Sistema de Reconhecimento de Ruídos Urbanos
Mestrado em Desenvolvimento de Software e Sistemas Interativos

Autor:

Ricardo António Fernandes Leitão

Orientador:

Ana Paula Neves Ferreira da Silva

Coorientador:

Arlindo Ferreira da Silva

Trabalho de Projeto apresentado à Escola Superior de Tecnologia do Instituto Politécnico de Castelo Branco para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Desenvolvimento de Software e Sistemas Interativos, realizado sob a orientação científica do Professor Adjunto Doutora Ana Paula Neves Ferreira da Silva e coorientação do Professor Adjunto Doutor Arlindo Ferreira da Silva, do Instituto Politécnico de Castelo Branco.

novembro de 2021

Composição do júri

Presidente do júri

Alexandre José Duro da Fonte, Professor Adjunto da UTC de Informática da Escola Superior de Tecnologia do Instituto Politécnico de Castelo Branco.

Vogais

Anabela Borges Simões, Professor Adjunto do Departamento de Engenharia Informática e de Sistemas do Instituto Superior de Engenharia de Coimbra do Instituto Politécnico de Coimbra.

Fernando Reinaldo da Silva Garcia Ribeiro, Professor Adjunto da UTC de Informática da Escola Superior de Tecnologia do Instituto Politécnico de Castelo Branco.

Ana Paula Neves Ferreira da Silva, Professor Adjunto da UTC de Informática da Escola Superior de Tecnologia do Instituto Politécnico de Castelo Branco. (Orientador).

Dedicatória

Dedicado a todos os amigos, família e professores, que de alguma forma me ajudaram a concluir este projeto.

Em especial:

Dedicado à minha mãe e aos meus avós, que me encorajaram e incentivaram a ingressar e terminar este percurso.

Agradecimentos

Um agradecimento aos meus orientadores, Professora Doutora Ana Paula Neves Ferreira da Silva, e Professor Doutor Arlindo Ferreira da Silva, pela orientação e disponibilidade fornecidas, e apoio prestado ao longo de todo o processo.

Um agradecimento ao meu amigo Tiago Sobreira, que me ajudou a concluir a licenciatura como colega de projeto e trabalhos, e a ajudar a tornar possível a minha candidatura ao mestrado.

Um agradecimento ao meu amigo Nuno Batuca, que me ajudou a concluir as restantes unidades curriculares do mestrado, como colega de trabalhos, e que me ajudou a chegar a este ponto.

Um agradecimento ao meu amigo Aurélio Silva, pelas conversas, momentos de partilha e pela caminhada conjunta a nível profissional e académico que tivemos ao longo destes anos.

Sumário

Este trabalho consiste no desenvolvimento de um sistema de reconhecimento de ruídos urbano, composto por duas aplicações, uma *Android* e outra *Web*.

Através da aplicação *Android*, o sistema tem a capacidade de classificar amostras de áudio de sons urbanos, e registar essas classificações numa base de dados remota de incidentes. Estas funções são suportadas por um modelo de inteligência artificial, mais concretamente uma rede neuronal convolucional, integrada na aplicação.

Na aplicação *Web* é possível uma posterior análise aos dados recolhidos pela primeira aplicação, no formato de gráficos e mapeamento, em função dos incidentes registados na base de dados remota.

O sistema desenvolvido destina-se a um possível complemento dos sistemas de segurança já existentes.

Palavras-Chave

Rede Neuronal Convolucional, Classificação de Som, *Machine Learning*, *Deep Learning*, Áudio-Vigilância.

Abstract

This work consists in the development of an urban noise recognition system, consisting of two applications, one Android and the other Web.

Through the Android application, the system can classify audio samples of urban sounds and record those classifications in a remote incident database. These functions are supported by an artificial intelligence model, more specifically a convolutional neural network, integrated in the application.

It is possible with the web application further analyze the data collected by the first application, in the format of graphics and mapping, according to the incidents registered in the remote database.

The developed system is intended as a possible complement to the already developed system of security systems.

Keywords

Convolutional Neural Networks, Sound Classification, Machine Learning, Deep Learning, Audio-Surveillance.

Índice geral

Índice de figuras

Índice de tabelas

Lista de abreviaturas, siglas e acrónimos

Siglas e Acrónimos:

DL – *Deep Learning*;

IOT – *Internet of Things*;

ML – *Machine Learning*;

MFCC - *Mel-frequency cepstral coefficients*;

GMM - *Gaussian Mixture Model*;

DNN - *Deep Neural Network*;

RNN - *Recurrent Neural Network*;

CNN - *Convolutional Deep Neural Network*;

SED - *Sound Event Detection.*

Abreviaturas:

Web App – Aplicação *Web*.

1. Introdução

Neste capítulo, começar-se-á por apresentar o enquadramento e as motivações para o desenvolvimento deste trabalho, seguido de um contexto sobre os sistemas de segurança, e a possível proposta de integração do sistema desenvolvido nos mesmos, bem como os objetivos fundamentais do projeto e ainda o planeamento e organização do trabalho realizado.

1.1. Enquadramento e Motivação

Uma das grandes motivações para este trabalho é desenvolver um sistema de reconhecimento e análise de ruídos urbanos, usando técnicas da área de *Machine Learning* (ML). Esta área tem muito potencial para oferecer no que toca a conceitos como "cidades inteligentes" e IOT (*Internet of Things)* (Trust, 28 May 2019).

Neste trabalho, utiliza-se uma subárea de ML designada de *Deep Learning*. O sistema tem como propósito final trazer uma nova solução no ramo da segurança urbana, mais concretamente na áudio-vigilância e monitorização do meio urbano. Como uma primeira abordagem, o trabalho irá focar-se no desenvolvimento de uma aplicação móvel direcionada para o cidadão comum, para demonstração do conceito de áudio-vigilância suportada por *Machine Learning*. No entanto, é importante salientar as possibilidades no campo da segurança que este projeto pode ter em toda a sua extensão. Existem obviamente sistemas de segurança avançados suportados por vídeo-vigilância, monitorização de movimento e emissões termais entre outros (Okamura, et al., 2018). Porém, uma componente que é frequentemente menosprezada pela sua simplicidade, comparativamente às demais, é precisamente a da componente áudio. Enquanto a vigilância visual normalmente necessita de análises mais complexas ou até mesmo de suporte humano para interpretar as imagens captadas, num processo demorado e entediante, a áudio-vigilância que se pretende para este projeto pretende realizar um trabalho automatizado, interpretado e processado exclusivamente pelo sistema, que poderá, em trabalho futuro, ser integrado, na forma de sistemas embutidos, em sistemas de vigilância já existentes. Também é importante mencionar que se trabalha sob um formato de dados (áudio .mp3 ou .WAV por exemplo), menos dispendioso computacionalmente do que o formato de dados usado para vídeo (.mp4). O volume de dados produzido também será significativamente menor comparativamente à produção de vídeo, pois a informação áudio original não será armazenada pelo sistema. Este trabalhará sob o áudio obtido em tempo real para produzir informação que o caracteriza a qual designamos por incidentes. O processamento e análise posteriores serão realizados a partir dos incidentes e não do áudio em concreto.

Na Tabela 1, apresentam-se as vantagens e desvantagens do sistema que se pretende desenvolver neste projeto.

Tabela 1 - Vantagens e Desvantagens do Projeto

Tipo	Vantagens	Desvantagens
Componente Áudio	Abrangência de 360º para captar o *input*	Limitado à sensibilidade e alcance do *hardware* usado
	Volume de dados "leve"	Sujeitável a interferências por ruído
	Automatizado e continuo na recolha de dados	Pode vir a fazer falta para análises mais profundas
Sistema	Processamento em tempo real dos dados obtidos	Necessária ligação permanente à Internet
	A parte do sistema que gera os dados está localizada nos dispositivos dos utilizadores comuns	Privacidade do utilizador posta em causa
	Possível integração em sistemas de vigilância já existentes.	Necessária a aquisição de novo *hardware* dedicado à tarefa e adaptação do *software* a esse sistema.
	Produção de dados pertinentes e análises dos mesmos de forma automática.	Necessária inclusão de sistemas GPS nos dispositivos recolhedores dos dados a interpretar.
	O o*utput* obtido é automaticamente categorizado, distinguido e armazenado num servidor remoto.	X
	As análises produzidas serão apresentadas usando formatos de dados facilmente interpretáveis pelo utilizador comum (tabelas, gráficos, entre outros…).	X
	Não necessita de suporte humano para um funcionamento adequado (apenas a ação de ativar/desativar o sistema é necessária).	X

1.2. Sistemas de Segurança de Áudio

Um sistema de segurança é algo vantajoso quando se trata de aumentar o fator qualidade-de-vida de um indivíduo ou de uma organização. O mesmo tem o importante objetivo de preservar a segurança da entidade e das suas posses, prevenindo e dissuadindo a ocorrência de atividades ilegítimas e criminosas. A componente de monitorização de um sistema de segurança é uma parte fulcral do mesmo. A gestão de eventos, desde a deteção, registo e reporte de um incidente, até à análise e atuação do mesmo, é um processo que tem de ser minucioso, rápido e preferencialmente automatizado, de modo a tornar todo o processo o mais ágil possível (Marcondes, 2016). Existem vários modos de implementar um sistema de segurança, e aspetos do mesmo que devem ser ponderados, como por exemplo definir como será recolhido o *input* a ser interpretado pelo sistema. A vídeo-vigilância, deteção de movimento, leitura termal, **captação de áudio**, são algumas possíveis opções. Ainda existe o fim para o qual o sistema de segurança deve ser implementado, o que influenciará bastante a estrutura do sistema. O âmbito residencial, empresarial/comercial, **segurança pública** ou **urbana**, alarmes de incêndio, sistemas de defesa pessoal são alguns exemplos de tipos de implementação de sistemas de segurança (Castra, 2012).

O ambiente em que foi enquadrado este projeto consiste na componente de deteção de *inputs* de áudio para a proteção, segurança e monitorização do meio urbano (*Audio Surveillance*). A componente de áudio tende a ser pouco valorizada na implementação de um sistema de segurança, sendo estes mais focados nas restantes componentes de deteção de *inputs* mencionadas anteriormente. Na Tabela 2, são expostas algumas vantagens e desvantagens da áudio-vigilância e da vídeo-vigilância (Flammini, et al., 2013) e (Li, 2019).

Tabela 2 - Áudio-Vigilância VS Vídeo-Vigilância

Áudio-Vigilância	Vídeo-Vigilância
Menor complexidade no desenvolvimento e implementação.	Menos sujeitável a interferências no input recolhido.
Independente das condições de iluminação ou obstrução da imagem.	Independente ao ruído do local.
Captação de dados 360 graus e independente da complexidade do equipamento.	Maior alcance linear na captação de dados.
Volume de dados produzido não é elevado.	Permite mais formas de realizar análises mais variadas.
Pode ser mais facilmente automatizado.	Necessita de mais assistência humana.
Pode complementar os outros tipos de sistemas e é mais económico.	Presente em muitos sistemas de segurança e muito utilizado em todos os âmbitos.

A comparação entre o sistema escolhido para implementação (áudio-vigilância), e o sistema mais frequentemente utilizado (vídeo-vigilância) é pertinente. No decorrer das pesquisas efetuadas no âmbito deste projeto, pôde-se validar a criticidade e importância dos sistemas de segurança, e o quão os mesmos são complexos, dispendiosos, e requerem a colaboração de muita mão-de-obra operacional e técnica para o correto funcionamento dos mesmos. Existem muitos fatores a ter em conta, desde a garantia do suporte executivo, definição de objetivos, âmbito do sistema, análise de componentes e de risco, definição da estrutura, certificações, monitorização e manutenção dos mesmos, custos e profissionais envolvidos, entre outros (ins2outs, 2019). O que se pretende com este trabalho não é de todo substituir ou concorrer com os sistemas de segurança atuais ou alterar a forma como os mesmos são concebidos e geridos. O pretendido é **colaborar** e **aliar** esforços com os mesmos, criando mais uma opção de monitorizar o meio urbano, preferencialmente de uma forma mais **económica**, **ágil** e livre de grande intervenção humana, que são os dois maiores problemas dos sistemas atuais e que se pretende combater com a solução desenvolvida neste projeto (Bauer, 2018).

1.3. Componente de Vigilância/Monitorização Áudio

Os dados obtidos pela via de áudio são analisados à base da frequência e amplitude da onda analógica/digital que compõe os mesmos, suscetíveis a ruído e interferência. O sistema desenvolvido realiza a leitura deste tipo de dados, e esse processo será documentado posteriormente neste documento. Como já foi dito, este tipo de abordagem, aliada aos sistemas de vigilância clássicos, poderá ser uma boa adição à vigilância e monitorização do meio urbano, pois permite um tipo diferente de recolha de dados, que complementa os mesmos e poderá permitir análises mais variadas e claras sem grandes custos acrescidos.

Através da recolha inteligente de dados, poderá ser realizada uma análise de forma independente à intervenção humana, com base numa interceção contínua de dados, que consistem na leitura de um fluxo de som constante que está a ser interpretado pelo sistema ininterruptamente. Esta quantidade de dados processada continuamente pode não implicar necessariamente o seu armazenamento, pois os únicos dados relevantes que o sistema de análise por áudio irá produzir e armazenar remotamente são de facto os incidentes criados através das ocorrências detetadas pelo áudio, que vai sendo "lido" pelo modelo. Isto possibilita uma forma eficaz de analisar o meio envolvente sem que seja necessário produzir um grande volume de dados, sendo uma solução que alia a sua automatização ao seu "leve" e cíclico funcionamento.

A implementação desta solução, no âmbito deste projeto, passa por uma aplicação *mobile* extremamente simples, que é o ponto de captação dos incidentes de áudio no sistema. A cada ocorrência detetada pelo equipamento(s) que está a correr na aplicação, será gerado e processado um incidente representativo da ocorrência detetada naquele mesmo instante. Com vários pontos de origem de incidentes (os diversos equipamentos dos utilizadores) e com um banco de dados de incidentes situado num servidor remoto, será posteriormente possível remodelar a informação obtida de formas que poderão ajudar não tanto a uma solução imediata de uma ocorrência específica, mas a análises mais completas de conceção de padrões e de *Data Mining*, relacionados com as áreas das cidades inteligentes e cidades seguras. Poder-se-á considerar esta ideia como uma solução a médio-longo prazo, que tem como grande objetivo combater os problemas apresentados anteriormente, nomeadamente a complexidade, custos, e trabalho envolvidos na manutenção de um sistema de segurança.

1.4. Objetivos

Este projeto consiste no desenvolvimento de um sistema, cuja arquitetura se apresenta na Figura 1, composto por duas aplicações para monitorização de ruídos urbanos: uma aplicação para dispositivos móveis capaz de captar, reconhecer e registar ruídos urbanos e uma aplicação *web* que terá capacidade de apresentar e analisar os registos (designados futuramente por incidentes) armazenados pela aplicação móvel. O reconhecimento dos ruídos é conseguido à custa da utilização de técnicas de *Machine Learning*, mais concretamente à custa da utilização de uma rede neuronal que, depois de treinada, é integrada na aplicação móvel de forma a conseguir-se a classificação das amostras de áudio recolhidas.

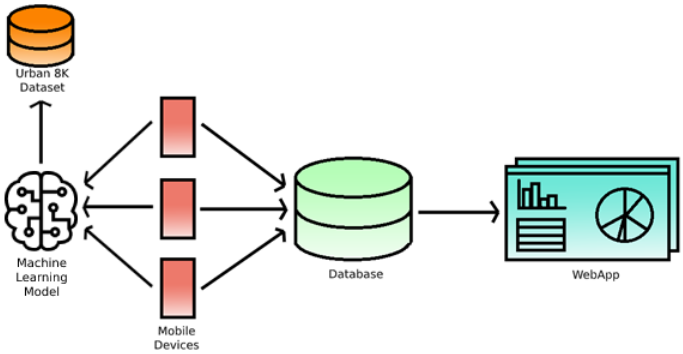


Figura 1 - Representação do Sistema de Informação

Assim a aplicação móvel, desenvolvida para o sistema operativo *Android*, é responsável pela recolha de dados, que irá compor a informação que será analisada pela aplicação *web*. A aplicação realiza a recolha de dados em formato de áudio do ambiente envolvente, e procede à classificação do *input* recolhido. Após a análise, a aplicação processa os dados gerados em forma de um **incidente**, para uma base de dados remota, que será utilizada pela aplicação ***web***, para elaboração de gráficos e análise dos dados. De salientar que o modelo usado pela aplicação *Android* para fazer a classificação dos ruídos foi treinado com os dados presentes numa base de dados de ficheiros áudio *open-source*, para fins académicos, de seu nome *Urban Sound 8K Dataset* (Urban8K, s.d.). Esta base de dados contém milhares de exemplos de ruídos típicos de um ambiente urbano, os quais estão categorizados nas seguintes classes, que são, também, aquelas de que a aplicação fará distinção e categorização:

- Ar Condicionado;
- Buzina de Carro;
- Crianças a Brincar;
- Cão a Ladrar;
- Berbequim;
- Motor de Carro;
- Tiro;
- Martelo Pneumático;
- Sirene;
- Música de Rua.

