UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE INGENIERÍA ELÉCTRICA

DECONVOLUCIÓN EN AUDIO UTILIZANDO MODELOS BASADOS EN MACHINE Y DEEP LEARNING

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL ELÉCTRICO

DIEGO JAVIER LEÓN GONZÁLEZ

PROFESOR GUÍA:
FELIPE TOBAR HENRÍQUEZ

MIEMBROS DE LA COMISIÓN:
JORGE SILVA SÁNCHEZ
JOAQUÍN FONTBONA TORRES

SANTIAGO DE CHILE
2021

RESUMEN DE LA MEMORIA PARA OPTAR
AL TÍTULO DE INGENIERO CIVIL ELÉCTRICO
POR: **DIEGO JAVIER LEÓN GONZÁLEZ**
FECHA: 2021
PROF. GUÍA: FELIPE ARTURO TOBAR HENRÍQUEZ

DECONVOLUCIÓN EN AUDIO UTILIZANDO MODELOS BASADOS EN MACHINE Y DEEP LEARNING

El problema de dereverberación es un problema con gran potencial de investigación en ciencias e ingeniería. El problema puede modelarse como uno de deconvolución y existe una vasta cantidad de modelos que buscan abordarlo. Una cantidad importante de métodos de dereverberación hacen uso de *Machine Learning* para una extracción de características espectrales para obtener una señal de salida dereverberada. Ideas como esta han evolucionado rápidamente y dicha extracción de características actualmente llega a ser automática dentro del aprendizaje, lo cual es un cambio de paradigma de *Machine* a *Deep Learning*.

En este trabajo se seleccionan 3 enfoques dereverberativos: *Context Information/Window*, *Late Reverberation Supression* y *Image to Image*. Estos 3 enfoques están representados por 7 modelos en este trabajo, las cuales son arquitecturas neuronales de *Context Information* (una con un MLP y otra con LSTM), *Late Reverberation Supression* LSTM, Weighted Prediction Error (WPE) en una variante de implementación y resolución denominada FD-NDLP y una arquitectura U-net. La arquitectura U-net se utiliza en 3 variantes, 2 de ellas solo con Error Cuadrático Medio (MSE) como función de pérdida y otra con aprendizaje de *Generative Adversarial Networks* (GAN). De las 2 arquitecturas U-net entrenadas solo con MSE, una es idea propia y utiliza *Late Reverberation Supression*. De todos estos modelos, se busca analizar en profundidad la robustez de estos frente al ruido, al nivel de reverberación y capacidad de generalización.

Los modelos escogidos son entrenados utilizando audios reverberados simulados mediante convolución. Se hace uso del dataset de audios de voz *LibriSpeech* y de los datasets OMNI y MARDY para respuestas al impulso. Se evalúan los modelos en diferentes escenarios de ruido, en diferentes niveles de reverberación y en ambientes reales, mediante audios retransmitidos de *LibriSpeech*. Se utilizan métricas de calidad, inteligibilidad y reverberación de señales de voz, con énfasis en la métrica *Speech to Reverberation Modulation Energy Ratio* (SRMR), la cual es especializada en reverberación.

La arquitectura U-net con y sin aprendizaje GAN en este problema es una idea ya existente, pero se incursiona aún más en ella proponiendo una variante de esta misma, la cual utiliza *Late Reverberation Supression*. Los resultados obtenidos dan cuenta que una arquitectura U-net con aprendizaje GAN tiene los mejores resultados en la mayoría de escenarios, seguida de la variante de U-net propuesta. El aprendizaje GAN en U-net permitió una mejora cuantitativa significativa en comparación a entrenar la misma arquitectura utilizando solamente el MSE. Las arquitecturas U-net entrenadas en este trabajo tienen una amplia capacidad de generalizar sobre datos simulados y sobre datos reales. El modelo FD-NDLP (que es no supervisado) muestra resultados sobresalientes sobre varios modelos neuronales, pero no supera a las arquitecturas basadas en U-net.

Agradecimientos

Partiré agradeciendo a mi familia. A mis padres Héctor León y María González quienes me apoyaron con la elección de esta carrera, la cual no era de su gusto, por lo cual fue totalmente mi opción. Lamentablemente a mi papá no le alcanzó la vida para estar con mi familia actualmente, pero estoy seguro de que estaría feliz de verme finalmente con el título universitario. Mi papá fue quien más me apoyó, dando siempre ánimo si es que no me estaba yendo muy bien en algunos ramos y siempre preocupado que no me faltara nada desde mis tiempos en el colegio.

En cuanto a instituciones debo agradecer primero al liceo José Victorino Lastarria y sus profesores (los que conocí en el lapso 2011-2014). Fue aquí donde empecé mis intereses por el lado científico y de ingeniería y siento que la preparación que tuve para la universidad fue muy buena, pude enfrentar todo con más madurez, con mayor motivación y también aprender a levantarme si me va mal en algo. En la universidad debo agradecer a 3 departamentos que más aportaron en mi formación y sus respectivos profesores: DIE en Área de Inteligencia Computacional y también la de Control de Sistemas, DCC y DFI.

Por último, gracias a la comisión completa de esta memoria. Gracias por colaborar en todo este proceso de salida de la universidad.

Tabla de Contenido

Índice de Tablas

Índice de Ilustraciones

Capítulo 1

Introducción

La reverberación consiste en reflexiones de señales auditivas sobre superficies y objetos dentro de una sala, recinto o habitación. Esto supone un problema, debido a que degrada la calidad de la señal para ser procesada y escuchada [1]. En concreto, afecta la inteligibilidad de la señal, es decir, la capacidad de poder comprender esta señal en términos linguisticos [1, 2]. Lo anterior dificulta apreciablemente el reconocimiento automático de voz de una persona, clasificación de sonidos, entre muchas otras aplicaciones. La reverberación matemáticamente se puede modelar como una convolución de una señal auditiva con la respuesta al impulso de una sal o habitación, además de ruido aditivo ambiental. De esta manera, el problema de dereverberación se puede modelar como uno de deconvolución en el contexto de audio. Este problema consiste en recuperar una señal auditiva dereverberada, cuando el filtro (respuesta la impulso) en cuestión es desconocido.

Una elevada cantidad de estrategias se han propuesto para dereverberar una señal de audio, para así obtener una mejora en la calidad e inteligibilidad. Los métodos más antiguos corresponden a filtros estadísticos con parámetros aprendibles como puede ser el denominado *Weight Prediction Error* (WPE) [28] que mediante Máxima Verosimilitud es capaz de aprender pesos que se operan con observaciones de una señal reverberada para lograr obtener una señal dereverberada. También se han planteado filtros que utilizan un arreglo espacial de micrófonos como algoritmo para dereverberación [3]. Sin embargo este último es útil solo en contextos multi-micrófonos.

En el contexto de reconocimiento de voz, se han planteado también numerosas estrategias neuronales de lidiar con la reverberación. Una primera manera es utilizar una red neuronal sobre características espectrales de una señal de audio en dominio espectral [4, 5]. Sin embargo, con esta metodología no se ocupan de recuperar la señal de audio en dominio temporal. En este trabajo se busca un enfoque diferente donde sea posible recuperar una señal temporal dereverberada, ya que resuelve el problema de una manera más general, de modo que pueda ser de utilidad en un mayor número de aplicaciones.

En el contexto de recuperar una señal de audio dereverberada, ha tenido relevancia y popularidad una red neuronal dereverbativa que es un *MultiLayer Perceptron* (MLP) [6] y una modificación a esta red incorpora capas *Long Short-term Memory* (LSTM) que es un bloque de redes neuronales recurrentes (RNN) que suele entregar mejores resultados que MLP en este contexto [2, 30]. Estas redes operan a nivel de frames de espectrogramas. Un

espectrograma es una representación bilinear donde la potencia espectral de una señal no estacionaria es representada tanto en tiempo como en frecuencia [11]. La reverberación en el dominio espectral (espectrograma) "esparce" el contenido de potencia espectral a lo largo del tiempo [6]. Por esta razón, las arquitecturas de redes neuronales basadas en MLP y LSTM en vez de usar el espectrograma completo, toman un determinado número de *frames* (cuadros temporales) y los mapean a un *frame* dereverberado.

Existen también arquitecturas diferentes a MLP y LSTM que ocupan el espectrograma completo como *input* para realizar el proceso de deconvolución. Sin embargo, la arquitectura crece en cuanto a complejidad y parámetros aprendibles. Una red neuronal convolucional (CNN) se puede utilizar sobre el espectrograma completo, para lo cual se construye la estructura de un autoencoder con redes profundas como lo puede ser la arquitectura U-Net [12, 8]. Con una estrategia similar, se puede construir una arquitectura *Generative Adversarial Network* (GAN) donde la red generadora pueda aprender un mapeo de un espectrograma reverberado a uno dereverberado [8]. La ventaja de estas recientes arquitecturas es que al operar sobre el espectrograma completo no requieren de extracción de características y explotan de una forma más completa toda la información disponible del espectrograma.

1.1. Objetivo General

Este trabajo tiene como objetivo evaluar diferentes enfoques basados en Machine y Deep Learning en el problema de dereverberación en señales de voz.

1.2. Objetivos Específicos

Entre los objetivos específicos de este trabajo, cabe destacar los siguientes:

- El problema de deconvolución en audio en primera instancia no tiene relación alguna con Machine y Deep Learning. De esta manera, se busca formular el problema como uno adecuado para resolver por esta vía mediante diferentes enfoques.
- Para entrenar arquitecturas de redes neuronales se requiere de un dataset. Por lo anterior, se busca generar audios de voz reverberados a partir de datasets de audio existentes.
- Caracterizar los modelos entrenados analizando su robustez frente a ruido y frente a diferentes niveles de reverberación.
- Como se ha mencionado anteriormente, se busca contribuir a mejorar la calidad e inteligibilidad de una señal temporal a partir de la dereverberación. Por ende, se requiere de métricas que permitan evaluar ambos aspectos de una señal. Se plantea como objetivo explorar indicadores que den cuenta del nivel de reverberación en una señal temporal.

1.3. Contribución de este trabajo

Este trabajo busca dar una visión global de cómo formular el problema de dereverberación desde un punto de vista de Machine y Deep Learning. Para esto, se da un recorrido por diferentes métodos, haciendo énfasis en redes neuronales, pero no limitando el problema solo a estas arquitecturas. Lo anterior incluye las ventajas y desventajas de cada uno de estos modelos, haciendo énfasis en robustez de estos frente al ruido y al nivel de reverberación.

El modelo con mejores resultados corresponde a una arquitectura de autoencoder con *skip connections* denominada U-net, la cual con aprendizaje GAN se mostró sobresaliente. Sin embargo, la contribución no es utilizar esta red para este problema específico, ya que esta es una idea ya existente [8]. En lugar de ello, se utiliza esta idea para entrenar con un conjunto de entrenamiento más completo, con mayor variabilidad de ruido adicionado y del nivel de reverberación. Con esta forma más completa de entrenamiento, se puede evaluar en diferentes contextos, de modo de obtener una idea detallada de su capacidad de generalizar y así compararla con otros enfoques y con otros modelos. Adicionalmente, se propone una variante de U-net utilizando *Late Reverberation Supression* [2], es decir, se combinan 2 ideas de la literatura para generar un método adicional, el cual en general se obtuvo que obtiene mejores resultados que U-net tradicional.

1.4. Estructura del Trabajo de Título

Este trabajo se encuentra estructurado en 5 capítulos:

- Capítulo 1 Introducción: es el presente capítulo.
- Capítulo 2 Marco Teórico: entrega los fundamentos necesarios para comprender representaciones de audio y formulación del problema de dereverberación. Además, expone herramientas básicas de Machine y Deep Learning y cómo son aplicadas específicamente para el problema de dereverberación.
- Capítulo 3 Metodología: expone cual fue el procedimiento realizado para obtener resultados. Detalla las bases de datos, cómo se procesaron dichos datos, qué modelos se escogieron, cómo fueron implementados y cómo se evalúan.
- Capítulo 4 Resultados y Análisis: se enfoca en mostrar tablas y figuras con su respectivo análisis, que den cuenta de cómo se desempeñaron los modelos entrenados en diferentes contextos.
- Capítulo 5 Conclusiones: Se expone sobre el cumplimiento de objetivos, qué se puede concluir a grandes rasgos de los resultados obtenidos y además se plantea cómo se podría extender este trabajo en el futuro.

Capítulo 2

Marco Teórico y estado del arte

2.1. Representaciones de audio

En el problema de dereverberación, el cual se modela como uno de deconvolución, es necesario utilizar representaciones gráficas de un audio. Con este propósito en las subsecciones siguientes se explica en detalle qué tipo de representaciones de audio existen y cómo son utilizadas.

2.1.1. Formas de onda

Este tipo de representación de audio sirven para mostrar amplitud en función del tiempo. La Figura 2.1 muestra un ejemplo de señal de audio de voz [1] muestreado a 16 kHz.

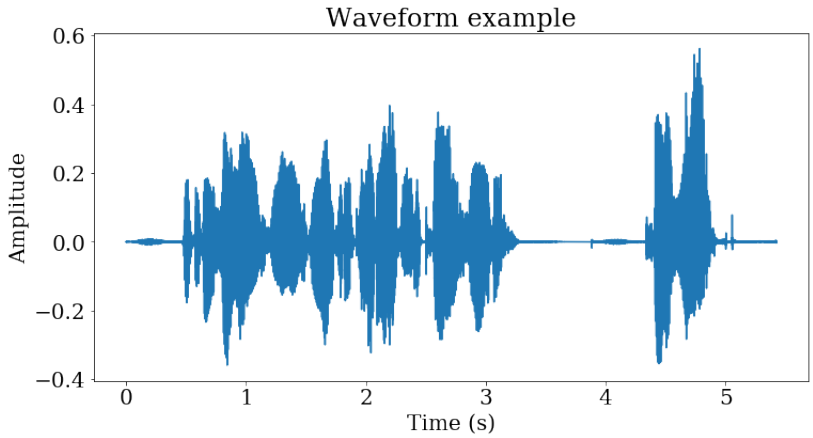


Figura 2.1: Señal de audio en dominio temporal. Elaboración propia

Este tipo de representación no posee información del contenido armónico del audio, sino de alzas o bajas de amplitud en el tiempo. También esta representación es sensible al ruido y a la reverberación, la cual se detallará más adelante. Esta forma de onda en el tiempo es la base sobre la que se puede construir una representación en frecuencia.

[1] Gráfico de elaboración propia, pero el audio es tomado de LibriSpeech Dataset http://www.openslr.org/12/

2.1.2. Respuestas al impulso en audio

La respuesta al impulso de una sala u habitación (RIR) es una señal de representación temporal (forma de onda) que depende de las dimensiones, cantidad de superficies presentes, materiales, entre otros factores. La Figura 2.2 muestra las partes principales de una RIR [2].

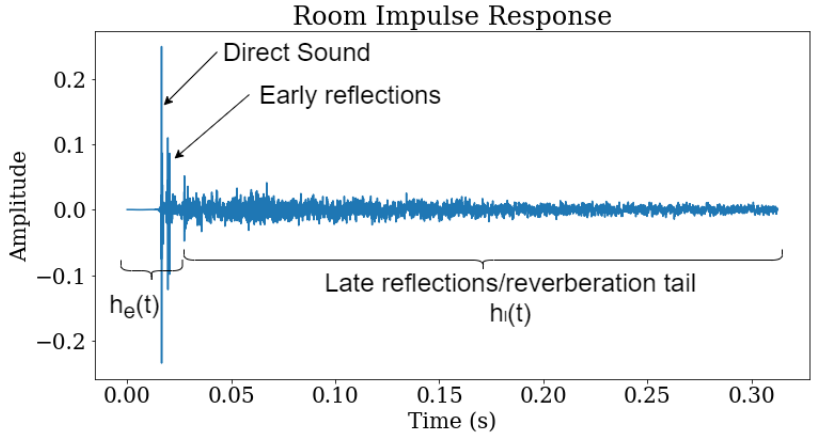


Figura 2.2: Ejemplo de RIR. Elaboración propia.

Como se puede ver, se marcan 3 partes:

- *Direct Sound*/Sonido directo: se puede entender como el primer peak que se aprecia en la señal temporal, el cual corresponde al impulso propiamente tal.
- *Early reflections*/Reflexiones tempranas: peaks más pequeños posteriores al sonido directo. Son reflexiones del sonido directo sobre superficies cercanas.
- *Late reflections*/reflexiones tardías: reflexiones posteriores a las reflexiones tempranas y que se van atenuando en amplitud con el transcurso del tiempo hasta anularse. Estas reflexiones son las principales causantes de que un sonido sea audiblemente reverberado, por lo que una gran variedad de métodos de reverberación se ocupan de suprimir las reflexiones tardías.

En la Figura 2.2 se puede ver que se hizo una separación temporal de la RIR en $h_l(t)$ que son las reflexiones tardías o *Late Reflections* y el resto de la RIR hacia la izquierda denotada por $h_e(t)$. La reverberación resultante utilizando una RIR, se puede modelar como una convolución, ya que la habitación o sala es modelada como un sistema lineal e invariante en el tiempo (LTI) [3]. Lo anterior permite escribir la convolución $(x * h)(t)$ de la siguiente forma:

$$y(t) = (x * h)(t) + \eta(t) = \sum_{\tau=-\infty}^{t} x(t)h_e(t-\tau) + \sum_{\tau=-\infty}^{t-t_e} x(t)h_l(t-\tau) + \eta(t). \tag{2.1}$$

La variable t_e separa ambas partes de la RIR y $\eta(t)$ es ruido aleatorio. Notar que el segundo término contiene netamente a las reflexiones tardías, las cuales son las principales responsables de que el sonido sea audiblemente reverberado. Dado lo anterior, existen estrategias

[2] Gráfico de elaboración propia, pero el audio tomado de base de datos de RIRs medidas profesionalmente, University of London http://isophonics.net/content/room-impulse-response-data-set

[3] Ver detalle matemático de sistemas LTI en el anexo A

de dereverberación que buscan sustraer las reflexiones tardías, para lo cual teóricamente la estrategia corresponde a restar de la ecuación anterior al segundo término.

Es importante mencionar que es frecuente que una RIR sea caracterizada por el llamado **tiempo de reverberación** T_{60}, el cual es definido como el tiempo que le toma a la señal RIR en decaer en 60 dB por debajo del nivel del sonido directo [15, 16]. El tiempo de reverberación de una RIR depende de la geometría de la habitación, de la reflectividad sobre superficies cercanas y de la posición del receptor [15]. Este tiempo se utiliza como una forma de cuantizar la "cantidad de reverberación" presente en la señal, ya que mientras mayor es su valor, las reflexiones tardías tienen lugar por más tiempo. Usualmente el valor de T_{60} ronda el intervalo [0.2, 1] segundos.

Una respuesta al impulso se mide profesionalmente como se ilustra en la Figura 2.3, la cual corresponde a la medición dentro de una salón de clases [4]

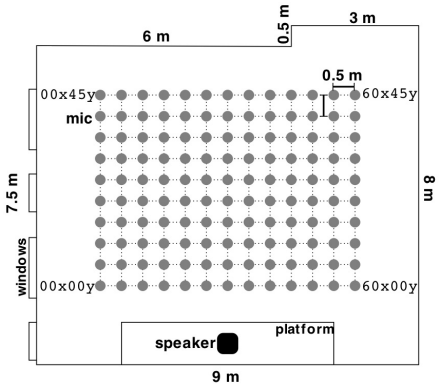


Figura 2.3: Medición de una respuesta al impulso.

La señal de impulso es similar a un disparo al oído y se emite desde el speaker marcado en negro. De manera estratégica, se van tomando diferentes realizaciones también llamadas *utterances* desde diferentes puntos marcados en la sala. Por cada punto marcado en gris en la imagen se ubica el micrófono y se toma una *utterance*. Por cada posición se pueden ir captando diferentes reflexiones del sonido, dependiendo de las superficies más cercanas. Usualmente los datasets de RIRs guardan utterances de una sala para diferentes canales de micrófonos y en diferentes posiciones.

[4] Extraída de http://isophonics.net/content/room-impulse-response-data-set, donde se ilustra el dataset de RIR's.

2.1.3. Representación de audio en frecuencia

El mapeo de dominio de tiempo discreto para una señal $x(n)$ a dominio de frecuencia $X(\omega)$ está dado por la siguiente expresión matemática:

$$X(\omega) = \sum_{n=0}^{N-1} x(n) \mathrm{e}^{-j\frac{k\omega n}{N}}, \tag{2.2}$$

o de manera equivalente usando $\omega = 2\pi f$

$$X(f) = \sum_{n=-\infty}^{\infty} x(n) \mathrm{e}^{-j\frac{2\pi f k n}{N}}. \tag{2.3}$$

$T_s = \frac{k}{N}$ representa un periodo de muestreo.

El propósito de este mapeo desde el dominio del tiempo, es visualizar el contenido armónico de la señal temporal. La cantidad $|X(f)|^2$ recibe el nombre de **densidad espectral de potencia**. En la Figura 2.4 [5] se puede ver la conexión entre el dominio temporal y el dominio de frecuencia, se ilustra que señales sinusoidales tienen una representación de impulso en el dominio frecuencial.

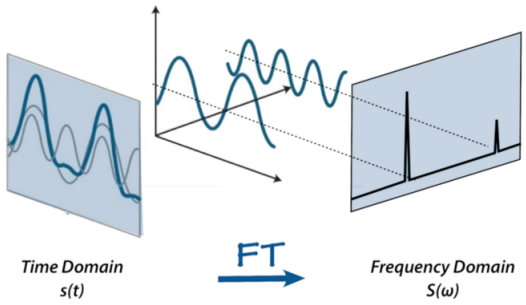


Figura 2.4: Esquema Transformada de Fourier.

Una propiedad importante de la transformada de fourier es que dadas 2 señales temporales $x_1(n)$, $x_2(n)$, entonces la transformada de Fourier de la convolución de ambas señales $(x_1 * x_2)(n)$, está dada por la multiplicación $X_1(f)X_2(f)$, donde $X_i(f)$ $i = 1, 2$ son las respectivas transformadas de Fourier. En resumen, si $\mathcal{F}[.]$ representa la aplicación de transformada de Fourier, la propiedad anterior se escribe de la siguiente manera:

$$\mathcal{F}[(x_1 * x_2)(n)] = \mathcal{F}[x_1(n)]\mathcal{F}[x_2(n)] = X_1(f)X_2(f). \tag{2.4}$$

La representación espectral a partir de la transformada de Fourier permite visualizar cómo varía la densidad espectral de potencia a lo largo de un eje de frecuencias, pero no permite visualizar cómo varia dicho contenido en el tiempo. Con este propósito, se busca una representación que muestre cómo varía la densidad espectral de potencia en función de la frecuencia y a la vez del tiempo. Esta constituye la base de una representación denominada espectrograma.

[5] Tomada de https://aavos.eu/glossary/fourier-transform/

2.1.4. Espectrograma

El espectrograma es una distribución bilinear tiempo-frecuencia donde la potencia espectral de una señal no estacionaria es representada tanto en tiempo como en frecuencia [11]. En términos matemáticos, el espectrograma es la magnitud al cuadrado de la transformada de fourier discreta, pero incorpora una ventana temporal tal como se muestra a continuación:

$$S(n,f) = |\sum_{m=0}^{N-1} x(m)w(m-n)\mathrm{e}^{-j2\pi fmT_s}|^2. \tag{2.5}$$

En la ecuación anterior, $w(n)$ representa una ventana de observación temporal. El término dentro del módulo, es decir, el término mostrado en la siguiente ecuación:

$$X(n,f) = \sum_{m=0}^{N-1} x(m)w(m-n)\mathrm{e}^{-j2\pi fmT_s}. \tag{2.6}$$

Se denomina **transformada de fourier de tiempo reducido (STFT)**. Para obtener este tipo de transformada a partir de la señal temporal discreta se usa un procedimiento como se muestra en la Figura 2.5 [6]

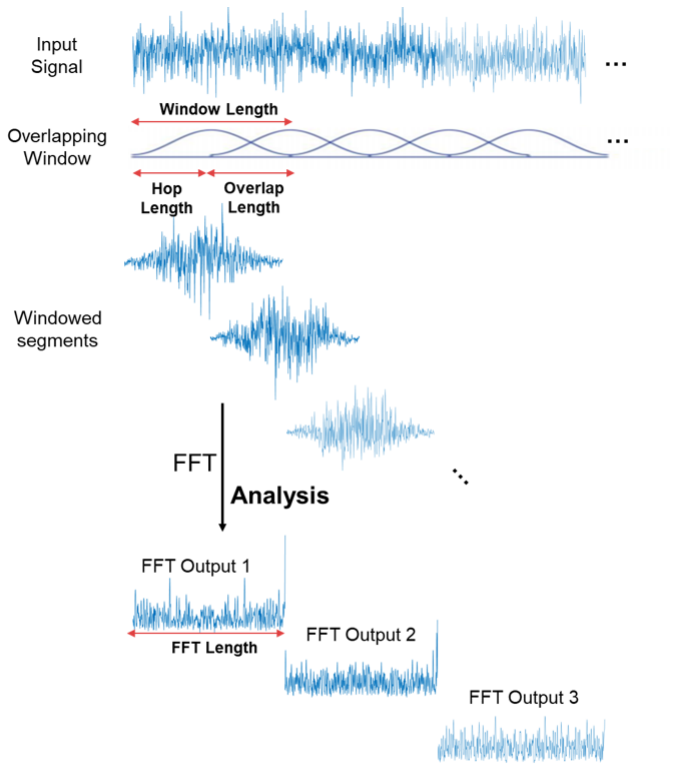


Figura 2.5: Esquema de obtención de STFT.

[6] Tomada de https://www.mathworks.com/help/dsp/ref/dsp.stft.html

Como se puede observar el procedimiento que se sigue es el siguiente:

- Se parte de una señal temporal y se toman ventanas de esta señal de una cierta longitud (*window length* en la figura).
- Estas ventanas temporales tienen una cierta cantidad de puntos comunes, los cuales constituyen un traslape u *overlap*. La longitud donde 2 ventanas consecutivas se traslapan se denomina longitud de *overlap* y la longitud donde no hay traslape con otra ventana se denomina distancia *hop*. Se puede ver en la figura 2.5 que la distancia de *hop* se puede ver como la longitud de la ventana menos la longitud de *overlap*.
- Por cada ventana se calcula la transformada de fourier, resultando también en un espacio en frecuencia separada por ventanas. De esta manera se puede generar un valor de densidad espectral por cada par de ventanas tiempo-frecuencia.

Cada cuadro de tiempo donde es calculada la STFT se le denomina *frame* y la longitud de cada *frame* determina un número de *bins*.

2.1.5. Escala logarítmica y escala de Mel

Con la metodología planteada anteriormente se puede calcular densidad espectral de potencia y visualizar su evolución tanto en tiempo como en frecuencia. Sin embargo, esta se puede visualizar de una forma débil y tenue. Para modificar esto, se convierte la densidad espectral a escala logarítmica y se cambia de escala el eje de frecuencia.

