UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Informática

Sistemas de síntesis de voz basados en redes neuronales para lenguas europeas

Trabajo Fin de Grado

Grado en Ingeniería Informática

AUTOR/A: Iranzo Sánchez, Jorge

Tutor/a: Juan Císcar, Alfonso

Cotutor/a: Iranzo Sánchez, Javier

Director/a Experimental: PEREZ GONZALEZ DE MARTOS, ALEJANDRO MANUEL

CURSO ACADÉMICO: 2021/2022

Resum

La síntesi de veu (TTS, de l'angles Text-To-Speech) és una dels àrees més actives dins de la intel·ligència artificial, particularment en el camp de l'aprenentatge automàtic. Recentment, aquesta àrea ha sigut el focus d'atenció per part d'importants figures tecnològiques com Google, Facebook, Microsoft, etc. a causa de les millores de rendiment obtingudes per aquesta tecnologia gràcies a la incorporació de xarxes neuronals artificials. En aquest sentit, la nova era de sistemes TTS basats en xarxes neuronals ha portat amb si sistemes de síntesis de veu de gran naturalitat que, en contrast amb els sistemes tradicionals, no requereixen de gran coneixement expert en processament del senyal i aspectes lingüístics. En aquest treball es proposa estudiar i implementar models avançats de TTS en llengües europees i, en particular, en castellà i alemany. Per a això, es farà ús de dades, tecnologia i experiència del grup MLLP del VRAIN, adquirits en el marc de projectes d'investigació i transferència tecnològica desenvolupats en els últims cinc anys.

Paraules clau: Intel·ligència artificial, Aprenentatge automàtic, Xarxes neuronals, Síntesi de veu

Resumen

La síntesis de voz (TTS, del inglés Text-To-Speech) es una de las áreas más activas dentro de la inteligencia artificial, particularmente en el campo del aprendizaje automático. Recientemente, esta área ha sido el foco de atención por parte de importantes figuras tecnológicas como Google, Facebook, Microsoft, etc. debido a las mejoras de rendimiento obtenidas por esta tecnología gracias a la incorporación de redes neuronales artificiales. En este sentido, la nueva era de sistemas TTS basados en redes neuronales ha traído consigo sistemas de síntesis de voz de gran naturalidad que, en contraste con los sistemas tradicionales, no requieren de gran conocimiento experto en procesado de la señal y aspectos lingüísticos. En este trabajo se propone estudiar e implementar modelos avanzados de TTS en lenguas europeas y, en particular, en castellano y alemán. Para ello, se hará uso de datos, tecnología y experiencia del grupo MLLP del VRAIN, adquiridos en el marco de proyectos de investigación y transferencia tecnológica desarrollados en los últimos cinco años.

Palabras clave: Inteligencia artificial, Aprendizaje automático, Redes neuronales, Síntesis de voz

Abstract

Text-To-Speech (TTS) is one of the most active areas within artificial intelligence, particularly in the field of machine learning. Recently, this area has been the focus of attention of important technological figures such as Google, Facebook, Microsoft, etc. due to the performance improvements obtained by this technology thanks to the incorporation of artificial neural networks. In this sense, the new era of TTS systems based on neural networks has brought with it highly natural speech synthesis systems that, in contrast to traditional systems, do not require great expertise in signal processing and linguistic aspects. In this work we propose to study and implement advanced TTS models in European languages and, in particular, in Spanish and German. For this purpose, we will make use of data, technology and experience of the MLLP group of the VRAIN, acquired in the framework of research and technology transfer projects developed in the last five years.

Key words: Artificial intelligence, Machine learning, Neural networks, Speech synthesis

Índice general

Índice de figuras

Índice de tablas

CAPÍTULO 1
Introducción

En este trabajo se analiza y estudia la aplicación de redes neuronales dentro del campo de la síntesis del habla (TTS). En concreto, se estudia el uso de distintos modelos del área y la puesta en marcha de experimentos con el propósito de obtener sistemas capaces de sintetizar voces de lenguas europeas en alta calidad. Para ello, se propone un método basado en *Transfer Learning* para el entrenamiento de estos modelos con el fin de, por un lado, intentar mejorar la calidad resultante de estos cuando se emplea un conjunto de datos reducido y, por otro lado, agilizar el proceso de entrenamiento para nuevos idiomas. Además, se ha llevado a cabo una evaluación formal con el fin de valorar si existen mejoras mediante el uso de este mecanismo frente a un entrenamiento tradicional.

1.1 Motivación

Es indiscutible la importancia del habla como acto comunicativo entre los seres humanos. El lenguaje oral permite que las personas se relacionen entre sí, complementando las distintas vías de expresión de los individuos. A través de la agrupación de varias combinaciones fonéticas, la voz humana reproduce los sonidos asociados al léxico y las reglas de las lenguas, siendo el movimiento de los órganos que forman el tracto vocal el mecanismo biológico que hace esto posible. Esta capacidad no es innata, sino que se aprende desde la infancia mediante la interacción humana, evolucionando y estando sujeta a los aspectos socioculturales del entorno.

A lo largo de la historia, el campo de la lingüística ha estudiado este fenómeno exhaustivamente, y se han llevado a cabo numerosos intentos de reproducirlo y generarlo artificialmente. Sin embargo, no es hasta principios del siglo XX, que se empezaron a conseguir resultados comparables a la calidad de la voz humana gracias a la creación de los primeros dispositivos electrónicos con procesamiento digital de señales. Junto a estos avances tecnológicos, el aumento gradual de la globalización reavivó el interés en esta rama, que había dado lugar a una nueva necesidad de sistemas que rompiesen con las barreras lingüísticas y automatizasen el proceso de la comunicación.

Gracias a los grandes avances de esta tecnología en las últimas décadas, llegamos al día de hoy, donde la síntesis del habla ha logrado superar sus límites previamente establecidos y forma parte de aplicaciones que abarcan un amplio abanico de necesidades. Ejemplos de esto varían desde el empleo de asistentes virtuales en teléfonos móviles, coches o en el apoyo a personas con trastornos del habla, a entornos artísticos como el uso en la música mediante cantantes virtuales.

Esta popularidad no ha pasado desapercibida por grandes figuras tecnológicas como Google o Amazon, que participan activamente en el área y la utilizan en sus productos y

servicios junto a otras tecnologías como el reconocimiento automático del habla (ASR) o la traducción automática (MT).

Mucha de esta atención se debe a las mejoras drásticas de rendimiento y calidad obtenidas por la utilización de redes neuronales artificiales que, actualmente, reciben un amplio uso y han revolucionado con su incorporación otros campos como la visión por computador (CV).

Teniendo en cuenta la situación actual descrita anteriormente, es interesante el estudio del uso de estos modelos de sistemas de síntesis de voz neuronal de alta fidelidad. En particular, ha surgido un interés en sistemas capaces de facilitar la comunicación entre lenguas que suelen interactuar habitualmente. Un caso evidente es aquellas utilizadas en el continente europeo, donde el turismo y el comercio es común, y no es inusual que en un país haya más de una lengua cooficial. Es a partir de este marco en el que se plantea el desarrollo de este trabajo.

1.2 Marco de trabajo

Este trabajo se ha realizado utilizando la infraestructura y los recursos proporcionados por el *Machine Learning and Language Processing Group* (MLLP) de la UPV. Los modelos teóricos y el *software* utilizados han sido desarrollados por el Dr. Alejandro Pérez González de Martos, director experimental de este trabajo. Estos son explorados en profundidad en su tesis [1]. El trabajo del autor se ha centrado en el estudio del área, el entrenamiento de estos modelos y la recogida de datos y elaboración del proceso de evaluación realizado.

1.3 Objetivos

Basándose en el enfoque y la motivación explicados, los objetivos de este trabajo son los siguientes:

- Estudiar el estado del arte y los métodos utilizados en el aprendizaje automático y la síntesis del habla con el fin de obtener una mayor comprensión de estos.
- Construir y entrenar sistemas de síntesis de habla de última generación que sean flexibles y puedan generar audio de alta calidad en múltiples lenguas.
- Evaluar la calidad de los sistemas actuales de síntesis del habla identificando futuros temas de investigación en el área.

1.4 Estructura de la memoria

Además de este capítulo introductorio, este documento cuenta con otros 5 capítulos que tratan los objetivos propuestos en este trabajo. En el capítulo 2 se introducen algunos de los conceptos más relevantes en el área del aprendizaje automático y el procesamiento de lenguajes naturales, explicando los tipos generales de redes neuronales utilizadas y el funcionamiento de los sistemas de síntesis del habla actuales. En el capítulo 3 se introduce la síntesis de voz neuronal no autorregresiva y el modelo base de TTS en el que se basa el resto del trabajo. En el capítulo 4 se expande este modelo para que pueda clonar las características de las voces de múltiples parlantes y se introduce la idea de modelos de TTS *end-to-end*. En el capítulo 5 se introduce el concepto de *Transfer Learning* y se evalúa

su eficacia en la tarea dada respecto al resto de modelos entrenados en previos capítulos. Por último, en el capítulo 6 se concluye el contenido principal de esta memoria con un resumen del trabajo realizado, las conclusiones que se han extraído de este y el posible trabajo futuro a realizar. Se recomienda que el lector siga un orden de lectura secuencial de los capítulos, pero lectores experimentados en la materia pueden saltarse las secciones con información general o descripciones de modelos con los que estén familiarizados.

CAPÍTULO 2
Preliminares

En este capítulo se introducen algunos conceptos generales, ideas y terminología de los campos de la ciencia de la computación utilizados a lo largo de este trabajo, y que el lector necesitará para comprender el resto del trabajo. Algunos detalles de conceptos más específicos se introducen más adelante en sus capítulos correspondientes.

2.1 Aprendizaje Automático

2.1.1. Conceptos generales

Dentro del vasto campo que es la Inteligencia Artificial (IA), el área del Aprendizaje Automático (ML) se encarga del estudio y desarrollo de aplicaciones y sistemas capaces de aprender a partir de datos suministrados o experiencias pasadas, con la finalidad de resolver tareas y problemas específicos [2].

Fundamentalmente, el funcionamiento de estos sistemas está gobernado por modelos estadísticos. El aprendizaje del sistema consiste en la búsqueda de algún modelo que generalice una serie de datos suministrados, es decir, que logre predecir y devolver el resultado deseado al introducir nuevos datos de entrada que no ha visto previamente. Así, el objetivo es que mediante algún algoritmo $\mathcal{A}$, a través de un conjunto de datos de entrenamiento, el valor óptimo de los parámetros o pesos que determinan la salida del sistema sean aprendidos por este.

Los problemas de Aprendizaje Automático se distinguen según el dominio de la salida del sistema. En problemas de clasificación se desea predecir la salida y que toma valores entre una serie de clases C tal que $y \in \{1, ..., C\}$. Por otro lado, en problemas de regresión se desea predecir la salida y que toma valores dentro de un conjunto A no numerable como puede ser $\mathbb{R}$. Dicho de otra forma, en problemas de clasificación se hace uso de valores discretos y en los de regresión se utilizan valores continuos.

Dependiendo de la información disponible que tiene el modelo sobre los datos de entrenamiento, los algoritmos de Aprendizaje Automático se clasifican en diferentes paradigmas. En el contexto de este trabajo, se hace uso del aprendizaje supervisado, que suele utilizarse al abordar problemas dentro del procesamiento natural de lenguajes. En este, para cada dato de entrada x se proporciona una etiqueta y identificando el valor que representa la muestra. De esta manera, en el entrenamiento del modelo, se puede tomar una decisión más adecuada al variar sus parámetros, dependiendo de si al procesar x se ha identificado correctamente la muestra como y o no.

En el caso de este trabajo, el TTS está clasificado como un problema de regresión, y se suele estudiar bajo el uso de modelos de aprendizaje supervisado.

2.1.2. Redes neuronales

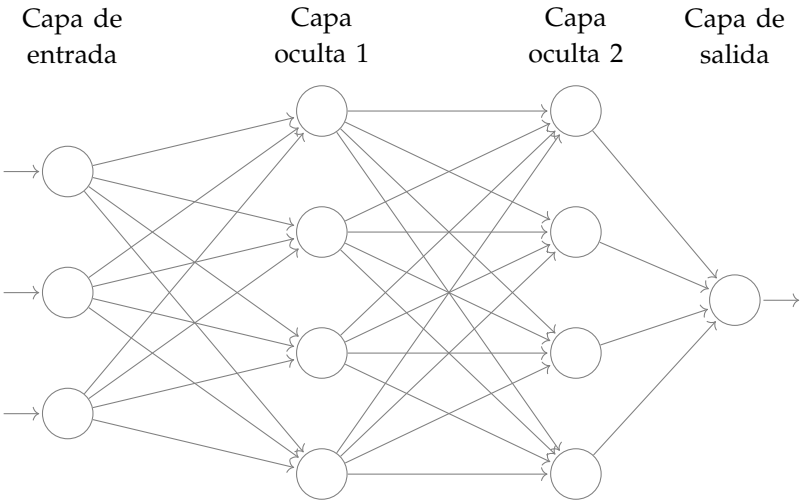

Figura 2.1: Ejemplo de un Perceptrón multicapa con 3 neuronas en la capa de entrada, 2 capas ocultas con 4 neuronas cada una y una capa de salida con una única neurona.[1]

Aunque hoy en día las redes neuronales son el modelo más común para aplicar el aprendizaje automático, la primera noción de red neuronal se puede remontar a 1958, con el Perceptrón [3]. Esta estructura, junto al algoritmo de retropropagación de errores [4] introducido en 1986 (*Backprop*), son los pilares fundamentales de las redes neuronales modernas.

Para una mejor comprensión del funcionamiento subyacente de estas, se introduce a continuación la red más básica que implementa estos conceptos, el Perceptrón multicapa. Se puede ver un ejemplo de su estructura en la figura 2.1. Formalmente, se define como una red neuronal prealimentada o *feed-forward* (sin ciclos en su grafo de conexiones). Su estructura básica se divide en una capa de entrada, varias capas ocultas y una capa de salida. Cada una de estas capas está formada por un número de nodos, referidos comúnmente como neuronas, que están completamente conectadas a todas las neuronas de la siguiente capa[2]. Las ecuaciones que definen los valores que toman las neuronas en la capa oculta $\mathbf{h}^{(i)}$ son:

$$\mathbf{h}^{(i)} = f^{(i)}(z^{(i)}) \tag{2.1}$$

$$z^{(i)} = \begin{cases} \boldsymbol{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)} & si\ i = 1 \\ \boldsymbol{W}^{(i)}\mathbf{h}^{(i-1)} + \mathbf{b}^{(i)} & en\ cualquier\ otro\ caso \end{cases} \tag{2.2}$$

- $\mathbf{x}$ es el vector inicial de características suministradas a la capa de entrada.
- $\boldsymbol{W}^{(i)}$ es la matriz que contiene los pesos de las conexiones entrantes a las neuronas de la capa oculta i.
- $f^{(i)}$ es el vector de funciones de activación de la capa. Una función de activación $f(\cdot)$ está definida normalmente por una función no lineal que transforma la salida de una neurona definida a su vez por una función $z^{(i)}$. De esta forma, el modelo pueda aprender relaciones no lineales. Ejemplos de estas funciones de activaciones son la sigmoide, la tangente hiperbólica o la ReLU [5].

