UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

PROGRAMA DE POSGRADO EN CIENCIA E
INGENIERÍA DE LA COMPUTACIÓN

SEPARACIÓN DE VOZ UTILIZANDO MODELOS DE APRENDIZAJE PROFUNDO: ANÁLISIS DEL RENDIMIENTO EN LÍNEA

T E S I S

QUE PARA OPTAR POR EL GRADO DE:

MAESTRO EN CIENCIA E INGENIERÍA DE LA COMPUTACIÓN

PRESENTA:

FÉLIX SÁNCHEZ MORALES

DIRECTOR DE TESIS:

DR. CALEB ANTONIO RASCON ESTEBANÉ
IIMAS, UNAM

CIUDAD DE MÉXICO, ENERO 2024

A mi familia, gracias por su amor incondicional, por estar en los mejores y perores momentos.

A Mar, por todo el amor que me ha dado.

Agradecimientos

Quiero agradecerle a Dios por permitirme estar en este punto de mi vida.

Agradezco al Instituto de Investigaciones en Matemáticas Aplicadas y en Sistemas de la Universidad Nacional Autónoma de México por haberme abierto sus puertas durante mi vida académica.

A mi tutor el Dr. Caleb Rascón Estebané, le agradezco por todo el apoyo que me brindo durante el desarrollo de este proyecto. Las numerosas charlas y consejos fueron fundamentales para la culminación de este trabajo. Muchas gracias por mostrarme lo interesante y divertido que es el mundo del audio y más cuando se comparte con muchas personas.

Agradezco a mis sinodales, el Dr. Jorge Pérez González, el Dr. Gibran Fuentes Pineda, el Dr. Erik Molino Minero y la Dra. Lucía Medina Gómez. Sus observaciones fueron de gran ayuda para mejorar mi trabajo.

Agradezco a mis profesores por haberme brindado su orientación y conocimiento a lo largo de mi trayectoria académica. Incluso cuando el mundo pasaba por una pandemia siempre se esforzaron por brindar educación de calidad, adaptándose a las condiciones adversas con creatividad y empatía.

Les agradezco a mis padres por el apoyo incondicional que siempre me han brindado. Sin sus enseñanzas no seria la persona que soy hoy, gracias por todo el amor que me dan.

A mis hermanas por haberme ayudado a crecer como persona, su influencia y apoyo me han permitido continuar en los retos que se han presentado en mi vida.

A mis sobrinitos que me han traído mucha felicidad, les agradezco por llenar de risas y de momentos inolvidables mi vida. Espero algún día poder compartir con ustedes las experiencias, valores y conocimientos que tengo.

A Mar, mi compañera de vida, gracias por todo el apoyo comprensión y cariño que me brindas día tras día.

A Uriel, por la gran amistad y por todo el apoyo y enseñanza que me ha dado.

A mis tios y mi primo Mario, muchas gracias por brindarme un segundo hogar, por su apoyo incondicional.

Expreso mi profundo agradecimiento al Consejo Nacional de Humanidades, Ciencias y Tecnologías(CONAHCYT) por la beca que me fue otorgada durante mi maestría.

Al Programa de Apoyo a Proyectos de Investigación e Innovación Tecnológica (PAPIIT) de la UNAM con clave IA100222.

Resumen

La implementación e incorporación de modelos para la separación y mejora del habla que sean capaces de trabajar con recursos computacionales limitados es un paso esencial para mejorar la comunicación en línea. En la literatura, múltiples modelos logran efectuar la separación y mejora del habla al recibir una secuencia de audio completa, sin embargo, muchos de estos modelos no han sido puestos a prueba frente a escenarios donde deban realizar dicha tarea en línea.

El poder asegurarse de que un modelo es robusto ante variaciones en la longitud de entrada que recibe es uno de los primeros pasos que se deben de hacer para una caracterización de su funcionamiento en línea. No obstante, no solo es necesaria la capacidad de procesar de manera eficiente fragmentos cortos del habla, sino también mantener un rendimiento óptimo cuando se le pone a prueba con una señal más compleja. En este trabajo se presenta una caracterización de los modelos ConvTasNet, Sudo rm -rf y Sepformer para la separación del habla en línea y Demucs-Denoiser, DCCRNet y MLT mimic loss para la mejora del habla en línea. Cada uno de estos modelos ha sido evaluado en términos de la razón señal-interferencia en la separación, el tiempo de respuesta en función de la longitud de secuencia de entrada y del factor de tiempo real.

Con respecto a la tarea de separación del habla, todos los modelos evaluados muestran una capacidad para reducir el ruido y mitigar la presencia de otras fuentes, sin embargo, debido a la gran cantidad de parámetros del modelo Sepformer, este presenta un factor de tiempo real mayor a 1 para todas las longitudes de secuencia de entrada probadas, indicándonos que no es apto para procesamiento en línea. El modelo que mejor resultados brindo al trabajar en línea fue Sudo rm -rf seguido de la arquitectura ConvTasNet. En cuanto los modelos de mejora del habla evaluados, se obtuvo que todos son aptos para tareas de procesamiento en línea. Demucs-Denoiser ha sido el que ha presentado mejores resultados en cuanto a la evaluación de la razón señal-interferencia y también ha sido el más robusto ante la variación en la longitud de la secuencia de entrada.

Abstract

The implementation and incorporation of models for speech separation and speech enhancement that are capable of working with limited computational resources is an essential step towards improving online communication. In the literature, multiple models have been successful in performing speech separation and enhancement when given a complete audio sequence, however, many of these models have not been tested in scenarios where they need to perform this task online.

Ensuring that a model is robust to variations in the input length it receives is one of the first steps that should be taken to characterize its performance online. However, it is not only necessary for the model to efficiently process short speech fragments, but also to maintain optimal performance when tested with a more complex signal. This work presents a characterization of the ConvTasNet, Sudo rm -rf, and Sepformer models for online speech separation, and the Demucs-Denoiser, DCCRNet, and MLT mimic loss models for online speech enhancement. Each of these models has been evaluated in terms of signal-to-interference ratio in separation, response time as a function of input sequence length, and real-time factor.

Regarding the speech separation task, all evaluated models show an ability to reduce noise and mitigate the presence of other sources. However, due to the large number of parameters in the Sepformer model, it has a real-time factor greater than 1 for all tested input sequence lengths, indicating that it is not suitable for online processing. The model that provided the best results for online work was Sudo rm -rf, followed by the ConvTasNet architecture. As for the evaluated speech enhancement models, it was found that all of them are suitable for online processing tasks. Demucs-Denoiser has shown the best results in terms of signal-to-interference ratio evaluation and has also been the most robust to variations in input sequence length.

Índice general

Índice de figuras

Índice de cuadros

1 Introducción

La mejora y la separación del habla, es un pilar fundamental para procesos subsecuentes en el análisis de escenas auditivas, tal es el caso del reconocimiento automático de la voz. En la actualidad, con el avance de la tecnología y la incorporación del *Deep Learning*, múltiples dispositivos de asistencia por voz cuentan con mecanismos que les permiten llevar a cabo dichas tareas, tal es el caso de *Google Home* [1].

No obstante, a pesar de los avances en el desempeño de la separación de fuentes de sonido, la complejidad computacional de métodos donde se emplea el uso del *Deep Learning* suele dificultar su uso en sistemas integrados que se emplean en robots de servicio, dispositivos de ayuda auditiva, asistentes por voz, etc. donde los recursos computacionales son limitados [2][3]. A pesar del buen funcionamiento de modelos desarrollados para la separación y la mejora del habla, éstos suponen un procesamiento fuera de línea, es decir, la arquitectura toma el audio completo para realizar la inferencia. Esta limitación plantea la necesidad de explorar el funcionamiento de estas arquitecturas en línea con la finalidad de crear estrategias que permitan adaptarlas para trabajar de manera más eficiente con recursos computacionales reducidos.

En este trabajo se propone la implementación de un sistema que nos permita la caracterización de diversos modelos para la separación y la mejora del habla. Estos modelos serán entrenados con el propósito de encontrar los hiperparámetros que generen el mejor desempeño. Mediante el uso de diversas métricas de evaluación se comprobará si los modelos son viables para trabajar en línea o no, permitiéndonos de esta manera marcar un avance significativo en el área de la comunicación en línea.

1.1. Planteamiento del problema

El problema a resolver en este trabajo se centra en la necesidad de realizar una evaluación de modelos para separación y mejora del habla, con la finalidad de determinar cuales son aptos para un funcionamiento en línea con recursos computacionales limitados, es decir, sistemas con restricciones de potencia de procesamiento. La importancia de un funcionamiento óptimo de este tipo de modelos es esencial para la mejora de la comunicación en tiempo real (≤ 1 segundo).

1.2. Objetivos

En esta sección, se describen de manera concisa y precisa los objetivos de esta tesis.

1.2.1. Objetivo general

Evaluar arquitecturas de aprendizaje profundo con el propósito de determinar su viabilidad para el filtrado en línea de fuentes de sonido, específicamente en la separación de voces. El enfoque se centra en analizar si estas arquitecturas son capaces de realizar la separación de manera continua y en tiempo real, lo cual es fundamental para lograr la separación de las fuentes de sonido en aplicaciones que requieran un bajo costo computacional.

1.2.2. Objetivos particulares

- Realizar un estudio exhaustivo del estado del arte en la separación de fuentes de sonido, con un enfoque particular en la separación de la voz.
- Determinar que arquitectura de aprendizaje profundo es la más idónea para la separación de voz en línea.
- Realizar la conexión entre los modelos elegidos y un sistema para la simulación de procesamiento en línea.

- Caracterizar con respecto al tiempo de respuesta a las arquitecturas seleccionadas.

1.3. Contribución

- De acuerdo con la bibliografía consultada, en la actualidad muchos de los algoritmos empleados para la separación de fuentes de sonido no son aptos para su incorporación en sistemas integrados. La caracterización de diversas arquitecturas para separación de voz permitirá determinar cuáles son aptas para funcionar con bajos recursos computacionales y también si son aptas para funcionar en aplicaciones que requieran un procesamiento en línea.

- En la actualidad existen diversas tareas dónde la separación de voz es requerida, tal es el caso del reconocimiento de voz, identificación del hablante y múltiples procesos de clasificación. Por dicho motivo, tener un sistema de separación de voz es de suma importancia para procesos subsecuentes en el procesamiento del habla.

1.4. Hipótesis

Se plantea qué arquitecturas de separación y mejora el habla pueden ser adaptadas y optimizadas de manera efectiva con la finalidad de poder ser empleadas en aplicaciones de comunicación en línea. Se cree que el desempeño de arquitecturas que conforman el estado del arte (un SIR alrededor de los 15-22 dB) en separación y mejora del habla será suficientemente bueno al trabajar en línea, lo que permitirá su uso en dispositivos con recurso computacionales limitados.

1.5. Estructura de la tesis

El trabajo presente se encuentra estructurado de la siguiente manera, en el capítulo 2 se presenta el marco teórico dónde se detallan los conceptos importantes para comprender el trabajo realizado. En el capítulo 3 se describe el diseño del sistema de separación, se reportan los parámetros utilizados tanto para las arquitecturas de separación como

para las arquitecturas de mejora del habla. También se presenta el procedimiento para la implementación del sistema de separación en línea. Posteriormente, en el capítulo 4 se presentan los resultados de las evaluaciones efectuadas y se realiza una discusión con la finalidad de resaltar los aspectos más importantes de cada modelo. Para finalizar, en el capitulo 5 se hace una recapitulación de todo el trabajo mediante las conclusiones con base a los objetivos planteados. También se explora las posibles alternativas que se podrían seguir a futuro con la finalidad de continuar la investigación.

2 Marco teórico

En esta sección se presentan algunos de los antecedentes más importantes y los conceptos fundamentales que permiten comprender este trabajo. Se comienza brindando una descripción detallada del efecto "*Cocktail party*".

2.1. El efecto *Cocktail party*

En la cotidianidad de nuestra vida nos encontramos inmersos en un entorno donde somos capaces de detectar múltiples sonidos, desde el sonido de un carro pasando por la calle, hasta los vecinos ruidosos del vecindario donde vivimos, estas interferencias acústicas en algunas ocasiones nos imposibilitan escuchar lo que realmente queremos escuchar, sin embargo, poseemos la habilidad de seleccionar la fuente de sonido que queremos oír. La tarea de extraer un sonido de interés en un entorno con ruido se ha denominado como "*Cocktail party*", y ha sido de gran importancia para la comunidad científica [4]. El término de "*Cocktail party*" fue acuñado en 1953 por Cherry en su artículo *"Some Experiments on the Recognition of Speech with One and with Two Ears"* donde describe una serie de pruebas, una de ellas consiste en colocarle a una persona un mensaje hablado en el oído derecho y otro mensaje diferente en el oído izquierdo mediante auriculares, de esta forma se le pide al sujeto que trate de reconstruir el mensaje que escucha en uno de los oídos, sorprendentemente se reporta que el sujeto logra separar el mensaje casi por completo sin alguna dificultad [5]. Posteriormente, en 1994 Bregman se centró en el estudio de la separación de voz en entornos con ruido, a dicho proceso lo denominó análisis de escenas auditivas (ASA, por sus siglas en inglés) [6].

El problema de la "*Cocktail party*" hace referencia a cuando asistimos a una fiesta (ver Figura 2.1) y nos encontramos rodeados de sonidos provenientes de los otros invitados, la música para ambientar la fiesta y el ruido exterior. A pesar de que el entorno se describe como una escena auditiva compleja, somos capaces de poder mantener una conversación

con otra persona debido a nuestra capacidad de poder seleccionar un mensaje hablado sobre otro. Dicha habilidad no solo la poseen los humanos, ya que se ha reportado que muchas otras especies que se encuentran en entornos con mucho ruido deben recurrir a su sentido auditivo para poder escapar de depredadores o encontrar a integrantes de su misma especie [7].

La separación de fuentes de sonido se ha vuelto fundamental en múltiples campos de investigación, tales como dispositivos de ayuda auditiva, audición robótica, asistentes virtuales, etc., por dicho motivo el campo de estudio es bastante activo hoy en día.

Figura 2.1: Escena de la caricatura Norman normal de Warner Bros [8] donde se muestra una fiesta de cóctel.

2.2. El sonido y la voz humana

El sonido se define como un fenómeno físico que se caracteriza por la propagación de ondas acusticas producidas por la vibración de un objeto. Estas vibraciones hacen que las partículas del medio en el que se encuentra el objeto, si el medio lo permite, también comiencen a vibrar, transportando así energía a otros lugares. Las ondas de sonido pueden propagarse por distintos medios, como lo son: sólidos, líquidos y gases [9].

La evolución del ser humano ha traído consigo un mecanismo vocal mediante el cual somos capaces de generar y transmitir mensajes para poder comunicarnos los unos con los otros, dicho mecanismo se caracteriza por la producción de sonidos particulares a los que se les ha asignado un significado. En la producción de la voz humana (ver Figura 2.2) están involucrados varios componentes de nuestro cuerpo, comenzando por una corriente de aire impulsada por los pulmones pasando por la tráquea hasta la laringe donde dicha corriente hace vibrar las cuerdas vocales, estas vibraciones generan ondas sonoras que pasan por la faringe, la boca y la nariz [10].

La frecuencia con la que vibran las cuerdas vocales durante la producción del sonido es conocida como la frecuencia fundamental del habla. Es importante destacar que esta frecuencia no es única, sino que varía significativamente entre mujeres y hombres, niños y adultos. De acuerdo con [11] la frecuencia fundamental promedio en hombres oscila entre 100 y 146 Hz, mientras que en mujeres esta en un rango de 188 a 221 Hz. En el trabajo de [12] se reporta que no existe una diferencia significativa en la frecuencia fundamental de niños y niñas en un rango de edad de 6 y 10 años, reportando una frecuencia fundamental promedio de 262 Hz para niños y de 281 Hz para niñas.

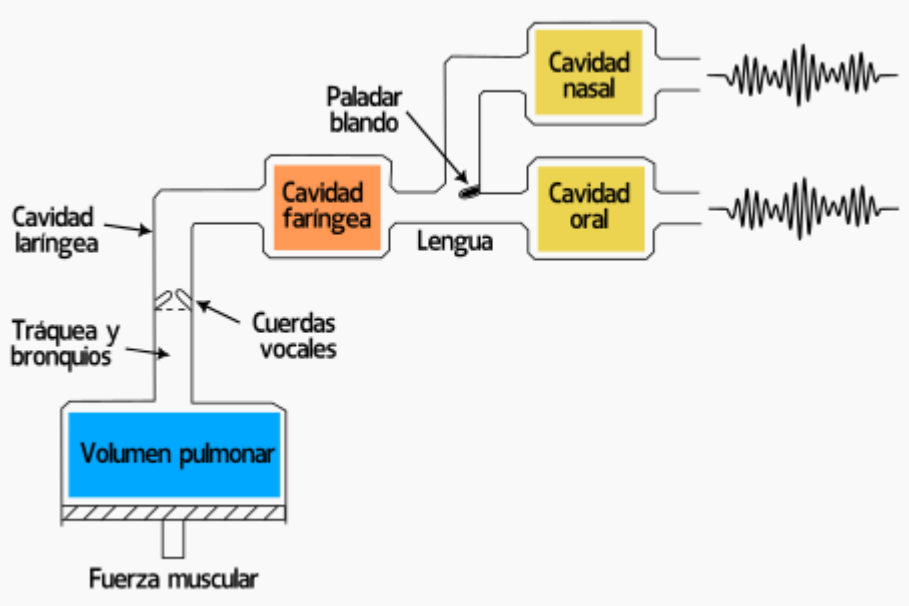


Figura 2.2: Diagrama del sistema de producción de voz. Adaptado de [10].

2.3. Procesamiento digital de señales de audio

Una señal representa un fenómeno físico que varia en función de una variable independiente, como lo puede ser el tiempo o el espacio. Dichas variaciones pueden ser de amplitud, temperatura, presión o alguna otra propiedad relevante [13]. El procesamiento digital de señales se caracteriza por muestrear una señal analógica (continua) en intervalos finitos para posteriormente realizar un procesamiento y después regresar a la forma continua (ver Figura 2.3).

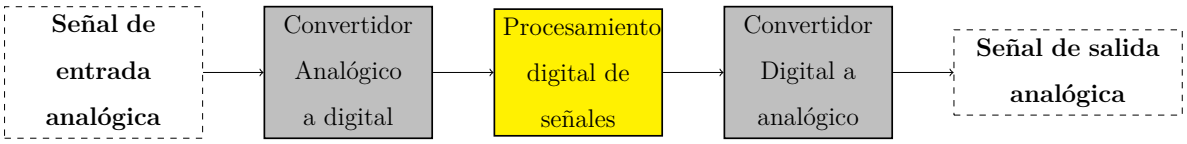


Figura 2.3: Procesamiento digital de señales.

El procesamiento digital de señales (DSP por sus siglas en inglés) es utilizado en diversos campos de la ciencia e ingeniería con la principal finalidad de obtener o alterar información proveniente de una señal. Algunas de las aplicaciones del DSP son el radar y el sonar donde se emplea para la detección de objetos en navegación marítima y área; en el procesamiento de señales biomédicas como el electrocardiograma (ECG), electroencefalograma (EEG) y la imagenología por resonancia magnética (IRM); también es utilizado en la sismología permitiendo detectar y localizar temblores [14].

Además, el procesamiento digital de señales ha tenido una gran relevancia en el procesamiento de señales de audio, en particular en el procesamiento del habla. Algunas de las posibles aplicaciones son: la síntesis de voz, la cual consiste en generar voces artificiales a partir de texto escrito [15]; el reconocimiento de voz, que permite generar textos a partir de señales habladas, teniendo posibles aplicaciones en asistentes inteligentes, transcripción automática, etc [16]; la separación del habla, que permite aislar fuentes individuales a partir de una mezcla con múltiples hablantes y ruido, teniendo aplicaciones en dispositivos de ayuda auditiva, telecomunicaciones u otros procesos del procesamiento del habla como lo es el reconocimiento de voz [17].

2.3.1. Teorema del muestreo

Para procesar una señal continua $x(t)$ en una computadora se recurre al proceso de muestreo o discretización de la señal. Esto implica representar a la señal tomando en cuenta determinadas instancias de tiempo t_n con $n = 0, 1, 2, \ldots, \infty$ de tal forma que:

$$x_n = x(t_n) \tag{2.1}$$

en la practica se suelen tomar puntos equidistantes para el muestreo, es decir, el tiempo de separación en segundos T_s entre cada intervalo de la señal es el mismo, teniendo:

$$t_n = T_s n, \quad \text{para} \quad n = 0, 1, 2, \ldots, N \tag{2.2}$$

donde N es un número finito de muestras seleccionadas para representar la señal de manera efectiva. A medida que el valor de T_s disminuye, la señal se muestrea más detalladamente; sin embargo, esto conlleva un mayor requerimiento de memoria computacional para su representación (ver Figura 2.4).

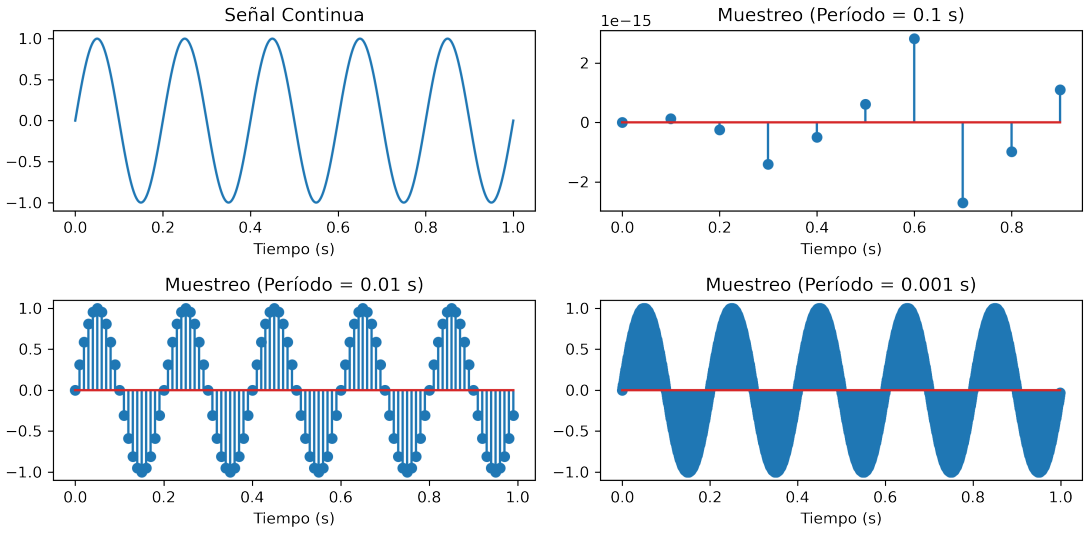


Figura 2.4: Muestro de una señal continua a periodos de tiempo de: 0.1s, 0.01s y 0.001s.

Podemos obtener el número de muestras por segundo mediante la frecuencia de muestreo, que está dada por la ecuación 2.3.

$$f_s = \frac{1}{T_s} \tag{2.3}$$

Hemos visto que si la frecuencia de muestreo es lo suficientemente grande, no habrá pérdida de información al muestrear una señal, lo que nos lleva a plantearnos la siguiente pregunta: ¿cuál es el tamaño adecuado de la frecuencia de muestreo para capturar correctamente una señal? El teorema del muestreo o teorema de *Nyquist-Shannon* nos brinda una respuesta a dicha interrogante estableciendo que:

Teorema del muestreo

Para poder reconstruir completamente una señal continua a partir de sus muestras, la frecuencia de muestreo debe ser al menos el doble de la frecuencia máxima, f_{max}, presente en la señal original.

$$f_s \geq 2f_{max} \tag{2.4}$$

2.3.2. Transformada de Fourier discreta

En diversos trabajos sobre separación del habla se emplean métodos que son efectuados en el dominio de la frecuencia debido a que se ha mostrado que la distribución de la energía del habla humana tiene una distribución dispersa, es decir, se muestra que la energía suele ser mayor en partes aisladas del espectro, lo que permite una mejor discriminación de las fuentes voz [18].

Una señal discreta $x[n]$ con duración N, donde n corresponde a una muestra o instante de tiempo discreto, puede ser representada en el dominio de la frecuencia como una combinación lineal de exponenciales complejas. Dicha representación se obtiene mediante la transformada discreta de Fourier (DFT) que está dada por la ecuación 2.5.

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-jk\frac{2\pi}{N}n}, \quad k = 0, 1, 2, \ldots N-1 \tag{2.5}$$

La DFT toma a la señal y la descompone en sus N componentes frecuenciales fundamentales $X[k]$, donde k es el índice que representa a la k-ésima frecuencia (harmónico).

La transformada inversa discreta de Fourier se obtiene mediante la ecuación 2.6

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k] e^{jk\frac{2\pi}{N}n}, \quad n = 0, 1, 2, \ldots N-1 \tag{2.6}$$

Propiedad	**Señal**	**DFT**
Linealidad	$ax[n] + by[n]$	$aX[e^{j\omega}] + bY[e^{j\omega}]$
Desplazamiento (tiempo)	$x[n - n_0]$	$e^{-j\omega n_0} X[e^{j\omega}]$
Desplazamiento (frecuencia)	$x[n] \cdot e^{j\frac{2\pi}{N}\omega_0 n}$	$X[e^{j(\omega-\omega_0)}]$
Convolución	$x[n] \cdot y[n]$	$X[e^{j\omega}]Y[e^{j\omega}]$
Periodicidad	$x[n+N] = x[n]$	$X[k+N] = X[k]$

Cuadro 2.1: Propiedades de la DTF.

2.3.3. Transformada Rápida de Fourier

La Transformada Rápida de Fourier (FFT, por sus siglas en inglés) es un algoritmo revolucionario utilizado para calcular eficientemente los coeficientes de la DFT. Su descubrimiento por Cooley y Tukey en 1965 significó un gran avance en el campo del procesamiento de señales, ya que logró reducir drásticamente la complejidad computacional de la DFT sin degradar su precisión mediante la eliminación de una gran cantidad de cálculos redundantes, cabe mencionar que dicho algoritmo también fue reportado en libretas de Gauss al utilizarlo en la interpolación de órbitas de asteroides en 1805 [19].

La DFT, como se muestra en la ecuación 2.5, normalmente requería N multiplicaciones complejas y $(N-1)$ sumas complejas, lo que llevaba a una complejidad computacional de $\mathcal{O}(N^2)$. Sin embargo, con la FFT, se puede calcular la DFT con una complejidad de $\mathcal{O}(N \log_2 N)$ [9].

Una de las implementaciones más utilizadas en la actualidad para la FFT es la conocida como *Fastest Fourier Transform in the West* (FFTW) [20].

2.3.4. Espectrograma

Las señales de audio pueden ser representadas de diversas maneras. Una de las representaciones más comunes es mediante la amplitud de la señal en función del tiempo, la cual es conocida como forma de onda. Otra forma de representación es en el dominio de la frecuencia, donde la señal se descompone en sus componentes espectrales mediante la transformada de Fourier.

Una alternativa que muestra como el espectro cambia en el tiempo es el espectrograma. En esta representación visual, las variaciones de frecuencia se ubican en el eje vertical, mientras que las variaciones temporales del espectro se sitúan en el eje horizontal. Utilizando cambios de tono o colores en la imagen para representar la amplitud, el espectrograma ofrece una visión completa de cómo la energía asociada a distintas frecuencias varía en el tiempo.

Un espectrograma se puede obtener mediante la transformada de Fourier de tiempo corto (STFT por sus siglas en inglés) que divide la señal en segmentos o ventanas más pequeñas y calcula la transformada de Fourier de cada una de estas ventanas. La STFT está definida mediante la ecuación 2.7 [21].

$$X[m,k] = \sum_{n=0}^{N-1} x[n+mH]w[n]e^{-2\pi ikn/N} \tag{2.7}$$

Podemos observar que la STFT es una función de dos variables, m es el índice del tiempo y k para la frecuencia. También se usa la función ventana, w, que multiplica a la señal. El parámetro N determina el tamaño de los segmentos considerados, H es el tamaño de salto e indica el tamaño de paso en el que se desplazará la ventana [21].

En la literatura, se reportan distintos tipos de ventanas que se utilizan en aplicaciones específicas de acuerdo a la forma de la ventana. Por ejemplo, la ventana de Hann es especialmente útil para reducir los efectos de borde causados por discontinuidades en la señal, en la Figura 2.5 se presentan algunas de las ventanas más comunes.

