Singing audio synthesis by deep learning models: implementation and analysis of an asymmetric U-Net architecture

Gabriel Pena[2,1] y Ricardo A. Veiga[1]

[1] Universidad de Buenos Aires, Facultad de Ingeniería, Ciudad de Buenos Aires, Argentina
[2] Universidad Nacional de Tres de Febrero, Buenos Aires, Argentina

Abstract Singing audio synthesis is a challenging problem in the field of signal processing, due to the need to preserve not only the main harmonics but also fine tonal information. Specifically, the problem of converting speech into singing (*speech-to-singing, S2S*) is particularly complex, as it requires separating and properly recombining information from different source signals: phonetic and timbral content from the spoken audio and tonal and rhythmical structure from the melody. In this work, we implement an asymmetric U-Net architecture previously used in this field. Seven different models are proposed, and their performance is evaluated through objective metrics. The experiments were conducted using the same dataset and under the same conditions reported in the original reference work. In this context, some of the proposed models showed performance comparable to that of the original work, by using a simpler architecture.

Keywords: learning, neural networks, audio synthesis, spectrograms, unet

Síntesis de audio cantado mediante modelos de aprendizaje profundo: implementación y análisis de una arquitectura U-Net asimétrica

Resumen La síntesis de audio cantado constituye un problema desafiante en el campo del procesamiento de señales, dada la necesidad de preservar no solo los armónicos principales sino también información tonal fina. En particular, el problema de conversión de habla en canto resulta especialmente complejo pues requiere separar y recombinar distintas fuentes de información: el contenido fonético y tímbrico del habla y la estructura tonal y rítmica de una melodía. En este trabajo se implementa una arquitectura de tipo U-net asimétrica previamente utilizada

en este campo. Se proponen siete modelos y se evalúa su desempeño mediante métricas objetivas. Los experimentos se realizaron utilizando el mismo conjunto de datos y bajo las mismas condiciones reportadas en el trabajo original de referencia. En este contexto, varios de los modelos propuestos obtuvieron una puntuación comparable a la del original en las métricas de comparación utilizando una arquitectura más simple.

Keywords: aprendizaje, redes neuronales, síntesis de audio, espectrogramas, unet

1 Introducción

En este trabajo se aborda la transformación de audio hablado en audio cantado, de acuerdo con la estructura de una melodía proporcionada. Se trata de una tarea compleja de procesamiento de audio que tiene aplicación en áreas como producción musical, restauración, doblaje y generación de contenido audiovisual en general (Brum & Moreno, 2020; Shen et al., 2025; Zhang et al., 2025). El problema más general de síntesis de canto (Bai et al., 2025; Bonada & Serra, 2007) constituye además la base de sintetizadores comerciales como Vocaloid (Kenmochi & Oshita, 2007; Yamaha Corporation, s.f.) o Synthesizer V (Dreamtonics Co., Ltd., s.f.).

La complejidad del problema radica en la necesidad de extraer información espectral fina, en particular el contenido timbral del habla, y recombinarla coherentemente siguiendo una regla (la estructura proporcionada por la melodía) codificada en una señal externa. En este contexto, los métodos tradicionales basados en modelado explícito y análisis espectral presentan limitaciones. Las señales involucradas suelen ser altamente variables, aperiódicas y no estacionarias. Más aún, la separación explícita de una señal de audio en componentes como contenido tonal, timbre y estructura rítmica constituye un desafío. Actualmente, en este campo se utilizan modelos de aprendizaje profundo (Cho et al., 2021; Pan et al., 2025), dado que este enfoque se destaca por la capacidad de capturar relaciones complejas, patrones no lineales y potencialmente no estacionarios.

J. Parekh (Parekh et al., 2020) propuso un modelo basado en una arquitectura U-net (Ronneberger et al., 2015) asimétrica. El autor utiliza un enfoque de aprendizaje multitarea (*multi-task learning*), que consiste en una combinación de distintos regímenes de aprendizaje. En otros trabajos recientes se abordan propuestas similares, incluyendo modelos adversariales (Wu & Yang, 2020) e incorporación de capas de atención cruzada (Agarwal et al., 2022). En este trabajo proponemos siete variantes de la arquitectura de Parekh (Parekh, 2020), con el propósito de analizar el desempeño de cada una. De esta manera, se encuentra que la arquitectura puede ser simplificada sin perder calidad en los resultados. Para realizar una validación coherente, entrenamos la red con el *dataset* NUS-48E (Duan et al., 2013) (mismo *dataset* empleado en Agarwal et al., 2022; Parekh

et al., 2020) y comparamos nuestros resultados contra los obtenidos por Parekh mediante métricas de calidad.

A continuación se describe la metodología propuesta, incluyendo el preprocesamiento de los datos, la arquitectura utilizada y la reconstrucción del audio. Luego se describen los experimentos realizados, seguido de los resultados obtenidos. Finalmente, se presentan las conclusiones.

