Escola Tècnica Superior d'Enginyeria Electrònica i Informàtica La Salle

Treball Final de Màster

Màster Universitari en Data Science

Deep Learning aplicat al reconeixement d'esdeveniments acústics

Alumne

Xavier Armenteras Bosom

Professor Ponent

Francesc Alías Pujol

Professor Coponent

Alejandro González Alzate

ACTA DE L'EXAMEN
DEL TREBALL FI DE CARRERA

Reunit el Tribunal qualificador en el dia de la data, l'alumne

D. Xavier Armenteras Bosom

va exposar el seu Treball de Fi de Carrera, el qual va tractar sobre el tema següent:

Deep Learning aplicat al reconeixement d'esdeveniments acústics

Acabada l'exposició i contestades per part de l'alumne les objeccions formulades pels Srs. membres del tribunal, aquest valorà l'esmentat Treball amb la qualificació de

Barcelona,

VOCAL DEL TRIBUNAL VOCAL DEL TRIBUNAL

PRESIDENT DEL TRIBUNAL

Resum

El *Deep Learning* és un conjunt d'eines d'aprenentatge automàtic que en els últims anys ha revolucionat el món gràcies als bons resultats que ofereix a l'hora de modelitzar dades complexes. Avui en dia el *Deep Learning* és un camp punter de la ciència i l'augment d'escenaris d'aplicació d'aquest tipus d'aprenentatge és part de la revolució que ha generat aquesta tecnologia.

Buscant solucions per al que s'anomena com a *SmartCities,* sorgeix la necessitat d'entendre l'entorn que ens envolta basant-se en els sons propis de la ciutat; en aquest projecte s'ha buscat aquesta comprensió de l'entorn fent ús del *Deep Learning* (p.ex. xarxes neuronals convolucionals (CNN) i xarxes neuronals recurrents (RNN)).

En el projecte s'ha treballat amb dades reals recollides en el marc d'un projecte europeu que té per objectiu la monitorització acústica del soroll de trànsit de la ciutat. L'objectiu passa per aconseguir sistemes de reconeixement d'esdeveniments acústics més robustos i / o precisos.

Índex

1. Contextualització del projecte

1.2 Formulació del problema

Degut al nivell d'eficiència que estan adquirint els sistemes de transmissió, emmagatzematge i tractament de dades, estan sorgint moltes aplicacions on es tracta amb una immensitat de dades que fa uns anys eren problemes impensables d'abordar.

Concretament, el problema tractat en aquest projecte entre en l'àmbit de les *Smart Cities,* on s'està aconseguint moltes coses però també queda molt per fer.

En el Grup de recerca en Tecnologies Mèdia (GTM) de La Salle s'està treballant en un projecte europeu anomenat DYNAMAP per tal de classificar esdeveniments sonors en la ciutat, per així aïllar el soroll de trànsit i poder detectar la intensitat d'aquest. Fins ara s'han utilitzat mètodes classificadors més tradicionals, però s'ha detectat que la comunitat universitària esta obtenint molt bons resultats en aquesta disciplina utilitzant *Deep Learning.* Per aquesta raó el repte que s'afronta amb aquest treball és crear un model de xarxes neuronals artificials i comparar el seu rendiment amb els models actuals del equip de La Salle.

Pel que fa a les dades amb les que es tractarà són àudios del projecte *LIFE DYNAMAP* (LIFE DYNAMAP, 2018) mostrejats a diferents punts de Milà i Roma. Aquest àudios estan etiquetats, per tant es tracte amb un problema d'aprenentatge supervisat.

El projecte *LIFE DYNAMAP* està destinat a desenvolupar un sistema dinàmic de cartografia de soroll capaç de detectar i representar en temps real l'impacte acústic de la infraestructura viària. Amb aquesta finalitat, s'ha desenvolupat i construït un sistema de monitoratge automàtic, basat en sensors de baix cost personalitzats i una eina de programari implementada en una plataforma GIS d'ús general, tot això per a dues zones pilot situades a l'autopista A90 que envolta la ciutat de Roma i a dins de la ciutat de Milà.

Els àudios estan classificats en dues classes: "Soroll de trànsit" i "Esdeveniments sonors anòmals". Això fa que el problema sigui de classificació binària. Això és degut a que el projecte del *DYNAMAP* el què vol es establir un mapa amb el nivell del soroll de tràfic i per tant amb una classificació binària de tràfic o no tràfic es suficient.

1.3 Objectius del projecte

En aquest treball es defineix un objectiu molt clar, que és crear un model de classificació d'esdeveniments sonors per al data set del *DYNAMAP* amb xarxes neuronals artificials i comparar el seu rendiment amb els models actuals creats per el grup de recerca de La Salle.

Per fer això s'ha d'elaborar tot un procés d'extracció de característiques dels àudios, fer una recerca en la literatura per veure quin tipus de models de xarxes neuronals artificials s'estan utilitzant en aquest camp i finalment crear els models que es cregui adients. Un cop creats, comparar-los entre ells i per acabar, comparar el millor model creat amb els models del GTM de La Salle.

Fent l'esmentat en l'anterior paràgraf, sempre i quant el model final sigui elaborat correctament, d'acord amb els models vistos a la literatura i es faci una bona comparació els models del GTM, es compliria objectiu i es donaria per satisfet el treball realitzat en el TFM. Òbviament un objectiu que pot ser s'aconsegueixi complir o pot ser que no, es superar el rendiment del millor model actual i per tant obtenir un millor classificador per a aquest problema.

1.4 Estat actual del problema

En aquest apartat es descriuen dues qüestions. Per una banda l'estat actual del problema en el GTM de La Salle, i per altra banda com l'estat de la qüestió en l'àmbit de la recerca sobre identificació automàtica d'esdeveniments acústics. Concretament hem utilitzat els *papers* del *DCASE Challenege 2017* (DCASE2017, 2018) com a referència, ja que es un *challenge* dels més importants en aquest àmbit i planteja un problema molt similar al nostre.

1.4.1 Estat actual del problema en el grup de recerca

Actualment en l'equip de La Salle s'està treballant amb els anomenats *Mel Frequency Cepstral Coefficients (MFCC)* els quals són utilitzades en gran part de les aplicacions que tenen a veure amb reconeixement de so. (Shikha Gupta, 2013)

L'àudio del que es disposa consta de 9 hores i 8 minuts de so etiquetat, gravat en entorn urbà (Milà) i suburbà (Roma).

En l'àrea de Roma consta de 16492 segons de so de tràfic i 543 segons d'altres sons. Per altra banda a Milà hi ha 11600 i 1932 segons dels sons prèviament anomenats.

Per fer aquesta extracció s'utilitza una finestra *Hamming* de 30 ms de llargada amb intercalació del 50% i un cop fet això s'extreuen 13 característiques mel dins del rang de freqüències de 0-24000Khz .

Amb les característiques ja extretes s'elaboren varis classificadors, els millors dels quals resulten ser un KNN i un SVM, tant en l'entorn de Milà com en el de Roma, tot i que en aquest segon als models els hi costa més fer la classificació. (Joan Claudi Socoró, 2017)

Finalment l'avaluació es fa amb el F1-score macro, ja que és una mètrica que encara que tinguem el *dataset* desbalancejat segueix sent representativa. Els millors resultats son un 74.43% per Roma i un 81.44% per Milà.

1.4.2 Estat actual del problema en les propostes per al *DCASE Challenge 2017*

En aquest apartat es farà una breu revisió dels models proposats en de les solucions a la Taks3 del DCASE CHALLENGE 2017. Primer de tot veiem en què consisteix aquest repte:

Les dades són àudios gravats al carrer de durades de 3 a 5 minuts, gravant a 44.1 kHz i 24 bits de resolució. En les quals tenim el "resultat" de cada instant, és a dir el problema es d'aprenentatge supervisat.

El problema consisteix en multi-classificar(en un instant poden haver-hi n solucions) els següents esdeveniments sonors:

- Frens de cotxes "xirigant"
- Cotxe
- Nens
- Vehicles pesats
- Gent parlant
- Gent caminant

Com podem veure aquesta tasca es bastant similar a la que es planteja des de l'equip de recerca de La Salle.

Fixant-se tant en l'estructura dels models creats com en els resultats que donen i pensant què pot aportar cada tipus de capa de les xarxes per a la resolució del problema apareixen dos models molt interessants.

Ambdós models consten de capes convolucionals que serviran per veure patrons entre bandes de freqüència *mel* i instants de temps propers i de capes recurrents que tracten el so com a sèrie temporal que és.

Així doncs el primer model (Adavanne, 2017) combina una sèrie de blocs convolucionals amb uns altres de recurrents de manera seqüencial.

Es pot veure en la **figura 1** que cada bloc convolucional esta format per la capa convolucional en si, una activació *ReLu* i un *MaxPool,* finalment, després de cada bloc també hi ha una capa de *drop-out* per evitar *l'overfitting* i fer les xarxes més robustes a dades no vistes.