De forma resumida, podem destacar-se os seguintes objetivos para este trabalho:

1. Conceber um produtor e gestor de informação, a partir de dados recolhidos no meio urbano, com o propósito de monitorizar os mesmos e complementar os sistemas de segurança já existentes;
2. Realizar análises dos dados recolhidos, através de representações gráficas, facilmente interpretáveis pelo utilizador comum, de modo a tirar conclusões sobre uma dada população, local, período de tempo, entre outros parâmetros possíveis.
3. Criação de um modelo, usando técnicas de *Machine Learning*, que consiga reconhecer sons urbanos. Para tal, será necessário escolher um *dataset* (de preferência *open-source)* adequado ao tema e o modelo a utilizar para o obter.
4. Integrar o modelo ML no sistema proposto, mais concretamente na aplicação *Android*, de modo que o mesmo tenha a capacidade de captar o som urbano, e classificar o mesmo utilizando o modelo obtido.
5. Desenvolver uma plataforma *web* que possibilite a visualização e análise dos dados recolhidos pelo sistema.

1.5. Planeamento

Ao longo do desenvolvimento dos trabalhos, foram realizadas as seguintes tarefas:

- Estudo do estado de arte;
- Elaboração do planeamento e modelação da aplicação *Web*;
- Desenvolvimento da aplicação *Web*;
- Planeamento e modelação da aplicação móvel;
- Desenvolvimento da aplicação móvel;
- Estudo, Planeamento, Modelação e Implementação do Modelo de DL;
- Interligação das duas Aplicações.

Na Tabela 3, apresenta-se a sequência temporal de cada uma destas tarefas ao longo do tempo de desenvolvimento dos trabalhos.

Tabela 3 - Diagrama Temporal de Trabalhos

Tarefas	Tempo												
	nov/20	dez/20	jan/21	fev/21	mar/21	abr/21	mai/21	jun/21	jul/21	ago/21	set/21	out/21	nov/21
Estudo do estado de arte;													
Elaboração do planeamento e modelação da aplicação *Web*;													
Desenvolvimento da aplicação *Web*;													
Planeamento e modelação da aplicação móvel;													
Desenvolvimento da aplicação móvel;													
Estudo, Planeamento, Modelação e Implementação do Modelo de DL;													
Interligação das duas Aplicações.													
Revisões													
Entrega													

1.6. Organização do Documento

A estrutura do relatório é composta primeiramente por uma abordagem teórica dos conceitos de *Machine Learning* e *Deep Learning*, que aborda algumas técnicas de aprendizagem possíveis de implementar nestas áreas, ao longo do capítulo 2.

No capítulo 3, é apresentado o trabalho realizado no âmbito do estudo do estado da arte, e nos trabalhos já existentes nas áreas de *Machine Learning* e *Deep Learning*, mais concretamente na análise e classificação de sons do meio urbano, bem como aplicações com funções similares àquelas que se pretendem para o sistema desenvolvido neste projeto.

No capítulo 4, é apresentado o trabalho desenvolvido na área de *Deep Learning*: a escolha, conceção, treino, e extração do modelo desenvolvido para posterior integração no sistema.

Por fim, no capítulo 5, são apresentados os componentes que integram o sistema desenvolvido, sendo eles as duas aplicações e a base de dados remota. É ainda apresentado o modo como foi realizada a integração do modelo na aplicação *Android*, bem como os testes realizados para o modelo já inserido no sistema.

2. *Machine Learning* e *Deep Learning*

Com este capítulo, pretende-se fornecer alguns dos conceitos fundamentais das tecnologias utilizadas para o desenvolvimento deste trabalho, com grande foco na área do *Machine Learning*, mais especificamente na subárea do *Deep Learning*, bem como diferentes técnicas de treino/aprendizagem utilizadas pelos modelos característicos de ML e DL.

2.1 Machine Learning

O conceito de *Machine Learning,* como o próprio nome poderá indicar, consiste na arte de instruir e programar uma máquina a conseguir aprender um dado comportamento através de um reservatório de dados, de forma autónoma. Numa abordagem mais técnica, pode-se dizer que um programa aprende da experiência **E** que diz respeito a uma dada tarefa **T** e uma dada medida de desempenho **D**. Se **T** for desempenhado conforme foi medido em **D**, o programa foi aprendido com base na experiência adquirida em **E** (Géron, 2019).

Por vezes, surgem problemas que não poderão ser resolvidos pelas abordagens de desenvolvimento e programação tradicionais, devido não só à sua complexidade, mas também à constante atualização de novas formas de contornar as medidas tomadas na programação realizada. Considere-se como exemplo um algoritmo presente nos gestores de *email* para detetar o *spam* no fluxo de emails recebidos. Ora, algumas medidas que se poderiam tomar na conceção de um algoritmo que sirva este propósito seria, por exemplo, algo que analisasse o texto dos *emails* e procurasse por palavras-chaves como "*free*", "*click here*", "*credit card*", entre outras. No entanto, os compositores destes *emails* enganosos estão sempre à procura de formas de contornar os algoritmos desenvolvidos para intercetar as suas tentativas de atos maliciosos, que muito são tentadas neste tipo de *emails*. É aqui que entram os algoritmos de ML. Estes possuem algumas vantagens únicas, que os algoritmos mais comuns não têm capacidade de cobrir. Um algoritmo de ML tem a capacidade de, através de um *dataset* (neste caso constituído por e *emails* enganosos, designados por *spam* e *emails* legítimos, designados por *ham*) de realizar a sua própria análise do que seria ou não *email spam*, e realizar a filtragem mais adequada. Após treino do modelo, o mesmo seria capaz de realizar o processo de forma mais automatizada, já para não falar que ainda se poderia sempre manter o mesmo atualizado, pois o *dataset* (experiência **E**), que o mesmo usou para conseguir realizar a filtragem (tarefa **T**) poderá ser atualizado para que a filtragem seja sempre adequadamente realizada (medida de desempenho **D**), e nesse mesmo *dataset* estejam sempre presentes os exemplos mais recentes utilizados pelos *spammers*.

As abordagens suportadas pelos algoritmos de ML permitem realizar uma análise mais aprofundada no campo do *Data Mining*, através da deteção de padrões mais subliminares e difíceis de detetar pelas abordagens convencionais, que têm mais dificuldade em ler e interpretar este tipo de comportamentos em grandes volumes de dados. Quando o **problema** passa por algo que é muito repetitivo, entediante e trabalhoso, envolve um grande volume de dados, e poderá requerer a constante atualização do *dataset* de dados a utilizar, a **solução** poderá passar pela implementação de algoritmos de ML, conforme explicado na Figura 2.

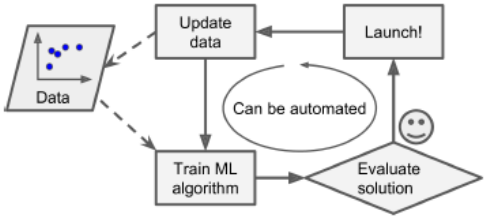


Figura 2 - Algoritmo de ML adaptativo à mudança, imagem retirada de (Géron, 2019)

Existem alguns tipos distintos de algoritmos de ML que podem ser utilizados para o treino do modelo, consoante a situação em causa e a vontade dos programadores. A diferença entre eles passa pela quantidade de ajuda humana que é preciso embutir no *dataset* e no algoritmo em si, que será inversamente proporcional à dificuldade de implementação desse mesmo algoritmo. Quanto mais autónomo este for, mais complexo terá de ser. Assim, geralmente, os algoritmos de ML podem ser classificados de quatro formas, como se descreve a seguir.

2.1.1 Aprendizagem por Reforço

Este é o método mais complexo. Consiste na instrução reforçada do algoritmo a desempenhar uma dada tarefa, através de uma sucessão de condições binárias que podem levar ao sucesso do mesmo e uma consequente recompensa, ou ao insucesso que levará a uma penalização. Nesta abordagem, o algoritmo é treinado forçosamente a descobrir de forma totalmente autónoma o que deve realizar para obter o sucesso pretendido e o que não deve realizar para não ser penalizado. Tem-se um exemplo de um videojogo, em que pretende instruir um programa a jogar autonomamente esse mesmo jogo, de forma a "passar" o mesmo, e não perder no processo. Com base numa base de dados de grande dimensão, de filmagens de pessoas humanas a jogar, o algoritmo compara esses comportamentos e tenta perceber o que deve fazer para obter o sucesso pretendido, visto nos vídeos através de várias tentativas exaustivas, com as quais vai aprendendo iterativamente o comportamento correto. A cada tentativa falhada, o algoritmo praticamente "traumatiza" essa experiência, e tenta a todo o custo não voltar a passar pelo mesmo novamente. Apenas são repetidas as experiências que o mesmo considera essenciais para a realização do objetivo final, que é terminar o jogo.

2.1.2 Aprendizagem não-supervisionada

Aqui o algoritmo aprende através de um conjunto de dados não categorizados, e tenta realizar análises comparando conjuntos de dados semelhantes, fazendo as suas próprias categorizações e agrupamentos consoante as semelhanças encontradas nos exemplos disponíveis. O *Clustering* é um dos algoritmos que se enquadra na aprendizagem não supervisionada (Géron, 2019). Tendo-se como exemplo os visitantes de um dado *website* como, por exemplo, um blog. Usar um algoritmo de *clustering* para agrupar os visitantes por interesses semelhantes ou em comum poderá ser vantajoso para esse *blog*, pois mostrará a um dado grupo resultante das agrupações realizadas, apenas conteúdo relevante ao mesmo e de interesse para os utilizadores que constituem esse mesmo grupo. Existe ainda a possibilidade de subdividir os vários grupos em grupos ainda mais específicos com a tecnologia de *hierarchical clustering*.

2.1.3 Aprendizagem semi-supervisionada

No caso da aprendizagem semi-supervisionada, o *dataset* usado está parcialmente categorizado, numa pequena minoria, e a restante parte dele está presente sem categorização. Neste tipo de aprendizagem, apenas é necessária a categorização parcial dos dados, para que o algoritmo de ML consiga realizar a sua aprendizagem e treinar o seu comportamento. Neste tipo de abordagem, ainda existe comportamento autónomo do algoritmo, no entanto, a assistência humana é necessária para que o algoritmo cumpra o seu propósito final. Tem-se um excelente exemplo nos *hosting services* do *Google Photos*, ou nas galerias de alguns *smartphones*. Num dado conjunto de fotografias, o algoritmo consegue identificar a presença de uma dada pessoa (ou várias) em uma ou várias fotografias, e esta seria a componente não supervisionada do algoritmo. Contudo, o algoritmo necessita que o utilizador identifique as pessoas presentes nas fotografias, ou seja, categorize os grupos formados pelo algoritmo de *clustering*. Esta seria a componente supervisionada do algoritmo. A maioria dos algoritmos semi-supervisionados são combinações de algoritmo não supervisionados e supervisionados.

2.1.4 Aprendizagem supervisionada

A aprendizagem supervisionada, de entre as abordagens estudadas, foi a escolhida para resolver a tarefa que se pretende implementar no âmbito deste projeto. Os algoritmos de aprendizagem supervisionada usam um *dataset* totalmente categorizado e com os dados presentes totalmente agrupados e distinguidos uns dos outros. Pode-se afirmar que tem a solução agrupada individualmente a cada objeto de estudo disponível. No caso do problema apresentado, o *dataset* consistirá numa grande quantidade de ficheiros áudio agrupados consoante os tipos disponíveis. A estratégia deste algoritmo consiste na comparação intensiva de um exemplo novo introduzido no sistema, procurando pontos em comum com os exemplos já presentes no *dataset*.

2.2 Deep Learning

A área de *Deep Learning* é uma subárea de *Machine Learning* (Figura 3). Estuda o processo de aprendizagem de uma máquina através da formação de representações pertinentes dos dados, e da extração de conhecimento dos mesmos (Fridman, 2019). Esta tecnologia dependerá sempre do tipo de dados e âmbito do estudo, bem como da forma como estes são representados e do tipo do modelo de ML.

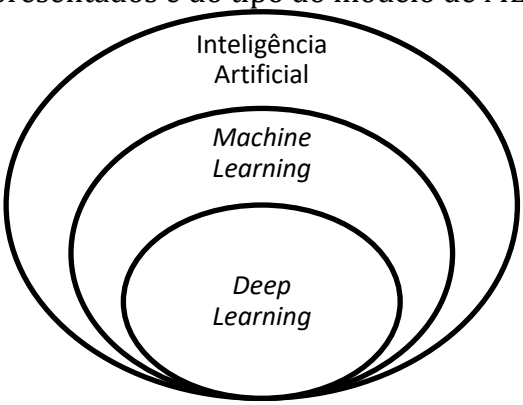


Figura 3 – Enquadramento de Áreas de Inteligência Artificial

Este processamento passa pelo uso de *encoders* e *decoders*, que são responsáveis pela interpretação dos dados e pela forma como estes poderão ser transformados e analisados (Chollet, 2018). A Figura 4 ilustra, de forma bastante sintetizada, a forma como é obtido através do *input* disponível, o conhecimento pretendido ou previsão gerada pelo algoritmo.

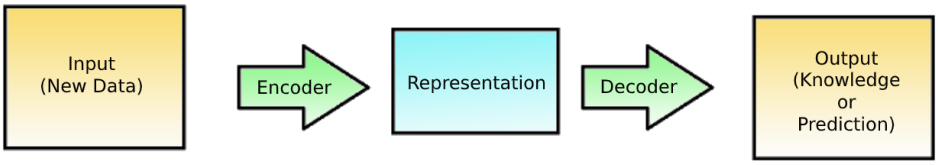


Figura 4 – Processo de codificação dos dados em DL

Pretende-se que a representação dos dados seja o mais significativa possível, e a grande dificuldade dos algoritmos de ML passa por perceber qual a melhor forma de representar os dados de modo a atingir este objetivo. Quanto mais significativa a representação se tornar, mais próximo o resultado ficará do *output* esperado, que é precisamente aquilo que se pretende alcançar com o algoritmo. Uma das diferenças essenciais dos algoritmos de ML para os algoritmos mais comuns consiste na capacidade de encontrar de forma automática, ou semiautomática, a **transformação** que torna os dados em representações tão adequadas quanto possível para uma determinada tarefa. Este processo é consideravelmente complexo e complicado, e não é totalmente automatizado, pois o algoritmo recorre a um "*set*" ou conjunto de operações pré-definidas para definir os comportamentos a realizar. Este *set* é conhecido como *Hyphotesis Hyperspace* (Chollet, 2018).

Os algoritmos de *Deep Learning* formam, como já foi dito anteriormente, uma subdivisão do *Machine Learning*, que dá enfâse à aprendizagem por camadas sucessivos de representações cada vez mais significativas e próximas do resultado pretendido. Quanto mais camadas o algoritmo possuir, mais profundidade e complexidade este terá (*depth*). A vertente moderna do DL baseia-se na existência de centenas de camadas de representações, todas formuladas e aprendidas automaticamente, devido à exposição aos dados para treino disponíveis. Estas representações obtidas com base em cada camada são, sistematicamente, colocadas em prática através de modelos de redes neuronais. Pode-se afirmar que o DL é uma *framework* matemática para a aprendizagem de *data* (Chollet, 2018). Na Figura 5, pode observar-se a representação de uma rede neuronal de 4 camadas, em que o *input* consiste num número manuscrito e o objetivo da rede (o seu *output*) é realizar uma previsão do número a que a versão manuscrita corresponde.

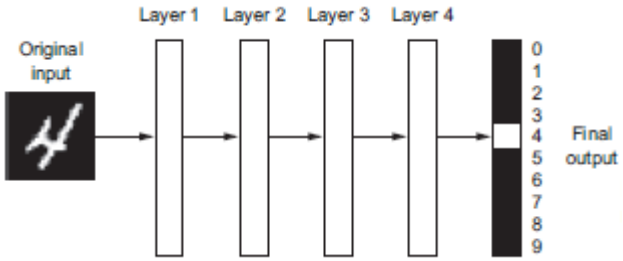


Figura 5 - Classificação Numérica, retirado de (Chollet, 2018)

Em cada camada existente no exemplo apresentado na Figura 6, são formuladas representações que vão sendo cada vez mais diferentes do *output* original, mas, por outro lado, vão sendo cada vez mais informativas acerca do resultado final. Passa-se por um processo de "dilatação da informação por multi-estágio", em que a informação vai sendo filtrada ao longo das várias etapas, e sendo consequentemente cada vez mais "purificada", de modo a obter a representação final, conforme se pode ver na imagem da Figura 6.

