Universitat Oberta de Catalunya

uoc.edu

Tècniques de *deep learning* per reconeixement i classificació d'àudio

Jordi Garriga Muñoz
Grau en Enginyeria Informàtica
Intel·ligència Artificial

Gabriel Moyà Alcover
Tutor

16 de juny de 2024

FITXA DEL TREBALL FINAL

Títol del treball:	*Tècniques de deep learning per reconeixement i classificació d'àudio*
Nom de l'autor:	*Jordi Garriga Muñoz*
Nom del consultor/a:	*Gabriel Moyà*
Nom del PRA:	*Susana Acedo Nadal*
Data de lliurament (mm/aaaa):	06/2024
Titulació:	*Grau en Enginyeria Informàtica*
Àrea del Treball Final:	*Intel·ligència Artificial*
Idioma del treball:	Català
Nombre de crèdits:	12
Paraules clau	*deep learning,* classificació d'àudio, CNN

Resum del Treball (màxim 250 paraules): *Amb la finalitat, context d'aplicació, metodologia, resultats i conclusions del treball.*

El món de la Intel·ligència Artificial cada cop avança més ràpid i cada dia veiem novetats en tots els seus vessants, especialment en el processament i generació d'imatges i vídeo. Les actuals tècniques de *deep learning* com les *Convolutional Neural Networks* o els models *transformers* han permès grans avenços mai pensats abans.

En aquest treball es pretén introduir el lector en l'ús d'aquestes tècniques adaptades al processament del so i audio. Per fer-ho, s'ha triat un camp concret com és el del reconeixement i classificació de sons.

El treball consta de dues parts: d'una banda, s'introdueixen els conceptes teòrics bàsics sobre aquesta disciplina; característiques del so i mètodes de conversió analògica-digital, processament, i tractament posterior. També s'expliquen les solucions disponibles actualment al mercat i els diferents estudis i recerca que diversos investigadors duen a terme en aquest camp.

De l'altra, es vol mostrar un cas pràctic concret de l'ús del *deep learning* per la classificació de so. Mitjançant la creació d'un model convenientment entrenat a partir d'un *dataset* amb gran quantitat de referències sonores, aquest ha de ser capaç d'identificar i classificar amb el major nivell de precisió possible fragments sonors del mateix tipus.

Per fer-ho, s'utilitzarà una combinació de vàries tècniques, englobades dins d'un concepte teòric conegut com CLAP (*Contrastive Language Audio Procesing*), que fa servir CNNs per processar els fragments sonors del conjunt d'entrenament juntament amb etiquetes de text que descriuen el so que conté el fragment.

Abstract (in English, 250 words or less):

The world of Artificial Intelligence is advancing faster and every day we see innovations in all its facets, especially in the processing and generation of images and video. Current deep learning techniques such as Convolutional Neural Networks or transformer models have enabled great advancements never thought of before.

This work aims to introduce the reader to the use of these techniques adapted to sound and audio processing. To do this, a specific field such as sound recognition and classification has been chosen.

The work is divided in two parts: on one hand, the basic theoretical concepts of this discipline are introduced: sound characteristics and analog-to-digital conversion methods, processing, and subsequent treatment. It also explains the applications currently available on the market and the different studies and research carried out by various researchers in this field.

On the other hand, it aims to show a specific practical case of the use of deep learning for sound classification. Through the creation of a properly trained model from a dataset with a large number of sound references, it should be able to identify and classify sound fragments of the same type with the highest level of accuracy possible.

To do this, a combination of various techniques will be used, encompassed within a theoretical concept known as CLAP (Contrastive Language Audio Processing), which uses CNNs to process sound fragments from the training set along with text labels describing the sound contained in the fragment.

Índex

Llista de figures

Llista de taules

1. Introducció

1.1. Context i justificació del Treball

En aquests dos últims anys hem contemplat l'esclat de l'ús de la Intel·ligència Artificial(IA) més enllà de l'entorn acadèmic i professional. Les noves eines de generació i comprensió de textos són una demostració de les enormes possibilitats que aporta l'aprenentatge profund (*deep learning)* per ajudar els computadors a interpretar el món real.

Un dels camps que ha despertat més expectació és el de la generació de vídeo i imatge. Recentment OpenAI ha presentat Sora [1], la seva eina per generació de vídeo, amb resultats sorprenents i altres solucions com Midjourney [2] o Stable Diffusion [3] no han deixat d'avançar en aquest mateix camp. Tenint en compte l'estreta vinculació entre imatge i so, cal progressar doncs en l'aplicació d'aquestes tecnologies en aquest últim camp.

Especialment pel que fa al reconeixement de sons, hem vist grans avenços en el camp del reconeixement de veu [4], que han donat lloc a solucions *speech-to-text* o fins i tot *speech-to-speech*, podent crear assistents virtuals que permeten un cert nivell de conversa amb l'usuari, però no passa el mateix amb els altres sons que ens envolten: el so dels animals, els vehicles de combustió, la maquinària de la indústria treballant, i fins i tot els sons del propi cos humà, com els batecs del cor o la respiració.

El tractament adequat d'aquests sons a través de sistemes de IA ha donat lloc a infinitat d'aplicacions. Les més populars probablement siguin les solucions per separar els diferents instruments presents en una cançó o peça musical; d'entre elles destaca Demucs [5] la solució creada per un ex enginyer de l'empresa tecnològica Meta (antiga Facebook) que es pot fer anar a través d'una *Application Program Interface* (API) pròpia, però la gamma de possibilitats és enorme: des d'aplicacions mèdiques per detectar malalties a partir de sons (patrons de respiració o mal funcionament d'alguns òrgans) fins a millores d'accessibilitat per persones amb discapacitat. En aquest últim camp els sistemes de reconeixement de sons podran ser de gran ajuda per gent amb limitacions sonores o auditives, així com en sistemes de transcripció o de traducció simultània. Actualment ja existeix programari que utilitza la IA i el reconeixement de sons per afegir audiodescripcions en pel·lícules o programes de TV, per exemple.

Des d'una perspectiva més global, hi ha un gran consens en què el so i l'àudio haurien d'anar de la mà de la investigació de la visió per computador, així com de la generació de vídeo, per crear una experiència més complerta per l'usuari i facilitar la interacció de les màquines amb l'entorn. Un bon treball en aquest camp pot permetre a una sistema robòtic humanoide o industrial, a través de la instal·lació de sensors acústics, reconèixer i definir l'espai on està; això també pot ajudar a augmentar la percepció que les màquines tenen del món que els envolta. Alguns dels avenços més recents els podem veure en la Industria de l'Internet de les Coses(IoT, per les seves sigles en anglés), com per exemple aquest projecte de l'Imperial College de London [6] on s'ha desenvolupat un sistema de reconeixement de so sota terra per mitjà de sensors per reconèixer l'entorn i saber a quina distància estan els objectes combinant la visió amb el so.

Espero, doncs, poder posar llum a un tema que em sembla apassionant i la recerca del qual va molt més enllà del que veurem aquí; no obstant, crec que aquest pot ser un bon punt de partida per tothom qui estigui interessat en el món del so, la música i l'àudio en general. No hem d'oblidar que, al cap i a la fi el so en totes les seves formes és una part essencial de la comunicació humana.

1.2. Objectius del Treball

Com s'ha explicat anteriorment, el treball busca assolir dos grans objectius: l'aprofundiment en els conceptes bàsics sobre processament d'àudio digital i la demostració pràctica de com es pot dur a terme el reconeixement i classificació dels sons fent servir tècniques d'aprenentatge profund. Més concretament, els objectius són:

Teòrics, sobre els coneixements elementals del camp

- Introduir el lector en els conceptes més elementals del processament digital d'àudio per poder entendre correctament del què s'està parlant i elaborar un judici propi.
- Documentar i resumir les tècniques utilitzades fins ara en aquest camp.
- Investigar sobre les eines, idees i solucions en les quals s'està treballant actualment en el camp de la recerca.

Pràctics, sobre el model que es construirà:

- Tria i justificació o bé construcció d'un conjunt de dades (*dataset*) que servirà de base per crear el model.
- Tractament (preprocessament, neteja i selecció) del conjunt escollit.
- Creació dels conjunts d'entrenament i de test degudament justificada.
- Entrenament del model.
- Avaluació del model amb els conjunts de test.
- Modificar o refinar el model si les mètriques no són les esperades.

Per últim, es pretén aconseguir també que el model sigui capaç de tractar entrades de dades més avançades:

- Que pugui reconèixer sons d'altres conjunts diferents del triat; amb això tindrem una bona mètrica de l'eficàcia real del model, ja que aquest tipus de conjunt de dades no són abundants, i fer servir dades de les mateixes fonts pot viciar el model.
- Que, donada una escena sonora en la qual es barregen diferents sons, sigui capaç de distingir i identificar cadascun d'ells per separat, el que es coneix com a *segmentació de àudio*, una de les aplicacions més útils avui dia de la classificació d'àudio.

1.3. Enfocament i abast del treball

La idea principal del treball és crear un model amb el llenguatge de programació Python, aprofitant principalment les diferents llibreries que tenim disponibles per al tractament d'àudio. La major part d'aquestes llibreries fan us de algoritmes d'aprenentatge profund per entrenar els models i de mètodes de neteja i processament de dades específic per àudio.

Per entrenar el model farem servir un conjunt amb gravacions sonores que són petits fragments d'àudio que poden anar des dels 2 segons fins a quasi mig minut de durada. Aquests sons van acompanyats d'un arxiu de metadades, habitualment en format *.csv*, que conté una descripció textual de l'arxiu sonor corresponent.

El resultat ha de ser un model d'inferència el qual ha de ser capaç d'analitzar un fragment de so d'entrada i reconèixer a quina classe pertany (quin tipus de so és). La informació de sortida del model estarà condicionada pel tipus de dades d'entrenament, de manera que depenent de l'estructura de les dades d'entrenament, les dades d'inferència seran més o menys complertes.

El mètode serà el següent:

1. *Es triarà un conjunt de dades adequat a les necessitats del treball*. Per fer-ho, es tindran en compte diversos factors: mida del conjunt, varietat de sons, tècniques utilitzades per obtenir els sons (micròfons, qualitat de samplejat, etc.) i l'etiquetat de les dades.
2. *Es preprocessarà el conjunt de dades per adaptar*-lo correctament a les necessitats d'entrenament.
3. *Es triarà un model,* basat en tècniques de tipus llenguatge contrastiu (*constrative language*), que s'explicarà a fons més endavant en el treball.
4. *S'entrenarà el model.* Es crearan els conjunts d'entrenament i de validació; es provaran diverses tècniques i mètriques de validació.
5. *Es farà l'avaluació del model,* primer amb les dades de prova reservades del propi conjunt i posteriorment amb dades d'altres fonts.

En aquest context, es especialment sensible la tria del joc de dades, ja que les seves metadades, que són les que contenen la descripció textual de l'arxiu sonor, defineixen en gran mida la precisió els resultats obtinguts. Com més específic sigui l'etiquetat dels arxius, més específic pot ser el nostre model, però també té més possibilitats de confondre's o de donar resultats inesperats

Tota aquesta estratègia s'engloba, a un nivell més alt, en el mètode **CRISP-DM** [7] de mineria de dades: *comprensió del negoci, comprensió de les dades, preparació de les dades, modelat, avaluació, desplegament i refinament iteratiu del mode*l segons els resultats de l'avaluació.

Les citacions i referències bibliogràfiques seran en format **ISO690**.

1.4. Planificació del Treball

Per planificar el treball seguirem les orientacions de les PAC definides al Pla Docent. Les primeres i la última PACs (la núm.0, la núm.1 i la núm.5) tenen els objectius clarament definits; per les altres tres hem creat un diagrama de Gantt que ens ajudarà a portar un control dels temps del treball.

1. **PAC0**: Triar el tema del treball i elaborar un primer guió.
2. **PAC1**: Documentació més específica sobre el tema triat, establir objectius i crear la introducció del treball i la planificació temporal.
3. **PAC2**: Documentació extensiva, redacció de part teòrica del treball amb els conceptes més importants, tria del *dataset* i primeres accions per crear el model.
4. **PAC3**: Construcció del model definitiu; entrenament i proves d'inferència; primeres accions d'ajustament fi.
5. **PAC4**: *Fine-tuning* per elaborar el model definitiu; objectius secundaris (fragments d'altres *datasets* i escenes sonores completes). Redacció definitiva de la memòria.
6. **PAC5a**: Elaborar la presentació.
7. **PAC5b**: Preparació de la defensa pública el dia 30 de juny i redacció de l'informe d'autoavaluació.

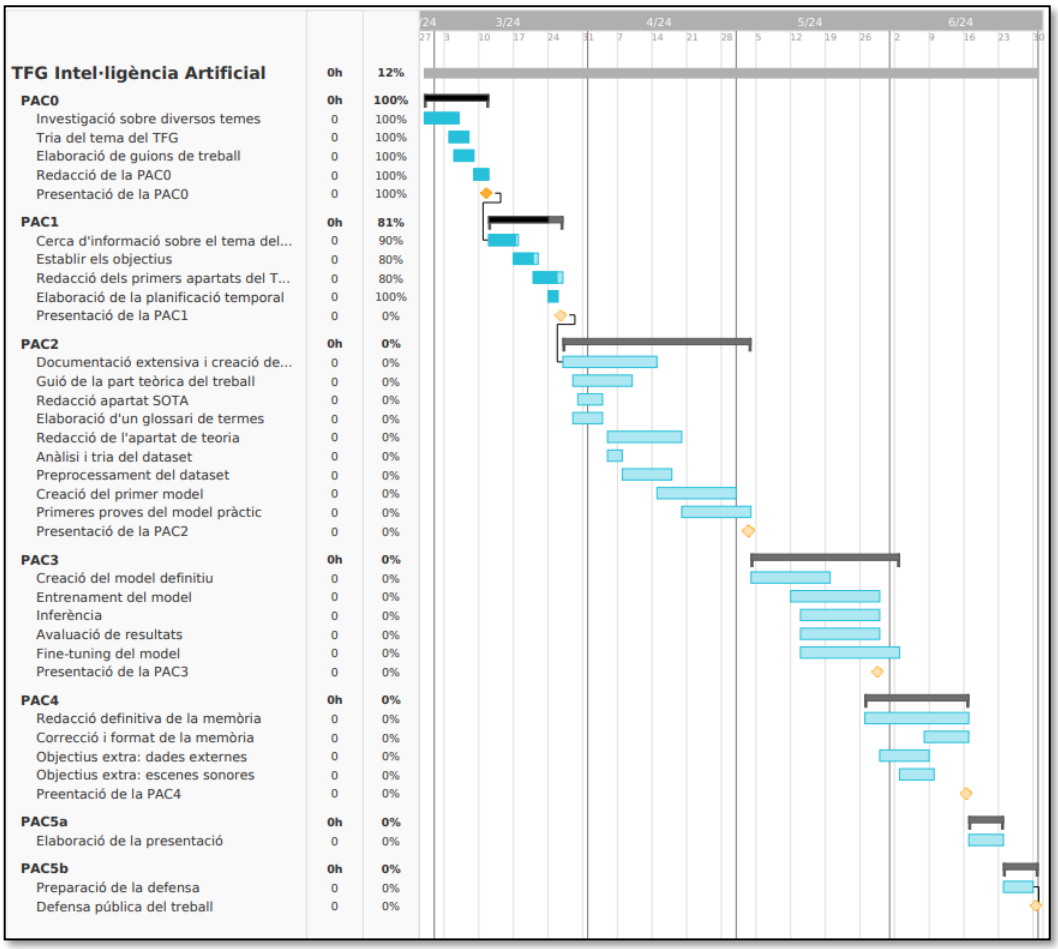


Figura 1 – Diagrama de Gantt per la planificació temporal [creat amb programari en línia Team Gantt].

Addicionalment, durant la l'elaboració de les PAC es facilita un enllaç permanent a la pàgina de l'eina Team Gantt des de la qual es farà el seguiment de la planificació:
https://app.teamgantt.com/projects/gantt?ids=3896240

1.5. Breu sumari de contribucions i productes obtinguts

En data 16/06/2024, s'ha obtingut els següents entregables:

- 3 informes de seguiment de cada una de les primeres PAC.
- Aquesta memòria final amb l'explicació del procés en detall .
- El projecte en *python* del model, disponible en el repositori https://github.com/Gardenou/TFG_IA_Model , que consta de:
    - Les classes per implementar el model.
    - Una carpeta amb els arxius *.pth* dels pesos resultants més significatius.
    - Una carpeta amb els arxius *.csv* amb els *logs* d'entrenament més significatius.
    - Els arxius *requirements.txt* i *pyvenv.cfg*, que contenen tota la informació sobre les dependències necessàries per executar el projecte.

2. Teoria del so

Abans de crear o analitzar cap mena de model de classificació de so, cal explicar alguns conceptes bàsics sobre quina és la naturalesa d'aquest, així com alguns conceptes de matemàtica bàsica per processament d'àudio digital, imprescindibles per entendre el funcionament d'aquest tipus de models.

