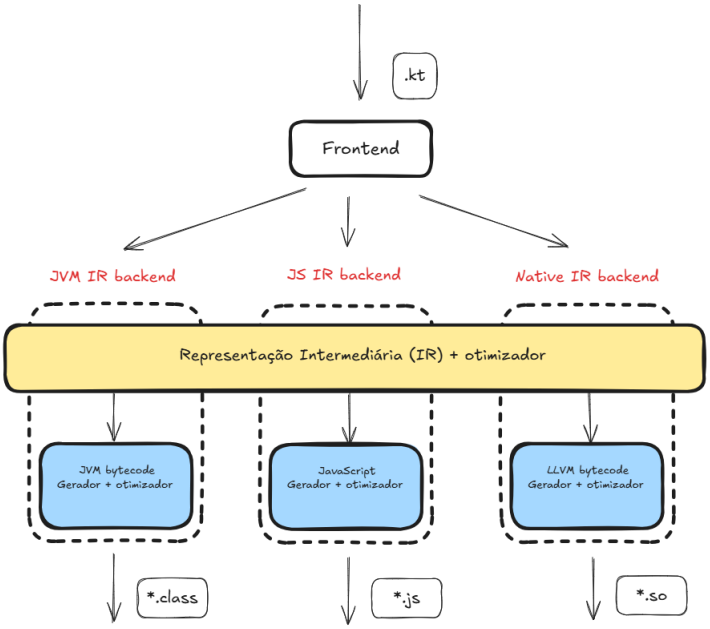

Figura 3 – Compilador Kotlin

A Figura 3 ilustra de forma simplificada esse processo de compilação do Kotlin, evidenciando como o código-fonte é convertido em uma representação intermediária (IR) que pode ser otimizada e posteriormente direcionada para diferentes backends (JVM, JavaScript ou nativo), garantindo a flexibilidade e a portabilidade da linguagem em múltiplas plataformas.

O crescimento de Kotlin no desenvolvimento móvel foi acelerado pelo Google. Em maio de 2017, o time Android anunciou oficialmente o suporte a Kotlin, passando a incluí-lo por padrão no Android Studio 3.0. Na ocasião, o Google elogiou Kotlin como "uma linguagem brilhantemente projetada e madura" que tornaria o desenvolvimento Android mais rápido e agradável (BELLA, 2021). Dois anos depois, no Google I/O de 2019, Kotlin foi declarado a linguagem preferencial para desenvolvimento Android (WINER, 2019). Desde então, mais de 60 % dos desenvolvedores profissionais de Android adotam Kotlin por seu aumento de produtividade e segurança de código (GOOGLE, 2025a). Essa adoção inclui a colaboração entre Google e JetBrains para aprimorar o compilador Kotlin e para introduzir APIs Android "Kotlin-first", consolidando o papel de Kotlin na plataforma.

A linguagem Kotlin foi concebida com uma filosofia altamente pragmática, voltada para facilitar o desenvolvimento e resolver problemas reais dos programadores. Ao invés de propor paradigmas completamente novos, Kotlin reúne e aperfeiçoa conceitos já consagrados em linguagens como Java, Scala e C# (KOTLIN FOUNDATION, 2025a). Por exemplo, herda a sintaxe orientada a objetos familiar do Java, mas adiciona recursos funcionais (como funções de ordem superior e literais lambda) para tornar o código mais expressivo. Conforme a especificação oficial, esse compromisso com a praticidade se reflete no seu princípio fundamental: "uma das ideias principais por trás do Kotlin é ser pragmático, ou seja, ser uma linguagem útil para o desenvolvimento do dia a dia, ajudando o usuário a realizar seu trabalho através de seus recursos e ferramentas" (KOTLIN FOUNDATION, 2025a). Esse foco no desenvolvimento diário mantém um conjunto mínimo de mecanismos poderosos, privilegiando a produtividade e a legibilidade.

As principais características do kotlin são:

- **Tipagem estática e segurança de nulos:** O sistema de tipos de Kotlin distingue, em tempo de compilação, tipos anuláveis e não anuláveis, prevenindo acessos nulos indevidos (KOTLIN FOUNDATION, 2025b).
- **Sintaxe concisa:** Recursos como data classes, parâmetros nomeados e valores padrão, inferência de tipos e funções de extensão reduzem drasticamente o código repetitivo.
- **Interoperabilidade total com Java:** Kotlin foi desenhado para coexistir com código Java no mesmo projeto, permitindo adoção gradual e aproveitamento do vasto ecossistema Java (CLERON, 2017a).

- **Concorrência estruturada:** Com suporte nativo a *coroutines*, Kotlin simplifica a programação assíncrona (ex.: chamadas de rede e operações em segundo plano) de forma eficiente e legível (GOOGLE, 2025b).
- **Ferramentas inteligentes:** O compilador realiza *smart casts* (conversões automáticas de tipo após verificações), *dead code elimination*, e oferece extensões ricas para bibliotecas Android (como Android KTX) que tornam o código mais claro e seguro.

Um dos pilares do Kotlin é a segurança de tipos, especialmente no controle de nulidade. Erros de NullPointerException são combatidos diretamente pelo sistema de tipos da linguagem. Em Kotlin, qualquer variável deve ser explicitamente declarada como anulável (String?) ou não anulável (String); o compilador impede, por padrão, que uma variável não anulável receba null, eliminando assim grande parte dos NPEs (NullPointerException) em tempo de execução (KOTLIN TEAM, 2025). Conforme afirma a documentação oficial, "a segurança de nulidade é um recurso do Kotlin projetado para reduzir significativamente o risco de referências nulas".

Outro aspecto marcante do Kotlin é a sintaxe concisa, que elimina grande parte do código repetitivo (boilerplate). Por exemplo, ao definir uma simples classe de dados em Java (com diversos *getters*, *setters*, construtores, equals(), hashCode() e toString()), o Kotlin permite fazê-la em uma única linha usando uma data class.

Essa concisão reduz erros e acelera o desenvolvimento. Recursos como parâmetros nomeados e valores padrão em funções, lambdas e extensões de biblioteca permitem escrever funcionalidades complexas com código mais enxuto.

Os parâmetros nomeados (*named arguments*) permitem que os argumentos sejam passados para uma função nomeando-os explicitamente na chamada, o que aumenta significativamente a clareza e a legibilidade do código, especialmente em funções com muitos parâmetros ou de tipos semelhantes.

Já os valores padrão (*default parameters*) possibilitam definir um valor padrão diretamente na assinatura da função, tornando certos parâmetros opcionais e reduzindo a necessidade de criar múltiplas versões sobrecarregadas (*overloaded*) da mesma função.

As expressões lambda (*lambda expressions*) representam funções anônimas ou seja, sem nome que podem ser armazenadas em variáveis, passadas como parâmetros para outras funções ou retornadas por elas. Elas simplificam a programação funcional e são amplamente utilizadas em operações com coleções e APIs assíncronas, tornando o código mais conciso, expressivo e fácil de manter.

Por fim, as extensões de biblioteca (*extension functions and properties*) permitem adicionar novas funcionalidades a classes já existentes sem modificar seu código-fonte nem criar subclasses herdadas. Esse mecanismo é amplamente explorado na biblioteca padrão

do Kotlin para incorporar métodos como *map*, *filter* e *forEach* a tipos básicos herdados do Java, promovendo um código mais limpo, coeso e reutilizável.

O Google observa que essas características modernas de linguagem "permitem focar em expressar suas ideias e escrever menos código genérico" (GOOGLE, 2025c), o que resulta em maior rapidez de desenvolvimento e manutenção menos custosa.

O Kotlin foi projetado para interoperar perfeitamente com Java, facilitando a adoção em projetos já existentes. É possível chamar código Java apartir do Kotlin e vice-versa sem a necessidade de *wrappers* manuais, tornando a transição muito suave (CLERON, 2017a). Por exemplo, anotações e convenções automáticas cuidam de detalhes como *getters/setters*, permitindo que *codebases* mistas funcionem lado a lado. Essa integração é fundamental para aproveitar o imenso ecossistema de bibliotecas Java. Desde a IDE, o plugin de Kotlin já vem integrado no Android Studio, até bibliotecas oficiais (como Jetpack e Firebase, que oferecem extensões Kotlin idiomáticas), todo o conjunto de ferramentas Android dá suporte de primeira classe à linguagem (CLERON, 2017b). A JetBrains e o Google têm trabalhado juntos para que as ferramentas de build (Gradle, R8, etc.) e bibliotecas sejam cada vez mais eficazes no Kotlin, o que inclui também APIs novas (como o Jetpack Compose) desenvolvidas com DSLs Kotlin, tornando o ciclo de desenvolvimento ainda mais ágil.

Além disso, o Kotlin incorpora recursos sofisticados que aumentam a clareza do código, o *smart casting*, por exemplo: depois de um teste de tipo ou nulidade, o compilador é capaz de ajustar o tipo da variável sem sintaxe extra. Isso evita a necessidade de *cast* explícito e torna o código mais limpo. Outros recursos incluem *when expressions* (uma versão poderosa de switch), *destructuring declarations* (desempacotamento de propriedades de objetos), *infix functions*, e *extension functions*, que permitem "estender" classes existentes com novos métodos sem herança ou modificar código original.

Para lidar com concorrência, Kotlin introduz corrotinas como elemento de primeira classe. Elas permitem escrever código assíncrono de forma quase sequencial, facilitando tarefas comuns de aplicativos (por exemplo, chamadas de rede ou acesso a banco de dados) com menor risco de erros e mais desempenho. Esse modelo de concorrência estruturada simplifica cenários complexos sem fragmentar o código em callbacks ou lambdas excessivamente aninhados.

Em resumo, Kotlin combina uma base estável e familiar (herdada do Java) com avanços de linguagens modernas. Seu design focado na praticidade do desenvolvedor, na segurança do código e na redução de verbosidade traz ganhos diretos em produtividade e qualidade de software, aspectos cruciais para sistemas de engenharia complexos.

A prova de conceito desenvolvida para esse trabalho utilizou as configurações conforme descrito na Tabela 3.

Tabela 3 – Detalhes do SDK Android

Item	**Detalhes**
Android SDK (API alvo)	35 (Android 15 – Tiramisu)
Versão mínima Android	15 (Ice Cream Sandwich, API Level 15)
Java (OpenJDK)	17
Linguagem de Programação	Kotlin (rodando na JVM, compatível com Java 17)
Bibliotecas/Ferramentas	Google ML Kit (Visão computacional - Text Recognition), TensorFlow Lite, OpenCV, Android Image Cropper, AndroidX AppCompat

Fonte: Elaborado pelo autor.

O aplicativo permite que o usuário selecione uma imagem tanto pela câmera quanto pela galeria. Ao acionar o botão de entrada de imagem, é exibido um menu pop-up oferecendo as opções Câmera ou Galeria. Se o usuário escolher a câmera, o app solicita permissão de uso da câmera (via ActivityResultContracts para verificar a permissão). Com a permissão concedida, cria-se uma entrada no MediaStore (ContentValues com título/descrição) e inicia-se a intenção de tirar uma foto. O URI (*Uniform Resource Identifier*) gerado pelo MediaStore (imageUri) é então passado ao componente de recorte. Se o usuário escolher a galeria, é lançada uma intenção de ACTION PICK no armazenamento externo, permitindo navegar nas imagens disponíveis. Após a imagem ser obtida (seja pela câmera ou galeria), seu URI é atribuído à CropImageView através de setImageUriAsync(uri), iniciando o fluxo de recorte.

A **CropImageView** (da biblioteca Android Image Cropper) permite que o usuário defina interativamente uma Região de Interesse (ROI) na imagem. O usuário pode ampliar, rotacionar e arrastar a seleção para ajustar o retângulo de recorte. Quando o corte é finalizado, um *listener* assíncrono captura o bitmap recortado e o exibe novamente no CropImageView. Em caso de erro no recorte, é exibida uma mensagem Toast informando a falha. O Toast é um componente do Android utilizado para exibir notificações curtas e não intrusivas ao usuário, aparecendo temporariamente sobre a interface do aplicativo. Dessa forma, caso ocorra uma falha durante o processo de recorte, o aplicativo comunica imediatamente o problema ao usuário de maneira rápida, discreta e intuitiva, preservando a continuidade da experiência de navegação.

Após o recorte da imagem, o usuário aciona o botão de reconhecimento de texto. Nesse momento, o aplicativo inicia o processamento da imagem para extrair texto da região selecionada. O bitmap recortado é convertido em um objeto InputImage do ML Kit, e um cliente de reconhecimento de texto (TextRecognizer) é instanciado com opções padrão para línguas latinas. O ML Kit então processa a imagem assincronamente. Cada sequência

de dígitos extraída é então formatada em variáveis numeradas (var1 = dígito, var2 = dígito, etc.) e exibida numa TextView de resultados. Todos os eventos são registrados no *log* de depuração para análise posterior conforme Figura 4.

Entretanto, caso o ML Kit falhe ou não consiga identificar corretamente os caracteres, o aplicativo recorre a um modelo desenvolvido com o TensorFlow e convertido para o formato TensorFlow Lite. Esse modelo foi treinado especificamente para reconhecer dígitos exibidos em displays de sete segmentos, como os encontrados em oxímetros e medidores de pressão arterial.

