FRANCISCO REBELO DE ANDRADE DA COSTA

Licenciado em Engenharia Eletrotécnica e de Computadores

DESENVOLVIMENTO DE UMA DASH CAMERA COM CAPACIDADES DE INTELIGÊNCIA ARTIFICIAL PARA PREVENÇÃO DE ACIDENTES RODOVIÁRIOS

DISSERTAÇÃO PARA A OBTENÇÃO DO GRAU DE MESTRE EM ENGENHARIA ELETROTÉCNICA E DE COMPUTADORES

MESTRADO EM ENGENHARIA ELETROTECNICA E DE COMPUTADORES

Universidade NOVA de Lisboa

Setembro, 2023

Orientador: Professor João Rosas,
Professor Auxiliar, Universidade NOVA de Lisboa

DEPARTAMENTO DE ENGENHARIA ELETROTECNICA E DE COMPUTADORES

Júri:

Presidente: Doutor Pedro Miguel Figueiredo Amaral
Professor Auxiliar, FCT-NOVA

Arguentes: Doutor José Manuel Matos Ribeiro da Fonseca
Professor Associado, FCT-NOVA

Orientador: Doutor João Almeida Rosas
Professor Auxiliar, FCT-NOVA

UTILIZAÇÃO DE INTELIGÊNCIA ARTIFICIAL PARA O DESENVOLVIMENTO DE UMA APLICAÇÃO DASH CAMERA

AGRADECIMENTOS

Dedico esta dissertação à minha mulher, por todo o apoio dado durante o desenvolvimento da mesma; aos meus filhos por terem servido de motivação para a terminar, aos meus pais por terem apoiado e incentivado nos momentos mais difíceis do curso e ao Professor João Rosas pela disponibilidade demonstrada e apoio dado durante o desenvolvimento desta dissertação mesmo com a sua disponibilidade altamente limitada. Um especial agradecimento e abraço de saudade ao Professor Tiago Cardoso que nunca será esquecido.

"Behind every successful man there's a lot of unsuccessful years." (Bob Brown).

RESUMO

No mundo acelerado de hoje, a segurança rodoviária tornou-se uma grande preocupação. Todos os anos, ocorrem milhões de acidentes rodoviários, levando a perdas significativas de vidas humanas. Embora as *Dash Câmaras (DC)* sejam uma ferramenta popular para registar incidentes, torna-se essencial a necessidade do processamento destes dados em tempo real para prevenir acidentes. A solução proposta neste trabalho, através de algoritmos avançados de Inteligência Artificial (IA), não é apenas uma ferramenta de gravação, já que esta propõe o processamento de dados em tempo real dos sensores, câmara e áudio do smartphone para detetar potenciais situações de perigo e proactivamente alertar o condutor ou iniciar chamadas de emergência. A aplicação é capaz de reconhecer objetos na estrada, desde peões até veículos e sinais de trânsito, consegue reconhecer o estado emocional do condutor e, ao analisar os dados dos sensores, é capaz de detetar padrões de uma condução agressiva e oferecer feedback para promover hábitos de condução mais seguros. A aplicação possui um assistente de bordo inteligente que reconhece comandos de voz, permitindo que os condutores controlem as funcionalidades sem tirar as mãos do volante. **Nem um byte de dados sai do telefone –** ao aplicar conceitos de *Edge Computing*, todos os dados e modelos são processados localmente, garantido assim uma total privacidade, baixa latência e total funcionamento em áreas sem ligação à internet. O objetivo deste trabalho é desenvolver uma prova de conceito de uma aplicação móvel potenciada por IA, acessível a todos os utilizadores que possuam um smartphone, independentemente da sua qualidade. Acreditamos que a segurança rodoviária é um direito básico e todos devem ter acesso às ferramentas que melhor garantam a sua segurança.

Palavras-chave: Segurança Rodoviária, *Dash Camera*, Inteligência Artificial, *Edge Computing*, *Computer Vision*, *Tiny ML*, Telemática, Machine Learning, Sensores, Tensorflow

ABSTRACT

In today's fast-paced world, road safety has become a major concern. Every year, millions of road accidents occur, leading to significant loss of lives. While Dash Cameras are a popular tool for recording incidents, there is an essential need for real-time data processing to prevent accidents. The solution proposed in this work, through advanced Artificial Intelligence algorithms, is not just a recording tool; it processes real-time data from smartphone sensors, camera, and audio to detect potential hazardous situations and trigger actions such as issuing alerts to the driver or initiating emergency calls. The application is capable of recognizing objects on the road, from pedestrians to vehicles and traffic signs, can recognize the emotional state of the driver, and by analyzing sensor data, can detect patterns of aggressive driving and offer feedback to promote safer driving habits. The application has an intelligent onboard assistant that recognizes voice commands, allowing drivers to control features without taking their hands off the wheel. Not a byte of data leaves the phone - by applying Edge Computing concepts, all data and models are processed locally, thus ensuring total privacy, low latency, and full functionality in areas without an internet connection. The aim of this work is to demonstrate the feasibility of an AI-powered mobile application, making it accessible to all users who own a smartphone, regardless of its quality. We believe that road safety is a right, and everyone should have access to the best tools to ensure their safety.

Keywords: Road Safety, Dash Camera, Artificial Intelligence, Edge Computing, Computer Vision, TinyML, Telematics, Machine Learning, Sensors, Tensorflow

ÍNDICE

1 Introdução 1

2 Estado da Arte 4

2.1 Mortalidade em condutores recém encartados 4

2.2 As principais causas 5

2.3 Dados telemáticos na condução de um veículo 7

2.4 Aplicações práticas do uso de dados telemáticos 8

2.5 Barreiras e *Pitfalls* 10

2.6 Tecnologias para monitorização do estilo de condução 11

2.7 Armazenamento de dados 13

2.8 Machine Learning 14

2.9 Ferramentas para projetos IoT de provedores de serviços Cloud 22

2.10 Aplicações práticas de ML na deteção de condução negligente através de sensores e câmara do telefone 23

2.11 Limitações das soluções disponíveis no mercado 23

2.12 Identificação das lacunas 27

3 Desenvolvimento do Sistema 28

3.1 Descrição das Necessidades 28

3.2 Requisitos Funcionais e Use Cases 31

3.3 Metodologia de Trabalho 33

3.4 Arquitetura do Sistema 34

3.5 Criação dos Datasets 36

3.6 Modelos Machine Learning 39
3.7 Aplicação Mobile 64
4 Testes e Validação 82
4.1 Testes no mundo real 82
4.2 Verificação dos requisitos 85
4.3 Validação do sistema 87
4.4 Análise crítica e reflexão 88
5 Conclusões e Trabalho Futuro 91
5.1 Síntese de Trabalho Desenvolvido e Resultados Obtidos 91
5.2 Trabalho futuro 92
6 Referências 95

ÍNDICE DE FIGURAS

Figura 1 - Distribuição do número de feridos e mortes por idade [12] 4
Figura 2 - Causas de Acidentes de automóvel. Fonte: www.calljed.com 5
Figura 3 - Distribuição das causas de acidentes [5] 6
Figura 4 - Exemplo de sensores presentes num automóvel inteligente 8
Figura 5 - Identificação do condutor e monitorização das expressões faciais. Fonte: www.valeo.com 10
Figura 6 - Protocolos de Comunicação - Fonte: https://hashstudioz.com 12
Figura 7 - Principais conceitos e interseção - Fonte: https://medium.com/ 16
Figura 8 - Deep Compression [37]. 20
Figura 9 - Placas de desenvolvimento ESP - comparação. Fonte: https://www.kuongshun-ks.com 21
Figura 10 - Arduino Nano 33 BLE Sense. Fonte: https://gilberttanner.com 21
Figura 11 - IoT Framework. Fonte: https://blog.mobcoder.com 22
Figura 12 - Distribuição de aplicações da Play Store por classificação 24
Figura 13 - Imagem do projeto TJ Free Dash Camera 26
Figura 14 - Exemplo de uma tarefa no JIRA 33
Figura 15 - Exemplo de um commit associado a uma tarefa do JIRA. 34
Figura 16 - Arquitetura Proposta 34
Figura 17 - Anotação de uma 'bike' numa imagem recolhida da internet 37
Figura 18 - Explicação dos pontos de dados, intervalo de tempo e eventos. 38
Figura 19 - Especificações da instância de treino dos modelos 41
Figura 20 - Pipeline de treino do modelo de classificação de movimento do sensor. 41
Figura 21 - Eixo do acelerómetro e giroscópio 42
Figura 22 - Exemplo de leituras de sensores do dataset de treino. 42
Figura 23 - Distribuição de amostras de treino por tipo de movimento - imagem de autor. 43
Figura 24 - Exemplos de leituras de sensores de cada classe de movimento 44
Figura 25 - Matriz de correlação do subconjunto do dataset de treino 44
Figura 26 - Amostras de 'hard break' plotados para o acelerómetro x e y 45
Figura 27 - Curva de distribuição de Z-score - imagem de Vitalflux 45
Figura 28 - Loss Function e precisão ao longo das épocas de treino (RNN à esquerda e LSTM à direita) 48
Figura 29 - Métricas dos modelos (RNN à esquerda e LSTM à direita) 49

Figura 30 - Matriz de confusão do melhor modelo RNN 50
Figura 31 - Exemplo dos resultados de compressão com Clustering – Imagem da documentação do Tensorflow 51
Figura 32 - Resultados de métodos de Quantização. Fonte: Documentação do Tensorflow 52
Figura 33 - Resultados da aplicação de PTQ e QAT em diferentes modelos. Fonte: documentação do Tensorflow 52
Figura 34 - Pipeline de treino do modelo de deteção de sinais de trânsito 53
Figura 35 - Exemplo de Data Augmentation – Fonte: Data Camp 54
Figura 36 - Curvas PR e ROC. Imagem de machinelearningmastery.com 55
Figura 37 - Registo das métricas do modelo ao longo do treino 56
Figura 38 - Sumário do modelo. 57
Figura 39 - Precisão e loss function de treino ao longo das épocas 58
Figura 40 - Comparação dos fatores de decisão nos modelos treinados. 61
Figura 41 - Imagem do dataset FER2013. Fonte: Kaggle 61
Figura 42 - Pipeline de treino do modelo de reconhecimento de expressão facial 62
Figura 43 - Funcionalidade de Dash Camera da aplicação 66
Figura 44 - Definição da lógica de navegação entre os módulos da aplicação 67
Figura 45 - Ficheiros de definição de variáveis estáticas 68
Figura 46 - Exemplo de estrutura do diretório de armazenamento de vídeos e dados 69
Figura 47 - Gestão das permissões da aplicação 69
Figura 48 - Lógica funcional do sistema de ações 70
Figura 49 - Definição das ações a desencadear através de comandos 71
Figura 50 - Lista de ações desencadeadas 72
Figura 51 - Exemplo da lógica que desencadeia uma ação 73
Figura 52 - Exemplo dos parâmetros de uma ação 76
Figura 53 - Ecrã inicial e de Definições da aplicação 78
Figura 54 - Módulo Dash Camera e informação da camada de Overlay 80
Figura 55 - Imagens do dataset FER 2013 (em cima) e imagens do dataset customizado (em baixo) 83

ÍNDICE DE TABELAS

Tabela 1 - Casos de estudo 31
Tabela 2 - Requisitos Funcionais 32
Tabela 3 - Requisitos Não Funcionais 32
Tabela 4 - Métricas de avaliação e tamanho dos modelos finais quantizados. 49
Tabela 5 - Progresso do modelo ao longo das várias épocas de treino: 58
Tabela 6 - Resultados das experiências e modelo inicial de deteção de objetos na estrada 59
Tabela 7 - Tamanho e precisão dos modelos finais antes e depois PTQ 60
Tabela 8 - Resultados dos modelos de reconhecimento de expressão facial treinados.. 62
Tabela 9 - Testes da aplicação no mundo real 85
Tabela 10 - Verificação dos Requisitos Funcionais 86
Tabela 11 - Verificação dos Requisitos Não Funcionais 87

SIGLAS

RFID	Radio Frequency ID
MCU	Micro-controller Unit
BLE	Bluetooth Low energy
NFC	Near Field Communication
LORAWAN	Long Range Wide Area Network
OSS	Open Source Software
ML	Machine Learning
AI	Artificial Intelligence
NAS	Neural Architecture Search
DIY	Do It Yourself
SVM	Support Vector Machines
DOE	Deteção de Objetos na Estrada
REF	Reconhecimento de Expressão Facial
CM	Classificação de Movimento
RCV	Reconhecimento de Comandos de Voz
RNN	Recurrent Neural Network
LSTM	Long Short Term Memory
PTQ	Post Training Quantization
mAP	Mean Average Precision
CNN	Convolutional Neural Network
NN API	Neural Network API
FPS	Frames Per Second

1

INTRODUÇÃO

Num mundo cada vez mais acelerado, desastres de automóvel representam uma das maiores causas de mortalidade, tirando a vida todos os anos a aproximadamente 1.2M de pessoas e causando mais de 50M de feridos [1]. Em 2022, nos Estados Unidos, mais de 2.000 jovens-adultos, com idades compreendidas entre os 16 e os 19 anos perderam a vida na estrada e cerca de 300.000 foram levados de emergência com lesões graves para o hospital [2].

Um outro estudo realizado pela "*American Journal of Preventive Medicine*" aponta que pessoas com menos de um ano de carta têm uma propensão para estar envolvidos num acidente de automóvel quase duas vezes superior. As principais causas aparentam ser não só a falta de experiência em manusear o automóvel, mas também a incapacidade de prever situações e comportamentos dos outros condutores, a subestimação de potenciais situações e zonas de perigo e a condução negligente (direta ou indiretamente) tais como a distração na condução, comportamentos de risco sob o efeito de álcool ou estupefacientes [3]. Sendo a "Condução Distraída" e cansaço as principais causas, o avanço tecnológico mais recente vem trazer uma esperança no combate a estes fatores.

Os carros recentes vêm equipados com centenas de sensores que permitem uma captação do ambiente envolvente na condução, tanto internamente como externamente, combinando os dados dos sensores com imagens e áudio. Estes veículos são desenvolvidos com tecnologia de ponta e com processamento local dos dados que permite uma reação autónoma a potenciais casos de desastre, tais como detetar uma mudança de faixa perigosa, uma colisão iminente com outro veículo, o possível atropelamento de um peão ou mesmo a deteção de cansaço e sonolência do condutor. Sendo que estes carros mais modernos ainda representam menos de 3,3% dos novos registos [4], temos ainda uma grande maioria da população sem acesso nem condições financeiras para a aquisição dos mesmos.

Historicamente, o mundo académico tem tido um grande papel no desenvolvimento deste tipo de tecnologias, com diversos projetos em campos como a deteção de condução distraída [5] e fadiga [6] , a classificação de eventos de colisão e pré-colisão [7], deteção e reconhecimento de entidades e sinais de trânsito na estrada, deteção de padrões de condução agressiva através dos sensores do veículo e deteção de atropelamento eminente [8].

No entanto, ao analisar a quantidade de condutores que beneficiam diariamente deste tipo de tecnologias, parece haver ainda uma barreira significativa na sua adoção em massa.

Este trabalho propõe o desenvolvimento de uma *Dash Camera* móvel com capacidades preventivas, utilizando Inteligência Artificial na sua base. Esta solução será mais do que um simples dispositivo de filmagem; queremos que seja um copiloto que ajude na prevenção de acidentes ao detetar potenciais situações de perigo, tais como sinais de trânsito, condução agressiva ou condução distraída. Estes modelos são meticulosamente criados para compreender um espectro de fatores, desde a expressão facial do condutor até às entidades ou sinais de trânsito presentes na estrada e ao movimento do veículo. Para garantir uma experiência centrada no utilizador, a aplicação possui um assistente de bordo comandado por comandos de voz, permitindo um controlo total da aplicação sem o uso das mãos. A utilidade da aplicação não se limita à interpretação do ambiente envolvente, pois a mesma alerta proactivamente os condutores sobre potenciais perigos através da emissão de sons ou *flash* - estes avisos podem ser definidos pelo utilizador para proporcionar uma experiência de condução única e personalizada. Em situações de crise, a aplicação poderá automaticamente iniciar uma chamada para um número de emergência bem como enviar uma mensagem de texto e WhatsApp com o pedido de socorro e partilhando as coordenadas GPS - esta redundância de avisos aumenta a probabilidade de sucesso na entrega da mensagem de socorro considerando a potencial falha na rede móvel ou serviço de dados.

Uma das principais características da aplicação é o facto de todo o processamento ser feito localmente. Devido à sua capacidade de "Edge Computing", a mesma opera de forma contínua sem necessitar de ligação à internet, com todos os modelos de ML otimizados para dispositivos móveis e sem que nunca sequer um byte de dados saia do dispositivo. Esta escolha de implementação abdica de modelos mais robustos servidos em grandes servidores através de APIs em prol de uma total autossuficiência, segurança e privacidade dos dados e uma velocidade de processamento e inferência dos modelos significativamente maior e ininterrupta em áreas com fraca ligação.

Durante a utilização, a aplicação vai recolhendo um largo volume de dados - leituras dos sensores e respostas dos modelos de imagem e áudio - que ficam registados numa base de dados local no telefone. Estes dados podem futuramente ser utilizados para gerar informação analítica e automaticamente atribuir uma avaliação do nível de condução segura e ecológica.

Ao "gamificar" a experiência de condução [9], a aplicação encoraja hábitos de condução mais seguros e desta forma contribuirá, hipoteticamente, para a redução do número de acidentes rodoviários. [10]

De forma a oferecer uma solução democratizada e acessível a todos os condutores - sendo o único requisito um smartphone - a aplicação terá em consideração as diferentes especificações dos telemóveis aquando do desenvolvimento dos modelos e restantes funcionalidades. Esta variedade e flexibilidade garante um desempenho otimizado, seja num smartphone económico ou num dispositivo topo de gama.

A estrutura da dissertação começa com este capítulo introdutório (Introdução), onde é dado um contexto sobre o problema, seguindo-se de um estudo alargado (Estado da Arte) sobre pesquisas, investigações ou projetos que abordem os mesmos problemas e uma análise detalhada às soluções propostas bem como as conclusões e resultados e a deteção das lacunas existentes. Após este estudo e uma reflexão sobre as lacunas das soluções investigadas, o terceiro capítulo (Proposta) irá conter o objetivo deste trabalho, ou seja, uma proposta de solução para o problema apresentado e adicionalmente serão também fundamentadas algumas das decisões e pressupostos assumidos para a execução da mesma. O capítulo seguinte (Desenvolvimento) irá relatar a execução do projeto passo a passo, todos os problemas e obstáculos ultrapassados e as decisões que foram tomadas. O trabalho termina com a apresentação dos Resultados e Conclusão que irá conter uma análise crítica e reflexão sobre o sucesso do projeto e pontos de melhoria, bem como uma análise detalhada sobre os requisitos funcionais implementados e o que foi selecionado para trabalho futuro.

2

ESTADO DA ARTE

2.1 Mortalidade em condutores recém encartados

Os acidentes rodoviários são uma preocupação crescente em muitos países, sendo considerados como uma das principais causas de morte entre jovens adultos. A resolução deste problema é um desafio devido à natureza imprevisível dos acidentes, que podem ocorrer devido a uma distração ou erro minúsculo, mas que podem ter consequências fatais.

Num artigo publicado no website "Psychology Today" [11] , o autor especialista em tratar vítimas que sobreviveram a acidentes rodoviários refere que entre muitas outras, as principais causas de mortalidade nas estradas de jovens recém encartados se deve sobretudo à falta de prática para a obtenção da carta em alguns países – por exemplo, nos EUA para obter a licença de condução é necessário menos de 10 vezes o número de horas que na Alemanha.

Como evidenciado na figura 1, é possível observar uma clara correlação entre a probabilidade de acidentes fatais com a idade do condutor.

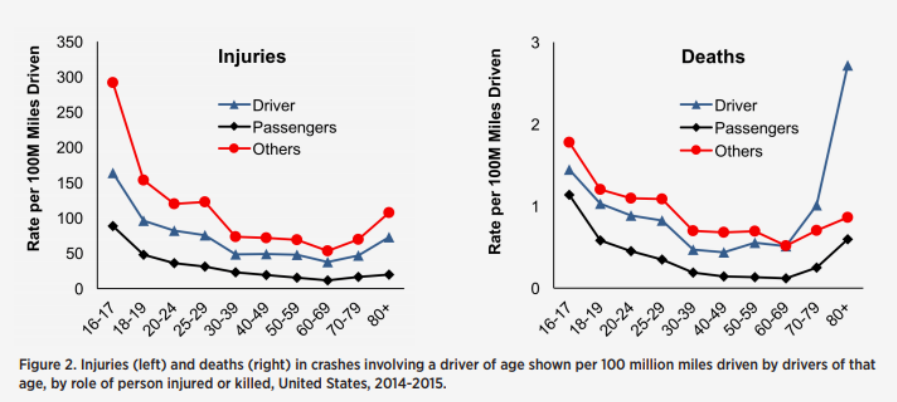

Figura 1 - Distribuição do número de feridos e mortes por idade [12]

Tirando os fatores mais expectáveis, tais como a falta de experiência ou a fraca capacidade de prever o comportamento de outros condutores e peões, muitas outras causas são originadas pela negligência do condutor jovem.

2.2 As principais causas

A principal causa de acidentes rodoviários têm vindo a sofrer alterações ao longo das últimas décadas. Antigamente as principais causas deviam-se à menor qualidade dos veículos e falta de normas de segurança como a obrigatoriedade do cinto, sendo que atualmente, como observamos na figura 2, as principais causas devem-se na esmagadora maioria dos casos a erro humano.

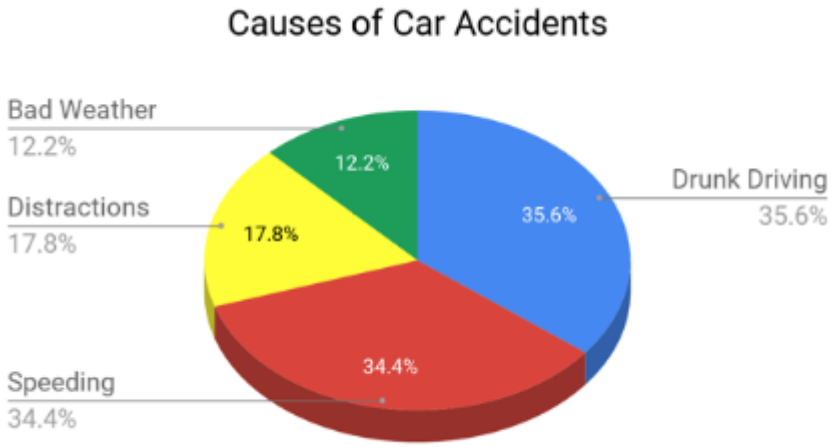

Figura 2 - Causas de Acidentes de automóvel. Fonte: www.calljed.com

2.2.1 O fator humano

Inúmeros estudos foram feitos recorrendo a dados de sensores e câmaras nos momentos que antecedem os acidentes de viação. Alguns destes estudos apontam que houve uma grande alteração nas principais causas de acidentes ao longo dos anos sendo que nos casos mais atuais, perto de 94% dos acidentes demonstraram que nos segundos que antecedem o acidente, algum tipo de fator humano esteve envolvido [13]. As principais causas analisadas neste estudo foram "*Impairments*" que significa uma "desconexão" com a condução - tais como o uso de substâncias nocivas/álcool, cansaço ou sob algum tipo de emoção forte (tristeza, raiva, entre outras) - os erros de performance e os erros de julgamento na tomada de decisão perante determinadas situações reúnem os principais fatores de culpa humana tal como transcreve a figura 3.

Distracted	Error	Impaired	Prevalence
YES	YES	YES	3.4%
		NO	51.1%
	NO	YES	0.1%
		NO	13.7%
NO	YES	YES	2.7%
		NO	16.5%
	NO	YES	0.2%
		NO	12.3%
Total			100%

Figura 3 - Distribuição das causas de acidentes [5]

2.2.2 Condução Distraída

O mesmo estudo sobre "Distracted Driving" [14] refere os seguintes dados:

1. Um terço dos jovens inquiridos admitiu enviar mensagens enquanto conduzia.
2. Marcar um número de telefone aumenta a probabilidade de acidente em 6 vezes.
3. Enviar uma mensagem de texto aumenta a probabilidade de acidente em 23 vezes.

Estes são apenas alguns dados que referem o perigo da distração na condução. Não se limitando apenas ao uso do telefone, outras simples ações como comer ou beber, aplicar maquilhagem, alternar a estação de rádio ou o destino no GPS representam também fatores de distração que potenciam a probabilidade de acidente por retirarem a atenção da estrada e por consequência aumentam o tempo de reação do condutor na eventualidade de uma potencial situação de perigo iminente.

2.2.3 Álcool e Substâncias Nocivas

A condução sob o efeito de álcool ou substâncias nocivas que afetam o comportamento e lucidez do condutor é também uma das grandes causas de mortalidade nas estradas. Apesar de alguns países terem leis muito severas sobre o nível máximo de álcool aceitável dos condutores (em alguns países é tolerância zero), existem ainda muitos casos. Nos EUA, 29% dos jovens entre os 15 e os 20 anos que faleceram num acidente rodoviário estavam sob o efeito de álcool. [2]

2.2.4 Excesso de velocidade

Os jovens são mais propícios a conduzir em excesso de velocidade e em média circulam a distâncias menores do carro da frente, comportamentos estes que em caso de necessidade

reduzem a margem de tempo para o condutor agir. De acordo com o mesmo estudo [1], cerca de 35% dos condutores que perderam a vida em 2020 nos EUA circulavam em excesso de velocidade. De notar que os comportamentos de condução sob o efeito de álcool e o excesso de velocidade têm uma grande correlação.

2.2.5 Falta de uso de cinto de segurança

Por fim, outra das grandes causas mais referidas em estudos sobre o tema é a falta do uso de cinto de segurança. Os números publicados pelo "*Centres for Disease Control and Prevention*" referem que de todos os condutores jovens que morreram em 2020, 56% não tinha o cinto de segurança devidamente colocado.

A educação e formação dos jovens são fundamentais para reduzir o número de vítimas de acidentes rodoviários. Embora ações de sensibilização e diálogo sobre os perigos da condução possam contribuir para diminuir estes riscos, a falta de maturidade de alguns jovens pode dificultar a eficácia destas medidas.

2.3 Dados telemáticos na condução de um veículo

Cada vez mais ouvimos falar do conceito de *smart cars* e *autonomous driving*. Estas tecnologias estão cada vez mais presentes nas estradas e já existem mesmo carros capazes de percorrer longos e complexos percursos com total autonomia sem necessidade de intervenção humana.

É expectável que muito em breve todos os carros possuam sensores embutidos e com a possibilidade de comunicar estes dados em tempo real para os seus provedores, empresas de manutenção, seguradoras, condutores, e mesmo entre si. Em média, os novos carros possuem cerca de 400 sensores que permitem a captura do estado do ambiente em redor e no interior do veículo em intervalos de tempo de poucos milissegundos [15]. Observamos na figura 4 um exemplo dos típicos sensores presentes num carro atual bem como algumas das funcionalidades inteligentes possíveis de desencadear com os mesmos.

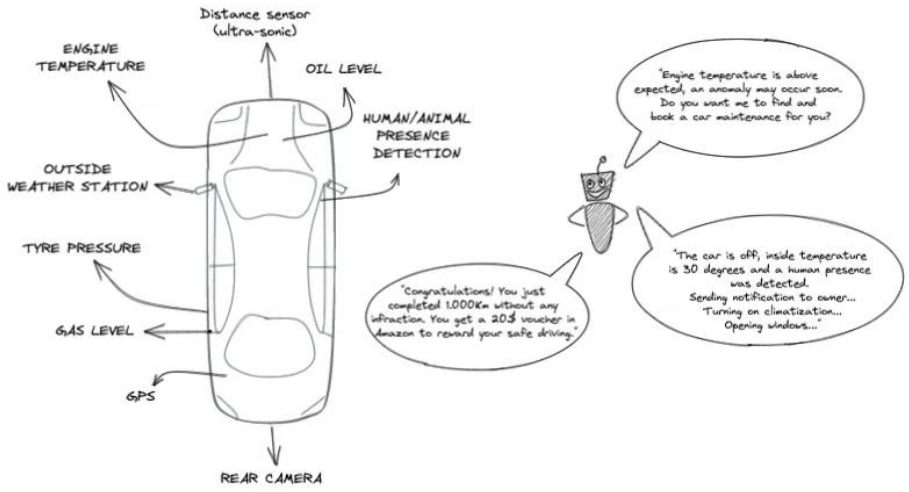

Figura 4 - Exemplo de sensores presentes num automóvel inteligente

Como visto anteriormente, as principais causas de desastres rodoviários são quase todas derivadas de erros humanos; a tecnologia presente nestes carros permite, em teoria, mitigar quase totalmente possíveis acidentes rodoviários por assegurar um tipo de condução segura, seja através do bloqueio dos limites de velocidade, restrição de circulação em zonas de maior perigo e mesmo a deteção de um tipo de condução mais agressiva ou sob o efeito álcool ou substâncias ilícitas. Os "reflexos" da tecnologia são também muito mais rápidos que os humanos já que necessitam apenas de poucos frames (milissegundos) para detetar uma situação de potencial perigo enquanto um humano demora, caso de estar pleno das suas capacidades, em média 600 milissegundos a reagir [16].