[1] Basado en: https://tex.stackexchange.com/q/362238.
[2] Las neuronas de la capa de salida son la única excepción a esta norma.

- $\mathbf{b}^{(i)}$ es el vector de umbrales (*bias*). Estas constantes tienen la función de añadir parámetros a aprender a cada neurona del modelo para que este pueda modificar el valor correspondiente a la función de salida de cada neurona.

Globalmente, la red neuronal define una función discriminante compuesta por otras funciones. Se quiere que los valores óptimos de los parámetros ($\boldsymbol{\theta}$) que la definen, que en este caso son los pesos y *bias* descritos anteriormente, sean aprendidos por esta con el fin de optimizar una función objetivo de coste o pérdida $\mathcal{L}(\cdot)$. En problemas de regresión, esta función se suele definir como el error absoluto medio (MAE o $L1$):

$$MAE = \frac{1}{n}\sum_{i=1}^{n} \mid y_i - x_i \mid \tag{2.3}$$

o el error cuadrático medio (MSE o $L2$):

$$MSE = \frac{1}{n}\sum_{i=1}^{n} (y_i - x_i)^2 \tag{2.4}$$

para los valores predichos x_i y los valores reales y_i. El objetivo pues, es minimizar esta función. En lo que a esto se refiere, el cálculo viene a partir de descenso por gradiente, donde para la iteración i del modelo, los parámetros θ son:

$$\theta_i = \theta_{i-1} - \rho \nabla_\theta \mathcal{L}(\theta) \tag{2.5}$$

siendo ∇_θ el gradiente de la función de pérdida respecto a sus parámetros y ρ un factor de aprendizaje que regula el ratio de actualización de los parámetros. La premisa de este método se puede resumir en el cálculo de unos parámetros que, sabiendo la inclinación de la pendiente de la función y yendo en dirección contraria, nos acerca más rápido a un buen mínimo de la función de pérdida.

El ya mencionado algoritmo *Backprop* es el encargado del cálculo del gradiente. Los detalles de este algoritmo están fuera del alcance de este trabajo, pero se recomienda la lectura de la § 6.5 de [6] para una mejor comprensión de este.

Es importante tener en cuenta que el coste de calcular el gradiente es alto, especialmente a medida que aumenta el tamaño de los conjuntos de datos y la red, por lo que en redes neuronales modernas normalmente se suele utilizar una variación *mini-batch* del algoritmo, donde los datos de entrenamientos se dividen en bloques (*batches*) y en cada uno de ellos se calcula su gradiente y consecuente actualización de pesos, resultando en una aproximación del gradiente global de la función del modelo.

Otro hecho interesante a destacar es que, debido a la forma en que se calcula el gradiente, a veces los valores calculados pueden resultar ser muy pequeños. En estos casos, algunos pesos pueden no actualizarse de forma significativa mediante *Backprop* y, en el peor de los casos, esto puede resultar en una paralización de la red, donde los pesos de la red no pueden cambiarse y no se puede seguir entrenando el modelo. Este fenómeno es denominado como el problema de desvanecimiento del gradiente [7] [8].

2.1.3. Redes Recurrentes y Redes Convolucionales

A lo largo de los años se han desarrollado diferentes variaciones de redes neuronales a partir de las ideas planteadas por el Perceptrón multicapa.

Por ejemplo, las Redes Neuronales Recurrentes (RNN) [9] permiten la existencia de ciclos en su grafo de conexiones con la finalidad de poder transmitir información de

instantes de tiempo anteriores a través de la red. Formalmente, de forma similar a la ecuación 2.2, para calcular el valor de una de las neuronas de las capas ocultas $\mathbf{h}^{(i)}$ para un instante de tiempo t tenemos que:

$$z_t^{(i)} = \boldsymbol{W}^{(i)}\mathbf{h}_t^{(i-1)} + \boldsymbol{V}^{(i)}\mathbf{h}_{t-1}^{(i)} + \mathbf{b}^{(i)} \tag{2.6}$$

donde se añade una matriz de pesos $\boldsymbol{V}$ que controla la influencia que tienen el estado oculto de la capa en el instante anterior de tiempo $\mathbf{h}_{t-1}^{(i)}$ para calcular el actual $\mathbf{h}_t^{(i)}$.

Además, dentro del entorno de las RNNs, se han propuesto diferentes unidades especiales como LSTMs [10] o GRUs [11]. Estas han recibido un amplio uso como una forma eficaz para modelar dependencias a largo plazo y aliviar el problema del desvanecimiento del gradiente en RNNs.

Otro modelo popular dentro del campo es el de las Redes Neuronales Convolucionales (CNN). Su origen se remonta a los años 80 con el Neocognitron [12], pero su popularidad no empezó a aumentar hasta su aplicación en trabajos como [13] para el procesamiento de imágenes. Esta popularidad se disparó a partir del 2010, cuando CNNs como AlexNet [14] consiguieron resultados previamente no vistos aprovechando nuevas técnicas y el poder computacional de GPUs, idea que se había estado popularizando por trabajos como [15] o [16].

En términos estructurales, las CNNs están compuestas de forma similar a las redes tradicionales, constituyéndose una entrada, una salida y una serie de capas ocultas. Formalmente, como se define en § 9 de [6], las CNNs son modelos de redes neuronales que tienen una topología en forma de cuadrícula (tensor n-dimensional) y que utilizan el concepto matemático de convolución en lugar de la multiplicación de matrices de redes tradicionales para el cálculo de los valores de sus capas ocultas. En la figura 2.2 se ilustra un ejemplo de una serie de convoluciones (denotada por $*$) sobre una matriz $\boldsymbol{M}$ (tensor 2D) de entrada. Cada convolución es aplicada a través de un filtro o *kernel* $\boldsymbol{K}$ que recorre gradualmente la matriz. Otros parámetros a tener en cuenta en este proceso son el paso (*stride*) y el relleno (*padding*). El primero hace referencia al rango de desplazamiento de la matriz en cada paso, siendo 1 en este ejemplo. El segundo hace referencia a valores adicionales en los bordes de la entrada que, dependiendo de las dimensiones de la entrada y el *kernel*, pueden resultar útiles si no se quiere perder información cercana a estos.

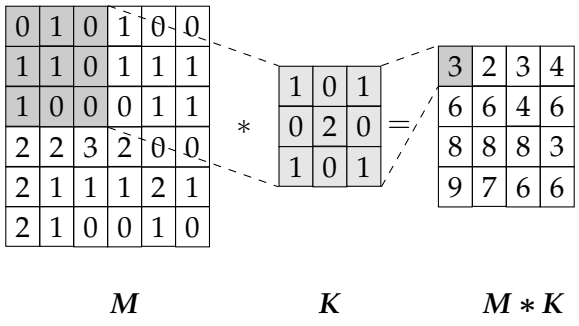

Figura 2.2: Ejemplo de una convolución sobre una matriz. [3]

En el proceso de entrenamiento de las CNNs, los propios *kernels* actúan como los pesos del modelo a aprender, por lo que inicialmente sus valores son arbitrarios o se obtienen a partir de algún método heurístico.

Respecto a las capas ocultas, en los CNNs se suelen definir distintos niveles [17]:

[3] Adaptado de https://github.com/PetarV-/TikZ/tree/master/2D%20Convolution.

- **Capas convolucionales**: Estas capas aplican el proceso de convolución explicado en esta sección.
- **Capas de agrupación o *pooling***: Estas capas combinan los valores de entrada correspondientes con la salida de las neuronas de la capa anterior, reduciendo las dimensiones de los datos (*downsampling*). Una de las funciones más comunes de *pooling* es la de *max-pooling*, en la que se toma el valor máximo de cada grupo obtenido por un *kernel*.
- **Capas completamente conectadas entre sí (*fully-connected*)**: Estas capas actúan de la misma forma que en un Perceptrón multicapa. Debido a que normalmente estas capas suelen estar previamente conectadas por capas convolucionales que han resultado en tensores multidimensionales, es necesario un preprocesamiento de la entrada que transforme esta en un vector unidimensional. Este proceso se suele denominar en la literatura como aplanamiento (*flatten*).

En la figura 2.3 se muestra un ejemplo de una hipotética CNN para un problema de clasificación. Se toma de entrada una imagen de 128 píxeles de ancho y alto, con una profundidad de 3 (canales RGB). Esta es procesada por la red, transformándose mediante capas convolucionales y *pooling*, y llegando a las dos últimas capas de la red (*fully-connected*) devolviendo un vector con 8 pesos, donde cada uno de estos representa la probabilidad de que la imagen corresponda con una de las clases disponibles en el corpus de datos.

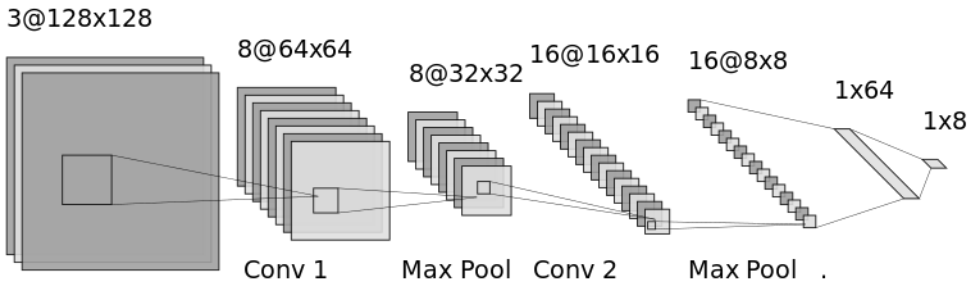

Figura 2.3: Ejemplo de una CNN genérica.[4]

2.1.4. Redes Generativas Adversariales

Las Redes Generativas Adversariales (GAN) fueren introducidas en [18]. Originalmente propuestas como una forma de aprendizaje no supervisado, su uso se ha visto expandido a otros entornos.

De forma intuitiva, la base en el que este sistema está estructurado se puede ver como un juego de suma cero *minimax*, en el que la ganancia de un agente es la pérdida de otro y cada uno busca minimizar las posibles pérdidas en el peor de los casos. En el caso de las GANs, estas dos partes que compiten entre sí consisten en un generador G y un discriminador D. El primero intenta engañar al segundo generando muestras que se acercan a la distribución real de los datos de entrenamiento, y este último tiene la tarea de distinguir estas muestras falsas de los ejemplos reales de la distribución. De esta forma, en cada iteración del entrenamiento los dos componentes van mejorando progresivamente en su trabajo con el objetivo de triunfar el uno sobre el otro. En la figura 2.4 se puede ver el esquema general descrito previamente.

[4]Imagen generada mediante NN-SVG: https://alexlenail.me/NN-SVG/.

En el ámbito de la síntesis del habla neuronal, las GANs actúan como base para la construcción de *vocoders* de alta fidelidad, que son introducidos en la sección 2.2, siendo su uso el estándar *de facto* para estos sistemas de síntesis.

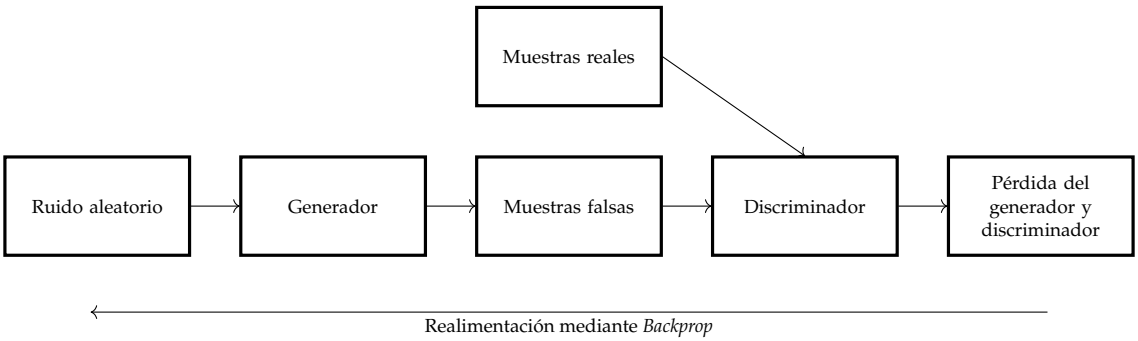

Figura 2.4: Esquema del funcionamiento de un GAN.

2.1.5. Problemas Seq2Seq y el mecanismo de atención

En el aprendizaje automático, los problemas donde la entrada y salida son secuencias de longitud variable se suelen clasificar como *Sequence-2-Sequence* (Seq2Seq). Un ejemplo de este tipo de problemas son los del procesamiento de lenguajes naturales, como el TTS, donde la entrada corresponde a una secuencia de texto y la salida a una secuencia de características acústicas.

En la mayoría de este tipo de problemas, es importante mantener parte de la información de entrada a lo largo de múltiples iteraciones. Por ejemplo, en un problema de MT, donde se ha omitido el sujeto en una oración, interesaría que el sistema accediese a quien se está refiriendo en esta frase observando el resto de información de la entrada.

Para tratar este tipo de problemas se propuso en trabajos como [19] el uso de modelos con una estructura basada en un codificador y un decodificador. El codificador normalmente consiste de una red neuronal recurrente formada por componentes como LSTMs [10] o GRUs [11]. Esta procesa la entrada y produce una proyección a un espacio multidimensional fijo N y abstracto (espacio latente) que contiene algunas de sus características. Esta representación, denominada como los estados ocultos del codificador ($\mathbf{h}$), es procesada por el decodificador (normalmente otra RNN), generando la salida deseada. Representaciones como $\mathbf{h}$, que comprimen información sobre algunas características de entrada, toman el sobrenombre de *embeddings*.

