Ecuaciones diferenciales ordinarias neuronales y su aplicación a la resolución de problemas de aprendizaje automático

Miriam González Atienza

Ecuaciones diferenciales ordinarias neuronales y su aplicación a la resolución de problemas de aprendizaje automático

Miriam González Atienza

Memoria del **Trabajo Fin de Máster**.
Máster en Física y Matemáticas (FisyMat)
Universidad de Granada.

Tutorizado por:

Dr. José Andrés González López

Índice general

Siglas

ANN Artificial Neural Network. 16, 18, 21

BAP Band Aperiodicity. 42, 44, 49, 52

DNN Deep Neural Network. 5, 6, 8, 18, 39, 40, 44–46, 49, 51, 52

EDO Ecuación Diferencial Ordinaria. 6–8, 25, 26, 28–30, 32–35, 40–42, 46, 55, 57

LPC Linear Predictive Coding. 13

MCD Mel-Cepstral Distortion. 42, 44
MFCC Mel Frequency Cepstral Coefficients. 1–3, 12, 13, 38–43, 45, 46, 49, 52
MSE Mean Square Error. 39, 40

NODE Neural Ordinary Differential Equations. 1–4, 6–8, 16, 25, 28, 29, 33–35, 39, 41, 44–53, 59–61

PCA Principal Component Analysis. 39
PINNS Physics Informed Neural Networks. 59
PMA Permanent Magnet Articulography. 7, 11, 25, 37–39, 42–44, 51

ResNet Residual Neural Network. 6, 8, 21, 39, 42, 44–50, 52
RMSE Root Mean Squared Error. 2, 3, 42–44

sEEG Stereotactic Electroencephalography. 7
SGD Stochastic Gradient Descent. 19, 40
SSI Silent Speech Interfaces. 9, 10, 12

Índice de figuras

Índice de tablas

Resumen

En este proyecto se aborda un problema de especial interés en el campo del aprendizaje automático: la síntesis de voz a partir de registros de bioseñales generadas por el cuerpo humano durante el proceso de producción del habla. El interés en esta aplicación estriba en que un sistema como éste podría ser la base en un futuro para dispositivos capaces devolver el habla a personas que hayan perdido esta capacidad tras una enfermedad o accidente incapacitante.

Para modelar la relación entre las bioseñales y los parámetros acústicos extraídos de las señales de voz, en este trabajo proponemos la aplicación de un tipo de red neuronal recientemente propuesta en la literatura: las ecuaciones diferenciales ordinarias nueuronales (NODE). En este trabajo presentemos las bases teóricas de estos nuevo modelos mátemáticos y los comparamos con otros modelos de aprendizaje automático del estado del arte, como son las redes neuronales profundas y redes neuronales residuales. Además, los resultados obtenidos se compararán con un modelo simple de regresión lineal.

En este trabajo se emplea una base de datos formada por grabaciones de voz de cinco pacientes, así como de datos de tipo articulatorio adquiridos de forma simultánea.

El sistema completo incluye una etapa de procesado de las señales de voz y los datos extraídos de las bioseñales. Los parámetros acústicos extraídos de las grabaciones son: coeficientes cepstrales en escala Mel (Mel Frequency Cepstral Coefficients (MFCC)), aperiodicidad, logaritmo de la frecuencia fundamental, y decisión sordo / sonoro de cada trama de la voz (*voicing*). La siguiente etapa consiste en aplicar la técnica de reducción de dimensionalidad basada en el análisis de los componentes principales. Una vez caracterizadas las señales de voz y los datos de tipo articulatorio, estos servirán para entrenar los modelos de regresión, con la finalidad de predecir, a partir de las bioseñales, los parámetros acústicos. Para cada tipo de parámetro acús-

tico, se ha entrenado un modelo diferente. En la última etapa, se utiliza el codificador de voz WORLD para sintetizar la voz a partir de las predicciones llevadas a cabo por cada modelo.

Las métricas para evaluar la calidad de la voz sintetizada han sido: métricas objetivas (distorsión de los coeficientes MFCC, error cuadrático medio (Root Mean Squared Error (RMSE)) y porcentaje de error), y métricas subjetivas. Estas últimas se han obtenido mediante un test subjetivo de tipo ABX, en el que los participantes debían elegir qué audio sintetizado es más parecido al audio original. Estas métricas se han calculado para los cuatro modelos de aprendizaje automático, con el objetivo de determinar qué método es capaz de ofrecer una mayor calidad en la voz sintetizada. Los resultados objetivos muestran que la red neuronal NODE es capaz de predecir los MFCC, aperiodicidad y logaritmo de la frecuencia fundamental con una mayor precisión que la red neuronal residual y que el modelo de regresión lineal. Sin embargo, no ocurre lo mismo para el parámetro binario *voicing*, para el que el error de predicción es mayor que para la red neuronal residual. Esta parámetro proporciona una medida de la forma de articular la voz durante el habla, por lo que una predicción errónea de este puede llevar a una falta de entonación en la resultante voz sintetizada. Por otra parte, los resultados obtenidos con el test subjetivo indican que la red neuronal residual ofrece una mayor calidad en la voz sintetizada, habiendo sido elegida el 82.81 % de las veces. El segundo modelo más elegido ha sido NODE, con el 65.91 %.

Además, se han estudiado algunas de las ventajas que supone utilizar una NODE frente a una red neuronal residual. En primer lugar, la naturaleza continua de esta nueva arquitectura de red da lugar a un menor consumo de memoria CPU, que no se ve incrementado al aumentar la complejidad de la red, como ocurre con la red neuronal residual. Además, analizando la etapa de entrenamiento de ambos tipos de red, se ha comprobado que NODE es capaz de disminuir el error de validación durante el entrenamiento en un menor número de iteraciones que las utilizadas con la red residual.

Finalmente, para ilustrar otras aplicaciones de las NODE, se ha aplicado este modelo para resolver un problema clásico de ecuaciones diferenciales ordinarias. En particular, se muestran los resultados para la ecuación de Burgers.

English Abstract

The aim of this project is to address the problem of voice synthesis through data captured from biosignals. We will evaluate the recently proposed neural netowork architecture based on ordinary differential equations (NODE), in order to compare it with other neural networks models. Those are: deep neural networks and residual neural networks. Besides, the results obtained will be compared with a simple linear regression model.

In this study, the data set used contains voice recordings from five patients, as well as articulatory data acquired simultaneously.

The complete system includes a stage for processing the voice signals and the data extracted from the biosignals. The acoustic features extracted from the recordings are: MFCC, aperiodicity, logarithm of the fundamental frequency, and the voiced / unvoiced decision of each frame (voicing). In the next stage, we apply a dimmensionaluty reduction technique based on the analysis of principal components. Once the voice signals and the articulatory data have been characterized, these will be used to train the regression models, in order to predict the acoustic features from the biosignals. For each type of acoustic feature, a different model has been trained. In the last stage, the WORLD vocoder is used to synthesize the speech from the predictions made by each model.

The metrics to evaluate the quality of the synthesized voice are the following: objective metrics (mel cepstral distortion (), mean square error (RMSE), and error rate), and subjective metrics. The latter were obtained through a subjective ABX-tyoe test, in which the participants had to choose which synthesized audio is most similar to the original one. These metrics have been calculated for the four machine learning algorithms, in order to determine which method offers the highest quality in the synthesized speech. The objective results show that the neural network NODE is able to predict the MFCC, aperiodicity and logarithm of the fundamental frequency more

precisely than the residual neural network and the linear regression model. However, for the binary feature *voicing*, the error rate is greater with NODE than with the residual neural network. This type of feature provides a measure of how to voice is articulated during speech. Therefore, a wrong prediction can lead to a lack of intonation in the resulting synthesized voice. On the other hand, the results obtained with the subjective test indicate that the residual neural network offers a higher quality in the synthesized voice, due to the fact that 82.81 % of the participants have chosen the audio corresponding to this model. The second most chosen model was NODE, with 65.91 %.

Finally, some of the advantages of using a NODE over a residual neural network have been studied. Firstly, the continuous nature of this new network architecture results in less CPU memory consumption, which is not increased by increasing network complexity. This is not the case when using a residual neural network. Furthermore, analyzing the training stage of both types of networks, we found that NODE reduces the validation error during training in a smaller number of epochs than those used with the residual network.

To illustrate other applications of NODE, this model has been applied to solve a problem of ordinary differential equations. In particular, we show the results for the Burgers equation.

1 Introducción

Por aprendizaje automático (*machine learning*, en inglés) se denomina a un conjunto de técnicas estadísticas cuyo objetivo es el desarrollo de modelos predictivos capaces de modelar matemáticamente la relación entre un conjunto de datos, para posteriormente aplicar dichos modelos a otros datos sin la necesidad de la intervención humana. Las técnicas de aprendizaje automático se han aplicado en distintos campos, desde reconocimiento de patrones, visión por ordenador, ingeniería espacial, finanzas, entretenimiento y biología computacional hasta aplicaciones médicas. Los algoritmos de aprendizaje automático se dividen normalmente en supervisados o no supervisados. En los algoritmos de tipo supervisado, los modelos predictivos se entrenan usando conjuntos de datos previamente etiquetados (p. ej. imágenes de animales con sus respectivas etiquetas de espacios), con el objetivo de predecir la etiqueta para conjuntos de datos no etiquetados. Por otra parte, los algoritmos de tipo no supervisado se utilizan cuando los datos de entrenamiento de los que disponen no se encuentran etiquetados. Este tipo de algoritmos tratan de inferir la función que mejor describe la estructura de los datos y la relación entre los mismos.

Dentro de las técnicas de aprendizaje automático supervisado, las redes neuronales artificiales han experimentado un gran desarrollo durante las últimas dos décadas debido a los buenos resultados obtenidos en problemas prácticos como la clasificación de imágenes [3], el reconocimiento automático de voz [4], o el procesado del lenguaje natural [5], entre otros. Las redes neuronales artificiales son modelos matemáticos bio-inspirados capaces de aprender directamente de los datos con pocas/ningunas suposiciones. Constituyen una de las herramientas principales utilizadas en aprendizaje automático. Otro de los avances importantes en las redes neuronales ha sido la llegada de las redes neuronales profundas, las Deep Neural Networks (DNNs) (del inglés, *Deep Neural Networks*) [6]. Una DNN consiste en una red neuronal artificial formada por una capa de entrada, una o varias capas ocultas, y una capa de salida. Las DNN se pueden utilizar para resolver problemas tanto de clasificación como de regresión y,

como se ha comentado anteriormente, a día de hoy constituyen una de las principales herramientas usadas en el campo del aprendizaje automático [7]. El mercado del aprendizaje automático tenía un valor de 2,28 mil millones de dólares en 2017 y se espera que alcance los 18,16 mil millones de dólares en 2023 [8]. En términos de aplicaciones, el reconocimiento de imágenes tiene la mayor participación en el mercado del aprendizaje profundo. Se espera que el mercado de la minería de datos experimente el mayor crecimiento del aprendizaje profundo durante el período de pronóstico. La creciente demanda de reconocimiento de patrones, reconocimiento óptico de caracteres, reconocimiento de códigos, reconocimiento facial, reconocimiento de objetos y procesamiento de imágenes digitales está impulsando el crecimiento del reconocimiento de imágenes en el mercado. Con el advenimiento de las nuevas tecnologías, el procesamiento del lenguaje natural y la minería de datos visuales se han desarrollado utilizando técnicas de aprendizaje profundo.

Recientemente, una nueva arquitectura de DNNs ha despertado gran interés entre la comunidad científica al combinar el concepto clásico de red neuronal artificial con las ecuaciones diferenciales ordinarias (Ecuación Diferencial Ordinaria (EDO)): son las denominadas ecuaciones diferenciales ordinarias neuronales (NODE) [2]. Las NODE fueron propuestas en 2018 por Tian Qi Chen, Yulia Rubanova, Jesse Bettencourt y David Duvenaud en [2], siendo este trabajo premiado como el mejor artículo en el prestigioso congreso NeurIPS. En concreto, dos ejemplos citados en este artículo generaron especial interés. Estos son el modelado de flujo continuo y la pseudo-equivalencia entre la profundidad de las conexiones en las redes neuronales residuales (Residual Neural Network (ResNet)) [9] y el número de evaluaciones de la función que describe la dinámica del estado oculto en las NODE. Los flujos de normalización continuo se utilizan para modelar la distribución de un conjunto de puntos que tienen una forma determinada. Pueden ser vistos como un campo vectorial en el espacio euclidiano de tres dimensiones, que induce una distribución de puntos mediante la transformación de una distribución previa genérica (por ejemplo, una Gaussiana estándar). Por otra parte, la equivalencia entre redes residuales y redes NODE se estudia con detalle en el capítulo 3. Las NODE, al contrario que las redes neuronales clásicas, utilizan una DNN para modelar la tasa de cambio en el estado de un sistema.

En aplicaciones de aprendizaje profundo, las NODE ofrecen dos ventajas principales. En primer lugar, estas pueden ser entrenadas con un coste de memoria constante en el número de evaluaciones de la función, mientras que una red neuronal estándar tendría un coste de memoria que crece linealmente. Otra de las ventajas que ofrecen las NODE, es la computación adaptativa. El enfoque estándar para construir *solvers* de ecuaciones diferenciales ordinarias (EDO) adaptativos consiste en calcular la diferen-

cia en la trayectoria predicha realizada por dos métodos de extrapolación distintos. Si esta diferencia crece, significa que al menos uno de los métodos de extrapolación no está generando las predicciones correctas. Por lo tanto, estos métodos intentan recuperarse comenzando de nuevo y haciendo predicciones con pasos más pequeños en el tiempo. Algunas de las aplicaciones de las NODE incluyen modelado de series temporales, y flujos de normalización continuos [10]. Un flujo de normalización es un mapeo invertible entre una distribución de probabilidad y una distribución normal estándar. Se suele utilizar para estimación de densidad e inferencia estadística. Además de estas aplicaciones, las NODE se pueden utilizar para asociar datos de entrada con un conjunto de parámetros, y para aprender las funciones que mejor describen el conjunto de datos, es decir, problemas de regresión o clasificación. En este caso, la arquitectura del modelo de red neuronal NODE consta de una capa EDO, seguida de una capa lineal.

En este trabajo se aborda un problema de aprendizaje automático en el que el objetivo es sintetizar voz a partir de señales capturadas durante el proceso de producción del habla. Para ello, se modela la relación entre los vectores de parámetros extraídos de las bioseñales, y los vectores de parámetros calculados a partir de la señal de voz usando técnicas de aprendizaje automático. Existen distinas técnicas para capturar la actividad cerebral, como Permanent Magnet Articulography (PMA), que utiliza datos de tipo articulatorio obtenidos de los labios y la lengua [11, 12], o electrodos de tipo Stereotactic Electroencephalography (sEEG), en el que se utilizan electrodos profundos para definir coordenadas en el cerebro [13]. El proceso de adquisición de datos se realiza de forma paralela, de forma que los datos de los sensores que capturan las bioseñales se graban al mismo tiempo que la voz del sujeto.

1.1 Objetivos

Los objetivos generales de este trabajo son:

- Evaluar la novedosa arquitectura de NODE en un problema de aprendizaje automático práctico en el que se pretende sintetizar voz a partir de datos capturados del tracto vocal usando PMA.
- Comparar las NODE con otras arquitecturas del estado del arte para el problema mencionado.

Finalmente, aunque no directamente relacionado con los tópicos de este trabajo,

he considerado ilustrador añadir en un apéndice una breve aplicación de las NODE a la resolución de problemas clásicos de EDO.