Já nos carros menos modernos, não há nenhuma maneira simples e direta de monitorizar e controlar a condução. Assim sendo, para atingir este objetivo é necessário recorrer a meios alternativos tais como dispositivos externos que permitam recolher os dados da condução e processá-los em tempo real ou algum tipo de aplicação móvel instalada num smartphone.

2.4 Aplicações práticas do uso de dados telemáticos

Durante as últimas décadas, inúmeros estudos foram realizados sobre Telemática na área de automóveis. Nesta secção irão ser apresentados alguns dos projetos mais relevantes bem como um resumo do estudo e as conclusões dos mesmos.

Identificar **anomalias no consumo de combustível** em grandes frotas é um fator crítico para otimizar o consumo e reduzir custos de empresas que possuam grandes frotas, tais como transportadoras, serviços de táxi, entre outros. Alguns dos projetos realizados nesta área passaram por exemplo pelo desenvolvimento de algoritmos de Deteção de Anomalias e acompanhados de um módulo *de "Model Explainability"* para explicar potenciais causas de

consumo 'anormal' de combustível e usar estas descobertas para gerar recomendações de otimização de combustível. Alguns dos padrões encontrados neste estudo foram o excesso de eventos de circulação em altas rotações e a velocidade média não otimizada [17].

Como referido no estudo anterior, o conceito de *Model Explainability* tem vindo a ganhar uma enorme importância pois a ideia de ter um modelo a tomar decisões em modo "black box" tem tendência para não ser aceite pelos mais céticos. Foi realizado um **inquérito a 37 Examinadores de Condução** com a finalidade de perceber a sua opinião relativamente a uma possível integração de diversos sensores para recolher e processar dados para ajudar no apoio à tomada de decisão sobre a passagem de um candidato. Embora céticos no que toca a ter um "sistema" que avaliasse se o candidato deveria passar ou não, a aceitação foi bastante positiva na eventual hipótese de ter dados estruturados e organizados com informação relevante e agregada que os ajudassem a tomar essa decisão. Foram também recetivos a outras ideias tais como os dados recolhidos durante as aulas de condução pudessem sugerir que um candidato já estaria ou não apto a ir a exame [18]. Isto traduz o paradigma atual em que há ainda uma grande relutância na confiança em Inteligência Artificial para tomar decisões autonomamente, mas uma aceitação generalizada no uso de AI na sintetização de informação que permita uma tomada de decisão na mesma feita por humanos mas com maior fundamento em dados.

Um estudo realizado a 100 condutores e recolhendo mais de 2 milhões de Km percorridos com diversos tipos de condução, tais como agressiva, sob fadiga, propensão a risco, infrações, entre outros e com o objetivo de **detetar padrões de condução e causalidade nos momentos precedentes a um acidente**, detetou como principais causas a distração na condução, a comunicação do condutor enquanto guia entre outros comportamentos de risco que potenciam a probabilidade de ocorrer um acidente [19]. Outra investigação que também procurava estudar a relação entre a qualidade da condução aquando da realização de uma tarefa em paralelo concluiu que condutores que estejam envolvidos numa outra tarefa em simultâneo tal como o uso do telefone, conversa com outro passageiro ou beber/comer são fatores que deterioram a qualidade da condução pois aumentam o tempo de reação a imprevistos [20].

Driver Identification é outro dos casos de estudo mais típicos estudados para aplicações dos dados recolhidos por sensores de telemáticas em veículos, sendo possível em apenas alguns segundos de condução detetar quem é o condutor (de um leque de possíveis condutores) com quase 100% de precisão [15]. Esta funcionalidade, exemplificada na figura 5, traz muito valor acrescentado a diversas indústrias tais como *Insurtechs*, uma vez que

processam os sinistros de forma automática e saber quem era o condutor no momento do acidente é crucial.

Figura 5 - Identificação do condutor e monitorização das expressões faciais. Fonte: www.valeo.com

Alguns projetos também tentaram avaliar a capacidade cognitiva do condutor através de um sistema de 'Eye Tracking' e sensores fisiológicos [21].

Outros dos principais casos de estudo são feitos pela indústria de seguros que ao ter dados sobre a condução dos seus clientes consegue não só aplicar modelos de Preço mais sofisticados [22] como também prever a frequência e severidade que um cliente poderá vir a representar de modo a poder avaliar o verdadeiro risco que o mesmo apresenta.

2.5 Barreiras e *Pitfalls*

Como já foi mencionado anteriormente, o registo de dados telemáticos de veículos apresenta uma enorme quantidade de benefícios. Nesta secção, vamos explorar algumas das implicações inerentes a estas tecnologias.

Em primeiro lugar, é de destacar as possíveis violações de privacidade dos utilizadores; os dados que são recolhidos têm um enorme valor comercial pois permitem treinar novos modelos de Machine Learning ou descobrir insights e podem por isso ser monetizados. A isto acrescentamos o facto de no caso de haver uma falha de segurança na aplicação, os dados podem cair nas mãos erradas e a partir desses dados pode-se facilmente identificar quem é o condutor com 97% de precisão e através dos registos de velocidade e ângulo do volante reconstruir todos os percursos efetuados e assim saber a localização em tempo real [23] [24].

Carros autónomos geram dezenas de fluxos de dados, podendo mesmo chegar a produzir perto de 1TB/hora [15] de dados em bruto. Se considerarmos que a tendência é cada

vez mais a adoção deste tipo de veículos, é expectável que o enorme **volume de dados gerados** se poderá tornar num desafio. Assim, torna-se importante a criação de plataformas de análise e processamento de dados poderosas e capazes de transformar TBs de informação bruta em poucos MBs de informação relevante.

2.6 Tecnologias para monitorização do estilo de condução

Nesta secção iremos observar que tecnologias temos disponíveis no mercado que nos permitam a captação de dados telemáticos através de sensores e os protocolos de comunicação.

2.6.1 Tipos de sensores utilizados

Se observarmos os carros recentes mais tecnológicos, podemos ver que vêm já equipados na com diversos tipos de sensores. Alguns dos mais comum são os LIDAR/RADAR que permitem medir a proximidade em torno do veículo, sensores do motor para monitorizar a temperatura, pressão e velocidade, sensores ultra sónicos, sensores de oxigénio e "Knock" que permitem não só uma maximização da eficiência do consumo de combustível, mas também a deteção prévia de anomalias no motor [25]. Sensores de peso para detetar passageiros, sensores de qualidade de ar na cabine, acelerómetros, giroscópios são apenas alguns exemplos destes sensores.

2.6.2 Protocolos de comunicação

Um dos principais pilares destas tecnologias são os protocolos de comunicação. São estes que garantem que os dados recolhidos pelos sensores são enviados para onde possam ser processados e os atuadores possam receber informação em tempo real.

Dos vários tipos de protocolos existentes, é relevante para este projeto a separação em dois grupos diferentes: os de pequeno alcance, que garantem uma comunicação local entre sensores, unidades de processamento e atuadores - *edge computing* - e os de longo alcance, que permitem comunicação com sistemas remotos e provedores de serviços remotos - *Cloud computing*.

Com o crescente número de sensores presentes nos carros, torna-se necessário um meio de comunicação wireless. No entanto estes protocolos – exemplificados na figura 6 - de alcance limitado têm de ser altamente eficientes no que toca a consumo de energia uma vez que tipicamente não há uma ligação à rede elétrica - são dispositivos wireless - e dependem de pequenas baterias para se manterem ligados.

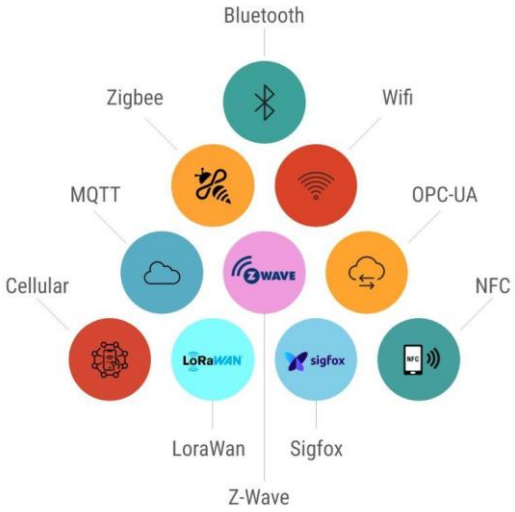

Figura 6 - Protocolos de Comunicação - Fonte: https://hashstudioz.com

Temos por exemplo tecnologia **RFID (radiofrequência)** ativa e passiva, sendo que a grande diferença é o facto de a ativa ter a sua própria fonte de energia e a passiva é alimentada por ondas de RF fornecidas por uma antena do dispositivo recetor, tornando esta segunda alternativa uma boa opção por ser bastante barata e eficiente, podendo ter uma autonomia de vários anos, apesar de apenas funcionar em zonas mais fechadas e com pouco alcance (poucos metros). Nos carros modernos todos os sensores estão embutidos e comunicam com uma unidade central (microcomputador ou MCU) por fio, garantindo uma comunicação praticamente sem falhas. Num estudo realizado na Carnegie Mellon University, os autores queriam estudar uma alternativa wireless para a rede de sensores e procederam à implementação de uma rede que comunicava entre si através de RFID. O estudo conclui que apesar de ser uma alternativa considerável, a localização da antena tem um grande impacto uma vez que se o sinal não for suficientemente forte para chegar ao recetor colocado no sensor, torna-se inviável a comunicação [26].

Bluetooth Low Energy (BLE) é uma derivação do famoso protocolo de comunicação Bluetooth de baixo consumo desenvolvido para pequenos dispositivos cujo consumo de energia tenha de ser muito eficiente. Este protocolo opera na banda de 2.4GHz e utiliza uma topologia de estrela onde todos os dispositivos estão conectados diretamente a um dispositivo central, habitualmente chamado de Hub [27].

Presente em muitos dos atuais telemóveis e utilizado por exemplo para pagamentos, a tecnologia **Near Field Communication (NFC)** é um protocolo de comunicação que atua na banda dos 13,56MHz e que possibilita de uma forma muito simples uma interação bidirecional entre dois dispositivos eletrónicos que estejam no máximo a poucos centímetros de distância.

Zigbee é um protocolo eficiente tanto ao nível de consumo de energia como na transmissão de dados, opera também na banda de frequências de 2.4GHz e diferencia-se pelo seu maior alcance ao utilizar uma topologia *Mesh*, onde todos os dispositivos podem comunicar entre si dispensando um *Hub* central para agilizar a comunicação e estendendo assim o range de comunicação.

Por fim temos também o protocolo **Long Range Wide Area Network (LoRaWAN)** que é um protocolo com base na técnica de modelação LoRa (Long Range) que lhe permite uma comunicação de longo alcance sendo bastante eficiente no consumo de energia. Este protocolo além de eficiente no consumo de energia apresenta também um excelente ratio de sinal-noise e uma forte resistência a interferência por usar um tipo de sinal 'wide', espalhado por uma grande banda de frequências. Oferece também uma grande segurança por encriptar o sinal de ponta a ponta e Gestão de Chaves de Acesso fazendo com que apenas os dispositivos autorizados consigam aceder aos dados [28].

Uma vez mais é importante referir que as várias soluções apresentam vantagens e desvantagens sendo que o(s) protocolo(s) a utilizar em cada projeto dependem sempre dos requisitos do mesmo.

2.7 Armazenamento de dados

Havendo largas centenas de alternativas de bases de dados, iremos agora estudar os principais tipos de Bases de Dados e as suas diferenças bem como as vantagens e desvantagens de cada um.

2.7.1 Dados estruturados e não estruturados

Bases de dados SQL contêm dados estruturados e usam um modelo relacional para organizar os dados, tornando-os uma boa alternativa para projetos que exijam queries complexas.

As alternativas *No SQL* são definidas tipicamente de um modo 'key-value' para organizar os dados e são consideradas mais indicadas para projetos que possam necessitar de escalabilidade, flexibilidade e sincronização em tempo real.

Em termos de performance, as ofertas NoSQL tais como Cassandra e MongoDB são melhores e mais escaláveis apesar de apresentarem mais falhas de segurança [29].

2.7.2 Real Time

As alternativas de bases de dados *Real-Time* são mais restritas, por isso é correto ponderar quais as vantagens que nos trazem. Um estudo conclui que estas são as mais

indicadas para projetos IoT uma vez que habitualmente estes envolvem grandes volumes de dados e requerem baixa latência, grande escalabilidade e tolerância a falhas [30].

2.7.3 Mobile

Ambas categorias de SQL e NoSQL oferecem alternativas para mobile ou *edge devices*. O facto de os recursos serem limitados faz com que as bases de dados tradicionais não obtenham a performance necessária e por isso surgiu a necessidade de se criar versões mais leves das mesmas. As suas principais vantagens face às bases de dados tradicionais são a otimização da performance minimizando o consumo de energia (problema que não se põe com bases de dados não mobile) e eficiência no armazenamento de dados uma vez que habitualmente em dispositivos edge e mobile é uma grande limitação [31] .

Algumas das ofertas mais conhecidas no mercado são o Firebase, Amazon DynamoDB, SQLite, MongoDB, Cassandra, InfluxDB, Azure Cosmos DB e Redis. Cada uma destas apresenta as suas vantagens e a sua escolha dependerá sempre dos requisitos do projeto.

2.7.4 Managed Services vs Open Source Software (OSS)

Uma grande parte das bases de dados seguem o modelo de OSS, ou seja, são gratuitas e acessíveis para todos os utilizadores. No entanto, as mesmas requerem uma infraestrutura com servidores, configurações de redes e acessos, políticas de privacidade e segurança, escalabilidade, etc. que têm de ser asseguradas pelo utilizador. Uma alternativa para contornar esta barreira técnica, são as ofertas de *Managed Services*, onde um provedor de Cloud disponibiliza uma versão destas bases de dados pronta a utilizar em que todas estas configurações e requisitos de infraestrutura são geridas automaticamente. Este serviço tem um custo habitualmente associado ao uso, sendo que habitualmente todas as opções oferecem um *tier* grátis para projetos experimentais.

Estas alternativas garantem muita eficácia na escalabilidade e disponibilidade (*availability*) e uma vez mais a sua escolha depende dos requisitos de cada projeto.

2.8 Machine Learning

A área de Machine Learning (ML) teve os seus primeiros desenvolvimentos em 1943 quando Walter Pitts e o neurocientista Warren McCulloch publicaram o primeiro modelo matemático de uma rede neuronal para criar algoritmos que imitassem o pensamento

humano. No final da década de 80 começou a ter um grande desenvolvimento com a criação de redes neuronais artificiais, mas sempre com uma grande limitação pois o poder computacional ainda era muito escasso e por isso o treino de um simples modelo poderia demorar vários dias.

2.8.1 Introdução e conceitos principais

Existem várias definições do que é Machine Learning, unindo-se numa linha comum que afirma que ML é uma área de estudo que visa o desenvolvimento de algoritmos e modelos estatísticos que permitem aos computadores aprender com dados [32]. Existem outros conceitos que apesar de diferentes são frequentemente confundidos e assim sendo, faz sentido referi-los, tal como apresenta a figura 7, e realçar quais as principais diferenças:

1. **Inteligência Artificial** é o ramo da Ciência de Computadores que tem como objetivo fazer com que as máquinas pensem como humanos [33].
2. **Data Science** é o processo de extração de conhecimento a partir de dados em diversos formatos estruturados (tabelas) e não estruturados, tais como áudio, imagem, texto [34].
3. **Machine Learning** é o estudo e desenvolvimento de algoritmos que melhoram automaticamente com a experiência. A grande diferença entre AI e ML é que a primeira procura criar máquinas/sistemas capazes de realizar tarefas que normalmente requerem inteligência humana enquanto que a segunda é uma ramificação da primeira que se foca especificamente no desenvolvimento de algoritmos que aprendem com dados [32].
4. **Neural Networks** é um tipo de algoritmos de Machine Learning modelado à semelhança da estrutura do cérebro humano. Consiste em várias camadas de neurónios artificiais conectados que são capazes de aprender e representar padrões complexos de dados [35].
5. **Deep Learning** é uma subárea de Neural Networks sendo que o que a distingue das restantes redes neuronais é o uso de múltiplas camadas de neurónios artificiais e de maior complexidade (*deep*) que permite a deteção de padrões muito mais profundos e complexos [36].

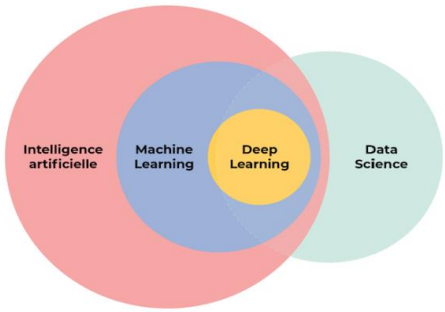

Figura 7 - Principais conceitos e interseção - Fonte: https://medium.com/

2.8.2 Tipos de aprendizagem de ML

Machine Learning, como referido anteriormente, consiste em criar algoritmos treinados com dados que consigam imitar o pensamento humano. Para isto, existem diversos modos de aprendizagem, sendo os mais comuns:

Supervised Learning

O tipo de aprendizagem mais comum, onde o modelo é treinado com dados com *label*. Podemos fazer a analogia com um estudo através de um conjunto de exercícios com solução. O modelo depois de aprender os exercícios (ao observar o que é pedido e a solução correspondente), é testado com outros exercícios que nunca "viu", mas que também é sabida a solução e desta forma podemos avaliar a performance do modelo [37].

Os principais **objetivos** (*tasks*) de *supervised learning* são a Regressão, onde o modelo tenta prever um número (como por exemplo, o preço de uma casa) ou Classificação (binária ou multi-classe) onde o modelo tenta prever uma classe (por exemplo se um pedido de crédito vai entrar em incumprimento (sim ou não).

Dentro destes algoritmos, destacam-se os modelos lineares tais como Regressão Linear, regressão logística, Naive Bayes, Support Vector Machines, k Nearest Neighbors, e modelos de árvores tais como Decision Trees, Random Forest ou XG Boost. O que os difere é o modo como estes modelos aprendem os dados sendo que o objetivo é o mesmo: aprender a identificar a *label*.

Semi-supervised Learning

Neste tipo de aprendizagem o modelo aprende com uma tabela em que apenas parte dos dados tem *label*. A ideia é primeiro treinar o modelo apenas com os dados que têm *label* e de seguida o próprio modelo vai classificar os restantes dados que não têm *label* [38].

Algumas das principais técnicas de semi supervised learning são por exemplo *Self-Training*, *Co-Training*, *Multi-view training*, *deep generative models* e *graph-based models*.

Unsupervised Learning

Neste tipo de aprendizagem o modelo aprende com dados sem nenhuma *label*, havendo diversos objetivos tais como encontrar padrões nos dados que permitam agrupar registos semelhantes [39]. As principais técnicas são:

1. **Clustering –** tem como objetivo criar grupos (clusters) de registos semelhantes. Um dos principais usos desta técnica é por exemplo criar grupos de clientes semelhantes para fazer ofertas ou marketing mais customizado.
2. **Dimensionality Reduction –** um dos principais problemas de ML é o chamado 'Curse of Dimensionality' onde o número de features (colunas) é tão grande que se torna difícil para o modelo detectar padrões. Assim, este algoritmo comporta-se como um sintetizador de informação que procura reduzir o número de colunas ao condensar a informação.
3. **Anomaly Detection –** é um dos tipos de aprendizagem mais frequente em IoT já que este tipo de técnica permite solucionar problemas relacionados com Deteção de Falhas ou Manutenção Preventiva utilizando dados de sensores em tempo real. Também conhecido como *outlier detection*, o objetivo destes modelos é identificar padrões diferentes da norma que poderão indicar um problema ou possível ameaça. Existem diferentes opções de modelos consoante os dados em que queiramos detectar anomalias (imagens, séries temporais, etc.) [40].

Reinforcement Learning

Uma das áreas mais promissoras de ML, é um tipo de aprendizagem em que um Agente aprende a tomar decisões ao interagir com um ambiente definido e com o objetivo de maximizar a função de *Reward* - definida pelo modelador [41].

Um dos casos mais conhecidos é o famoso jogo "Go", onde um modelo de ML aprendeu o jogo e superou o melhor jogador humano. Depois de uma primeira fase de Supervised Learning, onde o modelo aprendeu a jogar ao ser treinado com uma tabela de dados de profissionais de Go, deu-se uma segunda fase de *Reinforcement Learning* onde um agente foi treinado ao jogar contra si próprio. Assim, o modelo após milhões de iterações e de tentativas, aprendeu para cada estado (State) qual a melhor ação a tomar de forma a maximizar a *Reward Function*.

Transfer Learning

Apenas disponível em modelos de Redes Neuronais, *Transfer Learning* é uma técnica que permite que o conhecimento aprendido para desempenhar uma tarefa possa ser utilizado na aprendizagem de uma nova tarefa [42].

Os chamados "*pre-trained models*", são modelos que foram treinados tipicamente com grandes volumes de dados para desempenhar uma tarefa. Estes modelos podem posteriormente ser utilizados como base para se treinar para outras tarefas que estejam relacionadas. Um exemplo são os modelos de deteção de objetos em imagens; podemos utilizar um modelo que já tenha aprendido a detetar centenas ou até milhares de objetos em imagens e utilizá-lo como base para uma nova tarefa, como por exemplo, detetar um novo objeto que não lhe seja conhecido. Podemos utilizar uma analogia e ver esta nova tarefa como se se fosse uma especialização médica; supondo que queríamos ensinar uma nova técnica de cirurgia, seria sempre mais eficiente ensinar uma pessoa já com formação médica e experiência em operações, dado que o seu atual conhecimento seria uma base muito sólida para aprender esta nova técnica, ao invés de ter de formar uma pessoa de raiz que teria de aprender todas as bases de medicina e experiência de operações.

2.8.3 Automated Machine Learning

Automated Machine Learning (AutoML) é uma das áreas mais emergentes em Inteligência Artificial e tem como objetivo automatizar a criação de modelos preditivos, desde o pré processamento dos dados até á escolha do melhor modelo com otimização dos híper-parâmetros, sem que seja necessário qualquer tipo de conhecimento prévio [43].

Pipeline AutoML

Neste ramo de Auto ML temos ferramentas como o H2O ou AWS Autogluon que apenas necessitam de receber uma tabela de dados sem qualquer tipo de pré processamento, e são capazes de criar todo o pipeline de pré processamento, que inclui a otimização do tipo de dados de cada coluna, limpeza dos dados, preencher valores nulos, codificação de variáveis categóricas e engenharia de novas features e a componente de modelação, onde estas ferramentas vão testar diferentes famílias de modelos com diferentes híper-parâmetros e proceder a técnicas mais avançadas como *Stacking*, *Ensembles* e *Blending* que permitem a combinação de vários modelos.

***Neural Architecture search* (NAS)**

Este ramo de Auto ML foca-se na automatização do desenvolvimento da arquitetura de Redes Neuronais. Estas ferramentas procuram a melhor arquitetura para uma tarefa específica e para isto recorrem a algoritmos de otimização e Machine Learning para chegar ao melhor número de camadas, neurônios, funções de ativação, etc. Existem diversos tipos de NAS, tais como "*Reinforcement Learning-based NAS*", "*evolution-based NAS*", "*gradient-based NAS*" ou "*Bayesian optimization-based NAS*". Este processo é altamente complexo e requer muito poder computacional pelo que tipicamente é executado em provedores de serviços Cloud tais como GCP, Azure ou AWS [44].

2.8.4 Tiny ML

Com o recente avanço da tecnologia, temos atualmente microprocessadores (MCUs) que custam apenas alguns euros que são capazes de correr algoritmos de Machine Learning com bom desempenho. Dado que os seus recursos computacionais são limitados face ao enorme volume que os típicos modelos de Machine Learning requerem, nasceu o conceito de Tiny ML. Considerado o 'pai fundador' do Tiny ML, Pete Warden é uma das principais caras do Tensorflow Lite da Google, um dos pioneiros nesta área.

TinyML consiste no treino de modelos de Deep Learning e posterior compressão e otimização para que se tornem menos complexos e por isso mais 'leves'. Um modelo mais leve requer um menor poder de processamento para poder ser executado e por isso consome menos recursos computacionais e energia. As principais vantagens deste novo conceito acabam por ser uma resposta às principais barreiras dos modelos de Machine Learning comuns.

O facto de o próprio dispositivo realizar as previsões, garante uma decisão em tempo e real ao dispensar a comunicação com um servidor remoto diminuindo não só o risco de falha (no caso de haver uma interrupção ou falha na rede) mas também descendo drasticamente a latência na resposta uma vez que o modelo está onde os dados estão e por isso reduz também a quantidade de dados a serem transmitidos na rede. O baixo custo destes dispositivos torna possível o uso em várias indústrias. Ao serem processados localmente, os dados estão mais protegidos contra eventuais desvios tornando-os assim mais seguros.

Como referido anteriormente, o principal objetivo de TinyML é comprimir um modelo de Deep Learning para que este possa ser executado num dispositivo de IoT ou telefone. Existem várias técnicas para comprimir os modelos tais como a *Deep Compression* ilustrada na figura 8 e que consiste em *Pruning*, *Training Quantization e Huffman Encoding* aplicados sequencialmente ao modelo com o objetivo de reduzir o número de parâmetros usados [45].

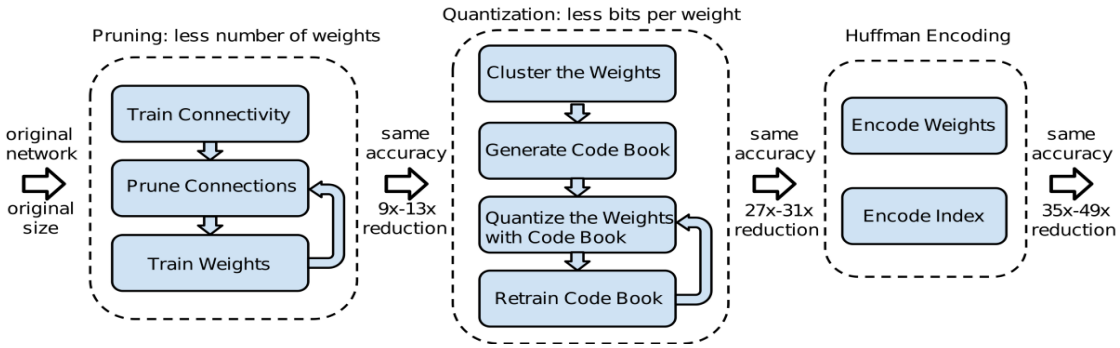

Figura 8 - Deep Compression [37].

É relevante mencionar que ao comprimir o modelo estamos a retirar complexidade e por isso a perder algum poder preditivo. Assim, e mais uma vez, cada projeto tem as suas especificidades e o nível de compressão terá que ser efetuado com base nos requisitos do mesmo. Alguns fatores a ter em conta na altura do planeamento são o poder de processamento do dispositivo onde o modelo vai correr, a facilidade de acesso à energia para definir quão eficiente tem que ser, o custo e tamanho do dispositivo e qual o nível de desempenho mínimo que o modelo tem que garantir.

De salientar que as principais *ferramentas* de desenvolvimento de Machine Learning já apresentam soluções simples que permitem selecionar qual o parâmetro que queremos otimizar, entre os quais performance, latência e tamanho.

As alternativas que permitem uma maior customização dos modelos são por exemplo o Tensorflow Lite e Pytorch mobile, as versões de Tiny ML de duas das maiores ferramentas de Deep Learning.

O Tensorflow Lite é uma plataforma open source de desenvolvimento de modelos de Tiny ML baseada na biblioteca Tensorflow e otimiza modelos de ML para que sejam corridos em pequenos dispositivos móveis ou smartphones de um modo eficiente com baixa latência e baixo consumo de energia.

Embedded Devices

Microcontroladores de uso geral, tais como o *esp8266* e *esp32* da Espressif, estão aptos para correr pequenos algoritmos de ML e custam poucos euros. Na figura 9 observamos algumas especificações destas placas de desenvolvimento.