Sin embargo, un problema surge con este tipo de estructura. La naturaleza dimensional reducida y fija de los estados ocultos $\mathbf{h}$ restringe la memoria que tiene el sistema sobre la entrada. Esto no solo afecta a la velocidad de inferencia, sino que se dan malos resultados en secuencias donde el tamaño es lo suficientemente grande y hay dependencias a largo plazo en la frase.

Para resolver este problema, en [20] se introdujo el concepto de atención. En este, se sustituye el vector de estados ocultos $\mathbf{h}$ por un vector de contexto $\mathbf{c}$ que es calculado dinámicamente para cada iteración del decodificador tal que:

$$\mathbf{c_i} = \sum_{j=1}^{N} \alpha_{ij} h_j \tag{2.7}$$

donde los pesos $\alpha_{i,j}$ son calculados para cada estado oculto h_j mediante la función *softmax*:

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{N} \exp(e_{ik})} \tag{2.8}$$

y en problemas Seq2Seq

$$e_{ij} = a(s_{i-1}, h_j) \tag{2.9}$$

siendo s_{i-1} el estado oculto del sistema tras la última iteración del decodificador y $a(\cdot)$ una función de puntuación que devuelve el grado de atención o energía e_{ij}, implementada normalmente por una red neuronal prealimentada.

Una forma alternativa de ver el mecanismo de atención, como se define en [21], es como una búsqueda sobre un diccionario. En este se compara un valor de consulta $\boldsymbol{Q}$ sobre un conjunto de pares clave-valor $(\boldsymbol{K}, \boldsymbol{V})$, devolviéndose una suma ponderada de los valores, donde el peso asignado a cada valor es calculado mediante una función que mide la compatibilidad de la consulta con la clave correspondiente. Por ejemplo, en el caso anterior la función de compatibilidad sería la ecuación 2.8.

2.1.6. Redes Transformers

La arquitectura de redes Transformers, originalmente introducida en [21], surgió como un substituto al uso de RNNs usando el mecanismo de atención introducido en el apartado 2.1.5. La estructura general codificador-decodificador propuesta en [21] de este tipo de red neuronal se puede ver en la figura 2.5, y a continuación, se introducen los conceptos principales presentados en este artículo para entender su funcionamiento:

- **Función de puntuación:** Tomando la definición de atención como búsqueda en un diccionario presentada en el apartado 2.1.5, se eligen matrices para representar $\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}$ con dimensiones d_q, d_k, d_v, siendo la función de puntuación $a(\cdot)$ el producto escalar, escalado por un factor $\frac{1}{\sqrt{d_k}}$ y una función *softmax*, tal que:

$$Atención(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = softmax\left(\frac{\boldsymbol{Q}\boldsymbol{K}}{\sqrt{d_k}}\right)\boldsymbol{V} \tag{2.10}$$

- **Atención multicabeza:** En vez de calcular una sola función de atención para cada consulta $\boldsymbol{Q}$, se calculan varias funciones de atención (*heads*) mediante proyecciones para cada conjunto de consultas, clave y valor $\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}$, para luego volver a proyectarse todas estas en un espacio común:

$$MultiHead(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = Concat(head_1, \ldots, head_h)\boldsymbol{W}^{\boldsymbol{O}} \tag{2.11}$$

y

$$head_i = Atención(\boldsymbol{Q}\boldsymbol{W}_{\boldsymbol{i}}^{\boldsymbol{Q}}, \boldsymbol{K}\boldsymbol{W}_{\boldsymbol{i}}^{\boldsymbol{K}}, \boldsymbol{V}\boldsymbol{W}_{\boldsymbol{i}}^{\boldsymbol{V}}) \tag{2.12}$$

donde $\boldsymbol{W}_{\boldsymbol{i}}^{\boldsymbol{Q}}, \boldsymbol{W}_{\boldsymbol{i}}^{\boldsymbol{K}}, \boldsymbol{V}_{\boldsymbol{i}}^{\boldsymbol{V}}$ son las matrices de proyecciones correspondientes y $\boldsymbol{W}^{\boldsymbol{O}}$ es la matriz proyección del espacio común. En estas matrices, el valor y dimensión de estas es aprendido durante el entrenamiento de la red neuronal.

Este procedimiento permite que el sistema logre adaptarse e identifique similitudes y dependencias en varios rangos dimensionales. Además, al dividir el cálculo de la atención, se reduce el número de operaciones secuenciales, permitiendo un mayor grado de paralelización del modelo, y por lo tanto, unas velocidades superiores de entrenamiento.

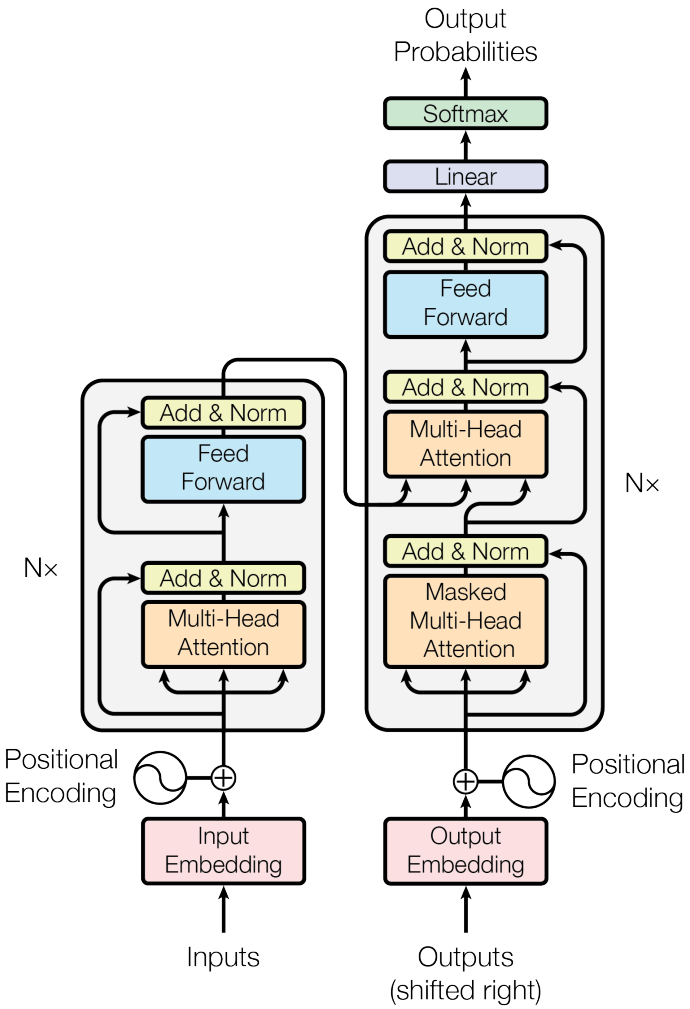

Figura 2.5: Estructura original del Transformer de [21].

- **Autoatención:** La previamente explicada atención multicabeza es implementada de varias forma en la estructura del Transformer. Por una parte, en capas de la red donde es preferible simular el mecanismo de atención simple de redes codificador-decodificador, se procesan las consultas de la capa anterior del decodificador, donde las claves y los valores provienen de la salida actual del codificador. De esta forma, cada posición del decodificador puede atender a todas las posiciones de la secuencia de entrada simultáneamente. Por otra parte, dentro del codificador y el decodificador se definen capas de autoatención, donde todas las consultas, claves y valores provienen del mismo lugar, en este caso, la salida de la capa anterior del codificador o decodificador respectivamente.

- **Codificación posicional:** En RNNs el orden de los *tokens* de entrada puede inferirse fácilmente, ya que de forma implícita está definido con el análisis secuencial que hay *token* a *token*. Sin embargo, con la estructura del Transformer esta información desaparece y se ha de implementar algún otro mecanismo que extraiga la posición relativa o absoluta de los *tokens* en la secuencia de entrada. Para esto, los autores originales proponen una representación donde la posición de cada *token* se incluye en un vector cuyos valores se calculan:

$$PE_{pos,2i} = sin\left(\frac{pos}{G^{2i/d_{model}}}\right) \tag{2.13}$$

$$PE_{pos,2i+1} = cos\left(\frac{pos}{G^{2i/d_{model}}}\right) \tag{2.14}$$

donde pos_i corresponde a la posición de dimensión i en el vector, representado mediante una onda, donde su longitud toma un valor dependiendo de G (los autores proponen 10.000) que es representado dentro de una progresión geométrica tal que $[2\pi, 2\pi G]$. Para una visión más detallada de este mecanismo y su razonamiento, remitimos al lector a [22].

2.2 Procesamiento de lenguajes naturales

El procesamiento de lenguajes naturales (NLP) engloba el estudio de métodos computacionales para el tratamiento del lenguaje humano y sus usos en interacciones persona-ordenador. Dentro de esta disciplina se incluyen campos como el reconocimiento óptico de caracteres (OCR), el ASR, el MT o el TTS, siendo este último el objeto de estudio de este trabajo. En los últimos años se ha podido observar un cambio de paradigma en los sistemas de NLP. Desde los años 80 predominaba el uso de métodos estadísticos en esta área, pero en la última década se ha generalizado el uso de redes neuronales gracias al poder computacional y paralelización derivados del uso de GPUs. [23]

2.2.1. Síntesis del habla

La síntesis del habla (TTS), tiene como objetivo sintetizar el habla de forma natural e inteligible a partir de texto [24]. Previamente al auge de las redes neuronales, el campo estaba dominado por el uso de modelos basados en síntesis concatenativa (CSS) y síntesis estadística paramétrica (SPSS), que son explicados brevemente a continuación.

El primero de ellos se basa en la unión y procesamiento de fragmentos vocales. Estos son extraídos de una base de datos segmentada en unidades básicas, donde un algoritmo selecciona aquellos que mejor se ajustan al sonido que se quiere sintetizar [25]. Aunque este método da resultados similares de inteligibilidad y timbre a la fuente acústica original, surgen dos grandes inconvenientes con este enfoque. Por una parte, el tamaño requerido en las bases de datos ha de ser lo suficientemente grande para que se cubran todas las combinaciones posibles de unidades correspondientes a las palabras que quieren generarse. Por otra parte, las voces generadas no suelen abarcar la prosodia de la voz natural, resultando en voces monótonas y sin emoción.

Es por estas razones que los modelos SPSS surgieron como una alternativa para mitigar estos defectos, sustituyendo el uso de segmentos de audio pregrabados por modelos estadísticos [24]. Su estructura suele dividirse en tres componentes principales:

- Un analizador de texto, con funciones como la normalización del texto de entrada y la extracción de sus fonemas.
- Un predictor de parámetros, que normalmente se denomina *modelo acústico* y suele estar basado en modelos ocultos de Markov (HMM).
- Un *vocoder*, encargado de sintetizar el habla a partir de las características acústicas extraídas.

Mediante esta estructura, se generan las características acústicas necesarias para sintetizar el habla y, teniendo en cuenta los valores obtenidos, se reconstruye el audio deseado. La síntesis de voz resultante es más natural y se consigue un coste reducido de datos

de entrenamiento. Además los parámetros de entrada del modelo pueden ajustarse fácilmente. Desafortunadamente, también es más común la aparición de interferencias como ruido o zumbidos, resultando en un menor grado de inteligibilidad.

Con el objetivo de mejorar el rendimiento de estos modelos, hacia el 2010 algunos trabajos como [26] o [27] empezaron a introducir el uso de redes neuronales profundas como sustitutos a HMM, consiguiendo unos resultados muy prometedores. Esto eventualmente condujo al establecimiento de la síntesis del habla neuronal como el estado del arte en al área.

2.2.2. Síntesis del habla neuronal

Siguiendo las implementaciones en SPSS, los primeros sistemas completos de síntesis de habla neuronal empezaron a aparecer hacia el 2016, con la implementación de modelos acústicos y *vocoders* neuronales autorregresivos *end-to-end*. Wavenet [28], considerado habitualmente como el primer *vocoder* neuronal, seguido de otros modelos como Tacotron 1/2 [29] [30], Deep Voice 3 [31] o FastSpeech 1/2 [32] [33] son trabajos pioneros en el área del TTS neuronal.

Su éxito se puede explicar por la combinación de buenos resultados y simplificación del proceso de entrenamiento en comparación con modelos de CSS y SPSS. Todos estos modelos suelen tener en común una menor atención en el analizador de textos y una aproximación en dos etapas a su estructura:

- Una primera etapa donde se transforma directamente del texto (caracteres o fonemas) a una representación intermedia a través de un modelo acústico. Esta representación suele ser en la forma de un espectrograma Mel, que refleja mejor la forma en que los seres humanos perciben valores tonales como equidistantes. Adicionalmente, se suele aplicar el logaritmo natural para que los distintos niveles de energía del espectrograma se asemejen a los percibidos por los seres humanos.

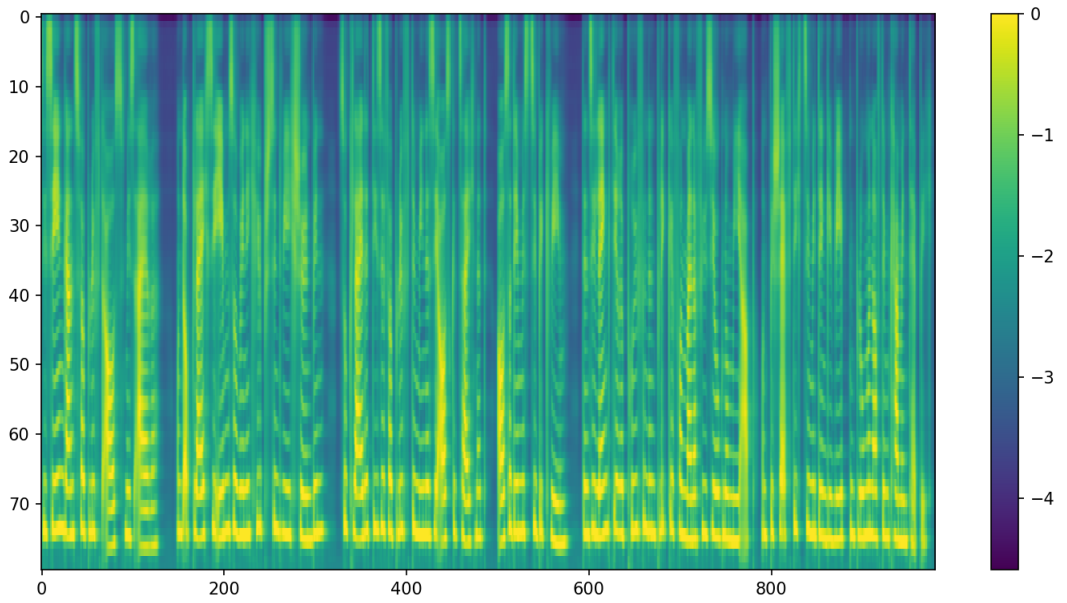

Figura 2.6: Ejemplo de un log-espectrograma Mel escalado con el logaritmo natural. El eje vertical representa el número de bandas Mel utilizadas y el horizontal los *frames* del audio. La coloración indica el valor escalado de dB/Hz.

