Universidad Tecnológica Nacional
Facultad Regional Córdoba

Clasificación automática del grado general de disfonía

Mario Alejandro García

Director:
Eduardo A. Destéfanis

Tesis presentada para obtener el grado de
Doctor en Ingeniería mención Sistemas de Información

3 de diciembre de 2021

Resumen

Doctor en Ingeniería mención Sistemas de Información

Clasificación automática del grado general de disfonía

por Mario Alejandro García

El análisis audioperceptivo es una parte principal de la rutina de evaluación clínica de pacientes con trastornos de la voz para medir y registrar la calidad vocal. Esta valoración repercute en la detección y tratamiento de enfermedades vocales. GRBAS es una escala de valoración de cinco dimensiones usualmente utilizada en este proceso, donde la dimensión "G" representa el grado general de disfonía.

En este trabajo se desarrolla un modelo de aprendizaje profundo para calcular el grado general de disfonía en escala GRBAS con el propósito de contribuir a la comprensión y mejora de este tipo de modelos en el ámbito de la calidad vocal.

La arquitectura de la red neuronal se definió a partir de dos versiones de un modelo llamado modelo inicial, creado en base a modelos más pequeños. Estos modelos neuronales pequeños, diseñados a partir del conocimiento del dominio del problema, se enfocan en resolver la representación frecuencial del audio, la extracción de características y la clasificación. Una de las versiones recibe el audio sin procesar como entrada y la otra recibe el cepstrograma. Los modelos fueron entrenados y evaluados con los datos de una base de datos pública que contiene audios y valoraciones en escala GRBAS, a los que se añadieron valoraciones de calidad vocal realizadas por un evaluador local.

Las métricas utilizadas para evaluar los modelos de clasificación son la exactitud y el error absoluto medio. Debido a que las valoraciones audioperceptivas tienen un alto grado de variación entre distintos evaluadores médicos y también entre distintas valoraciones del mismo evaluador médico, se plantea que el rendimiento de los modelos automáticos se debe evaluar en relación al índice de concordancia medio interevaluador e intraevaluador de la base de datos.

En base a los resultados obtenidos, por un lado se concluye que la versión del modelo que recibe el cepstrograma como entrada es capaz de predecir el grado general de disfonía, para los datos utilizados, con una exactitud cercana a la de un evaluador humano. Por otro lado, se concluye que una red neuronal profunda diseñada para reconocer patrones de perturbación de amplitud, perturbación de frecuencia y ruido, obtiene información útil para la predicción del grado general de disfonía, que para el diseño utilizado (donde

la extracción de características está basada en el cepstrum) no es posible utilizar el audio sin procesar como entrada y que el modelo presentado es un buen punto de partida para futuros desarrollos de clasificadores de la calidad vocal aplicables en la práctica clínica.

Agradecimientos

A Eduardo Destéfanis por su apoyo y dirección.

A Lorena Rosset por sus aportes de conocimiento, de trabajo y todo el apoyo.

A Juan Ignacio Godino-Llorente de la Universiad Politécnica de Madrid, por compartir los datos de HUPA.

A la Escuela de Fonoaudiología de la Universidad Nacional de Córdoba, por la cooperación en el análisis y valoración de las muestras de audio.

A Diego Evin por las charlas y consejos.

A Roberto Muñoz, Marcelo Marciszack, Juan Calloni, Marta y José de la SCyT por el apoyo de siempre.

A NVIDIA, por la donación de una GPU Titan Xp a través del NVIDIA GPU *Grant Program*, sin la cual habría sido imposible realizar los experimentos.

Índice general

Índice de figuras

Índice de tablas

Dedicado a mis padres, mis hijos y mi esposa

Un agradecimiento especial para todos ellos

Capítulo 1

Introducción

En este trabajo se lleva a cabo la clasificación de la calidad vocal con técnicas de aprendizaje profundo (AP). El objetivo de la clasificación es el grado general de disfonía (G) en escala GRBAS. Para lograrlo se desarrolla un modelo de clasificación basado en redes neuronales artificiales (RNA) que recibe audio como entrada y predice el grado de disfonía asignado por un profesional de la voz.

1.1. Motivación

El análisis audioperceptivo es una parte principal de la rutina de evaluación clínica de pacientes con trastornos de la voz para documentar la calidad vocal [1]. Esta valoración repercute en la detección y seguimiento del tratamiento tanto de patologías vocales como de otras enfermedades que, si bien no tienen origen en los órganos productores de la voz, presentan como síntoma o signo el deterioro de la calidad vocal, por ejemplo la enfermedad de Parkinson [2, 3].

La necesidad de estandarizar la valoración y de interrelacionar los aspectos auditivos y fisiológicos de la producción vocal, estimuló la creación de distintos métodos y escalas de evaluación audioperceptiva, como por ejemplo GRBAS y CAPE-V[1].

Aún con la utilización de estas escalas, la valoración de la voz tiene una gran variabilidad entre distintos profesionales de la salud (variabilidad interevaluador) y también entre distintas valoraciones del mismo evaluador (variabilidad intraevaluador) [4]. La clasificación automática de la calidad vocal podría reducir la subjetividad del análisis actual, estandarizando la valoración entre distintos profesionales y permitiendo resultados repetibles e independientes de los cambios ambientales para un mismo profesional.

1.1.1. GRBAS

La escala GRBAS es un método de valoración audioperceptiva de la voz. Está basada en estudios que se iniciaron en el año 1966 de la *Japan Society of Logopedics and Phoniatrics* [5] y que posteriormente fueron divulgados por Minoru Hirano en el año 1981 [6]. Consiste en la valoración de la voz a nivel de la fuente glótica a través de 5 dimensiones que forman el acrónimo GRBAS:

- G: Grado general de disfonía (*Grade*).
- R: Rugosidad (*Roughness*), irregularidad de la onda glótica.
- B: Soplosidad (*Breathiness*), sensación de escape de aire en la voz.
- A: Astenia (*Asteny*), pérdida de potencia.
- S: Tensión (*Strain*), sensación de hiperfunción vocal.

Se valora en cuatro grados, desde el "0" hasta el "3". El grado "0" indica ausencia de disfonía y el "3" disfonía severa. La escala fue mundialmente adoptada y validada en numerosos países [7–10]. Actualmente se utiliza en la investigación y de manera rutinaria en los consultorios de los profesionales que hacen clínica vocal. Sirve como metodología simple y al alcance de la mano para valorar la evolución pre/post tratamiento.

La debilidad de los métodos audioperceptivos reside en la subjetividad de la valoración de la voz y en la necesidad de que sea realizada por oyentes experimentados en la escucha y la disociación de los parámetros [11, 12]. Una alternativa al método audioperceptivo es el análisis acústico.

1.1.2. Análisis acústico y calidad vocal

El análisis acústico (no automático) de la voz, a pesar de ser más objetivo que el método audioperceptivo [13], necesita de la intervención del evaluador y esto impone también un grado de subjetividad. Por este motivo surge la necesidad de la estandarización de todos los pasos en los que el sujeto evaluador interviene: elección del material de habla que se graba, grabación, dispositivos electrónicos, software, elección de tipos de análisis a realizar (no se puede analizar todas las voces, más o menos disfónicas, con las mismas pruebas) y finalmente, el análisis físico-acústico y fisiológico de los datos obtenidos. Se han realizado esfuerzos por encontrar relaciones entre características acústicas del audio y los niveles de GRBAS u otras escalas [14, 15], algunos relacionados con la voz normal y otros con diferentes patologías. Por ejemplo, el trabajo de Nuñez Batalla *et al.* [16]

establece una relación entre la dimensión A de GRBAS y la pérdida de componentes armónicos en ciertas porciones del espectrograma de banda angosta. A pesar de los esfuerzos realizados, no se ha de logrado encontrar una relación entre medidas acústicas de la voz y la calidad vocal (o las enfermedades que la afectan) que permita mapear las primeras en el dominio de la segunda.

Existe una gran cantidad de medidas acústicas y de herramientas (métodos y paquetes de software) para calcularlas. Una complicación para el cálculo de estas medidas es que gran parte depende de la frecuencia fundamental (F_0) de la señal, que se utiliza como un estimador de la frecuencia de vibración de las cuerdas vocales [17]. La frecuencia es fácil de definir para funciones sinusoidales y periódicas, pero este no es el caso de la voz, ni siquiera para segmentos muy cortos de tiempo [18–20]. Para señales no periódicas la noción de frecuencia es ambigua y muchas veces confusa, lo que ha dado lugar a una multitud de técnicas para medirla y la consecuente discrepancia entre las medidas [19, 20]. En [14], Freitas *et al.* calculan la correlación de medidas acústicas tradicionales individuales y las dimensiones de GRBAS, encontrando gran variabilidad según el software de análisis utilizado (Dr. Speech, MDVP, Praat y Voice Studio).

Actualmente se continúa con el desarrollo, tanto de nuevas formas de calcular F_0 [19, 21, 22], como de nuevas medidas acústicas para la voz, algunas de estas muy complejas basadas en sistemas dinámicos no lineales [13, 23, 24]. Los investigadores agrupan, según su conveniencia, las medidas acústicas en "derivadas de F_0", "ratio señal-ruido", "medidas *wavelet*" y "coeficientes cepstrales de frecuencias de mel"; "*short term*", "*long time analysis*", "de cuantización de energía" y "glóticas"; "de sistemas lineales" y "de sistemas no lineales"; etc. Algunas de estas clasificaciones con sus respectivos conjuntos de medidas acústicas de la voz se pueden ver en [25–30]. Las medidas relevantes para este trabajo se explican en el capítulo 5.

1.1.3. Clasificación automática de la calidad vocal

La clasificación automática de la calidad vocal en las escalas de valoración audioperceptiva es un tema activo de investigación. Habitualmente se toma un conjunto de medidas acústicas como características para ajustar algún modelo de aprendizaje automático. Un resumen de algunos trabajos relevantes se puede ver en la sección 1.5.

Las redes neuronales profundas se han convertido en el estado del arte del reconocimiento de patrones. La tendencia de estos modelos es utilizar los datos de entrada sin procesar o con procesamiento muy básico y realizar la extracción de características en el interior de la red neuronal. En este trabajo se estudia la arquitectura que una red neuronal debe tener para clasificar el grado de disfonía G recibiendo audio como entrada. Se busca

una arquitectura capaz de extraer del audio información similar a las medidas acústicas relacionadas con la calidad vocal.

1.2. Objetivos

Esta tesis busca contribuir a la comprensión y mejora de las redes neuronales profundas aplicadas al reconocimiento de patrones relacionados con la calidad vocal, para lo cual se propuso el siguiente objetivo general: Lograr la clasificación automática del grado general de disfonía mediante una red neuronal profunda desarrollada a partir del conocimiento del dominio.

En relación con el objetivo general, se establecieron los siguientes objetivos particulares:

1. Desarrollar y comparar modelos neuronales que generen distintas representaciones de los datos de audio.
2. Desarrollar modelos neuronales, basados en medidas acústicas, que extraigan información útil para estimar el grado general de disfonía.
3. Desarrollar un modelo neuronal capaz de estimar el grado general de disfonía.

1.3. Contribuciones

La principales contribuciones de esta tesis son:

- Un modelo de aprendizaje profundo para estimar el grado general de disfonía G de la escala GRBAS.
- Redes neuronales para crear una representación frecuencial interna en la clasificación extremo a extremo de audio.
  - Capa STHadamard para realizar la operación de *windowing* con redes neuronales. Se logra la segmentación de una señal con ventanas entrenables de ancho fijo.
  - Una red neuronal para la generación del *power spectrogram* mediante el cálculo de la transformada de Fourier de término reducido con capas de convolución.
  - Una red neuronal para el cálculo del *power cepstrum*.
- Una red neuronal para el cálculo de *shimmer* sobre audio sintetizado.

- Un conjunto de operaciones de aumentación de datos para la clasificación de la calidad vocal.

Durante el desarrollo del plan de tesis se realizaron otros aportes que, si bien fueron publicados, no se utilizaron en el modelo final presentado. Estos son:

- Una red neuronal para la estimación de la frecuencia de vibración de las cuerdas vocales.
- Una red neuronal para el cálculo del máximo en una secuencia.
- Un método de ajuste en la distribución de la variable objetivo en los datos de entrenamiento según los niveles de error. Trabajo relacionado con el cálculo de *shimmer* sobre audio sintetizado.

1.4. Publicaciones

1.4.1. Publicaciones en revistas y congresos con referato

1. M.A. García y E.A. Destéfanis. Deep Neural Networks for Shimmer Approximation in Synthesized Audio Signal. En: *XXIII Congreso Argentino de Ciencias de la Computación*, Springer International Publishing, 2017.
2. M.A. García y colab. Ajuste de Distribución de Datos para Aproximación de Shimmer en Audio con Deep Neural Networks. En: *V Congreso Nacional de Ingeniería Informática - Sistemas de Información*, Santa Fe, 2017.
3. M.A. García y E.A. Destéfanis. Spectogram Prediction with Neural Networks. En: *XXIV Congreso Argentino de Ciencias de Computación*. Tandil, 2018.
4. M.A. García y colab. Predicción de la frecuencia fundamental de la voz con redes neuronales convolucionales. En: *VI Congreso Nacional de Ingeniería Informática – Sistemas de Información*, Mar del Plata, 2018
5. M.A. García y E.A. Destéfanis. Power Cepstrum Calculation with Convolutional Neural Networks. En: *Journal of Computer Science & Technology*, 19(2):132–142, 2019.
6. M.A. García y colab. Cálculo de la posición del valor máximo en una secuencia con redes neuronales. En: *VII Congreso Nacional de Ingeniería Informática – Sistemas de Información*, Buenos Aires, 2019.

7. M.A. García y E.A. Destéfanis. Trainable Windowing Coefficients in DNN for Raw Audio Classification. En: *JCC-BDET 2020*, CCIS 1291, 153–166, Springer Nature Switzerland AG, 2020.

1.4.2. Publicaciones en talleres

En esta sección se enumeran trabajos sin referato publicados como resúmenes, pósteres y/o presentaciones en reuniones científicas.

1. M.A. García. Valoración de la Calidad Vocal a través de técnicas de Aprendizaje Automático. En: *7ma. Escuela Argentina de Biología y Matemática*, Huerta Grande, 2016.

2. M.A. García y colab. Aplicación de Técnicas de Aprendizaje Automático para Estimar la Calidad de la Voz en Escala GRBAS. En: *XIX Workshop de Investigadores de Ciencias de la Computación.*, Buenos Aires, 2017.

3. M.A. García y colab. Deep Neural Networks para Análisis Acústico. En: *XX Workshop de Investigadores de Ciencias de la Computación*, Corrientes, 2018.

4. M.A. García. Cálculo del Espectrograma con Redes Neuronales. En: *Jornada de Difusión de los Resultados de I+D del Centro de Investigación en Informática para la Ingeniería*, Córdoba, 2018.

5. M.A. García y colab. Extracción de Características en Audio con Redes Neuronales Convolucionales. En: *XXI Workshop de Investigadores de Ciencias de la Computación*, San Juan, 2019.

6. M.A. García. Clasificación automática de la calidad vocal. En: *Jornadas de Intercambio y Difusión de los Resultados de Investigación de los Doctorandos en Ingeniería*, Córdoba, 2019.

7. M.A. García. Data Augmentation para la Clasificación Automática de la Calidad Vocal. En: *Jornadas de Intercambio y Difusión de los Resultados de Investigación de los Doctorandos en Ingeniería*, Virtual, 2020.

8. M.A. García y colab. Red neuronal multiescala para clasificación de la calidad vocal. En: *XXIII Workshop de Investigadores de Ciencias de la Computación*, Chilecito, 2021.

1.5. Trabajos relacionados

La clasificación automática de la calidad vocal se encuentra en pleno desarrollo y no existe consenso sobre los métodos de clasificación ni las medidas acústicas a utilizar. A continuación se resumen los trabajos más relevantes de los últimos años, con foco en la utilización de redes neuronales artificiales (RNA) y aprendizaje profundo (AP). Los trabajos se agrupan, en lo posible, por equipos de investigadores y, dentro de cada equipo, se ordenan en forma cronológica. Se incluyen los trabajos de detección de voces patológicas y de clasificación de enfermedades de la voz. Ambos están muy relacionados con la calidad vocal porque se basan en reconocer patrones de voces anormales. Es común que los mismos grupos de investigación trabajen en los tres objetivos de clasificación. Las características (*features*) y métodos de clasificación también son con frecuencia los mismos, pero a diferencia de la clasificación de la calidad vocal, la detección de voces patológicas y la clasificación de enfermedades cuentan con la ventaja de que la salida esperada no es subjetiva, esta se obtiene de diagnósticos médicos confirmados.

Durante la lectura de esta sección recordar que G, R, B, A y S representan cada una de las dimensiones de GRBAS.

En el año 2014 Yu *et al.* en [30] utilizan 65 atributos acústicos incluyendo los Coeficientes Cepstrales en las Frecuencias de Mel (MFCC), *Glottal to Noise Excitation* y *Vocal Fold Excitation Ratio* para evaluar el grado general de disfonía G. El mejor resultado alcanzado tiene una exactitud del 70 %. Los datos están compuestos por 746 muestras de 300 personas emitiendo una vocal sostenida /a/ de la base de datos china grabada en el *People's Liberation Army General Hospital*. La selección de atributos se realiza con *Relief* y *Minimum Redundancy Maximum Relevance Feature Selection*, se reduce la dimensión con Análisis de Componentes Principales (PCA) y se clasifica con Maquinas de Soporte Vectorial (SVM) utilizando como *kernels* Funciones de Base Radial (RBF). En el año 2015 un equipo formado mayormente por los mismos investigadores reporta una exactitud del 77.55 % utilizando SVM con RBF y del 80.54 % con *Extreme Learning Machine* en la predicción de G [31]. El trabajo tiene pocos cambios con respecto al anterior. Los autores atribuyen la mejora a una mejor distribución de la frecuencia de las clases lograda mediante la incorporación de 57 voces con grado 3 de disfonía. El mismo equipo, con algunos cambios, utiliza una red neuronal profunda en 2016 para predecir G[32]. En este nuevo trabajo logran una exactitud del 81.53 % utilizando un modelo de 5 capas densas con 512 neuronas cada una y función de activación softmax. Las entradas son vectores de dimensión 44 que incluyen medidas calculadas sobre audio continuo y medidas y representaciones frecuenciales tomadas de la vocal /a/ sostenida. Estas últimas son *Modulation Spectra* (MS), MFCC, *Shoothed Cepstral Peak Prominence* (CPPS) y *Long-Term Average Spectrum*. En el último trabajo se menciona que los audios fueron

valorados por cinco patologistas practicantes, pero no se informa la variabilidad inter-evaluador. No se hace referencia a dobles evaluaciones, por lo tanto se asume que cada evaluador valoró solo una vez a cada audio. Tampoco se explica qué valoración se tomó para el entrenamiento y validación.

Ritchings *et al.* en el año 2002 clasifican la calidad de voces patológicas en una escala de 7 niveles con una red neuronal [33]. Utilizan 77 grabaciones propias. Las grabaciones incluyen audio y la señal de un electroglotógrafo (EGG), aunque para la clasificación se utiliza un conjunto de características extraídas solo de la señal del EGG. El modelo tiene dos capas con neuronas densamente conectadas y funciones de activación softmax. Reportan una exactitud superior al 92 %, pero los resultados son discutibles porque se utilizan segmentos de la misma grabación tanto para entrenar como para validar el modelo. En un trabajo conjunto [34] del año 2003, Ritchings y Godino-Llorente revisan el trabajo anterior bajo la hipótesis de que en lugar de clasificar la calidad de la voz se estaba reconociendo al hablante y devolviendo la calidad asociada. Cuidadosamente se seleccionan los segmentos de grabación para que los utilizados en el entrenamiento no se usen para validar. Con este cambio, la exactitud se reduce en un 40 %. Godino-Llorente es un referente en área de la clasificación automática de la calidad vocal. Durante el resto de este párrafo se mencionan otros trabajos relevantes más actuales de Godino-Llorente y su equipo de la Universidad Politécnica de Madrid (UPM). En el año 2015 proponen caracterizar el audio mediante MS para predecir G y R [35]. Comparan dos grupos de entradas, uno basado en MS y otro en MFCC, con exactitud para G de 81.6 % y 80.5 % respectivamente. Realizan una reducción de la dimensión con PCA y *Linear Discriminant Analysis*, para clasificar después con *Gaussian Mixture Models* (GMM). Los datos son vocales /a/ sostenidas de la *Kay Elemetrics Voice Disorders Database*, grabada por el *Massachusetts Eye and Ear Infirmary Voice Laboratory* (MEEI). En 2019 presentan un trabajo orientado a emular la capacidad de percepción del evaluador humano en la escala GRBAS para las dimensiones G, B y R[36]. El foco del trabajo está en la naturaleza ordinal de los niveles de valoración. Se comparan dos modelos de predicción, *Extreme Learning Machines with Ordered Partition* para clasificación ordinal y Regresión Gaussiana para regresión. Se usan datos de tres bases de datos para entrenar, Hospital Gregorio Marañón, *Saarbrücken Voice Database* (SVD) y Hospital Universitario Príncipe de Asturias (HUPA), perteneciente a la UPM. La validación se realiza sobre el set de datos Doctor Negrín y el corpus Aplicación de las Tecnologías de la Información y las Comunicaciones (ATIC). Los audios de todos los dataset fueron evaluados para este trabajo por un profesional con 15 años de experiencia. Se realizó una doble valoración (salvo para ATIC), pero no se informa la variabilidad interevaluador obtenida. Como características se utiliza un conjunto de 24 medidas agrupadas en perturbaciones, basadas en MS, complejidad y espectrales y cepstrales. La métrica de

evaluación del modelo es el error absoluto medio (MAE) y una versión del mismo para clases no balanceadas. El MAE alcanzado es aproximadamente 0.5. En el año 2020, este mismo grupo publicó pruebas de clasificación de G, R y B con aprendizaje profundo sobre la base de datos SVD [37]. Se utilizaron tres modelos, uno con tres capas densas que recibe un conjunto de medidas similares a las del trabajo anterior, otro con capas de convolución, *max pooling* y densas que recibe los valores máximos, medios y la desviación estándar del MS del audio; y por último una combinación de las dos arquitecturas anteriores. Las métricas para la evaluación de los modelos son versiones de exactitud y MAE para datos no balanceados. Los mejores resultados se obtuvieron con el primer modelo (solo capas densas), 62 % de exactitud y MAE 0.45 para la predicción de G.

Harar *et al.* en 2017 clasifican los audios de la SVD en las clases "patológica" y "no patológica" con una red neuronal profunda [38]. El modelo está compuesto por capas de convolución, *max pooling*, *long short-term memory* y densas. Los datos de entrada son señales de *raw audio* de 3.2 s en segmentos de 64 ms multiplicados por una ventana Hamming con solapamiento de 30 ms. Logran un 68 % de exactitud. Los resultados deberían ser verificados porque, según los datos informados, el juego de datos de test está muy desbalanceado y no se tomó ninguna precaución al respecto. En 2019, el mismo grupo detecta voces patológicas sobre MEEI, SVD, *Arabic Voice Pathology Database* y HUPA con *Gradient Boosted Trees* y redes neuronales profundas [39]. La red neuronal utilizada es una modificación del modelo ResNet[40] de la librería *Keras*[1] y se entrena con tres tipos de entradas, MFCC, espectrogramas y *raw audio*. Se utiliza la métrica *F1-score* o valor-F en lugar de la exactitud. El mejor resultado se logra con árboles de decisión, 0.733. Con redes neuronales y MFCC se alcanzó 0.621, con el espectrograma 0.460 y con *raw audio* la red no logró aprender.

Otro caso de detección de patologías vocales con redes neuronales profundas es el publicado por Fang *et al.* en 2019 [41]. Los datos fueron grabados en el *Far Eastern Memorial Hospital* (FEMH), Taiwán. Se compara el rendimiento del modelo con SVM y GMM para tres variantes de datos de entrada (todos basados en MFCC). En los resultados sobre FEMH, AP logra mejor rendimiento que el resto, con exactitud de 94.2 % para hombres y 90.5 % para mujeres. Se realiza una validación extra sobre MEEI, donde se obtiene un llamativo 99.1 % de exactitud y se invierte el orden de la mejor variante de datos de entrada. El modelo neuronal parece estar fomrmado solo por capas densamente conectadas, pero no está claramente explicado.

Muhammad, Alhamid y otros en 2018 presentan un trabajo con dos objetivos, por un lado se detectan voces patológicas y por el otro, se clasifican las voces patológicas en 6 enfermedades, todo sobre vocales /a/ sostenidas de la SVD [42]. Se usa el modelo

[1] https://keras.io

de AP CaffeNet. Las entradas son espectrogramas a los que se les aplican tres filtros lineales de regresión. La salida de cada filtro se toma como uno de los canales RGB de la red. Las salidas se reemplazan por 2 y 6 neuronas con activación softmax para detectar patologías y clasificar enfermedades respectivamente. Los modelos se inicializan con pesos preentrenados y luego se reentrenan. Se alcanza una exactitud de 98.5 % en la detección voces patológicas y de 96.5 % en la clasificación de patologías. También en 2018, Muhammad y Alhamid vuelven a utilizar AP parar la detección de voces patológicas, ahora sobre SVD y MEEI [43]. Comparan dos modelos, VGG-16 y CaffeNet. Las entradas, tomadas de una /a/ sostenida, son espectrogramas filtrados con filtros pasabandas en las frecuencias de mel más la primera y segunda derivada de los mismos. Las capas de salida de los modelos son modificadas para una salida binaria. Nuevamente se entrenan las redes con *transfer learning*. Al final, las capas de salida se reemplazan por SVM (NeuroSVM). Se hacen pruebas entrenando y validando con SVD (exactitud 93.5 % para VGG y 93.9 % para CaffeNet), entrenando con SVD y validando con MEEI (94.5 % y 94.1 % en el mismo orden) y entrenando con MEEI y validando con SVD (73.3 % y 75.1 %).

Otro trabajo relacionado con esta tesis es el de Choi *et al.* en 2017, donde se presenta Kapre[2], una red neuronal que calcula la Transormada Discreta de Fourier (DFT) [44]. Este desarrollo se realizó en paralelo con un desarrollo similar presentado en esta tesis. Kapre está pensado para que la red neuronal realice la DFT como parte del preprocesamiento de audio. En el artículo (*preprint*) se prioriza el análisis de la eficiencia de la red para pesos no ajustables. No se profundiza en los fundamentos del cálculo ni se analiza la capacidad de entrenamiento o adaptación de la red. Si bien este trabajo tiene objetivos y experimentos diferentes, el desarrollo está relacionado con el tema presentado en la sección 4.1.

1.6. Organización de la tesis

En el capítulo 2 se presenta el marco teórico, donde se describen conceptos importantes para la tesis como la producción de la voz, el aprendizaje automático, las redes neuronales, y el aprendizaje profundo.

El capítulo 3, Materiales y Métodos, contiene las definiciones sobre los métodos y métricas de evaluación del modelo de clasificación desarrollado, la descripción de las bases de datos utilizadas en la tesis y el enfoque para el diseño del modelo, donde se explica la generación de un modelo neuronal inicial a partir de modelos más pequeños que resuelven la representación frecuencial del audio, la extracción de características y la clasificación.

[2] https://github.com/keunwoochoi/kapre

El capítulo 4 contiene el desarrollo de la representación frecuencial y el capítulo 5 la extracción de características y clasificación. En el capítulo 6 se presenta el modelo inicial resultante y sus variantes.

En el capítulo 7 se detallan los experimentos realizados para evaluar el rendimiento de los modelos y definir el modelo final. Por último, en los capítulos 8, 9 y 10 se presentan los resultados, la discusión y las conclusiones respectivamente.

Capítulo 2

Marco teórico

2.1. Producción de la voz y calidad vocal

La producción de la voz es un proceso que involucra diversos órganos y funciones. El aparato respiratorio impulsa el aire (energía aerodinámica) que pasa a través de las cuerdas vocales, ubicadas en la laringe. Cuando las cuerdas vocales se cierran se produce una transducción de la energía, que se convierte en sonido (energía acústica) y que posteriormente es modulado en el tracto vocal, donde hay elementos móviles que permiten variaciones: vestíbulo laríngeo, faringe, velo, lengua, labios y maxilar. El sonido final que se escucha es el que resulta de este proceso modulación.

El cierre de las cuerdas vocales y su vibración implican un fenómeno complejo sostenido por propiedades biomecánicas de masa, rigidez y viscoelasticidad de los tejidos de las cuerdas vocales. La estructura de la cuerda vocal es descripta por Titze *et al.* en dos partes funcionalmente diferentes: cuerpo y cubierta [45] (figura 2.1). La cubierta (formada por el epitelio y las capas superficial e intermedia de la lámina propia) es flexible, elástica y no contráctil, mientras que el cuerpo (constituido por la capa profunda de la lámina propia y el músculo vocal) es más rígido y tiene propiedades contráctiles para ajustar la rigidez y concentrar la masa. Durante la contracción del músculo, el cuerpo aumenta su rigidez, mientras que la cubierta se vuelve más laxa y flexible.

El proceso de apertura y cierre de la glotis (pseudoespacio entre las cuerdas vocales, que desaparece cuando estas entran en contacto al fonar) se puede explicar desde las diversas fuerzas que intervienen [46]; la primera es el efecto Bernoulli, que hace que el flujo de aire a través de las cuerdas cree una fuerza que las mueve hacia el centro, luego interviene la elasticidad o retroceso pasivo de las cuerdas vocales para que estas recobren su forma original y, finalmente, el aire escapando a través de la glotis hace que disminuya la

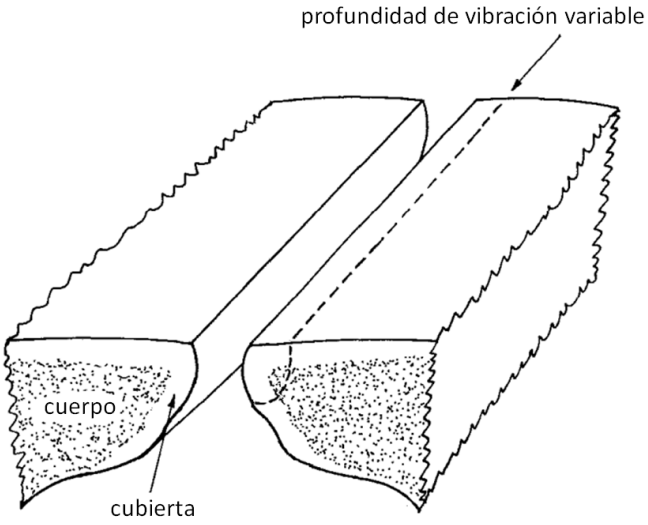


FIGURA 2.1: Cuerdas vocales. Imagen tomada de [45].

presión subglótica y la fuerza que mantiene apartados los tejidos de las cuerdas vocales. Luego las cuerdas se cierren nuevamente hacia su posición de aproximación, para obstruir el flujo de aire e incrementar otra vez la presión subglótica hasta que la presión logra deformar los tejidos de las cuerdas e iniciar otro ciclo, tal como se muestra en la figura 2.2. El ciclo glótico se repite con una frecuencia aproximada de 110 veces por segundo en un hombre sano y de 210 veces por segundo en una mujer. Esta frecuencia de vibración de las cuerdas vocales es la frecuencia fundamental F_0.

Fant, en 1960, describe un modelo denominado fuente-filtro que constituye la base de la teoría acústica de la producción del habla [48]. Dicho modelo describe que las cuerdas vocales son la fuente que genera el sonido y el tracto vocal es el filtro que lo modifica, y a su vez, la configuración que éste adquiera, puede modificar también el comportamiento de la fuente glótica. En su forma más básica, el sistema para producir sonidos de voz hablada consiste solo de la fuente vocal y el tracto, tal como se muestra en la figura 2.3. La señal de la fuente $s(t)$ son los soplos periódicos de aire emitidos a través de las cuerdas vocales. El efecto del tracto vocal está completamente definido por su respuesta al impulso $h(t)$ y la señal de habla $v(t)$ a la salida es igual a la convolución de $s(t)$ y $h(t)$:

$$v(t) = s(t) * h(t)$$

En la producción de la voz hay dos elementos muy importantes, los armónicos (frecuencias generadas junto con la F_0 en la fuente, múltiplos de esta) y los formantes (frecuencias de vibración en el tracto vocal). Estos formantes son modos de vibración que se generan con la configuración del tracto vocal para cada fonema. Hay configuraciones que generan mayor interacción entre el filtro y la fuente, así como hay configuraciones que

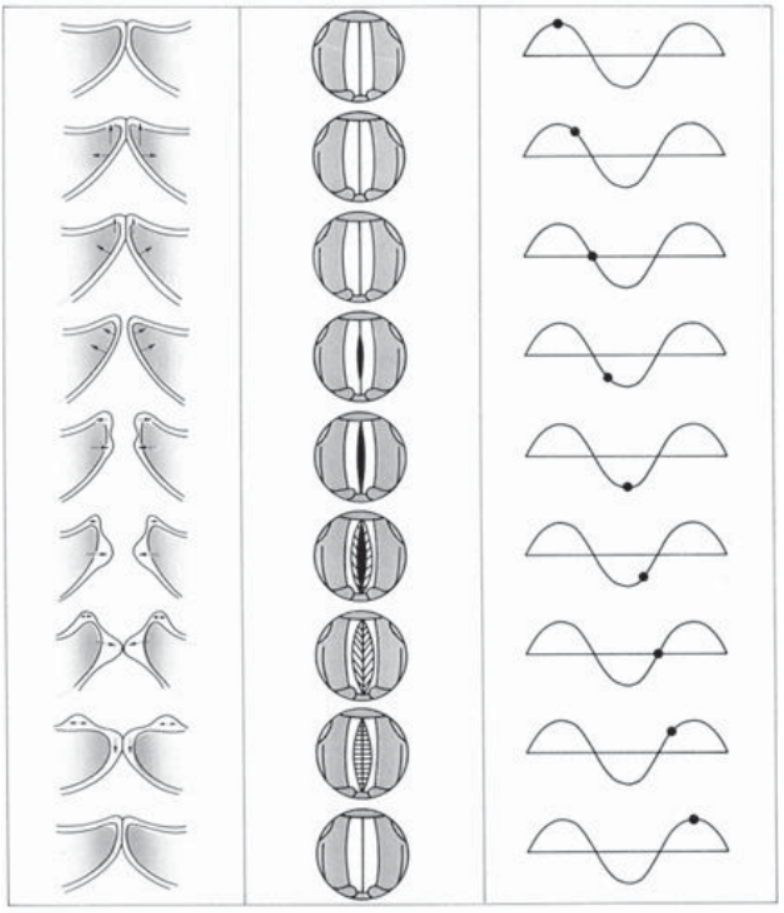


FIGURA 2.2: Esquema del ciclo glótico. En la columna de la izquierda se representa una sección coronal de las cuerdas vocales a lo largo de un ciclo vocal. Al centro se representan las posiciones de la mucosa durante las distintas fases del ciclo vocal tal y como se observarían en una exploración laringoscópica. A la derecha se representa las correspondientes fases del ciclo vocal. Imagen tomada de [47].

permiten mayor o menor amplificación o atenuación de determinadas zonas de armónicos. El aumento de la interacción produce una mayor eficiencia en la producción de la voz y cambios en el patrón de vibración de las cuerdas vocales, como disminución en la duración del cierre, mayor amplitud de vibración, descenso del umbral de presión de la fonación y reducción de la presión transglótica [50], lo que supone una diferencia en la calidad de la voz. El modelo fuente-filtro permite estudiar los problemas en la calidad de la voz separando los efectos de la fuente y del filtro.

2.2. Inteligencia artificial y aprendizaje automático

La inteligencia artificial (IA) es un área de las ciencias de la computación que tiene su origen (oficial) en 1956 durante un taller en Dartmouth organizado por John McCarthy, al que asistieron investigadores interesados en la teoría de autómatas, las redes

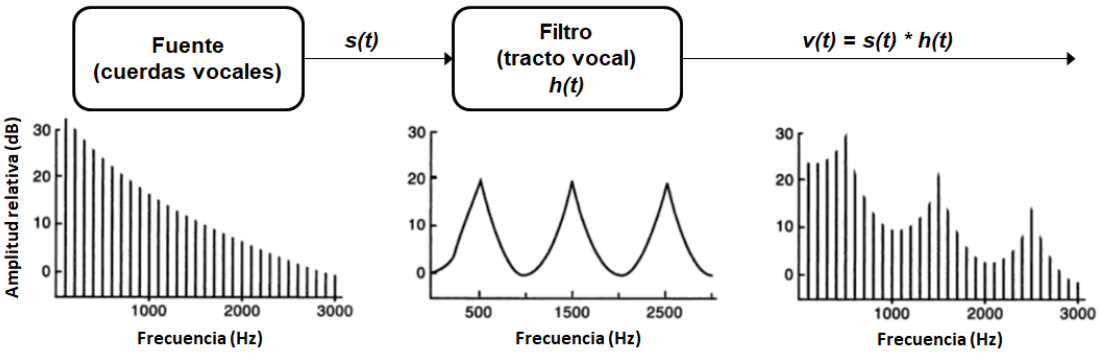


FIGURA 2.3: Esquema fuente-filtro. El gráfico de la izquierda muestra el espectro de la fuente. El gráfico de derecha es el espectro de la fuente después del filtrado por la función de transferencia del tracto vocal (centro). Espectros tomados de [49].

neuronales y el estudio de la inteligencia.

No existe un acuerdo total sobre la definición y los límites de la inteligencia artificial. Algunas de las definiciones varían en torno a dos dimensiones principales, razonamiento-conducta y desempeño humano-ideal. Bajo estas dimensiones, las definiciones se agrupan en cuatro clases (tabla 2.1) [51].

TABLA 2.1: Tipos de definiciones de la inteligencia artificial según las dimensiones razonamiento-conducta y desempeño humano-ideal.

	Desempeño humano-ideal	
Razonamiento-conducta	Sistemas que **piensan** como **humanos**	Sistemas que **piensan racionalmente**
	Sistemas que **actúan** como **humanos**	Sistemas que **actúan racionalmente**

Una definición que, aunque se podría decir que es un tanto incompleta, define con claridad el espíritu de este campo es la de Rich y Knight [52]:

> La Inteligencia Artificial (es la disciplina que) estudia cómo lograr que las máquinas realicen tareas que, por el momento, son realizadas mejor por los seres humanos.

El Aprendizaje Automático es una rama de la inteligencia artificial donde los pasos para la solución del problema no están explícitamente escritos en un programa, sino que los programas "aprenden" o se ajustan a distintos problemas en base a la experiencia.

Mitchell *et al.* definen formalmente el aprendizaje automático de esta manera [53]:

> Se dice que un programa **aprende** de la experiencia E con respecto a algún tipo de tareas S y medida de desempeño P, si su desempeño al realizar la tarea T y medirlo con P, mejora con la experiencia E.

En este contexto, la experiencia sobre algún problema es un conjunto de ejemplos de casos particulares en el que se incluyen las características específicas (entradas) de cada ejemplo y, para un caso de aprendizaje llamado aprendizaje supervisado, también se incluyen las soluciones (salidas).

2.2.1. Reconocimiento de patrones

El tipo de problema de inteligencia artificial que se afronta en este trabajo pertenece a la categoría de reconocimiento de patrones. El concepto de reconocimiento de patrones está relacionado con el reconocimiento humano, el cual se concibe como la estimación del parecido relativo con que los datos de entrada pueden ser asociados con un conjunto de poblaciones estadísticas conocidas, las que dependen de la experiencia pasada y que forman pistas e información *a priori* para el reconocimiento. De este modo, el problema de reconocimiento puede ser concebido como el de discriminar, clasificar o categorizar la información de entrada, no entre patrones individuales sino entre poblaciones, por medio de la búsqueda de características o atributos invariantes entre los miembros de una población [54].

Como se puede ver, los conceptos de aprendizaje automático y reconocimiento de patrones están muy relacionados y hasta son utilizados como sinónimos en muchas ocasiones. Una distinción útil es que el reconocimiento de patrones divide la solución del problema en tres etapas (figura 2.4) muy bien definidas, el sensado, la extracción de características y la predicción, mientras que el aprendizaje automático se centra en las dos últimas.

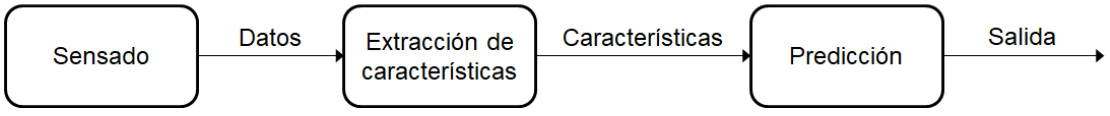


FIGURA 2.4: Etapas del reconocimiento de patrones. La salida puede ser una categoría o un valor numérico continuo.

Sensado. La etapa de sensado se refiere a la medición y representación de la información obtenida. Por ejemplo, en el caso del audio, la medición se realiza con un sensor o micrófono que capta las variaciones de presión en el aire y las convierte en una diferencia de potencial eléctrico. Mediante un proceso de muestreo sobre la señal eléctrica se registra una secuencia de valores continuos que indican la intensidad del sonido en distintos instantes de tiempo. Finalmente, esta secuencia de valores se representa en un vector donde cada elemento es la intensidad para un tiempo determinado.

Extracción de características. En la etapa de extracción de características se parte desde el conjunto inicial de mediciones generado en la etapa de sensado y se produce un conjunto de valores derivados comunmente llamados características o *features*. Las características obtenidas deben ser informativas (aportar información útil para asociar un caso particular a una de las poblaciones conocidas) y no redundantes para facilitar la etapa de clasificación. Frecuentemente se busca que la salida de esta etapa tenga una dimensión menor a la de los datos sensados. La reducción de la dimensión se puede dar de una manera natural, como en el ejemplo a continuación, o utilizando alguna técnica específica como el análisis de componentes principales (PCA por su sigla en inglés). En el caso del audio, las características podrían ser algunas medidas acústicas calculadas sobre el vector que contiene las intensidades en el tiempo. *Shimmer* por ejemplo, es un indicador de las perturbaciones de amplitud que tiene una señal. La utilización de *shimmer* como una de las características para clasificar el audio produciría dos efectos: Por un lado, el clasificador recibiría información sobre las perturbaciones de amplitud que es difícil de obtener sobre la señal sin procesar. Por el otro lado, al calcular *shimmer* sobre un audio de, por ejemplo, un segundo muestreado a 25 KHz, se obtiene un escalar, entonces, si *shimmer* fuera la única característica utilizada, la dimensión se reduciría de 25000 a 1.

Predicción. Por último, en la etapa de predicción hay una medida de salida, la cual puede ser cuantitativa (como volumen de ventas, temperatura, etc.) o categórica (como fraude/no fraude, ganador_A/empate/ganador_B, etc.), y que se desea predecir en base al conjunto de características obtenidas en la etapa anterior. Se cuenta con un conjunto de datos de entrenamiento, en el cual se observan la salida y las características para un grupo de objetos (como voces, personas, etc.). En el aprendizaje supervisado, se utilizan estos datos para construir un modelo de predicción capaz de predecir la salida para nuevos objetos nunca antes vistos. Un buen modelo de predicción es aquel que predice la salida con gran exactitud. Cuando la medida de salida es una variable categórica, en lugar de predicción usualmente se llama clasificación a esta etapa, cuando la salida es una variable continua, usualmente se la llama regresión [55].

2.2.2. Aprendizaje supervisado y no supervisado

Se ha mencionado el aprendizaje supervisado, donde el modelo de predicción se ajusta en base al conocimiento *a priori*. El nombre "supervizado" proviene de una metáfora donde el alumno (modelo) aprende gracias a las correcciones del maestro. Formalmente, se quiere predecir los valores de una o más variables de salida $Y = [Y_1, \ldots, Y_m]$ para un conjunto dado de variables de entrada $X^\intercal = [X_1, \ldots, X_n]$. Si $\boldsymbol{x}_i^\intercal = [x_{i1}, \ldots, x_{in}]$ son las entradas del i-ésimo ejemplo de entrenamiento y $\boldsymbol{y}_i$ es el vector de salida esperado, las predicciones estarán basadas en las muestras de entrenamiento $(\boldsymbol{x}_1, \boldsymbol{y}_1), \ldots, (\boldsymbol{x}_L, \boldsymbol{y}_L)$, donde L es la cantidad de ejemplos de entrenamiento. Continuando con la metáfora, el "alumno" presenta una respuesta $\boldsymbol{y}'_i$ para cada $\boldsymbol{x}_i$ del conjunto de entrenamiento, y el supervisor o "maestro" provee la respuesta correcta (salida esperada) o la desviación asociada con la respuesta del alumno. Esta desviación es usualmente representada por una función error (*loss*) $E(\boldsymbol{y}, \boldsymbol{y}')$, por ejemplo, $E(\boldsymbol{y}, \boldsymbol{y}') = (\boldsymbol{y} - \boldsymbol{y}')^2$ [55].

Existe otro tipo de aprendizaje, el no supervisado, donde los modelos se ajustan sin un valor de salida esperada. En el aprendizaje no supervisado, el objetivo es inferir las propiedades de la densidad marginal conjunta de los elementos del vector de características [55]. Este tipo de aprendizaje no será utilizado en el presente trabajo.

2.2.3. Clasificación

Suponiendo un problema de reconocimiento de m clases distintas denominadas $\omega_1, \ldots, \omega_m$, puede considerarse al espacio de las entradas compuesto por m regiones, donde cada una de las cuales contiene a los elementos de una clase. La solución puede interpretarse como la generación de límites de decisión entre las regiones. Estos límites pueden estar dados por funciones de decisión o funciones discriminantes $d_1(\boldsymbol{x}), \ldots, d_m(\boldsymbol{x})$, que son funciones escalares de los vectores de entrada. La función que obtiene el mayor valor es la que determina la pertenencia del vector a la clase correspondiente a esa función, es decir, si $d_i(\boldsymbol{x}) > d_j(\boldsymbol{x})$ para $i, j = 1, \ldots, m$ y $i \neq j$, entonces $\boldsymbol{x}$ pertenece a la clase ω_i.