	ESP8266	ESP32
		
MCU	Xtensa Single-core 32-bit L106	Xtensa Dual-Core 32-bit LX6 with 600 DMIPS
802.11 b/g/n Wi-Fi	HT20	HT40
Bluetooth	X	Bluetooth 4.2 and BLE
Typical Frequency	80 MHz	160 MHz
SRAM	X	✓
Flash	X	✓
GPIO	17	36
Hardware /Software PWM	None / 8 channels	None / 16 channels
SPI/I2C/I2S/UART	2/1/2/2	4/2/2/2
ADC	10-bit	12-bit
CAN	X	✓
Ethernet MAC Interface	X	✓
Touch Sensor	X	✓
Temperature Sensor	X	✓
Hall effect sensor	X	✓
Working Temperature	-40°C to 125°C	-40°C to 125°C

Figura 9 - Placas de desenvolvimento ESP - comparação. Fonte: https://www.kuongshun-ks.com

Alguns microcontroladores tais como o Edge TPU da google foram desenvolvidos especificamente para executar modelos de ML e são por isso mais eficientes para este propósito. Também o Arduino lançou uma versão direcionada ao uso de algoritmos de ML, o Arduino Nano 33 BLE Sense, que dispõe não só de um microprocessador capaz de correr algoritmos, mas também diversos sensores como podemos observar na figura 10.

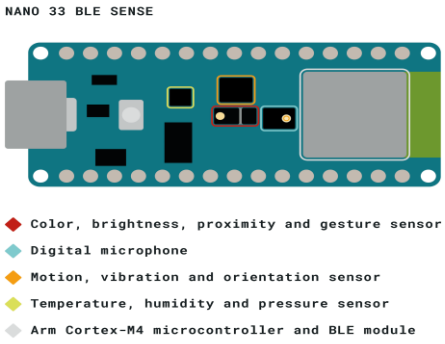

Figura 10 - Arduino Nano 33 BLE Sense. Fonte: https://gilberttanner.com

No que toca a plataformas para o desenvolvimento dos modelos de Tiny ML, temos plataformas low-code como o Edge Impulse que possibilita a integração diretamente com o dispositivo IoT permitindo assim a recolha de dados, treino de modelos e *deploy* dos mesmos nos dispositivos IoT.

2.9 Ferramentas para projetos IoT de provedores de serviços Cloud

Os principais provedores de serviços de Cloud oferecem uma vasta gama de serviços que cobrem todas as necessidades nas diferentes fases de um projeto de IoT. Existem módulos principais como o IoT Core da AWS, o Cloud IoT Core da Google Cloud Platform ou o IoT Hub da Microsoft Azure sendo que estes asseguram uma fácil integração dos dispositivos físicos com outros módulos que tratam por exemplo da comunicação entre os dispositivos IoT e a Cloud, protocolos de segurança, transmissão e armazenamento de diversos tipos de dados (tabulares, time series, imagens, etc.) e até mesmo soluções de Machine Learning otimizadas para pequenos dispositivos ou telemóveis. Na figura 11 observamos a típica arquitetura de sistemas IoT onde é vemos como os dados são processados desde os dispositivos móveis, os serviços Cloud onde são armazenados os dados e as aplicações que permitem a monitorização dos mesmos.

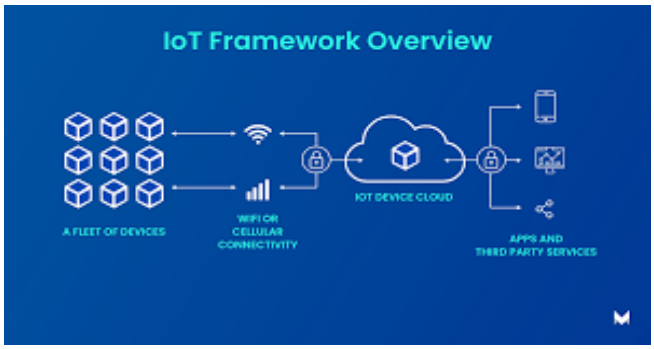

Figura 11 - IoT Framework. Fonte: https://blog.mobcoder.com

Os sistemas na Cloud permitem que sejam acedidos em qualquer lugar e ter modelos robustos de ML a atuar uma vez que o tamanho dos mesmos não é um problema. O sistema envia os dados para o servidor Cloud onde está alojado o modelo e o mesmo processa e devolve o resultado da inferência. No caso de o modelo e os dados recolhidos ficarem alojados no dispositivo *Edge*, evitamos vários problemas tais como privacidade e segurança e também conseguimos obter um sistema de maior confiança/estabilidade e de baixa latência, uma vez que pode correr offline e é independente de comunicações. Por outro lado, os recursos móveis são limitados pelo que os modelos teriam de ser comprimidos e otimizados perdendo assim alguma eficácia.

2.10 Aplicações práticas de ML na deteção de condução negligente através de sensores e câmara do telefone.

O objetivo deste projeto é detetar padrões de condução negligente em tempo real de forma a poder agir. Foram no passado estudadas diversas maneiras de detetar sonolência no condutor através de deteção de expressão facial [46], [47] ou piscar dos olhos [48], a deteção de condução sob o efeito de álcool [49] e também outros projetos para detetar fadiga ou cansaço [50].

Alguns projetos obtiveram bons resultados na deteção de condução agressiva ao detetar padrões tais como fortes acelerações e travagens [51] ou detetar excesso de velocidade utilizando os dados do acelerómetro e GPS de um smartphone [52].

2.11 Limitações das soluções disponíveis no mercado

Após uma pesquisa de soluções que ajudassem a monitorizar a condução e torná-la mais segura, a principal oferta são aplicações que permitem ao condutor uma interação com o telemóvel hands-free. Como visto anteriormente, o foco do condutor deve estar na estrada e não no telefone e por isso estas aplicações, ao proporcionarem um uso do telefone "hands-free", estão por outro lado a incentivar o uso do mesmo o que na mesma retira a atenção do condutor.

Ainda assim, após uma pesquisa em alguns blogs de parentalidade, algumas aplicações foram encontradas que permitem a monitorização em tempo real através de modelos conjuntos de regras que detetam padrões na condução e a classificam como segura ou insegura. Algumas destas apps são o TrueMotion, Road Ready, FamiSafe ou LifeSaver (Distracted Driving).

Quanto a soluções para Dash Camera, foram encontrados vários tipos de soluções diferentes. Na secção seguinte, vamos elaborar sobre os resultados de uma pesquisa destas soluções disponíveis no mercado.

2.11.1 Dash Cameras Tradicionais

As Dash Cameras tradicionais estão à venda em muitas superfícies comerciais sendo o acesso a estas relativamente fácil. No entanto, de acordo com o artigo escrito no "Vision Security" [53] , as mesmas apresentam algumas limitações:

Custo: As Dash Cameras tradicionais são tipicamente dispositivos autónomos que podem ser bastante caros, desde algumas dezenas de euros até várias centenas. O custo torna-se uma barreira significativa para a adoção generalizada, especialmente para aqueles que só precisam do dispositivo ocasionalmente.

Falta de análise em tempo real: As câmaras tradicionais são excelentes para gravação, mas não oferecem análise de imagem em tempo real, eliminando assim a hipótese de integrar ações preventivas.

Sem ações autónomas: Estes dispositivos não têm a capacidade de desencadear ações imediatas, como enviar alertas ou iniciar chamadas de emergência em iminente situação de perigo.

Finalmente, com o avanço das tecnologias smartphone, estas câmaras têm agora uma forte competição uma vez que qualquer smartphone possui uma câmara perfeitamente capaz de gravar o percurso de condução dispensando assim a compra de um dispositivo adicional.

2.11.2 Aplicações móveis Dash Camera

Após uma extensa pesquisa das ofertas disponíveis na Google Play Store (loja de aplicações do sistema operativo Android) para Dash Cameras, centenas de aplicações de foram encontradas, sendo a distribuição das classificações abaixo das taxas médias da Play Store tal como é ilustrado na figura 12 retirada da plataforma Statistica [54].

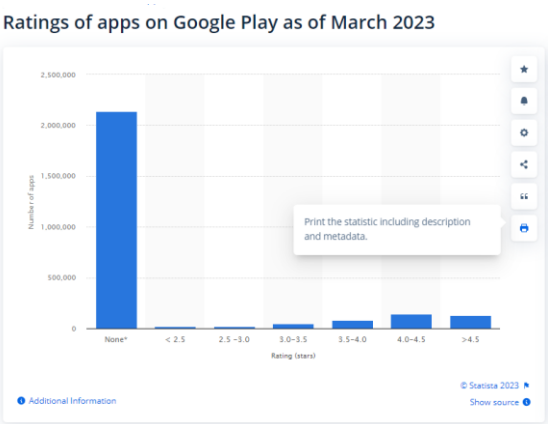

Figura 12 - Distribuição de aplicações da Play Store por classificação

Das 30 primeiras aplicações sugeridas para a palavra-chave "dashcam", nenhuma está na faixa > 4,5 e apenas 4 estão na faixa > 4. Isto pode indiciar que de um modo geral as soluções apresentadas não saciam as expectativas dos utilizadores.

Entre vários pontos de melhoria, as principais críticas às aplicações assentam sobre fatores tais como a estabilidade da aplicação, os problemas de armazenamento dos dados e a falta de funcionalidades além das básicas de gravação de imagem.

Além disso, estas aplicações trazem aos utilizadores possíveis preocupações com a privacidade dos dados, pois enviam dados recolhidos para a Cloud, levantando preocupações sobre a privacidade e segurança dos dados, uma vez que possibilita o rastreamento da localização dos utilizadores e reconhecer a matrícula de outros veículos facilmente [55]. Mesmo havendo aplicações que afirmam ter os dados encriptados, não se sabe que técnicas de criptografia [56] foram implementadas por isso ainda assim o facto de os dados do utilizador saírem do dispositivo representa um risco.

Estas aplicações geralmente exigem uma ligação à Internet para diversas funcionalidades, o que os torna a sua utilização menos viável em áreas remotas ou em situações onde a conectividade está comprometida. Também não têm a capacidade de desencadear ações de forma autónoma com base em análises em tempo real, reduzindo a sua eficácia como solução de segurança proactiva.

2.11.3 Dash Câmaras com tecnologia avançada em dispositivos dedicados

Nas últimas décadas, extensas investigações académicas foram conduzidas neste domínio. Os avanços contínuos e as reduções de custos em componentes de *hardware*, incluindo Raspberry Pi, Arduinos e vários sensores, abriram caminho para uma infinidade de desenvolvimentos de protótipos para dispositivos inteligentes. No entanto, a integração de múltiplos sensores, camaras, microfones, écrans, placas de comunicação wireless, baterias e aceleradores de computação, apresentam desafios complexos na criação de um dispositivo que seja sofisticado e fácil de usar. Dada a natureza centralizada em hardware destes produtos, introduz uma dimensão adicional de complexidade para a massificação destas soluções no mercado, necessitando de uma infraestrutura grande para a produção e manutenção destes dispositivos, com investimentos financeiros astronómicos necessários para o seu desenvolvimento. Na figura 13 vemos um exemplo de um projeto *Do it Yourself* (DIY) disponível num tutorial de um blog de tecnologia.

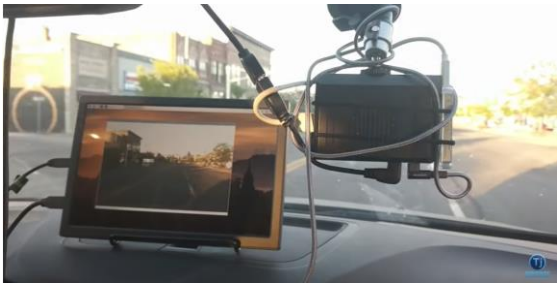

Figura 13 - Imagem do projeto TJ Free Dash Camera

Embora com algumas limitações este protótipo tenha já capacidades avançadas de IA tais como reconhecer objetos na estrada, a transição para um produto final "*user friendly*", produção em massa e o lançamento no mercado exigiria um investimento financeiro substancial. Esta barreira financeira é predominantemente responsável pelo facto de tais esforços de investigação permanecerem na fase de protótipo.

2.11.4 Dash Cameras modernas com tecnologia de IA

No domínio da segurança rodoviária, existem já Dash Cameras com fortes recursos de IA implementados, tendo como alguns exemplos disponíveis no mercado as soluções apresentadas pela Qualcomm e Lytx. Estes dispositivos de última geração estão equipados com recursos como deteção de objetos, identificação de condução distraída e mecanismos de alerta em tempo real. No entanto, a sua aquisição é muitas vezes acompanhada por um preço elevado e, para agravar o fator custo, há uma cobrança mensal recorrente para aceder a uma plataforma dedicada onde os utilizadores podem aceder aos seus dados. Estes dispositivos são desenvolvidos e direcionados principalmente para frotas de veículos empresariais, tais como empresas de distribuição ou de transportes de mercadorias e passageiros, o que os torna inadequados e em grande parte dos casos não estão sequer disponíveis para o cliente final.

Adicionalmente, o facto de ser um dispositivo externo, adiciona eventuais problemas e encargos financeiros no que toca a anomalias/avarias no aparelho e respetiva manutenção.

Concluindo, embora estas soluções inteligentes e com ações preventivas possam parecer promissoras como soluções de ponta para segurança rodoviária, o fator financeiro combinado do investimento inicial e das despesas contínuas, juntamente com potenciais preocupações de segurança e privacidade de dados, coloca estes dispositivos como uma escolha excessiva, terminando com o facto de a vasta maioria dos aparelhos descobertos na pesquisa de mercado não estarem disponíveis para o cliente final (utilizador comum) e apenas para uso comercial.

2.12Identificação das lacunas

A maioria dos dispositivos externos inteligentes para integrar nos veículos - com capacidades semelhantes às que serão desenvolvidas neste trabalho - é **direcionado para grandes frotas** com o objetivo de aumentar a eficiência de tempo e consumo e prevenir acidentes. Adicionalmente, estes dispositivos representam um **elevado custo e requerem manutenção** por serem **dispositivos físicos** o que torna o seu uso menos apelativo.

Para o consumidor final, com o efeito de monitorizar a condução, existem aplicações móveis mencionadas anteriormente que alcançam um bom desempenho nas funcionalidades de gravar o percurso e detetar alguns padrões de condução agressiva através de regras aplicadas aos valores lidos dos sensores. Algumas das principais falhas detetadas na maioria destas aplicações são principalmente os **problemas de ligação à internet** e o facto das aplicações dependerem disso para funcionar faz com que deixem de ser úteis em certas ocasiões.

Em nenhuma das aplicações averiguadas é possível obter os dados recolhidos dos sensores do telefone em bruto e em todas elas os dados são em algum momento enviados para a o provedor da aplicação aumentando assim os **problemas de segurança e privacidade** do utilizador.

Não sendo possível aceder ao código fonte destas aplicações, é difícil perceber se as as funcionalidades de deteção de movimento são desenvolvidas com algoritmos de Machine Learning ou com simples conjuntos de regras aplicados aos dados dos sensores. No entanto, em nenhuma aplicação foi observada a possibilidade de utilizar algoritmos de Computer Vision em tempo real para **detetar objetos na estrada** e **reconhecer a expressão facial** do condutor, nem com a possibilidade de **usar os dados dos sensores para detetar o movimento** e **comandos de voz para controlar a aplicação**. Por último, nenhuma das aplicações possui um modelo de **ações preventivas** desencadeadas pelos modelos e a possibilidade de **extrair os dados das viagens** de forma fácil.

Assim concluímos o capítulo de Estado da Arte onde foram sondadas as soluções possíveis para o problema apresentado bem como uma análise detalhada às principais lacunas das soluções existentes no mercado. Na secção seguinte irá ser discutida a proposta da solução apresentada bem como apresentado exaustivamente todo o desenvolvimento da mesma.

3

DESENVOLVIMENTO DO SISTEMA

Após realizado o Estado da Arte e detetadas algumas áreas de atuação ainda pouco exploradas, há ainda alguns pontos prévios a ponderar para decidir o rumo do projeto.

3.1 Descrição das Necessidades

A proposta para esta dissertação é a criação de uma solução inteligente que proactivamente alerte o utilizador de potenciais situações de perigo de modo a evitar possíveis acidentes. Esta solução irá utilizar diferentes algoritmos de Machine Learning que agem sobre os dados dos sensores do dispositivo móvel, imagens da câmara e comandos de voz do condutor. O foco do trabalho estará no desenvolvimento de uma prova de conceito que passa pela recolha, processamento e análise de dados bem como o desenvolvimento de algoritmos de Machine Learning que permitam desempenhar as funcionalidades de deteção de objetos e sinais de trânsito na estrada, utilização dos sensores para detetar condução agressiva, reconhecer o estado emotivo do condutor através da expressão facial e um modelo de classificação de áudio que permita ao condutor controlar a aplicação exclusivamente através de comandos de voz.

De modo a proteger a privacidade do utilizador, a aplicação deverá funcionar de forma totalmente autónoma sem que os dados saiam do dispositivo móvel aumentando também assim a segurança do utilizador.

3.1.1 Considerações relevantes

Nesta secção irão ser abordados alguns tópicos e tomadas as decisões para os problemas e pressupostos definidos. Irão ser realizadas algumas comparações entre diversas opções e a decisão será fundamentada com base nas vantagens e desvantagens que cada solução apresenta.

Edge vs. Cloud

A opção de os algoritmos de ML estarem alojados no dispositivo (Edge computing) traz enormes vantagens sendo que será a principal escolha pois evita problemas de privacidade e segurança e acima de tudo só assim é capaz de garantir uma resposta rápida (baixa latência) e independente de uma ligação à internet. Os modelos servidos em Cloud poderiam ser bastante mais robustos, mas com a desvantagem de requerer uma constante ligação à internet e a partilhar os dados do utilizador.

Mobile vs. Microcontrolador

Como visto anteriormente, a solução planeada para este projeto requer um conjunto extenso de hardware, incluindo não só os processadores e memórias como também sensores - acelerómetro e giroscópio - bateria, ecrã, câmara, microfone, flash, coluna, entre outros. Assim, foi feita uma pesquisa minuciosa a dispositivos móveis (smartphones) para avaliar se estão presentes todas as condições para serem utilizados neste projeto. Concluiu-se que a maioria dos telefones já têm a grande maioria dos recursos físicos necessários para a implementação desta solução.

Bases de dados

Segundo os artigos científicos mencionados no Estado da Arte, dados sequenciais (séries temporais) devem ser tratados de maneira especial. No entanto, esta solução está orientada para o consumidor final e por isso os dados gerados pelos modelos serão armazenados em ficheiros CSV de forma que o utilizador consiga, caso necessário, aceder facilmente e processar os seus próprios dados.

Sensores a utilizar

Como visto anteriormente, um telefone Android comum tem pelo menos os sensores mais básicos de acelerómetro, giroscópio, GPS, câmara e microfone que são uma base sólida para realizar este projeto. Futuramente os outros sensores poderão ser equacionados para o treino de modelos mais sofisticados.

Supervised vs. Unsupervised Learning

Inicialmente serão recolhidos dados dos sensores e processados para que se possa desenvolver algoritmos supervisionados. Futuramente, tal como será explicado na secção seguinte, serão considerados outro tipo de modelos não supervisionados na deteção de acidentes rodoviários.

Plataforma: Android vs. iOS

Representando 70% da do mercado de smartphones [45], o sistema operativo Android foi o escolhido não só pelo tamanho, mas também pela maior facilidade de desenvolvimento de aplicações por ser menos restrito que iOS. Mais de 80% dos telemóveis Android têm pelo menos sensores giroscópio, acelerómetro, câmara, GPS e microfone pelo que foi considerado o primeiro alvo para esta prova de conceito. Futuramente será equacionado o desenvolvimento de uma versão para iOS.

Traditional ML vs Deep Learning (TinyML)

Os primeiros desenvolvimentos de modelos serão feitos num computador e para prova de conceito poderão ser usados modelos tradicionais de ML, tais como regressões lineares ou Support Vector Machines (SVM) ou modelos de árvores por serem mais rápidos de treinar e perceber de uma forma rápida qual o poder preditivo dos dados recolhidos.

Numa fase posterior do projeto, de modo a poder testar os modelos desenvolvidos em tempo real, irão ser treinados modelos de Redes Neuronais e posteriormente comprimidos para que possam ser utilizados num dispositivo móvel com recursos de processamento mais limitados.

3.1.2 Definição das pré-condições

De modo a poder executar o projeto com foco no que é proposto, algumas assunções serão feitas e por isso os problemas derivados das mesmas não serão considerados.

O modo como os dados foram recolhidos num ambiente altamente controlado e pouco diversificado, acaba por trazer a possibilidade de os modelos não generalizarem da forma pretendida.

Para contornar isto o modelo teria de ser treinado recolhendo dados de múltiplas pessoas a conduzir diferentes carros para conseguir uma melhor generalização. Os modelos foram treinados apenas com um condutor e um carro e desta forma, para averiguar a performance no mundo real dos modelos, apenas será testado nas mesmas condições em que foram gerados os dados.

Relativamente ao dispositivo utilizado para a recolha de dados, será utilizado apenas um modelo de smartphone e a solução será avaliada no mesmo modelo.

No capítulo dos resultados irá ser feito um teste que medirá a capacidade de generalização do modelo, isto é, quão se degrada a performance dos modelos ao mudar variáveis como o veículo, o condutor, o smartphone ou as condições atmosféricas.

3.2 Requisitos e Use Cases da Aplicação

Para desenvolver as funcionalidades inteligentes, alguns Use Cases – listados na tabela 1 - vão ser investigados e analisados para serem possivelmente utilizados na prova de conceito das capacidades inteligentes da aplicação:

Caso de estudo	Descrição	Tipo de dados
C.E. 1	Deteção do grau de perigo de circulação por zona	Sensores, GPS
C.E. 2	Deteção de condução agressiva e colisão	Sensores, GPS
C.E. 3	Deteção de condução sob o efeito de álcool ou outras substâncias nocivas	Sensores, Imagens
C.E. 4	Deteção de Distracted Driving e cansaço	Sensores, Imagens
C.E. 5	Deteção de sinais de trânsito e luminosos	Imagens
C.E. 6	Deteção de entidades na estrada (peões, carros, motas, bicicletas, etc.)	Imagens
C.E. 7	Controlo de aplicação por comandos de voz	Áudio

Tabela 1 - Casos de estudo

De forma a se desenvolver uma solução que vá de encontro à visão, é de extrema importância a correta definição dos requisitos funcionais a implementar para que a mesma possa ser desenvolvida com coerência e elevado desempenho. Assim, após uma análise intensiva das necessidades do projeto, foi elaborada na tabela 2 a lista de Requisitos Funcionais.

ID	**Requisito Funcional**	**Descrição**
R.F.1	Padrões de Condução	A aplicação deverá detetar padrões de condução agressiva
R.F.2	Deteção de Objetos na Estrada	A aplicação deverá identificar Objetos e outras entidades relevantes na Estrada
R.F.3	Reconhecimento da Expressão do Condutor	A aplicação deverá permitir o Reconhecimento da Expressão Facial do Condutor
R.F.4	Assistente de Bordo	A aplicação deverá ter um Assistente de Bordo inteligente capaz de reconhecer comandos de voz
R.F.5	Registo	A aplicação deverá permitir o Registo de Utilizador e Login e também a possibilidade de uso sem registo
R.F.6	Processamento em Tempo Real	A aplicação deverá processar todos os dados em tempo real
R.F.7	Localização	A aplicação deverá indicar a Localização e coordenadas do dispositivo
R.F.8	Velocidade	A aplicação deverá indicar a velocidade atual do veículo

R.F.9	Alertas	A aplicação deverá poder desencadear alertas automáticos ao condutor em situações de perigo iminente
R.F.10	Emergência	A aplicação poderá desencadear uma chamada de emergência
R.F.11	Gravação do percurso	Será possível proceder à gravação do percurso efetuado
R.F.12	Customização	Possibilidade de o utilizador parametrizar os modelos e as ações a serem desencadeadas
R.F.14	Gravação dos dados	Registo dos resultados dos modelos em ficheiros CSV
R.F.15	Adaptação	Diferentes níveis de complexidade dos modelos disponibilizados para cada tarefa
R.F.16	Feedback	Possibilidade de o utilizador enviar feedback ou reportar bugs

Tabela 2 - Requisitos Funcionais

Adicionalmente a tabela 3 demonstra os Requisitos Não Funcionais a serem tidos em consideração no desenvolvimento da aplicação:

ID	**Requisito Não Funcional**	**Descrição**
R.N.F.1	Privacidade	Todos os dados devem ser processados localmente de modo a assegurar a total privacidade e segurança do utilizador
R.N.F.2	Baixa Latência	Os modelos devem apresentar uma latência de pelo menos 2 FPS de modo a produzir os alertas antes de o condutor detetar o perigo
R.N.F.3	Funcionamento offline	A aplicação deverá ser 100% funcional mesmo em áreas sem ligação à internet
R.N.F.4	UI apelativa	A aplicação deverá ter uma Interface de Utilizador intuitiva e de fácil controlo
R.N.F.5	Escalabilidade	A aplicação deverá permitir escalabilidade independentemente do número de utilizadores
R.N.F.6	Fiabilidade	A aplicação deverá ser fiável sob diversas condições assegurando sempre um desempenho consistente
R.N.F.7	Eficiência	A aplicação deverá ser eficiente de modo a ter um bom desempenho em qualquer smartphone sem consumir demasiados recursos e bateria
R.N.F.8	Compatibilidade	A aplicação deverá ser compatível com uma vasta gama de smartphones Android com diferentes versões
R.N.F.9	Customização	A aplicação deverá permitir a cada utilizador moldar a mesma às suas necessidades
R.N.F.10	Interoperabilidade	A aplicação deverá permitir de forma simples a partilha dos dados gerados com outras aplicações
R.N.F.11	Atualizações	A aplicação deverá suportar atualizações do código e dos modelos de forma fácil e segura

Tabela 3 - Requisitos Não Funcionais

3.3 Metodologia de Trabalho

Após feito um estudo e levantamento das necessidades, na secção anterior foram elaborados os requisitos funcionais que irão ser desenvolvidos durante a execução do projeto. Para um bom planeamento e estruturação, foram necessárias a adoção de algumas ferramentas de gestão de projetos.

O JIRA, apresentado na figura 14, foi a opção escolhida como software de gestão de projeto. Seguindo uma metodologia *Agile*, o projeto será faseado em *'sprints'* de 2 semanas. Cada sprint tem um objetivo (o 'sprint goal') e são atribuídas tarefas a serem desenvolvidas durante o mesmo. à medida que o projeto se vai desenvolvendo, novas tarefas podem surgir e estas deverão ser adicionadas ao *'backlog'* onde futuramente serão atribuídas a um sprint. No final de cada sprint é feita uma retrospetiva para averiguar se o projeto continua no rumo certo ou se é necessário fazer alterações e também ver o que correu bem e o que pode ser melhorado.

Github, apresentado na figura 15, será usado como repositório de código e controlo de versões. Desta forma o código do projeto irá estar mais estruturado, seguindo uma política de *atomic commit*, fazendo a ligação entre uma tarefa e o código desenvolvido para a execução da mesma.

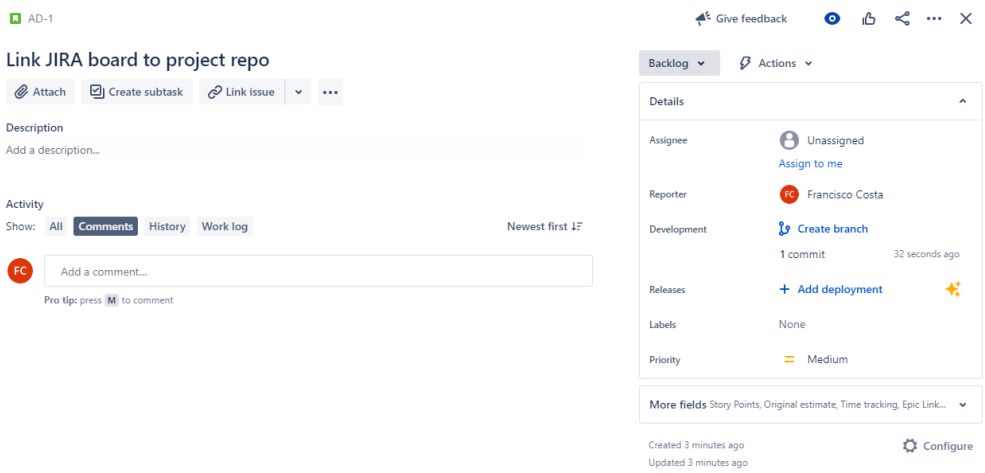

Figura 14 - Exemplo de uma tarefa no JIRA

A integração destas duas plataformas é crucial para um desenvolvimento eficiente de código, controlo de versões, organização/estruturação do código e documentação de suporte.

