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

SEPARACIÓN DE FUENTES DE AUDIO CON DEEP LEARNING PARA LA INTERACCIÓN HUMANO-ROBOT

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO ELÉCTRICO

NICOLÁS ALBERTO JIMÉNEZ BOHMER

PROFESOR GUÍA:
Nestor Becerra Yoma

MIEMBROS DE LA COMISIÓN:
Francisco Rivera Serrano
Rodrigo Mahu Sinclair

Este trabajo ha sido parcialmente financiado por:
Proyecto Fondecyt 1211946

SANTIAGO DE CHILE
2022

RESUMEN DE MEMORIA PARA OPTAR
AL TITULO DE INGENIERO ELÉCTRICO
POR: NICOLÁS JIMÉNEZ BOHMER
FECHA: 2022
PROF. GUÍA: NESTOR BECERRA YOMA

SEPARACIÓN DE FUENTES DE AUDIO CON DEEP LEARNING PARA LA INTERACCIÓN HUMANO-ROBOT

En los últimos años se ha notado un aumento de interés en interacciones entre humano y robot, o abreviado al inglés HRI (*Human-Robot interaction*). Cada vez se hace más necesario desarrollar robots que sean capaces de comunicarse de manera efectiva para el entendimiento/interacción con un humano. Es por esto, que hay una necesidad hoy en día de desarrollar *software* para que los robots puedan procesar la información que llega desde la contra parte humana. La manera más directa de comunicarse es usando la voz, por la cual, es necesario realizar modelos para el procesamiento de señales.

En el presente trabajo se lleva a cabo el desarrollo de un modelo capaz de procesar señales acústicas de escenarios dinámicos en el tiempo en donde el robot y/o usuario pueden estarse moviendo con presencia de ruido y reverberación. Los modelos que se desarrollaron en el presente trabajo de título son usando la herramienta de *deep learning*. Dicha herramienta es sumamente popular debido a la gran capacidad que tiene de generar modelos que aprenden de grandes cantidades de datos.

La resolución del problema se separa en distintas tareas. El objetivo principal es separar la señal objetivo (la voz del usuario) de la interferencia ambiental. Para ello, se deben generar algoritmos de separación de fuentes usando *deep learning* y desarrollar técnicas de *beamforming*. Además, hay que medir la calidad de señal acústica procesada por la red usando un ASR o métricas de calidad de voz.

La metodología utilizada consiste en primero, realizar una replicación de 3 trabajos previos en donde se utilizan determinadas técnicas o modelos *deep learning* para realizar separación de fuentes. Dicha replicación tiene el propósito de tener una base para poder comparar resultados cuando se pruebe el sistema propuesto. Segundo, desarrollar el sistema propuesto, que consiste en adaptar las redes *deep learning* GRNN-BF y ADL-MVDR a una base de datos con escenarios dinámicos reales en el tiempo (reales en el sentido que fueron grabadas en vivo y en directo con el robot y fuentes de *speech* moviéndose, más ruido y reverberación natural).

Los resultados obtenidos demuestran que la red GRNN-BF y la red ADL-MVDR tiene potencial para generar un modelo robusto, capaz de realizar separación de fuentes bajo escenarios dinámicos en el tiempo con movimiento entre el usuario y el robot. Si bien los resultados son prometedores, hay presencia de inconsistencias con las métricas utilizadas para evaluar los modelos. Estas inconsistencias se atribuyen a problemas de generalización de las redes, debido a que hay diferencias en los conjuntos de entrenamiento y test.

Es ist die wichtigste
Kunst des Lehrers, die
Freude am Schaffen und
am Erkennen zu wecken.

Albert Einstein

Tabla de Contenido

Índice de Tablas

Índice de Ilustraciones

Capítulo 1

Introducción

1.1. Identificación y Formulación de Problema

La colaboración Humano-Robot será un componente estratégico en aplicaciones comerciales dentro de los próximos 10 a 20 años. Por consiguiente, robots sociales son uno de los desafíos más críticos en la robótica e ingeniería. Al hablar de robots sociales, se hace referencia a que el robot debe ser capaz de comunicarse de manera efectiva, entendiendo el contexto del usuario y características de él. Bajo lo anterior, es necesario que el robot pueda hacer un perfil del usuario lo más completo posible.

Para poder realizar un perfil de usuario de manera física, cognitiva y social es crucial que el robot sea capaz de observar múltiples entradas que vienen de la contra-parte humana como, por ejemplo, cómo se ve, su tono de voz, latidos del corazón, etc. Sin embargo, algunas de estas entradas que el robot requiere captar, pueden ser invasivos para el usuario (por ejemplo, para captar la señales fisiológicas habría que ponerle sensores al usuario, como lo es medir los latidos del corazón). Otras de estas entradas pueden no estar siempre disponibles, dependiendo de las condiciones, como hacer un procesamiento de imagen que dependen de la luz disponible en el ambiente. Por lo contrario, el lenguaje, la mayoría de la veces, esta disponible como entrada hacia el robot y este conlleva una gran cantidad de información lingüística y paralingüistica para extraer. La información lingüística es el mensaje con las reglas y convenciones gramaticales correspondientes. Por otro lado, la información paralingüistica son los elementos que acompañan y completan la información lingüística como, por ejemplo, el tono de voz, velocidad y risa; permiten extraer información anímica y contextual del usuario.

La voz, además de ser una herramienta para dictar comandos al robot, da paso a extraer la condición física, psicología y emocional del humano. Sin embargo, el procesamiento y análisis de voz es muy sensible al ruido de ambiente (*cocktail party effect*), reverberación y canales acústicos de duración variable, que dan resultado a escenarios dinámicos. Por esto, es necesario abordar el problema de procesamiento de señales, *speech emotion recognition* y *verbal content recognition* en la interacción natural humano-robot. Como interacción natural, entendemos escenarios dinámicamente variables en el tiempo donde uno o más usuarios pueden estar en movimiento y comunicándose con el robot, que también puede estar en movimiento, con presencia de ruido.

1.2. Objetivos del Trabajo de Título

Por lo anterior descrito, la memoria trata sobre interacción humano-robot, donde el robot tiene que ser capaz de identificar al usuario de forma de entender qué dice. Como se mencionó anteriormente, el procesamiento de la voz es muy sensible al ruido, reverberaciones, etc., por lo que la memoria tratará en el desarrollo de técnicas de procesamiento de señales. El objetivo general es el siguiente.

- Separar la señal objetivo de la interferencia ambiental (reverberación/ruido).

Por otra parte los objetivos específicos, son los siguientes.

- Encontrar y/o generar una base de datos de interacciones naturales entre humano-robot.
- Generar algoritmos de *source separation* usando *deep learning*.
- Desarrollo de técnicas de *beamforming*.
- Obtención de la señal objetivo y evaluación de la calidad de esta con un ASR (*automatic speech recognition*) o con métricas de la calidad de voz percibida.

Se nombraron varios términos que serán explicados en el marco teórico. A groso modo, *source separation* es extraer la señal objetivo (a quién debe escuchar el robot) y *beamforming* es para poder ubicar al usuario espacialmente. El *deep learning* será la herramienta para entrenar y crear modelos de *source separation* y (quizás) de *beamforming*.

1.3. Estructura de la memoria

Esta memoria se divide en 5 partes siendo esta el capítulo 1 en donde se realiza una introducción del tema, los objetivos generales y específicos del trabajo de título.

El capítulo 2 contextualiza el problema de separación de fuentes. En dicho capitulo se realiza un marco teórico enmarcando los principales conceptos necesarios para comprender este trabajo. También, se describe el estado del arte analizando soluciones propuestas por otros autores. Se termina dicho capítulo describiendo la contribución de este trabajo de título.

El capítulo 3 describe las técnicas que se desarrollaron en esta memoria, como los modelos neuronales que se utilizaron, preprocesamientos necesarios para la data, métricas de desempeño, etc. También, se describen las bases de datos que se utilizaron.

El capítulo 4 muestra el resultado de los experimentos que se realizaron y se analizan los resultados obtenidos. Se presentan tablas y gráficos de los resultados obtenidos al implementar los modelos para, posteriormente, finalizar con una discusión de los resultados.

El capítulo 5 presenta las conclusiones del trabajo realizado, se discute si se cumplieron los objetivos, si los modelos implementados cumplen los objetivos que se esperaban y formas para mejorar en una propuesta para trabajos futuros.

Capítulo 2

El problema de separación de fuentes y *Deep Learning*

En dicho capítulo se realiza un marco teórico en modo de introducir y contextualizar al lector con los conceptos necesarios para el completo entendimiento de esta memoria. A su vez, se realiza una revisión bibliográfica en donde se enmarca el proyecto de separación de fuentes usando soluciones propuestas por otros autores. El análisis hecho se realizó desde lo global hasta lo especifico. El objetivo de la memoria es poder separar señales, o sea, realizar *source separation* en interacción humano-robot (HRI) con movimiento mediante la voz.

2.1. Marco Teórico

En esta sección se presentarán de manera pertinente los fundamentos teóricos relacionados con el proyecto. Ya en la introducción, se mencionaron varios conceptos que deben sen explicados para un total entendimiento del trabajo.

2.1.1. Interacción humano-robot por voz

La interacción humano-robot (HRI) es sumamente relevante en las situaciones en cuales los robots no son totalmente autónomos y requieren interacción con los usuarios para recibir instrucciones y/o información para tomar decisiones. Por ello, la comunicación entre humano-robot es esencial para una correcta colaboración de las partes. El habla es la forma más directa y natural de comunicarse, por consecuencia, las interacciones entre humano-robot por voz debería ser la forma más natural de entregar información.

El gran problema de la interacción mediante la voz, es que la voz experimenta reverberación (reflexiones dentro de la sala) que ensucian el audio, entorpeciendo la comunicación. Además, se le puede añadir ruido al ambiente, lo cual genera más interferencia. Por esto, las HRI mediante la voz son susceptibles a la interferencia que se produce por la reverberación/ruido.

En especifico, cuando hay presencia de movimiento entre el robot y/o el usuario, la situación es aún más complicada. Cuando hay presencia de movimiento la reverberación ya no se puede tratar analíticamente. Esto genera que el procesamiento de los audios sea aún más complicado, dificultando la comunicación efectiva entre el humano y el robot.

2.1.2. *Deep Learning*

Deep Learning es de la familia de métodos de *machine learning* que está basada en redes neuronales. Estas redes aprenden tareas a través de los datos que son entregados y este aprendizaje de la red puede ser supervisado, semi-supervisado o no supervisado. Las arquitecturas de *deep learning* han sido aplicadas en varias áreas, como procesamiento de imágenes, procesamiento de señales, traductores de lenguaje, medicina, etc., donde *deep learning* ha logrado resultados comparables a un experto o incluso mejor.

Las redes neuronales están basadas en las neuronas humanas. Es un modelo matemático que imita las conexiones dentro del cerebro humano. La unidad básica es el perceptrón (análogo a la neurona humana) y estos se conectan entre si para formar una red neuronal. Cuando las conexiones son muchas y se tienen bastantes capas de perceptrones, la red neuronal pasa a tener el adjetivo "deep". Estas redes neuronales profundas son entrenadas para realizar cierta tarea y en nuestro caso, se utilizarán estas redes para abordar el problema de *source separation*.

Más específicamente, veremos ahora cómo es un perceptrón y cómo es una arquitectura de una red neuronal general. Cuáles son sus partes y cómo se entrena.

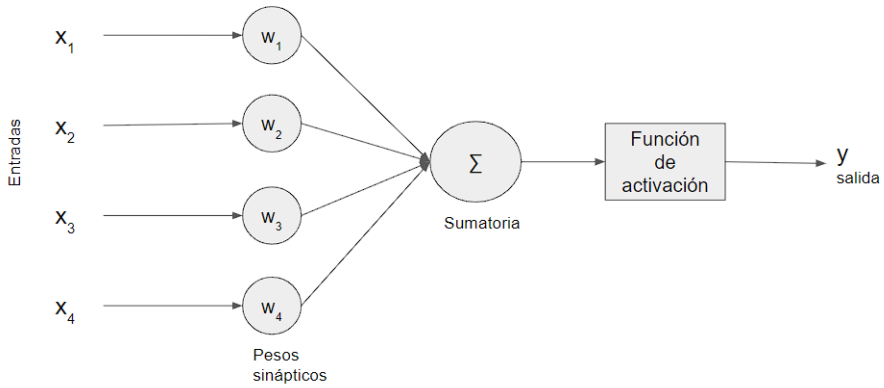

Figura 2.1: Modelo perceptrón.

De la figura 2.1 notamos que el modelo del perceptrón es bien simple; un perceptrón puede tener múltiples entradas (x_n) y estas entradas tienen un peso asociado (w_n). Las entradas se multiplican por sus pesos y se suman. Después, esta suma pasa por alguna función de activación y se obtiene la salida. Existen varias funciones de activación como la tanh, relu, sigmoid, entre otras. Después al conectar muchos perceptrones podemos formar una red, como se muestra en la figura 2.2.

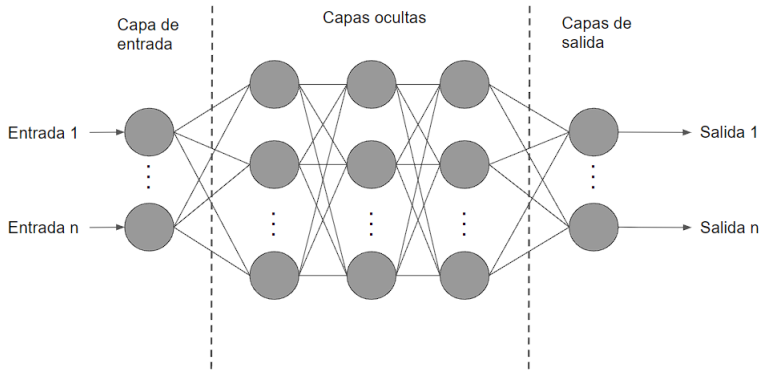

Figura 2.2: Red Neuronal General (Percetrón Multicapa o MLP).

De la figura 2.2, podemos extraer de inmediato que hay un flujo de información. Primero, se parte por la entrada y la información se va procesando perceptrón a perceptrón, donde la salida del perceptrón anterior es la entrada del siguiente. Finalmente, se llega a una salida final que será el resultado para lo que se estaba entrenando la red. El entrenamiento, básicamente, es ocupar un algoritmo de optimización para actualizar los pesos de la red (*backpropagation*). Un ejemplo, para entender como funciona, es una red que debe reconocer si hay un perro en la foto, por esto la entrada será una foto, esta será procesada y al final el output puede ser un 1 si es que hay un perro y un 0 si es que no hay un perro.

Análogamente al ejemplo, se puede tener una red que separe audio mezclado con ruido y otras voces; la red debe ir procesando la información para obtener una salida limpia. Hay varias formas de abordar el problema usando *deep learning*: la entrada puede ser el audio raw (en el tiempo), o se le puede pasar el audio en frecuencia, o también se le puede pasar el audio en forma de espectrograma, etc. Al fin y al cabo, la mayorías de las veces se hace un pre-procesamiento de los datos para entregárselos a la red. Este pre-procesamiento depende de que se quiera hacer con la red, la forma de la entrada, la salida que se quiere, etc.

2.1.2.1. Redes convolucionales

Las redes convolucionales son un tipo de red las cuales ocupan una técnica llamada convolución. En este trabajo de titulo se utilizaron redes neuronales convolucionales, por lo que es necesario explicarlas. Las redes neuronales convolucionales es un tipo de red neuronal que imita el procesamiento visual que realizan los humanos. Estas redes convolucionales consisten de varias capas de filtros convolucionales de N dimensiones. Generalmente, después de un cierto número de capas convolucionales hay una capa de *pooling*, donde el *pooling* son capas que sub-mapean la dimensionalidad. Todo lo anterior mencionado será explicado con más detalle a continuación.

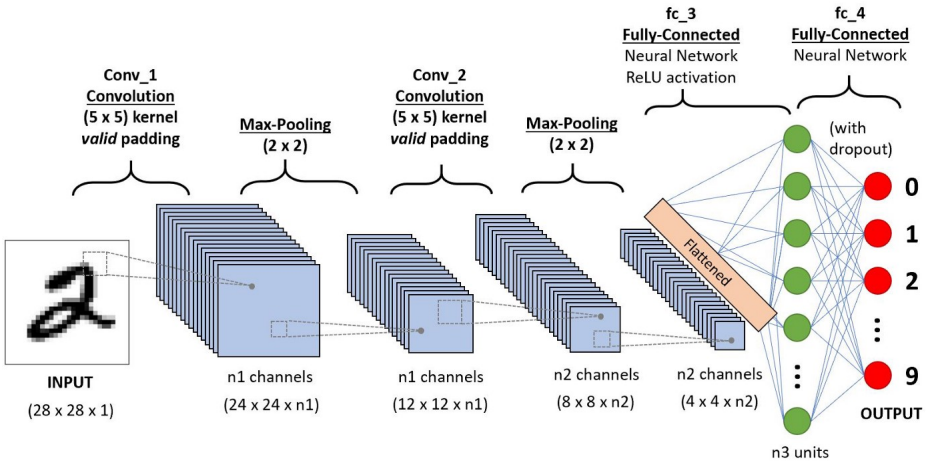

Figura 2.3: Ejemplo de red convolucional, de [39].

Con la figura 2.3 podemos apreciar el comportamiento del procesamiento de imágenes dentro de una red convolucional. Las capas convolucionales contienen neuronas convolucionales, que son las que realizan la labor de extraer características. Al fin y al cabo lo que se tiene es un núcleo (*kernel*) que es una matriz con unos operadores. Cada elemento dentro de la matriz es una neurona convolucional. El núcleo recorre la imagen y realiza la convolución para extraer de la imagen sus características. Así lo que se entrena en estas capas convolucionales son los pesos del *kernel*. La salida de cada neurona convolucional se calcula como:

$$Y_j = g(b_j + \sum_i K_{ij} \bigotimes Y_i) \tag{2.1}$$

Donde Y_j es la neurona j del *kernel* (matriz) se calcula como la combinación lineal de Y_i (neuronas de la capa anterior) con el núcleo de la convolución K_{ij} correspondiente a esa conexión. Se le suma un bias b_j y se le aplica una función de activación g().

Al convolucionar el núcleo con la imagen se obtiene un nuevo vector el cual vuelve ciertas características de la imagen más dominantes. Una vez que la imagen pasa por una capa convolucional, esta se abstrae y a eso se le llama *feature map* o *activation map*. La red entrena el peso de los núcleos de tal forma que pueda detectar el objetivo.

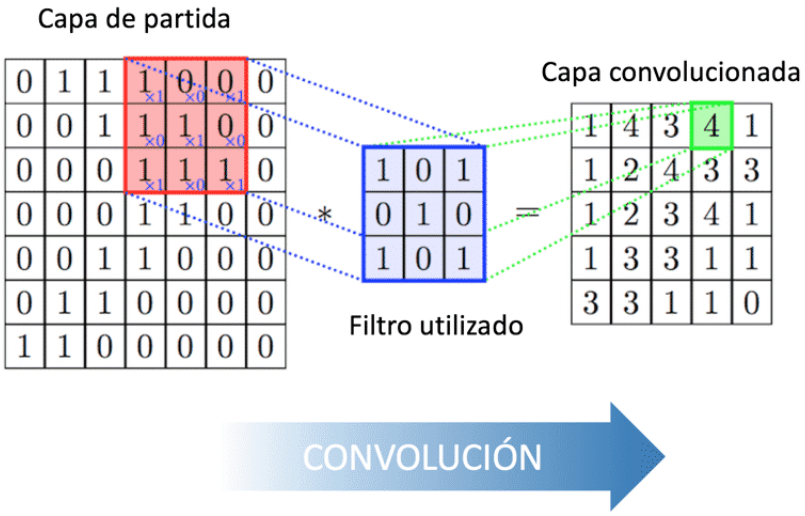

Figura 2.4: Convolución con *kernel*, de [3].

La convolución es una operación matemática que se puede observar gráficamente en la figura 2.4. En simples palabras la convolución es una multiplicación elemento a elemento con una suma.

También en la figura 2.3 podemos notar las capas de *pooling*. Estas capas son meramente para reducir la dimensionalidad del *feature map*. Hay diferentes algoritmos para hacer reducción de dimensionalidad, pero los típicos son tomar el máximo valor de un *cluster*, mínimo valor de un *cluster*, el promedio de los valores de un *cluster*. Los *clusters* son secciones (submatrices) del *feature map* y a eso se le aplica la reducción de dimensionalidad.

Por último, podemos notar que se tiene la capa *fully conected*, esta capa es lo mismo que una red MLP, solo que como entrada tiene un *feature map* y como salida el objetivo para el cual se entrena la red.

Con esto, se puede notar que el flujo de una red neuronal convolucional (CNN) es bien simple: se tiene una serie de capas convolucionales que abstrayen la imagen para poder extraer información de ella y se tiene una serie de capas de *pooling* para hacer reducción de dimensionalidad. Una vez terminado con el procesamiento de la imagen (capas de convolucionales y pooling) se ingresa el *feature map* (imagen abstraída) a una red mlp.

2.1.2.2. Redes recursivas

Las redes neuronales recursivas o RNN (*recurrent neuroral network*) es un tipo de redes neuronales que se especializan en tratar con datos secuenciales. Estas redes, que son una derivación de las *feed foward network* (como la MLP), pueden usar un estado interno llamado memoria para procesar entradas de largo variable. Es por esto que las RNN son ampliamente utilizadas en tareas como procesamiento de texto o de voz.