En los problemas, muy frecuentes, donde la salida solo puede pertenecer a una clase de un total de dos clases (clasificación binaria), usualmente se utiliza una única función de decisión $d(\boldsymbol{x})$. Si $d(\boldsymbol{x} > 0)$, $\boldsymbol{x}$ pertenece a una clase, en caso contrario, a la otra.

2.2.3.1. Clasificador lineal

El caso más simple de función de decisión es la función de decisión lineal. El modelo de clasificación que implementa esta función se llama clasificador lineal. Para un vector

de entradas $\boldsymbol{x}^\intercal = [x_1, \ldots, x_n]$, la función de decisión lineal es $d(\boldsymbol{x}) = w_1 x_1 + \cdots + w_n x_n + w_{n+1}$, donde los coeficientes $w_i \in \mathbb{R}$ son los parámetros del clasificador. El vector formado por todos los coeficientes se llama vector de parámetros $\boldsymbol{w} = [w_i, \ldots, w_{n+1}]$. Es útil calcular el valor de la función de decisión como un producto entre vectores, para lo cual el vector $\boldsymbol{x}$ se redefine como $\boldsymbol{x}^\intercal = [x_1, \ldots, x_n, 1]$, lo que lleva el nombre de vector de entradas aumentado. Con el nuevo vector $\boldsymbol{x}$, la función de decisión se puede calcular como $d(\boldsymbol{x}) = \boldsymbol{w}\boldsymbol{x}$.

Para el caso una clasificación binaria, $\boldsymbol{x}$ pertenecerá la clase ω_1 si $d(\boldsymbol{x}) < 0$ y pertenecerá a la clase ω_2 si $d(\boldsymbol{x}) > 0$. Entonces, la frontera entre las clases en el espacio de las entradas queda definida por el hiperplano $d(\boldsymbol{x}) = 0$. En la figura 2.5 se muestra un ejemplo de clasificador lineal. Los elementos de la clase ω_1 están pintados de color amarillo y los de ω_2 en color azul. La función de decisión se ve como un plano inclinado y la frontera entre las clases se indica con la línea de puntos.

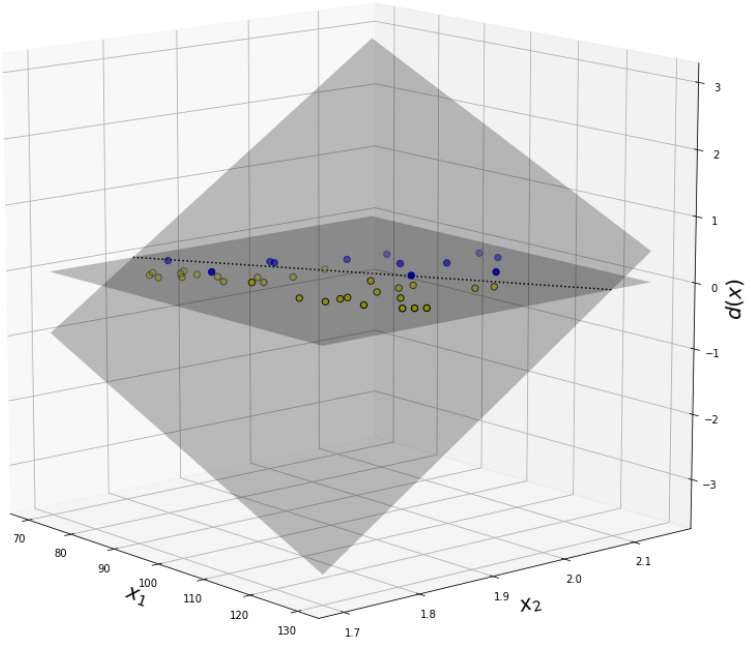


FIGURA 2.5: Ejemplo de función de decisión lineal en el espacio de las entradas.

El entrenamiento del modelo se realiza ajustando los valores del vector de parámetros $\boldsymbol{w}$. En primer lugar, se crea un vector $\boldsymbol{y} = [y_1, \ldots, y_L]$ con las salida esperada para cada vector de entradas, donde $y_j = -1$ para $\boldsymbol{x}_j \in \omega_1$ y $y_j = 1$ para $\boldsymbol{x}_j \in \omega_2$. Después, se calcula el valor óptimo de $\boldsymbol{w}$ como la combinación de parámetros que minimiza la función del error cuadrático medio $mse = \frac{1}{L}\sum_{j=1}^{L}(y_j - \boldsymbol{w}\boldsymbol{x}_j)^2$. Partiendo de la derivada del error cuadrático medio con respecto a los parámetros $\frac{\partial mse}{\partial \boldsymbol{w}}$, se obtiene la expresión del vector de parámetros óptimo $\boldsymbol{w} = (XX^\intercal)^{-1}X\boldsymbol{y}^\intercal$, donde X es la matriz formada por los L vectores de entrada.

Para un caso simple, donde el vector de entradas (no aumentado) tiene dimensión 2, el espacio formado por las entradas y la salida tendrá dimensión 3 (como en la figura 2.5), el hiperplano discriminante dimensión 2 y la frontera entre las clases, calculada a partir de $d(\boldsymbol{x}) = 0$ será $x_2 = -\frac{w_1}{w_2}x_1 - \frac{w_3}{w_2}$, una línea recta (dimensión 1). Dado que la frontera entre las clases es una línea recta, este clasificador solo puede separar clases linealmente separables. Esta es la gran limitación del clasificador lineal, ya que la mayoría de los problemas reales no son linealmente separables.

Existen varias estrategias para lidiar con los problemas no-linealmente separables. El enfoque que se deriva del clasificador lineal es la función de decisión generalizada, donde la función de decisión se define como $d(\boldsymbol{x}) = w_1 f_1(\boldsymbol{x}) + \cdots + w_2 f_q(\boldsymbol{x}) + w_{q+1}$. Cada $f_i(\boldsymbol{x})$ es una función real simple del vector de entradas completo. La última ecuación representa una infinita variedad de funciones de decisión, dependiendo de la elección de las $f_i(\boldsymbol{x})$ y de la cantidad de términos utilizados. Para el cálculo de los parámetros, primero se realiza una transformación del vector de entradas. El vector utilizado, $\boldsymbol{x}^* = [f_1(\boldsymbol{x}), \ldots, f_q(\boldsymbol{x}), 1]^\intercal$, está formado por los valores de las $f_i(\boldsymbol{x})$ previamente calculados, entonces el cálculo de los coeficientes se reduce a la misma expresión que la del clasificador lineal.

2.2.3.2. Hiperparámetros

Las coeficientes $\boldsymbol{w}$ de la función de decisión son los parámetros que se ajustan durante el aprendizaje. Todos los modelos de aprendizaje automático tienen un conjunto de parámetros para ajustar, en las redes neuronales por ejemplo, son los pesos sinápticos. La elección de los parámetros determina la calidad de la respuesta de un modelo. También existe un conjunto de decisiones de mayor nivel que afecta el rendimiento de los modelos, estos son los hiperparámetros. Los hiperparámetros representan las decisiones de diseño de los modelos, por ejemplo las funciones y la cantidad de términos utilizados en un clasificador basado en el concepto de la función de decisión generalizada. En el caso de las redes neuronales, estos son la cantidad y tipo de capas, la cantidad de neuronas de cada capa, las funciones de activación, la tasa de aprendizaje, etc.

2.2.4. Evaluación

La habilidad para predecir correctamente nuevos ejemplos que difieren de aquellos utilizados durante el entrenamiento es conocida como generalización. En las aplicaciones prácticas, la variabilidad de los vectores de entrada es tal que los datos de entrenamiento comprenden solo una pequeña fracción de todos los vectores de entrada posibles, por lo que la generalización es un objetivo central en el reconocimiento de patrones [56]. Para

medir la calidad de las predicciones de un modelo y su capacidad de generalización, es necesario definir métricas de rendimiento y procedimientos de evaluación.

2.2.4.1. Métricas

La valoración del rendimiento de los modelos con aprendizaje supervisado se lleva a cabo mediante la comparación de las salidas (predicciones) del modelo para un conjunto de datos de entrada y las salidas esperadas para los mismos datos.

En el caso de la predicción de valores continuos o regresión, la comparación entre las salidas está basada en la magnitud de la diferencia. Las métricas más frecuentes son el error cuadrático medio (usualmente llamado MSE por su sigla en inglés), la raíz del error cuadrático medio (RMSE) y el error absoluto medio (MAE).

$$\text{MSE} = \frac{1}{L}\sum_{j=1}^{L}(y - y')^2 \tag{2.1}$$

$$\text{RMS} = \sqrt{\frac{1}{L}\sum_{j=1}^{L}(y - y')^2} \tag{2.2}$$

$$\text{MAE} = \frac{1}{L}\sum_{j=1}^{L}|y - y'| \tag{2.3}$$

La métrica RMS es preferible a MSE porque mide en la misma unidad en que se expresa la salida [56]. Lo mismo pasa con MAE.

Para el caso de la clasificación, la métrica más utilizada es la exactitud o *accuracy* [57]. La exactitud se calcula como la relación entre la cantidad de casos bien clasificados y la cantidad total de los casos L.

$$\text{exactitud} = \frac{1}{L}\sum_{j=1}^{L}\gamma(y_j, y'_j) \tag{2.4}$$

donde

$$\gamma(y_j, y'_j) = \begin{cases} 0 & \text{si} \quad y_j \neq y'_j \\ 1 & \text{si} \quad y_j = y'_j \end{cases}$$

Para modelos de clasificación es frecuente contabilizar las ocurrencias de cada una de las combinaciones posibles de (y_j, y'_j) para formar una tabla llamada matriz de confusión. La matriz de confusión es una tabla de contingencia que permite visualizar y calcular información más detallada sobre el rendimiento del modelo. En los casos de clasificación binaria, los nombres de las clases utilizadas suelen ser "positivo" o "P" y "negativo" o "N", indicando si se está en presencia o no de una situación que se pretende detectar, como por ejemplo una patología determinada. En la tabla 2.2 se muestra un ejemplo donde, sobre 190 ejemplos, hubo 100 verdaderos positivos (VP), es decir 100 casos positivos que fueron predichos como positivos, 10 falsos positivos (FP), es decir casos negativos que fueron predichos como positivos, 0 falsos negativos (FN) y 80 verdaderos negativos (VN).

Tabla 2.2: Ejemplo de matriz de confusión para clasificación binaria.

		Predicción	
		P	N
Real	P	100	0
	N	10	80

En la tabla 2.2 (y en el resto del trabajo) se utilizaron las filas para ubicar las ocurrencias de las clases reales y las columnas para las predicciones, pero en la literatura se puede encontrar también la distribución contraria.

Partiendo de una matriz de confusión como la de la tabla 2.2, la exactitud se puede calcular como:

$$\text{exactitud} = \frac{VP + VN}{VP + VN + FP + FN}$$

Además de la exactitud, la matriz de confusión de un caso de clasificación binaria, se puede utilizar para calcular otras métricas útiles para la evaluación de un modelo, como por ejemplo la sensibilidad y la especificidad. En base a estas métricas se suele realizar un gráfico llamado curva ROC, donde se puede ver cómo reacciona el modelo cuando se le ponderan los errores (FP y FN). Si bien estos indicadores de calidad son importantes, no se pueden aplicar en este trabajo porque la clasificación del grado general de disfonía no es binaria, por lo tanto no se profundiza en ellos.

Para los casos de clasificación multiclase, la matriz de confusión contiene los nombres de cada clase en lugar de "P" y "N". La tabla 2.3 muestra un ejemplo para las clases "A", "B", "C" y "D".

TABLA 2.3: Ejemplo de matriz de confusión para clasificación multiclase.

		Predicción			
		A	B	C	D
Real	A	10	5	4	1
	B	11	11	7	6
	C	2	6	1	8
	D	5	3	9	13

Con la matriz de confusión de la tabla 2.3, la exactitud se puede calcular como el cociente entre la suma de los elementos de la diagonal principal y la suma de todos los elementos de la tabla.

2.2.4.2. Sobreajuste

Un problema habitual en aprendizaje automático es que los modelos se ajusten a los datos de entrenamiento en lugar de generalizar. Este comportamiento se llama sobre-entrenamiento o sobreajuste (*overfitting*). Por un lado, para resolver problemas reales los modelos de predicción usualmente necesitan un alto grado de complejidad, es decir, muchos grados de libertad en la función de decisión. Por otro lado, un modelo complejo es propenso al sobreajuste.

En la figura 2.6 se puede ver un ejemplo de sobreajuste utilizando un problema de regresión. El objetivo es la función $t = \sin(2\pi x)$ y el modelo de predicción utiliza un polinomio de grado M. Se muestra el mejor ajuste de los polinomios de grado $M = 0, 2, 3$ y 9 a los datos de entrada, que son los puntos marcados con círculos azules (muestreados sobre t y con ruido añadido). Si se observan las curvas completas (todos los puntos), claramente el polinomio de grado $M = 3$ es el que mejor se ajusta a la curva, aunque coincide de forma casi exacta con pocos valores del vector de entradas. El polinomio de grado $M = 9$ se ajusta de forma perfecta a los datos, pero la curva obtenida oscila fuertemente y el ajuste a t es muy pobre.

Los factores que inducen al sobreajuste son principalmente la complejidad del modelo, la cantidad (pequeña) de datos y el tiempo de entrenamiento. En la siguiente sección se verá cómo tener en cuenta el sobreajuste durante la evaluación del modelo y en la sección 2.2.6 se volverá sobre el tema para presentar algunas técnicas que permiten lidiar con el problema.

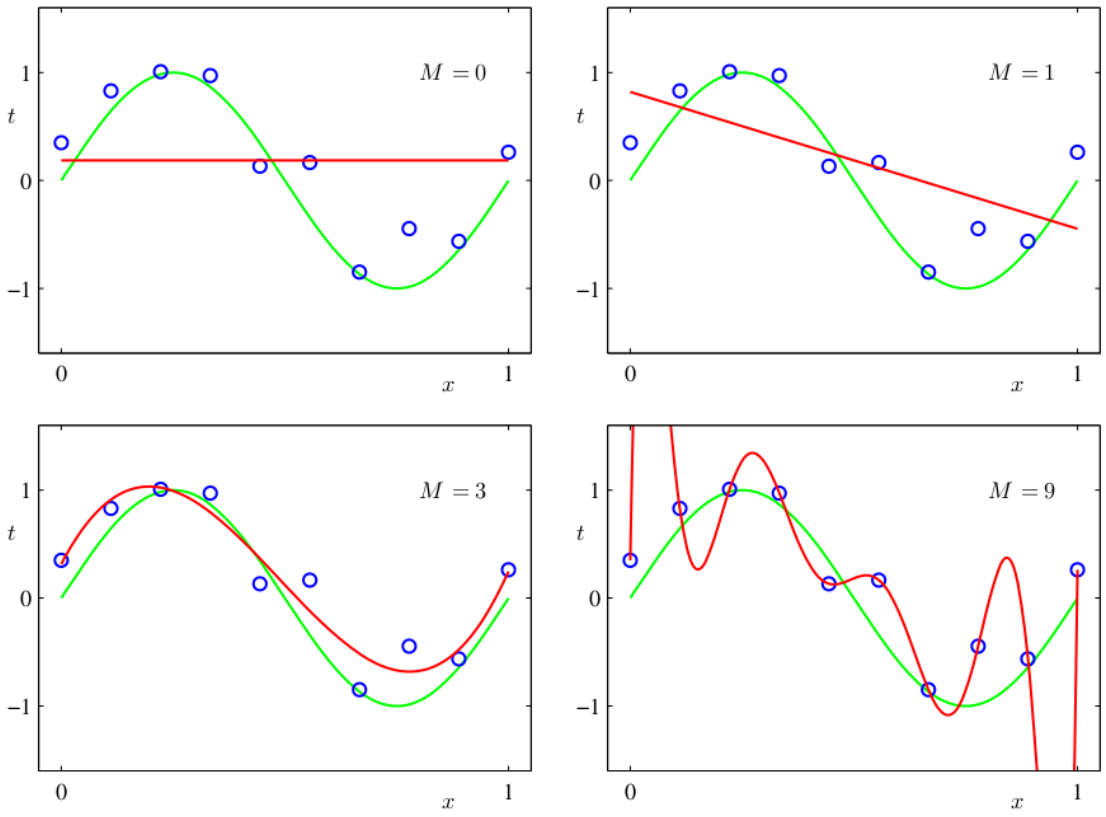


FIGURA 2.6: Ejemplo de sobreajuste con el ajuste de un polinomio de grado M a la curva de la función $t = \sin(2\pi x)$ [56].

2.2.4.3. Esquemas de evaluación

El nivel de generalización de un modelo de predicción está relacionado con la capacidad de predicción sobre un conjunto de datos independiente. Típicamente un modelo tiene hiperparámetros que varían su complejidad y se desea encontrar el conjunto de hiperparámetros que minimice el error de predicción. Es importante notar que existen en realidad dos objetivos distintos que se debe tener en mente:

Selección del modelo: estimación del rendimiento de distintos modelos (distintos conjuntos de hiperparámetros) con el objetivo de elegir el mejor.

Evaluación del modelo: estimación del rendimiento, sobre datos nuevos, del modelo elegido.

El mejor enfoque para ambos problemas, aunque depende de la cantidad de datos con que se cuenta, es dividir aleatoriamente los datos en tres conjuntos, un juego de datos de entrenamiento, uno de validación y uno de test. Los datos de entrenamiento son utilizados para ajustar los parámetros de los modelos, los datos de validación se utilizan en la estimación del rendimiento de cada modelo para la elección del mejor y los datos de test se usan para evaluar el rendimiento del modelo final elegido. El conjunto de test

se debe mantener separado y ser utilizado solo al final del proceso, en caso contrario, se puede producir una adaptación de los hiperparámetros a estos datos [55].

Frecuentemente la cantidad de datos para entrenamiento y pruebas es limitado y, para construir buenos modelos, se desea utilizar en el entrenamiento la mayor cantidad posible de datos disponibles. Sin embargo, si el conjunto de validación es pequeño, dará una estimación poco confiable de la calidad de la predicción. Una solución a este dilema es utilizar la validación cruzada [56].

Validación cruzada El método de validación cruzada o *k-fold* permite utilizar todos los datos de entrenamiento y validación tanto para entrenar como para validar el modelo. Los datos destinados a este fin se dividen aleatoriamente en k subconjuntos o *folds* del mismo tamaño. La validación se realiza en k iteraciones. En cada iteración se elige un *fold* para validación y el resto para entrenamiento. El rendimiento se calcula como la media de los k rendimientos obtenidos.

En la figura 2.7 se muestra el esquema de entrenamiento, validación y test utilizando validación cruzada para $k = 5$. En la parte superior de la figura se representa la división del conjunto total de datos en dos subconjuntos llamados entrenamiento/validación y test. El primer subconjunto se divide en 5 *folds* y con estos se forman las 5 combinaciones (*splits*) de conjuntos de entrenamiento y validación, donde los datos de validación se muestran en color celeste y los de entrenamiento en gris claro. Los 5 *splits* se utilizan durante el proceso de selección del modelo, mientras que los datos reservados para el test se usan en la evaluación del modelo final.

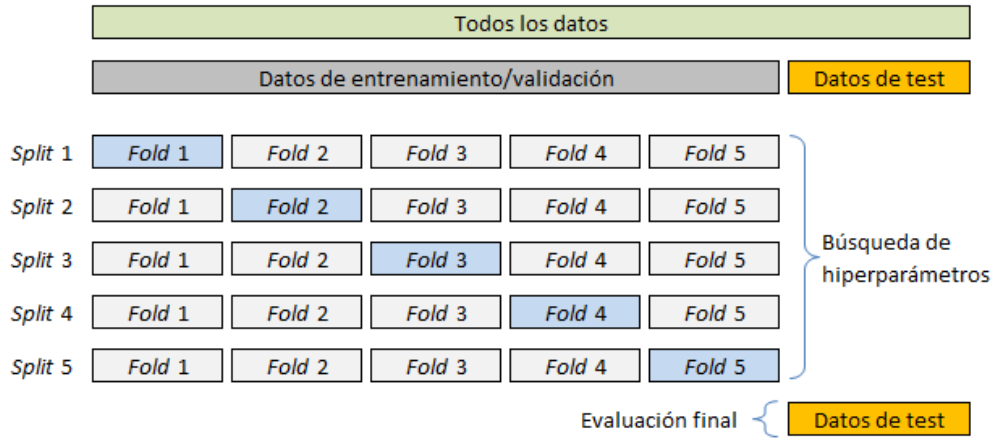


FIGURA 2.7: Esquema de división de datos en subconjuntos de entrenamiento/validación y test con aplicación posterior de validación cruzada sobre los datos de entrenamiento/validación.

2.2.5. Redes neuronales artificiales

Al intentar construir máquinas inteligentes surge de forma natural un modelo: la mente humana. Por lo tanto, una idea obvia en el campo de la inteligencia artificial es la de simular directamente el funcionamiento del cerebro en una computadora [52]. Desde la perspectiva de las aplicaciones prácticas del reconocimiento de patrones, el "realismo biológico" impondría restricciones completamente innecesarias [56], por lo tanto, si bien las redes neuronales artificiales están inspiradas en el modelo biológico, se debe pensar en ellas como modelos que extraen combinaciones lineales de las entradas, convirtiéndolas en características (*features*) derivadas y después modelan la salida como una función no lineal de esas características [55]; no como una simulación del cerebro.

2.2.5.1. Perceptron

El perceptron fue ideado por Frank Rosenblatt en 1958 y es el primer modelo de neurona artificial. Es un método de aprendizaje supervisado que realiza una clasificación binaria en base a una transformación lineal, al igual que el clasificador lineal presentado en la sección 2.2.3.1.

El perceptron representa una neurona biológica donde las dendritas son las entradas del perceptron, el axón es la salida, las sinapsis son los coeficientes de la función de decisión y el comportamiento (muy simplificado) se replica acumulando la intensidad de los impulsos recibidos que, al superar cierto umbral, "activan" la neurona emitiendo una señal por la salida. En la figura 2.8

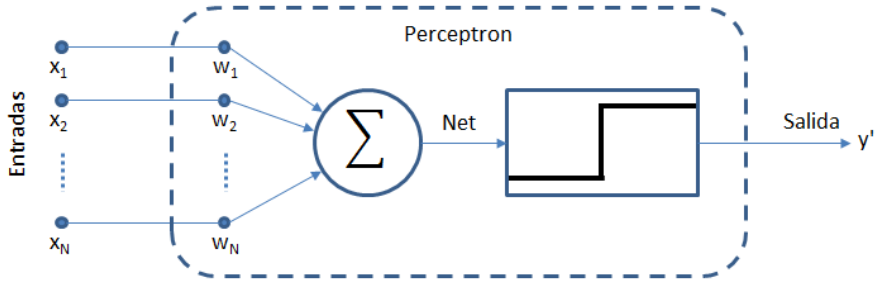


FIGURA 2.8: Esquema del perceptron.

Para el vector de entrada $\boldsymbol{x} = [x_1, x_2, ..., x_N]$, un vector de pesos $\boldsymbol{w} = [w_1, w_2, ..., w_N]$, el umbral de activación θ y la función umbral (también *threshold* o *hard-lim*) f, la salida y' del perceptron se calcula como:

$$y' = f\left(\left(\sum_{i=1}^{N} w_i x_i\right) - \theta\right) \tag{2.5}$$

Es común llamar Net a la suma ponderada de las entradas $\sum_{i=1}^{N} w_i x_i$ para simplificar las expresiones. Entonces, en la ecuación 2.5:

$$f(Net, \theta) = \begin{cases} 0 & \text{si} \quad Net - \theta < 0 \\ 1 & \text{si} \quad Net - \theta \geq 0 \end{cases}$$

Para hacer más simple el cálculo, usualmente se incluye el umbral θ dentro de Net añadiendo un nuevo peso sináptico $w_0 = -\theta$. Para esto se redefine el vector de entrada como $\boldsymbol{x} = [1, x_1, x_2, ..., x_N]$ con el mismo criterio utilizado al aumentar el vector de entrada del clasificador lineal. A partir de estos cambios, la salida se calcula como:

$$y' = f\left(\sum_{i=0}^{N} w_i x_i\right) \tag{2.6}$$

donde

$$f(Net) = \begin{cases} 0 & \text{si} \quad Net < 0 \\ 1 & \text{si} \quad Net \geq 0 \end{cases}$$

En la figura 2.9 se muestra el esquema más usual del perceptron, donde se reflejan los cambios realizados sobre el umbral. Notar la entrada fija en 1 del peso w_0.

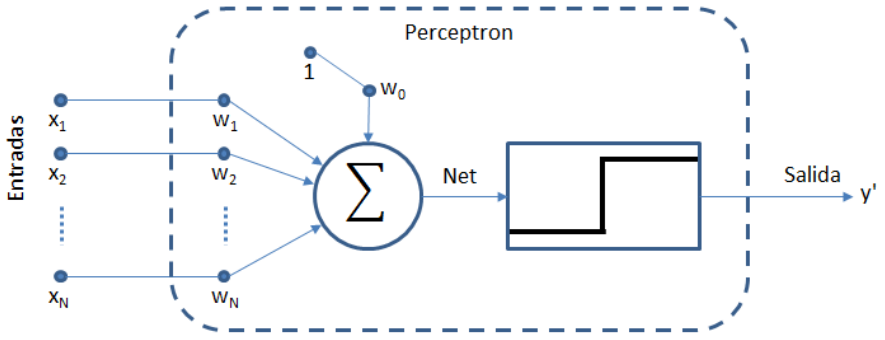


FIGURA 2.9: Esquema del perceptron para $w_0 = -\theta$ y vector de entrada aumentado.

El ajuste de los pesos del perceptron se realiza con la siguiente *ley de aprendizaje*:

$$\Delta_{w_i} = \alpha(y - y')x_i \tag{2.7}$$

donde y es la salida esperada y α es la tasa de aprendizaje. El proceso es el siguiente:

- Inicializar los pesos con valores aleatorios.
- Mientras $error > 0$ para algún vector de entradas, hacer:

- Ejecutar ciclo completo de entrenamiento (*epoch*). Mientras existan vectores de entrenamiento hacer:
  - Tomar un vector entrada/salida del conjunto de datos de entrenamiento.
  - Calcular la salida $y' = f\left(\sum_{i=0}^{N} w_i x_i\right)$
  - Calcular el $error = y - y'$
  - Calcular $\Delta_{w_i} = \alpha(y - y')x_i$
  - Modificar los pesos haciendo $w_{i_{t+1}} = w_{i_t} + \Delta_{w_i}$

Notar que tal como está declarado el método, y a menos que se defina una cantidad máxima de ciclos, el entrenamiento termina solo cuando todos los vectores están bien clasificados.

El perceptron tiene la misma desventaja que el clasificador lineal, solo es capaz de clasificar correctamente las entradas de problemas linealmente separables. El problema clásico para ejemplificar las clases no-linealmente separables es el de la compuerta XOR (figura 2.10), donde resulta obvio que una linea recta no puede separar los puntos (0,0) y (1,1) de los puntos (0,1) y (1,0) en el espacio de las entradas.

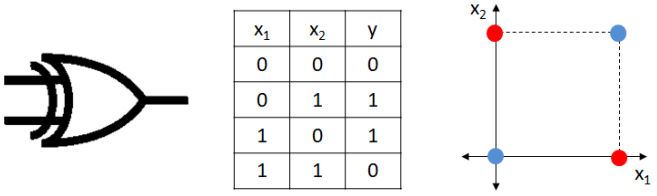


FIGURA 2.10: Problema de clases no-linealmente separables ejemplificado con la compuerta XOR.

El análisis de la compuerta XOR es interesante porque permite vislumbrar la solución para que las redes neuronales, a pesar de realizar transformaciones lineales en su interior, puedan resolver problemas de clases no-linealmente separables.

Una compuerta XOR se puede construir con otras compuertas, por ejemplo las del circuito lógico de la figura 2.11.

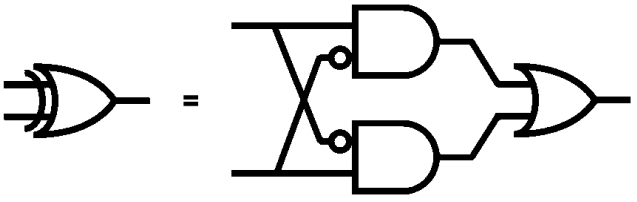


FIGURA 2.11: Uno de los circuitos lógicos posibles para crear una compuerta XOR a partir de compuertas AND, OR y negadores.

Si tres perceptrones (P1, P2 y P3) se entrenan, por separado, para funcionar como cada una de las compuertas del circuito de la figura 2.11 y después se conectan como en la figura 2.12, se podría predecir el comportamiento de la compuerta XOR.

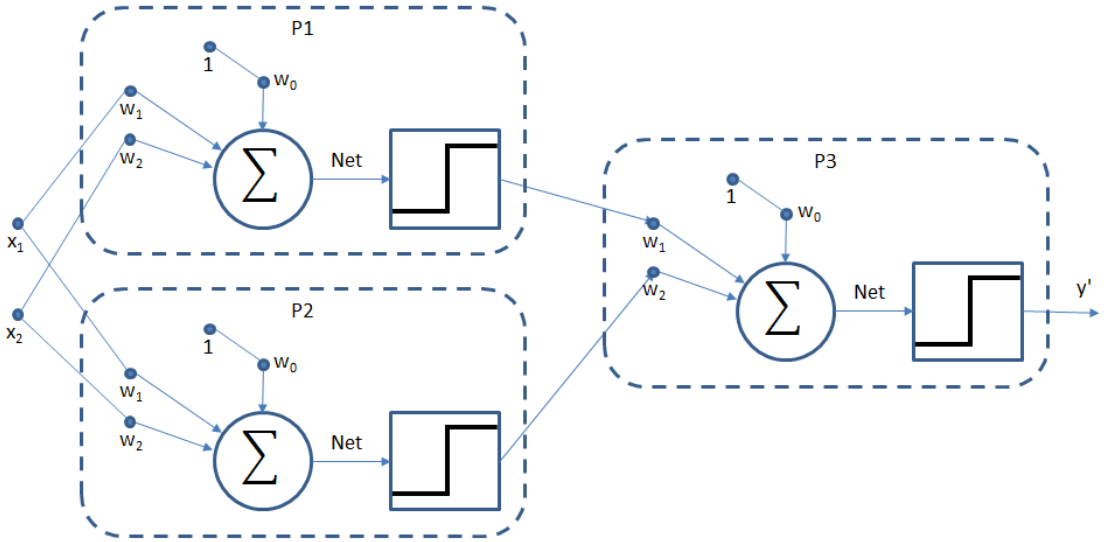
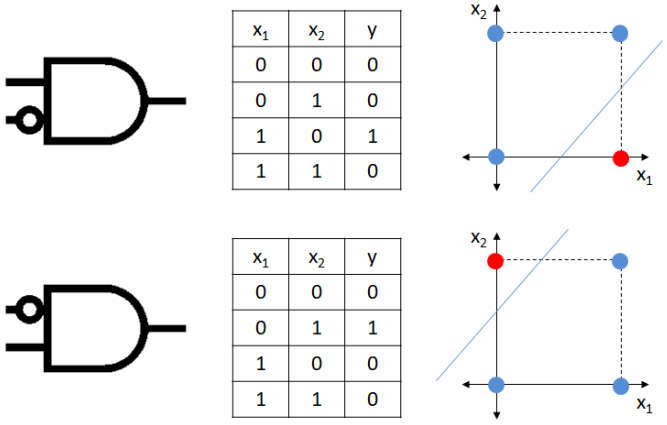


FIGURA 2.12: Conexión de tres perceptrones para predecir la salida de una compuerta XOR.

En la figura 2.13 se muestran las tablas de verdad de las compuertas C1 (AND con la segunda entrada negada) y C2 (AND con la primera entrada negada) y las fronteras entre clases en el espacio de las entradas para los perceptrones P1 y P2 entrenados. Es fácil ver que una linea recta puede resolver ambos casos porque para las dos compuertas solo una de las combinaciones pertenece a la clase de salida "1".



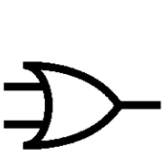
x_1	x_2	y
0	0	0
0	1	0
1	0	1
1	1	0

x_1	x_2	y
0	0	0
0	1	1
1	0	0
1	1	0

FIGURA 2.13: Clasificación de la salida de las compuertas C1 con el perceptron P1 (arriba) y de la compuerta C2 con el perceptron P2 (abajo).

El perceptron P3 se entrena para funcionar como una compuerta OR, lo que también se puede lograr fácilmente porque una sola de sus salidas pertenece a la clase "0". En la figura 2.14 se puede ver la tabla de verdad de la compuerta OR incluyendo las verdaderas

entradas de la compuerta C3, que son las salidas de C1 y C2. El perceptron P3 funciona como una compuerta OR, por lo tanto no podría predecir la salida de una compuerta XOR a partir de las entradas x_1 y x_2, pero sí lo puede hacer en el nuevo espacio c_1 - c_2 formado por las salidas intermedias del circuito de la figura 2.12. La función de activación escalón introduce una no-linealidad que evita que las sucesivas transformaciones lineales se conviertan en una sola transformación lineal.



x_1	x_2	c_1	c_2	y
0	0	0	0	0
0	1	0	1	1
1	0	1	0	1
1	1	0	0	0

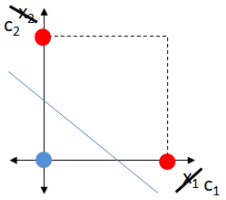


FIGURA 2.14: Clasificación de la salida de la compuerta C3 con el perceptron P3. En la tabla de verdad, la salida y se calcula para las entradas c_1 y c_2, no para x_1 y x_2.

Como conclusión, una red de perceptrones es capaz de predecir la salida de una compuerta XOR si utilizamos el conocimiento del dominio del problema para determinar el valor de los pesos sinápticos. Esta conclusión se puede extender a cualquier problema con clases no-linealmente separables.

La conclusión del párrafo anterior es cierta, pero no implica que una red formada por perceptrones sea capaz de resolver este tipo de problemas en casos reales porque no es posible entrenarla. Si se cuenta con conocimiento del dominio del problema, sencillamente no tiene sentido utilizar una red neuronal para resolverlo. Si no se cuenta con este conocimiento, tampoco se conoce la salida esperada para los perceptrones P1 y P2, por lo tanto, no es posible ajustar sus pesos. El desafío es encontrar un método que permita ajustar los parámetros a pesar de no conocerse los valores esperados de las salidas intermedias.

Se suele generar una confusión relacionada con este tema. Uno de los modelos más exitosos de aprendizaje automático son las redes neuronales con conexiones hacia adelante (*feed-forward neural networks*), también llamadas perceptrones multicapa (*multilayer perceptrons*). En realidad, "perceptron multicapa" es un nombre inapropiado, porque el modelo está compuesto por múltiples capas de modelos de regresión logística (con no-linealidades continuas), en lugar de perceptrones (con no-linealidades discontinuas) [56].

2.2.5.2. Adaline

Adaline (*ADAptative LINear Element*) es un modelo de red neuronal artificial que tiene dos diferencias con respecto al perceptron.

La primera diferencia es la función de activación, en Adaline se utiliza una función lineal. La salida de Adaline es directamente $y' = Net$. Si este modelo se aplica a una predicción del tipo regresión (cosa imposible para el perceptron), la salida utilizada es Net. En el caso de la clasificación, será necesario utilizar una función escalón en algún momento. En este punto, hay distintos criterios en la bibliografía. Algunos autores plantean que la función de activación de Adaline es lineal y, fuera del modelo, se aplica una función umbral para determinar la clase. Otros autores plantean que la salida de Adaline es binaria, pero que el error para la modificación de los pesos se calcula antes de la función de activación. Si bien el resultado es el mismo, esta diferencia suele generar confusiones. En este trabajo se utiliza el primer enfoque. En la figura 2.15 se muestra el esquema del modelo.

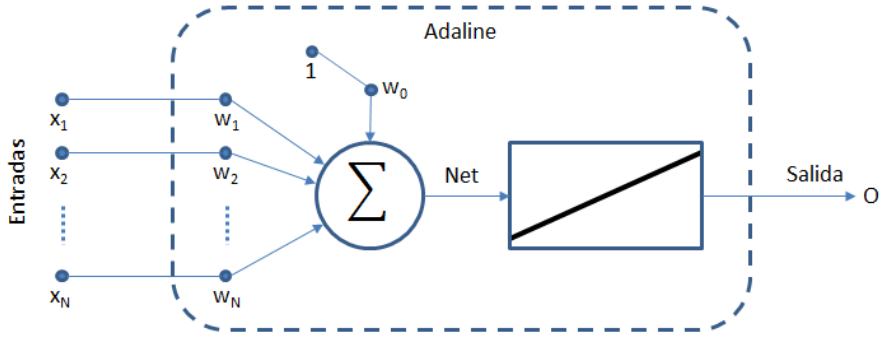


FIGURA 2.15: Esquema de Adaline.

La segunda diferencia es la ley de aprendizaje. En Adaline se utiliza la regla delta, basada en el mínimo error cuadrático medio (*Least Mean Squared* o LMS *error*). Los pesos de la red se ajustan mediante una búsqueda guiada por el gradiente descendiente de la función del error.

Antes de abordar la explicación de la regla delta se realiza un cambio en los nombres de algunas variables para facilitar el desarrollo de la sección 2.2.5.3:

- O: salida de la red (por ***o**utput*). En Adaline $O = \sum_{i=0}^{N} w_i x_i = Net$.
- k: identificador de un vector de entrenamiento.
- E: error cuadrático medio. $E = \frac{1}{2L} \sum_{k=1}^{L} (y_k - O_k)^2$, donde L es la cantidad de vectores de entrenamiento.

La adaptación de los pesos se lleva a cabo a través de una búsqueda sobre la superficie del error. Para encontrar el mínimo de la función del error los pesos se modifican en cantidades proporcionales al gradiente decreciente de la función E. La ley de aprendizaje se define como:

$$\Delta_{w_{ik}} = -\alpha \frac{\partial E_k}{\partial w_i} \tag{2.8}$$

donde α es la tasa de aprendizaje.

Aplicando la regla de la cadena,

$$\frac{\partial E_k}{\partial w_i} = \frac{\partial E_k}{\partial O_k}\frac{\partial O_k}{\partial Net_k}\frac{\partial Net_k}{\partial w_i}$$

$$\frac{\partial E_k}{\partial O_k} = \frac{\partial\left(\frac{1}{2}(y_k - O_k)^2\right)}{\partial O_k} = \frac{1}{2}2(y_k - O_k)(-1) = -(y_k - O_k)$$

$$\frac{\partial O_k}{\partial Net_k} = 1 \tag{2.9}$$

$$\begin{aligned}\frac{\partial Net_k}{\partial w_i} &= \frac{\partial\left(\sum_{l=1}^{N} w_l x_{kl}\right)}{\partial w_i} = \frac{\partial\left(w_0 x_0 + \cdots + w_i x_i + \cdots + w_N x_N\right)}{\partial w_i} \\ &= \frac{\partial\left(w_i x_i\right)}{\partial w_i} = x_i\end{aligned}$$

Llamando $\delta = \frac{\partial E_k}{\partial O_k}\frac{\partial O_k}{\partial Net_k}$, la regla delta queda definida como:

$$\begin{aligned}\Delta_{w_{ik}} &= -\alpha\delta\ x_i \\ &= -\alpha\left(-(y_k - O_k)\right)x_i \\ &= \alpha(y_k - O_k)x_i\end{aligned} \tag{2.10}$$

Finalmente, la expresión de la ley de aprendizaje de Adaline (ecuación 2.10) es igual a la del perceptron (2.2.5.1), pero en el caso de Adaline el error puede tomar cualquier valor en $\mathbb{R}$.

El método de entrenamiento también es similar al del perceptron, con la diferencia de que el error nunca es cero, así que se necesita otra condición de corte. Típicamente se corta el entrenamiento cuando se llega a un umbral de error previamente definido.

En cuanto a las limitaciones, Adaline solo puede clasificar problemas que sean linealmente separables. La conexión en capas de unidades Adaline, como se hizo en la sección anterior con perceptrones, no tiene sentido. Como no hay no-linealidades en las funciones de activación, Adaline realiza una transformación lineal de las entradas. Las transformaciones lineales sucesivas se pueden resumir en una sola transformación, por lo tanto, conectar Adalines en capas da el mismo resultado que una sola neurona Adaline.

2.2.5.3. Redes con conexiones hacia adelante

Se ha visto que conectar neuronas (no lineales) en capas puede resolver problemas de clasificación con clases no-linealmente separables. En esta sección se presenta un método para ajustar los pesos de redes neuronales multicapa.

Una red neuronal con conexiones hacia adelante o *feed-forward* es una red donde las neuronas se disponen en capas y cumplen con las siguientes condiciones:

- Todas las neuronas de una capa conectan su salida a una de las entradas de cada una de las neuronas de la capa siguiente (conexiones hacia adelante). Se dice que estas redes están densamente conectadas.
- No hay conexiones hacia neuronas de capas anteriores.
- No hay conexiones con neuronas de la misma capa.

En la figura 2.16 se muestra un esquema de red *feed-forward* de tres capas o una capa oculta.

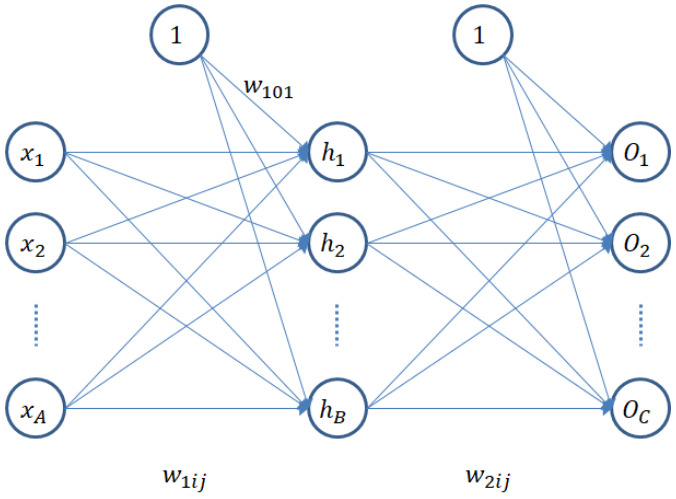


FIGURA 2.16: Esquema de red neuronal con conexiones hacia adelante de tres capas.

Los nodos x_i de la figura 2.16 son las neuronas de la capa de entrada, aunque en realidad no ejecutan ningún cálculo ni tienen parámetros para ajustar, sus salidas son directamente los valores de los A elementos del vector de entradas. Los nodos h_i (por *hidden*) son las neuronas que forman la única capa oculta del ejemplo. En principio no hay límite en la cantidad de capas ocultas, ni una cantidad de neuronas definida para cada capa. Los nodos O_i son las neuronas de la capa de salida. Si la red se aplica en un caso de clasificación, la cantidad de neuronas de la capa de salida es la cantidad de clases. Si se trata de un caso de regresión, la capa de salida usualmente tendrá una sola neurona

y función de activación lineal [55]. En la figura, el primer subíndice de los coeficientes indica la capa (1 para la capa oculta y 2 para la de salida), y los dos subíndices siguientes la i-ésima entrada de la neurona j.

Todas las neuronas de una capa tienen la misma función de activación. Estas funciones deben ser continuas, derivables y no decrecientes. La función más utilizada (por lo menos hasta la aparición del aprendizaje profundo) es la función sigmoidal [58]:

$$f(z) = \frac{1}{1 + e^{-z}}$$

con derivada:

$$\frac{\partial f(z)}{\partial z} = f(z)\,(1 - f(z))$$

Esta función, junto a la tangente hiperbólica, es muy utilizada porque tiene forma de escalón suavizado. La tangente hiperbólica varía entre -1 y 1, mientras que la función sigmoidal lo hace entre 0 y 1. Al reemplazar la función de activación lineal, en la regla delta cambia la ecuación 2.9. Por ejemplo, si las neuronas de salida tienen activación sigmoidal, la salida se calcula como:

$$O_j = f(Net_j) = \frac{1}{1 + e^{-Net_j}}$$

y la derivada de la salida con respecto a Net:

$$\frac{\partial O_j}{\partial Net_j} = \frac{1}{1 + e^{-Net_j}} \left(1 - \frac{1}{1 + e^{-Net_j}}\right) = O_j(1 - O_j)$$

El ajuste de los pesos se lleva a cabo con una variante de la regla delta llamada regla delta generalizada o retropropagación (*backpropagation*). Cada paso o ciclo del método tiene dos fases. En la primera fase, se toma un vector de entrada y se lo propaga hacia adelante, a través de todas las capas, hasta calcular la salida. En la segunda fase se calcula el error de la capa de salida y se lo propaga hacia atrás, calculando el error de las neuronas de las capas ocultas en base al error cometido por las neuronas de la capa siguiente ponderado por los pesos que las conectan. Una vez calculados los errores, se modifican los pesos. Los pasos son:

- Inicializar los pesos con valores aleatorios
- Mientras no se alcance un error aceptable o el límite de ciclos:

- Ejecutar ciclo completo de entrenamiento (*epoch*). Mientras existan vectores de entrenamiento hacer:
  - Tomar un par entrada/salida del conjunto de datos de entrenamiento.
  - Calcular la salida de cada capa hasta llegar a la capa de salida.
  - Calcular el error de las neuronas de la capa de salida.
  - Calcular el error de las neuronas de la última capa oculta.
  - Repetir el punto anterior para todas las capas ocultas (desde la salida hacia la entrada).
  - Calcular los Δw_i de cada neurona en función de su propio error.
  - Modificar los pesos.

A continuación se muestran, como ejemplo, las expresiones de cálculo que se utilizarían durante el entrenamiento del modelo de la figura 2.16 para funciones de activación sigmoidales.