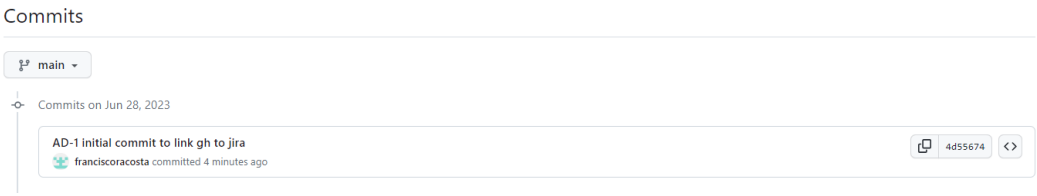

Figura 15 - Exemplo de um commit associado a uma tarefa do JIRA.

Nas figuras anteriores vemos como a tarefa criada no JIRA *'Link JIRA to project repo'* foi associada ao commit *'AD-1 initial commit to link gh to jira'*. Desta forma, poderá no futuro ser consultada uma determinada tarefa e aceder diretamente aos commits que contêm o código desenvolvido para a execução da mesma.

3.4 Arquitetura do Sistema

A arquitetura proposta foi definida com base nos pressupostos e assunções mencionados nas seções anteriores e pode ser observada na figura 16.

1. A recolha de dados será feita através de um dispositivo móvel (sensores e imagens).
2. Os dados serão armazenados num servidor do Firebase (Google).
3. O processamento dos dados e treino dos modelos será feito num ambiente de python. num computador ou provedor de serviços Cloud.
4. A análise dos modelos e benchmark dos resultados será feita num computador ou provedor de serviços Cloud.
5. Os modelos depois de treinados e comprimidos serão armazenados no Firebase.
6. O teste dos modelos em ambiente de produção será feito num dispositivo móvel descarregando o modelo armazenado no Firebase.

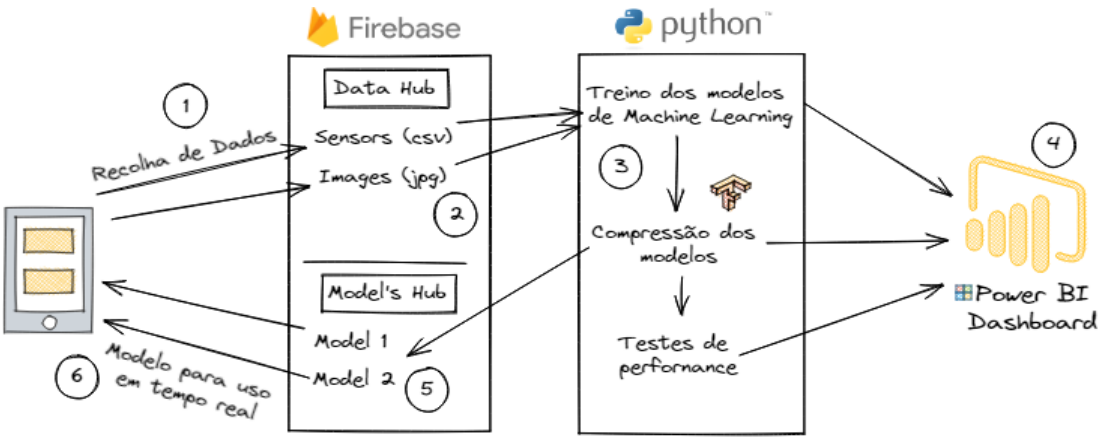

Figura 16 - Arquitetura Proposta

Firebase foi a ferramenta escolhida para armazenar dados e modelos num servidor que possa ser acedido tanto num computador como na app de forma simples e eficiente. Aqui serão guardados os modelos treinados onde serão posteriormente descarregados na aplicação.

A proposta de solução de Dash Camera é conceptualizada como uma aplicação móvel, projetada para encapsular as funcionalidades das soluções mais avançadas com AI integrada, enquanto aborda sistematicamente as limitações e lacunas acima mencionadas. Os pilares fundamentais da visão da solução são:

Democratização: Ao aproveitar a infraestrutura e recursos existentes nos smartphones ou ao reutilizar dispositivos inutilizados, evitamos a necessidade de aquisição de equipamentos dedicados e de elevado custo.

Versatilidade: Para além da função básica de gravação, a solução apresentada está dotada de capacidades analíticas em tempo real, facilitadas por algoritmos de ML locais. Esta abordagem multifacetada amplia as funcionalidades para incluir uma tomada de decisão autónoma e análises abrangentes de viagens, como deteção de sinais de trânsito, reconhecimento do estado emocional do condutor, identificação do estilo de condução e um assistente de bordo inteligente controlado por voz.

Processamento na Periferia (Edge Computing): Através do design meticuloso de modelos de ML eficientes em termos de recursos e adaptados para dispositivos móveis, garantimos que os dados permaneçam sempre no dispositivo. Esta estratégia facilita o processamento em tempo real dos dados com baixa latência, eliminando a necessidade de transmissão externa de dados, reforçando assim a privacidade e a segurança dos dados sem estar dependente da ligação à Internet.

Proatividade: A nossa solução é competente em desencadear autonomamente protocolos de segurança com base em *insights* analíticos - tais como a deteção de sinais de trânsito, estado emocional do condutor ou agressividade na condução - em tempo real. Isto inclui o envio de alertas por SMS e *WhatsApp*, a realização de chamadas de voz para contactos de emergência ou a emissão de avisos auditivos e visuais, elevando assim a dimensão proativa da segurança rodoviária.

Acessibilidade: A nossa solução, compatível com o sistema operativo Android, está facilmente acessível através da Play Store. Esta abordagem direta ao consumidor evita as complexidades e os custos associados à aquisição de *hardware* e consequentes taxas recorrentes de subscrição de software. O facto desta solução ser apenas *Software* retira muitos entraves à sua adoção e evita todos os problemas inerentes ao *Hardware* mencionados anteriormente.

Dadas as características e vantagens descritas para a nossa aplicação Dash Camera como ponto de partida, vamos proceder ao passo seguinte que trata a recolha dos dados que serão utilizados no treino dos modelos.

3.5 Criação dos Datasets

A qualidade de um modelo de Machine Learning é diretamente proporcional à qualidade dos dados em que é treinado. Para a nossa solução, fizemos uma seleção cuidadosa dos dados a utilizar, adaptados a cada tarefa específica de ML. Na continuação desta seção irá ser relatado o processo de aquisição e criação dos *datasets*.

3.5.1 Deteção de Objetos na Estrada (DOE)

Esta tarefa implica reconhecer objetos comuns presentes na condução diária. Várias fontes de dados foram usadas combinando conjuntos de dados públicos com imagens extraídas da internet para mostrar como podemos, por um lado, aproveitar conjuntos de dados públicos já anotados, como também podemos introduzir dados customizados quando não estão disponíveis as imagens e anotações pretendidas.

O dataset escolhido para deteção de objetos chama-se 'Road Sign Detection', disponibilizado publicamente na plataforma Kaggle e inclui imagens anotadas de 4 categorias de objectos: 'crosswalk', traffic_light, 'stop' e 'speed_limit'. Este dataset tem imagens capturadas sob diversas condições ambientais – luz do dia e noite – em diversos ambientes urbanos e suburbanos.

Adicionalmente, imagens personalizadas foram extraídas da Internet e fotografadas pelos autores para complementar o dataset do Kaggle. As imagens obtidas continham quase 200 amostras de 'bikes' e 12 amostras de 'radares' que foram anotadas com "*bounding boxes*" no formato Pascal VOC através do software LabelImg, tal como é ilustrado na figura 17.

Figura 17 - Anotação de uma 'bike' numa imagem recolhida da internet

Especificamente, para cada imagem anotada, é gerado um ficheiro XML que contém informações detalhadas, como dimensões da imagem, classes de objetos e coordenadas xmin, ymin, xmax, ymax das bounding boxes.

3.5.2 Reconhecimento de Expressão Facial (REF)

Para detetar o rosto do condutor e reconhecer a sua emoção, foi feita uma primeira abordagem utilizando apenas dados públicos disponíveis. O dataset base para reconhecimento de expressões faciais foi obtido também da plataforma Kaggle, denominado por FER 2013 e que oferece uma ampla variedade de expressões faciais, tais como feliz, irritado, triste, surpresa, neutro, medo e nojo.

Sendo que estas expressões faciais não vão de encontro ao que é pretendido para esta solução, uma segunda abordagem será testada usando imagens recolhidas de amigos e familiares realizando as expressões faciais específicas necessárias para o projeto de forma a enriquecer o dataset com expressões mais relevantes.

Tanto a primeira como a segunda abordagem requerem o uso de algoritmos de ML para detetar a expressão facial. Outras abordagens que não requerem Machine Learning, tais como rastreamento ocular [57] poderiam ser usados como alternativa, mas não serão consideradas para este trabalho.

3.5.3 Classificação de Movimento (CM)

Os dados dos sensores foram recolhidos durante uma sessão de condução num ambiente controlado onde as classes específicas a serem detetadas poderiam ser realizadas múltiplas vezes com segurança. Estas classes são **'idle', 'driving', 'hard acceleration' e 'hard brake'.**

Dados os requisitos exclusivos para a obtenção de dados de sensores, foi desenvolvida uma simples aplicação que permite a recolha de dados de sensores do telefone em tempo real, com cada registo de dados consistindo num *array* de 59 valores de sensores mapeados com a classe do movimento definida e armazenados num ficheiro CSV, sendo posteriormente feito o upload para o Firebase para ser processado e utilizado no treino dos modelos.

Para criar as amostras, tiveram de ser definidos dois parâmetros que irão ser explicados com mais detalhe na secção de Treino dos Modelos:

1. **Time Step:** Número de registos que formam uma amostra (sinal discreto).
2. **Time Interval:** Duração entre cada registo.

Efetuados alguns testes, e dado que todos os tipos de movimentos têm de partilhar estes mesmos parâmetros, concluiu-se que os melhores valores eram um Time Step de 20 pontos e um *Time Interval* de 100 milissegundos, o que significa que vão ser recolhidos dados dos sensores 10 vezes por segundo (a cada 100 ms), perfazendo uma duração de cada evento de 1 segundo, tal como é ilustrado na figura 18.

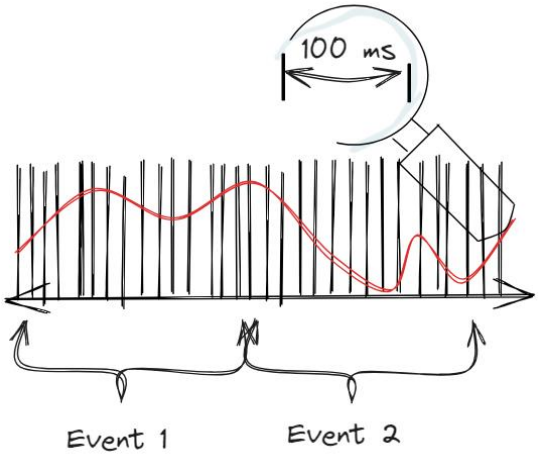

Figura 18 - Explicação dos pontos de dados, intervalo de tempo e eventos.

Assim, **com o apoio de um familiar de forma a garantir a segurança**, foram recolhidas várias amostras de cada um dos tipos de movimentos durante a condução de um veículo.

3.5.4 Reconhecimento de Comandos de Voz (RCV)

Para este efeito, não foram recolhidos dados personalizados nem usados datasets públicos. Sendo que o que se pretende é mostrar que é possível controlar a aplicação com comandos de voz, um modelo pré treinado com um subconjunto do dataset de Command Speech [58] que contém amostras de 8 comandos: "down", "go", "left", "no", "right" , "stop",

"up" e "yes" e outro dataset contendo ruído de fundo. Estes comandos serão usados para realizar ações como iniciar e parar a gravação ou ligar para o contacto de emergência.

3.6 Modelos Machine Learning

3.6.1 Abordagem de modelação

Nesta seção, o objetivo principal não passa por desenvolver os modelos mais avançados ou de última geração, mas sim demonstrar a viabilidade de treinar e otimizar modelos que possam servir aplicações em dispositivos móveis. Como referido anteriormente, até os modelos otimizados para dispositivos móveis requerem bastante poder de computação para poderem correr com um desempenho aceitável. Com isto, pretendemos desenvolver modelos mais eficientes, com um bom equilíbrio entre precisão e eficiência computacional, garantindo que os modelos possam funcionar sem falhas e em tempo real em qualquer dispositivo Android.

No processo de seleção de uma ferramenta apropriada para o desenvolvimento dos modelos, consideramos três das opções Open Source mais robustas disponíveis: PyTorch, Transformers e TensorFlow. De seguida iremos fazer uma breve introdução a cada uma destas opções bem como realçar as vantagens e desvantagens e terminar com a decisão e fundamentação da escolha efetuada.

Transformers

Desenvolvida maioritariamente pela Hugging Face, esta ferramenta permite uma fácil utilização da arquitetura de modelos que partilha o mesmo nome (Transformers) e que é considerada das mais eficientes tanto para tarefas de Computer Vision como NLP e Áudio. Esta fornece um enorme *hub* de modelos pré-treinados, facilitando o *fine-tuning* de modelos para tarefas específicas. A mesma oferece suporte a back-ends PyTorch e TensorFlow.

O foco principal desta framework está em modelos de grande escala, e o suporte para conversão para modelos leves ainda está pouco desenvolvido. Após algumas experiências efetuadas, foi possível treinar um modelo e proceder apenas a parte da otimização sendo que devido a falta de documentação e suporte dificultou a utilização deste framework.

TensorFlow

Desenvolvido pela Google, o TensorFlow é uma ferramenta madura com soluções robustas para desenvolvimento em dispositivos móveis. O formato TensorFlow Lite (TFlite) é altamente compatível com o sistema operativo Android, garantindo uma inferência do modelo

eficiente no dispositivo. A biblioteca de python "*tflite_model_maker*" torna o processo de modelação extremamente intuitivo e agiliza todo o processo de desenvolvimento do modelo, desde a preparação dos dados até ao treino, otimização e exportação do modelo para o formato TFlite.

PyTorch

Desenvolvido pela Meta AI (antigo Facebook) e conhecido pela sua sintaxe intuitiva, e frequentemente adotado como solução para modelação na comunidade de investigação académica.

No entanto, embora ofereça uma solução de otimização para dispositivos móveis através da plataforma PyTorch Mobile, pode não ser tão simples ou otimizado para implementação em Android como o TensorFlow. Foi também testado a framework PyTorch Lightning para treinar modelos e convertidos para tflite, no entanto, sem sucesso uma vez que o objecto do pré processador não pode ser convertido para um formato compatível com mobile e deste modo esta lógica de preparação dos dados teria de ser replicada em Java/Kotlin na aplicação. Foi também testado um package chamado Exporters que auxilia na conversão de modelos Transformers para formatos compatíveis com mobile, mas o mesmo não suporta ainda tflite, apenas MLcore.

A combinação de compatibilidade e facilidade de uso tornou o TensorFlow a escolha ideal para os objetivos do nosso projeto. Os pipelines de Deteção de Objetos na Estrada, Reconhecimento de Expressão Facial e Reconhecimento de Comandos de Voz foram desenvolvidos com a framework "TFlite Model Maker". O modelo de Classificação de Movimento foi desenvolvido com Tensorflow/Keras e posteriormente convertido para TFlite.

Com as tarefas e ferramentas definidas, é fundamental decidir qual a métrica que será utilizada para averiguar a qualidade dos modelos. Escolher a métrica certa é fundamental para avaliar o desempenho real do modelo para a tarefa para a qual foi treinado. Alguns fatores a serem considerados são:

1. **Tipo de problema:** Existem métricas específicas para classificação, regressão ou ranking.
2. **Requisitos de aplicação:** Dependendo da aplicação e da tarefa, o objetivo pode passar por minimizar os Falsos Positivos, os Falsos Negativos ou tratar ambos com igual importância. Deste modo podemos escolher as métricas de otimização do modelo que melhor representem o problema.
3. **Desequilíbrios das classes:** Se o dataset tiver um desequilíbrio de classes significativo, existem métricas como F1-score ou Mean Average Precision que podem ser mais informativas.

4. **Interpretabilidade:** Algumas métricas são mais fáceis de explicar e compreender do que outras. Por exemplo, a precisão ou *Mean Average Error* é mais interpretável para não especialistas do que AUC-ROC ou RMSE.

Cada modelo terá sua própria métrica e a sua escolha será detalhada e fundamentada na próxima seção.

Os modelos serão treinados numa instância EC2 na AWS - c6a.8xlarge - com 32 vCPUs e 64 Gb de RAM, tal como mostra a figura 19 – que tem um custo aproximado de 1,30$/h. Tendo em conta os tempos de treino, este custo será considerado marginal e não será tido em consideração para a tomada de decisão na escolha dos modelos

Compute Specs

Group	Type	Cost/Hr	Cores	Memory	Storage	I/O
Core	c6a.8xlarge	$1.31	32	64 GiB	EBS only	12500 Megabit

Figura 19 - Especificações da instância de treino dos modelos

Após uma introdução à abordagem de modelação, iremos agora aprofundar os *pipelines* de treino dos diversos modelos.

3.6.2 Classificação de Movimento

A capacidade de classificar com precisão o movimento do veículo é um recurso fundamental nesta aplicação. Esta seção foca-se no desenvolvimento do modelo para identificar quatro tipos distintos de movimentos: "Idle", "Driving", "Hard Acceleration" e "Hard Break" de modo a poder avaliar o nível de agressividade na condução.

Na figura 20 podemos observar o pipeline de treino do modelo usado para o modelo.

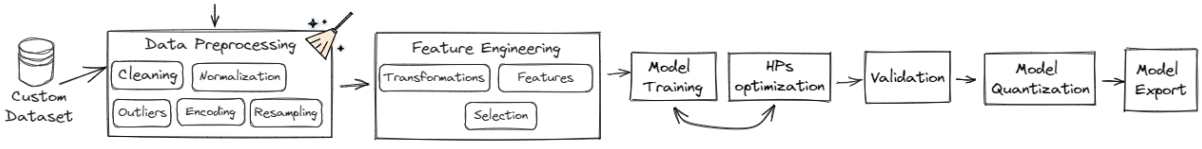

Figura 20 - Pipeline de treino do modelo de classificação de movimento do sensor.

1. Utilização de dados do sensor

Para efeitos de classificação de movimento, recorremos a uma combinação de acelerómetros e giroscópios disponíveis nos smartphones. Os acelerómetros fornecem dados sobre aceleração linear ao longo dos eixos x, y e z, enquanto que os giroscópios oferecem medições da velocidade angular também sobre estes 3 eixos tal como é ilustrado na figura 21. Estes sensores capturam o estado dinâmico do veículo, permitindo a identificação de vários tipos de movimento. Sensores mais avançados foram excluídos para priorizar um acesso mais abrangente à aplicação, uma vez que os mesmos podem não estar presentes em dispositivos móveis mais antigos.

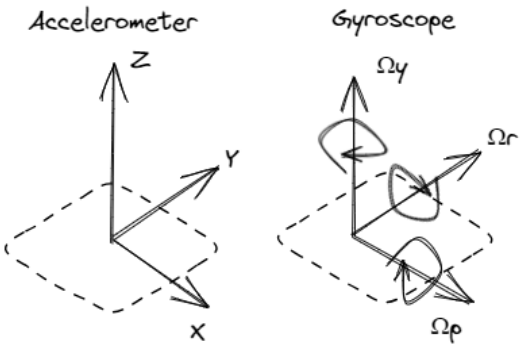

Figura 21 - Eixo do acelerómetro e giroscópio

2. Análise Exploratória de Dados

É importante, antes de iniciar a modelação, compreender os dados obtidos dos sensores. Na figura 22 vemos uma amostra dos dados com apenas um subconjunto de 6 dos 59 valores dos sensores e a classe de movimento correspondente.

	accelerometer_x	accelerometer_y	accelerometer_z	gyroscope_x	gyroscope_y	gyroscope_z	motion_label
0	9.547063	0.043669	3.133999	-0.001298	-0.000076	0.003283	idle
1	9.422636	0.184846	2.900699	0.003054	0.002062	0.000611	idle
2	9.409476	0.341575	2.709273	0.003054	0.002062	0.000611	idle
3	9.207282	0.266202	3.116053	-0.002291	-0.000382	-0.001451	idle
4	9.800702	-0.120239	3.147160	0.000382	-0.000764	0.005498	idle

Figura 22 - Exemplo de leituras de sensores do dataset de treino.

Podemos observar na figura 23 que uma das classes representa quase 80% das amostras. Isto traduz-se num *dataset* muito desequilibrado. Existem várias maneiras de lidar com este fator, incluindo técnicas de *resampling* ou a escolha de uma métrica adequada. Este tópico será discutido em detalhes na próxima seção.

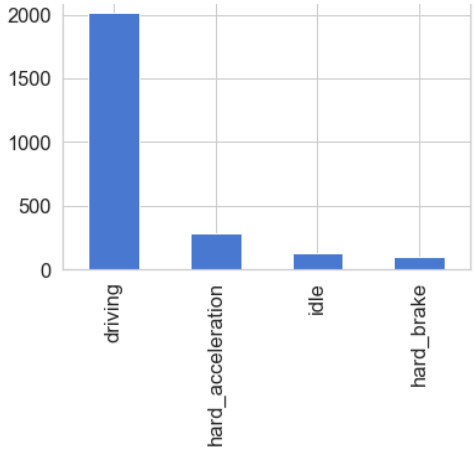

Figura 23 - Distribuição de amostras de treino por tipo de movimento - imagem de autor.

Os dados dos sensores, tais como as leituras de acelerómetros e giroscópios, são sequenciais. Para modelar estes dados, as leituras dos sensores são agrupadas em sequências de registos que formam eventos. Neste caso, definimos que cada evento deveria durar 2 segundos e conter 20 registos, perfazendo um intervalo de tempo de 100ms entre cada amostra como podemos observar na Equação 1.

$$Evento\ (seg) = \ Amostras\ de\ Dados\ *\ Intervalo\ de\ Tempo(segundos)$$

Equação 1 – Cálculo da duração de um evento

Conforme mencionado anteriormente, cada registo de dados consiste em 59 valores de sensores e uma das quatro categorias de movimento.

Podemos observar nos exemplos da figura 24 que é muito difícil diferenciar os diferentes movimentos a olho nu. No entanto, os modelos de Machine Learning são muito bons na deteção de padrões nos dados mais complexos.

Figura 24 - Exemplos de leituras de sensores de cada classe de movimento

Na figura 25 podemos observar um *heatmap* representando a matriz de correlações entre os 6 sensores principais.

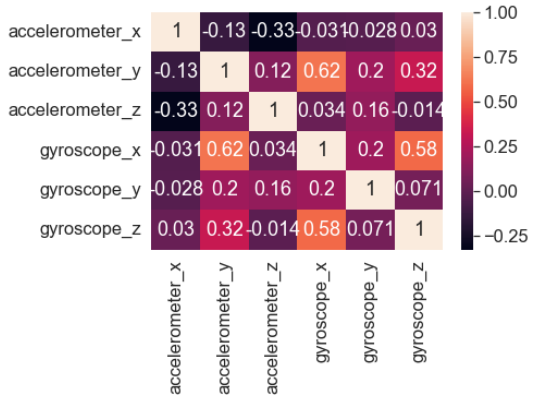

Figura 25 - Matriz de correlação do subconjunto do dataset de treino

Nesta análise exploratória ficamos com uma ideia do que consistem os registos de dados, como se forma um evento e como os diferentes sensores se correlacionam. No próximo passo iremos dar início à fase de pré processamento dos dados para serem posteriormente modelados.

3. Preprocessamento de dados

Dada a natureza ruidosa dos dados de sensores em bruto, é necessário fazer um bom pré processamento dos dados para garantir a qualidade dos mesmos. De seguida será explicado alguns destes processamentos e qual o seu objetivo.

Outliers

Sendo uma rede neuronal sensível a *outliers*, é necessário proceder à remoção dos mesmos. Na figura 26 observamos um gráfico de dispersão do acelerómetro y e x da classe *hard brake* que contém um potencial *outlier* no canto superior direito.

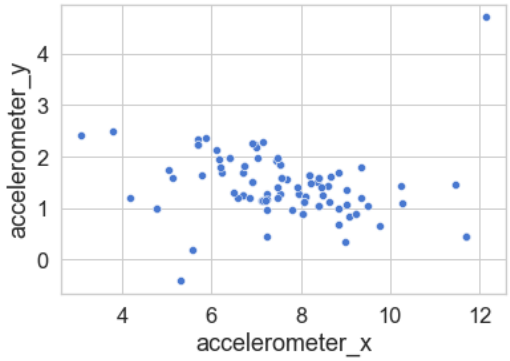

Figura 26 - Amostras de 'hard break' plotados para o acelerómetro x e y

O método Z-Score é uma técnica de estatística usada para medir o desvio de um registo de dados da média, tal como se observa na figura 27 e é comumente aplicado para detetar *outliers* presentes nos dados. Ao contrário dos modelos baseados em árvore, as Redes Neurais são mais sensíveis a outliers [59] uma vez que todos os valores devem ser escalados para evitar dar mais importância a determinados sensores com base na sua gama de valores sendo que os outliers têm um grande impacto nesta etapa de preprocessamento.

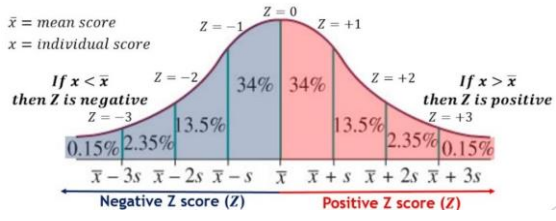

Figura 27 - Curva de distribuição de Z-score - imagem de Vitalflux

Dos 2.525 registos de dados, 4 foram considerados *outliers* e removidos do dataset.

Scaling

O dataset contém sensores que possuem diferentes graus de magnitude, alcance e unidades. Se queremos que os modelos interpretem os diferentes sensores com igual importância, é necessário aplicar uma técnica de escalamento. Os valores serão normalizados com um "*min-max scaler*" segundo a fórmula apresentada na Equação 1 que irá manter a mesma distribuição de dados, mas variando apenas entre 0 e 1.

$$X_{scaled} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

Equação 1 - Cálculo da Min-Max-scaler

4. Engenharia e seleção de *Features*

A engenharia de *features* é habitualmente uma etapa fundamental no pipeline de Machine Learning pois envolve a criação de novas features mais informativas a partir de outras presentes nos dados originais e melhora substancialmente o desempenho dos modelos. Algumas das potenciais features a serem geradas para esta tarefa seriam por exemplo a Magnitude de Aceleração e Taxa de Mudança de Aceleração.

No entanto, embora estas *features* possam ser extremamente valiosas para análise off-line e treino de modelos, há uma consideração significativa para aplicações em tempo real, especialmente em dispositivos móveis. Estes cálculos em tempo real podem ser computacionalmente intensivos, uma vez que este processo de gerar novas features teria de ser replicado na aplicação e processado cada vez que o modelo é chamado (ou seja, a cada 100ms). Dadas as restrições dos dispositivos móveis em termos de poder de processamento e duração da bateria, esta computação intensiva pode não ser eficiente.

Desta forma, em prol de uma aplicação mais rápida e eficiente, esta etapa não será implementada.

5. Seleção de Algoritmo

Esta seção detalha as arquiteturas disponíveis, vantagens e desvantagens, e fornece *insights* sobre o modelo mais adequado para esta tarefa. As arquiteturas mais comuns na abordagem deste tipo de problemas são:

Redes Neurais Recorrentes (RNNs)

As RNNs são projetadas para lidar com dados sequenciais, mantendo um estado oculto que captura informações sobre etapas anteriores. O estado oculto é atualizado a cada intervalo de tempo com base na nova entrada e no estado oculto anterior. Esta arquitetura é adequada principalmente para lidar com **sequências curtas de dados** e a sua implementação é extremamente simples. No entanto, com sequências maiores, é propenso a sofrer de problemas de *vanishing gradient*.

Long Short Term Memory (LSTM)

LSTMs são uma extensão das RNNs projetadas para captar dependências de longo prazo nos dados. Estas incluem uma célula de memória e três *gates* (gate de entrada, saída e *forget*) para controlar o fluxo de informações. Embora este tipo de Redes Neuronais seja menos suscetível ao problema de *vanishing gradient*, a sua arquitetura é mais complexa, levando a um tempo de treino mais longo e exigindo mais recursos computacionais no momento da inferência.

Embora as LSTMs sejam mais poderosas e capazes de capturar dependências de longo prazo em dados sequenciais, há cenários em que uma simples RNN pode ser mais adequada, especialmente ao lidar com dados de sensores sequenciais em eventos de curta duração. Podemos observar as principais razões:

1. **Arquitetura mais simples** - Se os dados do sensor forem simples e não tiverem dependências de longo prazo, os recursos de memória das LSTMs poderão ser desnecessários. Uma RNN pode fornecer uma arquitetura de modelo mais simples e mais fácil de treinar e interpretar.
2. **Deteção de padrões locais -** As RNN são projetadas para detetar padrões locais nos dados. Se o movimento do veículo puder ser inferido a partir de padrões de curto prazo nos dados do sensor, uma RNN poderá ser mais eficiente.
3. **Eficiência Computacional -** LSTM têm uma estrutura mais complexa com múltiplos gates, o que as torna computacionalmente mais intensivas, especialmente para sequências longas. Tendo em conta o facto de ser desejada a sua implementação na aplicação e obter inferência em tempo real num dispositivo com recursos potencialmente limitados, uma RNN irá obter um desempenho mais eficiente.
4. **Overfitting -** Para dados de sensores mais simples, as capacidades das LSTM podem ser muito altas, levando à deteção de padrões altamente complexos e assim causando

overfitting. Uma RNN simples, com menos parâmetros, poderá generalizar melhor para dados não observados nos dados de treino.