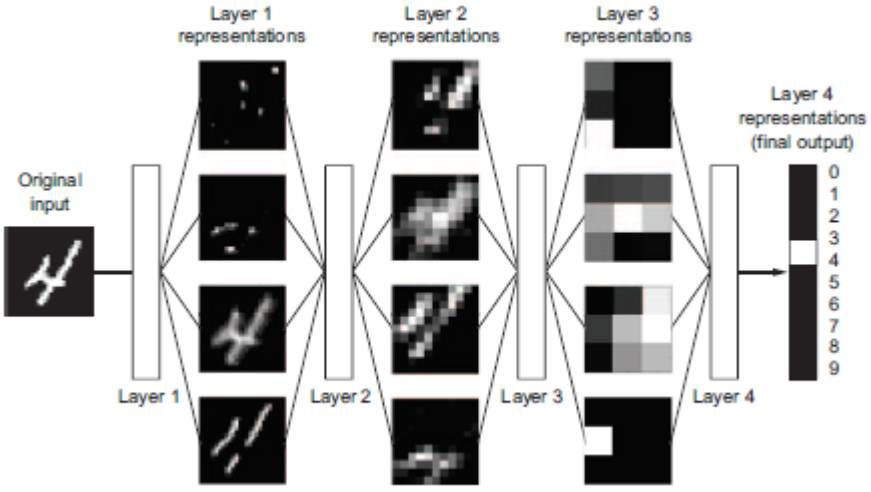


Figura 6 - Exemplos de Representações em Classificação Numérica, retirado de (Chollet, 2018).

A lógica destas operações segue o princípio de *input-to-target*, que consiste na tentativa de mapear os *inputs* nos objetivos ou categorias definidos (alvos). Esta ação é ponderada através da observação exaustiva de vários *inputs* e alvos. Este mapeamento é conseguido através das transformações dos dados realizadas nas *camadas* anteriormente mencionados. Estas tranformações são aprendidas na fase de treino a partir de vários exemplos. Todas as especificações e comportamentos que uma dada camada pode tomar ao realizar estas transformações são definidas nos pesos das camadas, que consistem essencialmente num conjunto de números. Pode-se considerar que o processo de aprendizagem do algoritmo se baseia no constante ajuste dos valores numéricos presentes nestes pesos. Este processo vai aumentando de dificuldade, à medida que o tamanho e profundidade da rede neuronal aumentam (Chollet, 2018). Na Figura 7, pode observar-se uma parte bastante importante deste tipo de algoritmos, que consiste em avaliar a qualidade e proximidade da previsão formulada pelo modelo de DL em relação ao resultado esperado (*True Target)*. A *Loss Function* computa um valor de seu nome *Loss Score*, que representa a distância da previsão em relação ao resultado esperado. O objetivo é atingir valores o mais pequenos possíveis, quando se fala deste valor.

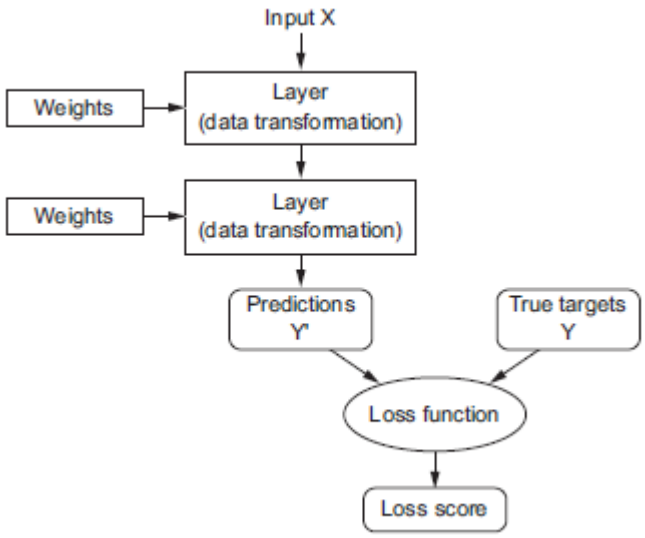


Figura 7 - Análise da qualidade do *output* obtido, retirado de (Chollet, 2018)

Pondo em prática o princípio anteriomente explicado num sistema deste género, em que se vai ajustado o valor dos pesos à medida que as várias simulações vão acontecendo, e as previsões vão sendo formuladas, obtemos um processo cujo funcionamento pode ser descrito pelo esquema apresentado na Figura 8. Desta forma, através do *Loss Score* obtido, é possível ir corrigindo os valores numéricos presentes nos pesos através de um otimizador, de modo a obter previsões cada vez mais próximas do resultado esperado, descendo assim o valor do *Loss Score*. Este conceito é conhecido como Algoritmo de Retropropagação, e é o principal ponto de distinção de um algoritmo de DL para um algoritmo mais geral de ML.

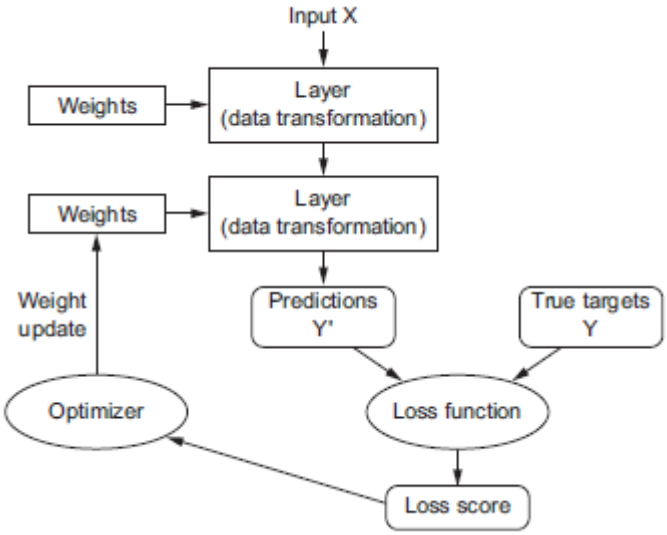


Figura 8 - Análise da qualidade do *output* obtido com ajuste dos *pesos*, retirado de (Chollet, 2018)

Este comportamento possibilita uma correção automática ao algoritmo geral, dado que o modelo vai sendo corrigido ao longo das iterações, através da alterações dos pesos. Numa primeira fase, originalmente, os valores são maioritariamente aleatórios, e irão originar *Loss Scores* muito altos. À medida que estas correções se vão sucedendo, esses valores, inicialmente aleatórios, vão sendo ajustados, e o *Loss Score* vai sendo cada vez mais reduzido. Na imagem da Figura 9, é possível visualizar graficamente a redução constante do *Loss Scor,e* ao longo das centenas de tentativas realizadas, e ainda a percentagem de "*accuracy*" do algoritmo em causa de 0 a 1.

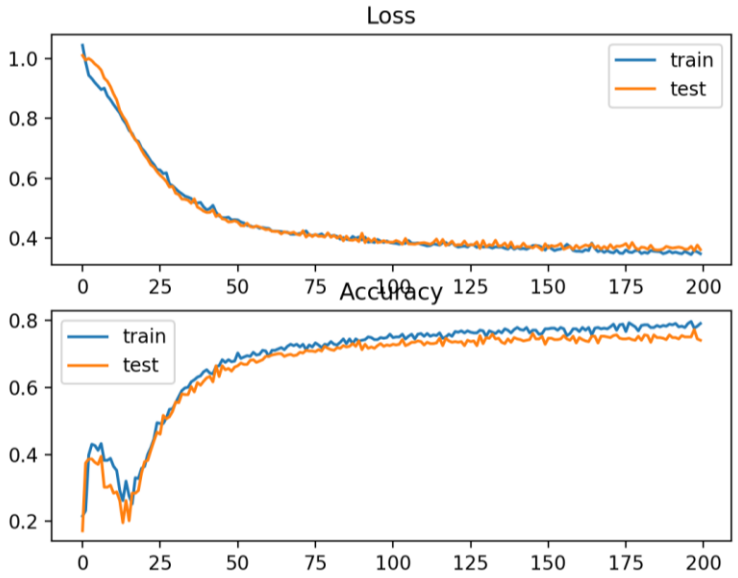


Figura 9 - *Loss Score* e *Accuracy*, retirado de (Brownlee, 2019)

Nesta abordagem ao DL, é importante salientar mais um fator bastante vantajoso do mesmo, que tem como base evitar o *Feature Engineering*. Este conceito trata-se da ação humana de tornar o *output* mais amigável à leitura para o algoritmo, de modo a este ser processado mais facilmente pelo mesmo. Isto era um comportamento bastante recorrente em algoritmos de ML, os quais envolviam transformar o *input* com base em um ou dois paradigmas de formulação de representações, como por exemplo projeções *high-dimensional non-linear* como os SVM's (*Support Machine Vector*) ou *Decision Trees*, que se baseiam em condições sucessivas que levam à categorização dos dados. Os modelos Kernel também se suportam muito de SVM's. Um modelo Kernel é um algoritmo de classificação que permite definir os *Decision Boundaries* (Figura 10), que são as separações formuladas, para que seja possível classificar dados com características semelhantes, formando assim as várias categorias.

Figura 10 - Decision Boundary, retirado de (Chollet, 2018)

O DL passa por ignorar completamente esta etapa de tratamento dos dados, simplificando *workflows* e algoritmos anteriormente demasiado sofisticados. As *features* ignoradas neste passo são facilmente aprendidas no início do processo de aprendizagem do algoritmo. As duas principais características que definem como um algoritmo de DL aprende informação consistem num esquema incremental "camada a camada", em que múltiplas e complexas representações são formuladas, e no facto dessas representações serem aprendidas como "um todo". Neste processo, o comportamento do camada anterior é constantemente atualizado de modo não só a corresponder às necessidades do camada superior, mas também adaptando-se às alterações que os camadas inferiores vão sofrendo para o satisfazer (Chollet, 2018).

3. Trabalho Relacionado

Neste capítulo, apresenta-se o estudo do estado da arte realizado no âmbito deste projeto, dando particular foco a trabalhos e projetos realizados na área de *Deep Learning*, com ênfase na leitura e classificação de conteúdo áudio. O objetivo é concluir quais são as tecnologias e técnicas mais populares e eficazes para esta área, nomeadamente os tipos de redes neuronais, algoritmos de conversão de áudio e leitura das métricas, bem como analisar as percentagens obtidas nas validações e testes de cada estudo.

3.1 Estudo do estado da arte

Nesta secção, irão ser descritos alguns dos trabalhos analisados durante o período de pesquisa e investigação realizadas no desenvolvimento do projeto. O principal intuito foi o de conhecer as tecnologias mais utilizadas e populares, bem como o processo de desenvolvimento de um modelo de ML e DL.

O trabalho apresentado em (Li, et al., 2017) é focado na comparação de diversos modelos de DL, enquadrados no desafio de deteção e classificação de cenários acústicos e eventos com base no *dataset* usado para este estudo (DCASE 2016). O trabalho apresenta a classificação de 15 cenários acústicos, tanto em contexto de espaço aberto como de espaço fechado. Os autores recorem a MFCC (*Mel-frequency cepstral coefficients*), extraídos através de OpenSMILE, que posteriormente são usados em quatro modelos: GMM (*Gaussian Mixture Model*), DNN (*Deep Neural Network*), RNN (*Recurrent Neural Network*), CNN (*Convolutional Deep Neural Network*) e i-vector. Para cada um dos modelos, foram usadas características de processamento de áudio diferentes, de modo a obter mais resultados e realizar uma análise mais aprofundada da taxa de sucesso dos modelos, com base na técnica de processamento de áudio que foi utilizada. As técnicas usadas neste estudo foram as seguintes:

- Monaural and Binaural MFCC;
- Smile983 & Smile6k;
- LogMel.

A comparação das técnicas para o sucesso de cada modelo pode ser visualizada na Figura 11, em função da média da percentagem de sucesso em cada caso.

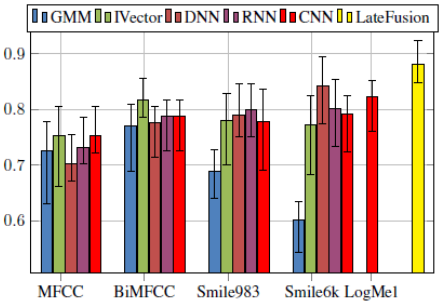


Figura 11 - Comparação de técnicas de processamento de áudio, retirado de (Li, et al., 2017)

Nos resultados apresentados no que toca a valores de *accuracy* para os tipos de modelos utilizados neste estudo, de entre o modelos utilizados, sendo eles GMM, I-Vector, DNN, RNN e CNN, as redes neuronais geraram bons valores na casa dos 80-100% em alguns dos locais testados. A maioria desses locais estão enquadrados no meio urbano, como por exemplo sons de carro, metro, elétrico, escritório, entre outros. As, DNN, RNN e CNN tiveram percentagens de cerca de 84.2%, 80.2% e 82.2% respetivamente. Na Tabela 4, pode-se ver estas comparações entre o desempenho dos vários modelos para cada ambiente disponível no estudo em causa, medida em percentagem de sucesso na identifação do local correto.

Tabela 4 - Precisão de Modelos, retirado de (Li, et al., 2017)

	GMM	I-Vector	DNN	RNN	CNN	Fusion
Beach	69.3	80.7	89.8	80.3	78.7	92.3
Bus	79.6	82.4	95.3	88.6	72.1	95.3
Cafe/Rest.	83.2	70.0	69.9	64.7	66.4	79.9
Car	87.2	96.1	87.2	88.8	99.1	97.2
City	85.5	90.0	97.3	96.2	93.5	89.2
Forest	81.0	92.0	96.4	95.0	99.8	99.8
Grocery	65.0	93.8	79.3	75.5	85.3	96.2
Home	82.1	65.2	84.8	75.7	82.9	88.2
Library	50.4	76.1	81.2	81.6	72.7	86.2
Metro	94.7	83.5	97.3	93.7	98.7	92.3
Office	98.6	93.1	99.7	79.6	97.6	99.7
Park	13.9	78.6	49.4	45.8	45.7	71.2
Resident	77.7	66.5	76.9	68.7	81.6	77.0
Train	33.6	72.4	51.1	61.2	59.2	65.2
Tram	85.4	84.6	97.0	90.7	91.7	92.2
Average	72.5	81.7	84.2	80.2	82.2	88.1

Vale a pena salientar os bons resultados para quase todos os ambientes, à exceção de locais como “Parque”, “Área Residencial”, “Comboio” e “Metro”. É importante referir que nos sons provenientes do meio ambiente existe uma maior dificuldade em encontrar padrões do que, por exemplo, nos padrões vocais do ser humano. Isto porque existe uma aleatoriedade maior, mais inconsistência e até mesmo interferência no som proveniente do meio-urbano. Neste estudo, pôde-se concluir que os modelos GMM e i-vectors, neste tipo de situações, sofrem de um conceito chamado de “*curse of dimensionality*”, ou seja, o espaço dimensional tende a crescer exponencialmente, enquanto o volume de dados em si se mantém o mesmo, o que leva a amostras altamente escassas, o que poderá trazer piores resultados em zonas como “Comboio” e “Metro”. Já as DNN com o uso de Smile6K têm uma percentagem de acerto de 88,2%, o valor mais alto do estudo. Isto deve-se à maior facilidade que as DNN têm de aprender representações da técnica em si, no caso o Smile6K. No entanto, é importante frisar que, apesar de nesta situação, o modelo DNN ter sido o que teve melhor desempenho, isso não significa, forçosamente, que o comportamento seja idêntico para todas as situações do espaço urbano, devido à sua aleatoriedade. Adicionalmente, também, não se pode concluir que não existe nenhum modelo (incluindo o DNN) que supere todos os outros modelos, em todas as técnicas de processamento de áudio utilizadas neste estudo.

Em (Adavanne & Virtanen, 2017) são usados modelos CNN e RNN com técnicas de canal singular e binária para a deteção de eventos sonoros. A avaliação de desempenho dos mesmos é reportada usando as métricas de percentagem de erro e F-score. Através de um sinal de áudio que faz de *input*, as técnicas de processamento de áudio originam extrações de intervalos de tempo consecutivos, de cada intervalo dos canais de áudio da amostra. Consequentemente, estes são inseridos nos modelos CNN e RNN multi-canal, sendo posteriormente mapeados pelo modelo numa das categorias disponíveis do *dataset* em questão, neste caso o DCASE 2017. Os SED (*Sound Event Detection*) em questão poderão ser mapeados conforme mostra a Tabela 5.