2 Metodología

2.1 Arquitectura

El modelo que proponemos emplea una arquitectura de tipo U-Net (Ronneberger et al., 2015) asimétrica, que consiste de dos ramas de *encoder* y un único *decoder* compartido (ver Fig. 1). La red recibe dos entradas: un espectrograma de audio hablado (al que nos referiremos como *speech*) y un contorno de melodía, ambos de dimensión $F \times T$. El contorno es una imagen binaria de igual dimensión que el espectrograma correspondiente, la cual representa la frecuencia fundamental f_0 presente en cada unidad de tiempo. La salida de la red es un espectrograma que contiene la información melódica del contorno adaptada al timbre de voz del audio hablado. Los *encoders* cuentan principalmente con una

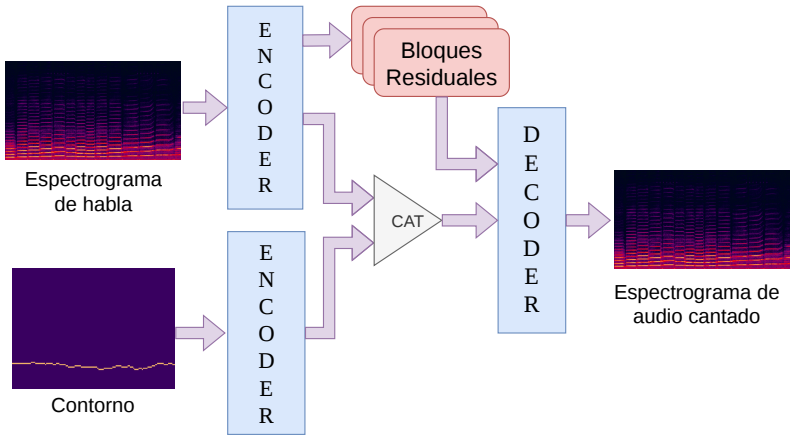

Figura 1: Esquema general del modelo de Parekh (Parekh et al., 2020).

secuencia de bloques convolucionales y un segmento recurrente, como se muestra en la Fig. 2. Cada bloque convolucional opera sobre una de las dimensiones, reduciéndola a la mitad. La salida de cada encoder es, entonces, una matriz de dimensión $F_b \times T_b = \frac{1}{8}F \times \frac{1}{8}T$, previa amplificación por un factor de ganancia K. Ambas matrices son concatenadas en la dimensión de características, formando

un único arreglo de $2F_b \times T_b$, con el cual se alimenta el decoder. Adicionalmente, el encoder del *speech* cuenta con tres conexiones residuales que alimentan el decoder en distintos niveles (Fig. 2). El encoder del contorno no emplea conexiones residuales.

El único decoder también consiste en tres bloques convolucionales acompañados de uno recurrente. El decoder no es un reflejo simétrico del encoder, pues el bloque recurrente aparece ubicado en lugares diferentes. En el encoder, dicho bloque se localiza en un nivel próximo al *bottleneck*; el hecho de operar en una dimensión temporal reducida en un factor de 4 le permite aprender relaciones de largo alcance. En el decoder, su ubicación es en el nivel más alto, previo a la salida, con el objetivo de aprender relaciones temporales de rango corto. A la salida del decoder se agrega una activación final de tipo ReLU, para garantizar que la señal de *output* sea un espectrograma de magnitud. Los bloques individuales se muestran en la Fig. 3. Los bloques convolucionales siguen la misma estructura que en el modelo original (Parekh et al., 2020), utilizando convoluciones 1D, normalizaciones *BatchNorm* y activaciones *LeakyReLU*, además de una capa de *dropout*. Como elemento recurrente se utiliza una *long short-term memory* (LSTM) unidireccional.

Las conexiones residuales pasan a través de un bloque que contiene una convolución, un elemento recurrente y una convolución transpuesta (Fig. 3), cuya salida se suma con una conexión directa. Por consiguiente, estas conexiones no alteran dimensiones, y se suman a las salidas parciales del decoder en los niveles que les corresponde por dimensión.

Una particularidad de esta arquitectura es que, dado que las dimensiones temporal y frecuencial se reducirán en un factor de 8, se requiere que ambas sean divisibles por este número. Esto permite mantener una estructura interna fija que no dependa de los datos ni requiera reducciones variables o interpolaciones. Al definir 512 bandas de frecuencia, el problema se reduce únicamente a garantizar que la longitud temporal de todos los datos sea divisible por 8. Este problema se resuelve mediante la aplicación de un *padding* dinámico.

