Visão Computacional para Detecção de Comportamentos Atentivos

CAIO LORRAN PEREIRA PIZA

Dissertação para obtenção do grau de Mestre em:

Engenharia Eletrotécnica e de Computadores.

Este trabalho foi realizado sob orientação de:

Prof. Dr. José Luís Lima

Prof. Dr. Marcos Roberto Bombacini

Bragança

2024

Visão Computacional para Detecção de Comportamentos Atentivos

CAIO LORRAN PEREIRA PIZA

Dissertação para obtenção do grau de Mestre em:

Engenharia Eletrotécnica e de Computadores.

Este trabalho foi realizado sob orientação de:

Prof. Dr. José Luís Lima

Prof. Dr. Marcos Roberto Bombacini

Bragança

2024

"É muito melhor lançar-se em busca de conquistas grandiosas, mesmo expondo-se ao fracasso, do que alinhar-se com os pobres de espírito, que nem gozam muito nem sofrem muito, porque vivem numa penumbra cinzenta, onde não conhecem nem vitória, nem derrota." (Theodore Roosevelt)

"Se o dinheiro for a sua esperança de independência, você jamais a terá. A única segurança verdadeira consiste numa reserva de sabedoria, de experiência e de competência." (Henry Ford)

Dedicatória

Dedico este trabalho, em primeiro lugar, à minha noiva, Gabrieli Pessoa Meroti, minha companheira do início ao fim desta trajetória. Ela comemorou da minha alegria nos momentos felizes, foi meu consolo nos momentos difíceis, e, em resumo, é o amor da minha vida.

Dedico também à minha família como um todo, com um agradecimento especial a Aparecida e Claudecir, Maura e Marcos, Hosana e Edilson, Matheus e Miguel, Beatriz e João, Caique e Camila. Mesmo à distância, estiveram presentes em minha vida, sendo meu alicerce e minha base, sempre prontos para me amparar, não importa onde eu caia.

Agradeço ao Saac, que me ofereceu serenidade e conforto durante os dias distantes ao saber que ao retornar me acolherá com carinho ao retornar.

Expresso também minha gratidão aos meus amigos, em especial Luiz, Endryl, Vinicius, Marcos, Eduardo, Vitor, Jonatas, Phablo e Pilasky. Agradeço por me ouvirem, por seus conselhos e por me fazerem parte de suas vidas, mesmo estando longe.

Por último, guardo meus mais profundos agradecimentos àqueles que não estão mais presentes. Em especial, à minha mãe, a pessoa mais importante da minha vida, que, ao enfrentar os desafios, me ensinou o verdadeiro significado da palavra "força". Ela compartilhou dos meus sonhos e foi minha cúmplice nesta jornada. Espero que, onde quer que esteja, possa sentir esta vitória. Ao Antônio, que, como figura paterna, me ensinou o valor do trabalho digno e a importância de aproveitar o presente, pois a vida é curta demais para ser vivida na mediocridade. E aos meus avós, que, em sua simplicidade, me mostraram que o verdadeiro fundamento da educação não é algo que se aprende em escolas, que a verdadeira riqueza não é um número no banco e que a honra é inegociável.

Agradecimentos

Agradeço ao meu orientador, Prof. Dr. José Luís Lima, e ao meu coorientador, Prof. Dr. Marcos Roberto Bombacini, por todo o auxílio, pelas ideias e pelas correções realizadas ao longo deste período. A orientação de ambos foi fundamental para o desenvolvimento deste trabalho, oferecendo direções valiosas e suporte constante.

Expresso também minha gratidão às instituições UTFPR e IPB, pela oportunidade de realizar a dupla diplomação. Esse apoio institucional foi essencial para a concretização deste projeto e para o avanço da minha formação acadêmica.

Resumo

Com o crescimento exponencial dos telemóveis, esses dispositivos tornaram-se essenciais no cotidiano, auxiliando em diversas atividades e trazendo soluções práticas para os desafios do dia a dia. Este trabalho explora o potencial destes equipamentos como ferramentas de apoio à segurança no trânsito, abordando especificamente o problema da distração visual ao volante. Para isso, foi desenvolvido um sistema inovador que combina redes neurais convolucionais com a funcionalidade de dispositivos móveis. A metodologia adotada focou na aquisição de um amplo conjunto de imagens para treinar um modelo de inteligência artificial capaz de classificar uma variável qualitativa em duas categorias distintas: atenção e distração do motorista. Em particular, o estudo concentrou-se na criação de uma aplicação móvel que utiliza a câmera do telemóvel para monitorizar o motorista e emitir alertas sonoros caso detecte distração prolongada. Os resultados obtidos destacaram a eficácia do modelo, especialmente após sua otimização para o formato TensorFlow Lite, adequado para dispositivos móveis. Com uma execução 11,6 vezes mais rápida que o modelo padrão e uma redução de tamanho de 18,5 MB para 3,86 MB, evidenciando alta eficiência em velocidade e consumo de recursos.

Palavras-chave: segurança no trânsito, redes neurais convolucionais, aplicativos móveis, detecção de distração

Abstract

With the exponential growth of smartphones, these devices have become essential in daily life, assisting in various tasks and providing practical solutions to everyday challenges. This paper explores the potential of smartphones as tools to support traffic safety, specifically addressing the issue of visual distraction while driving. To this end, an innovative system was developed that combines convolutional neural networks with the functionality of mobile devices. The adopted methodology focused on collecting a broad set of images to train an artificial intelligence model capable of classifying a qualitative variable into two distinct categories: driver attention and distraction. In particular, the study focused on creating a mobile application that uses the smartphone's camera to monitor the driver and issue auditory alerts if it detects prolonged distraction. The results obtained highlighted the model's effectiveness, especially after its optimization to the TensorFlow Lite format, making it suitable for mobile devices. With a runtime 11.6 times faster than the standard model and a size reduction from 18.5 MB to 3.86 MB, it demonstrates high efficiency in speed and resource consumption.

Keywords: traffic safety, convolutional neural networks, mobile applications, distraction detection

Conteúdo

Lista de Tabelas

Lista de Figuras

Capítulo 1

Introdução

No Capítulo 1, são apresentados uma introdução ao tema do trabalho, a motivação, os objetivos e a estrutura desta dissertação.

1.1 Contextualização

A presença constante de distrações permeia sutilmente a vida contemporânea, desafiando incessantemente nossas habilidades de concentração e foco. No contexto do trânsito, qualquer elemento que desvie a atenção do condutor para algo não relacionado à condução é considerado uma distração, comprometendo sua capacidade de reação em várias situações. As três principais formas de distração são identificadas como: visual, quando o condutor perde a visão da via; manual, caracterizada pelo desvio das mãos ou dos pés da condução; e cognitiva, manifestada pela falta de concentração [1]. Este trabalho propõe identificar o primeiro tipo de distração, a distração visual, estabelecendo estratégias para mitigá-la no contexto do trânsito.

Em 2018, a Organização Mundial da Saúde (OMS) trouxe à tona uma realidade alarmante ao destacar que o número anual de mortes no trânsito chegou a 1,35 milhão, tornando-se a principal causa de morte entre pessoas de 5 a 29 anos [2]. A OMS identificou a falta de atenção como um dos fatores críticos a serem abordados para reduzir esses números, levando à publicação, em 2011, do documento intitulado “Uso de telefone

celular: um problema crescente de distração ao volante" [3]. Neste sentido, este trabalho foi elaborado para conscientizar os motoristas sobre os riscos associados ao uso de dispositivos móveis durante a condução.

A urgência em enfrentar o desafio de reduzir o impacto da distração visual aumenta diante do uso disseminado de celulares. Um aumento de 2,1% no uso desses dispositivos, conforme relatado pelo Instituto Brasileiro de Geografia e Estatística (IBGE) em 2022, destaca a complexidade do cenário de distração ao qual estamos expostos [4]. A taxa, que passou de 84,4% para 86,5% das pessoas com 10 anos ou mais possuindo celulares, reflete não apenas mudanças na dinâmica social, mas também acentua o crescente desafio de manter o foco, especialmente no contexto do trânsito.

Estudos revelam números alarmantes de acidentes de trânsito em Portugal, com cerca de 1.000 mortes anuais e milhares de feridos graves, especialmente entre os jovens. Esses acidentes têm como principais causas a velocidade excessiva e distrações ao volante [5], [6]. O PENSE 2020, por exemplo, destaca iniciativas para a redução dos atropelamentos e a proteção de grupos vulneráveis [6]. Um estudo na cidade de Braga também analisou o perfil dos acidentes, destacando a gravidade das lesões e a vulnerabilidade dos pedestres [7].

A necessidade de implementar medidas para reduzir o número de acidentes de trânsito tem sido evidente nas últimas décadas. Em 2010, a Assembleia Geral das Nações Unidas emitiu uma resolução que definiu os esforços de 2011 a 2020 como a "primeira década de ação para a segurança no trânsito" [8]. O objetivo era conscientizar os países sobre a importância de tomar medidas eficazes para reduzir o número de mortes no trânsito. Como meta global, a ONU propôs uma redução de 50% na mortalidade até o final da década.

No entanto, um estudo conduzido pelo Instituto de Pesquisa Econômica Aplicada (Ipea), utilizando dados do DataSUS e ocorrências em rodovias brasileiras registradas pela Polícia Rodoviária Federal (PRF) entre 2010 e 2019, revelou um aumento de 2,3% na taxa de mortalidade por 100.000 habitantes [9]. Esses dados frustram a meta anteriormente estabelecida pela ONU, indicando a necessidade urgente de revisar e fortalecer

as estratégias adotadas para melhorar a segurança no trânsito.

Em outubro de 2021, a OMS, ciente do aumento contínuo das taxas de mortalidade no trânsito, lançou um projeto para a década de 2021 a 2031, mantendo a mesma meta ambiciosa de reduzir em 50% as fatalidades por acidentes [10].

Uma abordagem crescente para enfrentar esse obstáculo desafiador é o uso de Sistemas de Assistência ao Motorista (ADAS), que incorporam tecnologias nos veículos para prevenir erros humanos. Pesquisas demonstram como essas tecnologias contribuíram para uma redução significativa nos acidentes, alcançando aproximadamente uma diminuição de 19% nos incidentes causados por erros humanos [11]. No entanto, um desafio associado a essas inovações é a acessibilidade restrita, pois apenas uma pequena parcela da população pode adquirir veículos equipados com esses sistemas devido a seu alto valor.

Com o avanço crescente da Inteligência Artificial (IA) na identificação e análise de imagens, sua aplicação se expande em diversos setores, como na área da saúde. Estudos recentes, como o de Najjar (2023), demonstram que a IA melhorou significativamente a eficiência e a precisão no reconhecimento de padrões complexos em imagens médicas, beneficiando diagnósticos e tratamentos personalizados [12]. Esse progresso ilustra como a análise de imagem com IA possui grande potencial em outras áreas, trazendo melhorias ao facilitar a interpretação visual e reduzir erros humanos.

Soluções mais integrativas e acessíveis tornam-se cruciais para alcançar as metas estabelecidas pela OMS. Portanto, este trabalho propõe uma pesquisa de aquisição de imagens de motoristas, com o objetivo de identificar momentos de distração durante a condução através da implementação de Inteligência Artificial em uma aplicação móvel. A proposta inclui sinalizar períodos de distração que duram mais de 3 segundos, contribuindo, assim, para o desenvolvimento de abordagens práticas e eficazes na promoção da segurança viária.

1.2 Motivação

A motivação para esta dissertação surge diante do cenário alarmante em que vidas são perdidas no trânsito devido a erros humanos, dos quais a falta de atenção se destaca como um fator crucial. Tais mortes, em sua maioria, poderiam ser evitadas, destacando a importância de buscar soluções práticas e eficazes. Campanhas de conscientização em diversos países refletem que a segurança no trânsito é uma preocupação global e merece estudo contínuo [8]. Paralelamente, o avanço significativo da IA nos últimos anos ampliou consideravelmente seu poder de processamento e a capacidade de análise de dados em tempo real, abrindo portas para novas possibilidades no monitoramento e prevenção de comportamentos de risco.

A motivação principal deste trabalho, portanto, é salvar vidas, aproveitando o potencial da IA para detectar e alertar sobre momentos de distração ao volante. Com a atual atenção global voltada para a redução de fatalidades no trânsito e o suporte de novas tecnologias, este estudo se torna ideal para explorar como a IA pode contribuir de forma prática e acessível para a segurança no trânsito.

1.3 Objetivos

Os objetivos desta dissertação serão apresentados a seguir. A subseção 1.3.1 aborda o objetivo geral, enquanto a subseção 1.3.2 detalha os objetivos específicos.

1.3.1 Objetivo Geral

O objetivo geral deste projeto consiste em desenvolver uma solução baseada em Inteligência Artificial para identificar e sinalizar momentos de distração ao volante, com foco em aumentar a segurança no trânsito. Através da captura e análise de imagens dos motoristas, o sistema visa detectar períodos de desatenção que excedam uma duração crítica (3 segundos), fornecendo alertas em tempo real. Dessa forma, o projeto busca reduzir o número de acidentes causados por falhas humanas, contribuindo para a diminuição das

taxas de mortalidade e para a promoção de comportamentos mais seguros na condução.

1.3.2 Objetivos Específicos

Para atender ao objetivo geral, esta dissertação propõe uma solução que abrange os seguintes objetivos específicos:

- Estudar e implementar um modelo de rede neural utilizando TensorFlow.
- Adaptar o modelo para melhorar seu desempenho em dispositivos móveis.
- Adquirir e classificar imagens para compor um *dataset* focado em distrações veiculares.
- Treinar o modelo desenvolvido.
- Implementar um aplicativo móvel utilizando o *framework* Flutter.
- Integrar o modelo ao aplicativo.
- Coletar dados e realizar a análise e discussão dos resultados obtidos.

1.4 Estrutura do Documento

O Capítulo 2 aborda a fundamentação teórica, incluindo tópicos como redes neurais, TensorFlow, desenvolvimento *mobile*, técnicas de aquisição de imagem, pré-processamento de imagem, problemas de treinamento, banco de dados, Avaliação de Modelos Preditivos , além de uma revisão de trabalhos relacionados.

O Capítulo 3 detalha as principais metodologias desenvolvidas ao longo da dissertação.

