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

CANCELACIÓN DE RUIDO Y REVERBERACIÓN PARA RECONOCIMIENTO DE VOZ EN INTERACCIÓN HUMANO-ROBOT

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL ELÉCTRICO

PEDRO JOSÉ CORREA ÁLVAREZ

PROFESOR GUÍA:
NÉSTOR BECERRA YOMA

MIEMBROS DE LA COMISIÓN:
JORGE SILVA SÁNCHEZ
RODRIGO MAHU SINCLAIR

SANTIAGO DE CHILE
2020

RESUMEN DE LA MEMORIA PARA OPTAR
AL TÍTULO DE INGENIERO CIVIL ELÉCTRICO
POR: PEDRO JOSÉ CORREA ÁLVAREZ
FECHA: 2020
PROF. GUÍA: NÉSTOR BECERRA YOMA

CANCELACIÓN DE RUIDO Y REVERBERACIÓN PARA RECONOCIMIENTO DE VOZ EN INTERACCIÓN HUMANO-ROBOT

Actualmente las tecnologías de reconocimiento automático de voz tienen un rol protagónico en el desarrollo de plataformas para la interacción entre personas y sistemas robóticos. Considerando que el lenguaje hablado es la principal forma de comunicación entre los seres humanos, el desarrollo de estas plataformas ha apuntado a imitar dicha forma de comunicación, y se ha traducido en populares aplicaciones como *Siri* o *Google Now*. De la misma manera, en el área de interacción humano robot el reconocimiento automático de voz es crucial, y en este contexto se suele contar con una serie de sensores. Los robots normalmente están equipados con arreglos de micrófonos, además de cámaras de video y térmicas, radares, sonares, entre otros. Esto entrega una amplia gama de posibilidades para mejorar el desempeño de las distintas tecnologías que han sido implementadas en ellos, particularmente las de reconocimiento de voz.

En esta Memoria de Título se ha trabajado sobre señales distorsionadas con ruido y reverberación, simulando escenarios típicos de interacción humano robot. Con el fin de mejorar el desempeño del reconocimiento automático de voz, se han implementado redes neuronales artificiales y otras técnicas de procesamiento de audio que permiten reducir el efecto del ruido y la reverberación. Para esto, se utilizó *Tensorflow*, una librería de libre acceso para *Python* que permite la programación de redes neuronales artificiales. Hecho esto, se comparan los sistemas implementados en base a su desempeño en un sistema de reconocimiento automático de voz.

Por último, se ha concluido en base a los resultados obtenidos en las distintas etapas de la investigación. Para el caso de la reverberación, se compara el desempeño de una red neuronal LSTM y el método WPE, siendo este último el que da mejores resultados. Por otra parte, para la reducción de ruido se implementó una red neuronal *feedforward*, y se probaron distintas formas de normalización sobre los datos. Se observó que ciertas normalizaciones permiten mejoras considerables en el desempeño del reconocedor de voz.

A mi abuelo Horacio Álvarez, con cuyo ejemplo
se forjó el pensamiento crítico de sus nietos.

Agradecimientos

Este documento es el resultado de un largo proceso y fue posible gracias al apoyo y cariño de muchas personas. Quisiera agradecer en primer lugar a mis papás, quienes me dieron la fuerza para seguir estudiando incluso en los momentos más duros de la carrera, pero sin dejar nunca de incentivar mi pasión más profunda: la música.

También quiero darles las gracias a mis amigos y amigas, con quienes en más de alguna ocasión desahogué mi estrés, mis crisis vocacionales o incluso mis penas de amor. A lo largo de estos años ustedes se han transformado en lo que le da sentido a lo que hago y lo que soy. Sin duda este proceso universitario no habría sido posible sin su cariño.

Agradezco además a los compañeros y compañeras que han mantenido en pie a los grupos organizados de la facultad, particularmente el Grupo de Música de Ingeniería y el Centro de Desarrollo Estudiantil, de los cuales fui parte durante los primeros años de la carrera, y que contribuyeron al desarrollo integral de mis habilidades. En este mismo punto, agradezco al departamento de humanidades de la facultad, especialmente al profesor Ricardo Loebell por ayudarme a tomar conciencia de mi función como ingeniero en la sociedad.

Por último, agradezco enormemente al profesor Néstor y a todo el equipo del Laboratorio de Procesamiento y Transmisión de la Voz, con quienes fue posible llevar a cabo este trabajo. Gracias por brindarme la oportunidad de conocer y ser parte del mundo de la investigación científica, con todo el aprendizaje que esto conlleva.

Tabla de Contenido

Índice de Tablas

Índice de Ilustraciones

Capítulo 1

Introducción

1.1. Motivación

Los algoritmos de reconocimiento de voz hoy en día son ampliamente utilizados en diversos contextos, entre ellos, interacción humano-robot (*Human-Robot Interaction* en inglés, HRI). Si bien estos métodos son cada vez más robustos, hay aspectos específicos en los que todavía hay mucho por mejorar.

Por otra parte, las redes neuronales artificiales han sido calificadas como "aproximadores universales", y como tales, presentan una gran flexibilidad para resolver problemas de diversa índole. Particularmente en procesamiento de voz, las redes neuronales constituyen una buena solución a problemas como el ruido y la reverberación.

1.2. Definición del problema

Uno de los ejes centrales de HRI es el reconocimiento automático de voz (*Automatic Speech Recognition* en inglés, ASR). El desempeño de un reconocedor de voz se ve afectado principalmente por dos fenómenos acústicos: el ruido aditivo y la reverberación. Si alguno de estos factores está presente, la decodificación de la señal de voz se dificulta.

La reverberación ocurre en cualquier espacio cerrado, y es causado por la reflexión de las ondas sonoras en las paredes o superficies, lo que se traduce en una serie de ondas similares a la original, pero desfasadas en el tiempo. La magnitud de la reverberación viene dada principalmente por la forma de la sala y los materiales que la componen, entre otros factores.

El ruido y la reverberación, pueden tratarse de múltiples formas. En cualquier caso, es conveniente conocer las posiciones del locutor y el ruido. El problema de reconocimiento y seguimiento de objetos (particularmente rostros) está relativamente resuelto, lo que permitiría

estimar, por ejemplo, la posición de una persona que está hablando. Usando esta información y un arreglo de micrófonos, se pueden implementar técnicas como *beamforming*, que permiten reducir la intensidad del ruido y la reverberación al concentrar la ganancia del arreglo en la dirección deseada.

Sin embargo, las señales obtenidas luego de este proceso aun presentan cierto nivel de ruido y de reverberación, dejando espacio para la aplicación de nuevas técnicas de procesamiento de audio.

1.3. Objetivos

1.3.1. Objetivos generales

En términos generales, el objetivo de este trabajo consiste en mejorar el desempeño del reconocimiento automático de voz en contextos de interacción humano robot. Para esto se deben estudiar e implementar técnicas de procesamiento de audio que permitan reducir los efectos del ruido y la reverberación.

1.3.2. Objetivos Específicos

Dentro de los objetivos específicos se encuentran:

- Investigar sobre la técnica de *beamforming* para dirigir el patrón polar de un arreglo de micrófonos a la posición deseada.
- Usar *beamforming* para enfatizar la señal de voz a partir de la posición del locutor. Para este trabajo se asume que dicha posición es obtenida mediante el procesamiento de la información visual, y se considera conocida.
- Considerando el ruido y la reverberación, describir matemáticamente las señales obtenidas del *beamforming* para generar bases de datos.
- Utilizar técnicas de aprendizaje de máquinas (*Machine Learning* en inglés) para reducir efecto del canal (reverberación) en señales generadas y cancelar el ruido aditivo.
- Con todo lo anterior, mejorar el desempeño del ASR, es decir disminuir la tasa de error de palabras (*Word Error Rate* en inglés, WER).

1.4. Alcances

En este trabajo se usará un robot PR2 (Willow Garage) equipado con una Kinect (Microsoft) para grabar una base de datos que luego se utilizará para sintetizar señales con las que se implementarán y se probarán los sistemas propuestos.

El procesamiento de la información visual escapa de los alcances de este trabajo, por lo que se asumirá que se cuenta con estos datos ya procesados. Es decir, aquellos datos que

puedan ser obtenidos a partir de las cámaras del robot, se asumirán conocidos.

Una vez generados los datos, se estudiará la técnica *beamforming* acústico y se aplicará a las señales de audio grabadas por el arreglo de micrófonos de la Kinect, con el propósito de obtener una señal menos ruidosa que la señal directa de los micrófonos.

Finalmente, se aplicarán técnicas de *Machine Learning* para mitigar los efectos de la reverberación y el ruido en el ASR. Esto se traduce en los resultados señalados a continuación.

1.5. Resultados esperados

Con este trabajo se pretende investigar e implementar métodos que permitan integrar información visual y acústica con el fin de reducir significativamente en el ASR los errores causados por el ruido y la reverberación.

1.6. Estructura de la memoria

Este documento se divide en cinco capítulos, siendo la introducción el capítulo 1. A continuación se describen brevemente los cuatro capítulos restantes.

En el capítulo 2 se realiza una descripción general del ASR y el estado del arte en que se encuentra. Luego se revisa el fenómeno de la reverberación y los principales métodos existentes que la reducen. A continuación se presenta el estado del arte de las técnicas de procesamiento de imágenes, particularmente en el área de seguimiento de fuente, que resulta de mayor interés para los propósitos de esta investigación. Luego de esto, se hace una descripción de la técnica de *beamforming* acústico, y los métodos más comunes para llevarla a cabo. Por último, se explica en qué consisten las redes neuronales artificiales, cómo se modelan, y sus principales arquitecturas.

A continuación, en el capítulo 3, se da a conocer la metodología de trabajo que se utilizó a lo largo del trabajo, es decir, la modelación del problema, los métodos que se implementaron para solucionarlo, y las métricas con las que finalmente se evaluaron dichos métodos.

Más tarde, en el capítulo 4, se entregan los resultados obtenidos luego de implementar las técnicas y aplicando las métricas definidas en el capítulo anterior. Además se realiza un análisis de lo obtenido y se muestran algunos resultados adicionales.

Finalmente, en el capítulo 5, se exponen las conclusiones de el trabajo, tomando en cuenta los objetivos que fueron planteados cuando este comenzó.

Capítulo 2

Marco Teórico

2.1. Reconocimiento automático de voz

2.1.1. Antecedentes

El reconocimiento automático de voz (*Automatic Speech Recognition* en inglés, ASR) corresponde al proceso de transcripción de la voz humana a palabras y la tecnología asociada a esto, tal como se muestra en la figura 2.1.

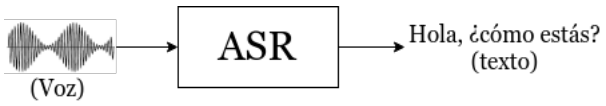

Figura 2.1: Diagrama del ASR común.

Por otra parte, de acuerdo al teorema de Bayes, el problema del ASR se puede formular de la siguiente manera [99]:

$$\hat{W} = \underset{X}{\operatorname{argmax}}\{p(W|X)\} = \underset{X}{\operatorname{argmax}}\{p(X|W) \cdot p(W)\} \tag{2.1}$$

donde $\hat{W}$ es la salida del ASR, W es la secuencia óptima de etiquetas (palabras o fonemas); X es la secuencia observada de voz que representa una sentencia o frase dada (en adelante se utilizará su traducción al inglés, *utterance*); $p(W)$ denota el modelo de lenguaje que describe las probabilidades de combinaciones de palabras; y $p(X|W)$ representa el modelo acústico. De esta manera, la tarea de un sistema de ASR es encontrar (mediante un proceso llamado decodificación, en inglés *decoding*, realizado por el algoritmo de Viterbi [40]) la secuencia de etiquetas W más probable dada la observación de un vector característico (comúnmente llamado *feature vector*) que corresponde a la utterance. El modelo de lenguaje puede ser representado con [41]: modelos estadísticos, gramática libre de contexto probabilística (en inglés *Stochastic context-free grammar*, SCFG), o modelos estocásticos de estado finito. En el caso de los modelos estadísticos, que son ampliamente usados en investigación, la probabilidad

a priori de una secuencia de palabras $W = w_1, ..., w_l$ en la ecuación 2.1 puede ser aproximada con *N-gramas* (secuencia de N fonemas):

$$p(W) = \prod_{l=1}^{L} p(w_l | w_{l-1}, w_{l-2}, ..., w_{l-N+1}) \tag{2.2}$$

donde N es típicamente un entero entre 2 y 4. El modelo de lenguaje define la probabilidad de transición desde un *N-grama* a la siguiente palabra para guiar la búsqueda de una interpretación de la entrada acústica. Adicionalmente, el tamaño del vocabulario y la perplejidad (informática) [8] son factores críticos para el desempeño del ASR. La perplejidad mide la incertidumbre de las palabras que pueden seguir a un *N-grama* dado. Un modelo de lenguaje con baja perplejidad definido por un contexto o una tarea dada limitará la decodificación y se desempeñará mejor que uno de alta perplejidad.

La modelación acústica define representaciones estadísticas para la secuencia de *feature vectors* acústicos X obtenidos de la forma de onda de la señal de voz. Las *utterances* son divididas en ventanas de 20 a 30 [ms] con traslape, por ejemplo, del 50 %. Usualmente se obtiene el set de *features* aplicando a cada ventana la transformada rápida de Fourier (*fast Fourier transform* en inglés, FFT) [40] [9] [15]. Los coeficientes de velocidad y aceleración (llamados coeficientes *delta* y *delta-delta*) también son típicamente usados, y el *feature vector* final se compone de las características estadísticas junto a los coeficientes recién mencionados. [20]. Además se puede utilizar normalización de media y varianza en los coeficientes.

2.1.2. Estado del arte

Hasta hace pocos años la mayoría de las sistemas de reconocimiento de voz usaban modelos ocultos de Markov (en inglés *hidden Markov models*, HMMs) para tratar la variabilidad temporal de la voz, y modelos gaussianos mixtos (en inglés *Gaussian mixture models*, GMMs) para representar $p(X|W)$. Dado un set de *feature vectors* de voz $X = \{x_t\}_{t=1}^{T}$, la función de densidad de probabilidad de observación del *feature vector* x_t en el estado s_i se expresa como [40]:

$$p(x_t | s_\mathrm{i}) = \sum_{m=1}^{M} c_{\mathrm{i},m} \cdot N(x_t; \mu_{\mathrm{i},m}, \sum_{\mathrm{i},m}) \tag{2.3}$$

donde $c_{\mathrm{i},m}$, $\mu_{\mathrm{i}.m}$ y $\sum_{\mathrm{i},m}$ corresponden a los pesos de mezcla, vectores de medias, y matrices de covarianzas respectivamente, para M componentes de mezcla Gaussianos. En los últimos años, las redes neuronales artificiales (como por ejemplo las *Deep Neural Networks*, DNNs) han demostrado un desempeño significativamente mejor que los modelos basados en GMMs. En un sistema DNN-HMM, la DNN entrega una pseudo-verosimilitud logarítmica definida como:

$$\log[p(x_t | s_j)] = \log[p(s_j | x_t)] - \log[p(s_j)] \tag{2.4}$$

donde s_j denota uno de los estados; y las probabilidades *a priori* $\log(s_j)$ pueden ser entrenadas usando los alineamientos de estado obtenidos con la base de datos de entrenamiento. Finalmente, el arreglo de palabras decodificado queda determinado por:

$$\hat{W} = \underset{W}{\operatorname{argmax}}\{\log[p(X|W)] + \lambda \cdot \log[p(W)]\} \tag{2.5}$$

donde probabilidad del modelo acústico $p(X|W)$ depende de la función de pseudo-verosimilitud logarítmica $\log[p(s)]$ entregada por la DNN. Esta es la constante empleada para balancear los *scores* del modelo acústico y el modelo de lenguaje [4]. Los resultados reportados en [32] muestran que el ASR con DNN y HMM permite una reducción de la tasa de error de palabras (*Word Error Rate* en inglés, WER) del 32 % relativo, al compararlo con el sistema GMM-HMM común en el *Switchboard task* [25]. Sin embargo, el entrenamiento de una DNN no es una tarea sencilla. La función objetivo puede ser altamente no-convexa y el algoritmo de entrenamiento puede converger fácilmente a un mínimo local sub-óptimo. Además, las redes neuronales artificiales necesitan una mayor cantidad de datos de entrenamiento que los sistemas GMM-HMM [86]. Cabe mencionar que los sistemas de ASR basados en redes neuronales que han sido publicados emplean al menos miles de horas de datos de voz para ser entrenados [54] [85] [95]. Otras arquitecturas de redes neuronales artificiales han sido también aplicadas al ASR: LSTM [38]; CNN [1]; y RNN [27]. Los resultados obtenidos usando sistemas DNN-HMM son competitivos al compararlos con los resultados de otras arquitecturas [27] [88] [56] [83] [61] [100]. En algunos casos, los sistemas superan el desempeño de las DNNs, LSTMs, o CNNs empleando combinaciones arquitecturas como *very deep* CNN [76] o fCNN [66]. Sin embargo, al aumentar el número de parámetros de una red neuronal artificial, se requiere una cantidad mayor de datos para su entrenamiento.