2.2 Preprocesamiento

Para entrenar la red utilizamos el *dataset* abierto NUS-48E (Duan et al., 2013). Este *dataset* es el mismo que han utilizado previamente otros autores de trabajos similares (Agarwal et al., 2022; Parekh et al., 2020; Wu & Yang, 2020). Se compone de grabaciones de 20 canciones del folclore inglés cantadas por 12 sujetos para un total de 115 minutos de audio cantado, emparejados con sus respectivos audios hablados. Para adecuar los datos a las necesidades de nuestra red, se empleó una versión simplificada del *pipeline* descrito en Parekh et al., 2020, como se detalla a continuación.

Segmentación de audios. Los audios en crudo no son adecuados para utilizar en el entrenamiento, ya que se requiere de un número significativo de datos y el *dataset* proporciona solo 48 grabaciones completas. Segmentando y recombinando los audios adecuadamente se puede obtener un *dataset* extenso formado por muestras sencillas, como audios de unos pocos segundos. Adicionalmente,

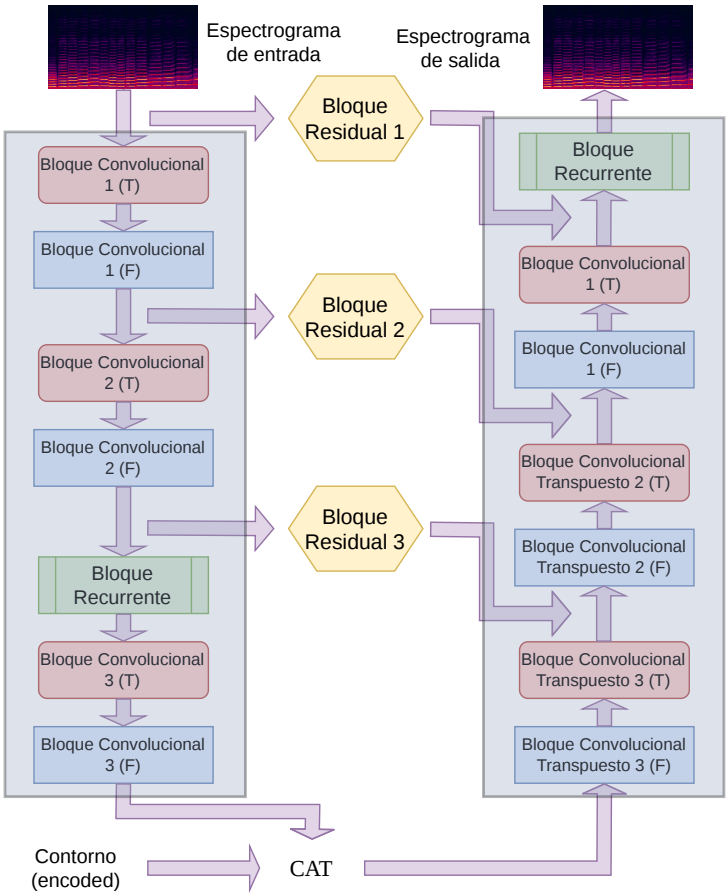

Figura 2: Diagrama esquemáticos de los dos componentes principales de la red. Izquierda: *encoder*. Derecha: *decoder*.

usar segmentaciones de diferentes longitudes con redundancia nos permitió obtener una cantidad de datos mucho mayor. El procedimiento usado es el mismo que se indica en Parekh et al., 2020: (i) separar en frases; (ii) separar en palabras; (iii) construir combinaciones de tres o más palabras consecutivas por frase. Este procedimiento, además, permite eliminar todos los silencios sin requerir procesamiento adicional. El *dataset* resultante contiene 16578 elementos.

Remuestreo. Como se propone en Agarwal et al., 2022; Parekh et al., 2020, todos los audios (originalmente muestreados a 44 kHz) se remuestrean a 16 kHz, una frecuencia muestral suficiente para el procesamiento de habla y canto.

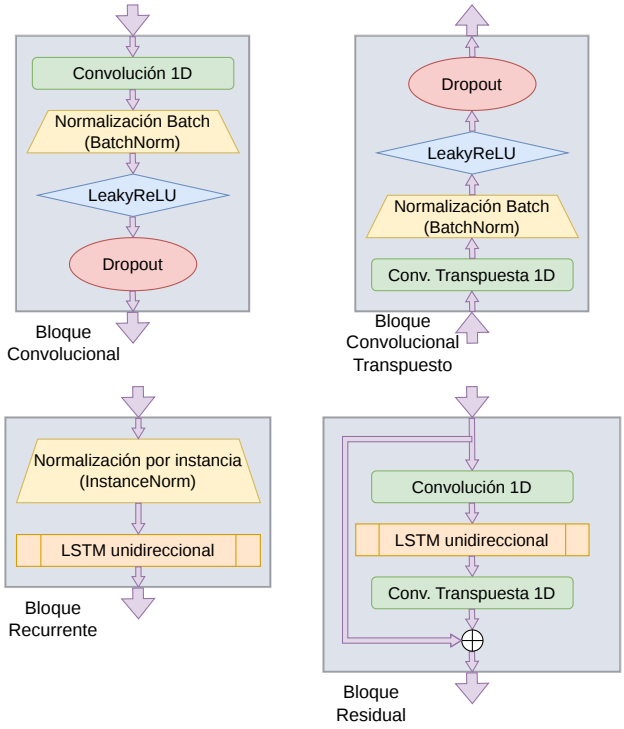

Figura 3: Bloques individuales de la red.