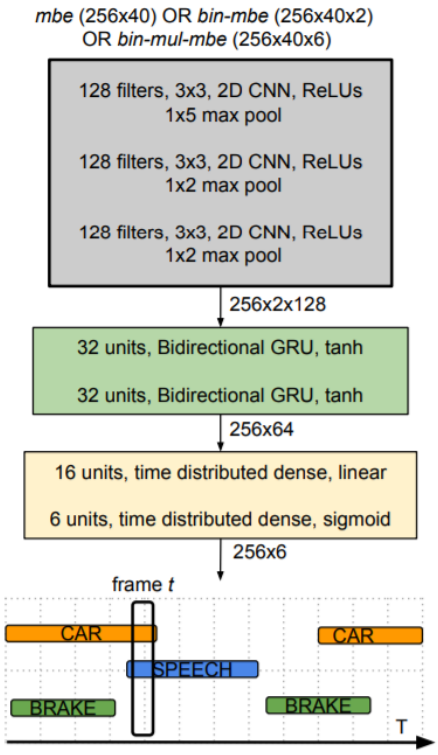


Figura 1: CNN I RNN apilades (Adavanne, 2017)

Després de la part convolucional de la xarxa les dades son passades a la part recurrent de la xarxa.

Aquesta part està composta per dues capes bidireccionals de *Gated Recurrent Unit (GRU)* per aprendre patrons distribuïts en períodes de temps.

Finalment veiem dues capes *time distributed dense*, on s'acaba d'integrar la informació dels 64 valors que surten per a cada unitat de temps del bloc recurrent per donar una sortida de la xarxa de 256x6, on gràcies a la *sigmoid activation* de la última capa, per cada unitat de temps i cada classe de les 6 possibles s'obté una probabilitat entre 0 i 1.

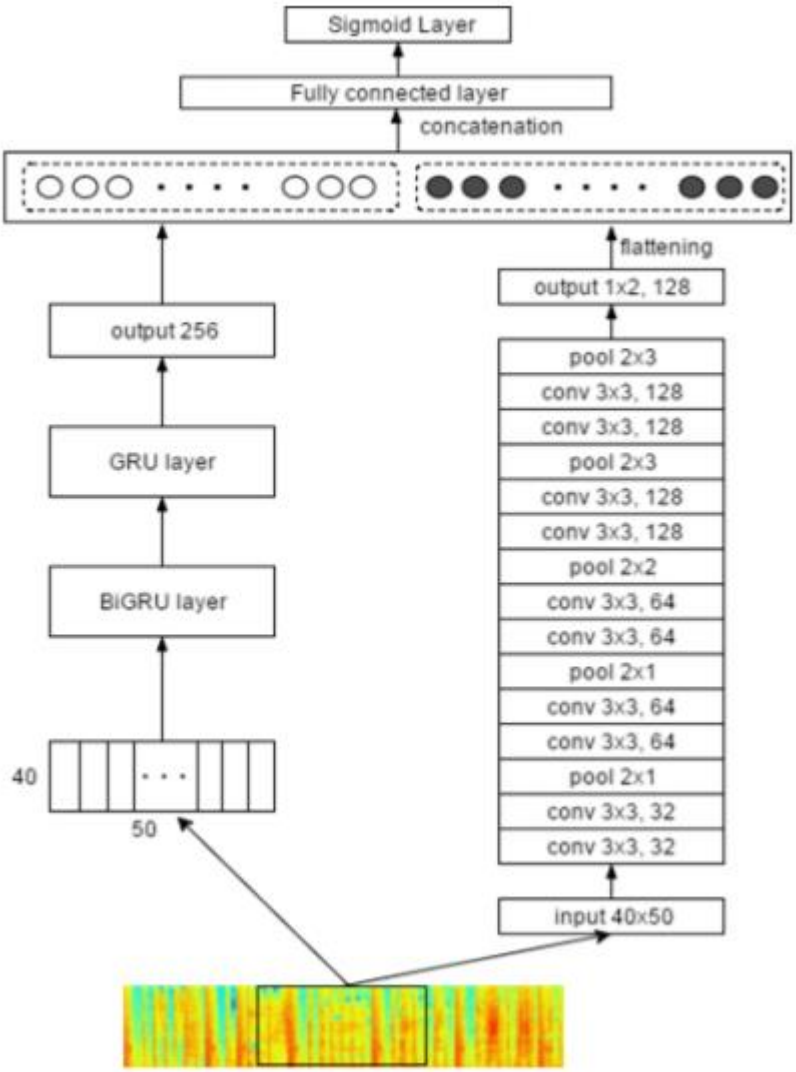


Figura 2: CNN i RNN paraleles (Dang, 2017)

Per altre banda el model utilitzat en (Dang, 2017) també integra una part recurrent i una convolucional, però en aquest cas de forma paral·lela. Les dades d'entrada de la xarxa entren tan a una part convolucional formada per blocs com els anteriors i també a una part recurrent, formada per dues capes de *GRUs* la primera de les quals bidireccional i amb 512 neurones i la segona amb 256 neurones.

Finalment els *outputs* d'ambdós blocs son concatenats i passats a dues capes denses, de 128 neurones la primera i de 6 amb activació *sigmoid* la segona per donar el resultat del problema.

Com hem comentat, tant per resultats com per estructura aquests son els dos models que han servit de referents per a l'elaboració d'aquest TFM, però en el *challenge* també hi havia altre tipus de models de *Deep Learning*, com per exemple MLP, CNN, RNN(tant amb GRU com amb LSTM) i fins i tot hi ha un article que proposa un model amb un *deep gradient boosted trees.*

1.6 Recursos

Per tal de dur a terme aquest projecte s'ha utilitzat un seguit d'eines, algunes de les quals ja han anat sortint al llarg dels punts anteriors però que ara s'explicaran i es justificaran:

1.6.1 *Python*

És un dels llenguatges de programació més utilitzats en *Deep Learning* degut a la seva potència de càlcul i a la gran quantitat de llibreries existents per la creació de models i bàsicament fer quasi bé qualsevol cosa. (Python Software Foundation, 2018)

En el cas d'aquest projecte, s'han utilitzat les llibreries *Keras* i *Scikit-learn* que s'expliquen a continuació.

1.6.2 *Keras*

Keras es defineix com una llibreria d'alt nivell per a la creació de xarxes neuronals artificials. Aquesta llibreria esta implementada com a una capa per sobre de *TensorFlow o Theano*, dues de les millors eines de cara a la creació de xarxes neuronals artificials. *Keras* és compatible amb *python* 2.7-3.6.

Dos aspectes importants per al projecte de *Keras* són que es capaç d'entrenar els seus models en GPU, fet molt important de cara a l'eficiència en el testejat dels models ja que ens redueix molt el temps d'entrenament. L'altre aspecte és que permet la creació de capes convolucionals i recurrents, les quals tenim previst utilitzar en els nostres models. (Keras, 2018)

1.6.3 Scikit-learn

Llibreria nascuda al 2017 de la mà de David Cournapeau. Conté funcions per a gairebé qualsevol tasca que es vulgui utilitzar de Data Science. En el projecte s'han utilitzat bàsicament les mètriques que proporciona per avaluar els models creats. (Pedregosa, 2011)

1.6.4 NumPy

Llibreria que permet el tractament de grans quantitats de dades de manera eficient en Python. (NumPy, 2018)

1.6.5 Librosa

Aquesta llibreria permet des de *python* llegir arxius d'àudio y processar-los per tal de poder utilitzar aquestes dades per fer una modelització. (Librosa, 2018)

1.6.3 Google Colab Research

També anomenat C*olaboratory,* aquest es un projecte de Google creat per facilitar la divulgació i la creació de contingut sobre aprenentatge automàtic. És un entorn basat en els coneguts *Jupiter Notebooks* que no requereix configuració i que s'executa en el núvol.

Des de els quaderns de *Colaboratory* es pot accedir a l'arbre de directoris de Google Drive i es guarden com un fitxer més d'aquesta unitat.

Cal destacar que aquesta és l'única eina gratuïta que hem trobat que ens permeti executar el nostre codi sobre les GPU, cosa que, com ja s'ha comentat, fa que la velocitat d'execució dels models sigui molt més ràpida que en la CPU. (Colaboratory, 2018)

2. Estat de la qüestió

2.1 Extracció d'Audio Features

Per tal 'extreure les característiques adients dels àudios a la literatura es segueix un procés que està bastant estandarditzat. (http://haythamfayek.com/2016/04/21/speech-processing-for-machine-learning.html, 2018)

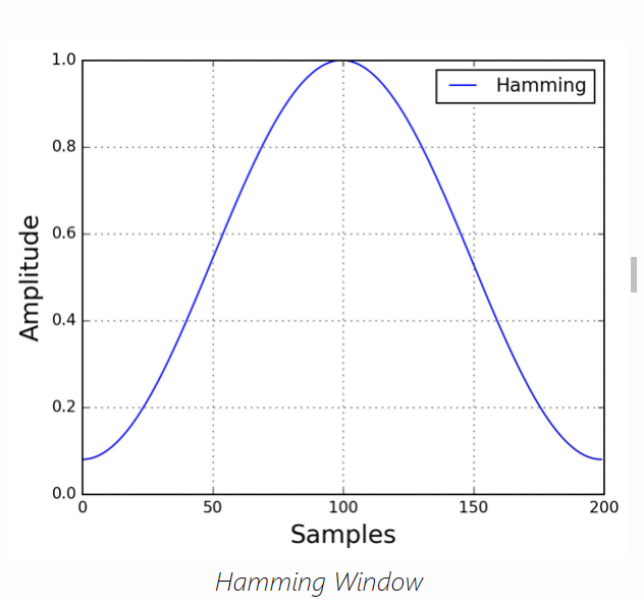
2.1.1 Framing

El primer de tot es dividir la senyal en instants de temps curts, on per a cada instant s'obtindrà un conjunt de característiques. Aquest pas es fa perquè la freqüència del senyal varia amb el temps, i per tant no té sentit aplicar una transformada en períodes llargs ja que es perdria molta informació. Normalment, aquests períodes van de 20 a 40 ms, amb una intercalació del 40% al 60%.

