Politécnico de Leiria
Escola Superior de Tecnologia e Gestão
Departamento de Engenharia Eletrotécnica
Mestrado em Engenharia Eletrotécnica – Eletrónica e Telecomunicações

DISSERTAÇÃO

CLASSIFICAÇÃO DE CENAS ACÚSTICAS EM DISPOSITIVOS COM CONSTRANGIMENTOS COMPUTACIONAIS UTILIZANDO INTELIGÊNCIA ARTIFICIAL

RICARDO RIBEIRO ANASTÁCIO

Leiria, Setembro 2022

Esta página foi propositadamente deixada em branco.

Politécnico de Leiria
Escola Superior de Tecnologia e Gestão
Departamento de Engenharia Eletrotécnica
Mestrado em Engenharia Eletrotécnica – Eletrónica e Telecomunicações

DISSERTAÇÃO

CLASSIFICAÇÃO DE CENAS ACÚSTICAS EM DISPOSITIVOS COM CONSTRANGIMENTOS COMPUTACIONAIS UTILIZANDO INTELIGÊNCIA ARTIFICIAL

RICARDO RIBEIRO ANASTÁCIO
Número: 2192601

Dissertação realizada sob a orientação do Professor Doutor Luís Manuel Conde Bento (luis.conde@ipleiria.pt) e Professora Doutora Mónica Jorge Carvalho de Figueiredo (monica.figueiredo@ipleiria.pt).

Leiria, Setembro 2022

Esta página foi propositadamente deixada em branco.

AGRADECIMENTOS

Gostaria de começar por agradecer aos meus orientadores Professor Doutor Luís Manuel Conde Bento e Professora Doutora Mónica Jorge Carvalho de Figueiredo pela orientação, apoio e prontidão, fatores fundamentais na elaboração desta pesquisa. Contribuíram partilhando o seu conhecimento, esclarecendo dúvidas e fornecendo os recursos necessários para o desenvolvimento deste projeto, contribuindo para o sucesso do trabalho.

Agradeço também ao aluno Luís Ferreira estudante de Mestrado em Engenharia Eletrotécnica da Universidade de Coimbra, pela ajuda e partilha de conhecimento durante a participação no concurso DCASE-2022.

Por fim, gostaria de agradecer à minha família todo suporte, apoio e incentivo durante este percurso.

Esta página foi propositadamente deixada em branco.

RESUMO

Ao longo desta dissertação é apresentada a pesquisa e desenvolvimento de um modelo neuronal de arquitetura *Convolutional Neural Network* (CNN) para classificar dez ambientes acústicos distintos, destinado a dispositivos baixa complexidade, *i.e.* limite 128K em número de parâmetros e máximo de 30 MMAC (milhões de *Multiply-ACcumulate* (MAC)) por inferência.

É utilizada a pesquisa de hiperparâmetros (*hypertuning*), para conseguir retirar o máximo de informação pertinente dos recursos de áudio e também para otimizar a arquitetura do modelo.

São utilizadas técnicas de aumento de dados, suavização de rótulos e paragem de treino antecipada para melhorar a generalização do modelo, melhorando a resposta na presença de novos dados não utilizados para treino.

São propostas ainda três abordagens com o objetivo de aumentar o campo de aprendizagem e melhorar a diferenciação entre classes. Estas abordagens combinam métodos de otimização e aprendizagem, como agrupamento de modelos *ensemble*, separação de classes ou aprendizagem um contra todos (OvA). Foi ainda aplicada a técnica de destilação de conhecimento (KD), que permitiu reduzir a sua complexidade do modelo, esta técnica acabou por funcionar também como regularizador diminuindo o sobreajuste.

As abordagens propostas foram validadas através da participação no desafio "Task 1, Low-Complexity Acoustic Scene Classification 2022" proposto pela comunidade internacional DCASE, conseguindo obter o 4.º lugar na classificação de equipas num universo de 19 equipas, e 11.º lugar perante 48 modelos em avaliação.

O modelo submetido que obteve melhores resultados é designado AI4EDGE_4 é um *ensemble* de dez modelos OvA, utilizado como "professor" num processo de destilação de conhecimento num "aluno" de arquitetura TBM2. O modelo base fornecido em DCASE obteve uma exatidão de ACC=44.2% e uma perda de LOSS=1.532, o modelo AI4EDGE_4 obteve melhor desempenho *i.e.* uma exatidão de AC=51.6% e uma perda de LOSS=1.330.

Palavras-chave: Rede neuronal artificial; Classificação de cenas acústicas; Dispositivos de ponta; Rede neuronal convolucional; Pesquisa de hiperparâmetros; Redes neuronais conjuntas (ensemble); Destilação de conhecimento.

Esta página foi propositadamente deixada em branco.

ABSTRACT

The identification and classification of acoustic scenes enables the development of new applications, *e.g.* triggering numerous actions inline with the user surroundings. However, the classification of acoustic scenes presents a huge complexity challenge, since in many instances, these scenes are composed by various events that can overlap each other. Typically, acoustic scenes classification applications, are developed aiming its integration in high mobility devices. It is essential to develop efficient and low-complexity algorithms, due to the small computational "power" of these devices, *i.e.* 128K for the maximum number of parameters and 30 MMAC (millions MAC) for the maximum number of MACS per inference.

This thesis presents the development of an low-complexity CNN able to classify 10 acoustic scenes. Aiming to optimise not only the model architecture, but also to maximise relevant information of the audio resources, the proposed CNN model is obtained though the use of hypertuning. To decrease the model loss when presented with new data, techniques such has data augmentation, early stopping and soft labelling where employed. Three neural model architecture/techniques are proposed to improve the network specificity. The first method consisted on models ensemble, the second on the class separation by grouping the most distinct classes, and the third on the One-vs-All (OvA) method. It was also applied a Knowledge Distillation (KD) method, resulting ina a reduction of the model complexity and consequently the reduction of the network overfiting.

The proposed approaches were validated through the participation on the challenge "Task 1, Low-Complexity Acoustic Scene Classification 2022", promoted by the internacional DCASE community, achieving the 4^{th} place in 19 participating teams. Out of 48 models evaluated by the competition the best proposed model described in this thesis achieved the 11^{th} place. The best proposed model (AI4EDGE_4) is composed by an *ensemble* of ten OvA models, this model is used as the "teacher" during the knowledge distillation process in a "student". The baseline model provided by the DCASE attained an accuracy of 44.2% and a loss of 1.532, while the developed model AI4EDGE_4, achieved an accuracy of 51.6% and a loss of 1.330.

Keywords: Artificial neural network; Acoustic scene classification; TinyML; CNN; Hypertuning; Ensemble; Knowledge distillation;

Esta página foi propositadamente deixada em branco.

ÍNDICE

LISTA DE FIGURAS

LISTA DE TABELAS

Esta página foi propositadamente deixada em branco.

LISTA DE ACRÓNIMOS

AEC	*Acoustic Event Classification.*
AED	*Acoustic Event Detection.*
AI	*Artificial Intelligence.*
ASC	*Acoustic Scene Classification.*
ASD	*Anomalous Sound Detection.*
Cloud	*Cloud Computing.*
CloudML	*Cloud Machine Learning.*
CNN	*Convolutional Neural Network.*
DCASE	*Detection and Classification of Acoustic Scenes and Events.*
DL	*Deep Learning.*
EdgeML	*Edge Machine Learning.*
FFT	*Fast Fourier transform.*
FPM	*Feature Pyramid Module.*
FS	Sampling Rate.
IoT	*Internet of Things.*
KD	*Knowledge Distillation.*
LTH	*Lottery Ticket Hypothesis.*
MAC	*Multiply-ACcumulate.*
MCU	*Microcontroller Unit.*
MFA	*Multi-scale Frequency-channel Attention.*
ML	*Machine Learning.*
OvA	*One-vs-All.*
OvO	*One-vs-One.*
QAT	*Quantization Aware Training.*
ResNet	*Residual neural Network.*
SHA	*Successive Halving Algorithm.*
SRAM	*Static Random Access Memory.*
STFT	*Short Time Fourier Transform.*

TinyML *Tiny Machine Learning.*

1

INTRODUÇÃO

1.1 ENQUADRAMENTO

Ao longo das últimas décadas tem vindo a aumentar o interesse em emular processos de inteligência humana em máquinas e sistemas computorizados *Artificial Intelligence (AI)* [1]. É desejado por muitos investigadores que as máquinas sejam autónomas e consigam desempenhar as suas funções com base em decisões tomadas de forma automática, de preferência sem que tenham sido programadas especificamente para tal (*Machine Learning* (ML)). Por outro lado, o aumento do número de dados e sensores fez disparar o interesse em resolver problemas cada vez mais complexos levando à procura pelo aumento do poder de computação, aspetos considerados fundamentais e impulsionadores da evolução de algoritmos e aplicações.

O surgimento de novos desafios cada vez mais complexos e com maior pressão por parte da indústria, necessitam de soluções também mais complexas e naturalmente de desenvolvimento mais demorado. Estes aspetos dificultam a escalabilidade e acompanhamento dos algoritmos tradicionais ML existentes até então, tornando-os gradualmente menos úteis em problemas de elevada complexidade. Estes desafios impulsionam o interesse em algoritmos ML modernos e progressivamente mais parecidos com o raciocínio humano, apresentando novos algoritmos e soluções *Deep Learning* (DL).

A crescente complexidade dos desafios exigiu também um aumento da capacidade de processamento e armazenamento de dados. Esta tecnologia esta atualmente disponível na sua maioria através de servidores em rede com acesso à Internet (*Cloud Computing* (Cloud)). A utilização de ML em processamento Cloud designa-se *Cloud Machine Learning* (CloudML).

A proliferação de equipamentos móveis e a procura por novos serviços e aplicações totalmente conectadas (*Internet of Things* (IoT)), exige um amplo acesso a redes de comunicação, para distribuir todo o processamento e armazenamento ao longo de plataformas de alto desempenho. Embora o processamento seja amplamente distribuído em rede existem alguns aspetos críticos consoante a finalidade da aplicação, como o atraso de comunicação (latência), o aumento de custos energéticos, problemas de privacidade e segurança, mobilidade e ocupação desnecessária de tráfego. Estes problemas podem provocar atrasos, falhas e eventual colapso da

aplicação ou serviço, adicionalmente os serviços de Cloud são também geralmente dispendiosos, encarecendo as aplicações e produtos.

Muitas das aplicações IoT não necessitam de sobrecarregar a rede de telecomunicações para desempenhar as suas funções, caso tenham a possibilidade de realizar o processamento ou parte dele nas imediações do ponto de aplicação (*Edge*), ou seja, próximo da fonte de dados origem [2, 3]. A implementação na *Edge* de algoritmos ML designa-se *Edge Machine Learning (EdgeML)* e permite aproveitar os recursos de computação com maior proximidade do utilizador final, resultando soluções mais eficientes e tempos de resposta mais curtos [4].

A maioria dos dispositivos *Microcontroller Unit* (MCU) é atualmente utilizada para transmitir e receber dados, embora saibamos que os equipamentos MCU têm capacidade de processamento e armazenamento incluída, ainda que de forma reduzida [5]. A implementação de redes neuronais e algoritmos ML em dispositivos de capacidade diminuída (MCU) designa-se *Tiny Machine Learning* (TinyML). Existem no mundo mais de 250 mil milhões de MCUs, só em 2018 foram vendidos cerca de 28 mil milhões de unidades. Entre 2022-2030 prevê-se um crescimento anual na ordem dos 13% [6–8]. Os MCUs são largamente utilizados em muitas das aplicações IoT que conhecemos funcionando essencialmente para recolha de dados e atuação equipamentos.

O paradigma TinyML encontra-se numa fase ainda embrionária, no entanto, segundo pesquisas recentes, é considerado um passo essencial para uma utilização inteligente e eficiente das estruturas IoT disponíveis. Permitindo combater os principais problemas de aplicações IoT quando totalmente dependentes de serviços processados em Cloud. Colocando os processos de decisão junto da origem, o sistema beneficia de aspetos como, custo reduzido, baixo consumo energético, escalabilidade, flexibilidade de reprogramação, autonomia, privacidade, segurança e menos constrangimentos típicos de serviços Cloud [9, 10].

1.2 ÂMBITO E OBJETIVOS

A inferência TinyML apresenta diversos obstáculos e desafios que estão amplamente relacionados com as limitações impostas pelos MCUs, tornando a implementação de algoritmos ML um processo bastante engenhoso e criativo. Uma rede neuronal destinada a ser implementada em MCU deve ser suficientemente pequena para acomodar as restrições do dispositivo [10] e bastante eficiente para que com poucos recursos se consiga o máximo de desempenho. Muitas das aplicações TinyML atualmente implementadas em IoT realizam todo o processamento e armazenamento através de ligações à rede. São vários os casos de uso possíveis para aplicações

TinyML tais como cuidados de saúde inteligentes, condução autónoma, segurança pública, agricultura, aplicações de emergência, classificação e deteção de objetos, sons, padrões, entre outros.

A identificação e classificação de uma cena acústica permite desencadear uma série de ações enquadradas no contexto de determinado ambiente envolvente, possibilitando o desenvolvimento de inúmeras aplicações em prol do utilizador. O objetivo desta dissertação é investigar e desenvolver um algoritmo de classificação de cenas acústicas (em inglês, Acoustic Scene Classification (ASC)), capaz de classificar dez ambientes acústicos distintos *e.g.* aeroporto, centro comercial, viagem de metro, etc. São utilizados áudios de 1 segundo captados em 12 cidades europeias através de 4 dispositivos diferentes para cada uma das classes. O algoritmo deve ser desenvolvido com o objetivo de ser futuramente implementado num dispositivo MCU, como tal devem ser consideradas as limitações de complexidade típicas destes equipamentos. É necessário obter um modelo de baixa complexidade computacional e simultaneamente com bom desempenho. Pretende-se ainda que o sistema, para além de boa capacidade de aprendizagem, tenha capacidade de generalização, mantendo o desempenho mesmo na presença de novos dados. Os objetivos propostos para esta dissertação, estão em linha com o desafio apresentado na tarefa 1 da competição internacional *Detection and Classification of Acoustic Scenes and Events* (DCASE) 2022. A métrica utilizada pelo desafio para avaliação do desempenho dos modelos é *Log loss*. É necessário também respeitar as limitações de complexidade impostas, nomeadamente quantização a 8bits, limite de parâmetros 128k e um máximo de 30 milhões de operações *Multiply-ACcumulate* (MAC) por inferência.

1.3 FERRAMENTAS UTILIZADAS

O trabalho realizado ao longo da dissertação foi desenvolvido utilizando a linguagem de programação de alto nível *Python* versão 3.6 [11]. Foi utilizada a biblioteca *Librosa v0.8.1* [12] na análise e extração de características do áudio e espectrogramas de mel. A plataforma *Tensorflow v2.6.0* [13] que inclui a ferramenta *Keras v2.7.0* [14] permitiu realizar todo processo e desenvolvimento de algoritmos ML. Foi utilizada a biblioteca *Audiomentations v0.24.0* [15] para realizar o aumento de dados. A ferramenta *Kapre v0.3.7* possibilitou realizar o processamento de sinal em simultâneo com processo de treino, durante a pesquisa automática de hiperparâmetros. O modelo base fornecido pela competição [16] é acompanhado pela biblioteca *Dcase-util v0.2.18* [17] que facilita a manipulação e organização do conjunto de dados. Os algoritmos desenvolvidos foram treinados utilizando o *Tensorflow* no modo de processamento em unidade gráfica (GPU), utilizando a placa gráfica *NVIDIA GeForce RTX 3080*.

1.4 CONTRIBUIÇÕES

Este trabalho permitiu o desenvolvimento de uma rede de classificação de dez cenas acústicas destinada a dispositivos de baixa complexidade. Foi realizada a pesquisa e desenvolvimento de diversas abordagens, possibilitando obter melhorias significativas comparativamente ao modelo base fornecido, no desafio "Task 1, Low-Complexity Acoustic Scene Classification 2022" proposto pela comunidade DCASE.

As principais contribuições desta tese são:

- Ajuste dos recursos característicos obtidos pelo espetrograma de mel, através da realização de uma pesquisa de hiperparâmetros, possibilitando encontrar as especificações do áudio que favorecem a classificação e diminuir as amostras necessárias na representação do sinal. Esta pesquisa foi utilizada como base para o restante trabalho desenvolvido.
- É apresentado também uma versão melhorada do modelo *baseline* fornecido em DCASE, designado nesta tese como TBM2. Este modelo é resultado de uma otimização através da pesquisa de hiperparâmetros para os novos recursos de espectrograma. Este modelo é utilizado como base da pesquisa realizada ao longo desta dissertação, nomeadamente na implementação de outras abordagens.
- Para diminuir a *confusão* entre algumas classes, é proposta a separação em 2 grupos distintos. São agrupadas as classes mais facilmente confundíveis entre si e treinados dois modelos, um para cada grupo de classes, posteriormente os modelos são combinados através de *ensemble*.
- É desenvolvido um *ensemble* de dez modelos que realiza a separação de classes segundo o método "um contra todos", com o objetivo de aumentar a capacidade de aprendizagem e melhorar a distinção entre classes. É ainda utilizado a destilação de conhecimento que reduziu a complexidade do modelo e possibilitou a diminuição do *overfitting*.
- Com o objetivo de melhorar o desempenho e reduzir o *overfitting*, são fornecidos mais dados para treino. Os modelos utilizados como base para *ensemble* são treinados apenas com o conjunto de treino, após o treino os modelos são bloqueados para não perderem as características aprendidas. É inserida uma camada de classificação que combina os dois modelos sendo realizado um último treino com a totalidade do conjunto de dados. Assim é possível melhorar o desempenho do modelo e reduzir ligeiramente o *overfitting*.

1.5 ESTRUTURA DA DISSERTAÇÃO

A dissertação está organizada da seguinte forma:

O capítulo 2, contém a revisão da literatura efetuada, são identificadas características da implementação de ML em dispositivos MCU bem como benefícios e possíveis aplicações. É definida toda a envolvente de problemas acústicos e apresentadas algumas abordagens utilizadas recentemente na classificação de cenas acústicas. No Capítulo 3, são introduzidos os principais conceitos e técnicas que servem de suporte a realização desta dissertação.

O capítulo 4, apresenta as características relevantes da competição internacional DCASE 2022, conjunto de dados e modelo base.

No capítulo 5, é apresentado o trabalho realizado, abordagens e resultados experimentais obtidos.

Por fim, o capítulo 6 evidencia as principais conclusões do trabalho apresentado nesta dissertação, são propostas algumas experiências futuras em linha com o trabalho realizado.

Esta página foi propositadamente deixada em branco.

2

ESTADO DA ARTE

O presente capítulo, apresenta o estado da arte no que respeita à implementação de ML em dispositivos MCU, referindo algumas vantagens e desafios comparativamente à forma mais tradicional de implementação através de processamento em rede. São identificados alguns setores e possíveis aplicações onde a utilização de algoritmos ML em MCUs é especialmente interessante. Posteriormente é apresentada a área de aplicação em que se inserem os objetivos desta dissertação. Finalmente são apresentados alguns trabalhos relevantes na área específica da *Acoustic Scene Classification* (ASC) e as abordagens tipicamente utilizadas para o objetivo em questão, salientando elementos comuns e divergências.

2.1 IMPLEMENTAÇÃO ML EM DISPOSITIVOS MCU

Atualmente dispositivos MCU desempenham maioritariamente funções de recolha de informação sensorial e/ou acionamento de atuadores. O processamento e tomada de decisão é frequentemente realizado em servidores na Cloud, distantes do ponto de aplicação. Os dados são transmitidos para um local de armazenamento, depois são processados, novamente armazenados e retransmitidos para o dispositivo no *Edge*. Associado a um processamento amplamente distribuído em rede são observadas algumas limitações que podem ser relevantes consoante a aplicação, como atrasos de comunicação, aumento de custos energéticos, problemas de privacidade ou segurança, restrições de mobilidade e aumento do congestionamento da rede. Muitas das funções desempenhadas por dispositivos MCU não necessitam realmente de uma conexão com a rede para desempenhar a sua função ou parte dela. Neste caso, seria interessante realizar parte do processamento, ou mesmo a sua totalidade, próximo da fonte de dados [2,3], utilizando as capacidades de processamento e armazenamento presentes nos dispositivos. Este é o paradigma TinyML, (*vide* Fig. 2.1).

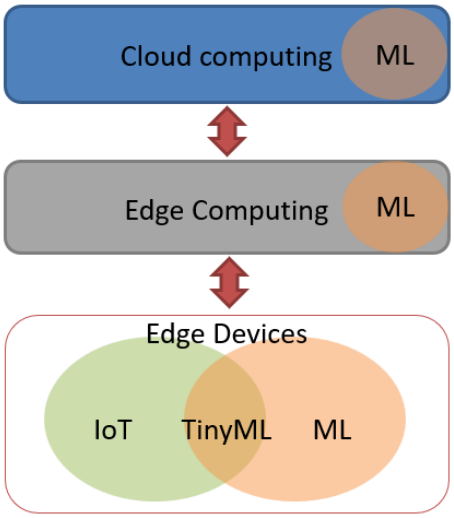


Figura 2.1: Paradigma TinyML.

Dispositivos TinyML possibilitam diversas vantagens tais como [9, 10]:

- Custo reduzido;
- Baixo consumo energético;
- Escalabilidade e flexibilidade de reprogramação;
- Autonomia;
- Privacidade e segurança;
- Mobilidade;
- Funcionamento em tempo real;

A implementação de uma rede neuronal num dispositivo pressupõem a alocação de alguns recursos. É necessário armazenar um grande número de constantes que configuram a estrutura do modelo, parte delas são determinadas durante o processo de treino do modelo sendo designadas de pesos. É ainda necessário alocar espaço de memória com o objetivo de ser utilizado para armazenar o resultado de cálculos intermédios realizados ao longo do processo de predição (inferência), estes resultados são também designados de ativações. Mais informação acerca de constituintes base de uma rede neuronal esta disponível na secção 3.2.1.

Uma rede neuronal destinada a ser implementada em MCU deve ser suficientemente pequena para acomodar as restrições do dispositivo, que apresenta memória *Static Random Access Memory* (SRAM) limitada (*e.g.* 128-512 kB) e de capacidade de armazenamento *flash* em torno dos (256kB-2MB) [10]. A memória SRAM é preferencialmente utilizada para o armazenamento de ativações de entrada e saída da rede neuronal para que a inferência seja rápida. Enquanto os pesos, estrutura do modelo, conexões e código de suporte são armazenados tipicamente na memória *flash*.

A introdução de algoritmos ML em MCUs é um conceito recente, prova disso é a desproporcionalidade que existe entre o número de dispositivos existentes

e a quantidade de recursos de implementação TinyML. A Fig. 2.2 representa a desproporcionalidade entre o número de dispositivos existentes para os vários géneros de processamento, em oposição à quantidade de recursos e informação ML disponível para cada tipo de dispositivo. Apesar de existirem milhões de vezes mais dispositivos MCU, são poucos os recursos disponíveis para auxiliar à implementação de algoritmos TinyML.

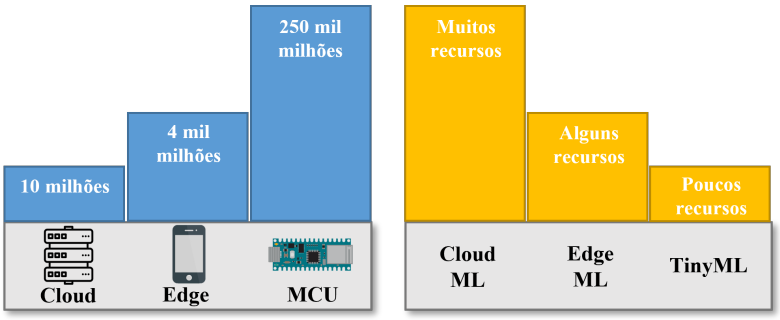


Figura 2.2: Proporção, recursos vs dispositivos ML [18].

2.2 ÂMBITO DE APLICAÇÃO TINYML

O conceito TinyML tem um enorme potencial e aplicação em diversos casos de uso. É visto como um complemento ao IoT contribuindo para resolver alguns dos seus problemas. A principal vantagem TinyML é o facto de não necessitar de processar a informação em rede através de servidores, utilizando apenas a rede para o estritamente necessário. Desta forma é possível conseguir inúmeras melhorias em aplicações IoT que conhecemos atualmente. A aplicação de algoritmos AI em dispositivos MCU permite aplicações bastante diversas e úteis em diversos setores de atividade, como sejam a saúde, o sector industrial, a eletrónica de consumo ou a agropecuária Fig. 2.3.

Na saúde o paradigma TinyML apresenta um grande potencial. A sua utilização em dispositivos *wearable* permite realizar a monitorização continua de sinais biológicos em tempo real, *e.g.* monitorizar e classificar um eletrocardiograma [19], detetar problemas respiratórios [20], detetar quedas [21], identificar movimentos e atividades desportivas [22]. Os cenários de aplicação são vastos e estendem-se a soluções complexas para auxiliar pessoas com problemas auditivos, físicos, problemas mentais e visuais, ou soluções simples como alerta para a toma de medicamentos.

O sector industrial usufrui também deste conceito. É possível analisar dados localmente e monitorizar sistemas industriais com restrições temporalmente críticas, evitando falhas e tempos de paragem prolongados (reduzindo naturalmente os custos associados). Neste contexto pode servir para detetar anomalias [23], calendarizar manutenções preditivas, controlar sistemas autónomos e robóticos, apoiar serviços

de localização e até na gestão e segurança de trabalhadores através de equipamentos *wearable*. Aplicações com base em sensores acústicos podem também ser utilizadas a nível industrial, medindo vibração e ultrassons para detetar falhas em máquinas elétricas, falhas de integridade estrutural e fadiga de materiais nas mais diversas estruturas (mesmo que estas ainda sejam microscópicas).

Num âmbito da eletrónica de consumo existe uma vasta gama de aplicações. A vida em sociedade pode ser amplamente simplificada e segura com o aparecimento de sistemas de vigilância onde algoritmos de ML facilitam a deteção de sons e ambientes acústicos [24], pessoas, atividades, objetos e rostos [25]. Sistemas de segurança informática [26] onde os dispositivos são programados para distinguir padrões de tráfego de rede inesperados, detetando anomalias e/ou identificando tipos de ameaça e origem do ataque. Veículos e robôs autónomos [27] onde dispositivos MCU conseguem auxiliar equipamentos de maior desempenho utilizando a proximidade ao sensor para um pré-processamento [28], por exemplo, numa arquitetura de articulação em camada. Aplicações de realidade aumentada e espaços domésticos automatizados. Também as cidades podem beneficiar de controlos inteligentes de iluminação, temperatura, tráfego [29] e de qualidade do ar e água [30] auxiliando entidades governamentais na gestão dos seus territórios.

A Agricultura e Pecuária também beneficia desta transformação digital. Através de aplicações TinyML é possível incrementar a eficiência e saúde de uma produção, auxiliando os proprietários a reduzir custos de manutenção e alimentação. É possível avaliar, monitorizar e controlar estragos em estruturas. Prevenir doenças e eliminar pragas melhorando a saúde de culturas e produções [31].

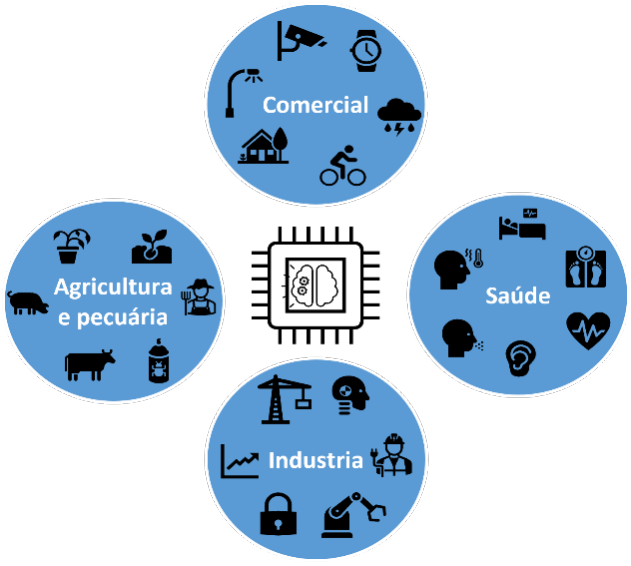


Figura 2.3: Setores de atividade TinyML.

2.3 TINYML EM PROBLEMAS ACÚSTICOS

A resolução de problemas acústicos com ML proporciona uma enorme variedade de possíveis aplicações em dispositivos IoT, especialmente em aparelhos com elevada mobilidade e/ou *wearables*, tem recebido por isso bastante atenção por parte da comunidade académica.

Existem vários géneros de problemas acústicos que podem ser diferenciados pela sua abrangência, como problemas de deteção ou classificação de cenas ou eventos acústicos. Apesar desta segmentação, os diversos tipos de problemas estão relacionados si.

A classificação de cenas acústicas (em inglês, *Acoustic Scene Classification* (ASC)) tem como objetivo identificar e classificar cenas acústicas, isto é atribuir de forma automática um rótulo identificativo a cada cena de áudio consoante o ambiente acústico onde foi capturado, *e.g.* aeroporto, parque, metro, etc. Problemas ASC são bastante desafiadores para nós humanos quando comparados com imagens. Por exemplo, num desafio de classificação de 25 cenas acústicas os humanos atingem uma exatidão em torno de 70%. Enquanto que para um problema semelhante utilizando imagens (ImageNet) a exatidão alcançada foi de 95% [32].

