UNIVERSITAT POLITÈCNICA DE VALÈNCIA

_ TELECOM ESCUELA TÉCNICA VLC SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN

CLASIFICACIÓN ACÚSTICA DE SALAS MEDIANTE ALGORITMOS DE MACHINE LEARNING

Juan Fraga Domingo

Tutora: Gema Piñero Sipán

Cotutora: María de Diego Antón

Trabajo Fin de Grado presentado en la Escuela Técnica Superior de Ingenieros de Telecomunicación de la Universitat Politècnica de València, para la obtención del Título de Graduado en Ingeniería de Tecnologías y Servicios de Telecomunicación

Curso 2018-19

Valencia, 7 de septiembre de 2019

Escuela Técnica Superior de Ingeniería de Telecomunicación
Universitat Politècnica de València
Edificio 4D. Camino de Vera, s/n, 46022 Valencia
Tel. +34 96 387 71 90, ext. 77190
www.etsit.upv.es

VLC/ CAMPUS
VALENCIA, INTERNATIONAL CAMPUS OF EXCELLENCE

EMAS
Gestión ambiental verificada
REG.NO. ES-CV- 000030

Resumen

El presente TFG estudia la posibilidad de estimar parámetros acústicos de un audio grabado bajo determinadas condiciones de reverberación utilizando una red neuronal artificial conocida como perceptrón multicapa. El perceptrón multicapa es un algoritmo de aprendizaje máquina (*machine learning*, ML) conocido por sus siglas inglesas MLP (*multilayer perceptron*).

Para ello, se genera en primer lugar una amplia base de datos con audios que corresponden a diferentes valores de los parámetros a estimar. Esto se consigue realizando la convolución de audios de voz grabados en condiciones libres de reverberación con respuestas al impulso artificiales que simulan la reverberación de distintas salas.

De forma previa a entrenar el perceptrón, se realiza un procesado de los audios para extraer las características de entrada al algoritmo. En este sentido, se comparan los resultados de clasificación correspondientes a dos tipos de procesados diferentes.

Además, se prueban también múltiples configuraciones del perceptrón, cambiando la función de activación, el número de capas ocultas, el número de neuronas en las capas ocultas, el orden de introducción de los datos y algunas otras variables del entrenamiento; todo ello con el objetivo de optimizar la precisión de los resultados.

Resum

El present TFG estudia la possibilitat d'estimar paràmetres acústics d'un àudio gravat en determinades condicions de reverberació utilitzant una xarxa neuronal artificial coneguda com perceptró multicapa. El perceptró multicapa és un algorisme d'aprenentatge màquina (*machine learning*, ML) conegut per les seues sigles angleses MLP (*multilayer perceptron*).

Per a això, es genera en primer lloc una àmplia base de dades amb àudios que simulen correspondre a diferents valors dels paràmetres a estimar. Això s'aconsegueix realitzant la convolució d'àudios de veu gravats en condicions lliures de reverberació amb respostes a l'impuls artificials que simulen la reverberació de diferents habitacions.

Abans d´entrenar el perceptró, es realitza un processament dels àudios per a extraure les característiques d'entrada a l'algorisme. En aquest sentit, es comparen els resultats de classificació corresponents a dos tipus de processaments diferents.

A més, es proven també múltiples configuracions del perceptró, canviant la funció d'activació, el nombre de capes ocultes, el nombre de neurones en les capes ocultes, l'ordre d'introducció de les dades i algunes altres variables de l'entrenament; tot això amb l'objectiu d'optimitzar la precisió dels resultats.

Abstract

This Bachelor´s Thesis evaluates the possibility of estimating acoustic parameters from an audio recorded under certain reverberation conditions using an artificial neural network known as Multilayer Perceptron. The Multilayer Perceptron is a machine learning algorithm also referred to as MLP.

For that purpose, an extensive database is generated with audio that belongs to different values of the desired parameters. This is achieved by calculating the convolution of speech audio recorded under reverberation-free conditions with room impulse responses that simulate the reverberation of different rooms.

Prior to training the perceptron, the audio is processed in order to extract the input features of the algorithm. Two types of processing are applied and its classification results are compared.

In addition, multiple configurations are tested for the perceptron, changing the activation function, the number of hidden layers, the number of neurons in hidden layers, the order in which the data are introduced and some other training variables; with the objective of optimizing the accuracy of the results.

Índice

1. Introducción

El presente TFG estudia la posibilidad de estimar dos parámetros ampliamente utilizados para caracterizar la reverberación en una sala, como son el Tiempo de Reverberación (en inglés *Reverberation Time*, **RT**) y la ratio entre la energía presente en un primer intervalo de tiempo frente a la presente tras este primer intervalo (en inglés *Early to Late Ratio*, **ELR**); a partir de audios de voz.

Estos parámetros pueden ser calculados midiendo la respuesta al impulso de una sala entre el emisor y el receptor del sonido; si bien, ello requeriría conocer la sala en la que han sido grabados los audios, la posición del emisor y del receptor, así como realizar el estudio previo de su respuesta al impulso.

En este trabajo, se plantea la posibilidad de estimar estos valores 'a ciegas', a partir de audios de voz que hubieran sido grabados en una sala, sin tener que realizar mediciones *in situ* de la sala previamente. Para ello, se generará una amplia base de datos con la que entrenar una red neuronal artificial conocida como Perceptrón Multicapa, basada en aprendizaje supervisado.

La motivación para conseguir este tipo de mediciones 'a ciegas' radica en la posibilidad de que puedan ser posteriormente perfeccionadas e implementadas en aplicaciones acústicas y de telecomunicaciones en las que la reverberación provoca un deterioro en las prestaciones y en las que no sea práctico o no sea posible realizar el cálculo de la respuesta al impulso con anterioridad. Ejemplos de esto serían desde aplicaciones en tiempo real de reconocimiento de voz hasta aplicaciones forenses donde se disponga de una grabación y se necesite investigar en qué tipo de sala podría haberse realizado.

Así, poder estimar el RT y el ELR a partir de un audio, sin necesidad de conocer *a priori* la sala o la posición donde ha sido grabado, resulta de gran interés para este tipo de situaciones.

2. Modelo de reverberación de una sala

La reverberación acústica se define como la persistencia de un sonido una vez deja de actuar la fuente que lo originaba [1].

Como menciona Naylor en [2], los dos efectos principales de la reverberación son los siguientes:

1. Efecto caja: el audio grabado en una sala se escucha como si se emitiera desde varias posiciones, escuchándose por tanto con diferentes retardos e intensidades.
2. Efecto de lejanía: el audio se percibe como si hubiera sido grabado a demasiada distancia del micrófono, cuando no es así.

Estos efectos no son necesariamente negativos si están presentes de forma moderada y controlada. Si bien, cuando aparecen en exceso, complican la inteligibilidad del habla en los audios, lo que genera problemas en aplicaciones de procesado de señal como el reconocimiento de voz.

La respuesta al impulso de una sala (en inglés *Room Impulse Response*, RIR) es una función que caracteriza la propagación acústica del sonido entre dos puntos en un recinto cerrado y, por tanto, permite recuperar la señal de sonido original emitida por la fuente a partir de la señal recibida en otro punto de la sala. Esta señal recibida será diferente de la emitida debido a la reverberación que provoca la sala. Realizando la operación de convolución inversa de la señal recibida con la RIR se obtendría la señal original.

Como en la mayoría de los problemas reales no se conoce la RIR de la sala, resulta de gran utilidad estimar dos parámetros (RT y ELR) que se podrían calcular a partir de la RIR, en caso de conocerse, y que se emplean en diferentes técnicas de control de la reverberación. Algunos ejemplos de estas son la propuesta por Habets [3] o la propuesta por Gaubitch y Naylor [4].

2.1 Tiempo de reverberación (RT)

El tiempo de reverberación es una característica relevante de un espacio acústico ampliamente utilizada. Para definirla, del mismo modo que hace Naylor [2], es necesario aludir primero al concepto de *Energy Decay Curve* (EDC).

La forma más intuitiva de entender lo que representa la EDC es atendiendo a cómo se calcula; lo cual se detalla en los siguientes pasos:

1. Se excita una sala con una señal acústica de banda ancha hasta conseguir una distribución de energía sonora estable y uniforme.
2. Se corta la emisión de la señal acústica.
3. Una vez se ha dejado de emitir señal, se registra la caída del cuadrado de la presión acústica en el tiempo.

La EDC es precisamente la gráfica que recoge esa caída del cuadrado de la presión acústica frente al tiempo.

Retomando el concepto de tiempo de reverberación, este se define como el tiempo en segundos en el que la EDC cae 60 dB.

2.2 *Early to Late Ratio* (ELR)

El ELR es la ratio entre la energía recibida en un primer intervalo de tiempo desde que se corta la emisión de sonido, y el resto de energía recibida posteriormente. En este trabajo, se tomará un primer intervalo de 50 ms que incluirá energía correspondiente al camino directo así como parte de la reverberante.

La motivación para escoger este tiempo radica en la naturaleza del oído humano. Este interpreta las componentes de una señal multicamino como si formaran una única señal si la diferencia en el tiempo de recepción de las mismas es menor a 50 ms, aproximadamente.

Así, con este tiempo, tal y como detalla Naylor [2], el ELR proporciona una medida de la ratio entre el nivel de señal que el oído humano asocia al camino directo, frente a la señal identificada como reverberante; a pesar de que parte de la señal asociada al camino directo será también reverberante.

El ELR se puede calcular a partir de la RIR (denotada como *h* en la ecuación 2.1):

$$ELR = 10 \cdot log_{10}\left(\frac{\int_{t=0}^{t=50\text{ ms}} h^2(t)}{\int_{t=50\text{ ms}}^{\infty} h^2(t)}\right) \quad \textbf{(2.1)}$$

3. Aprendizaje automático

El aprendizaje automático consiste en que una computadora sea capaz de aprender a partir de unos datos, con el fin de generar predicciones y estimaciones.

Tradicionalmente, la forma de generar algoritmos que interpretaran datos era la siguiente:

1. Una persona escribía el algoritmo y se lo transmitía a la máquina en forma de código.
2. La máquina se limitaba a leer unos datos y ejecutar el algoritmo.
3. De esta forma, se generaban las predicciones.

El aprendizaje automático otorga mayor relevancia e inteligencia a la máquina. En este, no es la persona quien genera el algoritmo que formulará las predicciones, sino que es la máquina la que lo genera y entrena en función de los datos de los que dispone, creando y optimizando sus predicciones.

Aun así, el ser humano sigue siendo fundamental a la hora de delimitar las características del entrenamiento, la selección y procesado de los datos, así como la interpretación de las predicciones, como se verá en la parte experimental.

Los dos tipos de aprendizaje principales son:

- Aprendizaje supervisado.
- Aprendizaje no supervisado.

3.1 Aprendizaje supervisado

El aprendizaje supervisado es el empleado para estimar los parámetros objeto de estudio en este trabajo. Citando a Mello [5], "consiste en encontrar el mejor clasificador posible $f: X \rightarrow Y$ para un problema dado. Al algoritmo responsable de realizar este mapeo se le conoce como algoritmo de clasificación."

En este tipo de aprendizaje, será necesaria una base de datos de entrenamiento. Esta deberá incluir los datos de entrada y las variables de salida de forma que el algoritmo pueda entrenarse hasta generar unas predicciones lo más precisas posibles.

Para verificar el comportamiento del algoritmo, será necesaria otra base de datos, la de pruebas. Esta también contendrá datos de entrada con sus correspondientes variables de salida. La diferencia fundamental con la base de datos de entrenamiento es que contiene datos nuevos, que no han sido utilizados para entrenar el algoritmo. Así, se introducirán los datos de entrada al algoritmo para que sea este quien genere las variables de salida. Tras ello, se podrán comparar las predicciones del algoritmo con las variables de salida correctas.

¿Por qué no realizar las pruebas directamente con la base de datos de entrenamiento?

Esto no es recomendable porque se puede dar la situación de que el algoritmo genere predicciones muy buenas para esta base de datos, pero muy malas cuando se introducen datos nuevos. Este fenómeno es el conocido como sobre-entrenamiento, en el que el algoritmo es capaz de memorizar los ejemplos que ha recibido para entrenarse pero es incapaz de generalizar ante datos nuevos.

Como se detallará en la parte experimental, se ha empleado un tercer tipo de base de datos para evitar el sobre-entrenamiento: la de validación.

3.2 Aprendizaje no supervisado

La diferencia fundamental con el supervisado es que el algoritmo no entrena asociando unos datos de entrada a unas salidas, puesto que no posee las salidas. El aprendizaje no supervisado consiste por tanto en construir modelos en base a las características inherentes de los datos de entrada [6].

Algunas aplicaciones de este tipo de entrenamiento son la de agrupar los datos de entrada que comparten ciertas similitudes o la de encontrar datos anómalos, entre otras.

No se emplea este tipo de aprendizaje en el presente trabajo.

4. Perceptrón multicapa

El perceptrón multicapa (en inglés *Multilayer Perceptron*, MLP) es el tipo de red neuronal artificial empleada.

Está formado por, al menos, tres capas, de forma que se asemeja a la estructura de neuronas interconectadas del cerebro humano [7]:

1. Capa de entrada.
2. Capa(s) oculta(s): puede haber una o más.
3. Capa de salida.

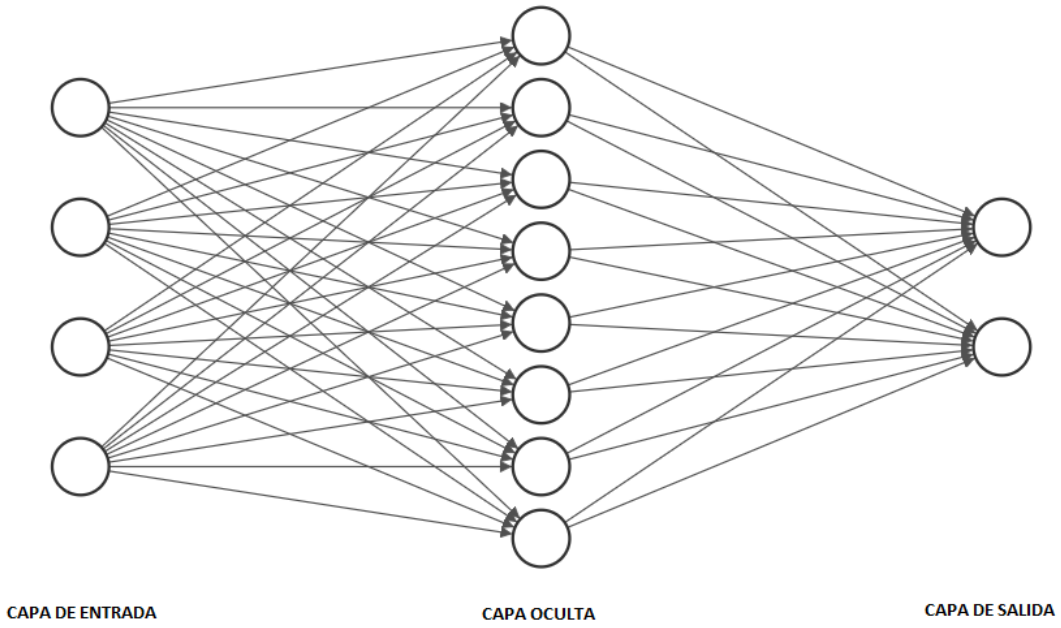

Figura 1: esquema de un perceptrón multicapa con una capa oculta.