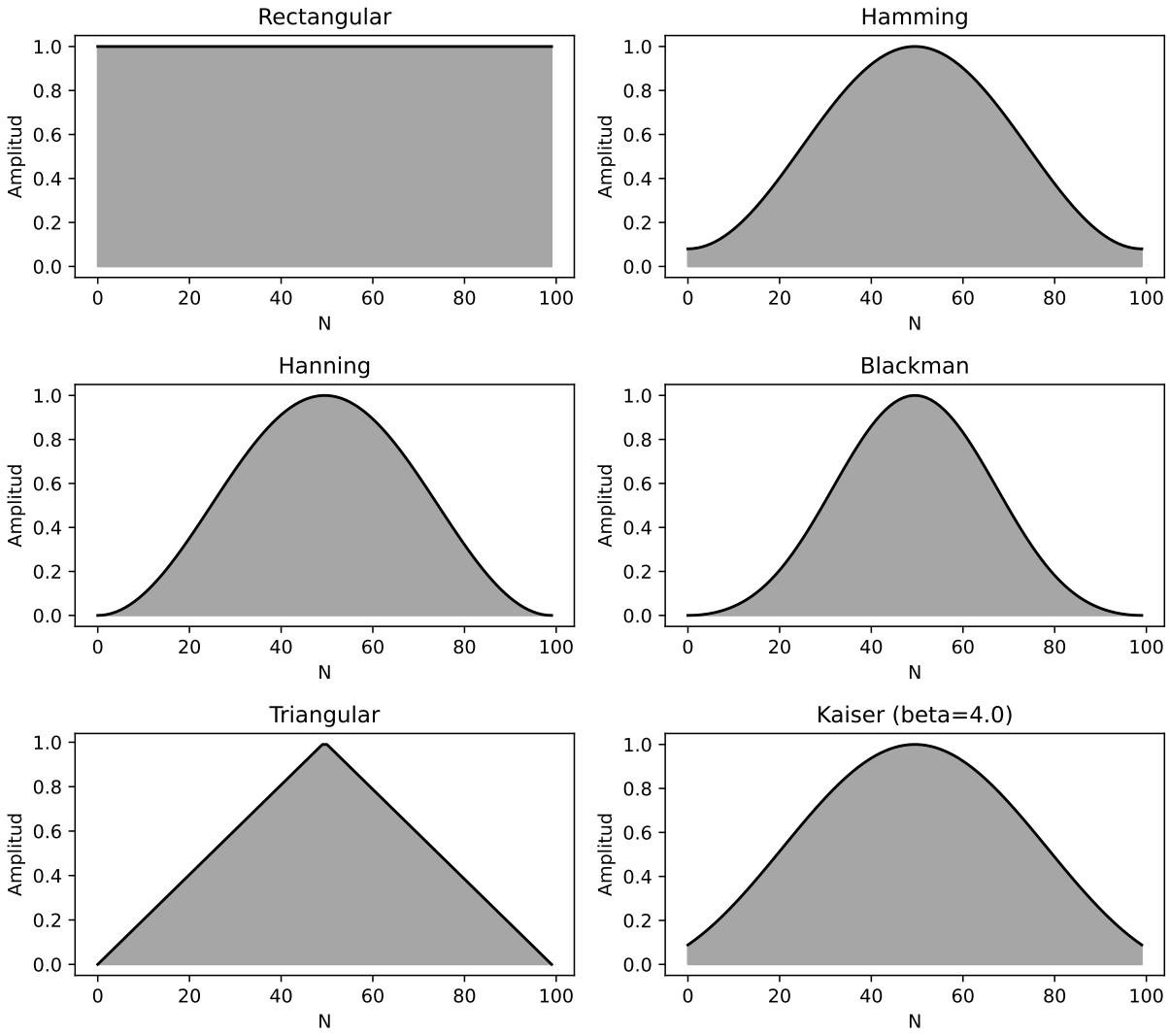
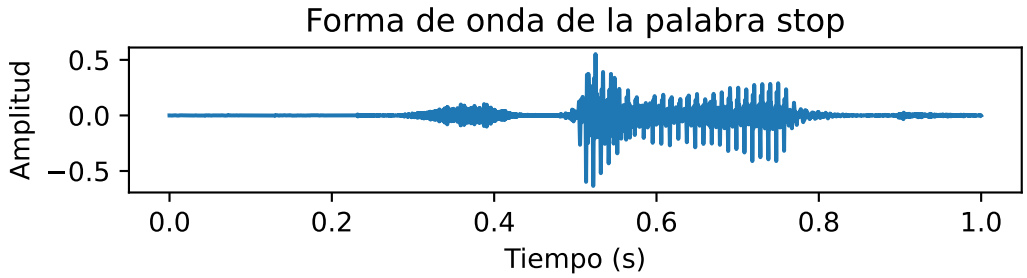
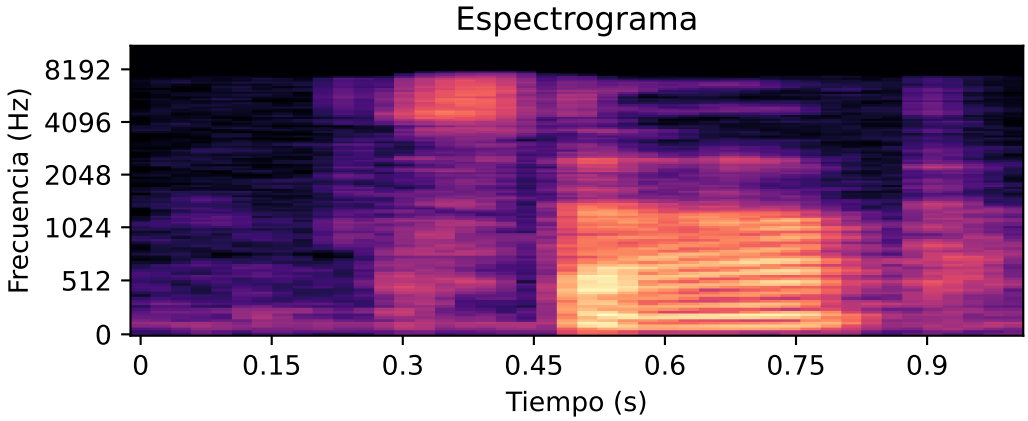


Figura 2.5: Tipos de funciones ventana.

A continuación se muestra un ejemplo de un espectrograma (ver Figura 2.6) el cual es generado a partir del audio de la palabra *stop* que forma parte de la base de datos *speech commands* [22].



(a)



(b)

Figura 2.6: Representación tiempo-frecuencia del audio de la palabra stop. (a) Forma de onda. (b) Espectrograma.

2.3.5. Procesamiento en línea

Al hablar de sistemas integrados donde se efectúa una interacción directa con el usuario nos debemos enfocar en que el sistema pueda funcionar en línea, es decir, el tiempo de respuesta del sistema debe ser menor a la ventana de tiempo que se procesa. En el procesamiento fuera de línea el tiempo de respuesta es más largo, ya que primero se deben grabar los audios y posteriormente se efectúa su procesamiento para poder brindar una respuesta [2]. El procesamiento *online* (o, en línea) trae consigo algunos retos computacionales como el de tener que aplicar una superposición y una suma de las ventanas de tiempo que se procesan para evitar la pérdida de información al principio y al final de estas [23].

2.4. Separación de fuentes de voz

El problema de la separación de fuentes de voz se refiere a la tarea de descomponer una señal mezclada en sus componentes individuales, es decir, separar las diferentes fuentes de voz que contribuyen a la mezcla (ver Figura 2.7). Para abordar este problema, se utilizan técnicas de procesamiento de señales y algoritmos de separación de fuentes.

Supongamos que se tienen C fuentes a separar dado un tiempo t, de tal manera que: $s_1(t), s_2(t), \ldots, s_C(t) \in \mathbb{R}^{1\times T}$ y sea $x(t) \in \mathbb{R}^{1\times T}$ la señal mezclada, de tal forma que:

$$x(t) = \sum_{i=0}^{C} s(t)_i$$

el objetivo es estimar $s_1(t), s_2(t), \ldots, s_C$ a partir de $x(t)$.

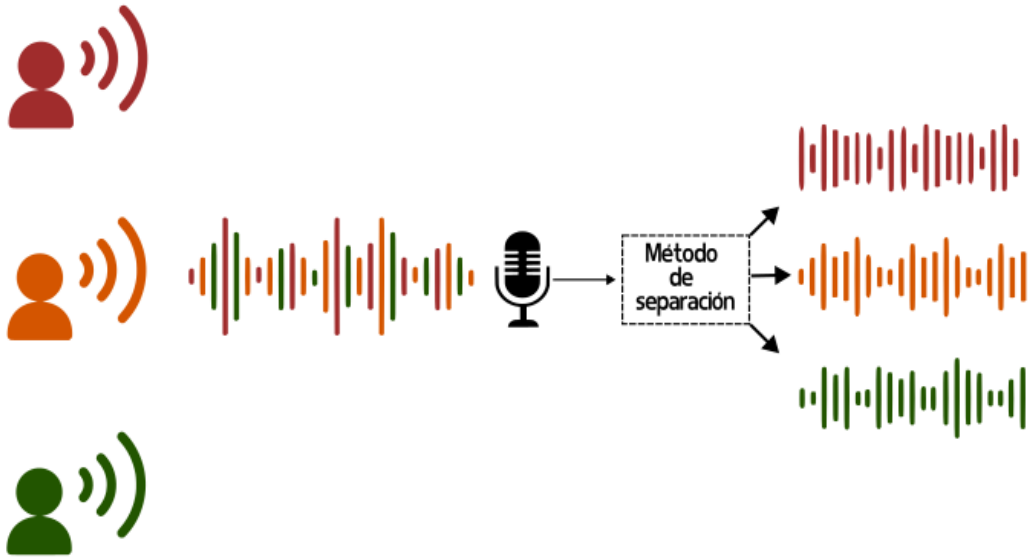


Figura 2.7: Separación de voz.

2.4.1. Métodos algorítmicos de separación ciega de componentes

La comunicación ha sido una pieza fundamental para la humanidad, permitiéndonos entablar conversaciones con personas ubicadas a grandes distancias, gracias al teléfono. No

obstante, es importante destacar que existen factores que pueden dificultar la comprensión del habla al utilizar este dispositivo, entre ellos el ruido ambiental, la reverberación o alguna otra interferencia. Estos obstáculos pueden afectar la calidad de las comunicaciones y crear desafíos para una interacción fluida y clara. Con el avance en el campo de procesamiento de señales, se han desarrollado técnicas que permiten extraer una señal de voz de una mezcla que ha sido corrompida por algún tipo de interferencia. Estas técnicas reciben el nombre de "separación ciega de fuentes" o BBS por sus siglas en inglés (*Blind Source Separation*) [24].

Un método representativo de este tipo de técnicas es el análisis de componentes independientes (ICA por sus siglas en inglés). Este método se basa en la suposición de que las señales a separar en una mezcla de audio son estadísticamente independientes, es decir, dada una señal s_1 y dada una señal s_2, los valores de s_1 no proporcionan ninguna información sobre s_2 y viceversa. De modo que si tenemos N fuentes de audio estadísticamente independientes, $s_i(t)$, con $i = 1, 2, 3 \ldots, N$ las cuales son grabadas por N micrófonos simultáneamente, $x_i(t)$ con $i = 1, 2, 3 \ldots, N$ el modelo de la captura de audio puede ser representado como:

$$x(t) = As(t) \tag{2.8}$$

donde A es una matriz desconocida la cual es llamada la matriz de mezcla, $s(t)$ es el vector de fuentes independientes y $x(t)$ es un vector que contiene a las señales mezcladas grabadas por los micrófonos. De tal manera que la separación por medio de ICA se centra en encontrar una matriz que estime la separación de las fuentes, la cual denominan como matriz des-mezcladora, W, que es la matriz inversa de A [25].

2.4.2. Máscaras de frecuencia

La separación por máscaras de frecuencia (ver Figura 2.8) es uno de los enfoques que se emplea para separar señales de voz, efectuando la separación en el dominio de tiempo-frecuencia (T-F). De acuerdo con [18] el proceso de separación de una señal con ruido esta dado por los siguientes pasos:

- La señal se procesa mediante algún método de análisis T-F, por ejemplo, la STFT para generar una representación T-F.
- Se hace uso de un algoritmo de separación para obtener la máscara T-F.
- Se realiza un paso de "síntesis" para recuperar la forma de onda de la señal separada.

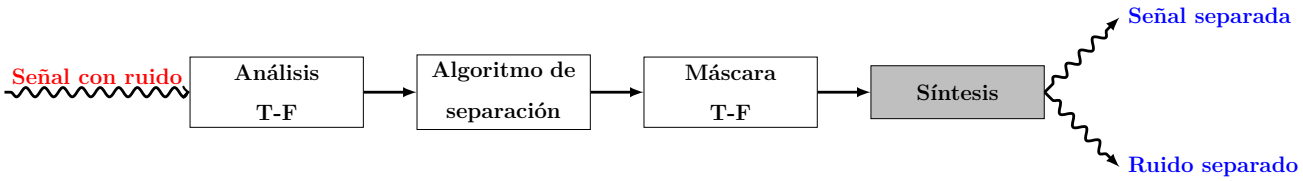


Figura 2.8: Diagrama para representar la separación mediante máscaras de frecuencia. Imagen adaptada de [18]

.

A pesar de que el filtrado por máscara de frecuencias resulta prometedor en múltiples casos de separación de fuentes, el estimar dicha máscara resulta una tarea muy complicada y más cuando la escena auditiva es muy compleja, con mucho ruido y reverberación [26]. En la literatura se reportan diversos algoritmos para la estimación de máscaras de frecuencia [27]. Uno de los enfoques es el filtrado de Wiener que es comúnmente utilizado para crear máscaras de T-F que posteriormente se aplican al espectrograma de la señal a separar [28, 29].

2.4.3. Separación por *beamforming*

Las técnicas de separación de fuentes que involucran el uso de múltiples micrófonos han desempeñado un papel fundamental en el desarrollo de métodos avanzados de separación. Entre estas técnicas destaca el *beamforming* o filtrado espacial, que se basa en la premisa de conocer previamente la dirección de las fuentes que se desean separar. Para lograrlo, se emplea un arreglo de micrófonos que capturan las señales provenientes de diferentes direcciones permitiendo capturar la información espacial de las fuentes [2, 30].

Uno de los filtros espaciales más sencillo es el *delay-and-sum* (DAS por sus siglas en inglés). En este enfoque, las señales captadas por cada micrófono se retrasan de manera

específica para alinear temporalmente las señales provenientes de distintas direcciones. Posteriormente, se promedian ponderadamente, enfocando así la sensibilidad del sistema hacia una dirección determinada y atenuando las señales provenientes de otras direcciones no deseadas.

2.4.4. Separación por mejora del habla (*Speech enhancement*)

La separación por medio de mejora del habla se centra en realzar la señal de interés, en este caso, la voz y suprimir el ruido y las interferencias. Esta tarea es complicada debido a que debe existir un balance entre disminuir el ruido y conservar la inteligibilidad del hablante [31]. En la literatura se reportan diversos métodos para la reducción del ruido en una señal, uno de los primeros fue el de substracción espectral, propuesto para mejora del habla de un sólo canal. La sustracción espectral tiene como finalidad estimar ruido mediante las pausas presentes en la señal ruidosa completa, para posteriormente restarlo a la señal y así mejorarla.

2.5. Aprendizaje profundo

2.5.1. Redes neuronales biológicas

La información que recibimos tanto del interior como del exterior de nuestro cuerpo es procesada mediante el sistema nervioso, este se encuentra compuesto por las células neuronales y las células gliales. Las células neuronales se encargan de transmitir información alrededor de todo el cuerpo mediante señales eléctricas y las células gliales tienen la función de brindarles sostén a las neuronas.

La estructura neuronal (ver Figura 2.9) esta constituida por ramificaciones que les permiten interactuar con otras neuronas, las dendritas se extienden desde el cuerpo de la célula neuronal y son las encargadas de recibir información de otras células neuronales [32]. El axón es el componente de la neurona encargado de la conducción del potencial de acción hasta el sitio donde se lleva interacción sináptica con otra neurona, esta interacción se da entre el axón terminal y la dendrita de la otra neurona. De acuerdo con [33] el axón puede

llegar a medir hasta un metro de longitud o más y puede ramificarse en hasta 500,000 terminales axónicas. El axón se encuentra cubierto por vainas de mielina que facilitan la conducción eléctrica. El ser humano cuenta con 100,000 neuronas aproximadamente y se ha reportado que las células gliales o de sostén igualan dicha cantidad.

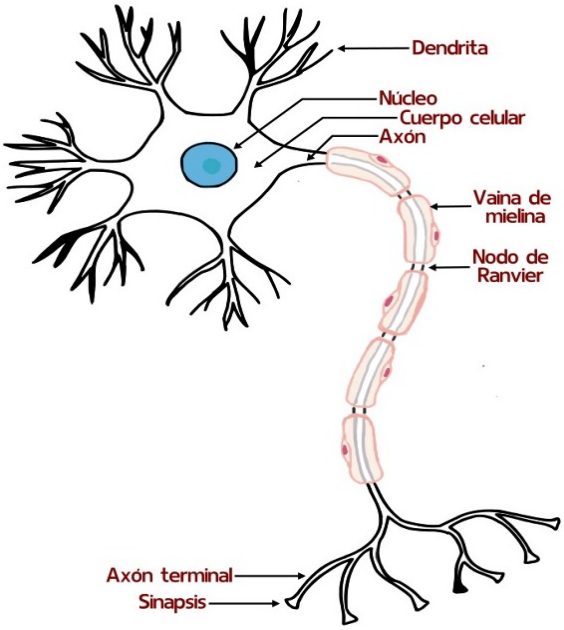


Figura 2.9: Neurona biológica. Adaptado de [34].

2.5.2. Redes neuronales

La primera red neuronal artificial fue introducida en 1943 por Warren McCulloch y Walter Pitts en su artículo: *A logical calculus of the ideas immanent in nervous activity.* En dicho escrito, los autores proponen un modelo teórico para simular el funcionamiento de la actividad neurológica del cerebro mediante lógica proposicional [35].

En 1958 Frank Rosenblatt presentó un modelo basado en el funcionamiento de las redes neuronales biológicas al cual llamó "perceptrón". El perceptrón (ver Figura 2.10) es una red neuronal artificial para clasificación binaria y consta de una capa de entradas $x_1, x_2, \ldots, x_n$ y le asigna a cada entrada un peso sináptico correspondiente $w_1, w_2, \ldots, w_n$ que expresa la importancia de la entrada. Posteriormente, se realiza una suma ponderada de las entradas multiplicadas por sus respectivos pesos más un sesgo, esto se representa

mediante la ecuación 2.9.

$$z = \sum_{i=1}^{n} (x_i \cdot w_i) + b \tag{2.9}$$

Posteriormente, el valor de la salida, 0 o 1, se decide mediante un valor umbral u:

$$f(z) = \begin{cases} 0 & \text{si } z \leq u \\ 1 & \text{si } z > u \end{cases} \tag{2.10}$$

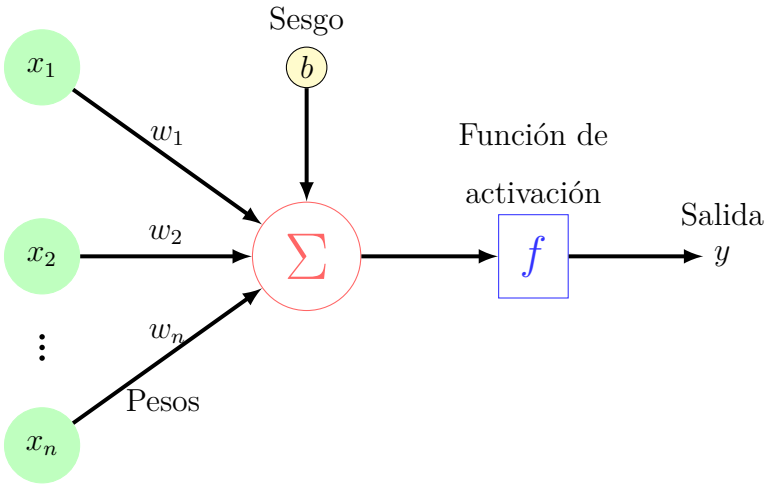


Figura 2.10: Red neuronal artificial (perceptrón).

Perceptrón multicapa

El perceptrón es un algoritmo de tiene la desventaja de que únicamente puede ser utilizado para resolver problemas de clasificación lineal. Sin embargo, este problema logró ser abordado introduciendo arquitecturas más complejas como el perceptrón multicapa o también llamado red neuronal *feedforward* debido a que el flujo de la información va hacia adelante sin alguna retroalimentación. Este tipo de red tiene una capa de entrada, una o varias capas ocultas y una capa de salida (ver Figura 2.11). En este caso a la salida de cada capa oculta se le aplica un mapeo conocido como función de activación lo cual permite introducir no linealidad al modelo. Las funciones de activación ocupadas en cada

capa pueden ser diferentes, usándose comúnmente: la función sigmoide, ReLU, lineal, tanh, Sofmax, entre otras (ver Cuadro 2.2).

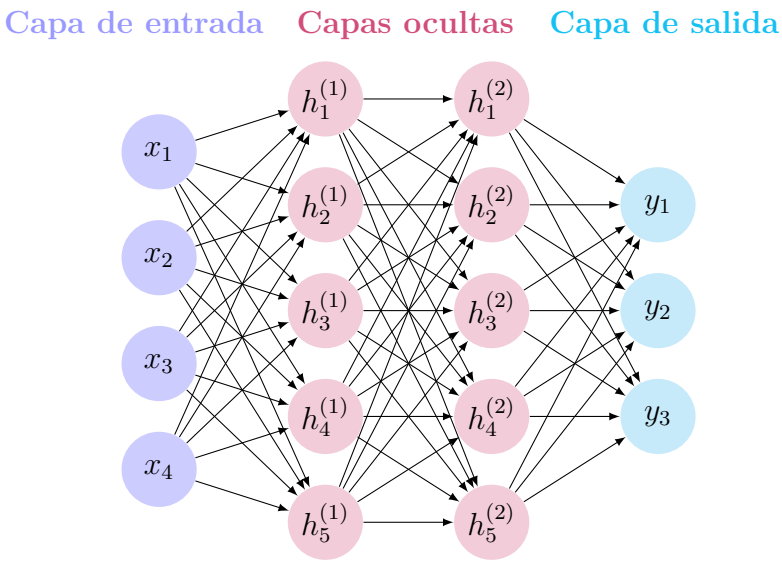


Figura 2.11: Red neuronal *feedforward* con dos capas ocultas.

Función de activación	**Definición**
Hard Limit	$f(z) = \begin{cases} 0, & \text{si } z < 0 \\ 1, & \text{si } z \geq 0 \end{cases}$
Lineal	$f(z) = z$
ReLU (*Rectified Linear Unit*)	$f(z) = \text{máx}(0, z)$
Sigmoid	$f(z) = \frac{1}{1+e^{-z}}$
Tangente hiperbólica (tanh)	$f(z) = \tanh(z)$
Softmax	$f_i(z) = \frac{e^{z_i}}{\sum_j^k e^{z_j}}$

Cuadro 2.2: Definiciones de diferentes funciones de activación en redes neuronales.

Retropropagación

Como ya se ha mencionado, con la incorporación de los redes neuronales multicapa se logran realizar diversas tareas de clasificación que no se podían hacer con el perceptrón de

una capa pero eso también conlleva tener que encontrar la combinación correcta de todos los pesos, ajustar todos los pesos a mano sería una tarea bastante larga. Para ello se ocupa una combinación de los algoritmos de retropropagación y de descenso del gradiente. De acuerdo con [36] la primera descripción la técnica de retropropagación está en un trabajo publicado por Paul Werbos en 1974, sin embargo, no fue hasta 1986 cuando dicho algoritmo fue redescubierto y publicitado por David Rumelhart, Geoffrey Hinton y Ronald Williams. Cuando se usa una red neuronal feedforward se tiene un conjunto de ejemplos de la forma:

$$\{x_1, y_1\}, \{x_2, y_2\}, \ldots, \{x_n, y_n\}$$

Aquí $\mathbf{x} = x_1, x_2, \ldots, x_n$ representa las entradas de la red mientras que $\mathbf{y} = y_1, y_2, \ldots, y_n$ son las salidas deseadas. Durante el proceso de cálculo dada una entrada $\mathbf{x}$, la información fluye hacia delante propagándose por las capas ocultas con el objetivo de producir una salida $\mathbf{y}$. A ésta etapa se le conoce como propagación hacia delante. El valor resultante de $\mathbf{y}$ es comparado con el valor real mediante una función de pérdida, que cuantifica el error entre la salida predicha y la salida deseada. El algoritmo de retropropagación permite que la información del error fluya hacia atrás para calcular el gradiente con la finalidad de ajustar los parámetros de la red [37]. Este proceso se repite hasta que la función de costo alcance un mínimo local o converja a un valor aceptable.

Descenso del gradiente

El descenso del gradiente es un algoritmo de optimización que se emplea para encontrar los pesos y sesgos que minimizan el error durante el entrenamiento de una red neuronal. Durante dicho proceso el objetivo es minimizar la función de pérdida C. En la práctica, la elección de la función de pérdida dependerá de la aplicación y el tipo de salida que debe generar la red, por ejemplo, el error cuadrático medio (MSE por sus siglas en inglés) se suele usar para problemas de regresión, mientras que la entropía cruzada binaria suele ser más apta para problemas de clasificación.

El descenso del gradiente computa las derivadas parciales de la función de pérdida con respecto a los pesos y sesgos, y utiliza estas derivadas para calcular su gradiente. El gradiente nos brinda información de la dirección en la cual la función de pérdida cambia más

rápido. Al moverse en la dirección opuesta al gradiente, la función de pérdida disminuye con cada ajuste de los pesos y sesgos, de tal forma que nos acerca progresivamente al mínimo local de dicha función.

Para obtener la regla de actualización de los pesos y sesgos dada la siguiente función de pérdida:

$$C(w,b) = \frac{1}{2}\sum_{i=0}^{n}(y_i - \hat{y}_i)^2 \tag{2.11}$$

donde y_i es la salida deseada y $\hat{y}_i$ es la salida predicha por la red, tenemos que obtener la dirección de máxima pendiente de dicha función como se muestra en las ecuaciones 2.12 y 2.13.

$$w_i(t+1) \leftarrow w_i(t) - \alpha\left(\frac{\partial C(w,b)}{\partial w_i(t)}\right) \tag{2.12}$$

$$b(t+1) \leftarrow b(t) - \alpha\left(\frac{\partial C(w,b)}{\partial b(t)}\right) \tag{2.13}$$

Donde α representa la tasa de aprendizaje y determina el tamaño de las actualizaciones que se realizan para actualizar los parámetros durante el entrenamiento.

2.5.3. Redes neuronales convolucionales

A pesar de que las redes neuronales feedforward son una herramienta poderosa que permite resolver múltiples problemas aprendiendo características y clasificando, se tiene un inconveniente al trabajar con imágenes. En dicho caso se tiene que procesar cada píxel de la imagen como una entrada a la red, esto ocasiona que el número de parámetros a entrenar para la red incremente demasiado. Las redes neuronales convolucionales (CNN por sus siglas en inglés) fueron propuestas por Yann LeCun en 1998 [38], este tipo de redes neuronales son utilizadas especialmente en procesamiento de datos ordenados en una cuadrícula bidimensional (2D), como lo es una imagen, pero también se pueden aplicar a señales de audio. De acuerdo con [39], las redes neuronales convolucionales: "son, simplemente, redes multicapa en las que algunas capas realizan una operación de convolución en lugar de la tradicional multiplicación matricial de entradas por pesos". Para el caso discreto la convolución se define matemáticamente como:

$$(f \star g)[n] = \sum_{m=-\infty}^{\infty} f[n-m]g[m] \tag{2.14}$$

Para el caso de una señal discreta dependiente de dos variables (una imagen) la convolución con un filtro de tamaño $N \times M$ se obtiene mediante:

$$(x \star k)[i,j] = \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} k[n,m]x[i-n, j-m] \tag{2.15}$$

Las CNN se centran en tres ideas básicas: conectividad local, pesos compartidos y capas de submuestreo (*pooling*) [39, 37, 40]. A continuación se describe cada una de dichas ideas.

- **Conectividad local**: la conectividad local se refiere al hecho de que en una CNN no todas las neuronas están conectadas a una entrada, en este caso una neurona de la capa convolucional está conectada a un conjunto de entradas, al cual se le suele denominar como campo receptivo, de esta forma cada neurona se centra en capturar una característica de dicha región.

- **Pesos compartidos**: en lugar de asignar un solo peso a cada conexión individual, las CNN aplican el mismo conjunto pesos a las distintas partes de la entrada, asegurándose que se detecte una sola característica por neurona, esto ayuda a reducir el número de parámetros que deben ajustarse.

- **Pooling**: las capas de submuestro o pooling se encargan de reducir el tamaño de entrada para que las capas posteriores puedan procesar únicamente la información esencial, esto ayuda a reducir el costo computacional de una CNN pero también proporciona invarianza frente a translaciones en la señal de entrada.

2.5.4. Redes neuronales recurrentes

Una gran desventaja de las redes neuronales *feedforward* es que carecen de memoria. Para aplicaciones donde se requiere el procesamiento de secuencias, como lo son: traducción automática, clasificación de textos, reconocimiento de voz o cualquier aplicación en la que se tenga dependencias secuenciales entre los atributos, se recurre a redes neuronales recurrentes (RNN por sus siglas en inglés). De acuerdo con la definición de [41], “una

RNN es una red neuronal que mapea desde un espacio de secuencias de entrada a un espacio de secuencias de salida en forma de estado". En una RNN, las secuencias de salida no únicamente dependen de las secuencias de entrada, sino que dichas entradas también interactúan con los estados ocultos creados a partir de las entradas en tiempos anteriores, un estado oculto para un tiempo t esta dado por la ecuación 2.16.

$$h_t = f(h_{t-1}, x_t) \tag{2.16}$$

En la ecuación previa, la función $f()$ desempeña el papel de función de activación. En el contexto de las redes neuronales recurrentes (RNN), es común emplear la función de activación tangente hiperbólica (tanh), sin embargo, se puede utilizar alguna otra función de activación. La ecuación 2.16 puede ser escrita como:

$$h_t = tanh(W_{xh}x_t + W_{hh}h_{t-1} + b_h) \tag{2.17}$$

donde W_{xh} denota la matriz de pesos que conecta las entradas con las capas ocultas, W_{hh} es la matriz de pesos entre las capas ocultas y b_h es el vector de sesgo.