- Una segunda etapa donde se toma el espectrograma y se reconstruye el audio representado mediante un *vocoder* neuronal.

En el entrenamiento, el modelo acústico es entrenado sobre pares de textos y audios, buscándose minimizar la función de pérdida del sistema entre la diferencia de los espectrogramas reales y los generados por el modelo. El *vocoder* por su parte, es entrenado solamente con audios y sus representaciones a espectrogramas.

2.2.3. Métodos de evaluación de la síntesis del habla

Tradicionalmente, la evaluación de sistemas dentro del aprendizaje automático está basado en métricas que están bien definidas, como es el caso del WER en ASR [34] o el BLEU en MT [35]. En comparación con otras subáreas del NLP, el TTS no cuenta con ninguna norma establecida capaz de medir óptimamente la calidad del habla. Las expectativas que tienen los oyentes al escuchar estos sistemas o la naturaleza de la variabilidad en la producción del habla humana suelen usarse como ejemplo de algunos de los obstáculos a los que se enfrenta la evaluación del TTS. [36]

De este modo, por lo general se suelen utilizar métodos de evaluaciones subjetivas, como el *Mean Opinion Score* o MOS [37] [38], basado en una escala de Likert típicamente de 1 (peor) a 5 (mejor), o tests A/B entre audios sintetizados por distintos modelos. La valoración se realiza tras la reproducción de una o varias muestras reales o sintéticas, donde se les pide a los participantes su opinión sobre la calidad del habla que acaban de escuchar.

CAPÍTULO 3
Síntesis de voz neuronal no autorregresiva

En este capítulo se analiza y presenta el modelo de síntesis de voz neuronal inicial que se ha utilizado a lo largo de este trabajo. Para entender el razonamiento de su elección, primero se explican los problemas asociados con modelos de síntesis de voz autorregresivos, que hasta hace poco dominaban el campo. Además, se presentan los resultados de un entrenamiento de este modelo inicial sobre un corpus en inglés.

3.1 Hacia la síntesis de voz neuronal no autorregresiva

3.1.1. Los modelos autorregresivos y sus problemas

Hasta hace muy poco, el uso de modelos neuronales puramente autorregresivos como Transformers era la opción más habitual para construir modelos de síntesis de voz neuronal de alta calidad. En la mayoría de estos, los modelos acústicos hacen uso de un codificador y decodificador, apoyándose en un mecanismo de atención como se ha visto en los apartados 2.1.5 y 2.1.6. Por otra parte, los *vocoders* tienden a variar más en su estructura.

Esta arquitectura, sin embargo, plantea una serie de problemas. Entre los más notables, los investigadores del campo identifican que:

- Los modelos acústicos autorregresivos suelen sufrir de un cierto grado de inestabilidad durante la fase de inferencia. Ejemplos de esta falta de robustez son la omisión y repetición de fonemas y palabras o la generación de habla incomprensible [39] [40]. Estos problemas se atribuyen típicamente a errores de alineamiento del mecanismo de atención derivados del uso de *teacher-forcing* [41] [42] como mecanismo de entrenamiento. Este genera una discrepancia entre las fases de entrenamiento e inferencia conocido como *exposure bias* [43]. Este tipo de inestabilidades se pueden observar especialmente en situaciones donde el audio objetivo a ser generado tiene características que están fuera del dominio del conjunto de datos de entrenamiento.
- Los modelos acústicos y *vocoders* son poco paralelizables y tienen una velocidad de inferencia lenta en sus respectivas tareas. Adicionalmente, los *vocoders* tienden a carecer de opciones capaces de controlar características generales del habla como el ritmo o la prosodia [44].

Esto ha llevado a que en los últimos años se hayan desarrollado nuevos modelos que abandonan la autorregresión en favor de otros mecanismos.

3.1.2. Eliminación del mecanismo de atención en modelos acústicos

Como hemos explicado, el mecanismo de atención es una herramienta muy potente para aportar información contextual a los modelos de TTS, pero su uso puede implicar la aparición de algunos defectos en el audio generado, especialmente durante el tiempo de inferencia del modelo.

Es por esto que, trabajos como [32] o [45] remplazan el uso de la atención para mejorar la robustez durante la fase de inferencia. En estos se proponen un enfoque similar a modelos de SPSS clásicos, realizando una predicción explícita de las duraciones a nivel de fonema (en *frames*), y posteriormente replicando (mediante *upsampling*) la salida del codificador a partir de estas duraciones.

Estos modelos utilizan valores de referencia (*ground-truth*) extraídos a partir de los alineamientos entre la entrada (fonemas) y la salida (*frames*) para entrenar un predictor de duración junto al resto del modelo [32]. Para estimar dichos alineamientos, se suele hacer uso de un modelo externo capaz de extraer esta información. Típicamente, este suele ser un modelo de alineamiento forzado como [46] (ASR) o un modelo de TTS basado en el mecanismo de atención como [29].

Formalmente, a partir los estados ocultos del codificador $\mathbf{h} = [\mathbf{h}_1, \ldots, \mathbf{h}_N]$ de longitud N y las predicciones de duración de los fonemas $d = [d_1, \ldots, d_N]$, $d_i \in \mathbb{Z}$ extraídas previamente por un modelo o alineador externo E, se replica $\mathbf{h}_i$ una cantidad de d_i veces pasando de una secuencia de longitud N (número de fonemas) a una secuencia de longitud M (número de *frames* del espectrograma) teniendo en cuenta que $\sum_{i=1}^{N} d_i = M$. Este esquema se puede visualizar en la figura 3.1.

Adicionalmente, mediante esta formulación de los fonemas, la velocidad del habla puede controlarse durante la inferencia del modelo mediante un hiperparámetro que modifique d antes de la expansión de los estados $\mathbf{h}$.

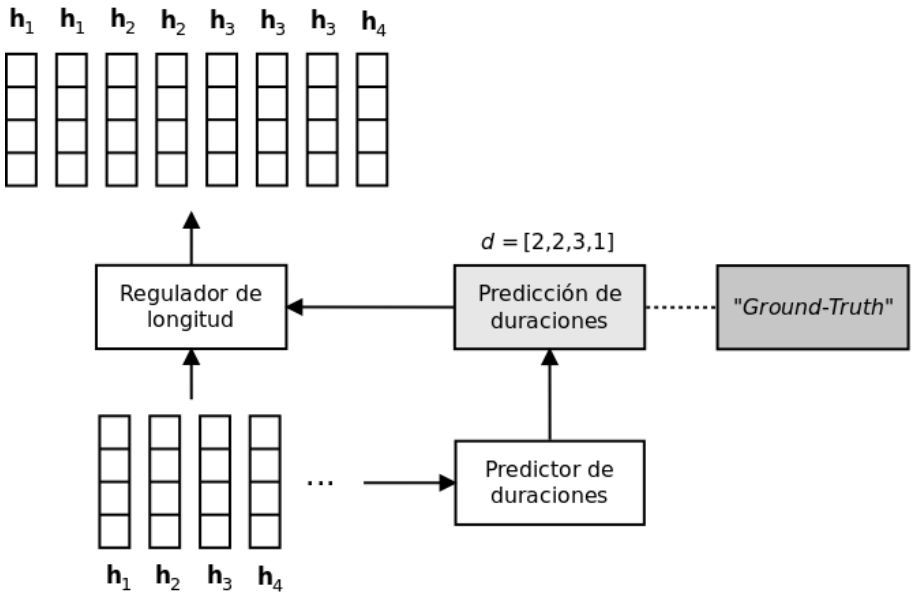

Figura 3.1: Esquema del *upsampling* de fonemas mediante la ampliación de estados ocultos y el predictor de duración. Las líneas discontinuas indican conexiones que solo están activas durante del entrenamiento. Adaptado de [1].

3.1.3. Predicción del tono y la energía

De forma similar a otros procesos generativos, a grandes rasgos la síntesis del habla puede verse como un proceso de descompresión de información donde se pasa de texto a audio. En este tipo de procesos suele aparecer el llamado problema *one-to-many*, donde para una entrada dada puede haber potencialmente una cantidad de salidas infinitas, dando lugar a modelos que devuelven resultados subóptimos.

En el contexto de modelos TTS autorregresivos este problema es aliviado parcialmente debido al procedimiento de entrenamiento *teacher-forcing* mencionado previamente, pero este no es el caso en los modelos de TTS no autorregresivos. En estos, para reducir la gravedad de este problema, se suele proporcionar información adicional sobre la variabilidad del habla durante el entrenamiento e inferencia del modelo [33] [47]. Un procedimiento habitual es la extracción de los correspondientes valores de tonos y energía por *frame* de los fragmentos originales de audio, que posteriormente se pueden incluir como *embeddings* en los estados codificados h.

De forma similar al caso de los fonemas, el modelado explícito del tono y energía puede proporcionar un control más preciso de estos parámetros durante la inferencia del modelo mediante la introducción de parámetros que modifiquen el valor de predicción de estos.

3.2 Modelo inicial

A continuación se presenta el modelo inicial utilizado en este capítulo. La implementación de las redes neuronales de este se apoya mediante el uso de la popular librería de aprendizaje automático PyTorch [48].

3.2.1. Alineador: Predictor de duración de fonemas

Basándose en los trabajos más recientes en modelos de TTS no autorregresivos, para hacer uso del predictor de duración de fonemas propuesto en la sección 3.1.2 es necesario que previamente se haya calculado la duración real (*ground-truth*) a nivel de fonemas de los audios de los corpus de datos utilizados.

Para ello, tomando como inspiración a DeepForceAligner[1], se plantea el uso de un modelo *autoencoder* para la tarea como en [1], ilustrándose su estructura en la figura 3.2. Esta arquitectura se puede ver como un subtipo de los ya estudiados modelos codificador-decodificador, solo que en este caso el dominio de entrada y el de salida son iguales [49].

Para extraer los alineamientos correctos entre fonemas y espectrogramas, el modelo hace uso de una función de pérdida auxiliar CTC [50], utilizando el mismo corpus de datos del modelo general. En comparación con otros modelos basados en atención, esta arquitectura proporciona una mayor robustez y una convergencia significativamente más rápida que estos. Además, en comparación a otras herramientas de alineamiento forzado, este enfoque resulta en mejores alineamientos al realizarse el entrenamiento sobre la misma tarea que el modelo general (generar un espectrograma) en comparación con un alineador típico, donde se realiza una tarea de clasificación de ASR. Asimismo, el hecho de que se proporcione en este caso el mismo corpus en el que el modelo acústico se entrena también beneficia a que se obtengan mejores alineamientos.

En la arquitectura propuesta, en particular, se proponen dos módulos que actúan respectivamente como el codificador y decodificador del *autoencoder*. El primero de estos

[1] https://github.com/as-ideas/DeepForcedAligner

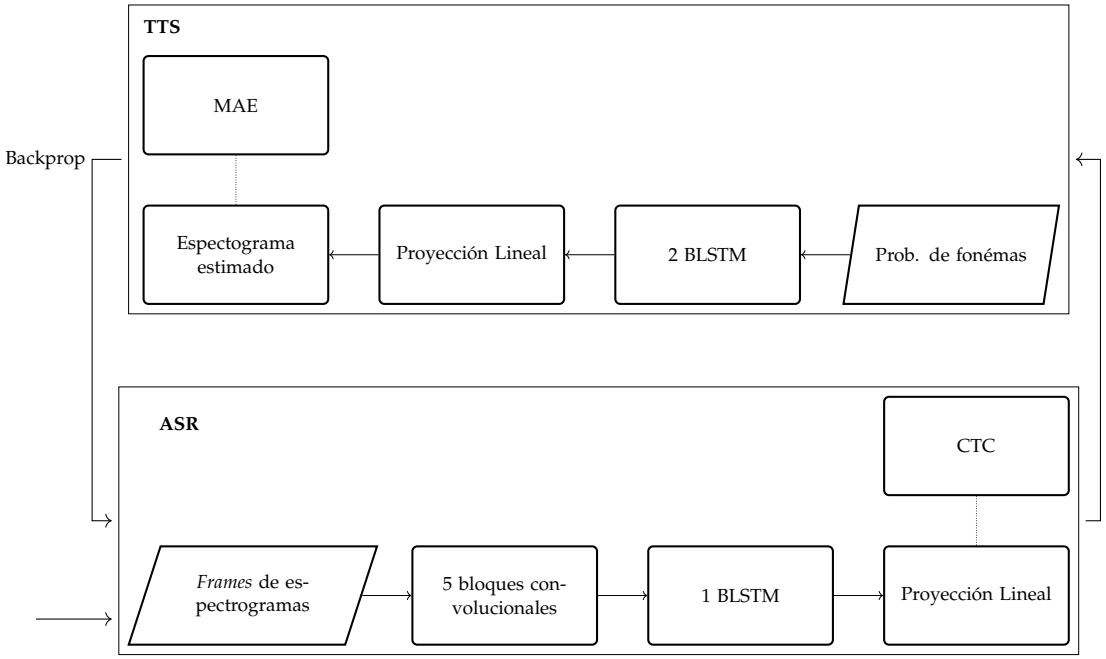

Figura 3.2: Esquema del predictor de fonemas.

actúa como un modelo de ASR, donde se pasa de los espectrogramas a una representación de la probabilidad de los fonemas que el modelo ha asociado al espectrograma de entrada (espacio latente). La entrada pasa por 5 capas convolucionales unidimensionales, seguidas de una normalización por *batches* y funciones de activaciones ReLU. Después de estas capas, se hace uso de una capa LSTM [10] bidireccional (BLSTM) [51], seguido de una proyección lineal con funciones de activación *softmax* que da la representación al espacio latente, al que se le aplica la función de perdida auxiliar CTC respecto a las secuencias reales de los fonemas del espectrograma.