2.1.2 Window

El següent pas és aplicar una finestra al senyal. Típicament es fa servir la finestra de *Hamming*, que té la següent equació:

$$w[n] = 0.54 - 0.46cos(\frac{2\pi n}{N-1})$$



Hamming Window

Figura 3: Equació i plot de la finestra de *Hamming*

Aquest pas es realitza entre altres coses per tal de contrarestar la suposició feta per el FFT de que les dades son infinites i per reduir la freqüència espectral.

2.1.3 Fourier-Transform

Un cop a aquest punt es calcula l'espectre de la freqüència fent una *N-point FFT*, com que es calcula per a cada període de temps dels segmentats anteriorment, a aquesta transformada també se l'hi anomena *Short-Time Fourier Transform*. Normalment N = {256,512} i es calcula l'espectre de potència amb la següent equació:

$$P = \frac{|FFT(x_i)|^2}{N}$$

Figura 4: Equació STFT

2.1.4 Filter Banks

El pas final per calcular les característiques és aplicar filtres triangulars, típicament 40, en una escala Mel al espectre de potència per extreure'n bandes. L'escala Mel té com a objectiu imitar la percepció no lineal del oïda humana, que és més discriminat a les freqüències mes baixes. Per convertir entre Mel i Hertz s'utilitzen les equacions següents:

$$m = 2595 \log_{10}(1 + \frac{f}{700})$$

$$f = 700(10^{m/2595} - 1)$$

Figura 5: Transformacions Mel-Hertz

Cada filtre del banc es triangular, amb una resposta de 1 al centre que va disminuint linealment fins que arriba a 0 al límit del triangle, però on ja ha arribat al triangle adjacent.

La funció que modelitza aquest comportament és la que veiem a la Figura 6.

$$H_m(k) = \begin{cases} 0 & k < f(m-1) \\ \dfrac{k - f(m-1)}{f(m) - f(m-1)} & f(m-1) \leq k < f(m) \\ 1 & k = f(m) \\ \dfrac{f(m+1) - k}{f(m+1) - f(m)} & f(m) < k \leq f(m+1) \\ 0 & k > f(m-1) \end{cases}$$

Figura 6: Funció extracció bandes de freqüència MEL

2.2 Anàlisis de dades o Data Science

Data Science és un camp multidisciplinari que es recolza en les ciències de la computació i l'estadística, que te per objectiu estudiar el comportament de les dades i intentar extreure informació que aporti valor a partir d'aquestes, per dur a terme això s'utilitzen tècniques d'Intel·ligència Artificial, aprenentatge automàtic i estadística.

2.2.1 Procés

En la majoria de processos de Data Science, es poden identificar la majoria dels punts següents:

1. Selecció del conjunt de dades.
2. Anàlisis de les propietats de les dades: localització de *null vàlues* i càlcul de dispersions i mitjanes entre altres.
3. Pre-procés: tractament dels possibles dèficits trobats en el pas anterior a la base de dades, després d'aquest punt el *data set* ha d'estar apunt per aplicar la tècnica d'anàlisi desitjada.
4. Selecció de la tècnica a utilitzar: com es veurà en el pròxim apartat hi ha moltes tècniques de DS, s'han d'analitzar i seleccionar les que més s'adeqüi al propòsit de l'estudi.
5. Aplicació de la tècnica i extracció de coneixement.

6. Interpretació i avaluació dels resultat: és la conclusió i el propòsit de tots els passos anteriors i el que decidirà si el projecte ha tingut o no una utilitat real.
7. Producció de coneixement: un cop interpretats els resultat, aquests han de ser transmesos de la manera adequada per tal d'aportar coneixement a l'àrea implicada.

2.2.2 Algorismes no supervisats

Aquests algorismes fan un estudi de les dades per veure patrons i tendències en aquestes. Algunes de les tècniques més utilitzades són:

1. Regressió lineal (John Neter, 2018): Una de les més senzilles i eficaces a l'hora de veure la relació entre un conjunts de dades. El fet de ser senzilla fa que pugui ser ineficaç en casos complexos.
2. *Clustering* (Berkhin, 2006)*:* Hi ha molts algorismes de *clustering*, tots amb un mateix objectiu: agrupar les dades amb conjunts similars d'aquestes.

2.2.3 Algorismes supervisats

Es tracta d'un conjunt de tècniques de mineria de dades que estan enfocades a utilitzar un conjunt de dades històric per tal de fer una predicció sobre un nou individu.

Aquestes tècniques també són anomenades tècniques d'Aprenentatge o *Machine Learning*, algunes de les més conegudes són:

1. Arbres de decisió (S.R. Safavian, 2018): Consisteix en una sèrie de decisions que van portant l'algorisme per un camí o per un altre fins a arribar a una predicció.
2. *Support Vector Machine* (Vapnik, 2018): Aquesta tècnica intenta separar l'espai on hi ha les dades i predir la nova a partir de la seva posició a l'espai.
3. Xarxes Neuronals Artificials (Rosenblatt, 2018): Tècnica inspirada en el cervell dels essers humans. Es tracta d'una gran quantitat de processos anomenats neurones que poden treballar en paral·lel per processar *datasets* molt grans.

2.2.4 Xarxes neuronals Artificials

En aquest apartat es farà una menció especial a les xarxes neuronals, a la seva distribució, al seu funcionament i a les seves aplicacions ja que ha estat el model aplicat en el projecte.

Les xarxes neuronals són xarxes interconnectades massivament en paral·lel d'elements simples (neurones) i amb una organització jeràrquica. La xarxa esta construïda per tres parts. Un capa d'entrada, un conjunt de capes ocultes o intermèdies i una capa de sortida.

Distribució

Com ja s'ha dit la distribució de les neurones està feta en diferents capes:

- Capa d'entrada: és la capa que rep directament la informació provinent de les variables que alimenten a la xarxa. Conté una neurona per cada variable d'entrada.

- Capes ocultes o intermèdies: Són les internes a la xarxa, que reben informació de neurones i la seva sortida alimenta neurones, així que no tenen contacte amb l'exterior de la xarxa. El tipus de capa, nombre de capes ocultes, el nombre de neurones de cada capa i les connexions entre aquestes es defineixen a cada xarxa.

- De sortida: Transfereixen el resultat de la xarxa cap a l'exterior, són les que donen el resultat d'aquesta.

Funcionament

El funcionament de les xarxes neuronals és basa en tres funcions de les neurones i l'estructura en la que aquestes estan distribuïdes:

- Funció d'entrada: Cada neurona ha de tractar un conjunt de valors com si fos un, així que és necessària una funció que pugui combinar totes les entrades simples. Algunes de les funcions d'entrada més utilitzades són:

    - Sumatòria amb pesos: suma tots els valors d'entrada de la neurona multiplicats pel seu pes.

    - Multiplicatòria amb pesos: multiplica tots els valors d'entrada de la neurona multiplicats pel seu pes.

    - Màxim amb pesos: agafa el valor d'entrada que multiplicat pel seu pes és el més gran.

- Funció d'activació: Aquesta funció transforma l'entrada obtinguda per la funció d'entrada de la neurona en un valor, normalment entre 0 i 1 o entre -1 i 1. Algunes funcions d'activació són la *funció lineal*, la *funció sigmoide* o la *funció tangent hiperbòlica*.

- Funció de sortida: aquesta funció és l'encarregada de retornar el valor que serà el valor de sortida de la neurona i per tant aquesta funció també determina el valor que la neurona en qüestió transmet a les neurones vinculades a aquesta. Els valors que pot retornar aquesta funció normalment ment estan compresos en els rangs [0,1] o [-1,1], tot i que també poden ser valors binaris {0,1} o {-1,1}. Les dues funcions de sortida més comuns són:

    - Identitat: és la més senzilla i simplement no fa res.

    - Binària: Retorna 1 si el valor és més gran que un cert límit, altrament retorna 0.

Un cop definides les funcions i la distribució de neurones en la xarxa cal entrenar la xarxa per tal d'adaptar els pesos que hi ha entre les neurones. Aquest procés s'anomena procés d'aprenentatge i durant aquest els pesos en les connexions de les neurones van variant. Aquest procés és dona per complet quant els pesos en totes les connexions s'estabilitzen.

