Universidad de Buenos Aires

Facultad de Ciencias Exactas y Naturales

Exploración de la limpieza de ruido en habla hispana con redes neuronales

Tesis de Licenciatura en Ciencias de Datos

Axel Fridman

Director: Pablo Riera

Buenos Aires, 2023

RESUMEN

Las señales de habla pueden estar contaminadas por ruidos que dificultan nuestra capacidad de comunicarnos, para resolver esta problemática existe la tarea de eliminación de ruido. En este trabajo se estudia un modelo basado en redes neuronales tomando como punto de partida el modelo HiFi-GAN [1]. A diferencia de este trabajo, nosotros decidimos utilizar habla en español. Se presentan una serie de experimentos centrados en mejorar resultados, que abarcan aspectos diversos como: evaluación de tres funciones de pérdida alternativas, entrenamiento con diferentes conjuntos de datos, comparación de distintas arquitecturas de red generadora y la implementación de curriculum learning.

Los resultados indican que una arquitectura reducida puede alcanzar resultados similares a los que se obtienen con la arquitectura original (para el conjunto de datos utilizados en este trabajo). También se utilizaron estrategias como decaimiento de la tasa de aprendizaje para optimizar el entrenamiento.

Además, se estudian las predicciones del modelo bajo diversas condiciones de ruido y para diferentes grupos de hablantes. También se incluye el código del proyecto proporcionando una herramienta valiosa para futuros trabajos.

Palabras claves: eliminación de ruido, habla, habla hispana, castellano, señales de voz, WaveNet, funciones de pérdida, Arquitecturas de red, curriculum learning, PostNet, tasa de aprendizaje, métricas Objetivas.

ABSTRACT

Speech signals can be contaminated by noises that hinder our ability to communicate, to solve this problem there is the task of denoising. In this work we study a model based on neural networks taking the HiFi-GAN [1] model as a starting point. Unlike this work, we decided to use Spanish speech. A series of experiments focused on improving results are presented, covering various aspects such as: evaluation of three alternative loss functions, training with different data sets, comparison of different generator network architectures and the implementation of curriculum learning.

The results indicate that a reduced architecture can achieve similar results to those obtained with the original architecture (for the data set used in this work). Strategies such as curriculum learning and learning rate decay were also used to optimize training.

In addition, the predictions of the model under various noise conditions and for different groups of speakers are studied. The code of the project is also included, providing a valuable tool for future work.

Keywords: Speech enhancement, Noise, Speech, Hispanic speech, Castilian, Spanish, Voice signals, WaveNet, Loss functions, Network architectures, Curriculum learning, PostNet, Learning rate, Objective metrics, Machine learning, Artificial intelligence, Deep learning model, Network training, Prediction model, Signal processing, Convolutional neural network, Optimization, Hyperparameter tuning, Noise addition.

AGRADECIMIENTOS

A mi madre por apoyarme, siempre preguntando por lo que hago y esforzándose por entender de manera técnica, con curiosidad, energía e ilusión.

A mi padre por bancarme y estar presente durante tantos años de carrera.

A mis hermanos, Dylan por acercarme a la ciencia, a la programación y a un mundo nerd fantástico e Ian por el cariño y las risas que nunca nos faltaron.

A mis amigos de la facultad con quienes armamos un grupo de estudio maravilloso. Con ellos no solamente pude avanzar en la carrera sino que además me llenaron la memoria de momentos increíbles. Específicamente quiero agradecer por ayudarme en esta tesis a Cecilia Bolaños, Giovanni Marraffini, Juan Wisznia, Andrés Cotton y Noé Hsueh.

A Valentina Adjimann por acompañarme, creer en mí, y lo que puedo alcanzar.

A mi director Pablo Riera por ser toda la guía que necesitaba y mucho más, estando presente prácticamente 24/7, cuando las cosas no funcionaban, no las entendía, y por calmar mis nervios y motivarme en momentos de mayor incertidumbre.

A Laura Pérgola por corregir y ayudarme con la gramática de esta tesis.

A MercadoLibre y al equipo de optimización logística por ceder parte del tiempo dedicado a realizar esta tesis.

A los y las profesores, y a la facultad por ser un ambiente lleno de potencial donde suceden cosas grosas diariamente. Que para mí es tanto más que un lugar donde se estudia.

Índice general

1. INTRODUCCIÓN

1.1. Definición del problema y relevancia

A la hora de comunicarnos mediante el habla nos encontramos con ruidos propios de los ambientes en donde habitamos, sonidos de la calle, de otras personas hablando, incluso la acústica de la habitación puede ser desfavorable para comprender lo que nos quieren comunicar. Si llevamos ese problema al ámbito digital, en donde estamos grabando y escuchando a través de nuestros dispositivos, que a su vez sufren distorsiones propias del canal en el que son transmitidos, se vuelve un problema complejo y muy importante.

Nuestro objetivo será entonces mejorar la calidad del audio limpiando el ruido de fondo. Esto nos provee un amplio abanico de aplicaciones de gran relevancia, entre estas aplicaciones se incluyen:

- Comunicación: la claridad y la inteligibilidad del habla son cruciales en situaciones de comunicación, como llamadas telefónicas, videoconferencias y transmisiones de audio en tiempo real.
- Transcripción automática: para sistemas de reconocimiento de voz, la reducción de ruido es muy útil para garantizar una transcripción precisa, especialmente en entornos ruidosos o con grabaciones de baja calidad.
- Asistentes de voz y dispositivos inteligentes: para asistentes virtuales y dispositivos inteligentes que utilizan reconocimiento de voz; la mejora en la calidad del audio es beneficiosa para una interacción con menos malentendidos.
- Mejora de la calidad de datos para detectar emociones: la limpieza del habla (*enhancement* en inglés) sirve para facilitar la detección precisa de las emociones expresadas en el discurso, lo que es útil en áreas como la psicología.

1.2. Objetivos y contribución

En este trabajo, se tiene como propósito principal replicar y extender los hallazgos del artículo de referencia "HiFi-GAN: High-Fidelity Denoising and Dereverberation Based on Speech Deep Features in Adversarial Networks" de Jiaqi Su, Zeyu Jin y Adam Finkelstein [1], ya que tuvo resultados superiores a todos los trabajos previos de la época. En aquel modelo utilizaron principalmente la arquitectura WaveNet [2] junto con otras componentes tales como redes convolucionales y densas. También incluyeron una función de pérdida sobre la base de discriminadores adversarios tanto en el dominio temporal como en el espectrograma. El artículo cuenta con una versión mejorada por los mismos autores llamada HiFi-GAN-2 [3] publicada en 2021.

Se decidió utilizar este tipo de modelo porque es práctico para el propósito de nuestra investigación. Actualmente, se están utilizando arquitecturas Transformer para procesamiento de audio, aunque en la mayoría de los casos implica un incremento en el costo de entrenamiento.

En el laboratorio donde se realizó nuestro trabajo, ya se habían intentado replicar los resultados de este modelo sobre la base de un código liberado no oficial, sin tener buenos resultados. En esta oportunidad intentamos replicar los resultados, pero desarrollando el código desde cero.

Sobre la capacidad de replicar resultados de artículos publicados, podemos mencionar cómo la reproducibilidad en el área de aprendizaje automático tiene sus dificultades [4], más aún cuando no hay código oficial disponible.

Otra complejidad por la cual no es sencillo replicar resultados está relacionada al acceso a los datos. Si bien en este caso era posible trabajar con una gran parte de los datos utilizados, decidimos diferenciarnos con el trabajo original e introdujimos una variante y nos enfocamos en habla en español en vez de en inglés.

Las otras variantes que se exploraron están relacionadas a variaciones de arquitectura e hiperparámetros. Específicamente, exploramos en una serie de experimentos los impactos de la cantidad de capas, tasa de aprendizaje, aprendizaje por currículum, funciones de pérdida y otros. Estos experimentos fueron realizados sin agregar los discriminadores adversarios que estaban presentes en el artículo original. Para estas exploraciones se desarrolló una metodología para explorar estos hiperparámetros y medir sus resultados en cuanto a métricas objetivas y funciones de pérdida.

Además de estudiar variantes de hiperparámetros, analizamos la facilidad que tiene el modelo de limpiar el ruido según el género y país de cada hablante. Siendo que el español es un idioma con diversos dialectos, ver el rendimiento en cada grupo (género, país) nos permite detectar sesgos. De estar presentes, resultan en una experiencia desigual para diferentes grupos de hablantes, lo que limita su aplicabilidad en situaciones del mundo real con diversidad de hablantes.

Este problema es incluso más frecuente cuando se cuenta con datasets desbalanceados, como los que se utilizaron en este trabajo. Es por esto que si bien es un tópico difícil de tratar, ser conscientes y comunicar este tipo de disparidades permite tener modelos más transparentes.

2. TRABAJOS PREVIOS

2.1. Revisión bibliográfica

La problemática de reducir el ruido en el habla representa un desafío técnico importante que fue estudiado considerablemente a lo largo de los años.

En las décadas de 1960 y 1970, los primeros intentos de abordar el problema del ruido en el habla se centraron en técnicas de filtrado y mejora de señales para reducir el ruido de fondo. Se desarrollaron técnicas básicas de filtrado, como los filtros de promedio y los filtros de mediana. El desarrollo de la cancelación de eco, una forma de filtrado adaptativo, fue un hito importante durante este período [5].

Posteriormente, en la década de 1980, se introdujeron técnicas más avanzadas para la supresión de ruido, como el uso de algoritmos de cancelación de ruido adaptativos [6]. Estos algoritmos son capaces de adaptarse dinámicamente al entorno y reducir el ruido de fondo.

Durante la década de 1990 se produjo un avance significativo con la introducción de técnicas basadas en el dominio de la frecuencia, como el espectrograma de potencia mínima y el espectrograma de potencia mínima estimada [7]. Estos métodos permitieron una mejor separación entre el habla y el ruido. Sin embargo, estas técnicas a menudo se combinaban con otros enfoques, como el filtrado adaptativo, para obtener los mejores resultados.

Ya en los años 2000, las técnicas de modelado estadístico como los Modelos Ocultos de Markov (HMMs) [8], se popularizaron para el reconocimiento del habla en entornos ruidosos. Los enfoques de aprendizaje profundo comenzaron a utilizarse hacia finales de esta década.

En la segunda década del siglo XXI, con el auge de las redes neuronales profundas, se han desarrollado nuevos enfoques para el problema de la reducción de ruido en el habla. La aplicabilidad de las redes neuronales profundas (DNN) en este ámbito fue examinada por Park y Shin [9], quienes propusieron un método de reducción de ruido para señales de voz utilizando DNN. Más adelante, Weninger *et al.* [10] se centraron en la mejora del habla haciendo uso de las redes neuronales recurrentes de memoria a largo plazo (LSTM). Su estudio concluyó que las LSTM pueden utilizarse con éxito para mejorar la robustez del reconocimiento de voz en entornos ruidosos. Este trabajo puso de manifiesto la fortaleza de los modelos secuenciales para la tarea de procesamiento del habla. En 2017 Rethage *et al.* [11] utilizaron la arquitectura WaveNet para hacer limpieza del habla.

Los modelos de aprendizaje profundo también han evolucionado hacia estructuras más complejas, como las redes neuronales convolucionales recurrentes (CRNN) [12] y la arquitectura transformer para la limpieza del habla ruidosa [13]. El estudio de Pandey y Le [12] mostró que las CRNN pueden ser útiles para realizar una mejora espectral del habla.

El trabajo de Hu y Loizou [13] propuso la utilización de la arquitectura del transformer, originalmente propuesta para tareas de procesamiento del lenguaje natural, el cual ha demostrado su potencial también en la tarea de mejora del habla. Los avances recientes

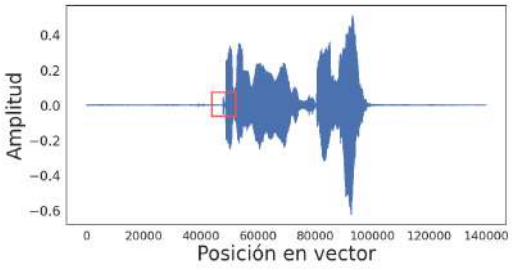
han visto un cambio hacia enfoques más completos para el procesamiento de voz, como lo demuestra el Wave Transformer [14] de Kong *et al.* Este enfoque unifica muchos de los desafíos de procesamiento del habla en una sola arquitectura, es decir, resuelve problemas relacionados con el procesamiento del habla (por ej., reducción de ruido, reverberación, etc.) de forma simultánea en lugar de tratarlos como tareas individuales. Aquel trabajo representa un prometedor camino a seguir para futuras investigaciones.

En resumen, la investigación en la reducción de ruido en el habla ha evolucionado significativamente y actualmente se beneficia de las técnicas de aprendizaje profundo. Aunque se han logrado avances considerables, el desafío de mejorar el habla en entornos muy ruidosos sigue siendo una tarea abierta.

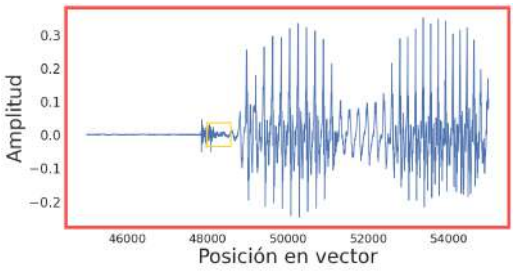
3. MODELADO

3.1. Audio

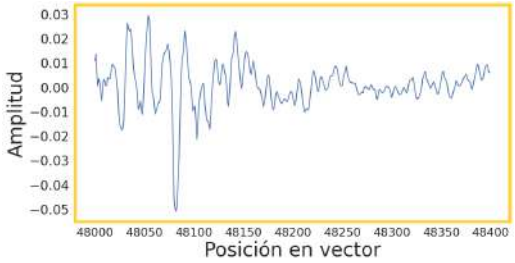
La representación de audio en formato digital se almacena mediante un par de elementos distintos: la tasa de muestreo y un vector de valores. La tasa de muestreo, un número entero positivo, denota la cantidad de valores en el vector que conforman un segundo completo de audio. Por otro lado, cada elemento dentro del vector nos indica la amplitud de la señal en esa posición y se guarda usualmente como un número de punto flotante de 16 bits, cuyo rango esta entre -1 y 1. De esta forma si un vector tiene longitud 60000 y la tasa de muestreo es 20000 sabemos que el audio tiene 3 segundos de duración. Para poder apreciar la forma de la señal ver Figura 3.1.



(a) Forma de onda con silencio al comienzo y final de la grabación, de duración 8,75 segundos.



(b) Zoom del rectángulo rojo en la imagen 3.1a.



(c) Zoom del rectángulo amarillo en la imagen 3.1b.

