UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola Superior d'Enginyeries Industrial, Aeroespacial i Audiovisual de Terrassa

UNIVERSITAT POLITÈCNICA DE CATALUNYA

GRAU EN ENGINYERIA DE SISTEMES AUDIOVISUALS

ESCOLA SUPERIOR D'ENGINYERIES INDUSTRIAL, AEROESPACIAL I AUDIOVISUAL DE TERRASSA

Classificació d'àudio amb la base de dades Freesound/Audioset

TREBALL DE FI DE GRAU

Autor
Laura Gotarra Sànchez

Director
Ignasi Esquerra Llucià

Juny del 2019

Agraïments

En primer lloc, vull agrair profundament al meu tutor de Treball de Fi de Grau, Ignasi Esquerra, per la seva predisposició, el seu acompanyament i per tota la paciència.

També agraeixo que em facilités l'accés al servidor que m'ha estat gairebé indispensable en gran part del treball, així com al departament de Teoria del Senyal i Comunicacions.

Per últim, gràcies als familiars i amics que m'han donat suport al llarg de la realització d'aquest estudi.

Resum

Avui dia existeix una quantitat enorme d'informació sonora que rebem i emmagatzemem de tot allò que ens envolta. Gestionar-la cada cop és més complicat i, d'aquesta necessitat, han començat a sorgir sistemes per al processat automàtic de grans bases de dades d'àudio per extreure'n el més rellevant.

Tot i això, encara queda molta recerca pendent per poder obtenir sistemes veritablement fiables per a usar de manera habitual en el nostre entorn.

Ens centrarem més concretament en els sistemes d'etiquetatge d'àudio d'ús general, tot usant una base de dades elaborada a partir de mostres de Freesound i les etiquetes proposades pels investigadors d'Audioset (Google Inc.).

Com a referèrencia s'usarà el sistema proposat pels investigadors Frederic Font Corbera (Universitat Pompeu Fabra), Eduardo Fonoseca (Universitat Pompeu Fabra), Manoj Plakal (Google, Inc.) i Daniel P. W. Ellis (Google, Inc.) en el repte DCASE Challenge 2018 (Tasca 2).

La intenció no és participar en el repte ni desenvolupar una proposta millorada del sistema, sinó entendre el seu funcionament i posar-lo en marxa per avaluar-lo. En certa manera, aquest treball ha de servir de guia introductòria per aquells que vulguin endinsar-se en el món dels sistemes automatitzats amb l'ús de xarxes neuronals convolucionals a través del llenguatge Python.

Per tant, més enllà de trobar-hi un anàlisi detallat del processament d'àudio també s'hi troben les guies per poder usar aquest mateix sistema amb altres bases de dades (a condició que compleixin els mateixos requisits que la base de dades de Freesound utilitzada).

Tanmateix, a partir de l'experiència i els coneixements adquirits al llarg del Grau en Enginyeria de Sistemes Audiovisuals, cursat a l'Escola Superior d'Enginyeries Industrial, Aeroespacial i Audiovisual de Terrassa, s'oferiran algunes propostes de millores tècniques per tal de continuar aquest projecte més endavant.

Paraules clau: Audioset, Freesound, DCASE, etiquetatge d'àudio, Python, xarxes convolucionals.

Abstract

Nowadays, we receive and store an enormous amount of sound information about our environment. Data management complexity is rising every day due to its size and, in order to make this task simpler, systems for automatic processing of large audio databases have started to emerge to extract the most relevant one.

Nevertheless, there is still a lot of research pending to get truly reliable systems. In this project, we will focus more specifically on audio classification systems for general use. We will be using a database elaborated from the Freesound samples, which are classified with the labels proposal from Audioset researchers (Google Inc.).

The reference system that will be used in this project is a proposal from researchers Frederic Font Corbera (Universitat Pompeu Fabra), Eduardo Fonoseca (Pompeu Fabra University), Manoj Plakal (Google, Inc.) and Daniel P. W. Ellis (Google, Inc.) in the DCASE Challenge 2018 (Task 2).

The main objective is not to participate in the challenge or develop an improved proposal of the system, but to understand its operation and run it to evaluate its results. This study should serve as an introductory guide for those who want to get into automated systems which use convolutional neural networks through Python language.

So this project includes a detailed analysis of the audio processing and also a guideline to use the reference system with other audio databases (dataset should meet the same requirements as the database used in the study).

Also, there will be some improvement proposals in order to continue this project later based on the experience and knowledge acquired over the degree in audiovisual systems engineering (Grau en Enginyeria de Sistemes Audiovisuals) at ESEIAAT (Escola Superior d'Enginyeries Industrial, Aeroespacial i Audiovisual de Terrassa).

Keywords: Audioset, Freesound, DCASE, audio classification, Python, convolutional nerworks.

Índex

Índex de figures

Índex de taules

Acrònims

CNN *Convolutional Neural Network*: Xarxa convolucional neuronal.

DCASE *Detection and Classification of Acoustic Scenes and Events*: Detecció i classificació d'escenes i esdeveniments acústics.

FSD *Freesound Datasets platform*: Plataforma de conjunts de dades etiquetades de Freesound.

mAP@3 *Mean Average Precision @ 3*: Mitjana de la precisió mitjana per classe de les 3 primeres respostes del sistema.

MFCC *Mel Frequency Cepstral Coefficients*: Coeficients cepstrals de les freqüències de Mel.

MLP *Multilayer Perceptron*: Perceptró multicapa.

STFT *Short Time Fourer Transform*: Transformada de Fourier enfinestrada.

TSC Teoria del Senyal i Comunicacions.

YFCC *Yahoo Flickr Creative Commons 100M dataset*: Base de dades de fotografies i vídeos amb llicència Creative Commons.

1 INTRODUCCIÓ ALS SISTEMES AUTOMATITZATS

Avui dia existeix una quantitat enorme d'informació sonora que rebem i emmagatzemem de tot allò que ens envolta. Gestionar-la cada cop és més complicat i, d'aquesta necessitat, han començat a sorgir sistemes per al processat automàtic de grans bases de dades d'àudio per extreure'n el més rellevant.

Tot i això, encara queda molta recerca pendent per poder obtenir sistemes veritablement fiables per a usar de manera habitual en el nostre entorn.

Per aquest motiu, existeixen cada cop més comunitats que treballen conjuntament per desenvolupar sistemes fiables, útils i cada cop més optimitzats en el temps.

1.1 Tipus de sistemes

Dintre del món dels sistemes automatitzats existeixen diverses classificacions[1] segons els models generats, els tipus d'algorismes usats per al seu desenvolupament i el tipus de problema que estan destinats a resoldre. Els més habituals es mostren a la figura 1.1.

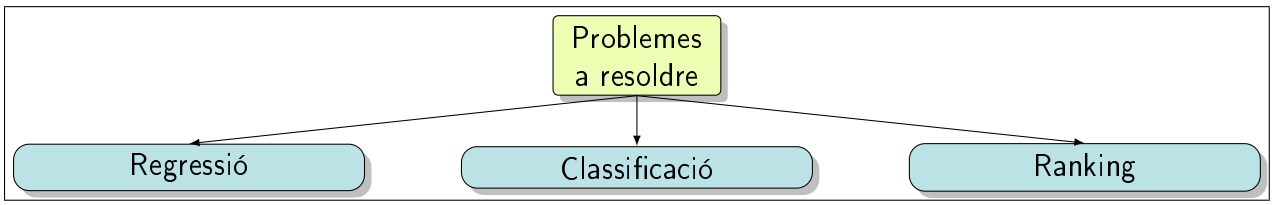


Figura 1.1: Classes de problemes per a sistemes d'aprenentatge automatitzat més habituals

Aquesta classificació queda determinada segons el tipus de sortida que esperem rebre.

a) Regressió

Intenten predir un valor real segons els seus precedents. Per exemple, el valor que un producte del mercat tindrà demà segons el seu històric de preus en el darrer mes.

b) Classificació

Intenten classificar els objectes a partir d'un conjunt determinat de classes establertes. Si només es permet una classe, es tracta d'una classificació *binària* i, si pel contrari, n'admet més, estaríem parlant d'una classificació de *multiclasse*.

[1] La classificació presentada en aquest projecte és la que es proposa a les referències [2] i [18]

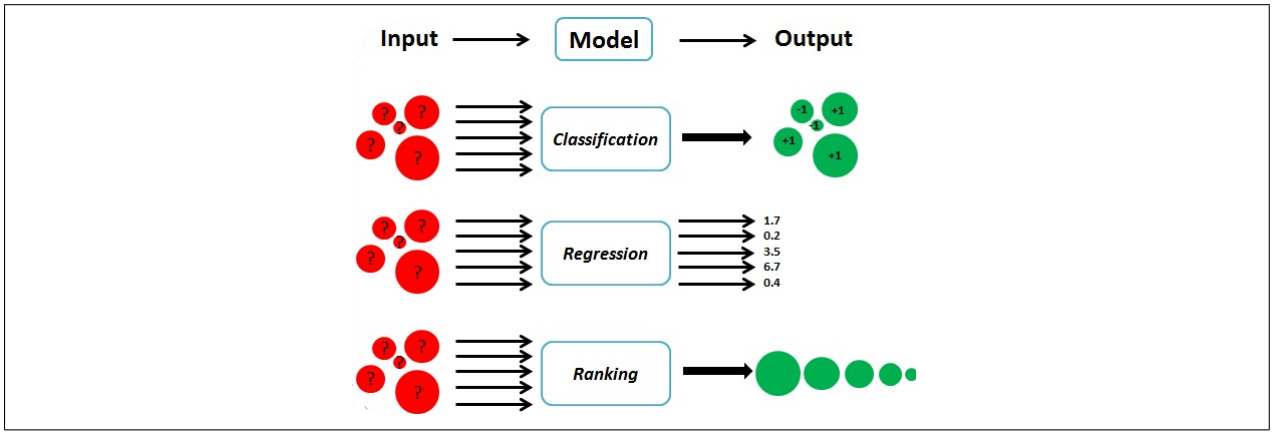


Figura 1.2: Exemples de tipus de resposta segons el problema a resoldre. (Referència [10]: Figura sense títol)

c) Ranking

Intenta predir l'ordre òptim d'un conjunt d'objectes segons un ordre de rellevància predefinit. Els buscadors d'internet en són un bon exemple.

Com ja es detallarà més endavant a l'apartat 2, ens centrarem en els sistemes automatitzats de classificació. És per això que no es profunditzarà més en cap altre tipus de problema que es pugui proposar resoldre per sistemes d'aprenentatge automatitzat.

1.1.1 Models

Cada sistema d'aprenentatge automàtic, després de ser entrenat[2], genera un model per a resoldre la tasca per al qual ha estat dissenyat. Es poden classificar de dues maneres diferents.

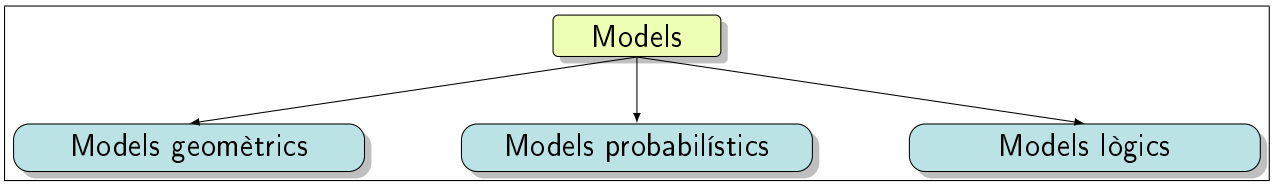


Figura 1.3: Models de sistemes automatitzats segons com es constueixen

Segons com estan construïts els models tenim tres possibles sistemes, tal com es mostra a la figura 1.3.

a) Models geomètrics

Es construeixen sobre l'espai que contempla totes les *instàncies*[3] i poden tenir tantes dimensions com variables tenen les dades. Es genera un *espai d'instàncies* que acostuma a tenir una estructura geomètrica.

Són molt útils quan es treballa amb classificadors lineals bàsics, els quals són força convenients quan existeix una línia que separa les dues classes: a aquest tipus de models se'ls denomina *linealment separables*.

[2] Entrenar un sistema implica haver detectat els patrons entre les dades d'entrada.
[3] Una instància és cadascuna de les dades de les quals es disposen per fer l'anàlisi.

b) Models probabilístics

Intenten determinar la distribució de probabilitat que relaciona les dades d'entrada al sistema amb les seves prediccions. Procuren minimitzar les pèrdues entre ambdues distribucions.

Un dels conceptes clau que determinen aquest tipus de models és l'*estadística bayesiana.*

c) Models lògics

Expressen les probabilitats en regles organitzades en forma d'arbres de decisió. És a dir, generen múltiples camins de decisions.

Aquests arbres de decisió consten d'un node inicial que no és apuntat per cap fletxa. La resta de nodes que s'hi troben, només poden estar apuntats per una única fletxa, per la qual cosa es dedueix que hi ha una única manera d'arribar a cadascuna de les solucions finals, ja que les decisions preses són excloents.

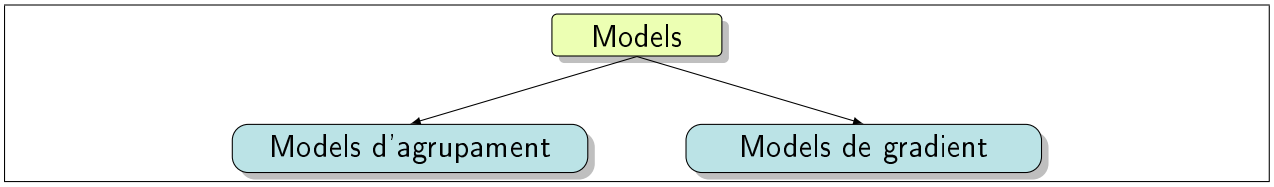
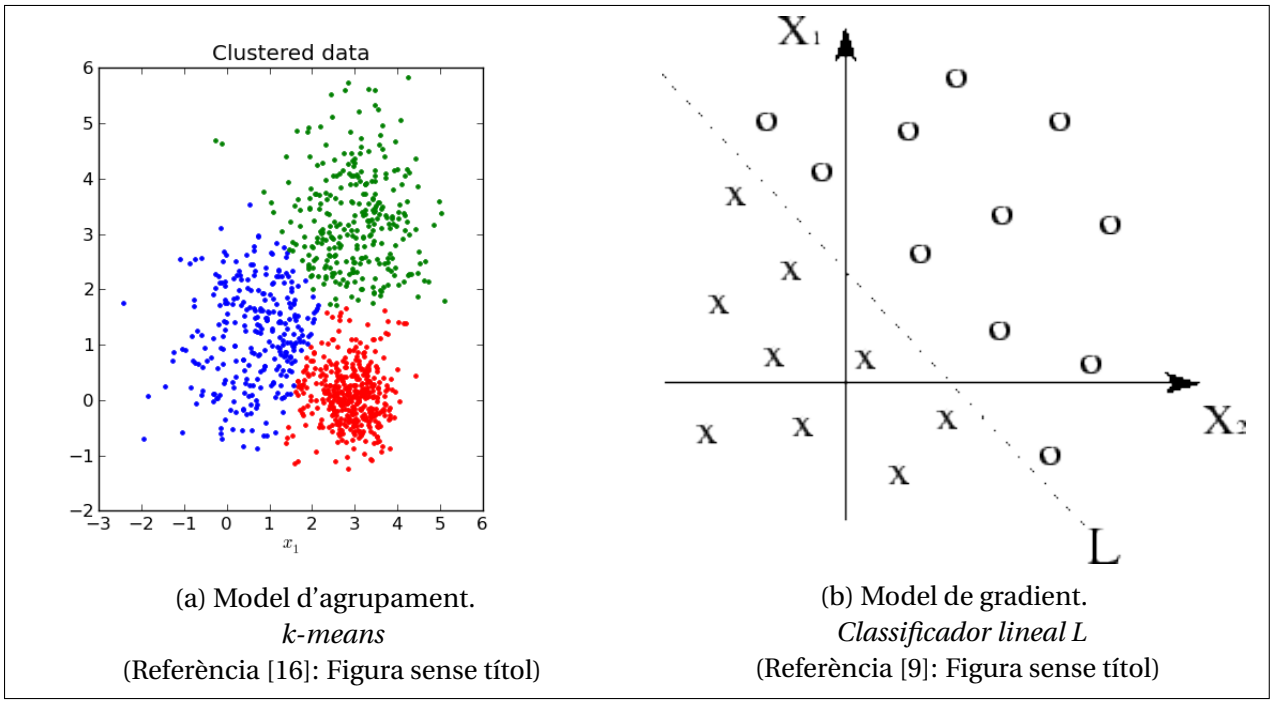


Figura 1.4: Models de sistemes automatitzats segons com es distribueixen

D'altra banda i pel que fa a la classificació expressada a la figura 1.4, tenim dos possibles models.



