



UNIVERSITAT DE
BARCELONA

Treball final de grau
**GRAU D'ENGINYERIA
INFORMÀTICA**

Facultat de Matemàtiques i Informàtica
Universitat de Barcelona

**XARXES NEURONALS PER A
LA GENERACIÓ I
REPRESENTACIÓ D'ONES**

Autor: Pablo Laiz Treceño

Director: Dr. Santi Seguí
Realitzat a: Departament
Barcelona, de Matemàtiques
26 de gener de 2017 i Informàtica

Abstract

During the last few years, neural networks have become a relevant discipline in the field of machine learning. In this project, several neural networks architectures were proposed for the representation and generation of sound waves. The final goal of this project is the generation of sounds to accompany silent videos, reproducing the Foley Effect. However, it was required to study how to obtain the sound characteristics and how to generate audios. To determine if the architectures are suitable, two different data sets were used: a group of synthetic notes, and a set of sounds extracted from tennis matches. In the representation of waves, autoencoders networks were used to obtain a feature vector. Two representation types were employed: the sound wave, and the spectrogram. To generate waves, sound was created using Google Wavenet network. The amplitud was predicted as a function of the previous amplitudes.

Resum

En els últims anys, les xarxes neuronals s'han convertit en una disciplina rellevant dintre de l'aprenentatge automàtic. En aquest treball, es presenten diferents arquitectures de xarxes neuronals per a la representació i generació d'ones de so. L'objectiu final d'aquest treball és la generació de sons relacionats amb vídeos muts, intentant reproduir l'efecte Foley. Però, prèviament s'han d'estudiar com obtenir les característiques del so i com generar àudios. Per determinar si les arquitectures són adequades es fan servir dos conjunts diferents de dades: un conjunt sintètic de notes i un segon generat amb sons extrets de partits de tennis. Per la representació del so s'han utilitzat xarxes *autoencoders* per tal d'obtenir un vector de característiques i s'han fet servir dos tipus de representacions d'àudio, l'ona de so i l'espectrograma. Referent a la generació d'ones, s'han creat sons a partir de la xarxa WaveNet de Google, predient cada amplitud de l'àudio en funció de les anteriors.

Agraïments

En primer lloc, vull agrair al Dr. Santi Seguí tota la seva feina, ajuda i hores dedicades en aquest treball final de grau, que sobrepassen les seves obligacions com a tutor.

En segon lloc, vull agrair a la meva família pel suport, ajuda i paciència que han tingut amb mi durant aquests últims anys.

En tercer i últim lloc m'agradaria agrair a tots els meus amics i companys la motivació i força donada al llarg d'aquests sis anys de grau.

Contingut

1	Introducció	1
1.1	El projecte	1
1.1.1	Context	1
1.1.2	Motivació	1
1.1.3	Objectius	2
1.2	Estructura de la memòria	2
2	Estat de l'Art	4
3	Teoria del So	6
3.1	Què és el so?	6
3.1.1	Amplitud i pressió	6
3.1.2	Freqüència i tons	6
3.2	Representació digital del so	7
3.2.1	Teoria del mostreig	8
3.3	Representació del so en imatges	9
3.3.1	Sèrie i transformada de Fourier	9
3.3.2	Espectrograma	9
4	Xarxes Neuronals Artificials	11
4.1	Què són?	11
4.2	Representació del model	11
4.3	Funcions d'activació	14
4.4	Tipus d'aprenentatge	16
4.4.1	Aprenentatge supervisat	16
4.4.2	Aprenentatge no supervisat	17
4.5	Tipus de xarxes neuronals	17
4.5.1	Xarxa de perceptró	17
4.5.2	Xarxes <i>fully connected</i> o denses	18
4.5.3	Xarxes convolucionals	18
4.5.4	<i>Autoencoders</i>	20
4.5.5	Xarxes recurrents	21
4.6	Aprendentatge de la Xarxa	23
4.6.1	Funció de cost	23

4.6.2	Descens del gradient	24
4.6.3	Algorisme <i>BackPropagation</i>	25
4.6.4	Tècnica d'aprenentatge <i>Dropout</i>	26
5	Framework del Treball	27
5.1	Theano	27
5.2	TensorFlow	27
5.3	Keras	28
5.4	Discussió	28
6	Representació del so	29
6.1	Arquitectura	29
6.1.1	Mètode 1: Representació amb l'ona de so	29
6.1.2	Mètode 2: Representació amb l'espectrograma	31
6.2	Conjunt de dades	31
6.2.1	Notes musicals sintètiques	32
6.2.2	Sons corresponents a un partit de tennis	33
6.3	Resultats obtinguts i discussió	33
6.3.1	Mesura de la qualitat	33
6.3.2	Comparació de funcions de cost	36
6.3.3	Resultats de representació amb ones de so	36
6.3.4	Resultats de representació amb espectrogrames	44
7	Generació d'ones de so	47
7.1	Arquitectures	47
7.1.1	Mètode 1: Xarxes neuronals recurrents	47
7.1.2	Mètode 2: WaveNet	48
7.1.3	Mètode 3: Metodologia que es segueix en l'article <i>Visually Indicated Sounds</i>	50
7.2	Conjunt de dades	51
7.3	Resultats obtinguts i discussió	51
7.3.1	Resultats amb la xarxa neuronal recurrent	51
7.3.2	Resultats amb la xarxa WaveNet	52
8	Conclusions	57
8.1	Objectius proposats	57

8.2 Treball Futur	57
-----------------------------	----

1 Introducció

1.1 El projecte

1.1.1 Context

Avui en dia les noves tecnologies permeten la implementació de noves formes de programació. Algunes d'aquestes intenten emular el funcionament del cervell humà aprenent a partir de les dades sense la necessitat d'haver d'especificar cap model.

Si es pensa en la relació entre sons i sistemes informàtics, hi ha múltiples opcions: la reproducció de sons provinents de pel·lícules o jocs, la gravació de sons o, inclús la composició de música a partir de programes musicals. En tots aquests casos, l'ésser humà ha hagut d'intervenir per generar prèviament els sons d'una forma o una altra. Per tant, sembla que els ordinadors no puguin generar sons per si mateixos.

A partir del final del segle XX, amb l'evolució de la tecnologia, comencen a aparèixer cassetes, reproductors de CD portables, mp3, mp4, entre d'altres. El principal inconvenient dels primers aparells de reproducció de so era la seva poca memòria, fet que no permetia la introducció de tota la música desitjada. Amb el pas dels anys, es van començar a generar compressors per disminuir el pes dels arxius i poder emmagatzemar més informació en menys espai.

L'any 1895 els germans Lumière van produir la primera pel·lícula de la història [1] i aquesta no tenia so. Les pel·lícules posteriors continuaven sent mudes, però algunes d'elles tenien un acompanyament de piano en directe durant la projecció. Anys més tard, a algunes d'aquestes produccions se'ls va generar un so, i aquest procés es coneix com Efecte de Sala o Efectes Foley [2]. Aquest concepte, d'una forma general, consisteix en construir o produir el so adequat per escenes on no n'hi havia.

En aquest projecte, es voldrà simular la fita anterior, però amb la intenció que els propis sistemes informàtics puguin trobar ells sols la solució. Serà això possible?

1.1.2 Motivació

En els últims anys, el camp de recerca de l'aprenentatge automàtic (*Machine Learning*) i de les xarxes neuronals s'ha tornat a enlairar. Això és conseqüència del descobriment de tècniques d'aprenentatge per les xarxes profundes i les millores constants en l'àmbit del *hardware*.

Durant el transcurs dels sis anys del doble Grau de Matemàtiques i Enginyeria Informàtica en cap assignatura s'ha fet incís en l'aprenentatge automàtic ni en les xarxes neuronals. Fer aquest treball representa una oportunitat única d'aprendre en un camp fonamental en la recerca dels propers anys

1.1.3 Objectius

L'objectiu general que es proposa en aquest treball és relacionar les xarxes neuronals artificials i la representació i generació de so a partir de l'ordinador. Sempre que es parli de sons, música, àudio, etcètera, en aquesta memòria ens referirem a l'ona de so.

El primer objectiu és trobar una representació mitjançant un vector de característiques d'una ona de so. Aquest haurà d'ocupar un menor espai d'emmagatzematge sense perdre informació rellevant.

El segon objectiu és la generació de diversos tipus de sons o música i entendre l'arquitectura utilitzada.

Finalment, i si tots els objectius anteriors s'aconsegueixen, es vol crear un sistema que mitjançant xarxes neuronals sigui capaç d'aplicar el concepte d'efectes de Foley a un partit de tennis. És a dir, proporcionar un vídeo mut a la xarxa i que aquesta generi el so conforme a la seqüència d'imatges.

Per tal de poder assolir els nostres propòsits, cal prèviament adquirir uns coneixements bàsics sobre les xarxes neuronals, les seves parts, característiques, defectes, funcionament, etcètera.

1.2 Estructura de la memòria

Per a la redacció d'aquesta memòria s'ha seguit un model que va de menys a més, és a dir, que comença pels coneixements més bàsics i acaba explicant l'arquitectura necessària per arribar als objectius proposats. Aquest treball es dividirà en dues parts: la primera part formada pels punts 2, 3 i 4 correspondrà al marc teòric i la segona part formada pels punts 5, 6 i 7 correspondrà a la part experimental del treball. A continuació es tractaran cadascuna de les parts de les quals consta aquesta memòria. La numeració va associada als apartats del treball.

2. **Estat de l'Art:** curta exposició del que s'ha investigat fins el moment en la representació i generació de so.
3. **Teoria del So:** breu introducció sobre què és el so, com es treballa amb les ones de so de forma digital i la seva representació en imatges.
4. **Xarxes Neuronals Artificials:** conté els coneixements bàsics necessaris per entendre les arquitectures i implementacions d'aquest treball.
5. **Framework del Treball:** explicació dels diversos frameworks disponibles avui en dia per a la programació d'aprenentatge automàtic i xarxes neuronals. Justificació de l'elecció del framework.
6. **Representació del so:** recull de les arquitectures que s'han estudiat i els seus inconvenients. Explicació de les dades utilitzades. Anàlisi dels resultats obtinguts en els diversos experiments i la comparació entre ells.

7. **Generació del so:** recull de les arquitectures que s'han estudiat i els seus inconvenients. Explicació de les dades utilitzades. Anàlisi dels resultats obtinguts en els diversos experiments i comparació d'alguns d'ells.
8. **Conclusions:** resum del treball i exposició de les conclusions.

Finalment a l'adreça de GitHub <https://github.com/laizpablo/NeuralNetwork> es podrà accedir al codi i resultats d'aquest treball.

2 Estat de l'Art

En aquesta secció es parlarà sobre els últims mètodes, algorismes, xarxes, etcètera, que s'han investigat recentment en la representació i generació d'ones de so.

En el camp de la representació de les ones de so, o en el camp de la compressió, existeixen una gran quantitat de mètodes i còdecs que redueixen la mida de la informació sense que apparentment es noti la pèrdua [3, 4]. Molts d'aquests compressors funcionen eliminant les freqüències que estan per sobre i per sota d'un llindar que les persones no escolten. Eliminant aquestes freqüències la mida del fitxer disminueix i les persones continuen percebent d'una forma aproximada els sons. Sound Forge o Audacity són alguns dels programes més coneguts que permeten el tractament d'ones de so i les conversions de diversos tipus d'àudios a d'altres formats.

Una xarxa neuronal és un conjunt de neurones computacionals que simulen el comportament de les neurones d'un cos humà. Aquesta xarxa té com a objectiu aprendre a partir de mostres donades. Alguns dels conceptes que sorgeixen en els següents paràgrafs seran explicats més endavant.

Avui en dia es coneixen diverses xarxes neuronals que generen música amb resultats positius. A continuació es comentaran algunes d'elles:

- **Magenta** [5] és la xarxa neuronal de Google Brain Team, de codi obert. El projecte es va donar a conèixer al juny del 2016 i està format per una xarxa neuronal recurrent amb unitats LSTM's (Long short-term memory)¹ [6, 7], implementada amb TensorFlow [8]. Tot i ser coneguda per a la generació de música, també està creada de manera que pugui generar art en forma de pintures.

Centrant-se en la part musical, el conjunt de dades que utilitza com a entrenament per aprendre a generar música és un conjunt de cançons en format midi.

Magenta només pot generar una seqüència de notes simultàniament. En l'actualitat, l'equip de Google està estudiant el processament de música polifònica. Estan intentant generar una seqüència de notes per a cadascun dels instruments que participen en la cançó.

- **DeepJazz** [9] és el resultat d'una hackathon de Ji-Sun Kim on l'arquitectura està composta per una xarxa neuronal recurrent amb dues capes d'unitats LSTM que aprenen d'un fitxer midi. Aquesta va ser implementada amb els frameworks Keras [10] i Theano [11].
- **FlowMachines** [12] és la xarxa neuronal creada pel grup *European Research Council* i coordinats per François Pachet. El seu objectiu és que el sistema generi partitures musicals basades en el mateix estil de música que el compositor de les partitures. Els processos de decisió de Markov [13] s'utilitzen en aquest cas com a tècnica d'aprenentatge de la xarxa neuronal. El conjunt de dades d'entrenament està format per 13 000 partitures musicals.

¹Concepte explicat a la secció 4.5.5

- **WaveNet** [14] és l’arquitectura creada pel grup de recerca de *DeepMind* de Google, de codi tancat. La xarxa va ser ideada per a aplicacions de conversió parla-veu (*text-to-speech*), on generen un so de veu més natural i on s’han obtingut resultats positius. Tot i que no és la idea original, aquest mètode també es pot aplicar a la generació de música i a d’altres àmbits relacionats amb els sons.

La part negativa d’aquest sistema és que pot trigar una quantitat d’hores considerable per entrenar un segon de so.

- **Gruv** [15] és la xarxa neuronal creada per un grup d’investigació de Stanford el juny del 2015. Aquesta xarxa treballa amb ones d’àudio com a entrada, igual que la WaveNet. L’arquitectura està formada per una xarxa recurrent amb cèl·lules LSTM i GRU [7]. Els investigadors a l’article de presentació comenten que la seva implementació és insuficient per entrenar amb una gran quantitat de dades.

Els sistemes anteriors són alguns dels creats fins el moment. En general, la generació de música es podria dividir en dos conjunts: els que generen notes i els que generen qualsevol tipus de so a partir d’ones d’àudio. El primer conjunt es limita a produir notes per un instrument, o més d’un. En canvi, el segon es pot utilitzar en més àmbits a part de la generació de música.

La característica més important és que la generació de notes té poc cost computacional mentre que la generació de l’ona d’àudio és molt més costosa.

En aquest projecte es vol generar qualsevol tipus de so, per tant s’haurà de generar tota l’ona de so. L’única xarxa que genera tota l’ona de so és la xarxa anomenada WaveNet.

3 Teoria del So

3.1 Què és el so?

El so és un fenomen que relaciona la física i la percepció dels éssers vius [16]. El so s'entén com la percepció de les ones mecàniques en el cervell quan aquestes arriben a través d'algun medi.

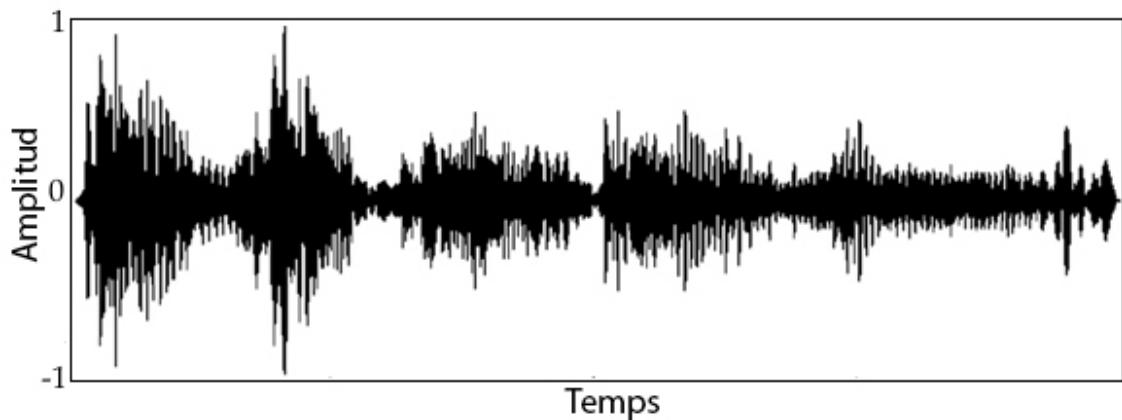


Figura 1: Ona de So. Imatge extreta de [16]

3.1.1 Amplitud i pressió

A la Figura 1 es mostra la representació d'una ona de so. L'eix x expressa el temps i l'eix y l'amplitud per cada unitat de temps. Mitjançant aquests tipus de gràfica s'obtenen les representacions més simples per visualitzar els sons. És obvi que en aquesta gràfica, també anomenada funció, l'eix temporal és l'entrada de la funció i l'amplitud (pressió), la sortida. A causa d'això, s'observa que la variable del temps és una variable contínua perquè per cada parella d'instants de temps, existeix un tercer instant entre els dos anteriors.

Durant aquest treball el so s'entendrà com una seqüència d'amplituds que van variant en funció del temps. Quan l'amplitud té, durant un cert temps, un valor que és aproximadament zero, llavors es tindrà silenci. Quan es treballa amb el so digitalitzat es tindrà una llista amb els valors de les amplituds a diversos instants de temps, i passarà a ser una funció discreta.

3.1.2 Freqüència i tons

La freqüència és la quantitat de vegades que es repeteix un esdeveniment en el temps. Els hertz (Hz) són la unitat física per mesurar la freqüència.

El rang de freqüències del so està catalogat entre 0 Hz i 20 KHz, però l'oïda humana només pot escoltar aproximadament entre 20 Hz i 20 KHz.

El to és una característica dels sons que permet saber si un so és més agut o més greu. Aquest concepte està relacionat de manera directa amb la freqüència. Quan una nota té un to greu hi ha menys cicles per segon, i si una nota té un to més agut, el nombre de cicles per segon és més elevat.

Els músics acostumen a parlar entre ells sobre la freqüència en termes de tons, on un conjunt d'aquests s'anomena escala musical. Aquestes persones s'entenen amb termes com acord en Sol - menor o escala de blues, però no sobre 440 Hz que equival a la freqüència que té la nota LA.

3.2 Representació digital del so

Quan es vol generar la representació de l'ona d'un to es fa mitjançant la fórmula següent:

$$f(t) = A \sin(2 \cdot \pi \cdot f \cdot t) \quad (1)$$

on A és l'amplitud, f és la freqüència i t és el temps.

A la Figura 2 es representa l'ona analògica i l'ona digital. En el cas de l'ona analògica, els canvis d'amplitud es fan suauament, mentre que en la digital es fan els canvis com si fos una escala.

El principal problema de la representació digital és com guardar un conjunt infinit de mostres d'amplitud per unitat de temps.

Per passar a un conjunt finit de mostres, el que es fa és guardar el valor de l'amplitud en certs instants de temps. La quantitat d'instants que es guarden per segon s'anomena freqüència de mostreig o *sampling rate*. La distància entre dues mostres consecutives serà sempre la mateixa (Figura 2).

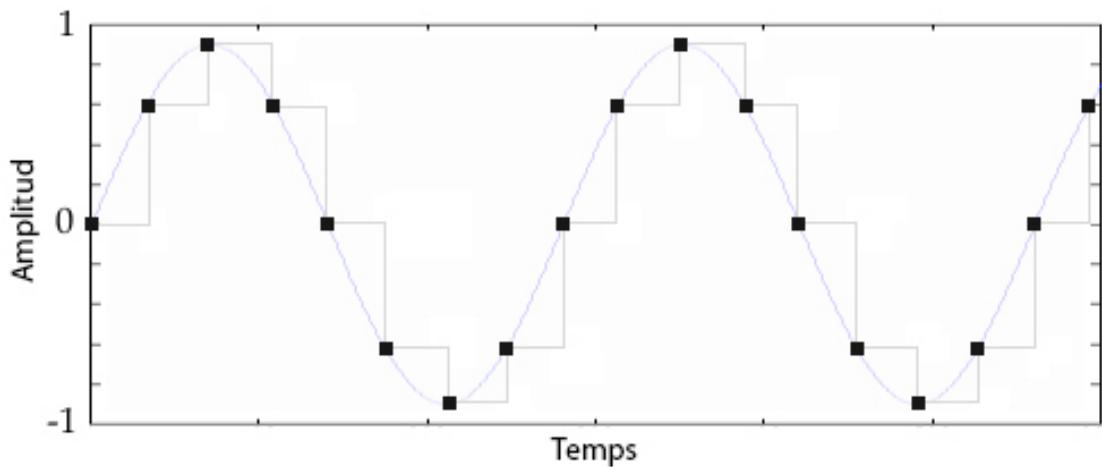


Figura 2: Ona anàlogica i digital. Imatge extreta de [16]

El procés de passar d'infinits valors d'amplitud d'entre -1 i 1 a un conjunt finit s'anomena quantificació [4]. Quan s'augmenta la quantitat de valors de la quantització fins a un nombre considerable, llavors s'obté una aproximació de l'ona en la qual l'oïda no trobarà diferència entre l'ona digital i l'analogica.

3.2.1 Teoria del mostreig

És necessari saber cada quant temps s'ha de guardar una mostra per obtenir una bona representació del so. El teorema de mostreig de Nyquist-Shannon [16, 17] resol aquesta incògnita. El teorema estableix que per representar adequadament un senyal ondulatori, la freqüència de mostreig necessària ha de ser almenys el doble de la freqüència més alta continguda en el so del senyal. Cal puntualitzar que, de vegades, el doble no és suficient i, per tant, cal un valor superior.

Per establir una idea de quines freqüències de mostreig són utilitzades, s'exposaran diversos casos o exemples [17].

- 8.000 Hz per la telefonia mòbil, comunicació sense fils i transmissió de micròfons sense fil.
- 16.000 Hz per la majoria de veu per IP (VoIP) moderna i productes de comunicació veu i vídeo per IP (VVoIP).
- 44.100 Hz per a àudios dels CD i per l'àudio amb format MPEG-1. Originalment triada per Sony per poder ser registrada en els seus equips.
- 96.000 Hz per a DVD-Audio, BD-ROM (Blu-ray Disc) pistes d'àudio, HD DVD (High-Definition DVD) pistes d'àudio. Aquesta freqüència de mostreig és el doble de l'estàndard de 48 kHz utilitzada amb àudio en equips professionals.

A continuació s'exposen dos exemples sobre la importància del mostreig.