Fig. 3.1: Señal en dominio temporal con 3 niveles de aumento.

Por otro lado, se puede transformar el audio en un espectrograma, que es una representación de dos dimensiones (tiempo y frecuencia) del audio (Figura 3.2). Se calcula aplicando la transformada de Fourier de tiempo corto (STFT) a una señal de audio, la cual divide la señal en segmentos temporales y aplica la transformada de Fourier a cada segmento. Luego del resultado de esto se toma el valor absoluto y se eleva al cuadrado, representando la intensidad de las frecuencias en función del tiempo.

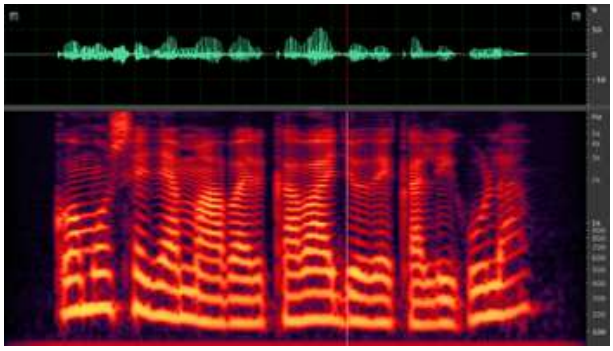


Fig. 3.2: En la parte superior se tiene la forma de onda y en la inferior su espectrograma. Eje Y representa frecuencias mientras que el eje X el tiempo. Archivo tomado de aquí.

3.2. Redes neuronales

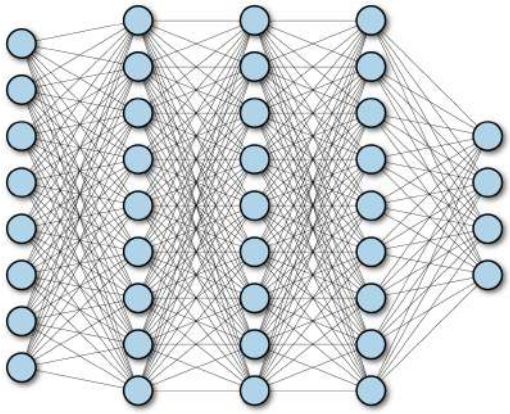
Las redes neuronales son modelos de aprendizaje automático formadas por unidades interconectadas (neuronas) que transforman un conjunto de entradas en algún *output* o salida. Cada neurona se encarga de procesar una pequeña parte de la información y su funcionamiento se basa en la combinación lineal de las entradas y una función de activación no lineal.

En términos generales, una red neuronal funciona al procesar las entradas a través de capas sucesivas de neuronas, transformándolas hasta generar las salidas. Este proceso se denomina propagación hacia adelante (*feed-forward*). Además, las redes neuronales tienen la capacidad de aprender de los datos, es decir, pueden ajustar sus parámetros internos (conocidos como pesos) para mejorar su rendimiento. Este aprendizaje se realiza mediante un proceso llamado retropropagación (*backpropagation*), que utiliza un algoritmo de optimización (como el descenso por gradiente) para reducir la diferencia entre las salidas generadas por la red y las salidas deseadas.

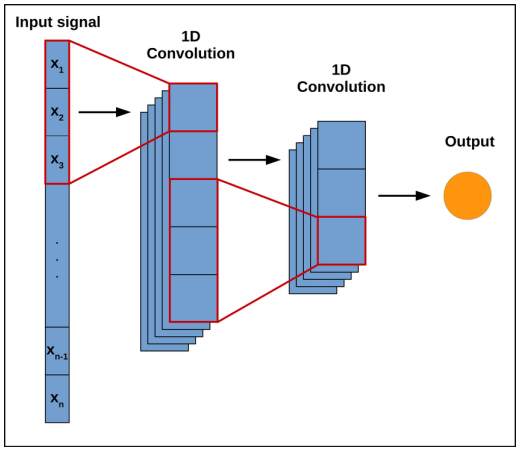
En cuanto a la arquitectura, cada capa de una red puede ser densa. Esto quiere decir que cada neurona de la capa está interconectada con todas las neuronas de la capa anterior, ver Figura 3.3a. Notemos que, como el audio está representado como un vector en su formato natural (dominio temporal), esta entrada es unidimensional. Esto significa que una red neuronal completamente conectada (densa) para que funcione en este dominio debe tener una cantidad de parámetros en la primera capa de la red equivalente a la longitud del vector del audio multiplicado por la cantidad de neuronas de esa capa. Este tipo de conexiones, además de implicar un alto número de parámetros, requieren también saber el tamaño de entrada de antemano tanto para entrenamiento como inferencia.

Alternativamente, las redes neuronales convolucionales (CNN) utilizan operaciones de convolución para procesar datos. Esta red es especialmente útil para trabajar con imágenes y sonidos debido a su capacidad de extraer información con invarianza a traslaciones de los datos de entrada.

En una CNN, cada neurona en la capa de convolución recibe información de un número limitado de nodos en la capa anterior (Figura 3.3b). Esto se realiza mediante el uso de filtros, cuyo tamaño y profundidad son definidos por el tamaño del kernel y el número de canales respectivamente. Estos filtros recorren la capa de entrada, analizando pequeñas



(a) Red neuronal densa de 5 capas, figura tomada de O'Reilly.



(b) Red neuronal con 2 convoluciones con tamaño de kernel 3. La primera convolución va de 1 canal a 5 canales, y la segunda convolución va de 5 canales a 5 canales. Figura proveniente de [15].

Fig. 3.3: Distintas arquitecturas de redes neuronales artificiales.

porciones de esta en cada paso.

Adicionalmente, la aplicación de estos filtros puede ser con dilatación, que es un parámetro que determina la distancia entre los valores que serán considerados por el filtro, así permite detectar características más globales de los datos. En la Figura 3.4 se ve como hay 4 convoluciones, cada convolución con una dilatación distinta (de arriba hacia abajo [8, 4, 2, 1]), generando que el nodo naranja reciba información de 31 nodos sin necesidad de tener un tamaño de kernel de esa longitud. Se puede notar como en la Figura 3.4 las convoluciones están centradas, pero no necesariamente deben estarlo, si una neurona solo recibe información de neuronas predecesoras de la capa anterior entonces se denomina convolución causal.

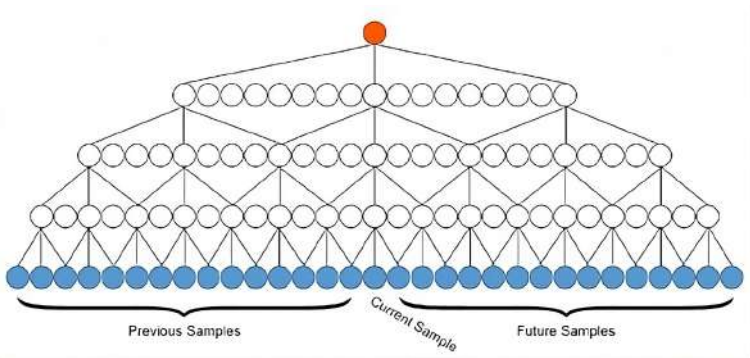


Fig. 3.4: Convoluciones dilatadas no causales con un único canal tomadas del trabajo de [11].

Al aplicar estos filtros, la red neuronal puede detectar características locales, cada una asociada a un canal específico, como cambios tonales o patrones temporales en el audio. Los canales refieren a la dimensión de profundidad del filtro aplicado en la red neuronal. En una

imagen típica RGB, cada canal corresponde a uno de los colores de la imagen (rojo, verde y azul). Mientras que, en el caso de una CNN aplicada al análisis de audio, cada canal puede ser interpretado como un filtro que se centra en extraer una característica o frecuencia específica del audio. Así, el número de canales en una convolución determina la cantidad de distintos tipos de características que la convolución puede extraer del audio.

Otra componente importante de las redes neuronales son las conexiones de salto (*skip connections*) que sirven para combatir el problema del desvanecimiento del gradiente. Estas permiten que el gradiente salte de una capa a otra sin ser modificado por las capas intermedias.

Las conexiones residuales llevan esto un paso más allá. En lugar de simplemente enviar la salida de una capa a otra capa posterior pero no siguiente, suman esa salida a la entrada de la capa posterior. Esto ayuda a prevenir la desaparición del gradiente, especialmente en redes muy profundas, y facilita el aprendizaje de la red.

La clave de esta estrategia es que la red no necesita aprender la función completa $f(X)$ para producir la salida deseada Y, Figura 3.5. En su lugar, puede concentrarse en aprender solo la diferencia $f(X) = Y - X$ que necesita ser agregada a la entrada para producir la salida correcta.

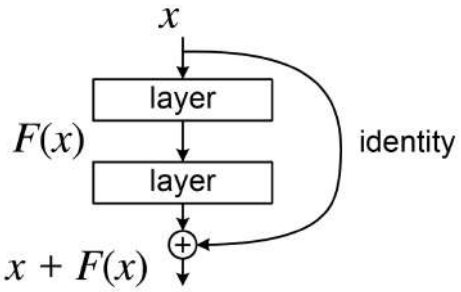


Fig. 3.5: Ejemplo de conexión residual

Esta simple, pero efectiva, estrategia ha permitido la formación de redes mucho más profundas, abriendo la puerta al aprendizaje de características aún más complejas.

Al trabajar con audios, podemos introducir los datos en su formato original de una sola dimensión, donde cada punto representa una amplitud del vector en un momento específico. O, alternativamente, se puede trabajar con el espectrograma del audio. Al hacer esto convertimos el audio en un espectrograma que la CNN puede analizar y procesar de la misma manera que lo haría con una imagen regular, tomando como filtro en este caso una matriz 2D (o 3D si es una capa con varios canales de profundidad). Al largo y ancho de esa matriz se lo llama tamaño del kernel.

Una PostNet es simplemente una red neuronal que va después de otra red predecesora. Suele estar basada en convoluciones y ayuda a realizar ajustes finales en la salida, mejorando la calidad de la señal de audio generada.

3.3. Arquitectura

3.3.1. WaveNet

WaveNet es una arquitectura de red neuronal convolucional dilatada profunda introducida por van den Oord *et al.* [2] en 2016. Esta red ha sido diseñada específicamente para modelar y generar secuencias de audio y se caracteriza por su habilidad para capturar patrones temporales complejos. Fue usada con éxito para hacer limpieza de ruido desde 2017 [11] al modificar ligeramente la arquitectura para hacerla no causal, ya que en la tarea de eliminación de ruido en el habla es común contar con muestras futuras que mejoran las predicciones. También en aplicaciones de tiempo real con cierta latencia en la respuesta del modelo se accede a información útil sobre muestras que suceden inmediatamente después de la muestra de interés.

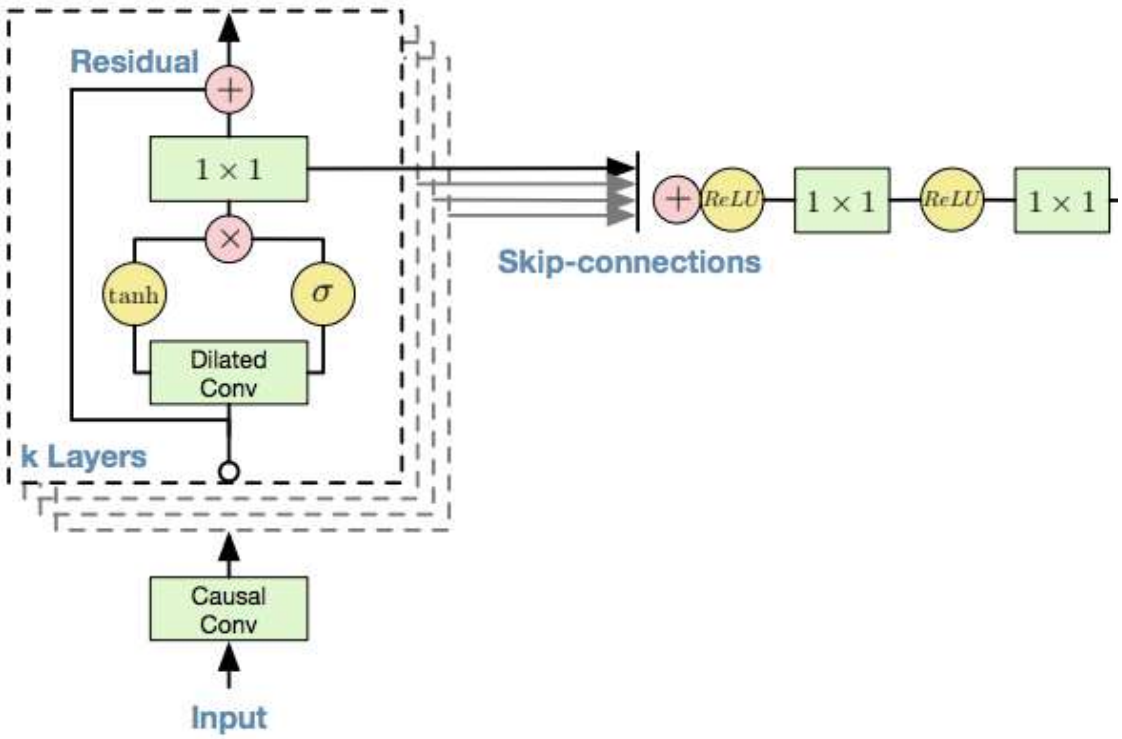


Fig. 3.6: Arquitectura de la WaveNet, Figura proveniente del trabajo original [2].

La arquitectura WaveNet está constituida por tres etapas:

1. Convolución inicial.
2. Multiples pilas residuales ordenados en serie. Cada pila residual con una cantidad de K capas donde cada capa tiene una dilatación distinta.
3. Dos convoluciones finales.

Convolución inicial

La convolución inicial (en la Figura *3.6* se muestra como *Causal conv*) es la primera etapa de la red y tiene como propósito llevar del canal único original a la cantidad de canales

de profundidad del modelo, de 1 a 128 en nuestro caso. Trabaja con un tamaño de kernel de 3 y, como ya se mencionó, en nuestro caso no es causal ya que en pruebas iniciales se encontró que se tenía mejores resultados.

Pila residual

En cada pila residual hay K capas, para cada numero de capa i entre [1, K] se cuenta con convoluciones distintas:

1. Una convolución dilatada $D_i(X)$ cuyo tamaño de kernel es 3, dilatación 2^i y de 128 canales a 128 canales. La convolución principal de la capa.
2. Una convolución simple $R_i(X)$ cuyo tamaño de kernel es 1, sin dilatación y de 128 a 128 canales.
3. Una convolución simple $S_i(X)$ cuyo tamaño de kernel es 1, sin dilatación y va desde 128 canales a 1 unico canal. Su objetivo es además de cambiar la cantidad de canales, es transformar linealmente el vector para ser sumado con las otras conexiones de salteo.