O Capítulo 4 apresenta e discute os resultados obtidos, incluindo a avaliação dos modelos de *machine learning*, a análise de atenção e o protótipo desenvolvido.

Por fim, o Capítulo 5 expõe as conclusões do trabalho, além de recomendações e sugestões para pesquisas futuras.

Capítulo 2

Fundamentação Teórica

Nesta seção, são abordados os principais fundamentos teóricos do estudo, incluindo redes neurais e suas arquiteturas, os paradigmas de aprendizado, os desafios de treinamento, e a otimização do modelo para dispositivos móveis com TensorFlow Lite. Também são descritas técnicas de pré-processamento e métodos de avaliação de desempenho para apoiar o desenvolvimento do modelo proposto.

2.1 Redes Neurais

A história das Redes Neurais e IA começou formalmente na década de 1940, quando neurocientistas e matemáticos se inspiraram na estrutura e no funcionamento do cérebro para criar modelos matemáticos que pudessem simular o comportamento dos neurônios. Em 1943, Warren McCulloch e Walter Pitts desenvolveram o primeiro modelo teórico de um neurônio artificial, um sistema binário de ativação que representava a ideia de como os neurônios no cérebro processam sinais [13]. Esse modelo, considerado o primeiro passo em direção às redes neurais artificiais, foi a base para várias teorias subsequentes sobre aprendizado computacional e redes de neurônios.

Em 1958, Frank Rosenblatt desenvolveu o Perceptron, o primeiro modelo funcional de rede neural, que conseguia realizar classificações simples ao ajustar pesos para diferentes entradas de dados [14]. O perceptron funcionava bem em problemas lineares, mas suas

limitações foram expostas em 1969, quando Marvin Minsky e Seymour Papert mostraram que ele não era capaz de resolver problemas complexos de maneira eficaz, como o problema do XOR (exclusão lógica), um desafio essencial para a classificação não linear [15]. Esse trabalho desacelerou o desenvolvimento de redes neurais por mais de uma década, à medida que pesquisadores perderam a confiança nas redes neurais como uma abordagem promissora.

O campo experimentou uma revitalização na década de 1980 com a introdução da retropropagação do erro, um algoritmo que permite o ajuste dos pesos em camadas ocultas para melhorar o desempenho em problemas complexos [16]. Esse avanço, proposto por David Rumelhart, Geoffrey Hinton e Ronald Williams, permitiu a criação de redes neurais profundas, reavivando o interesse pela pesquisa em redes neurais. Na mesma época, surgiram outros avanços em algoritmos de aprendizado supervisionado e técnicas de regularização, que permitiram que as redes neurais se tornassem mais eficientes e generalizáveis [17].

Nos anos 2000 e 2010, as redes neurais experimentaram uma nova era de progresso, especialmente com o aumento da capacidade computacional e o desenvolvimento de algoritmos mais avançados, como as Redes Neurais Convolucionais (CNNs) para processamento de imagens e as Redes Neurais Recorrentes (RNNs) para dados sequenciais [18], [19]. As CNNs, popularizadas por Yann LeCun e outros, revolucionaram o campo de visão computacional, tornando-se uma das arquiteturas mais usadas em reconhecimento de imagens e detecção de objetos [18]. Na mesma linha, RNNs e seus aprimoramentos, como *LSTMs* (Long Short-Term Memory), foram cruciais para aplicações de processamento de linguagem natural e séries temporais [19].

Atualmente, com o avanço da capacidade computacional, especialmente devido às *GPUs* e *TPUs*, e a disponibilidade de grandes volumes de dados, as redes neurais evoluíram para redes neurais profundas ou *Deep Learning*, um campo que utiliza redes com várias camadas para extrair padrões complexos dos dados [20]. Esse progresso trouxe uma série de aplicações avançadas, como carros autônomos, diagnósticos médicos automatizados e sistemas de recomendação em plataformas de *streaming* [21].

2.1.1 Neurônios Biológicos e artificiais

O cérebro humano é constituído por bilhões de neurônios, cada um composto por três partes essenciais, cada qual desempenhando uma função específica. As dendrites captam estímulos elétricos do ambiente e os transmitem para o corpo do neurônio, onde ocorre o processamento desse estímulo. Posteriormente, o neurônio envia um novo estímulo que é propagado pelo axônio, transmitindo-o ao neurônio adjacente, fenômeno conhecido como sinapse. Esse processo pode se repetir através de várias camadas neuronais, possibilitando o processamento de informações pelo cérebro e a subsequente execução de ações físicas. Para uma representação simplificada das partes de um neurônio, é possível consultar a Figura 2.1. A capacidade de um indivíduo para realizar tarefas complexas e, principalmente, sua aptidão para aprender derivam do processamento simultâneo e distribuído da rede de neurônios no cérebro. O córtex, a camada externa do cérebro, desempenha um papel crucial no processamento cognitivo. A introdução de novos conhecimentos ou experiências pode resultar em modificações estruturais no cérebro, as quais são realizadas por meio do rearranjo das redes neurais, fortalecendo ou inibindo determinadas sinapses[22].

A concepção da reprodução artificial dos conceitos presentes nas redes neurais humanas tornou-se viável graças aos avanços tecnológicos recentes. As bases desse campo foram estabelecidas nas décadas de 40 e 50 por pioneiros como McCulloch e Pitts (1943), Hebb (1949) e Rosenblatt (1958). O primeiro modelo artificial de neurônio, conhecido como *perceptron* e ilustrado na Figura 2.2, surgiu como uma representação inicial do funcionamento de um neurônio real. Este modelo, que realiza uma soma ponderada das entradas e retorna 0 ou 1 como saída, teve uma utilidade limitada inicialmente. No entanto, desempenhou um papel crucial ao possibilitar a evolução para modelos mais complexos e diversas topologias de redes neurais. Atualmente, exploramos redes neurais com um número significativamente maior de neurônios artificiais, replicando de maneira mais fiel as características dos neurônios reais. Estas redes podem apresentar uma ou várias camadas de neurônios.

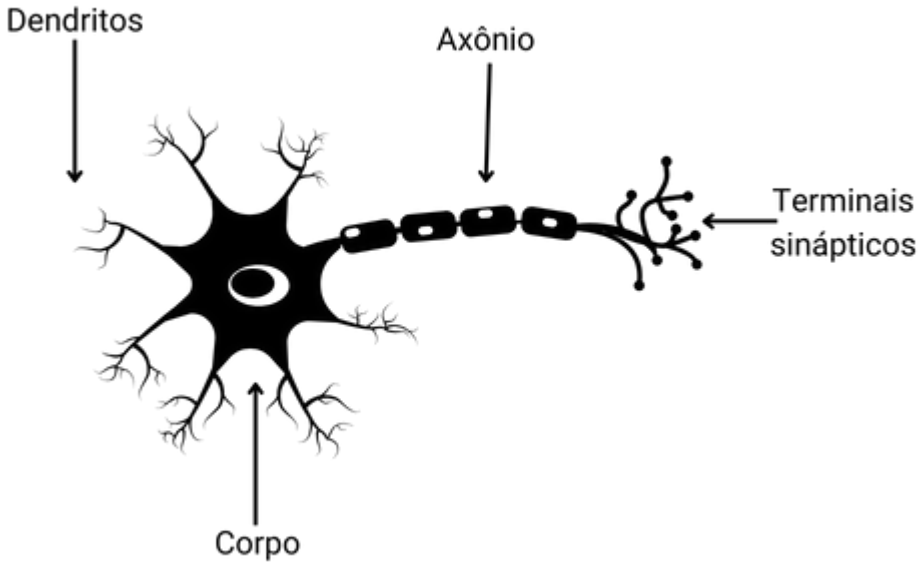

Figura 2.1: Ilustração simplificada de um neurônio

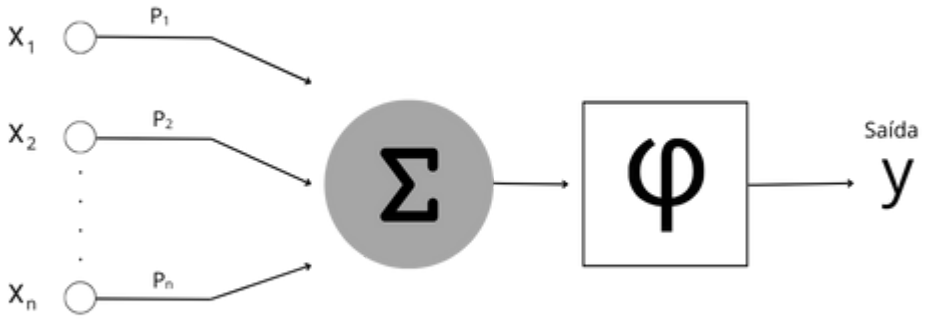

Figura 2.2: Ilustração simplificada de um neurônio artificial.
Fonte: Adaptado de Modelos Dinâmicos [23].

2.1.2 Redes neurais

Uma arquitetura de rede neural feedforward refere-se a um tipo específico de configuração em que a informação progride unidirecionalmente, representado na Figura 2.3 , fluindo da entrada para a saída, sem loops ou ciclos [22]. Nessa estrutura, os neurônios são interligados aos neurônios da camada subsequente por meio de pesos ajustáveis durante o treinamento da rede, visando aprimorar o desempenho do modelo. A topologia de uma rede neural feedforward é caracterizada pela disposição e quantidade de camadas, além do número de neurônios em cada camada. A complexidade dessas redes pode variar, desde arquiteturas simples com uma única camada oculta até modelos mais profundos com várias camadas ocultas. As redes neurais feedforward destacam-se em tarefas de classificação binária ou multiclasse, assim como em problemas de regressão nos quais a relação entre entradas e saídas é relativamente direta.

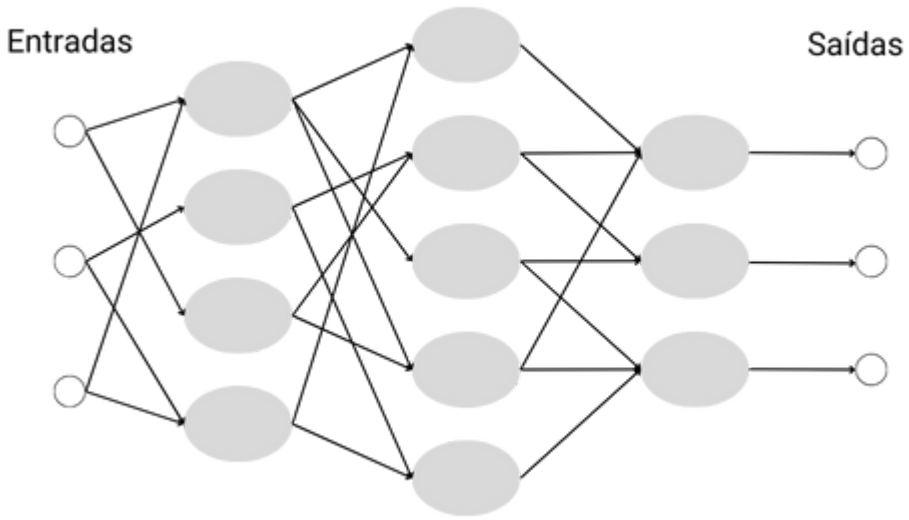

Figura 2.3: Ilustração simplificada de uma rede neural feedforward
Fonte: Adaptado de Modelos Dinâmicos [23].

As RNNs representam uma arquitetura especializada de redes neurais desenvolvida para lidar com dados sequenciais ou temporais, incorporando conexões retroativas que

permitem a retenção e processamento de informações ao longo do tempo como visto na Figura 2.4. A característica fundamental das RNNs reside na capacidade de manter uma "memória"interna ou estado oculto, que é atualizado em cada passo temporal, exercendo influência nas previsões futuras. O processo básico de uma RNN engloba a propagação para frente, onde a informação flui da entrada para a saída, e a retropropagação do erro, que ajusta os pesos da rede com base na discrepância entre as previsões e os rótulos reais [24]. Apesar de sua eficácia em lidar com dados sequenciais, as RNNs enfrentam desafios

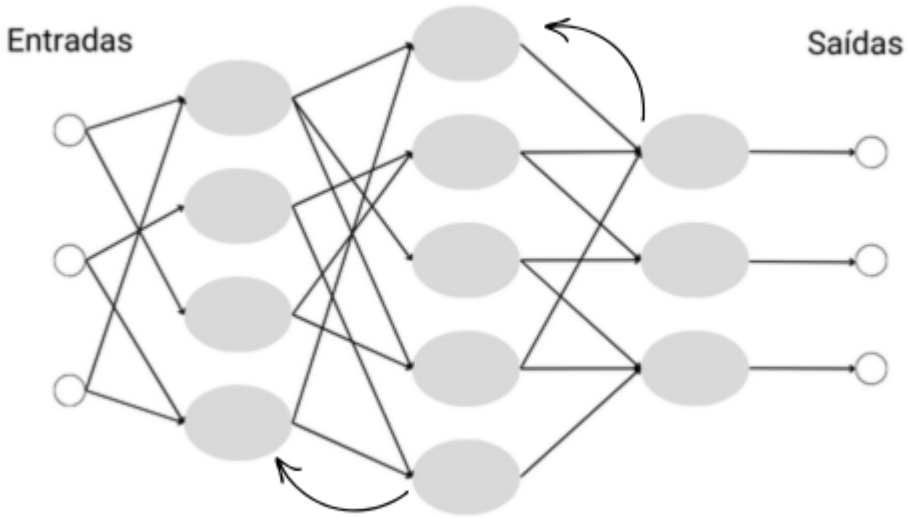

Figura 2.4: Ilustração simplificada de uma rede neural recorrente

relacionados à eficácia e eficiência computacional, especialmente quando comparadas a arquiteturas mais recentes, como as CNNs. Em particular, as RNNs encontram aplicabilidade significativa no campo do Processamento de Linguagem Natural (PLN), sendo amplamente empregadas em tarefas como tradução e geração de texto.