Salida de la capa oculta:

$$Net_{1j} = \sum_{i=0}^{A} w_{1ij} x_i$$

$$h_j = \frac{1}{1 + e^{-Net_{1j}}}$$

Salida de la capa de salida:

$$Net_{2j} = \sum_{i=0}^{B} w_{2ij} h_i$$

$$O_j = \frac{1}{1 + e^{-Net_{2j}}}$$

Errores de la capa de salida:

$$\delta_{2j} = O_j(1 - O_j)(y_j - O_j)$$

Errores de la capa oculta:

$$\delta 1j = h_j(1 - h_j) \sum_{i=1}^{C} \delta_{2i} w_{2ji}$$

Ajuste de pesos de la capa de salida:

$$\Delta w_{2ij} = -\alpha \delta_{2j} h_i$$

$$w_{2ij(t+1)} = w_{2ij(t)} + \Delta w_{2ij}$$

Ajuste de pesos de la capa oculta:

$$\Delta w_{1ij} = -\alpha \delta_{1j} x_i$$

$$w_{1ij(t+1)} = w_{1ij(t)} + \Delta w_{1ij}$$

Error cuadrático medio:

$$E = \frac{1}{L} \sum_{k=1}^{L} \sum_{j=1}^{C} \delta_{2jk}^2$$

donde L es la cantidad de vectores de entrenamiento.

Las decisiones de diseño de las redes neuronales, como la cantidad de capas ocultas, la cantidad de neuronas de cada capa oculta, las funciones de activación, las distribuciones de los valores aleatorios de inicialización de los pesos y la tasa de aprendizaje, entre muchas otras, no tienen recomendaciones generales que funcionen en todos los casos. Existen enfoques de diseño basados en búsquedas en el espacio de los hiperparámetros para definir la arquitectura. Algunos comienzan con modelos simples que crecen en complejidad hasta lograr resultados aceptables y otros siguen el orden inverso.

2.2.6. Aprendizaje profundo

El aprendizaje profundo (*deep learning*) es la evolución de las redes neuronales artificiales. El término enfatiza que ahora es posible entrenar redes neuronales más profundas que antes, es decir con mayor cantidad de capas, y pone el foco de atención en la importancia de esta profundidad. Este nuevo término va más allá de la perspectiva neurocientífica utilizada en la generación anterior de modelos de aprendizaje automático. Apela a un principio más general de aprendizaje compuesto de múltiples niveles y que se puede aplicar en contextos de aprendizaje automático que no están necesariamente inspirados en la neurociencia. Hoy la neurociencia se considera una fuente importante de inspiración para los investigadores de aprendizaje profundo, pero ya no es la guía predominante [57].

Las redes neuronales profundas han superado a los sistemas de inteligencia artificial, tanto basados en otras tecnologías de aprendizaje automático, como diseñados con funcionalidades *ad hoc* [57]. Hoy son el estado del arte para la amplia mayoría de tareas de reconocimiento de patrones.

El aprendizaje profundo es resultado de (muchos) avances sobre distintos temas propios de las redes neuronales, como también de avances tecnológicos más generales, por ejemplo el aumento de la capacidad de cálculo, la posibilidad de paralelizar operaciones, la conectividad y la creciente cantidad de datos disponibles.

Una de las particularidades que distingue al aprendizaje profundo, y que probablemente sea la causa de su buen desempeño, es la capacidad para aprender representaciones complejas de características y ajustar los parámetros del clasificador al mismo tiempo [59]. En esta tecnología, tal como lo ilustra la figura 2.17, la extracción de características se hace en el mismo modelo que la clasificación.

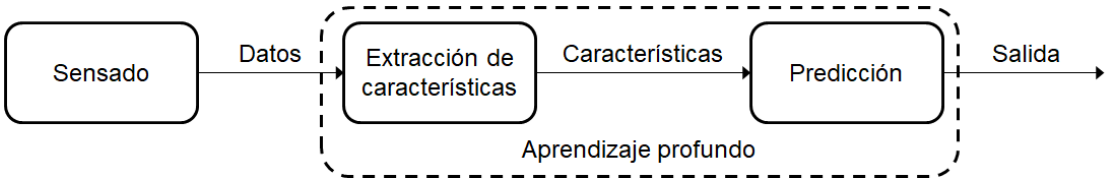


FIGURA 2.17: Esquema de aprendizaje profundo donde se resalta la integración de la extracción de características y la clasificación en el mismo modelo.

El neocognitron, creado en 1982 por Fukushima [60], es considerado por muchos autores el primer modelo de aprendizaje profundo. Es un modelo extremo a extremo de aprendizaje automático que reconoce caracteres escritos a mano. Su arquitectura se inspira en la naturaleza jerárquica del procesamiento de señales en la corteza visual de los mamíferos. Está formado por capas llamadas simples y complejas similares a las capas de convolución y *pooling* utilizadas actualmente. Esta red reconoce jerárquicamente patrones visuales simples en las primeras capas y más complejos en las siguientes, como los modelos actuales de aprendizaje profundo, aunque el método de entrenamiento es distinto y más complicado. Recién en 1989, LeCun *et al.* [61] aplican el método de retropropagación a un modelo muy parecido al neocognitron y comienza la nueva era de las redes neuronales. A continuación se mencionan los cambios más importantes que hicieron posible esta revolución y que son relevantes para este trabajo.

2.2.6.1. Funciones de activación

Una de las complicaciones para entrenar redes neuronales con muchas capas es que el gradiente de la función del error se desvanece a medida que se propaga. La causa son

las regiones de derivada casi nula en las funciones de activación sigmoidal y tangente hiperbólica. Este problema fue descubierto en la década de 1990 [62, 63]. En el año 2011 Glorot *et al.* [64] demuestran que la función de activación ReLU (por *Rectified Linear Unit*) reduce el efecto de desvanecimiento. Con ReLU los modelos típicamente aprenden mucho más rápido que con las anteriores funciones de activación [57, 65]. En el presente, y a pesar de una gran cantidad de variantes, ReLU es la función de activación más popular [57].

En la tabla 2.4 se muestran algunas de las funciones de activación más comunes y en la figura 2.18 se pueden ver sus respectivas curvas. Algunas de estas funciones introducen "mejoras" como, por ejemplo, tener derivada no nula para $z < 0$, en los casos de *Leaky* ReLU y PReLU (por *parametric*), o transición suave para $z = 0$ en el caso de ELU (por *exponential*).

Tabla 2.4: Funciones de activación

Función	Expresión $f(z)$	Derivada $f'(z)$	Rango
Sigmoidal	$\frac{1}{1+e^{-z}}$	$f(z)(1-f(z))$	$[0,1]$
Tan. Hiperbólica	$\tanh(z)$	$1-\tanh(z)^2$	$[-1,1]$
ReLU	$\max(0,z)$	$\begin{cases} 0 & \text{si} \quad f(z) \leq 0 \\ 1 & \text{si} \quad f(z) > 0 \end{cases}$	$[0,\infty]$
Leaky ReLU	$\begin{cases} 0,01z & \text{si} \quad z \leq 0 \\ z & \text{si} \quad z > 0 \end{cases}$	$\begin{cases} 0,01 & \text{si} \quad z \leq 0 \\ 1 & \text{si} \quad z > 0 \end{cases}$	$[-\infty,\infty]$
PReLU	$\begin{cases} \alpha z & \text{si} \quad z \leq 0 \\ z & \text{si} \quad z > 0 \end{cases}$	$\begin{cases} \alpha & \text{si} \quad z \leq 0 \\ 1 & \text{si} \quad z > 0 \end{cases}$	$[-\infty,\infty]$
ELU	$\begin{cases} \alpha(e^z-1) & \text{si} \quad z \leq 0 \\ z & \text{si} \quad z > 0 \end{cases}$	$\begin{cases} \alpha e^z & \text{si} \quad z \leq 0 \\ 1 & \text{si} \quad z > 0 \end{cases}$	$[-\alpha,\infty]$

2.2.6.2. Redes neuronales convolucionales.

Parte del éxito del aprendizaje profundo se debe a nuevos tipos de neuronas, capas de neuronas u operaciones que componen las redes neuronales. Además de las capas de neuronas densamente conectadas con conexiones hacia adelante (capas densas) vistas en la sección 2.2.5.3, son muy utilizadas las neuronas recursivas, las *Long short-term memory* (LSTM), las capas de convolución y las capas de *pooling*. De las mencionadas, las dos primeras se aplican sobre series temporales, en problemas relacionados con el lenguaje por ejemplo, y no son relevantes para esta trabajo.

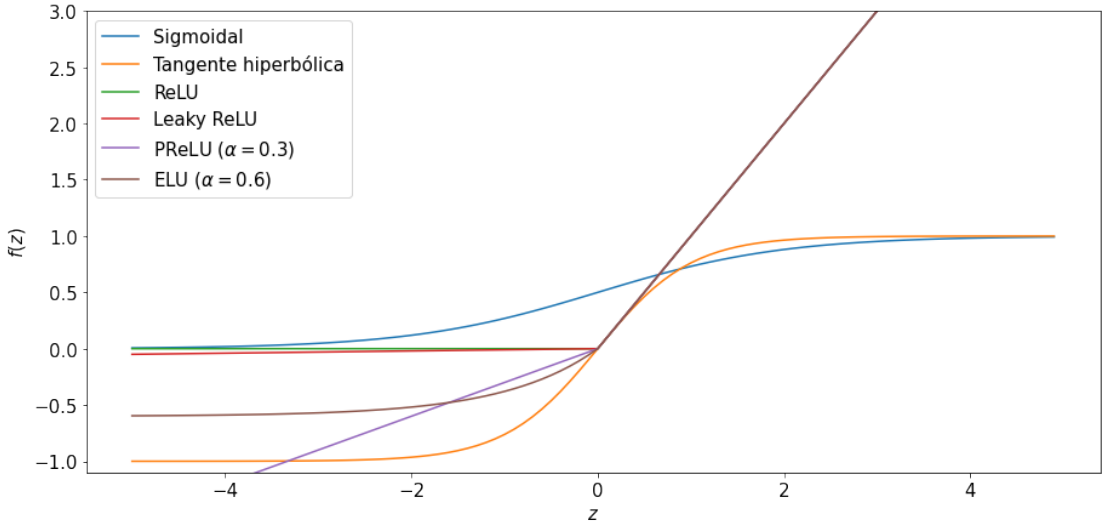


FIGURA 2.18: Funciones de activación sigmoidal, tangente hiperbólica, ReLU, Leaky ReLU, PReLU (para $\alpha = 0,3$) y ELU (para $\alpha = 0,6$).

Las redes neuronales convolucionales (CNN por su sigla en inglés) son un caso particular de red con conexiones hacia adelante cuyas neuronas no están densamente conectadas. Han sido diseñadas para procesar datos en forma de arreglos (*arrays*), donde la ubicación de los elementos es parte de la información, como por ejemplo los pixeles de una imagen. Sobre este tipo de datos las CNN son muchos más fáciles de entrenar y generalizan mucho mejor que las redes que solo tienen capas de neuronas densas [58]. En el caso del audio, si está sin procesar es un arreglo de dimensión 1 y si se usa una representación frecuencial, como el espectrograma, se convierte en un arreglo 2D al igual que una imagen. En ambos casos es importante la ubicación de cada dato y la relación con los datos vecinos.

Hay cuatro conceptos fundamentales detrás de las CNN que aprovechan las propiedades de las señales naturales: conexiones locales, pesos compartidos, *pooling* y el uso de muchas capas [58].

La arquitectura de una CNN típica está formada por una serie de etapas. Las primeras etapas se componen de dos tipos de capas, las capas convolucionales y las capas de *pooling*.

Capas de convolución Cada neurona de una capa de convolución, al igual que las neuronas de las capas densas, realiza una transformación lineal de sus entradas antes de la aplicación de la función de activación. La salida de la neurona j se define como:

$$s_j = f(\sum_{i=0}^{N} w_{ij}\ x_i)$$

donde $f()$ es la función de activación, x_i es la entrada i, w_{ij} es el peso sináptico correspondiente a la entrada i de la neurona j y w_{0j} es el término independiente o bias de la neurona j ($x_0 = 1$).

Las neuronas en una capa de convolución se organizan en mapas de características (*feature maps*), dentro de los cuales cada unidad está conectada a regiones determinadas de los mapas de características de la capa anterior a través de un conjunto de pesos llamado *kernel* o *filter bank*. Todas las neuronas del mismo mapa de características comparten el mismo *kernel*. Mapas de características distintos en una capa usan *kernels* diferentes. En la figura 2.19 se ilustra la conexión de las neuronas. En el ejemplo de la figura, las neuronas de la derecha se conectan a los elementos de la capa anterior (izquierda) mediante un *kernel* de tres pesos. Se puede ver que cada neurona no se conecta a todos los elementos de la capa anterior, sino que tiene un "campo receptivo" acotado. Cada uno de los tres pesos se dibujó con un color y estilo de linea distinto para reforzar la idea de que son compartidos. Notar que, por ejemplo, el primer peso de la primera neurona es igual al primer peso de la segunda neurona.

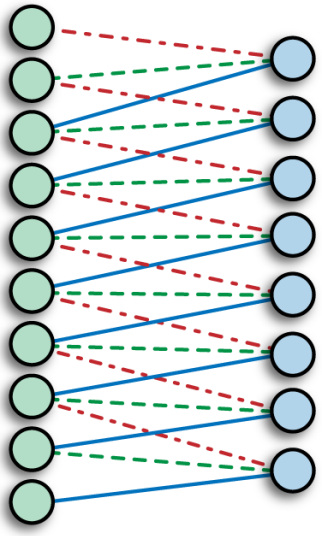


FIGURA 2.19: Pesos compartidos por neuronas del mismo mapa de características de una CNN. Las neuronas de la derecha tienen un campo receptivo de tamaño = 3 sobre la capa anterior (izquierda).

Las neuronas de la misma posición de mapas de características distintos comparten el campo receptivo, es decir, se conectan a las mismas entradas. De esta forma la salida de una capa de CNN es la convolución discreta de las entradas por los *kernels*. Una vez entrenada la red, cada *kernel* se especializa en reconocer o transformar cierto patrón de los datos de entrada.

La razón para esta arquitectura es doble. En primer lugar, en un arreglo los grupos locales de valores frecuentemente forman patrones fácilmente detectables. En segundo

lugar, las distribuciones estadísticas locales de las imágenes y otras señales (como el audio) son invariantes a la ubicación. En otras palabras, si un patrón puede aparecer en una parte del arreglo, este puede aparecer en cualquier parte, de ahí la idea de que neuronas en diferentes ubicaciones compartan los mismos pesos y detecten los mismos patrones en distintas partes del arreglo [58].

En la figura 2.20 se muestra el campo receptivo de dos neuronas de un mapa de características. Si se piensa en la operación de convolución discreta, donde los pesos del *kernel* son los valores de la máscara de convolución, cada neurona representa uno de los lugares donde se posiciona la máscara sobre la entrada. En el caso de la figura, el tamaño del *kernel* o máscara es 3×4. Además del tamaño es necesario definir la cantidad de pasos (*strides*) de desplazamiento vertical y horizontal, es decir la cantidad de elementos de diferencia entre una posición y la posición contigua. En este caso, el campo receptivo se mueve dos elementos en ambos sentidos. Otra decisión que se debe tomar es qué pasa con los bordes. La máscara no puede tener elementos fuera del espacio de la entrada, por lo tanto no todos los valores de entrada tienen la misma probabilidad de pertenecer a un campo receptivo o de ser multiplicados por determinado peso aunque los desplazamientos fueran (1, 1). Es común agregar elementos nulos alrededor de la entrada para que la máscara pueda abarcar más valores. Esta técnica se llama *padding*. En las librerías de aprendizaje profundo generalmente se manejan dos tipos de *padding*: "valid", que indica que la máscara solo se posiciona en lugares válidos de la entrada original, y "same", que indica que se agregarán tantos valores nulos alrededor de la entrada como sean necesarios para que el tamaño del mapa de características sea igual al tamaño de la entrada.

Capas de *pooling* Las capas de agrupación o *pooling* tienen una mecánica similar a las capas de convolución, en el sentido de que cada elemento tiene su propio campo receptivo y realiza la misma operación que los elementos vecinos. Estas capas no se ajustan, no tienen pesos y su funcionamiento no cambia durante el entrenamiento. La salida de cada elemento se calcula como el valor máximo (*max pooling*) o el valor medio (*average pooling*) del campo receptivo. A diferencia de las capas de convolución, donde usualmente los campos receptivos de una neurona se forman sobre todos los mapas de características de la capa anterior, en las capas de *pooling* generalmente existe un mapa de características por cada mapa de características de la capa anterior.

Mientras que la función de la capa de convolución es detectar conjunciones locales de características de la capa anterior, la función de la capa de *pooling* es fusionar varias características semánticamente similares en una. Debido a que las posiciones relativas de las características que forman un patrón pueden variar, la detección confiable del patrón se puede realizar haciendo un "granulado grueso" de la posición de cada característica,

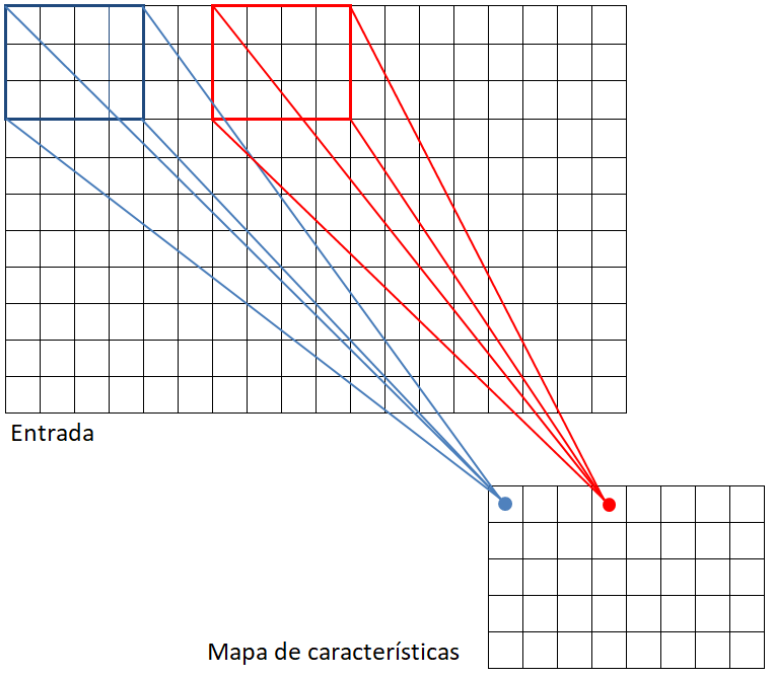


FIGURA 2.20: Campo receptivo de una neurona de una CNN para un *kernel* de tamaño 3×4, desplazamiento vertical 2, desplazamiento horizontal 2 y *padding* = "valid".

lo que reduce la dimensión de la representación y aporta invariancia a pequeños cambios y distorsiones [58].

En las CNN es común que se combinen algunas capas de convolución intercaladas con capas de *pooling* (para la extracción de características) y capas densas al final (para la clasificación). La retropropagación del gradiente del error a través de una CNN es tan simple como a través de una red con capas densas, lo que permite entrenar los pesos de todos *kernels* [58]. En la figura 2.21 se muestra como ejemplo la red LeNet5, para reconocimeinto de caracteres, presentada por LeCun *et al.* en 1998. Se pueden ver dos capas de convolución (la primera con 6 mapas y la segunda con 16), dos capas de *pooling* (*subsampling*) y tres capas densas.

2.2.6.3. Regularización

Las redes neuronales profundas son modelo complejos, con muchos parámetros para ajustar, lo que las hace propensas al sobreajuste. Goodfellow *et al.* definen a la regularización como "Cualquier modificación que podemos hacer sobre un algoritmo de aprendizaje con el objetivo de reducir su error de generalización pero no su error de entrenamiento" [57]. Es decir, el foco no está puesto en mejorar el resultado para los datos de entrenamiento, sino en reducir la brecha entre el rendimiento medido con los datos de entrenamiento y el rendimiento medido con los datos de validación. En esta

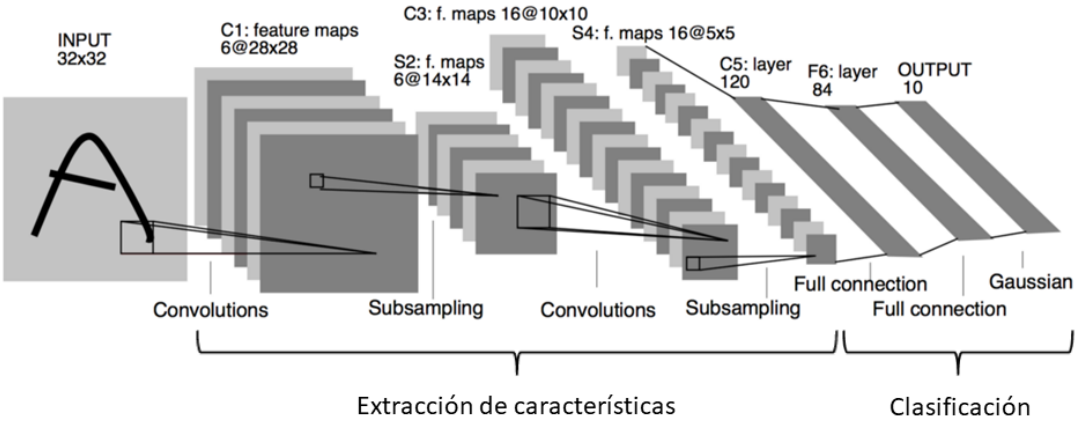


FIGURA 2.21: Capas de extracción de características y clasificación en LeNet5 [66]. Imagen tomada del artículo original y modificada en la parte inferior (llaves).

sección se explican brevemente cuatro estrategias para mejorar la generalización: *early stop*, penalización, *dropout* y aumentación de datos.

Early stop Al entrenar modelos grandes, con suficiente capacidad de representación como para sobreajustar, a menudo se observa que después de un punto, el error de entrenamiento disminuye de manera constante con el tiempo, pero el error del conjunto de validación comienza a aumentar nuevamente. En la figura 2.22 se muestra un ejemplo de este comportamiento. En lugar de tomar la configuración final de los pesos, se puede obtener un modelo con un mejor error para los datos de validación (y por lo tanto, con suerte, un mejor error para el conjunto de test) volviendo a la configuración de parámetros en el momento donde el error de validación es mínimo. El método es simple, cada vez que mejora el error en el conjunto de validación, se almacena una copia de los parámetros del modelo. Cuando termina el proceso de entrenamiento, se devuelven estos parámetros, en lugar de los últimos parámetros. El algoritmo finaliza cuando ningún parámetro ha mejorado con respecto al mejor error de validación registrado para un número preestablecido de iteraciones [57].

Penalización Otra estrategia es imponer restricciones adicionales al modelo de aprendizaje automático. Típicamente se agregan restricciones a los valores de los parámetros como términos adicionales en la función objetivo. Si se eligen con cuidado, estas restricciones o penalizaciones adicionales pueden conducir a un mejor desempeño en los datos de validación. En general, se puede regularizar un modelo que aprende la función $f(\boldsymbol{\theta}, \boldsymbol{x})$ agregando una penalización $\Omega(\boldsymbol{w})$ llamada "regularizador" a la función error. En la oración anterior, $\boldsymbol{\theta}$ es el vector de parámetros que incluye al *bias* (término independiente), mientras que $\boldsymbol{w}$ es el vector de pesos sin *bias*, es decir, $\boldsymbol{w}$ contiene solo

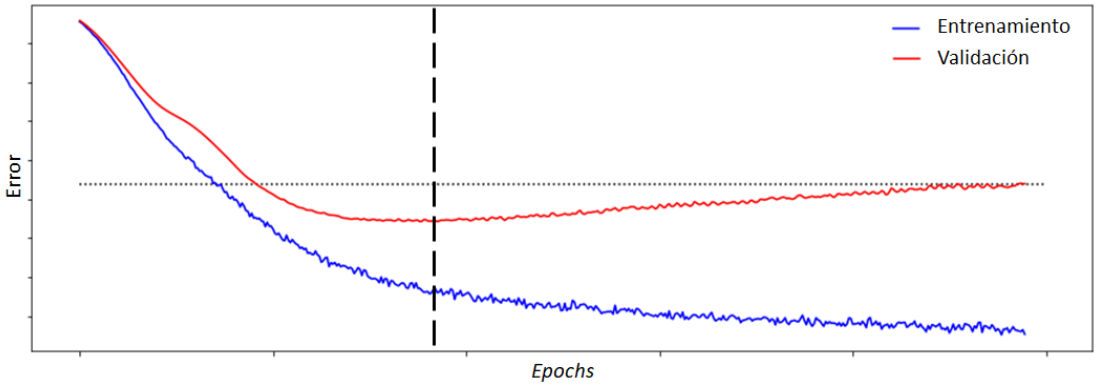


FIGURA 2.22: Error de entrenamiento y validación durante el aprendizaje. La linea de puntos horizontal marca el error de validación final. La linea segmentada vertical indica el punto donde el error de validación es mínimo.

los pesos que se multiplican por las entradas. Esto se debe a que los *bias* típicamente requieren menos datos para ajustarse correctamente porque solo controlan una variable, no la interacción entre dos variables como los pesos de $\boldsymbol{w}$. Regularizar el *bias* podría inducir a un infra-ajuste [57].

Las penalizaciones más utilizadas son:

- Regularización L1

$$\Omega(\boldsymbol{w}) = \beta \sum_{i=1}^{N} |w_i|$$

  donde β es una constante que pondera la contribución relativa de la penalidad.

- Regularización L2

$$\Omega(\boldsymbol{w}) = \beta \sqrt{\sum_{i=1}^{N} |w_i|^2}$$

- Regularización L1 y L2

$$\Omega(\boldsymbol{w}) = \beta \sum_{i=1}^{N} |w_i| + \gamma \sqrt{\sum_{i=1}^{N} |w_i|^2}$$

Dropout Es una estrategia potente de regularización, donde se eliminan al azar algunas neuronas de las capas ocultas durante el entrenamiento. En cada ciclo, con una probabilidad previamente definida (generalmente por capas), se eligen ciertas neuronas y se anulan sus salidas. En la figura 2.23 se muestra un ejemplo.

Srivastava *et al.* [67] demostraron que *dropout* es más efectivo que otras estrategias de regularización de bajo costo, como por ejemplo la penalización.

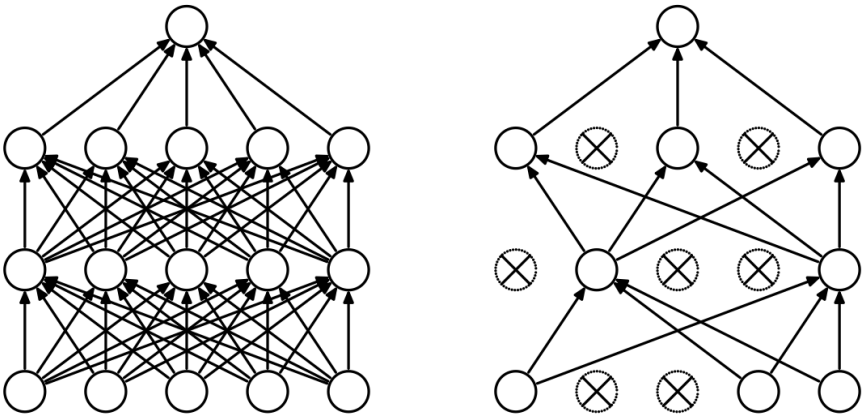


FIGURA 2.23: Ejemplo de aplicación de *dropout*. Izquierda: Red neuronal original con dos capas ocultas. Derecha: Red neuronal obtenida al aplicar *dropout* sobre la red de la izquierda. Imagen tomada del articulo de Srivastava *et al.* [67].

Dropout se puede aplicar a una amplia familia de modelos y también se puede combinar con otras técnicas de regularización reforzando la mejora obtenida [67].

Aumentación de datos Goodfellow *et al.* aseguran que, si bien son necesarias ciertas habilidades para obtener un buen rendimiento con un modelo de aprendizaje profundo, "afortunadamente" la cantidad de habilidades se reduce en la medida que la cantidad de datos se incrementa. La mejor manera de aumentar el grado de generalización de un modelo de aprendizaje automático es entrenarlo con más datos. Por supuesto, en la práctica, se cuenta con una cantidad limitada de datos. Una forma de solucionar este problema es crear datos nuevos [57].

Para algunas tareas de aprendizaje automático, es relativamente sencillo crear datos nuevos. En los casos de clasificación de imágenes y audio, un clasificador necesita tomar una entrada $\boldsymbol{x}$ compleja y de alta dimensión y resumirla a una categoría única y. Esto significa que la tarea principal a la que se enfrenta un clasificador es ser invariante ante una amplia variedad de transformaciones. Podemos generar nuevos pares ($\boldsymbol{x}$, y) simplemente transformando las entradas $\boldsymbol{x}$ de los datos existentes con las mismas transformaciones a las que se pretende ser invariante. En el caso del reconocimiento de patrones en imágenes, algunas de las transformaciones más comunes son la rotación, el *flipping* (espejado horizontal o vertical), los cambios de color, cambio de escala, el *cropping* (recortes), la traslación y la adición de ruido [68]. El la figura 2.24 se muestra un ejemplo de aplicación sobre imágenes.

Algunos ejemplos de transformaciones utilizadas en audio son cambios de tono, estiramiento del tiempo (*time stretching*) y filtrado de algunas frecuencias [69] o transformaciones que simulan perturbaciones por variación de la longitud del tracto vocal [70].

FIGURA 2.24: Ejemplo de transformaciones de aumentación de datos en imágenes.

Capítulo 3

Materiales y Métodos

3.1. Diseño de la red neuronal

3.1.1. Enfoque de diseño

¿Cómo diseñar una red neuronal desde cero? Partiendo desde una arquitectura determinada se pueden modificar algunos hiperparámetros y evaluar cómo afectan estos cambios sobre los resultados, pero difícilmente se pueda llegar a una solución aceptable con este método si el punto de partida no está suficientemente cerca del objetivo. El diseño de una red neuronal profunda, por lo menos cuando se cuenta con una cantidad reducida de datos, requiere conocimiento sobre el problema a resolver. Si bien los modelos de redes neuronales profundas no necesitan generalmente de un proceso previo de extracción de características porque son capaces de adaptar los filtros de las primeras capas realizando una extracción de características dentro de la misma red, es necesario conocimiento del dominio del problema como orientación para definir, por ejemplo, la cantidad de capas, su tamaño, la cantidad, tamaño y desplazamiento de los *kernels* si se usan capas de convolución y el tipo de capas de *pooling*.

3.1.2. Extracción de características

Para obtener la red neuronal presentada en este trabajo se partió desde un modelo inicial creado mediante la combinación de modelos más pequeños. Se diseñó cada parte para que fuera capaz de calcular una medida acústica relacionada con la calidad de la voz. Después, mediante un proceso iterativo de prueba, análisis y mejora del modelo completo, se obtuvo la red neuronal definitiva.

Las figuras 3.1 y 3.2 muestran el modelo tradicional para la clasificación de la calidad vocal y modelo inicial respectivamente.

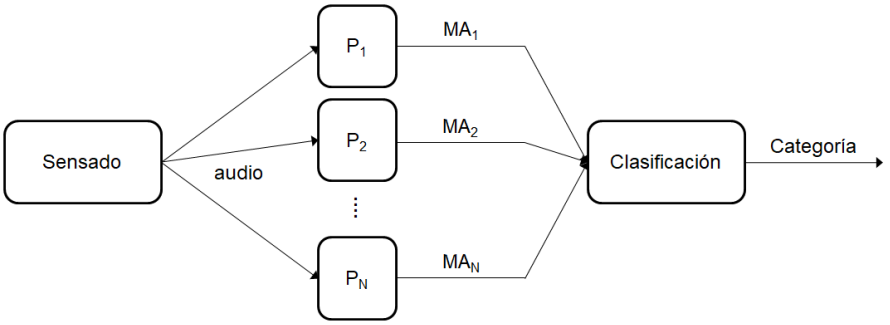


FIGURA 3.1: Modelo tradicional para la clasificación de la calidad vocal basado en medidas acústicas.

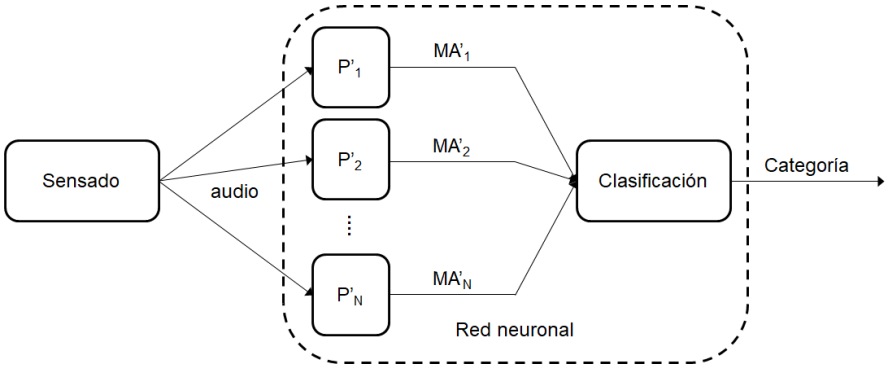


FIGURA 3.2: Modelo inicial general de red neuronal para el cálculo de la calidad vocal.

En el modelo tradicional existen procesos P_i que calculan ciertas medidas acústicas MA_i sobre la señal de audio. Luego estas medidas son las entradas (características) del módulo clasificador.

El modelo inicial es una red neuronal que recibe el audio y lo clasifica. Internamente, las partes P'_i calculan los valores MA'_i, que luego son utilizadas como entrada en las últimas capas de la red para realizar la clasificación. Los valores MA'_i no son exactamente las medidas acústicas MA_i, pero deben contener la información importante que estas aportan a la clasificación del grado general de disfonía.

3.1.3. Representación frecuencial

Según lo planteado hasta el momento, la entrada de la red neuronal será la señal de audio. Existen varias representaciones posibles para la señal de audio. Los modelos neuronales de clasificación de audio (en general, no solo en el dominio de la salud vocal) se pueden dividir en dos clases según la representación de las entradas. En la primera clase están los modelos que utilizan el audio sin procesar (*raw* audio) y en la segunda los que utilizan

representaciones frecuenciales, por ejemplo el espectrograma, el cepstrum y los MFCC. En la figura 3.3 se muestra el esquema de clasificación de una red neuronal de la segunda categoría. Se puede observar que aparece un nuevo proceso, la transformación del *raw* audio en su representación frecuencial.

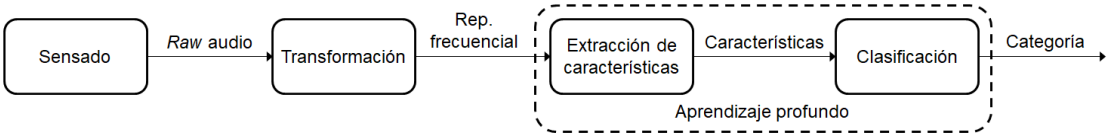


FIGURA 3.3: Esquema de un clasificador basado en redes neuronales que recibe como entrada la representación frecuencial del audio.

Nam *et al.* explican en [71] la evolución entre las dos clases de representación de las entradas. En los primeros modelos de aprendizaje profundo sobre audio las entradas eran las salidas de bancos de filtros de frecuencias (*frequency-filter banks*). En modelos posteriores, se utilizaba la salida de la transformada discreta de Fourier (DFT por *discrete Fourier transform*), la cual en realidad es una transformada rápida de Fourier (STFT por *short-time Fourier transform*). Finalmente, a medida que los recursos computacionales lo permitieron, algunos modelos comenzaron a tomar el *raw* audio como entrada. Durante esta evolución, las redes neuronales profundas reemplazaron las operaciones de los enfoques tradicionales por transformaciones cuyos parámetros pueden ser adaptados. La STFT es usualmente reemplazada por bancos adaptables de filtros en el dominio del tiempo y los filtros MFCC son reemplazados por bancos adaptables de filtros en el dominio de la frecuencia. Ambos bancos de filtros son usualmente implementados por capas de convolución.

Tal como se mencionó en la sección 2.2.6, en los modelos de aprendizaje profundo, las primeras capas de la red neuronal se encargan de la extracción de características, mientras que las siguientes realizan la clasificación. Se supone que en esta integración de las dos etapas en un mismo modelo, es donde reside la ventaja del método, ya que es posible encontrar los parámetros óptimos de extracción de características y clasificación en conjunto [72]. Hoshen *et al.* demostraron en [73] que una capa de convolución es capaz de aproximar un banco de filtros que contiene valores comparables con los coeficientes de la DFT. En [74], Sainath *et al.* fueron capaces de mejorar el estado del arte en reconocimiento del habla utilizando filtros entrenados con una capa de convolución.

Para este trabajo se optó por integrar en la red neuronal no solo la extracción de características y la clasificación, sino también la representación frecuencial del audio (se utiliza el término "representación frecuencial" por simplicidad, aunque en algunos casos, por ejemplo en el caso del cepstrum, los datos no están en el dominio de la frecuencia). De esta forma, durante el entrenamiento el modelo puede ajustar los parámetros de toda la transformación. Cuando se elige una representación como MFCC, espectrograma o

cepstrum, estas representaciones son rígidas y no permiten adaptaciones. En la figura 3.4 se muestra el esquema del enfoque elegido para la representación del audio.

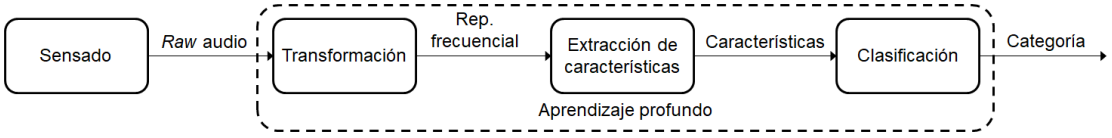


FIGURA 3.4: Esquema de un clasificador basado en redes neuronales que recibe *raw audio* como entrada.

3.1.4. Consideraciones

De acuerdo con el enfoque de diseño planteado, el modelo de clasificación será del tipo extremo a extremo (*end-to-end*), porque la red neuronal recibirá audio, sin procesar o su representación frecuencial en el caso de una variante presentada más adelante, y devolverá el grado general de disfonía. No hay ningún cálculo de características externo a la red neuronal, todo la información necesaria se calcula en las distintas partes de la red.

Durante el diseño y la integración de estas partes es necesario asegurar que cada parte cumpla con las siguientes condiciones:

Propagación hacia adelante Durante la etapa de propagación hacia adelante de la red neuronal, cada parte debe calcular, o transmitir y preservar (según el caso) información útil para la clasificación del grado general de disfonía.

Propagación hacia atrás Durante la etapa de propagación hacia atrás, cada parte debe ser capaz de propagar el error de forma adecuada para que las capas anteriores puedan adaptar sus pesos. Las redes neuronales de aprendizaje profundo, además de capas con neuronas y capas de *pooling*, suelen introducir otro tipo de operaciones, como cambio de forma (*reshape*) de vectores, aplicación de funciones matemáticas y hasta cálculos más complejos definidos por el desarrollador. Es posible que alguna de estas operaciones impida que el error se propague hacia atrás con la calidad necesaria. Por ejemplo, el uso de una función que no sea continua, derivable y no decreciente generaría una superficie de error donde no sería posible adaptar los pesos de neuronas que se encuentren antes de dicha función para conseguir un resultado aceptable.

En los siguientes capítulos se presentan los modelos desarrollados para procesar por partes la entrada de *raw audio* hasta lograr finalmente la clasificación del grado general de disfonía.

3.2. Evaluación de resultados

La calidad de la voz es una interacción entre el estímulo acústico proveniente de la voz y un oyente; la señal acústica en sí misma no posee calidad vocal, esta se evoca en el oyente [75]. La valoración audioperceptiva de la voz tiene la debilidad de ser subjetiva y se ve afectada por múltiples factores. Como consecuencia, el nivel de concordancia intraevaluador y, más aún, el nivel de concordancia interevaluador, pueden ser muy pobres [11]. Algunos de los factores que inciden en la variabilidad interevaluador son el sesgo del oyente frente al conocimiento de los antecedentes del diagnóstico médico, la experiencia del evaluador, la formación profesional (por ejemplo, en otorrinolaringología, fonoaudiología, profesores de canto, fonetistas y estudiantes de pregrado o postgrado), la formación musical y el entrenamiento en el juicio perceptivo auditivo (por ejemplo, de oyentes nativos) [4].

Los factores mencionados en el párrafo anterior no pueden explicar la variación intra-evaluador, pero aún así bajo las mismas condiciones ambientales, distintas instancias de evaluación pueden resultar en valoraciones diferentes. Entonces, se puede afirmar que no existe una "respuesta correcta" para la valoración de una voz, sino múltiples valoraciones con resultados distintos y cierto nivel de consistencia.

La ausencia de una referencia estándar y la imposibilidad de mapear sin error una voz a un nivel de calidad mediante una función, por más compleja que ésta sea [76], complica tanto el entrenamiento como la evaluación de un modelo de reconocimiento automático.

Si suponemos que la "respuesta correcta" existe, las variaciones deben ser consideradas ruido, donde el ruido es un valor aleatorio cuya distribución depende tanto del conjunto de datos como del evaluador. Este componente aleatorio implica que es imposible lograr una clasificación automática que concuerde exactamente con una clasificación humana particular sobre determinado conjunto de datos. Si esto último ocurre, la explicación podría ser algún tipo de sobreajuste. En consecuencia con lo planteado, para medir el rendimiento de un sistema de clasificación automática de la calidad vocal, su exactitud debe ser analizada según el contexto de la base de datos (voces + valoraciones) utilizada. Si la desviación entre la predicción del modelo y la valoración del profesional es cercana a la variabilidad intraevaluador, se puede decir que el clasificador tiene una capacidad de valoración cercana a la humana.

En este trabajo se entrena una red neuronal artificial utilizando como referencia la valoración de un evaluador particular. Después, la variación entre los resultados del método automático y los datos de referencia se comparan con la variaciones intraevaluador e interevaluador de expertos humanos sobre los mismos datos. Para los datos utilizados

en este trabajo, la concordancia media en la valoración del grado general de disfonía intraevaluador es del 73 % y 53 % para las valoraciones interevaluador.

En la tabla 3.1 se muestra como ejemplo la matriz de confusión, o tabla de contingencias, entre las primeras valoraciones de los evaluadores 1 y 2 de la base de datos utilizada. Las filas indican los resultados del evaluador 1 y las columnas los del evaluador 2. Por ejemplo, si a un caso particular el evaluador 1 lo valoró con nivel "0" y el evaluador 2 con nivel "1", se acumula una unidad en la celda de la primera fila y segunda columna. La diagonal contiene todos los elementos donde hubo coincidencia. En la tabla 3.2 se muestra la matriz de confusión para la primera y segunda valoración del evaluador 1.

TABLA 3.1: Matriz de confusión para las primeras valoraciones del grado general de disfonía por los evaluadores 1 y 2 sobre los audios elegidos para el experimento.

		Eval. 2			
		0	1	2	3
Eval. 1	0	48	7	1	0
	1	31	30	17	0
	2	5	16	10	3
	3	0	5	19	14

TABLA 3.2: Matriz de confusión para las dos valoraciones del grado general de disfonía realizadas por el evaluador 1 sobre los audios elegidos para el experimento.

		Valorac. 2			
		0	1	2	3
Valorac. 1	0	39	17	0	0
	1	19	49	8	2
	2	1	9	20	4
	3	0	0	4	34

El modelo desarrollado en este trabajo clasifica el grado general de disfonía G. Se espera que el rendimiento sea cercano al rendimiento humano intraevaluador en las métricas de exactitud y error absoluto medio.

La exactitud se calcula como la relación entre la cantidad de casos correctamente clasificados (concordantes) y la cantidad total de casos. Es la métrica más frecuente en aprendizaje automático.

El grado G es una variable categórica ordinal, sus valores representan una jerarquía y, por lo tanto, no todos los desacuerdos tienen la misma importancia. Por ejemplo, un desacuerdo entre las categorías "0" y "3" obviamente representa una mayor desacuerdo que entre "0" y "1". La métrica del error absoluto medio tiene en cuenta esta diferencia. Se calcula sobre los valores numéricos que representan los nombres de las clases, es decir 0 para la clase "0", 1 para la clase "1", etc.

En los datos utilizados, el error absoluto medio interevaluador es 0.52 y 0.28 para el caso intraevaluador. La base de datos utilizada contiene seis valoraciones realizadas por tres profesionales distintos para cada audio más una valoración extra por audio que se añadió durante este trabajo.

Mas información sobre los datos mostrados en esta sección, sobre la distribución entre las clases y sobre los detalles de la evaluación del modelo se pueden ver en la sección 7.2.

3.3. Bases de datos

En esta sección se presentan las tres bases de datos utilizadas durante el desarrollo de la investigación. Cada una fue elegida para lograr avances en distintos objetivos particulares.

3.3.1. *Perceptual Voice Qualities Database*

La *Perceptual Voice Qualities Database*[1] (PVQD) es una base de datos de acceso público provista por *The Voice Foundation*. Contiene muestras de voz clasificadas por profesionales experimentados con el objeto de brindar material estandarizado a educadores de nuevos profesionales. Está formada por 296 archivos de audio que constan de las vocales sostenidas /a/ e /i/, además de oraciones de habla continua definidas por el *Consensus Auditory-Perceptual Evaluation of Voice* (CAPE-V). Todas las grabaciones se realizaron en un entorno clínico silencioso utilizando un micrófono de condensador montado en la cabeza a una distancia de 6 centímetros de la boca. El audio está codificado en 16 bits con una frecuencia de muestreo de 44.1 KHz. Los archivos de audio fueron editados para eliminar las instrucciones del profesional que realizó la grabación, pero persisten algunos sonidos que contaminan la muestra, por lo tanto es necesario revisar, editar y tomar decisiones sobre cada archivo para utilizar estos datos en un proceso de aprendizaje automático. Además de los audios se provee información sobre género, edad, diagnóstico y valoración de la voz en las escalas GRBAS y CAPE-V. Las valoraciones fueron realizadas por cuatro evaluadores, aunque el último de ellos (*rater* 4) valoró solamente el 16 % de los casos. Cada evaluador valoró dos veces cada audio. La clasificación se realizó durante varios días para evitar que el cansancio auditivo de los evaluadores influyera sobre los resultados. Más detalles sobre la PVQD se pueden encontrar en [77].

Esta base de datos se utiliza en los experimentos del capítulo 7, donde se realiza la clasificación del grado general de disfonía G. Uno de los factores que influyeron en la

[1] https://voicefoundation.org/health-science/videos-education/pvqd/

elección de PVQD es que presenta mejor distribución entre las clases de GRBAS. En la tabla 3.3 se muestra la cantidad de valoraciones por grado de G para cada evaluador y cada instancia de valoración. En las tablas 3.4, 3.5, 3.6 y 3.7 se pueden ver las valoraciones de rugosidad, soplosidad, astenia y tensión respectivamente.