Dada una entrada X de 128 canales a la capa i (salida capa i-1 de una pila residual), se obtienen 2 salidas, por un lado la entrada de la siguiente capa de la pila ($ResCapa_i$) y por otro lado la conexión de salteo ($SalteoCapa_i$).

$$\text{ResCapa}_i(X) = X + R_i(\tanh(D_i(X)) \cdot \sigma(D_i(X)))$$
$$\text{SalteoCapa}_i(X) = S_i(\tanh(D_i(X)) \cdot \sigma(D_i(X)))$$

Donde σ(X) es la función sigmoidea y tanh(X) la tangente hiperbólica.

Se puede notar que $\tanh(D_i(X)) \cdot \sigma(D_i(X)$ es equivalente a haberle aplicado a $D_i(X)$ la función señalada en rojo en la Figura 3.7.

Luego para todo $i > 1$

$$\text{Residual}_i(X) := \text{ResCapa}_i(\text{Residual}_{i-1}(X))$$

donde X es la entrada de la pila. En caso de ser la primera pila, proviene como salida de la convolución inicial de la WaveNet, y en caso contrario, como salida de la pila anterior.

Mientras que de esta forma la salida de la conexión de salteo de la capa i está definida como:

$$\text{Salteo}_i(X) := \text{SalteoCapa}_i(\text{Residual}_{i-1}(X))$$

Finalmente, la salida de todas las pilas residuales que pasará hacia las convoluciones finales de la WaveNet será la suma de todas las salidas de salteo de todas las capas de todas las pilas residuales:

$$\sum_{i=1}^{n} \text{Salteo}_i(X)$$

Se puede notar que $ResCapa_K$ no será usada en ningún momento.

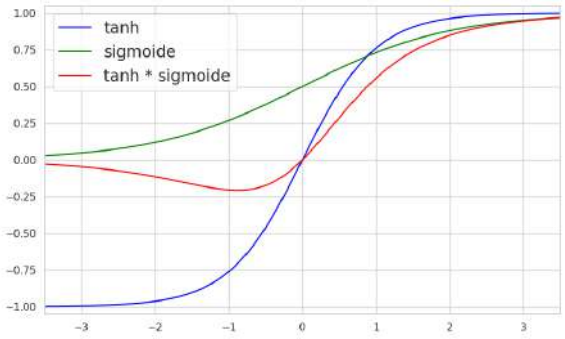


Fig. 3.7: Funciones de activación

Convoluciones finales

Luego de sumar todas las conexiones de salteo y obtener así un solo vector de un solo canal de la longitud del audio original, se les aplicará 2 convoluciones de tamaño de kernel 3, con una función de activación ReLU entre medio.

3.3.2. PostNet

Utilizar PostNets ha mostrado en otros trabajos mejorar la calidad del audio [16]. Es por eso que se implementó una PostNet que toma como entrada la salida de la WaveNet.

Esta PostNet será activada después de una cantidad de pasos de entrenamiento, 105k pasos en principio. Cuando se dice que será activada se refiere a que antes de alcanzar esta cantidad de pasos el gradiente solo se está propagando desde el final de la WaveNet, es decir, se ignoran los efectos de la PostNet. El motivo de esto es que se quiere dejar a la WaveNet aprender lo máximo posible sin tener que pagar el costo de, en simultáneo, ajustar los parámetros de la PostNet y que la WaveNet capte la estructura de onda del habla.

Una vez que se prende la PostNet, no solamente se retropropaga desde la salida de la PostNet, sino que además desde la salida de la WaveNet. Los autores del trabajo original mencionan que esto es para que la WaveNet se concentre en generar el grueso del audio, mientras que la PostNet solamente elimine artefactos de sonido.

La forma en la que se ha implementado esto fue simplemente sumar los errores de predicción de la WaveNet sola y con la PostNet, agregando un hiperparámetro de peso que multiplica los errores de la PostNet. En este caso se utilizó 3, es decir, los errores de predicción de la PostNet valían 3 veces lo que valen los errores de la WaveNet.

Esta PostNet cuenta con una convolución inicial que lleva el audio de 1 canal a 128, y es seguido por 12 convoluciones no causales y sin dilatación con tamaño del kernel de 33. Cada una de estas 12 convoluciones tienen sesgo y luego de la doceava se toma una última convolución final también de con tamaño de kernel 33 para llevar los canales de 128 a 1 para ser el *output* final del generador.

3.4. Funciones de pérdida

Utilizar múltiples funciones de pérdida como L1 y mel espectrograma en la limpieza de audio es considerado beneficioso, ya que L1 se enfoca en detalles temporales, mientras que el mel espectrograma atiende características perceptuales. Además, se definen funciones de pérdida novedosas para estudiar su impacto en la optimización del modelo. Las funciones utilizadas fueron:

1. Función de pérdida L1
2. Función de pérdida mel espectrograma
3. Función de pérdida proporcional
4. Función de pérdida de amplitud

3.4.1. Función de pérdida L1

Actúa en el dominio temporal y es la distancia promedio entre la señal predicha y la real.

$$\sum_{i=1}^{n} \frac{1}{n} |\hat{y}_i - y_i|$$

Es rápida de calcular, pero tiene una capacidad limitada de capturar patrones complejos, es sensible a *outliers*, y trata todos los errores igual sin importar su impacto en la percepción del habla. Además traslaciones de pocos milisegundos en la señal predicha relativa a la real generan errores a pesar de que puedan ser indistinguibles auditivamente. Errores de 0.05 tienen impactos perceptiblemente distintos al producirse en la realidad con una onda de 0, comparado con una onda de 0.8, pese a que la función de pérdida no hace tal diferenciación. Para más claridad sobre este ejemplo ver Figura 3.8.

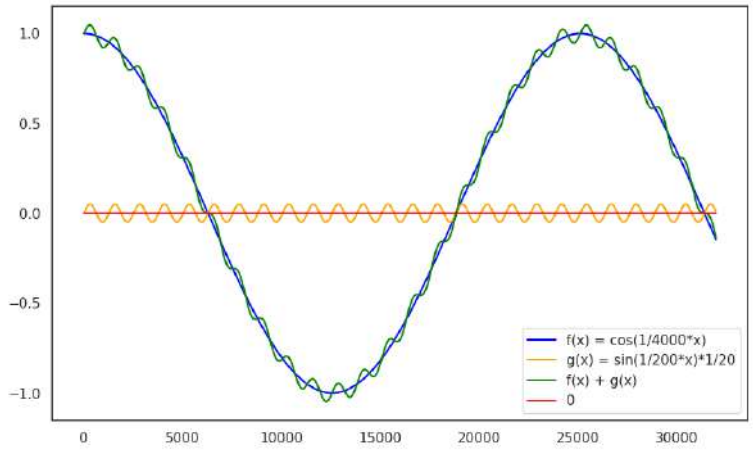


Fig. 3.8: Caso en donde el error de la función naranja al predecir la roja debería ser más grave que el de la verde tratando de predecir la azul, sin embargo su error L1 es el mismo.

3.4.2. Función de pérdida mel espectrograma

Para mejorar la precisión y la percepción del habla en señales de audio, se propuso una función de pérdida que trabaje con dos tipos de mel espectrogramas. Los mel espectrogramas tienen la ventaja de tener una mayor resolución de frecuencias en las áreas en las que los humanos tenemos una percepción del habla más precisa. Uno de los 2 espectrogramas tiene mayor resolución temporal y el otro mayor resolución de frecuencias.

La función de pérdida propuesta, $\mathcal{L}$, se calcula en dos etapas. Primero, transformamos los espectrogramas a una escala logarítmica (decibeles), truncándola para decibeles muy bajos para que no trate de ajustar a esos sonidos. Así, el Log-Espectrograma se calcula de la siguiente manera:

$$S_{dB} = \text{máx}(10 \cdot \log_{10}(S - \text{mín}(S) + \epsilon), mínimoDecibeles)$$

donde S_{dB} es el espectrograma calculado en escala de decibeles, S es el espectrograma original, mín(S) devuelve el valor mínimo de S, ϵ es una constante pequeña utilizada para la estabilidad numérica y mínimoDecibeles es una constante para determinar el rango dinámico.

En la segunda etapa, calculamos el ECM ponderado para cada tipo de Log-Espectrogramas,

$$\mathcal{L} = \frac{1}{N * V} \sum_{i=1}^{N} \sum_{j=1}^{V} (1 + \frac{i \cdot l}{N}) \cdot (S_{dB_{i,j}} - \hat{S}_{dB_{i,j}})^2$$

donde S_{dB} y $\hat{S}_{dB}$ representan los valores del espectrograma verdadero y predicho respectivamente, l es el hiperparámetro de aumento de peso a frecuencias agudas (ya que se asume que i = 1 es la frecuencia más grave, e i=N la más aguda), N es el número de bins de frecuencia en el espectrograma, V es el número de ventanas en el espectrograma y $\mathcal{L}$ es la pérdida calculada.

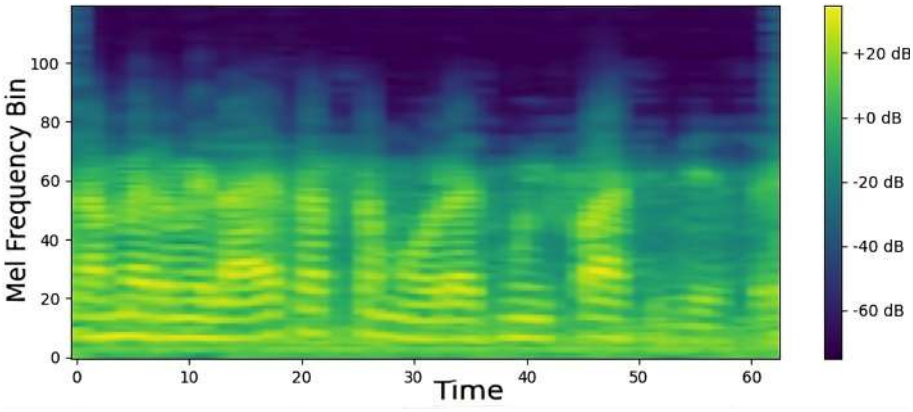
La función de l es que a las frecuencias más agudas se las pese por l, mientras que a las más graves por 1. Esto fue agregado en vista a analizar espectrogramas predichos y verdaderos y ver sus fallas, Figura 3.9. Se ve como claramente las frecuencias más agudas son las más difíciles de reproducir por el generador, entonces para asistir en su generación se agregó este hiperparámetro y se fijó l en 2.

El primer tipo de mel espectrograma, denotado como mel espectrograma 1, tiene una mayor resolución de frecuencia. Esta resolución se logra utilizando los siguientes parámetros: n_fft = 2048, n_mels = 120 y hop_length = 512.

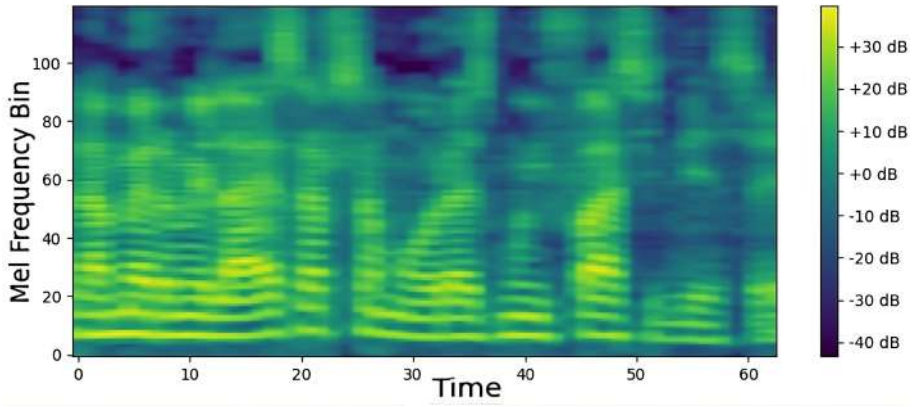
El segundo tipo, denominado Mel Espectrograma 2, tiene una mayor resolución temporal. Los parámetros utilizados son: n_fft = 512, n_mels = 80 y hop_length = 128. Para ambos tipos de espectrogramas, se utiliza la ventana de Hann.

Como cada mel espectrograma tiene su propio peso y nuestras funciones de pérdida funcionan de manera independiente para cada uno, este procedimiento se calcula para ambos tipos.

En resumen lo que nos aportan estas dos funciones de pérdida son definir de una manera distinta el ECM del espectrograma ponderado en bins de frecuencia.



(a) Mel espectrograma predicho



(b) Mel espectrograma limpio

.

Fig. 3.9: Espectrograma 1 de validación durante el entrenamiento, en el que las frecuencias más agudas se encuentran en la parte superior de las Figuras.

3.4.3. Función de pérdida proporcional

También se propuso en este trabajo evaluar si, en el dominio temporal, el error relativo a la amplitud de la señal original al cuadrado aceleraba el entrenamiento o mejoraba la percepción del habla, definida como:

$$\sum_{i=1}^{n} \frac{1}{n} \cdot \left(\frac{\hat{y}_i^2}{{y_i}^2 + \epsilon} - 1 \right)^2$$

Valores mas grandes en esta función de pérdida provienen cuando y_i tiene un valor pequeño a nivel amplitud (silencio) pero $\hat{y}$ no comparte la misma magnitud. El término ϵ se introduce para evitar divisiones por cero.

3.4.4. Función de pérdida de amplitud

Viendo gráficos de la señal temporal predicha en comparación con la real se veía como consistentemente la amplitud de la predicha era menor a la real. Para tratar este tema se ha propuesto esta función de pérdida que solo se activa cuando la amplitud de la real es superior absolutamente a un valor, lo que nos permite ponderar con un peso mayor que con la L2 normal. Se define entonces la función indicadora de cuando deberá activarse la función:

$$G(X) = \begin{cases} 1 & \text{si } X > 0{,}1 \text{ o } X < -0{,}1 \\ 0 & \text{en otro caso} \end{cases}$$

Mientras que la función de pérdida de amplitud en sí será definida entonces como:

$$\sum_{i=1}^{n} \frac{1}{n} \cdot G(y_i) \cdot (y_i - \hat{y}_i)^2$$

3.5. Métricas objetivas

Para evaluar la calidad de un entrenamiento de modelo en el que se están adaptando los pesos a las distintas funciones de pérdida es importante tener un punto de referencia de qué se quiere optimizar. Después de todo si se pusiera 0 de peso a todas las funciones de pérdida el modelo podria devolver cualquier señal y su error en validación sería 0. Es por eso que se utilizó PESQ, SNR, SRMR y STOI. Una métrica es considerada invasiva si requiere la señal original o de referencia para realizar la evaluación

PESQ (*Perceptual Evaluation of Speech Quality*): es una métrica invasiva que evalúa la calidad perceptual de señales de habla procesadas. Sus valores oscilan entre -0.5 y 4.5, donde 4.5 representa una calidad de habla sin degradación y valores más bajos indican una mayor degradación perceptual.