A classificação de eventos acústicos (em inglês, *Acoustic Event Classification* (AEC)) é aparentemente um desafio semelhante, no entanto, o objetivo consiste em rotular eventos acústicos, ou seja, classificar os vários eventos que ocorrem durante uma cena acústica *e.g.* veículos, buzinas e passos. Este é um desafio consideravelmente mais complexo, pois em situações reais eventos acústicos são muitas vezes sobrepostos temporalmente.

Tal como demonstrado na Fig. 2.4, uma cena acústica pode ser constituída por diversos eventos que podem estar temporalmente sobrepostos.

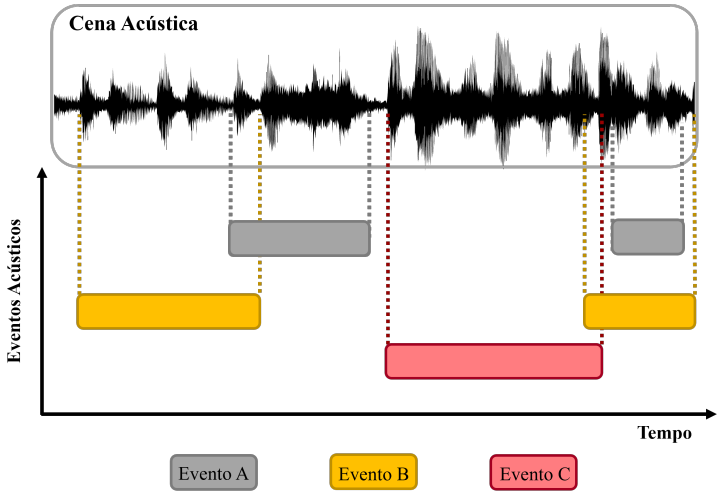


Figura 2.4: Problemas acústicos são constituídos por eventos e cenas.

Um problema mais simples é a deteção de eventos acústicos (em inglês, *Acoustic Event Detection* (AED)) um processo semelhante ao AEC, diferenciando-se no objetivo final, em que apenas é necessário detetar em vez de classificar, no entanto, o processo de deteção implícito acaba por ser semelhante. Problemas AEC e a AED estão normalmente mais associados à origem de uma fonte sonora, o que torna mais importante as relações temporais, em oposição problemas ASC estão mais focados no áudio de fundo de forma mais abrangente, tornando as relações temporais menos importantes.

Outro desafio bastante relevante é a deteção de sons anómalos (em inglês, *Anomalous Sound Detection* (ASD)), o objetivo é detetar sons atípicos, *e.g.* ruídos mecânicos, falhas de equipamentos, entre outros. Este pode ser considerado um caso especial da AED, pela dificuldade em construir um conjunto de dados adequado, dado que é extremamente difícil capturar sons atípicos [33].

Neste trabalho, pretende-se implementar algoritmos de ASC em TinyML. Estes algoritmos têm de rotular um determinado ambiente, atribuindo uma importância independente a cada um dos eventos envolventes, fazendo-o de forma equilibrada e consoante a sua importância para a tarefa. Muitas vezes o ruído de fundo de um local é essencial para classificar o ambiente evolvente, no entanto, é facilmente ofuscado por outros eventos sonoros presentes em primeiro plano no áudio, considerados "ruído" para a classificação em causa [34]. Note-se também que o sistema não pode ajustar-se em demasia a eventos específicos de um áudio. Deve evitar-se que o sistema fique sobreajustado (*overfitting*), respondendo erradamente na presença de novos dados. O *overfitting* é uma propensão bastante associada ao desenvolvimento de algoritmos ASC.

A integração de desafios de classificação e deteção acústica em dispositivos IoT tem aumentado significativamente nos últimos anos. Este é um género de tarefa muito promissora, que pode ser útil na caracterização espacial de ambiente acústicos, localização de fontes sonoras e na deteção e monitorização de eventos. Quando aplicada em dispositivos IoT, pode ter diversas aplicações [35] como, por exemplo:

- Sistemas inteligentes de vigilância;
- Deteção de emergências através ruídos e anomalias;
- Deteção precoce e preventiva de falhas em ambientes industriais.
- Dispositivos de áudio vestíeis;

2.4 ABORDAGENS A TAREFAS ASC

Para melhor compreender o estado da arte na classificação de cenas acústicas, foi realizado um estudo bibliográfico das mais recentes publicações na área. Para o efeito foram analisados os relatórios técnicos das participações com melhor desempenho

na competição internacional DCASE, uma comunidade de referência há vários anos na área de classificação e deteção de eventos e cenas acústicas. Desde 2013 que são realizados diferentes desafios enquadrados na deteção e classificação de cenas acústicas, e que têm contribuído para o desenvolvimento da investigação nesta área. Um destes desafios, a tarefa 1 (*"Task 1, Low-Complexity Acoustic Scene Classification"*), está perfeitamente enquadrado nos objetivos desta dissertação. Nas secções que se seguem são sumariamente descritas as técnicas mais relevantes utilizadas pelos participantes das duas últimas edições deste evento, para a resolução dos desafios propostos na tarefa 1.

A equipa sul-Coreana **"Kim_QTI"** [36], vencedora da edição DCASE-2021 "task1a", propõe a realização de aumento de dados através da criação espectrogramas, utilizando uma rede neuronal geradora de imagens. Desta forma, converte um espectrograma presente no *dataset*, num novo espectrograma de outro dispositivo. Utiliza também outras técnicas de aumento de dados como *Mixup* [37], *Specaugment* [38] e *time Shift* [39]. O modelo base é uma rede residual de transmissão de recursos *Convolutional Neural Network* (CNN), denominada BC-ResNet [40] (*Broadcasted Residual Network*), originalmente esta rede foi desenvolvida para a deteção de palavras-chave. Foram realizadas algumas modificações para aperfeiçoar a rede a problemas ASC, regularizando o campo recetivo e adaptando o modelo ás diferenças no domínio de entrada. É proposto também um método de normalização residual com o objetivo de melhorar a perceção da rede a novos dispositivos, permitindo manter a exatidão da classificação enquanto minimiza o impacto das respostas em frequência dos diferentes dispositivos. Para isso é realizada uma normalização de instância em frequência, é ainda adicionado um parâmetro treinável para compensar possíveis perdas de informação de domínio. São propostos também alguns métodos de compressão de modelo: a poda (em inglês *Pruning*) de magnitude não estruturada, realizada em todas as camadas CNN seguindo-se um treino adicional; a quantização QAT [41] de 8 bits de precisão realizada durante o treino nas camadas CNN; e a destilação de conhecimento (em inglês *Knowledge Distillation* (KD)) [42], onde é utilizado um modelo de grande para treinar outro modelo de menor dimensão.

A equipa internacional **"Yang_GT"** [43] atingiu o segundo lugar. Tal como a equipa vencedora, utiliza diversos métodos de aumento de dados, *Mixup* [37], *Specaugment* [38], *Spectrum correction* [44], *Pitch shift* [39], *Speed change* [39], *Random noise* [39]. Para além disso, a equipa propõe a fusão de modelos CNN (*ensemble*) [45] treinados em tarefas diferentes. São rotuladas "soluções" para problemas diferentes - um modelo rotula as cenas acústicas (objetivo da tarefa), enquanto outro classifica os áudios com outros rótulos mais genéricos (dentro de edifícios, fora de edifícios ou em transportes). O mesmo áudio é classificado então para os dois problemas, a arquitetura Inception [46] é utilizada como base. Com o objetivo de comprimir o modelo é utilizado um sistema de *Pruning* denominado

Lottery Ticket Hypothesis (LTH) [47]. O método LTH realiza o treino de modelos podados utilizando exatamente os mesmos pesos aleatórios do primeiro treino. É utilizado também KD e ainda quantização simples realizada pós-treino, convertendo o formato de 32 bits para 8 bits.

Os austríacos da equipa **"Koutini_CPJKU"** [48] que alcançaram o terceiro lugar no concurso, utilizaram as técnicas de aumento de dados *Mixup* [37] e *Pitch shift* [39]. Propõem a utilização de uma variante da arquitetura CP_ResNet [49] e com o objetivo de reduzir a complexidade do modelo realizam o agrupamento de blocos. É também aplicada uma técnica de "rolagem" (em inglês *rolling*) que permite a troca de informação entre grupos vizinhos para reduzir possíveis perdas durante o agrupamento dos blocos. Foi efetuado *pruning* de magnitude não estruturada realizado longo do treino e utilizada quantização de 16 bits. Foi ainda utilizada uma técnica de treino adversário, como adaptação de domínio [50], utilizando um bloco residual classificador de domínio em paralelo ao bloco classificador de rótulos.

O estado da arte foi atualizado com a adição das equipas melhor classificadas no concurso realizado em 2022, apesar de esta informação não ter sido utilizada como base para o trabalho desenvolvido e descrito nesta dissertação.

A equipa austríaca **"Schmid_CPJKU"** [51], vencedora da edição DCASE-2022 "task1", utilizou técnicas de aumento de dados *Mixup* [37], *Pitch shift* [39] e ainda *MixStyle/Freq-MixStyle* [52]. Características diferenciadoras dos dispositivos estão presentes essencialmente nas estatísticas de frequência, média e desvio padrão. Assim, ao aplicar *Freq-MixStyle*, é realizada uma mistura estatística na frequência melhorando a generalização entre dispositivos. Para aumentar a diversidade de dados foram agrupados áudios do mesmo local e dispositivo e realizada uma divisão aleatória de novos trechos para treino. Utilizaram a arquitetura base do tipo *transformer* mais concretamente PaSST(*The Patchout faSt Spectrogram Transformer*) [53], em que o conhecimento aprendido pelo modelo é utilizado como professor num processo KD, destilando o conhecimento numa arquitetura CNN base CP_ResNet de complexidade reduzida. Realizaram ainda quantização pós-treino.

A equipa sul-Coreana **"Chang_HYU"** [54] conseguiu o segundo lugar. Utiliza também alguns métodos de aumento de dados *Mixup* [37], *Specaugment* [38], *time and frequency masking* [55] e *Time shuffle* [56]. Propõe o uso do modelo BC-Res2Net, baseado no BC-ResNet (vencedor na edição 2021). O modelo é modificado adicionando a estrutura Res2Net [57]. São introduzidos também métodos considerados eficazes para agregar características mais discriminativas e de alta resolução como *Multi-scale Frequency-channel Attention* (MFA) [58] e *Feature Pyramid Module* (FPM) [59]. Utilizam quantização durante o treino *Quantization Aware Training* (QAT) [60], bem como a técnica KD. Esta última é usada não com o objetivo comum de redução de complexidade do modelo, mas para generalizar dispositivos

utilizando o *dataset* de forma diferente entre "professor" e "aluno". É ainda sugerida uma técnica de ajuste fino através da queda aleatória de dados, com o objetivo de generalizar melhor dispositivos minoritários.

A equipa austríaca **"Morocutti_JKU"** [61] atingiu terceiro lugar no concurso. À semelhança da maioria das propostas a concurso foram utilizadas técnicas de aumento de dados, *Mixup* [37], *pitch shift* [39], *time stretching* [62], *shifting* [39] e *adição de ruído* [39]. Foram utilizados dois modelos CNN de complexidades diferentes. O de maior complexidade designado "professor" tem uma arquitetura CNN amortecida [63]. O "aluno" tem camadas CNN convencionais apresentando complexidade reduzida. Durante o processo de treino utilizaram uma taxa de aprendizagem variável ao longo das épocas. Realizaram também a inicialização de pesos na primeira camada CNN tal como no artigo [64]. Utilizaram ainda a técnica KD e *ensemble* de três modelos obtendo a previsão média.

2.5 DISCUSSÃO

Com base na informação disponibilizada em DCASE [65], relativa a edições anteriores da tarefa 1, é possível verificar que todos os participantes extraem os recursos de áudio construindo o espectrograma de mel. Algumas equipas utilizam simultaneamente outras técnicas como *HPSS* [66] (*Harmonic and Percussive Sound Separation*) e *delta/delta-deltas* [67](primeira e segunda derivadas) no entanto, combinar estas estratégias parece ser redundante e promove o aumento desnecessário do número de parâmetros, limitando o desenvolvimento do modelo. É também bastante utilizada a redução do número de amostras (*downsample*). Esta técnica, embora signifique uma perda de informação em altas frequências, parece beneficiar o desempenho na tarefa, reduzindo o número amostras necessárias para representar o sinal, direcionando a investigação para baixas frequências. Foi consensual a extração de recursos utilizando espectrograma de mel logarítmico com 2048 STFT em 2021. A tabela.2.1 permite visualizar as principais características utilizadas pelos participantes melhor classificados.

Tabela 2.1: Técnicas de extração de recursos das propostas top-3.

Artigo	FS (kHz)	Nº filtros mel	Sobreposição de janela	Recursos
Dcase-2021				
Kim_QTI [36]	16.0	256	23%	LogMel
Yang_GT [43]	44.1	ND	50%	LogMel
Koutini_CPJKU [48]	22.05	280	ND	LogMelP

FS-Frequência de amostragem; **LogMel**-Espectrograma de mel logaritmico;
LogMelP-Espectrograma de mel logaritmico ponderado; **ND**-Não disponível.

Verifica-se a utilização massiva de técnicas de aumento de dados, por praticamente todos os participantes ao longo dos últimos anos, com o objetivo de melhorar o poder de generalização dos modelos. As técnicas mais utilizadas são *Mixup* [37], *SpecAugment* [38], mudança de tom [39] (em inglês, *Pitch shift*), mascaramento em temporal e frequência [55] (em inglês,*time and frequency masking*), e alongamento temporal [62] (em inglês, *Time stretching*). As técnicas de aumento de dados mais utilizadas pelas equipas que alcançaram melhores resultados em 2021 podem ser observadas na Tabela. 2.2.

Tabela 2.2: Técnicas de aumento de dados das propostas top-3.

Artigo	Técnicas de aumento de dados
Dcase-2021	
Kim_QTI [36]	Mixup | SpecAugment | ET
Yang_GT [43]	Mixup | CA | MC | SpecAugment |CE | MT | MV | RA
Koutini_CPJKU [48]	Mixup | MT

MT-Mudança de tom; **ET**-Embaralhamento de tempo; **RA**-Ruído aleatório; **CA**-Corte aleatório;
MC-Mistura de canais; **CE**-Correção de espectro; **MV**-Mudança de velocidade

Ao longo dos últimos anos, verifica-se que a maioria das abordagens DCASE em tarefas ASC é baseada na utilização de camadas CNNs [36, 43, 48]. A adição de conexões residuais em conjunto com camadas de extração de recursos CNN é também comum [36, 48]. A utilização de saltos entre blocos e ligações de atalho *Residual neural Network* (ResNet), tem como objetivo facilitar o fluxo de informações ao longo da rede e reduzir eventuais problemas de degradação. O caminho residual permite ignorar algumas das conexões entre camadas caso estas prejudiquem o desempenho. De notar que uma rede CNN comum explora bastante o espaço característico, e pode ficar vulnerável a características pouco úteis como ruídos.

Alguns participantes utilizam também técnicas de dilatação e agrupamento, aplicadas em camadas CNN, com o objetivo de reduzir o número de parâmetros. A técnica *rolling* sugerida [48] tem como objetivo minimizar a perda de informação verificada ao utilizar dilatação e agrupamento. A Tabela. 2.3 apresenta resumidamente os principais modelos e técnicas utilizados em pelas equipas que obtiveram melhores resultados em 2021.

Tabela 2.3: Arquiteturas e técnicas de compressão das propostas top-3.

Name	**Arquitetura**	**Técnicas**	***Ensemble***
Dcase-2021			
Kim_QTI [36]	CNN; BC-ResNet;	Quantização; Poda; Destilação de conhecimento;	✓
Yang_GT [43]	Inception;	Quantização; Poda LTH; Destilação de conhecimento;	✓
Koutini_CPJKU [48]	RF-regularized CNNs;	Quantização; Poda;	✗

São utilizados diversos métodos com o objetivo de reduzir e otimizar o modelo tais como:

- Quantização simples realizada após o treino, ou quantização *QAT* realizada durante o treino. Métodos de quantização são utilizados por praticamente todos os participantes e são essenciais para conseguir reduzir significativamente a complexidade do modelo.
- A utilização de *Pruning* não estruturado e realização de ajuste fino, com o objetivo de aumentar a esparsidade reduzindo o tamanho do modelo.
- Outro método utilizado pelas três equipas em 2021 é a destilação de conhecimento KD, um processo que para além de permitir reduzir a complexidade do modelo, melhora bastante a generalização reduzindo o *overfitting*.

Esta página foi propositadamente deixada em branco.

3

TECNOLOGIAS DE SUPORTE

Serve o presente capítulo para introduzir as principais técnicas e conceitos que servem de suporte ao desenvolvimento, avaliação e implementação de um sistema de classificação de cenas acústicas (ASC), destinado a dispositivos de baixa capacidade computacional (MCU).

3.1 EXTRAÇÃO DE RECURSOS E PRÉ-PROCESSAMENTO

A extração de recursos e pré-processamento tem um papel muito importante no desempenho de qualquer rede neuronal. Deve conseguir retirar do conjunto de dados o máximo de informação pertinente possível de acordo a finalidade/problema em causa e minimizar ao máximo os requisitos computacionais da rede neuronal. É ainda importante equilibrar o conjunto de dados e aumentar a sua robustez para que o algoritmo apresente uma boa capacidade de generalização.

3.1.1 *Processamento de dados áudio*

O som é uma onda mecânica que se propaga através de um meio de transmissão. Os microfones permitem converter o som num sinal elétrico, análogo ao mecânico (e por isso um sinal analógico) que pode ser convertido para o formato digital através de um processo de amostragem e quantização. A taxa de amostragem diz-nos quantas amostras são adquiridas por segundo (discretização do sinal no tempo) e o número de bits da quantização define o número de amplitudes diferentes que é possível discriminar (discretização do sinal em amplitude).

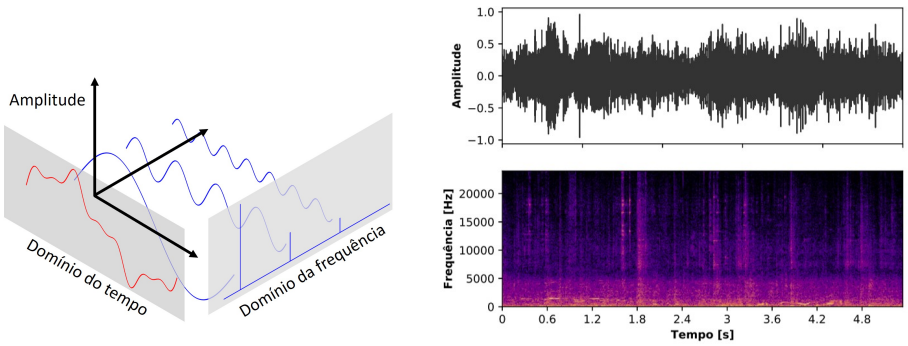
Na classificação de imagens é relativamente vulgar a utilização de dados brutos como entrada de modelos neuronais, mas o mesmo não deve ser aplicada a sinais áudio. Embora seja possível desenvolver um algoritmo assente apenas em sinais áudio no domínio temporal, é extremamente difícil diferenciar cenas acústicas através de representações exclusivamente temporais. Por outro lado, a utilização de áudio a partir de dados brutos implica geralmente um elevado número amostras. Por exemplo, um áudio estéreo adquirido a uma frequência de amostragem (em inglês, Sampling Rate (FS)) de 48kHz, com a duração de 10 segundos, necessita aproximadamente 1 milhão de amostras. As técnicas de pré-processamento e os

métodos de extração de recursos permitem efetivamente reduzir a complexidade do processo de classificação, bem como aumentar o seu desempenho. As mais relevantes são discutidas nos parágrafos que se seguem.

- Transformada de Fourier

  Sabe-se que utilização do domínio de frequência tende a superar sinais no domínio do tempo em tarefas de classificação de áudio [68,69]. A transformada de *Fourier* (em inglês, *Fast Fourier transform* (FFT)) permite representar eficientemente um sinal de áudio complexo, num determinado período de tempo, através da soma ponderada de sinusoides de várias frequências. Desta forma, é possível converter o sinal para o domínio da frequência e obter uma representação espectral linear do sinal [70, 71].

  Ao realizar a transformada de *Fourier* sobre a totalidade de um sinal, são perdidas variações e referências temporais que são muitas vezes importantes. Para preservar estas referências temporais é utilizada uma variante da transformada de *Fourier* designada *Short Time Fourier Transform* (STFT). Esta técnica recorre a uma janela deslizante ao longo do domínio tempo do sinal, calculando a FFT ao longo de pequenos segmentos. Assim são preservadas informações e variações ao longo do domínio do temporal. Durante a segmentação do sinal é importante que as janelas adjacentes apresentem alguma sobreposição (*overlap* em inglês), com o objetivo de suavizar as transições temporais entre janelas. Um espectrograma, tal como ilustra a Fig. 3.1, é uma representação visual e tridimensional do áudio, por incluir os domínios temporal, frequência e amplitude [72].



(a) **(b)**

Figura 3.1: FFT [73] (a) Áudio espectrograma STFT [69] (b)

- Espectrograma de mel
  O espectrograma de mel é obtido através da aplicação da escala de mel ao espectrograma, conseguindo fornecer uma representação de menor complexidade. A escala de mel é um descritor de áudio, baseado na perceção humana. O ser

humano tem uma perceção logarítmica e não linear da escala de frequência, isto é, os humanos diferenciam mais facilmente sinais de baixa frequência em oposição aos de alta frequência, mesmo que dentro do seu espectro auditivo. Consoante as características de audição humana é atribuindo mais importância (peso) a elementos de baixa frequência comparativamente as altas frequências. A escala mel é então uma escala perceção não linear, construida de forma experimental. O espectrograma de mel pode ser obtido através da aplicação de um banco de filtros sobre o espectrograma no qual é posteriormente calculado o logaritmo tal como representado na Fig. 3.2. Em algoritmos de classificação e deteção acústica, geralmente, os espectrogramas de mel superam os espectrogramas lineares comuns [74].

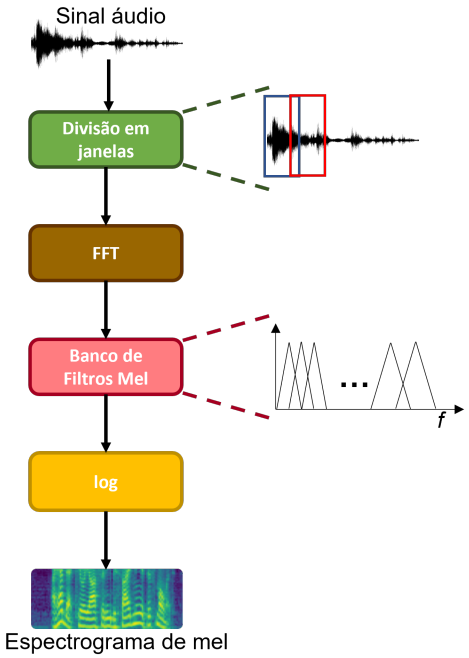


Figura 3.2: Etapas gerais de processamento de áudio para espectrograma de mel.

- Downsample

  Como referido em 2.1, existe a necessidade de respeitar exigentes limites de complexidade no paradigma TinyML. A redução da taxa de amostragem de um sinal (em inglês, *downsample*), é uma técnica que permite reduzir o número de amostras necessárias para representar um sinal digital. Conseguindo diminuir a complexidade do sinal e consequentemente do algoritmo, reduzindo também o tempo de treino. A aplicação de *downsample* resulta numa perda substancial de informação em frequências mais elevadas, no entanto, esta diminuição da largura de banda do sinal pode não prejudicar a tarefa em causa. Em muitas aplicações, a informação mais relevante se encontra geralmente em frequências mais baixas. Efetivamente, esta diminuição pode até beneficiar o algoritmo classificação [75]. Portanto, é necessário sempre avaliar o impacto do *downsample* especificamente para o problema em causa.

3.1.2 *Aumento de dados*

O desempenho de uma rede neuronal é tanto melhor, quanto melhor a qualidade do *dataset* que a alimenta. É desejo da maioria das aplicações que um modelo classifique de forma eficaz mesmo para dados não treinados, ou seja, o modelo deve generalizar corretamente, o que significa que tem de aprender recursos úteis sem se ajustar em demasia ao *dataset*. Quando uma rede neuronal está ajustada em demasia ao conjunto de dados tende a aprender recursos muito específicos que se tornam prejudiciais à tarefa, originando mau desempenho na presença de novos dados. Para evitar problemas deste género é necessário um *dataset* equilibrado, com uma grande quantidade de dados rotulados e preferencialmente adquiridos a partir de vários dispositivos. Nem sempre é fácil ou viável, obter grandes quantidades de dados de forma natural, por exemplo, em áudio é importante ter vários dispositivos de captura, despender bastante tempo de aquisição, recolher dados em ambientes e situações temporalmente distintos e realizar um processo de rotulagem minucioso.

O aumento de dados (em inglês, *Data Augmentation*) é uma etapa do processamento de dados amplamente utilizada, que visa aumentar artificialmente o conjunto de dados a partir de dados existentes [76], através da aplicação de deformações e transformações aos dados de treino. Técnicas de aumento de dados, são utilizadas com sucesso em diversas tarefas ML, tais como reconhecimento de voz [37], deteção de texto [77], deteção de objetos [78] e classificação acústica [79]. Em tarefas relacionadas com áudio, as transformações podem ser aplicadas tanto ao sinal em domínio temporal, como no espectrograma.

De seguida são apresentadas algumas técnicas vulgarmente utilizadas em problemas ASC. Técnicas de mudança de tom, deslocamento e alongamento temporal são exemplos de transformações básicas realizadas sobre o domínio temporal. Já os métodos Mixup [37] e SpecAugment [38] são aplicados sobre o espectrograma.

- Mudança de tom [39] (em inglês, *Pitch shifting*), permite aumentar ou diminuir o tom do áudio segundo a escala musical.
- Deslocamento temporal [39] (em inglês, *Time shift*), deslocamento das amostras segundo a dimensão temporal.
- Alongamento temporal [62] (em inglês, *Time stretching*), alteração da duração temporal do sinal através da velocidade.
- Mixup [37, 80] é um método de mistura de informação utilizado em imagens e espectrogramas, implementado durante o processo de treino. Utiliza dois elementos do *dataset* de forma aleatória misturando os seus espectrogramas e respetivos rótulos. Esta técnica é implementada pela equação 3.1, onde $((x_i, y_i)$ e $((x_j, y_j)$ representam duas amostras do conjunto de dados, com x a

representar o espectrograma e y o rótulo. O parâmetro ($\lambda \in [0,1]$) é responsável por atribuir um determinado peso às amostras a misturar representando a proporção da mistura, tal como pretende ilustrar a Fig. 3.3.

$$\begin{aligned} \widehat{x} &= \lambda x_i + (1-\lambda)\, x_j \\ \widehat{y} &= \lambda y_i + (1-\lambda)\, y_j \end{aligned} \tag{3.1}$$

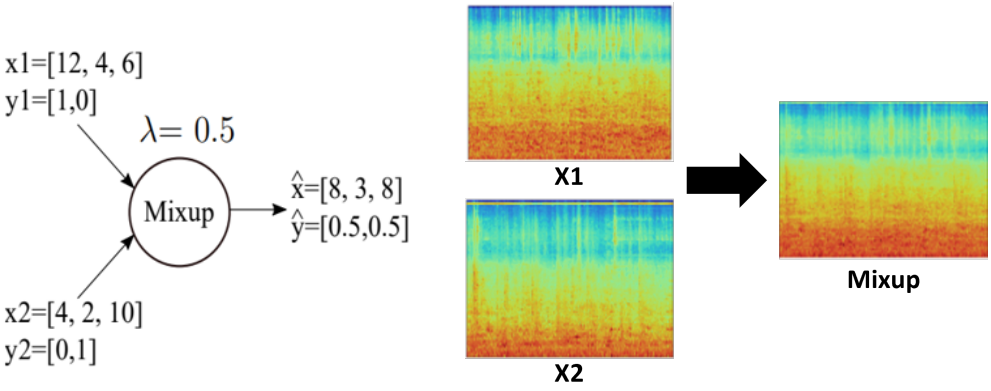


Figura 3.3: Utilização de Mixup com mistura a 50%: Exemplo numérico de Mixup a) [80]. b) Aplicação Mixup entre dois espectrogramas [37, 81].

- SpecAugment [38] é outra estratégia de aumento de dados, proposta inicialmente em problemas de reconhecimento de fala. Consiste na aplicação de uma máscara independente nos domínios temporal e/ou de frequência, que atribui uma determinada importância a uma porção do domínio. O *SpecAugment*, coloca amostras consecutivas do espectrograma log mel de um determinado intervalo a zero, como mostra a Fig. 3.4. Aplicando estas distorções de forma aleatória ao longo do conjunto de dados, é possível aumentar o *dataset* com o objetivo de melhorar a generalização do modelo. Esta técnica pode ser aplicada durante o treino ou no pré-processamento dos dados.

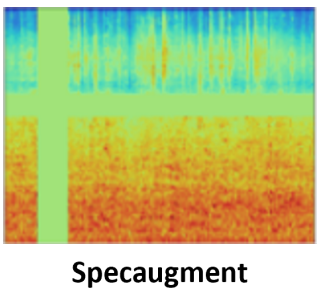


Figura 3.4: Método *SpecAugment* aplicado ao domínio temporal e de frequência [38, 81].