Dadas las secuencias ocultas, la secuencia de salida puede ser calculada de la siguiente forma:

$$y_t = W_{hy}h_t + b_y \tag{2.18}$$

donde W_{hy} es la matriz de pesos entre las capas ocultas y las capas de salida. La pérdida en una RNN es la suma de pérdidas por paso de tiempo:

$$\mathcal{L}(\hat{y}, y) = \sum_{t=1}^{T} \mathcal{L}(\hat{y}_t, y_t) \tag{2.19}$$

Para determinar los gradientes en una red neuronal recurrente (RNN), se emplea una variante de la técnica de retropropagación conocida como retropropagación a través del tiempo (BPTT por sus siglas en inglés). En la BPTT se toma en cuenta la suma de los errores en cada paso temporal, teniendo en cuenta que los pesos son los mismos en cada marca de tiempo de la red. Este proceso puede generar problemas a la hora de entrenar una RNN por un fenómeno conocido como desvanecimiento del gradiente. Esto ocurre cuando los gradientes disminuyen exponencialmente a medida que se retropropagan en el tiempo,

lo que lleva a que las actualizaciones de pesos en las capas iniciales sean insignificantes. Como resultado, la red deja de aprender [42].

Este tipo de procesamiento brinda a estas redes la habilidad de poder capturar dependencias temporales en los datos, dándoles una especie de "memoria". En la Figura 2.12 se muestra una representación gráfica de una RNN.

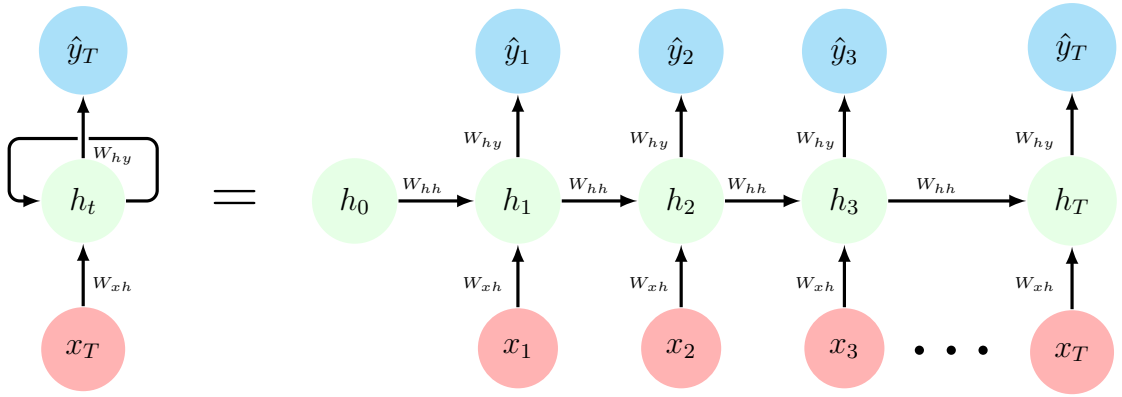
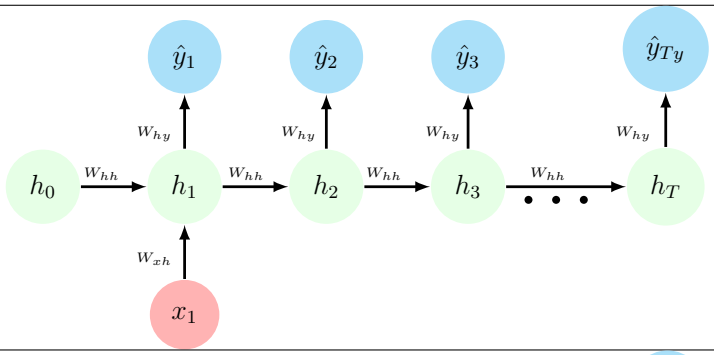
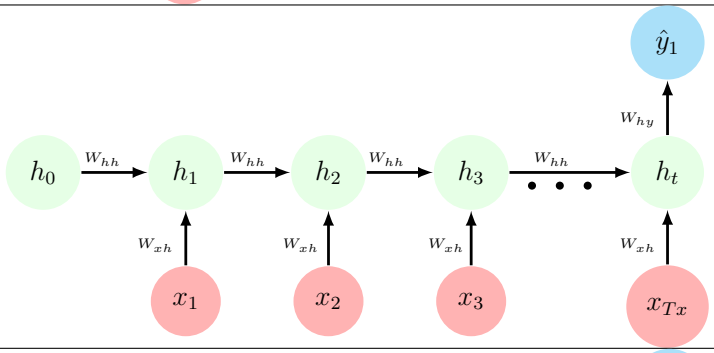
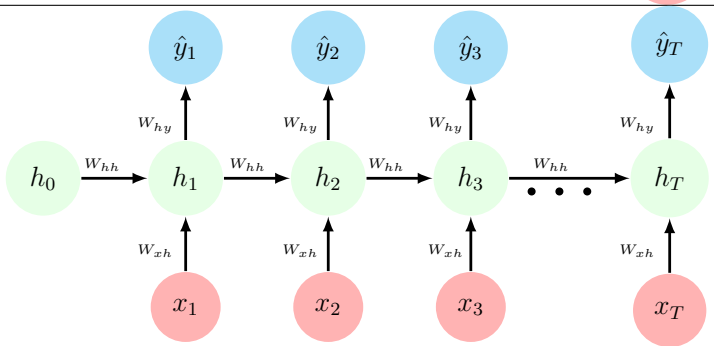
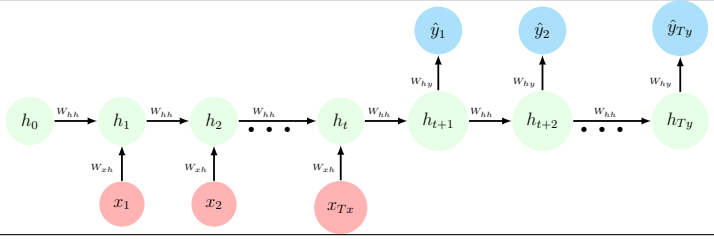


Figura 2.12: Red neuronal recurrente. El diagrama de la izquierda corresponde a una versión simplificada. El diagrama de la derecha corresponde a una versión desplegada.

Se pueden diseñar múltiples estructuras de redes neuronales recurrentes implementando el uso compartido de sus parámetros. En el Cuadro 2.3 se muestran los diferentes tipos de RNN y sus distintas aplicaciones.

Tipo de RNN	Ilustración	Ejemplo
Uno a muchos $T_x = 1,\ T_y > 1$		**Generación de música**
Muchos a uno $T_x > 1,\ T_y = 1$		**Clasificación de sentimientos**
Muchos a muchos $T_x = T_y$		**Reconocimiento de entidades nombradas**
Muchos a muchos $T_x \neq T_y$		**Traducción automática**

Cuadro 2.3: Ejemplos de tipos de RNNs. Adaptado de [43].

Unidades LSTM (Long Short-Term Memory)

La dificultad del desvanecimiento del gradiente al procesar secuencias muy largas fue abordada por Hochreiter y Schmidhuber en 1997 mediante las unidades LSTM. Las celdas

LSTM se caracterizan por el manejo del flujo de información, descartando datos irrelevantes y manteniendo aquellos que son significativos. En la Figura 2.13 se muestra la estructura básica de la unidad LSTM. A diferencia de la unidad recurrente básica, la LSTM posee una celda de estado, denotada como $\mathbf{C}_t$ que se encarga de gestionar el almacenamiento de la información a través del tiempo, evitando así el problema del desvanecimiento del gradiente. La LSTM también cuenta con un sistema de compuertas que se utilizan para gestionar el flujo de información hacia o desde la celda de estado [44, 45]. A continuación se describe cada una de estas compuertas:

- Compuerta de olvido: La compuerta de olvido decide qué información se remueve de la celda de estado, para ello toma el estado oculto anterior h_{t-1} y la entrada actual x_t y los pasa por una función sigmoide (ver ecuación 2.20), en este caso la matriz W_f y el sesgo b_f se aprenden durante el entrenamiento. Mediante el proceso anterior se genera el vector f_t que contiene números entre 0 y 1, donde 0 significa que se removerá por completo la información mientras que 1 significa que la información se conservará.

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{2.20}$$

- Compuerta de entrada: Esta compuerta determina qué nueva información debe almacenarse en la celda de estado, para ello, al igual que en la compuerta de olvido se toma el estado oculto anterior h_{t-1} y la entrada actual x_t y se pasan por una función sigmoide y una función tanh para crear los vectores i_t y $\hat{C}_t$ respectivamente (ver ecuaciones 2.21 y 2.22). Estos dos vectores se emplean para generar una celda de estado actualizada mediante el vector e_t (ver ecuación 2.23).

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{2.21}$$

$$\hat{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \tag{2.22}$$

$$e_t = i_t \odot \hat{C}_t \tag{2.23}$$

- Compuerta de salida: La compuerta de salida se encarga de generar el nuevo estado oculto h_t, nuevamente se reciben el estado oculto anterior h_{t-1} y la entrada actual x_t y se pasan por una función sigmoide para generar el vector o_t (ver ecuación 2.24),

este vector se multiplica punto por punto con el estado actual de la celda de estado C_t para generar el nuevo estado oculto (ver ecuación 2.25).

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{2.24}$$

$$h_t = o_t \odot tanh(C_t) \tag{2.25}$$

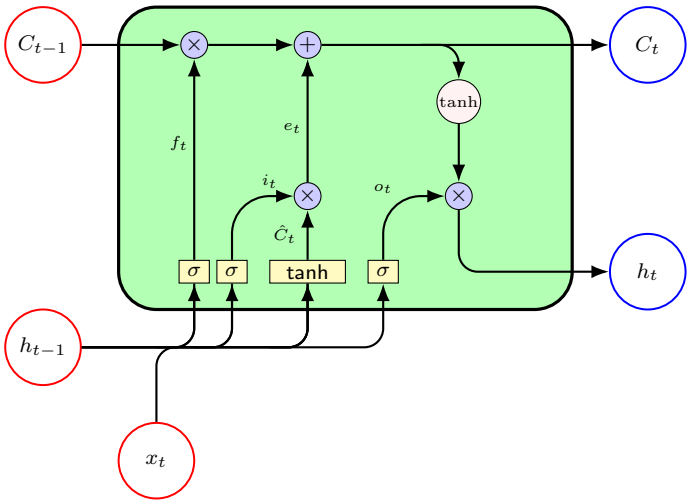


Figura 2.13: Diagrama de red neuronal recurrente LSTM.

Existen varias variantes de la unidad LSTM, siendo la unidad recurrente cerrada (GRU por sus siglas en inglés) la más utilizada, esta variante fue introducida por Cho en 2014. La arquitectura GRU ofrece una alternativa efectiva a las LSTM, simplificando los cálculos al combinar las compuertas de entrada y olvido en una única compuerta.

2.5.5. Mecanismo de atención

El mecanismo de atención fue introducido por Bahdanau en 2015 mediante una arquitectura a la que denominó RNNsearch [46]. La finalidad de dicho mecanismo fue mejorar las tareas de traducción automática de la familia de modelos codificador-decodificador. En el modelo codificador-decodificador, el codificador se encarga de generar una representación de la secuencia de entrada formando un vector de tamaño fijo, posteriormente a partir de este vector el decodificador genera una traducción. Uno de los problemas presentados con este enfoque es la necesidad de comprimir toda la información de entrada en un vector

de tamaño fijo. Para secuencias muy largas esto se puede lograr entrenando una RNN suficientemente grande, sin embargo este enfoque no es el más eficiente y pueden surgir problemas cuando a la RNN se le presenta una secuencia más larga que las secuencias del conjunto de entrenamiento [37].

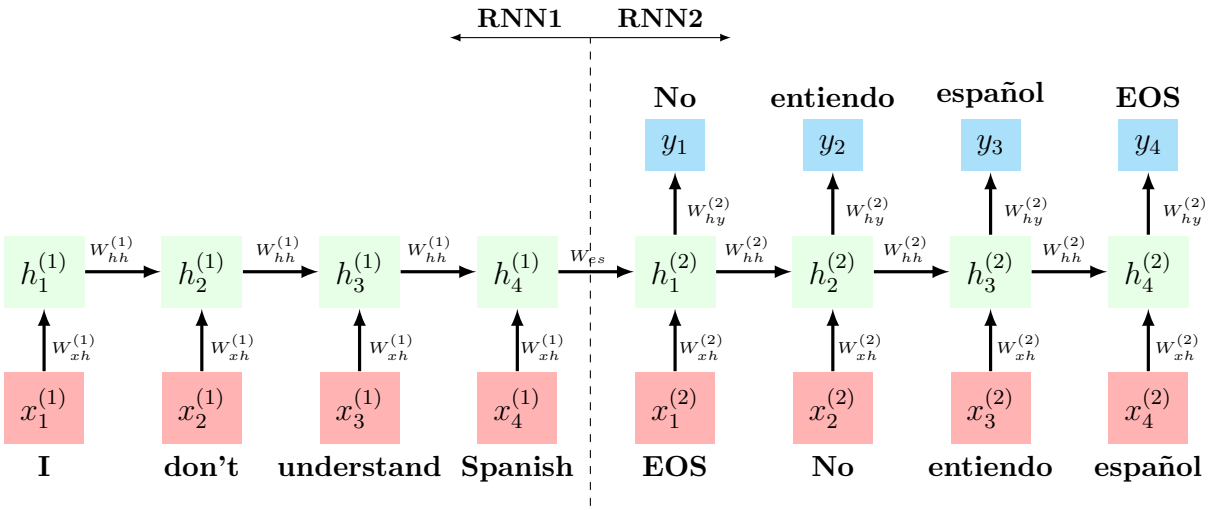


Figura 2.14: Modelo codificador-decodificador sin atención. Adaptado de [47].

La idea detrás del mecanismo de atención es concentrarse en partes determinadas de la secuencia de entrada. Esto se logra mediante la adición de un bloque de atención al decodificador. Mediante el bloque de atención se calcula el vector de contexto $\mathbf{c_t}$ el cual representa la relación de contexto entre la salida y todas expresiones del conjunto de entrada [47, 46] .

$$\mathbf{c_t} = \sum_{j=1}^{T} \alpha_{tj} h_j^{(1)} \tag{2.26}$$

donde T es la longitud de la oración de entrada, $h_j^{(1)}$ es el estado oculto de la j-esima entrada y α_{tj} es el peso de la atención el cual indica la importancia de la palabra de origen j para la palabra destino t.

$$\alpha_{tj} = \frac{\exp\left(h_j^{(1)} \cdot h_t^{(2)}\right)}{\sum_{j=1}^{T} \exp\left(h_j^{(1)} \cdot h_t^{(2)}\right)} \tag{2.27}$$

2.5.6. Transformers

El procesamiento del lenguaje natural (NLP por sus siglas en inglés) ha sido un campo de estudio de gran interés, siendo las RNN un pilar fundamental para dicho ámbito. Sin embargo, una de las grandes desventajas al trabajar con RNN es su naturaleza secuencial, ya que esto imposibilita una paralelización efectiva volviéndolas ineficientes cuando se procesan conjuntos de datos con largas secuencias [48].

La implementación de la arquitectura transformer [49] ha sido sin duda una de las aportaciones más relevantes al NLP. La innovación de los transformer recae en su mecanismo basado totalmente en atención. La estructura básica de un transformer se muestra en la Figura 2.15 que corresponde a la parte del codificador de la arquitectura propuesta por Vaswani. En la imagen se puede observar que la entrada consta de una secuencia de entrada de 4 elementos representadas con vectores de 3 dimensiones que pasan por un bloque de auto-atención multicabeza con conexiones residuales y normalización por capa. Posteriormente, los vectores resultantes de la capa de normalización pasan por una red *feedforward* con conexiones residuales para posteriormente pasar por otra capa de normalización.

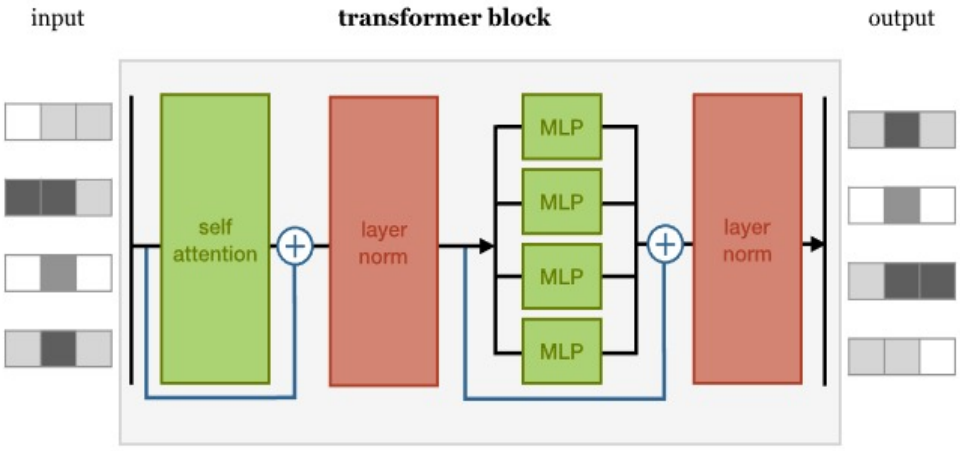


Figura 2.15: Bloque tipo transformer. Imagen obtenida de [50].

El decodificador que se encarga de mapear la entrada $X = \{x_1, x_2, \ldots, x_T\}$ a una secuencia de salida $Y = \{y_1, y_2, \ldots, y_T\}$. Una capa de atención está dada por:

$$Attention(Q, K, V) = sofmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.28}$$

donde Q, K y V son las matrices de consulta, llave y valor respectivamente, las cuales a su vez están dadas por:

$$Q = W_Q X \tag{2.29}$$

$$K = W_K X \tag{2.30}$$

$$V = W_V X \tag{2.31}$$

mientras que W_Q, W_K y W_V son parámetros entrenables.

El mecanismo de atención de múltiples cabezas se emplea para el funcionamiento de capas te atención en paralelo y está dado por:

$$\begin{aligned} &MultiHeadAttention(Q, K, V) \\ &= Concat(head_1, \ldots, head_h W^o), \\ &\text{donde} \quad head_i = Attention(Q_i, K_i, V_i) \end{aligned} \tag{2.32}$$

2.6. Aprendizaje profundo para separación del habla

En esta sección se describen algunos de los modelos más sobresalientes en la tarea de separación del habla. La finalidad es comprender y generar una visión clara de algunas de las arquitecturas propuestas hasta la actualidad.

En el Cuadro 2.4 se presentan algunos de los modelos para la separación del habla, el año en que fueron propuestos y su evaluación en el desempeño de dicha tarea mediante las métricas razón señal distorsión (SDR por sus siglas en inglés) y la razón señal a ruido invariante ante escala[1] (SI-SNRi por sus siglas en inglés) . La tabla tiene como finalidad brindar un enfoque rápido de la evolución de la separación del habla y las capacidades de cada modelo. Sin embargo, en esta sección nos limitamos a explicar las arquitecturas que se caracterizaron en este trabajo, las cuales son: ConvTasNet, Sudo rm -rf y Sepformer.

[1]Entre mayor sea el SDR y el SNRi mejor es la separación.

Modelo	Año	Base de datos	Evaluación
Recurrent Selective Attention Network (RSAN) [51]	2019	WSJ0-2mix	SDR: 13.5 [dB]
Filter-and-Sum Network (FaSNet) [52]	2019	TIMIT	SI-SNRi: 12.2[dB]
Skipping Memory (SkiM) [53]	2022	LibriSpeech	SDRi: 18.7[dB]
Tasnet [54]	2018	WSJ0-2mix	SI-SNRi: 10.8[dB]
ConvTasnet [55]	2019	Libri2Mix-Noisy	SI-SNRi: 15.3[dB]
Sepformer [56]	2021	Libri2Mix-Noisy	SI-SNRi: 22.3[dB]
TasTas [57]	2020	WSJ0-2mix	SI-SNRi: 19.47[dB]
DPTNet [58]	2020	WSJ0-2mix	SI-SNRi: 20.2[dB]
SuDoRM-RF [3]	2020	WSJ0-2mix	SI-SNRi: 18.9[dB]
Chimera++ [59]	2017	WSJ0-2mix	SI-SNRi: 11.5[dB]

Cuadro 2.4: Modelos de separación del habla y sus evaluaciones.

2.6.1. ConvTasNet

La arquitectura [55] ConvTasNet (*Convolutional Time-Domain Audio Separation Network*) es un modelo de redes neuronales convolucionales utilizado para la separación de fuentes de audio. Fue propuesto como una solución eficaz y flexible para abordar el desafío de separar señales de audio mezcladas en entornos complejos.

En ConvTasNet, la red se compone de tres componentes principales: un codificador (*encoder*), una etapa de separación y un decodificador (*decoder*). El extractor de características se encarga de capturar las características relevantes de la señal de audio de entrada,

mientras que la red de separación se encarga de separar las diferentes fuentes de audio.

El extractor de características utiliza capas convolucionales para procesar la señal de audio en el dominio del tiempo. Estas capas convolucionales están diseñadas para capturar patrones locales en la señal y extraer representaciones de características. El uso de convoluciones permite que el modelo aprenda a extraer características relevantes en diferentes escalas de tiempo, lo que es crucial para la separación de fuentes de audio.

Después de pasar por el extractor de características, la señal se transforma en una representación de características de alta dimensionalidad. A continuación, esta representación se pasa a la red de separación, que se compone de capas convolucionales y capas de agrupación (*pooling*). Estas capas permiten que la red aprenda a separar las fuentes de audio mezcladas con base a las características extraídas previamente.

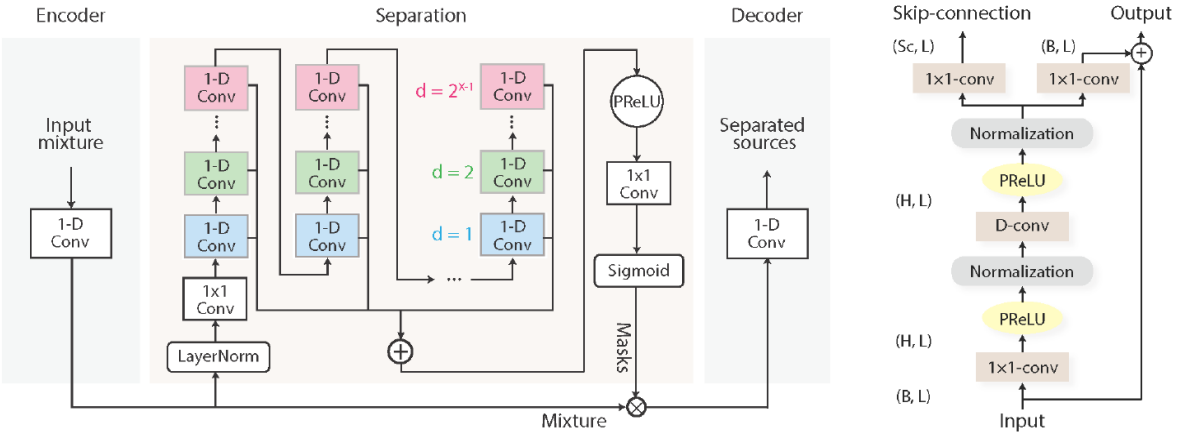


Figura 2.16: Arquitectura ConvTasNet. Imagen obtenida de [55].

2.6.2. Sudo rm -rf

La arquitectura Sudo rm -rf propuesta por [3] se centra en el muestreo y remuestreo múltiple de características. Al igual de la arquitectura ConvTasNet, Sudo rm -rf se compone de un codificador (*encoder*) una etapa de separación y un decodificador (*decoder*). La red recibe en el codificador una señal mezclada $\mathbf{x}$ y este genera una representación latente. La representación latente es entregada al modulo de separación el cual se encarga de generar las mascaras $\mathbf{m}_i$ para cada una de las fuentes que componen la mezcla. Posteriormente la representación latente generada por el codificador es multiplicada por cada una de

las mascaras generadas por el modulo de separación. Por ultimo, el resultado de estas multiplicaciones son procesadas por el decodificador el cual las transforma del estado latente al dominio del tiempo.

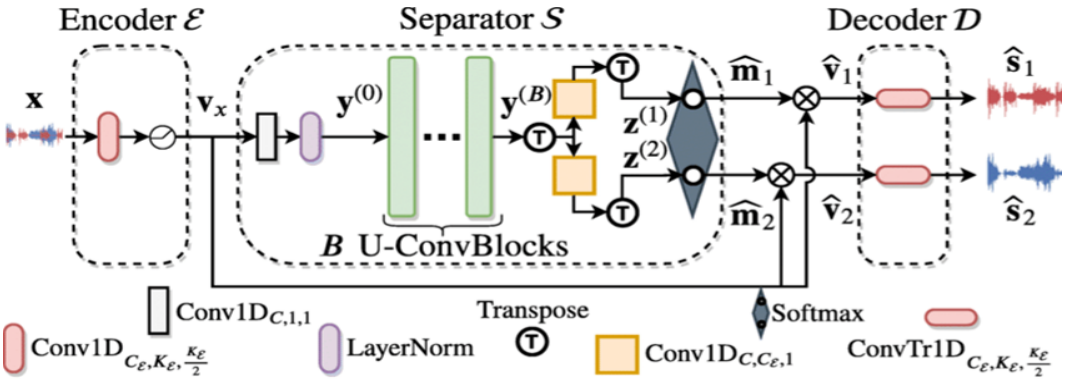


Figura 2.17: Arquitectura Sudo rm -rf. Imagen obtenida de [3]

2.6.3. Sepformer

En [48] proponen SepFormer (ver Figura 2.18), un transformer para la separación del habla. Sepformer cuenta con un *encoder* el cual recibe la señal de audio mezclada dividida en determinados segmentos de tiempo, el *encoder* se encarga de extraer características iniciales de la señal. Este transformer también cuenta con un codificador posicional el cual le permite obtener información de la relación entre las diferentes partes de la señal en el dominio temporal. Posterior al encoder, las características codificadas se pasan a través de múltiples capas Transformer. Cada capa cuenta con un mecanismo de atención multi-cabeza lo cual le permite aprender una representación de la señal de audio. Por último, las características resultantes se pasan a través de un decoder que produce las señales separadas.

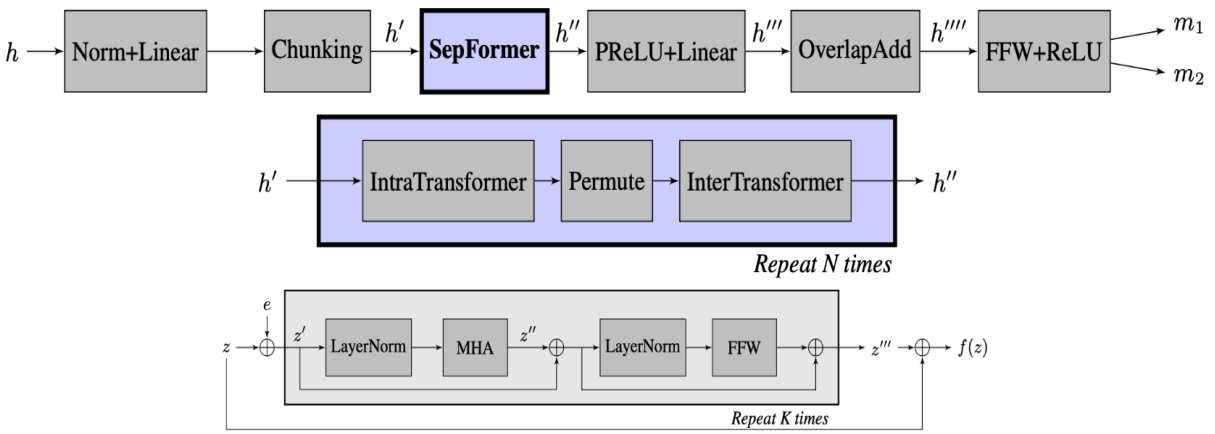


Figura 2.18: Arquitectura de sepformer. Imagen obtenida de [48].

2.7. Aprendizaje profundo para mejora del habla

En esta sección se presentan algunos de los modelos empleados para mejora del habla. Algunos modelos de separación del habla como lo son Sudo rm -rf y ConvTasNet también son citados en trabajos de mejora del habla. Por dicha razón es importante destacar la distinción entre estas dos técnicas. La principal diferencia recae en que la mejora del habla suprime el ruido proveniente del entorno donde se realizó la grabación, mientras que la separación del habla se centra en extraer al hablante objetivo de la mezcla de voces.

Modelo	Año	Base de datos	Evaluación
Demucs-Denoiser	2020	Valentini	PESQ: 3.07
MetricGAN	2019	TIMIT	PESQ: 2.86
DCCRNet	2019	WSJ0	PESQ: 2.97
FullSubNet	2020	DNS	PESQ: 2.96
Mlt mimic loss	2018	Voicebank	PESQ: 3.05

Cuadro 2.5: Modelos de mejora del habla y sus evaluaciones.

2.7.1. Demucs

Demucs-denoiser [60], es una arquitectura de red neuronal utilizada para la separación de fuentes de audio y la reducción de ruido en señales de audio. Esta arquitectura se basa en el modelo original Demucs (*Deep Extractor for Music Sources*) y ha sido diseñada específicamente para mejorar la calidad de las señales de audio mediante la eliminación del ruido no deseado.

La arquitectura comienza con un codificador (*encoder*) que se encarga de mapear la señal de audio de entrada a una representación latente. En el caso de Demucs-denoiser, este codificador se compone de capas de convolución y funciones de activación, que extraen características relevantes de la señal de audio y la comprimen en una representación de menor dimensión. El codificador se encuentra conectado a decodificador (*decoder*) mediante conexiones de salto, tomando dicha característica de la arquitectura U-Net. La capa de separación está constituida por una LSTM bidireccional lo que le permite captar las componentes temporales y lograr una buena separación.