Al usar las mismas condiciones para los datos de entrenamiento y los de prueba, el ASR presenta una gran ganancia en desempeño. Por el contrario, los modelos presentarán dificultades al reconocer datos de prueba si es que estos difieren de los datos de entrenamiento. Por esta razón, la robustez al ruido de un sistema con redes neuronales artificiales se puede lograr usando un entrenamiento con condiciones múltiples. Por ejemplo, una DNN entrenada con distintos tipos de ruido y niveles de SNR puede llevar a mejoras de alta precisión en aplicaciones reales [93].

Word Error Rate (WER)

En este trabajo se pretende mejorar el desempeño de un ASR mediante un procesamiento previo de las señales de audio. Por lo tanto, debemos utilizar algún indicador que permita evaluar dicho desempeño. En general, el indicador más utilizado para esto es conocido como *Word Error Rate* (en adelante, WER). Este se define como se muestra en la ecuación 2.6, donde S representa las palabras sustituidas por palabras incorrectas, D corresponde a las palabras omitidas e I las palabras insertadas incorrectamente. N corresponde al total de palabras en la referencia.

$$WER = \frac{S + D + I}{N} \tag{2.6}$$

Por ejemplo, si la referencia es "la casa de la esquina es azul", y el ASR entrega "la cama de la esquina azul oscuro", la palabra "casa" fue *sustituida* por "cama", la palabra "es" fue *omitida* y la palabra "oscuro" fue *insertada*. Es decir, $S = 1$, $D = 1$ e $I = 1$. El total de palabras es $N = 7$, por lo tanto la *Word Error Rate* es de $WER = (1 + 1 + 1)/7 = 3/7 \approx 42,86\,\%$.

2.1.3. Principales dificultades del ASR

Las señales de voz suelen ir acompañadas de distintas formas de ruido que dificultan su reconocimiento, lo que se traduce en un peor desempeño, y por lo tanto un aumento en los errores durante el proceso de *decoding* (es decir, una WER mayor). Los dos factores que tienen la mayor influencia sobre este desempeño son la reverberación y el ruido aditivo.

La primera se refiere al fenómeno dado por la reflexión de las ondas acústicas en las distintas superficies que rodean a la fuente de voz. Estas reflexiones se traducen en una cierta permanencia de la ondas sonoras luego de que la fuente ha dejado de emitirlas.

La segunda se refiere a todas aquellas señales acústicas que son emitidas por una fuente distinta a la de interés y que se suman a la señal de voz recibida.

En las siguientes secciones se describirán mayor detalle ambos factores.

2.2. Reverberación

2.2.1. Modelación

Las señales de voz reverberadas suelen modelarse como la convolución de la señal limpia $x(t)$ con la respuesta al impulso $h(t)$ de la sala en la que es grabada la señal (*Room Impulse Response* en inglés, RIR) [16] [24] [87] [52] [49], como se muestra en la ecuación 2.7.

$$y(n) = x(t) * h(t) \tag{2.7}$$

La RIR $h(t)$ contiene las propiedades de reverberación de la sala y depende de la absorción acústica de sus paredes, su configuración y la posición del locutor, micrófono y otros objetos presentes en la sala. La RIR es descrita generalmente en tres partes: la componente directa de la señal, las reflexiones tempranas y las reflexiones tardías. Las reflexiones tempranas corresponden a las reflexiones discretas que llegan al micrófono inicialmente, frecuentemente durante los primeros 50 [ms] luego de la llegada de la componente directa de la señal. Estas varían con la posición relativa entre el micrófono y la fuente, y al combinarse con la señal directa, pueden mejorar el proceso de reconocimiento de voz, tanto en el caso del ser humano

[55] como en de el ASR [87]. Las reflexiones tardías consisten en la respuesta acústica que se genera al recibir las reflexiones con una frecuencia tal que $h(t)$ se aproxima a una función continua en el tiempo. Estas son modeladas típicamente con un decaimiento exponencial y son independientes de la posición de la fuente y el micrófono. Algunos consideran estas reflexiones como la principal fuente de degradación en los sistemas de ASR [74].

Si bien la reverberación puede ser tratada como una distorsión convolucional de la voz, en salas reales el tiempo de reverberación (definido como el tiempo requerido para que la presión acústica decaiga 60 [dB]) es generalmente mucho mayor que los 20-35 [ms] de duración propios de un análisis típico por *frames* para ASR u otras tareas. Además, las propiedades de los ambientes reverberantes varían temporal y espacialmente. Por estas razones, los métodos tradicionales propuestos para reducir la distorsión convolucional, como la normalización por media cepstral (*cepstral mean normalization* en inglés, CMN) [58] y el filtrado espectral relativo (RASTA) [31] [30], no logran buenos resultados al al intentar reducir el efecto de la reverberación en el ASR, ya que solo pueden eliminar o compensar las distorsiones convolucionales de sistemas que tengan respuestas al impulso cortas [16] [63].

2.2.2. Métodos de de-reverberación

Se han propuesto múltiples métodos para tratar la distorsión por reverberación, y normalmente se dividen en tres grupos de acuerdo a la etapa en la que son implementados [89] [46] [12]:

- **ASR front-end**: apuntan a mejorar la robustez del *feature vector* a la reverberación. Ejemplos de esto se pueden encontrar en [63], [72] y [77].
- **ASR back-end**: buscan mejorar la robustez del sistema de ASR adaptando el canal acústico, como por ejemplo en [87], [36] y [26].
- **Speech preprocessing**: se aplican directamente a la señal de voz antes de generar los *feature vectors*. Un ejemplo de esto se puede ver en [89].

En [67] se exploran distintos *features* robustos para ser usados en un modelo acústico basado en DNNs convolucionales (CDNN) para realizar ASR en condiciones reverberantes. Los *features* usados sus experimentos se inspiran en el sistema auditivo humano, y logran obtener WERs muy bajas, usando los datos de *REVERB challenge* [44], una base de datos ampliamente utilizada en el campo de *speech enhancement* (SE).

En [23], los autores tratan el problema del reconocimiento de voz distante para ambientes ruidosos y reverberantes. Ellos proponen una arquitectura que combina un sistema GMM junto con una red neuronal recurrente (RNN) del tipo LSTM entrenada para estimar fonemas por *frame*, que luego son convertidos a observaciones de verosimilitud para ser usadas como modelo acústico. Dado que las redes neuronales LSTM pueden aprender dependencias temporales de largo plazo, la robustez al ruido y la reverberación del sistema aumenta.

Muchos algoritmos de *speech preprocessing* se han propuesto, entre los que destacan *Sup-*

pression of Slowly-varying components and the Falling edge (SSF) [42], *Non-negative Matrix Factorization* (NMF) [47], y *Weighted Predicted Error* (WPE) [98] [97]. Este último es de particular interés, y será usado como método de referencia.

El algoritmo WPE se centra en la deconvolución "ciega" (es decir, sin previo conocimiento de la RIR) basada en una predicción lineal de largo plazo, la cual apunta a la reducción de las reflexiones tardías de la reverberación.En este sentido, el algoritmo recibe una única señal de voz, que puede contener varios locutores, ruido de fondo y, lógicamente, reverberación. El proceso de de-reverberación se realiza mediante una predicción lineal de largo plazo en el dominio de la transformada de Fourier de tiempo reducido (*short-time Fourier transform*, en inglés STFT), y permite disminuir considerablemente la distorsión causada por la reverberación.

2.3. Seguimiento de fuente con información visual

Como se mencionó en el capítulo anterior, el estudio de las técnicas de procesamiento de imágenes escapa de los alcances de este trabajo. En cambio, se pretende utilizar directamente los datos que puedan ser obtenidos a partir de la información visual, integrándolos con la información acústica para optimizar el reconocimiento de voz.

La posición de la fuente de voz es un dato que puede usarse para mejorar el desempeño de las técnicas de *speech preprocessing*. Actualmente, los métodos de procesamiento de imágenes permiten conocer dicha posición mediante *object tracking* (seguimiento de objetos). Esto significa seguir el movimiento de un objeto detectado mientras este se mueve entre cuadros de video sucesivos.

Existen múltiples herramientas que pueden usarse para realizar *object tracking*. Una de ellas es la llamada *You Only Look Once* (YOLO) [80], la cual consiste en un sistema de detección y seguimiento de objetos en tiempo real. Esta emplea redes neuronales convolucionales y es ampliamente usada para diversos propósitos, como por ejemplo la detección y seguimiento de organismos marinos [62]. YOLO también fue usada por los ganadores de la *Robocup 2017* en Montreal [64] [39].

Cabe mencionar que las técnicas de procesamiento de audio, como por ejemplo *beamforming*, son sensibles a los errores de seguimiento de fuente. Sin embargo, YOLO ha demostrado ser una herramienta de alta precisión en relación a otras que se encuentran en el estado del arte, y que por lo tanto permite eliminar eficientemente la incertidumbre relacionada al seguimiento del locutor. En la tabla 2.1 se muestra el desempeño de YOLO (particularmente la versión 3-320) junto a otras técnicas de *object tracking*.

En la tabla 2.1 se muestra la *mean Average Precision* (mAP) y el tiempo de inferencia de los algoritmos. La mAP es comúnmente utilizada para medir el desempeño de los algoritmos de *object tracking*. Esta representa la precisión del algoritmo para reconocer un cierto

Método	mAP	Tiempo [ms]
SSD321 [59]	28.0	61
R-FCN [14]	29.9	85
SSD513 [60]	31.2	125
FPN-FRCN [57]	**36.2**	172
YOLOv3-320	28.2	**22**

Tabla 2.1: Desempeño diferentes métodos de *object tracking*.

número de clases de objetos. Para obtenerla, primero se calcula la precisión promedio (AP) en cada clase de la base de datos. La AP se relaciona con el área bajo la curva de precisión-exhaustividad [17]. Por último, para obtener la mAP, se promedian todas las APs.

Por otra parte, el tiempo de inferencia corresponde al tiempo que le toma al método realizar el reconocimiento de una imagen.

Como se puede ver en la tabla 2.1, si bien YOLO no es el método más preciso, es el más rápido. Por su parte, la técnica FPN-FRCN es la más precisa, aunque es casi nueve veces más lenta que YOLO.

La información obtenida usando técnicas como YOLO, al estimar la posición de la fuente, permite calcular la dirección de llegada del audio (*Direction of Arrival* en inglés, DOA), que a su vez sirve como entrada para los algoritmos de *beamforming*, tal como se ha hecho en [69] y [70].

2.4. Beamforming acústico

2.4.1. Descripción general

Un arreglo de micrófonos es una cierta cantidad de micrófonos trabajando juntos. El uso de este puede reducir el efecto de la reverberación y el ruido al eliminar las señales acústicas que no lleguen directamente desde la posición de la fuente [45].

Desde la invención de la *microphone antenna* (antena de micrófonos) en 1974, la técnica de *beamforming* acústico ha presentado grandes avances [65]. Hoy día se utiliza principalmente para localizar fuentes [92] y/o aislarlas del ruido de fondo [21].

Esta técnica consiste en el uso de un arreglo de micrófonos, con el fin de generar el patrón polar que se desee, y así dirigirlo hacia una cierta fuente de audio de interés. De esta manera, dicha fuente se puede aislar parcialmente del resto de las fuentes.

2.4.2. Técnicas más usadas

Delay and sum

Si bien existen muchos métodos para realizar *beamforming*, en este trabajo utilizaremos el más común, denominado *Delay and Sum beamforming*. Este consiste en agregar desfases a las señales capturadas por los micrófonos del arreglo, de manera que al sumarlas, interfieran constructivamente para la señal deseada, y destructivamente para las demás, como se muestra en la figura 2.2.

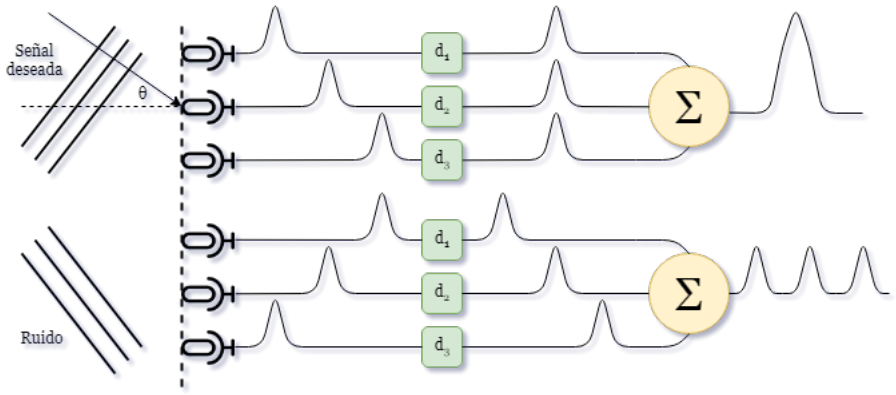

Figura 2.2: *Delay and Sum beamforming*

Asumiendo que las ondas sonoras son planas (como se muestra en la figura 2.2), y tomando $s(t)$ como nuestra señal de interés, entonces para un arreglo lineal de micrófonos tenemos que las señales obtenidas en cada micrófono quedan de la forma:

$$x_{\mathrm{i}}(t) = s(t + \tau_{\mathrm{i}}) \tag{2.8}$$

donde $x_{\mathrm{i}}(t)$ corresponde a la señal capturada por el micrófono i y τ_{i} su respectivo retraso.

Si elegimos, por ejemplo, x_1 como la señal de referencia, los retrasos de cada señal se pueden calcular con la ecuación 2.9, donde l_{i} es la distancia entre el micrófono i y el micrófono 1 (referencia), θ es el ángulo de incidencia de la señal de interés, v_s es la velocidad del sonido en el aire y τ_{i} corresponde al retraso de la señal i.

$$\tau_{\mathrm{i}} = \frac{l_{\mathrm{i}} \cos\theta}{v_s} \tag{2.9}$$

Al aplicar estos retrasos a cada señal y luego sumarlas, se produce interferencia constructiva para la señal que incide con ángulo θ (señal deseada), mientras que las señales que inciden desde otras direcciones quedan desalineadas y por lo tanto se interfieren destructivamente. Por lo tanto, la técnica *Delay and Sum* queda descrita por la siguiente ecuación:

$$b(t) = \sum_{\mathrm{i}=1}^{M} x_{\mathrm{i}}(t - \tau_{\mathrm{i}}) \tag{2.10}$$

donde M es el número de micrófonos y $b(t)$ es la señal resultante. Cabe mencionar que a cada micrófono se le pueden agregar pesos, lo que permite asignarles mayor o menor influencia en la señal resultante. Esta técnica se conoce como *Weighted Delay and Sum*, y queda descrita por la siguiente ecuación:

$$b(t) = \sum_{\mathrm{i}=1}^{M} w_{\mathrm{i}} \cdot x_{\mathrm{i}}(t - \tau_{\mathrm{i}}) \tag{2.11}$$

donde w_{i} corresponde al peso asignado al micrófono i.

Minimum variance distortionless response (MVDR)

La técnica MVDR es usada para mejorar la capacidad de reducción de ruido del *beamforming* suprimiendo ruido correlacionado espacialmente en las direcciones que no son de interés, sin afectar la ganancia en la DOA.