2.1. Definició

En aquest treball, el pilar fonamental és el concepte del so. Segons **l'Institut d'Estudis Catalans**, el so es defineix com la:

> *"Impressió produïda en l'òrgan de l'oïda per les vibracions elàstiques d'un cos que es propaguen en tots els medis materials en forma d'ones".*

El cos productor de les vibracions pot ser qualsevol: el propi cos humà, un objecte, un instrument musical acústic o un dispositiu d'amplificació elèctrica, com ara un altaveu.
Tot i que la definició ja ens aclareix que el so es pot propagar per qualsevol mitjà, el més habitual és la seva propagació per l'aire, concretament en forma d'**ones sinusoidals.**

2.2. Conceptes matemàtics bàsics per la manipulació del so

Al llarg del nostre treball, ens caldrà manipular el so des de l'ordinador per aconseguir una sèrie de transformacions o dur a terme una sèrie de processos. Per fer-ho, és essencial presentar primer uns quants conceptes bàsics de matemàtica aplicada al so. No és l'objectiu d'aquest treball fer un aprofundiment en aquest camp sinó simplement tenir un coneixement de base que ajudi a entendre'ls.
Per ajudar-nos en l'explicació, farem servir principalment programari i llibreries Python per processament i visualització de so. En el nostre cas, fem anar la versió 3.11 de Pyhton i les llibreries *numpy* i *scipy* i *matplotlib*. Per algunes visualitzacions més complexes utilitzarem el programari *Sonic Visualizer* i la llibreria *torchaudio*.

Concepte d'ona sinusoidal

Tal com hem explicat anteriorment, el so es representa en forma d'una ona que té forma sinusoidal; podem considerar que aquesta ona és la unitat bàsica del so. Aquesta ona representa, com diem, una funció trigonomètrica, que oscil.la entre dos valors $-x$ i x de manera periòdica i amb una certa freqüència. Aquesta ona la podem definir, entre d'altres, amb la següent equació:

$$x[n] = A\cos(\omega nT + \varphi) = A\cos(2\pi f nT + \varphi)$$

on tenim que:

- A és la amplitud, la desviació màxima de la funció del punt 0.
- ω és la freqüència angular expressada en radians/segon.
- f és la freqüència representada en Hertzs (que s'obté dividint la freqüència angular per 2π).
- φ és la fase inicial de la ona, el valor de la mateixa quan t=0.

Habitualment també podem definir el temps t com: $t = nT$, on:

- n és una variable que representa un índex temporal i
- T és la freqüència de mostreig (*sampling rate*) del so o període.

Així doncs, podem entendre una ona com un vector $x[n]$ on n és el número de mostres, és a dir, els valors de la funció en cada instant de temps t.

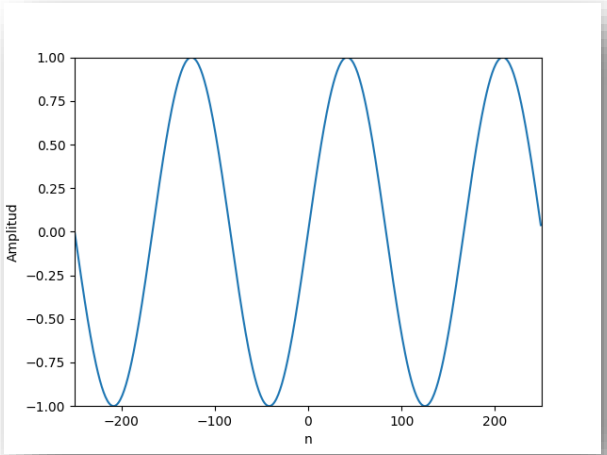


Figura 2 – Ona sinusoidal elemental [creació pròpia amb programari Python].

Transformada discreta de Fourier (DFT)

L'exemple d'ona que hem vist abans és un cas particular d'un sinusoide que representa una sola freqüència; és el que es coneix com a to purament acústic. Els sons que sentim en el nostre dia a dia, com la música, els sorolls, inclús la nostra veu, estan composats, però, de diverses capes de freqüències; és el que es coneix com a freqüències constituents del so, les quals donen al so un timbre, una riquesa i una diversitat que el fan únic i distingible de la resta. En aquest cas, la forma de la ona és bastant més complexa que l'anterior:

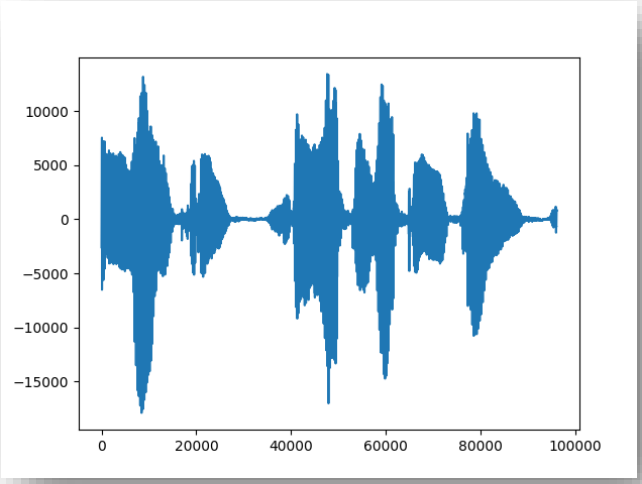


Figura 3 – Ona sinusoidal complexa [representació d'arxiu sonor amb programari Python].

Per poder extreure i analitzar aquestes freqüències, comptem amb el procés de la transformació Discreta de Fourier, un tipus d'anàlisi de Fourier [8] sobre els valors discrets que representen les mostres de la funció sonora $\mathrm{x[n]}$. Gràcies a aquesta transformació, obtenim una nova seqüència de mostres $\mathrm{x[k]}$, que anomenem **espectre**, i que representa els harmònics presents en el so. La DFT ve representada per la següent equació:

$$X[k] = \sum_{N=0}^{N-1} x[n] e^{-j2\pi kn/N} \qquad per\ a\ k = 0,1,\ldots,N-1$$

on tenim que:

- n és l'índex temporal discret
- k és l'índex de freqüències discret o període de la ona
- N és el nombre de mostres
- $x[n]$ és la funció que representa la senyal d'entrada, amb n mostres
- $X[k]$ és la funció de sortida (espectre)
- $e^{-j2\pi kn/N}$ és el que es coneix com exponent complex o funció base de la DFT

A més, cal recordar que per obtenir la freqüència absoluta tenim les fórmules:

- $\omega_k = 2\pi\, k/N$, que ens dona la freqüència en *radians/segon*
- $f_k = f_s\, k/N$, on f_s és la freqüència de mostreig, ens dona la freqüència en *Hertz*

Al multiplicar les dues funcions de la DFT, obtenim el valor complex de cada mostra de la funció d'entrada. Posteriorment sumem totes les mostres per obtenir l'espectre final:

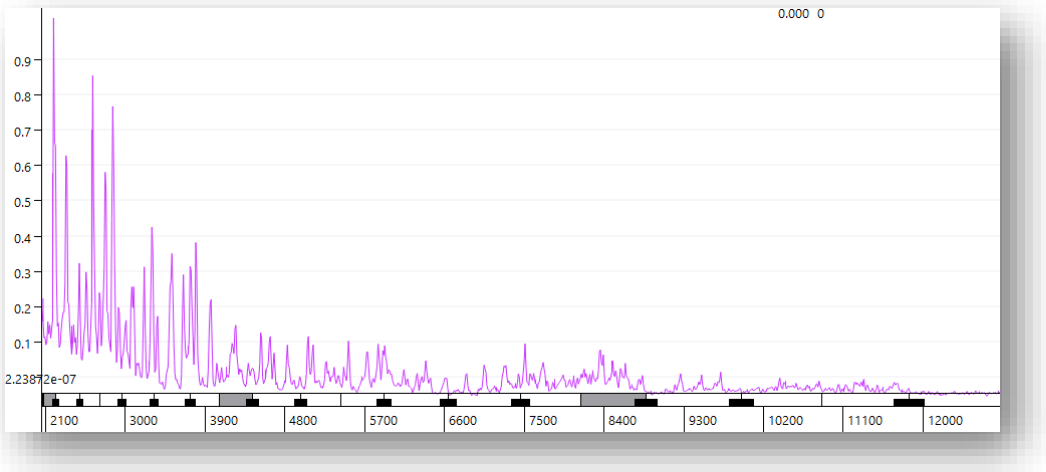


Figura 4 – Espectre d'ona [creació pròpia amb programari Sonic Visualizer].

Podríem definir l'espectre com una projecció de la senyal original, de la qual en podem obtenir la magnitud, que ens indica tots els sinusoides presents a la senyal i la fase, que ens indica la seva posició respecte del temps t=0. Val a dir que un espectre tindrà dues parts, corresponents a l'oscil·lació de la ona original: la positiva i la negativa. Les dues parts són simètriques, i per tant, per simplicitat representarem només una de les dues.

D'altra banda, hem vist la representació de l'espectre on l'eix y indica el valor de l'amplitud en un moment de temps t determinat. Aquesta s'obté computant el valor absolut del sinusoide original, el qual és un valor lineal. Una manera més intuïtiva de representar-ho és en una escala logarítmica, amb la qual podem visualitzar millor les oscil·lacions:

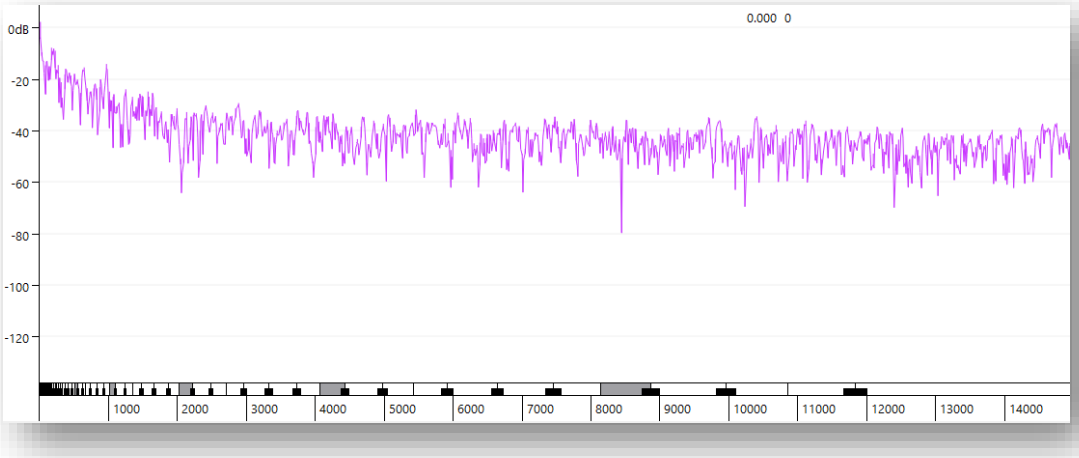


Figura 5 – Espectre d'ona en escala decibèlica [creació pròpia amb programari Sonic Visualizer].

En el camp del processament de so, habitualment es fa servir com a mesura de l'amplitud els decibels. Aquesta mesura és la més comunament acceptada i representa el nivell de pressió sonora d'una ona, entenent com a pressió sonora la diferència de

pressió atmosfèrica generada per una ona de so. Per calcular l'escala logarítmica dels decibels, es pren com a mesura de base una pressió sonora de referència i se'n calcula el logaritme:

$$L_p = 20 \log_{10}\left(\frac{p}{p_0}\right)$$

on p és la pressió sonora que volem transformar i $p0$ és la pressió de referència, $20\ uPa$. Aquesta referència està considerada com el llindar absolut d'audició [9], i per tant, en els espectres es representaran valors entre 0 i 20 microPascals, que són els valors que es considera que els humans som capaços de reconèixer.

Fast Fourier Transform i Inverse Discrete Fourier transform

El procés matemàtic de la DFT és complex i a pesar de que podem trobar gran quantitat d'algorismes per fer-ne una implementació eficient en un ordinador, el cost computacional sempre està a l'entorn de $O(n2)$. Per millorar la eficiència i velocitat de càlcul, en informàtica habitualment es fa servir una implementació especial de la DFT coneguda com a Transformació Ràpida de Fourier (FFT per les seves sigles en anglès). Diverses implementacions algorítmiques d'aquesta fórmula aconsegueixen costos propers a $O(N\ log\ N)$, millorant de manera qualitativa l'eficiència. Així doncs, podem veure en gran quantitat de *papers* publicats com, en realitat, la equació que es fa servir per fer l'anàlisi d'ona és la FFT.

De la mateixa manera que hem fet el procés d'anàlisi sobre una senyal gràcies a la DFT i la FFT, també podem fer el procés invers gràcies a la transformada Inversa Discreta de Fourier (IDFT, en anglès) en la qual donada una funció que representa l'espectre, la multipliquem per la funció base positiva, en fem el sumatori i per últim multipliquem el resultat per $1/N$ on N, com abans, és el nombre de mostres.

$$x[n] = \frac{1}{N}\sum_{N=0}^{N-1} X[k]e^{j2\pi kn/N} \qquad per\ a\ k = 0,1,\ldots,N-1$$

D'aquesta manera podem fer el procés de síntesi d'ona: a partir d'un espectre analitzat, obtenim el sinusoide original exacte del qual prové.

Short-Time Fourier Transform

Les transformacions de Fourier que hem vist fins ara s'apliquen únicament a un instant de temps; podríem dir que són com una fotografia, una "instantània" de les freqüències en un instant determinat. Amb això no en tenim prou per extreure la informació necessària per al nostre model; necessitem una representació continua de l'espectre al llarg de tot el temps que dura el nostre so. Aquesta representació la podem aconseguir gràcies a la transformació de Temps Reduït de Fourier (també anomenada Amb Finestra). En aquesta, dividim la senyal d'entrada en petits segments de la mateixa longitud i en calculem la DFT de cadascun, seguint la següent equació:

$$X_l[k] = \sum_{n=-N/2}^{\frac{N}{2}-1} w[n]\cdot x[n+lH]e^{-j2\pi kn/N} \qquad per\ a\ l = 0,1,\ldots$$

On tenim que:

- $W[n]$ és la finestra d'anàlisi.
- L és el numero de *frame*, el temps d'inici de l'anàlisi.
- H és el *Hop*, el valor de salt, que marcarà la mida del fragment analitzat.

L'equació de la STFT és pràcticament igual que la de la DFT, però afegint el concepte de **finestra d'anàlisi** [10]: la finestra és una funció que defineix una sèrie de valors en un interval; dins d'aquest interval, la funció filtra qualsevol valor, i fora d'aquest interval, els valors són 0. Al multiplicar per la funció finestra, obtenim una imatge acurada de les freqüències constituents, ja que filtrem aquelles que no ens interessen.

Cal vigilar, però amb els valors de la finestra: un valor de mostreig alt permet filtrar més freqüències, i per tant, tenir una imatge més precisa dels harmònics presents al so; un valor més baix, en canvi, ens ofereix una imatge amb més soroll però amb les marques de temps molt més definides. Aquesta particularitat de la mida de la finestra es podrà fer servir per afinar el model segons els tipus de sons que tinguem.

2.3. L'espectrograma

La forma sinusoidal de l'ona no ens dona suficient informació per crear un model per sí sola, ja que només ens ofereix una imatge de l'amplitud i la freqüència en un temps determinat. Per intentar captar totes les freqüències constituents d'un so, podem recórrer a l'espectrograma que es deriva del mateix. A partir de la STFT que hem vist abans podem obtenir una representació gràfica de les freqüències constituents del nostre so en cada moment de temps. Aquesta imatge l'anomenem espectrograma:

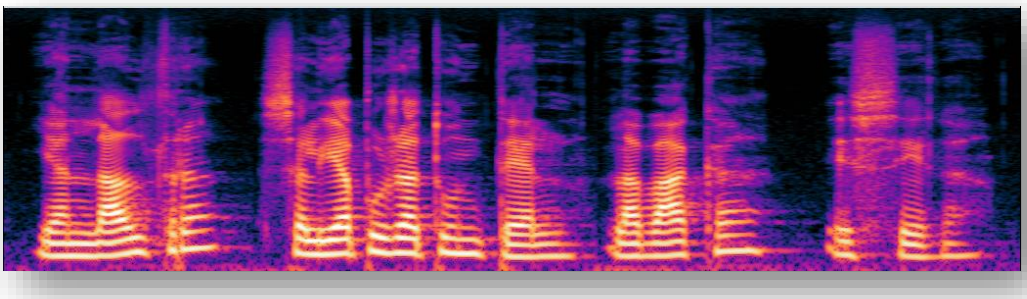


Figura 6 – Espectrograma [extret de fragment sonor amb programari Audacity].