Estiramiento temporal. Para poder comparar coherentemente el *target* con el *output* del modelo, se requiere que sus dimensiones coincidan. Sin embargo, el audio hablado y su audio cantado asociado tienen en general duraciones distintas, siendo el audio cantado usualmente más largo. Para solucionarlo, reescalamos la frecuencia de muestreo del audio hablado, lo que equivale a "estirarlo". Este procedimiento no debe realizarse de manera uniforme sobre todo el audio, sino fonema a fonema, de lo contrario la duración relativa de cada sonido cambia, produciendo artefactos.

Extracción del contorno. El siguiente paso es generar la imagen de contorno de la melodía. Para ello utilizamos el algoritmo CREPE (Kim et al., 2018), una red neuronal preentrenada. El resultado es que a partir de cada audio obtenemos un contorno $\boldsymbol{C} \in \{0,1\}^{512\times T}$, con 512 niveles frecuenciales.

Construcción de espectrogramas. Se convierte cada señal de audio en un espectrograma mediante la transformada corta de Fourier (STFT). Se utilizan los mismos parámetros especificados en Agarwal et al., 2022, lo que resulta en un espectrograma con 513 bandas de frecuencia y *frames* temporales de 64 ms con un solapamiento del 75% entre cada dos sucesivos. Dado que el último nivel

frecuencial es el menos relevante, lo descartamos para quedarnos con 512 bandas. De esta manera, el espectrograma de voz cantada tendrá exactamente las mismas dimensiones que el contorno previamente construido. Los espectrogramas además son convertidos a escala logarítmica natural en magnitud (Eq. 1):

$$f(x) = \ln(1 + x), \qquad f^{-1}(x) = e^x - 1. \tag{1}$$

Parekh (Parekh et al., 2020) utiliza la misma escala, en tanto en Wu y Yang, 2020 emplean la escala Mel.

Normalización. Para que el modelo sea capaz de aprender correctamente, es necesario que haya una coherencia en la escala de los datos de entrenamiento. Para ello se aplica una normalización *min-max* para mapear los valores al intervalo $[0, 1]$.

El procedimiento completo produce ternas $(\boldsymbol{S}, \boldsymbol{C}, \boldsymbol{Y})$, donde $\boldsymbol{S}, \boldsymbol{Y} \in [0, 1]^{512 \times T}$ son los espectrogramas de habla y canto respectivamente, y $\boldsymbol{C} \in \{0, 1\}^{512 \times T}$ es el contorno melódico binario.

2.3 Entrenamiento

Todos los modelos que describiremos en la Sección 3 fueron entrenados bajo los mismos parámetros. Se empleó el optimizador RAdam (Liu et al., 2019) con una tasa de aprendizaje $\lambda = 0.002$ y parámetros $(\beta_1, \beta_2) = (0.9, 0.9)$, utilizando *minibatches* de tamaño 4 y una función de costo *MSE* (error cuadrático medio). para entrenar durante 30 épocas, utilizando *PyTorch* sobre una GPU NVIDIA RTX 3070 Ti. Del *dataset* se separó el 3% para pruebas y el resto se utilizó para el entrenamiento.

Como previamente indicamos, es necesario asegurar que las señales tengan ambas dimensiones divisibles por 8. Para resolver esto se utiliza un *padding*. En la Fig. 4 se observa que los *frames* presentan una duración media de 152 *frames* y una desviación estándar de 80, lo que indica una alta variabilidad. Se observa, además, la presencia de *outliers* de duraciones mayores a los 500 *frames*. Por consiguiente, no es posible realizar el *padding* directamente sobre todo el conjunto, pues en tal caso la mayoría de las señales resultantes quedarían compuestas por mayoritariamente por silencio. Para sortear esta dificultad, el *padding* se realiza de manera dinámica sobre cada *minibatch* al comenzar el entrenamiento: todas las señales son extendidas a una duración igual al múltiplo de 8 más cercano a la duración máxima presente en el *minibatch*. Al utilizar *minibatches* de tamaño 4, el impacto del *padding* se reduce significativamente, pues los que contengan señales problemáticas serán pocos (notar que la mayoría de las señales tiene una duración de entre 30 y 300 *frames* aproximadamente).