1. La primera capa es la que recibe los datos de entrada a la red neuronal. Como se ve en la figura 1, en este caso la capa de entrada cuenta con cuatro neuronas, representadas con círculos. El número de neuronas en la capa de entrada coincide con el número de variables que se introducen a la red por cada muestra, tanto para entrenarse como posteriormente para realizar las predicciones. Las neuronas de la primera capa tienen un comportamiento simple, se limitan únicamente a transmitir los datos de entrada a la siguiente capa [8].

2. Cada neurona de la primera capa oculta recibe a su entrada las variables de la capa de entrada multiplicadas por distintos pesos. En este caso, la primera neurona recibiría lo siguiente:

$$x_1 \cdot w_{1,1} + x_2 \cdot w_{2,1} + x_3 \cdot w_{3,1} + x_4 \cdot w_{4,1} \quad \textbf{(4.1)}$$

   donde x_n denota la variable n-ésima y $w_{n,m}$ denota el peso por el que se multiplica la variable *n*-ésima para transmitirse como entrada a la *m*-ésima neurona de la primera capa oculta.

Adicionalmente a las variables de entrada, se puede hacer uso también de una constante conocida como sesgo (en inglés *bias*). Esta otorga a cada neurona un valor de entrenamiento constante que ayuda al modelo a realizar mejores ajustes.

Siguiendo con el ejemplo anterior y teniendo en cuenta la constante *bias*, la primera neurona de la capa oculta recibiría lo siguiente:

$$x_1 \cdot w_{1,1} + x_2 \cdot w_{2,1} + x_3 \cdot w_{3,1} + x_4 \cdot w_{4,1} + 1 \cdot w_{bias,1} \quad \textbf{(4.2)}$$

La función que relaciona el valor que cada neurona recibe a su entrada con el valor a su salida se denomina: **función de activación**. Siguiendo con el ejemplo:

- A la entrada de la primera neurona de la capa oculta se tiene el resultado de la fórmula 4.2.
- Siendo f la función de activación, a la salida se tendría:

$$f(x_1 \cdot w_{1,1} + x_2 \cdot w_{2,1} + x_3 \cdot w_{3,1} + x_4 \cdot w_{4,1} + 1 \cdot w_{bias,1}) \quad \textbf{(4.3)}$$

Existen multitud de funciones de activación, como se comentará más adelante.

En caso de existir más capas ocultas, el comportamiento sería similar. Las neuronas de la capa oculta *n*-ésima generarían salidas que se multiplicarían por pesos para formar la entrada a las neuronas de la capa (*n*+1)-ésima; con su consecuente función de activación y salida.

3. Las salidas de la última capa oculta multiplicadas por sus respectivos pesos generan la entrada a las neuronas de la capa de salida. Como argumenta Brownlee en [8], la capa de salida tiene la particularidad de que tiene que conseguir ajustar sus entradas al formato de salida requerido en el problema, por lo que su función de activación tendrá esta restricción. Así, pueden darse los siguientes casos:
   a. Problema de regresión: sólo habría una neurona de salida que podría no tener función de activación.
   b. Problema de clasificación binaria: se podría emplear también una única neurona con una función de activación que delimitara la probabilidad de tener la clase A o la clase B. Por ejemplo, si la salida está en el intervalo [0 - 0,5) → clase A; mientras que si está en el intervalo complementario hasta 1 → clase B.
   c. Problema de clasificación con dos o más clases: este tipo de problema será el que se desarrolla en el presente trabajo. Lo común para este caso es tener tantas neuronas en la capa de salida como número de clases a clasificar. De esta forma, se utiliza un tipo de función de activación que otorga la probabilidad de que la predicción corresponda a cada determinada clase.

Siguiendo con el ejemplo, se observa en la gráfica de la figura 1 que existen dos neuronas en la capa de salida. Al ser solo dos clases, se podría haber utilizado también una única neurona como se ha descrito en el problema de clasificación binaria. Supongamos que la neurona primera denota la probabilidad de que la muestra pertenezca a la clase A mientras que la segunda a la clase B.

- La salida de la neurona primera sería:

$$f(valor\ de\ entrada\ de\ la\ primera\ neurona) = 0{,}8; \text{ por ejemplo.}$$

- La salida de la segunda neurona sería:

$$f(valor\ de\ entrada\ de\ la\ segunda\ neurona) = 1 - 0.8 = 0{,}2$$

Se concluiría que la muestra pertenece a la clase A puesto que la probabilidad es del 80% frente al 20% de que pertenezca a la B.

4.1 Funciones de activación

Como se ha comentado, la función de activación f de una neurona genera su salida Y en base a su entrada x: $Y = f(x)$. Esta función puede ser lineal o no. Como explica Karn [9], el propósito principal de la función no lineal es que introduce no linealidad a la salida de la neurona; lo cual es muy útil teniendo en cuenta que la mayoría de problemas del mundo real no son lineales. Las funciones de activación que se emplean en este trabajo son:

- Sigmoide: esta se ha utilizado en la capa oculta. Genera salidas entre cero y uno; su forma y fórmula se muestra a continuación:

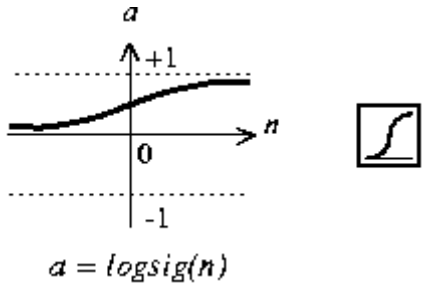

Figura 2: función de activación sigmoide [10].

- Unidades lineales rectificadas: esta se utiliza en una de las pruebas preliminares en la capa oculta, pero se desestima para los experimentos finales dado su peor rendimiento respecto a la sigmoide. Su forma y expresión son las siguientes:

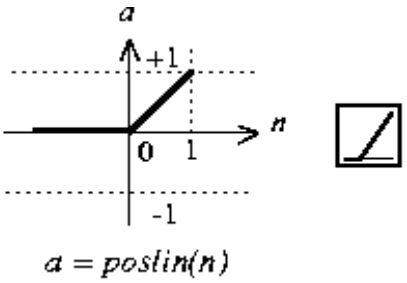

Figura 3: función de activación de unidades lineales rectificadas [11].

- *Softmax*: esta función se emplea habitualmente en la capa de salida para problemas de clasificación, al igual que se ha hecho en este trabajo. Genera la probabilidad de que la salida pertenezca a una u otra clase. Así, el sumatorio de todas las salidas es la unidad.

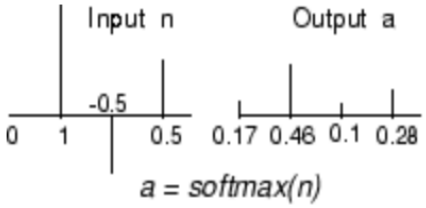

Figura 4: función de activación *softmax* [12].

Otras funciones de activación relevantes, pero no utilizadas en el trabajo son la de la tangente hiperbólica o la lineal.

Hasta ahora, se ha explicado cómo las variables de entrada se multiplican por distintos pesos, atraviesan funciones de activación, se multiplica su salida por más pesos y atraviesan otras neuronas pero, ¿qué valor han de tener los pesos? Esto es precisamente lo que tiene que determinar la máquina cuando entrena el algoritmo.

4.2 Entrenamiento de la red neuronal

Al iniciar el entrenamiento los pesos tienen un determinado valor. Si bien existen diferentes técnicas para realizar esta inicialización, se ha seguido la más habitual que es inicializarlos con valores aleatorios. Pero estos valores iniciales no consiguen generar predicciones correctas. Por ello, la red neuronal tiene como objetivo ajustar los pesos hasta conseguir predicciones lo más precisas y generalizables posibles. Para desarrollar este proceso se utiliza un algoritmo de entrenamiento.

Existen multitud de algoritmos de entrenamiento. El clásico es el del descenso por gradiente estocástico. Consiste en lo siguiente [8]:

1. Se introduce una muestra de entrada en la red.
2. La red genera un valor de salida.
3. El valor de salida se compara con el correcto y se calcula el error.
4. El error atraviesa la red de vuelta, actualizando los pesos en función de lo que cada uno haya contribuido al error.
5. Se repiten los pasos anteriores para cada muestra.
6. Una vez se han introducido todas las muestras se puede repetir el proceso.

Otros algoritmos de entrenamiento, disponibles en Matlab, son [13]:

- Levenberg-Marquardt: adecuado para problemas de regresión.
- Retropropagación resiliente: converge rápido en problemas de clasificación, pero sus prestaciones se degradan rápido cuando se reduce el error.
- **Gradiente conjugado escalado**: es el algoritmo de entrenamiento empleado en este trabajo. Se ha escogido en base a la sugerencia de Matlab, con los siguientes argumentos:
    - Consigue un buen desempeño en una amplia variedad de problemas.
    - Converge casi tan rápido en problemas de clasificación como el de retropropagación resiliente, pero tiene la ventaja de no sufrir una degradación tan rápida conforme el error disminuye.
    - Tiene unos requerimientos de memoria relativamente modestos, lo cual es importante teniendo en cuenta que las redes se entrenan en un ordenador comercial de gama media.

  Este algoritmo, cuyo comportamiento detalla Meiller [14], permite entrenar la red calculando la derivada del desempeño o *performance* respecto a los pesos y actualizando los mismos en cada iteración.

Matlab cuenta con varias formas de medir el desempeño o *performance*:

- Realizando la media de los errores absolutos (en inglés *Mean Absolute Error*, MAE).
- Con el error cuadrático medio (en inglés *Mean Squared Error*, MSE).

- Calculando la entropía cruzada [15]: este es el método utilizado en este trabajo para evaluar el desempeño de las redes. Mientras que los dos primeros son preferibles en problemas de regresión, la entropía cruzada es más adecuada para problemas de clasificación, como demuestra McCaffrey [16].
  La entropía cruzada penaliza severamente cuando las clasificaciones son extremadamente imprecisas mientras que penaliza levemente si la predicción se acerca a un resultado correcto. En concreto, siendo y la predicción para una determinada muestra de entrada y t el valor correcto de salida para dicha muestra, se realizan las siguientes operaciones:
  - La entropía cruzada de cada muestra, c_j, se calcula según la siguiente fórmula:

    $$c_j = -t \cdot \log(y) \quad \textbf{(4.4)}$$

  - La entropía cruzada del conjunto de muestras sería ya el desempeño o *performance* de la red. Se calcula como la media de la entropía cruzada de todas las muestras:

    $$performance = \frac{\sum_{j=1}^{J} c_j}{J} \quad \textbf{(4.5)}$$

5. Clasificador de salas según su reverberación

Para construir el clasificador se siguen los siguientes puntos, que se detallarán en los apartados siguientes:

- Generación de los audios.
- Procesado de los audios y extracción de características.
- Entrenamientos preliminares de los perceptrones.
- Entrenamientos definitivos.
- Resultado

El proceso seguido se puede ver representado en el siguiente esquema:

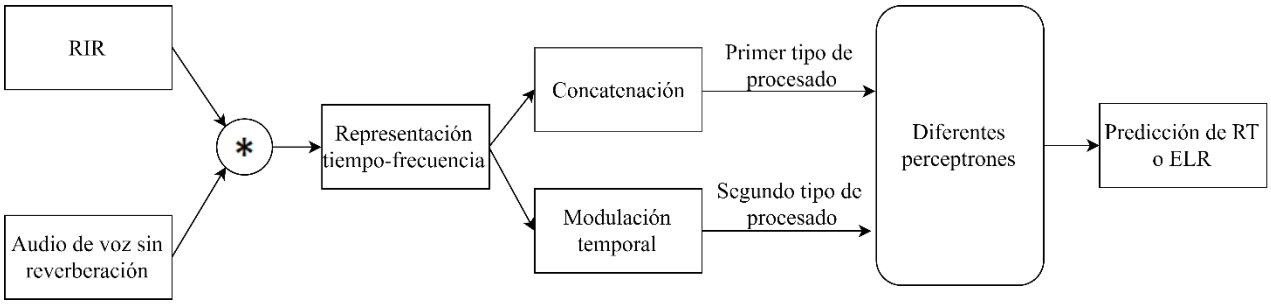

Figura 5: esquema del proceso seguido para crear el clasificador de salas.

6. Generación de las bases de datos

6.1 Obtención de los audios

El objetivo de este apartado es conseguir una gran base de datos para poder entrenar el perceptrón multicapa. Un número reducido de datos impediría que el perceptrón estuviera entrenado de manera correcta para conseguir buenos resultados en heterogeneidad de salas y hablantes. Un número demasiado alto genera problemas de memoria a la hora de entrenar la red. Por tanto, se busca un equilibrio entre un entrenamiento adecuado y una viabilidad para ser implementado en un ordenador comercial de gama media.

A la entrada de cada perceptrón se introducirán diferentes características extraídas de multitud de audios. El objetivo de uno de los perceptrones será determinar el RT, mientras que el otro tendrá que estimar el ELR.