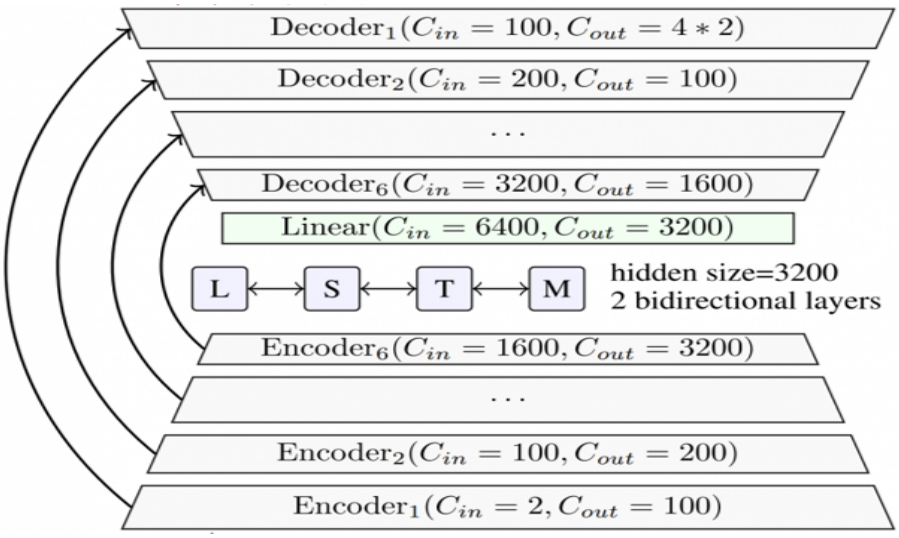


Figura 2.19: Arquitectura demucs. Imagen obtenida de [60]

2.7.2. DCCRNet

La arquitectura *Deep Complex Convolution Recurrent Network* (DCCRNet) propuesta por [61] extiende a la estructura general de una red recurrente convolucional. El modelo

DCCRNet cuenta con la estructura codificador-decodificador. Los autores proponen añadir un codificador complejo, el cual consta de Conv2d complejas y normalización por lotes compleja. En este caso, la operación Conv2d incluye el filtro convolucional de valores complejos el cual se define mediante la ecuación:

$$W = W_r + jW_i \tag{2.33}$$

donde W_r y W_i son las matrices que representan la parte real e imaginaria de la convolución compleja respectivamente. Para poder efectuar la convolución se recibe una entrada compleja como se muestra en la ecuación 2.34.

$$X = X_r + jX_i \tag{2.34}$$

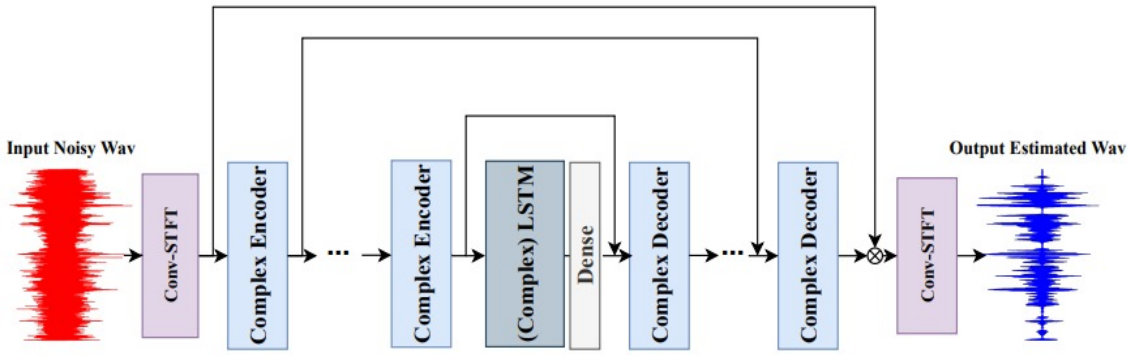


Figura 2.20: Arquitectura de DCCRNet. Imagen obtenida de [61].

2.7.3. MLT mimic loss

Multitask mimic loss o también llamado mapeo de características espectrales con pérdida mímica (spectral feature mapping with mimic loss) se refiere a una metodología en la cual se combina la mejora del habla con reconocimiento automático del habla (ASR por sus siglas en inglés). De acuerdo con [62] el proceso comienza con el entrenamiento de un clasificador espectral de voz limpia mediante entropía cruzada binaria para reconocer y agrupar eventos sonoros similares, conocidos como “senones”. De igual forma se entrena mapeador espectral, este módulo tiene la función de transformar la señal de habla ruidosa

en una versión más limpia. Para el entrenamiento del mapeador espectral se emplea como función de perdida el error cuadrático medio.

Posteriormente, se utiliza una combinación de pérdidas provenientes tanto de la señal de habla limpia como de las salidas del clasificador cuando se le proporciona habla limpia (esto es lo que se refiere como "mimic loss"). Esta técnica se basa en la idea de que el mapeador espectral debe aprender a producir una salida que se asemeje a la representación que el clasificador espectral generaría a partir de la señal de habla limpia.

3 Metodología

En esta sección se presenta la metodología a detalle empleada para la elaboración de este trabajo. Se incluye el procedimiento seguido durante el entrenamiento y el análisis de datos con la finalidad de realizar una adecuada comparación entre las distintas técnicas de separación y mejora del habla.

3.1. Elección de la base de datos

El enfoque inicial de este trabajo se centró en realizar un estudio detallado del estado del arte en separación de voz, así como examinar las bases de datos disponibles para efectuar dicha tarea. En el Cuadro 3.1 se presentan algunas de las bases de datos encontradas en la literatura así como una breve descripción de cada una. Un aspecto importante a mencionar es que no todas las bases de datos tienen acceso libre, es decir, para algunas bases de datos se debe pagar una licencia para su uso.

Nombre	Descripción	Fecha	Acceso	Referencia
AVSpeech	Base de datos de discursos de voces en entornos audiovisuales.	2018	Libre	[63]
DAMP-VSEP	Base de datos de mezclas de música para separación de voz y acompañamiento.	2019	Restringido	[64]
FUSS	Free Universal Sound Separation dataset.	2020	Libre	[65]
Kinect-WSJ	Base de datos con grabaciones de habla en entornos Kinect.	2019	Restringido	[66]
LibriMix	Mezclas de habla generada con el conjunto de datos LibriSpeech.	2020	Libre [67]	
MUSDB18	Multi-track dataset para separación de música en mezclas.	2017	Libre	[68]
SMS-WSJ	Base de datos de mezclas de habla para separación de voz.	2019	Restringido	[69]
WHAM!	Utiliza la base de datos wsj0-2mix para crear mezclas de hablantes con ruido de fondo.	2019	Restringido	[70]
WHAMR!	Extiende a WHAM! incorporando reverberación.	2019	Restringido	[71]
MedleyVox	Base de datos de canciones en mezclas estéreo para separación de voz.	2022	Libre	[72]
LibriCSS	Utiliza señales de audio superpuestas para similar audio conversacional.	2020	Libre	[73]

Cuadro 3.1: Bases de datos para separación de voz.

Se llevó a cabo la selección de un conjunto de datos adecuada para el entrenamiento de los modelos propuestos en el campo de la separación de voz. Se buscaron audios que representaran escenarios realistas, incluyendo condiciones de ruido y reverberación, con el objetivo de mejorar la capacidad de generalización de los modelos. Además, se consideró crucial que la información estuviera disponible de forma gratuita.

Tras un exhaustivo análisis, se optó por utilizar la base de datos LibriMix [74], que combina grabaciones de la base de datos de LibriSpeech con mezclas simuladas. Esta elección se basó en el hecho de que LibriMix ofrece un entorno de entrenamiento más realista

al incorporar tanto grabaciones de habla limpia como mezclas con ruido y reverberación simulados. Otro factor clave para la elección fue que de acuerdo con [67] ha tenido un gran impacto ya que ha sido utilizada en diversos trabajos de investigación.

LibriMix cuenta con una versión con dos hablantes denominada Libri2Mix y una versión con tres hablantes llamada Libri3Mix. Para este trabajo se creó Libri2Mix utilizando el script disponible en el repositorio de GitHub[1]. Libri2Mix consta del conjunto de entrenamiento "train-100"que incluye 13,900 grabaciones muestreadas a 8 kHz, y el conjunto de validación "dev"que contiene 3,000 grabaciones muestreadas a la misma frecuencia. Para llevar a cabo el entrenamiento de los modelos en este proyecto, se generaron subconjuntos de entrenamiento y validación más pequeños, compuestos por 1360 datos para entrenamiento y 340 para validación, lo que corresponde a un 80 % y un 20 %, respectivamente.

3.2. Entrenamiento de los modelos

Para el desarrollo del sistema de separación de voz se llevó a cabo una búsqueda de herramientas y recursos que pudieran brindar una mejor comprensión de las arquitecturas propuestas para abordar esta tarea. Se optó por utilizar las bibliotecas de herramientas (toolkits) Asteroid [75] y Speechbrain [76] para el entrenamiento de los modelos.

Para el entrenamiento de los modelos se utilizó un servidor que se encuentra en el laboratorio del Dr. Caleb Antonio Rascón Estebané el cual se puede usar de forma remota. El servidor cuenta con las siguientes características:

- Procesador Intel core i7 6700k.
- 2 tarjetas gráficas NVIDIA GTX 1080.
- 64 GB de RAM.

3.2.1. Entrenamiento con Asteroid

Asteroid es un toolkit especializado en la separación de fuentes de sonido. Está basado en Pytorch y Pytorch-lightning, provee un amplia variedad de arquitecturas para la sepa-

[1] https://github.com/JorisCos/LibriMix.git

ración de fuentes de sonido, de igual forma, proporciona una serie de *recetas* que permiten reproducir algunos de los artículos más importantes en el estado del arte.

Se empleó Asteroid para el entrenamiento de los modelos: ConvTasNet, Sudo rm -rf, y DCCRN. El entrenamiento consiste en importar el modelo seleccionado para posteriormente generar una instancia de este. En el repositorio de Asteroid están los códigos fuente de los modelos, donde se pueden consultar los parámetros que se pueden modificar para el entrenamiento. Los parámetros varían de acuerdo a la arquitectura seleccionada. A continuación se presentan los parámetros para cada modelo.

ConvTasNet	
Símbolo	**Descripción**
bn_chan	Número de canales después del *bottleneck*.
n_blocks	Número de bloques convolucionales en cada repetición.
kernel	Tamaño del kernel en los bloques convolucionales.
hid	Número de canales en los bloques convolucionales

Cuadro 3.2: Parámetros de la arquitectura ConvTasNet.

Sudo rm -rf	
Símbolo	**Descripción**
K	Tamaño del kernel.
C	Número de canales en cada bloque *U-ConvBlock*.
B	Número de bloques *U-ConvBlock*.
Q	Profundidad de sobremuestreo.

Cuadro 3.3: Parámetros de la arquitectura Sudo rm -rf.

DCCRNet	
Símbolo	**Descripción**
stft_kernel_size	Longitud de ventana de STFT a utilizar.
stft_stride	Longitud del salto STFT a utilizar.
sample_rate	Frecuencia de muestreo del modelo.

Cuadro 3.4: Parámetros de la arquitectura DCCRNet.

Para el entrenamiento de los modelos se utilizó el criterio llamado: entrenamiento invariante de permutación (PIT por sus siglas en inglés) [77]. Este criterio permite resolver el problema de la permutación de etiquetas, es decir, permite identificar las fuentes correctas sin importar el orden en el que aparezcan en la mezcla de audio. Para ello, PIT evalúa todas las permutaciones posibles y selecciona la que minimiza la función de perdida. Como función de perdida se utilizó la razón señal-distorsión invariante de escala (SI-SDR).

3.2.2. Entrenamiento con Speechbrain

Speechbrain es una biblioteca de código abierto especializada en el procesamiento de audio. A diferencia de Asteroid que se enfoca únicamente en la separación de fuentes de sonido, Speechbrain incluye diversas herramientas para el procesamiento del habla en general, incluyendo tareas de reconocimiento de voz, clasificación de voz, etc.

Speechbrain fue utilizado para el entrenamiento del modelo Sepformer. Los hiperparámetros del modelo se definen en un archivo *.yaml*, los parámetros que se pueden modificar se presentan a continuación:

Entrenamiento:

- N_epochs = número de épocas.
- batch_size = tamaño de lote.
- lr = taza de aprendizaje.

Arquitectura:

- Codificador y decodificador:
  - in_channels = dimensión de entrada.
  - out_channels = dimensión de salida.
  - kernel_size = tamaño del kernel.
  - kernel_stride = desplazamiento.
- Mascara de frecuencias:
  - in_channels = dimensión de entrada.
  - out_channels = dimensión de salida.
  - num_layers = número de capas.
  - K = tamaño del segmento.
  - Inter e intra transformers:
    - num_layers = capas de auto-atención.
    - nhead = número de cabezas de atención.

Figura 3.1: Hiperparámetros de la arquitectura Sepformer.

Para el entrenamiento de Sepformer, también se empleó el enfoque PIT, minimizando la función de perdida SI-SDR.

3.2.3. Modelos pre-entrenados

Con la finalidad de realizar una comparación más amplia se incluyeron los modelos: Demucs-Denoiser y Spectral Feature Mapping with Mimic Loss. Dichos modelos cuentan con un pre-entrenamiento el cual puede ser descargado de *Hugging Face* para el caso de Spectral Feature Mapping with Mimic Loss[2], y mediante el uso de la librería Demucs[3] para

[2] https://huggingface.co/speechbrain/mtl-mimic-voicebank

[3] https://github.com/facebookresearch/denoiser

Demucs-Denoiser.

3.3. Sistema de separación de voz en línea

La implementación del sistema de separación de voz en línea sigue un enfoque basado en la técnica de segmentación por ventanas, una metodología comúnmente empleada en aplicaciones de procesamiento de audio de baja latencia, como lo es JACK Audio Connection Kit [78]. De forma general, el sistema consiste en tomar un ejemplo de la base de datos para evaluación, aplicar separación por ventanas y hacer inferencia con el modelo (ver Figura 3.2).

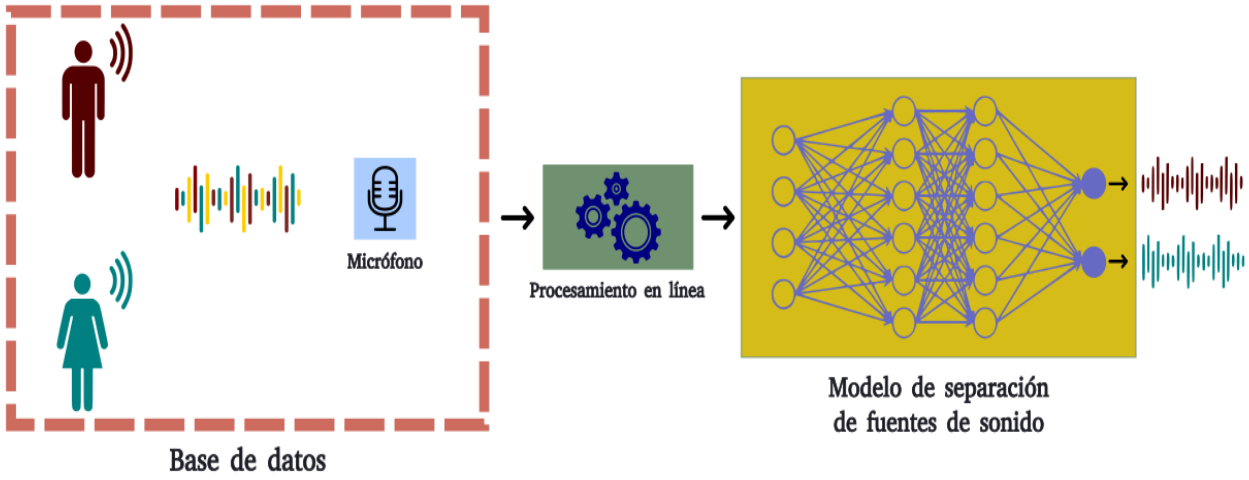


Figura 3.2: Diagrama donde se presenta la metodología a seguir para la separación de fuentes de sonido en linea.

3.3.1. Segmentación por ventanas

La segmentación por ventanas consiste en dividir una señal entera en pequeños segmentos a los cuales denominamos como ventanas. Cada una de las ventanas es procesada consecutivamente en un instante de tiempo t con la finalidad de que el sistema entregue una salida constante de información.

Supongamos que se tiene una señal de audio $x[t]$. Definimos un tamaño de ventana W de tal forma que podemos expresar la señal como ventanas individuales $x_n[t]$, donde n es

el índice de la ventana. Cada ventana n abarca desde t_n hasta t_{n+1}, donde $t_n = n \cdot W$ y $t_{n+1} = (n+1) \cdot W$, con esto podemos expresar a la n-ésima ventana como:

$$x_n[t] = \begin{cases} x[t], & \text{si } t \in [t_n, t_{n+1}] \\ 0, & \text{en otro caso} \end{cases} \tag{3.1}$$

Ahora definamos una función de procesamiento $P(x)$, la cual se aplica a cada una de las ventanas y genera una salida de tal forma que:

$$y_n[t] = P(x_n[t]) \tag{3.2}$$

Por último, la señal procesada final se obtiene al concatenar todas las ventanas procesadas $y_n[t]$:

$$y[t] = \sum_{n=0}^{T} y_n[t] \tag{3.3}$$

3.3.2. Obtención del factor de tiempo real (RTF)

Una métrica que es útil para indagar si un sistema de procesamiento de audio puede funcionar en línea es el factor de tiempo real, este está dado por la relación entre el tiempo necesario para procesar la entrada y la duración de la entrada.

Si el tiempo que toma aplicar la función de procesamiento $P(x)$ es t_p, el factor de tiempo real se puede expresar como:

$$RTF = \frac{t_p}{t_{n+1} - t_n} \tag{3.4}$$

Si el factor de tiempo real es inferior a 1, significa que el procesamiento es más rápido que la velocidad de reproducción de las ventanas, lo que asegura que el sistema funcione en tiempo real sin retrasos notables. En nuestro caso, el tiempo que lleva aplicar la función de procesamiento $P(x)$ a una ventana es un componente crítico para alcanzar un alto factor de tiempo real.

Para garantizar una experiencia eficiente, nuestro objetivo es optimizar el proceso de $P(x)$ para que funcione en un tiempo menor o igual a la duración de una ventana de audio.

La optimización se centra en la elección de algoritmos eficientes, técnicas de paralelización y ajuste de parámetros para lograr el mejor equilibrio entre la calidad del procesamiento y la velocidad de ejecución.

3.3.3. Pre-procesamiento de los datos

Para llevar a cabo la evaluación de los modelos de separación y mejora del habla, se empleó la base de datos AIRA [79] que incluye grabaciones de sonido de entornos reales con variados parámetros, tales como: el nivel de ruido, la reverberación y el número de micrófonos utilizados.

Se generaron tres subconjuntos de la base de datos AIRA, correspondientes a uno, dos y tres hablantes respectivamente, cada uno con 12 muestras. Debido a que cada muestra dura aproximadamente 30 segundos, se procedió a reducir su duración a 15 segundos con la finalidad de agilizar el proceso de inferencia. Además, es importante destacar que la base de datos AIRA fue grabada a una frecuencia de muestreo de 48000 Hz, por lo que se realizaron re-muestreos a 8000 Hz y 16000 Hz, que son las frecuencias con las cuales los modelos han sido entrenados.

3.3.4. Obtención de las métricas de evaluación para la separación y mejora del habla

Las métricas que se emplearon para la evaluación del sistemas de separación y mejora del habla son:

Métricas para la separación del habla

Las métricas de desempeño para determinar que tan buena es la separación del habla se calculan por cada fuente estimada $\hat{s}_j$, la cual es comparada con una fuente verdadera dada s_{tarjet}. La fuente estimada $\hat{s}_j$ se puede descomponer como:

$$\hat{s}_j = s_{target} + e_{interd} + e_{noise} + e_{artif} \tag{3.5}$$

donde e_{interd}, e_{noise} y e_{artif} son los términos para el error de interferencia, ruido y artefactos respectivamente. A partir de la descomposición de la señal estimada, se calculan las siguientes métricas de desempeño que representan las relaciones de energía en decibeles (dB), entre mayor sea el valor numérico calculado, mayor es la presencia de la fuente de interés estimada [80]:

- Razón de Señal Distorsión (SDR): Se considera como una medida general de que también suena una fuente de sonido.

$$SDR = 10\log_{10}\frac{\|s_{target}\|^2}{\|e_{interd} + e_{noise} + e_{artif}\|^2}$$

- Razón Señal Interferencia (SIR): Representa la cantidad de otras fuentes en la estimación de la fuente de interés.

$$SIR = 10\log_{10}\frac{\|s_{target}\|^2}{\|e_{interf}\|^2}$$

- Razón Señal Artefactos (SAR): Representa la cantidad de artefactos insertados durante la separación.

$$SAR = 10\log_{10}\frac{\|s_{target} + e_{interf} + e_{noise}\|^2}{\|e_{artif}\|^2}$$

Métricas para la mejora del habla

- Evaluación Perceptiva de la Calidad del Habla (PESQ): Es una métrica utilizada para medir la calidad del audio teniendo en cuenta aspectos como: nitidez del audio, volumen, ruido e interferencia. El valor de esta métrica puede variar en un rango de -0.5 a 4.5 donde las puntuaciones más altas significan una mejor calidad del audio [81].

- Inteligibilidad Objetiva a Corto Plazo (STOI): STOI es una métrica empleada para evaluar algoritmos de reducción de ruido y su valor va de 0 a 1, donde un STOI más alto indica una mayor inteligibilidad [82].

Para calcular las métricas para evaluar la separación del habla se ocupó un implementación llamada: mir_eval [83]. En cuanto a las métricas para evaluar la mejora del habla se ocuparon las implementaciones: pesq y pystoi [84, 85].

3.4. Caracterización del sistema

Para la caracterización del funcionamiento de los modelos en línea, se realizó la conexión entre los modelos entrenados y el sistema de segmentación por ventanas descrito previamente. Se midió el tiempo de respuesta en función de la longitud de la secuencia de entrada procesada.

Se emplearon los subconjuntos previamente generados a partir de la base de datos AIRA para llevar a cabo 10 inferencias por cada una de las longitudes de entrada que se probaron (ver Cuadro 3.5), esto se realizó para cada muestra del subconjunto. El procesamiento consistió en segmentar al audio completo en ventanas de determinada duración con las cuales se iba alimentado al modelo para generar ventanas separadas que posteriormente fueron concatenadas para generar el audio completo separado (ver Figura 3.3).

Las especificaciones del equipo de cómputo con el cual se realizaron las inferencias son las siguientes:

- **Procesador:**
  - Modelo: AMD® Ryzen 3 2300u
  - Velocidad: 2.00 GHz
  - Núcleos: 4
- **Memoria RAM:**
  - Capacidad: 12 GB
  - Tipo: DDR4 SDRAM 300MHz

Durante cada inferencia se obtuvieron las siguientes métricas de evaluación:

- Tiempo de respuesta
- Factor de tiempo real
- Razón señal-interferencia

Además, para el caso de los modelos de mejora del habla se incluyeron las siguientes métricas:

- PESQ
- STOI

	Longitud de secuencia de entrada [s]						
Modelos	0.128	0.256	0.512	1.024	2.048	4.096	8.192

Cuadro 3.5: Longitudes de secuencia de entrada empleadas para hacer inferencia con los modelos de mejora y separación del habla.

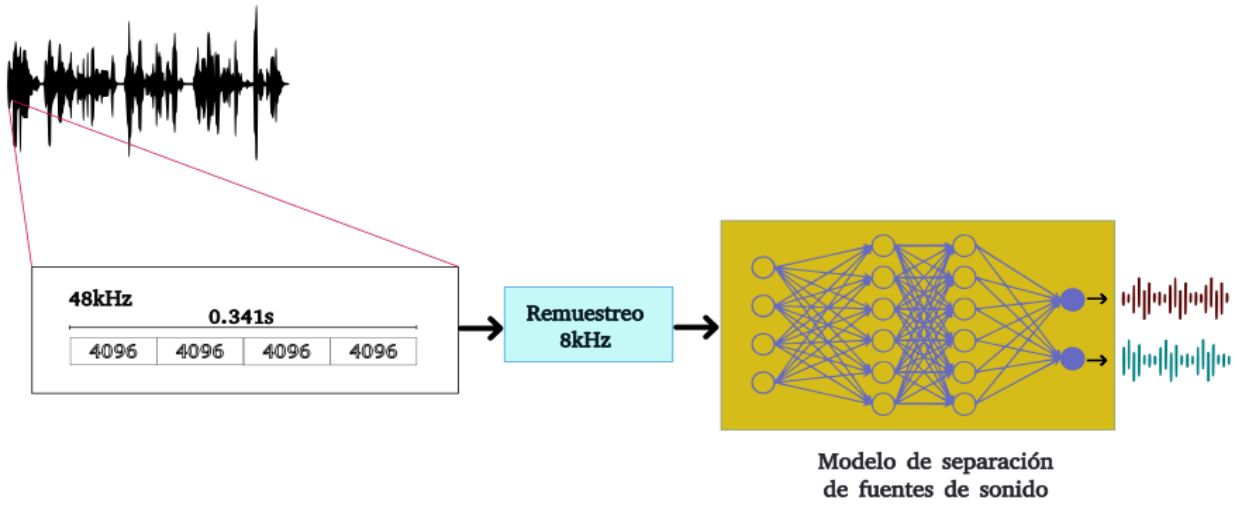


Figura 3.3: Diagrama donde se presenta la metodología a seguir para el procesamiento en línea. Se ejemplifica el procesamiento con una longitud de secuencia de entrada de 0.341s.

4 Resultados y discusión

En esta sección se presentan los distintos resultados obtenidos durante la evaluación de las arquitecturas de separación y mejora del habla. De igual forma, se realiza una discusión de dichos resultado con la finalidad de destacar los aspectos más relevantes del presente trabajo. Con el objetivo de una organización más amena para el lector, tanto los resultados como la discusión se encuentran divididos en secciones para modelos de separación del habla y modelos de mejora del habla. Se incluye un análisis de los modelos haciendo inferencia con el audio completo como una sola ventana, esto con la finalidad de tener un contraste con el funcionamiento en línea y el funcionamiento fuera de línea.

Las arquitecturas exhibidas en los resultados son las que mejor desempeño mostraron, dado que se sometieron a entrenamiento un total de 30 modelos con diversos parámetros: 10 para Sepformer, 10 para CovTasNet y 10 para Sudo rm -rf (ver apéndice A).

4.1. Modelos para separación del habla

ConvTasNet:

A continuación se presentan los parámetros para el entrenamiento de la arquitectura ConvTasNet que mejor desempeño mostraron.

Modelo	ConvTasNet
n_blocks	8
hid_chan	512
conv_kernel_size	3
n filters	64
kernel_size	16
stride	8
optimizer	adam
learning_rate	1e-3
epochs	400
batch	4

Cuadro 4.1: Parámetros para la arquitectura ConvTasNet.

En la Figura 4.1 se ilustran los diagramas de cajas y bigotes obtenidos de las inferencias que se realizaron con la arquitectura ConvTasNet. Se gráfica el tiempo de respuesta con respecto a la variación de la longitud de entrada.

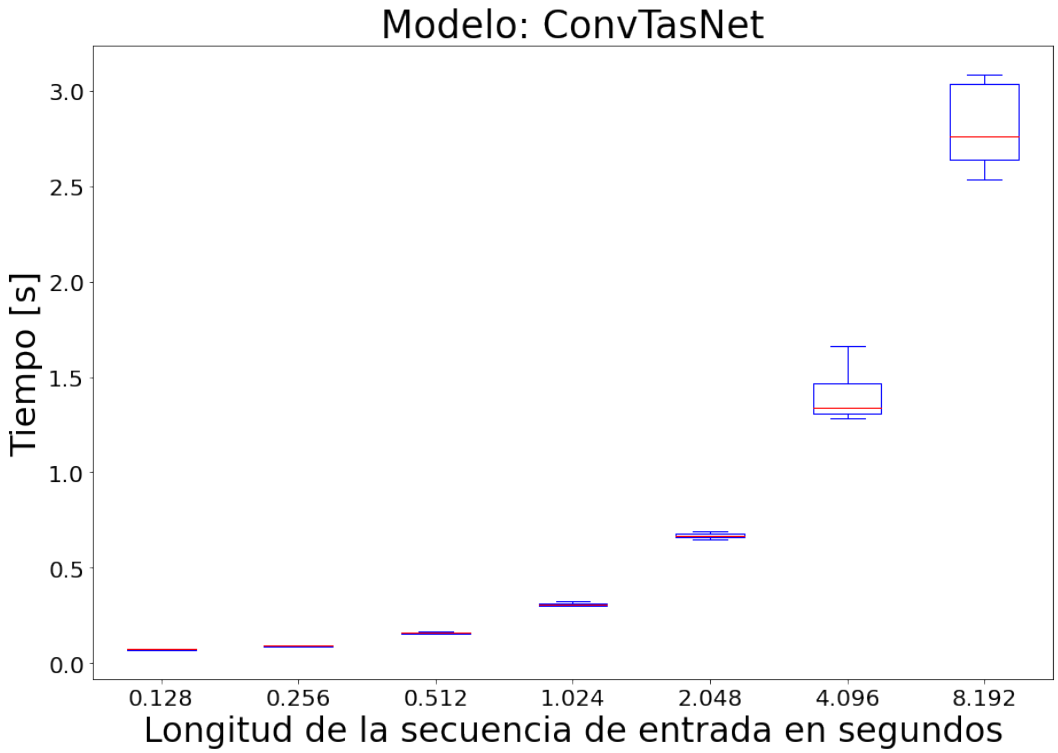


Figura 4.1: Diagrama de cajas y bigotes del tiempo de respuesta con respecto a la variación de la longitud de entrada para el modelo ConvTasNet.

La Figura 4.1 muestra una tendencia positiva entre el tiempo de respuesta y la longitud de secuencia de entrada. Esto implica que a medida que la longitud de la secuencia de entrada aumenta, al modelo le toma más tiempo para procesarla y generar una respuesta. Sin embargo, podemos observar que en todos los casos el tiempo de respuesta es menor a la longitud de la secuencia de entrada que se está procesando, lo que nos indica que es factible su funcionamiento en línea en un dispositivo con los recursos mencionados en el presente trabajo.