(a) Model d'agrupament.
k-means
(Referència [16]: Figura sense títol)

(b) Model de gradient.
Classificador lineal L
(Referència [9]: Figura sense títol)

Figura 1.5: Exemples de models segons la classificació expressada a la figura 1.4

Models d'agrupament Procuren dividir l'espai de les instàncies[3] per grups segons la seva proximitat. Es pot triar la quantitat de divisions segons si volem més o menys precisió. La figura 1.5a n'és un exemple.

Models de gradient Com el seu nom indica, representen un gradient en el qual es pot diferenciar entre cadascuna de les instàncies[3]. La figura 1.5b n'és un exemple.

1.1.2 Algorismes

Els diversos algorismes que podem trobar dintre de l'aprenentatge automàtic s'agrupen en una taxonomia en funció de la seva sortida. Els més habituals són els que es mostren a la figura 1.6.

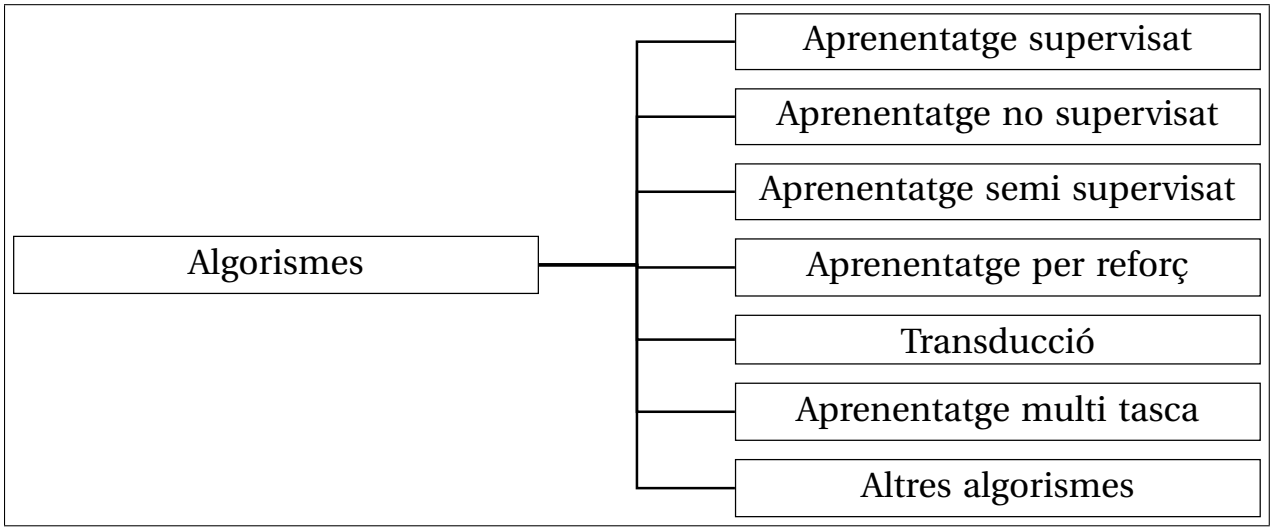


Figura 1.6: Principals algorismes possibles usats en sistemes d'aprenentatge automàtic

a) Aprenentatge supervisat

L'algorisme produeix una funció que estableix una certa correspondència entre les entrades i les sortides desitjades del sistema. Les categories de les instàncies estan prèviament establertes per un factor humà.

b) Aprenentatge no supervisat

L'algorisme només utilitza un conjunt d'instàncies format per entrades del sistema. No es té cap informació sobre les categories d'aquestes instàncies. Per tant, el sistema ha de ser capaç de reconèixer patrons entre les dades per a poder etiquetar les noves entrades.

c) Aprenentatge semi supervisat

És una combinació dels dos algorismes anteriors. Existeixen dades etiquetades i no etiquetades.

d) Aprenentatge per reforç

Es tracta d'un algorisme que aprèn a partir del seu entorn. La informació d'entrada que se li dóna és la retroalimentació que obté de l'exterior derivada de les seves accions.

En comptes d'indicar al sistema que ha de fer, ha d'aprendre a comportar-se mitjançant recompenses o càstigs, segons el seu èxit o fracàs respectivament. L'objectiu és que aprengui a valorar i maximitzar el senyal de recompensa per tal d'optimitzar el seu funcionament i poder prendre decisions correctament.

e) Transducció

Aquest algorisme és similar a l'usat en l'aprenentatge supervisat, malgrat que el seu objectiu principal no és generar una funció de forma explícita, sinó simplement intentar predir les categories de les dades de test segons les instàncies d'entrada i les seves categories.

En certa manera es tracta d'un aprenentatge supervisat dinàmic, ja que totes les instàncies estan etiquetades manualment però els resultats del sistema poden variar en funció de les noves instàncies afegides, ja que no genera cap funció determinant.

f) Aprenentatge multi tasca

Es tracta d'un algorisme que utilitza coneixements previs en un àmbit per resoldre problemes similars. Compta amb les condicions originals i la resolució d'un conjunt de problemes i, a partir d'aquesta informació, desenvolupa un criteri de resolució pròxim a l'utilitzat anteriorment.

1.1.3 Classificació

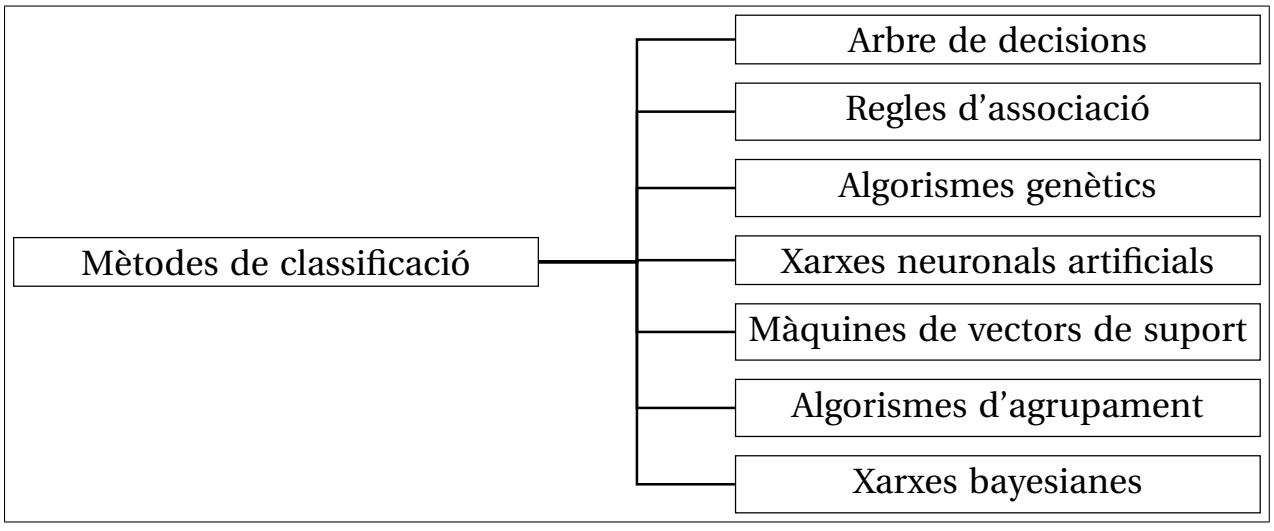


Figura 1.7: Principals mètodes de classificació usats en sistemes d'aprenentatge automàtic

a) Arbre de decisions

Aquest aprenentatge utilitza un arbre de decisions com a model predictiu[4].

[4]Podeu trobar més informació a l'apartat 1.1.1 dintre de la secció **c) Models lògics**.

b) Regles d'associació

S'utilitzen per a descobrir fets o característiques en comú dintre d'un determinat conjunt de dades. Es centren a buscar relacions entre les instàncies a partir de les característiques que se n'hagin extret o bé una relació d'esdeveniments que succeeixen amb el mateix patró de manera habitual.

Es basen en un *precedent* i un *conseqüent*.

c) Algorismes genètics

Són algorismes de cerca heurística que simulen la selecció natural. Les característiques principals d'aquest tipus de cerca són que no garanteix trobar una solució malgrat que existeixi, que si es troba una solució, no necessàriament serà la més òptima i que, en alguns casos, trobarà una solució prou bona en un temps raonable.

d) Xarxes neuronals artificials

És un tipus de classificació inspirada en el funcionament de les neurones en els sistemes nerviosos dels animals.

Es tracta d'un sistema d'enllaços de neurones que col·laboren entre si per produir un estímul de sortida. Cada connexió té un pes numèric que s'adapta en funció de l'experiència. Així doncs, les xarxes neuronals artificials s'adapten als impulsos i són capaces d'aprendre.

e) Màquines de vectors de suport

Es tracta d'una sèrie de mètodes d'aprenentatge supervisats usats per a problemes de regressió i classificació. Utilitzen un conjunt d'instàncies d'entrenament classificats en dues categories.

f) Algorismes d'agrupament

L'agrupament o *clustering* en anglès és la classificació d'observacions en grups segons certs criteris de similitud. És un mètode d'aprenentatge no supervisat.

Normalment es guia per una mesura de similitud específica i pel nivell d'aquesta i la separació entre els diferents grups.

g) Xarxes bayesianes

Es tracta d'un model probabilístic que representa variables a l'atzar i les seves dependències condicionals a través d'un graf acíclic dirigit, la qual cosa implica que no té cicles: per a cada vèrtex v no hi ha un camí directe que comenci i acabi en v.

Per a més informació podeu consultar el Teorema de Bayes a l'Annex A (5.1).

1.2 Classificació d'àudio per etiquetatge

Recollint tota la classificació de l'apartat 1.1 i com a preludi a què es definirà en profunditat més endavant en les seccions 2 i 3, podem definir el sistema objecte d'estudi a partir de la taula 1.1.

Sistema d'etiquetatge d'àudio d'ús general de contingut de *Freesound* amb etiquetes d'*Audioset*	
Model	Model probabilístic
Algorisme	Semi supervisat
Tècnica de classificació	Xarxes neuronals artificials

Taula 1.1: Característiques principals del sistema d'aprenentatge automàtic objecte d'estudi

Es valora com a un sistema semi supervisat, ja que part de les mostres estan etiquetades manualment però no totes.

Tot això es dedueix per la definició que els creadors donen del sistema.

1.3 Objectius

Els objectius principals del projecte són, amb igualtat d'importància:

- Posar en funcionament un sistema real d'aprenentatge automàtic i configurar l'entorn necessari
- Analitzar els resultats obtinguts pel sistema i la seva fiabilitat i eficiència
- Elaborar el plantejament de diverses propostes de millora per aquest sistema seguint la línia del repte proposat al DCASE Challenge 2018: Tasca 2

Existeixen també objectius secundaris, els quals deriven de l'acompliment dels anteriors.

- Augmentar (i reafirmar) els coneixements personals pel que fa als sistemes d'aprenentatge automatitzats de caràcter general
- Millorar l'elaboració de scripts en llenguatge Python per a optimitzar o gestionar el sistema objecte d'estudi
- Elaborar una breu guia introductòria als sistemes automatitzats de propòsit general
- Elaborar una guia a partir de la metodologia seguida per tal de poder posar en funcionament el sistema d'aprenentatge automàtic objecte d'estudi i configurar l'entorn necessari

2 DCASE CHALLENGE

El DCASE Challenge és un repte que neix l'any 2013 de la mà del Comitè Tècnic de Processament de Senyals d'Àudio i Acústics de IEEE (referència [8]), una organització professional dedicada a fer avançar la tecnologia en benefici de la humanitat.

Per estimular la investigació en aquest camp, van considerar convenient llançar la primera edició del DCASE Challenge, un repte de recerca públic accessible per a tots els investigadors.

Avui dia se n'han organitzat un total de 5 edicions amb entre 3 i 5 tasques diferents, totes elles relacionades amb la classificació i detecció d'àudio de manera automatitzada.

En aquest projecte, ens centrarem en les dues darreres edicions, 2018 i 2019, la primera ja conclosa i la segona encara vigent i sense resultats.

2.1 DCASE Challenge 2018

En aquesta edició del DCASE Challenge existeixen un total de 5 tasques, una de les quals consta de tres subtasques diferents.[1]

- Tasca 1: Classificació d'escenes acústiques

  Subtasca 1: Classificació d'escenes acústiques

  Subtasca 2: Classificació d'escenes acústiques amb dispositius de grabació no coincidents

  Subtasca 3: Classificació d'escenes acústiques usant dades externes

- Tasca 2: Etiquetatge d'àudio d'ús general de contingut de *Freesound* amb etiquetes d'*Audioset*

- Tasca 3: Detecció d'àudio per a ocells

- Tasca 4: Detecció d'esdeveniments de so semisupervisada pobrament etiquetada a gran escala d'àmbit domèstic

- Tasca 5: Monitoratge d'activitats domèstiques basada en sons multicanal

[1] Per a més informació, vegeu la referència [4].