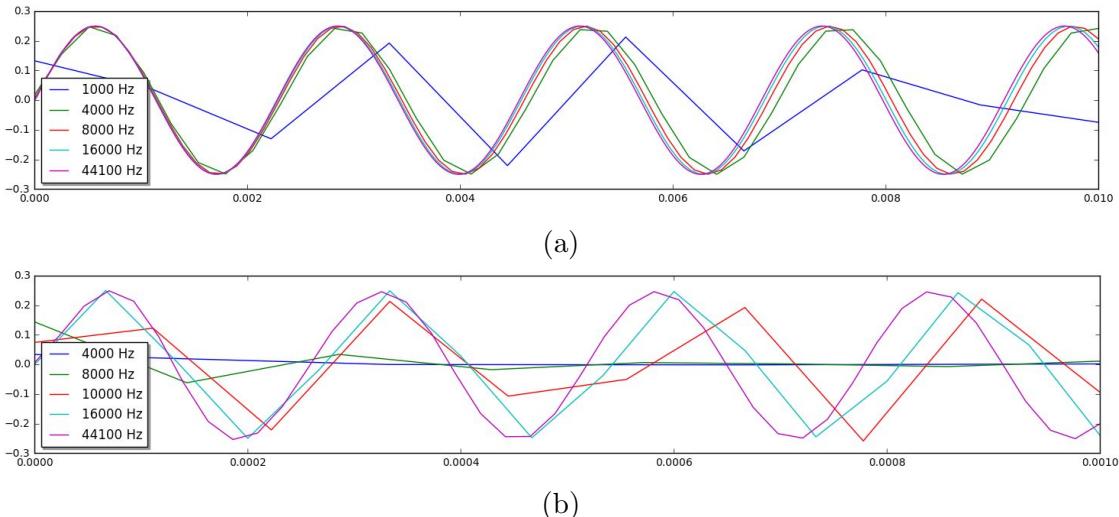


Figura 3: Representació de dues notes musicals en funció de la freqüència de mostreig. (a) la nota LA de la quarta octava i (b) la nota SI de la setena octava

A la Figura 3 s'observen dues notes mostrejades amb diverses freqüències. Per poder observar adequadament les representacions, l'interval de temps de la Figura

3b és menor que el temps de la Figura 3a. Això és degut a que la segona nota és més aguda que la primera i, per tant, el valor de la freqüència d'ona és més gran.

El teorema que s'ha comentat en aquesta secció estipula la freqüència de mostreig mínima per obtenir una bona representació. A les dues representacions es pot observar que les freqüències que estan per sota o al voltant del doble de la freqüència de la nota, no s'aproximen adequadament a la funció. En contraposició, aquelles freqüències que sobrepassen el doble de la freqüència acostumen a generar una bona representació de la nota.

3.3 Representació del so en imatges

La forma més usual de representar un so en imatges és mitjançant espectrogrames. Per poder obtenir-los és necessari tenir un petit bagatge sobre la sèrie i transformada de Fourier [16, 18].

3.3.1 Sèrie i transformada de Fourier

Una sèrie de Fourier en la disciplina de les matemàtiques és una sèrie infinita que convergeix puntualment a una funció periòdica i contínua. Aquest tipus de sèries poden representar qualsevol so a partir de les sumes infinites de funcions sinusoïdals. Aquesta sèrie s'expressa de forma general

$$f(t) = A_0 + \sum_{n=1}^{\infty} A_n \sin(2\pi * n\omega t) + \sum_{n=1}^{\infty} B_n \cos(2\pi * n\omega t) \quad (2)$$

Els dos termes A_n i B_n que accompanyen als sumatoris s'anomenen els coeficients de Fourier de la funció $f(x)$. Els coeficients de Fourier donen un conjunt de valors anomenats espectre del so. L'espectre indica la quantitat d'una mateixa freqüència que hi ha en el so.

Teòricament és possible agafar un so complex i descompondre'l en un conjunt d'ones sinusoïdals on cadascuna té una freqüència, amplitud i fase diferents. Per calcular-ho es fa una anàlisi de Fourier. Aquest procés d'analitzar un so basat en les seves ones sinusoïdals s'anomena la realització d'una transformada de Fourier.

3.3.2 Espectrograma

Un espectrograma de so és una visualització d'un senyal acústic, expressat mitjançant una gràfica tridimensional que representa com varia la freqüència i l'amplitud d'un senyal quan canvia el temps (Figura 4a). L'espectrograma s'obté calculant les transformades de Fourier en finestres temporals consecutives.

A l'hora d'aplicar la transformada de Fourier als espectrogrames, s'utilitza la transformada de Fourier amb finestra (STFT) [17]. Aquesta determina l'espectre i la fase d'un senyal sinusoïdal en seccions locals a mesura que canvia en el temps.

A la pràctica, consisteix en dividir el senyal en fragments més curts i calcular la transformada de Fourier en cada fragment.

Els espectrogrames representats en dues dimensions, són projeccions d'una successió de transformades de Fourier aplicades en finestres temporals consecutives, on l'energia i l'espectre del senyal varien al llarg del temps.

A la Figura 4 es mostren dos espectrogrames com a exemple.

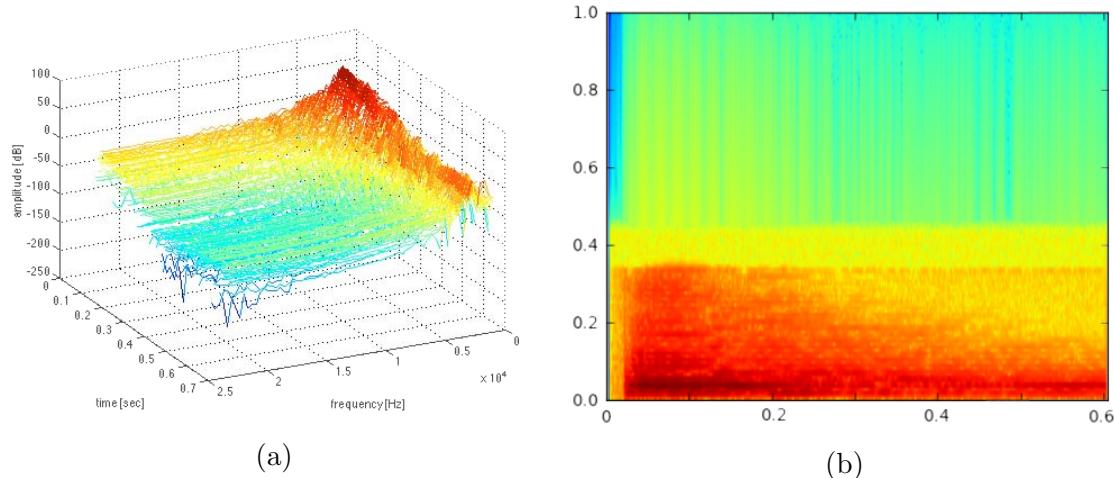


Figura 4: Representació (a) d'espectrograma en 3D d'una nota i (b) d'espectrograma en 2D. Imatge extreta de [16]

4 Xarxes Neuronals Artificials

4.1 Què són?

Les xarxes neuronals artificials (ANN) o xarxes neuronals [19, 20, 21] són un paradigma de programació inspirat en la biologia, que permet que un ordinador aprengui a partir de l'observació de dades. Es tracta d'un sistema de neurones artificials interconnectades que col·laboren entre si per produir un estímul de sortida a partir de les entrades.

L'objectiu de les xarxes neuronals artificials és resoldre els problemes de la mateixa manera que ho fa el cervell humà. A mesura que la investigació sobre el cervell avança, es coneixen nous patrons que s'intenten aplicar en les xarxes neuronals per aproximar la forma d'aprenentatge.

Aquest tipus de xarxes es poden utilitzar per una gran varietat de tasques, que són complicades de resoldre amb la programació típica, com la visió per computador [22] o el reconeixement de veu [23].

La primera xarxa neuronal va ser proposada per McCulloch i Pitts l'any 1943 [24]. Aquesta primera xarxa moderna era un model computacional d'activitat nerviosa, és a dir, era la modelització del comportament d'una neurona biològica. Aquesta unitat és la part essencial per construir una xarxa neuronal artificial. Aquest concepte ha servit d'inspiració pel desenvolupament d'altres arquitectures neuronals.

Dintre de les xarxes neuronals es pot fer una petita classificació segons el model en què es basen:

- **Models inspirats en la biologia:** Conjunt format per les xarxes que simulen els sistemes neuronals biològics i les funcions auditives i de visió.
- **Models artificials aplicats:** Conjunt de xarxes que no tenen gaires similituds amb els sistemes biològics. Les seves arquitectures estan dissenyades específicament per assolir l'objectiu que té la xarxa.

4.2 Representació del model

Per entendre com està estructurada una xarxa neuronal, s'ha d'explicar des de l'entitat més petita fins al conjunt que formen totes aquestes. Per poder entendre les xarxes neuronals, cal entendre la representació d'una neurona del cervell i el seu model matemàtic equivalent.

Una neurona biològica (veure Figura 5a) està formada per un cos cel·lular, un conjunt de cables d'entrada anomenats dendrites i un cable de sortida anomenat axó. Les dendrites són les que reben els impulsos i l'axó el que els reenvia a les altres neurones.

Simplificant el model biològic, la neurona es pot entendre com una unitat computacional que realitza càlculs a partir de les dades rebudes per les dendrites i envia els resultats obtinguts mitjançant l'axó.

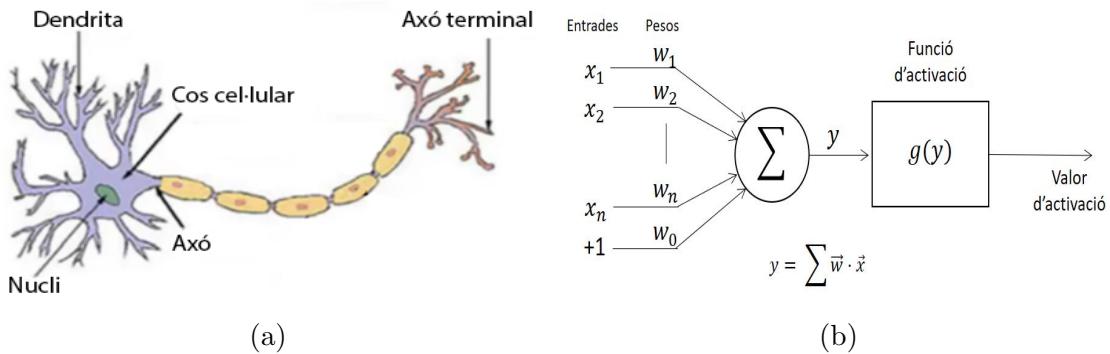


Figura 5: Representació de les neurones. (a) Model biològic i (b) Model matemàtic

A la Figura 5b es veu el model matemàtic d'una neurona, també anomenada unitat. Si s'agrupen els cables d'entrada en un vector \vec{x} , s'obtenen les dades d'entrada de la neurona. El nucli de la cèl·lula s'expressa com $g(y)$, on y és la suma del producte escalar de \vec{w} i \vec{x} on \vec{w} representa els pesos del sistema, $y = \sum \vec{w} \cdot \vec{x}$ i $g(x)$ és la funció d'activació. Així doncs, s'obté l'expressió que resumeix el comportament del model matemàtic de la neurona:

$$\overrightarrow{activacio} = g(\sum \overrightarrow{w} \cdot \overrightarrow{x}) \quad (3)$$

És important remarcar que els valors dels pesos w , o també anomenats paràmetres, aniran variant per tal que la ANN aprengui a resoldre el problema donat.

Una xarxa neuronal està formada per un conjunt finit de neurones. Quan les neurones estan separades en diversos subconjunts, s'anomenen capes. Cadascuna de les capes té un nom determinat depenent de quina sigui la seva posició.

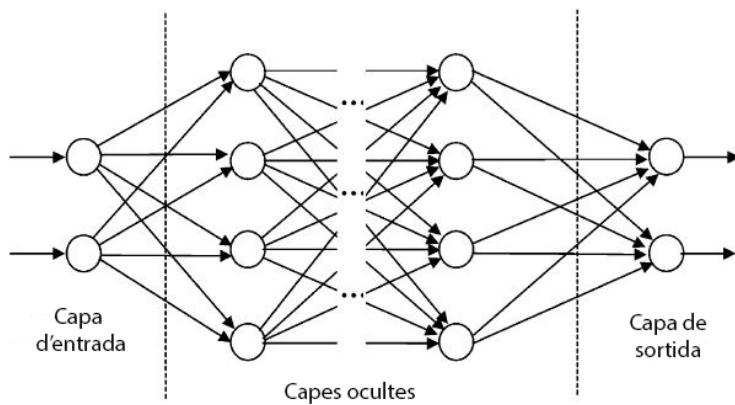


Figura 6: Capes d'una ANN

A la Figura 6, la primera capa s'anomena capa d'entrada o *input layer*, les capes del mig s'anomenen capes ocultes o *hidden layers* i l'última capa s'anomena capa de sortida o *output layer*.

El nombre de capes que pot tenir una xarxa neuronal defineix la complexitat i el cost computacional d'aquesta. En el cas de les xarxes que tenen més de dues capes, s'anomenen xarxes neuronals profundes. Les xarxes més simples no tenen capes ocultes, directament relacionen l'entrada amb la sortida.

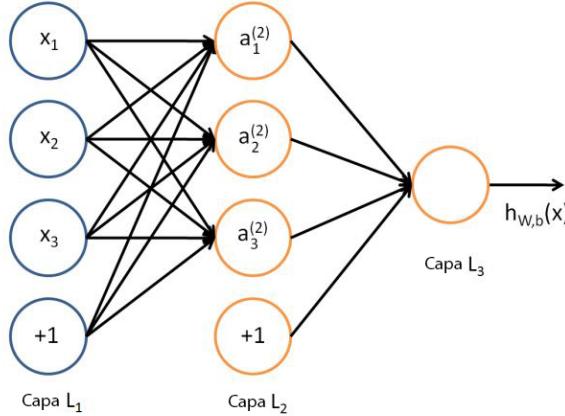


Figura 7: Notació de les unitats d'una ANN

A la Figura 7 es mostra una xarxa neuronal de tres capes, on les dues primeres capes estan formades per quatre unitats cadascuna, i la tercera capa té una única unitat.

A la capa d'entrada estan els nombres reals x_1, x_2, x_3 , que són els valors d'entrada i una neurona extra amb $+1$. A la capa intermitja els valors $a_1^{(2)}, a_2^{(2)}, a_3^{(2)}$, que són els valors d'activació i una neurona extra amb $+1$. La notació general que s'utilitza pels valors d'activacions és $a_i^{(j)}$ a la neurona i en la capa j i equival a l'equació (3). Les neurones amb valor $+1$ són anomenades valors d'oscil·lació, o més coneeguts com biaixos. Un biaix és una neurona que només té sortides i n'acostuma a haver una per cada capa.

Cada fletxa de la figura és un pes que relaciona una parella de neurones. El conjunt de fletxes que relacionen les neurones són representades com una matriu de pesos. Aquesta matriu conté els vincles de pesos de la capa j a la capa $j+1$ definida per $W^{(j)}$.

Els càlculs per obtenir els valors d'activacions de cada unitat de les capes dos i tres de la Figura 7 es realitzen tal com es mostra a continuació:

$$\begin{aligned}
 a_1^{(2)} &= g(W_{10}^{(1)}x_0 + W_{11}^{(1)}x_1 + W_{12}^{(1)}x_2 + W_{13}^{(1)}x_3) \\
 a_2^{(2)} &= g(W_{20}^{(1)}x_0 + W_{21}^{(1)}x_1 + W_{22}^{(1)}x_2 + W_{23}^{(1)}x_3) \\
 a_3^{(2)} &= g(W_{30}^{(1)}x_0 + W_{31}^{(1)}x_1 + W_{32}^{(1)}x_2 + W_{33}^{(1)}x_3) \\
 h_W(x) = a_1^{(3)} &= g(W_{10}^{(2)}a_0^{(2)} + W_{11}^{(2)}a_1^{(2)} + W_{12}^{(2)}a_2^{(2)} + W_{13}^{(2)}a_3^{(2)})
 \end{aligned} \tag{4}$$

S'ha de remarcar que els elements x_0 i $a_0^{(2)}$ són els biaixos. Per una altra banda, també hi ha els $W_{k,l}^{(j)}$ on la k i la l indiquen la relació entre les unitats de la capa j

i les unitats de la capa $j + 1$. En aquest cas la matriu té dimensió 3×4 , 3 de les unitats de la capa $j + 1$ i 4 de les unitats de la capa j tenint en compte el biaix.

En general, si la xarxa té s_j unitats a la capa j i s_{j+1} unitats a la capa $j + 1$, $W^{(j)}$ serà de dimensió $s_{j+1} \times s_j + 1$.

4.3 Funcions d'activació

Les funcions d'activació d'una neurona defineixen la sortida d'aquesta a partir de les entrades. Per a l'aprenentatge de les xarxes neuronals és necessari que aquestes funcions tinguin algunes de les següents propietats matemàtiques:

- No lineals: Aquesta propietat introduceix que les sortides no siguin combinacions lineals de les entrades. Al mateix temps evita que una xarxa amb múltiples capes es comporti com si només en tingüés una.
- Diferenciable infinit cops: Aquesta propietat permet utilitzar els mètodes d'optimització basats en el gradient.
- Rang: Quan el rang, o imatge de la funció, és finit, els mètodes basats en el gradient són més estables perquè el patró que segueixen les dades afecta només a un nombre limitat de pesos. Quan el rang és infinit, l'entrenament és més eficient perquè afecta a tots els pesos.

A continuació s'exposen les funcions d'activació més conegudes. Algunes d'elles també seran utilitzades al llarg d'aquest treball.

Lineal

La funció d'activació lineal (Figura 8a), és la més bàsica. Aquesta està definida per la identitat i per tant, el seu domini i el rang és la recta dels reals en els dos casos. Òbviament aquesta funció és C^∞ , i s'expressa com:

$$g(x) = x \tag{5}$$

Aquesta funció no satisfà la propietat de no linealitat, ja que quan múltiples capes utilitzen aquesta funció, tota la xarxa és equivalent a un model d'una sola capa. Quan es calcula la derivada s'obté $g'(x) = 1$.

Llindar Binari

La funció d'activació Llindar Binari (Figura 8b) està definida en $\mathbb{R} - \{0\}$, és a dir, no és contínua a l'origen. El seu rang està format pels punts $\{0, 1\}$.

$$g(x) = \begin{cases} 1 & \text{si } x \text{ és positiu} \\ 0 & \text{altres casos} \end{cases} \tag{6}$$

Aquesta funció no és diferenciable en $x = 0$ i la derivada en tots els altres punts és 0. Com a conseqüència d'aquest fet, no es poden aplicar mètodes del gradient amb aquesta funció d'activació.

Sigmoide

La funció sigmoide (Figura 8c) està definida en tota la recta dels reals. El rang de la funció és l'interval obert $(0, 1)$, és a dir, és una funció acotada. Aquesta està expressada mitjançant la fórmula:

$$g(x) = \frac{1}{1 + e^{-x}} \quad (7)$$

Aquesta funció d'activació és C^∞ i calculant la seva derivada s'obté

$$g'(x) = \frac{e^x}{(e^x + 1)^2} = \frac{1}{1 + e^{-x}} \left(1 - \frac{1}{1 + e^{-x}} \right) = g(x)(1 - g(x)) \quad (8)$$

Tangent Hiperbòlica

La funció d'activació de tangent hiperbòlica (Figura 8d) està definida en tota la recta dels reals. El rang és un interval obert $(-1, 1)$, és a dir, també és una funció acotada.

$$g(x) = \tanh(x) = \frac{e^t - e^{-t}}{e^t + e^{-t}} \quad (9)$$

També és C^∞ i calculant la seva derivada s'obté:

$$g'(x) = \operatorname{sech}^2(x) = 1 - \tanh^2(x) = 1 - g^2(x) \quad (10)$$

La principal diferència entre les funcions d'activació sigmoide i tangent hiperbòlica està en el fet que la segona té més pendent i més rang.

Unitats de Rectificació Lineal

La funció d'activació d'Unitats de Rectificació Lineal (ReLU) [25] (Figura 8e), és una evolució de la funció d'activació lineal, que millora l'aprenentatge de la xarxa. Una característica rellevant és l'augment de l'estabilitat del gradient, fet que permet aprendre més ràpidament. És per aquest motiu que, cada cop més, s'introduceix aquest tipus de neurona a les xarxes neuronals.

Aquesta funció està definida a tota la recta dels reals amb rang $[0, \infty)$ i és contínua, C^0 . La funció d'activació ReLU s'expressa com:

$$g(x) = \begin{cases} x & \text{si } x \text{ és positiu} \\ 0 & \text{altres casos} \end{cases} \quad (11)$$

en aquest cas s'inclou el zero dintre dels punts positius. La seva derivada és

$$g'(x) = \begin{cases} 1 & \text{si } x \text{ és positiu} \\ 0 & \text{altres casos} \end{cases} \quad (12)$$

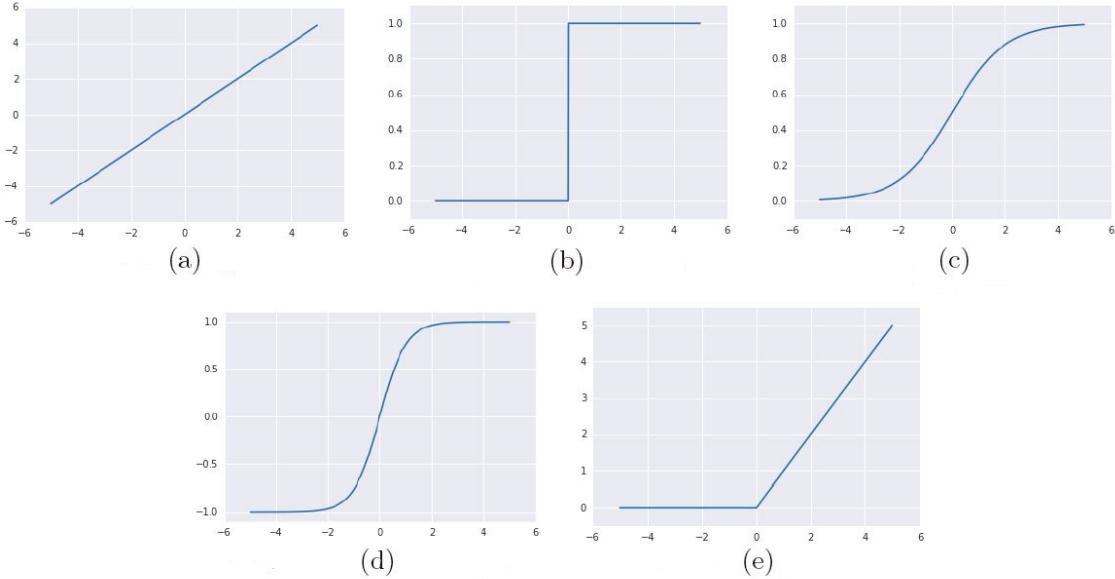


Figura 8: Representació gràfica de les funcions d'activació. (a) Lineal (b) Llindar Binari (c) Sigmoide (d) Tangent hiperbòlica (e) Unitats de Rectificació Lineal

4.4 Tipus d'aprenentatge

Totes les xarxes neuronals necessiten un conjunt de dades per a poder entrenar. Un conjunt de dades és una col·lecció d'informació relacionada i discreta a la qual es pot accedir de forma individual o en bloc. En la majoria dels casos aquesta col·lecció se separa en un conjunt d'entrenament i un conjunt de test. Amb el primer s'entrena la xarxa i amb el segon es comprova si la xarxa funciona.

Les xarxes neuronals es divideixen segons l'algorisme d'entrenament que utilitzen, en xarxes d'aprenentatge supervisat i aprenentatge no supervisat.

4.4.1 Aprendentatge supervisat

En l'aprendentatge supervisat [26], s'alimenta al sistema mitjançant un conjunt d'entrenament on ja es coneix quina serà la sortida correcta i com hauria de ser la relació entre l'entrada i la sortida.

Aquest tipus d'aprendentatge es divideix en la resolució de dos problemes diferents. El primer és el de regressió, en el qual tracta de predir els resultats d'una funció contínua [27, 28]. El segon problema és el de classificació, on es tracta de

predir els resultats d'una funció discreta, és a dir, catalogar l'entrada en diversos conjunts [29, 30].

Els següents exemples permeten entendre millor cadascun dels problemes anteriors:

- Regressió: A partir de la imatge d'una persona, predir l'edat que té aquesta.
- Classificació: Donat un pacient amb un tumor, decidir si és benigne o maligne.