El cambio de escala del eje de frecuencia se realiza debido a que el ser humano no percibe frecuencias en una escala lineal. *"Somos hábiles detectando diferencias en bajas frecuencias que en altas frecuencias"* [7]. Por esta razón se utiliza una transformación a las frecuencias, la cual sigue la siguiente forma [14]:

$$M(f) = (2595\text{Hz}) \text{ x } \log_{10}(1 + \frac{f}{700}). \tag{2.7}$$

Como se puede ver, es una transformación logarítmica y su aspecto gráfico se muestra en la Figura 2.6 (a).

Esta escala de Mel como se mencionó anteriormente se utiliza para realizar un mapeo de frecuencias altas a bajas, de modo de concentrar el contenido espectral en un intervalo más reducido. Notar en la Figura 2.6 que un intervalo de 10000 Hz en escala lineal de frecuencia en esta escala de Mel ronda los 3000 Hz, es decir, lo reduce a menos de 1/3.

Utilizando esta escala, es de importancia en el contexto de espectrogramas un banco de filtros conocido como *Mel Scale Filterbank*[8]. Este conjunto de filtros pasa banda es aplicado *frame* por *frame* del espectrograma, buscando adecuar su contenido a un número dado de bins. Se utilizan tantos filtros como número de bins sean deseados en la salida. Los filtros dentro del *filterbank* son de espectro triangular. En la Figura 2.6 (b) se pueden ver 10 filtros

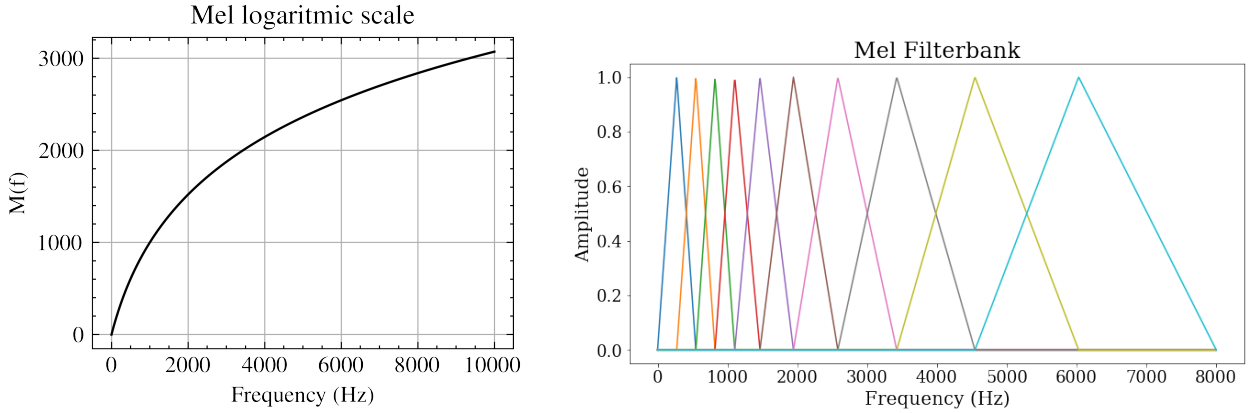
[7] Extracto de https://medium.com/analytics-vidhya/understanding-the-mel-spectrogram-fca2afa2ce53
[8] Gran parte de la notación es tomada de https://haythamfayek.com/2016/04/21/speech-processing-for-machine-learning.html

para obtener 10 *bins* de salida. Para el cálculo de los filtros triangulares se utiliza el *sampling rate* f_s de la señal de entrada (16 kHz en la figura 2.6(b)) y el número de puntos utilizados en la transformada de fourier N_{fft}. Los filtros triangulares se ubican en un intervalo de frecuencia entre 0 Hz y $f_s/2$. El m-ésimo filtro se puede describir mediante $H_m(k)$ usando las variables mencionados anteriormente según las siguientes ecuaciones:

$$H_m(k) = \begin{cases} 0 & \text{si } k \leq f(m-1) \\ \frac{k-f(m+1)}{f(m)-f(m-1)} & \text{si } f(m-1) \leq k \leq f(m) \\ \frac{f(m+1)-k}{f(m+1)-f(m)} & \text{si } f(m) \leq k \leq f(m+1) \\ 0 & \text{si } k \geq f(m+1), \end{cases} \tag{2.8}$$

$$f(i) = floor(\frac{(N_{fft}+1)h_{\text{mel}}(i)}{f_s}). \tag{2.9}$$

Dados el número de filtros totales (triángulos en la Figura 2.6), se les asigna una frecuencia en el rango [0, $f_s/2$], donde este intervalo es dividido en cuantas partes como número de filtros. Todas las frecuencias asignadas son transformadas usando escala de Mel, donde para el i-ésimo filtro le corresponde una frecuencia $h_{\text{mel}}(i)$ en Hz. Por último, es importante mencionar que los filtros usualmente son normalizados, de modo que los triángulos de la Figura 2.6(b) tienen la misma altura.



(a) Escala logarítmica de Mel.

(b) Mel *Filterbank* utilizando 10 filtros.

Figura 2.6: Escala de Mel y *filterbank* asociado.

Adicionalmente a la escala de Mel y el banco de filtros, un procesamiento habitual al contenido de densidad espectral de potencia es convertirla también a una escala logarítmica, cuya salida $S_{out}(n, f)$ tiene la forma siguiente ante una entrada $S_{in}(n, f)$:

$$S_{out}(n, f) = 10 \text{ x } \log_{10}(\frac{S_{in}(n, f)}{S_{ref}}). \tag{2.10}$$

Bajo esta transformación $S_{out}(n, f)$ está en decibels (dB). La variable S_{ref} constituye una referencia, puede ser por ejemplo la mediana de todos los valores espectrales $S(n, f)$ o también puede ser el promedio.

2.2. Reverberación en audio

2.2.1. Reverberación en la forma de onda

Sea $h(t)$ la respuesta al impulso de una sala (RIR), sea $x(t)$ una señal auditiva temporal y sea $\eta(t)$ ruido aleatorio aditivo. La señal de audio reverberada $y(t)$ construida a partir de $x(t)$ se puede modelar según la siguiente ecuación:

$$y(t) = (x * h)(t) + \eta(t). \tag{2.11}$$

El símbolo $*$ representa convolución, ya que la sala o habitación se modela como un sistema LTI, por lo que la respuesta está completamente determinada por la convolución con la respuesta al impulso.

Habitualmente, para cuantificar el nivel de ruido se utiliza una variable denominada *Signal to Noise Ratio* (SNR), la cual está dada por la siguiente expresión:

$$\text{SNR} = 10\text{x}\log_{10}(\frac{P_{\text{signal}}}{P_{\text{noise}}})[\text{dB}]. \tag{2.12}$$

En la ecuación anterior, P_{signal} es la potencia de la señal que es contaminada por ruido y P_{noise} es la potencia de la señal de ruido. Si se poseen $\mathcal{T}$ observaciones $x_1, x_2, ..., x_{\mathcal{T}}$ de una señal temporal x, entonces su potencia P es matemáticamente definida como $P = \sum_{i=1}^{\mathcal{T}} |x_i|^2$. De esta manera, utilizando un valor de SNR y la potencia de una señal *source* (señal a contaminar con ruido), se puede calcular la potencia de ruido implicada y en base a esta construir una señal temporal de ruido. Es de esta forma que surge el denominado *Additive White Gaussian Noise* (AWGN) generado a partir de un valor de SNR, donde el ruido está dado por $\eta(t) = \sqrt{P_{\text{noise}}} z(t)$ donde $z(t) \sim \mathcal{N}(0, 1)$.

La convolución y por ende reverberación en audio tiene un efecto tanto en representación temporal como frecuencial. En la Figura 2.7 se puede visualizar el efecto de la reverberación en la forma de onda representación de audio temporal del mismo audio de voz que se utilizó en las secciones anteriores. Para reverberar se usó una respuesta al impulso de una sala de clases [9].

[9] Tomado de base de datos de RiRs medidas profesionalmente, University of London http://isophonics.net/content/room-impulse-response-data-set

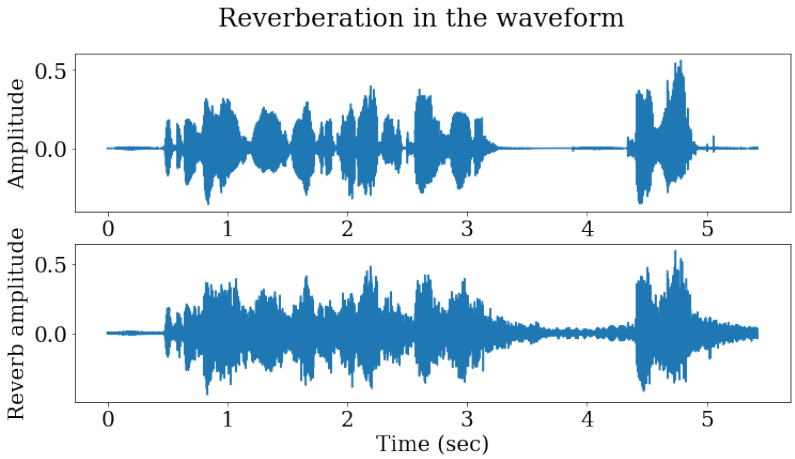
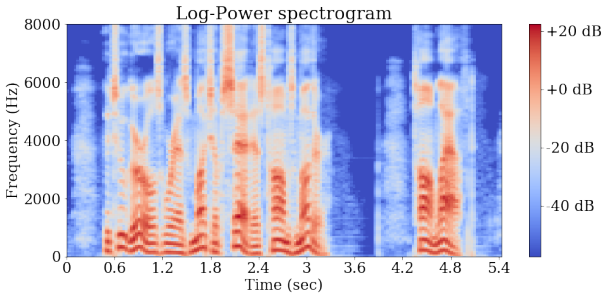


Figura 2.7: Reverberación de voz en dominio temporal. Elaboración propia

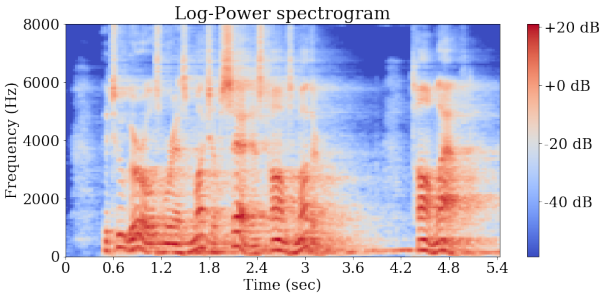
Como se puede ver la forma de onda en las zonas con mayor amplitud tiende a permanecer invariante, pero en las zonas con menor amplitud, esta tiende a aumentar considerablemente.

2.2.2. Reverberación en el espectrograma

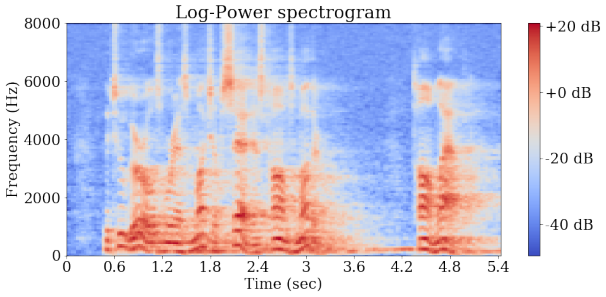
En los espectrogramas se puede visualizar la reverberación y ver sus efectos en el dominio de tiempo-frecuencia. La Figura 2.8 muestra los espectrogramas de Mel de una señal limpia, reverberada sin considerar ruido y luego considerando ruido con SNR = 30dB.



(a) Espectrograma sin reverberar.



(b) Espectrograma reverberado sin añadir ruido junto a la convolución.



(c) Espectrograma reverberado y ruidoso con SNR = 30dB.

Figura 2.8: Efectos de la reverberación en el espectrograma. Elaboración propia.

Para los espectrogramas se utilizó una longitud de ventana y hop de 2048 y 512 respectivamente para calcular STFT. Para transformar a dB se utiliza de referencia el promedio de los valores de potencia espectral.

Como se puede ver el efecto en el espectrograma de la reverberación es diferente a lo que se observa en la forma de onda. Se puede ver que las frecuencias excitadas de la señal en el espectrograma (zonas con mayor dB) tienden a ser las mismas, pero en el eje del tiempo el contenido de potencia espectral se expande, se estira o se "prolonga" horizontalmente. Esta observación es clave para algunas estrategias de dereverberación que se describen más adelante en este trabajo. El problema de la reverberación es que degrada la calidad y en el caso de voz la inteligibilidad de la señal [1, 2]. Para plantear una estrategia de dereverberación, se debe tener en cuenta que la respuesta al impulso es completamente desconocida.

Se puede observar adicionalmente en la Figura 2.8 que el efecto del ruido más notorio visualmente es que las zonas del espectrograma de menor potencia espectral aumentan de magnitud (zonas en tonos de azul).

2.3. Métodos no supervisados de dereverberación

Dado el problema que representa la reverberación, se han planteado una gran variedad de métodos de dereverberación, es decir, contribuir a mejorar la calidad e inteligibilidad de una señal de audio buscando reducir el nivel de reverberación. En esta sección se presenta el conocido método *Weighted Prediction Error* y una variante llamada *Frequency Domain Normalized Delayed Linear Prediction* que permite ventajas de implementación. Este filtro se denomina no supervisado debido a que no necesita comparar la señal a dereverberar con otra señal limpia o *target*.

2.3.1. Weighted Prediction Error (WPE)

Este método data de 2008 [26] y es considerado uno de los métodos de dereverberación más populares [28]. Este método es considerado un filtro estadístico y a priori no se considera una estrategia que tenga relación alguna con *Machine y Deep Learning*. Años después (2019), el método WPE ha sido planteado como uno que puede ser resuelto con redes neuronales [27]. Sin embargo, lo anterior es solo una forma más práctica de resolver el mismo problema, ya que la estrategia de resolución del problema sigue siendo la misma. En este trabajo es de interés el método WPE con un enfoque que se denomina *Delayed Linear Prediction* [28], el cual se detalla más adelante.

El método consiste en buscar un filtro inverso con pesos aprendibles, tal que se obtenga una señal resultante dereverberada. El planteamiento considera la convolución con la cual es definida la reverberación mediante la señal *source* s (señal limpia de reverberación) y una respuesta al impulso $h_k^{(m)}$ $k = 0, ..., L_h - 1$ con L_h observaciones de h captadas por el m-ésimo micrófono. De esta manera la señal reverberada captada por el micrófono m está dada por la siguiente ecuación:

$$x_t^{(m)} = \sum_{k=0}^{L_h-1} h_k^{(m)} s_{t-k} + b_t^{(m)}. \tag{2.13}$$

El método busca un filtro $w_k^{(m)}$ $k = 0, ..., L_w - 1$, tal que se obtenga una señal dereverberada dada por:

$$y_t = \sum_{m=1}^{L_m} \sum_{k=0}^{L_w-1} w_k^{(m)} x_{t-k}^{(m)}. \tag{2.14}$$

En la ecuación anterior se consideran L_m micrófonos totales.

La variante a WPE mediante NDLP en dominio temporal (TD-NDLP) para encontrar los pesos $\bar{w_t}^{(m)} = [w_0^{(m)}, w_1^{(m)}, ..., w_{L_w-1}^{(m)}]$ parte por separar la señal $x_t^{(m)}$ en 2 partes, tal que $x_t^{(m)} = d_t^{(m)} + r_t^{(m)}$. La señal $d_t^{(m)}$ es la señal deseada (dereververada) y $r_t^{(m)}$ representa **reflexiones tardías** responsables de la reverberación. Ambas señales se pueden expresar matemáticamente en las siguientes 2 ecuaciones:

$$d_t^{(m)} = \sum_{k=0}^{D-1} h_k^{(m)} s_{t-k}, \tag{2.15}$$

$$r_t^{(m)} = \sum_{k=0}^{L_h-1} h_k^{(m)} s_{t-k}. \tag{2.16}$$

La variable D representa un índice que separa a la RIR h_k en reflexiones tempranas y tardías, conceptos los cuales ya fueron introducidos en secciones anteriores de este capítulo. Utilizando $d_t^{(m)}$ y $r_t^{(m)}$.

$$x_t^{(m)} = \sum_{m'=1}^{L_m} \sum_{k=0}^{L_c-1} x_{t-D-k}^{(m')} c_k^{(m',m)} + d_t^{(m)}. \tag{2.17}$$

O equivalentemente en forma vectorial:

$$x_t^{(m)} = (\bar{c}^{(m)})^T \bar{x}_{t-D} + d_t^{(m)}, \tag{2.18}$$

donde:

$\bar{x}_t = [(\bar{x}_t^{(1)})^T, (\bar{x}_t^{(2)})^T]^T$, $\bar{x}_t^{(m)} = [x_t^{(m)}, x_{t-1}^{(m)}, ..., x_{t-L_c+1}^{(M)}]^T$, $\bar{c}^{(m)} = [(\bar{c}^{(1,m)})^T, (\bar{c}^{(2,m)})^T]^T$ y $\bar{c}^{(m',m)} = [c_1^{(m',m)}, c_2^{(m',m)}, ..., c_{L_c}^{(m',m)}]^T$.

Se ha asumido $L_m = 2$, pero las ecuaciones anteriores se pueden generalizar para L_m arbitrario. Este enfoque, tal que se introduce un retraso temporal D en $\bar{x}_t$ es el denominado *Delayed Linear Prediction* y busca encontrar los coeficientes de regresión $\bar{c}^m$, los cuales en adelante se denotan como $\hat{\bar{c}}^{(m)}$ donde $\hat{}$ da cuenta que son valores estimados.

La resolución en dominio temporal (TD-NDLP) se muestra en el Anexo B. A continuación se plantea equivalentemente el mismo problema en dominio frecuencial o espectral.

2.3.2. Frequency Domain Normalized Delayed Linear Prediction (FD-NDLP)

Este algoritmo opera eficientemente sobre la transformada de Fourier de tiempo reducido (STFT) [28]. El planteamiento es esencialmente es el mismo al ya realizado en el tiempo, solo que en todas las señales aparecen índices n y f denotando el tiempo y frecuencia respectivamente. La señal reverberada $x^{(1)}_{n,f}$ y dereverberada $d_{n,f}$ se pueden escribir de la siguiente forma:

$$x^{(1)}_{n,f} = \bar{c}^{*T}_f \bar{x}_{n-D,f} + d_{n,f}, \tag{2.19}$$

$$d_{n,f} = \sum_{k=0}^{D-1} (h^{(1)}_{k,f})^{*T} s_{n-k,f}. \tag{2.20}$$

En la ecuación anterior, $\bar{c}_f = [(\bar{c}^{(1)}_f)^T, (\bar{c}^{(2)}_f)^T]^T$ y $\bar{c}^{(m)}_f = [(c^{(m)}_{1,f})^T, ..., (c^{(m)}_{L_c,f})^T]^T$, donde se asumen $L_m = 2$ micrófonos para facilitar la notación. Se puede generalizar para L_m arbitrario, basta agregar más columnas en la matrices anteriores.

Se plantea una verosimilitud sobre la señal deseada o dereverberada $d_{n,f}$:

$$L_f(\theta_f) = \sum_n \log(p(d_{n,f} = x^{(1)}_{n,f} - \bar{c}^{*T}_f \bar{x}_{n-D,f}; \theta_f)). \tag{2.21}$$

Notar que la verosimilitud está definida sobre todos los *frames* n, pero sobre una sola banda f. De esta manera, se puede ver que el aprendizaje esta vez se realiza sobre cada frecuencia f para obtener parámetros θ_f.

Se puede asumir un proceso guassiano sobre la señal *source*, resultando en un proceso gaussiano en $d_{n,f}$. Lo anterior permite escribir $p(d_{n,f})$ tal que $p(d_{n,f}) = \mathcal{N}(d_{n,f}; 0, \rho^2_{n,f})$, donde $\mathcal{N}(.)$ es un proceso gaussiano con media cero y varianza $\rho_{n,f} = \mathbb{E}[d_{n,f} d^*_{n,f}]$. Usando la distribución definida anteriormente, la verosimilitud se puede reescribir de la siguiente forma:

$$\begin{aligned} L_f(\theta_f) &= \sum_n \log(p(d_{n,f} = x^{(1)}_{n,f} - \bar{c}^{*T}_f \bar{x}_{n-D,f}; \theta_f)) \\ &= -\sum_n \frac{|x^{(1)}_{n,f} - \bar{c}^{*T}_f \bar{x}_{n-D,f}|^2}{\rho^2_{n,f}} - \sum_n \log(\rho^2_{n,f}) + \text{const.} \end{aligned} \tag{2.22}$$

Los parámetros a estimar son $\theta_f = [\bar{c}_f, \rho^2_f]$ con $\rho_f = [\rho^2_{1,f}, \rho^2_{2,f}, ...]$. De esta manera, el algoritmo en este caso se puede plantear de la siguiente forma:

Algorithm 1: Algoritmo para NDLP en dominio temporal-frecuencial o FD-NDLP.

Inicializar $\hat{\rho}^2_{n,f} = \max\{|x_{n,f}|^2, \epsilon_f\}$ con $\epsilon_f > 0$ un hiper-parámetro ajustable (en caso que $\hat{\rho}^2_{n,f}$ sea menor que un umbral permitido)

Repetir hasta convergencia:

- Actualizar $\hat{\bar{c}}_f^{(1)}$ como $\hat{\bar{c}}_f^{(1)} = \hat{\Psi}^{+}\hat{\phi}$, donde $\hat{\Psi} = \sum_n \frac{\bar{x}_{n-D,f}\bar{x}^*_{n-D,f}{}^T}{\hat{\rho}^2_{n,f}}$ y $\hat{\phi} = \sum_n \frac{\bar{x}_{n-D,f}x^{*(1)}_{n,f}}{\hat{\rho}^2_{n,f}}$
- Actualizar $\hat{d}_{n,f}$ como $\hat{d}_{n,f} = x^{(1)}_{n,f} - (\hat{\bar{c}}_f^{(1)})^{*T}\bar{x}_{n-D,f}$
- $\hat{\rho}^2_{n,f} = \max\{|\hat{d}_{n,f}|^2, \epsilon_f\}$

$+$ denota la pseudo-inversa de Moore-Penrose, $\hat{}$ hace referencia a que es una estimación y $*$ denota conjugado. Repitiendo este proceso en cada banda f, se obtiene una señal en dominio temporal-frecuencial (n, f), la cual está dereverberada. Este enfoque ha mostrado mejores resultados que la implementación en dominio temporal [28], además de ser más eficiente, ya que reduce radicalmente el número de observaciones al trabajar con la transformada de Fourier de tiempo reducido y no sobre la forma de onda directamente.

2.4. Métricas de calidad, inteligibilidad y reverberación

Dada la extensa investigación en el contexto de reverberación en audio surge la necesidad de medir cuán reverberada se encuentra una señal de audio. Con este fin existen una gran variedad de métricas para evaluar **inteligibilidad** y **calidad** de una señal de voz. Algunas métricas miden solo inteligibilidad, otras solo calidad y otras ambas características. Recordar que debe entenderse como inteligibilidad a la capacidad de comprender un audio linguisticamente. La calidad es un concepto más amplio y puede disminuirla tanto el ruido como la reverberación.

Es importante destacar que en el contexto de reconocimiento automático de voz (ASR) es habitual que se use como métrica de dereverberación al llamado *Word Error Rate (WER)*. Esta métrica corresponde a la tasa de error de palabras que entrega un modelo de transferencia de audio a texto. Sin embargo, WER sirve solo en este tipo de aplicación, no es una métrica que haya sido diseñada con el fin de medir la reverberación y puede verse perjudicado por otros aspectos que no sean la reverberación. Por estas razones, WER no es utilizado en este trabajo, ya que se busca dar un enfoque diferente y de validez más general. A continuación se presentan cada una de las métricas de importancia en este trabajo.

2.4.1. Perceptual Evaluation of Speech Quality (PESQ)

Este indicador data de 2001 [18] en el contexto de redes telefónicas. PESQ funciona como una medida de calidad de una señal de voz comparando una señal de referencia y una degradación de esta misma. La Figura 2.9 muestra un diagrama de bloques que ilustra el funcionamiento de PESQ.

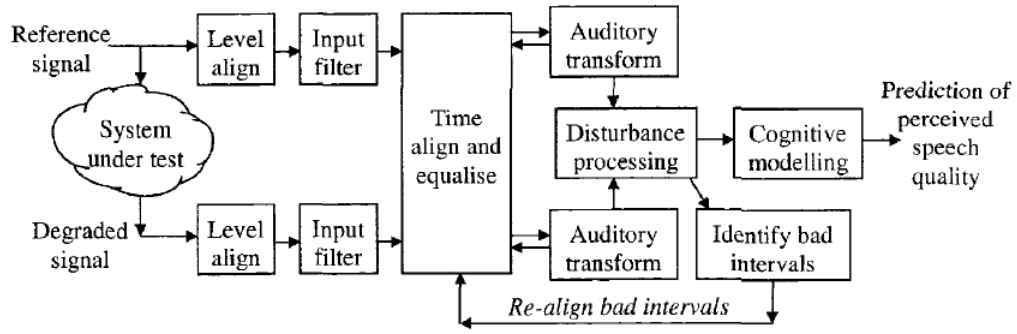


Figura 2.9: Diagrama de Bloques de PESQ (Perceptual Evaluation of Speech Quality) [18].

Como se puede apreciar es un sistema de procesamiento de señales de alta complejidad, donde este trabajo no pretende entrar en los detalles de cada bloque. Como se ha mencionado, se utiliza una señal de referencia y una versión degradada, entregando una **predicción cognitiva** de la calidad de la señal de audio. Internamente PESQ utiliza transformada de Fourier (STFT) con 42 *bins* de frecuencia para medir un "disturbio" o *disturbance frame* a *frame* de la señal de prueba respecto a la señal de referencia. Es un indicador de utilidad en señales de voz/*speech*. Es considerada una de las métricas objetivas para medir la calidad de una señal de voz ante perturbaciones del mundo real como lo puede ser un ambiente ruidoso y reverberativo [19, 20].

En términos matemáticos PESQ puede verse de la siguiente forma [19]:

$$\text{PESQ} = a_0 + a_1 D_{\text{ind}} + a_2 A_{\text{ind}}, \tag{2.23}$$

donde:

- a_0, a_1 y a_2 son parámetros a ajustar. Suelen ajustarse mediante regresiones o estrategias similares en datos que tengan relaciones con mejora de calidad en audios de voz [19].
- D_{ind} como una alteración o disturbio promedio entre las señales de referencia y la degradada o de prueba. Este promedio se escribe como $\frac{1}{N}\sum_{m=1}^{N}\text{disturbance[m]}^p$, para N *frames* o cuadros de tiempo. La variable disturbance[m] denota a cuánto se aleja el m-ésimo *frame* de la señal de prueba con respecto a la señal de referencia. Esto es parte del bloque *Disturbance Processing* en la figura 2.9.
- A_{ind} como una alteración o disturbio asimétrico promedio entre las señales de referencia y la degradada. Esto es parte del bloque *Disturbance Processing* en la figura 2.9.