Estas redes son llamadas recurrentes, debido a que tienen un *loop* de retroalimentación infinito. Lo que hace especial a estas redes es que el almacenaje de memoria esta en total control de la red. Almacenaje de memoria se les llama: puertas de estado o puertas de me-

moria.

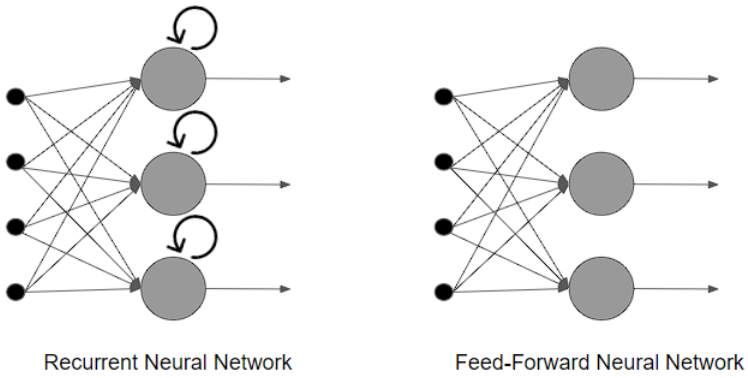

Figura 2.5: Comparación de RNN y *Feed-Forward*.

De la figura 2.5 podemos notar la diferencia entre una red recurrente y una *feed foward*. En las redes *feed foward* (clásica MLP) la información se mueve solo en una dirección, nunca se toca un nodo dos veces. Estas redes no tienen memoria de la entrada que recibieron y son malas prediciendo que es lo que viene a continuación. Las redes *feed foward* solo ven la entrada actual, no tiene conocimiento de otros tiempos. En cambio, las RNN tiene ciclos recursivos, esto hace que el output depende de la entrada actual y también de las entradas pasadas.

Es por esto, que las RNN recibe entradas secuenciales, donde extraen y memorizan las secuencias y patrones para entender la secuencia en si. Con ello estas redes pueden predecir que es lo que viene. Para el caso de procesamiento de señales estas redes son bastante utilizadas debido a que, una señal de audio es una secuencia de datos con un determinado orden y estas redes son capaces de utilizar esta secuencia de datos y entender la relación entre diferentes tiempos de datos.

En especial para el desarrollo de esta memoria se utilizaron redes recurrentes GRU. Las GRU, del ingles *Gated Recurrent Unit* son redes recurrentes parecidas a las LSTM (*Long Short-Term Memory*). Las LSTM son bien conocidas y utilizadas en el mundo académico para distintas tareas, en simples palabras una celda LSTM esta consituida por una puerta de entrada, puerta de olvido y puerta de salida. Las GRU se diferencia de la LSTM debido a que tiene menos parámetros que una LSTM.

2.1.2.3. *Padding*

El *padding* es una técnica usadas en las CNN (redes convolucionales) para darle más espacio al kernel para procesar la imagen. En otras palabras, el *padding* le agrega píxeles exteriores a la imagen en orden de que el kernel tenga más precisión al momento de recorrer la imagen y calcular la operación de convolución. Generalmente, se hace *padding* para que el kernel calce bien en la imagen para realizar la convolución.

Existen varias técnicas de *padding* como, por ejemplo, *zero padding* que agrega ceros al marco exterior de la imagen, *constant padding* que agrega una constante definida por el usuario al marco exterior de la imagen. Como estos ejemplos hay muchos más.

Es necesario mencionar un *padding* que es utilizado en el desarrollo de esta memoria, el cual es el *reflect padding*. *Reflect padding*, simplemente, refleja las filas y columnas en los píxeles exteriores que se agregan. Se puede ver un ejemplo de *reflect padding* en la figura 2.6.

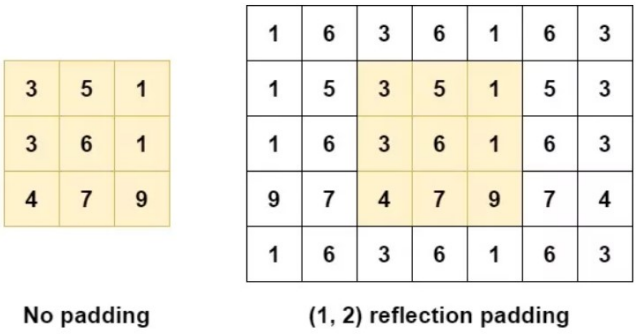

Figura 2.6: *Reflect padding*, de [43].

De la figura 2.6 se aprecia la reflección de la imagen para formar el *padding*. Esta técnica es útil, debido a que asegura que el *output* resultante de la convolución será suave entre el *padding*. Esta también ayuda al modelo, debido a que el padding se ve como la imagen original y no se está agregando información extra.

2.1.2.4. Concatenar

En múltiples ocasiones ocurre que en arquitecturas de *deep learning* se requiere combinar señales. Hay varias técnicas para unir información como lo es la concatenación, *pair-wise multiplication* y *bilinear pooling*. Generalmente, las redes neuronales no son de arquitectura lineal, sino que tienen varias ramas o bucles para después unirse en una salida común. El problema es que cuando una red tiene varias ramas y se necesitan unir en un solo vector para obtener una salida, hay una variedad de técnicas para concatenar/unir vectores.

En esta memoria veremos dos tipos de fusión, concatenación y *compact bilinear pooling*. La concatenación es bien simple: si se tienen dos vectores de una misma dimensión, se pueden unir directamente "pegándolos" uno arriba de otro o uno al lado del otro.

CBP (*compact bilinear pooling*) es una técnica más complicada. CBP realiza un producto externo de los dos vectores a fusionar. Esto provoca múltiples relaciones entre cada elemento de los vectores. De inmediato, se puede pensar que realizar un producto cruz entre vectores aumenta innecesariamente la dimensionalidad del vector. CBP aborda este problema haciendo una reducción de dimensionalidad al realizar un mapeo a un espacio de baja dimensionalidad.

2.1.2.5. *Kernel dilatation*

Como ya se mencionó anteriormente, el *kernel* es la matriz con la cual se desempeña la operación de convolución en una CNN (convolutional neural network). Típicamente, el núcleo es compacto y es barrido por la imagen o *feature map* para extraer características de la imagen operando la convolución. Pero hay variantes del núcleo.

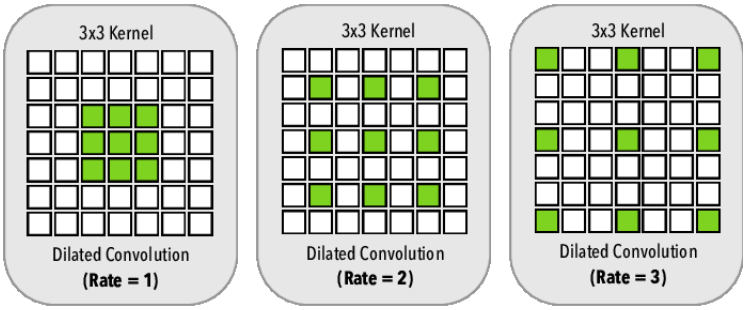

Figura 2.7: *Kernel dilatation*, de [20].

En la figura 2.7 podemos observar, gráficamente, la dilatación del núcleo. El *kernel* de más a la izquierda es un núcleo compactado, el típico que se usa para las CNN. Pero las figuras del medio y de la derecha muestran un *kernel* expandido o dilatado.

La ventaja de usar *kernels* dilatados es que sin aumentar la cantidad de parámetros se puede tener un filtro más largo. Con ello si se tiene una serie de capas convolucionales con una dilatación de kernel progresiva, se podrían ver distintas resoluciones de la imagen para extraer distintos *features*.

2.1.2.6. *Skip connection*

Skip connection, o conexiones residuales, es una técnica muy utilizada hoy en día en las redes neuronales. En la redes neuronales se pueden utilizar *skip conection* o *shortcuts* para saltarse algunas capas. El salto de las conexiones residuales puede variar según la arquitectura; puede ser de N saltos de capa. Típicamente, los *shortcuts* son utilizados para saltarse 2 o 3 capas que contienen funciones de activación no lineares y *batch normalization*. En la figura 2.8 se puede notar como funciona una conexión residual saltándose 2 capas.

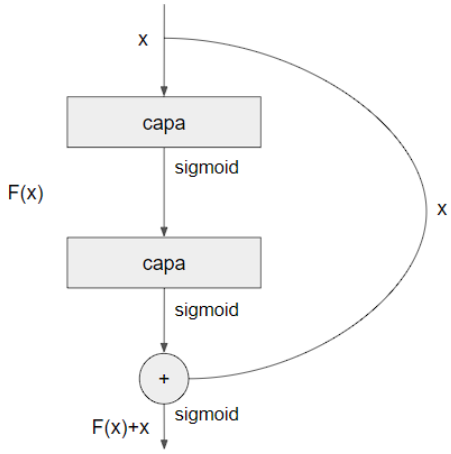

Figura 2.8: *Skip conection.*

Usar *skip conections* tiene principalmente una razón, la cual es mitigar el problema de la degradación del gradiente (*vanishing gradients*). Las redes neuronales profundas tienden a tener altos errores de entrenamiento debido a la degradación del gradiente. Al agregar conexiones residuales el gradiente no se va degradando capa a capa, sino que tiene información del gradiente de capas aun más atrás para ayudar a no degradar el gradiente.

2.1.2.7. Máscaras

Las máscaras o *mask* son utilizadas para realizar segmentación de imágenes. El objetivo principal de la segmentación de imágenes es particionar una imagen digital en secciones. Esto tiene como propósito localizar objetos o delimitadores dentro de la imagen. En simples palabras, un máscara es una matriz que se convoluciona con la imagen para tener como resultado una imagen segmentada, o sea, una imagen con determinadas formas resaltadas.

Esto es muy útil para el reconocimiento de imagen, ya que se pueden entrenar redes que estimen máscaras para poder reconocer formas de una persona, perro, casa, o cualquier objeto. Así, las redes convolucionales puedes estimar máscaras que si esas máscaras se convolucionan con la imagen, el resultado será una imagen con el objeto en cuestión resaltado.

Esto mismo puede ser aplicado a separación de audio. Al fin y al cabo, un espectrograma es una imagen y se pueden entrenar CNN que estimen máscaras para que cuando se convolucionan con el espectrograma, se resaltan las fuentes por separado, obteniendo una separación de audio de forma visual.

2.1.3. *Source Separation*

Source separation, *blind signal separation*, *blind source separation* o separación de fuentes, como dice su nombre, es la separación de señales fuentes a partir de un *set* de señales mezcladas, sin la pérdida de información (o poca pérdida de información). Generalmente, hoy en día se trabaja en la separación digital de señales, donde el objetivo es extraer la señal original de una señal mezclada.

El típico caso para explicar la separación de fuentes es el problema de *cocktail party*. Este es un problema donde hay una cierta cantidad de personas hablando simultáneamente en una habitación y un oyente trata de seguir solo una de las discusiones (focalizarse en una conversación). La mente humana puede resolver este problema, es capaz de separar las fuentes que escucha, pero esto es difícil para el procesamiento en una computadora.

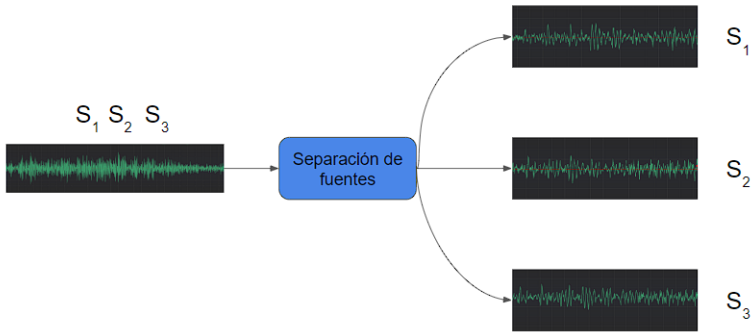

Figura 2.9: Ilustración de *source separation*.

En la figura 2.9 se ilustra como un audio con varias fuentes en su interior es separado logrando extraer fuentes por separado. La figura 2.9 es para el caso de 3 fuentes cualquiera, pero el problema de separación de fuentes puede ser aplicado para N fuentes en cualquier escenario.

Para el caso de esta memoria, se tienen fuentes de ruido y fuentes de *speech*. Por lo que se busca, con el algoritmo de *source separation*, es poder separar el ruido del habla, teniendo así la señal de *speech* aislada. Además de ruido, se experimenta reverberación en la sala, que será explicado más adelante.

Para abordar este problema computacionalmente, existen hoy en día varios algoritmos que lo realizan. Cada algoritmo tiene sus pro y contras en la labor de cómo separa las señales, ya que depende de cómo fue diseñado. En nuestro caso, este problema lo abordaremos utilizando la herramienta de *deep learning*.

2.1.3.1. Reverberación

Típicamente, en *source separation* se debe lidear con la reverberación. La reverberación son rebotes del sonido en el suelo, pared, etc., hasta llegar al receptor. Estos rebotes hacen que el sonido recibido sea distorsionado en el tiempo.

Todos estos rebotes ensucian la señal acústica que le llega al micrófono haciendo que no sea directamente comprensible con lo que se dice en la señal. Por esto mismo, existen técnicas para limpiar la señal y desreverberarla. Hoy en día hay redes neuronales especializadas en realizar esta tarea, pero también existen algoritmos no tan complejos que ayudan a disminuir el efecto de la reverberación.

2.1.3.2. *Singal to Noise Ratio* (SNR) y *Scale Invariant-SNR* (SI-SNR)

El SNR es una métrica de las señales que mide la relación señal ruido. Por sus siglas en ingles (*Singal to noise ratio*) esta métrica mide la relación entre la potencia de la señal versus la potencia del ruido. Se espera que la potencia de la señal supere al ruido, por lo que mientras más SNR mejor.

$$SNR_{dB} = 10log_{10}(\frac{P_{signal}}{P_{noise}}) \tag{2.2}$$

El Si-SNR mide lo mismo que el SNR, solo que se le proyecta a una escala invariante (*scale invariant-SNR*) [25]. Dicha medida calcula la relación entre la potencia de la señal y la potencia del ruido, solo que se pasa la señal estimada a la escala de la señal original. En este contexto se habla de señal estimada a la señal que es propagada por la red neuronal. Es decir, se normaliza la señal estimada por la red a la escala de la señal original. Las ecuaciones de SI-SNR se definen como.

$$\begin{aligned} s_{target} &= \frac{<\overline{s}, s> s}{||s||^2} \\ e_{noise} &= \overline{s} - s_{target} \\ SI-SNR &= 10log_{10}\frac{||s_{target}||^2}{||e_{noise}||^2} \end{aligned} \tag{2.3}$$

Donde s es la señal original y $\overline{s}$ es la señal estimada (limpia) después de pasar por el modelo de *source separation*. El Si-SNR, al fin y al cabo, es una métrica de error de estimación.

2.1.3.3. *Weigth Prediction Error* (WPE)

Weigth prediction error (WPE) es un algoritmo que consiste en un enfoque estadístico de desreverberación del habla [30]. Esta técnica ha demostrado ser muy efectiva y modela el fenómeno de reverberación como un modelo auto-regresivo. WPE ha sido altamente utilizado y desreverba la señal para cada bin de frecuencia en la *Short-Term Fourier Transform* (STFT) de una señal dada usando un filtro linear con dependencia a la frecuencia.

2.1.3.4. *Room Impulse Response* (RIR)

Una respuesta al impulso de un sistema es la salida que se presenta cuando se introduce al sistema la señal impulso. Un impulso en señales y sistemas es un pulso infinitamente corto en el tiempo, por lo que en teoría tiene una punta infinitamente alta. En la práctica es imposible realizar un impulso, ya que no se puede obtener amplitud infinita en un intervalo de tiempo infinitamente pequeño. Matemáticamente, el impulso esta definido como un Delta de Dirac.

Una sala puede ser modelada como un sistema, entonces al aplicarse a ese sistema un Delta de Dirac, se obtendrá la respuesta del impulso de la sala (*Room Impulse Response*).

2.1.3.5. *Direction Of Arrival* (DOA)

El DOA, del ingles *Direction of Arrival*, es la dirección en la que viene la onda acústica al micrófono que captará la señal [2]. En un escenario real, se tienen fuentes de *speech* y fuentes de ruido, tanto como el micrófono que capta las señales. Estos pueden estar distribuidos de cualquier forma y en movimiento, por lo que la dirección en la que la onda de la señal de *speech* llega al micrófono puede formar cualquier ángulo.

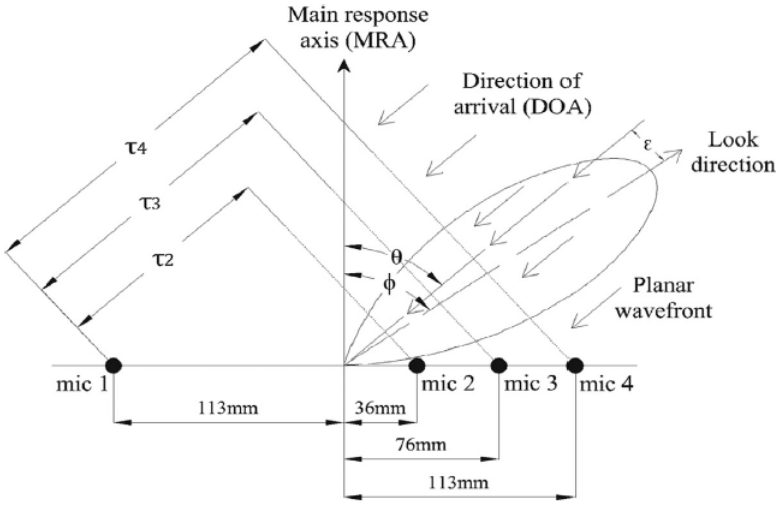

Figura 2.10: Ilustración del DOA [2].

Con la figura 2.10 se puede apreciar el ángulo θ que es el DOA. También se le puede decir AOI (*Angle of Incidence*). Con ello, el DOA es el ángulo que se forma entre la onda plana y el micrófono.

2.1.3.6. *Interchannel Phase Difference* (IPD)

El IPD viene de la sigla *interchannel phase difference*. Esta métrica refleja los cambios en el AOI y a su vez refleja los desfases entre los distintos canales [7]. El IPD se calcula como la diferencia entre los espectrogramas.

$$\phi_{i,tf} = \measuredangle \frac{y_{i,tf}}{y_{1,tf}} \tag{2.4}$$

Donde i es el micrófono (hasta M), tf son los índices frecuencial-temporal de los bines, $y_{i,tf}$ es el espectrograma complejo y $y_{1,tf}$ es el espectrograma del micrófono de referencia. Para el IPD es necesario elegir un micrófono de referencia, con el cual se va a hacer la comparación con los otros canales. Con el IPD se tendrá el ángulo de desfase que existe entre los diferentes canales.

2.1.3.7. *Directional Feature* (DF)

. El *feature* DF (*Directional Feature*) se extrae a través del DOA [7]. Es un cálculo de vectores que se hace usando el DOA. La ecuación del DF tiene la siguiente forma.

$$DF = \sum_{i}^{M} \frac{e_n^{i,f} \frac{y_{i,tf}}{y_{1,tf}}}{\left| e_n^{i,f} \frac{y_{i,tf}}{y_{1,tf}} \right|} \tag{2.5}$$

Donde $e_n^{i,f}$ es el *steering vector* para el *speaker* n en el micrófono i. El *steering vector* se calcula como $[e^{-jw\tau_0}, ..., e^{-jw\tau_{M-1}}]$. Donde M es el numero de canales y τ representa el el tiempo de desfase en llegar la señal a un micrófono con respecto al micrófono de referencia. La ecuación para τ es la siguiente.

$$\tau = \frac{sin(\phi) * \Delta L}{v} \tag{2.6}$$

Donde ϕ son los ángulos de incidencia entre la onda plana y los micrófonos (AOI), ΔL son las distancias entre los distintos micrófonos y el micrófono de referencia y v es la velocidad del sonido. La distancia entre los micrófonos y la velocidad del sonido siempre es conocida, los ángulos de incidencia no necesariamente son conocidos.

2.1.4. *Beamforming*

Beamforming o *spatial filtering*, es una técnica del procesamiento de señales para distinguir las propiedades espaciales de una señal objetivo. Esta técnica es usada, en general, para trackear al usuario dentro del ambiente. Beamforming presenta varias complicaciones, en particular beamforming tiene dos factores claves:

- La señal objetivo proviene de una dirección desconocida.
- No hay información de las señales de interferencia.

En general, lo que hace *beamforming* es hacer que la señal, en determinados ángulos, experimente interferencia constructiva y en otros ángulos destructiva; con ello se logra la selección espacial. *Beamforming* ha sido adoptado por el mundo académico, principalmente porque logra separar (o extraer) el *speech* del ruido de ambiente. *Beamforming*, con señales de audio, básicamente lo que se hace es tener un arreglo de micrófonos formados para crear un filtro espacial, el cual puede extraer la señal de una dirección en específico y reducir la contaminación de la señales en las otras direcciones. Así, en concreto, *beamforming* se logra filtrando las señales de los micrófonos y combinándolas de manera constructiva para extraer la señal deseada y de manera destructiva para rechazar las señales de interferencia.