3.2 ARQUITETURAS TÍPICAS ASC

Tal como identificado na secção 2.4, as arquiteturas tipicamente utilizadas em problemas ASC utilizam camadas convolucionais. A maioria dos participantes DCASE utiliza variações de arquiteturas CNN e ResNet.

3.2.1 *Conceitos introdutórios de redes neuronais*

O elemento base de uma rede neuronal, designa-se neurónio, foi introduzido pela arquitetura *Perceptron*. Um neurónio é no fundo, uma função matemática que calcula uma saída ponderada com base num sinal de entrada. Um neurónio, Fig. 3.5 é essencialmente constituído por um conjunto de pesos responsáveis pela ponderação simples do sinal de entrada (operação de multiplicação), o "bias" também conhecido como limiar de ativação é uma variável somada ao resultado da ponderação cujo objetivo é deslocar a função melhorando a sua capacidade de ajuste aos dados fornecidos, o bias também permite que a saída do neurónio seja diferente de zero em caso de todas as entradas apresentarem valor nulo. Pesos e bias são aprendidos ao longo do treino por retropropagação com base no erro obtido entre a saída desejada e a obtida [82, 83].

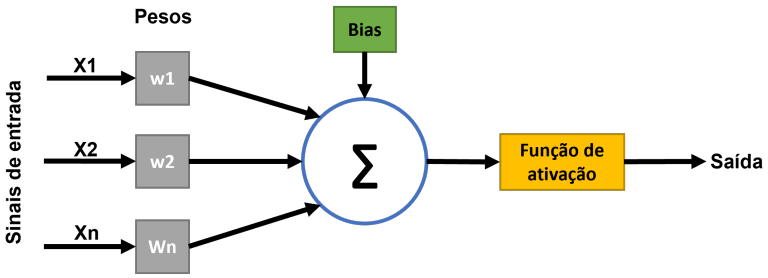


Figura 3.5: Esquema de funcionamento de um neurónio.

O neurónio pode representar tanto regressões lineares simples, como funções mais complexos, para isso são aplicadas funções de ativação. Funções de ativação são habitualmente aplicadas a todas as camadas de aprendizagem de uma rede neuronal, por exemplo, densas e convolucionais. A maioria das plataformas de desenvolvimento de redes neuronais aplica atualmente estas funções como padrão na criação das camadas. Uma camada que não utilize funções de ativação apenas consegue reproduzir funções lineares, tornado o processo de aprendizagem demorado, difícil e impossível em problemas complexos. A utilização de funções de ativação (não lineares) permite que o modelo consiga aprender funções bastante mais complexas e não lineares, facilitando o processo de aprendizagem [84, 85].

A função *ReLU* é uma função não linear e computacionalmente leve, definida pela Eq. 3.2. Esta função retorna 0 para valores negativos o que significa inatividade do neurónio durante determinada época (acelerando o treino), para valores positivos a função tem um comportamento linear e retorna o próprio valor permitindo a retro-propagação do erro possibilitando a aprendizagem [84, 85].

$$f(x) = \begin{cases} x, & \text{se } (x \geq 0) \\ 0, & \text{se } (x < 0) \end{cases} \tag{3.2}$$

A função *softmax* é normalmente aplicada na última camada com o objetivo de efetuar a classificação de classes múltiplas, é vista como uma função de probabilidade que atribui para cada classe valores entre [0,1]. A soma de todas as classes é 1, logo pode ser entendida como probabilidade de acerto [86]. As funções de ativação utilizadas nesta dissertação são também as mais usuais.

Redes densas ou totalmente conectadas são constituídas por vários neurónios agrupados em várias camadas que estão totalmente conectados entre si, aumentando a capacidade de aprendizagem, Fig. 3.6.

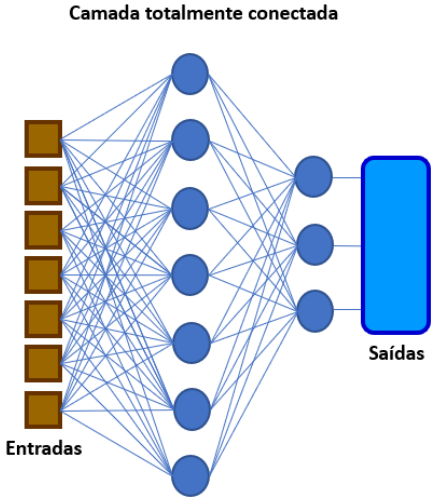


Figura 3.6: Camada totalmente conectada.

3.2.2 *Arquitetura CNN*

Modelos CNNs são amplamente utilizados e especializados em processamento de dados com duas dimensões (2D). Conseguem aprender recursos e características relevantes de forma hierárquica e autónoma, ao contrário de métodos ML mais primitivos [84]. As camadas CNN apresentam boa capacidade para capturar referências temporais, espaciais e de frequência, conseguindo oferecer um bom desempenho a

detetar objetos, distinguir sons, ruídos e ambientes acústicos. As redes CNN têm capacidade de aprender e identificar padrões espectro-temporais o que é relevante em tarefas de classificação ASC. Tal como todas as redes DL o seu desempenho está limitado pela qualidade e quantidade de dados do *dataset*, para conseguir classificar dados não treinados [72]. Comparativamente a redes tradicionais totalmente conectadas, a aplicação de camadas convolucionais permite utilizar um menor número de pesos partilhando o mesmo núcleo (em inglês *kernel*) ao longo de toda amostra de entrada. Assim é computacionalmente menos exigente, necessitando também de menos armazenamento, pois não tem de aprender um número de pesos por cada píxel da amostra, como acontece em camadas totalmente conectadas [87].

Um modelo CNN é constituído essencialmente por duas etapas, a primeira é a extração de recursos utilizando uma ou várias camadas convolucionais e a segunda é a etapa de classificação responsável por aprender a função que dá resposta ao problema com base na informação extraída na camada convolucional. Um processo de aprendizagem em redes CNN pode conter várias camadas convolucionais, como mostra a Fig. 3.7. Quanto mais camadas convolucionais são utilizadas mais profunda é a rede, o que permite extrair informação mais sofisticada. Camadas mais próximas da entrada extraem características de baixo nível, enquanto camadas mais distantes da entrada extraem particularidades mais abstratas [68]. Estas camadas convolucionais são compostas essencialmente por um bloco convolucional e um bloco de redução/agrupamento de recursos (*pooling*). Já o classificador é formado por uma camada totalmente conectada que interliga os neurónios da etapa anterior a uma camada densa responsável pela classificação final.

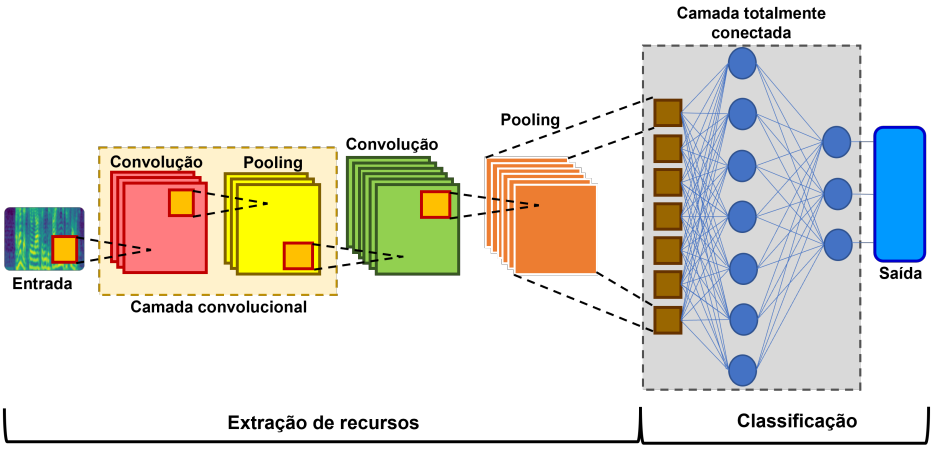


Figura 3.7: Arquitetura de uma rede neuronal CNN tradicional, com duas camadas convolucionais.

Os blocos que tipicamente compõem as camadas convolucionais de redes CNN são descritos de seguida em maior detalhe.

- Bloco Convolucional - Este bloco é responsável por extrair os recursos, atuando como um filtro que deteta e procura padrões característicos do sinal. As camadas de convolução são compostas por um conjunto de filtros onde cada um representa uma característica apreendida pela rede [68]. A operação de convolução é implementada pelas bibliotecas DL, como um produto escalar entre a amostra de entrada e um filtro de pesos designado em inglês de *kernel*. Como resultado é obtido o chamado mapa de características. Este cálculo é realizado deslizando o *kernel* ao longo da amostra, sendo os seus valores multiplicados e somados para obter um único valor escalar. O processo é repetido até percorrer toda a amostra, tal como ilustra a Fig. 3.8. O *kernel* é uma matriz de valores inicialmente aleatórios também designados de pesos. Estes pesos vão sendo ajustados ao longo de cada época de treino, aprendendo a extrair recursos mais significativos e ficando por isso cada vez mais eficientes. Depois da operação de convolução é tipicamente aplicada uma função de ativação não linear [84, 88].

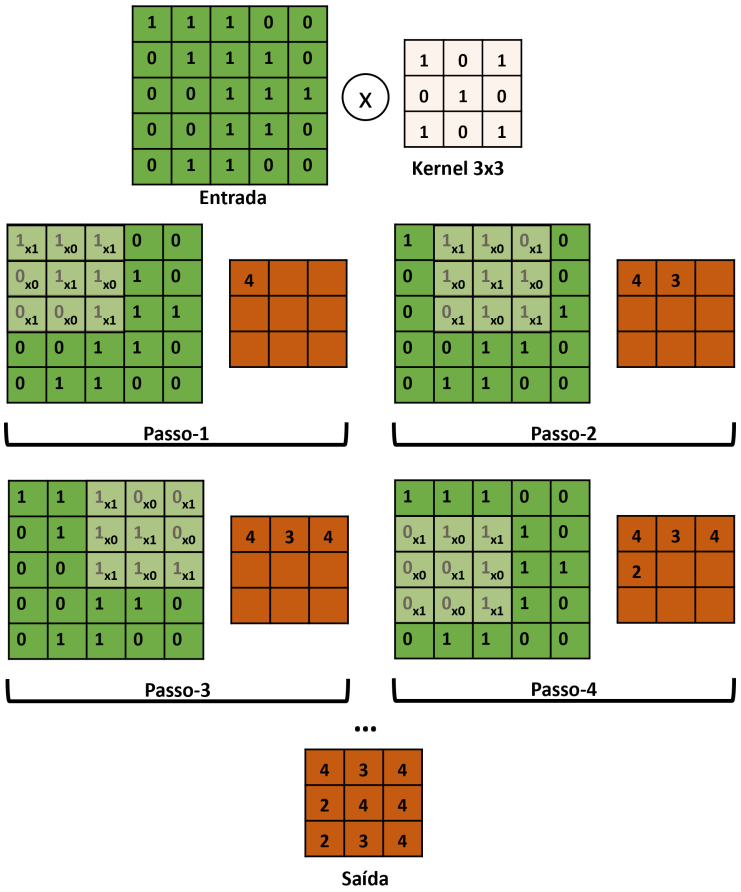


Figura 3.8: Operação convolução.

- Operação de redução de recursos (*pooling*) - A operação de *pooling* é utilizada para simplificar, reduzir/agrupar o mapa de características diminuindo a sua dimensão e com isso reduzir o custo computacional. Com a utilização de

pooling a rede melhora a generalização, conseguindo abstrair-se de pequenos detalhes e focando-se em características mais genéricas [84]. Os tipos de *pooling* mais utilizados são o *Max-Pooling* e *Average-Pooling*, que são aplicados numa região designada de *kernel*. O *Max-Pooling* determina o valor da amostra máximo em cada uma dessas regiões, enquanto o *Average-Pooling* determina o valor médio [84, 88], como representa a Fig. 3.9.

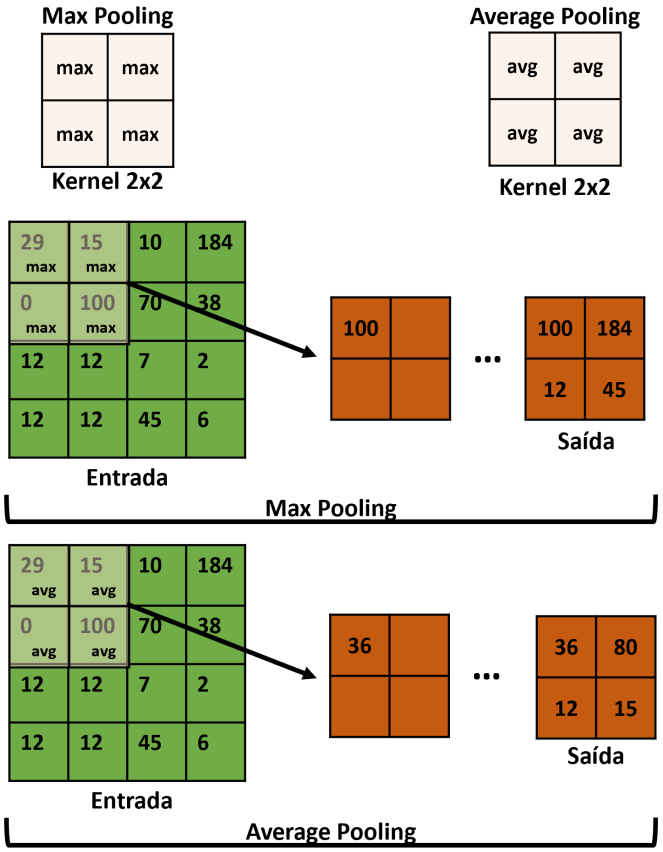


Figura 3.9: Operação pooling.

Para além destes dois blocos fundamentais, é ainda bastante comum a utilização de normalização em lote (em inglês, *Batch Normalization*) e regularização por abandono (em inglês, *Dropout*).

- Batch Normalization - Esta é uma técnica de normalização comum, utilizada em diversas arquiteturas [89], que se formaliza a partir da aplicação da equação 3.3. Para cada mapa de características, a normalização é obtida utilizando a média (μ_i) e desvio padrão (σ_i), sendo ainda adicionados dois parâmetros (γ) e (β) aprendidos durante o treino. Durante o processo de treino, é ainda calculada e armazenada a média e desvio padrão móvel. O objetivo é normalizar todos os recursos e aplicar durante o processo de inferência [90]. A utilização de *Batch Normalization*, ajuda também a acelerar e estabilizar o processo de treino, melhorando o desempenho da rede.

$$\widehat{x_i} = \gamma \cdot \frac{x_i - \mu_i}{\sigma_i} + \beta \tag{3.3}$$

- Dropout - O *Dropout* [91] é uma técnica de regularização (evitar o excesso de confiança) por abandono de neurónios, aplicada durante o processo de treino. Tem como objetivo principal evitar o *overfitting*. Durante o processo de treino e ao longo de cada época, uma percentagem de neurónios é selecionada aleatoriamente para ser ignorada temporariamente. Os neurónios selecionados para abandono e respetivas conexões são então ignorados e os seus pesos não são atualizados. Desta forma a rede é forçada a utilizar todos os seus neurónios de forma equilibrada e sem dependerem muito uns dos outros, não confiando em demasia em nenhum, forçando assim que todos aprendam (melhorando a generalização). No fundo, o *Dropout* permite treinar diferentes modelos (compostos por diferentes neurónios) a cada época, compartilhando os seus pesos entre si [92]. Após a aplicação de *Dropout*, o modelo necessitará de mais épocas para convergir, embora seja diminuído o tempo de treino a cada época. Quanto maior for a dimensão da rede ou o número de épocas de treino, maior a probabilidade da rede se ajustar demasiado, sendo nestes casos especialmente importante a utilização de *Dropout*.

3.2.3 *Arquitetura ResNet*

A arquitetura ResNet [93], introduz o conceito de conexão residual, com o objetivo de evitar o problema de dissipação/explosão de gradiente, fenómeno que provoca instabilidade durante o treino do modelo [84]. Um modelo ResNet é constituído essencialmente por diversas camadas conectadas entre si e interligadas através de caminhos residuais/identidade, como mostra a Fig. 3.10. O caminho residual é um atalho que permite ignorar algumas das conexões entre camadas. Assim, caso exista por algum motivo camadas que prejudiquem o desempenho da arquitetura, o seu impacto é reduzido, permitindo acelerar a convergência da rede. Estas ligações de atalho são úteis especialmente em redes neuronais profundas, minimizando a propagação de uma possível degradação em profundidade [94].

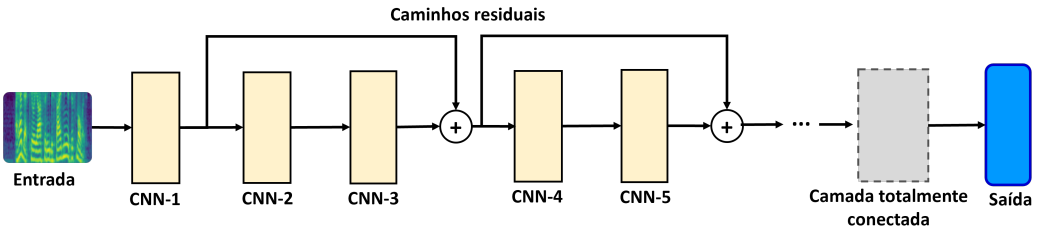


Figura 3.10: Exemplo de arquitetura tipo ResNet.

3.3 MÉTRICAS PARA AVALIAÇÃO DE DESEMPENHO

A escolha de métricas para avaliação de desempenho deve ter em consideração o tipo de problema a avaliar e ainda a qualidade do conjunto de dados utilizado no processo de avaliação. De seguida são apresentadas algumas métricas de avaliação vulgarmente utilizadas em problemas de classificação.

3.3.1 *Entropia Binária Cruzada*

Entropia Binária Cruzada (ou em inglês, *Log loss*) é uma métrica bastante utilizada em problemas de classificação binária em tarefas com múltiplas classes, mede a divergência entre as probabilidades previstas pelo modelo e a saída real penalizando logaritmicamente os resultados entre 0 e 1. Quanto mais próximas as previsões forem dos valores reais menor será o *Log loss* o que significa melhor desempenho do modelo [95, 96].

A Entropia Binária Cruzada pode ser calculada pela seguinte Eq. 3.4

- (M) - Numero de classes possíveis.
- (N) - Numero de instâncias (amostras).
- (y_{ij}) - Variável binária que indica se a classe j classifica corretamente a instância i.
- (p_{ij}) - Probabilidade prevista pelo modelo de atribuir a classe j instância i.

$$\text{Log\_Loss} = -\frac{1}{N}\sum_{i}^{N}\sum_{j}^{M} y_{ij} \cdot \log\left(p_{ij}\right) \tag{3.4}$$

3.3.2 *Matriz de confusão*

A matriz de confusão (em inglês, *confusion matrix*) é uma matriz que permite avaliar o desempenho de um modelo de classificação. Esta matriz condensa os

resultados da previsão do modelo num determinado *dataset*, comparando os valores alvo (verdadeiros) com os previstos. Através desta matriz é possível observar o desempenho do modelo, identificar rótulos com melhor/pior desempenho, localizar rótulos mais confundíveis entre si e calcular diversas outras métricas [97]. Quanto mais amostras se encontrarem na diagonal principal, melhor será o desempenho do modelo.

A Fig. 3.11 permite observar uma matriz de confusão analisada a partir da classe (b), onde:

- Verdadeiros Positivos (**VP**): indica que o valor real e o previsto são ambos de classe (b), ou seja, o único resultado certo.
- Verdadeiros Negativos (**VN**): indica que os restantes valores fora do foco classe (b), nunca serão (b).
- Falsos Positivos (**FP**) e Falsos Negativos(**FN**): indicam que os elementos foram classificados incorretamente na linha/coluna da classe (b).

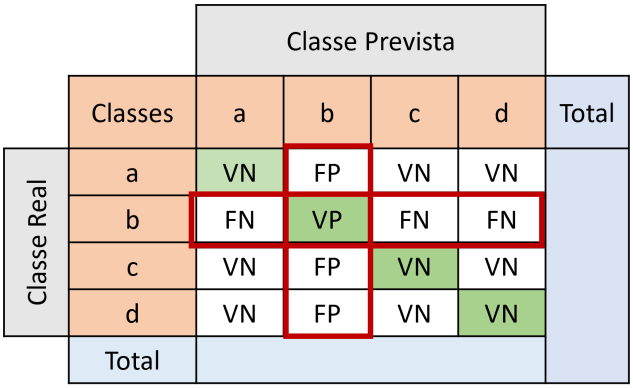


Figura 3.11: Exemplo matriz de confusão, foco na classe (b).

3.3.3 *Exatidão*

A exatidão (em inglês, *accuracy*) é uma métrica muito popular, utilizada para medir o desempenho de um modelo de classificação através da percentagem de acerto. Permite analisar a razão de acerto do modelo, ou seja, a percentagem de vezes que a classe prevista é igual à classe verdadeira. Genericamente é calculada a partir da expressão Eq. 3.5.

$$\text{Accuracy} = \frac{\text{Numero de previsões corretas}}{\text{Total de previsões}} \tag{3.5}$$

Em classificação de múltiplas classes, a *accuracy* pode ser calculada com base na matriz de confusão, como demonstrado pela Eq. 3.6. No numerador colocamos os valores corretamente classificados pelo modelo, que correspondem à diagonal

principal da matriz confusão. No denominador consideramos todos os elementos da matriz.

$$\text{Accuracy} = \frac{\text{VP+VN}}{\text{VP+VN+FP+FN}} \tag{3.6}$$

Em conjuntos de dados desequilibrados, a utilização da *accuracy* pode ocultar erros de previsão elevados em classes pouco relevantes na dimensão total de dados [97]. Nestes casos é preferível calcular a exatidão macro-média (em inglês,*macro-average accuracy*), segundo a expressão Eq. 3.7. A *macro-average accuracy* é a razão entre os elementos verdadeiramente positivos (VP), sobre o total de elementos previstos (VP+FP) de uma classe (k), ou seja, diz-nos a razão de positivos que é realmente positiva para cada classe [97].

$$\text{Macro-Average Accuracy} = \frac{\sum_{k=1}^{K} \frac{\text{VP}_k}{\text{VP}_k+\text{FP}_k}}{\text{k}} \tag{3.7}$$

3.3.4 *Overfitting*

O sobreajuste (em inglês, *overfitting*) é uma condição verificada quando um modelo se ajusta excessivamente aos dados de treino. Durante o processo de aprendizagem, o modelo pode acabar por se focar em características demasiado específicas e que não serão úteis, como detalhes irrelevantes ou ruídos. Como consequência o algoritmo acaba por não conseguir manter o bom desempenho que aparenta durante o treino quando exposto a novos dados. Esta dificuldade de generalização do modelo é facilmente detetada quando o valor do erro para os dados de treino atinge valores mínimos, no entanto, quando o modelo é exposto ao conjunto de teste atinge valores de erro muito elevados. Significando que a rede memorizou características das amostras de treino, que não são úteis, não aprendeu recursos que lhe permitam generalizar para novos dados.

Este problema pode ser minimizado recorrendo a técnicas como:

- Aumentar o *dataset* de treino, introduzir mecanismos de aumento de dados (secção 3.1.2), ou aplicar regularização de rótulos (secção 3.4.1).
- Fazer uma seleção adequada de recursos e do conjunto de dados de treino, bem como realizar um processamento e extração de recursos eficaz (secção 3.1.1).
- Escolher o modelo, e a profundidade, que melhor se adequa ao problema (secção 3.5) - quanto mais complexo o modelo maior será a probabilidade de *overfitting*.

- Utilizar técnicas de regularização, por exemplo, *dropout* (secção 3.2.2) ou regularização "L1/L2" [98], que penalizam os pesos de maior magnitude para equilibrar a aprendizagem do modelo durante o treino.
- Aplicar algumas formas de agrupamento de modelos ou previsões, que permitam obter resultados mais precisos e equilibrados.
- Realizar a paragem antecipada do treino (em inglês, *early stopping*) (secção 3.4.2).

Existe também a possibilidade "contrária" ao *overfitting*, ou seja, o modelo não consegue sequer aprender o conjunto de treino, designa-se subajuste (em inglês, *underfitting*). Neste caso o modelo é insuficiente para aprender e não consegue estabelecer relações entre o sinal de entrada e a saída desejada [99]. Geralmente este problema ocorre devido a:

- Modelos simples, i.e., de baixa complexidade.
- Pouco tempo dedicado ao treino.
- Seleção de recursos pouco adequada ao problema.
- Métodos de regularização exagerados.

A Fig. 3.12, exemplifica as possíveis situações de ajuste de um modelo relativamente ao conjunto de dados. Do lado esquerdo temos uma possível divisão entre duas classes aprendidas pelo modelo e do lado direito o estado de aprendizagem ao longo do treino, considerando que o conjunto de validação tem dados novos. Em a) é possível observar o caso de *underfitting* o modelo não aprendeu o suficiente o erro é maior comparativamente as restantes figuras o que resulta num fraco desempenho, por essa razão reage de forma semelhante tanto para os dados de treino como de validação. Em b) o modelo está em *overfitting* aprendeu padrões muito específicos e ajustados ao conjunto de treino, por essa razão apresenta mau desempenho quando exposto ao conjunto de validação, o erro de validação dispara e aumenta o valor. Por fim a situação mais equilibrada c), o erro de validação diminui e estabiliza de forma semelhante ao erro de validação, significando que o modelo aprendeu ao longo de todo o treino. Apesar de o erro de validação ser ainda maior que o de treino, o modelo está equilibrado e vai responder de forma semelhante tanto para novos dados como para dados utilizados durante o treino.

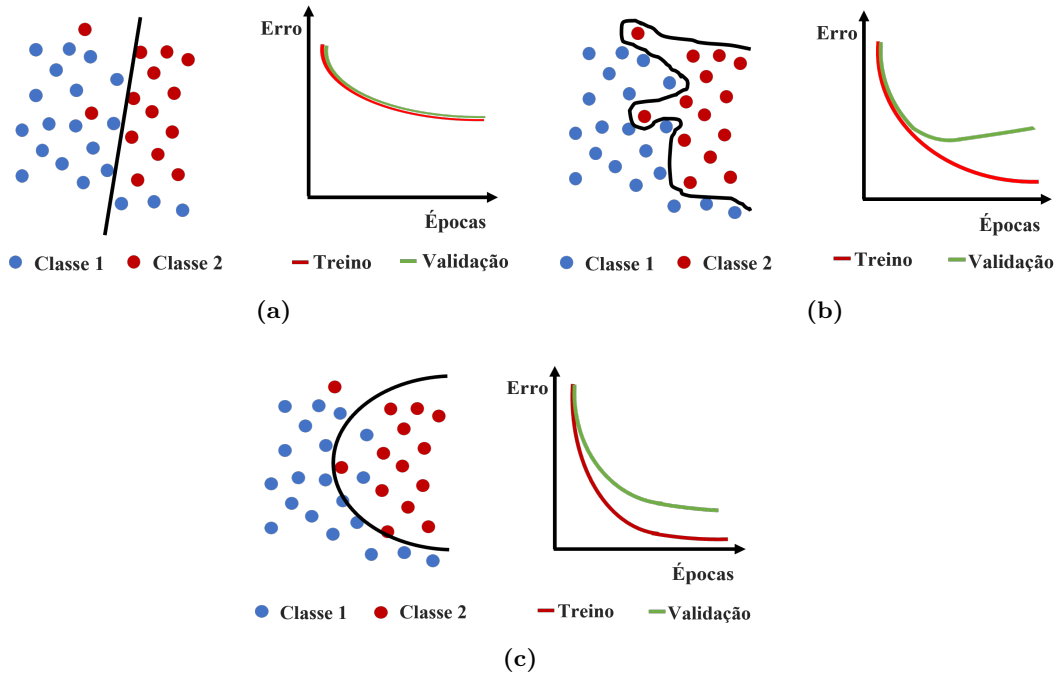


Figura 3.12: Tipos de ajuste entre modelo e conjunto de dados: (a) sobajuste; (b) Sobreajuste; (c) Equilibrado.

3.4 TÉCNICAS DE REGULARIZAÇÃO E MELHORIA APRENDIZAGEM

Técnicas de regularização ou suavização, são habitualmente fáceis de implementar e produzem bons resultados, conseguindo que o modelo generalize melhor. Estes métodos tendem a evitar que o modelo se ajuste demasiado a determinada classe ou conjunto de dados, especialmente em conjuntos de dados desequilibrados. Técnicas de aprendizagem de classificação, como classes múltiplas e aprendizagem de modelos conjuntos, podem também melhorar e regularizar a predição de um sistema, auxiliando os algoritmos a delimitar espacialmente os rótulos a classificar ou através da combinação de modelos a obter melhores resultados.