As CNNs são um tipo de rede neural especialmente eficaz para o processamento de dados com uma estrutura de grade, como imagens. Inspiradas pelo funcionamento do córtex visual animal, as CNNs têm como objetivo identificar automaticamente características importantes dentro de uma imagem, como bordas, formas e texturas, sem a necessidade

de intervenção humana para selecionar essas características [25]. Isso é possível devido à arquitetura das CNNs, que utiliza camadas convolucionais para extrair informações relevantes, visto na Figura 2.5. Cada camada convolucional aplica filtros que percorrem a imagem e produzem mapas de características, permitindo ao modelo identificar padrões locais e construir uma representação hierárquica da imagem [26].

O processo de treinamento das CNNs envolve a otimização dos pesos associados a cada filtro, utilizando algoritmos como o de retropropagação e funções de perda específicas, como a entropia cruzada para problemas de classificação [27]. Com o treinamento, o modelo aprende a ajustar esses filtros para maximizar sua capacidade de detectar padrões relevantes para a tarefa específica. Além das camadas convolucionais, as CNNs geralmente incluem camadas de pooling, que reduzem a dimensionalidade dos dados e ajudam a prevenir o sobreajuste, e camadas totalmente conectadas, onde ocorre a combinação final das características extraídas para realizar a classificação ou predição [28]. Essa estrutura torna as CNNs especialmente úteis em aplicações que requerem alta precisão na análise de imagens, como reconhecimento facial, diagnóstico médico e monitoramento de atenção ao volante [29].

O aprendizado supervisionado é uma abordagem de aprendizado de máquina em que o modelo é treinado com um conjunto de dados rotulados, onde cada exemplo de entrada está associado a uma saída conhecida [30]. Esse processo permite que o modelo aprenda a mapear as entradas para as saídas corretas, a fim de realizar previsões em novos dados de maneira precisa. Essa técnica é amplamente utilizada para resolver problemas de classificação e regressão, onde o objetivo é prever categorias ou valores contínuos, respectivamente [31].

Durante o treinamento supervisionado, o modelo ajusta seus parâmetros para minimizar a diferença entre suas previsões e os valores verdadeiros dos dados de treinamento. Este ajuste é feito por meio de funções de perda, que calculam o erro entre a saída prevista e a saída real e algoritmos de otimização, como o Gradiente Descendente, que minimizama função de perda ajustando os pesos do modelo [20]. Esse processo iterativo de treinamento permite que o modelo aprenda com os dados e melhore sua precisão ao longo do tempo,

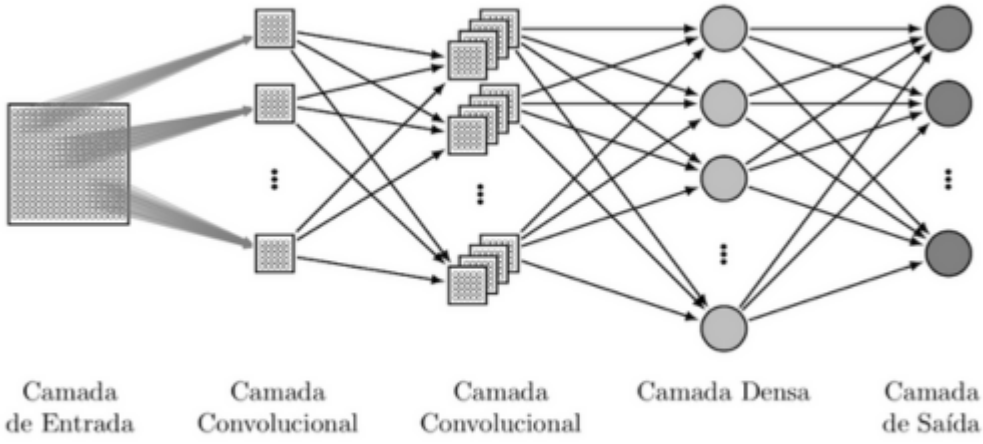

Figura 2.5: Ilustração simplificada de uma rede neural convolucional

aproximando-se das respostas corretas com cada iteração [32].

O aprendizado supervisionado é amplamente aplicado em diversas áreas, como em sistemas de detecção de fraudes, onde modelos de classificação identificam transações suspeitas com base em padrões aprendidos [33].

O aprendizado não supervisionado é uma abordagem em aprendizado de máquina onde o modelo trabalha com dados sem rótulos ou informações pré-definidas sobre as categorias [20]. Diferente do aprendizado supervisionado, em que o modelo aprende a associar entradas a saídas específicas, o aprendizado não supervisionado permite que o modelo explore e identifique padrões ou estruturas nos dados de maneira autônoma [30]. Esse tipo de aprendizado é particularmente útil em situações onde a categorização dos dados não é clara ou onde o volume de dados é grande demais para rotulagem manual [34].

Entre as técnicas mais comuns de aprendizado não supervisionado estão a análise de *clusters*, que agrupa dados semelhantes em categorias ou *clusters*, e a redução de dimensionalidade, que simplifica a representação dos dados mantendo suas características

principais [31]. A análise de *clusters*, por exemplo, inclui métodos como o *K-means* e o *clustering* hierárquico. O *K-means* particiona o conjunto de dados em um número definido de *clusters*, onde cada ponto é atribuído ao centroide mais próximo [35]. O algoritmo ajusta iterativamente a posição dos centroides com base na média dos pontos em cada *cluster*, até que as posições se estabilizem. Esse método é amplamente utilizado por sua simplicidade e eficiência, embora exija que o número de *clusters* seja definido previamente e seja sensível a *outliers* [36]. Já o *clustering* hierárquico organiza os dados em uma estrutura hierárquica, podendo ser aglomerativo, onde cada ponto de dado começa como um *cluster* individual que se combina progressivamente, ou divisivo, onde o conjunto começa como um único *cluster* e é sucessivamente dividido [37]. O resultado final é um dendrograma, que permite visualizar os diferentes níveis de agrupamento e as similaridades entre os dados.

A redução de dimensionalidade é outra técnica fundamental em aprendizado não supervisionado, que visa simplificar os dados mantendo as informações mais relevantes [38]. Uma das técnicas mais populares é a Análise de Componentes Principais(ACP), que transforma as variáveis originais em um novo conjunto de variáveis chamadas componentes principais. Essas componentes são combinações lineares das variáveis originais e são ordenadas de forma que as primeiras componentes capturam a maior variância dos dados, ou seja, a maior parte da informação útil [39], [40]. A redução de dimensionalidade com ACP não apenas facilita a visualização e interpretação dos dados, mas também melhora a eficiência computacional, especialmente em conjuntos de dados com muitas variáveis [41].

Essas técnicas de aprendizado não supervisionado são amplamente aplicadas em diversos contextos. Por exemplo, na segmentação de clientes em marketing, os *clusters* ajudam a identificar grupos de consumidores com comportamentos ou preferências semelhantes, permitindo a personalização de estratégias de marketing [42].

O modelo *MobileNet* é uma arquitetura de rede neural convolucional projetada para oferecer alta eficiência computacional e um bom equilíbrio entre precisão e rapidez de processamento. Desenvolvido pelo Google, o *MobileNet* foi criado especificamente para ser leve e otimizado para dispositivos com restrições de *hardware*, como celulares e outros

dispositivos móveis, onde os recursos de memória e processamento são limitados [43].

A principal inovação do *MobileNet* é o uso de convoluções separáveis em profundidade. Em vez de aplicar uma convolução tradicional a cada canal de entrada, o *MobileNet* realiza uma operação de convolução separada para cada canal, seguida por uma convolução ponto a ponto. Esse método reduz significativamente o número de parâmetros e operações, tornando o modelo mais eficiente e rápido em comparação com redes tradicionais [44].

O *MobileNet* também utiliza dois hiper parâmetros importantes: o fator de largura e o fator de resolução. O fator de largura permite ajustar a quantidade de filtros em cada camada, reduzindo ou aumentando a complexidade do modelo conforme necessário. O fator de resolução, por outro lado, ajusta a resolução das imagens de entrada, o que afeta a precisão e o tempo de processamento. Esses parâmetros permitem que o modelo *MobileNet* seja adaptável a diferentes necessidades de precisão e consumo de recursos [43].

Devido à sua arquitetura eficiente, o *MobileNet* é amplamente utilizado em aplicações de visão computacional em dispositivos móveis, como reconhecimento de imagem, detecção de objetos e até reconhecimento facial. Sua leveza e flexibilidade tornaram o modelo uma escolha popular para integração em sistemas de visão computacional onde a economia de recursos é essencial, sem comprometer significativamente o desempenho [45].

No processo de treinamento de redes neurais, alguns dos principais desafios são o *underfitting* e o *overfitting*. Esses problemas ocorrem quando a rede neural não é capaz de aprender adequadamente com os dados ou quando ela se ajusta excessivamente aos dados de treinamento, prejudicando sua capacidade de generalização em novos dados [20], [30].

Underfitting

O *underfitting* ocorre quando o modelo é excessivamente simplificado para capturar os padrões dos dados de treinamento, como visto na Figura 2.6, resultando em um desempenho insatisfatório tanto no conjunto de treinamento quanto no de teste. Esse problema geralmente surge quando o modelo não possui capacidade suficiente para representar a complexidade dos dados. Estratégias para reduzir o *underfitting* incluem o uso de modelos mais complexos e o ajuste de hiper parâmetros [31].

Overfitting

O *overfitting* acontece quando o modelo aprende em excesso os detalhes dos dados de treinamento, inclusive ruídos e padrões específicos, levando a uma perda de generalização assim como na Figura 2.6. Nesse caso, o modelo tem um bom desempenho no conjunto de treinamento, mas falha ao ser testado em novos dados. Técnicas como regularização, *dropout* e validação cruzada são comuns para mitigar o *overfitting* e melhorar a robustez do modelo em ambientes variados [20], [46].

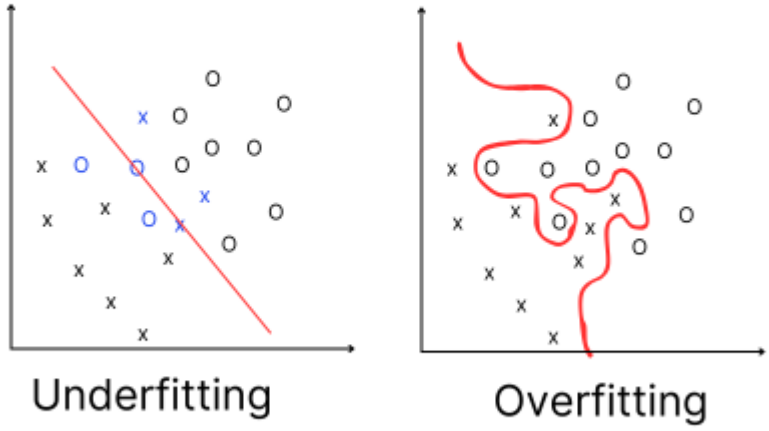

Figura 2.6: Comparação entre Underfitting e Overfitting

Uma rede neural é estruturada em camadas, onde cada camada desempenha uma função específica no processamento de dados e no aprendizado de padrões complexos. Essas camadas podem variar em termos de operação e finalidade, desde camadas convolucionais para a extração de características visuais até camadas totalmente conectadas para classificação. A arquitetura de redes neurais com várias camadas é chamada de rede neural profunda, onde cada camada contribui para a complexidade e a capacidade de aprendizado do modelo [20], [32].

Global Average Pooling

A camada de *Global Average Pooling* atua ao final da rede convolucional, reduzindo as dimensões da saída convolucional para um único valor por filtro, o que diminui a quantidade de parâmetros necessários e ajuda na generalização do modelo. Ela calcula a média de cada mapa de características, preservando a informação essencial para classificação e evitando o *overfitting* [47].

Dense Layer

A *Dense Layer* conecta todos os neurônios de uma camada à camada anterior, permitindo a combinação e transformação de características extraídas em camadas anteriores. A função de ativação *ReLU* é comumente usada em camadas densas para melhorar a eficiência computacional e permitir que a rede aprenda representações dos dados [48].

Dropout Layer

A *Dropout Layer* é uma técnica de regularização usada em redes neurais para reduzir o *overfitting*. Durante o treinamento, ela desativa aleatoriamente uma porcentagem dos neurônios (tipicamente 50%), o que impede que o modelo dependa excessivamente de neurônios específicos e promove uma maior capacidade de generalização para novos dados [46].

Final Dense Layer com Softmax

A camada final densa com ativação Softmax transforma os valores em uma distribuição de probabilidade para cada classe de saída. Isso é especialmente útil em tarefas de classificação, permitindo que o modelo produza uma probabilidade para cada categoria, facilitando a interpretação dos resultados [20].

Conv2D Layer

A *Conv2D Layer* é uma camada convolucional que aplica filtros para extrair características de uma imagem, como bordas, texturas e padrões. A quantidade de filtros (por exemplo, 64 filtros) determina o número de características que a camada irá aprender, aumentando a capacidade de reconhecimento de diferentes aspectos dos dados [21].

MaxPooling Layer

A MaxPooling Layer reduz a dimensionalidade dos mapas de características ao selecionar o valor máximo em cada região da matriz de características. Essa operação reduz o número de parâmetros e preserva as informações mais importantes, o que contribui para melhorar a eficiência computacional e a robustez do modelo [18].

Flatten Layer

A *Flatten Layer* converte a saída de uma matriz de características bidimensional em um vetor unidimensional. Esse passo é essencial para conectar as camadas convolucionais e de *pooling* às camadas densas, formando uma rede totalmente conectada que processa os dados para as operações de classificação [20].

2.2 TensorFlow

TensorFlow é uma das bibliotecas de aprendizado de máquina e aprendizado profundo mais amplamente usadas para pesquisa e desenvolvimento em Inteligência Artificial. Desenvolvido pela equipe do Google Brain em 2015, o TensorFlow foi projetado para oferecer uma plataforma aberta, flexível e eficiente para implementar modelos de aprendizado profundo, especialmente redes neurais artificiais [49]. A estrutura do TensorFlow foi criada para suportar desde a prototipagem de modelos de pesquisa até a produção em larga escala, o que permitiu sua rápida adoção pela comunidade de desenvolvedores e pesquisadores.