Si tomamos $N(\omega) = [N_1(\omega), N_2(\omega), ..., N_{\mathrm{i}}(\omega), ..., N_M(\omega)]$ como el ruido correlacionado espacialmente en cada micrófono, MVDR ajusta los pesos del *beamforming* minimizando la varianza del ruido a la salida de acuerdo a:

$$argmin_w\{w^H \cdot \sum_N(\omega) \cdot w\} \tag{2.12}$$

donde $\sum_N(\omega) = E\{N(\omega)N_H(\omega)\}$, siendo $E(\cdot)$ la el valor esperado y ω^H la matriz hessiana de ω. Esto define lo siguiente (ecuación 8 en [48]):

$$w^H(\omega) = \frac{v^H(k,\omega) \cdot \sum_N(\omega)}{v^H(k,\omega) \cdot \sum_N(\omega) \cdot v(k,\omega)} \tag{2.13}$$

donde $v(k,\omega) = [\mathrm{e}^{-j\omega\tau_0}, \mathrm{e}^{-j\omega\tau_1}, ..., \mathrm{e}^{-j\omega\tau_{t-1}}]$, y k corresponde al vector de dirección de propagación de la señal de interés.

2.5. Redes neuronales artificiales

2.5.1. Descripción general

Las redes neuronales artificiales (*artificial neural netrworks* en inglés, ANNs o simplemente NNs) son un campo muy importante dentro de la inteligencia artificial. Inspirándose en

el comportamiento conocido del cerebro humano (principalmente el referido a las neuronas y sus conexiones), trata de crear modelos artificiales que solucionen problemas difíciles de resolver mediante técnicas algorítmicas convencionales.

La aparición de los computadores digitales y el desarrollo de las teorías modernas acerca del aprendizaje y del procesamiento neuronal se produjeron aproximadamente al mismo tiempo, a finales de los años cuarenta. Desde entonces, la investigación neurofisiológica y el estudio de sistemas neuronales artificiales han ido de la mano. Los modelos de redes neuronales artificiales toman conceptos e ideas del campo de las ciencias naturales para aplicarlos a la resolución de problemas pertenecientes a otras ramas de las ciencias y la ingeniería. Los primeros ejemplos de estos sistemas aparecen al final de la década de los cincuenta. La referencia histórica más corriente es la que alude al trabajo realizado por Frank Rosenblatt en un dispositivo denominado perceptrón.

2.5.2. El perceptrón simple

El perceptrón es el modelo matemático más simple de una neurona. Corresponde a la unidad básica de inferencia en forma de discriminador lineal, a partir de lo cual se desarrolla un algoritmo capaz de generar un criterio para seleccionar un sub-grupo de un grupo de componentes más grande.

En un sentido moderno, el perceptrón es un algoritmo de aprendizaje de un clasificador binario: una función que mapea su entrada x (un vector de valor real) a un valor de salida $f(x)$ con un valor binario único:

$$f(x) = \begin{cases} 1 & \text{si } w \cdot x > 0 \\ 0 & \text{en otro caso} \end{cases} \tag{2.14}$$

donde w es un vector de pesos de valor real, $w \cdot x$ es el producto punto $\sum_{\mathrm{i}}^{m} \omega_{\mathrm{i}} \cdot x_{\mathrm{i}}$, m es el número de entradas al perceptrón y b es el sesgo o *bias*. El sesgo desplaza el límite de decisión lejos del origen y no depende de ningún valor de entrada. En la figura 2.3 se ilustra un perceptrón con cinco señales de entrada.

En el contexto de redes neuronales, un perceptrón es una neurona artificial que utiliza la función de escalón unitario como función de activación. El perceptrón también se denomina perceptrón de una sola capa, para distinguirlo de un perceptrón multicapa, que es una red neuronal más compleja. Como un clasificador lineal, el perceptrón de una sola capa es la red neuronal *feedforward* más simple.

2.5.3. El perceptrón multicapa

Un perceptrón multicapa (*Multi-Layer Perceptron* en inglés, MLP) es una clase de red neuronal artificial *feedforward*. Un MLP consiste en al menos tres capas de nodos, exceptuando los nodos de entrada. Cada nodo es una neurona que usa una función de activación

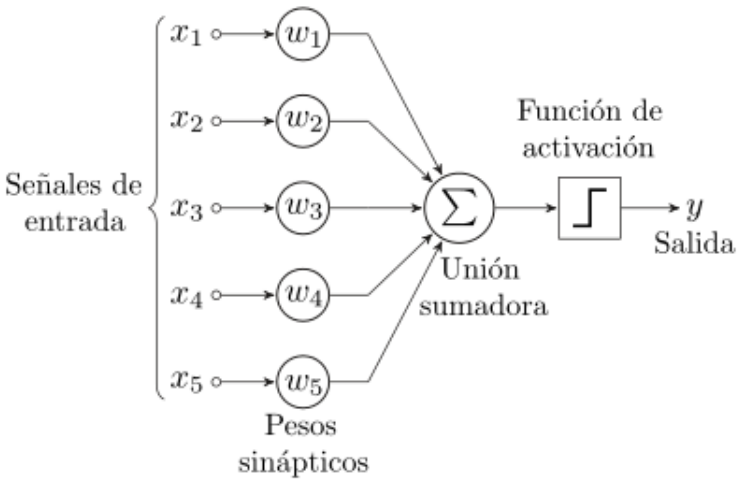

Figura 2.3: Perceptrón simple con cinco entradas.

no lineal. A diferencia del perceptrón de una capa, un MLP puede distinguir datos que no son separables linealmente.

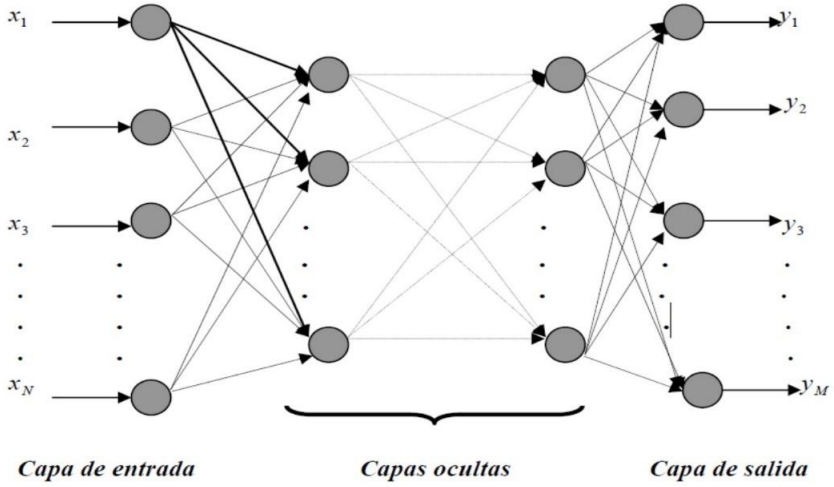

Figura 2.4: Diagrama de un MLP.

Para entrenar un MLP se utiliza una técnica de aprendizaje supervisado llamada *back-propagation* [81]. Esta actúa sobre el perceptrón al cambiar los pesos de conexión después de procesar cada dato, según la cantidad de error en el resultado en comparación con el resultado esperado.

Idealmente, los parámetros del modelo deben ser entrenados para minimizar el costo esperado, dado por:

$$J_{EL} = E(J(W, b; o, y)) = \int_0 J(W, b; o, y))p(o)\mathrm{d}(o) \tag{2.15}$$

donde $J(W, b; o, y)$ es la función de costo dado los parámetros $\{W, b\}$, la observación o, y el correspondiente vector de salida y. Además, $p(o)$ es la función de densidad de probabilidad de la observación o, la que típicamente se estima del set de entrenamiento.

Existen dos criterios populares de entrenamiento para este modelo. Para problemas de regresión, típicamente se usa el criterio de error cuadrático medio (*Mean Square Error* en inglés, MSE):

$$J_{MSE}(W, b; S) = \frac{1}{M} \sum_{m=1}^{M} J_{MSE}(W, b; o^m, y^m) \tag{2.16}$$

donde S corresponde a las muestras de entrenamiento $S = \{(o^m, y^m) | 0 \leq m < M\}$, y $J_{MSE}(W, b; o, y) = \frac{1}{2}||v^L - y||^2 = \frac{1}{2}(v^L - y)^T(v^L - y)$.

Para tareas de clasificación, y es una distribución de probabilidad y comúnmente se usa el criterio de entropía cruzada (*cross-entropy* en inglés, CE):

$$J_{CE}(W, b; S) = \frac{1}{M} \sum_{m=1}^{M} J_{CE}(W, b; o^m, y^m) \tag{2.17}$$

con:

$$J_{CE}(W, b; o, y) = -\sum_{\mathrm{i}=1}^{C} y_{\mathrm{i}} \log v_{\mathrm{i}}^L \tag{2.18}$$

donde $y_{\mathrm{i}} = P_{emp}(\mathrm{i}|o)$ es la probabilidad empírica (observada en el entrenamiento) de que la observación o pertenezca a la clase i, y $v_{\mathrm{i}}^L = P_{\mathrm{dnn}}(\mathrm{i}|o)$ es la misma probabilidad pero estimada por la red.

Los parámetros $\{W, b\}$ del modelo pueden ser estimados con el algoritmo de *backpropagation* [82] el cual se deriva de la regla de la cadena usada para el cálculo del gradiente.

En su forma más simple, los parámetros del modelo pueden ser mejorados en base a la información del gradiente de primer orden como:

$$\begin{aligned} W_{t+1}^l &= W_t^l - \varepsilon \Delta W_t^l \\ b_{t+1}^l &= b_t^l - \varepsilon \Delta b_t^l \end{aligned} \tag{2.19}$$

donde W_t^l y b_t^l son la matriz de pesos y el vector de sesgos de la capa l después de la t-ésima actualización,

$$\begin{aligned} \Delta W_t^l &= \frac{1}{M_b} \nabla_{W_t^l} J(W, b; o^m, y^m) \\ \Delta b_t^l &= \frac{1}{M_b} \nabla_{b_t^l} J(W, b; o^m, y^m) \end{aligned} \tag{2.20}$$

son, respectivamente, el gradiente promedio de la matriz de pesos y el gradiente promedio de los sesgos en la iteración t estimado de la entrada de entrenamiento consistente en M_b muestras, ε es la tasa de aprendizaje, y $\nabla_x J$ es el gradiente de J con respecto a x.

Entonces, para cambiar los pesos de la capa oculta, los pesos de la capa de salida cambian de acuerdo con la derivada de la función de activación, por lo que este algoritmo representa una retropropagación de la función de activación [29].

2.5.4. Teorema de aproximación universal

Los MLPs son aproximadores universales de funciones como lo muestra el teorema de Cybenko [13], por lo que pueden usarse para crear modelos matemáticos mediante análisis de regresión. Como la clasificación es un caso particular de regresión cuando la variable de respuesta es categórica, los MLP son buenos algoritmos clasificadores.

El teorema de aproximación universal se puede enunciar de la siguiente manera: "Sea $\phi(\cdot)$ una función continua no constante, acotada y monótonamente creciente. Sea I_m el hipercubo unitario m-dimensional $[0,1]^m$. Sea $C(I_m)$ el espacio de funciones continuas definidas en I_m. Entonces, dada cualquier función f en $C(I_m)$ y cualquier $\varepsilon > 0$, existe un entero N, constantes reales v_i, $b_\mathrm{i} \in R$ y vectores reales $\omega_\mathrm{i} \in R^m$, donde $\mathrm{i} = 1, \ldots, N$, tales que es posible definir:

$$F(x) = \sum_{\mathrm{i}=1}^{N} v_\mathrm{i} \phi(\omega_\mathrm{i}^T x + b_\mathrm{i}) \tag{2.21}$$

como una realización aproximada de la función f, que es independiente de ϕ; es decir,

$$|F(x) - f(x)| < \varepsilon \tag{2.22}$$

para todo x en I_m. En otras palabras, las funciones de la forma de $F(x)$ son densas en $C(I_m)$".

2.5.5. Deep learning

Los métodos de *deep learning* apuntan a aprender jerarquías de características. Las características de aprendizaje automático en múltiples niveles de abstracción permiten que un

sistema aprenda funciones complejas mapeando la entrada a la salida directamente desde los datos, sin depender completamente de las características creadas por el hombre. Esto es especialmente importante para las abstracciones de alto nivel, que los humanos a menudo no sabemos cómo especificar explícitamente en términos de entrada sensorial cruda. La capacidad de aprender automáticamente funciones de alta complejidad será cada vez más importante a medida que la cantidad de datos y el rango de aplicaciones para los métodos de aprendizaje automático continúe creciendo [5].

Preentrenamiento

Inspirados por la arquitectura del cerebro, los investigadores de redes neuronales quisieron entrenar redes neuronales multicapa profundas durante décadas [7] [90], pero no se reportaron intentos exitosos sino hasta el año 2006: los investigadores informaban resultados experimentales positivos con típicamente dos o tres niveles (es decir, una o dos capas ocultas), pero el entrenamiento de redes más profundas arrojaba de manera consistente resultados peores. En 2006 Geoffrey Hinton y un grupo de científicos en la Universidad de Toronto introdujeron las *Deep Belief Networks* (DBN) [33], usando un método que entrena una capa a la vez mediante un algoritmo de aprendizaje no supervisado llamado *Restricted Boltzmann Machine* (RBM) [19]. Poco después, se propusieron algoritmos relacionados basados en *autocoders* [6] [78], explotando aparentemente el mismo principio: guiar el entrenamiento de niveles intermedios de representación usando aprendizaje no supervisado, que puede realizarse localmente en cada nivel. Recientemente se han propuesto otros algoritmos para arquitecturas profundas que no explotan RBM ni *autocoders* y que explotan el mismo principio [68] [94].

Desde 2006, las redes profundas se han aplicado con éxito no solo en tareas de clasificación [3] [6] [51], sino también en regresión [35], reducción de dimensionalidad [34], modelado de texturas [71], segmentación de objetos [53], recuperación de información [79], robótica [28], procesamiento de lenguaje natural [11] y filtrado colaborativo [84]. Aunque los *autoencoders*, las RBM y las DBN se pueden entrenar con datos no etiquetados, en muchas de las aplicaciones anteriores, se han utilizado con éxito para inicializar redes neuronales *feedforward* profundas supervisadas aplicadas a una tarea específica [5].

2.5.6. Arquitecturas

Feedforward

Las redes profundas *feedforward* (*deep feedforward networks*), también llamadas redes neuronales prealimentadas, o perceptrones multicapa (MLP), son los modelos de *deep learning* por excelencia. El objetivo de una red *feedforward* es aproximar una función f^*. Por ejemplo, para un clasificador, $y = f^*(x)$ mapea una entrada de una categoría. Una red *feedforward* define un mapeo $y = f(x; \theta)$ y aprende el valor de los parámetros θ que resultan en la mejor aproximación de función.