3.4.1 *Label Smoothing*

A suavização de rótulos (em inglês, *Label Smoothing*) [100], é uma ferramenta de regularização simples que atua nos rótulos. É aplicada durante o treino e tem como principais objetivos, evitar o *overfitting* e melhorar a calibração do modelo evitando o excesso de confiança da rede. Geralmente em tarefas de classificação, as classes são representadas através de um grupo de bits, técnica designada *one-hot*. Ao utilizar *Label Smoothing*, o rótulo verdadeiro doa um determinado fator de incerteza para os restantes, retirando alguma confiança ao modelo. Esta técnica atua como um

regularizador, diminuindo e equilibrando a taxa de aprendizagem. Ao atribuir um fator de incerteza ao modelo este não aprende com tanta "confiança", ou seja, os seus pesos de maior magnitude são penalizados [101]. Desta forma a rede terá de utilizar todos os seus neurónios de uma forma mais equilibrada sem confiar demasiado. Atualmente a técnica *Label Smoothing* é amplamente utilizada nos mais diversos modelos de classificação DL. No entanto, segundo os autores em [102], a utilização de suavização de rótulos em conjunto com destilação de conhecimento pode diminuir o desempenho do modelo. Esta técnica pode ser implementada segundo a Eq. 3.8.

- (y_{ls}) - Rótulo suavizado.
- (y_{hot}) - Rótulo original formato *one-hot*.
- (α) - Fator de incerteza $(0 \geq \alpha < 1)$
- (k) - Número total de classes.

$$y_{ls} = (1 - \alpha) \cdot y_{hot} + (\alpha / k) \tag{3.8}$$

3.4.2 *Early stopping*

É uma técnica de regularização, que interrompe o processo de treino durante a fase de aprendizagem. À medida que o treino é realizado, esta técnica vai a acompanhado a diminuição do erro tanto para os dados treino como de validação. Quando o modelo começa a despertar sinais de *overfitting* (o erro de avaliação começa a aumentar, enquanto o erro de treino continua a diminuir), o treino deve ser terminado Fig 3.13. Ao utilizar *Early stopping* corre-se o risco de parar o processo de treino antecipadamente limitando a aprendizagem do modelo, pelo que deve ser dada uma tolerância de algumas épocas de treino [103].

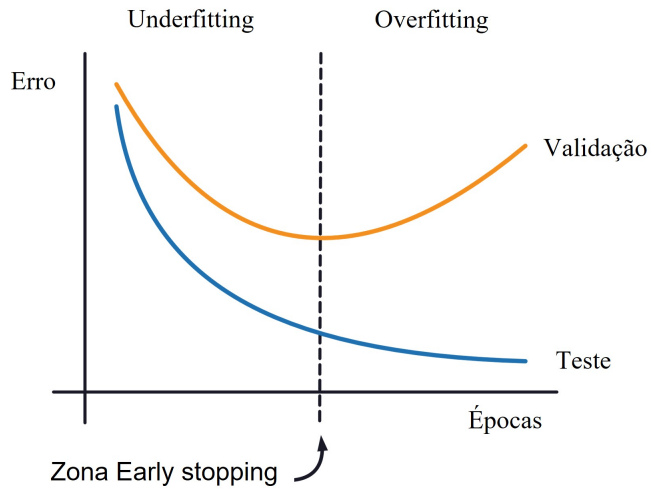


Figura 3.13: Exemplo de *Early stopping* [104].

3.4.3 *Técnicas de classificação classes múltiplas*

Numa classificação binária de classes múltiplas, tradicionalmente todas as classes são treinadas em simultâneo, no entanto, outras técnicas podem ser utilizadas com o objetivo de melhorar o desempenho do modelo. As abordagens mais comuns são "Um contra todos" e "Um contra um" em inglês, (*One-vs-All* (OvA) e *One-vs-One* (OvO)) respetivamente. Habitualmente é mais utilizado o método OvA pela maior facilidade de implementação em problemas com um grande número de classes, mas é o OvO quem costuma produzir melhores resultados. Ambas as técnicas necessitam habitualmente de reajustar o conjunto de dados necessitando de vários subconjuntos, aumentando assim a complexidade de implementação destas técnicas com o aumento do número de classes. Para além da necessidade de múltiplos conjuntos de dados, deve ser realizado um modelo para cada conjunto e o respetivo treino. O agrupamento final dos modelos pode ser realizado recorrendo a *Ensemble* através de métricas de decisão simples ou formas mais complexas como modelos de decisão [105–107].

A técnica *One-vs-All* (OvA) é provavelmente a mais utilizada, coloca a classificação de cada uma das classes contra todas as outras. Para isso é necessário reajustar o *dataset*, colocando a classe em análise a "1" e todas as restantes a "0". O objetivo é que cada um dos modelos aprenda a distinguir uma classe de todas as restantes. A utilização do método OvA apresenta como principal desvantagem o facto de requerer que o *dataset*, modelo e treino sejam reajustados/realizados k vezes (sendo k o número de classes), tal como ilustrado na Fig.3.14. Outra desvantagem é que colocar todas as classes contra uma provoca um desequilíbrio nos dados, podendo enganar as métricas de avaliação do modelo - a classe minoritária pode ser confundida com ruído perdendo-se padrões importantes [105, 106].

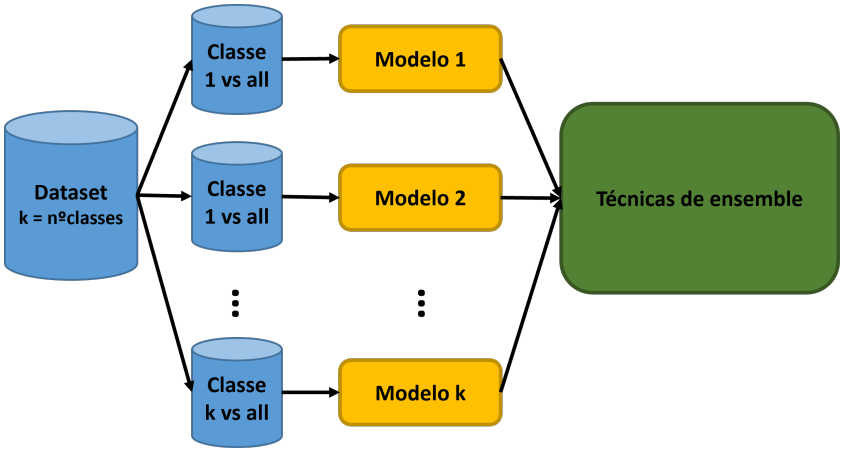


Figura 3.14: Esquema de implementação One-vs-all.

O método de classificação *One-vs-One* (OvO), é também usado com regularidade em problemas de classificação, geralmente em problemas com poucas classes. Esta técnica é útil para discriminar todas as classes entre si. Obtém habitualmente melhores resultados que o OvA. São aprendidos limites de decisão para todas as classes, no entanto em amostras muito semelhantes o modelo pode ter dificuldade em aprender esses limites. Infelizmente este sistema apresenta uma enorme desvantagem, que se agrava com o aumento do número de classes. O método OvO é implementado de forma semelhante ao OvA, mas necessita que o *dataset*, modelo e treino sejam reajustados e realizados $k(k-1)/2$ vezes (sendo k o número de classes), o que torna este método difícil de implementar. Contrariamente ao OvA, as classes neste método disputam entre si, equilibrando o conjunto de dados, favorecendo o desempenho do conjunto [105, 106].

3.4.4 *Ensemble learning*

A aprendizagem em conjunto (em inglês, *ensemble learning*), é a designação dada a combinação de dois ou mais modelos para realizar determinada previsão. Embora o *ensembling* possa levar ao aumento da complexidade computacional, este tipo de abordagem melhora o desempenho do modelo. Existem inúmeras formas de combinar modelos, várias variantes e abordagens. Habitualmente destacam-se três abordagens mais populares [108], [109]: *bagging*; *boosting*; e *stacking*.

O "ensacamento" (em inglês, *bagging* [110]), é atualmente um conceito de agregação de modelos classificadores tipicamente homogéneos (mesma arquitetura) com diversidade entre si. A diversidade é normalmente introduzida a partir de amostras diferentes do conjunto de dados. Para isso combina dois ou mais classificadores de forma paralela oriundos de modelos independentes, assim é possível obter diferentes "vistas" do problema e conseguir um resultado combinado que supere o desempenho que seria obtido utilizando apenas um modelo, como mostra a Fig.3.15. A combinação dos classificadores pode ser realizada de muitas formas. É comum em problemas de classificação utilizar-se o voto maioritário enquanto que, em problemas de regressão é comum utilizar a média. Este processo pode ser realizado de forma paralela, reduzindo o tempo de treino [108, 109].

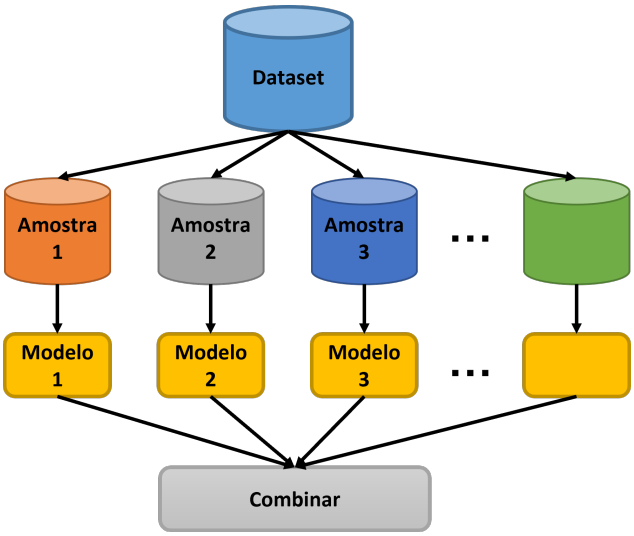


Figura 3.15: Bagging.

A técnica "Impulsão" (em inglês, *boosting*), é um método sequencial de combinação de modelos com o objetivo de corrigir erros de previsão melhorando o desempenho final. Este método utiliza normalmente modelos com a mesma arquitetura dispostos de maneira sequencial, como mostra a Fig.3.16, para que o erro se propague e possa ser atenuado/corrigido pelo modelo seguinte. Inicialmente temos apenas um determinado modelo treinado de forma tradicional, depois o modelo seguinte é ajustado com a informação errada do modelo anterior e assim sucessivamente [111]. Esta passagem de informação errada pode ser fornecida ao modelo seguinte, por exemplo, ajustando o *dataset* - treinando apenas com os dados errados pelo modelo anterior, ou mais usualmente através de um algoritmo de ponderação que seleciona os dados de forma "aleatória", mas com mais probabilidade para os dados errados do modelo anterior mantendo o *dataset* original intacto. Desta forma o foco dos modelos seguintes é ajustado aos erros do modelo anterior, conseguindo-se melhorar gradualmente as previsões. O resultado final é obtido pela combinação dos resultados de todos os modelos através de uma média ponderada [108, 109].

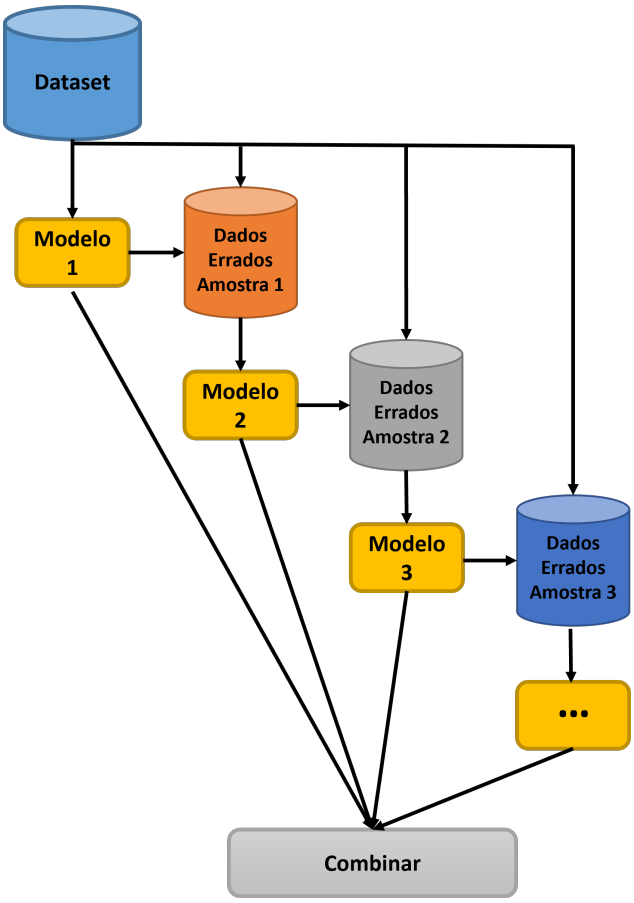


Figura 3.16: Boosting.

O "empilhamento" (em inglês, *stacking* [112]), ilustrado na Fig.3.17, é um conceito de agregação de modelos tipicamente heterogéneos. Habitualmente a diversidade necessária para obter bons resultados é assegurada por diferentes arquiteturas nos modelos a agregar (modelos-base), no entanto, é possível utilizar também variações no *dataset*. Ao contrário dos métodos anteriores que utilizam o cálculo de métricas para a decisão final, em *Stacking* é utilizado um algoritmo de aprendizagem, por exemplo, uma rede neuronal ou regressão linear, para combinar os modelos-base e produzir a classificação final. Existem diversas formas de explorar o conceito *stacking*, existindo a possibilidade de empilhar mais modelos colocando camadas intermédias vertical e horizontalmente [108, 109].

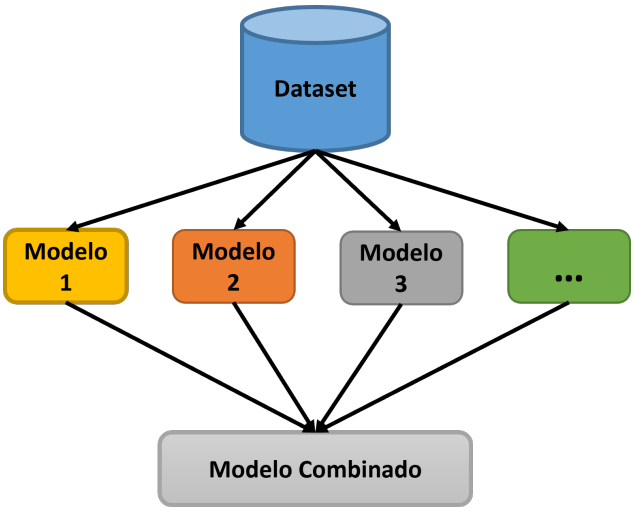


Figura 3.17: Stacking.

3.5 PESQUISA DE HIPERPARÂMETROS

A arquitetura de uma rede neuronal é composta por inúmeros parâmetros. A maioria dos parâmetros são habitualmente designados "pesos" e são aprendidos e ajustados pelo modelo durante o treino num processo totalmente automático, com objetivo de minimizar o erro e obter melhor desempenho. Já os hiperparâmetros são parâmetros que não podem ser estimados pelo modelo, são responsáveis por controlar todo o processo de treino, definindo todas as características do modelo e atuando como variáveis de configuração [113]. Por exemplo, selecionam a arquitetura do modelo, definem o número de camadas ocultas, as dimensões do *kernel*, número de filtros, número de neurónios, taxa de aprendizagem, etc. Um suma, os parâmetros especificam como o modelo transforma determinados dados de entrada na saída desejada, enquanto os hiperparâmetros definem como o modelo é estruturado e treinado.

A pesquisa de hiperparâmetros (também conhecida por *hyperTuning*), é um processo muito importante que tem como objetivo determinar a combinação ideal de hiperparâmetros que maximiza o desempenho do modelo [114, 115]. Idealmente, para encontrar a melhor solução deveriam ser testadas todas as combinações de hiperparâmetros. No entanto, este processo seria extremamente demorado e computacionalmente muito exigente - seria necessário realizar todo o processo de treino exaustivamente para cada combinação [116]. Por este motivo, a escolha de hiperparâmetros é geralmente realizada de forma manual por experimentação ou com base em conhecimento adquirido anteriormente [117].

Este processo iterativo, complexo, computacionalmente exigente e demorado, pode ser simplificado usando diferentes técnicas de otimização automatizadas. Embora

seja um processo automático, é sempre necessário definir um espaço de pesquisa para cada hiperparâmetro.

3.5.1 *Grid Search*

O método de pesquisa em grelha (em inglês, *grid search*), é um algoritmo exaustivo e demorado que cria uma grelha de pesquisa com base nos hiperparâmetros definidos. Para cada combinação é construido e executado o modelo e registado o desempenho, no final são retornados o conjunto de hiperparâmetros que produziram o melhor resultado. O número de configurações a ser testada cresce exponencialmente com o número de hiperparâmetros. No entanto, o método *grid search* pode ser paralelizado, pois não existe relação entre os vários testes realizados, sendo assim possível reduzir o tempo de pesquisa (embora aumentando o custo computacional) [116].

3.5.2 *Random Search*

A pesquisa aleatória (em inglês, *random search* [118]), é um processo semelhante à *grid search*, onde a pesquisa é feita de forma aleatória. Desta forma torna-se temporalmente mais eficiente, pois não percorre todas as combinações. A pesquisa é limitada por um intervalo temporal ou por um número de execuções e como tal, se o espaço de pesquisa for muito grande, não é garantido que seja encontrado resultado ideal. Tal como em *grid search* é registado o desempenho do modelo a cada teste de forma independente, é um processo passível de paralelização [114].

3.5.3 *Bayesian Optimization*

Contrariamente aos métodos anteriores que apenas registam o desempenho final de cada teste, a otimização Bayesiana (em inglês, *Bayesian optimization*) [119], considera as iterações anteriores na escolha dos hiperparâmetros do próximo teste. Este é por isso um método complexo e difícil de implementar. Este método substitui a função objetivo por outra de menor custo e menor complexidade, depois utiliza a interpretação Bayesiana de probabilidade, para avaliar e encontrar a próxima combinação de hiperparâmetros a utilizar de modo a minimizar ou maximizar a função objetivo. É bastante eficiente para ajustar poucos hiperparâmetros, mas para muitos torna-se semelhante ao método *random search*. Isto deve-se ao facto de na otimização Bayesiana os testes serem realizados de forma sequencial, ou seja, as iterações seguintes estão dependentes da anterior [116].

3.5.4 *Hyperband*

A técnica hiperbanda (em inglês, *hyperband*) [120, 121], é implementada com o objetivo de acelerar a pesquisa aleatória. Para isso este método recorre a duas funções cíclicas, o ciclo interno e o externo. O ciclo interno utiliza o método *Successive Halving Algorithm* (SHA) [122] como rotina para realizar paragem antecipada do teste/treino, acelerando a pesquisa. O algoritmo SHA identifica e encerra configurações de hiperparâmetros que apresentam baixo desempenho, sem necessitar de treinar totalmente e avaliar individualmente cada configuração de hiperparâmetros. A cada ciclo é mantida a melhor metade das configurações e descartada a outra metade que tem pior desempenho. O ciclo externo tem como objetivo encontrar as configurações ótimas que minimizem a função objetivo, avaliando cada hiperparâmetro proveniente do ciclo anterior [115]. Este método é computacionalmente adaptável com base no seu processo de seleção e foca a maioria dos seus recursos nos resultados mais promissores. O início da pesquisa *SHA* é considerado um pouco agressivo, como é possível observar na Fig.3.18 e pode provocar a eliminação precoce de algumas configurações que poderiam ser relevantes.

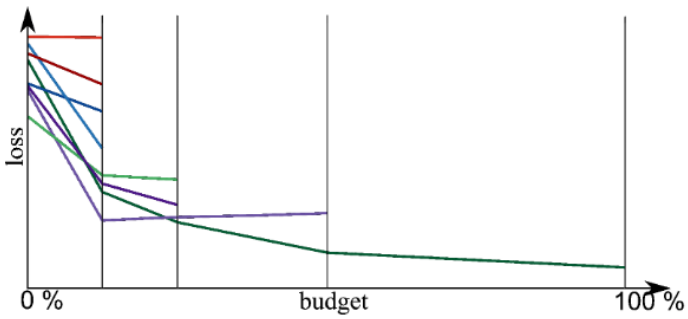


Figura 3.18: Exemplo SHA [123].

3.6 TÉCNICAS DE COMPRESSÃO DE MODELOS

Para alcançar bons desempenhos são habitualmente utilizadas redes de grande dimensão, que comportam milhões de parâmetros. No entanto, estas redes necessitam de elevado poder computacional o que nem sempre é possível.

A utilização de redes neuronais em dispositivos MCU impõe grandes limitações ao desenvolvimento, é exigido um grau de otimização e adaptação bastante elevado dos modelos. Os recursos de processamento e armazenamento de equipamentos MCU são restritos, o que dificulta a alocação do modelo e pode provocar atrasos na inferência. A procura em comprimir grandes modelos com o prejuízo mínimo de desempenho é um dos principais focos do desenvolvimento de TinyML. Técnicas de redução como *Quantização*, *Pruning* e *Knowledge Distillation*, além de auxiliarem no

combate ao *overfitting* podem reduzir imenso os recursos computacionais necessários para acomodar o modelo à custa de uma pequena perda de desempenho.

3.6.1 *Quantização*

A quantização é o processo de converter um intervalo contínuo de grande dimensão num intervalo discreto bastante menor. Consequentemente é possível reduzir o número de bits utilizados na representação de uma determinada grandeza. Através desta técnica é possível representar pesos e/ou ativações utilizando um menor número de bits, reduzindo a memória, largura de banda e energia à custa de uma pequena perda de exatidão da rede. Redes neuronais, pesos e ativações são geralmente representados com números de vírgula flutuante com 32 bits. Utilizando, por exemplo, uma quantização de 8 bits é possível reduzir em teoria o armazenamento necessário cerca de 4x. Por outro lado, cálculos realizados com números inteiros exigem menos recursos de processamento comparativamente aos cálculos com vírgula flutuante, conseguindo-se reduzir os tempos de atraso na inferência.

A quantização pode ser aplicada a pesos e ativações, de forma diferente, isto é, não é necessário quantizar todo o modelo com o mesmo numero de bits. Esta técnica pode ser implementada durante o treino ou posteriormente (forma tradicional). Normalmente a quantização é mais eficaz quando aplicada durante o treino, no entanto também é um processo mais complexo não justificando por vezes a sua utilização [124]. O processo de quantização pode ser automatizado através de uma estrutura de pesquisa (*framework*) que tem como objetivo encontrar a melhor quantização a empregar num determinado dispositivo. O método proposto por K. Wang *et al.* [125] é exemplo disso - utiliza uma rede neuronal de aprendizagem por reforço para conseguir definir a quantização a aplicar cumprindo os requisitos de *hardware* [126]. Para a maioria das aplicações, uma quantização realizada após o treino é o suficiente para acomodar o modelo aos requisitos computacionais, apresentando prejuízos mínimos.

3.6.2 *Poda*

A poda ou (em inglês, *pruning*) [127], é uma técnica de redução física do modelo, através de supressão de alguns dos seus elementos constituintes considerando o menor prejuízo possível para o desempenho. Assim é possível reduzir os requisitos computacionais e de armazenamento da rede, bem como a sua latência. Habitualmente a rede é submetida a treino após o *pruning*, utilizando um pequeno subconjunto do *dataset*, para reduzir o seu impacto negativo [126].

As principais formas de *pruning* [128] são:

1. *Pruning* de pesos e/ou ativações;

   O *pruning* de pesos e/ou ativações, é o método mais simples de poda. Habitualmente é utilizada a técnica de poda pelo critério magnitude que "remove" conexões individuais entre as camadas da rede neuronal com menor importância (menor magnitude) após cada época de treino, aumentando assim a esparsidade da rede (número de elementos a zero). São também utilizados algoritmos iterativos de *Pruning* em que é definida uma esparsidade alvo e o modelo realiza vários ciclos de ajuste fino até atingir o alvo, o que permite obter ainda melhores resultados. A redução das conexões individuais para zero é aproveitada pelas ferramentas de compressão reduzindo o tamanho do modelo. No entanto, durante a inferência, a maioria dos dispositivos *Edge* não consegue aproveitar a representação esparsa dos pesos, executando operações de multiplicação por zero em cada elemento removido.

2. *Pruning* de estrutura ou canal;

   O *pruning* de estrutura ou canal, ocorre quando são removidos filtros de camadas convolucionais ou neurónios de camadas totalmente conectadas. Com menos operações consegue-se reduzir a memória necessária para implementar o modelo e tornar as inferências mais rápidas. O fluxo geral do algoritmo é muito semelhante ao *pruning* de pesos e/ou ativações, no entanto, ao selecionar um filtro ou neurónio este tem influência sobre as camadas a jusante, implicando uma redução maior que o método anterior. Este método tem um maior prejuízo no desempenho que o método anterior, mas os dispositivos MCU conseguem aproveitar o *Pruning* estrutural, pois os elementos removidos perdem todas as conexões que dependem de si, sendo removidos na totalidade (eliminando os cálculos e operações).

3. *Pruning "inteligente"*.

   O *pruning "inteligente"* é um conceito direcionado ao *hardware* de um dispositivo. Assim é possível otimizar o modelo com base em métricas consideradas relevantes em TinyML como memória, tempos de atraso, exatidão e consumo de energia. Exemplos de algoritmos deste género são: *NetAdapt* [129] e *once-for-all* [130].

A Fig.3.19 ilustra a aplicação destas técnicas em redes neuronais.

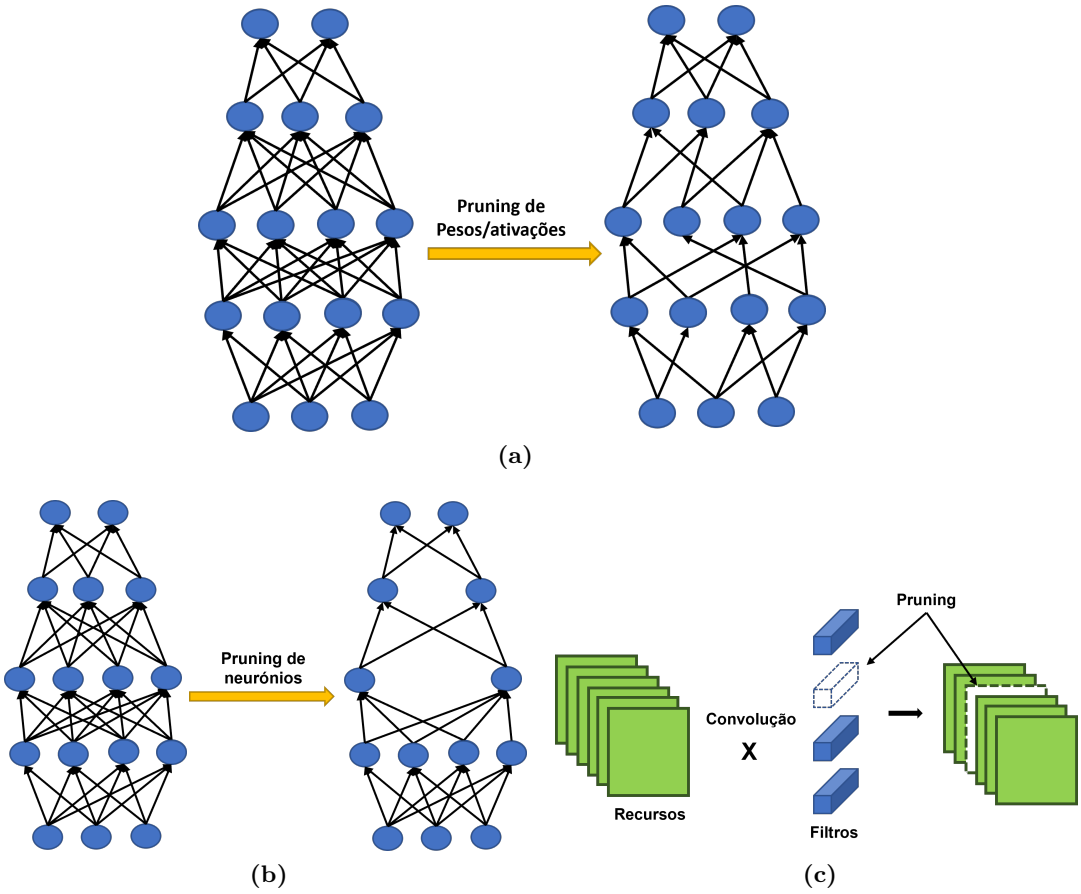


Figura 3.19: Exemplos de pruning: (a) Pruning de pesos e/ou ativações; (b) Pruning de estrutura ou canal numa rede *fully connected*; (c) Pruning de canal numa rede CNN;.

3.6.3 *Destilação de conhecimento*

A destilação de conhecimento, (em inglês, *Knowledge Distillation* (KD) [131]), é uma técnica de redução que permite que um modelo de pequena dimensão e baixa complexidade (designado estudante) aprenda características semelhantes a outro modelo de maior complexidade (designado professor). Esta técnica para além de permitir libertar espaço no armazenamento e diminuir a latência, atua como regularizador, melhorando a generalização e permitindo bons desempenhos. Embora os modelos professor tenham maior capacidade para adquirir conhecimento por utilizarem redes neuronais mais profundas e complexas, esta capacidade não pode ser utilizada na totalidade [132]. Contrariamente, o aluno atinge um desempenho que dificilmente atingiria se treinado da forma tradicional. Através de KD, é possível retirar muito mais "desempenho vs. recursos" do modelo estudante, proporcionalmente ao conseguido com professor. O modelo professor, tem de ser treinado previamente de forma a alcançar bom desempenho, antes de se iniciar a passagem de conhecimento. Durante o KD é calculada uma métrica de divergência normalmente com base na entropia cruzada de cada modelo. Essa métrica é responsável pela retro-propagação

do conhecimento, "forçando" o modelo estudante a imitar o professor durante do treino, tal como ilustrado na Fig.3.20.