Capes d'una xarxa

Com ja s'ha vist en l'apartat on es detalla el software utilitzat, a l'hora de crear els models de xarxes neuronals per fer la classificació d'esdeveniments sonors s'ha utilitzat Keras per Python. Els models creats estan fets a partir de varies capes que podríem separar en capes d'extracció d'informació i en capes auxiliars les quals tenen funcions molt diverses.

Capes d'extracció d'informació

Aquí podrem trobar tres capes principals que, al ser les capes bàsiques de una xarxa, són les que donen nom als tres tipus de xarxa neuronal artificial més coneguts, la DNN o MLP(*dense neural network o multi layer perception*), CNN *(convolutional neural network)* i RNN *(recurrent neural network).*

Dense

Segurament la capa més coneguda de les xarxes neuronals i capa base de les MLP. Aquesta capa te *n* entrades i *m* sortides connectades totes amb totes i en cada connexió s'aplica una funció lineal.

Normalment, en les *CNN* i les *RNN*, aquest tipus de capa s'inclou al final, per acabar d'integrar la informació de les capes prèvies i treure una predicció a partir d'una de les funcions d'activació següents:

- **Softmax**: utilitzada per prediccions de *k* classes amb un sol possible resultat.
- **Sigmoid**: utilitzada per prediccions de *k* classes amb més d'un possible resultat. Aquesta es la que utilitzem en els nostres models, ja que hem de poder identificar si en un instant hi ha més d'un esdeveniment sonor.
- **Euclidian**: utilitzada per problemes de regressió.

Convolutional

Aquest tipus de capa esta inspirada en els estímuls causats per una neurona visual.

L'input d'una capa convolucional és un vector de n-dimensions i el que fa aquesta capa és aplicar una sèrie de *m* filtres sobre aquest vector. Mitjançant aquests filtres s'aconsegueixen dues coses. La primera és que permet a la xarxa aprendre informació relativa en un espai concret del vector i la segona reduir el número de neurones necessàries per processar un vector de n-dimensions en comparació amb una capa *dense*.

Finalment, l'output d'aquesta capa es un vector de les mateixes dimensions que el d'entrada amb algun valor menys ja que al principi i al final el filtre pot eliminar alguna dada depenent de la implementació i amb *m* canals, un per cada filtre aplicat. (Convolutional Networks, 2018)

Recurrent

Les capes recurrents, son parts de la xarxa que tenen memòria, i que l'utilitzen per tal d'utilitzar entrades anteriors per a calcular *l'output* actual. Això fa que siguin capes ideals per problemes amb dades que tenen una component temporal, com per exemple reconeixement de so, o processament de llenguatge natural, entre molts altres.

En aquest treball es tracta amb dos tipus de capes recurrents:

- **LSTM**: Les capes *Long-Short Term Memory*, tal i com indica el seu nom, son capes capaces de mantenir en memòria durant un temps de llarga durada esdeveniments de curta durada. Això fa d'aquestes capes un element molt bo a l'hora de predir sèries temporals on els esdeveniments importants per a la classificació tenen espais de durades diferents entre ells.
- **GRU**: Les capes *Gated recurrent unit*, la *performance* de les quals s'ha vist que és semblant a les LSTM a l'hora de reconèixer sons o parla. Tot i això, sembla que donen més bons resultats que les LSTM en *datasets* no molt grans.

Capes auxiliars

Aquestes capes serveixen bàsicament per a aplicar diferents funcions a les dades en el seu recorregut dins la xarxa.

Pooling

Aquest és un tipus de capa que sol estar inclòs en les xarxes convolucionals. Com hem dit anteriorment una capa convolucional aplica *m* filtres a un vector de *n*-dimensions. Els *m* filtres aplicats en una mateixa posició donen *m* canals del vector d'entrada. Per ajuntar aquests canals es sol utilitzar una capa de *pooling*, que el què fa és ajuntar-los en un. Alguns *pooling* que es fan servir son el *max-pooling* que es queda amb el valor màxim o el *average-pooling* que fa la mitjana.

En resum el què fa aquesta capa és aplicar un filtre de mida *a x b* a les dades per tal de reduir les dimensions d'aquestes.

Rectified Linear Units

Coneguda com a capa *ReLU*, aquesta unitat aplica una funció d'activació que incrementa les propietats no lineals de la funció de decisió. Existeixen altres funcions d'activació, però degut a l'eficiència computacional d'aquesta i els bons resultats que ofereix, la *ReLU* és la que s'utilitza amb més freqüència.

Sol ser utilitzada després d'una capa convolucional.

Dropout

Aquesta operació s'aplica a les capes d'extracció d'informació i el què fa és reiniciar el pes de les connexions de la capa amb una probabilitat. Això s'utilitza perquè xarxa no tingui *overfiting* i sigui més robusta a informació no vista en el procés d'entrenament.

Flatten

Més que una capa això seria una operació per adaptar els *inputs* i els *outputs* de les capes explicades anteriorment. Com hem vist, *l'input* d'una capa *fully connected* no és el mateix que *l'output* d'una convolucional. Això fa que per passar els resultats de capes convolucionals a una capa densa s'hagi de passar per una operació de *flatten* la qual rep un vector de n-dimensions i el transforma en un d'una dimensió.

Entrenament del model: *loss function & optimizer*

De cara a la realització del model creat amb les capes adients de les comentades en l'anterior punt, cal definir dos aspectes per al procés l'entrenament d'aquest:

- *Loss function:* Aquesta és la funció que es vol minimitzar, és a dir mitjançant les dades d'entrenament, els pesos del model s'aniran modificant per tal de minimitzar aquesta funció. També s'anomena funció objectiu, ja que es la que indica si l'entrenament del model avança en la direcció adequada. Algunes de les *loss functions* més utilitzades són el *MSE* o la *cross-entropy.*

- *Optimizer*: Minimitzar la funció objectiu anomenada anteriorment és un problema d'optimització, l'*optimizer* és la tècnica que s'utilitza per dur a terme aquest procés. Alguns dels més utilitzats són el SGD, Adam o Adamax

3 Metodologia - Elaboració del projecte

En aquest apartat es detallaran els passos seguits en l'elaboració del projecte. El codi ha estat elaborat amb *Python* i en l'annex del projecte es poden trobar els *nootebooks* amb el codi dissenyat.

També dir que el procés que es detalla a continuació es el mateix per a les dades de Milan i per a les dades de Roma, la diferència la veurem en la part de resultats.

3.1 Selecció de l'entorn d'execució

Els models de *Deep Learning* i en aquest cas les xarxes neuronals artificials són models que a en temps d'entrenament consumeixen molts recursos de la màquina on s'executen. És per aquesta raó que cal escollir bé l'entorn d'execució del projecte.

La primera decisió és si executar-ho en local o buscar una solució al *cloud*. Degut a que el PC del qual es disposa per fer el projecte només te CPU i no consta de GPU, la decisió de buscar una alternativa al núvol es clara ja que a les GPU es pot paralitzar gran part del treball d'entrenament i per tant accelerar molt aquest procés.

Buscant eines gratuïtes per internet, l'única que deixa utilitzar GPU és el Colaboratory de Google i per tant s'ha decidit utilitzar aquesta eina.

3.1 Extracció de mel-band frequency features

El primer pas és obtenir dades representatives dels àudios del DYNAMAP per tal de poder començar a entrenar i provar diferents models amb aquestes.

Per extreure les dades s'han utilitzat els mateixos paràmetres que s'estan fent servir a l'equip que treballa amb el DYNAMAP a La Salle. Primer de tot es passa una finestra *hamming* de 30 ms amb intercalació al 50% sobre l'àudio i un cop fet això s'extreuen 40 característiques mel dins del rang de freqüències 0-2400 KHz.

El resultat de tot el procés anterior és un vector de 40 nombres reals per a cada instant mostrejat. Per tal de poder utilitzar aquestes dades en tots els models s'ha decidit guardar-les amb l'estructura següent.

Es crea un vector amb tantes posicions com àudios existents. Dins d'aquest vector un altre vector amb una posició per a cada un dels instants mostrejats de l'àudio en qüestió i finalment per a cada instant els 40 mels.

D'aquesta manera es guarden les dades ben organitzades i abans d'entrenar cada un dels models que veurem a continuació es podran transformar fàcilment a l'estructura requerida per aquest.

3.2 Creació i avaluació dels models

Com hem vist en el pas anterior hem deixat les dades ordenades per tal de que sigui fàcil adaptar-les a *l'input* requerit per a cada model. D'aquesta manera per a cada model utilitzat s'han elaborat tres passos:

1. Tractament del format de les dades per tal d'adaptar-les a l'entrada del model.

2. Creació del model.

3. Entrenament i avaluació del model creat.

Pel que fa la pas 1 i 2 seran específics de cada model i per tant es detallaran en cada un d'ells. Per altre banda el tercer és el mateix per a tots i per tant l'expliquem a continuació.

Finalment, dir que com es veurà a continuació, deixant apart un model de MLP que s'utilitza bàsicament per fer una primera proba sobre les dades extretes, les altres dues xarxes proposades integren capes convolucionals i recurrents. Entre tots els models que s'han vist en els *papers* del DCASE2017 on es tracta un problema similar al nostre s'han escollit aquests dos tipus de xarxes ja que combinen la potència de les convolucionals a l'hora de detectar característiques entre dades properes i per altre banda les capes recurrents permeten fer un bon anàlisis de la component temporal dels sons.