1.2 Estructura

Esta memoria se ha estructurado de la siguiente forma. En el capítulo 2 se revisan los antecedentes de la literatura en los que se aborda el problema de síntesis de voz a partir de parámetros del habla, así como los estudios recientes relacionados con las NODE. Además, se describe el proceso de extracción de parámetros, así como los fundamentos de los modelos de regresión implementados modelo de regresión lineal y logística, DNN, ResNet y NODE. El capítulo 3 está dedicado a la descripción de las redes neuronales NODE. Finalmente, en el capítulo 4 se expone la base de datos utilizada, los detalles de implementación de los distintos métodos y los resultados obtenidos.

2 Revisión del estado del arte

En este capítulo se describe el estado de arte del problema que se aborda en este trabajo. Así, en la sección 2.1, se introduce el concepto de interfaz oral silenciosa (SSI; del inglés *Silent Speech Interface*), que es un sistema automático encargado de descodificar el habla (i.e. las palabras que ésta pronuncia) a partir de registros de bioseñales generadas por el cuerpo humano durante el proceso de producción del habla[1]. Para descodificar el habla a partir de susodichas bioseñales, las SSIs suelen hacer uso de técnicas de aprendizaje automático, de las cuales en este capítulo se presenta una breve introducción en la sección 2.2.

2.1 Interfaces orales silenciosas

Las interfaces orales silenciosas (SSI) son dispositivos de asistencia para restaurar la comunicación oral mediante la descodificación del habla a partir de bioseñales no acústicas generadas durante la producción del habla.

La Figura 2.1 muestra el diagrama de bloques para un sistema de comunicación basado en SSI. En primer lugar, se capturan las actividades relacionadas con la producción del habla utilizando sensores específicos, dando lugar a distintas bioseñales. El siguiente paso consiste en extraer un conjunto de parámetros a partir de estas bioseñales mediante técnicas de procesado de señal. Finalmente, se descodifica el habla a partir de estos parámetros. Las siguientes secciones están dedicadas a explicar cada paso en el proceso completo.

[1] Un ejemplo por todos conocidos de *habla silenciosa* es el proceso de lectura de labios. La lectura automática de labios sería un ejemplo de interfaz oral silenciosa.

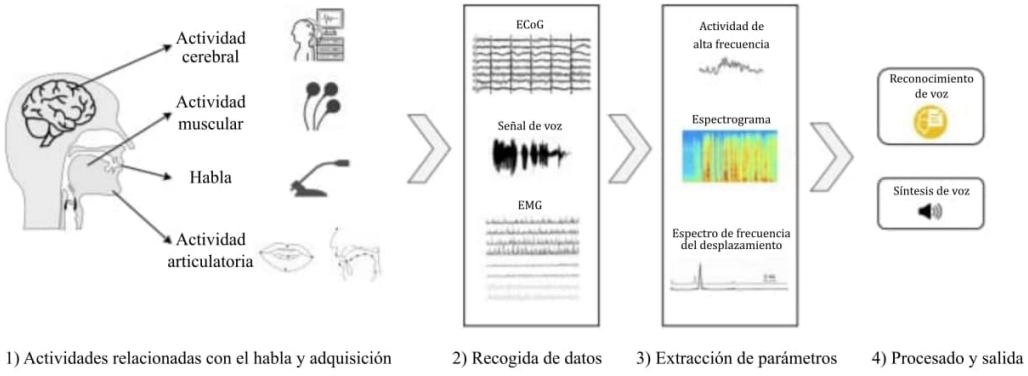

Figura 2.1: Diagrama de bloques de un sistema basado en SSI. Adaptado de [1].

2.1.1 Extracción de parámetros

En esta sección se describen los parámetros más utilizados en la literatura para representar la voz y las bioseñales, y que han sido empleados en esta memoria.

2.1.2 Procesamiento de las bioseñales

Para la extracción de parámetros a partir de las señales cerebrales proporcionadas por los electrodos sEEG, en los estudios [14, 15] se centraron en la actividad cerebral en la banda de alta frecuencia *gamma*, entre 70 y 200 Hz, ya que contiene información relevante de los procesos del habla y el lenguaje. Para la extracción de parámetros en la banda gamma y la atenuación del primer y segundo armónico del ruido de línea en 50 Hz, utilizaron un filtro IIR pasobanda y dos filtros elípticos IIR de tipo *notch*, respectivamente, cada uno de orden 8. Las señales resultantes fueron segmentadas en ventanas de 50 ms con un solapamiento de 10 ms, para capturar la compleja dinámica de los procesos del habla. Para cada una de estas ventanas, se calculó la potencia de la señal. Además, a cada ventana se le añadió información relacionada con el contexto (200 ms previos) para incorporar cambios temporales relacionados con la dinámica del cerebro.

En el estudio [16], a partir de los datos recogidos mediante ECoG, extrajeron la amplitud de la banda de alta frecuencia *gamma* mediante una transformada de Hilbert, seguido de un submuestreo a 200 Hz. Además, tambiéne xtrajeron la componente de

baja frecuencia (1-30 Hz) utilizando un filtro Butterworth de quinto orden.

Articulografía magnética permanente

En el estudio [11], las señales PMA provenientes de 9 canales fueron primero sobremuestreadas de 100 Hz a 200 Hz para que coincidiesen con los 5 ms de frecuencia de trama. Las tramas PMA se definieron como segmentos superpuestos de datos calculados cada 5 ms a partir de ventanas de análisis de 25 ms (la misma velocidad de tramas que para las características de voz). En este trabajo, utilizaremos las bioseñales obtenidas con esta técnica.

2.1.3 *Vocoder* WORLD

En este trabajo se ha empleado un *vocoder* del estado del arte denominado WORLD [17]. Esta técnica, divide la señal en segmentos de 25 ms de duración (con un solapamiento de 20 ms) y, de cada segmento, calcula los siguientes conjuntos de parámetros: envolvente espectral, frecuencia fundamental (F_0) y aperiodicidades. Estos parámetros se describen en más detalle en los siguientes apartados. El *vocoder* WORLD Está compuesto por tres algoritmos para obtener tres parámetros acústicos y por un algoritmo de síntesis. La Figura 2.2 ilustra el funcionamiento del sistema. Los algoritmos utilizados se describen en más detalle en la siguiente sección.

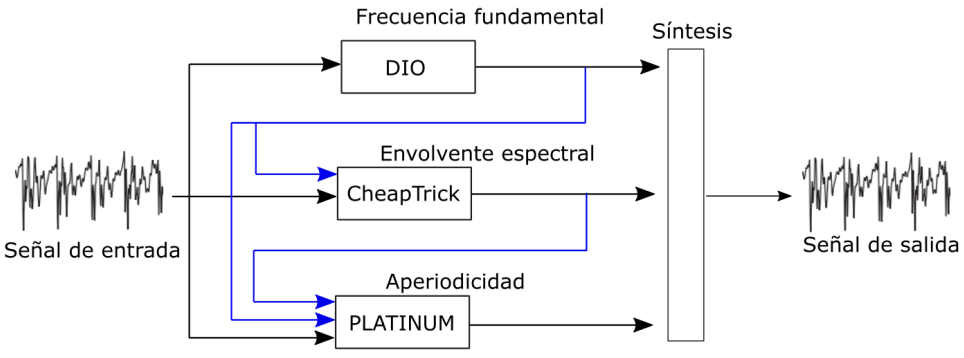

Figura 2.2: Esquema del *vocoder WORLD*

2.1.4 Parámetros acústicos

Como se aprecia en la Figura 2.1, durante la etapa de entrenamiento de una SSI las señales acústicas son procesadas de forma automática con objeto de reducir su redundancia y disminuir el número de parámetros necesarios para representarlas, en un proceso denominado *vocoding*. En concreto, las señales de voz son procesadas por un *vocoder* que las divide en segmentos de longitud fija donde la señal de voz se supone estacionaria en sentido estadístico (típicamente de unos 20 ms de duración) y de cada segmento extrae una serie de parámetros acústicos que modelan dicho segmento de forma compacta. En la fase de síntesis, el *vocoder* es capaz de resintetizar la señal de voz a partir de dicha secuencia de vectores de parámetros.

- **Coeficientes Cepstrales en escala Mel (MFCC**:
  Los Coeficientes Cepstrales en escala Mel son utilizados para la representación del habla basados en la percepción auditiva humana. Es decir, aportan información sobre la función de transferencia del tracto vocal. Representan las correlaciones existentes entre las distintas componentes en frecuencia del espectro logarítmico.
  Los MFCC, introducidos por Davis y Mermelstein en 1980, son coeficientes ampliamente utilizados en reconocimiento automático de voz [**?**, **?**]. Se ha utilizan en numerosos estudios para representar el habla [11, 13, 14, 16, 18].
  Se derivan de la siguiente forma:
  - Se divide la señal en tramas de 20 a 40 ms, con solapamiento (10 ms típicamente). Los MFCC se calculan sobre cada una de estas tramas.
  - Se calcula la transformada discreta de Fourier (DFT) de cada uno de los fragmentos de la señal y se obtiene su potencia espectral.

    $$S_i(k) = \sum_{n=1}^{N} s_i(n)h(n)e^{-j2\pi kn/N} \quad 1 \leq k \leq K, \tag{2.1}$$

    donde $s(n)$ es la señal original, $s_i(n)$ representa la señal dividida en tramas, $h(n)$ es la ventana que se le aplica a la señal $s(n)$, de longitud igual a N, y K representa la longitud de la DFT.
  - A continuación, se calcula el valor absoluto de la DFT compleja, $S_i(k)$ y se eleva al cuadrado. El resultado obtenido es el periodograma,

    $$P_i(k) = \frac{1}{N}|S_i(k)|^2. \tag{2.2}$$

- Se aplica el banco de filtros correspondientes a la escala *Mel* al espectro obtenido y se suman las potencias espectrales en cada uno de ellos. Consiste en un conjunto de, normalmente, 26 filtros triangulares aplicados al periodograma calculado en 2.2. La escala *Mel* asigna mayor resolución a las bajas frecuencias de forma análoga a como se hace en el sistema auditivo humano.
- Se toma el logaritmo de todas las energías de cada frecuencia *Mel* y se aplica la transformada de coseno discreta (DCT) a estos logaritmos.
  Por ejemplo, es habitual escoger 24 filtros triangulares, por lo que tras aplicar la DCT a los logaritmos calculados obtenemos 24 coeficientes cepstrales.

Con el objetivo de obtener una caracterización dinámica de la señal de voz, también se suelen extraer las primeras derivadas de estos coeficientes. En este trabajo, hemos extraídos los primeros 25 MFCC de las señales de voz.

Frecuencia fundamental (F_0):

$F0$ se define como la inversa del mínimo periodo de la señal periódica. En particular, existen dos tipos de características relacionadas con este parámetro: de tipo temporal, como la autocorrelación, y características espectrales como el Cepstrum [19]. El codificador de voz WORLD [17] implementa el método DIO [20] para estimar F_0. El algoritmo consta de tres etapas. El primer paso consiste en aplicar un filtro paso baja a la señal con distintas frecuencias de corte. Si la señal filtrada está compuesta únicamente por el componente fundamental, forma una onda sinusoidal con periodo T_0, que es el periodo fundamental. Debido a que la frecuencia fundamental es un parámetro desconocido, es necesario aplicar un gran número de filtros con distintas frecuencias de corte. El siguiente paso consiste en calcular las frecuencias candidatas a F_0 en cada señal filtrada. Debido a que una señal que está únicamente compuesta por la componente principal tiene forma sinusoidal, los cuatro intervalos de la forma de onda tienen el mismo valor. Su desviación estándar es entonces asociada con una medida de fiabilidad, y su media se define como un candidato a F_0. En el último paso, se elige al candidato con la mayor fiabilidad.

Envolvente espectral: La envolvente espectral es un parámetro importante en procesamiento de voz. Algoritmos como Cepstrum y Codificación Predictiva Lineal (Linear Predictive Coding (LPC)) se suelen utilizar frecuentemente para su cálculo. El *vocoder* WORLD emplea el algoritmo CheapTrick [21]. Este utiliza un análisis síncrono de *pitch* y una ventana de Hanning de tamaño $3T_0$. En primer lugar, se calcula la

potencia del espectro a partir de la señal enventanada. La potencia total de esta señal se estabiliza temporalmente mediante la siguiente ecuación:

$$\int_0^{3T_0} (y(t)\omega(t))^2 \, dt = 1.125 \int_0^{T_0} y^2(t) \, dt, \tag{2.3}$$

donde $y(t)$ representa la forma de onda, y $\omega(t)$ representa la ventana que se le aplica. Posteriormente, se suaviza la potencia del espectro con una ventana rectangular de ancho $2\omega_0/3$, de la siguiente forma:

$$P_s(\omega) = \frac{3}{2\omega_0} \int_{-\omega_0/3}^{\omega_0/3} P(\omega + \lambda) d\lambda, \tag{2.4}$$

donde ω_0 se define como $2\pi/T_0$. Finalmente, se aplica el siguiente proceso:

$$P_l(\omega) = \exp(\mathscr{F}[l_s(\tau) l_q(\tau) p_s(\tau)]), \tag{2.5}$$

$$l_s(\tau) = \frac{\sin(\pi f_0 \tau)}{\pi f_0 \tau}, \tag{2.6}$$

$$l_q(\tau) = q_0 + 2q_1 \cos\left(\frac{2\pi\tau}{T_0}\right), \tag{2.7}$$

$$p_s(\tau) = \mathscr{F}^{-1}[\log(P_s(\omega))], \tag{2.8}$$

donde $l_s(\tau)$ representa la función para suavizar el logaritmo de la potencia del espectro, y eliminar componentes que varían con el tiempo. $l_q(\tau)$ representa la función para la recuperación del espectro. Esta función de recuperación mejora la estimación. $p_s(\tau)$ representa el Cepstrum de la potencia del espectro $P_s(\omega)$. Los símbolos $\mathscr{F}[]$ y $\mathscr{F}^{-1}[]$ representan las transformada de Fourier e inversa usuales. q_0 y q_1 son parámetros relacionados con la recuperación del espectro. Valores típicos de los parámetros $q_0 = 1.18$ y $q_1 = -0.09$ se han obtenido en [21] a través de una evaluación exploratoria, y serán los usuados en esta memoria.

Aperiodicidad: La aperiodicidad [22] es un parámetro utilizado para representar la aperiodicidad de la señal de voz. El codificador de voz WORLD [17] implementa el método PLATINUM [23] para extraer el parámetro de aperiodicidad, en el que utiliza la señal de excitación directamente calculada a partir de la forma de onda, la frecuencia fundamental (F_0), y la envolvente espectral. PLATINUM aplica una ventana de longitud $2T_0$, donde T_0 es el periodo de la señal. El espectro de la señal enventanada

$X(\omega)$ se divide entre el mínimo espectro de fase $S_m(\omega)$, calculado como

$$S_m(\omega) = \exp(\mathscr{F}[c_m(\tau)]), \quad c_m(\tau) = \begin{cases} 2c(\tau) \ (\tau > 0) \\ c(\tau) \ (\tau = 0), \\ 0 \ (\tau < 0) \end{cases} \quad c(\tau) = \mathscr{F}^{-1}[\log(P_l(\omega))], \quad (2.9)$$

La señal de excitación extraída $x_p(t)$ se expresa de la siguiente forma:

$$x_p(t) = \mathscr{F}^{-1}[X_p(\omega)], \quad (2.10)$$

$$X_p(\omega) = \frac{X(\omega)}{S_m(\omega)}. \quad (2.11)$$

El mencionado PLATINUM precisa determinar las posiciones temporales asociadas con cada vibración de las cuerdas vocales. Para ello, se hace uso del contorno de la frecuencia fundamental y de la forma de onda. En primer lugar, se determina la parte sonora (aquella que se produce mediante la vibración de las cuerdas vocales). En segundo lugar, se determina la posición central de dicha sección (en el dominio del tiempo), t_a. El siguiente paso consiste en calcular un intervalo $t_a \pm T_0$, y la posición temporal en este intervalo con un mayor valor de $y(t)^2$ se define como el origen. Una vez que el punto de origen se ha establecido, el algoritmo de síntesis del *vocoder* automáticamente calcula las otras posiciones temporales asociadas con la vibración de las cuerdas vocales. Este proceso se lleva a cabo en todos los tramos sonoros.