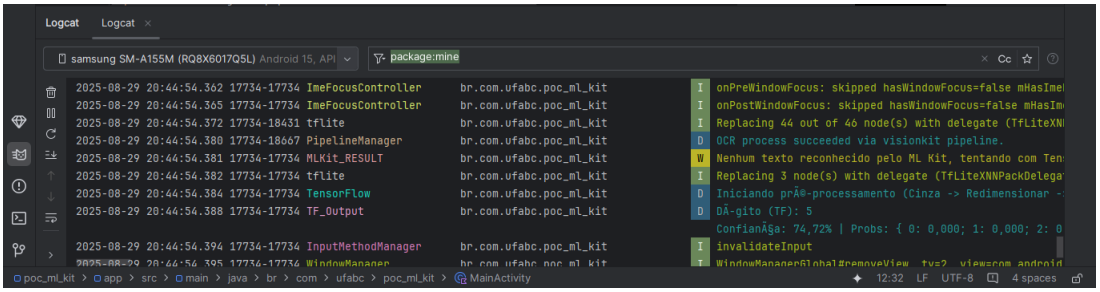

Figura 4 – Logs de depuração aplicativo

O aplicativo também gerencia cuidadosamente as permissões e ciclo de vida da atividade para garantir estabilidade. Antes de acessar a câmera, verifica-se dinamicamente se a permissão foi concedida; caso contrário, solicita-se permissão ao usuário. Se o usuário negar a permissão, é exibido um aviso. A obtenção de imagens da galeria e da câmera usa a API de ActivityResultContracts, evitando métodos obsoletos. No ciclo de vida da MainActivity, uma flag (isActivityActive) é usada para interromper *callbacks* assíncronos (como o retorno do recorte) caso a atividade já tenha sido destruída. No método onDestroy, o ProgressDialog de carregamento é fechado se estiver visível, prevenindo vazamentos de contexto. Essas precauções impedem exceções relacionadas ao estado da aplicação durante mudanças de orientação ou saídas inesperadas.

As Figuras 5 ilustram as telas do aplicativo durante as fases de seleção da imagem, definição da ROI e exibição dos resultados do reconhecimento de texto. Cada etapa foi implementada para oferecer uma experiência interativa e robusta ao usuário, conforme detalhado acima.

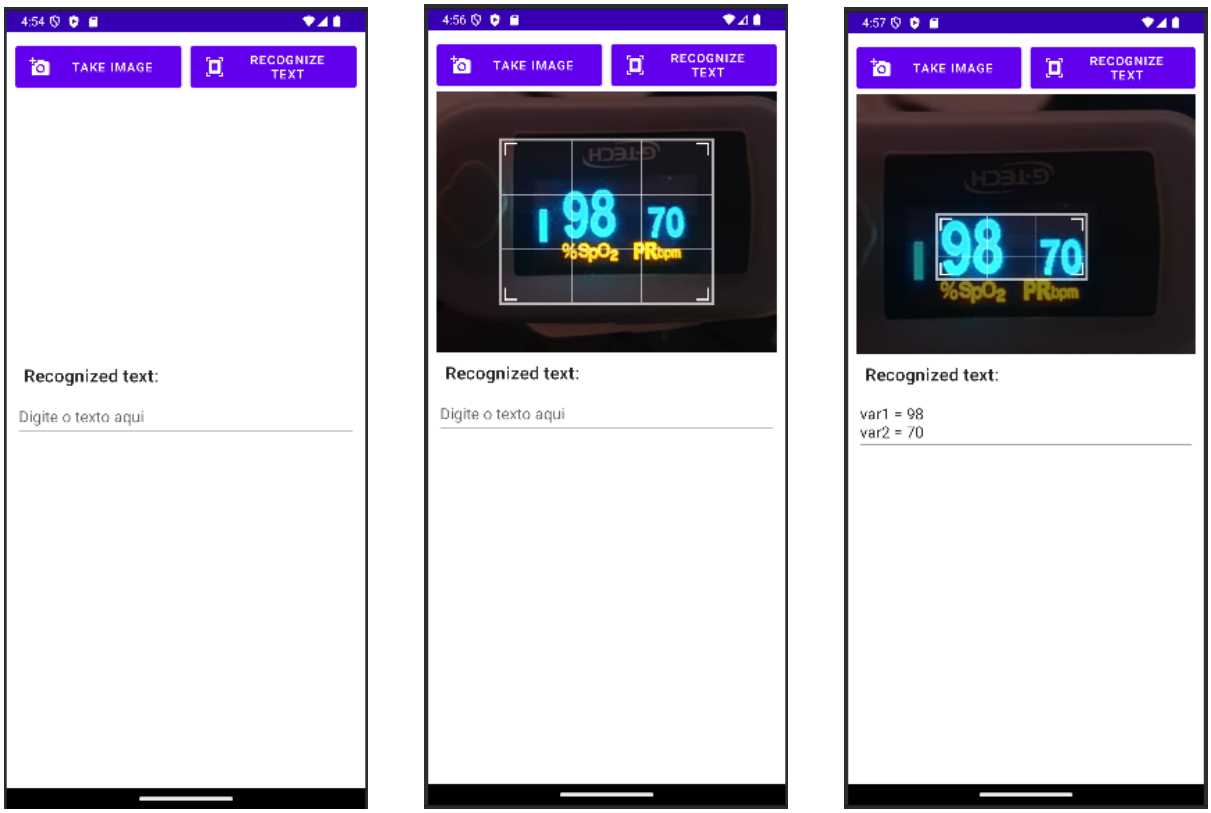

Figura 5 – Telas do aplicativo: (a) tela inicial, (b) definição da região de interesse (ROI) e (c) imagem processada com os valores medidos

Além desses cuidados relacionados ao ciclo de vida da atividade, o processo completo de desenvolvimento incluiu também as etapas de compilação, simulação e implantação do aplicativo no dispositivo móvel.

O processo de compilação foi realizado utilizando o *Android Studio*, ambiente oficial de desenvolvimento para Android. A ferramenta utiliza o sistema de *build Gradle*, que coordena as etapas de transformação do código-fonte em um pacote instalável (`.apk` ou `.aab`). Durante a compilação, ocorrem as seguintes etapas:

1. **Análise e síntese do código Kotlin**
   O compilador Kotlin transforma o código-fonte em *bytecode* compatível com a JVM, realizando otimizações como *smart casting*, remoção de código morto e inferência automática de tipos. Esse *bytecode* é posteriormente convertido para formato DEX pela ferramenta D8, etapa necessária para execução em dispositivos Android.

2. **Processamento de recursos (AAPT2 - *Android Asset Packaging Tool 2)***
   O *Android Asset Packaging Tool 2* (AAPT2) é a ferramenta do sistema de *build* do Android responsável por processar os recursos do aplicativo. Durante essa etapa, arquivos XML de layout, ícones, imagens, textos e declarações de tema são analisados, compilados, comprimidos e indexados. O AAPT2 também gera a classe R,

utilizada para o acesso programático aos recursos internos do aplicativo durante o desenvolvimento.

3. **Linkagem e minificação (R8)**
   O R8 é a ferramenta oficial do Android para linkagem, minificação e otimização de código durante o processo de *build*. Seu objetivo é reduzir o tamanho final do aplicativo e melhorar o desempenho. Para isso, o R8 executa as etapas de *shrink* (remoção de classes, métodos e campos não utilizados), *obfuscation* (renomeação de símbolos para dificultar engenharia reversa) e *optimization* (aplicação de otimizações em nível de bytecode). Essa etapa mostra-se particularmente relevante neste trabalho devido ao uso de bibliotecas externas, como o ML Kit e o TensorFlow Lite, que adicionam um volume significativo de código ao projeto.

4. **Empacotamento e assinatura do APK**
   O Android Studio gera o pacote instalável contendo:

   - código DEX;
   - bibliotecas nativas pré-compiladas (incluindo as do TensorFlow Lite);
   - recursos comprimidos;
   - metadados do *Android Manifest*.

   Após o empacotamento, o APK é assinado automaticamente com uma *debug key* durante a fase de desenvolvimento.

Esse *pipeline* assegura que o aplicativo esteja otimizado para execução local, principalmente considerando as operações de visão computacional empregadas no sistema.

Antes da implantação no dispositivo físico, o aplicativo foi submetido a testes funcionais no *Android Emulator*, permitindo validar:

- fluxo de navegação entre telas;
- permissões de câmera e armazenamento;
- funcionamento do recorte de imagem (*CropImageView*);
- execução do ML Kit *On-Device* OCR;
- integração do modelo TensorFlow Lite embarcado.

O emulador foi configurado com especificações semelhantes às de um *smartphone* intermediário, porém apresentou limitações importantes:

1. **Simulação limitada da câmera**
   O simulador utiliza imagens estáticas ou fluxo simulado, impossibilitando testar a captura real de displays de dispositivos médicos.

2. **Variações de latência irrealistas**
   O desempenho em CPU virtualizada não representa com precisão a velocidade de inferência do modelo TensorFlow Lite em hardware ARM real.

3. **Aceleradores ausentes**
   A simulação não oferece suporte a NNAPI (*Neural Networks API*), GPU móvel ou DSP, recursos que influenciam diretamente o desempenho de modelos de IA em dispositivos reais.

Por essas razões, a simulação serviu principalmente para validar o fluxo de interface e depurar exceções, enquanto a avaliação de desempenho foi realizada exclusivamente em hardware físico.

Para validar adequadamente o aplicativo, o APK (*Android Package*) foi carregado e executado em um *smartphone* Samsung Galaxy A15. O processo envolveu:

1. **Ativação do modo desenvolvedor**
   Para permitir a comunicação do dispositivo com o ambiente de desenvolvimento, foram realizados:

   - sete toques sobre o número da versão do Android;
   - habilitação da opção *Depuração USB*.

2. **Conexão ao Android Studio via ADB (Android Debug Bridge)**
   O ADB estabeleceu um canal de comunicação entre o computador e o smartphone, permitindo:

   - instalação automática do APK;
   - acesso aos logs em tempo real por meio do *Logcat*;
   - captura de exceções durante o recorte, pré-processamento e inferência.

3. **Implantação do modelo TensorFlow Lite no dispositivo**
   O arquivo `.tflite` foi incluído na pasta `assets` do aplicativo. Durante a execução:

   - o *TensorFlow Lite Interpreter* carrega o modelo em memória;
   - o pipeline de pré-processamento é aplicado ao *bitmap* recortado;
   - a inferência é executada diretamente no processador ARM do dispositivo.

4. **Testes com imagens reais capturadas no aparelho**
   Foram avaliados:

   - desempenho do ML Kit em diferentes condições de iluminação;
   - acionamento do *fallback* para o modelo TFLite quando o OCR falha;
   - tempos de inferência e estabilidade da aplicação;
   - comportamento diante de variações de orientação de aquisição, reflexos e ruído.

A execução no smartphone Samsung Galaxy A15 e sistema operacional Android 14 permitiu medir com precisão a responsividade do sistema, validando a abordagem de processamento totalmente local e confirmando que a solução atende aos requisitos de privacidade, baixa latência e robustez.

5.2 ML KIT Text Recognition

O ML Kit é uma biblioteca de aprendizado de máquina desenvolvida pelo Google com foco em dispositivos móveis, oferecendo APIs pré-treinadas para visão computacional e processamento de linguagem natural. Projetado para facilitar a integração em aplicativos Android e iOS, o ML Kit permite a utilização de modelos otimizados sem exigir que o desenvolvedor construa ou treine redes neurais a partir de uma inicialização arbitrária. Embora seja de código fechado, sua base técnica é construída sobre o TensorFlow Lite (TFLite) (GOOGLE, 2025b) e a Android Neural Networks API (NNAPI), tecnologias que garantem execução eficiente em smartphones com recursos limitados de hardware. Segundo GOOGLE (2025a), o objetivo do ML Kit é democratizar o acesso a modelos de aprendizado de máquina em aplicações móveis.

No caso do reconhecimento óptico de caracteres (OCR), o processo geralmente é estruturado em duas etapas principais (KUMAR; PATEL; SINGH, 2024). A primeira é a detecção de regiões de texto, na qual redes neurais convolucionais (CNNs – Convolutional Neural Networks) identificam áreas da imagem que contêm caracteres (LECUN; BENGIO, 1998). A segunda etapa é o reconhecimento de caracteres, no qual os segmentos detectados são processados por modelos especializados em sequências, como as redes neurais recorrentes do tipo LSTM (Long Short-Term Memory) (HOCHREITER; SCHMIDHUBER, 1997).

As LSTMs são uma variante das redes recorrentes (RNNs) projetadas para superar a dificuldade de aprendizado de dependências de longo prazo em sequências de dados. Sua arquitetura inclui portas (*gates*) que regulam o fluxo de informação, permitindo "lembrar" ou "esquecer" estados anteriores conforme necessário. No contexto do OCR, isso é fundamental, pois possibilita reconstruir corretamente sequências de caracteres preservando a ordem, mesmo em casos de ruídos visuais, rotações ou distorções na imagem.

O resultado do reconhecimento é estruturado em diferentes níveis hierárquicos: *Block* (bloco de texto), *Line* (linha de texto), *Element* (palavra) e *Symbol* (caractere individual). Essa organização permite maior flexibilidade, possibilitando desde a simples extração de números até a análise da disposição espacial do texto em uma cena.

O ML Kit disponibiliza dois modos de operação para OCR: *On-Device* e *Cloud-Based*. No modo *On-Device*, todo o processamento ocorre no próprio dispositivo móvel, sem necessidade de conexão à internet. Essa abordagem garante baixa latência, maior privacidade dos dados e é adequada para cenários que envolvem informações sensíveis, como os dados médicos utilizados neste trabalho. Já no modo *Cloud-Based*, o processamento ocorre nos servidores do Google (GOOGLE, 2025a), o que amplia o suporte a idiomas e melhora a precisão em textos complexos, manuscritos ou com fontes incomuns. Entretanto, essa modalidade depende de conexão com a internet, e assim pode introduzir maior tempo de resposta e expõe potenciais riscos à confidencialidade.

Internamente, os modelos utilizados pelo ML Kit passam por otimizações como quantização e *pruning*, tornando-os mais leves e rápidos. Estudos como o de Han, Mao e Dally (2016) demonstram que essas técnicas podem reduzir significativamente o tamanho dos modelos, mantendo sua precisão, o que reforça a viabilidade de uso em dispositivos móveis.

Apesar de suas vantagens, o ML Kit apresenta limitações, sobretudo por não permitir ajustes ou re-treinamento dos modelos utilizados. Em aplicações que envolvem caracteres em displays de sete segmentos, comuns em dispositivos médicos, o desempenho do ML Kit pode ser limitado. Para superar essa restrição, optou-se neste trabalho por complementar o sistema com um modelo personalizado treinado em TensorFlow Lite (GOOGLE, 2025b), capaz de lidar com esse tipo específico de representação numérica.