Entrenament i avaluació dels models

Degut a que aquests models són computacionalment costosos d'entrenar, per comparar-los entre ells no es farà *cross-validation*, sinó que l'avaluació es realitzarà amb un *traint-test Split* del 75-25%. Finalment al model escollit com a millor si que se li realitzar un *4-fold cross-validation* per fer una comparació justa amb els models creats per al GTM de La Salle. Els entrenaments de les xarxes es realitzen amb 200 *epochs* i un *bach size* de 30.

Cal esmentar que degut a que l'entrenament de les xarxes és molt costos, per tal de comparar els models entre sí no s'utilitzen totes les dades del DYNAMAP, sinó que s'agafa un subconjunt aleatori per tal de fer l'entrenament més lleuger, òbviament quan s'entreni la xarxa final es farà amb totes les dades, i utilitzant el temps que això requereixi, però s'ha hagut d'adoptar aquesta mesura per fer les probes per tal de guanyar una mica de temps.

Un cop entrenats els models amb les condicions descrites anteriorment, es calcula el *F1-score macro* (definida com la mitjana del F1 de l'equació de la Figura 7 per a totes les classes del problema) de cada model, ja que al tenir una *dataset* no balancejat és la mètrica que ens donarà més informació sobre la validesa d'aquest.

$$F_1 = \frac{2}{\frac{1}{\text{recall}} + \frac{1}{\text{precision}}} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}.$$

Figura 7: Equació F1-score

Vista la part en comú de tots els models, passem a veure doncs quins han estat els models que s'han provat en aquest treball i com ha estat el tractament de dades i la creació del model en sí en cada cas.

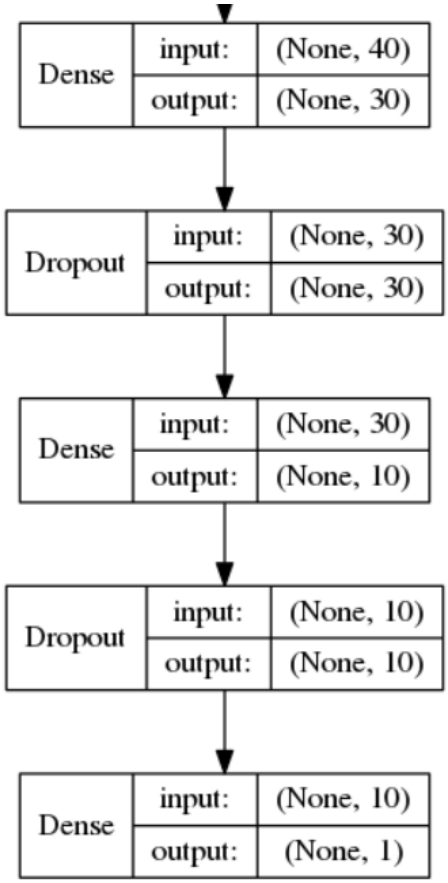
3.2.1 MLP

En principi aquest és el model més simple, ja que només s'ha creat una xarxa neuronal amb dues capes denses. Tot i això aquesta xarxa anirà bé per fer la primera pobre sobre el conjunt de dades obtingut i tenir un primer punt de partida.

Tractament de les dades

Aquest model rep com a *input* una conjunt de vectors d'una sola dimensió. Així doncs s'ha fet un simple *reshape* al conjunt de dades base explicat anteriorment per tal d'eliminar la primera dimensió del vector i obtenir un vector de dues dimensions on la primera és cada instant mostrejat de cada àudio i la segona els 40 mels.

Creació del model



Com podem veure en l'anterior imatge, el model consta de tres capes *fully-connected*, la primera i la segona amb 30 i 10 neurones respectivament, les dues amb activació *ReLu* i seguides d'un *dropout* del 50%. Finalment trobem la capa que dona la sortida del model, amb una neurona i activació *sigmoid*. D'aquesta manera obtindrem un nombre entre 0 i 1 que indicarà lo a prop que esta el registre de la classe 0 o 1.

Pel què fa a la forma en què es realitzarà l'entrenament s'ha pensat utilitzar la *loss funciton binary crossentropy*, ja que és la que es sol utilitzar a la literatura en la majoria de casos de classificació binaria en aquest tipus de xarxes. Finalment l'optimitzador serà un *adam.*

3.2.2 CRNN

Aquest segon model esta inspirat en el model creat en (Adavanne, 2017). El què s'ha fet és modificar el model per adaptar-lo a les dimensions i a les dades del nostre problema. Com el nom indica aquest model conte una part convolucional i una part recurrent. Les dades passen seqüencialment ambdues parts de la xarxa, en l'ordre que les hem mencionat.

Tractament de les dades

En aquest cas les dades entren a la xarxa com a vectors de dues dimensions, una de temporal i una altre amb els 40 mels.

Per tant per entrar les dades en aquesta xarxa la transformació consisteix en ajuntar *K* instants de temps consecutius de cada àudio, de manera que les dades d'entrada tenen tres dimensions, una per a cada dada, una amb les *k* dimensions temporals d'aquesta dada i finalment una amb els 40 mels de cada dimensió temporal.

En el aquest cas es treballarà amb *k* = 2.

Creació del model

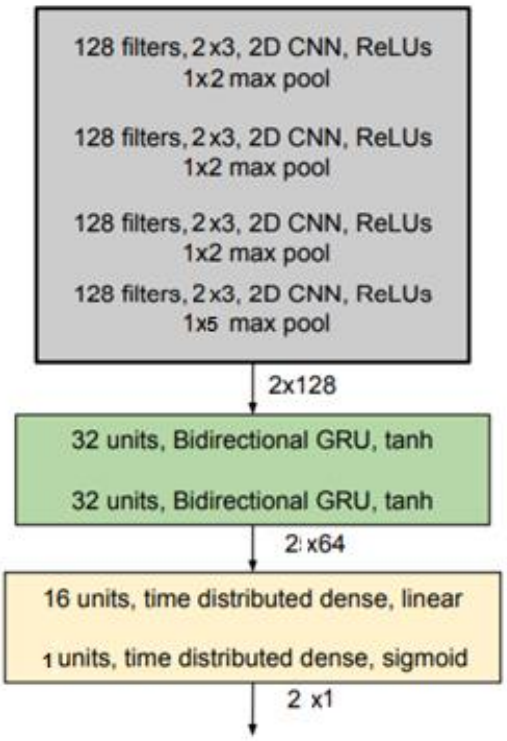


Figura 8: Model CRNN

En aquest cas el model passa a ser més complicat que el primer que hem explicat de **mlp**. Primer de tot les dades entren en un conjunt de blocs on la capa principal és una capa convolucional. L'estructura d'aquests blocs és una capa convolucional amb activació *ReLu,* una capa de *Dropout* del 50% i finalment una capa de *MaxPooling* que actua sobre la dimensió de les 40 mels, ja que la

temporal no la disminuïm. Els detalls de cada capa els podem veure en el plot del model de la Figura 3.

Les dades surten de la part convolucional amb unes dimensions de *k*x128 i entren a la part recurrent. Aquesta part esta formada per dues capes bidireccionals de GRU amb activació tangencial hiperbòlica i 32 neurones cada una.

Finalment les dades entren en dues capes denses i que amb activació *sigmoid* fan la classificació amb un enter entre 0 i 1 de la mateixa manera que en el model anterior.

Amb aquest model, s'aconsegueixen dues coses gràcies a les dues parts que conté. En la primera, gràcies a la part convolucional es troben patrons que hi puguin haver entre les característiques de bandes de freqüència i instants de temps propers.

Un cop vistes relacions locals amb la capa recurrent es relacionen els instants de temps propers entre si, ja que el so no deixa de ser una seqüència temporal i s'ha d'analitzar com a tal.

Finalment la compilació del model s'ha fet com en el cas anterior amb *binary crossentropy* i un optimitzador *adam* orientant-nos en la literatura existent sobre el tema.

3.2.3 pCRNN

Aquest tercer model esta inspirat en el model creat en (Dang, 2017). Com en el cas anterior, el què s'ha fet és modificar el model per adaptar-lo a les dimensions i a les dades del nostre problema. Com el nom indica aquest model conte una part convolucional i una part recurrent, però a diferència del cas anterior, ara les dades passen paral·lelament per ambdues parts de la xarxa i els resultats es concatenen i son classificats per una capa densa al final de la xarxa.

Tractament de les dades

El tractament de les dades per a aquest model és exactament el mateix que l'anterior, així que no el tornarem a repetir.