La forma més habitual d'un espectrograma és una imatge en 2-dimensions, on l'eix vertical és la freqüència, l'eix horitzontal és el temps i els colors representen la magnitud de cada freqüència al llarg del temps.

Aquest espectrograma serà la base sobre la que construirem el nostre model. Tal com expliquem a l'apartat de metodologia [**enllaç**], els models es construiran en base a la similaritat entre dues imatges que representen el so concret. Aquestes imatges són els espectrogrames. El fet que l'espectrograma tingui aquest format el fa ideal per

treballar amb xarxes neurals convolucionals (CNN) i poder extreure'n les característiques principals, que permetran al model buscar similaritats entre sons.

Espectrograma de Mel

Per afinar encara més els resultats, els nostres espectrogrames contindran una representació logarítmica del valors de l'amplitud de les freqüències. En un espectrograma com el de la imatge, els valors de l'eix d'abscisses representen el valor absolut de les freqüències representades. L'ésser humà, però, té una tolerància auditiva significativa als canvis de freqüència i, per exemple, l'oïda no percep sempre igual la diferència entre dues freqüències. En concret, per freqüències baixes (fins a uns 1000Hz) els humans percebem molt clarament una petita diferència entre freqüències però aquesta mateixa diferència és pràcticament imperceptible en freqüències altes. Per posar un exemple concret, entre 1000Hz i 1100Hz, la diferència és molt més significativa que entre 10000Hz i 10100Hz, on ens serà pràcticament impossible distingir les dues freqüències clarament.

Per pal·liar aquest problema i fer més realista l'anàlisi de freqüències de l'espectrograma, el 1987 la *Journal of the Acoustical Society of America* [11] va implementar **l'escala de Mel**:

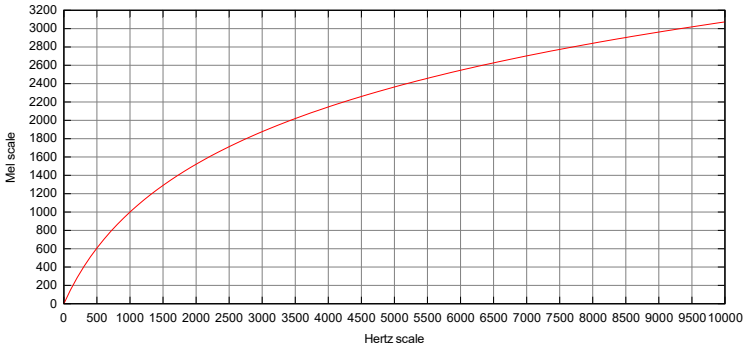


Figura 7 – Escala de Mel [Wikimedia Commons, original de Krishna Vedala].

De l'experiència de treballs similars [12], veiem que podem obtenir millors resultats si apliquem aquesta escala al espectrogrames, el qual dona lloc al que es coneix com espectrograma de Mel. Aquest és el tipus final d'imatge amb la qual treballarem.

2.4. Representació digital de l'àudio

Per últim, cal fer esment també de la manera com representem el so en un ordinador. Per a que les màquines puguin comprendre i manipular arxius d'àudio amb precisió en sistemes digitals ens calen una sèrie de processos per convertir les ones de so analògiques en dades digitals.

Analògic vs. Digital

El so analògic és continu i infinit, semblant a les fluctuacions naturals de les ones sonores a l'entorn. Habitualment entenem per analògic el so “natural”; en realitat, però, analògic es refereix a la naturalesa de la representació, i no al so en sí. En contrast, el so digital és discret i finit, representat com una sèrie de dígits binaris (bits). Aquesta representació implica *samplejar* la senyal analògica (crear-ne mostres d’intervals regulars, ja ho hem vist als apartats anteriors) i *quantitzar* cada mostra a un nivell d'amplitud específic. L'àudio digital ofereix avantatges en termes de precisió, manipulació i preservació comparat amb l'àudio analògic; a més amb la tecnologia actual el so digital pot arribar a tenir la mateixa qualitat (o fins i tot superior) que la representació analògica homòloga.

Conversió Analògic-Digital

Per convertir el so analògic a format digital, s'utilitza un convertidor analògic-digital (ADC per les seves sigles en anglès). L'ADC mostreja la senyal analògica a una taxa específica (mesurada en quilohertzs, kHz) i quantitza cada mostra en valors binaris. La precisió de la representació digital depèn de factors com la tasa de mostreig i la profunditat de bits (el nombre de bits utilitzats per representar cada mostra).

En el procés invers, per reproduir àudio digital s'utilitza un convertidor digital-analògic (DAC) per convertir les dades digitals en ones de so analògiques. El DAC reconstrueix la senyal analògica original interpolant entre les mostres digitals discretes, produint una sortida contínua que s'aproxima a l'ona original.

Formats i Extensions de Fitxers

Els fitxers d'àudio digital es guarden en diversos formats, cadascun amb els seus propis algoritmes de compressió i extensions de fitxers. Alguns dels fitxers d'àudio més populars inclouen MP3 (*MPEG Audio Layer III*, molt utilitzat en serveis *d’streaming* de música online), AAC (*Advanced Audio Coding*, es fa servir especialment com a pistes d’àudio en vídeos), FLAC (*Free Lossless Audio Codec*, àudio digital de gran qualitat, molt apreciat pels audiòfils) i WAV, que significa *Waveform Audio File Format*. Els fitxers WAV contenen típicament dades d'àudio sense compressió, preservant la qualitat de so original però resultant en mides de fitxer més grans comparades amb formats comprimits com MP3. A causa de la seva naturalesa sense pèrdua, els fitxers WAV sovint són preferits per a la producció d'àudio professional. Aquest tipus de format és capaç de generar espectrogrames molt realistes i amb gran precisió; és el format, doncs, que farem servir per les pistes d’àudio del nostre model.

3. Creació del model

Després de fer una introducció teòrica, podem començar a bastir el model en base a tot el que hem aprés. Començarem explicant la importància del conjunt de dades per aquest tipus de treball, després introduirem els conceptes més importants sobre xarxes neurals i aprenentatge profund adaptat a la classificació d'àudio, juntament amb un repàs a alguns dels treballs d'investigació més destacats dels darrers anys, i acabarem muntant el projecte del model de classificació, que penjarem en un repositori de GitHub.

3.1. Tria i justificació del conjunt de dades

Com ja hem explicat, l'objectiu del treball és crear un model d'inferència mitjançant tècniques d'aprenentatge profund el qual permeti reconèixer sons a partir de fragments d'àudio.
En general, sabem de la importància dels conjunts de dades per l'èxit del model; en el nostre cas l'abast del projecte, degut al temps i les eines de què es disposa, és limitat, i de cara a poder desenvolupar correctament el model en totes les seves fases, tot mirant d'obtenir uns bons resultats, és especialment rellevant la tria del conjunt de dades d'entrenament del model.

Els conjunts

Actualment, hi ha una sèrie de conjunts de dades disponibles a Internet enfocats al processament i reconeixement de so. Aquests conjunts han de tenir unes característiques determinades: en concret, han d'estar formats per parells d'elements $\{audio, metadades\}$, on $audio$ és un arxiu sonor en format *.wav*, de durada curta (entre 1 i 15 segons com a màxim) i $metadades$ és un arxiu de text (habitualment en format .txt) que conté una descripció textual del so. Per simplicitat del treball, s'espera que les pistes d'àudio siguin en mono (tot i que en un conjunt amb pistes en estèreo també podríem fer-ne la conversió a mono abans de processar-les), d'una durada més o menys igual entre elles, i que els arxius de text tinguin una estructura similar entre ells.
Els conjunts més representatius a dia d'avui, i que es podrien adaptar millor a les nostres necessitats són:

- **ESC-50** [13], és un conjunt compost de 2000 mostres de 5 segons de durada, organitzades en 50 classes, amb 40 sons per classe; està enfocat a la classificació de so de l'entorn, i les mostres es divideixen en 5 grans grups: animals, escenes naturals, sons humans no-verbals, sons domèstic i sons urbans. Els sons estan extrets de Freesound.org i seleccionats manualment.
- **UrbanSound8k** [14], compta amb 8732 sons de diferent durada (però de 4 segons com a màxim), classificats en 10 tipus de so; les classes són bastant més concretes que les del ESC-50. Les metadades venen en un arxiu en format *.csv* i contenen, apart de la descripció textual del so, informació variada sobre el mateix. Així mateix, UrbanSound8k té la

particularitat que totes les mostres venen organitzades en 10 carpetes, amb les dades expressament disposades així per poder aplicar un sistema de validació creuada de 10 subconjunts (*Fold Cross Validation*).

- **FSD50K [15],** probablement el més complert disponible (sense comptar conjunts que inclouen vídeo), amb prop de 51000 mostres etiquetades en prop de 500 classes diferents. A més de la quantitat de dades disponibles, el FSD50K resulta especialment interessant ja que les fonts d'obtenció dels fragments d'àudio són molt diverses, i de diversa qualitat: està pensat per contemplar tots els possibles escenaris en un entorn realista: el so no sempre té la mateixa qualitat, ni durada, ni prové del mateix lloc; igual passa amb les metadades, no tenen una estructura clara, el que obliga a fer un tractament previ molt intens del conjunt.

Òbviament hi ha més conjunts disponibles en format de codi obert (*open-source)*: el DESED **[16],** molt enfocat en sons domèstics i d'interior, el MS SNSD **[17]**, que conté fragments de parla barrejats amb sons ambientals, etc. Creiem, però que aquests tres són els més representatius de la direcció del treball, i en concret, ens decantem per el UrbanSound8K, tot i que no descartem poder utilitzar recursos creuats dels altres dos per comprovar l'eficàcia del model final.

Estructura

El UrbanSound8K està compost per un total de 8132 fragments d'àudio, que es classifiquen en 10 tipus de so urbà, present principalment en entorns de ciutats densament poblades. Cada classe té un identificador, del 0 al 9, com segueix:

0 = aire condicionat (la màquina que està en els exteriors dels edificis)
1 = clàxon de cotxe
2 = nens jugant (no distingeix volum, parla, noi o noia)
3 = gos bordant
4 = trepant
5 = motor de cotxe (al ralentí, mentre espera en un semàfor, per exemple)
6 = tret de pistola (no distingeix tipus de pistola ni de bala)
7 = martell hidràulic
8 = sirena (no distingeix tipus de sirena ni de vehicle)
9 = musica al carrer (una banda tocant o algú amb un aparell de radio)

Les metadades venen en un fitxer *.csv* que representa un *dataframe* on cada fila correspon a un dels fragments d'àudio i la seva informació.

3.2. Metodologia

Abans de començar a treballar en el model, farem una ràpida aproximació a la metodologia que farem servir.

Gràcies a les llibreries i programari disponibles en python podrem analitzar les característiques físiques de l'arxiu de so del conjunt: freqüències, amplitud i anàlisi d'espectrograma. Mitjançant un mètode similar al de **CLIP** (*Contrastive Language Image Processing*), en el qual s'etiqueten conjunts d'imatges amb una etiqueta de text que defineix el que hi ha a la imatge, es crearà un mètode d'aprenentatge auto-supervisat, que alguns autors han batejat com a **CLAP** (*Contrastive Audio Language Processing*).

La base del model seran les xarxes neurals convolucionals (**CNN**). Les CNN són una arquitectura d'aprenentatge profund molt utilitzada en la visió per computador i reconeixement d'imatges. La xarxa neural utilitza un filtre (*kernel*) que recorre una imatge d'entrada i hi aplica una sèrie d'operacions matemàtiques que generen el que es coneix com convolució, procés amb el qual es poden extraure les característiques més importants de cada part de la imatge i finalment, de la imatge sencera.

Aquí farem servir una arquitectura de xarxes neurals convolucionals per crear els mapes de característiques (*feature maps*) corresponents a la forma sonora de cada arxiu; codificarem text i àudio per separat i implementarem un classificador lineal per crear la inferència.

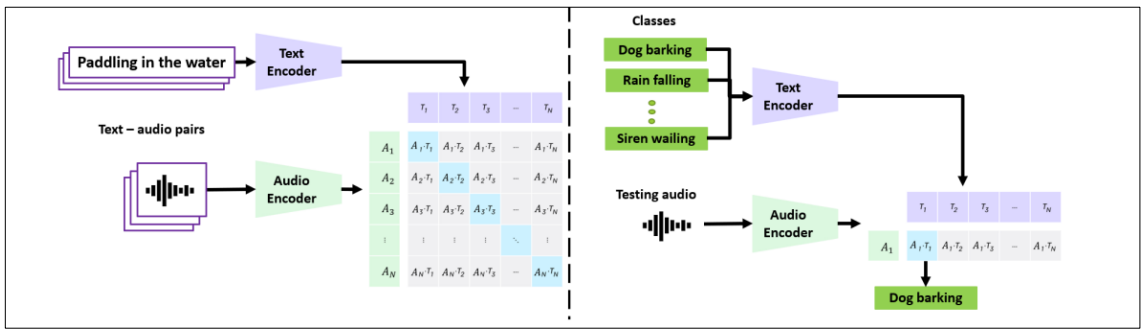


Figura 8 – Esquema de funcionament del sistema CLAP. Extret de ***Learning Audio Concepts From Natural Language Supervision*** *[18].*

3.3. Estat de l'art

Els últims 10 anys han proliferat gran quantitat de treballs i estudis al voltant del reconeixement de so gràcies a solucions de IA. Cadascun a la seva manera, tots aporten diferents enfocaments pràctics amb un rerefons similar al que hem explicat en l'apartat anterior: la transformació d'una pista d'àudio en espectrograma per poder treballar amb ella com si fos una imatge, aprofitant la gran potència i la eficàcia sobradament reconeguda de les xarxes neurals convolucionals per processar i classificar imatges.

Les CNN i les imatges

Les xarxes neurals s'han demostrat com una de les millors arquitectures per al reconeixement d'imatges en 2D. Com ja vam veure a l'assignatura de Aprenentatge Computacional, les CNN són un tipus de xarxa neural que consten de diverses capes amb funcions diferents cadascuna, i que van filtrant progressivament les imatges per

aconseguir extraure'n les seves característiques intrínseques i per tant, poden ajudar a identificar-les i distingir-les entre elles. Aquesta extracció de característiques (*feature extraction*, en anglès) es realitza de forma jeràrquica a través de les capes convolucionals, on a cada convolució es va afinant la imatge, reconeixent cada vegada parts, textures o patrons més complicats a mida que s'aprofundeix en la xarxa.

Apart de la capa convolucional, que és on es realitza principalment la extracció de característiques, les CNN disposen també de :

- Una funció d'activació **ReLU** (*Rectified Linear Unit*): Aquesta funció d'activació s'aplica després de la convolució i introdueix no linealitat al model, el que permet a la xarxa aprendre relacions més complexes. La funció **ReLU** es defineix com:

  $$\boldsymbol{ReLU(x) = max(0, x)}$$

  La funció pren un valor $\boldsymbol{x}$ d'entrada i retorna $\boldsymbol{x}$ si $\boldsymbol{x}$ és positiu, i $\boldsymbol{0}$ si $\boldsymbol{x}$ és negatiu.

- Una capa d'agrupació o ***pooling*** (*max-pooling* o *average-pooling*), que serveix per reduir la dimensionalitat de manera controlada, per agilitzar els càlculs I per aportar invariància al model, evitant així el sobre ajustament (*overfitting*).

- Una segona xarxa de **capes totalment connectades** (*fully-connected layers*) que es situen al final de la CNN, agafa el resultat de la última convolució i en fa els càlculs pertinents, permetent la posterior tasca de classificació o regressió.

- Un procés d'aplanament (***flattening)***, just abans de les capes totalment connectades, que converteix els tensors de sortida de les capes convolucionals, que són multidimensionals (habitualment en el cas de les imatges, de 3 dimensions, $\boldsymbol{hxwxd}$, on $\boldsymbol{h}$ i $\boldsymbol{w}$ són l'alçada i amplada de la imatge en 2D i $\boldsymbol{d}$ és el nombre de canals o característiques) en vectors unidimensionals que es poden tractar com arrays permetent a l'ordinador fer els càlculs necessaris.

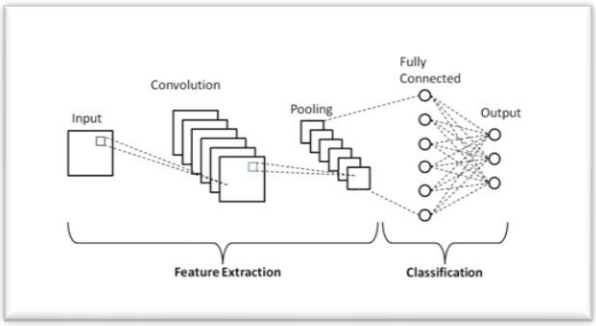


Figura 9 – Estructura bàsica d'una CNN. Extret de ***A High-Accuracy Model Average Ensemble of Convolutional Neural Networks for Classification of Cloud Image Patches on Small Datasets.*** PHUNG Van Hiep, RHEE Eun Joo .