2.2 DCASE Challenge 2019

En aquesta edició del DCASE Challenge existeixen un total de 5 tasques, una de les quals consta de tres subtasques diferents.[2]

- Tasca 1: Classificació d'escenes acústiques

  Subtasca 1: Classificació d'escenes acústiques

  Subtasca 2: Classificació d'escenes acústiques amb dispositius de grabació no coincidents

  Subtasca 3: Conjunt obert de classificació d'escenes acústiques

- Tasca 2: Etiquetatge d'àudio amb etiquetes amb soroll i supervisió mínima
- Tasca 3: Localització i detecció de sons
- Tasca 4: Detecció d'esdeveniments de so d'àmbit domèstic
- Tasca 5: Etiquetatge de sons urbans

2.3 Relació entre la tasca 2 del DCASE Challenge 2018 i la tasca 2 del DCASE Challenge 2019

Després de comentar breument quins eren els reptes proposats en aquestes dues edicions del DCASE Challenge (2018 i 2019), les dues tasques que presenten més interès (i una relació directa) són la tasca número 2 de cadascun dels esdeveniments.

Es tracta de dos reptes que comparteixen coordinadors, gran part de l'estructura[3] i gairebé el mateix objectiu principal. En ambdós casos s'utilitza una base de dades a partir de fitxers de *Freesound* i les etiquetes proposades per *Audioset*.

Pel que fa a la base de dades, les principals diferències són que el repte del 2018 utilitza 41 etiquetes i està completament formada per àudios de *Freesound Datasets platform*: Plataforma de conjunts de dades etiquetades de Freesound (FSD), mentre que a l'edició del 2019 s'utilitzen també sons del *Yahoo Flickr Creative Commons 100M dataset*: Base de dades de fotografies i vídeos amb llicència Creative Commons (YFCC) i un total de 80 etiquetes diferents.

En relació als objectius principals d'ambdós reptes, la tasca 2 del 2018 està pensada per avaluar sistemes genèrics d'etiquetatge d'àudio tot usant dades amb anotacions de diferent fiabilitat. La tasca 2 del 2019, en comptes de lidiar amb etiquetes de diferent fiabilitat, avalua sistemes de classificació de sons amb més d'una etiqueta tot utilitzant un petit conjunt de dades etiquetades manualment i un major conjunt de dades etiquetades amb soroll.

[2] Per a més informació, vegeu la referència [5].

[3] Es pot comprovar a partir de les referències [14] i [13].

	Repte 2018	**Repte 2019**
Base de dades	FSD	FSD i YFCC
Etiquetes d'àudio	41 etiquetes possibles 1 etiqueta per fitxer	80 etiquetes possibles Múltiples etiquetes per fitxer
Problemàtica principal	Dades amb etiquetes de diferent fiabilitat	Múltiples etiquetes per a dades etiquetades manualment i dades etiquetades amb soroll.

Taula 2.1: Comparativa de les principals diferècies entre la tasca 2 del DCASE Challenge 2018 i la tasca 2 del DCASE Challenge 2019

Per tant, el repte 2018 és aconseguir bons resultats amb etiquetes de diferent fiabilitat. Per contra, al repte de 2019 es busquen bons resultats en dades amb soroll.

Donada les similituds de les dues tasques, es contempla la possibilitat de fer-ne una comparativa. (Vegeu l'apartat 4.4.2).

A continuació s'exposa un petit resum d'ambdues tasques per separat.

a) Etiquetatge d'àudio d'ús general de contingut de *Freesound* amb etiquetes d'*Audioset* (2018)

Existeixen principalment dos reptes: el primer és construir un model capaç de reconèixer una gran quantitat de sons amb naturaleses molt distants. El segon repte consisteix a aprofitar subconjunts de dades d'entrenament que contenen anotacions de fiabilitat variades, com a reflex de la despesa de tenir anotacions d'alta qualitat.

L'objectiu és proporcionar informació sobre el desenvolupament de classificadors d'esdeveniments de so d'aplicació general amb gran quantitat i diversitat de categories.

Es parteix d'un sistema de referència elaborat pels coordinadors i una base de dades creada a partir de *Freesound* i usant etiquetes d'*Audioset*.

b) Etiquetatge d'àudio amb etiquetes amb soroll i supervisió mínima (2019)

Aquesta tasca està pensada per avaluar sistemes de classificació de sons amb més d'una etiqueta tot utilitzant un petit conjunt de dades etiquetades manualment i un major conjunt de dades etiquetades amb soroll.

La qüestió principal que pretén abordar aquesta tasca és com explotar adequadament una petita però confiable quantitat de dades verificades manualment. A més a més, en procedir les dades de diferents fonts, també demana adaptar-se per a unificar-les.

L'objectiu és proporcionar informació sobre el desenvolupament de classificadors d'esdeveniments de so capaços de fer front al soroll de les etiquetes i les condicions mínimes de supervisió.

No es classifiquen els sons per a una etiqueta, sinó que en poden tenir més d'una. D'altra banda, la quantitat de dades verificades manualment (i que, per tant, té un cost elevat) és inferior per reduir la supervisió humana i el conjunt de dades sorolloses pot contenir més d'un so diferent (per això la sortida pot ser "multi-etiqueta").

3 SISTEMA DE REFERÈNCIA

El sistema de referència de la tasca 2 del DCASE Challenge 2018 es tracta, com bé s'expressa a l'anterior apartat 1.2, d'un sistema d'aprenentatge automatitzat de classificació d'esdeveniments sonors a partir d'un model probabilístic elaborat amb xarxes neuronals i semi supervisat, donada la naturalesa de la base de dades utilitzada.

Aquest sistema ha estat desenvolupat pels investigadors Frederic Font Corbera (Universitat Pompeu Fabra), Eduardo Fonoseca (Universitat Pompeu Fabra), Manoj Plakal (Google, Inc.) i Daniel P. W. Ellis (Google, Inc.).

La valoració del sistema no s'efectua només amb una predicció sinó amb 3. És a dir, la sortida que es retorna donada una entrada a l'atzar és un conjunt de tres etiquetes possibles, ordenades de més a menys probable (segons el model generat pel sistema durant l'entrenament).

Per tant, d'ara en envadant, quan es valori l'eficiència d'aquest s'utilitzarà el *Mean Average Precision @ 3*: Mitjana de la precisió mitjana per classe de les 3 primeres respostes del sistema (mAP@3).

Podeu consultar l'ús de recursos per part dels organitzadors per posar en marxa el sistema a les referències [14] i [7].

3.1 Base de dades

Com bé es detalla a la pàgina web del DCASE Challenge 2018 (referència [4]), s'utilitza una base de dades formada per mostres que es troben a `https://freesound.org/`, un dels majors repositoris col·laboratius amb llicència Creative Commons.

Estan etiquetades a partir de 41 categories proposades per Google a la seva ontologia AudioSet (referència [3]) que es troben a la taula 3.1.

Aquesta base de dades és un subconjunt reduït de la FSD, la qual és una plataforma col·laborativa de col·leccions lliures d'àudio, en constant creixement, de gran dimensió i de caràcter general.

Totels les dades tenen una sola etiqueta, malgrat que no totes han estat manualment etiquetades (esmentat amb anterioritat als apartats 1.2 i **??**).

Tearing	Saxophone	Violin, fiddle
Bus	Oboe	Double bass
Shatter	Flute	Cello
Gunshot, gunfire	Clarinet	Chime
Fireworks	Acoustic guitar	Cough
Writing	Tambourine	Laughter
Computer keyboard	Glockenspiel	Applause
Scissors	Gong	Finger snapping
Microwave oven	Snare drum	Fart
Keys jangling	Bass drum	Burping, eructation
Drawer open or close	Hi-hat	Cowbell
Squeak	Electric piano	Bark
Knock	Harmonica	Meow
Telephone	Trumpet	

Taula 3.1: Llistat de categories usades en el sistema de classificació de la tasca 2 del DCASE Challenge 2018. (Referència [4]: Taula *Vocabulary of 41 labels from Google's AudioSet Ontology*)

a) Dades d'entrenament

El conjunt d'entrenament per a desenvolupar el sistema inclou aproximadament 9.5k de mostres distribuïdes de manera desigual entre les 41 etiquetes que es mostren a la taula 3.1. El nombre de mostres d'àudio per a cadascuna de les categories oscil·la entre 94 i 300. La seva durada varia entre 300 mil·lisegons i 30 donada la diversitat de les classes de sons i les preferències dels usuaris de Freesound a l'hora de gravar-los.

Del total de dades d'entrenament, aproximadament unes 3.7k han estat verificades manualment, mentre que el 5.8k restant, no. La qualitat de les anotacions no verificades s'ha estimat aproximadament al 65-70% en cada categoria de so. També es proporciona una anotació per a cadascuna de les mostres on s'indica si ha estat o no manualment verificada.

b) Dades de test

El conjunt de test està compost d'aproximadament 1.6k anotacions verificades manualment amb una distribució de categoria similar a la del conjunt d'entrenament.

Totes les dades tenen una freqüència de mostratge de 44.1kHz, estan codificades en 16 bits amb signe en PCM, són mono i es desen en format WAV.

Les dades estan recollides en 4 arxius comprimits amb l'estructura de la figura 3.1

3.2 Codi del sistema de referència

El sistema de referència implementa un classificador que usa una *Convolutional Neural Network*: Xarxa convolucional neuronal (CNN), similar als models convolucionals profunds usats en el domini visual però amb menys nivells.

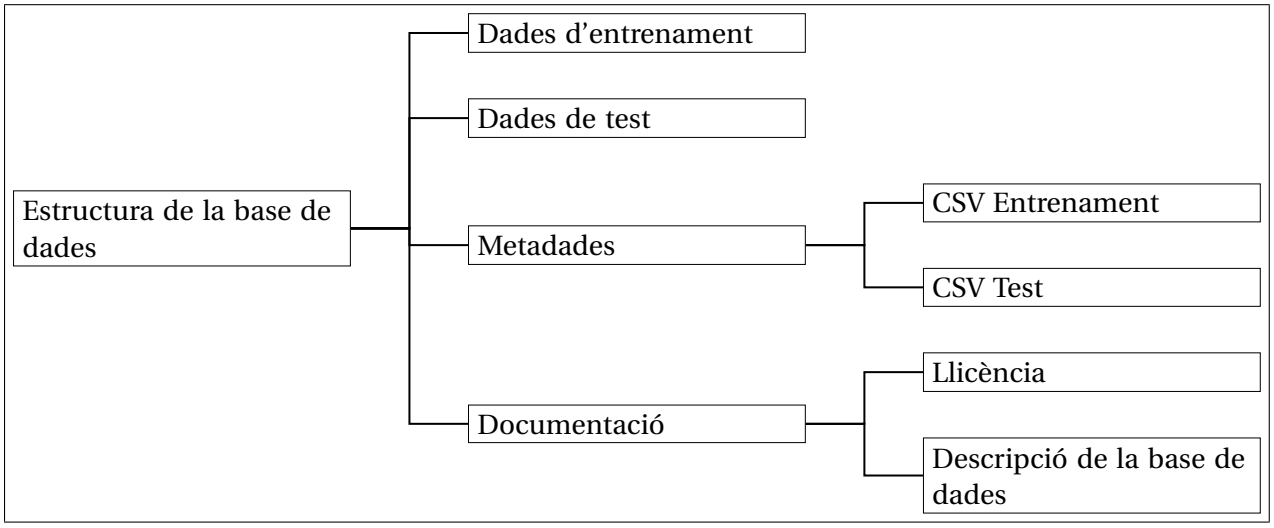


Figura 3.1: Estructura de la base de dades preparada per a descarregar-la

El model agafa instàncies fraccionades de l'espectrograma dels *Mel Frequency Cepstral Coefficients*: Coeficients cepstrals de les freqüències de Mel (MFCC) com a entrada del sistema i produeix prediccions classificades sobre les 41 classes del conjunt de dades.

També podem entrenar el sistema com a un classificador *Multilayer Perceptron*: Perceptró multicapa (MLP), completament connectat, el qual és més senzill que una xarxa convolucional neuronal.

3.2.1 Estructura

Estructura de fitxers Segons la referència [14], l'estructura de fitxers de codi que forma el sistema és la següent:

- **main.py**: Motor principal del sistema. S'usa per posar en marxa tots els processos.
- **train.py**: Bucle d'entrenament. El crida *main.py* quan s'activa la opció - -*mode train*.
- **evaluation.py**: Bucle d'avaluació. Calcula les mètriques d'avaluació. El crida *main.py* quan s'activa la opció - -*mode eval*.
- **inference.py**: Bucle de test. Genera les prediccions segons els models entrenats. El crida *main.py* quan s'activa la opció - -*mode test*.
- **inputs.py**: Entrada de *tensorflow* per a decodificar les dades dels fitxers CSV i WAV i generar instàncies fraccionades de l'espectrograma de Mel.
- **model.py**: Definició dels models en el format de *tensorflow*.
- **make_class_map.py**: Eina per crear un mapa de les classes des de les dades d'entrenament.

L'ordre d'ús d'aquest comença per *make_class_map.py*, ja que sense el mapa d'etiquetes el sistema no pot treballar.

A partir d'aquí, s'entrena el sistema amb *train.py*, s'avalua (si es desitja) amb *evaluation.py* i, finalment, es generen les prediccions de les dades de test amb *inference.py*.

Característiques de les dades d'entrada Com s'ha anat mencionant al llarg de l'apartat 3, la informació dels àudios utilitzada pel sistema està en MFCC. Això implica un càlcul previ de totes les dades que s'utilitzaran.

1. Es genera l'espectrograma a partir de la *Short Time Fourer Transform*: Transformada de Fourier enfinestrada (STFT) amb una finestra de 25ms, amb 10ms entre transformades successives i un enfinestrament periòdic de Hann.

2. L'espectrograma de Mel es càlcula tot mapejant l'espectrograma original a 64 grups que ocupen el rang 125-7500 Hz. L'escala de Mel està representada de manera que sigui el més semblant possible a la percepció humana d'àudio, de manera que hi ha més conjunts de mostres a baixa freqüència i menys en alta freqüència.

3. L'espectrograma estabilitzat dels logarítmes de Mel es calcula a partir del logaritme de l'espectrograma de Mel tot afegint-hi un *offset* de 0.001 per evitar el logaritme de 0.

4. L'espectograma logarítmic de Mel es divideix en instàncies superposades amb una finestra de 0.25s i un espai de 0.125s entre finestres successives. La superposició permet generar més instàncies a partir de les mateixes dades, la qual cosa ajuda a fer més efectiva la dimensió de la base de dades.

Arquitectura xarxa neuronal convolucional El sistema de referència utilitza un model de xarxa neuronal convolucional de 3 capes bidimensionals convolucionals que alterna capes de *max-pooling*[1] seguit d'una capa final de *max-reduction*[2] i una capa *softmax*[3]. (Vegeu la taula 3.2).

El format en el qual es troben escrites les capes és (mida del kernel, punt d'estat del procés, mapa de característiques) i (mida del kernel, punt d'estat del procés).

Podeu trobar més informació sobre aquesta arquitectura a les referències [14] i [11].

3.2.2 Paràmetres

Els paràmetres usats per defecte pel sistema són els que es mostra a la figura 3.2.

L'ús dels més destacats, com la dimensió de la finestra o del salt entre finestres successives, s'explica a la secció anterior (apartat 3.2.1).

Per a més informació, consulteu la referència [14].