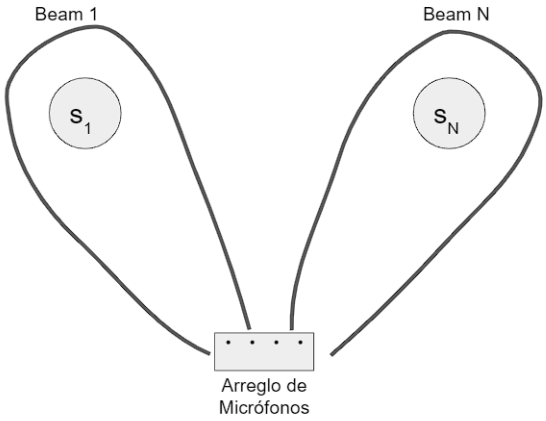

Figura 2.11: Ejemplo de localización de usuarios usando *Beamforming*.

De la figura 2.11 podemos visualizar como es el *beamforming* cuando se esta apuntando a alguna fuente S_n. El *beamforming* apunta hacia la respectiva fuente cuando la fuente que se quiere localizar experimenta interferencia constructiva.

2.1.4.1. *Sum and Delay*

Sum and Delay es un tipo de *bearforming* muy usado. Como ya mencionamos, para realizar *beamforming* se tienen un arreglo de micrófonos que forman un filtro espacial. La señal de audio en reproducción llegará a los micrófonos en diferentes intervalos de tiempos. La señal capturada por cada micrófono son similares entre si, pero contienen diferentes retrasos y fases. Así la señal de cada micrófono es "corrida" al foco correcto. Con ello se alinean las señales (quedan en fase) para después ser sumadas. Una vez sumadas las señales, se hace una normalización de esta por el número de canales de los micrófonos. Gráficamente, el algoritmo se representa en la figura 2.12.

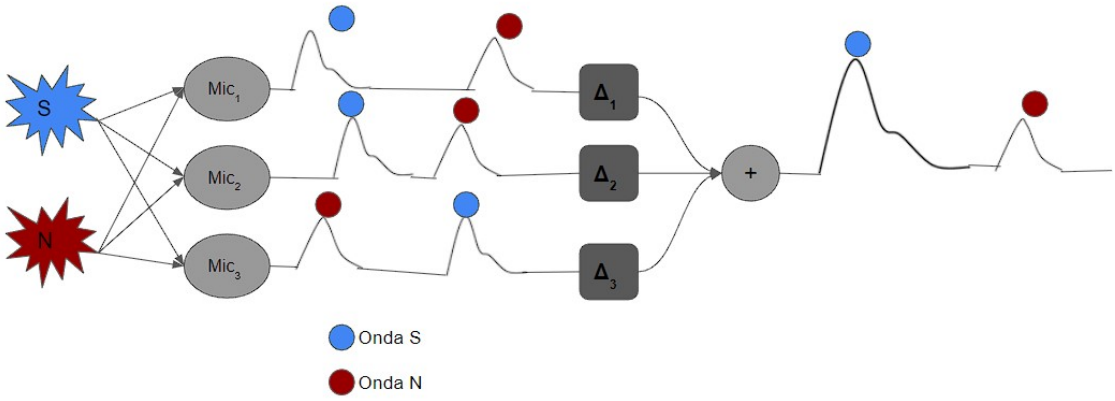

Figura 2.12: Algoritmo de *sum and delay*.

Los desfases que se deben aplicar para realizar este *beamforming* se calculan con 2.6, pero para ello se necesitan los AOI o DOA. Para estimar estos ángulos de incidencia hay 2 opciones. La primera forma es estimar estos ángulos a través de propiedades de las señales

en sí, o sea, solo usar las señales recibidas por los micrófonos para estimar. La segunda forma es tener a disposición el ángulo de incidencia por una cámara, por ejemplo.

2.1.4.2. *Generalized Eigenvector* (GEV)

GEV (*Generalized Eigenvector*) es una técnica de *beamforming* donde el objetivo es maximizar el SNR de cada bin de frecuencia. La optimización de dicho problema lleva al bien conocido problema de los valores propios generalizado. El *beamforming* óptimo es la componente principal generalizada. Un problema es que GEV puede introducir distorsiones arbitrarias.

Para poder calcular los pesos del *beamforming* GEV, se necesita obtener la matriz de covarianza del ruido y la matriz de covarianza del hablante (*speaker*) [49]. La matrices de covarianza de se puede calcular analíticamente o estimarla como, por ejemplo, con un detector de actividad de voz (VAD). Al resolver el problema, se llega a la siguiente solución.

$$w_{GEV} = P(\Phi_N^{-1}\Phi_S) \tag{2.7}$$

Donde Φ_N es la matriz de covarianza del ruido, Φ_S la matriz de covarianza del *speech* y P(.) es aplicar PCA (componentes principales).

2.1.4.3. *Minimum Variance Distortionless Response* (MVDR)

MVDR es una técnica de *beamforming* que en sus siglas en inglés significa respuesta sin distorsión de varianza mínima (*Minimum Variance Distortionless Response*). MVDR se usa para mejorar la cancelación del ruido del *beamforming* mediante supresión adaptativa del ruido espacialmente correlacionado, colocando nulos en las señales de interferencia sin afectar a la ganancia en la dirección de enfoque. MVDR ajusta los pesos del *beamforming* mediante la minimización de la varianza del ruido.

Al igual que en *sum and delay*, el MVDR utiliza la formula 2.6 para calcular los desfases usando el *steering vector*. Para estimar los ángulos de incidencia también hay dos opciones: estimar los ángulos de incidencia mediante la información de las señales recibidas en los micrófonos o tener a disposición los AOI (usar una cámara, por ejemplo).

Para poder calcular los pesos del *beamforming* MVDR, se necesita obtener la matriz de covarianza del ruido y el *steering vector* [49]. La matriz de covarianza de ruido se puede calcular analíticamente o estimarla como, por ejemplo, con un detector de actividad de voz (VAD). El *steering vector*, como se vio en la descripción del *feature* DF, es un vector que presenta los desfases de la señal. Con ello la fórmula para los pesos del MVDR es.

$$w_{MVDR}(f) = \frac{\Phi_N^{-1}v(f)}{v^H(f)\Phi_N^{-1}v(f)} \tag{2.8}$$

Donde Φ_N es la matriz de covarianza del ruido y $v(f)$ es el *steering vector*.

2.1.5. *Automatic Speech Recognition* (ASR)

El reconocimiento de voz es un campo interdisciplinario que combina la informática con lingüística computacional, desarrollando metodología y técnicas que permiten el reconocimiento y traducción del lenguaje hablado a texto por las computadoras. Los ASR son la tecnología que permite a los humanos emplear su voz para hablar con una interfaz computacional. El funcionamiento de un ASR es el siguiente.

1. La señal de audio es separado en fonemas. Los fonemas son las articulación mínimas de un sonido vocálico o consonántico.

2. Cada fonema es una cadena que se une entre si y al analizar esta cadena, el ASR usa probabilidades y análisis para deducir la palabra completa.

3. Ahora que el ASR descifró la palabra, se puede responder o hacer lo que deba realizar.

Así notamos que el ASR es un software capaz de entender las palabras dichas en una señal de audio. Para que el ASR funcione de buena manera, las señales que se le entregan a este deben ser lo más limpias posibles, de tal forma que sea posible capturar el contenido. Por este motivo es importante hacer un preprocesamiento de las señales, como *source separation*, para pasárselo al ASR y que puede este capturar la información lingüística. Hoy en día existen muchos modelos de algoritmo que realizan ASR, en particular, está lo que se llama *Natural Language Processing* (NLP).

Los ASR son modelos complejos que son entrenados para que sean capaces de transcribir señales acústicas a texto. Los ASR cuentan con varios modelos en su interior para calcular las probabilidades de los fonemas y las secuencias de estos para poder hacer la transcripción. Es decir, los ASR son entrenados únicamente para trascribir señales acústicas.

Para esta memoria se utiliza el ASR EBT2 [29]. Dicho ASR es un DNN-HMM [18] usando Kaldi [36]. El estudio propone el uso de un DNN (*Deep Neural Network*) mezclado con HMM (*Hidden Markov Models*). HMM son modelos estadísticos en el cual el objetivo es determinar los parámetros ocultos a partir de los parámetros observables. En este caso se usa HMM para tratar la variabilidad temporal de la voz. La DNN es una red neuronal profunda la cual entrega, para este caso, una pseudo-verosimilitud logarítmica. Si se unen los modelos para formar una DNN-HMM se tiene un modelo estadístico en el cual la probabilidad de un fonema depende de la función de verosimilitud entregada por la DNN.

El ASR EBT2 fue entrenado con señales de *AURORA-4* [35]. Un 25 % de la data se convolucionó con 1 RIR, el 75 % de la data se convolucionó con 32 RIR estáticas y con ruido añadido a un SNR entre 10 y 20 dB; el SNR del ruido está entre -5 y 5 dB. Primero entrenó un GMM-HMM con Kaldi. Los modelos GMM (*Gaussian Mixture Models*) se usan para representar la probabilidad condicional de un evento. Posteriormente, se reemplaza el modelo GMM por una DNN, donde la DNN fue entrenada usando filtros de bancos Mel. Finalmente, el sistema se obtiene re-entrenando la DNN mediante re-entrenamiento discriminativo sMBR [44]. Es de notar que ASR EBT2 del LPTV fue entrenado con señales con interferencia, lo que implica que el ASR es capaz de procesar y trascribir señales con ruido/reveberberación.

2.1.5.1. *Word Error Rate* (WER)

El WER viene de su sigla en inglés *Word Error Rate*. Esta es una métrica que mide el desempeño del reconocimiento del habla. Es difícil medir el rendimiento, debido a que la secuencia de palabras detectada puede tener un largo diferente a la secuencia de palabras de referencia (la correcta). El WER proporciona una métrica para evaluar sistema de traducción de voz, pero el WER no da información de la naturaleza de los errores. El WER es calculado de la siguiente manera.

$$WER = \frac{S + D + I}{N} \tag{2.9}$$

Donde S son el número de sustituciones, D es el número de eliminaciones e I en número de inserciones que se le deben hacer a la secuencia de palabras reconocida para llegar a ser la secuencia de palabra de referencia. N es el número de palabras en la secuencia de referencia.

2.2. Estado del arte

2.2.1. Estudios de separación de fuentes

En esta sección analizaremos las posibles soluciones presentadas anteriormente por otros autores sobre el problema de separación de fuentes. Existen varias soluciones al problema, por lo que las analizaremos, caracterizaremos y se argumentará si solucionan o no el problema.

Las soluciones encontradas se pueden dividir en 2 grupos. El primer grupo es el cual la tarea esta totalmente determinada por el *dataset*. Generalmente, los *dataset* fueron creados para realizar una tarea determinada, por ejemplo, un set de datos que cuenta con diferentes audios de personas hablando. La tarea principal del *dataset* es mezclar dos personas hablando y separar un hablante del otro. Es por esto, que al decir que el primer grupo esta determinada por la tarea del *dataset*, quiere decir que las soluciones planteadas solo evalúan el desempeño del algoritmo usando los objetivos estipulados que presenta el set de datos usado. El segundo grupo son las soluciones que tienen evaluaciones más reales, en donde se utilizaron al menos micrófonos (reales o simulados) para la evaluación del método de *source separation*.

Se ha propuesto el uso de GSS con un ASR para separar fuentes [4]. En dicho trabajo se utiliza el set de datos CHiME-6 [47] para realizar separación de fuentes. Este *dataset* contiene *speech* con cierto solapamiento, ruido y conversaciones naturales. *Guided source separation* (GSS) se basa en modelos gausianos mixtos centrales complejos para separar las señales, combinados con modelos *deep-learning*. La arquitectura del sistema es presentado en la figura 2.13.

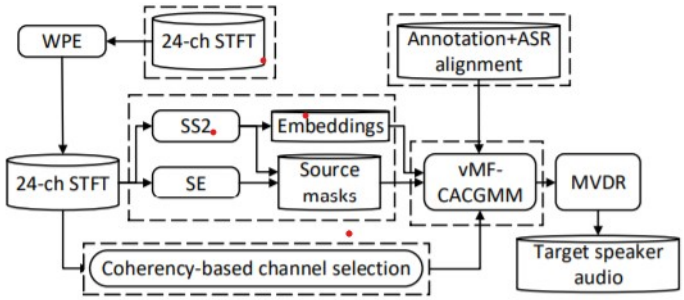

Figura 2.13: Diagrama de flujo del GSS mejorado [4].

Otro trabajo utiliza un *dual path transformer network* para separar fuentes [5]. Se utiliza la data WSJ0-2mix [17] y LS-2mix [32]. Estos *dataset* contienen *speech* grabado, lo que se hace es mezclar 2 *speech* y agregarles ruido, para después separarlas usando un *Dual-Path Transformer Network* (ver figura 2.14).

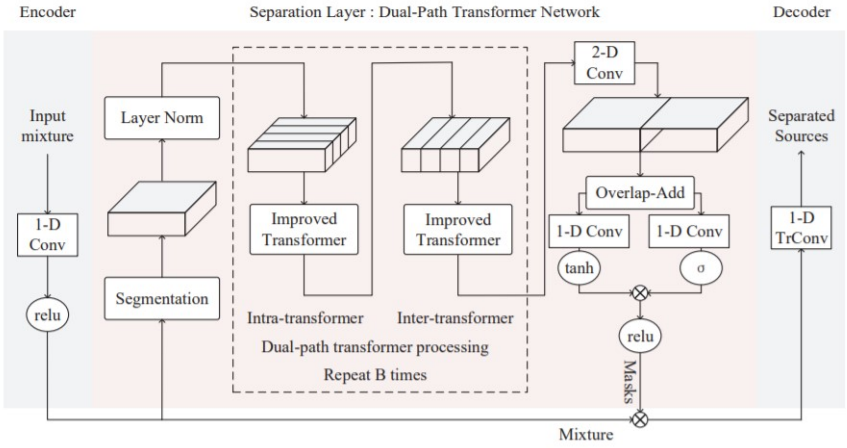

Figura 2.14: Arquitectura *dual-path transformer network* [5].

Se ha desarrollado un estudio que propone el uso de redes adversarias y convolucionales como la red Conv-TasSAN [9]. Se usa el set de datos WSJ0-2mix [17]. Acá se utilizan redes adversarias, que no han sido bien exploradas en la tarea de *source separation*. Por otra parte, también utilizan redes convolucionales para la tarea de separar (Conv-TasNET). El objetivo es separar las fuentes de una señal mixta. La arquitectura que se utilizó puede verse en la figura 2.15.

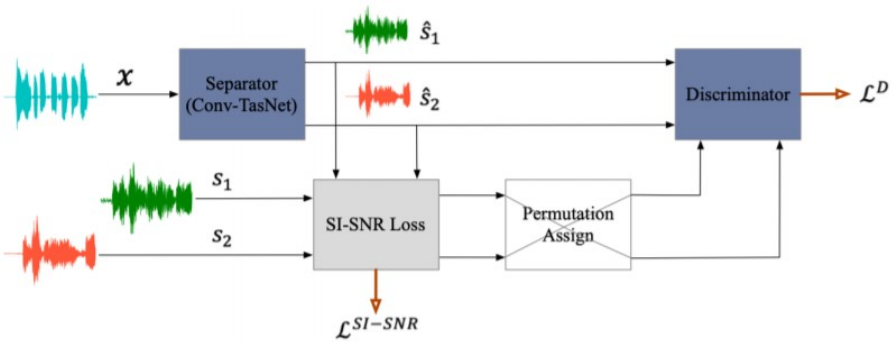

Figura 2.15: Arquitectura Conv-TasSAN [9].

Estos 3 ejemplos de soluciones ([4], [5] y [9]) pertenecen al primer grupo, donde la tarea que se resuelve esta totalmente ligada a la base de datos que se utilizó. Existen varias bases de datos similares a las ya mencionadas (CHiME-6 [47], WSJ0-2mix [17] y LS-2mix [32]). Todas son similares ya que son grabaciones de entrevistas, fiestas, gente narrando cuentos, etc. La tarea a resolver con esta base de datos consiste en mezclar 2 o más *speakers* (hablantes), agregarle artificialmente ruido, reverberación u otras interferencias. Después, con este mix de señales, se busca separar las señales con el algoritmo propuesto.

También, se han desarrollado trabajos utilizando métodos probabilísticos para realizar *source separation* [23]. Se utiliza el *dataset* TIMIT [11] para directamente del el elegir la fuente objetivo y la interferencia. También, se añade ruido no correlacionado. Se tiene un arreglo de 4 micrófonos separados por 3 cm cada uno y el objetivo e interferencia están a 2m. A partir de la configuración descrita anteriormente, se utiliza un algoritmo de post filtrado que es independiente del número y direcciones del punto de interferencia, para así realizar *source separation*. Este método de *post-filtering* es un algoritmo probabilístico que estima la señal.

Otra solución propone usar *mentoring-reverse* que es un *framework* para aprendizaje no supervisado [27]. Este *framework* quiere mejorar 2 sistemas, *junior* y *senior*, el *junior* aprende del *senior* y, después, los conocimientos son compartidos. Utilizan el set de datos TIMIT [11], DEMAND [42] Y MIRD [15] para simular conversaciones simultaneas y después así aplicar el método de *source separation*. Además, la fuente esta separada a 1m del arreglo de micrófonos. La arquitectura del sistema se puede ver en el figura 2.16.

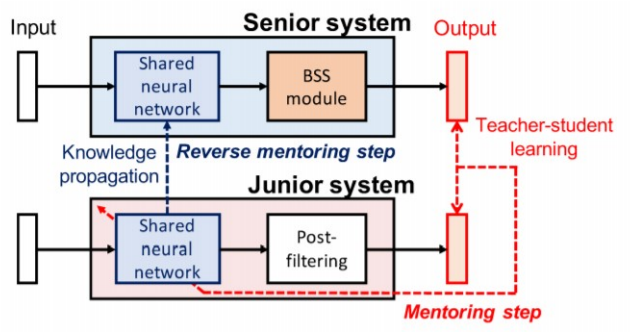

Figura 2.16: Arquitectura mentoring-reverse [27].

También, se ha desarrollado estudios entrenando una red neuronal para pares de micrófonos con diferente espaciado y condiciones acústicas del ambiente y así, realizar una separación de fuentes más efectiva [14]. El propósito es usar estas NN para estimar las máscaras en el tiempo y frecuencia de todos los pares de micrófonos formados. Se utiliza el *dataset* LibriSpeech ASR corpus [32] y se genera el audio mezclado a través de salas de reverberación simuladas. Además, el espacio entre los micrófonos es variable y la fuente se ubica *random* en la sala. El flujo de información del sistema se puede observar en la figura 2.17.

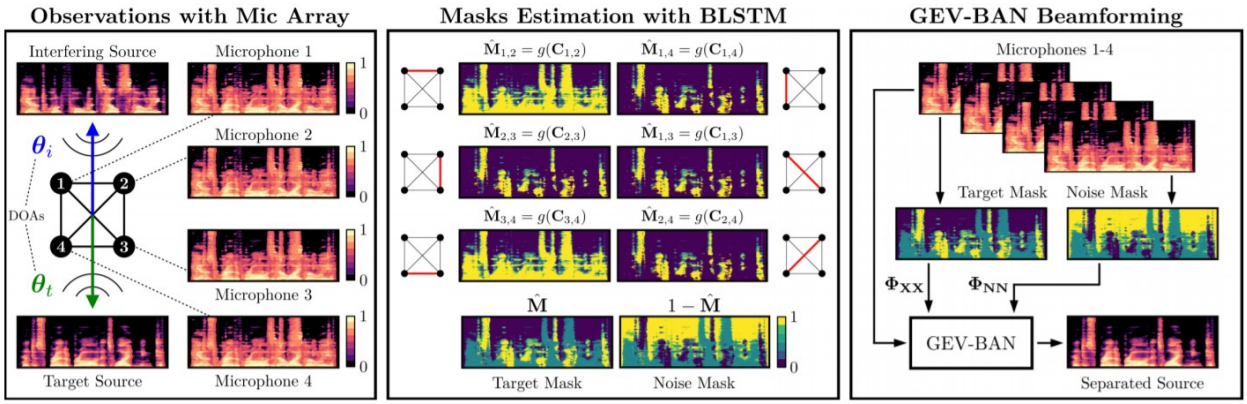

Figura 2.17: Flujo de extracción de características y separación de fuentes para GEV [14].

Se puede notar que los trabajos ([23], [27] y [14]) pertenecen al segundo grupo, debido a que se utilizan micrófonos para captar la señal y probar el sistema. Se utilizan distintos set de datos para formar las señales y, después, se reproducen las señales en salas reales o simuladas para captarla con micrófonos. Con las señales capturadas por los micrófonos se realiza separación de fuentes.

Si analizamos el primer grupo de soluciones, la tarea a resolver esta sujeta al *dataset*. Si bien las técnicas de *source separation* son buenas e innovadoras, las condiciones no son las que se esperan. Estos datos fueron ocupados para, sintéticamente, mezclar señales y así después separarlas. Esto resuelve, insatisfactoriamente, el problema planteado para el trabajo de titulo. Si bien están realizando *source separation*, lo hacen bajo escenarios simulados y no hay una evaluación del método en un escenario real (con gente y micrófonos). Las señales son

mezcladas artificialmente; lo ideal sería encontrar o generar una base de datos en donde ya estén mezclados los audios bajo condiciones reales (varia gente hablando, ruido, reverberación real, etc). Otro punto clave es que no existe movimiento, que es un factor importante que se quiere resolver en el tema de memoria.