Tabela 5 - *Sound Event Detection*, retirado de *(Adavanne & Virtanen, 2017)*

Sound events	Length
brakes squeaking	67.6
car	2541.5
children	346.1
large vehicle	727.0
people speaking	630.6
people walking	1079.2
Total	5391.9

As métricas obtidas neste estudo para o ER (*error rate*) e para o F-score foram os valores que se podem visualizar na Tabela 6, em função de cada técnica de processamento de áudio utilizada.

Tabela 6 – Resultados obtidos em função das técnicas usadas, retirado de (Adavanne & Virtanen, 2017)

	Development		Challenge	
Audio features	ER	F	ER	F
baseline [22]	0.69	56.7	0.94	42.8
mbe	0.55	69.3	**0.79**	41.7
bin-mbe	0.52	69.1	0.80	**42.9**
bin-mul-mbe	**0.50**	**70.3**	0.85	41.4
bin-fft	0.55	66.9	0.87	36.2

Algumas das conclusões retiradas deste estudo afirmam que o uso da técnica de canal de áudio singular (*mbe*) foi superior ao uso da *baseline*. Também se pode ver que o uso de canais binaurais ou singulares tiveram resultados similares no *dataset* em questão, apesar do ER binário ser igual ou ligeiramente melhor que o singular. A conclusão mais importante a salientar é o facto de *bin-mul-mbe* ter um ER consideravelmente menor que o uso de *mbe*. A validação e o *training loss* do modelo que usou o *bin-fft* foram também sensivelmente menores do que as outras técnicas de processamento de áudio utilizadas, sugerindo que a quantidade dos dados para esta técnica era possivelmente mais pequena para a mesma encontrar o valor dos pesos dos modelos CNN e RNN.

O trabalho apresentado em (Park & Cho, 2020) utiliza como *dataset* o Urban8K Dataset, com um leque de mais de 8000 amostras de áudio categorizadas em 10 classes diferentes, conforme se pode observar na Tabela 7. Também recorre ao uso de *mel-spectograms* como representação principal dos ficheiros de áudio.

Tabela 7 - Urban8K, retirado de (Park & Cho, 2020)

No	Class name	No	Class name
1	Air_conditioner	6	Engine_idling
2	Car_horn	7	Gun_shot
3	Children_playing	8	Jackhammer
4	Dog_bark	9	Siren
5	Drilling	10	Street_music

Os WAVs, formato de áudio disponível no *dataset* do Urban8K, foram convertidos para imagens PNG (*Portable Network Graphics)*, e usaram-se características MFFC extraídas com a biblioteca Librosa do Python. Através do MFFC é conseguido o PNG, formato de dados que é utilizado no treino e aprendizagem do modelo usado deste estudo. O processo de conversão e funcionamento descrito neste artigo pode ser visualizado na Figura 12. Com o uso de MATLAB, Googlenet, SqueezeNet e Resnet50 envolvidos na criação dos PNG, os autores obtiveram os resultados apresentados na Figura 13.

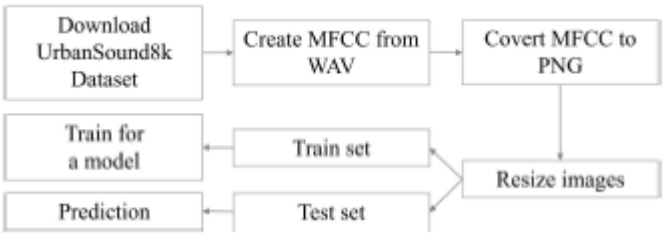


Figura 12 - Diagrama Lógico do processo do modelo de D, retirado de L (Park & Cho, 2020)

Epoch	Accuracy (Elapsed time)			The Number of GPU
	GoogLeNet	SqueezeNet	Resnet50	
32	90.44% (52' 24")	85.62% (11' 14")	96.29% (122Z' 57")	Single
	89.33% (34' 50")	92.3% (17' 35")	96.38% (81' 10")	Dual
128	89.7% (107' 7")	95.17% (31' 40")	97.22% (231' 32")	Single
	92.95% (67' 13")	93.97% (35'13")	95.73% (169' 40")	Dual
256	92.49% (157' 15 ")	94.25% (48' 11")	96.38% (320' 58")	Single
	93.14% (74' 47")	93.78% (40' 08")	96.01 (206' 28")	Dual

Figura 13 - Medidas de desempenho obtidas, retirado de (Park & Cho, 2020)

Em Choi, et al., 2018 é apresentado um estudo que compara uma variedade de técnicas de *input* de pré-processamento de áudio, que podem variar na forma como a relação tempo-frequência das representações é feita, a magnitude da compressão logarítmica, a influência dos pesos aplicados às frequências, e a escala das mesmas. É objetivo deste estudo fazer perceber e provar que a eficiência e desempenho do treino do modelo pode variar consoante o método de preprocessamento do *input*. As redes neuronais podem representar uma dada função, mas isso não significa que essa fução será, garantidamente, bem aprendida. O estudo refere que os *mel-spectogtams* (forma usada no trabalho para representar um espectrograma) são vantajosos para representar os dados, pois os mesmos podem conter informação suficiente para uma análise vantajosa, apesar do seu tamanho mais pequeno, em comparação a, por exemplo, *short-time Fourier*. Neste trabalho, foram utilizadas estratégias de áudio pré-processadas de modo a alimentar o modelo escolhido, que consistiu em CNN (*Convolutional Neuronal Networks*) com 2D Kernels e 2D *Convolution Axes*. O *dataset* utilizado foi o MSD (*Million Song Dataset*). No que toca às bibliotecas e *frameworks* de DL usadas em Python, foram respetivamente a Librosa e Kapre, e Keras e Theano. Na Figura 14, pode ver-se uma comparação do desempenho do modelo CNN, com base no método de pré-processamento de áudio utilizado, que é o foco deste treino.

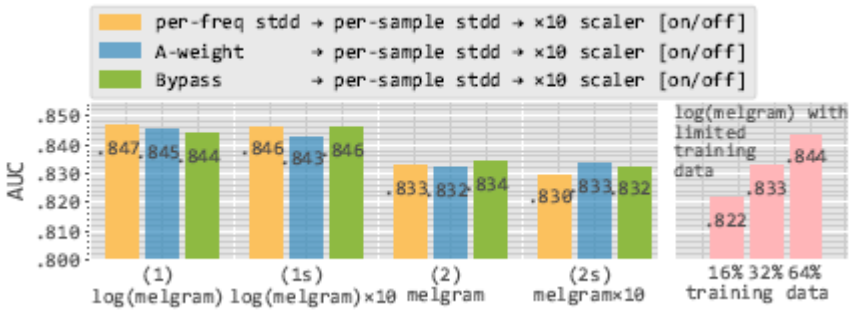


Figura 14 – Resultados obtidos com base na estratégia pré-processada de áudio, retirado de (Choi, et al., 2018)

O fator importante a retirar deste estudo é a análise ao desempenho de um modelo CNN quando usado uma dada técnica de pré-processamento de áudio, que, neste caso, foi o *mel-spectogram* e as suas variantes.

Em (KHAMPARIA, et al., 2019) é analisada a aplicação de modelos DL para a deteção e classificação de eventos sonoros, com mais foco no reconhecimento vocal, baseados na criação de espetrogramas. É importante frisar que a classificação do *Big Data* com recurso a algoritmos de DL tem sido cada vez mais valorizada no que toca a assuntos relacionados com o conceito de IoT (*Internet of Things*). Foram usados dois *datasets* neste artigo, o ESC-10 e o ESC-50. Neste trabalho, a deteção de som é conseguida à custa de três técnicas diferentes de pré-processamento de sinais áudio, que permitem dividir os mesmos em diferentes segmentos. Pode-se ver, na Figura 15, um exemplo de espetrograma com uma forte componente visual.

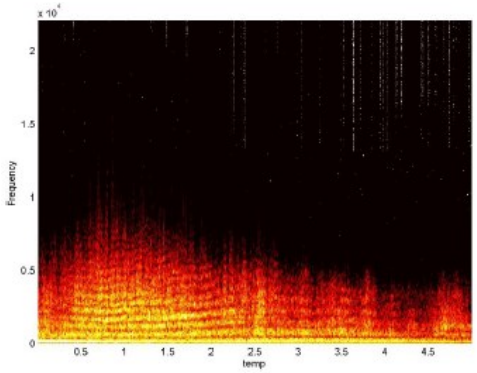
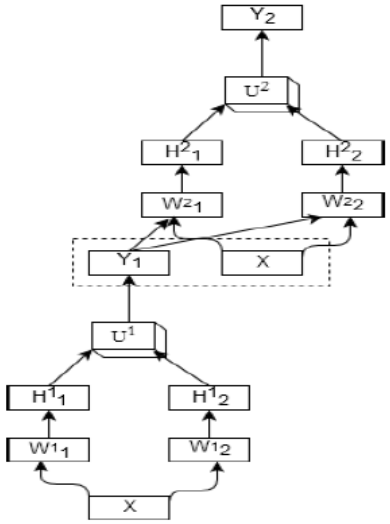


Figura 15 - Exemplo de espectrograma, retirado de (KHAMPARIA, et al., 2019)

Alguns exemplos de modelos utilizados para o reconhecimento de voz humana são: *Decision Trees*, *Random Forest* e K *Nearest Neighbor*. Nos últimos anos também se tem usado bastante os seguintes modelos: *Spectrogram image features* (SIF), *Stabilized auditory image* (SAI), *Linear prediction coefficients* (LPC); ou ainda técnicas de *machine learning* mais simples como: *Gaussian mixture model*, *random forest*, *multi-layer perceptron* e *deep learning networks* em sistemas de reconhecimento de áudio. No estudo apresentado, o autores usaram uma técnica de ML baseada num *Unsupervised encoder* para um mapeamento não linear. Foi, também, usado o Tensor *Deep Stacking Network* (semelhante ao conceito de *back-propagation* já abordado anteriormente na secção 0), com uso de mapeamento bilinear e balanceamento das previsões e pesos da próxima camada com base num tensor de terceira ordem (U), conforme se pode ver na Figura 16.



$$y = \mu\left(h_{(1)}, h_{(2)}\right) \cong (\mu \times_1 h_1) \times_2 h_2$$

Figura 16 - Tensor *Deep Stacking Network*, retirado de (KHAMPARIA, et al., 2019)

No diagrama lógico da Figura 17, pode ver-se o processo criado para a composição do modelo concebido para este estudo. Foram usadas algumas tecnologias, como por exemplo MATLAB, TDSN TOOLKIT e Keras (para uso de CNN).

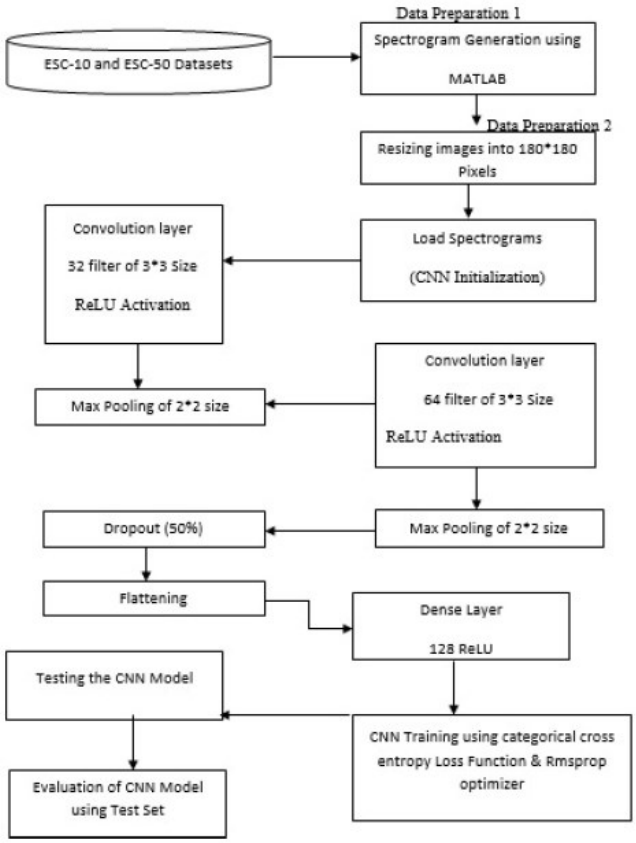


Figura 17 - Diagrama Lógico, retirado de (KHAMPARIA, et al., 2019)

Tabela 8 apresenta os resultados publicados relativos às combinações de tecnologias em função da técnica de produção de espetrograma, modelo utilizado, e percentagem de sucesso obtida. Como se pode ver, o desempenho com melhores valores de precisão (do inglês *accuracy*) foi obtido com o uso de uma CNN.

Tabela 8 - Resultados obtidos, retirado de (KHAMPARIA, et al., 2019)

Spectrogram driven Sound Systems	**Techniques/Architectures Used**	**Performance (Accuracy %)**
MFCC-SVM	Support Vector Machine (SVM)	34.1 %
MPEG-7	Decision Trees	33.6 %
Gabor	Random Forest	39.0%
GTCC	K-Nearest Neigbor	40.8%
MFCC-MP	Multi layer Perceptron	43.24%
CNN (Proposed)	**Convolutional Neural Network**	**73%**
TDSN (Proposed)	**Tensor Deep Stack Network**	**56%**

A Tabela 9 mostra as percentagens de sucesso conseguidas, mas desta vez em função do modelo usado e do *dataset* utilizado.

Tabela 9 - Resultados obtidos com base no *dataset* usado, retirado de (KHAMPARIA, et al., 2019)

Technique	Dataset Used	Accuracy
CNN (K. Piczak, 2015)	ESC-10	73%
CNN (Our Approach)	ESC-10	77%
CNN (K. Piczak, 2015)	ESC-50	44%
CNN (Our Approach)	ESC-50	49%
TDSN (B. Hutchinson et.al, 2013)	ESC-10	53%
TDSN (Our Approach)	ESC-10	56%

Os modelos com maior precisão utilizados neste estudo foram, como se pode observar nas tabelas anteriores, as CNN e as TDSN. Pôde-se constatar que as CNN são vantajosas quando se pretende aprender as técnicas de processamento de áudio e realizar as previsões dos próximos *layers* com melhor rigor. Uma vantagem que se encontrou, que pode melhorar os resultados obtidos, é a realização de *drop connect* ou *dropout* com 50% de probabilidade. Existem sempre mais pesos do que "nós"(do inglês *nodes*) presentes na rede neuronal, e aqui pode-se tirar vantagem no uso destas técnicas de modo a tentar encontrar o melhor modelo com o treino mais eficiente. Este artigo reforça a possibilidade de usar CNN e TDSN para problemas de classificação de sinais de áudio com uso de espetrogramas, com percentagens de sucesso obtidas na ordem dos 77%, 49% e 56%.

A análise dos trabalhos descritos permitiu compreender melhor os principais aspectos da componente de processamento, análise e categorização de áudio, com recursos variados de DL, os quais se podem dividir nas seguintes etapas:

- *Dataset* utilizado e categorização disponível;
- Tipos de dados e escolha do formato de *input*;
- Técnica de processamento de áudio escolhida;
- Escolha, treino e desenvolvimento do modelo de *Deep Learning* escolhido;
- Análise de desempenho e conclusões a retirar.

Todos os trabalhos apresentados passam sensivelmente por este processo, e/ou por comparações entre as várias possibilidades de escolhas, combinações de tecnologias, e consequências no desempenho que isso possa trazer. Como mencionado em alguns dos artigos, consoante o âmbito para que se pretendem os dados, condições do meio-envolvente, e a própria aleatoriedade desta área e dos SED (*Sound Event Detection*), o desempenho do modelo DL poderá variar de caso para caso. A conclusão mais importante a retirar é que a eficiência de um modelo poderá não só depender da melhor combinação de tecnologias, mas também das condições a que o mesmo for submetido no seu treino e desenvolvimento.

3.2 Aplicações Relacionadas

Nesta secção, serão apresentadas duas aplicações já disponíveis e que se consideram pertinentes para o estudo global do projeto, pois possuem características semelhantes às que se pretendem implementar no sistema final. Assim, será descrita, de forma resumida, a aplicação (Shazam, 2021), a qual consiste numa ferramenta de reconhecimento de música, e o Skynet, que consiste num sistema de segurança desenvolvido com recurso a Inteligência Artificial e que é usado pelo estado chinês.