[1] Procés senzill de discretització.
[2] Capa que serveix per a produir només un valor per cada mapa de característiques.
[3] Capa que serveix per a 'comprimir' un vector k-dimensional.

```
hparams = tf.contrib.training.HParams(
    # Window and hop length for Short-Time Fourier Transform applied to audio
    # waveform to make the spectrogram.
    stft_window_seconds=0.025,
    stft_hop_seconds=0.010,
    /* Parameters controlling conversion of spectrogram into mel
    spectrogram. */
    mel_bands=64,
    mel_min_hz=125,
    mel_max_hz=7500,
    # log mel spectrogram = log(mel-spectrogram + mel_log_offset)
    mel_log_offset=0.001,
    # Window and hop length used to frame the log mel spectrogram into
    # examples.
    example_window_seconds=0.250,
    example_hop_seconds=0.125,
    # Number of examples in each batch fed to the model.
    batch_size=64,
    /* For the 'mlp' multi-layer perceptron, nl=# layers, nh=# units per
    layer. */
    nl=2,
    nh=256,
    # Standard deviation of the normal distribution with mean 0 used to
    # initialize the weights of the model. Biases are initialized to 0.
    weights_init_stddev=1e-3,
    # Learning rate.
    lr=1e-4,
    # Epsilon passed to the Adam optimizer.
    adam_eps=1e-8,
    # Classifier layer: one of softmax or logistic.
    classifier='softmax')
```

Figura 3.2: Fragment de codi extret directament de *main.py* del sistema de referència

Layer	**Actvation Shape**	**Weights**	**Multiplies**
Input	(25,64,1)	0	0
Conv2D(7x7, 1, 100)	(25. 64. 100)	4.9K	7.8M
MaxPool2D(3x3, 2x2)	(13, 32, 100)	0	0
Conv2D(5x5, 1, 150)	(13, 32, 150)	375K	156M
MaxPool2D(3x3, 2x2)	(7, 16, 150)	0	0
Conv2D(3x3, 1, 200)	(7, 16, 200)	270K	30.2M
ReduceMax	(1, 1, 200)	0	0
Softmax	(41,)	8.2K	8.2K
Total		**658.1K**	**194.1M**

Taula 3.2: Arquitectura de la xarxa neuronal convolucional usada al sistema de referència de la Tasca 2 del DCASE Challenge 2018. (Referència [14]: Taula *Architecture*)

4 POSADA EN MARXA DEL SISTEMA

En aquest capítol és on té lloc les proves i avaluació del sistema de referència esmentat al capítol 3. Per tal de complir un dels objectius secundaris (Elaborar una guia a partir de la metodologia seguida per tal de poder posar en funcionament el sistema d'aprenentatge automàtic objecte d'estudi i configurar l'entorn necessari, apartat 1.3), s'hi incorpora la secció 4.1, on es parla de les necessitats del sistema i algunes problemàtiques que es pot trobar hom que desitgi engegar-lo.

D'altra banda també, s'hi adjunta un anàlisi detallat dels resultats obtinguts, el qual procura destacar l'evolució del sistema al llarg del procés i no només la qualitat del millor model de predicció.

Per completar l'estudi, dintre d'aquest capítol també trobem una comparativa entre els diferents resultats segons els paràmetres inicials i una breu conclusió amb altres tasques que s'han realitzat al llarg d'aquest projecte.

4.1 Configuració del sistema i consideracions previes

Un cop analitzat el sistema de referència que ens proporcionen els responsables de la tasca 2 del DCASE Challenge 2018, és hora de posar-lo en marxa. Amb aquest objectiu, partim de la documentació que s'hi adjunta a la web.

Tal com trobem en el repositori Github del DCASE Challenge 2018: Task 2, per fer funcionar el sistema necessitem les següents llibreries de Python:

- Numpy
- Tensorflow

Per complir aquests requisits, s'ha creat un entorn virtual de Python amb l'ajuda de l'eina **virtualenv**. Aquest està configurat per usar la versió de Python 2 i, a través del sistema de gestió de paquets **pip**, s'hi instal·len també les extensions esmentades en la llista anterior.

Les versions usades en aquest projecte són les que es mostren a la taula 4.1.

Pel que fa a la llibreria de *tensorflow*[1], s'han utilitzat dues versions diferents així com, al llarg del projecte, també s'han utilitzat dos setups: un PC i el servidor del departament de TSC de la Universitat Politècnica de Catalunya, ubicat al Campus Nord.

[1] Per a més informació, vegeu la referència [1].

Python	2.7.15rc1
Numpy	1.16.3
Tensorflow	1.5.0 / 1.13.1

Taula 4.1: Versions de llibreries i de Python usades per al correcte funcionament del sistema

Aquest servidor compta amb 9 particions diferents, algunes de les quals estan reservades per a tasques concretes. Cadascuna d'aquesta compta amb diferents recursos i les tasques que s'hi envien es distribueixen segons els requeriments de l'usuari propietari. Com aquesta assignació és aleatòria (dintre de les condicions de cada sol·licitud) i no totes les particions tenen la mateixa arquitectura de processador, no es pot preveure si el sistema presentarà algun tipus d'incompatibilitat amb el *build* de *tensorflow* (s'utilitza el de *pip*). Així doncs, s'opta per la versió 1.5.0, la qual no presenta errors en cap de les particions.

Partició	**CPUs**	**Memòria RAM (GB)**	**Freqüència CPUs (GHz)**
veuc01.tsc.upc.edu	40	251.80	3.10
veud02	12	62.59	1.70
veuc02	8	31.41	3.00
veuc04	32	94.36	2.80
veuc05	48	251.79	3.10
veuc06	48	251.79	3.10
veuc07	24	94.35	2.79
veuc08	24	94.35	2.79
veuc09	16	251.57	3.00

Taula 4.2: Mapa de recursos del servidor *Veu* del departament de TSC. Dades extretes de `http://calcula.tsc.upc.edu/ganglia/`

Tanmateix, la tria d'aquesta versió impossibilita l'ús de *tensorboard* (automàticament instal·lat amb *tensorflow* 1.13). Malgrat que el sistema de referència està preparat per suportar-lo i, per tant, visualitzar l'evolució en temps real de l'entrenament, avaluació o test del sistema, en aquesta versió no es pot usar.

El codi utilitzat és l'extret del repositori Github del DCASE Challenge 2018: Task 2. No s'hi ha realitzat cap modificació al respecte. El conjunt de dades (disponibles a la competició de Kaggle de l'any 2018 [6]) sí que ha patit alguns canvis.

Tal com recomanen a la documentació del repositori, s'han extret un 10% de les dades d'entrenament per tal de tenir dades d'avaluació. D'altra banda, també s'han canviat els documents CSV que emmagatzemen totes les dades dels fitxers d'entrenament, avaluació i test respectivament, ja que la lectura no es realitzava correctament dintre del sistema. Hi resten les columnes del nom i l'etiqueta en tots tres casos i també la de si s'ha verificat manualment en train i eval.

Per a generar les dades d'avaluació s'ha usat un parell de scripts de Python. Tots dos es troben disponibles a l'Annex B. El primer llegeix l'arxiu CSV amb tots els fitxers d'entrenament i n'extreu un 10% de dades de manera aleatòria. Tot seguit, genera un nou document CSV amb el llistat d'àudios i característiques d'aquests de les dades d'avaluació. El segon script s'encarrega de moure tots els fitxers des de la carpeta d'entrenament a la carpeta d'avaluació.

Entrenament	8525 fitxers d'àudio
Avaluació	948 fitxers d'àudio
Test	1600 fitxers d'àudio

Taula 4.3: Dades usades per al sistema de referència

El conjunt de dades resultant i l'estructura de fitxers després de configurar el sistema es mostren a la taula 4.3 i a la figura 4.1 respectivament.

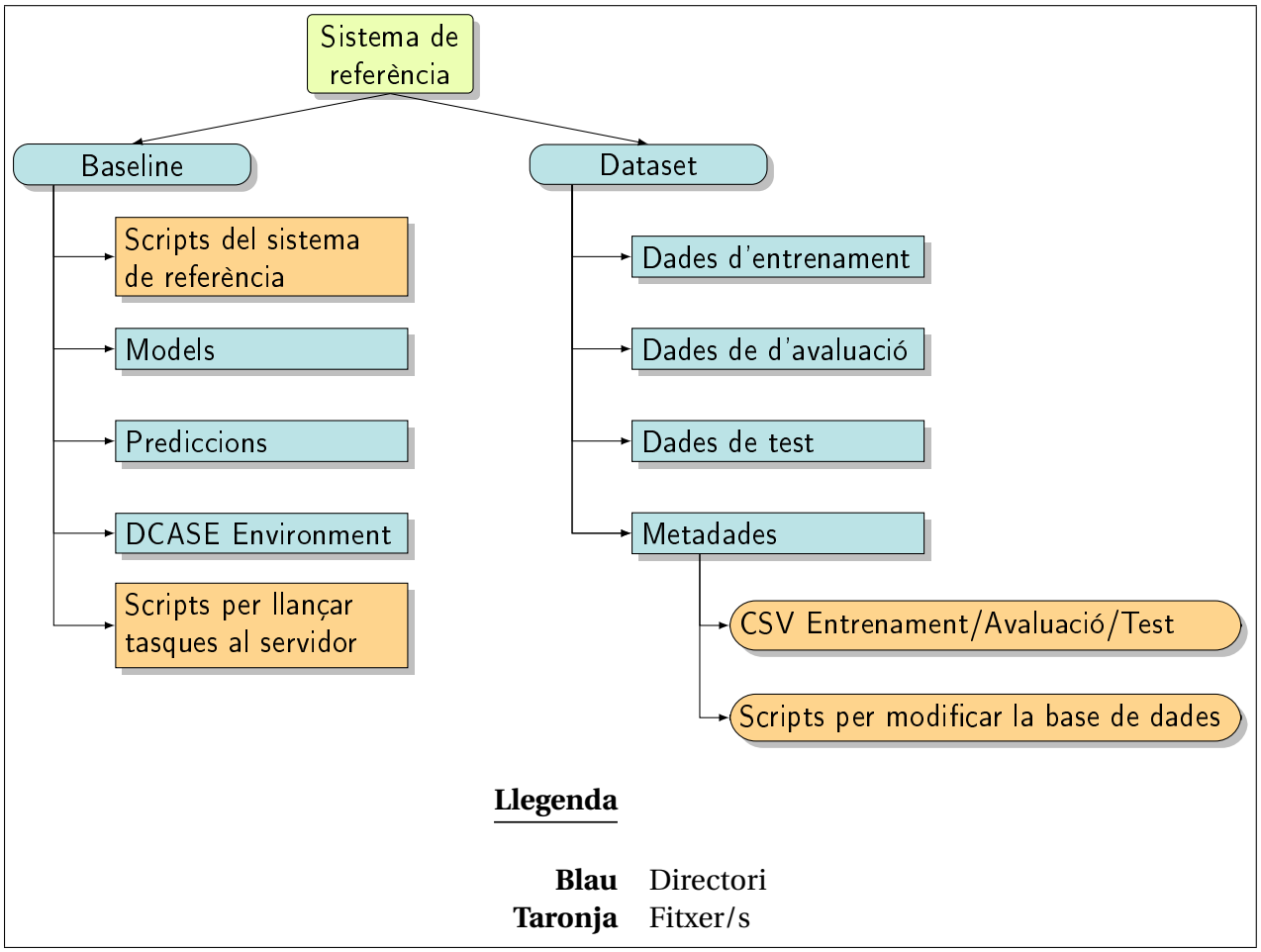


Figura 4.1: Diagrama de l'estructura de dades de la configuració

4.2 Resultats i anàlisi

Un cop tenim preparada la base de dades i la documentació que hi fa referència, procedim a iniciar el sistema com a tal. Per a cadascuna de les fases del sistema (entrenament, avaluació i test), s'ha elaborat un o més scripts per tal de llançar-los al servidor.

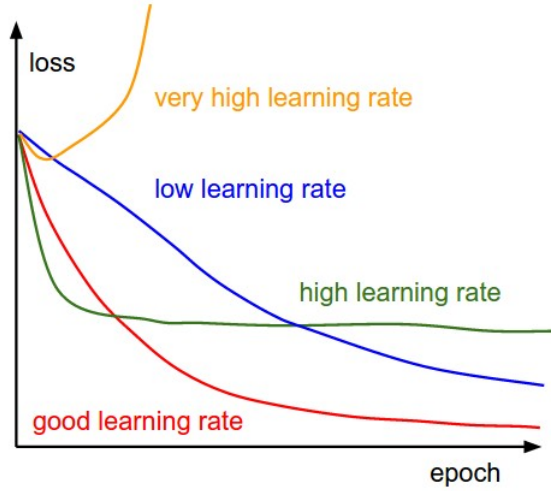
Els recursos utilitzats per a cada tasca són els especificats en la taula 4.4.

Malgrat que el cost computacional pot ser similar en els tres processos, el consum de memòria és força més elevat a l'avaluació i al test, ja que la lectura de fitxers es manté accessible i, com s'usen tots els models generats, les dades s'acumulen.

Tasca	Entrenament	Avaluació	Test
CPUs	24	24	24
RAM	12 GB	24 GB	24 GB
Freqüència	Sense definir	mínim 3 GHz	mínim 3 GHz

Taula 4.4: Recursos del servidor usats per a cadascuna de les etapes del sistema

Pel que fa a l'entrenament, s'han utilitzat un total de 8525 fitxers de so, els quals han generat un total de 405 models[2]. Al llarg d'aquest procés ens retorna l'evolució de les pèrdues entre el model real i el model pronosticat pel sistema, és a dir, l'entropia creuada, malgrat que aquests valors no són probabilitats, ja que usem un sistema no escalat.