Tabla 3.3: Grado general de disfonía por evaluador e instancia de valoración en PVQD.

		Grado de G				
Evaluador	Instancia	0	1	2	3	Total
1	1	79	119	48	50	296
1	2	86	111	48	51	296
2	1	113	98	61	24	296
2	2	114	96	62	24	296
3	1	137	71	52	36	296
3	2	120	85	55	36	296
4	1	14	27	8	-	49
4	2	21	23	4	1	49

Tabla 3.4: Rugosidad por evaluador e instancia de valoración en PVQD.

		Grado de R				
Evaluador	Instancia	0	1	2	3	Total
1	1	130	112	34	20	296
1	2	128	106	38	24	296
2	1	139	104	40	13	296
2	2	139	106	36	15	296
3	1	146	98	43	9	296
3	2	141	99	41	15	296
4	1	18	27	4	-	49
4	2	24	19	5	1	49

Tabla 3.5: Soplosidad por evaluador e instancia de valoración en PVQD.

		Grado de B				
Evaluador	Instancia	0	1	2	3	Total
1	1	169	72	26	29	296
1	2	174	66	27	29	296
2	1	182	70	32	12	296
2	2	182	72	28	14	296
3	1	151	90	31	24	296
3	2	155	76	39	26	296
4	1	29	16	4	-	49
4	2	29	15	4	1	49

Tabla 3.6: Astenia por evaluador e instancia de valoración en PVQD.

		Grado de A				
Evaluador	Instancia	0	1	2	3	Total
1	1	200	55	21	20	296
1	2	196	54	29	17	296
2	1	195	56	21	24	296
2	2	189	57	31	19	296
3	1	193	49	29	25	296
3	2	190	58	26	22	296
4	1	41	6	2	-	49
4	2	42	6	1	-	49

Tabla 3.7: Tensión por evaluador e instancia de valoración en PVQD.

		Grado de S				
Evaluador	Instancia	0	1	2	3	Total
1	1	147	82	41	26	296
1	2	141	86	39	30	296
2	1	174	72	34	16	296
2	2	165	75	41	15	296
3	1	151	85	34	26	296
3	2	156	80	40	20	296
4	1	29	17	3	-	49
4	2	31	13	5	-	49

3.3.2. Base de datos del Hospital Universitario Príncipe de Asturias

La base de datos del Hospital Universitario Príncipe de Asturias (HUPA) contiene grabaciones de voces pronunciando una vocal /a/ sostenida y una oración acústicamente balanceada. Se grabaron 520 voces sanas, de las cuales 356 fueron seleccionadas por superar ciertos criterios de inclusión. Luego se añadieron 203 voces patológicas afectadas por diversas enfermedades. Sobre la totalidad de las voces se valoró la calidad en escala GRBAS [78].

La versión de la base de datos a la que se tuvo acceso durante la realización de este trabajo está formada por 440 muestras en total, incluyendo voces sanas (239) y patológicas (201). Los audios están grabados en formato WAV, codificados con 16 bits y una frecuencia de muestreo de 25 KHz. La duración de los audios varía entre los 1,5 y 4 segundos.

Las personas grabadas son 175 hombres y 265 mujeres. Las edades varían entre los 8 y 78 años. Entre las voces patológicas se diagnosticaron 15 patologías diferentes, las cuales están registradas en la base de datos.

En la tabla 3.8 se muestra la distribución de la valoración GRBAS. Se puede observar que si bien esta base de datos tiene mayor cantidad de muestras que la PVQD, las clases están más desbalanceadas. Por ejemplo, para el grado general de disfonía G, en el caso con menor frecuencia PVQD tiene 36 muestras con grado 3 (sin tener en cuenta el evaluador 4), mientras que HUPA tiene solo 5.

Tabla 3.8: Distribución de la valoración en escala GRBAS de la base de datos HUPA.

	Grado				
Categoría	0	1	2	3	Total
G	237	127	71	5	440
R	232	119	82	7	440
B	282	110	37	11	440
A	210	169	52	9	440
S	251	134	50	5	440

3.3.3. *Speech Commands*

La base de datos *Speech Commands* es un estándar para el entrenamiento y evaluación de un tipo simple de tarea de reconocimiento del habla. El objetivo principal de *Speech Commands* es ofrecer una vía para crear y testear modelos pequeños que detecten la pronunciación de una palabra (comando) de un conjunto de diez o menos palabras objetivo y las diferencie de ruido de fondo o habla no relacionada. Este tipo de tarea de reconocimiento es conocida como *keyword spotting* [79].

El juego de datos consiste en 105,829 pronunciaciones de 35 palabras en inglés ("*Backward*", "*Bed*", "*Bird*", "*Cat*", "*Dog*", "*Down*", "*Eight*", "*Five*", "*Follow*", "*Forward*", "*Four*", "*Go*", "*Happy*", "*House*", "*Learn*", "*Left*", "*Marvin*", "*Nine*", "*No*", "*Off*", "*On*", "*One*", "*Right*", "*Seven*", "*Sheila*", "*Six*", "*Stop*", "*Three*", "*Tree*", "*Two*", "*Up*", "*Visual*", "*Wow*", "*Yes*" y "*Zero*") pronunciadas por 2,618 personas y capturadas a través de micrófonos de teléfonos celulares o notebooks. Cada palabra es almacenada en un archivo con duración de un segundo o menos, en formato WAV con tasa de muestreo de 16 KHz.

Se incluyen además audios con ruido ambiente para ser utilizados en el proceso de validación de modelos que se describe en el mismo artículo donde se presenta la base de datos [79]. Dicho protocolo se explica en la sección 4.3.2.2.

Capítulo 4

Representación frecuencial del audio

En este capítulo se desarrollan tres redes neuronales aplicables a la representación frecuencial del audio. Estas calculan el espectrograma, el cepstrograma (cepstrum en el tiempo) y la operación de *windowing*. Para cada uno de los modelos se ejecutan experimentos, se analizan los resultados y se llega a una conclusión.

4.1. Cálculo del espectrograma

El espectrograma es una representación de la variación del espectro de frecuencias de una señal. Esta variación puede ocurrir en el tiempo (audio por ejemplo), espacio (imágenes) y otros dominios. El espectro de frecuencias de una señal se obtiene a través de la transformada de Fourier (FT). Para datos discretos, como es el caso de este trabajo, se utiliza la transformada discreta de Fourier (DFT).

En aprendizaje automático es común utilizar la información espectral de los datos para encontrar características no evidentes en el dominio de origen. Existen muchos trabajos de aprendizaje profundo que utilizan como entrada la magnitud del espectro de frecuencias [80–82]. En esta sección se muestra que es posible calcular la misma información adicionando capas al principio de una red neuronal, con la posible ventaja de que el cálculo del espectro se puede adaptar al caso particular de clasificación, es decir, los coeficientes de las capas que calculan la DFT se pueden entrenar. Moreira *et al* propusieron previamente el cálculo de la DFT con redes neuronales en [83] separando los pesos en grupos que representan a la partes real e imaginaria, pero no se entrena la red, los pesos se asignan de forma directa. Velik, en [84], predice la DFT con pesos calculados

a partir de funciones exponenciales complejas también directamente asignadas. En este último trabajo se intentó entrenar la red partiendo de pesos aleatorios, pero sin éxito.

Durante el resto de la sección se explican algunos conceptos importantes y se presenta un modelo de red neuronal con una capa de convolución que calcula la transformada discreta de Fourier de tiempo reducido (STFT) para calcular la magnitud del espectrograma de la señal de entrada. Al elegir como salida la magnitud del espectro y descartar la fase, es posible implementar el modelo neuronal sin realizar operaciones de números complejos. Se presenta en detalle la estructura de la red y el cálculo de los coeficientes para esta alternativa.

Los coeficientes del modelo se pueden calcular de forma directa o se pueden obtener mediante entrenamiento con el método del gradiente descendiente.

4.1.1. Transformada discreta de Fourier.

La DFT convierte una secuencia finita de N números complejos (muestras) $\{x_n\} := x_0, x_1, ..., x_{N-1}$ en otra secuencia de $K = N$ números complejos $\{X_k\} := X_0, X_1, ..., X_{N-1}$.

$$X_k = \sum_{n=0}^{N-1} x_n \; e^{-i2\pi kn/N} \tag{4.1}$$

Según la fórmula de Euler,

$$X_k = \sum_{n=0}^{N-1} x_n \; [\cos(-2\pi kn/N) + i \; \sin(-2\pi kn/N)] \tag{4.2}$$

Es importante resaltar que la DFT es un operador lineal [85]. Se puede definir entonces la DFT como el mapa lineal $\mathcal{F} : \mathbb{C}^N \to \mathbb{C}^N$ tal que $X = \mathcal{F}(x)$ con la siguiente representación matricial.

$$\boldsymbol{X} = \boldsymbol{F}\boldsymbol{x}$$

donde

$$\boldsymbol{x} = \begin{bmatrix} x_0 & x_1 & x_2 & \dots & x_{N-1} \end{bmatrix}^\intercal,$$
$$\boldsymbol{X} = \begin{bmatrix} X_0 & X_1 & X_2 & \dots & X_{N-1} \end{bmatrix}^\intercal$$

y según la ecuación 4.1,

$$\boldsymbol{F} = \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & e^{-\frac{i2\pi}{N}} & e^{-2\frac{i2\pi}{N}} & \dots & e^{-(N-1)\frac{i2\pi}{N}} \\ 1 & e^{-2\frac{i2\pi}{N}} & e^{-4\frac{i2\pi}{N}} & \dots & e^{-2(N-1)\frac{i2\pi}{N}} \\ 1 & \vdots & \vdots & \ddots & \vdots \\ 1 & e^{-(N-1)\frac{i2\pi}{N}} & e^{-2(N-1)\frac{i2\pi}{N}} & \dots & e^{-(N-1)^2\frac{i2\pi}{N}} \end{bmatrix} \tag{4.3}$$

Para el caso de la ecuación 4.2,

$$\boldsymbol{F} = \boldsymbol{F}_C + i\boldsymbol{F}_S \tag{4.4}$$

donde

$$\boldsymbol{F}_C = \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & cos(-1\frac{i2\pi}{N}) & cos(-2\frac{i2\pi}{N}) & \dots & cos(-(N-1)\frac{i2\pi}{N}) \\ 1 & cos(-2\frac{i2\pi}{N}) & cos(-4\frac{i2\pi}{N}) & \dots & cos(-2(N-1)\frac{i2\pi}{N)} \\ 1 & \vdots & \vdots & \ddots & \vdots \\ 1 & cos(-(N-1)\frac{i2\pi}{N}) & cos(-2(N-1)\frac{i2\pi}{N}) & \dots & cos(-(N-1)^2\frac{i2\pi}{N}) \end{bmatrix}$$

$$\boldsymbol{F}_S = \begin{bmatrix} 0 & 0 & 0 & \dots & 0 \\ 0 & sin(-1\frac{i2\pi}{N}) & sin(-2\frac{i2\pi}{N}) & \dots & sin(-(N-1)\frac{i2\pi}{N}) \\ 0 & sin(-2\frac{i2\pi}{N}) & sin(-4\frac{i2\pi}{N}) & \dots & sin(-2(N-1)\frac{i2\pi}{N)} \\ 0 & \vdots & \vdots & \ddots & \vdots \\ 0 & sin(-(N-1)\frac{i2\pi}{N}) & sin(-2(N-1)\frac{i2\pi}{N}) & \dots & sin(-(N-1)^2\frac{i2\pi}{N}) \end{bmatrix}$$

4.1.1.1. Espectrograma.

El espectrograma es el reultado de la aplicación de la STFT. Para el caso de una señal discreta de longitud L, la STFT es simplemente la DFT de segmentos de longitud N, para $N < L$, de la señal. El resultado es una matriz $\boldsymbol{S}$ de valores complejos con la magnitud

y fase de la señal para cada frecuencia en cada segmento (tiempo). Generalmente en las columnas de la matriz se representa la dimensión del tiempo y en las filas las distintas frecuencias. La elección del valor de N depende del objetivo de la representación espectral de los datos. Para valores pequeños de N se obtiene alta definición en la dimensión del tiempo y baja en la de la frecuencia, mientras que para valores altos de N, el efecto se invierte. Los segmentos pueden estar superpuestos en una cantidad de muestras m entre 0 y $N-1$. Frecuentemente, en lugar de utilizar los elementos X_k del espectro, el espectrograma se forma con la magnitud del espectro, $|X_k|$, o con el cuadrado de la magnitud (*power spectrum*), $|X_k|^2$.

4.1.2. Experimento

4.1.2.1. Datos

La red neuronal se creó para predecir las magnitudes del espectrograma de señales de audio de dos segundos de duración.

Los audios son parte de la base de datos HUPA, presentada en la sección 3.3.2.

Entradas. Los audios se encuentran en formato WAV con una tasa de 25000 muestras por segundo. Se seleccionaron todos los archivos de duración ≥ 2 segundos y se tomaron las 50000 muestras centrales.

Los datos de entrada quedaron definidos entonces por 430 vectores de dimensión $L = 50000$, de los cuales se eligieron aleatoriamente 300 para entrenamiento y 130 para validación.

Salidas. Las salidas se calcularon como el valor absoluto de la STFT de las entradas con segmentos de tamaño $N = 1760$ y superposición $m = 1540$ elementos, lo que implica un desplazamiento de 220 elementos en cada transformación. Se calculó el módulo para obtener la magnitud del espectrograma descartando la fase. El espectrograma calculado de esta forma tiene 220 columnas (tiempo) por 881 filas (frecuencia). Debido a la naturaleza de los audios (voces en tono habitual) la energía se concentra casi por completo entre las primeras 200 filas (0 Hz a 2826 Hz). Por esta razón se decidió entrenar la red para predecir solo la magnitud de las filas 1 a 200 del espectrograma. Para este caso entonces, $K = 200$.

Las salidas quedaron definidas por vectores de dimensión 200×220. Notar que los valores de la salida pertenecen a $\mathbb{R}$ porque se toma el módulo del espectro.

4.1.2.2. Red neuronal

Para obtener la salida tal como se definió, la red neuronal debe estar formada por dos partes, una que calcule la STFT y otra que obtenga el valor absoluto.

El cálculo de la STFT se realiza con una capa de convolución, donde los pesos sinápticos son los elementos de la matriz $\boldsymbol{F}$ (ecuación 4.3) y la función de activación es lineal. Esto es posible debido a que, tanto la DFT como la operación realizada por cada neurona, son transformaciones lineales (ver sección 2.2.6.2). Es importante notar que los valores de la matriz $\boldsymbol{F}$ son constantes.

Existen dos formas de implementar el cálculo, según se elija la ecuación 4.1 o la ecuación 4.2 de la DFT. Si se utiliza la matriz $\boldsymbol{F}$ correspondiente a la ecuación 4.1, la matriz de pesos $\boldsymbol{W}$ formada por los coeficientes complejos w_{ij} (ecuación 4.3), tendrá tamaño $N \times K$. En el caso de la ecuaciíon 4.2, $\boldsymbol{W}$, de tamaño $N \times 2K$ estará formada por los valores (reales) de las matrices $\boldsymbol{F}_C$ y $\boldsymbol{F}_S$ (ecuación 4.4).

En términos de eficiencia no hay diferencia entre las dos alternativas. Para este trabajo se elige la segunda porque, debido a que la salida solo conserva información de la magnitud del espectro, es posible evitar el uso de operaciones de números complejos. Esto podría ser una ventaja práctica porque entre las librerías de software para redes neuronales todavía no es general el soporte a los números complejos [86]. Por lo demás, los dos enfoques son equivalentes.

Capa de convolución Para calcular la STFT con una capa de convolución es conveniente escribir la ecuación 4.2 de la siguiente manera.

$$X_k = \sum_{n=0}^{N-1} x_n \ \cos(-2\pi kn/N) + \sum_{n=0}^{N-1} i \ x_n \ \sin(-2\pi kn/N) \tag{4.5}$$

Matricialmente,

$$\boldsymbol{X} = \boldsymbol{F}_C\boldsymbol{x} + i\boldsymbol{F}_S\boldsymbol{x}$$

La figura 4.1 muestra el modelo neuronal propuesto. Se puede ver que la salida de la capa de convolución contiene los valores de $\boldsymbol{F}_C\boldsymbol{x}$ y $\boldsymbol{F}_S\boldsymbol{x}$. Estos se obtienen realizando la convolución de la entrada por cada uno de los $2K$ (400) *kernels*.

En el caso de la asignación directa de los pesos sinápticos (sin entrenamiento), a los primeros K *kernels* se les asignan en orden los elementos de las K columnas de la

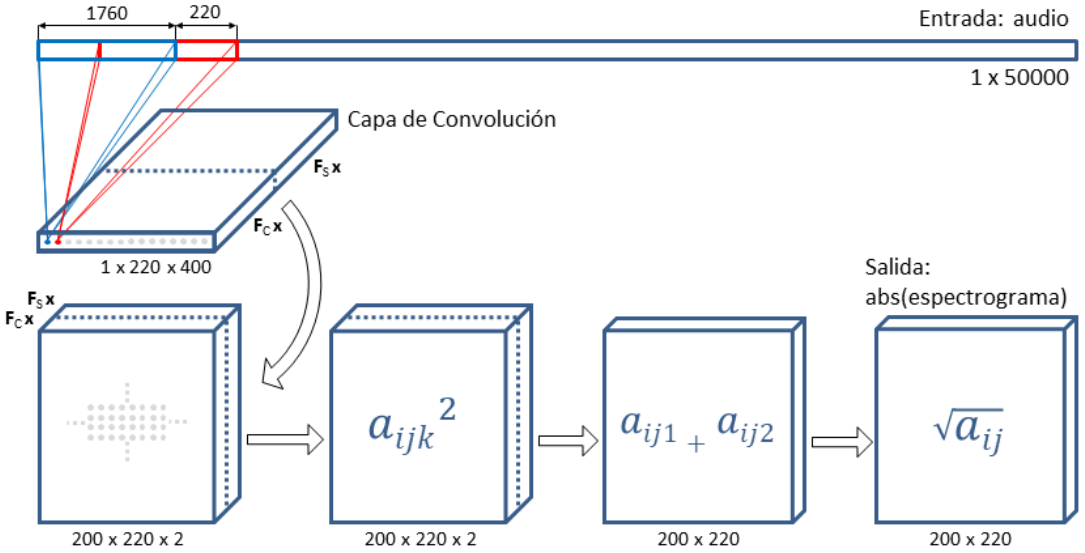


FIGURA 4.1: Modelo de red neuronal artificial que recibe audio como entrada y predice la magnitud del espectrograma.

matriz $\boldsymbol{F}_C$ y al resto los de $\boldsymbol{F}_S$. En este modelo las neuronas no utilizan el peso w_{0j} porque la DFT no tiene término independiente. Formalmente, la asignación de los pesos se realiza de la siguiente manera:

$$w_{ijk} = f_{Cik} \tag{4.6}$$

$$w_{ij(k+K)} = f_{Sik} \tag{4.7}$$

donde

w_{ijk} es el peso sináptico de la entrada i de la neurona j del *kernel* k.

f_{Cik} es el elemento de la fila i, columna k de $\boldsymbol{F}_C$.

f_{Sik} es el elemento de la fila i, columna k de $\boldsymbol{F}_S$.

Como los pesos son compartidos entre las neuronas del mismo *kernel*, solo se almacena un set de pesos por *kernel*. Notar que el subíndice j no se encuentra en los segundos términos de las ecuaciones 4.6 y 4.7. La matriz de pesos W tiene entonces tamaño $N \times 2K$ (1760×400).

Magnitud del espectro de frecuencias. Partiendo de la ecuación 4.5, la magnidud del espectro de frecuencias es la siguiente.

$$
\begin{aligned}
|X_k| &= \left| \sum_{n=0}^{N-1} x_n \ \cos(-2\pi kn/N) + \sum_{n=0}^{N-1} i \ x_n \ \sin(-2\pi kn/N) \right| \\
&= \sqrt{\left(\sum_{n=0}^{N-1} x_n \ \cos(-2\pi kn/N) \right)^2 + \left(\sum_{n=0}^{N-1} x_n \ \sin(-2\pi kn/N) \right)^2}
\end{aligned} \tag{4.8}
$$

Los resultados de las sumatorias de la ecuación 4.8 son los escalares en la posición k de los vectores $\boldsymbol{F}_C\boldsymbol{x}$ y $\boldsymbol{F}_S\boldsymbol{x}$. Por lo tanto,

$$
|X_k| = \sqrt{(\boldsymbol{F}_C\boldsymbol{x})_k^2 + (\boldsymbol{F}_S\boldsymbol{x})_k^2}
$$

donde $(\boldsymbol{F}_C\boldsymbol{x})_k$ y $(\boldsymbol{F}_S\boldsymbol{x})_k$ son los elementos de la posición k en los vectores $\boldsymbol{F}_C\boldsymbol{x}$ y $\boldsymbol{F}_S\boldsymbol{x}$ respectivamente.

La salida de la capa de convolución del modelo de la figura 4.1 es un arreglo que contiene los vectores $\boldsymbol{F}_C\boldsymbol{x}$ y $\boldsymbol{F}_S\boldsymbol{x}$ correspondientes a todos los segmentos de la señal de entrada. Después de la convolución se realizan cuatro operaciones dispuestas en capas. La primera operación es un cambio en la forma del arreglo para simplificar la tercera operación, la segunda calcula el cuadrado de cada elemento, la tercera suma los pares de valores correspondientes a la misma frecuencia (k) y al mismo segmento de tiempo, y por último, se realiza el cálculo de la raíz cuadrada de cada elemento. El resultado es una matriz de tamaño 200×220 con las magnitudes de los elementos del espectrograma.

Entrenamiento. Tal como se mencionó anteriormente, en lugar de asignar los pesos de forma directa, se puede someter a la red a un proceso de entrenamiento. Esta posibilidad es importante porque implica que la red puede ser inicializada mediante la asignación de pesos (aleatorios o no) y después adaptarlos a necesidades particulares del problema de clasificación. Tener en cuenta que esta posibilidad también incluye a los pesos de capas anteriores (si hubiera) al cálculo del espectrograma.

El cálculo del gradiente incluye las derivadas de las tres últimas etapas del modelo. En la siguiente sección se presentan los resultados de este proceso y se comparan con los obtenidos mediante la asignación directa.

4.1.3. Resultados

A continuación se exponen los resultados de 30000 ciclos de entrenamiento del modelo propuesto. Los pesos se inicializaron con valores aleatorios entre -10^6 y 10^6 uniformemente distribuidos. Se utlizó el método de optimización Adam (*Adaptive Moment Estimation*) [87], una variante del método del gradiente descendiente, con los parámetros provistos por los autores y las actualizaciones de los pesos se realizaron en lotes de tamaño 300 (la totalidad de datos de entrenamiento). Los cálculos se realizaron sobre una GPU NVIDIA Titan Xp.

El MSE de validación alcanzado fue $1{,}41 \times 10^{-6}$ ($9{,}79 \times 10^{-6}$ % del valor medio de la salida esperada), mientras que para el mismo modelo con los pesos asignados de forma directa se logra un MSE $< 10^{-9}$. En la Fig. 4.2 se muestra la salida esperada para uno de los datos de validación y la salida obtenida por la red después del entrenamiento. A simple vista no hay diferencias.

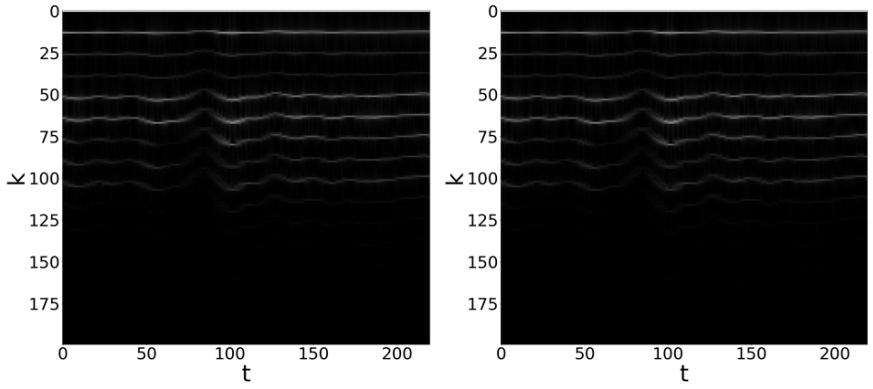


FIGURA 4.2: Espectrogramas de salida de la red neuronal. Salida esperada (izquierda) y salida obtenida con pesos entrenados (derecha).

El resultado obtenido mediante entrenamiento es ampliamente satisfactorio. A continuación se realiza una comparación entre los pesos teóricos calculados a partir de las ecuaciones de la DFT y los entrenados.

En la Fig. 4.3 se pueden ver los valores de $\boldsymbol{F}_C$ y $\boldsymbol{F}_S$ para pesos asignados de forma directa (teóricos) y para pesos entrenados. En todos los casos se observa claramente que las filas superiores de las matrices transpuestas representan a las frecuencias bajas y las inferiores a las frecuencias altas.

Es evidente también que los dos grupos, pesos teóricos y entrenados, presentan patrones de imagen distintos. Los pesos entrenados tienen una apariencia más "desordenada". El origen de este fenómeno es que DFT realiza una descomposición de la entrada en una suma ponderada de las señales sinusoidales que se encuentran en las matrices $\boldsymbol{F}_C$ y $\boldsymbol{F}_S$. El método de entrenamiento, para cada valor de k, encuentra un par $\boldsymbol{F}_C$ y $\boldsymbol{F}_S$ formado

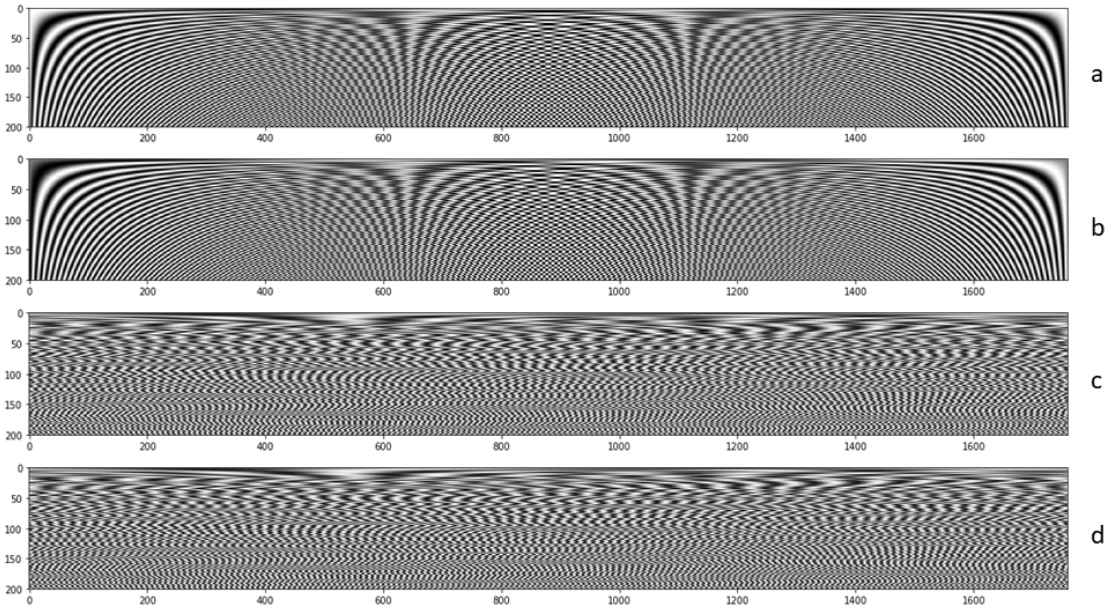


FIGURA 4.3: Matrices transpuestas de coeficientes. $\boldsymbol{F}_C$ con pesos teóricos (a), $\boldsymbol{F}_S$ con pesos teóricos (b), $\boldsymbol{F}_C$ con pesos entrenados (c) y $\boldsymbol{F}_S$ con pesos entrenados (d).

por ondas sinusoidales defasadas en 90° que permite la descomposición deseada, pero esta solución no es única, por lo tanto no se llega necesariamente a los mismo coeficientes de la ecuación 4.5.

La Fig. 4.4 muestra algunos ejemplos de pesos entrenados vs. teóricos para valores particulares de k. Observar que, tanto entre los pares de pesos teóricos como entre los pares de pesos entrenados, se respeta siempre el defasaje de 90° entre $\boldsymbol{F}_C$ y $\boldsymbol{F}_S$. Esto se puede comprobar calculando $mod = \sqrt{\boldsymbol{F}_C^2 + \boldsymbol{F}_S^2}$ para cualquier valor de k. Para los pesos teóricos, lógicamente $mod = 1$, mientras que para los pesos entrenados se obtiene siempre un valor muy cercano a 1. De esta forma, se encuentra una base ortogonal para realizar la descomposición.

4.1.4. Conclusiones

Se concluye que un modelo neuronal es capaz de calcular la DFT, tanto para pesos teóricos como entrenados, y que los pesos entrenados no necesariamente tienden hacia los teóricos, aunque comparten frecuencia y la condición de ortogonalidad. Además, una red convolucional con las características presentadas puede entrenarse para calcular el espectrograma de la señal de entrada. Si la salida esperada es la magnitud del espectrograma, es posible evitar las operaciones de números complejos mediante la adición de operaciones en capas que calculen la distancia euclídea entre los componentes de una misma frecuencia.

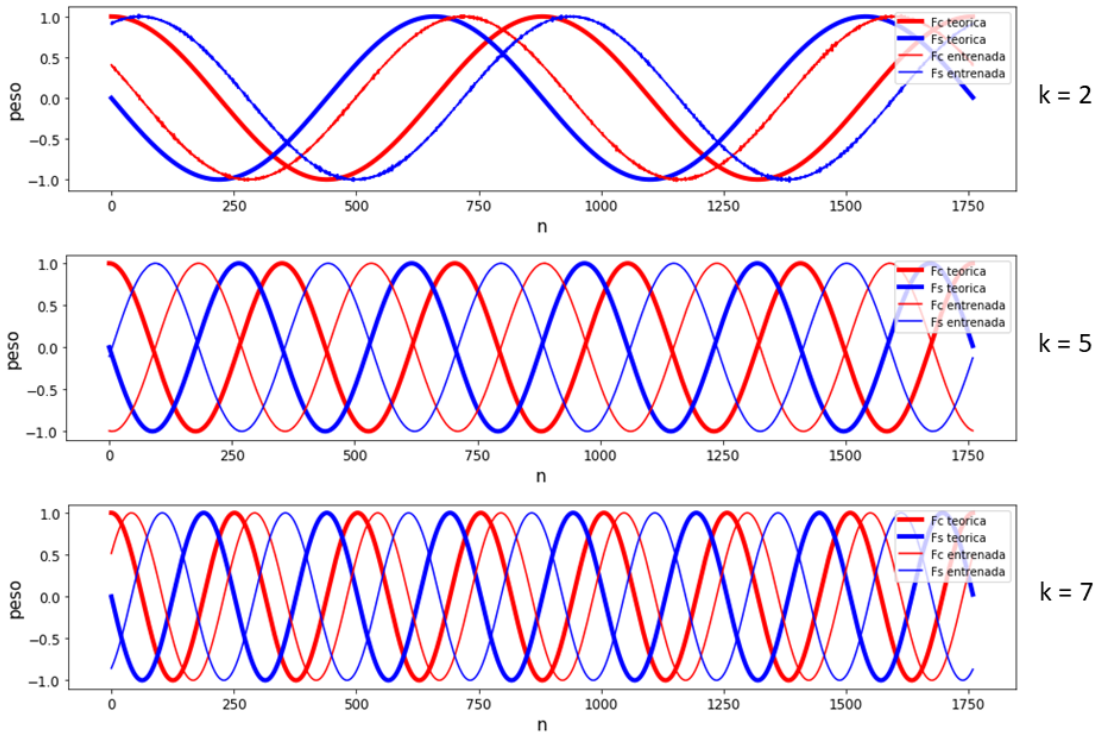


FIGURA 4.4: Pesos sinápticos entrenados (lineas finas) y teóricos (lineas gruesas) para $k = 2$, 5 y 7. Los valores de $\boldsymbol{F}_C$ se muestran en rojo y los de $\boldsymbol{F}_S$ en azul.

La ventaja de entrenar el modelo es que se puede adaptar a poblemas particulares. Por ejemplo, es común utilizar una ventana que suavice los extremos de la señal cuando se calcula la STFT (*windowing*). Estas funciones se puede lograr con la red propuesta atenuando los extremos de los pesos. Existen varias funciones de ventana, pero para un problema particular puede ser mejor una distinta.

Para resolver un problema particular, como la predicción del grado general de disfonía, los pesos se podrían inicializar con valores aleatorios, con los pesos teóricos de la DFT o con pesos teóricos modificados por alguna función (ventana por ejemplo) y luego reentrenar para encontrar la combinación óptima para el problema.

4.2. Cálculo del cepstrum

Considerando el espectro de frecuencias como si fuera una señal, su información espectral puede ser analizada con el objeto de encontrar patrones no evidentes en el dominio de la frecuencia. En consecuencia, se crea un nuevo nivel de análisis espectral, el análisis cepstral, cuya representación (nuevamente en el dominio del tiempo) es llamada cepstrum.

En esta sección se presenta una red neuronal que calcula el espectro de frecuencias (*power spectrum*), el espectrograma y el cepstrum (*power cepstrum*) en el tiempo, es decir, el cepstrograma. Se realizan pruebas de entrenamiento con datos de audio y se analiza la capacidad de adaptación del modelo con el objetivo de establecer si este puede ser utilizado en las capas de extracción de características de un modelo de aprendizaje profundo mayor.

4.2.1. Cepstrum

El *power cepstrum*, o simplemente cepstrum, es el *power spectrum* del logaritmo del *power spectrum* de la señal [88]. Entre otras aplicaciones, se utiliza en análisis de señales para encontrar la "frecuencia" de ocurrencia de los armónicos en el espectro. La variable independiente del cepstrum representa una nueva frecuencia en el dominio de la frecuencia, lo que resulta en una variable en el dominio del tiempo. Con el objeto de evitar confusiones, pero enfatizando la conexión con conceptos familiares, Bogert *et al.* llamaron a esta variable con el nombre de *quefrency* ("*frequency*" cambiando el orden de las primeras sílabas), utilizando el mismo criterio elegido para el nombre "cepstrum", "*spectrum*" con las cuatro primeras letras invertidas [89].

El cepstrum fue definido en [90] como:

$$C = \left| \mathcal{F}^{-1} \left\{ \log \left(|\mathcal{F}\{f(x)\}|^2 \right) \right\} \right|^2 \tag{4.9}$$

Notar que, como $log\left(|\mathcal{F}\{f(x)\}|^2\right)$ es una función par de la frecuencia, el seno (parte imaginaria) de $\mathcal{F}^{-1}$ se anula y, por lo tanto, el cepstrum es igual al cuadrado de la transformada coseno de $log\left(|\mathcal{F}\{f(x)\}|^2\right)$ multiplicado por $1/N$. Por la misma razón también es frecuente calcular el cepstrum como:

$$C = \frac{1}{N} \left| \mathcal{F} \left\{ \log \left(|\mathcal{F}\{f(x)\}|^2 \right) \right\} \right|^2$$

El factor $1/N$ proviene de la inversa de la DFT (IDFT). Para las aplicaciones más comunes del cepstrum, que implican la detección de su pico máximo, o el cálculo de la relación entre la energía del pico y la energía total, este factor se puede eliminar.

El uso del cepstrum es habitual en el análisis automático de la voz. La ubicación de su pico máximo (en cierto rango), se utiliza para determinar F_0 [91]. Además, la relación entre las magnitudes del pico del cepstrum y el resto del cepstrum se usa para determinar ciertas características de la calidad vocal. Esta relación permite también clasificar voces

según su nivel de ruido, soplosidad y nasalidad [92, 93]. Por otro lado, los coeficientes cepstrales de las frecuencias de mel (MFCC) tienen gran utilidad en reconocimiento del habla [72].

Si bien en esta sección se calcula el *power cepstrum*, es importante mencionar que existen otras versiones de cepstrum tales como *real cepstrum*, *complex cepstrum* y *phase cepstrum*. La base del cálculo de las versiones mencionadas es la misma, por lo tanto, estas pueden ser calculadas también con redes neuronales. Otra operación con un cálculo similar al del cepstrum es la autocorrelación, la que también ha sido utilizada en algunos casos para determinar F_0 en voces [94]. La autocorrelación es el espectro del *power spectrum* [88], por lo tanto, el cuadrado de la autocorrelación se puede calcular entonces de la misma forma que el *power cepstrum*, pero sin aplicar el logaritmo.

4.2.2. Experimento

4.2.2.1. Datos

La red neuronal se creó para predecir el *power cepstrum* de señales de audio de dos segundos de duración. Al igual que en el caso anterior, se utilizaron audios de la base de datos HUPA, presentada en la sección 3.3.2.

Entradas. Los audios están grabados en formato WAV con una tasa de muestreo de 25000 muestras por segundo. Para todos los archivos con duración mayor que 2 segundos se tomaron las 50000 muestras centrales. Los datos de entrada resultaron 430 vectores de tamaño $L = 50000$, de los cuales fueron seleccionados aleatoriamente 300 para entrenamiento 130 para validación.

Salidas. El cálculo de las salidas se realizó, de acuerdo con la ecuación 4.9, de la siguiente forma:

En primer lugar, se calculó el espectrograma como el cuadrado del valor absoluto de la STFT de las entradas con segmentos de tamaño $N = 880$ y superposición de $m = 660$ elementos, lo que implica un desplazamiento de 220 elementos en cada transformación. El espectrograma calculado de esta forma tiene 224 columnas (tiempo) por 880 filas (frecuencia). Después, la salida se calculó como el cuadrado del valor absoluto de la IDFT del logaritmo de cada uno de los *power spectrums* (columnas) del espectrograma. La salida queda definida como vectores de tamaño 880×224.

4.2.2.2. Red neuronal

En esta sección se presenta el modelo propuesto para calcular el cepstrum. Más adelante se realizarán pruebas de entrenamiento sobre este modelo y dos modificaciones.

Para facilitar la comprensión del modelo, este se divide en dos partes: la primera calcula el *power spectrum* y la segunda el *power cepstrum*. A su vez, cada parte debe cumplir con dos funciones, el cálculo de la DFT y el cálculo de las demás operaciones (módulo, cuadrado y logaritmo). Para entrenar la red es necesario calcular el gradiente del error. Entonces, las operaciones mencionadas se incluyen como capas de la red y la retropropagación del error se realiza a través de cada capa considerando la derivada de la función calculada en cada una de ellas.

Primera capa de convolución. El cálculo de la STFT del *power spectrum* se hace con una capa de convolución, que salvo por la cantidad y el tamaño de los *kernels*, es igual a la capa de convolución del cálculo del espectrograma de la sección 4.1.2.2. En este caso se utilizan 1760 *kernels* (880 con los coeficientes de $\boldsymbol{F}_C$ y 880 con los de $\boldsymbol{F}_S$) de tamaño 880. La figura 4.5 muestra el modelo neuronal completo. Se puede ver que la salida de la primera capa de convolución resulta un vector de tamaño 224×1760.

En el caso de asignación directa de pesos sinápticos (sin entrenamiento), a los primeros $K = 880$ *kernels* se le asignan los elementos de las K filas de la matriz $\boldsymbol{F}_C$ y al resto, los coeficientes de $\boldsymbol{F}_S$.

Cuadrado de la magnitud del espectro de frecuencias. La magnitud del espectro de frecuencias $|X_k|$ se calcula en la ecuación 4.8. Notar que las operaciones de números complejos han sido eliminadas. Los resultados de las sumatorias de la ecuación 4.8 son los escalares en posición k de los vectores $\boldsymbol{F}_C\boldsymbol{x}$ y $\boldsymbol{F}_S\boldsymbol{x}$. Entonces,

$$|X_k| = \sqrt{(\boldsymbol{F}_C\boldsymbol{x})_k^2 + (\boldsymbol{F}_S\boldsymbol{x})_k^2}$$

donde $(\boldsymbol{F}_C\boldsymbol{x})_k$ y $(\boldsymbol{F}_S\boldsymbol{x})_k$ son los elementos en posición k de los vectores $\boldsymbol{F}_C\boldsymbol{x}$ y $\boldsymbol{F}_S\boldsymbol{x}$ respectivamente. Entonces, el *power spectrum* se calcula como

$$|X_k|^2 = (\boldsymbol{F}_C\boldsymbol{x})_k^2 + (\boldsymbol{F}_S\boldsymbol{x})_k^2$$

La salida de la primera capa de convolución en el modelo de la figura 4.5 es un arreglo que contiene los vectores $\boldsymbol{F}_C\boldsymbol{x}$ y $\boldsymbol{F}_S\boldsymbol{x}$ correspondientes a todos los segmentos de la

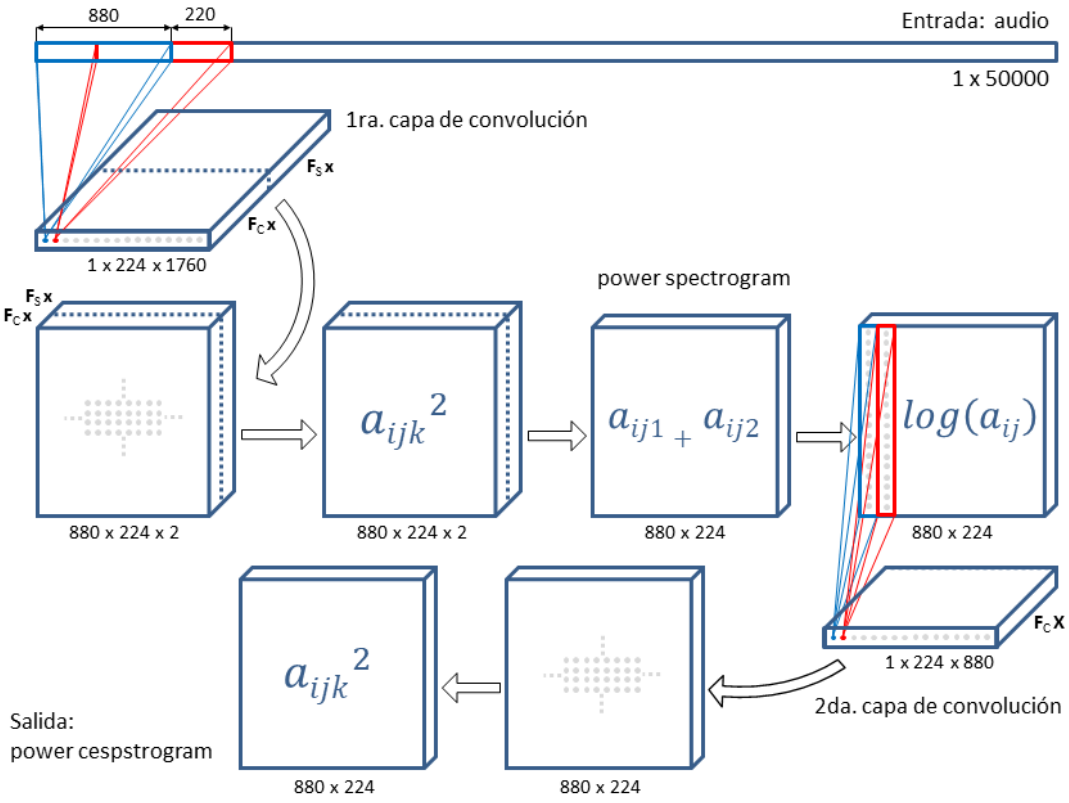


FIGURA 4.5: Modelo de red neuronal artificial que recibe audio como entrada y calcula el *power cepstrum*.

señal de entrada. Después de la convolución se realizan tres operaciones dispuestas en capas: 1) Un cambio en la forma del arreglo para simplificar la tercera operación (capa 2), 2) El cuadrado de cada elemento (capa 3) y 3) La suma de los pares de valores correspondientes a la misma frecuencia (k) y al mismo segmento de tiempo (capa 4). El resultado es una matriz de tamaño 880×224 con los elementos del espectrograma.

Segunda capa de convolución. La función de la segunda capa de convolución es el cálculo de la IDFT del logaritmo del *power spectrum*. Recibe como entrada la salida de la quinta capa, que calcula el logaritmo de los elementos del espectrograma (880×224). A diferencia de la primera convolución, en esta capa la entrada tiene dos dimensiones: frecuencia $\times$ tiempo. Cada *kernel* tiene el tamaño de una columna de la matriz de entrada, 880 (*power spectrum* completo)$\times 1$. De esta forma, para cada segmento de tiempo se calcula:

$$C_k^* = \frac{1}{N} \sum_{n=0}^{N-1} |X_n|^2 \ \cos(-2\pi kn/N)$$

o matricialmente,

$$\boldsymbol{C}^* = \frac{1}{N}\boldsymbol{F}_C|\boldsymbol{X}|^2$$

donde $\boldsymbol{C}^*$ es el resultado de la transformada inversa. En la sección 4.2.1 se explicó por qué, para este caso, la IDFT puede ser reemplazada por la transformada coseno discreta (DCT por *Discrete Cosine Transform*).

En caso de asignación directa de pesos, $w_{ijk} = \frac{1}{N} f_{Cik}$.

Cuadrado de la segunda convolución. Finalmente, el *power cepstrum* se obtiene en la octava capa. Esta calcula el cuadrado de la salida de la segunda capa de convolución. $C_k = C_k^{*2}$.

4.2.2.3. Entrenamiento

Al igual que en el caso del espectrograma (sección 4.1.2.2), los pesos sinápticos se pueden adaptar mediante el método del descenso por el gradiente. El cálculo del gradiente incluye las derivadas del logaritmo, suma y cuadrado.