Algo importante a destacar, es la complejidad de los algoritmos de separación. Aunque la Figura 4.1 podría sugerir una complejidad superior a $O(n)$, es importante tener en cuenta que, debido a consideraciones de visualización, la escala en el eje x no representa con precisión los valores reales. Un análisis más detallado sobre este aspecto se presenta en el Apéndice B.

A continuación se presentan los espectrogramas de la mezcla de las fuentes, que inclu-

yen tanto ruido como reverberación, así como los espectrogramas de una de las fuentes separadas para cada una de las diferentes longitudes de secuencia de entrada.

En la Figura 4.2 podemos observar que conforme la longitud de secuencia de entrada aumenta se presenta un espectrograma más limpio. Es decir, se eliminan múltiples frecuencias que corresponden a uno de los hablantes, logrando de esta forma la separación. Esto nos indica que la arquitectura ConvTasNet funciona de una mejor forma para secuencias de audio más largas.

Por último, en la Figura 4.3 se presenta la razón señal-interferencia en función de la longitud de secuencia de entrada para las inferencias realizadas con la arquitectura ConvTasNet.

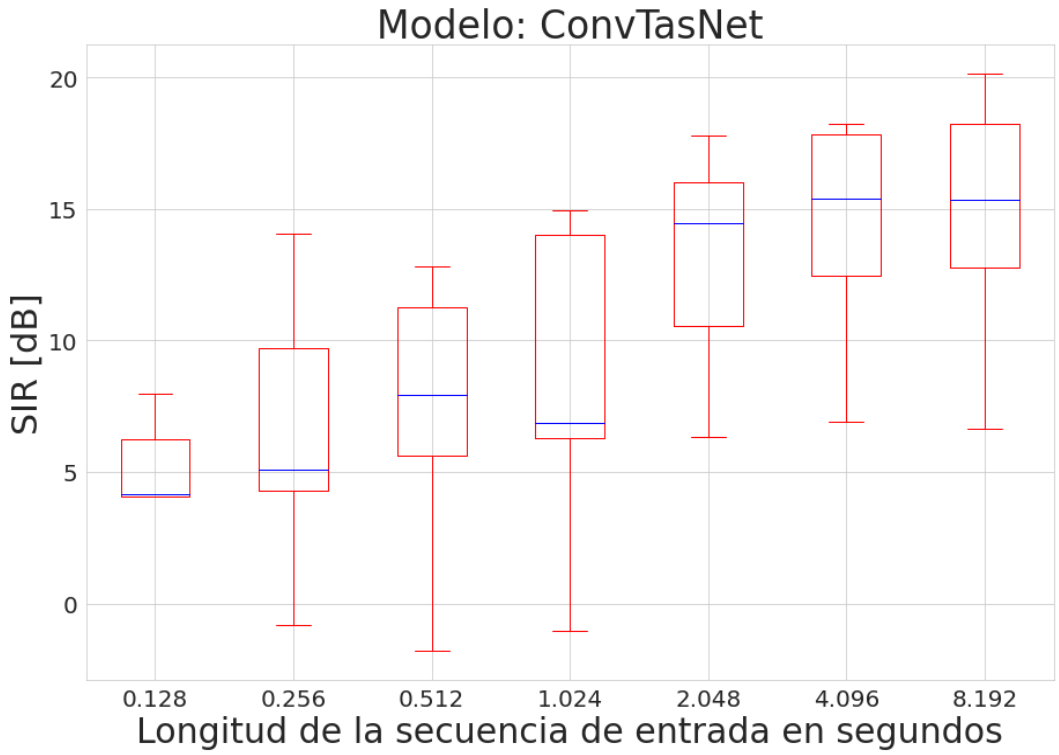


Figura 4.3: Diagrama de cajas y bigotes de la razón señal-interferencia (SIR) con respecto a la variación de la longitud de entrada para el modelo ConvTasNet.

Es evidente que a medida que aumenta la longitud de la secuencia de entrada, se observa un incremento significativo en el rendimiento reflejado por el SIR. Un aspecto importante de mencionar es que para secuencias de entrada menores a 2.048 segundos,

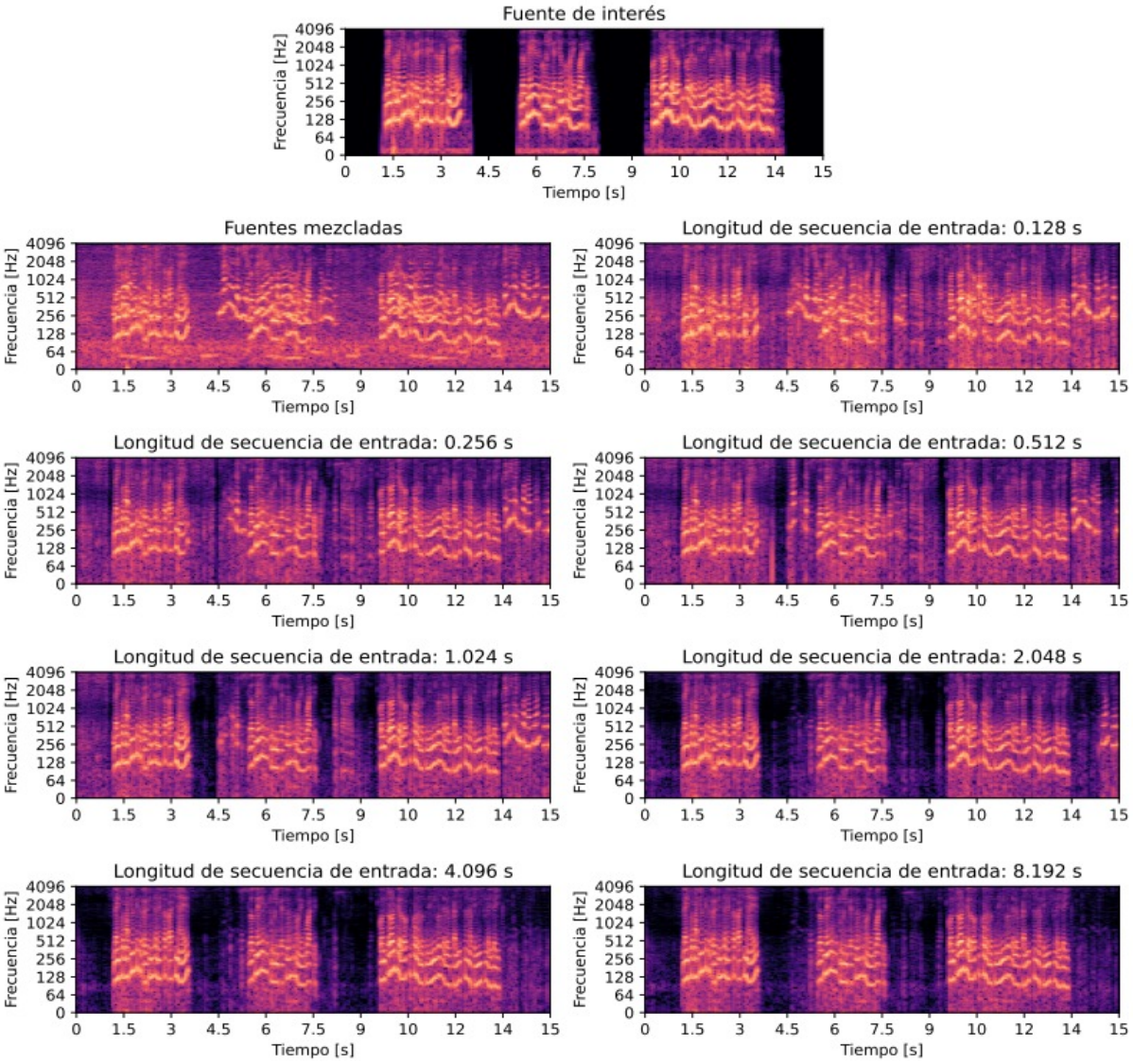


Figura 4.2: Espectrogramas de la separación de fuentes de sonido mediante el uso de la arquitectura ConvTasNet. Se presentan los espectrogramas para las diferentes longitudes de secuencia de entrada.

la separación puede presentar cierta limitación en la reducción de la presencia de ruido, reverberación y la otra fuente. Sin embargo, a partir de este punto, se observa un aumento en la capacidad de la arquitectura para realizar la separación. Los hechos mencionados anteriormente concuerdan con los espectrogramas mostrados en la Figura 4.2.

Sudo rm -rf:

En el Cuadro 4.2 se presentan los parámetros utilizados para el entrenamiento de la arquitectura Sudo rm -rf que mejoró el desempeño.

Modelo	Sudo rm -rf
bn_chan	128
num_blocks	16
upsampling_depth	4
kernel_size	21
n_filters	512
stride	10
optimizer	adam
learning rate	1e-3
epochs	200
batch	12

Cuadro 4.2: Parámetros para la arquitectura Sudo rm -rf.

Para visualizar el rendimiento de la arquitectura, en la Figura 4.4 se presentan los diagramas de cajas y bigotes obtenidos de las inferencias hechas con el modelo. Se presenta la variabilidad del tiempo de respuesta en función de la longitud de de secuencia de entrada.

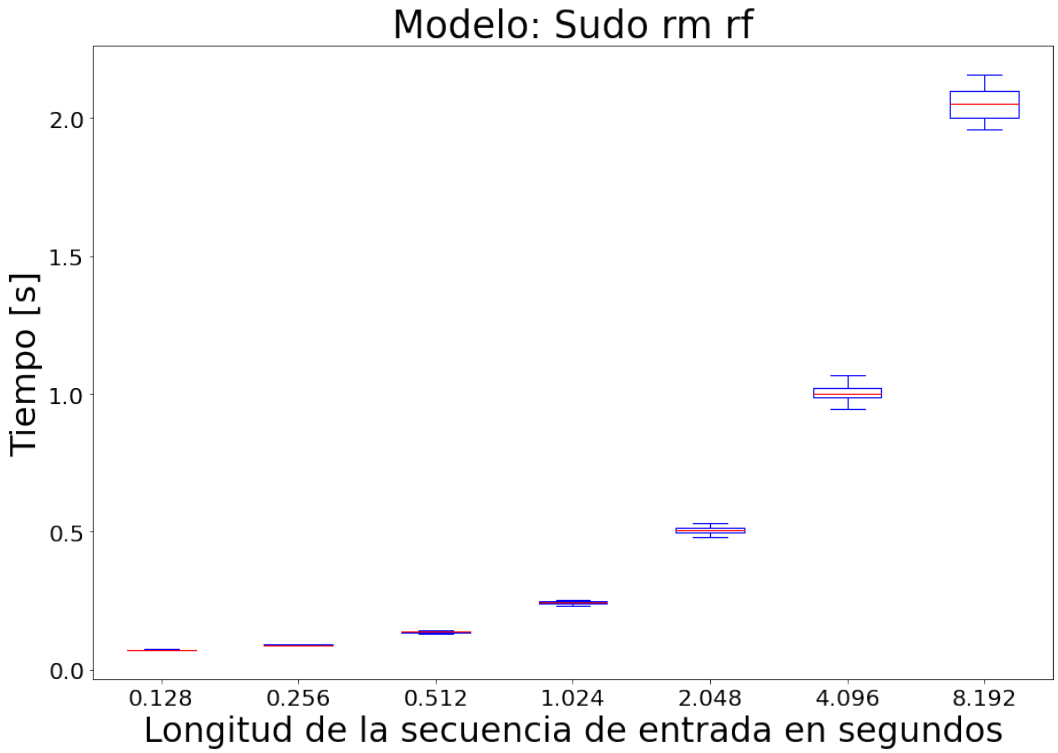


Figura 4.4: Diagrama de cajas y bigotes del tiempo de respuesta con respecto a la variación de la longitud de entrada para el modelo Sudo rm -rf.

Podemos observar que al igual que en la arquitectura ConvTasNet, se presenta una tendencia positiva donde el tiempo de respuesta aumenta conforme la longitud de secuencia de entrada aumenta. De acuerdo con la Figura 4.4, el tiempo de respuesta es menor la duración de la secuencia de entrada en todos los casos, esto nos indica que es factible el uso de esta arquitectura para su uso en aplicaciones en línea.

En la Figura 4.5 se presenta una comparación visual a través de los espectrogramas de las diversas separaciones efectuadas mediante la arquitectura Sudo rm -rf, variando la longitud de la secuencia de entrada.

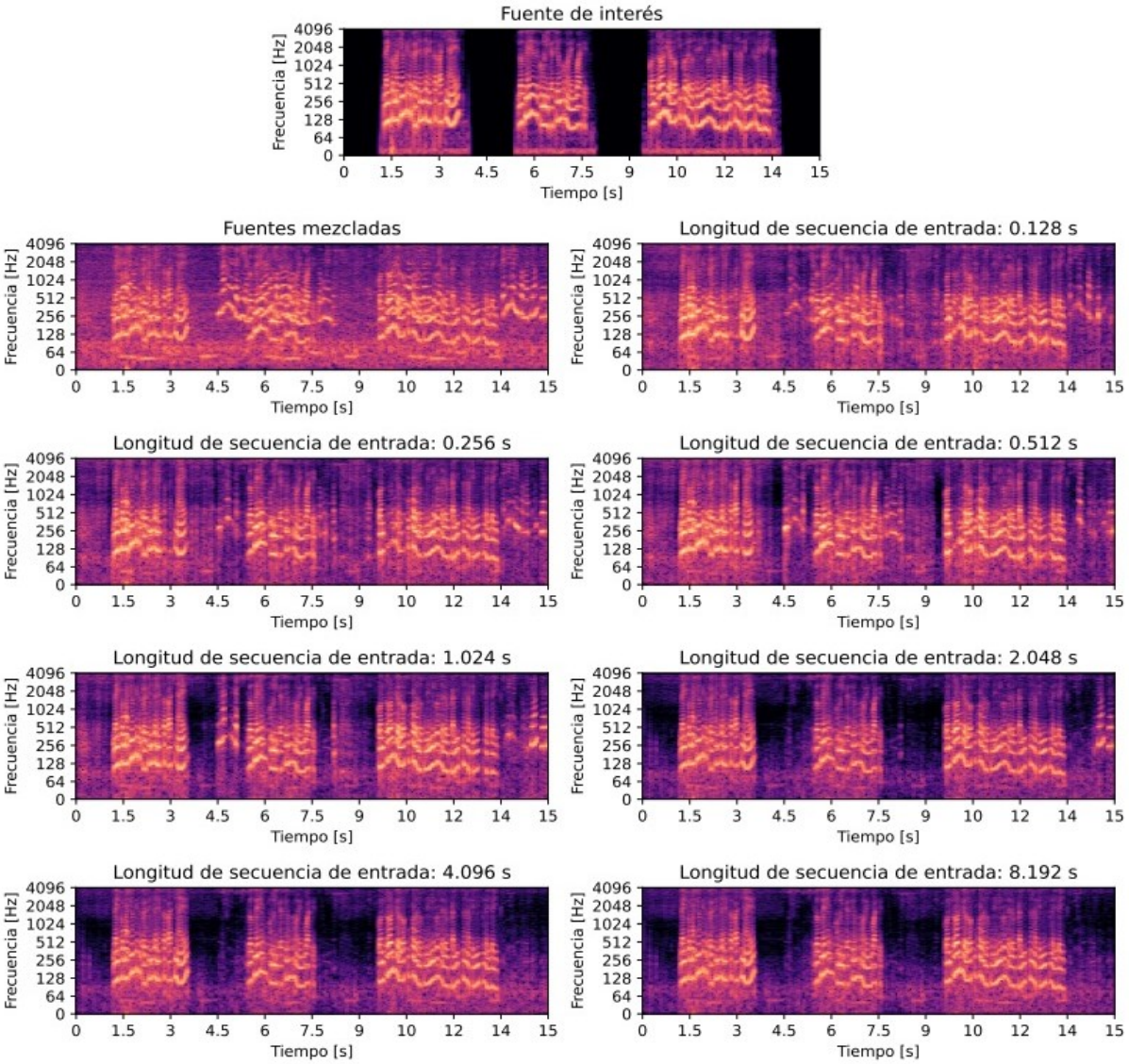


Figura 4.5: Espectrogramas de la separación de fuentes de sonido mediante el uso de la arquitectura Sudo rm -rf. Se presentan los espectrogramas para las diferentes longitudes de secuencia de entrada.

En la Figura anterior se puede observar que a medida que aumenta la longitud de la secuencia de entrada, se observa un incremento en la eficiencia de la separación. Esto se refleja en los espectrogramas de las secuencias más largas, donde se aprecia la ausencia de frecuencias que no pertenecen a la fuente de interés.

La Figura 4.6 ilustra los diagramas de cajas y bigotes de la razón señal-interferencia en función de la longitud de secuencia de entrada en segundos de la arquitectura Sudo rm

-rf.

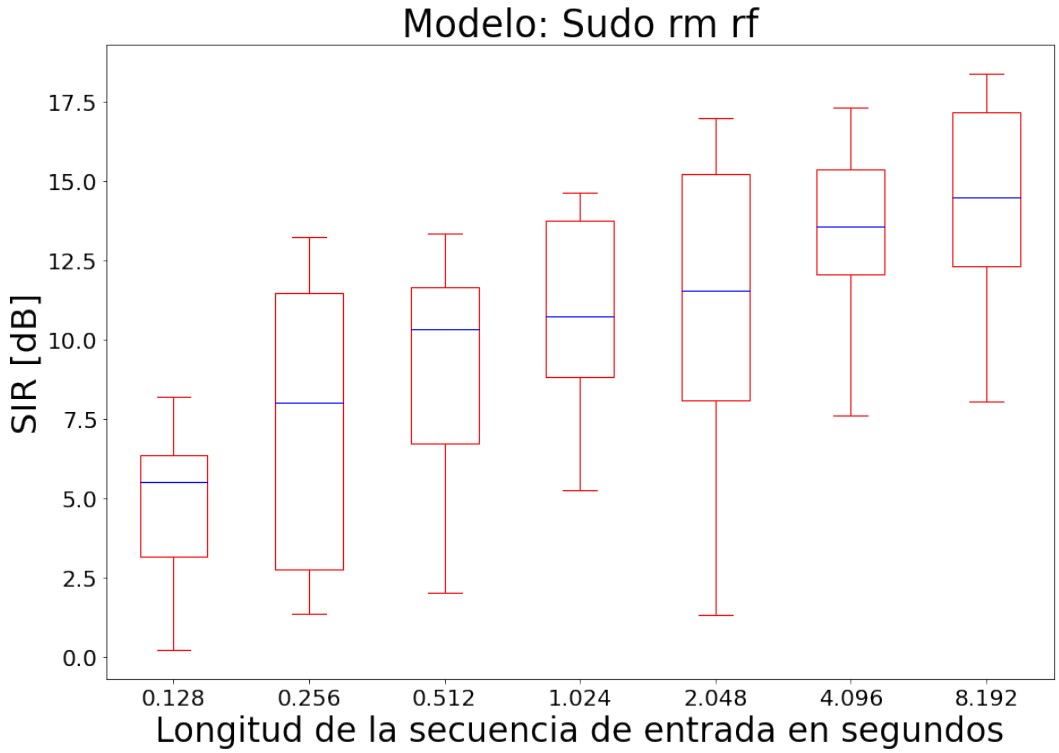


Figura 4.6: Diagrama de cajas y bigotes de la razón señal-interferencia con respecto a la variación de la longitud de entrada para el modelo Sudo rm -rf.

La gráfica anterior muestra una tendencia de aumento en el SIR al incrementar la longitud de secuencia de entrada. A pesar de que la mediana en cada una de las cajas muestra un aumento respecto a la longitud de secuencia de entrada, se puede observar gran variabilidad en los valores del SIR, dicha variabilidad no muestra una tendencia clara. Sin embargo las gráficas respaldan a los espectrogramas mostrados en la figura 4.5, donde se muestra una mejoría en el desempeño del modelo a partir de la duración de secuencia de entrada de 2.048 segundos.

Sepformer:

En el Cuadro 4.3 se muestran los parámetros utilizados durante el entrenamiento de la arquitectura Sepformer.

Modelo	Sepformer
Nintra	2
Ninter	2
Heads	16
DFF	1024
PosEnc	True
optimizer	adam
learning rate	0.00015
epochs	200
batch	1

Cuadro 4.3: Parámetros para la arquitectura Sepformer.

En la Figura 4.7 se presentan los diagramas de cajas y bigotes del tiempo de respuesta en función de la longitud de la secuencia de entrada generados mediante las inferencias con la arquitectura Sepformer.

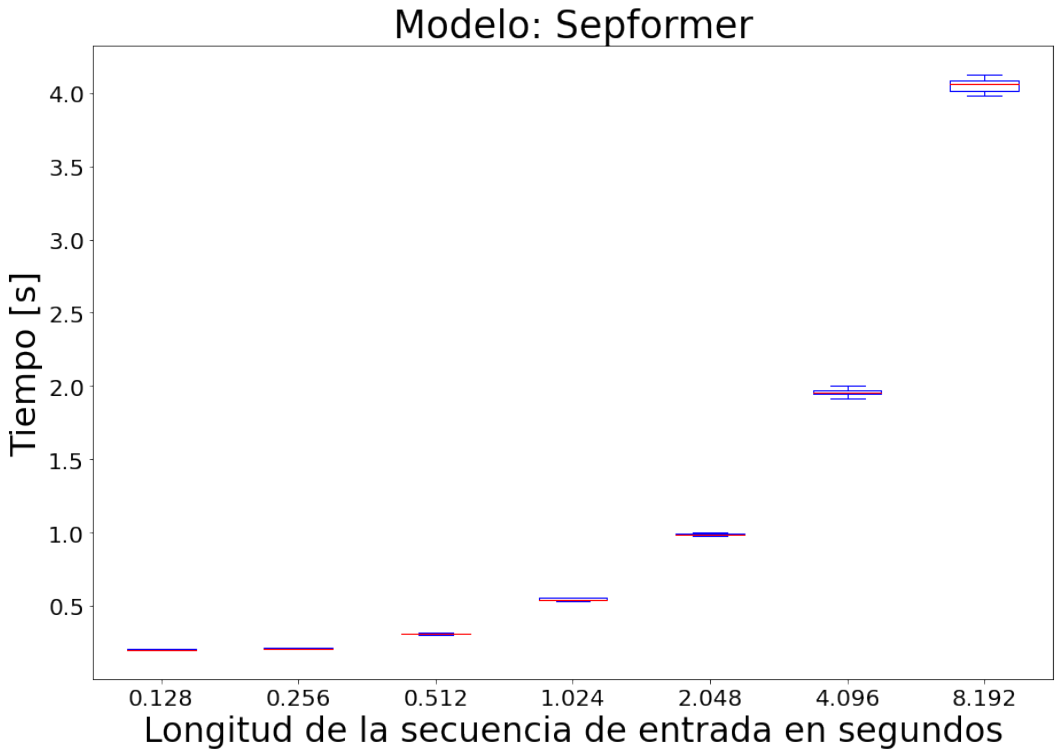


Figura 4.7: Diagrama de cajas y bigotes del tiempo de respuesta con respecto a la variación de la longitud de entrada para el modelo Sepformer.

En la Figura 4.7 se puede observar una tendencia en el aumento del tiempo de respuesta conforme aumenta la longitud de secuencia de entrada. En este caso podemos observar que el tiempo de respuesta es aproximadamente la mitad del tiempo del segmento con el que se alimenta a la arquitectura.

La Figura 4.8 se muestran los espectrogramas generados a partir de las inferencias con la arquitectura Sepformer de las distintas longitudes de secuencia de entrada. Se puedo observar que no hay una diferencia notable en los espectrogramas de entradas de secuencias de tiempo cortas con los espectrogramas de las entradas de secuencias grandes. Para que este transformer lograra trabajar en línea se tuvieron que disminuir las capas de auto-atención del inter e intra transformer a 1 y 3 respectivamente . Es importante destacar que a pesar de que los tiempos de respuesta son menores a los tiempos de los segmentos con los que se hace inferencia en el modelo, este no muestra un desempeño favorable en la separación.

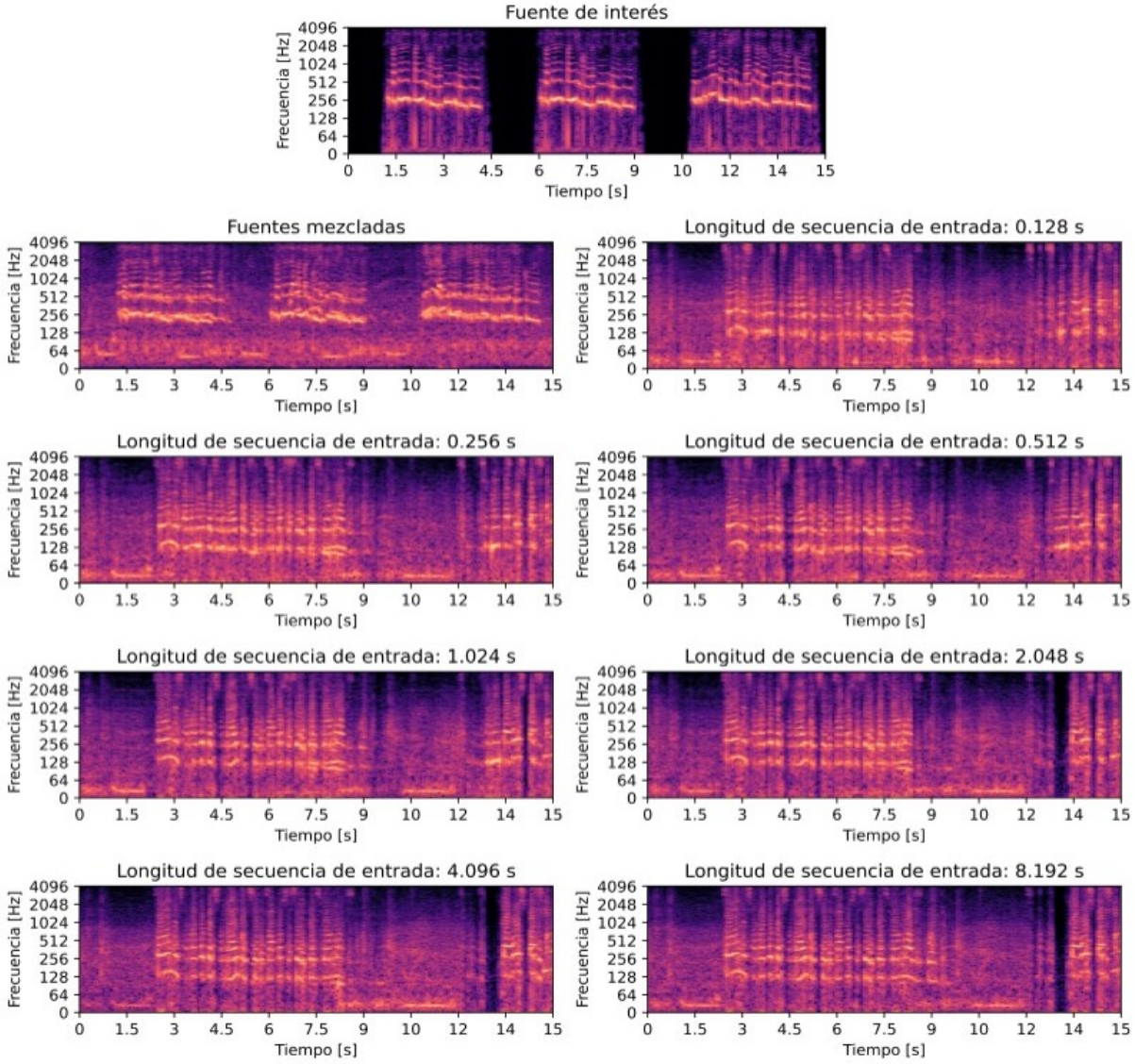


Figura 4.8: Espectrogramas de la separación de fuentes de sonido mediante el uso de la arquitectura Sepformer. Se presentan los espectrogramas para las diferentes longitudes de secuencia de entrada.

Para contextualizar nuestro estudio, consideremos la Figura 4.9 donde se presenta la razón señal-interferencia con respecto a la longitud de la secuencia de entrada para el modelo Sepformer. Para esta arquitectura podemos notar que a pesar de que existe una tendencia en el aumento del SIR conforme aumenta la longitud de secuencia de entrada, el desempeño no supera un SIR de 12 lo que nos indica que no está efectuando un desempeño favorable en la separación. Otro aspecto importante que se debe destacar es la

gran variabilidad que se presenta cuando se hace inferencia con segmentos de 8.192 segundos. Esto nos puede dar indicios de que el modelo entrenado presenta un sobreajuste y la variabilidad se debe a que no se logra generalizar la inferencia para los datos de prueba.