2.2 Aprendizaje automático

En un problema clásico de modelado de datos, tenemos un conjunto de N pares de puntos, denominado conjunto de entrenamiento, $D = \{(x_1, y_1), (x_2, y_2), ..., (x_N, y_n)\}$, donde X es el dominio de entrada e Y es el dominio de salida. Dado un nuevo punto, $\widehat{x}$, queremos predecir cuál será su valor correspondiente $\widehat{y}$. En función de los valores de Y, el problema se denomina de clasificación (cuando sus valores son discretos), o de regresión (cuando sus valores son continuos).

El enfoque del aprendizaje automático consiste en encontrar una función que aproxime la relación $y = f(x)$. Esto es, el objetivo es encontrar una relación entre los dominios X e Y, $f : X \rightarrow Y$.

Dos técnicas básicas permiten resolver este problema: mediante regresión, y mediante ecuaciones diferenciales ordinarias. A continuación, describimos la primera

técnica, y dejamos para el capítulo 3 la descripción de la segunda, ya que nos servirá de introducción al cambio de paradigma que suponen las NODE en el marco de este trabajo.

2.2.1 Regresión lineal por mínimos cuadrados

Supongamos que X e Y son simplemente la recta real ordinaria $\mathbb{R}$, y tenemos un conjunto de datos distribuidos de forma arbitraria, de forma que queremos encontrar la función $f : \mathbb{R} \to \mathbb{R}$ que mejor describe los datos. Una posible forma de resolver este problema es aplicar *regresión lineal*. Asumiendo que los datos describen una forma aproximadamente lineal, el objetivo es buscar una recta de la forma:

$$\widehat{Y} = a + bX, \tag{2.12}$$

de modo que se ajuste a la nube de puntos. Para ello, se puede utilizar el conocido método de mínimos cuadrados que consiste en minimizar la suma de los cuadrados de los errores (diferencias entre los valores reales observados (y_i) y los valores estimados ($\widehat{y}_i$)).

$$\sum_{i=1}^{n} e_i^2 = \sum_{i=1}^{n} (y_i - \widehat{y}_i)^2 \tag{2.13}$$

2.2.2 Redes neuronales artificiales

Antes de introducir las Ecuaciones Diferenciales Ordinarias Neuronales (NODE), es necesario dedicar un apartado a explicar brevemente cómo funciona una red neuronal, ya que ésta será la base de las NODE.

Las redes neuronales artificiales (Artificial Neural Network (ANN)) son sistemas bio-inspirados capaces de aprender directamente de los datos con muy escasas suposiciones sobre éstos. Fueron presentadas por McCulloch y Pitts en 1943 [24]. Las ANN constituyen una de las herramientas principales utilizadas en aprendizaje automático. Esto se debe a la utilización de la técnica *backpropagation,* la cual permite a la red ajustar los parámetros de sus capas ocultas en aquellas situaciones en las que la salida no se ajusta a su valor esperado, es decir, cuando se comete un error. Existen distintos tipos de redes neuronales, como redes neuronales recurrentes (RNN), redes neuronales convolucionales (CNN), o redes neuronales profundas (DNN). En los siguientes apartados nos centraremos únicamente en las redes de tipo *feed-forward*, en

las que las conexiones entre las unidades no forman un ciclo, sino que la información se mueve siempre en una única dirección: hacia adelante.

2.2.3 Redes neuronales *feed-forward*

Neuronas El elemento básico de procesamiento en una red neuronal es el perceptrón, creado por Frank Rosenblatt [25]. Se refiere a la neurona artificial, y es el modelo matemático más simple de una neurona.

Consiste en una combinación lineal de los elementos de entrada, junto con una función de activación no lineal. A nivel esquemático, el modelo de neurona se representa en la figura 2.3.

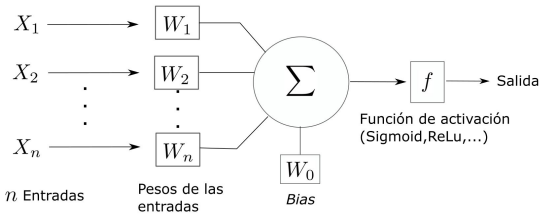

Figura 2.3: Modelo de neurona

Las entradas (X_i) representan el estímulo que la neurona artificial recibe del entorno que la rodea, y la salida (y) es la respuesta a dicho estímulo, previa aplicación de la función de activación. De un modo simple

$$y = f\left(\sum_{j=1}^{n} W_j \mathbf{X}_j + b\right) \tag{2.14}$$

donde W_j es un peso convenientemente aplicado a cada entrada, y b es una constante que se conoce como *bias* en la literatura.

La función de activación se elige de acuerdo a la tarea que realiza la neurona y determina la salida del modelo, su precisión y la eficiencia computacional del modelo de entrenamiento. Esta función se encuentra asociada a cada neurona de la red, que decide si se activa o no basándose en la relevancia de dicha neurona en la predicción final del modelo. Funciones de activación comunes son la función Sigmoide o tangente hiperbólica, que permiten a normalizar la salida de cada neurona en rangos distintos

([0, 1] o [−1, 1]). Las redes neuronales suelen utilizar funciones de activación no lineales, lo cual ayuda a la red a aprender datos complejos y proporcionar predicciones fiables. Algunas de las funciones de activación más utilizadas se muestran en la Tabla 2.1.

Nombre	**Rango**	**Definición**
Sigmoidal	$[0, 1]$	$f(x) = \dfrac{1}{1 - e^{-x}}$
Tangente hiperbólica	$[-1, 1]$	$f(x) = \dfrac{2}{1 + e^{-2x}} - 1$
ReLU (*rectified linear unit*)	$[0, \infty)$	$f(x) = \max(0, x)$
Softmax	$[0, 1]$	$f(x_j) = \dfrac{e^{x_j}}{\sum_{k=1}^{K} e^{x_k}}$

Tabla 2.1: Tabla resumen con las principales funciones de activación usadas por las redes neuronales artificiales.

Capas. Las neuronas se organizan por niveles o capas con un número determinado de neuronas en cada una de ellas. Cuando las ANNs se organizan por niveles o capas, estamos ante una DNN, en la que se pueden distinguir tres tipos de capas:

- **Capa de entrada.** Es la capa cuyos nodos (entradas) reciben directamente la información proveniente de las fuentes externas de la red (variables de entrada).
- **Capas ocultas.** Son internas a la red y no tienen contacto directo con el entorno exterior. Las neuronas de esta capa pueden estar interconectadas de distintas maneras, lo que determina la topología de la red.
- **Capa de salida.** Los nodos (salidas) de esta capa permiten realizar la transferencia de información de la red hacia el exterior (variables de salida).

A modo de ejemplo, en la Figura 2.4 se muestra la arquitectura de una red neuronal profunda con dos capas ocultas.

2.2.4 Entrenamiento de una red neuronal: *backpropagation*

En primer lugar, denotaremos a la función de pérdida de la red neuronal como $\mathcal{L}$. Se trata de una función que evalúa la desviación entre las predicciones realizadas por la red neuronal y los valores reales de las observaciones utilizadas durante el aprendizaje. Cuanto menor es el resultado de esta función, más eficiente es la red neuronal. Su minimización, es decir, reducir al mínimo la desviación entre el valor predicho y

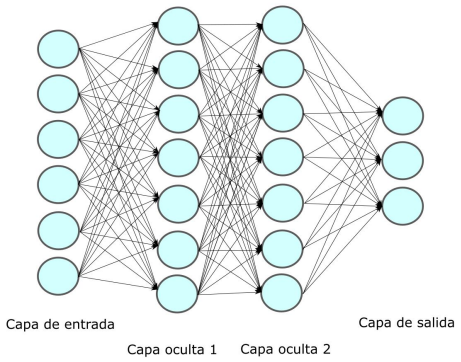

Figura 2.4: Arquitectura de una red neuronal profunda con dos capas ocultas

el valor real para una observación dada, se lleva a cabo ajustando los distintos pesos de la red neuronal.

Una vez que la red neuronal ha sido diseñada, el proceso de entrenamiento es el encargado de asegurar que los pesos individuales W_j asociados a cada entrada de la neurona sean correctos. Es decir, que la red neuronal proporcione la salida correcta. La técnica de *BackPropagation* es una aplicación de la regla de la cadena, capaz de optimizar estos pesos. El objetivo es minimizar la función de error. Para ello, reajusta los pesos W_j utilizando el método de descenso en gradiente. Es decir, encuentra el camino dentro de la red que proporciona un menor error. El gradiente de la función f respecto de la variable de entrada $\mathbf{x}$ define el cambio de ésta con el incremento/decremento de $\mathbf{x}$ (su derivada parcial, porque f depende de varios parámetros). Por lo tanto, como indica el nombre del propio método, se está realizando un descenso desde un valor elevado de la función objetivo hasta un valor menor, siguiendo la dirección que corresponda al camino con el mínimo gradiente.

La Figura 2.5 muestra el funcionamiento del algoritmo de *Backpropagation* con descenso en gradiente

Una alternativa al método de gradiente clásico, es el método de Descenso en Gradiente Estocástico (Stochastic Gradient Descent (SGD)), que, en lugar de calcular el gradiente completo de la función objetivo que promedia entre todas las muestras, el gradiente estocástico solamente toma una muestra, calcula la pérdida $\mathcal{L}$ y el gradiente de la pérdida con respecto a los parámetros, y da un paso en la dirección negativa del

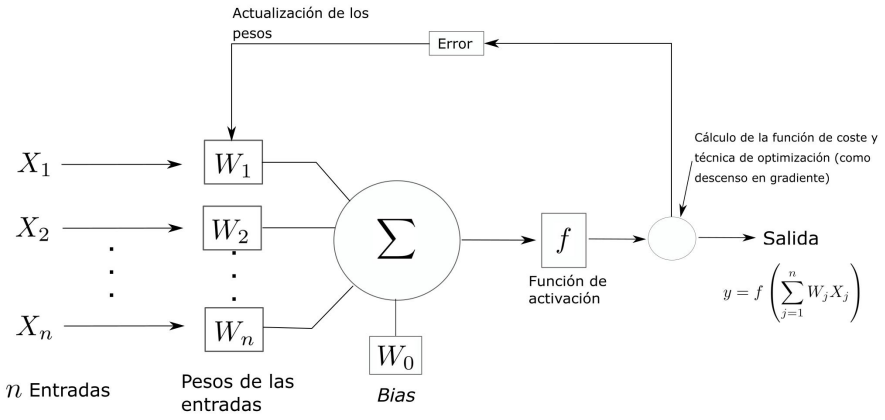

Figura 2.5: Algoritmo de *Backpropagation*

gradiente

$$w \leftarrow w - \eta \frac{\partial \mathcal{L}(x[p], y[p], w)}{\partial w}, \tag{2.15}$$

donde η es una constante, y w son los parámetros a optimizar. Notar que 2.15, simplemente aproxima w por su valor original minorado en el tamaño del paso por gradiente de la función de pérdida con relación a w (evaluada en una muestra dada, $(x[p], y[p])$).

El proceso de entrenamiento es el siguiente:

- Se dispone de un conjunto de vectores de entrenamiento: $\{(\mathbf{x}(k), \mathbf{y}(k)); k = 1, ...T\}$, donde T es el tamaño del conjunto de entrenamiento.
- Consideramos que la red aplica una función sobre cada vector de entrada,

$$\mathbf{x}(k) \ : \ \mathbf{y}^{(L)}(k) = F(\mathbf{x}(k); \mathbf{W}, \mathbf{b}).$$

- Se establece una función de coste o pérdida, $\mathcal{L}(\mathbf{W}, \mathbf{b})$ que debe optimizarse en función de los parámetros de la red $(\mathbf{W}, \mathbf{b})$.

Uno de los problemas comunes al entrenar un algoritmo de *machine learning* es el sobreentrenamiento (*overfitting*). Este fenómeno da lugar cuando el modelo sólo se ajusta a aprender los casos particulares con los que ha sido entenado y es incapaz de reconocer nuevos datos de entrada.

2.2.5 Redes neuronales residuales

Un tipo de ANN que merece una sección aparte son las redes neuronal residuales (ResNet). Las redes residuales se inspiran en el hecho biológico de que algunas neuronas en el cerebro se conectan con neuronas en capas no necesariamente contiguas, saltando capas intermedias. Fueron propuestas por Kaiming et al.[26]. Su mayor impacto se debe a que el uso de este tipo de redes permitió por primera vez entrenar redes muy profundas (de más de 100 capas), controlando con éxito el problema de Desvanecimiento de Gradiente (*Vanishing Gradient*). En pocas palabras, este problema consiste en que redes con muchas capas ocultas presentan problemas durante el entrenamiento debido a que la propagación del gradiente en estas redes puede hacer que el gradiente resulte extremadamente pequeño. El término "residuo" se refiere al cambio que hay que introducir en la predicción para que coincida con el valor real. Las funciones básicas de una red residual son las siguientes:

- Utilizar la función identidad para controlar la degradación de la precisión y de la tasa de error, causadas por un aumento del número de capas en la red. Al incrementar la profundidad de una red neuronal, la precisión comienza a saturarse y se degrada rápidamente. Esta degradación no es debido a un problema de sobreentrenamiento (como se podría pensar).
- Continuar aprendiendo el valor de los residuos para que el valor predicho coincida con el valor real.

Una ResNet se compone de un conjunto de bloques residuales como el que se muestra de forma esquemática en la Figura 2.6. Denotaremos x a la entrada. Asumimos que la función que queremos obtener mediante el proceso de aprendizaje de la red es $f(x)$, la cual se utilizará como entrada a la función de activación al final. En 2.6, la porción encerrada en una caja de puntos es la encargada de aprender la asociación residual $f(x) - x$, razón por la cual el bloque residual recibe su nombre. Si el mapeo de identidad $f(x) = x$ es la asociación deseada, solo es necesario igualar los pesos y los *bias* de la capa superior dentro de la línea de puntos a cero. La línea continua que lleva la entrada de la capa x al operador suma se denomina " conexión residual". Con la utilización de bloques residuales, las entradas se pueden propagar hacia delante más rápidamente a través de las conexiones residuales, que si no se utilizaran bloques residuales.

Utilizando la definición de función identidad de [26],

$$y = F(x, \{W_i\}) + x, \tag{2.16}$$

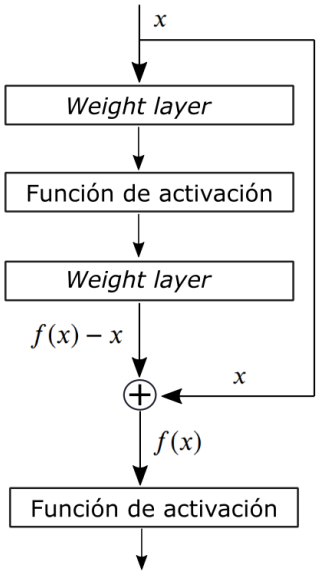

Figura 2.6: Esquema de un bloque residual

con x e y los vectores de entrada y salida de las capas, esa función identidad $F(x, \{W_i\})$ representa aquí el residuo que se va a aprender.