SNR (*Signal-to-Noise Ratio*): es una métrica invasiva que mide la relación entre la potencia de la señal deseada y la potencia del ruido de fondo. Los valores de SNR se expresan en decibelios (dB), y no hay límites teóricos para su rango, pero valores más altos, en general, indican una mejor claridad de la señal hablada.

SRMR (*Speech-to-Reverberation Modulation Energy Ratio*): es una métrica no invasiva que evalúa la inteligibilidad de señales de habla. No tiene límites teóricos específicos, pero los valores más altos indican una mayor inteligibilidad. Combina medidas objetivas de relación señal-ruido y redundancia espectral en intervalos cortos para proporcionar una estimación precisa de la inteligibilidad.

STOI (*Short-Time Objective Intelligibility*): es una métrica invasiva que mide la inteligibilidad de señales de habla. Sus valores oscilan entre 0 y 1, donde 1 representa una inteligibilidad perfecta y 0 indica una inteligibilidad nula. STOI es especialmente útil en entornos ruidosos y para evaluar sistemas de comunicación, siendo sensible a la degradación de la señal.

3.6. Hiperparámetros

Se utilizó el optimizador AdamW con sus parámetros por defecto, la tasa de aprendizaje inicial en 0.001 y con decaimiento exponencial detallado en experimento 3.

Los hiperparámetros eran más de 80 y se tuvieron que estructurar en archivos json que eran leídos de una pila para poder ser utilizados. La estructura de los hiperparámetros puede encontrarse en el apéndice A5.12.

3.7. Detalles sobre el desarrollo y entrenamiento

En este trabajo, se siguió un enfoque orientado a mantener una versión estable del código mediante un procedimiento iterativo e incremental. Se puso especial énfasis en la modularidad del programa para facilitar el diagnóstico rápido y sencillo de los problemas que se fueron teniendo.

Se optó por utilizar los servidores de nuestra facultad, accediendo de manera remota. Esto aceleró el ritmo de trabajo y permitió aprovechar la GPU del grupo, una NVIDIA GeForce RTX 3060 Lite Hash Rate con aproximadamente 12 GB de memoria.

Utilizando este hardware, se fijo el tamaño del lote (*batch size*) en 4 que era lo máximo que entraba en memoria, mientras que se decidió utilizar acumulación de gradiente por cada 2 lotes. Esto quiere decir que el modelo actualizaba sus parámetros y realiza retropropagación luego de ver 8 audios distintos.

El diseño experimental implicó una serie de entrenamientos con diferentes hiperparámetros. Para optimizar su manejo, se implementó una cola de experimentos que corrían de manera automatizada a lo largo de días y semanas. Cada ciclo de entrenamiento, incluyendo pre-entrenamiento y fine-tuning, procesaba 255.000 audios (200.000 únicos), equivalentes a 111 horas de habla no repetida.

Para el monitoreo de la evolución del modelo, se utilizó Tensorboard para trazar formas de onda, espectrogramas, métricas objetivas y pérdidas. Estas métricas se evaluaban cada 1000 audios de entrenamiento. No obstante, dada la magnitud del conjunto de validación, se seleccionó un subconjunto de 100 audios para la validación intermedia (la que se corre para ver curvas de aprendizaje en validación). Al final de cada entrenamiento, el modelo se validaba de forma completa para un punto de evaluación más robusto.

Finalmente, después de cada ejecución se exportaban automáticamente todas las métricas y pérdidas a un archivo CSV. Este proceso de trabajo estructurado permitió manejar las corridas y obtener evaluaciones comparables de los modelos.

4. DATOS

Para la tarea de limpieza de ruido en audio es necesario tener un conjunto de datos limpios de buena calidad, diverso que debe incluir diferentes tipos de voces (por ejemplo, voces masculinas y femeninas, voces de diferentes edades y dialectos, etc.) y diferentes contextos (tales como diferentes volúmenes, tonos, estilos de habla, etc.).

La variedad es clave para reducir sesgos hacia grupo de hablantes, ya que los datos que se necesitan deben ser los más parecidos a los datos que serán limpiados en la realidad.

Por la misma línea, tener distintos tipos de ruidos, de distintos ambientes e intensidades son tan importantes como las grabaciones. Se utilizan ruidos de tránsito, de personas caminando al aire libre, en el subte, etc., porque son los lugares y los ruidos en donde las personas suelen graban audios en la realidad.

Otro componente necesario a nivel materiales es un conjunto de datos de respuestas al impulso. Las respuestas al impulso son las características acústicas de un entorno o sistema, capturadas al registrar la salida de dicho sistema cuando se le aplica un impulso breve. Dicho coloquialmente nos dice cómo llegan de manera directa e indirecta (rebotando) las señales sonoras en una sala hasta retornar al dispositivo que está grabando. En la Figura 4.1 se muestra la forma de onda de una respuesta al impulso.

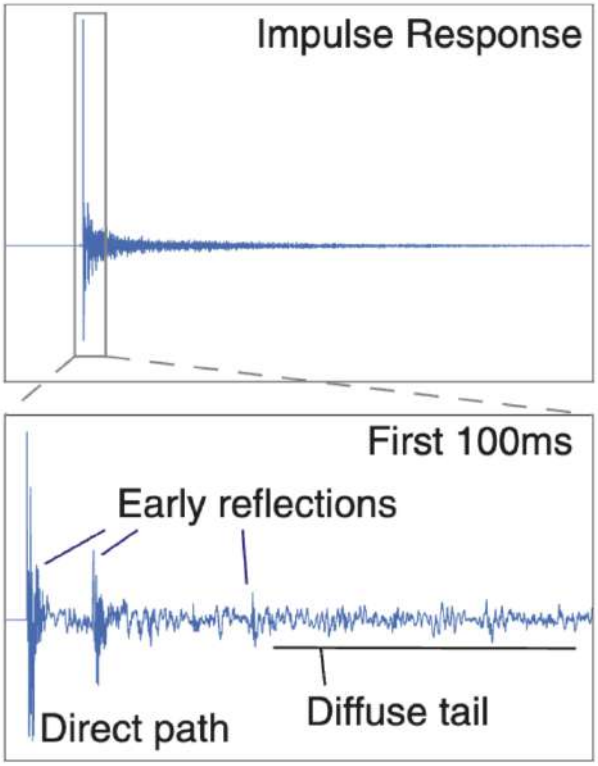


Fig. 4.1: Respuesta al impulso grabada en un restaurante en Cambridge en el trabajo de [17].

Además de la diversidad, el volumen de un conjunto de datos de entrenamiento también es muy importante para que un modelo generalice a distintos hablantes y ruidos.

Una forma de trabajar con audios para limpieza es contar con audios de habla limpios y con audios de ruidos e ir ensuciando los audios limpios para que sean la entrada de un modelo. Una alternativa, si se cuenta con grabaciones sucias, y esos audios fueron limpiados, es entrenar como entrada con la grabación ya sucia (sin tener que ensuciar el audio con ruidos).

4.1. Conjunto de datos

Se utilizaron seis datasets, tres de habla, uno de respuestas al impulso para reverberación y dos de ruidos ambientes. Se tomaron tres datasets de habla con el objetivo de capturar diversidad de dialectos utilizando distintos países.

4.1.1. Datasets de habla

El primer dataset "Crowdsourcing Latin American Spanish for Low-Resource Text-to-Speech" (CLAS) [18] de Guevara-Rukoz *et al.* incluye 19.446 grabaciones de 174 hablantes tanto masculinos como femeninos de naciones como la Argentina, Colombia, Perú, Venezuela, Chile y Puerto Rico. Esta diversidad geográfica garantiza la representatividad de distintos acentos y pronunciaciones, lo cual es fundamental en un contexto lingüístico tan variado como el hispano. Este dataset fue manualmente chequeado y es de buena calidad acústica, además cada audio tiene catalogado su género, hablante, país y su transcripción.

El segundo dataset utilizado fue el "West Point Heroico Spanish Speech" (WPHSS)[19] de John Morgan que consiste en un conjunto de audios recolectados por una academia militar mexicana y sus hablantes, tanto del género masculino como del femenino, son de ese país. De este dataset se utilizó el subconjunto Heroico con 12.000 audios.

El tercer dataset que se utilizó fue el "CML-TTS" [20] de Oliveira *et al.* que consiste en un corpus de habla tomado de audio libros en distintos idiomas, se seleccionó los audios de habla hispana y un subconjunto de 168.525 audios (443 horas) grabado por 77 hablantes y si bien se tiene registros que 42 son mujeres y los restantes 35 hombres, no se tiene identificado cuál es cuál en las grabaciones y tampoco su país de origen. Mientras que la transcripción de estos audios sí se encuentra disponible.

4.1.2. Datasets de respuestas al impulso (IR)

El cuarto dataset es de respuestas al impulso (IR) procedente del MIT IR Survey [21] de Traer y McDermott. Su estudio cuenta con grabaciones de 271 ubicaciones distintas (1 grabación por ubicación) tales como gimnasio, supermercado, bosque, restaurante, etc. Se decidieron cuáles fueron esos ambientes monitoreando los lugares que frecuentaban siete voluntarios de su trabajo durante una semana. Este conjunto de datos es crucial para simular entornos acústicos realistas y evaluar la robustez del modelo frente a condiciones de grabación variables.

4.1.3. Datasets de ruido ambiente

Estos conjuntos proporcionan una amplia variedad de ruidos de fondo que pueden encontrarse en entornos cotidianos, lo que enriquece la capacidad del modelo para adaptarse a condiciones reales.

El quinto dataset fue de ruidos proveniente de "MUSAN: A Music, Speech, and Noise Corpus" [22] de Snyder *et al.* Se utilizaron ruidos provenientes del conjunto *free-sound* que es de dominio público y cuenta con aproximadamente seis horas de ruidos varios.

El sexto dataset fue TUT [23] de Mesaros y Heittola. Cuenta con 15 ambientes distintos de ruido ambiente, cada ambiente con 52 minutos de grabación para un total de 13 horas totales. Algunos de los ambientes presentes son el subterráneo, una oficina, un auto, el parque, entre otros.

4.2. Partición de datos

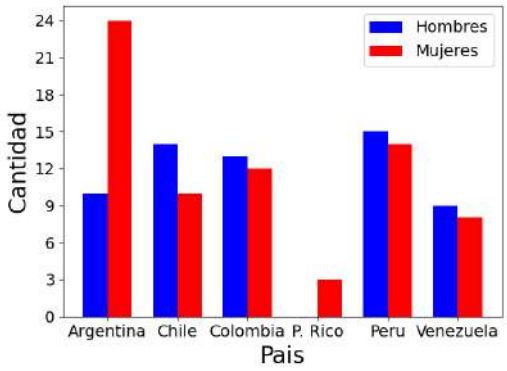
Dado que, los datasets de habla (Heroico, CML-TTS y Crowdsourced) tienen diferente calidad de audio, se utilizaron algunos para pre-entrenamiento y otros para fine-tuning. El primer y el segundo dataset se utilizaron para el pre-entrenamiento debido a que la calidad del audio era inferior a la deseada. Sin embargo, al tener más variedad y mayor volumen enriquecen al entrenamiento. El tercer dataset cuenta con una calidad superior a los demás y, a su vez, tiene catalogado el género, el país y la transcripción. Por estas razones, se lo eligió para ser parte del fine-tuning y validación. Esta decisión genera que el modelo pueda esforzarse más en mejorar las minuciosidades del audio sintetizado. Además, al estar catalogado se puede tener un mayor detalle del desempeño en cada subgrupo (género o país).

Para garantizar una evaluación precisa y confiable del modelo, se realizó una cuidadosa partición de los datos en tres conjuntos distintos: entrenamiento, validación y prueba. Cada conjunto de datos (habla, ruido e IR) se dividió en estos tres subconjuntos. Es importante destacar que se tuvo un especial cuidado en reservar hablantes que nunca antes habían sido vistos por el modelo para los conjuntos de validación y prueba. Como los datasets son desbalanceados esto fue una tarea compleja, ya que se busca tener representatividad de todos los géneros y de todos los países en cada uno de los 3 conjuntos de datos, sin mezclar audios provenientes de un mismo hablante, y manteniendo la proporción 80 %, 10 % y 10 % de entrenamiento, validación y test. La Figura 4.2 muestra cómo se mantuvo la misma proporción en todos los conjuntos.

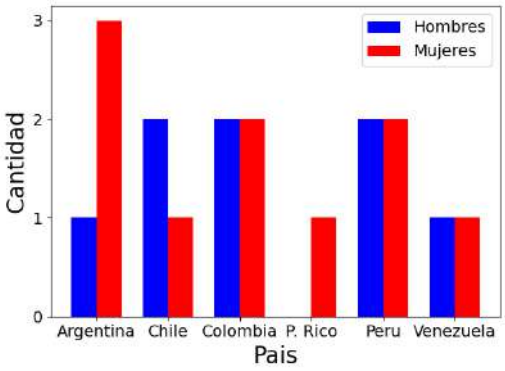
Del mismo modo, los ruidos provenientes de lugares nuevos fueron reservados para el conjunto de validación, asegurando así la generalización del modelo a escenarios no vistos durante el entrenamiento. Esta estrategia de partición contribuye a la fiabilidad y validez de los resultados obtenidos.

4.3. Limpieza y preparación de datasets

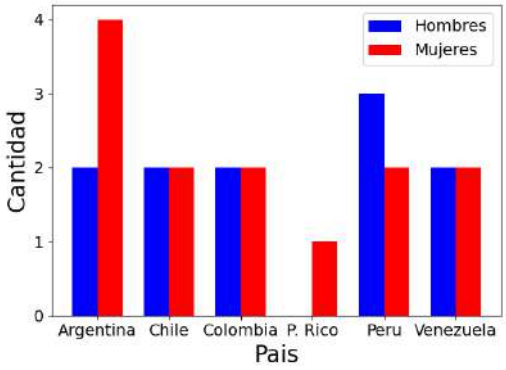
La gestión de datos es un paso crítico para garantizar el buen desempeño de cualquier modelo de aprendizaje automático. Al entrenar modelos con datos que son irrelevantes o que presentan poco valor informativo, se corre el riesgo de enfrentar una disminución en el rendimiento general de los modelos [24]. Ante este desafío, es crucial implementar



(a) Entrenamiento (Fine-tuning).