PESQ varía entre -0.5 y 4.5 y en ocasiones es polémica de usar en el contexto de dereverberación [19]. Mientras mayor es su valor, mayor es la calidad de la señal que se está

evaluando. Es polémico utilizarla en ocasiones, debido a que fue diseñada en un contexto totalmente diferente que es el de redes telefónicas.

2.4.2. Log-likelihood Ratio (LLR)

Log-likelihood Ratio (LLR) corresponde a una medida de cercanía entre una señal *target* o limpia y una señal de prueba que se busca evaluar [19, 23]. Es considerada una medida de calidad de la señal de prueba. Matemáticamente se expresa de la siguiente forma:

$$\text{LLR} = \log(\frac{\vec{a_p}^T R_c \vec{a_p}}{\vec{a_c}^T R_c \vec{a_c}}), \tag{2.24}$$

donde:

- $\vec{a_c}$ es el vector *Linear Predictive Coding* (LPC) asociado a la señal *target*. Si $y(n)$ es la señal *target*, entonces las componentes $(a_c)_i$ del vector LPC están dados como coeficientes autoregresivos, lo cual matemáticamente está dado por $y(n) = \sum_{i=1}^{p} (a_c)_i y(n-i)$.
- $\vec{a_p}$ es el vector LPC asociado a la señal de prueba. Donde las componentes $(\vec{a_p})_i$ están dadas de la misma forma que para $\vec{a_c}$
- R_c es la matriz de autocorrelación de la señal *target*. Donde si y representa la señal *target*, entonces $R_c = \mathbb{E}(yy^H)$ donde H denota traspuesto conjugado.

Esta métrica es positiva y mientras menor es su valor, mayor es la calidad de la señal que se está evaluando.

2.4.3. Cepstral Distorsion (CD)

Cepstral Distorsion (CD) corresponde a una medida de distancia entre una señal de prueba y una señal *target* u objetivo con el fin de cuantificar la calidad de esta señal de prueba [21, 28]. Sean β_k los coeficientes cepstrales de la señal de prueba y $\hat{\beta}_k$ los de la señal *target*, entonces se define la métrica como:

$$\text{CD} = \frac{10}{\ln(10)} \sqrt{(\hat{\beta}_0 - \beta_0)^2 + 2 \sum_{k=1}^{12} (\hat{\beta}_k - \beta_k)^2}. \tag{2.25}$$

Expresada en la ecuación anterior, CD es medida en dB. Esta métrica es usada como medida de calidad de una señal, pero no mide necesariamente el nivel de reverberación presente en la señal de prueba.

Los coeficientes cepstrales son 13 valores y se calculan de la siguiente forma:

- Se calcula transformada de Fourier discreta por ventanas. De esta manera se tiene un número determinado de *frames*.
- El espectro del paso anterior es usado de entrada a un banco de filtros conocido como *Mel Scale FilterBank*. Estos corresponden a los filtros triangulares que fueron descritos en detalle en la sección de "Representaciones de Audio" de este capítulo. Este banco de filtros deja a su salida un espectro con 13 *bins*.

- Se calcula el *Log Power Spectrum*, que no es más que aplicar el logaritmo a la potencia espectral resultante del paso anterior.
- Se calcula la transformada Coseno discreta por cada *frame* del espectro anterior, como si dicho *frame* representara una señal temporal. La transformada Coseno una simplificación de la transformada de Fourier, la cual toma solo la parte real. Lo anterior significa que de la exponencial compleja en la transformada de Fourier, esta transformada utiliza solo el coseno.
- La señales resultantes por cada *frame* del paso anterior contienen 13 valores de amplitud, lo cuales son los llamados coeficientes cepstrales.

Notar que expresada matemáticamente CD está definida sobre un *frame*. Dado lo anterior, el valor final de CD se obtiene promediando sobre todos los *frames*.

Esta métrica es positiva y mientras menor es su valor, mayor es la calidad de la señal que se está evaluando.

2.4.4. Frequency-Weighted Segmental Signal to Noise Ratio (fwSNRseg)

La métrica fwSNRseg es considerada una medida de calidad e inteligibilidad de una señal de audio [6, 19, 22]. Para evaluar una señal de prueba con una señal *target* o "limpia" se utiliza la siguiente expresión:

$$\text{fwSNRseg} = \frac{10}{M} \sum_{m=0}^{M-1} \frac{\sum_{j=1}^{K} W(j,m) \log_{10}\left(\frac{|S(j,m)|^2}{|S(j,m)-\hat{S}(j,m)|^2}\right)}{\sum_{j=1}^{K} W(j,m)} \tag{2.26}$$

,

donde:

- $W(j,m)$ representa un peso puesto en la j-ésima banda
- K es el número de bins en las que es separado el espectro de las señales de prueba y *target*
- M es número total de frames (cuadros de tiempo)
- $S(j,m)$ corresponde a la magnitud espectral de la señal *target* o señal limpia
- $\hat{S}(j,m)$ corresponde a la magnitud espectral de la señal de prueba en la j-ésima banda en el m-ésimo frame

Los pesos $W(j,m)$ se escogen como $W(j,m) = S(j,m)^{\gamma}$, donde γ se ajusta para maximizar la correlación entre ambas señales de prueba y *target* en un rango $[0.2, 2]$.

Esta métrica es positiva y mientras mayor es su valor, mayor es la calidad de la señal que se está evaluando.

2.4.5. Speech to Reverberation Modulation Energy Ratio (SRMR)

Esta métrica es una de las más utilizadas debido a que no necesita comparar la señal a evaluar con otra señal "limpia", sino que es capaz de entregar una medida cuantitativa solamente con una señal de entrada. Planteada por Falk et al [20], esta métrica es **especializada en medir reverberación** y consta de las etapas descritas a continuación:

- Sea $\hat{x}(n)$ la señal que se desea evaluar, esta es filtrada a 23 canales utilizando un conjunto de filtros conocidos como *Gammatone filterbank*. Como resultado se tienen 23 señales $\hat{x_1}(n), \hat{x_2}(n), ..., \hat{x_{23}}(n)$. La idea de este tipo de procesamiento no se limita solo al uso de esta métrica SRMR, sino que tiene más tiempo en uso y fue planteado originalmente como un conjunto de filtros que actuen en paralelo como filtros pasa-banda, cada uno centrado en una frecuencia distinta y que emulen el funcionamiento del oído interno [31]. Un filtro de *Gammatone* en concreto tiene una respuesta al impulso dada por una distribución Gamma multiplicada por un tono puro en una frecuencia dada (sinusoide). Dicha respuesta al impulso se puede escribir en consecuencia como $h(t) = at^{n-1}\mathrm{e}^{-2\pi bt}\cos(2\pi ft + \phi)$ donde f es la frecuencia del tono puro o frecuencia central, b el ancho de banda del filtro, n el orden del filtro y a y ϕ son amplitud y fase respectivamente. Para este caso particular hay un *filterbank* compuesto de 23 filtros, por lo que hay 23 frecuencias centrales (1 para filtro). La frecuencia central más baja utilizada en esta etapa son 125 Hz y la más alta es cercana a la mitad de la frecuencia de muestreo de la señal de entrada (en el caso de este trabajo son 8000 Hz).

- Se captura una "envolvente" a las 23 señales temporales provenientes de los filtros de *Gammatone*. Por cada canal $\hat{x_j}(n)$ $j = 1, ..., 23$ de la etapa anterior, esta pasa por una etapa denominada *Temporal envelope* resultando en señales $\hat{e_j}(n) = \sqrt{\hat{x_j}(n)^2 + \mathcal{H}[\hat{x_j}(n)]^2}$ $j = 1, ..., 23$, donde $\mathcal{H}$ denota **transformada de Hilbert**. La transformada de Hilbert es definida como una convolución con una señal de la forma $h(t) = \frac{1}{\pi t}$. Las variaciones de la envolvente han mostrado empíricamente ser efectivas al utilizarse en métricas objetivas de calidad e inteligibilidad [20]. Al calcular STFT de cada una de estas envolventes, el contenido energético "reverberante" suele alojarse en ciertas zonas de frecuencia en específico [20].

- Por cada canal $\hat{e_j}(n)$ $j = 1, ..., 23$ de la etapa anterior se calcula transformada de fourier discreta por ventanas (STFT) resultando en $E_j(m; f)$ $j = 1, ..., 23$ con m denotando tiempo discreto y f es la frecuencia. Notar que esta frecuencia es una **frecuencia modulada** (resultante de la transformada de Fourier de las envolventes).

- Cada $E_j(m; f)$ $j = 1, ..., 23$ es dividida en 8 bandas para cada una de las 23 de la etapa anterior. La salida de esta última parte se denota como $\epsilon_{j,k}(m)$ $k = 1, ..., 8$ $j = 1, ..., 23$. Se denota $\bar{\epsilon}_{j,k}$ al promedio sobre todos los frames m en $\epsilon_{j,k}(m)$. Este promedio $\bar{\epsilon}_{j,k}$ es llamado **espectrograma modulado**.

- Se calcula el promedio sobre los 23 canales provenientes del filtro de *Gammatone* en el espectrograma modulado, lo cual se denota como $\bar{\epsilon}_k$ y se expresa matemáticamente de la siguiente forma:

$$\bar{\epsilon}_k = \frac{1}{23}\sum_{j=1}^{23}\bar{\epsilon}_{j,k}. \tag{2.27}$$

Finalmente el valor otorgado por la métrica se obtiene de la siguiente forma:

$$\text{SRMR} = \frac{\sum_{k=1}^{4} \bar{\epsilon}_k}{\sum_{k=5}^{K*} \bar{\epsilon}_k}. \tag{2.28}$$

.

Como se puede ver este ratio expresa cuán superior es la energía acumulada en las primeras 4 bandas respecto a las restantes $K*$ bandas. El parámetro $K*$ se ajusta a la señal bajo prueba [20], pero usualmente es fijado en $K* = 8$. Se desprende que el trasfondo de esta métrica es lograr que la parte de energía de una señal asociada a reverberación o perturbaciones similares quede alojada en una o varias bandas en particular. Mientras mayor es el valor de SRMR, menor es su "nivel" de reverberación.

La Figura 2.10 muestra un resumen esquemático de todo el procedimiento realizado en la métrica SRMR.

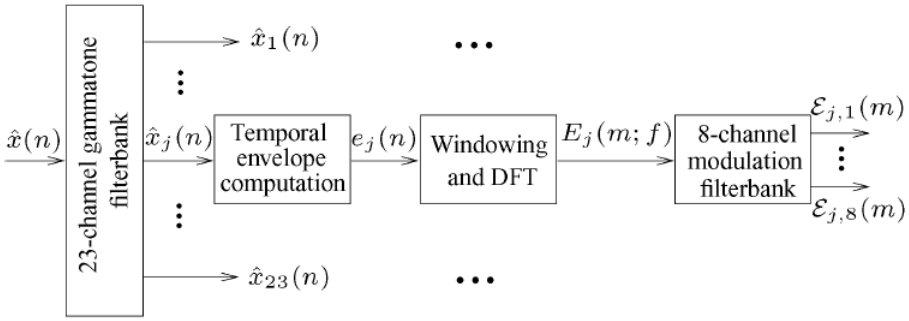


Figura 2.10: Diagrama de Bloques de SRMR [20].

Se muestra desde el paso por el *gammatone filterbank* hasta la separación en 8 bandas.

Esta métrica es positiva y mientras mayor es su valor, menor es el nivel de reverberación en la señal que se está evaluando.

2.5. Machine y Deep Learning

2.5.1. Perceptrón Multicapa (MLP)

El perceptrón multicapa es una red neuronal artificial estructurada en capas interconectadas. En la Figura 2.11 se observa la estructura más sencilla, con una capa de entrada de 3 unidades, una capa intermedia que se denomina capa oculta de 4 unidades y una capa de salida de 2 unidades. Las capas ocultas se denominan de esta manera, ya que no tienen contacto ni con la entrada ni con la salida de manera directa. Cada unidad de cada capa está conectada con cada unidad de la capa siguiente (flechas en la figura), lo cual constituye lo que se denomina capas **Fully Connected**.

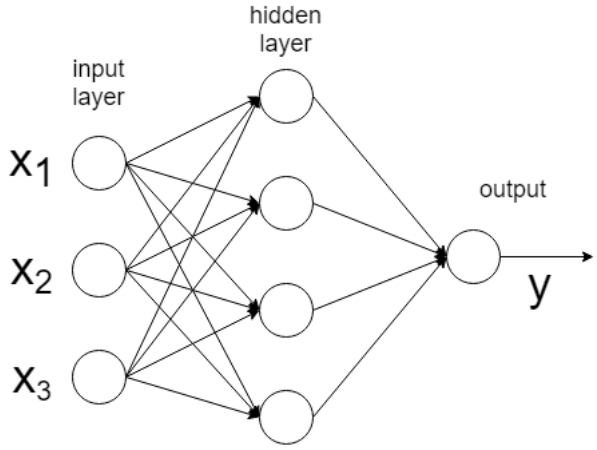


Figura 2.11: Esquema del MLP.

Entre aspectos relevantes de este tipo de red neuronal, cabe destacar los siguientes:

- Cada conexión (flecha en la figura) entre una unidad de una capa y otra unidad de la siguiente capa tiene un peso aprendible asociado. Cada neurona tiene además un bias aprendible
- Cada capa tiene una función de activación. Esto quiere decir que todas las unidades de una misma capa tienen la misma función de activación.
- La capa de entrada opera con un fin de recibir los valores de la entrada y conectarlos hacia la siguiente capa. Esta capa no posee función de activación.
- Cada unidad de cada tiene una salida cuya expresión matemática es la definida anteriormente para el perceptrón.

Para la arquitectura de ejemplo en la Figura 2.11, esta posee $N_h = 4$ neuronas en capa oculta y $N_{in} = 3$ neuronas en la capa de entrada. Sea w_{ij} el peso que conecta la i-ésima neurona de la capa de entrada con la j-ésima neurona de la capa oculta y sea b_j el bias asociado a la j-ésima neurona de la misma, entonces la salida h_j en esa unidad se puede construir usando la definición del perceptrón de la manera siguiente:

$$h_j = \phi_h(\sum_{i=1}^{N_{in}=3} w_{ij}x_i + b_j) \tag{2.29}$$

donde x_i representa la i-ésima entrada, la cual está asociada a la i-ésima salida de la capa de entrada y ϕ_h la función de activación de la capa oculta. Con la salida definida de cada unidad de la capa oculta, se puede definir finalmente la salida y_k en cada unidad de la capa de salida de la siguiente forma:

$$y_k = \phi_{out}(\sum_{i=1}^{N_h=4} w_{ik}h_i + \bar{b_k}), \tag{2.30}$$

donde w_{ik} conecta la i-ésima unidad de la capa oculta con la k-ésima de salida y $\bar{b_k}$ el bias asociado. Una simplificación muy usada es unir los pesos w y bias b en un solo vector, de modo que la última ecuación se puede rescribir como sigue:

$$y_k = \phi_{out}(w^T h + \bar{b_k}) = \phi_{out}(\theta^T H_k), \tag{2.31}$$

donde $w = (w_{1k}, w_{2k}, ..., w_{N_h k})$, $h = (h_1, h_2, ..., h_{N_h})$, $\theta = (w, \bar{b_k})$ y $H_k = (h, 1)$

La ventaja de escribir una salida de esta manera, es que se puede ver solamente θ como un **único parámetro aprendible** y no los pesos y bias por separado.

2.5.2. Redes Neuronales Convolucionales (CNN)

2.5.2.1. Convolución sobre una imagen

La convolución sobre una imagen I se realiza utilizando un filtro w de menor tamaño que la imagen. La operación matemática que se realiza se define de la siguiente forma:

$$S(i,j) = (I * w)(i,j) = \sum_m \sum_n I(i+m, j+n)w(m,n). \tag{2.32}$$

Cada elemento de la imagen de salida se calcula de esta manera con una convolución. El filtro o también llamado kernel se desplaza por la imagen para calcular la convolución sobre todas las posiciones posibles. La Figura 2.12 [10] muestra cómo se utiliza un kernel 3x3 sobre la imagen de entrada de 8x8 para generar un píxel en la imagen de salida.

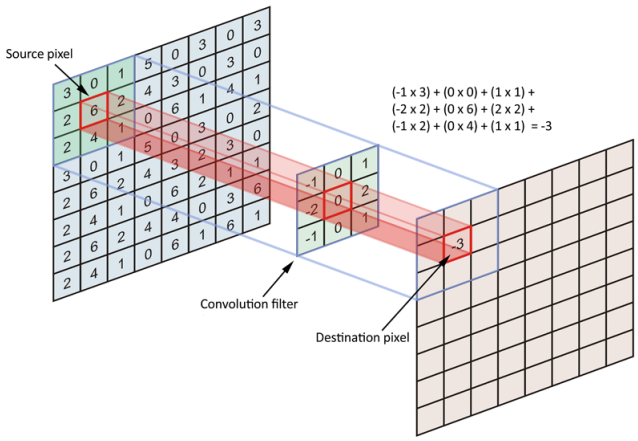


Figura 2.12: Ilustración de convolución sobre una imagen.

[10] Extraída de https://datascience.stackexchange.com/questions/23183/why-convolutions-always-use-odd-numbers-as-filter-size

El kernel se debe desplazar sobre todas las posiciones posibles de la imagen, pero existe un parámetro ajustable llamado Stride que sirve para que el kernel se pueda ir moviendo saltándose algunas posiciones sobre la imagen, este parámetro se detallará más adelante.

El objetivo de utilizar un filtro w sobre la imagen es capturar información relevante en términos visuales como pueden ser texturas, sombras, bordes, entre otros. Los valores del filtro se aprenden de la misma manera explicada con anterioridad para un MLP con gradiente descendiente u otro algoritmo similar.

2.5.2.2. Conceptos básicos

1. Stride: corresponde al paso al cual el filtro o kernel se desplaza dentro de la imagen para calcular la convolución sobre una imagen.

2. Capas de Pooling: una capa de Pooling divide la imagen de entrada (ancho y largo) en ventanas. Por cada ventana se realiza una operación, donde las más utilizadas son tomar el máximo o el promedio de la ventana, los cuales se denominan **Max Pooling** y **Average Pooling** respectivamente. El tamaño de la ventana debe ser menor que la imagen original. En la Figura 2.13 [11] se muestra un ejemplo de una imagen de entrada 4x4 y un tamaño de ventana 2x2, donde se realiza una operación de Max Pooling.

3. Zero-Padding: la estrategia de Zero-Padding consiste en añadir píxeles extra a la imagen antes de aplicar una convolución. Como consecuencia se expande el área de la imagen. El propósito de utilizar esta estrategia es que cuando la ventana recorre la imagen de entrada calculando la convolución visite una mayor cantidad de veces los bordes.

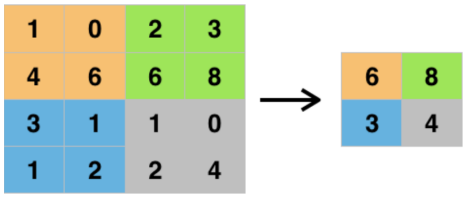


Figura 2.13: Operador de Max Pooling en 2D.

2.5.2.3. Capa Convolucional

Una capa convolucional sobre una imagen corresponde a sucesivas convoluciones utilizando diferentes filtros. Una capa convolucional está definida por los siguientes aspectos:

- Dada la profundidad de la imagen de entrada N_{in} y salida N_{out}. Notar por ejemplo que la profundidad de una imagen de colores es 3 (1 para color rojo, verde y azul, denotado RGB) y la de una imagen en escala de grises es 1.

- Definir tamaño (ancho y largo) del filtro o kernel. Este tamaño se denotará (kw, kh) para el ancho y largo del filtro. Se utilizan N_{out} filtros de este tamaño para calcular convoluciones sobre la imagen.

[11] Extraída de https://deepai.org/machine-learning-glossary-and-terms/max-pooling

- Definir Stride. Se debe definir el paso al cual cada filtro se desplaza por la imagen
- Definir si habrá o no Zero-Padding. En caso afirmativo se debe definir el ancho del margen de ceros que se agrega a la imagen.

Los parámetros aprendibles en una capa convolucional son los valores de cada filtro utilizado. El total de valores de un filtro es kw x kh. Hay un total de N_{out} filtros a utilizar sobre cada una de los N_{in} canales de entrada o profundidad de entrada. De esta manera, la cantidad de parámetros aprendibles N_θ se calcula según la siguiente ecuación:

$$N_\theta = (\text{kw x kh x } N_{in} + 1) \text{ x } N_{out}. \tag{2.33}$$

2.5.3. Autoencoder

Un autoencoder es una red neuronal que es entrenada para copiar la entrada en la salida [17]. Está formado por 2 partes: un encoder y un decoder. El encoder mapea la entrada X a un **código**, la cual es una **representación** en una capa oculta $h = f(X)$ de la red. El decoder busca reconstruir la entrada a partir del código h, entregando una salida $X' = g(h)$. El origen de este tipo de red neuronal surge en la reducción de dimensionalidad y búsqueda de características, lo cual es plasmado en el código h. La Figura 2.14 [12] muestra un esquema de la arquitectura de esta red.

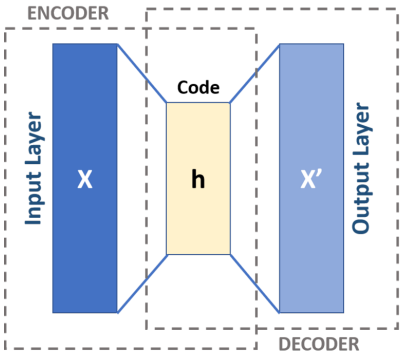


Figura 2.14: Esquema de Autoencoder.

El encoder y decoder pueden ser lineales, es decir, estar conformados solo por capas Fully Connected. En tal caso, el código es un análisis de componentes principales (PCA) de la entrada X. El autoencoder de interés en este trabajo es aquel en el cual el encoder y decoder son redes neuronales convolucionales.

Como se busca reconstruir la entrada, la función de pérdida a utilizar $L(X, X') = L(X, g(h(X)))$ es el error cuadrático medio entre X y X'. Sin embargo, esto es solo en el caso de un autoencoder determinístico, ya que se puede considerar un autoencoder variacional en el cual el encoder y decoder están caracterizados por distribuciones $p_{\text{encoder}}(h|X)$ y $p_{\text{decoder}}(X'|h)$.

Basado en la idea de autoencoder, se puede construir una red que contribuya en el problema de dereverberación, pero varía en el sentido que no se busca reconstruir la entrada. En su

[12] Extraída de https://en.wikipedia.org/wiki/Autoencoder

lugar, se busca que la entrada (espectrograma reverberado) sea mapeado a un espectrograma dereverberado. Para lograr este objetivo, se utiliza una red con una estructura y aprendizaje similar a un autoencoder determinístico [8], pero en vez de calcular el error cuadrático medio entre la entrada X y salida X', lo hacen entre la salida X' y un espectrograma target o etiqueta Y. Este espectrograma de etiqueta está limpio de reverberación, de modo que al entrenar la red, esta realiza el mapeo de un espectrograma reverberado a uno dereverberado.

2.5.4. Redes Neuronales Recurrentes (RNN)

Las redes neuronales recurrentes son una familia de redes para procesamiento secuencial de datos. Así como una red convolucional es especializada en una grilla de valores (imagen), una red neuronal recurrente está especializada en una secuencia de valores $x^{(1)}, ..., x^{(\tau)}$ [17]. Las redes recurrentes incorporan **bucles de realimentación** que permiten usar salidas anteriores como entradas. Lo anterior permite un tratamiento de datos más eficiente sobre secuencias, sin importar el largo de estas. La Figura 2.15 [13] muestra como una lazo de realimentación es incorporado en una red neuronal habitual (MLP).

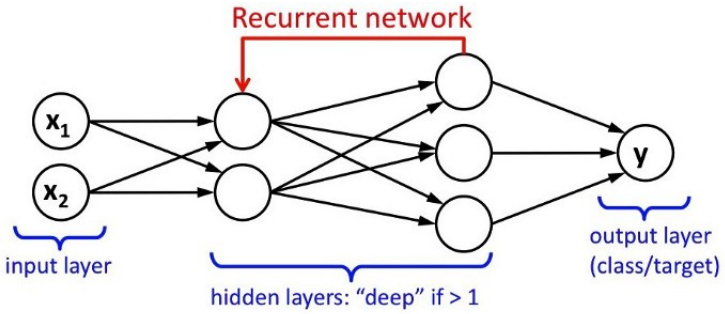


Figura 2.15: Esquema de una RNN.

Como se puede ver, hay 2 capas ocultas y una unidad de la segunda capa oculta se conecta con la primera mediante el lazo en rojo en la Figura 2.15. Este corresponde al esquema más básico de una red recurrente. Bajo este funcionamiento las redes recurrentes se plantean con una ecuación de la forma siguiente:

$$h^{(t)} = f(h^{(t-1)}, x^{(t)}; \theta). \tag{2.34}$$

La variable $h^{(t)}$ representa un estado, el cual en el esquema anterior corresponde a las unidades ocultas de la red. Esta formulación permite ver que una red recurrente define un sistema **con memoria**, es decir, para predecir el estado en t, no solo se toma la entrada en t $x^{(t)}$, sino que también se consulta el estado en $t-1$ denotado como $h^{(t-1)}$. Adicionalmente, f se utiliza para denotar un funcional que resume la relación entre el estado en t, del estado en $t-1$ y de la entrada en t. Los parámetros aprendibles del modelo se denotan como θ.

Las redes recurrentes de interés en este trabajo son las llamadas *Gated RNNs*, las cuales están basadas en la idea de generar diferentes vías en el tiempo, que eviten que al calcular

[13] Extraída de https://medium.com/@curiousily/making-a-predictive-keyboard-using-recurrent-neural-networks-tensorflow-for-hackers-part-v-3f238d824218

el gradiente se anule o que diverja [17]. Lo anterior puede ser un problema común en ciertos contextos de redes neuronales, puesto que estas aprenden por *backpropagation* que necesitan del cálculo de un gradiente. Las "vías" en el tiempo que introducen las *Gated RNNs* introducen nuevas derivadas, lo cual permite ver intuitivamente que aleje al gradiente de anularse. El problema de que el gradiente se anule es que llegado ese punto el modelo no es capaz de continuar el aprendizaje, quedando estancado erróneamente. Unas de las *Gated RNNs* más conocidas es la llamada *Long Short-Term Memory* (LSTM), la cual se muestra en bloques en la figura 2.16 [14].