Con esta finalidad, lo primero será generar un conjunto de audios grabados en salas con distintos valores de RT y ELR conocidos. Para ello, y como se detallará a continuación, se han convolucionado audios de voz grabados en una cámara anecoica con respuestas al impulso de distintas salas generadas de manera artificial y valores de RT y ELR determinados.

6.1.1 Audios de voz

Los audios de voz se han extraído de la base de datos TIMIT [17]. Esta contiene frases con riqueza fonética correspondientes a 630 angloparlantes clasificadas en los ocho dialectos principales del inglés americano, grabadas a 16 kHz de frecuencia.

En particular, para la base de datos de este trabajo se han utilizado 80 audios cuyas voces provienen de 4 hombres, dos de ellos con acento clasificado como *New England* y otros dos con acento *Northern*; y 4 mujeres, con la misma distribución de acentos que los hombres. En total se emplean 10 frases por cada uno de los hablantes.

Para facilitar el posterior procesamiento de los audios, se recortan de forma que todos tengan la misma duración, que será la del audio más corto de entre los descargados. En concreto, su duración será de 1,408 segundos, que corresponde a 22528 muestras. También se realiza en Matlab una normalización de sus valores en el intervalo [-1,1].

6.1.2 Respuestas al impulso

Para simular que los audios han sido grabados en salas con unas características predeterminadas, se han generado diferentes respuestas al impulso sintéticas siguiendo el método imagen propuesto por Allen y Berkley [18], utilizando la función de Matlab *rir_generator* creada por Habets [19]. Estas RIRs corresponden a una amplia variedad de salas y posiciones relativas entre el emisor y el receptor que se detallarán más adelante.

Cabe mencionar que a la hora de generar las RIRs con el *rir_generator*, se introduce el RT deseado como variable de entrada. Sin embargo, el ELR no está contemplado como tal. Para determinar el valor de ELR de cada RIR se sigue el método empleado por Xiong et al. [20], que consiste en los siguientes pasos:

1. El silencio que precede al RIR se elimina.
2. Se determina que el ELR objeto de estudio será la ratio entre la energía de los primeros 50 ms y el resto. La motivación para escoger este valor se menciona en el apartado 2.2.
3. Se aplica la fórmula siguiente, con $t_e = 50\,ms$ y donde k_e denota la última muestra incluida en el sumatorio del numerador, que es el producto de t_e por la frecuencia de muestreo ($f_s = 16\,kHz$) :

$$ELR = 10 \cdot log_{10}\left(\frac{\int_{t=0}^{t=t_e} h^2(t)}{\int_{t=t_e}^{\infty} h^2(t)}\right) \backsimeq 10 \cdot log_{10}\left(\frac{\sum_{k=1}^{k=k_e} h^2[k]}{\int_{k=k_e+1}^{L_h} h^2[k]}\right) \quad \textbf{(6.1)}$$

En resumen, para cada RIR, el RT es una variable que se introduce para crearlo y el ELR se calcula siguiendo los pasos mencionados. La importancia de guardar estos valores radica en que serán utilizados posteriormente para vincular cada audio a su clase correspondiente de forma que el algoritmo supervisado del perceptrón pueda entrenarse.

En tanto que los RIRs simulan la respuesta de una sala, otras variables de entrada importantes son las dimensiones de la sala simulada. Para este trabajo, se han simulado nueve salas con las siguientes dimensiones:

- Sala ejemplo: [*x y z*], donde *x* denota el ancho, *y* el largo y *z* la altura; en metros.
- Sala 1: [2 1 3]
- Sala 2: [2 4 3]
- Sala 3: [4 5 4]
- Sala 4: [3 7 2,5]
- Sala 5: [4 9 3]
- Sala 6: [8 10 5]
- Sala 7: [10 10 4]
- Sala 8: [12,5 15 5]
- Sala 9: [16 15 8]

En cada una de las salas, se han generado 5 RIRs diferentes para cada valor de RT entre 0,3 s y 0,68 s, con un paso de 0,02 s; salvo en la Sala 9, donde no se han generado respuestas con valor de RT igual a 0,3 s ya que es imposible generar RIRs con un valor de RT tan pequeño dadas sus dimensiones.

Así, en total, se generan 895 RIRs. En todas ellas, se simula que la posición de la fuente de sonido es el centro de la sala y el receptor se sitúa en posiciones aleatorias dentro de las dimensiones de la misma.

Como se explicará más adelante, los intervalos de interés considerados para el trabajo serán:

- Para RT: [0,3; 0,68] segundos
- Para ELR: [0; 19] decibelios

Lo ideal para entrenar la red neuronal sería conseguir una distribución uniforme de estos valores. Esto se consigue para los valores de RT ya que se controlan a la hora de generar las RIRs. No ocurre así con los valores de ELR. Además, resulta más complicado conseguir muestras en la parte alta del intervalo de ELR.

Con el fin de equilibrar la distribución, se eliminan algunas RIRs cuyos valores de ELR están sobrerrepresentados. Así mismo, se eliminan también aquellas RIRs cuyos valores de ELR están fuera del intervalo objeto de estudio.

Una vez realizados los ajustes, el número total de RIRs es de 687, con la siguiente distribución de RT y ELR:

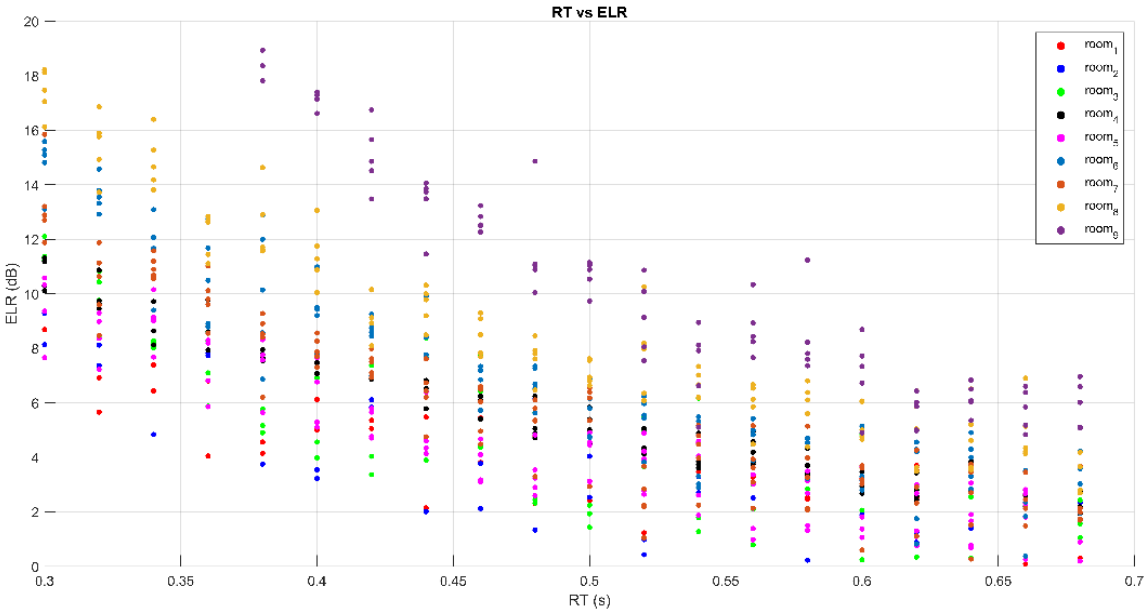

Figura 6: distribución de valores de ELR y RT de las RIRs utilizadas.

Como se observa en la Figura 6, existe cierta correlación negativa entre el ELR y el RT. Además, cuanto mayores son las dimensiones de la sala, mayores valores de ELR se consiguen.

6.1.3 Convolución de los audios de voz con las respuestas al impulso de las salas

Una vez se han descargado los audios de voz y se han simulado las respuestas al impulso de diversas salas, se procede a generar la base de datos que, una vez procesada, se utilizará para entrenar y testear el perceptrón más adelante. En total, se generan 54960 ficheros. Cada uno contendrá el audio resultante de la convolución de uno de los audios de voz de la TIMIT con la respuesta al impulso de una sala. Además, cada fichero incluirá también información relativa al valor de RT y ELR de la RIR correspondiente. Para realizar dicha convolución, se ha utilizado la función *filter* de Matlab.

Nótese que los 54960 ficheros son el resultado de convolucionar todos los audios de voz descargados, 80, con todas las RIRs seleccionadas tras los ajustes, 687.

6.2 Procesado para la extracción de características

En este punto ya ha sido generada la base de datos de los audios que se utilizarán tanto para entrenar como para testear el perceptrón. Si bien, no es posible introducir estos audios directamente a la entrada del perceptrón. Para conseguir que el algoritmo entrene adecuadamente es necesario realizar un procesado y extraer una serie de características de los audios que serán, ahora sí, las que se introduzcan a la entrada del perceptrón. En este trabajo, se extraen características de los audios con dos tipos de procesado:

- Representación tiempo-frecuencia con filtros Gammatone.
- Banco de filtros de modulación temporal.

6.2.1 Representación tiempo-frecuencia con filtros Gammatone

Para realizar este primer tipo de procesado se han seguido los siguientes pasos:

1. Se segmentan los audios en tramas de 25 ms, se enventanan con una ventana Hamming y con un solape entre tramas de 15 ms, es decir, entra una trama nueva cada 10 ms. La trama se denota por $x_l[n]$.
2. Se filtra cada trama con un banco de filtros Gammatone [21]. Citando a Xiong [20], "el banco de filtros Gammatone descompone la señal de voz en bandas de frecuencia, que modelan los filtros auditivos y muestran una resolución mayor para las frecuencias bajas en comparación al espectrograma (en inglés *Short-Time Fourier Transform*, STFT)". Este banco de filtros consiste en 40 filtros con frecuencias centrales entre 100 y 7943 Hz para una frecuencia de muestreo de 16 kHz.
3. De cada trama de audio de 25 ms filtrada con el banco de filtros Gammatone, se calcula la energía a corto plazo (en inglés *Short-Term Energy*, STE) que se define como [22]:

$$E'_i[l] = \sum_{n=n_0[l]}^{n_0[l]+N} u_{il}^2[n], \quad \textbf{(6.2)}$$

donde $n_0[l]$ es el índice temporal de la primera muestra de $x_l[n]$ e i denota el número del filtro Gammatone

4. Finalmente, con el objetivo de reducir el rango de los valores en las características extraídas, se aplica una compresión logarítmica.

La figura 7 representa de forma esquemática los pasos seguidos para extraer las características de este primer tipo de procesado:

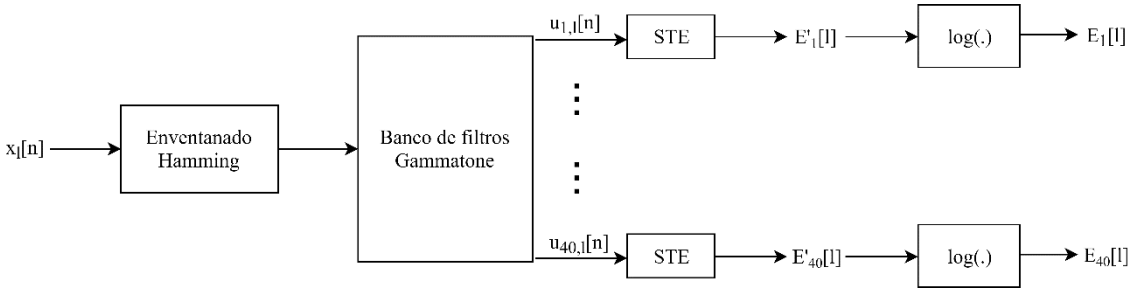

Figura 7: esquema del procesado con filtros Gammatone de cada trama de 25 ms, representada como $x_l[n]$.

Así, tras procesar un audio de los generados con este primer método, se obtiene una matriz de dimensiones 40x139, donde 139 es el número de tramas procesadas de la señal de voz. Esto significa que cada audio se representa por 139 vectores con 40 elementos (características) cada uno de ellos que podrían ya ser introducidos a la entrada del perceptrón.

A continuación se muestra una comparativa entre la representación conseguida al aplicar este primer procesado a un audio frente a realizar un espectrograma:

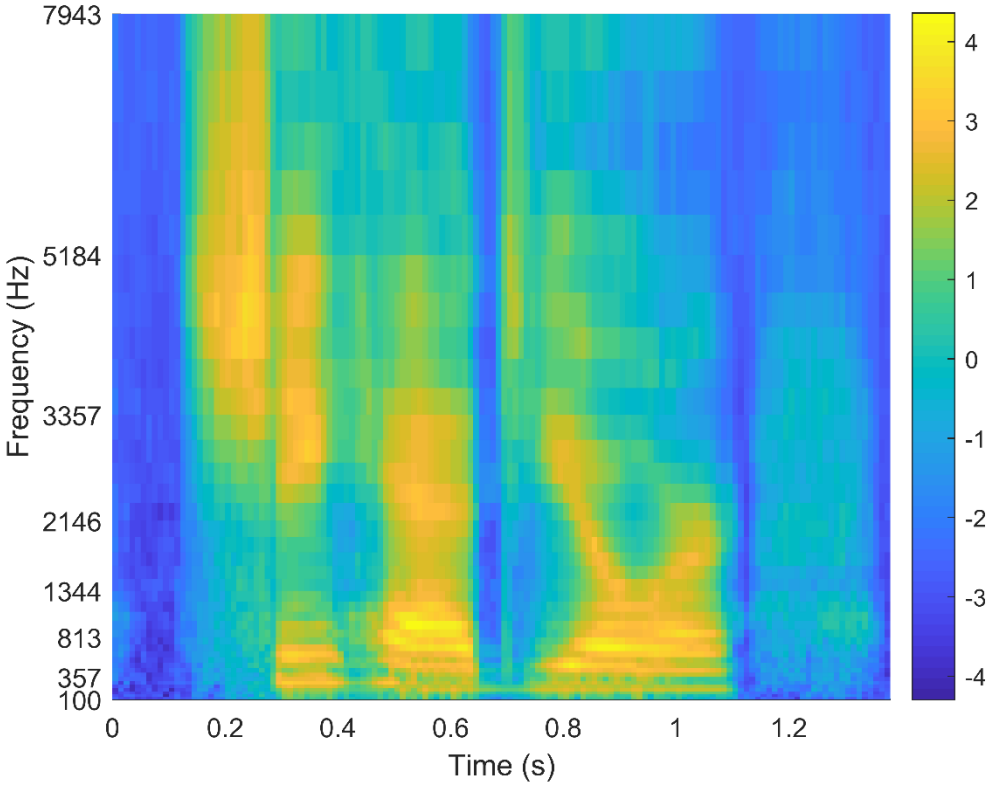

Figura 8: representación tiempo-frecuencia mediante el procesado de la Figura 7 de un audio de voz de mujer que pronuncia las palabras en inglés, *She had your*.