El procés

Amb tots els elements descrits al punt anterior podem posar en marxa una CNN. Es fa una inicialització aleatòria dels pesos (*weights*), que són els valors que es faran servir en cada node de la xarxa per calcular el valor de sortida en funció de l'entrada.

Es prossegueix amb el mètode conegut com a propagació endavant (*forward-propagation*) on els valors d'entrada recorren la xarxa fins que s'obté una predicció. El

model calcula llavors la funció de pèrdua (*loss function*), segons les diferències entre la predicció i la classe real de la imatge etiquetada durant l'entrenament. Amb aquesta funció, es calcula el gradient dels pesos, que ens servirà per ajustar els pesos dels nodes per minimitzar la funció de pèrdua, i en conseqüència millorar la precisió del model. Per fer-ho es fa servir el mètode conegut com propagació enrere (*backpropagation*) en el qual es busca en cada punt de la xarxa quina ha estat la pèrdua i s'optimitza el gradient per tornar a ajustar els pesos i actualitzar-lo.

Cada cicle complet d'aquest procés s'anomena època (*epoch*); en un procés d'entrenament d'un model correrem el model durant vàries èpoques per millorar les mètriques d'avaluació. A mida que augmentem les èpoques, el descens del gradient optimitza millor els pesos i ajuda augmentar la precisió de les prediccions. Haurem de vigilar, però, amb les èpoques: un nombre massa petit no donarà prou informació al model per classificar correctament; un nombre massa gran farà que els pesos siguin massa ajustats a les dades concretes d'entrenament i dificultarà la generalització del mateix (la seva capacitat per reconèixer i classificar dades diferents de les del conjunt d'entrenament).

Les CNN en el reconeixement d'àudio

Partint de la base del que hem explicat en el punt anterior, s'han dut a terme nombrosos experiments utilitzant les CNN per al reconeixement d'àudio a partir de l'espectrograma del so.

La idea és realitzar el procés d'anàlisi de Fourier a una pista d'àudio (un arxiu sonor a l'ordinador, habitualment en format *.wav*) i extreure'n les seves freqüències constituents i com aquestes es posicionen en el temps, és a dir, extreure'n l'espectrograma, tal com ja hem vist en el punt 2 del treball. Aquest espectrograma, per l'ordinador, no és res més que una imatge en 2 dimensions, i per tant pot ser tractada a traves d'una CNN com a tal. D'aquesta manera, la xarxa extreu les característiques úniques de la imatge, i en conseqüència, del so que ens interessa. Amb aquestes dades, un model entrenat pot reconèixer i classificar posteriorment un altre so similar, comparant les dues imatges. Aquest mètode s'ha mostrat molt eficient en el reconeixement d'àudio**[19]** , especialment en el camp del reconeixement de la parla, en el qual són la base de la gran majoria de treballs des de mitjans de 2010 **[20]**.

Apart de gran quantitat de treballs acadèmics en la matèria, a dia d'avui hi ha diverses iniciatives en marxa per fomentar l'aplicació d'aquestes solucions (o d'altres) al reconeixement d'àudio; potser la més significativa són els DCASE Challenges, un event que s'organitza cada any des del 2013 per diverses universitats de prestigi internacional(el primer es va celebrar a la Queen Mary University of London) en el qual

es proposen diverses tasques en forma de reptes de diferent dificultat. Del DCASE han sortit alguns dels avenços més importants en aquest camp.

La metodologia Contrastive Language

La segona part del reconeixement és el tractament de les etiquetes (labels) de text que acompanyen cada pista de so. El model creat ha de ser capaç de processar cada etiqueta de text i comparar-la amb el so per retornar la classe correcta de la pista d'àudio. En aquest punt trobem diversitat de solucions per fer aquest tractament, tot i que nosaltres ens basarem en el mètode de llenguatge contrastiu(c*ontrastive-language)*.

En concret, ens interessa el CLIP (*Contrastive Language Image Processing*), un tipus de mètode d'aprenentatge auto-supervisat (*self-supervised*) que entrenem amb grans quantitats de dades convenientment etiquetades amb fitxers de text i que posteriorment es capaç d'inferir, donada una imatge, el text que pot definir-la millor.

Durant l'entrenament, aquests models agafen els inputs separats dels fragments sonors i les metadades amb el text descriptiu (que d'entrada tenen dimensionalitats diferents) i els codifiquen per projectar-los en un mateix espai, anomenat *espai multimodal*. Un cop en aquest espai, es pot treballar amb ambdós conjunts de dades per crear una matriu de similaritats que serveix per classificar els fragments en el moment de fer la inferència.

El model CLAP

L'equip de recerca de Microsoft va desenvolupar el 2022 un mètode que van anomenar CLAP(*Contrastive Language Audio Processing*) **[18],** una adaptació del CLIP per generar el text que defineix un espectrograma, i per tant, el so que aquest representa. A diferència dels models anteriors de reconeixement d'àudio, capaços de classificar un so en base a un identificador de classe, que podia ser una paraula o una petita frase, el CLAP processa les etiquetes de metadades dels conjunts amb una eina de supervisió de llenguatge natural que li permet generar *prompts* o frases senceres que defineixen l'escena sonora amb diferents nivells de precisió. Això pot ser una eina especialment útil per tasques com la generació de subtítols amb audiodescripció i que obra la porta a la integració del so en el món de la generació de continguts amb Intel·ligència Artificial.

Altres models han presentat grans avenços també en aquest camp, com és el cas del Wav2Clip **[23]**, un dels primers estudis on es va provar extensivament l'eficàcia d'aquest mètode en els diferents conjunts de dades disponibles.

En ambdós casos s'ha posat molt interès en la capacitat dels models de fer prediccions zero-shot; això significa que tant CLAP com Wav2CLIP poden generalitzar a noves classes d'àudio utilitzant descripcions textuals, eliminant la necessitat de dades etiquetades específiques per a cada nova classe que es vol reconèixer. Aquest és un dels camps on més feina hi ha per fer, ja que a pesar de les grans possibilitats que suposa, els nivells de precisió dels models encara no és prou bo ni molt menys superior al de l'aprenentatge supervisat. Els recents avenços en el camp dels llenguatges de gran escala (LLM, Large Language Models) poden donar peu a experimentar amb la

implementació dels *embeddings* resultants d'aquests models per millorar aquesta capacitat, i per tant, aconseguir que el procés sigui més natural, més semblant a l'humà.

3.4. Creació del model

Finalment, doncs, es decideix crear un model base per fer les proves, basat, com hem explicat a l'apartat anterior, en una CNN, que serà el nucli del processament del model. Enlloc de crear aquest model des de zero, s'ha decidit aprofitar alguns algoritmes que s'han trobat derivats de diverses fonts (articles, grups de recerca, publicacions especialitzades, etc.). Finalment, s'ha creat una base aprofitant alguns d'aquests algoritmes i afegint altres classes i característiques. El resultat final està disponible online en el meu repositori personal: https://github.com/Gardenou/TFG_IA_Model . En la bibliografia estan especificats els enllaços als recursos utilitzats **[Annex].**
Per claredat, he decidit dividir el projecte en les següents classes, cadascuna de les quals compleix una funció concreta:

- La classe **dataLoader** carrega les dades del conjunt i els aplica un preprocessament. Les dades queden preparades per treballar-hi directament amb el model.
- La classe **loadAudio** defineix els mètodes de preprocessament de les dades, que veurem al següent apartat.
- La classe **TrainingData** carrega les dades del conjunt disponible (en el nostre cas, el UrbanSound8K) des del directori on estan desades i genera els conjunts d'entrenament i validació.
- La classe **TrainingLoop** defineix el procés d'entrenament del model.
- La classe **Model** defineix el model pròpiament dit, basat en les CNN.
- La classe **Inference** extreu les prediccions del model a partir de les dades de validació.
- Finalment, les classes **run** i **train** serveixen per llençar totes les etapes del model en ordre; **train** realitza l'entrenament i guarda els pesos en un arxiu i **run** carrega aquests arxius de pesos per crear les prediccions del model.

Per completar el projecte, afegim més endavant dos classes utilitàries:

- Amb ***calculateMetrics*** realitzarem una sèrie de càlculs bàsics de les mètriques d'avaluació; es crea primer la matriu de confusió amb els resultats de les prediccions i s'utilitzen els valors per trobar la precisió (*precision*), la puntuació F1 (*F1-score*) i l'exhaustivitat (*recall*).
- La classe ***plotSpectrogram*** és una classe d'ajuda amb la que podem veure els espectrogrames per pantalla, traient-los directament del joc de dades.

Processament de les dades

Abans de passar les dades al model, caldrà fer un preprocessament d'aquestes per adaptar-les i millorar els resultats. Aquest acostuma a ser un dels punts de la

metodologia CRISP-DM, de la qual ja hem parlat abans, i en aquest cas està plenament justificat. Dividirem el procés en dues parts: tractament del format de les dades i augment de les dades (*data augmentation*).
Per començar caldrà veure com estar organitzat el conjunt; en el nostre cas ja hem explicat que disposem de 10 carpetes que corresponen als *folds* típics d'un sistema de validació creuada *(k-fold cross validation*). Apart, tenim una carpeta més que inclou els fitxers amb les metadades en format *.csv*. Analitzem aquest fitxer i muntarem la classe **dataLoader** per extreure'n la informació corresponent. Abans de començar, podem extreure un primer espectrograma d'un canal del fragment de so original:

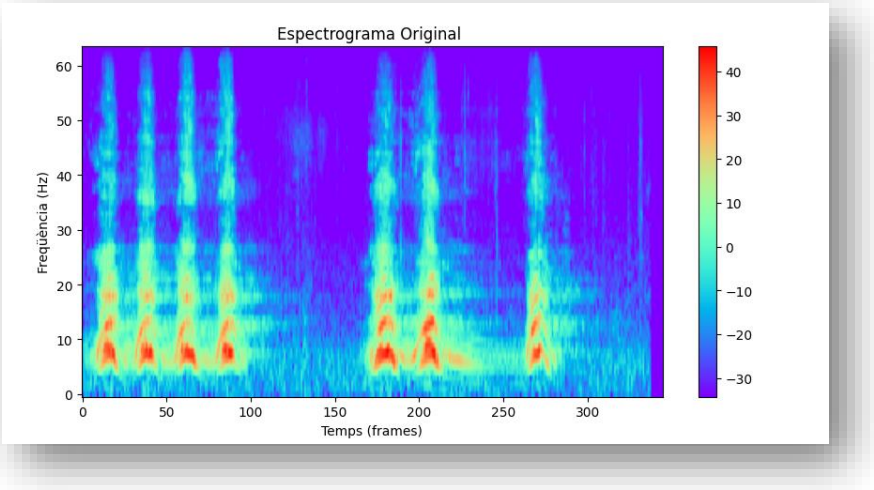


Figura 10 – Espectrograma generat a partir d'un arxiu sonor del conjunt UrbanSound8K (arxiu 74723-3-0-0.wav).

En el preprocessament haurem d'igualar els fragments d'àudio; l'experiència de la gran majoria de treballs consultats ens diu que s'obtenen millors resultats quan els fragments sonors són més semblants. Aquest fet no deixa de ser similar al procés d'estandardització o normalització que podríem fer en un altre conjunt de dades preparat per fer mineria. En el nostre cas, posarem el focus en els següents característiques:

- Primer de tot, igualarem el nombre de canals de cada pista. Algunes pistes són en mono (un canal) i d'altres són estèreo(dos canals). En el cas de les pistes de dos canals sabem que ambdós són iguals (contenen la mateixa informació), però pot ser que l'espectrograma sigui lleugerament diferent, degut al sistema de gravació; ens podem trobar per exemple, que el so s'hagi enregistrat amb dos micròfons, i tot i que el so sigui el mateix, la posició del micròfon respecte del so podria fer-lo variar. Les pistes estèreo doncs les deixarem tal com estan, i a les mono, duplicarem el canal per que en tinguin dos.
- Estabilitzarem també la freqüència de mostreig (*sampling rate*), que marca la precisió de la representació sonora; no totes les pistes tenien la mateixa, la qual cosa impedeix fer els càlculs de la CNN correctament (ja que els *arrays* que representen cada fitxer no tenen la mateixa longitud). Després de fer alguns primers experiments guiats per la literatura disponible, igualarem totes les pistes a 44100 Hz.

- La longitud de la pista (la durada); la durada mitjana aproximada de totes les pistes és d'uns 4 segons. Així doncs, truncarem els arxius que durin més de 4 segons, o omplirem amb *zero-padding* els que no hi arribin.

Un cop fet aquest primer processament, introduirem lleugeres variacions en les dades disponibles en un procés que es coneix com *data augmentation*. Aquest procés és força freqüent en l'entrenament de models de tractament d'imatges, i s'ha demostrat molt efectiu per millorar la precisió. En el cas de les imatges, s'introdueixen lleugeres variacions en la longitud o amplitud, o es fan petites rotacions o s'introdueix soroll en la imatge; d'aquesta manera el model s'acostuma a reconèixer petites variacions en les dades i així generalitza millor. En el cas de l'àudio, els augments tradicionalment es feien sobre els arxius sonors, ja que amb els espectrogrames una lleugera variació pot confondre el model i llençar resultats pitjors; cal recordar que un espectrograma, a pesar de que el tractem com una imatge, no deixa de ser una representació espai-temporal, i la posició dels píxels en la imatge defineix el caràcter únic del so. Diversos estudis han treballat sobre aquest tema els últims anys, i en concret aquest de l'equip de recerca d'àudio de Google **[21]** del 2019 va introduir la idea d'augmentar l'espectrograma aplicant tres transformacions bàsiques:

1. *Deformació longitudinal* (al llarg de l'eix x, l'eix temporal): el dibuix s'estira o es contrau en el temps, de manera similar a com es fa una transformació tímbrica (*pitch deformation*, molt utilitzada en efectes especials sonors), de manera que canvia lleugerament la posició i ocurrència de cada una de les freqüències, però no la estructura general del so. També es coneix com *time shift*. En aquest punt, podem veure com queda una imatge final preprocessada, on podem apreciar el desplaçament produït per aquest últim mètode *:*

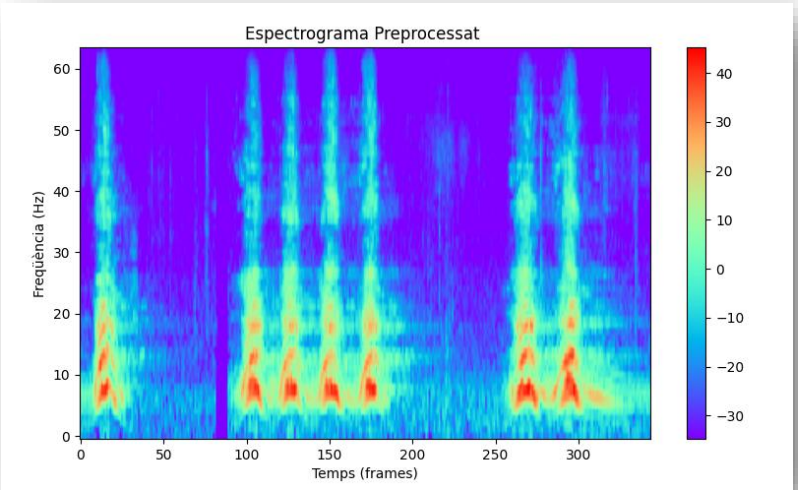


Figura 11 – Espectrograma amb preprocessament generat a partir d'un arxiu sonor del conjunt UrbanSound8K.

2. *Emmascarament de freqüències*: s'introdueixen línies horitzontals aleatòries amb l'objectiu d'eliminar una sèrie de freqüències en concret, les quals, al ser la

línia completament recta, sempre seran consecutives i no afectaran a la resta de freqüències del so.

3. *Emmascarament temporal*: s'introdueixen línies verticals aleatòries per eliminar ocurrències temporals de les freqüències; amb això també modifiquem lleugerament la imatge, però sense desequilibrar l'estructura global sonora.