2.4 Reconstrucción del audio

El modelo devuelve un espectrograma de magnitud en escala logarítmica, por lo que para poder evaluar perceptualmente el resultado, necesitamos regenerar el audio a partir de él. Este procedimiento no es directo porque no se dispone de

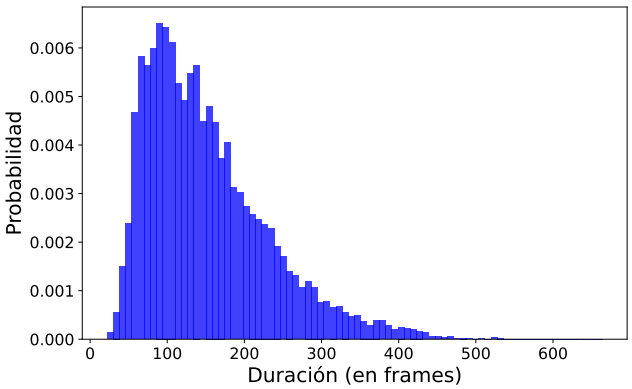

Figura 4: Histograma de la duración en *frames* de los espectrogramas del *dataset*.

información de la fase, así que ésta debe estimarse. Para lograr esto empleamos el algoritmo Griffin-Lim (Griffin & Lim, 1984) en su versión *Fast Griffin-Lim* que se encuentra implementada en el paquete Librosa (Librosa Development Team, s.f.) de Python. Las intensidades del espectrograma (que el modelo aprende a representar en el rango $[0, 1]$) son reescaladas de modo que su nivel máximo sea de -1 dBFS, lo que corresponde a un decibel por debajo del umbral de saturación digital.

3 Diseño experimental

Se realizaron tres experimentos distintos, cada uno orientado a evaluar posibles modificaciones en la arquitectura de la red. Se utilizó como elemento de comparación el único dato provisto de forma completa por Parekh (Parekh, 2020). Dicho autor provee, además del audio generado por su modelo, los audios hablado y cantado originales y el audio hablado estirado correspondientes. Enfatizamos que el no disponer de los *outputs* originales de Parekh et al. para otros ejemplos (y la imposibilidad de replicar exactamente su rutina de preprocesamiento) impidió realizar un análisis de tipo estadístico más completo. Para nuestros experimentos extrajimos el contorno melódico del audio cantado mediante CREPE y alimentamos cada modelo con dicho contorno junto con el audio estirado. Para dichos experimentos se entrenaron siete modelos, los cuales se describirán en esta sección. Las predicciones de cada uno fueron contrastadas entre sí tanto perceptualmente como mediante diversas métricas objetivas.

3.1 Métricas de evaluación

Para disponer de un criterio objetivo de evaluación se utilizaron tres métricas específicas, que a continuación se describen brevemente.

Perceptual Evaluation of Speech Quality (PESQ). Introducida en Rix et al., 2001, constituye una familia de estándares para evaluar la calidad del

audio teniendo en cuenta la audición humana. De acuerdo con Kelleher, 2023, los valores del PESQ se correlacionan fuertemente con los del *Mean Opinion Score (MOS)* en la evaluación de calidad del habla. No obstante, en el presente trabajo lo utilizaremos como métrica de referencia para evaluar la reconstrucción del audio cantado.

Short-Time Objective Intelligibility (STOI). Es una métrica específica de procesamiento del habla, cuyo objetivo es cuantificar la inteligibilidad (Taal et al., 2010, 2011). Un valor alto de STOI es indicador de una alta correlación local entre las señales, tanto en tiempo como en frecuencia.

Fundamental Frequency Root-Mean-Square Error (F0-RMSE). Se trata de una métrica estándar en síntesis de voz, cuyo propósito es medir la diferencia entre las frecuencias fundamentales de dos señales, expresada en Hertz. En particular, se extrae mediante CREPE el contorno melódico de la señal de salida y se la compara con el contorno de entrada. Este procedimiento permite medir de forma específica la calidad de reconstrucción de la melodía, independientemente del contenido timbral.

3.2 Experimentos

Dividimos nuestros experimentos en tres.

Experimento 1 — Bloques residuales. El objetivo de este experimento fue verificar si la utilización de conexiones residuales es necesaria. El modelo de Parekh se empleó como *baseline* para la comparación, y se entrenaron dos nuevos modelos. El modelo A elimina las capas residuales por completo, mientras que el modelo B cambia el origen de las conexiones al *encoder* del contorno.

Experimento 2 — Factor de ganancia. En este experimento se evalúa el impacto de agregar un factor de amplificación de la señal antes del *decoder*. Se entrenaron para ello los modelos C, D y E tomando como base el modelo A y agregando distintos factores de ganancia.