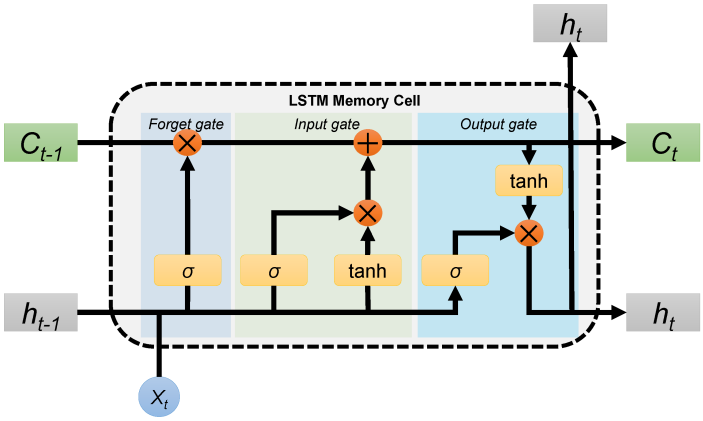


Figura 2.16: Esquema de LSTM.

Existen 3 partes principales: *Forget gate*, *Input Gate* y *Output gate* conectadas mediante otras variables. Las cuales se detallan a continuación [15]:

- *Gates*: existen 4 de estas variables, las cuales son *Input gate* i_t, *Forget gate* f_t, *Cell gate* g_t y *Output gate* o_t. Matemáticamente se definen como sigue:

$$i_t = \sigma(W_{ii}x_t + b_{ii} + W_{hi}h_{t-1} + b_{hi}) \tag{2.35}$$

$$f_t = \sigma(W_{if}x_t + b_{if} + W_{hf}h_{t-1} + b_{hf}) \tag{2.36}$$

$$g_t = \tanh(W_{ig}x_t + b_{ig} + W_{hg}h_{t-1}) \tag{2.37}$$

$$o_t = \sigma(W_{io}x_t + b_{io} + W_{ho}h_{t-1} + b_{ho}) \tag{2.38}$$

  Se puede ver que las 4 ecuaciones comparten la misma "forma", donde x_t es la entrada a tiempo t, h_{t-1} es el estado oculto a tiempo $t-1$. Las funciones de activación σ (sigmoide) y tanh confinan las salidas a [0,1] y [-1,1] respectivamente. Adicionalmente W y b representan pesos y bias (aprendibles) respectivamente, con subíndices que denotan las variables que conectan. Por ejemplo b_{hi} es el bias que conecta el estado oculto con la *input gate*

- *States*: Lo conforman el estado oculto h_t ya introducido con anterioridad y un estado adicional *Cell state* c_t, los cuales se actualizan a tiempo t matemáticamente según las siguiente ecuaciones:

[14] Tomada de https://www.mdpi.com/2073-4441/12/1/175/htm
[15] Notación tomada de la documentación de Pytorch. Disponible en https://pytorch.org/docs/stable/generated/torch.nn.LSTM.html

$$c_t = f_t \otimes c_{t-1} + i_t \otimes g_t, \tag{2.39}$$

$$h_t = o_t \otimes \tanh(c_t). \tag{2.40}$$

El símbolo $\otimes$ denota el producto matricial punto a punto, también conocido como producto de Hadamard. Se puede ver que h_t depende explícitamente de c_t a tiempo t sin dependencias explícitas en $t-1$, donde c_t la única dependencia que tiene de $t-1$ es actualizada ponderada por f_t, lo cual permite inferir el por qué del nombre de f_t (olvido o en inglés *forget*).

2.5.5. Redes Generativas Adversarias (GAN)

Las redes Generativas Adversarias (en adelante GAN) consisten en 2 redes neuronales que conforman un **juego minimax**. Fueron planteadas por primera vez en 2014 [24] y están compuestas por una red generativa G y una red discriminadora D. La red generativa G busca capturar la distribución de los datos y la red discriminadora D busca clasificar un dato para saber si proviene del conjunto de entrenamiento o de una muestra creada por G. La red generadora buscará "engañar" a la red discriminadora, de tal forma que D sea incapaz de distinguir entre una muestra proveniente de G y una del conjunto de datos de entrenamiento. Poniendo como ejemplo un dataset de dígitos del 0 al 9 (dataset MNIST) la Figura 2.17 muestra a nivel de bloques cómo funcionan las redes discriminadora y generadora [16]

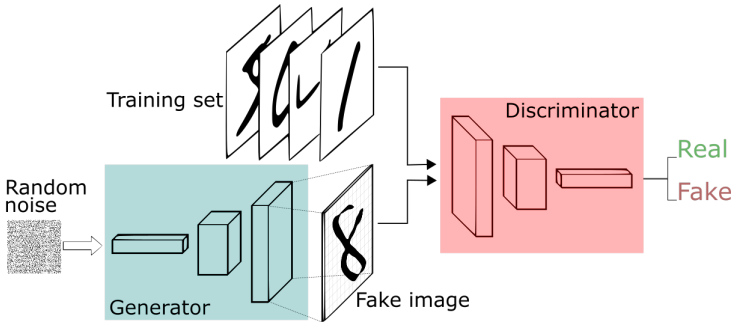


Figura 2.17: Esquema de GAN.

La red generadora recibe ruido aleatorio de entrada y en su salida entrega una muestra generada (un 8 en la figura). La red discriminadora recibe tanto muestras creadas por G y los datos de números del set de entrenamiento. La red discriminadora entrega su predicción acerca de si el dato es real o falso, donde debe decidir que un dato es falso si proviene de la red generadora. El entrenamiento de estas redes idealmente finaliza cuando la red discriminadora recibe una muestra creada por la red generadora y no es capaz de identificar si es real o falso, i.e con probabilidad de 50 % el dato es falso y con probabilidad de 50 % el dato es real.

El planteamiento matemático que permite expresar la idea de las GANs se basa en su función de pérdida L. La red generadora G distingue entre muestras reales $x \sim p(x)$ y

[16] Extraída de https://playerone-studio.com/gan-2d-tiles-generative-adversarial-network, autor Thalles Silva.

muestras de la red generadora $G(z)$ con $z \sim p(z)$ ruido aleatorio. La red generadora es entrenada para reproducir muestras indistinguibles de las muestras reales. La función de pérdida se plantea de la siguiente forma:

$$\min_G \max_D \quad L_{\text{GAN}} = \mathbb{E}_{p(x)}[\log(D(x))] \quad + \mathbb{E}_{p(z)}[\log(1 - D(G(z)))]. \tag{2.41}$$

Como se puede ver la optimización es minmax, razón por la cual es común hablar de GANs como un "juego minimax". La red discriminadora busca maximizar la probabilidad de que una muestra sea clasificada correctamente, mientras que la red generadora busca minimizar la probabilidad de que la red discriminadora clasifique una de sus muestras como falsa. En su enfoque original, no hay una restricción hacia el tipo de red que sean tanto D como G, pero en el contexto de imágenes es frecuente elegir ambas como redes convolucionales. Las aplicaciones de este tipo de arquitectura es muy amplio, pudiendo ser útil para generar realistas muestras de rostros u otro tipo de imagen, contribuir a métodos de imputación de datos y en particular ser de utilidad en el contexto de dereverberación. Este último aspecto será tratado en detalle en secciones posteriores.

2.6. Aplicaciones de Machine y Deep Learning en el problema de dereverberación

2.6.1. MLP y LSTM para dereverberación

La primera estrategia a considerar para resolver el problema es por medio de un modelo *Multilayer Perceptron* (MLP) [6, 7, 30]. El modelo opera a nivel de espectrogramas, pero esto **NO consiste en dar de entrada el espectrograma completo a la red neuronal**. La forma de resolver el problema es notando el efecto de la reverberación en el espectrograma descrito en el marco teórico. Como se ha descrito anteriormente, el efecto de la reverberación en el espectrograma es el de expander o estirar los cuadros o frames temporales. Con esta inspiración, la red neuronal recibe una cantidad 2p+1 de **cuadros temporales del espectrograma** y los mapea a 1 solo cuadro, lo cual se denomina ***Context Information***. En la Figura 2.18 se muestra un esquema de este tipo de red neuronal. En adelante a esta arquitectura se le denotará **Context-MLP**.

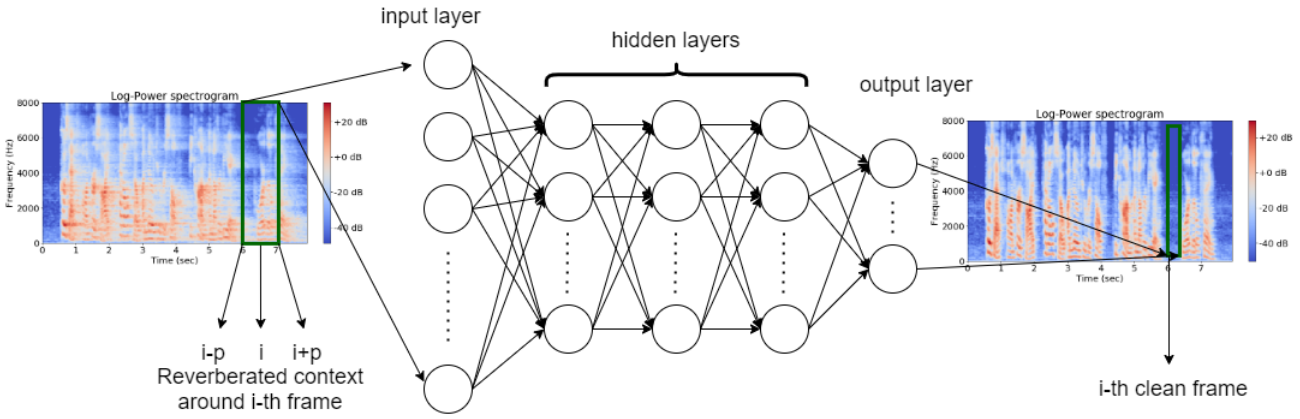
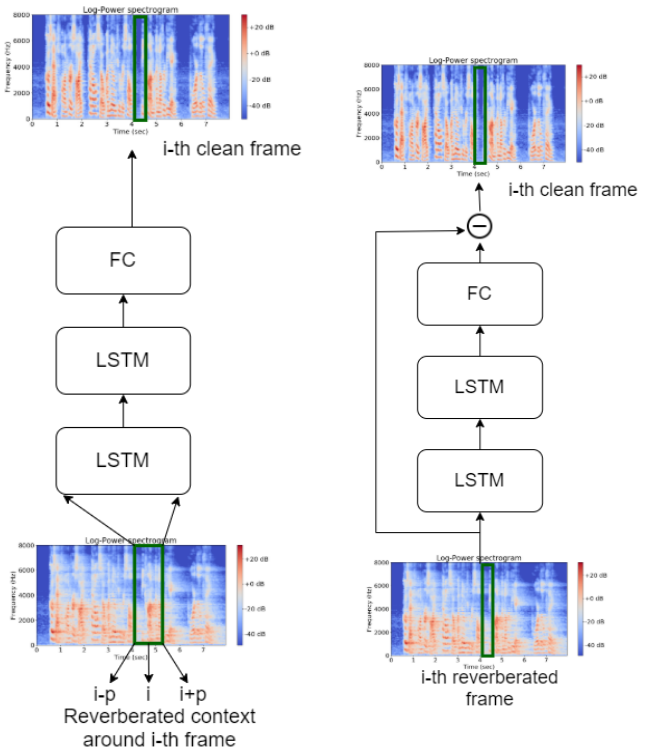


Figura 2.18: Esquema Context-MLP. Elaboración propia.

Como se puede observar se recibe un espectrograma reverberado y se toman una cantidad 2p+1 de *frames*. Específicamente, se toman p *frames* hacia la izquierda y p hacia la derecha del i-ésimo *frame*. Estos cuadros forman un "contexto" en torno al i-ésimo *frame* y están encerrados en un rectángulo verde en la Figura 2.18. Notar que este cuadro verde cubre todo el eje de frecuencia (eje vertical), pero solo una fracción de tiempo. En la salida de la red neuronal se obtiene solo 1 cuadro temporal de los 2p+1 que se tomaron de entrada. Repitiendo este procedimiento por cuadros, se puede construir un espectrograma dereverberado en la salida. Para poder entrenar la red, es necesario tener pares de espectrogramas, donde 1 de ellos sea audio limpio y el otro reverberado (generado a partir del audio limpio), tal como se tiene en la base datos construida en este trabajo. Esta red neuronal puede ser entrenada de forma supervisada, ya que cada cuadro que se obtiene en la salida se puede comparar con el espectrograma limpio. Lo anterior equivale a comparar un cuadro del espectrograma limpio de reverberación que sirve de etiqueta con el cuadro de un espectrograma dereverberado construido por la red. Como función de pérdida es usual el **error cuadrático medio** entre un cuadro del espectrograma target y el cuadro de la salida de la red. No obstante lo anterior, han habido esfuerzos por mejorar la función de pérdida con ligeras variaciones al error cuadrático medio [7], pero los resultados no mejoran apreciablemente.

Bajo el enfoque mencionado anteriormente con MLP, también ha tenido relevancia un arquitectura similar, pero que utilice bloques LSTM, en vez de solo capas *Fully Connected* [30]. En adelante a este enfoque se le denotará como **Context-LSTM**. Lo anterior corresponde a formar una red con un *Context Information* en su entrada, pero esta no es la única forma como se utilizan los bloques LSTM. Una segunda forma corresponde a utilizar *frame* a *frame* del espectrograma reverberado como entrada, y restar el *frame* de salida de la red al de entrada. La Figura 2.19 muestra una comparación entre las 2 formas de utilizar bloques LSTM para dereverberación.



(a) Context-LSTM.

(b) *Late Reverberation Supression* LSTM (LS-LSTM).

Figura 2.19: Esquema de dereverberación usando LSTM. Elaboración propia.

A la izquierda se muestra la arquitectura utilizando *Context Information* o Context-LSTM. Como se puede ver, en la Figura 2.19, esta es exactamente la misma idea utilizada con MLP, es decir, aprender un mapeo de un contexto de 2p+1 *frames* a 1 solo. La arquitectura de la derecha fue planteada en Yan Zhao et al [2] y busca sustraer de cada *frame* las reflexiones tardías (ver concepto en sección 2.1.2 dedicada a explicar los fundamentos de RIRs). La red neuronal en este último caso, en vez de aprender un mapeo reverberado-dereverberado, su acción es "estimar la reverberación" (causada por reflexiones tardías) y sustraerla del *frame* de entrada. A esta última arquitectura en adelante se le denotará LS-LSTM (LS por *Late Supression*).

2.6.2. El Autoencoder para dereverberación

El Autoencoder en el contexto de audio es utilizado en una gran variedad de problemas. Se utiliza con frecuencia en sintesis de audio [12], donde se busca manipular un **espacio latente** para crear notas musicales o sonidos. El espectrograma es tratado como una imagen, sobre la cual puede operar una red neuronal convolucional. En el problema de dereverberación, el autoencoder se ha utilizado frecuentemente con espectrogramas [8]. Basado en la idea del autoencoder, se puede construir una idea similar, pero que en vez de reconstruir la entrada, se realice un mapeo a un espectrograma dereverberado. La Figura 2.20 muestra cómo opera este tipo de red neuronal en el espectrograma.

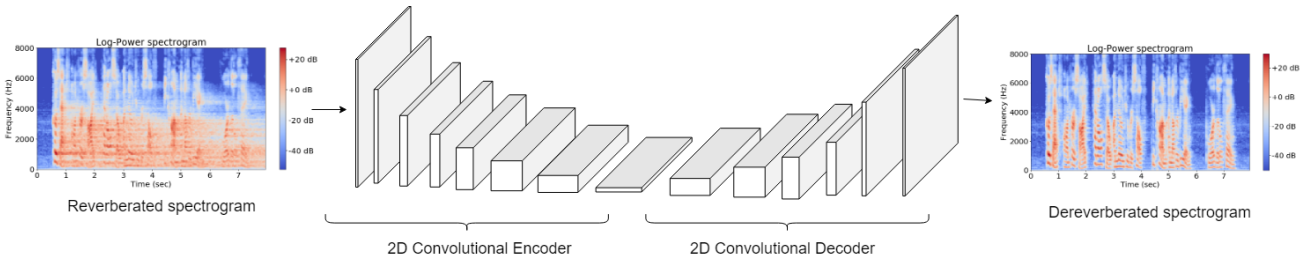


Figura 2.20: Esquema de Autoencoder sobre espectrogramas. Elaboración propia.

Como se puede ver, la red neuronal utiliza de entrada el espectrograma reverberado completo y lo comprime (utilizando un encoder convolucional) a un espacio tal que la imagen tenga un menor ancho y alto, pero una mayor profundidad. Posteriormente un decoder convolucional es ocupado para devolver la imagen al tamaño original. Utilizando la base de datos construida para este trabajo, se ocupa un espectrograma limpio (target) para compararlo con el espectrograma en la salida de la red. La comparación mencionada se realiza por medio de error cuadrático medio.

2.6.3. GAN para dereverberación

Una arquitectura de tipo GAN puede ser utilizada en este contexto de dereverberación. La idea fundamental es que el generador G sea capaz de aprender a crear una muestra dereverberada a partir de un espectrograma reverberado en su entrada. Este enfoque es mostrado en la Figura 2.21.

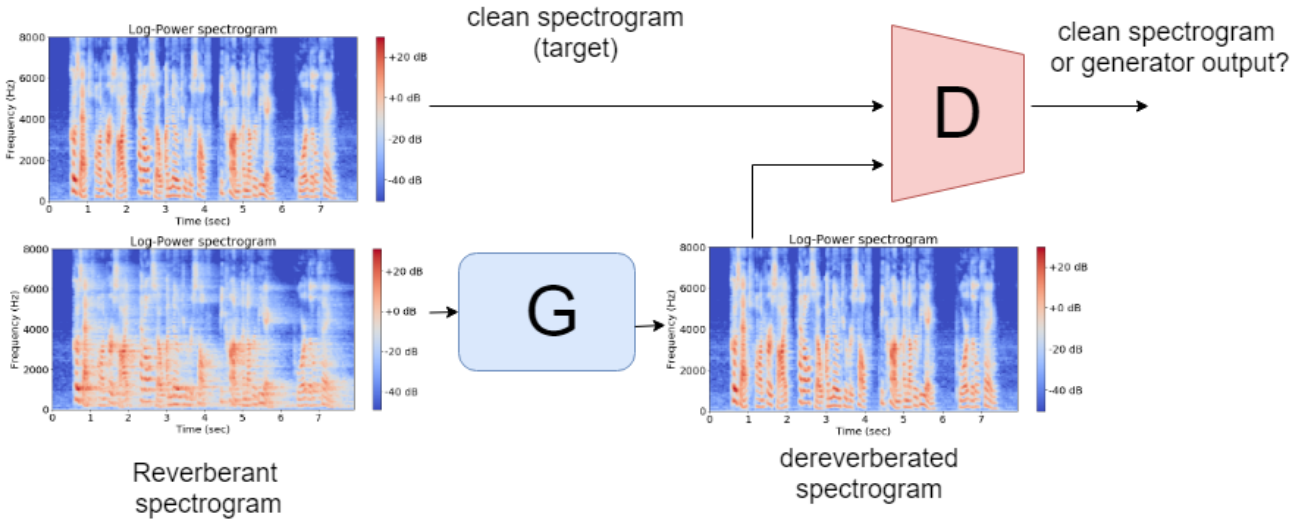


Figura 2.21: Esquema GAN sobre espectrogramas. Elaboración propia.

Basado en la idea de la estructura de un autoencoder con aprendizaje GAN [10, 8], la red generadora G recibe un espectrograma reverberado e intenta "engañar" a la red discriminadora D haciendo que esta sea incapaz de distinguirla de los espectrogramas limpios (en términos de reverberación) en el conjunto de entrenamiento. Al finalizar el entrenamiento,

la red generadora G es un modelo con la capacidad de dereverberar un espectrograma. La función de pérdida L se modela de la siguiente forma:

$$L(G, D) = L_{\text{GAN}}(G, D) + \lambda L_{\text{MSE}}(G) \tag{2.42}$$

donde L_{MSE} es el error cuadrático medio mencionado para el autoencoder dereverberativo y L_{GAN} la función de pérdida de GANs descrita en el marco teórico. Expandiendo este último término L queda expresado equivalentemente como sigue:

$$\min_G \max_D \quad L = \mathbb{E}_{p(x)}[\log(D(x))] \quad + \mathbb{E}_{p(z)}[\log(1 - D(G(z)))] + \lambda L_{\text{MSE}}(G) \tag{2.43}$$

λ es un hiperparámetro ajustable que puede ser elegido de forma óptima. El propósito de usar este tipo de arquitectura es que explora un tipo de aprendizaje que no utiliza solamente el MSE como función de pérdida.

2.7. Estado del arte de métodos dereverberativos

A modo de resumen de los métodos dereverberativos que se han presentado con anterioridad, existen 3 principales enfoques de dereverberación. Estos enfoques son los siguientes:

1. *Context Information* o *Context Window*: corresponde a analizar una ventana de 2p+1 *frames* (p es ajustable, pero muy comúnmente utilizado con p=5 o p=7) del espectrograma reverberado y aprender un mapeo a 1 solo *frame* dereverberado. Este método se ilustró con un MLP y con una red con bloques recurrentes LSTM.

2. Supresión de Reflexiones Tardías (*Late Reflections Supression*): corresponde a sustraer la reverberación del espectrograma reverberado a partir restar de este las reflexiones tardías [17]. El método no supervisado WPE (y por ende FD-NDLP) corresponde a este enfoque. Adicionalmente se mostró una arquitectura basada en LSTM (denotada LS-LSTM) con este enfoque, la cual opera a nivel de *frames* de espectrogramas, pero *frame* a *frame* y no utilizando ventanas de ellos (a diferencia de *Context Information*).

3. *Image to Image*: Este enfoque se ilustró con anterioridad mediante autoencoders. Corresponde a aprender una vía de dereverberación con el espectrograma completo (como si fuera una imagen) como entrada y no dividirlo por *frames*.

Los 3 enfoques han tenido éxito en el contexto de dereverberación en datasets masivos muy populares como pueden ser *CHiME Challenge* [18] y *Reverb Challenge* [19] (no utilizados en este trabajo) . Sin embargo, de forma escasa se ha puesto de manifiesto cuáles son las ventajas y desventajas de cada uno, cuál es su nivel de robustez frente al ruido y capacidad de generalizar en diferentes contextos.

[17] Las reflexiones tardías fueron explicadas en la sección de respuestas al impulso
[18] Se puede leer sobre su contenido en http://spandh.dcs.shef.ac.uk/chime_challenge/CHiME4/
[19] Se puede leer sobre su contenido en https://reverb2014.dereverberation.com

Capítulo 3

Metodología

En esta sección se explica en detalle qué modelos se escogen y cómo se modifican, qué datos son los utilizados para entrenar los diferentes modelos y cómo se evalúa. Se explica además el procesamiento necesario a los datos para poder utilizarlos. El lenguaje de programación utilizado en este trabajo es Python y el *framework* Pytorch es utilizado para todos los modelos neuronales.

3.1. Dataset de audios de habla inglesa

Para resolver el problema de dereverberación por medio de aprendizaje supervisado se deben poseer 2 representaciones (ya sea espectral o temporal) de un mismo audio, donde una de ellas esté limpia de reverberación y el otro reverberado. Para ello, primero se busca una base de datos de audios de voz limpios de reverberación. Con este propósito se utiliza *LibriSpeech* [1]. El dataset se compone de cerca de 1000 horas totales de audios en formato .flac de habla inglesa, muestreados a 16 kHz. Este dataset está estructurado como se muestra en Tabla 3.1.

Tabla 3.1: Dataset LibriSpeech.

División	**Ejemplos**
dev_clean	2,703
dev_other	2,864
test_clean	2,620
test_other	2,939
train_clean100	28,539
train_clean360	104,014
train_clean500	148,688

Como se puede ver, está dividido en *development* (dev), *train* y *test*. Todas estas divisiones tienen datos de audio limpios (prefijo *clean*), lo cuales son de utilidad en el presente trabajo. Con el prefijo *clean* se se refiere a audios con reverberación y ruido prácticamente nulos, por

[1] Disponible en http://www.openslr.org/12 y también descrito en TensorFlow https://www.tensorflow.org/datasets/catalog/librispeech

lo que el WER [2] que entregan sistemas de transferencias de audio a texto es muy bajo [32] (en torno al 5 % o menor).

El dataset *LibriSpeech* fue escogido debido a su cantidad de ejemplos de audio y a que no es de los datasets masivos que se utilizan con frecuencia en el contexto de reverberación[3]. De esta manera, los datos simulados (reverberados mediante convolución) son generados por autoría propia, pudiendo elegir que RIRs utilizar, cuánto ruido introducir, entre otros aspectos.

3.2. Datos de respuestas al impulso (RIRs)

Con un dataset de audios de hablantes de habla inglesa, se plantea la necesidad de generar audios reverberados. Para esto se utiliza un segundo dataset, abreviado con frecuencia como OMNI y es de respuestas al impulso [34] [4]. Este dataset tiene respuestas al impulsos de 3 habitaciones en 4 canales diferentes. La Tabla 3.2 muestra la estructura de este dataset.

Tabla 3.2: Dataset RIRs.

Nombre habitación	**Ejemplos (utterances)**
Great Hall	169
Octagon	169
Classroom	130

Este dataset constituye en total **468 *utterances* totales** muestreadas a 96 kHz de RIRs de 3 habitaciones. *Great Hall*, *Octagon* y *Classroom* se diferencian en dimensiones, geometría y distribución de mobiliario, lo cual permite variabilidad de respuestas al impulso entre las 3 habitaciones.

3.3. Preparación de datos

Con las RIRs descritas anteriormente y con el dataset *LibriSpeech* se pueden "simular" audios reverberados para así poder entrenar los diferentes modelos. Se realiza el siguiente procedimiento.

- Se toman 20,000 *utterances* de la división train_clean100 de *LibriSpeech* para formar el conjunto de entrenamiento.
- Tomar un ejemplo de audio, elegir una *utterance* aleatoria del dataset de RIRs y convolucionarla con el audio. La RIR pasa por *resample* antes de la convolución para que ambas señales estén muestreadas en la misma frecuencia (16 kHz). La convolución se realiza en el dominio de frecuencia, ya que en este dominio basta multiplicar las señales

[2] Descrito en el marco teórico al comienzo de la sección 2.4 "Metricas de calidad, inteligibilidad y reverberación"
[3] Como suelen ser *Reverb Challenge* o *CHiME*, los cuales fueron mencionados al finalizar el marco teórico
[4] Disponible en http://www.isophonics.net/content/room-impulse-response-data-set de la Universidad de Londres, 2008

punto a punto. La salas *Octagon* y *GreatHall* son usadas solo en el conjunto de entrenamiento, *Classroom* solo para el conjunto de prueba. Al elegir una RIR al azar se utiliza un solo canal, ya que el dataset contiene mediciones en 4 canales.