Addicionalment, es pot també afegir soroll aleatori (però de manera regular) al llarg de l'espectrograma, o afegir-hi silencis periòdics; aquestes dues tècniques han demostrat gran eficàcia en models de reconeixement de veu, però aquest no és l'objectiu d'aquest treball i per tant, no els farem servir.
Totes les transformacions mencionades es poden implementar amb el paquet **transforms**, inclòs a la llibreria *Pytorch*.
Un cop fet tot el procés d'augment de dades, mostrem la imatge del mateix fragment:

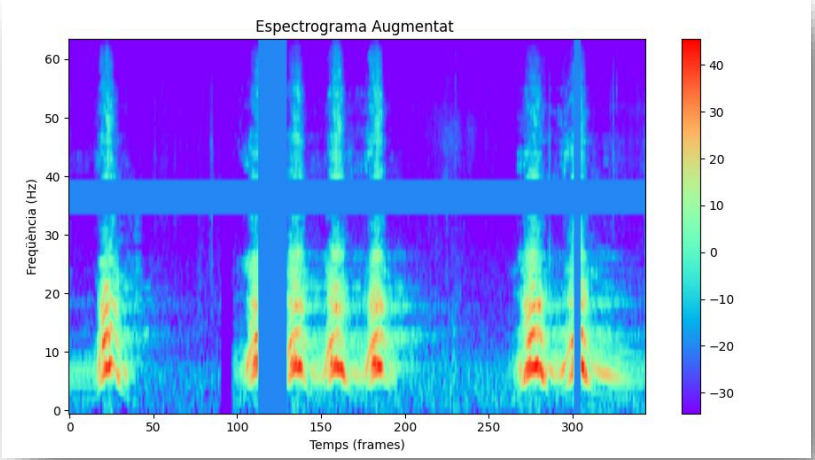


Figura 12 – Espectrograma amb SpecAugment i preprocessament, generat a partir d'un arxiu sonor del conjunt UrbanSound8K.

Podem apreciar-hi les línies verticals i horitzontals que representen l'emmascarament temporal i de freqüències, introduïts de manera aleatòria.

El model

En l'apartat anterior hem definit l'arquitectura i el funcionament bàsic de les CNN. En aquest projecte, implementem tots aquests elements per crear el nostre model. En concret, disposem de:

- Quatre xarxes convolucionals, que anomenarem blocs. Els blocs seran els encarregats de tractar la informació. Cada bloc està estructurat en cinc capes:

```python
# First Convolution Block
self.conv1 = nn.Conv2d(2, 8, kernel_size=(5, 5), stride=(2, 2), padding=(2, 2))
self.relu1 = nn.ReLU()
self.bn1 = nn.BatchNorm2d(8)
init.kaiming_normal_(self.conv1.weight, a=0.1)
self.conv1.bias.data.zero_()
conv_layers += [self.conv1, self.relu1, self.bn1]
```

Figura 13 – Esquema en Python d'un bloc convolucional bàsic del model.

- La primera capa és la xarxa neural pròpiament dita, implementada amb **Conv2d**; els paràmetres inicials són, lògicament, els dos canals d'àudio (recordem que hem transformat totes les pistes a estèreo), els quals convolucionarem a 8 canals de sortida. Al següent bloc li passarem aquests 8 canals com a entrada i en retornarem 16, i així successivament fins arribar als 64 canals de sortida del quart bloc.
- Posteriorment, apliquem la funció **ReLU**,tal com hem explicat abans.
- Tot seguit, normalitzarem els lots amb la funció **BatchNorm2d**. Aquest tipus de normalització per lots és bastant utilitzada en les CNN per tractament d'imatges; el seu objectiu és estandarditzar i escalar la sortida en funció de la mitjana i la desviació estàndard del conjunt del lot.
- El següent pas serà inicialitzar els pesos de cada neurona amb una distribució normal, en aquest cas fent servir una inicialització de tipus **Kaiming**.
- Per últim, posem a zero els valors d'esbiaix (*bias*) de cada neurona abans de començar a fer els càlculs.

Tot plegat, acabarem agregant les capes a l'estructura *conv_layers* per executar-ho posteriorment de manera seqüencial.

- Una capa de *pooling*, just a continuació de la sortida dels blocs; provarem amb diversos mètodes, però comencem de base amb un **AveragePooling**.
- El procés d'aplanament (*flattening*) per convertir els tensors en dades lineals.
- El classificador lineal, que fa el tractament del text descriptiu i donades les dades de la sortida anterior ens retorna la predicció final.

Per implementar cada procés farem servir principalment l'àmplia varietat de mètodes inclosos a *torchaudio*, concretament els definits al paquet ***nn***, que incorpora tota mena de processos per treballar amb xarxes neurals.

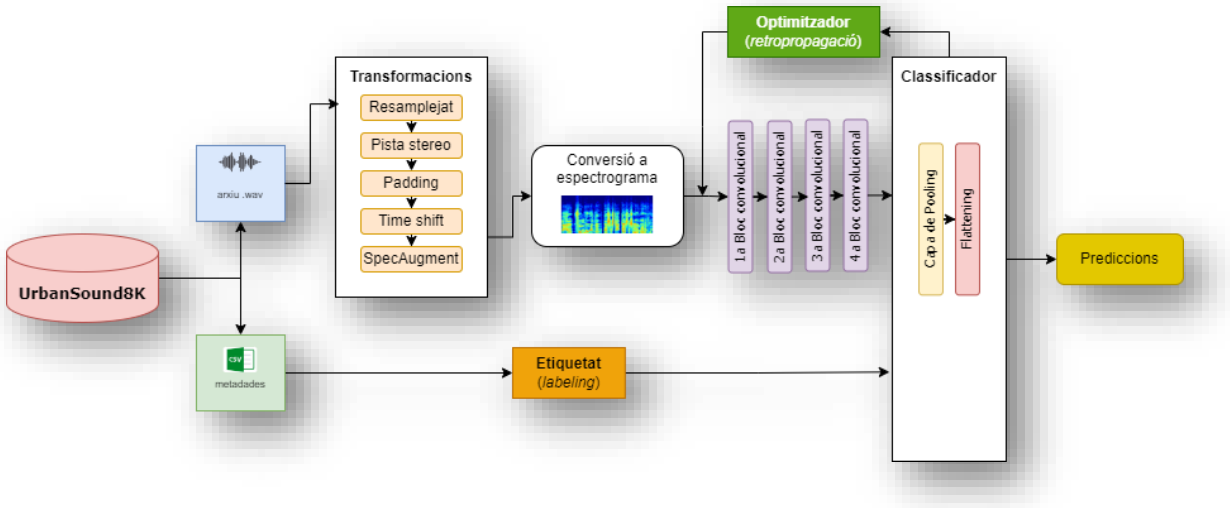


Figura 14 – Diagrama de l'estructura del model construït per al treball.

Entrenament del model

Un cop s'han tractat les dades adequadament i hem implementat el model, la següent fase del procés és entrenar-lo amb una part del conjunt de dades disponibles al joc. En concret, hem seguit la norma bastant estesa de separació de conjunts del **80/20**, un 80% del total del conjunt per entrenar el model i un 20% per validar-lo i avaluar la capacitat d'inferència.

Habitualment en aquesta part del procés cal buscar un mètode per barrejar primer les dades convenientment de manera completament aleatòria per evitar sobre ajustaments i esbiaxios en el model resultant. L'UrbanSound, però, ja està prèviament separat en carpetes amb mostres aleatòries però equilibrades segons les diferents classes.

En la classe *TrainingData* definim les rutes en local on tenim les dades disponibles, i extraiem de l'arxiu de metadades la informació i ruta específica de cada arxiu sonor, així com la seva categoria. Després separem les dades en els dos conjunts mencionats, *train_dl* i *val_dl*; cada conjunt estarà composat de paquets que anomenem lots(*batches*), que la xarxa neural processarà per tongades i de manera aleatòria. Cadascun d'aquests lots està format per parells de tensors del tipus $\{audio, metadades\}$ amb la informació que passarem a les xarxes d'entrada. Cada lot conté 16 mostres, tot i que hem fet algunes proves augmentant o disminuint el tamany, principalment de cara a millorar la eficiència de la GPU i així reduir el temps de cada entrenament.

A la classe *TrainingLoop* crearem el bucle d'entrenament, amb la següent estructura:

- La **funció de pèrdua**, en el cas del processament d'imatges, i també per la classificació d'àudio s'ha comprovat [22] que la funció que ofereix més bons resultats és la funció de pèrdua d'entropia categòrica, que en torchaudio podem implementar amb **Cross Entropy Loss**.

- **L'optimitzador**; utilitzarem d'entrada l'optimitzador Adam, tot i que en els diversos treballs consultats hem trobat gran quantitat d'usos d'altres mètodes, els quals compararem quan fem el refinament del model.
- El **planificador** (scheduler), en el qual definirem la tasa d'aprenentatge, les èpoques i l'estrategia d'atenuació lineal, amb la qual controlarem la variabilitat i de la tasa d'aprenentatge al llarg de l'entrenament.
- El bucle pròpiament, en el qual separarem audio i text per enviar-ho al model, per posteriorment recollir els resultats i propagar-los enrere. Finalment, extraiem la predicció de la classe agafant l'etiqueta amb el valor més alt de similaritat.

```python
def training(model, train_dl, max_epochs):
    # Funcions de pèrdua, optimització i planificador
    criterion = nn.CrossEntropyLoss()
    optimizer = torch.optim.Adam(model.parameters(), lr=0.01)
    scheduler = torch.optim.lr_scheduler.OneCycleLR(optimizer, max_lr=0.001,
                                                    steps_per_epoch=int(len(train_dl)),
                                                    epochs=max_epochs,
                                                    anneal_strategy='linear')
    # Bucle segons èpoques
    for epoch in range(max_epochs):
        running_loss = 0.0
        correct_prediction = 0
        total_prediction = 0

        # Bucle per cada lot del conjunt
        for i, data in enumerate(train_dl):
        # Passa les dades separades en audio i metadades al model
        inputs, labels = data[0].to(mo.device), data[1].to(mo.device)

        # Normalització dels vectors de audio
        inputs_m, inputs_s = inputs.mean(), inputs.std()
        inputs = (inputs - inputs_m) / inputs_s

        # Posem a zero el gradient de l'optimitzador en cada volta
        optimizer.zero_grad()

        # procés d'optimització / backpropagation
        outputs = model(inputs)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()
        scheduler.step()

        running_loss += loss.item()

        # Busquem la classe amb major puntuació (score)
        _, prediction = torch.max(outputs, 1)
```

Figura 15 – Codi corresponent al procés d'entrenament del model.

Originalment hem definit una quantitat fixa d'èpoques, però hem comprovat que era poc pràctic haver de modificar-ne cada vegada la quantitat en cada entrenament, més tenint en compte que la convergència del model canvia amb cada experiment; hem optat doncs, per executar el bucle d'entrenament amb un màxim de 100 èpoques (en els diferents experiments no hem passat mai de les 80) i un paràmetre paciència (*patience*). En el moment en què el model comença a perdre precisió, mantenim el bucle fins la quantitat d'iteracions marcades per aquest paràmetre; arribat aquest punt, considerem que el model ja no millora els resultats sinó que empitjora, i per tant, recuperem els resultats de la última millor època. Addicionalment, s'ha creat un fitxer *.csv* per guardar dades de cadascun dels experiments duts a terme.

Inferència

Un cop acabat l'entrenament, des de la classe *train* guardem en un fitxer els pesos resultants per reutilitzar-los posteriorment. L'entrenament s'ha realitzat sobre una màquina PC de sobretaula, amb una CPU Intel i3-10300K, 32Gb de memòria RAM i una GPU Nvidia GTX1650Ti. Amb els arxius de pesos resultants s'ha pogut utilitzar el model en màquines menys potents per poder fer els diversos experiments en diversos entorns. Com ja hem explicat abans, per fer la inferència hem fet servir el conjunt de dades de validació creat en la classe *TrainingData;* hem creat un bucle en què anem passant els lots d'aquest conjunt al model, separats en el fragment de so i l'etiqueta de text. La classe *inference* retorna un tensor de 10 valors puntuats (*scores*), i compara el valor més alt amb la classe real del fragment sonor (*ground truth*). Finalment es fa el càlcul de les mètriques amb la funció descrita anteriorment.

Avaluació del model

Amb tots els elements anteriors, engeguem el projecte amb la classe *train* per fer el primer entrenament amb el conjunt de dades *train_dl* i guardar els pesos corresponents. Per començar hem llençat la classe *run* que ens farà les prediccions amb el conjunt de validació *val_dl.* Veiem un primer resultat, amb els paràmetres que hem marcat per defecte i corrent 10 èpoques d'entrenament:

```
Epoca: 8, Pèrdua: 0.94, Accuracy: 0.69, Quantitat arxius processats: 6986
[10,     1] perdua: 0.089
[10,   101] perdua: 9.636
[10,   201] perdua: 18.718
Epoca: 9, Pèrdua: 0.93, Accuracy: 0.69, Quantitat arxius processats: 6986
Fi de l`entrenament

Process finished with exit code 0
```

Figura 16 – Resultat d'entrenament base.

L'entrenament ens marca una exactitud(*accuracy)* d'entrenament del 69%, que òbviament no és suficient per generar un model amb un mínim de qualitat.
En la creació del model hem definit a la funció **CalculateMetrics**() una matriu de confusió generada a partir del número de prediccions correctes de cada classe: la representem com un conjunt de 10 tensors, on cada un ens diu la quantitat de prediccions de cada classe (representada com una posició del tensor). En la diagonal veiem, doncs, les prediccions correctes. Amb els resultats de la matriu extraurem la resta de mètriques. A continuació llencem el programa per fer inferència amb els pesos obtinguts en aquest primer entrenament:

```
Accuracy: 0.71, Total items: 1746
Precisio: 0.7078
Exhaustivitat: 0.7040
Puntuació F1: 0.7036
Confusion Matrix:
tensor([[160.,   1.,   1.,   0.,   1.,  19.,   0.,   4.,   4.,   9.],
        [  0.,  49.,   0.,  12.,   4.,   0.,   8.,   5.,   3.,   3.],
        [  9.,   2., 110.,  10.,   6.,  12.,   0.,   3.,  10.,  20.],
        [  5.,   2.,  27., 133.,   4.,   4.,  16.,   0.,  12.,   9.],
        [  2.,   5.,   5.,  10., 160.,   5.,   3.,  18.,   8.,   8.],
        [ 10.,   1.,   7.,   2.,   1., 154.,   0.,   6.,   2.,   2.],
        [  0.,   2.,   0.,  22.,   0.,   0.,  50.,   0.,   0.,   2.],
        [  6.,   2.,   1.,   1.,   8.,   7.,   3., 175.,   1.,   6.],
        [  8.,   2.,   6.,   3.,   1.,   5.,   0.,   1., 146.,   8.],
        [ 14.,   2.,  35.,   1.,   5.,  11.,   0.,   3.,  12., 111.]])

Process finished with exit code 0
```

Figura 17 – Resultat de mètriques d'inferència base.

Les mètriques d'exhaustivitat i precisió no són excessivament dolentes per ser una primera prova: estan totes dues sobre 0.70. Una exactitud (*accuracy*) d'inferència del 71% no obstant, no és idònia; recordem que, en models basats en CNN, amb el *dataset* UrbanSound8K, les mètriques SOTA estan actualment en un 98% per el model **FACE** [ref], molt més complert, però amb la mateixa base de treball. El model **1DCNN**, presentat el 2019 i molt similar al nostre, està a l'entorn del 90%.

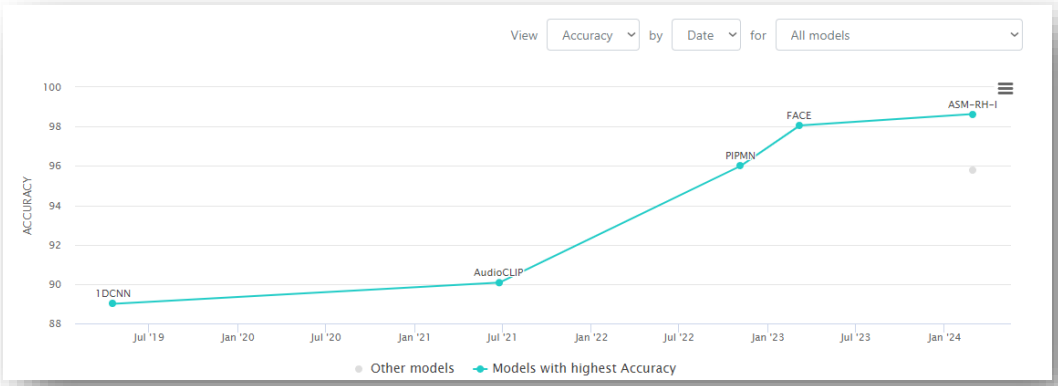


Figura 18 – Resultats SOTA per al dataset UrbanSound8k a maig de 2024..

Una de les primeres reflexions que podem fer és que el nombre d'èpoques és insuficient. En aquest punt és on hem decidit augmentar les èpoques, i després de diverses proves, acabem fent el bucle anteriorment explicat, amb un màxim de 100 èpoques (a les quals sabem que no arribarem mai) i un paràmetre de paciència de 3. Tornem a engegar l'entrenament i tot i que ara ens porta un temps considerable (més de dues hores per a l'UrbanSound, tenint en compte que el conjunt d'entrenament són quasi 7000 mostres d'àudio), els resultats milloren notablement:

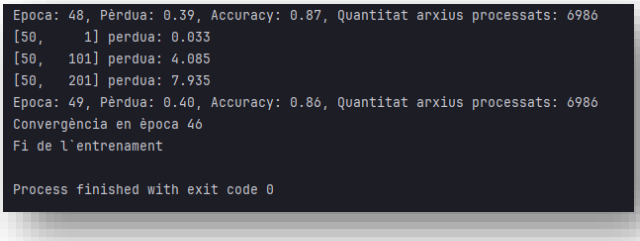


Figura 19 – Resultat d'entrenament amb max_epochs = 100 i augment de mida dels lots.