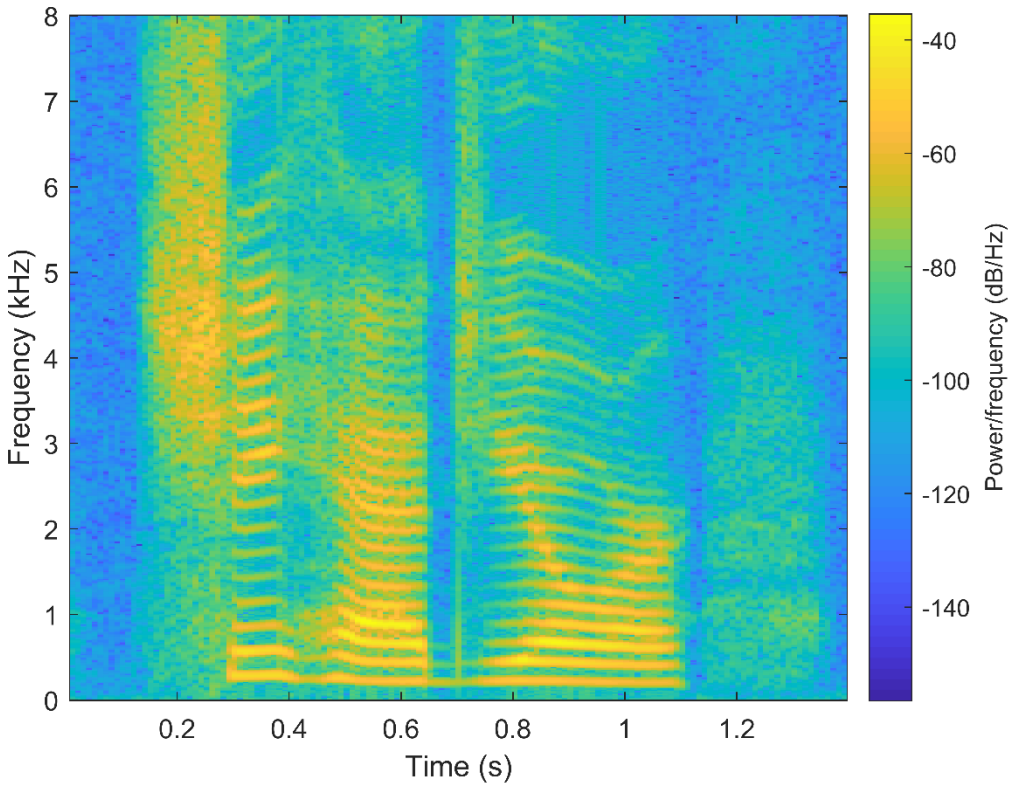

Figura 9: espectrograma del mismo audio que en la Figura 8.

El procesado tiempo-frecuencia empleado presenta una representación más suavizada que el espectrograma, pero es capaz de resaltar mejor las transiciones entre palabras y fonemas como se puede apreciar de la comparación entre ambas figuras.

Si bien, todavía se puede aplicar una última optimización para que el perceptrón pueda entrenarse mejor.

En base a los experimentos ASR [23], las redes neuronales se benefician de información temporal que se consigue concatenando tramas. Así, se han concatenado las muestras de forma que cada una contenga características correspondientes a 11 periodos de tiempo consecutivos. Esto implica que las matrices correspondientes a cada audio pasan de tener unas dimensiones de 40x139 a 440x129 tras la concatenación.

En resumen, tras aplicar el primer tipo de procesado, cada señal de audio estará representada por 129 muestras, o vectores, con 440 características cada una y, por tanto, el vector de características (*features*) de entrada al perceptrón consistirá en un vector de 440 elementos.

Adicionalmente, se detectan aquellos valores de las matrices que provienen de calcular la STE de tramas donde las muestras son todas 0, lo que resulta tras su logaritmo en un valor de $-\infty$, y se sustituyen por el mínimo valor de STE entre todos los valores obtenidos para ese audio y filtro.

Para culminar esta fase, también se resta la media a los valores de la matriz de entrada de forma que el valor de la media sea nulo. Cabe mencionar también que Matlab, al iniciar el entrenamiento de la red, aplica por defecto un preprocesado que se ha mantenido para entrenar las redes en este trabajo de forma que normaliza las entradas para que se encuentren en el intervalo [-1, 1]. Esta normalización es recomendable por el siguiente razonamiento [24]:

- A la entrada de las neuronas de la capa oculta se introduce el producto de los datos de entrada por sus respectivos pesos más el *bias*.
- La función de activación sigmoide de la capa oculta se puede saturar si se introducen a la entrada de la neurona valores elevados (mayores a tres).
- Si no se realiza la normalización de los datos en la capa de entrada, los pesos deben ser muy pequeños para no saturar las funciones de activación de las neuronas de la capa oculta.
- La inicialización de los pesos es aleatoria y puede producirse la saturación.
- Si esta se produce, los gradientes serán muy pequeños y el entrenamiento de la red muy lento.

Así, es una práctica habitual reducir el valor absoluto máximo de los datos de entrada a la unidad con el fin de evitar la saturación.

6.2.2 Filtros de modulación temporal

El segundo tipo de procesado que se realiza en este trabajo está fundamentado en el desarrollado por Xiong en [20]. Se basa en considerar la STE como una señal modulada en el tiempo y donde los parámetros de dicha modulación dependen de la reverberación de la sala.

Sus características se extraen, por tanto, de la STE obtenida en la representación tiempo-frecuencia del procesado del apartado 6.2.1.

Las funciones utilizadas son las siguientes:

$$g_m[l] = s_{carr}[l, f_m] \cdot w_{env}[l, L_m] \tag{6.1}$$

$$s_{carr}[l, f_m] = e^{-i2\pi f_m \cdot T \cdot (l - l_0)} \tag{6.2}$$

$$w_{env}[l, L_m] = \begin{cases} cos^2\left(\frac{\pi(l-l_0)}{L_m}\right) & si \quad 1 < l < L_m \\ 0 & si \quad l = \{1, L_m\} \end{cases} \tag{6.3}$$

El filtro de modulación temporal $g_m[l]$, donde la m denota el índice del filtro, es el resultado de multiplicar la función portadora exponencial compleja s_{carr} por w_{env}, que es un enventanado Hann de fase cero.

Definición de las distintas variables:

- f_m: frecuencia central del filtro.
- L_m: longitud del filtro.
- T: periodo de muestreo de las frecuencias de modulación (10 ms; este valor se extrae del tiempo entre segmentos consecutivos utilizado en las representaciones tiempo-frecuencia).
- $l \in \{1, \dots, L_m\}$: índice de muestras del filtro; donde $l_0 = {}^{L_m}/_{2}$ representa la muestra central.

La selección de las frecuencias centrales y anchos de banda de g_m se basan en el banco de filtros de modulación auditiva propuesto por Dau et al. [25]. Las siete frecuencias centrales resultantes son: {0, 3, 6, 10, 17.1, 29.2, 50} Hz. El resultado son 7 filtros reales y 5 imaginarios, en total 12 filtros. Estos se muestran en las siguientes figuras:

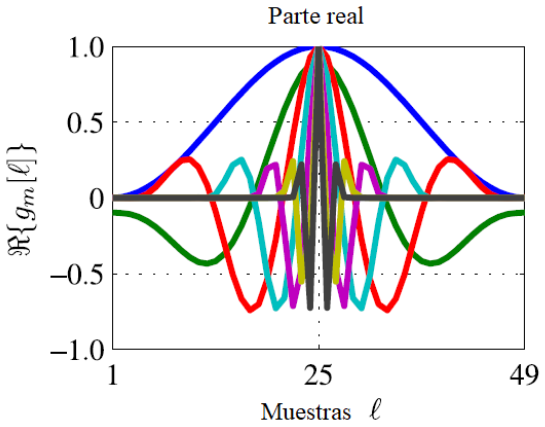

Figura 10: los 7 filtros reales [20].

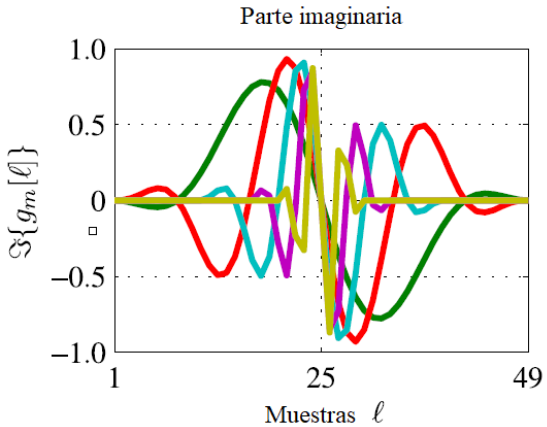

Figura 11: los 5 filtros imaginarios [20].

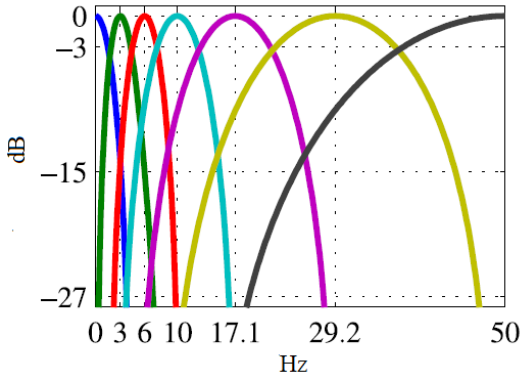

Figura 12: respuesta en frecuencia de los filtros reales [20].

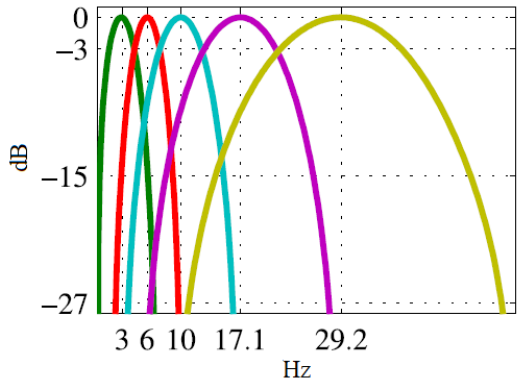

Figura 13: respuesta en frecuencia de los filtros imaginarios [20].

Con la representación tiempo-frecuencia del primer tipo de procesado se transformaba cada audio en una matriz de 139 muestras con 40 características por muestra (40x139). El resultado del segundo procesado, con filtros de modulación temporal, consiste en filtrar la STE (una vez calculado su logaritmo) por cada uno de los 12 filtros generados. Así, cada audio que se introduzca a la entrada del perceptrón siguiendo este segundo procesado estará representado por una matriz de 139 muestras con 480 (40x12) características cada una (480x139). El vector de características (*features*) de entrada contiene, por tanto, 480 elementos.

Resumiendo, los vectores de características resultantes para ambos tipos de procesado poseen dimensiones similares: 440 elementos si se usa la representación tiempo-frecuencia con filtros Gammatone y 480 elementos si se usa el banco de filtros de modulación temporal. El número de vectores que representa una particular señal de audio también es ligeramente diferente, pero sigue siendo del mismo orden: 129 vectores si se usa la representación tiempo-frecuencia con filtros Gammatone y 139 vectores si se usa el banco de filtros de modulación temporal.

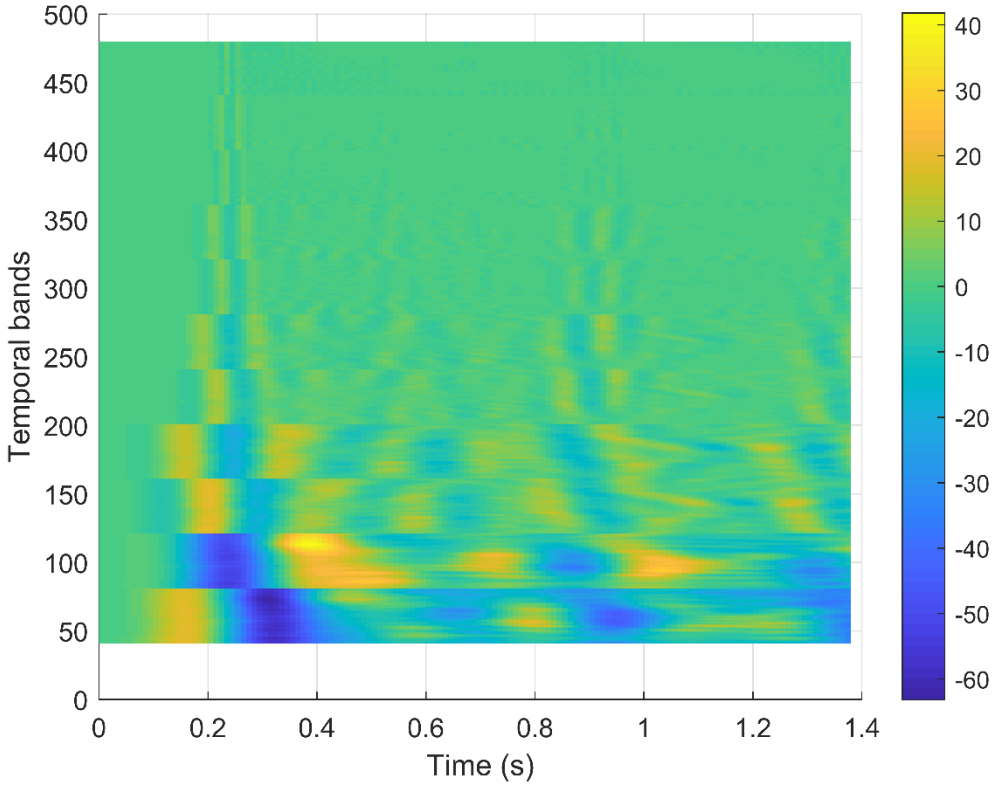

Figura 14: representación de las características extraídas por el procesado de modulación temporal. El audio empleado es el mismo que en la Figura 8.