4.4.2 Aprendentatge no supervisat

L'aprendentatge no supervisat [26] permet treballar amb sistemes dels quals no es coneix quina serà la sortida. Aquest tipus d'aprendentatge s'utilitza majoritàriament quan cal fer clústers o sistemes similars. Fer una agrupació o clusterització consisteix en agafar un elevat nombre dades i separar-les en grups diferents automàticament, sense haver fet la classificació prèvia o haver determinat els conjunts.

4.5 Tipus de xarxes neuronals

En el món de les xarxes neuronals existeixen diversos tipus amb característiques i finalitats concretes. Tot seguit s'exposa la xarxa neuronal més bàsica i alguna més que apareixerà al llarg de la memòria.

4.5.1 Xarxa de perceptró

La xarxa perceptró [31] és la més simple que existeix. Aquesta xarxa té dues cèl·lules d'entrada i una única cèl·lula de sortida. Aquest fet permet la implementació de portes lògiques.

La concatenació d'aquesta xarxa crea les xarxes *feedforward*. Aquestes es caracteritzen pel fet que no poden formar cap cicle i que les neurones de la capa j estan connectades només amb les neurones de la capa $j + 1$. Això provoca que la informació únicament viatgi en una direcció, des de l'entrada fins a la sortida. Les xarxes *feedforward* estan formades per L capes amb m cèl·lules d'entrada.

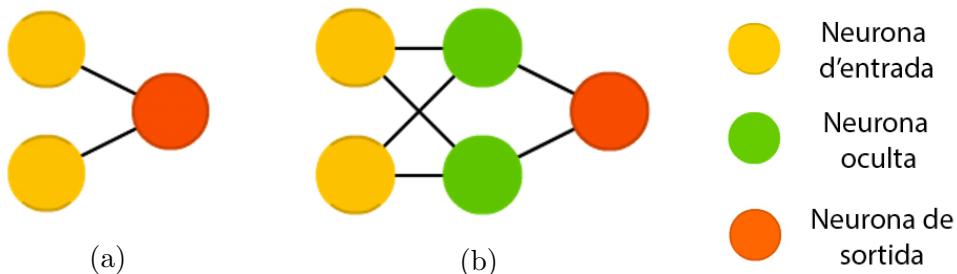


Figura 9: (a) Model d'un perceptró i (b) Model *feedforward*

4.5.2 Xarxes *fully connected* o *denses*

Les xarxes *fully connected* o també anomenades *denses* segueixen el model de les xarxes *feedforward*. Aquest tipus de xarxes estan formades per diverses capes, on totes elles són denses. La principal característica és que cada neurona està connectada a totes les neurones de la capa anterior. Un exemple d'aquest tipus de xarxes es troba en la Figura 7.

Un dels problemes principals d'aquest tipus de xarxes és la quantitat de connexions entre neurones, també anomenats pesos o paràmetres, que s'estableixen quan es defineix l'arquitectura. Aquest fet és evident quan es treballa amb valors emmagatzemats en format *float32*, on cada valor ocupa 4 bytes. Si la capa té $k \times l$ paràmetres, llavors necessita un espai a memòria de $k \times l \times 4$ bytes per una única capa. Si la xarxa té diverses capes de dimensions grans, llavors la capacitat necessària augmenta.

Un altre problema és el sobreajustament (*overfitting*) que sorgeix quan una xarxa entrena amb poques dades i aleshores tendeix a aprendre els pesos necessaris per tenir bons resultats en el conjunt d'entrenament. Però aquest fet pot provocar la pèrdua de la capacitat d'aprendre a generar bons resultats pel conjunt de test. Si s'augmenta la quantitat de capes i les seves dimensions, aquest problema es fa més greu i evident.

4.5.3 Xarxes convolucionals

Les xarxes convolucionals [30, 29] són multi-capa i segueixen el model de les xarxes *feedforward*, però la gran diferència es troba en la quantitat de paràmetres i en la connexió entre les neurones de les capes. Els diferents tipus de capes que s'utilitzen són: capa de convolució i capa de *pooling*.

Les xarxes convolucionals s'utilitzen generalment per reconèixer patrons visuals a les imatges, tot i que també es poden utilitzar per treballar amb ones de so. Un cas típic d'ús d'aquest tipus de xarxes és la classificació d'imatges.

Capa de convolució

Una capa de convolució és el bloc més important d'aquesta xarxa. El conjunt de paràmetres de la capa defineix els valors del filtre i aquests s'hauran d'aprendre durant l'entrenament. Aquests filters seran aplicats per totes les dades d'entrada realitzant l'operació de convolució en petites regions anomenades camp receptiu. Les dimensions dels filters acostumen a ser $k \times p$, on k i p són valors senars. Aquestes han de ser menors a la dimensió de les dades d'entrada de la capa. El nombre de paràmetres que caldrà entrenar corresindrà a la mida del filters i la quantitat de filters que es tinguin.

L'exemple d'una imatge de dimensions $n \times m$ permetrà entendre d'una manera més simple com funciona una capa convolucional (Figura 10). Sobre aquesta imatge s'aplica l'operació convolucional que consisteix en reemplaçar el valor del píxel després de multiplicar element a element els píxels de la imatge amb una màscara i sumar-los. Aquest procés es repetirà amb tots els píxels de la imatge.

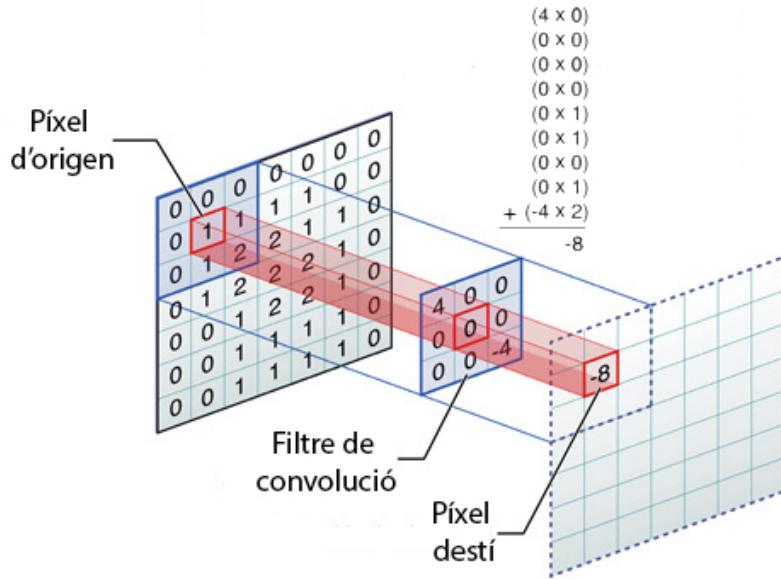


Figura 10: Capa de convolució amb filtre 3x3. Imatge extreta d'*Apple Developer*

Capa de Reducció o *Pooling*

Quan es treballa amb xarxes convolucionals s'acostuma a introduir capes de *pooling* entre les diferents capes de convolució. Aquestes capes redueixen la dimensió de la imatge de manera no lineal i disminueixen el nombre de paràmetres que la xarxa haurà d'entrenar. Aquest procés ha de mantenir les propietats espacials de la imatge i la relació entre els objectes que estiguin en ella.

Per poder aplicar *pooling* s'ha de dividir la imatge en rectangles que no se sobreposin i s'ha d'especificar quina funció es vol utilitzar: el màxim, el mínim, la mitja o un dels valors a l'atzar de la divisió. Normalment la mida del rectangle és de 2×2 .

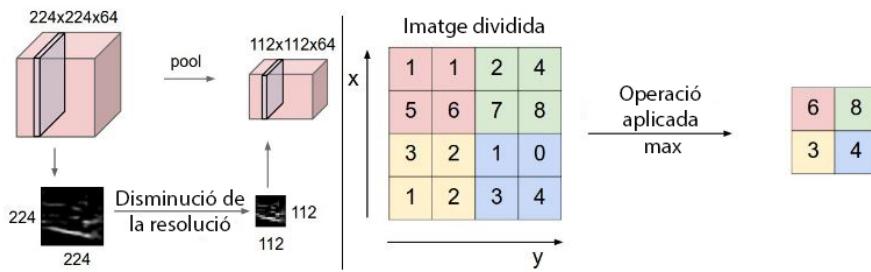


Figura 11: Exemple de capa de *pooling*. Es mostra com es redueix una imatge i com s'aplica amb una regió 2×2 l'operació de màxim. Imatge extreta de la Universitat de Stanford

Un exemple de l'aplicació de les capes de convolució i *pooling* es troba en la xarxa convolucional LeNet [29] (Figura 12). Aquesta xarxa està formada pel conjunt de

capes: convolució $(5 \times 5 \times 6)^2$, *pooling* $(2 \times 2)^3$, convolució $(5 \times 5 \times 6 \times 16)$, *pooling* (2×2) i tres capes denses. Aquesta xarxa reconeix els caràcters numèrics escrits a mà o a ordinador.

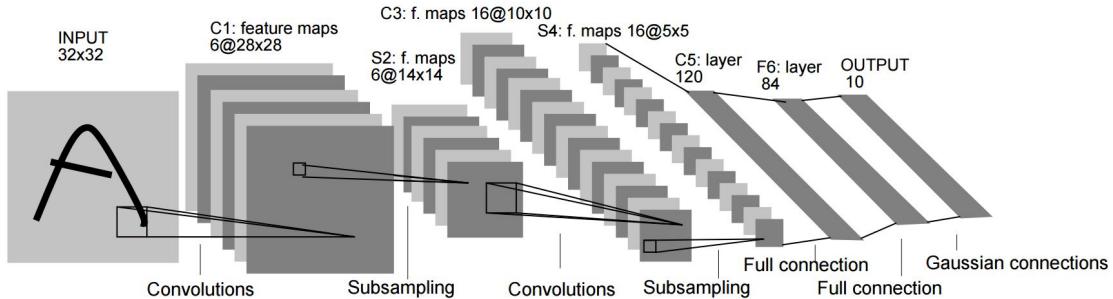


Figura 12: Arquitectura de la xarxa convolucional LeNet pel reconeixement de dígits. Imatge extreta de [29]

4.5.4 Autoencoders

Les xarxes anomenades *autoencoders* [33, 34] tenen l'objectiu de comprimir i descomprimir la informació de forma automàtica. Per tal d'arribar al seu objectiu intenten trobar una representació més simple de les dades introduïdes. Per aconseguir aquesta representació s'utilitza una xarxa neuronal densa que té la forma d'un rellotge de sorra. El nombre de neurones a la capa d'entrada i de sortida és el mateix però, a les capes del mig hi ha un nombre inferior de neurones. Així, doncs, la informació queda comprimida en la capa amb menor nombre de neurones. Aquest tipus de xarxes intenten aprendre la funció $h_W(x) \approx x$. La funció identitat pot semblar fàcil d'obtenir, però introduint limitacions en la xarxa, com la reducció del nombre de neurones per capa, es complica l'aprenentatge. Però és quan es genera un aprenentatge exitós que es poden descobrir estructures interessants sobre les dades.

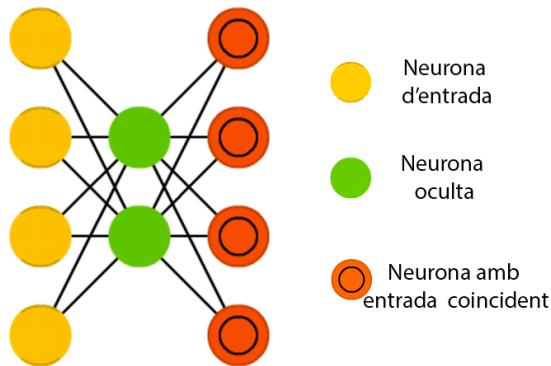


Figura 13: Representació d'una xarxa *autoencoder*

²Dimensió de la màscara de convolució.

³Dimensió de la regió de *pooling*.

Aquest tipus de xarxes es poden separar en dues parts. La part que va des de l'entrada a la capa del mig s'anomena part de compressió o *encoding* i la resta de la xarxa, és a dir, la part que va des de la capa del mig fins a la sortida s'anomena part de descompressió o *decoding*.

Les xarxes *autoencoders* s'utilitzen normalment per codificar i descodificar característiques de les dades, com poden ser píxels d'una imatge, amplituds d'una ona de so o propietats extretes d'un fitxer.

Quan s'entrenen aquests tipus de xarxes, es mira l'error com la diferència entre la sortida i el valor original d'entrada.

Un exemple molt comú és la classificació dels objectes d'una imatge. Computacionalment és molt costós treballar amb imatges d'alta resolució, però si es troba una representació més petita d'aquesta, es pot fer la classificació a partir de la representació. També es pot aplicar aquesta idea a la detecció de patrons en imatges i sons.

4.5.5 Xarxes recurrents

Les xarxes neuronals recurrents [6] es caracteritzen pel fet que les relacions de les neurones poden formar cicles. Aquestes xarxes permeten introduir dades amb diverses dimensions i treballar amb dades de manera seqüencial. Això provoca que es generi un estat en el qual la xarxa pot recordar informació temporalment. Per tant, és rellevant l'ordre d'entrada de les dades.

Aquest tipus de xarxes són especialment adients pel modelatge i generació del llenguatge, la traducció automàtica, el reconeixement de veu [23], etcètera.

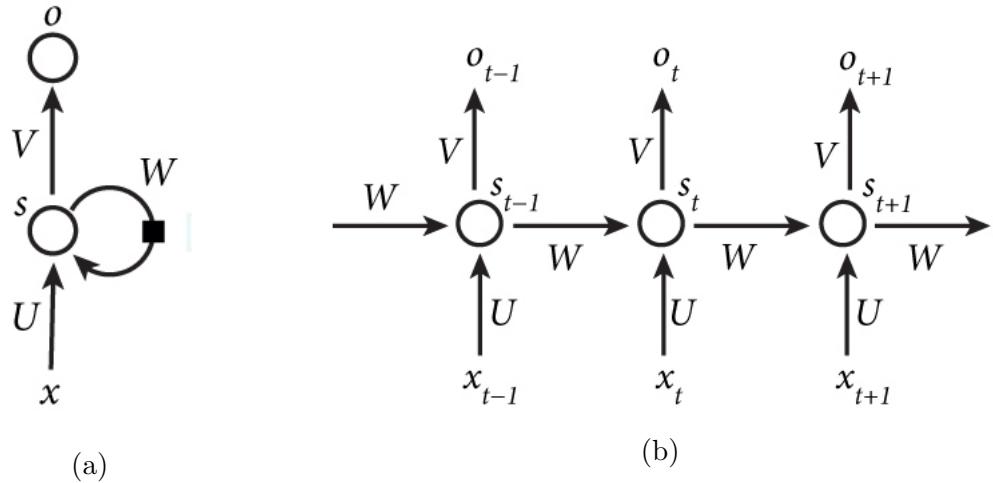


Figura 14: Xarxa Recurrent típica. (a) xarxa plegada i (b) xarxa desplegada

A la Figura 14 es mostra una xarxa recurrent. A l'esquerra està sense desplegar i a la dreta desplegada. A la figura s'observen els paràmetres i variables del sistema:

- x_t correspon a l'entrada a l'instant t .

- s_t correspon a la "memòria" o estat de la xarxa a l'instant t . Aquesta memòria guarda informació sobre els temps anteriors. Aquest estat es calcula mitjançant l'equació $s_t = \tanh(U \cdot x_t + W \cdot s_{t-1})$.
- o_t correspon a la sortida a l'instant t . Encara que es puguin calcular les sortides en cada instant de temps, no sempre és necessari. Per calcular aquest valor s'utilitza l'equació $o_t = \text{softmax}(V \cdot s_t)$ ⁴. Una altra notació que també s'utilitza és h_t .
- W, U, V corresponen als paràmetres que es comparteixen al llarg de tota la xarxa.

El major inconvenient d'aquest tipus de xarxes és l'explosió del gradient que provoca la pèrdua d'informació ràpidament en el temps. Aquest esdeveniment depèndrà també de la funció d'activació utilitzada.

Xarxa neuronal de memòria a llarg i curt termini

Per resoldre el problema anterior amb el gradient, s'han desenvolupat xarxes de memòria a llarg i curt termini (Long short term memory o LSTM) [7, 35]. Aquestes tenen la mateixa arquitectura que les xarxes recurrents, però varien el comportament de la unitat central. A la Figura 15 s'observa la diferència entre una unitat d'una xarxa recurrent estàndard i una xarxa LSTM.

Cada unitat de la xarxa té una cèl·lula de memòria i tres portes: entrada(i), sortida(o) i oblit(f).

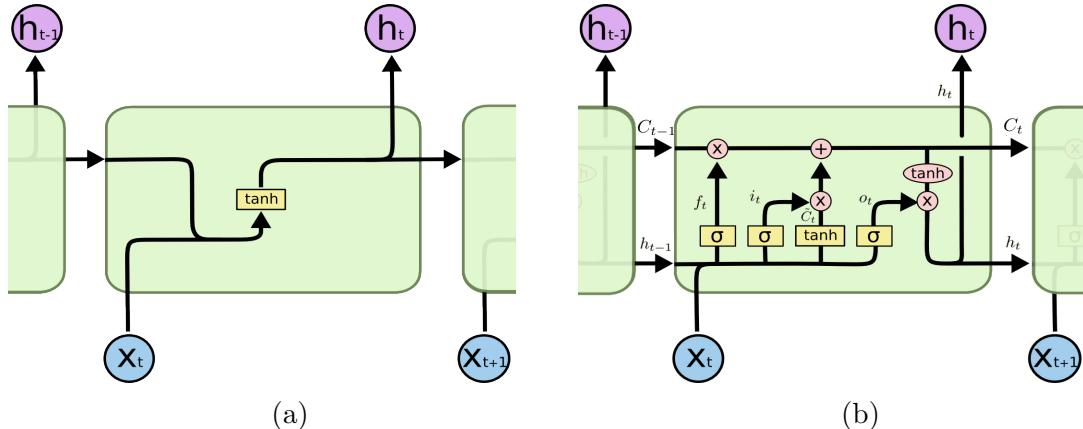


Figura 15: Representació d'un estat de la xarxa en (a) una xarxa recurrent (b) una xarxa LSTM. Imatge extreta de [35]

El primer pas que fa la cèl·lula és mirar quina informació ha d'oblidar següent l'equació següent:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \quad (13)$$

⁴Softmax regression és una generalització de la regressió logística pel cas on la sortida pot prendre diversos valors o classes.

on h_{t-1} és la sortida de l'estat anterior i x_t és l'entrada en l'instant t .

El següent pas és decidir quina informació es guarda a la cèl·lula

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \quad (14)$$

$$C'_t = \tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \quad (15)$$

on i_t escull quins valors s'actualitzaran i C'_t són els candidats a ser afegits a l'estat.

El següent pas actualitza l'estat de la cèl·lula

$$C_t = f_t \cdot C_{t-1} + i_t \cdot C'_t \quad (16)$$

L'últim pas decideix quina serà la sortida

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \quad (17)$$

$$h_t = o_t \cdot \tanh(C_t) \quad (18)$$

on h_t és la sortida de la xarxa en l'instant t .

4.6 Aprenentatge de la Xarxa

Per aprendre, les xarxes neuronals fan servir els valors dels pesos mitjançant l'algorisme anomenat *BackPropagation* [19, 20] que intenta disminuir la diferència entre la sortida generada per la xarxa i la sortida real. Els paràmetres s'optimitzen buscant el mínim global de la funció creada a partir del descens del gradient i la funció de cost.

4.6.1 Funció de cost

La funció de cost C [19] és l'error, també anomenat pèrdua del sistema, entre la resposta generada per la xarxa i la resposta desitjada. L'objectiu és que aquest valor sigui el més proper a zero.

Les funcions de cost per poder ser utilitzades per l'algorisme d'aprenentatge han de complir dues propietats:

- Expressar-se com una mitjana sobre la funció de cost de cadascun dels exemples d'entrenament, x .

$$C = \frac{1}{n} \sum_x C_x \quad (19)$$

Aquesta condició permet calcular el gradient respecte els pesos i els biaixos per cada mostra d'entrenament. Mitjançant aquest gradient es coneix la velocitat de canvi del cost.

- Expressar-se en funció de les sortides de la xarxa neuronal.

Normalment s'utilitzen dos tipus de funcions de cost, encara que n'existeixen més.

- **Cost Quadràtic:** També conegut com l'error quadràtic mitjà, de màxima versemblança o suma d'errors al quadrat:

$$C = \frac{1}{2} \|a^L - y\|^2 = \frac{1}{2} \sum_j (a_j^L - y_j)^2 \quad (20)$$

on a_j^L representa el valor d'activació de la neurona j a la capa L i y representa la sortida esperada. a^L és la capa de sortida.

El gradient d'aquesta funció de cost respecte a la sortida de la xarxa neuronal és el següent:

$$\nabla_a C = \left(\frac{\partial C}{\partial a_1}, \dots, \frac{\partial C}{\partial a_n} \right) = (a^L - y) \quad (21)$$

- **Cost de l'Entropia Creuada:** Normalment utilitzada per resoldre problemes de classificació.

$$C = - \sum_j (y_j \ln(a_j^L) + (1 - y_j) \ln(1 - a_j^L)) \quad (22)$$

El gradient d'aquesta funció de cost respecte la sortida de la xarxa neuronal:

$$\nabla_a C = \frac{(a^L - y)}{(1 - a^L)(a^L)} \quad (23)$$

4.6.2 Descens del gradient

El descens del gradient és un mètode que iterativament disminueix el valor de la funció. En aquest cas s'optimitzaran els paràmetres de la funció de cost.

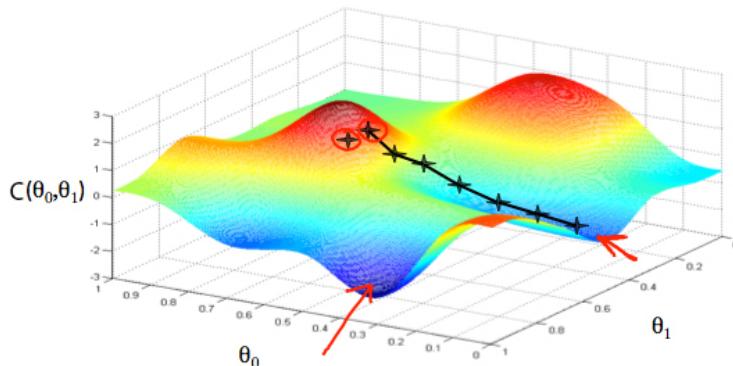


Figura 16: Representació gràfica de la funció de cost

A la Figura 16 s'observa la gràfica de la funció de cost, on els paràmetres θ_0 i θ_1 són els eixos x, y i la funció és l'eix z . Quan es comença a entrenar la xarxa els

paràmetres poden tenir qualsevol valor i acostumen a estar molt lluny del mínim global. Per tal d'arribar fins aquest punt és necessari definir la direcció i la velocitat en què es mouran per la gràfica.

La velocitat està definida per una variable donada, normalment per l'usuari, que s'anomena taxa d'aprenentatge o *learning rate*, α .

La direcció s'obté utilitzant el càlcul del gradient que determina en quin sentit hi ha més diferència de la funció de cost. Per tant, si es pren la direcció oposada a la del gradient, s'estarà movent cap a un valor de la funció de cost inferior. En aquest cas el gradient de la funció de cost seria:

$$\nabla C = \left(\frac{\partial C}{\partial \theta_0}, \frac{\partial C}{\partial \theta_1} \right) \quad (24)$$

4.6.3 Algorisme *BackPropagation*

BackPropagation és el nom que rep l'algorisme de les xarxes neuronals per sistemes d'aprenentatge supervisat. L'objectiu d'aquest algorisme és minimitzar la funció de cost fent servir els paràmetres (pesos i biaixos) òptims. Aquest algorisme modifica els valors dels pesos però no canvia la funció d'activació.