Durante el entrenamiento, la red residual modifica el valor de los pesos hasta que la salida es equivalente a la función identidad (es decir, el valor predicho es igual al valor real). Durante el proceso, el resultado de la función residual se convierte en 0 en algún momento, y el valor de x se inserta en las capas ocultas. Por lo tanto, no es necesario aplicar corrección de errores. A su vez, la función identidad ayuda a construir una red más profunda. La función residual mapea la identidad, los pesos y los bias para ajustar la predicción al valor real.

Respecto a su implementación general en Python, el primer bloque[2] de la red residual aplica una convolución en una dimensión a la señal de entrada, compuesta a su vez por varios canales de entrada. En el caso más simple, el valor de salida de la capa con tamaño de entrada N, C_{in}, L y salida N, C_{out}, L_{out} se describe como:

$$out(N_i, C_{out\,j}) = bias(C_{out\,j}) + \sum_{k=0}^{C_{in}-1} weight(C_{out\,j}, k) * input(N_i, k) \tag{2.17}$$

[2]En este apartado, vamos a prescindir de una notación abstracta, en favor de una notación de pseudo-código, que es usual en la literatura específica de este tema [27].

donde $*$ es el operador de correlación cruzada, N es el tamaño del bloque, C es el número de canales, y L es la longitud de la señal de entrada. La variable *weights* son los pesos que la red debe ir aprendiendo, de tamaño

$$(\textit{canales de salida}, \textit{canales de entrada}, \textit{tamaño del kernel}).$$

El valor de estos pesos se muestrea de una distribución uniforme $U(-\sqrt{k}, \sqrt{k})$, donde $k = \frac{groups}{C_{in} \cdot Kernel}$

El tamaño del bloque de salida es igual a

$$L_{out} = \frac{L_{in} + 2 \cdot padding - dilation(kernel - 1) - 1}{stride} + 1 = 128 - (3-1) - 1 + 1 = 126$$

El argumento "stride" indica el número de saltos en la convolución. Excepto en la primera convolución, este argumento toma el valor 2, lo que afecta al tamaño del resultado de la convolución, disminuyéndolo.

Después de cada bloque de convolución se aplica una normalización de grupo. Los canales de entrada se separan en subgrupos, y se calcula la media y la desviación estándar de cada grupo por separado. El valor de salida viene dado por

$$y = \frac{x - E[x]}{\sqrt{Var[x] + \epsilon}} \cdot \gamma + \beta \tag{2.18}$$

donde ϵ es un valor que se añade en el denominador para la estabilidad numérica ($\epsilon = 10^{-5}$), y γ y β son parámetros que la red va aprendiendo.

En el siguiente bloque se aplica la función de activación (p.ej ReLU). Se pueden agrupar las operaciones descritas en bloques residuales. A la salida del conjunto de bloques residuales, se aplica una normalización de grupo, la función de activación y la función "Adaptive Average Pooling", que devuelve la media del tensor de entrada. A continuación, se aplica la función "Flatten" para convertir el tensor de entrada en un tensor en una dimensión (en este caso ya es de una dimensión).

Finalmente, se aplica la transformación lineal mediante la función "Linear", que aplica a la entrada la transformación $y = xA^T + b$, donde b representa el bias, y es un parámetro que debe aprender la red.

2.3 Resumen

En primer lugar, hemos comenzado describiendo el estado del arte del problema de síntesis de voz a partir de bioseñales que se aborda en este trabajo. Además, hemos descrito los parámetros acústicos extraídos de la voz así como los parámetros extraídos de las bioseñales. Finalmente, se han dedicado las siguientes secciones para explicar los fundamentos de tres de los cuatro modelos de regresión utilizados en este trabajo. Estos son: regresión lineal, redes neuronales profundas y redes neuronales residuales.

3 Ecuaciones Diferenciales Ordinarias Neuronales

En este capítulo se describe la principal aportación de este trabajo: la aplicación de las ecuaciones diferenciales ordinarias neuronales (NODE, del inglés *Neural Ordinary Differential Equations*) para la resolución de problemas de aprendizaje automático. Las NODE fueron propuestas en 2018 por Tian Qi Chen, Yulia Rubanova, Jesse Bettencourt y David Duvenaud en [2], siendo este trabajo premiado como el mejor artículo en el prestigioso congreso NeurIPS.

En concreto, en este trabajo se aplican estos nuevos modelos matemáticos a la síntesis de voz a partir de registros del movimiento de los órganos del habla obtenidos usando la técnica PMA descrita en el apartado 2.1.2. Las NODE nacen como alternativa a la estrategia de aproximación *directa* descrita por el modelo de regresión visto en la sección 2.2.1.

3.1 Optimización como un problema de ODE

Como alternativa a la estrategia de aproximación *directa* descrita por el modelo de regresión visto en la sección 2.2.1, es posible hallar la función f de forma indirecta a través de su derivada, esto es, diferenciando la relación de regresión (3.1)

$$\frac{dy}{dx} = f'(x). \tag{3.1}$$

Esta es la forma básica de una ecuación diferencial ordinaria (EDO), que es equivalente a resolver (salvo constantes) la integral $f(x) = \int f'(x)dx$.

No obstante, en general, la derivada no solo depende de x, sino también de y. De aquí en adelante utilizaremos g para denotar esa función de varias variables y escribiremos ((3.1)) genéricamente como

$$y'(x) = g(x, y), \quad y(x_0) = y_0, \tag{3.2}$$

y el objetivo será encontrar la función g que describe el ritmo de cambio en función de la variación en x. Esta nueva perspectiva de resolución del problema, como veremos más adelante, reduce el número de parámetros del modelo y el número de evaluaciones de la función requeridas para encontrar los parámetros óptimos.

La mayoría de los problemas descritos por EDOs no se pueden resolver analíticamente, sino que necesitan métodos numéricos que no resuelven de forma analítica la integral, sino que devuelven la función evaluada en un conjunto de puntos futuros. Entonces, únicamente necesitamos un punto inicial, (x_0, y_0), y un número de puntos adicionales de nuestro conjunto de datos para ajustar la aproximación.

A modo de ejemplo, desarrollamos a continuación este enfoque para el modelo lineal descrito en la sección 2.2.1, y que es, en cierto modo, una alternativa *indirecta* a la técnica de regresión por mínimos cuadrados presentada allí. En ese caso, asumíamos

$$y \approx g(x) = ax + b. \tag{3.3}$$

En (3.3) hay dos parámetros libres, a y b. El término *libre* se refiere a que estos parámetros pueden variar durante el proceso de minimización de la función de pérdida. Son los parámetros que nos interesa optimizar. Si utilizamos una EDO como alternativa, sabemos que su forma paramétrica es la siguiente

$$\frac{d\hat{y}}{dx} = a \tag{3.4}$$

y estamos tratando de aproximar

$$\frac{dy}{dx} \approx g(x, y) = a. \tag{3.5}$$

donde ahora únicamente hay un parámetro libre, a.

La solución de ((3.5)) puede ser abordada mediante varias técnicas de integración de EDOs. E particular, por simplicidad, vamos a asumir que utilizamos el conocido método de Euler descrito brevemente en el apéndice 6. Sin embargo, para utilizar este método, necesitamos definir $g(x, y)$ de forma explícita, lo que requiere el conocimiento

previo del parámetro a. Asumamos, por tanto que asignamos un valor inicial para a y empleemos el método de Euler para evolucionar nuestros datos (fase que llamaremos *forward pass*). El flujo es el siguiente:

1. Se elige un valor inicial para a y un tamaño de paso fijo δ.
2. Se elige un punto inicial (x_0, y_0) y k puntos de evaluación.
3. Se añaden puntos adicionales de X para utilizar el método de Euler con intervalos regulares.
4. Se itera a lo largo de la trayectoria de cálculo, evaluando también en los puntos elegidos del conjunto de datos.

Finalmente, para calcular la función de pérdida, se compara con los valores reales, los valores obtenidos mediante Euler, evaluado en los k puntos elegidos. En la fase de *backward pass*, se calcula la derivada de la función de pérdida con respecto al parámetro a, para cada punto de evaluación, y se ajusta tal y como se haría con el método de descenso en gradiente.

En conclusión, utilizando el método de Euler en el contexto de un problema de optimización en aprendizaje automático, tratamos a como un parámetro libre (el único en nuestro caso), que puede ser utilizado para describir una relación lineal que típicamente requiere dos parámetros.

3.2 Generalización: ODEs neuronales

Apliquemos este procedimiento ahora una red neuronal donde los estados ocultos (*hidden states*) tienen todos el mismo número de parámetros. Cada estado oculto depende de una capa neuronal (*neural layer*), g, que a su vez depende de parámetros libres θ_t, donde t es el número de orden de la capa dentro de las capas ocultas de la red. Entonces,

$$h_{t+1} = g(h_t, \theta_t). \tag{3.6}$$

En una red residual, el estado oculto tiene la siguiente forma:

$$h_{t+1} = h_t + g(h_t, \theta_t), \tag{3.7}$$

donde $t \in \{0, ..., T\}$ y $h_t \in \mathbb{R}^D$.

La red neuronal residual tiene funciones de transformación diferentes $g(h(t), \theta_t)$ entre cada capa, por lo que almacena los pesos en cada una de estas. Al aumentar la profundidad de la red, los recursos necesarios para mantener los pesos empiezan a ser cada vez más costosos. Sin embargo, en NODE, todas las transformaciones desde la entrada de datos hasta la salida son continuas, por lo que la trayectoria $h(t)$ es una curva suave (ver Figura 3.1). Por lo tanto, la red puede ser parametrizada mediante una ecuación diferencial ordinaria. Una NODE puede ser vista como un tipo especial de red neuronal residual cuya estructura es una secuencia de bloques residuales, cada uno de los cuales comparte la misma función de transformación g.

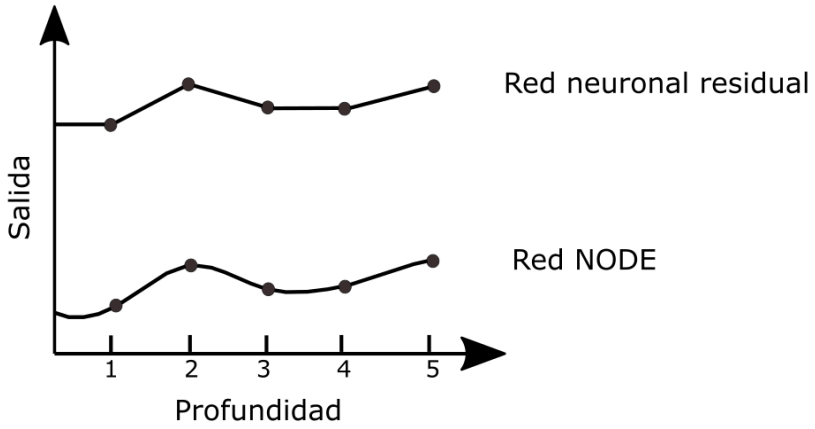

Figura 3.1: Trayectoria del estado oculto $h(t)$ en una red residual y en una NODE

Estas actualizaciones iterativas pueden interpretarse como la discretización de una transformación continua utilizando el método de Euler, lo que sirvió de inspiración a los autores que propusieron originalmente las NODE en [2].

El modelo de las redes neuronales residuales siguen el patrón de una EDO, debido a que la relación continua se modela al nivel de la derivada. A medida que se añaden más capas a la red y se disminuye el paso temporal. Tomando el límite se puede parametrizar la dinámica continua de las unidades ocultas empleando una ecuación diferencial ordinaria, definida por una red neuronal:

$$\frac{dh(t)}{dt} = g(h(t), t, \theta), \tag{3.8}$$

donde $h(t)$ es el valor del estado oculto evaluado para un valor de t, el cual podemos entender como una parametrización continua de la profundidad de la neurona.

3.2.1 Dinámica continua de los estados ocultos.

En el método de Euler, definimos una trayectoria de cálculo partiendo de un punto inicial (t_0, y_0), y evaluando recursivamente la EDO en tamaño de paso fijos. El tamaño fijo de los pasos representa una escala de tiempo en la evolución de la EDO al resolverla numéricamente. En una red neuronal artificial el procesamiento entre las capas de entrada y de salida se lleva a cabo en una serie de capas ocultas discretas. Esta metodología es similar a la trayectoria computacional para el método de Euler. La diferencia se encuentra en que ahora el tamaño de pasos fijos se corresponde con las capas ocultas de la red neuronal, lo cual define una dinámica del estado oculto con respecto a la profundidad de la capa. Esta dinámica puede ser visualizada como una evolución discreta del estado oculto, evaluada en cada capa. En una NODE, esta dinámica del estado oculto se parametriza de la siguiente forma:

$$\frac{dh(t)}{dt} = g(t, h(t), \theta_t), \tag{3.9}$$

donde $g(t, h(t), \theta_t)$ representa una red neuronal parametrizada por θ_t en la capa t-ésima. De esta forma, podemos evaluar el valor del estado oculto en cualquier valor de profundidad resolviendo la integral

$$h(t) = \int g(t, h(t), \theta_t)dt \tag{3.10}$$

Esta integral la podemos resolver utilizando cualquier método de integración numérica de EDO (y/o sistemas de EDOs), que a partir de ahora notaremos genéricamente como ODESolve$(\cdot)$. La suposición final consiste en considerar la entrada como $h(t_0) = x$ (el valor inicial de la EDO) y la salida la evaluamos en un instante t_1, tal que $h(t_1) = y$. El problema de aproximación de la función ahora tiene lugar sobre una dinámica continua de estado oculto.

Debido a que estamos trabajando con un algoritmo de aprendizaje automático, podemos tratar a los parámetros t_0 y t_1 como parámetros libres que deben ser optimizados. Uniendo todos los parámetros, podemos definir las predicciones de la salida y como sigue,

$$\widehat{y} = h(t_1) = \text{ODESolve}(h(t_0), t_0, t_1, \theta, f), \tag{3.11}$$

donde f define a una red neuronal. El operador ODESolve permite definir una función que representa la solución a un sistema de ecuaciones diferenciales ordinarias, sujeto a los valores iniciales (t_0, t_1).

3.2.2 *Backpropagation en NODEs*

La principal dificultad técnica al entrenar redes de profundidad continua se encuentra en aplicar el algoritmo de *backpropagation* a lo largo del sistema que resuelve la EDO. Este sistema es tratado como una "caja negra", y calcula los gradientes aplicando el método de *adjoint sensitivity* [28]. Esta aproximación calcula los gradientes resolviendo un modelo de EDO extendido hacia atrás en el tiempo y es aplicable a todos los métodos para resolver EDOs.