Si analizamos el segundo grupo de soluciones, se tiene una evaluación real con micrófonos. Las técnicas usadas en estas soluciones son de gran interés y, además, se resuelve el problema del primer grupo de soluciones, ya que sí hay evaluación del método en un escenario real con micrófonos. De todas formas, este grupo de soluciones también son insatisfactorios, debido a que no aborda el problema del movimiento de las fuentes y/o micrófono/robot.

Si bien se nombraron 3 ejemplos de estudios para el primer y segundo grupo de soluciones, existen más ejemplos de estudios para cada grupo: grupo 1 [6] [10] [12] [16] [21] [22] [24] [26] [28] [31] [37] [48] [50] [53], grupo 2 [13] [19] [40] [46]. En general, los estudios se dividen en esos dos grupos ya descritos y existen otros estudios que hablan de técnicas para extraer más información de la señal o de *frameworks* para trabajar [33] [45].

Las únicas soluciones que han utilizado bases de datos con movimiento entre las fuentes y el robot son estudios que no utilizan modelos *deep learning* de *source separation*. Por ejemplo, se tiene el estudio que utiliza técnicas de *beamforming sum and delay* y MVDR usando seguimiento con cámara de la fuente de *speech*, bajo bases de datos con movimiento [2]. También, se encuentra el trabajo que propone un ASR DNN-HMM para comparar técnicas de *beamforming sum and delay*, Kinect SDK (técnica propia del micrófono Kinect), BeamformIt [1]; bajo otra base de datos con movimiento [29]. Dichas soluciones solo usan técnicas de *beamforming*, pero no evalúan las bases de datos con modelos neuronales de *source separation*.

2.2.2. Técnicas de separación de fuentes

Con la sección anterior, se puede apreciar que existe un marco general de las soluciones ya existentes para el problema de *source separation*. Las soluciones presentadas por otros autores se pueden agrupar en los dos grupos ya presentados. En esta sección se analizarán las técnicas presentes en estos dos conjuntos de soluciones, para así especificar el cómo se aborda el problema de *source separation*.

Últimamente, distintas técnicas han ganado popularidad en el mundo académico para resolver el problema propuesto para el trabajo de titulo. Una de ellas es el uso de redes convolucionales (CNN) [25]. En dicha solución utilizan una red convolucional que estiman las máscaras que se utilizarán para limpiar los espectrogramas de las señales (ConvTasNet). El uso de redes que estiman máscaras para separar las señales es una técnica que proviene del procesamiento de imágenes. Las máscaras en el procesamiento de imágenes son filtros que se le aplican a las imágenes para resaltar determinados grupos de píxeles, o sea, resaltar píxeles de otros. Bajo este concepto hoy en día se están usando máscaras para *source separation*, debido a que, básicamente, el espectrograma es una imagen. Con esto se pueden encontrar máscaras que al aplicarla al espectrograma, se separa el ruido del objetivo.

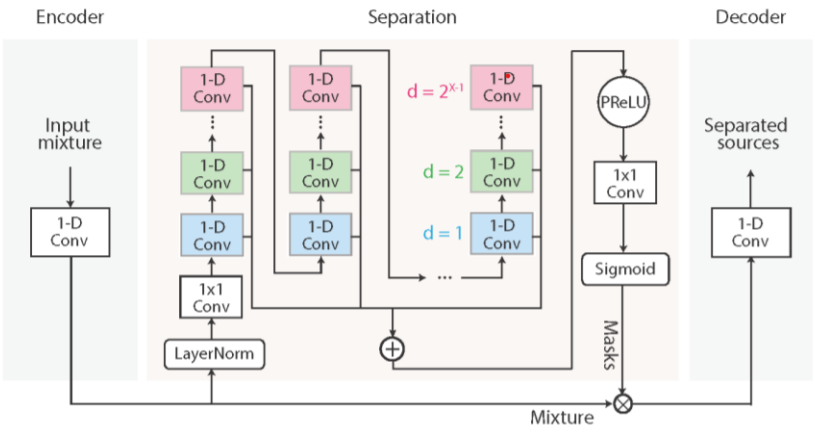

Figura 2.18: Diagrama de bloques ConvTasNet [25].

En la figura 2.18 se puede apreciar el flujo de cómo se estiman las máscaras para separar las señales. Lamentablemente, dicha solución pertenece al primer grupo de las sección anterior, es decir, no tienen una evaluación real con micrófonos.

Bajo esta misma técnica de estimar máscaras, se han presentado distintos modelos que puedan realizar esta tarea. Ya se mostraron algunos ejemplos en la sección anterior, como lo es el estudio *ConvTasSan* [5] y *Dual Path Transformer* [9]. *ConvTasSan* usa como separador a *ConvTasNet* y *Dual Path Transformer*, utiliza la red neural recurrente *transformer* para estimar las máscaras que separarán los audios, pero el principio es el mismo. Similar a la ConvTasNet es la red TCN/CBP la cual utiliza, al igual que la ConvTasNet, una CNN para estimar máscaras usando *skip conections* y *kernel dilatation* [8].

Aparte de las redes convolucionales que están siendo muy explotadas para la tarea de *source separation*, existen otros tipos de redes sumamente populares. Las redes neuronales recurrentes o RNN están siendo sumamente usadas. Se ha propuesto el uso de CNN para estimar mascaras [52] y también se ha propuesto el uso de RNN para replicar técnicas especificas de *beamforming* [49] [51]. En dichos trabajos se utilizan RNN debido a que en principio estas redes son buenas utilizando información secuencial, como lo es el habla, para resolver la tarea que se les presenta.

Estas dos redes, las CNN y RNN, han sido sumamente utilizadas para las tarea de *source separation* en los últimos años. En particular es interesante observar un estudio reciente en donde mezclan las redes CNN y RNN para realizar *source separation* [49] [51]. Se utiliza una CNN para estimar máscaras combinándolo con un bloque RNN para realizar el *beamforming*. Estos dos bloques trabajan en conjunto y cruzan información para así obtener las señales limpias. Es de mencionar que esta solución pertenece al segundo grupo de la sección anterior, debido a que si tienen simulación y evaluación real con micrófonos.

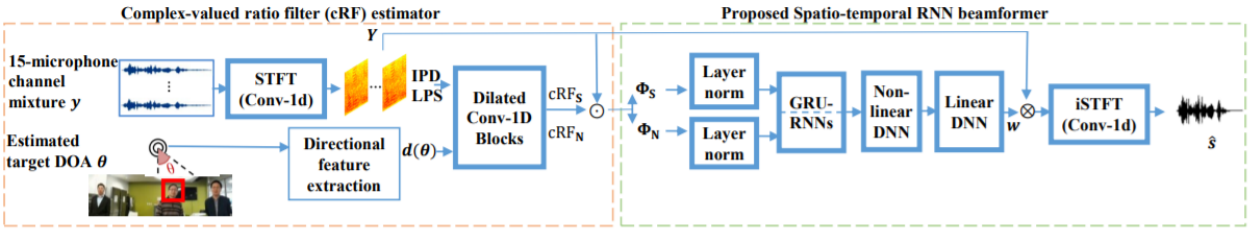

Figura 2.19: Diagrama de GRNN-BF [49].

En la figura 2.19 se puede ver la arquitectura GRNN-BF compuesta por un bloque CNN y un bloque RNN. La red es sumamente interesante, debido a que utiliza la red CNN Conv-TasNet [9] para estimar las máscaras para separación de fuentes. La ConvTasNet se crea juntando bloques de 3 capas convolucionales con cierto factor incremental de dilatación de kernel (ver figura 2.18), donde la red GRNN-BF utiliza 3 de estos bloques en la ConvTasNet. Notar que la red ConvTasNet utiliza *kernel dilatation* y *skip conection* en su arquitectura. Posteriormete, se utilizan las máscaras que se obtienen por el bloque CNN para calcular las matrices de covarianzas del *speech* y del ruido para que ingresen al módulo RNN GRU de *beamforming*. Dicho módulo RNN emula el *beamforming* GEV (ver fórmula 2.7) para estimar los pesos. Con los pesos del *beamforming* que estima la RNN se aplica al espectrograma y se reconstruye la señal.

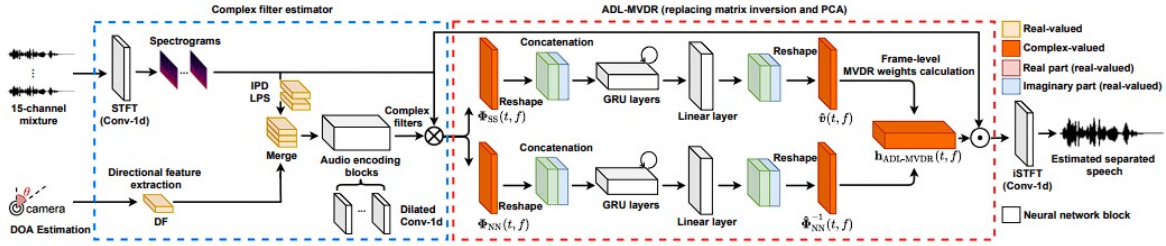

Figura 2.20: Diagrama de ADL-MVDR [51].

En la figura 2.20 se puede observar otro estudio que se hizo con una arquitectura de red combinando una CNN con RNN. La red ADL-MVDR propuesta es bastante similar a 2.19, debido a que usan la misma red ConvTasNet para estimar máscaras y calcular las matrices de covarianza. El segundo bloque es diferente, debido a que la red RNN donde se propagan las matrices de covariza no emulan el *beamforming* GEV en este caso, sino que cada matriz pasa independientemente por una rama de RNN independiente para ser refinadas. Posterior al refinamiento de las matrices, se aplica la fórmula MVDR para calcular los pesos (fórmula 2.8). Con los pesos calculados se aplica al espectrograma y se reconstruye la señal.

2.3. Contribución

Las contribuciones de este proyecto pueden ser bastante influyentes para el mundo científico y (eventualmente) industrial. No existen técnicas que aborden el problema del movimiento para la separación de fuentes, lo que provoca que la interacción de voz entre Humano-Robot sea poco natural. Con la solución que se busca encontrar para el problema planteado para la

memoria, se espera desarrollar técnicas de separación de fuentes robustas que sean capaz de lidiar con el problema del movimiento. Con dicho aporte se podrán tener robots que sean capaz de recrear una experiencia más real y natural a la hora de comunicarse con los humanos. Por fin, se tendrán técnicas de procesamiento robusto de voz que se encarguen del problema del movimiento entre humano y robot.

Con las secciones anteriores se puede notar que en general existen técnicas novedosas para resolver el problema de *source separation*, pero estas técnicas están aplicadas a bases de datos en donde no lidian con el movimiento. Es por esto que el aporte que se hace con este trabajo de titulo, es aplicar técnicas del estado del arte a una base de datos real con movimiento. En especifico se espera adaptar la red propuesta en el trabajo [49], donde se utiliza una arquitectura que mezcla una CNN con una RNN, a una base de datos real.

El aporte de este trabajo es adaptar las arquitecturas GRNN-BF [49] y ADL-MVDR [51] a una base de datos real, probar su desempeño con bases con movimiento simulado y bases con movimiento real y verificar si las redes aportan al momento de separar fuentes o no.

Capítulo 3

Separación de fuentes aplicado a la interacción humano-robot

Para poder lograr el objetivo del proyecto se esperan una serie de resultados acorde a la linea de investigación de este trabajo. Los resultados esperados son los siguientes.

- **1. Replicación**: para este ítem se espera poder replicar 3 soluciones anteriores realizadas por el laboratorio de procesamiento y transmisión de voz de la Universidad de Chile (LPTV). Se deben replicar los resultados de los trabajos [8], [2], [29]. En dichos estudios se utilizan bases de datos con movimiento simulado y real, por lo que se tienen que replicar para efectos de este trabajo de título. Para este ítem se deben entregar tablas replicadas de todos los estudios mencionados.
- **2. Modelos CNN-RNN aplicado a base de datos real**: para este ítem se espera aplicar el la red presentada en GRNN-BF [49] y ADL-MVDR [51] a las bases de datos con movimiento. Por esto, se debe modificar la red para que sea posible introducir esta base de datos a la red deseada. Para este ítem se deben generar tablas de desempeño del modelo frente a las distintas bases de datos.

Primero se empieza el capitulo describiendo las bases de datos que se utilizaron en el proyecto de título. Posteriormente se describe los aspectos metodológicos y modelos CNN-RNN ya mencionados.

3.1. Base de datos

En esta sección se explicarán con detalle las bases de datos con las que se realizó el trabajo de título. Principalmente, se utilizaron dos bases de datos: una base de datos con movimiento simulado, en la cual el movimiento entre las fuentes y el robot fue simulado computacionalmente; y otra base de datos con movimiento real, en la cual las grabaciones fueron hechas en un escenario real con movimiento entre las fuentes y el robot. A continuación, se describirán en detalle los dos set de datos.

3.1.1. Base de datos con movimiento simulado

Esta base de datos fue creada por la investigación realizada por el laboratorio de procesamiento y transmisión de voz de la Universidad de Chile (LPTV) [8]. La base de datos

consta de señales de audios de interlocutores con ruido y reverberación simulado. Hay dos tipos de escenarios en esta base de datos: las señales de audio que solo contienen el *speech* más ruido que se le agrega y las señales de audio que contienen el *speech* más el ruido más reverberación dentro de una sala. La reverberación se simula usando una librería en *Python* llamada *pyroomacosutics* [41]. En dicha simulación se ubican la fuente, ruido y micrófonos de cierta manera para captar la señal.

Las señales de audios de los interlocutores (el *speech*), son señales de la base de datos *AURORA-4 corpus* [35]. Dicho set de datos esta basado en *WSJ0-dataset* [17] que contiene grabaciones de conversacion en entrevistas de wall street (Wall Street Journal): solo un hablante por *utterance*.

La sala de reverberación creada en *pyroom* tiene dimensiones de 2.5m x 6.0m x 6.0m (alto x ancho x largo) con una variación uniforme de $\pm 20\,\%$ en todas las dimensiones. El arreglo de micrófonos es posicionado *random* en la sala con un ángulo de variación de 45 $\pm$30 grados entre la fuente de ruido y fuente de *speech*. La distancia entre el *speaker* y el micrófono es una distribución uniforme entre 1.6m y 2.4m. El hablante y micrófono son posicionados en forma aleatoria en la sala, con las condiciones que estén al menos a 1m de todas las paredes, piso y techo. El arreglo de micrófonos es de 4 micrófonos, posicionados en -11.3cm, 3.6cm, 7.6cm y 11.3 medido desde el centro del array (las dimensiones emulan el micrófono Kinect). Cada micrófono fue modelado como omnidireccional (micrófono con sensibilidad de respuesta en todas las direcciones). La sala se puede apreciar en la figura 3.1.

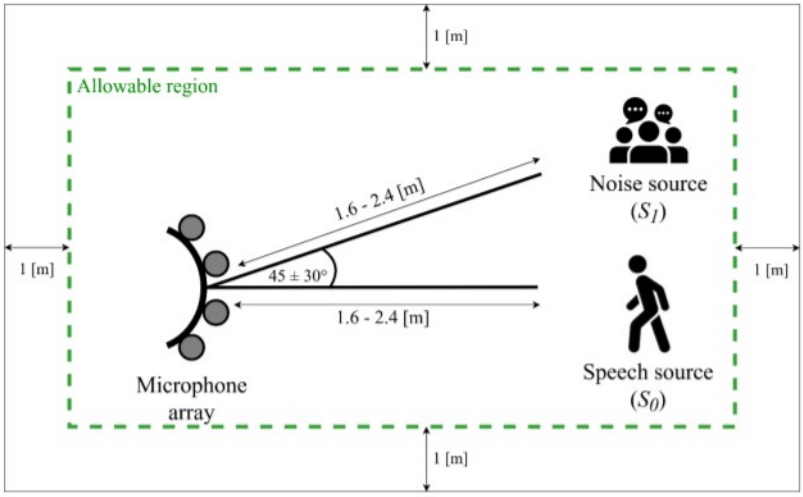

Figura 3.1: Ilustración de sala simulada en *pyroom* [8].

Así, de *AURORA-4* se extraen los hablantes que serán reproducidos. Estos audios, más la sala simulada, se utilizaron en diferentes configuraciones para generar distintos escenarios. Se tienen 4 diferentes escenarios (sub-database).

- **Soloruido**: La base de datos Soloruido contiene los *speech* de *AURORA-4 corpus* con ruido agregado; no hay reverberación.
- **Matched**: La base de datos Matched contiene fuentes de *speech*, ruido y reverberación de la sala simulada en *pyroom*. La configuración Matched corresponde a cuando la respuesta

al impulso es calculada justo al medio de la sala. La respuesta al impulso de la sala (RIRs) será entonces siempre la misma para todas las *utterance* de *training*, *dev* y *test*.

- **Multicondition**: La base de datos Multicondition es generada disponiendo aleatoriamente (en cierto rango) el arreglo de micrófono, ruido y fuentes para cada utterance dentro de la sala simulada en *pyroom*. Posteriormente, con dicha configuración, se calcula la respuesta al impulso de la sala (RIR). Así para cada *clean utterance* se tiene diferentes respuestas al impulso en *training*, *dev* y *test*.

- **Movil**: La base de datos Movil simula movimiento dentro de la sala generada en *pyroom*. Para simular el movimiento se varía el SNR. Al variar la relación señal ruido se genera el mismo efecto de alejarse o acercarse a los micrófonos. Se realiza un movimiento lineal entre 1.0m y 3.0m a una velocidad media de 5.0km/h con respecto al arreglo de micrófonos. La energía de la señal fue modificada de tal forma que la ganancia sea proporcional a $\frac{1}{x^2}$, donde x es la distancia entre la fuente de *speech* y el arreglo de micrófonos. Por otra parte para cada utterance se dispone de una posición de la fuente de ruido, *speech* y micrófono en la sala simulada en *pyroom*. Con ello, se calcula para cada utterance el RIRs.

Las respuestas al impulso, mencionadas en Matched, Multicondition y Movil, se utilizan para calcular los *beamforming*, apuntando a la fuente y al ruido. El *beamforming* que se utiliza es el *sum and delay*, ya que se tiene información del tiempo que se debe desplazar las señales. Todo fue simulado y se cuenta con dicha información. Utilizando la siguiente fórmula derivada del *sum and delay* y ciertas suposiciones:

$$b_{S_0}(t) \cong h_{S_0,S_0} * s_0(t) + h_{S_0,S_1} * s_1(t) \tag{3.1}$$

$$b_{S_1}(t) \cong h_{S_1,S_0} * s_0(t) + h_{S_1,S_1} * s_1(t) \tag{3.2}$$

Donde los h corresponden a las respuestas impulsivas observadas por los micrófonos en la dirección de S_0 y S_1 (fuente y ruido). En dichas ecuaciones h_{S_0,S_0} y h_{S_0,S_1} corresponden a las respuestas impulsivas observadas por el micrófono en la dirección S_0 y S_1, respectivamente, cuando el *beamforming* se realiza en dirección de S_0. En cambio, h_{S_1,S_0} y h_{S_1,S_1} corresponden a las respuestas impulsivas observadas por el micrófono en la dirección de S_0 y S_1, respectivamente, cuando el *beamforming* se realiza en dirección de S_1. Así, de las configuraciones Matched, Multicondiction y Movil se extraen los h (RIR) para calcular las señales *beamforming*, apuntando al ruido y a la fuente. Solo ruido no experimenta esto, debido a que no sufre reverberación, solo se le añade ruido.

Para llegar a las ecuaciones (3.1) y (3.2) se asume que a los micrófonos llega una onda plana, debido que se cumple que la distancia entre el array de micrófonos y las fuentes de sonidos es mayor que 5-10 veces la longitud del arreglo. Además, se asume que la señal que llega a los N micrófonos tiene la misma energía. Esto es razonable debido a que la distancia entre el array de micrófonos y las fuentes de sonido es mayor que 5-10 veces la longitud del arreglo y además, si los micrófonos son omnidireccionales. Por otra parte, se considera un sistema quasi-estático.

Así, la base de datos de movimiento simulado tiene 4 sub-database: Soloruido, Matched, Multicondition y Movil. Donde en cada uno de los set de datos, menos el Soloruido, se tiene

los *beamforming* apuntando a la fuente y al ruido. En el set de datos Soloruido se tiene la señal sucia y la señal limpia. Todas son señales en el tiempo.

A estas *utterance* en el tiempo se les extrae el espectrograma, con una ventana de 25ms y 15ms de *overlap* a un paso de 10ms. Con ello los audios pasan a ser imágenes. También, se crean variantes de los *datasets* Matched, Multicondition y Movil, donde se utiliza la técnica WPE para desreverberar los bines de frecuencia de los espectrogramas. Finalmente, la base de datos de movimiento simulado, son imágenes. Principalmente, se trabaja y se procesa la sub-bases Matched, Multicondition y Movil (con y sin WPE), debido a que tienen reverberación. La sub-base Soloruido solo se utiliza para hacer comparaciones de métricas en situaciones sin reverberación, pero no será utilizada para entrenar modelos neuronales. A esta base de datos se le referirá como BD_{sim} por simplicidad. Para cada subdataset se tienen 7.138 señales de *train*, 330 señales para validación y 330 señales de *testing*.