Si bien, previo a la convolución con los filtros de modulación temporal, es necesario convertir los valores de $-\infty$ de la representación tiempo-frecuencia, de la misma forma que se hizo en el apartado anterior. Asimismo, para introducir estas características al perceptrón se realiza un último preprocesado que consiste en convertir los datos de entrada para que tengan media nula y en realizar la normalización para que se encuentren en el intervalo [-1,1].

7. Entrenamiento de los perceptrones multicapa

En este trabajo se han programado y entrenado múltiples perceptrones multicapa para estimar, por un lado, el RT y, por otro, el ELR. Tanto la programación como el entrenamiento y posterior testeo de estos se ha realizado en Matlab, utilizando la *Deep Learning Toolbox™*.

En todos los experimentos que a continuación se detallan, se han empleado los mismos 32000 audios seleccionados al azar de entre los 54960 generados en total para entrenar el perceptrón. De los 22960 audios restantes, se han seleccionado 2000 al azar para realizar el testeo del desempeño preliminar y 4000 para los resultados finales. Así, se hipotetiza que los perceptrones tendrán suficiente diversidad entre sus muestras de entrenamiento para estimar resultados generalizables.

Sin embargo, con los medios técnicos de los que se ha dispuesto no es posible introducir a la entrada del perceptrón las características extraídas de todos los 32000 audios a la vez. Esto se debe a que no se pueden almacenar en una sola matriz las características extraídas de 32000 audios porque exceden los límites de memoria de Matlab.

Tras realizar algunas pruebas, se concluye que los audios se han de separar en bloques de 4000, de los cuales se extraerán sus características del procesado y, todas ellas, ahora sí, se introducirán a la vez en el perceptrón formando una matriz de 440x516000 o de 480x556000 según se aplique el primer o el segundo tipo de procesado. De esta forma, el perceptrón se entrenará con bloques de datos.

7.1 Clasificador de RT

El objetivo del clasificador de RT será vincular los datos de entrada a una clase de salida que determinará el valor de RT del audio.

Para definir las clases de salida, se han asociado los audios con valores de RT pertenecientes a un mismo intervalo de 100 ms a una misma clase de salida. Si bien esto introduce un error, se trata de una diferencia apenas perceptible, tal y como se recoge en [26] y [27]. Además, tiene la ventaja de que se aumenta el número de muestras en cada clase y se reduce la probabilidad de que existan clases con un número insuficiente de datos.

Cada una de las columnas de la matriz de entrada tiene por tanto vinculada una columna en la matriz de salida que determina la clase a la que pertenece. En la Tabla 1 se muestran las clases posibles con su respectiva columna de salida:

Clase	Intervalo de RT (s)	Columna a la salida del perceptrón
1	[0,3 - 0,4)	[1;0;0;0]
2	[0,4 - 0,5)	[0;1;0;0]
3	[0,5 - 0,6)	[0;0;1;0]
4	[0,6 - 0,68]	[0;0;0;1]

Tabla 1: Clases del RT

Nótese que, a pesar de que todas las columnas de la matriz de entrada correspondientes a un mismo audio (129 consecutivas o 139 según sea el primer o segundo tipo de procesado) corresponden a una misma clase de salida; a la salida estarán todas y cada una de las 129 (o 139)

columnas de salida, con el mismo valor. Esto permite introducir en el perceptrón las columnas desordenadas porque todas ellas estarán vinculadas a una columna de salida concreta. Por lo tanto, se evita que el perceptrón lea las columnas correspondientes a un mismo audio todas seguidas, lo que podría distorsionar su entrenamiento.

7.1.1 Experimentos preliminares para RT

Con el fin de investigar las características más adecuadas que el perceptrón debe tener para realizar un entrenamiento eficiente y unas predicciones lo suficientemente adecuadas, se han realizado una serie de experimentos preliminares. En ellos, se ha modificado tanto el número de capas del perceptrón como el orden y número de iteraciones en el entrenamiento de los bloques. Todo ello se detallará a continuación.

El tipo de procesado utilizado en este apartado es el de la representación tiempo frecuencia con filtros Gammatone, descrito en el apartado 6.2.1.

El parámetro que se ha usado como referencia para caracterizar una u otra forma de entrenamiento como mejor o peor es el desempeño o *performance*, medido como la entropía cruzada. Esta ya se definió en el apartado 4.2. Cabe mencionar que, cuanto más cercano a cero es el valor de la entropía cruzada, mejor es el desempeño del perceptrón.

En el entrenamiento de los perceptrones con Matlab, existen también una serie de características de forma que, de cumplirse alguna de ellas, el entrenamiento del bloque se interrumpe. Estas son:

- Número máximo de iteraciones: en cada iteración, la red se entrena con todos los datos de los que dispone a la entrada y salida, actualizando los pesos cada vez con el fin de mejorar su desempeño.
- Tiempo máximo.
- Comprobaciones del bloque de validación máximas: al introducir los datos al perceptrón para realizar el entrenamiento, se realiza una separación en los datos de forma que el 85% se utilice verdaderamente para entrenar y el 15% restante para conformar el bloque de validación (estos porcentajes se pueden modificar). La red actualiza sus pesos en función del desempeño de los datos de entrenamiento, pero también calcula en cada iteración el desempeño del bloque de validación. Puede ocurrir que, llegado un determinado punto, el desempeño del bloque de entrenamiento continúe mejorando pero no así el del bloque de validación. En este momento, es probable que la red se esté sobreentrenando, "memorizando" los datos de los que dispone para entrenar y empiece a ser menos generalizable ante datos nuevos. Es por esto que existen las comprobaciones del bloque de validación máximas, de forma que si el desempeño de este bloque empeora durante, en este caso, 6 iteraciones consecutivas, se interrumpe el entrenamiento.
- Alcanzar un valor determinado de *performance* objetivo.
- Alcanzar un valor determinado de gradiente.

Las siguientes características vienen por defecto en Matlab y no se han modificado para ninguno de los experimentos:

- Tiempo máximo: infinito.
- Máximo número de comprobaciones del bloque de validación: 6
- Valor de performance: 0
- Valor del gradiente: 1e-6

7.1.1.1 Primer experimento preliminar

Con este primer experimento, se pretende dar respuesta a la siguiente pregunta: ¿es más adecuado emplear una capa oculta de 128 neuronas, como hace Xiong [20], o dos capas ocultas, de 60 y 15 neuronas respectivamente?

Para ello, se han entrenado los perceptrones procediendo con los siguientes pasos:

1. Se introducen las características extraídas del primer bloque de 4000 audios. El número máximo de iteraciones por bloque es de 1000.
2. El entrenamiento del primer bloque finaliza, bien porque se ha cumplido alguna de las condiciones de interrupción o porque se han realizado 1000 iteraciones.
3. Se guarda el perceptrón resultante.
4. Se continúa entrenando el perceptrón introduciendo el resto de bloques, repitiendo con cada uno los 3 primeros pasos.
5. Finalmente, se habrán guardado 8 perceptrones de forma que el octavo habrá sido entrenado con los 32000 audios.

La razón para guardar todos los perceptrones y no solo el último radica en que de esta forma podremos medir el desempeño de cada uno de ellos y representar su evolución. También permitiría observar, si se diera el caso, el que un tipo de perceptrón consiguiera mejores desempeños con los primeros n bloques, pero fuera superado por el otro tipo de perceptrón al añadir el resto de bloques al entrenamiento.

En la siguiente gráfica se muestra la evolución del desempeño de cada perceptrón:

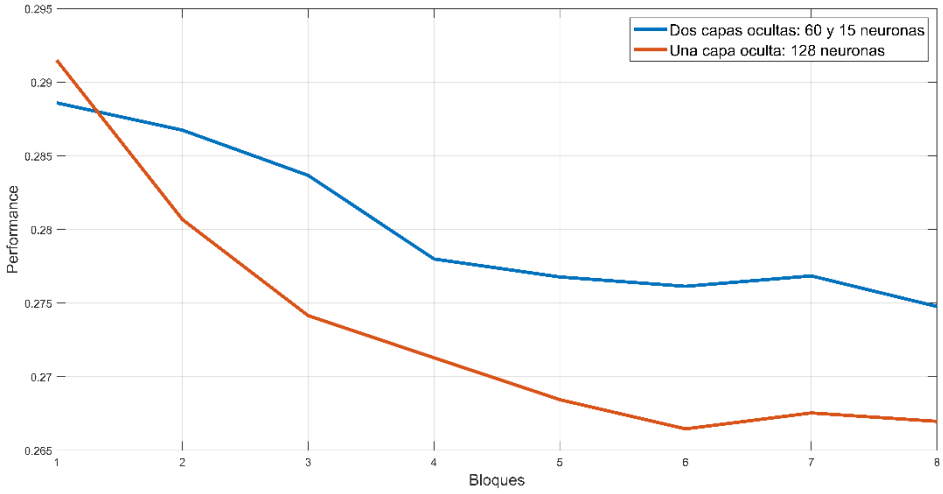

Figura 15: comparativa de la evolución del rendimiento del perceptrón utilizando dos capas ocultas de 60 y 15 neuronas frente a una sola capa oculta de 128 neuronas.

Como se puede comprobar en la Figura 15, si bien tras entrenarlos con el primer bloque de 4000 audios el rendimiento del que tiene dos capas ocultas es ligeramente mejor, conforme avanza el entrenamiento se aprecia un mejor rendimiento en el que tiene una capa oculta de 128 neuronas.

7.1.1.2 Segundo experimento preliminar

En este segundo experimento, se investiga lo siguiente: ¿es mejor entrenar cada bloque de 4000 audios una vez con un máximo de 1000 iteraciones o entrenarlos múltiples veces pero con un máximo de 100 iteraciones cada vez?

Los dos perceptrones considerados en este experimento tendrán una capa oculta de 128 neuronas, en tanto que esta configuración ha demostrado conseguir mejores resultados tras el primer experimento.

Al entrenar cada bloque con 1000 iteraciones se tiene el riesgo de que el bloque no sea lo suficientemente representativo y que los pesos queden con valores demasiado definidos tras el primer bloque. Con el fin de limitar este riesgo, se prueba a realizar los entrenamientos con un máximo de 100 iteraciones consecutivas por bloque. La secuencia de este segundo método es la siguiente:

1. Se entrena el perceptrón con el primer bloque de 4000 audios y un máximo de 100 iteraciones.
2. Se guarda este primer perceptrón
3. Se continúa el entrenamiento repitiendo los dos primeros pasos con cada uno de los otros siete bloques de 4000 audios.
4. Una vez se ha entrenado con los ocho bloques, se reordenan estos al azar y se repite el entrenamiento siguiendo los tres primeros pasos.

El reordenamiento y posterior entrenamiento se realiza 12 veces. Así, se generan 96 perceptrones en total, que son el resultado de generar 8 perceptrones por cada una de las 12 veces.

En la Figura 16 se puede comparar el desempeño de uno y otro método:

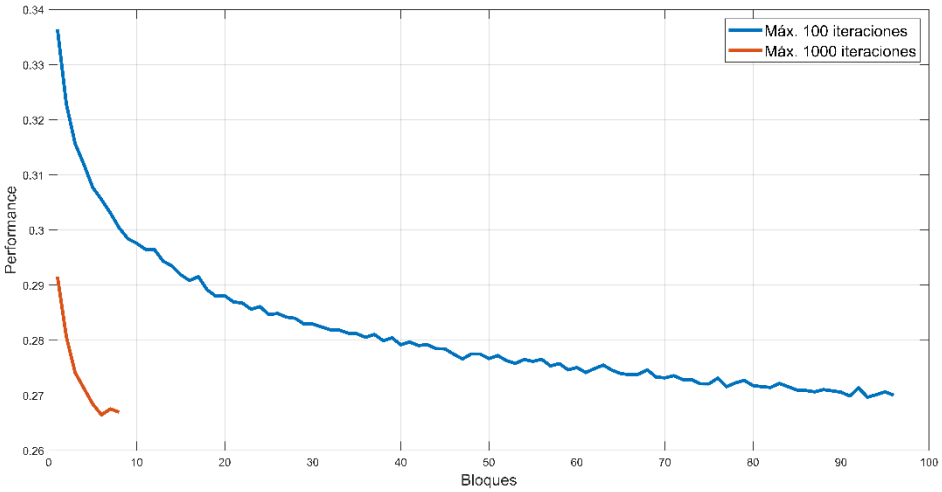

Figura 16: comparativa de la evolución del rendimiento del perceptrón aplicando un máximo de 100 iteraciones consecutivas por bloque frente a 1000.

Suponiendo que en ambos métodos se han finalizado los entrenamientos cuando se ha alcanzado el número máximo de iteraciones; para la gráfica en azul se habrían realizado 9600 iteraciones (96x100) y, para la naranja, 8000 (8x1000). A pesar de ello, la segunda consigue un rendimiento mejor.

En lo que respecta a los tiempos de entrenamiento, la red representada en azul ha requerido más de 27 horas únicamente para entrenarse, sin contar la carga de los bloques que se realiza 96 veces; mientras que la red representada en naranja requiere alrededor de 11 horas y realiza la carga de los bloques únicamente 8 veces.

Se concluye por tanto que es mejor realizar el entrenamiento con cada bloque una vez con un número de iteraciones elevado a reducir el número de iteraciones máximo e ir cambiando de bloques con más frecuencia.

7.1.1.3 Tercer experimento preliminar

Realizar el entrenamiento de cada bloque una vez con 1000 iteraciones parece lo más adecuado hasta el momento. Sin embargo, solo se ha probado a introducir los bloques en el perceptrón siguiendo un determinado orden. Si bien los audios de cada bloque fueron seleccionados al azar entre una muestra amplia, no está de más asegurar que el buen rendimiento de este entrenamiento no está vinculado al orden en el que se entrenan los bloques. De ahí que se realice el tercer experimento.

¿Existirán diferencias notables en el desempeño del perceptrón si el orden en el que se introducen los ocho bloques para su entrenamiento es diferente?