La innovación principal del trabajo [2] es precisamente el algoritmo de *backpropagation* a lo largo de la dinámica continua de los estados ocultos. En la red neuronal de tipo EDO, nuestros parámetros libres no son solamente θ_t, sino también los instantes de evaluación t_0 y t_1. Podemos definir cualquier función de pérdidas (diferenciable) en función de estos parámetros libres:

$$\mathcal{L}(t_0, t_1, \theta_t) = \mathcal{L}(\text{ODESolve}(h(t_0), t_0 t_1, \theta, f)). \tag{3.12}$$

Para optimizar la función de pérdidas, necesitamos calcular los gradientes respecto a los parámetros libres. Al igual que con el algoritmo de *backpropagation* en el área de *deep learning*, el primer paso consiste en calcular el gradiente de la función de pérdidas con respecto a los estados ocultos:

$$\frac{\partial \mathcal{L}}{\partial h(t)} \tag{3.13}$$

Sin embargo, el estado oculto depende a su vez del tiempo (profundidad de la capa), por lo que podemos calcular la derivada con respecto al tiempo. Para calcular estos gradientes, se utiliza el **método adjunto** (*adjoint method*), el cual es capaz de calcular gradientes de manera eficiente. Una explicación detallada se encuentra en el Apéndice B de [2].

Para mantener un seguimiento de la dinámica temporal, definimos el llamado **estado adjunto**

$$a(t) = -\frac{\partial \mathbf{L}}{\partial h(t)} \tag{3.14}$$

Anteriormente, hemos definido la derivada temporal del estado oculto como

$$\frac{dh(t)}{dt} = g(t, h(t), \theta_t), \tag{3.15}$$

donde θ_t son parámetros. Es fácil demostrar que si definimos el estado adjunto dado por (3.14), entonces su derivada temporal viene dada por la fórmula

$$\frac{da(t)}{dt} = -a(t)^T \frac{\partial g(t, h(t), \theta_t)}{\partial h(t)}. \tag{3.16}$$

El estado adjunto representa el gradiente con respecto al estado oculto en un instante de tiempo t. En las redes neuronales comunes, el gradiente de una capa oculta h_t depende del gradiente de la siguiente capa h_{t+1}. Aplicando la regla de la cadena

$$\frac{d\mathcal{L}}{dh_t} = \frac{d\mathcal{L}}{dh_{t+1}} \frac{dh_{t+1}}{dh_t}. \tag{3.17}$$

En un estado oculto continuo, podemos escribir la transformación tras un cambio ϵ en el tiempo como

$$h(t+\epsilon) = \int_t^{t+\epsilon} g(h(t), t, \theta)dt + h(t) = T_\epsilon(h(t), t) \tag{3.18}$$

Aplicamos de nuevo la regla de la cadena

$$\frac{d\mathcal{L}}{dh(t)} = \frac{d\mathcal{L}}{dh(t+\epsilon)} \frac{dh(t+\epsilon)}{dh(t)} \tag{3.19}$$

o, equivalentemente,

$$a(t) = a(t+\epsilon) \frac{\partial T_\epsilon(h(t), t)}{\partial h(t)} \tag{3.20}$$

Para demostrar (3.16), utilizamos la definición de la derivada:

$$\frac{da(t)}{dt} = \lim_{\epsilon \to 0^+} \frac{a(t+\epsilon) - a(t)}{\epsilon} \tag{3.21}$$

$$= \lim_{\epsilon \to 0^+} \frac{a(t+\epsilon) - a(t+\epsilon)\dfrac{\partial T_\epsilon(h(t), t)}{\partial h(t)}}{\epsilon} \quad (según\ 3.20) \tag{3.22}$$

$$= \lim_{\epsilon \to 0^+} \frac{a(t+\epsilon) - a(t+\epsilon)\dfrac{\partial}{\partial h(t)}(h(t) + \epsilon g(h(t), t, \theta) + O(\epsilon^2))}{\epsilon} \quad (Serie\ de\ Taylor\ en\ h(t)) \tag{3.23}$$

$$= \lim_{\epsilon \to 0^+} \frac{a(t+\epsilon) - a(t+\epsilon)\left(I + \epsilon \dfrac{\partial g(h(t), t, \theta)}{\partial h(t)} + O(\epsilon^2)\right)}{\epsilon} \tag{3.24}$$

$$= \lim_{\epsilon \to 0^+} \frac{-\epsilon a(t+\epsilon)\dfrac{\partial g(h(t), t, \theta)}{\partial h(t)} + O(\epsilon^2)}{\epsilon} \tag{3.25}$$

$$= \lim_{\epsilon \to 0^+} -a(t+\epsilon)\frac{\partial g(h(t), t, \theta)}{\partial h(t)} + O(\epsilon) \tag{3.26}$$

$$= -a(t)\frac{\partial g(h(t), t, \theta)}{\partial h(t)} \tag{3.27}$$

Esta derivada se puede calcular debido a que la función de pérdidas, $\mathcal{L}$, y g son ambas diferenciables. Además, esta derivada también es una EDO, por lo que podemos escribir la solución del estado adjunto como la integral:

$$a(t) = \int -a(t)\frac{\partial g(h(t), t, \theta_t)}{\partial h(t)} dt, \tag{3.28}$$

o, equivalentemente,

$$\frac{\partial \mathcal{L}}{\partial h(t)} = \int a(t)\frac{\partial g(h(t), t, \theta_t)}{\partial h(t)} dt, \tag{3.29}$$

Hemos señalado la similitud entre el método de *backpropagation* y el método adjunto (3.20). De forma similar a *backpropagation*, la EDO para el estado adjunto se resuelve "hacia atrás" en el tiempo. Por lo tanto, para calcular el gradiente en t_0, solucionamos la EDO inversamente en el tiempo desde el punto inicial (conocido) en el tiempo t_1

$$a(t_0) = \int_{t_1}^{t_0} -a(t)\frac{\partial g(h(t), t, \theta_t)}{\partial h(t)} dt. \tag{3.30}$$

La Figura 3.2 muestra el funcionamiento del método adjunto.

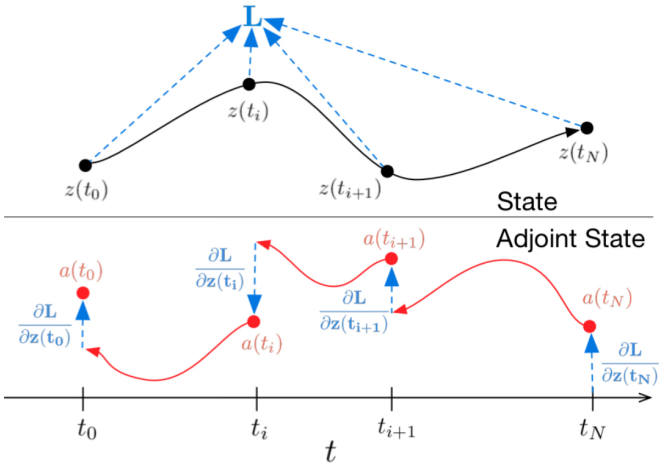

Figura 3.2: Método adjunto. Fuente: [2]

El adjunto captura información relacionada con el cambio en la función de pérdidas $\mathcal{L}$ con respecto al estado oculto definido en la ecuación (3.14). Tomando como punto de partida la salida de la red, podemos recalcular el estado oculto (en la Figura 3.2 denotado como $z(t)$), hacia atrás en el tiempo, a la vez que se calcula el adjunto en cada instante.

Con este método ya tenemos una forma de calcular el gradiente con respecto a t_1 y t_0. Para calcular los gradientes respecto al parámetro θ, resolvemos la EDO:

$$\frac{\partial \mathcal{L}}{\partial \theta} = \int_{t_1}^{t_0} a(t) \frac{\partial g(h(t), t, \theta_t)}{\partial \theta} dt. \tag{3.31}$$

Una vez más, esta integral puede ser resuelta utilizando cualquier método para resolver EDOs.

3.3 Resumen

En este capítulo hemos comenzado formulando el problema de modelado de datos utilizando EDOs. A continuación, hemos extendido este modelo a uno más complejo que incluye una red neuronal (NODE), tal como explica [2]. Finalmente, hemos explicado el algoritmo mediante el cual los parámetros de la red neuronal pueden ser

optimizados mediante el algoritmo de *backpropagation* a lo largo de la EDO haciendo uso del método adjunto.

Los siguientes pasos resumen el procedimiento por el cual se resuelve un problema de modelado de datos mediante una red neuronal de tipo NODE.

1. Tenemos un conjunto de N pares de puntos, $\{(x_1, y_1), ..., (x_N, y_N)\}$. Dado un nuevo punto, x^*, queremos predecir su valor y^*. El objetivo es encontrar una aproximación a la relación entre los dominios de entrada y salida.
2. En lugar de modelar esta relación directamente, modelamos la derivada:

$$\frac{dy}{dx} = g(x, y). \tag{3.32}$$

3. Parametrizamos esta aproximación con una red neuronal con estados ocultos $h(t)$, que dependen de forma continua de la profundidad de la capa, t, con $h(t_0) = x$ y $h(t_1) = y$. El problema de aproximación de la función es ahora el siguiente:

$$\frac{dh(t)}{dt} = g(t, h(t), \theta), \tag{3.33}$$

   donde f es una red neuronal.
4. Esta aproximación consiste en una EDO que describe la dinámica continua del estado oculto. Por lo tanto, el problema de modelado de datos se puede resolver hallando la solución a esta EDO, es decir, resolviendo la integral

$$h(t) = \int g(t, h(t), \theta)dt. \tag{3.34}$$

   Para obtener las predicciones, resolvemos la integral definida:

$$\widehat{y} = h(t_1) = \int_{t_0}^{t_1} g(t, h(t), \theta)dt. \tag{3.35}$$

   No conocemos la solución analítica de esta integral, por lo que utilizamos un método numérico para resolverla en los puntos requeridos de evaluación:

$$\widehat{y} = h(t_1) = \text{ODESolve}(h(t_0), t_0, t_1, \theta, f) \tag{3.36}$$

5. Los parámetros libres del modelos son t_0, t_1 y θ. Estos se optimizan aplicando *backpropagation* utilizando el método adjunto.

En conclusión, utilizando una EDO, se requiere únicamente un punto inicial para empezar el proceso. Se llevan a cabo N evaluaciones de la función en cada punto en la etapa de *forward pass*. Entonces, solo es necesario evaluar la función de pérdida en un conjunto de datos, antes de decidir cómo actualizar los parámetros libres. Finalmente, utilizar una EDO requiere un número menor de evaluaciones, particularmente en *backward pass*.

A modo de epílogo, en este capítulo hemos introducido las ecuaciones diferenciales ordinarias neuronales (NODE) de forma explícita, aprovechando, *grosso modo*, la potencia de las redes neuronales como aproximadores universales para de la función g.

4 Evaluación

En este capítulo se presentan los resultados de la evaluación experimental de las técnicas de aprendizaje automático descritas en los capítulos anteriores. Para ello se llevarán a cabo diversos experimentos en los que la tarea será sintetizar voz a partir de datos de captura de movimiento de los labios y la lengua de distintos sujetos capturados con la técnica PMA, tal y como vimos en la sección 2.1.2. Para evaluar el rendimiento de las técnicas, se usarán tanto medidas objetivas de calidad de la voz ampliamente usadas en la literatura, como medidas subjetivas obtenidas mediante pruebas de escucha realizadas por sujetos humanos.

Este capítulo se organiza de la siguiente forma. En primer lugar, en la sección 4.1, se describe el marco experimental usado para llevar a cabo la evaluación de las técnicas propuestas, incluyendo las bases de datos empleadas, el tipo de parametrización usada para cada tipo de señal y, por último, detalles de implementación de los modelos de aprendizaje automáticos usados. Seguidamente, en la sección 4.2, se exponen los resultados obtenidos por las distintas técnicas en la tarea de síntesis de voz expuesta antes.

4.1 Marco experimental

4.1.1 Base de datos

Para evaluar las técnicas de aprendizaje automático descritas anteriormente, se ha utilizado una base de datos con grabaciones simultáneas de voz y datos PMA realizadas a cinco sujetos sanos (cuatro hombres y una mujer) mientras estos leían en voz alta secuencias de dígitos en inglés, tal y como se describe en [29]. En concreto, las señales de audio y de PMA de 9 canales se grabaron simultáneamente a frecuencias

de muestreo de 48 kHz y 100 Hz, respectivamente, utilizando un micrófono de condensador AKG C1000S y un dispositivo de PMA como el mostrado en la figura **??**. A continuación, a las señales de PMA se les aplicó una técnica de supresión de ruido de fondo para compensar el efecto del campo magnético de la Tierra en los datos capturados. Por último, los datos se segmentaron usando una técnica de detección de voz basada en la energía de la señal. La tabla 4.1 muestra los detalles de la base de datos usada en este trabajo.

Sujeto	Nº frases grabadas	Nº frases entrenamiento	Nº frases test
F1	308 (8.5 min)	233	75
M1	308 (8.0 min)	233	75
M2	308 (9.7 min)	233	75
M3	308 (7.4 min)	233	75
M4	308 (7.2 min)	233	75

Tabla 4.1: Detalles del conjunto de datos usado en los experimentos. Para cada sujeto se muestra la siguiente información: (a) el número de secuencias de dígitos grabadas por el sujeto incluyendo la duración total de todas sus grabaciones, (b) el nº de frases usadas para entrenar los modelos y (c) el número de frases usadas en evaluación.

Para el entrenamiento de los modelos implementados, se utilizó un conjunto separado previamente de entrenamiento y test, de un total de 308 grabaciones realizadas por cada uno de los 5 sujetos, tal y como se muestra en la tabla 4.1. Para cada sujeto se seleccionaron aleatoriamente 233 grabaciones para entrenar los modelos, mientras que las 75 frases restantes se usaron como conjunto de test durante la evaluación.

4.1.2 Extracción de parámetros

Las señales extraídas de los datos PMA y las grabaciones de voz han sido parametrizadas como una serie de vectores de parámetros calculados cada 5 ms a partir de ventanas de análisis de 25 ms. En concreto, las señales de voz fueron primero submuestreadas desde 48 kHz hasta 16 kHz, para ser posteriormente transformadas en vectores de 32 parámetros utilizando el *vocoder* WORLD, tal y como se describió en la sección 2.1.3. Estos parámetros son: 25 MFCCs, 1 valor de aperiodicidad, 1 valor para la frecuencia fundamental F_0 en escala logarítmica, y 1 valor para las decisiones sordo/sonoro de cada trama (*voicing*).

Por otra parte, los nueve canales correspondientes a las señales PMA fueron sobremuestreadas desde 100 Hz hasta 200 Hz para coincidir con los 5 ms de periodo

de trama.

A todas las matrices de parámetros se les aplicó la técnica de reducción de dimensión Principal Component Analysis (PCA) [30]. Se trata de un método estadístico que permite simplificar la complejidad de espacios muestrales con un número elevado de dimensiones, a la vez que conserva su información. PCA permite encontrar un número de factores subyacentes que explican aproximadamente lo mismo que las variables originales. Cada una de estas nuevas variables recibe el nombre de *componente principal*. En este caso, hemos seleccionado el número de componentes que explica el 99 % de varianza de los parámetros extraídos. Antes de proceder al entrenamiento de los modelos, tanto los parámetros acústicos como los extraídos de las bioseñales se normalizan en media y varianza, para que todos los datos de entrada tengan el mismo orden de magnitud, evitando que se produzcan predicciones erróneas.

4.1.3 Modelos de regresión utilizados

Una vez extraídas las matrices de parámetros correspondientes a la información de los sensores PMA y a las grabaciones de voz, el siguiente paso consiste en entrenar los modelos de aprendizaje automático para que sean capaces de estimar los parámetros de la voz a partir de parámetros extraídos de las señales PMA. Los modelos implementados en este trabajo son los descritos en las secciones 2.2 y 3.2, a saber: regresión lineal y logística, red neuronal profunda (DNN), red neuronal residual (ResNet) y red neuronal NODE. Para hacer la comparativa más justa, todos los modelos de redes neuronales han sido entrenados con, aproximadamente, el mismo número de parámetros, en torno a $\tilde{1}45000$ parámetros. En los siguientes apartados se describen los detalles de implementación de cada uno de los modelos.