Tornem a llençar la classe **run** per veure la inferència del model, molt similar a l'anterior, però amb les mètriques lleugerament per sota:

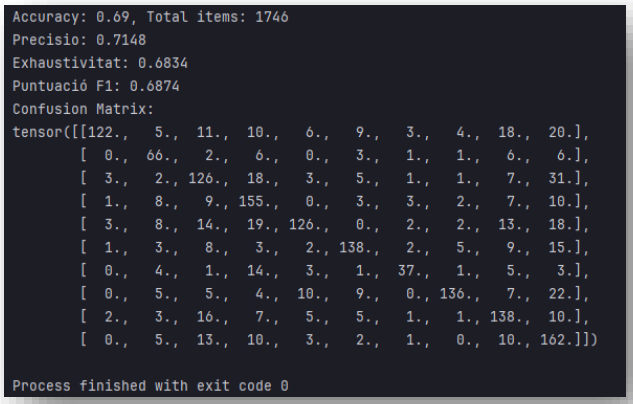


Figura 20 – Resultat de mètriques max_epochs = 100 i augment de mida dels lots.

La precisió ha augmentat lleugerament (per tant el model és més precís en les vegades que encerta la classe), però no millora la resta de xifres. En general no són males xifres, tot i que inferiors a la precisió de l'entrenament, probablement degut a un punt de sobreajustament del model, causat per excés d'entrenament (ja sigui per la quantitat d'èpoques, o per els paràmetres de l'optimitzador o el planificador,...).
Un altre detall que podem apreciar en aquests dos primers experiments (i que es repetirà al llarg de les diverses proves) és que el model no té la mateixa precisió en totes les classes:

CLASSE	ITEMS	CORRECTES	ACCURACY
0 = aire condicionat	208	122	***0.58***
1 = clàxon de cotxe	91	66	***0.72***
2 = nens jugant	197	126	***0.64***
3 = gos bordant	198	155	***0.78***
4 = trepant	205	126	***0.61***
5 = motor de cotxe	186	138	***0.74***
6 = tret de pistola	69	37	***0.53***
7 = martell hidràulic	198	136	***0.68***
8 = sirena	188	138	***0.73***
9 = musica al carrer	206	162	***0.78***

Taula 1. *Resultats de les mètriques d'avaluació de la primera fase d'experiments.*

En concret, per aquest últim exercici, les classes 7 (*martell hidràulic*) i 0 (*màquina d'aire condicionat*) han tingut només un 53% i un 58% respectivament d'exactitud. Això també ens indica les febleses del model, al qual li costa més reconèixer sons més monòtons i repetitius com aquests dos. En canvi, mostra el millor rendiment (un 78% *d'accuracy*) en les classes 3 (*gossos bordant*) i 9 (*música al carrer*), que són les classes més riques tímbricament i de les quals tenim més varietat de fragments.

Refinament

A partir dels resultats anteriors i amb tot l'aprés sobre teoria de processament del so, decidim ajustar alguns paràmetres del nostre model.
El primer que podem provar és a modificar la finestra d'anàlisi; sabem que un nombre més alt ens donarà més informació sobre la quantitat de freqüències presents a l'espectrograma, però desdibuixarà les marques temporals; per contra, reduint la finestra, podem arribar fins i tot a perdre freqüències, però tindrem més ben definides les marques de temps. Per veure clarament la diferència, provarem a modificar la quantitat de mels i la mida de la finestra, de manera proporcional, a 32, 64 i 1024 mels i finestres de 512, 1024 i 2048:

```
Epoca: 8, Pèrdua: 0.93, Accuracy: 0.69, Quantitat arxius processats: 6986
[10,     1] perdua: 0.076
[10,   101] perdua: 9.315
[10,   201] perdua: 18.515
Epoca: 9, Pèrdua: 0.92, Accuracy: 0.69, Quantitat arxius processats: 6986
Fi de l`entrenament

Process finished with exit code 0
```

```
Accuracy: 0.71, Total items: 1746
Precisio: 0.7025
Exhaustivitat: 0.7049
Puntuació F1: 0.7024
Confusion Matrix:
tensor([[154.,   1.,   5.,   1.,   3.,   9.,   0.,   9.,   2.,  10.],
        [  3.,  63.,   1.,   3.,   4.,   1.,   8.,   2.,   4.,   4.],
        [  9.,   2., 135.,  11.,   2.,  17.,   0.,   3.,   7.,  20.],
        [  5.,   2.,  12., 125.,   6.,   8.,  11.,   1.,   4.,   7.],
        [  6.,  12.,   7.,   8., 136.,   0.,   3.,  26.,   2.,   7.],
        [  9.,   0.,   9.,  10.,   1., 155.,   0.,   3.,   4.,   8.],
        [  0.,   6.,   0.,  17.,   3.,   0.,  43.,   0.,   0.,   0.],
        [  2.,   1.,   1.,   2.,  13.,   6.,   4., 154.,   0.,   5.],
        [  6.,   2.,  14.,   8.,   1.,   1.,   1.,   1., 149.,   5.],
        [ 19.,   6.,  35.,   3.,   9.,  13.,   0.,   6.,   6., 124.]])

Process finished with exit code 0
```

Figura 21 – Resultat d'entrenament i mètriques amb 128 mels i finestra de 2048..

```
Epoca: 8, Pèrdua: 0.93, Accuracy: 0.69, Quantitat arxius processats: 6986
[10,     1] perdua: 0.083
[10,   101] perdua: 9.352
[10,   201] perdua: 18.038
Epoca: 9, Pèrdua: 0.90, Accuracy: 0.71, Quantitat arxius processats: 6986
Fi de l`entrenament

Process finished with exit code 0
```

```
Accuracy: 0.72, Total items: 1746
Precisio: 0.7253
Exhaustivitat: 0.7158
Puntuació F1: 0.7154
Confusion Matrix:
tensor([[141.,   0.,   4.,   5.,  12.,  11.,   0.,   3.,   4.,   2.],
        [  1.,  55.,   2.,   7.,   7.,   1.,  10.,   3.,   1.,   1.],
        [ 20.,   0., 129.,  14.,   9.,   8.,   0.,   1.,   5.,  12.],
        [ 10.,   5.,  18., 156.,   7.,   4.,   7.,   0.,   6.,   1.],
        [  3.,   5.,   2.,   4., 167.,   3.,   6.,  11.,   4.,   0.],
        [ 34.,   0.,   3.,   3.,   1., 125.,   0.,  17.,   4.,   3.],
        [  1.,   5.,   0.,   8.,   2.,   1.,  65.,   0.,   0.,   0.],
        [  7.,   0.,   0.,   0.,  16.,   3.,   3., 166.,   0.,   0.],
        [ 18.,   2.,  11.,  11.,   9.,   6.,   0.,   3., 138.,   6.],
        [  6.,   4.,  41.,   3.,   6.,   2.,   0.,   5.,  11., 110.]])

Process finished with exit code 0
```

Figura 22 – Resultat d'entrenament i mètriques amb 32 mels i finestra de 512..

```
Epoca: 7, Pèrdua: 0.91, Accuracy: 0.70, Quantitat arxius processats: 6986
[9,     1] perdua: 0.109
[9,   101] perdua: 8.983
[9,   201] perdua: 17.903
Epoca: 8, Pèrdua: 0.89, Accuracy: 0.70, Quantitat arxius processats: 6986
[10,     1] perdua: 0.085
[10,   101] perdua: 8.568
[10,   201] perdua: 17.182
Epoca: 9, Pèrdua: 0.85, Accuracy: 0.72, Quantitat arxius processats: 6986
Fi de l`entrenament

Process finished with exit code 0
```

```
Accuracy: 0.89, Total items: 1746
Precisio: 0.8959
Exhaustivitat: 0.8925
Puntuació F1: 0.8935
Confusion Matrix:
tensor([[180.,   1.,   4.,   1.,   2.,   2.,   0.,   0.,   2.,   3.],
        [  0.,  82.,   1.,   7.,   2.,   0.,   0.,   1.,   2.,   0.],
        [  1.,   0., 208.,   2.,   1.,   1.,   1.,   0.,   0.,  11.],
        [  0.,   5.,  10., 158.,   4.,   0.,   0.,   0.,   1.,   4.],
        [  1.,   6.,   1.,   9., 185.,   1.,   0.,   0.,   2.,   5.],
        [  5.,   2.,   1.,   2.,   0., 187.,   0.,   2.,   0.,   3.],
        [  0.,   2.,   0.,   0.,   3.,   0.,  61.,   0.,   0.,   0.],
        [  0.,   1.,   0.,   0.,   4.,   5.,   1., 191.,   1.,   1.],
        [  4.,   2.,   1.,   5.,   1.,   3.,   0.,   0., 153.,   3.],
        [  1.,   1.,  29.,   3.,   1.,   0.,   0.,   0.,   6., 154.]])

Process finished with exit code 0
```

Figura 23 – Resultat d'entrenament i mètriques amb 64 mels i finestra de 1024..

Observem que els resultats de l'entrenament no són gaire diferents entre ells (ja que els paràmetres intrínsecs del model no han variat), però si els de la inferència: veiem com l'augment de mels o del tamany de la finestra (s'ha provat també per separat) empitjora els resultats del model, mentre que la seva reducció els millora. Els resultats òptims en aquest cas els trobem en 64 mels i una finestra de 1024; això probablement guarda relació amb el que ja hem explicat en l'apartat de teoria: una finestra menor ens dona una millor imatge dels atacs temporals de les freqüències, mentre que una de major ens dona més precisió en la quantitat de freqüències. En aquest joc, però, els sons són força repetitius i no són gaire diversos tímbricament. En molts casos fins i tot (lladruc de gos, sirena, etc.) el fragment de so és una repetició periòdica de les mateixes freqüències, així que el que millor defineix el so és la posició temporal de les freqüències en l'espectrograma, més que no pas les pròpies freqüències.
Ara volem posar en valor el procés d'augment de dades per millorar la capacitat de generalització del model. Hem fet una prova entrenant el model amb diferents optimitzadors i finestres, amb i sense el procés **SpecAugment**; en molts casos, sense SpecAugment la precisió de l'entrenament millora qualitativament, però la inferència cau en picat:

```
Epoca: 49, Pèrdua: 0.15, Accuracy: 0.96, Quantitat arxius processats: 6986
[51,     1] perdua: 0.015
[51,   101] perdua: 1.427
Epoca: 50, Pèrdua: 0.14, Accuracy: 0.96, Quantitat arxius processats: 6986
Convergència en època 47
Fi de l'entrenament

Process finished with exit code 0
```

```
Accuracy: 0.26, Total items: 1746
Precisio: 0.2944
Exhaustivitat: 0.2741
Puntuació F1: 0.2345
Confusion Matrix:
tensor([[ 30.,   1.,  46.,  18.,  22.,   8.,   8.,   1.,  10.,  54.],
        [  0.,   4.,   8.,  27.,   0.,   2.,  20.,   1.,   1.,  35.],
        [  4.,   0.,  82.,  53.,  21.,   2.,   6.,   1.,  14.,  19.],
        [  2.,   3.,  29., 124.,   8.,   0.,  27.,   0.,   2.,   5.],
        [ 14.,   8.,  16.,  32.,  60.,   3.,  13.,   3.,   3.,  37.],
        [ 21.,   1.,  56.,  31.,  16.,  30.,  10.,   1.,   7.,  33.],
        [  0.,   1.,   1.,  16.,   1.,   0.,  40.,   0.,   1.,   4.],
        [ 20.,   8.,  24.,  14.,  24.,  23.,   6.,  18.,   2.,  48.],
        [  7.,   6.,  78.,  36.,  13.,   1.,   8.,   0.,   9.,  43.],
        [  3.,   3.,  58.,  50.,   6.,   2.,   1.,   3.,  16.,  59.]])

Process finished with exit code 0
```

Figura 24 – Resultat d'entrenament i mètriques amb optimitzador Adam, 32 mels i finestra de 512 ***sense*** *SpecAugment..*

Un 96% de precisió (xifra poc realista, d'altra banda) ens indica un clar sobreajustament del model a les dades d'entrenament, que dificulta després fer les prediccions de manera correcta; els mateixos paràmetres, amb el mètode SpecAugment, ens redueixen la precisió de l'entrenament a un 89%, però ens llencen una exactitud del 71% en les prediccions, remarcant la importància de la tècnica de cara a millorar la generalització:

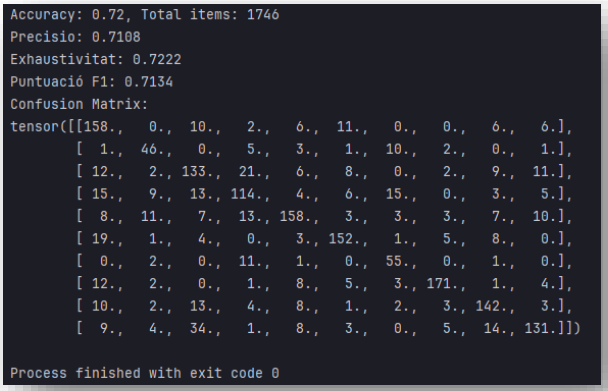
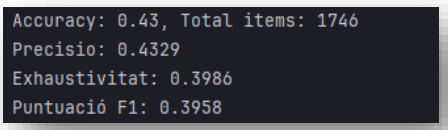


Figura 25 – Resultat d'entrenament i mètriques amb optimitzador Adam, 32 mels i finestra de 512 ***amb*** *SpecAugment..*

Els optimitzadors

Per últim, un altre dels paràmetres que s'han analitzat és l'optimitzador. El bucle d'entrenament original estava definit amb un optimitzador **Adam**, força popular en aquest tipus de solucions d'àudio (de fet és l'utilitzat pel model CLAP i també pel **Wav2Clip** [23] , un dels primers models en implementar la solució *Contrastive-Language)*, però he decidit provar també altres solucions; en concret s'ha provat a arrancar el bucle amb l'optimitzador **AdamW**, el **RMSProp**, reduint el nombre d'èpoques a 20 per no allargar excessivament el temps de les proves i poder veure com variaven els resultats; finalment, també s'ha optat per un **SGD** (*Stochastic Gradient Descent*), utilitzat en el model desenvolupat pels autors de la investigació **Transformers for Audio Classification [24]**, d'on s'han tret gran part de les idees de processament i us dels espectrogrames d'aquest treball.



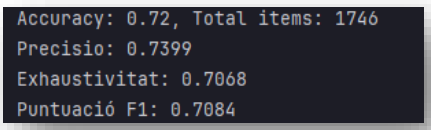


Figura 26 – Resultat de les mètriques amb optimitzador ***RMSprop*** *a la dreta i* ***SGD*** *a l'esquerra. 64 mels i finestra de 1024.*

Veient els resultats de les mètriques i l'entrenament, ens adonem de la gran diferència que suposa utilitzar un o altre optimitzador; però veure-ho més gràficament, hem fet diverses proves d'entrenament amb 20 èpoques (per agilitzar el procés) i veure més clarament com evoluciona l'aprenentatge segons cadascun:

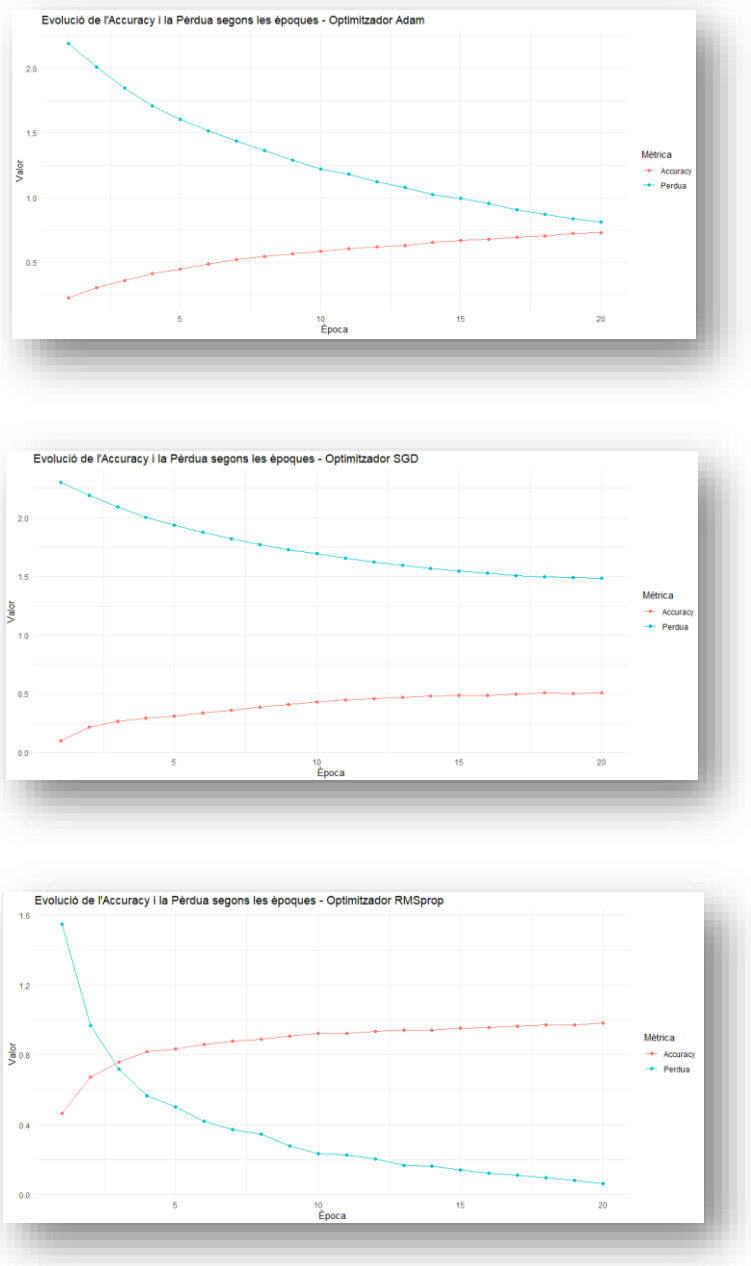


Figura 27 – Evolució de l'aprenentatge segons optimitzadors Adam, SGD i RMSprop..