Se prueba a entrenar 4 perceptrones usando los ocho bloques, con 1000 iteraciones por bloque, con un orden de introducción de los bloques para el entrenamiento diferente en cada perceptrón. El resultado es el siguiente:

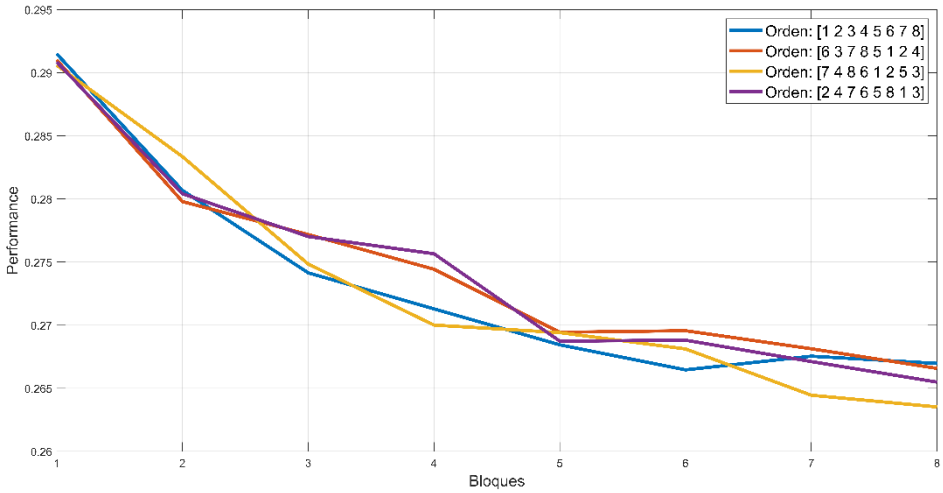

Figura 17: comparativa de la evolución del rendimiento del perceptrón modificando el orden de entrada de los bloques.

Se comprueba que el orden en que se introducen los bloques para realizar el entrenamiento apenas afecta al rendimiento de la red, situándose la *performance* en los 4 perceptrones finales por debajo de 0,27; valor que no conseguían alcanzar los otros métodos que se han probado.

7.1.1.4 Cuarto experimento preliminar

Hasta ahora, se ha utilizado la sigmoide como función de activación en la capa oculta debido a su popularidad. Sin embargo, según Nair [28], las unidades lineales rectificadas mejoran el rendimiento en cierto tipo de redes neuronales. Esto conduce al siguiente experimento:

¿Se conseguirán mejores resultados utilizando unidades lineales rectificadas como función de activación en lugar de la sigmoide?

Se prueba a entrenar dos perceptrones, siguiendo el esquema que mejores resultados ha dado hasta ahora, uno de ellos con unidades lineales rectificadas como función de activación en la capa oculta y el otro con una sigmoide.

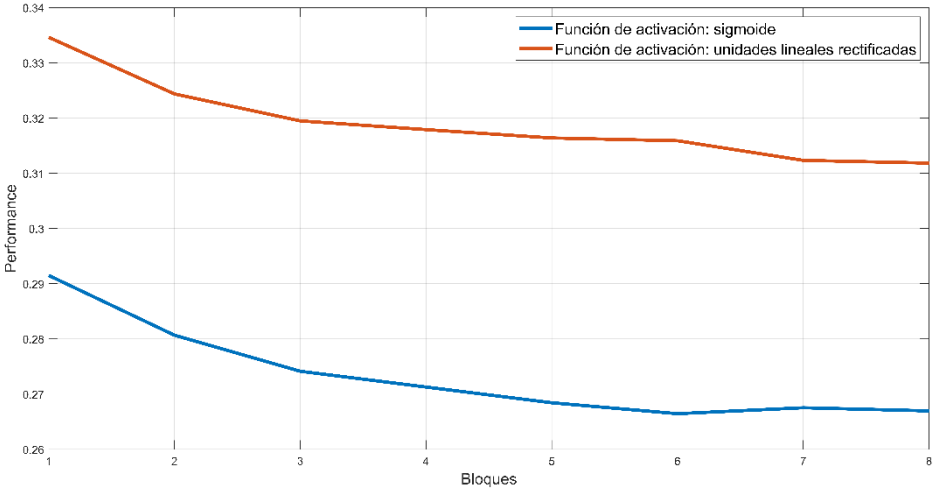

Figura 18: comparativa de la evolución del rendimiento del perceptrón utilizando unidades lineales rectificadas frente a la sigmoide como función de activación.

Tal y como se observa, la sigmoide consigue un rendimiento mejor para todos los bloques.

7.1.2 Conclusiones tras los experimentos preliminares

Tras comparar el rendimiento de múltiples configuraciones de perceptrones, se concluye que lo más adecuado para realizar los experimentos finales del RT y, extrapolando, del ELR, será:

- Entrenar el perceptrón por bloques de características correspondientes a 4000 audios y un número máximo de 1000 iteraciones por bloque.
- Utilizar una sola capa oculta, de 128 neuronas para el RT (pendiente de una última prueba para el ELR).
- Emplear la sigmoide como función de activación de la capa oculta.

7.1.3 Entrenamientos definitivos para RT

Teniendo en cuenta todo lo aprendido de los experimentos preliminares, se realiza el entrenamiento final de los perceptrones para predecir el RT, empleando por un lado las características de la representación tiempo frecuencia con filtros Gammatone y, por otro, las extraídas de la modulación temporal. En la Figura 19 se pueden comparar los rendimientos conseguidos utilizando uno y otro procesado.

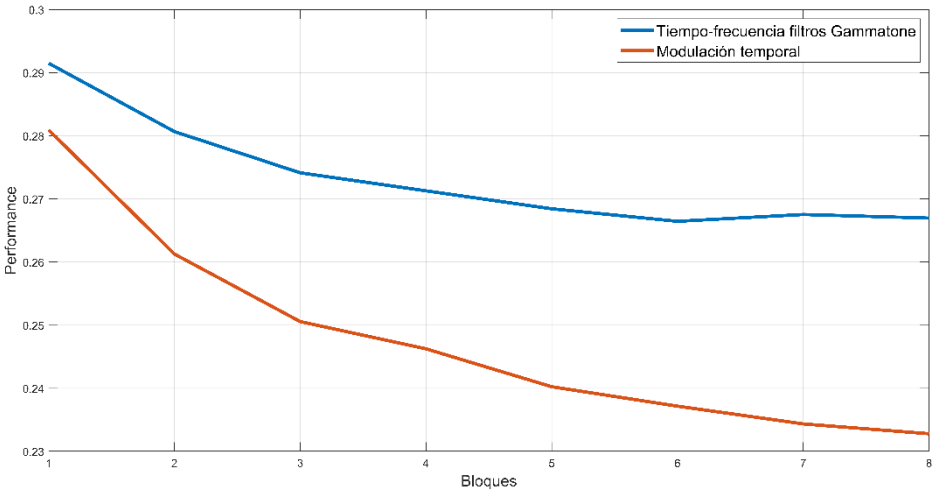

Figura 19: comparativa de la evolución del rendimiento del perceptrón de RT utilizando los dos tipos de procesado.

La diferencia entre rendimientos es notable a favor del procesado de modulación temporal, aunque también lo es la diferencia en tiempos de entrenamiento. El tiempo de entrenamiento del perceptrón que utiliza las características extraídas de la modulación temporal es de aproximadamente 20 horas; esto es casi el doble que para el otro, el cual requiere alrededor de 11 horas.

7.2 Clasificador de ELR

La principal diferencia a la hora de predecir el RT o el ELR, en este trabajo, es el número de clases de salida de cada clasificador. Mientras que para el caso del RT únicamente se definen 4, para el ELR habrá 19; tal y como se recoge en la siguiente tabla:

Clase	**Intervalo de ELR(dB)**
1	[0 , 1)
2	[1 , 2)
3	[2 , 3)
[...]	[...]
18	[17 , 18)
19	[18 , 19]

Tabla 2: Clases de ELR.

Se observa que los audios cuyo valor de ELR se encuentra en un mismo intervalo de ±0.5 dB se asocian a una misma clase. El objetivo es el mismo que en las clases de RT, tener un número suficiente de muestras por cada clase. Además, el error que se introduce debido a esta agrupación es prácticamente imperceptible, como se sugiere en [26] y [27].

7.2.1 Experimento preliminar para ELR

Para entrenar este perceptrón se parte de las conclusiones obtenidas de los resultados preliminares del apartado 7.1. Si bien, debido al elevado número de clases de salida, se contempla la posibilidad de aumentar el número de neuronas en la capa oculta. Con más neuronas, se hipotetiza lo siguiente:

- El desempeño del perceptrón será mejor.
- El tiempo de entrenamiento será mayor.

Lo ideal será encontrar un equilibrio entre estas dos variables. Para ello, se lleva a cabo el siguiente experimento en el que se compara el rendimiento del perceptrón con 128 y 256 neuronas en la capa oculta, empleando el primer tipo de procesado:

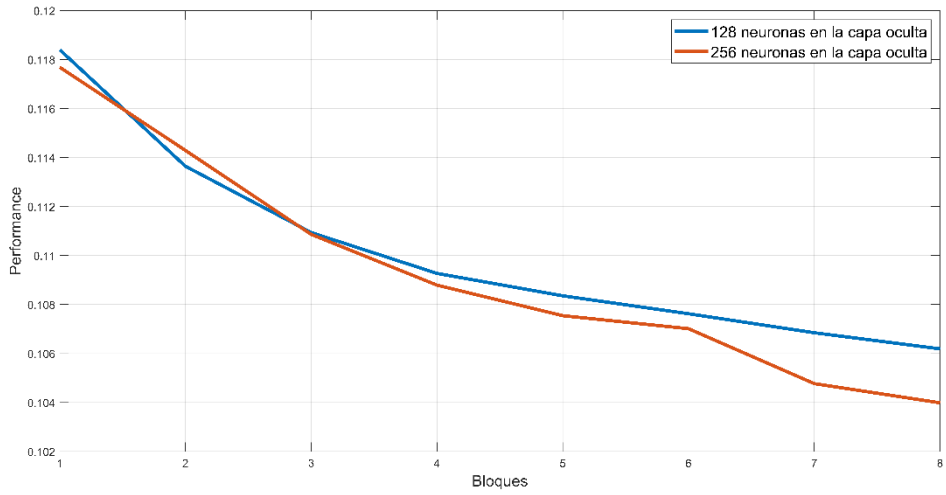

Figura 20: comparativa de la evolución del rendimiento del perceptrón del ELR con 256 neuronas frente a 128, en la capa oculta

Se observa en la Figura 20 que el rendimiento conseguido es muy similar, alcanzándose diferencias del orden de las milésimas con los últimos bloques a favor del perceptrón de 256 neuronas.

En lo relativo a los tiempos de entrenamiento:

Neuronas en la capa oculta	Tiempo de entrenamiento total (horas)
128	18,88
256	41,24

Tabla 3: comparativa de los tiempos de entrenamiento utilizando 256 neuronas en la capa oculta frente a 128.

Doblar el número de neuronas implica también multiplicar el tiempo de entrenamiento total por más de dos, todo ello para conseguir un rendimiento muy similar.

Se concluye que emplear 128 neuronas será lo más adecuado también para el perceptrón del ELR. Por tanto, las características de los perceptrones para el RT y para el ELR serán iguales, salvo en el número de neuronas en la capa de salida, que está vinculado al número de clases de salida (4 y 19 respectivamente).

7.2.2 Entrenamientos definitivos para ELR

Finalmente, se realizan los entrenamientos de los perceptrones para predecir el ELR utilizando los dos tipos de procesados y la configuración más adecuada, obtenida del conjunto de experimentos preliminares. A continuación se muestran las diferencias de rendimiento empleando los distintos procesados:

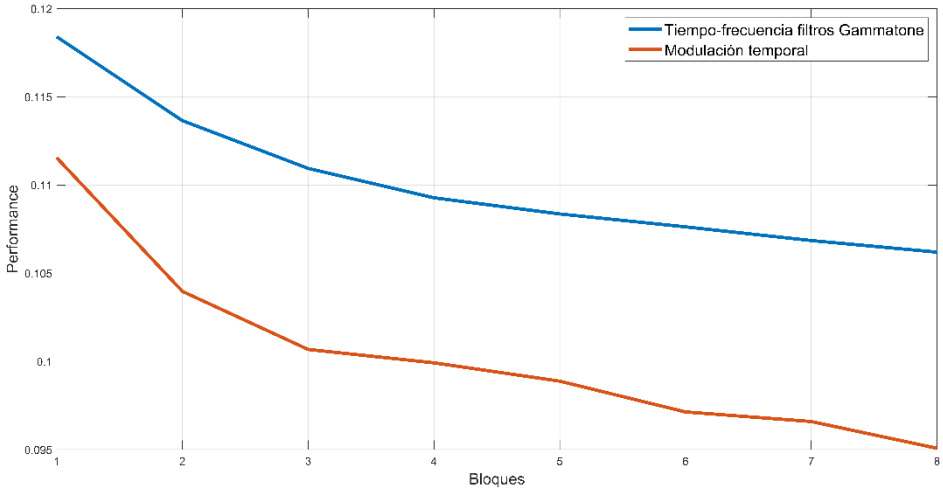

Figura 21: comparativa de la evolución del rendimiento del perceptrón de ELR utilizando los dos tipos de procesado.

Al igual que en el caso del RT, se consiguen resultados notablemente superiores con el procesado de modulación temporal. En cuanto al tiempo de entrenamiento, el de modulación temporal requiere de 8,5 horas más, tardando alrededor de 27 horas en total.

8. Resultados

En el apartado anterior se han visualizado las diferencias en rendimiento de los distintos perceptrones hasta encontrar los más adecuados. Si bien, midiendo este rendimiento como la entropía cruzada no resulta del todo intuitivo entender hasta qué punto se generan buenos o malos resultados. El objetivo de este apartado es precisamente clarificar este aspecto y comparar los resultados de los 4 perceptrones finales. Dos de ellos serán los perceptrones que clasifican el RT, utilizando cada uno un tipo de procesado distinto para los datos de entrada. Los otros dos serán los que clasifican el ELR, también utilizando cada uno distintos procesados.

Para generar los resultados se utilizarán 4000 audios, seleccionados al azar de entre el bloque de audios no utilizados en el entrenamiento.

8.1 Muestra a muestra

Cada audio se introduce al perceptrón formando 129 o 139 muestras, según el tipo de procesado que se emplee; tal y como se describió en el apartado 6.2. El objetivo de este apartado es medir la precisión de los perceptrones para generar las predicciones muestra a muestra.