A continuación, se pasa al segundo módulo propuesto, que actúa como TTS, intentando reconstruir el espectrograma original a partir de la salida del primer módulo. Para ello, la entrada pasa por 2 capas BLSTM seguidas de una proyección lineal a la dimensión del espectrograma, generando la predicción del espectrograma de entrada, que junto al espectrograma real se utilizan para calcular el MAE y realimentar al módulo de ASR mediante *backprop*. De esta forma, el gradiente del error de reconstrucción del espectrograma se propaga ayudando a refinar los alineamientos de ASR para la tarea de TTS.

Durante la inferencia de este modelo, se descarta el decodificador (TTS) y se hace uso exclusivo del codificador (ASR) para extraer la matriz de atención que, para cada *frame* incluye las probabilidades de que este corresponda con todos y cada uno de los fonemas posibles del vocabulario dado. Posteriormente, se utiliza el algoritmo de Dijkstra para obtener el camino monótono más probable a partir de esta matriz y extraer la secuencia de fonemas resultante.

3.2.2. Modelo acústico

El modelo acústico base utilizado se fundamenta en ForwardTacotron[2], una variante no autorregresiva de Tacotron [29] de código abierto e inspirada en otros modelos TTS

[2] https://github.com/as-ideas/ForwardTacotron

como FastSpeech [32]. Su estructura está compuesta por dos capas *bottlenecks* (Pre-Net) y un codificador CBHG como en [29], un predictor de varianza de la duración de fonemas similar a [32], un decodificador BLSTM de 2 capas y una red convolucional residual (Post-Net) como en [30].

Esta arquitectura ha sido expandida en [1] y [52], introduciendo varias modificaciones que mejoran el rendimiento de este modelo. Su estructura puede visualizarse en la figura 3.3.

La primera de estas modificaciones es la substitución del codificador base por la estructura introducida en Tacotron 2 [30]. Esta estructura, consta inicialmente de *embeddings* aprendidos de fonemas con 512 dimensiones. Estos pasan por una pila de 3 capas convolucionales unidimensionales, seguidas de una normalización por *batches* y funciones de activación ReLU. La salida de la última de estas capas convolucionales es entonces procesada por una BLSTM de una única capa, generando los estados ocultos del codificador.

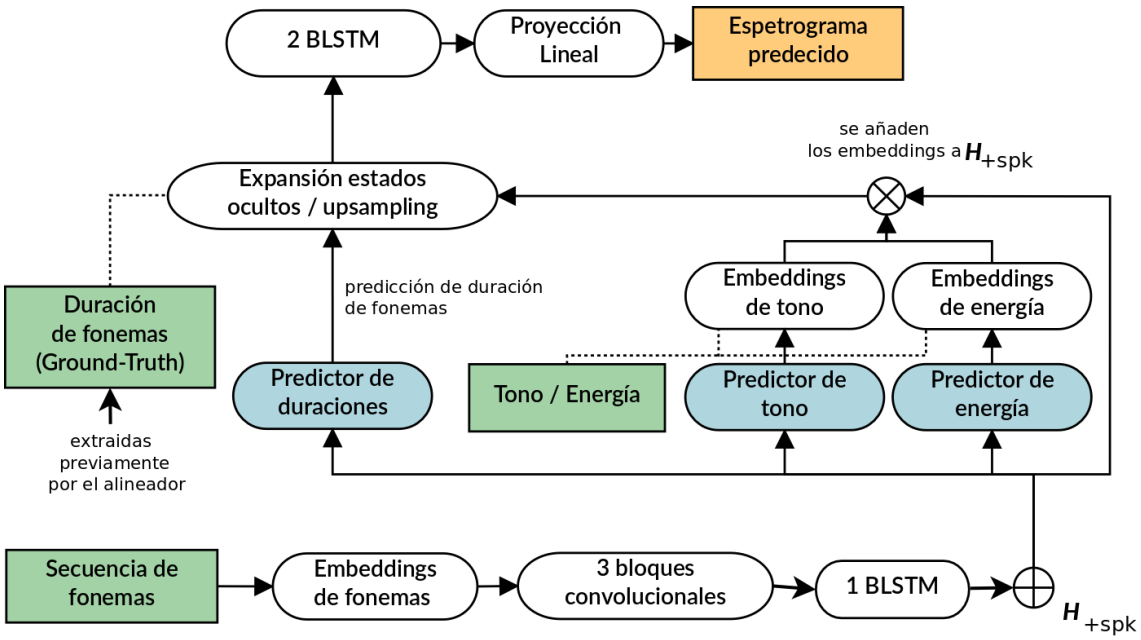

Figura 3.3: Esquema del modelo acústico base. Las líneas discontinuas indican conexiones que solo están activas durante del entrenamiento. Adaptado de [1] y [52].

También se ha eliminado la red Post-Net, al observarse en previos experimentos realizados por el grupo que esta no aportaba ninguna mejora significativa en la tarea. Además, se ha utilizado el predictor de varianza de fonemas de redes convolucionales introducido en [33], añadiendo del mismo trabajo los predictores de tono y energía que comparten esta estructura.

Es importante mencionar las funciones de pérdidas utilizadas en este modelo. Globalmente se hace uso de una combinación del MAE y el SSIM [53] sobre el espectrograma previsto y el objetivo.

3.2.3. Vocoder

En función de los problemas explicados por los modelos autorregresivos, en los últimos años se ha puesto un gran énfasis en la investigación de *vocoders* ligeros y veloces. Entre estos, hay una gran variedad de arquitecturas propuestas, desde modelos basados en flujo como WaveGlow [54] a modelos de difusión como DiffWave [55].

Sin embargo, de entre todos estos, destaca la utilización de GANs como la opción más común para el modelado de estos *vocoders* debido a sus buenos resultados y velocidad de inferencia. Algunos de los modelos más populares son MelGAN [56], Parallel WaveGAN [57], VocGAN [58] y HiFi-GAN [59].

Para este trabajo, se hace uso de una implementación de Univnet [60], un *vocoder* basado en GANs que consigue resultados comparables a los mejores *vocoders* como HiFi-GAN, pero que a su vez mejora considerablemente la velocidad de inferencia de modelos previos.

3.2.4. Preprocesamiento

Previamente al entrenamiento del modelo, todos los datos empleados se someten a un preprocesamiento.

En primer lugar, se eliminan los silencios iniciales y finales de los audios. Acto seguido, de forma similar a [61], se hace una reducción de la frecuencia de muestreo de los audios a 16kHz para su uso en el entrenamiento del modelo acústico y a 24kHz para el entrenamiento del *vocoder*, de forma que en este último genere audio a 24kHz, condicionado a las predicciones de espectrogramas de 16kHz del modelo acústico. Se ha demostrado que este esquema suele lograr un buen equilibrio entre la eficacia y estabilidad del modelo y la calidad de la voz resultante. Adicionalmente se extrae el *ground-truth* de la duración de los fonemas mediante la arquitectura descrita en la sección 3.2.1. También se calculan los valores correspondientes al tono por *frame* de los audios mediante el *toolkit* WORLD [62] [63].

Respecto a las transcripciones a texto de los audios utilizados, se normalizan (transformación a minúsculas, eliminación de caracteres especiales...) y posteriormente extraen las secuencias de fonemas correspondientes mediante la herramienta de código abierto eSpeakNG[3].

3.3 Entrenamiento y verificación del modelo inicial

Para verificar el funcionamiento correcto del modelo inicial y explorar sus capacidades, se ha realizado un entrenamiento de este sobre un pequeño corpus de datos de dominio público.

Dicho corpus es LJ Speech [64]. Este está compuesto por datos de habla en inglés, recopilados en 13.100 clips de audios de corta duración de un solo hablante que lee pasajes de 7 libros de no-ficción. Cada clip corresponde únicamente a una transcripción y están a una frecuencia de muestreo de 22kHz. La duración de los clips varía entre 1 y 10 segundos, y el corpus tiene una duración total de aproximadamente 24 horas. Dentro del campo, LJ Speech se ha popularizado como uno de los corpus más utilizados como *benchmark* para comprobar la efectividad entre distintos modelos *single-speaker*.

El proceso de entrenamiento de este modelo sigue las pautas establecidas en este capítulo, realizándose primero el preprocesado, seguido del entrenamiento en orden secuencial del alineador, modelo acústico y *vocoder*. El número de iteraciones y tamaño de *batch* escogido para el entrenamiento de cada componente puede visualizarse en la tabla 3.1. En este caso, todas estas etapas se han realizado mediante el uso de una GPU Nvidia GTX 2080Ti, resultando en un tiempo total de entrenamiento de ∼157 horas.

[3] http://espeak.sourceforge.net/

Tabla 3.1: Número de iteraciones y tamaño de *batch* para el entrenamiento de cada componente del modelo base.

	Núm. de iteraciones	Tamaño de *batch*
Alineador	22k	32
Modelo acústico	350k	64
Vocoder	400k	0k-50k: 32 50k-400k: 64

Para medir la calidad del modelo entrenado, se ha realizado una evaluación informal que recoge las primeras impresiones de este sin profundizar en ninguna métrica de TTS. En esta se han generado 50 frases, asegurándose que no se hubiesen visto por el modelo durante el entrenamiento, y se ha verificado que los audios resultantes estaban en los niveles de calidad esperados.

Una vez comprobado que se cumplían estas condiciones, y que efectivamente los resultados eran buenos, se ha pasado a entrenar a partir de este modelo inicial diferentes sistemas que se presentan en los siguientes capítulos.

CAPÍTULO 4
TTS de múltiples hablantes adaptado a casos no vistos

En este capítulo se presenta el enfoque de estudio tras el TTS de múltiples hablantes, donde se le presta especial atención a casos donde durante la inferencia se tiene poca información de estos hablantes. Para apoyar esta explicación, se expande el modelo inicial explicado en el capítulo anterior y se presenta un nuevo entrenamiento de este modelo modificado sobre un nuevo corpus en castellano.

4.1 Adaptación de modelos a múltiples hablantes

Un requisito típico en la construcción de aplicaciones de TTS es la capacidad de ofrecer el acceso a un conjunto variado de voces, de forma que la aplicación pueda adaptarse a las distintas necesidades que cada usuario pueda tener.

Por lo general, los modelos de TTS se entrenan de forma que las voces ofrecidas están restringidas a un rango predefinido. Este rango se obtiene mediante la inclusión en el conjunto de datos de entrenamiento de aquellas grabaciones de hablantes que se quiere que el modelo aprenda a imitar, de forma que posteriormente el usuario pueda indicar al modelo la voz correspondiente que desea generar.

A veces, este enfoque para construir aplicaciones TTS no siempre funciona. En ocasiones solo se dispone de una pequeña cantidad de datos del hablante que se quiere imitar, y es complicado que los modelos puedan producir buenos resultados en estos casos. Este tipo de situación se suele categorizar como un problema de aprendizaje *few-shot*, y conlleva una serie de problemas que hay que tener en cuenta a la hora de construir y entrenar los modelos. Un enfoque común en la construcción de modelos de TTS que funcionen en este tipo de contexto, como se presenta en [65], es el modelado de un sistema TTS generalista que pueda ser entrenado adicionalmente con los datos disponibles de las voces que se quieren imitar. Por lo general, los modelos actuales suelen lograr buenos resultados con unos pocos minutos de datos.

Sin embargo, hay situaciones en las que este planteamiento simplemente no es factible. De base, el entrenamiento adicional necesario para afinar el modelo introduce un coste computacional que a veces no puede ignorarse. No solo esto, sino que los datos de entrenamiento extra deben estar correctamente etiquetados, algo que no siempre puede conseguirse.

Un ejemplo de un sistema que incluye TTS, pero no permite modelado *few-shot*, es una aplicación *Speech-to-Speech* en tiempo real. En esta hay una falta de datos (e.g. segundos de habla) y un tiempo de respuesta limitado. Además, las lenguas de origen y destino son

diferentes, por lo que no se puede trivialmente trasladar el conocimiento de un modelo que ha sido entrenado con datos de una lengua específica de salida para que genere audio similar que presenta el hablante de origen en otro idioma.

Este tipo de situaciones, en las que es necesario que el modelo se adapte a clases de datos que no han sido observadas durante el entrenamiento, es lo que se conoce comúnmente como problemas de aprendizaje *zero-shot*. Dado el contexto del TTS, trabajos como [66], [67], [68] o [69] exploran este entorno con modelos capaces de adaptarse a hablantes previamente no vistos con pocos segundos de audio.

A continuación se presentan una serie de modificaciones al modelo inicial descrito en el capítulo 3, de forma que se expande para poder hacer una adaptación *zero-shot* a locutores no vistos a partir de un audio de referencia que contiene unos pocos segundos de habla. Esta estructura ha sido propuesta por el MLLP en [1] y [52]. El modelo resultante después de las modificaciones está ilustrado en la figura 4.1.

4.2 Mejoras al modelo inicial

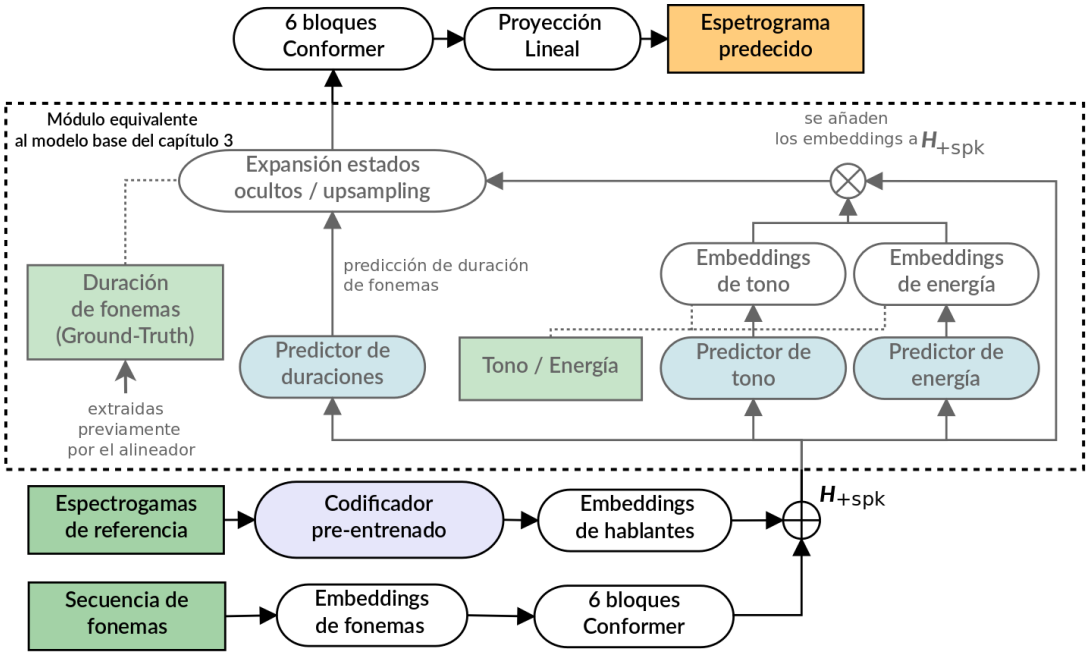

Figura 4.1: Esquema del modelo acústico resultante tras las mejoras para la adaptación *zero-shot*. Adaptado de [1] y [52].