Uma das principais características do TensorFlow é o uso de tensores, estruturas de dados multidimensionais que permitem representar grandes volumes de dados e operações matemáticas complexas de forma eficiente. Cada operação no TensorFlow é representada como um nó em um gráfico computacional, permitindo a execução distribuída de cálculos e a otimização do fluxo de dados, tanto em *CPUs* quanto em *GPUs* [49]. Esse sistema de gráficos permite que o TensorFlow atinja alto desempenho, mesmo em cenários de treinamento massivo, o que é fundamental para o desenvolvimento de modelos de redes neurais profundas [50].

Além de suas capacidades para prototipagem, o TensorFlow também é amplamente utilizado para a implementação de sistemas de aprendizado profundo em produção, com suporte para dispositivos móveis e em navegadores via TensorFlow Lite e TensorFlow.js, respectivamente [51]. O TensorFlow Lite permite a execução de modelos em dispositivos com recursos limitados, como smartphones e dispositivos de *Internet* das Coisas (IoT), enquanto o TensorFlow.js possibilita a execução de modelos diretamente em navegadores, expandindo o alcance das aplicações de IA [52]. [53].

2.3 Aplicativo

Os *smartphones* tornaram-se uma presença ubíqua na sociedade contemporânea, com números impressionantes de adoção em todo o mundo. De acordo com dados da Statista, em 2024, o número de *smartphones* em uso atingiu a marca impressionante de 4,883 bilhões de unidades. Esta proliferação reflete a crescente dependência e importância desses dispositivos em nossas vidas diárias. Com uma projeção estimada pela Statista de alcançar 6,377 bilhões de aparelhos em uso até 2029, fica evidente que sua popularidade continua a crescer exponencialmente [54].

Com isso, cresce também o uso de aplicativos móveis como uma ferramenta indispensável na sociedade contemporânea, ampliando sua utilidade além do entretenimento para se tornarem instrumentos vitais na organização pessoal, gestão financeira, mobilidade urbana e até mesmo na interação com o setor público. Com sua capacidade de simplificar

tarefas complexas e facilitar o acesso a uma variedade de serviços, os aplicativos móveis estão transformando a maneira como vivemos e nos relacionamos com o mundo ao nosso redor.

Flutter é um *framework* de desenvolvimento de aplicativos móveis que tem ganho destaque pela sua eficiência e versatilidade, especialmente na criação de aplicativos para plataformas Apple e Android. Desenvolvido pelo Google, o Flutter oferece uma abordagem única de desenvolvimento de *UI* (Interface do Usuário), permitindo que os desenvolvedores criem aplicativos com uma única base de código que funcionam de maneira consistente em diferentes dispositivos e sistemas operacionais. Com seu conjunto abrangente de *widgets* personalizáveis e ferramentas de desenvolvimento, o Flutter simplifica o processo de criação de interfaces atraentes e responsivas. Além disso, o Flutter oferece excelente desempenho, graças à sua capacidade de renderização rápida e suave[55].

2.4 Técnicas de Aquisição de Imagens

Na seleção de tecnologias para captura de imagens dos condutores, é imprescindível avaliar uma variedade de opções, cada uma apresentando vantagens e desafios distintos. Enquanto as câmeras convencionais se destacam pela acessibilidade e simplicidade de implementação, podem enfrentar dificuldades em condições de baixa luminosidade. Por outro lado, as câmeras infravermelhas oferecem uma solução para esses cenários, embora com um custo potencialmente mais elevado. Os sensores de movimento e profundidade fornecem uma abordagem alternativa, capturando dados tridimensionais que enriquecem a compreensão do comportamento do condutor, embora sua integração possa ser mais complexa. Por fim, as câmeras integradas em sistemas de assistência à condução apresentam uma opção conveniente, aproveitando a infraestrutura existente nos veículos modernos. Contudo, a integração pode ser desafiadora devido à natureza proprietária do *hardware*, além de que a disponibilidade dessa tecnologia nos veículos em circulação pode ser limitada. A seleção entre essas tecnologias foi baseada na acessibilidade, visando alcançar o maior número de condutores possível, optando pelo uso da câmera convencional dos

smartphones.

2.4.1 Posicionamento e Angulação das Câmeras

É crucial considerar a melhor posição e ângulo de visão para assegurar a eficácia da monitorização. A região facial proporciona uma riqueza de informações sobre a atenção do usuário, com especial importância nos olhos, tornando imprescindível que a câmera abranja esses elementos. Levando em conta as restrições de suportes nos veículos e aproveitando soluções disponíveis no mercado, os suportes frontais que se integram ao espelho frontal à esquerda do motorista ou ao centro do painel são ideais. É crucial ressaltar que o dispositivo não deve obstruir a visão total ou parcial do condutor da via, a fim de evitar multas e garantir a segurança durante a condução.

2.4.2 Resolução e Taxa de Captura de Imagens

A resolução e a taxa de captura de imagens desempenham papéis críticos na eficácia dos sistemas de detecção dos estados de atenção do condutor. A qualidade de uma imagem digital é medida pela sua resolução, que influencia diretamente a definição dos detalhes visíveis e a fidelidade da reprodução de cores, brilho e contraste. Essa resolução é determinada pela quantidade de *pixels* presentes na imagem, sendo que quanto maior esse número, maior será a qualidade e nitidez da imagem resultante [56][57]. Em sistemas de detecção da atenção do condutor, uma alta resolução é essencial para capturar detalhes sutis que podem indicar estados de atenção ou distração. Por exemplo, microexpressões faciais ou movimentos oculares podem ser indicadores importantes, mas só podem ser identificados com precisão em imagens de alta resolução.

A taxa de captura de imagem, ou *frames* por segundo (FPS), refere-se à taxa na qual uma série de imagens é capturada. Isso influencia diretamente a fluidez da animação ou vídeo, determinando quão suave e realista ele parece aos olhos do espectador. Simplificando, FPS está relacionado à velocidade com que as imagens são geradas e apresentadas, afetando a experiência visual do usuário [58]. É crucial para capturar eventos em tempo

real de forma precisa. Taxas de captura mais altas permitem uma detecção mais sensível de mudanças na atenção do condutor, uma vez que proporcionam uma representação mais contínua e detalhada do comportamento visual.

2.4.3 Processamento de Imagens em Tempo Real

Isolates em Dart são especialmente úteis para tarefas de processamento intensivo, como o processamento de imagens em tempo real em aplicativos Flutter. Esse recurso permite que operações complexas, como captura de vídeo, detecção de objetos e reconhecimento de padrões, sejam realizadas sem bloquear a *thread* principal da interface do usuário [59]. Ao separar o processamento da imagens do *thread* principal, os *isolates* possibilitam que o aplicativo continue capturando e processando quadros de vídeo de forma assíncrona, garantindo que a interface permaneça responsiva e evitando travamentos durante a execução [60].

Além disso, os *isolates* permitem uma melhor utilização dos recursos de *hardware*, distribuindo o processamento de imagens entre várias *threads*. Isso resulta em uma análise mais rápida e eficiente de imagens em tempo real, aprimorando a experiência do usuário mesmo em tarefas de processamento intensivo [61]. Essa técnica é amplamente aplicada em aplicativos que exigem alta performance, como em sistemas de visão computacional e processamento de vídeo, onde a responsividade da interface é crucial para a interação do usuário [62].

2.5 Pré-processamento de Dados

A fase de pré-processamento de dados é fundamental para melhorar o desempenho do modelo, pois permite preparar e transformar dados brutos para garantir que estejam em um formato adequado para o treinamento. Esse processo não só aumenta a precisão e a velocidade do treinamento, como também melhora a capacidade do modelo de generalizar para novos dados [20], [63].

2.5.1 Redimensionamento de Imagens

O redimensionamento de imagens consiste em ajustar as dimensões das imagens para um tamanho uniforme, garantindo que todas as entradas tenham a mesma resolução ao entrar na rede neural, exemplificado na Figura 2.7. Isso é particularmente importante em CNNs, onde um tamanho padrão facilita o processamento em lotes e simplifica a arquitetura da rede. O redimensionamento reduz a complexidade computacional do modelo, otimiza o uso de memória e aumenta a velocidade do treinamento. Utilizar resoluções padronizadas, como 224x224 ou 256x256 pixels, também evita que o modelo precise lidar com variações de tamanho que poderiam prejudicar a consistência do aprendizado [64].

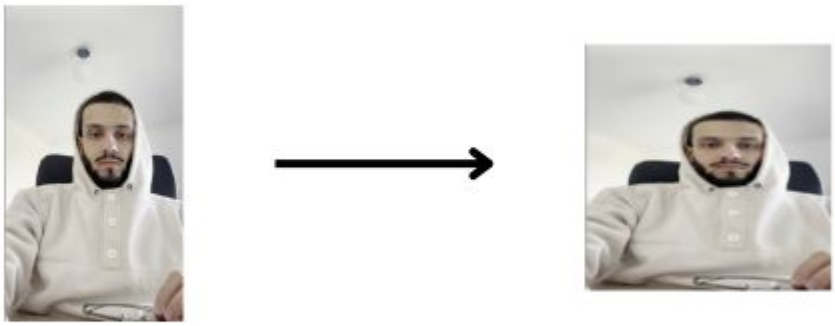

Figura 2.7: Imagem 1080x1920 px redimensionada para 250x250 px

2.5.2 Conversão para Arrays de Pixels

A transformação de imagens em *arrays* numéricos é um passo essencial em muitas aplicações de aprendizado de máquina e visão computacional. Essa conversão representa cada imagem como uma matriz onde cada valor corresponde à intensidade de um *pixel*, o

que torna possível o uso de operações matemáticas e facilita o processamento com bibliotecas de aprendizado profundo. Esse formato é essencial para redes convolucionais, pois permite que os filtros da rede identifiquem padrões e características importantes nas imagens de maneira eficiente. Converter as imagens para *arrays* numéricos também melhora o desempenho computacional, já que os dados são organizados para serem rapidamente manipulados por funções de otimização [65].

2.5.3 Normalização

A normalização é uma etapa essencial no pré-processamento de dados, ajustando os valores dos *pixels* para um intervalo específico, o que facilita a convergência do modelo durante o treinamento ao garantir que todos os dados estejam na mesma escala. Para normalizar as intensidades dos *pixels* no intervalo de $[-1, 1]$, utiliza-se a seguinte fórmula:

$$x_{\text{norm}} = \frac{x - 127.5}{127.5} \tag{2.1}$$

Neste caso, x representa a intensidade original de um pixel, enquanto x_{norm} corresponde ao valor normalizado. O valor 127.5 utilizado na fórmula é a média da escala de intensidade de *pixels* para imagens com valores de cor entre 0 e 255, o que centraliza os dados em torno de zero. Essa centralização e normalização ajudam a estabilizar o treinamento, pois ao ajustar os valores para um intervalo uniforme, minimiza-se o risco de instabilidades nas atualizações de gradiente, acelerando o processo de aprendizado do modelo [18], [66].

2.6 Dataset para Redes Neurais

A qualidade e a quantidade de dados de treinamento, ou *dataset*, são elementos cruciais para o sucesso de modelos de aprendizado profundo. O *dataset* fornece ao modelo informações suficientes para aprender os padrões, características e variabilidades dos dados, o que é fundamental para a generalização e precisão das previsões. Um bom *dataset* é composto por amostras representativas da tarefa a ser aprendida, balanceadas e diversificadas

o suficiente para cobrir diferentes cenários [20], [63].

Ao preparar um *dataset*, o processo de seleção e organização dos dados influencia diretamente o desempenho do modelo, especialmente quando a quantidade de dados é limitada. Técnicas como o aumento de dados artificial podem ser utilizadas para ampliar a quantidade de dados de treinamento sem a necessidade de coleta adicional [67].

O aumento de dados artificial é uma técnica utilizada para ampliar a quantidade de dados de treinamento a partir de dados já existentes. Em imagens, métodos comuns de aumento incluem a rotação de imagem, ajuste de brilho, espelhamento horizontal e corte. Essas transformações criam variações das imagens originais, ajudando o modelo a aprender a reconhecer os padrões sob diferentes condições e perspectivas [68].

Essas transformações aumentam a robustez e a capacidade de generalização do modelo, reduzindo o risco de *overfitting*. Técnicas de aumento de dados são particularmente úteis em áreas onde a coleta de novos dados é complexa ou dispendiosa [67].

2.7 Avaliação de Modelos Preditivos

A avaliação de modelos preditivos é uma etapa essencial no desenvolvimento de sistemas de aprendizado de máquina e redes neurais. Este processo permite medir o desempenho do modelo em relação aos dados de teste, oferecendo informações detalhadas sobre sua capacidade de generalização e precisão. Diversas métricas são empregadas para avaliar o desempenho dos modelos, sendo as principais a matriz de confusão, precisão, revocação e *F1-score.*[69], [70].

2.7.1 Matriz de Confusão

A matriz de confusão é uma ferramenta que organiza as previsões do modelo em quatro categorias: verdadeiros positivos, verdadeiros negativos, falsos positivos e falsos negativos, exemplificado na Tabela 2.1. Esta estrutura permite uma visão clara das previsões corretas e incorretas, possibilitando cálculos adicionais, como precisão e revocação, além de auxiliar na análise de erros específicos do modelo [71].

Tabela 2.1: Matriz de Confusão – Exemplo

		Predita	
		Falsa	Verdade
Real	Falsa	50	6
	Verdade	5	39

2.7.2 Precisão

A precisão é uma métrica que avalia a proporção de previsões corretas entre todas as previsões positivas realizadas pelo modelo. Ela é calculada como

$$\text{Precision} = \frac{TP}{TP + FP} \tag{2.2}$$

e é especialmente importante em contextos onde o custo de falsos positivos é elevado, como em diagnósticos médicos, nos quais um falso positivo pode levar a intervenções desnecessárias [72].

2.7.3 F1-Score

O F1-score é a média harmônica entre precisão e revocação, calculado como

$$\text{F1} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2.3}$$

Esta métrica é especialmente útil em *datasets* desbalanceados, onde há uma predominância de uma classe sobre a outra, pois oferece um equilíbrio entre a precisão e a revocação, sendo amplamente utilizada em classificações onde ambos os tipos de erros (falsos positivos e falsos negativos) são relevantes [73].