- A cada convolución realizada en el paso anterior se le agrega ruido aditivo aleatorio eligiendo cada vez un **SNR aleatorio entre 15 y 35 dB**, de modo de generar variabilidad en los datos en cuanto a ruido, pero que el ruido no degrade en demasía cada señal. Entrenar con ejemplos con diferente ruido adicionado (en este caso barriendo un intervalo de SNR de 20dB) se espera que contribuya a una mejor capacidad de generalizar.

- Para las señales ruidosas y reverberadas de la etapa anterior se generan espectrogramas utilizando 2048 de longitud de ventana FFT, con longitud de hop de 512. Se utiliza un *Mel Filterbank* para adecuar los espectrogramas a 128 *bins*. Experimentalmente se pudo corroborar que es suficiente los 128 *bins* para poder recuperar la señal temporal adecuadamente desde el espectrograma. Con menos *bins* ya se comienza a recuperar con dificultades (64 por ejemplo). Además, los espectrogramas se ajustan en el eje del tiempo a 340 *frames* utilizando la interpolación Lanczos de OpenCV en Python. Se eligen 340 debido a que en torno a ese valor es el promedio de número de *frames* sobre todos los espectrogramas de entrenamiento. Como resultado de lo anterior, todos los espectrogramas tienen el mismo tamaño de 128x340.

- Guardar el resultado de las partes anteriores y de esta manera se tiene un dataset construido donde cada ejemplo es un par de señales, donde la primera es limpia de reverberación, la cual sirve de target/etiqueta y la segunda es reverberada con una respuesta al impulso.

En adelante a los datos reverberados según el procedimiento anteriormente descrito, vale decir por convolución con RIRs, se le llamarán **datos simulados**.

Adicionalmente se incorporan las respuestas al impulso del dataset de MARDY [35] [5]. Este dataset es de utilidad, ya que pese a que las utterances son de 1 sola sala, estas son separadas en 3 niveles de distancia micrófono-*speaker*. De este modo, se puede hacer la separación entre micrófonos cercanos y lejanos en datos simulados. A los datos simulados con RIRs de este dataset se le denotará en adelante como MARDY *room*.

Resumiendo en la Tabla 3.3 se muestra la estructura de los datos generados.

Tabla 3.3: Dataset de datos simulados.

Tipo de conjunto	**Cantidad de ejemplos**	**Nombre de Sala**
Entrenamiento	10,000	*Octagon*
Entrenamiento	10,000	*Great Hall*
Prueba	500	*Classroom*
Prueba	500	MARDY *room* (Mic. cercanos)
Prueba	500	MARDY *room* (Mic. lejanos)

[5] Disponible en https://www.commsp.ee.ic.ac.uk/~sap/resources/mardy-multichannel-acoustic-reverberation-database-at-york-database/

Es importante los audios de voz del conjunto de prueba no tienen ningún ejemplo en común con lo de entrenamiento.

3.4. Respuestas al impulso simuladas

Las respuestas al impulso descritas anteriormente corresponden a mediciones de estas, las cuales son creadas en laboratorios de transmisión de voz. El problema de utilizar estas RIRs, es que en la medición hay ruido implicado. Con este propósito, es habitual en el problema de dereverberación no solo considerar mediciones en laboratorio, sino también utilizar RIRs simuladas computacionalmente. Una respuesta al impulso simulada es creada mediante un programa utilizando las dimensiones de una sala, las posiciones de *speaker* y receptor, la velocidad del sonido y lo más importante es que se pueden crear con el T_{60} o tiempo de reverberación que se desee. Con esta motivación están disponibles una gran variedad de generadores de RIRs en la web. Se seleccionó un generador de RIRs disponible en lenguaje de programación Python, el cual fue desarrollado en lenguaje de Programación C++. Este generador está disponible en https://github.com/ehabets/RIR-Generator y su documentación en https://pypi.org/project/rir-generator/.

En el presente trabajo se utiliza el generador de RIRs con el fin de visualizar en los diferentes modelos cómo influye variar el tiempo de reverberación T_{60}. Recordar que mientras mayor es el T_{60}, mayor es el nivel reverberación resultante mediante convolución. No es necesario utilizar los 500 ejemplos por sala que se mencionaron con las RIRs medidas en laboratorio, sino ocupar menos ejemplos y visualizar gráficamente cómo varía alguna de las métricas de desempeño en cada modelo al ir aumentando o disminuyendo el tiempo de reverberación. Se plantea utilizar solo 50 ejemplos de audios de voz, donde se reverbere mediante convolución con RIRs cuyo T_{60} esté en el rango [0.2, 1.0] segundos. Las dimensiones de la sala a utilizar para el generador de RIRs es de 5x4x6 (ancho x largo x profundidad en metros), un *speaker* en posición (2, 3.5, 2) y un receptor en posición (2, 1.5, 1). Notar que lo anterior son coordenas dentro de la sala y el receptor se encuentra en torno a 2 metros del *speaker*.

3.5. Datos de audio reverberados reales

Con el fin de aumentar el conjunto de prueba se busca un dataset de audios que tengan un alto nivel de reverberación, pero que estos no sean generados artificialmente mediante convolución, sino que sean medidos reverberados naturalmente. Con este fin se utiliza el conjunto de prueba de *LibriSpeech* descrito con anterioridad, pero **retransmitido**. Lo anterior significa que los audios se emiten desde un *speaker* en una habitación (que genera reverberación) y se re-captan con micrófonos en diferentes posiciones. La ventaja de utilizar estos datos es que se pueden evaluar los modelos con datos reales, ya que en aplicaciones donde se usen modelos de dereverberación estos son probados en este tipo de datos y no simulados (mediante convolución con RIRs). Se utiliza el dataset *BUT Speech@FIT Reverb Database* [6] [33], el cual contiene la división test_clean de *LibriSpeech* retransmitida. La sala donde se retransmite que es ocupada en este trabajo es una oficina, la cual en adelante se le llamará

[6] Disponible en https://speech.fit.vutbr.cz/software/but-speech-fit-reverb-database

office. Se toman 500 ejemplos en la medición más cercana y la más lejana de los micrófonos. A los datos retransmitidos en adelante se les referirá como **datos reales**. Un resumen de todos los datos utilizados (tanto simulados como reales) se muestran a continuación en la Tabla 3.4.

Tabla 3.4: Resumen de datos a utilizar.

Conjunto	**Tipo de dato**	**# ejemplos**	**Nombre Sala**
Entrenamiento	Simulado	10,000	*Octagon*
Entrenamiento	Simulado	10,000	*Great Hall*
Prueba	Simulado	500	*Classroom*
Prueba	Simulado (Mic. Cercanos)	500	MARDY *room*
Prueba	Simulado (Mic. Lejanos)	500	MARDY *room*
Prueba	Real (Mic. Cercanos)	500	*Office*
Prueba	Real (Mic. Lejanos)	500	*Office*

3.6. Modelos de dereverberación

3.6.1. Arquitecturas MLP y LSTM

Las arquitecturas utilizadas para Context-MLP, Context-LSTM y LS-LSTM [7] se detallan a continuación en las Tablas 3.5 y 3.6

Tabla 3.5: Arquitectura Context-MLP.

Context-MLP
Fully Connected con 1600 unidades + ReLU
Fully Connected con 1600 unidades + ReLU
Fully Connected con 1600 unidades + ReLU
Fully Connected con 128 unidades

Tabla 3.6: Arquitectura LSTM.

Context y LS-LSTM
Bloque LSTM con 512 unidades ocultas
Bloque LSTM con 512 unidades ocultas
Fully Connected con 128 unidades

ReLU corresponde a una función de activación que es idénticamente nula para argumentos negativos y la recta identidad para argumentos mayores o iguales a cero.

Para la arquitectura Context-MLP se toman 3 capas ocultas con las mismas unidades descritas en [6, 7], ya que con un número menor de unidades el desempeño resultó inferior y con

[7] Ver notación en el marco teórico sección 2.6 "Aplicaciones de Machine y Deep Learning en el problema de dereverberación"

un mayor número se obtienen resultados similares, pero con un tiempo de aprendizaje más lento. Similarmente la "forma" de la arquitectura LSTM-DNN es basada en [30, 2], donde las 512 unidades fueron elegidas también de forma manual, buscando un desempeño similar o mayor al MLP, pero que no suponga un tiempo de entrenamiento exageradamente alto. Tanto Context-MLP como Context-LSTM reciben 128x11 valores de entrada (11 *frames*/cuadros de tiempo). La arquitectura LS-LSTM tiene la misma estructura que Context-LSTM, solo que cambia el número de unidades de entrada y el lazo que conecta la entrada con la salida (ver Figura 2.19).

Para Context MLP y LSTM se realiza una normalización de los espectrogramas de modo de que posean una media de 0 y varianza unitaria. Para LS-LSTM se realiza la misma normalización, pero *frame* a *frame*.

3.6.2. Arquitectura U-net

En el marco teórico se mencionó cómo se puede utilizar la estructura de un autoencoder para el problema de dereverberación. Se utiliza como base una arquitectura U-net [9] diseñada para un problema completamente diferente que corresponde a segmentación de imágenes [8]. La arquitectura utilizada se describe en la Figura 3.1

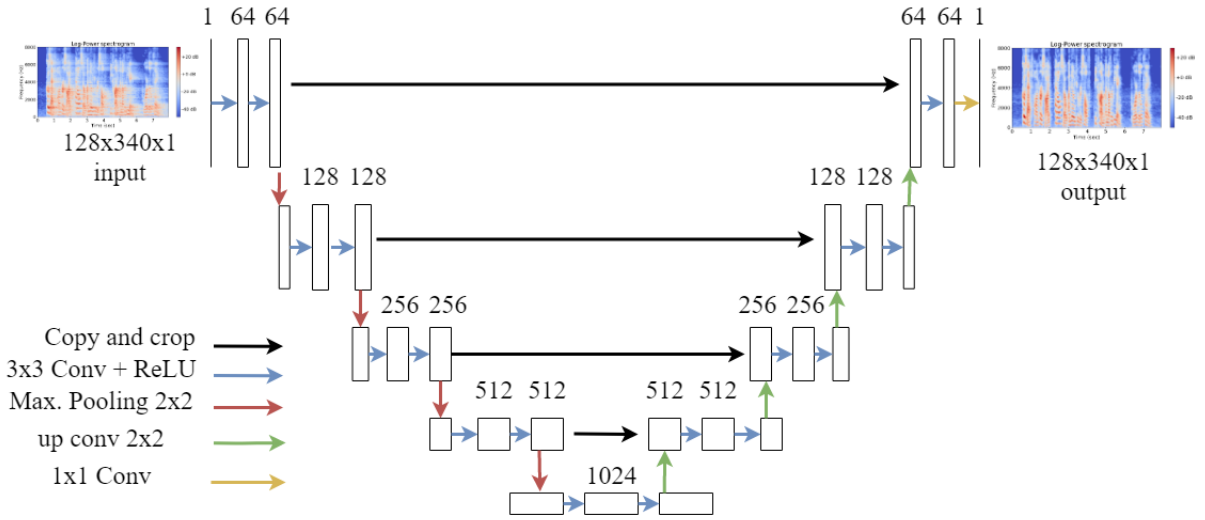


Figura 3.1: Arquitectura U-net. Elaboración propia.

La red U-net se caracteriza por *skip connections*, es decir, conexiones entre bloques de un mismo número de canales, las cuales son las flechas de color negro en la Figura 3.1. El espectrograma de entrada a la red es de 128x340x1 y la salida es del mismo tamaño. La red U-net ha mostrado resultados adecuados en el problema de dereverberación, pero usualmente es entrenado y evaluado con ruido fijo, lo cual no permite ver todo el potencial de la red en ambientes ruidosos [8].

Para esta parte **no se realiza normalización en los espectrogramas**, debido a que experimentalmente se pudo corroborar que termina por degradar las generaciones. El procesamiento de los espectrogramas de Mel utilizando *Librosa* (de Python) permite mantener los

[8] Disponible en https://github.com/milesial/Pytorch-UNet

valores de potencia espectral en un intervalo deseado, por lo cual la normalización no contribuye en este caso. Lo anterior ocurre tanto con normalización standard (media 0 y varianza unitaria) como con normalización lineal que confina la salida en exactamente el intervalo [-1, 1].

3.6.3. Late Reverberation Supression U-net

Basado en la arquitectura *Late Reverberation Supression* LSTM (LS-LSTM) planteada por Yan Zhao et al [2], se propone utilizar un lazo similar conectando el *input* con el *output* de la red, pero utilizando una arquitectura U-net. La Figura 3.2 muestra cómo se propone utilizar *Late Reverberation Supression*, pero usando una arquitectura U-net, el lazo que conecta entrada/salida se muestra en línea punteada. La idea base es que utilizando las capas convolucionales de la arquitectura U-net, no se aprenda un mapeo reverberado-dereverberado, sino que se aprenda a estimar qué restarle al espectrograma de entrada para que la salida sea dereverberada. Lo anterior, al igual que la idea original para LSTM se interpreta como "estimar la reverberación" y sustraerla al espectrograma de entrada.

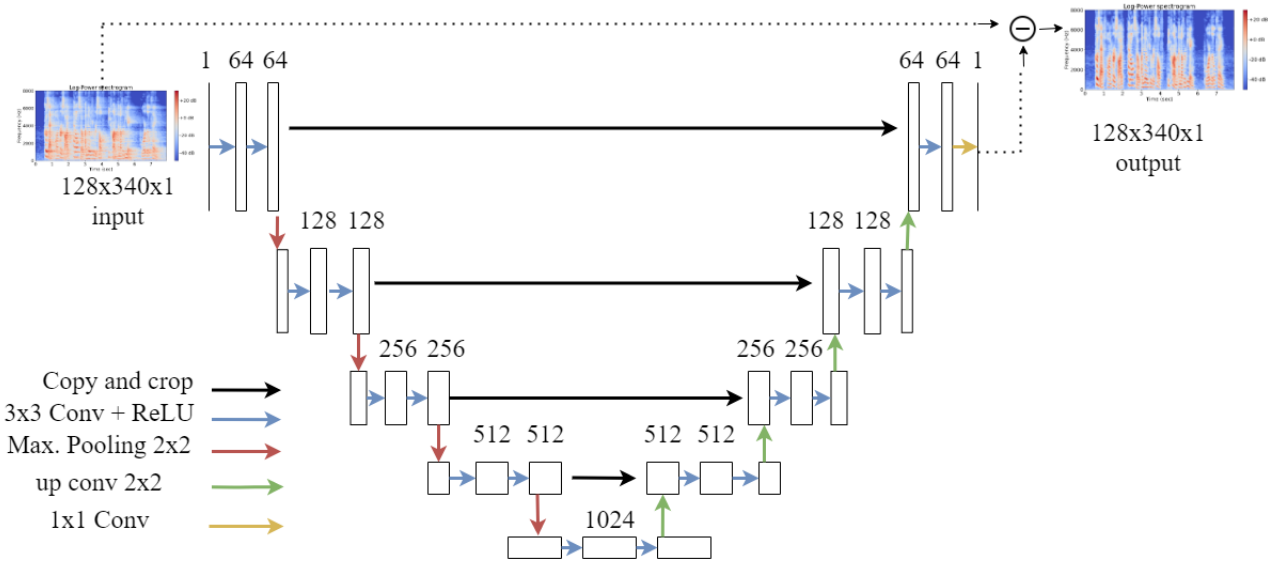


Figura 3.2: Arquitectura *Late Reverberation Supression* U-net (LS U-net). Elaboración propia.

Se espera que esta arquitectura obtenga mejores resultados que U-net tradicional planteada por Ernst et al [8] (sin el lazo conectando entrada con salida). En adelante se le denotará a esta arquitectura propuesta como LS U-net.

3.6.4. Arquitectura GAN

Para esta parte se debe plantear una arquitectura tanto para el generador, como también para el discriminador. Como generador se utiliza la misma arquitectura U-net descrita con anterioridad y la arquitectura del discriminador se muestra a continuación en la Tabla 3.7

Tabla 3.7: Arquitectura Discriminador GAN.

Arquitectura Discriminador
Convolución con 64 filtros de 7x7 + leakyReLU (alpha=0.1)
Convolución con 128 filtros de 7x7 + leakyReLU (alpha=0.1) + BN
Convolución con 256 filtros de 7x7 + leakyReLU (alpha=0.1) + BN
Convolución con 512 filtros de 7x7 + leakyReLU (alpha=0.1) + BN
Convolución con 1024 filtros de 7x7 + leakyReLU (alpha=0.1) + BN
Fully Connected con 1 unidad de salida + leakyReLU (alpha=0.1)
Sigmoide (confina la salida entre 0 y 1)

leakyReLU corresponde a una función de activación dada por $\text{leakyReLU}(x) = \begin{cases} x & \text{si } x \geq 0 \\ \alpha x & \text{si } x < 0 \end{cases}$, con α un parámetro ajustable que en este caso es 0.1

BN denota *Batch Normalization* y esta red es utilizada con una función de pérdida de *Binary Cross Entropy*. El generador es utilizado también con BCE, pero con el término MSE sumado (ponderado por un hiperparámetro λ) entre la salida y el espectrograma *target*, lo cual implementa la función de pérdida descrita en la ecuación (3.2). Para esta parte no se utiliza normalización sobre los espectrogramas, ya que se observó experimentalmente que se terminan por degradar las generaciones.

3.7. Métricas

Las métricas PESQ, CD, LLR, fwSNRseg y SRMR descritas el marco teórico están originalmente estudiadas e implementadas con código disponible en un libro en lenguaje de programación Matlab [29]. En este trabajo se utiliza una reciente adaptación (2019-2020) de estos códigos de Matlab a lenguaje de programación Python. Estas implementaciones se pueden encontrar en https://github.com/schmiph2/pysepm y son las que se ocupan en este trabajo.

3.8. FD-NDLP

Este método se hace necesario utilizarlo en este trabajo, debido a su popularidad y a que también es un método no supervisado. El método de *Weighted Prediction Error* WPE [26] fue publicado con una implementación incluida en Matlab por lo mismos creadores. La forma de resolverlo mediante NDLP tiene también implementación de sus creadores tanto en dominio temporal, como temporal-frecuencial (FD-NDLP), también en Matlab. Una adaptación de estos códigos a Python está disponible en https://github.com/helianvine/fdndlp y es la utilizada en este trabajo. La implementación es ampliamente detallada y documentada.

3.9. Resumen de la metodología

A modo de resumen, la Figura 3.3 muestra por pasos cómo está estructurada la metodología del trabajo de título.

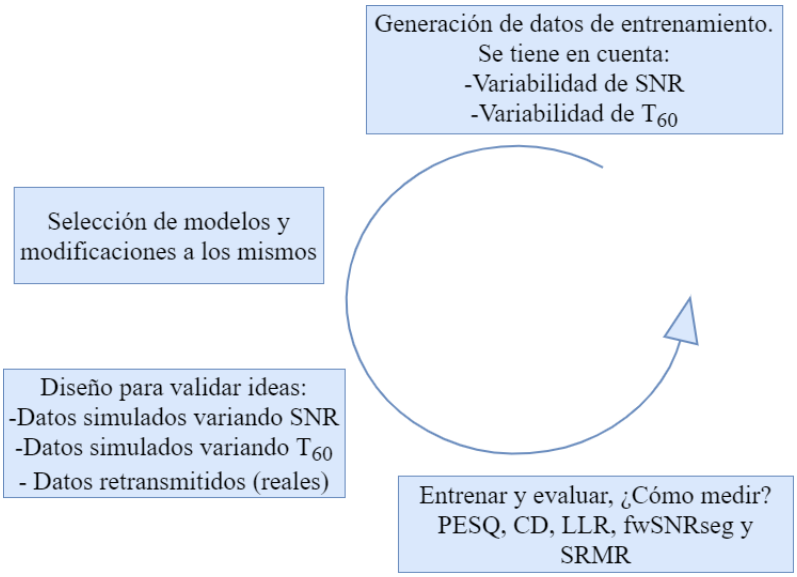


Figura 3.3: Arquitectura U-net. Elaboración propia.

Como se puede ver, se considera desde la generación de los datos de entrenamiento hasta la evaluación de los modelos mediante las métricas escogidas. El proceso se considera como de flujo circular, ya que luego del entrenamiento y evaluación de los modelos, estos dan indicios a si el funcionamiento es el esperable o se debe volver a iterar alguno de los pasos anteriores.

Notar la importancia del segundo paso de selección de modelos, ya que es en ese punto donde se seleccionan todos los enfoques de resolver el problema a utilizar en el trabajo. Adicionalmente, es aquí donde se proponen variaciones a métodos existentes y también se propone una arquitectura nueva combinando diferentes ideas existentes en la literatura.

Capítulo 4

Resultados y Análisis

Este capítulo se enfoca en 3 aspectos principales: resultados y dificultades en el entrenamiento de los modelos, evaluación cuantitativa del desempeño de de los modelos y luego evaluación cualitativa de los mismos. La evaluación cuantitativa se realiza utilizando métricas PESQ, CD, LLR, fwSNRseg y SRMR. Se evalúa en diferentes niveles de ruido adicionado, en diferentes niveles de reverberación y también tanto en datos simulados (generados mediante convolución) como en datos reales (naturalmente reverberados). La evaluación cualitativa se realiza analizando espectrogramas de salida de los diferentes modelos tanto para datos reales como simulados.

Todo el código utilizado para obtener los resultados expuestos en esta sección se puede encontrar en su totalidad en https://github.com/DiegoLeon96/Neural-Speech-Dereverberation

4.1. Resultados de entrenamiento de modelos neuronales

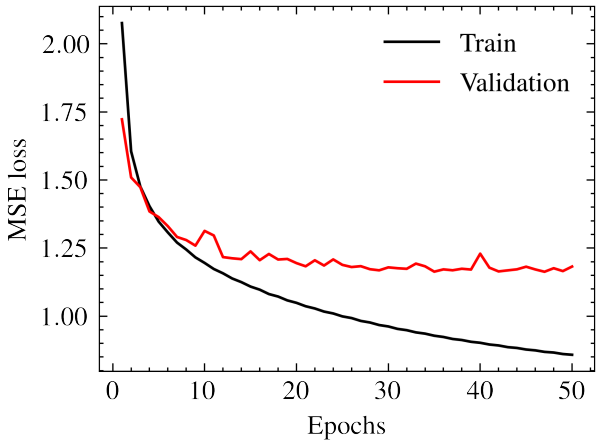
Las divisiones de datos y arquitecturas específicas fueron presentadas en el capítulo anterior. En esta parte, se exponen los resultados de los entrenamientos. La Tabla 4.1 muestra los datos específicos que se utilizaron para el entrenamiento de los 4 modelos neuronales.

Tabla 4.1: Parámetros de entrenamiento.

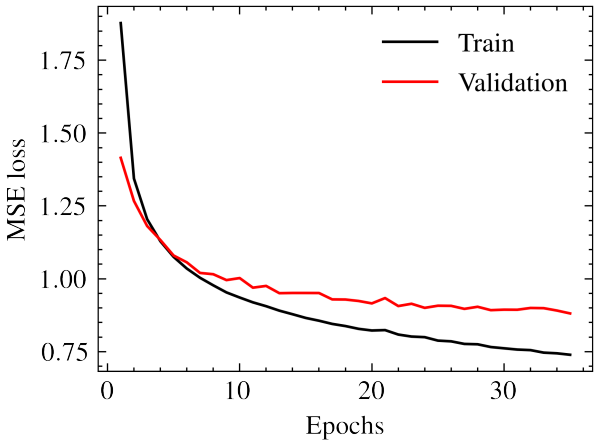
	Context-MLP	Context-LSTM	LS-LSTM	U-net	LS U-net	U-net GAN
Épocas de entrenamiento	50	35	30	50	30	10
Tiempo de entrenamiento	$\approx$ 6h	$\approx$ 8h	$\approx$ 8h	$\approx$ 5h	$\approx$ 5h	$\approx$ 1.5h
Tasa de aprendizaje	2e-4	2e-4	2e-4	1e-3	2e-4	1e-3
Tamaño de batch	1	1	1	16	16	16

Todos estos modelos fueron entrenados en GPU utilizando Google Colaboratory. Se utilizó optimizador Adam en todos los casos. Los modelos entrenados por menos de 50 épocas se debe a que no se vislumbraron cambios en la función de pérdida de validación tras las últimas 5 épocas. El entrenamiento GAN en especifico, se realizó por 10 épocas $\approx$ 10.5k iteraciones, debido a que la convergencia en este caso es más rápida que el resto de modelos. Notar que el tamaño de batch entregado es un número de espectrogramas, por lo que los modelos Context-MLP, Context-LSTM y LS-LSTM (que actúan sobre *frames* y no espectrogramas completos) en realidad tienen un tamaño de batch equivalente en *frames*. El modelo LS-LSTM tiene un tamaño de batch de 340 *frames* (dimensionalmente 1 espectrograma) y los modelos que utilizan *Context Information* (Context-MLP y LSTM) utilizan 330 *frames* (340 agrupados en ventanas de 11).

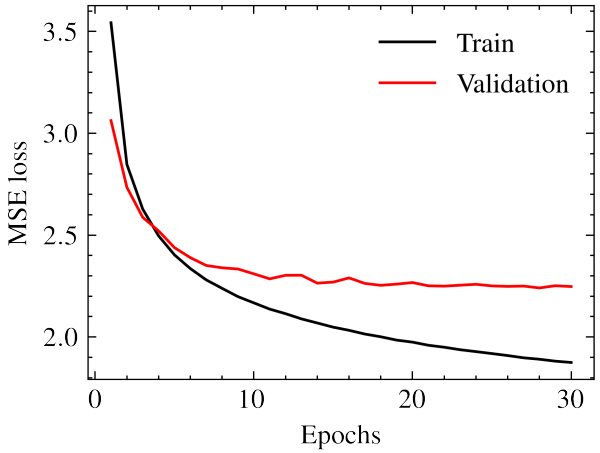
Las figura 4.1 muestra las funciones de pérdida de los 4 modelos neuronales ocupados. En todos los modelos se reservó un 15 % para validación.



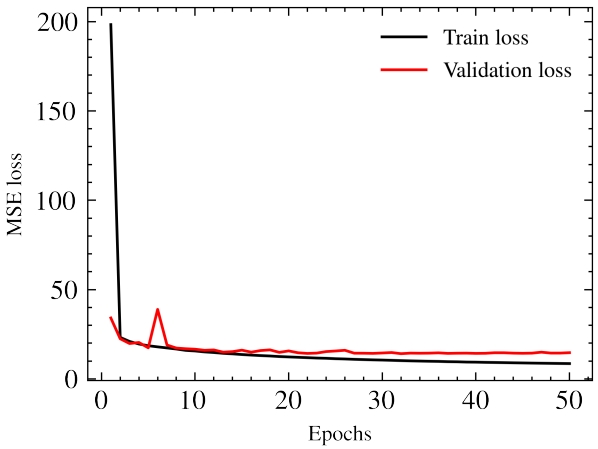
(a) Función de pérdida MSE en Context-MLP.