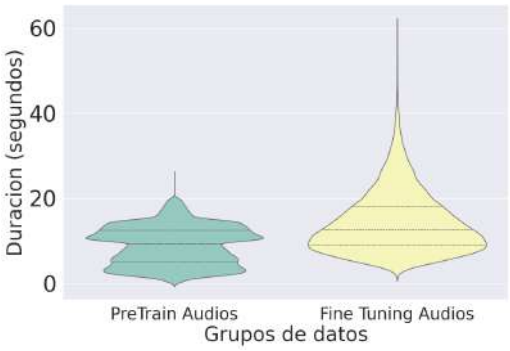


(b) Validación.

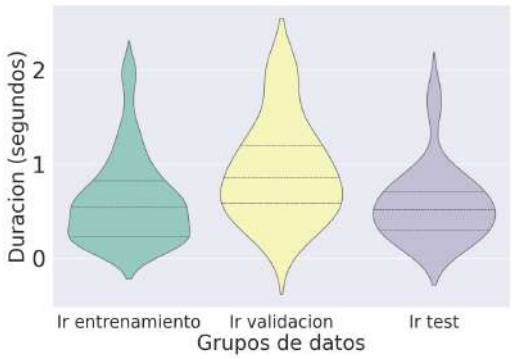


(c) Test.

Fig. 4.2: Cantidad de hablantes de cada país y género según conjunto de datos.



(a) Audios



(b) Respuestas al impulso

Fig. 4.3: Longitud en segundos de cada grupo de audios presentes.

medidas de limpieza de datos que nos permitan eliminar o minimizar el impacto de estos datos irrelevantes en nuestros modelos.

En el caso de los conjuntos de datos de audio y, específicamente, los conjuntos de datos de voz que se utilizaron para este trabajo, se encontró que muchos de los audios incluidos en ellos tenían varios segundos iniciales o finales en los que no se emitía ningún sonido. Este silencio inicial y final en los audios no aporta ninguna información valiosa para la tarea

Etapa	Dataset habla	Ruidos usados	IR
Pre-entrenamiento	(WPHSS 100 %) (CML-TTS 100 %)	(TUT 80 %) (MUSAN 100 %)	(MIT IR 80 %)
fine-tuning	CLAS 80 %		
Validación	CLAS 10 %	(TUT 10 %)	(MIT IR 10 %)
Testing	CLAS 10 %	(TUT 10 %)	(MIT IR 10 %)

Tab. 4.1: Datasets utilizados en cada etapa del modelo.

de aprendizaje de los modelos, y podría, de hecho, introducir ruido en los entrenamientos.

Como solución a este problema, se decidió implementar una estrategia de recorte de estos silencios. La metodología fue bastante sencilla: se decidió recortar todo el principio y el final de los audios en los que el valor absoluto de la onda era inferior a 0.1. De este modo, conservamos solo el intervalo central de cada audio, que es donde se encuentra la información realmente relevante. Si bien es una solución *ad hoc*, es un procedimiento que dio resultados razonables.

Es estándar trabajar con frecuencia de muestreo a 16.000 Hz y permite comparar resultados con otros trabajos, además como es una arquitectura convolucional puede adaptarse a un muestreo arbitrario. Otros trabajos como el de Feng *et al.* [25] mencionan que, para lograr una mejora en la calidad de audio, se puede up-muestrear a 44.100 Hz con una red neuronal con el fin de que este nuevo muestreo permita mejorar la percepción del habla.

4.4. Data Augmentation

4.4.1. Generación de audio con ruido ambiente

En primer lugar, se implementó la adición de ruido al audio original. Este proceso simula condiciones realistas en las que el habla puede estar contaminada por ruido ambiente. Se incorporó una cantidad aleatoria de decibeles para simular entornos más ruidosos que otros. Siendo que los ruidos son variados y de amplitud variable, se los normalizó antes de combinarlos con la grabación.

4.4.2. Ruido blanco

Luego de agregar este ruido ambiental, se probó también los efectos de sumarle ruido blanco con algún coeficiente. Sin embargo, se terminó optando por no agregar este tipo de ruido al entrenamiento, ya que generaba que el modelo se enfoque demasiado en promediar valores cercanos generando que las ondas de alta frecuencia desaparecieran, lo cual empeora la calidad del audio. El modelo está enfocado en quitar ruido ambiental y señales no estacionarias como el ruido de una ambulancia.

4.4.3. Convolución en Modo Completo

Posteriormente, se aplicó la técnica de convolución en modo completo. Esto implica una operación matemática que combina la señal de audio con una respuesta al impulso (IR), emulando así la respuesta acústica de diferentes ambientes. Este paso es esencial para

entrenar al modelo en condiciones variadas y garantizar su capacidad de adaptación a distintos contextos.

4.4.4. Recorte y selección de audio

Por las limitaciones de hardware tales como falta de memoria en la GPU, se seleccionan siempre 2 segundos de cada audio ya procesado (remuestreado y quitado el silencio del comienzo). En las etapas de entrenamiento (pre-entrenamiento y fine-tuning) se seleccionan 2 segundos al azar de cada audio. Esto permite que en distintos épocas se escuchen distintas partes del audio trayendo mayor riqueza y volumen de datos al entrenamiento.

Mientras que en validación para hacer los resultados más comparables se decidió optar por siempre usar los primeros 2 segundos.

4.4.5. Enfoque dinámico en la intensidad del ruido

A medida que la red neuronal progresaba en su entrenamiento y mejoraba en la optimización de la función de pérdida, se incrementó gradualmente la dificultad del ruido introducido. Este enfoque es conocido como curriculum learning [26] y ha mostrado tener efectividad en acelerar el entrenamiento. Se incrementa no solamente la proporción de ruido en relación con la señal sino que también la reverberación. La reverberación se aumenta como combinación lineal de audio con reverberación sumado a audio sin reverberación. Para ver desarrollo y resultados comparativos ver experimento 1.

5. EXPERIMENTOS

Explorar el espacio de hiperparámetros en la construcción de modelos puede ser un desafío considerable, dada su alta dimensionalidad. La identificación de la combinación óptima de valores para el entrenamiento de modelos no es un proceso sencillo y puede implicar un tiempo considerable, especialmente en modelos que requieren largos períodos para obtener resultados significativos.

Es por eso que se empleó una estrategia tipo golosa (*greedy*) de exploración de hiperparámetros, en la que se realizaron múltiples experimentos en los cuales en cada experimento solo se variaba un hiperparámetro a la vez.

Esto puede traer resultados subóptimos al ser una estrategia muy local, porque es posible que un hiperparámetro A con valor X no traiga beneficios, un hiperparámetro B con valor Y tampoco, pero A y B con valores X e Y sí los traiga.

Los experimentos realizados exploran la forma en la que se entrena el modelo a nivel cómo y cuándo se ensucian los datos de entrada, la tasa de aprendizaje, la arquitectura de la red, los conjuntos de datos a utilizar y la combinación de pesos de distintas funciones de pérdida para maximizar el rendimiento de las métricas objetivas. Estos últimos experimentos con métricas objetivas son muy importantes, ya que al estar modificando los pesos de las funciones de pérdida cambia el valor de error del modelo pero no necesariamente la calidad de las predicciones. Los experimentos realizados fueron:

1. *Curriculum learning*: exploración de cómo ajustar gradualmente el nivel de dificultad del modelo durante la etapa de entrenamiento puede mejorar su eficacia.

2. Variación de capas en la WaveNet: investigación de la influencia de la profundidad de la red en su rendimiento.

3. Tasa de aprendizaje: análisis del impacto de la velocidad de aprendizaje y si el decaimiento de la tasa mejora el rendimiento.

4. Peso de la función de pérdida del mel-espectrograma: pruebas sobre cómo la relevancia atribuida a la pérdida del mel-espectrograma influye en el rendimiento del modelo.

5. Peso de la función de pérdida de la amplitud: evaluación de cómo el peso otorgado a la pérdida de amplitud altera el rendimiento del modelo.

6. Peso de la función de pérdida proporcional: examinación de la eficacia del modelo tras modificar el peso atribuido a las pérdidas proporcionales.

7. Pre-entrenamiento y fine-tuning: Comparación de resultados de entrenamiento utilizando distintos conjuntos de datos.

8. PostNet vs. WaveNet: Prueba sobre utilización de distintas componentes de la arquitectura y su etapa de activación.

5.1. Experimento 1 - Curriculum learning

El primer experimento se centra en el concepto de Curriculum Learning en el entrenamiento de modelos. El Curriculum Learning es una estrategia de entrenamiento que implementa una adición gradual de dificultad al modelo a medida que avanza el entrenamiento. El objetivo de este experimento es determinar el impacto y la efectividad de esta estrategia en la mejora del rendimiento del modelo.

Para llevar a cabo este experimento, se realiza una comparación entre dos corridas. La primera es una corrida Baseline, en la que se mantiene constante el nivel de dificultad a lo largo del entrenamiento. En este caso, esta dificultad es equivalente a la del proceso de validación.

La segunda corrida incorpora un incremento progresivo de la dificultad en relación con la cantidad de datos de audio que el modelo ha procesado previamente. Vale la pena destacar que este ajuste de dificultad se realiza independientemente del rendimiento actual del modelo en las fases de entrenamiento o validación.

El incremento en la dificultad se estructura en tres etapas distintas a lo largo del entrenamiento. En la primera etapa, que abarca hasta 6600 audios, no se introduce ruido a la entrada del modelo. En esencia, el modelo tiene la tarea de predecir la entrada sin ninguna alteración.

A partir de 6600 audios y hasta 100,000 audios, la etapa segunda, se introduce gradualmente más ruido a la entrada del modelo. Este incremento es lineal hasta que finalmente alcanza el nivel de dificultad correspondiente a los datos de validación.

El proceso de actualización de la dificultad se realiza cada 9,000 audios aproximadamente. Al final del proceso, el nivel de dificultad final (DF) alcanzado es un ruido de 0.3 y una reverberación de 0.35.

Nombre corrida	Baseline	Con curriculum learning
Dificultad entrenamiento	DF	Progresiva
Dificultad validación	DF	DF
PESQ	1.698	**1.736**
SNR	6.201	**6.258**
SRMR	8.534	**8.570**
STOI	0.840	**0.857**
Error val	0.368	**0.364**

Tab. 5.1: Resultados de experimento curriculum learning

El resultado es muy claro y es el esperado, podría ser porque el curriculum learning permite una progresión natural, que lleva al modelo a aprender patrones más fácilmente y construir una buena representación de los datos. Además, al comenzar con casos más sencillos, se mejora la estabilidad del proceso de entrenamiento, evitando problemas como el estancamiento temprano.

5.2. Experimento 2 - Distinta cantidad de capas en la WaveNet

Este experimento se centra en la variación en la cantidad de capas que componen la WaveNet, este hiperparámetro indica la dilatación máxima posible. En el trabajo de referencia, se utilizan 20 capas divididas en dos pilas residuales, es decir, diez capas por pila. Esto sería que tiene las siguientes dilataciones: 1, 2, 4,...512.

Sin embargo, este nivel de dilatación podría ser insuficiente, particularmente en un entorno donde la reverberación es muy prolongada. Por otro lado, también podría ser excesivo, y se podrían estar empleando parámetros innecesarios que el modelo eventualmente ignora durante las convoluciones.

Es por eso que se realizó un experimento en el que la WaveNet tenía 8 capas por pila y otro donde tenía 14 capas por pila, utilizando como base de comparación el original propuesto con 10 por pila.

Nombre corrida	Baseline (20 capas)	WaveNet 28 capas WaveNet	WaveNet 16 capas
PESQ	1.698	1.629	**1.737**
SNR	6.201	6.274	**6.352**
SRMR	8.534	8.50	**8.679**
STOI	0.840	0.856	**0.865**
Error val	0.368	0.365	**0.357**

Tab. 5.2: Resultados de experimento de distinta cantidad de capas en WaveNet.

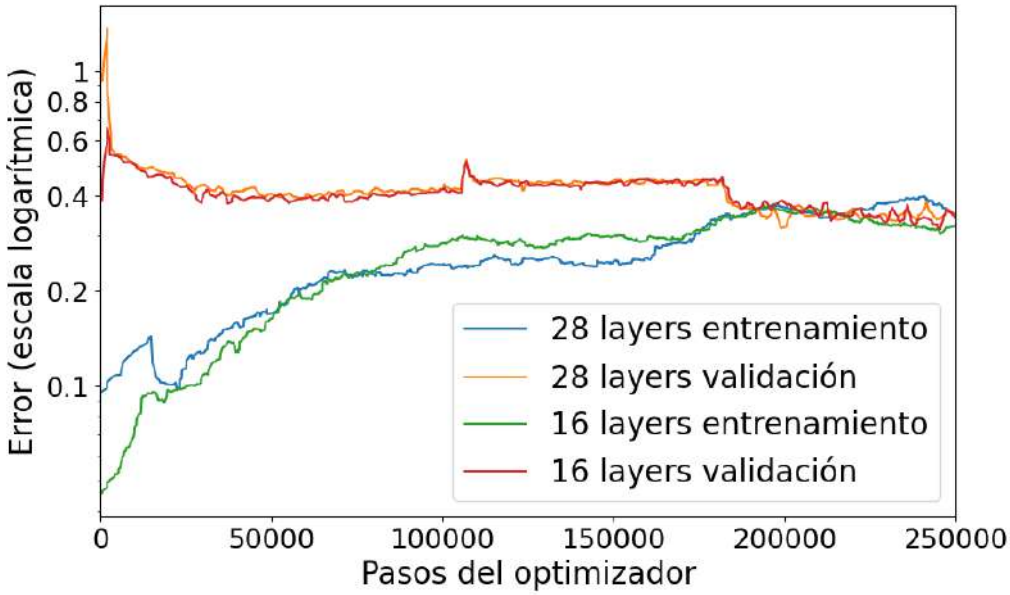


Fig. 5.1: Error total en función de los pasos del optimizador, en este caso el error es solo la L1. Visualización con suavización de valores.

Los resultados, presentados en la Tabla 5.2, son inesperados. Normalmente, se asumiría que una mayor cantidad de parámetros conduciría a un mejor rendimiento, con un mayor riesgo de sobre-ajuste. Sin embargo, los datos de validación muestran que el modelo de 16

capas supera al de 28 capas, tanto en la validación final como en la curva de aprendizaje. Esto sugiere que los parámetros adicionales añadidos por la profundidad en el modelo de 28 capas no están siendo aprovechados del todo.

Estos hallazgos han sido visualizados en la Figura 5.1, que representa la función de pérdida total en función del tiempo.

En cuanto a los otros patrones identificados en las curvas de aprendizaje, destacamos que el aumento observado hasta los 100K es producto del curriculum learning que incrementa la dificultad de limpiar el audio (curriculum learning no se aplica a los datos de validación, que siempre mantienen la dificultad final).