Aquest procediment es divideix en dues parts:

Propagació cap endavant

Primerament cal alimentar la xarxa amb els paràmetres i funcions necessàries. D'aquesta forma s'obté la sortida generada pel sistema.

Propagació cap enrere

En aquest pas es calcula l'error a partir de la sortida de la xarxa i el valor esperat. Posteriorment aquest error es propagarà proporcionalment des de la sortida fins l'entrada actualitzant els valors dels paràmetres. Realitzant aquest procés iterativament s'aconseguirà reduir l'error mitjançant el descens del gradient. Aquest concepte es basa en la idea que cada neurona assumeix una part de culpa en l'error total.

Per reduir C s'utilitza el càlcul del gradient respecte dels pesos individuals $\frac{\partial C}{\partial \omega_{i,j}}$ mitjançant la regla de la cadena

$$\frac{\partial C}{\partial \omega_{i,j}} = \frac{\partial C}{\partial \text{Sortida}_j} \cdot \frac{\partial \text{Sortida}_j}{\partial \text{Capa}_j} \cdot \frac{\partial \text{Capa}_j}{\partial \omega_{i,j}} \quad (25)$$

Quan es treballa en l'última capa i suposant la funció d'activació $g(x)$ s'obté

$$\frac{\partial \text{Sortida}_j}{\partial \text{Capa}_j} = \frac{\partial g(x)}{\partial x} = g'(x) \quad (26)$$

En cas que es vulgui conèixer l'error d'una capa que està al mig de la xarxa, l'error vindrà donat per l'acumulació d'errors de les altres capes.

$$\frac{\partial C}{\partial \text{sortida}_j} = \sum \frac{\partial C}{\partial \text{Sortida}_j} \cdot \frac{\partial \text{Sortida}_j}{\partial \text{Capa}_j} \quad (27)$$

Per últim, per actualitzar els paràmetres, es fa servir el mètode del descens del gradient utilitzant la taxa d'aprenentatge α i el gradient segons l'equació

$$\Delta\omega_{i,j} = -\alpha \cdot \frac{\partial C}{\partial \omega_{i,j}} \quad (28)$$

Normalment s'utilitza una versió de l'algorisme del descens del gradient anomenada descens del gradient estocàstic per tal de poder aproximar els paràmetres sense la necessitat d'introduir totes les dades en el mateix instant.

Alguns optimitzadors més complexes utilitzen el concepte del **momentum** que preveu que el descens del gradient tendeixi a un mínim local.

4.6.4 Tècnica d'aprenentatge *Dropout*

Dropout és una tècnica que soluciona el problema del sobreajustament explícit anteriorment. Aquesta tècnica desactiva algunes neurones fent que el seu valor d'activació no es pugui utilitzar per alguns conjunts d'entrenament. Per tal d'escol·lir quines neurones es desactiven, se segueix una distribució uniforme al llarg de la capa. S'acostuma a desactivar entre un 10% i un 70% de les neurones de la capa.

Aquesta desactivació disminueix el sobreajustament perquè obliga a la xarxa a buscar patrons més generals i no tan específics.

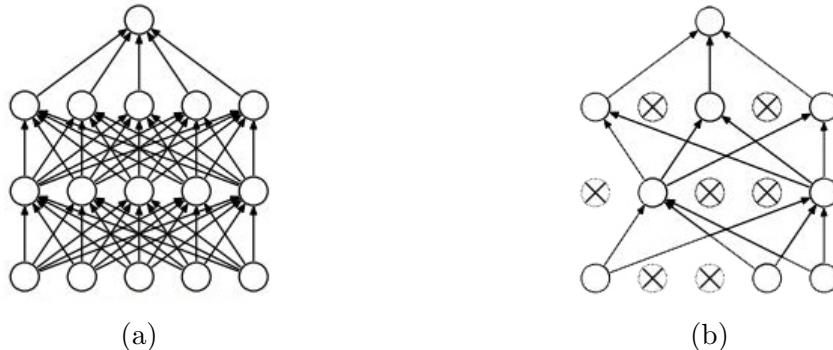


Figura 17: Representació (a) d'una xarxa estàndard i (b) d'una xarxa després d'aplicar *dropout*

5 Framework del Treball

Fins ara tot el que s'ha exposat ha estat la part teòrica del treball. Ara és el moment d'escollar quines eines s'utilitzaran per implementar les arquitectures necessàries per arribar a les nostres fites. L'ús de les llibreries permetrà utilitzar millor els recursos del sistema i reduir el temps en la implementació.

El *framework* haurà de tenir una llibreria amb un gran nombre d'implementacions per poder generar les arquitectures necessàries. També haurà d'aprofitar tots els recursos disponibles del sistema per minimitzar els temps d'entrenament.

En aquesta secció s'introduiran diversos *frameworks* coneguts per la implementació de xarxes neuronals. Tot seguit es comentarà breument quin d'aquests s'ha escollit per dur a terme aquest treball i el motiu de la decisió.

5.1 Theano

Theano [11] és una llibreria de codi obert pel càlcul numèric en llenguatge Python. Ha estat desenvolupat principalment per un grup d'investigació de la Universitat de Montreal.

És una llibreria que s'utilitza per definir, optimitzar i avaluar expressions matemàtiques que contenen matrius. Els càlculs s'expressen de manera similar a les operacions amb la llibreria NumPy, paquet per la computació científica. Theano executa les arquitectures de manera eficient en *central processing unit* (CPU) i *graphics processing unit* (GPU) i calcula automàticament les derivades de les funcions d'activacions, sigui amb una o diverses variables.

5.2 TensorFlow

Tensorflow [8] és una llibreria creada per Google ideada per la computació numèrica utilitzant grafs. Aquesta llibreria és similar a Theano però destaca per diverses propietats:

- Flexibilitat: Només és necessari escriure la xarxa com un gràfic o com un flux de dades. Proporciona eines que faciliten la implementació de parts comunes de xarxes neuronals, tot i que els usuaris poden crear les seves pròpies llibreries a alt nivell.
- Portabilitat: Es pot executar tant en CPU com en GPU sense necessitat de canviar el codi.
- Autodiferenciació: genera el càlcul de la derivada automàticament al introduir una nova capa o operació.
- Llenguatges de Programació: es pot utilitzar tant Python com C++ per la construcció i execució dels grafs computacionals.

- Rendiment: Permet utilitzar tota la potència del sistema informàtic agrupant diversos recursos, o només utilitzar una part d'aquests.

5.3 Keras

Keras [10] és una llibreria per programar xarxes neuronals creada per Francois Chollet, un enginyer de Google. Aquesta llibreria està programada sobre Theano i TensorFlow proporcionant un entorn fàcil i ràpid a l'hora de crear prototips. Amb aquesta llibreria només es poden construir xarxes convolucionals i xarxes recurrents. De la mateixa manera que les anteriors, es pot executar tant en CPU com en GPU. Els principis que segueix aquesta llibreria són:

- Modularitat: Els models s'entenen com un graf on totes les parts es poden configurar i connectar amb altres parts. Les capes, neurones, funcions d'activacions i optimizadors són mòduls independents i es poden combinar per crear nous models.
- Minimalisme: Cada mòdul és simple i breu i s'ha d'entendre fàcilment.
- Extensibilitat: Afegir nous mòduls és simple.
- Python: Tots els models estan programats en Python, que és compacte, fàcil de depurar i fàcil d'estendre.

5.4 Discussió

Keras és una de les llibreries més utilitzades i més fàcil d'entendre i fer servir en l'actualitat, però és més interessant tenir les opcions de crear tot a baix nivell i no dependre dels mòduls ja creats. Això fa més fàcil adaptar-los a les idees i arquitectures que es desitgin implementar. Com a conseqüència d'aquests fets, es va decidir no utilitzar-la com a llibreria en aquest projecte.

L'elecció de TensorFlow sobre Theano ve donada per dos motius. El primer és que TensorFlow té implementat de manera eficient els càlculs en GPU en paral·lel i en canvi Theano a l'inici d'aquest treball no els tenia. El segon motiu és que TensorFlow està sent més utilitzat i està prenent més força en el camp de l'aprenentatge automàtic i les xarxes neuronals.

6 Representació del so

En aquesta secció es vol resoldre el problema de trobar una nova representació del so. En aquesta memòria s'han vist dos tipus de representacions possibles: l'ona de so i l'espectrograma.

L'objectiu d'aquesta nova representació és que estigui formada per un vector de característiques de llargada n on la informació de l'ona estigui comprimida. Però, a priori, no és possible conèixer la forma d'aquest vector. Per obtenir una compressió de les dades, les xarxes neuronals *autoencoders* són idònies perquè tenen almenys una capa amb dimensió menor que la capa d'entrada. Aprofitant aquesta propietat de la xarxa, s'obtindrà el vector de característiques de menor dimensió que les dades d'entrada.

Per tal d'obtenir la representació, s'haurà d'entrenar aquest tipus de xarxa i aconseguir que l'entrada i la sortida siguin el més semblants possibles. Quan es compleixi això, la capa del mig tindrà un bon conjunt de característiques de les dades i aquestes seran la nova representació.

Si s'obté una bona representació, aquest vector de característiques podrà ser utilitzat per treballar amb altres xarxes neuronals. Això és conseqüència de que s'expressarà la mateixa informació en menys espai, reduint la quantitat de neurones que la xarxa necessitarà. Alguns exemples de l'aplicació d'aquesta representació són classificadors i detectors de sons.

6.1 Arquitectura

En aquest apartat es proposen dues metodologies per trobar el vector de característiques simplificant les ones de so. En la primera metodologia es treballarà amb ones de so i, en la segona, amb els espectrogrames.

6.1.1 Mètode 1: Representació amb l'ona de so

En aquest primer mètode es treballa directament amb l'ona de so. La intenció és fer una compressió amb l'ona i que en fer la descompressió s'obtingui la mateixa o una aproximadament igual. Quan això succeeixi s'obtindran els vectors de característiques adequats.

El procés de reconstrucció d'un senyal és el mateix per qualsevol durada d'ona, és a dir, la reconstrucció és igual per un segon de so que per una hora. És per aquest motiu que es considera que una entrada de dimensió 1024 amb una freqüència de mostreig de 16000 Hz és suficient per trobar una representació acurada del senyal.

L'arquitectura utilitzada per fer aquest *autoencoder* és una xarxa densa amb una capa d'entrada, cinc capes ocultes i una capa de sortida (Figura 18). La capa d'entrada i la capa de sortida han de tenir la mateixa quantitat de neurones per tal de recuperar la mateixa mida de l'àudio. Les cinc capes ocultes són denses i s'han generat amb 900, 700, 512, 700, 900 neurones respectivament. La xarxa

utilitza la tècnica del *dropout* per evitar el sobreajustament. La informació digital quan es comprimeix per sota de la meitat perd informació, que posteriorment no serà possible recuperar. És per aquest motiu que per a la representació a partir de l'ona de so s'utilitza un mínim de 512 característiques per cada 1024 del fitxer original. En alguns casos específics, es podria estudiar com reduir la quantitat de característiques per representar l'ona de so.

Com a conseqüència de que aquesta xarxa és densa s'obté fàcilment el nombre de paràmetres del sistema, amb l'equació $s_{j+1} \times s_j + 1$. Per tant, si se substitueixen les mides de la capa a l'equació anterior s'obté:

$$\begin{aligned}
& (1024 + 1) \times 900 + (900 + 1) \times 700 + (700 + 1) \times 512 + \\
& (512 + 1) \times 700 + (700 + 1) \times 900 + (900 + 1) \times 1024 \\
& = 922\,500 + 630\,700 + 358\,912 + 359\,100 + 630\,900 + 922\,624 \\
& = 3\,824\,736
\end{aligned}$$

Per tant, s'entrenen 3 824 736 paràmetres per trobar una representació del so. Pel que fa a la funció d'activació, s'ha implementat la sortida de cada capa amb ReLU's. Referent a l'algorisme d'aprenentatge, s'ha utilitzat l'optimitzador d'Adam [36] amb una taxa d'aprenentatge de 10^{-3} . Aquest algorisme està basat en el concepte de *BackPropagation* i està completament implementat en la llibreria de TensorFlow.

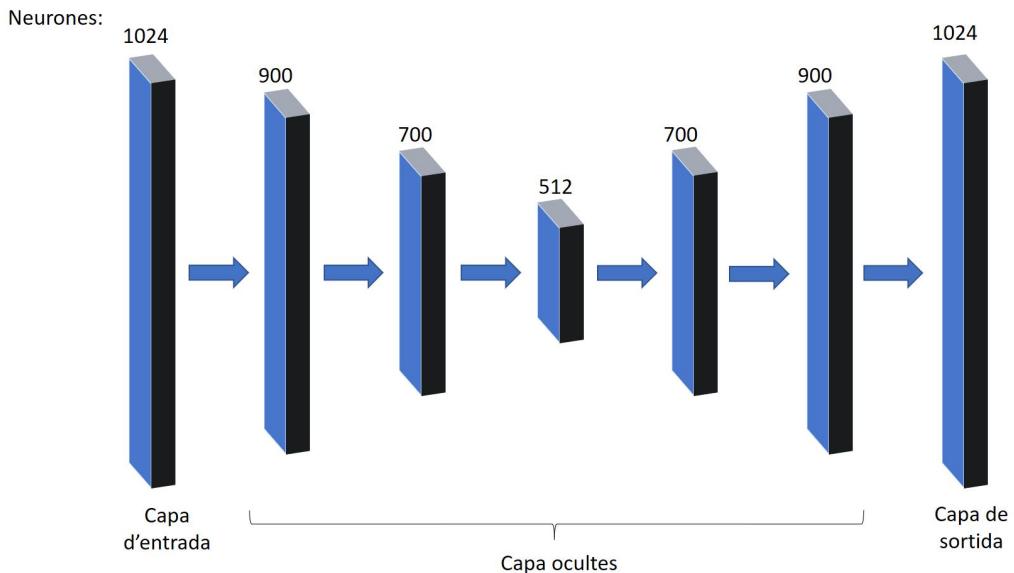


Figura 18: Arquitectura utilitzada per la representació del so en forma d'ona

6.1.2 Mètode 2: Representació amb l'espectrograma

En aquest segon mètode es treballa amb una representació visual de la imatge, l'espectrograma. Aquest procediment requereix la transformació d'una ona de so a un espectrograma, convertint les 1024 mostres a una matriu de 129×7 valors. La dimensió de la matriu és conseqüència del paràmetre que determina la mida de la finestra utilitzat en l'algorisme de càlcul. L'espectrograma es calcula mitjançant la funció *specgram* que pertany a la llibreria *matplotlib.mlab*. Com que la xarxa utilitzada és un *autoencoder*, les dimensions de l'entrada i la sortida hauran de tenir $129 \times 7 = 903$ neurones. Llavors el procés de compressió i descompressió serà semblant al mètode anterior, però els resultats obtinguts seran en forma d'imatge.

L'arquitectura (Figura 19) és molt similar a la del cas anterior. Però cal modificar la quantitat de neurones de les capes d'entrada i de sortida per tal d'adaptar-se al nou nombre de neurones. En resum, hi ha les següents neurones per capa: 903, 900, 700, 512, 700, 900, 903. Els paràmetres que s'han d'entrenar es calculen de la mateixa forma que en el mètode anterior, obtenint un total de 3 606 815 paràmetres. Referent a funcions d'activació, optimitzadors, taxa d'aprenentatge, etcètera. són els mateixos. Considerant condicions similars es podrà fer una comparació més equilibrada entre el vector de característiques del mètode anterior i d'aquest.

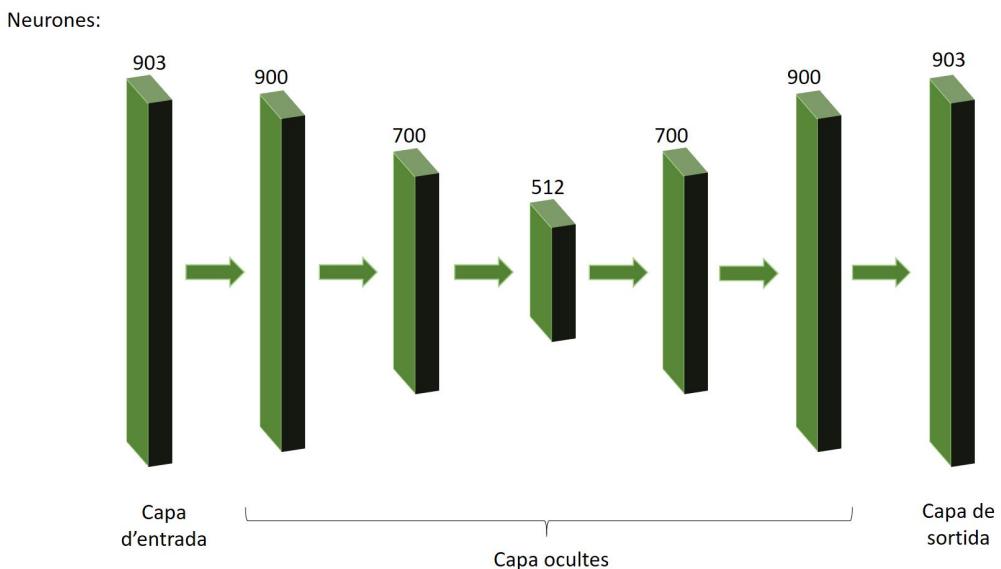


Figura 19: Arquitectura utilitzada en representació amb espectrograma

6.2 Conjunt de dades

Per tal de que les xarxes entrenin i aprenguin es requereixen conjunts de dades. Tant pel mètode de representació de l'ona de so com la representació del espectrograma s'utilitzen els mateixos conjunts de dades: notes musicals sintètiques i sons corresponents a un partit de tennis.

6.2.1 Notes musicals sintètiques

El primer conjunt de dades correspon a una col·lecció de notes musicals. Aquestes notes han estat generades de manera sintètica, és a dir, a partir de la freqüència que correspon a cada nota. Aquest conjunt està format per una escala cromàtica⁵ (Figura 20) a 5 octaves diferents.



Figura 20: Escala Cromàtica

A mesura que les notes són més agudes, la freqüència d'aquestes també augmenta. Per a incrementar la quantitat de mostres del conjunt de dades s'han creat combinacions de dues i tres notes.

Tots els àudios que formen aquest conjunt tenen una duració de mig segon i una freqüència de mostreig de 16000 mostres per segon.

A la Figura 21 es mostren 6 exemples de les notes sintètiques generades en aquest conjunt de dades. Els gràfics estan ampliats per tal de poder apreciar la freqüència de cadascuna de les notes i les combinacions de diverses notes.

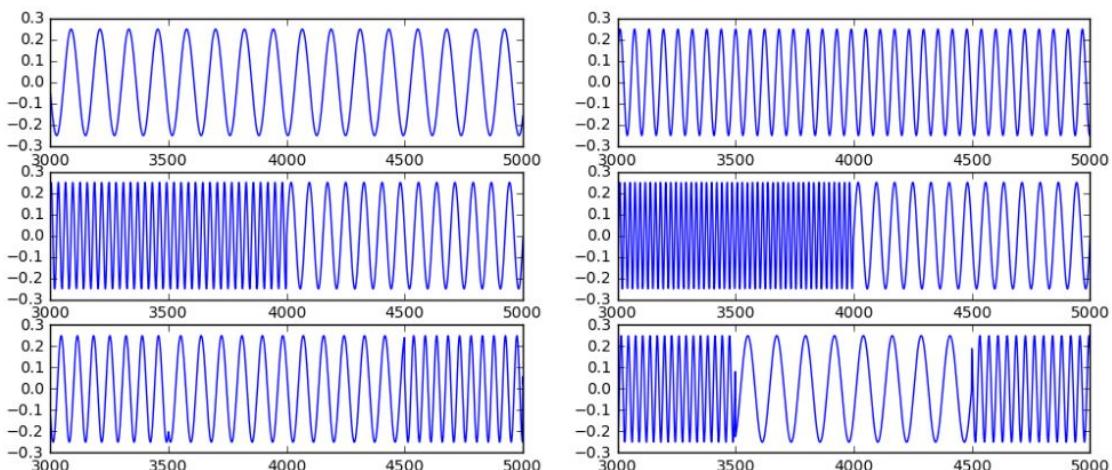


Figura 21: Sis exemples de notes sintètiques

En una representació d'ona de so convencional es representa l'amplitud en funció del temps. En contraposició a aquesta, els diversos exemples mostrats (Figura 21) representen en l'eix horitzontal la quantitat de mostres i en l'eix vertical l'amplitud de cada mostra. En aquestes figures es mostren 0,125 segons.

⁵Una escala cromàtica és una seqüència de dotze notes separades per mig to.

6.2.2 Sons corresponents a un partit de tennis

El segon conjunt de dades utilitzat per entrenar s'ha creat a partir dels sons que envolten els partits de tennis. Per crear-ho s'han escollit quaranta vídeos de *YouTube* i s'ha extret el so fragmentant-los en àudios de dos segons. Aquests vídeos es podran trobar en l'enllaç a *Github* indicat a la introducció. En els diferents àudios es pot escoltar l'impacte de la pilota amb la raqueta de tennis, les sabates dels jugadors amb el terra, el públic i altres esdeveniments que succeeixen durant els partits.

A la Figura 22 s'observa que els àudios ja no són tan perfectes com en el cas anterior.

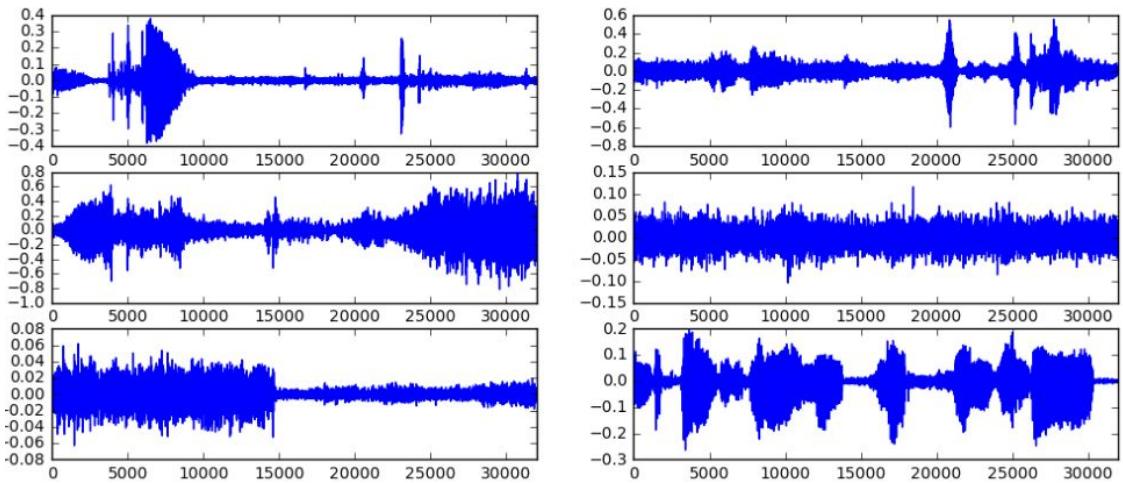


Figura 22: Sis exemples de sons de partits de tennis

6.3 Resultats obtinguts i discussió

6.3.1 Mesura de la qualitat

Els algorismes d'aprenentatge típics utilitzen l'error quadràtic com la funció de cost. Però per aquesta aplicació, l'error quadràtic no representa adequadament l'error en la compressió i descompressió. És per aquest motiu que per mostrar els resultats es faran servir l'error quadràtic i una altra mesura de qualitat, la relació senyal-soroll (*Signal-to-noise ratio* o *SNR*).