Llegenda

Groc	Taxa d'aprenentatge molt alta
Blau	Taxa d'aprenentatge baixa
Verd	Taxa d'aprenentatge alta
Vermell	Taxa d'aprenentatge bona

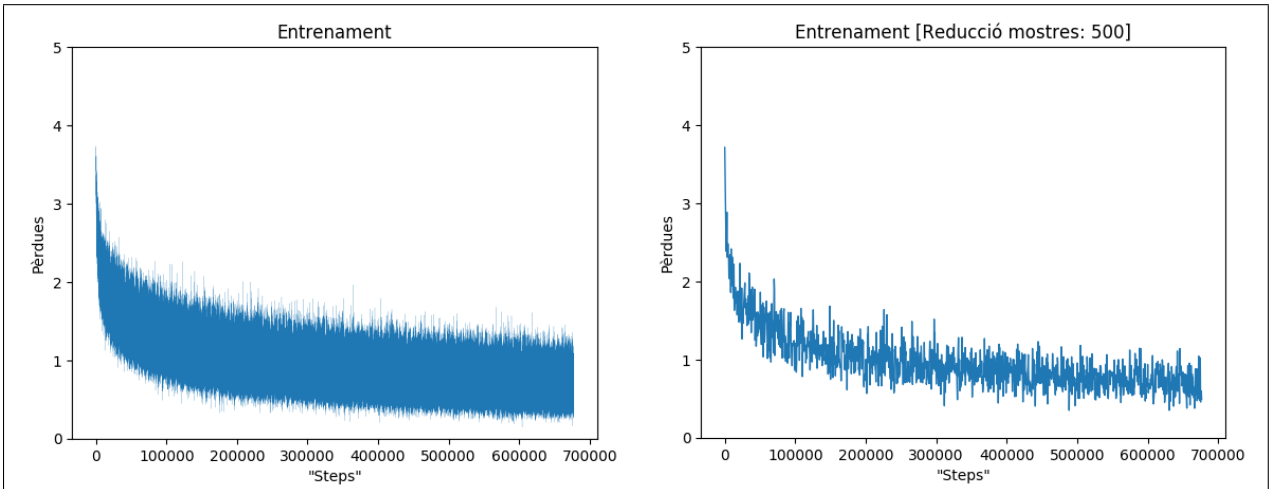
Figura 4.2: Corbes segons la taxa d'aprenentatge. (Referència [15]: Figura *Learning Rates*)

Dintre dels possibles resultats, ens trobem principalment amb 4 tipus de sistemes segons la seva taxa d'aprenentatge[3]. En valors molt elevats, el resultat és gairebé exponencial i el sistema no funciona correctament. Si pel contrari és molt baix, l'evolució de les pèrdues, tot i ser positiva, és gairebé lineal i, per tant, no resulta gaire òptima.

Entre una bona taxa d'aprenentatge i una lleugerament elevada, el sistema treballa correctament i redueix les pèrdues amb prou agilitat. Com més alta sigui la taxa i, per tant, més rapidament decreixin les pèrdues, els resultats tendeixen a estancar-se en alts valors. Pel que fa a un valor més correcte d'aprenentatge, l'entropia creuada tendirà a 0 i, per tant, el model pronosticat distarà molt poc del real.

[2]Cada 15 minuts d'entrenament es generà un arxiu de *checkpoint* amb un model entrenat. Per tant, l'entrenament ha durat 4 dies i 13 minuts aproximadament. Aquest temps de salvat és modificable.

[3]Per a més informació, vegeu les referències [20] i [15].



(a) Total de mostres

(b) Mostres reduïdes a $\frac{\text{Total de mostres}}{500}$

Figura 4.3: Evolució de les pèrdues al llarg de l'entrenament

Com es pot comprovar a través de les gràfiques de la figura 4.3 i comparant amb la figura 4.2[4], tot i tenir pics de més o menys pèrdues, aquestes es redueixen significativament en els 100.000 primers *steps*. Tanmateix, la taxa d'aprenentatge és prou bona, ja que no és lineal ni tampoc s'estanca en valors gaire alts.

Havent finalitzat l'entrenament, prosseguim tot avaluant els models amb un total de 948 mostres d'àudio, extretes prèviament del conjunt inicial d'entrenament. Per poder visualitzar l'evolució real de la precisió dels models, s'han avaluat les dades amb els 405 models generats durant l'entrenament.

Per a cadascun dels models, se'ns retorna el mAP@3 global del sistema i el de cadascuna de les etiquetes possibles. També ens mostra les 3 primeres prediccions per a cada so, ordenades de major a menor semblança amb les classificacions possibles, en el format:

[Etiqueta del so], [Predicció 1, predicció 2, predicció 3]

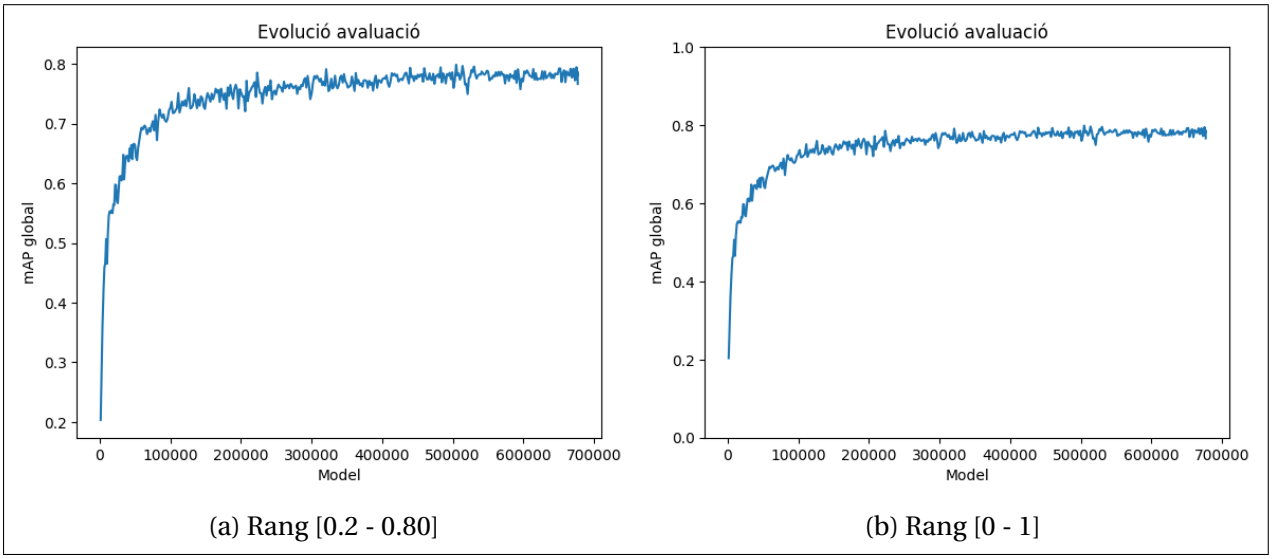
De tal manera que així podem veure quins són els sons que es confonen més sovint i amb quina prioritat.

En el gràfic de la figura 4.4 es mostra com evoluciona el mAP@3 de l'avaluació segons el model usat. S'hi pot seguir com milloren els models al llarg de l'entrenament. A partir d'aquí hem extret el model amb millors resultats, el qual es mostra a la figura 4.5.

A partir d'aquest gràfic podem visualitzar en quins àmbits funciona millor el nostre sistema. Per exemple, determina sense cap error tots els sons procedents de metal·lòfons (*glockenspiel* amb 16 mostres[5]). En canvi, el so de les tisores (*scissors* amb 19 mostres[5]) és l'etiqueta amb menor percentatge d'encerts.

[4]Un *epoch* és el conjunt de *steps* que triga el sistema a recórrer totes les mostres. Per tant, ambdós gràfics (Figures 4.2 i 4.3) es troben en funció de la mateixa variable (1 *epoch* = x *steps*). Per a més informació, vegeu la referència [19].

[5]Per a totes les etiquetes hi ha un total d'entre 15 i 19 mostres.



(a) Rang [0.2 - 0.80]

(b) Rang [0 - 1]

Figura 4.4: mAP@3 de cadascun dels models entrenats en l'avaluació

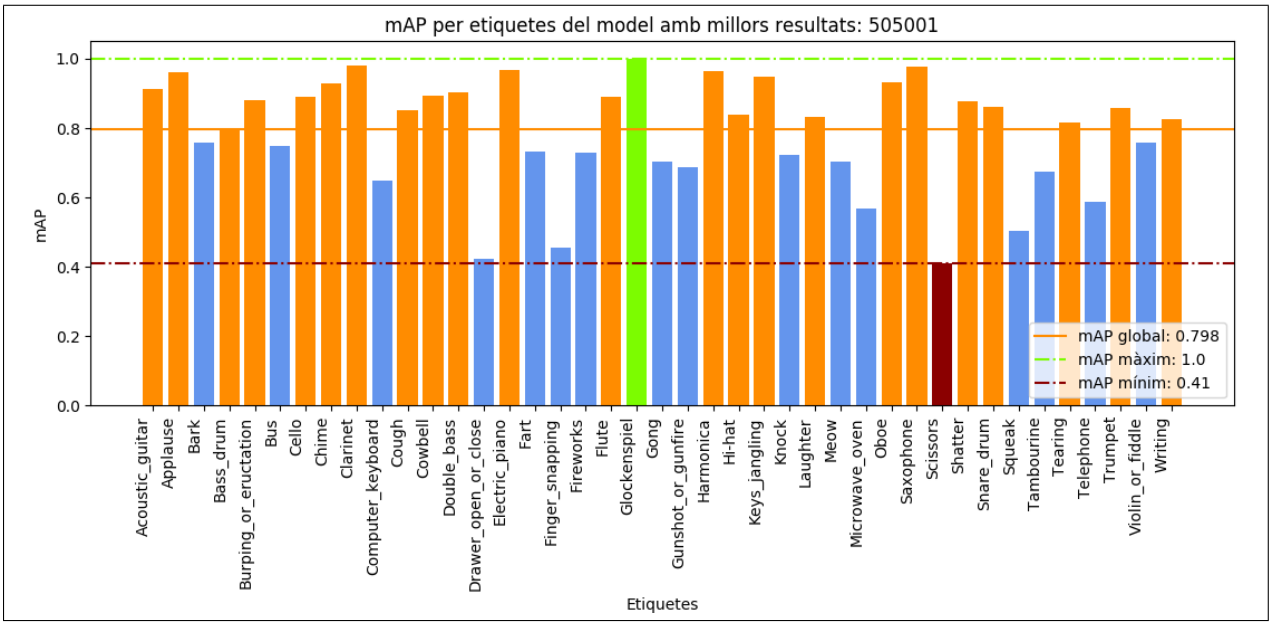


Figura 4.5: mAP@3 per etiquetes del model amb millors resultats a l'avaluació

També és important destacar que, d'un total de 41 etiquetes, 23 han estat classificades amb un mAP@3 de més de 0.798 (mAP@3 global del model) i, per tant, estem davant d'un sistema que té un bon rendiment per a gairebé el 50% de les categories establertes tot usant les dades d'avaluació.

Per finalitzar el procés, valorarem el sistema a partir de les dades de test, que són un conjunt de 1600 fitxers d'àudio. Així com hem fet amb l'avaluació, hem usat absolutament tots els models generats durant l'entrenament per així comparar si els millors resultats ens els ofereix el mateix model de l'avaluació.

El diagrama de la figura 4.6 és la millora de resultats segons el model usat. És l'equivalent de la figura 4.5 però amb les dades de test.

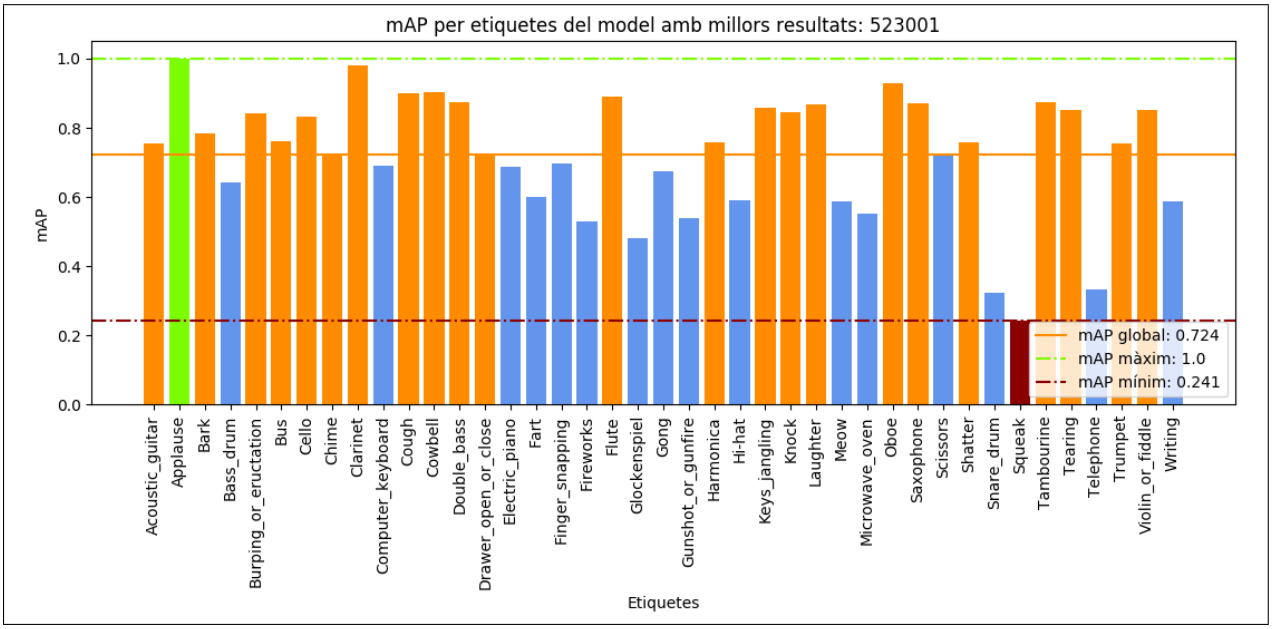
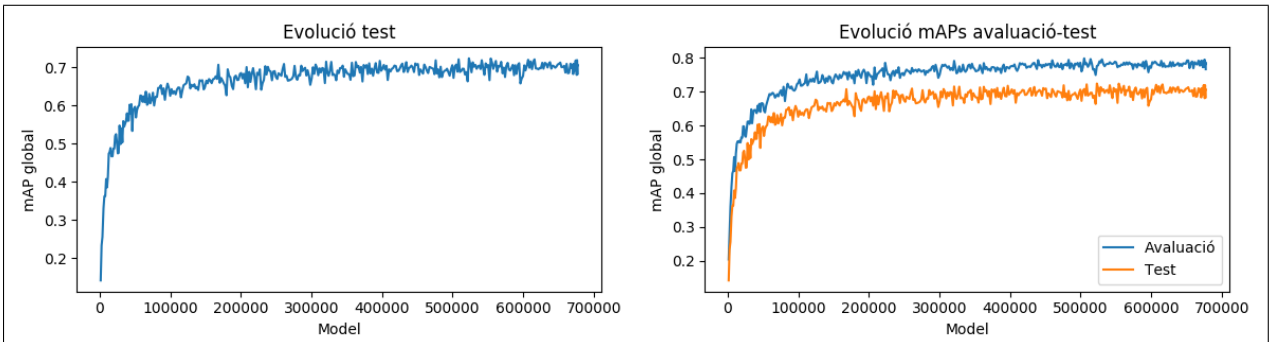


Figura 4.6: mAP@3 per etiquetes del model amb millors resultats al test

En aquest cas podem veure que l'etiqueta amb millors resultats, amb un 100% d'encerts, són els aplaudiments (*applause* amb 32 mostres[6]). D'altra banda, l'etiqueta amb menor percentatge d'encert, amb un mAP@3 de només 0.241, són els grinyols (*squeak* amb 29 mostres[6]).

De la mateixa manera que amb el millor model en l'avaluació, d'un total de 41 etiquetes, 23 han estat classificades amb un mAP@3 de més de 0.724 (mAP@3 global del model). Per tant, tornem a obtenir més d'un 50% d'etiquetes amb un rendiment superior a la mitja.