Un conjunto típico de ecuaciones para redes neuronales multicapa [82] es el siguiente. Como se ilustra en la figura 2.5, la capa k calcula un vector de salida h^k utilizando la salida de la capa anterior $h^{(k-1)}$, comenzando por la entrada $x = h^0$, de acuerdo a:

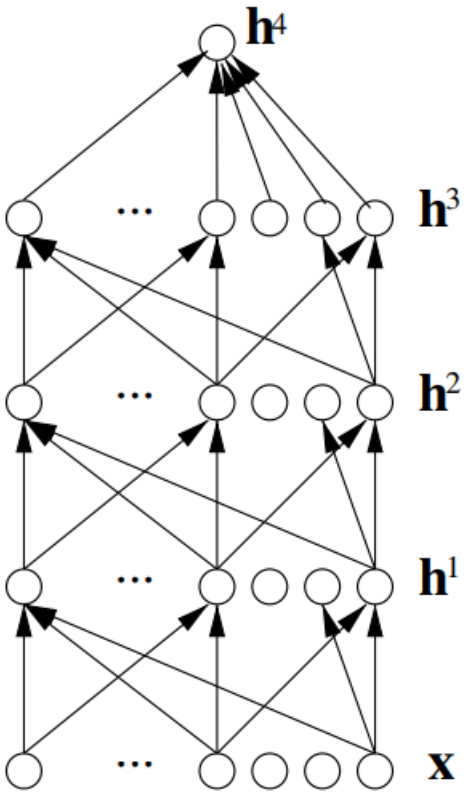

Figura 2.5: Arquitectura *feedforward*.

$$h^k = \tanh\left(b^k + W^k h^{(k-1)}\right) \tag{2.23}$$

con parámetros b^k (un vector de sesgos) y W^k (una matriz de pesos). La tangente hiperbólica se aplica elemento a elemento y puede ser reemplazada por $\sigma(u) = \frac{1}{1+\mathrm{e}^{-u}} = \frac{1}{2}(\tanh u + 1)$ u otras funciones no lineales de saturación. La salida de la capa superior h^l se usa para hacer una predicción y se combina con un objetivo supervisado y en una función de costo $L(h^l, y)$, típicamente convexa en $b^l_{\mathrm{i}} + W^l_{\mathrm{i}} h^{(l-1)}$. La capa de salida puede tener una función no lineal diferente de la utilizada en otras capas, por ejemplo, la función *softmax*:

$$h^l_{\mathrm{i}} = \frac{\mathrm{e}^{b^l_{\mathrm{i}} + W^l_{\mathrm{i}} h^{(l-1)}}}{\sum_j \mathrm{e}^{b^l_j + W^l_j h^{(l-1)}}} \tag{2.24}$$

donde W^l_{i} es la i-ésima fila de W^l, h^l_{i} es positiva y $\sum_{\mathrm{i}} h^l_{\mathrm{i}} = 1$. La salida *softmax* h^l_{i} puede usarse como un estimador de $P(Y = \mathrm{i}|x)$, con la interpretación de que Y es la clase asociada con el patrón de entrada x. En ese caso, a menudo se usa el negativo de la verosimilitud de condicional $L(h^l, y) = -\log P(Y = y|x) = -\log h^l_y$ como función de costo, cuyo valor esperado sobre los pares (x, y) debe ser minimizado.

Recurrent neural networks

Las redes neuronales recurrentes, o RNN [82], son una familia de redes neuronales usadas para procesar datos secuenciales y pueden escalar a secuencias mucho más largas de lo que sería práctico para redes convencionales *feedforward*. La mayoría de las redes recurrentes también pueden procesar secuencias de longitud variable. La idea detrás de las RNN es hacer uso de la información secuencial. En una red neuronal tradicional suponemos que todas las entradas (y salidas) son independientes entre sí, lo que para muchas tareas no es adecuado. Las RNN se llaman recurrentes porque realizan la misma tarea para cada elemento de una secuencia, y la salida depende de los cálculos previos.

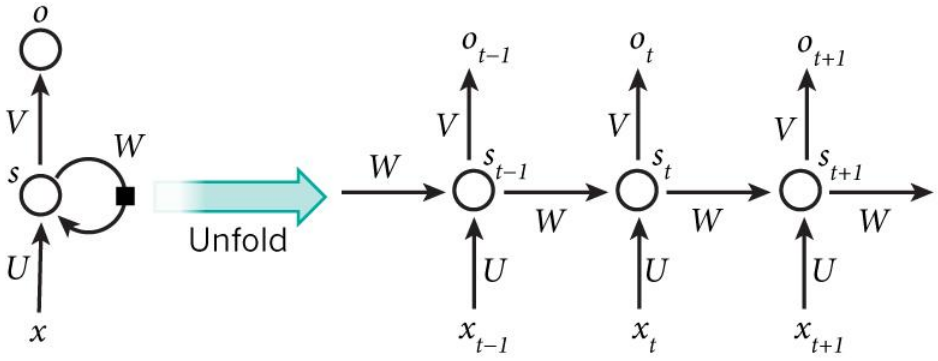

Figura 2.6: Esquema de una RNN.

La figura 2.6 muestra una RNN desplegada en una red completa, donde:

- x_t es la entrada en el paso de tiempo t. Por ejemplo, x_1 podría ser un vector correspondiente a la segunda palabra de una oración.
- s_t es el estado oculto en el paso de tiempo t (la "memoria" de la red). s_t se calcula en función del estado oculto anterior y la entrada actual como $s_t = f(Ux_t + Ws_{t-1})$. La función f generalmente es no-lineal (por ejemplo tanh o ReLU).
- o_t es la salida en el tiempo t y se calcula como $o_t = softmax(Vs_t)$.

Las RNN, y en menor medida los MLPs, sufren del denominado problema de desvanecimiento del gradiente (*vanishing gradient problem*). En *deep learning*, el problema de desvanecimiento del gradiente es una dificultad que se encuentra en el entrenamiento de redes neuronales artificiales con métodos de aprendizaje basados en gradiente y *backpropagation*. En dichos métodos, cada uno de los pesos de la red neuronal se actualiza de manera proporcional a la derivada parcial de la función de costo con respecto al peso actual en cada iteración de entrenamiento. El problema es que, en algunos casos, el gradiente será infinitamente pequeño, lo que evitará que el peso cambie su valor. En el peor de los casos, esto puede detener por completo el aprendizaje de una red neuronal. Como un ejemplo de la causa del problema, las funciones de activación tradicionales, como la función de tangente hiperbólica, tienen gradientes en el rango $(0, 1)$ y el algoritmo de *backprogation* calcula los gradientes según la regla de la cadena. Esto tiene el efecto de multiplicar n de estos pequeños números para calcular los gradientes de las capas "frontales" en una red de n capas, lo que significa que el gradiente (señal de error) disminuye exponencialmente con n, haciendo que las capas frontales se entrenen muy lentamente. Este problema, identificado por Hochreiter

en 1991 [37], no solo afecta a las redes *feedforward* de muchas capas, sino también a las redes recurrentes. Estas últimas se entrenan desplegándolas en redes *feedforward* muy profundas, donde se crea una nueva capa para cada paso de una secuencia de entrada procesada por la red.

Una forma de resolver este problema son las redes *Long-Short Term Memory* (LSTM), que serán explicadas a continuación.

LSTM

Una red LSTM es una red neuronal recurrente que tiene bloques de células LSTM en lugar de capas de redes neuronales estándar. Estas celdas se componen de una puerta de entrada (*input gate*), una puerta de olvido (*forget gate*) y una puerta de salida (*output gate*), las cuales se describen más adelante. En la figura 2.7 se muestra una representación gráfica de la celda LSTM.

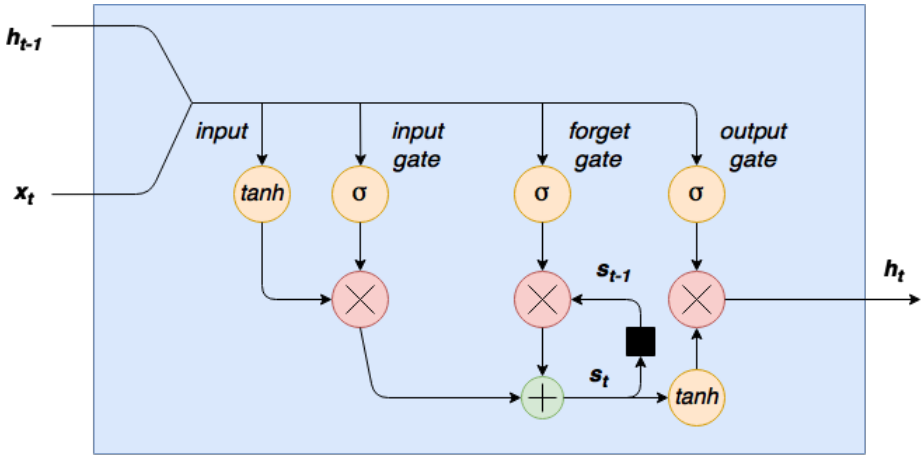

Figura 2.7: Esquema de una red LSTM.

La entrada a la celda consiste en la secuencia x_t concatenada con el resultado anterior de la celda h_{t-1}. Esta entrada alimenta tanto a una capa *tanh* como a las puertas de entrada, olvido y salida. Una puerta de entrada es una capa de nodos con activación sigmoide cuya salida se multiplica por la salida de la capa *tanh*. Las activaciones sigmoides de la puerta de entrada actúan eliminando o conservando cualquier elemento del vector de entrada según sea necesario, entrenando los pesos que conectan la entrada a estos nodos para generar valores de salida cercanos a 0 o a 1 para distintos valores de entrada.

El siguiente paso en el flujo de datos a través de esta celda es el ciclo estado interno/-puerta de olvido. Las células LSTM tienen una variable de estado interna s_t. Esta variable, rezagada un paso de tiempo (es decir, s_{t-1}) se agrega a los datos de entrada para crear una capa efectiva de recurrencia. Esta operación de adición, en lugar de una operación de multiplicación, ayuda a reducir el riesgo de desvanecimiento de gradiente. Sin embargo, este ciclo de recurrencia está controlado por una puerta de olvido que funciona de manera similar a la

compuerta de entrada, pero ayuda a la red a saber qué variables de estado deben "recordarse" u "olvidarse".

Finalmente, tenemos una función *tanh* como capa de salida, controlada por una puerta de salida. Esta puerta determina qué valores están realmente permitidos como salida de la celda h_t.

Las ecuaciones que rigen la celda LSTM para cada etapa son las siguientes:

Capa de entrada:

$$g = \tanh(b^g + x_t U^g + h_{t-1} V^g) \tag{2.25}$$

donde U^g y V^g son los pesos para la entrada y la salida de celda anterior, respectivamente, y b_g es el sesgo de entrada.

Puerta de entrada:

$$\mathrm{i} = (b^{\mathrm{i}} + x_t U^{\mathrm{i}} + h_{t-1} V^{\mathrm{i}}) \tag{2.26}$$

donde U^{i} y V^{i} son los pesos para la entrada y la salida de celda anterior, respectivamente, y b_{i} es el sesgo de entrada. La salida de la sección de entrada de la celda LSTM viene dada por: $g \odot \mathrm{i}$. Donde el operador $\odot$ expresa multiplicación a nivel de elementos.

Loop estado interno/puerta de olvido:

$$f = (b^f + x_t U^f + h_{t-1} V^f) \tag{2.27}$$

La salida del producto por elemento del estado anterior y la puerta de olvido se expresa como $s_{t-1} \odot f$. La salida del ciclo es: $s_t = s_{t-1} \odot f + g \odot \mathrm{i}$

Puerta de salida:

$$o = (b^o + x_t U^o + h_{t-1} V^o) \tag{2.28}$$

Por lo tanto, la salida final de la celda corresponde a:

$$h_t = \tanh(s_t) \odot o \tag{2.29}$$

Capítulo 3

Metodología

Si bien los sistemas de ASR hoy en día permiten WER's de menos del 5 % [10], en ambientes ruidosos y/o reverberantes, este desempeño puede decaer drásticamente. Por lo mismo, el problema radica principalmente en reducir la intensidad de ambos factores (ruido y reverberación).

Por simplicidad, en este trabajo se asumirá que la fuente de ruido es estática, aunque la metodología expuesta puede extenderse al caso dinámico.

Por otra parte, la reverberación estará determinada por las características de la sala en que se realizarán los experimentos, es decir, sus dimensiones, materiales de las superficies (paredes, ventanas, techo y piso) y los elementos presentes en ella (sillones, cortinas, pantallas, etc.).

3.1. Modelo

3.1.1. Distorsión con ruido aditivo

Como se dijo anteriormente, se asumirá que la fuente de ruido es estática, y que se estimó su posición usando información visual. La señal resultante $b(t)$ se puede escribir como:

$$b(t) = g_s \cdot s(t) + g_n \cdot n(t) \tag{3.1}$$

donde $s(t)$ corresponde a la señal de voz, $n(t)$ al ruido y las constantes g_s y g_n a sus ganancias respectivas.

3.1.2. Distorsión por reverberación

En base a la ecuación 2.7 presentada en el capítulo anterior, los efectos del canal sobre la señales de voz y ruido se pueden modelar como la convolución entre estas y sus respectivas RIRs, con lo que la señal resultante $b_{rev}(t)$ queda:

$$b_{rev}(t) = h_s(t) * \Big(g_s \cdot s(t)\Big) + h_n(t) * \Big(g_n \cdot n(t)\Big) \tag{3.2}$$

donde $h_s(t)$ y $h_n(t)$ representan las RIRs que modelan el canal acústico en el tiempo correspondientes a la señal de voz $s(t)$ y al ruido $n(t)$ respectivamente.

3.2. Bases de datos

Dado que para entrenar redes neuronales artificiales se requiere un gran volumen de datos, resulta más conveniente sintetizarlos de manera automática a partir de señales limpias. Se usará la base de datos *Aurora-4* [73] como set de señales limpias y señales de ruido obtenidas de la misma base de datos.

La base de datos *Aurora-4* limpia contiene 7138 *utterances* de 83 locutores distintos para el set de entrenamiento, 330 *utterances* de 10 locutores para evaluación y 330 *utterances* de otros 8 locutores para el set de prueba. Todas las señales fueron grabadas usando un micrófono *Sennheiser HMD 414*.

En primer lugar, se debe aplicar convolución entre las señales, y las respuestas al impulso (descritas más adelante), obteniendo así las señales de voz y ruido reverberadas por separado.

Una vez que las señales de voz y ruido están reverberadas, estas se suman ponderadas por distintas ganancias, de acuerdo al SNR (Signal to Noise Ratio) deseado.

3.2.1. Grabación de las RIRs

Para llevar a cabo los experimentos, se calculó un set de respuestas acústicas al impulso de la sala para los cuatro micrófonos de la Kinect, utilizando el método de Farina [18]. Estas RIRs sirven para sintetizar las bases de datos que luego serán necesarias en el entrenamiento y testeo de las distintas redes neuronales utilizadas en este trabajo.

La grabación se realizó en una sala de reuniones, cuyas dimensiones se describen en la figura 3.1.

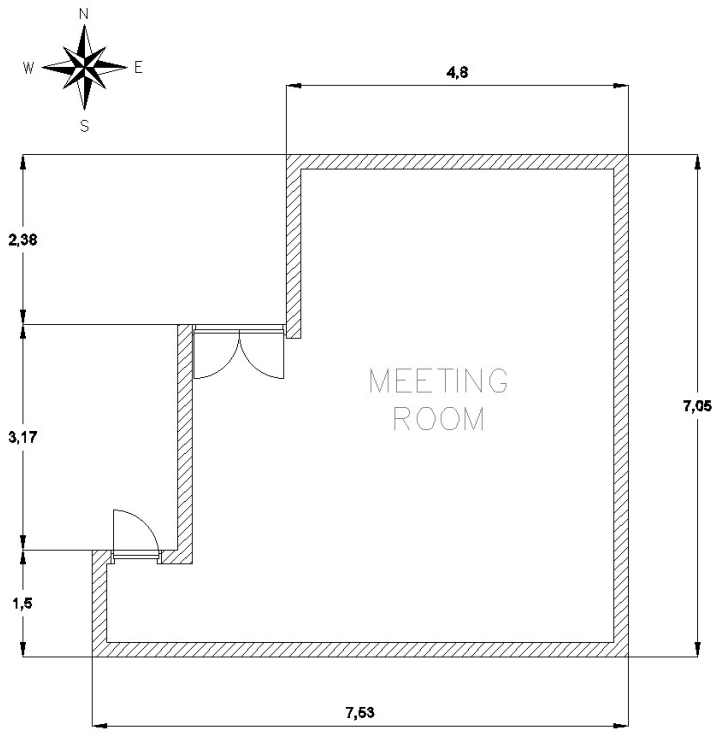

Figura 3.1: Plano de la sala de reuniones usada para grabar la base de datos.

Se utilizó el robot PR2 (figura 3.2), para grabar el audio en cuatro pistas en formato WAV mediante el arreglo de micrófonos de una Kinect montada en su cabeza, bajo la configuración descrita en la figura 3.3. Se ubicó un monitor de estudio (TANNOY 501a) a un metro del robot, el cual fue usado como fuente de audio para reproducir un barrido exponencial en frecuencia, cuya grabación permite calcular una RIR mediante el método señalado anteriormente.

Figura 3.2: Robot PR2 utilizado para grabar la base de datos de RIRs.