Concluindo este passo, ambas as arquiteturas RNN e LSTM serão consideradas nos testes de treino e algumas versões destes modelos serão selecionadas para serem implementadas na aplicação de modo a oferecer aos utilizadores uma escolha adaptada às suas necessidades específicas e capacidades do dispositivo.

6. Treino e Validação

Neste passo iremos proceder ao treino de diversos modelos. Irão ser testadas iterativamente várias hipóteses, técnicas e híper-parâmetros com a finalidade de obter os modelos mais eficientes. Os híper-parâmetros iniciais para os modelos RNN e LSTM foram os que estão definidos como padrão. Estes incluem o *learning rate*, *batch size*, *hidden_layers* e *units_per_ layer*.

Na figura 28 podemos observar o processo de treino dos modelos iniciais.

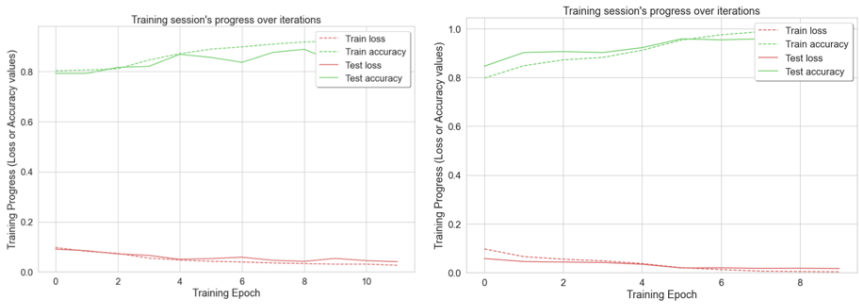

Figura 28 - Loss Function e precisão ao longo das épocas de treino (RNN à esquerda e LSTM à direita)

Após realizado o treino e testado o modelo nos dados de teste, podemos observar na figura 29 as métricas obtidas individualmente para cada classe (primeiras quatro linhas) e tambem as métricas globais calculadas de duas formas diferentes. A diferença entre a "macro avg" e a "weighted avg" traduz-se no simples facto de a primeira ser uma média geral, ou seja, trata todas as classes com igual importância e a segunda ser uma média ponderada, ou seja, dará mais importância às classes com maior frequência. A escolha do método de cálculo da média vai mais uma vez ser decidido conforme os requisitos do projecto.

	precision	recall	f1-score	support
idle	0.91	0.95	0.93	203
driving	0.75	0.41	0.53	29
hard_brake	0.67	0.44	0.53	9
hard_acceleration	0.62	1.00	0.77	10
accuracy			0.87	251
macro avg	0.74	0.70	0.69	251
weighted avg	0.87	0.87	0.86	251

	precision	recall	f1-score	support
idle	0.94	0.98	0.96	203
driving	0.88	0.72	0.79	29
hard_brake	0.83	0.56	0.67	9
hard_acceleration	0.89	0.80	0.84	10
accuracy			0.93	251
macro avg	0.88	0.76	0.82	251
weighted avg	0.93	0.93	0.92	251

Figura 29 - Métricas dos modelos (RNN à esquerda e LSTM à direita)

Apesar dos resultados do modelo terem mostrado um desempenho aceitável sem overfitting (comparando as métricas dos dados de validação com os dados de teste), avançamos com algumas etapas adicionais para otimizar ainda mais os modelos. Na última etapa de modelação, foram aplicadas algumas técnicas para evitar o overfitting, tais como Regularização e experimentados diversos algoritmos de otimização como *Adam*, *RMSProp* ou *SGD* para melhorar a precisão. De salientar que é muito difícil perceber teoricamente quais os métodos de otimização e híper-parâmetros mais indicados para o modelo, isto vai depender dos dados, arquitetura dos modelos e outras variáveis pelo que o método mais eficiente de otimização de modelos é precisamente 'tentativa-erro' de modo a achar o conjunto de variáveis ótimas.

7. Métricas de desempenho

Para ter uma visão mais completa da comparação de modelos, diversas métricas foram consideradas no *benchmark*, incluindo Precisão, Recall e F1-Score. Foram selecionados 4 modelos finais para poderem ser comparados e os resultados estão assinalados na tabela 4.

Arquitetura	Parâmetros (milhares)	Precisão Média macro	Recall Média macro	f1-score	Tamanho (Kb)
RNN	13	0,85	0,84	0,85	22
RNN	43	0,86	0,85	0,86	52
LSTM	70	0,81	0,75	0,77	90
LSTM	274	0,82	0,76	0,74	292

Tabela 4 - Métricas de avaliação e tamanho dos modelos finais quantizados.

Macro Avg foi a metodologia de média utilizada para *benchmarking* dos resultados, pois é capaz de obter uma visão geral do desempenho do modelo dando igual importância a todas as classes. A técnica de compressão *Post-Training Quantization* foi aplicada e causou apenas um decréscimo marginal na performance do modelo do modelo e reduziu o tamanho dos modelos cerca de 40% em média.

Os modelos RNN apresentaram melhor desempenho. Como referido anteriormente, isto deve-se à sua arquitetura mais simples, robustez contra overfitting e capacidade de aprender padrões em sequências mais curtas. Podemos observar na figura 30 a Matriz de Confusão do segundo modelo listado na tabela 4 de forma a entender com mais detalhe as capacidades do modelo

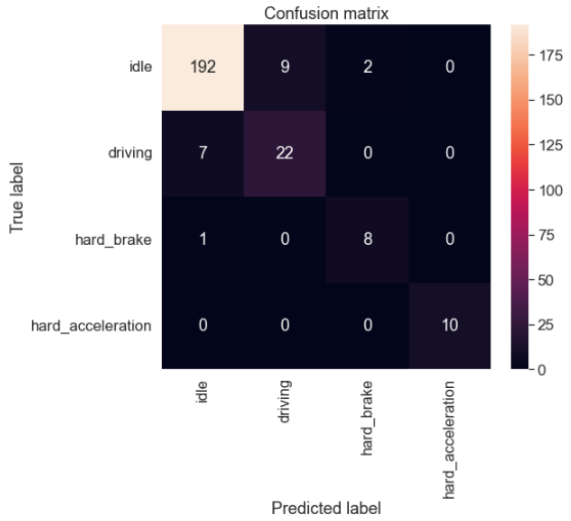

Figura 30 - Matriz de confusão do melhor modelo RNN

A Matriz de Confusão mostra que o modelo foi capaz de identificar com bom desempenho as classes 'idle', 'hard_accelaration' e 'hard brake' e teve resultados satisfatórios para 'Driving', que se explica pela enorme variedade de movimentos associados a 'Driving' comparados com as outras classes que são bastante mais objetivas e definidas. Estes resultados estão acima das expectativas dado o tamanho altamente restrito dos dados usados para treinar o modelo. Aumentando o tamanho dos dados de treino levaria hipoteticamente a um modelo capaz de identificar todas as classes com uma elevada precisão. Os dados foram obtidos em ambiente bastante restrito pelo mesmo condutor, veículo e condições atmosféricas. Na secção de 'Testes' iremos observar se o modelo é capaz de generalizar bem o suficiente para se adaptar a outros condutores, veículos e condições ambientais.

8. Otimização e Exportação

De um modo simples, o tamanho do modelo e a complexidade do cálculo no momento da inferência é proporcional ao número de pesos (ou parâmetros) que o compõe. Cada peso, que assume um formato numérico, pode ter diferentes graus de precisão, sendo que quanto maior a precisão, mais espaço de memória irá ocupar e mais complexos - e precisos - serão os cálculos. Assim sendo, conseguimos otimizar o modelo tanto reduzindo o número de

parâmetros como reduzindo a precisão - e consequentemente o tamanho. Foram então experimentadas as seguintes técnicas de otimização:

1. **Pruning** - esta técnica remove os pesos e conexões que têm pouco ou nenhum impacto no modelo, fazendo com que apenas fiquem os parâmetros mais relevantes.

2. **Clustering -** agrupamento dos pesos de cada layer em *clusters*, sendo o valor do peso arredondado para o valor do centroide do *cluster*. Assim, por exemplo, numa layer com 1000 parâmetros, se aplicarmos a técnica de clustering com 100 clusters, conseguimos uma redução aproximada de 10x no tamanho do modelo. Na figura 31 podemos observar uma tabela retirada da documentação do Tensorflow e que mostra com detalhe o nível de compressão para diferentes números de *clusters*.

Model	Original		Clustered			
	Top-1 accuracy (%)	Size of compressed .tflite (MB)	Configuration	# of clusters	Top-1 accuracy (%)	Size of compressed .tflite (MB)
MobileNetV1	70.976	14.97	Selective (last 3 Conv2D layers)	16, 16, 16	70.294	7.69
			Selective (last 3 Conv2D layers)	32, 32, 32	70.69	8.22
			Full (all Conv2D layers)	32	69.4	4.43
MobileNetV2	71.778	12.38	Selective (last 3 Conv2D layers)	16, 16, 16	70.742	6.68
			Selective (last 3 Conv2D layers)	32, 32, 32	70.926	7.03
			Full (all Conv2D layers)	32	69.744	4.05

The models were trained and tested on ImageNet.

Figura 31 - Exemplo dos resultados de compressão com Clustering – Imagem da documentação do Tensorflow

3. **Operations Fusion** - durante o processo de inferência, várias operações são realizadas - como por exemplo *convolution*, *pooling*, *flattening*, *padding*, entre outras. Uma forma de reduzir a computação é combinar múltiplas operações numa só operação - de notar que nem todas as operações podem ser combinadas

4. **Quantization** - é a técnica mais comum e que consiste em reduzir a precisão dos pesos do modelo - por exemplo, convertendo um float de 32 bits para um float de 16 bits. Desta forma, abdica-se de um pouco de precisão do modelo em troca de uma redução no tamanho significativa e assim maior rapidez na inferência. Existem diferentes tipos de Quantização, sendo os mais comuns *Post-Training Quantization* ou *Quantization-aware Trainin*. Na figura 32 podemos observar as diferentes técnicas e o potencial

impacto que têm na redução do tamanho do modelo bem como o modo como afetam a precisão do mesmo.

Technique	Data requirements	Size reduction	Accuracy	Supported hardware
Post-training float16 quantization	No data	Up to 50%	Insignificant accuracy loss	CPU, GPU
Post-training dynamic range quantization	No data	Up to 75%	Smallest accuracy loss	CPU, GPU (Android)
Post-training integer quantization	Unlabelled representative sample	Up to 75%	Small accuracy loss	CPU, GPU (Android), EdgeTPU, Hexagon DSP
Quantization-aware training	Labelled training data	Up to 75%	Smallest accuracy loss	CPU, GPU (Android), EdgeTPU, Hexagon DSP

Figura 32 - Resultados de métodos de Quantização. Fonte: Documentação do Tensorflow

Foram realizados pela equipa do Tensorflow alguns testes a diferentes modelos e como se comportam antes e depois da quantização aplicada. Na figura 33 podemos observar os resultados destes testes e que de uma forma geral a técnica QAT tem um decréscimo marginal na precisão e uma redução significativa na latência do modelo enquanto que a técnica PTQ tem algum impacto na precisão e pouco impacto na latência.

Model	Top-1 Accuracy (Original)	Top-1 Accuracy (Post Training Quantized)	Top-1 Accuracy (Quantization Aware Training)	Latency (Original) (ms)	Latency (Post Training Quantized) (ms)	Latency (Quantization Aware Training) (ms)	Size (Original) (MB)	Size (Optimized) (MB)
Mobilenet-v1-1-224	0.709	0.657	0.70	124	112	64	16.9	4.3
Mobilenet-v2-1-224	0.719	0.637	0.709	89	98	54	14	3.6
Inception_v3	0.78	0.772	0.775	1130	845	543	95.7	23.9
Resnet_v2_101	0.770	0.768	N/A	3973	2868	N/A	178.3	44.9

Figura 33 - Resultados da aplicação de PTQ e QAT em diferentes modelos. Fonte: documentação do Tensorflow

Para estes modelos apenas a técnica de **Post Training Quantization** (PTQ) foi aplicada pois é a mais fácil de implementar, uma vez que se limita a arredondar os pesos já depois do treino a um grau de precisão menor. Outras técnicas podem garantir uma maior eficiência dos modelos e serão testadas futuramente.

9. Sensores adicionais para fornecer contexto do comportamento do motorista

A utilização de dados de sensores adicionais, como nível de bateria, chamadas telefónicas recebidas, luminosidade e localização, pode melhorar significativamente o desempenho dos modelos e permite também a implementação de novas funcionalidades que

melhoram a experiência do utilizador da aplicação. Por exemplo, os dados do nível da bateria podem ser usados para otimizar o consumo de recursos da aplicação, garantindo que a mesma descarregue a bateria em momentos críticos fazendo a alternação para modelos mais leves automaticamente. Sensores de luminosidade podem ajustar o brilho do ecrã ou sugerir acender os faróis, enquanto que os dados de localização podem oferecer *insights geográficos*, como zonas potencialmente perigosas.

Com este passo é terminado o treino dos modelos de Classificação de Movimento. O bom desempenho num ambiente de treino não garante um bom desempenho no mundo real. Na secção de Testes irão ser realizados testes exaustivos no mundo real para avaliar a verdadeira qualidade do modelo.

3.6.3 Deteção de objetos na Estrada

O modelo de deteção de objetos na estrada é um componente essencial deste projeto, responsável por identificar e classificar sinais de trânsito e outras entidades - tais como bicicletas e radares - encontrados durante a condução. Esta seção aborda as complexidades do treino deste tipo de modelos e respetivos testes, critérios de benchmarking e técnicas de otimização.

O pipeline de treino do modelo é ilustrado na figura 34 e mostra um passo a passo do processo, desde a limpeza dos dados até a conversão do modelo para um formato compatível com dispositivos móveis.

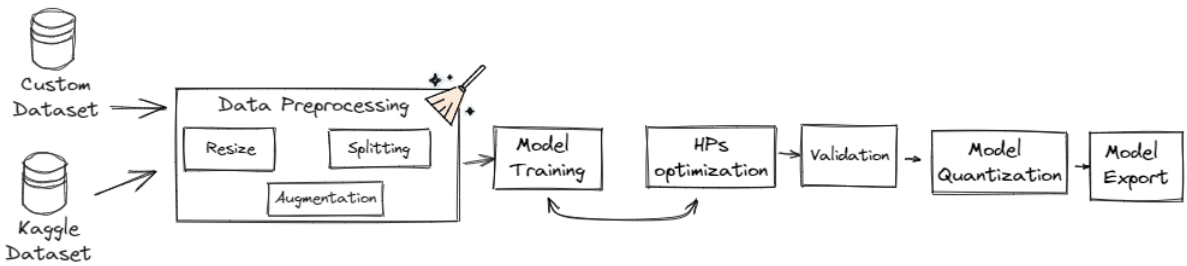

Figura 34 - Pipeline de treino do modelo de deteção de sinais de trânsito

1. Carregamento e divisão do dataset

Após a anotação dos dados customizados, tanto o dataset público como as imagens customizadas foram padronizados no mesmo formato (Pascal VOC) e organizadas no mesmo

diretório. Os dados são carregados para o ambiente de treino e divididos em grupos de treino, validação e teste (80/10/10).

2. Pré Processamento dos dados

Tendo o *dataset* criado, é necessário aplicar algumas transformações às imagens para que estejam no formato pretendido pelo modelo. Este pré processamento é composto pelos seguintes passos:

1. **Normalização**, que consiste na transformação da imagem para o formato pretendido pelo modelo. Por exemplo, o modelo pré-treinado Efficient Det-Lite 0 necessita das imagens no formato [224, 224, 3], que se traduz numa imagem de 224x224 pixels com 3 canais de cor (RGB).
2. **(Opcional) Data Augmentation**, uma técnica muito utilizada em Machine Learning especialmente quando o volume de dados não é suficiente. Esta técnica vai criar cópias das imagens do dataset e aplicar transformações às mesmas tais como rotação, espelhamento, cortes, diferentes cores e resoluções, entre outras, tal como é ilustrado na figura 35.

Figura 35 - Exemplo de Data Augmentation – Fonte: Data Camp

3. Seleção de modelo pré-treinado

Uma Neural Network pode ser treinada de raiz, definindo o número de neurões, hidden layers, funções de ativação, entre outros, ou alternativamente pode ser definida uma arquitetura de modelo pré-treinado e aplicar a técnica de *Transfer Learning* que é bastante eficiente, reduz o tempo de treino e aumenta a performance significativamente.

Esta etapa consiste em escolher um modelo pré-treinado que se alinhe às restrições de hardware e requisitos de desempenho da solução proposta. Para aplicações móveis, os modelos EficienteDetLite, a versão mais leve da conhecida arquitetura EfficientDet [60], parecem ser uma boa escolha devido à sua arquitetura leve. A ferramenta de treino TFlite Model Maker oferece 5 versões deste modelo - de 0 a 4 -, tendo cada versão arquiteturas

ligeiramente diferentes, sendo as inferiores mais leves e exigindo menos computação, com a desvantagem de uma menor precisão.

4. Definição da métrica de avaliação do modelo

Conforme visto na introdução da seção, escolher a métrica certa é crucial para selecionar os melhores modelos. Após uma avaliação criteriosa, a métrica selecionada a ser utilizada para benchmarking dos modelos é a *mean Average Precision*, exemplificada na Equação 2.

$$mAP = \frac{1}{n}\sum_{k=1}^{k=n} AP_k$$
$$AP_k = \text{ the AP of class } k$$
$$n = \text{ the number of classes}$$

Equação 2 - Cálculo da mean Average Precision

A Precisão Média - ou Average Precision (AP) - é uma métrica única que encapsula a precisão e o recall, dando uma visão abrangente do desempenho do modelo. Isto é possível calculando a área debaixo da curva *Precision-Recall* como podemos observar na figura 36, resumindo efetivamente o desempenho do modelo em diferentes *thresholds*. Ao lidar com múltiplas classes, é usada a mean Average Precision (mAP), que calcula a média da AP de todas as classes tornando-a assim muito robusta para lidar com datasets com desequilíbrio de classes quando a importância das mesmas é semelhante. No eventual caso de haver uma classe mais importante de detetar do que as outras, como seria possivelmente o caso de uma classe de 'crash', teríamos de proceder a uma análise de métricas ao nível da classe em vez de um modo mais generalizado.

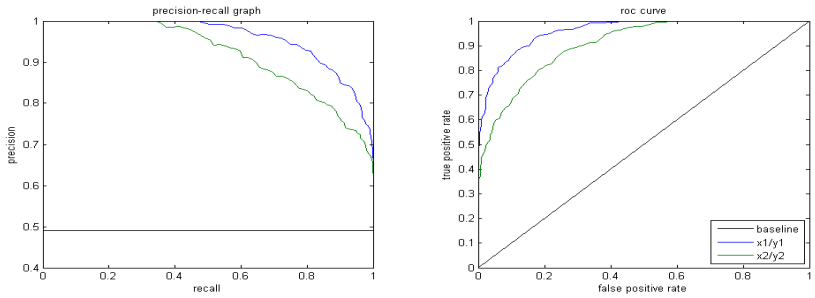

Figura 36 - Curvas PR e ROC. Imagem de machinelearningmastery.com

5. Treino e ajuste de híper-parâmetros do modelo

Esta etapa é frequentemente considerada a mais importante do *pipeline* de desenvolvimento do modelo, onde ocorrem os testes e experimentações mais extensos. Esta etapa é crucial para o desempenho e eficiência finais do modelo. O objetivo é encontrar o conjunto ideal de híper-parâmetros que maximizem a aprendizagem, minimizem o tempo de treino e evitem o overfitting e underfitting. Os híper-parâmetros testados foram o *learning rate*, *batch size* e número de *epochs.* Após cada iteração de treino, é necessário avaliar cuidadosamente o comportamento do modelo para garantir que a próxima iteração seja melhor. Por exemplo, um learning rate baixo pode fazer com que o modelo aprenda de forma mais eficaz, mas pode aumentar significativamente o tempo de treino. Por outro lado, uma learning rate mais alto pode acelerar o treino, mas pode levar a um overfitting.

Não há na verdade nenhuma fórmula que nos diga quais são os híper-parâmetros mais indicados; os mesmos dependem da qualidade e complexidade dos dados e do modelo. Assim, a única forma é experimentar e fazer várias iterações de treino, sempre observando os resultados e percebendo se o modelo está a fazer underfitting ou overfitting, entre outros.

O primeiro modelo foi treinado com base no modelo pré treinado EfficientDet-Lite 0 e utilizando os híper-parâmetros padrão (50 epochs com batch size de 64) e não treinando o modelo todo. Foram utilizadas as 4 categorias do dataset do Kaggle bem como as 2 categorias de imagens customizadas- *radar* e *bike*.

Na figura 37 observamos o processo de treino. Em cada *epoch*, os vários *batches* de imagens vão sendo passados ao modelo e o mesmo vai processar os dados, atualizar os valores dos pesos com o objetivo de minimizar a *Loss Function*. É expectável que com o avançar das *epochs*, os valores de *loss* e *val_loss* vão baixando.

Figura 37 - Registo das métricas do modelo ao longo do treino

Podemos observar na figura 38 que o modelo treinou quase 3.2M de parâmetros com a duração de 17 minutos. Tendo o modelo treinado, vamos testar a sua performance nos dados de teste - que o modelo nunca 'viu' - para averiguar se de facto aprendeu a detetar os padrões

e identificar os objetos ou se se limitou a "memorizar" os dados de treino. Neste caso, a métrica que foi estipulada (Average Precision) foi de 37,8%.

```
Model: ""
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 keras_layer (KerasLayer)    multiple                  3234464

 class_net/class-predict (Se  multiple                 4671
 parableConv2D)

 box_net/box-predict (Separa  multiple                 2916
 bleConv2D)

=================================================================
Total params: 3,242,051
Trainable params: 3,194,915
Non-trainable params: 47,136
_________________________________________________________________
```

Figura 38 - Sumário do modelo.

Podemos também observar ao detalhe qual a precisão para cada objeto:

1. 'AP_/trafficlight': 0.2114612,
2. 'AP_/speedlimit': 0.7382128,
3. 'AP_/crosswalk': 0.30975205,
4. 'AP_/stop': 0.6362452,
5. 'AP_/bike': 0.36242583,
6. 'AP_/radar': 0.012646835

A performance dos objetos 'speed limit', 'crosswalk', 'stop' e 'bike' foram satisfatórios, por outro lado, os objetos 'traffic light' e 'radar' não foram aceitáveis. Isto deve-se à pouca qualidade dos dados mas especialmente ao facto da reduzida quantidade. Por forma a não comprometer a qualidade final dos modelos, foi decidido nesta fase ignorar o objeto 'radar' e futuramente, após a recolha de mais e melhores dados, retreinar o modelo.

Podemos observar na tabela 5 que a *loss function* dos dados de validação vai sendo minimizada ao longo de todo o treino sendo que a mesma começa a estabilizar após a 35ª *epoch*. Isto significa que após esta epoch o modelo já não está a aprender informação relevante e corremos o risco de começar a fazer *overfitting* aos dados de treino. Uma estratégia utilizada para combater este fator é chamada de Early Stopping onde se pode parametrizar o modelo para automaticamente interromper o treino caso a melhoria de uma epoch para a outra seja inferior a um valor estipulado, evitando desta forma não só um treino adicional desnecessário como por outro lado, temos a garantia de que o modelo vai ser treinado de forma otimizada e sem overfitting.

Epoch	Learning Rate	Loss	Val Loss
1	0.0435	1.22	0.60
5	0.0783	0.60	0.33
10	0.0728	0.53	0.430
15	0.0639	0.49	0.28
20	0.0526	0.46	0.24
25	0.0400	0.45	0.22
30	0.0274	0.44	0.22
35	0.0161	0.42	0.21
40	0.0072	0.40	0.20
45	0.0017	0.41	0.20
50	0.0002	0.41	0.20

Tabela 5 - Progresso do modelo ao longo das várias épocas de treino:

No seguimento deste processo de treino foram efetuadas novas rondas de treino testando-se diferentes híper-parâmetros e observando o impacto no resultado do modelo com a finalidade de otimizar os mesmos.

Na primeira iteração foi usado o EfficientDet-Lite 0 como modelo pré-treinado com os híper-parâmetros padrão e alcançou 37,8% de mAP.

Podemos observar na figura 39 o *learning rate* do modelo a diminuir ao longo do tempo e a *loss function* a ser minimizada e a estabilizar após a 40ª época.

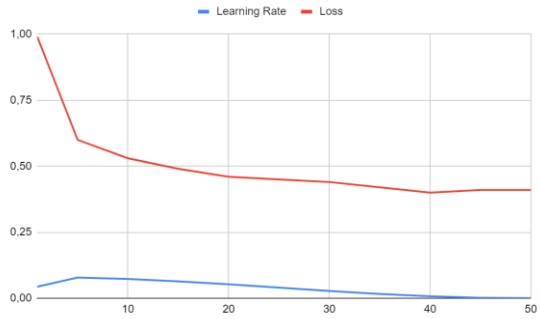

Figura 39 - Precisão e loss function de treino ao longo das épocas.

Após várias iterações experimentais, o conjunto final de otimização incluiu a remoção do objeto 'radar' que devido à pouca quantidade de amostras, o modelo não foi capaz de aprendê-lo, ajuste do learning rate progressivamente em cada *epoch* e reduzir o número de *epochs* para evitar overfitting. Estas mudanças melhoraram o mAP de 37,8% anteriores para 56,2%.

Posteriormente, outros modelos foram treinados para realizar a mesma tarefa, mas com aumento progressivo de complexidade por forma a ir de encontro aos requisitos da aplicação em oferecer uma gama de modelos com diferentes performances e complexidade. No final, apresentamos uma lista de 5 modelos descritos na tabela 6 que obtiveram uma boa performance e eficiência em qualidade vs. complexidade.

Modelo pré treinado	Tamanho do batch	Epochs	Tempo de treino	mAP	Tamanho
EficienteDet-Lite0	32	40	35 minutos	56,2%	4,2 MB
EficienteDet-Lite1	32	40	66 minutos	61,9%	5,7 MB
EficienteDet-Lite2	16	40	98 minutos	63,8%	7,1 MB
EficienteDet-Lite3	16	40	129 minutos	65,9%	11,2 MB
EficienteDet-Lite4	8	40	347 minutos	68,3%	19,6Mb

Tabela 6 - Resultados das experiências e modelo inicial de deteção de objetos na estrada

Podemos observar que os modelos mais pesados demoram mais tempo a treinar, exigem batches menores de dados para evitar problemas de falta de memória e o tamanho é significativamente maior, sendo que alcançaram também uma melhor *Average Precision*.

6. Otimização e Exportação dos modelos

Vários elementos podem influenciar a precisão de um modelo quando ele é convertido para formato TFLite. A quantização reduz significativamente o tamanho do modelo — até um fator de x4 — mas pode resultar numa ligeira diminuição na precisão. Embora existam maneiras mais eficientes [61] de efetuar este processo, foram usados os métodos de quantização disponíveis na ferramenta utilizada tflite_*model_maker*.

A etapa experimental termina com 5 modelos otimizados com diferentes necessidades de hardware descritos na tabela 7. Terminado o desenvolvimento dos modelos, iremos na secção de "Testes" verificar a performance dos mesmos no mundo real.

Modelo PT	Parâmetros (Milhões)	mAP (%)	Quantização	mAP PTQ (%)	Tamanho (mb)
EficienteDet-Lite0	3,2	57,7	int8	56,49	4,2
			fp16	57,16	6,6
			dynam	57,09	4
EficienteDet-Lite1	4,2	63,67	int8	62,06	5,7
			fp16	63,24	8,7
			dynam	62,91	5,4
EficienteDet-Lite2	5,2	64,32	int8	62,34	7,1
			fp16	63,19	10,9
			dynam	62,51	6,6
EficienteDet-Lite3	8,3	67,73	int8	65,87	11,2
			fp16	67,16	17,8
			dynam	66,18	10,2
EficienteDet-Lite4	15,1	70,09	int8	68,72	19,6
			fp16	69,26	32,3
			dynam	69,01	17,9

Tabela 7 - Tamanho e precisão dos modelos finais antes e depois PTQ

Tendo em conta a tipologia dos modelos e quantidade de dados, os custos de treino são praticamente marginais (1,32€/hora) e por isso para já é um fator na tomada de decisão que tem pouco peso - no entanto, futuramente com maior volume de dados e mais objetos, o impacto será mais significativo e por isso iremos utilizá-lo na comparação entre os modelos.