Outra questão importante é a diferença entre o ML Kit e soluções de OCR open source. O ML Kit se destaca pela praticidade, pela integração simples com aplicativos móveis e pela otimização para rodar em smartphones. Em contrapartida, modelos open source, como o Tesseract OCR (SMITH, 2007), o MMOCR (WAN et al., 2021), o TrOCR (LI et al., 2021b), e *frameworks* voltados a manuscritos como Calamari (WICK; REUL; PUPPE, 2018) e Kraken (KIESSLING, 2019), oferecem maior flexibilidade, pois podem ser re-treinados e adaptados para cenários específicos, ainda que com custo maior de processamento ou integração mais complexa. Modelos recentes baseados em Transformers, como o TrOCR, apresentam alta precisão, mas são mais custosos computacionalmente e voltados principalmente para uso em servidores ou desktops.

Tabela 4 – Comparação entre ML Kit e soluções de OCR open source

Solução	**Tipo**	**Treinamento Customizado**	**Tamanho/ Latência**	**Idiomas Suportados**
ML Kit	Proprietário (Google)	Não	Baixa latência, otimização de memória e processamento para execução *on-device* (dispositivo móvel)	100+ (limitados *on-device*)
Tesseract	Open Source	Sim	Médio, dependente da versão	100+
MMOCR/ TrOCR	Open Source	Sim (avançado)	Alto custo computacional (desktop/servidor)	Multilíngue
Calamari/ Kraken	Open Source	Sim	Médio	Focado em manuscritos

Fonte: Elaborado pelo autor.

No presente trabalho, optou-se pelo uso do ML Kit em modo *On-Device* por garantir privacidade, simplicidade de integração e desempenho adequado. Contudo, em situações de falha na leitura de caracteres, especialmente em displays de sete segmentos um modelo alternativo baseado em TensorFlow Lite foi integrado como solução de *fallback*, assegurando maior robustez ao sistema. Dessa forma, a aplicação combina a praticidade do ML Kit com a flexibilidade de um modelo customizado, alcançando melhores resultados no contexto de telemonitoramento médico.

5.3 TensorFlow

O TensorFlow é uma biblioteca open-source desenvolvida pelo Google para a criação, treinamento e implantação de modelos de machine learning (ML) e deep learning (DL). Sua arquitetura flexível e otimizada permite aplicações que vão desde pesquisas acadêmicas até soluções comerciais em larga escala. Por essa razão, é amplamente adotado por pesquisadores, engenheiros e desenvolvedores para resolver problemas complexos em áreas como visão computacional, processamento de linguagem natural, sistemas de recomendação e controle robótico (TENSORFLOW AUTHORS, 2024).

Uma das principais características do TensorFlow é a flexibilidade e escalabilidade. Ele permite a construção de modelos simples, como regressões lineares, até arquiteturas complexas de redes neurais profundas com milhões de parâmetros. Além disso, oferece

suporte para execução em diferentes tipos de hardware, como CPUs, GPUs e TPUs (*Tensor Processing Units*), otimizando o desempenho e reduzindo o tempo de treinamento, especialmente em grandes conjuntos de dados (ABADI et al., 2016).

O funcionamento do TensorFlow baseia-se em grafos computacionais nos quais os nós representam operações matemáticas (como multiplicações de matrizes, funções de ativação e convoluções) e as arestas representam o fluxo de dados entre essas operações. Os dados manipulados pelo TensorFlow são armazenados em tensores, estruturas multidimensionais semelhantes a arrays do NumPy, mas com suporte à execução otimizada em diferentes dispositivos. Esses tensores podem representar desde valores escalares até grandes matrizes de alta dimensão.

Durante o treinamento de um modelo, o TensorFlow realiza um ciclo contínuo que começa com a propagação direta (*forward pass*), na qual os dados de entrada percorrem as diferentes camadas da rede neural até gerar uma saída prevista. Em seguida, é calculada a função de perda (*loss function*), que quantifica a discrepância entre a saída prevista e a saída real esperada.

A escolha da função de perda depende do tipo de problema. Em tarefas de regressão, utiliza-se com frequência o Erro Quadrático Médio (*Mean Squared Error* – MSE), que mede o desvio médio quadrático entre as previsões ($\hat{y}_i$) e os valores reais (y_i):

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Essa função penaliza de forma mais intensa erros maiores.

Já em tarefas de classificação, especialmente em redes neurais com saídas probabilísticas (como *softmax*), é comum empregar a Entropia Cruzada (*Cross-Entropy Loss*), que mede a distância entre a distribuição de probabilidade prevista pelo modelo ($\hat{y}_i$) e a distribuição real das classes (y_i):

$$\text{Cross-Entropy} = -\frac{1}{n}\sum_{i=1}^{n} y_i \log(\hat{y}_i)$$

Com base nesse erro, inicia-se a retropropagação (*backpropagation*), processo no qual o erro é propagado de volta pelo grafo da rede neural e os passos de adaptação do modelo são determinados de acordo com o algoritmo de otimização dos pesos, como Gradiente Descedente, Adam ou RMSProp. Esses passos de adaptação servem para ajustar os pesos da rede, minimizando a função de perda a cada iteração, até que se alcance o desempenho desejado. Esse fluxo, ilustrado na Figura 6, evidencia como a estrutura baseada em grafos do TensorFlow, aliada ao suporte para processamento paralelo e distribuído, proporciona eficiência tanto em treinamentos complexos realizados em *clusters* de alto desempenho quanto na execução de modelos leves em dispositivos móveis.

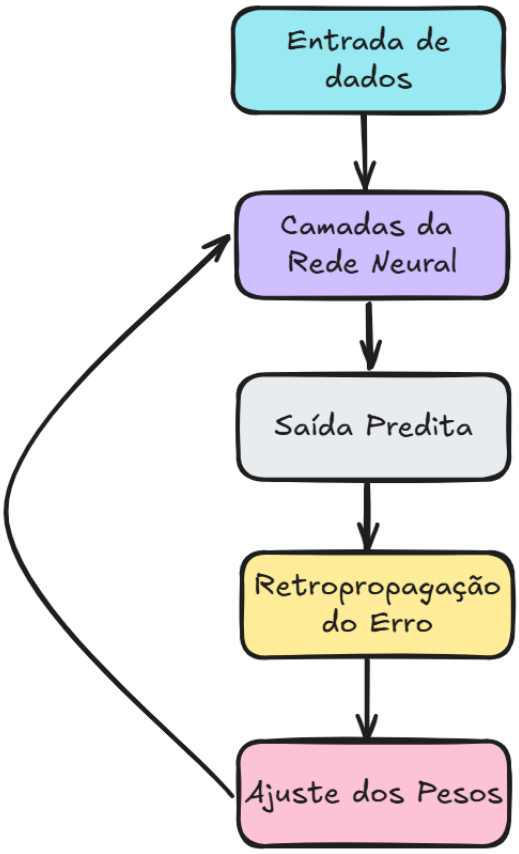

Figura 6 – Fluxo básico de treinamento em um modelo TensorFlow

No desenvolvimento deste trabalho, foi utilizado o TensorFlow Lite (TFLite), uma versão otimizada do TensorFlow voltada para execução de modelos em dispositivos móveis e sistemas embarcados. O TFLite permite o uso de diversas técnicas de otimização para reduzir o tamanho do modelo e melhorar a velocidade de execução, como a quantização, que converte pesos e ativações de 32 bits (*float32*) para formatos mais compactos, como *int8* ou *float16*; o *pruning* e *clustering*, que eliminam pesos desnecessários e agrupam valores semelhantes para melhorar a compressão; e a conversão de operadores, que substitui operações complexas por implementações mais eficientes para hardware móvel (TEAM, 2022).

A arquitetura do TensorFlow Lite é composta por um conversor, responsável por transformar um modelo treinado no TensorFlow convencional (nos formatos `.pb` ou `SavedModel`) em um arquivo otimizado no formato `.tflite`, e por um intérprete, que executa esse modelo diretamente no dispositivo, podendo explorar aceleração por GPU, DSP ou pela *API Android Neural Networks* (NNAPI), como pode ser observado na Figura 7. Essa abordagem foi adotada no aplicativo Android desenvolvido neste trabalho devido à necessidade de processamento rápido e totalmente offline, eliminando a dependência de conexões com servidores externos. Essa característica garante respostas imediatas às solicitações e contribui para a preservação da privacidade dos dados do usuário.

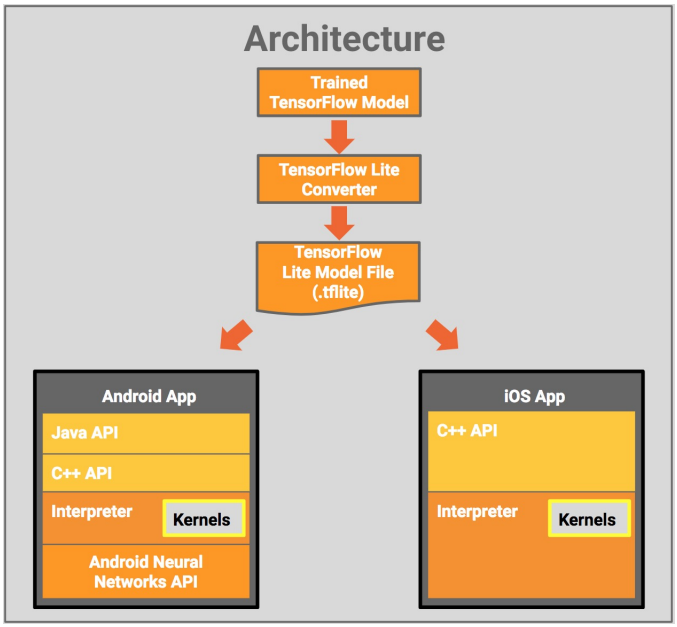

Figura 7 – Arquitetura do TensorFlow Lite

Fonte: (MAUNG; AYE, 2019)

O TensorFlow fornece suporte nativo à API Keras, que simplifica a criação e treinamento de modelos de DL por meio de uma interface de alto nível. O Keras permite prototipagem rápida com poucas linhas de código, mantendo a possibilidade de ajustes avançados utilizando os recursos de baixo nível do TensorFlow. Essa compatibilidade contribuiu para acelerar a fase de desenvolvimento e testes do modelo antes de sua conversão para TensorFlow Lite.

Embora o ML Kit seja uma ferramenta prática para reconhecimento de texto, ele apresenta limitações ao lidar com caracteres de sete segmentos, comuns em *displays* de dispositivos eletrônicos. Por essa razão, optou-se por desenvolver e treinar um modelo personalizado no TensorFlow, posteriormente convertido para TFLite, garantindo leitura precisa e confiável desses caracteres no aplicativo Android.

5.4 Treinamento e Pré-processamento do Modelo

Para o treinamento do modelo, foi utilizado o conjunto de dados *Digits Segment Display* (Conjunto de Dígitos de Segmento), disponibilizado na plataforma Kaggle (KAGGLECOMMUNITY, 2024). Esse conjunto contém imagens representativas de dígitos exibidos em displays de sete segmentos, organizadas de forma a facilitar a divisão entre conjunto de treinamento e conjunto de validação. Especificamente, o conjunto de treinamento é composto por 1.000 imagens para cada dígito (0–9), enquanto o conjunto de validação contém 400 imagens por classe, garantindo balanceamento inicial entre as categorias.

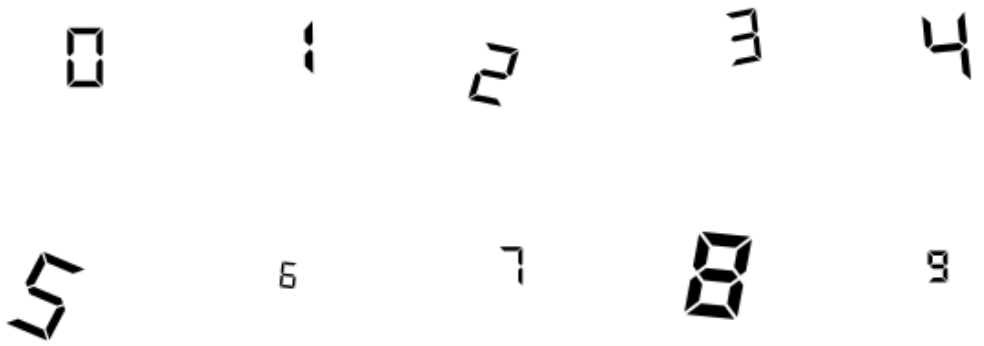

Figura 8 – Exemplos de imagens do dataset Digits Segment Display para os dígitos de 0 a 9.

O desenvolvimento do modelo teve como principal referência o trabalho de (FINNEGAN et al., 2019), intitulado *Automated method for detecting and reading seven-segment digits from images of blood glucose metres and blood pressure monitors* (Método automatizado para detectar e ler dígitos de sete segmentos a partir de imagens de medidores de glicose e monitores de pressão arterial). Esse estudo fornece uma abordagem robusta para a detecção e leitura automatizada de caracteres em visores de sete segmentos, sendo amplamente aplicável a cenários com condições de captura variáveis, como iluminação irregular, ruídos e distorções ópticas. Além de orientar o desenho da arquitetura da rede, o artigo serviu de base para o *pipeline* (fluxo de processamento) de pré-processamento, a escolha dos parâmetros e as estratégias de aumento de dados (*data augmentation*).

Além dessas contribuições gerais, o trabalho de (FINNEGAN et al., 2019) apresenta um conjunto de técnicas específicas que influenciaram diretamente a concepção do modelo desenvolvido neste estudo. Os autores estruturam um *pipeline* de pré-processamento voltado à padronização das imagens de displays de sete segmentos, envolvendo etapas como conversão de espaço de cor, aplicação de filtros para redução de ruído e uso de limiarização para destacar os segmentos ativos. O uso combinado de *median blur* e *Gaussian blur* mostrou-se eficaz para suavizar artefatos sem comprometer a estrutura dos segmentos, enquanto a conversão para o espaço HSV facilita o isolamento das regiões de interesse sob diferentes condições de iluminação. Esse conjunto de operações que antecede a fase de classificação forneceu uma base sólida para o desenho do *pipeline* empregado neste trabalho, sobretudo no que diz respeito à normalização das imagens e ao controle de variações provocadas por iluminação irregular.