Los barridos se reprodujeron 21 veces con pausas de 10 segundos, sin alterar el montaje, con el fin de obtener RIRs similares. Cada barrido fue grabado por los cuatro micrófonos de

la Kinect y luego estas cuatro señales fueron procesadas usando la técnica *Delay and sum* descrita en la sección 2.4, obteniendo una sola señal a partir de estas cuatro. Con esta señal resultante se calcula finalmente la RIR.

Este proceso se repite con la fuente ubicada a 0° y 45° del eje perpendicular la Kinect, y con el *beamforming* apuntando en 0°, obteniendo de esta manera 2 RIRs por cada uno de los 21 sets. Estas 2 RIRs son:

- $h_s(t)$: corresponde al canal entre la fuente de voz y la Kinect.
- $h_n(t)$: corresponde al canal entre la fuente de ruido y la Kinect.

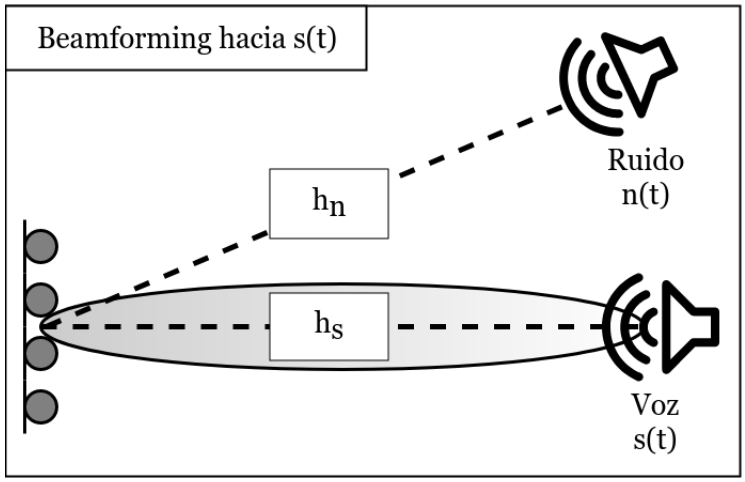

Figura 3.3: Descripción del montaje usado para grabar las RIRs.

3.2.2. Datos para cancelación de reverberación

Como se mostrará en la próxima sección, el problema de la reverberación fue abordado mediante dos métodos. El primero corresponde al llamado WPE (descrito en el capítulo anterior, sección 2.2), y el segundo corresponde a una red LSTM basada en la que se usa en [101]. Ambas técnicas son comparadas para tres los siguientes casos:

- **Sin ruido**: convolución entre la señal limpia y las RIRs. Es decir, $b_1(t) = h(t) * s(t)$.
- **Con ruido y una RIR**: convolución entre la señal con ruido y las RIRs. Es decir, $b_2(t) = h(t) * (s(t) + n(t))$.
- **Con ruido y dos RIRs**: suma entre la señal limpia convolucionada con una RIR y el ruido convolucionado con otra RIR distinta. Es decir, $b_3(t) = h_s(t) * s(t) + h_n(t) * n(t)$.

Por lo tanto, fue necesario generar tres bases de datos, una para cada caso. Estas se describen en la tabla 3.1.

Para las tres bases de datos se usó la siguiente distribución de los 21 sets de RIRs: 10 sets para datos de entrenamiento, 5 para evaluación y 6 para pruebas.

Base de datos	Señales de entrada	Señales de referencia
1. Sin ruido	$h(t) * s(t)$	$s(t)$
2. Con ruido, 1 RIR	$h(t) * (s(t) + n(t))$	$s(t) + n(t)$
3. Con ruido, 2 RIRs	$h_s(t) * s(t) + h_n(t) * n(t)$	$s(t) + n(t)$

Tabla 3.1: Base de datos para cancelación de reverberación.

3.2.3. Datos para cancelación de ruido

Para la implementación de las redes neuronales usadas para la cancelación de ruido se generaron señales de audio con ruido a distintos SNRs, implementando en MATLAB la ecuación 3.1 (sección 3.1). Para esto, usando la base de datos *Aurora-4* con señales limpias, y la señal de ruido "*RESTAURANT*" de dicha base de datos, se siguió el siguiente algoritmo:

1. Seleccionar aleatoriamente un intervalo de la señal de ruido con la misma duración que la señal de voz.
2. Normalizar la señal de voz ($s(t)$) y el ruido seleccionado ($n(t)$) para que tengan potencia unitaria.
3. Definir la ganancia de la señal de voz (g_s) como un número aleatorio entre 0.5 y 1.5.
4. Definir el SNR de manera aleatoria en un intervalo de 0 a 15 [dB].
5. Calcular la ganancia de la señal de ruido (g_n) a partir de g_s y el SNR generado.
6. A partir de g_s, g_n, $s(t)$ y $n(t)$, calcular la señal con ruido $b(t)$ de acuerdo a la ecuación 3.1.
7. Guardar g_s, g_n, $s(t)$, $n(t)$ y $b(t)$ en archivos formato *.mat*.

Este procedimiento se aplicó de dos maneras. La primera consiste en dividir la señal en ventanas de tiempo (*frames*) y aplicar el algoritmo a cada ventana, y la segunda, aplicar el algoritmo a las señales completas, simulando con esto las señales reales. Los datos obtenidos mediante la primera manera fueron utilizados para seleccionar el mejor método en términos de error de las señales (lo cual se explicará en la sección 3.5), mientras que los datos obtenidos de la segunda manera se usaron para realizar reconocimiento automático de voz.

Además, para el segundo caso, se calcularon los SNRs de cada frame y se guardaron en una variable aparte para cada *utterance*. Estos se usarán para obtener un resultado anexo que permitirá establecer si el error de aproximación de cada *frame* se relaciona de alguna manera con su SNR.

3.3. Dereverberación

3.3.1. Métodos implementados

Como se vio en el capítulo 2, existen diversos métodos para cancelar la reverberación. En este trabajo se comparan dos técnicas que han demostrado ser efectivas. La primera corresponde a la llamada Weighted Prediction Error (WPE), descrita en la sección 2.2. Para su

implementación, se trabajó en MATLAB usando el *toolkit* disponible en la página web de los creadores de WPE [50].

Por otra parte, también se revisaron sistemas de de-reverberación en base a redes neuronales *Long-Short Term Memory* (LSTM) con las que también se han reportado buenos resultados, por lo que el segundo método corresponde una red LSTM, cuya implementación se describe a continuación.

3.3.2. Descripción de la red LSTM

La red LSTM que se implementó está basada en el sistema que propone Yan Zhao et al. en [101], y se describe a continuación.

Para ingresar a la red, las señales son divididas en ventanas de Hamming de 25 [ms], con un traslape de 10 [ms]. A cada ventana se le aplica una FFT de 512 puntos y se le extrae el módulo, con lo que se obtiene un *feature vector* de 257 *bins*, los cuales finalmente son normalizados a promedio cero y varianza unitaria. La red se compone de dos capas LSTM, con 400 unidades ocultas cada una y una capa lineal a la salida, como se ilustra en la figura 3.4. Se utilizó el optimizador *Adam* [43] y MSE como función de costo.

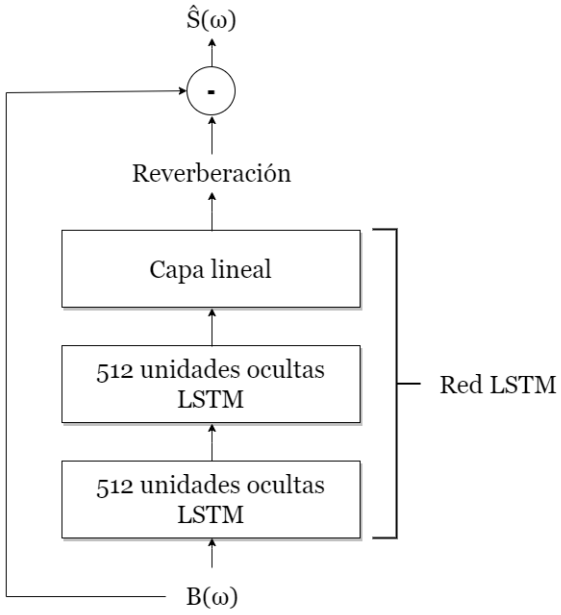

Figura 3.4: Esquema de la red LSTM usada para de-reverberar las señales

Si describimos la red como una caja negra, esta recibe una señal reverberada y estima una señal sin reverberación. Sin embargo, lo que la red estima internamente es la reverberación, y luego la sustrae de la señal de entrada, lo cual se muestra en la figura 3.4.

3.4. Cancelación de ruido

Si asumimos que la reverberación pudo ser completamente eliminada, las señales resultantes corresponden a la suma entre la señal limpia y el ruido (ecuación 3.1). Por lo tanto, para estimar la señal de voz limpia se debe sustraer el ruido. De acuerdo al teorema de la aproximación universal (sección 2.5.4), podemos estimar la señal limpia usando una red neuronal artificial. Para esto se implementó una arquitectura *feedforward* (descrita en la sección 2.5.6) usando la librería *Tensorflow* para *Python 3*. A continuación se describe el sistema implementado en su totalidad, desde el la extracción de *features* y la posterior normalización de estos, hasta la arquitectura de la red.

3.4.1. Features

Para ingresar a la red, las señales fueron subdivididas en ventanas de Hamming de 25 [ms] con traslape de 15 [ms]. A esta ventanas se les aplicó la transformada rápida de Fourier (FFT) de 512 puntos, a la cual se le calculó la magnitud, resultando vectores con 257 *bines* para cada *frame*. Este proceso (llamado *feature extraction*) se ilustra en la figura 3.5.

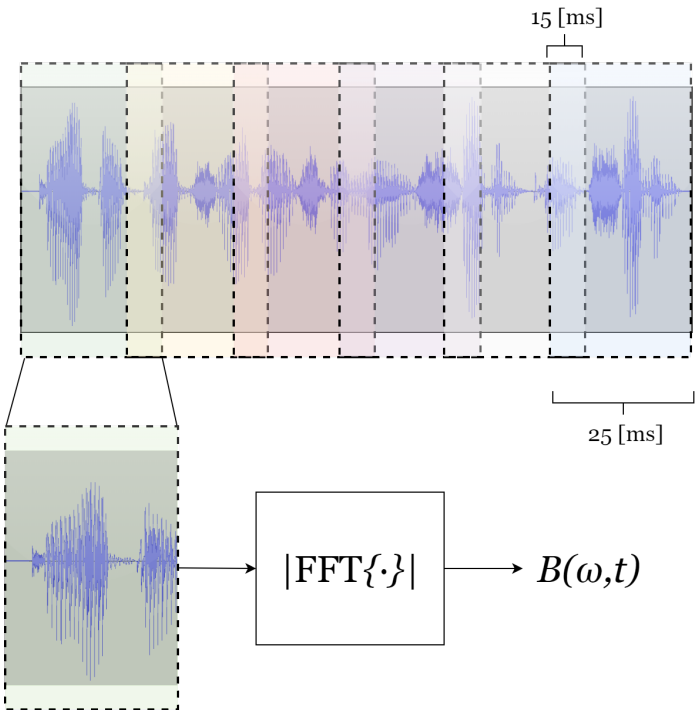

Figura 3.5: Proceso de extracción del vector característrico (feature extraction).

Luego de obtener los vectores característicos de las señales, estos se deben normalizar para ingresar a la red. En la siguiente sección se describen las normalizaciones aplicadas.

3.4.2. Normalización

La normalización es una técnica que a menudo se aplica como parte de la preparación de datos para una red neuronal. El objetivo de la normalización es cambiar los valores numéricos del conjunto de datos para usar una escala común, sin distorsionar las diferencias

en los rangos de valores o perder información. Para los *features* que se utilizaron, es posible normalizar de dos maneras:

- **Normalización por frame**: se aplica la normalización a cada *frame* por separado, de manera que los *bins* de un mismo *frame* tomen valores en un rango determinado, sin perder la relación entre ellos, aunque perdiendo la relación entre distintos *frames*.
- **Normalización por bin**: se aplica la normalización a la trayectoria del *bin*, con el objetivo de que esta se mantenga dentro de un rango de valores a lo largo de la señal, aunque perdiendo la relación entre los *bins* de un mismo *frame*.

A su vez, existen muchas técnicas para normalizar. En este trabajo se usaron dos:

- **Raíz cuadrada de la potencia**: consiste en dividir los datos por la raíz cuadrada de la potencia de estos, es decir, la constante de normalización se define para cada *frame* o *bin* n (dependiendo del caso) como:

$$K_{norm} = \sqrt{\frac{1}{N} \cdot \sum_{n=1}^{N} B_n^2} \tag{3.3}$$

  donde N es el total de datos. Con esto, los datos normalizados resultan:

$$B_{pow} = \frac{B}{K_{norm}} \tag{3.4}$$

- **Log-MVN**: se aplica la función logarítmica a los datos, luego se resta el promedio de los datos resultantes y se divide por la desviación estándar de estos, con el fin de que el promedio de los datos sea cero y la varianza unitaria, es decir:

$$B_{log-MVN} = \frac{\log B - \mu(\log B)}{\sigma(\log B)} \tag{3.5}$$

  donde μ es el promedio de los datos y σ la desviación estándar.

En la tabla 3.2 se muestra un resumen de las normalizaciones utilizadas. $B(n,k)$ corresponde al *feature* sin normalizar para el *frame* n y el *bin* k, N y K son el total de *frames* y *bins* respectivamente, μ_n y σ_n son los promedios y desviaciones estándar del *frame* n, y μ_k y σ_k son los promedios y desviaciones estándar del *bin* k.

	Por frame	Por bin
Raíz de la potencia	$\frac{B(n,k)}{\sqrt{\frac{1}{N}\cdot\sum_{n=1}^{N} B(n,k)^2}}$	$\frac{B(n,k)}{\sqrt{\frac{1}{K}\cdot\sum_{k=1}^{K} B(n,k)^2}}$
Log-MVN	$\frac{\log B(n,k) - \mu_n\left(\log B(n,k)\right)}{\sigma_n\left(\log B(n,k)\right)}$	$\frac{\log B(n,k) - \mu_k\left(\log B(n,k)\right)}{\sigma_k\left(\log B(n,k)\right)}$

Tabla 3.2: Tabla de normalizaciones

3.4.3. Arquitectura de la DNN

La DNN implementada para realizar la reducción del ruido aditivo consta de tres capas ocultas y una capa lineal de salida. Las capas contienen 1024 unidades cada una. Los *features* ingresan a la red en *batches* (o "lotes") de 512 *frames* elegidos aleatoriamente de todo el conjunto de entrenamiento. Se utilizó el optimizador Adam [43] y la función de costo empleada fue el error cuadrático medio (MSE). En la figura 3.6 se ilustra la red neuronal implementada.

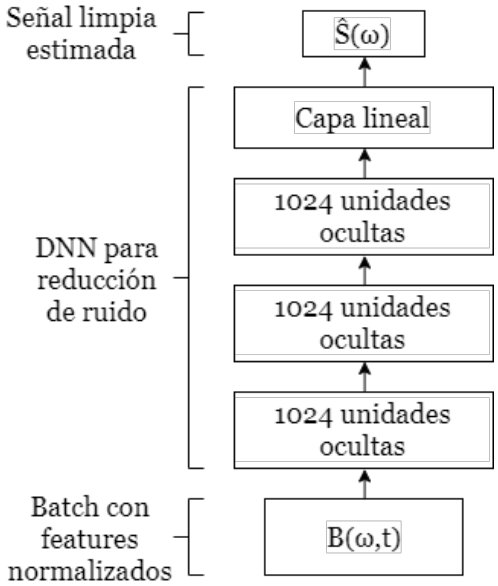

Figura 3.6: Arquitectura de la DNN implementada.

Para el pre-entrenamiento de las tres capas ocultas se utilizaron RBMs (descritas en 2.5). Particularmente se usó una RBM *Gaussian-Bernoulli* (GBRBM) para pre-entrenar la primera capa y una RBM *Bernoulli-Bernoulli* (BBRBM) para las dos restantes. La capa lineal se pre-entrenó de manera supervisada con el método de *backpropagation* implementado en *Tensorflow*.