Además, se entrenó un modelo distinto con función de pérdida el Mean Square Error (MSE) para cada conjunto de parámetros extraídos de las grabaciones: MFCC, aperiodicidad y logaritmo de la frecuencia fundamental. Sin embargo, para el parámetro de tipo binario *voicing*, debido a que la salida únicamente puede ser 0 o 1, se entrenó un modelo de regresión con función de pérdida *binary cross entropy*. Para todas las arquitecturas de redes neuronales, se ha utilizado como función de activación la función Sigmoide, ya que comprime la salida a un rango entre 0 y 1.

Regresión lineal y logística

En este caso, se entrenó un modelo distinto de regresión lineal para cada conjunto de parámetros extraídos de las grabaciones, y un modelo de regresión logística (salida entre 0 y 1) para el parámetro de tipo binario *voicing*.

Red neuronal profunda (DNN)

El modelo de DNN implementado consta de 3 capas ocultas, con 70, 350 y 105 unidades ReLU por capa. Los pesos de la red fueron optimizados utilizando el algoritmo de descenso en gradiente estocástico (SGD) con tasa de aprendizaje $3e-3$ y tamaño de *mini-batch* igual a 512 muestras. A la hora de entrenar la DNN en el caso de los parámetros continuos se usó el MSE, descrito en (**??**).

Para el parámetro *voicing*, se usó la función de entropía binaria cruzada (*binary cross entropy*), utilizada comúnmente para problemas binarios de clasificación.

Red neuronal residual (ResNET)

Para entrenar la red neuronal residual se ha utilizado una tasa de aprendizaje igual a $3 \cdot 10^{-3}$ y tamaño de *batch* igual a 512. La Tabla 4.3 muestra un resumen de la arquitectura utilizada, para un número de bloques residuales de 4 y los parámetros MFCC (25) junto con sus derivadas (50 en total)

El tipo de capa y su función se ha explicado previamente en la sección 2.2.5.

Ecuaciones diferenciales ordinarias neuronales (NODE)

Para una red neuronal continua como este caso, adoptamos los mismos métodos utilizados para entrenar una red neuronal, manteniendo las características de la EDO.

El tamaño del *mini-batch* ha mantenido en 512, con una dimensión de cada bloque de entrada igual a 64.

Las primeras capas de la red son iguales a las capas de la red neuronal residual, ya que su función es aplicar una serie de transformaciones para reducir la dimensión de los datos de entrada.

Tipo de capa	**Tamaño de salida**	Parámetros
Conv1d-1	64 x 3 x 510	256
GroupNorm-2	64 x 3 x 510	128
ReLU-3	64 x 3 x 510	0
Conv1d-4	64 x 2 x 254	20544
GroupNorm-5	64 x 2 x 254	128
ReLU-6	64 x 2 x 254	0
Conv1d-7	64 x 2 x 126	20544
ResBlock-8	64 x 1 x 126	24832
ResBlock-9	64 x 1 x 126	24832
ResBlock-10	64 x 1 x 126	24832
ResBlock-11	64 x 1 x 126	24832
GroupNorm-12	64 x 1 x 126	128
ReLU-13	64 x 1 x 126	0
AdaptiveAvgPool-14	64 x 126 x 1	0
Flatten	64 x 1	0
Linear	50 x 1	3250
Parámetros totales:	-	144306

Tabla 4.2: Arquitectura utilizada para la red neuronal residual

En lugar del conjunto de bloques residuales, definimos una red neuronal con una serie de capas secuenciales, en las que el orden es igual al definido para el bloque residual.

Una vez definida esta red, definimos los estados ocultos que dependen de forma continua de la profundidad de la capa (ver Ecuación 3.33). Para obtener la expresión de los estados ocultos, hallamos la solución a esta EDO, resolviendo la integral 3.34 con la función *odeint* de la librería Scipy de Python. Donde la función g es la red neuronal de seis capas que hemos definido anteriormente. Los valores iniciales son los datos de entrada a la red, y los puntos de evaluación de $h(t)$ son $[0, 1]$. Entonces, como se explicó en el capítulo 2, para obtener las predicciones, $h(t_1)$, aplicamos un ODESolve sobre h_{t_0}.

Al igual que para la red neuronal residual, para entrenar la red NODE se ha utilizado una tasa de aprendizaje igual a $3 \cdot 10^{-3}$. La Tabla **??** muestra un resumen de la arquitectura utilizada, para un número de bloques igual a 4 y los parámetros MFCC (25) junto con sus derivadas (50 en total):

Para implementar el algoritmo de *backpropagation*, se utiliza el método adjunto, en el que se calculan los gradientes respecto a los parámetros libres del modelo hacia

Tipo de capa	**Tamaño de salida**	Parámetros
Conv1d-1	64 x 3 x 510	256
GroupNorm-2	64 x 3 x 510	128
ReLU-3	64 x 3 x 510	0
Conv1d-4	64 x 2 x 254	20544
GroupNorm-5	64 x 2 x 254	128
ReLU-6	64 x 2 x 254	0
Conv1d-7	64 x 2 x 126	20544
ODENet-8	64 x 1 x 126	25472
ODENet-9	64 x 1 x 126	25472
ODENet-10	64 x 1 x 126	25472
ODENet-11	64 x 1 x 126	25472
GroupNorm-12	64 x 1 x 126	128
ReLU-13	64 x 1 x 126	0
AdaptiveAvgPool-14	64 x 126 x 1	0
Flatten	64 x 1	0
Linear	50 x 1	3250
Parámetros totales:	-	146866

Tabla 4.3: Arquitectura utilizada para la red neuronal residual

atrás en la red. La diferencia con el método de *backpropagation* utilizado en ResNet se encuentra en la forma de calcular estos gradientes, ya que se resuelve la EDO 3.31.

Además, se ha utilizado el método de integración explícita de Runge-Kutta *dopri5* (Dormand-Prince 4/5), que une un método de orden 4 y de orden 5 [31]. Aunque con este método se obtienen resultados similares (y en algunos casos mejores) que utilizando la red ResNet, es del orden de 10 veces más lento en la etapa de entrenamiento que ResNet.

4.1.4 Evaluación de la calidad de la voz

Métricas objetivas

Para evaluar la calidad de las señales de audio sintetizadas por los modelos anteriores a partir de los datos PMA, calculamos un conjunto de métricas objetivas ampliamente utilizadas en síntesis de voz. Estas son: distorsión de los MFCC (Mel-Cepstral Distortion (MCD)) medida en dB, la raíz cuadrada del error cuadrático medio (RMSE) de la matriz de aperiodicidades (Band Aperiodicity (BAP)) predicha, así como de los

valores de F_0, y el porcentaje de error para las predicciones del parámetro de (*voicing*). Estas medidas se describen en mayor detalle a continuación.

- **Distorsión de los MFCCs (en dB):** Medida utilizada para cuantificar la calidad de la voz sintetizada. Mide la diferencia entre la secuencia de MFCCs predichos a partir de los datos PMA y los MFCCs reales extraídos de las grabaciones originales realizadas por los sujetos. Se calcula de la siguiente forma, según [32]:

$$\text{MCD} = \frac{10\sqrt{2}}{\ln 10}\frac{1}{M}\sum_{m=1}^{M}\sqrt{\sum_{i}(c_{mi}-\hat{c}_{mi})^2}, \quad \textit{(dB)} \tag{4.1}$$

  donde c y $\hat{c}$ son los MFCCs predichos y reales, respectivamente, y $M = 25$ es la dimensión del vector de parámetros con los MFCCs.
- **RMSE (dB):** La raíz del error cuadrático medio es una medida de uso frecuente de las diferencias entre los valores predichos por el modelo y los valores reales. Esta medida se utiliza para calcular la diferencia entre las aperiodicidades predichas y las reales. Se calcula del siguiente modo:

$$\text{RMSE} = 10 \cdot \log_{10}\sqrt{\frac{\sum_{m=1}^{M}(\widehat{ap}_m - a_m)^2}{M}}, \quad \textit{(dB)} \tag{4.2}$$

  donde $\widehat{ap}$ y ap son las aperiodicidades predichas y reales, respectivamente.
  En el caso del logaritmo de F_0, se ha utilizado la misma métrica de error. Sin embargo, las unidades son Hz en lugar de dB.
- **Porcentaje de error (%) :** Para el parámetro *voicing*, se ha calculado el porcentaje de error como 1 menos el número de predicciones correctas dividido entre el número total de predicciones, es decir,

$$Voicing_{err} = 1 - \frac{TP}{TP+FP}\,\%, \tag{4.3}$$

  donde FP indica los falsos positivos, es decir, los casos en los que se ha predicho un 1 cuando el valor correspondiente era el 0. TP son los verdaderos positivos: los casos en los que el modelo ha predicho un 1 cuando la clase correspondiente es un 1. Hemos calculado esta métrica con el parámetro *voicing* ya que se trata de un problema de clasificación, en el que las posibles predicciones son binarias.

Métricas subjetivas

Además de las métricas objetivas, se llevó a cabo un test subjetivo de tipo ABX para evaluar de forma subjetiva la calidad de las señales de voz sintetizadas por los modelos anteriores (Regresión Lineal, DNN, ResNet y NODE) a partir de las señales PMA. En este test participaron 10 personas, cada una de las cuales evaluó 8 parejas de audios para cada una de las 6 posibles combinaciones entre modelos, es decir, 48 pares de audios evaluados en total por cada participante. En cada uno de los 48 ensayos a los participantes se les indicó que debían seleccionar cuál de las dos versiones de un mismo audio (A o B), cada una generada por un modelo distinto de los anteriores, era perceptualmente más parecida al audio de referencia X, que se correspondía con uno de los audios originales grabados por los sujetos de nuestra base de datos. Tanto los ensayos como las respuestas fueran aleatorizadas para evitar sesgos en la selección.

4.2 Resultados

La siguiente sección está dedicada a la exposición de los resultados obtenidos por las distintas técnicas de *machine learning* evaluadas en este trabajo. Además de nuestra técnica propuesta NODE, se comparan los resultados obtenidos por las técnicas de regresión lineal/logística, red neuronal profunda DNN y red residual ResNet.

La Tabla 4.4 muestra los resultados objetivos obtenidos por cada una de las técnicas en el conjunto de evaluación. Los resultados mostrados se han obtenido promediando las métricas para todos los sujetos del conjunto de test. Para cada una de las métricas, por tanto, se muestra la media global obtenida y la desviación típica de las medidas (representado con $\pm$ en la Tabla 4.4).

	MFCC	**BAP**	F_0	***Voicing***
Método	MCD (dB)	RMSE (dB)	RMSE (Hz)	Tasa de error (%)
Regr. lineal	9.02 ± 0.08	8.48 ± 0.72	45.63 ± 0.10	44.17 ± 0.13
DNN	8.62 ± 0.31	$\mathbf{6.92 \pm 0.67}$	$\mathbf{41.35 \pm 0.15}$	$\mathbf{25.96 \pm 0.14}$
ResNet	7.90 ± 0.48	7.84 ± 1.49	41.42 ± 0.10	39.17 ± 0.27
NODE	$\mathbf{7.82 \pm 0.44}$	7.66 ± 1.64	41.39 ± 0.13	41.22 ± 0.31

Tabla 4.4: Resultados objetivos (media $\pm$ desv. típica) obtenidos por los distintos modelos de aprendizaje automático en el conjunto de evaluación.

Como era de esperar, en la tabla se puede observar que para todos los tipos de parámetros, con las arquitecturas de red neuronal (DNN, ResNet y NODE) se obtienen resultados significativamente mejores que utilizando únicamente regresión lineal. Esto se debe, en parte, a la mayor flexibilidad de estos modelos para modelar relacionales no lineales. Por otra parte, respecto a la comparación entre las arquitecturas ResNet y NODE, con esta última se obtienen, en general, mejores métricas de error que utilizando el modelo ResNet, salvo para el parámetro de *voicing*. Cabe destacar que esta disminución del error está relacionada con el método de integración seleccionado. En este caso, el método utilizado Dormand-Prince 4/5 calcula siete pendientes distintas, que posteriormente son combinadas linealmente para encontrar dos aproximaciones del siguiente punto. El error de la primera aproximación es de orden 4, mientras que el error de la segunda aproximación es de orden 5. Los coeficientes de la aproximación de orden 5 son elegidos para minimizar su error.

En la Figura 4.1 se desglosan los resultados obtenidos por las técnicas ResNet y NODE, que son las que mejores resultados han obtenido en la tabla 4.4, para cada uno de los 5 sujetos en el conjunto de evaluación. En esta figura se muestra únicamente la métrica de distorsión de los parámetros MFCCs, ya que son los que más influyen en la inteligibilidad de la voz al representar la envolvemente espectral de la señal. Se puede observar que con NODE se obtiene un error similar y en algunos casos menor que con la red neuronal ResNet. Además, también se puede observar que el menor error para estos coeficientes se obtiene para los sujetos F.1 y M.1, con NODE. Estas diferencias entre sujetos se deben a que los parámetros acústicos así como la forma de articular de cada sujeto es distinta, por lo que los modelos entrenados también lo serán.

En la Figura 4.2 se muestran las curvas de entrenamiento para la red neuronal NODE y ResNet con 32, 64 y 256 capas. Se puede observar cómo, en el caso de la red ResNet, al aumentar el número de capas, el error de validación no disminuye sino que comienza a aumentar ligeramente. Además, a partir de un cierto número de iteraciones (*epochs*), el error comienza converger. Este mismo comportamiento se puede observar en el caso de la red NODE (ver Figura 4.2). Sin embargo, en este caso el error de validación en el modelo comienza a converger en un número mayor de iteraciones. Además, al aumentar el número de capas, se puede observar que para este modelo de red el error de validación es menor que en el caso de ResNet. A partir de estas gráficas se puede concluir que el modelo NODE es capaz de conseguir resultados similares a ResNet con un menor número de iteraciones. Esto se debe a que modela las capas de la red de forma continua.

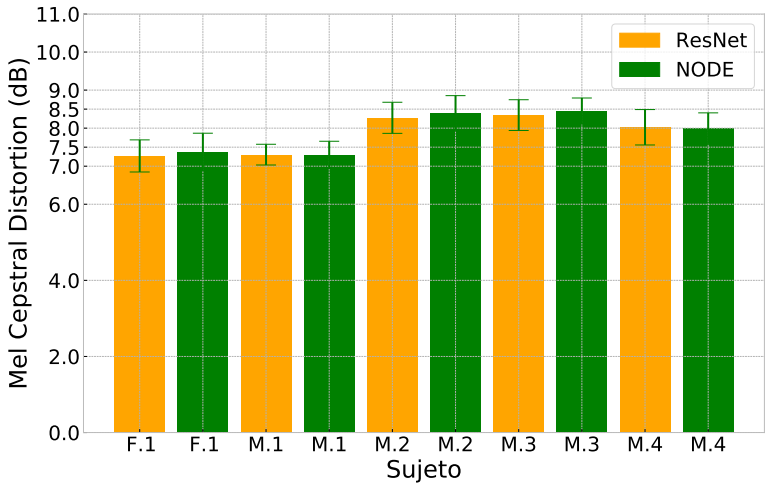

Figura 4.1: Métricas objetivas (MCD) para todos los sujetos de test

4.2.1 Resultados subjetivos

La Figura 4.3 muestra los resultados del test subjetivo ABX sobre calidad perceptual de la voz generada por los distintos modelos. En primer lugar, como era de esperar, el modelo menos elegido es la regresión lineal. Esto se debe a que en este caso se asume que los datos están relacionados de forma lineal. Sin embargo, esto no es cierto para la asociación entre los parámetros acústicos y los datos de tipo articulatorio, por lo que con este modelo simple de regresión se obtienen, tanto objetivamente como subjetivamente, la peor calidad de voz sintetizada.