4.2.2.4. Variantes del modelo

La quinta capa del modelo de la figura 4.5 calcula el logaritmo del *power spectrum*. Las características de la derivada de $\log(z)$ (valores muy altos para $0 < z < 1$ y cercanos a cero para $z > 1$) dificultan el entrenamiento. Por lo tanto, puede ocurrir que los pesos de la primera capa de convolución no puedan ser entrenados. Con el objeto de estudiar la capacidad de entrenamiento de la red neuronal considerando la situación mencionada, se hacen pruebas de entrenamiento sobre dos variantes del modelo:

- Modelo I. Calcula el *power cepstrum*. Es el modelo completo mostrado en la figura 4.5.
- Modelo II. Se modifica el modelo I removiendo la quinta capa (logaritmo). El modelo resultante retorna el cuadrado de la autocorrelación.

4.2.2.5. Condiciones iniciales

En la sección 4.1 se mostró que una red neuronal con una capa de convolución puede calcular la magnitud del espectro de frecuencias $|X_k|$. Los modelos I y II tienen dos

capas con pesos en lugar de una, por lo tanto el entrenamiento debería ser más difícil que el del modelo de la sección 4.1. Con el objeto de hacer un análisis más detallado del entrenamiento de los modelos I y II, las pruebas se realizan con tres condiciones iniciales diferentes:

- R: Inicializando los pesos con valores aleatorios. Esta condición supone mayor dificultad que las condiciones siguientes porque los pesos se encuentran más lejos de los valores óptimos.
- N: Inicializando los pesos con los valores teóricos y adicionando un ruido del 10 %.
- W: Inicializando los pesos con los valores teóricos, pero reemplazando la salida esperada por el *power cepstrum* (o el cuadrado de la autocorrelación, según corresponda) del audio multiplicado por una ventana Tukey (*tapered cosine window*) con $r = 0{,}25$.

La figura 4.6 muestra un ejemplo de un audio particular, de la función ventana y del audio multiplicado por la ventana para ilustrar el cambio introducido en la condición W.

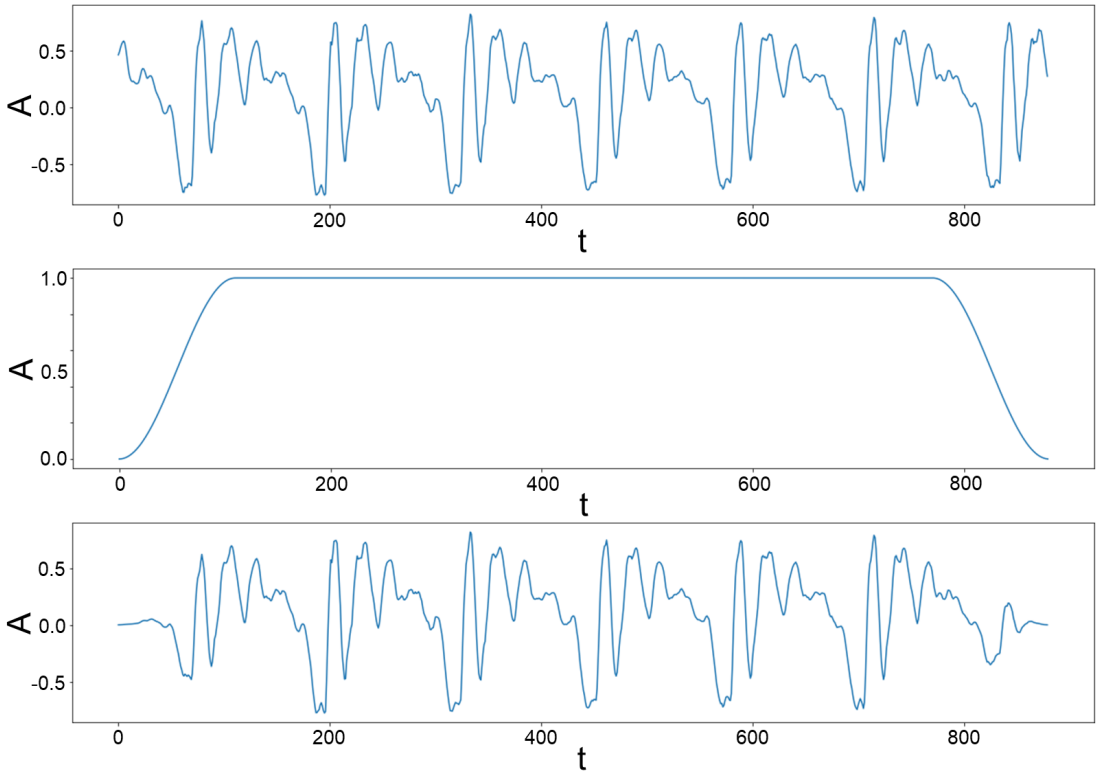


FIGURA 4.6: Amplitud (A) en el tiempo (t) de 880 muestras (0.0352 s) de una vocal /a/ sostenida (arriba). Ventana Tukey para $r = 0{,}25$ (centro). Audio multiplicado por ventana Tukey (abajo).

Adicionalmente, se realizan las siguientes pruebas en ambos modelos para cada una de las condiciones iniciales:

- TCM: Entrenamiento del modelo completo.
- CL1: Se entrena únicamente la primera capa de convolución. Se utiliza el modelo completo. La segunda capa de convolución se inicializa con pesos teóricos. Después, el modelo se entrena sin modificar los pesos de la segunda capa de convolución.
- CL2: Se entrena únicamente la segunda capa de convolución. Se utiliza el modelo completo. La primera capa de convolución se inicializa con pesos teóricos. Después, el modelo se entrena sin modificar los pesos de la primera capa de convolución.

El objetivo de estas tres pruebas es determinar, en caso de que el modelo completo no pueda ser entrenado con éxito, si cada una de las capas de convolución puede alcanzar pesos óptimos cuando la otra capa ya está adaptada.

Para todos los casos , cuando los pesos son inicializados aleatoriamente, estos tienen una distribución uniforme entre -0.001 y 0.001.

Durante el entrenamiento se utiliza el método de optimización Adam con $\alpha = 0{,}001$, $\beta_1 = 0{,}9$ y $\beta_2 = 0{,}999$. Los pesos son modificados en lotes de 300 vectores de entrenamiento (el total de datos disponibles).

4.2.3. Resultados

Las tablas 4.1 y 4.2 muestran los errores obtenidos durante el entrenamiento de los modelos I y II respectivamente.

Como los valores de las salidas no están estandarizados y las salidas de los modelos comparados tienen magnitudes diferentes, en lugar de mostrar el MSE, se utiliza la relación entre el error absoluto medio y el valor medio de la salida. Los espacios vacíos en las tablas significan que el entrenamiento no fue exitoso.

4.2.3.1. Modelo I

Para el modelo I, solo se pudieron entrenar los casos CL2-R y CL2-N. En el primer caso, los pesos de la primera capa de convolución se inicializaron con pesos teóricos y permanecieron fijos, mientras que los de la segunda capa de convolución se inicializaron con valores aleatorios y se entrenaron. El segundo caso es similar, pero los pesos de la

Tabla 4.1: Razón entre el error absoluto medio el valormedio de la salida para el modelo I de acuerdo a las condiciones definidas en la sección 4.2.2.5.

	R	N	W
TCM	-	-	-
CL1	-	-	-
CL2	$1.17\ 10^{-5}$	$5.27\ 10^{-6}$	-

Tabla 4.2: Razón entre el error absoluto medio el valormedio de la salida para el modelo II de acuerdo a las condiciones definidas en la sección 4.2.2.5.

	R	N	W
TCM	-	$1.33\ 10^{-4}$	$6.46\ 10^{-4}$
CL1	$1.46\ 10^{-4}$	$7.89\ 10^{-5}$	$1.73\ 10^{-5}$
CL2	$1.22\ 10^{-5}$	$5.50\ 10^{-6}$	-

segunda capa se inicializaron con pesos teóricos y se agregó ruido. No es de extrañar que estos casos sean entrenables, ya que el cálculo es similar al del *power spectrum* porque hacen el cuadrado de la magnitud DFT.

En ninguno de los casos de TCM y CL1 se pudo entrenar el modelo I. Para estos casos, es necesario modificar los pesos de la primera capa de convolución y, por lo tanto, el error debe propagarse a través de la quinta capa (logaritmo). Debido a las características ya mencionadas de la derivada del logaritmo, la información propagada no permite una búsqueda adecuada por descenso de gradiente. Por lo tanto, los casos de TCM y CL1 no se pueden entrenar.

El caso CL2-W tampoco puede ser entrenado. En la siguiente sección se explican las razones.

4.2.3.2. Modelo II

Para el modelo II (sin logaritmo), el único caso que no se pudo entrenar es el TCM-R, el modelo completo con pesos inicializados aleatoriamente. El resto de los casos (excepto por CL2-W) pudieron ser entrenados exitosamente. Esto significa que, cuando el modelo II es inicializado con pesos cercanos a los óptimos (por ejemplo los pesos teóricos), este puede ser entrenado para adaptarse a un problema particular. Esto también confirma que el problema en el modelo I es la derivada del logaritmo.

Notar que en el caso CL2-W el modelo no puede ser entrenado porque la solución requiere la modificación de los pesos de la primera capa de convolución. La figura 4.7 muestra algunos de los pesos de la primera capa de convolución para CL1-W. Se puede ver claramente que los pesos definitivos tienden al producto entre los pesos originales y la ventana utilizada. Entonces, si no se permite la modificación de los pesos de la primera capa de convolución, en el caso CL2-W (tanto para el modelo I como para el modelo II) la red no se puede entrenar hasta un resultado satisfactorio.

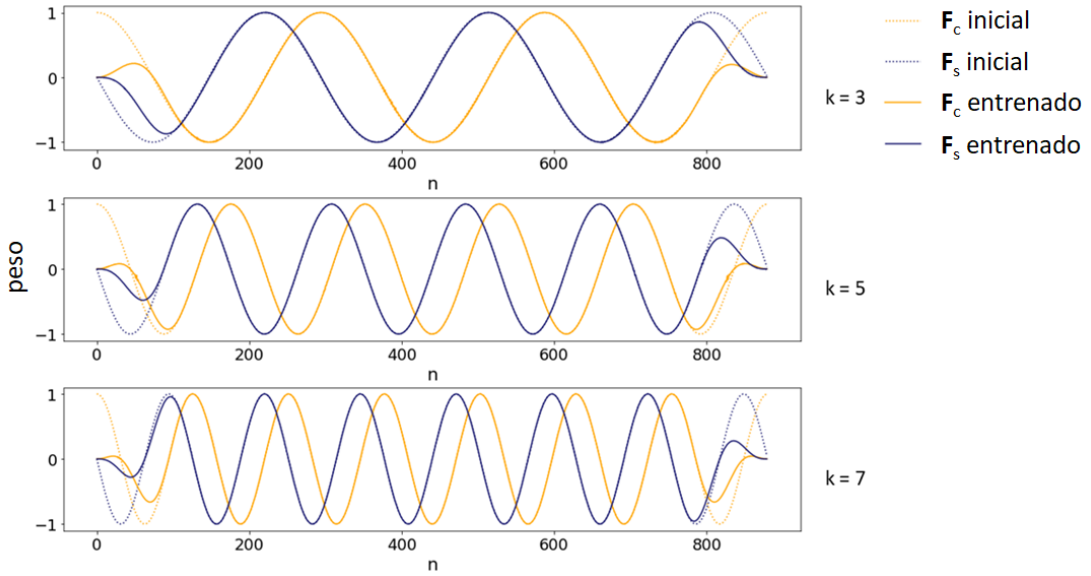


FIGURA 4.7: Pesos entrenados vs. pesos iniciales de los *kernels* 3, 5 y 7 del modelo II para para el caso CL1-W.

4.2.4. Conclusiones

Se concluye que el modelo completo presentado es capaz de calcular el *power cepstrum* correctamente, pero no se puede entrenar ni tiene la capacidad de adaptarse a otros problemas mediante el entrenamiento debido a que la derivada de la función logaritmo no es adecuada para retropropagar el error hacia la primera capa de convolución.

Si bien el objetivo principal del trabajo presentado en esta sección es determinar la estructura de un modelo neuronal que calcule el *power cepstrum* y estudiar el comportamiento del método de entrenamiento, un aporte importante surge del modelo alternativo analizado. La red neuronal sin el logaritmo, sí se puede entrenar para adaptarse a otros casos. Este modelo alternativo, equivalente a la función cuadrado de la autocorrelación, mantiene algunas de las ventajas del cepstrum, como por ejemplo, información para detectar la frecuencia fundamental de la señal. La ventaja de que este modelo sea entrenable es que se puede adaptar a problemas particulares. Esto queda demostrado con la adaptación de los pesos teóricos a la salida calculada sobre el producto del audio por la

ventana Tukey. La red se inicializó con los pesos teóricos, una combinación mucho mas cercana al óptimo que los pesos aleatorios. Luego, mediante entrenamiento, los pesos de la primera capa de convolución convergieron hacia el producto entre la ventana Tukey y los pesos originales.

4.3. *Windowing*

En esta sección se desarrolla una red neuronal que realiza la operación de *windowing*. El modelo presentado se puede adaptar, mejorando así los resultados de la clasificación. Este aporte, al igual que los dos anteriores, se puede aplicar en cualquier tarea de reconocimiento de patrones en audio que utilice redes neuronales profundas, no solo en la clasificación de la calidad vocal. El reconocimiento de patrones de audio se aplica a una amplia gama de tareas, como el reconocimiento automático de voz (ASR por *automatic speech recognition*) [73, 74, 95–112], el reconocimiento del hablante [113–117], el reconocimiento de emociones [118, 119], la detección de enfermedades [120] y la recuperación de información musical [121–124] entre otras. Para dar mayor visibilidad al trabajo desarrollado y facilitar que otros investigadores puedan reproducir los resultados, los experimentos se realizaron en un dominio distinto al de la calidad vocal.

La base de datos elegida fue *Speech Commands* (ver sección 3.3.3). Este juego de datos está disponible gratuitamente y en [79] se define un protocolo preciso para realizar pruebas sobre sus datos. Además, el código fuente de *Simple Audio Recognition* (SAR), un proyecto conocido que usa el conjunto de datos *Speech Commands*, también está disponible[1]. SAR incluye funciones de preprocesamiento de datos según el protocolo de la sección 4.3.2.2 y un modelo redes neuronales profundas basado en [125] para clasificar los datos de *Speech Command*.

El desarrollo logrado en estas sección se publicó en [126]. El modelo propuesto es una modificación del modelo neuronal de SAR, las funciones para preprocesar los datos también son los de SAR y para comparar los resultados se usa el modelo de SAR como referencia, por lo tanto, es fácil para cualquier investigador reproducir los resultados.

4.3.1. Operación de *windowing*

Windowing es el proceso de dividir señales largas en tramas o segmentos cortos de N muestras. Un segmento $\boldsymbol{x} = (x_0, x_1, \ldots, x_{N-1})$ se obtiene multiplicando la señal $s(n)$ por la ventana $w(n)$, formada por una secuencia de valores distintos de cero.

[1] https://github.com/tensorflow/tensorflow/tree/master/tensorflow/examples/speech_commands

$$x(n) = s(n)w(n), \qquad n = 0, \ldots, N-1$$

La DFT de una señal dividida en tramas genera valores distintos de cero para frecuencias que no existen en la señal original. Este efecto (no deseado) es usualmente llamado *spectral leakage* o manchado espectral. El tipo de información que puede ser extraída del espectro de frecuencias depende principalmente de la distribución del manchado espectral sobre el espectro. A su vez, la forma de la función de ventana determina la distribución del manchado espectral, por lo tanto, la elección de la ventana depende del objetivo de la clasificación.

La ventana más simple es la ventana rectangular $w_r = (1, 1, \ldots, 1)$. Una ventana rectangular no modifica en absoluto los valores originales. Solo se multiplica por 1 dentro de la ventana y por 0 afuera.

Se han diseñado varias funciones ventana. Cada una produce un efecto distinto sobre el manchado espectral. Algunas son Hann, Hamming, flat top, Blackman y Tukey. Una descripción detallada se puede ver en [127].

Las ventanas Hann y Hamming son las más frecuentemente utilizadas en reconocimiento de patrones sobre audio [95, 97, 101, 108–110, 116, 120, 128, 129], sin embargo, algunos investigadores han preferido utilizar otras, como la ventana Blackman-Harris [123, 124], o proponer nuevas funciones ventana. Morales-Cordovilla *et al.* [130], Alam *et al.* [131] y Rozman *et al.* [132] propusieron ventanas asimétricas, mientras que Sahidullah *et al.* [133] proponen una nueva técnica de *windowing* basada en las derivadas de la DFT.

Los modelos de aprendizaje profundo que toman el *raw audio* como entrada usan ventanas rectangulares. Se podría pensar que los filtros aprendidos incluyen la multiplicación por otra función de ventana (no rectangular), lo que implica una ventana diferente para cada filtro (algo similar a lo ocurrido en el caso CL1-W del modelo II en la sección 4.2.3.2). No se encontraron estudios sobre la mejora proporcionada por filtros y ventanas adaptables aplicados por separado en una misma red neuronal.

4.3.2. Experimento

En esta sección se estudia el comportamiento de una red neuronal profunda para el reconocimiento de comandos de audio. La red neuronal propuesta es un modelo de clasificación extremo a extremo, que recibe el *raw audio* como entrada y predice la palabra mencionada. La primera capa del modelo multiplica la entrada por una ventana que se adapta durante el entrenamiento.

Durante el experimento se intenta responder las siguientes preguntas:

- ¿Es posible adaptar los coeficientes de la ventana? Para adaptar los pesos de la ventana es necesario propagar el gradiente de la función del error desde la capa de salida a la capa que se encarga de la operación de *windowing*. Como se verá más adelante, la red neuronal propuesta calcula internamente el *power spectrum*. Por lo tanto, el gradiente debe propagarse correctamente a través del cálculo del *power spectrum*.
- Si los coeficientes de la ventana se pueden adaptar, ¿Esta adaptación mejora la capacidad de reconocimiento del modelo?

Para responder estas preguntas se realizaron pruebas sobre el *Speech Commands dataset*. Las pruebas consisten en reconocer las palabras mencionadas en los audios de la base de datos.

4.3.2.1. Modelo propuesto

Tal como se comentó anteriormente, el proyecto SAR resuelve el problema de clasificación sobre el *Speech Commands dataset* con una red neuronal profunda. SAR ofrece distintos modos de ejecución (relacionados con la complejidad y necesidad de recursos computacionales) y cada modo utiliza un modelo neuronal diferente. El modelo más grande y con mejores resultados de SAR fue elegido como modelo de referencia para el experimento. Este modelo recibe un *power spectrogram* como entrada.

Se propone una red neuronal, llamada wSTFT, que calcula el *power spectrogram* y la operación de *windowing*. Se analiza el efecto de reemplazar las entradas del modelo de referencia por la red wSTFT y adaptar los pesos de *windowing* en wSTFT (figura 4.8). Para calcular el *power spectrogram*, wSTFT debe multiplicar el *raw audio* por los coeficientes de la ventana, calcular la STFT y calcular el cuadrado de la STFT.

***Windowing* con redes neuronales.** Para una señal s (*raw audio*) de longitud L y una ventana w de tamaño N, el producto Hadamard (o producto elemento a elemento) $x = w \odot s'$ se debe calcular para cada posición de w sobre s, donde s' el segmento de s bajo w.

Al igual que en una capa de convolución, la ventana se mueve un número fijo de elementos (*strides*) en cada paso y los coeficientes w_i son los mismos para cada ubicación de la ventana (pesos compartidos). A diferencia de una capa convolucional, el resultado de cada paso es un vector, no un escalar. No existen capas estándar en la librerías de redes

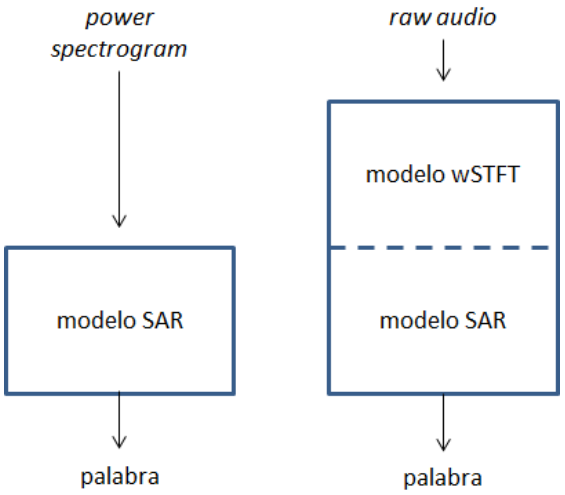


FIGURA 4.8: Modelo de referencia (izquierda) vs. modelo propuesto (derecha).

neuronales artificiales con este comportamiento, por lo tanto fue necesario programarla. En la figura 4.9 se muestra el funcionamiento de la nueva capa, STHadamard, resaltando los pesos compartidos mediante los colores de las conexiones. Notar que la salida es un vector de dos dimensiones donde cada columna es una trama multiplicada por la ventana.

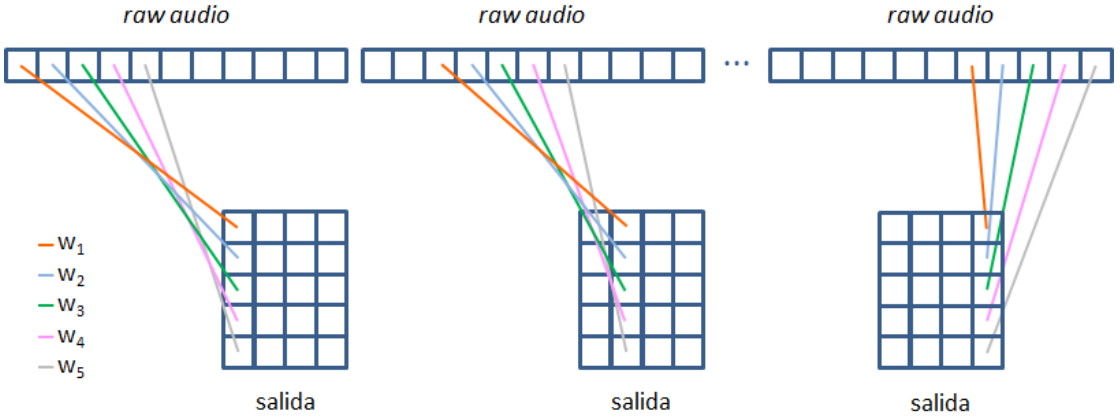


FIGURA 4.9: Esquema de cálculo de la capa STHadamard para $L = 11$, $N = 5$ y *strides*$= 2$.

En lo sucesivo, los coeficientes de la ventana serán también referidos como pesos de capa de *windowing* o de la capa STHadamard.

Cuadrado de la STFT. El cálculo se realiza de forma similar al cuadrado de la magnitud del espectro de frecuencias en la sección 4.2.2.2 (cálculo del *power cepstrum*). Hay una diferencia en la capa de convolución, como la salida de la capa STHadamard tiene dos dimensiones, ahora se utiliza una convolución 2D. El tamaño de cada *kernel* es $N \times 1$ y se desplaza una columna en cada paso.

El modelo wSTFT completo se muestra en la figura 4.10.

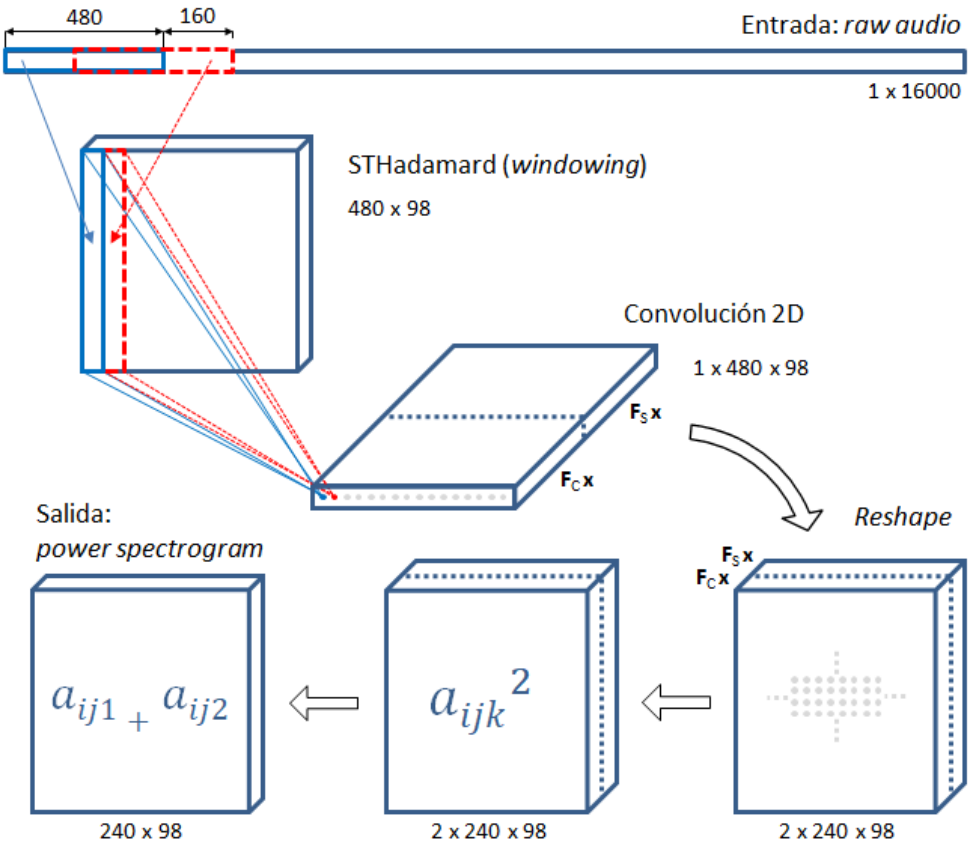


FIGURA 4.10: Modelo wSTFT completo.

4.3.2.2. Datos

Tal como se mencionó con anterioridad, el protocolo de prueba sobre los datos de *Speech Commands* se define en [79]. El protocolo especifica cuáles son los archivos de audio que deben ser utilizados para entrenar el modelo y cuáles deben usarse para evaluarlo. También se define que solo las diez palabras "*Yes*", "*No*", "*Up*", "*Down*", "*Left*", "*Right*", "*On*", "*Off*", "*Stop*", y "*Go*" deben ser clasificadas. Además, se crean dos etiquetas especiales adicionales llamadas "*Unknown Word*" y "*Silence*". Los juegos de datos quedan formados por la misma cantidad de elementos para cada una de las doce categorías, lo que significa que cada clase contiene aproximadamente el 8.3 % de los ejemplos. La categoría "*Unknown Word*" contiene palabras elegidas aleatoriamente de las clases que no son parte de las diez palabras objetivo. La categoría "*Silence*" tiene audios de un segundo de duración extraídos aleatoriamente de un grupo de archivos con ruido de fondo también provisto con la base de datos.

Durante este experimento se sigue el protocolo definido en [79]. Adicionalmente, al igual que en el proyecto SAR, los audios correspondientes a las palabras se mezclan con ruido

de fondo. También como en SAR, para todos los archivos de audio se calcula el *power spectrogram* después de multiplicar el audio por una ventana Hann. Los espectrogramas, de tamaño 240 × 98, son las entradas para el modelo de referencia, mientras que los *raw audios*, de tamaño 16000, son las entradas del modelo propuesto.

4.3.2.3. Detalles de implementación

Modelo de referencia. En la tabla 4.3 se muestran las capas del modelo de referencia. Es igual al modelo SAR, salvo por dos diferencias. La primera diferencia es que en el modelo SAR la tercera capa es una capa de *max pooling*. La segunda diferencia es el valor del parámetro *padding* en las capas de *pooling*. En el modelo SAR, *padding* = "same", mientras que en el modelo de referencia *padding* = "valid". Este úlimo cambio se debe a que se observó que algunas columnas del espectrograma no se utilizaban en el modelo SAR. Con estos cambios la exactitud del modelo de referencia mejoró aproximadamente un 3 % con respecto al modelo de proyecto SAR.

Tabla 4.3: Capas del modelo de referencia

Capa	Tamaño de salida	Parámetros
Reshape	(1, 240, 98)	0
Dropout	(1, 240, 98)	0
Average pooling 2D	(1, 40, 98)	0
Convolución 2D	(64, 14, 98)	10304
Dropout	(64, 14, 98)	0
Max pooling 2D	(64, 7, 49)	0
Convolución 2D	(64, 7, 49)	41024
Dropout	(64, 7, 49)	0
Flatten	(21952)	0
Densa	(12)	263436

Modelo propuesto. El modelo propuesto surge de insertar las capas del modelo wSTFT al principio del modelo de referencia. La tabla 4.4 muestra las capas del modelo wSTFT.

Los 230400 pesos de la capa de convolución de la tabla 4.4 se mantienen fijos. Los únicos parámetros que se entrenan en wSTFT son los 480 pesos de la capa STHadamard.

Tabla 4.4: Capas del modelo wSTFT

Capa	Tamaño de salida	Parámetros
STHadamard	(480, 98)	480
Reshape	(1, 480, 98)	0
Convolución 2D	(480, 1, 98)	230400
Reshape	(2, 240, 98)	0
Cuadrado	(2, 240, 98)	0
Suma	(240, 98)	0

Se analiza el comportamiento del modelo propuesto (*prop_model*) y de dos variantes. Las variantes son:

- *prop_model_smooth*. Se agrega una penalidad en la función del error para que la ventana tienda a tener curvas suaves. Se penalizan las altas frecuencias en los pesos de la ventana.
- *prop_model_symm*. Se fuerza la simetría de la ventana. En este caso, la capa STHadamard tiene $N/2$ pesos.

Inicialización y entrenamiento. Los pesos de la capa STHadamard se inicializaron con los coeficientes de una ventana Hann. Los pesos de la capa de convolución en wSTFT se inicializaron con los coeficientes de la DFT. El resto de los pesos se inicializó con valores aleatorios uniformemente distribuidos entre -0.001 y 0.001.

Se utilizó nuevamente el método de optimización Adam. Se ejecutaron 150 ciclos de entrenamiento (*epochs*). Para todos los ciclos, los parámetros del método Adam fueron $\beta_1 = 0{,}9$ y $\beta_2 = 0{,}999$, mientras que se usó $\alpha = 0{,}001$ durante los primeros 100 ciclos y $\alpha = 0{,}0001$ en el resto. Los pesos se adaptaron en lotes de 500 vectores de entrada.

4.3.3. Resultados

El entrenamiento de cada modelo se repitió 30 veces. La inicialización de los números aleatorios se hizo con las mismas 30 semillas para los tres modelos. La tabla 4.5 muestra la exactitud media y las mejoras relativas sobre el juego de datos de test. Las mejoras de *Prop_model_smooth* y *Prop_model_symm* no son significantes. Para *Prop_model* la mejora es pequeña, pero es importante destacar que *Prop_model* fue mejor que el modelo de referencia en las 30 ejecuciones del experimento.

TABLA 4.5: Exactitud media sobre datos de test y mejora relativa en 30 entrenamientos.

Modelo	Exactitud	Mejora relativa
Modelo de referencia	0.9121	-
Prop_model	**0.9185**	**0.7 %**
Prop_model_smooth	0.9130	0.1 %
Prop_model_symm	0.9151	0.33 %

La diferencia entre las exactitudes del modelo de referencia y *Prop_model* para los datos de entrenamiento es menor que para los datos de validación (figura 4.11). Con los datos de test, la diferencia es similar a la de los datos de validación. Esto implica que el modelo propuesto reduce el sobreajuste.

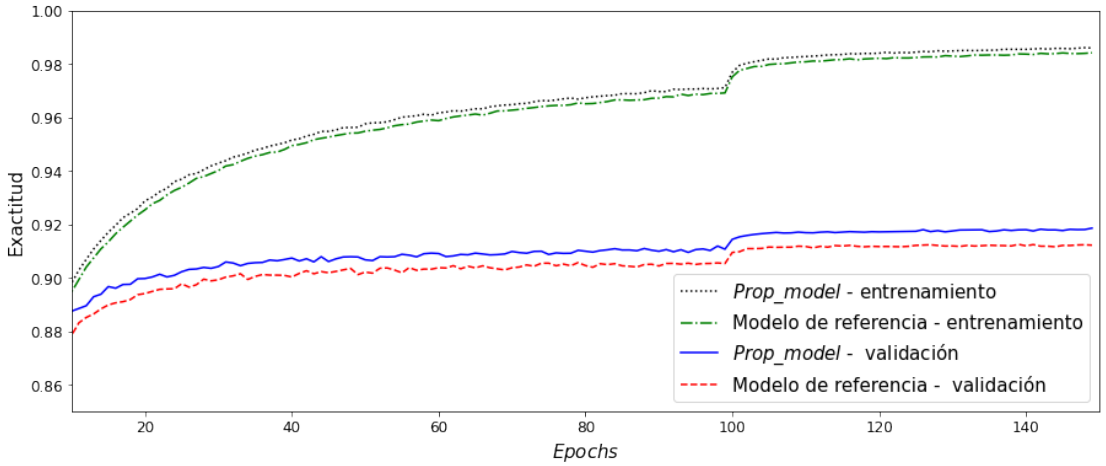


FIGURA 4.11: Evolución de la exactitud para los datos de entrenamiento y validación durante el entrenamiento de ambos modelos.

La figura 4.12 muestra un ejemplo de los pesos originales vs. entrenados de la capa STHadamard para el modelo *prop_model*. En todos los entrenamientos de este modelo se obtuvieron pesos con las mismas características, la campana se vuelve más alta y se añaden algunas frecuencias altas y medias. El modelo *Prop_model_smooth* elimina las altas frecuencias de los pesos y la curva se vuelve más suave, pero la exactitud disminuye. Esto sugiere que los componentes frecuenciales que los pesos de la capa STHadamard adquieren durante el entrenamiento son útiles para la clasificación.

4.3.4. Conclusiones

En primer lugar, se concluye que es posible adaptar los coeficientes de la ventana con el modelo propuesto, lo que implica que el gradiente de la función error se propaga

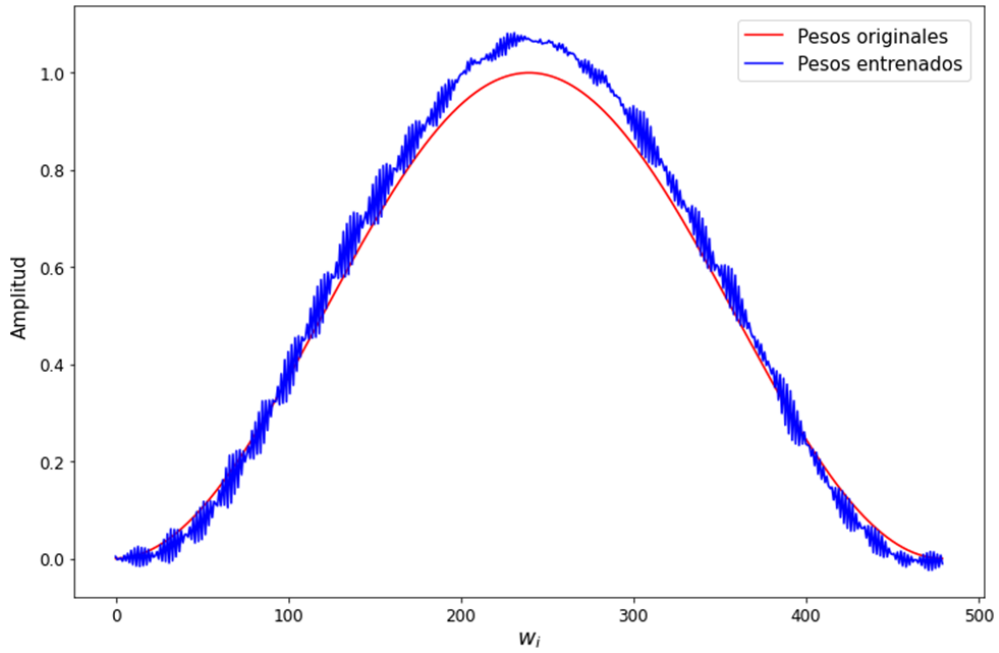


FIGURA 4.12: Pesos entrenados vs. pesos originales de la capa STHadamard del modelo *prop_model*.

correctamente a través del cálculo neuronal del *power spectrum*. En segundo lugar, se concluye que la adaptación de los pesos de la ventana pueden mejorar el rendimiento de la clasificación, aunque en una cantidad moderada.

En los coeficientes adaptados de la ventana aparece añadidas frecuencias medias y altas. No está claro qué rol cumplen estas frecuencias. Se podría pensar que son adaptaciones para clasificar algunas entradas particulares, pero dado que el modelo propuesto incrementa la capacidad de generalización (reduce el sobreajuste), parece más posible que estas frecuencias añadidas sean realmente importantes para la tarea de clasificación.

El modelo wSTFT podría ser insertado de manera directa en todas las redes neuronales que reciben el *power spectrum* como entrada. El proceso de entrenamiento es naturalmente menos eficiente, pero después del entrenamiento el modelo original puede ser utilizado cambiando la ventana inicial por la ventana entrenada, es decir, no es necesario agregar las nuevas capas al modelo de la etapa de producción.

Capítulo 5

Extracción de características y clasificación

En este capítulo se definen las características (cualidades) del modelo inicial en relación a la extracción de características (*features*) y la clasificación.

5.1. Extracción de características

Para la extracción de características, el diseño de la red neuronal se enfoca en tres particularidades de las voces patológicas, las cuales se agrupan en medidas de perturbación, que miden la variabilidad a corto tiempo de los ciclos glóticos con respecto a la frecuencia (*jitter*) y la amplitud (*shimmer*); y las de medidas de ruido, las cuales indican la presencia de ruido aditivo en la emisión de la voz. Estas son medidas tradicionales en el análisis acústico de la voz [134] y también son las de uso más frecuente en la actualidad [135].

5.1.1. Perturbaciones de amplitud

Shimmer es una medida acústica relativa a las perturbaciones de amplitud de una señal. Las variaciones de este tipo en la voz humana son perceptibles al oído y permiten caracterizar ciertas propiedades, tanto de la voz, como de las personas que la emiten [136].

El valor de *shimmer* está asociado a la calidad vocal [13, 14, 134, 137–139], estado de ánimo [140–145], edad [146] y género [147] de las personas. Existen numerosos trabajos de investigación que utilizan, entre otras, la medida *shimmer* con objetivos que van desde la detección de patologías [26, 134, 148] hasta la mejora de interfaces humano-máquina

a través de la estimación de la intencionalidad de una frase hablada [149]. Con respecto a las voces sintetizadas, en [150] se determina que cierto nivel se *shimmer* aumenta el grado de naturalidad.

El cálculo de *shimmer* tiene una complicación importante, depende de la detección previa de la frecuencia fundamental (F_0) de vibración de las cuerdas vocales. F_0 es difícil de estimar en voces patológicas [151, 152]. La estimación del valor real de F_0 continúa siendo materia de investigación [19, 21, 22, 151–154].

En esta sección se busca encontrar la arquitectura mínima de una red neuronal para realizar una aproximación del valor de *shimmer* en una señal de audio. La red neuronal presentada se entrena con audio sintetizado. Las entradas al modelo neuronal son los valores del *power spectrum* de la señal. Por un lado, se espera obtener un método de cálculo *shimmer* independiente de F_0, y por el otro, la respuesta al interrogante sobre el grado en que las perturbaciones de amplitud del audio original pueden influir en la salida de un modelo de aprendizaje profundo con entrada de datos de audio o espectrales. En otras palabras, cuánta información de *shimmer* se preserva hasta las últimas capas del modelo.

5.1.1.1. Cálculo de *shimmer*

Existen distintas versiones de *shimmer*. La mayor diferencia entre ellas es el tamaño de la ventana (cantidad de ciclos de F_0) utilizada para el cálculo. Algunas se pueden ver en [25]. La versión elegida es la de Klingholz y Martin [155], también conocida como *relative shimmer*.

Relative shimmer, de acá en más solo "*shimmer*" es una forma de medir las perturbaciones de amplitud ciclo a ciclo de la frecuencia fundamental de una señal y se muestra como una relación entre las perturbaciones y la amplitud.

$$shimmer = \frac{\frac{1}{N-1}\sum_{i=1}^{N-1}|A_i - A_{i+1}|}{\frac{1}{N}\sum_{i=1}^{N}|A_i|} \tag{5.1}$$

donde N es la cantidad de periodos de F_0 que tiene la señal y A_i es la amplitud máxima para el periodo i.

5.1.1.2. Experimento

Datos Se generaron datos de audio modulados en amplitud por una onda sinusoidal con la siguiente expresión:

$$y(t) = \frac{1}{1+k} \sin\left(\alpha + \frac{2\pi t F_0}{s}\right)\left(1 + k \sin\left(\beta + \frac{2\pi t f_{mod}}{s}\right)\right) \tag{5.2}$$

donde $y(t)$ es la señal generada, t es tiempo [s], F_0 es la frecuencia de vibración glótica [Hz], f_{mod} es la frecuencia de modulación [Hz], k es la constante de sensibilidad en amplitud del modulador, s es la frecuencia de muestreo, α y β son constantes para variar la fase de la señal a modular y de la señal moduladora respectivamente.

Para la generación de los datos de entrenamiento, validación y test, se asignaron valores aleatorios con distribución uniforme. F_0 tomó valores en el intervalo [200, 1000] Hz, f_{mod} en [5, 10] Hz, k en [0, 0.4], α y β en $[0, 2\pi]$.

En la Fig. 5.1 se pueden ver 250 ms de audio generados con $F_0 = 200$ Hz, $f_{mod} = 8$ Hz y $k = 0{,}4$.

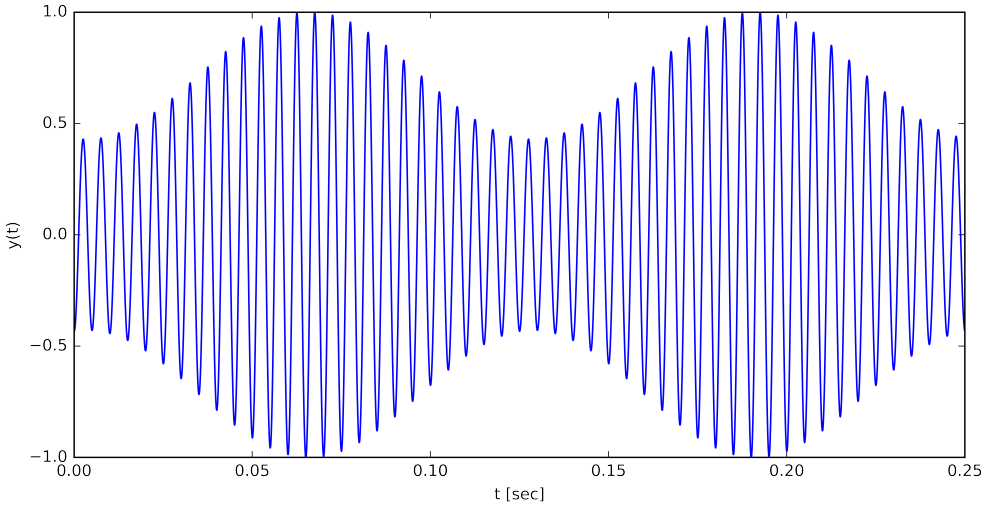


FIGURA 5.1: Audio generado para $F_0 = 200$ Hz, $f_{mod} = 8$ Hz y $k = 0{,}4$.

Se generaron tres juegos de datos, uno de entrenamiento con 2500 elementos, uno de validación con 500 elementos y uno de test con 500 elementos. Cada elemento está compuesto por el valor de *shimmer* a estimar y el espectrograma del audio generado. El valor de *shimmer* se puede calcular con exactitud porque F_0 es conocido al momento de generar el audio. El espectrograma se calcula sobre 2 s de audio generado con 44100 muestras/s. Para el cálculo se utilizó una ventana tipo Tukey(0.25) de ancho = 256, lo que determina un arreglo de forma 129×393 (frecuencia/tiempo) que contiene la densidad espectral de la señal. En la figura 5.2 se pueden ver los valores de la segunda, tercera y cuarta filas (índices 1 a 3) del espectrograma de la señal de la figura 5.1.

Para el entrenamiento los datos de los espectrogramas y los de *shimmer* se estandarizaron entre 0 y 1. En las señales generadas se observó que solo las 15 primeras filas del

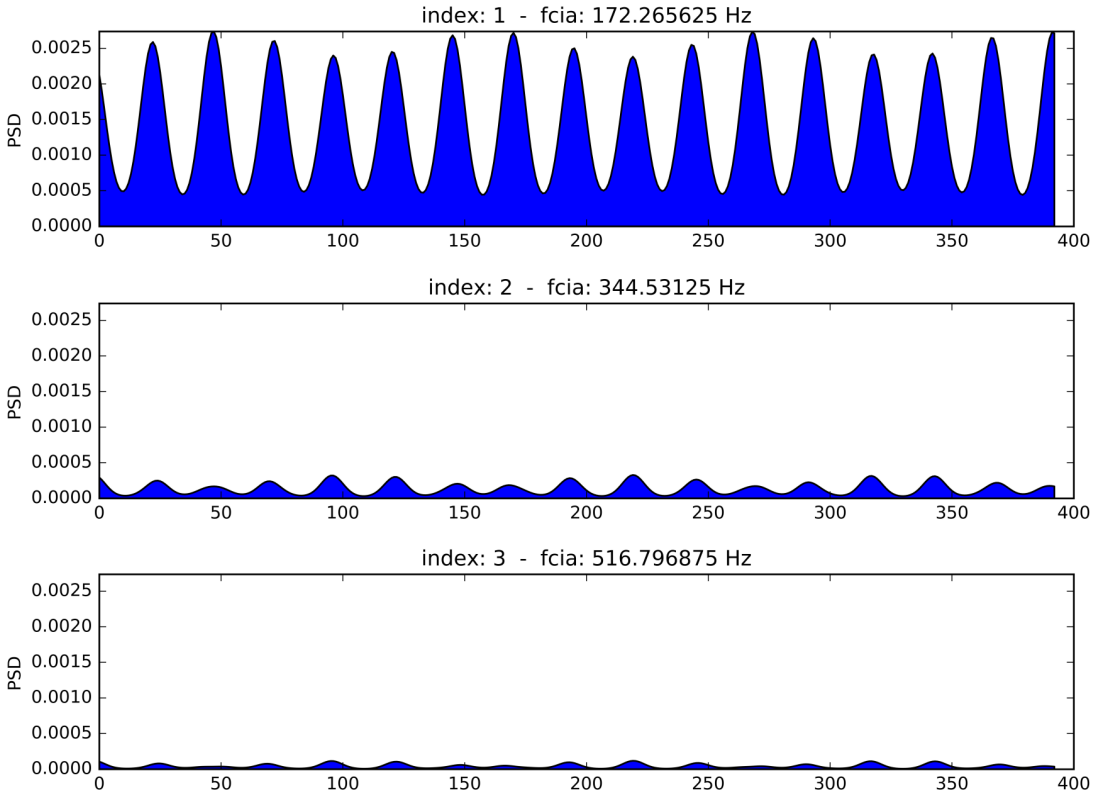


FIGURA 5.2: Tres filas con mayor valor medio de PSD (*Power Spectral Density*) en espectrograma de audio generado con $F_0 = 200$ Hz, $f_{mod} = 8$ Hz y $k = 0{,}4$.

espectrograma (frecuencias más bajas) tenían información significativa. Entonces, solo para el alcance propuesto en este experimento, se eliminó el resto de las frecuencias y el espectrograma cambió del tamaño 129×393 a 15×393.