Experimento 3 — *Dropout*. Las capas de *dropout* tienen la función de apagar de forma aleatoria un cierto número de neuronas durante el entrenamiento, a fin de evitar la co-adaptación. El modelo de Parekh consideraba un 5% de *dropout*; aquí hemos probado elevar ese número a 10% y 20% respectivamente en los modelos F y G, construidos sobre la base del modelo D.

La Tabla 1 describe de manera sintética los experimentos realizados.

4 Resultados y discusión

Se presentan aquí los resultados numéricos y perceptuales obtenidos en la experimentación, junto con una breve discusión de los mismos. Los resultados se sintetizan en la Tabla 2. Para el caso particular de la métrica PESQ, dado que el estándar está diseñado para señales de habla y no de canto, los valores absolutos no son relevantes; nos interesan los valores relativos como elemento de comparación.

Tabla 1: Estructura de los ocho modelos considerados.

Experimento	Modelo	Conexión residual	Ganancia	*Dropout*
1	Parekh	Desde el *encoder* del habla	1	0.05
	A	—	1	—
	B	Desde el *encoder* del contorno	1	—
2	C	—	2	—
	D	—	4	—
	E	—	10	—
3	F	—	4	0.10
	G	—	4	0.20

Tabla 2: Valores de las métricas objetivas de calidad calculados en los experimentos.

Experimento	Modelo	PESQ	STOI	F0-RMSE
1	Parekh	1.16	0.50	7.14 Hz
	A	1.30	0.56	6.07 Hz
	B	1.12	0.40	6.49 Hz
2	C	1.23	0.54	6.08 Hz
	D	1.36	0.54	6.32 Hz
	E	1.28	0.55	5.52 Hz
3	F	1.33	0.57	5.28 Hz
	G	1.25	0.56	6.13 Hz

Experimento 1. Se observa que el modelo A obtiene una puntuación similar a la del de Parekh en todas las métricas con una arquitectura más simple. Perceptualmente, este audio también es el más similar al original. En cuanto a las métricas de calidad, nuestro modelo B y el de Parekh muestran resultados similares, sugiriendo que las conexiones residuales desde el *encoder* del contorno no aportan información relevante. De los tres modelos comparados, el A tuvo el mejor desempeño, por lo cual fue utilizado como base en los demás experimentos.

Experimento 2. Al introducir un factor de amplificación en el *bottleneck*, la calidad del audio mejora significativamente en los modelos C y D respecto del de Parekh. En particular, el modelo D resulta ser el que mejor calidad de audio produce en términos de PESQ, superando incluso a los del experimento 1. Este hecho refleja que la inclusión de una ganancia en el *bottleneck* no es neutra, sino que produce efectos emergentes significativos. Se observa además que el impacto ocurre mayoritariamente en la pronunciación y calidad general del habla, en tanto que la melodía se reconstruye en calidades similares en todos los casos. Perceptualmente, el modelo D fue el más satisfactorio, en coincidencia con lo indicado por el PESQ. Dicho modelo fue empleado como base para el último experimento.

Experimento 3. Finalmente, se evaluó el impacto de utilizar capas de *dropout* durante el entrenamiento. El modelo F, con un 10% de *dropout*, produce resultados similares a los del modelo D, con una ligera mejora en el STOI y un ligero declive en el PESQ, además de una mejora más notoria en el F0-RMSE. Para el modelo G, el declive de calidad es más pronunciado, especialmente a ni-

vel de PESQ, lo que sugiere que el rango óptimo de *dropout* se localiza en valores moderados, menores al 20%. El modelo F es claramente superior al G, pero si se lo compara contra los de experimentos anteriores, la diferencia ya no es tan clara. En particular, el modelo D y el modelo F presentan resultados similares en las métricas de habla y en lo perceptual, y ambos se mostraron superiores al de Parekh. En este experimento, se observó que la utilización de *dropout* no constituyó un factor determinante.

Análisis sobre el *dataset* de prueba. Con el propósito de realizar un estudio complementario del desempeño de la red, se evalúa la reconstrucción de datos que el modelo no ha visto durante el entrenamiento. El análisis fue realizado sobre el modelo D. Para cada dato se computaron todas las métricas, y luego se calcularon sus descriptores estadísticos: el valor mínimo, máximo, media y desviación estándar del conjunto. Los resultados se muestran en la Tabla 3.

Tabla 3: Descriptores estadísticos de las métricas de calidad en el *dataset* de prueba.

Descriptor	Mínimo	Máximo	Media	Desviación estándar
PESQ	1.02	1.95	1.32	0.19
STOI	0.14	0.86	0.62	0.14
F0-RMSE	1.43 Hz	264.56 Hz	9.28 Hz	25.48 Hz

Con excepción de la F0-RMSE, la variabilidad es muy baja, lo que sugiere una consistencia del modelo en casos no considerados durante el entrenamiento. En el caso de la F0-RMSE, la alta variabilidad es producto de la diversidad de registros tonales del *dataset*: una misma distancia tonal se traduce, en registros altos (e.g. una soprano) en diferencias en Hertz mucho mayores, algo que el cálculo directo de la F0-RMSE no tiene en cuenta. En la Fig. 5 se muestra un *violin plot* de la F0-RMSE en el conjunto de prueba, donde se observa claramente la presencia de valores en todo el rango de entre 0 y 280 Hz.