La relació senyal-soroll és una mesura que compara els nivells del senyal desitjat amb el nivell de soroll (senyal desitjat menys senyal generat). Es defineix com la relació de la potència d'un senyal (informació significativa) i la potència de soroll de fons (senyal no desitjada):

$$SNR = \frac{P_{\text{senyal}}}{P_{\text{soroll}}} \quad (29)$$

on P és la potència mitjana. Si els senyals es mesuren de la mateixa manera, llavors

el càlcul del SNR es pot fer mitjançant

$$SNR = \frac{P_{senyal}}{P_{soroll}} = \left(\frac{A_{senyal}}{A_{soroll}} \right)^2 \quad (30)$$

on A és la mitjana de l'arrel quadrada de l'amplitud. Per expressar aquest valor en decibels, es fa la següent conversió:

$$P_{senyal,dB} = 10 \log_{10}(P_{senyal}) \quad (31)$$

$$A_{senyal,dB} = 10 \log_{10}(A_{senyal}) \quad (32)$$

De forma equivalent,

$$SNR_{dB} = 10 \log_{10}(SNR) \quad (33)$$

I si es vol expressar el valor en funció de les amplituds

$$SNR_{dB} = 10 \log_{10} \left(\frac{A_{senyal}}{A_{soroll}} \right)^2 = 20 \log_{10} \left(\frac{A_{senyal}}{A_{soroll}} \right) \quad (34)$$

Per entendre el que representa aquest valor, s'exposaran tres exemples diferents. En aquests es compara la mesura de l'error quadràtic i la relació senyal-soroll.

A la Figura 23 s'observa que el senyal original i el generat són iguals, aleshores no hi ha soroll. És per aquest motiu que fent el càlcul de les mesures s'obté zero amb l'error quadràtic i infinit per la relació senyal-soroll.

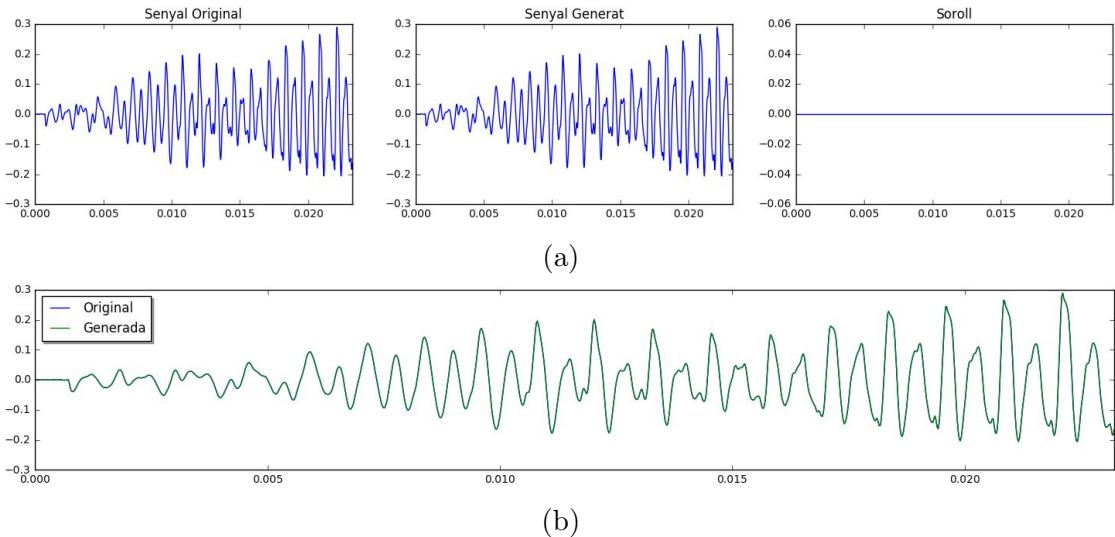


Figura 23: Exemple 1. (a) Comparació dels senyals original, generat i el soroll. (b) Superposició de l'ona original i generada.

Error Quadràtic = 0.0 i $SNR = +\infty$ dB

A la Figura 24a s'observa que el senyal original i el generat no són idèntics però sí similars. És per aquest motiu, que el senyal de soroll no és zero. Fent els càlculs de les mesures s'obté $5,48 \cdot 10^{-4}$ amb l'error quadràtic i 23.33 dB amb la relació

senyal-soroll. Tal com es pot veure a la Figura 24b les dues ones superposades no són coincidents.

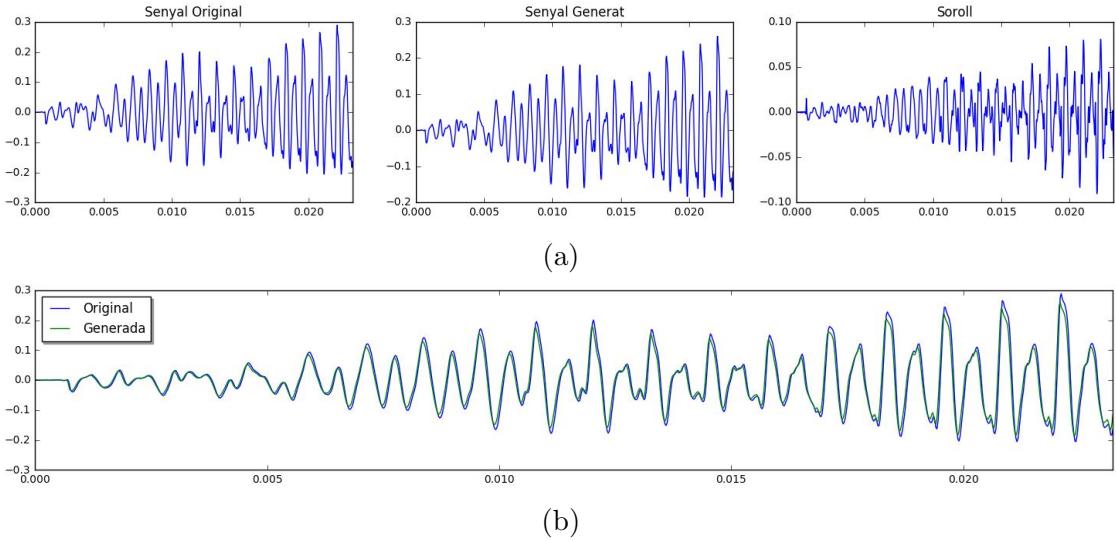


Figura 24: Exemple 2. (a) Comparació dels senyals original, generat i el soroll. (b) Superposició de l'ona original i generada.

Error Quadràtic = $5,48 \cdot 10^{-4}$ i $SNR = 23.33$ dB

Per últim, a la Figura 25, les ones originals i generades són completament diferents. Calculant les mesures s'obté 0.014 amb l'error quadràtic i -5.052 dB amb la relació senyal-soroll. El valor negatiu amb la segona mesura significa que el soroll és més gran que el senyal original i en conseqüència el so generat és diferent a l'original.

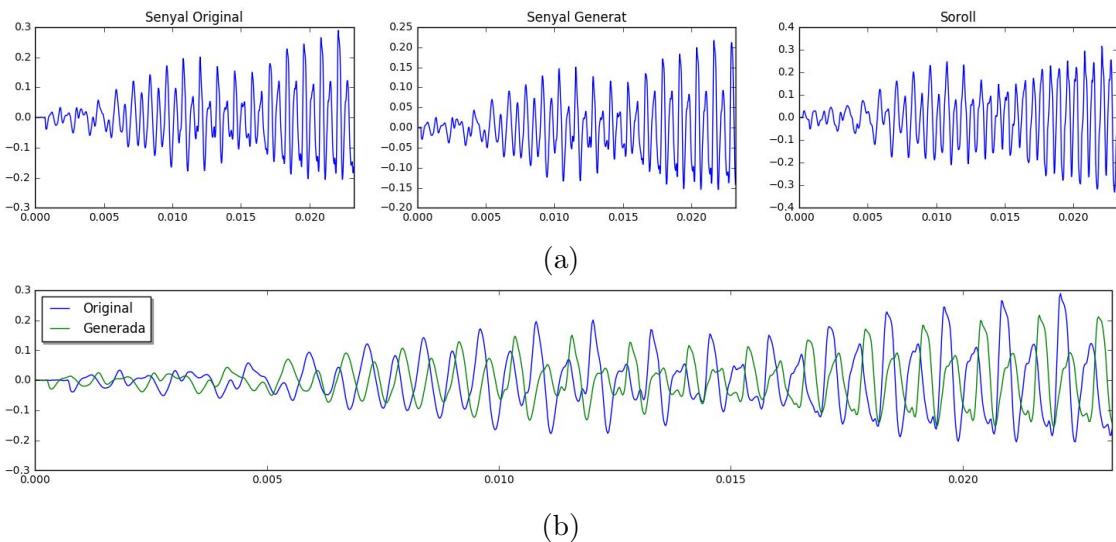


Figura 25: Exemple 3. (a) Comparació dels senyals original, generat i el soroll. (b) Superposició de l'ona original i generada.

Error Quadràtic = 0.014 i $SNR = -5.052$ dB

En el cas dels spectrogrames s'utilitza l'error quadràtic com a mesura. Per obtenir aquest valor es calcularà la distància entre els spectrogrames original i generat a partir de les matrius.

6.3.2 Comparació de funcions de cost

La relació senyal-soroll es pot considerar una funció de cost si es modifica adequadament.

$$error = \frac{1}{1 + SNR(\epsilon)} \quad (35)$$

Aquesta funció té rang $(0, 1]$ i no és pot comparar amb l'error quadràtic perquè aquest no és acotat superiorment.

Per veure quina funció té millors resultats s'ha fet un petit estudi entre la relació senyal-soroll i l'error quadràtic. Per fer aquesta comparació s'ha utilitzat el conjunt de dades de les notes sintètiques. Mitjançant aquests s'han entrenat i avaluat les xarxes per cadascuna de les funcions de cost. Amb tots els resultats del test es calculen la mitjana de l'error quadràtic i la relació senyal-soroll.

Els resultats obtinguts s'expressen a la següent taula:

Taula 1: Comparativa d'errors entre funcions de cost.

	Funció Error Quadràtic	Funció SNR
Mitjana Quadràtica	0.001	0.006
Mitjana Relació S-R	30.947	25.759

Els millors resultats s'obtenen quan l'error quadràtic és més proper a zero i quan la relació senyal-soroll té el valor més gran. A la taula anterior s'observa que amb la funció d'error quadràtic s'obtenen millors resultats que amb la funció SNR .

En definitiva, les xarxes neuronals s'entrenaran amb la funció de cost de l'error quadràtic, però els resultats es mostraran amb la relació senyal-soroll perquè és més intuïtiva.

6.3.3 Resultats de representació amb ones de so

En aquest apartat es presenten els resultats obtinguts per a la representació del so mitjançant ones. Abans de mostrar els resultats, seran comentats alguns detalls de l'entrenament. Referent al primer conjunt de dades, l'entrenament va ser realitzat amb un total de 3 456 880 mostres, en un temps inferior a tres hores. Per a aprendre es van realitzar 432 110 iteracions on, per cadascuna d'elles, es treballa amb un *batch*⁶ de 8 mostres.

A la Figura 26 s'observa com el valor de l'error quadràtic va disminuint a mesura que avancen les iteracions.

⁶El *batch* és el conjunt de dades que alimenten la xarxa en cada iteració.

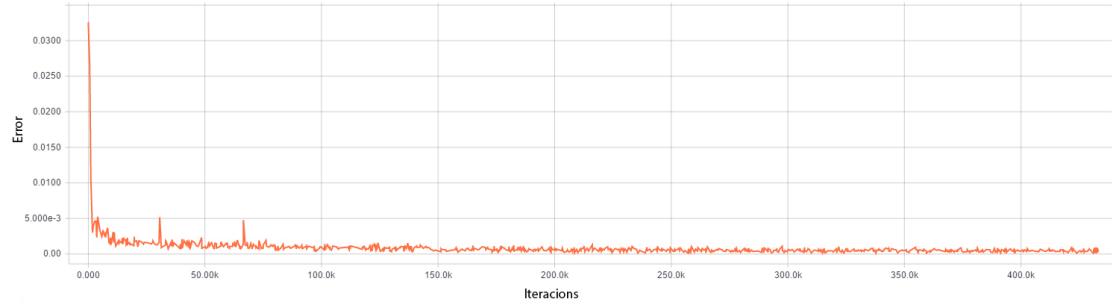


Figura 26: Gràfica de l'error de l'entrenament pel primer conjunt de dades

El segon conjunt de dades és més complex ja que les ones no són tan perfectes. Amb l'objectiu d'obtenir un error menor es va incrementar el nombre de dades fins a 20 705 520 mostres i la xarxa va entrenar un temps inferior a trenta hores.

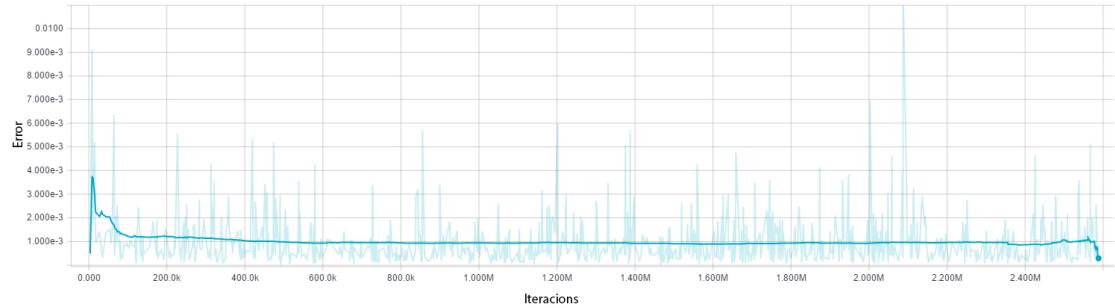


Figura 27: Gràfica de l'error de l'entrenament pel segon conjunt de dades

A la Figura 27 s'observen dues tonalitats de blau. La tonalitat més clara correspon al valor de l'error en cada iteració, mentre que la tonalitat més fosca correspon a la suavització dels errors. Amb aquesta es pot observar com l'error va disminuir amb el pas de les iteracions. Tot i això, una vegada acabat l'entrenament, l'error en aquest conjunt de dades és major que amb el conjunt de dades de les notes sintètiques.

Un cop finalitzat el procés d'entrenament s'analitzaran els resultats obtinguts amb el conjunt de test. Aquests resultats es representaran de la mateixa forma que els exemples de la mesura de la qualitat. En el peu de figura estarà indicat el valor de la relació senyal-soroll.

Notes sintètiques

Seguint l'arquitectura proposada s'ha aconseguit una xarxa que permet comprimir i descomprimir l'ona de so. Utilitzant el conjunt de dades de notes sintètiques s'ha obtingut una mitjana de la relació senyal-soroll al voltant de 43.463 dB.

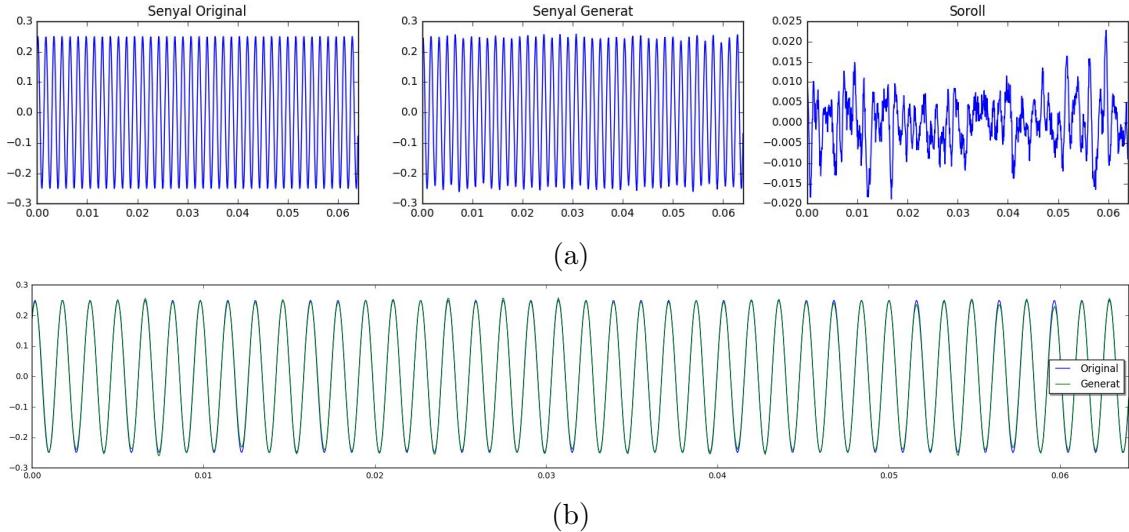


Figura 28: (a) Senyals originals, generat i soroll (b) superposició.
SNR = 57.808 dB

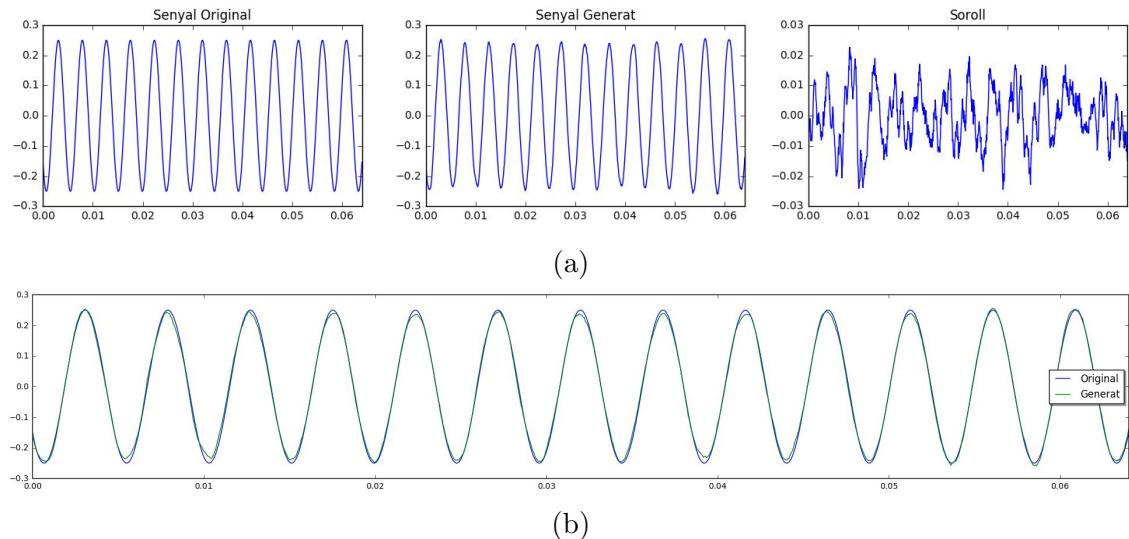


Figura 29: (a) Senyals originals, generat i soroll (b) superposició.
SNR = 51.655 dB

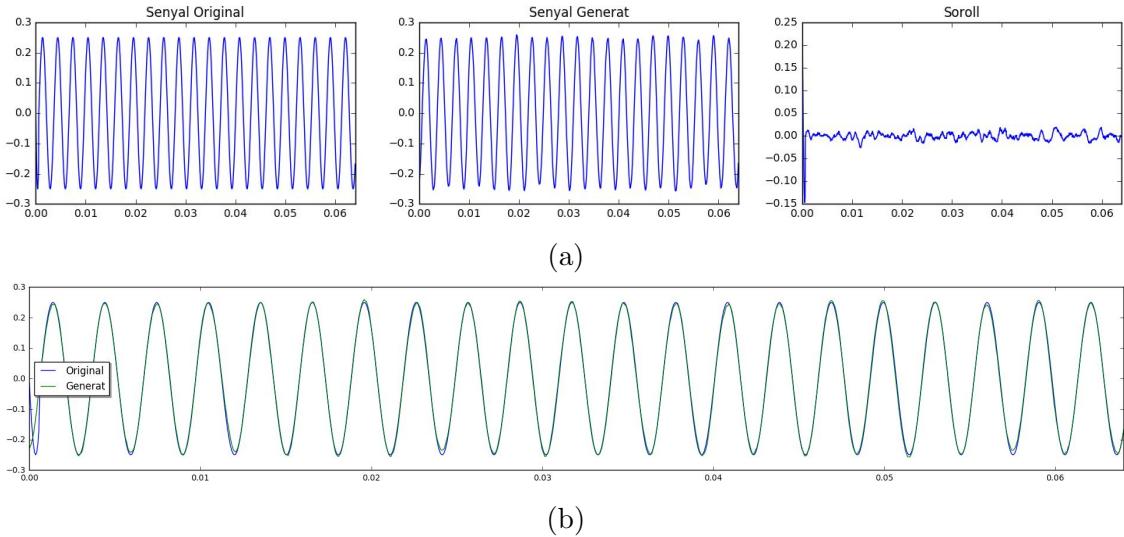


Figura 30: (a) Senyals originals, generat i soroll (b) superposició.
SNR = 43.534 dB

Les Figures 28, 29 i 30 s'han comprimit i descomprimit adequadament, generant diferències a l'amplitud de l'ordre de 10^{-2} .

Es pot observar a les Figures 31 i 32 que l'arquitectura té problemes quan s'ha d'adaptar a canvis de freqüència sobtats, fet que succeeix quan hi ha més d'una nota a la mostra.

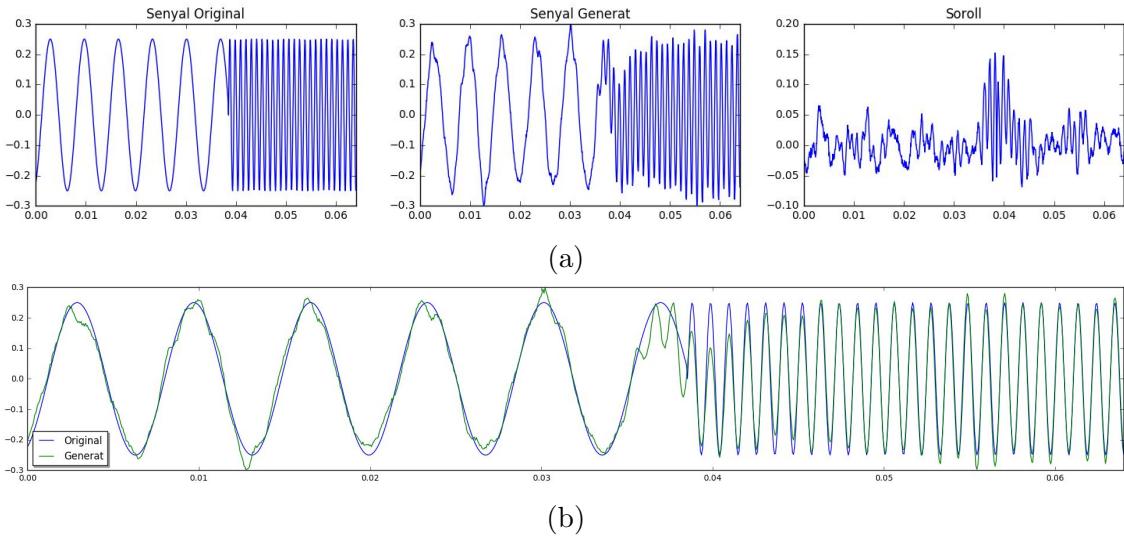


Figura 31: (a) Senyals originals, generat i soroll (b) superposició.
SNR = 29.243 dB

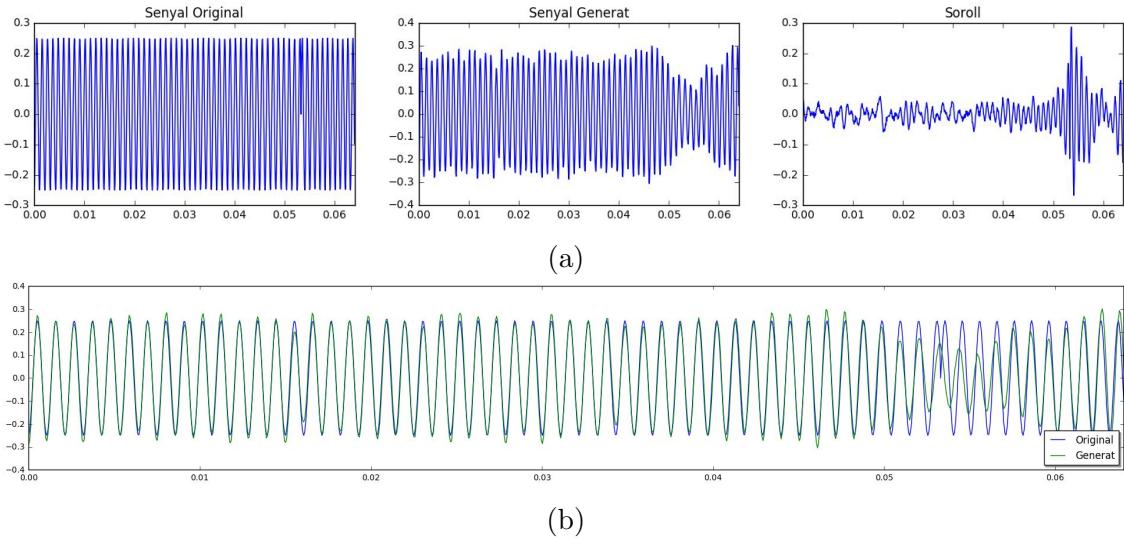


Figura 32: (a) Senyals originals, generat i soroll (b) superposició.
SNR = 22.349 dB

A l'últim exemple (Figura 33) s'obté un valor negatiu de la relació senyal-soroll. L'ona generada i l'original no són coincidents i, per tant, el soroll té més rellevància que el senyal.