Notamos también un impacto en los cambios de etapa: en el paso 105K el modelo activa la PostNet y añade nuevos parámetros no entrenados, causando que momentáneamente la métrica se empeore. En 185K, al finalizar el pre-entrenamiento y comenzar el fine-tuning, se observa un incremento en el rendimiento de validación. Esto sugiere que el modelo, al trabajar con datos de fine-tuning más similares a los datos de validación, logra eliminar minucias de audio que antes no conocía con los datos de menor calidad. En cuanto a por qué la métrica de entrenamiento empeora con fine-tuning puede deberse a la sencillez relativa de los audios de pre-entrenamiento en comparación con los de fine-tuning. Sin embargo, esto último no está confirmado.

5.3. Experimento 3 - Tasa de aprendizaje

Este experimento analiza la tasa de aprendizaje, que es sabido que es el hiperparámetro más importante [27]. En particular, este experimento se centra en el decaimiento (*decay*) de la tasa de aprendizaje, un método en el que la tasa de aprendizaje se ajusta con el tiempo.

Se sabe que este método puede acelerar el entrenamiento del modelo y, como se ha demostrado recientemente, también puede aumentar la capacidad del modelo para generalizar patrones más complejos [28]. Permite que el entrenamiento converja de manera más rápida en comparación con una tasa de aprendizaje pequeña, y evita la divergencia que podría ocurrir con una tasa de aprendizaje grande, al ubicar óptimos locales más eficientemente.

El experimento compara tres diferentes condiciones: una corrida base con una tasa de aprendizaje constante y relativamente pequeña; una corrida con un decaimiento exponencial de la tasa de aprendizaje rápida (la tasa de aprendizaje se multiplica por 0.5 cada 40.000 audios) y una tercera corrida con un decaimiento exponencial de la tasa de aprendizaje lenta (la tasa de aprendizaje se multiplica por 0.85 cada 20.000 audios).

En vista de estos resultados (Tabla 5.3), es evidente que si la tasa de aprendizaje es constantemente baja no permite el aprendizaje del modelo a los datos, mientras que si decae muy rápido sucede algo similar. Es por esto también que la curva de aprendizaje en la Figura 5.3 de aquellas corridas con tasa más baja cuando se prende la PostNet tardan más tiempo en ajustar sus parámetros para que la PostNet pase a ser útil en la limpieza.

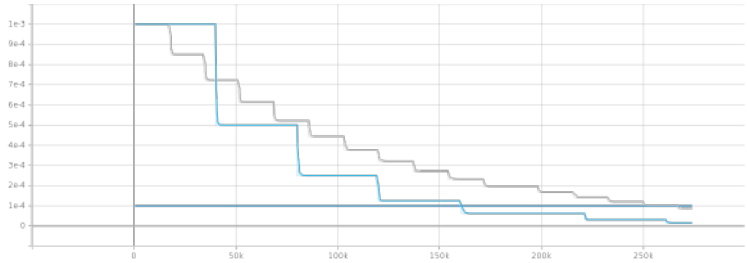


Fig. 5.2: Tasas de aprendizaje de cada uno de los tres entrenamientos en función del tiempo. En azul aprendizaje constante lento, en gris decaimiento lento y en celeste decaimiento rápido.

Nombre corrida	Baseline (constante)	Decaimiento lento	Decaimiento rapido
Tasa de aprendizaje inicial	0.0001	0.001	0.001
Decaimiento de aprendizaje	-	0.85	0.5
Cantidad de pasos entre decaimiento	-	20000	40000
PESQ	1.663	**1.685**	1.632
SNR	5.509	**5.791**	4.918
SRMR	8.796	8.506	**8.808**
STOI	0.817	**0.831**	0.815
Error val	0.386	**0.384**	0.405

Tab. 5.3: Resultados de experimento de tasa de aprendizaje.

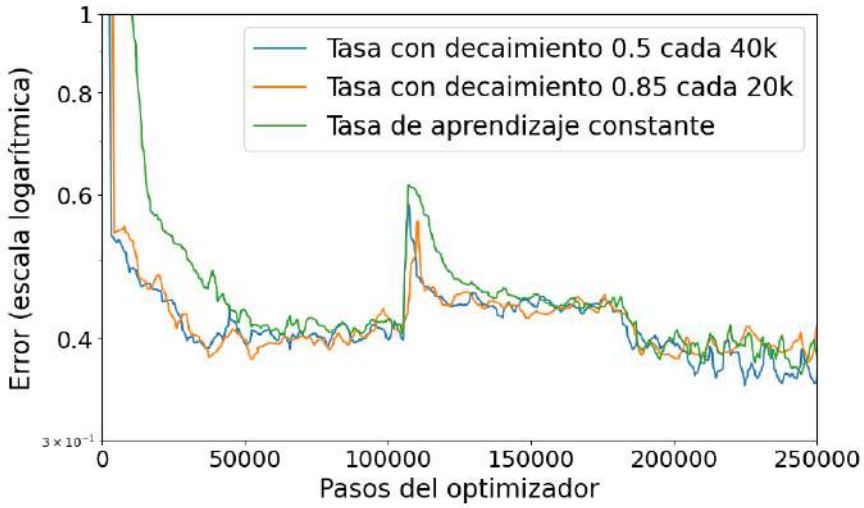


Fig. 5.3: Función de pérdida L1 validación de las 3 corridas del experimento 3. 105k se activa PostNet, 185k empieza fine-tuning. Visualización con valores suavizados.

5.4. Experimento 4 - Peso de función de pérdida mel espectrograma

El objetivo de este experimento es determinar la incorporación de una función de pérdida basada en el mel espectrograma 3.4.2 que puede aportar valor a la mejora del modelo.

Para este experimento evaluamos tres escenarios: una configuración de línea de base sin

incorporación de la función de pérdida de mel espectrograma; otra con la incorporación de dicha función, pero sin penalización adicional en frecuencias agudas y finalmente, una con la incorporación de dicha función con penalización extra en las frecuencias agudas. El peso asignado a las configuraciones con la función de pérdida de mel espectrograma se estableció en 0.004.
CPESP = Con pérdida de espectrograma sin penalización.
CPEPE = Con pérdida de espectrograma y penalización extra

	Baseline	CPESP	CPEPE
Peso de mel espectrograma	0	0.004 para ambos	0.004 para ambos
Factor de penalización	-	0	2
PESQ	**1.696**	1.492	1.503
SNR	**6.311**	2.819	2.6268
SRMR	8.481	**8.905**	8.883
STOI	0.861	**0.870**	0.865
Error val (no comparable)	0.365	1.084	1.070

Tab. 5.4: Resultados de experimento de función de pérdida de mel espectrograma.

Teniendo en cuenta que las métricas objetivas PESQ, STOI y SRMR se calculan utilizando espectrogramas, es esperable que en estas tres métricas las corridas con peso en la pérdida de mel espectrograma les fuera mejor. Sin embargo, se observa en 5.4 que para PESQ no es el caso y por un buen margen, en vista de esto no queda en claro si es una buena idea incorporar esta función de pérdida, ya que las ganancias más marginales en las otras dos métricas (SRMR y STOI) podrían no justificar adoptar este error.

5.5. Experimento 5 - Peso de función de pérdida amplitud

Para comprobar si la función de pérdida propuesta en 3.4.4 (que penaliza cuando el audio limpio tiene amplitud mayor a 0.1 con el ECM) aportaba valor al entrenamiento se realizó una corrida con un poco de peso en esta función. El peso que se le asigna a la corrida de este experimento fue elegido para que los órdenes de magnitud de las distintas funciones de pérdida sean iguales.

	Baseline	Con pérdida amplitud
Peso de pérdida de amplitud	0	300
PESQ	**1.698**	1.697
SNR	**6.201**	6.109
SRMR	**8.534**	7.555
STOI	0.840	**0.871**
Pérdida validación total (no comparable)	0.368	1.018

Tab. 5.5: Resultados de experimento de pérdida de amplitud.

En la Tabla 5.5 se observa que si bien no aporta valor en la mayoría de las métricas objetivas, la inteligibilidad se ve afectada positivamente. STOI es una de las métricas

que compara el audio predicho con el real en el dominio de frecuencias. Más análisis es requerido para poder entender de dónde viene la diferencia al agregar esta función de pérdida entre STOI y las otras métricas objetivas.

5.6. Experimento 6 - Peso de función de pérdida proporcional

En este experimento se busca determinar si la función de pérdida definida en 3.4.3 que pesa las diferencias relativas a la amplitud del audio limpio traía un beneficio a nivel entrenamiento.

Nuevamente para comparar entre corridas se toma un Baseline en el que no se aplica esta función de pérdida, más aún, solo la L1 participa del error total.

	Baseline	Con función de pérdida proporcional
Peso de pérdida de proporcional	0.000	0.0005
PESQ	**1.698**	1.468
SNR	**6.201**	3.273
SRMR	**8.534**	8.467
STOI	**0.840**	0.747
Error val (no comparable)	0.368	0.522

Tab. 5.6: Resultados de experimento con función de pérdida proporcional.

Como se puede notar en la Tabla 5.6, los resultados no fueron positivos para la función propuesta. Existe la posibilidad de que el motivo por el cual sucede esto es que haya valores extremos cuando el audio limpio tiene amplitud cercana al cero.

5.7. Experimento 7 - Pre-entrenamiento y fine-tuning

En este experimento se busca explorar el impacto de diferentes conjuntos de datos en el proceso de entrenamiento: pre-entrenamiento, fine-tuning o una combinación de ambos.

Al examinar las curvas de aprendizaje, se observó que al pasar a fine-tuning, el aprendizaje mejoraba considerablemente, lo que se reflejaba tanto en la disminución de la función de pérdida como en la mejora de las métricas objetivas.

Los datos de fine-tuning son de mayor calidad, mientras que los datos de pre-entrenamiento aportan un mayor volumen de muestras. Una hipótesis sobre la mejora notada durante la fase de fine-tuning sugiere que el modelo es capaz de enfocarse en detalles más sutiles del audio cuando dispone de datos de calidad superior.

Otra explicación planteada sobre el aumento de rendimiento durante el fine-tuning reside en la procedencia del conjunto de validación. Dado que este conjunto es tomado del mismo dataset que el usado en fine-tuning, y que el de pre-entrenamiento proviene de un dataset distinto, es posible que el modelo mejore su desempeño en la validación debido a la similitud de condiciones acústicas, evitando conflictos de dominio desigual (*domain mismatch* en ingles) que pueden darse durante la etapa de pre-entrenamiento.

Sobre la base de esto se quería ver cómo era la influencia de distintos conjuntos de datos y cómo afectan el rendimiento del modelo. En concreto, se buscó observar el impacto de excluir uno de los conjuntos de datos - ya sea el de pre-entrenamiento o el de fine-tuning.

	Baseline (ambos conjuntos)	Solo fine-tuning	Solo pre-entrenamiento
PESQ	1.698	**1.758**	1.587
SNR	6.201	**6.405**	3.455
SRMR	8.534	8.283	**9.273**
STOI	0.840	**0.892**	0.837
Error val	0.368	**0.363**	0.460

Tab. 5.7: Resultados de experimento de conjuntos de datos.

Al concluir estos experimentos se ve cómo no utilizar pre-entrenamiento fue beneficioso a nivel general por sobre utilizar ambos. Hay varias causas que pueden estar impactando en este resultado:

1) Debido a la existencia de un decaimiento en la tasa de aprendizaje, si inicialmente se procesan datos de inferior calidad, cuando se tiene una mayor capacidad de aprendizaje, al llegar a los datos de superior calidad la tasa de aprendizaje puede estar ya cercana a cero. Esto resulta en un aprovechamiento subóptimo de los datos de alta calidad.

2) Al sumar a los audios de pre-entrenamiento podría existir una discrepancia en términos de la naturaleza del habla en comparación con los de evaluación (dominio desigual). Esta disparidad podría estar afectando negativamente el rendimiento del modelo. También explica como en fine-tuning que tiene el mismo dominio que validación no sucede eso, lo que no explica es por qué pre-entrenamiento + fine-tuning no supera a solo fine-tuning.

En cuanto al motivo por el cual el entrenamiento exclusivo con pre-entrenamiento resulta en una mejor métrica de SRMR (la métrica objetiva que mide la reverberación), no se lo tiene muy claro. Es importante destacar que las respuestas al impulso utilizadas en todos los tipos de entrenamiento son las mismas, sin tener en cuenta el conjunto de datos de habla seleccionado.

5.8. Experimento 8 - PostNet vs. WaveNet

En vista de que agregar la PostNet tuvo un impacto muy positivo en el trabajo de referencia, y de que los autores solo mencionan que lo activaban únicamente después de 500k pasos, surgió la pregunta de cuánto valor agrega hacerlo de manera diferida a la WaveNet.

La lógica detrás de activar la PostNet luego de haber entrenado la WaveNet es que hay menos parámetros para ir optimizando cuando todavía no es necesario y porque la PostNet se ocupa de limpiar los artefactos propios que genere la WaveNet.

Es por eso que en este experimento se compara la corrida Baseline en la que se activa

luego de 105k pasos, un entrenamiento en la cual siempre esta activada desde el inicio y una última corrida en donde nunca se activa.

	Baseline	PostNet desde siempre	No PostNet
Activación de PostNet	105000	0	-
PESQ	1.698	1.729	**1.817**
SNR	6.201	**6.446**	6.378
SRMR	8.534	**8.663**	8.242
STOI	0.840	0.889	**0.913**
Error val	0.368	**0.356**	0.374

Tab. 5.8: Resultados de experimento PostNet vs. WaveNet.

Viendo los valores de la Tabla 5.8 es sorpresivo como activar la PostNet desde un inicio tiene mejores resultados que hacerlo de manera diferida, dado que la WaveNet no tuvo una etapa de entrenamiento dedicada. De hecho, el resultado esperado era que el Baseline supere a las otras dos corridas. Una explicación para esto podría ser un decaimiento de la tasa de aprendizaje tan grande que cuando se activa la PostNet ya es demasiado baja como para actualizar considerablemente los parámetros de la PostNet. Sin embargo, como en las tres corridas se tiene iniciado la tasa de aprendizaje en 1e-3 y con decaimiento de 0.85 cada 20k cuando se active la PostNet en el Baseline tendría de tasa en 4.4e-4, solo un 56 % menos, la diferencia de tasa no debería ser tan relevante.

Mientras que los resultados de no activar la PostNet muestran que el grueso de la limpieza sucede en la WaveNet. En cualquier caso no queda del todo claro cuánto es el valor que aporta la PostNet en este escenario.

5.9. Resultados en datos separados

En esta sección se utilizó para todos los resultados el conjunto de datos de testeo que nunca había sido utilizado previamente, ni durante el entrenamiento ni en la validación para comparar entre distintas corridas de entrenamiento.