Com a curiositat, podríem destacar que, malgrat que el so del saxòfon es troba present en 110 del total de 1600 mostres i, per tant, és l'etiqueta amb major representació a les dades de test, no obté el millor resultat. Això pot ser degut a la poca presència de fitxers d'àudio d'aquest tipus al llarg de l'entrenament o bé la diferència entre tensors no és tan notable com en altres casos.



(a) Evaluació mAP@3s en test

(b) Comparativa dels avaluació-test

Figura 4.7: mAP@3 de cadascun dels models entrenats en el test

Per finalitzar amb l'anàlisi dels resultats, cal remarcar la diferència entre l'evolució de l'eficiència dels models amb les dades d'avaluació i amb les dades de test. Tal com es pot visualitzar a la figura 4.7b, el creixement inicial és similar però pel que fa al test s'estanca abans.

[6]Per a totes les etiquetes hi ha un total d'entre 25 i 110 mostres.

També es distancien els resultats pel que fa al seu punt àlgid. El percentatge més gran d'encerts amb les dades d'avaluació es troba abans que amb les dades de test, sent el primer superior al segon (vegeu la taula 4.5).

	Model	**Valor**
Màxim mAP@3 global en avaluació	505001	0.798
Màxim mAP@3 global en test	523001	0.724
Mínim mAP@3 global en avaluació	1251	0.204
Mínim mAP@3 global en test	1251	0.142

Taula 4.5: Resum dels resultats obtinguts

Per acabar amb aquesta secció i tot fent referència als resultats exposats a la comunitat del DCASE Challenge 2018 [4], el mAP@3 obtingut és diferent ja que hem usat part de la base de dades d'entrenament per efectuar una avaluació prèvia al test.

4.3 Comparativa entre paràmetres inicials

Tal com s'ha comentat a l'inici d'aquest capítol, el sistema també s'ha avaluat canviant alguns dels paràmetres inicials (detallats a la secció 3.2.2). Ara bé, donat l'elevat cost de temps del procés d'entrenament, només s'ha utilitzat un 5% de les dades d'entrenament i de test (aproximadament uns 500 fitxers d'entrenament i 100 de test).

La intenció és obtenir un resultat aproximat de la tendència dels models entrenats segons els paràmetres inicials establerts i no pas una resposta exacta.

Els paràmetres modificats són:

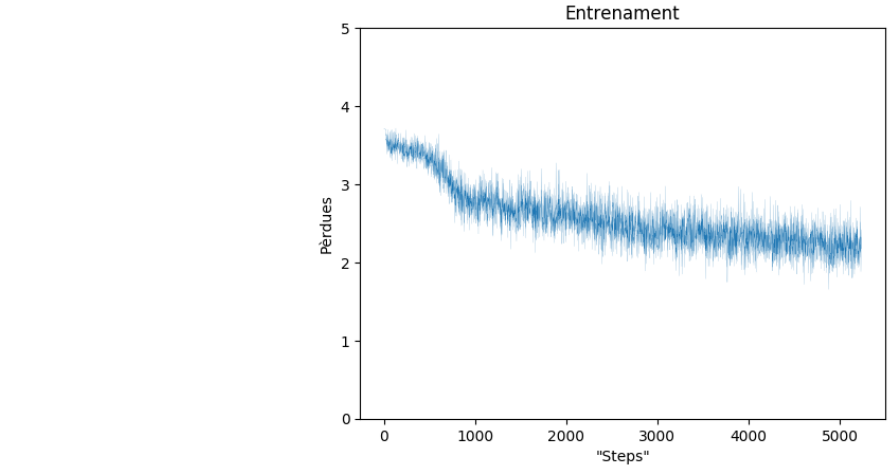
- Durada de la finestra i del salt d'enfinestrament
- Quantitat de bandes de Mel
- Quantitat d'instàncies a cada iteració (*batch size*)
- Taxa d'aprenentatge

En tots els casos s'han utilitzat un valor inferior i un de superior respecte al valor per defecte.

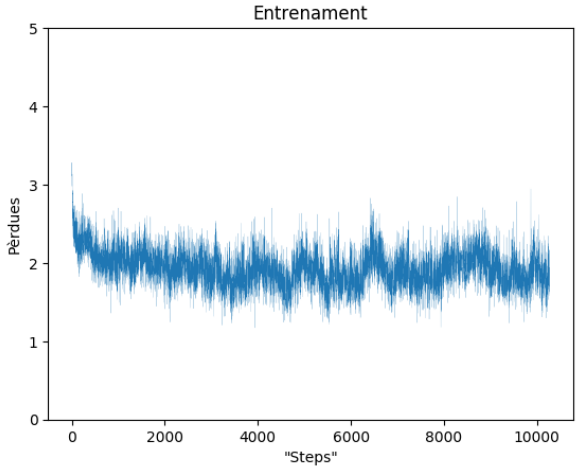
Com les dades usades són diferents de les de l'apartat 4.2, també s'ha avaluat el sistema amb els paràmetres per defecte i el conjunt d'àudio reduït. Es consideraran els primers 5 *epochs* de cada entrenament.

A partir de la figura 4.8 podem extreure les següents conclusions:

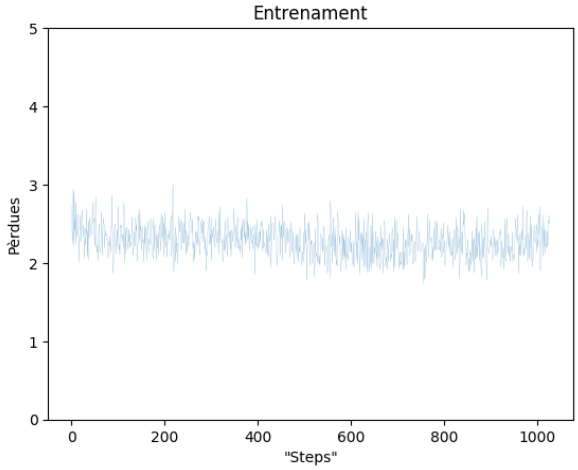
Durada de la finestra i del salt d'enfinestrament: A partir de les figures 4.8b i 4.8c podem comprovar com, a menor és l'enfinestrament, obtenim més models i les pèrdues disminueixen una mica més. Ara bé, això té un cert cos de temps, malgrat que poder no és prou elevat per a suposar un gran inconvenient.



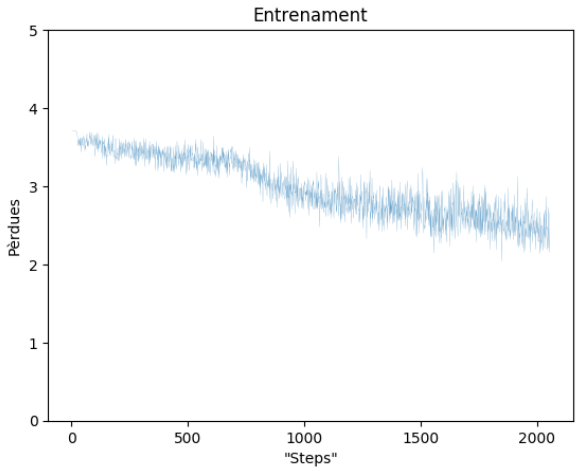
(a) Per defecte



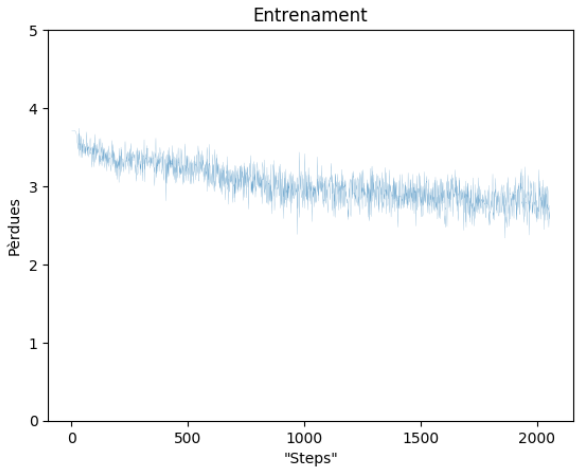
(b) Finestra = 0.05s i salt = 0.025s



(c) Finestra = 0.5s i salt = 0.25s

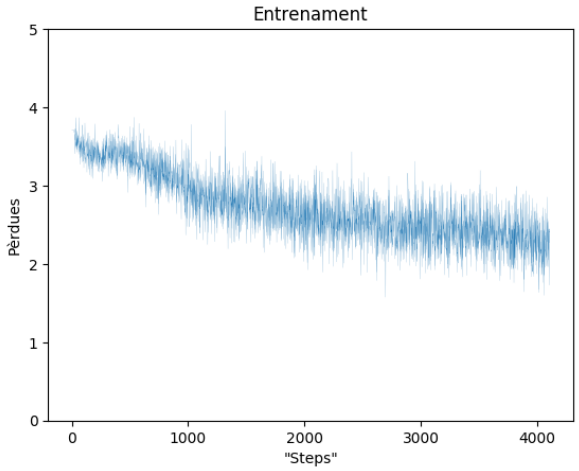


(d) Bandes = 32

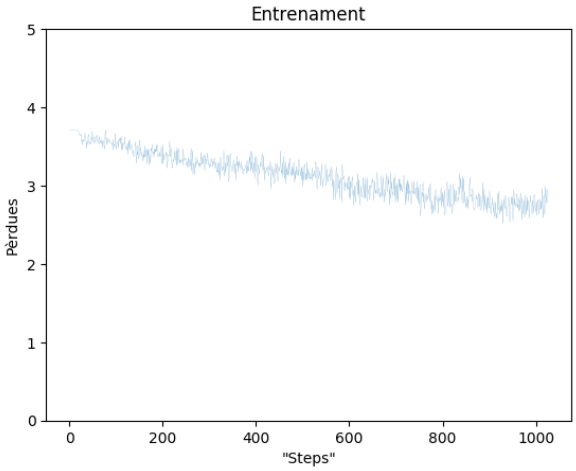


(e) Bandes = 128

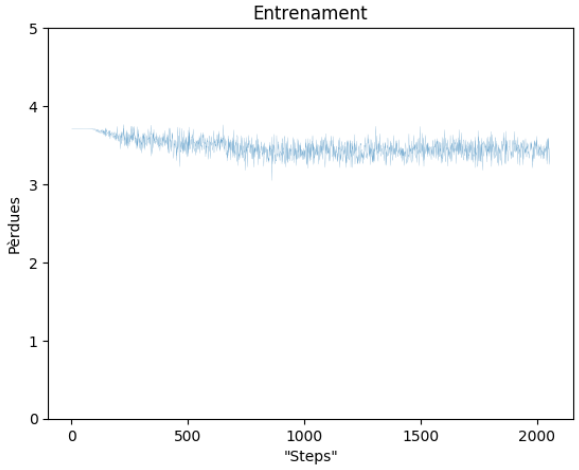
Figura 4.8: Evolució de les pèrdues d'entrenament per a 5 *epochs* i el 5% de les dades



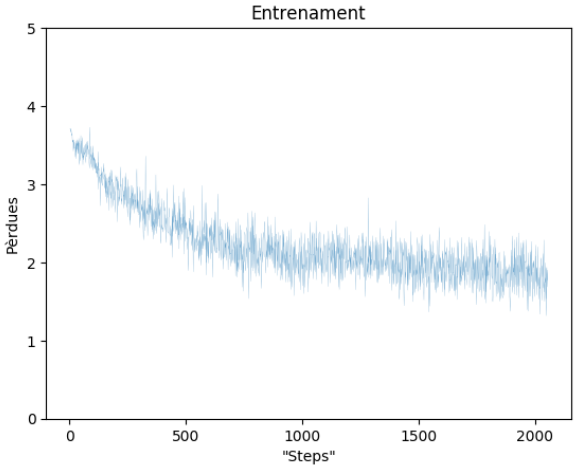
(f) Batch = 32



(g) Batch = 128



(h) Taxa d'aprenentatge = 1e-5



(i) Taxa d'aprenentatge = 1e-3

Continuació Figura 4.8: Evolució de les pèrdues d'entrenament per a 5 *epochs* i el 5% de les dades

Quantitat de bandes de Mel: Malgrat que a través de la comparativa de les figures 4.8d i 4.8e no es pot obtenir un resultat gaire clar, s'intueix com, a menor nombre de bandes, el sistema és més veloç, malgrat que, probablement, això comporti una pèrdua de precisió.

Tanmateix, augmentar el nombre de bandes a 128 implica un gran cost de temps donada la complexitat de càlcul que comporta. És molt probable que l'augment de bandes de Mel no sigui proporcional a l'augment de fiabilitat del sistema.

Quantitat d'instàncies a cada iteració (*batch size*): A l'inici d'aquest apartat es menciona que s'usen els 5 primers *epochs* d'entrenament, la qual cosa suposa que el sistema visualitza absolutament totes les dades 5 cops.

En augmentar la quantitat d'instàncies per a cada iteració, disminueix el total d'iteracions per *epoch*, és a dir, el total d'iteracions és menor, la qual cosa sempre suposa un nombre inferior de models generats.

Entre les figures 4.8f i 4.8g s'endevina una lleugera millora de les pèrdues a menor quantitat d'instàncies per iteració, probablement una conseqüència directa del superior nombre d'iteracions i que les dades es visualitzen amb més detall.

Taxa d'aprenentatge: Finalment i pel que fa a la taxa d'aprenentatge, la diferència entre les figures 4.8h i 4.8i és força notable. Podem determinar sense cap dubte que com major sigui la taxa d'aprenentatge obtindrem millors resultats.

Ara bé, s'ha de tenir present que una molt alta taxa d'aprenentatge pot arribar a ser contraproduent, com bé es detalla a la secció 4.2.

A partir d'aquesta breu comparativa entre els paràmetres que usa el sistema, podem concloure que la tendència per a optimitzar el sistema de referència de classificació d'àudio serà usar una finestra petita, el mínim de bandes de Mel que ens garanteix una precisió acceptable, una quantitat d'instàncies per iteració més aviat petita i una taxa d'aprenentatge alta malgrat que amb un cert límit.

4.4 Altres línies de treball

Al llarg d'aquest projecte, a part de fer funcionar el sistema de referència amb la base de dades proposada, també s'ha intentat elaborar una proposta de millora sòlida i una comparativa amb el sistema de referència de la tasca 2 del DCASE Challenge 2019.

Si bé és cert que no s'ha arribat a cap resultat fructífer en cap dels dos casos, pot ser rellevant a l'hora de continuar amb aquest projecte. És per això que es presenten els apartats següents.

Tanmateix, sintetitzar les dades generades pel sistema també ha suposat una certa càrrega en el projecte. Tots els gràfics referents als resultats obtinguts a través del sistema de referència, han estat generats a partir de scripts de Python. Es poden consultar a la documentació annexa.

4.4.1 Modificacions al sistema de referència de la tasca 2 del DCASE Challenge 2018

Després de posar en marxa el sistema tal com es mostra en l'apartat 4.1 i analitzar el funcionament del sistema (descrit a l'apartat 2.1), es van revisar altres propostes penjades a la tasca 2 del DCASE Challenge 2018 per part dels participants.