3.1.2. Base de datos con movimiento real

Hay a disposición, dos bases de datos con movimiento real. Las dos bases de datos contienen las mismas señales, pero fueron generadas en diferentes condiciones. A continuación, se describen los set de datos.

3.1.2.1. Base de datos THRI

Esta base de datos fue creada por el LPTV. Para generar dicha base de datos se utilizó un escenario real donde el robot PR2 se mueve hacia adelante y hacia atrás con presencia de señales fuente y ruido [29]. El robot PR2 cuenta con un micrófono Kinect de 4 canales para grabar las señales. A esta base de datos se le hará referencia como BD_{THRI}.

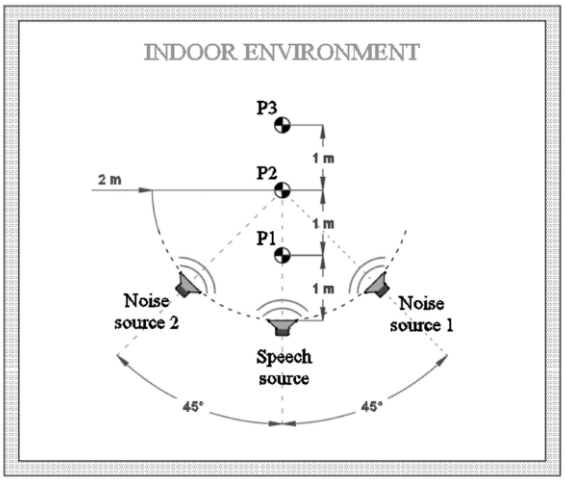

Figura 3.2: Ilustración de las posiciones para la grabación del *dataset* BD_{THRI} [29].

La figura 3.2 muestra las posiciones en la cual se dispusieron los diferentes elementos en la sala. Se tienen los puntos P1, P2 y P3, los cuales son los puntos por donde se mueve el robot PR2. Cada punto esta separado en 1 metro del otro. Del punto P2 se posiciona la fuente de *speech* alineada al movimiento del robot y 2 fuentes de ruido a 45 y -45 grados; las tres a

una distancia de 2 metros con respecto al punto P2. Con este *setup* notamos que el robot PR2 puede moverse hacia adelante y hacia atrás, teniendo la fuente de ruido alineada a su movimiento y dos fuentes de ruido laterales.

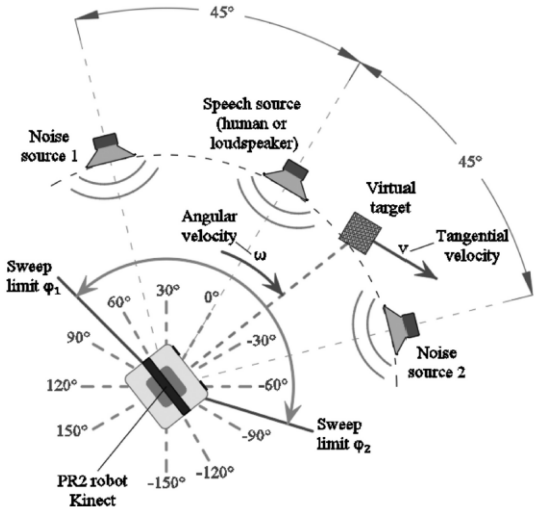

Figura 3.3: Ilustración del movimiento de la cabeza para el *dataset* BD_{THRI} [29].

De la figura 3.3 se puede apreciar el movimiento de la cabeza del robot PR2. Además del movimiento hacia adelante y hacia atrás, entre los puntos P1 y P3, el robot puede mover la cabeza periódicamente, haciendo un barrido. Los limites del barrido angular del robot son $\phi_1 = -150$ y $\phi_2 = 150$. Esos son los ángulos extremos en cual el robot puede mirar; al llegar a esos puntos el robot cambia de sentido. Así, el robot PR2 puede moverse hacia adelante y hacia atrás teniendo también una velocidad angular en el movimiento de la cabeza. Con esas variables se graban 3 *subdatasets*, los cuales se explican a continuación.

Tabla 3.1: *Subdatasets* grabados para la base de datos BD_{THRI} [29].

	Condition ID	Displacement Vel. [m/s]	Angular Vel. [rad/s]	Head Angle	Ext. Noise Sources
Database-I		0	0	0	0
			0.28	-	0
			0.42	-	0
			0.56	-	0
		0.3	0	0^o	0
			0.28	-	0
			0.42	-	0
			0.56	-	0
		0.45	0	0^o	0
			0.28	-	0
			0.42	-	0
			0.56	-	0
		0.6	0	0^o	0
			0.28	-	0
			0.42	-	0
			0.56	-	0
Databse-II	Static 1	0	0	0^o	1
	Static 2	0	0	45^o	1
	Dynamic 1	0	0.42	-	1
	Dynamic 2	0.45	0.42	-	1
Database-III	Static 1	0	0	0^o	2
	Static 2	0	0	45^o	2
	Dynamic 1	0	0.42	-	2
	Dynamic 2	0.45	0.42	-	2

En la tabla 3.1 se sintetizan los experimentos que se grabaron con el *setup* ya descrito. La primera columna indica el nombre del *subdataset*; la segunda, es el ID del experimento; la tercera, indica la velocidad que lleva el robot moviéndose hacia adelante y atrás; la cuarta, es la velocidad angular de la cabeza cuando hace el barrido; la quinta, es a dónde esta mirando fijamente el robot (si es que la velocidad angular es 0). La ultima columna, es la cantidad de fuentes de ruido. Los momentos cuando el robot está con velocidad 0 en el desplazamiento, significa que está parado en el punto P2.

Se puede notar que la Database-I, es una base de datos con diferentes combinaciones de velocidades del movimiento que realiza el robot con las diferentes velocidades angulares con la cual puede mover la cabeza. La Database-I no cuenta con fuentes de ruido. La Database II y III son parecidas. Un experimento es el robot parado en P2 mirando a la fuente de *speech* o la fuente de ruido (Static 1 y Static 2, respectivamente). Y también, hay experimentos donde el robot no cuenta con movimiento en el desplazamiento pero si angular y otro en el cual el robot cuenta con movimiento en el desplazamiento y también angular (Dynamic 1 y Dynamic 2, respectivamente). La diferencia entre la Database-II y Database-III es que la II presenta una fuente de ruido, mientras que la III presenta dos.

En particular, es de interés el Database-II y Database-III, debido a que tienen movimiento y fuentes de ruido. Para todos los *subdatasets* solo se cuenta con 330 señales de *test*. Las señales de *speech* vienen del *AURORA-4* [35], que fue explicada en la BD_{sim}. Las fuentes de ruido reproducen ruido de restaurant calibrado a 5dB.

Lo que se tiene para cada *subdatasets* son las 4 señales captadas por la Kinect y un archivo donde se guardan los tiempos y ángulos que registra el motor que mueve la cabeza del robot.

3.1.2.2. Base de datos VS

Esta base de datos fue creada por el LPTV. Para generar dicho set de datos se utilizó un escenario real donde el robot PR2 se mueve en direcciones laterales con presencia de señales fuente y ruido [2]. El robot PR2 cuenta con un micrófono Kinect de 4 canales para grabar las señales. A esta base de datos se le hará referencia como BD_{VS}.

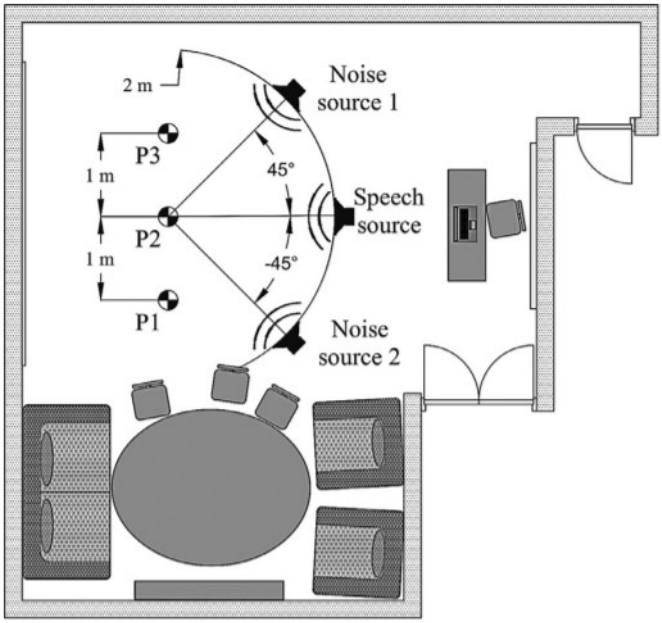

Figura 3.4: Sala utilizada para la grabación del *dataset* [2].

La figura 3.4 representa el escenario que se utilizó para grabar las señales. El robot PR2 realiza movimientos laterales periódicos, moviéndose a una máxima velocidad de 0.45m/s entre P1 y P3. Por otra parte, el robot sufre una aceleración y desaceleración al salir y acercarse a cada punto. Es importante mencionar que el movimiento lateral del robot entre los puntos P1 al P3 fue programado pero, aun así, necesitaba correcciones menores debido a que se desviaba.

Las fuentes de *speech* y ruido fueron ubicadas a 2m del punto P2. Las señales de *speech* vienen de la base de datos *AURORA-4* [35], que ya fue explicada en la BD_{sim}. Se ubicaron 2 fuentes de ruido, la primera "Noise source 1" formando un ángulo de 45 grados con la fuente de habla, ubicada a 2m del punto P2. La segunda fuente de ruido fue ubicada a -45 grados de la fuente de habla, a 2m del punto P2. Las fuentes de ruido reproducen ruido de un restaurant calibrado a un SNR de 5dB.

Por otra parte, se utiliza YOLO (red neuronal para el procesamiento de imágenes) para reconocer la fuente de *speech* [38]. Con esta información se puede calcular la diferencia angular entre el centro de la imagen y la fuente de habla. En la figura 3.5 se puede apreciar como funciona YOLO detectando la fuente de habla y el centro de la imagen. El movimiento

siempre esta centrado y se conoce la altura; no se mueve hacia arriba o abajo.

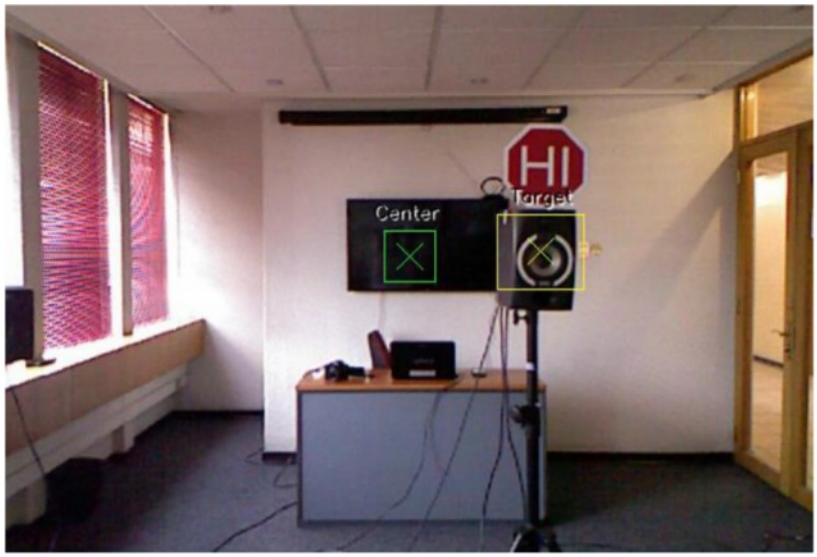

Figura 3.5: *Image tracking* usando YOLO [38], el rectángulo verde es el centro de la imagen y el rectángulo amarillo es la fuente de *speech* [2].

Por otra parte, el robot puede realizar VS (*Visual Servoing*) con la información extraída con YOLO para mover la cabeza para apuntar a la fuente. Con la diferencia angular entre el centro y la fuente de habla, se puede mandar una señal de comando al robot para que mueva su cabeza y corrija la mirada.

Con todo lo mencionado anteriormente, se tiene claro el movimiento del robot, escenario, fuentes de habla/ruido y *tracking* de la fuente de habla. Con estos principios se generan 6 distintos escenarios. Los cuales se en listan a continuación.

- **Mov-1**: El robot se mueve entre puntos P1 a P3 como fue ya descrito y la cabeza del robot esta fija en 0º (no mueve la cabeza). Solo se tiene la fuente de ruido "Noise source 1". Con esta configuración se reproducen las señales con el ruido y se graban las señales captadas por el micrófono del robot mientras el robot se mueve. También, al mismo tiempo, se registran la diferencia en ángulos entre el centro y la fuente de *speech*.
- **Mov-2**: El robot se mueve entre puntos P1 a P3 como fue ya descrito y la cabeza del robot esta fija en 0º (no mueve la cabeza y esta centrada). Se tiene las fuentes de ruido "Noise source 1" y "Noise source 2". Con esta configuración se reproducen las señales con el ruido y se graban las señales captadas por el micrófono del robot mientras el robot se mueve. También, al mismo tiempo, se registran la diferencia en ángulos entre el centro y la fuente de *speech*.
- **VS-Mov-1**: El robot se mueve entre puntos P1 a P3 como fue ya descrito y la cabeza del robot se mueve siguiendo a la fuente de *speech*. Solo se tiene la fuente de ruido "Noise source 1". Con esta configuración se reproducen las señales con el ruido y se graban las señales captadas por el micrófono del robot mientras el robot se mueve. También, al mismo tiempo, se registran la diferencia en ángulos entre el centro y la fuente de *speech*.

- **VS-Mov-2**: El robot se mueve entre puntos P1 a P3 como fue ya descrito y la cabeza del robot se mueve siguiendo a la fuente de *speech*. Se tiene las fuentes de ruido "Noise source 1" y "Noise source 2". Con esta configuración se reproducen las señales con el ruido y se graban las señales captadas por el micrófono del robot mientras el robot se mueve. También, al mismo tiempo, se registran la diferencia en ángulos entre el centro y la fuente de *speech*.

- **ST-1**: El robot está estático en P2 y la cabeza esta fija en 0^{o} mirando la fuente de *speech*. Solo se tiene la fuente de ruido "Noise source 1". Con esta configuración se reproducen las señales con el ruido y se graban las señales captadas por el micrófono del robot quieto. Y como esta quieto, no se registran los ángulos.

- **ST-2**: El robot está estático en P2 y la cabeza esta fija en 0^{o} mirando la fuente de *speech*. Se tiene las fuentes de ruido "Noise source 1" y "Noise source 2". Con esta configuración se reproducen las señales con el ruido y se graban las señales captadas por el micrófono del robot quieto. Y como esta quieto, no se registran los ángulos.

Para cada escenario (*subdataset*) se tienen las señales de audio captadas por el micrófono del robot en 4 canales y un archivo con el tiempo/ángulos usando YOLO (diferencia de angular entre el centro y la fuente de *speech*). Al igual que en la base de datos BD_{THRI} solo se tienen 330 señales de *test*.

3.2. Aspectos Metodológicos

Para abordar el problema propuesto para la memoria, se debe describir cúal será la metodología a seguir. En esta sección se describirá la metodología, técnicas y estándares que se seguirán para desarrollar el proyecto.

La metodología empleada para la memoria consistirá, como ya se mencionó, en realizar 3 réplicas más el sistema propuesto, que consiste en adaptar las redes GRNN-BF y ADL-MVDR a las bases con movimiento real. Con las réplicas de estudios anteriores, se tendrá un *baseline* para comparar con el sistema propuesto.

3.2.1. Generación de data simulada

Como la data de las bases de datos reales BD_{THRI} y BD_{VS} son solo 330 señales, no se puede entrenar un modelo robusto con ello. La solución es generar datos de entrenamiento simulados y probar el modelo con las bases de datos reales. De *AURORA-4* se tienen 7.138 señales de entrenamiento, 330 de dev y 330 de test [35]; esas serán utilizadas para generar las bases de datos simulada.

Los datos de entrenamiento deben ser simulados convolucionando las RIR con las señales de *AURORA-4* limpias. Con ello, se simulan los rebotes que puede tener la señal en la sala. Las RIR fueron calculados de manera real en la sala del laboratorio de LPTV y esos son los que se ocuparon para convolucionar las señales. Las RIR fueron creados bajo las mismas condiciones de las bases de datos BD_{THRI} y BD_{VS}. Las fuentes de *speech* y ruido fueron ubicadas en las mismas posiciones que las bases de datos reales. El robot fue posicionado en el punto P1,P2 y P3 (ver figura 3.2); se hizo que el robot mire a distintos ángulos. Con

esa configuración, se calcularon las RIR desde la fuente de *speech* situada en los 0º. Así, estimaron las distintas respuestas impulsivas con el robot mirando en diferentes ángulos con la fuente sonora siempre en los 0º.

En total se tienen 33 RIR, entre [-50, 50] grados avanzando de a 10 grados para las posiciones P1, P2 y P3. Con ellos se convolucionan las señales de *AURORA-4* limpias para realizar la reverberación. Por otra lado, también se le añade ruido de restaurant y el ruido que genera internamente el robot. Bajo dichas condiciones se generan dos set de datos simulados.

- BD II Static 1 simulated

  Esta base de datos simulada replica las condiciones de la BD_{THRI} BD II Static 1 real. En dicha base de datos real se tiene al robot PR2 mirando a la fuente de *speech* en los 0º y la fuente de ruido esta en los 45º. Las RIR utilizadas para realizar la convolución siempre fueron cuando el robot PR2 estaba situado en P2. Para replicar dicho escenario se toman las señales de *AURORA-4* y se convolucionan con la RIR en angulo 0º (RIR calculada cuando el robot mira en 0º). Esto simula que el robot está mirando a la fuente de *speech*. Por otro lado, se añade ruido de restaurant convolucionado con la RIR en angulo -50º (RIR calculada cuando el robot mira en -50º). Así se simula que el ruido viene desde la izquierda (50º). Posteriormente, también se le añade ruido propio del robot. Al ruido de restaurant se le añade un desfase temporal con respecto al DOA que se simula con la mirada del PR2 que, en este caso, es de 50º.

- BD II Static 2 simulated

  Esta base de datos simulada replica las condiciones de la BD_{THRI} BD II Static 2 real. En dicha base de datos real se tiene al robot PR2 mirando a la fuente de ruido en los 45º y la fuente de *speech* esta en los 0º. Las RIR utilizadas para realizar la convolución siempre fueron cuando el robot PR2 estaba situado en P2. Para replicar dicho escenario se toman los audios de *AURORA-4* y se convolucionan con la RIR en angulo 50º (RIR calculada cuando el robot mira en 50º). Esto simula que el robot esta mirando en 50º y el *speech* viene de los 0º. Por otro lado, se añade ruido de restaurant convolucionado con la RIR en ángulo 0º (RIR calculada cuando el robot mira en 0º). Así se simula que el ruido viene de alfrente, o sea a los 0º. Posteriormente, también se le añade ruido propio del robot. Al *speech* se le añade un desfase temporal con respecto al DOA que se simula con la mirada del PR2 que, en este caso, es de -50º.

- CrossRever

  Esta base de datos replica una situación *multicondition*, en el sentido que la fuente de *speech* y la fuente de ruido ocupan diferentes posiciones para distintas *utterance*. En dicha base de datos se elige una RIR con cierto ángulo para el *speech* y una RIR con cierto ángulo para la fuente de ruido, la única condición que deben cumplir es que la diferencia de ángulo entre las RIR de mayor a 20º. Con ello, se itera sobre las 33 RIR y se convolucionan las RIR con las señales de audio y con el ruido de restaurant. Posteriormente a la convolución, se desfasan temporalmente las señales de *speech* y ruido con respecto al DOA de la RIR y se suman. Finalmente, se añade el ruido del robot a la señal resultante.

Con las bases de datos BD II Static 1 simulated y BD II Static 2 simulated se entrenaron los modelos y se testearon en las bases de datos con movimiento real. La base de datos CrossRever

se usa para entrenar un nuevo ASR. Se tienen en total 7.138 señales de entrenamiento, 330 de validación/dev y 330 de test. Al desfasar las señales en el tiempo se simulan 4 canales por audio grabado con un micrófono Kinect. Además, se tiene un csv con los mirada del robot según la posición de la fuente de *speech*.

3.2.2. Pre-procesamiento de los datos

3.2.2.1. Base de datos BD_{sim}

Se replica el proceso de pre-procesamiento para la base de datos BD_{sim} igual como se hizo en [8]. El procesamiento que se describirá será solo aplicado a las bases de datos Matched, Multicondition y Movil. Primero, se extraen métricas de los espectrogramas para poder hacer una normalización, debido a que es conocido y recomendado normalizar los datos cuando se trabaja con redes neuronales. La normalizaciones en redes neuronales son necesarias, ya que se evita que ciertos datos tengan una valor más elevado, lo que equivale a tener un peso más elevado de influencia hacia la red. Con esto claro, se sacan el promedio y desviación estándar a todo el conjunto de *train* y *test* por bin de frecuencia. No se extrae el promedio y desviación estándar del conjunto *dev*, debido a que el conjunto de *train* es lo suficientemente grande y representativo para utilizar los promedios y desviación estándar del conjunto de *train*.