Modelos Para comenzar a diseñar la red neuronal, inicialmente se analizó la relación que tiene *shimmer* con F_0, f_{mod} y k. Esta es una relación no lineal, pero no demasiado compleja. Se realizaron pruebas sobre redes neuronales formadas por capas densamente conectadas para aproximar *shimmer* utilizando como entrada los parámetros F_0, f_{mod} y k (notar que son valores escalares, no audio). Como resultado de una búsqueda en rejilla (*grid search*) en el espacio de los hiperparámetros, se determinó que una red con dos capas de 20 neuronas cada una y función de activación tanh() más una neurona lineal de salida, era capaz de aproximar *shimmer* con alta precisión. Sobre esta base se generaron modelos de aprendizaje profundo para problemas de aproximación de *shimmer* con complejidad ascendente. Los modelos se formaron con capas de convolución y capas densamente conectadas. En primer lugar se aproximó *shimmer* para F_0 variable, k constante y f_{mod} constante. En segundo lugar se aproximó *shimmer* para F_0 variable, k variable y f_{mod} constante. Por último, se encontró un modelo para aproximar *shimmer* con F_0, k y f_{mod} variables. Más detalles sobre el proceso se pueden ver en [156].

5.1.1.3. Resultados

En esta sección se presenta el modelo más simple que se obtuvo para aproximar *shimmer* con F_0, k y f_{mod} variables. Para F_0 en el intervalo [200, 1000] Hz, k en [0, 0.4] y f_{mod} en [5, 10] Hz fue necesario crear un modelo con una capa de convolución y tres capas densas (figura 5.3).

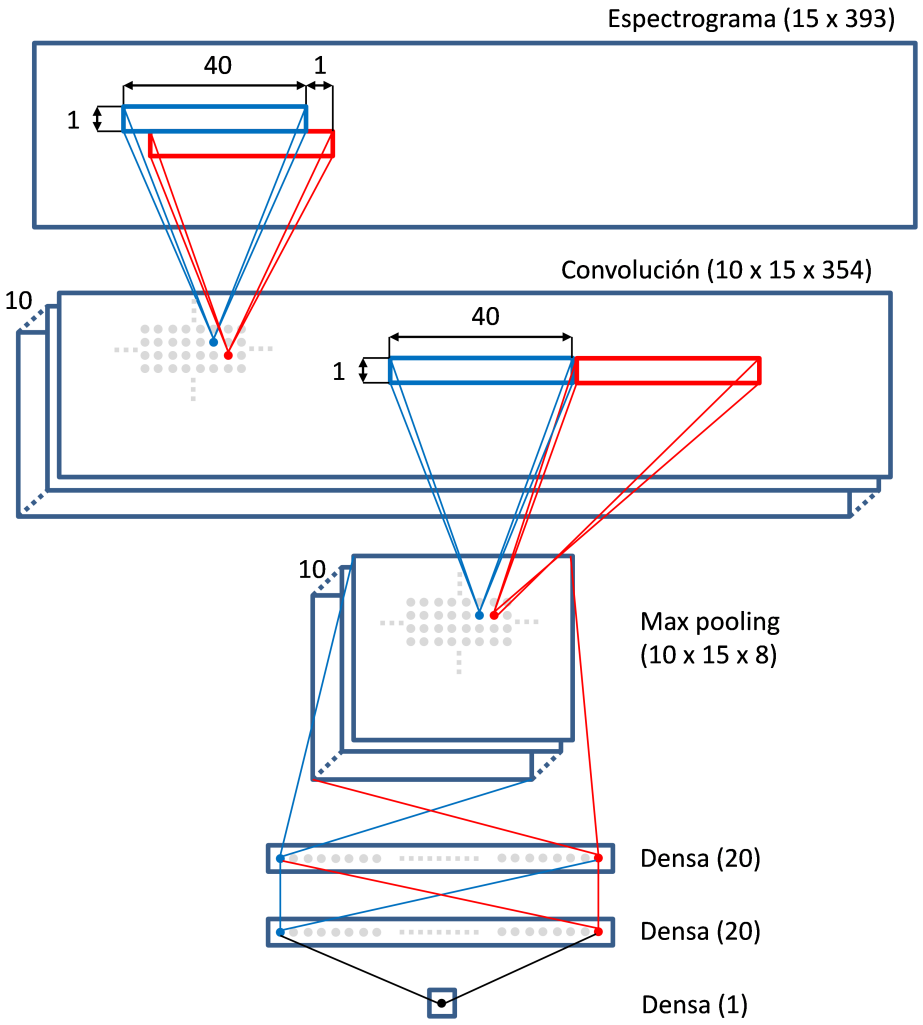


FIGURA 5.3: Modelo para cálculo de *shimmer* en señales con F_0 en el intervalo [200, 1000] Hz, k en [0, 0,4] y f_{mod} en [5, 10] Hz.

Capa de convolución. Tal como se mencionó en la sección 5.1.1.2, capas densamente conectadas que reciben la suficiente información sobre F_0, k y f_{mod} pueden aproximar el valor de *shimmer*. El espectrograma, para los datos generados, claramente contiene información sobre F_0. El valor de k afecta la cantidad total de energía del espectrograma, por lo tanto la información está disponible. La información de f_{mod} también está presente en el espectrograma, se puede medir el periodo de la señal de modulación, pero (por lo

menos para señales con fase variable en F_0 y f_{mod}) no es posible hacerlo con capas densas. Para calcular f_{mod} es necesaria una nueva transformación sobre el audio (la primera fue la STFT para el cálculo del espectrograma). La segunda transformación se realiza en una capa de convolución en la entrada del modelo.

Cada neurona de la capa de convolución se conecta al espectrograma con una ventana de tamaño 1×40 (alto $\times$ ancho). La convolución se realiza sobre una sola frecuencia (altura $= 1$) para que no se pierda el nivel de detalle de F_0 necesario en las capas densas. El ancho de 40 elementos es el mínimo necesario para contener un ciclo de mín(f_{mod}). La cantidad de elementos del espectrograma por ciclo de modulación (C) para un espectrograma de ancho W_s y un audio de duración L segundos es:

$$C = \frac{W_s}{L \times \text{mín}(f_{mod})} = \frac{393 \text{ elementos}}{2 \text{ s} \times 5 \text{ Hz}} = 39{,}3 \text{ elementos/ciclo}$$

El desplazamiento entre ventanas es 1 en ambas direcciones, lo que implica que en la dimensión de la frecuencia no hay solapamiento y en la dimensión del tiempo hay 39 elementos solapados entre las ventanas de neuronas contiguas. Finalmente, de acuerdo con estas definiciones, el tamaño de salida de cada filtro es 15×354. La capa de convolución tiene 10 *kernels*. Esta cantidad es una decisión de compromiso entre el rendimiento y el detalle con que la información de f_{mod} llega a las capas densas. Las neuronas de esta capa tienen función de activación lineal y los pesos se inicializan con valores aleatorios ortogonales. Se intentó incializarlos con familias de *wavelets* para ondas senoidales entre 5 Hz y 10 Hz, pero no se logró mejorar la exactitud de la predicción.

Capa de *max pooling*. Las neuronas de la capa de *max pooling* tienen una ventana de tamaño 1×40 sobre la capa de convolución. Nuevamente, la altura $= 1$ permite que la información de F_0 se transmita a las capas densas sin perder detalles. El ancho $= 40$ amplía el campo receptivo de las neuronas de esta capa a 2 ciclos de mín(f_{mod}) sobre el espectrograma. De esta forma, el valor salida es invariante a las traslaciones de la señal de modulación. No hay solapamiento entre las ventanas, por lo tanto el tamaño de cada una de los 10 filtros es de 15×8 neuronas.

Capas densas. Las salidas de la capa de *max pooling* se conectan a tres capas con conexiones densas iguales a las de la sección 5.1.1.2.

Rendimiento. Para el modelo presentado se realizaron 20 pruebas de entrenamiento con 2500 datos y se obtuvo un MSE $= 5{,}8 \times 10^{-5}$ sobre los datos de test. En la Fig. 5.4

se muestran los valores esperados y calculados de *shimmer* en orden ascendente para los 500 datos de test en una de las pruebas.

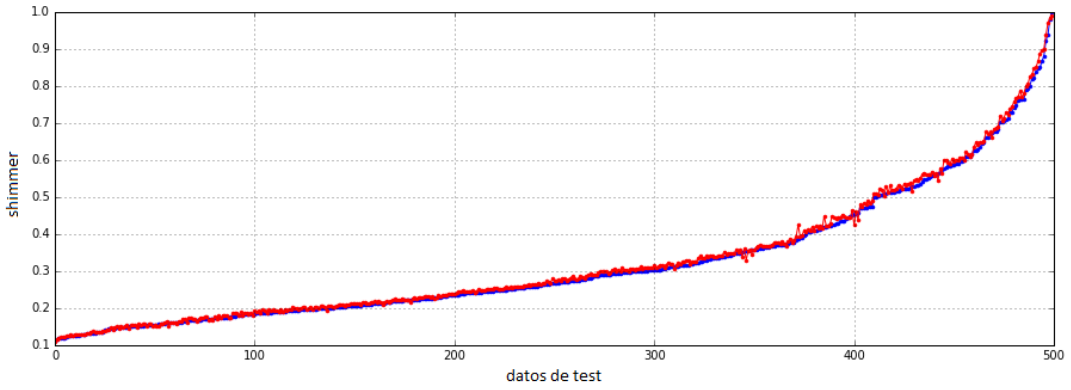


FIGURA 5.4: *Shimmer* normalizado esperado (azul) vs. calculado (rojo) para los 500 datos ordenados de test.

5.1.1.4. Conclusión

Se concluye que, bajo las condiciones planteadas en este experimento, es posible calcular *shimmer* sin conocer el valor exacto de F_0 y se puede afirmar que un modelo de aprendizaje profundo que respete la estructura del modelo presentado en sus primeras capas es capaz de utilizar el valor de *shimmer*, internamente calculado, para realizar clasificaciones de otro tipo, como por ejemplo, la calidad vocal.

5.1.2. Perturbaciones de F_0

La voz humana es una señal cuasi-periódica. Para que una voz produzca sensación de naturalidad deben existir pequeñas variaciones en la duración de los ciclos glóticos, pero una irregularidad excesiva en la forma de onda durante la fonación sostenida se asocia con patologías de la voz. Por ejemplo, algunos tipos de esclerosis producen ataxia (falta de control muscular) sobre la laringe originando perturbaciones en F_0. Estas perturbaciones suelen ser periódicas [25].

La medida de las perturbaciones de F_0 es conocida como *jitter*. Es la medida históricamente más aplicada en el análisis de la calidad vocal [25]. Por ejemplo, en [157], Jiang *et al.* encuentran diferencias significantes en los valores de *jitter* entre voces de pacientes con pólipos o nódulos en las cuerdas vocales y pacientes sanos. Oliveira *et al.* clasifican 21 patologías y encuentran que, entre las medidas utilizadas, *jitter* es la característica más relevante [158]. En [159] se caracterizan las voces patológicas y se observa que las medias de *jitter* y *shimmer* son mayores en las voces disfónicas. Fezari *et al.* detectan

voces patológicas con *jitter*, *shimmer* y MFCC en [160]. En el trabajo de Fonseca y Pereira [161] se clasifican muestras de personas sanas y personas con edema de Reinke para detectar la patología; se prueban varias medidas acústicas y la más relevante es *jitter*. En este último caso, las perturbaciones de F_0 no se calculan sobre la señal original, sino sobre la transformada *wavelet* discreta.

Jitter se calcula como la variación de la frecuencia fundamental entre sucesivos periodos de la señal.

$$jitter = \frac{\frac{1}{N-1}\sum_{i=1}^{N-1}|T_i - T_{i+1}|}{\frac{1}{N}\sum_{i=1}^{N}|T_i|} \tag{5.3}$$

donde T_i es el periodo de la frecuencia fundamental en el ciclo i y N es la cantidad de ciclos.

Existen muchas variantes para el cálculo de *jitter*. Todas las variantes enfrentan el mismo problema, intentan medir la desviación de la periodicidad, pero alcanzan su límite cuando la periodicidad se vuelve débil o el análisis se ve afectado por otras fuentes de fluctuación y/o modulación. La dificultad principal es el cálculo de la frecuencia fundamental [25]. Por esta razón no se intenta que el modelo neuronal de clasificación de la calidad vocal calcule internamente el valor de *jitter*, sino que obtenga cierta información relativa a la perturbación de F_0 que sea de utilidad.

En la representación frecuencial del audio es fácil notar estas perturbaciones. Para mostrarlo, a continuación se genera una señal modulada en frecuencia y se analiza su espectrograma.

$$y(t) = \sin\left(k\sin\left(\beta + \frac{2\pi t F_{mod}}{s}\right) + \alpha + \frac{2\pi t F_0}{s}\right) \tag{5.4}$$

donde $y(t)$ es la señal generada, t es tiempo [s], F_0 es la frecuencia de vibración glótica [Hz], F_{mod} es la frecuencia de modulación [Hz], k es la constante de sensibilidad del modulador, s es la frecuencia de muestreo, α y β son constantes para variar la fase de la señal a modular y de la señal moduladora respectivamente.

En la figura 5.5 se grafica $y(t)$ en función del tiempo y en la figura 5.6 se muestra su espectrograma. En el espectrograma, la perturbación se presenta como cambios de fila periódicos (debido a que la señal moduladora utilizada es periódica) en la posición de F_0.

Con las voces naturales la situación es similar. En la figura 5.7 se muestran los espectrogramas (izquierda de a, b y c) y el cepstrum en el tiempo (derecha de a, b y c) de tres

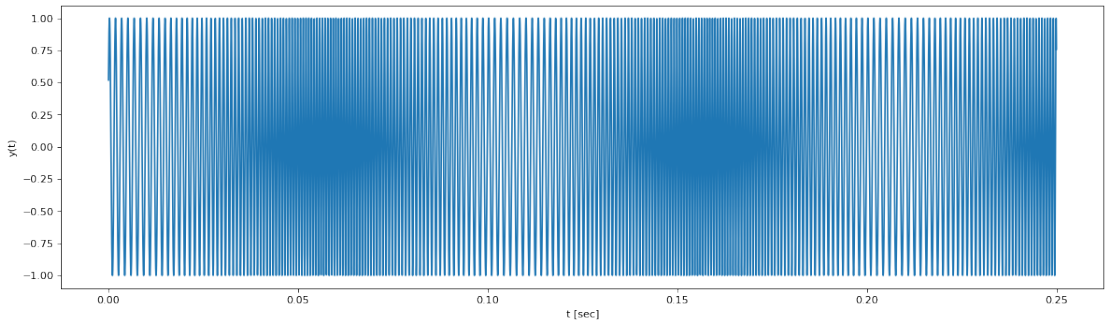


FIGURA 5.5: Señal de audio generada con la expresión 5.4.

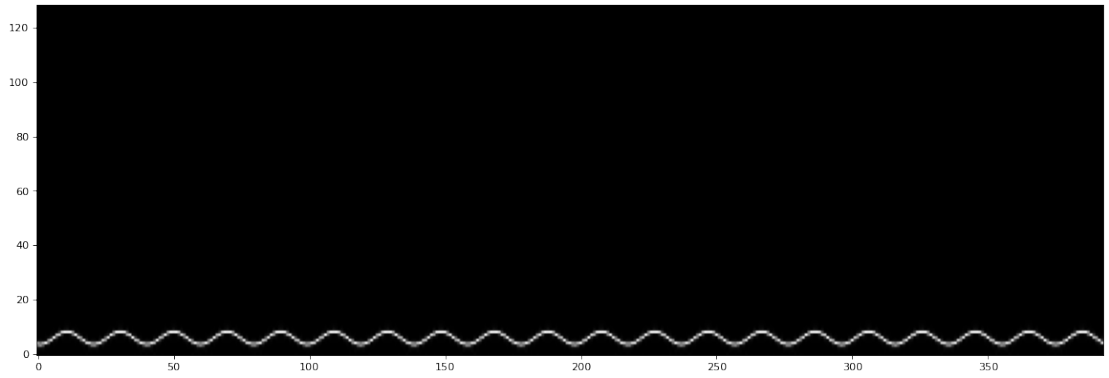


FIGURA 5.6: Espectrograma del audio generado con la expresión 5.4.

audios de la PVQD. En el primer caso (a) se puede ver un segundo del audio LA9017, donde se observa que al inicio no hay variación de frecuencia pero después aparece un tramo con *jitter* alto. El segundo caso (b), para el audio LA9015, presenta valores altos de *jitter*. El tercer audio (c), PT003, no tiene variaciones significativas de frecuencia. Es importante notar que en todos los casos, tanto los valores altos, como los valores bajos de *jitter* tienen patrones consistentes para el cepstrograma y el cepstrum en el tiempo. Esto indica que la información que brinda *jitter* se puede obtener con el mismo método tanto del espectrograma como del cepstrograma.

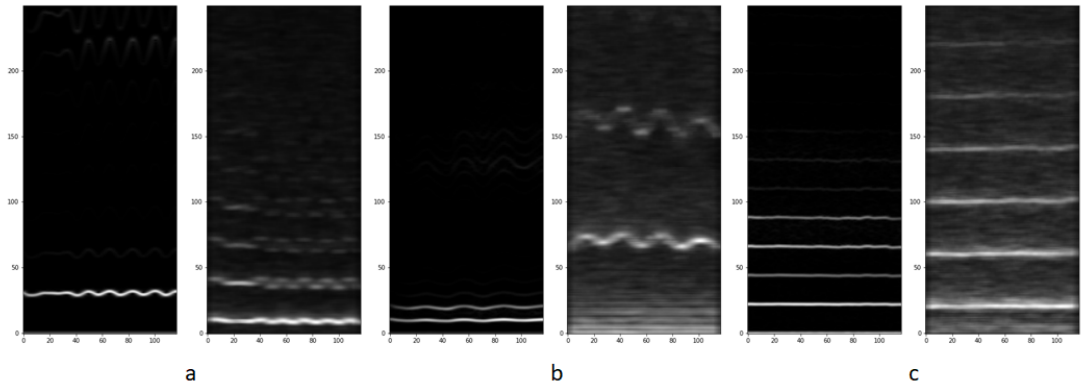


FIGURA 5.7: Comparación del espectrograma (izquierda) y el cepstrograma (derecha) de tres audios (a, b y c) de la PVQD.

Las perturbaciones de F_0 son evidentes en las imágenes del espectrograma y del cepstrograma. En ambas representaciones la perturbación se presentan como cambios de fila, algunas veces periódicos, en la posición de F_0 y sus armónicos. Dado este efecto, se considera que utilizando capas de convolución, una red neuronal no debería tener dificultad en reconocer las perturbaciones de F_0 si estas contribuyen a minimizar el error de clasificación. Este caso es más simple que el de *shimmer*, donde las perturbaciones de amplitud se hacen presentes en las magnitudes de las representaciones frecuenciales, pero sin provocar cambios en la posición de los máximos de energía. Por esta razón no se realiza para *jitter* un experimento similar al de la sección anterior. Tal como se mencionó anteriormente, no se espera que el modelo final calcule internamente el valor de *jitter*, pero se asume que las capas de convolución del modelo completo tienen la capacidad para extraer la información necesaria sobre las variaciones de F_0.

5.1.3. Ruido

El ruido puede ser una manifestación del esfuerzo vocal, y al mismo tiempo, el esfuerzo vocal es un síntoma clínico frecuente en personas con patologías vocales [93].

Las voces patológicas se caracterizan por la presencia de ruido. La medida más frecuente para cuantificar el ruido en las señales de voz es probablemente *armonics-to-noise ratio* (HNR) y se calcula como la relación entre la energía de los armónicos de F_0 y la energía del resto de la señal. En otras palabras, HNR es la relación entre energía periódica y el ruido producido por vibraciones no periódicas de las cuerdas vocales [93]. Las voces disfónicas tienen valores más bajos de esta medida que las voces normales. HNR se puede calcular tanto en el dominio del tiempo como en el de la frecuencia. En [162] se mencionan algunos de los métodos para ambos dominios. La ventaja de calcularlo en el dominio de la frecuencia es que se evita el cálculo de F_0. También se puede calcular HNR a partir del cepstrum [163, 164].

Otra medida relacionada con el ruido es la prominencia del pico del cepstrum (CPP por su sigla en inglés). Este se calcula como la diferencia de amplitud entre el pico cepstral (altura del mayor *rahmonic*) y el valor correspondiente en la línea de regresión directamente debajo del pico. Una señal muy periódica tiene una estructura armónica bien definida y un pico cepstral más prominente que una señal menos periódica.

En este trabajo, el diseño de las capas de extracción de características permite que la red neuronal cuente con la información necesaria para calcular un valor cercano a la *smoothed cepstral peak prominence* (CPPS), una variante del CPP. Las capas de extracción de características, o por lo menos las relacionadas con la CPPS, deben recibir entonces el cepstrum de la señal como entrada.

Se elige CPPS en lugar de HNR por las siguientes razones:

- El cálculo (o aproximación) de ninguna de las dos medidas ha sido implementado hasta ahora con redes neuronales, pero dadas sus características, la implementación de CPP es más simple.
- En el año 2018 un panel de expertos de la *American Speech-Language-Hearing Association* propuso protocolos para la evaluación instrumental de la función vocal. En el caso de la evaluación con medidas acústicas, se sugiere el uso de CPP para medir el nivel general de ruido en la señal. Según el mismo documento, después de mucha discusión, se eligió al CPP como una medida general de disfonía que refleja la relación global de energía periódica versus aperiódica en una señal [165]. Además, en la bibliografía se menciona numerosas veces a CPP como la medida que mayor correlación tiene con la disfonía. En [166–168] por ejemplo, se obtienen resultados satisfactorios de clasificación utilizando solo CPP o CPPS como características.
- El cálculo de HNR depende de la periodicidad de la señal en mayor medida que el cálculo de CPP, por lo tanto en voces con grado 2 y 3 de disfonía, CPP es más confiable [92].

La elección de CPP no implica que la información provista por HNR no sea utilizada. Las dos medidas están relacionadas porque ambas comparan la estructura de los armónicos y el resto de la señal. En [169] Murphy analiza gmHNR, una variante de HNR obtenida sobre los dB del espectro, y esta resulta directamente proporcional al primer *rharmonic* (armónico en el cepstrum). Samlam *et al.* reportan en [170] que HNR y CPP están fuertemente relacionados entre sí y relacionados linealmente con la dimensión B de GRBAS según los resultados de un experimento sobre 364 muestras de audio. Fraile y Godino-Llorente en [171] hacen un estudio analítico del CPP y concluyen que la afirmación de Murphy se puede extrapolar al CPPS.

5.1.3.1. CPP y CPPS

Las primeras *quefrencies* del cepstrum corresponden a la señal moduladora que define los formantes. Esta señal se forma en el tracto vocal y no es de interés en el análisis de la calidad vocal. Para el cálculo de la CPP se eliminan las primeras *quefrencies* y se mantiene el resto, que es donde se encuentra la información sobre la señal producida en las cuerdas vocales.

En la figura 5.8 se muestran las 200 primeras *quefrencies* del cepstrum calculado sobre un segmento de 70.4 ms del audio LA9018 de la PVQD. La línea de puntos vertical

indica el lugar donde se corta el cepstrum para separar la señal proveniente del tracto vocal de la de las cuerdas vocales. En este caso, se eliminan las 20 primeras *quefrencies*. Los cuatro picos que se ven a la derecha de la línea vertical son los *rharmonics*.

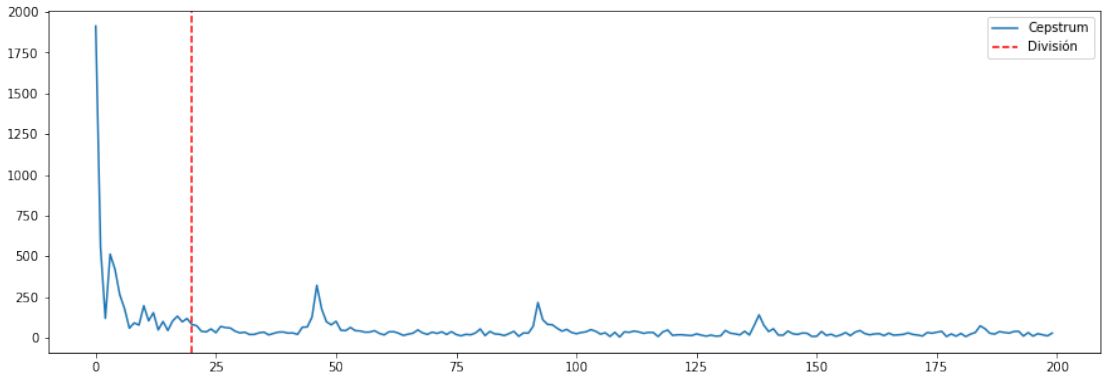


FIGURA 5.8: 200 primeras *quefrecies* del cepstrum del audio LA9018 de la PVQD.

La señal resultante de eliminar los 20 primeros elementos se puede ver (primeras 180 *quefrencies*) en la figura 5.9. Sobre esta señal se calcula la CPP. La prominencia del pico es la distancia entre los dos puntos marcados con "×" en la figura. El punto superior es la amplitud del *rharmonic* más alto (máximo absoluto), mientras que el punto inferior es el valor, justo bajo el máximo, de la línea obtenida por regresión lineal de toda la señal.

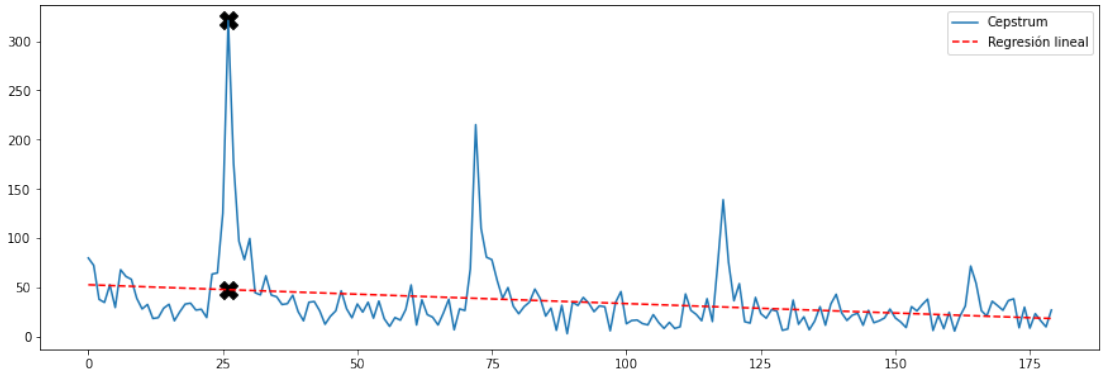


FIGURA 5.9: Prominencia del pico del cepstrum (CPP) del audio LA9018 de la PVQD.

En este trabajo se utiliza la CPPS, una versión de la CPP frecuentemente elegida. CPPS fue propuesto en el trabajo de Hillenbrand y Houde en 1996 [172], donde mediante experimentación encontraron que suavizando el cepstrum antes de calcular la prominencia del pico se mejora la exactitud de las predicciones. En la figura 5.10 se compara el cepstrum original con el suavizado y se marcan los puntos para el cálculo de la CPPS. En este ejemplo el suavizado se realizó convolucionando el cepstrum con una campana gaussiana de tamaño = 5 y desviación estándar = 2.

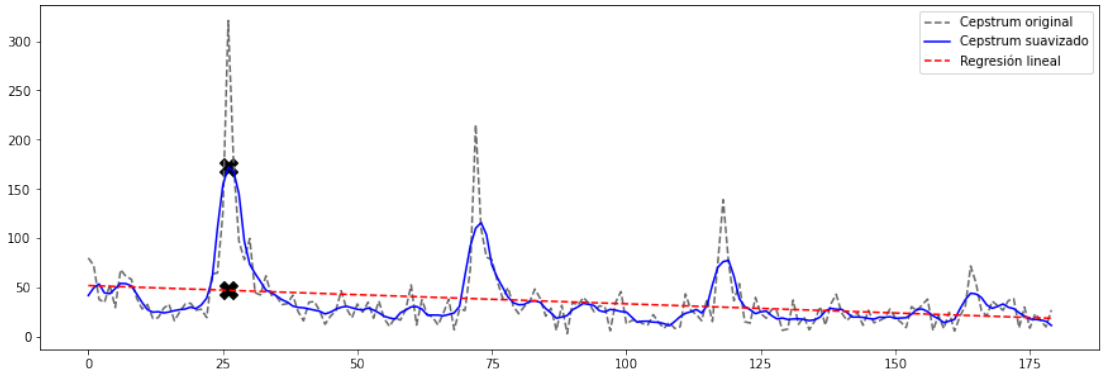


FIGURA 5.10: Prominencia del pico del cepstrum suavizado (CPPS) del audio LA9018 de la PVQD.

5.1.3.2. Red neuronal

El cálculo de la CPPS con redes neuronales se complica debido a la regresión lineal del cepstrum. Para obtener una red simple se cambia el valor del referencia para el cálculo de la prominencia por la media de los valores del cepstrum. En la figura 5.11 se puede ver, utilizando los mismos datos que en las figuras anteriores, una comparación entre las dos referencias (media y regresión lineal). También se muestra la prominencia del pico tomada desde el valor de la media.

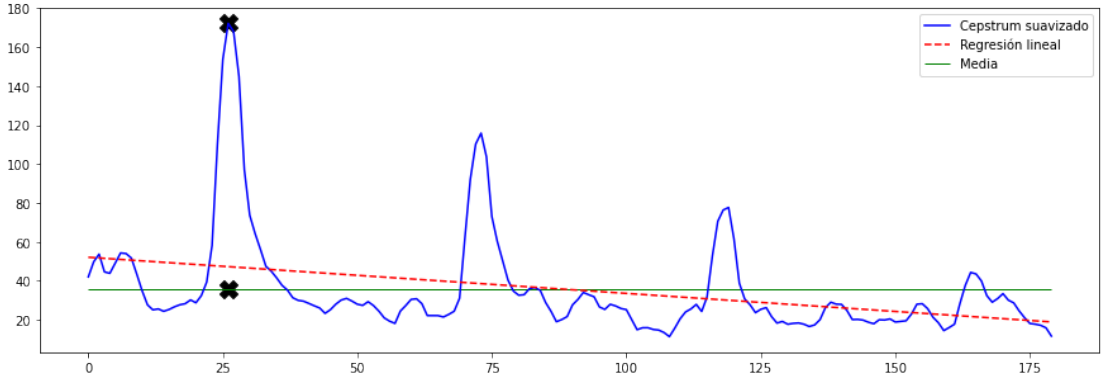


FIGURA 5.11: Comparación entre el uso de la media y de la regresión lineal para calcular la prominencia del pico del cepstrum suavizado.

La red neuronal que se presenta en esta sección calcula internamente los valores máximo y medio del cepstrum suavizado. Claramente, con estos valores no se puede calcular la CPPS tradicional, pero sí se puede obtener una idea de la amplitud del pico. La amplitud del pico cepstral refleja el nivel de organización armónica. Se espera que las señales de voces normales tiendan a mostrar mayor grado de organización armónica que las de voces patológicas que contienen escapes de aire (B en GRBAS) y/o perturbaciones de frecuencia [172].

La figura 5.12 muestra la arquitectura de la red neuronal utilizada como parte del modelo inicial. La entrada es el cepstrograma, o cepstrum en el tiempo. El suavizado se realiza con una capa de convolución 2D que se conecta a la entrada con un *kernel* de tamaño $n \times 1$. Los pesos del *kernel* se inicializan con los valores de una campana gaussiana y pueden mantenerse fijos o permitir que se ajusten. Sobre la salida completa de la convolución se calculan el valor máximo y el valor medio con capas de *max pooling* y *average pooling* respectivamente. Finalmente, las salidas de las capas de *pooling* se conectan a una única neurona con activación lineal. El valor de n se define en el proceso de optimización de la arquitectura del modelo final.

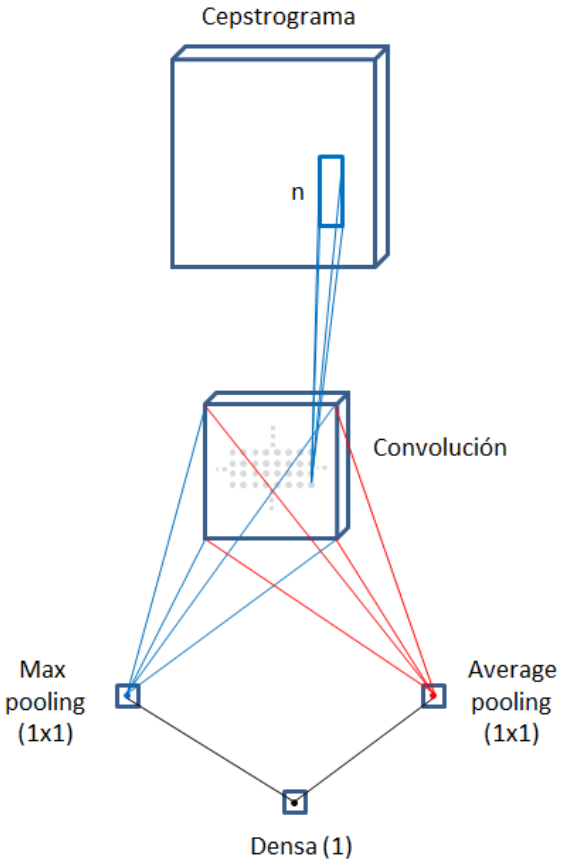


FIGURA 5.12: Arquitectura de la red neuronal de extracción de características de ruido

5.2. Clasificación

El grado G de disfonía es una variable categórica ordinal. En consecuencia, los errores de clasificación tienen mayor importancia en la medida que aumenta la distancia entre la clase predicha y la clase esperada. Además, tal como lo expresan Gómez *et al.* en [36], dado que G se clasifica con cuatro puntos o etiquetas, la escala es insensible a otros estados sutiles pero potencialmente informativos entre las etiquetas. Por lo tanto,

el sistema debe considerar la existencia de más de cuatro estados que caracterizan a las voces bajo análisis.

Desde el punto de vista de la evaluación, esta situación se tiene en cuenta a través de la métrica del error absoluto medio. Para que el clasificador tienda a minimizar los errores entre las clases más distantes, el objetivo del modelo se planteó en primer momento como una regresión, es decir utilizando una neurona lineal de salida y codificando la salida con funciones umbral para determinar la clase. Se comparó el rendimiento con el de un clasificador que utilizaba cuatro neuronas (una por clase) con función de activación sigmoidal. La última configuración obtuvo mejores resultados, por lo tanto el modelo inicial tiene cuatro neuronas de salida.

5.2.1. Red Neuronal

La clasificación se realiza con capas densamente conectadas. Tal como se adelantó, la capa de salida tiene cuatro neuronas con función de activación sigmoidal. El resto de las capas utiliza función de activación ReLU. La cantidad de capas, la cantidad de neuronas por capas y el uso de técnicas de regularización o *dropout* se definió durante el proceso de optimización del modelo.

Capítulo 6

Modelo inicial

En este capítulo se presenta una red neuronal, el modelo inicial, creada a partir de los desarrollos de los capítulos 4 y 5. El modelo inicial es el punto de partida del proceso de optimización que se lleva a cabo más adelante para obtener el clasificador del grado general de disfonía.

6.1. Datos de entrada

La entrada del modelo es un audio de un segundo de duración muestreado con una tasa de 25000 muestras por segundo. En consecuencia, cada vector de entrada tiene tamaño 1×25000.

6.2. Representación frecuencial

La red neuronal calcula internamente el cuadrado de la autocorrelación, variante del *power cepstrum* sin logaritmo, tal como se hace en la sección 4.2.1. Se eligió esta representación frecuencial porque, a diferencia del cepstrum y tal como se concluyó en la sección 4.2.4, permite entrenar las capas anteriores del modelo. El logaritmo en el cálculo del cepstrum ayuda a identificar y separar fácilmente los efectos del filtro y la fuente en la producción de la voz [88], por lo tanto, eliminarlo supone una desventaja. Para determinar cuál de las dos alternativas es conveniente, se probarán dos modelos iniciales:

- El modelo completo presentado en este capítulo, que recibe *raw audio* como entrada y calcula internamente el cuadrado de la autocorrelación (MIA en la sección 6.4).

- Un modelo reducido que recibe el *power cepstrum* como entrada y lo inserta directamente en las capas de extracción de características (MIB en la sección 6.4).

A partir de las capas de extracción de características (incluidas), los dos modelos iniciales son exactamente iguales. Con estas variantes se busca determinar si es más importante, para este caso particular, la ventaja obtenida al utilizar logaritmo en el cálculo del cepstrum o la ventaja de ajustar los pesos de las capas previas. Por conveniencia, en ambos casos se utilizará el término *quefrency* para indicar la posición de un elemento determinado en el cepstrum o en el cuadrado de la autocorrelación, según corresponda.

Antes del cálculo del cuadrado de la autocorrelación, se multiplica el audio de entrada, en tramos de longitud 1760, por una ventana adaptable utilizando una capa STHadamard. Los segmentos de audio de longitud 1760 resultan en una duración de 70.4 ms, con lo que se asegura que cada segmento contenga por lo menos 7 ciclos de F_0. La salida de la capa STHadamard tiene tamaño 1760×117, mientras que el cuadrado de la autocorrelación resulta con tamaño 420×117 porque se eliminan las 20 *quefrencies* más bajas.

En la figura 6.1 se muestra el modelo completo, donde cada capa u operación tiene una etiqueta formada por una letra y un número. La capa STHadamard, por ejemplo, tiene la etiqueta A1. El cálculo del cuadrado de la autocorrelación se realiza entre las operaciones A2 y A8.

6.3. Extracción de características y clasificación

La extracción de características se hace por dos caminos distintos de la red neuronal. Uno de los caminos (camino 1) se diseñó para extraer información similar a la de las perturbaciones de amplitud y de F_0 según lo planteado en las secciones 5.1.1 y 5.1.2 respectivamente. El camino 2 fue pensado para obtener información similar a la del CPPS y sigue la arquitectura del modelo propuesto en la sección 5.1.3.2. A continuación se describen los dos caminos.

Camino 1. Comienza con una capa de convolución 2D (B1) compuesta por 10 *kernels* de tamaño 1×20, desplazamiento 1×1 y *padding* = "same". La capa siguiente (B2) realiza la operación de *max pooling* con una ventana de tamaño 1×11, desplazamiento 1×11 y *padding* = "valid", lo que resulta en una salida de dimensión $420 \times 10 \times 10$. El camino termina con tres capas densamente conectadas, B3, B4 y B5, con 20, 20 y 1 neuronas respectivamente y funciones de activación ReLU para las dos primeras y lineal para la última.

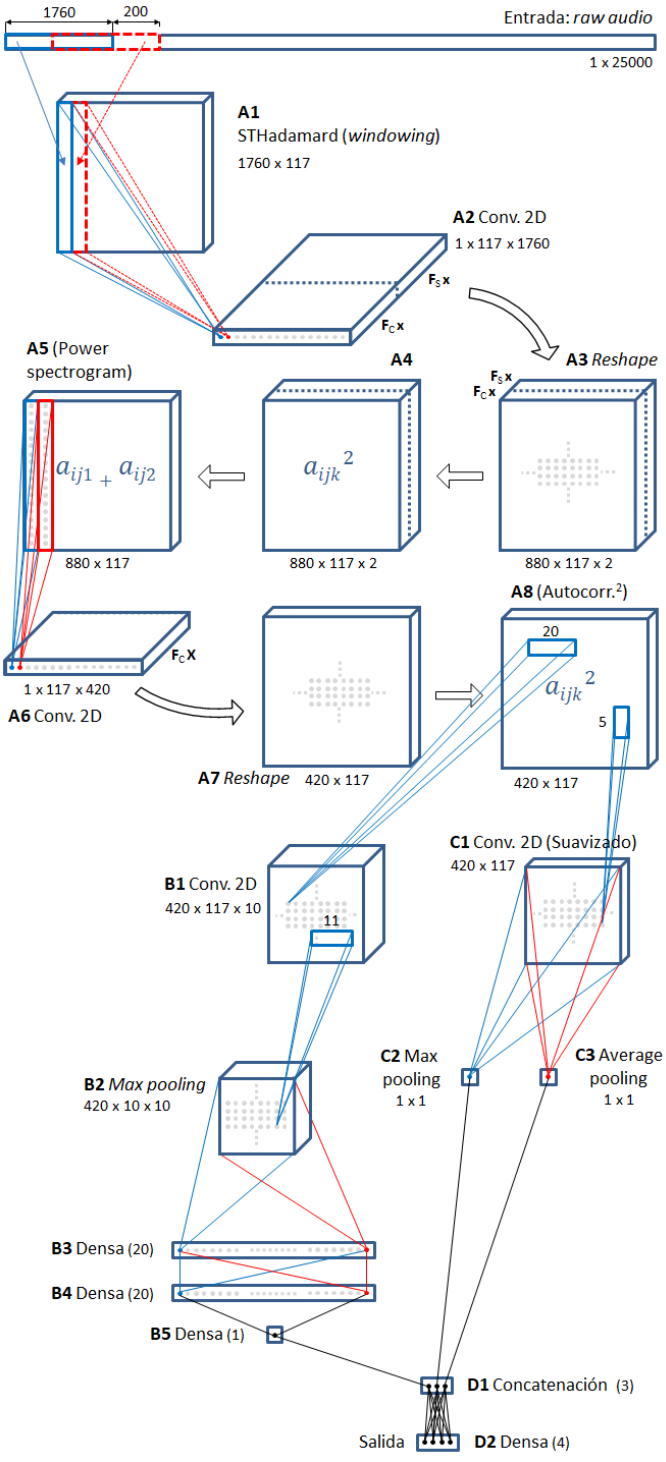


FIGURA 6.1: Modelo inicial MIA.

Camino 2. El segundo camino empieza con el suavizado del cepstrum o cuadrado de la autocorrelación, que se realiza en la capa C1 con una convolución 2D de un solo *kernel* inicializado con una campana gaussiana de tamaño 5×1, desplazamiento 1×1 y *padding* = "same". Sobre el suavizado completo, se calculan el máximo y la media con las capas C2 y C3, ambas con ventanas de tamaño 420×117.

La clasificación se realiza con cuatro neuronas sigmoidales (D2) sobre las salidas de los caminos 1 y 2 concatenadas en D1. Tanto en la clasificación como en la extracción de características, se eligieron opciones que mantuvieran la red con baja complejidad, para que esta se incrementara, en caso de ser necesario, durante el proceso de optimización. La estrategia inversa, de mayor a menor complejidad, requiere más recursos.

6.4. Variantes

En el capítulo siguiente se detallan los experimentos realizados. Tal como se explicó en la sección 6.2, los experimentos se realizan sobre dos variantes.

La primera variante, MIA (por modelo inicial A), corresponde al modelo completo tal como se lo presentó en este capítulo.

La segunda variante, MIB, es un red neuronal reducida que no incluye la etapa de representación frecuencial y recibe el cepstrum como entrada en lugar de *raw audio*. La arquitectura se obtiene tomando el modelo completo y eliminando la capa A8 y capas anteriores.

Capítulo 7

Experimentos

En este capítulo se presentan los detalles de los experimentos realizados sobre los modelos de clasificación iniciales MIA y MIB. El objetivo de estos experimentos es tanto la definición del modelo final como su evaluación.

7.1. Definición del modelo final

El modelo final es el resultado de un proceso de búsqueda en el espacio de los hiperparámetros, donde se realizan pruebas sobre cada modelo y se ensayan modificaciones, como por ejemplo, la cantidad de capas densas o de convolución, cantidad de neuronas, tamaño de los *kernels*, tamaño de las ventanas en capas de *pooling*, funciones de activación, valores de inicialización de parámetros y estrategias de regularización entre otros.

Debido a la cantidad de variables de ajuste y al alto costo computacional de cada entrenamiento, no se realizó una búsqueda en rejilla (*grid search*), sino un proceso de optimización guiado por los resultados sobre los datos de validación, donde las decisiones sobre las modificaciones de hiperparámetros se tomaron en base al análisis de la evolución de ambas métricas durante el entrenamiento, la evolución y configuración final de los parámetros (pesos sinápticos) y de las salidas intermedias (salidas de las capas ocultas).

El proceso de búsqueda se repitió para cada uno de los modelos iniciales.

7.2. Datos

Se utilizan los datos de la PVQD, presentada en la sección 3.3.1. A continuación se explican brevemente las razones de esta elección.

La base de datos de referencia para la clasificación de la calidad vocal ha sido hasta hace pocos años la *Massachusetts Eye and Ear Infirmary* (MEEI) *Voice Database*[173], una base de datos comercial que fue luego muy criticada en la literatura por tener condiciones de grabación distintas para individuos de control y patológicos, lo que limita la validez de las conclusiones que se saquen sobre sus datos. En la actualidad no es posible conseguir la base de datos de MEEI porque, a raíz de las críticas, dejó de venderse.

Se analizó el uso de varias bases de datos, pero por problemas de acceso, cantidad reducida de datos y falta de valoraciones de la calidad, la elección se redujo solo a dos, PVQD y HUPA.