Veiem com amb *l'Stochastic Gradient* la precisió augmenta més lentament i els resultats finals de la inferència són lleugerament inferiors que els del *RMSprop*, i en tots dos casos, inferiors al millor cas del Adam. Sorprèn molt, però, els resultats del *RMSprop*, que augmenta molt ràpidament la *accuracy*, però sobretot, disminueix els resultats de la funció de pèrdua en molt poques èpoques. És un pas interessant, que ens permetria entrenar els models més ràpidament, però ja hem vist que els resultats són inferiors als de l'Adam, que tot i anar més lent, acaba sent més precís. Això probablement sigui a causa de que l'RMS s'ajusta molt ràpidament a les dades d'entrenament ja que moltes són relativament semblants. Per intentar compensar això i trobar l'equilibri perfecte, reduïm la tasa d'aprenentatge del *RMSprop* a 0,001, perquè reduexi els passos d'actualització i generalitzar millor. Llencem l'entrenament i veiem la gràfica resultant:

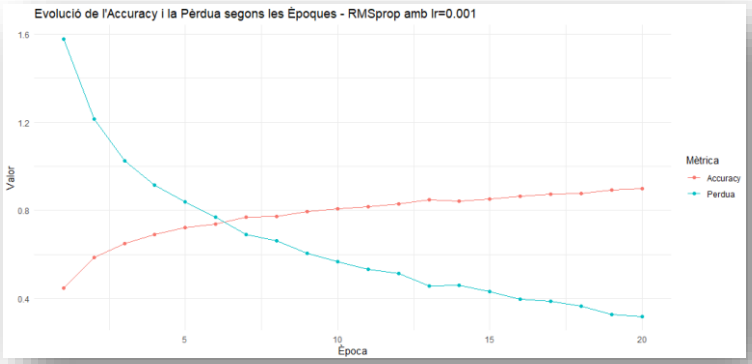


Figura 28 – Evolució de l'aprenentatge optmiitzador RMSprop amb tasa d'aprenentatge 0,001.

En aquest cas, la corba és més suau, però l'aprenentatge segueix sent molt més ràpid que en els altres dos optimitzadors.
En aquest punt, podem deduir que la velocitat del RMSprop és deguda a la seva pròpia naturalesa: aquest optimitzador, acrònim de *Root Mean Square Propagation*, ajuda a actualitzar els pesos de manera dinàmica, calculant en cada pas la mitjana quadràtica dels gradients i adaptant així la tasa d'aprenentatge en cada iteració. Podem teoritzar també que els gradients no canvien excessivament (almenys per aquest tipus de dades, la naturalesa del so pot canviar si canviem el conjunt d'entrenament, lògicament) i per tant això explicaria l'eficiència del RMSprop en aquest cas concret. L' SGD, en canvi, es veu penalitzat, ja que té una tasa d'aprenenatge fixa en totes les iteracions. L'Adam fa servir una combinació de les dues tècniques. Val a dir que a cada modificació dels optimitzadors, a més de la tasa d'aprenentage (*learning rate*), s'ha ajustat també el planificador utilitzant els tipus *OneCycle* o *Standard Reduce on Plateau*. El OneCycleLR és el que ha donat millors resultats en tots els casos

Avaluació final

Després d'unes quantes proves més, veiem que podem combinar una finestra més baixa (512) i 32 mels amb aquest optimitzador; entrenem el model i veiem els resultats:

```
Accuracy: 0.91, Total items: 1746
Precisio: 0.9017
Exhaustivitat: 0.9122
Puntuació F1: 0.9058
Confusion Matrix:
tensor([[194.,   0.,   0.,   1.,   1.,   2.,   0.,   1.,   2.,   1.],
        [  0.,  81.,   0.,   1.,   3.,   0.,   3.,   0.,   2.,   2.],
        [  1.,   0., 164.,   5.,   2.,   4.,   0.,   2.,   4.,  20.],
        [  1.,  10.,   6., 188.,   1.,   1.,   3.,   0.,   4.,   4.],
        [  0.,   1.,   0.,   1., 180.,   0.,   6.,   5.,   2.,   2.],
        [  0.,   0.,   1.,   0.,   0., 186.,   1.,   0.,   0.,   2.],
        [  0.,   1.,   0.,   2.,   1.,   0.,  71.,   0.,   0.,   0.],
        [  0.,   0.,   0.,   0.,   1.,   1.,   3., 175.,   0.,   0.],
        [  1.,   0.,   1.,   3.,   1.,   3.,   2.,   0., 193.,   3.],
        [  2.,   1.,  11.,   1.,   4.,   2.,   0.,   1.,   3., 158.]])

Process finished with exit code 0
```

Figura 29 – Resultat mètriques del model amb millor optimització.

Un 91 % d'accuracy, molt a prop dels resultats actuals SOTA, i amb bones xifers també en exhaustivitat i precisió; això ens indica que el model és molt més precís a l'hora de classificar correctament les classes. Així mateix, podem comprovar que han desaparegut les diferències de predicció entre classes, i ara tenim una distribució quasi normal de l'exactitud en totes elles gràcies a les adaptacions realitzades al model.

A mode de resum, podem analitzar millor l'evolució dels experiments a la següent taula:

EXPERIMENT (arxiu *weights*)	Optimitz.	LR	N°MELS	Finestra	Accuracy
model base (10 epochs)	Adam	0,001	64	1024	***71%***
audio_classifier_weights_64m_1024w_Adam	Adam	0,001	64	1024	***69%***
audio_classifier_weights_128m_2048w_Adam_SpecAug	Adam	0,001	128	2048	***71%***
audio_classifier_weights_32m_512w_Adam	Adam	0,001	32	512	***26%***
audio_classifier_weights_32m_512w_Adam_SpecAug	Adam	0,001	32	512	***72%***
audio_classifier_weights_64m_1024w_AdamOK	Adam	0,001	64	1024	***89%***
audio_classifier_weights_64m_1024w_SGD	SGD	0,01	64	1024	***43%***
audio_classifier_weights_64m_1024w_RMSprop	RMSprop	0,01	64	1024	***72%***
audio_classifier_weights_32m_512w_RMSprop_0001	RMSprop	0,001	128	2048	***91%***

Taula 2. *Resultats de les mètriques d'avaluació de la segona fase d'experiments (fine-tuning).*

3.5. Altres models

Per acabar de completar el treball, hem realitzat una sèrie de proves amb conjunts de dades i models diferents. Tot i que un dels objectius del treball era precisament desenvolupar aquesta segona part, no ha estat possible degut al temps disponible i a la complexitat de les tasques. Tot i això, no volem deixar passar la oportunitat de fer menció de les diferents possibilitats que hi ha en aquest camp.

D'entrada, s'ha analitzat el model que va inspirar el treball en primer lloc, el model CLAP de Microsoft. Aquest és un projecte molt més complert que el presentat aquí però que parteix de la mateixa base teòrica. L'estructura del projecte li permet agafar d'entrada diversos conjunts de dades, alguns dels quals (com *l'ESC50* o l'*AudioSet*) ja venen predefinits i entrenats amb els corresponents pesos, que es poden descarregar d'un repositori online directament des del propi projecte. El CLAP implementa un sistema de sis blocs convolucionals, que reben 1 sol canal d'entrada i acaben generant-ne 2048, i incorpora a més un mòdul que implementa el concepte de atenció (*attention*) **[25]**, una tècnica que permet al model separar la seqüència d'entrada de dades en diverses parts,

per focalitzar-se en aquelles parts que siguin més rellevants segons la tasca a dur a terme.

El codificador d'àudio, tot i que més sofisticat, és en essència similar al vist aquí, realitzant gran part de les tasques de preprocessament dels fragments i extracció de característiques; el codificador de text, però, és el que realment marca la diferència. En el moment de la seva publicació, el 2020, el CLAP utilitzava un codificador basat en BERT per processar les descripcions textuals. El BERT és un popular model de llenguatge a gran escala (LLM) entrenat amb més de 110 milions de paràmetres, que permet crear una matriu d'incrustacions *(embeddings)* molt complerta amb la qual el model aconsegueix crear *prompts* de text molt similars al llenguatge natural a l'hora de fer les prediccions. En les últimes actualitzacions, però, el CLAP ja incorpora la possibilitat de triar la versió GPT2 de OpenAI com a codificador de text, oferint resultats fins i tot millors que els del BERT original.

Per veure-ho, provem a llençar un arxiu de l'UrbanSound, que a les metadades es defineix com a *engine idling* (motor en marxa), a pesar de que s'hi senten més sons (és un cotxe amb el motor en marxa en el qual està sonant una alarma de porta oberta que de cop s'atura). La resposta del CLAP, en canvi, és molt més complerta i capaç de distingir i verbalitzar l'escena complerta, tot i que no és precisa del tot:

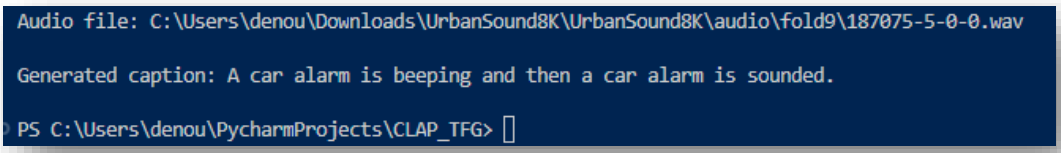
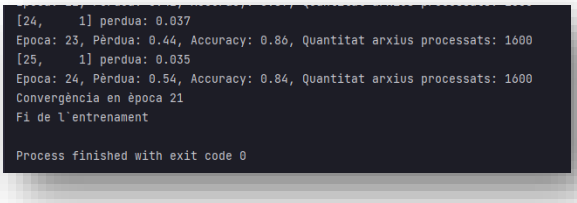


Figura 30 – Prompt de sortida del model CLAP, amb codificador GPT2, per un dels sons del conjunt UrbanSound8K.

El projecte disponible al respositori oficial de *GitHub* ja conté uns quants exemples amb alguns conjunts de dades preassignats. Un d'ells és el ESC50 [13], que es va valorar com a possible conjunt per aquest treball i finalment es va descartar. Per fer la comparació hem llençat el mateix conjunt en els dos models, el CLAP i el nostre.

Figura 31 – Resultat de l'accuracy del model CLAP amb el conjunt ESC-50 i el model amb pesos preentrenats.

Es pot veure que tenim un nivell de quasi el 94% de precisió amb el CLAP, gens malament, tenint en compte que és un conjunt amb moltes més classes que l'Urban Sound (50 classes enfront a 10). Per fer la prova hem fet una adaptació de les classes *dataLoader* i *TrainingData* del nostre model per que agafin la estructura de directoris del ESC50 i l'identificador de la classe del fitxer de metadades del conjunt. D'aquesta manera podem llençar el mateix conjunt amb el nostre model; utilitzarem els paràmetres que ens han donat millors resultats amb el UrbanSound: *32 mels, finestra de 512 SamplingRate a 44100, i SpecAgment; canviarem, però, l'optimitzador Adam original per el RMSprop, amb una tasa d'aprenentatge del 0,001*. Aquest és el resultat:



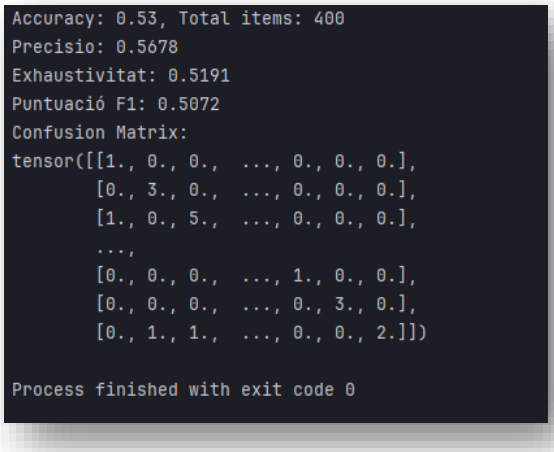


Figura 32 – Resultat d'entrenament i mètriques amb optimitzador RMSprop, 32 mels i finestra de 512 ***per al conjunt ESC-50****..*

Veiem una precisió molt alta en l'entrenament, però que es rebaixa substancialment quan fem inferència. Per intentar millorar aquests resultats farem alguns petits ajustaments:

- SI fem un anàlisi dels fragments d'audio veiem que a l'ESC50 tots els fitxers *duren exactament 5 segons*; augmentem doncs aquesta xifra en el procés de preparació de les dades
- Tenim moltes menys dades, 2000 enfront de 8000; per tant, podem fer els lots més petits sense perjudicar en excés el temps d'entrenament. Aprofitarem també

per *reduir encara més la tasa d'aprenentatge de l'optimitzador*, que amb els lots més petits ens permetrà millorar la dinàmica de l'entrenament.

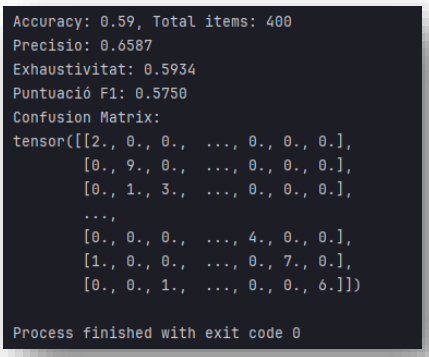


Figura 33 – Resultat de les mètriques amb optimitzador RMSprop, 64 mels i finestra de 1024 per al conjunt ESC-50 amb modificacions varies..

Hem augmentat lleugerament els nombres, però no en excés. Pensem que els sons d'aquest conjunt són, en general, més rics tímbricament que el de l'UrbanSound; hi ha 40 classes més i les classes són molt més variades: sons d'animals, de diversos vehicles, sons barrejats, etc. Tal com hem vist abans, una finestra petita ens podria penalitzar, així que decidim fer unes quantes proves més i finalment optem per *augmentar el nombre de mels a 128 i la finestra a 2048*, mantenint la resta de paràmetres iguals a l'hora de fer l'espectrograma. El resultat:

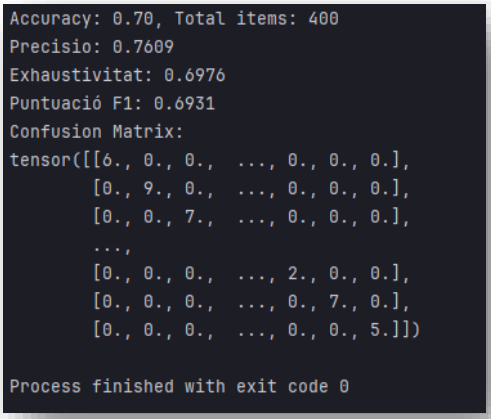


Figura 34 – Resultat de les mètriques amb optimitzador RMSprop, 128 mels i finestra de 2048 per al conjunt ESC-50 amb modificacions varies..

Així doncs, sembla que això confirma la nostra teoria, a pesar de que seguim molt lluny de les mètriques del CLAP; aquí ens cal tenir en compte que el CLAP està preentrenat amb milers de dades de tot tipus, mentre que nosaltres només tenim les pròpies dades del conjunt.