8.1.1 Primer tipo de procesado, RT

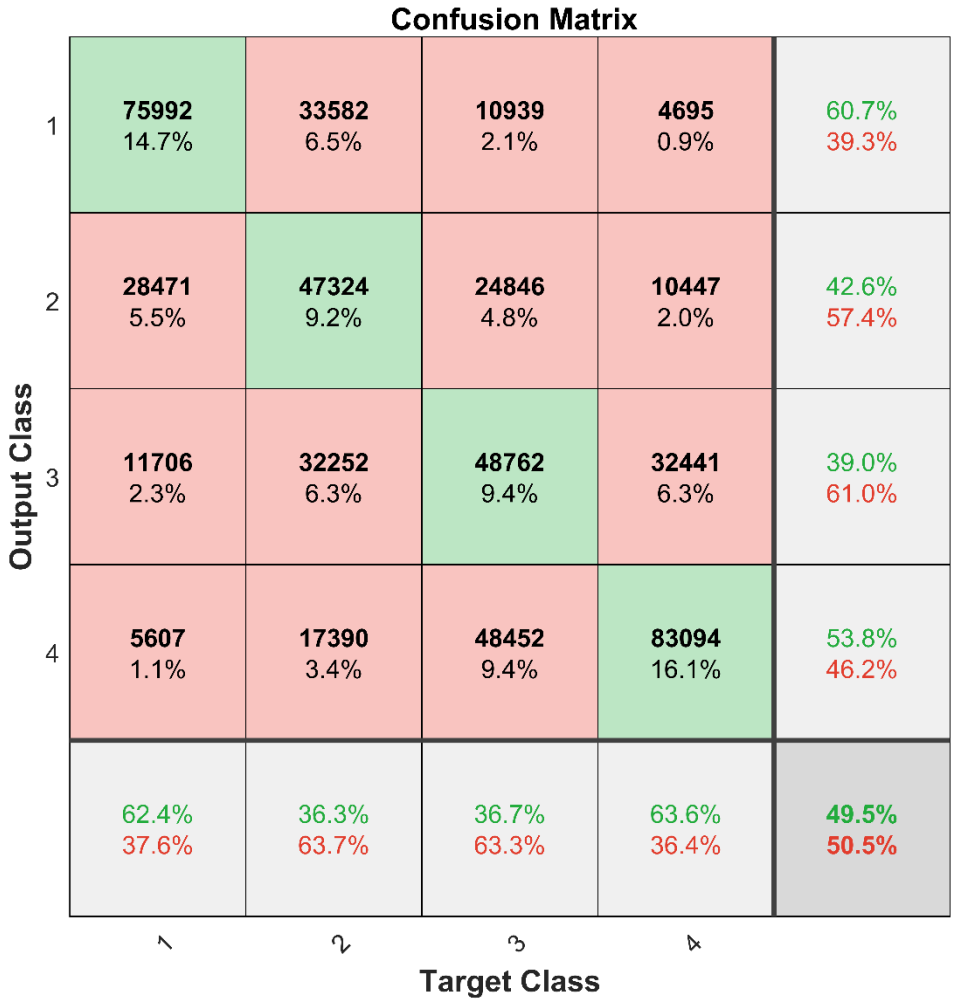

Figura 22: Matriz de confusión para RT con el primer tipo de procesado, muestra a muestra.

La matriz de confusión indica que, si se toma una muestra al azar de un audio cualquiera, habrá un 49,5% de probabilidad de que el perceptrón estime un valor de RT correcto.

Además, dependiendo de la clase de salida que estime el perceptrón, esta probabilidad podrá también aumentar o disminuir. Por ejemplo, si el perceptrón estima que la clase de salida es la primera, habrá un 60,7% de probabilidad de que la predicción sea correcta. Este dato se extrae del cuadro superior derecho de la Figura 22.

Como es lógico, donde menos error comete el perceptrón es en las clases de los extremos.

8.1.2 Segundo tipo de procesado, RT

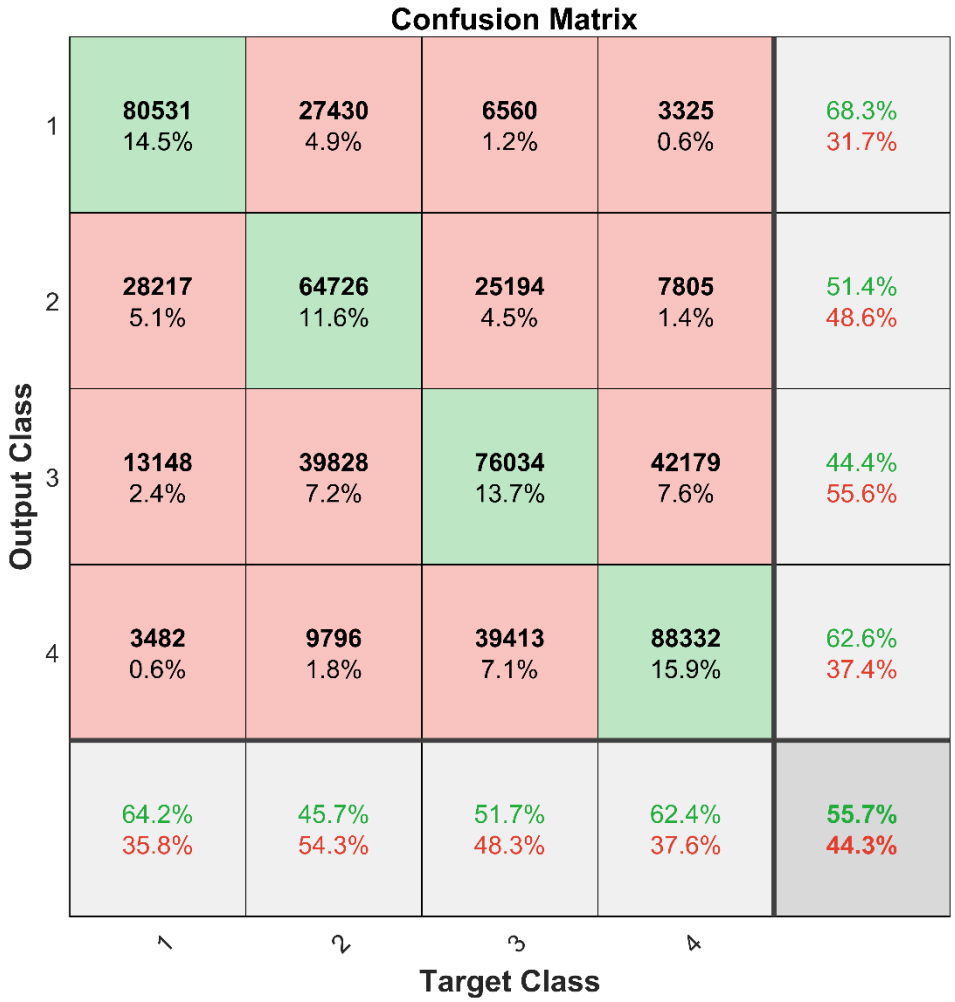

Figura 23: Matriz de confusión para RT con el segundo tipo de procesado, muestra a muestra.

En este caso, el perceptrón es capaz de generar estimaciones correctas en el 55,7% de los casos. Mejora en más de un 6% con respecto al primer tipo de procesado.

8.1.3 Primer tipo de procesado, ELR

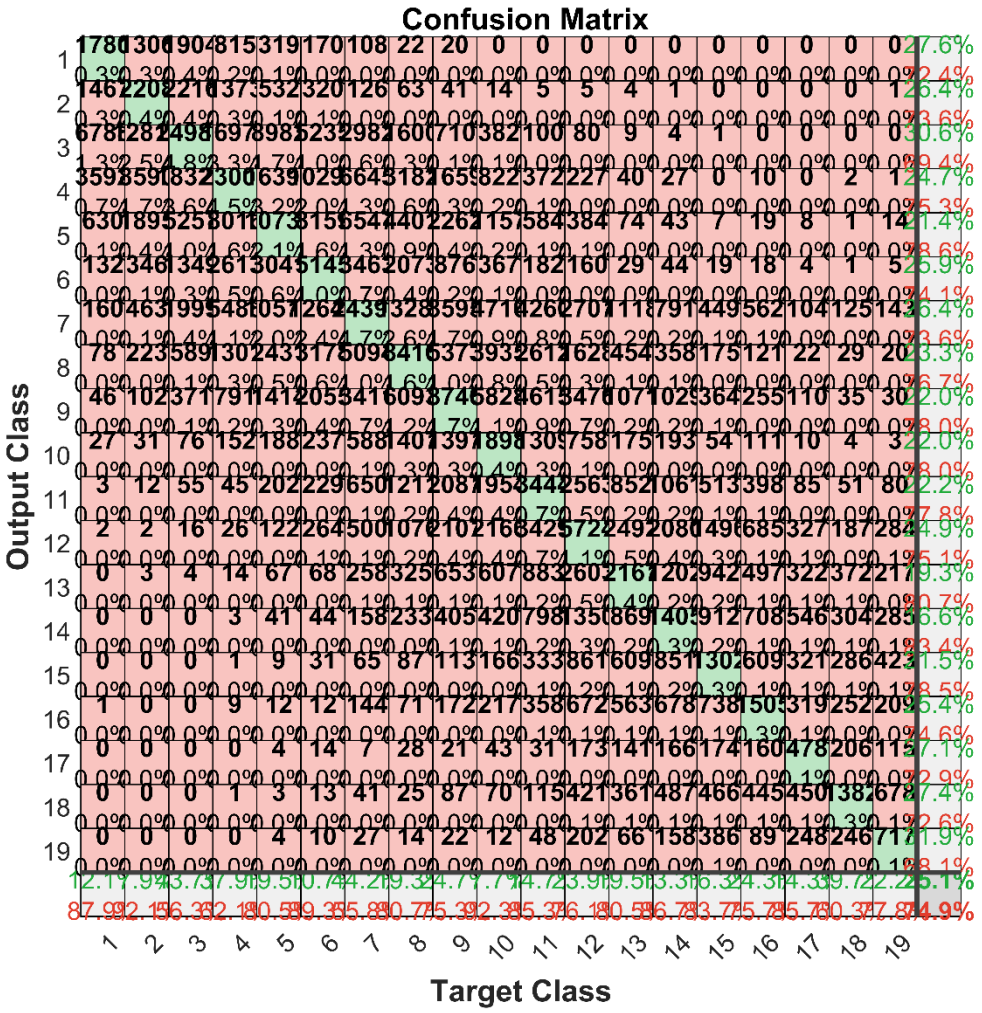

Figura 24: Matriz de confusión para ELR con el primer tipo de procesado, muestra a muestra.

La Figura 24 se ha incluido, a pesar de que no se puedan ver todos los números, para visualizar cómo el perceptrón apenas comete errores entre clases distantes (los valores son nulos o reducidos en las esquinas superior derecha e inferior izquierda de la matriz de confusión). En cuanto al porcentaje de acierto, este es de un 25,1%. Teniendo en cuenta que existen 19 clases, es un resultado aceptable.

8.1.4 Segundo tipo de procesado, ELR

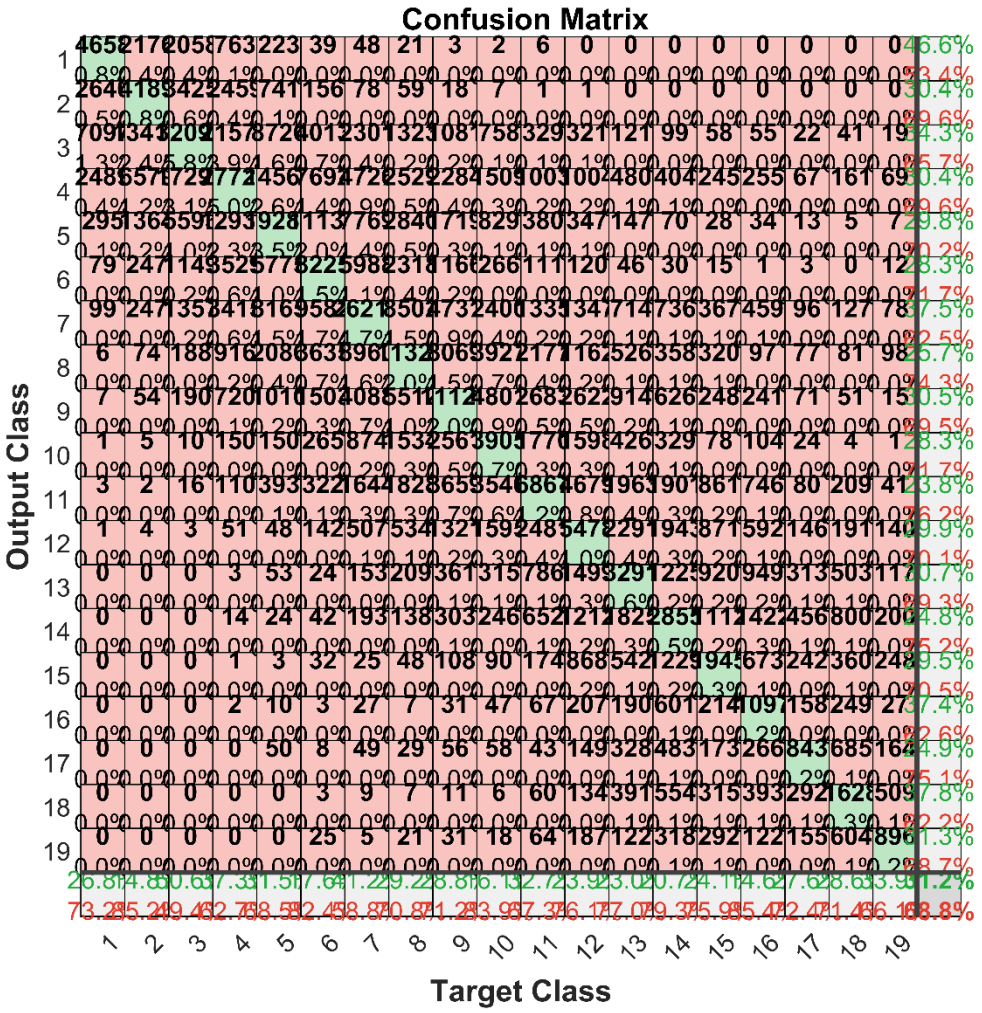

Figura 25: Matriz de confusión para ELR con el primer tipo de procesado, muestra a muestra.