Después, se prosigue a realizar *reflect padding* a los espectrogramas que no calcen con el *shape* de la entrada (ventana de análisis) de la red que se utilizará. A la red se le entregarán pedazos del espectrograma de una utterance y puede pasar que al cortar esos pedazos queden al final restos que no coincidan con la entrada de la red. Es por esto, que en los casos que no coincida el resto de los espectrogramas con el *input shape*, se realiza *reflect padding* para completar los espectrogramas y hacer que calcen.

Ya teniendo los espectrogramas con el *shape* deseado para que los pedazos entren en la red, se aplica MVN a todos los espectrogramas. MVN que es una técnica de normalización que utiliza el promedio y la varianza (ya calculados). Estos espectrogramas con *padding* y normalizados serán utilizados para entrenar una red convolucional de *source separation*. Todo este procesamiento se realiza en *Python*.

3.2.2.2. Base de datos BD_{THRI} y BD_{VS}

El pre-procesamiento que se le hace a ambas bases de datos con movimiento real, es el mismo. Las dos bases de datos con movimiento presentan las señales de los 4 micrófonos más un archivo con los ángulos registrados por el motor del robot (BD_{THRI}) o YOLO (BD_{VS}). Los ángulos que se encuentran en los respectivos archivos de las bases de datos se encuentras sub-muestreados. Es decir, que si los audios tienen una tasa de muestreo de 16 [kHz], los audios no cumplen con esa tasa, no existe un ángulo por cada muestra de tiempo. Es más, los ángulos fueron registrados de forma aleatoria; no siguen un tasa de muestreo constante.

Es por esto que se sobre-muestrean los ángulos registrados, para así obtener un ángulo por cada muestra de tiempo y que coincida con la tasa de muestreo de la señal. Por ello, para cada muestra de tiempo, se buscan los dos ángulos más cercanos registrados en los archivos originales de las bases de datos. Al extraer los dos ángulos más cercanos, se hace una interpolación y se asigna el resultado de la operación a la muestra de tiempo t.

Realizando el proceso descrito en el párrafo anterior, se logrará tener un nuevo archivo de ángulos, en donde para cada tiempo t se tiene un ángulo registrado (que fue obtenido por interpolación). Donde el tiempo sigue la misma tasa de muestreo de las señales de audio, que para este caso ambas bases de datos tienen una frecuencia de muestreo de 16 [kHz].

3.2.3. *Beamforming* y *Sum Without Delay* (SWD)

Se aplicará *beamforming sum and delay* para replicar los experimentos realizados en [29] y [2] . Con ello, se replican los resultados obtenidos en cada trabajo con la base BD_{THRI} Y BD_{VS}, respectivamente [29] y [2]. Se utiliza el algoritmo *sum and delay* usando la información visual, es decir se utilizan los ángulos de incidencia registrados en las bases de datos. Además, también se le hace *sum and delay* a las bases de datos BD II Static 1 y Static 2 simulated para poder hacer comparaciones más adelante.

Por otra parte también se obtiene el resultado SWD (*sum without delay*) de las bases de datos, debido a que ese resultado representa el no aplicarle ningún tipo de procesamiento a las señales. SWD es sumar directamente, sin aplicar algún tipo de desfase, los canales del audio. Por esto se usará como punto de partida para comparar si los algoritmos están funcionando.

Realizando *sum and delay* y SWD, se tendrá un punto de comparación para las técnicas que se implementarán en este trabajo de título. Así, se tiene un *baseline* de los desempeños con las bases de datos con movimiento real y, con ello, se podrán comparar métricas de los resultados obtenidos con el sistema propuesto. El *beamforming* se realiza usando el lenguaje de programación *Python*.

3.2.4. Modelo de *source separation*

3.2.4.1. Red convolucional TCN/CBP

Se replica la red convolucional TCN/CBP para la base de datos BD_{sim} [8]. Los datos que entran al modelo tiene el procesamiento descrito en la sección 3.2.2.1.

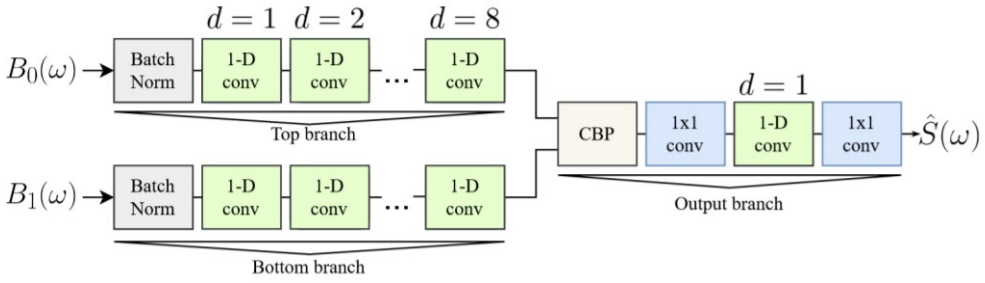

Figura 3.6: CNN TCN/CBP [8].

En la figura 3.6 se puede apreciar que la red tiene como entrada los espectrogramas de los *beamforming* apuntando a la fuente y al ruido. Estos son procesados en dos ramas distintas para después ser concatenados usando la técnica de CBP y tener un output. Se puede apreciar que los bloques convolucionales "1-D conv" experimentan *kernel dilatation*

con diferentes factores (d = 1, 2, ... , 8). Estos bloques "1-D conv" están compuestos por: *batchnormalization*, capas convolucionales dilatadas, función de activación ReLU y una *skip connection*. El bloque "1-D conv" se puede apreciar en la figura 3.7.

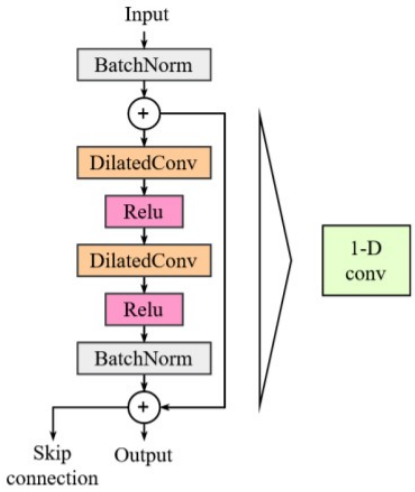

Figura 3.7: Bloque "1-D conv" de red TCN/CBP [8].

La ventana de análisis (*input shape*) es de largo 160, 320 o 640. Esto quiere decir que la red toma pedazos de largo 160, 320 o 640 de los espectrogramas (para que siempre los pedazos que tomó sean de esos largos se tuvo que realizar *reflect padding* como ya se mencionó) y así se procesan por la CNN. Esta red fue implementada en *Python* con las librerías *Keras* y *Tensorflow*. Así, esta red TCN/CBP es una red que toma pedazos de espectrograma y les corrige la reverberación. La salida de la red tiene el mismo *shape* de entrada.

3.2.4.2. Red neuronal Deep-Gev

Como se mencionó antes, la contribución que se plantea hacer es adaptar la red GRNN-BF a las bases de datos con movimiento real [49]. La arquitectura que se usa para realizar los experimentos se puede observar en la figura 3.8.

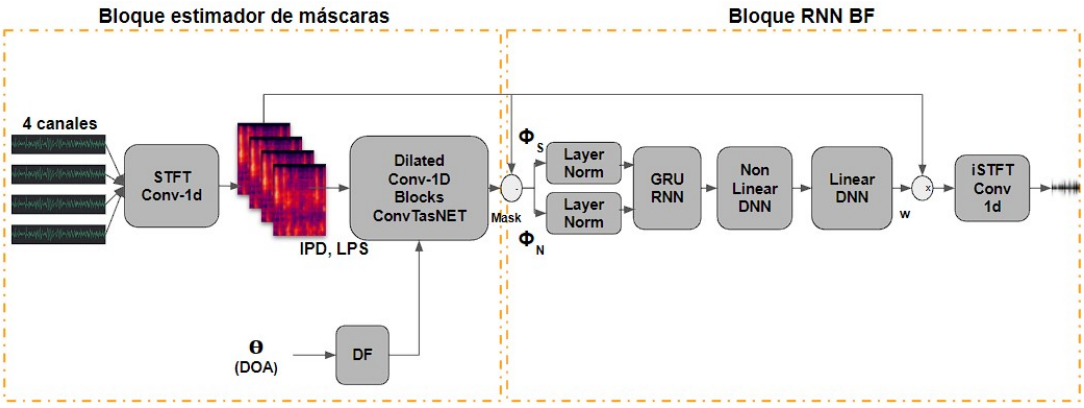

Figura 3.8: Diagrama de bloques red Deep-GEV.

De la figura 3.8 se puede observar el flujo de información de la red neuronal Deep-GEV.

La red Deep-GEV cuando con un primer bloque CNN que estima máscaras y un segundo bloque de RNN *beamforming*. La red tiene como entrada 4 canales, posteriormente a cada canal se le hace una normalización por potencia objetivo. Se elige un canal de referencia y se normaliza las potencias de los otros canales según la potencia del canal de referencia, con ello se mantiene la potencia relativa entre los canales. La red original GRNN-BF no contaba con dicha normalización.

Posteriormente a la normalización, se introducen los canales al bloque STFT (transformada de Fourier) resultando en un espectrograma por canal. A continuación, se le calculan a los espectrograma de cada canal las métricas LPS e IPD. La métrica LPS es el *log-power spectra*, lo cual significa que es el logaritmo del espectrograma. La métrica IPD fue mencionada en el marco teórico.

El *feature* IPD se calcula como en las ecuaciones 2.4. Además, a la red se le ingresa el *feauture* DF. Este se calcula como en 2.5 utilizando los ángulos registrados por el robot (DOA) posterior a ser sobre-muestreados. Estos 3 *features* (LPS, IPD y DF), ingresan al bloque "Dilated Conv-1D Blocks ConvTasNET".

Los *features* LPS, IPD y DF se concatenan y entran al bloque convolucional. Este bloque convolucional no es más que una ConvTasNet (figura 2.18). La salida de la red convolucional es la estimación de una máscara. Dicha máscara sirve para calcular las matrices de covarianza. En el estudio original se estima la máscara cRF que, en simples palabras, es una máscara de máscaras; se tiene una máscara para cada píxel del espectrograma. En cambio, en este trabajo se empleó una máscara para todo el espectrograma. Posteriormente, con la máscara estimada y el espectrograma, se calculan las matrices de covarianza del ruido y de la señal de *speech*. Estas matrices de covarianza de la señal y ruido ingresan al bloque RNN de la red.

Las matrices de covarianza entran a la red recurrente GRU. Esta red RNN hace el trabajo de *beamforming*. La red aprende a estimar los pesos del *beamforming* GEV a través de las matrices de covarianza. El *beamforming* GEV aplica componentes principales (PCA) y el bloque RNN de la red Deep-GEV tiene una capa "Linear DNN" que aplica una transformación lineal, lo cual es lo mismo que PCA. Con ello el bloque RNN emula el *beamforming* GEV (fórmula 2.7) y al final de la RNN se obtienen los pesos (w). La red recorre recursivamente, independientemente, cada bin de frecuencia de las matrices de covarianza. Con los pesos estimados, se convolucionan los pesos con los espectrogramas obteniendo la señal estimada. La señal estimada se pasa por una transformada de Fourier inversa (iSTFT) para llevarlo al dominio del tiempo. Así, con esto se tiene un modelo que le entrenan 4 señales (canales) que contienen ruido y reverberación, pudiendo separar la interferencia de la señal de *speech*, obteniendo así la señal objetivo estimada.

A la red le ingresan segmentos de audio, es decir, que se toman *chunks* de 4 segundos y esos son procesados dentro de la red. Al momento de entrenar, se toman 4 segundos al azar del audio con sus respectivos ángulos y se ingresan a la red. A la hora de validar y testear, se deben segmentar los audios para poder ingresarlos por la red y, después, reconstruir el audio. Para ello, se aplican ventanas hamming de 512 muestras con 50 % de traslape para los audios y, paralelamente, se toman las mismas 512 muestras con una ventana rectangular con también 50 % de traslape para los angulos. Así, se obtienen *chunks* de 4 segundos para

el audio con sus respectivos ángulos. Con ello, si se quiere ingresar un audio completo a la red, se debe segmentar usando ventanas con 50 % de traslape para ingresar los audios con sus respectivos ángulos y, posteriormente, reconstruir el audio juntando los *chunks* de salida de la red.

La red Deep-GEV es una versión reducida de la original GRNN-BF [49]. La ConvTasNET se crea juntando bloques de 3 capas convolucionales con cierto factor incremental de dilatación de kernel (ver figura 2.18). La red Deep-GEV tiene solo 1 bloque de 3 capas convolucionales con dilatación de kernel en la ConvTasNET, en cambio la red GRNN-BF tiene 3 bloques de 3 capas convolucionales con dilatación de kernel. Además, la red Deep-GEV tiene 100 unidades en la red GRU y la red GRNN-BF original tiene 500 unidades. La red Deep-GEV es más chica que la original, debido a que solo se tienen 15 horas de entrenamiento, lo cual es muy poco para que la red converja a una solución. Por ello, se reducen los parámetros de la red.

La red es entrenada con la base de datos BD II Static 1 simulated, usando la función de optimización SI-SNR. El objetivo es maximizar la relación relativa entre la señal y el ruido. Los códigos fueron implementados en *Pytorch* en *Python* [34].

El objetivo de la red Deep-GEV son las señales clean reverberadas (sin ruido), por lo que la red actúa como una red *denoise*, lo que quiere decir que la red elimina el ruido, no quita la reverberación. Esto es debido a que el ASR puede procesar señales reverberadas por como está entrenado y, además, si se entrena la red para dos tareas (desreverberar y denoise), se obtendrán malos resultados debido a que es muy complicado.

3.2.4.3. Red neuronal Deep-MVDR-steering

La red ADL-MVDR se piensa adaptar a las bases de datos con movimiento real [51]. La arquitectura que se usa para realizar los experimentos se puede observar en la figura 3.9.

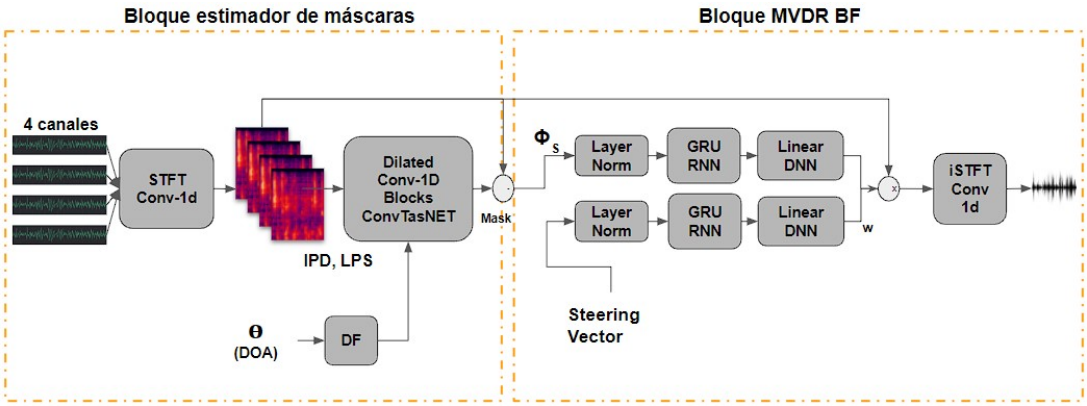

Figura 3.9: Diagrama de bloques red Deep-MVDR-steering.

En la figura 3.9 se puede observar el flujo de información de la red neuronal Deep-MVDR-steering. La red es muy parecida a la red Deep-GEV (figura 3.8). El bloque estimador de máscaras es el mismo, se introducen los 4 canales, se normalizan por potencia relativa, se cal-

cula transformada de Fourier, se usan los mismos *features* y se usa la red CNN ConvtasNET para estimar las máscaras que darán como resultado las matrices de covarianza del *speech* y del ruido. El bloque que cambia es el de *beamforming*. El la red Deep-GEV el bloque RNN propagaba las matrices de covarianzas para calcular los pesos del *beamforming* GEV. En la red Deep-MVDR-steering se tienen dos ramas de RNN GRU, a la primera rama RNN se le ingresa la matriz de covarianza del ruido para ser propagada, a la segunda rama RNN se le ingresa el *steering vector* para ser propagado. La matriz de covarianza del ruido y el *steering vector* se propagan independientemente en RNN separadas y al final de la red, se utiliza la fórmula del MVDR (ecuación 2.8) para calcular los pesos del *beamforming* MVDR. Así, lo único que cambia es el bloque de *beamforming*, todo lo demás queda igual a como la red Deep-GEV. Posterior a aplicar los pesos del *beamforming*, al igual que la red Deep-GEV, se calcula la anti-transformada de Fourier para volver al dominio del tiempo.

A la red Deep-MVDR-steering le ingresan segmentos de 4 segundos de audio y ellos son procesados al interior de la red. A la hora de entrar se toman 4 segundos al azar del audio con sus respectivos ángulos. A la hora de validar y testear, se deben segmentar los audios para poder ser ingresados a la red y, después, reconstruir el audio. A los audios que se usan para validar/testear se les aplica una ventana hamming de 512 muestras con 50 % de traslape y paralelamente a los ángulos se le aplica una ventana rectangular de 512 muestras con 50 % de traslape, con ello se tienen 4 segundos de audio con sus respectivos ángulos. Así, para ingresar un audio completo, primero se debe segmentar los audios con sus ángulos y, posteriormente, juntar los *chunks* de salida de la red para reconstruir el audio.

La red Deep-MVDR-steering es una versión reducida de la red original ADL-MVDR [51]. La ConvTasNET se crea juntando bloques de 3 capas convolucionales con cierto factor incremental de dilatación de kernel (ver figura 2.18). La red Deep-MVDR-steering utiliza 1 bloque de 3 capas convolucionales con dilatación de kernel en la ConvTasNET, en cambio la red ADL-MVDR utiliza 3 bloques de 3 capas convolucionales con dilatación de kernel. Además, la red Deep-MVDR-steering tiene 100 unidades en las redes GRU y la red ADL-MVDR utiliza 500 unidades. La red Deep-MVDR-steering es más chica que la original, debido a que solo se tienen 15 horas de entrenamiento, lo cual es muy poco para que la red converja a una solución. Por ello, se reducen los parámetros de la red.

La red es entrenada con la base de datos BD II Static 2 simulated, usando la función de optimización SI-SNR. El objetivo es maximizar la relación relativa entre la señal y el ruido. Los códigos fueron implementados en *Pytorch* en *Python* [34].

El objetivo de la red Deep-MVDR-steering son las señales clean reverberadas (sin ruido), por lo que la red actúa como una red *denoise*, lo que quiere decir que la red elimina el ruido, no quita la reverberación. Esto es debido a que el ASR puede procesar señales reverberadas por como está entrenado y, además, si se entrena la red para dos tareas (desreverberar y denoise), se obtendrán malos resultados debido a que es muy complicado.

3.2.5. ASR STE-CROS

Usando la misma arquitectura de la red ASR EBT2 [29], se crea un nuevo ASR STE-CROS. Con el mismo procedimiento de entrenamiento de la red ASR EBT2, se entrena este

nuevo ASR usando las señales de la base de datos CrossRever propagadas por la red Deep-MVDR-steering.

Primero, se entrena la red Deep-MVDR-steering con la base de datos CrossRever. Una vez entrenada se propagan las señales de train y dev por la red, obviamente el resultado van a hacer señales sumamente limpias debido a que la red entreno con dichos sets de datos. Es por esto, que a las señales de train y dev propagadas por la red se les suma ruido de restaurant convolucionado y ruido del robot hasta llegar a un SNR de 12dB. Con las señales de train y dev bajadas a un SNR de 12dB se entrena el ASR STE-CROS.

La motivación detrás de esto es debido a que las redes Deep-GEV y Deep-MVDR-steering agregan artefactos a los audios que son propagados por ellas. Por esto, se entrena un ASR con audios que tengan el artefacto que agrega las redes. La red ASR STE-CROS se entrena usando el lenguaje de linux bash.

3.2.6. Métricas de desempeño

La principal métrica para comparar los desempeños de los distintos modelos de *source separation* será el WER y SNR. El WER mide el porcentaje de palabras erróneas en una cierta transcripción, por lo que la métrica permite validar qué modelo es el que obtiene menor WER. Si un modelo tiene menor WER que otro, significa que el modelo pudo realizar su tarea de mejor forma separando las señales del ruido y reverberación para después ser trascribida de una forma más clara y con menos errores. El SNR nos permitirá medir si se ha suprimido lo suficientemente el ruido de los audios, lo que implica en un SNR más elevado (menos ruido más señal).

EL WER se obtendrá con el ASR EBT2 ya entrenado por el LPTV en [29] y con el ASR STE-CROS propiamente entrenado. Con dichos ASR se realizará el *decode* de las señales. El *decode* de una señal significa pasar la señal o espectrograma resultante del modelo al reconocedor y transcribir la señal a texto. Con la señal en texto se podrá calcular la métrica WER para comparar cómo se transcribió la señal y cómo era el texto original. El ASR esta programado con *Kaldi* en bash [36].

Las señales de test de las bases de datos BD II Static 1 simulated, BD II Static 2 simulated, BD_{THRI} y BD_{VS}, serán propagadas por las redes Deep-GEV y Deep-MVDR-steering y se obtendrá su SNR y WER (con ambos ASR). Dichos resultados se comparan con las señales SWD de las bases de datos con las respectivas métricas SNR y WER.