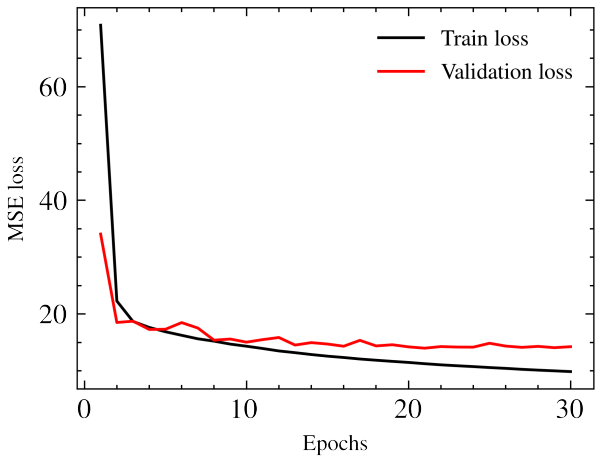
(b) Función de pérdida MSE en Context-LSTM.



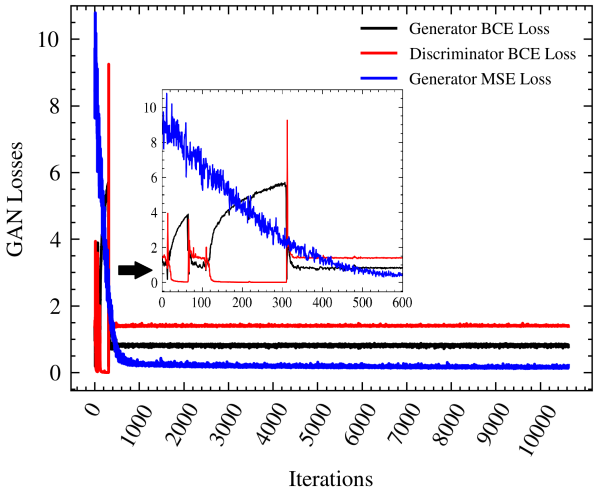
(c) Función de pérdida MSE en LS-LSTM.



(d) Función de pérdida MSE en U-net.



(e) Función de pérdida MSE en LS U-net.



(f) Función de pérdida MSE y BCE en U-net como generador GAN.

Figura 4.1: Funciones de pérdida

Se puede ver que el MSE en todos los casos es decreciente y no se observa sobre-entrenamiento u *overfitting* observando las funciones de pérdida de validación. En el caso de las funciones

de pérdidas BCE en el modelo GAN, se puede ver que en el caso del discriminador disminuye hasta aproximadamente las 500 iteraciones y luego aumenta abruptamente hasta converger. En el caso del generador, se puede ver que la parte BCE primero aumenta hasta cercano a las 500 iteraciones y luego disminuye hasta converger. Este comportamiento muestra que el entrenamiento GAN fue exitoso, ya que al aumentar el BCE del discriminador, se muestra que este presenta dificultades para distinguir entre los espectrogramas de salida del generador y los espectrogramas *target*. El hiper-parámetro λ mencionado en el marco teórico para la arquitectura GAN dereverberativa se fijó en λ = 1e-2, el cual se eligió experimentalmente para que el MSE no predomine sobre el término BCE.

Con respecto al modelo U-net con entrenamiento GAN, se destacan las siguientes dificultades, las cuales son también mencionadas en la publicación donde se presentaron las redes Generativas Adversarias por primera vez en 2014 [24]:

- Mantener un balance de "poder" entre el discriminador y el generador. Si el discriminador es profundo respecto al generador por ejemplo, el discriminador nunca se confundirá con los ejemplos del generador. El término *Binary Cross Entropy* (BCE) de la función de pérdida asociada al discriminador decae a 0 y no volverá a aumentar. Como consecuencia, el aprendizaje en este caso fracasa, ya que para el entrenamiento GAN el término BCE del discriminador debe aumentar hasta que logre converger.
- Ajustar los parámetros del discriminador solo si el término de la función de pérdida *Binary Cross Entropy* (BCE) asociado está por sobre un umbral. De lo contrario, puede ocurrir que dicho término decaiga a 0 sin volver a aumentar, independientemente del balance entre discriminador y generador mencionado en el punto anterior. Para el entrenamiento en este caso bastó ajustar el discriminador hasta la iteración 500 y en el generador se ajustan sus parametros por todo el entrenamiento (10.5k iteraciones). Lo anterior permitió que nunca el término BCE decaiga a 0, resultando en un entrenamiento GAN exitoso.

4.2. Evaluación cuantitativa

Los resultados obtenidos en esta sección se muestran con tablas y gráficos de barra. En cada tabla, se muestran PESQ, CD, LLR, fwSNRseg y SRMR **promedio** sobre 500 ejemplos (no necesariamente los mismos 500 ejemplos en todas las tablas). Cada tabla destaca en negrita cuál es el mejor modelo según cada métrica, ya que dependiendo de la métrica el mejor desempeño puede ser el modelo con el menor o el mayor valor en dicha métrica.

4.2.1. Datos simulados variando el ruido adicionado

La tabla 4.2 muestra la evaluación mediante las 5 métricas descritas en el marco teórico en la sala *Classroom* descrita en la metodología. Recordar que las utterances de RIRs de *Classroom* no fueron usadas para el entrenamiento de los modelos. El SNR del ruido utilizado en cada ejemplo es aleatoriamente elegido en el intervalo [15, 35] dB.

Se puede ver que los modelos Context-MLP y Context-LSTM no muestran un desempeño sobresaliente en este caso, donde en las métricas que supone una mejora, esta es leve en comparación a la señal reverberada. Lo mismo ocurre en menor medida para el caso de

Tabla 4.2: Evaluación de modelos en datos simulados en *Classroom*

	PESQ	CD	LLR	fwSNRseg	SRMR
Reverberant	2.02	5.26	0.77	7.07	3.21
Context-MLP	2.01	4.95	0.84	6.88	3.24
Context-LSTM	2.09	4.84	0.82	7.30	3.24
LS-LSTM	2.13	4.53	0.69	7.72	3.89
FD-NDLP	2.29	5.27	0.68	9.05	5.25
U-net	2.67	**3.62**	0.43	9.99	5.98
LS U-net	**2.72**	**3.62**	**0.42**	**10.14**	6.42
U-net GAN	2.67	3.68	0.43	9.78	**7.10**

LS-LSTM. Preliminarmente, se plantea que esto se debe a que el ruido en algunos casos puede perjudicar apreciablemente a estos 3 modelos o también puede darse que ambos modelos posean una pobre capacidad de generalizar sobre nuevos ejemplos independiente del ruido.

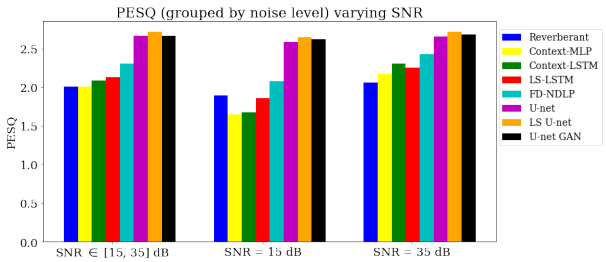
El modelo FD-NDLP que no es neuronal no es el con mejor desempeño, pero supone una mejora frente a la señal reverberada y ruidosa según 4 de las 5 métricas (excepto CD). Las métricas PESQ y LLR muestran una mejora en este caso, pero no es abrupta. Se puede ver que según las métricas fwSNRseg y SRMR la mejora es drástica en relación a la señal reverberada, lo cual muestra su efectividad pese a que es un método no supervisado.

Los métodos de U-net en sus 3 variantes muestran un desempeño radicalmente superior a la señal reverberada y ruidosa, lo cual se observa **bajo todas las métricas**. Se puede ver que la arquitectura propuesta LS U-net muestra los mejores resultados según PESQ, CD, LLR y fwSNRseg, pero la mejora no es abrupta en comparación a U-net y U-net GAN. En la métrica SRMR es donde se marcan más diferencias. Según la métrica SRMR, la arquitectura GAN (seguida de LS U-net) muestra mejor desempeño, lo cual se interpreta a que por el tipo de aprendizaje en GAN se aprenden aspectos o patrones dereverberativos dentro del espectrograma que no son posibles utilizando solamente el MSE como función de pérdida. Estos modelos basados en U-net es esperable que capturen información más descriptiva para la dereverberación, dado que exploran el espectrograma completo a través de capas convolucionales. Los modelos Context-MLP, Context-LSTM y LS-LSTM no esploran la totalidad del espectrograma, sino que a trozos, a través de *frames* o cuadros de tiempo.

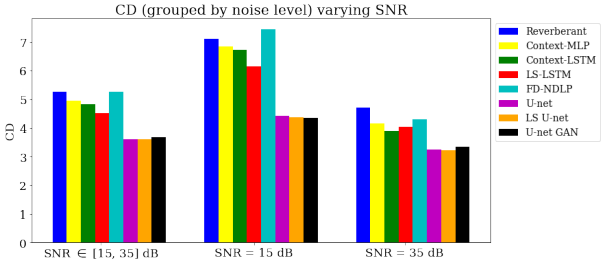
Con el fin de corroborar el comportamiento de los modelos y de las métricas de desempeño con el ruido, se repite el procedimiento para obtener los resultados de la Tabla 4.2, pero esta vez con ruido fijo en todos los ejemplos. En la Tabla 4.3 se muestran los resultados también simulados con las RIRs de *classroom* para exactamente los mismos audios utilizados en la Tabla 4.2, pero esta vez con SNR fijo de SNR = 15dB y SNR = 35dB. Los mismos resultados se muestran gráficamente en las Figuras 4.2 y 4.3 para PESQ, CD, LLR y fwSNRseg, mientras que en las Figuras 4.4 y 4.5 se expone solamente SRMR (por separado debido a que es la métrica que de momento mejor capta la reverberación). Notar que los valores 15 y 35 son los extremos del intervalo de SNR utilizado en los resultados de la Tabla 4.2.

Tabla 4.3: Evaluación con ruido fijo para SNR = 15**dB** y SNR = 35**dB**

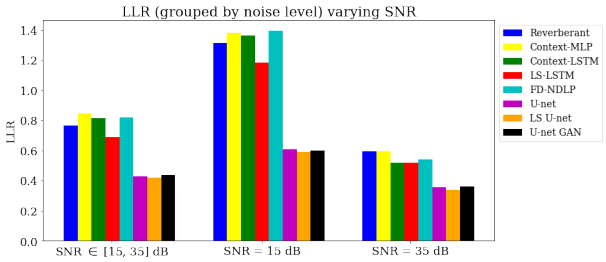
	PESQ		CD		LLR		fwSNRseg		SRMR	
SNR (dB) ⟶	15	35	15	35	15	35	15	35	15	35
Reverberant	1.98	2.11	7.30	5.39	1.36	0.81	6.38	7.69	3.08	3.17
Context-MLP	1.66	2.18	6.84	4.16	1.38	0.59	5.04	7.74	2.06	3.94
Context-LSTM	1.68	2.31	6.73	3.91	1.36	0.52	5.20	8.42	1.93	4.06
LS-LSTM	1.86	2.25	6.17	4.04	1.18	0.52	5.98	8.34	2.71	4.75
FD-NDLP	2.09	2.43	7.45	4.30	1.39	0.54	6.96	9.66	4.25	4.47
U-net	2.59	2.66	4.44	3.26	0.61	0.36	9.35	10.00	5.93	5.61
LS U-net	**2.65**	**2.72**	4.38	**3.23**	**0.59**	**0.34**	**9.56**	**10.20**	6.30	5.98
U-net GAN	2.62	2.69	**4.37**	3.34	0.60	0.36	9.15	9.82	**7.18**	**6.73**



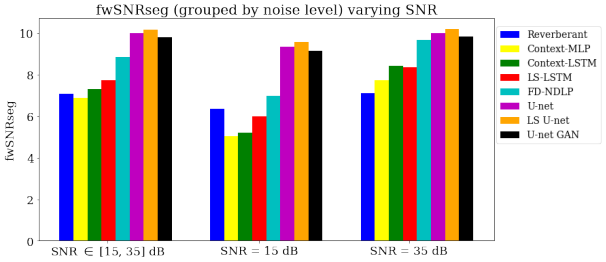
(a) Desempeño en PESQ.



(b) Desempeño en CD.

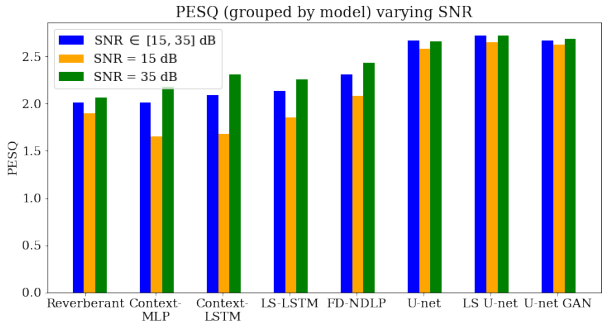


(c) Desempeño en LLR.

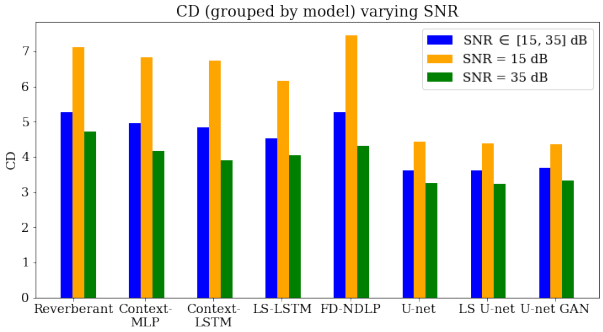


(d) Desempeño en fwSNRseg.

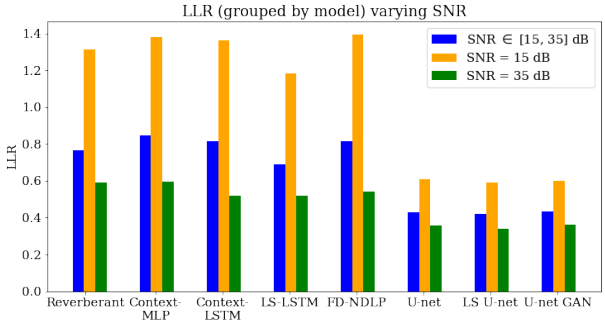
Figura 4.2: Desempeño en PESQ, CD, LLR y fwSNRseg particionando el gráfico por SNR.



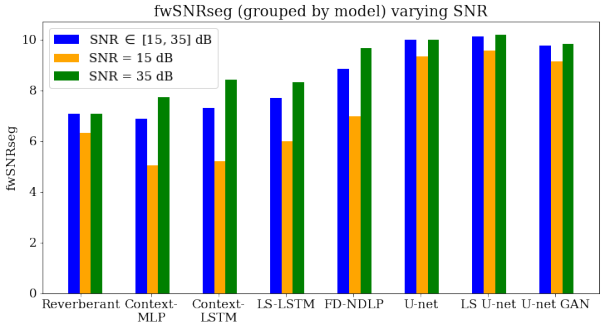
(a) Desempeño en PESQ.



(b) Desempeño en CD.



(c) Desempeño en LLR.



(d) Desempeño en fwSNRseg.

Figura 4.3: Desempeño en PESQ, CD, LLR y fwSNRseg particionando el gráfico por modelo.

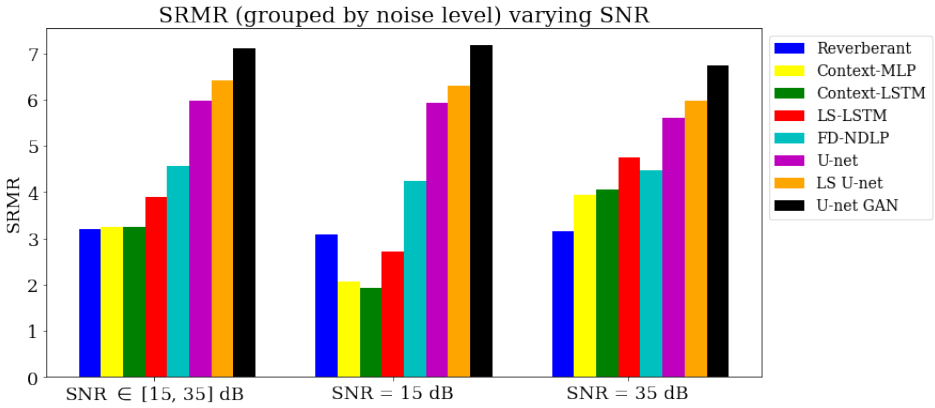


Figura 4.4: Evaluación mediante SRMR para SNR fijo y SNR variable particionando por SNR.

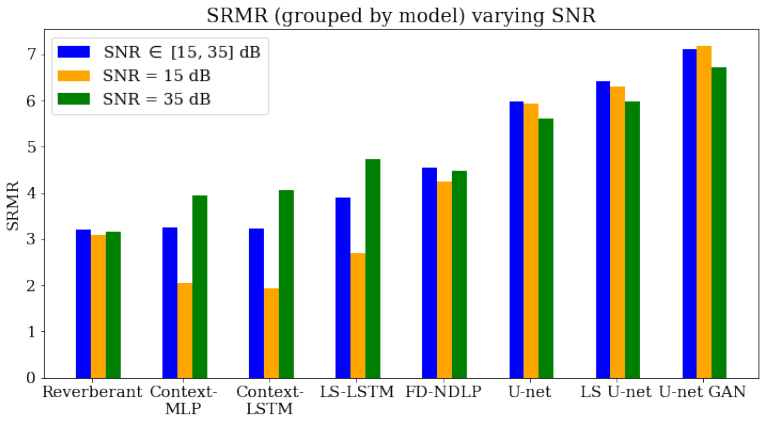


Figura 4.5: Evaluación mediante SRMR para SNR fijo y SNR variable particionando por modelo.

Se puede ver que los modelos Context-MLP y Context-LSTM esta vez si suponen una mejora apreciable según todas las métricas para SNR = 35dB observando la Tabla 4.3. Se puede ver que no suponen una mejora tan elevada como lo hacen el resto de modelos, pero si son capaces de contribuir apreciablemente al problema de dereverberación. Adicionalmente, con SNR = 15dB el desempeño de estos modelos es radicalmente inferior en comparación a la señal reverberada, lo cual da cuenta que con este nivel de ruido no se contribuye adecuadamente a la dereverberación.

Los modelos basados en U-net varían respecto a los resultados expuestos con SNR variable en la Tabla 4.2, pero no apreciablemente, sino que tienden a permanecer prácticamente invariantes (sobretodo en la métrica SRMR). Se puede ver en la Figura 4.5 que la variación de estos últimos según SRMR es insignificante frente a la variación que presentan los modelos MLP y LSTM con *Context Information*.

En la Figura 4.5 se puede ver que el valor de SRMR en el audio reverberado prácticamente no cambia al variar el SNR. Lo anterior, permite desprender la efectividad de la métrica SRMR en medir **netamente reverberación**, no siendo sensible al ruido. Lo anterior no ocurre para las métricas CD y LLR (observar figuras 4.2 y 4.3), ya que se puede ver en el método FD-NDLP por ejemplo, que este contribuye a la dereverberación con SNR = 15dB (según SRMR y fwSNRseg), pero las 2 métricas mencionadas no muestran una mejora. Se vislumbra preliminarmente una **deficiencia en estas métricas en este contexto**, ya que no son capaces de medir una mejora en cuanto a reverberación si el nivel de ruido es apreciablemente alto (en términos de SNR).

La métricas PESQ y fwSNRseg se puede ver en las Figuras 4.2 y 4.3 que se comportan coherentemente en comparación a CD y LLR. Independientemente del ruido suponen una mejora en todos los modelos, excepto Context-MLP y LSTM (los cuales con 15 dB de ruido no son capaces de contribuir al problema). Sin embargo, se puede observar, que pese a que estas 2 métricas son más robustas al ruido que CD y LLR, estas no son tan robustas como SRMR.

Con el fin de ampliar las ideas desprendidas acerca de cómo influye el ruido en todos los modelos, se realiza un barrido de ruido con SNR en cada valor del intervalo [15, 35] dB. Para acelerar el proceso computacionalmente, se consideran solo 50 ejemplos de los 500 con

los que se venía evaluando anteriormente (en tablas y gráficos de barra). En la Figura 4.6 se muestra el SRMR promedio para cada valor de SNR en [15, 35] dB para 50 ejemplos. Se utiliza solamente el SRMR como métrica en esta parte, ya que anteriormente se corroboró que es la métrica que mejor permite medir un desempeño en el contexto de dereverberación.

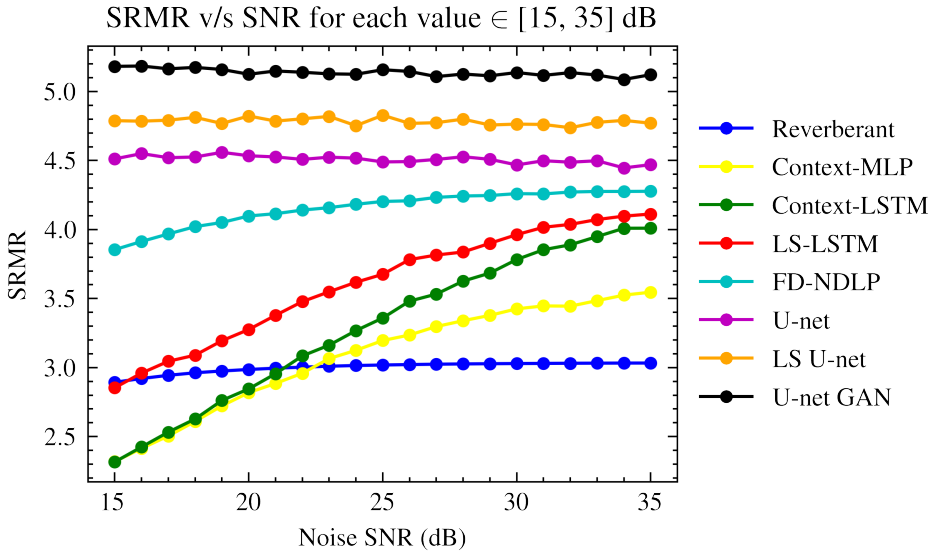


Figura 4.6: Barrido de ruido con SNR en todo el intervalo [15, 35] dB.

Se puede apreciar que los modelos basados en U-net muestran un SRMR prácticamente invariante en todo el intervalo de SNR. Se puede ver que lo mismo ocurre para la señal reverberada. Se desprende una vez más la efectividad de la métrica SRMR en cuantificar un desempeño en el contexto dereverberación, incluso bajo un nivel apreciable de ruido.

Los modelos basados en MLP y LSTM se puede apreciar que disminuyen abruptamente en el valor de SRMR promedio al disminuir el SNR (es decir, empeora su desempeño si aumenta el ruido). Con 15 dB el modelo LS-LSTM no mejora frente a la señal reverberada y los modelos de Contexto (Context-MLP y LSTM) muestran un desempeño inferior a esta. Al aumentar el valor de SNR (es decir, disminuir el ruido) todos estos modelos aumentan progresivamente el valor de SRMR, donde con SNR en torno a 35 dB el modelo LS-LSTM comienza a mostrarse competitivo frente a FD-NDLP.

En el caso de FD-NDLP se puede ver que varía el SRMR al variar el SNR, ya que disminuye su desempeño al aumentar el ruido. Sin embargo, lo anterior no ocurre abruptamente como en el caso de los modelos basados en MLP y en LSTM. Se desprende que FD-NDLP posee cierta robustez frente al ruido, pese a ser no supervisado y a que no contribuye en absoluto a cancelar o eliminar el ruido (*Denoising*).

En base a los resultados obtenidos, para la utilización de los modelos basados en MLP y LSTM es necesaria una estrategia de *Denoising* (anterior a la dereverberación) si el SNR está por bajo de un umbral, es decir, una estrategia de cancelación o de reducción de ruido. Esto no ocurre para el modelo no supervisado y para las arquitecturas basadas en U-net, ya que se muestran robustas en todo el intervalo de SNR. Lo anterior se debe a que se aprende a eliminar en parte el ruido de manera implícita, ya que explora todo el espectrograma a

través de sus capas convolucionales.

4.2.2. Datos simulados variando la distancia *speaker*-micrófono

En esta sección se muestran los resultados para los datos simulados utilizando las RIRs del dataset de MARDY descrito en la metodología. Las RIRs de MARDY poseen un T_{60}[1] de 447 ms para mediciones lejanas al *speaker* (a 3m) y de 291 ms para mediciones cercanas (a 1m) [35]. Notar que 291 ms como T_{60} representan un nivel muy bajo de reverberación, lo cual permite evaluar los modelos en situaciones desafiantes donde la reverberación no es predominante (pero está presente) en los audios de voz. La Tabla 4.4 muestra los resultados para micrófonos lejanos y cercanos respectivamente. El ruido es fijado en SNR = 35dB para esta parte, de modo que el ruido no sea un factor que empeore el desempeño de los modelos.

Tabla 4.4: Resultados en datos simulados utilizando el dataset de RIRs MARDY. Los tiempos de reverberación son de 291 y 447 ms para micrófonos cercanos y lejanos respectivamente.

	PESQ		CD		LLR		fwSNRseg		SRMR	
	Near	Far	Near	Far	Near	Far	Near	Far	Near	Far
Reverberant	2.57	2.15	5.25	5.71	0.85	0.97	8.68	6.58	5.21	4.49
Context-MLP	2.09	1.87	5.48	5.62	1.02	1.07	6.72	5.82	3.13	2.81
Context-LSTM	2.14	1.90	5.49	5.64	0.99	1.05	7.21	6.12	3.05	2.60
LS-LSTM	2.38	2.07	4.97	5.29	0.81	0.92	8.06	6.64	4.53	4.12
FD-NDLP	2.71	2.24	5.44	5.83	0.90	1.00	8.57	6.60	6.03	5.34
U-net	2.65	2.28	4.12	4.57	0.54	0.65	9.23	7.52	5.47	4.88
LS U-net	**2.74**	**2.36**	**4.09**	4.59	**0.52**	**0.64**	**9.32**	**7.63**	5.73	5.36
U-net GAN	2.72	**2.36**	4.11	**4.56**	0.53	**0.64**	9.24	**7.63**	**6.60**	**6.19**

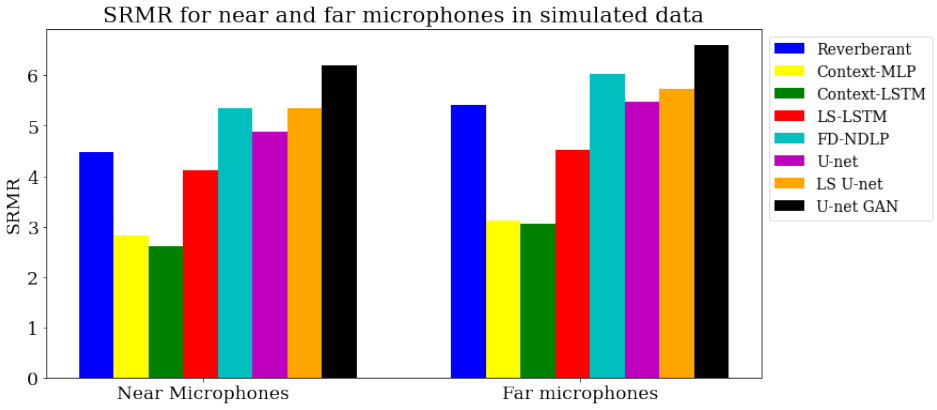


Figura 4.7: Resultados sobre datos simulados para micrófonos cercanos y lejanos.