Como referido anteriormente, vamos selecionar 5 modelos em que apesar de todos desempenharem a mesma tarefa, os mesmos requerem recursos de hardware diferentes e consecutivamente originam uma performance diferente. Após um levantamento das necessidades, os fatores escolhidos para comparar os modelos são o consumo de bateria, latência, mAP, tamanho e custo e podemos observar na figura 40 uma comparação entre estes.

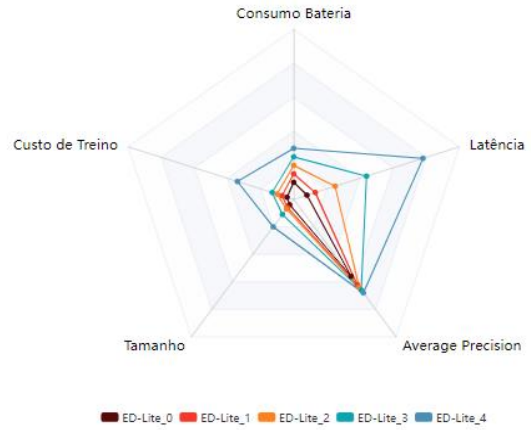

Figura 40 - Comparação dos fatores de decisão nos modelos treinados.

3.6.4 Reconhecimento de Expressão Facial

O reconhecimento de expressão facial é uma funcionalidade inovadora nesta solução, e tem como objetivo avaliar o estado emocional do condutor para detetar estados potencialmente perigosos e assim melhorar a segurança e a experiência do mesmo.

Como referido na secção "Criação dos Datasets", foi utilizado um banco de imagens que possui expressões faciais – ilustradas na figura 41 - de diversas categorias: neutro, feliz, zangado, surpreso, medo, nojo e triste.

Figura 41 - Imagem do dataset FER2013. Fonte: Kaggle

Apesar de estas emoções não servirem exatamente o propósito da funcionalidade pretendida - detetar cansaço, sonolência ou distração - iremos na mesma proceder à sua utilização para demonstrar nesta prova de conceito que é possível a integração de um modelo que interprete a emoção do condutor e que a aplicação desencadeie ações autonomamente com base nas mesmas.

O treino do modelo está ilustrado na figura 42.

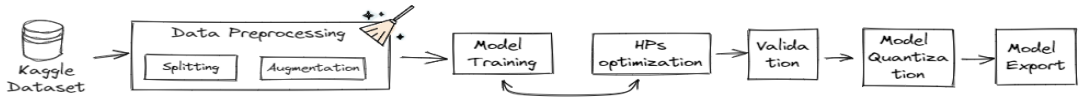

Figura 42 - Pipeline de treino do modelo de reconhecimento de expressão facial

1. Preprocessamento de dados

Dada a natureza do dataset, não foram necessárias muitas transformações de imagem. O dataset já está padronizado em tamanho e formato. Foi adicionado ao pipeline um processo de Data Augmentation para aumentar o tamanho do dataset de forma a criar novas imagens geradas artificialmente, aplicando transformações a imagens existentes, como rotação ou espelhamento. O dataset é posteriormente dividido em conjuntos de treino, validação e teste (80/10/10).

2. Seleção e Arquitetura de Algoritmos

As arquiteturas de modelos pré-treinados consideradas foram Efficient Net e Mobile Net. Estas arquiteturas são Convolutional Neural Networks (CNN) robustas e altamente otimizadas para aplicações móveis. Na próxima etapa alguns testes serão realizados utilizando estas arquiteturas de modelos pré-treinados.

3. Treino e Validação

A estratégia de treino passa por uma abordagem inicial muito básica e posteriores iterações até serem obtidos 3 modelos com performance eficiente para os diferentes níveis de complexidade, descritos na tabela 8.

Modelo base	Parâmetros (M)	Quantização	mAP (%)	PTQ mAP (%)	Tamanho do modelo (Mb)
EficienteNet_0	3.2	int8	63,9	62,93	3,8
		fp16		63,43	6,5
		dynamique		63,49	3,7
EficienteNet_1	4.2	int8	66,6	66,54	4,7
		fp16		66,85	8
		dynamique		66,97	4,6
EficienteNet_2	4.8	int8	68,7	68,33	5.4
		fp16		68,64	9.3
		dynamique		68,71	5.3

Tabela 8 - Resultados dos modelos de reconhecimento de expressão facial treinados

Como podemos ver na tabela 8, a arquitetura que teve melhor desempenho foi a Efficient Net. Os resultados do modelo ficaram abaixo das expectativas, com o melhor modelo a atingir uma mAP de 68,71%. Podemos observar na tabela acima que o modelo pré-treinado utilizado como base tem um impacto significativo no desempenho e que as versões mais robustas desta arquitetura não foram possíveis de ser utilizadas no treino devido à sua complexidade e ao facto de requererem uma máquina mais potente e com mais recursos físicos. No entanto, o tamanho do modelo ainda é pequeno, especialmente depois de aplicada a Quantização pós treino, e funciona perfeitamente mesmo num telefone Android mais antigo. Este modelo é importante para permitir que a aplicação atue sempre que detete uma determinada expressão facial, porém não queremos que Falsos Positivos aconteçam com frequência, pois podem causar uma experiência ruim ao usuário.

4. Métricas de desempenho

Embora tenham sido obtidos resultados algo satisfatórios (mAP de 68,7%) e desenvolvido modelos capazes de identificar com sucesso a expressão facial, a próxima abordagem passará por adquirir um dataset mais completo de expressões faciais de condutores e replicar o pipeline de treino. Na última secção do artigo 'Conclusões e Trabalho Futuro' este tópico será discutido com mais detalhe.

3.6.5 Reconhecimento de comando de voz

O reconhecimento de comando de voz é um recurso integral na nossa solução pois a mesma foi projetada para permitir um controlo com as mãos livres e melhorar a experiência do utilizador. Esta seção elabora o modelo de Machine Learning desenvolvido para reconhecer comandos de voz personalizados para acionar ações como "Iniciar gravação", "Parar gravação" e "Chamada de Emergência".

1. Seleção e Arquitetura dos Modelos

Uma das principais considerações em projetos de Machine Learning é averiguar a necessidade de desenvolver um modelo de raiz, utilizar um modelo pré-treinado e fazer fine-tuning ou utilizar diretamente os modelos pré-treinados. Para a maioria das tarefas desta aplicação, o treino em dados customizados foi essencial devido aos requisitos específicos e à natureza única dos dados. No entanto, para esta funcionalidade, observamos que no *TensorFlow Model Hub* é oferecida uma variedade de modelos pré-treinados que se alinham às necessidades do projeto. Desta forma podemos não só reduzir custos de treino e poupar

tempo, mas também escolher um modelo com desempenho comprovado, o que muitas vezes é difícil de avaliar em modelos treinados com dados customizados.

Com esta funcionalidade queremos mostrar que é possível o **condutor interagir com a aplicação recorrendo apenas a comandos de voz** de forma a proporcionar uma condução mais segura e sem distrações.

O modelo selecionado foi um YAMNet treinado com 8 palavras-chave diferentes - "up", "down", "left", "right", "go", "stop", "on", "off", "background" - e barulho de fundo. Estas palavras-chave podem ser usadas para desencadear múltiplas ações. Para esta prova de conceito, foram definidas três ações a serem desencadeadas com três comandos de voz diferentes:

1. **Go** -> Iniciar Gravação
2. **Stop** -> Pausar Gravação
3. **Down** -> Fazer chamada de emergência

Como já foi referido, optou-se por utilizar um modelo mais robusto com palavras-chave menos significativas, pois o mesmo permite que estas sejam generalizadas e reconhecidas pela maioria dos utilizadores mesmo com sotaques, dicções e pronúncias diferentes. Treinando um modelo com palavras-chave customizadas iria requerer uma extensa recolha de amostras de várias pessoas - à semelhança da tarefa de reconhecimento de expressão facial - e que para esta prova de conceito faria menos sentido.

Estando este modelo pré-treinado bem documentado e já está no formato TFlite, o mesmo reúne todas as condições para ser integrado na aplicação.

3.6.6 Disponibilização dos modelos na aplicação

Após o treino dos modelos concluídos, é necessário que a aplicação consiga aceder aos mesmos e corrê-los localmente. Assim, os mesmos são copiados para o Firebase na diretoria de modelos onde a aplicação poderá posteriormente aceder e fazer o download dos mesmos localmente.

3.7 Aplicação Mobile

Nesta secção irá ser abordado com detalhe as metodologias e o desenvolvimento da aplicação móvel que permite a utilização dos modelos descritos na secção anterior.

3.7.1 Conceptualização da Aplicação

A primeira decisão foi referente ao sistema operativo para qual a aplicação seria desenvolvida, pois apesar de haver ferramentas que permitem que o desenvolvimento da aplicação seja compilado para iOS e Android de uma forma bastante simples com a mesma base de código, tais como React Native ou Flutter, o tipo de aplicação que é pretendido desenvolver requer um contacto muito 'próximo' com o hardware do telefone e por isso optou-se por utilizar uma linguagem nativa. Foi então decidido fazer o desenvolvimento exclusivamente para Android pois este representa 80% do mercado de smartphones e o desenvolvimento e licenciamento para este sistema operativo é muito mais livre e rápido.

Outro fator muito importante na escolha deste OS, sendo um dos grandes princípios da programação a reutilização de código e tendo em conta os requisitos funcionais da aplicação, o primeiro passo foi procurar projetos open source cujo código pudesse ser reaproveitado e integrado na aplicação - com o devido crédito. Assim, a linguagem de programação escolhida para o desenvolvimento da aplicação foi um misto de Java e Kotlin, dependendo do módulo que foi implementado e em que linguagem foi escrito o código base para o desenvolvimento do mesmo e linguagem XML para a criação da interface de utilizador. A razão pela qual é possível uma interoperabilidade entre scripts de Kotlin e Java deve-se ao facto de ambos partilharem do mesmo formato de compilação binário, ocorrido no *build time* da aplicação.

Um destaque nesta aplicação é a capacidade de armazenar todos os resultados dos modelos localmente em ficheiros CSV dentro do smartphone, servindo tanto como uma ferramenta analítica para as viagens de carro como uma "caixa negra" virtual para análise de incidentes ocorridos. Tendo sempre como prioritária a segurança e privacidade do utilizador, em nenhum momento algum tipo de dados sai do dispositivo, garantindo assim uma proteção contra ameaças externas e um desempenho confiável independente da existência de uma ligação à Internet.

Como iremos ver mais à frente, o desenvolvimento da aplicação assenta em 3 módulos principais:

1. **Modelos ML** - Para a implementação dos modelos de Machine Learning, foram utilizados os tutoriais de exemplo do repositório TensorFlow. Estes tutoriais contêm o código fonte (com licença Apache) que permite para as diferentes tarefas a recolha dos dados (imagem e áudio) e posterior uso do modelo pretendido. De forma a construir o módulo Dash Camera, o código fonte das diferentes tarefas foi agrupado num só módulo. Este processo vai ser descrito com maior detalhe na secção seguinte.

2. **Ações** - O módulo "*Actions*", desenvolvido em Java ativa funcionalidades específicas disponíveis em qualquer smartphone, tais como enviar SMS, iniciar chamadas para um

número pré-definido ou reproduzir um áudio, com base nos resultados do modelo, preenchendo a lacuna entre os modelos de ML e os elementos operacionais da aplicação. Esta classe foi desenvolvida em java pois uma parte do código base foi reaproveitada de um projeto anterior.

3. **Dash Camera** - Como referido anteriormente, esta é a principal funcionalidade da aplicação e consiste na utilização dos modelos desenvolvidos. O código do tutorial de cada tarefa foi utilizado e todos os blocos de código foram agrupadas para correr no mesmo ecrã. Este contém a imagem da câmara traseira do smartphone com o modelo de Deteção de Objetos na Estrada e em simultâneo utiliza a câmara frontal para o modelo de Reconhecimento de Expressão Facial sendo que a classe obtida no modelo é disponibilizada na camada de overlay (informação que está por cima da imagem transmitida pela câmara). O mesmo sucede com o modelo de Classificação de Movimento que após utilizar os valores dos sensores para detectar o tipo de movimento, disponibiliza a classe na camada de overlay. Por fim, o modelo de Reconhecimento de Comandos está sempre em *listening* permitindo ao utilizador acionar determinadas tarefas. Na figura 43 observamos um exemplo do ecrã que desempenha esta funcionalidade.

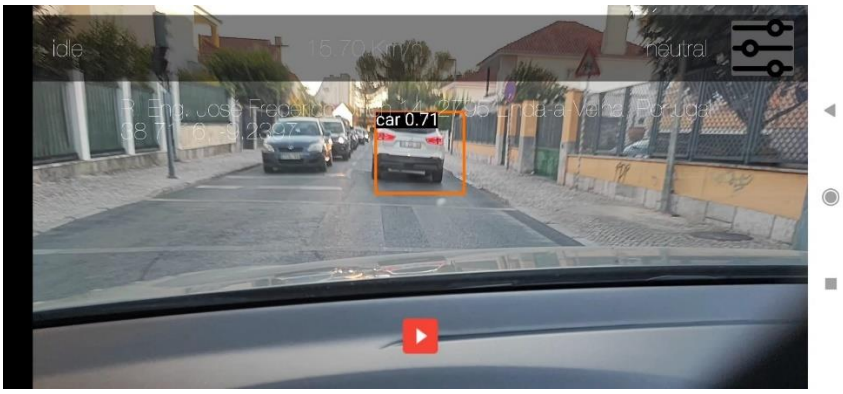

Figura 43 - Funcionalidade de Dash Camera da aplicação

Estas funcionalidades serão descritas com maior detalhe no seguimento desta secção.

3.7.2 Ferramentas de Desenvolvimento

Nesta seção iremos abordar alguns temas associados ao desenvolvimento da aplicação e debater quais as opções existentes para a execução de cada um.

Design

O design da aplicação foi inicialmente desenvolvido utilizando o software UIzard que permite de uma forma muito simples obter ou gerar designs de nível profissional. Esta componente do projeto foi desprioritizada em prol de uma dedicação maior aos módulos funcionais ficando a componente estética para trabalho futuro. No entanto, por forma a garantir uma aplicação com um UI/UX que assegure um mínimo de qualidade, foi utilizado o código fonte de uma aplicação gratuita disponível no *marketplace* do "Material Ui Ux" [62] para definir os *designs* e *layouts*.

Componentes

A aplicação foi desenvolvida com base em fragmentos e uma atividade principal, o que permite uma melhor adaptação aos diferentes ecrãs e dispositivos e também uma melhor gestão e reaproveitamento do código.

Navegação

A interação entre os componentes e processo de navegação entre os diversos fragmentos foi implementada com recurso ao package Navigation.

Como podemos observar na figura 44, este *package* permite de forma fácil através de um *navigation graph* (parte esquerda da figura) definir a lógica de navegação entre os diversos fragmentos recorrendo a linguagem XML (parte central da figura) para definir o fragmento de origem, o fragmento de destino e qual a ação que desencadeia o processo - por exemplo um clique num botão ou outro componente. Existe também uma funcionalidade mais *user-friendly* que permite definir as navegações com recurso a blocos (parte direita da figura)

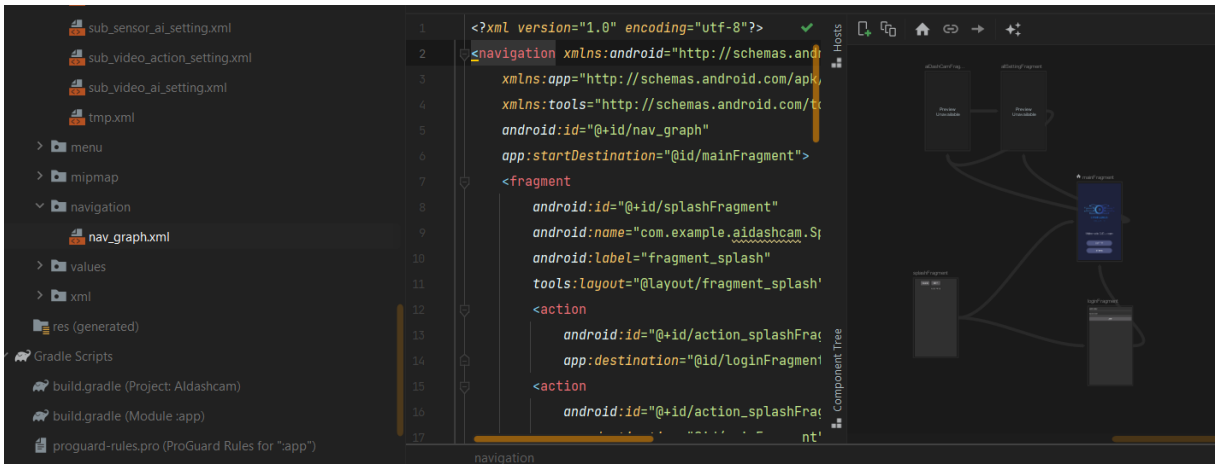

Figura 44 - Definição da lógica de navegação entre os módulos da aplicação

Variáveis e Assets

Todas as variáveis são definidas em ficheiros XML e guardadas na pasta *'values'* dentro dos Recursos (*Resources*) do código fonte, como podemos observar na parte esquerda da figura 45.

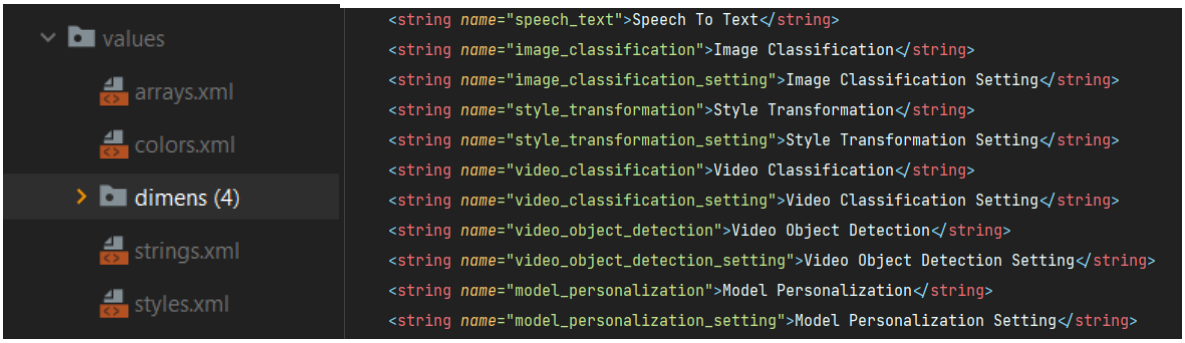

Figura 45 - Ficheiros de definição de variáveis estáticas

As variáveis guardadas incluem strings, onde tipicamente está todo o conteúdo escrito da aplicação, as cores onde são definidos os códigos de cores utilizados, arrays, listas, etc. Desta forma, podemos não só alterar todas as variáveis de forma fácil sem ter que alterar o código fonte da aplicação em inúmeros sítios mas também conseguimos desenhar a aplicação para que suporta diversas línguas.

Na pasta 'assets' guardamos objetos de diversos tipos, tais como *letter fonts*, *thumbnails*, imagens até mesmo os objetos dos modelos de ML.

Dados de Imagem

Para a utilização de dados de imagem, o package utilizado foi o CameraX. Este package traz uma API fácil e consistente para interagir com a câmara dos *smartphones* Android. Foram usados os métodos setUpCamera() e bindCameraUseCases() que permitem simultaneamente o preview da imagem da câmara e o processamento da mesma. Estes métodos serão abordados na próxima secção com mais detalhe.

Armazenamento de dados

Como base de dados externa para armazenar os datasets de treino e modelos de Machine Learning da aplicação foi escolhido o Firebase. Esta Base de Dados é gratuita, permite o armazenamento tanto de dados tabulares como de ficheiros (*object store*), é otimizada para IoT e dispositivos móveis e por isso indicada para este projecto.

Quanto aos dados gerados pelo modelo, havia como hipótese o armazenamento na base de dados local SQLite da aplicação ou em formato CSV. De forma a ir de encontro à visão da aplicação, em que o utilizador é o dono dos seus dados e por isso o acesso aos mesmos deve

ser simplificado, optou-se por usar o formato CSV, recorrendo ao *package open source "CSV Writer"*.

Sempre que é iniciada uma nova viagem (*'Trip'*), é criada uma nova pasta no directório Movies/AIDashCam/ e todos os artefatos gerados associados a esta Viagem são armazenados na mesma, de forma a facilitar o acesso.

Na figura 46 vemos o exemplo da estrutura de diretórios que armazena os dados de uma Viagem.

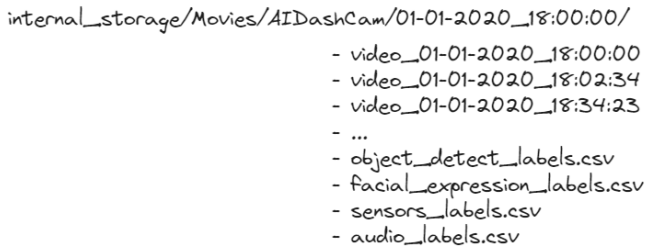

Figura 46 - Exemplo de estrutura do diretório de armazenamento de vídeos e dados

Numa versão mais recente da aplicação optou-se por converter os resultados dos modelos para CSV e guardar nesta pasta. No entanto, sendo que este diretório está definido numa variável no ficheiro strings.xml, o mesmo pode ser facilmente alterado no futuro.

Permissões

De forma a poder executar todas as funcionalidades, a aplicação necessita de ter permissões para aceder ou acionar diversos mecanismos do smartphone. Na figura 47 observamos uma seção do ficheiro XML *Android Manifest* onde são definidas quais as permissões necessárias para um correto desempenho da aplicação.

```
<uses-permission android:name="android.permission.CAMERA" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.POST_NOTIFICATIONS" />
<uses-permission android:name="android.permission.VIBRATE" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.RECORD_AUDIO" />
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
<uses-permission android:name="android.permission.SEND_SMS" />
```

Figura 47 - Gestão das permissões da aplicação

3.7.3 Módulos Funcionais

Nesta seção iremos abordar alguns dos principais módulos desenvolvidos e fazer uma profunda análise da sua implementação e das decisões tomadas.

Inferência dos Modelos de ML

Como abordado no início deste capítulo, foram treinados modelos de Machine Learning que serão usados pela aplicação. O código foi reaproveitado das aplicações de teste Tensorflow cuja licença é *open source*. De forma a garantir uma experiência customizada, existem alguns parâmetros que são expostos nas definições para que o utilizador possa adaptar a performance da aplicação ao seu smartphone e às suas necessidades. Estes parâmetros permitem controlar os recursos físicos do telefone que serão disponibilizados à aplicação que será elaborada com mais detalhe na próxima subsecção "Definições da aplicação".

Apesar de haver quatro tarefas de ML diferentes, o pipeline de inferência de cada um é semelhante tal como podemos observar na figura 48. Cada tarefa tem uma classe de suporte (por exemplo *ObjectDetectorHelper.kt*, *ImageClassificationHelper.kt* e *AudioClassificationHelper.k*t), também ela desenvolvida pela equipa do Tensorflow e que trata de todo o pipeline desde a captação dos dados - recorrendo à câmara para tarefas de imagem ou ao microfone para tarefas de áudio - passando pelo pré processamento dos dados, utilização do modelo e devolução dos resultados.

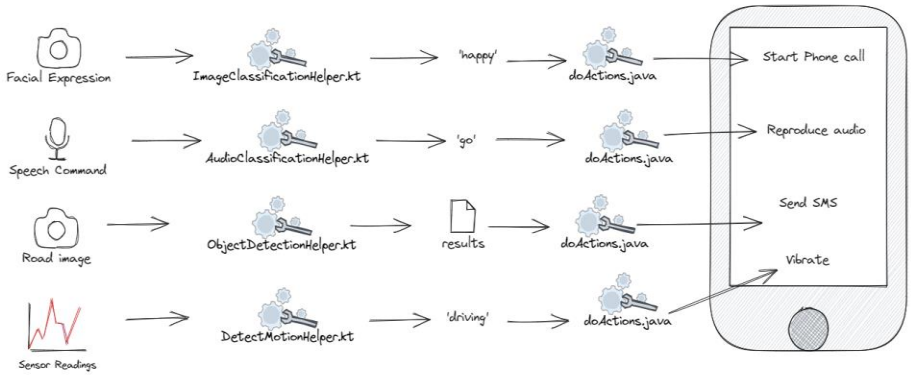

Figura 48 - Lógica funcional do sistema de ações

Na secção seguinte iremos aprofundar com maior detalhe o desenvolvimento deste módulo de ações desencadeadas.

Assistente de bordo

Para a implementação do assistente de bordo, tendo o modelo de classificação de áudio funcional e uma lista de ações encapsuladas numa função, apenas é necessário fazer o mapeamento entre a palavra-chave que resulta do modelo com a operação a desempenhar. Como referido anteriormente, para esta prova de conceito foram definidas três operações - iniciar gravação, terminar gravação e efetuar chamada de emergência - a serem desencadeadas por três palavras-chave diferentes. A implementação desta lógica é aplicada diretamente no fragmento principal da Dash Camera e segue a sintaxe descrita na figura 49.

```
when(results[0].label) {
    "go" → startRecord()
    "stop" → stopRecord()
    "down" → callPhone()
}
```

Figura 49 - Definição das ações a desencadear através de comandos

Como podemos observar, a estruturado código foi criada de forma a simplificar a futura implementação de novas operações a serem realizadas pelo assistente de bordo, sendo que apenas é necessário adicionar no seguimento das outras operações o mapa de "palavra-chave -> operação".

Ações

O módulo de Ações (*Actions*) tem como objetivo utilizar os recursos do telemóvel para de alguma forma interagir com o utilizador ou desencadear mecanismos de pedido de socorro - a ideia é que o utilizador possa parametrizar nas definições da aplicação que ações devem ser desencadeadas mediante um determinado resultado de um dos modelos.

Para isto, foi desenvolvida uma classe principal *DoActions.java*, apresentada na figura 50, que permite realizar as seguintes ações:

```
public void doAllActions(String categoryName) {
    try {
        doActionVibrate(categoryName);
        doActionPhoneCall(categoryName);
        doActionSMS(categoryName);
        doActionWhatsapp(categoryName);
        doActionEmail(categoryName);
        doActionScreenshot(categoryName);
        doActionPlaySound(categoryName);
        doActionRecordVideo(categoryName);
        doActionNotification(categoryName);
        doActionFlash(categoryName);
    } catch (Exception e) {
        Log.e( tag: "Action Error", e.getMessage());
    }
}
```

Figura 50 - Lista de ações desencadeadas

A função `doAllActions(category)`, tem como parâmetro de entrada a categoria (label) detetada pelo modelo, é chamada depois de cada modelo fazer inferência e procede à execução de todas as ações sequencialmente sendo que cada ação só será de facto desencadeada se cumprir a seguinte série de requisitos:

1. Utilizador ativou a ação nas definições.
2. Os campos de texto abertos estão devidamente preenchidos (Ex: email, telefone, duração, categoria, etc.).
3. A categoria detetada pelo modelo é igual à categoria definida pelo utilizador.
4. A ação não foi desencadeada nos últimos N segundos (valor a definir pelo utilizador nas Definições)

No exemplo da figura 51 observamos o código que desencadeia a ação de iniciar uma chamada.

Figura 51 - Exemplo da lógica que desencadeia uma ação

Podemos observar que de facto a ação (neste caso iniciar uma chamada) só é de facto desencadeada - na linha 193 - caso satisfaça a série de condições definidas entre as linhas 180 e 185. Para esta ação, estas seis condições são:

1. Existência de um objeto `settingActionModel` que contém as várias definições definidas pelo utilizador para parametrizar as ações associadas a um determinado modelo
2. O método `enabledPhoneCall` tem de retornar a variável booleana no estado `True` que determina que o utilizador ativou esta ação nas definições.
3. A variável `phoneCallWhen` contém a classe especificada pelo utilizador que quer que seja detetada. Por defeito esta classe é uma string vazia pelo que esta condição, ao exigir que o nome da classe tenha mais que 0 caracteres, implica que houve uma classe a ser especificada. Nota: esta condição foi implementada desta maneira pois inicialmente este campo era de texto aberto. Numa versão mais recente foi atualizada para uma lista com as classes disponibilizadas pelo modelo para ser mais intuitivo para o utilizador.
4. Esta condição compara a categoria especificada pelo utilizador nas definições com a categoria detetada pelo modelo. A opção '==' seria a mais correta de utilizar no entanto estava a originar alguns problemas pelo que se alterou para a função '*contains*' que embora menos eficiente, garante uma maior fiabilidade pois é mais robusta contra possíveis caracteres escondidos.
5. Esta condição implica que o número de telefone tenha no mínimo 1 caracter para garantir que o utilizador especificou um contacto.
6. Por fim, a última condição garante que a ação só é desencadeada caso tenha sido passado o período mínimo de intervalo entre a mesma ação definido pelo utilizador em `actionInterval`.