2.8 Estado da arte

Na literatura, diversos estudos propuseram soluções para mitigar o problema da fadiga durante a condução. Um estudo publicado em 2019 aborda essa questão propondo

um método eficaz para detectar o estado de fadiga por meio das características espaço-temporais dos olhos do motorista. Os autores tratam a detecção de fadiga como um problema de reconhecimento de sequência baseado em imagem e desenvolvem uma rede neural convolucional com unidades de memória de longo prazo (*LSTM*) para esse fim. A camada *LSTM* pode ser utilizada em conjunto com outras camadas para aumentar a robustez do modelo, possibilitando a construção de modelos *LSTM* que incluem camadas totalmente conectadas [74]. Inicialmente, é aplicado um *framework* em cascata de múltiplas tarefas para extrair a região dos olhos a partir de vídeos em infravermelho, seguido pela análise das características espaciais usando camadas convolucionais profundas e das relações entre quadros adjacentes por meio de unidades *LSTM* [75].

Outra pesquisa relevante focou-se na detecção da fadiga do motorista usando uma rede neural convolucional para análise de imagens faciais. O sistema desenvolvido captura imagens usando uma câmera infravermelha com comunicação *USB*, realiza a detecção e alinhamento facial, identifica a região dos olhos e classifica o estado de alerta com base na CNN. Esse método permitiu a detecção eficaz e em tempo real da fadiga do motorista, com uma taxa de detecção superior a 20 quadros por segundo, atendendo às demandas de detecção em tempo real. A integração da tecnologia *ARM* proporcionou um processamento de dados eficiente, enquanto o critério *PERCLOS* (Percentual de Fechamento dos Olhos) foi adotado como métrica principal para avaliar o estado de alerta do motorista, destacando a importância de uma abordagem abrangente e tecnologicamente avançada para prevenir acidentes de trânsito [76].

Empresas também estão ativamente engajadas em pesquisar e desenvolver tecnologias inovadoras para melhorar a segurança automotiva. Um exemplo notável é a iniciativa da Bosch, que emprega inteligência artificial e câmeras em um sistema avançado para monitoramento interno do veículo. Esse sistema utiliza uma rede de câmeras estrategicamente posicionadas, incluindo uma integrada ao volante e outras próximas ao retrovisor, para capturar e analisar continuamente o comportamento do motorista. A inteligência artificial processa essas informações em tempo real, identificando sinais de distração, fadiga ou comportamentos potencialmente perigosos. Com base nessa análise, o sistema pode

realizar ações proativas, como alertar o motorista, recomendar uma pausa ou até ajustar automaticamente a velocidade do veículo [77].

Um estudo de 2018 propôs um sistema de detecção de sonolência em tempo real para dispositivos Android, empregando técnicas de redes neurais profundas. O modelo desenvolvido foi capaz de alcançar uma precisão superior a 80%, demonstrando a viabilidade de implementar sistemas de monitoramento de sonolência em plataformas móveis, o que facilita a ampla adoção dessa tecnologia sem a necessidade de hardware especializado [78].

Em 2019, outra pesquisa apresentou um método para monitoramento em tempo real da sonolência do motorista em plataformas móveis, utilizando redes neurais 3D. A abordagem combinou convoluções 3D separáveis em profundidade com uma fusão precoce de informações espaciais e temporais, equilibrando alta precisão de predição com os requisitos de inferência em tempo real. O estudo também desenvolveu um aplicativo para smartphone baseado em TensorFlow, demonstrando a eficácia do monitoramento em tempo real para alertar motoristas sonolentos [79].

Em 2020, foi desenvolvido um modelo de detecção de sonolência do motorista utilizando técnicas de redes neurais convolucionais para aplicativos Android. O modelo utilizou marcos faciais detectados pela câmera, que foram processados por uma rede neural convolucional para classificar o estado de sonolência. A solução proposta apresentou uma precisão média superior a 83% em todas as categorias testadas, com um tamanho máximo de modelo de 75 KB, tornando-o adequado para implementação em sistemas embarcados e dispositivos Android [80].

No contexto brasileiro, um trabalho de conclusão de curso desenvolvido na Universidade Federal de Santa Catarina em 2023 focou na detecção de distração e sonolência em condutores veiculares utilizando redes neurais convolucionais. O sistema proposto empregou uma câmera convencional para capturar imagens faciais do condutor, aplicando algoritmos de visão computacional e redes neurais convolucionais para identificar padrões de distração e sonolência. Os resultados preliminares indicaram uma precisão de 95% e 84% em testes distintos de identificação de padrões de sonolência em condutores [81].

Capítulo 3

Metodologia

Neste Capítulo 3, serão descritos os materiais e métodos desenvolvidos para esta dissertação, incluindo uma apresentação inicial das Ferramentas Utilizadas, seguida pela Metodologia para Aquisição de Imagens, criação do dataset, pré-processamento de imagens, desenvolvimento do modelo de rede neural, criação do aplicativo móvel e a metodologia de testes.

3.1 Ferramentas Utilizadas

Nesta seção serão mostradas as ferramentas utilizadas para a construção e implementação dessa dissertação.

3.1.1 Computador

O computador utilizado como plataforma de desenvolvimento e teste neste projeto foi um MacBook Air APPLE, equipado com o processador Apple M2 de 8 núcleos, 8 GB de memória RAM e 256 GB de armazenamento SSD, além de uma GPU de 8 núcleos integrada.

3.1.2 Celulares

Para a captura das imagens, foram utilizados smartphones Motorola Moto E40 e XIAOMI Redmi Note 10, ambos configurados para gravação de vídeo com a câmera frontal em resolução de 720p.

3.1.3 Google Colab

Para o treinamento e testes do modelo, foi utilizado o Google Colab , uma plataforma baseada na nuvem que permite a execução de códigos Python diretamente no navegador, com acesso a recursos de computação poderosos, incluindo *GPUs* e *TPUs*. O Colab oferece um ambiente pré-configurado com as bibliotecas necessárias para aprendizado de máquina, como TensorFlow, facilitando o desenvolvimento e o treinamento de redes neurais.

3.1.4 Python - Linguagem de programação

Para o desenvolvimento e execução dos *scripts* deste projeto, utilizou-se a linguagem Python, devido à sua versatilidade e ampla adoção no campo de aprendizado de máquina e processamento de dados. Python possui uma vasta gama de bibliotecas específicas para aprendizado profundo e manipulação de dados, como TensorFlow e NumPy que foram essenciais para a construção, treinamento e análise dos modelos.

3.1.5 TensorFlow - Biblioteca de programação

Para o desenvolvimento e treinamento dos modelos, foi utilizada a TensorFlow, uma biblioteca de aprendizado profundo que permite criar e treinar redes neurais com eficiência. O TensorFlow facilita a manipulação de grandes volumes de dados e o processamento de operações matemáticas complexas, o que é essencial para redes neurais convolucionais e outras arquiteturas avançadas.

3.1.6 Numpy - Biblioteca de Programação

Neste projeto, a biblioteca NumPy foi utilizada para manipulação e processamento eficiente de dados numéricos, especialmente para a criação e operação de arrays multidimensionais. NumPy permite a transformação de imagens em matrizes de *pixels*, facilitando o pré-processamento e a entrada dos dados nas redes neurais. Suas funções vetorizadas e a capacidade de realizar operações matemáticas complexas com rapidez são essenciais para otimizar o tempo de execução dos *scripts* e preparar os dados para análise e modelagem no TensorFlow.

3.1.7 Flutter - Framework

Para a criação da aplicação móvel neste projeto, foi utilizado o Flutter, um *framework* de desenvolvimento multiplataforma desenvolvido pelo o Google. Flutter permite a construção de interfaces de usuário (UI) de alta performance para Android e iOS a partir de um único código base, utilizando a linguagem de programação Dart. Esta escolha foi motivada pela necessidade de desenvolver uma solução prática e responsiva, capaz de executar tarefas de detecção e classificação de imagens em tempo real.

3.2 Metodologia para Aquisição de Imagem

Nesta seção, são detalhados os métodos utilizados para a aquisição das imagens que compõem o *dataset*, fundamental para o treinamento e avaliação dos modelos de aprendizado de máquina.

3.2.1 Posicionamento

O posicionamento dos dispositivos para a aquisição de imagem foi realizado no painel de um Volkswagen Gol 2023, utilizando os engates de celular do próprio veículo, como visto na Figura 3.1. Essa escolha de posicionamento foi feita para garantir um ângulo de visão estável e direto da área do motorista, permitindo capturar detalhes faciais e

corporais relevantes para o modelo. O suporte no painel oferece uma fixação segura para o *smartphone*, mantendo a estabilidade necessária durante a gravação de vídeos, mesmo com possíveis movimentações do veículo.

Figura 3.1: Suporte para fixação de celular

3.2.2 Definição dos Trajetos de Captura

Para garantir um *dataset* diversificado e representativo, foram definidos dois trajetos distintos, um em ambiente rodoviário e outro em ambiente urbano.

Trajeto Rodoviário

O primeiro trajeto foi realizado em uma estrada rodoviária, proporcionando um ambiente de direção contínua e com menor quantidade de interrupções, caracterizado por velocidades moderadas a altas. Esse cenário permite capturar dados sobre o comportamento do motorista em condições de maior estabilidade e atenção prolongada, representativas de trajetos mais longos e típicos de rodovias. Esse trajeto é ilustrado na Figura 3.2.

Figura 3.2: Trajetória entre Caarapó á Água boa no MS

Trajeto Urbano

O segundo trajeto foi realizado em um ambiente urbano, marcado por maior densidade de tráfego, cruzamentos e paradas frequentes em semáforos. Esse percurso foi selecionado para capturar dados em um contexto dinâmico, com constantes acelerações e frenagens, típico da condução em áreas residenciais e comerciais. A direção urbana oferece dados que refletem mudanças rápidas de atenção e a necessidade de resposta a múltiplos estímulos. Esse trajeto é representado na Figura 3.3.

3.2.3 Recorte

Os vídeos capturados, com uma duração média de 30 minutos cada, foram extraídos dos dispositivos mencionados e processados para a criação do *dataset* de imagens. Durante o processamento, cada vídeo foi recortado em intervalos de 5 segundos, resultando em *frames* representativos das condições ao longo do trajeto. Essa taxa de amostragem permitiu uma captura eficiente de dados, equilibrando o volume de imagens coletadas

Figura 3.3: Trajetória dentro da cidade de São Paulo

com a necessidade de reduzir redundâncias

3.3 Criação de Dataset

Nesta seção, abordam-se os processos de criação do *dataset* utilizado para o treinamento e validação do modelo, incluindo as etapas de classificação das imagens e aumento de dados.

3.3.1 Classificação das Imagens

Como não foram encontrados datasets prontos que atendessem à necessidade específica deste trabalho, foi criado um novo conjunto de dados. A classificação das imagens do dataset foi realizada manualmente, com o objetivo de assegurar a precisão e consistência dos rótulos aplicados para o treinamento e validação do modelo. As imagens foram classificadas em dois estados principais de atenção do motorista, representados por uma

estrutura binária: 'atenção' (Figura **??**) e 'falta de atenção' (Figura **??**). A codificação adotada utilizou '1' para o estado de atenção e '0' para indicar a falta de atenção.

Para a rotulagem das imagens na classe "atenção" (indicada pelo valor "1"), foram seguidos critérios específicos, incluindo:

- **Mãos no Volante**: As imagens foram rotuladas como "atenção" apenas quando ambas as mãos do motorista estavam visivelmente posicionadas no volante, indicando uma postura de controle e foco na condução.
- **Olhos Direcionados para a Pista**: A classificação na categoria "atenção" exigiu que o motorista mantivesse os olhos voltados para a estrada, sem desviar o olhar para dispositivos eletrônicos, objetos laterais ou elementos que pudessem indicar distração.
- **Acessórios Permitidos**: Somente imagens em que a visão dos olhos do motorista estava livre de bloqueios por itens que não interferem no campo de visão, como óculos, foram classificadas como "atenção". A presença de objetos como celulares ou garrafas de água bloqueando os olhos desqualificava a imagem da categoria "atenção".

As imagens classificadas como "falta de atenção" (indicadas pelo valor "0") incluíam situações em que esses critérios não eram atendidos, refletindo condições de distração do motorista.

3.3.2 Aumento de Dados

Para expandir o banco de imagens e aumentar a diversidade de amostras no dataset, as imagens foram replicadas com modificações em brilho, contraste, zoom e rotação. Essas transformações foram aplicadas com o objetivo de melhorar a robustez e a capacidade de generalização do modelo, permitindo que ele seja capaz de reconhecer padrões sob diferentes condições visuais.

As modificações realizadas incluíram:

Figura 3.4: Exemplo de imagens que caracterizam atenção. (Elaboração própria)

Figura 3.5: Exemplo de imagens que caracterizam falta de atenção. (Elaboração própria)

- **Ajuste de Brilho e Contraste**: Foram aplicados coeficientes de multiplicação de brilho e contraste com valores de 0.5, 1.5 e 5. Esse ajuste simula diferentes condições de iluminação, como variações de luz natural e artificial, aumentando a capacidade do modelo de processar imagens capturadas em diferentes contextos

visuais, resultado visto na Figura 3.6.

Figura 3.6: Imagem original e imagem com redução de brilho em 50%

- **Zoom**: Foi aplicada uma variação de zoom de 0.75, Figura 3.7, permitindo que o modelo analise imagens com diferentes proporções e proximidade de foco, o que auxilia na identificação de características em diferentes escalas.

- **Rotação**: Para aumentar a variabilidade das perspectivas, foram aplicadas rotações de 10 graus à direita e 8 graus à esquerda. Essas rotações permitem que o modelo reconheça características faciais e corporais, mesmo quando o motorista está levemente inclinado ou em ângulos não frontais, representado na Figura 3.8.

3.3.3 Ética

Para assegurar a conformidade ética e o respeito à privacidade dos participantes, todos os indivíduos envolvidos nos vídeos concederam permissão explícita para o uso de suas imagens neste projeto. O consentimento foi obtido de forma clara, garantindo que

Figura 3.7: Imagem original e imagem com zoom 75%

Figura 3.8: Imagem original e imagem com rotação de 8 graus a direita

cada participante estivesse ciente do objetivo e do uso específico das imagens coletadas. Esse cuidado com a privacidade dos dados visuais visa cumprir com os padrões éticos de pesquisa e proteger a integridade e os direitos dos envolvidos.