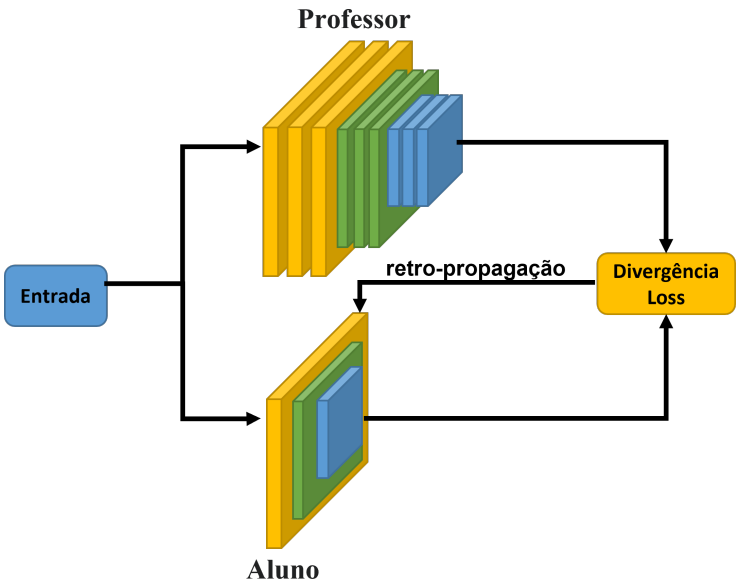


Figura 3.20: Esquema destilação de conhecimento.

4

PROBLEMA DESAFIO

Neste capítulo é apresentado o desafio da DCASE que permitiu colocar em prática o conhecimento adquirido no decorrer desta dissertação. Este desafio está perfeitamente alinhado com os objetivos da dissertação - investigar e desenvolver algoritmos de DL com a intenção de os preparar para serem aplicados em dispositivos de baixa capacidade computacional e de recursos limitados, tais como MCUs.

4.1 DESAFIO

A comunidade de pesquisa, deteção e classificação de cenas, eventos acústicos DCASE [65], disponibiliza todos os anos um conjunto de desafios que atraem vários investigadores a nível internacional. Nos últimos dois anos participaram alunos e investigadores de diversos laboratórios e universidades de todo mundo em países como *e.g.*, China, Coreia do Sul, Áustria, Japão, Espanha, Reino Unido, Estados Unidos, Alemanha, México, Itália entre outros. Os desafios DCASE têm também despertado o interesse de várias empresas relevantes levando as suas equipas de investigação a participar na competição, tais como *e.g.*, Qualcomm AI, Intel Labs, Hitachi Ltd, Xiaomi, valeo.ai, 3M, entre outras.

O desafio da edição *DCASE-2022* mais alinhado com os objetivos desta dissertação designa-se, em inglês, por *"Task 1, Low-Complexity Acoustic Scene Classification"*, e tem como objetivo desenvolver e aperfeiçoar um sistema de classificação de cenas acústicas de baixa complexidade ASC, destinado a MCUs, a Fig. 4.1 apresenta uma visão superficial do desafio. Para isso é necessário desenvolver um modelo capaz de classificar gravações áudio consoante o ambiente acústico onde são adquiridas, respeitando os limites de complexidade do modelo definidos para a tarefa. Além disso, o sistema tem de apresentar boa capacidade de aprendizagem e generalização, apresentando bom desempenho também na presença de novos dados mesmo que oriundos de outros dispositivos. A métrica utilizada pelo desafio para avaliação do desempenho é *Log loss*, a diminuição *Log loss* significa uma melhoria de desempenho.

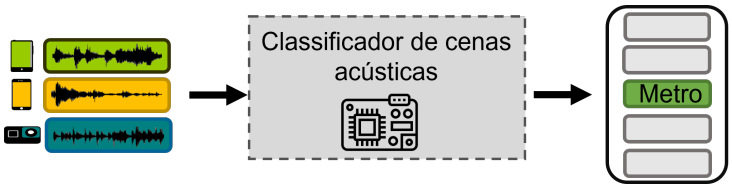


Figura 4.1: Visão superficial do desafio *Task 1*.

4.2 CONJUNTO DE DADOS

O conjunto de dados (*dataset*) utilizado para o desenvolvimento desta tarefa designa-se *TAU Urban Acoustic Scenes 2022 Mobile* [133,134]. Este é composto por gravações de 10 cenas acústicas de ambientes diferentes, descritos na Tabela. 4.1, capturadas em 12 cidades europeias através de 4 dispositivos reais (A,B,C,D). Contém ainda 11 dispositivos gerados sinteticamente (S1-S11) a partir de dados reais do dispositivo A [65]. O conjunto de dados tem na totalidade cerca 230 mil trechos de 1s, totalizando aproximadamente 64 horas, é fornecido a uma taxa de amostragem de 48kHz de 24 bits, em canal único.

Tabela 4.1: Cenas acústicas do *dataset TAU Urban Acoustic Scenes 2022 Mobile*.

Cena acústica	Rótulos
Aeroporto	*airport*
Centro comercial	*shopping_mall*
Estação de metro	*metro_station*
Rua de pedestres	*street_pedestrian*
Praça pública	*public_square*
Rua movimentada	*street_traffic*
Viagem de comboio	*tram*
Viagem de autocarro	*bus*
Viagem de metro	*metro*
Parque urbano	*park*

4.2.1 *Divisão do conjunto de dados*

O conjunto de dados é dividido pelos organizadores do desafio em dois conjuntos, o de desenvolvimento e o de avaliação. O *dataset* de avaliação não é fornecido aos participantes e serve unicamente para os organizadores avaliarem os trabalhos submetidos. Este contém vários dados provenientes de dispositivos "novos" que não estão contemplados no conjunto de desenvolvimento.

O *dataset* de desenvolvimento destina-se à resolução da tarefa e está subdividido da seguinte forma: 70% das amostras são destinadas ao processo de treino e validação,

os restantes 30% destinam-se a testes após o treino. Os dados de treino e validação são 139620 amostras, distribuídas como mostra a Tabela. 4.2, foram divididos usando os mesmos pesos (70% para o treino e 30% para o processo de validação, realizado durante o treino). Pode-se verificar que este conjunto não é equilibrado - o dispositivo "A" tem bastante mais amostras que os restantes. No entanto, ao nível de cenas acústicas (classes) está próximo do equilíbrio.

Tabela 4.2: Dados de treino e validação.

	Dispositivos						
Rótulos	A	B	C	S1	S2	S3	Total
airport	10190	750	740	750	750	750	13930
bus	10250	750	750	750	750	750	14000
metro	10070	750	750	750	750	750	13820
metro_station	10050	750	750	750	750	750	13800
park	10540	750	750	750	750	750	14290
public_square	10530	740	750	750	750	750	14270
shopping_mall	9990	750	740	750	750	750	13730
street_pedestrian	10110	750	750	750	750	750	13860
street_traffic	10380	750	750	750	750	750	14130
tram	10040	750	750	750	750	750	13790
Total	102150	7490	7480	7500	7500	7500	139620

Os dados de teste são compostos por 29680 amostras e estão perfeitamente distribuídos tanto em termos de dispositivos como pelas classes, como mostra a Tabela. 4.3. Comparativamente ao conjunto de desenvolvimento, tem mais 3 dispositivos sintéticos, sendo por isso o conjunto ideal para avaliação do modelo. Embora este conjunto possa também ser utilizado para treino, há que ter em atenção que isso aumenta o risco de *overfitting*.

Tabela 4.3: Dados de teste.

	Dispositivos									
Rótulos	A	B	C	S1	S2	S3	S4	S5	S6	Total
airport	330	320	330	330	330	330	330	330	330	2960
bus	330	330	330	330	330	330	330	330	330	2970
metro	330	330	330	330	330	330	330	330	330	2970
metro_station	330	330	330	330	330	330	330	330	330	2970
park	330	330	330	330	330	330	330	330	330	2970
public_square	330	330	330	330	330	330	330	330	330	2970
shopping_mall	330	330	330	330	330	330	330	330	330	2970
street_pedestrian	330	330	330	330	330	330	330	330	330	2970
street_traffic	330	330	330	330	330	330	330	330	330	2970
tram	330	330	320	330	330	330	330	330	330	2960
Total	3300	3290	3290	3300	3300	3300	3300	3300	3300	29680

Os espetrogramas de mel utilizados para alimentar o modelo *baseline* são adquiridos a partir de um sinal com frequência de amostragem de $44.1kHz$, é utilizada uma janela de pesquisa de 40ms e sobreposição a 50%.

As figuras 4.4, 4.2 e 4.3, são exemplos de espectrogramas obtidos para o dispositivo real A em cada uma das dez classes. É possível observar que a maioria dos ambientes que conhecemos como tipicamente mais ruidosos (aeroporto, metro, autocarro, ruas movimentadas), também o são para o conjunto de dados, apresentam um sinal com mais intensidade, sobretudo em baixas frequências abaixo de $4kHz$. À primeira vista parece relativamente fácil para os exemplos apresentados adivinhar o ambiente acústico em questão.

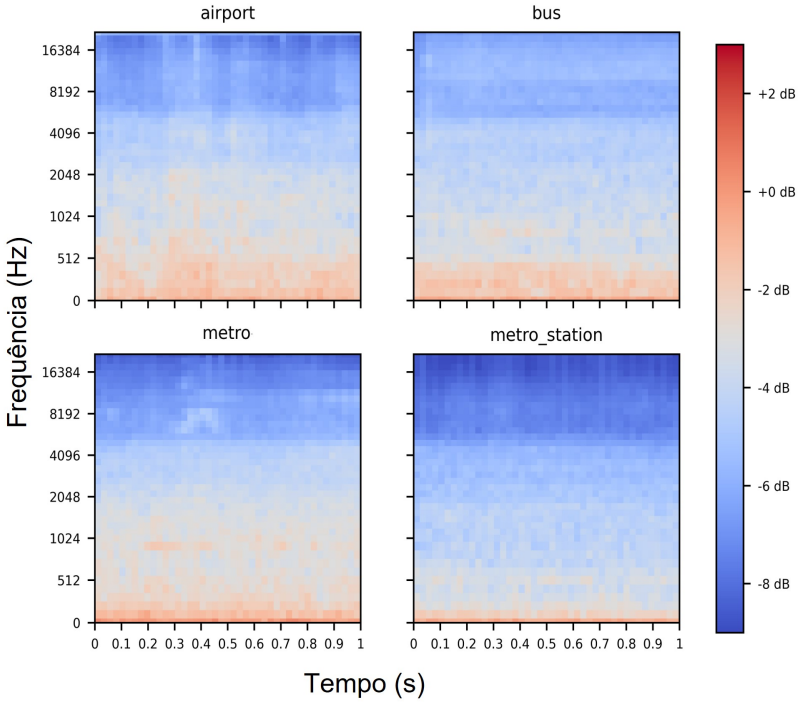


Figura 4.2: Espectrograma de mel 44.1kHz - airport, bus, metro, metro_station - dispositivo A

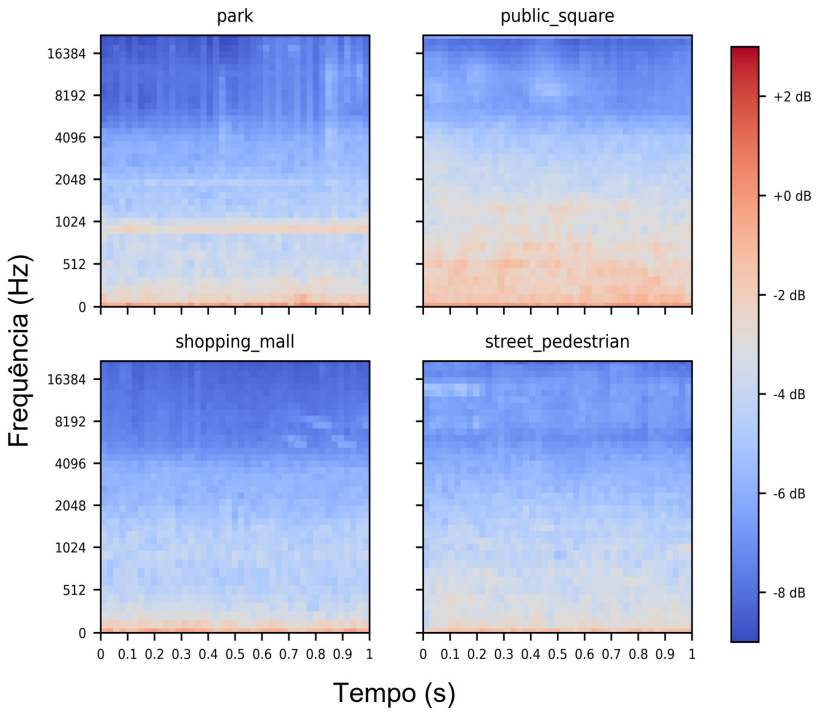


Figura 4.3: Espectrograma de mel 44.1kHz - park, public_square, shopping_mall, street_pedestrian - dispositivo A

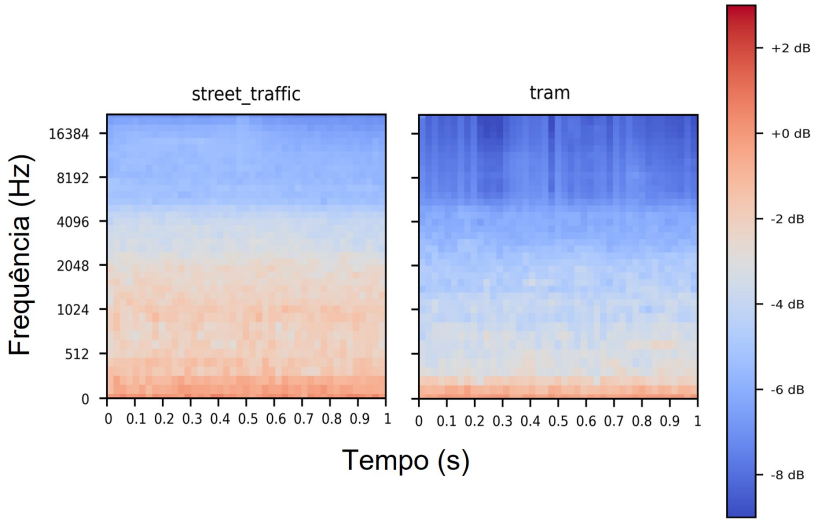


Figura 4.4: Espectrograma de mel 44.1kHz - street_traffic, tram - dispositivo A

A Fig. 4.5, permite observar espectrogramas temporalmente consecutivos adquiridos com o dispositivo A para a classe autocarro, verificamos que existem grandes variações ao longo dos 4s apresentados.

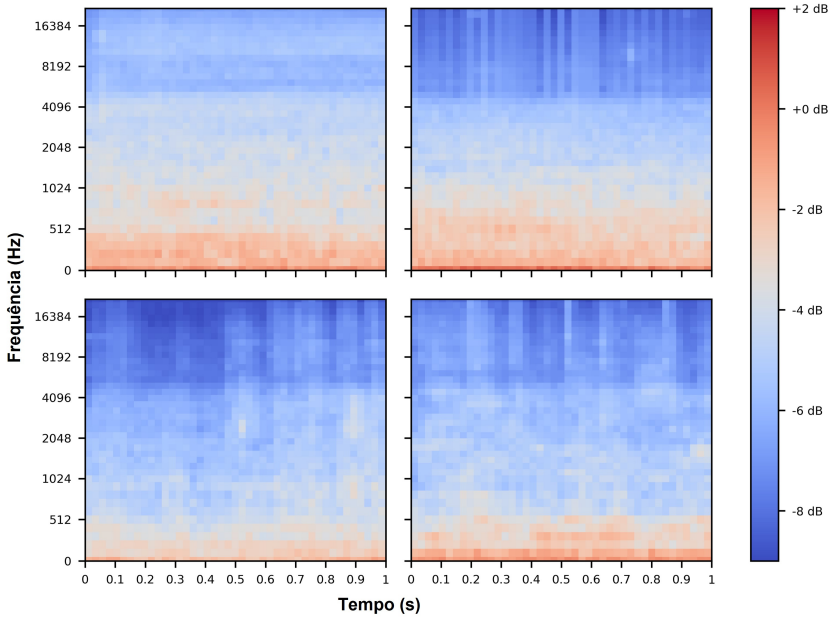


Figura 4.5: Espectrograma de mel 44.1kHz - bus - dispositivo A - instantes temporais consecutivos - Barcelona.

Na Fig. 4.6, são apresentados espectrogramas para o mesmo instante temporal de vários dispositivos (A, B, C, S1) também para a classe autocarro, é possível visualizar que existem diferenças consideráveis entre os vários dispositivos, embora o sinal pareça semelhante a intensidade de sinal apresenta algumas variações em cada dispositivo.

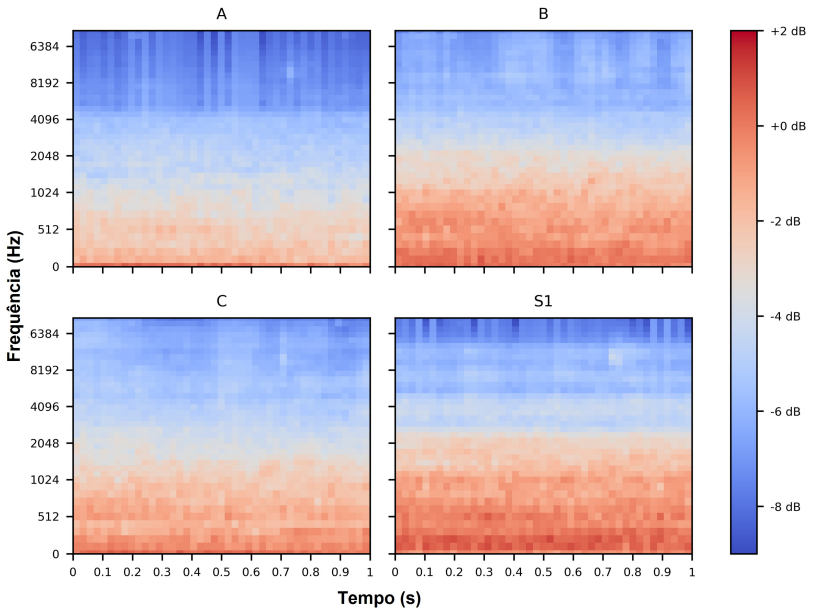


Figura 4.6: Espectrograma de mel 44.1kHz - bus - dispositivo A - mesmo instante temporal - Barcelona.

Por fim, na Fig. 4.7, são apresentados espectrogramas adquiridos pelo dispositivo A para a classe autocarro em cidades diferentes, verificam-se variações de sinal, não é visualmente possível encontrar padrões em comum.

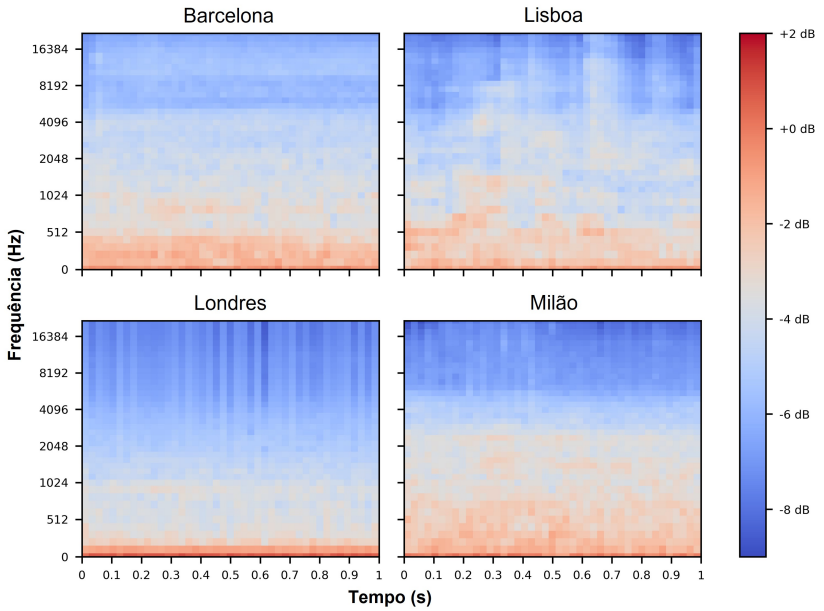


Figura 4.7: Espectrograma de mel 44.1kHz - bus - dispositivo A - diferentes cidades.

Com base nos espectrogramas de mel apresentados anteriormente, é possível concluir que a classificação de cenas acústicas para o conjunto de dados *TAU Urban Acoustic Scenes 2022 Mobile*, é um processo extremamente difícil complexo. Existe uma enorme disparidade e diversidade de espectrogramas possíveis mesmo tendo em comum a mesma cena acústica.

4.3 REQUISITOS DE COMPLEXIDADE

O desafio coloca algumas limitações de complexidade cujo objetivo é reproduzir limites computacionais típicos de MCUs. O limite máximo de parâmetros que o modelo pode apresentar é 128k (incluindo parâmetros de valor zero), é obrigatório a utilização de variáveis inteiras de 8 bits (quantização). É também limitado o número máximo de operações de multiplicação-acumulação (em inglês, MACs) de 30 MMACs (milhões de MACs) por inferência, com o objetivo de representar um limite de memória SRAM.

4.4 MODELO BASE

O desafio fornece um modelo base (em inglês, *baseline*) que serve como modelo comparativo ou ponto de partida para o desenvolvimento. A *baseline* é um modelo de arquitetura CNN, alimentado por espectrogramas logarítmicos de mel de formato (40,51), como mostra a Fig.5.3. Estes espectrogramas são adquiridos a partir de áudios de 1s, utilizando uma STFTs de 2048 pontos, uma janela deslizante de 40ms e sobreposição a 50%. É também utilizado um banco de 40 filtros de mel.

O modelo de classificação é constituído por três camadas CNN, responsáveis pela extração de recursos/padrões e uma camada totalmente conectada responsável pela atribuição de rótulos. A Tabela. 4.4 detalha toda a estrutura da rede bem como as operações utilizadas em cada camada. As camadas convolucionais são compostas essencialmente por operações de convolução (*Conv2d*), normalização em lote (*Batch Normalization*), função de ativação *ReLU*, agrupamento máximo (*max pooling*) e *dropout*. A parte totalmente conectada de classificação (*FC*) é composta essencialmente por uma camada *flatten* (responsável por remodelar os recursos para um formato vetorial), uma camada densa (*dense*), com ativação *ReLU* e *dropout*. Por fim a classificação realizada através de uma camada *dense* com ativação *sofmax* para que se obtenha a previsão num formato probabilidade.

O modelo é treinado por 200 épocas utilizando o otimizador *ADAM* com uma taxa de aprendizagem de $10e^{-3}$. O desempenho é avaliado a cada época segundo o conjunto de validação e escolhido o modelo com melhor desempenho segundo a métrica *Log loss* (Loss). É também calculada a *accuracy* (ACC) embora esta não seja determinante para a avaliação das proposta submetidas pelos participantes no desafio. Os resultados oficiais obtidos pela *baseline* podem ser visualizados na Tabela. 4.5.

Tabela 4.4: Baseline.

Camada	Detalhes	
CNN1	Conv2d	filtros: 16, kernel size: 7
CNN2	Conv2d	filtros: 16, kernel size: 7
	2D max pooling	pool size: (5, 5)
	Dropout	taxa: 30%
CNN3	Conv2d	filtros: 32, kernel size: 7
	2D max pooling	pool size: (4, 10)
	Dropout	taxa: 30%
FC	Flatten	
	Dense	neurónios: 100
	Dropout	taxa: 30%
Saída	Dense	neurónios: 10
	softmax activation	

Todas as Camadas CNN utilizam as operações *Batch normalization* e *ReLu activation*.

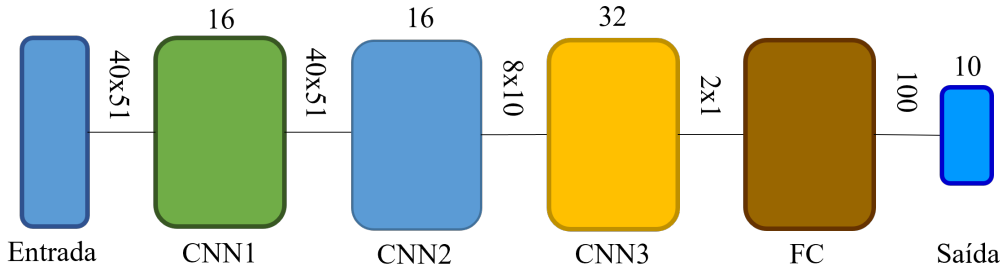


Figura 4.8: Esquema de arquitetura Baseline.

Tabela 4.5: Resultados oficiais *baseline* DCASE-2022.

Acc	LOSS	Parâmetros	MACS
42.9 (+/-0.770)	1.575 (+/-0.018)	46.512k	29.23M

Esta página foi propositadamente deixada em branco.

5

TRABALHO REALIZADO E RESULTADOS EXPERIMENTAIS

A maioria das abordagens DCASE identificadas na secção 2.4 utiliza variações de modelos CNN e caminhos identidade ResNet. O desafio *DCASE-Task1* é destinado a equipamentos de baixa complexidade e portanto, deve ser bastante otimizado e ajustado para que o recursos despendidos tenham o máximo aproveitamento. Neste trabalho foi dedicado maior esforço na otimização de toda a envolvente do modelo partindo da *baseline* fornecida. Foi realizada uma pesquisa de hiperparâmetros que permitiu de forma automatizada determinar as características do espectrograma que beneficiam a tarefa em causa adaptando o sinal de entrada ao modelo. São propostas três abordagens que recorrem a técnicas de otimização, redução e aprendizagem na tentativa de aumentar a capacidade de aprendizagem e desempenho do modelo. As técnicas utilizadas podem ser aplicadas aos mais variados modelos, possibilitando a criação futura de algoritmos genéricos de adaptação automática de modelos, ajustando arquiteturas comuns a aplicações TinyML, tendo em conta os requisitos de complexidade.

Em 2021, apesar de o desafio *DCASE2021-Task1a* ser objetivamente semelhante ao *DCASE2022-Task1*, existem algumas diferenças relevantes:

- O *Dataset* apesar de ser o mesmo, em 2021 era fornecido em trechos de 10s ao invés 1s em 2022.
- O requisito máximo de parâmetros em 2021 era limitado a 128K parâmetros diferentes de zero, enquanto em 2022 o limite é para todos os parâmetros.
- Também não existia um limite máximo de operações MACs em 2021, em oposição aos 30MMAC exigidos em 2022.
- Em 2021 a utilização de quantização era livre e opcional, enquanto que em 2022 é obrigatório quantizar todo o modelo em inteiros de 8bits (INT8).

Atendendo as diferenças verificadas entre desafios de 2021 e 2022, são esperados resultados inferiores em 2022 comparativamente aos registados na tarefa *DCASE2021-Task1a*. A diminuição do sinal de entrada é a maior exigência de limitações de complexidade, aumentam de forma clara a dificuldade do desafio, exigindo dos participantes modelos mais compactos e robustos.

Serve o presente capítulo, para apresentar o trabalho de investigação desenvolvido, os resultados obtidos e respetivas conclusões. A abordagem para o desenvolvimento e avaliação dos modelos encontra-se apresentada na Fig. 5.1.

A métrica de avaliação utilizada nesta dissertação é também a escolhida para avaliar o desempenho dos modelos submetidos durante o desafio DCASE. A entropia cruzada *Log loss* (loss), durante a análise dos resultados obtidos a expressão melhor desempenho pressupõe a diminuição do loss.

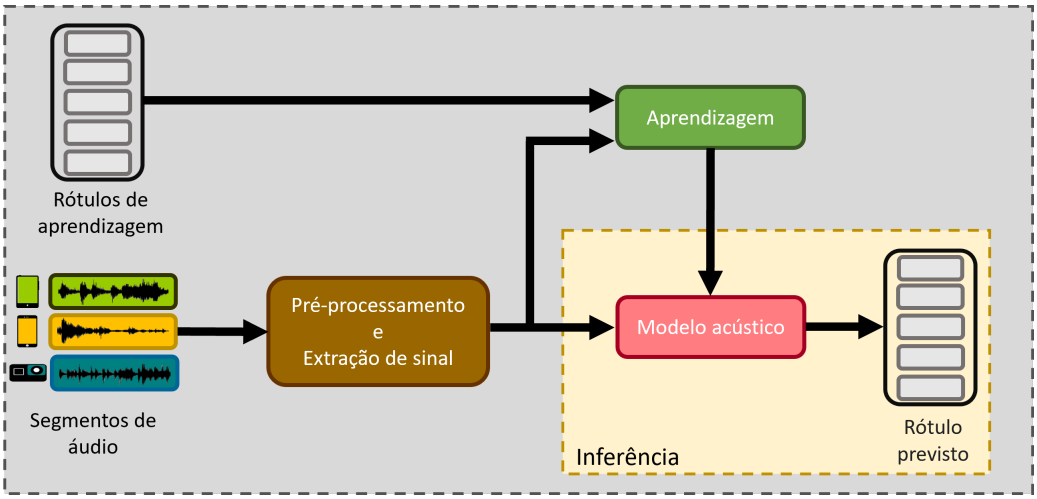


Figura 5.1: Visão geral de um sistema de classificação de cenas Acústicas.