Outro ponto relevante do estudo é a estratégia proposta para segmentar e identificar cada dígito no visor. O método utiliza uma abordagem morfológica para localizar agrupamentos de segmentos e extrair as regiões correspondentes a cada número. Essa técnica mostra-se particularmente eficiente em dispositivos médicos que apresentam múltiplos dígitos alinhados, garantindo que cada região seja tratada individualmente antes

da classificação. Embora o presente trabalho utilize uma arquitetura baseada em redes neurais convolucionais (CNN), em vez de regras morfológicas definidas manualmente, os princípios de segmentação estabelecidos por (FINNEGAN et al., 2019), especialmente a importância do isolamento e recorte preciso da ROI influenciaram diretamente o fluxo adotado para separar os dígitos a serem reconhecidos.

Os autores também enfatizam a importância da robustez do modelo frente à variação de orientação da aquisição, reflexos, ruído e qualidade do sensor da câmera, propondo diversas transformações para aumentar a diversidade do conjunto de treinamento. As técnicas de *data augmentation* descritas por (FINNEGAN et al., 2019), como variações de rotação, brilho, contraste e distorções geométricas serviram como referência para a composição das estratégias utilizadas neste trabalho, contribuindo para a capacidade do modelo de generalizar em condições reais de captura.

Por fim, embora o método original de (FINNEGAN et al., 2019), utilize um classificador baseado em características geométricas extraídas manualmente dos segmentos, o presente estudo aproxima-se de sua filosofia central: construir um *pipeline* capaz de lidar com imagens ruidosas e não padronizadas, mantendo precisão elevada mesmo em condições adversas. Assim, o trabalho representou não apenas uma referência conceitual, mas também um guia prático para diversas decisões de implementação adotadas durante o desenvolvimento do modelo.

A arquitetura da rede neural convolucional (*Convolutional Neural Network* - CNN) utilizada neste trabalho é composta por três camadas convolucionais, cada uma seguida por camadas de *max pooling* (amostragem máxima), que reduzem a dimensionalidade espacial e preservam as características mais relevantes para a classificação. Após as camadas convolucionais e de *pooling*, a rede conta com camadas densas (*fully connected* — totalmente conectadas), responsáveis por realizar a classificação final dos dígitos. Para mitigar o *overfitting* (sobreajuste), foram incorporadas técnicas de *dropout* e regularização L2. O treinamento empregou o otimizador Adam e a função de perda *categorical crossentropy* (entropia cruzada categórica), apropriada para problemas de classificação multiclasse.

A Figura 9 ilustra de forma esquemática a arquitetura da rede convolucional aplicada ao reconhecimento de dígitos, destacando a sequência de camadas convolucionais, pooling e densas.

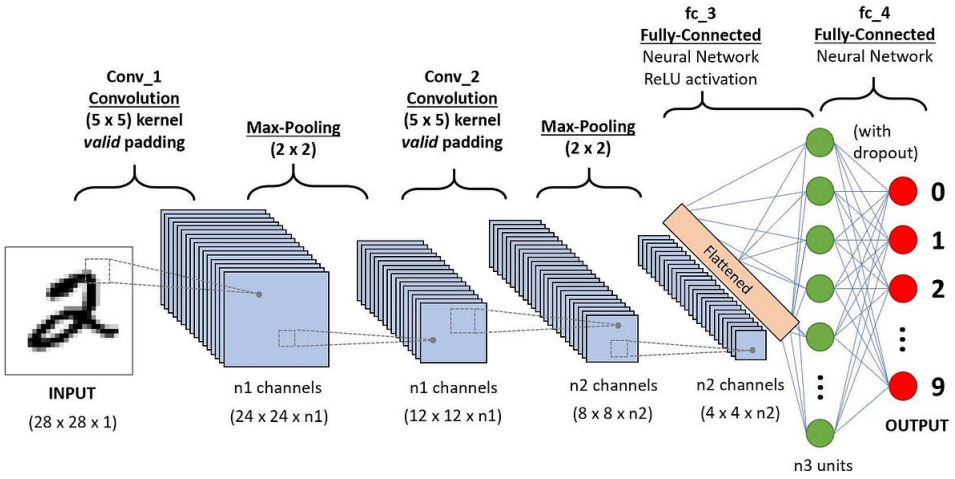

Figura 9 – Arquitetura da rede neural convolucional

Fonte: (JAIN, 2025)

Para aumentar a robustez do modelo em condições reais, foi aplicada a técnica de *data augmentation*, que introduz variações artificiais nas imagens de treinamento, incluindo rotações, translações e transformações de escala. Essa estratégia permite que a rede aprenda a reconhecer os dígitos mesmo diante de distorções ou variações naturais durante a captura por dispositivos móveis. Adicionalmente, foi utilizado um *callback* (função de retorno) de *Early Stopping* (parada antecipada) para interromper o treinamento caso a perda de validação não apresentasse melhoria após três épocas consecutivas. Ajustes de pesos de classes (*class weights*) também foram realizados, de forma a compensar eventuais desequilíbrios na distribuição de amostras.

Seguindo as diretrizes de (FINNEGAN et al., 2019), foi implementado um *pipeline* de pré-processamento antes da extração de características, composto por:

1. **Conversão para tons de cinza e redimensionamento** As imagens originais são convertidas para escala de cinza, reduzindo a dimensionalidade do problema e eliminando informações de cor irrelevantes. Em seguida, são redimensionadas para 56×56 pixels, padronizando a entrada da rede e facilitando a extração consistente de características.

2. **Binarização com o método de Sauvola** A segmentação adaptativa é realizada por meio do algoritmo de Sauvola (LAZZARA; GéRAUD, 2014), utilizando uma janela de 25 pixels e parâmetro $k = 0.34$. Essa técnica é mais eficiente que métodos globais, pois se adapta a variações locais de iluminação, comuns em fotos capturadas por usuários. Caso a binarização adaptativa falhe, aplica-se automaticamente o método de Otsu (OTSU, 1979) como *fallback* (procedimento de contingência), garantindo continuidade no processamento.

3. **Preparação para o descritor HOG** Embora no artigo original de (FINNEGAN et al., 2019) o *Histogram of Oriented Gradients* (HOG — Histograma de Gradientes Orientados) seja extraído sobre imagens binarizadas, neste trabalho optou-se por aplicá-lo diretamente sobre as imagens em tons de cinza, mantendo gradientes de intensidade mais ricos para a CNN. O descritor HOG, com os parâmetros ajustados conforme o estudo original, extrai 1.296 características de cada imagem, capturando padrões estruturais essenciais para distinguir os dígitos.

Essas características extraídas pelo HOG são então utilizadas como entrada para a rede convolucional, que aprende a mapear padrões visuais específicos para as classes correspondentes (0–9). Essa combinação de pré-processamento tradicional e aprendizado profundo permitiu um desempenho elevado mesmo em condições adversas de captura, como iluminação irregular, presença de ruídos visuais, distorções ópticas causadas por ângulo de inclinação ou foco impreciso, variações de contraste entre os segmentos acesos e o fundo, e oclusões parciais dos dígitos no display.

A Figura 10 apresenta um exemplo prático do processo de pré-processamento e classificação. À esquerda, observa-se a imagem do dígito redimensionada, enquanto à direita está a versão binarizada para extração de descritores HOG. A parte inferior exibe a saída do modelo, indicando a classe mais provável (dígito “2”) e a distribuição de probabilidades entre todas as classes. Esse resultado evidencia a robustez do modelo em identificar corretamente os caracteres de sete segmentos mesmo após transformações de pré-processamento.

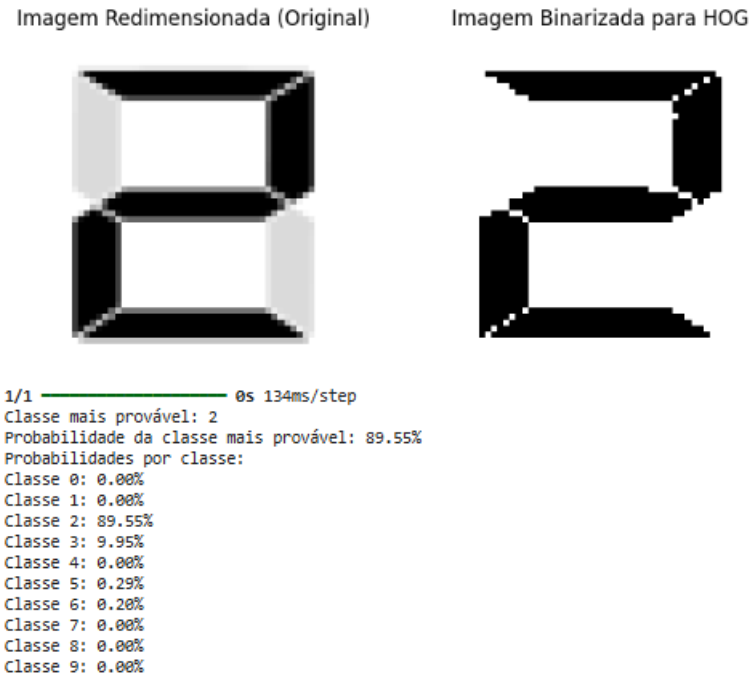

Figura 10 – Pré-processamento e classificação de um dígito em display de sete segmentos

Durante o desenvolvimento do sistema, observou-se que a etapa de pré-processamento é particularmente sensível às variações presentes nas imagens capturadas pelos usuários. Como destacado por (FINNEGAN et al., 2019), dispositivos domésticos como glicosímetros, oxímetros e medidores de pressão arterial apresentam displays sujeitos a condições não controladas de captura, incluindo variações de iluminação, reflexos, ruídos, ângulo da câmera, distância e contraste entre o dígito e o fundo. Essa heterogeneidade afeta diretamente a eficiência das técnicas de binarização e segmentação aplicadas neste trabalho.

No caso específico dos displays de sete segmentos, pequenas mudanças no ambiente podem alterar a intensidade e a espessura aparente dos segmentos acesos. Conforme observado no estudo, tais variações podem fazer com que algumas partes dos segmentos desapareçam durante a binarização, ou que regiões não pertencentes ao dígito sejam erroneamente identificadas como iluminadas. Isso compromete a etapa de extração de características, uma vez que o classificador passa a receber representações distorcidas ou incompletas do dígito original.

Além disso, erros no pré-processamento tendem a se propagar para as etapas seguintes. Quando o contraste é insuficiente, quando há trepidação na captura ou quando reflexos encobrem parte do display, a imagem resultante pode apresentar falhas de segmentação, ruído residual ou perda de partes relevantes do dígito. Nesses casos, o modelo de classificação torna-se mais propenso a equívocos, especialmente em dígitos visualmente semelhantes (como 5 e 6, ou 3 e 8), reforçando a necessidade de robustez nas etapas de pré-processamento.

Assim, em concordância com os resultados apresentados por (FINNEGAN et al., 2019), este trabalho evidencia que a eficácia da classificação depende fortemente da qualidade do pré-processamento. A normalização apropriada, a escolha correta do método de binarização e o tratamento dos artefatos visuais tornam-se essenciais para mitigar os impactos das condições variáveis de captura, comuns no uso doméstico de dispositivos médicos.

Tendo em vista essa sensibilidade do modelo às etapas iniciais, buscou-se garantir que o mesmo *pipeline* de pré-processamento fosse reproduzido de forma consistente durante a inferência em ambiente móvel. Para isso, o modelo final foi convertido para TensorFlow Lite (TFLite) e integrado ao aplicativo Android desenvolvido em Kotlin. As etapas de pré- processamento implementadas em Python, incluindo conversão para escala de cinza, redimensionamento e aplicação de binarização adaptativa ou global, foram replicadas no aplicativo, assegurando que as previsões realizadas no dispositivo refletissem com precisão o comportamento do modelo treinado.

As Figuras 11 apresentam exemplos da interface do aplicativo exibindo dígitos corretamente reconhecidos pelo modelo embarcado, demonstrando a eficácia da integração e a aplicabilidade prática da solução proposta para leitura de displays de sete segmentos

em dispositivos móveis. A Figura (c), especificamente, mostra a imagem original, a imagem após o pré-processamento, o dígito classificado pelo modelo e a respectiva estimativa confiança da previsão aproximada pela saída da função *softmax*.