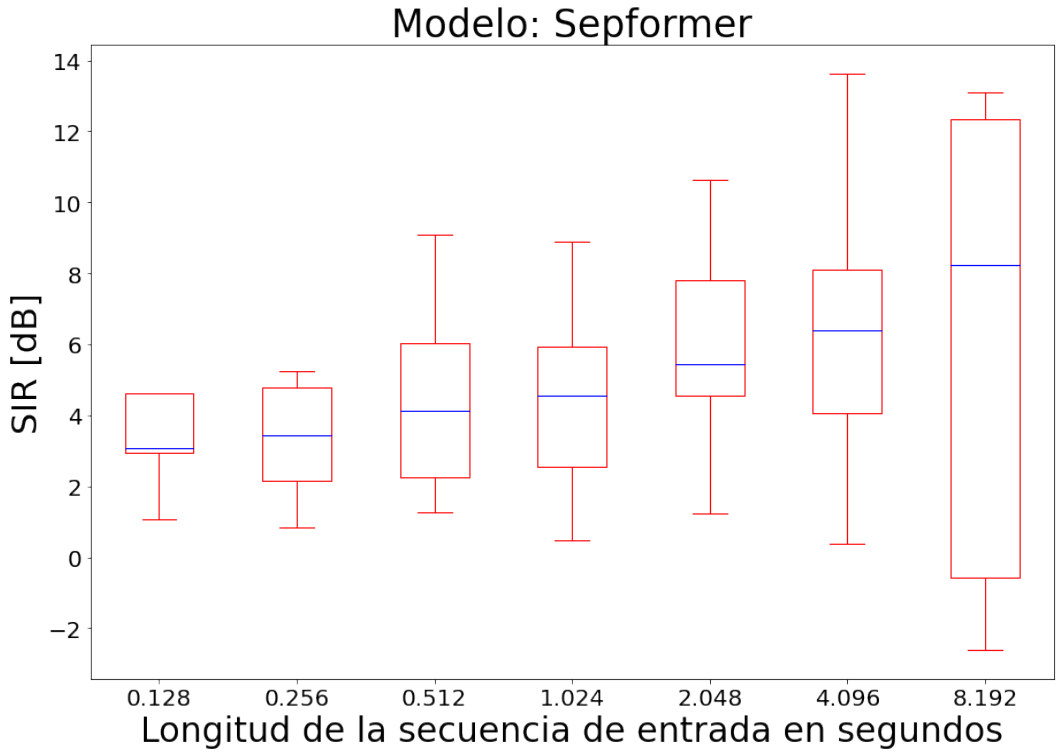


Figura 4.9: Diagrama de cajas y bigotes del tiempo de respuesta con respecto a la variación de la longitud de entrada para el modelo Sepformer.

4.1.1. Evaluación de los modelos de separación del habla para dos y tres hablantes

Los modelos ConvTasNet, Sudo rm -rf y Sepformer fueron entrenados para separar dos fuentes de voz, sin embargo, para mostrar su desempeño en diversos escenarios, en las figuras 4.10, 4.11, 4.12, 4.13, 4.14 y 4.15 se presenta una comparación del desempeño de los modelos en la separación de audios que contienen dos y tres hablantes. En las figuras se presenta el tiempo de repuesta, el factor de tiempo real y la razón señal-interferencia en función de la longitud de secuencia de entrada.

De acuerdo con la Figura 4.10 la arquitectura Sudo rm -rf es la que presenta un menor

tiempo de respuesta en función de la longitud de secuencia de entrada, mientras que Sepformer triplica el tiempo de respuesta de dicha arquitectura. Por otro lado, ConvTasNet muestra un desempeño similar al del modelo Sudo rm -rf pero no lo supera.

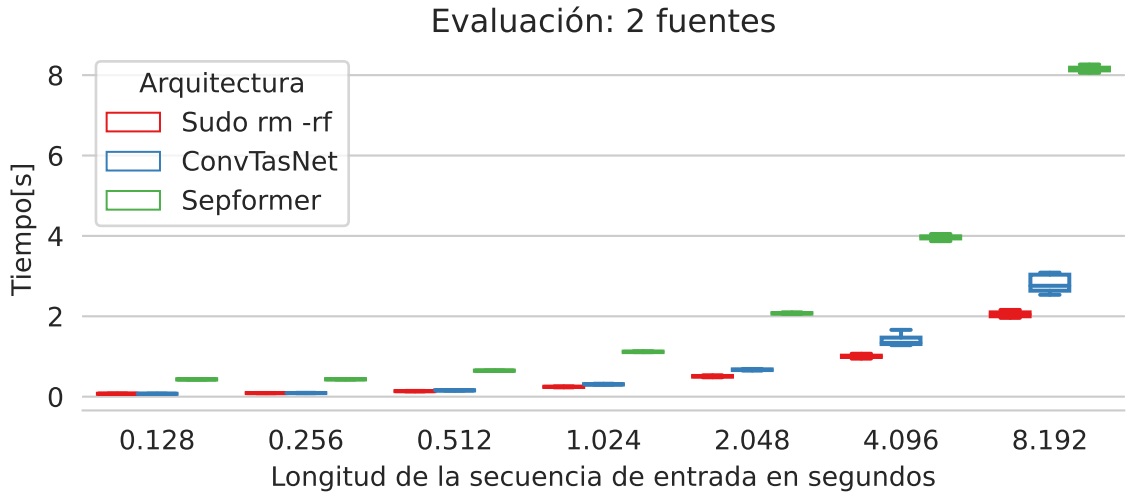


Figura 4.10: Comparación de los tiempos de respuestas de las arquitecturas: Sudo rm -rf, ConvTasNet y Sepformer.

Al realizar las inferencias con audios que contienen la mezcla de tres hablantes, de acuerdo con la Figura 4.11, podemos observar que no se presenta un cambio significativo en los tiempos de respuesta en ninguno de los modelos.

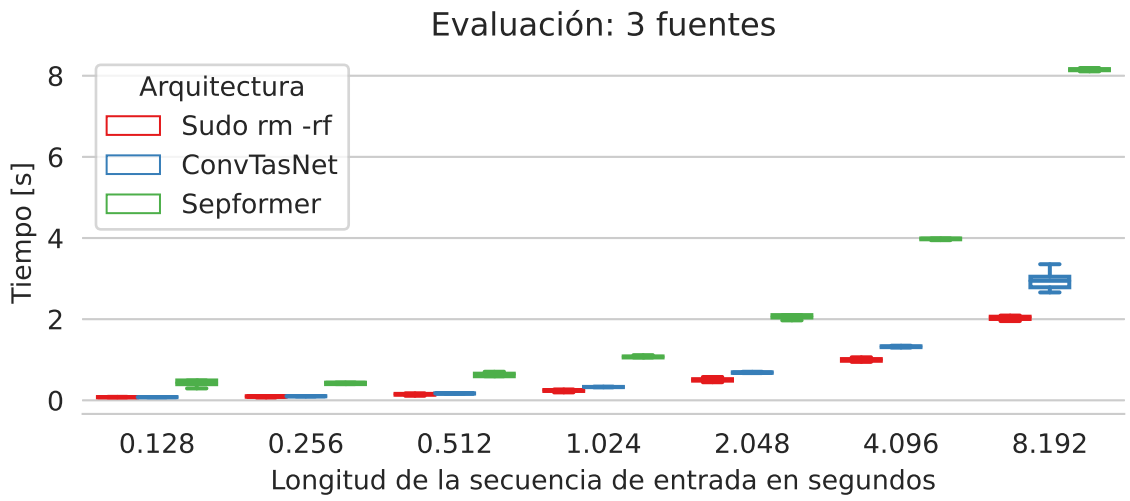


Figura 4.11: Comparación de los tiempos de respuestas de las arquitecturas: Sudo rm -rf, ConvTasNet y Sepformer.

Como se puede apreciar en la Figura 4.12, los modelos Sudo rm -rf y ConvTasNet son aptos para trabajar en aplicaciones en línea debido a que muestran un RTF menor a 1 para todas las longitudes de secuencia de entrada. Por otro lado, la arquitectura Sepformer muestra un $RTF \geq 1$ para todos los casos de secuencia de entrada por lo que se tendría que buscar una estrategia de optimización para ocupar al modelo en aplicaciones en línea.

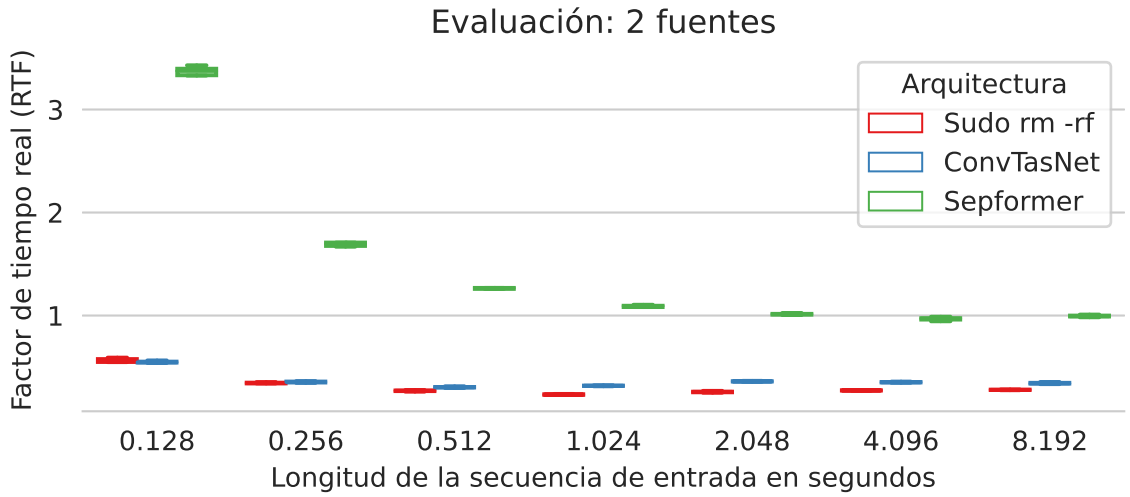


Figura 4.12: Comparación de los tiempos de respuestas de las arquitecturas: Sudo rm -rf, convTasNet y sepformer

Los resultados de RTF para las inferencias con audios que contienen tres hablantes (ver Figura 4.13) no muestran un cambio significativo par alguno de los modelos.

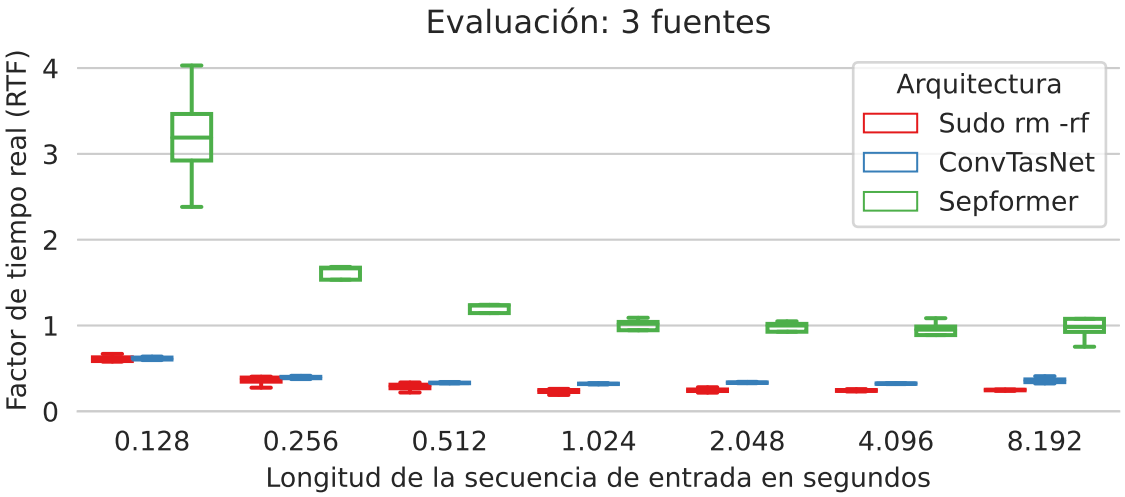


Figura 4.13: Comparación del RTF para las arquitecturas: Sudo rm -rf, convTasNet y sepformer.

En la Figura 4.14 podemos observar la comparación de la razón señal-interferencia para los modelos Sudo rm -rf, ConvTasNet y Sepformer. Se destaca que para el caso de la separación de dos hablantes el modelo que mejor desempeño tiene es ConvTasNet mientras que Sepformer presenta valores de SIR en el rango de -2 a 13, siendo el modelo que peor desempeño tiene.

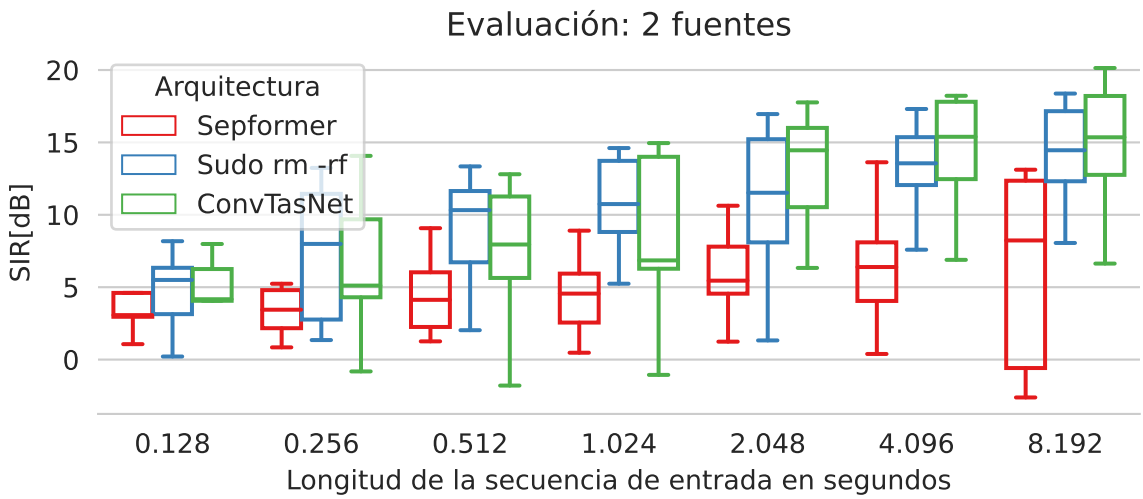


Figura 4.14: Comparación de la razón señal-interferencia de las arquitecturas: Sudo rm -rf, convTasNet y sepformer.

Al realizar inferencia con los tres modelos para audios con tres hablantes (ver Figura 4.15) nos podemos dar cuenta que tanto el modelo ConvTasNet como el modelo Sudo rm -rf disminuyen su desempeño en la separación de las fuentes. Por otro lado, la arquitectura Sepformer muestra un comportamiento similar al mostrado en la inferencia con dos hablantes, este hecho nos siguiere que Sepformer es robusto ante nuevos escenarios auditivos.

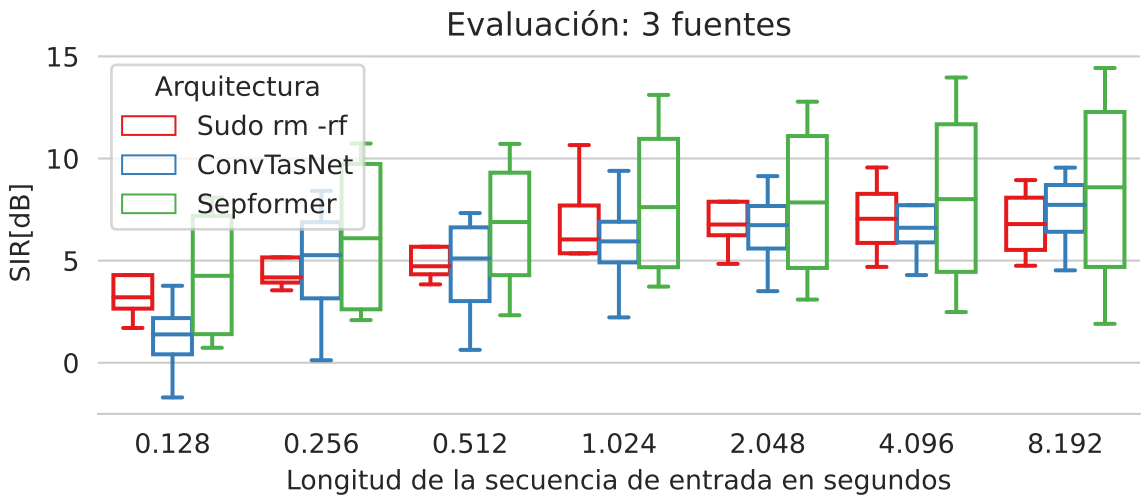


Figura 4.15: Comparación de la razón señal-interferencia de las arquitecturas: Demucs-Denoiser y DCCRN.

4.2. Modelos para mejora del habla

En esta sección se presentan los resultados y discusión referentes a los modelos empleados para la mejora del habla.

Deep Complex Convolution Recurrent Network (DCCRNet):

En el Cuadro 4.4 se presentan los parámetros empleados para el entrenamiento de la arquitectura DCCRNet.

Modelo	DCCRNet
stft_n_filters	512
stft_kernel_size	400
stft_stride	100
optimizer	adam
learning rate	1e-3
epochs	200
batch	12

Cuadro 4.4: Parámetros para la arquitectura DCCRNet.

En la Figura 4.16 se grafican los diagramas de cajas y bigotes obtenidos mediante las inferencias realizadas con el modelo DCCRNet. En dicha figura se está graficando el tiempo de respuesta en función de la longitud de la secuencia de entrada.

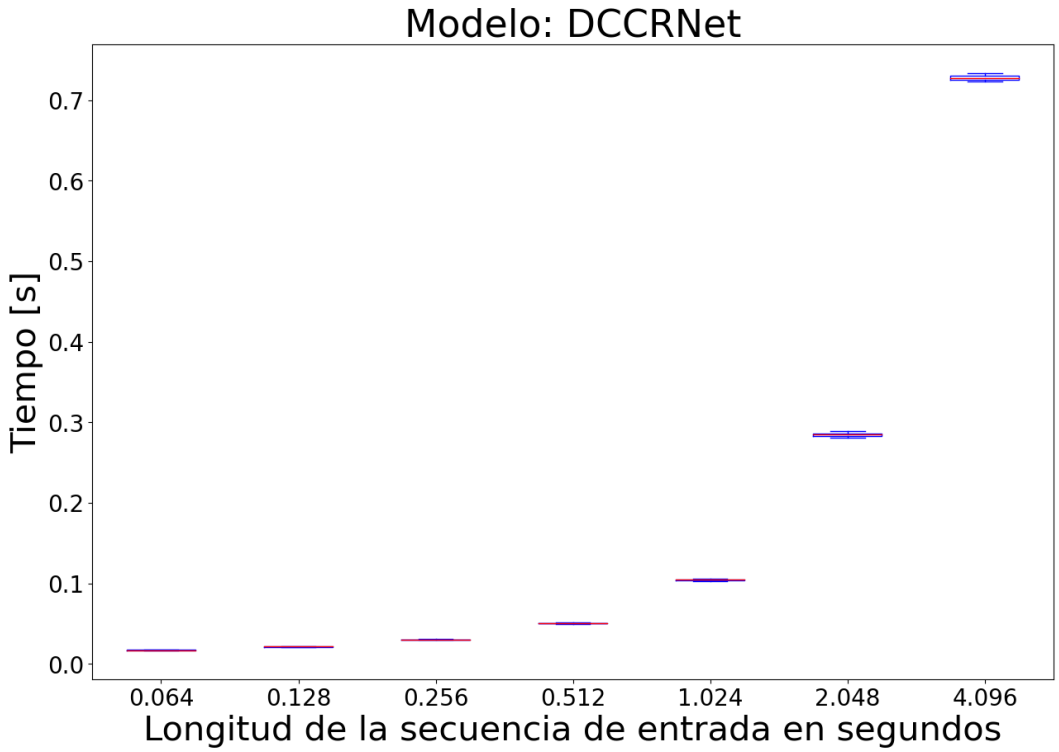


Figura 4.16: Diagrama de cajas y bigotes del tiempo de respuesta con respecto a la variación de la longitud de entrada para el modelo DCCRNet.

La Figura 4.16 muestra un amento en el tiempo de respuesta conforme aumenta la longitud de secuencia de entrada, lo que tiene sentido, ya que entre más datos sean procesados más tiempo le tomara al modelo realizar la inferencia. Podemos ver que el tiempo de respuesta del modelo no supera a la duración del segmento de entrada. Con esto se puede deducir que la arquitectura DCCRNet puede ser utilizada en aplicaciones en línea.

En la Figura 4.17 se presentan los espectrogramas de antes y después de realizar la inferencia con el modelo DCCRNet. Se incluyes las diversas evaluaciones realizadas para las longitudes de secuencia de entrada.

De acuerdo con los espectrogramas mostrados en la Figura, podemos observar una clara mejora en el audio de entrada . Los espectrogramas de los audios que fueron procesados

con 4.096 y 8.192 segundos destacan en su mejoría sobre los de secuencias más cortas.

Como se puede apreciar en la Figura 4.18, el SIR obtenido mediante las inferencias con el modelo DCCRNet aumenta conforme la longitud de secuencia de entrada también aumenta. Se logra observar que los valores del SIR poseen una gran variabilidad para todas las duraciones de entrada, presentándose una mejoría en su valor para secuencias largas.

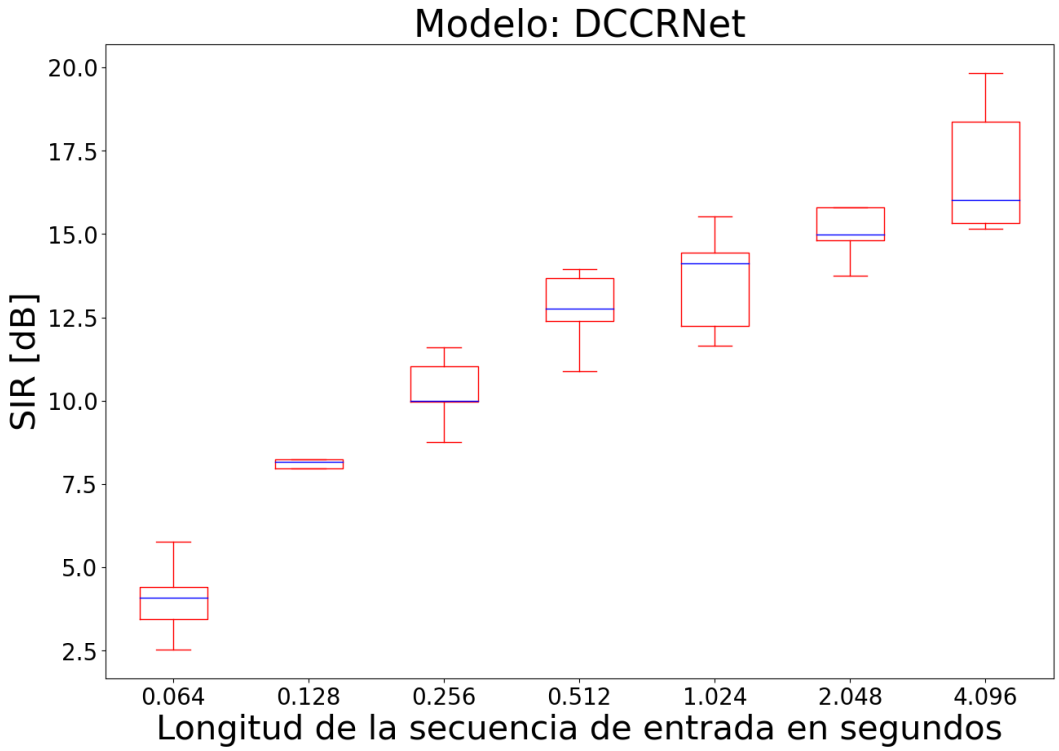


Figura 4.18: Diagrama de cajas y bigotes del tiempo de respuesta con respecto a la variación de la longitud de entrada para el modelo DCCRNet.

Demucs:

En el Cuadro 4.5 se muestran los parámetros utilizados para el entrenamiento con el modelo Demucs.

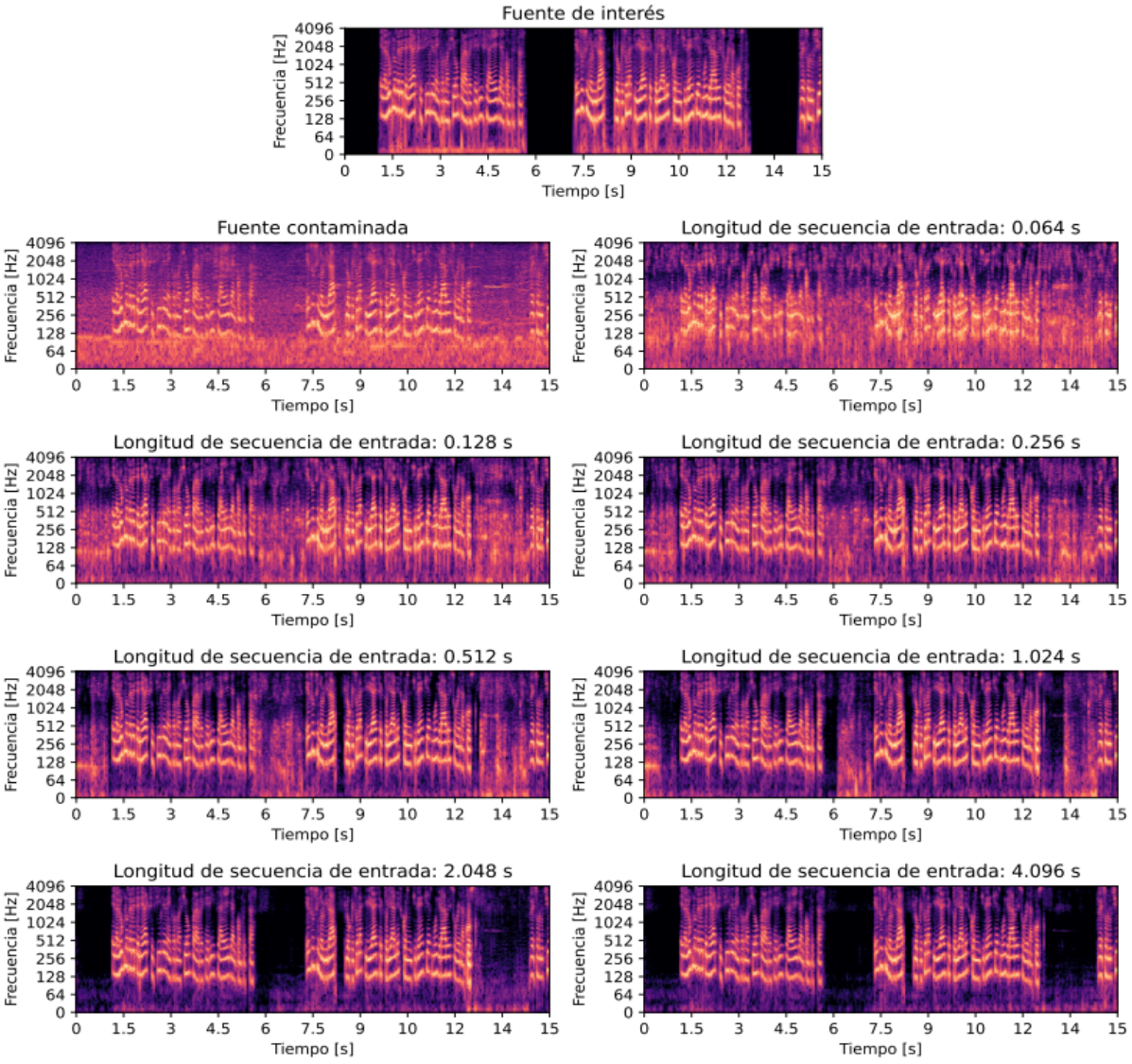


Figura 4.17: Espectrogramas de la separación de fuentes de sonido mediante el uso de la arquitectura DCCRNet. Se presentan los espectrogramas para las diferentes longitudes de secuencia de entrada.

Modelo	Demucs
hid_chan	48
kernel_size	8
stride	4
optimizer	adam
learning rate	3e-4
epochs	500
batch	12

Cuadro 4.5: Parámetros para la arquitectura Demucs.

La Figura 4.19 proporciona una representación visual clara de cómo el tiempo de respuesta del modelo varía en relación con las longitud de la secuencia de entrada. A medida que la longitud aumenta, podemos notar un correspondiente aumento en el tiempo de la inferencia. Es importante destacar que incluso con secuencias de entrada largas, el tiempo de respuesta del modelo siempre permanece por debajo de la longitud de la secuencia. Esto demuestra la eficiencia del modelo para el procesamiento de los datos y nos da indicios de que el modelo es apto para aplicaciones de procesamiento en línea.

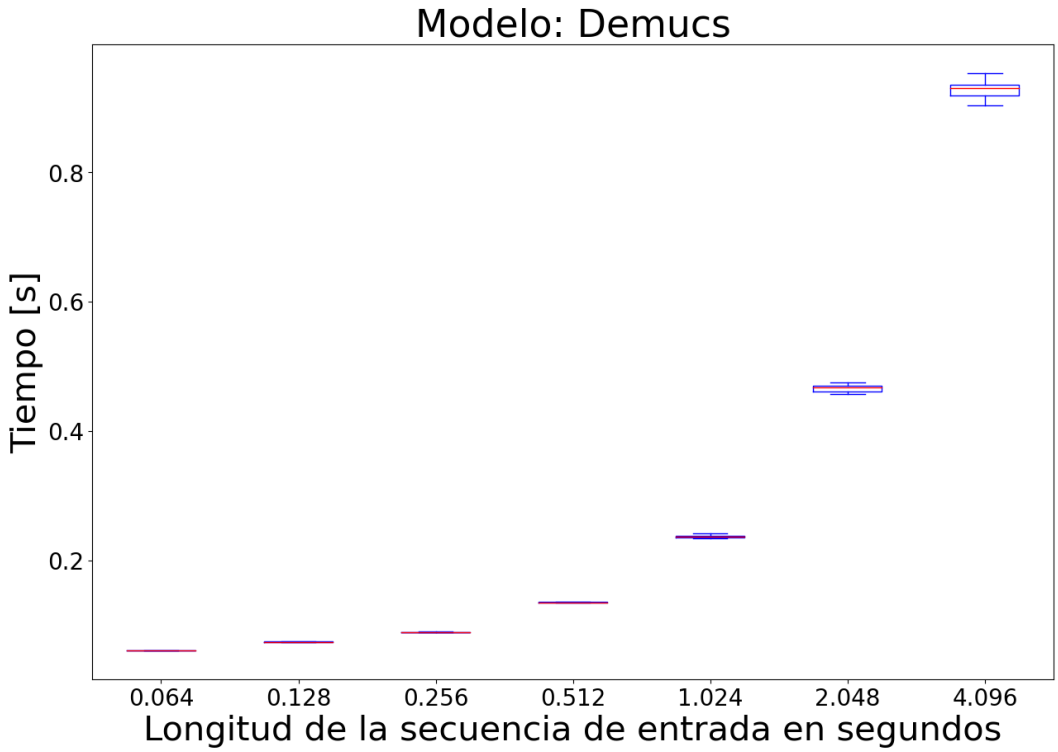


Figura 4.19: Diagrama de cajas y bigotes del tiempo de respuesta con respecto a la variación de la longitud de entrada para el modelo Demucs.