4.2.1. Conformer

Aunque el modelo acústico del apartado 3.2.2 da buenos resultados, existen actualmente otras alternativas que consiguen mejores resultados en la construcción de modelos de TTS *end-to-end*. El Conformer [70] es uno de estos modelos. Este tipo de modelo actúa como una red Transformer, integrando parte de su arquitectura junto a componentes de redes convolucionales. La idea principal es que, juntando estos dos tipos de modelos, se logren captar a la vez los distintos niveles de correlación correspondientes a la entrada: global mediante *self-attention* y local mediante las capas convolucionales.

Como una forma de mejorar nuestro modelo ante la dificultad añadida por la adaptación *zero-shot*, se ha decidido en el resto del trabajo utilizar el Conformer en el modelo

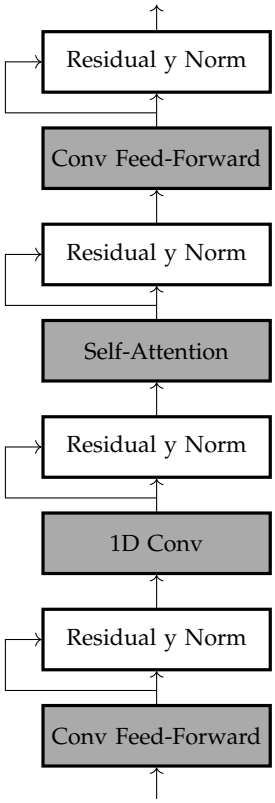

Figura 4.2: Esquema del módulo Conformer modificado.

acústico, sustituyendo a la vez el codificador y decodificador por 6 bloques Conformer en cada uno, modificados siguiendo las recomendaciones indicadas en [61].

La estructura de este bloque Conformer modificado se puede ver en la figura 4.2. Consiste en 4 módulos: un módulo con una red prealimentada convolucional (*convolutional feed-forward*), un módulo convolucional unidimensional, un módulo de *self-attention* y otro módulo que repite la red convolucional. De forma similar a un Transformer, en cada módulo se normaliza su salida y se hace uso de una conexión residual [71].

4.2.2. Clonación de identidad de hablantes con modelos prentrenados

Trabajos como [68] o [69] plantean el uso de modelos prentrenados que actúan como codificadores que generan *embeddings* sobre la identidad de los hablantes y que se pasan posteriormente al modelo principal. Los modelos son entrenados sobre corpus de datos grandes y diversos en cantidad de hablantes, de forma de que se logre llegar a una generalización del dominio principal para que luego ayude a estos a poder adaptarse a nuevos dominios que sean similares.

Por ello, se ha decidido hacer uso de un modelo público prentrenado siguiendo este tipo de estructura para añadirlo al modelo modificado de este capítulo. El modelo de codificador que se ha utilizado se basa en la arquitectura presentada en [72]. En específico, se hace uso de una versión [1] de esta [73] pensada para conseguir el máximo rendimiento. Los autores se refieren a esta implementación como H/ASP, y consiste en una variación de una red neuronal residual (ResNet) [71] de 34 capas, entrenada sobre el corpus de datos VoxCeleb [74] [75].

[1] Implementación de código abierto de los autores: https://github.com/clovaai/voxceleb_trainer

4.3 Entrenamiento y verificación del modelo mejorado

A continuación se presenta el entrenamiento sobre el modelo modificado de este capítulo, por un lado, con el fin de construir un modelo de TTS en castellano con la capacidad de realizar adaptación *zero-shot* a locutores no vistos, y por otro, para su posterior comparación con un modelo entrenado sobre un mismo corpus de datos, pero bajo una configuración de *Transfer Learning*, concepto que se trata en el próximo capítulo.

En este caso, se ha hecho uso del corpus DeX-TTS [76], que viene derivado del plan *Docència en Xarxa* de la UPV. Este tiene el objetivo de incentivar la elaboración de materiales educativos reutilizables en formato digital y mejorar el rendimiento académico de los estudiantes del centro. El corpus fue creado por el grupo MLLP junto al ASIC (UPV) durante los cursos académicos 2016-17 y 2017-18. En total, el corpus comprende 59 horas de habla multilingüe, con 47.000 frases pronunciadas por 98 participantes del ámbito académico. Los audios se clasifican según la lengua contenida en estos, donde se distingue español (61 %), catalán (15 %) e inglés (24 %). Cabe indicar que se tiene en cuenta el número de lenguas habladas por los participantes, distinguiéndose entre casos monolingües (35 %), bilingües (42 %) o trilingües (23 %).

El entrenamiento sigue el mismo esquema presentado en el capítulo 3, diferenciándose en el uso de una GPU Nvidia RTX 3090 para el entrenamiento del modelo acústico y *vocoder* por restricciones de memoria GPU, ya que el modelo Conformer no cabe en una GPU de 11GB como es la GPU Nvidia GTX 2080Ti. Esto resulta en un total de $\sim$142 horas de entrenamiento. El número de iteraciones y tamaño de *batch* escogidos para cada componente del modelo modificado se muestra en la tabla 4.1. La evaluación completa de la calidad del modelo resultante se trata en el próximo capítulo.

Tabla 4.1: Número de iteraciones y tamaño de *batch* para el entrenamiento de cada componente del modelo modificado.

	Núm. de iteraciones	Tamaño de *batch*
Alineador	100k	64
Modelo acústico	500k	8
Vocoder	800k	0k-100k: 48 100k-800k: 24

CAPÍTULO 5
Transfer Learning y evaluación de modelos

En este capítulo se presenta el concepto de *Transfer Learning* y se plantea un mecanismo que facilita el entrenamiento de los sistemas de TTS en múltiples idiomas. Se presenta una evaluación detallada de un modelo entrenado bajo esta nueva configuración en comparación con el modelo entrenado en el capítulo 4 para determinar la calidad resultante del habla generada. Además, a partir de los resultados, se entrena este nuevo modelo basado en *Transfer Learning* sobre un corpus en alemán para demostrar la adaptabilidad de esta arquitectura a diferentes entornos.

5.1 Transfer Learning

Como se ha comentado brevemente en el apartado 4.1, en muchos casos no es factible la recopilación de datos de entrenamiento suficientes sobre un dominio específico, ya sea por su coste computacional, temporal o al no tener acceso a estos datos.

Debido a esto, durante los últimos años se ha popularizado el uso del aprendizaje por transferencia o *Transfer Learning*[1] como solución a este problema. Este tiene como objetivo aprovechar el conocimiento de un modelo sobre un dominio para transferir sus conocimientos sobre este a otro dominio relacionado [77]. El método de clonación de voz del apartado 4.2.2 puede verse como un caso particular de *Transfer Learning*.

En la sección 4.3 se ha planteado el entrenamiento desde cero de un modelo de TTS para cada vez que se utilizase un corpus con un nuevo idioma, pero este método sería costoso a medida que se aumentase el número de idiomas. En este caso, para conseguir un sistema robusto y adaptable de TTS, el *Transfer Learning* ofrece una solución eficaz.

Por esta razón, a continuación se propone el entrenamiento de un modelo generalista de TTS capaz de ser afinado a distintos idiomas mediante el entrenamiento adicional sobre el corpus correspondiente y su posterior evaluación respecto el modelo entrenado en el capítulo 4.

[1] Ocasionalmente en la literatura el término *fine-tuning* es usado para referirse a *Transfer Learning*. Esto puede llevar a confusión, ya que el término *fine-tuning* también es utilizado para referirse a la exploración de los hiperparámetros de un modelo para mejorar su rendimiento. En este trabajo el término *fine-tuning* se ha utilizado para referirse al proceso de entrenamiento adicional que se ha utilizado en este capítulo.

5.2 Entrenamiento mediante Transfer Learning

Para el entrenamiento del modelo de TTS generalista, se hace uso del mismo modelo explicado en el capítulo 4, empleándose en este caso un corpus con un tamaño comparativamente mayor, LibriTTS [78].

Este corpus de varios hablantes incluye aproximadamente 585 horas de habla en inglés. Los audios están a una frecuencia de muestreo de 24kHz y en total hay 2.456 hablantes distintos. Los contenidos derivan de los materiales originales del corpus de ASR LibriSpeech [79], adaptados para la investigación de TTS. De forma similar a LJ Speech, actualmente LibriTTS es una de las opciones más populares en la evaluación de modelos *multi-speaker* de TTS, y es por esta razón por la que se ha seleccionado.

El entrenamiento hace uso del mismo *hardware* que en el modelo del capítulo 4, dividiéndose el entrenamiento entre una GPU Nvidia GTX 2080Ti para el preprocesamiento y alineador externo y una GPU Nvidia RTX 3090 para el modelo acústico y *vocoder*. El número de iteraciones y tamaño de *batch* escogidos para cada componente de este modelo generalista se muestra en la tabla 5.1.

En total, el entrenamiento de este modelo generalista resulta en un tiempo total de ∼251 horas. Partiendo de este, se realiza *fine-tuning* del modelo, entrenándose adicionalmente sobre el ya mencionado corpus DeX-TTS con un tiempo adicional de entrenamiento de ∼13 horas. A partir de este y con el interés de evaluar la importancia del número de iteraciones en este proceso de *fine-tuning*, se guardan dos modelos, siendo uno de ellos con un nivel reducido de iteraciones, **LibriTTS + FT DeX-TTS 10k**, y otro al final del entrenamiento, **LibriTTS + FT DeX-TTS 50k**.

Tabla 5.1: Número de iteraciones y tamaño de *batch* para el entrenamiento de cada componente del modelo generalista con *Transfer Learning* en LibriTTS. Las iteraciones donde se realiza *fine-tuning* sobre DeX-TTS están indicadas mediante *FT*.

	Núm. de iteraciones	Tamaño de *batch*
Alineador	250k	64
Modelo acústico	500k + 50k *FT*	8
Vocoder	800k + 50k *FT*	0k-100k: 48 100k-850k: 24

5.3 Evaluación de sistemas de TTS en español

Para evaluar la efectividad del *Transfer Learning*, se va a comparar el modelo entrenado en DeX-TTS del capítulo 4, que no hace uso de esta técnica, respecto los dos modelos de *Transfer Learning* entrenados sobre el mismo corpus del apartado anterior, **LibriTTS + FT DeX-TTS 10k** y **LibriTTS + FT DeX-TTS 50k**. Para ello se ha realizado un estudio en profundidad que mide la calidad del TTS obtenido por estos modelos, dividiéndose en dos aspectos principales a estudiar:

- **Naturalidad**: Incluye la calidad de sonido y prosodia del habla generada.
- **Clonación de hablantes**: Mide la adaptación *zero-shot* de los modelos para replicar la identidad y características de hablantes que previamente no se han visto durante el entrenamiento.

Recordando el apartado 2.2.3 y las cualidades que caracterizan la evaluación de sistemas de TTS, estas dos características se han decidido evaluar con una evaluación subjetiva de la ya mencionada métrica del MOS [37] [38].

Para la evaluación, se ha hecho uso de audios en castellano del corpus de datos Multilingual TEDx [80]. Se han seleccionado de forma arbitraria 10 fragmentos de audio de no más de 30 segundos, correspondiendo cada uno de estos con un único locutor (5 hombres y 5 mujeres). A partir de estos audios, se ha definido un conjunto de test de 20 frases, seleccionando aleatoriamente 3 de estas por locutor y generándose los audios correspondientes por cada modelo a evaluar, resultando en un total de 90 muestras de audio [2] (30 por cada modelo).

A partir de estas muestras, primero se ha definido el conjunto de test para evaluar la naturalidad de los modelos, cogiendo las 90 muestras y añadiendo 10 audios extraídos de DeX-TTS como muestras de control, resultando en 100 muestras a ser evaluadas por cada participante. Respecto a la clonación de voz de locutores no vistos de los modelos, se han escogido 30 muestras del total, correspondiendo cada una con una combinación única entre locutor y modelo, que en la evaluación forman pares junto al audio de referencia que ha servido para generar el audio sintetizado y del que se está intentando copiar sus características.

Un total de 13 expertos en el procesamiento de lenguajes naturales y hablantes nativos de castellano han participado en la evaluación subjetiva de los modelos. Los resultados de cada aspecto de los modelos se ilustran en las tablas 5.2 y 5.3 respectivamente.

Tabla 5.2: MOS de la naturalidad de los modelos con un intervalo de confianza del 95 %.

	MOS naturalidad	Muestras evaluadas
DeX-TTS (Cap.4)	4.15 ± 0.09	390
LibriTTS + FT DeX-TTS 10k	3.66 ± 0.10	390
LibriTTS + FT DeX-TTS 50k	4.07 ± 0.08	390
Muestras de control	4.94 ± 0.05	130

Tabla 5.3: MOS de la clonación de voz de locutores no vistos de los modelos con un intervalo de confianza del 95 %.

	MOS clonación	Muestras evaluadas
DeX-TTS (Cap.4)	3.15 ± 0.17	130
LibriTTS + FT DeX-TTS 10k	3.05 ± 0.17	130
LibriTTS + FT DeX-TTS 50k	3.24 ± 0.16	130

Observándose la tabla 5.2, podemos ver cómo los modelos han recibido una evaluación positiva respecto a su naturalidad, siendo el modelo sin *fine-tuning* y el modelo *fine-tuned* con un mayor número de iteraciones los que han obtenido mejores resultados. Aunque en la tabla 5.3 podemos ver que los modelos en general pueden clonar la identidad de hablantes, se puede observar como los 3 modelos descritos han obtenido unos resultados muy similares, existiendo todavía un margen considerable de mejora.

A partir de estos resultados, podemos analizar y extraer una serie de ideas clave de estos modelos. Respecto al *fine-tuning* se ha confirmado que, en línea con los trabajos en esta área, este sirve como una alternativa viable para entrenar modelos desde cero que obtengan una calidad comparable a aquellos que no hacen uso de esta técnica. Por

[2] Se ha asegurado que todas las frases del conjunto de test aparecían al menos una vez.

otra parte, se ha observado la importancia en esta técnica del número de iteraciones de entrenamiento, y en este caso, cómo puede influir de forma significativa en los resultados obtenidos para cada dominio que se quiere evaluar.