Donat que apareixia regularment i suposava una optimització del temps, es va considerar modificar la freqüència de mostratge dels fitxers per així reduir-ne les mostres totals i també el temps de processat. Això es va dur a terme tot elaborant un previ al sistema. No va presentar cap problema i funcionava correctament.

Ara bé, un cop finalitzat l'entrenament, els models desats no funcionaven correctament. En l'avaluació del sistema no es podien llegir els models generats i, per tant, no es podia efectuar cap predicció. La impressió és que, a l'hora de desar-los, no es guardava cap dada dintre dels fitxers.

És probable que *tensorflow* consti d'alguna limitació pel que fa a la freqüència de mostratge esperada que no es percep a simple vista. Es recomana seguir treballant amb aquesta hipòtesi.

Per a més informació, podeu consultar el codi elaborat al repositori de Github `https://github.com/lgotarra/Research-Audio-classification-using-Audioset-Freesound-Databases`.

4.4.2 Comparativa amb el sistema de referència de la tasca 2 del DCASE Challenge 2019

Un cop havent obtingut els resultats del funcionament del sistema de referència de la tasca 2 del DCASE Challenge 2018 es va valorar la possibilitat de comparar-ne els resultats amb els de la tasca 2 del DCASE Challenge 2019.

Malgrat que es tracta de tasques lleugerament diferents per les condicions inicials de la base de dades tal com es detalla a l'apartat 2, comparteixen el mateix objectiu i gairebé la mateixa estructura de funcionament.

Seria lògic doncs pensar que els requisits del sistema són els mateixos i, per tant, amb la mateixa configuració detallata anteriorment a la secció 4.1. Malauradament, en aquest cas si que es requereix de la versió de *tensorflow* 1.13, ja que se n'utilitzen algunes funcions disponibles a partir d'aquestes.

Com s'ha expressat amb anterioritat, posar en funcionament el sistema de referència de la tasca 2 del DCASE Challenge 2019 implicaria la necessitat de configurar *tensorflow* manualment dintre del servidor, del qual no en som superusuaris i, per tant, qualsevol instal·lació que haguem efectuar haurà de ser només accessible per al nostre compte.

Es tracta d'una problemàtica que suposaria destinar temps i recursos necessaris per finalitzar l'estudi, sense que això suposés un gran valor afegit per al projecte en si. Si hom amb més experiència i/o coneixements en l'àmbit ho considerés viable, es recomana seguir treballant segons la metodologia proposada anteriorment però tenint present les consideracions esmentada en aquest mateix apartat.

5 CONCLUSIONS

Al llarg d'aquest estudi, hem pogut conèixer el funcionament i la precisió d'un sistema en concret de classificació d'àudio. Ara bé, com queda patent a l'inici del projecte, la finalitat d'aquest no és només analitzar el sistema de referència de la tasca 2 del DCASE Challenge 2018, sinó també estudiar les possibilitats que ofereix i elaborar una guia extrapolable per aquells que busquin introduir-se en el món dels sistemes d'anàlisi de dades automatitzats amb Python.

Queda patent que per poder experimentar amb aquest tipus de sistemes és necessària una bona configuració de maquinària: és vertaderament l'element que s'enduu gran part del pressupost per a qualsevol projecte similar, sobretot si implica xarxes convolucionals i grans bases de dades. En aquest cas no es pot donar una xifra concreta, ja que no es té accés a la informació respecte a l'adquisició del servidor del departament de TSC de la UPC, malgrat que hom pot dissenyar una configuració al seu gust a partir de les pautes donades, i per tant el valor dependrà de cadascú.

D'altra banda, la dificultat d'aquest treball no només radica en entendre el codi, sinó també en saber analitzar els resultats obtinguts i com s'ha arribat fins aquest punt. És per això que els scripts[1] creats per a sintetitzar les dades de l'apartat 4.2 a vegades han arribat a un nivell de profunditat que pot resultar excessiu donada la complexitat per accedir a les dades.

Seguint la línia de complicacions, resulta evident que a la xarxa no hi abunda l'homogeneïtat pel que fa a la configuració de l'entorn del sistema. Això comporta que cada usuari que s'aventura a provar un sistema de classificació com el del DCASE Challenge difícilment trobarà els mateixos problemes que la resta. Continua sent un àmbit reservat per aquells que tenen nocions bàsiques de la configuració del seu entorn.

És per aquest motiu que la preparació per a poder posar en funcionament el sistema de referència guanya força importància en comparació a altres projectes de sistemes de classificació que es troben a la xarxa.

En relació als objectius proposats a la secció 1.3 d'aquest projecte, s'han acomplert tots, malgrat que no amb la mateixa profunditat esperada inicialment (visible a l'apartat 4.4). El capítol 1 seria l'equivalent a l'elaboració d'una breu guia introductòria als sitemes automatitzats de propòsit general i el capítol 4 és la guia a partir de la metodologia seguida per tal de poder posar en funcionament el sistema d'aprenentatge automàtic objecte d'estudi i configurar l'entorn necessari.

[1] Esmentats també a l'apartat 4.4

Com a resum a títol personal, considero que el desenvolupament d'aquest estudi m'ha resultat molt enriquidor. Ja no només m'ha permès eixamplar el meu coneixement en sistemes d'aprenentatge automàtic (dels quals se'n dóna una pinzellada al llarg del grau) sinó també la capacitat de cerca i resolució d'errors que han anat sorgint (directament o indirectament relacionats amb la temàtica del projecte).

5.1 Propostes i alternatives per a l'optimització

La primera proposta d'optimització que es presenta és l'esmentada a l'apartat 4.4.1. Si bé no s'ha pogut arribar a cap resultat per problemes tècnics, múltiples propostes presentades al DCASE Challenge 2018: Task 2 evidencien els seus beneficis.

La reducció de la freqüència de mostratge, tenint en compte la dimensió de la base de dades, suposa una reducció notable del temps d'execució de l'entrenament a canvi de perdre un cert grau d'exactitud de les dades, el qual no suposa grans pèrdues en la corba d'aprenentatge. Així doncs, presenta un balanç positiu per al sistema.

D'altra banda i una qüestió que ha suposat problemes regularment és la lectura de dades. El sistema de referència desa la informació de cadascun dels fitxers en memòria cada cop que s'executa algun dels passos (entrenament, avaluació o test).

Això suposa una càrrega excessiva que pot resultar en un desbordament i conseqüent finalització de la tasca, ja que sobrepassa el tamany del *buffer*[2].

Per aquest motiu es planteja la possibilitat de fer una lectura única de tota la base de dades i desar-la en fitxers que el sistema pugui consultar regularment.

Es recomana també seguir treballant a partir de les conclusions extretes de la secció 4.3 pel que fa a la modificació dels paràmetres per defecte del sistema.

Tanmateix, i tenint en compte que s'hi poden visualitzar tant els resultats com abstractes de tots els participants, és una bona opció en la cerca d'alternatives per a l'optimització donar un cop d'ull a les propostes d'altres investigadors penjades al DCASE Challenge 2018.

[2]Els primers cops que es van llançar els processos d'avaluació i test, donada l'alta càrrega del servidor, van ser finalitzats abans d'acabar.

ANNEXOS

Annex A: Diccionari de conceptes usats però no detallats a l'estudi

Teorema de Bayes

Es tracta d'un model probabilístic que expressa la probabilitat condicional d'un esdeveniment aleatori A segons B en termes de la distribució de probabilitat condicional de l'esdeveniment B segons A i la distribució de probabilitat marginal de només A.

Per exemple, donada la probabilitat que un pacient pateixi d'una certa malaltia per un símptoma en concret i la probabilitat de tenir aquesta malaltia, podem expressar la probabilitat que aquest pacient en concret tingui aquesta malaltia determinada.

$$P(b|a) = \frac{P(a|b)P(b)}{P(a)} \tag{5.1}$$

Sent $P(a|b)$ la relació causal entre la probabilitat que un pacient de la malaltia pateixi un dels seus símptomes, $P(b)$ la probabilitat de patir la malaltia i $P(a)$ la probabilitat de tenir qualsevol símptoma (independent de la malaltia).

Per a més informació podeu consultar la referència [17].

Entropia creuada

Es tracta de la relació entre dues distribucions properes o idèntiques entre sí. En sistemes de classificació amb models probabilístics, serveix per evaluar la distància entre el resultat real (i desitjat) i el model de prediccions obtingut.

$$c = \sum_{0}^{n} p_i \log\left(\frac{1}{q_i}\right) \tag{5.2}$$

On p_i és la distribució de probabilitat real i q_i la distribució de probabilitat estimada.

No es tracta d'una mesura de probabilitat i, per això, el sumatori dels seus valors pot ser diferent de 1. Com més petit sigui, més semblants seran les distribucions.

Per a més informació podeu consultar la referència [12].

Xarxes neuronals convolucionals

És un tipus de xarxa neuronal artificial (definida breument a l'apartat 1.1.3 dintre de la secció **d) Xarxes neuronals artificials**) on les neurones corresponen a camps receptius d'una manera molt similar a les neurones del còrtex visual primari d'un cervell biològic.

Es tracta d'un algorisme d'aprenentatge profund capaç, a partir d'una entrada determinada, d'assignar una certa importància a diversos aspectes de la mostra d'entrada per poder diferenciar-la d'una altra.

Perceptró multicapa

És la versió simplificada de les xarxes neuronals convolucionals, ja que s'aplica en una única dimensió. (Vegeu la definició de **Xares neuronals convolucionals** per a més informació).

Annex B: Scripts usats per a la gestió de dades generades pel sistema de referència i altres tasques relacionades amb l'ús del sistema

En aquest annex es troben tots els scripts usats al llarg del projecte, tant per a la gestió de dades resultants com per a usar el sistema com a tal.

La majoria de fitxers usen la sortida per pantalla del sistema en diverses situacions.

Script 1: Evolució de les pèrdues d'entrenament (*train_graph.py*)

Script usat per a generar els gràfics de les figures 4.3 i 4.8.

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

from matplotlib import pyplot as plt
import pandas as pd
import numpy as np
from tqdm import tqdm
import csv

x = []
y = []

with open('train.csv') as csv_file:
    csv_reader = csv.reader(csv_file, delimiter=',')
    for row in csv_reader:
        try:
                x.append(int(row[0]))
                y.append(float(row[1]))
        except:
                pass
csv_file.close()

plt.plot( x, y, linewidth=0.1)
plt.title(u"Entrenament")
plt.xlabel(u'"Steps"')
plt.ylabel(u'Perdues')
axes = plt.gca()
axes.set_ylim([0,5])
plt.savefig('train_evolve_3.png', bbox_inches='tight')
plt.clf()

x_down_100 = []
y_down_100 = []

for i in range(0,int(round(len(x)/500))):
        x_down_100.append(x[i*500])
```

```
37            y_down_100.append(y[i*500])
38
39  plt.plot(x_down_100,y_down_100,linewidth=1)
40  plt.title(u"Entrenament [Reduccio mostres: 500]")
41  plt.xlabel(u'"Steps"')
42  plt.ylabel(u'Perdues')
43  axes = plt.gca()
44  axes.set_ylim([0,5])
45  plt.savefig('train_evolve_down.png', bbox_inches='tight')
```

Script 2: Evolució mAP@3 dels models a l'avaluació (*eval_overall_map.py*)

Script usat per a generar el gràfic de la figura 4.4.

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

from matplotlib import pyplot as plt
import pandas as pd
import numpy as np
import statistics

f = open("eval.out","r")
fl = f.readlines()
num = open("models.csv","r")
numl = num.readlines()
models = [[0, 0] for i in range(len(numl))]
i = 0

for line in numl:
        string = line.replace("model.ckpt-","")
        models[i][0] = int(string)
        i+=1

i = 0
map = 0
cur_check = 0
prev_check = 0

for line in fl:
        if line.find("Overall␣MAP:␣") != -1:
                string = line.replace("Overall␣MAP:␣", '')
                map = float(string)

        elif line.find("Checkpoint:␣./train_dir/model.ckpt-") != -1:
                if cur_check == 0:
                        string = line.replace("Checkpoint:␣./train_dir
                            /model.ckpt-", '')
                        cur_check = int(string)
                else:
                        models[i] = [cur_check,map]
                        prev_check = cur_check
                        string = line.replace("Checkpoint:␣./train_dir
                            /model.ckpt-", '')
                        cur_check = int(string)
                        map = 0
                        i+=1

while i < len(models):
        if models[i][1] == 0:
```

```
45                      models.remove(models[i])
46              else:
47                      i+=1
48
49  models_s = sorted(models, key = lambda models: models[0])
50
51  plt.plot(*zip(*models_s))
52  plt.title(u"Evolucio avaluacio")
53  plt.xlabel('Model')
54  plt.ylabel('mAP global')
55  plt.savefig('eval_models.png', bbox_inches='tight')
56
57  plt.clf()
58
59  plt.plot(*zip(*models_s))
60  plt.title(u"Evolucio avaluacio")
61  plt.xlabel('Model')
62  plt.ylabel('mAP global')
63  axes = plt.gca()
64  axes.set_ylim([0,1])
65
66  plt.savefig('eval_models_0-1.png', bbox_inches='tight')
```

Script 3: mAP@3 per classe del millor model a l'avaluació (*eval_top_model.py*)

Script usat per a generar el gràfic de la figura 4.5.