5.1 BASELINE

O processo de desenvolvimento foi iniciado com a realização de algumas experiências com o modelo *baseline*, procurando encontrar pequenas características que pudessem favorecer o desempenho do mesmo: a escolha do tamanho de lote; a percentagem de dados a utilizar na quantização; e a aplicação de variantes na técnica *label smoothing*.

O tamanho do lote (em inglês, *batch size*) é um hiperparâmetro importante, responsável por definir a quantidade de amostras do conjunto de dados que são utilizadas a cada época de treino. Este parâmetro tem impacto tanto no desempenho como na velocidade de treino. É possível utilizar a totalidade do *dataset* em cada época de treino, no entanto, num *dataset* de grande dimensão seria necessária uma grande quantidade de memória para acumular os erros de milhares de amostras. Utilizou-se então mini-lotes definindo um baixo valor de *batch size*, reduzindo assim os requisitos de capacidade computacional durante o treino [135]. Ao definir o *batch size* existe possibilidade de a função perda não atingir o mínimo absoluto durante o processo de otimização, pelo que a forma mais eficaz de determinar o hiperparâmetro adequado é através experimentação. A Tabela. 5.1 mostra as métricas de desempenho obtidas para diferentes tamanhos de lote usados no treino.

Verifica-se que o *batch size* que produziu melhores resultados para o teste realizado foi *batch size* = 64.

Tabela 5.1: Desempenho da *baseline* para diferentes tamanhos de lote.

Batch Size	Acc	Loss
32	42.2	1.579
64	42.9	1.543
128	41.9	1.587
256	42.7	1.565

O processo de quantização apresenta na maioria das aplicações um prejuízo mínimo no desempenho. Para o desafio em questão é exigida quantização de 8 bits para todo o modelo. A quantização pode ser realizada durante ou após o processo de treino. No entanto, o processo de implementação durante o treino é bastante complexo de implementar, por esta razão foi preferível implementar a quantização após o treino. A quantização necessita de ser calibrada ao conjunto dados, para isso é necessária uma porção representativa do *dataset*, onde o modelo será avaliado sendo calculada a escala de quantização. A percentagem de *dataset* utilizada neste processo influencía o desempenho do modelo, como tal foram realizados alguns testes cujos resultados podem ser observados na Tabela. 5.2. Através destes resultados, conclui-se que a percentagem de *dataset* utilizada para calibração que favorece o desempenho é 50%. A utilização de uma maior percentagem de *dataset* provoca dificuldade na generalização de novos dados, resultando na redução de desempenho verificada.

Tabela 5.2: Desempenho da *baseline* para diferentes níveis de quantização.

Dataset (%)	Acc	Loss
25%	42.7	1.601
50%	42.9	1.543
75%	42.8	1.551
90%	42.75	1.560

Foi também verificado o impacto da suavização de rótulos (*label smoothing*), esta técnica atua reduzindo os índices de confiança do modelo, melhorando a generalização. Esta técnica foi testada para diferentes fatores de incerteza e o desempenho da rede avaliado. Através dos resultados apresentados na Tabela. 5.3, pode-se observar que o fator de incerteza que consegue melhor desempenho é $fator = 0{,}1$, verifica-se a utilidade da técnica de suavização de rótulos conseguindo melhorias significativas quando utilizada.

Tabela 5.3: Desempenho da *baseline* para diferentes parâmetros de suavização de rótulos.

Factor	Acc	Loss
✗	41.69	1.837
0,05	42.6	1.588
0,1 (original)	42.9	1.543
0,15	41.3	1.613
0,2	40.6	1.643

A Tabela 5.4, permite analisar os resultados de cada dispositivo perante cada uma das classes, facilitando a identificação de problemas de generalização. Verifica-se que os dispositivos reais (A,B,C) apresentam melhor desempenho (loss inferior) comparativamente aos que os dispositivos simulados (s1-s6). Os dispositivos utilizados durante o treino (A-S3), apresentam bastante melhores resultados que os dados novos (S4-S6) sugerindo dificuldade de generalização. As classes mais difíceis de classificar que prejudicam o desempenho médio do modelo *baseline* correspondem aos rótulos bus, metro, metro_station, Public_square, street_pedestrian e tram.

Tabela 5.4: Resultados detalhados da *baseline*.

Cenas acústicas	A	B	C	S1	S2	S3	S4	S5	S6	Acc	Loss
airport	0.977	1.236	1.231	1.769	1.489	1.279	1.666	1.729	2.043	37.0%	1.492
bus	1.099	1.893	1.330	1.954	1.628	1.763	2.382	2.010	2.376	31.1%	1.826
metro	1.025	1.519	1.462	2.312	1.801	1.530	1.358	1.989	1.354	43.1%	1.594
metro_station	1.579	1.809	1.880	1.993	1.918	1.519	1.889	1.953	1.574	35.2%	1.791
park	0.354	0.468	0.386	1.300	0.971	1.353	1.961	1.485	1.930	66.7%	1.134
public_square	1.477	1.582	1.693	1.442	1.748	1.645	1.917	2.124	2.460	33.2%	1.788
shopping_mall	1.120	1.051	1.116	1.477	1.554	1.729	1.585	1.383	1.450	50.2%	1.385
street_pede..	1.397	1.587	1.554	1.495	1.691	1.759	1.954	1.682	2.005	30.0%	1.680
street_traf..	0.841	1.230	1.080	0.814	1.158	1.136	1.027	0.878	1.511	67.8%	1.075
tram	1.322	1.963	1.589	1.420	1.734	1.313	1.851	2.219	1.653	35.9%	1.674
Loss	1.119	1.434	1.331	1.597	1.569	1.503	1.759	1.745	1.836		1.544
Accuracy	59.5%	47.0%	49.5%	41.4%	41.3%	43.6%	35.2%	36.2%	33.4%	43.0%	

5.2 TÉCNICAS DE PRÉ-PROCESSAMENTO E PESQUISA DE RECURSOS

Com o objetivo de ajustar e otimizar ao máximo todo o desenvolvimento, também os dados de entrada foram alvo de adaptação à tarefa em questão. Para conseguir esta adaptação foi realizada uma pesquisa de espectrogramas de mel, utilizando o modelo *baseline* como discriminador. Desta forma foi possível alcançar as configurações de hiperparâmetros que mais beneficiam o sistema. Esta busca pelos hiperparâmetros "ótimos" foi realizada de forma automática com recurso à ferramenta KerasTuner

[136] através do método *Hyperband*. Foi ainda utilizada a ferramenta *Kapre* [137] que permite, através da adição de uma camada extra ao modelo, realizar o processamento de sinal (nomeadamente a aquisição do espectrograma mel), em tempo real durante o processo de treino.

O objetivo é então determinar os valores "ideais" a utilizar no processamento do sinal e extração do espectrograma, tais como tamanho da janela deslizante, a dimensão de sobreposição, número de filtros mel e frequência de amostragem. Para isso foi definida a seguinte janela de pesquisa:

- Opções de pesquisa da frequência de amostragem:
  $FS \in [8, 16, 22.05, 44.1]kHz$
- Opções de pesquisa de tamanho de janela SFFT:
  $SFFT_{Wsize} \in [256, 512, 1024, 2048]$
- Opções de pesquisa do número de filtros de mel (nMels):
  $MEL_{range} \in [20, 300]$; $MEL_{Step} = 20$
- Opções de pesquisa de sobreposição (*Overlap*) proporcional da janela:
  $Overlap \in [25\%, 50\%, 75\%]$

O modelo utilizado é a *baseline* com uma pequena modificação na 3ª camada convolucional - na operação de *Max-Pooling* as dimensões do *kernel* foram alteradas de $4x10$ para $2x2$. Isto foi feito para que o modelo não reduza tanto a amostra e dê a mesma importância a ambos os eixos do espectrograma, embora também permita evitar incompatibilidades durante o ciclo de pesquisa.

Foi realizada uma pesquisa de recursos com o objetivo de adaptar o sinal de entrada (espectrograma de mel) ao modelo. Analisando os resultados obtidos Tabela 5.5, verifica-se que a diminuição da frequência de amostragem e o aumento do número de filtros de mel favorecem melhores resultados. O que pode significar que a informação mais relevante contida no áudio está localizada a baixas frequências. O aumento do número de filtros de mel pode significar que os áudios têm muitas características semelhantes e então é necessário um maior número de filtros de mel aplicados num espaço mais reduzido para os conseguir distinguir. Pode-se observar que as duas configurações de entrada com melhores resultados foram as apresentadas nas duas últimas linhas da Tabela. 5.5. A estas configurações será dada a designação de configuração de entrada 1 (IC1), com $FS = 8kHz$ e $nMels = 260$, e configuração de entrada 2 (IC2), com $FS = 8kHz$ e $nMels = 140$. Note-se também que, enquanto a *baseline* tem uma entrada de $(40x51) = 2040$ amostras, a configuração IC1 tem um formato de $(260x8) = 2080$ e a configuração IC2 tem um formato de $(140x8) = 1120$ amostras.

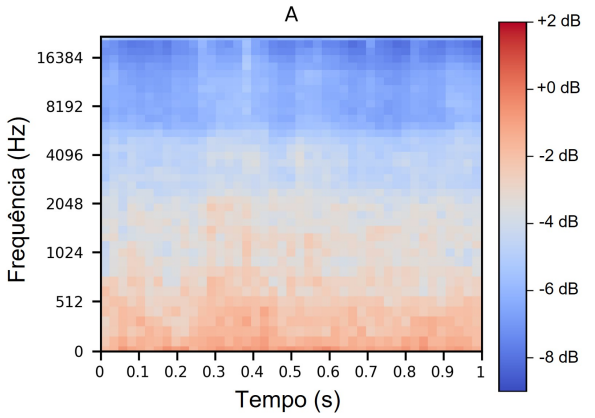
A configuração IC1 apresenta ligeiramente melhor resultados, comparativamente a IC2, no entanto, ao contrário de IC1, a configuração IC2 permite obter uma

redução substancial de recursos, cerca de 46% das amostras, comparativamente ao formato do sinal de entrada utilizado na *Baseline*. Numa tarefa onde os limites de complexidade são limitados este aspeto deve se amplamente considerado. A utilização de *hypertuning*, é uma forma de pesquisa rápida e eficaz de ajustar o *dataset* ao modelo, conseguindo-se uma boa robustez e generalização a novos dados de entrada. Este processo poderia ser realizado através de conhecimento empírico e cientifico, no entanto, seria necessário bastante mais tempo apenas para conhecer o conjunto de dados e a maioria da pesquisa seria realizada na mesma por experimentação.

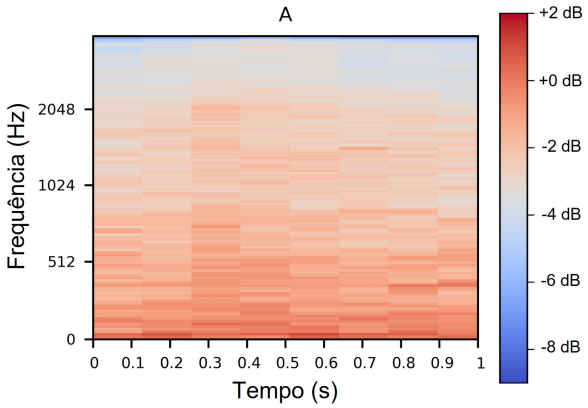
Tabela 5.5: Ajuste do espectrograma de mel - resultados mais significativos obtidos com 2048 STFT.

FS	Overlap	nMels	Acc	Loss	Parametros	MACs
44.1 kHz	1024	40	41.99	1.740	392.2k	34.9M
44.1 kHz	1024	60	43.21	1.713	568.2k	52.4M
22.05 kHz	1536	60	43.90	1.610	184.2k	17.5M
22.05 kHz	1536	140	47.44	1.558	376.2k	40.7M
16 kHz	1024	180	49.47	1.474	616.2k	57.2M
16 kHz	1024	120	48.88	1.496	424.2k	38.1M
8 kHz	1024	260	50.70	1.428	456.2k	41.3M
8 kHz	1024	140	49.50	1.450	264.2k	22.2M

A Fig. 5.2 permite visualizar as mudanças alcançadas ao nível do espectrograma após a pesquisa de recursos realizada. As imagens são obtidas a partir do mesmo áudio. A figura a) diz respeito as configurações de espectrograma utilizadas no modelo *baseline*, o sinal tem uma frequência de amostragem de $44.1kHz$, e uma janela de pesquisa de 40ms e sobreposição a 50%. Já a imagem b) diz respeito a pesquisa efetuada (IC2), utiliza uma frequência de amostragem de $8kHz$, uma janela de pesquisa de 256ms (2048 STFT) e sobreposição a 50%. Verifica-se que a maioria da informação contida na figura a) encontra-se abaixo dos 4kHz talvez por essa razão a pesquisa de recursos tenha determinado a característica $FS = 8kHz$. A diminuição da frequência de amostragem permitiu aumentar o tamanho da janela deslizante como é possível verificar na figura b).



(a)



(b)

Figura 5.2: Configurações de extração de espectrogramas de mel: Baseline Fs=44kHz, janela de 40ms (a); IC2 Fs=8kHz, janela de 256ms (b).

5.3 HYPERTUNING BASELINE

Adotando as características com melhores resultados da pesquisa descrita na secção 5.2, ou seja, as configurações de entrada IC1 e IC2, iniciou-se um processo de pesquisa de arquitetura em torno do modelo *baseline*. Para agilizar o processo foi novamente utilizada a ferramenta KerasTuner [136] e o método *Hyperband*. Esta pesquisa teve a intenção de aumentar o desempenho do modelo e ajustá-lo ao conjunto de dados do desafio, de forma a que o modelo consiga aprender mais informação, preferencialmente útil. Este objetivo pode ser alcançado através do aumento da largura (número de filtros/neurónios) e/ou profundidade (número de camadas) do modelo, ou através de outros hiperparâmetros como dimensões de *kernel*, o tipo e dimensão do *pooling*, a percentagem de *Dropout* entre outros. Obviamente o aumento destes hiperparâmetros tem impacto direto tanto no número de parâmetros do modelo, como na complexidade da rede, pelo que foi necessário

realizar algumas modificações à ferramenta de pesquisa de hiperparâmetros para considerar os limites de 128k de parâmetros e 30MMACs.

Foram pesquisadas duas topologias de modelos, uma para cada configuração de entrada (IC1 e IC2). O alcance da pesquisa definida é o seguinte:

-CNN layer #1:

$Filters = [4, 32]$ $FiltersStep = 4$

-CNN layer #2/#3:

$Filters = [4, 64]$ $FiltersStep = 4$

$Pooling\_type = [avg, max]$

$Pooling\_size = [([1, 2, 3, 4, 6], [1, 2])]$

-Additional CNN layers #[1,2]:

$Filters = [4, 32]$ $FiltersStep = 4$

$Pooling\_type = [avg, max]$

$Pooling\_size = [([1, 2], [1, 2])]$

-Global Pooling $[GlobalMax, GlobalAvg, noPool]$

-Dense $units = [32, 256]$ $unitsStep = 32$

- All CNN layers have:

$Kernel\_size = [([3, 5, 7], [3, 5, 7])]$

$Dropout = [0, 0.7]$ $DropoutStep = 0.1$

É realizada a pesquisa de arquitetura utilizando as entradas pesquisadas anteriormente, os espectrogramas de características IC1 e IC2, a pesquisa é executada a partindo da arquitetura *Baseline*. Os resultados deste *hypertuning* podem ser visualizados na Tabela. 5.6. O modelo designado por TBM1 utiliza a configuração de entrada IC1, enquanto o modelo TBM2 utiliza a configuração IC2 (descritas na secção 5.2). Através desta pesquisa foi possível encontrar modelos "ótimos" com bom desempenho que cumprem os requisitos de complexidade, Tabela 5.6. O modelo que obteve melhor resultado foi o TBM2. Comparativamente ao modelo *Baseline* Tabela 4.5, o modelo TBM2 apresenta melhor desempenho, tem ligeiramente mais parâmetros e menos operações MACs.

Tabela 5.6: Melhor resultado obtido para cada topologia de modelo proposta: detalhes e métricas de desempenho.

Modelo	FS	Input Shape	Acc	Loss	Parametros	MACs
DCASE baseline	44.1 kHz	40x51	42.9	1.575	46.512k	29.23M
TBM1	8 kHz	260x8	49.40	1.425	26.7k	23.6M
TBM2	8 kHz	140x8	49.72	1.386	52.9k	25.5M

5.4 MODELO TBM2

A arquitetura do modelo TBM2 é descrita na Tabela 5.8, é semelhante ao modelo *baseline*, os hiperparâmetros assumem outros valores, é adicionada uma nova camada *CNN* e a camada totalmente conectada *FC* inicia agora com uma camada de *pooling* médio, realizada para cada canal (*Global Average Pooling*). Este modelo foi treinado e testado em dois conjuntos de dados diferentes, com o objetivo identificar com maior clareza a presença de *overfitting*. Os resultados podem ser observados na Tabela 5.7. Os conjuntos utilizados foram os seguintes:

- Treino realizado segundo a divisão do *dataset* padrão (secção 4.2.1) (*Dataset*_padrão).
- Treino realizado utilizando todo o *dataset* de desenvolvimento (*Dataset*_total).

Tabela 5.7: Treino com divisão *dataset* original e com todo o *dataset*.

Dataset	Acc	Loss
*Dataset*_padrão	49.72	1.386
*Dataset*_total	70.00	0.896

A Tabela 5.7, permite comparar as diferenças de desempenho verificadas quando o modelo é treinado apenas com o *dataset* de treino ou com todo o *dataset* de desenvolvimento. Verificam-se resultados bastante dispares, o que sugere *overfitting*. O modelo tem capacidade de aprendizagem, pois consegue obter bons resultados quando treinado para todo *dataset* de desenvolvimento, no entanto, aprende padrões demasiado ajustados ao conjunto de dados, perdendo muito desempenho quando exposto a dados não utilizados em treino.

A arquitetura TBM2, Tabela 5.8, obtida através de *hypertuning* tem fundamentalmente mais 1 camada CNN, 28 filtros e 156 neurónios comparativamente a *baseline* Tabela 4.4. Estas características conferem um aumento substancial da capacidade de aprendizagem da rede tanto em profundidade como em largura, promovendo a aprendizagem de mais características e padrões.

Tabela 5.8: TBM2.

Camada	Detalhes	
CNN1	Conv2d	filtros: 20, kernel size: (7,5)
CNN2	Conv2d	filtros: 28, kernel size: (7,3)
	2D max pooling	pool size: (1, 2)
	Dropout	taxa: 10%
CNN3	Conv2d	filtros: 28, kernel size: (3,7)
	2D max pooling	pool size: (2, 2)
	Dropout	taxa: 30%
CNN4	Conv2d	filtros: 16, kernel size: (7,5)
	Dropout	taxa: 30%
FC	Global Average Pooling	
	Dense	neurónios: 256
	Dropout	taxa: 40%
Saída	Dense	neurónios: 10
	softmax activation	

Todas as Camadas CNN utilizam as operações *Batch normalization* e *ReLu activation*.

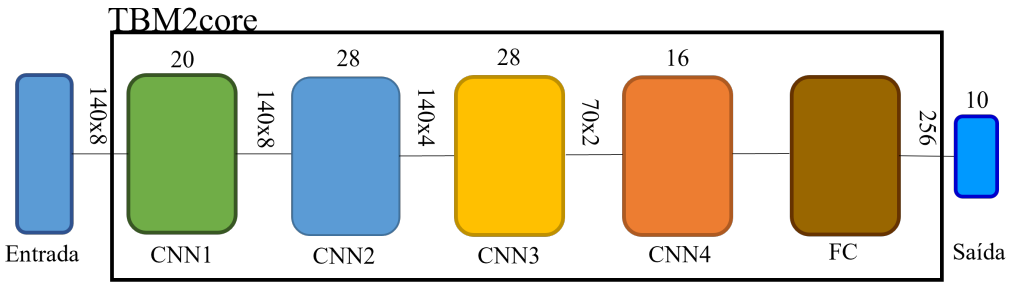


Figura 5.3: Modelo TBM2.

Com o objetivo de melhorar os resultados obtidos, foi também realizado aumento de dados através de diversas técnicas para que o modelo conseguisse generalizar melhor. Como observado na secção 4.2.1, o *dataset* de treino e validação é equilibrado em cenas acústicas, mas é bastante desequilibrado em termos de dispositivos - o dispositivo A tem aproximadamente 14 vezes mais amostras que os restantes. Este problema faz com que o modelo se ajuste demasiado ao dispositivo A perdendo poder de generalização. O problema poderia ser resolvido eliminando dados do dispositivo A, mas como os dados sintéticos são gerados a partir dele, poderíamos estar a perder informação relevante. Resolveu-se então aumentar artificialmente o *dataset* a partir dos dados existentes, utilizando técnicas abordadas na secção 3.1.2. A ferramenta utilizada que possibilitou realizar este o aumento designa-se *Audiomentations* [15]. Os aumentos foram realizados em todos os dispositivos, à exceção do dispositivo A. São propostas três abordagens - aumento de duas (x2), três (x3) e seis vezes(x6) o

número de dados presentes no conjunto de treino e validação, realizadas de forma aleatória e combinando várias técnicas. Uma das técnicas de aumento de dados mais utilizadas segundo a pesquisa realizada na secção 2.4, designa-se *Mixup*, no entanto, esta técnica não foi utilizada pois é incompatível com a técnica de suavização de rótulos anteriormente implementada.

As técnicas de *data augmentation* utilizadas foram:

- Mudança de tom
- Deslocamento temporal
- Alongamento temporal
- Specaugment (mascaramento temporal e de frequência)

Como se pode observar na Tabela. 5.9, os melhores resultados foram obtidos para o maior aumento de dados testado (x6). Verifica-se que o aumento de dados favorece a melhoria dos resultados, logo pode-se inferir que o aumento dos dados até equilibrar o conjunto de dados em termos de dispositivos leva a um aumento do desempenho da rede. No entanto, o tempo de treino aumentaria consideravelmente, *e.g.* para (x6) o tempo de treino aumentou cerca de três vezes relativamente ao tempo de treino sem aumento de dados.

Tabela 5.9: Aumento de dados.

Aumento de dados	**Acc**	**Loss**
✗	49.72	1.386
x2	49.91	1.385
x3	50.12	1.381
x6	50.31	1.378

A Fig. 5.4 mostra a matriz de confusão do modelo *TBM2*, utilizando o *dataset* padrão após o aumento de dados. Idealmente, uma matriz de confusão teria todos os seus valores ao longo da diagonal principal, ou seja, a classe prevista pelo modelo corresponde sempre a classe real. Como se pode observar isso não acontece, embora a maioria dos valores estejam ao longo da diagonal, existem vários valores na restante matriz através desses valores é possível, por exemplo, verificar que a classe com o rótulo *tram* é várias vezes confundida com as classes *bus* e *metro*. Com base na matriz verifica-se que as classes com pior desempenho (inferior a 50% de exatidão) são *public_square*, *street_pedestrian*, *shopping_mall*, *metro_station* e *tram*.

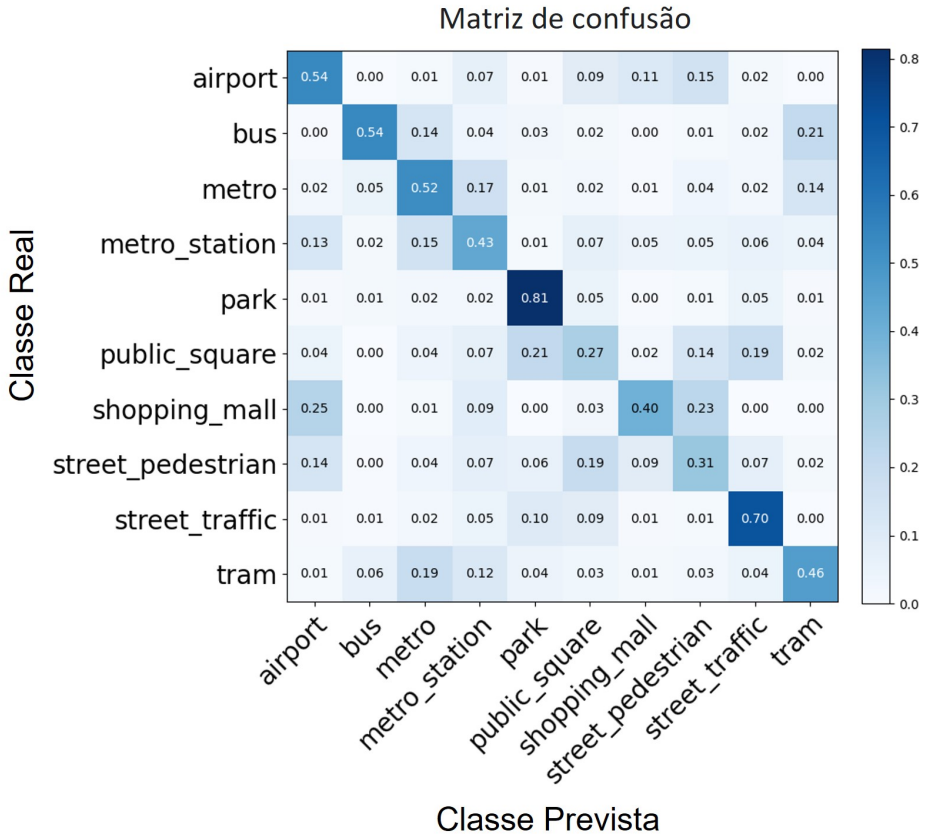


Figura 5.4: Matriz de confusão TBM2.

Comparando a Tabela 5.10 com os resultados verificados anteriormente para o modelo *baseline* Tabela 5.4, verifica-se uma diminuição geral do loss indicando uma melhoria significativa do desempenho, mesmo para dispositivos não treinados, significando melhor generalização.

Tabela 5.10: Resultados detalhados TBM2.

Cenas acústicas	A	B	C	S1	S2	S3	S4	S5	S6	Acc	Loss
airport	1.333	1.344	1.520	1.527	1.264	1.162	1.625	1.618	1.419	46.301	1.424
bus	0.849	1.446	1.379	1.153	1.302	1.088	2.000	1.602	2.519	49.066	1.482
metro	0.916	1.312	1.224	1.494	1.316	1.450	1.332	1.360	1.891	45.800	1.366
metro_station	1.351	1.681	1.945	1.472	1.691	1.143	1.820	1.673	1.475	44.259	1.584
park	0.679	0.462	0.481	1.110	1.287	1.177	0.617	0.642	1.170	76.599	0.847
public_square	1.476	1.438	1.355	1.882	1.992	1.609	1.820	2.029	2.294	29.781	1.766
shopping_mall	1.211	1.314	1.340	1.445	1.126	1.241	1.515	1.426	1.211	51.229	1.314
street_pedestrian	1.732	1.442	1.491	1.691	1.923	1.804	2.181	1.943	1.970	34.411	1.797
street_traffic	0.999	0.958	0.762	0.830	1.100	1.234	1.016	0.969	1.668	66.035	1.060
tram	0.842	1.322	1.040	0.813	1.365	0.831	1.329	1.167	1.550	59.662	1.140
Loss	1.139	1.272	1.254	1.342	1.437	1.274	1.526	1.443	1.717		1.378
Accuracy	57.962	53.388	52.681	50.826	49.023	53.871	47.674	47.689	39.735	50.314	

5.5 ABORDAGENS PROPOSTAS

Verificou-se através da matriz de confusão TBM2 (Fig. 5.4), que seria interessante melhorar a discriminação entre as várias classes. Sabe-se que o desempenho de um modelo neuronal de classificação está diretamente relacionado com número de classes que o mesmo tem - quanto maior o número de rótulos a classificar maior dificuldade terá o modelo em aprender a distinguir entre todas elas. Nesse caso são necessários modelos mais robustos e profundos que consigam extrair e aprender mais informação. Caminhando nesse sentido são então propostas três abordagens que aplicam técnicas como, divisão de classes, agrupamento de modelos (*ensemble*) e destilação de conhecimento KD. O objetivo da aplicação destas técnicas é que o modelo melhore a sua capacidade de descriminação entre classes, aumente a capacidade de aprendizagem e continue a cumprir os requisitos de complexidade. Todas as abordagens utilizam a técnica de interrupção de treino *Early stopping*, na tentativa de regularizar o processo de treino.

As propostas são as seguintes:

- Five-Class Two Ensemble network + Final Classification Dense Layer (5C2EN+FCDL), que será descrita na secção 5.5.1.
- Six-Class Two Ensemble network + Final Classification Dense Layer (6C2EN+FCDL), que será descrita na secção 5.5.2.
- Teacher/Student Knowledge Distillation of Two-Class OvA Ten Ensemble network + Final Classification Dense Layer(TSKD@2C10EN+FCDL), que será descrita na secção 5.5.3.

As abordagens foram treinadas de várias formas, variando o conjunto de treino. É importante que os modelos submetidos ao desafio sejam treinados para todo o conjunto de dados, incluindo os dados de teste, pois contêm dados novos oriundos de outros dispositivos. No entanto, é necessário considerar que a utilização de dados de teste durante o treino aumenta a probabilidade de *overfitting*, originando resultados aparentemente muito bons quando na verdade não são. A utilização da técnica *ensemble* com a adição de um classificador na última camada permite que o modelo seja treinado com um conjunto de dados diferente, do utilizado para treinar os modelos individuais que o constituem, conseguindo-se outras variantes de testes que podem ajudar a melhorar a generalização. De seguida são apresentados as configurações de treino usadas nos vários testes.