3.4 Pré-processamento de Imagens

A fase de pré-processamento é essencial para otimizar o desempenho do modelo durante o treinamento, reduzindo o tempo necessário e melhorando sua precisão. Em projetos de aprendizado profundo, o pré-processamento garante que as imagens estejam em um formato e escala consistentes, o que facilita o aprendizado da rede neural ao remover variações indesejadas e padronizar os dados de entrada. Considerando que o hardware utilizado para a execução final do modelo é um celular, qualquer otimização adicional é vantajosa para o desempenho e a velocidade de execução. As principais etapas de pré-processamento aplicadas neste estudo incluem o redimensionamento das imagens, conversão para arrays de pixels e normalização.

3.4.1 Redimensionamento

Inicialmente, as imagens foram carregadas e redimensionadas para um tamanho padrão de 224x224 pixels como na Figura 3.9. Esse redimensionamento é necessário para garantir que todas as entradas tenham dimensões compatíveis com o modelo neural em uso, neste caso, o MobileNet, que foi projetado para processar imagens de entrada com essas dimensões. Padronizar o tamanho das imagens ajuda a reduzir a variabilidade nos dados, permitindo que o modelo aprenda a partir de entradas homogêneas e otimizando o uso de memória e processamento.

3.4.2 Conversão para Arrays de Pixels

Após o redimensionamento, as imagens são convertidas em *arrays* de *pixels*, utilizando a biblioteca NumPy. Cada imagem é transformada em um *array* NumPy, que representa

Figura 3.9: Imagem original e imagem redimensionada 250 x 250 px

uma matriz de valores de *pixels*, permitindo que a imagem seja processada pelo TensorFlow. Essa etapa é fundamental, pois traduz a imagem em um formato que a rede neural consegue interpretar, facilitando operações matemáticas subsequentes e o fluxo de dados para o treinamento do modelo.

3.4.3 Normalização

A última etapa do pré-processamento envolve a normalização dos valores dos *pixels*, ajustando-os para o intervalo esperado pelo modelo MobileNet. Este processo é realizado subtraindo a média das intensidades dos *pixels* (127.5) e dividindo pelo desvio padrão (127.5), resultando em valores de *pixels* no intervalo de -1 a 1. A normalização assegura que os dados de entrada estejam em uma escala uniforme, o que facilita a convergência do modelo durante o treinamento e melhora a estabilidade numérica.

3.5 Criação de Modelo de Rede Neural

Esta seção descreve o processo de criação do modelo de rede neural, detalhando a seleção das camadas, o treinamento e a conversão do modelo para o formato otimizado para dispositivos móveis.

3.5.1 Seleção de Layers

A seleção das camadas da rede neural baseou-se na arquitetura *MobileNet*, escolhida pela sua eficiência computacional e aplicabilidade em dispositivos móveis devido à sua estrutura leve e de baixo custo computacional. A *MobileNet* utiliza uma série de camadas convolucionais otimizadas, como convoluções separáveis em profundidade, que permitem uma extração eficiente de características enquanto reduzem o número de parâmetros do modelo.

Para adaptar a *MobileNet* à tarefa específica deste projeto, a arquitetura foi ajustada com a adição das seguintes camadas, além da seleção de dois tipos distintos de configuração de camadas para comparação, a fim de avaliar o impacto dessas variações no desempenho do modelo:

- **Camada de Pooling Global**: Uma camada de *global average pooling* foi adicionada em uma das configurações para reduzir a dimensionalidade e sintetizar as características globais da imagem antes de passá-las para as camadas densas.
- **Camada Convolucional e MaxPooling**: Na segunda configuração, foi incluída uma camada *Conv2D* com 64 filtros, seguida de uma camada de *MaxPooling*, para permitir uma extração mais detalhada de características espaciais, aumentando a precisão do modelo em cenários complexos.
- **Camada Densa com Ativação ReLU**: Uma camada densa com a função de ativação $ReLU$ foi incluída em ambas as configurações para introduzir não linearidade no modelo, permitindo o aprendizado de relações mais complexas entre as características das imagens [82].

- **Camada de Dropout**: Para reduzir o *overfitting*, uma camada de *dropout* foi adicionada em ambas as configurações, desativando aleatoriamente uma fração das unidades durante o treinamento, o que melhora a capacidade de generalização do modelo.

- **Camada Densa Final com Softmax**: Uma camada densa final com ativação *softmax* foi incorporada para gerar previsões probabilísticas das classes, permitindo a classificação em múltiplas categorias.

3.5.2 Treinamento

O treinamento do modelo foi realizado com o uso do otimizador Adam, que adapta dinamicamente as taxas de aprendizado durante o processo, e com a função de perda *categorical crossentropy*, ideal para tarefas de classificação multi classes [83]. Durante o treinamento, os pesos das camadas pré-treinadas da *MobileNet* foram mantidos congelados, preservando os padrões aprendidos no *dataset ImageNet* e evitando que os pesos fossem atualizados, o que ajuda a manter a estabilidade do modelo.

O treinamento foi executado em um ambiente de computação que inclui suporte a *GPUs*, Google colab, permitindo um processamento mais rápido e eficiente. Diversas configurações de hiper parâmetros foram testadas para otimizar o desempenho do modelo, incluindo ajustes no tamanho dos lotes e no número de épocas, por fim utilizou-se 50 lotes e 20 épocas.

3.5.3 Conversão

Para permitir a implementação do modelo em dispositivos móveis, foi realizada a conversão para o formato TensorFlow Lite. Esta conversão foi acompanhada da aplicação da técnica de quantização inteira, que reduz o tamanho do modelo e permite que ele opere com operações inteiras, otimizando o tempo de inferência. A quantização foi configurada para representar entradas e saídas no formato *uint8*, otimizando o modelo para dispositivos de baixo consumo energético, como *smartphones*, sem perda significativa de precisão [84].

3.6 Desenvolvimento de Aplicativo

Nesta seção, são descritos os processos envolvidos no desenvolvimento do aplicativo móvel, incluindo o design da interface, a implementação em Flutter, e a integração do modelo de rede neural ao sistema.

3.6.1 Design da Interface

O aplicativo foi projetado para oferecer uma interface intuitiva e de fácil uso, como ilustrado na Figura 3.10. A interface permite que o usuário realize previsões tanto em fotos pré-carregadas da galeria quanto em tempo real, utilizando a câmera do dispositivo. Para maximizar a acessibilidade, o layout foi organizado de forma limpa e funcional, com opções claras de navegação.

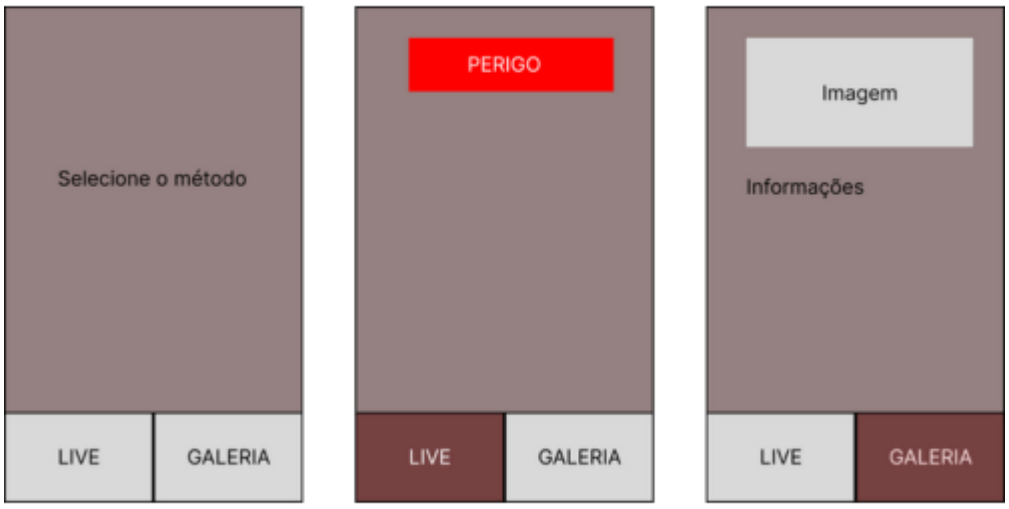

Figura 3.10: Wireframes de aplicativo

3.6.2 Desenvolvimento em Flutter

O aplicativo foi desenvolvido utilizando o Flutter na versão 3.16 da linguagem Dart. O Flutter é um *framework* multiplataforma que permite a criação de aplicativos nativos para Android e iOS a partir de um único código-base. A escolha do Flutter foi motivada pela flexibilidade e pela capacidade de construir interfaces modernas e responsivas que se adaptam bem a diferentes dispositivos.

Para o processamento do modelo de rede neural, foi utilizado o pacote tflite_flutter, que permite a integração e execução de modelos TensorFlow Lite diretamente no aplicativo. Além disso, visando otimizar o desempenho do aplicativo, o processamento da câmera e a coleta de imagens para a rede neural foram isolados utilizando o pacote nativo *Isolate* do Dart. O uso de *Isolates* permite que o aplicativo realize o processamento intensivo em segundo plano, sem comprometer a responsividade da interface, garantindo uma experiência de usuário fluida mesmo em operações de inferência em tempo real.

3.6.3 Integração de Modelo ao Aplicativo

Para integrar o modelo de rede neural ao aplicativo, o modelo foi convertido para o formato TensorFlow Lite e carregado utilizando o pacote tflite_flutter. O sistema permite a entrada de imagens tanto do armazenamento interno quanto em tempo real pela câmera do dispositivo. As imagens são divididas em quadros e redimensionadas para corresponder às dimensões de entrada do modelo, assegurando que estejam no formato correto para o processamento.

No modo de visualização em tempo real, o aplicativo monitora o comportamento do usuário e ativa um alerta auditivo sempre que uma situação de desatenção persiste por mais de 3 segundos.

3.7 Metodologia para Testes

A metodologia de testes foi conduzida com o objetivo de avaliar a eficácia e a eficiência do modelo de rede neural implementado. Todos os testes foram realizados com uma série de 10 imagens selecionadas, permitindo a obtenção de resultados representativos e consistentes. As métricas principais utilizadas para a avaliação foram: matriz de confusão, tempo de predição e porcentagem de certeza da predição. A seguir, detalha-se o procedimento de cada teste, especificando o processo e as condições em que foram executados.

3.7.1 Matriz de Confusão

Para avaliar o desempenho do modelo na classificação correta das imagens, foi construída uma matriz de confusão com base nos resultados das predições realizadas sobre a série de 10 imagens de teste. Este conjunto de teste foi formado por imagens previamente rotuladas e separadas do *dataset* principal, garantindo que os dados não fossem vistos pelo modelo durante o treinamento.

Cada imagem de teste foi processada pelo modelo, e os resultados das predições foram comparados com os rótulos reais das imagens. Com isso, foi possível calcular os valores de verdadeiros positivos, verdadeiros negativos, falsos positivos e falsos negativos para cada classe, que neste caso se dividem entre “atenção” e “falta de atenção”.

3.7.2 Tempo de Predição

O tempo de predição foi medido para avaliar a velocidade dos modelos ao realizar inferências sobre a série de 10 imagens. Esse teste foi realizado em um ambiente de teste específico e controlado, sem considerar a implementação no aplicativo, devido à variação significativa de desempenho entre diferentes dispositivos móveis, cujas configurações de hardware podem impactar o tempo de resposta.

Cada imagem da série foi processada pelo modelo, e o tempo necessário para realizar cada predição foi registrado. Com esses valores, calculou-se o tempo médio de predição,

permitindo uma análise precisa do desempenho do modelo e auxiliando na escolha da versão mais adequada para aplicações em tempo real.

3.7.3 Porcentagem de Certeza da Predição

A porcentagem de certeza da predição foi registrada para cada uma das 10 imagens processadas. Essa métrica indica o nível de confiança do modelo na classificação atribuída e é derivada dos valores probabilísticos fornecidos pela função *softmax* na última camada do modelo.

Capítulo 4

Resultados e Discussões

Nesta seção, são apresentados os resultados das comparações entre os modelos de rede neural nos formatos Lite (.tflite) e padrão (.h5), além de uma análise das diferentes configurações de camadas para os modelos Lite. São discutidos os dados coletados para cada formato e realizadas comparações entre configurações de camadas no modelo Lite, incluindo a precisão de cada modelo, ilustrada por uma matriz de confusão.

4.1 Comparação entre Modelos .H5 e .tflite

Na comparação inicial entre os modelos .H5 e Lite, com dados da Tabela 4.1, foi realizada uma avaliação dos valores de certeza, tempo de execução e tamanho do modelo. Os resultados indicam que o modelo .tflite teve uma média de certeza de 73,05%, enquanto o modelo H5 padrão alcançou uma média de 81,49%. Essa diferença de 8,44% em certeza reflete a perda de precisão ocasionada pela quantização para uint8.

Apesar da queda na certeza, o modelo .tflite apresentou um tempo médio de execução 11,60 vezes mais rápido que o modelo padrão (63,20 ms versus 733,37 ms), uma diferença significativa que favorece sua aplicação em dispositivos móveis. Além disso, o modelo Lite ocupa menos espaço, com uma redução de 14,64 MB em relação ao modelo H5, tornando-o mais eficiente para uso em sistemas com restrições de memória.