Existen dos razones importantes en la elección de la PVQD. Por un lado, las clases (niveles de G) están mejor distribuidas que en la base de datos HUPA. Si se desea que los datos estén balanceados, la cantidad de ocurrencias de la clase menos frecuente determina la cantidad de datos que se pueden utilizar. La clase menos frecuente es G = "3" para ambas bases de datos (al igual que para la población), pero en la PVQD la cantidad es mayor. Por otro lado, la PVQD es la única base de datos que tiene el detalle de las instancias de valoración, por lo tanto es la única que se adapta a las características de la evaluación planteada en este trabajo, para la cual se requieren las variabilidades intraevaluador e interevaluador.

7.2.1. Revisión de PVQD y nueva valoración GRBAS

Tal como se comentó anteriormente, la PVQD contiene dos valoraciones en escala GRBAS por cada uno de cuatro evaluadores. El cuarto evaluador valoró solo el 16 % de los casos, por lo tanto solo se pueden usar las valoraciones de los tres primeros. El autor de PVQD asegura que los evaluadores tienen experiencia suficiente en el área de la voz, pero no ofrece información sobre cuál de ellos es el más experimentado, ni cuánta experiencia tiene. Este dato sería útil, tanto para confiar en las valoraciones de PVQD, como para decidir qué conjunto de valoraciones utilizar durante el entrenamiento y evaluación de la red neuronal.

Debido a que los audios de la PVQD deben ser controlados (ver sección 3.3.1) y que es deseable tener una referencia concreta sobre el evaluador, la base de datos completa se

analizó y valoró por un profesional local con más de 20 años de experiencia en tratamientos clínicos de la voz. Para ello se firmó un acuerdo de cooperación con la Escuela de Fonoaudiología de la Facultad de Ciencias Médicas de la Universidad Nacional de Córdoba. A continuación se detallan los pasos realizados para el control y valoración de los audios de PVQD.

1. Segmentación de vocal /a/ sostenida. Los audios de la base de datos contienen las vocales /a/ e /i/ sostenidas, algunas frases de audio continuo y en algunos casos también glissandos. Estos elementos no están siempre en el mismo orden, no tienen la misma duración, ni existen todos en todos los archivos. La vocal /a/ sí se encuentra siempre. En este paso del procedimiento se editaron manualmente todos los archivo de audio con el software Audacity[1] dejando solo la vocal /a/ sostenida. Se mantuvieron el nombre y la codificación originales de los archivos. Los archivos con audios de duración menor a un segundo se eliminaron.

2. Renombrado de archivos y generación de planilla. El nombre de los archivos tiene cierta relación con la calidad vocal. Existen grupos de archivos que comienzan con las mismas letras y las grabaciones de cada grupo tienen niveles de calidad vocal muy distintos. Para evitar un sesgo en la nueva valoración, se generaron nombres aleatorios para los archivos y se entregó una planilla para completar por el evaluador. En la planilla, los audios se ordenaron alfabéticamente (por los nuevos nombres).

3. Análisis y valoración. En un ambiente sin ruido, con auriculares de alta calidad Sennheiser HD 202, una PC con placa de sonido externa Audiobox PreSonus, el software Audacity y cuidando los tiempos de descanso recomendados, el profesional local analizó y valoró en escala GRBAS los audios según el orden de aparición en la planilla. Siempre que el profesional consideraba que un audio no era apto para la valoración lo indicaba en la planilla y escribía un comentario. También se podía escribir comentarios sin recomendación.

4. Selección y corrección de archivos de audio. Se eliminaron los archivos marcados por el profesional local y se analizaron los comentarios. Para los archivos con comentarios en la planilla se tomó la decisión de eliminar, editar o mantener en estado original según corresponda.

5. Generación de base de datos transaccional. En una base de datos SQLite[2] se cargaron los resultados de las planillas haciendo referencia a los nombres originales de los archivos. También se cargaron las valoraciones GRBAS de PVQD.

[1] https://www.audacityteam.org
[2] https://www.sqlite.org

En el primer paso se eliminaron 6 archivos de audio porque no cumplían con la condición de duración mínima.

En la tabla 7.1 se muestran los resultados de la valoración del paso 3. La columna NC contiene la cantidad de archivos que no pudieron ser valorados en cada categoría GRBAS. Las razones son ruidos en la grabación, voces inestables (la valoración varía en el tiempo) y voces cantadas con vibrato (variación periódica intencional de la frecuencia fundamental). Los archivos no valorados fueron marcados para borrar.

Tabla 7.1: Valoración del evaluador local durante el tercer paso del proceso.

	Grado					
Categoría	0	1	2	3	NC	Total
G	61	126	65	30	8	290
R	129	79	62	11	9	290
B	59	139	65	18	9	290
A	212	54	14	-	10	290
S	238	24	13	5	10	290

En el paso 4 se eliminaron 17 archivos más, 2 por vibrato, 5 por ruido, 2 por errores de grabación y 8 por falta de uniformidad. Los archivos con valoraciones no nulas que fueron eliminados tenían grados de G iguales a “0”, “1” o “2”, por lo tanto no afectan a la cantidad de datos en el juego de datos balanceado. Además, 8 archivos fueron editados, se les cortó alguno de los extremos para eliminar ruidos o la voz del profesional médico/técnico grabada por error.

7.2.2. Aumentación de datos

Tal como se menciona en la sección 2.2, las redes neuronales profundas tienen muchos parámetros internos para ajustar y, por lo tanto, son propensas al sobreajuste. Una de las formas de lidiar con el sobreajuste es aumentar la cantidad de datos de entrenamiento. En este trabajo se realizaron tres transformaciones para aumentar la cantidad de datos, desplazamiento en frecuencia, segmentación por tiempo (*cropping*) y *flipping*. Después de las tres transformaciones la cantidad de datos se multiplicó por 18, salvo cuando no se pudo realizar el incremento en frecuencia (en ese caso, cada archivo se multiplicó por 12).

Antes de transformar los audios, estos fueron remuestrados a 25000 muestras por segundo, la misma frecuencia utilizada en HUPA. La frecuencia de muestreo original de

PVQD (44100 Hz) es excesiva para representar voces. Si se mantuviera la frecuencia original los vectores de entrada (un segundo de audio) serían significativamente más grandes, el modelo más complejo, más costoso de entrenar y, lo más importante, con mayor tendencia al sobreajuste. Los remuestreos se realizaron con la función *resample* de la librería Librosa[3].

7.2.2.1. Desplazamiento en frecuencia

Para cada archivo de audio se realizaron dos desplazamientos en frecuencia, incrementándola un 20 % en el primero y decrementándola en la misma proporción para el segundo. Cada desplazamiento generó un nuevo archivo de audio, multiplicando por 3 la cantidad de datos. El proceso implica un remuestreo (*resample*) a 20000 muestras por segundo en el primer caso y 31250 en el segundo caso. Después del remuestreado los datos se guardaron con frecuencia de muestreo de 25 KHz en los metadatos del archivo de audio. El resultado de estos cambios se muestra en la figura 7.1, donde se puede ver el efecto en los dominios del tiempo y la frecuencia de la transformación sobre un segmento de audio. Desde arriba hacia abajo, se puede ver que la señal en el dominio del tiempo se acorta, es decir, aumenta la frecuencia, mientras que en el dominio de la frecuencia el cambio se hace evidente en la ubicación de los picos y la separación entre los armónicos.

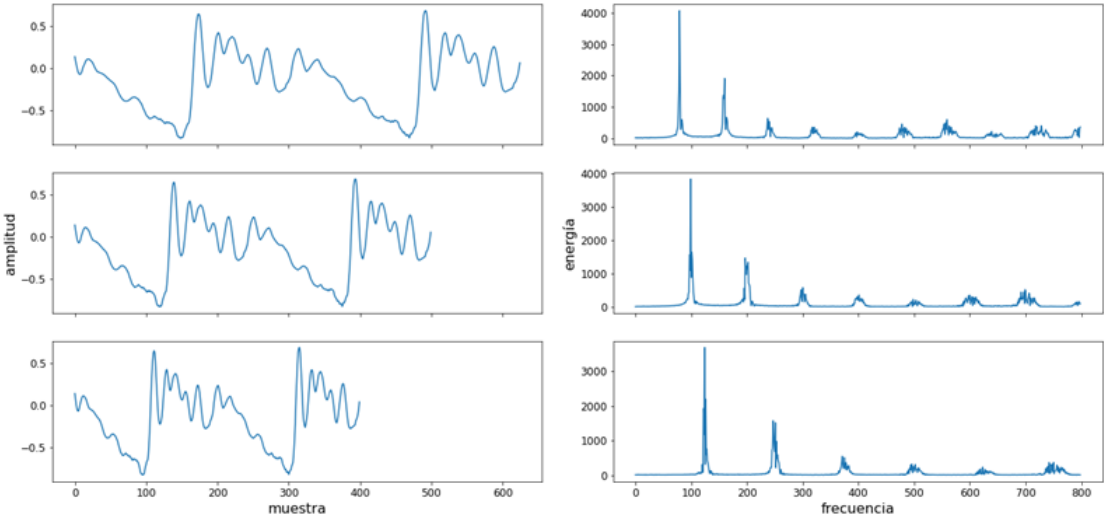


FIGURA 7.1: Desplazamiento en frecuencia. En la fila central se ubica la señal original, en la fila superior la señal con un decremento del 20 % en la frecuencia y en la fila inferior la misma señal con un 20 % de incremento. En la columna de la izquierda se muestra el efecto en el dominio del tiempo (25000 muestras por segundo) y a la derecha el espectro de frecuencias.

El remuestreado a 20.000 muestras por segundo puede resultar en un audio de longitud menor a un segundo. En esos casos se descartó la nueva señal.

[3] https://librosa.org

7.2.2.2. Segmentación por tiempo

De cada archivo de audio se extrajeron tres segmentos de 1 segundo de duración. La ubicación de los segmentos izquierdo (I), central (C) y derecho (D) depende de la longitud L del audio original. Para $L > 2$ segundos, se tomó el segmento central S de dos segundos, en otro caso, el segmento S fue el audio original completo. El segmento I es el primer segundo de S, C es el segundo central de S y D el último segundo de S. En la figura 7.2 se muestra gráficamente la segmentación.

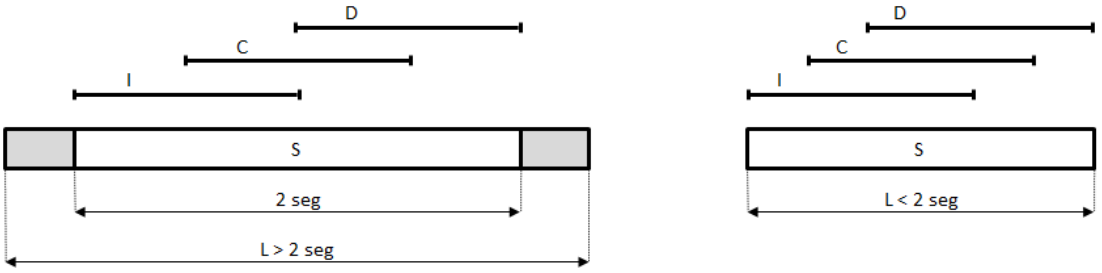


FIGURA 7.2: Extracción de tres segmentos de audio para un archivo original de longitud $L > 2$ segundos (izquierda) y para un archivo original de $L < 2$ segundos (derecha).

7.2.2.3. *Flipping*

Con esta transformación se generó un nuevo audio a partir de cada uno de los audios existentes invirtiendo el orden la señal en el tiempo. En la Figura 7.3 se muestra la transformación de un segmento pequeño de audio.

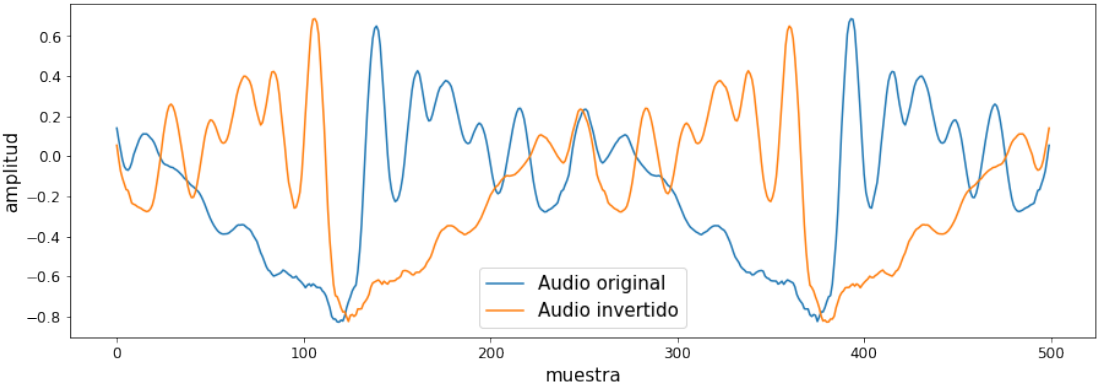


FIGURA 7.3: Segmentos de audio original y transformado en la etapa de *flipping*.

Esta técnica de aumentación de datos no es aplicable al reconocimiento del habla porque cambia el orden y la forma de los fonemas, pero sí resulta útil para la clasificación de la calidad vocal si se utiliza, como en este caso, audios de vocales sostenidas. El espectro de frecuencias de la señal original es el mismo que el de la señal invertida, pero su espectrograma resulta en un espejado horizontal, tal como se puede ver en la figura 7.4.

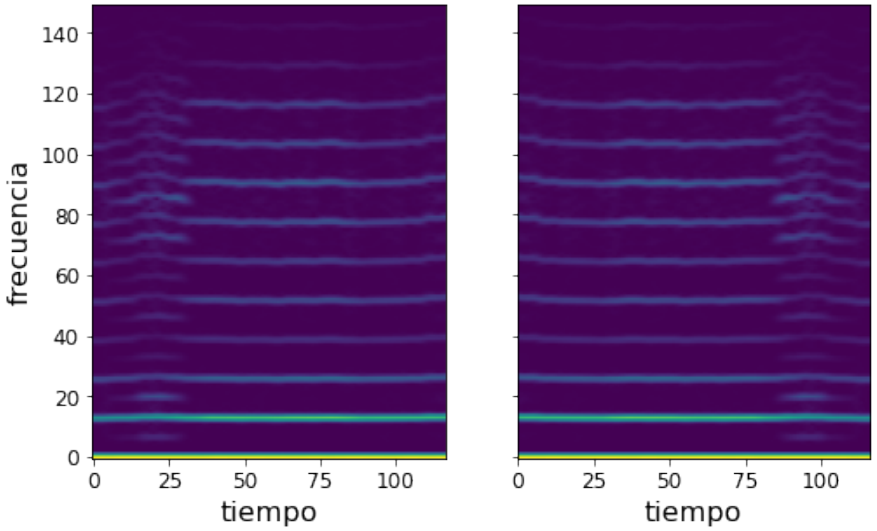


FIGURA 7.4: Espectrogramas de audio original (izquierda) y del audio transformado en la etapa de *flipping* (derecha).

7.2.3. Generación de los juegos de datos

Para la evaluación final del modelo (test) se reservó el 20 % de los datos. Con los datos restantes se utilizó un esquema de validación cruzada de k iteraciones (*k-fold*) para k=4. En total fue necesario crear 5 juegos de datos del mismo tamaño.

Para evitar problemas similares a los del trabajo de Ritchings *et al.* [33, 34] (ver sección 1.5), el proceso de creación de los subconjuntos de datos se llevó a cabo de manera tal que, en ningún caso, dos audios generados (durante la aumentación de datos) a partir del mismo archivo original, pudieran pertenecer a juegos de datos distintos. Los pasos son:

1. División por datos originales. Se generó una tabla con los nombres de los archivos originales y la valoración de G según el evaluador local. La tabla se dividió en k partes (para k=5) llamadas k-tablas. La división se hizo de forma estratificada, es decir, manteniendo en cada k-tabla la misma proporción de cada clase (grado de G) que tenía la tabla original.

2. Creación de k grupos de datos aumentados. Para cada k-tabla se creó un grupo que contiene todos los audios generados en la aumentación de datos con los archivos de origen de la k-tabla correspondiente.

3. Creación de los k subconjuntos. En este paso se generó un subconjunto balanceado de datos, es decir, que contiene la misma cantidad de audios de cada clase, por cada grupo de datos aumentados. Para cada grupo generado en el paso anterior se calculó la cantidad de ocurrencias de cada clase. La cantidad más baja de

cada grupo, mc, es el límite de datos para todos las clases. Para los datos de este trabajo, mc siempre está definido por el grado "3" de G. Finalmente, cada subconjunto se formó eligiendo aleatoriamente mc audios de cada clase en el grupo correspondiente.

Los resultados de este proceso fueron cinco subconjuntos de datos balanceados que contienen, como máximo, 432 audios cada uno. El tamaño se redujo en algunos casos, aunque no significativamente, dependiendo de la longitud de los archivos de audio originales para el nivel G="3" (en los casos donde no se pudo incrementar la frecuencia).

7.3. Detalles de implementación

7.3.1. Entrenamiento

El entrenamiento se realizó con el método del descenso por el gradiente de la función del error cuadrático medio.

Se utilizaron lotes del tamaño del juego de datos de entrenamiento completo para el modelo MIB y de tamaño 216 para MIA. La cantidad de ciclos dependió del rendimiento del modelo. En todos los casos el entrenamiento se cortó cuando el modelo comenzó a sobreajustar o dejó de mejorar. Cada 100 ciclos se controló el error absoluto medio. Si el error de validación aumentaba y el de entrenamiento disminuía significativamente, el entrenamiento se detenía y se tomaban los valores de los parámetros y rendimiento del modelo al momento del corte. No se tomaron los parámetros en el mínimo del error de validación para evitar caer en soluciones adaptadas al juego de datos de validación.

7.3.2. Condiciones iniciales

Salvo por las excepciones enumeradas a continuación, los pesos sinápticos se inicializaron con valores aleatorios tomados de una distribución uniforme con límites $[-lim, lim]$, donde $lim = \sqrt{6/(n_e + n_s)}$, n_e es la cantidad de entradas de la capa y n_s es la cantidad de salidas.

Excepciones:

- Los pesos de la capa STHadamard (solo modelo MIA) se inicializaron con una ventana Hann discretizada.

- Los pesos de la capa A2 (solo modelo MIA) se inicializaron con los coeficientes de DFT (matrices $\boldsymbol{F}_C$ y $\boldsymbol{F}_S$ en la sección 4.2.2.2).
- Los pesos de la capa A6 (solo modelo MIA) se inicializaron con los coeficientes de la DCT (ver sección 4.2.2.2).
- Los pesos de las capa de suavizado (C1) se inicializaron con una ventana gaussiana discretas de la longitud correspondiente.

La condiciones iniciales presentadas se utilizaron sobre los primeros modelos. Algunos detalles cambiaron durante el proceso de optimización.

7.3.3. Entorno de desarrollo y ejecución

El lenguaje de desarrollo del modelo es Python 3.7. Para la red neuronal se utiliza la librería *Keras*[4] y TensorFlow[5] como *backend*. En el ajuste de los pesos, con el error cuadrático medio como función error, se usa el optimizador Adam provisto por *Keras* con tasa de aprendizaje = 0.001. El entorno de ejecución es una PC con procesador Intel Core i7-8700, 32 GB de memoria RAM, GPU NVIDIA Titan XP, sistema operativo Ubuntu 20.04 y la plataforma CUDA versión 10.0 para el uso de la GPU.

7.4. Esquema de evaluación de resultados

Tal como se explica en la sección 3.2, las métricas para evaluar el rendimiento del modelo son la exactitud y el error absoluto medio. Durante la validación cruzada k-fold se calculó la media aritmética de las métricas entre las k iteraciones. El proceso completo se repitió 10 veces inicializando el generador de números aleatorios con semillas distintas y se calculó la media de las métricas. Si en alguno de los entrenamientos se detectaba que la red caía en un mínimo local, se repetían todas las iteraciones con una semilla distinta. Se consideró que esta situación ocurría cuando el error sobre los datos de validación era mayor que 0.5 y existían pesos en las capa B1 (o B2 del modelo final presentado en el capítulo 8) que no se modificaron durante el entrenamiento o, en caso de usar algún tipo de regularización, se hicieron nulos.

Para la validación final, solo sobre el modelo definitivo, se realizaron 10 entrenamientos bajo las mismas condiciones utilizando el juego de datos de test y entrenando con los cuatro restantes.

[4] https://keras.io
[5] https://www.tensorflow.org

Capítulo 8

Resultados

Durante el proceso de optimización de los modelos iniciales, y a pesar de probar muchas alternativas, no se logró ninguna configuración que produjera resultados satisfactorios partiendo del modelo MIA. En contraste, partiendo desde el modelo MIB sí se logró predecir el grado general de disfonía. En las secciones siguientes se presentan los resultados del modelo definitivo obtenido a partir de MIB.

8.1. Modelo final

El modelo final se obtuvo a partir del modelo MIB, por lo tanto no incluye las capas de representación frecuencial. La entrada del modelo es el cepstrum en el tiempo o cepstrograma sin las 20 primeras *quefrencies* para eliminar el efecto del filtro, tal como se explicó en la sección 5.1.3.1. Al igual que en MIA y MIB, el tamaño del cepstrograma es 420×117. En la figura 6.1 se muestra el modelo final completo.

8.1.1. Escalas de suavizado

Durante el proceso de optimización se observó que suavizar la entrada, no solo para el cálculo de CPPS, sino para toda la extracción de características, mejora la clasificación. El suavizado, al igual que en el modelo inicial, se realiza con una capa de convolución.

Durante las pruebas también se observó que era beneficioso realizar distintos grados o escalas de suavizado, es decir más de un suavizado con ventanas de ancho distinto, y extraer características de cada escala. Las razones de este comportamiento se discuten en el capítulo 9.

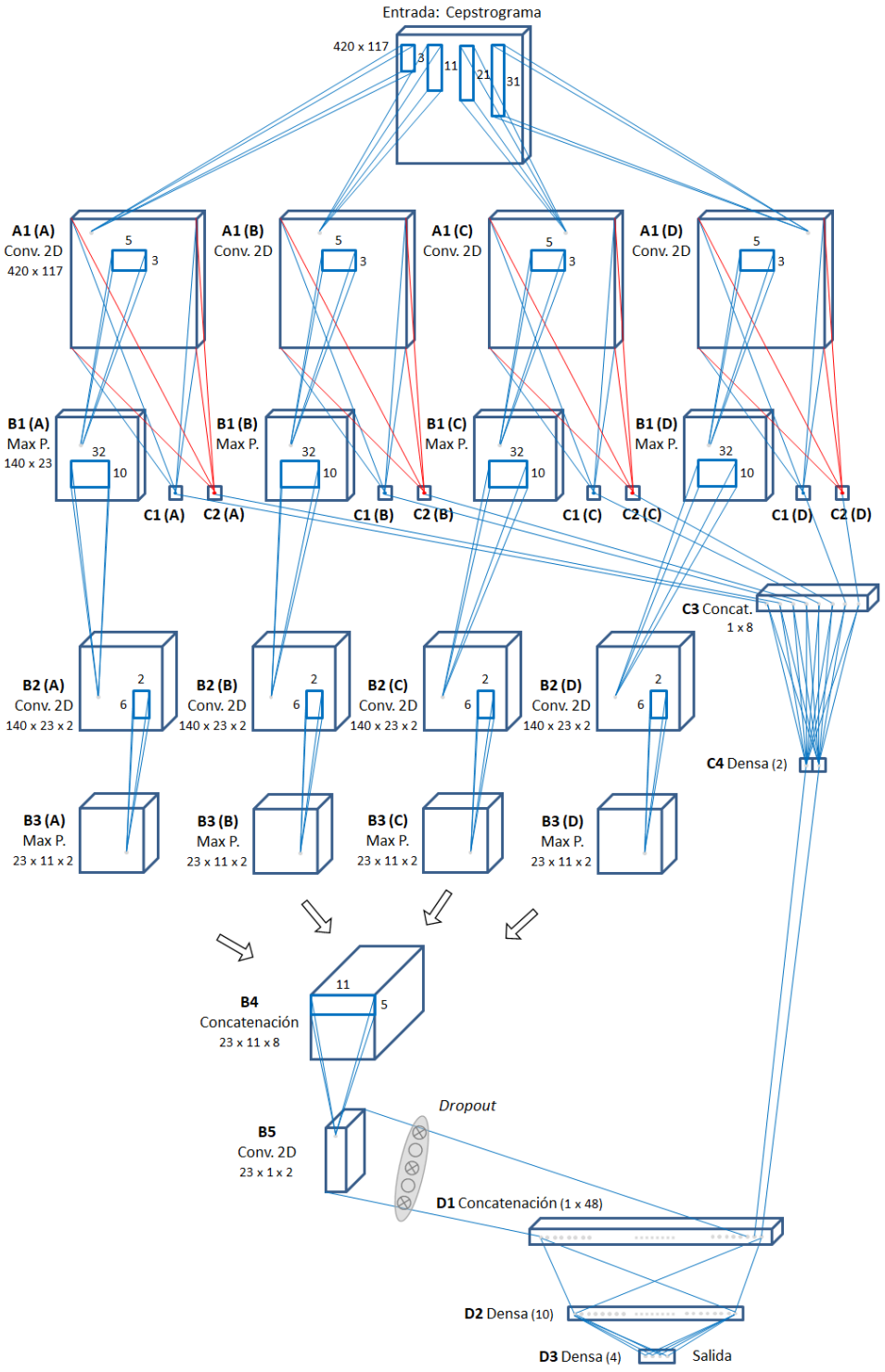


FIGURA 8.1: Modelo final de clasificación del grado general de disfonía.

En el modelo final se realizan suavizados en paralelo con *kernels* de ancho 1, pero distintas alturas (3, 11, 21 y 31) y desplazamientos de 1×1. El cálculo se realiza en las capas de convolución con activación lineal A1. Los pesos son inicializados con los valores de una campana gaussiana y no se modifican durante el entrenamiento. En la figura 8.2 se muestra un ejemplo de la salida de las cuatro capas de convolución del suavizado.

Estas capas tienen *padding* = "same", por lo tanto mantienen el tamaño de la capa anterior.

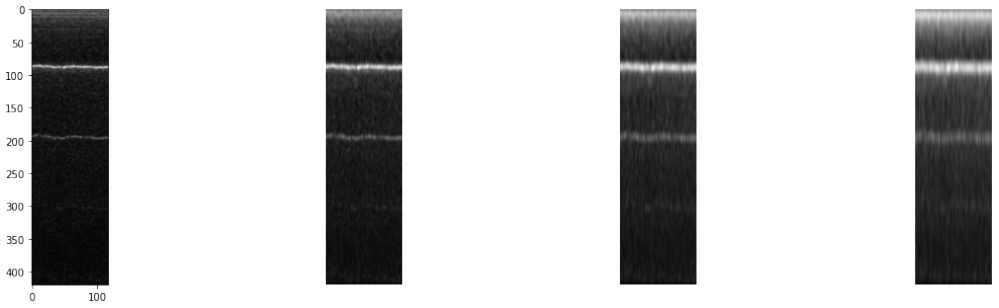


FIGURA 8.2: Salidas de las capas de suavizado en el modelo final. De izquierda a derecha, A1(A), A1(B), A1(C) y A1(D).

8.1.2. Extracción de características

La extracción de características, al igual que en el modelo inicial, se hace por dos caminos distintos de la red. A diferencia del modelo inicial, en el modelo final la entrada de cada camino no es el cepstrograma sino los suavizados del mismo.

Camino 1. Comienza con una capa de *max pooling* (B1) para cada capa de suavizado. La ventana de estas capas tiene tamaño 3×5 (alto $\times$ ancho) y desplazamiento 3×5. Se usa *padding* = "valid", lo que resulta en un vector de salida de tamaño $140 \times 23 \times 1$. La salida de estas capas es la entrada de las capas de convolución B2, con 2 *kernels* de tamaño 10×32, desplazamiento 1×1, *padding* = "same" y función de activación ReLU. El tamaño de las salidas de las capas B2 es $140 \times 23 \times 2$. Después, la señal pasa por las capas de *max pooling* B3, con ventanas de tamaño 6×2, desplazamiento 6×2 y *padding* = "valid", los que resulta en vectores de tamaño $23 \times 11 \times 2$. La salida de las últimas cuatro capas de *max pooling* se reordena (B4) en un vector de tamaño $23 \times 11 \times 8$. Sobre este vector se hace una nueva convolución (B5) con 2 *kernels* de tamaño 5×11, desplazamiento 1×11, *padding* = "same" y función de activación ReLU. El tamaño de la salida de este camino es $23 \times 1 \times 2$. En las capas B2 y B5 se aplica regularización L2 con penalidad $\lambda = 0,001$.

Camino 2. Sobre cada capa de suavizado (A1) se conecta una capa de *max pooling* (C1) y una de *average pooling* (C2). Ambas con ventana de tamaño 420×117, el mismo tamaño que la salida de las capas de suavizado, por lo tanto el tamaño de las salidas de C1 y C2 es 1×1. Luego, las cuatro capas C1 y las cuatro C2 se reordenan en C3 y se utilizan como entrada para dos neuronas densamente conectadas (C4) con función de activación ReLU.

8.1.3. Clasificación

En D1 los dos caminos de extracción de características se unen, las salidas de ambos se reordenan y concatenan formando un vector de tamaño 48. Este vector es la entrada de una capa densa (D2) de 3 neuronas con función de activación ReLU, la que se conecta a la capa densa D3, con 10 neuronas con activación ReLU y, por último, la salida de D3 es la entrada de las cuatro neuronas de salida, que tienen función de activación sigmoidal.

Para el cálculo de los indicadores, la salida de la red con mayor valor se toma como 1 y el resto es 0.

8.1.4. Parámetros

La tabla 8.1 muestra las capas del modelo final que tienen parámetros entrenables y la cantidad de parámetros de cada una.

Tabla 8.1: Cantidad de parámetros entrenables por capa del modelo final.

Código	Tipo	Función	Parámetros
A1	Conv. 2D	Suavizado	66
B2	Conv. 2D	Ext. caract.	2568
B5	Conv. 2D	Ext. caract.	882
C4	Densa	Ext. caract.	18
D2	Densa	Clasificación	147
D3	Densa	Clasificación	40
D4	Densa	Clasificación	44
		Total	3765

8.2. Métricas

El modelo final obtuvo una exactitud media de 0.711 y error absoluto medio 0.303 como resultado de las evaluaciones realizadas según las definiciones de la sección 7.4.

En la figura 8.3 se muestran los diagramas de caja de las dos métricas para las 10 ejecuciones del proceso. Los resultados se comparan con la concordancia y el error absoluto intraevaluador e interevaluador para los mismos datos.

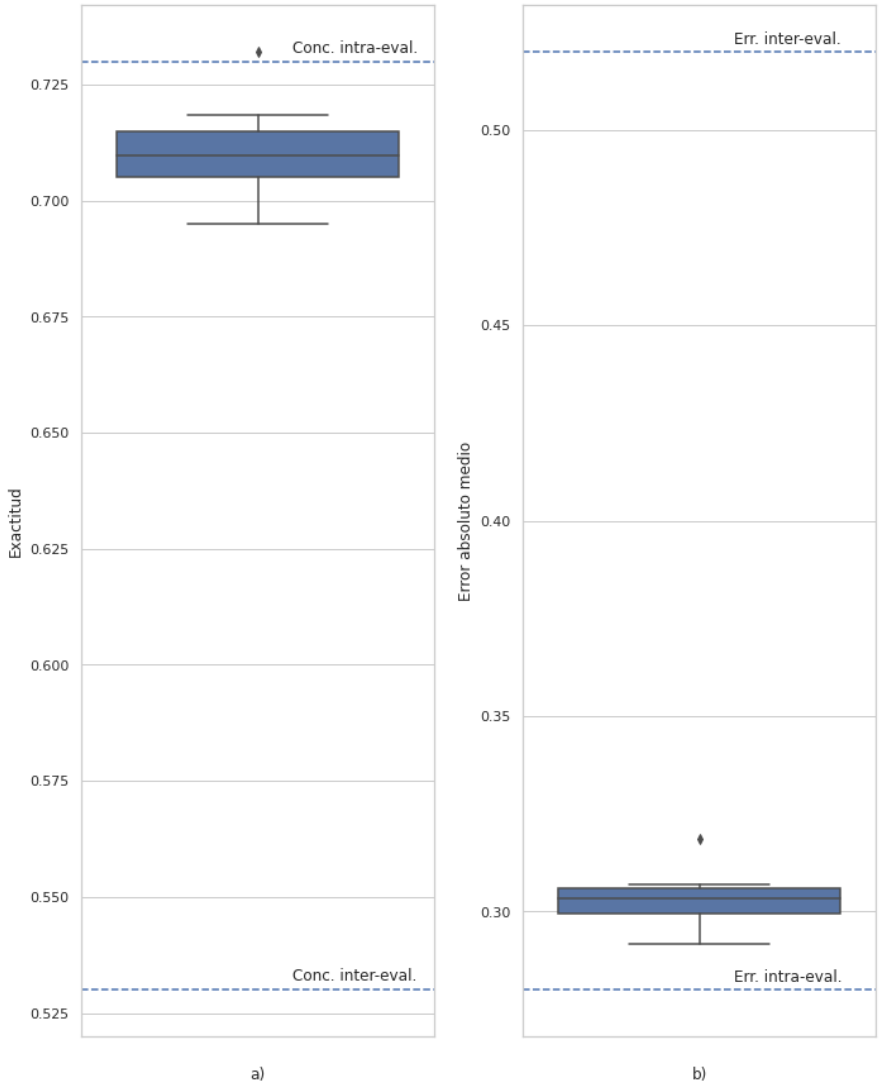


FIGURA 8.3: Diagrama de caja para la exactitud (a) y el error absoluto medio (b) en los resultados del experimento. En ambos casos se puede ver la comparación con los valores (concordancia y error absoluto) medios interevaluador e intraevaluador.

En la matriz de confusión de la tabla 8.2 se comparan las valoraciones del evaluador local con las salidas del clasificador para la primera ejecución. En esta iteración, la exactitud fue 0.713 y el error absoluto medio 0.292.

8.3. Pesos y salidas intermedias

Las redes neuronales artificiales frecuentemente son consideradas un modelo de "caja negra" porque no es posible interpretar, al menos de forma intuitiva, cuáles son las relaciones que detectan en los datos de entrada para realizar la clasificación. Asimismo, en algunos casos el análisis de los parámetros obtenidos durante el entrenamiento puede ayudar en la comprensión.

Tabla 8.2: Matriz de confusión entre la valoración del evaluador local y la valoración automática para la primera ejecución de la evaluación.

		Red neuronal			
		0	1	2	3
Eval. local	0	67	41	0	0
	1	14	70	24	0
	2	0	26	74	8
	3	0	2	9	97

Como es natural, los pesos donde es más fácil interpretar qué información es útil para el modelo son los pertenecientes a los primeros filtros de extracción de características. En este caso, son los pesos de las capas B2.

En la figura 8.4 se muestran los pesos de las capas B2 del modelo final entrenado para 500 ciclos de entrenamiento y su evolución en el tiempo. Intencionalmente se grafican los pesos de un caso donde dos *kernels* (el primero para suavizado $n = 21$ y el segundo para suavizado $n = 31$) no llegaron a una configuración útil. Se puede ver que los pesos antes mencionados tienden a cero por el efecto de la regularización. El resto de los pesos de B2 toman formas bien definidas, con variación de magnitud más rápida (frecuencia más alta) en el sentido vertical (*quefrency*) que en el horizontal (tiempo). Claramente la frecuencia de variación en el sentido vertical disminuye a medida que aumenta n.

En la figura 8.5 se muestran las salidas de las capas B2 para un caso de G="0" utilizando los pesos de la figura 8.4. Notar que las salidas de los primeros *kernels* que reciben como entrada los suavizados para $n = 21$ y $n = 31$ son nulas.

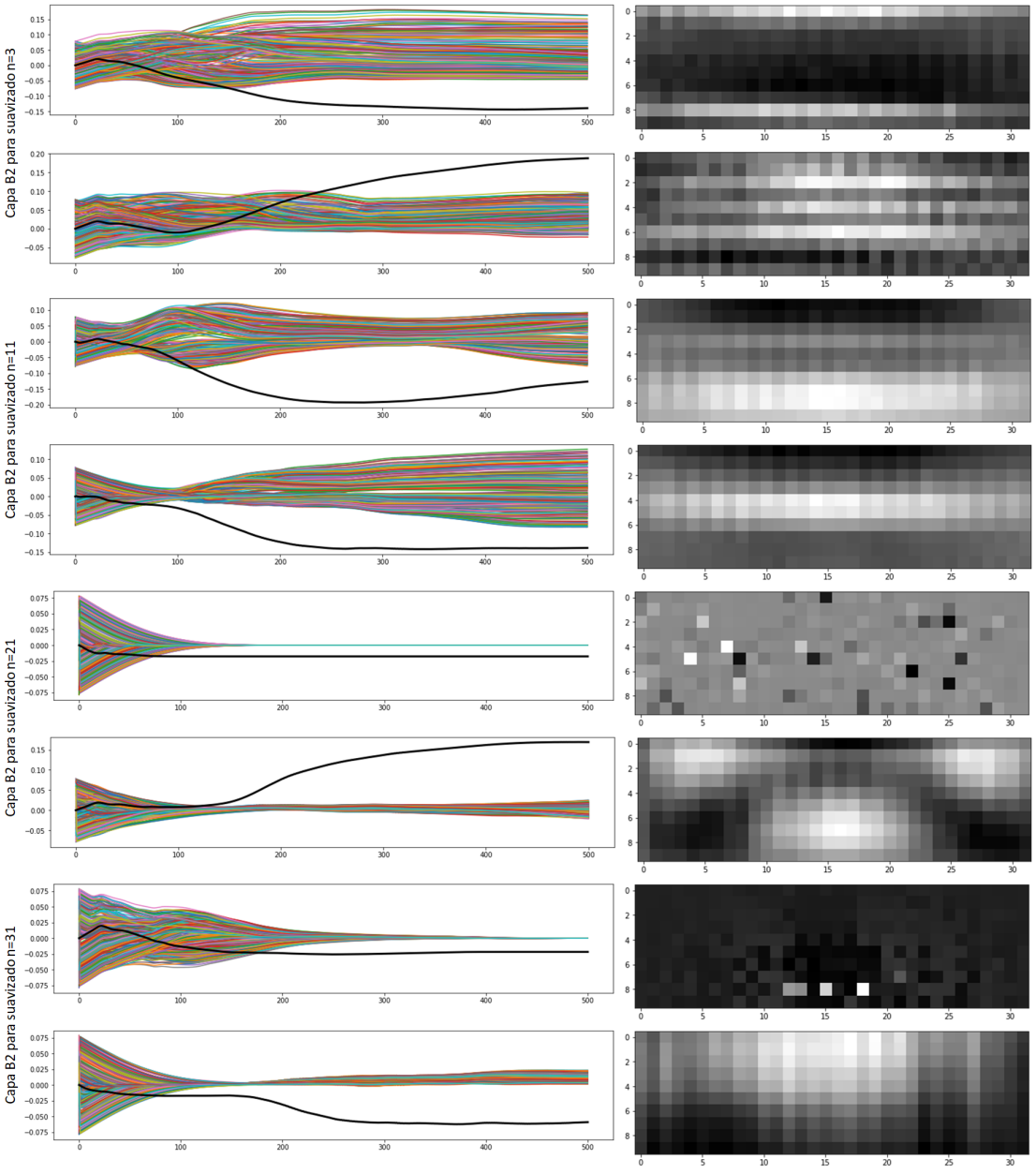


FIGURA 8.4: Pesos de las capas B2 en el modelo final entrenado. A la derecha se muestran los pesos de los *kernels* de convolución y a la izquierda la evolución de cada uno durante el entrenamiento. Las líneas gruesas en negro representan los *bias*.

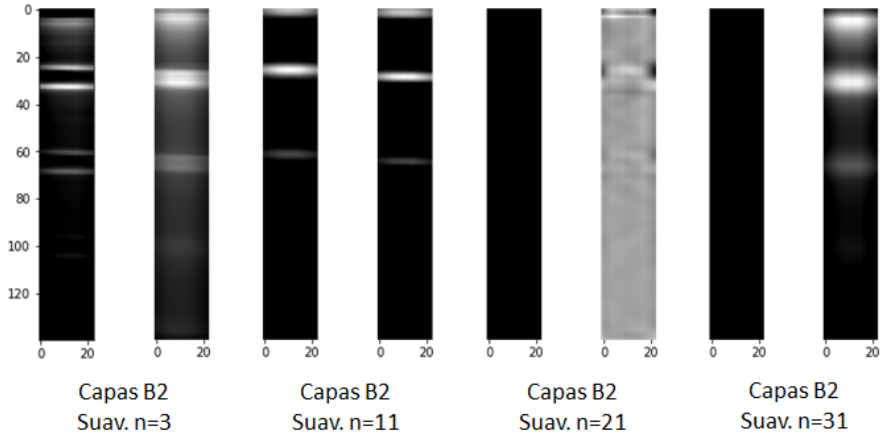


FIGURA 8.5: Salidas de las capas B2 del modelo final para los pesos de la figura 8.4 utilizando una entrada con G="0".

Capítulo 9

Discusión

9.1. Interpretación de los resultados

9.1.1. Métricas

Una clasificación, con datos balanceados, realizada al azar sobre cuatro clases tendría un acierto del 25 %, por lo tanto el 71.1 % de acierto informado en el capítulo 8 evidencia que la red neuronal profunda obtenida es capaz de reconocer ciertos patrones de la voz que están relacionados con la percepción de la calidad.

No solo la exactitud indica un buen rendimiento, el error absoluto medio y la matriz de confusión (tabla 8.2) muestran que, para los casos donde no hay coincidencia, el modelo generalmente asigna clases cercanas a la de referencia; tal como ocurre entre expertos humanos, donde es poco frecuente que la valoración difiera en más de un grado de disfonía. Notar que en la matriz de confusión, las clasificaciones erróneas (fuera de la diagonal) se concentran cerca de la diagonal y son nulas en los extremos más lejanos de la misma.

Es difícil comparar los resultados obtenidos con los resultados de otros trabajos sobre el tema, principalmente, porque se utilizan bases de datos diferentes. Tal como se puede ver en la sección 1.5, en la literatura se encuentran artículos con predicciones de G superiores al 80 % de exactitud [31, 32, 35]. Por otro lado, el grupo de Godino-Llorente, un referente en el área, obtuvo resultados (exactitud y error absoluto medio) inferiores a los de esta investigación en dos trabajos recientes [36, 37]. Un análisis más útil que la comparación directa de las exactitudes logradas en distintos trabajos podría obtenerse de la camparación de las relaciones entre las exactitudes y los índices de concordancia interevaluador e intraevaluador de las bases de datos utilizadas, tal como se propone en

la sección 3.2, pero lamentablemente esta información no está disponible en los artículos mencionados.

En este trabajo, y para ambas métricas, el clasificador automático logra valoraciones con variaciones muy cercanas a las que obtendría un evaluador medio tras evaluar por segunda vez el mismo conjunto de audios (variabilidad intraevaluador). También para ambas métricas se puede ver una gran mejora comparando con la concordancia y el error interevaluador.

Los resultados son prometedores y permiten pensar en futuras aplicaciones clínicas de clasificadores basados en la red neuronal obtenida. Por un lado, los valores medios de las métricas son cercanos a los valores equivalentes de concordancia entre distintas valoraciones de un mismo evaluador experto, por lo tanto la utilización clínica permitiría que profesionales con menor experiencia obtengan una valoración automática como referencia que se pueda tomar en momentos distintos de un tratamiento. Por otro lado, la exactitud es mucho mayor que la concordancia entre evaluadores humanos, por lo tanto el uso extendido de la clasificación automática tendría un efecto estandarizador entre distintos profesionales y centros de salud.

9.1.2. Arquitectura del modelo

El aprendizaje profundo es el estado del arte en la mayoría de las tareas de reconocimiento de patrones. El reconocimiento de patrones en audio no es la excepción, aunque los avances importantes se han logrado en el reconocimiento del habla (*speech recognition*). Las áreas de la calidad vocal y detección/clasificación de patologías han sido poco abordadas con esta tecnología. Entre los trabajos realizados, varios utilizan modelos neuronales existentes que fueron desarrollados para reconocimiento de imágenes [40, 42, 43], mientras que otros solo utilizan capas densas [32, 41]. Se han presentado pocos modelos desarrollados *ad hoc*. En la categoría de los modelos *ad hoc* están los desarrollados por Arias-Lodoño *et al.* [37], pero los autores no reportan buenos resultados, ya que los comparan con un modelo neuronal más simple que recibe medidas acústicas como entrada, y este último funciona mejor. Harar *et al.* también presentan una red neuronal diseñada *ad hoc* para clasificar entre voces sanas y patológicas [38]. El modelo de Harar es completamente distinto al de Arias-Lodoño y los resultados generan dudas porque los datos utilizados no están bien balanceados.

En este contexto, donde no hay conocimiento sobre las características que debe tener una red neuronal profunda para clasificar la calidad vocal, los desarrollos del presente trabajo pueden aportar información útil.

En primer lugar, las capas de extracción de características divididas en dos caminos, donde uno fue pensado para obtener información sobre el ruido y el otro sobre las perturbaciones de amplitud y frecuencia, son novedosas y, de acuerdo con los resultados, cumplen su propósito. El uso de dos caminos separados con capas convolucionales fue utilizado también en el trabajo de Arias-Lodoño, aunque con objetivos distintos; se utilizó para obtener características acústicas y de modulación por separado sobre entradas de *modulation spectra*.

Con respecto a las distintas escalas de suavizado, el efecto de mejora puede estar relacionado con un concepto bien conocido en el reconocimiento de patrones sobre imágenes, la teoría del espacio escalar. Esta propone que el ojo humano es capaz de reconocer patrones en distintos niveles de suavizado (al mismo tiempo) de la imagen original y que los patrones reconocidos en diferentes niveles mejoran la clasificación general. Existen trabajos de reconocimiento de patrones en audio que utilizan la misma idea [104, 112, 174, 175], pero no sobre la calidad vocal.