Creació del model

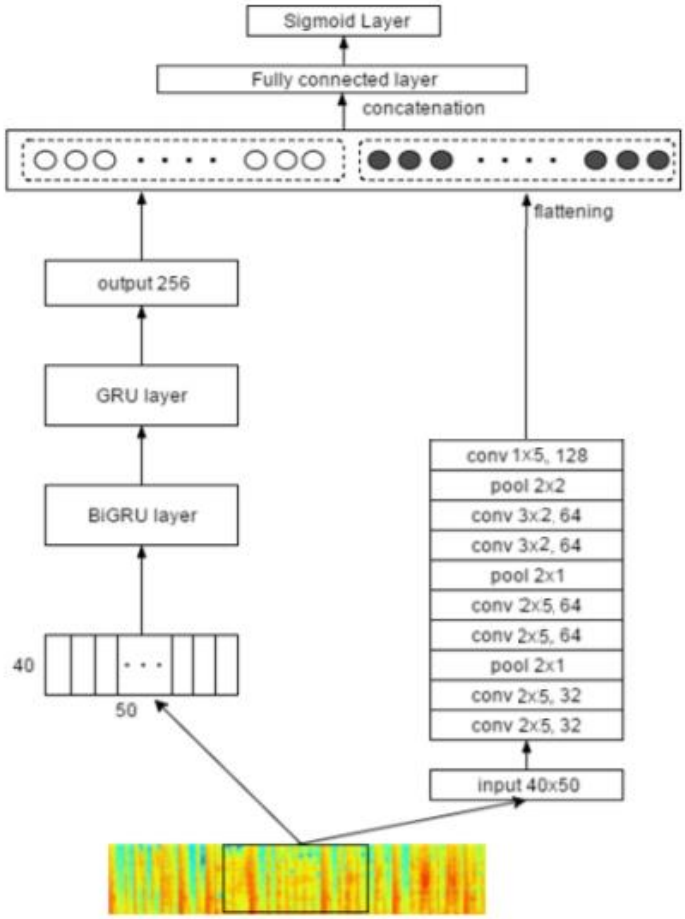


Figura 9: Model pCRNN

Com ja hem comentat aquesta xarxa no te totes les capes apilades seqüencialment sinó que té dos mòduls paral·lels. El primer d'aquests és una xarxa convolucional, formada per blocs amb capes convolucionals, de *dropout* i de *pooling* com en la xarxa anterior. Es poden veure el nombre exacte de capes i les configuracions dels paràmetres importants en la imatge del model (Figura 4). Aquest mòdul acaba amb una capa de *flatten* per aplanar les dades per tal de tenir-les amb el format adient per a la pròxima capa.

Per altre banda les dades passen per dues capes GRU, la primera d'elles bidireccional, ambdues amb activació hiperbòlica tangencial i amb 512 i 256 neurones respectivament.

Finalment les dades dels dos mòduls son concatenades i entren a dues capes denses de 128 i 1 neurones respectivament les quals serveixen com en tots els casos per donar el resultat de la classificació.

3.3 Entrenament del model escollit

Finalment, el model escollit (veurem la justificació en els resultats) ha estat el pCRNN.

Com s'ha explicat degut al temps que es requereix per entrenar els models, el procés de selecció d'aquest no ha estat amb un entrenament tant intensiu com s'hauria desitjat. No s'ha entrenat amb totes les dades i no s'ha avaluat amb un *4-fold cross validation* com en el cas dels models creats per el GTM.

Per tal d'obtenir el millor model possible, ara que s'ha decidit quin utilitzar, es farà també un *train-test split* del 75% però ara amb totes les dades per tenir un model basat en més informació i amb aquest calcular el *F1-macro* final.

3.4 Anàlisis de rendiment dels models

Cal dir que respecte al rendiment dels models, tant respecte al temps d'entrenament com en temps d'avaluació dels registres a predir no s'ha pogut fer una comparativa significativa degut a que l'execució com ja s'ha comentat anteriorment no s'ha realitzat en un entorn controlat, sinó en un entorn *cloud* on no sabem exactament quin hardware ens està donant suport en cada instant.

Tot i això, per tenir una mesura estimada del temps d'entrenament utilitzant l'entorn escollit, per als dos models complexos que son el **CRNN** i el **pCRNN** es tarda aproximadament 10 hores cada 10 iteracions si l'entorn no té GPUs disponibles i per tant has d'executar en CPU i 2 hores cada 10 iteracions si es disposa de GPU.

4 Resultats

Tal i com s'ha detallat a la metodologia, els resultats obtinguts són amb un *traint-test split* del 75%-25% i utilitzant com a mesura el F1 macroaveraged, la qual és una mètrica adequada per veure la *performance* de models sobre dades no balancejades com és aquest cas.

Pel què fa al nombre d'iteracions d'entrenament realitzades (*epochs)* se n'han realitzat 200 per a cada model i cada entorn (Milà i Roma).

4.1 Milà

4.1.1 Comparativa de models

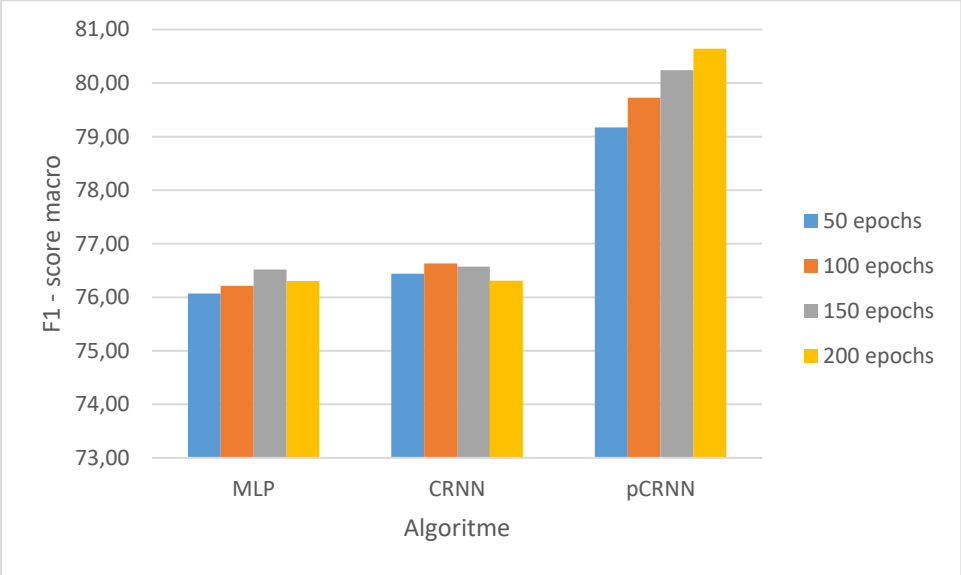


Figura 10: F1-score macro per Algoritme i nombre d'epochs (Milà)

Com hem vist en l'apartat de metodologia, s'han provat tres tipus de xarxes per tal de resoldre el problema plantejat. En la Figura 10 es pot veure una gràfica amb el conjunt de proves elaborades sobre les dades de Milà.

Es pot veure que els resultats han estat extrets durant el procés d'aprenentatge, de tal manera que a part del *F1-macro* final s'observa com ha evolucionat aquest durant les iteracions d'aprenentatge del algoritme.

El primer que es pot veure en la gràfica és que hi ha un algoritme que clarament obté resultats millors que els altres. Mentre que el **MLP** i el **CRNN** no passen del 77% el **pCRNN** gairebé arriba al 81%.

A part del F1-score final, el qual indica que el **pCRNN** modela millor aquestes dades que els altres dos mètodes, també és important veure la tendència, ja que els models de **MLP** i **CRNN** sembla que ja no aprenen més, però que el model de **pCRNN** encara no està estancat, i per tant que amb més iteracions d'entrenament encara podria millorar. Seria interessant veure què passa amb més iteracions, però degut a l'elevat temps d'entrenament de les xarxes i a la necessitat de fer altres execucions abans que provar mes *epochs* aquí, es deixa plantejada aquesta qüestió en l'apartat de treballs futurs.

Mirant amb més detall els resultats d'aquest model, podem veure en la següent matriu de confusió com el problema principal recau en identificar els **ANE** ja que només s'aconsegueix identificar el 61.87% mentre que en la classe **RTN** s'obté un 82.17% d'encert. Això segurament es degut a la homogeneïtat de la classe **RTN** les dades d'aquesta classe són relativament fàcils d'identificar, per altre banda **ANE** és molt més heterogènia (des de sons d'ocells fins al so d'un clàxon) i conté moltes menys dades per a l'entrenament, això fa que hi hagi molta variabilitat en aquesta classe i per tant sigui complicat identificar-la.

real\prediction	RTN	ANE
RTN	11106	983
ANE	2410	3911

Figura 11: pCRNN Confussion Matrix (Milà)

4.2 Roma

4.1.1 Comparativa de models

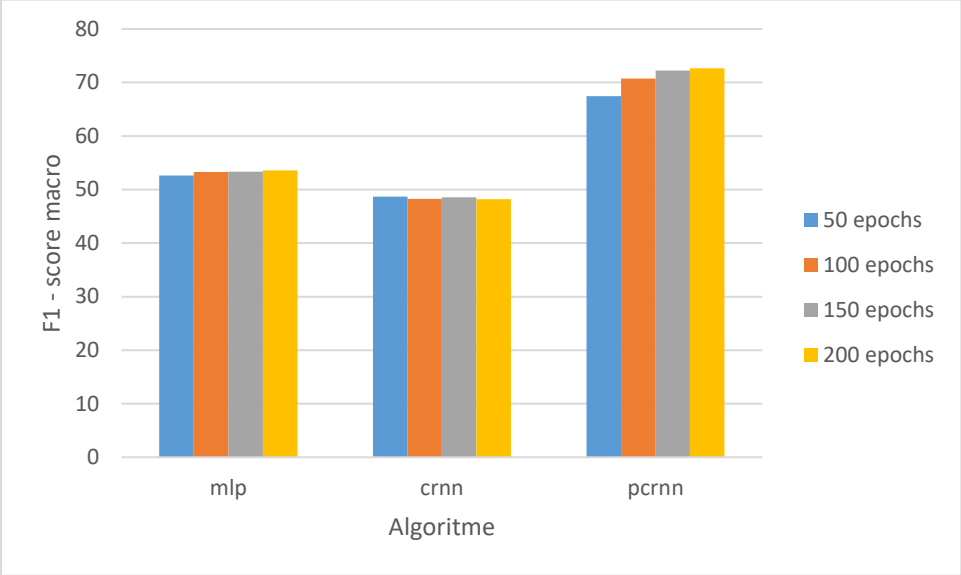


Figura 12: F1-score macro per Algoritme i nombre d'epochs (Roma)

A la Figura 12, es pot veure una gràfica similar al presentat anteriorment per el cas de Milà.