El único modelo utilizado en estos resultados fue entrenado sobre la base de hallazgos en los experimentos previos. Se utilizó una WaveNet con dos pilas (*stacks*) residuales y 10 capas por pila (20 capas totales) y 128 canales. Como no quedó absolutamente claro si agregar la PostNet en este caso de uso traía beneficios en las métricas objetivas 5.8, se optó por no utilizar la PostNet en el entrenamiento. Además sobre la base de los resultados de la Tabla 5.7 sobre los distintos conjuntos de datos, en el entrenamiento el modelo únicamente vio los datos de fine-tuning. Se optó por usar las pérdida de Mel espectrograma no pesado por frecuencias más agudas y la L1, ya que mostraron ser beneficiosas juntas para el entrenamiento a nivel métricas objetivas, excluyendo las otras dos funciones de pérdida propuestas.

5.9.1. Tiempos de inferencia

Cuando se examinan los resultados y se realizan inferencias con una red convolucional, se tiene la ventaja de no tener que ajustar la longitud de los audios a una medida predeterminada. Este ajuste si es necesario durante la fase de entrenamiento del modelo cuando la memoria de la GPU impone limitaciones.

Se observa que el factor de tiempo real es de 264, por lo tanto el modelo es 264 veces más rápido que la duración del audio de entrada. Este factor se calculó promediando el tiempo de inferencia y promediando la duración de los audios.

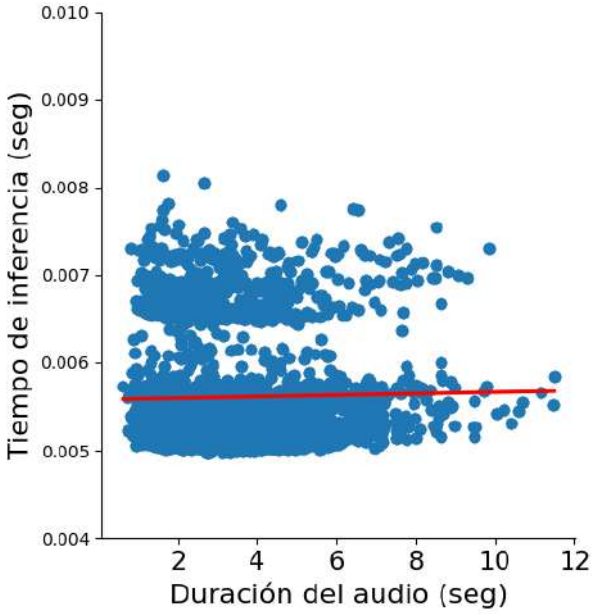


Fig. 5.4: Tiempo de inferencia en función de la duración del audio. Tiempos calculados con la misma GPU utilizada en entrenamiento (3.7) . En rojo la recta que minimiza el ECM.

La Figura en 5.4 nos muestra que el tiempo de inferencia es altamente agnóstico a la duración del audio, ya que el coeficiente de pendiente de la regresión es prácticamente cero.

5.9.2. Resultados con distinta cantidad de ruido

También se realiza un análisis del rendimiento del modelo en diferentes condiciones. Una situación en la que el nivel de ruido es mayor al nivel de ruido con el cual el modelo fue entrenado y otra en la que es menor a la cantidad de ruido en relación con la dificultad del entrenamiento. El nivel de ruido agregado en entrenamiento fue de 0.3 y de reverberación 0.35. Este análisis sirve para ver cómo se adapta a diferentes circunstancias, además se presenta de referencia las métricas para el audio ruidoso sin ser limpiado por el modelo.

En la Tabla 5.9 se aprecia como en todas las métricas el modelo tiene resultados que mejoran al audio de entrada excepto en STOI que es marginalmente peor.

Nivel de ruido	Nivel de reverberación	Etapa	PESQ	SNR	SRMR	STOI	Error
-	-	Audio limpio	4.643	94.63	9.015	1	0
0.2	0.2	WaveNet	1.639	8.707	7.363	0.926	0.241
0.2	0.2	Audio ruidoso	1.51	-0.3	6.963	0.927	0.708
0.2	0.2	Diferencia (Δ)	+0.129	+9.007	+0.400	-0.001	-0.467
0.3	0.35	WaveNet	1.429	6.756	6.3	0.884	0.302
0.3	0.35	Audio ruidoso	1.323	-2.039	5.945	0.883	0.859
0.3	0.35	Diferencia (Δ)	+0.106	+8.795	+0.355	+0.001	-0.557
0.6	0.65	WaveNet	1.215	3.707	4.554	0.790	0.425
0.6	0.65	Audio ruidoso	1.149	-5.005	4.261	0.788	1.194
0.6	0.65	Diferencia (Δ)	+0.066	+8.712	+0.293	+0.002	-0.769

Tab. 5.9: Resultados de testing del modelo en distintos niveles de ruido.

Además se observa como a medida que se va ensuciando más el audio de entrada (y consecuentemente decaen las métricas), la diferencia entre el error del modelo y del audio de entrada se agranda. Esto puede deberse a que el modelo a pesar de solo haber sido entrenado con un nivel de ruido puede generalizar a ambientes más ruidosos.

Como el ruido incorporado a cada audio es elegido al azar dentro de su conjunto de datos, tanto en entrenamiento como en inferencia, se propuso también estudiar dado un mismo audio cuán variable era su error al agregarle diferentes ruidos. Para esto se fue iterando sobre el conjunto de ruidos de testing y para cada ruido se media la capacidad del modelo de limpiarlo.

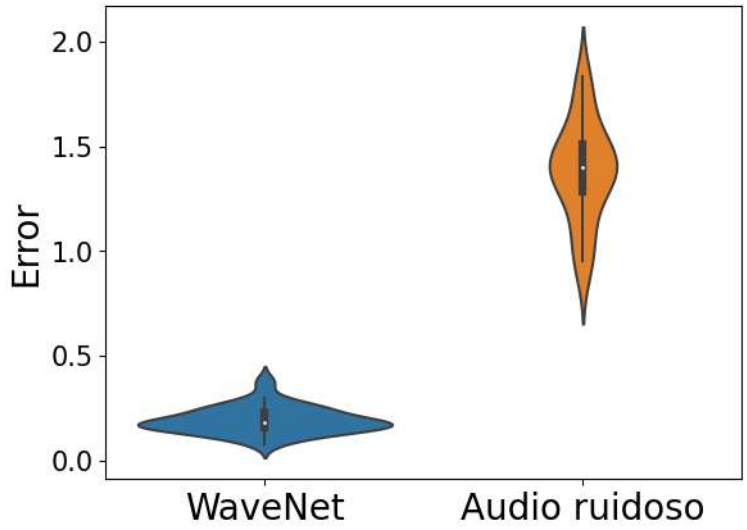


Fig. 5.5: Variabilidad de errores para un único audio tomado al azar y múltiples ruidos del conjunto de testeo.

Este breve resultado no solamente nos dice cuán variable es el ruido dentro del conjunto de testeo sino que también nos muestra como el modelo se adapta a todos esos ruidos tan diversos. En 5.5 se ve como claramente el error del modelo no tiene la misma varianza

que el error del audio ruidoso, esto posiblemente signifique que tiene cierta robustez a las diferentes formas de ensuciar el audio.

5.9.3. Relación entre métricas objetivas y el error

Como el objetivo es hacer limpieza de ruido en habla, la única métrica que realmente es de interés es cuan limpio se siente el audio al escucharlo por un ser humano, es decir, es subjetivo. Como no es posible medir eso cuando se está entrenando un modelo, se miden métricas objetivas. Estas están diseñadas para representar cuan ruidoso es un audio, sin embargo no es posible entrenar al modelo con estas. Es por eso que se utiliza una o varias funciones de pérdida para la optimización en sí.

Entonces sobre la base de esta relación entre funciones de pérdida y métricas objetivas se quiso ver cuan correlacionados linealmente estaban estas mediciones en el conjunto de testing.

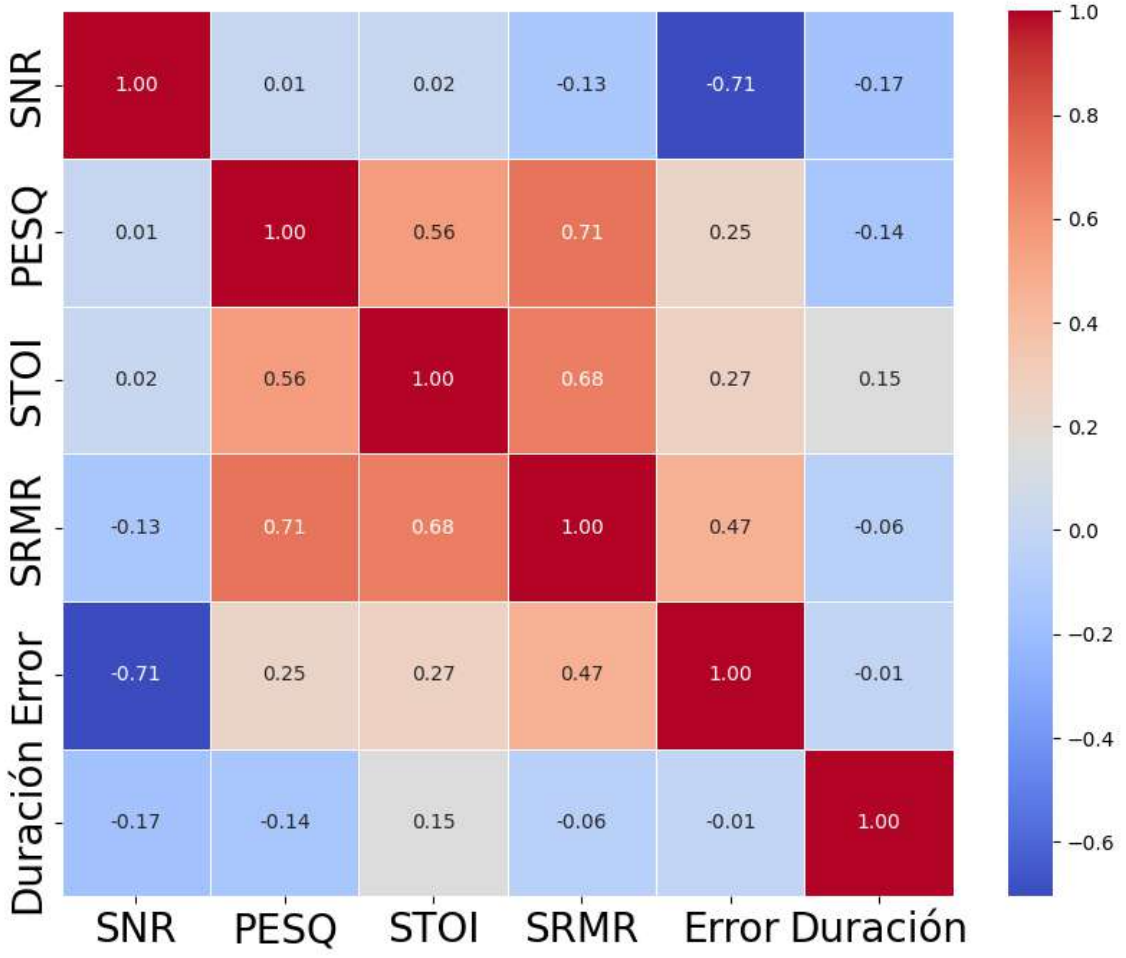


Fig. 5.6: Matriz de correlación lineal entre cada una de las métricas, duración y error.

Idealmente se quisiera en la Figura 5.6 que entre todas las métricas y error haya números lo más extremos posibles, es decir, cerca de -1 o alternativamente cerca de 1. Sin embargo, como cada métrica analiza un aspecto diferente del audio no necesariamente siempre van

de la mano.

Es llamativo como nuestro error tiene una buena relación con todas estas medidas, pero no la tiene con la duración, esto es una característica deseable, ya que no se quiere que tenga peores predicciones para audios más largos. Este buen resultado posiblemente se deba a la invarianza a la traslación de las redes convolucionales.

5.9.4. Resultados para distintos grupos de hablantes

Para poder detectar sesgos, es importante presentar resultados para cada grupo de hablantes que se pueda identificar del conjunto de datos. Como en el conjunto de testing no se tiene información de la edad de los hablantes, pero sí de su país y género, se analiza dividiendo en subgrupos (país, género). Se cuenta con todos los pares excepto hombres de Puerto Rico, que no están presentes en ningún conjunto de datos.

Luego de mirar varias distribuciones de cada métrica objetiva, llamó la atención el gráfico de SRMR en la Figura 5.7. Lo que se observa es que para las mujeres el SRMR tiene mejores resultados que para los hombres. Esta diferencia no solamente es visible a nivel agregado sino que el patrón se repite en cada país. Como el SRMR mide la calidad de audio en condiciones reverberantes, se estudia el motivo por el cual las mujeres podrían tener en sus grabaciones menos reverberación.

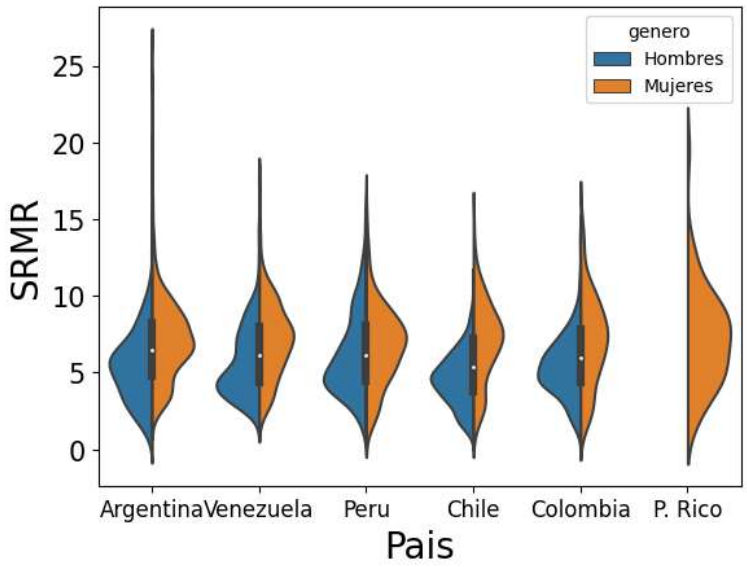


Fig. 5.7: SRMR por país y género

Una posibilidad es que como las mujeres suelen tener un tono de voz más agudo, y como las altas frecuencias pueden ser absorbidas con mayor facilidad por los materiales del entorno, su señal no logra reflejarse y causar reverberación.

Por otro lado, como los resultados en otras métricas objetivas son similares entre ambos géneros, difícilmente tenga que ver con la cantidad de minutos de audio que el modelo en entrenamiento escuchó de mujeres relativa a hombres. Es decir, solo difieren en esta métrica y en ninguna otra.