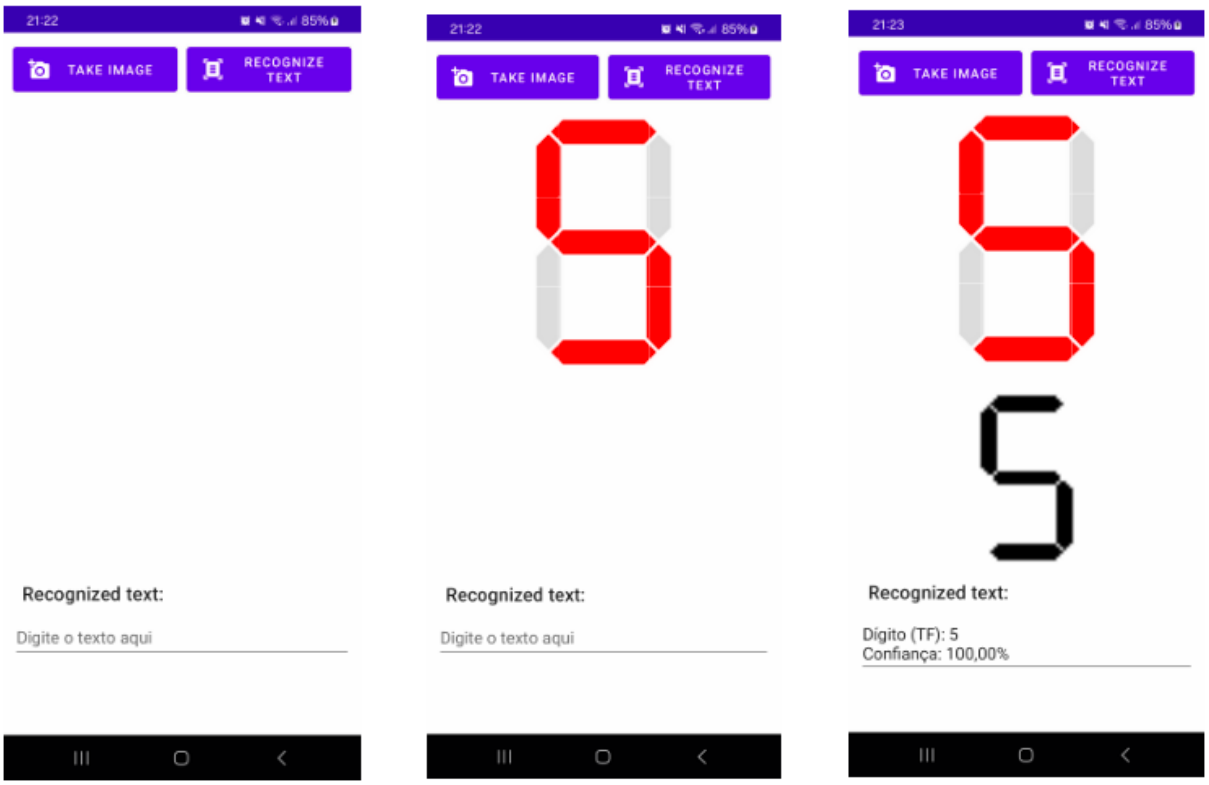

Figura 11 – Etapas do processamento no aplicativo: (a) captura da imagem, (b) definição da região de interesse (ROI) e (c) imagem pré-processada com os valores extraídos

5.5 Processamento Local vs Processamento na Nuvem

Ao desenvolver uma solução baseada em visão computacional para leitura automática de dígitos, uma decisão arquitetural importante envolve a escolha entre realizar o processamento localmente no dispositivo móvel ou de forma remota em um servidor na nuvem. Essa decisão afeta diretamente latência, privacidade, disponibilidade e experiência do usuário, aspectos amplamente discutidos em aplicações móveis (DENG, 2019).

O processamento local, empregado neste trabalho tanto com o TensorFlow Lite (TFLite) quanto com o ML Kit, apresenta como principais vantagens a baixa latência e a maior privacidade. A inferência ocorre diretamente no *smartphone*, sem a necessidade de enviar imagens a servidores externos, o que elimina atrasos de rede e reduz riscos associados ao tráfego de dados sensíveis (GOOGLE, 2024). Além disso, essa abordagem permite funcionamento mesmo em ambientes com conectividade limitada, realidade comum no uso doméstico de dispositivos médicos no Brasil. Em contrapartida, a execução local

depende da capacidade computacional do dispositivo, o que pode limitar o uso de modelos mais complexos (DENG, 2019).

O processamento na nuvem possibilita o uso de modelos maiores e mais robustos, aproveitando infraestrutura escalável baseada em GPUs ou TPUs e facilitando atualizações centralizadas (KRISHNAMOORTHY; KONIDENA, 2024). Entretanto, essa arquitetura depende fortemente de conexão estável e introduz desafios adicionais de privacidade e conformidade com legislações como a LGPD, já que as imagens precisam ser enviadas a servidores externos (HAWRYLISZYN; COELHO; BARJA, 2021).

Diante desse cenário, a adoção do processamento totalmente local neste projeto, contemplando ML Kit, TFLite e todo o *pipeline* de pré-processamento, foi guiada pela necessidade de reduzir latência, garantir privacidade e assegurar operação mesmo em condições de baixa conectividade. Ainda assim, reconhece-se que o processamento em nuvem permanece uma alternativa adequada em cenários que demandam arquiteturas de maior complexidade e conectividade confiável.

A Tabela 5 apresenta um resumo comparativo entre as duas abordagens discutidas.

Tabela 5 – Comparativo entre processamento local e processamento na nuvem

Critério	**Processamento Local (On-device)**	**Processamento na Nuvem**
Latência	Inferência muito rápida, executada diretamente no dispositivo.	Maior latência devido ao envio das imagens e ao tempo de resposta do servidor.
Privacidade e segurança	Alta privacidade, pois os dados não deixam o dispositivo.	Riscos aumentados, já que dados sensíveis são transmitidos e processados externamente.
Dependência de Internet	Funciona mesmo offline ou com conexão limitada.	Requer conexão estável para envio e processamento das imagens.
Capacidade computacional	Limitado pela CPU/GPU/NPU do smartphone; modelos muito grandes podem não executar.	Permite uso de GPUs/TPUs e modelos mais robustos e complexos.
Atualizações de modelo	Exigem atualização do aplicativo para distribuir novos modelos.	Atualizações são centralizadas e aplicadas diretamente no servidor.
Consumo de bateria	Maior uso do hardware local, aumentando o consumo energético.	Menor uso do hardware do usuário, reduzindo consumo de bateria.

Fonte: Elaborado pelo autor.

6 Resultados

6.1 ML KIT Text Recognition

Para a etapa de validação do aplicativo desenvolvido e da biblioteca ML Kit, foi utilizado como dispositivo de medição um oxímetro digital da marca G-TECH, modelo OLED Graph. Esse equipamento foi selecionado por ser amplamente empregado em medições clínicas e domiciliares de frequência cardíaca (FC) e saturação periférica de oxigênio (SpO_2). As principais características técnicas do dispositivo estão apresentadas na Tabela 6, que detalha parâmetros como faixa de medição, precisão, resolução, condições de operação e consumo de energia, fornecendo uma visão clara das capacidades e limitações do equipamento utilizado.

Tabela 6 – Especificações técnicas do oxímetro G-Tech OLED Graph

Parâmetro	**Especificação**
Tipo de tela	OLED
Faixa de medição SpO_2	70–99%
Variação exibida SpO_2	0–99%
Precisão SpO_2	70–99%: ±3%; abaixo de 70%: não definido
Resolução SpO_2	1%
Faixa de medição FC	30–235 bpm
Variação exibida FC	0–254 bpm
Precisão FC	30–99 bpm: ±2 bpm; 100–235 bpm: ±2%
Resolução FC	1 bpm
Fonte de energia	2 pilhas alcalinas AAA de 1,5V
Consumo de energia	< 30 mA
Autonomia das pilhas	até 30 horas de uso contínuo
Condições de operação	5–40°C; 15–93% UR; 70–106 kPa
Condições de armazenamento	-25–70°C; $\leq$ 93% UR; 70–106 kPa

Fonte: Elaborado pelo autor.

Além do oxímetro, o processo de coleta de dados foi realizado por meio de um smartphone Samsung Galaxy A15, escolhido por sua disponibilidade no contexto experimental e por apresentar desempenho adequado para execução das tarefas de captura e processamento de imagens. As principais especificações técnicas do dispositivo estão reunidas na Tabela 7, permitindo uma visão clara de sua capacidade de hardware.

Tabela 7 – Especificações técnicas do smartphone Samsung Galaxy A15

Parâmetro	**Especificação**
Tela	Super AMOLED 6,5” FHD+ (1080 x 2340)
Processador	MediaTek Helio G99 (octa-core, até 2,2 GHz)
Memória RAM	8 GB
Armazenamento interno	128 GB (expansível via microSD até 1 TB)
Sistema operacional	Android 14 (One UI 6)
Câmera traseira	Tripla: 50 MP (principal), 5 MP (ultrawide), 2 MP (macro)
Câmera frontal	13 MP
Bateria	5.000 mAh, carregamento rápido 25 W
Conectividade	4G LTE, Wi-Fi, Bluetooth 5.3, USB-C
Dimensões	160,1 x 76,8 x 8,4 mm
Peso	200 g

Fonte: TechTudo (2024) (TECHTUDO, 2024).

Para a etapa de análise dos resultados, foi adotado um critério binário de validação, cujo objetivo foi avaliar a eficiência do processamento das imagens capturadas. Considerou-se como processamento correto os casos em que todos os caracteres exibidos no visor do oxímetro foram identificados corretamente pelo sistema, sem qualquer omissão ou erro de leitura. Por outro lado, qualquer discrepância, seja na interpretação dos caracteres ou no enquadramento da Região de Interesse (ROI), foi classificada como processamento incorreto.

Ao todo, foram utilizadas 56 imagens coletadas a partir do oxímetro G-Tech OLED Graph, processadas pelo aplicativo instalado no smartphone Galaxy A15. A Figura 12 apresenta um exemplo de imagem empregada nos testes, ilustrando a forma como os dados eram capturados e submetidos ao reconhecimento automático.

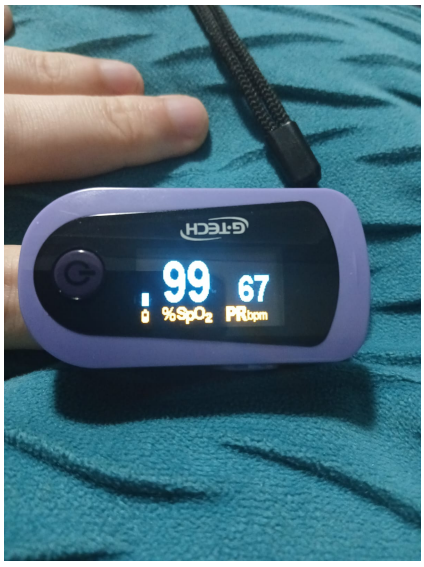

Figura 12 – Exemplo de imagem capturada do visor do oxímetro G-Tech OLED Graph.

A consolidação dos resultados está apresentada na Tabela 8, que resume o desempenho do sistema frente às condições de validação propostas.

Tabela 8 – Resultados da validação do reconhecimento das imagens do oxímetro

	Oxímetro
Acertos	50
Erros	6
Acurácia (%)	89%

Fonte - Elaborado pelo autor.

A partir da análise dos 56 registros processados, observou-se que o sistema obteve 50 acertos e 6 erros, resultando em uma acurácia de 89%. Esse índice evidencia que, na maioria dos casos, o aplicativo foi capaz de realizar o reconhecimento correto dos caracteres exibidos pelo oxímetro, confirmando a viabilidade do uso da biblioteca ML Kit em conjunto com as estratégias de pré-processamento adotadas.

No entanto, os 11% de falhas identificados revelam a existência de fatores que impactam a robustez do sistema. Entre as principais causas potenciais de erro, destacam-se:

- **Variações na iluminação durante a captura das imagens**, que podem gerar sombras ou reflexos no visor;
- **Posicionamento inadequado do smartphone em relação ao oxímetro**, comprometendo a definição da Região de Interesse (ROI);
- **Caracteres parcialmente exibidos ou de baixo contraste**, o que dificulta a segmentação e posterior reconhecimento.

6.2 Modelo TensorFlow

A avaliação de modelos de redes neurais convolucionais é realizada com base em métricas que permitem mensurar sua capacidade de generalização e seu desempenho em dados de treinamento e validação. As principais métricas utilizadas neste trabalho foram acurácia, função de perda (*loss*) e a matriz de confusão.

A acurácia é a métrica mais utilizada em problemas de classificação e mede a proporção de predições corretas em relação ao número total de amostras avaliadas (SHALEV-SHWARTZ; BEN-DAVID, 2014). Sua definição matemática é dada por:

$$\text{Acurácia} = \frac{TP + TN}{TP + TN + FP + FN}$$

em que:

- *TP (True Positives)* são os verdadeiros positivos,
- *TN (True Negatives)* são os verdadeiros negativos,
- *FP (False Positives)* são os falsos positivos,
- *FN (False Negatives)* são os falsos negativos.

No treinamento do modelo, a Figura 13 apresenta a evolução da acurácia ao longo de 100 épocas. Observa-se que tanto nos dados de treinamento quanto nos de validação os valores ultrapassam rapidamente 99,16%, estabilizando-se em níveis elevados, com margens mínimas de variação. Isso demonstra que a rede neural conseguiu aprender de forma eficiente e consistente.

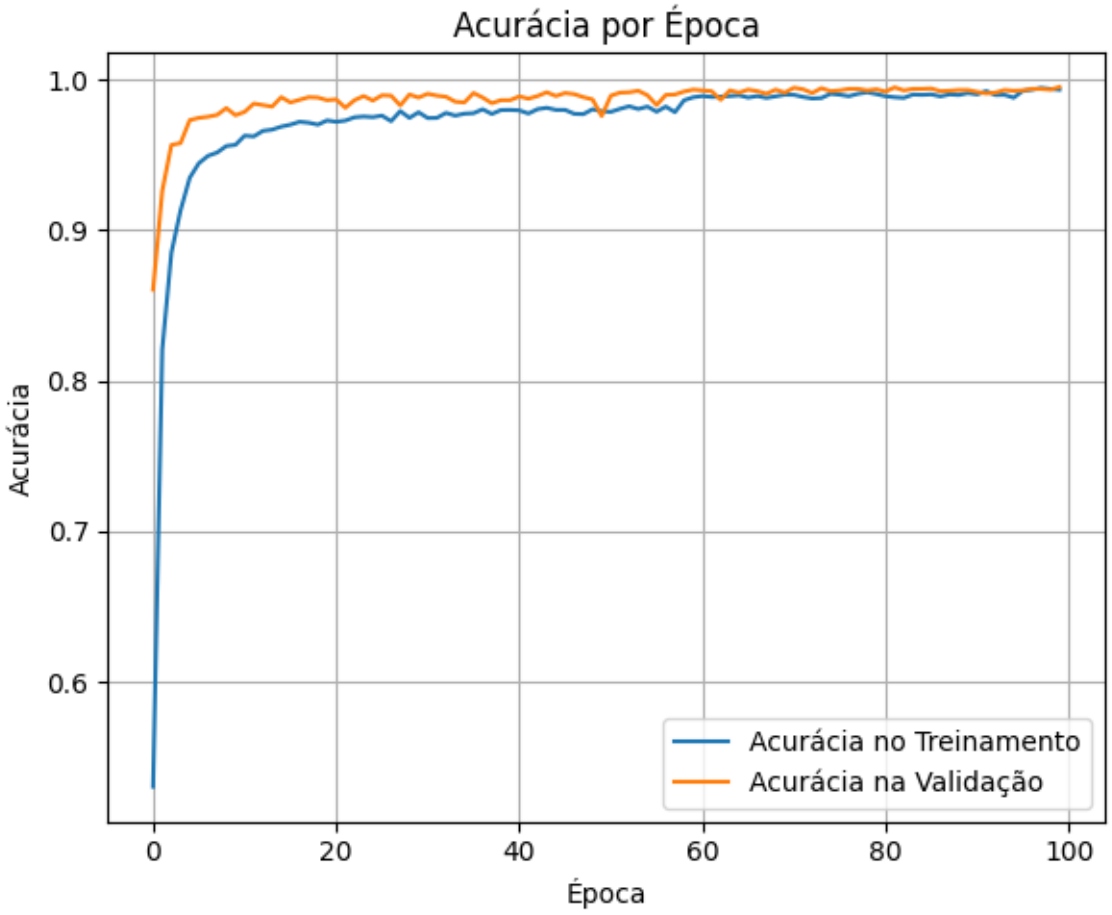

Figura 13 – Gráfico Acurácia por Época.