Capítulo 4

Discusión de Resultados

4.1. Resultados

En este capítulo se describirán todos los resultados generados para el desarrollo de la memoria. Se tienen diferentes secciones en donde se detallará los resultados obtenidos.

4.1.1. Replicación resultados investigación LPTV con red TCP/CBP

Estos resultados son producto de la replica de los experimentos que utilizan la red TCP/CBP con la base de datos BD_{sim} [8]. Usando el pre-procesamiento y red CNN TCN/CBP, descrito en la sección de metodologías, se generan las mismas tabla de la referencia mencionada.

En la tabla 4.1 se obtuvieron resultados entrenado el modelo TCN/CBP con los *dataset* Matched, Multicondition, Matched + WPE, Multicondition + WPE y Movil (Time-Varying SNR). Los resultados son bajo el conjunto de test obteniendo el WER EBT2 con el ASR del LPTV.

Tabla 4.1: Tabla replicación resultados usando TCN/CBP con las 5 distintas bases de datos con movimiento simulado y largo de ventana 160, 320 y 640.

Testing Conditition	Analysis window size (frames)		
	160	320	640
RIR- Matched TCN/CBP	4.51 %	4.68 %	4.33 %
RIR- Multicondition TCN/CBP	5.98 %	6.45 %	6.85 %
RIR- Matched TCN/CBP + WPE	4.41 %	4.35 %	4.47 %
RIR- Multicondition TCN/CBP + WPE	5.90 %	5.93 %	6.44 %
Time-Varying-SNR TCN/CBP	6.52 %	6.74 %	7.10 %

En la tabla 4.2 se obtienen resultados al usar entrenar el modelo TCN/CBP solo usando la base de datos con movimiento simulado Multicondition. En dicha tabla se exploran variaciones del modelo usando la concatenación *Compact Billinear Pooling* (original del modelo) versus la concatenación normal (pegar dos vectores). También, se varía el modelo usando solo una rama, rama $B_0(t)$ o $B_1(t)$ que corresponde a usar solo el *beamforming* apuntando a la fuente o ruido, respectivamente (ver figura 3.6). Los resultados son bajo el conjunto de test obteniendo el WER con el ASR del LPTV.

Tabla 4.2: Replicación de resultados base de datos con movimiento simulado Multicondition variando el modelo TCN/CBP y usando largo de ventana 160, 320 y 640.

Testing Condition	*Analysis window size (frames)*		
	160	*320*	*640*
TCN/CBP	5.98 %	6.45 %	6.85 %
TCN/Concat	6.13 %	6.34 %	6.47 %
TCN with B_0(w) only	6.70 %	6.67 %	7.06 %
TCN with B_1(w) only	8.34 %	8.24 %	8.85 %

En la tabla 4.3 se obtienen resultados al usar entrenar el modelo TCN/CBP solo usando la base de datos con movimiento simulado Movil. En dicha tabla se exploran variaciones del modelo usando la concatenación *Compact Billinear Pooling* (original del modelo) versus la concatenación normal (pegar dos vectores). También, se varía el modelo usando solo una rama, rama $B_0(t)$ o $B_1(t)$ que corresponde a usar solo el *beamforming* apuntando a la fuente o ruido, respectivamente (ver figura 3.6). Los resultados son bajo el conjunto de test obteniendo el WER con el ASR del LPTV.

Tabla 4.3: Replicación de resultados base de datos con movimiento simulado Movil variando el modelo TCN/CBP y usando largo de ventana 160, 320 y 640.

Testing Condition	*Analysis window size (frames)*		
	160	*320*	*640*
TCN/CBP	6.52 %	6.74 %	7.10 %
TCN/Concat	6.73 %	6.65 %	6.86 %
TCN with B_0(w) only	7.11 %	6.99 %	7.30 %
TCN with B_1(w) only	9.09 %	9.24 %	9.37 %

4.1.2. Replicación de *beamforming* con base de datos real THRI

Se replican los resultados obtenidos usando *beamforming sum and delay* bajo la base de datos BD_{THRI} [29]. Se utilizan los audios y los ángulos registros en la base de datos para replicar la técnica.

En la tabla 4.4 se replican los resultados usando la base de datos BD_{THRI}. Para ello, se tomaron las 330 señales y se les calcula el WER usando el ASR EBT2 y ASR STE-CROS. Las señales SWD vienen de la sigla *sum without delay*; es puramente sumar los canales sin

hacer desfases. Por ultimo, las señales *Bost* es el algoritmo *sum and delay* usando el DOA.

Por otra parte, se calcula el SNR de cada canal y se ponderan. También, se calcula el SNR de las señales SWD y se calculan los SNR de las señales BOST.

Tabla 4.4: Tabla replicación usando señales SWD (*sum without delay*) y Bost (*sum and delay*) de BD_{THRI}.

Base de datos	SNR por canal [dB]	WER EBT2 SWD [%]	WER STE-CROS SWD [%]	SNR SWD [dB]	WER EBT2 BOST [%]	WER STE-CROS BOST [%]	SNR BOST [dB]
BD II Static 1	4,93	15,73	17,30	7,28	15,82	16,93	7,27
BD II Static 2	3,58	54,12	46,55	0,99	21,32	22,57	6,22
BD II Dynamic 1	4,48	34,75	31,38	4,04	20,06	21,02	6,30
BD II Dynamic 2	3,52	43,12	36,86	3,20	27,39	26,73	5,36
BD III Static 1	3,83	21,24	23,87	4,80	21,54	23,80	4,85
BD III Static 2	4,03	46,78	43,53	2,03	23,73	27,05	5,14
BD III Dynamic 1	3,39	41,51	40,67	2,47	25,54	29,37	4,45
BD III Dynamic 2	3,30	45,99	40,65	2,85	31,59	32,26	4,82

4.1.3. Replicación de *beamforming* con base de datos real VS

Se replicaron los resultados de la base de datos con movimiento real BD_{VS} haciendo *beamforming sum and delay* usando la información visual (DOA) [2].

La tabla 4.5 muestra la replica de resultados usando la base de datos BD_{VS}. Para ello, se toman las 330 señales y se les calcula el WER usando el ASR EBT2 y ASR STE-CROS. Las señales SWD vienen de la sigla *sum without delay*; es puramente sumar los canales sin hacer desfases. Por ultimo, las señales *Bost* es el algoritmo *sum and delay* usando el DOA.

Por otra parte, se calcula el SNR de cada canal y se ponderan. También, se calcula el SNR de las señales SWD y se calculan los SNR de las señales BOST.

Tabla 4.5: Tabla replicación usando señales SWD (*sum without delay*) y Bost (*sum and delay*) de BD_{VS}.

Base de datos	SNR por canal [dB]	WER EBT2 SWD [%]	WER STE-CROS SWD [%]	SNR SWD[dB]	WER EBT2 BOST [%]	WER STE-CROS BOST [%]	SNR BOST [dB]
ST-1	4,93	15,90	17,07	7,28	15,82	17,09	7,28
ST-2	3,83	21,22	23,87	4,81	21,17	23,84	4,85
Mov-1	0,64	45,75	43,08	1,84	40,58	38,33	2,27
Mov-2	1,50	51,15	45,62	2,14	45,10	40,37	2,67
VS-Mov-1	2,68	30,19	28,81	4,37	29,85	28,79	4,39
VS-Mov-2	2,26	33,23	29,55	3,93	32,88	29,35	3,95

4.1.4. Resultados de red Deep-GEV/Deep-MVDR-steering sobre base de datos real

Se entrenó la red Deep-GEV usando la base de datos simulada BD II Static 1 simulated. La red Dee-MVDR-steering se entrenó usando la base de datos simulada BD II Static 2 simulated.

Las redes fueron entrenadas con 200 epocas, *learning rate* de 0.0001 y *batch size* de 6. Bajo esas condiciones la red tuvo el siguiente desempeño.

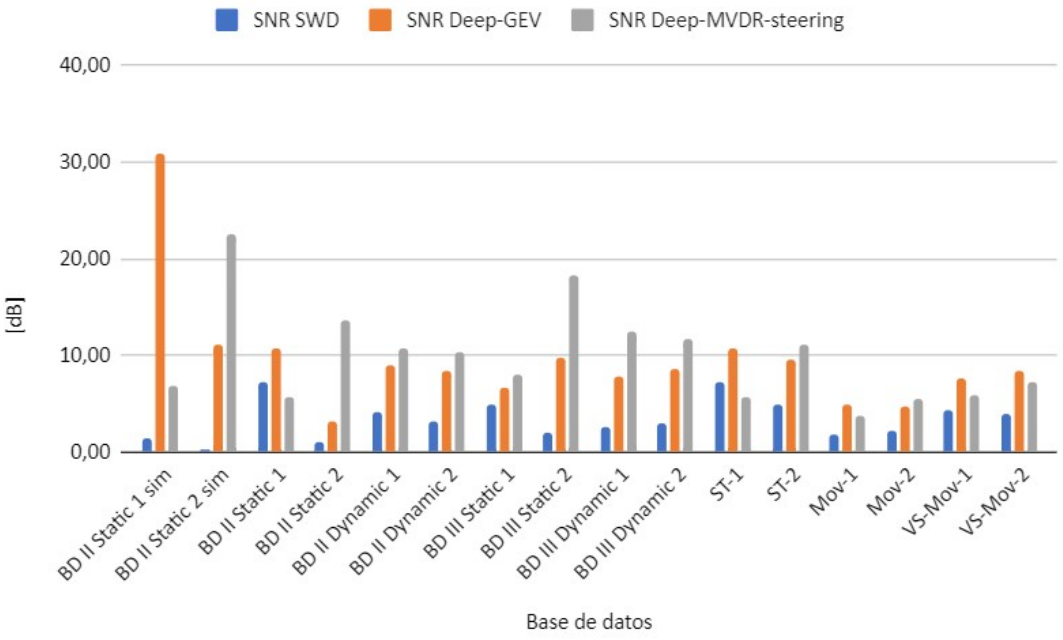

Figura 4.1: Resultados SNR de redes Deep-GEV y Deep-MVDR-steering

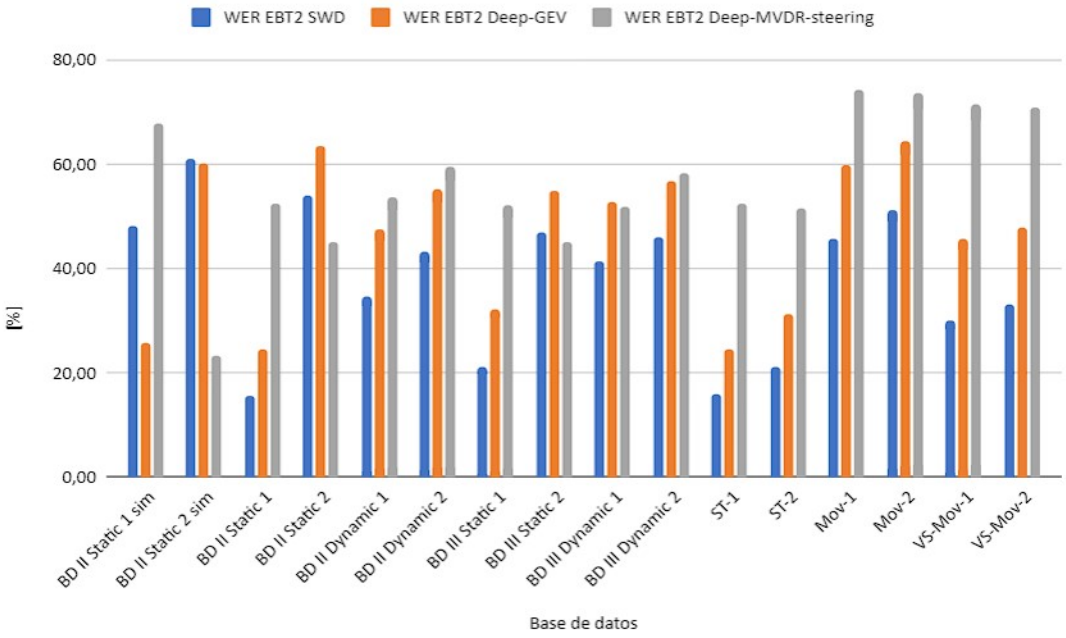

Figura 4.2: Resultados WER EBR2 de redes Deep-GEV y Deep-MVDR-steering

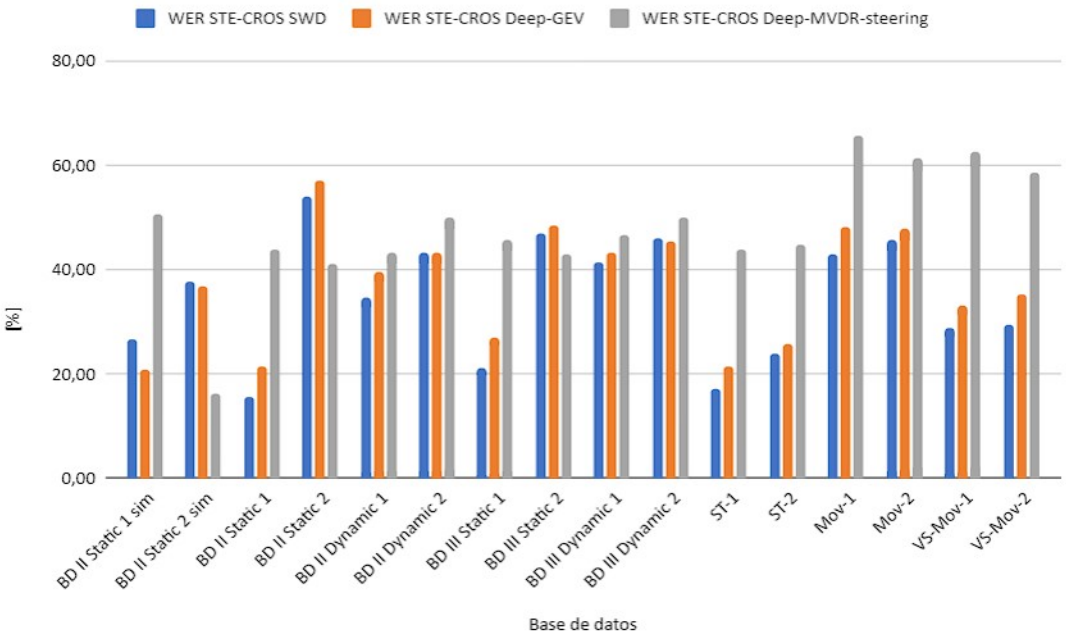

Figura 4.3: Resultados WER STE-CROS de redes Deep-GEV y Deep-MVDR-steering

Para las figuras 4.1, 4.2 y 4.3 se puede notar que se tiene el SNR/WER de la propagación de las distintas bases de datos por las redes Deep-GEV y Deep-MVDR-steering. La figura 4.2 tiene el WER EBT2 de las señales propagadas por las redes y la figura 4.3 tiene el WER STE-CROS de las señales propagadas por las redes. Además, se adjunta a los gráficos los SNR/WER de las distintas bases de datos en modo SWD (*sum without delay*), o sea, sumar los canales directamente, así se tiene un punto de comparación.

4.2. Análisis de resultados

De los resultados anteriores se pueden extrapolar distintas conclusiones. Los resultados de la red TCN/CBP es completamente diferente a los resultados de las redes Deep-GEV y Deep-MVDR-steering. La red TCN/CBP tiene muy buenos resultados, debido a que solo se prueba su desempeño con bases de datos simuladas. Estas bases de datos simuladas (BD_{sim}) son aproximaciones de un escenario real, por lo cual es menos compleja la base de datos y la red obtiene mejores resultados. Por otra parte, la base de datos simulada (BD_{sim}) tiene solo un *subdataset* con movimiento simulado, el cual se logra variando el SNR, lo cual es una aproximación muy grande de un escenario con movimiento real, pues falta la reverberación, ruido, atenuaciones, etc. Debido a esto, la red TCN/CBP logra en promedio un WER EBT2 de 6 % en la base de datos simulada (ver tabla 4.1). También, las variaciones de la red TCN/CBP dan buenos resultados en las bases de datos simuladas Multicondition (tabla 4.2) y Movil (tabla 4.3). Dichos resultados dejan en evidencia que variar la red TCN/CBP no afecta mucho los resultados de la red original, lo cual no es bueno debido a que la hipótesis es que usar *Compact Billinear Pooling* es una mejor técnica de concatenación. Pero, al parecer, la técnica CBP no está aportando como se tenia esperado. Además en las ta-

blas 4.2 y 4.3, se puede notar que usar una rama o la otra sí tiene consecuencias, debido a que si se usa la rama con el *beamforming*, apuntando a la fuente, tiene un 2 % de WER EBT2 menos que usar la rama con el *beamforming* apuntando al ruido. Esto tiene sentido debido a que la fuente aporta más información respecto a lo que se quiere escuchar que el ruido.

En cambio, la red Deep-GEV y Deep-MVDR-steering logran resultados inesperados. Si se observa la figura 4.1 podemos observar que ambas redes tienen buenos resultados cuando son probadas en las bases de datos de test BD II Static 1 y 2 simuladas. Esto tiene coherencia debido a que las red Deep-GEV fue entrenada con la base de datos BD II Static 1 simulated y la red Deep-MVDR-steering fue entrenada con la base de datos BD II Static 2 simulated. Ambas redes tienen SNR muy altos en los respectivos conjuntos de test simulados. Pero hay un cambio brusco cuando las redes son probadas en las bases de datos reales BD_{THRI} y BD_{VS}. Si bien el SNR siempre sube con respecto al SNR de SWD, no sube tanto como cuando se prueba en las bases Static 1 y Static 2 simuladas de test.

Al ver la figura 4.2, a nivel de WER EBT2 se puede observar que las redes tienen un buen desempeño en los casos de las bases de datos de test BD II Static 1 y 2 simuladas. Ya que, el WER EBT2 de las redes es bastante menor al WER EBT2 de SWD. En cambio, a nivel de las bases de datos con movimiento real, no hay una mejora: el WER EBT2 de las redes es más elevado a el WER EBT2 de SWD. Esto quiere decir que en general para las bases de datos BD_{THRI} y BD_{VS}, las redes no esta aportando para que el WER EBT2 baje con respecto a las señales SWD.

Si se comparan la red Deep-GEV versus la red Deep-MVDR-steering, se puede notar que a nivel de SNR la red Deep-MVDR-steering supera en promedio en 3dB a la Deep-GEV en casi todas las bases de datos. Pero a nivel de WER EBT2, se puede observar un comportamiento anómalo, debido a que si bien la red Deep-MVDR-steering aumenta el SNR más que la red Deep-GEV, a nivel de WER EBT2 la red Deep-GEV tiene en promedio un 11 % menos de WER ETB2 que la red Deep-MVDR-steering.

Por otra parte, si analizamos el comportamiento de las redes Deep en las bases de datos con movimiento BD_{THRI} Y BD_{VS}, se puede notar que la red Deep-GEV tiene mejores resultados en los casos que las bases de datos sean más parecidas a la base de datos de entrenamiento de la red (BD II Static 1 simulated). Esos casos serían las bases de datos BD II Static 1, BD III Static 1, ST-1 y ST-2, VS-Mov-1 y VS-Mov2 debido a que el robot esta mirando (o sigue) a la fuente de *speech*. En dichos casos la red Deep-GEV siempre supera en SNR/WER (EBT2) a la red Deep-MVDR-steering pero, particularmente, en los casos de BD III Static 1 y ST-2 la red Deep-MVDR-steering supera solamente en SNR a la red Deep-GEV. Esto se puede dar debido a que al tener dos fuentes de ruido la red que fue entrenada mirando a la fuente de ruido (Deep-MVDR-steering) puede procesar mejor los audios. En cambio la red Deep-MVDR-steering obtiene mejores resultados en los casos que las bases de datos sean más parecidas a la base de datos de entrenamiento de la red (BD II Static 2 simulated). Esos casos serían las bases de datos BD II Static 2 y BD III Static 2. En dichas bases de datos la red Deep-MVDR-steering obtiene un menor WER EBT2 que su contrincante y logra aumentar el SNR mucho más que la red Deep-GEV. En los casos de desplazamiento, como lo es DB II Dynamic 1, BD II Dynamic 2, Mov-1 y Mov-2, la red Deep-GEV supera en general en WER EBT2 a la red Deep-MVDR-steering pero, a nivel de SNR, la red Deep-MVDR-steering tiene

mejores resultados (SNR más alto).

La perspectiva general es que la redes tienen un mejor comportamiento en los casos estáticos que son parecidos a las bases de entrenamiento de las redes. En promedio para los casos estáticos de la red Deep-GEV (DB II Static 1, DB III Static 1, ST-1 y ST-2) la red obtuvo un SNR de 9dB y un WER EBT2 de 28 %. En cambio para los casos estáticos de la red Deep-MVDR-steering (DB II Static 2 y DB III Static 2) la red obtuvo, en promedio, un SNR de 15dB y WER EBT2 de 45 %. En cambio, vemos que ambas redes se degradan al probarse en los casos dinámicos como lo son DB II Dynamic 1, DB II Dynamic 2, DB III Dynamic 1, DB III Dynamic 2, Mov-1, Mov-2, VS-Mov-1 y VS-Mov-2, dando en promedio la red Deep-GEV un SNR de 7dB y WER EBT2 de 50 % y la red Deep-MVDR-steering un SNR de 8dB y WER EBT2 de 64 %.