- Teste 1: todos os treinos utilizam o *dataset* padrão 4.2.1 (*Dataset*_padrão).
- Teste 2: todos os treinos utilizam o *dataset* de desenvolvimento (*Dataset*_total).

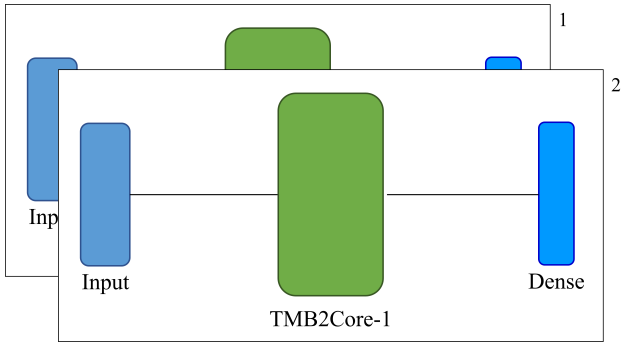
- Teste 3: modelos individuais treinados com *Dataset*_total e o conjunto final com *Dataset*_padrão.
- Teste 4: modelos individuais treinados com *Dataset*_padrão e o conjunto final com *Dataset*_total.

5.5.1 *Proposta 5C2EN+FCDL*

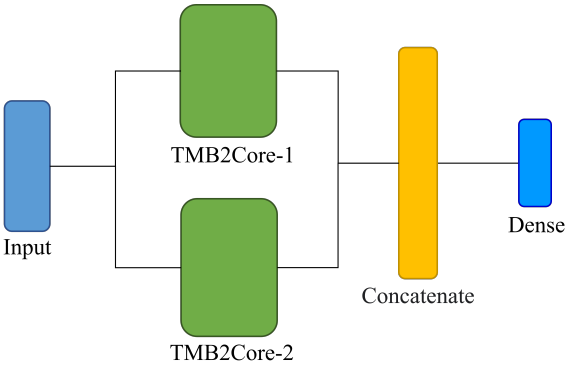
A abordagem 5C2EN+FCDL, propõe a realização de um *ensemble* de dois modelos TBM2' treinados a partir de dois conjuntos de dados com cinco classes cada. O *ensemble* é realizado através da concatenação dos recursos provenientes dos dois modelos e a classificação é realizada por uma camada final totalmente conectada. A divisão das classes em dois grupos foi realizada por observação da matriz de confusão TBM2 (Fig. 5.4), tendo sido criados dois conjuntos de cinco classes, em que cada conjunto contém os rótulos considerados mais confundíveis entre si.

- Grupo 1: Aeroporto, Praça pública, Centro comercial, Rua de pedestres, Parque urbano;
- Grupo 2: Viagem de autocarro, Estação de metro, Viagem de metro, Rua movimentada, Viagem de comboio;

Para atender aos requisitos de complexidade DCASE, os 2 conjuntos são treinados em modelos TBM2', uma variante de TBM2 onde as duas primeiras camadas convolucionais são reduzidas para metade. Cada conjunto de dados é treinado individualmente no respetivo modelo. Após o treino as camadas responsáveis pela extração de recursos (*core*) são removidas e os seus pesos são bloqueados, para manterem as propriedades aprendidas, diminuindo o risco ajuste em demasia. O processo *ensemble* junta então as camadas *core* (TBM2'core-1 e TBM2'core-2) de ambos os modelos através de uma concatenação, é adicionada novamente uma camada densa de classificação e realizado um novo treino. O esquema estrutural desta abordagem pode ser observado na Fig. 5.5.



(a)



(b)

Figura 5.5: Arquitetura 5C2EN+FCDL: Learning (a) Ensemble (b).

Analisando os resultados da proposta de abordagem 5C2EN+FCDL Tabela. 5.11, em comparação com resultados TBM2 Tabela. 5.7 é possível concluir que quando o treino é realizado com todo o *dataset* de desenvolvimento o modelo apresenta melhores resultados, o que significa que proposta apresentada aumentou a sua capacidade de aprendizagem. No entanto, observando os dados treinados com o *dataset* padrão, conclui-se que esta aprendizagem apresenta *overfitting*. Ao utilizar o *ensemble* de dois modelos treinados para conjuntos de classes diferentes, conclui-se que o modelo aumenta a sua capacidade de aprender conseguindo aprender uma maior quantidade de recursos. No entanto, grande parte dos recursos aprendidos não têm utilidade quando o objetivo é obter um modelo que robusto a presença de novos dados, foram aprendidos provavelmente ruídos e informações irrelevantes para diferenciar as classes.

Tabela 5.11: Resultados 5C2EN+FCDL.

Teste	Acc	Loss	Parâmetros	MACs
*Dataset*_padrão	45.5	2.521	70.6k	21.1M
*Dataset*_total	74.5	0.764	70.6k	21.1M
*Dataset*_total+*Dataset*_padrão	39.7	2.446	70.6k	21.1M
*Dataset*_padrão+*Dataset*_total	71.8	0.831	70.6k	21.1M

5.5.2 *Proposta 6C2EN+FCDL*

A proposta 5C2EN+FCDL (secção 5.5.1), foi treinada para extrair recursos de 2 conjuntos de 5 classes. Isto significa que uma classe, apesar de pertencer apenas a um dos grupos, terá também um peso no outro grupo e pode "confundir" o classificador. Este aspeto pode penalizar o desempenho do modelo pois é criada uma "confusão" que influencia a camada densa de classificação final. A abordagem 6C2EN+FCDL tenta minimizar esse problema. Para isso é adicionada uma camada extra a cada um dos conjuntos para conter a informação de todas as classes que não pertencem aquele conjunto. Assim, sempre que uma classe não pertença aquele conjunto, os seus recursos são extraídos de forma diferente e terá menor influência no resultado.

A abordagem 6C2EN+FCDL vem na tentativa de melhorar a abordagem anterior, através da adição de uma classe a cada grupo, com o objetivo de aumentar a distância espacial dos dois grupos. Observando os resultados Tabela 5.12, verifica-se uma diminuição da métrica loss em ambos os conjuntos de dados testados, indicando uma ligeira melhoria no desempenho e diminuição do *overfitting*, no entanto, está longe de ser suficiente.

Tabela 5.12: Resultados 6C2EN+FCDL.

Teste	Acc	Loss	Parâmetros	MACs
*Dataset*_padrão	44.7	2.571	70.6k	21.1M
*Dataset*_total	75.6	0.742	70.6k	21.1M
*Dataset*_total+*Dataset*_padrão	50.1	2.199	70.6k	21.1M
*Dataset*_padrão+*Dataset*_total	73.5	0.791	70.6k	21.1M

5.5.3 *Proposta TSKD@2C10EN+FCDL*

Esta abordagem utiliza a destilação de conhecimento Professor/aluno (KD) de uma rede *ensemble* constituída por 10 modelos TBM2 OvA, conforme ilustrado na Fig. 5.6. É utilizado o modelo TBM2 para classificar cada rótulo de uma forma binária - conceito OvA (um contra todos) - fazendo com que cada modelo aprenda a diferenciar uma classe de todas as restantes. Esta técnica necessita de dez modelos TBM2, um para cada classe. A técnica *ensemble* é aplicada da mesma forma que as abordagens anteriores, dando origem a um modelo bastante complexo constituído por dez TBM2core, que é designado "professor". O modelo "professor", não cumpre os requisitos de complexidade exigidos pelo concurso. Foi então necessário recorrer à técnica de redução KD, que permite baixar complexidade do modelo através da destilação de conhecimento de um modelo de grande dimensão

o "professor" para um mais reduzido designado "aluno". A arquitetura TBM2 cumpre os requisitos de complexidade e vai atuar como "aluno" sendo a escolhida para aprender características semelhantes ao modelo "professor".

A Tabela 5.14 permite comparar o desempenho alcançado pelos modelos "professor" e "aluno". É possível observar que o desempenho dos modelos "professor" apresentam desempenhos opostos consoante o *dataset* utilizado no treino, significando a presença de *overfitting*. Por outro lado, o modelo treinado com todos os dados alcança um desempenho bastante superior às abordagens anteriores, significando que o modelo conseguiu aprender bastante mais recursos. Mais uma vez os modelos aumentaram a sua capacidade de aprender, no entanto, acabaram por aprender recursos pouco relevantes que são facilmente ludibriados na presença de novos dados na avaliação. Analisando os modelos "alunos" Tabela 5.13 é possível verificar que esta abordagem conseguiu melhores resultados que todas as abordagens anteriores. A técnica de destilação de conhecimento revelou excelentes resultados, conseguido para além da expectável redução de complexidade do modelo, uma ajuda significativa no combate ao *overfitting*.

Tabela 5.13: Proposta TSKD@2C10EN+FCDL (alunos).

Teste	Acc	Loss	Parâmetros	MACs
*Dataset*_padrão	50.5	1.347	52.9k	25.5M
*Dataset*_total	58.0	1.170	52.9k	25.5M
*Dataset*_total+*Dataset*_padrão	51.3	1.355	52.9k	25.5M
*Dataset*_padrão+*Dataset*_total	60.5	1.103	52.9k	25.5M

Tabela 5.14: Resultado: destilação de conhecimento, professor e aluno.

Modelo	Ensemble	KD	Acc	Loss	Parâmetros	MACs
*Dataset*_padrão						
Professor	10	✗	42.6	2.707	529k	255M
Aluno	10	✓	50.5	1.347	52.9k	25.5M
*Dataset*_padrão+*Dataset*_total						
Professor	10	✗	98.1	0.221	529k	255M
Aluno	10	✓	60.5	1.103	52.9k	25.5M

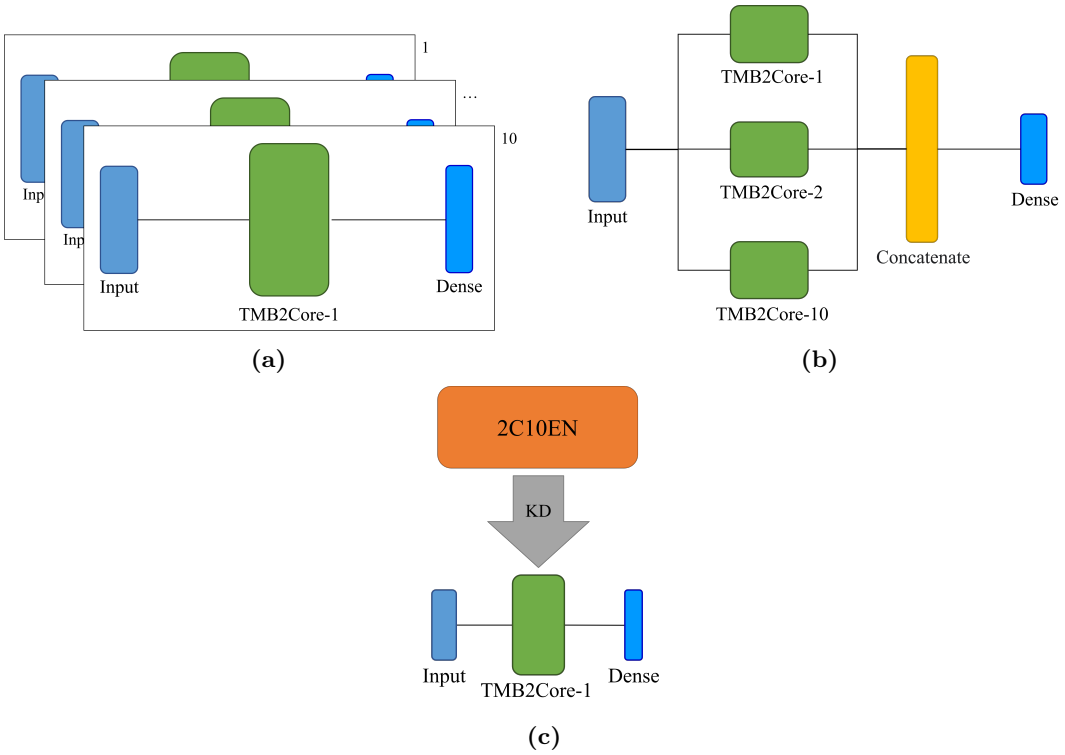


Figura 5.6: Arquitetura TSKD@2C10EN+FCDL: Learning (a), Ensemble (Teacher) (b), Knowledge Distillation (TBM2 Student)(c).

5.6 RESULTADOS SUBMETIDOS A CONCURSO

Considerando todas as propostas desenvolvidas, a abordagem TSKD@2C10EN+FCDL secção 5.5.3 teste *Dataset*_padrão+*Dataset*_total, parece a mais promissora. Esta abordagem consegue utilizar o aumento o campo de aprendizagem introduzido durante o *ensemble* de forma eficiente, com a utilização da destilação de conhecimento foi possível obter um modelo que aprendeu uma maior quantidade de características sem comprometer o *overfitting* contrariamente ao verificado nas abordagens anteriores. A destilação de conhecimento foi um importante regularizador do modelo aproveitando sobretudo a informação útil presente no modelo professor.

Com base na Tabela 5.15, verifica-se que comparativamente ao modelo base TBM2 Tabela 5.10, o modelo "aluno" conseguiu aprender características que não aprenderia se treinado de forma tradicional secção 5.4. Verifica-se uma diminuição substancial do *loss* para todos os dispositivos avaliados, os dispositivos novos apresentam um *loss* bastante mais próximo dos restantes dispositivos indicando uma melhoria significativa na generalização.

Tabela 5.15: Resultados detalhados TSKD@2C10EN+FCDL aluno.

Cenas acústicas	A	B	C	S1	S2	S3	S4	S5	S6	Acc	Loss
airport	1.113	1.044	1.211	1.083	0.973	0.876	1.199	1.172	1.004	61.1%	1.075
bus	0.772	1.332	1.125	0.695	0.909	0.679	1.084	0.773	1.596	68.3%	0.996
metro	0.776	1.029	1.075	1.218	1.115	1.111	1.123	0.982	1.434	60.4%	1.096
metro_station	1.065	1.309	1.633	1.102	1.275	0.983	1.464	1.353	1.195	55.5%	1.264
park	0.480	0.311	0.319	0.791	1.259	0.917	0.489	0.434	0.987	82.6%	0.665
public_square	1.467	1.546	1.402	1.573	1.681	1.297	1.583	1.725	1.903	33.5%	1.575
shopping_mall	0.981	1.004	0.988	0.941	0.770	0.864	0.973	0.877	0.849	67.5%	0.916
street_pede..	1.525	1.255	1.279	1.608	1.846	1.402	1.524	1.665	1.666	39.9%	1.530
street_traf..	0.875	0.813	0.638	0.624	0.835	0.873	0.698	0.681	1.099	75.2%	0.793
tram	0.873	1.317	0.966	0.808	1.560	0.878	1.274	1.098	1.310	61.1%	1.121
Loss	0.993	1.096	1.064	1.044	1.222	0.988	1.141	1.076	1.304		1.103
Accuracy	65.6%	60.3%	61.1%	61.7%	56.5%	65.7%	58.8%	61.7%	53.2%	60.5%	

Com base matriz de confusão da Fig. 5.7, é possível verificar que existe uma diagonal principal bastante mais carregada que na Fig. 5.4, indicando uma clara melhoria de desempenho. Verifica-se que as classes com pior desempenho (inferior a 50% de exatidão) são public_square, street_pedestrian, para a Fig. 5.4 foram contabilizadas cinco classes com exatidão inferior a 50%, sai reforçada a ideia que o modelo "aluno TBM2" apresenta o melhor desempenho.

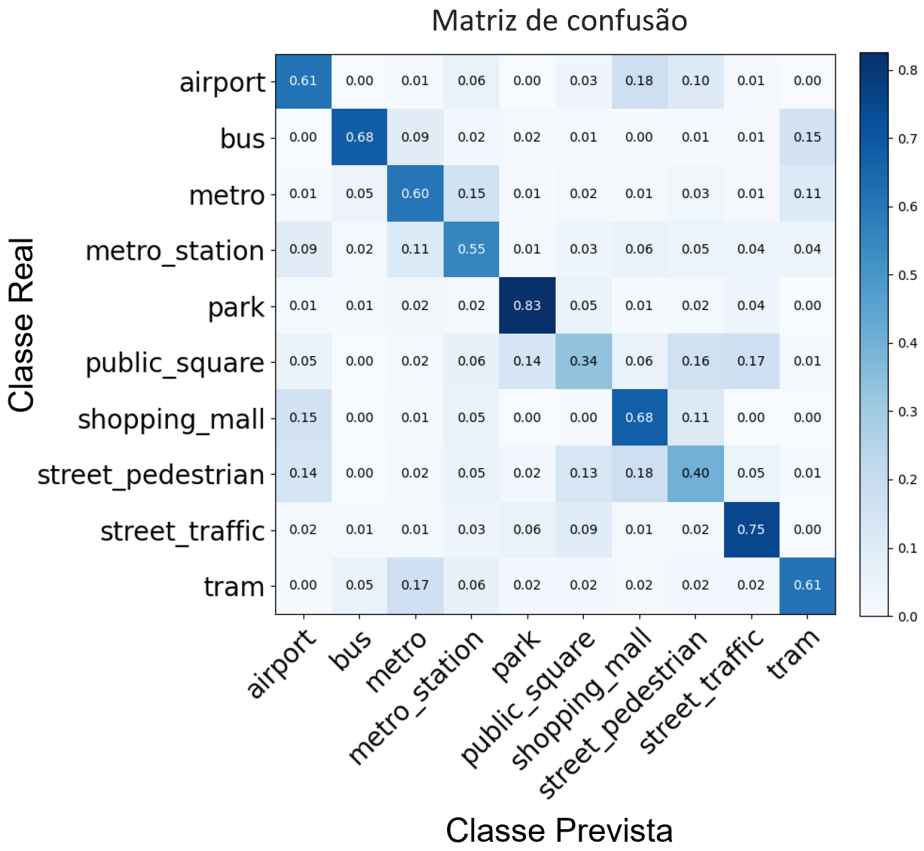


Figura 5.7: Matriz de confusão AI4EDGE_4

A Tabela 5.16 contem as características e resultados dos quatro modelos apresentados a concurso. Os modelos submetidos são baseados nas arquiteturas 6C2EN+FCDL (descritas na secção 5.5.2) e TSKD@2C10EN+FCDL (descritas na secção 5.5.3). O modelos AI4EDGE_1 e AI4EDGE_2 representam a abordagem *ensemble* (6C2EN+FCDL), que junta dois modelos pré-treinados em diferentes conjuntos de rótulos. Os modelos AI4EDGE_3 e AI4EDGE_4 representam a abordagem do sistema (TSKD@2C10EN+FCDL), um conjunto de 10 modelos TBM2 de classificação OvA usados como modelo professor na técnica KD.

Tabela 5.16: Resultados dos modelos submetidos no concurso DCASE2022 Task 1.

Modelo	Ensemble	KD	Acc	Loss	Parâmetros	MACs
AI4EDGE_1	2	✗	75.6	0.742	70.6K	21.1M
AI4EDGE_2	2	✗	73.5	0.791	70.6K	21.1M
AI4EDGE_3	10	✓	50.5	1.347	52.9k	25.5M
AI4EDGE_4	10	✓	60.5	1.103	52.9k	25.5M

O modelo considerado mais promissor submetido a concurso é o AI4EDGE_4. Este apresenta um resultado bastante interessante, comparativamente à *baseline*,

reduzindo o *loss* em 0.472 e aumentado a exatidão (Acc) em 17.6%. Aparenta melhor capacidade de generalização na presença de novos dados comparativamente aos restantes modelos submetidos.

O modelo mais promissor AI4EDGE_4 obteve o seguinte resultado no desafio, Tabela.5.17. Verifica-se que o resultado obtido foi superior à maioria dos 48 modelos avaliados em concurso ficando atingindo o 11° lugar.

Tabela 5.17: Melhor resultado alcançado DCASE2022 Task 1.

Cenas acústicas	Acc	Loss
Airport	42.1 %	1.429
Bus	68.9 %	0.959
Metro	56.2 %	1.193
Metro Station	45.8 %	1.549
Park	73.8 %	0.901
Public square	25.5 %	1.877
Shopping mall	52.7 %	1.314
Street pedestrian	34.7 %	1.722
Street traffic	61.4 %	1.160
Tram	54.8 %	1.192
Total	51.6 %	1.330

A participação no concurso internacional DCASE2022 - Task 1, Low-Complexity Acoustic Scene Classification, decorreu entre 15 de março e 1 de julho de 2022. Participaram 19 equipas de 7 países distintos totalizando 47 modelos submetidos. Foi obtido o 4° Lugar na classificação geral de equipas, o modelo mais promissor AI4EDGE_4 ficou em 11° lugar de um total de 48 modelos.

- R. Anastácio, L. Ferreira, F. Mónica, e C. B. Luís, **"Ai4edgept - Low Complexity Acoustic Scene Classification Task1"**, *Submission to DCASE 2022*, Portugal, junho, 2022 [138];

5.7 DISCUSSÃO

Analisando os vários testes e abordagens como um todo, chegamos a conclusão que a utilização métodos de pesquisa de hiperparâmentros é um processo fundamental e deve ser aplicado tanto na pesquisa de arquitetura como na recolha de características do conjunto de dados. O hiperparâmetro *Batch Size* deve também ser testado, assim como a quantidade de dados utilizados para calibrar o processo de quantização. Verificou-se que o ajuste destas simples características permitiram melhorar o desempenho do modelo. A realização de aumento de dados mostrou também ser uma

ajuda significativa - o modelo aumenta a sua capacidade de generalizar conseguindo melhorar o seu desempenho.

A utilização de técnicas como divisão classes e OvA são úteis para que o modelo aprenda a distinguir com maior facilidade as classes entre si. Estas foram aplicadas sobre *ensemble*, o que proporcionou um grande aumento da capacidade de aprendizagem da rede. O aumento da capacidade de aprendizagem resultou num ajuste exagerado do modelo ao conjunto de dados, reduzindo a sua capacidade de generalização. A utilização da técnica KD, provou ser fundamental para atenuar os problemas gerados pelo uso de *ensemble*, para além de permitir reduzir o modelo cumprindo os requisitos de complexidade, mostrou bons resultados melhorando a generalização do modelo. Através de KD foi possível obter um modelo de arquitetura TBM2, otimizado e com melhores resultados, algo que seria difícil de atingir treinando de uma forma natural.

6

CONCLUSÕES E TRABALHOS FUTUROS

6.1 CONCLUSÕES

O principal foco desta dissertação foi explorar e desenvolver um modelo neuronal de baixa complexidade, com o objetivo de classificar cenas acústicas ASC. Os principais obstáculos ao desenvolvimento do trabalho foram os limitados requisitos de complexidade, típicos de dispositivos MCU e a complexidade inerente dos problemas ASC. O modelo tem de conseguir aprender características do sinal essenciais à classificação e em simultâneo distanciar-se de particularidades mais específicas que podem ser prejudiciais. Por vezes o "ruído" de fundo útil para a classificação é confundido ou ofuscado com "ruídos" parasita e características dos dispositivos de captura, pelo que o modelo tem de filtrar os ruídos que estão naturalmente misturados e sobrepostos.

Durante a realização desta dissertação foram investigadas diversas técnicas de pesquisa, regularização, otimização, aprendizagem e redução de modelos neuronais. Com o propósito de desenvolver um modelo CNN com arquitetura semelhante ao modelo *baseline* (fornecido pelo concurso DCASE), otimizado para obter melhor desempenho e respeitando os limites de complexidade.

Foi realizada uma pesquisa automática de recursos (*hypertuning*) com o intuito de ajustar o conjunto de dados ao modelo e encontrar as especificações de pré-processamento mais adequadas. Isto permitiu descobrir quais características do espectrograma de mel que favorecem o desempenho. Concluiu-se ser mais eficaz manter o foco de aprendizagem em frequências mais baixas e em intervalos de tempo maiores, o que permite melhorar a generalização e o desempenho, permitindo ainda reduzir o número de amostras necessárias para representar o sinal. Também o modelo foi ajustado ao conjunto de dados através de *hypertuning*. Foi possível obter hiperparâmetros que otimizam o modelo *baseline*, alterando a sua arquitetura, conseguindo-se aumentar campo de aprendizagem de forma equilibrada e melhorar o desempenho, dentro dos limites de complexidade.

Durante os testes realizados foi possível detetar várias vezes que os modelos e abordagens desenvolvidas tendem a ajustar-se demasiado a informações pouco relevantes, generalizando de forma pouco eficaz. As técnicas de suavização de rótulos,

paragem antecipada do treino e aumento de dados, conseguiram melhorar problemas de *overfitting* ainda que ligeiramente.

As três abordagens propostas utilizaram técnicas de aprendizagem com o objetivo de distanciar classes facilmente confundíveis entre si. As técnicas de aprendizagem foram implementadas através do agrupamento de modelos *ensemble*. A separação de classes e conjuntos de dados em grupos, permitiu em conjunto com o *ensemble* aumentar a capacidade de aprendizagem do modelo resultante, provocando o aumento de *overfitting*. Foram testadas algumas abordagens de treino para atenuar este problema, *e.g.* utilizar todo o conjunto de dados apenas no treino final de *ensemble* o que permitiu reduzir o *overfitting* provocado pelo aumento do campo de aprendizagem.

A técnica de destilação do conhecimento permitiu uma redução significativa do tamanho do modelo, conseguindo transferir várias características aprendidas para um modelo aluno de dimensão 10x menor. Este método mostrou também ter um impacto positivo combatendo o *overfitting*.

As abordagens desenvolvidas permitiram melhorar o desempenho comparativamente ao modelo *baseline*, cumprindo os requisitos de complexidade do desafio. Foi realizada a participação no concurso *DCASE2022 - Task 1* tendo sido obtido 4° Lugar na classificação por equipas refletindo a qualidade do trabalho desenvolvido. O melhor modelo AI4EDGE_4 obteve os seguintes resultados ACC=51.6% e LOSS=1.330 enquanto que o modelo *baseline* obteve ACC=44.2% e LOSS=1.532.

6.2 TRABALHOS FUTUROS

Futuramente, é importante continuar a combater o principal obstáculo ao desenvolvimento desta tarefa, o *overfitting*. É interessante prosseguir com aumento de dados utilizando mais técnicas até equilibrar o *dataset* para todos os dispositivos. Seria interessante testar a utilização de redes adversárias generativas como técnica de aumento de dados gerando espectrogramas a partir de outros já existentes.

A introdução de conexões residuais ou a utilização da arquitetura ResNet podem também beneficiar o desempenho do modelo ignorando as camadas que prejudiquem o desempenho, minimizando a propagação do erro.

As abordagens utilizadas nesta dissertação poderiam também ser aplicadas em arquiteturas de baixa complexidade computacional habitualmente utilizadas em classificação de imagem como SqueezeNet [139], MobileNet [140] e ShuffleNet [141].

Seria interessante testar a utilização *pruning* estruturado, que para além de reduzir a complexidade do modelo pode ajudar a diminuir o *overfitting* através da remoção de filtros e neurónios com menor impacto no desempenho. A utilização

de *pruning* estruturado terá provavelmente um impacto negativo no desempenho, no entanto, pode melhorar a generalização se utilizada como método regularizador após o aumento do campo de aprendizagem.

A utilização do método de classificação OvO pode também ser uma ajuda importante na descriminação entre classes.

Esta página foi propositadamente deixada em branco.

BIBLIOGRAFIA

[1] J. McCarthy, M. L. Minsky, N. Rochester, and C. E. Shannon, "A proposal for the dartmouth summer research project on artificial intelligence, august 31, 1955," *AI Magazine*, vol. 27, pp. 12–12, 12 2006. [Online]. Available: https://ojs.aaai.org/index.php/aimagazine/article/view/1904

[2] W. Bao, C. Wu, S. Guleng, J. Zhang, K.-L. A. Yau, and Y. Ji, "Edge computing-based joint client selection and networking scheme for federated learning in vehicular iot," *China Communications*, vol. 18, pp. 39–52, 6 2021.

[3] M. Muniswamaiah, T. Agerwala, and C. C. Tappert, "A survey on cloudlets, mobile edge, and fog computing," *Proceedings - 2021 8th IEEE International Conference on Cyber Security and Cloud Computing and 2021 7th IEEE International Conference on Edge Computing and Scalable Cloud, CSCloud-EdgeCom 2021*, pp. 139–142, 6 2021.

[4] C. H. Lu and X. Z. Lin, "Toward direct edge-to-edge transfer learning for iot-enabled edge cameras," *IEEE Internet of Things Journal*, vol. 8, pp. 4931–4943, 3 2021.

[5] I. Fedorov, R. P. Adams, M. Mattina, and P. N. Whatmough, "Sparse: Sparse architecture search for cnns on resource-constrained microcontrollers," *Advances in Neural Information Processing Systems*, vol. 32, 5 2019. [Online]. Available: https://arxiv.org/abs/1905.12107v1

[6] Arm, "Tinyml brings ai to smallest arm devices - arm blueprint," 2021. [Online]. Available: https://www.arm.com/blogs/blueprint/tinyml

[7] S. research, "Iot microcontroller (mcu) market share, growth, forecast to 2030," 2021. [Online]. Available: https://straitsresearch.com/report/iot-microcontroller-market

[8] I. Insights, "Microcontrollers will regain growth after 2019 slump," 2019. [Online]. Available: https://www.icinsights.com/news/bulletins/Microcontrollers-Will-Regain-Growth-After-2019-Slump/

[9] P. P. Ray, "A review on tinyml: State-of-the-art and prospects," *Journal of King Saud University - Computer and Information Sciences*, vol. 34, pp. 1595–1623, 4 2022.