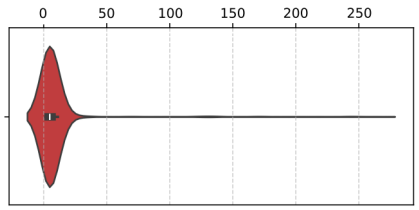

Figura 5: *Violin plot* de la F0-RMSE en el *dataset* de prueba.

4.1 Discusión

En conjunto, los resultados experimentales muestran que eliminar las conexiones residuales del modelo original y utilizar un factor moderado de ganancia en el *bottleneck* produce mejoras consistentes en calidad perceptual y métricas objetivas. Se observó que las conexiones residuales propuestas originalmente por el autor no resultaron neutras, sino que empeoraron la calidad. Su eliminación no solo produjo una mejora apreciable en el resultado sino que además permitió simplificar el modelo, pues cada conexión residual implica el agregado de una cantidad no despreciable de parámetros adicionales. Por otro lado, el uso de capas de *dropout* (técnica usual para prevenir sesgos de co-adaptación) en un nivel bajo o moderado no produce un incremento notable en la calidad del audio, en tanto que un nivel alto se mostró perjudicial. El uso de métricas representativas de aspectos de la calidad, con alta correlación perceptual, permite además mostrar que la energía del espectrograma por sí sola no constituye un indicador confiable de la calidad del audio. Los mejores resultados se obtuvieron con los modelos D y F, que resultaron las variantes más efectivas de la arquitectura evaluada de acuerdo con lo observado en los experimentos. Dichos resultados fueron superiores a los obtenidos con el modelo original de Parekh.

Es importante destacar que los valores numéricos de las métricas son relativos, y solo nos valemos de ellos a efectos comparativos. Por ejemplo, suele considerarse que un PESQ inferior a 2 es indicador de audios incomprensibles; sin embargo, tanto el audio de Parekh como los de nuestros modelos tienen valores de PESQ muy inferiores y, perceptualmente son comprensibles. Los ejemplos se encuentran disponibles y pueden verificarse (Parekh, 2020; Pena, 2025). Destacamos, además, que si bien el audio de Parekh no se encuentra escalado a -1 dBFS, reescalarlo no alteró significativamente el valor de las métricas de calidad.

5 Conclusiones

En este trabajo hemos presentado diversas variantes de un modelo de aprendizaje profundo para sintetizar voz cantada a partir de un audio hablado y una melodía. Dicho modelo, que utiliza una arquitectura U-Net asimétrica compuesta por capas convolucionales y recurrentes, fue entrenado con el *dataset* NUS-48E. También evaluamos las diferentes variaciones de la arquitectura y las comparamos entre ellas y con la original de Parekh. En particular, analizamos el impacto de las conexiones residuales, del uso de un factor de ganancia en el *bottleneck* y de la presencia de capas de *dropout*. Los resultados mostraron que ni el uso de conexiones residuales ni el de capas de *dropout* aportó beneficios significativos, por lo cual la arquitectura puede simplificarse. Respecto del modelo de Parekh, casi todos nuestros modelos mostraron una mejora en las métricas de calidad sobre el audio de prueba, siendo los mejores el D (ganancia 4, sin *dropout*) y el F (ganancia 4, *dropout* del 10%). Adicionalmente, el análisis sobre el conjunto de prueba mostró que el sistema mantiene un desempeño consistente sobre datos no observados, sugiriendo que el modelo entrenado consigue generalizar de manera adecuada.

Acknowledgments. Se agradece a la Universidad Nacional de Tres de Febrero por el soporte bajo el proyecto 80020240600005TF.

Referencias

Agarwal, S., Ganapathy, S., & Takahashi, N. (2022). Leveraging Symmetrical Convolutional Transformer Networks for Speech to Singing Voice Style Transfer. *INTERSPEECH 2022 - Voice Conversion and Adaption II*. https://doi.org/10.48550/arXiv.2208.12410

Bai, P., Zheng, M., & Shi, X. (2025). A Survey of Singing Voice Synthesis. *Proceedings of the 10th Conference on Sound and Music Technology*.