Caso todas as condições sejam satisfeitas, a ação vai então ser desencadeada. Cada acção tem as suas particularidades, mas de forma geral o que acontece no processo de desencadeamento da ação é a atualização do valor da variável *lastActionVibrate* para o timestamp *'now'*, definir o objecto de acção ou *Intent*, parametrizar este objeto com as definições necessárias (durações, contactos, etc.) e, por fim, a mesma é efetivada.

A lista de 11 ações implementadas descritas no seguimento desta secção permitem ao utilizador uma maior customização da sua experiência. Esta lista de ações é versátil e divide-se em 3 grupos:

Avisos ao Condutor

1. **Ativar Vibração** - o utilizador pode definir para cada modelo uma categoria a ser detetada e a duração da vibração em segundos.
2. **Reprodução de Áudio** - o utilizador insere uma categoria e seleciona um ficheiro de áudio a ser reproduzido no caso de a categoria ser detetada.
3. **Notificação** - o utilizador insere uma categoria e é recebida uma notificação do sistema sempre que a mesma é detetada.
4. **Ligar Flash** - o utilizador insere uma categoria e a duração do flash em segundos

Comunicação

5. **Iniciar Chamada** - pode ser definido um número de telefone e uma categoria para cada modelo.
6. **SMS** - pode ser definido um recipiente e uma categoria para cada modelo, o corpo da mensagem é definido no ficheiro strings.xml
7. **WhatsApp message** - idêntico à SMS mas utiliza a aplicação WhatsApp para o envio da mensagem (em desenvolvimento)
8. **WhatsApp call** - (em desenvolvimento)
9. **Envio de Email** - o utilizador insere um email e uma categoria a ser detetada para cada modelo.

Registo de evidências

10. ***Screenshot*** - o utilizador insere uma categoria para cada modelo e caso a mesma seja detetada é tirado um screenshot.
11. **Gravação de Vídeo** - o utilizador insere uma categoria e a duração do vídeo a ser gravado e a câmara - frontal ou traseira - no caso da categoria ser detetada.

Na pesquisa efetuada no Estado da Arte, muitos utilizadores de aplicações semelhantes afirmaram estar a utilizar um telefone mais antigo para uso exclusivo da aplicação. Sendo então possível que o condutor possa optar por ter um telefone dedicado para a *Dash Camera*, foram desenvolvidas ações de comunicação que dispensam um cartão SIM, tais como chamadas e mensagem WhatsApp, permitindo assim ao condutor fornecer internet através de um hotspot de outro smartphone e assim não perder estas capacidades de segurança.

Este módulo foi desenvolvido de forma que haja liberdade para que cada utilizador possa parametrizar a aplicação de acordo com as suas necessidades. Alguns exemplos:

- Se o modelo de Classificação de Movimento detetar a categoria 'hard brake' -> gravar vídeo durante 20 segundos.
- Se o modelo de Reconhecimento de Expressão Facial detetar a categoria 'tired' -> emitir um áudio de alerta.
- Se o modelo de Deteção de Sinais Rodoviários detetar um sinal de stop -> emitir um áudio de alerta.

De momento não é possível a utilização de múltiplas ações para o mesmo modelo nem a deteção de mais do que uma categoria por modelo em simultâneo. Esta implementação apesar de ser relativamente fácil, irá trazer processamento adicional e requer ser testada devidamente para garantir que não afeta a performance da mesma.

Definições

As definições da aplicação permitem a customização do comportamento da aplicação a cada utilizador. Estas dividem-se em dois grupos distintos: as definições dos modelos e as definições das ações. Relativamente aos modelos, cada uma das quatro tarefas - classificação de movimento, deteção de objetos na estrada, reconhecimento de expressão facial e assistente de bordo - apresenta as seguintes opções:

1. **Modelo**: o utilizador pode selecionar um dos vários modelos disponíveis para a realização da tarefa em questão sendo que o que os diferencia é o grau de performance e precisão do modelo (maior performance consome mais recursos e bateria).
2. **Hardware**: NN API (Neural Network API), CPU ou GPU. O utilizador pode escolher qual o componente de hardware pretendido desde que o mesmo esteja disponível no telefone. Futuramente esta opção deverá ser automatizada para detetar os hardwares existentes e automaticamente escolher o melhor (NN API > GPU > CPU).

3. **Thresholds**: grau de certeza do modelo entre 0 e 1 - aumentando este valor irá resultar num modelo mais preciso (apenas válido para o assistente de bordo e deteção de objetos na estrada) e assim reduzir o número de Falsos Positivos.
4. **Número de threads:** número de tredas do processador disponibilizadas que o modelo pode utilizar para efetuar o processamento dos dados e inferência. Também este passo poderá eventualmente ser ou automatizado ou convertido em algo mais percetível ao utilizador comum, como por exemplo uma barra de 1-4 cuja descrição é por exemplo "Recursos do Smartphone".
5. **Categoria**: Para cada tarefa, podemos ter diversas categorias a desempenhar diferentes ações. Isto pode ser facilmente mapeado como podemos observar na figura 52.

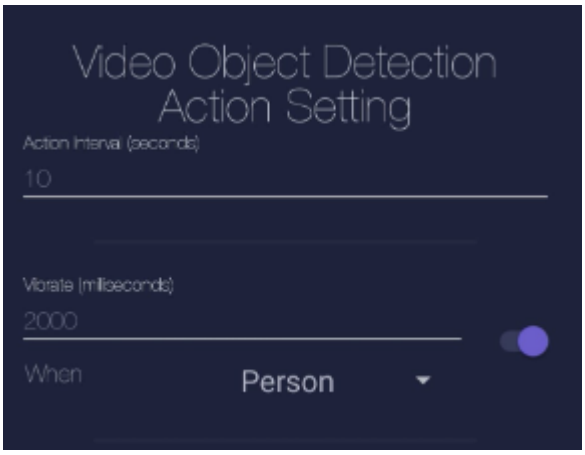

Figura 52 - Exemplo dos parâmetros de uma ação

Adicionalmente, cada modelo tem uma lista de ações que podem ser ativadas ou desativadas através de um botão *toggle*. Estas ações podem ser parametrizadas através dos seguintes campos:

1. **Intervalo de Acão**: de forma a evitar um contínuo trigger da ação, como por exemplo a deteção de um sinal de stop quando o carro está parado, é definido um tempo de Intervalo de ação para garantir que os avisos são devidamente espaçados
2. **Tempo de Vibração**: duração da vibração em segundos
3. **Contacto**: contacto a ser utilizado nas ações de chamadas, SMS e WhatsApp.
4. **Email**: email a ser utilizado na ação de envio de email com as coordenadas atuais
5. **Ficheiro Áudio**: Ficheiro de áudio a reproduzir no caso de a ação de aviso sonoro ser ativada
6. **Duração de gravação de Vídeo**: duração em segundos da ação de gravação automática de vídeo
7. **Duração do Flash:** Duração do flash ligado em segundos.

Vimos nesta seção e nas anteriores como foram implementadas as ações e como podem ser parametrizadas nas definições bem como mapeadas com os modelos desejados. Sendo que alguns dos termos e conceitos não são de linguagem comum, ficará para trabalho futuro por exemplo a implementação de uma caixa de texto flutuante que aparece quando o utilizador clica na definição que pretende saber mais informações.

Funcionalidades Adicionais

Apesar de não relacionadas com o conceito da aplicação, existem algumas funcionalidades básicas que qualquer *Dash Camera* deve ter implementadas. Sendo o objetivo desta aplicação superar as limitações das aplicações existentes, é também necessário que a mesma seja desenvolvida de forma a não apresentar limitações face às existentes. Desta forma, foram adicionadas duas funcionalidades:

1. **Localização:** através das coordenadas GPS obtidas diretamente dos sensores do telefone através da classe nativa *Location*, juntamente com o package open source *Geocoder* que, com base numa latitude e longitude, devolve o endereço correspondente.
2. **Velocidade:** também através das classes *Location*, *LocationManager* e *LocationListener,* a velocidade é recolhida diretamente através da variável *speed* e convertida de Milhas para Kms por hora. Futuramente nas definições o utilizador poderá alterar qual a escala pretendida ou pode ser automaticamente parametrizado conforme a escala padrão no país do utilizador.
3. **Nível de Bateria:** através da classe *BatteryManager* conseguimos obter a capacidade atual da bateria. Este método foi implementado para conseguir medir o nível e o *drain rate* da bateria em tempo real de forma a poder sugerir automaticamente a adaptação de modelos mais ajustados às capacidades do telefone e à carga atual da bateria. Para o trabalho futuro fica a implementação de um modo de gestão inteligente que permita ao utilizador especificar um *drain rate* máximo e um limite mínimo de bateria - por exemplo 10%/hora e 25% - ficando a aplicação encarregue da otimização da gestão dos recursos físicos e da escolha dos níveis de modelos das diversas tarefas.

Ambas estas funcionalidades serão apresentadas com maior detalhe na secção do fragmento da funcionalidade Dash Camera da aplicação.

UI screens

Como referido anteriormente, o design da aplicação não foi priorizado nesta fase do projeto por forma a dar preferência a desenvolvimentos funcionais e redação do documento.

Futuramente o UI e experiência de navegação na aplicação serão melhorados após uma fase de testes com um grupo de utilizadores restrito. No seguimento da secção podemos observar os principais ecrãs da aplicação apresentados na figura 53.

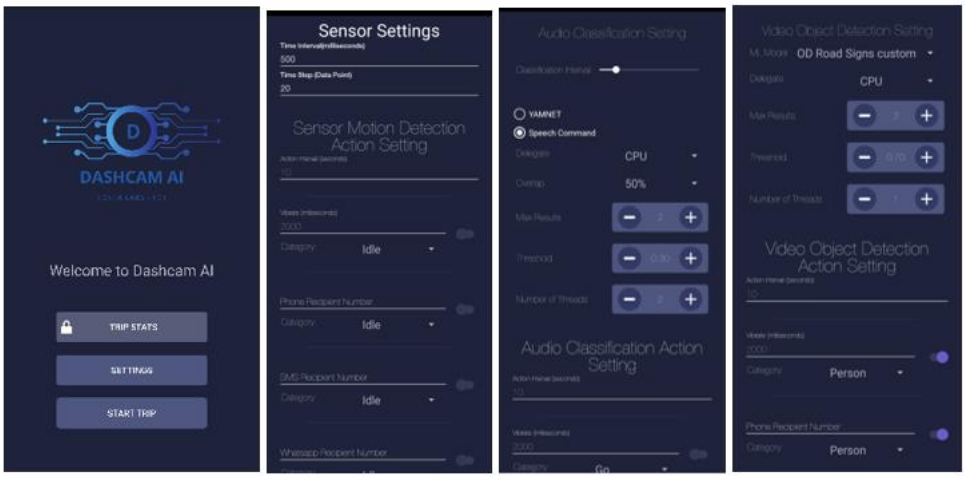

Figura 53 - Ecrã inicial e de Definições da aplicação

3.7.4 Dash Camera

Considerado o coração da aplicação, este módulo foi desenvolvido sob a forma de um fragmento e junta todas as tarefas a correr de forma assíncrona. Nesta seção irão ser descritos todos os passos desta funcionalidade bem como uma detalhada explicação sobre o seu comportamento.

Iniciamos com a definição de uma classe denominada *AIDashCamFragment*, que extende a classe *SensorBaseFragment*. Esta classe extendida foi desenvolvida no início do projeto e implementa a interface de SensorEventListener, que de forma resumida tem como objetivo a interação com os sensores do smartphone, sendo responsável por adquirir o estado atualizado de cada um dos 59 sensores definidos para que o modelo de Classificação de Movimento possa usar para a inferência.

Definida a classe, vão ser definidas todas as variáveis necessárias para o funcionamento da mesma, incluindo as inerentes às diferentes tarefas.

Estando definidas as variáveis e feito o setup dos objetos que irão ser responsáveis pela execução das diferentes tarefas de Machine Learning, são definidos os vários estados do *lifecycle* do fragmento bem como os procedimentos a desencadear em cada um.

1. **onAttach** - Este método é chamado quando o fragmento é associado pela primeira vez ao seu *context*. Neste estado é criado um objecto de classe *SettingModel* que irá conter

todas as variáveis parametrizadas pelo utilizados nas Definições da aplicação, tanto dos Modelos como das Ações.

2. **onCreateView**: Este método é chamado para renderizar o layout do fragmento definido no ficheiro XML com o mesmo nome. Este layout é definido através de um View Binding, um método comum para gerir as ligações entre o ficheiro de layout e o ficheiro de lógica.
3. **onViewCreated**: Neste estado assume-se que o layout já foi criado. Várias inicializações são então feitas:
    a. Criação da diretoria onde vão ser registados os vídeos e ficheiros dos resultados dos modelos.
    b. Inicialização dos ficheiros de *Save Helper Functions* criados para cada uma das tarefas. Estes ficheiros contêm todas as variáveis que vão auxiliar na gravação dos resultados dos modelos em ficheiros CSV, tais como os nomes das colunas e o sufixo do nome do ficheiro.
    c. Um *single-thread* executor dedicado exclusivamente à câmara dado o seu elevado processamento
    d. Definidos os botões de iniciar e parar a gravação bem como o botão de aceder às definições da aplicação.
    e. Carregamento do modelo de classificação de movimento.
4. **onResume**: Este método é chamado no momento em que o layout é renderizado e fica visível para o utilizador. São então realizadas vários processos:
    a. O objeto criado no método onAttach() vai agora carregar todas as definições parametrizadas pelo utilizador - este passo tem de acontecer neste estado para permitir ao utilizador o acesso às definições a meio da Viagem e aplicar essas alterações com efeito imediato quando retornar ao fragmento da Dash Camera.
    b. A câmera é inicializada através do método setUpCamera() que usa o package CameraX como referido anteriormente. Após esta inicialização, é definido um listener que vai ser desencadeado assim que a câmara estiver disponível. Estando o setup da câmara terminado, é executado um segundo método *bindCameraUseCases* que vai fazer o binding dos use cases mencionados anteriormente - preview da imagem e análise/processamento da mesma - utilizando um *cameraProvider* e um *cameraSelector*.
    c. São criados os objetos de *Helper Functions* para as diferentes tarefas recorrendo aos parâmetros passados no objecto *SettingModel* acima definido. Estas classes foram reutilizadas dos tutoriais fornecidos na documentação do Tensorflow e

têm licença aberta para utilização e comercialização pelo que o seu uso nesta aplicação é totalmente legítimo.

   d. Adicionalmente são definidos os objetos de desencadeamento de ações - como por exemplo mDoActionsVideo e mDoActionsAudio

5. **onPause**: Este método é chamado quando o utilizador muda de fragmento e o mesmo deixará de estar visível (por exemplo quando o utilizador acede às Definições a meio da Viagem). Neste são executadas as seguintes operações:
   a. *Running Flags* são colocadas a Falso; isto irá fazer com que certas funcionalidades da aplicação que só devem funcionar com o fragmento em resumo sejam interrompidas
   b. Toda a captação de dados de imagem, som e sensores é interrompida e a câmara é fechada
   c. As threads criadas para funcionar em background são também encerradas bem como o *cameraExecutor*.
6. **onDestroyView**: Este método é chamado quando a *View* associada ao fragmento é destruída e o fragmento é destruído. Neste estado é feita uma "limpeza" dos recursos utilizados por esta *View* para evitar processos abandonados que ocupem espaço de memória e prejudiquem o bom funcionamento da aplicação.

O resultado deste módulo pode ser observado na figura 54 onde vemos um exemplo da informação captada pela aplicação bem como os resultados dos modelos obtidos e as informações adicionais tais como a velocidade e a localização.

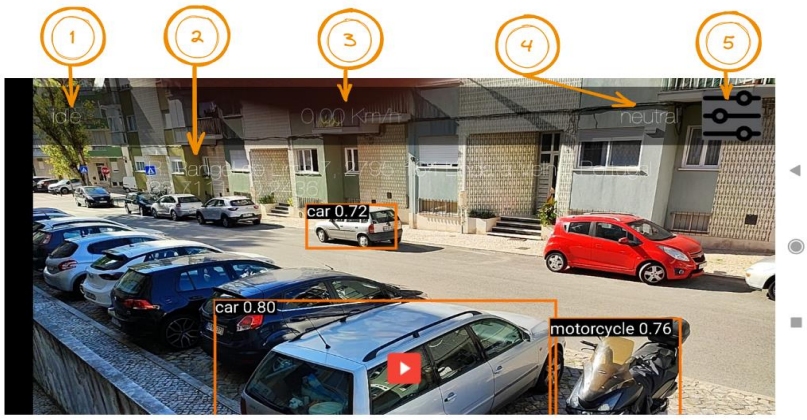

Figura 54 - Módulo Dash Camera e informação da camada de Overlay

Na figura 54 podemos observar uma imagem de uma rua onde a aplicação deteta alguns objetos como carros e motas (está nas definições limitada a um máximo de 3 objetos). Adicionalmente podemos ver no topo as seguintes informações:

1. Resultado do modelo de Classificação de movimento
2. Localização atual (morada e coordenadas GPS)
3. Velocidade atual
4. Resultado do modelo de Reconhecimento de Expressão Facial
5. Definições

A informação é de difícil leitura pois como referido anteriormente não foi priorizado o design e experiência do utilizador, futuramente algumas alterações irão ser feitas de modo a proporcionar uma melhor experiência.

4

TESTES E VALIDAÇÃO

Terminada a implementação da aplicação, iremos nesta seção fazer testes extensivos sobre a sua utilização. Iremos testar a performance dos modelos individualmente bem como a aplicação como um todo.

4.1 Testes no mundo real

Os testes no mundo real são cruciais para validar a eficácia e a fiabilidade da aplicação e dos modelos de Machine Learning incorporados. Esta seção descreve diversas metodologias de teste que podem ser aplicadas para avaliar o desempenho, a precisão e a experiência do utilizador na aplicação em cenários do mundo real.

4.1.1 Teste dos Modelos

Reconhecimento de Expressão Facial

O teste de reconhecimento de expressão facial mostrou que o modelo não funcionou tão bem quando o telefone estava a executar a aplicação no dispositivo de suporte. Como referido anteriormente, isto era previsível e pode ser facilmente explicado comparando amostras de imagens do dataset de treino do modelo com aquelas que o modelo vai ver no "mundo real", tal como observamos na figura 55.

Figura 55 - Imagens do dataset FER 2013 (em cima) e imagens do dataset customizado (em baixo)

A segunda abordagem foi feita usando imagens de familiares e amigos enquanto conduziam sendo que por forma a tornar o processo mais simples, as classes eram apenas "sleeping" e "not sleeping". No entanto, mesmo para um modelo básico são necessárias largas centenas de imagens de cada categoria para que o modelo consiga aprender e generalizar.

Assistente de bordo

O assistente de bordo demonstrou resultados muito positivos, com cada comando especificado a desencadear a ação correta com bastante precisão. Como as palavras-chave escolhidas são palavras comuns no vocabulário inglês, o modelo irá desencadear as ações sempre que o condutor – ou até mesmo algum passageiro – disser a palavra-chave no meio de uma conversa. Idealmente, teríamos um dataset com palavras-chave personalizadas que não fossem de linguagem corrente para evitar o excesso de Falsos Positivos, ou seja, ações a serem desencadeadas por engano.

Classificação de Movimento

Este modelo foi treinado com um dataset extremamente pequeno, somando todas as amostras pouco mais de 5 minutos. No entanto, como observado na secção de Desenvolvimento dos Modelos, ainda assim foram obtidos resultados bastante aceitáveis com uma precisão de 85%.

O modelo no geral funciona de forma correta e consegue detetar a classe de movimento com sucesso durante grande parte do tempo de percurso. Sendo que o modelo está a classificar o movimento a cada 100ms, o facto de a precisão não estar muito perto dos 100% resulta em algum ruído indesejado, especialmente se tivermos em consideração que esse ruído pode ser responsável pelo desencadeamento de uma ação.

Na secção de Resultados e Conclusões irá ser detalhado o problema detetado com este modelo e as possíveis soluções para a mitigação do mesmo

Deteção de Objetos na Estrada

Este modelo é o que mais recursos do telefone consome pois não só necessita de em tempo real utilizar a câmara para recolher a imagem como também realizar um preprocessamento, utilizar o modelo para detetar os objetos e renderizar as *bounding boxes* no ecrã. A sua implementação nesta aplicação é absolutamente indispensável pois devido à sua capacidade de detetar objetos e situações potencialmente perigosas, é das que apresenta maior potencial preventivo.

Os modelos detetam com bastante sucesso todos os objetos que aprenderam durante o treino. Apesar da métrica utilizada mAP rondar os 70% no melhor modelo, esta métrica no mundo real não reflete o verdadeiro desempenho do modelo. Tendo em conta que o objetivo deste modelo é detetar um determinado objeto num *frame*, o mesmo na realidade tem várias "hipóteses" - os vários frames - para conseguir detetar o objeto. Se o processamento for por exemplo a 60 *frames* por segundo (FPS), significa que em 1 segundo o modelo terá 60 oportunidades de detetar corretamente o objeto pretendido sendo que basta uma vez para ser considerado sucesso. Após uma pequena pesquisa efetuada, não foi encontrado nenhum método de validação deste tipo de modelos no mundo real.

4.1.2 Cenários de teste

Para testar a aplicação no mundo real, foram definidos alguns cenários de teste com o objetivo de se poder verificar os requisitos funcionais especificados anteriormente. Na tabela 9 observamos uma lista de testes com o respetivo resultado.

Teste	Objetivo	Resultado
Travagem de Emergência	Aplicar uma travagem numa condução de elevada velocidade e ligar automaticamente para o número de emergência	O modelo detetou com sucesso a travagem brusca. No entanto, durante a condução, o modelo gerou alguns falsos positivos, resultando em chamadas de emergência indesejadas. Para resolver isso, a sensibilidade do modelo foi ajustada e reduz a quantidade de falsos positivos.
Reconhecimento de Comando	Uso dos vários comandos de voz para controlar as funcionalidades da aplicação	Todos os comandos de fala funcionaram perfeitamente. As palavras-chave também foram testadas numa simulação de fala normal em português e nunca foram acionadas. No entanto, ao falar em inglês, em alguns casos a aplicação reconheceu a palavra-chave no meio de uma frase e acionou o aviso. Isto pode ser resolvido com um dataset personalizado de comandos que não façam parte do corpus de palavras mais usadas em inglês.

Deteção de sinal de Stop	Detetar um sinal de stop e automaticamente reproduzir um áudio de aviso.	O modelo Deteção de Objetos na Estrada fez um trabalho notório na deteção dos objetos treinados. Nunca durante os testes deu um falso positivo para nenhum dos objetos e detetou a placa de Stop em 8 locais diferentes (por vezes apenas passado alguns frames).
Drenagem de Bateria	Testar quanto tempo dura o smartphone a utilizar a aplicação com os melhores modelos e com as versões light.	Executar 4 modelos em tempo real consome muitos recursos. Mesmo com os modelos altamente otimizados, a bateria descarrega rapidamente. Para um uso adequado da aplicação, é recomendável ter o telefone ligado ao carregador do carro.
Funcionamento offline	Usar a aplicação e testar todas as funcionalidades offline	Todos os recursos da aplicação funcionam sem ligação à internet. Desligando o WiFi e os dados móveis, conseguimos abrir a aplicação, iniciar uma viagem, utilizar todos os modelos com sucesso, acionar todas as ações e registar os dados da viagem e aceder aos mesmos.

Tabela 9 - Testes da aplicação no mundo real

Realizados os testes à aplicação, concluímos que a sua performance é bastante aceitável tendo em conta que é apenas uma prova de conceito. Na secção de Análise Crítica e Conclusões irá ser detalhado os problemas detetados na aplicação bem como a implementação de uma estratégia para a sua resolução.

4.2 Verificação dos requisitos

Na tabela 10 iremos observar os requisitos funcionais presentes na proposta de desenvolvimento da aplicação e averiguar os que foram cumpridos com sucesso e os que ficaram por implementar.

ID	**Requisito Funcional**	**Resultado**
R.F.1	Padrões de Condução	Implementado
R.F.2	Deteção de Objetos na Estrada	Implementado
R.F.3	Reconhecimento da Expressão do Condutor	Implementado
R.F.4	Assistente de Bordo	Implementado
R.F.5	Registo	Parcialmente Implementado
R.F.6	Processamento em Tempo Real	Implementado
R.F.7	Localização	Implementado

R.F.8	Velocidade	Implementado
R.F.9	Alertas	Implementado
R.F.10	Emergência	Implementado
R.F.11	Gravação do percurso	Implementado
R.F.12	Customização	Implementado
R.F.14	Gravação dos dados	Implementado
R.F.15	Adaptação	Implementado
R.F.16	Feedback	Não Implementado

Tabela 10 - Verificação dos Requisitos Funcionais

A grande parte dos requisitos funcionais foram implementados com sucesso. Não foi criado um registo de utilizador pois esta funcionalidade não é crítica nesta fase do projeto. Por fim, o botão que permite ao utilizador reportar bugs ou solicitar novas funcionalidades ou dar feedback não foi ainda implementado pois a aplicação ainda não foi disponibilizada para outros utilizadores.

Procedemos agora a uma análise dos requisitos não funcionais listados na tabela 11.

ID	**Requisito Não Funcional**	**Descrição**
R.N.F.1	Privacidade	Todos os dados devem ser processados localmente de modo a assegurar a total privacidade e segurança do utilizador
R.N.F.2	Baixa Latência	Os modelos devem apresentar uma latência de pelo menos 2 FPS de modo a produzir os alertas antes de o condutor detetar o perigo
R.N.F.3	Funcionamento offline	A aplicação deverá ser 100% funcional mesmo em áreas sem ligação à internet
R.N.F.4	UI apelativa	A aplicação deverá ter uma Interface de Utilizador intuitiva e de fácil controlo
R.N.F.5	Escalabilidade	A aplicação deverá permitir escalabilidade independentemente do número de utilizadores
R.N.F.6	Fiabilidade	A aplicação deverá ser fiável sob diversas condições assegurando sempre um desempenho consistente
R.N.F.7	Eficiência	A aplicação deverá ser eficiente de modo a ter um bom desempenho em qualquer smartphone sem consumir demasiados recursos e bateria
R.N.F.8	Compatibilidade	A aplicação deverá ser compatível com uma vasta gama de smartphones Android com diferentes versões
R.N.F.9	Customização	A aplicação deverá permitir a cada utilizador moldar a mesma às suas necessidades

R.N.F.10	Interoperabilidade	A aplicação deverá permitir de forma simples a partilha dos dados gerados com outras aplicações
R.N.F.11	Atualizações	A aplicação deverá suportar atualizações do código e dos modelos de forma fácil e segura

Tabela 11 - Verificação dos Requisitos Não Funcionais

A aplicação foi toda desenvolvida sem que seja necessário que os dados do utilizador saiam do telefone garantindo assim uma mais privacidade. Os modelos apresentam uma baixa latência mesmo com telefones mais antigos – utilizando os modelos mais leves – conseguindo sempre alcançar um mínimo de 2 FPS de dados processados garantindo assim um processamento mais rápido que a média de tempo de reflexo do condutor.

A aplicação funciona 100% offline e dispõe de uma UI bastante simples e intuitiva, embora alguns termos mais complexos possam não ser tão compreensíveis para alguns utilizadores. Sendo que a aplicação faz todo o processamento localmente no smartphone, a escalabilidade pode ser alcançada de forma bastante fácil uma vez que não há nenhum processo a ser servido de fora.

A eficiência da aplicação poderá ainda ser melhorada sendo que a mesma será sempre muito dependente dos recursos físicos do smartphone.

A aplicação foi testada em 3 smartphones diferentes e apesar de algumas pequenas diferenças, a grande parte das funcionalidades core funcionaram da mesma forma e sem nenhum problema. O utilizador consegue através das definições moldar a aplicação às suas necessidades ao ser possível a escolha dos modelos e da lógica da ação a desencadear.

Os dados podem ser facilmente extraídos do disco do telefone ou partilhados por email.

Por fim, as atualizações da aplicação poderão ser feitas de forma simples uma vez que foi utilizado versionamento de código, fica em falta a montagem de um robusto grupo de testes (*unit tests*, *functional tests* e *integration tests* que permita o desenvolvimento de novas funções e uma passagem fácil para produção.