Para cada forma de normalización, la red se entrenó de dos maneras: primero con la señal de voz limpia $\hat{S}(\omega)$ como referencia (que es el caso ilustrado en la figura 3.6), y luego la misma señal multiplicada por su ganancia, es decir, $g_s \cdot \hat{S}(\omega)$.

3.5. Métricas

Para medir el desempeño de los métodos implementados, se utilizaron diferentes métricas. En el caso del sistema de de-reverberación se usó la WER (sección 2.1.2), mientras que para el sistema de reducción de ruido, además de la WER, se calcularon otras dos métricas: el error relativo y el MSE sobre los filtros de Mel. Estas tres métricas se definen a continuación.

3.5.1. Error relativo

Para cuantificar el error de estimación del sistema de reducción de ruido, se calculará para cada *frame* n la siguiente métrica:

$$\varepsilon(n) = \frac{1}{K} \sum_{k=1}^{N} \frac{|S(n,k) - \hat{S}(n,k)|}{S(n,k)} \tag{3.6}$$

donde $S(n,k)$ corresponde al *bin* k del *frame* n de la señal de referencia, $\hat{S}(n,k)$ es la estimación realizada, y K es el número de *bins* de la FFT. Finalmente, el error de estimación se calculará como el promedio de estos errores:

$$\varepsilon_{relativo} = \frac{1}{N} \sum_{n=1}^{N} \varepsilon(n) \tag{3.7}$$

donde N es el número total de frames.

3.5.2. MSE sobre filtros Mel

Como se verá en la próxima sección, el sistema de ASR que se utilizó usa un banco de filtros Mel (MelFB) sobre las señales de entrada, por lo que se hace muy conveniente estimar el error cuadrático medio (MSE) entre los MelFB de la señal estimada y la señal limpia. Esto se calcula para cada *frame* como sigue:

$$\varepsilon(n) = \frac{1}{K} \sum_{k=1}^{K} \Big(\text{MelFB}\{S(n,k)\} - \text{MelFB}\{\hat{S}(n,k)\} \Big) \tag{3.8}$$

donde $S(n,k)$ corresponde al *bin* k del *frame* n de la señal de referencia, $\hat{S}(n,k)$ es la estimación realizada, y K es el número de *bins* de la FFT. Luego, calculando el promedio de estos errores para los N *frames* se obtiene:

$$\varepsilon_{MelFB} = 100 \cdot \frac{1}{N} \sum_{n=1}^{N} \varepsilon(n) \tag{3.9}$$

3.5.3. Word Error Rate

La WER, descrita en la sección 2.1.2, entrega una medida de la efectividad de las técnicas de reducción de ruido y/o reverberación usadas antes del proceso de reconocimiento automático de voz. Por lo tanto, se usó un sistema ASR para realizar *decodes* con las señales

obtenidas luego de la aplicación de los métodos propuestos y así obtener esta métrica.

El sistema de ASR a utilizar fue diseñado e implementado en el Laboratorio de Procesamiento y Transmisión de la Voz de la Universidad de Chile [96]. Este se compone de una DNN con siete capas ocultas de 2048 unidades cada una, en conjunto con un Modelo Oculto de Márkov (en inglés, *Hidden Markov Model*, HMM) para realizar el reconocimiento automático de voz. El sistema se ilustra en la figura 3.7.

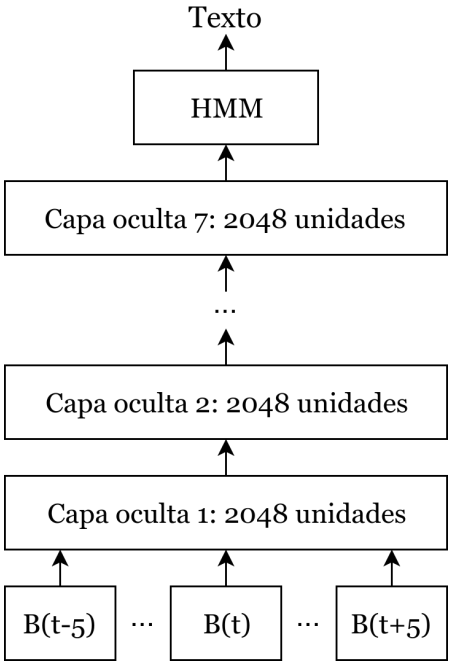

Figura 3.7: Arquitectura del ASR utilizado

La implementación del sistema se hizo usando el *Kaldi Speech Recognition Toolkit* [75]. En primer lugar, se entrenó un GMM-HMM de acuerdo a lo señalado en la receta *tri2b* de Kaldi con la base de datos *Aurora-4* descrita en la sección 3.2, usando Coeficientes Cepstrales en las Frecuencias de Mel (MFCC) como *features*, análisis discriminante lineal (LDA) [2] y transformaciones lineales de máxima verosimilitud (MLLT). Primero se entrenó un sistema usando mono-fonemas, para luego usar los alineamientos de este en el entrenamiento del sistema con tri-fonemas. A continuación, se reemplazó el GMM por la DNN de siete capas descrita anteriormente, cuya entrada considera una ventana de 11 *frames* de contexto. Esta DNN fue entrenada usando filtros de bancos Mel (MelFB) de las señales limpias como *features*, mediante el criterio de entropía cruzada. Por último, el sistema final se obtiene re-entrenando la DNN con mediante el entrenamiento discriminativo sMBR propuesto en [91].

Para realizar el proceso de *decoding*, se utilizó modelo de lenguaje estándar de la base de datos DARPA del *Wall Streen Journal* [22]. Una vez obtenidas las transcripciones, se calcularon las WERs de acuerdo a la ecuación 2.6.

Capítulo 4

Resultados

4.1. Dereverberación

Usando las bases de datos de prueba descritas en la sección 3.2 se procedió a realizar *decodes* de las señales obtenidas por la técnica WPE y la red LSTM implementada. Estos *decodes* se llevaron a cabo en el sistema ASR descrito en la sección 3.5.3, con el fin comparar el desempeño de ambas técnicas en términos de WER.

Como se señaló en la sección 3.2, se consideraron tres casos: uno sin ruido aditivo y una RIR; otro con ruido y una RIR; y por último uno con ruido y dos RIRs (una para la señal de voz y otra para el ruido). A continuación se exponen los resultados obtenidos para los tres casos.

4.1.1. Sin ruido aditivo y una RIR

Los resultados para este caso permiten analizar el desempeño de los métodos implementados para el caso ideal, es decir, cuando la señal está distorsionada exclusivamente por la reverberación. En la tabla 4.1 se muestran los resultados para este caso.

Método	**WER (%)**
WPE$\{h(t) * s(t)\}$	5,98
LSTM$\{h(t) * s(t)\}$	11,15

Tabla 4.1: Resultados de la dereverberación para el caso sin ruido aditivo.

Como se puede deducir de la tabla, el uso de la técnica WPE conlleva una WER 46,37 % menor relativa al uso de la red LSTM. Por lo tanto, para el caso ideal es preferible utilizar el método WPE.

4.1.2. Con ruido aditivo y una RIR

Este caso considera que sí hay ruido, pero se asume que este comparte el mismo canal acústico que la señal de voz. Los resultados para este caso se muestran a continuación en la tabla 4.2.

Método	**WER (%)**
WPE$\big\{h(t) * \big(s(t) + n(t)\big)\big\}$	34,74
LSTM$\big\{h(t) * \big(s(t) + n(t)\big)\big\}$	48,98

Tabla 4.2: Resultados de la dereverberación para el caso con ruido aditivo y una sola RIR.

En este caso la WER obtenida usando WPE vuelve a ser menor que la correspondiente al uso de la red LSTM para de-reverberar las señales con ruido aditivo y reverberación. El uso de WPE en este caso permite una WER un 29,07 % menor que el caso LSTM.

4.1.3. Con ruido aditivo y dos RIRs

El tercer caso considera que el canal acústico del ruido aditivo es diferente al de la señal de voz, y por lo tanto modela el caso real de la manera más precisa. En la tabla 4.3 se muestran los resultados para este caso.

Método	**WER (%)**
WPE$\big\{h_s(t) * s(t) + h_n(t) * n(t)\big\}$	27,39
LSTM$\big\{h_s(t) * s(t) + h_n(t) * n(t)\big\}$	38,60

Tabla 4.3: Resultados de la dereverberación para el caso con ruido aditivo y dos RIRs.

El uso de WPE para de-reverberar las señales otorga una WER un 29,04 % menor relativo al uso de la red LSTM, por lo que en este caso los resultados también son favorables para el caso del método WPE.

Considerando que en los tres casos el uso de la técnica WPE otorga resultados muy superiores a los que se obtienen usando la red LSTM, se hace evidente que la primera presenta un mejor desempeño que la segunda.

4.2. Reducción de ruido

En esta sección se presentan resultados usando las dos primeras métricas descritas en la sección 3.5: Error relativo (ε_{rel}) y MSE sobre los *melFB* (ε_{melFB}). A modo de referencia, la tabla 4.4 muestra las tasas de error aplicadas a las señales sin procesar, es decir, $B(\omega)$.

Entrada	**Referencia**	ε_{rel} (%)	ε_{melFB}
$\hat{B}(\omega)$	$\hat{S}(\omega)$	111,30	0.259772
$\hat{B}(\omega)$	$g_s \cdot \hat{S}(\omega)$	99,01	0.198389

Tabla 4.4: Tabla con errores de referencia.

A continuación se mostrarán los resultados del sistema de reducción de ruido para las distintas normalizaciones utilizadas.

4.2.1. Normalización por frame

En la tabla 4.5 se muestran las tasas de error obtenidas (error relativo, ε_{rel}; y MSE sobre *MelFB*, ε_{melFB}) aplicando la normalización correspondiente a cada *frame* de las señales de entrada y referencia del sistema, para las dos formas de entrenamiento descritas en la sección 3.4 (usando $\hat{S}(\omega)$ o $g_s \cdot \hat{S}(\omega)$ como referencia).

Normalización	**Entrada**	**Referencia**	ε_{rel} (%)	ε_{melFB}
Potencia	$\hat{B}(\omega)$	$\hat{S}(\omega)$	104,10	**0,161697**
	$\hat{B}(\omega)$	$g_s \cdot \hat{S}(\omega)$	93,53	**0,115937**
Log-MVN	MVN($\log(B)$)	MVN($\log(S)$)	**63,59**	0,180346
	MVN($\log(B)$)	MVN($\log(g_s \cdot S)$)	**52,45**	0,127841

Tabla 4.5: Resultados usando normalización por frame

Como se puede ver, usando la normalización Log-MVN se obtiene un menor error relativo. Sin embargo, al usar la métrica del MSE sobre los *MelFB*, los resultados son mejores normalizando por potencia, por lo que no queda claro cuál de las dos formas de normalización es más conveniente.

4.2.2. Normalización por bin

Al igual que en la sección anterior, en la tabla 4.6 se muestran las tasas de error obtenidas, pero esta vez aplicando las normalizaciones correspondientes a cada *bin* de la señal.

En este caso, la normalización Log-MVN se traduce en errores considerablemente menores respecto a la normalización por potencia (tanto ε_{rel} como ε_{melFB}). Es importante señalar que estas tasas de error son también menores que los casos correspondientes a la normalización por *frame*, por lo que es posible afirmar que la normalización Log-MVN por *bin* es la que entrega los mejores resultados.

Para comprender por qué la normalización por *bin*, particularmente usando Log-MVN, funciona mejor que al hacerlo por *frame*, es conveniente observar los *features* luego de ser

Normalización	**Entrada**	**Referencia**	ε_{rel} (%)	ε_{melFB}
Potencia	$\hat{B}(\omega)$	$\hat{S}(\omega)$	64,98	0,127816
	$\hat{B}(\omega)$	$g_s \cdot \hat{S}(\omega)$	56,69	0,121607
Log-MVN	MVN($\log(B)$)	MVN($\log(S)$)	**33,93**	**0,092597**
	MVN($\log(B)$)	MVN($\log(g_s \cdot S)$)	**33,04**	**0,089341**

Tabla 4.6: Resultados usando normalización por bin

normalizados mediante un método u otro. En la figura 4.1 se muestra un mismo *feature* luego de ser normalizado en potencia por *frame* y por *bin*. Como se puede ver, la normalización por *bin* permite que las frecuencias bajas (primeros bines) se muevan en un rango similar a las más altas, mientras que al normalizar por *frame* las frecuencias altas se mueven en un que está rango por debajo de las más frecuencias bajas.

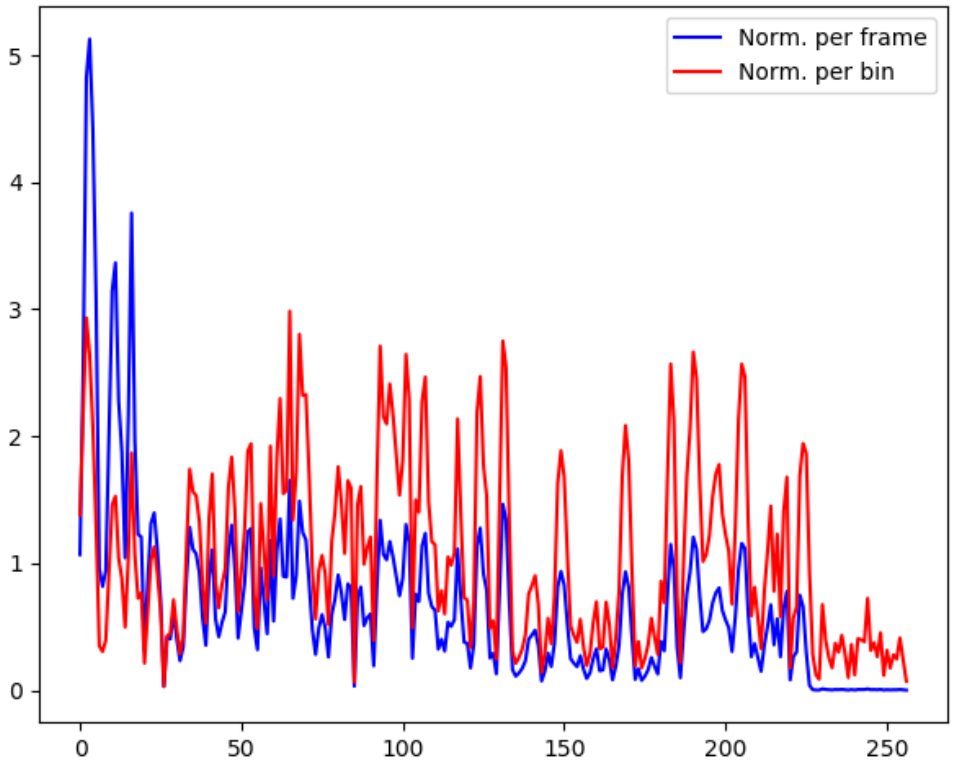

Figura 4.1: Vector de *features* normalizado con potencia por *frame* (azul) y por *bin* (rojo).

Si normalizamos con Log-MVN, el efecto es muy similar. Como se puede observar en la figura 4.2, al normalizar por *bin* se atenúan los bines correspondientes a las frecuencias más bajas, dejando todos los bines en un rango similar.