Por otra parte, la red neuronal ResNet ha sido la más elegida por los participantes del test (85.61 %), por delante de EDO con un 65.91 % y DNN con un 41.29 %. Esto se debe a que el error de predicción para el parámetro *voicing* dado por ResNet es menor que en el caso de NODE, lo que provoca que las señales de voz sintetizadas con este modelo tengan una entonación más adecuada que las sintetizadas con NODE, a pesar de que esta última proporciona mejores predicciones para los parámetros MFCC, aperiodicidad y el logaritmo de la frecuencia fundamental. Respecto al modelo de DNN, ha sido el tercero más elegido, a pesar de proporcionar un error de predicción menor que los modelos de ResNet y NODE para *voicing*, aperiodicidad y frecuencia fundamental. Esto se debe a que la distorsión de los MFCCs predichos es más elevada que la de estos dos modelos, lo que provoca que la voz sintetizada con esta técnica presente una mayor ininteligibilidad.

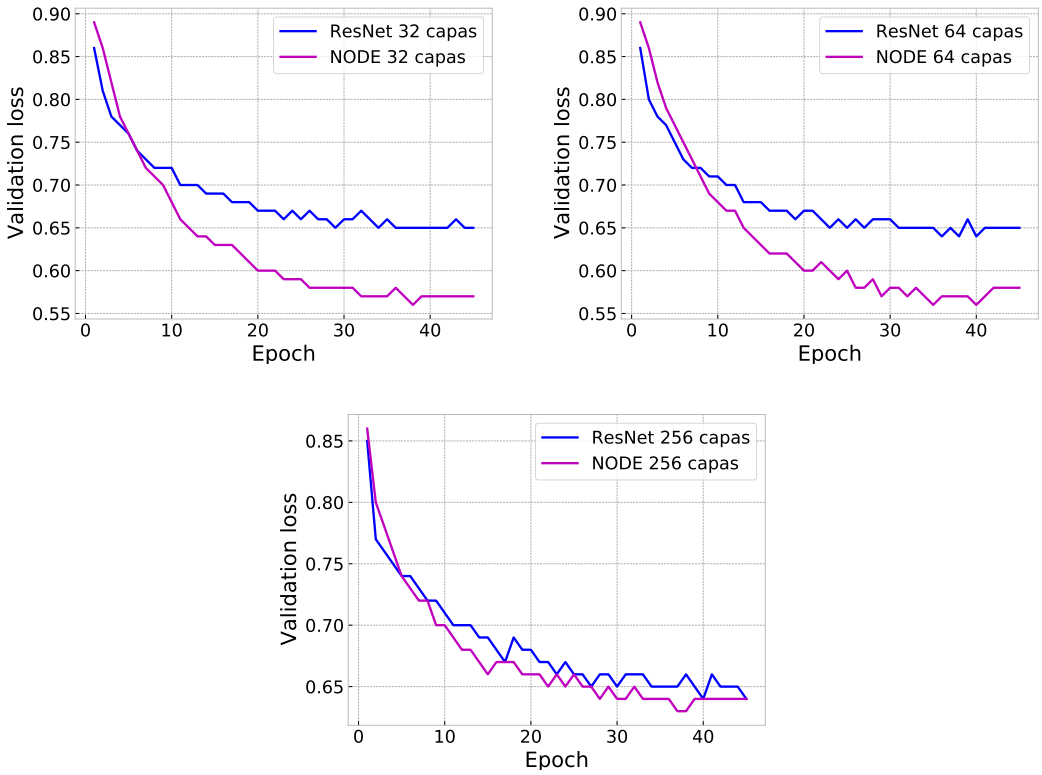

Figura 4.2: Error de validación frente al número de *epochs* para modelos de ResNet y NODE con distinto número de capas.

Finalmente, en la siguiente web `https://github.com/myriam123/TFM-sintesis-voz.git` pueden encontrarse algunos ejemplos de los audios sintetizados por los distintos modelos.

4.2.2 Complejidad computacional

Finalmente, se ha evaluado la complejidad computacional de las arquitecturas de red neuronal que mejores resultados han obtenido, esto es, ResNet y NODE, en un ordenador personal con las siguientes características: 16 GB de memoria RAM, procesador intel Core i7, SSD 512 GB, Sistema Operativo Windows 10 y versión de Python 3.6.

La Tabla 4.5 refleja los recursos computacionales necesarios para ejecutar únicamente la etapa de entrenamiento de los modelos ResNet y NODE, ambas con aproxi-

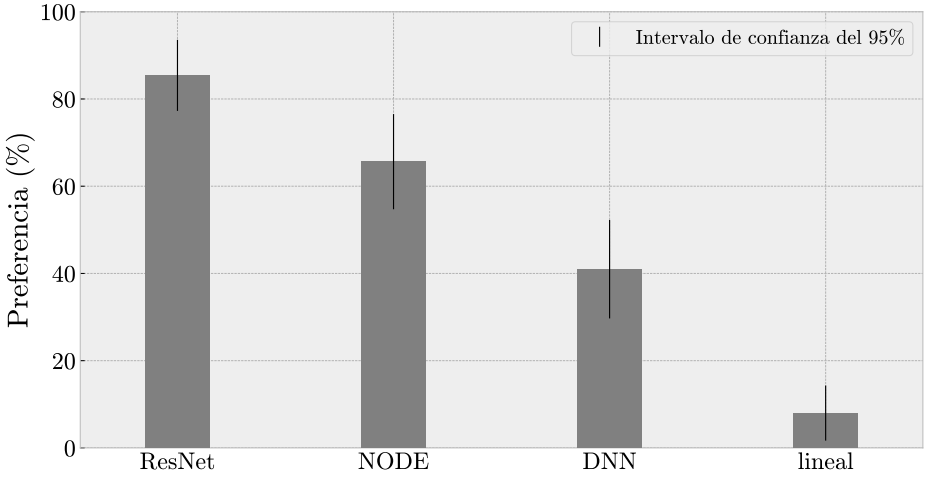

Figura 4.3: Resultados del test subjetivo ABX

madamente el mismo número de parámetros (146866 parámetros para NODE y 144306 para ResNet), para el método de integración *dopri5* y Euler.

En general, el modelo de red NODE con el método adjunto (explicado en el apartado 3.2.2) utiliza menos memoria que ResNet. Al aumentar la profundidad de una red neuronal estándar, los recursos necesarios para mantener los parámetros intermedios empiezan a ser más costosos computacionalmente. Esto se debe a que la red almacena las funciones de transformación aplicadas en cada capa. Sin embargo, en NODE, estas transformaciones son continuas y no es necesario almacenar ningún parámetro intermedio de la red. Esto permite entrenar el modelo con un consumo constante de memoria a medida que aumenta la profundidad de la red.

Cabe destacar que el método utilizado en los experimentos es *dopri5* con un valor de tolerancia de 0.001, por ser un método adaptativo y más estable que otros métodos fijos como Runge-Kutta de orden 4 o Euler. Sin embargo, esto implica un mayor uso de memoria al realizar más cálculos, como aparece reflejado en la tabla 4.5.

Método	**Memoria CPU**
NODE (*dopri5*)	631.30 MB
NODE (Euler)	29.01 MB
ResNet	732.10 MB

Tabla 4.5: Comparación de uso de memoria CPU para NODE con distintos métodos de integración y ResNet.

4.3 Discusión

De los resultados obtenidos en este trabajo, se infiere que las distintas arquitecturas de redes neuronales evaluadas obtienen mejores resultados que un modelo clásico de regresión lineal para la resolución de nuestro problema de síntesis de voz. Entre estos, con el modelo de DNN se obtienen mejores métricas para el valor predicho de BAP, frecuencia fundamental y *voicing*. Sin embargo, en el test subjetivo (Figura 4.3) se puede observar que la DNN sólo fue escogida el 41.29 % de las veces, seguido por el modelo de regresión lineal, con un 7.31 % de elección.

Por otra parte, de las métricas objetivas obtenidas para las redes ResNet y NODE, se puede observar que esta última ofrece unos resultados mejores en todos los tipos de parámetros, excepto en el *voicing*, para el cual se obtiene una tasa de error del 41.22 % frente al 39.17 % de ResNet. Este parámetro, como me explicó en el apartado 4.1.2, únicamente puede tomar dos valores en función de si el segmento de voz es sorda o sonoro. Un error de predicción elevado en este parámetro se refleja en una baja entonación en las señales de voz sintetizadas, como ocurre en este caso. Por el contrario, ResNet es capaz de predecir el parámetro *voicing* con un menor error, lo que provoca que las señales de voz sintetizadas con esta arquitectura de red tengan una entonación más natural y, por lo tanto, hayan sido elegidas un 82.81 % de las veces en el test subjetivo (ver Figura 4.3). El modelo NODE ha sido elegido el 65.91 % de las veces, siendo el segundo más elegido de entre los cuatro. No obstante, los participantes del test subjetivo ABX hicieron notar que aunque el modelo NODE producía una voz con una naturalidad ligeramente inferior a la del modelo ResNet, su inteligibilidad era claramente mayor. Esto puede deberse a que, como se refleja en la tabla 4.4, las redes NODE son capaces de predecir mejor los parámetros MFCCs, que influyen directamente en la inteligibilidad de la voz.

De la comparación de recursos computacionales usados por cada método (ver Ta-

bla 4.5) se infiere que el modelo NODE reduce el coste de memoria CPU frente a ResNet. Sin embargo, la utilización de memoria depende en gran parte del método de integración implementado para el módulo ODESolve. *dopri5* ofrece la ventaja de ser un método preciso y estable, con la desventaja de una mayor lentitud en el proceso de entrenamiento, lo que influye en el coste de memoria. Por el contrario, el método de Euler es el menos preciso. Debido a su bajo número de cálculos, el coste de memoria se reduce, disminuye la precisión del método y presenta problemas de estabilidad. En nuestro caso, de la Figura 4.2 se puede concluir que con el método *dopri5* se consigue disminuir el error de validación frente a ResNet, incluso al aumentar el número de capas de la red.

5 Conclusiones y trabajo futuro

En este trabajo se han evaluado distintos modelos de aprendizaje automático para una interfaz oral silenciosa capaz de sintetizar voz a partir de datos de captura de movimiento obtenidos de los órganos del habla. El objetivo es restaurar el habla a personas que hayan perdido la capacidad de hablar tras sufrir una enfermedad o un accidente incapacitante. Para ello, antes de perder el habla, se graban datos de voz y bioseñales de forma paralela para entrenar un modelo de aprendizaje automático y, posteriormente, cuando el paciente ha perdido el habla, aplicar estos modelos para predecir la voz a partir de las bioseñales. Estas señales pueden ser cerebrales, actividad eléctrica de los músculos de la cara o movimiento de los articuladores. El uso de un tipo u otro depende del tipo de enfermedad que padezca el paciente.

Las bioseñales utilizadas en este trabajo provienen de una base de datos en la que utilizaron un dispositivo PMA para recoger el movimiento de los órganos del habla. A partir de la recogida de estos datos junto con las correspondientes grabaciones de voz, hemos entrenado tres modelos del estado del arte, a saber, regresión lineal/logística y dos modelos de redes neuronales artificiales (DNN y redes residuales), y se ha propuesto el uso de una técnica recientemente propuesta para modelar la relación entre los dos dominios: las ecuaciones diferencias ordinarias neuronales. El objetivo último de este trabajo es demostrar si es posible utilizar las NODE como alternativa a las redes neuronales estándar para resolver problemas de aprendizaje automático, en general, y el problema de síntesis de voz a partir de bioseñales, en particular. Este modelo (NODE) utiliza una ecuación diferencial ordinaria para parametrizar la red. Se trata de un tipo especial de red neuronal residual, en el que todas las transformaciones desde la entrada hasta la salida son continuas.

5.1 Conclusiones

A partir de los resultados objetivos y subjetivos obtenidos, podemos concluir que las métricas objetivas obtenidas con redes neuronales NODE para los coeficientes MFCC, BAP y F_0 son mejores que para la red neuronal residual. Sin embargo, para el parámetro binario *voicing*, ResNet ofrece una menor tasa de error. Respecto a los resultados subjetivos, las NODE han sido escogidas un 65.91 % de las veces, resultando en el segundo modelo más elegido por los participantes del test. El modelo más elegido ha sido ResNet con un 85.61 %, y el menos elegido el modelo de regresión lineal, con un 7.31 %. Cabe destacar la diferencia entre la red NODE y el modelo de red neuronal profunda utilizado. Los resultados indican que utilizar NODE disminuye el error en las predicciones de los parámetros (excepto *voicing*) frente a utilizar únicamente una DNN. Además, una mayor elección de NODE en comparación con el 41.29 % de elección de DNN también indica que con el primer modelo de red neuronal la calidad de la voz sintetizada es mayor que con este último.

Finalmente, de la comparación del uso de memoria entre NODE y ResNet, hemos concluido que la red neuronal con ecuaciones diferenciales disminuye el uso de memoria CPU (dependiendo del método de integración), frente al modelo de red neuronal residual. Esto supone una ventaja, ya que disminuye el número de parámetros que calcula la red en la etapa de entrenamiento. El análisis de la evolución del error de validación frente a las iteraciones también ha revelado otra de las ventajas de las NODE frente a las redes neuronales residuales: hemos obtenido una convergencia más lenta del modelo. Es decir, un menor error de validación al aumentar el número de iteraciones. Además, el método de integración *dopri5* ha asegurado la estabilidad del modelo, a pesar de aumentar el tiempo de entrenamiento.

Por lo tanto, podemos concluir que se ha conseguido sintetizar voz a partir de bioseñales utilizando redes neuronales con ecuaciones diferenciales ordinarias, y se ha demostrado que ofrecen un conjunto de ventajas frente a las redes neuronales estándar, como las redes neuronales profundas o residuales.

5.2 Contribuciones

La principal contribución de este trabajo respecto a los estudios presentes en la literatura es la aplicación de las redes neuronales NODE al problema de síntesis de

voz a partir de datos de tipo articulatorio. En general, no se han encontrado estudios que utilicen este tipo de redes neuronales para sintetizar voz a partir de bioseñales.

5.3 Trabajo futuro

En este trabajo nos hemos centrado en un único tipo de bioseñales, relacionadas con el movimiento articular. Sería interesante extender este modelo de NODE a otro tipo de señales, como registros de actividad cerebral obtenida mediante métodos invasivos como la electrocorticografía (ECoG) o Electroencefalografía estereotáctica (sEEG). En los estudios [15, 14], por ejemplo, utilizan este tipo de señales para entrenar modelos de redes neuronales profundas, con el objetivo de descocidificar el habla a partir de su actividad cerebral. Ésta es una línea de trabajo en la que nos encontramos trabajando en este momento.

Una de las ventajas mencionadas de este tipo de redes neuronales es la precisión del método de integración. Si se elige un método de un alto orden, se reducen los problemas de estabilidad, a cambio de un aumento en el número de operaciones que se realizan. Por lo tanto, la paralelización en memoria distribuida del algoritmo ayudaría a disminuir el tiempo de entrenamiento de la red, conservando la estabilidad del método de integración.