En aquest cas es veu que la diferència entre els tres models és molt més important, tot i que el model que obté un millor F1-score segueix essent el mateix: el **pCRNN**.

S'observa que els models **MLP** i **CRNN** mantenen el seu F1-score proper al 50% en totes les iteracions d'entrenament realitzades per als algoritmes.

Per altra banda i també com en el cas de Milà el **pCRNN** és l'únic que sembla que pot no estar estancat en l'aprenentatge, tot i que en aquest cas els resultats no són tant bons com en el cas anterior ja que es queda al 72.63%.

Mirant la matriu de confusió de la Figura 13, com és d'esperar després dels resultats vistos, es torna a veure un comportament similar al d'aquest model amb les dades de Milà. El principal problema recau en identificar la classe ANE on s'obté una precisió del 39.34%, mentre que per a la classe principal s'obté un 99.09% d'encert.

real\prediction	RTN	ANE
RTN	76652	700
ANE	2555	1657

Figura 13: pCRNN Confussion Matrix (Roma)

4.3 Models finals

Tal i com hem dit a la metodologia finalment es fa una comparació dels resultats obtinguts amb l'escollit com a millor model dels provats en el projecte respecte els dos models amb els quals s'han obtingut millors resultats en el GTM, el **KNN** i **SVM**.

Finalment, s'ha decidit elaborar aquesta part només per a Milà, ja el conjunt de dades és més similar als vistos al DCASE Challenge i els models inspirats en els vistos en la literatura d'aquest esdeveniment han resultat més bons per a aquest conjunt que per a les dades de Roma.

Mirant doncs els resultats per a les dades de Milà, en la Figura 14 es veu com el millor resultat obtingut des del GTM es d'un F1-score del 81.44% Milà amb un **SVM**.

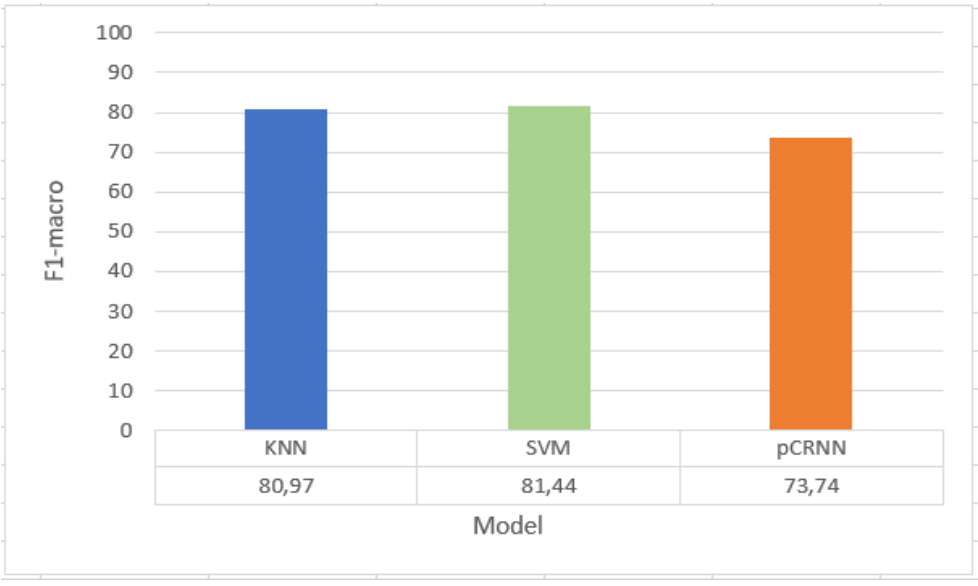


Figura 14: Comparació models GTM

Pel què fa al model del projecte, com ja s'ha vist el millor model és el **pCRNN**, per això, tal i com s'ha explicat en la metodologia en aquest punt s'ha entrenat aquest model amb tot el conjunt de dades de Milà per tal de poder compara els resultats obtinguts amb el **SVM** creat per el GMT.

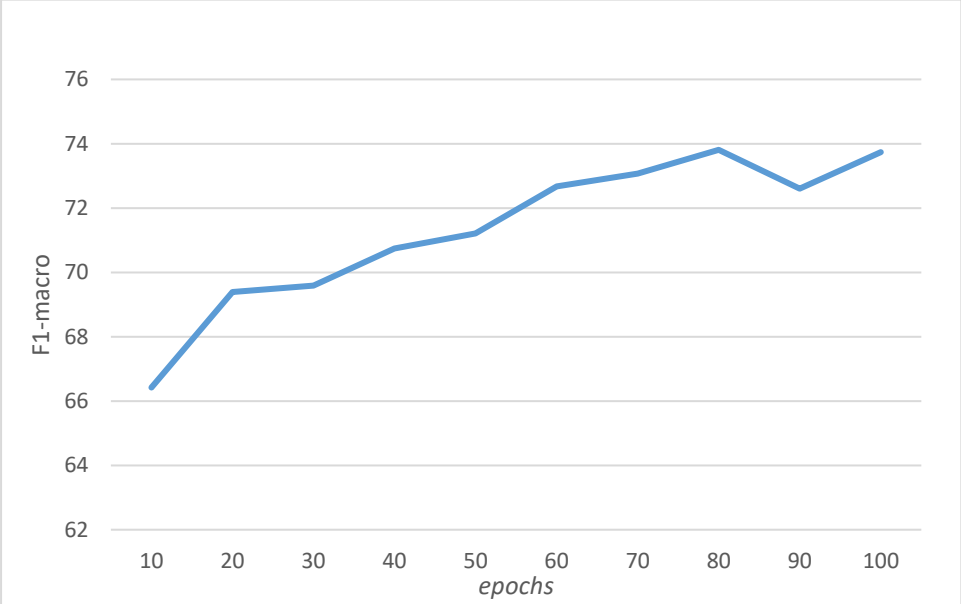


Figura 15: Evolució del F1-macro en el pCRNN amb dades de Milà

Els resultats obtinguts amb el model creat en el projecte poden ser considerats bons resultats, ja que el F1-macro de 73,74% no és tant bo com el de 81.44% obtingut per el **SVM** però s'hi acosta i la Figura 15 permet pensar que el model té marge de millora per arribar als nivells d'aprenentatge del **SVM** o inclús superar-lo.

Com s'observa en la Figura 15 l'entrenament sembla que no està estancat, així que com es veurà més endavant, a l'apartat de treballs futurs es proposa establir un entorn d'execució en què es redueixi el temps d'entrenament d'aquestes xarxes per poder fer més *epochs* i veure si la progressió que es veu a la Figura 15 segueix. Cal tenir en compte que en la literatura vista del *DCASE2017* els models eren entrenats amb 500 *epochs,* però en l'entorn on s'està executant aquest nombre d'iteracions és inviable, tant per el temps d'execució com per els *time-out* que té el servei i que no et permeten deixar una tasca executant-se indefinidament.

Finalment, mirant la matriu de confusió resultant de l'avaluació de l'últim model (Figura 16), s'observa el mateix problema que s'ha observat al llarg de tota la memòria, la classe ANE és la que posa més dificultats als models.

real\prediction	RTN	ANE
RTN	238011	5074
ANE	18928	12463

Figura 16 : pCRNN Confussion Matrix

Cal dir que la comparativa no es del tot justa ja que els models de *La Salle* han estat avaluats amb un *4-fold cross validation* en canvi degut al seu elevat temps d'entrenament els models del projecte han estat testejats amb un traint-test *split* del 75-25%. Tot i això el *split* 75-25% és el mateix que es fa servir en el *4-fold cross validation* i per tant l'únic que es perd és la normalització de la mètrica al prendre-la 4 vegades, però la proporció de dades amb què es pren és la mateixa.