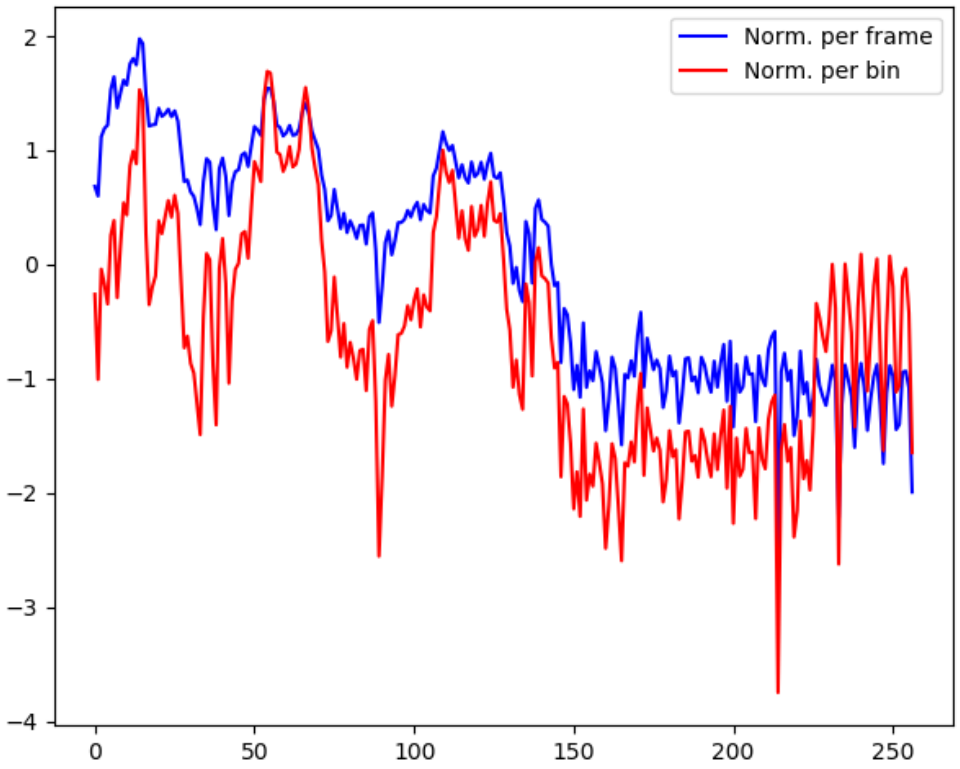

Figura 4.2: Vector de *features* normalizado con Log-MVN por *frame* (azul) y por *bin* (rojo).

Como la normalización por *bin* permite que todos los *features* de un mismo *frame* tomen valores con rangos similares, la información contenida en las altas frecuencias adquiere una mayor resolución, y la contenida en las bajas frecuencias disminuye su resolución. Por el contrario, al normalizar por *frame*, los valores de las altas frecuencias se mantienen muy pequeños, y por lo tanto pierden resolución respecto a las bajas frecuencias. De esta manera, si el entrenamiento de la DNN se hace con los *features* normalizados por *bin*, la información de las altas frecuencias tendrá la misma resolución que la información de las frecuencias bajas, lo cual se traduce en un sistema más preciso al momento de estimar la señal limpia, ya que actuará sobre todo el espectro en vez de focalizarse en las frecuencias bajas.

4.3. Experimentos con ASR

Dado que la finalidad de este trabajo es mejorar el desempeño del ASR, la métrica que adquiere mayor relevancia es la WER de los *decodes*. Para hacer esto, se usa el sistema ASR descrito en la sección 3.5.3. A modo de referencia, la WER obtenida con las señales sin procesar es de 15,9 %, y con la señal limpia es de 2,19 %, tal como se muerta en la tabla 4.7.

Luego de hacer la reducción de ruido, las WERs deben ser más bajas que el caso de las

Señales	**WER(%)**
Limpias	2,19
Con ruido	15,90

Tabla 4.7: WERs de referencia.

señales con ruido (estas son las que entran al sistema) y más altas que el caso de las señales limpias (sin ruido).

En la tabla 4.8 se muestran las WERs obtenidas a realizar *decodes* sobre las señales obtenidas luego de la cancelación de ruido, nuevamente bajo las distintas normalizaciones.

Normalización		**WER(%)**
Por frame	Potencia	7,30
	Log-MVN	12,78
Por bin	Potencia	7,14
	Log-MVN	**5,36**

Tabla 4.8: Resultados de los *decodes* usando las señales obtenidas del sistema de cancelación de ruido.

Como se puede ver, la normalización Log-MVN por *bin* vuelve a ser la que da los mejores resultados, permitiendo una mejora relativa del 66,29 % respecto a la señal de entrada.

4.4. Relación entre SNR y errores de estimación

Por último, como resultado anexo al trabajo, luego de realizar la cancelación de ruido de los datos se procedió a representar la relación entre el SNR de cada *frame* y el error que se obtiene mediante una nube de puntos. El error para cada *frame* se calculó como el MSE entre el *MelFB* del *frame* limpio y el *frame* estimado. Una vez obtenidos los errores para cada *frame*, se generó la nube de puntos que se muestra en la figura 4.3, donde cada punto representa un *frame*. El eje de las abscisas representa el SNR del *frame* y el eje de las ordenadas izquierdo su error de estimación. Además se muestra un histograma con la distribución de los SNRs de los *frames*.

Como se puede ver, el error de estimación se mueve en un rango acotado (la mayor parte entre 0 y 20, con algunas excepciones) y no hay una tendencia clara de su comportamiento respecto al SNR. Se ve un aumento en el error para los SNRs cercanos a 0 [dB], aunque a la vez se puede notar en el histograma que la cantidad de *frames* en dicha zona va en aumento, por lo que es razonable que algunos de estos *frames* estén fuera del rango. Luego de los 0 [dB], a medida que aumenta el SNR, los errores comienzan a descender, lo cual también es

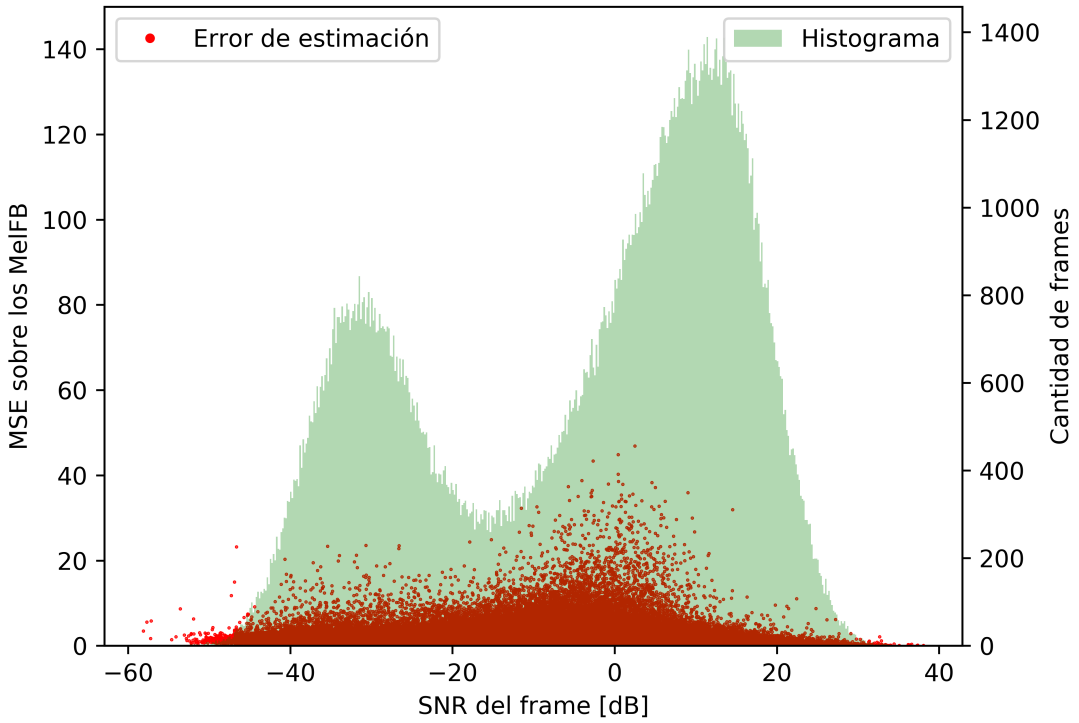

Figura 4.3: Nube de puntos con la relación entre el SNR de cada *frame* y su respectivo error de estimación considerando la señal limpia como objetivo.

de esperar ya que cuando el SNR es mayor, la señal de entrada se asemeja más a la señal limpia, y por tanto la estimación se vuelve más sencilla.

Por último, en la figura 4.4 se agrega el error de la señal de entrada al mismo gráfico. Se puede observar cómo el sistema de cancelación de ruido efectivamente disminuye el error de cada *frame* respecto a la referencia, y es tan efectivo en los *frames* con altos niveles de ruido (SNR bajo) como en aquellos donde la señal es más fuerte (SNR alto).

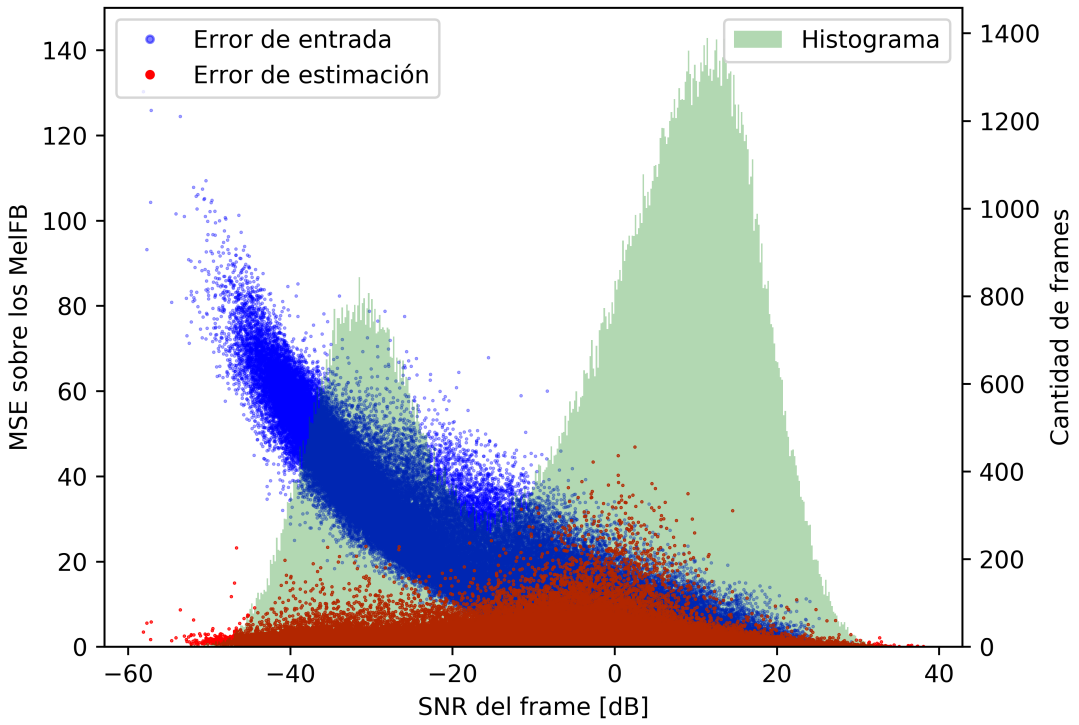

Figura 4.4: Nube de puntos con la relación entre el SNR de cada *frame* y el error de estimación del mismo, para la señal de entrada (en azul) y la estimación (en rojo).

Capítulo 5

Conclusiones

En esta Memoria de Título se utilizó un robot PR2 para estudiar e implementar múltiples técnicas de procesamiento de audio y voz. Con esto se esperaba realzar las señales de audio con el fin de obtener tasas de error más bajas al realizar reconocimiento automático de voz. Este realce se traduce en reducir el ruido y la reverberación, ambos factores que perjudican significativamente el desempeño de los reconocedores de voz. Para esto, se implementaron distintos sistemas en escenarios de interacción humano robot.

En primer lugar, se implementó la técnica Delay and Sum (beamforming) sobre una base de datos de RIRs grabada con el robot PR2. La dirección a la que debía apuntar el *beamforming* fue entregada al algoritmo como dato, asumiendo que esta se podía obtener procesando la información visual entregada por las cámaras del robot.

A continuación se atacó el problema de la reverberación mediante la implementación de dos técnicas: WPE y una red neuronal LSTM. La primera se implementó directamente en MATLAB con el código entregado por sus autores. La segunda se implementó en *Python* usando la librería *Tensorflow*, en base a una red propuesta en un artículo por Wang et al. Si bien los resultados de dicho artículo sugerían que este sistema superaba el desempeño del método WPE (usando métricas de percepción, las cuales son subjetivas), al implementarla y usar una métrica objetiva (WER) el desempeño obtenido fue más bajo, por lo que se concluye que WPE es más efectivo, al menos para el reconocimiento automático de voz.

Una vez resuelto el problema de la reverberación, el siguiente paso fue reducir los efectos del ruido aditivo. Para esto se implementó, también en *Python* con *Tensorflow*, una DNN con arquitectura *feedforward*. Se probaron diferentes formas de normalización, aplicadas tanto a nivel de *frames* como a la trayectoria de los *bins*. Con esto se obtuvo que la normalización MVN (media cero y varianza unitaria) sobre el logaritmo de los datos, aplicada por *bin*, presentaba los mejores resultados, reduciendo el error cuadrático medio en más del 55 % respecto a la señal original.

Para comprobar la efectividad del sistema de reducción de ruido, se realizaron experimentos de reconocimiento automático de voz con un sistema ASR del estado del arte. La WER obtenida luego de hacer la reducción de ruido usando la normalización Log-MVN por bin disminuyó significativamente respecto al error que se obtenía sin tratar el ruido.

Luego de notar que al normalizar por *bin* (en particular mediante Log-MVN) se obtenían mejores resultados que al hacerlo por *frame*, se graficaron los *features* normalizados de ambas maneras, y se observó que la normalización por *bin* otorga un equilibrio en la resolución de todas las frecuencias, mientras que al normalizar por *frame* la resolución se concentra en las frecuencias bajas. Por lo tanto, la normalización por *bin* permite tomar en cuenta el espectro completo al momento de realizar la estimación de la señal limpia en lugar de privilegiar las frecuencias más graves.

Por último, se analizó el efecto del ruido en la estimación de la señal limpia. Se observó que el sistema es robusto frente a los SNRs bajos, ya que el error de estimación se mantiene en el mismo rango que para el caso de las señales con SNRs altos.

Queda propuesto como trabajo futuro la integración de todas las técnicas presentadas en este trabajo (*beamforming* con información visual, WPE para reducción de reverberación y DNN *feedforward* para reducción de ruido), y su aplicación sobre una base de datos grabada en una situación real de interacción humano robot.

Capítulo 6

Bibliografía

[1] Ossama Abdel-Hamid, Abdel-rahman Mohamed, Hui Jiang, Li Deng, Gerald Penn, and Dong Yu. Convolutional neural networks for speech recognition. *IEEE/ACM Transactions on audio, speech, and language processing*, 22(10):1533–1545, 2014.

[2] Hervé Abdi. Discriminant correspondence analysis. *Encyclopedia of measurement and statistics*, pages 270–275, 2007.

[3] Amr Ahmed, Kai Yu, Wei Xu, Yihong Gong, and Eric Xing. Training hierarchical feed-forward visual recognition models using transfer learning from pseudo-tasks. In *European Conference on Computer Vision*, pages 69–82. Springer, 2008.

[4] LR Bahl. Language-model/acoustic channel balance mechanism. *IBM Technical Disclosure Bulletin*, 23(7):3464–3465, 1980.

[5] Yoshua Bengio et al. Learning deep architectures for ai. *Foundations and trends® in Machine Learning*, 2(1):1–127, 2009.

[6] Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy layer-wise training of deep networks. In *Advances in neural information processing systems*, pages 153–160, 2007.

[7] Yoshua Bengio, Yann LeCun, et al. Scaling learning algorithms towards ai. *Large-scale kernel machines*, 34(5):1–41, 2007.

[8] Stanley F Chen, Douglas Beeferman, and Ronald Rosenfeld. Evaluation metrics for language models. In *DARPA Broadcast News Transcription and Understanding Workshop*, pages 275–280. Citeseer, 1998.

[9] M Chetouani, B Gas, JL Zarader, and C Chavy. Discriminative training for neural predictive coding applied to speech features extraction. In *Proceedings of the 2002 International Joint Conference on Neural Networks. IJCNN'02 (Cat. No. 02CH37290)*, volume 1, pages 852–857. IEEE, 2002.

[10] Chung-Cheng Chiu, Tara N Sainath, Yonghui Wu, Rohit Prabhavalkar, Patrick Nguyen, Zhifeng Chen, Anjuli Kannan, Ron J Weiss, Kanishka Rao, Ekaterina Gonina, et al. State-of-the-art speech recognition with sequence-to-sequence models. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4774–4778. IEEE, 2018.

[11] Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM, 2008.

[12] Martin Cooke, Phil Green, Ljubomir Josifovski, and Ascension Vizinho. Robust automatic speech recognition with missing and unreliable acoustic data. *Speech communication*, 34(3):267–285, 2001.