3.2.1 Shazam

O Shazam (Figura 18) é uma aplicação desenvolvida para *smartphone,* pela Apple Inc., com o propósito de identificar músicas através do microfone disponível no dispositivo no qual a aplicação está instalada. Esta aplicação gratuita permite aos utilizadores identificarem, facilmente, apenas por um pequeno excerto de áudio captado, a música de fundo que está a ser reproduzida num dado momento. Isto é conseguido através de comparações com vários registos de músicas, os quais estão disponíveis numa base de dados *online,* de modo a tentarem ser encontradas semelhanças. Estas comparações são feitas com base na frequência, amplitude de onda, e nas próprias oitavas musicais que constituem as pautas de música tradicionais. Cada nota musical tem uma frequência característica associada, conforme explicado em (Christophe, 2015), e através dos sinais analógicos detetados é possível formular espectrogramas e amostras, que são consideradas as representações visuais de uma amostra de som transformada para imagem.

Figura 18 – Shazam (Christophe, 2015)

A análise realizada na aplicação segue a lógica representada no fluxograma apresentado na Figura 19. Como pode ser observado, é realizada uma consulta ao *dataset,* pela amostra disponível no mesmo, mais semelhante à amostra recém-recolhida. Caso seja encontrado algum resultado compatível com a amostra nova, serão carregados todos os dados associados a esse resultado.

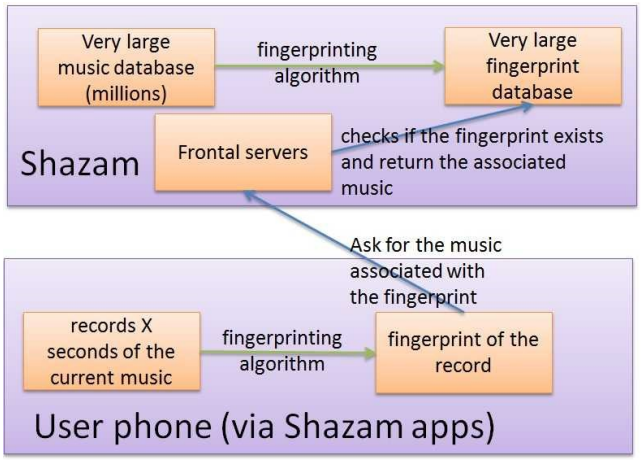


Figura 19 - Fluxograma Shazam, retirado de (Christophe, 2015)

As características e lógica de análise da aplicação Shazam são bastante vantajosas e de utilização intuitiva e simples para o utilizador, não só pelo ágil funcionamento, como também pelos menus de aplicação bastante simplificados que se pode ver na Figura 20.

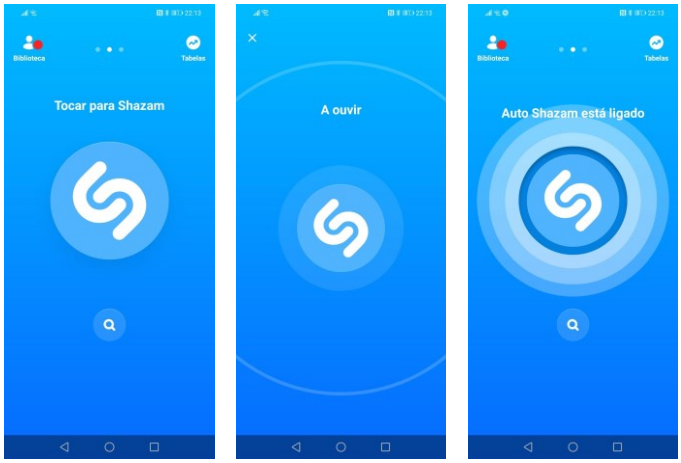


Figura 20 - Menus Shazam

3.2.2 Skynet

O Skynet é um sistema de segurança, usado pelo governo chinês, que tem como propósito vigiar e monitorizar a população, com a premissa de precaver e combater a criminalidade urbana. É um sistema de segurança altamente equipado com mais de 20 milhões de câmaras presentes no meio-urbano, muito proclamado como "o maior sistema de segurança do planeta" (Mozur, 2018). É um sistema que se suporta em vídeo captado em tempo real, com recurso a algoritmos de inteligência artificial capazes de encontrar e identificar um cidadão com grande precisão e percentagens de erro reduzidas. O reconhecimento facial é realizado em tempo real (ver Figura 21).

Figura 21 - Reconhecimento Facial Skynet (Mozur, 2018)

Como nem tudo são vantagens, com o crescimento desta tecnologia, e o uso deste tipo de recursos, surgem problemas relacionados com a violação de privacidade, e levantar-se-ão questões no que toca a este direito, bem como outros que poderão surgir do uso deste sistema. Que outras utilizações poderá dar o governo chinês a este sistema, para além do combate à criminalidade? Este sistema tem forte presença em locais estratégicos de grande afluência, como estações de comboios e metro, aeroportos, espaços abertos de grande movimento. Com um recurso tão poderoso, evidentemente, poder-se-á criar mais propósitos para este sistema, alguns dos quais poderão ser considerados polémicos e/ou controversos, como espionagem e uso indevido da informação recolhida. Todos os sistemas de segurança têm de lidar com este tipo de assuntos, e não é um tema que será essencial para o âmbito deste projeto. Alguns detalhes sobre os lados mais negativos e controversos de um sistema desta dimensão poderão ser encontrados em (Smith, 2017). O importante e pertinente a retirar deste sistema de segurança passa pelo posicionamento estratégico dos pontos de captação de dados, bem como a forte presença de uma componente de Inteligência Artificial, de modo a realizar leituras e análise do *input* recolhido. Através de enormes sistemas de informação e de *Big Data* (Yu, 2017) , o sistema cumpre o seu propósito de identificar indivíduos e fazer a monitorização do meio urbano.

4. Desenvolvimento e Treino do Modelo

Neste capítulo, é apresentado todo o processo de desenvolvimento do modelo que se pretende integrar na aplicação móvel para classificação de ruídos urbanos, validação e realização de testes de desempenho com dados externos aos utilizados no *dataset* de treino.

4.1. Observação de amostras de ruído

Antes do desenvolvimento do modelo para classificação de ruídos foi feito um trabalho prévio que consistiu em obter exemplos visuais que refletem os aspetos que diferenciam os vários tipos de som, num formato de onda de som, conforme pode ser observado na **Erro! A origem da referência não foi encontrada.**.

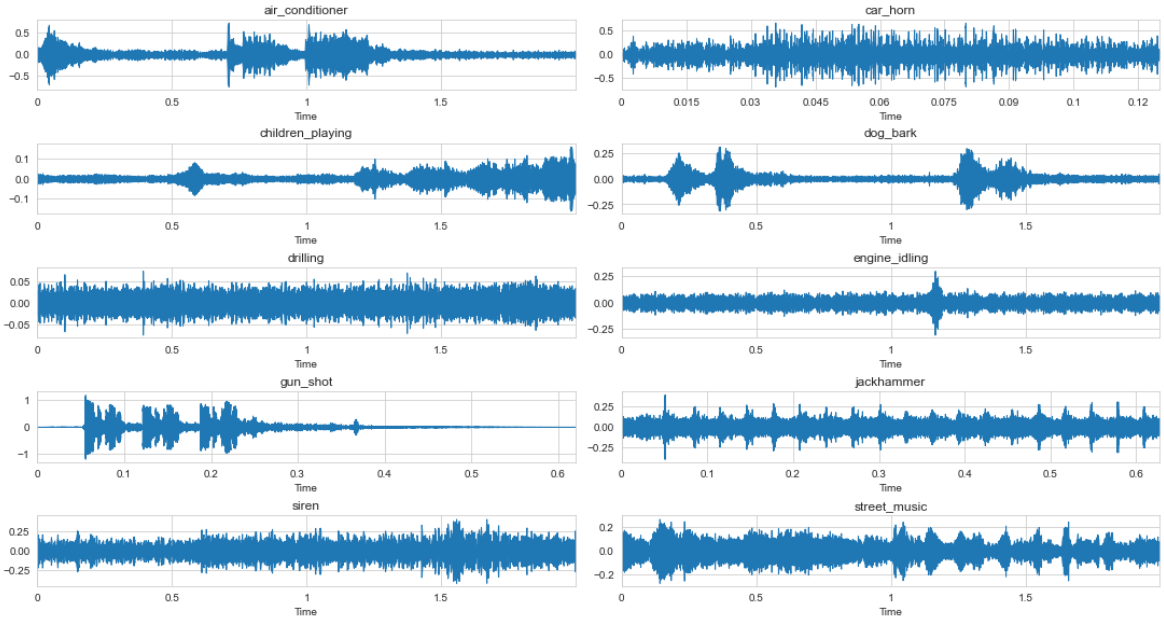


Figura 22 - Waveplot exemplo de output

Foram também recolhidos exemplos no formato de espetrograma, para cada uma das classificações possíveis, ilustrados na Figura 23. De salientar, que o processo de criação destes exemplos não é utilizado neste treino, no entanto, para contexto de entendimento humano, é uma excelente forma de visualizar as propriedades de cada categoria de áudio, num formato de fácil compreensão para o utilizador comum.

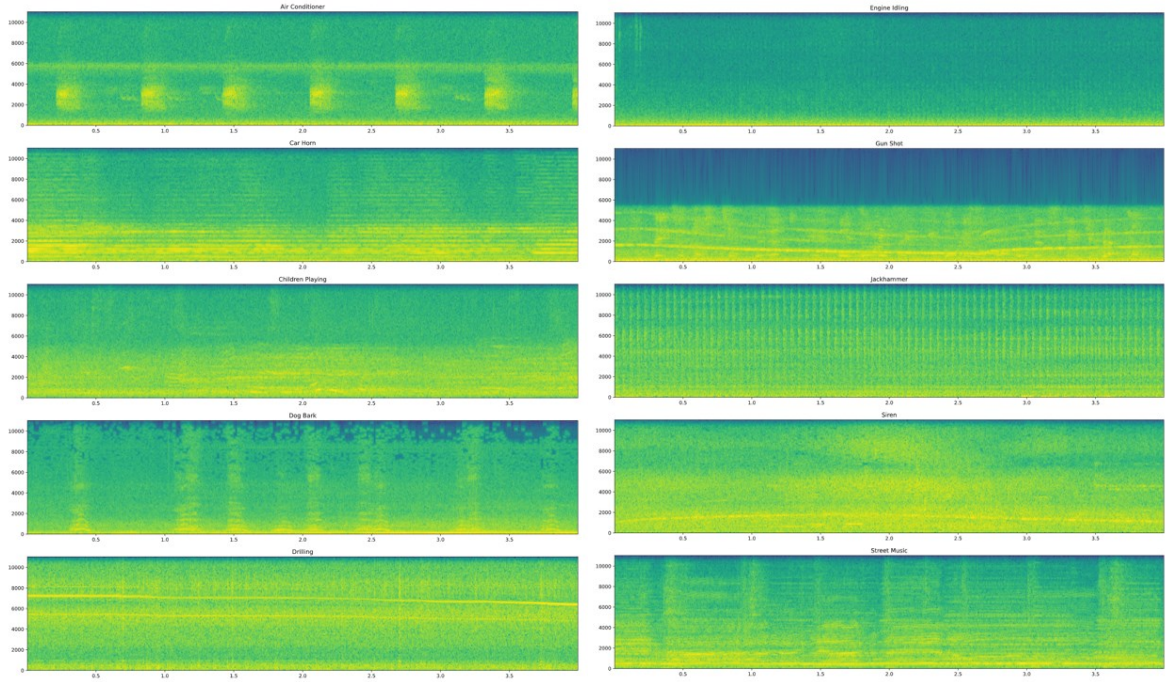


Figura 23 - Espetrogramas por classificação Urban 8K (Exemplo 1)

Através da forma como a onda padrão de som de cada classificação está representada visualmente, a perceção humana consegue notar diferenças nas características de amplitude de onda e frequência entre as classificações, como, por exemplo, um padrão mais constante e forte em *drilling*, ou algo mais suave e irregular como *children_playing*.

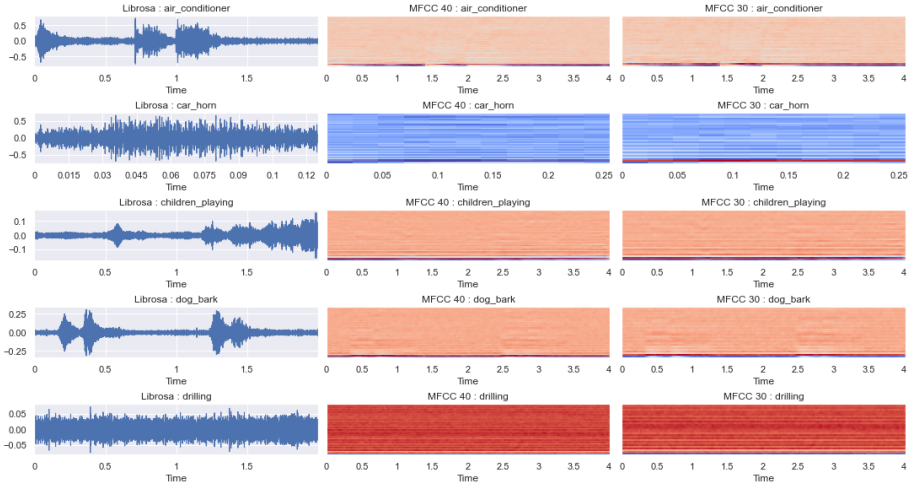


Figura 24 – Exemplo de Espetrograma Urban 8K (Exemplo 2)

4.2. *Dataset* de Treino

O *dataset* utilizado denomina-se *Urban8K Dataset*, e contém 8732 amostras de áudio categorizadas numa de 10 categorias, sendo elas: *air_conditioner, car_horn, children_playing, dog_bark, drilling, enginge_idling, gun_shot, jackhammer, siren,* e *street_music*. Os ficheiros de áudio estão no formato .wav; têm uma duração de cerca de 2 a 7 segundos; estão distribuídos por 10 pastas (fold1 a fold 10), o que possibilita uma separação entre as amostras em partes de igual importância, tornando assim possível o uso parcial do *dataset*. O mesmo inclui um .CSV de metadados essencial para o processo, que contém a classificação das amostras, e é o ponto chave da leitura do *dataset*.

Este *dataset* foi escolhido entre outros pela sua fácil acessibilidade e pelo facto do mesmo ser *open-sourcing* e, além disso, também pelo mesmo ser utilizado em alguns dos estudos apresentados nos trabalhos realizados deste documento, no capítulo 3.

4.3. Redes Neuronais Convolucionais

Uma CNN (exemplo na Figura 25) é um tipo de modelo bastante recetivo a dados em formato de *arrays*, com um dado formato para posterior processamento nas camadas da rede. Ao contrário de, por exemplo, uma rede neuronal recorrente (RNN), uma rede convolucional baseia-se em transformações de dados, utilizando os filtros baseados nos *arrays* que estão a ser utilizados como métricas. Já as redes recorrentes focam-se mais na reutilização das funções de ativação anteriores, para ir tentando aprimorar os *outputs* de uma camada para a sua seguinte. As RNN são mais focadas numa componente em que a informação muda ao longo do tempo. Apesar de um ficheiro de áudio, ao contrário de, por exemplo, uma imagem, envolver variação da informação ao longo do tempo, à semelhança de um vídeo, neste trabalho a análise do áudio é feita seguindo a lógica implementada na análise de imagens, ou seja, com processamento dos dados e métricas através de *arrays*, em vez de uma recorrência temporal que é característica das RNN. É importante ainda mencionar que, nos trabalhos realizados da área de ML, analisados ao longo do capítulo 3, também apresentam, regra geral, resultados favoráveis às redes neuronais no geral, por exemplo (KHAMPARIA, et al., 2019), (Li, et al., 2017) e (Choi, et al., 2018).

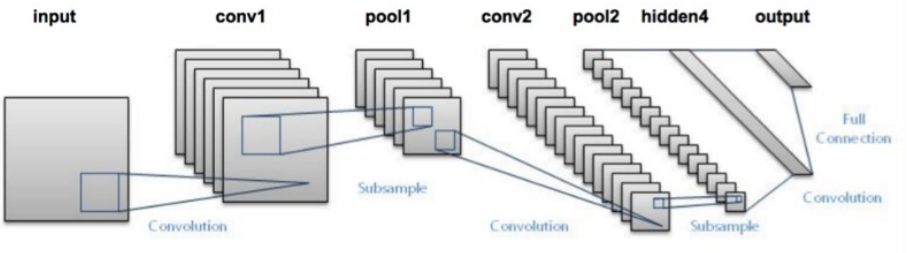


Figura 25 – Estrutura de uma Rede Convolucional

No processamento de imagem, geralmente, os valores processados nesses arrays são os valores RGBs de cada pixel. Neste caso, os valores processados são os valores MFCC extraídos anteriormente. Estes dados são representados cada um deles em duas dimensões, sendo eles x_train e y_train, e ainda x_test e y_test, respetivamente. Existem várias formas de efetuar o treino de um modelo. Para redes neuronais, uma forma muito comum é a "validação cruzada", em que o *dataset* é divido em subconjuntos. Este método de treino usa um desses subconjuntos do *dataset* (que está posteriormente catalogado, seguindo a abordagem de uma aprendizagem supervisionada), para servir como coleção de testes para todos os restantes 9 subconjuntos, que serão materiais de treino. As quantidades de dados do *dataset* a utilizar para treino e para testes são definidas na fase de conceção do método de treino do modelo e, neste caso, foi definido um conjunto de treino com cerca de 90% dos dados e os restantes 10% para dados de teste.