A função de perda é responsável por quantificar a discrepância entre as classes das amostras de entrada e das classes preditas pelo modelo durante o processo de aprendizado, sendo usada pelo algoritmo de otimização para ajustar os pesos da rede. Para problemas de classificação multiclasse, a função mais empregada é a entropia cruzada categórica (*categorical cross-entropy*) (BISHOP; BISHOP, 2023), definida como:

$$\mathcal{L} = -\sum_{i=1}^{N}\sum_{c=1}^{M} y_{i,c} \cdot \log(\hat{y}_{i,c})$$

em que:

- N é o número de amostras,
- M é o número de classes,
- $y_{i,c}$ é a variável indicadora (1 se a classe correta da amostra i for c, caso contrário 0),
- $\hat{y}_{i,c}$ é a probabilidade prevista pelo modelo de que a amostra i pertença à classe c.

Na Figura 14, observa-se que a função de perda apresenta uma queda acentuada nas primeiras épocas de treinamento, convergindo para valores muito baixos (próximos de zero). Esse comportamento indica que o modelo ajustou-se adequadamente aos dados, reduzindo significativamente os erros de classificação sem apresentar indícios de overfitting, visto que as curvas de validação acompanham as de treinamento.

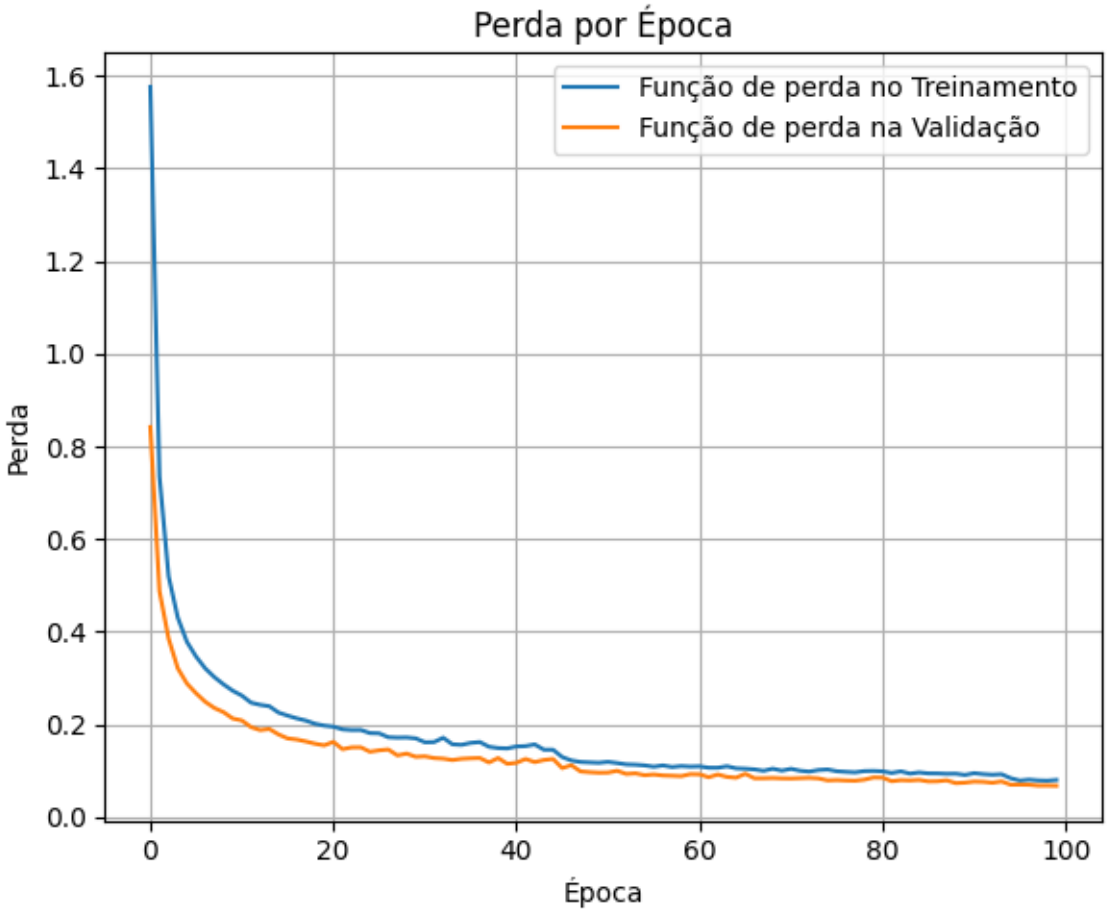

Figura 14 – Gráfico Perda por Época.

A matriz confusão é uma ferramenta amplamente utilizada para avaliar o desempenho de modelos de classificação supervisionada. Essa matriz organiza os resultados das previsões do classificador em forma de matriz, permitindo comparar as classes *reais* com as classes *preditas*.

Formalmente, o elemento $C_{i,j}$ da matriz de confusão representa o número de amostras cuja classe real é i e que foram classificadas pelo modelo como pertencentes à classe j:

$$C_{i,j} = \text{número de amostras da classe real } i \text{ previstas como classe } j,$$

onde $C_{i,j}$ corresponde ao elemento na linha i e coluna j.

A diagonal principal da matriz ($C_{i,i}$) contém os **acertos** do classificador, ou seja, os casos em que a classe predita coincide com a classe verdadeira. Já os elementos fora da diagonal correspondem aos **erros de classificação**, revelando quais classes o modelo tende a confundir.

Essa estrutura é especialmente útil porque permite extrair diversas métricas de desempenho, como *acurácia*, *precisão*, *revocação (sensibilidade)*, *F1-score* e a curva ROC, que fornecem uma visão mais completa da qualidade do modelo do que apenas a taxa global de acerto.

Na Figura 15, nota-se uma forte predominância de valores concentrados na diagonal principal, o que significa que o modelo foi capaz de classificar corretamente a maioria absoluta das imagens de dígitos. Os poucos erros aparecem como valores residuais fora da diagonal, demonstrando alto desempenho global da rede.

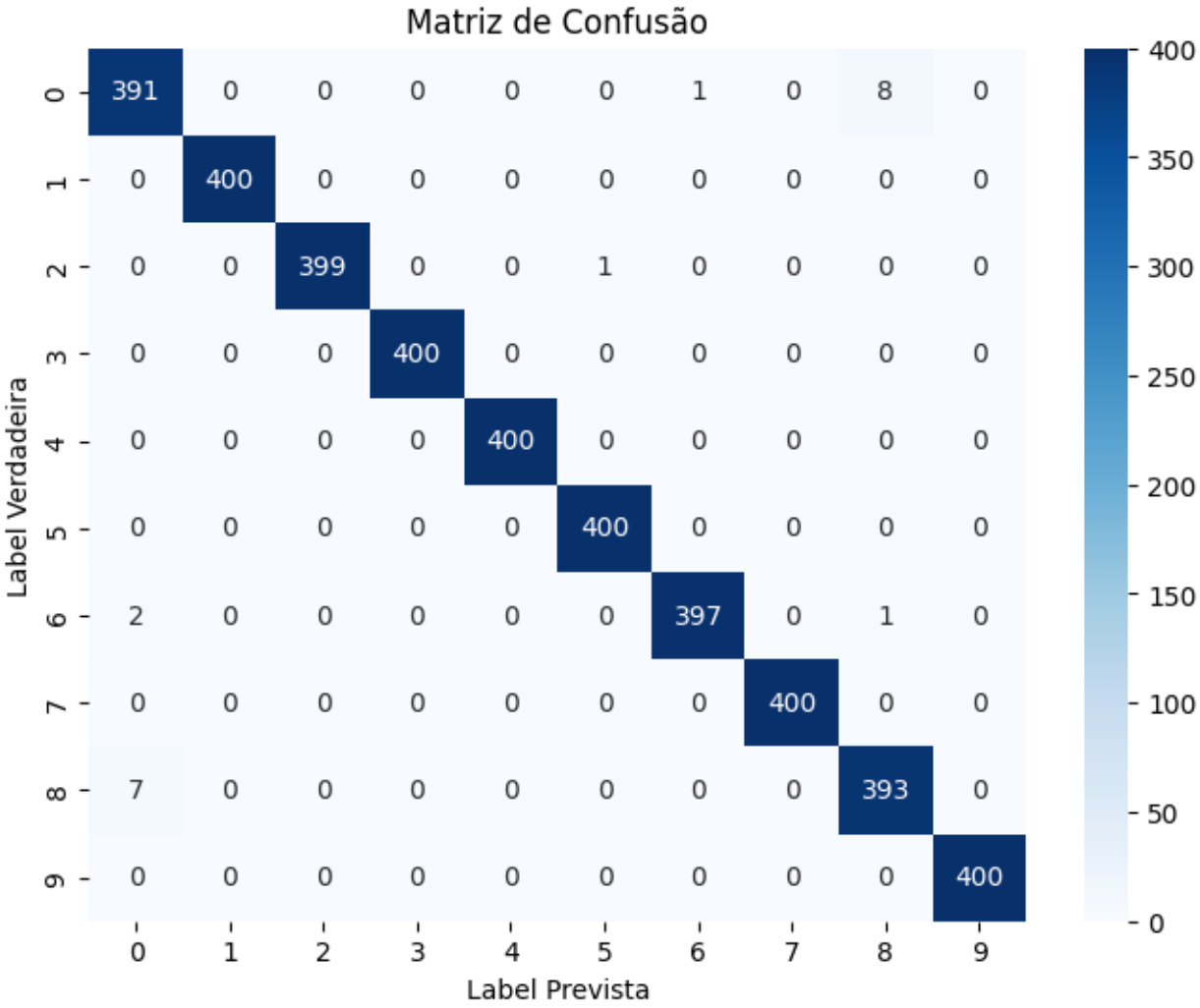

Figura 15 – Matriz Confusão.

Para ilustrar o funcionamento do modelo desenvolvido, a Figura 16 apresenta um exemplo prático de captura e reconhecimento de um dígito em um display de sete segmentos. Na parte superior da imagem, observa-se a etapa de aquisição, em que a região de interesse (ROI) foi delimitada sobre o dígito exibido no display. Em seguida, o dígito extraído passa pelo processo de pré-processamento (parte inferior da figura), no qual ocorre a segmentação e normalização da imagem para adequação ao modelo.

O resultado do reconhecimento é mostrado logo abaixo, onde o sistema classificou corretamente o dígito como **7**, com um nível de confiança de 99,37%. Esse exemplo evidencia a eficácia do modelo em lidar com dados reais, mantendo elevado desempenho mesmo em condições de iluminação variada.

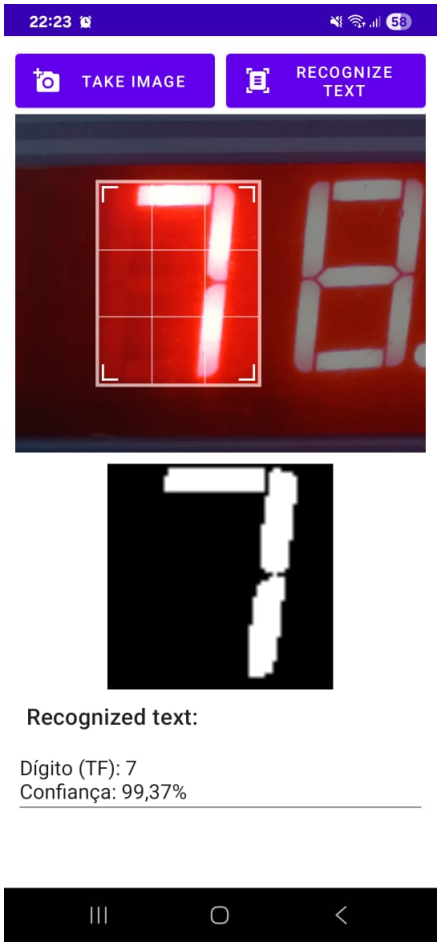

Figura 16 – Exemplo de reconhecimento display sete segmento.

Apesar dos resultados promissores apresentados pelo modelo desenvolvido em TensorFlow Lite, é importante reconhecer suas limitações atuais. O treinamento realizado concentrou-se em um conjunto de dados relativamente restrito, baseado em um único tipo de dispositivo e em condições de captura controladas. Esse recorte limita a capacidade de generalização do modelo, sobretudo quando exposto a displays de sete segmentos de diferentes fabricantes, variações no design dos dígitos, contrastes reduzidos e condições adversas de iluminação. Essa restrição evidencia a necessidade de treinamento com um *dataset* mais abrangente e diverso, contemplando uma gama maior de cenários de uso. A incorporação dessa diversidade tende a tornar o modelo mais robusto e confiável, reduzindo a ocorrência de erros residuais e ampliando sua aplicabilidade em situações reais de telemonitoramento, nas quais a heterogeneidade dos dispositivos utilizados pelos pacientes é uma característica inevitável.