[13] George Cybenko. Approximations by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2:183–192, 1989.

[14] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. R-fcn: Object detection via region-based fully convolutional networks. In *Advances in neural information processing systems*, pages 379–387, 2016.

[15] Namrata Dave. Feature extraction methods lpc, plp and mfcc in speech recognition. *International journal for advance research in engineering and technology*, 1(6):1–4, 2013.

[16] Marc Delcroix, Tomohiro Nakatani, and Shinji Watanabe. Static and dynamic variance compensation for recognition of reverberant speech with dereverberation preprocessing. *IEEE Transactions on Audio, Speech, and Language Processing*, 17(2):324–334, 2009.

[17] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010.

[18] Angelo Farina. Simultaneous measurement of impulse response and distortion with a swept-sine technique. In *Audio Engineering Society Convention 108*. Audio Engineering Society, 2000.

[19] Yoav Freund and David Haussler. Unsupervised learning of distributions on binary vectors using two layer networks. In *Advances in neural information processing systems*, pages 912–919, 1992.

[20] Sadaoki Furui. Speaker-independent isolated word recognition using dynamic features of speech spectrum. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 34(1):52–59, 1986.

[21] Burshtein D. Gannot, S. and E. Weinstein. Signal enhancement using beamforming and nonstationarity with applications to speech. *IEEE Transactions on Signal Processing*, 8(49):1614–1626, 2001.

[22] Jean-Luc Gauvain, Lori Lamel, and Martine Adda-Decker. Developments in continuous

speech dictation using the arpa wsj task. In *1995 International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages 65–68. IEEE, 1995.

[23] Jürgen T Geiger, Felix Weninger, Jort F Gemmeke, Martin Wöllmer, Björn Schuller, and Gerhard Rigoll. Memory-enhanced neural networks and nmf for robust asr. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, 22(6):1037–1046, 2014.

[24] Ritwik Giri, Michael L Seltzer, Jasha Droppo, and Dong Yu. Improving speech recognition in reverberation using a room-aware deep neural network and multi-task learning. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5014–5018. IEEE, 2015.

[25] John J Godfrey and Edward Holliman. Switchboard-1 release 2. *Linguistic Data Consortium, Philadelphia*, 926:927, 1997.

[26] Randy Gomez and Tatsuya Kawahara. Robust speech recognition based on dereverberation parameter optimization using acoustic model likelihood. *IEEE transactions on audio, speech, and language processing*, 18(7):1708–1716, 2010.

[27] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 6645–6649. IEEE, 2013.

[28] Raia Hadsell, Ayse Erkan, Pierre Sermanet, Marco Scoffier, Urs Muller, and Yann LeCun. Deep belief net learning in a long-range vision system for autonomous off-road driving. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 628–633. IEEE, 2008.

[29] Simon Haykin. *Neural networks: a comprehensive foundation*. Prentice Hall PTR, 1994.

[30] Hynek Hermansky and Nelson Morgan. Rasta processing of speech. *IEEE transactions on speech and audio processing*, 2(4):578–589, 1994.

[31] Hynek Hermansky, Nelson Morgan, and H-G Hirsch. Recognition of speech in additive and convolutional noise based on rasta spectral processing. In *1993 IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 2, pages 83–86. IEEE, 1993.

[32] Geoffrey Hinton, Li Deng, Dong Yu, George Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Brian Kingsbury, et al. Deep neural networks for acoustic modeling in speech recognition. *IEEE Signal processing magazine*, 29, 2012.

[33] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

[34] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

[35] Geoffrey E Hinton and Ruslan R Salakhutdinov. Using deep belief nets to learn covariance kernels for gaussian processes. In *Advances in neural information processing systems*, pages 1249–1256, 2008.

[36] Hans-Günter Hirsch and Harald Finster. A new approach for the adaptation of hmms to reverberation and background noise. *Speech Communication*, 50(3):244–263, 2008.

[37] Sepp Hochreiter. Untersuchungen zu dynamischen neuronalen netzen. *Diploma, Technische Universität München*, 91(1), 1991.

[38] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[39] Sansei Hori, Yutaro Ishida, Yuta Kiyama, Yuichiro Tanaka, Yuki Kuroda, Masataka Hisano, Yuto Imamura, Tomotaka Himaki, Yuma Yoshimoto, Yoshiya Aratani, et al. Hibikino-musashi@ home 2017 team description paper. *arXiv preprint arXiv:1711.05457*, 2017.

[40] Biing Hwang Juang and Laurence R Rabiner. Hidden markov models for speech recognition. *Technometrics*, 33(3):251–272, 1991.

[41] Raquel Justo and M Inés Torres. Integration of complex language models in asr and lu systems. *Pattern Analysis and Applications*, 18(3):493–505, 2015.

[42] Chanwoo Kim and Richard M Stern. Nonlinear enhancement of onset for robust speech recognition. In *Eleventh Annual Conference of the International Speech Communication Association*, 2010.

[43] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[44] Keisuke Kinoshita, Marc Delcroix, Sharon Gannot, Emanuël AP Habets, Reinhold Haeb-Umbach, Walter Kellermann, Volker Leutnant, Roland Maas, Tomohiro Nakatani, Bhiksha Raj, et al. A summary of the reverb challenge: state-of-the-art and remaining challenges in reverberant speech processing research. *EURASIP Journal on Advances in Signal Processing*, 2016(1):7, 2016.

[45] Birger Kollmeier, Thomas Brand, and Bernd Meyer. Perception of speech and sound. In *Springer handbook of speech processing*, pages 61–82. Springer, 2008.

[46] Alexander Krueger and Reinhold Haeb-Umbach. Model-based feature enhancement for reverberant speech recognition. *IEEE Transactions on Audio, Speech, and Language Processing*, 18(7):1692–1707, 2010.

[47] Kshitiz Kumar, Rita Singh, Bhiksha Raj, and Richard Stern. Gammatone sub-band magnitude-domain dereverberation for asr. In *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4604–4607. IEEE, 2011.

[48] Kenichi Kumatani, Takayuki Arakawa, Kazumasa Yamamoto, John McDonough,

Bhiksha Raj, Rita Singh, and Ivan Tashev. Microphone array processing for distant speech recognition: Towards real-world deployment. In *Proceedings of The 2012 Asia Pacific Signal and Information Processing Association Annual Summit and Conference*, pages 1–10. IEEE, 2012.

[49] Heinrich Kuttruff. Room acoustics. London & New York, 2009.

[50] NTT Communication Science Laboratories. WPE MATLAB toolkit official website, 2019.

[51] Hugo Larochelle, Dumitru Erhan, Aaron Courville, James Bergstra, and Yoshua Bengio. An empirical evaluation of deep architectures on problems with many factors of variation. In *Proceedings of the 24th international conference on Machine learning*, pages 473–480. ACM, 2007.

[52] Volker Leutnant, Alexander Krueger, and Reinhold Haeb-Umbach. Bayesian feature enhancement for reverberation and noise robust speech recognition. *IEEE Transactions on Audio, Speech, and Language Processing*, 21(8):1640–1652, 2013.

[53] Ilya Levner. Data driven object segmentation. 2009.

[54] Bo Li, Tara N Sainath, Arun Narayanan, Joe Caroselli, Michiel Bacchiani, Ananya Misra, Izhak Shafran, Hasim Sak, Golan Pundak, Kean K Chin, et al. Acoustic modeling for google home. In *Interspeech*, pages 399–403, 2017.

[55] Jinyu Li, Li Deng, Reinhold Haeb-Umbach, and Yifan Gong. *Robust automatic speech recognition: a bridge to practical applications*. Academic Press, 2015.

[56] Jinyu Li, Abdelrahman Mohamed, Geoffrey Zweig, and Yifan Gong. Lstm time and frequency recurrence for automatic speech recognition. In *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pages 187–191. IEEE, 2015.

[57] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017.

[58] Fu-Hua Liu, Richard M Stern, Xuedong Huang, and Alejandro Acero. Efficient cepstral normalization for robust speech recognition. In *Proceedings of the workshop on Human Language Technology*, pages 69–74. Association for Computational Linguistics, 1993.

[59] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

[60] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

[61] Yuzong Liu and Katrin Kirchhoff. Novel front-end features based on neural graph

embeddings for dnn-hmm and lstm-ctc acoustic modeling. In *INTERSPEECH*, pages 793–797, 2016.

[62] Huimin Lu, Tomoki Uemura, Dong Wang, Jihua Zhu, Zi Huang, and Hyoungseop Kim. Deep-sea organisms tracking using dehazing and deep learning. *Mobile Networks and Applications*, pages 1–8, 2018.

[63] Xugang Lu, Masashi Unoki, and Satoshi Nakamura. Sub-band temporal modulation envelopes and their normalization for automatic speech recognition in reverberant environments. *Computer Speech & Language*, 25(3):571–584, 2011.

[64] Mauricio Matamoros, SEIB Viktor, and Dietrich Paulus. Trends, challenges and adopted strategies in robocup@ home. In *2019 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, pages 1–6. IEEE, 2019.

[65] U. Michel. History of acoustic beamforming. *Berlin Beamforming Conference*, pages 21–22, November 2006.

[66] Vikramjit Mitra and Horacio Franco. Coping with unseen data conditions: Investigating neural net architectures, robust features, and information fusion for robust speech recognition. In *INTERSPEECH*, pages 3783–3787, 2016.

[67] Vikramjit Mitra, Wen Wang, and Horacio Franco. Deep convolutional nets and robust features for reverberation-robust speech recognition. In *2014 IEEE Spoken Language Technology Workshop (SLT)*, pages 548–553. IEEE, 2014.

[68] Hossein Mobahi, Ronan Collobert, and Jason Weston. Deep learning from temporal coherence in video. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 737–744. ACM, 2009.

[69] S Mohsen Naqvi, Wenwu Wang, M Salman Khan, Mark Barnard, and Jonathon A Chambers. Multimodal (audio–visual) source separation exploiting multi-speaker tracking, robust beamforming and time–frequency masking. *IET Signal Processing*, 6(5):466–477, 2012.

[70] Syed Mohsen Naqvi, Miao Yu, and Jonathon A Chambers. A multimodal approach to blind source separation of moving sources. *IEEE Journal of Selected Topics in Signal Processing*, 4(5):895–910, 2010.

[71] Simon Osindero and Geoffrey E Hinton. Modeling image patches with a directed hierarchy of markov random fields. In *Advances in neural information processing systems*, pages 1121–1128, 2008.

[72] Kalle J Palomäki, Guy J Brown, and Jon P Barker. Techniques for handling convolutional distortion withmissing data'automatic speech recognition. *Speech Communication*, 43(1-2):123–142, 2004.

[73] David Pearce and J Picone. Aurora working group: Dsr front end lvcsr evaluation au/384/02. *Inst. for Signal & Inform. Process., Mississippi State Univ., Tech. Rep*,

2002.

[74] Rico Petrick, Kevin Lohde, Matthias Wolff, and Rüdiger Hoffmann. The harming part of room acoustics in automatic speech recognition. In *Eighth Annual Conference of the International Speech Communication Association*, 2007.

[75] Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, et al. The kaldi speech recognition toolkit. In *IEEE 2011 workshop on automatic speech recognition and understanding*, number CONF. IEEE Signal Processing Society, 2011.

[76] Yanmin Qian, Mengxiao Bi, Tian Tan, and Kai Yu. Very deep convolutional neural networks for noise robust speech recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 24(12):2263–2276, 2016.

[77] Bhiksha Raj, Michael L Seltzer, and Richard M Stern. Reconstruction of missing features for robust speech recognition. *Speech communication*, 43(4):275–296, 2004.

[78] Marc'Aurelio Ranzato, Christopher Poultney, Sumit Chopra, and Yann L Cun. Efficient learning of sparse representations with an energy-based model. In *Advances in neural information processing systems*, pages 1137–1144, 2007.

[79] Marc'Aurelio Ranzato and Martin Szummer. Semi-supervised learning of compact document representations with deep networks. In *Proceedings of the 25th international conference on Machine learning*, pages 792–799. ACM, 2008.

[80] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv*, 2018.

[81] Frank Rosenblatt. Principles of neurodynamics. perceptrons and the theory of brain mechanisms. Technical report, Cornell Aeronautical Lab Inc Buffalo NY, 1961.

[82] David E Rumelhart, Geoffrey E Hinton, Ronald J Williams, et al. Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1, 1988.

[83] Tara N Sainath and Bo Li. Modeling time-frequency patterns with lstm vs. convolutional architectures for lvcsr tasks. 2016.

[84] Ruslan Salakhutdinov, Andriy Mnih, and Geoffrey Hinton. Restricted boltzmann machines for collaborative filtering. In *Proceedings of the 24th international conference on Machine learning*, pages 791–798. ACM, 2007.

[85] George Saon, Hong-Kwang J Kuo, Steven Rennie, and Michael Picheny. The ibm 2015 english conversational telephone speech recognition system. *arXiv preprint arXiv:1505.05899*, 2015.

[86] Jens Schröder, Jörn Anemüller, and Stefan Goetze. Performance comparison of gmm, hmm and dnn based approaches for acoustic event detection within task 3 of the dcase 2016 challenge. In *Proc. Workshop Detect. Classification Acoust. Scenes Events*, pages 80–84, 2016.

[87] Armin Sehr, Roland Maas, and Walter Kellermann. Reverberation model-based decoding in the logmelspec domain for robust distant-talking speech recognition. *IEEE transactions on audio, speech, and language processing*, 18(7):1676–1691, 2010.

[88] Zhiyuan Tang, Dong Wang, and Zhiyong Zhang. Recurrent neural network training with dark knowledge transfer. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5900–5904. IEEE, 2016.

[89] Alexandros Tsilfidis, Iosif Mporas, John Mourjopoulos, and Nikos Fakotakis. Automatic speech recognition performance in different room acoustic environments with and without dereverberation preprocessing. *Computer Speech & Language*, 27(1):380–395, 2013.

[90] Paul E Utgoff and David J Stracuzzi. Many-layered learning. *Neural Computation*, 14(10):2497–2529, 2002.

[91] Karel Veselỳ, Arnab Ghoshal, Lukás Burget, and Daniel Povey. Sequence-discriminative training of deep neural networks. In *Interspeech*, volume 2013, pages 2345–2349, 2013.

[92] D. B. Ward and R. C. Williamson. Particle filter beamforming for acoustic source localization in a reverberant environment. *2002 IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2:II–1777, May 2002.

[93] Chao Weng, Dong Yu, Michael L Seltzer, and Jasha Droppo. Single-channel mixed speech recognition using deep neural networks. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5632–5636. IEEE, 2014.

[94] Jason Weston, Frédéric Ratle, Hossein Mobahi, and Ronan Collobert. Deep learning via semi-supervised embedding. In *Neural Networks: Tricks of the Trade*, pages 639–655. Springer, 2012.

[95] Wayne Xiong, Lingfeng Wu, Fil Alleva, Jasha Droppo, Xuedong Huang, and Andreas Stolcke. The microsoft 2017 conversational speech recognition system. In *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 5934–5938. IEEE, 2018.

[96] N. B. Yoma. Laboratorio de procesamiento y transmisión de la voz (LPTV), 2019.

[97] Takuya Yoshioka, Xie Chen, and Mark JF Gales. Impact of single-microphone dereverberation on dnn-based meeting transcription systems. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5527–5531. IEEE, 2014.

[98] Takuya Yoshioka, Tomohiro Nakatani, Masato Miyoshi, and Hiroshi G Okuno. Blind separation and dereverberation of speech mixtures by joint optimization. *IEEE Transactions on Audio, Speech, and Language Processing*, 19(1):69–84, 2010.

[99] Steve Young. Hmms and related speech recognition technologies. In *Springer Handbook of Speech Processing*, pages 539–558. Springer, 2008.

[100] Dong Yu, Wayne Xiong, Jasha Droppo, Andreas Stolcke, Guoli Ye, Jinyu Li, and Geoffrey Zweig. Deep convolutional neural networks with layer-wise context expansion and attention. In *Interspeech*, pages 17–21, 2016.

[101] Wang D. Xu B. & Zhang T. Zhao, Y. Late reverberation suppression using recurrent neural networks with long short-term memory. *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5434–5438, April 2018.