La Figura 4.20 muestra una serie de espectrogramas que ilustran la mejora de audio que ha sido procesado con diferentes longitudes de secuencia de entrada. Es evidente que a medida de que la longitud de secuencia de entrada aumenta los patrones de frecuencia y tiempo en los espectrogramas también presentan variaciones significativas. Podemos observar una reducción de ruido y una mayor claridad en las frecuencias del hablante incluso en secuencias de entrada cortas. Esto sugiere que Demucs es un modelo idóneo para aplicaciones que requieren de una baja latencia.

En la Figura 4.21 se muestran los diagramas de cajas y bigotes de la razón señal-interferencia con respecto a la longitud de la secuencia de entrada. Es importante notar que hay un aumento significativo en el valor del SIR conforme la longitud de secuencia de entrada se hace más grande. Podemos observar que incluso para secuencias cortas, Demucs muestra un buen valor en el SIR. Esto refuerza la idea de que incluso para secuencias cortas el modelo tendrá un buen desempeño de la mejora del habla.

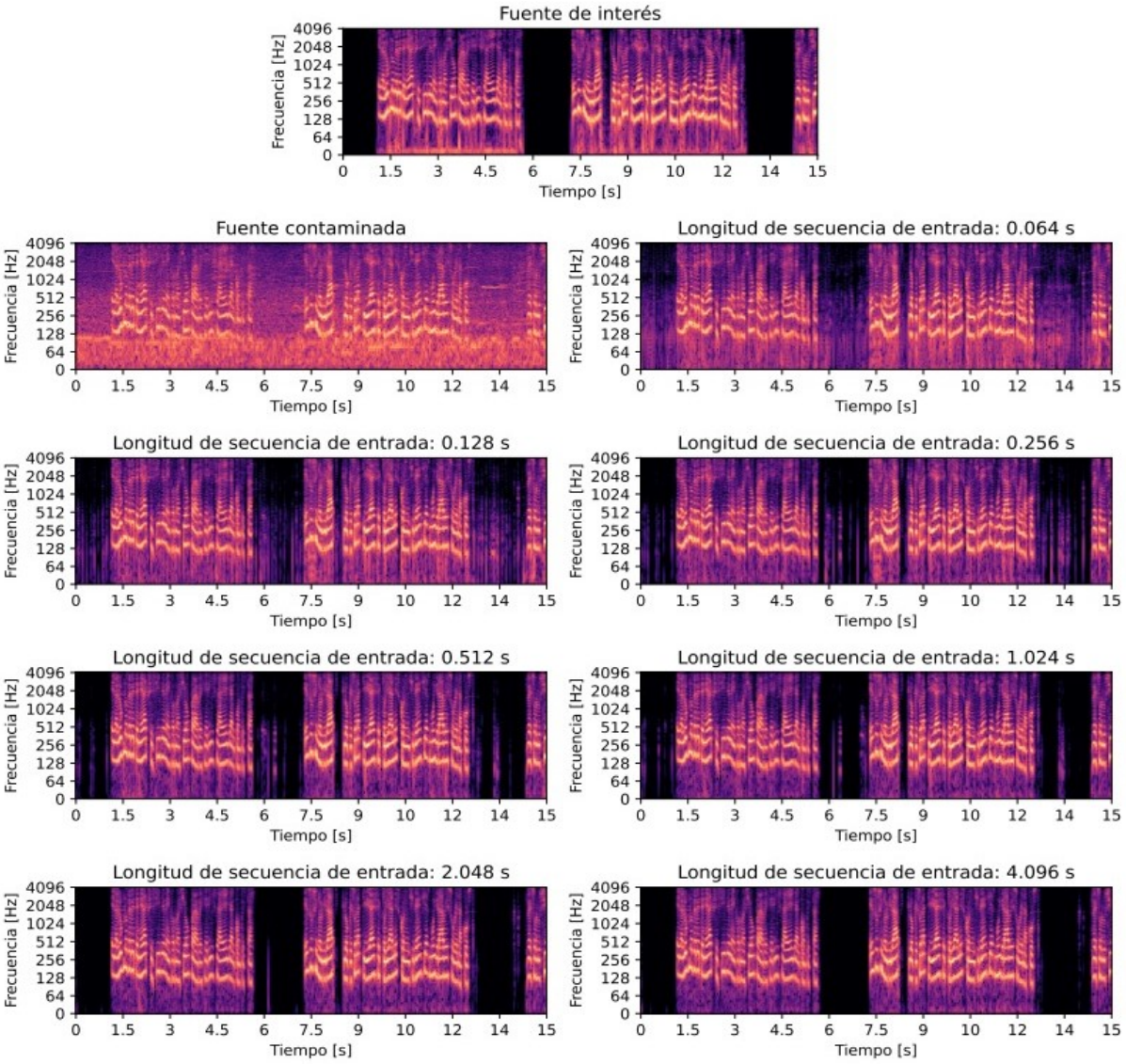


Figura 4.20: Espectrogramas de la separación de fuentes de sonido mediante el uso de la arquitectura Demucs. Se presentan los espectrogramas para las diferentes longitudes de secuencia de entrada.

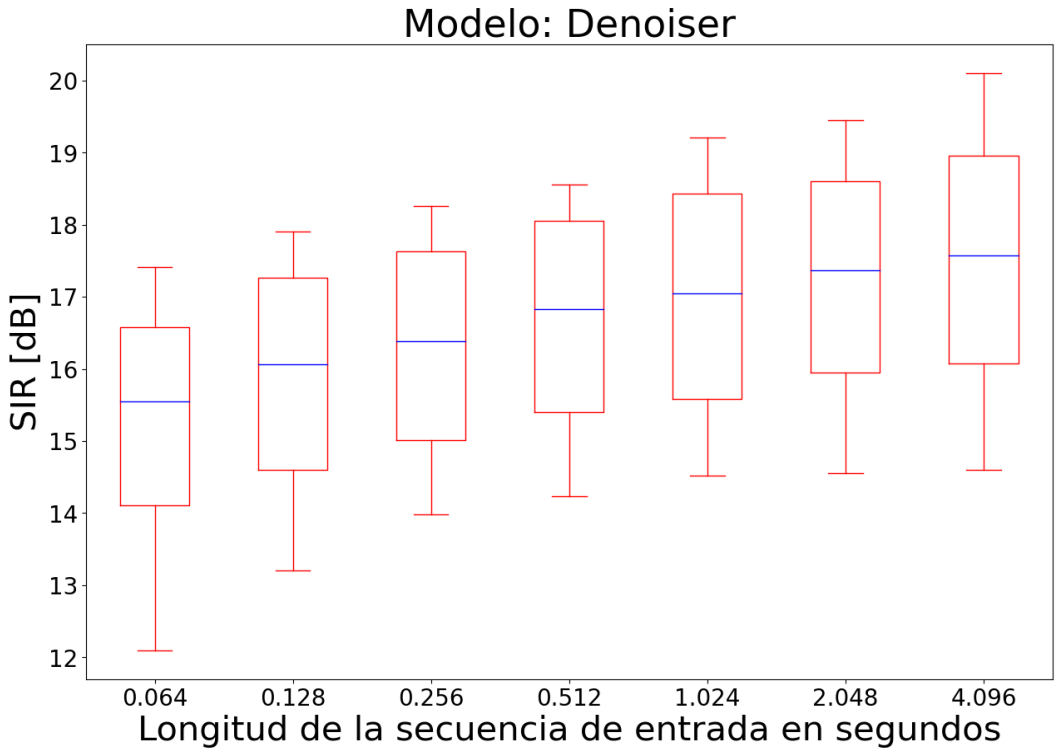


Figura 4.21: Diagrama de cajas y bigotes del tiempo de respuesta con respecto a la variación de la longitud de entrada para el modelo Demucs.

MLT mimic loss:

En la Figura 4.22 se presentan diagramas de cajas y bigotes del tiempo de respuesta en función de la longitud de secuencia de entrada en las inferencias con el modelo MLT mimic loss. Se observa que conforme aumenta la longitud de la secuencia de entrada también aumenta el tiempo de respuesta.

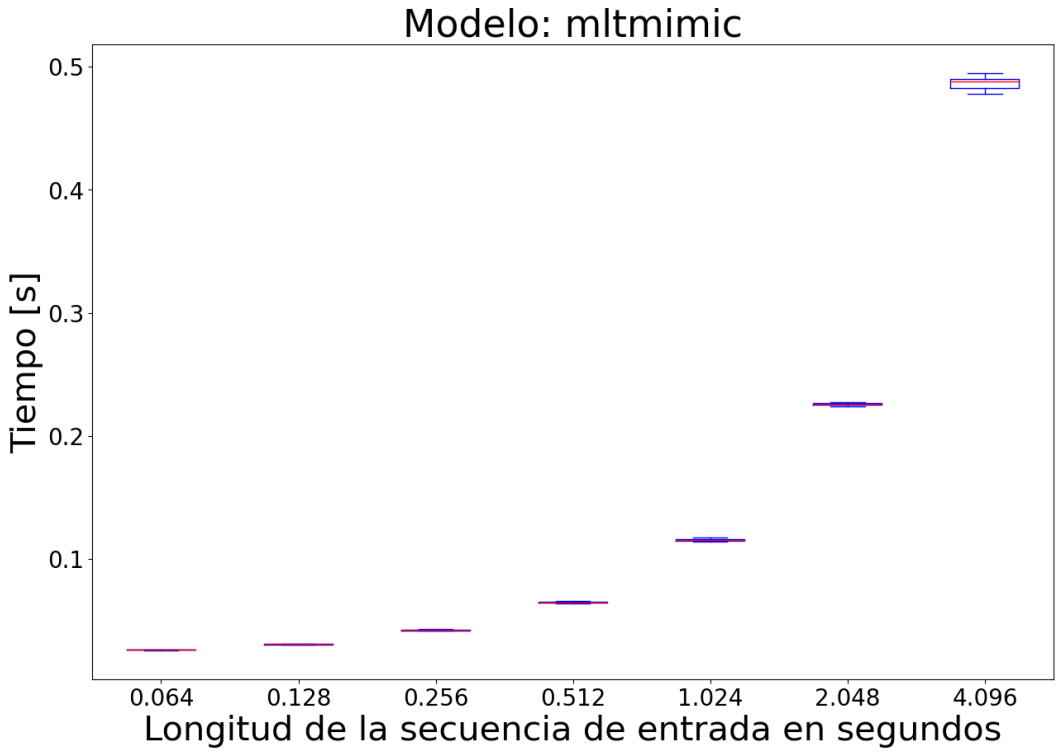


Figura 4.22: Diagrama de cajas y bigotes del tiempo de respuesta con respecto a la variación de la longitud de entrada para el modelo MLT mimic losss.

En la Figura 4.23 se presentan los espectrogramas obtenidos para distintas longitudes de secuencia de entrada mediante la inferencia con el modelo MLT mimic loss. Podemos observar que este modelo presenta una limpieza de frecuencias incluso con secuencias de entrada de 0.064 segundos.

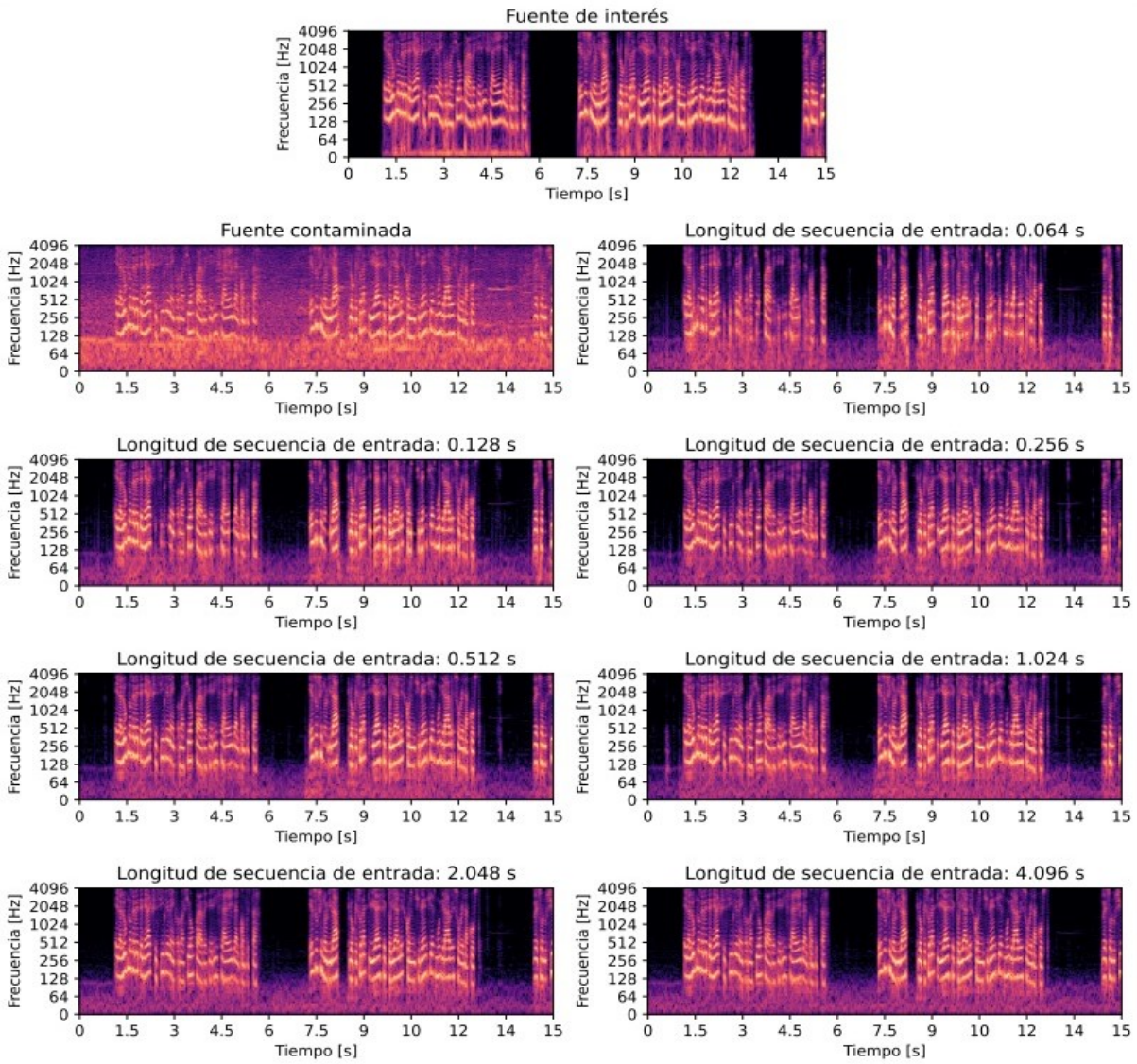


Figura 4.23: Espectrogramas de la separación de fuentes de sonido mediante el uso de la arquitectura MLT mimic losss. Se presentan los espectrogramas para las diferentes longitudes de secuencia de entrada.

Finalmente, en la Figura 4.24 se presenta la razón señal-interferencia en función de la longitud de secuencia de entrada. Se logra observar un aumento en la razón señal-interferencia conforme aumenta la longitud de la secuencia de entrada.

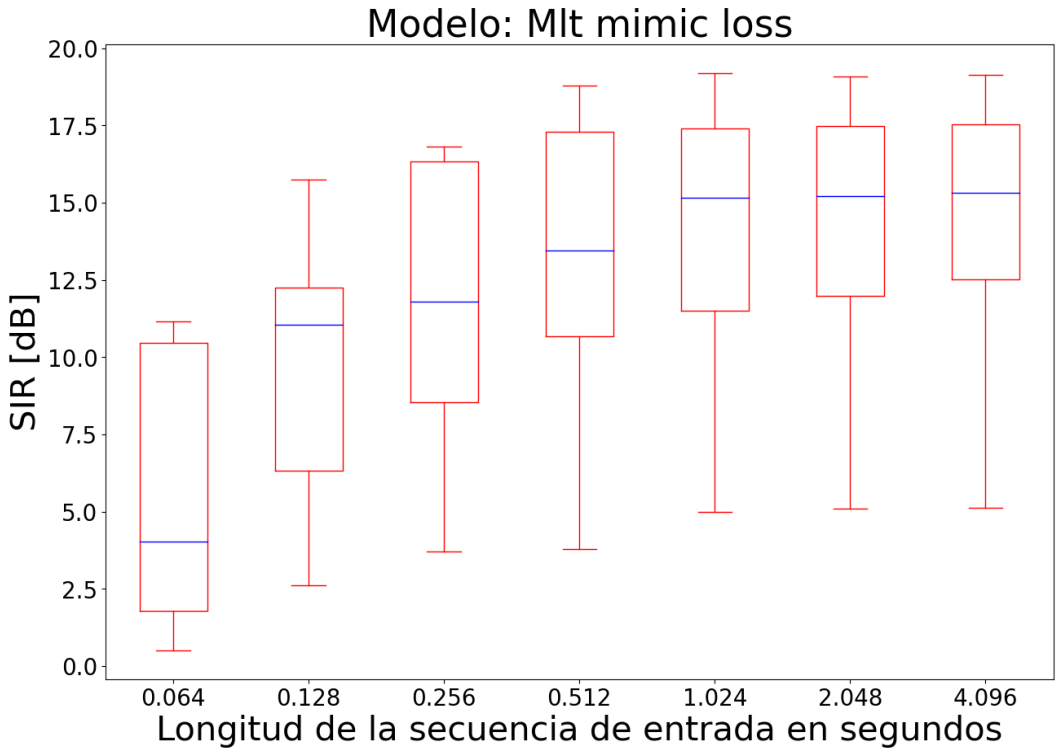


Figura 4.24: Diagrama de cajas y bigotes del tiempo de respuesta con respecto a la variación de la longitud de entrada para el modelo MLT mimic loss.

En la Figura 4.25 se muestra una comparación de los tiempos de respuesta en función de la longitud de secuencia de entrada para todos los modelos de mejora del habla estudiados. Se logra observar como el modelo Mlt-mimic es el que menor tiempo de respuesta presenta en todos los casos, este es seguido por el modelo DCCRNet.

Al observar los diagramas de cajas y bigotes que se presentan en la Figura 4.26, es evidente que el factor de tiempo real disminuye conforme la longitud de la secuencia de entrada aumenta. Las tres técnicas de mejora del habla son aptas para correr en línea ya que en todos los casos se muestra un $RTF \leq 1$, teniendo un mejor rendimiento para longitudes de secuencia de entrada largas. Sin embargo, el hecho de ocupar secuencias de entrada largas se reflejara en una latencia mayor que presentara la aplicación.

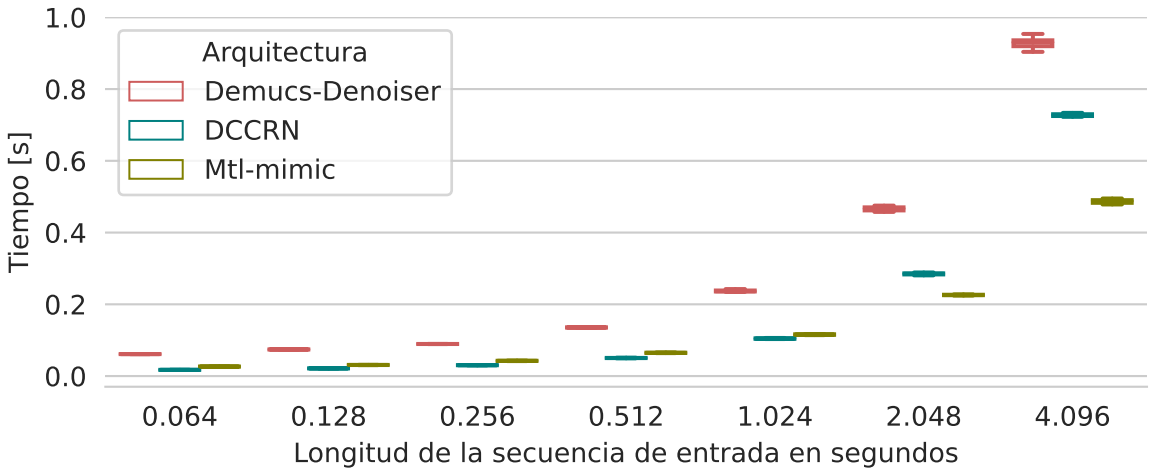


Figura 4.25: Comparación de los tiempos de respuestas de las arquitecturas: Demucs-Denoiser y DCCRN.

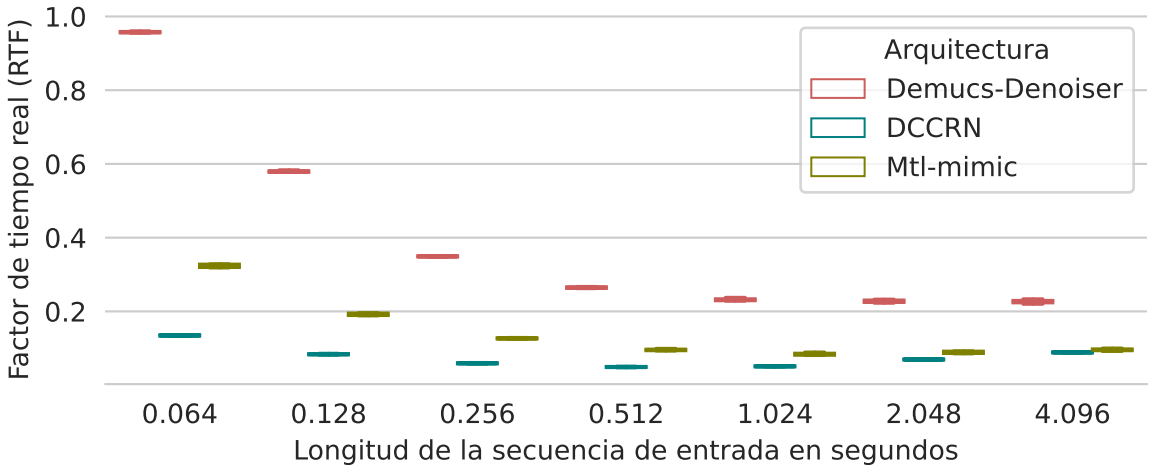


Figura 4.26: Comparación del RTF en las arquitecturas: Demucs-Denoiser y DCCRN.

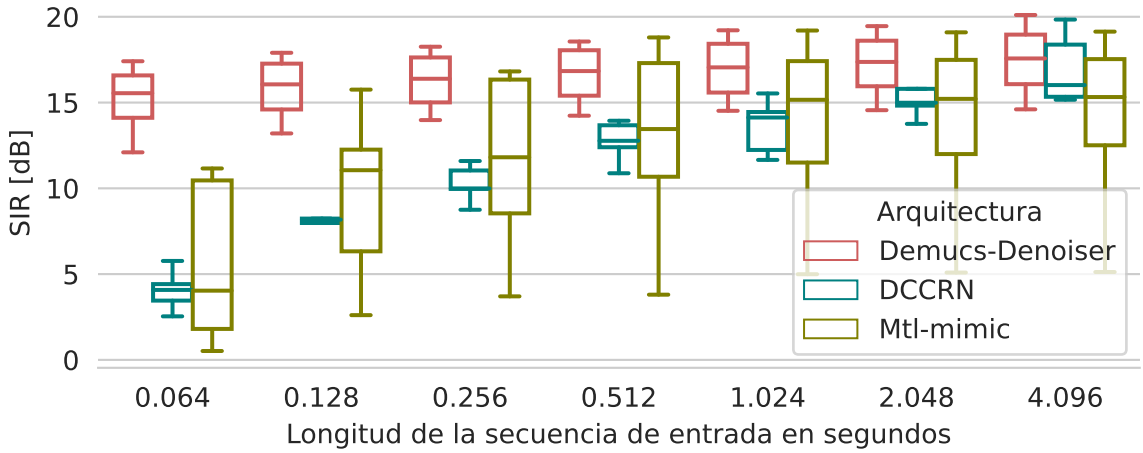


Figura 4.27: Comparación de la razón señal-interferencia de las arquitecturas: Demucs-Denoiser, DCCRN y MLT mimic loss.

En las figuras 4.28 y 4.29 se presentan múltiples diagramas de cajas y bigotes que ilustran como el valor del PESQ y del STOI varían en relación con diferentes longitudes de secuencia de entrada en los modelos Demucs, DCCRNet y Mlt mimic. Podemos observar que se presenta una mayor calidad perceptual del habla para secuencias largas. Se muestra un comportamiento similar para el STOI, sin embargo, cabe resaltar que el modelo Demucs mantiene las características fundamentales de la señal del habla original en la señal mejorada incluso para secuencias de entradas muy cortas, esto se logra deducir al ver que el STOI se mantiene en un valor mayor a 0.9 para todas las secuencias de entrada.

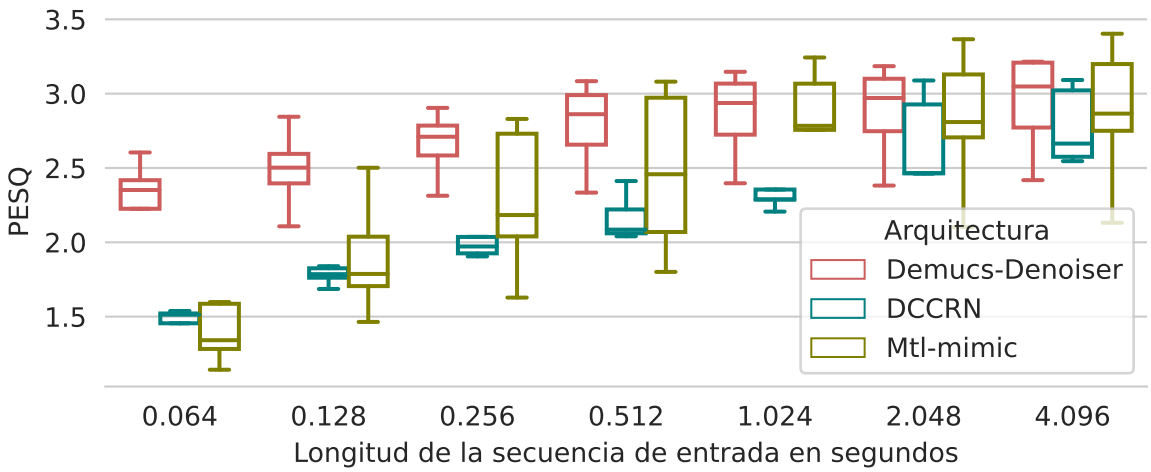


Figura 4.28: Comparación del PESQ en las arquitecturas: Demucs-Denoiser y DCCRN.

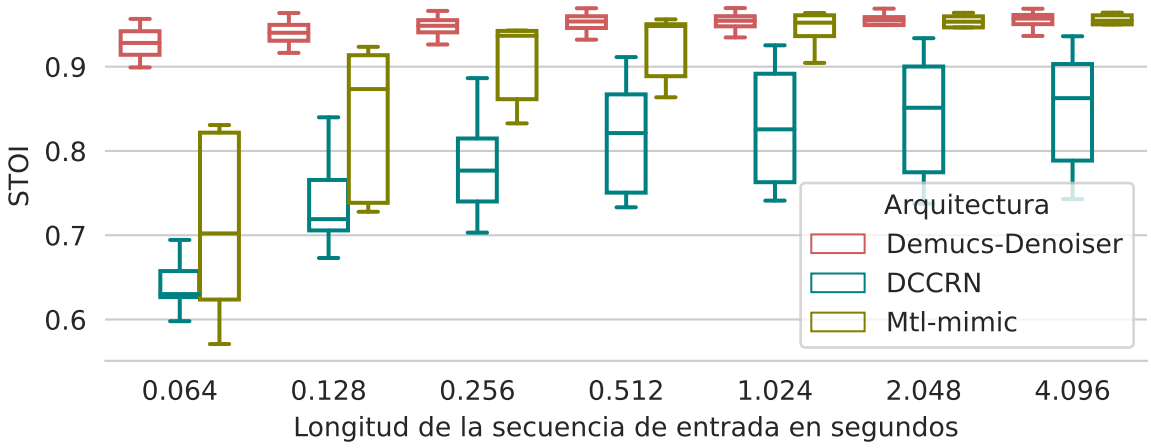


Figura 4.29: Comparación del STOI en las arquitecturas: Demucs-Denoiser y DCCRN.

4.3. Discusión

De acuerdo con los resultados expuestos en las secciones previas, nos podemos dar cuenta de que tanto los métodos de separación del habla como los enfoques de mejora del habla muestran un impacto significativo en la calidad y comprensibilidad de la señal. Al haber trabajado con tres métodos distintos para cada categoría, se destaca la diversidad de enfoques empleados para abordar esta tarea.

En lo que respecta a las arquitecturas ConvTasNet, Sudo rm -rf y Sepformer, podemos observar que su rendimiento para secuencias de entrada cortas (0.128 segundos) no es suficientemente bueno, presentando un SIR menor a 10dB en los tres casos. ConvTasNet y Sudo rm -rf parecen tener una mejora en el SIR alcanzando valores de 15dB para longitudes de secuencia de entrada de 1.024 segundos. Esta mejora también se refleja en los espectrogramas generados durante estas inferencias, donde se aprecia una mayor distinción de las frecuencias que pertenecen a la voz separada.

Algo importante a considerar con respecto a los valores reportados, es la aplicación para la cual los modelos pueden ser empleados, ya que la latencia de audio que puede ser tolerada depende en gran medida del contexto en el cual se esté utilizando el audio.