Con respecto a la representación frecuencial del audio, en los experimentos se mostró que, para una extracción de características basada en el cepstrum, la utilización del logaritmo en el cálculo es necesaria y la ventaja de ajustar los parámetros de las capas anteriores no es suficiente para eliminarlo. No obstante, se piensa que el ajuste de pesos en la capa STHadamard y en el cálculo de la STFT con redes neuronales podría ser beneficioso para otro tipo de extracción de características, aunque es necesario diseñar nuevos experimentos para probarlo. En la sección 4.3 se mostró que la capa STHadamard se puede entrenar y que, si bien marginal en ese caso, el efecto fue positivo.

Durante los experimentos se analizó la evolución y la configuración final de los pesos sinápticos, pero por esta vía no se logró obtener información clara sobre las características relevantes del modelo para la clasificación. La falta de complejidad del modelo (pocos *kernels* en este caso) puede tener relación con la dificultad para encontrar estos patrones. Tampoco se encontraron relaciones entre la presencia de entradas pertenecientes a ciertas clases y zonas de activación de neuronas en las capas de convolución. Sí se notó que las activaciones de las capas B2 disminuyen levemente a medida que aumenta el grado de disfonía. Esto podría indicar que algunos *kernels* están detectando las regularidades que existen en las voces normales y que se pierden en los niveles más bajos de calidad, donde la señal se vuelve más caótica.

9.2. Limitaciones

Debido a la escasa cantidad de datos disponibles para entrenar con clases balanceadas, no se pudo utilizar modelos más complejos. Es posible que incrementando la complejidad de la red neuronal, como por ejemplo aumentando la cantidad de *kernels* en las capas de convolución de extracción de características, el modelo mejore el rendimiento, pero con la cantidad de datos utilizados el aumento de los parámetros produce sobreajuste.

Con respecto a la representación frecuencial (en ambos casos, utilizada como entrada o calculada dentro de la red neuronal), existen muchas opciones además del cepstrum, como por ejemplo el espectrograma y las transformadas *wavelet*. El estudio de estas alternativas quedó fuera del alcance de este trabajo, pero es necesario un análisis más completo para determinar cuál es la representación frecuencial más útil en la arquitectura de una red neuronal profunda diseñada con el enfoque de esta investigación.

En el último párrafo de la sección 9.1.1 se menciona la posibilidad de lograr aplicaciones clínicas basadas en la red neuronal obtenida. Si bien se piensa que el modelo presentado es un buen punto de partida para futuros trabajos y que el conocimiento generado ayudará a acercar los sistemas inteligentes a la práctica clínica en el área vocal, no se debe interpretar que el modelo final presentado se puede utilizar de forma directa en la práctica clínica. Esta es una situación frecuente en la inteligencia artificial aplicada a la medicina. A pesar de los esfuerzos, la mayoría de los avances que la inteligencia artificial y el aprendizaje profundo logran en los trabajos científicos no llegan a utilizarse en la práctica clínica. Este tema es actualmente muy estudiado en el área de procesamiento automático de imágenes médicas y los desafíos que aún deben abordarse antes de que la inteligencia artificial pueda alcanzar un valor clínico han sido tratados en numerosos trabajos [176–181]. Muchos de estos desafíos son comunes a los encontrados en el área de la patología vocal. Uno de los obstáculos más importantes es que los datos utilizados para ajustar los modelos generalmente pertenecen a una población particular. Un ejemplo claro es el del trabajo de Kather et al. [182], donde se reconoce la inestabilidad de microsatélites (MSI) sobre imágenes de microscopio utilizando dos redes neuronales ResNet-18, una para reconocer el tumor y otra para clasificar la MSI. El rendimiento se evaluó con el área bajo la curva ROC (métrica para evaluación de modelos cuyo valor óptimo es 1) y el área obtenida fue 0.84 sobre la base de datos de validación. El mismo modelo logró un área bajo la curva menor que 0.69 cuando se evaluó con una base de datos japonesa, probablemente reflejo de que la base de datos de entrenamiento estaba compuesta por un 80 % de no asiáticos. Esto indica la necesidad de datos de entrenamiento multicéntricos para obtener modelos de clasificación más generales. Además de las diferencias entre pacientes de distintas regiones, las muestras de distintos laboratorios tienen gran variabilidad en la tinción, la calidad de la imagen, las características

de escaneo y la preparación de los tejidos. En la clasificación de la calidad vocal, para lograr un clasificador suficientemente general, será necesario contar con bases de datos que tengan variabilidad en la edad, ocupación, género, idioma y región geográfica de las personas, además de variabilidad en las condiciones de grabación (ambiente, micrófono, placa de sonido y formato entre otros). Obtener grabaciones con la variabilidad necesaria es una tarea difícil, pero también lo es la clasificación de todas las voces por un número suficiente de especialistas (por lo menos dos veces cada uno para medir la concordancia interevaluador). El principal desafío para lograr la aplicación de la inteligencia artificial en la práctica de la clínica vocal probablemente esté relacionado con lograr datos generales y en mayor volumen para ajustar los modelos. Después, el uso de más datos y más descentralizados, permitirá y/o forzará cambios en la arquitectura de la red neuronal.

9.3. Trabajos futuros

En investigaciones futuras se debería ampliar el estudio de modelos de representación frecuencial del audio para utilizar como entrada *raw audio*. Se puede trabajar tanto con redes neuronales basadas en representaciones predeterminadas, como con arquitecturas más generales. Para los casos donde la red debe ajustarse a la señal original (*raw audio*) sin configuraciones predeterminadas (a diferencia del trabajo actual, donde se inicializaron los pesos con los coeficientes de la DFT por ejemplo) es necesario contar con volúmenes de datos mayores. En estos casos se podría entrenar previamente modelos que calculen medidas acústicas con el objetivo de ajustar los pesos de las primeras capas. Para entrenar modelos que calculen medidas acústicas no es necesario contar con datos clasificados, por lo tanto se pueden utilizar bases de datos que solo contienen audios, como la *Saarbrücken Voice Database*[1].

Una tarea necesaria es la de obtener nuevos datos con grabaciones locales y nuevas valoraciones para datos locales y externos con el objetivo de ampliar la base de datos. En todos los casos será necesaria la evaluación repetida de varios evaluadores. Con mayor cantidad de datos se pueden estudiar modelos más complejos para clasificar el grado general de disfonía G. Además, la generación de datos locales aportaría variabilidad al universo de datos vocales, lo que es necesario para lograr la aplicación en la práctica clínica de los sistemas automáticos de clasificación de la calidad vocal.

Otra línea de investigación que se desprende de este trabajo es la clasificación del resto de las dimensiones GRBAS. La distribución de las clases para estas dimensiones en la base de datos utilizada es menos favorable que para G (existen menos ocurrencias con grado “3”), en consecuencia, como primer paso sería necesaria la ampliación de la base

[1] http://stimmdb.coli.uni-saarland.de

de datos. G es la dimensión más compleja porque incluye componentes del resto, por lo tanto el modelo presentado también puede ser un buen punto de partida para clasificar R, B, A y S.

Capítulo 10

Conclusión

Se concluye que el modelo obtenido es capaz de predecir el grado general de disfonía con una exactitud cercana a la de un evaluador humano para los datos de la *Perceptual Voice Qualities Database*.

Con respecto a la arquitectura del modelo se concluye que el cepstrum no puede ser calculado por una red neuronal si los pesos de las capas anteriores deben ser ajustados y que, al menos bajo las condiciones de los experimentos realizados, una red neuronal no puede calcular una representación interna equivalente al cepstrum que sea útil para la clasificación. En referencia a la extracción de características, se concluye que una red neuronal profunda diseñada para reconocer patrones de perturbación de amplitud, perturbación de frecuencia y ruido obtiene información útil para la predicción del grado general de disfonía.

Por último, se concluye que el modelo presentado puede ser un punto de partida para futuros desarrollos de clasificadores de la calidad vocal, los cuales deberían ser entrenados y evaluados con bases de datos más grandes y con mayor variabilidad para lograr su aplicación en la práctica clínica.

Bibliografía

[1] Ben Barsties and Marc De Bodt. Assessment of voice quality: current state-of-the-art. *Auris Nasus Larynx*, 42(3):183–188, 2015.

[2] Hananel Hazan, Dan Hilu, Larry Manevitz, Lorraine O Ramig, and Shimon Sapir. Early diagnosis of parkinson's disease via machine learning on speech data. In *2012 IEEE 27th Convention of Electrical and Electronics Engineers in Israel*, pages 1–4. IEEE, 2012.

[3] Max Little, Patrick McSharry, Eric Hunter, Jennifer Spielman, and Lorraine Ramig. Suitability of dysphonia measurements for telemonitoring of parkinson's disease. *Nature Precedings*, pages 1–1, 2008.

[4] Luisa Fernanda Angel Gordillo. Hitos de la evaluación perceptual auditiva de la voz: ¿hay evidencia? *Areté*, 18(2):65–74, 2018.

[5] N Isshiki, N Yanagihara, and M Morimoto. Approach to the objective diagnosis of hoarseness. *Folia Phoniatrica et Logopaedica*, 18(6):393–400, 1966.

[6] Minoru Hirano and Karen R McCormick. Clinical examination of voice by minoru hirano, 1986.

[7] Young Sun Yun, Eun Kyung Lee, Chung Hwan Baek, and Young Ik Son. The correlation of grbas scales and laryngeal stroboscopic findings for the assessment of voice therapy outcome in the patients with vocal nodules. *Korean Journal of Otolaryngology-Head and Neck Surgery*, 48(12):1501–1505, 2005.

[8] Huangfu Hui, Kong Weijia, Gong Shusheng, et al. The validation of acoustic analysis and subjective judgment scales of several voice disorders [j]. *Journal of Audiology and Speech Pathology*, 3(010), 2007.

[9] Michael P Karnell, Sarah D Melton, Jana M Childes, Todd C Coleman, Scott A Dailey, and Henry T Hoffman. Reliability of clinician-based (grbas and cape-v) and patient-based (v-rqol and ipvi) documentation of voice disorders. *Journal of Voice*, 21(5):576–590, 2007.

[10] Luis MT Jesus, Anna Barney, Pedro Sá Couto, Helena Vilarinho, and Ana Correia. Voice quality evaluation using cape-v and grbas in european portuguese. In *Sixth International Workshop on Models and Analysis of Vocal Emissions for Biomedical Applications*, 2009.

[11] Jody Kreiman and Bruce R Gerratt. Perceptual assessment of voice quality: past, present, and future. *Perspectives on Voice and Voice Disorders*, 20(2):62–67, 2010.

[12] Faustino Núnez-Batalla, Juan Pablo Díaz-Molina, Isabel García-López, Adriana Moreno-Méndez, María Costales-Marcos, Carla Moreno-Galindo, and Pablo Martínez-Camblor. El espectrograma de banda estrecha como ayuda para el aprendizaje del método grabs de análisis perceptual de la disfonía. *Acta Otorrinolaringológica Española*, 63(3):173–179, 2012.

[13] Max Little, Declan Costello, and Meredydd Harries. Objective dysphonia quantification in vocal fold paralysis: comparing nonlinear with classical measures. *Nature Precedings*, pages 1–1, 2009.

[14] Susana Vaz Freitas, Pedro Melo Pestana, Vítor Almeida, and Aníbal Ferreira. Integrating voice evaluation: correlation between acoustic and audio-perceptual measures. *Journal of Voice*, 29(3):390–e1, 2015.

[15] Natalia Gabriela Elisei. Percepción auditiva de voces patológicas. In *XIV Reunión Nacional y III Encuentro Internacional de la Asociación Argentina de Ciencias del Comportamiento*, 2013.

[16] Faustino Núñez-Batalla, P. Corte Santos, B. Señaris González, N. Rodríguez Prado, and Carlos Suárez Nieto. Evaluación espectral cuantitativa de la hipofunción vocal. *Acta Otorrinolaringológica Española*, 55(7):327–333, 2004.

[17] Svante Granqvist and Britta Hammarberg. The correlogram: a visual display of periodicity. *The Journal of the Acoustical Society of America*, 114(5):2934–2945, 2003.

[18] John Kane and Christer Gobl. Identifying regions of non-modal phonation using features of the wavelet transform. *Proc. Interspeech 2011*, pages 177–180, 2011.

[19] Athanasios Tsanas, Matías Zañartu, Max A Little, Cynthia Fox, Lorraine O Ramig, and Gari D Clifford. Robust fundamental frequency estimation in sustained vowels: Detailed algorithmic comparisons and information fusion with adaptive kalman filtering. *The Journal of the Acoustical Society of America*, 135(5):2885–2901, 2014.

[20] Rick M Roark. Frequency and voice: perspectives in the time domain. *Journal of Voice*, 20(3):325–354, 2006.

[21] Gastón Schlotthauer, María Eugenia Torres, and Hugo L Rufiner. A new algorithm for instantaneous f 0 speech extraction based on ensemble empirical mode decomposition. In *2009 17th European Signal Processing Conference*, pages 2347–2351. IEEE, 2009.

[22] Dennis Gladis, Usha Dalvi, et al. A study of f0 estimation based on rapt framework using sustained vowel. In *2015 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pages 2290–2295. IEEE, 2015.

[23] Silvana C Costa, C de A Washington, Suzete EN Correia, Joseana MFR de Araújo, and Vinícius JD Vieira. Análise de sinais de voz para caracterização de patologias na laringe. *Revista de Tecnologia da Informação e Comunicação*, 4(2):63–70, 2014.

[24] Athanasios Tsanas, Max A Little, Patrick E McSharry, and Lorraine O Ramig. Nonlinear speech analysis algorithms mapped to a standard metric achieve clinically useful quantification of average parkinson's disease symptom severity. *Journal of the royal society interface*, 8(59):842–855, 2011.

[25] Eugene Buder. Acoustic analysis of voice quality: A tabulation of algorithms 1902-1990. *Voice Quality Measurement*, 9:119–244, 2000.

[26] Athanasios Tsanas, Max A Little, Cynthia Fox, and Lorraine O Ramig. Objective automatic assessment of rehabilitative speech treatment in parkinson's disease. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 22(1):181–190, 2013.

[27] Juan Ignacio Godino-Llorente, Pedro Gomez-Vilda, and Manuel Blanco-Velasco. Dimensionality reduction of a pathological voice quality assessment system based on gaussian mixture models and short-term cepstral parameters. *IEEE transactions on biomedical engineering*, 53(10):1943–1953, 2006.

[28] Meisam Khalil Arjmandi, Mohammad Pooyan, Mohammad Mikaili, Mansour Vali, and Alireza Moqarehzadeh. Identification of voice disorders using long-time features and support vector machine with different feature reduction methods. *Journal of Voice*, 25(6):e275–e289, 2011.

[29] Shanshan Liu, Nan Yan, Manwa L Ng, Lan Wang, and Zhijian Wang. Multidimensional acoustic analysis for evaluation of voice quality of unilateral vocal fold paralysis. In *2014 4th IEEE International Conference on Information Science and Technology*, pages 706–709. IEEE, 2014.

[30] Ping Yu, Zhijian Wang, Shanshan Liu, Nan Yan, Lan Wang, and Manwa Ng. Multidimensional acoustic analysis for voice quality assessment based on the grbas scale. In *The 9th International Symposium on Chinese Spoken Language Processing*, pages 321–325. IEEE, 2014.

[31] Zhijian Wang, Ping Yu, Nan Yan, Lan Wang, and Manwa L Ng. Automatic assessment of pathological voice quality using multidimensional acoustic analysis based on the grbas scale. *Journal of Signal Processing Systems*, 82(2):241–251, 2016.

[32] Simin Xie, Nan Yan, Ping Yu, Manwa L Ng, Lan Wang, and Zhuanzhuan Ji. Deep neural networks for voice quality assessment based on the grbas scale. *Interspeech 2016*, pages 2656–2660, 2016.

[33] RT Ritchings, Mark McGillion, and CJ Moore. Pathological voice quality assessment using artificial neural networks. *Medical engineering & physics*, 24(7-8):561–564, 2002.

[34] Juan Ignacio Godino-Llorente, Tim Ritchings, and Carl Berry. The effects of inter and intra speaker variability on pathological voice quality assessment. In *Third International Workshop on Models and Analysis of Vocal Emissions for Biomedical Applications*, 2003.

[35] Laureano Moro-Velázquez, Jorge Andrés Gómez-García, Juan Ignacio Godino-Llorente, and Gustavo Andrade-Miranda. Modulation spectra morphological parameters: a new method to assess voice pathologies according to the grbas scale. *BioMed research international*, 2015.

[36] Jorge Gómez-García, Laureano Moro-Velázquez, Janaina Mendes-Laureano, Germán Castellanos-Dominguez, and Juan Ignacio Godino-Llorente. Emulating the perceptual capabilities of a human evaluator to map the grb scale for the assessment of voice disorders. *Engineering Applications of Artificial Intelligence*, 82:236–251, 2019.

[37] Julián D Arias-Londoño, Jorge A Gómez-García, and Juan I Godino-Llorente. Multimodal and multi-output deep learning architectures for the automatic assessment of voice quality using the grb scale. *IEEE Journal of Selected Topics in Signal Processing*, 14(2):413–422, 2019.

[38] Pavol Harar, Jesus B Alonso-Hernandezy, Jiri Mekyska, Zoltan Galaz, Radim Burget, and Zdenek Smekal. Voice pathology detection using deep learning: a preliminary study. In *2017 international conference and workshop on bioinspired intelligence (IWOBI)*, pages 1–4. IEEE, 2017.

[39] Pavol Harar, Zoltan Galaz, Jesus B Alonso-Hernandez, Jiri Mekyska, Radim Burget, and Zdenek Smekal. Towards robust voice pathology detection. *Neural Computing and Applications*, pages 1–11, 2018.

[40] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[41] Shih-Hau Fang, Yu Tsao, Min-Jing Hsiao, Ji-Ying Chen, Ying-Hui Lai, Feng-Chuan Lin, and Chi-Te Wang. Detection of pathological voice using cepstrum vectors: A deep learning approach. *Journal of Voice*, 33(5):634–641, 2019.

[42] Ghulam Muhammad, Mohammed F Alhamid, Mansour Alsulaiman, and Brij Gupta. Edge computing with cloud for voice disorder assessment and treatment. *IEEE Communications Magazine*, 56(4):60–65, 2018.

[43] Musaed Alhussein and Ghulam Muhammad. Voice pathology detection using deep learning on mobile healthcare framework. *IEEE Access*, 6:41034–41041, 2018.

[44] Keunwoo Choi, Deokjin Joo, and Juho Kim. Kapre: On-gpu audio preprocessing layers for a quick implementation of deep neural network models with keras. *arXiv preprint arXiv:1706.05781*, 2017.

[45] Ingo R Titze, Jiaqi Jiang, and David G Drucker. Preliminaries to the body-cover theory of pitch control. *Journal of Voice*, 1(4):314–319, 1988.

[46] Ignacio Cobeta, Faustino Núñez, and Secundino Fernández. *Patología de la voz.* Marge Médica Books, 2013.

[47] Secundino Fernández González, Noelia Sánchez, Francisco Vázquez de la Iglesia, and Jorge Rey Martínez. Técnicas digitales para la valoración laringoscópica. *Revista de Medicina de la Universidad de Navarra*, pages 19–30, 2006.

[48] Gunnar Fant. *Acoustic Theory of Speech Production.* D A C S R Series. De Gruyter, 1970.

[49] Lawrence Raphael, Gloria Borden, and Katherine Harris. *Speech Science Primer: Physiology, Acoustics, and Perception of Speech.* Communication sciences. Lippincott Williams & Wilkins, 2007.

[50] Marco Guzmán. Terapia y entrenamiento de la voz con tracto vocal semiocluido. *Artículo de divulgación científica del área vocal, Escuela de Fonoaudiología, Universidad de Chile, Chile*, 2010.

[51] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall series in artificial intelligence. Prentice Hall/Pearson Education, 2003.

[52] Elaine Rich and Kevin Knight. *Artificial Intelligence*. Artificial Intelligence Series. McGraw-Hill, 1991.

[53] Tom Mitchell. *Machine Learning*. McGraw-Hill series in Computer Science. McGraw-Hill, 1997.

[54] Eduardo Destéfanis. *Inteligencia Artificial Aplicada*. Alejandría, 2011.

[55] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer Series in Statistics. Springer New York, 2013.

[56] Christopher Bishop. *Pattern Recognition and Machine Learning*. Information Science and Statistics. Springer, 2006.

[57] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. Adaptive Computation and Machine Learning series. MIT Press, 2016.

[58] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[59] Dong Yu and Li Deng. *Automatic Speech Recognition: A Deep Learning Approach*. Signals and Communication Technology. Springer London, 2014.

[60] Kunihiko Fukushima and Sei Miyake. Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition. In *Competition and cooperation in neural nets*, pages 267–285. Springer, 1982.

[61] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[62] Yoshua Bengio, Paolo Frasconi, and Patrice Simard. The problem of learning long-term dependencies in recurrent networks. In *IEEE international conference on neural networks*, pages 1183–1188. IEEE, 1993.

[63] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.

[64] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial*

intelligence and statistics, pages 315–323. JMLR Workshop and Conference Proceedings, 2011.

[65] Ovidiu Calin. *Deep Learning Architectures: A Mathematical Approach*. Springer Series in the Data Sciences. Springer International Publishing, 2020.

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

[67] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[68] Connor Shorten and Taghi M Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1):1–48, 2019.

[69] Jan Schlüter and Thomas Grill. Exploring data augmentation for improved singing voice detection with neural networks. In *ISMIR*, pages 121–126, 2015.

[70] Xiaodong Cui, Vaibhava Goel, and Brian Kingsbury. Data augmentation for deep neural network acoustic modeling. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 23(9):1469–1477, 2015.

[71] Juhan Nam, Keunwoo Choi, Jongpil Lee, Szu-Yu Chou, and Yi-Hsuan Yang. Deep learning for audio-based music classification and tagging: Teaching computers to distinguish rock from bach. *IEEE signal processing magazine*, 36(1):41–51, 2018.

[72] Li Deng Dong Yu. *Automatic speech recognition, a deep learning approach*. Springer-Verlag London, 1 edition, 2014.

[73] Yedid Hoshen, Ron J Weiss, and Kevin W Wilson. Speech acoustic modeling from raw multichannel waveforms. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pages 4624–4628. IEEE, 2015.

[74] Tara N Sainath, Ron J Weiss, Andrew Senior, Kevin W Wilson, and Oriol Vinyals. Learning the speech front-end with raw waveform cldnns. In *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[75] Jody Kreiman and Bruce R Gerratt. Validity of rating scale measures of voice quality. *The Journal of the Acoustical Society of America*, 104(3):1598–1608, 1998.

[76] Jody Kreiman and Bruce R Gerratt. The perceptual structure of pathologic voice quality. *The Journal of the Acoustical Society of America*, 100(3):1787–1795, 1996.

[77] Patrick Walden. Perceptual voice qualities database (pvqd): Database characteristics. *Journal of Voice: Official Journal of the Voice Foundation*, 2020.

[78] Juan Ignacio Godino-Llorente, Víctor Osma-Ruiz, Nicolás Sáenz-Lechón, Ignacio Cobeta-Marco, Ramón González-Herranz, and Carlos Ramírez-Calvo. Acoustic analysis of voice using wpcvox: a comparative study with multi dimensional voice program. *European archives of oto-rhino-laryngology*, 265(4):465–476, 2008.

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

[80] Ronan Collobert, Christian Puhrsch, and Gabriel Synnaeve. Wav2letter: an end-to-end convnet-based speech recognition system. *arXiv preprint arXiv:1609.03193*, 2016.

[81] Yong Xu, Jun Du, Li-Rong Dai, and Chin-Hui Lee. An experimental study on speech enhancement based on deep neural networks. *IEEE Signal processing letters*, 21(1):65–68, 2014.

[82] Michel JAM Putten, Sebastian Olbrich, and Martijn Arns. Predicting sex from brain rhythms with deep learning. *Scientific reports*, 8(1):3069, 2018.

[83] Oscar Moreira-Tamayo and J Pineda De Gyvez. Filtering and spectral processing of 1-d signals using cellular neural networks. In *Circuits and Systems, 1996. ISCAS'96., Connecting the World., 1996 IEEE International Symposium on*, volume 3, pages 76–79. IEEE, 1996.

[84] Rosemarie Velik. Discrete fourier transform computation using neural networks. In *2008 International Conference on Computational Intelligence and Security*, pages 120–123. IEEE, 2008.

[85] J McClellan and T Parks. Eigenvalue and eigenvector decomposition of the discrete fourier transform. *IEEE Transactions on Audio and Electroacoustics*, 20(1):66–74, 1972.

[86] Chiheb Trabelsi, Olexa Bilaniuk, Ying Zhang, Dmitriy Serdyuk, Sandeep Subramanian, João Felipe Santos, Soroush Mehri, Negar Rostamzadeh, Yoshua Bengio, and Christopher J Pal. Deep complex networks. *arXiv preprint arXiv:1705.09792*, 2017.

[87] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[88] A Michael Noll. Cepstrum pitch determination. *The journal of the acoustical society of America*, 41(2):293–309, 1967.

[89] Alan V Oppenheim and Ronald W Schafer. From frequency to quefrency: A history of the cepstrum. *IEEE signal processing Magazine*, 21(5):95–106, 2004.

[90] Bruce P Bogert. The quefrency alanysis of time series for echoes; cepstrum, pseudo-autocovariance, cross-cepstrum and saphe cracking. *Time series analysis*, pages 209–243, 1963.

[91] Robert Randall, Jérôme Antoni, and Wade Smith. A survey of the application of the cepstrum to structural modal analysis. *Mechanical Systems and Signal Processing*, 118:716–741, 2019.

[92] Catherine Madill, Duong Duy Nguyen, Kristie Yick-Ning Cham, Daniel Novakovic, and Patricia McCabe. The impact of nasalance on cepstral peak prominence and harmonics-to-noise ratio. *The Laryngoscope*, 129(8):E299–E304, 2019.

[93] Victoria S McKenna and Cara E Stepp. The relationship between acoustical and perceptual measures of vocal effort. *The Journal of the Acoustical Society of America*, 144(3):1643–1658, 2018.

[94] Shahidur Rahman and Tetsuya Shimamura. Pitch determination using autocorrelation function in spectral domain. *Proc. Interspeech 2010*, pages 653–656, 2010.

[95] Abdel-rahman Mohamed, Geoffrey Hinton, and Gerald Penn. Understanding how deep belief networks perform acoustic modelling. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4273–4276. IEEE, 2012.

[96] Tara N Sainath, Brian Kingsbury, Abdel-rahman Mohamed, and Bhuvana Ramabhadran. Learning filter banks within a deep neural network framework. In *2013 IEEE workshop on automatic speech recognition and understanding*, pages 297–302. IEEE, 2013.

[97] Zoltán Tüske, Pavel Golik, Ralf Schlüter, and Hermann Ney. Acoustic modeling with deep neural networks using raw time signal for lvcsr. In *Fifteenth annual conference of the international speech communication association*, 2014.

[98] Pegah Ghahremani, Hossein Hadian, Hang Lv, Daniel Povey, and Sanjeev Khudanpur. Acoustic modeling from frequency domain representations of speech. In *Interspeech*, pages 1596–1600, 2018.

[99] Dimitri Palaz, Ronan Collobert, and Mathew Magimai Doss. Estimating phoneme class conditional probabilities from raw speech signal using convolutional neural networks. *Proc. Interspeech 2013*, pages 1766–1770, 2013.

[100] Aditay Tripathi, Aanchan Mohan, Saket Anand, and Maneesh Singh. Adversarial learning of raw speech features for domain invariant speech recognition. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5959–5963. IEEE, 2018.

[101] Younggwan Kim, Myungjong Kim, Jahyun Goo, and Hoirin Kim. Learning self-informed feature contribution for deep learning-based acoustic modeling. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 26(11):2204–2214, 2018.

[102] Jinxi Guo, Ning Xu, Xin Chen, Yang Shi, Kaiyuan Xu, and Abeer Alwan. Filter sampling and combination cnn (fsc-cnn): A compact cnn model for small-footprint asr acoustic modeling using raw waveforms. In *INTERSPEECH*, pages 3713–3717, 2018.

[103] Tobias Menne, Zoltan Tüske, Ralf Schlüter, and Hermann Ney. *Learning acoustic features from the raw waveform for automatic speech recognition.* Universitätsbibliothek der RWTH Aachen, 2018.

[104] Patrick von Platen, Chao Zhang, and Philip Woodland. Multi-span acoustic modelling using raw waveform signals. *Proc. Interspeech 2019*, pages 1393–1397, 2019.

[105] Sina Alisamir, Seyed Mohammad Ahadi, and Sanaz Seyedin. An end-to-end deep learning model to recognize farsi speech from raw input. In *2018 4th Iranian Conference on Signal Processing and Intelligent Systems (ICSPIS)*, pages 1–5. IEEE, 2018.

[106] Ryu Takeda, Kazuhiro Nakadai, and Kazunori Komatani. Multi-timescale feature-extraction architecture of deep neural networks for acoustic model training from raw speech signal. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2503–2510. IEEE, 2018.

[107] S Pavankumar Dubagunta, Selen Hande Kabil, and Mathew Magimai Doss. Improving children speech recognition through feature learning from raw speech signal. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5736–5740. IEEE, 2019.

[108] Neil Zeghidour, Nicolas Usunier, Gabriel Synnaeve, Ronan Collobert, and Emmanuel Dupoux. End-to-end speech recognition from the raw waveform. *Proc. Interspeech 2018*, pages 781–785, 2018.

[109] Neil Zeghidour, Nicolas Usunier, Iasonas Kokkinos, Thomas Schaiz, Gabriel Synnaeve, and Emmanuel Dupoux. Learning filterbanks from raw speech for phone

recognition. In *2018 IEEE international conference on acoustics, speech and signal Processing (ICASSP)*, pages 5509–5513. IEEE, 2018.

[110] Hiroshi Seki, Kazumasa Yamamoto, and Seiichi Nakagawa. A deep neural network integrated with filterbank learning for speech recognition. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5480–5484. IEEE, 2017.

[111] Hardik B Sailor and Hemant A Patil. Novel unsupervised auditory filterbank learning using convolutional rbm for speech recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 24(12):2341–2353, 2016.

[112] Zhenyao Zhu, Jesse H Engel, and Awni Hannun. Learning multiscale features directly from waveforms. *Interspeech 2016*, pages 1305–1309, 2016.

[113] Mirco Ravanelli and Yoshua Bengio. Speaker recognition from raw waveform with sincnet. In *2018 IEEE Spoken Language Technology Workshop (SLT)*, pages 1021–1028. IEEE, 2018.

[114] Heinrich Dinkel, Nanxin Chen, Yanmin Qian, and Kai Yu. End-to-end spoofing detection with raw waveform cldnns. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4860–4864. IEEE, 2017.

[115] Hannah Muckenhirn, Mathew Magimai-Doss, and Sébastien Marcel. End-to-end convolutional neural network-based voice presentation attack detection. In *2017 IEEE international joint conference on biometrics (IJCB)*, pages 335–341. IEEE, 2017.

[116] Hong Yu, Zheng-Hua Tan, Yiming Zhang, Zhanyu Ma, and Jun Guo. Dnn filter bank cepstral coefficients for spoofing detection. *Ieee Access*, 5:4779–4787, 2017.

[117] Hannah Muckenhirn, Mathew Magimai-Doss, and Sébastien Marcel. On learning vocal tract system related speaker discriminative information from raw signal using cnns. In *INTERSPEECH*, pages 1116–1120, 2018.

[118] Siddique Latif, Rajib Rana, Sara Khalifa, Raja Jurdak, and Julien Epps. Direct modelling of speech emotion from raw speech. *Proc. Interspeech 2019*, pages 3920–3924, 2019.

[119] Panagiotis Tzirakis, Jiehao Zhang, and Bjorn W Schuller. End-to-end speech emotion recognition using deep neural networks. In *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 5089–5093. IEEE, 2018.

[120] Juliette Millet and Neil Zeghidour. Learning to detect dysarthria from raw speech. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5831–5835. IEEE, 2019.

[121] Sander Dieleman and Benjamin Schrauwen. End-to-end learning for music audio. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6964–6968. IEEE, 2014.

[122] Jordi Pons, Oriol Nieto, Matthew Prockup, Erik Schmidt, Andreas Ehmann, and Xavier Serra. End-to-end learning for music audio tagging at scale. In *19th International Society for Music Information Retrieval Conference (ISMIR 2018)*, pages 637–644. ISMIR, 2018.

[123] Ning Chen and Shijun Wang. High-level music descriptor extraction algorithm based on combination of multi-channel cnns and lstm. In *18th International Society for Music Information Retrieval Conference (ISMIR 2017)*, pages 509–514. ISMIR, 2017.

[124] Jordi Pons, Thomas Lidy, and Xavier Serra. Experimenting with musically motivated convolutional neural networks. In *2016 14th international workshop on content-based multimedia indexing (CBMI)*, pages 1–6. IEEE, 2016.

[125] Tara N Sainath and Carolina Parada. Convolutional neural networks for small-footprint keyword spotting. In *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[126] Mario A García, Eduardo A Destéfanis, and Ana L Rosset. Trainable windowing coefficients in dnn for raw audio classification. In *Conference on Cloud Computing, Big Data & Emerging Topics*, pages 153–166. Springer, 2020.

[127] Fredric J Harris. On the use of windows for harmonic analysis with the discrete fourier transform. *Proceedings of the IEEE*, 66(1):51–83, 1978.

[128] Huy Phan, Lars Hertel, Marco Maass, and Alfred Mertins. Robust audio event recognition with 1-max pooling convolutional neural networks. *Interspeech 2016*, pages 3653–3657, 2016.

[129] Ryu Takeda and Kazunori Komatani. Sound source localization based on deep neural networks with directional activate function exploiting phase information. In *2016 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 405–409. IEEE, 2016.

[130] Juan A Morales-Cordovilla, Victoria Sánchez, Angel M Gómez, and Antonio M Peinado. On the use of asymmetric windows for robust speech recognition. *Circuits, Systems, and Signal Processing*, 31(2):727–736, 2012.

[131] Md Jahangir Alam, Patrick Kenny, and Douglas O'Shaughnessy. On the use of asymmetric-shaped tapers for speaker verification using i-vectors. In *Odyssey 2012-The Speaker and Language Recognition Workshop*, 2012.

[132] Robert Rozman and Dušan M Kodek. Using asymmetric windows in automatic speech recognition. *Speech communication*, 49(4):268–276, 2007.

[133] Md Sahidullah and Goutam Saha. A novel windowing technique for efficient computation of mfcc for speaker recognition. *IEEE signal processing letters*, 20(2):149–152, 2012.

[134] Leonardo Wanderley Lopes, Layssa Batista Simões, Jocélio Delfino da Silva, Deyverson da Silva Evangelista, Ana Celiane da Nóbrega e Ugulino, Priscila Oliveira Costa Silva, and Vinícius Jefferson Dias Vieira. Accuracy of acoustic analysis measurements in the evaluation of patients with different laryngeal diagnoses. *Journal of Voice*, 31(3):382–e15, 2017.

[135] Giovanni Saggio and Giovanni Costantini. Worldwide healthy adult voice baseline parameters: a comprehensive review. *Journal of Voice*, 2020.

[136] Mehdi Jafari, James A Till, and Cindy B Law-Till. Interactive effects of local smoothing window size and fundamental frequency on shimmer calculation. *Journal of Voice*, 7(3):235–241, 1993.

[137] Roger Gómez Nieto, Jorge Iván Marín-Hurtado, Luis Miguel Capacho-Valbuena, Alexander Amaya Suarez, and Elkyn Alexander Belalcazar Bolaños. Pattern recognition of hypernasality in voice of patients with cleft and lip palate. In *2014 XIX Symposium on Image, Signal Processing and Artificial Vision*, pages 1–5. IEEE, 2014.

[138] K Uma Rani and Mallikarjun S Holi. A hybrid model for neurological disordered voice classification using time and frequency domain features. *Artif. Intell. Research*, 5(1):87–94, 2016.

[139] James Hillenbrand. Perception of aperiodicities in synthetically generated voices. *The Journal of the Acoustical Society of America*, 83(6):2361–2371, 1988.

[140] Xi Li, Jidong Tao, Michael T Johnson, Joseph Soltis, Anne Savage, Kirsten M Leong, and John D Newman. Stress and emotion classification using jitter and shimmer features. In *2007 IEEE International Conference on Acoustics, Speech and Signal Processing-ICASSP'07*, volume 4, pages IV–1081. IEEE, 2007.

[141] Margarita Kotti and Yannis Stylianou. Effective emotion recognition in movie audio tracks. In *2017 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 5120–5124. IEEE, 2017.

[142] Agnes Jacob. Speech emotion recognition based on minimal voice quality features. In *2016 International Conference on Communication and Signal Processing (ICCSP)*, pages 0886–0890. IEEE, 2016.

[143] Savita Sondhi, Ritu Vijay, Munna Khan, and Ashok K Salhan. Voice analysis for detection of deception. In *2016 11th International Conference on Knowledge, Information and Creativity Support Systems (KICSS)*, pages 1–6. IEEE, 2016.

[144] Hemanta Kumar Palo, Mihir Narayan Mohanty, and Mahesh Chandra. Sad state analysis of speech signals using different clustering algorithm. In *2016 2nd International Conference on Next Generation Computing Technologies (NGCT)*, pages 714–718. IEEE, 2016.

[145] Björn Schuller, Stefan Steidl, and Anton Batliner. The interspeech 2009 emotion challenge. In *Tenth Annual Conference of the International Speech Communication Association*, 2009.

[146] Hye-Jin Kim, Kyungsuk Bae, and Ho-Sub Yoon. Age and gender classification for a home-robot service. In *RO-MAN 2007-The 16th IEEE International Symposium on Robot and Human Interactive Communication*, pages 122–126. IEEE, 2007.

[147] João Paulo Teixeira and Paula Odete Fernandes. Jitter, shimmer and hnr classification within gender, tones and vowels in healthy voices. *Procedia technology*, 16:1228–1237, 2014.

[148] Annel Gómez-Coello, Victor Manuel Valadez-Jiménez, Bulmaro Cisneros, Paul Carrillo-Mora, Martha Parra-Cárdenas, Oscar Hernández-Hernández, and Jonathan J Magaña. Voice alterations in patients with spinocerebellar ataxia type 7 (sca7): clinical-genetic correlations. *Journal of Voice*, 31(1):123–e1, 2017.

[149] Margarita Kotti and Fabio Paternò. Speaker-independent emotion recognition exploiting a psychologically-inspired binary cascade classification schema. *International journal of speech technology*, 15(2):131–150, 2012.

[150] Rosiane Yamasaki, Arlindo Montagnoli, Emi Z Murano, Eloisa Gebrim, Adriana Hachiya, Jorge Vicente Lopes da Silva, Mara Behlau, and Domingos Tsuji. Perturbation measurements on the degree of naturalness of synthesized vowels. *Journal of Voice*, 31(3):389–e1, 2017.

[151] João Paulo Teixeira and André Gonçalves. Algorithm for jitter and shimmer measurement in pathologic voices. *Procedia Computer Science*, 100:271–279, 2016.

[152] Celia Shahnaz, W-P Zhu, and M Omair Ahmad. A new technique for the estimation of jitter and shimmer of voiced speech signal. In *2006 Canadian Conference on Electrical and Computer Engineering*, pages 2112–2115. IEEE, 2006.

[153] Bin Dong. Characterizing resonant component in speech: A different view of tracking fundamental frequency. *Mechanical Systems and Signal Processing*, 88:318–333, 2017.

[154] Bin Liu, Jianhua Tao, Dawei Zhang, and Yibin Zheng. A novel pitch extraction based on jointly trained deep blstm recurrent neural networks with bottleneck features. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 336–340. IEEE, 2017.

[155] Fritz Klingholz and Frank Martin. Quantitative spectral evaluation of shimmer and jitter. *Journal of Speech, Language, and Hearing Research*, 28(2):169–174, 1985.

[156] Mario A García and Eduardo A Destéfanis. Deep neural networks for shimmer approximation in synthesized audio signal. In *Communications in Computer and Information Science*, pages 3–12. Springer International Publishing, 2018.

[157] Jack J Jiang, Yu Zhang, Julia MacCallum, Alicia Sprecher, and Liang Zhou. Objective acoustic analysis of pathological voices from patients with vocal nodules and polyps. *Folia Phoniatrica et Logopaedica*, 61(6):342–349, 2009.

[158] Marcelo de Oliveira Rosa, José Carlos Pereira, and Marcos Grellet. Adaptive estimation of residue signal for voice pathology diagnosis. *IEEE Transactions on Biomedical Engineering*, 47(1):96–104, 2000.

[159] Paulo Rogério Scalassara, María Eugenia Dajer, Carlos Dias Maciel, Rodrigo Capobianco Guido, and José Carlos Pereira. Relative entropy measures applied to healthy and pathological voice characterization. *Applied Mathematics and Computation*, 207(1):95–108, 2009.

[160] Mohamed Fezari, Fethi Amara, and Ibrahim MM El-Emary. Acoustic analysis for detection of voice disorders using adaptive features and classifiers. In *Proc. Int. Conf. Circuits, Syst. Control*, pages 112–117, 2014.

[161] Everthon S Fonseca and Jose C Pereira. Normal versus pathological voice signals. *IEEE Engineering in Medicine and Biology Magazine*, 28(5):44–48, 2009.

[162] Yingyong Qi and Robert E Hillman. Temporal and spectral estimations of harmonics-to-noise ratio in human voice signals. *The Journal of the Acoustical Society of America*, 102(1):537–543, 1997.

[163] Guus de Krom. A cepstrum-based technique for determining a harmonics-to-noise ratio in speech signals. *Journal of Speech, Language, and Hearing Research*, 36(2):254–266, 1993.

[164] Peter Murphy and Olatunji Akande. Cepstrum-based harmonics-to-noise ratio measurement in voiced speech. In *Nonlinear Speech Modeling and Applications*, pages 199–218. Springer, 2005.

[165] Rita R Patel, Shaheen N Awan, Julie Barkmeier-Kraemer, Mark Courey, Dimitar Deliyski, Tanya Eadie, Diane Paul, Jan G Švec, and Robert Hillman. Recommended protocols for instrumental assessment of voice: American speech-language-hearing association expert panel to develop a protocol for instrumental assessment of vocal function. *American journal of speech-language pathology*, 27(3):887–905, 2018.

[166] Arezoo Hasanvand, Abolfazl Salehi, and Mona Ebrahimipour. A cepstral analysis of normal and pathologic voice qualities in iranian adults: a comparative study. *Journal of Voice*, 31(4):508–e17, 2017.

[167] Cara Sauder, Michelle Bretl, and Tanya Eadie. Predicting voice disorder status from smoothed measures of cepstral peak prominence using praat and analysis of dysphonia in speech and voice (adsv). *Journal of Voice*, 31(5):557–566, 2017.

[168] Ketaki Vasant Phadke, Anne-Maria Laukkanen, Irma Ilomäki, Elina Kankare, Ahmed Geneid, and Jan G Švec. Cepstral and perceptual investigations in female teachers with functionally healthy voice. *Journal of voice*, 34(3):485–e33, 2020.

[169] Peter J Murphy. On first rahmonic amplitude in the analysis of synthesized aperiodic voice signals. *The Journal of the Acoustical Society of America*, 120(5):2896–2907, 2006.

[170] Robin A Samlan, Brad H Story, and Kate Bunton. Relation of perceived breathiness to laryngeal kinematics and acoustic measures based on computational modeling. *Journal of Speech, Language, and Hearing Research*, pages 1209–1223, 2013.

[171] Rubén Fraile and Juan Ignacio Godino-Llorente. Cepstral peak prominence: A comprehensive analysis. *Biomedical Signal Processing and Control*, 14:42–54, 2014.

[172] James Hillenbrand and Robert A Houde. Acoustic correlates of breathy vocal quality: dysphonic voices and continuous speech. *Journal of Speech, Language, and Hearing Research*, 39(2):311–321, 1996.

[173] Massachusetts Eye and Ear Infirmary. Voice disorders database, version. 1.03 (cd-rom). *Lincoln Park, NJ: Kay Elemetrics Corporation*, 1994.

[174] Taishih Chi and Shihab A Shamma. Spectrum restoration from multiscale auditory phase singularities by generalized projections. *IEEE transactions on audio, speech, and language processing*, 14(4):1179–1192, 2006.

[175] Nima Mesgarani, Malcolm Slaney, and Shihab A Shamma. Discrimination of speech from nonspeech based on multiscale spectro-temporal modulations. *IEEE Transactions on Audio, Speech, and Language Processing*, 14(3):920–930, 2006.

[176] Richard Colling, Helen Pitman, Karin Oien, Nasir Rajpoot, Philip Macklin, CM-Path AI in Histopathology Working Group, Velicia Bachtiar, Richard Booth, Alyson Bryant, Joshua Bull, et al. Artificial intelligence in digital pathology: a roadmap to routine use in clinical practice. *The Journal of pathology*, 249(2):143–150, 2019.

[177] Balázs Acs, Mattias Rantalainen, and Johan Hartman. Artificial intelligence as the next step towards precision pathology. *Journal of internal medicine*, 288(1):62–81, 2020.

[178] Annika Reinke, Matthias Eisenmann, Minu D Tizabi, Carole H Sudre, Tim Rädsch, Michela Antonelli, Tal Arbel, Spyridon Bakas, M Jorge Cardoso, Veronika Cheplygina, et al. Common limitations of image processing metrics: A picture story. *arXiv preprint arXiv:2104.05642*, 2021.

[179] Jeroen van der Laak, Geert Litjens, and Francesco Ciompi. Deep learning in histopathology: the path to the clinic. *Nature medicine*, 27(5):775–784, 2021.

[180] Hiroshi Yoshida and Tomoharu Kiyuna. Requirements for implementation of artificial intelligence in the practice of gastrointestinal pathology. *World Journal of Gastroenterology*, 27(21):2818, 2021.

[181] Soma Kobayashi, Joel H Saltz, and Vincent W Yang. State of machine and deep learning in histopathological applications in digestive diseases. *World Journal of Gastroenterology*, 27(20):2545, 2021.

[182] Jakob Nikolas Kather, Alexander T Pearson, Niels Halama, Dirk Jäger, Jeremias Krause, Sven H Loosen, Alexander Marx, Peter Boor, Frank Tacke, Ulf Peter Neumann, et al. Deep learning can predict microsatellite instability directly from histology in gastrointestinal cancer. *Nature medicine*, 25(7):1054–1056, 2019.