[1] Concepto descrito en el marco teórico, sección 2.1.2 de "Respuestas al impulso en audio"

Se puede ver nuevamente que tanto para micrófonos cercanos como lejanos los modelos Context-MLP y Context-LSTM no contribuyen a una mejora. Lo mismo ocurre, pero de una forma menos abrupta en el caso de LS-LSTM, ya que tiende a dejar la señal reverberada prácticamente inalterada. Lo anterior ocurre según todas las métricas a excepción de CD en algunos casos. En este caso, el ruido es fijo de SNR = 35dB, por lo que el ruido no es el factor que empeora su desempeño. Se desprende que estos modelos muestran problemas para generalizar sobre nuevos ejemplos dependiendo del tiempo de reverberación T_{60}. El hecho que estos modelos muestren un desempeño abruptamente inferior en comparación a la señal reverberada es inaceptable, ya que se espera que en el peor de los casos dejen la señal inalterada.

El modelo FD-NDLP supone una mejora en fwSNRseg y SRMR tanto para micrófonos cercanos como lejanos. Lo anterior muestra que contribuye a mejorar la calidad de la señal en ambos casos a través de dereverberación. Sin embargo, se observa que no supone una mejora según las métricas CD y LLR, pese a que contribuye a la dereverberación. Nuevamente, se desprende que CD y LLR no son métricas del todo adecuadas en el contexto de reverberación. Con anterioridad se pudo corroborar que estas métricas presentan sensibilidad al ruido, pero en este caso el ruido es fijo SNR = 35dB, por lo que su problema en este caso es que no son capaces de "detectar" un nivel de reverberación que no es predominante en la señal a evaluar.

Los modelos basados en U-net muestran los mejores resultados tanto para micrófonos lejanos como cercanos. Dentro de estos 3 modelos, se puede ver nuevamente que el modelo con aprendizaje GAN posee los mejores resultados según SRMR, seguido de U-net con *Late Reverberation Supression* y por último la arquitectura U-net tradicional. Las diferencias en SRMR que se marcan entre estos 3 modelos son apreciables, donde U-net queda atrás en desempeño, tanto para micrófonos cercanos como lejanos. Lo anterior permite desprender la efectividad en dereverberación de *Late Reverberation* y del aprendizaje GAN aplicado a una arquitectura U-net. Según el resto de métricas (PESQ, CD, LLR y fwSNRseg), el mejor desempeño lo tiene la arquitectura propuesta LS U-net, pero la mejora es pequeña en comparación a U-net tradicional y a U-net GAN. Los 3 modelos basados en U-net poseen una adecuada capacidad de generalizar sobre ejemplos nuevos para diferentes niveles de reverberación, además de ser robustos frente al ruido según se vio en la sección anterior variando el SNR.

Para el caso de micrófonos cercanos la mejora que supone el modelo U-net tradicional (sin aprendizaje GAN) no es excesiva, lo que es consistente, ya que para micrófonos cercanos la reverberación está presente, pero es muy reducida. Pese a ser reducida, no se muestra en ningún caso un desempeño inferior a la señal reverberada.

4.2.3. Comportamiento en RIRs simuladas

En el capítulo de metodología se mencionó que se utilizaría un generador de RIRs, de modo de poder variar el tiempo de reverberación T_{60}. La Figura 4.8 muestra esta variación y su efecto en la métrica SRMR, la cual anteriormente ha mostrado los mejores resultados detectando reverberación incluso en ambientes ruidosos. Se toman 8 valores de T_{60} en el intervalo [0.2, 1] segundos con espaciamento de 0.1 segundos. Se utiliza el SRMR promedio sobre 50 ejemplos de los 500 ejemplos por sala que fueron utilizados para las tablas con métricas y gráficos de barra. El ruido es fijado nuevamente en SNR = 35dB de modo que se

tenga un nivel reducido de ruido y no suponga un problema a ninguno de los modelos.

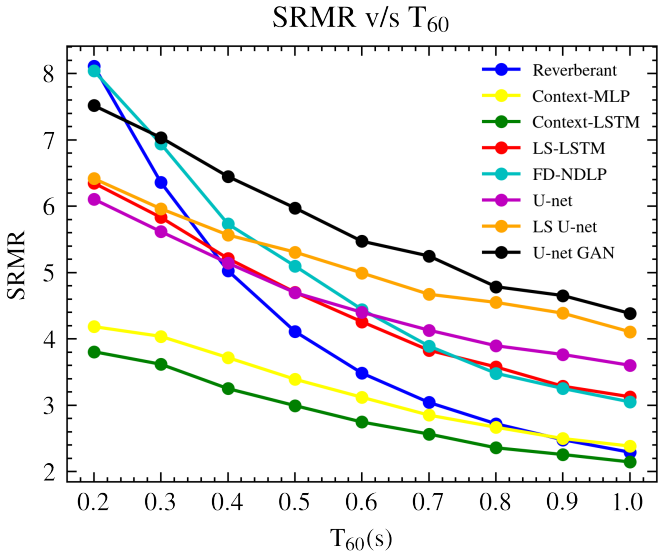


Figura 4.8: Variación de T_{60} utilizando RIRs simuladas

Es esperable que todos los modelos tengan mayores dificultades para dereverberar conforme aumenta el tiempo de reverberación, por lo cual es esperable que todas las curvas decaigan como se muestra en la Figura 4.8. Se puede observar que con $T_{60} = 0.2s$ ninguno de los modelos logra superar en SRMR a la señal reverberada, lo cual también es esperable, puesto que los 0.2s representan un ínfimo nivel de reverberación. Conforme aumenta el tiempo de reverberación T_{60}, poco a poco todos los modelos deben mostrar un SRMR por encima de la curva azul (señal reverberada). Lo anterior no ocurre en los modelos Context-MLP y LSTM, ya que se observa que están siempre por debajo o igual a la señal reverberada. Esto último da cuenta que estos modelos tienen problemas para generalizar sobre la reverberación generada mediante las RIRs simuladas.

Se puede ver en la Figura 4.8 que la arquitectura U-net con aprendizaje GAN exhibe los mejores resultados en general, ya que a partir de los 0.3s de T_{60} se encuentra por encima del resto de modelos. Este resultado muestra que esta arquitectura es capaz de dereverberar con éxito para un amplio rango de ”niveles” de reverberación. Lo anterior, sumado a que se corroboró su robustez frente al ruido lo convierte en un modelo con gran potencial en este problema de dereverberación. En el resto de modelos, se puede observar un comportamiento similar, pero por debajo de U-net GAN. En los otros 2 casos de U-net (sin aprendizaje GAN) y LS-LSTM se puede observar que suponen una mejora significativa con un tiempo de reverberación sobre 0.5s. Adicionalmente, sobre 0.3s se observa que la arquitectura propuesta LS U-net se muestra ampliamente superior a la señal reverberada y competitiva frente a U-net GAN.

4.2.4. Resultados en datos reales

En la Tabla 4.5 se muestran los resultados sobre los datos reales o retransmitidos mencionados en la metodología. La Figura 4.9 muestra los mismos resultados gráficamente. Se utiliza solo SRMR para esta parte, ya que cuando se desee evaluar los modelos sobre datos

reales, en escasas ocasiones se posee el audio limpio con el cual comparar. La única métrica mencionada en este trabajo que entrega un valor sin comparar la señal con un audio limpio es SRMR. Una métrica con estas características es denominada *Non-intrusive*.

Tabla 4.5: Evaluación de modelos mediante SRMR en datos reales

	Near	Far
Reverberant	3.99	4.36
Context-MLP	4.69	5.53
Context-LSTM	4.69	5.50
LS-LSTM	5.49	**8.16**
FD-NDLP	4.95	5.43
U-Net	4.88	5.96
LS U-Net	5.34	6.56
U-net GAN	**6.23**	7.55

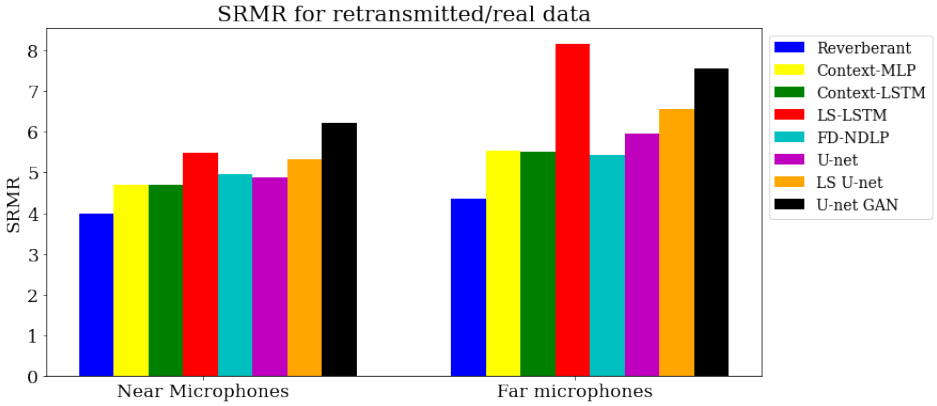


Figura 4.9: Resultados sobre datos reales para micrófonos cercanos y lejanos.

Como se puede ver, en este caso todos los modelos suponen una mejora significativa frente a la señal reverberada. El modelo con mejor desempeño según la métrica SRMR es U-net con aprendizaje GAN para micrófonos cercanos y LS-LSTM para micrófonos lejanos. La arquitectura LS-LSTM no había mostrado resultados de este nivel en datos simulados, por lo que se desprende su efectividad en este caso de datos reales. Se puede ver que la diferencia entre estos últimos para micrófonos lejanos es reducida, prácticamente insignificante.

Los 3 modelos basados en arquitectura U-net poseen un desempeño adecuado sobre la señal reverberada, por lo que tanto en datos simulados como reales suponen una mejora, lo cual da cuenta de su amplia capacidad para generalizar. Nuevamente, al igual que con datos simulados, de estos 3 modelos, U-net GAN es el con mejor desempeño y también se vuelve a observar que LS U-net tiene mejor desempeño que U-net tradicional.

Los modelos Context-MLP y Context-LSTM suponen una mejora, la cual en este caso es comparable a FD-NDLP e incluso a U-net (sin aprendizaje GAN) para micrófonos cercanos. Se desprende que el correcto desempeño de estas arquitecturas en datos reales muestra que

estos datos re-transmitidos poseen un bajo nivel de ruido y están apreciablemente reverberados. Se mostró en resultados anteriores para datos simulados, que estas arquitecturas pueden presentar problemas significativos con estos aspectos, es decir, ruido con SNR por debajo de un umbral y bajo nivel de reverberación.

El modelo FD-NDLP supone una mejora en este caso de datos reales y es esperable que siempre suponga una mejora en la métrica SRMR, pues está diseñado para dereverberación y en el peor de los casos debe dejar el espectrograma inalterado. Para micrófonos cercanos logra superar incluso a U-net (sin aprendizaje GAN). Se destaca que siendo un modelo no supervisado pueda presentar resultados de este nivel con datos reales.

4.2.5. Análisis General

A modo de contraste entre los 3 enfoques en el contexto de dereverberación presentados en el marco teórico se destacan los siguientes aspectos en base a los resultados obtenidos:

1. *Context Information*: reflejado en las arquitecturas Context-MLP y Context-LSTM se pudo corroborar que poseen una pobre capacidad de generalizar. En el único caso donde mostraron una mejora competitiva frente al resto de métodos fue en datos reales, donde el nivel de ruido es muy bajo.

2. *Late Reflections Supression*: este enfoque en este trabajo está representado por FD-NDLP y LS-LSTM. La arquitectura LS-LSTM como modelo solo se muestra competitiva con bajo nivel de ruido, pero tiene los mismos problemas que los modelos basados en *Context Information*, pese a tener un desempeño superior a estos últimos. El método no supervisado FD-NDLP se mostró robusto representando una mejora en todos los casos en cuanto a reverberación, incluso en ambientes apreciablemente ruidosos.

3. *Image to Image*: los representantes de este enfoque en este caso son las 3 arquitecturas U-net utilizadas, pese a que LS U-net es una combinación entre este enfoque y *Late Reverberation Supression*. Son modelos con una amplia capacidad de generalizar (en datos simulados y reales) y robustos frente al ruido y al nivel de reverberación (según diferentes tiempos de reverberación T_{60}). Se desprende que explorar la *totalidad del espectrograma* mediante capas convolucionales permite caracterizar de una manera más completa y efectiva la reverberación, captando patrones que se pierden al utilizar un solo *frame* (o una ventana de ellos) como *input*.

Como se ha mencionado, la arquitectura LS U-net propuesta es una combinación entre 2 de los enfoques. Cumple con ser *Image to Image*, ya que utiliza el espectrograma como *input* y lo explora a través de capas convolucionales. Sin embargo, también se cumple con *Late Reverberation Supression*, ya que su acción no es aprender un mapeo reverberado-dereverberado, sino que aprende a restar la reverberación del espectrograma. Incorporar *Late Reverberation Supression* en U-net mostró ser más efectiva que U-net tradicional, pero no en todos los casos logró superar a U-net GAN.

En todos los casos la métrica SRMR se mostró como la más efectiva para detectar la reverberación. Mediante los resultados para el método FD-NDLP se corrobora que las métricas PESQ, CD y LLR pueden mostrar falencias si se dereverbera en un ambiente apreciablemente ruidoso. El método FD-NDLP logra dereverberar para un SNR = 15dB, pero las métricas PESQ, CD y LLR no muestran una mejora de desempeño. En el caso de CD esto es esperable, puesto que los coeficientes cepstrales de la señal [2] es conocido que no son robustos al ruido.

En todos los casos, excepto el de datos reales para micrófonos lejanos (donde LS-LSTM obtuvo los mejores resultados), la arquitectura U-net con aprendizaje GAN mostró un desempeño sobresaliente sobre todo el resto de modelos. Sin embargo, se destacan las dificultades de entrenamiento para esta arquitectura, lo cual fue mencionado al comienzo de este capítulo. El aprendizaje GAN supuso una mejora importante en cuanto a SRMR (en comparación a U-net sin aprendizaje GAN).

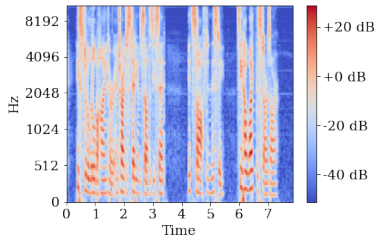
4.3. Evaluación cualitativa

En esta sección se muestran los resultados visualmente y no mediante métricas de los diferentes modelos utilizados.

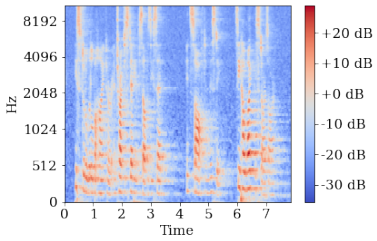
4.3.1. Visualización en datos simulados

En adelante se trabajan 2 ejemplos con el fin de ver el desempeño visual. Las Figuras 4.10 y 4.11 muestran para cada ejemplo el espectrograma limpio, el espectrograma reverberado y las respectivas salidas de los modelos.

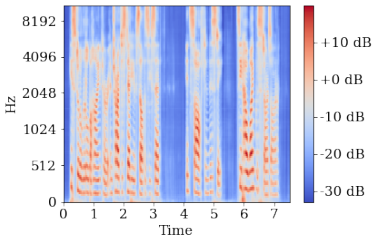
[2] descrito en el marco teórico sección 2.4 "Metricas de calidad, inteligibilidad y reverberación"



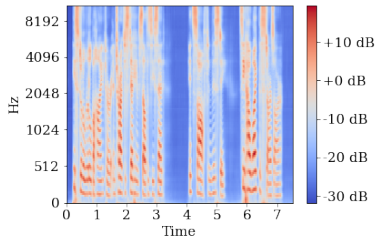
(a) Ejemplo 1 limpio.



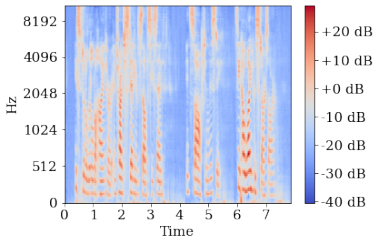
(b) Ejemplo 1 reverberado.



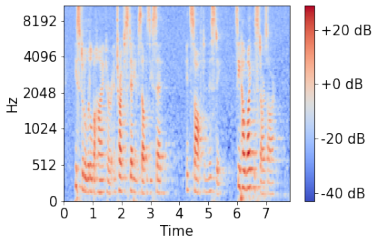
(c) Salida de Context-MLP al ejemplo 1.



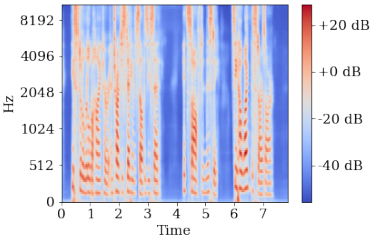
(d) Salida de Context-LSTM al ejemplo 1.



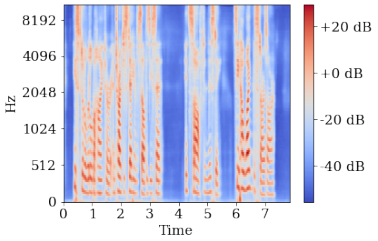
(e) Salida de LS-LSTM al ejemplo 1.



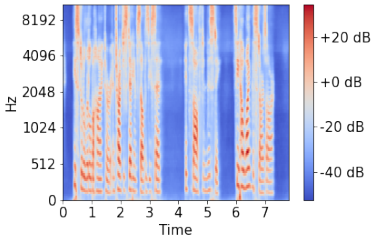
(f) Salida de FD-NDLP al ejemplo 1.



(g) Salida de U-net al ejemplo 1.

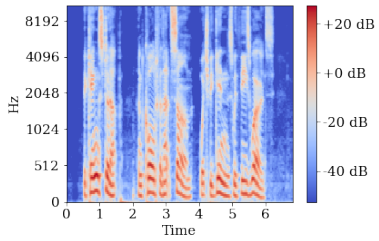


(h) Salida de LS U-net al ejemplo 1.

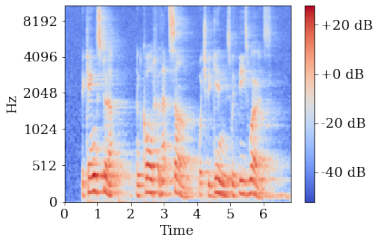


(i) Salida de U-net GAN al ejemplo 1.

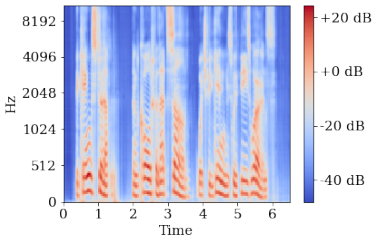
Figura 4.10: Ejemplo 1 para datos simulados y respectivas salidas de los modelos



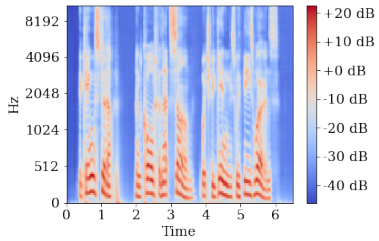
(a) Ejemplo 2 limpio.



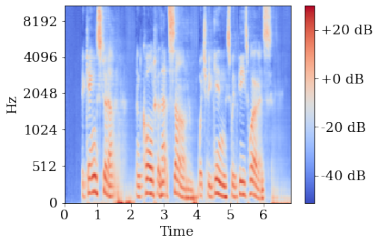
(b) Ejemplo 2 reverberado.



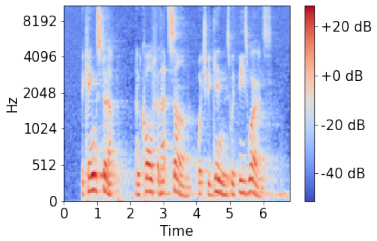
(c) Salida de Context-MLP al ejemplo 2.



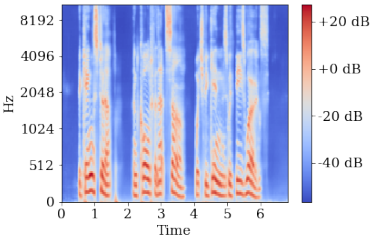
(d) Salida de Context-LSTM al ejemplo 2.



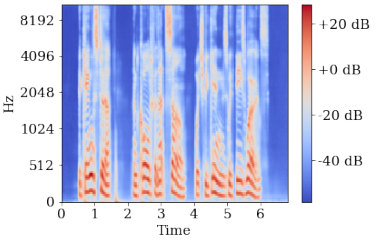
(e) Salida de LS-LSTM al ejemplo 2.



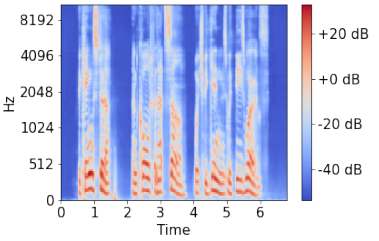
(f) Salida de FD-NDLP al ejemplo 2.



(g) Salida de U-net al ejemplo 2.



(h) Salida de LS U-net al ejemplo 2.



(i) Salida de U-net GAN al ejemplo 2.

Figura 4.11: Ejemplo 2 para datos simulados y respectivas salidas de los modelos

Notar observando ambos ejemplos en las figuras 4.10 y 4.11 (b) que hay 2 efectos apreciables visualmente. El primero es que se observa que sobre todo el espectrograma reverberado aparece un contenido de tonos de blanco (potencia espectral cercana a cero). Por ejemplo, visualmente se observa que este efecto torna zonas de azul (ver espectrograma limpio) en tonos más claros. Este efecto es principalmente ruido. En segundo lugar se puede ver que el contenido es ”esparcido” o prolongado horizontalmente, el cual es netamente reverberación como se explicó en la sección de marco teórico.

Se puede apreciar observando las Figuras 4.10 y 4.11 para los 2 ejemplos, que aparentemente todos los modelos cumplen con la tarea de dereverberar. Sin embargo, se destacan los siguientes aspectos:

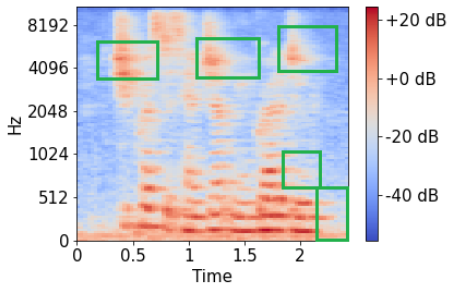
- El modelo FD-NDLP cumple con contribuir a dereverberar sobre los espectrogramas, pero no contribuye en absoluto a una mejora respecto al ruido adicionado. Se observa que en los 3 ejemplos se mantienen intactas las zonas ruidosas, las cuales ya se mencionó anteriormente como lucen visualmente.

- Los modelos Context-MLP y Context-LSTM se puede ver que visualmente también cumplen con la tarea de dereverberación. Sin embargo, se puede observar que dejan "huellas" sobre el espectrograma. En el modelo MLP se puede ver que en las zonas donde es más visible el ruido (zonas en tonos de azul), en el espectrograma de salida no se logran eliminar del todo. Como consecuencia de lo anterior, se termina por degradar esas zonas del espectrograma respecto al espectrograma limpio, en vez de contribuir a una mejora. Lo anterior, permite explicar en parte el por qué de la baja de SRMR en la evaluación cuantitativa para estos modelos.

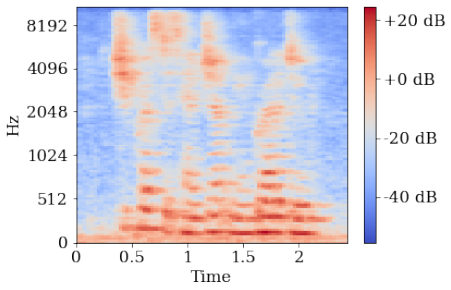
- Los modelos U-net en sus 3 variantes se muestran ampliamente superiores al resto de modelos visualmente. Se puede ver al comparar con los espectrogramas limpios que son muy similares, contribuyendo a la dereverberación independiente del nivel de ruido presente. Este es un resultado esperable, pues estos métodos exploran la totalidad del espectrograma a través de convoluciones en su aprendizaje.

4.3.2. Visualización en datos reales

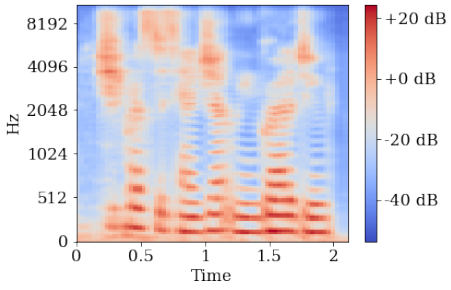
En esta sección se muestran 2 ejemplos de espectrogramas utilizados anteriormente en la evaluación cuantitativa en datos reales. Se marcan en cuadros de color verde las zonas en ambos espectrogramas que lucen apreciablemente reverberados (ya que en el caso de datos reales son menos notorias visualmente). En las Figuras 4.12 y 4.13 se muestran los ejemplos reverberados y las salidas de todos los modelos.



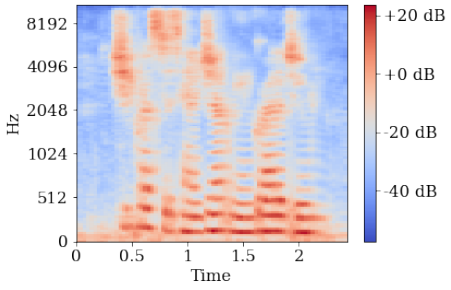
(a) Ejemplo 1 reverberado.