A pesar del esfuerzo por adaptar la arquitectura Sepformer para lograr su entrenamiento con los recursos computacionales disponibles para este proyecto, los resultados obtenidos en su desempeño en línea no alcanzaron los niveles deseados. Los valores obteniendo de SIR son menores a 13dB en todas las longitudes de secuencia de entrada que se probaron. La evaluación visual de los espectrogramas obtenidos para las distintas inferencias respalda la deficiencia en la separación del modelo con la configuración propuesta en este trabajo. De igual forma, el factor de tiempo real es mayor a uno en todos los casos.

La reducción del número de capas de auto-atención se realizó con la finalidad de disminuir la carga computacional tanto en entrenamiento como durante la inferencia con el modelo. Sin embargo, es posible que esta modificación haya afectado la capacidad del modelo para capturar relaciones complejas en la señal del habla.

Algo importante a destacar de la arquitectura Sepformer, es que es robusta ante el ruido. Esto se puede determinar ya que al hacer inferencia con audios que tienen tres hablantes, su rendimiento no se ve afectado, superando en la evaluación a los modelos ConvTasNet y Sudo rm -rf.

De acuerdo con los resultados obtenidos mediante la evaluación de los modelos para mejora del habla, el desempeño de las arquitecturas DCCRNet y MLT mimic loss resultan ser muy susceptibles a la variación de la longitud de la secuencia de entrada. Para una secuencia de entrada $\geq 0{,}512$ segundos el valor del SIR en las inferencias con MLT mimic loss no sufre cambios significativos por lo que podría recomendarse su uso para estas

longitudes de secuencia de entrada.

El modelo DCCRNet exhibe un rendimiento inferior cuando se le presenta secuencias de entrada más cortas, especialmente aquellas inferiores a 0.256 segundos. Este fenómeno se refleja claramente en los espectrogramas generados, donde se observa que dicho modelo funciona de manera más eficaz para secuencias de entrada más prolongadas, como las de 2.048 y 4.096 segundos.

Por último, el modelo Demucs es el más robusto ante los cambios en la duración de la secuencia de entrada, teniendo el mejor rendimiento de los tres modelos examinados. Al explorar los espectrogramas asociados con las inferencias de Demucs, se aprecia cómo este modelo mantiene una calidad constante a lo largo de las diversas longitudes de secuencia de entrada que se probaron.

5 Conclusiones y trabajo futuro

Se logró la implementación de un sistema diseñado para la evaluación en línea de modelos destinados a la separación y la mejora del habla. El sistema basado en un enfoque de segmentación por ventanas nos permitió explorar el rendimiento de las arquitecturas propuestas en este trabajo.

Los resultados de este estudio enfocado en la evaluación de métodos para la separación y la mejora del habla resaltan la complejidad de esta tarea en aplicaciones en línea. Al explorar el funcionamiento de diversas arquitecturas en un entorno donde hay una secuencia de entrada muy corta, nos percatamos de que la mayoría de los modelos carecen de la capacidad para adaptarse de manera efectiva a estas condiciones.

En lo que respecta a los modelos de separación del habla, si bien todos muestran una capacidad para reducir de cierta forma el ruido y mitigar la presencia de otras fuentes, no todos son viables para trabajar de forma en línea. El factor de tiempo real para todas las longitudes de secuencia de entrada evaluadas con los modelos ConvTasNet y Sudo rm -rf está por debajo de umbral permitido, demostrando un rendimiento adecuado para aplicaciones en línea. Sin embargo, estas arquitecturas presentan un SIR menor a 10dB al trabajar con secuencias de entrada $\leq$ 0,256 segundos.

Por otro lado, la adaptación de la arquitectura Sepformer para su entrenamiento con recursos computacionales limitados no logró alcanzar los niveles deseados de desempeño en línea. A pesar de que se determinó que el modelo es robusto ante el ruido en la evaluación con audios que contenían tres hablantes, los valores de SIR fueron consistentemente menores a 13 dB en todas las longitudes de secuencia de entrada, y el factor de tiempo real superó el umbral permitido en todos los casos.

En cuanto a los modelos de mejora del habla, las arquitecturas DCCRNet y MLT mimic loss muestran una alta sensibilidad a la variación en la longitud de la secuencia de entrada. Por otro lado, el modelo Demucs se destaca como el más robusto ante los cambios en la duración de la secuencia de entrada, presentando un SIR $\geq$ 10dB incluso para la secuencia

de entrada más corta de 0.064 segundos.

Esta investigación proporciona una base sólida para un estudio posterior en el ámbito de la separación y mejora del habla para comunicación en línea, dando pie al uso de este tipo de tecnologías para aplicaciones como: transmisión de audio en línea (aplicaciones de videoconferencias), sistemas de ayuda auditiva y asistentes virtuales.

Una posible dirección para investigaciones futuras puede ser la incorporación de técnicas de filtrado espacial con el uso de múltiples micrófonos. Explorar si la combinación de información proveniente de un arreglo de micrófonos puede potenciar el desempeño de los modelos, de tal forma que sean capaces de funcionar en entornos más desafiantes. Se tiene la hipótesis de que al combinar filtrado espacial con los modelos entrenados, no solo se puede mejorar la precisión de la separación, sino también reducir la complejidad de los modelos, lo que implica menos parámetros de entrenamiento, y por ende, menos recursos computacionales.

El incremento de la investigación en el campo de la inteligencia artificial y el procesamiento de señales trae consigo el desarrollo de múltiples modelos para la separación y la mejora del habla. La exploración de nuevas arquitecturas es otro factor clave para esta investigación, ya que esto nos permitirá determinar cuál es el enfoque más apropiado para aplicaciones de comunicación en línea.

Bibliografía

[1] Chanwoo Kim, Anjali Menon, Michiel Bacchiani, and Richard M Stern. Sound source separation ssing phase difference and reliable mask selection. *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2018.

[2] Alejandro Maldonado, Caleb Rascon, and Ivette Velez. Lightweight online separation of the sound source of interest through blstm-based binary masking. *Computación y Sistemas*, 24(3):1257–1270, 2020.

[3] Efthymios Tzinis, Zhepei Wang, and Paris Smaragdis. Sudo rm-rf: Efficient networks for universal audio source separation. In *2020 IEEE 30th International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6. IEEE, 2020.

[4] Adelbert W Bronkhorst. The cocktail party phenomenon: A review of research on speech intelligibility in multiple-talker conditions. *Acta Acustica united with Acustica*, 86(1):117–128, 2000.

[5] E Colin Cherry. Some experiments on the recognition of speech, with one and with two ears. *The Journal of the Acoustical Society of America*, 25(5):975–979, 1953.

[6] Albert S Bregman. *Auditory scene analysis: The perceptual organization of sound*. MIT press, 1994.

[7] Josh H McDermott. The cocktail party problem. *Current Biology*, 19(22):R1024–R1027, 2009.

[8] Paul Stookey. Norman normal. https://looneytunes.fandom.com/wiki/Norman_Normal, 1968. Warner Bros.-Seven Arts Animation.

[9] Mads G Christensen. *Introduction to audio processing*. Springer, 2019.

[10] Branko Kovačević, Milan M Milosavljevic, Mladen Veinovic, and Milan Marković. *Robust digital processing of speech signals*. Springer, 2017.

[11] WS Brown Jr, Richard J Morris, Harry Hollien, and Elizabeth Howell. Speaking fundamental frequency characteristics as a function of age and professional singing. *Journal of Voice*, 5(4):310–315, 1991.

[12] David N Sorenson. A fundamental frequency investigation of children ages 6–10 years old. *Journal of Communication Disorders*, 22(2):115–123, 1989.

[13] S Palani. *Principles of Digital Signal Processing*. Springer Nature, 2022.

[14] K Deergha Rao and Madisetti NS Swamy. *Digital signal processing: Theory and practice*. Springer, 2018.

[15] Zhizheng Wu, Oliver Watts, and Simon King. Merlin: An Open Source Neural Network Speech Synthesis System. In *Proc. 9th ISCA Workshop on Speech Synthesis Workshop (SSW 9)*, pages 202–207, 2016.

[16] Wouter Gevaert, Georgi Tsenov, and Valeri Mladenov. Neural networks used for speech recognition. *Journal of Automatic Control*, 20(1):1–7, 2010.

[17] DeLiang Wang and Jitong Chen. Supervised speech separation based on deep learning: An overview. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 26(10):1702–1726, 2018.

[18] DeLiang Wang. Time-frequency masking for speech separation and its potential for hearing aid design. *Trends in Amplification*, 12(4):332–353, 2008.

[19] Alan V. Oppenheim and Alan S. Willsky. *Signals and Systems*. Prentice Hall Signal Processing Series, Upper Saddle Riv er, New Jersey 07458, 1997.

[20] Matteo Frigo. A fast fourier transform compiler. In *Proceedings of the ACM SIGPLAN 1999 Conference on Programming Language Design and Implementation*, pages 169–180, 1999.

[21] Meinard Müller. *Fundamentals of music processing: Audio, analysis, algorithms, applications*, volume 5. Springer, 2015.

[22] Pete Warden. Speech commands: A dataset for limited-vocabulary speech recognition. *arXiv preprint arXiv:1804.03209*, 2018.

[23] SA Raki, Shoji Makino, Hiroshi Sawada, and Ryo Mukai. Reducing musical noise by a fine-shift overlap-add method applied to source separation using a time-frequency mask. In *Proceedings.(ICASSP'05). IEEE International Conference on Acoustics, Speech, and Signal Processing, 2005.*, volume 3, pages iii–81. IEEE, 2005.

[24] Shoji Makino, Te-Won Lee, and Hiroshi Sawada. *Blind speech separation*, volume 615. Springer, 2007.

[25] Ganesh R Naik and Dinesh K Kumar. An overview of independent component analysis and its applications. *Informatica*, 35(1), 2011.

[26] Yang Yu, Wenwu Wang, and Peng Han. Localization based stereo speech source separation using probabilistic time-frequency masking and deep neural networks. *EURASIP Journal on Audio, Speech, and Music Processing*, 2016:1–18, 2016.

[27] Vaninirappuputhenpurayil Gopalan Reju, Soo Ngee Koh, and Yann Soon. Underdetermined convolutive blind source separation via time–frequency masking. *IEEE Transactions on Audio, Speech, and Language Processing*, 18(1):101–116, 2009.

[28] Derry Fitzgerald and Rajesh Jaiswal. On the use of masking filters in sound source separation. In *Proc. of 15th International Conference on Digital Audio Effects, (DAFX12)*, 2012.

[29] Jonathan Le Roux, Emmanuel Vincent, Yuu Mizuno, Hirokazu Kameoka, Nobutaka Ono, and Shigeki Sagayama. Consistent Wiener filtering: Generalized time-frequency masking respecting spectrogram consistency. In *Latent Variable Analysis and Signal Separation: 9th International Conference, LVA/ICA 2010, St. Malo, France, September 27-30, 2010. Proceedings 9*, pages 89–96. Springer, 2010.

[30] Julien Bourgeois and Wolfgang Minker. *Time-domain beamforming and blind source separation: speech input in the car environment*. Springer, 2009.

[31] Jacob Benesty, Shoji Makino, and Jingdong Chen. *Speech enhancement*. Springer Science & Business Media, 2006.

[32] D. Purves. *Neurociencia*. Editorial Médica Panamericana S.A., 2015.

[33] David L Felten and Anil N Shetty. *Netter Atlas de neurociencia*. Elsevier España, 2010.

[34] NIH. ¿Cuáles son las partes del sistema nervioso? https://espanol.nichd.nih.gov/salud/temas/neuro/informacion/partes, 2019. Fecha de acceso: 4 de agosto de 2023.

[35] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, 5:115–133, 1943.

[36] Martin T Hagan, Howard B Demuth, and Mark Beale. *Neural network design*. PWS Publishing Co., 1997.

[37] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[38] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[39] Fernando Berzal. *Redes neuronales & deep learning: Volumen II*. Independently published, 2019.

[40] Michael A Nielsen. *Neural networks and deep learning*, volume 25. Determination press San Francisco, CA, USA, 2015.

[41] Kevin P Murphy. *Probabilistic machine learning: an introduction*. MIT press, 2022.

[42] IBM. What are recurrent neural networks? https://www.ibm.com/topics/recurrent-neural-networks. Fecha de acceso: 4 de agosto de 2023.

[43] Recurrent neural networks cheatsheet. https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks. Fecha de acceso: 2 de agosto de 2023.

[44] Miguel Sotaquirá. ¿Qué son las redes LSTM? https://www.codificandobits.com/blog/redes-lstm/. Fecha de acceso: 6 de agosto de 2023.

[45] Christopher Olah. Understanding LSTM networks. https://colah.github.io/posts/2015-08-Understanding-LSTMs/. Fecha de acceso: 6 de agosto de 2023.

[46] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[47] Charu C Aggarwal et al. Neural networks and deep learning. *Springer*, 10(978):3, 2018.

[48] Cem Subakan, Mirco Ravanelli, Samuele Cornell, Mirko Bronzi, and Jianyuan Zhong. Attention is all you need in speech separation. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 21–25. IEEE, 2021.

[49] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, 30, 2017.

[50] Peter Bloem. Transformers from scratch. https://peterbloem.nl/blog/transformers, 2019.

[51] Thilo Von Neumann, Keisuke Kinoshita, Marc Delcroix, Shoko Araki, Tomohiro Nakatani, and Reinhold Haeb-Umbach. All-neural online source separation, counting, and diarization for meeting analysis. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 91–95. IEEE, 2019.

[52] Yi Luo, Cong Han, Nima Mesgarani, Enea Ceolini, and Shih-Chii Liu. Fasnet: Low-latency adaptive beamforming for multi-microphone audio processing. In *2019 IEEE automatic speech recognition and understanding workshop (ASRU)*, pages 260–267. IEEE, 2019.

[53] Chenda Li, Lei Yang, Weiqin Wang, and Yanmin Qian. Skim: Skipping memory LSTM for low-latency real-time continuous speech separation. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 681–685. IEEE, 2022.

[54] Yi Luo and Nima Mesgarani. Tasnet: time-domain audio separation network for real-time, single-channel speech separation. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 696–700. IEEE, 2018.

[55] Yi Luo and Nima Mesgarani. Conv-tasnet: Surpassing ideal time–frequency magnitude masking for speech separation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 27(8):1256–1266, 2019.

[56] Cem Subakan, Mirco Ravanelli, Samuele Cornell, Francois Grondin, and Mirko Bronzi. On using transformers for speech-separation. *arXiv preprint arXiv:2202.02884*, 2022.

[57] Ziqiang Shi, Rujie Liu, and Jiqing Han. Speech separation based on multi-stage elaborated dual-path deep bilstm with auxiliary identity loss. *arXiv preprint arXiv:2008.03149*, 2020.

[58] Jingjing Chen, Qirong Mao, and Dong Liu. Dual-path transformer network: Direct context-aware modeling for end-to-end monaural speech separation. *arXiv preprint arXiv:2007.13975*, 2020.

[59] Yi Luo, Zhuo Chen, John R Hershey, Jonathan Le Roux, and Nima Mesgarani. Deep clustering and conventional networks for music separation: Stronger together. In *2017 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 61–65. IEEE, 2017.

[60] Alexandre Défossez, Nicolas Usunier, Léon Bottou, and Francis Bach. Demucs: Deep extractor for music sources with extra unlabeled data remixed. *arXiv preprint arXiv:1909.01174*, 2019.

[61] Yanxin Hu, Yun Liu, Shubo Lv, Mengtao Xing, Shimin Zhang, Yihui Fu, Jian Wu, Bihong Zhang, and Lei Xie. Dccrn: Deep complex convolution recurrent network for phase-aware speech enhancement. *arXiv preprint arXiv:2008.00264*, 2020.

[62] Deblin Bagchi, Peter Plantinga, Adam Stiff, and Eric Fosler-Lussier. Spectral feature mapping with mimic loss for robust speech recognition. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5609–5613. IEEE, 2018.

[63] Ariel Ephrat, Inbar Mosseri, Oran Lang, Tali Dekel, Kevin Wilson, Avinatan Hassidim, William T Freeman, and Michael Rubinstein. Looking to listen at the cocktail party: A speaker-independent audio-visual model for speech separation. *arXiv preprint arXiv:1804.03619*, 2018.

[64] Inc Smule. Damp-vsep: Smule digital archive of mobile performances-vocal separation, 2019.

[65] Scott Wisdom, Hakan Erdogan, Daniel PW Ellis, Romain Serizel, Nicolas Turpault, Eduardo Fonseca, Justin Salamon, Prem Seetharaman, and John R Hershey. What's all the fuss about free universal sound separation data? In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 186–190. IEEE, 2021.

[66] Sunit Sivasankaran, Emmanuel Vincent, and Dominique Fohr. Analyzing the impact of speaker localization errors on speech separation for automatic speech recognition. In *2020 28th European Signal Processing Conference (EUSIPCO)*, pages 346–350. IEEE, 2021.

[67] Meta AI. Librimix. https://paperswithcode.com/dataset/librimix. Fecha de acceso: 12 de agosto de 2023.

[68] Zafar Rafii, Antoine Liutkus, Fabian-Robert Stöter, Stylianos Ioannis Mimilakis, and Rachel Bittner. Musdb18-a corpus for music separation. 2017.

[69] Lukas Drude, Jens Heitkaemper, Christoph Boeddeker, and Reinhold Haeb-Umbach. Sms-wsj: Database, performance measures, and baseline recipe for multi-channel source separation and recognition. *arXiv preprint arXiv:1910.13934*, 2019.

[70] Gordon Wichern, Joe Antognini, Michael Flynn, Licheng Richard Zhu, Emmett McQuinn, Dwight Crow, Ethan Manilow, and Jonathan Le Roux. Wham!: Extending speech separation to noisy environments. *arXiv preprint arXiv:1907.01160*, 2019.

[71] Matthew Maciejewski, Gordon Wichern, Emmett McQuinn, and Jonathan Le Roux. Whamr!: Noisy and reverberant single-channel speech separation. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 696–700. IEEE, 2020.

[72] Chang-Bin Jeon, Hyeongi Moon, Keunwoo Choi, Ben Sangbae Chon, and Kyogu Lee. Medleyvox: An evaluation dataset for multiple singing voices separation. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE, 2023.

[73] Zhuo Chen, Takuya Yoshioka, Liang Lu, Tianyan Zhou, Zhong Meng, Yi Luo, Jian Wu, Xiong Xiao, and Jinyu Li. Continuous speech separation: Dataset and analysis. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7284–7288. IEEE, 2020.

[74] Joris Cosentino, Manuel Pariente, Samuele Cornell, Antoine Deleforge, and Emmanuel Vincent. Librimix: An open-source dataset for generalizable speech separation, 2020.

[75] Manuel Pariente, Samuele Cornell, Joris Cosentino, Sunit Sivasankaran, Efthymios Tzinis, Jens Heitkaemper, Michel Olvera, Fabian-Robert Stöter, Mathieu Hu, Juan M. Martín-Doñas, David Ditter, Ariel Frank, Antoine Deleforge, and Emmanuel Vincent. Asteroid: the PyTorch-based audio source separation toolkit for researchers. In *Proc. Interspeech*, 2020.

[76] Mirco Ravanelli, Titouan Parcollet, Peter Plantinga, Aku Rouhe, Samuele Cornell, Loren Lugosch, Cem Subakan, Nauman Dawalatabad, Abdelwahab Heba, Jianyuan Zhong, and et al. Ju-Chieh Chou. SpeechBrain: A general-purpose speech toolkit, 2021. arXiv:2106.04624.

[77] Dong Yu, Morten Kolbæk, Zheng-Hua Tan, and Jesper Jensen. Permutation invariant training of deep models for speaker-independent multi-talker speech separation. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 241–245. IEEE, 2017.

[78] Paul Davis. Jack connecting a world of audio. https://jackaudio.org/. Fecha de acceso: 18 de septiembre de 2023.

[79] Oscar Ruiz-Espitia, Jose Martinez-Carranza, and Caleb Rascon. Aira-uas: an evaluation corpus for audio processing in unmanned aerial system. In *2018 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 836–845. IEEE, 2018.

[80] Emmanuel Vincent, Rémi Gribonval, and Cédric Févotte. Performance measurement in blind audio source separation. *IEEE Transactions on Audio, Speech, and Language Processing*, 14(4):1462–1469, 2006.

[81] Josh O'Farrell. What is: PESQ? https://www.spearline.com/blog/what-is-pesq/. Fecha de acceso: 16 de agosto de 2023.

[82] Hongfeng Li, Yanyan Xu, Dengfeng Ke, and Kaile Su. Improving speech enhancement by focusing on smaller values using relative loss. *IET Signal Processing*, 14(6):374–384, 2020.

[83] Colin Raffel, Brian McFee, Eric J Humphrey, Justin Salamon, Oriol Nieto, Dawen Liang, Daniel PW Ellis, and C Colin Raffel. Mir_eval: A transparent implementation of common mir metrics. In *ISMIR*, volume 10, page 2014, 2014.

[84] Rafael G. Dantas Miao Wang, Christoph Boeddeker and Ananda Seelan. Pesq (perceptual evaluation of speech quality) wrapper for python users. https://zenodo.org/record/6549559. Fecha de acceso: 16 de agosto de 2023.

[85] M Pariente. Python implementation of STOI. https://github.com/mpariente/pystoi.

A

Apéndice A

A.1. Modelos entrenados

En está sección se muestran los parámetros utilizados en los modelos entrenados para este trabajo.

Modelo	bn_chan	n_blocks	kernel	hid	# parámetros	Tiempo	SIR
1	128	8	40	512	5.1 M	4.5669	16.9214
2	128	8	32	512	5.1 M	4.7887	14.1601
3	128	8	16	512	5.1 M	3.8490	17.2385
4	128	7	40	256	4.7 M	3.6996	14.3737
5	256	6	32	256	3.3 M	1.8318	13.5374
6	128	8	40	256	2.7 M	1.8122	13.8689
7	256	6	40	128	1.8 M	1.1315	11.1291
8	128	4	40	256	1.5 M	1.0074	8.5827
9	128	8	16	128	1.4 M	1.1084	10.3013
10	128	4	16	128	1.2M	0.9921	8.2562

Cuadro A.1: Arquitecturas ConvTasNet entrenadas.

Modelo	N	Nintra	Ninter	# Heads	DFF	Tiempo	SIR
1	2	4	1	8	1024	11.0009	16.8045
2	2	3	1	16	1024	10.6542	13.7564
3	2	3	1	8	2048	9.4939	3.7979
4	2	1	1	16	2048	5.9538	3.6374
5	2	1	2	16	2048	6.4213	3.1962
6	3	1	1	8	2048	6.7182	3.1356
7	2	1	2	8	2048	6.6873	9.1291
8	2	2	2	16	2048	6.0674	12.14061
9	2	3	3	8	2048	6.8819	9.1425
10	2	3	2	8	2048	6.6954	9.0212

Cuadro A.2: Arquitecturas Sepformer entrenadas.

Modelo	K	C	B	Q	# parámetros	Tiempo	SIR
1	21	128	16	4	1.3 M	1.7762	4.1428
2	17	128	16	4	1.3 M	2.1954	7.6730
3	17	256	16	4	2.3 M	2.5143	9.8604
4	21	256	20	4	2.9 M	2.4976	7.3466
5	41	256	32	4	4.7 M	2.3506	8.4233
6	41	256	20	4	4.6 M	2.3232	7.0037
7	21	512	18	2	3 M	3.7261	10.0037
8	21	512	20	2	3.2 M	3.9054	9.4853
9	21	512	34	4	5.3 M	3.7261	10.9542
10	21	512	16	4	2.6 M	2.9382	13.7412

Cuadro A.3: Arquitecturas Sudo rm -rf entrenadas durante 200 épocas.

Apéndice B

B.1. Ajuste lineal para los tiempos de respuesta

A continuación, se realiza un análisis utilizando un ajuste lineal para examinar los tiempos de respuesta promedio de los modelos que corresponden a las ventanas más grandes. Este análisis se centra en las ventanas de mayor tamaño debido a que, en el contexto de la notación asintótica, se busca comprender la complejidad del algoritmo a medida que el tamaño del problema tiende hacia infinito.

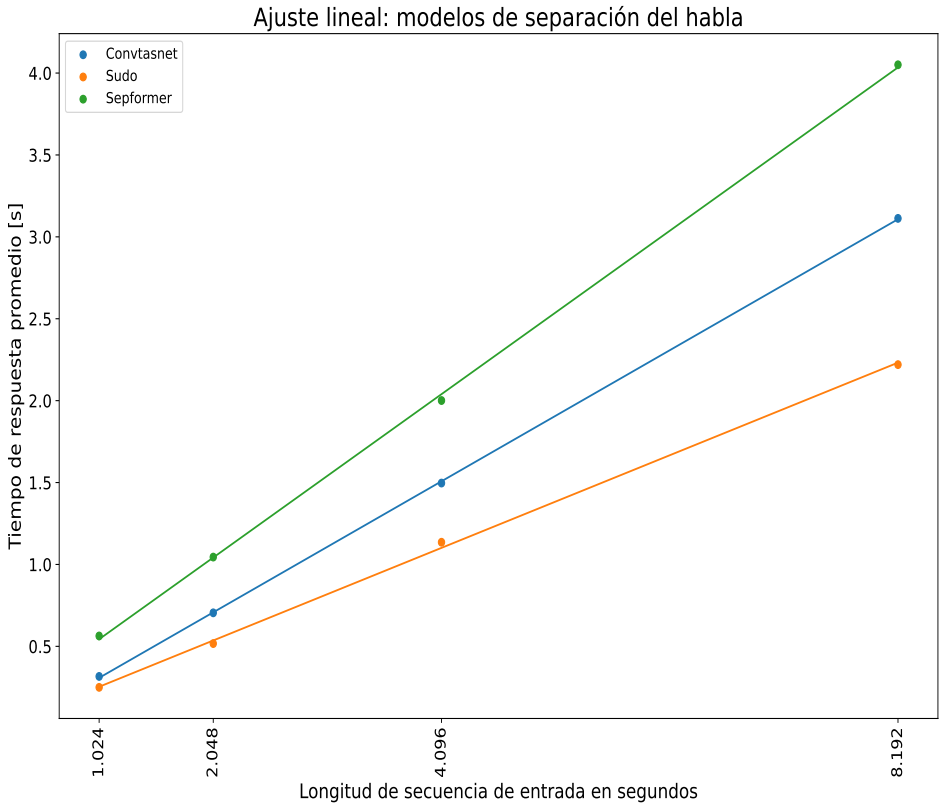


Figura B.1: Ajuste lineal del tiempo de respuesta promedio en relación con la longitud de la secuencia de entrada para los modelos de separación del habla.

La Figura B.1 sugiere que los tiempos de respuesta de los tres modelos de separación del habla examinados exhiben un comportamiento lineal.

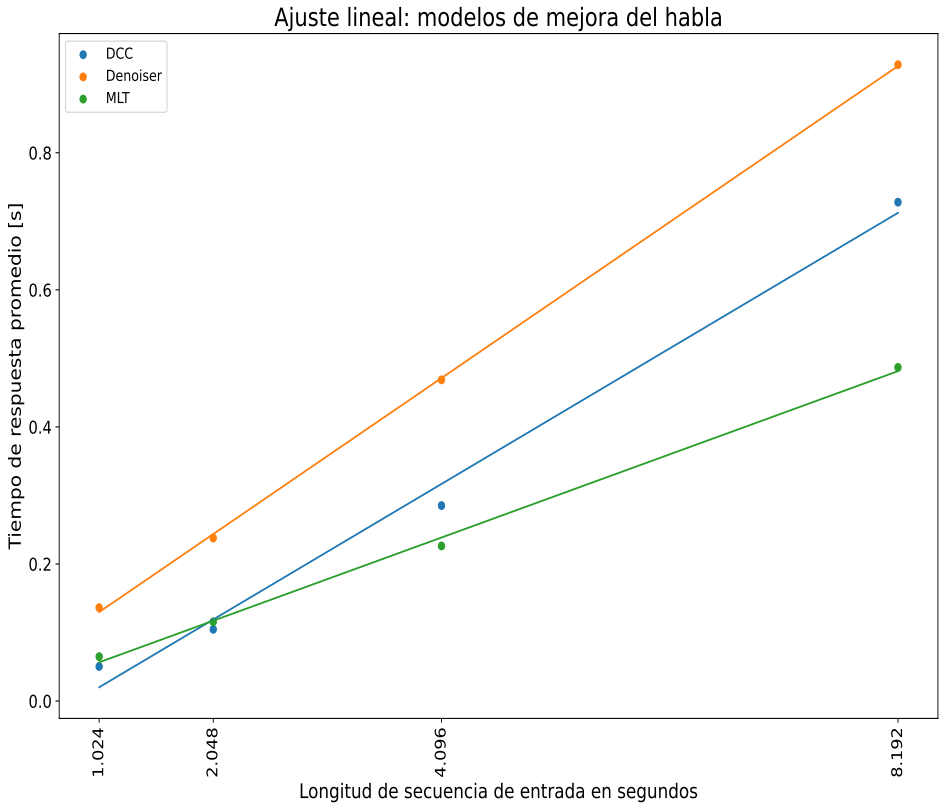


Figura B.2: Ajuste lineal del tiempo de respuesta promedio en relación con la longitud de la secuencia de entrada para los modelos de mejora del habla.

La Figura B.2 muestra que tanto el modelo MLT como el denoiser exhiben un comportamiento lineal en sus tiempos de respuesta. En contraste, el modelo DCCNet presenta una discrepancia notable, ya que la recta ajustada no intercepta ninguno de los valores graficados. Este fenómeno sugiere que el modelo DCCNet podría estar demostrando una complejidad diferente, posiblemente $O(n \log n)$, lo cual podría ser el resultado de ciertos aspectos de su diseño y estructura.