6.3 Comparação com Modelos Baseados em LLMs

Nos últimos anos, modelos baseados em arquiteturas *Transformer* passaram a ocupar posição de destaque no reconhecimento óptico de caracteres, especialmente em

cenários complexos envolvendo textos variados, documentos extensos ou fontes irregulares. Entre as soluções mais representativas, destacam-se modelos como o TrOCR (LI et al., 2021a), o DONUT (*Document Understanding Transformer*) (KIM et al., 2022) e variações como o *LayoutLMv3* (HUANG et al., 2022), que integram mecanismos de atenção global capazes de relacionar elementos textuais e visuais com alto nível de contextualização.

Esses modelos apresentam desempenho superior em tarefas de OCR de propósito geral, graças à capacidade de capturar padrões semânticos, interpretar estruturas de documentos e operar de forma robusta mesmo diante de ruído significativo, distorções ou fontes incomuns. Entretanto, tais vantagens estão associadas a requisitos computacionais substancialmente maiores, exigindo normalmente execução em servidores com unidades de processamento acelerado (GPU ou TPU), além de dependerem de comunicação com a nuvem para inferência em aplicações móveis.

Ao comparar essas soluções de ponta com a metodologia proposta neste trabalho, observa-se que a aplicação de LLMs não oferece benefícios práticos significativos no contexto específico da leitura de dígitos em displays de sete segmentos. Esse tipo de display apresenta padrões altamente estruturados e previsíveis, caracterizados por segmentos discretos e pouco variabilidade entre dispositivos, cenário no qual modelos especializados, como o desenvolvido em TensorFlow Lite, tendem a apresentar performance mais eficiente. O modelo embarcado alcançou acurácia de 99,16%, operando localmente no dispositivo, com baixa latência e sem dependência de conectividade de rede, atendendo plenamente às exigências do telemonitoramento.

Além disso, a execução local favorece a privacidade e elimina o envio de imagens contendo dados médicos para servidores externos, aspecto crítico em aplicações sensíveis e reguladas (HAWRYLISZYN; COELHO; BARJA, 2021). Por outro lado, modelos baseados em LLMs apresentam maior latência, maior custo de execução e implicações legais relativas ao tratamento de dados – fatores que se contrapõem aos princípios de simplicidade, baixo custo e operação offline estabelecidos como diretrizes deste projeto.

Assim, embora LLMs representem o estado da arte em OCR genérico, os resultados obtidos indicam que a abordagem adotada, combinação entre ML Kit On-Device e o modelo customizado em TFLite, mostra-se mais adequada ao objetivo do sistema. A acurácia obtida, aliada ao baixo custo computacional, escalabilidade e preservação da privacidade, demonstra que a solução especializada proporciona melhor relação entre desempenho e viabilidade para o reconhecimento em displays utilizados em dispositivos médicos domésticos.

7 Conclusão

O presente Trabalho de Graduação em Engenharia teve como objetivo propor, desenvolver e validar um protótipo de aplicativo móvel com visão computacional voltado ao monitoramento remoto de dados médicos. A proposta partiu da constatação de que muitos dispositivos de uso doméstico, como oxímetros, glicosímetros, balanças e medidores de pressão arterial, exibem suas medições apenas em displays digitais, em geral baseados em sete segmentos, mas não oferecem mecanismos de conectividade nativa para envio automático dos dados a plataformas de saúde. Essa limitação torna necessária a transcrição manual das informações pelo paciente ou cuidador, processo que é sujeito a erros, omissões e atrasos, comprometendo a confiabilidade do acompanhamento remoto.

Frente a esse cenário, a solução proposta buscou oferecer um meio simples, acessível e de baixo custo para digitalizar esses valores diretamente por meio da câmera de um smartphone. Para isso, foi implementado um fluxo de captura, pré-processamento, reconhecimento óptico de caracteres e exibição estruturada dos resultados, utilizando como base duas tecnologias principais: a biblioteca ML Kit Text Recognition, fornecida pelo Google, e um modelo customizado treinado em TensorFlow Lite. A combinação de ambas as abordagens foi decisiva para equilibrar praticidade e precisão, viabilizando uma aplicação que funciona inteiramente no dispositivo móvel, sem depender de servidores externos, reduzindo riscos de privacidade e aumentando a escalabilidade.

Os resultados experimentais demonstraram que o protótipo é funcional e promissor. Na etapa de testes com o ML Kit, foram processadas 56 imagens capturadas de um oxímetro G-Tech OLED Graph, resultando em 50 acertos e 6 erros, com acurácia global de 89%. Esse valor, embora relativamente elevado revelou vulnerabilidades em cenários com baixa iluminação, reflexos no visor, enquadramento inadequado ou caracteres de baixo contraste. Por outro lado, o modelo customizado em TensorFlow Lite apresentou desempenho significativamente superior, alcançando acurácia acima de 99% no conjunto de validação, com rápida convergência da função de perda e excelente robustez frente a variações artificiais inseridas via *data augmentation*. A análise da matriz de confusão permitiu a constatação da consistência do modelo, mostrando que os erros residuais ficaram concentrados em classes específicas, sem comprometer a confiabilidade geral.

A comparação entre as duas abordagens permitiu concluir que o ML Kit funciona bem como solução de primeira abordagem, oferecendo rapidez e integração direta com Android, mas carece de especialização para lidar com displays de sete segmentos. Já o modelo customizado em TensorFlow Lite demonstrou ser a solução mais robusta para esse tipo de aplicação, garantindo maior precisão, ainda que às custas de maior complexidade

no treinamento. O uso combinado, portanto, constitui uma estratégia eficaz: o ML Kit pode atuar em cenários simples e, em caso de falha, o sistema aciona automaticamente o modelo especializado.

Entre as contribuições mais relevantes deste trabalho, destacam-se:

- A comprovação da viabilidade de um sistema de telemonitoramento baseado apenas em smartphones, sem a necessidade de hardware adicional;
- A integração entre duas tecnologias complementares de OCR, conciliando desempenho e acessibilidade;
- A proposição de um pipeline de pré-processamento adaptado para imagens de visores de sete segmentos, incluindo binarização adaptativa e técnicas de correção;
- A demonstração prática da aplicabilidade do sistema em cenários cotidianos de acompanhamento médico, especialmente no cuidado de pacientes crônicos e idosos.

Por outro lado, o trabalho também evidenciou limitações importantes. A sensibilidade do sistema a condições ambientais adversas é um ponto que precisa ser tratado em futuras evoluções, especialmente em contextos de uso domiciliar, nos quais não há controle sobre iluminação e posicionamento. Além disso, o escopo experimental concentrou-se em um único dispositivo (oxímetro), o que restringe a generalização dos resultados. Outro aspecto a ser aprimorado é a ergonomia da interface, de forma a tornar a seleção da Região de Interesse (ROI) mais intuitiva, reduzindo a dependência da habilidade do usuário em enquadrar corretamente o visor.

Adicionalmente, identificou-se que, apesar do alto desempenho obtido em condições controladas, o modelo em TensorFlow Lite ainda apresenta restrições quando exposto a diferentes tipos de displays de sete segmentos. Essa limitação decorre do uso de um conjunto de treinamento reduzido e pouco diverso, concentrado em um único dispositivo e cenário de captura. Para que o sistema alcance maior capacidade de generalização e robustez em situações reais, torna-se essencial a criação de um *dataset* mais representativo, composto por imagens coletadas em diferentes dispositivos, marcas e condições de uso. O re-treinamento do modelo a partir desse conjunto ampliado permitirá reduzir erros residuais, aumentar a confiabilidade e aproximar o protótipo das exigências de aplicações práticas em larga escala no contexto do SUS e da telemedicina.

Diante desses pontos, diversas perspectivas de trabalhos futuros emergem naturalmente:

- Treinamento de modelos ainda mais especializados, utilizando bases de dados ampliadas e capturadas em condições reais de uso, incluindo diferentes marcas e modelos de dispositivos médicos;

- Automação da seleção da ROI, por meio de algoritmos de detecção de objetos, evitando que o usuário precise delimitar manualmente a área de interesse;
- Expansão do protótipo para outros dispositivos médicos de uso doméstico, como balanças digitais e glicosímetros, de modo a abranger um conjunto mais amplo de parâmetros fisiológicos;
- Integração com plataformas de telemedicina já utilizadas no SUS, permitindo que os dados capturados sejam enviados de forma padronizada e segura para os profissionais de saúde;
- Validação em campo, com testes realizados em ambientes clínicos e domiciliares, envolvendo pacientes reais e equipes médicas, de modo a avaliar não apenas a acurácia técnica, mas também a aceitação, usabilidade e impacto no acompanhamento clínico.

A relevância social e científica deste trabalho reside na possibilidade de oferecer uma alternativa concreta para ampliar o acesso à saúde digital. O Sistema Único de Saúde (SUS) enfrenta desafios crônicos de sobrecarga e dificuldade de cobertura em regiões remotas. Uma solução móvel, de baixo custo e fácil disseminação, pode contribuir para reduzir desigualdades de acesso, aumentar a agilidade no acompanhamento de doenças crônicas e melhorar a comunicação entre pacientes e profissionais. Do ponto de vista científico, o trabalho avança no estado da arte ao propor uma integração prática entre técnicas consolidadas de OCR e modelos customizados de *deep learning*, com foco em um problema específico pouco atendido pelas bibliotecas comerciais: a leitura de displays de sete segmentos.

Em síntese, este projeto demonstrou que a visão computacional aliada à inteligência artificial pode ser aplicada com sucesso em cenários de telemedicina, oferecendo resultados confiáveis sem necessidade de infraestrutura complexa. O protótipo desenvolvido não apenas comprova a viabilidade técnica da proposta, como também abre caminho para futuras inovações na área de saúde digital. Assim, este trabalho se insere em um esforço mais amplo de transformação digital na saúde, no qual soluções acessíveis e centradas no paciente desempenham papel estratégico na construção de um sistema mais eficiente, inclusivo e sustentável.

Referências

ABADI, M. et al. Tensorflow: A system for large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016. Citado na página 28.

ADI, S. E.; CASSON, A. J. Design and optimization of a tensorflow lite deep learning neural network for human activity recognition on a smartphone. In: *2021 43rd Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)*. [S.l.: s.n.], 2021. p. 7028–7031. Citado na página 12.

ALKHUZAIMI, F. et al. The impact of mobile health interventions on service users' health outcomes and the role of health professions: a systematic review of systematic reviews. *BMC Digital Health*, v. 3, n. 3, p. 3, 2025. Citado na página 2.

ALMEIDA, P. O. d.; PROTASIO, K. d. A. P.; REIS, A. d. S. Desigualdades regionais na saúde: mudanças observadas no Brasil de 2000 a 2016. *Ciência & Saúde Coletiva*, v. 26, n. 8, p. 3349–3360, 2021. Citado na página 2.

AMIRI, P.; NADRI, H.; BAHAADINBEIGY, K. Facilitators and barriers of mhealth interventions during the covid-19 pandemic: systematic review. *BMC Health Services Research*, v. 23, n. 1, p. 1176, 2023. Citado na página 2.

BASHSHUR, R. L. et al. The empirical foundations of telemedicine interventions for chronic disease management. *Telemedicine and e-Health*, v. 20, n. 9, p. 769–800, 2014. Citado na página 2.

BELLA, L. D. *Dez anos de Kotlin!* 2021. <https://blog.jetbrains.com/pt-br/kotlin/2021/08/ten-years-of-kotlin/#>. Acessado em: 19 de julho de 2025. Citado 2 vezes nas páginas 16 e 17.

BERATARRECHEA, A. et al. The impact of mobile health interventions on chronic disease outcomes in developing countries: a systematic review. *Telemedicine and e-Health*, v. 20, n. 1, p. 75–82, jan. 2014. Epub 2013 Nov 8. Citado na página 6.

BISHOP, C. M.; BISHOP, H. *Deep Learning: Foundations and Concepts*. Cham: Springer Nature, 2023. ISBN 978-3031234540. Citado na página 41.

CFM. *Resolução CFM nº 2.314/2022 – Define e regulamenta a telemedicina como forma de serviços médicos mediados por tecnologias de comunicação*. 2022. Publicado no Diário Oficial da União em 05 de maio de 2022. Disponível em PDF no site do CFM. Citado na página 4.

CLERON, M. *Android Announces Support for Kotlin*. 2017. <https://android-developers.googleblog.com/2017/05/android-announces-support-for-kotlin.html>. Acessado em: 19 de julho de 2025. Citado 2 vezes nas páginas 17 e 19.

CLERON, M. *Android Announces Support for Kotlin*. 2017. <https://android-developers.googleblog.com/2017/05/android-announces-support-for-kotlin.html>. Acessado em: 19 de julho de 2025. Citado na página 19.

DATAREPORTAL. *Digital 2025: April Global Statshot Report*. 2025. Accessed: 2025-06-28. Disponível em: <https://datareportal.com/reports/digital-2025-april-global-statshot>. Citado na página 6.

DENG, Y. Deep learning on mobile devices - a review. *SPIE Defense + Commercial Sensing*, 2019. Invited Paper. Acesso em: 07 dez. 2025. Approved for public release; unlimited distribution. Not export controlled per ES-FL-021919-0046. Citado 2 vezes nas páginas 36 e 37.

FINNEGAN, E. et al. Automated method for detecting and reading seven-segment digits from images of blood glucose metres and blood pressure monitors. *Healthcare Technology Letters*, v. 6, n. 5, p. 152–157, 2019. Citado 7 vezes nas páginas 3, 12, 31, 32, 33, 34 e 35.

FREIRE, M. et al. Telemedicina no acesso à saúde durante a pandemia de covid-19: uma revisão de escopo. *Rev Saude Publica*, v. 57, n. Supl 1, p. 4s, 2023. Citado na página 1.