As tabelas 4.2 e 4.3, que apresentam as matrizes de confusão para os modelos h5 e

Tabela 4.1: Resultados Comparativos dos Modelos Quantizado Lite e Padrão H5

Modelo Quantizado Lite					
Imagem	**Certeza (%)**	**Tempo (ms)**	**Tamanho do Modelo (MB)**	**Classificação**	**Real**
Imagem 1	98.83	80.93	3.86	Distraído	Distraído
Imagem 2	58.20	28.74	3.86	Atenção	Atenção
Imagem 3	69.92	27.82	3.86	Atenção	Atenção
Imagem 4	94.53	76.30	3.86	Distraído	Distraído
Imagem 5	92.97	45.85	3.86	Distraído	Distraído
Imagem 6	99.98	124.68	3.86	Distraído	Distraído
Imagem 7	73.05	194.49	3.86	Atenção	Atenção
Imagem 8	58.20	93.79	3.86	Atenção	Atenção
Imagem 9	73.05	50.10	3.86	Atenção	Atenção
Imagem 10	73.05	43.57	3.86	Distraído	Distraído
Média	73.05	63.20	-	-	-
Modelo Padrão H5					
Imagem	**Certeza (%)**	**Tempo (ms)**	**Tamanho do Modelo (MB)**	**Classificação**	**Real**
Imagem 1	87.35	739.45	18.50	Distraído	Distraído
Imagem 2	88.98	664.80	18.50	Atenção	Atenção
Imagem 3	84.98	733.08	18.50	Atenção	Atenção
Imagem 4	93.24	1105.48	18.50	Distraído	Distraído
Imagem 5	54.68	1031.24	18.50	Distraído	Distraído
Imagem 6	58.10	733.66	18.50	Distraído	Distraído
Imagem 7	92.02	835.63	18.50	Atenção	Atenção
Imagem 8	88.96	691.14	18.50	Atenção	Atenção
Imagem 9	89.97	697.92	18.50	Atenção	Atenção
Imagem 10	76.61	646.00	18.50	Distraído	Distraído
Média	81.49	733.37	-	-	-

tflite quantizado, fornecem uma análise detalhada da precisão de ambos os modelos nas classificações. Dos 10 exemplos testados, não houve falsos negativos nem falsos positivos em nenhum dos modelos, indicando que ambos foram capazes de classificar corretamente todas as amostras, com 5 verdadeiros positivos e 5 verdadeiros negativos cada. Esses resultados sugerem que os modelos H5 e Lite quantizado mantêm uma taxa de acerto elevada, tornando-os adequados para contextos de resposta rápida em dispositivos móveis.

Tabela 4.2: Matriz de Confusão – Modelo .H5

		Predita	
		Falsa	Verdade
Real	Falsa	5	0
	Verdade	0	5

Tabela 4.3: Matriz de Confusão - Modelo .tflite

		Predita	
		Falsa	Verdade
Real	Falsa	5	0
	Verdade	0	5

4.2 Comparação entre diferentes camadas

Os resultados da comparação entre as duas configurações de camadas dos modelos Lite, como exibido na Tabela 4.4 , mostram diferenças importantes em termos de precisão, tempo de inferência e tamanho dos modelos. O Modelo 1, com uma estrutura de camadas mais simples (*Global Average Pooling* e camada densa com 512 neurônios), apresentou uma precisão média ligeiramente superior, com 88,59% de certeza. Além disso, o tempo médio de inferência foi de 71,38 ms, o que se mantém próximo do tempo do Modelo 2, mas com uma leve vantagem em relação ao consumo de memória (3,86 MB contra 4,05 MB).

Por outro lado, o Modelo 2, com a adição da camada *Conv2D* de 64 filtros, seguida por *MaxPooling* e *Flatten*, obteve uma precisão média de 86,88%. Embora o tempo médio de

Tabela 4.4: Comparação de Performance entre Modelos TensorFlow Lite com Diversas Configurações de Camadas

Imagem	**Certeza (%)**	**Tempo (ms)**	**Tamanho do Modelo (MB)**	**Classificação**	**Real**
Modelo 1 (3,86 MB)					
Imagem 1	83.20	185.21	3.86	Atenção	Atenção
Imagem 2	95.31	167.45	3.86	Atenção	Atenção
Imagem 3	73.05	150.32	3.86	Não Atenção	Não Atenção
Imagem 4	98.44	42.05	3.86	Não Atenção	Não Atenção
Imagem 5	94.53	44.55	3.86	Não Atenção	Atenção
Imagem 6	80.47	47.86	3.86	Atenção	Atenção
Imagem 7	77.73	44.67	3.86	Não Atenção	Não Atenção
Imagem 8	94.53	45.30	3.86	Não Atenção	Não Atenção
Imagem 9	99.61	43.96	3.86	Não Atenção	Não Atenção
Imagem 10	89.06	42.63	3.86	Atenção	Atenção
Média para o Modelo 1					
	88.59	71.38	3.86		
Modelo 2 (4,05 MB)					
Imagem 1	83.20	209.55	4.05	Atenção	Atenção
Imagem 2	55.08	223.44	4.05	Atenção	Atenção
Imagem 3	99.81	48.55	4.05	Não Atenção	Não Atenção
Imagem 4	99.99	43.36	4.05	Não Atenção	Não Atenção
Imagem 5	85.94	51.17	4.05	Atenção	Atenção
Imagem 6	85.94	46.18	4.05	Atenção	Atenção
Imagem 7	59.77	45.39	4.05	Não Atenção	Não Atenção
Imagem 8	99.80	45.55	4.05	Não Atenção	Não Atenção
Imagem 9	99.99	46.95	4.05	Não Atenção	Não Atenção
Imagem 10	99.22	43.76	4.05	Atenção	Atenção
Média para o Modelo 2					
	86.88	70.34	4.05		

inferência seja similar ao do Modelo 1 (70,34 ms), houve maior variabilidade, com alguns tempos de inferência mais elevados nas primeiras imagens processadas. O aumento no tamanho do modelo para 4,05 MB no Modelo 2 sugere que a camada convolucional e a maior quantidade de parâmetros contribuíram para um modelo mais robusto, mas com um pequeno custo em desempenho e eficiência de processamento.

A análise das matrizes de confusão apresentadas nas Tabelas 4.5 e 4.6 revela diferenças importantes no desempenho dos Modelos 1 e 2 na tarefa de classificação. O Modelo 1 apresentou uma taxa de acerto elevada, mas com um erro ao classificar a classe "Real" como "Falsa" em uma ocasião, resultando em cinco acertos e um erro para a classe "Real". Para a classe "Falsa", o modelo atingiu uma precisão completa, classificando corretamente todas as quatro instâncias, sem erros.

Em contrapartida, o Modelo 2 demonstrou uma performance superior, alcançando 100% de precisão em ambas as classes. Todas as instâncias foram corretamente classificadas, tanto para "Falsa" quanto para "Real", indicando ausência de erros. Esse resultado sugere que a inclusão das camadas adicionais de *Conv2D* e *MaxPooling* no Modelo 2 proporcionou uma capacidade mais robusta para capturar as características relevantes das imagens, aprimorando a distinção entre as classes e resultando em uma classificação mais precisa.

Esses achados destacam que o Modelo 2, com sua configuração aprimorada de camadas convolucionais, é mais adequado para cenários que exigem alta acurácia na classificação de imagens, demonstrando potencial para uso em aplicações que requerem uma análise detalhada e precisa das características espaciais.

Tabela 4.5: Matriz de Confusão – Modelo 1

		Predita	
		Falsa	Verdade
Real	Falsa	4	0
	Verdade	1	5

Tabela 4.6: Matriz de Confusão - Modelo 2

		Predita	
		Falsa	Verdade
Real	Falsa	5	0
	Verdade	0	5

4.3 Resultados do Aplicativo Móvel

Após a implantação do modelo Lite selecionado no aplicativo móvel, o sistema demonstrou um desempenho satisfatório tanto em termos de precisão quanto de usabilidade. O design do aplicativo, planejado para uma navegação intuitiva e de fácil manuseio, facilitou a interação do usuário com as funcionalidades de monitoramento em tempo real. A interface se mostrou responsiva e clara, conforme o usuário compreendesse rapidamente as indicações e alertas gerados pelo modelo, visto na Figura 4.1.

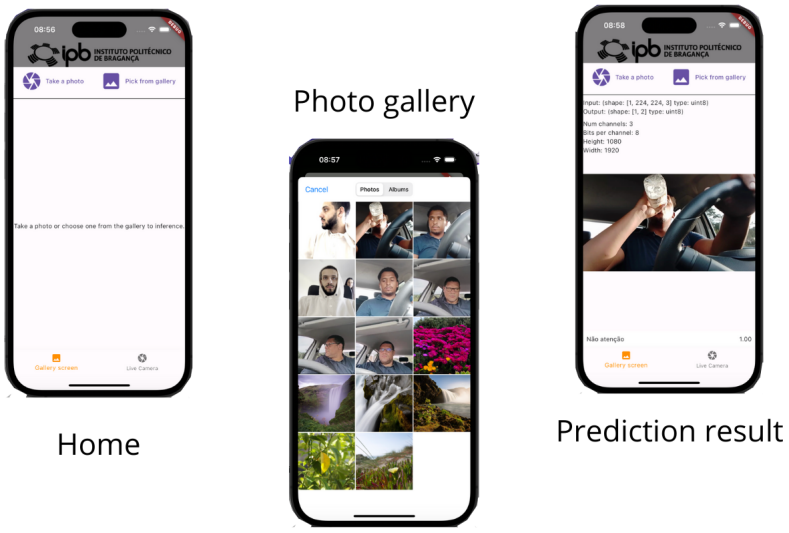

Figura 4.1: Telas do aplicativo

O modelo foi executado em dispositivos Android e obteve sucesso nas predições das classes de “Atenção” e “Não atenção”. Durante os testes, o modelo manteve uma taxa de acerto consistente, realizando inferências em tempo real com tempos de resposta adequados para o contexto móvel. Esse desempenho confirma a viabilidade do modelo Lite

quantizado para aplicação em dispositivos com recursos limitados, atendendo aos objetivos de precisão e eficiência, sem comprometer a experiência do usuário.

Capítulo 5

Conclusão e Trabalhos futuros

5.1 Conclusão

Os resultados gerais indicam que a quantização para o formato Lite é uma solução viável e eficaz para dispositivos móveis, proporcionando melhorias substanciais em termos de tempo de resposta e redução do tamanho de arquivo. Esses fatores são essenciais para a implementação em dispositivos móveis, onde a compactação de modelos reduz o consumo de memória, tornando-os ideais para ambientes com recursos limitados.

Além disso, as alterações nas camadas do Modelo 2 — especificamente a inclusão de uma camada *Conv2D* com 64 filtros e uma camada *MaxPooling* — foram fundamentais para aprimorar a capacidade de capturar características espaciais detalhadas nas imagens, resultando em uma classificação mais precisa. Esse aprimoramento é particularmente relevante em aplicações de monitoramento de atenção no trânsito, onde a capacidade de diferenciar rapidamente estados de atenção e distração pode aumentar significativamente a eficiência e a usabilidade do sistema em tempo real.

A utilização do framework Flutter com suporte à biblioteca TensorFlow Lite para o desenvolvimento do aplicativo móvel provou-se uma escolha bem-sucedida. Essa abordagem possibilitou a criação de um modelo otimizado para dispositivos móveis, integrando-se de forma eficaz ao sistema e operando conforme o esperado em dispositivos moveis. Com uma

interface intuitiva e responsiva e um modelo quantizado eficiente, essa solução se mostrou robusta e prática para o uso em ambientes com recursos limitados. Os resultados validam o uso de Flutter com TensorFlow Lite como uma plataforma completa e acessível para implementar modelos de aprendizado de máquina em dispositivos móveis.

5.2 Trabalhos futuros

A proposta desta tese de desenvolver uma solução que utiliza inteligência artificial em dispositivos móveis resultou em um método acessível e amplamente aplicável para monitorar a atenção no trânsito. Com a popularização dos smartphones, essa solução torna-se viável para um grande número de usuários, oferecendo uma ferramenta de baixo custo e alto impacto para combater a falta de atenção ao volante. Ao alertar para distrações em tempo real, o aplicativo contribui para a redução de acidentes e, potencialmente, para a preservação de vidas. O sucesso desse método ressalta a importância de explorar tecnologias móveis para enfrentar desafios sociais e de segurança, colocando nas mãos de qualquer motorista uma ferramenta eficiente e acessível para aumentar a segurança nas estradas.

Para aprimorar a eficácia e a aplicabilidade do sistema desenvolvido, alguns pontos foram identificados como direções promissoras para trabalhos futuros. Primeiramente, é necessário expandir o dataset utilizado, incorporando um número maior de usuários e variações nos cenários de captura das imagens. Essa expansão contribuiria para uma maior generalização do modelo, permitindo que ele reconheça padrões de distração em um conjunto mais amplo e diversificado de pessoas, o que é fundamental para aumentar a precisão e a robustez das predições.

Outro ponto de melhoria está no pré-processamento das imagens. Atualmente, o sistema utiliza a imagem inteira, o que poderia ser otimizado ao focar apenas em características diretamente associadas à falta de atenção. Para isso, seria possível aplicar uma máscara ao perfil da pessoa ou analisar somente a região do rosto, reduzindo o processamento da imagem completa e direcionando melhor o modelo. Essa abordagem

economizaria recursos computacionais e permitiria que o sistema fosse mais ágil e preciso.

Além disso, outra linha importante de desenvolvimento envolve o uso de câmeras de espectro infravermelho ou com ajuste de luz, possibilitando que o sistema funcione de maneira confiável em condições variadas de luminosidade, incluindo a condução noturna ou em ambientes de baixa iluminação. Esse aprimoramento permitiria que o sistema monitorasse o estado de atenção do motorista em tempo integral, independentemente das condições de luz externa.

Por fim, uma possibilidade interessante para o futuro é a integração do sistema com outros sistemas veiculares, como a frenagem automática e o envio de mensagens para terceiros em casos de desatenção prolongada. Esse tipo de integração poderia atuar preventivamente em situações de risco, por meio de ações automatizadas que interrompam a condução ou notifiquem contatos de emergência, aumentando a segurança e ampliando o impacto positivo do sistema na prevenção de acidentes no trânsito.

Bibliografia

[1] S. Kaplan, M. A. Guvensan, A. G. Yavuz e Y. Karalurt, "Driver Behavior Analysis for Safe Driving: A Survey", *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, n.º 6, pp. 3017–3032, 2015. DOI: 10.1109/TITS.2015.2462084.

[2] *Global Status Report on Road Safety 2018*. Geneva, Switzerland: World Health Organization, 2018, Accessed on: January 7, 2024. URL: https://www.who.int/violence_injury_prevention/road_safety_status/2018/en/.

[3] W. H. Organization, *Mobile phone use: a growing problem of driver distraction*. Geneva, Switzerland: World Health Organization, 2011, Accessed on: January 7, 2024. URL: https://www.who.int/publications/i/item/mobile-phone-use-a-growing-problem-of-driver-distraction.