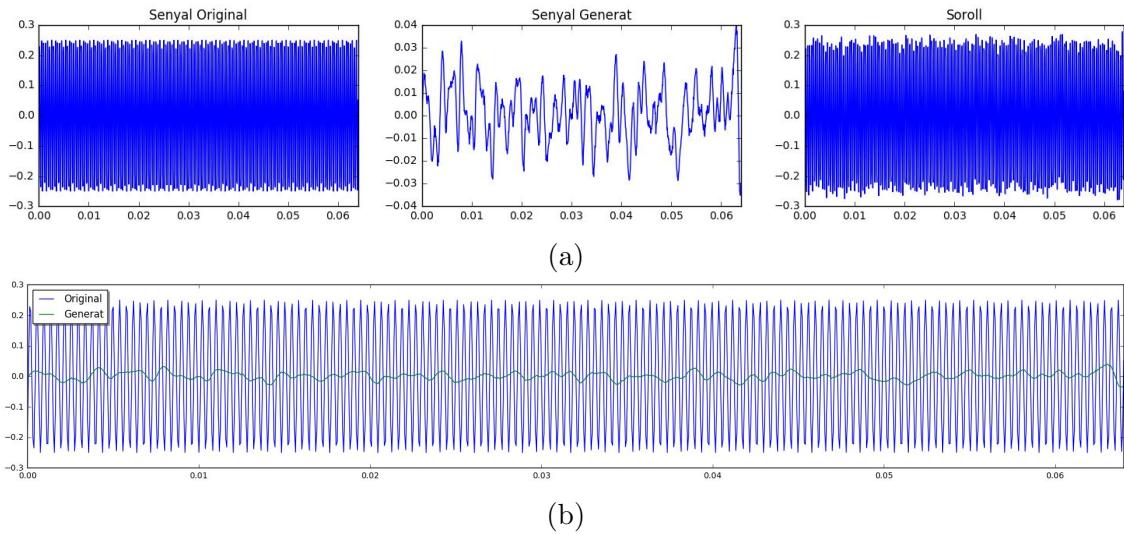


Figura 33: (a) Senyals originals, generat i soroll (b) superposició.
SNR = -0.034 dB

Fins a aquí arriba el primer conjunt d'entrenament. De moment la xarxa proposada és adequada per comprimir i descomprimir fragments d'àudio. D'aquesta manera, es pot afirmar que el vector de característiques que s'obté d'aquesta arquitectura és adequat per a les dades d'aquest conjunt d'entrenament.

Sons d'un partit de tennis

A continuació es mostren alguns dels resultats pel segon conjunt d'entrenament, els sons d'un partit de tennis. A partir d'aquests es discutirà si la mateixa arquitectura es pot utilitzar per trobar un vector de característiques amb aquest nou conjunt d'entrenament. En aquest experiment s'ha obtingut una mitjana⁷ de la relació senyal-soroll al voltant de 12.202 dB.

A les Figures 34, 35 i 36 s'observa que les descompressions s'aproximen adequadament al senyal original.

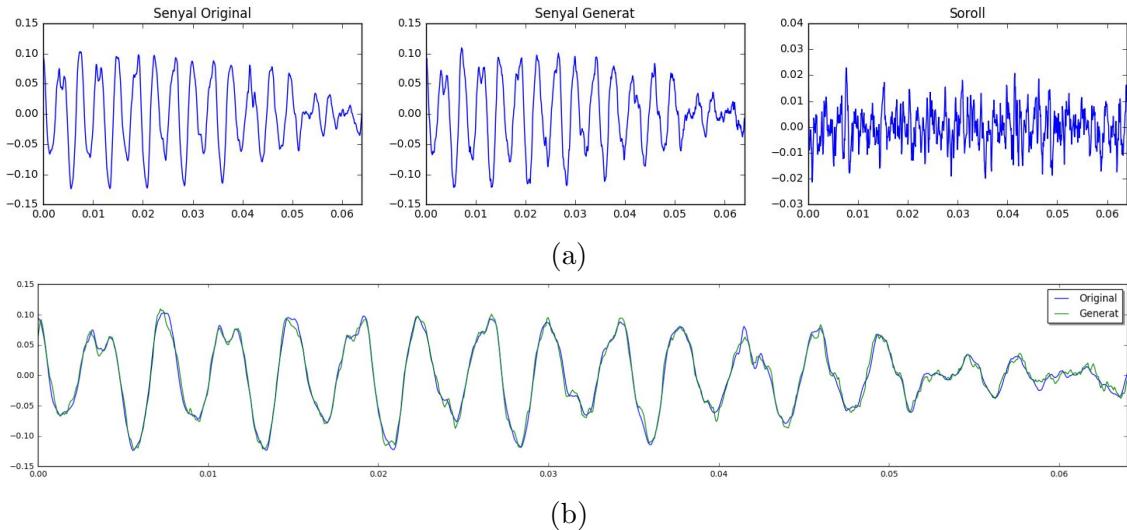


Figura 34: (a) Senyals originals, generat i soroll (b) superposició.
SNR = 35.754 dB

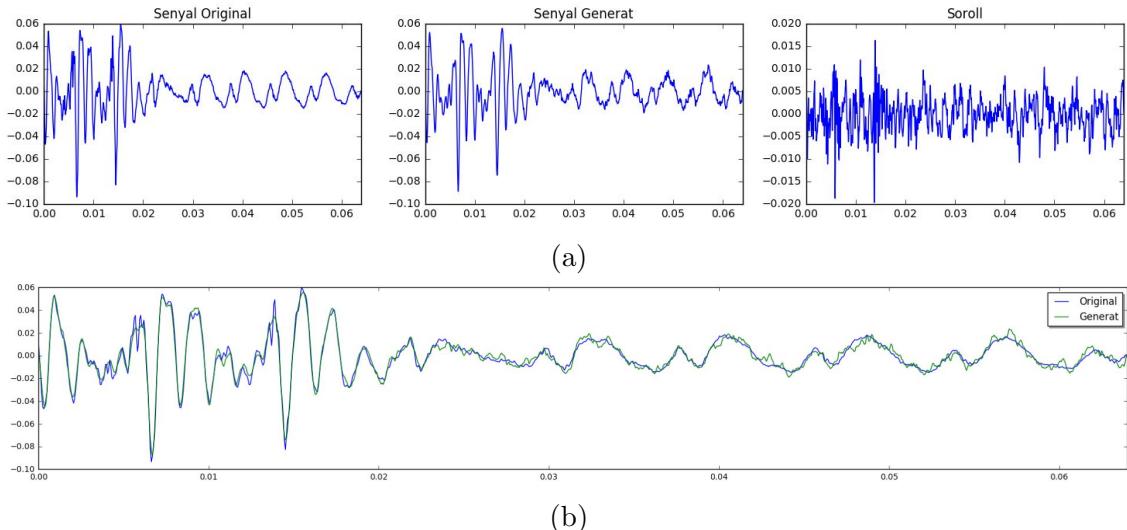


Figura 35: (a) Senyals originals, generat i soroll (b) superposició.
SNR = 26.719 dB

⁷No s'ha considerat una mostra que genera un valor de SNR $-\infty$.

Amb aquestes representacions es pot deduir que els problemes comentats amb el primer conjunt de dades són a causa de que el punt de canvi no és derivable⁸.

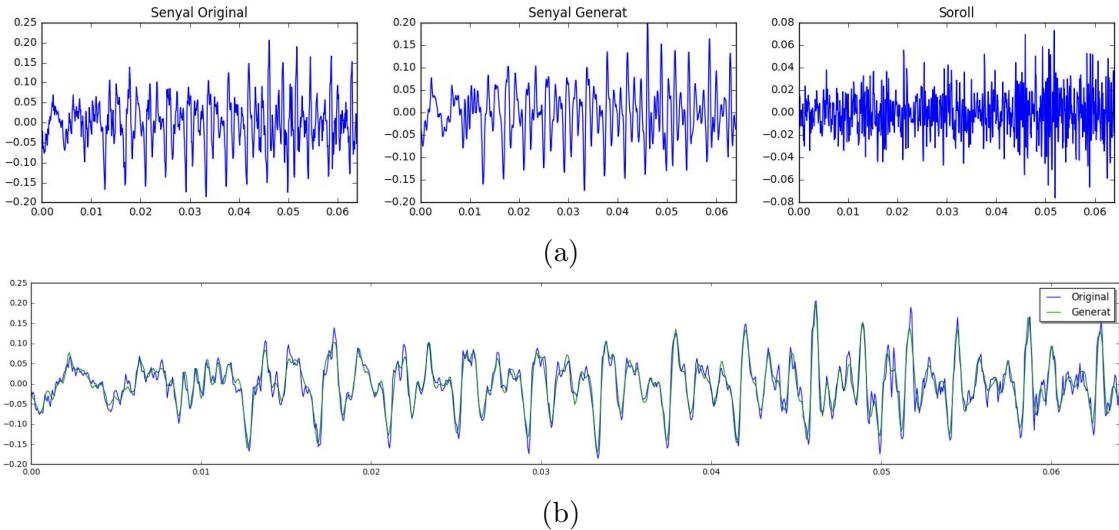


Figura 36: (a) Senyals originals, generat i soroll (b) superposició.
SNR = 20.146 dB

Continuant amb la mateixa arquitectura, la Figura 37 mostra un exemple on la descompressió només es capaç d'aproximar el senyal. Un possible motiu seria l'alta freqüència que s'obté en algunes zones de les mostres. Per evitar aquest problema es podria intentar entrenar amb més dades d'aquest tipus la xarxa o augmentar la freqüència de mostreig.

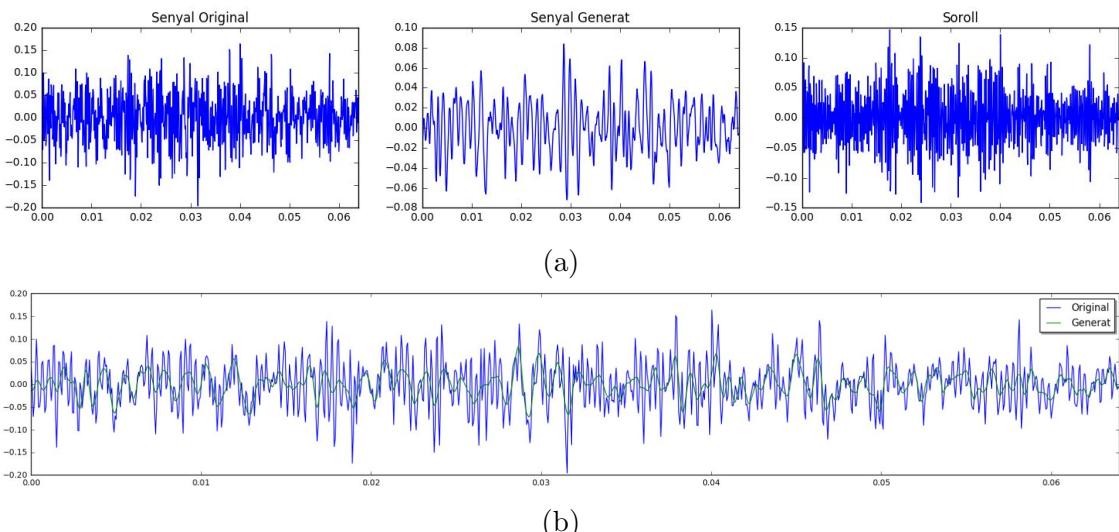


Figura 37: (a) Senyals originals, generat i soroll (b) superposició.
SNR = 3.041 dB

⁸Gràficament una funció contínua no és derivable en un punt si té una punxa.

Les figures 38 i 39 mostren els exemples més curiosos obtinguts en aquesta arquitectura. En aquests dos casos el valor de les amplituds és aproximadament zero o zero. En el primer cas s'obté un valor proper a zero de la relació senyal-soroll i, per tant, les ones no són coincidents. En el segon cas, quan el senyal original és silenci, el valor d'amplitud és manté constant a zero. Encara que el senyal generat sigui molt petit, la relació senyal-soroll serà $-\infty$ a causa de que si es considera $A_{original} \approx 0$ i $A_{soroll} \approx \epsilon$ a l'equació (34)

$$SNR_{dB} = 20 \log_{10} \left(\frac{A_{original}}{A_{soroll}} \right) \approx 20 \log_{10} \left(\frac{0}{\epsilon} \right) = -\infty \quad (36)$$

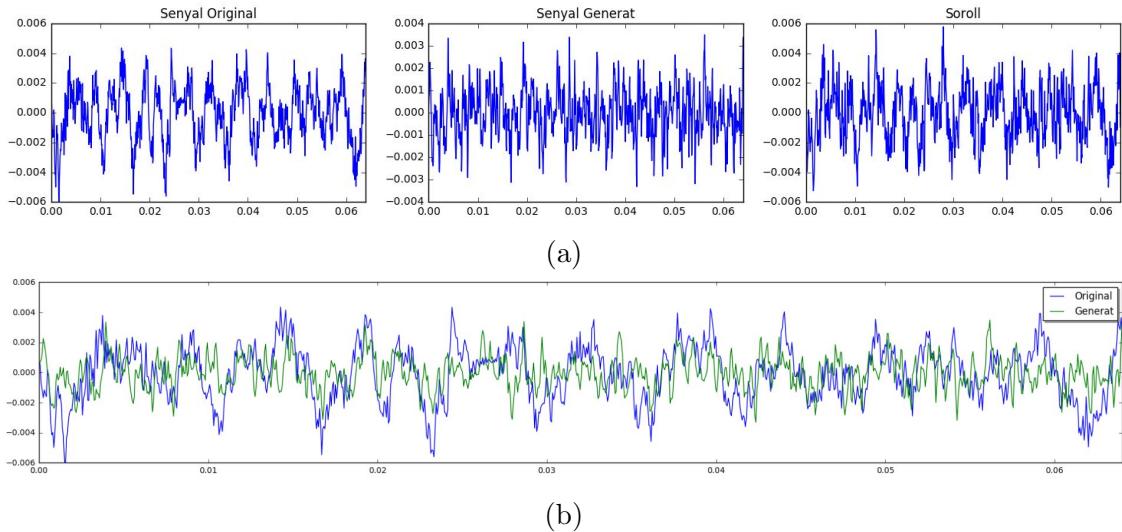


Figura 38: (a) Senyals originals, generat i soroll (b) superposició.
SNR = 0.210 dB

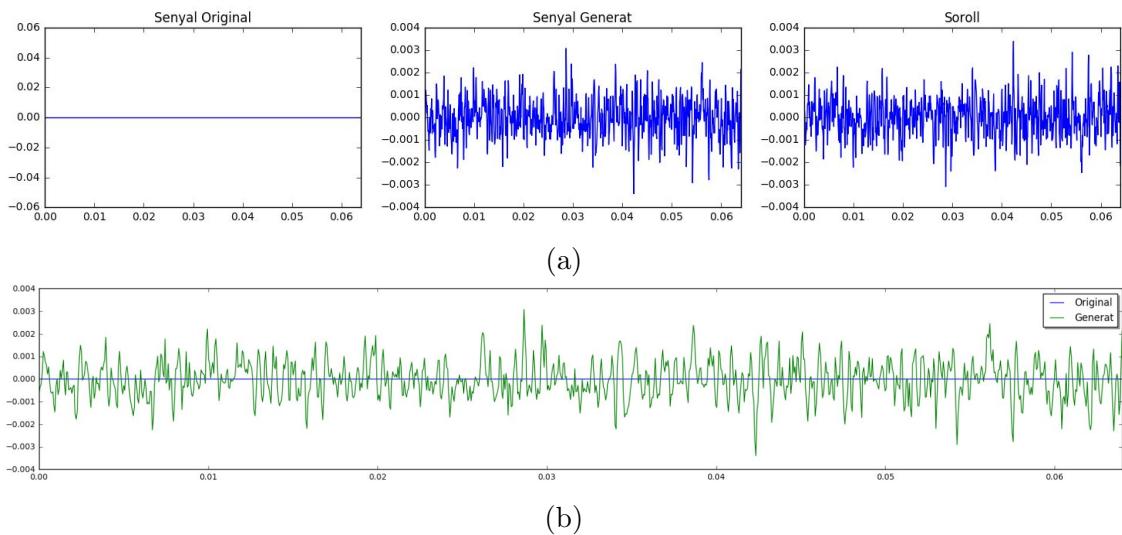


Figura 39: (a) Senyals originals, generat i soroll (b) superposició.
SNR = $-\infty$ dB

Amb els resultats obtinguts amb aquest conjunt d'entrenament es pot dir que la descompressió és adequada per un elevat percentatge de les mostres. Les mostres problemàtiques poden ser causades per una elevada freqüència o que l'amplitud del senyal original és poc intensa. Així doncs el vector de característiques obtingut amb aquesta arquitectura i aquest conjunt de dades és adequat, ja que les dades d'entrada i de sortida són molt similars.

6.3.4 Resultats de representació amb espectrogrames

En aquest apartat es presenten els resultats obtinguts per a la representació del so mitjançant espectrogrames. Referent al primer conjunt de dades, l'entrenament va ser realitzat amb un total de 3 456 880 espectrogrames en un temps inferior a les tres hores. De la mateixa manera que abans, s'entrena amb 432 110 iteracions amb 8 mostres per *batch*.

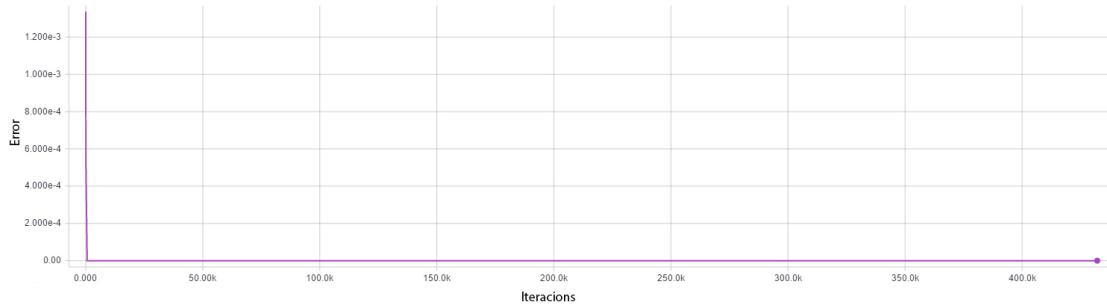


Figura 40: Gràfica de l'error de l'entrenament pel primer conjunt de dades.

Com a conseqüència de la gran quantitat d'iteracions no es pot veure adequadament l'inici del descens de l'error. A la Figura 41 hi ha una ampliació de les primeres 40 000 iteracions.

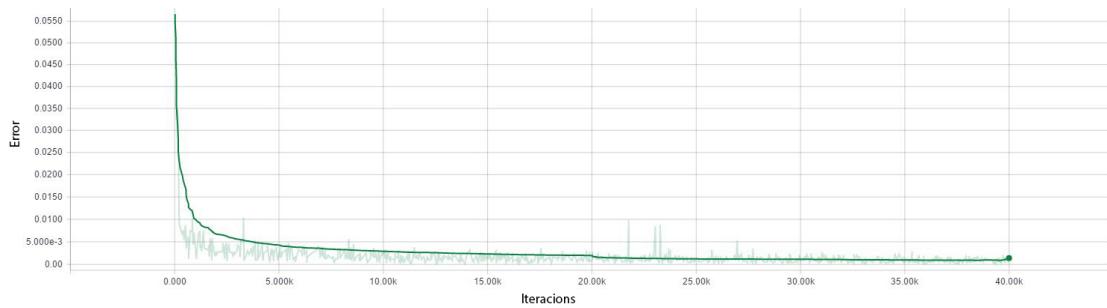


Figura 41: Ampliació de la gràfica de l'error de l'entrenament pel primer conjunt de dades.

El segon conjunt de dades va ser entrenat amb una quantitat inferior d'iteracions perquè no es van obtenir bons resultats.

A les Figures 41 i 42 s'observen dues tonalitats del mateix color. La tonalitat més clara correspon al valor de l'error en cada iteració. Mentre que la tonalitat

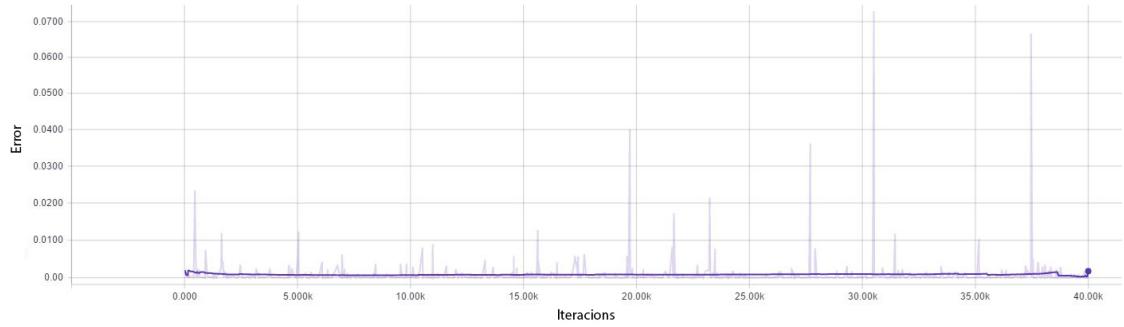


Figura 42: Gràfica de l'error de l'entrenament pel segon conjunt de dades

més fosca correspon a la suavització dels errors. Amb aquesta es pot observar com l'error va disminuint amb el pas de les iteracions.