GONZALEZ, R. C.; WOODS, R. E. *Processamento Digital de Imagens*. 3. ed. São Paulo: Pearson Prentice Hall, 2010. Tradução da obra original: Digital Image Processing. Revisão técnica de Marcelo A. da C. Vieira e Mauricio C. Escarpinati. Citado 2 vezes nas páginas 9 e 10.

GOOGLE. *ML Kit - Google for Developers*. 2024. <https://developers.google.com/ml-kit?hl=pt-br>. Acessado em: 5 de agosto de 2025. Citado na página 36.

GOOGLE. *Kotlin no Android*. 2025. <https://developer.android.com/kotlin?hl=pt-br>. Acessado em: 19 de julho de 2025. Citado na página 17.

GOOGLE. *Kotlin no Android*. 2025. <https://developer.android.com/kotlin?hl=pt-br>. Acessado em: 19 de julho de 2025. Citado na página 18.

GOOGLE. *Kotlin no Android*. 2025. <https://developer.android.com/kotlin?hl=pt-br>. Acessado em: 19 de julho de 2025. Citado na página 19.

GOOGLE. *ML Kit Documentation*. [S.l.], 2025. Disponível em: <https://developers.google.com/ml-kit>. Acesso em: 31 ago. 2025. Citado 2 vezes nas páginas 25 e 26.

GOOGLE. *TensorFlow Lite*. [S.l.], 2025. Disponível em: <https://www.tensorflow.org/lite>. Acesso em: 31 ago. 2025. Citado 2 vezes nas páginas 25 e 26.

GÜLER, N. F.; ÜBEYLI, E. D. Theory and applications of biotelemetry. *Journal of Medical Systems*, v. 26, n. 2, p. 159–178, apr 2002. Citado na página 5.

HAN, S.; MAO, H.; DALLY, W. J. *Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding*. 2016. ArXiv preprint arXiv:1510.00149. Disponível em: <https://arxiv.org/abs/1510.00149>. Acesso em: 31 ago. 2025. Citado na página 26.

HAWRYLISZYN, L. O.; COELHO, N. G. S. C.; BARJA, P. R. Lei geral de proteÇÃo de dados (LGPD): O desafio de sua implantaÇÃo para a saÚde. *Revista Univap*, v. 27, n. 54, 2021. Acesso em: 07 dez. 2025. Disponível em: <https://revista.univap.br/index.php/revistaunivap/article/view/2589>. Citado 2 vezes nas páginas 37 e 46.

HOCHREITER, S.; SCHMIDHUBER, J. Long short-term memory. *Neural Computation*, v. 9, n. 8, p. 1735–1780, 1997. Citado na página 25.

HUANG, Y. et al. Layoutlmv3: Pre-training for document ai with unified text and image masking. *arXiv preprint arXiv:2204.08387v3*, 2022. Versão v3, publicada em: 19 jul. 2022. Acesso em: 07 dez. 2025. Disponível em: <https://arxiv.org/abs/2204.08387v3>. Citado na página 46.

JAIN, A. *Understanding Convolutional Neural Networks (CNNs) with an Example on the MNIST Dataset*. 2025. <https://medium.com/@abhishekjainindore24/understanding-convolutional-neural-networks-cnns-with-an-example-on-the-mnist-dataset-a64815843
Acesso em: 3 set. 2025. Citado na página 33.

JETBRAINS. *FAQ - Kotlin Programming Language*. 2010. <https://kotlinlang.org/docs/faq.html>. Acessado em: 16 de julho de 2025. Citado na página 16.

KAGGLECOMMUNITY. *Digits Segment Display Dataset*. 2024. <https://www.kaggle.com/datasets>. Acessado em: 10 ago. 2025. Citado na página 30.

KANAGARATHINAM, K.; SEKAR, K. Text detection and recognition in raw image dataset of seven segment digital energy meter display. *Energy Reports*, Elsevier, v. 5, p. 842–852, 2019. Citado na página 12.

KAY, M.; SANTOS, J.; TAKANE, M. *mHealth: New horizons for health through mobile technologies*. Genève, Suíça, 2011. Citado na página 6.

KIESSLING, B. *Kraken – An Universal Text Recognizer for the Humanities*. 2019. Digital Humanities Conference (DH). Disponível em: <https://kraken.re/>. Acesso em: 31 ago. 2025. Citado na página 26.

KIM, G. et al. OCR-free document understanding transformer. *arXiv preprint arXiv:2111.15664v5*, 2022. Versão v5, publicada em: 6 out. 2022. Acesso em: 07 dez. 2025. Disponível em: <https://arxiv.org/abs/2111.15664v5>. Citado na página 46.

KOTLIN FOUNDATION. *Introduction*. 2025. <https://kotlinlang.org/spec/introduction.html>. Acessado em: 19 de julho de 2025. Citado na página 17.

KOTLIN FOUNDATION. *Introduction*. 2025. <https://kotlinlang.org/spec/introduction.html>. Acessado em: 19 de julho de 2025. Citado na página 17.

KOTLIN TEAM. *Null Safety*. 2025. <https://kotlinlang.org/docs/null-safety.html>. Acessado em: 19 de julho de 2025. Citado na página 18.

KRISHNAMOORTHY, G.; KONIDENA, B. K. Machine learning in edge computing: Opportunities and challenges. *International Journal of Innovative Science and Research Technology*, v. 9, n. 2, Fevereiro 2024. Citado na página 37.

KUMAR, A.; PATEL, R.; SINGH, M. Ocular biometry ocr: A deep learning approach for text detection and recognition in medical imaging. *Frontiers in Artificial Intelligence*, v. 7, p. 1428716, 2024. Disponível em: <https://www.frontiersin.org/articles/10.3389/frai.2024.1428716/full>. Citado na página 25.

LAZZARA, G.; GéRAUD, T. Efficient multiscale *sauvola*'s binarization. *International Journal of Document Analysis and Recognition (IJDAR)*, v. 17, n. 2, p. 105–123, 2014. Acesso em: 21 jun. 2025. Citado na página 33.

LECUN, Y.; BENGIO, Y. Convolutional networks for images, speech, and time series. In: ARBIB, M. A. (Ed.). *The handbook of brain theory and neural networks*. Cambridge, MA: MIT Press, 1998. p. 255–258. Citado na página 25.

LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. *Nature*, v. 521, n. 7553, p. 436–444, 2015. Citado na página 11.

LI, M. et al. Trocr: Transformer-based optical character recognition with pre-trained models. *arXiv preprint arXiv:2109.10282*, 2021. Acesso em: 07 dez. 2025. Disponível em: <https://arxiv.org/abs/2109.10282>. Citado na página 46.

LI, M. et al. *TrOCR: Transformer-based Optical Character Recognition with Pre-trained Models*. 2021. ArXiv preprint arXiv:2109.10282. Disponível em: <https://arxiv.org/abs/2109.10282>. Acesso em: 31 ago. 2025. Citado na página 26.

LIU, W. et al. Ssd: Single shot multibox detector. In: *European Conference on Computer Vision (ECCV)*. [S.l.: s.n.], 2016. p. 21–37. Citado na página 11.

LIU, X.; CHEN, L.; ZHANG, J. Smartphone-based real-time object recognition architecture for portable and constrained systems. *Journal of Real-Time Image Processing*, v. 18, n. 6, p. 2043–2056, 2021. Citado na página 12.

MALDONADO, J. M. S. d. V.; MARQUES, A. B.; CRUZ, A. Telemedicina: desafios à sua difusão no brasil. *Cadernos de Saúde Pública*, v. 32, n. Sup 2, p. e00155615, 2016. Citado na página 1.

MAMA. *Mobile Alliance for Maternal Action (MAMA)*. 2010–2016. Iniciativa global de mHealth para saúde materna e infantil, apoiada pela OMS, USAID, GSMA e Johnson & Johnson. Acesso em: 2025-06-28. Disponível em: <https://www.mobilemamaalliance.org/>. Citado na página 7.

MAUNG, S. Z. M.; AYE, N. Convolutional neural network based recognition of myanmar text warning sign for mobile platform. *International Journal of Engineering Research & Technology (IJERT)*, v. 8, n. 1, p. 216–221, 2019. Acesso em: 21 jun. 2025. Disponível em: <https://www.ijert.org/research/convolutional-neural-network-based-recognition-of-myanmar-text-warning-sign-for-mobile-platform-IJ pdf>. Citado na página 30.

MOREIRA, J. L.; NETO, F. et al. Automated medical device display reading using deep learning. *Sensors*, MDPI, v. 22, n. 17, p. 6398, 2022. Citado na página 12.

NETTO, A. V. Aplicação do cuidado híbrido como mecanismo de ação na construção de uma terapêutica digital. *einstein (São Paulo)*, v. 18, n. eMD5640, 2020. Citado na página 1.

NETTO, A. V.; TATEYAMA, A. G. P. Avaliação de tecnologia de telemonitoramento e biotelemetria para o cuidado híbrido para o idoso com condição crônica. *Journal of Health Informatics*, v. 10, n. 4, p. 103–111, 2018. Citado 2 vezes nas páginas 3 e 4.

NICHIATA, L. Y. I.; PASSARO, T. mHealth e saúde pública: a presença digital do Sistema Único de Saúde do Brasil por meio de aplicativos de dispositivos móveis. *RECIIS - Revista Eletrônica de Comunicação, Informação & Inovação em Saúde*, v. 17, n. 3, p. 503–516, 2023. Citado na página 2.

OLIVEIRA, J. e. et al. Saúde digital e a plataformização do Estado brasileiro. *Ciência & Saúde Coletiva*, v. 28, n. 7, p. 2143–2153, 2023. Citado na página 2.

OTSU, N. A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, v. 9, n. 1, p. 62–66, 1979. Acesso em: 21 jun. 2025. Citado na página 33.

PRAKRUTHI, K.; SRINATH, K.; RAMAKRISHNA, R. Application of convolutional neural networks in mobile devices for inferring readings from medical apparatus. *International Journal of Research and Scientific Innovation*, RSIS International, v. 4, n. 1, p. 49–52, 2017. Disponível em: <https://rsisinternational.org/IJRSI/Issue41/49-52.pdf>. Citado na página 12.

REDMON, J. et al. You only look once: Unified, real-time object detection. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. [S.l.: s.n.], 2016. p. 779–788. Citado na página 11.

ROCHA, T. A. H. et al. Saúde móvel: novas perspectivas para a oferta de serviços em saúde. *Epidemiologia e Serviços de Saúde*, v. 25, n. 1, p. 171–178, 2016. Citado na página 1.

SHALEV-SHWARTZ, S.; BEN-DAVID, S. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge: Cambridge University Press, 2014. ISBN 978-1107057135. Citado na página 40.

SMITH, R. An overview of the tesseract ocr engine. In: *Proceedings of the Ninth International Conference on Document Analysis and Recognition (ICDAR)*. [S.l.: s.n.], 2007. v. 2, p. 629–633. Citado na página 26.

TEAM, T. *TensorFlow Model Optimization Guide*. 2022. Acesso em: 10 ago. 2025. Disponível em: <https://www.tensorflow.org/model_optimization/guide>. Citado na página 29.

TECHTUDO. *Galaxy A15, ficha técnica*. 2024. <https://www.techtudo.com.br/guia/2024/02/galaxy-a15-e-bom-veja-ficha-tecnica-e-preco-do-celular-da-samsung-edmobile.ghtml>. Acesso em: 6 set. 2025. Citado na página 39.

TENSORFLOW AUTHORS. *TensorFlow: An Open Source Machine Learning Framework for Everyone*. 2024. <https://www.tensorflow.org/?hl=pt-br>. Acessado em: 10 de agosto de 2025. Citado na página 27.

VASLI, M. et al. The impact of mHealth interventions: Systematic review of systematic reviews. *International Journal of Medical Informatics*, v. 117, p. 57–69, 2018. Citado na página 7.

VOULODIMOS, A. et al. Deep Learning for Computer Vision: A Brief Review. *Computational Intelligence and Neuroscience*, v. 2018, p. 1–13, 2018. Citado na página 11.

WAN, C. et al. *MMOCR: A Comprehensive Toolbox for Text Detection, Recognition and Understanding*. 2021. ArXiv preprint arXiv:2108.06543. Disponível em: <https://arxiv.org/abs/2108.06543>. Acesso em: 31 ago. 2025. Citado na página 26.

WICK, C.; REUL, C.; PUPPE, F. Calamari – a high-performance ocr framework for historical prints. In: *2018 13th IAPR International Workshop on Document Analysis Systems (DAS)*. [S.l.]: IEEE, 2018. p. 433–438. Citado na página 26.

WINER, D. *Android's commitment to Kotlin*. 2019. <https://android-developers.googleblog.com/2019/12/androids-commitment-to-kotlin.html>. Acessado em: 19 de julho de 2025. Citado na página 17.

WORD HEALTH ORGANIZATION. *Global Diffusion of eHealth: Making Universal Health Coverage Achievable: Report of the Third Global Survey on eHealth*. Geneva: World Health Organization, 2016. Disponível em: <http://www.who.int/goe/publications/global_observatory_2016/en/>. Citado na página 4.

WORLD HEALTH ORGANIZATION. *mHealth: new horizons for health through mobile technologies: second global survey on eHealth*. [s.l.]: World Health Organization, 2011. 37 p. Citado na página 7.

YAN, R.; LIU, Z.; YANG, L. Deep Learning in Vision-Based Automated Inspection: Current State and Future Prospects. *Journal of Sensors*, v. 2021, p. 1–21, 2021. Citado na página 11.

YBARRA, M. L. et al. Pilot rct results of stop my smoking usa: a text messaging-based smoking cessation program for young adults. *Nicotine & Tobacco Research*, v. 15, n. 8, p. 1388–1399, ago. 2013. Citado na página 7.

ZOU, Y. et al. Effectiveness of mobile health interventions on diabetes and obesity treatment and management: Systematic review of systematic reviews. *JMIR mHealth and uHealth*, v. 8, n. 4, p. e15400, 2020. Disponível em: <https://mhealth.jmir.org/2020/4/e15400/>. Citado na página 7.