4.4. Definição e Treino do Modelo

Nesta fase de desenvolvimento do trabalho foram usadas várias bibliotecas Python, para as seguintes tarefas:

- *Pandas* – leitura de ficheiros CSV;
- *Numpy* – cálculos matemáticos e "reshaping" de arrays;
- *matplotlib.pyplot* – visualização de gráficos;
- *os* – interação com diretorias do sistema operativo;
- *tqdm* – visualização de barras de progressão para ciclos em Python;
- *librosa* – cálculo das métricas MFCC;
- *tensorlfow.keras* e *tensorflow* – conceção do modelo de ML.

Na conceção dos *arrays* de treino e teste, a forma como os dados do *dataset* foram divididos foi a seguinte: dos 10 *folders* existentes, os primeiros 9 foram usados para treino, e o último foi usado para testes, gerando um *array* de treino de 7896 amostras e um *array* de testes com 837 amostras. Estes valores são gravados posteriormente em ficheiros CSV, responsáveis por armazenar os parâmetros a utilizar no treino e teste do modelo a implementar.

O modelo desenvolvido consiste numa rede convolucional de duas dimensões, com duas camadas convolucionais. A estrutura da rede criada pode ser visualizada na Figura 26.

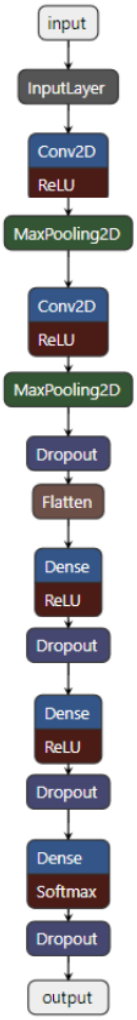


Figura 26 - Estrutura da CNN

Uma camada **Conv2D** tem dois argumentos obrigatórios, sendo eles um número inteiro que representa o número de filtros que a camada deve gerir, e ainda uma lista que determina as dimensões dos filtros desta camada, de seu nome **kernel_size**. Outro argumento opcional possível de especificar é o tipo de função de ativação existente nessa camada, que neste caso é do tipo **ReLu**. A função de ativação é responsável por aplicar a transformação ao *input* fornecido, de modo a passar essa transformação como *ouput* para a camada seguinte. O termo ReLu é um diminutivo para "*rectified linear activation function*", que é uma função linear que fornece os *outputs* mencionados anteriormente de forma binária, sendo essas duas opções o próprio *input* obtido ou 0. Este comportamento pode ser representado das seguintes formas (uma "*if condition*" ou uma função matemática na Figura 27):

```
if input > 0:
        return input
else:
        return 0
```

$$g(z) = \max\{0, z\}$$

Figura 27 – Representação da função de Ativação ReLu (Brownlee, s.d.)

Atualmente este tipo de função de ativação é muito utilizado pois o mesmo acelera a aprendizagem da rede, devido aos pequenos valores que vão sendo tratados (Brownlee, s.d.). Ainda existe outro tipo de função de ativação utilizada nesta estrutura, de seu nome **Softmax**, responsável pela normalização de um dado conjunto de valores K (números reais), para um intervalo de [0,1] para cada um desses valores.

A camada **Flatten** recebe um tensor de mais que uma dimensão, e simplifica o mesmo para uma única lista unidimensional. Na camada de **Pooling,** são analisados elementos que constituem um bloco de entrada, e é calculado um valor único a partir dos mesmos, o qual é guardado e utilizado como valor de saída. A ideia deste comportamento é diminuir o tamanho dos valores de entrada, em que, por exemplo, um bloco de tamanho 2x2 será transformado num *output* com metade do tamanho. Existem duas formas de realizar este processo: **MaxPooling2D** e **AveragePooling2D.** A **MaxPooling2D** foi a utilizada neste trabalho. Neste caso, o valor devolvido é o maior do conjunto. A opção **AveragePooling2D** devolve a média dos valores do conjunto. Numa camada de **Dense**, um dado conjunto de neurónios recebe *input* de vários neurónios posteriores. Cada neurónio recebe *inputs* de vários neurónios da camada anterior, existindo assim uma condensação da rede. A camada de **Dropout** é responsável por evitar o "*overfitting*" da rede, ou seja, tentar prolongar o tempo de treino da rede sem comprometer o futuro desempenho da rede. O *overfitting* acontece quando a rede converge prematuramente, comprometendo o seu desempenho na classificação de dados reais exteriores ao conjunto de treino.

Após definição da estrutura da rede, com o método model.compile() da biblioteca Keras, é concebido o compilador para o modelo. Este compilador pode receber três funções: função de otimização, função de perda e função de métrica. São passados 3 parâmetros String para cada uma destas funções, que definem qual das várias bibliotecas keras deve ser utilizada na compilação do modelo. Para o *otimizer* foi utilizada a *class* "Adam", que é um tipo de otimizador. Para a função de perda foi utilizada a *class* "Categorical_Crossentropy", que computa o valor de perda entre as *labels* fornecidas e as previsões obtidas. E, por fim, para as métricas, foi utilizada a *class* "Accuracy", que serve para calcular o quão frequentemente as previsões obtidas são iguais aos *labels*. Posteriormente, com a função model.fit() da biblioteca Keras, é inicializado o treino do modelo, e fornecidos os *arrays* de treino e teste. Também são fornecidos outros parâmetros, como o número de épocas de treino, que neste caso foram 30.

Ainda através da função model.evaluate() da biblioteca Keras, é possível obter os valores de *loss* e *accuracy* no modelo. Pode observar-se a progressão dos valores dos mesmos para este trabalho, em função das 30 épocas, relativos aos dados de treino, na Figura 28.

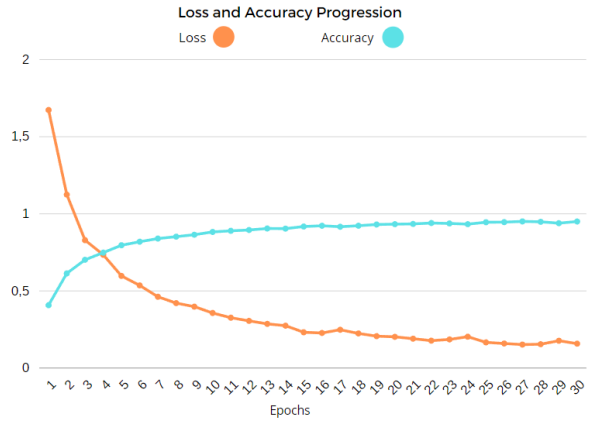


Figura 28 - Valores de perda e métricas no treino da CNN

O modelo, na primeira época de treino, tinha valores de perda (do inglês *loss value*) e precisão (do inglês *accuracy*) de 1.6736 e 0.4075, respetivamente, tendo terminado o treino com 0.1581de valor de perda e 0.9502 de precisão. Ainda relativamente à aplicação do modelo treinado nos dados de teste, obtiveram-se os seguintes valores de perda de 1.9718 e de precisão 0.6260.

Pretende-se a integração deste modelo na aplicação *Android*, portanto, foi realizada uma conversão do modelo criado previamente para um formato compatível com *Android*. O modelo é primeiramente extraído no seu formato original para uma diretoria isolada, com a função model.save() da biblioteca Keras e, de seguida, convertido para o formato TFLite (TFLite, s.d.), compatível com *Android*, com a função TFLiteConverter.from_keras_model(), da biblioteca Lite do Tensorflow.

Com o finalizar deste processo, obtém-se o modelo no seu formato original, o ficheiro TFLite para uso posterior do modelo num ambiente prático, e os valores utilizados no treino do modelo.

5. Componentes do Sistema Desenvolvido

Neste capítulo, são apresentados os restantes elementos do sistema global do projeto além do modelo de ML, que consistem em duas aplicações, uma *Android* para uso do utilizador comum e recolha de dados; e uma *Web* para fins administrativos e leitura de dados, dados esses armazenados numa base de dados comum às duas aplicações. Na Figura 29, pode visualizar-se uma representação da estrutura global do sistema, em que se pretende implementar uma "API" que é utilizada pelos clientes *Android,* de modo a analisar a informação recolhida, classificando-a e, posteriormente, registando-a na base de dados, que é, por fim, utilizada pela aplicação *Web* para visualização dos dados recolhidos.

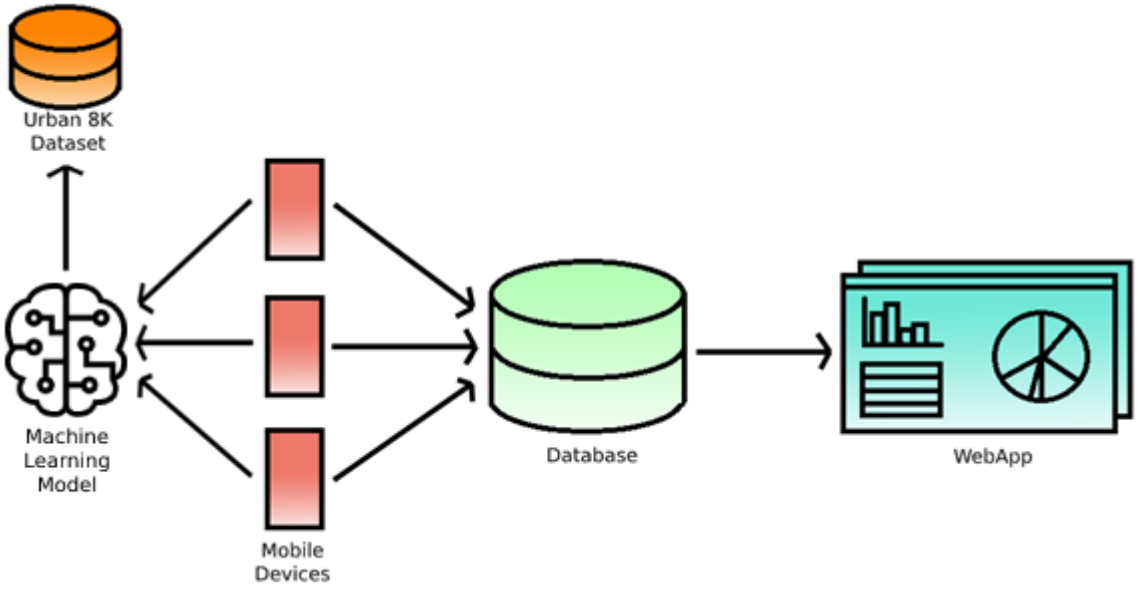


Figura 29 - Representação Global do Sistema

5.1. Análise de Requisitos e Estrutura do Sistema

Foi efetuada uma análise de requisitos para o sistema desenvolvido, com objetivo de não só determinar as funcionalidades a implementar, como também que parte do sistema teria a responsabilidade de as possuir. Na Tabela 10, são apresentadas de forma sucinta as tarefas a desempenhar pelo sistema de forma sequencial, da primeira ação a ser desempenhada até à última, de modo a obter uma iteração completa da análise de uma amostra de som.

Tabela 10 - Apresentação das funções do Sistema

Função	Componente do Sistema
Recolher uma amostra de som.	Aplicação *Android*
Processamento da amostra áudio de modo a classificá-la numa de 10 classes diferentes de sons urbanos.	Aplicação *Android*
Gerar informação acerca da localização via coordenadas GPS, e horário do incidente detetado, caso tenha sido obtida uma classificação válida para o mesmo.	Aplicação *Android*
Registo do Incidente na base de dados remota.	Aplicação *Android*
Local onde os vários incidentes criados pelas Aplicações *Android* ficam armazenados.	Base de Dados Remota
Visualização dos dados em função do tipo, quantidade, regularidade e localização dos mesmos, em forma de gráficos e de um mapa.	Aplicação *Web*.

Aplicação *Android* – Tem como principal responsabilidade a captação e respetiva classificação de ruídos registando-os sob a forma de **incidentes**, que são constituídos pela classificação atribuída, as coordenadas GPS, e a hora de criação. Um incidente é criado após a análise bem-sucedida de uma amostra de áudio, a qual é eliminada assim que é obtida a classificação. O incidente é gerado e registado na base de dados remota através de um POST feito por HTTP para um ficheiro PHP, que serve de *proxy* entre a aplicação e a base de dados.

Base de Dados Remota – Estrutura responsável por armazenar a informação dos incidentes e servir de ponte de comunicação entre as duas aplicações.

Aplicação *Web* – Responsável pela visualização dos dados, com representações de fácil leitura e interpretação pelos utilizadores do sistema. As leituras são feitas em função da quantidade global e parcial da informação presente na *pool* de dados, da altura dos registo, e da classificação atribuída. Os mesmos estão apresentados em três gráficos: um de barras, um circular, e um de funções; um "*top*" de incidentes mais registados, e um mapa carregado através dos recursos da *Google Cloud,* para visualização da localização de cada incidente.

5.2. Base de Dados (Audio Saver)

A base de dados utilizada no sistema é a componente do mesmo responsável não só por armazenar as informações dos incidentes bem como assumir o papel de "conector" entre as duas aplicações: *Android* e *Web*. A base de dados foi desenvolvida em MySQL, com um modelo relacional final (ver Figura 30) constituído apenas por duas tabelas: "Incident", e "UserA", sem qualquer ligação entre as mesmas. A segunda tabela apenas tem a função de guardar as contas administrativas da plataforma *Web*. As comunicações dos clientes *Android* com a base de dados são feitas pelo protocolo HTTP e com o auxílio de *proxies* PHP, presentes do lado do servidor, e a página *Web*, localizada no servidor.

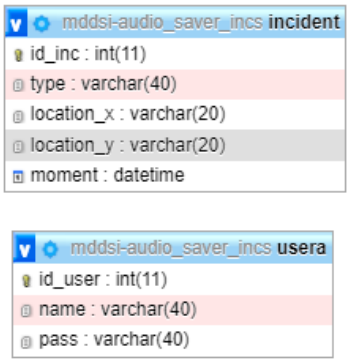


Figura 30 - Modelo Relacional (Base de Dados)

5.3. *Web App* (Audio Meter)

Como já foi referido, o propósito desta aplicação é conseguir visualizar os incidentes presentes na base de dados, de modo conclusivo e com uma forte componente visual. Com o objetivo de evitar tabelas e listas ou blocos enormes de texto, foram projetadas para esta ferramenta funcionalidades que se suportam de elementos visuais como gráficos e um mapa de incidentes, fazendo uso dos recursos da Google. O *template* utilizado para esta página *Web* foi inspirado e adaptado da página de *web designing* (Toolplate, s.d.), que possui uma enorme variedade de opções para vários ambientes e áreas profissionais, com opções gratuitas e pagas. De seguida, irão ser apresentadas as funcionalidades da aplicação, que se encontram identificadas no diagrama de casos de uso que se pode observar na Figura 31.

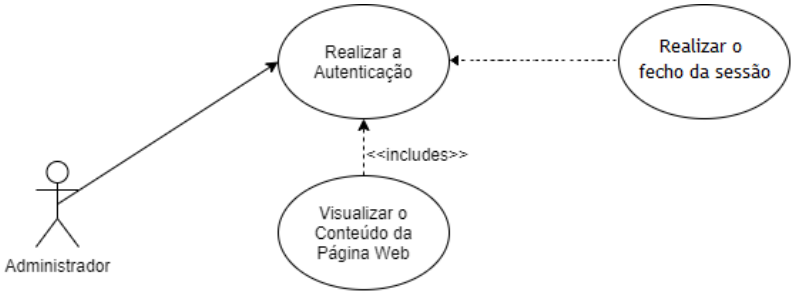


Figura 31 - Diagrama de Casos da *Web App*

Descrição do caso de uso "Realizar a Autenticação":

- Ator primário: *administrador*;
- *StakeHolders*: *administrador*;
- Precondições: Criação do *administrador* na base de dados;
- Garantia mínima: Erro na Autenticação;
- Garantia de Sucesso: Autenticação bem-sucedida;
- *Triggers*: Botão "*login*";
- Cenário Principal de sucesso:
    - O *administrador* insere o seu *username*;
    - O *administrador* insere a sua *password*;
    - O *administrador* prime "*login*";
    - O *administrador* é autenticado com sucesso;

Existe uma função de autenticação (ver Figura 32) nesta aplicação de modo a proteger os dados e limitar os mesmos apenas a quem possuir um perfil de administrador no sistema.