5 Conclusions i treballs futurs

5.1 Conclusions

Com a objectiu principal del projecte s'ha plantejat donar resposta a la qüestió de si el fet d'aplicar *Deep Learning* a les dades del DYNAMAP podia millorar els resultats obtinguts fins al moment, tenint en compte el bon comportament d aquest conjunt de tècniques en altres problemes similars tractats en la literatura.

Parlant primer de les dades utilitzades, s'han utilitzat el conjunt d'àudios del LIFE DYNAMAP, els quals han estat gravats en dos entorns, Roma i Milà. Aquests àudios estan etiquetats, i es disposa de 9 hores i 8 minuts d'àudio entre els dos entorns.

Pel què fa a les característiques extretes dels àudios, s'han extret les mateixes dades dels àudios que en el els *papers* vistos al *DCASE2017*, *40 mel-band freqüències.* En aquest punt en el projecte s'ha decidit actuar diferent que el GTM de La Salle, on s'utilitzen 13 MFCCs, ja que en tots els casos vistos del DCASE2017 on s'apliquen classificadors semblants als elaborats en el projecte s'utilitzen els 40 *mel-bands.*

Per dur a terme el procés de disseny i creació de les xarxes neuronals s'ha començat mirant models amb bons resultats al DCASE CHALLENGE 2017, una competició de prestigi en l'àmbit científic que tracta un problema molt similar al del DYNAMAP. Explorant aquesta literatura s'han identificat dos models de xarxes neuronals artificials els quals per la seva estructura i els seus resultats han resultat molt interessants i han servit com a base per a la creació dels models proposats en el projecte.

Com ja s'ha vist, ambdós models estan formats per capes convolucionals i capes recurrents, la diferència entre ells es que en el primer aquestes capes treballen en sèrie (**CRNN**) i en el segon en paral·lel (**pCRNN**). Ambdós models han estat adaptats per modelar les dades del DYNAMAP i finalment han estat testejats amb un *traint-test Split del 75-25%* juntament amb un model simple de MLP que ha permès dur a terme un primer estudi de viabilitat de l'aplicació d'aquestes tècniques sobre les dades del projecte DYNAMAP

Els resultats obtinguts en la comparació d'aquests mètodes entre ells, ja han donat un indici de que un model de *Deep Learning* es possible que millori la *performance* dels models actuals del GTM, tot i que s'haurien de realitzar més iteracions per tal de confirmar-ho.

Durant el procés de comparació de models, el qual s'ha executat amb un subconjunt aleatori del 50% de les dades, mantenint la distribució d'aquestes perquè el subconjunt sigui representatiu, ja que entrenar els models amb la totalitat d'elles requeria massa temps de computació, s'han obtingut els resultats següents. En ambdós casos, tant Milà com Roma el millor model ha estat el **pCRNN** donant uns F1-macro del 80.64% i 72.64% respectivament. També ha estat interessant veure que sembla que l'aprenentatge amb 200 *epochs* no queda estancat, així que és possible que el model millorés amb més iteracions d'entrenament. Cal destacar, que degut a la seva heterogeneïtat, la classe ANE ha estat la que ha portat més problemes als models.

Un cop vist que el **pCRNN** és el classificador que ofereix millor resultat, s'ha procedit a la creació d'una model amb la totalitat de les dades per a l'entorn de Milà, ja que és l'entorn més semblant als vistos en el *DCASE2017*. El resultat ha estat, un altre cop utilitzant un *traint-test Split del 75-25%* com a mètode d'avaluació, un F1-macro del 73,74%. Aquest resultat no ha aconseguit igualar l'obtingut per el millor model del GTM (un 81.44% amb **SVM**) però si que obté uns resultats bastant similars, i el més important és que, com es pot observar en l'apartat 4.3 de resultats, sembla que té marge de millora.

Per concloure, dir que en el conjunt de proves sobre les dades del DYNAMAP amb els models de *Deep Learning* elaborats en aquest projecte han permès obtenir resultats satisfactoris, que tot i no superar la *performance* dels models previs desenvolupats del GTM, permeten pensar que invertint el temps necessari per millorant els models proposats, es pot aconseguir un classificador millor que els actuals per al DYNAMAP.

5.2 Treball futur

Com s'ha vist durant la part de la metodologia i els resultats, un dels grans *stoppers* a l'hora de realitzar més probes de les que s'han fet ha estat el temps d'entrenament de les xarxes creades. Així doncs, el primer conjunt de treballs futurs que es plantegen a partir d'aquest projecte es centren en millorar aquesta part.

El primer pas proposat és buscar un entorn d'execució controlat, on sapiguem els recursos dels que disposem i que no pugui ser interromput per un *time-out* del navegador o de la pàgina utilitzada. Les opcions més viables serien un PC propi amb una GPU potent o un servidor *cloud* amb GPU fixe disponible. Per tal de instal·lar tots els paquets necessaris per a l'execució es podria utilitzar una eina com *Doker* que facilita molt aquest procés a vegades una mica farregós.

Amb això fet, es proposa bàsicament dur a terme més proves:

1. Comparar els models del projecte però amb totes les dades des del principi. S'ha suposat que un model que tenia una millor *performance* amb un subconjunt aleatori de les dades també el té amb totes les dades, però això pot no ser cert.

2. Tant el model de **CRNN** com el model de **pCRNN** tenen un paràmetre *k* que ha estat fixat a 2 en aquest projecte. Estaria bé fer proves augmentant aquesta *k* i veient la resposta dels models.

3. Provar més models. Els models provats, tal i com hem dit, estan inspirats en dos models vistos en la literatura del *DCASE CHALLENGE2017*, on han donat molt bon resultat, però poder seria interessant provar xarxes amb altres estructures.

Per altra banda, deixant ja apart les proves que es podrien fer amb més xarxes neuronals, es podria fer un model amb altres mètodes de *Deep Learning* com per exemple *Deep Trees*, per veure quina és la seva *performance* en aquest problema.

Finalment, el què s'ha vist en aquest projecte és només la creació del model, però aquest model que només es pot executar des de *Python* no té molta utilitat si no s'integra en una eina on traslladi aquesta informació a la població. Per aquesta raó seria convenient fer proves de rendiment en el servidor del DYNAMAP i en cas de satisfer les restriccions temporals, incluir el model en els sistemes de visualització d'aquest.

Ja per acabar, dir que seguint el model de treball utilitzat en el GTM de La Salle s'ha creat un classificador per Roma i un per Milà. És possible que es pogués elaborar un model vàlid per ambdós ambients així que aquesta és l'última prova que es proposa en aquest apartat de treballs futurs.

6 Referencies

Adavanne, S. a. (2017). A Report on Sound Event Detection with Different Binaural Features.

Berkhin, P. (2006). *A Survey of Clustering Data Mining Techniques*. Recollit de https://link.springer.com/chapter/10.1007/3-540-28349-8_2

Colaboratory. (5 / 5 / 2018). Recollit de https://colab.research.google.com/

Dang, A. a.-C. (2017). Deep Learning for {DCASE2017} Challenge.

DCASE2017. (6 / 5 / 2018). Recollit de http://www.cs.tut.fi/sgn/arg/dcase2017/challenge/index

http://haythamfayek.com/2016/04/21/speech-processing-for-machine-learning.html. (14 / 5 / 2018). Recollit de http://haythamfayek.com/2016/04/21/speech-processing-for-machine-learning.html

Joan Claudi Socoró, F. A.-P. (2017). *An Anomalous Noise Events Detector for Dynamic Road Traffic Noise Mapping in Real-Life Urban and Suburban Environments.* Recollit de http://www.mdpi.com/1424-8220/17/10/2323/htm

John Neter, M. H. (23 / 5 / 2018). *Applied Linear Regression Models*. Recollit de https://yytvb1kwd07.storage.googleapis.com/MDI1NjA4NjAxWA==07.pdf

Keras. (29 / 4 / 2018). Recollit de https://keras.io/

Librosa. (3 / 5 / 2018). Recollit de https://github.com/librosa/librosa

LIFE DYNAMAP. (27 / 4 / 2018). Recollit de http://www.life-dynamap.eu/

NumPy. (5 / 5 / 2018). Recollit de http://www.numpy.org/

Pedregosa, F. a. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 2825-2830.

Python Software Foundation. (3 / 05 / 2018). *glob — Unix style pathname pattern expansion*. Recollit de https://docs.python.org/3/library/glob.html

Rosenblatt, F. (4 / 5 / 2018). *The perceptron: A probabilistic model for information storage and organization in the brain.* Recollit de http://psycnet.apa.org/record/1959-09865-001?doi=1

S.R. Safavian, D. L. (16 / 5 / 2018). *A survey of decision tree classifier methodology*. Recollit de https://ieeexplore.ieee.org/abstract/document/97458/

Shikha Gupta, J. J. (2013). FEATURE EXTRACTION USING MFCC.

Vapnik, C. C. (23 / 5 / 2018). *Support-vector networks*. Recollit de https://link.springer.com/article/10.1007%2FBF00994018

7 Annex

1 *Notebook* Milà:

https://colab.research.google.com/drive/1lzLCwOEElq2NtyFO_JxRTYa911FAsnjl

2 *Notebook* Roma:

https://colab.research.google.com/drive/1bxzIBxw1sZyvx7faC6lvOKnZOvYOSsPF