[4] IBGE, *161,6 milhões de pessoas com 10 anos ou mais de idade utilizaram a Internet no país, em 2022*, https://encurtador.com.br/nEQS1, nov. de 2023.

[5] Direção-Geral da Saúde, *Relatório Mundial da Segurança Rodoviária*, Consultado em 12-08-2017, 2015. URL: https://www.dgs.pt.

[6] Autoridade Nacional de Segurança Rodoviária, *Plano Estratégico Nacional de Segurança Rodoviária - PENSE 2020*, Consultado em 2020, 2020. URL: https://www.ansr.pt.

[7] A. Desconhecido, "Estudo do Perfil dos Atropelamentos na cidade de Braga", *Relatório de Sinistralidade Rodoviária*, 2020, Análise quantitativa de acidentes e suas consequências. URL: https://comum.rcaap.pt.

[8] Organização das Nações Unidas (ONU), *Década de Ações para a Segurança no Trânsito 2011-2020*, [s.l.]: ONU, (Nota Técnica), 2011.

[9] C. H. R. de Carvalho e E. P. Guedes, *Balanço da primeira década de ação pela segurança no trânsito no Brasil e perspectivas para a segunda década*. Administração Pública. Governo. Estado, 2023, Publicado em Novembro de 2023.

[10] "OMS - Plano Global - Década de Ação pela Segurança no Trânsito 2021-2030", Organização Mundial da Saúde (OMS), rel. téc., out. de 2021, Documento Técnico.

[11] L. Masello, G. Castignani, B. Sheehan, F. Murphy e K. McDonnell, "On the road safety benefits of advanced driver assistance systems in different driving contexts", *Transportation Research Interdisciplinary Perspectives*, vol. 15, p. 100 670, 2022, ISSN: 2590-1982. DOI: https://doi.org/10.1016/j.trip.2022.100670. URL: https://www.sciencedirect.com/science/article/pii/S2590198222001300.

[12] R. Najjar, "Redefining Radiology: A Review of Artificial Intelligence Integration in Medical Imaging", *Diagnostics*, vol. 13, n.º 17, p. 2760, 2023. DOI: 10.3390/diagnostics13172760.

[13] W. S. McCulloch e W. Pitts, "A logical calculus of the ideas immanent in nervous activity", *The bulletin of mathematical biophysics*, vol. 5, n.º 4, pp. 115–133, 1943.

[14] F. Rosenblatt, "The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain", *Psychological Review*, vol. 65, n.º 6, pp. 386–408, 1958.

[15] M. Minsky e S. Papert, *Perceptrons: An Introduction to Computational Geometry*. MIT Press, 1969.

[16] D. E. Rumelhart, G. E. Hinton e R. J. Williams, "Learning representations by back-propagating errors", *Nature*, vol. 323, pp. 533–536, 1986.

[17] D. E. Rumelhart e J. L. McClelland, "Parallel distributed processing: Explorations in the microstructure of cognition. Vol. 1", 1986.

[18] Y. LeCun, L. Bottou, Y. Bengio e P. Haffner, "Gradient-based learning applied to document recognition", *Proceedings of the IEEE*, vol. 86, n.º 11, pp. 2278–2324, 1998.

[19] S. Hochreiter e J. Schmidhuber, "Long Short-Term Memory", *Neural Computation*, vol. 9, n.º 8, pp. 1735–1780, 1997.

[20] I. Goodfellow, Y. Bengio e A. Courville, *Deep Learning*. MIT Press, 2016.

[21] A. Krizhevsky, I. Sutskever e G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks", *Advances in Neural Information Processing Systems*, vol. 25, pp. 1097–1105, 2012.

[22] S. S. Haykin, *Redes Neurais: Princípios e Prática*. Brazil: Bookman, 2001, Translated title: Neural Networks: Principles and Practice.

[23] A. Desconhecido, *Modelos Dinâmicos*, `https://www.marilia.unesp.br/Home/Instituicao/Docentes/EdbertoFerneda/mrip08-modelosdinamicos.pdf`, Inspirado na representação de redes neurais artificiais, 2019.

[24] W. Zaremba, I. Sutskever e O. Vinyals, "Recurrent Neural Network Regularization", *arXiv e-prints*, arXiv:1409.2329, set. de 2014, Available at: https://arxiv.org/abs/1409.2329.

[25] Y. LeCun, L. Bottou, Y. Bengio e P. Haffner, "Gradient-based learning applied to document recognition", em *Proceedings of the IEEE*, Available at: https://ieeexplore.ieee.org/documer vol. 86, 1998, pp. 2278–2324.

[26] A. Krizhevsky, I. Sutskever e G. E. Hinton, "ImageNet classification with deep convolutional neural networks", em *Communications of the ACM*, Available at: https://dl.acm.org/doi/10.1145/3065386, vol. 60, 2012, pp. 84–90.

[27] D. E. Rumelhart, G. E. Hinton e R. J. Williams, "Learning representations by back-propagating errors", *Nature*, vol. 323, n.º 6088, pp. 533–536, 1986, Available at: https://www.nature.com/articles/323533a0.

[28] I. Goodfellow, Y. Bengio e A. Courville, *Deep Learning*. MIT Press, 2016, Available at: https://www.deeplearningbook.org/.

[29] W. Rawat e Z. Wang, "Deep convolutional neural networks for image classification: A comprehensive review", *Neural Computation*, vol. 29, n.º 9, pp. 2352–2449, 2017, Available at: https://www.mitpressjournals.org/doi/10.1162/NECO$_{a0}$0990.

[30] K. P. Murphy, *Machine Learning: A Probabilistic Perspective*. MIT Press, 2012.

[31] T. Hastie, R. Tibshirani e J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2009.

[32] M. A. Nielsen, *Redes Neurais e Aprendizagem Profunda*. Determination Press, 2015, Last updated on: December 26, 2019.

[33] E. W. T. Ngai, Y. Hu, Y. Wong, Y. H. Chen e X. Sun, "The application of data mining techniques in financial fraud detection: A classification framework and an academic review of literature", *Decision Support Systems*, vol. 50, n.º 3, pp. 559–569, 2011.

[34] A. K. Jain, M. N. Murty e P. J. Flynn, "Data clustering: A review", *ACM Computing Surveys (CSUR)*, vol. 31, n.º 3, pp. 264–323, 1999.

[35] J. MacQueen, "Some methods for classification and analysis of multivariate observations", em *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, 1967, pp. 281–297.

[36] D. Arthur e S. Vassilvitskii, "K-means++: The Advantages of Careful Seeding", *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 1027–1035, 2007.

[37] F. Murtagh e P. Contreras, "Algorithms for hierarchical clustering: an overview", *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 2, n.º 1, pp. 86–97, 2012.

[38] I. T. Jolliffe, *Principal Component Analysis*. Springer, 2002.

[39] K. Pearson, "On lines and planes of closest fit to systems of points in space", *Philosophical Magazine*, vol. 2, n.º 11, pp. 559–572, 1901.

[40] H. Hotelling, "Analysis of a complex of statistical variables into principal components", *Journal of Educational Psychology*, vol. 24, n.º 6, p. 417, 1933.

[41] S. Wold, K. Esbensen e P. Geladi, "Principal component analysis", *Chemometrics and Intelligent Laboratory Systems*, vol. 2, n.º 1-3, pp. 37–52, 1987.

[42] M. Wedel e W. A. Kamakura, *Market Segmentation: Conceptual and Methodological Foundations*. Springer, 2012.

[43] A. G. Howard, M. Zhu, B. Chen et al., "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications", em *arXiv preprint arXiv:1704.04861*, Available at: https://arxiv.org/abs/1704.04861, 2017.

[44] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov e L.-C. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks", em *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Available at: https://arxiv.org/abs/1801.04381, 2018, pp. 4510–4520.

[45] Y. Zhang, X. Huang e J. Zhang, "MobileNet Based Efficient Neural Network Design for Real-Time Object Detection on Mobile Devices", *Journal of Signal Processing Systems*, vol. 92, n.º 5, pp. 431–440, 2020, Available at: https://link.springer.com/article/10.1007/s112 019-01479-7.

[46] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever e R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting", *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014.

[47] M. Lin, Q. Chen e S. Yan, "Network In Network", *arXiv preprint arXiv:1312.4400*, 2013.

[48] V. Nair e G. E. Hinton, "Rectified Linear Units Improve Restricted Boltzmann Machines", pp. 807–814, 2010.

[49] M. Abadi, A. Agarwal, P. Barham et al., "TensorFlow: Large-scale machine learning on heterogeneous systems", *arXiv preprint arXiv:1603.04467*, 2016, Available at: https://www.tensorflow.org/.

[50] J. Dean, G. S. Corrado, R. Monga et al., "Large Scale Distributed Deep Networks", em *Advances in Neural Information Processing Systems*, vol. 25, 2012, pp. 1223–1231.

[51] TensorFlow, *TensorFlow Lite*, Available at: https://www.tensorflow.org/lite.

[52] TensorFlow, *TensorFlow.js*, Available at: https://www.tensorflow.org/js.

[53] M. Abadi, P. Barham, J. Chen et al., "TensorFlow: A System for Large-Scale Machine Learning", *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation*, pp. 265–283, 2017.

[54] Statista. "Number of smartphone users worldwide from 2014 to 2029". Acesso em: 11 de março de 2024. (fev. de 2024), URL: `https://www.statista.com/forecasts/1143723/smartphone-users-in-the-world`.

[55] Flutter Team. "Flutter documentation". Acesso em: 11 de março de 2024. (s.d.), URL: `https://docs.flutter.dev`.

[56] M. FITTIPALDI, "Por dentro da câmera digital", *Guia prático digital: fotografe melhor*, vol. 7, n.º 3, pp. 14–25, 2003.

[57] I. HUTCHINSON e P. WILLIAMS, "Digital cameras", *Br J orthod*, vol. 26, n.º 4, pp. 326–331, dez. de 1999.

[58] NVIDIA. "What Is FPS and How It Helps You Win Games". Acesso em: 07 de março de 2024. (s.d.), URL: `https://www.nvidia.com/en-us/geforce/news/what-is-fps-and-how-it-helps-you-win-games/`.

[59] Google Developers, *Dart Language Tour*, Available at: https://dart.dev/guides/language/language-tour, 2018.

[60] C. Sells, *Flutter Isolates for Concurrent Programming*, Available at: https://medium.com/flutter/flutt isolates-concurrent-programming-in-dart-d92c16a06bc8, 2019.

[61] F. Documentation, *Concurrent Programming in Dart with Isolates*, Available at: https://flutter.dev/docs/development/packages-and-plugins/using-packages.

[62] Dart Language Team, *Asynchronous Programming: Isolates, Event Loops, and Threading*, Available at: https://dart.dev.

[63] F. Chollet, *Deep Learning with Python*. Manning Publications, 2018.

[64] C. Szegedy, W. Liu, Y. Jia et al., "Going deeper with convolutions", *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015.

[65] T. E. Oliphant, "A guide to NumPy", 2006.

[66] S. Ioffe e C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift", *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, vol. 37, pp. 448–456, 2015.

[67] L. Perez e J. Wang, "The Effectiveness of Data Augmentation in Image Classification using Deep Learning", *arXiv preprint arXiv:1712.04621*, 2017.

[68] C. Shorten e T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning", *Journal of Big Data*, vol. 6, p. 60, 2019.

[69] D. M. Powers, "Evaluation: From Precision, Recall and F-measure to ROC, Informedness, Markedness Correlation", *Journal of Machine Learning Technologies*, vol. 2, n.º 1, pp. 37–63, 2011.

[70] M. Sokolova e G. Lapalme, "A systematic analysis of performance measures for classification tasks", *Information Processing & Management*, vol. 45, n.º 4, pp. 427–437, 2009.

[71] S. V. Stehman, "Selecting and interpreting measures of thematic classification accuracy", *Remote Sensing of Environment*, vol. 62, n.º 1, pp. 77–89, 1997.

[72] T. Saito e M. Rehmsmeier, "Precision-Recall Plot as a Diagnostic Tool for Multi-Class Classification", *Bioinformatics*, vol. 31, n.º 2, pp. 192–201, 2015.

[73] N. Chinchor, "MUC-4 evaluation metrics", em *Proceedings of the 4th Conference on Message Understanding*, 1992, pp. 22–29.

[74] F. Karim, "LSTM Fully Convolutional Networks for Time Series Classification", *IEEE Access*, vol. 6, pp. 1662–1669, dez. de 2017.

[75] J. Smith e S. Jones, "Driver fatigue detection using LSTM networks", *IET Intelligent Transport Systems*, vol. 12, pp. 567–574, 2018. DOI: 10.1049/iet-its.2018.5392.

[76] L. Geng, Z. Hu, Z. Xiao e et al., "Real-time fatigue driving recognition system based on deep learning and embedded platform", *American Scientific Research Journal for Engineering, Technology, and Sciences (ASRJETS)*, vol. 53, n.º 1, pp. 164–175, 2019.

[77] Bosch, *AI-based Driver Monitoring Systems*, https://www.bosch-ai.com/driver-monitoring, 2020.

[78] K. Lee e M. Park, "Android-based real-time drowsiness detection", *Mobile Computing Journal*, vol. 10, pp. 233–242, 2018.

[79] S. Kim e E. Cho, "3D Convolutional Networks for Real-Time Monitoring of Driver Drowsiness", *Journal of Mobile Applications*, vol. 15, pp. 98–115, 2019.

[80] H. Tanaka e K. Yamamoto, "Facial landmark-based drowsiness detection for Android systems", *Android Development Journal*, vol. 5, pp. 89–101, 2020.

[81] R. Silva, "Detection of Distraction and Drowsiness in Drivers using Neural Networks", tese de doutoramento, Universidade Federal de Santa Catarina, 2023.

[82] B. Krishnamurthy, "An Introduction to the ReLU Activation Function", *Built In*, 2024, Written by Bharath Krishnamurthy. URL: https://builtin.com/machine-learning/relu-activation-function.

[83] K. Team, *Adam Optimizer*, 2024. URL: https://keras.io/api/optimizers/adam/.

[84] T. Authors, *TensorFlow Lite Quantization Specification*, Accessed: 2024-06-07, 2024. URL: https://www.tensorflow.org/lite/performance/quantization_spec.