4.3 Validação do sistema

Nesta secção iremos fazer uma avaliação da aplicação de um ponto de vista global. A mesma foi desenvolvida com uma visão baseada em lacunas detetadas em aplicações anteriores e por isso queremos ver de um ponto de vista do utilizador se a solução apresentada neste trabalho colmatar essas lacunas.

De uma forma geral, a aplicação é capaz de reconhecer o ambiente em torno da condução bem como as capacidades do condutor. Estes dados permitem de uma forma proactiva emitir avisos ao condutor ou desencadear processos de emergência que garantam ao condutor uma possibilidade de pedir socorro recorrendo a um simples comando de voz.

Relativamente aos dados, foi respeitada a política proposta para a privacidade não sendo nenhum dado partilhado para fora do smartphone, sendo o utilizador a única entidade com acesso aos mesmos, podendo-o fazer de forma extremamente fácil ao consultar no disco do smartphone a pasta referente à Viagem em questão e podendo aceder tanto aos vídeos gravados como aos resultados dos modelos, em ficheiros CSV, que trazem informação sobre a expressão facial, classificação de movimento, objetos detetados e comandos de voz acionados durante todo o percurso.

De um ponto de vista funcional a aplicação tem pouca autonomia dado o uso exaustivo dos recursos físicos pelo que o seu uso sem estar ligado a uma fonte de alimentação pode ser bastante limitado. No entanto, dado as funcionalidades asseguradas e sabendo que qualquer carro tem a possibilidade de ter uma fonte de alimentação, acreditamos que este problema poderá não ser um fator crítico na adoção desta ferramenta.

4.4 Análise crítica e reflexão

Nesta caminhada para o desenvolvimento de uma solução de *Dash Camera* inteligente para dispositivos Android, foi averiguada a integração de múltiplas funcionalidades de Inteligência Artificial com a finalidade de revolucionar a experiência destas aplicações.

Os resultados do projeto, embora predominantemente bem-sucedidos, esclarecem também algumas áreas de melhoria e desafios enfrentados durante a fase de implementação. Aqui, apresentamos uma visão abrangente dos sucessos e das lições aprendidas.

4.4.1 Sucessos

Classificação de movimento do veículo: Embora com algum ruído derivado do reduzido tamanho de dataset de treino, a aplicação foi capaz de identificar com uma precisão aceitável o movimento do veículo, garantindo que os condutores estão bem conscientes do que os rodeia e podem ser avisados se o seu comportamento de condução for agressivo.

Assistente de Bordo: A integração do Assistente de Bordo com comando de voz foi uma conquista significativa. Este recurso permitiu que os condutores controlassem o dispositivo com as mãos livres, garantindo assim máxima segurança e comodidade. Futuramente vários outros comandos poderão ser integrados de forma bastante simples.

Deteção de objetos e sinais de trânsito: A capacidade da aplicação de detetar objetos e sinais de trânsito na estrada funcionou acima das expectativas. Este recurso auxilia na tomada

de decisões em tempo real, potencialmente prevenindo acidentes e garantindo o cumprimento das regras de trânsito. Apesar de os atuais objetos utilizados para treinar o modelo serem já serem relevantes, futuramente podemos ter um modelo capaz de detetar e identificar os diferentes sinais de trânsito, analisar as placas de limite de velocidade e avisar o condutor caso esteja a conduzir acima da velocidade permitida, detetar buracos na estrada ou animais, bem como outros objetos ou entidades que possam também ser relevantes para este efeito.

Ações: Todas as ações foram implementadas com sucesso permitindo ao utilizador através das definições da aplicação quais as ações que deseja acionar com base nos resultados dos modelos. Isto permite uma experiência de utilizador melhor e personalizada e em casos de emergência permite ao utilizador, por exemplo, acionar automaticamente uma chamada de emergência e enviar um SMS com a localização atual. Esta funcionalidade muda o paradigma deste tipo de aplicações ao deixarem de ser meramente uma câmara que passivamente regista o que se passa na estrada passando a ser como um copiloto que ajuda na prevenção de acidentes e é capaz de agir em caso de emergência.

4.4.2 Áreas para melhoria

Modelo de reconhecimento de expressão facial: Embora o modelo de reconhecimento de expressão facial tenha sido projetado para detetar as expressões faciais do condutor, o mesmo enfrentou alguns desafios durante os testes no "Mundo Real". Como mencionado anteriormente, o principal problema foi que os dados de treino consistiam em fotografias tiradas de um ângulo diferente daquele encontrado pela câmara nos cenários do mundo real e desta forma o modelo não consegue fazer um bom trabalho. No entanto, nesta prova de conceito mostrámos que é possível ter um modelo a identificar a expressão facial do condutor em tempo real e desencadear uma ação no caso de ser detetada uma determinada expressão. A solução para este problema passa por arranjar um dataset diversificado de imagens de condutores com as expressões necessárias para a implementação desta funcionalidade.

Ausência de dados de sensores da categoria 'crash': Sendo inicialmente a principal categoria de movimento a detetar, não foi possível proceder à implementação da mesma, pois teriam que ser realizadas inúmeras simulações de acidente para recolher os dados dos sensores e devido aos riscos inerentes e às considerações éticas, não foi possível realizar. Como resultado, o modelo de classificação de movimento não tem a capacidade de detetar acidentes rodoviários. Existem algumas soluções que poderiam ajudar a mitigar este problema:

1. **Sistema de regras** - sendo que um acidente de carro provoca movimentos não naturais, poderíamos tentar modelar com regras as potenciais situações de acidente. Por exemplo, sabemos que quando batemos de frente é de esperar um diferencial do eixo do X do acelerómetro muito grande num curto espaço de tempo ou então quando o carro perde o controlo é de esperar uma rotação (giroscópio) rápida acompanhada de uma velocidade linear constante. Isto são alguns exemplos que poderiam ser modelados e não requerem nenhum modelo, apenas o processamento dos dados obtidos dos sensores em tempo real.
2. **Modelos não supervisionados -** uma vez que estes modelos não requerem dados de treino com *label,* podiam ser equacionados para solucionar este problema. Como visto no estado da arte, existe uma tarefa de modelos não supervisionados denominada "Deteção de Anomalias" e a mesma consiste num modelo que aprende um conjunto de dados e na altura da inferência procura padrões que nunca tenha visto. Neste caso, se dermos ao modelo um dataset de treino com largas horas de condução segura sem nenhum acidente, o modelo vai aprender estes padrões e, hipoteticamente, quando houver um acidente de viação, o modelo irá observar padrões anómalos nas leituras dos sensores e desta forma identificar uma situação de acidente. Esta hipótese será estudada no trabalho futuro.

Um desafio recorrente enfrentado durante o projeto foi a qualidade dos dados utilizados para o treino dos modelos. Embora este projeto seja apenas uma prova de conceito mais generalizada e os modelos terem obtido um desempenho razoavelmente bom, tornou-se evidente que, com dados de maior qualidade, o desempenho dos mesmos poderia ser significativamente melhor. As iterações futuras deste projeto devem priorizar a recolha de dados de melhor qualidade, maior quantidade e mais diversificados para refinar ainda mais os modelos.

O empenho neste trabalho para criar uma solução de aplicação móvel Dash Camera acessível a todos os utilizadores foi marcado por sucessos significativos, com algumas áreas destacadas para melhorias. As realizações do projeto sublinham o potencial destas tecnologias no aumento da segurança rodoviária e na comodidade do condutor. Enquanto isto, os desafios enfrentados servem como lições valiosas, dando ênfase à importância de dados de alta qualidade e a necessidade de refinamento contínuo em aplicações do mundo real.

5

CONCLUSÕES E TRABALHO FUTURO

5.1 Síntese de Trabalho Desenvolvido e Resultados Obtidos

Conforme visto antes, existe um elevado número de fatalidades e ferimentos graves todos os anos derivado de acidentes rodoviários. Foram também abordadas as principais causas sendo que o erro humano, como a condução agressiva ou distraída, representam uma grande parte destes cases.

Este projeto visou contribuir para uma melhoria da segurança no tráfego, através do desenvolvimento de uma aplicação móvel que simula uma *Dash Camera* e que usa modelos de Machine Learning que processam dados localmente, oferecendo várias vantagens, incluindo uma maior privacidade e segurança dos dados. A capacidade da aplicação de desencadear ações de forma autónoma adiciona uma camada extra de segurança, tornando-o um passo revolucionário neste tipo de aplicações.

A arquitetura da aplicação, otimizada para dispositivos móveis, garante um processamento eficiente em tempo real de dados de sensores, câmaras e áudio. O processamento local destes dados oferece vantagens substanciais em termos de privacidade e segurança dos dados, bem como um total funcionamento independentemente de uma ligação à internet. Adicionalmente, a capacidade da aplicação de desencadear ações de forma autónoma, como enviar mensagens SMS ou iniciar chamadas telefônicas, adiciona uma camada extra na prevenção de acidentes.

Os testes no mundo real validaram a eficácia da solução apresentada, demonstrando uma baixa latência de inferência dos modelos com uma precisão bastante satisfatória. A solução desenvolvida foi capaz de detetar diversas situações de perigo, tais como determinados estados emocionais do condutor ou padrões de condução agressiva e desencadear avisos para incentivar o condutor a ter melhores práticas de condução. Também os testes de desencadear mecanismos de emergência funcionaram com sucesso, tendo os mesmos sido testados na deteção de uma travagem brusca – uma vez que a classe "crash" não pode ser desenvolvida – e também através de um comando de voz permitiu ao condutor ligar para um número de emergência para pedido de socorro. No entanto, como qualquer

tecnologia pioneira, a aplicação tem os seus desafios, como a compatibilidade de hardware e o potencial consumo excessivo de bateria, o que abre caminho para a continuação na investigação e desenvolvimento futuros de modelos mais eficientes e novas tarefas que adicionem valor acrescentado à aplicação e ao utilizador.

Em suma, como observado na secção anterior, todas as funcionalidades foram implementadas com sucesso sendo que há ainda margem para a implementação de soluções adicionais que consigam levar esta solução para o nível seguinte. Na próxima secção iremos abordar algumas destas funcionalidades já detetadas para desenvolvimento.

5.2 Trabalho futuro

A aplicação construída serve como ponto de partida para uma aplicação ainda mais robusta. Há um grande espaço de melhoria dos recursos e modelos atuais, mas também para a implementação de novos modelos e novas tarefas. O *backlog* atual inclui novas funcionalidades, tais como o desenvolvimento de um modelo de deteção de matrículas e faces, com o objetivo de desfocar as mesmas antes de armazenar os vídeos no smartphone para aumentar a privacidade dos dados e cumprir com a lei, que em muitos países [63] não permite a gravação de vídeos em locais públicos que captem imagens de pessoas e matrículas sem o seu expresso consentimento.

Está também em desenvolvimento um painel analítico onde o utilizador poderá consultar um histórico das suas viagens e observar uma análise do seu nível de condução - tanto a nível de agressividade como ecológico - e assim pontuar o condutor por forma a incentivar uma condução mais segura e preventiva.

Um modelo de reconhecimento facial está também em desenvolvimento, que requer uma técnica mais avançada de treino on-device, permitindo que no caso de haver um veículo com diversos condutores habituais, a aplicação consiga reconhecer pela face qual o condutor que está a conduzir.

Para o modelo de deteção de objetos rodoviários, embora o mesmo tenha obtido um desempenho admirável, ainda assim apresenta algumas limitações. Por exemplo, os objetos pequenos ou parcialmente obstruídos podem nem sempre ser detetados, bem como o facto de, por exemplo, os sinais de limite de velocidade serem diferentes de país para país e, portanto, o dataset de treino requer um conjunto de imagens mais abrangentes com a finalidade de reconhecer as diferentes formas e modelos de sinais de diferentes países.

Para o Assistente de Bordo, a principal e única limitação desta abordagem é a utilização de comandos genéricos, que no caso de utilizadores cuja língua seja o inglês, irão ter uma má

experiência já que a aplicação irá detetar as palavras-chave diversas vezes. Como trabalho futuro, irá ser pesquisado um dataset público com palavras-chave mais robustas ou então será criado um dataset customizado com a necessidade de recolher centenas/milhares de amostra de cada palavra-chave.

Como referido anteriormente, algumas funcionalidades a ser implementadas não são necessariamente tarefas, mas são funcionalidades de extrema importância pois serão desenvolvidas para melhorar ainda mais a experiência do utilizador. Algumas destas funcionalidades incluem por exemplo o ajuste automático dos modelos com base no nível da bateria e *drain rate*.

Chegando ao fim deste projeto, concluímos que a solução apresentada representa um avanço significativo na integração de modelos de Machine Learning em dispositivos mobile para segurança rodoviária. Embora a versão atual tenha presente ainda algumas limitações, o caminho a seguir está definido e o trabalho futuro engloba os próximos passos do projeto. Ao inovar e adaptar-se continuamente às necessidades dos utilizadores, esta tecnologia tem potencial para estabelecer novos padrões de qualidade de aplicações Dash Camera e segurança rodoviária.

Acreditamos que a solução proposta vai de encontro à visão do projeto de tornar a experiência de condução mais segura e reduzir assim o número de acidentes e fatalidades na estrada.

6 REFERÊNCIAS

[1] "Países do mundo com o maior número de acidentes de carro e caminhão." Accessed: Sep. 25, 2023. [Online]. Available: https://earthnworld.com/pt/n%C3%BAmero-mundial-de-acidentes-de-carro-e-caminh%C3%A3o/

[2] "Teen Drivers and Passengers: Get the Facts | Transportation Safety | Injury Center | CDC." Accessed: Sep. 25, 2023. [Online]. Available: https://www.cdc.gov/transportationsafety/teen_drivers/teendrivers_factsheet.html

[3] A. McCartt, D. Mayhew, K. Braitman, S. Ferguson, and H. Simpson, "Effects of Age and Experience on Young Driver Crashes: Review of Recent Literature," *Traffic Inj. Prev.*, vol. 10, pp. 209–19, Jul. 2009, doi: 10.1080/15389580802677807.

[4] P. Jones, "Smart Car Demographic: 29 Buyer Facts (Numbers & Stats) | Motor & Wheels." Accessed: Sep. 25, 2023. [Online]. Available: https://motorandwheels.com/smart-car-demographic-29-buyer-facts-numbers-stats/

[5] O. Oviedo-Trespalacios, M. King, A. Vaezipour, and V. Truelove, "Can our phones keep us safe? A content analysis of smartphone applications to prevent mobile phone distracted driving," *Transp. Res. Part F Traffic Psychol. Behav.*, vol. 60, pp. 657–668, 2019.

[6] G. Sikander and S. Anwar, "Driver Fatigue Detection Systems: A Review," *IEEE Trans. Intell. Transp. Syst.*, vol. 20, no. 6, 2019.

[7] L. Taccari *et al.*, "Classification of Crash and Near-Crash Events from Dashcam Videos and Telematics".

[8] E. Giovannini *et al.*, "Importance of dashboard camera (Dash Cam) analysis in fatal vehicle–pedestrian crash reconstruction," *Forensic Sci. Med. Pathol.*, vol. 17, no. 3, pp. 379–387, Sep. 2021, doi: 10.1007/s12024-021-00382-0.

[9] A. Vaezipour, A. Rakotonirainy, N. Haworth, and P. Delhomme, "Enhancing eco-safe driving behaviour through the use of in-vehicle human-machine interface: A qualitative study," *Transp. Res. Part Policy Pract.*, vol. 100, pp. 247–263, 2017.

[10] A. Vaezipour, A. Rakotonirainy, and N. Haworth, "Design of a Gamified Interface to Improve Fuel Efficiency and Safe Driving," in *Design, User Experience, and Usability: Novel User Experiences*, vol. 9747, A. Marcus, Ed., in Lecture Notes in Computer Science, vol. 9747. , Cham: Springer International Publishing, 2016, pp. 322–332. doi: 10.1007/978-3-319-40355-7_31.

[11] "How Can We Prevent Teen Driver Deaths? | Psychology Today." Accessed: Sep. 25, 2023. [Online]. Available: https://www.psychologytoday.com/intl/blog/the-new-normal/202111/how-can-we-prevent-teen-driver-deaths

[12] aaafoundation, "Rates of Motor Vehicle Crashes, Injuries and Deaths in Relation to Driver Age, United States, 2014-2015," AAA Foundation for Traffic Safety. Accessed: Sep. 25, 2023. [Online]. Available: https://aaafoundation.org/rates-motor-vehicle-crashes-injuries-deaths-relation-driver-age-united-states-2014-2015/

[13] T. Dingus *et al.*, "Driver crash risk factors and prevalence evaluation using naturalistic driving data," *Proc. Natl. Acad. Sci.*, vol. 113, p. 201513271, Feb. 2016, doi: 10.1073/pnas.1513271113.

[14] "Distracted Driving Dangers and Statistics | NHTSA." Accessed: Sep. 25, 2023. [Online]. Available: https://www.nhtsa.gov/risky-driving/distracted-driving

[15] S. Ezzini, I. Berrada, and M. Ghogho, "Who is behind the wheel? Driver identification and fingerprinting," *J. Big Data*, vol. 5, Feb. 2018, doi: 10.1186/s40537-018-0118-7.

[16] "Study measures how fast humans react to road hazards," MIT News | Massachusetts Institute of Technology. Accessed: Sep. 25, 2023. [Online]. Available: https://news.mit.edu/2019/how-fast-humans-react-car-hazards-0807
[17] A. Barbado and Ó. Corcho, "Interpretable machine learning models for predicting and explaining vehicle fuel consumption anomalies," *Eng. Appl. Artif. Intell.*, vol. 115, p. 105222, Oct. 2022, doi: 10.1016/j.engappai.2022.105222.
[18] T. Driessen, A. Picco, D. Dodou, D. Waard, and J. de Winter, *Driving Examiners' Views on Data-Driven Assessment of Test Candidates: An Interview Study*. 2021.
[19] V. Neale, S. Klauer, T. Dingus, J. Sudweeks, and M. Goodman, "An Overview of the 100-Car Naturalistic Study and Findings".
[20] A. O. Ferdinand and N. Menachemi, "Associations Between Driving Performance and Engaging in Secondary Tasks: A Systematic Review," *Am. J. Public Health*, vol. 104, no. 3, pp. e39–e48, Mar. 2014, doi: 10.2105/AJPH.2013.301750.
[21] D. He, Z. Wang, E. B. Khalil, B. Donmez, G. Qiao, and S. Kumar, "Classification of Driver Cognitive Load: Exploring the Benefits of Fusing Eye-Tracking and Physiological Measures," *Transp. Res. Rec.*, vol. 2676, no. 10, pp. 670–681, Oct. 2022, doi: 10.1177/03611981221090937.
[22] Y. Bian, C. Yang, J. L. Zhao, and L. Liang, "Good drivers pay less: A study of usage-based vehicle insurance models," *Transp. Res. Part Policy Pract.*, vol. 107, pp. 20–34, Jan. 2018, doi: 10.1016/j.tra.2017.10.018.
[23] A. Gazdag, S. Lestyán, M. Remeli, G. Ács, T. Holczer, and G. Biczók, "Privacy pitfalls of releasing in-vehicle network data," *Veh. Commun.*, vol. 39, p. 100565, Feb. 2023, doi: 10.1016/j.vehcom.2022.100565.
[24] B. Wang, S. Panigrahi, M. Narsude, and A. Mohanty, "Driver Identification Using Vehicle Telematics Data," Mar. 2017. doi: 10.4271/2017-01-1372.
[25] S. Campbell *et al.*, "Sensor Technology in Autonomous Vehicles : A review," in *2018 29th Irish Signals and Systems Conference (ISSC)*, Jun. 2018, pp. 1–4. doi: 10.1109/ISSC.2018.8585340.
[26] O. K. Tonguz, H.-M. Tsai, C. Saraydar, T. Talty, and A. Macdonald, "Intra-Car Wireless Sensor Networks Using RFID: Opportunities and Challenges," Jun. 2007, pp. 43–48. doi: 10.1109/MOVE.2007.4300802.
[27] R. Heydon, *Bluetooth Low Energy: The Developer's Handbook*. Prentice Hall, 2012.
[28] A. Lavric and V. Popa, "A LoRaWAN: Long range wide area networks study," in *2017 International Conference on Electromechanical and Power Systems (SIELMEN)*, Oct. 2017, pp. 417–420. doi: 10.1109/SIELMEN.2017.8123360.
[29] S. Kontogiannis, C. Asiminidis, and G. Kokkonis, "Comparing Relational and NoSQL Databases for carrying IoT data," Jan. 2019.
[30] A. Srinivasan, "IoT Cloud Based Real Time Automobile Monitoring System," Sep. 2018, pp. 231–235. doi: 10.1109/ICITE.2018.8492706.
[31] S. Amghar, S. Cherdal, and S. Mouline, "Which NoSQL database for IoT Applications?," in *2018 International Conference on Selected Topics in Mobile and Wireless Networking (MoWNeT)*, Jun. 2018, pp. 131–137. doi: 10.1109/MoWNet.2018.8428922.
[32] P. Domingos, "A few useful things to know about machine learning," *Commun. ACM*, vol. 55, no. 10, pp. 78–87, Oct. 2012, doi: 10.1145/2347736.2347755.
[33] M. Ergen, "What is Artificial Intelligence? Technical Considerations and Future Perception".
[34] F. Provost and T. Fawcett, "Data Science and Its Relationship to Big Data and Data-Driven Decision Making," *Big Data*, vol. 1, Mar. 2013, doi: 10.1089/big.2013.1508.
[35] E. Guresen and G. Kayakutlu, "Definition of artificial neural networks with comparison

to other networks," *Procedia Comput. Sci.*, vol. 3, pp. 426–433, Jan. 2011, doi: 10.1016/j.procs.2010.12.071.
[36] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, Art. no. 7553, May 2015, doi: 10.1038/nature14539.
[37] R. Choudhary and H. Gianey, "Comprehensive Review On Supervised Machine Learning Algorithms," Dec. 2017, pp. 37–43. doi: 10.1109/MLDS.2017.11.
[38] J. E. van Engelen and H. H. Hoos, "A survey on semi-supervised learning," *Mach. Learn.*, vol. 109, no. 2, pp. 373–440, Feb. 2020, doi: 10.1007/s10994-019-05855-6.
[39] M. Khanum, "A Survey on Unsupervised Machine Learning Algorithms for Automation, Classification and Maintenance," *Int. J. Comput. Appl.*, vol. 119.
[40] A. Patcha and J.-M. Park, "An overview of anomaly detection techniques: Existing solutions and latest technological trends," *Comput. Netw.*, vol. 51, no. 12, pp. 3448–3470, Aug. 2007, doi: 10.1016/j.comnet.2007.02.001.
[41] P. Yadav, A. Mishra, J. Lee, and S. Kim, "A Survey on Deep Reinforcement Learning-based Approaches for Adaptation and Generalization." arXiv, Feb. 16, 2022. doi: 10.48550/arXiv.2202.08444.
[42] K. Weiss, T. M. Khoshgoftaar, and D. Wang, "A survey of transfer learning," *J. Big Data*, vol. 3, no. 1, p. 9, May 2016, doi: 10.1186/s40537-016-0043-6.
[43] X. He, K. Zhao, and X. Chu, "AutoML: A survey of the state-of-the-art," *Knowl.-Based Syst.*, vol. 212, p. 106622, Jan. 2021, doi: 10.1016/j.knosys.2020.106622.
[44] T. Elsken, J. H. Metzen, and F. Hutter, "Neural Architecture Search: A Survey".
[45] S. Han, H. Mao, and W. J. Dally, "Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding." arXiv, Feb. 15, 2016. doi: 10.48550/arXiv.1510.00149.
[46] R. Jabbar, K. Al-Khalifa, M. Kharbeche, W. Alhajyaseen, M. Jafari, and S. Jiang, "Real-time Driver Drowsiness Detection for Android Application Using Deep Neural Networks Techniques," *Procedia Comput. Sci.*, vol. 130, pp. 400–407, Jan. 2018, doi: 10.1016/j.procs.2018.04.060.
[47] M. Tashakori, A. Nahvi, and S. Ebrahimian Hadi Kiashari, "Driver drowsiness detection using facial thermal imaging in a driving simulator," *Proc. Inst. Mech. Eng. [H]*, vol. 236, no. 1, pp. 43–55, Jan. 2022, doi: 10.1177/09544119211044232.
[48] M. Arif, D. Khattak, K. Fiaz, and A. Niaz, "A Real-Time Driver Drowsiness Detection and Warning System Based on an Eye Blinking Rate," 2020, pp. 106–117. doi: 10.1007/978-981-15-5232-8_10.
[49] J. Dai, J. Teng, X. Bai, Z. Shen, and D. Xuan, "Mobile phone based drunk driving detection," in *2010 4th International Conference on Pervasive Computing Technologies for Healthcare*, Mar. 2010, pp. 1–8. doi: 10.4108/ICST.PERVASIVEHEALTH2010.8901.
[50] T. Horberry, L. Hartley, G. P. Krueger, and N. Mabbott, "Fatigue detection technologies for drivers: a review of existing operator-centred systems," pp. 321–326, Jan. 2001, doi: 10.1049/cp:20010483.
[51] P. Singh, N. Juneja, and S. Kapoor, "Using mobile phone sensors to detect driving behavior," in *Proceedings of the 3rd ACM Symposium on Computing for Development*, in ACM DEV '13. New York, NY, USA: Association for Computing Machinery, Jan. 2013, pp. 1–2. doi: 10.1145/2442882.2442941.
[52] R. Chhabra, S. Verma, and C. Rama Krishna, "Detecting Aggressive Driving Behavior using Mobile Smartphone," in *Proceedings of 2nd International Conference on Communication, Computing and Networking*, C. R. Krishna, M. Dutta, and R. Kumar, Eds., in Lecture Notes in Networks and Systems. Singapore: Springer, 2019, pp. 513–521. doi: 10.1007/978-981-13-1217-5_49.

[53] "The Best Dash Cams vs. Traditional Car Security." Accessed: Sep. 25, 2023. [Online]. Available: https://visionsecurity.jp/en/dashcams.html
[54] "Google Play app ratings 2023," Statista. Accessed: Sep. 25, 2023. [Online]. Available: https://www.statista.com/statistics/266217/customer-ratings-of-android-applications/
[55] K. O. Chin and L. Connie, "Smart Apps - Automated Number Plate Recognition System," Sep. 2021, Accessed: Sep. 25, 2023. [Online]. Available: https://oer.ums.edu.my:443/handle/oer_source_files/1599
[56] C. Wanpeng and B. Wei, "Adaptive and dynamic mobile phone data encryption method".
[57] T. P. Nguyen, M. T. Chew, and S. Demidenko, "Eye tracking system to detect driver drowsiness," in *2015 6th International Conference on Automation, Robotics and Applications (ICARA)*, Feb. 2015, pp. 472–477. doi: 10.1109/ICARA.2015.7081194.
[58] P. Warden, "Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition." arXiv, Apr. 09, 2018. doi: 10.48550/arXiv.1804.03209.
[59] Y. A. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller, "Efficient BackProp," in *Neural Networks: Tricks of the Trade: Second Edition*, G. Montavon, G. B. Orr, and K.-R. Müller, Eds., in Lecture Notes in Computer Science. , Berlin, Heidelberg: Springer, 2012, pp. 9–48. doi: 10.1007/978-3-642-35289-8_3.
[60] M. Tan, R. Pang, and Q. V. Le, "EfficientDet: Scalable and Efficient Object Detection." arXiv, Jul. 27, 2020. Accessed: Sep. 25, 2023. [Online]. Available: http://arxiv.org/abs/1911.09070
[61] A. Jain, S. Bhattacharya, M. Masuda, V. Sharma, and Y. Wang, "Efficient Execution of Quantized Deep Learning Models: A Compiler Approach." arXiv, Jun. 17, 2020. doi: 10.48550/arXiv.2006.10226.
[62] "Fitness App designs - Android Studio – Material UiUx." Accessed: Sep. 25, 2023. [Online]. Available: https://2db60537.isolation.zscaler.com/profile/d0b69f16-221e-400f-a5b6-f5b23f89ce17/zia-session/?controls_id=dc699105-e719-49c7-a9de-e4de0088d5c3&region=par&tenant=9f50d95a09b7&user=111135585a9da4738d73ecf9a49d59f684ebf6c6b2853ba5a6df6321a22e635f&original_url=https%3A%2F%2Fwww.materialuiux.com%2Fproduct%2Ffitness-app-designs-android-studio%2F&key=sh-1&hmac=0e8e67ee7e1d318a91c9c674433afb3ec505b79d88d223c749f0ba1b75efdf44
[63] SAPO, "Filmar pessoas na rua com uma 'dash cam' é ilegal? CNPD diz que sim, mas especialistas dividem-se," Polígrafo. Accessed: Sep. 25, 2023. [Online]. Available: https://poligrafo.sapo.pt/sociedade/artigos/filmar-pessoas-na-rua-com-uma-dash-cam-e-ilegal-cnpd-diz-que-sim-mas-especialistas-dividem-se