[10] S. Soro, "Tinyml for ubiquitous edge ai," 2 2021. [Online]. Available: https://arxiv.org/abs/2102.01255v1

[11] Python, "Python release python 3.6.15 | python.org," 2021. [Online]. Available: https://www.python.org/downloads/release/python-3615/

[12] B. McFee, C. Raffel, D. Liang, D. P. Ellis, M. McVicar, E. Battenberg, and O. Nieto, "librosa: Audio and music signal analysis in python," 2015. [Online]. Available: https://librosa.org/

[13] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[14] F. Chollet *et al.*, "Keras," 2015. [Online]. Available: https://github.com/fchollet/keras

[15] I. Jordal, A. Tamazian, E. T. Chourdakis, C. Angonin, askskro, N. Karpov, T. Dhyani, O. Sarioglu, kvilouras, E. B. Çoban, F. Mirus, J.-Y. Lee, K. Choi, MarvinLvn, SolomidHero, and T. Alumäe, "iver56/audiomentations: v0.26.0," 8 2022. [Online]. Available: https://zenodo.org/record/7010042

[16] I. M. M. DCASE, "Baseline system for dcase 2022 task 1," 2022. [Online]. Available: https://github.com/marmoi/dcase2022_task1_baseline

[17] T. H. DCASE, "Dcase utilities 1.0 documentation." [Online]. Available: https://dcase-repo.github.io/dcase_util/index.html

[18] V. J. Reddi, B. Plancher, S. Kennedy, L. Moroney, P. Warden, L. Suzuki, A. Agarwal, C. Banbury, M. Banzi, M. Bennett, B. Brown, S. Chitlangia, R. Ghosal, S. Grafman, R. Jaeger, S. Krishnan, M. Lam, D. Leiker, C. Mann, M. Mazumder, D. Pajak, D. Ramaprasad, J. E. Smith, M. Stewart, and D. Tingley, "Widening access to applied machine learning with tinyml," *Harvard Data Science Review*, 6 2021. [Online]. Available: https://arxiv.org/abs/2106.04008v2

[19] "Project Highlight: An ECG Analyzer Powered by Edge Impulse - News - SparkFun Electronics." [Online]. Available: https://www.sparkfun.com/news/3903

[20] "Cough Detection with TinyML on Arduino - Arduino Project Hub." [Online]. Available: https://create.arduino.cc/projecthub/edge-impulse/cough-detection-with-tinyml-on-arduino-417f37

[21] "Medical Alert System Using TinyML Fall Detection and BLE - Hackster.io." [Online]. Available: https://www.hackster.io/NathanielF/medical-alert-system-using-tinyml-fall-detection-and-ble-5e0f1f

[22] "AI:HAR Detection." [Online]. Available: https://wiki.st.com/stm32mcu/wiki/AI:How_to_perform_motion_sensing_on_STM32L4_IoTnode

[23] "GitHub - ShawnHymel/tinyml-example-anomaly-detection: TinyML example showing how to do anomaly detection with Python and Arduino." [Online]. Available: https://github.com/ShawnHymel/tinyml-example-anomaly-detection

[24] Audioanalytic, "audioanalytic." [Online]. Available: https://www.audioanalytic.com/app/uploads/2020/03/tinyML-Summit-poster-Dominic-Binks-Audio-Analytic.pdf

[25] M. Giordano mgiordano, ethzch ETH Zürich Zürich, S. Philipp Mayer, I. -ETH Zürich Zürich, and S. Michele Magno, "A Battery-Free Long-Range Wireless Smart Camera for Face Detection; A Battery-Free Long-Range Wireless Smart Camera for Face Detection," 2020. [Online]. Available: https://doi.org/10.1145/3417308.3430273

[26] A. L. Buczak and E. Guven, "A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection," *IEEE Communications Surveys and Tutorials*, vol. 18, no. 2, pp. 1153–1176, apr 2016.

[27] M. De Prado, M. Rusci, A. Capotondi, R. Donze, L. Benini, and N. Pazos, "Robustifying the Deployment of tinyML Models for Autonomous Mini-Vehicles," 2021. [Online]. Available: https://doi.org/10.3390/s21041339

[28] G. Crocioni, G. Gruosso, D. Pau, D. Denaro, S. A. Brianza, I. L. Zambrano, G. D. Giore, L. Zambrano, and G. Di, "Characterization of Neural Networks Automatically Mapped on Automotive-grade Microcontrollers; Characterization of Neural Networks Automatically Mapped on Automotive-grade Microcontrollers," 2021.

[29] A. N. Roshan, B. Gokulapriyan, C. Siddarth, and P. Kokil, "Adaptive Traffic Control With TinyML," *2021 Sixth International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)*, pp. 451–455, mar 2021. [Online]. Available: https://ieeexplore.ieee.org/document/9419472/

[30] S. Chopade, H. P. Gupta, R. Mishra, A. Oswal, P. Kumari, and T. Dutta, "A Sensors based River Water Quality Assessment System using Deep Neural Network," *IEEE Internet of Things Journal*, 2021.

[31] S. P. Mohanty, D. P. Hughes, and M. Salathé, "Using Deep Learning for Image-Based Plant Disease Detection," *Frontiers in Plant Science*, vol. 0, no.

September, p. 1419, sep 2016.

[32] M. PANARIELLO, "Low-complexity neural networks for robust acoustic scene classification in wearable audio devices - thesis," Master's thesis, 2022. [Online]. Available: https://webthesis.biblio.polito.it/22584/

[33] K. Imoto, "Introduction to acoustic event and scene analysis," *Acoustical Science and Technology*, vol. 39, pp. 182–188, 2018.

[34] J. Abeßer, "A review of deep learning based methods for acoustic scene classification," *Applied Sciences (Switzerland)*, vol. 10, 3 2020.

[35] D. S. K. Zieliński, "Applied sciences | special issue : Applications of machine learning in audio classification and acoustic scene characterization," 2022. [Online]. Available: https://www.mdpi.com/journal/applsci/special_issues/Machine_Learning_Audio_Acoustic

[36] B. Kim, S. Yang, J. Kim, and S. Chang, "QTI submission to DCASE 2021: Residual normalization for device-imbalanced acoustic scene classification with efficient design," DCASE2021 Challenge, Tech. Rep., June 2021.

[37] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*, 10 2017. [Online]. Available: https://arxiv.org/abs/1710.09412v2

[38] D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le, "Specaugment: A simple data augmentation method for automatic speech recognition," 2019. [Online]. Available: http://dx.doi.org/10.21437/Interspeech.2019-2680

[39] H. Pettersson and O. Stensöta, "Data augmentation for audio based machine learning classifying brachycephalic obstructive airway syndrome (boas) in dogs," 2021. [Online]. Available: www.chalmers.se

[40] B. Kim, S. Chang, J. Lee, and D. Sung, "Broadcasted residual learning for efficient keyword spotting," *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, vol. 3, pp. 1853–1857, 6 2021. [Online]. Available: https://arxiv.org/abs/2106.04140v2

[41] J. Kim, K. Yoo, and N. Kwak, "Position-based scaled gradient for model quantization and pruning," 2020. [Online]. Available: https://github.com/Jangho-Kim/PSG-pytorch

[42] J. Kim, M. Hyun, I. Chung, and N. Kwak, "Feature fusion for online mutual knowledge distillation," *Proceedings - International Conference on Pattern Recognition*, pp. 4619–4625, 4 2019. [Online]. Available: https://arxiv.org/abs/1904.09058v2

[43] C.-H. H. Yang, H. Hu, S. M. Siniscalchi, Q. Wang, W. Yuyang, X. Xia, Y. Zhao, Y. Wu, Y. Wang, J. Du, and C.-H. Lee, "A lottery ticket hypothesis framework for low-complexity device-robust neural acoustic scene classification," DCASE2021 Challenge, Tech. Rep., June 2021.

[44] T. Nguyen, F. Pernkopf, and M. Kosmider, "Acoustic scene classification for mismatched recording devices using heated-up softmax and spectrum correction," *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, vol. 2020-May, pp. 126–130, 5 2020.

[45] H. Hu, C.-H. H. Yang, X. Xia, X. Bai, X. Tang, Y. Wang, S. Niu, L. Chai, J. Li, H. Zhu, F. Bao, Y. Zhao, S. M. Siniscalchi, Y. Wang, J. Du, and C.-H. Lee, "A two-stage approach to device-robust acoustic scene classification," *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, vol. 2021-June, pp. 845–849, 11 2020. [Online]. Available: http://arxiv.org/abs/2011.01447http://dx.doi.org/10.1109/ICASSP39728.2021.9414835

[46] C. Szegedy, W. Liu, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," 2014.

[47] J. Frankle and M. Carbin, "The lottery ticket hypothesis: Finding sparse, trainable neural networks," *7th International Conference on Learning Representations, ICLR 2019*, 3 2018. [Online]. Available: https://arxiv.org/abs/1803.03635v5

[48] K. Koutini, S. Jan, and G. Widmer, "Cpjku submission to dcase21: Cross-device audio scene classification with wide sparse frequency-damped CNNs," DCASE2021 Challenge, Tech. Rep., June 2021.

[49] K. Koutini, H. Eghbal-zadeh, and G. Widmer, "Receptive field regularization techniques for audio classification and tagging with deep convolutional neural networks," *IEEE/ACM Transactions on Audio Speech and Language Processing*, vol. 29, pp. 1987–2000, 5 2021. [Online]. Available: http://arxiv.org/abs/2105.12395http://dx.doi.org/10.1109/TASLP.2021.3082307

[50] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky, "Domain-adversarial training of neural networks," *Advances in Computer Vision and Pattern Recognition*, vol. 17, pp. 189–209, 5 2015. [Online]. Available: https://arxiv.org/abs/1505.07818v4

[51] F. Schmid, S. Masoudian, K. Koutini, and G. Widmer, "CP-JKU submission to dcase22: Distilling knowledge for low-complexity convolutional neural networks from a patchout audio transformer," DCASE2022 Challenge, Tech. Rep., June 2022.

[52] B. Kim, S. Yang, J. Kim, H. Park, J.-T. Lee, and S. Chang, "Domain generalization with relaxed in-stance frequency-wise normalization in 2d neural audio processing," 2021.

[53] K. Koutini, J. Schlüter, H. Eghbal-zadeh, and G. Widmer, "Efficient training of audio transformers with patchout," 10 2021. [Online]. Available: https://arxiv.org/abs/2110.05069v3

[54] J.-H. Lee, J.-H. Choi, P. M. Byun, and J.-H. Chang, "Hyu submission for the DCASE 2022: Efficient fine-tuning method using device-aware data-random-drop for device-imbalanced acoustic scene classification," DCASE2022 Challenge, Tech. Rep., June 2022.

[55] K. Doshi, "Audio deep learning made simple: Sound classification, step-by-step | towards data science," 2021. [Online]. Available: https://towardsdatascience.com/audio-deep-learning-made-simple-sound-classification-step-by-step-cebc936bbe5

[56] T. Inoue, P. Vinayavekhin, S. Wang, D. Wood, A. Munawar, B. Ko, N. Greco, and R. Tachibana, "Shuffling and mixing data augmentation for environmental sound classification," 10 2019.

[57] S.-H. Gao, M.-M. Cheng, K. Zhao, X.-Y. Zhang, M.-H. Yang, and P. Torr, "Res2net: A new multi-scale backbone architecture," *IEEE transactions on pattern analysis and machine intelligence*, vol. 43, pp. 652–662, 4 2019.

[58] T. Liu, R. K. Das, K. A. Lee, and H. Li, "Mfa: Tdnn with multi-scale frequency-channel attention for text-independent speaker verification with short utterances," pp. 7517–7521, 4 2022.

[59] Y. Jung, S. M. Kye, Y. Choi, M. Jung, and H. Kim, "Improving multi-scale aggregation using feature pyramid module for robust speaker verification of variable-duration utterances," *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, vol. 2020-October, pp. 1501–1505, 4 2020. [Online]. Available: http://arxiv.org/abs/2004.03194http://dx.doi.org/10.21437/Interspeech.2020-1025

[60] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 2704–2713, 12 2017. [Online]. Available: https://arxiv.org/abs/1712.05877v1

[61] T. Morocutti and D. Shalaby, "Receptive field regularized CNNs with traditional audio augmentations," DCASE2022 Challenge, Tech. Rep., June 2022.

[62] K. Paralkar, "Audio data augmentation in python | medium," 2020. [Online]. Available: https://medium.com/@keur.plkar/audio-data-augmentation-in-python-a91600613e47

[63] K. Koutini, F. Henkel, H. Eghbal-Zadeh, and G. Widmer, "Detection and classification of acoustic scenes and events 2020 cp-jku submissions to dcase'20: Low-complexity cross-device acoustic scene classification with rf-regularized cnns technical report."

[64] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," 2015.

[65] DCASE, "Low-complexity acoustic scene classification - dcase," 2022. [Online]. Available: https://dcase.community/challenge2022/task-low-complexity-acoustic-scene-classification

[66] H. Tachibana, N. Ono, H. Kameoka, and S. Sagayama, "Harmonic/percussive sound separation based on anisotropic smoothness of spectrograms," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 22, no. 12, pp. 2059–2073, 2014.

[67] T. Bäckström, "Deltas and delta-deltas - introduction to speech processing - aalto university wiki," 2019. [Online]. Available: https://wiki.aalto.fi/display/ITSP/Deltas+and+Delta-deltas

[68] C. A. Peron and D. Santos, "Classificação automática de cenas acústicas usando algoritmos de clusterização," 11 2019. [Online]. Available: http://repositorio.utfpr.edu.br:8080/jspui/handle/1/6002

[69] D. G. D. P. ZANCO, "Classificação de cenas acústicas utilizando técnicas de aprendizagem profunda," 2018.

[70] J. Kwon and D. Park, "Hardware/software co-design for tinyml voice-recognition application on resource frugal edge devices," *Applied Sciences 2021, Vol. 11, Page 11073*, vol. 11, p. 11073, 11 2021. [Online]. Available: https://www.mdpi.com/2076-3417/11/22/11073/htmhttps://www.mdpi.com/2076-3417/11/22/11073

[71] R. N. Tak, D. M. Agrawal, and H. A. Patil, "Novel phase encoded mel filterbank energies for environmental sound classification," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 10597 LNCS, pp. 317–325, 2017. [Online]. Available: https://www.researchgate.net/publication/320733074_Novel_Phase_Encoded_Mel_Filterbank_Energies_for_Environmental_Sound_Classification

[72] A. I. Iliev, M. Dewli, M. Kalkan, P. P. Kudva, and R. Turkar, "Acoustic event detection and sound separation for security systems and iot devices," 2021. [Online]. Available: www.freesound.org

[73] "Discrete fourier transform (dft) — python numerical methods," 2020. [Online]. Available: https://pythonnumericalmethods.berkeley.edu/notebooks/chapter24.02-Discrete-Fourier-Transform.html

[74] M. Huzaifah, "Comparison of time-frequency representations for environmental sound classification using convolutional neural networks," 6 2017. [Online]. Available: https://arxiv.org/abs/1706.07156v1

[75] J. Lee, J. Park, K. L. Kim, and J. Nam, "Samplecnn: End-to-end deep convolutional neural networks using very small filters for music classification," *Applied Sciences 2018, Vol. 8, Page 150*, 2018.

[76] Z. Zhong, L. Zheng, G. Kang, S. Li, and Y. Yang, "Random erasing data augmentation," *AAAI 2020 - 34th AAAI Conference on Artificial Intelligence*, pp. 13 001–13 008, 8 2017. [Online]. Available: https://arxiv.org/abs/1708.04896v2

[77] M. Bayer, M.-A. Kaufhold, and C. Reuter, "A survey on data augmentation for text classification," *ACM Computing Surveys*, 7 2021. [Online]. Available: http://arxiv.org/abs/2107.03158http://dx.doi.org/10.1145/3544558

[78] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *Journal of Big Data*, vol. 6, pp. 1–48, 12 2019. [Online]. Available: https://journalofbigdata.springeropen.com/articles/10.1186/s40537-019-0197-0

[79] N. Takahashi, M. Gygli, B. Pfister, and L. V. Gool, "Deep convolutional neural networks and data augmentation for acoustic event detection," *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, vol. 08-12-September-2016, pp. 2982–2986, 4 2016. [Online]. Available: https://arxiv.org/abs/1604.07160v2

[80] K. M. Abdellatif, "Mixup data augmentation for deep learning side-channel attacks," *Cryptology ePrint Archive*, 2021.

[81] G. Kim, D. K. Han, and H. Ko, "Specmix : A mixed sample data augmentation method for training withtime-frequency domain features," *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, vol. 1, pp. 6–10, 8 2021. [Online]. Available: https://arxiv.org/abs/2108.03020v1

[82] IBM, "What are neural networks? | ibm," 2020. [Online]. Available: https://www.ibm.com/cloud/learn/neural-networks

[83] S.-H. Han, K. W. Kim, S. Kim, and Y. C. Youn, "Artificial neural network: Understanding the basic concepts without mathematics," *Dementia and Neurocognitive Disorders*, vol. 17, p. 83, 2018. [Online]. Available: https://www.researchgate.net/publication/329920042_Artificial_Neural_Network_Understanding_the_Basic_Concepts_without_Mathematics

[84] L. Alzubaidi, J. Zhang, A. J. Humaidi, A. Al-Dujaili, Y. Duan, O. Al-Shamma, J. Santamaría, M. A. Fadhel, M. Al-Amidie, and L. Farhan, "Review of deep learning: concepts, cnn architectures, challenges, applications, future directions," *Journal of Big Data 2021 8:1*, vol. 8, pp. 1–74, 3 2021. [Online]. Available: https://journalofbigdata.springeropen.com/articles/10.1186/s40537-021-00444-8

[85] Y. Wang, Y. Li, Y. Song, and X. Rong, "The influence of the activation function in a convolution neural network model of facial expression recognition," *Applied Sciences 2020, Vol. 10, Page 1897*, vol. 10, p. 1897, 3 2020. [Online]. Available: https://www.mdpi.com/2076-3417/10/5/1897/htmhttps://www.mdpi.com/2076-3417/10/5/1897

[86] T. Wood, "Softmax function definition | deepai." [Online]. Available: https://deepai.org/machine-learning-glossary-and-terms/softmax-layer

[87] J. Teuwen and N. Moriakov, "Convolutional neural networks," *Handbook of Medical Image Computing and Computer Assisted Intervention*, pp. 481–501, 1 2020.

[88] M. Mishra, "Convolutional neural networks, explained | towards data science," 2020. [Online]. Available: https://towardsdatascience.com/convolutional-neural-networks-explained-9cc5188c4939

[89] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *32nd International Conference on Machine Learning, ICML 2015*, vol. 1, pp. 448–456, 2 2015. [Online]. Available: https://arxiv.org/abs/1502.03167v3

[90] K. Doshi, "Batch norm explained visually," 2021. [Online]. Available: https://towardsdatascience.com/batch-norm-explained-visually-how-it-works-and-why-neural-networks-need-it-b18919692739

[91] N. Srivastava, G. Hinton, A. Krizhevsky, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014.

[92] Sebastian, "Dropout regularization in neural networks: How it works and when to use it - programmathically," 2021. [Online]. Available: https://programmathically.com/dropout-regularization-in-neural-networks-

how-it-works-and-when-to-use-it/

[93] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2016-December, pp. 770–778, 12 2015. [Online]. Available: https://arxiv.org/abs/1512.03385v1

[94] H. Yadav, "Residual blocks in deep learning, towards data science," 2022. [Online]. Available: https://towardsdatascience.com/residual-blocks-in-deep-learning-11d95ca12b00

[95] scikit learn, "Metrics and scoring: quantifying the quality of predictions." [Online]. Available: https://scikit-learn.org/stable/modules/model_evaluation.html#log-loss

[96] S. Saxena, "Binary cross entropy/log loss for binary classification," 3 2021. [Online]. Available: https://www.analyticsvidhya.com/blog/2021/03/binary-cross-entropy-log-loss-for-binary-classification/

[97] M. Grandini, E. Bagli, and G. Visani, "Metrics for multi-class classification: an overview," 8 2020. [Online]. Available: https://arxiv.org/abs/2008.05756v1

[98] P. Murugan and S. Durairaj, "Regularization and optimization strategies in deep convolutional neural network," 12 2017. [Online]. Available: https://arxiv.org/abs/1712.04711v1

[99] IBM, "What is underfitting? | ibm," 2021. [Online]. Available: https://www.ibm.com/cloud/learn/underfitting

[100] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2016-December, pp. 2818–2826, 12 2015. [Online]. Available: https://arxiv.org/abs/1512.00567v3

[101] C. F. G. dos Santos and J. P. Papa, "Avoiding overfitting: A survey on regularization methods for convolutional neural networks," *ACM Computing Surveys*, 1 2022. [Online]. Available: http://arxiv.org/abs/2201.03299http://dx.doi.org/10.1145/3510413

[102] R. Müller, S. Kornblith, and G. Hinton, "When does label smoothing help?" *Advances in Neural Information Processing Systems*, vol. 32, 6 2019. [Online]. Available: https://arxiv.org/abs/1906.02629v3

[103] IBM, "What is overfitting? | ibm," 2021. [Online]. Available: https://www.ibm.com/cloud/learn/overfitting

[104] A. C. R. Holbrook, "Overfitting and underfitting | kaggle." [Online]. Available: https://www.kaggle.com/code/ryanholbrook/overfitting-and-underfitting

[105] A. Band, "Multi-class classification one-vs-all e one-vs-one towards data science," 5 2020. [Online]. Available: https://towardsdatascience.com/multi-class-classification-one-vs-all-one-vs-one-94daed32a87b

[106] P. Pawara, E. Okafor, M. Groefsema, S. He, L. R. Schomaker, and M. A. Wiering, "One-vs-one classification for deep neural networks," *Pattern Recognition*, vol. 108, p. 107528, 12 2020.

[107] M. Galar, A. Fernández, E. Barrenechea, H. Bustince, and F. Herrera, "An overview of ensemble methods for binary classifiers in multi-class problems: Experimental study on one-vs-one and one-vs-all schemes," *Pattern Recognition*, vol. 44, pp. 1761–1776, 8 2011.

[108] M. A. Ganaie, M. Hu, A. K. Malik, M. Tanveer, and P. N. Suganthan, "Ensemble deep learning: A review."

[109] JiYi, "06.ensemble training - deep learning." [Online]. Available: https://wikidocs.net/165452

[110] L. Breiman, "Bagging predictors," *Machine Learning 1996 24:2*, vol. 24, pp. 123–140, 1996. [Online]. Available: https://link.springer.com/article/10.1007/BF00058655

[111] C. Cortes, M. Mohri, and U. Syed, "Deep boosting," 2014.

[112] D. H. Wolpert, "Stacked generalization," *Neural Networks*, vol. 5, pp. 241–259, 1 1992.

[113] Google, "Overview of hyperparameter tuning | vertex ai | google cloud." [Online]. Available: https://cloud.google.com/vertex-ai/docs/training/hyperparameter-tuning-overview

[114] J. Navas, "Anyscale - what is hyperparameter tuning?" 2022. [Online]. Available: https://www.anyscale.com/blog/what-is-hyperparameter-tuning

[115] C. A. Lab, "Hyper-parameter optimization algorithms: a short review - criteo ai lab," 2019. [Online]. Available: https://ailab.criteo.com/hyper-parameter-optimization-algorithms-a-short-review/

[116] J. JORDAN, "Hyperparameter tuning for machine learning models." 2017. [Online]. Available: https://www.jeremyjordan.me/hyperparameter-tuning/

[117] D. Passos and P. Mishra, "A tutorial on automatic hyperparameter tuning of deep spectral modelling for regression and classification tasks," *Chemometrics and Intelligent Laboratory Systems*, vol. 223, p. 104520, 4 2022.

[118] J. Bergstra, J. B. Ca, and Y. B. Ca, "Random search for hyper-parameter optimization yoshua bengio," *Journal of Machine Learning Research*, vol. 13, pp. 281–305, 2012. [Online]. Available: http://scikit-learn.sourceforge.net.

[119] K. Eggensperger, M. Feurer, F. Hutter, J. Bergstra, J. Snoek, H. H. Hoos, and K. Leyton-Brown, "Towards an empirical foundation for assessing bayesian optimization of hyperparameters," 2013. [Online]. Available: www.automl.org/hpolib.

[120] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar, "Hyperband: A novel bandit-based approach to hyperparameter optimization," *Journal of Machine Learning Research*, vol. 18, pp. 1–52, 3 2016. [Online]. Available: https://arxiv.org/abs/1603.06560v4

[121] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, A. Talwalkar, and U. Berkeley, "Hyperband: Bandit-based configuration eval-uation for hyperparameter optimization," 2017. [Online]. Available: https://github.com/automl/RoBO.

[122] K. Jamieson and A. Talwalkar, "Non-stochastic best arm identification and hyperparameter optimization." PMLR, 5 2016, pp. 240–248. [Online]. Available: https://proceedings.mlr.press/v51/jamieson16.html

[123] M. Bahmani, "Hyperband and bohb: Understanding state of the art hyperparameter optimization algorithms - neptune.ai," 2022. [Online]. Available: https://neptune.ai/blog/hyperband-and-bohb-understanding-state-of-the-art-hyperparameter-optimization-algorithms

[124] "Otimização de modelo | TensorFlow Lite." [Online]. Available: https://www.tensorflow.org/lite/performance/model_optimization

[125] K. Wang, Z. Liu, Y. Lin, J. Lin, and S. Han, "HAQ: Hardware-Aware Automated Quantization with Mixed Precision."

[126] J. Myllenberg and J. Johansson, "Optimizing machine learning inference for mcu:s," 2020.

[127] D. Blalock, J. J. G. Ortiz, J. Frankle, and J. Guttag, "What is the state of neural network pruning?" 3 2020. [Online]. Available: https://arxiv.org/abs/2003.03033v1

[128] J. Sowerby, "Smart pruning: Improve mobile ml performance - ai and ml blog - arm community blogs - arm community," 2020. [Online]. Available: https://community.arm.com/arm-community-blogs/b/ai-and-ml-blog/posts/smart-pruning

[129] T.-J. Yang, Y.-L. Liao, and V. S. MIT, "Netadapt project," 2021. [Online]. Available: https://web.mit.edu/netadapt/

[130] H. Cai, C. Gan, T. Wang, Z. Zhang, and S. Han, "Once-for-all: Train one network and specialize it for efficient deployment," 8 2019. [Online]. Available: https://arxiv.org/abs/1908.09791v5

[131] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," 3 2015. [Online]. Available: https://arxiv.org/abs/1503.02531v1

[132] N. Romano and R. Schucker, "Distilling knowledge to specialist networks for clustered classification," 2016.

[133] T. Heittola, A. Mesaros, and T. Virtanen, "Tau urban acoustic scenes 2022 mobile, development dataset," 3 2022. [Online]. Available: https://zenodo.org/record/6337421

[134] A. Mesaros, T. Heittola, and T. Virtanen, "Detection and classification of acoustic scenes and events," 2018. [Online]. Available: https://doi.org/10.5281/zenodo.1228142,

[135] S. Barreto, "Why mini-batch size is better than one single "batch" with all training data | baeldung on computer science," 2021. [Online]. Available: https://www.baeldung.com/cs/mini-batch-vs-single-batch-training-data

[136] T. O'Malley, E. Bursztein, J. Long, F. Chollet, H. Jin, L. Invernizzi *et al.*, "Kerastuner," 2019. [Online]. Available: https://github.com/keras-team/keras-tuner

[137] K. Choi, D. Joo, and J. Kim, "Kapre: On-gpu audio preprocessing layers for a quick implementation of deep neural network models with keras," in *Machine Learning for Music Discovery Workshop at 34th International Conference on Machine Learning*. ICML, 2017.

[138] R. Anastácio, L. Ferreira, F. Mónica, and C. B. Luís, "Ai4edgept submission to DCASE 2022 low complexity acoustic scene classification task1," DCASE2022 Challenge, Tech. Rep., June 2022.

[139] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <0.5mb model size," 2 2016. [Online]. Available: https://arxiv.org/abs/1602.07360v4

[140] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," 4 2017. [Online]. Available: https://arxiv.org/abs/1704.04861v1

[141] X. Zhang, X. Zhou, M. Lin, and J. Sun, "Shufflenet: An extremely efficient convolutional neural network for mobile devices," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 6848–6856, 7 2017. [Online]. Available: https://arxiv.org/abs/1707.01083v2

Esta página foi propositadamente deixada em branco.

D E C L A R A Ç Ã O

Declaro, sob compromisso de honra, que o trabalho apresentado nesta dissertação, com o título *" CLASSIFICAÇÃO DE CENAS ACÚSTICAS EM DISPOSITIVOS COM CONSTRANGIMENTOS COMPUTACIONAIS UTILIZANDO INTELIGÊNCIA ARTIFICIAL "*, é original e foi realizado por Ricardo Ribeiro Anastácio (2192601) sob orientação de Professor Doutor Luís Manuel Conde Bento (luis.conde@ipleiria.pt) e Professora Doutora Mónica Jorge Carvalho de Figueiredo (monica.figueiredo@ipleiria.pt).

Assinado por: **Ricardo Ribeiro Anastácio**
Num. de Identificação: 14859289
Data: 2022.12.22 17:05:27+00'00'

Leiria, Setembro 2022

Ricardo Ribeiro Anastácio