Esto da la impresión que hay un inconsistencia o *missmatch*, debido a que si bien la red está aportando y mejora las señales a nivel de SNR, a nivel de WER EBT2 no está haciendo nada. O sea, la red genera una ganancia significativa de SNR, pero el WER EBT2 no mejora. Al ver un ejemplo de las tablas de SNR y WER EBT2 se puede notar, por ejemplo, que la red Deep-GEV en la base de datos BD II Dynamic 2 alcanza un SNR de 8,46 dB y SWD tiene 3,20 dB. En cambio, en WER EBT2 la red alcanza en 55,33 % y SWD un 43,12 %. Esto significa que a nivel de SNR la red mejora la señal en 5,25 dB, pero a nivel de WER EBT2 empeora en 12,21 %. Esto no tiene sentido, debido a que el SNR de las señales propagadas por la res es bastante superior a las señales SWD, pero la métrica WER EBT2 da resultados peores que SWD. Lo que se menciona es un ejemplo, pero si se observa el gráfico se puede notar que siempre el SNR sube y el WER EBT 2 empeora, lo cual es una contradicción.

Debido a esto, es de pensar que el problema viene con el ASR que se esta usando. Las redes están introduciendo un artefacto a los audios, que hace que el ASR EBT2 no sea capaz de transcribirlas. Por esto se utiliza el ASR STE-CROS que fue entrenado con señales propagadas por la red con un SNR más bajo. Al ver la figura 4.3 notamos que el WER STE-CROS baja en comparación con el WER EBT2 (figura 4.2). En promedio al usar el WER STE-CROS baja un 8 % el WER, lo que significa que este nuevo ASR puede transcribir mejor las señales. Aún así, se siguen observando los mismo comportamiento que con el WER EBT2, debido a que se mantienen las mismas relaciones en los desempeños de las redes Deep-GEV y Deep-MVDR-steering. Es decir, se sigue viendo el comportamiento que la red Deep-GEV tiene mejor desempeño que su contrincante en las bases de datos que son parecidas a la base de entrenamiento BD II Static 1 simulated y las red Deep-MVDR-steering tiene mejor desempeño que su contrincante en las bases de datos que son parecidas a la base BD II Static 2 simulated. De todas formas, las redes tienen un desempeño óptimo solamente en en las respectivas bases de test BD II Static 1 y 2 simulated. En general las redes tienen WER STE-CROS más alto que el WER STE-CROS de las señales SWD, lo cual es un problema.

A nivel de SNR las señales de las redes superan a SWD, pero en WER EBT2 y WER STE-CROS no. Particularmete para las bases de datos reales, la red Deep-GEV supera a SWD en WER EBT2 en BD III Dynamic 2, por muy poco. En cambio, la red Deep-MVDR-steering supera a SWD en WER (ambos ASR) en BD II Static 2 y BD III Static 2. Dichos son casos particulares, en general las redes no pueden superar el WER de SWD, de hecho lo ideal sería que las señales de las redes superaran al algoritmo *sum and delay* en SNR y WER

(ver tabla 4.4 y tabla 4.5, respectivamente, en la columna BOST). Esto da indicio, que hay un problema de generalización, las redes tienen un comportamiento genial en sus propias bases de test (BD II Static 1 y 2 simulated) superando a SWD y BOST tanto en SNR como en WER EBT2 y STE-CROS. Sin embargo, en las bases de datos con movimiento real BD_{THRI} y BD_{VS} tiene pésimos resultados. Este comportamiento es solo explicado debido a que la red no es buena extrapolando, teniendo solo buenos resultados en bases de datos parecidas a las suyas.

Capítulo 5

Conclusiones

A modo de conclusión se puede decir que se cumplieron los objetivos del trabajo de titulo, debido a que se logró implementar un sistema que pudiera separar la señal objetivo de la interferencia ambiental. Además, se pudieron cumplir los objetivos específicos como encontrar una base de datos adecuada (BD_{THRI} Y BD_{VS}), se generaron algoritmos de *deep learning* (TCN/CBP, Deep-GEV y Deep-MVDR-steering), se generaron técnicas de *beamforming* (*sum and delay*) y, finalmente, se evaluó la calidad de la señal con un ASR y métricas de la calidad de voz (SNR).

Los resultados de la red TCN/CBP son lo esperado, al entrenar y testear la red solo en bases simuladas BD_{sim} la red obtiene muy buenos resultados. Además, las variantes de la red siguen obteniendo un WER EBT2 cercano al 6 %. Esto es debido a que la simulación es una simplificación del escenario real, lo cual hace que sea un ambiente menos complejo y más fácil para que la red puede procesar y limpiar las señales acústicas. Además, se comprueba que el método CBP no tiene ninguna ventaja frente a la concatenación común y corriente.

Con respecto a las redes Depp-GEV y Deep-MVDR-steering, se puede concluir que bajo un conjunto de datos simulados tiene muy buenos resultados a nivel de SNR/WER al igual que TCN/CBP (que es lo esperado), debido a que supera extraordinariamente a SWD. Pero el desempeño con las bases de datos con movimiento real tienen otro efecto, debido a que la red Deep-GEV obtiene buenos resultados en comparación con su contrincante en las bases de datos que son parecidas a la base de datos de entrenamiento de estas, las cuales son BD II Static 1, BD III Static 1, ST-1, ST-2, VS-Mov-1 VS-Mov-2. Esto es debido a que en dichas bases de datos se mira (o sigue) a la fuente de *speech*. En cambio, la red Deep-MVDR-steering tiene buenos resultados en comparación con su contrincante en las bases de datos que se parecen a su base de datos de entrenamiento, las cuales son BD II Static 2, BD III Static 2. Para el caso del movimiento, sin incluir visual servoing (BD II Dynamic 1, BD II Dynamic 2, BD III Dynamic 1, BD III Dynamic 2, Mov-1, Mov-2), la red Deep-GEV supera en general en WER a la red Deep-MVDR-steering, pero la red Deep-MVDR-steering tiene SNR más alto.

Bajo lo anterior, se puede extrapolar que ambas redes están aportando algo cuando se prueban con las respectivas bases estáticas reales, lo cual tiene sentido debido a que las redes fueron entrenadas con una base simulada de igual naturaleza. Esto deja claro, que ambas redes tienen problemas de generalización, debido a que solamente se adaptan relativamente bien a una situación similar a la del entrenamiento. El problema de generalización es nor-

mal en las redes neuronales debido a que las redes son buenas interpolando y no extrapolando.

Para el caso de las base con movimiento lateral o hacia adelante/atrás (sin visual servoing), se puede notar un comportamiento inusual, debido a que una red supera a otra en dos diferentes métricas (la red Deep-GEV obtiene mejor WER, pero la red Deep-MVDR-steering obtiene mejor SNR). Lógicamente, la red Deep-GEV debería dar mejores resultados debido a que tiene más grados de libertad (mayor cantidad de parámetros), pero puede pasar que al no ser una base de entrenamiento suficientemente grande, la red no alcance a entrenarse bien o los datos no están siendo representativos para que la red entrene bien.

De todas manera lo que es evidente es que, si bien el SNR de ambas redes siempre supera al SNR de SWD, siempre el WER de las redes da más elevado que el WER SWD (para las bases de datos con movimiento real). Se probó la métrica WER con dos ASR y ninguno tuvo mejores resultados que SWD. Con el ASR STE-CROS se confirma el hecho que había un artefacto, debido a que bajó bastante el WER, pero no alcanzó a superar a SWD y ni siquiera cerca de superar a BOST (*sum and delay*). Al probar las redes en otras bases de datos no simuladas, están siendo capaz de suprimir ruido, debido a que si hay un aumento de SNR, pero no procesan de manera adecuada la señal, haciendo que el WER arroje malos resultados. Esto confirma el problema de generalización de las redes, siento incapaces de extrapolar a otras bases de datos.

Como se sabe que las redes neuronales tienen problemas de generalización, la redes deben ser entrenadas en condiciones similares a las de test, para obtener buenos resultados. Cuando las redes son entrenadas y testeadas en las mismas condiciones se puede ver que la red Deep-GEV, Depp-MVDR-steering y TCN/CBP obtiene buenos resultados en su respectivas bases de test simuladas. La hipótesis al simular las bases BD II Static 1 y 2 simulated, era que se estaba representando bien el ruido, la sala, las RIR, etc. La gran diferencia entre la simulación que se generó y el escenario real son las RIR que se utilizaron para reverberar el ruido. Las RIR que se utilizaron para reverberar están calculadas desde la fuente de *speech* en 0^{o}, no se tienen las RIR del ruido. Esto hace que la densidad espectral de potencia del ruido cambie en comparación al ruido real. Dando a entender que la simulación no fue una buena representación del escenario real, lo que hizo que las redes obtuvieran malos resultados en las bases de datos reales. Las redes se pudieron adaptar relativamente mejor a los casos estáticos reales que se parecen a las bases de entrenamiento simuladas, pero de todas formas el comportamiento es insatisfactorio para los casos estáticos y dinámicos reales por no superar en métricas a SWD.

Como trabajos futuros, para mejorar el desempeño de la red y que pueda superar en WER a SWD y BOST, se propone tres opciones. La primera opción y la más directa es grabar/generar una base de datos de entrenamiento con movimiento real de acuerdo a las condiciones de test que se quiera probar. La segunda opción es grabar las RIR del ruido que faltan para realizar una simulación lo más real posible. La ultima opción sería simular RIR del *speech* y ruido con alguna librería que pueda lidiar con movimiento. Simular siempre requerirá menos trabajo que grabar una base de datos real, por lo que se recomienda tratar de implementar una simulación lo más real posible y una vez obtenido resultados prometedores grabar las RIR o directamente el escenario real de la base de datos de entrenamiento que se quiere.

Bibliografía

[1] Anguera, X., Wooters, C., Hernando, J., 2007. "Acoustic beamforming for speaker diarization of meetings", IEEE Transactions on Audio, Speech and Language Processing, volume 15, number 7, pp.2011-2023.

[2] Becerra, N., Díaz, A., Mahu, R., Novoa, J., Wuth, J., Data, J., 2020. "Assessing the effect of visual servoing on the performance of linear microphone arrays in moving human-robot interaction scenarios", International Speech Communication Association (ISCA), online.

[3] Calvo, D., 2017. "Red Neuronal Convolucional CNN", Diego Calvo, https://www.diegocalvo.es/red-neuronal-convolucional/.

[4] Chen, H., Zhang, P., Shi, Q., Liu, Z., 2020. "Improved Guided Source Separation Integrated with a Strong Back-End for the CHiME-6 Dinner Party Scenari", Interspeech 2020, Shangai, China.

[5] Chen, J., Mao, Q., Liu, D., 2020. "Dual-Path Transformer Network: Direct Context-Aware Modeling for End-to-End Monaural Speech Separation", Interspeech 2020, Shangai, China.

[6] Chen, J., Mao, Q., Liu, D., 2020. "On Synthesis for Supervised Monaural Speech Separation in Time Domain", Interspeech 2020, Shangai, China.

[7] Chen, Z., Xiao, X., Yoshioka, T., Erdogan, H., Li, J., Gong, Y., 2018. "Multi-Channel Overlapped Speech Recognition with Location Guided Speech Extraction Network", IEEE Spoken Language Technology Workshop (SLT), Athens, Greece.

[8] Díaz, A., Pincheira, D., Mahu, R., Becerra, N., 2020. "Short-Time Deep-Learning based source separation for speech enhancement in reverberant environments with beamforming", arXiv preprint arXiv:2011.01965.

[9] Deng, C., Zhang, Y., Ma, S., Sha, Y., Song, H., Li, X., 2020. "Conv-TasSAN: Separative Adversarial Network Based on Conv-TasNet", Interspeech 2020, Shangai, China.

[10] Fan, C., Tao, J., Liu, B., Yi, J., Wen, Z., 2020. "Gated Recurrent Fusion of Spatial and Spectral Features for Multi-Channel Speech Separation with Deep Embedding Representations", Interspeech 2020, Shangai, China.

[11] Garofolo, J., et al., 1993. "TIMIT Acoustic-Phonetic Continuous Speech Corpus LDC93S1", Web Download, Philadelphia: Linguistic Data Consortium.

[12] Ge, M., Xu, C., Wang, L., Siong Chng, E., Dang, J., Li H., 2020. "SpEx+: A Complete Time Domain Speaker Extraction Network", Interspeech 2020, Shangai, China.

[13] Golokolenko, O., Schuller G., 2020. "The Method of Random Directions Optimization for Stereo Audio Source Separation", Interspeech 2020, Shangai, China.

[14] Grondin, F., Lauzon, J., Vincent, J., Michaud, F., 2020. "GEV Beamforming Supported by DOA-Based Masks Generated on Pairs of Microphones", Interspeech 2020, Shangai, China.

[15] Hadad, E., Heese, F., Vary, P., Gannot, S., 2014. "Multichannel audio database in various acoustic environments", 14th International Workshop on Acoustic Signal Enhancement (IWAENC), pp. 313–317.

[16] Hao, Y., Xu, Y., Shi, J., Zhang, P., Qin, L., Xu, B., 2020. "A Unified Framework for Low-L atency Speaker Extraction in Cocktail Party Environments", Interspeech 2020, Shangai, China.

[17] Hershey, J., Chen, Z., Le Roux, J., Watanabe, S., 2016. "Deep clustering: Discriminative embeddings for segmentation and separation", 2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE.

[18] Hinton, G., Deng, L., Yu, D., Dahl, G., Mohamed, A., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Kingsbury, B., et al. 2012. "Deep neural networks for acoustic modeling in speech recognition", IEEE Signal processing magazine.

[19] Hiroe, A., 2020. "Similarity-and-Independence-Aware Beamformer: Method for Target Source Extraction Using Magnitude Spectrogram as Reference", Interspeech 2020, Shangai, China.

[20] Jun. 2020. "[DL] 13. Convolution and Pooling Variants (Dilated Convolution, SPP, ASPP)", Medium, https://medium.com/jun-devpblog/dl-13-convolution-and-pooling-variants-dilated-convolution-spp-aspp-a954a282ff5c.

[21] Kinoshita, K., von Neumann, T., Delcroix, M., Nakatani, T., Haeb-Umbach, R., 2020. "Multi-Path RNN for Hierarchical Modeling of Long Sequential Data and its Application to Speaker Stream Separation", Interspeech 2020, Shangai, China.

[22] Li, C., Qian, Y., 2020. "Listen, Watch and Understand at the Cocktail Party: Audio-Visual-Contextual Speech Separation", Interspeech 2020, Shangai, China.

[23] Li, G., Liang, S., Nie, S., Liu, W., Yang, Z., Xiao L., 2020. "Microphone Array Post-Filter for Target Speech Enhancement Without a Prior Information of Point Interferers", Interspeech 2020, Shangai, China.

[24] Li, T., Lin, Q., Bao, Y., Li, M., 2020. "Atss-Net: Target Speaker Separation via Attention-based Neural Network", Interspeech 2020, Shangai, China.

[25] Luo, Y., Mesgarani, N., 2019. "Conv-TasNet: Surpassing Ideal Time-Frequency Magnitude Masking for Speech Separation", IEEE/ACM Transactions on Audio, Speech, and Language Processing, website.

[26] Luo, Y., Mesgarani, N., 2020. "Separating Varying Numbers of Sources with Auxiliary Autoencoding Loss", Interspeech 2020, Shangai, China.

[27] Nakagome, Y., Togami, M., Ogawa, T., Kobayashi, T., 2020. "Mentoring-Reverse Mentoring for Unsupervised Multi-channel Speech Source Separation", Interspeech 2020, Shangai, China.

[28] Narayanaswamy, V., Thiagarajan, J., Anirudh, R., Spanias A., 2020. "Unsupervised Audio Source Separation Using Generative Priors", Interspeech 2020, Shangai, China.

[29] Novoa, J., Mahu, R., Wuth, J., Escudero, J.., Fredes, J., Becerra, N., 2021. "Automatic Speech Recognition for Indoor HRI Scenarios", ACM Transactions on Human-Robot Interaction (THRI).

[30] Nakatani, T., Yoshioka, T., Kinoshita, K., Miyoshi, M., Juang, B., 2010. "Speech dereverberation based on variance-normalized delayed linear prediction", IEEE Transactions on Audio, Speech, and Language Processing, vol. 18, no. 7, pp. 1717-1731.

[31] Ochiai, T., Delcroix, M., Koizumi, Y., Ito, H., Kinoshita, K., Araki, S., 2020. "Listen to What You Want: Neural Network-Based Universal Sound Selector", Interspeech 2020, Shangai, China.

[32] Panayotov, V., Chen, G., Povey, D., Khudanpur, S., 2015. "Librispeech: an asr corpus based on public domain audio books", IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, pp. 5206–5210.

[33] Pariente, M., Cornell, S., Cosentino, J., Sivasankaran, S., Tzinis, E., Heitkaemper, J., Olvera, M., Stöter, F., Hu, M., Martín-Doñas, J., Ditter, D., Frank, A., Deleforge, A., Vincent, E., 2020. "Asteroid: The PyTorch-Based Audio Source Separation Toolkit for Researchers", Interspeech 2020, Shangai, China.

[34] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S., 2019. "PyTorch: An Imperative Style, High-Performance Deep Learning Library", Advances in Neural Information Processing Systems 32, p 8024–8035.

[35] Pearce, D., Picone, J, 2002. "Aurora working group: Dsr front end lvcsr evaluationau/384/02", Inst. for Signal and Inform. Process Mississippi State Univ..

[36] Povey, D., Ghoshal, A., Boulianne, G., Burget, L., Glembek, O., Goel, N., Hannemann, M., 2011. "The kaldi speech recognition toolkit", In Proceedings of the IEEE Workshop on Automatic Speech Recognition and Understanding.

[37] Qu, L., Weber, C., Wemter, S., 2020. "Multimodal Target Speech Separation with Voice and Face References", Interspeech 2020, Shangai, China.

[38] Redmon, J. Divvala, S., Girshick, R., Farhadi, A., 2016. "You Only Look Once: Unified, Real-Time Object Detection", 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 779-788, doi: 10.1109/CVPR.2016.91.

[39] Saha, S., 2018. "A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way", Towards Data Science, https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53.

[40] Scheibler, R., 2020. "Generalized Minimal Distortion Principle for Blind Source Separation", Interspeech 2020, Shangai, China.

[41] Scheibler, R., Bezzam, E., Dokmanić, I., 2018. "Pyroomacoustics: A Python package for audio room simulations and array processing algorithms", Proc. IEEE ICASSP, Calgary, CA.

[42] Thiemann, J., Ito, N., Vincent, E., 2013. "The diverse environments multi-channel acoustic noise database (DEMAND): A database of multichannel environmental noise recordings", The Journal of the Acoustical Society of America, vol. 133, p. 3591.

[43] Versloot, C., 2020. "Using Constant Padding, Reflection Padding and Replication Padding with TensorFlow and Keras", MACHINECURVE, https://www.machinecurve.com/index.php/2020/02/10/using-constant-padding-reflection-padding-and-replication-padding-with-keras/.

[44] Vesely, K., Ghoshal, A., Burget, L., Povey, D., 2013. "Sequence-discriminative training of deep neural networks", In Interspeech, volume 2013, pages 2345–2349.

[45] Wang, J., 2020. "Learning Better Speech Representations by Worsening Interference", Interspeech 2020, Shangai, China.

[46] Wang, Z., Na, Y., Liu, Z., Li, Y., Tian, B., Fu, Q., 2020. "A Semi-Blind Source Separation Approach for Speech Dereverberation", Interspeech 2020, Shangai, China.

[47] Watanabe, S., Mandel, M., Barker, J., Vincent, E., Arora, A., Chang, X., Khudanpur, S.,Manohar, V., Povey, D., Raj, D., Snyder, D., Subramanian, A., Trmal, J., Yair, B., Boeddeker, C., Ni, Z., Fujita, Y., Horiguchi, S., Kanda, N., Yoshioka, T., Ryant, N., 2020. "CHiME-6 Challenge:Tackling Multispeaker Speech Recognition for Unsegmented Recordings", The 6th International Workshop on Speech Processing in Everyday Environments.

[48] Xu, J., Hu, K., Xu, C., Chung Tran, D., Wang, Z., 2020. "Speaker-Aware Monaural Speech Separation", Interspeech 2020, Shangai, China.

[49] Xu, Y., Zhang, Z., Yu, M., Zhang, S., Yu, D., 2021. "Generalized Spatio-Temporal RNN Beamformer for Target Speech Separation", Interspeech 2021, Brno, Czechia.

[50] Yasuda, M., Ohishi, Y., Koizumi, Y., Harada, N., 2020. "Crossmodal Sound Retrieval Based on Specific Target Co-Occurrence Denoted with Weak Labels", Interspeech 2020, Shangai, China.

[51] Zhang, Zh., Xu, Y., Yu M., , Zhang, Sh, Chen, L., Yu D., 2021. "ADL-MVDR: All deep learning MVDR beamformer for target speech separation", Tencent AI Lab, Shenzhen, China.

[52] Zhang, Zi., He, B., Zhang, Zh., 2020. "X-TaSNet: Robust and Accurate Time-Domain Speaker Extraction Network", Interspeech 2020, Shangai, China.

[53] Zhao, J., Gao, S., Shinozaki, T., 2020. "Time-Domain Target-Speaker Speech Separation with Waveform-Based Speaker Embedding", Interspeech 2020, Shangai, China.