Notes sintètiques

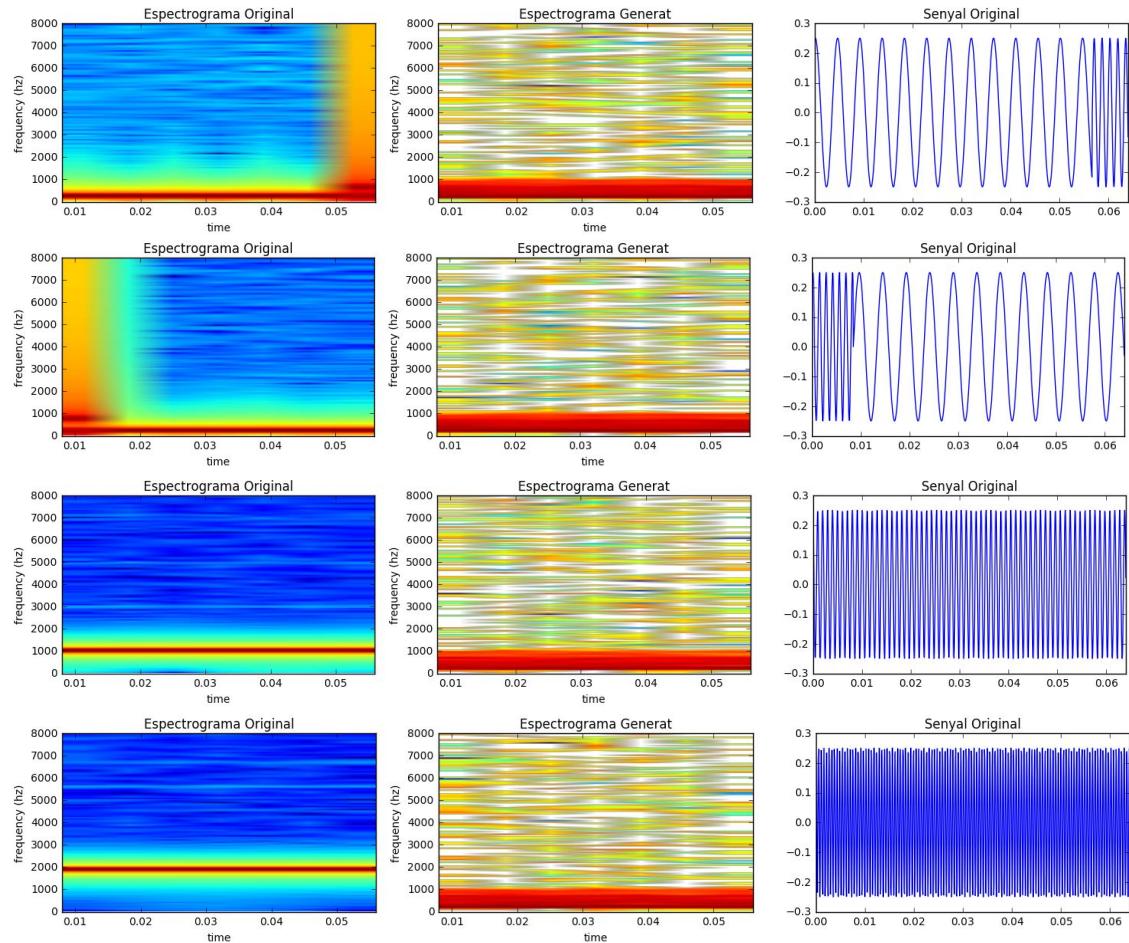


Figura 43: Resultats del test pel primer conjunt de dades

Tot i obtenir una mitjana d'error quadràtic de $7.59 \cdot 10^{-10}$, quan es comparen els

espectrogrames visualment, es dedueix que l'espectrograma generat, difícilment és una representació del senyal.

Sons d'un partit de tennis

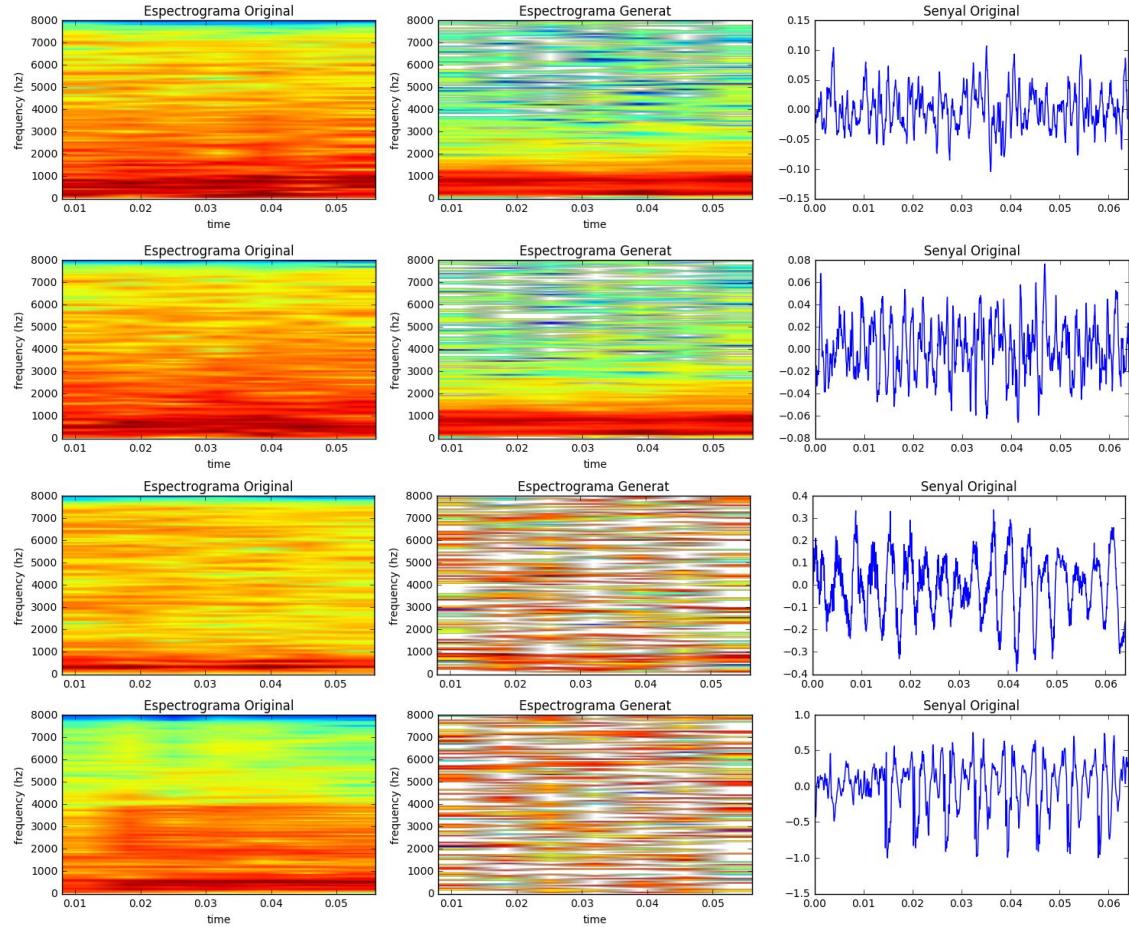


Figura 44: Resultats del test pel segon conjunt de dades

Observant els resultats queda clar que l'arquitectura proposada no pot reconstruir un espectrograma donat. En conseqüència, no es té la certesa de si el vector de característiques d'aquesta arquitectura és adequat per altres aplicacions.

Una de les possibles raons d'aquests resultats és que les dades de generació de l'espectrograma són insuficients. Per tant, caldria introduir-ne un nombre major per tal que la xarxa trobi patrons i relacions.

Una altra font d'error és que, potser, la xarxa proposada no és idònia per treballar amb imatges de dimensió 129×7 . Una alternativa a la xarxa de capes denses seria crear una nova xarxa a partir de capes convolucionals.

7 Generació d'ones de so

En aquesta secció es tracta el problema de generar ones de so. Tot seguit es comentaran els diversos models de sistemes que s'han estudiat per a solucionar el problema. Encara que s'han investigat i analitzat els tres models, únicament el primer i el segon s'han utilitzats per fer proves.

7.1 Arquitectures

7.1.1 Mètode 1: Xarxes neuronals recurrents

Les xarxes neuronals recurrents [6, 37] són utilitzades, normalment, per a la generació de seqüències de lletres o paraules mitjançant un vocabulari. Així doncs, en primer moment es va pensar que aquest tipus de xarxes també podria generar ones de so.

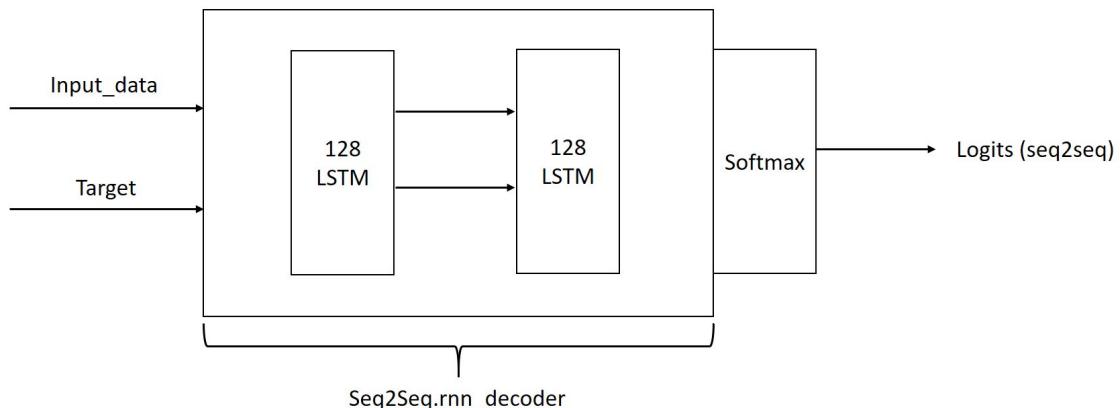


Figura 45: Representació de la xarxa del mètode 1

La Figura 45 mostra la primera arquitectura que es va estudiar i testejar. Aquesta consta de dues capes amb 128 unitats cadascuna.

El procediment que segueix la xarxa es pot separar en diversos passos. En el primer pas, les dades d'entrada s'han de dividir en grups de n elements, on n s'anomena la llargada de la seqüència. Aquesta variable, al mateix temps, defineix la quantitat de cicles que tindrà la xarxa. Les dades de l'entrada es codifiquen generant un vocabulari que, més endavant, s'utilitzarà per determinar l'element que es genera. A continuació les dades entrenen en les cèl·lules LSTM seqüencialment, generant una sortida i actualitzant els paràmetres de la xarxa. Això permet reduir l'error d'entrenament. A l'etapa final, el *softmax* descodifica el valor de sortida retornant un element del vocabulari.

Per poder generar ones de so, tant l'entrada com la sortida seran els valors de les amplituds del conjunt d'aprenentatge.

7.1.2 Mètode 2: WaveNet

WaveNet [14] és un sistema creat per Google Research. Encara que aquest codi no és obert, s'ha utilitzat una implementació publicada a *Github*:
<https://github.com/ibab/tensorflow-wavenet>.

Aquest sistema és una xarxa neuronal profunda que, inicialment, estava pensada per a convertir text en veu i, per tant, generar ones de so. La xarxa segueix un model probabilístic i autoregressiu. La probabilitat conjunta d'una ona $X = \{x_1, \dots, x_T\}$ ve donada per l'expressió:

$$p(X) = \prod_{t=1}^T p(x_t | x_1, \dots, x_{t-1}) \quad (37)$$

Aquesta probabilitat vol dir que cada mostra d'àudio està condicionada per les mostres anteriors.

Abans d'analitzar la xarxa s'exposaran les noves capes aplicades per Google.

Convolució Causal

Les capes causals són capes de convolució que garanteixen que el model no pot violar l'ordre de les dades introduïdes i que el sistema no pot utilitzar temps futurs per una predicción en l'instant t .

Aquest tipus de capes són més ràpides d'entrenar que els recurrents, ja que tenen menys connexions. Tot i que són més ràpides, es troba el problema de que calen moltes capes per tal d'augmentar el camp receptiu de la xarxa. En aquest cas, el camp receptiu d'aquesta capa és dos, és a dir, cada neurona de la capa j , està connectada a dues neurones consecutives de la capa $j - 1$.

Convolució Dilatada

Per a solucionar el problema anterior del camp receptiu, es dissenyen les capes de convolucions causals dilatades. Aquestes són capes de convolució causal on s'affageix una dilatació quan es fan les connexions amb les neurones de la capa anterior.

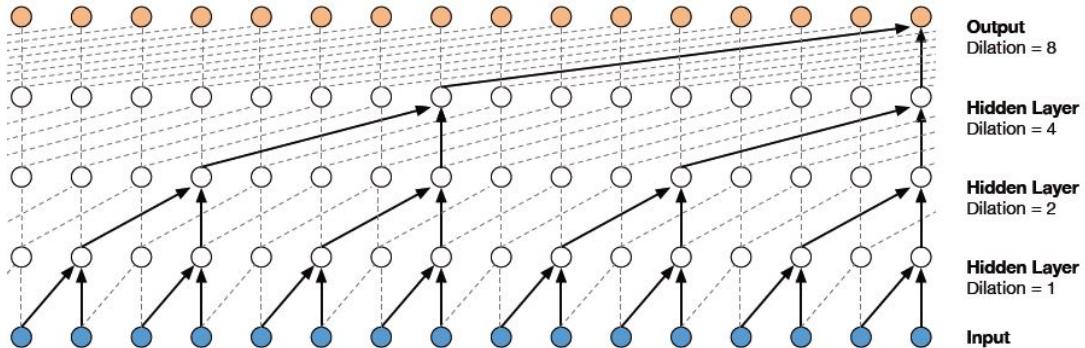


Figura 46: Visualització de les connexions de les capes convolucionals dilatades

La Figura 46 és un exemple de com funciona aquest tipus de convolució. La primera capa oculta té dilatació 1 i equival a una convolució causal. La segona

capa oculta té dilatació 2 i, per tant, de cada dues neurones realitza connexions amb una. I així successivament per qualsevol valor de dilatació.

Afegint la dilatació s'aconsegueix augmentar el camp receptiu de la xarxa i, en conseqüència, conèixer més quantitat d'elements amb menys capes.

Connexions Residual i de Salt

Les connexions residuals, bàsicament, incorporen una connexió directa des de l'entrada de la capa a la sortida. Aquesta connexió permet mantenir part de la informació original.

Les connexions de salt copien la sortida de cadascuna de les capes.

Arquitectura

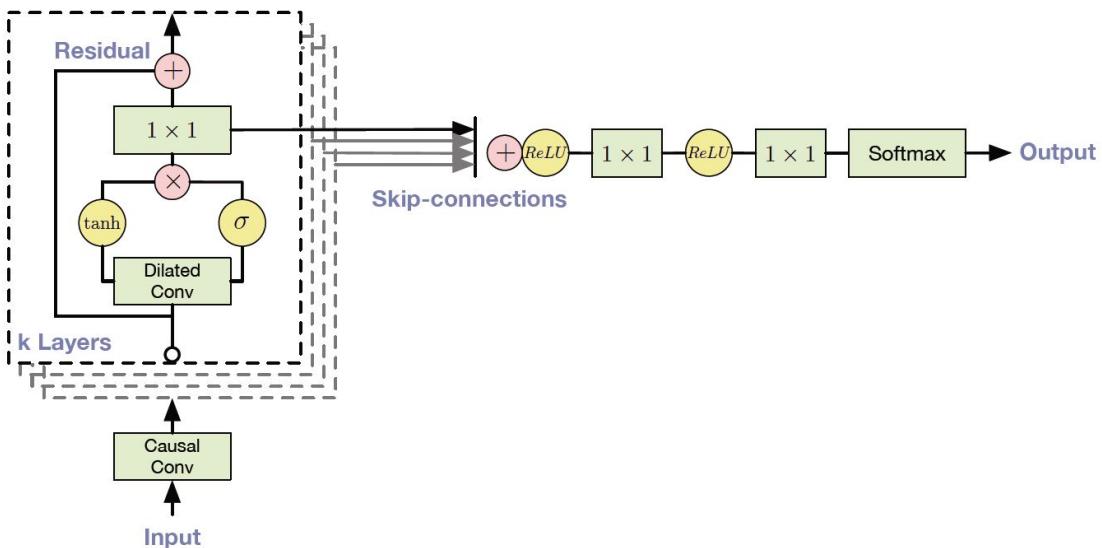


Figura 47: Arquitectura de la xarxa WaveNet

És important destacar que, normalment, una ona d'àudio està guardada en una seqüència de valors enters de 16-bits. D'aquesta forma la capa de *softmax* necessita una sortida de $2^{16} = 65\,536$ possibilitats en cada mostra per modelar tots els possibles valors. Per prevenir i reduir aquest cost computacional, primer s'ha de fer una quantització [4] a 256 valors aplicant la funció

$$f(x_t) = \text{sign}(x_t) \frac{\ln(1 + \mu|x_t|)}{\ln(1 + \mu)} \quad (38)$$

on $-1 < x_t < 1$ i $\mu = 255$.

Les dades s'introdueixen primer en una capa de convolució causal que genera la mostra corresponent a l'instant $t + 1$, només utilitzant les t anteriors (Figura 47). Un cop aplicades les convolucions necessàries, les dades entren a la pila de capes de convolucions causals dilatades que permetran tenir un camp receptiu més elevat, equivalent a la memòria del sistema. Dintre de cada capa hi ha dos tipus de connexions, les residuals i les de salt, que serveixen per augmentar la velocitat de convergència. Finalment, es fa el pas contrari a la quantització per a obtenir les ones de so.

7.1.3 Mètode 3: Metodologia que es segueix en l'article *Visually Indicated Sounds*

El darrer objectiu d'aquesta memòria consisteix en generar l'ona de so corresponent a un partit de tennis. A causa de l'alta similitud en els objectius, s'estudia aquest article *Visually Indicated Sounds* [38]. Investigadors del MIT, Berkeley i Google Research proposen l'objectiu de generar l'ona de so associada a l'impacte d'una baqueta contra diversos objectes partint d'un vídeo en silenci.

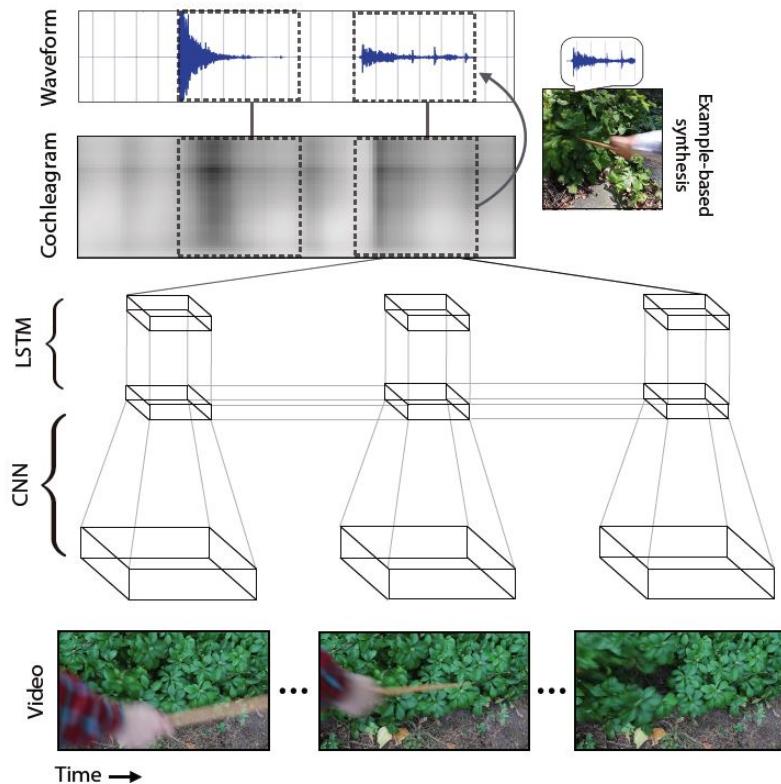


Figura 48: Arquitectura proposada a l'article

En el procés d'aprenentatge les dades d'entrenament estan formades per un vídeo mut i un *cochleagram* generat a partir del so del vídeo. Un *cochleagram* és una representació de les característiques del so generat a partir de l'aplicació d'un banc de filtres i la transformada de Hilbert sobre l'ona de so [17].

Quan s'avalua la xarxa amb el conjunt de test, l'algorisme agafa una seqüència d'un vídeo com entrada i prediu el *cochleagram*. L'arquitectura per tal de generar aquesta representació està formada per una xarxa convolucional i una xarxa recurrent. A partir de la xarxa convolucional volen entendre què succeeix al vídeo i, a partir de la xarxa recurrent, aprendre a generar les característiques del so corresponents a quan els objectes són colpejats. Mitjançant l'algorisme KNN es busquen les característiques més similars contingudes en la base de dades i es retorna el so associat.

Tot i que l'objectiu d'aquest article és la sintetització a partir d'exemples, en el

futur s'utilitzà part de la xarxa quan s'hagi de relacionar un vídeo amb la generació d'ones de so.

7.2 Conjunt de dades

Els diversos conjunts de dades utilitzats en el model de la xarxa recurrent (Mètode 1) són seqüències de valors entre -1 i 1 que variaran segons l'objectiu de cada prova.

Els dos conjunts de dades utilitzats en l'entrenament de la xarxa WaveNet de Google (Mètode 2) són semblants al conjunt de dades de l'apartat de representació. Per aquesta xarxa s'ha augmentat la durada de les notes sintètiques a dos segons, incloent-hi combinacions de dues i tres notes individuals. Aquest canvi és per obligar a que la xarxa aprengui a generar sons que tinguin una consistència en el temps. El segon conjunt és el mateix que en el cas anterior, sons d'un partit de tennis.

7.3 Resultats obtinguts i discussió

A diferència del cas de la representació del so, no hi ha cap mesura automàtica que confirmi si les dades generades són correctes o no. Quan es treballa amb la generació d'àudio s'ha d'utilitzar el criteri propi per determinar si l'ona generada és un so adequat o és simplement soroll.

7.3.1 Resultats amb la xarxa neuronal recurrent

El major inconvenient és que, usualment, les xarxes recurrents estan pensades per seqüències i no per valors numèrics. Tot i això, es van realitzar algunes proves.

En les primeres proves es va testejar si l'arquitectura de la xarxa recurrent era capaç d'aprendre seqüències numèriques. És per aquest motiu que les dues primeres proves que es fan són d'una complexitat baixa.

El primer experiment consistia en què la xarxa aprengués a generar una seqüència constant, és a dir, un valor li era donat i l'havia de mantenir. En aquest cas els resultats van ser positius i en poques iteracions la xarxa ja coneixia la seqüència a generar.

El segon experiment consistia en una seqüència de valors que s'anaven repetint periòdicament. La idea d'aquest entrenament era simular a un nivell molt simple el comportament d'una ona de so. La sèrie introduïda començava a 0 , augmentava fins a 1 , disminuïa a -1 i tornava a augmentar fins a 0 amb increments de 0.1 .

$$0, 0.1, 0.2, \dots, 1, 0.9, \dots, -1, -0.9, \dots, 0$$

Aquest experiment va sortir d'acord al que s'esperava.

Per la generació d'un segon de so es necessiten com a mínim 8000 mostres, tot i que seria millor generar 16000 per tenir una millor qualitat de so. L'objectiu és

generar aquesta quantitat de mostres i per això es comencen a fer experiments més realistes on es puguin generar 16000 mostres per segon.

Pel següent experiment es va augmentar la quantitat d'intervals, utilitzant un increment de 10^{-4} en la seqüència anterior. No es van obtenir els resultats desitjats. El motiu principal d'aquests resultats negatius és que les xarxes neuronals recurrents són adequades per generar seqüències arbitràries amb vocabularis limitat. En aquest últim experiment el vocabulari augmenta a una mida de l'ordre de 10000 elements diferents. És possible que amb l'arquitectura anteriorment esmentada no hi hagués prou unitats per treballar amb aquesta quantitat de valors de vocabulari.