Resumim els resultats dels experiments amb l'ESC-50 en la següent taula:

EXPERIMENT (arxiu *weights*)	Optimitz.	LR	N°MELS	Finestra	Accuracy
audio_classifier_weights_ESC50_32m_512w_4s	RMSprop	0,001	32	512	***53%***
audio_classifier_weights_ESC50	RMSprop	0,001	64	1024	***59%***
audio_classifier_weights_ESC50_64m_1024w	Adam	0.001	64	1024	***62%***
audio_classifier_weights_ESC50_128m_2048w	Adam	0.001	128	2048	***68%***
audio_classifier_weights_ESC50_128m_2048w_SpecAugmentOK	RMSprop	0.001	128	2048	***70%***

Taula 3. *Resultats de les mètriques d'avaluació de la tercera fase d'experiments (ESC-50).*

4. Conclusions

En aquest treball hem analitzat en detall l'estructura general d'un model d'aprenentatge profund dissenyat específicament per tasques de reconeixement d'àudio, en concret de sons de l'entorn urbà. Per fer-ho, hem vist les vicissituds de l'àudio digital i la pròpia naturalesa matemàtica d'aquest per poder comprendre millor com funcionen aquest tipus de models. Hem fet un repàs a algunes de les principals tècniques utilitzades en el reconeixement de sons de l'entorn i hem comprovat que algunes de les més eficaces són les que segueixen el model de llenguatge contrastiu. Hem pogut així mateix crear un projecte sencer de model de classificació, amb la fase d'entrenament i d'inferència, i hem vist la importància de parametritzar correctament el model per obtenir uns bons resultats de predicció.

4.1. Conclusions

A pesar de que hi ha una gran quantitats de solucions disponibles actualment per la classificació d'àudio, les arquitectures basades en sistemes d'aprenentatge profund, com ara les xarxes neurals han demostrat ser de les més eficaces en aquest camp. S'ha evolucionat molt els últims anys, especialment pel que fa al reconeixement de veu, i les CNN han jugat un paper molt important en això; la seva capacitat per processar imatges amb gran precisió, permet treballar directament, tal com hem vist, amb els espectrogrames derivats de fragments sonors concrets, extraient les seves característiques principals per posteriorment classificar-los de manera correcta.
Hem trobat diversos problemes al llarg del treball, però potser el més important sigui el *sobreajustament dels models.* El fet de contenir fragments de so molt similars entre ells dificulta al model aprendre més enllà del que veu en l'entrenament (generalització), especialment en el camp dels sons d'entorn com hem treballat nosaltres. Aquest tipus de models han demostrat gran eficàcia en el reconeixement de veu justament per això: donats uns fonemes o uns vocables d'un idioma determinat, l'entrenament per paraules o per síl·labes resulta altament precís **[26]**, perquè la generalització es limita a un conjunt finit de sons. En el cas que hem vist nosaltres, però, les dades tenen molt bones xifres de precisió quan ens limitem als sons del propi conjunt, però molt inferiors quan intentem passar sons diferents, amb diferents tractaments, captats amb altres tècniques (micròfons, sons ambient, sorolls de fons, etc.). Com en altres casos amb la IA, una de les claus és la quantitat de dades d'entrenament; com més dades es puguin utilitzar per entrenar el model, més capacitat de generalització tindrà, però no és la única: també és important definir bé les etiquetes, especialment en el cas del llenguatge contrastiu, ja que els pesos dels textos que descriuen els sons permeten jugar amb la variabilitat dels mateixos, facilitant la interpretació d'aquests per part del model en el moment de fer la inferència.

4.2. Línies de futur

Tot i els grans avenços en aquest camp, crec que encara queda feina per fer. Probablement seria interesant continuar desenvolupant solucions com el CLAP o similars, que permeten la incorporació del sistema d'incrustacions *(embeddings)* de pesos de text dels actuals LLMs. Molts dels treballs consultats són anteriors a 2022, l'any de la gran eclosió de ChatGPT i que va ser l'avançada de la gran quantitat de LLM disponibles avui dia, molts dels quals, a més, són de codi obert i permeten una gran personalització, ja sigui parametritzant el propi codi del model o a través de Generació Augmentada per Recuperació (*Retrieval-augmented Generation*, RAG). Totes aquestes tecnologies es podrien incorporar a tipus d'arquitectures com les que hem vist de manera que el resultat final fos una descripció exacta del que el model està escoltant en cada moment, permetent així interactuar de manera natural amb la IA aprofitant tots els recursos que ens ofereix el llenguatge o l'entorn.

4.3. A títol personal

L'elaboració d'aquest treball ha suposat un gran repte i alhora un aprenentatge. He pogut comprovar de primera ma la importància de la documentació, la investigació i l'aprofundiment en el coneixement per entendre al màxim nivell tots els vessants de la IA i les tècniques d'aprenentatge profund. Així mateix, he après que és clau la organització i planificació del temps. El punt més crític, però, crec que ha estat el dels recursos de computació; tot i que he pogut fer els experiments de manera més o menys solvent amb el meu equip, alguns altres experiments quedaven fora del meu abast, ja fos per la profunditat de les xarxes d'entrenament com per la quantitat de dades necessàries.

Espero que aquest treball serveixi per despertar l'interès de tots aquells que el llegeixin, i, qui sap, potser per animar-los a emprendre camí en l'apassionant món del so i la IA.

4.4. Seguiment de la planificació

S'han presentat en cada PAC els corresponents informes de seguiment de planificació, així com l'evolució del diagrama de Gantt de control del treball.

A data de la finalització d'aquesta memòria, s'han assolit totes les tasques previstes excepte la tasca de classificació d'escenes sonores.

5. Glossari

Amplitud: Mida màxima d'una ona mesurada des de la seva posició de repòs fins al seu pic màxim.

API (Application Programming Interface): Conjunt de regles que permeten que diferents programes de *software* es comuniquin entre ells.

Aprenentatge Profund (Deep Learning): Disciplina del *machine learning* que utilitza xarxes neurals artificials profundes per modelar i comprendre dades complexes.

CNN (Convolutional Neural Network): Tipus de xarxa neuronal especialment eficaç en tasques de processament d'imatges i reconeixement de patrons visuals.

Codi Obert (Opensource): Programari amb codi font accessible al públic, permetent la seva modificació i distribució per qualsevol persona.

Conjunt de dades (Dataset): Col·lecció estructurada de dades utilitzada per entrenar i avaluar models de *machine learning*.

Decibel: Unitat de mesura de la intensitat del so, expressada en una escala logarítmica.

DFT (Discrete Fourier Transform): Algorisme que transforma una seqüència de valors en una suma de components sinusoidals de diferents freqüències.

Espectre: Representació de les components de freqüència d'un senyal.

Espectrograma: Representació visual de l'espectre de freqüències d'un senyal en funció del temps.

Fase de la ona (phase): Posició relativa d'un punt dins d'un cicle d'una ona, mesurada en graus o radians.

FFT (Fast Fourier Transform): Algorisme eficient per calcular la Transformada de Fourier Discreta (DFT).

Finestra (Analysis Window): Segment de temps en el qual s'analitza un senyal durant el processament d'àudio.

Freqüència: Nombre de cicles complets d'una ona que passen per un punt fix en un segon, habitualment mesurat en Hertz (Hz).

IA (Intel·ligència Artificial): Camp de la informàtica que se centra en la creació de sistemes capaços de realitzar tasques complexes que requereixen intel·ligència humana.

IFT (Inverse Fourier Transform): Operació que serveix per reconvertir un espectre de freqüències a la seva forma original en el domini temporal.

Pesos (Weights): Paràmetres de càlcul en els nodes d'una xarxa neuronal que s'ajusten durant l'entrenament per minimitzar l'error de predicció.

Pressió sonora: Variació de la pressió de l'aire causada per una ona sonora, mesurada de manera lineal en pascals (Pa).

So: Vibració que es propaga com una ona mecànica a través d'un medi com l'aire o l'aigua, i és percebut per l'oïda humana.

STFT (Short-Time Fourier Transform): Variació de la Transformada de Fourier aplicada a segments temporals curts del senyal.

Tasa de mostreig (Sampling Rate): Nombre de mostres per segon preses d'un senyal continu per crear un senyal discret, mesurat en Hertz (Hz).

Xarxa Neuronal: Sistema de càlcul compost per unitats interconnectades (neurones) que processen informació de manera similar al cervell humà.

Internet de les Coses (IoT): Xarxa d'objectes físics equipats amb sensors, programari i altres tecnologies que permeten intercanviar dades amb altres dispositius i sistemes a través d'internet.

Sobreajustament (Overfitting): Problema en l'aprenentatge automàtic on un model aprèn massa bé els detalls i el soroll de les dades d'entrenament, cosa que fa que el seu rendiment disminueixi a l'hora de generalitzar (tractar dades noves i no vistes).

Generació Augmentada per Recuperació (RAG): Tècnica de IA que combina la recuperació de documents rellevants d'una base de dades amb la generació de text per proporcionar respostes en base a aquestes.

Llenguatges de Gran Escala (LLM): Models d'IA entrenat en grans quantitats de text capaços de comprendre i generar llenguatge natural així com de realitzar diverses tasques (traducció de textos, resposta a preguntes, resums, etc.)

6. Bibliografia

[1] Sora. OpenAI. [online] [Consultat l'Abril de 2024] Disponible a:
https://openai.com/index/sora

[2] Midjourney. [online] [Consultat l'Abril de 2024] Disponible a:
https://www.midjourney.com/home

[3] CONTRIBUTORS TO WIKIMEDIA PROJECTS. Stable Diffusion - Wikipedia. Wikipedia, the free encyclopedia [online]. 31 d'agost de 2022 [consultat el 20 d'abril de 2024]. Disponible a:
https://en.wikipedia.org/wiki/Stable_Diffusion

[4] Introducing a foundational multimodal model for speech translation. *AI at Meta* [online]. [sense data] [consultat el 18 d' abril de 2024]. Disponible a:
https://ai.meta.com/blog/seamless-m4t/

[5] GitHub - facebookresearch/demucs: Code for the paper Hybrid Spectrogram and Waveform Source Separation. *GitHub* [online]. [sense data] [consultat el 4 de abril de 2024]. Disponible a:
https://github.com/facebookresearch/demucs

[6] MALIK, Shahid et al. An Acoustic 3D Positioning System for Robots Operating Underground. *IEEE Sensors Letters* [online]. 2022, 1–4 [consultat el 25 d'abril de 2024]. ISSN 2475-1472. Disponible a:
https://ieeexplore.ieee.org/document/9893367

[7]. What is CRISP DM? . [online] [Consultat el Maig de 2024] Disponible a: https://www.datascience-pm.com/crisp-dm-2/

[8] CONTRIBUTORS TO WIKIMEDIA PROJECTS. Fourier analysis - Wikipedia. *Wikipedia, the free encyclopedia* [online]. 20 de desembre de 2001 [consultat a 4 d'abril de 2024]. Disponible a:
https://en.wikipedia.org/wiki/Fourier_analysis

[9] CONTRIBUTORS TO WIKIMEDIA PROJECTS. Absolute threshold of hearing - Wikipedia. *Wikipedia, the free encyclopedia* [online]. 16 d'agost de 2003 [consultado el 20 d'abril de 2024]. Disponible a:
https://en.wikipedia.org/wiki/Absolute_threshold_of_hearing

[10] CONTRIBUTORS TO WIKIMEDIA PROJECTS. Window function - Wikipedia. *Wikipedia, the free encyclopedia* [online]. 11 de juny de 2003 [consultat el 26 d'abril de 2024]. Disponible a:
https://en.wikipedia.org/wiki/Window_function

[11] CONTRIBUTORS TO WIKIMEDIA PROJECTS. Mel scale - Wikipedia. *Wikipedia, the free encyclopedia* [online]. 20 de desembre de 2001 [consultat el 25 d'abril de 2024]. Disponible a:
https://en.wikipedia.org/wiki/Mel_scale

[12] LIU Haohe, LIU Xubo, KONG Qiuqiang, WANG Wenwu, PLUMBLEY Mark D.. Learning Temporal Resolution in Spectrogram for Audio Classification [online]. 4 d'octubre de 2022.[Consultat el 28 d'abril de 2024]. Disponible a:
https://arxiv.org/abs/2210.01719

[13] ESC-50 Dataset. [online] [Consultat l'Abril de 2024] Disponible a:

https://paperswithcode.com/sota/audio-classification-on-esc-50

[14] URBANSOUND8K DATASET. [online] [Consultat l'Abril de 2024] Disponible a:
https://urbansounddataset.weebly.com/urbansound8k.html

[15] FSD50K Companion site. [online] [Consultat l'Abril de 2024] Disponible a:
https://annotator.freesound.org/fsd/release/FSD50K/

[16] Domestic Environment Sound Event Detection Dataset. [online] [Consultat el 4 de Maig de 2024] Disponible a:
https://project.inria.fr/desed/

[17] GitHub - The Microsoft Scalable Noisy Speech Dataset (MS-SNSD). *GitHub* [online]. [2019] [consultat el 28 de Maig de 2024]. Disponible a:
https://github.com/microsoft/MS-SNSD

[18] ELIZALDE Benjamin, DESHMUKH Soham, AL ISMAIL Mahmoud, WANG Huaming. Clap : learning audio concepts from natural language supervision. A: *Microsoft* [online]. 2022.[Consultat el juny de 2024]. Disponible a:
https://arxiv.org/pdf/2206.04769.pdf

[19] HERSHEY Shawn, CHAUDHURI Sourish, P. W. ELLIS Daniel, GEMMEKE Jort F. CNN Architectures for Large-Scale Audio Classification. [online] 29 de setembre de 2016 [consultat el 21 de maig de 2024]. Disponible a:
https://arxiv.org/abs/1609.09430

[20] SAINATH Tara N., MOHAMED Abdel-rahman; KINGSBURY Brian; RAMABHADRAN Bhuvana. Deep convolutional neural networks for LVCSR [online] 26 de maig de 2013 [consultat el 21 de maig de 2024]. Disponible a:
https://ieeexplore.ieee.org/document/6639347

[21] S.PARK Daniel, CHAN William. SpecAugment: A New Data Augmentation Method for Automatic Speech Recognition [online] 22 d'abril de 2019 [consultat el maig 2024]. Disponible a:
https://research.google/blog/specaugment-a-new-data-augmentation-method-for-automatic-speech-recognition/

[22] RADKOFF Evan. Loss Functions in Audio ML[online] 6 de setembre de 2021. [consultat el juny de 2024]. Disponible a:
https://www.soundsandwords.io/audio-loss-functions/

[23] WU Ho-Hsiang, SEETHARAMAN Prem, KUMAR Kundan , BELLO Juan Pablo. WAV2CLIP: LEARNING ROBUST AUDIO REPRESENTATIONS FROM CLIP. [ICASSP Conference 2022] 26 de maig de 2022. [consultat el 23 de Març de 2024]. Disponible a:
https://doi.org/10.48550/arXiv.2110.11499

[24] YIXIAO Zhang, LI Baihua, FANG Hui, MENG Qinggang. Spectrogram transformers for audio classification. [online] 2022. [consultat l'Abril de 2024]. Disponible a:
https://doi.org/10.1109/IST55454.2022.9827729

[25] VASWANI Ashish, SHAZEER Noam, PARMAR Niki, USZKOREIT Jakob, JONES Llion, N. GOMEZ Aidan, KAISER Łukasz, POLOSUKHIN Illia. Attention is all you need. [In Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17)] 2017. [consultat el 10 de Juny de 2024]. Disponible a:
https://dl.acm.org/doi/10.5555/3295222.3295349

[26] GUPTA Rohan, KUMAR Rohan. Syllable Based Deep Learning for Multi-Dialect Speech Recognition. [online] 2019. [consultat el 12 de Juny de 2024]. Disponible a:
https://www.ijedr.org/papers/IJEDR1904106.pdf

Annexos

Consultes, programari i creació del model:

1. Web de Papers With code, amb articles i models de diversos tipus: Audio Classification | Papers With Code
2. Curs online de Hugging Face, amb varietat de models preentrenats i bases per construir el teu propi model: Audio classification (huggingface.co)
3. Curs de Coursera sobre bàsics de tractament i processament d'audio digital, per Xavier Serra: https://www.coursera.org/lecture/audio-signal-processing/teaser-53YTy
4. SMS- Tools, paquet desenvolupat per la UPF específicament per treballar amb senyals d'audio: https://www.upf.edu/web/mtg/sms-tools
5. Web del Music Technology Group de la UPF, especialitzat en processament d'audio digital i solucions de IA per classificació i reconeixement de sons: https://www.upf.edu/web/mtg
6. Plataforma de creació i manteniment de conjunts de dades d'audio Freesounds, creadors dels conjunts de dades ESC50 i UrbanSound8K: https://freesound.org
7. Repositoris GitHub amb diversos models de proves:
    a. https://github.com/microsoft/Semi-supervised-learning
    b. https://github.com/YuanGongND/ast
    c. https://github.com/ketanhdoshi/ml
    d. https://github.com/YuanGongND/ssast
    e. https://github.com/cwx-worst-one/EAT
    f. https://github.com/DCASE-REPO