También, a partir de los resultados, se ha observado que no se ha conseguido mejorar de forma significativa la capacidad de adaptación *zero-shot* a locutores no vistos a través del mecanismo de *Transfer Learning*.

Asimismo se ha visto, que aunque para el uso del *Transfer Learning* es necesario el entrenamiento de un modelo generalista, el uso de *fine-tuning* reduce el tiempo total de entrenamiento si se quiere construir un sistema de TTS formado por múltiples modelos monolingües.

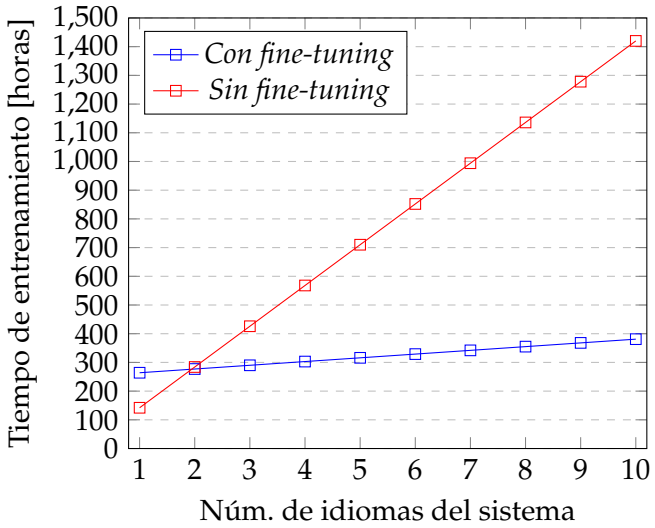

Figura 5.1: Tiempos de entrenamiento de un hipotético sistema de TTS formado por múltiples modelos monolingües a partir de los tiempos resultantes de los modelos entrenados en DeX-TTS.

Este hecho se puede visualizar fácilmente para nuestro caso en la figura 5.1. El tiempo total de entrenamiento de nuestros modelos DeX-TTS con *fine-tuning* es de ∼264 horas, mientras que el del modelo Dex-TTS base del capítulo 4 es de ∼142 horas. Si se desea hacer uso de un modelo adicional en otro idioma, en el caso de entrenar sobre un corpus similar al tamaño de DeX-TTS, nuestro enfoque de *Transfer Learning* añadiría ∼13 horas adicionales de *fine-tuning* al total, resultando en ∼277 horas. El enfoque desde cero, en este caso, tardaría más, sobre unas ∼284 horas. Esta diferencia no hace más que aumentar a medida que se necesita un mayor número de modelos en diferentes idiomas. Por ejemplo, si queremos un sistema de TTS capaz de generar audio en 10 idiomas diferentes, el enfoque de *Transfer Learning* daría un tiempo total de ∼381 horas, mientras que el enfoque desde cero alcanzaría las ∼1420 horas.

5.4 Evaluación de un sistema de TTS en alemán

Para completar la evaluación del *Transfer Learning* y cómo este puede utilizarse para crear con facilidad sistemas de TTS en múltiples idiomas, se ha realizado *fine-tuning* sobre el modelo planteado en este capítulo siguiendo el mismo esquema de los dos últimos apartados y entrenándose esta vez sobre el corpus de datos de habla de alemán HUI-Audio-Corpus-German [81]. Este forma parte del dominio público y consta de más de 326 horas de fragmentos de audio. Está pensado al mismo tiempo para su utilización en modelos de un solo hablante o varios, tomando cinco hablantes con 32 a 96 horas de

audio cada uno para construir modelos *single-speaker*, así como 97 horas de audio de 117 hablantes adicionales para modelos *multi-speaker* como es nuestro caso.

Para este modelo se ha elegido el mismo número de iteraciones y tamaño de *batch* que se indican en la tabla 5.1, exceptuando para el alineador, en cuyo caso se han elegido 100.000 iteraciones. El proceso de *fine-tuning* en este modelo ha durado un total de $\sim$24 horas.

Debido a la dificultad de encontrar expertos con conocimientos nativos de alemán, se ha optado por realizar una evaluación informal con 2 hablantes no nativos del lenguaje. Teniendo en cuenta los resultados obtenidos en el anterior apartado sobre el modelo **LibriTTS + FT DeX-TTS 50k** se les ha pedido que evalúen la naturalidad del modelo entrenado **LibriTTS + FT HUI-German 50k** con 20 audios generados, correspondiendo cada mitad a un locutor y a una locutora.

Según los participantes, la calidad obtenida es buena, y en algunos casos, se ha llegado a confundir si la voz que se estaba escuchando era sintética o correspondía con la de un ser humano. Aun así, se han detectado en algunos casos ligeros errores en la prosodia respecto a las pausas naturales entre palabras.

CAPÍTULO 6
Conclusiones

En este capítulo se recogen y resumen todas las tareas realizadas por el autor en este trabajo en colaboración con el MLLP tal y como se ha mencionado en la sección 1.2, así como las conclusiones y reflexiones sobre los objetivos propuestos en la sección 1.3. Además, se comentan ideas que podrían desarrollarse en el futuro para expandir las propuestas en este trabajo.

El primero de los objetivos planteados se ha tratado a lo largo de todo el trabajo. En el capítulo 2 se ha presentado una visión general del aprendizaje automático y se han explicado las aplicaciones de esta rama en la síntesis del habla, tratándose distintas arquitecturas de redes neuronales dentro del campo del procesamiento del lenguaje natural. Asimismo, en el capítulo 3 se ha profundizado sobre las últimas tendencias en la síntesis del habla mediante la investigación de modelos neuronales no autorregresivos y en el capítulo 4 a través de las técnicas necesarias para adaptar el TTS a múltiples hablantes y casos no vistos.

Respecto al segundo objetivo, este ha sido abordado a través de la iteración progresiva de un modelo de TTS que incluyese los últimos avances de la síntesis de habla y lograse resultados competitivos. En el capítulo 3 el autor se ha familiarizado con las herramientas facilitadas para el entrenamiento de modelos de TTS modernos, proponiéndose un modelo base que se ha utilizado para generar un sistema en inglés mediante LJ Speech. Este se ha ampliado en el capítulo 4 introduciendo la capacidad de clonar las características de hablantes no vistos. Esto ha servido para entrenar un modelo en castellano sobre Dex-TTS que, a su vez, se ha visto expandido en el capítulo 5 mediante el uso de modelos prentrenados con LibriSpeech y *Transfer Learning* para facilitar la creación de un sistema de TTS en múltiples idiomas. Adicionalmente en este último capítulo se ha hecho uso de HUI-Audio-Corpus-German para crear un sistema de TTS en alemán y demostrar la efectividad del modelo final de *Transfer Learning* para adaptarse a múltiples idiomas.

Cumpliendo el tercer objetivo, se ha llevado a cabo una evaluación subjetiva de los modelos propuestos, destacando las que se han descrito en el capítulo 5 para medir la efectividad del *Transfer Learning*. Se ha demostrado cómo esta técnica es una alternativa eficaz en la creación de sistemas de TTS multilingües. Se ha conseguido un sistema de síntesis del habla de alta calidad similar a la obtenida en modelos entrenados desde cero, pero de manera más rápida y eficiente.

Este trabajo ha servido como una primera aproximación del autor al TTS, pero se pretende expandir en investigaciones futuras en colaboración con el MLLP. Se han identificado múltiples vías de mejora para ampliar las capacidades de los modelos propuestos. Si bien la naturalidad y eficacia de los modelos neuronales de TTS aumentará a medida que aparezcan modelos y algoritmos más potentes en el futuro, existe un claro interés en mejorar la adaptación *zero-shot* de hablantes. En este caso, se prestará atención a la explo-

ración de nuevas arquitecturas que mejoren la clonación de la identidad de hablantes y que además, permitan que este proceso pueda extraer características de un hablante cuyo lenguaje nativo no corresponda con el de salida del modelo.

Bibliografía

[1] Alejandro Manuel Pérez-González de Martos. *Deep Neural Networks for Automatic Speech-To-Speech Translation of Open Educational Resources*. PhD thesis, Universidad Politécnica de València, 2022.

[2] Kevin P. Murphy. *Probabilistic Machine Learning: An introduction*. MIT Press, 2022.

[3] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65 6:386–408, 1958.

[4] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.

[5] Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, pages 807–814. Omnipress, 2010.

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

[7] Yoshua Bengio, Patrice Y. Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Trans. Neural Networks*, 5(2):157–166, 1994.

[8] Razvan Pascanu, Tomás Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *ICML (3)*, volume 28 of *JMLR Workshop and Conference Proceedings*, pages 1310–1318. JMLR.org, 2013.

[9] Robin M. Schmidt. Recurrent neural networks (rnns): A gentle introduction and overview. *CoRR*, abs/1912.05911, 2019.

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

[11] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *EMNLP*, pages 1724–1734. ACL, 2014.

[12] Kunihiko Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36:193–202, 1980.

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

[14] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, pages 1106–1114, 2012.

[15] Rajat Raina, Anand Madhavan, and Andrew Y. Ng. Large-scale deep unsupervised learning using graphics processors. In *ICML*, volume 382 of *ACM International Conference Proceeding Series*, pages 873–880. ACM, 2009.

[16] Dan C. Ciresan, Ueli Meier, and Jürgen Schmidhuber. Multi-column deep neural networks for image classification. In *CVPR*, pages 3642–3649. IEEE Computer Society, 2012.

[17] Keiron O'Shea and Ryan Nash. An introduction to convolutional neural networks. *CoRR*, abs/1511.08458, 2015.

[18] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. Generative adversarial nets. In *NIPS*, pages 2672–2680, 2014.

[19] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. In *NIPS*, pages 3104–3112, 2014.

[20] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *ICLR*, 2015.

[21] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, pages 5998–6008, 2017.

[22] Amirhossein Kazemnejad. Transformer architecture: The positional encoding. *kazemnejad.com*, 2019.

[23] Daniel W. Otter, Julian R. Medina, and Jugal K. Kalita. A survey of the usages of deep learning for natural language processing. *IEEE Trans. Neural Networks Learn. Syst.*, 32(2):604–624, 2021.

[24] Xu Tan, Tao Qin, Frank K. Soong, and Tie-Yan Liu. A survey on neural speech synthesis. *CoRR*, abs/2106.15561, 2021.

[25] Diemo Schwarz. Concatenative sound synthesis: The early years. *Journal of New Music Research*, 35(1):3–22, 2006.

[26] Heiga Zen, Andrew W. Senior, and Mike Schuster. Statistical parametric speech synthesis using deep neural networks. In *ICASSP*, pages 7962–7966. IEEE, 2013.

[27] Yuchen Fan, Yao Qian, Feng-Long Xie, and Frank K. Soong. TTS synthesis with bidirectional LSTM based recurrent neural networks. In *INTERSPEECH*, pages 1964–1968. ISCA, 2014.

[28] Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew W. Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. In *SSW*, page 125. ISCA, 2016.

[29] Yuxuan Wang, R. J. Skerry-Ryan, Daisy Stanton, Yonghui Wu, Ron J. Weiss, Navdeep Jaitly, Zongheng Yang, Ying Xiao, Zhifeng Chen, Samy Bengio, Quoc V. Le, Yannis Agiomyrgiannakis, Rob Clark, and Rif A. Saurous. Tacotron: Towards end-to-end speech synthesis. In *INTERSPEECH*, pages 4006–4010. ISCA, 2017.

[30] Jonathan Shen, Ruoming Pang, Ron J. Weiss, Mike Schuster, Navdeep Jaitly, Zongheng Yang, Zhifeng Chen, Yu Zhang, Yuxuan Wang, RJ-Skerrv Ryan, Rif A. Saurous, Yannis Agiomyrgiannakis, and Yonghui Wu. Natural TTS synthesis by conditioning wavenet on MEL spectrogram predictions. In *ICASSP*, pages 4779–4783. IEEE, 2018.

[31] Wei Ping, Kainan Peng, Andrew Gibiansky, Sercan Ömer Arik, Ajay Kannan, Sharan Narang, Jonathan Raiman, and John Miller. Deep voice 3: Scaling text-to-speech with convolutional sequence learning. In *ICLR (Poster)*. OpenReview.net, 2018.

[32] Yi Ren, Yangjun Ruan, Xu Tan, Tao Qin, Sheng Zhao, Zhou Zhao, and Tie-Yan Liu. Fastspeech: Fast, robust and controllable text to speech. In *NeurIPS*, pages 3165–3174, 2019.

[33] Yi Ren, Chenxu Hu, Xu Tan, Tao Qin, Sheng Zhao, Zhou Zhao, and Tie-Yan Liu. Fastspeech 2: Fast and high-quality end-to-end text to speech. In *ICLR*. OpenReview.net, 2021.

[34] Melvyn J. Hunt. Figures of merit for assessing connected-word recognisers. *Speech Commun.*, 9(4):329–336, 1990.

[35] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *ACL*, pages 311–318. ACL, 2002.

[36] Petra Wagner, Jonas Beskow, Simon Betz, Jens Edlund, Joakim Gustafson, Gustav Eje Henter, Sébastien Le Maguer, Zofia Malisz, Éva Székely, Christina Tånnander, and Jana Voße. Speech Synthesis Evaluation — State-of-the-Art Assessment and Suggestion for a Novel Research Program. In *Proc. 10th ISCA Workshop on Speech Synthesis (SSW 10)*, pages 105–110, 2019.

[37] ITU-T. *ITU-T P.800 : Methods for subjective determination of transmission quality*, 1996. https://www.itu.int/rec/T-REC-P.800-199608-I.

[38] ITU-T. *ITU-T P.800.1 - Mean Opinion Score (MOS) terminology*, 2016. https://www.itu.int/rec/T-REC-P.800.1-201607-I/en.

[39] Jing-Xuan Zhang, Zhen-Hua Ling, and Li-Rong Dai. Forward attention in sequence-to-sequence acoustic modeling for speech synthesis. In *ICASSP*, pages 4789–4793. IEEE, 2018.

[40] Mutian He, Yan Deng, and Lei He. Robust sequence-to-sequence acoustic modeling with stepwise monotonic attention for neural TTS. In *INTERSPEECH*, pages 1293–1297. ISCA, 2019.

[41] Ronald J. Williams and David Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural Comput.*, 1(2):270–280, 1989.