5.10. Conclusión

En este trabajo se implementó una WaveNet y una PostNet para limpieza de ruido en habla, logrando buenos resultados en las métricas objetivas tomadas. También se realizaron una serie de experimentos destinados a mejorar el entrenamiento del modelo. Los experimentos abarcaron diversos aspectos, incluyendo:

1. La introducción de tres funciones alternativas de pérdida y la evaluación de su impacto en las métricas de rendimiento.

2. La investigación sobre el entrenamiento utilizando diferentes conjuntos de datos, con un enfoque particular en procesos de pre-entrenamiento y fine-tuning. También se consideró el efecto que podría tener la discrepancia entre los dominios del conjunto de datos de entrenamiento y validación.

3. La comparación de diversas arquitecturas para la red generadora, considerando distintos números de capas en la WaveNet, y la inclusión o exclusión de un PostNet. Trayendo como resultado que la WaveNet puede tener una cantidad menor de capas (y luego parámetros) y mantener o aumentar su rendimiento y que la PostNet podría ser activada al comienzo del entrenamiento sin diferir su entrenamiento con la WaveNet.

4. La evaluación de la efectividad del decaimiento de tasa de aprendizaje mostrando que acelera el entrenamiento y al mismo tiempo permite una convergencia de los parámetros del modelo.

5. La implementación del curriculum learning en el entrenamiento demostró conducir a un mejor rendimiento, ya que proporciona una forma más natural y progresiva para que el modelo aprenda la estructura del habla.

También en este trabajo se hace un análisis de las predicciones del modelo en distintas condiciones de ruido y en distintos grupos de hablantes.

Además se aporta el código de manera abierta, en el que se encuentra toda la infraestructura para poder realizar múltiples corridas. Con la cual simplemente cambiando los hiperparámetros de un archivo se tiene la capacidad de explorar distintas arquitecturas para hacer limpieza de ruido y la capacidad de monitorear esas corridas en TensorBoard.

5.11. Trabajos futuros

Como direcciones futuras se puede explorar con más detalle los distintos hiperparámetros del modelo, específicamente de la arquitectura de la WaveNet. Si bien en este trabajo se probó una distinta cantidad de capas en las pilas residuales, los parámetros como el tamaño del kernel, la cantidad de canales o mismo como la cantidad de pilas podrían traer resultados interesantes que permitan tener limpiezas de mejor calidad o con menor costo computacional.

Además se pueden explorar variantes en las que en vez de tener un decaimiento en la tasa de aprendizaje en función de los pasos del optimizador, se tenga un incremento en el tamaño del lote (*batch size*) en función de estos pasos. Esto ha demostrado en otros trabajos ser un estrategia beneficiosa para los entrenamientos [29].

Si se quiere seguir explorando el artículo de referencia entonces también se puede estudiar la efectividad de la pérdida por discriminadores adversarios, ya que en este trabajo se logró implementar en el código, pero por restricciones de tiempo no se realizó un análisis de estos, ni corridas que los incluyan.

En el trabajo original también utilizan técnicas de data augmentation en las respuestas al impulso, y su importancia y efectividad podría ser explorada en futuros trabajos.

Otra exploración posible, por la misma línea que el análisis de rendimiento en cada grupo de hablantes, es estudiar la eficacia del modelo en sacar ruidos cuando el habla proviene de no nativos o de personas de todo el espectro de edades. Incluso se podría ver la efectividad del modelo comparándolo con modelos sencillos como los filtros de wiener.

También otra continuación posible es ver la efectividad del modelo para limpiar audio que luego será trascripto, utilizando la columna de transcripción de los datasets para comparar, donde se podría calcular qué fonemas el modelo limpia de peor manera y estudiar su causa.

Bibliografía

[1] Finkelstein A. Su J, Jin Z. Hifi-gan: High-fidelity denoising and dereverberation based on speech deep features in adversarial networks. *arXiv preprint arXiv:2006.05694*, 2020.

[2] Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alexander Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. In *Arxiv*, 2016.

[3] Jiaqi Su, Zeyu Jin, and Adam Finkelstein. Hifi-gan-2: Studio-quality speech enhancement via generative adversarial networks conditioned on acoustic features. In *2021 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, pages 166–170, 2021.

[4] Bo Li, Peng Qi, Bo Liu, Shuai Di, Jingen Liu, Jiquan Pei, Jinfeng Yi, and Bowen Zhou. Trustworthy ai: From principles to practices. *ACM Computing Surveys*, 55(9):Article 177, September 2023.

[5] Stockham T. Oppenheim A, Schafer R. Nonlinear filtering of multiplied and convolved signals. *IEEE Transactions on Audio and Electroacoustics*, 16(3):437–466, 1968.

[6] Haykin S. Gibson C. Learning characteristics of adaptive lattice filtering algorithms. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 28(6):681–691, 1980.

[7] Malah D. Ephraim Y. Speech enhancement using a minimum-mean square error short-time spectral amplitude estimator. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 32(6):1109–1121, 1984.

[8] Juang B. H. Rabiner, Lawrence R. *Statistical methods for the recognition and understanding of speech.* Encyclopedia of language and linguistics, 2004.

[9] S. R. Park and J. W. Shin. Deep learning based noise reduction for speech signals. In *2015 International Conference on Information and Communication Technology Convergence (ICTC)*, 2015.

[10] F. Weninger, H. Erdogan, S. Watanabe, E. Vincent, J. Le Roux, J. R. Hershey, and B. Schuller. Speech enhancement with lstm recurrent neural networks and its application to noise-robust asr. In *International Conference on Latent Variable Analysis and Signal Separation*, 2015.

[11] Dario Rethage, Jordi Pons, and Xavier Serra. A wavenet for speech denoising. 06 2017.

[12] P. Pandey and C. C. Le. Convolutional recurrent neural networks for speech enhancement. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019.

[13] Q. Hu and P. C. Loizou. Denoising of noisy speech with the transformer network. In *ICASSP 2021-2021*, 2021.

[14] Q. Kong, Y. Cao, S. J. Rennie, and M. Picheny. Wave transformer: An architecture unit for all neural speech processing. *arXiv preprint arXiv:2104.11257*, 2021.

[15] Alex Shenfield and Martin Howarth. A novel deep learning model for the detection and identification of rolling element-bearing faults. *Sensors (Basel, Switzerland)*, 20, 09 2020.

[16] Jonathan Shen, Ruoming Pang, Ron J. Weiss, Mike Schuster, Navdeep Jaitly, Zongheng Yang, Zhifeng Chen, Yu Zhang, Yuxuan Wang, RJ Skerry-Ryan, Rif A. Saurous, Yannis Agiomyrgiannakis, and Yonghui Wu. Natural tts synthesis by conditioning wavenet on mel spectrogram predictions, 2018.

[17] James Traer and Josh H. McDermott. Statistics of natural reverberation enable perceptual separation of sound and space. *Proceedings of the National Academy of Sciences*, 113(48):E7856–E7865, 2016.

[18] A. Guevara-Rukoz, I. Demirsahin, F. He, S.-H.C. Chu, S. Sarin, K. Pipatsrisawat, A. Gutkin, A. Butryna, and O. Kjartansson. Crowdsourcing latin american spanish for low-resource text-to-speech. In *Proceedings of The 12th Language Resources and Evaluation Conference (LREC)*, pages 6504–6513, Marseille, France, May 2020.

[19] J. Morgan. West point heroico spanish speech. Technical report, LDC Catalog No. LDC2006S37, 2006.

[20] F. S. Oliveira, E. Casanova, A. C. Junior, A. S. Soares, and A. R. Galvão Filho. Cml-tts: A multilingual dataset for speech synthesis in low-resource languages. In K. Ekštein, F. Pártl, and M. Konopík, editors, *Text, Speech, and Dialogue*, pages 188–199, Cham, 2023. Springer Nature Switzerland.

[21] J. Traer and J. H. McDermott. Statistics of natural reverberation enable perceptual separation of sound and space. *Biological Sciences*, 113(48):E7856–E7865, 2016.

[22] D. Snyder, G. Chen, and D. Povey. Musan: A music, speech, and noise corpus. *arXiv:1510.08484 [cs.SD]*, 2015.

[23] A. Mesaros, T. Heittola, and T. Virtanen. Tut database for acoustic scene classification and sound event detection. In *2016 24th European Signal Processing Conference (EUSIPCO)*, pages 1128–1132, Budapest, Hungary, 2016.

[24] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2009.

[25] B. Feng, Z. Jin, J. Su, and A. Finkelstein. Learning bandwidth expansion using perceptually-motivated loss. In *ICASSP 2019*, page 606–610. IEEE, 2019.

[26] P. Soviany, R. T. Ionescu, P. Rota, and N. Sebe. Curriculum learning: A survey.

[27] Leslie N. Smith. Cyclical learning rates for training neural networks. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 464–472, 2017.

[28] K. You, M. Long, J. Wang, and M. I. Jordan. How does learning rate decay help modern neural networks? *arXiv:1908.01878 [cs.LG]*, 2019.

[29] Samuel L. Smith, Pieter-Jan Kindermans, and Quoc V. Le. Don't decay the learning rate, increase the batch size. *CoRR*, abs/1711.00489, 2017.

5.12. Apéndices

```
{
    {
    "nameOfRun": "learningRateDecay085_20kT4",

    "hardwareAndSize":{
        "device": "cuda",
        "sample_rate" : 16000,
        "inputSize": 32000,
        "batch_size": 1
     },

    "fileLocations":{

        "directoryBase": "",
        "saveTensorboardLocation" : "extra/registros",

        "preTrainLocations": {
            "csvAudioFile" : "/wavenet/CSV/newCSV/combinadoPreTrain.csv",
            "folderAudioFiles": "/extra/preTrainAudios",
            "folderIRFiles": "/extra/irDivididos/16000/irtrain",
            "csvNoiseFile": "/wavenet/CSV/ruido_train.csv",
            "folderNoiseFiles": "/extra/ruidosDivididos/16000"
        },

       "FineTuneLocations": {
            "csvAudioFile" :  "/wavenet/CSV/newCSV/audiosTrain.csv",
            "folderAudioFiles": "/extra/16000AudioPaises",
            "folderIRFiles": "/extra/irDivididos/16000/irtrain",
            "csvNoiseFile": "/wavenet/CSV/ruido_train.csv",
            "folderNoiseFiles": "/extra/ruidosDivididos/16000"
        },

        "MiniValLocations": {
            "csvAudioFile" : "/wavenet/CSV/newCSV/audiosMiniVal.csv",
            "folderAudioFiles": "/extra/16000AudioPaises",
            "folderIRFiles": "/extra/irDivididos/16000/irval",
            "csvNoiseFile": "/wavenet/CSV/ruido_validation.csv",
            "folderNoiseFiles": "/extra/ruidosDivididos/16000"
        },

        "FullValLocations": {
            "csvAudioFile" : "/wavenet/CSV/newCSV/audiosVal.csv",
            "folderAudioFiles": "/extra/16000AudioPaises",
            "folderIRFiles": "/extra/irDivididos/16000/irval",
            "csvNoiseFile": "/wavenet/CSV/ruido_validation.csv",
```

```
            "folderNoiseFiles": "/extra/ruidosDivididos/16000"
        }


    },

    "noiseConfigs": {
        "trainDifficulty":{
            "maxRuido": 0.0,
            "snr":0,
            "add_impulse_response":true,
            "DificultadReverb":0.0,
            "useDistortions":false,
            "min_snr_coef": 0.75,
            "min_white_noise_coef": 0.1
        },

        "learningCurriculum":{
            "stepsTillDificultyIncrement":[416, 624, 1248, 1876, 2500, 2800, 3124, 3400,
                                           3750, 4000, 4370, 4700, 5000, 5310, 5624, 6250],
            "maxRuidoChange":[0,    0,   0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0],
            "SNRChange":     [0.01, 0.02,0.04, 0.06, 0.08, 0.10, 0.12, 0.14, 0.16, 0.18, 0.20, 0.22, 0.24, 0.26, 0.28, 0.30],
            "ReverbChange":  [0   ,0    ,0.01, 0.05, 0.07, 0.1,  0.12, 0.15, 0.17, 0.2,  0.22, 0.25, 0.27, 0.30, 0.32, 0.35]
        },
        "validationDifficulty":{
            "maxRuido": 0.0,
            "snr":0.30,
            "add_impulse_response":true,
            "DificultadReverb":0.35,
            "useDistortions":false,
            "min_snr_coef": 0.75,
            "min_white_noise_coef": 0.1
        }
    },


    "wavenetParams":{
        "layer_size": 10,
        "stack_size": 2,
        "in_channels": 1,
        "res_channels": 128,
        "use_batch_norm_Wave":false,
        "use_dropout_Wave":0.0
    },

    "PostNetParams":{
              "layers":12,
              "kernel_size":33,
```

```
                "channels":128,
                "use_batch_norm_Post":false,
                "use_dropout_Post":0.0
               },


    "trainingHyperParameters":{
        "PostNetActivated":false,
        "postActivateInSteps":105000,
        "wavenetParamsFrozen":false,

        "discriminatorTraining": false,
        "discriminatorRestart": true,

        "preTrainEpochs": 1,
        "fineTuneEpochs": 5,
        "log_every_n_steps": 6,
        "accumulate_grad_batches": 6,

        "learning_rate": 0.001,
        "learning_rate_decay_time" : 20000,
        "learning_rate_decay" : 0.85
    },



    "savingAndLogging":{
        "useSavedModel": false,
        "nameOfModelSaved": "wavenet/modelos/paramsRuns/
   learningRateDecay085_20kT4.pth",

        "saveModelIntervalEpochs": 100,
        "val_check_interval": 1000,
        "min_delta": 0.01,
        "patience": 15,
        "howManyAudiosValidationsSave":1
    },

    "lossesConfig":{
        "epsilon" : 0.00000001,
        "min_db_precision" : 60,

        "melspec1Config":{
                "n_fft":2048,
                "n_mels":120,
                "hop_length":512
        },

        "melspec2Config":{
                "n_fft":512,
                "n_mels":80,
                "hop_length":128
        }
```

```
    },

    "weightsOfLosses":{
        "PostNetWeightRelativeToWavenet": 3,
        "max_weight_short_pitch": 1,
        "weightOfMelspecLoss1": 0.00,
        "weightOfMelspecLoss2": 0.00,
        "weightOfL1Loss": 10,
        "weightOfCustomLoss": 0.0000,
        "weightOfLogLoss": 0,
        "weightOfAmplitudeLoss": 0,
        "weightOfDiscriminatorLoss":0
    }
}
```

Listing 5.1: Archivo de hiperpárametros del modelo.