Bonada, J., & Serra, X. (2007). Synthesis of the Singing Voice by Performance Sampling and Spectral Models. *IEEE Signal Processing Magazine*, *24*(2), 67-79. https://doi.org/10.1109/MSP.2007.323266

Brum, L. A. Z., & Moreno, E. D. (2020). Challenges and Perspectives on Real-time Singing Voice Synthesis. *Revista de Informática Teórica e Aplicada*, *27*(4), 118-126. https://doi.org/10.22456/2175-2745.107292

Cho, Y.-P., Yang, F.-R., Chang, Y.-C., Cheng, C.-T., Wang, X.-H., & Liu, Y.-W. (2021). A Survey on Recent Deep Learning-driven Singing Voice Synthesis Systems. *Proceedings of the IEEE International Conference on Artificial Intelligence and Virtual Reality (AIVR)*.

Dreamtonics Co., Ltd. (s.f.). Synthesizer V [Accessed: 2026-03-16]. https://dreamtonics.com/synthesizerv/

Duan, Z., Fang, H., Li, B., Sim, K. C., & Wang, Y. (2013). The NUS sung and spoken lyrics corpus: A quantitative comparison of singing and speech. *2013 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference*, 1-9. https://doi.org/10.1109/APSIPA.2013.6694316

Griffin, D. W., & Lim, J. (1984). Signal estimation from modified short-time Fourier transform. *IEEE Transactions on Acoustics, Speech, and Signal Processing*.

Kelleher, L. (2023). What is PESQ? - Cyara.

Kenmochi, H., & Oshita, H. (2007). VOCALOID — Commercial singing synthesizer based on sample concatenation. *Proceedings of the Interspeech 2007*, 4009-4010.

Kim, J. W., Salamon, J., Qi Li, P., & Bello, J. P. (2018). Crepe: A Convolutional Representation for Pitch Estimation. *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 161-165.

Librosa Development Team. (s.f.). Librosa [Accessed: 2026-03-31]. https://librosa.org/doc/main/index.html

Liu, L., Jiang, H., He, P., Chen, W., Liu, X., Gao, J., & Han, J. (2019). On the Variance of the Adaptive Learning Rate and Beyond. https://doi.org/10.48550/arXiv.1908.03265

Pan, C., Yao, D., Zhang, Y., Guo, W., Lu, J., Zhu, Z., & Zhao, Z. (2025). Synthetic Singers: A Review of Deep-Learning-based Singing Voice Synthesis Approaches. *Proceedings of the 14th International Joint Conference on Natural Language Processing and the 4th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics*.

Parekh, J. (2020). Speech-to-Singing Conversion in an Encoder-Decoder Framework - ICASSP 2020 Project Website. https://jayneelparekh.github.io/icassp20

Parekh, J., Rao, P., & Yang, Y.-H. (2020). Speech-To-Singing Conversion in an Encoder-Decoder Framework. *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 261-265. https://doi.org/10.1109/ICASSP40776.2020.9054473

Pena, G. (2025). SkyVoiceNet. 10.5281/zenodo.17119012

Rix, A. W., Beerends, J. G., Hollier, M. P., & Hekstra, A. P. (2001). Perceptual evaluation of speech quality (PESQ)-a new method for speech quality assessment of telephone networks and codecs. *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, *2*, 749-752. https://doi.org/10.1109/ICASSP.2001.941023

Ronneberger, O., Fischer, P., & Brox, T. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. *Medical Image Computing and Computer-Assisted Intervention (MICCAI 2015)*, 234-241.

Shen, C., Zhao, L., Fu, C., Gan, B., & Du, Z. (2025). Transinger: Cross-Lingual Singing Voice Synthesis via IPA-Based Phonetic Alignment. *Sensors*, *25*(13). https://doi.org/10.3390/s25133973

Taal, C. H., Hendriks, R. C., Heusdens, R., & Jensen, J. (2010). A short-time objective intelligibility measure for time-frequency weighted noisy speech. *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*.

Taal, C. H., Hendriks, R. C., Heusdens, R., & Jensen, J. (2011). An Algorithm for Intelligibility Prediction of Time–Frequency Weighted Noisy Speech. *IEEE Transactions on Audio, Speech, and Language Processing*, *19*(7), 2125-2136. https://doi.org/10.1109/TASL.2011.2114881

Wu, D.-Y., & Yang, Y.-H. (2020). Speech-To-Singing Conversion based on Boundary Equilibrium GAN. *INTERSPEECH 2020 - Singing and Multimodal Systems*. https://doi.org/10.48550/arXiv.2005.13835

Yamaha Corporation. (s.f.). VOCALOID [Accessed: 2026-03-16]. https://www.vocaloid.com

Zhang, Y., Guo, W., Pan, C., Yao, D., Zhu, Z., Jiang, Z., Wang, Y., Jin, T., & Zhao, Z. (2025). TCSinger 2: Customizable Multilingual Zero-shot Singing Voice Synthesis. *Findings of the Association for Computational Linguistics: ACL 2025*, 13280-13294. https://doi.org/10.18653/v1/2025.findings-acl.687