6 Apéndice: Método de Euler

Existen varios métodos de integración de EDOs, entre los que el método de Euler es el método mas conocido para resolver la ecuación a partir de valores iniciales especificadas. Permite obtener de modo explícito la solución haciendo uso de la la expresión

$$y(t+\delta) = y(t) + \delta g(t, y) \tag{6.1}$$

Para calcular aproximaciones de y utilizando el método de Euler, comenzamos discretizando los dominios. Empezando por un punto inicial (t_0, y_0), definimos una *trayectoria de cálculo* (ver Figura 6.2) de forma recursiva:

$$t_{n+1} = t_n + \delta(n+1), \quad n = 0, 1, 2, \ldots \tag{6.2}$$

$$y_{n+1} = y_n + \delta g(t_{n+1}, y_{n+1}), \quad n = 0, 1, 2, \ldots \tag{6.3}$$

$$\tag{6.4}$$

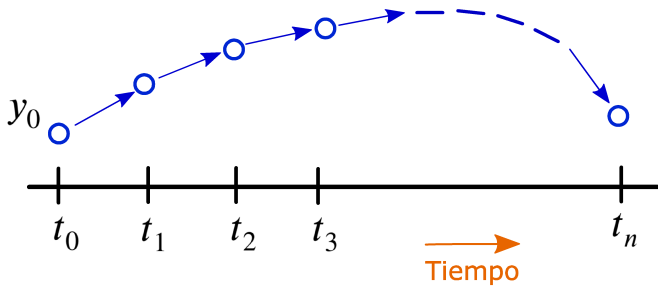

Figura 6.1: Trayectoria de cálculo en el método de Euler

Apéndice: Método de Euler

Existen varios métodos de integración de EDOs, entre los que el método de Euler es el método mas conocido para resolver la ecuación a partir de valores iniciales especificadas. Permite obtener de modo explícito la solución haciendo uso de la la expresión

$$y(t+\delta) = y(t) + \delta g(t, y) \tag{6.5}$$

Para calcular aproximaciones de y utilizando el método de Euler, comenzamos discretizando los dominios. Empezando por un punto inicial (t_0, y_0), definimos una *trayectoria de cálculo* (ver Figura 6.2) de forma recursiva:

$$t_{n+1} = t_n + \delta(n+1), \quad n = 0, 1, 2, \ldots \tag{6.6}$$

$$y_{n+1} = y_n + \delta g(t_{n+1}, y_{n+1}), \quad n = 0, 1, 2, \ldots \tag{6.7}$$

$$\tag{6.8}$$

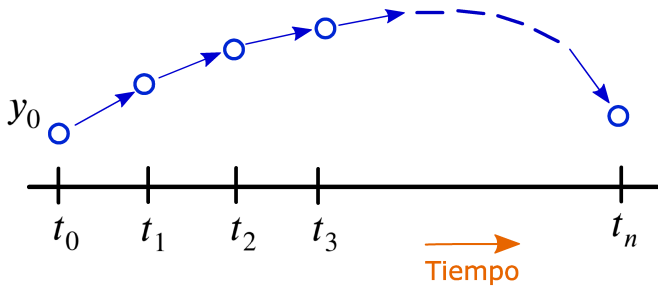

Figura 6.2: Trayectoria de cálculo en el método de Euler

7 | Aplicaciones

Un campo natural de aplicación de las NODE es la resolución de ecuaciones diferenciales en derivadas parciales. Aunque no es el objetivo principal de esta memoria, hemos creído conveniente incluir un ejemplo simple que puede servir de aclaración del método. Para ello, usaremos la ecuación de Burgers.

Ecuación de Burgers

La ecuación de Burgers es una ecuación diferencial en derivadas parciales que aparece en varias áreas de la matemática aplicada, como la mecánica de fluidos, la acústica no lineal, la dinámica de gases y el flujo de tráfico.

Para un campo dado $u(x,t)$ y un coeficiente de difusión ν, la forma general de la ecuación de Burgers en una dimensión espacial tiene la siguiente forma:

$$\frac{\partial u(t,x)}{\partial t} + u(t,x)\frac{\partial u(t,x)}{\partial x} = \nu \frac{\partial^2 u(t,x)}{\partial^2 x} \tag{7.1}$$

donde el dominio de x ha sido el intervalo $[-1,1]$, y el de la variable temporal t, el intervalo $[0,1]$. La condición inicial, en $t = 0$, se ha escogido igual a $u(0,x) = -\sin(\pi x)$, y las condiciones de contorno $u(t,-1) = u(t,1) = 0$. Para el coeficiente de difusión ν se ha tomado $\nu = 0.01/\pi$.. Cuando este coeficiente es igual a cero, $\nu = 0$, la ecuación de Burgers describe las ondas de choque.

La idea principal de las redes neuronales para resolver ecuaciones en derivadas parciales (Physics Informed Neural Networks (PINNS)) [33] es implementar una red que defina una relación entre el dominio x y t :

$$net(t, x) : [0, 1] \times [-1, 1] \mapsto \mathbb{R} \tag{7.2}$$

Esta red, una vez entrenada, debe satisfacer las condiciones de contorno y la ecuación diferencial. Para satisfacer la ecuación diferencial, debemos ser capaces de diferenciar la red en función de x y t. Este proceso se hace mediante el algoritmo de *backpropagation* de la red. Las redes neuronales son funciones no lineales con primeras derivadas no nulas. En este caso, también necesitamos la segunda derivada, por lo que utilizaremos la función de activación tangente hiperbólica. Esta función transforma los valores introducidos a una escala $(-1, 1)$, donde los valores altos tienden de manera asintótica a 1 y los valores bajos tienden de manera asintótica a -1

$$f(x) = \frac{2}{1 + e^{-2x}} - 1 \tag{7.3}$$

El objetivo de la red diseñada es minimizar las funciones:

$$f(t, x) = \frac{\partial u(t, x)}{\partial t} + u(t, x)\frac{\partial u(t, x)}{\partial x} - \nu\frac{\partial^2 u(t, x)}{\partial^2 x} \tag{7.4}$$

$$boundary(t, x) = u(t, x) + \sin(x/\pi) \;\; cuando \;\; t = 0 \;\; y \tag{7.5}$$

$$u(t, x) = 0 \;\; cuando \;\; x = -1 \;\; \& \;\; x = 1 \tag{7.6}$$

La red se ha entrenado con 50000 iteraciones (*epochs*) utilizando dos optimizadores: uno para las condiciones de contorno, y otro para la función f que define la ecuación diferencial. En cada iteración, generamos un conjunto aleatorio de muestras dentro del rectángulo $[0, 1] \times [-1, 1]$. Cada muestra es una tupla consistente en un valor para t, un valor para x y un valor u de frontera conocido (o nulo). Recordemos que estamos forzando a la función $f(t, x)$ a que valga cero, y a la condición de frontera a que valga, o cero o $u(0, x)$.

En la Figura 7.1 se ha representado la evolución de la solución estimada mediante NODE y la solución analítica. Cada línea representa $u(t, x)$ para un valor de t distinto. La línea azul es la condición inicial.

Los resultados obtenidos muestran concordancias evidentes, que sin duda, podrían mejorarse sintonizando ad–hoc los parámetros de la red (epochs, número de parámetros, etc.). No obstante, con este sencillo ejemplo sólo he querido ilustrar la dualidad

que existe entre problemas de redes neuronales NODE, y problemas de solución de ecuaciones diferenciales. Ambos, desde un punto de vista general, pueden verse como la búsqueda de funciones no lineales que, para NODE, y con la elección de funciones de activación adecuadas (en nuestro caso con primera y segunda derivadas suaves), pueden entrenarse para resolver ecuaciones diferenciales complejas, con requisitos computacionales, en general menores, además de ser adaptables y poder resolver problemas complejos con precisión.

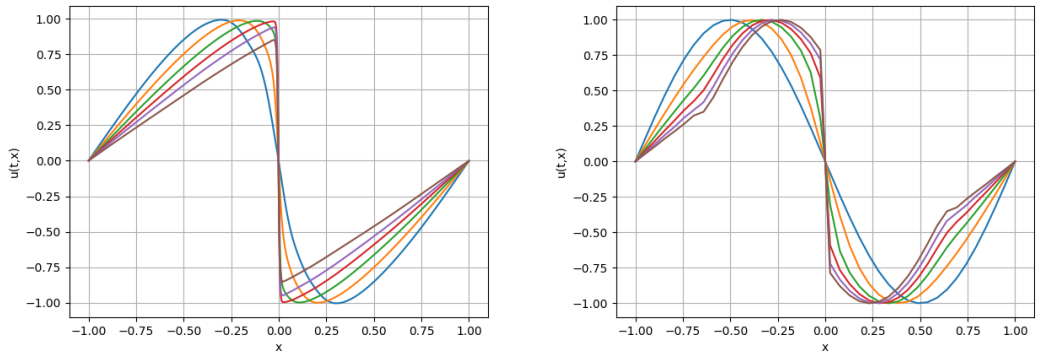

Figura 7.1: Solución de la ecuación de Burgers (para distintos instantes de tiempo). **Izda**: con NODE. **Dcha**: Solución analítica

Bibliografía

[1] J. A. Gonzalez-Lopez, A. Gomez-Alanis, J. M. Martín Doñas, J. L. Pérez-Córdoba, and A. M. Gomez, "Silent speech interfaces for speech restoration: A review," *IEEE Access*, vol. 8, pp. 177 995–178 021, 2020.

[2] R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. Duvenaud, "Neural ordinary differential equations," *Advances in Neural Information Processing Systems*, pp. 6571–6583, 2018.

[3] S. Shakya, "Analysis of artificial intelligence based image classification techniques," in *Journal of Innovative Image Processing*, 2020, pp. 44–54.

[4] G. Venayagamoorthy, V. Moonasar, and K. Sandrasegaran, "Voice recognition using neural networks," in *Proceedings of the 1998 South African Symposium on Communications and Signal Processing-COMSIG '98 (Cat. No. 98EX214)*, 1998, pp. 29–32.

[5] M. Gardner, J. Grus, M. Neumann, O. Tafjord, P. Dasigi, N. Liu, M. Peters, M. Schmitz, and L. Zettlemoyer, "Allennlp: A deep semantic natural language processing platform," *arXiv preprint arXiv:1803.07640*, 2018.

[6] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Networks*, vol. 61, pp. 85–117, 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0893608014002135

[7] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[8] "Deep Learning Market," https://www.marketsandmarkets.com/Market-Reports/deep-learning-market-107369271.html, accessed: 2021-07-03.

[9] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015.

[10] Y. Li, H. Yi, C. M. Bender, S. Shan, and J. B. Oliva, "Exchangeable neural ode for set modeling," *AAAI*, 2020.

[11] J. A. Gonzalez, L. A. Cheah, A. M. Gomez, P. D. Green, J. M. Gilbert, S. R. Ell, R. K. Moore, and E. Holdsworth, "Direct speech reconstruction from articulatory sensor data by machine learning," vol. 25, pp. 2362–2374, 2017.

[12] D. Erro, I. Hernaez, L. Serrano, I. Saratxaga, and E. Navas, "Objective comparison of four gmm-based methods for pma-to-speech conversion," in *International Conference on Advances in Speech and Language Technologies for Iberian Languages*, vol. Lecture Notes in Computer Science, vol 10077. Springer, Cham, Springer. IberSPEECH, 2016, pp. 24–32. [Online]. Available: https://doi.org/10.1007/978-3-319-49169-1_3

[13] M. Angrick, M. Ottenhoff, L. Diener, D. Ivucic, G. Ivucic, S. Goulis, J. Saal, A. J. Colon, L. Wagner, D. J. Krusienski *et al.*, "Real-time synthesis of imagined speech processes from minimally invasive recordings of neural activity," *bioRxiv*, 2020.

[14] M. Angrick, C. Herff, E. Mugler, M. C. Tate, M. W. Slutzky, D. J. Krusienski, and T. Schultz, "Speech synthesis from ecog using densely connected 3d convolutional neural networks," *Journal of neural engineering*, vol. 16, no. 3, p. 036019, 2019.

[15] M. Angrick, C. Herff, G. Johnson, J. Shih, D. Krusienski, and T. Schultz, "Speech spectrogram estimation from intracranial brain activity using a quantization approach."

[16] G. K. Anumanchipalli, J. Chartier, and E. F. Chang, "Speech synthesis from neural decoding of spoken sentences," *Nature*, vol. 568, pp. 493–498, 24 April 2019. [Online]. Available: https://doi.org/10.1038/s41586-019-1119-1

[17] M. MORISE, F. YOKOMORI, and K. OZAWA, "World: A vocoder-based high-quality speech synthesis system for real-time applications," vol. E99.D, pp. 1877–1884, 2016.

[18] C. Herff, L. Diener, M. Angrick, E. Mugler, M. C. Tate, M. A. Goldrick, D. J. Krusienski, M. W. Slutzky, and T. Schultz, "Generating natural, intelligible speech from brain activity in motor, premotor, and inferior frontal cortices," *Front. Neurosci.*, vol. 13:1267, 2019. [Online]. Available: 10.3389/fnins.2019.01267

[19] A. M. Noll, "Short-time spectrum and "cepstrum" techniques for vocal-pitch detection," *The Journal of the Acoustical Society of America*, vol. 36, pp. 296–302, 1964.

[20] m. morise, h. kawahara, and h. katayose, "fast and reliable f0 estimation method based on the period extraction of vocal fold vibration of singing voice and speech," *journal of the audio engineering society*, february 2009.

[21] M. Morise, "Cheaptrick, a spectral envelope estimator for high-quality speech synthesis," *Speech Communication*, vol. 67, pp. 1–7, 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167639314000697

[22] ——, "D4c, a band-aperiodicity estimator for high-quality speech synthesis," *Speech Communication*, vol. 84, pp. 57–65, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167639316300413

[23] ——, "Platinum: A method to extract excitation signals for voice synthesis system," *The Acoustical Society of Japan*, vol. 33, pp. 123–125, 2012.

[24] G. Palm, "Warren mcculloch and walter pitts: A logical calculus of the ideas immanent in nervous activity," pp. 229–230, 1986.

[25] R. F., "The perceptron: A probabilistic model for information storage and organization in the brain," pp. 386–408, 1958.

[26] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[27] "Pytorch," https://pytorch.org/, accessed: 2021-05-12.

[28] E. Dupont, A. Doucet, and Y. W. Teh, "Augmented neural odes."

[29] J. A. Gonzalez, L. A. Cheah, J. M. Gilbert, J. Bai, S. R. Ell, P. D. Green, and R. K. Moore, "A silent speech system based on permanent magnet articulography and direct synthesis," *Computer Speech & Language*, vol. 39, pp. 67–87, 2016.

[30] K. P. F.R.S., "Liii. on lines and planes of closest fit to systems of points in space," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 2, no. 11, pp. 559–572, 1901. [Online]. Available: https://doi.org/10.1080/14786440109462720

[31] J. R. Dormand and P. J. Prince, "A family of embedded runge-kutta formulae," *Journal of computational and applied mathematics*, vol. 6, no. 1, pp. 19–26, 1980.

[32] R. Kubichek, "Mel-cepstral distance measure for objective speech quality assessment," in *Proceedings of IEEE Pacific Rim Conference on Communications Computers and Signal Processing*, vol. 1, 1993, pp. 125–128 vol.1.

[33] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics informed deep learning (part ii): Data-drivendiscovery of nonlinear partial differential equations," 2017. [Online]. Available: arXiv:1711.10566[cs.AI]