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

from matplotlib import pyplot as plt
import pandas as pd
import numpy as np
import statistics

f = open("eval.out","r")
fl = f.readlines()
num = open("models.csv","r")
numl = num.readlines()
lab = open("class_map_2018.csv","r")
labl = lab.readlines()
models = [[0, 0] for i in range(len(numl))]
labels = [[0, 0, 0] for i in range(len(labl))]

i = 0

for line in labl:
        labels[i][0] = line
        i+=1

i = 0

for line in numl:
        string = line.replace("model.ckpt-","")
        models[i][0] = int(string)
        i+=1

i = 0
map = 0
cur_check = 0
prev_check = 0

for line in fl:
        if line.find("Overall␣MAP:␣") != -1:
                string = line.replace("Overall␣MAP:␣", '')
                map = float(string)

        elif line.find("Checkpoint:␣./train_dir/model.ckpt-") != -1:
                if cur_check == 0:
                        string = line.replace("Checkpoint:␣./train_dir
                            /model.ckpt-", '')
                        cur_check = int(string)
                else:
```

```
46                          models[i] = [cur_check,map]
47                          prev_check = cur_check
48                          string = line.replace("Checkpoint: ./train_dir
                                /model.ckpt-", '')
49                          cur_check = int(string)
50                          map = 0
51                          i+=1
52
53  while i < len(models):
54          if models[i][1] == 0:
55                  models.remove(models[i])
56          else:
57                  i+=1
58
59  models_s = sorted(models, key = lambda models: models[0])
60  maxpos = models_s.index(max(models_s, key = lambda models_s: models_s
        [1]))
61  minpos = models_s.index(min(models_s, key = lambda models_s: models_s
        [1]))
62  print models_s[minpos]
63  cur_check = 0
64  lab_map = []
65
66  for line in fl:
67          if line.find("Checkpoint: ./train_dir/model.ckpt-" + str(
                models_s[maxpos][0])) != -1:
68                  cur_check = models_s[maxpos][0]
69          elif line.find("Checkpoint: ./train_dir/model.ckpt-") != -1
                and cur_check != 0:
70                  break
71          if cur_check == models_s[maxpos][0]:
72                  if line.find("MAP for ") != -1:
73                          result = line[line.find("MAP for ")+8:line.
                                find(":")]
74                          string = line.replace("MAP for "+ str(result)
                                + ": ","")
75                          lab_map.append([result,float(string)])
76
77  #Get last appended
78
79  for j in range(len(lab_map)-1,0,-1):
80          index = 0
81          for i in range(len(labels)):
82                  string = lab_map[j][0] + '\n'
83                  if string == labels[i][0]:
84                          index = i
85                          break
86          if labels[index][1] == 0:
87                  labels[index][1] = lab_map[j][1]
```

```
88          labels[index][2] += 1
89
90  print labels
91  mostres = [item[2] for item in labels]
92  print min(mostres)
93  print max(mostres)
94  bars = [item[0] for item in labels]
95  height = [item[1] for item in labels]
96  y_pos = np.arange(len(bars))
97
98  colors = []
99  count = 0
100 for item in height:
101         if item == max(height):
102                 colors.append('lawngreen')
103         elif item == min(height):
104                 colors.append('darkred')
105         elif item >= models_s[maxpos][1]:
106                 colors.append('darkorange')
107                 count+=1
108         else:
109                 colors.append('cornflowerblue')
110 print count
111
112 plt.bar(y_pos, height, color=colors)
113 plt.title("mAP per etiquetes del model amb millors resultats: " + str(
        models_s[maxpos][0]))
114 plt.xlabel('Etiquetes')
115 plt.ylabel('mAP')
116 plt.xticks(y_pos, bars, rotation=90)
117 plt.subplots_adjust(bottom=0.4)
118 plt.axhline(y = models_s[maxpos][1],color='darkorange',linestyle='-',
        label = 'mAP global: '+  str(round(models_s[maxpos][1],3)))
119 plt.axhline(y = max(height),color='lawngreen',linestyle='-.', label =
        u'mAP maxim: ' + str(round(max(height),3)))
120 plt.axhline(y = min(height),color='darkred',linestyle='-.', label = u'
        mAP minim: ' + str(round(min(height),3)))
121
122 plt.legend(loc='lower right')
123 plt.show()
```

Script 4: Evolució mAP@3 dels models al test i comparació amb l'avaluació i mAP@3 per classe del millor model al test (*predictions.py*)

Script usat per a generar els gràfics de les figures 4.7 i 4.6.

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

from matplotlib import pyplot as plt
import pandas as pd
import numpy as np
import statistics
from os import listdir
from os.path import isfile, join

def column(matrix, i):
    return [row[i] for row in matrix]

resume_models = []
#for all files

onlyfiles = [f for f in listdir("./predictions") if isfile(join("./
    predictions", f))]

models = []

for item in onlyfiles:
        string = item[23:-4]
        models.append(int(string))

models.sort()
def_models = []

for item in models:

        f = open("./predictions/predictions_model.ckpt-" + str(item) +
            ".csv", "r")
        fl = f.readlines()
        lab = open("class_map_2018.csv","r")
        labl = lab.readlines()
        labels = [[0, 0, 0] for i in range(len(labl))]
        i = 0
        for line in labl:
                labels[i][0] = line[:-1]
                i+=1

        test = open("test.csv", "r")
        testl = test.readlines()
        testdata = []

```

```
44          for line in testl:
45                  string = line.split(',')
46                  testdata.append([string[0], string[1][:-2]])
47
48          testdata.pop(0)
49          fl.pop(0)
50          predictions = []
51          for line in fl:
52                  string = line.split(',')
53                  pred = string[1][:-2].split(' ')
54                  predictions.append([string[0], pred])
55
56          mapl = float(0)
57          count = 0
58          for i in range(len(labels)):
59                  for j in range(len(testdata)):
60                          if testdata[j][1] == labels[i][0]:
61                                  if predictions[j][1][0] == labels[i
                                        ][0]:
62                                          mapl += float(1)
63                                  elif predictions[j][1][1] == labels[i
                                        ][0]:
64                                          mapl += float(2/3)
65                                  elif predictions[j][1][2] == labels[i
                                        ][0]:
66                                          mapl += float(1/3)
67                                  count +=1
68                  labels[i][1] = mapl
69                  labels[i][2] = count
70                  mapl = 0
71                  count = 0
72
73          def_labels = []
74          recompte = []
75          for i in range(len(labels)):
76                  recompte.append([labels[i][0],labels[i][2]])
77                  def_labels.append([labels[i][0], float(labels[i][1])/
                        float(labels[i][2])])
78
79          def_models.append([item, statistics.mean(column(def_labels, 1)
                ), def_labels])
80
81  print recompte
82  maxpos = def_models.index(max(def_models, key = lambda def_models:
        def_models[1]))
83  minpos = def_models.index(min(def_models, key = lambda def_models:
        def_models[1]))
84
85  print "Max mAP per model: " + str(def_models[maxpos][1]) + " in model
```

```
    " + str(def_models[maxpos][0])
print "Min mAP per model: " + str(def_models[minpos][1]) + " in model
    " + str(def_models[minpos][0])

aux_lab = def_models[maxpos][2]
bars = [item[0] for item in aux_lab]
height = [item[1] for item in aux_lab]
y_pos = np.arange(len(bars))
colors = []

count = 0
for item in height:
        if item == max(height):
                colors.append('lawngreen')
        elif item == min(height):
                colors.append('darkred')
        elif item >= def_models[maxpos][1]:
                colors.append('darkorange')
                count+=1
        else:
                colors.append('cornflowerblue')
print count

plt.bar(y_pos, height, color=colors)
plt.title("mAP per etiquetes del model amb millors resultats: " + str(
    def_models[maxpos][0]))
plt.xlabel('Etiquetes')
plt.ylabel('mAP')
plt.xticks(y_pos, bars, rotation=90)
plt.subplots_adjust(bottom=0.4)
plt.axhline(y = def_models[maxpos][1],color='darkorange',linestyle='-'
    , label = 'mAP global: '+  str(round(def_models[maxpos][1],3)))
plt.axhline(y = max(height),color='lawngreen',linestyle='-.', label =
    u'mAP maxim: ' + str(round(max(height),3)))
plt.axhline(y = min(height),color='darkred',linestyle='-.', label = u'
    mAP minim: ' + str(round(min(height),3)))
plt.legend(loc='lower right')

#plt.show()
plt.clf()

plot_test = []
for item in def_models:
        plot_test.append([item[0], item[1]])

#models eval

f = open("eval.out","r")
fl = f.readlines()
```

```
num = open("models.csv","r")
numl = num.readlines()
models = [[0, 0] for i in range(len(numl))]
i = 0

for line in numl:
        string = line.replace("model.ckpt-","")
        models[i][0] = int(string)
        i+=1

i = 0
map = 0
cur_check = 0
prev_check = 0

for line in fl:
        if line.find("Overall␣MAP:␣") != -1:
                string = line.replace("Overall␣MAP:␣", '')
                map = float(string)

        elif line.find("Checkpoint:␣./train_dir/model.ckpt-") != -1:
                if cur_check == 0:
                        string = line.replace("Checkpoint:␣./train_dir
                            /model.ckpt-", '')
                        cur_check = int(string)
                else:
                        models[i] = [cur_check,map]
                        prev_check = cur_check
                        string = line.replace("Checkpoint:␣./train_dir
                            /model.ckpt-", '')
                        cur_check = int(string)
                        map = 0
                        i+=1

while i < len(models):
        if models[i][1] == 0:
                models.remove(models[i])
        else:
                i+=1

models_s = sorted(models, key = lambda models: models[0])

plt.plot(*zip(*models_s), label=u'Avaluacio')
plt.plot(*zip(*plot_test), label='Test')
plt.title(u"Evolucio␣mAPs␣avaluacio-test")
plt.xlabel('Model')
plt.ylabel('mAP␣global')
plt.legend()
plt.savefig('comp_eval_test.png', bbox_inches='tight')
```

```
176
177 plt.clf()
178
179 plt.plot(*zip(*plot_test))
180 plt.title(u"Evolucio test")
181 plt.xlabel('Model')
182 plt.ylabel('mAP global')
183 plt.savefig('test_models.png', bbox_inches='tight')
```

Script 5: Elaboració CSV amb la relació de les dades d'avaluació (*eval.py*)

Script usat per a generar un CSV amb un 10% de les dades d'entrenament que serveix de fitxer d'avaluació.

```
import csv, random
with open("train.csv", 'rb') as csvfile:

        eval = []
        for line in csvfile.readlines():
                eval.append(line)

eval.pop(0)
long = int(len(eval)*0.9)
random.shuffle(eval)

for i in range (0,long):
        eval.pop(0)

eval.insert(0,"fname,label,manually_verified␣\n")
f = open("eval.csv","w+")
for audio in eval:
        f.write(audio)
f.close()
```

Script 6: Elaboració directori amb els fitxers d'entrenament que s'usaran per a l'avaluació (*move_eval.py*)

Script usat per moure les dades descrites al CSV generat a través de l'script 5 del directori d'entrenament al d'avaluació.

```
import csv, os, shutil

with open("eval.csv",'rb')  as csvfile:
        eval=[]
        for line in csvfile.readlines():
                array = line.split(",")
                eval.append(array[0])

source = '/home/usuaris/veu4/laura.gotarra/dataset2018/audio_train/'
dest = '/home/usuaris/veu4/laura.gotarra/dataset2018/audio_eval/'

eval.pop(0)

for file in eval:
        shutil.move(source + file, dest)
```

Script 7: Llançament del procés d'entrenament contra el servidor (*train_script.sh*)

Script usat per llançar el procés d'entrenament del sistema contra el servidor del TSC.

```
#!/bin/bash

#SBATCH --job-name=train
#SBATCH -p veu
#SBATCH -c24
#SBATCH --mem=12GB

module load python
python main.py \
    --mode train \
    --model cnn \
    --class_map_path class_map_2018.csv \
    --train_clip_dir ../dataset2018/audio_train \
    --train_csv_path ../dataset2018/meta/train.csv \
    --train_dir train_dir/
```

Script 8: Llançament del procés d'avaluació per a tots els models generats contra el servidor (*eval_models.sh*)

Script usat per llançar el procés d'avaluació del sistema per a tots els models generats al llarg de l'entrenament contra el servidor del TSC.

```
#!/bin/bash
#SBATCH --job-name=eval
#SBATCH -o eval.out
#SBATCH -p veu
#SBATCH --mem=24G
#SBATCH -c24
#SBATCH --cpu-freq=3000000

cat models.csv | while IFS=, read -r col1
do
        python main.py \
                --mode eval \
                --model cnn \
                --class_map_path ./class_map_2018.csv \
                --eval_clip_dir ../dataset2018/audio_eval \
                --eval_csv_path ../dataset2018/meta/eval.csv \
                --checkpoint ./train_dir/$col1
done
```

Script 9: Llançament del procés de test per a tots els models generats contra el servidor (*test_models.sh*)

Script usat per llançar el procés de test del sistema per a tots els models generats al llarg de l'entrenament contra el servidor del TSC.

```
#!/bin/bash
#SBATCH --job-name=test
#SBATCH -o test.out
#SBATCH -p veu
#SBATCH --mem=24G
#SBATCH -c24
#SBATCH --cpu-freq=3000000

cat models.csv | while IFS=, read -r col1
do
        python main.py \
                --mode inference \
                --model cnn \
                --class_map_path ./class_map_2018.csv \
                --test_clip_dir ../dataset2018/audio_test \
                --checkpoint ./train_dir/$col1 \
                --predictions_csv_path ./predictions/predictions_$col1
                    .csv
done
```

Bibliografia

[1] *All symbols in TensorFlow | TensorFlow Core 1.13.* [https://www.tensorflow.org/api_docs/python].

[2] *Aprendizaje automático.* [https://es.wikipedia.org/w/index.php?title=Aprendizaje_autom%C3%A1tico&oldid=116395285].

[3] *AudioSet.* [https://research.google.com/audioset/].

[4] *DCASE2018 Challenge - DCASE.* [http://dcase.community/challenge2018/].

[5] *DCASE2019 Challenge - DCASE.* [http://dcase.community/challenge2019/].

[6] *Freesound General-Purpose Audio Tagging Challenge.* [https://kaggle.com/c/freesound-audio-tagging].

[7] *Machine Types | Compute Engine Documentation.* [https://cloud.google.com/compute/docs/machine-types].

[8] *AASP Home*, Sept. 2016. [https://signalprocessingsociety.org/get-involved/audio-and-acoustic-signal-processing/aasp-home].

[9] *Los Modelos Geometricos (Modelos parte I)*, Feb. 2017. [https://mlmexicanguy.wordpress.com/2017/02/09/los-modelos-geometricos-modelos-parte-i/].

[10] P. Almagro Blanco, D. Cabrera Mendieta, and F. Sancho Caparrini, *Aprendizaje Automático - Fernando Sancho Caparrini.* [http://www.cs.us.es/~fsancho/?p=machine-learning-aprendizaje-automatico].

[11] CVxTz, *Cnn 1d vs 2d audio classification.* [https://github.com/CVxTz/audio_classification].

[12] R. DiPietro, *A Friendly Introduction to Cross-Entropy Loss.* [https://rdipietro.github.io/friendly-intro-to-cross-entropy-loss/].

[13] E. Fonseca and F. Font Corbera, *DCASE 2019 Baseline system task 2.* [https://github.com/DCASE-REPO/dcase2019_task2_baseline].

[14] F. Font Corbera, E. Fonseca, D. P. W. Ellis, and M. Plakal, *DCASE 2018 Baseline system task 2.* [https://github.com/DCASE-REPO/dcase2018_baseline].

[15] A. KARPATHY, *CS231n Convolutional Neural Networks for Visual Recognition.* [http://cs231n.github.io/neural-networks-3/].

[16] M. NK, *K-Means Clustering in Python - Blog by Mubaris NK.* [https://mubaris.com/posts/kmeans-clustering/].

[17] J. L. R. REINA, *Aprendizaje de modelos probabilísticos.* [https://www.cs.us.es/cursos/ia2/temas/tema-04.pdf].

[18] F. SANCHO CAPARRINI, *Introducción al Aprendizaje Automático - Fernando Sancho Caparrini.* [http://www.cs.us.es/~fsancho/?e=75].

[19] S. TOLOTRA, *What is the difference between Step, Batch Size, Epoch, Iteration ? Machine Learning Terminology*, July 2018. [https://tolotra.com/2018/07/25/what-is-the-difference-between-step-batch-size-epoch-iteration-machine-learning-terminology/].

[20] H. ZULKIFLI, *Understanding Learning Rates and How It Improves Performance in Deep Learning*, Jan. 2018. [https://towardsdatascience.com/understanding-learning-rates-and-how-it-improves-performance-in-deep-learning-d0d4059c1c10].