La xarxa recurrent va ser la primera xarxa estudiada i al no tenir bons resultats, es va investigar el mètode 2.

7.3.2 Resultats amb la xarxa WaveNet

Els experiments corresponents a aquest mètode se centraven en el funcionament de la xarxa.

Notes Sintètiques

La primera prova que es va realitzar amb aquesta xarxa consistia en la generació de notes musicals o combinacions d'aquestes. Per dur a terme aquest experiment es va alimentar la xarxa amb el conjunt de dades format per les notes musicals sintètiques. La dificultat en aquest entrenament consistia en el fet que la xarxa aprengués a generar un cicle i posteriorment repetir-ho fins al canvi de freqüència o fi de la generació.

Aquesta prova va tenir bons resultats. Els sons que generava la xarxa després d'entrenar durant quatre dies eren els desitjats. Durant aquest temps el sistema va realitzar més de 100 000 iteracions, tal com es veu a la Figura 49. El temps de còmput per cada iteració és aproximadament tres segons, generant un error a l'última iteració de $1,75 \cdot 10^{-3}$. L'elevat temps d'entrenament mostra la complexitat de la xarxa WaveNet.

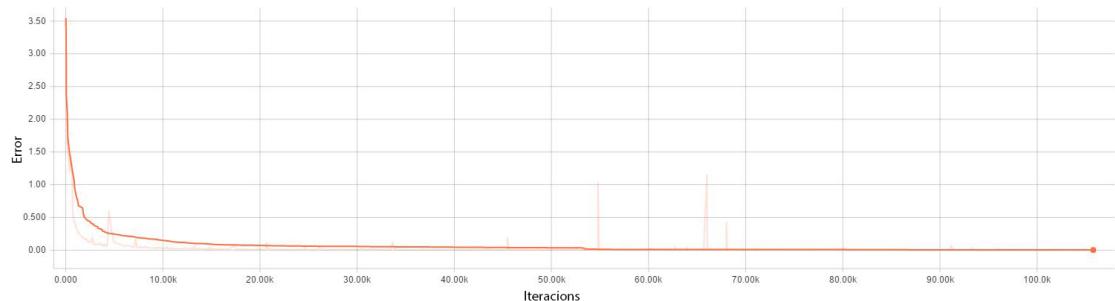


Figura 49: Arquitectura proposada a l'article

A la Figura 49 s'observen dues tonalitats de taronja. La tonalitat més clara correspon al valor de l'error en cada iteració. Mentre que la tonalitat més fosca

correspon a la suavització dels errors i amb aquesta es pot observar com l'error va disminuint amb el pas de les iteracions.

A continuació s'exposa una evolució de la generació d'ones al llarg de les iteracions. Per fer aquest exemple d'evolució es proporcionen 8000 mostres com a llavor o *seed* per la generació. Les ones de so de la Figura 50 només mostren un fragment de 4000 de les 32 000 mostres generades per poder veure l'evolució de la forma sinusoïdal amb el pas de les interaccions.

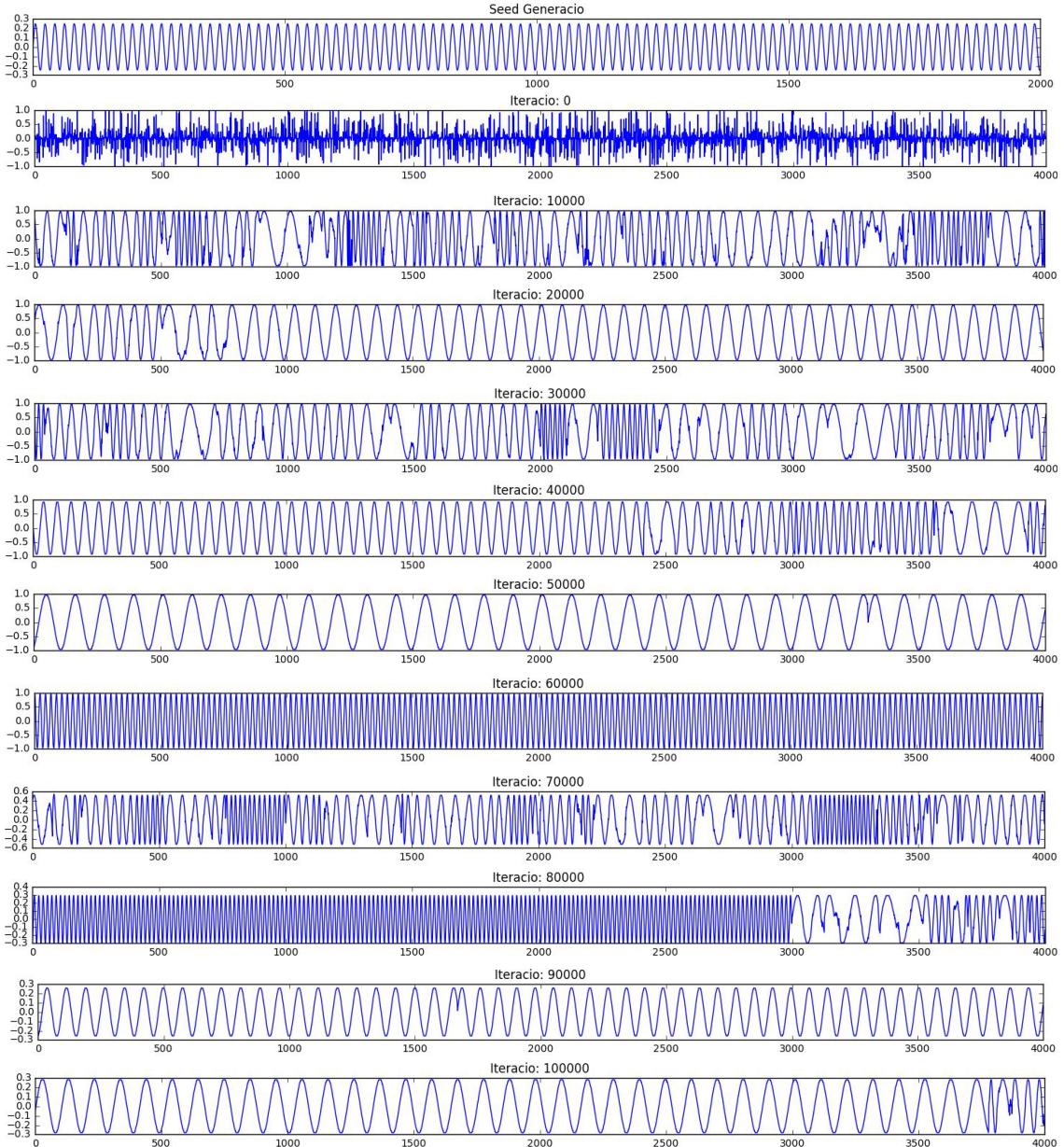


Figura 50: Evolució del so. Sons disponibles a *Github*

La Figura 50 està formada per fragments d'àudio generat per la xarxa WaveNet. A la part superior de cada representació es mostra el nombre d'iteracions.

En aquesta figura s'observa que des de la primera iteració a la 60 000, la xarxa

va aprenent a generar ones sinusoïdals cada cop més perfectes amb rang $(-1, 1)$. Tot i que el rang de l'amplitud és manté constant, la freqüència va variant en les diferents iteracions. A partir de les 60 000, s'observa com l'ona va reduint els seus valors d'amplitud fins a arribar a un rang $(-0.3, 0.3)$, que és el rang de les ones generades sintèticament.

A continuació s'exposa un exemple més de generació de notes (Figura 51). En aquest es mostra una única generació completa de 56 000 mostres, sent les 8000 primeres la llavor. El total de mostres està dividit en fragments de 8000 valors per poder observar les freqüències.

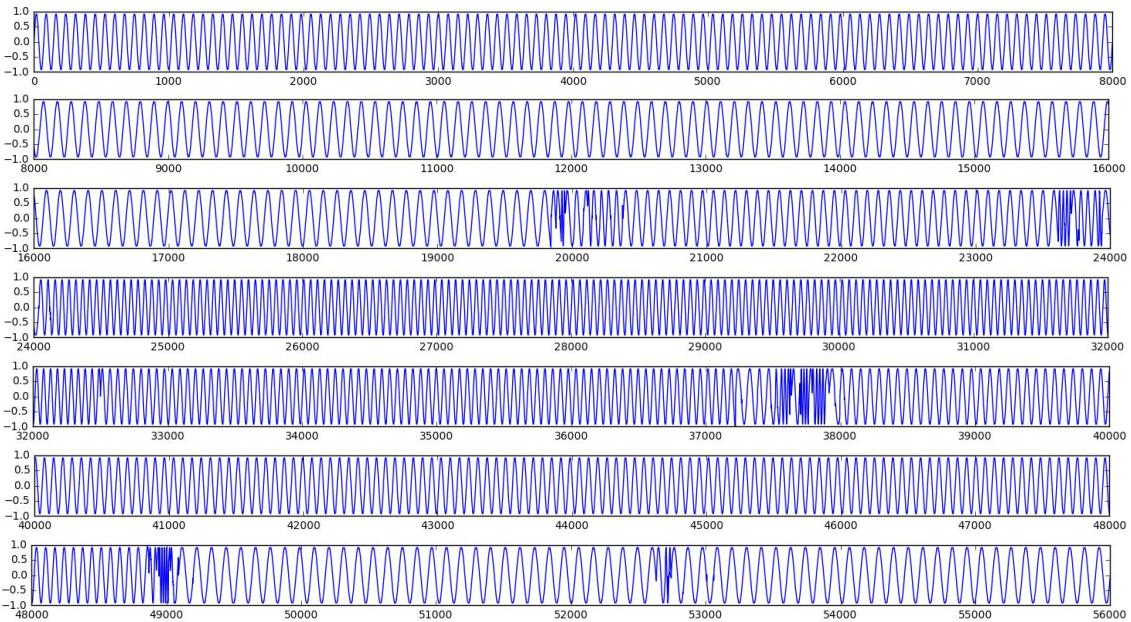


Figura 51: Fragmentació d'una generació. So disponible a *Github*

En l'exemple anterior (Figura 51) s'observa que hi ha canvis de freqüència al voltant de les mostres: 20000, 24000, 37000 i 49000 que indica un canvi de to. Per fer la comprovació del canvi de freqüència a l'enllaç següent es pot consultar l'àudio d'aquesta figura https://github.com/laizpablo/NeuralNetwork/blob/master/generacio/generacio_notes/resultats/generated65500_3s_la3.wav. En aquest enllaç també es poden escoltar altres generacions d'aquesta xarxa.

Sons d'un partit de tennis

La segona prova consistia a generar sons creïbles de partits de tennis. Per l'aprenentatge s'utilitza el segon conjunt de dades. S'ha de tenir present que el canvi de dificultat entre la primera prova i aquesta segona és important, ja que les ones són més complexes.

La xarxa alimentada amb el segon conjunt de dades ha d'aprendre a generar sons que corresponen a partits de tennis. Després d'entrenar durant més d'una setmana la xarxa, amb diversos entrenaments, alguns dels sons generats es poden catalogar

dintre d'un partit de tennis. Per contra, també hi ha un percentatge elevat de generacions de so que són soroll. La generació de soroll pot venir donada per la complexitat del problema i les dades d'entrada, ja que aquestes també contenen l'ambient del públic quan els jugadors disputaven els partits.

En aquest apartat cada iteració correspondrà a una gràfica i totes elles tindran en comú la llavor, que serà 8000 mostres. A continuació es visualitzaran alguns exemples de les generacions produïdes.

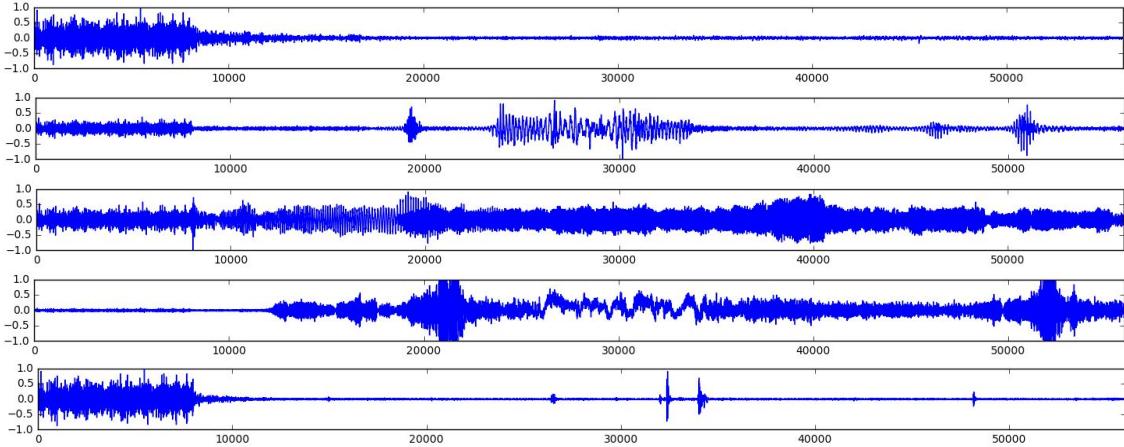


Figura 52: Exemple 1: Evolució del so d'un partit de tennis. Sons disponibles a *Github*

A l'última iteració de la figura anterior s'obté un resultat on s'aprecia un cop de pilota. De la mateixa manera que abans, els àudios es troben a *Github*.

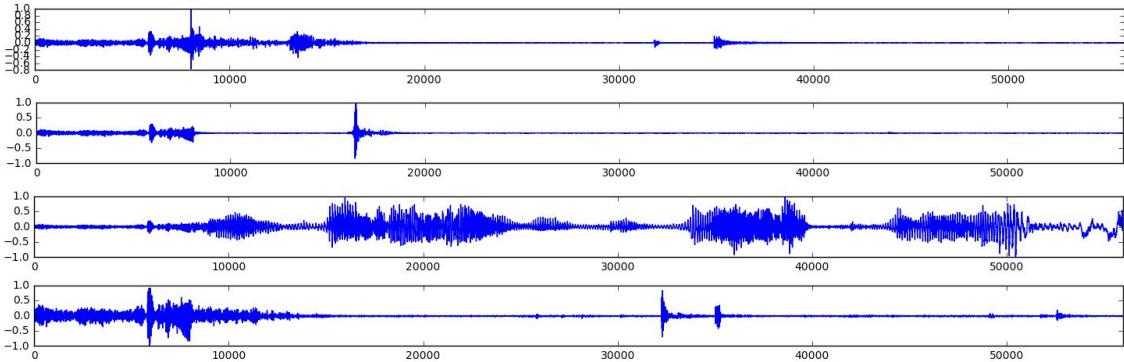


Figura 53: Exemple 2: Evolució del so d'un partit de tennis. Sons disponibles a *Github*

A la Figura 53 hi ha un altre exemple de l'evolució del so a partir de les iteracions, però amb una llavor diferent.

En aquests exemples, l'última iteració generada és on s'obtenen els sons associats a un partit de tennis. Concretament s'escolta el cop de la pilota amb la raqueta. En les altres iteracions es percep soroll o sembla que algú parla.

En resum, la xarxa ha generat bones mostres amb els dos conjunts d'entrenament. La complexitat d'aquest problema i la xarxa utilitzada provoca un elevat cost computacional durant l'entrenament.

8 Conclusions

Després d'haver realitzat aquest treball en generació i representació del so mitjançant xarxes neuronals es conclou el següent.

8.1 Objectius proposats

El primer objectiu era trobar una representació del so mitjançant un vector de característiques. Per assolir-ho es van dur a terme dos experiments amb dos conjunts de dades. El primer experiment tractava d'obtenir una nova representació a partir de les amplituds de l'ona de so. I el segon era similar, però treballant amb l'espectrograma. Després d'haver realitzat els experiments i obtingut els resultats, aquest objectiu només s'ha pogut aconseguir satisfactoriament mitjançant les amplituds de les ones de so. Pel contrari, l'arquitectura utilitzada pels espectrogrames no va ser exitosa. Per a poder obtenir-ho seria necessari una xarxa neuronal formada per capes convolucionals en lloc d'una construïda per capes denses.

El segon objectiu consistia en la generació de diversos tipus de sons o música. Per completar aquest objectiu s'ha utilitzat una xarxa neuronal de Google anomenada WaveNet, tot i que prèviament s'havia intentat amb una xarxa neuronal recurrent. Aquesta xarxa va ser entrenada amb dos conjunts diferents, un de notes sintètiques i un altre de sons de partits. Amb el primer conjunt els resultats són bons, i amb l'augment de les iteracions els àudios sonen millor. Amb el segon conjunt, tot i obtenir resultats desitjats, hi ha altres sons generats que eren soroll. Tot i així, els resultats són positius perquè s'han aconseguit mostres prometedores.

S'ha de tenir present que treballar amb xarxes neuronals té un l'alt cost computacional i és necessària una gran quantitat de recursos per poder obtenir algun resultats. És per aquest motiu que els resultats mostrats i obtinguts en aquest treball són limitats.

8.2 Treball Futur

L'últim objectiu pot semblar una mica ambiciós. Consisteix en crear un sistema que mitjançant xarxes neuronals sigui capaç d'aplicar el concepte d'efectes de Foley a un partit de tennis. Aquest sistema es pot dur a terme mitjançant l'article *Visually Indicated Sounds* explicat en una secció anterior.

Referències

- [1] Canales, J.. Desired Machines: Cinema and the World in Its Own Image. *Science in Context*. 2011, Vol. 24, pp. 329–359.
- [2] Bonebright, T. L.. Were those coconuts or horse hoofs? visual context effects on identification and perceived veracity of everyday sounds. *Conferència Internacional sobre la visualització auditiva*. 2012.
- [3] Wu, C. i Jay Juo, C. C.. Design of Integrated Multimedia Compression and Encryption Systems. *IEEE Transactions on Multimedia*. 2005, Vol. 7, N. 5.
- [4] Hanzo, L., Somerville, F. i Woodard, J.. Voice and Audio Compression for Wireless Communications. *IEEE* . 2007.
- [5] Google Brain Team. *Magenta* [en línia]. Edició 2016 [consulta: gener de 2017]. Disponible a: <https://magenta.tensorflow.org/>
- [6] Elman, J.. Finding Structure in Time. *Cognitive Science* 14. 1990.
- [7] Hochreiter, S. i Schmidhuber, J.. Long Short-Term Memory. *Neural Computation*. 1997.
- [8] Google Research. TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. *arXiv preprint arXiv:1603.04467v2*. 2016.
- [9] Kim, Ji-Sung. *Deep learning driven jazz generation*[en línia]. Edició 2016. [consulta: gener de 2017]. Disponible a: <https://deepjazz.io/>
- [10] Chollet, F.. *Keras*[en línia]. Edició 2015. [consulta: gener de 2017]. Disponible a: <https://keras.io/>
- [11] Universitat de Montreal. Theano: A Python framework for fast computation of mathematical expressions. *arXiv preprint arXiv:1605.02688v1*. 2016.
- [12] Sony CSL Research Laboratory. *AI makes pop music in different music styles. Flow Machines* [online]. 2016. [consulta: gener de 2017]. Disponible a: <http://www.flow-machines.com/ai-makes-pop-music/>
- [13] Hayes, B.. First Links in the Markov Chain. *American Scientist*. 2013, Vol. 101.
- [14] van den Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., i Kavukcuoglu, K.. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499v2*. 2016.
- [15] Nayebi, A. i Vitelli, M.. GRUV: Algorithmic Music Generation using Recurrent Neural Networks. 2015.

- [16] Burk, P., Plansky, L., Repetto, D., Roberts, M. i Rockmore, D.. *Music and Computers. A theoretical and historical approach*[en línia]. Edició 2011. [Consulta: gener de 2017] Disponible a: <http://cmc.music.columbia.edu/musicandcomputers/>
- [17] Baraniuk, R.. *Signals and System*. The Connexions Project. 2003.
- [18] Shewhart, W.A. i Wilks S. S.. *Fourier Analysis of Time Series, An Introduction*. 2n edició. A Wiley-Interscience Publication, 2000. ISBN 0-471-88948-2.
- [19] Nielsen, M.. *Neural Networks and Deep Learning*[en línia]. Edició 2016. [Consulta: gener de 2017] Disponible a: <http://neuralnetworksanddeeplearning.com/>
- [20] Basheera, I. A. i Hajmeer, M.. Artificial neural networks: fundamentals, computing, design, and application. *Journal of Microbiological Methods*. 2000, Vol. 43, pp. 3-31.
- [21] Graupe, D.. *Principles of Artificial Neural*. 2n edició. World Scientific Publishing, 2007. ISBN-13 978-981-270-624-9.
- [22] Jarrett, K., Kavukcuoglu, K., Ranzato, M., i LeCun, Y.. What is the best multi-stage architecture for object recognition? *IEEE Proc. International Conference on Computer Vision*. 2009, pp. 2146–2153.
- [23] Hinton, G., Deng, L., Yu, D., Dahl, G.E., Mohamed, A., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T.N. i Kingsbury, B.. Deep Neural Networks for Acoustic Modeling in Speech Recognition. *IEEE Signal Processing Magazine*. 2012.
- [24] McCulloch, W. i Pitts, W.. A logical calculus of the ideas immanent in nervous activity. *Bulletin of mathematical biophysics*. 1943, Vol. 5, N. 4, pp. 115–133.
- [25] Nair, V. i Hinton, G. E.. Rectified linear units improve restricted boltzmann machines. *Actes de la 27a Conferència Internacional de Machine Learning*. 2010, pp. 807–814.
- [26] Basu, J. K., Bhattacharyya, D. i Kim, T.. Use of Artificial Neural Network in Pattern Recognition. *International Journal of Software Engineering and Its Applications*. 2010, Vol. 4, N. 2, pp. 23-34.
- [27] Hill, T., Marquez, L., O'Connor, M. i Remus, W.. Artificial neural network models for forecasting and decision making. *International Journal of Forecasting*. 1994, Vol. 10.
- [28] Kalogirou, A.. Artificial neural networks in renewable energy systems applications. *Renewable and Sustainable Energy Reviews*. 2001, Vol. 5, pp. 373–401.

- [29] LeCun, Y., Bottou, L., Bengio, Y. i Haffner, P.. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*. 1998, Vol. 86, N. 11, pp. 2278–2324.
- [30] Krizhevsky, A., Sutskever, I. i Hinton, G. E.. Imagenet classification with deep convolutional neural networks. *En Advances in neural information processing systems*. 2012.
- [31] Rosenblatt, F.. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review*. 1958, Vol. 65, N. 6, pp. 386-408.
- [32] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., i Salakhutdinov, R.. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*. 2014, Vol. 15, N. 1, pp. 1929–1958.
- [33] Rumelhart, D.E., Hinton, G.E., i Williams, R.J.. Learning internal representations by error propagation. *Parallel Distributed Processing*. 1986, Vol. 1: Foundations. MIT Press, Cambridge, MA.
- [34] Yan-Tak Ng, A.. Sparse autoencoder. *Lecture notes Stanford*. 2011.
- [35] Google Brain Team. *Understanding LSTM Networks* [en línia]. Edició 2015 [consulta: gener de 2017]. Disponible a: <http://colah.github.io/posts/2015-08-Understanding-LSTMs/>
- [36] Kingma, D. i Ba, J.. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980v8*. 2015.
- [37] Sutskever, I., Vinyals, O. i Ve. Le, Q.. Sequence to Sequence Learning with Neural Networks. *arXiv preprint arXiv:1409.3215v3*. 2014.
- [38] Owens, A., Isola, P., McDermott, J., Torralba, P., H. Adelson, E., i T. Freeman, W.. Visually Indicated Sounds. *arXiv preprint arXiv:1512.08512v2*. 2016.