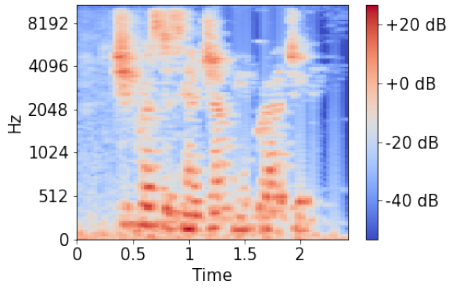
(b) Salida de Context-MLP al ejemplo 1.



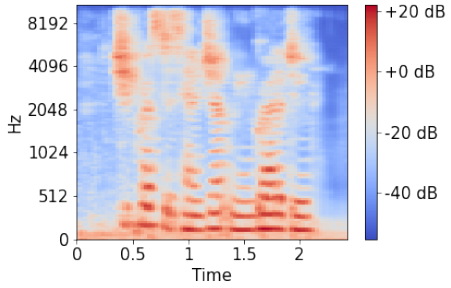
(c) Salida de Context-LSTM al ejemplo 1.



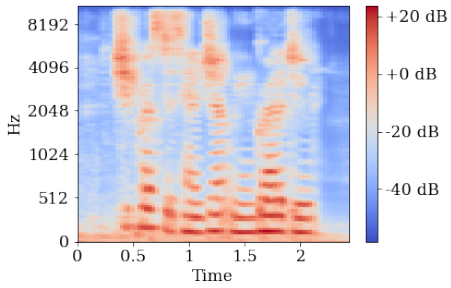
(d) Salida de LS-LSTM al ejemplo 1.



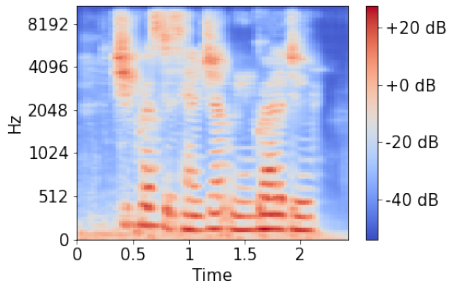
(e) Salida de FD-NDLP al ejemplo 1.



(f) Salida de U-net al ejemplo 1.

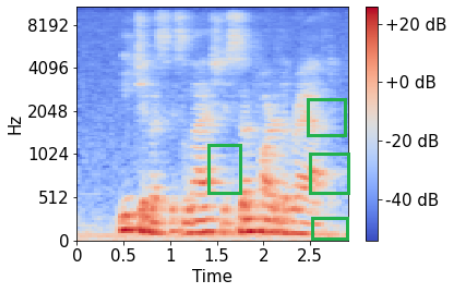


(g) Salida de LS U-net al ejemplo 1.

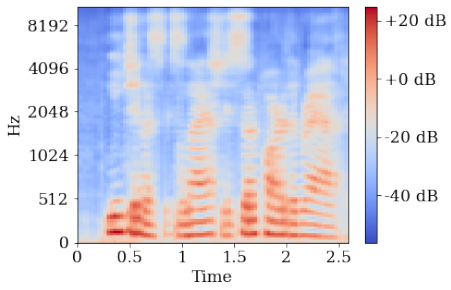


(h) Salida de U-net GAN al ejemplo 1.

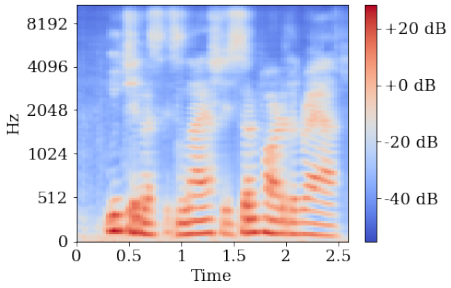
Figura 4.12: Ejemplo 1 para datos reales y respectivas salidas de los modelos



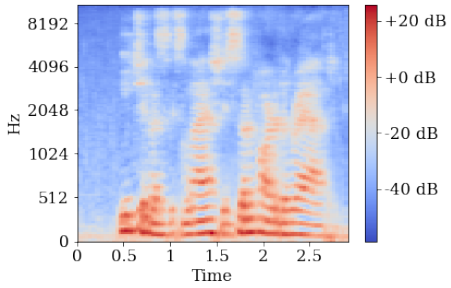
(a) Ejemplo 1 reverberado.



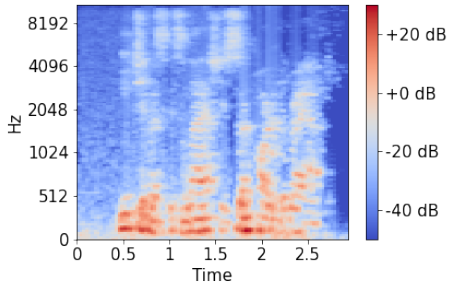
(b) Salida de Context-MLP al ejemplo 1.



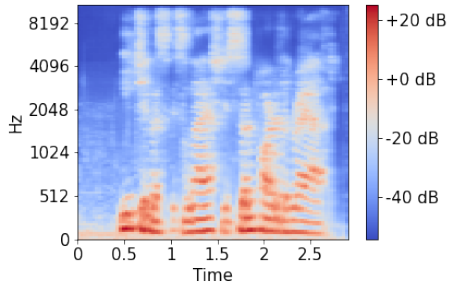
(c) Salida de Context-LSTM al ejemplo 1.



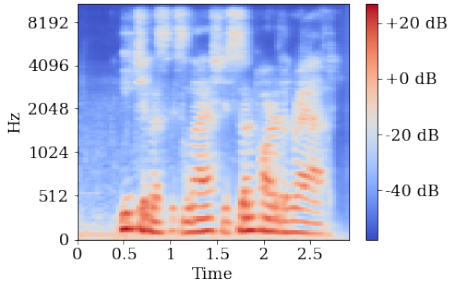
(d) Salida de LS-LSTM al ejemplo 1.



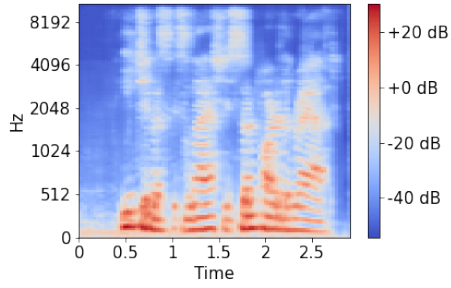
(e) Salida de FD-NDLP al ejemplo 1.



(f) Salida de U-net al ejemplo 1.



(g) Salida de LS U-net al ejemplo 1.



(h) Salida de U-net GAN al ejemplo 1.

Figura 4.13: Ejemplo 2 para datos reales y respectivas salidas de los modelos

Se puede observar en este caso que todos los modelos presentan espectrogramas dereverberados de salida con menos diferencias entre ellos, en comparación a los ejemplos trabajados para datos simulados. Esto último es consistente con los resultados cuantitativos obtenidos sobre datos reales, ya que en dichos resultados todos los modelos contribuyeron significativamente a una mejora.

Se puede ver que los espectrogramas reverberados en este caso de datos reales presentan una reverberación apreciable, pero no es exageradamente vistosa como en el caso de datos simulados. Visualmente, las salidas de FD-NDLP se diferencian levemente del resto, ya que deja visualmente "huellas" (ver figura 4.12 (e) las líneas verticales intercaladas en tonos de color azul-blanco). Sin embargo, pese a lo anterior, las salidas de FD-NDLP se aprecian visualmente dereverberadas.

Visualmente los modelos basados en MLP y LSTM tienen salidas que lucen muy similares y los modelos basados en U-net solo se diferencian del resto en que se eliminan parte de las zonas ruidosas (zonas en tonos de blanco que se tornan azul intenso). Esto último se observa pese a que el ruido en este caso no es apreciablemente alto (ya que los datos retransmitidos se graban profesionalmente evitando el ruido).

Capítulo 5

Conclusiones

Se concluye un cumplimiento del objetivo general planteado para este trabajo, ya que se logran evaluar de forma cualitativa y cuantitativa diferentes enfoques de abordar el problema de dereverberación. La evaluación cuantitativa se realizó mediante 5 métricas, pero la métrica SRMR resultó en la que dio mejores resultados detectando la reverberación en diferentes contextos. Cualitativamente los métodos basados en arquitectura U-net sobresalen debido a que contribuyen a *denoising* además de la dereverberación. Los modelos con *Context Information* cualitativamente se muestran inferiores principalmente debido al ruido.

Los modelos basados en arquitectura U-net se concluye que son los más completos en cuanto a robustez y capacidad de generalizar. Son ampliamente robustos al ruido y tienen una alta capacidad de generalizar frente a diferentes niveles de reverberación y también frente a datos simulados como reales. La variante propuesta de U-net, la cual utiliza *Late Reverberation Supression* se mostró con mejores resultados que la arquitectura U-net tradicional, pero sin superar a U-net con aprendizaje GAN. El aprendizaje GAN se concluye que utilizarlo puede otorgar ventajas en el problema de dereverberación. Se pudo corroborar bajo diferentes niveles de ruido y de reverberación que el aprendizaje GAN si supone una mejora según la métrica SRMR en comparación a solamente utilizar el MSE como función de pérdida.

Los modelos que utilizan *Context Information* reportaron resultados adecuados en datos reales y en datos simulados con bajo nivel de ruido solamente. Son redes neuronales que se mostraron con baja capacidad de generalizar y con alta sensibilidad al ruido. En el caso de LS-LSTM que busca suprimir las reflexiones tardías se desprende que tiene falencias muy similares a las arquitecturas con *Context Information*, pero en menor grado. Esta última arquitectura mostró los mejores resultados en datos reales para micrófonos lejanos, pero presentó problemas para generalizar sobre los datos reverberados utilizando el dataset de RIRs de MARDY, pese a utilizar un bajo nivel de ruido.

El método no supervisado FD-NDLP se concluye adecuado en todos los contextos, pese a no ser el método con mejor desempeño. En ambientes ruidosos este método dereverbera, pero deja el ruido intacto. Este resultado permitió vislumbrar debilidades en las métricas CD y LLR, puesto que no mostraron una mejora solo porque el método no elimina el ruido. Se desprenden falencias en estas métricas en el contexto de dereverberación.

La elección de una estrategia para resolver el problema de deconvolución en audio o equivalente dereverberación no es trivial y depende fuertemente del contexto donde es utilizada. En base a los resultados obtenidos en este trabajo se desprende que el enfoque *Image to Image* (arquitecturas U-net en este trabajo) es el más adecuado para utilizar en el caso de ambientes ruidosos. En el caso que no se tengan los recursos para entrenar un modelo *Image to Image* (tanto computacionales como de tiempo) puede ser de utilidad el método WPE en su variante de resolución FD-NDLP. Este método no supervisado pese a mostrarse inferior a los métodos *Image to Image* en todos los casos, no necesita de un entrenamiento como en el caso de los métodos neuronales y no mostró problemas para generalizar en ningún caso en base a los resultados obtenidos.

Es importante destacar el valor que entrega este trabajo. Se utilizaron métodos de dereverberación ya existentes y uno propuesto combinando ideas existentes en la literatura, evaluándoles exhaustivamente en diferentes contextos. Se contrastaron los diferentes enfoques utilizados con Deep Learning para el problema de deconvolución en audio, analizando sus fortalezas y debilidades. Lo anterior entrega valor, ya que usualmente en publicaciones científicas, un método de dereverberación es evaluado en contextos limitados, en un dataset conocido masivamente y comparado con una baja cantidad de otros métodos (comúnmente solo WPE).

Como trabajo futuro o extensión de este trabajo se puede realizar un análisis con métrica WER utilizando un sistema de transferencia de audio a texto y contrastarlo con el enfoque que es utilizado en esta memoria. Adicionalmente, se puede ampliar aún más el conjunto de prueba con audios de autoría propia medidos en laboratorio.

Bibliografía

[1] K.S Helfer and L.A Wilber, "Hearing Loss, Aging, and Speech Perception in Reverberation and Noise". *Journal of Speech and hearing research, vol. 33, no. March, pp. 149-150, 1990*

[2] Yan Zhao, Deliang Wang, Buye Xu y Tao Zhang, "Late Reverberation Supression using Recurrent Neural Networks with Long Short-Term Memory". *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2018.*

[3] Sandhya R. Deshpande and Mangesh S. Deshpande, "Multi-microphone speech dereverberation using spatial filtering", *2016 Conference on Advances in Signal Processing (CASP).*

[4] Ke Wang, Junbo Zhang, Sining Sun, Yujun Wang, Fei Xiang, Lei Xie, "Investigate Generative Adversarial Networks based Speech Dereverberation for Robust Speech Recognition". *School of Computer Science, Northwestern Polytechnical University, China.*

[5] Xue Feng, Yaodong Zhang and James Glass, "Speech Feature Denoising and Dereverberation via Deep Autoencoders for Noisy Reverberant Speech Recognition". *MIT computer Science and Artificial Intelligence Laboratory, Cambridge, MA, USA.*

[6] Kun Han, Yuxuan Wang, DeLiang Wang, William S. Woods, Ivo Merks and Tao Zhang, "Learning Spectral Mapping for Speech Dereverberation and Denoising". *IEEE Transactions on Audio, Speech, and Language Processing, June 2015*

[7] Xin Wang, Jun Du and Yannan Wang, "A Maximum Likelihood Aproach to Deep Neural Network Based Speech Dereverberation". *Proceedings of APSIDA Anual Summit and Conference, University of Science and Technology of China, December 2017.*

[8] Ori Ernst, Shlomo E. Chazan, Sharon Gannot and Jacob Goldberger, "Speech Dereverberation Using Fully Convolutional Networks". *Faculty of Engineering, Bar-Ilan University, 3 Apr, 2019.*

[9] Olaf Ronneberger, Philipp Fischer, Thomas Brox "U-Net: Convolutional Networks for Biomedical Image Segmentation," in *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, 2015, pp. 234-241.

[10] Daniel Michelsanti, Zheng-Hua Tan, "Conditional Generative Adversarial Networks for Speech Enhancement and Noise-Robust Speaker Verification ," in INTERSPEECH, 2017.

[11] Tan Jo Lynn, Ahmad Zuri bin Sha'ameri, "Comparison between the performance of spectrogram and multi-window spectrogram in digital modulated communication signals". *IEEE International Conference on Telecommunications and Malaysia International Conference on Communications, 2007*

[12] Jesse Engel, Cinjon Resnick, Adam Roberts, Sander Dieleman, Douglas Eck, Karen Simonyan, Mohammad Norouzi, "Neural Audio Synthesis of Musical Notes with WaveNet

Autoencoders", Apr. 2017. Available https://arxiv.org/abs/1704.01279

[13] John G. Proakis and Dimitris G. Manolakis, "Digital Signal Processing", Third Edition, pp. 63-77.

[14] Linus Lexford and Malte Johansson, "Audio representation for environmental sound using convolutional neural networks", Lunds University, Computer Science, 2018.

[15] Marvin Ritter, "Neural Network Arquitectures for Reverberated Lecture Speech Recognition", *Master Thesis, Karisruhe Institute of Technology.*

[16] P. A Naylor and N.D Gaubitch, *Speech Dereverberation.* Springer Science & Business Media, 2010.

[17] Ian Goodfellow, Yoshua Bengio and Aaron Courville, "Deep Learning", 2015.

[18] Antony W. Rix, John G. Beerends, Michael P. Hollier and Andries P. Hekstra, "Perceptual Evaluation of Speech Quality (PESQ) - A New Method for Speech Quality Assessment of Telephone Networks and Codecs", 2001.

[19] Hu and Loizou, "Evaluation of objective quality measures for speech enhancement", IEEE T-ASLP, 16(1), 229-238, 2008

[20] Falk, et al., "A non-intrusive quality and intelligibility measure of reverberant and dereverberated speech", IEEE T-ASLP, 18(7), 1766-1774, 2010

[21] S. Furui, "Digital Signal Processing, Synthesis, and Recognition, Second Edition, Revised and Expanded". New York: Marcel Dekker, 2001.

[22] J. Hansen and B. Pellom, "An effective quality evaluation protocol for speech enhancement algorithms", in *Proc. Int. Conf. Spoken Lang. Process., 1998, vol.7, pp. 1278-1281.*

[23] S. Quackenbush, T. Barnwell, and M. Clements, *Objective Measures of Speech Quality.* EngleWood Cliffs, NJ: Prentice-Hall, 1988.

[24] Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherlij Ozair, Aaron Courville and Yoshua Bengio, "Generative Adversarial Nets", Université de Montréal, 2014.

[25] Felipe Tobar, Arnaurd Robert, Jorge Silva, "Gaussian Process Deconvolution", University of Chile, Geneva University Hospitals, 2020.

[26] T. Nakatani, T. Yoshioka, K. Kinoshita, M. Miyoshi and B. Juang, "Blind speech dereverberation with multi-channel linear prediction based on short time fourier transform representation," 2008 IEEE International Conference on Acoustics, Speech and Signal Processing, Las Vegas, NV, USA, 2008, pp. 85-88, doi: 10.1109/ICASSP.2008.4517552.

[27] T. Taniguchi, A. S. Subramanian, X. Wang, D. Tran, Y. Fujita and S. Watanabe, "Generalized Weighted-Prediction-Error Dereverberation with Varying Source Priors For Reverberant Speech Recognition,"2019 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA), New Paltz, NY, USA, 2019, pp. 293-297, doi: 10.1109/WASPAA.2019.8937270.

[28] T. Nakatani, T. Yoshioka, K. Kinoshita, M. Miyoshi and B. Juang, "Speech Dereverberation Based on Variance-Normalized Delayed Linear Prediction," in IEEE Transactions on Audio, Speech, and Language Processing, vol. 18, no. 7, pp. 1717-1731, Sept. 2010, doi:

10.1109/TASL.2010.2052251.

[29] Philipos C. Loizou, "Speech Enhancement: Theory and Practice, Second Edition", 2013.

[30] Jorge Wuth, Richard M. Stern and Néstor Becerra Yoma, "Non causal deep learning based dereverberation". Speech Processing and Transmission Laboratory, Electrical Engineering Department University of Chile, Department of Electrical and Computer Engineering and Language Technologies Institute.

[31] M. Slaney, "An efficient implementation of the Patterson-Holdsworth auditory filterbank", Apple Computer, 1993, Tech. Rep.

[32] Vassil Panayotov, Guoguo Chen, Daniel Povey and Sanjeev Khudanpur, "LibriSpeech: an ASR corpus based on public domain audio books", ICASSP 2015.

[33] I. Szöke, M. Skácel, L. Mošner, J. Paliesek and J. Černocký, "Building and evaluation of a real room impulse response dataset", in IEEE Journal of Selected Topics in Signal Processing, vol. 13, no. 4, pp. 863-876, Aug. 2019, doi: 10.1109/JSTSP.2019.2917582.

[34] R. Stewart and M. Sandler, "Database of omnidirectional and B-format room impulse responses," 2010 IEEE International Conference on Acoustics, Speech and Signal Processing, Dallas, TX, USA, 2010, pp. 165-168, doi: 10.1109/ICASSP.2010.5496083.

[35] J. Y. C. Wen, N. D Gaubitch, E. a. P. Habets, T. Myatt, and P. a. Naylor, *"Evaluation of Speech Dereverberation Algorithms using the MARDY Database," Proc. Intl. Workshop Acoust. Echo Noise Control (IWAENC)*, pp. 12-15, 2006.

Anexos

Anexo A

Modelación con Sistemas LTI

Los sistemas lineales e invariantes en el tiempo (LTI) permiten modelar una sala y con ello la respuesta acústica de esta. Los sistemas LTI están caracterizados por la respuesta al impulso. A continuación se presentan sus propiedades hasta llegar a la expresión matemática de la convolución.

Parte de la notación y planteamiento es tomado de "Digital Signal Processing", John G.Proakis y Dimitris G. Manolakis [13]

A.1. Linealidad

Sea $T[\cdot]$ que define un sistema en tiempo discreto. Se dice que T es lineal si para 2 señales discretas $x_1(n)$, $x_2(n)$ y a, b constantes arbitrarias, se cumple que

$$T[ax_1(n) + bx_2(n)] = aT[x_1(n)] + bT[x_2(n)]. \tag{A.1}$$

A.2. Invarianza en el tiempo

Sea $T_k[\cdot]$ el operador de desplazamiento en el tiempo, este es operador es tal que para una señal discreta $x(n)$ su salida $y_k(n)$ está dada por la siguiente expresión:

$$y_k(n) = T_k[x(n)] = x(n-k). \tag{A.2}$$

Sea $T[\cdot]$ otro operador discreto e $y(n) = T[x(n)]$ la salida del sistema ante $x(n)$ como entrada. El sistema compuesto entre T y T_k está definido por $T_k[T[\cdot]]$ y $T[T_k[\cdot]]$ que no tienen por qué ser iguales en general. Esta composición permite escribir las siguientes relaciones:

$$T_k[T[x(n)]] = T_k[y(n)] = y(n-k), \tag{A.3}$$

$$T[T_k[x(n)]] = T[x(n-k)]. \tag{A.4}$$

Si $T[x(n-k)] = y(n-k)$, entonces las 2 ecuaciones anteriores se pueden igualar y entonces $T_k[T[\cdot]] = T[T_k[\cdot]]$. Lo anterior significa que $T_k[\cdot]$ y $T[\cdot]$ conmutan y en tal caso el sistema definido por $T[\cdot]$ se dice que es invariante en el tiempo. Si $T[\cdot]$ además es lineal, el sistema es entonces lineal e invariante en el tiempo (LTI).

A.3. Convolución en sistemas LTI

Sea una señal arbitraria discreta $x(n)$, sea $T[\cdot]$ un sistema lineal e invariante en el tiempo y sea $\delta(n)$ una señal de impulso, entonces la respuesta de este sistema a la entrada $x(n)$ está dada por la siguiente expresión:

$$T[x[n]] = T[\sum_{k=-\infty}^{\infty} x(k)\delta(n-k)] = \sum_{k=-\infty}^{\infty} x(k)T[\delta(n-k)]. \tag{A.5}$$

Si se denota $h(n) = T[\delta(n)]$ a la respuesta al impulso del sistema, entonces la respuesta $y(n)$ ante la entrada $x(n)$ se puede reescribir de la manera siguiente:

$$y(n) = \sum_{k=-\infty}^{\infty} x(k)h(n-k) = (x * h)(n). \tag{A.6}$$

El símbolo $*$ denota convolución. Esta ecuación permite ver que los sistemas LTI están **completamente caracterizados por la respuesta al impulso $h(n)$ del sistema**. En audio, esto es de utilidad, pues la respuesta auditiva que entrega una sala u otro recinto se modela como un sistema LTI. Esto significa que una señal auditiva medida está convolucionada con la respuesta al impulso de la sala.

Anexo B

TD-NDLP

Con el problema NDLP planteado, para poder estimar los coeficientes necesarios $\hat{\bar{c}}^{(m)}$ dominio temporal se plantea una verosimilitud L, la cual está descrita mediante la siguiente ecuación:

$$L(\theta) = \sum_{t=1}^{\mathcal{T}} \log(p(d_t^{(1)} = x_t^{(1)} - (\bar{c}^{(1)})^T \bar{x}_{t-D}; \theta)), \tag{B.1}$$

donde θ son los parámetros que se desean estimar para $\mathcal{T}$ observaciones de $x_t^{(1)}$. Para poder maximizar la verosimilitud es necesario plantear una distribución $p(d_t^{(1)})$, para lo cual se utiliza la señal *source* s_t como un proceso gaussiano con media cero y matriz de covarianza $R_t = \mathbb{E}[\bar{s}_t \bar{s}_t]$. Las observaciones se denotan convenientemente como $\bar{s}_t = [s_{t-L_f/2+1}, ..., s_{t+L_f/2}]^T$ con L_f la longitud del vector. Bajo este planteamiento en la señal *source*, la señal deseada $d_t^{(1)}$ también es un proceso gaussiano, por lo cual se plantea una densidad marginal pdf dada por:

$$p(d_t^{(1)}) = \mathcal{N}(d_t^{(1)}; 0, \sigma_t^2), \tag{B.2}$$

con $\sigma_t^2 = \mathbb{E}[|d_t^{(1)}|^2] \approx \frac{1}{L_f} \sum_{t'=t-L_f/2+1}^{t+L_f/2} |d_t^{(1)}|^2$. Los parámetros a encontrar son $\theta = [\bar{c}^{(1)}, \bar{\sigma}]$ con $\bar{\sigma} = [\sigma_1^2, ..., \sigma_{\mathcal{T}}^2]$

La distribución introducida permite reescribir la verosimilitud como sigue:

$$\begin{aligned} L(\theta) &= \sum_{t=1}^{\mathcal{T}} \log(p(d_t^{(1)} = x_t^{(1)} - (\bar{c}^{(1)})^T \bar{x}_{t-D}; \theta)) \\ &= -\frac{1}{2} \sum_{t=1}^{\mathcal{T}} \frac{|x_t^{(1)} - (\bar{c}^{(1)})^T \bar{x}_{t-D}|^2}{\sigma_t^2} - \frac{1}{2} \sum_{t=1}^{\mathcal{T}} \log(\sigma_t^2) + \text{const.} \end{aligned} \tag{B.3}$$

Finalmente, para encontrar los parámetros $\theta = [\bar{c}^{(1)}, \bar{\sigma}^2]$ maximizando la verosimilitud se utiliza el algoritmo siguiente:

Algorithm 2: Algoritmo para NDLP en dominio temporal o TD-NDLP.

Inicializar $\hat{\sigma}_t^2 = \max\{\frac{1}{L_f}\sum_{t'=t-L_f/2+1}^{t+L_f/2} |x_t^{(1)}|^2, \epsilon\}$ con $\epsilon > 0$ un hiper-parámetro ajustable (en caso que $\hat{\sigma}_t^2$ sea menor que un umbral permitido)

Repetir hasta convergencia:

- Actualizar $\hat{\bar{c}}^{(1)}$ como $\hat{\bar{c}}^{(1)} = \hat{\Psi}^{+}\hat{\phi}$, donde $\hat{\Psi} = \sum_{t=1}^{\mathcal{T}} \frac{\bar{x}_{t-D}\bar{x}_{t-D}^T}{\hat{\sigma}_t^2}$ y $\hat{\phi} = \sum_{t=1}^{\mathcal{T}} \frac{\bar{x}_{t-D}x_t^{(1)}}{\hat{\sigma}_t}$
- Actualizar $\hat{d}_t^{(1)}$ como $\hat{d}_t^{(1)} = x_t^{(1)} - (\hat{\bar{c}}^{(1)})^T\bar{x}_{t-D}$
- Actualizar $\hat{\sigma}_t$ como $\hat{\sigma}_t^2 = \max\{\frac{1}{L_f}\sum_{t'=t-L_f/2+1}^{t+L_f/2} |d_t^{(1)}|^2, \epsilon\}$, $\epsilon > 0$

En $\hat{\Psi}^{+}$, el símbolo + denota la pseudo-inversa de Moore-Penrose. Dicha inversa para una matriz arbitraria A está definida como $A^{+} = (A^TA)^{-1}A^T$