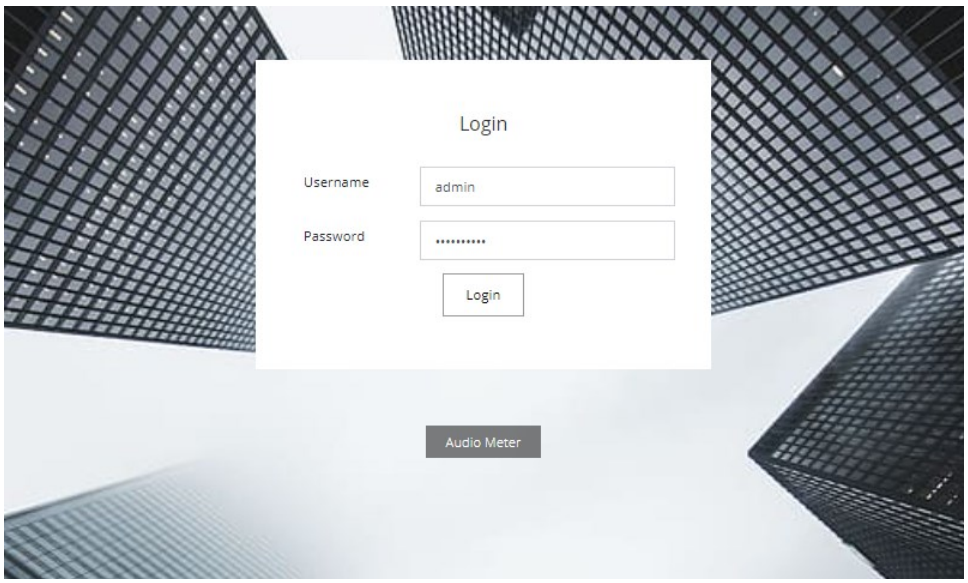


Figura 32 - Aplicação *Web* - Autenticação

Descrição do caso de uso "Realizar o fecho da sessão":

- Ator primário: *administrador*;
- *StakeHolders*: *administrador*;
- Precondições: Estar autenticado no sistema;
- Garantia mínima: Garantia mínima e de sucesso iguais;
- Garantia de Sucesso: Anulação da autenticação bem-sucedida;
- *Triggers*: Botão "*logout*";
- Cenário Principal de sucesso:
    - O *administrador* prime "*logout*";
    - O *administrador* tem a sua autenticação anulada com sucesso;

Da mesma forma que existe uma autenticação nesta aplicação, também existe o fecho da mesma (Figura 33). Após a utilização da plataforma, o utilizador deve efetuar o encerramento a sua sessão no sistema.

Figura 33 - Aplicação *Web* - Fecho da sessão

Descrição do caso de uso "Visualizar o conteúdo da página *Web*":

- Ator primário: *administrador*;
- *StakeHolders*: *administrador*;
- Precondições: Estar autenticado no sistema;
- Garantia mínima: Erro de carregamento da página;
- Garantia de Sucesso: visualização dos gráficos e mapas da página;
- *Triggers*: Carregamento da página;
- Cenário Principal de sucesso:
    - O *administrador* consegue ver os gráficos e interagir com os mesmos;

Os requisitos definidos para este componente do sistema foram a visualização de dados em função de alguns parâmetros básicos, presentes nos campos de um incidente: localização GPS, hora e data de criação do incidente e tipo de incidente detetado. Chegou-se então a quatro divisões importantes, em que são apresentados os dados relativos aos incidentes de formas diferentes e de modo a relacionar estes três fatores, sendo eles: a visualização do Histórico de Dados num Ano (Gráfico de Funções), Conteúdo Total na Base de Dados (Gráfico de Barras) e Distribuição por Quantidade de Incidentes (Gráfico Circular).

Todos os anteriores gráficos podem ser visualizados na mesma página, sendo ela a única e central página da aplicação *Web* (para além do ecrã de *login*), bem como o mapa que será descrito de seguida. Esta página está composta por duas secções: uma delas contendo os gráficos, cada um numa célula isolada, e outra contendo o mapa dos incidentes, conforme na Figura 34.

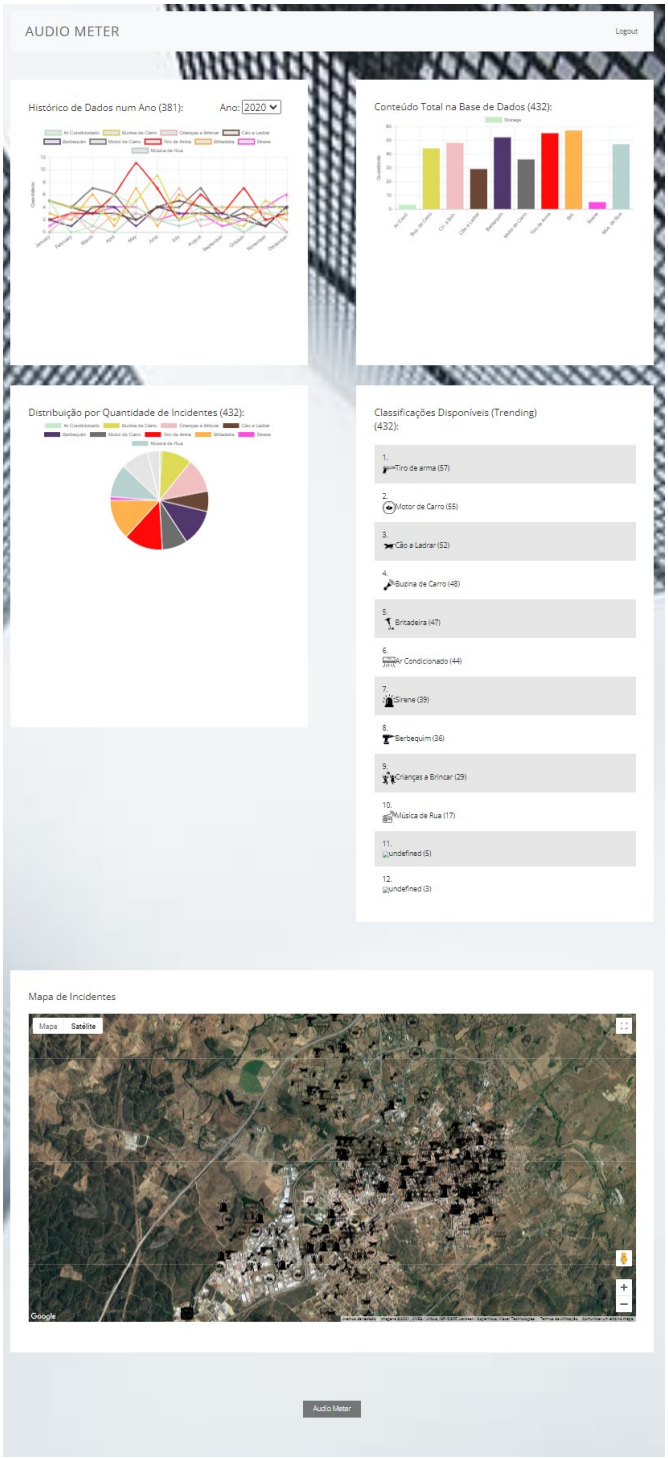


Figura 34 - Organização da Aplicação *Web*

Histórico de Dados num Ano (Gráfico de Funções)

Neste gráfico, é possível visualizar a quantidade de incidentes detetados ao longo de um ano (por mês), separados por tipos de incidente. É possível filtrar/escolher o ano que se pretende observar, bem como habilitar e desabilitar a visibilidade de um dado tipo de incidente no gráfico em questão. Este gráfico dá profundidade temporal aos dados presentes, sendo possível analisar os mesmos a um nível temporal, conforme na Figura 35:

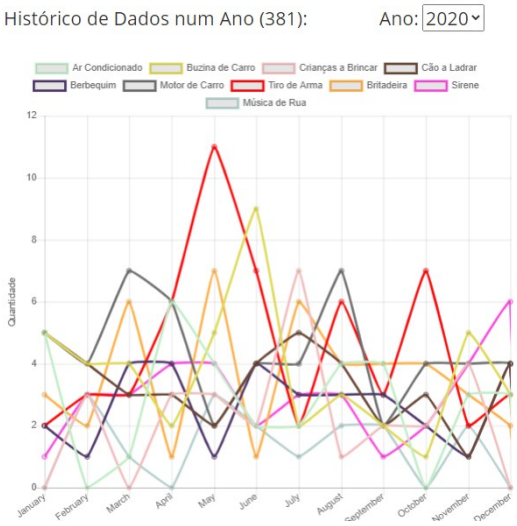


Figura 35 - Aplicação *Web* - Gráfico de Funções

Conteúdo Total na Base de Dados (Gráfico de Barras):

Neste gráfico, é possível visualizar o conteúdo total da base de dados na forma de um gráfico de barras, numa relação de quantidade total por tipo de incidente existente no sistema (Figura 36). O foco aqui é o volume de dados e discrepância entre as classificações possíveis, visualizando facilmente os dados mais representados:

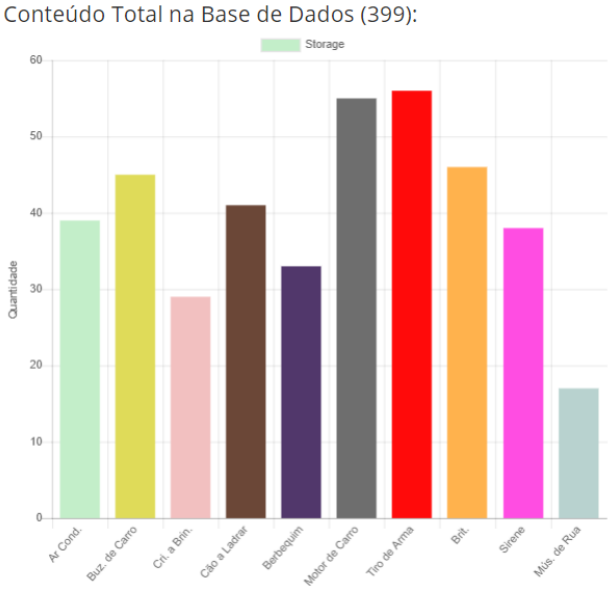


Figura 36 - Aplicação *Web* - Gráfico de Barras

Distribuição por Quantidade de Incidentes (Gráfico Circular):

Na Figura 37, representa-se a mesma relação de dados, no entanto num formato diferente. Neste caso, num gráfico circular, que tem foco na percentagem manifestada numa "fatia" ou "porção" para cada possível classificação, podendo assim assumir-se o volume que um dado tipo de incidente ocupa na base de dados em relação a outros:

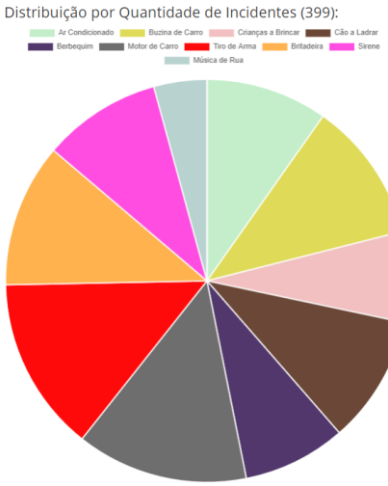


Figura 37 - Aplicação *Web* - Gráfico Circular

Mapa de Incidentes

Através de uma API fornecida pelos serviços *cloud* da (Google, s.d.), e com recurso às bibliotecas de mapeamento GPS Javascript, foi embutido um mapa interativo na aplicação que tem representados todos os incidentes criados na base dados, mapeados no local no qual foram detetados (Figura 38 e Figura 39). Desta forma, pode-se perceber as áreas da cidade que estão a ser monitorizadas e em que zonas existe mais volume de dados. Cada marcador tem a forma de um ícone respetivo à classificação desse mesmo incidente:

Figura 38 - Aplicação *Web* - Mapa de Incidentes

Figura 39 - Aplicação *Web* - Mapa de Incidentes com *zoom*

5.4. Aplicação (Audio Catcher)

A aplicação móvel tem como foco a captação, classificação e criação de incidentes para o sistema. A classificação é conseguida à custa do modelo de ML desenvolvido e apresentado no capítulo 4. Pode-se visualizar as funções possíveis da aplicação *Android* na Figura 40.

Figura 40 - Diagrama de Casos da Aplicação *Android*

Descrição do caso de uso "Recolha e classificação de uma amostra de Som":

Caso de Uso:

- Ator primário: *utilizador*;
- *StakeHolders*: *utilizador*;
- Precondições: Ter uma ligação à internet operacional;
- Garantia mínima: Erro na leitura da amostra;
- Garantia de Sucesso: Leitura do som envolvente e classificação (ou não) do mesmo em uma das classificações urbanas existentes;
- *Triggers*: Clicar no botão com um ícone de microfone;
- Cenário Principal de sucesso:
    - O *utilizador* recebe feedback da classificação realizada;

Tendo apenas esta função, a mesma contém apenas uma opção, conforme se pode ver na Figura 41. Esta opção permite a captação de um excerto de áudio do meio ambiente, que é posteriormente gravado na pasta de *downloads* do dispositivo. A aplicação requer as permissões do utilizador de escrita e leitura de dados, localização e uso do microfone do *smartphone*.

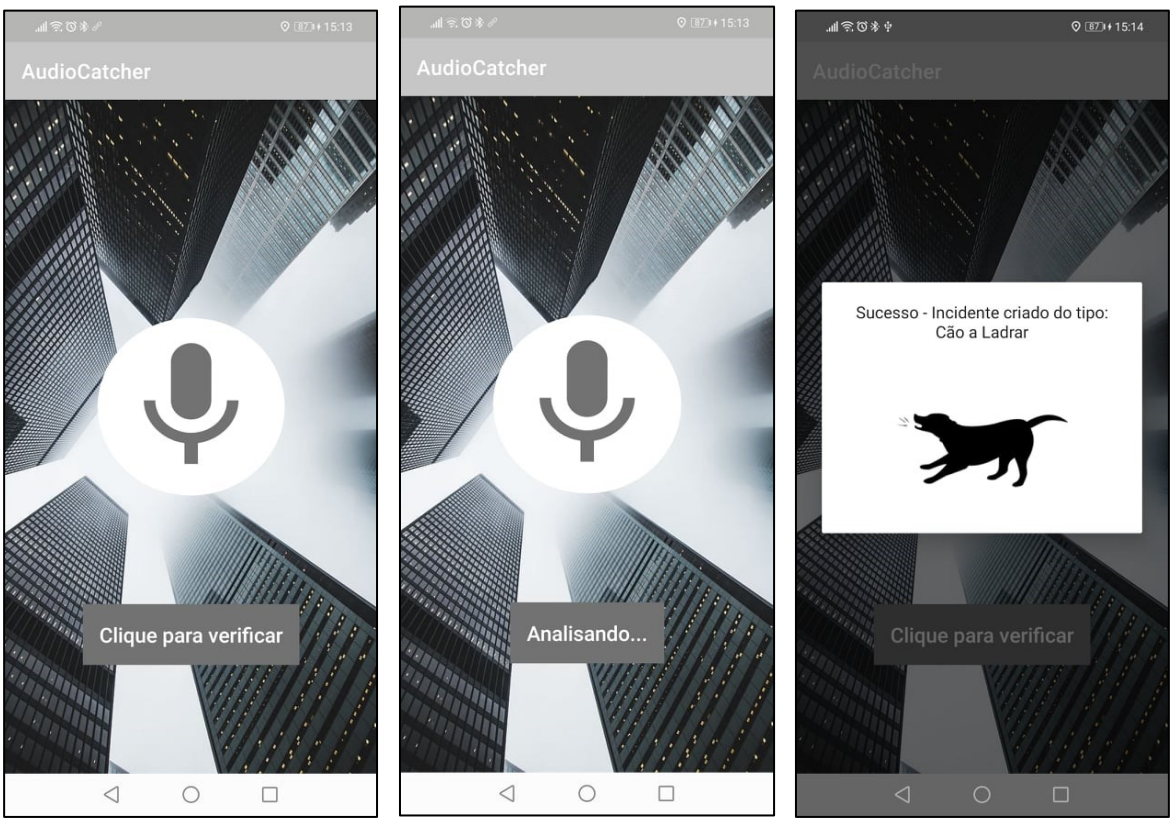


Figura 41 - Aplicação *Android* - Leitura de Amostra de Som

5.4.1. Integração do Modelo De ML

A integração do modelo obtido no capítulo 4 foi realizada com o suporte de uma ferramenta da biblioteca Tensorflow (Tensorflow, s.d.), de seu nome TFLite. A mesma consiste num modelo de formatação direcionada para uso de aplicações *Android*. Após o modelo ser carregado nos componentes da aplicação Android, ficando assim localmente alojado na mesma, foi necessário todo um processo de conversão do áudio recolhido para amostra pela aplicação, de modo que o mesmo pudesse ser inserido como valor *input* no modelo TFLite. Algumas da bibliotecas Java utilizadas neste processo foram: WavFile, WAVFileException, FFT e MFCC para tratamento dos ficheiros de áudio em formato .wav, e ainda Recognition, na ajuda ao tratamento das previsões obtidas. Na Figura 42, está esquematizado o processo de conversão realizado:

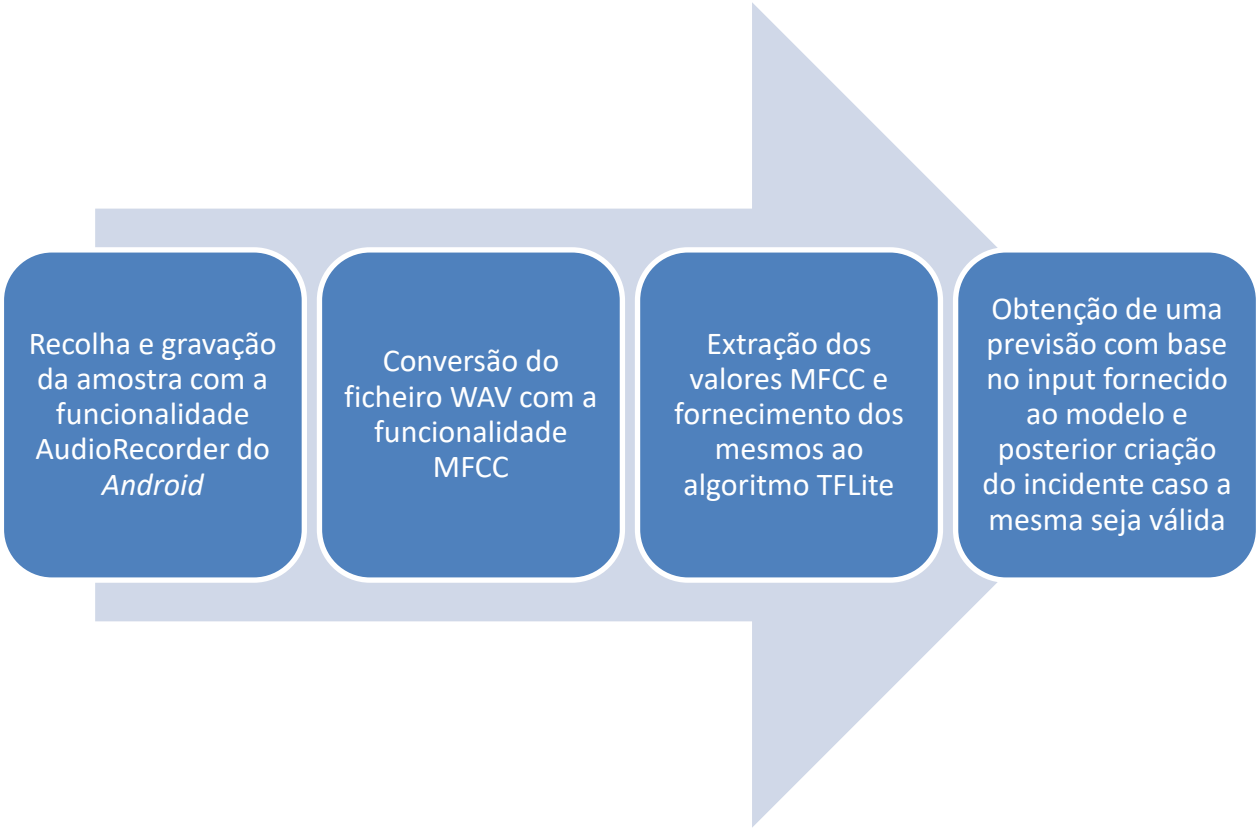


Figura 42 - Processo de Obtenção de uma Classificação na Aplicação Android

5.5. Testes do modelo na Aplicação *Android*

Nesta secção, apresentam-se os testes realizados com base em dados reais externos ao *dataset*. Foram usadas para testes 40 amostras de áudio (4 por cada uma das 10 categorias recolhidas pelo dispositivo), cada uma delas 3 vezes, dando um total de 120 amostras de áudio recolhidas e classificadas. Na Tabela 11, pode-se observar os valores obtidos pelo modelo num contexto real, e as percentagens finais de sucesso na classificação de cada ficheiro de áudio.

Tabela 11 – Validação do Modelo e Resultados Obtidos

Categoria	Número de classificações correctas	Tentativas Totais	Taxa de sucesso (%)	Média da taxa de sucesso (%)
air_conditioner	2	3	67	67
	2	3	67	
	1	3	33	
	3	3	100	
car_horn	3	3	100	92
	3	3	100	
	2	3	67	
	3	3	100	
children_playing	1	3	33	58
	2	3	67	
	1	3	33	
	3	3	100	
dog_bark	3	3	100	92
	3	3	100	
	2	3	67	
	3	3	100	
drilling	2	3	67	58
	2	3	67	
	2	3	67	
	1	3	33	
engine_idling	3	3	100	83
	3	3	100	

	2	3	67	
	2	3	67	
gun_shot	3	3	100	83
	3	3	100	
	1	3	33	
	3	3	100	
jackhammer	3	3	100	92
	3	3	100	
	2	3	67	
	3	3	100	
siren	2	3	67	92
	3	3	100	
	3	3	100	
	3	3	100	
street_music	1	3	33	58
	1	3	33	
	2	3	67	
	3	3	100	
TOTAIS	93	120	78	78

5.6. Discussão

O processo de recolha do áudio usado nesta fase teve como fonte sons externos aos que foram usados no treino do modelo. Os sons foram reproduzidos digitalmente por colunas e captados com o microfone do dispositivo que contém a aplicação *Android*, através do seu microfone nativo, numa habitação sem ruído envolvente. Noutras condições, evidentemente, os dados poderiam ter sido diferentes. Existiu alguma dificuldade em encontrar amostras adequadas para algumas categorias, como por exemplo *children_playing*, *engine_idling* e *street_music*, devido à variedade e abrangência de diferentes sons consideravelmente diferentes que podem ser reproduzidos por estas categorias. Com os dados obtidos pode-se constatar que as categorias com maior taxa de sucesso são as *car_horn, dog_bark, jackhammer* e *siren,* com 92%. Nota-se, portanto, que o modelo é mais recetivo a sons "estridentes" e menos eficaz em sons irregulares como *children_playing e street_music*.

6. Conclusão e Trabalhos Futuros

Neste capítulo, é feita uma conclusão ao trabalho desenvolvido bem como ao trabalho futuro que se considera ser relevante a desenvolver no âmbito do mesmo.

6.1. Conclusão

Com a conclusão deste trabalho, podemos dizer que foi possível desenvolver um sistema com potencial utilidade no ramo da segurança urbana, usando técnicas de inteligência artificial, mais concretamente na área do reconhecimento e classificação de áudio. Os resultados obtidos nos testes da aplicação, que se refletem numa percentagem de acerto total de 78%, dependeram de fatores como o equipamento utilizado na recolha dos dados, e podiam certamente ter sido obtidos melhores valores com outras condições. A integração do modelo de ML no sistema desenvolvido foi o maior desafio de todo o projeto, que é em parte o ponto central do trabalho desenvolvido. A área de programação que cobre o tratamento e conversão de áudio em Java é muito complexa, e teve grande importância na fase de integração do modelo na aplicação *Android*. Certamente, poderia haver pontos a melhorar nesta etapa do trabalho, que poderiam também trazer melhorias aos resultados obtidos.

6.2. Trabalho Futuro

Como em alguns dos estudos contidos no capítulo de trabalho realizado, um complemento a este projeto seria a implementação e testes de outras alternativas no que diz respeito a algoritmos de conversão de áudio, e outros tipos de redes neuronais como, por exemplo, as recorrentes, e cruzamento de todos os resultados obtidos. O trabalho futuro no âmbito do sistema desenvolvido nesta tese seria a sua integração num sistema já existente de segurança. Existirão possíveis dificuldades de compatibilidade de *hardware* e integração na infraestrutura da seguradora, trabalho esse que é importante mencionar que já se estenderia a outra área de também grande interesse e promissora, que seria a IoT (*Internet of Things*), pois todas as câmaras de vigilância e dispositivos que fizessem parte deste sistema necessitariam de sensores ou de um mini sistema computacional incorporados nos mesmos, e uma conexão permanente à Internet de modo a ser possível o registo dos incidentes em tempo real na base de dados.

Outro ponto importante a mencionar seria a necessidade de adaptar o armazenamento do áudio nos dispositivos em que o sistema estaria implementado. No atual ponto de situação, conforme mencionado na secção 5.4, as amostras de áudio ficam armazenadas na pasta de *downloads* do dispositivo. Para fins de testes académicos esta foi a solução mais prática, não comprometendo o desempenho da aplicação ou dispositivo. Noutro tipo de sistemas, uma adaptação teria de ser feita para que houvesse uma rotina constante de análise ao áudio captado, e após tentativa periódica de classificação ser realizada a eliminação do mesmo, de modo a não comprometer o armazenamento local do dispositivo.

7. Referências

Adavanne, S. & Virtanen, T., 2017. *A REPORT ON SOUND EVENT DETECTION WITH DIFFERENT BINAURAL FEATURES.* [Online]
Available at: https://arxiv.org/abs/1710.02997

Bauer, G., 2018. *Facility Executive.* [Online]
Available at: https://facilityexecutive.com/2018/08/surveillance-video-audio-equals-enhanced-security/
[Acedido em 13 12 2020].

Bello, J. P., Mydlarz, C. & Salamon, J., 2017. *Sound Analysis in Smart Cities.* [Online]
Available at: https://link.springer.com/chapter/10.1007/978-3-319-63450-0_13
[Acedido em 29 09 2020].

Brownlee, J., 2019. *How to Choose Loss Functions When Training Deep Learning Neural Networks.* [Online]
Available at: https://machinelearningmastery.com/how-to-choose-loss-functions-when-training-deep-learning-neural-networks/
[Acedido em 15 02 2021].

Brownlee, J., s.d. *A Gentle Introduction to the Rectified Linear Unit (ReLU).* [Online]
Available at: https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/

Castra, 2012. *Castra Better Security.* [Online]
Available at: https://www.castra.org.uk/
[Acedido em 13 12 2020].

Choi, K., Fazekas, G., Sandler, M. & Cho, K., 2018. *A COMPARISON OF AUDIO SIGNAL PREPROCESSING METHODS FOR DEEP NEURAL NETWORKS ON MUSIC TAGGING.* [Online]
Available at: https://ieeexplore.ieee.org/abstract/document/8553106

Chollet, F., 2018. *Deep Learning with Python.* 1ª ed. Shelter Island, NY 11964: Manning Publications Co..

Christophe, 2015. *Coding Geek - How does Shazam work.* [Online]
Available at: http://coding-geek.com/how-shazam-works/
[Acedido em 16 02 2021].

Dubey, A. et al., 2016. *Deep Learning the City : Quantifying Urban Perception At A Global Scale.* [Online]
Available at: https://arxiv.org/abs/1608.01769
[Acedido em 26 09 2020].

Flammini, F., Pappalardo, A. & Vittorini, V., 2013. *Research Gate.* [Online]
Available at: https://www.researchgate.net/figure/Audio-surveillance-pros-and-

cons_fig3_236270336
[Acedido em 13 12 2020].

Fridman, L., 2019. *MIT Deep Learning Basics: Introduction and Overview with TensorFlow.* [Online]
Available at: https://medium.com/tensorflow/mit-deep-learning-basics-introduction-and-overview-with-tensorflow-355bcd26baf0
[Acedido em 26 12 2020].

Géron, A., 2019. *Hands on Machine Learning with Scikit Learn Keras and TensorFlow 2nd Edition-2019.* 2º ed. United States of America: O'Reilly Media.

Google, s.d. *Maps JavaScript API.* [Online]
Available at: https://console.cloud.google.com/marketplace/product/google/maps-backend.googleapis.com?q=search&referrer=search&project=audiometer

ins2outs, 2019. *ins2outs.* [Online]
Available at: https://ins2outs.com/implement-information-security-management-system/
[Acedido em 13 12 2020].

KHAMPARIA, A., GUPTA, D., NGUYEN, N. G. & KHANNA, A., 2019. *Sound Classification Using Convolutional Neural Network and Tensor Deep Stacking Network.* [Online]
Available at: https://ieeexplore.ieee.org/abstract/document/8605515

Li, A., 2019. *Reolink.* [Online]
Available at: https://reolink.com/pros-cons-of-surveillance-cameras-in-public-places/
[Acedido em 13 12 2020].

Li, J. et al., 2017. *A COMPARISON OF DEEP LEARNING METHODS FOR ENVIRONMENTAL SOUND.* [Online]
Available at: https://ieeexplore.ieee.org/abstract/document/7952131

Lun, L., Elisabete, S., Chunyang, W. & Hui, W., 2017. *A machine learning-based method for the large-scale evaluation of the qualities of the urban environment.* [Online]
Available at:
https://www.sciencedirect.com/science/article/pii/S0198971516301831
[Acedido em 09 2020].

Marcondes, J. S., 2016. *Gestão de Segurança Privada.* [Online]
Available at: https://gestaodesegurancaprivada.com.br/seguranca-eletronica-conceito/
[Acedido em 13 12 2020].

Mozur, P., 2018. *NY Times - Inside China's Dystopian Dreams: A.I., Shame and Lots of Cameras.* [Online]

Available at: https://www.nytimes.com/2018/07/08/business/china-surveillance-technology.html
[Acedido em 16 02 2021].

Okamura, E., Fujita, S. & Hino, K., 2018. *Hitachi - Total Urban Security Solution Using Integrated Monitoring System.* [Online]
Available at: https://www.hitachi.com/rev/archive/2018/r2018_04/02b01/index.html
[Acedido em 07 12 2020].

Park, Y.-J. & Cho, H.-S., 2020. *An Experiment of Sound Recognition using Machine Learning.* [Online]
Available at: https://ieeexplore.ieee.org/abstract/document/9277368

Shazam, 2021. *Shazam.* [Online]
Available at: https://play.google.com/store/apps/details?id=com.shazam.android&hl=pt_PT&gl=US
[Acedido em 16 02 2021].

Shen, X., 2018. *"Skynet", China's massive video surveillance network.* [Online]
Available at: https://www.scmp.com/abacus/who-what/what/article/3028246/skynet-chinas-massive-video-surveillance-network
[Acedido em 29 09 2020].

Smith, M., 2017. *Skynet in China: Real-life 'Person of Interest' spying in real time.* [Online]
Available at: https://www.csoonline.com/article/3228444/skynet-in-china-real-life-person-of-interest-spying-in-real-time.html
[Acedido em 16 2 2021].

Tensorflow, s.d. *Tensorflow.* [Online]
Available at: https://www.tensorflow.org/

TFLite, s.d. *TFlite.* [Online]
Available at: https://www.tensorflow.org/lite/examples/audio_classification/overview

Toolplate, s.d. *Toolplate.* [Online]
Available at: https://www.tooplate.com/

Torres, J., 2020. *Towards Data Science - Deep Learning Basics.* [Online]
Available at: https://towardsdatascience.com/deep-learning-basics-1d26923cc24a
[Acedido em 26 12 2020].

Trust, B., 28 May 2019. *Internet Society - Enhancing IoT Security: Final Outcomes and Recommendations Report.* [Online]
Available at: https://www.internetsociety.org/resources/doc/2019/enhancing-iot-security-final-outcomes-and-recommendations-report/?gclid=CjwKCAiAwrf-

BRA9EiwAUWwKXrtSDuIKbXVPePnx7YT_MM_x0hgJsCBe98OHPhlEhbgHSBHK9Srs2xoCIk8QAvD_BwE
[Acedido em 07 12 2020].

Urban8K, s.d. *URBANSOUND8K DATASET.* [Online]
Available at: https://urbansounddataset.weebly.com/urbansound8k.html

Yu, Z., 2017. *Facial recognition, AI and big data poised to boost Chinese public safety.* [Online]
Available at: https://www.globaltimes.cn/content/1070546.shtml
[Acedido em 16 02 2021].