[42] Rui Liu, Berrak Sisman, Jingdong Li, Feilong Bao, Guanglai Gao, and Haizhou Li. Teacher-student training for robust tacotron-based TTS. In *ICASSP*, pages 6274–6278. IEEE, 2020.

[43] Florian Schmidt. Generalization in generation: A closer look at exposure bias. In *NGT@EMNLP-IJCNLP*, pages 157–167. Association for Computational Linguistics, 2019.

[44] Wei-Ning Hsu, Yu Zhang, Ron J. Weiss, Heiga Zen, Yonghui Wu, Yuxuan Wang, Yuan Cao, Ye Jia, Zhifeng Chen, Jonathan Shen, Patrick Nguyen, and Ruoming Pang. Hierarchical generative modeling for controllable speech synthesis. In *ICLR (Poster)*. OpenReview.net, 2019.

[45] Chengzhu Yu, Heng Lu, Na Hu, Meng Yu, Chao Weng, Kun Xu, Peng Liu, Deyi Tuo, Shiyin Kang, Guangzhi Lei, Dan Su, and Dong Yu. Durian: Duration informed attention network for speech synthesis. In *INTERSPEECH*, pages 2027–2031. ISCA, 2020.

[46] Michael McAuliffe, Michaela Socolof, Sarah Mihuc, Michael Wagner, and Morgan Sonderegger. Montreal forced aligner: Trainable text-speech alignment using kaldi. In *INTERSPEECH*, pages 498–502. ISCA, 2017.

[47] Adrian Lancucki. Fastpitch: Parallel text-to-speech with pitch prediction. In *ICASSP*, pages 6588–6592. IEEE, 2021.

[48] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[49] Shervin Minaee, Yuri Boykov, Fatih Porikli, Antonio Plaza, Nasser Kehtarnavaz, and Demetri Terzopoulos. Image segmentation using deep learning: A survey, 2020.

[50] Alex Graves, Santiago Fernández, Faustino J. Gomez, and Jürgen Schmidhuber. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *ICML*, volume 148 of *ACM International Conference Proceeding Series*, pages 369–376. ACM, 2006.

[51] M. Schuster and K.K. Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997.

[52] Javier Iranzo-Sánchez, Javier Jorge, Alejandro Pérez-González de Martos, Adrià Giménez, Gonçal V. Garcés Díaz-Munío, Pau Baquero-Arnal, Joan Albert Silvestre-Cerdà, Jorge Civera, Albert Sanchis, and Alfons Juan. MLLP-VRAIN UPV systems for the IWSLT 2022 Simultaneous Speech Translation and Speech-to-Speech Translation tasks. In *Proc. of 19th Intl. Workshop on Spoken Language Translation (IWSLT 2022)*, pages 255–264, Dublin (Ireland), 2022.

[53] Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Trans. Image Process.*, 13(4):600–612, 2004.

[54] Ryan Prenger, Rafael Valle, and Bryan Catanzaro. Waveglow: A flow-based generative network for speech synthesis. In *ICASSP*, pages 3617–3621. IEEE, 2019.

[55] Zhifeng Kong, Wei Ping, Jiaji Huang, Kexin Zhao, and Bryan Catanzaro. Diffwave: A versatile diffusion model for audio synthesis. In *ICLR*. OpenReview.net, 2021.

[56] Kundan Kumar, Rithesh Kumar, Thibault de Boissiere, Lucas Gestin, Wei Zhen Teoh, Jose Sotelo, Alexandre de Brébisson, Yoshua Bengio, and Aaron C. Courville. Melgan: Generative adversarial networks for conditional waveform synthesis. In *NeurIPS*, pages 14881–14892, 2019.

[57] Ryuichi Yamamoto, Eunwoo Song, and Jae-Min Kim. Parallel wavegan: A fast waveform generation model based on generative adversarial networks with multi-resolution spectrogram. In *ICASSP*, pages 6199–6203. IEEE, 2020.

[58] Jinhyeok Yang, Junmo Lee, Young-Ik Kim, Hoon-Young Cho, and Injung Kim. Vocgan: A high-fidelity real-time vocoder with a hierarchically-nested adversarial network. In *INTERSPEECH*, pages 200–204. ISCA, 2020.

[59] Jungil Kong, Jaehyeon Kim, and Jaekyoung Bae. Hifi-gan: Generative adversarial networks for efficient and high fidelity speech synthesis. In *NeurIPS*, 2020.

[60] Won Jang, Dan Lim, Jaesam Yoon, Bongwan Kim, and Juntae Kim. Univnet: A neural vocoder with multi-resolution spectrogram discriminators for high-fidelity waveform generation. In *Interspeech*, pages 2207–2211. ISCA, 2021.

[61] Yanqing Liu, Zhihang Xu, Gang Wang, Kuan Chen, Bohan Li, Xu Tan, Jinzhu Li, Lei He, and Sheng Zhao. DelightfulTTS: The Microsoft Speech Synthesis System for Blizzard Challenge 2021. *CoRR*, abs/2110.12612, 2021.

[62] Masanori Morise, Fumiya Yokomori, and Kenji Ozawa. WORLD: A vocoder-based high-quality speech synthesis system for real-time applications. *IEICE Trans. Inf. Syst.*, 99-D(7):1877–1884, 2016.

[63] Masanori Morise, Hideki Kawahara, and Haruhiro Katayose. Fast and reliable f0 estimation method based on the period extraction of vocal fold vibration of singing voice and speech. In *Audio Engineering Society Conference: 35th International Conference: Audio for Games*, Feb 2009.

[64] Keith Ito and Linda Johnson. The lj speech dataset. https://keithito.com/LJ-Speech-Dataset/, 2017.

[65] Yutian Chen, Yannis M. Assael, Brendan Shillingford, David Budden, Scott E. Reed, Heiga Zen, Quan Wang, Luis C. Cobo, Andrew Trask, Ben Laurie, Çaglar Gülçehre, Aäron van den Oord, Oriol Vinyals, and Nando de Freitas. Sample efficient adaptive text-to-speech. In *ICLR (Poster)*. OpenReview.net, 2019.

[66] Rama Doddipatla, Norbert Braunschweiler, and Ranniery Maia. Speaker adaptation in dnn-based speech synthesis using d-vectors. In *INTERSPEECH*, pages 3404–3408. ISCA, 2017.

[67] Sercan Ömer Arik, Jitong Chen, Kainan Peng, Wei Ping, and Yanqi Zhou. Neural voice cloning with a few samples. In *NeurIPS*, pages 10040–10050, 2018.

[68] Ye Jia, Yu Zhang, Ron J. Weiss, Quan Wang, Jonathan Shen, Fei Ren, Zhifeng Chen, Patrick Nguyen, Ruoming Pang, Ignacio Lopez-Moreno, and Yonghui Wu. Transfer learning from speaker verification to multispeaker text-to-speech synthesis. In *NeurIPS*, pages 4485–4495, 2018.

[69] Erica Cooper, Cheng-I Lai, Yusuke Yasuda, Fuming Fang, Xin Wang, Nanxin Chen, and Junichi Yamagishi. Zero-shot multi-speaker text-to-speech with state-of-the-art neural speaker embeddings. In *ICASSP*, pages 6184–6188. IEEE, 2020.

[70] Anmol Gulati, James Qin, Chung-Cheng Chiu, Niki Parmar, Yu Zhang, Jiahui Yu, Wei Han, Shibo Wang, Zhengdong Zhang, Yonghui Wu, and Ruoming Pang. Conformer: Convolution-augmented Transformer for Speech Recognition. In *Proc. Interspeech 2020*, pages 5036–5040, 2020.

[71] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778. IEEE Computer Society, 2016.

[72] Joon Son Chung, Jaesung Huh, Seongkyu Mun, Minjae Lee, Hee-Soo Heo, Soyeon Choe, Chiheon Ham, Sunghwan Jung, Bong-Jin Lee, and Icksang Han. In defence of metric learning for speaker recognition. In *INTERSPEECH*, pages 2977–2981. ISCA, 2020.

[73] Hee Soo Heo, Bong-Jin Lee, Jaesung Huh, and Joon Son Chung. Clova baseline system for the VoxCeleb speaker recognition challenge 2020. *arXiv preprint arXiv:2009.14153*, 2020.

[74] A. Nagrani, J. S. Chung, and A. Zisserman. Voxceleb: a large-scale speaker identification dataset. In *INTERSPEECH*, 2017.

[75] J. S. Chung, A. Nagrani, and A. Zisserman. Voxceleb2: Deep speaker recognition. In *INTERSPEECH*, 2018.

[76] Alejandro Pérez, Gonçal V. Garcés Díaz-Munío, Adrià Giménez, Joan Albert Silvestre-Cerdà, Alberto Sanchís, Jorge Civera, Manuel Jiménez, Carlos Turró, and Alfons Juan. Towards cross-lingual voice cloning in higher education. *Eng. Appl. Artif. Intell.*, 105:104413, 2021.

[77] Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. A comprehensive survey on transfer learning. *Proc. IEEE*, 109(1):43–76, 2021.

[78] Heiga Zen, Viet Dang, Rob Clark, Yu Zhang, Ron J. Weiss, Ye Jia, Zhifeng Chen, and Yonghui Wu. Libritts: A corpus derived from librispeech for text-to-speech. In *INTERSPEECH*, pages 1526–1530. ISCA, 2019.

[79] Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. Librispeech: An ASR corpus based on public domain audio books. In *ICASSP*, pages 5206–5210. IEEE, 2015.

[80] Elizabeth Salesky, Matthew Wiesner, Jacob Bremerman, Roldano Cattoni, Matteo Negri, Marco Turchi, Douglas W. Oard, and Matt Post. Multilingual TEDx corpus for speech recognition and translation. In *Proceedings of Interspeech*, 2021.

[81] Pascal Puchtler, Johannes Wirth, and René Peinl. Hui-audio-corpus-german: A high quality TTS dataset. In *KI*, volume 12873 of *Lecture Notes in Computer Science*, pages 204–216. Springer, 2021.

ANEXO

OBJETIVOS DE DESARROLLO SOSTENIBLE

Grado de relación del trabajo con los Objetivos de Desarrollo Sostenible (ODS).

Objetivos de Desarrollo Sostenible	**Alto**	**Medio**	**Bajo**	**No procede**
ODS 1. **Fin de la pobreza.**				**X**
ODS 2. **Hambre cero.**				**X**
ODS 3. **Salud y bienestar.**			**X**	
ODS 4. **Educación de calidad.**	**X**			
ODS 5. **Igualdad de género.**				**X**
ODS 6. **Agua limpia y saneamiento.**				**X**
ODS 7. **Energía asequible y no contaminante.**				**X**
ODS 8. **Trabajo decente y crecimiento económico.**			**X**	
ODS 9. **Industria, innovación e infraestructuras.**		**X**		
ODS 10. **Reducción de las desigualdades.**		**X**		
ODS 11. **Ciudades y comunidades sostenibles.**				**X**
ODS 12. **Producción y consumo responsables.**				**X**
ODS 13. **Acción por el clima.**				**X**
ODS 14. **Vida submarina.**				**X**
ODS 15. **Vida de ecosistemas terrestres.**				**X**
ODS 16. **Paz, justicia e instituciones sólidas.**				**X**
ODS 17. **Alianzas para lograr objetivos.**			**X**	

Reflexión sobre la relación del TFG con los ODS más relacionados.

Es muy importante analizar la contribución que se hace cuando se realizan trabajos científicos y las mejoras que se pretenden alcanzar con estos.

A partir de los Objetivos de Desarrollo Sostenibles elaborados por la ONU en el marco de la Agenda 2030 (aprobados en 2015), se han relacionado los siguientes como los más relevantes respecto a los objetivos del proyecto:

- **Educación de calidad (ODS 4):** La síntesis de voz es una herramienta poderosa para la mejora de la calidad educativa. El MLLP ha aplicado esta tecnología a escenarios reales para el ámbito educativo como en el proyecto **transLectures** para el subtitulado de videos educativos. Este ODS está estrechamente relacionado con este TFG, dado que el trabajo de investigación desarrollado por el MLLP ha sido implementado en este proyecto. En el futuro se tiene pensado continuar integrando esta tecnología, haciéndola disponible al público educativo general y mejorando sus capacidades.

- **Industria, innovación e infraestructuras (ODS 9):** En el trabajo se trata el aprendizaje automático, que con los recientes avances se ha establecido como una de las áreas líderes en innovación tecnológica, recibiendo multitud de usos en la industria y en el día a día. Desde los últimos años, el interés de esta área no hace más que aumentar a medida que siguen apareciendo más y más aplicaciones que integran esta tecnología en su flujo de trabajo diario.

- **Reducción de las desigualdades (ODS 10):** El uso de modelos de síntesis de voz neuronal en este trabajo tiene un especial interés para crear sistemas que logren romper las barreras lingüísticas y de comunicación. La síntesis de voz integrada junto a otras tecnologías del procesamiento natural del lenguaje como el ASR o el MT forman herramientas muy poderosas que nos sirven para ayudarnos a comunicarnos mejor.

Además de estos, se ha identificado que los ODS de **Salud y bienestar (ODS 3)**, **Trabajo decente y crecimiento económico (ODS 8)** y **Alianzas para lograr objetivos (ODS 17)** también pueden aplicarse a este trabajo. Respecto al **ODS 3**, aunque no se ha tratado directamente en el trabajo, el uso de la síntesis del habla puede extenderse fácilmente para ayudar a personas con impedimentos del habla o facilitar el acceso a servicios sanitarios o del bienestar a individuos que no dominen la lengua oficial del país en el que se encuentran. En el caso del **ODS 8**, el uso de la síntesis de voz también puede facilitar el crecimiento económico agilizando la comunicación y cooperación en el ámbito laboral de equipos con miembros en distintos países, junto con otras herramientas mencionadas anteriormente como el ASR o el MT. Por último, en relación con el **ODS 17**, la síntesis de habla puede facilitar la comunicación dentro de las instituciones y entre diferentes entidades. Además su aplicación puede ayudar a la comunicación en entornos gubernamentales internacionales.

Respecto al resto de ODS se ha considerado que no proceden, dado que o bien no están relacionados con el área de investigación de este trabajo, o bien el uso de la tecnología propuesta no ha podido extenderse para cubrirlos.

ETS Enginyeria Informàtica
Camí de Vera, s/n, 46022, València
T +34 963 877 210
F +34 963 877 219
etsinf@upvnet.upv.es - www.inf.upv.es