El porcentaje de acierto de nuevo es notablemente superior con el segundo procesado. Asciende a un 31,2%.

8.2 Audio completo

En el apartado anterior se analizan los resultados muestra a muestra. Por ello, la probabilidad de que el perceptrón se equivoque es relativamente alta ya que habrá muestras que no contengan información válida para generar una predicción a partir de las mismas.

El objetivo de este apartado es evaluar los resultados tomando los audios completos. Estos duran poco más de un segundo. Como las predicciones se generan muestra a muestra, se prueban dos alternativas:

1. Realizar la media de todas las salidas del perceptrón correspondientes a un mismo audio, de forma que se vinculará el audio completo a la clase que mayor probabilidad tenga.
2. Mismo método que en el punto anterior pero sustituyendo la media por la mediana.

8.2.1 Primer tipo de procesado, RT, realizando la media

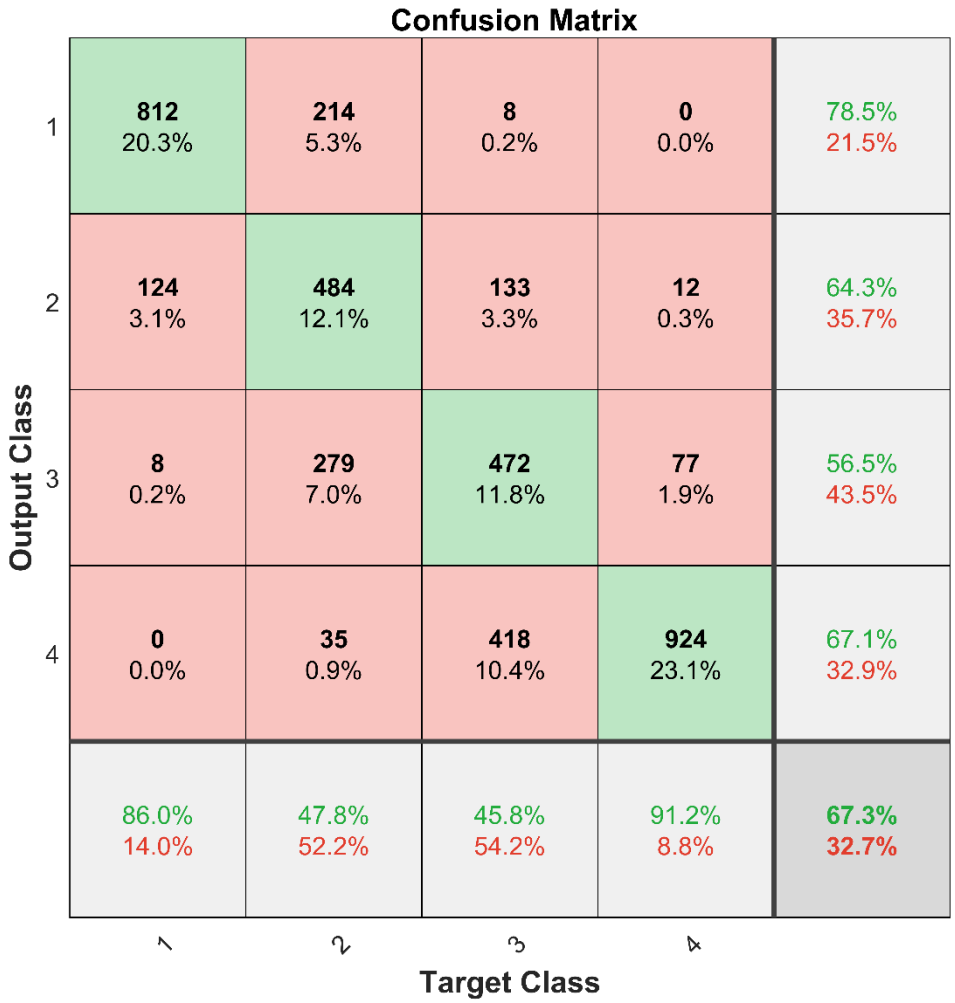

Figura 26: Matriz de confusión para RT con el primer tipo de procesado, audio a audio realizando la media.

8.2.2 Primer tipo de procesado, RT, realizando la mediana

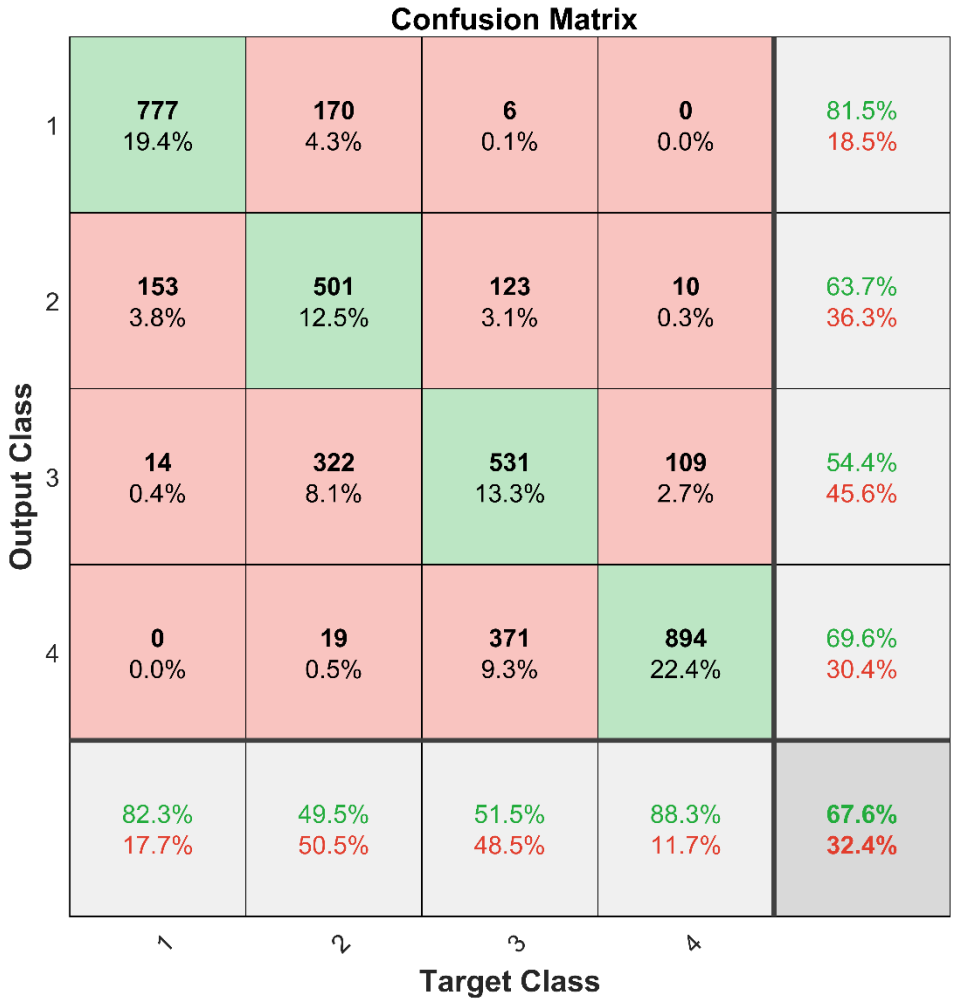

Figura 27: Matriz de confusión para RT con el primer tipo de procesado, audio a audio realizando la mediana.

8.2.3 Segundo tipo de procesado, RT, realizando la media

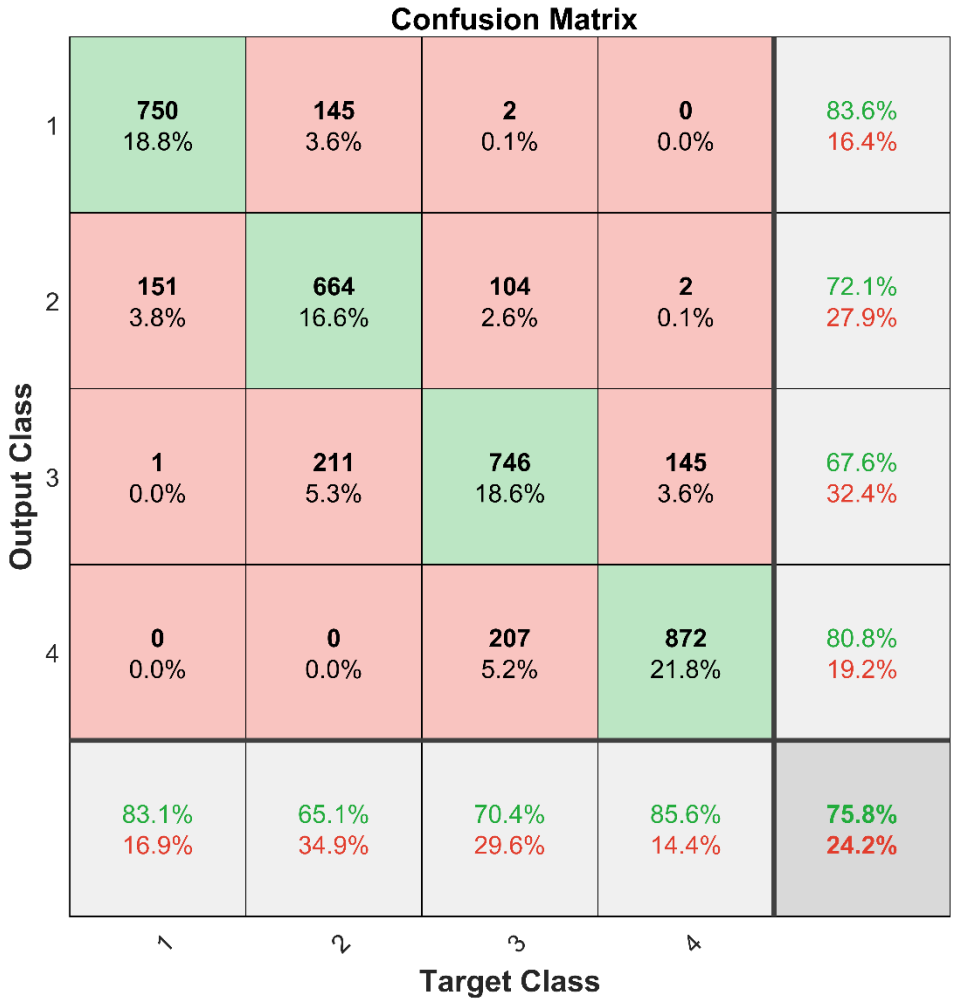

Figura 28: Matriz de confusión para RT con el segundo tipo de procesado, audio a audio realizando la media.

8.2.4 Segundo tipo de procesado, RT, realizando la mediana

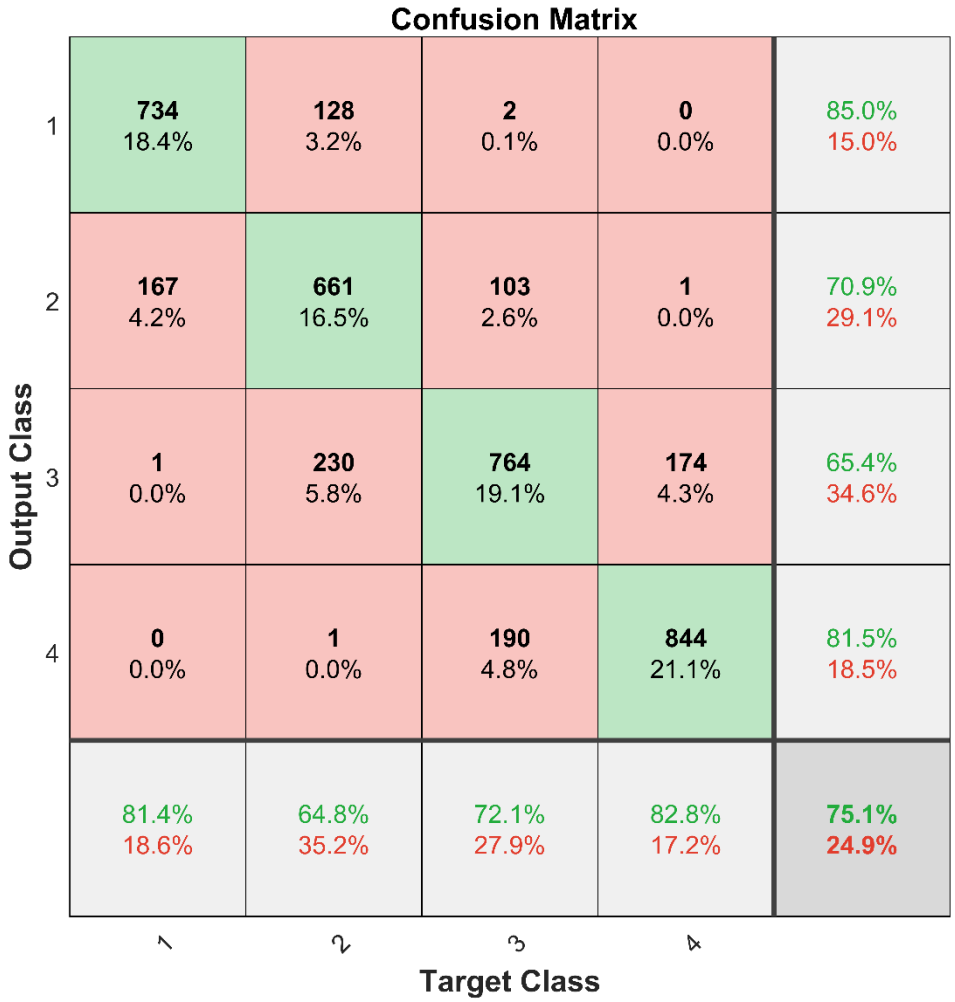

Figura 29: Matriz de confusión para RT con el segundo tipo de procesado, audio a audio realizando la mediana.

Los mejores resultados generales se consiguen con el segundo tipo de procesado y realizando la media. En ellos, la probabilidad de que un audio pertenezca a la clase de RT que indica el perceptrón es de un 75,8%.

Utilizando el primer tipo de procesado, los resultados son ligeramente mejores utilizando la mediana.

8.2.5 Primer tipo de procesado, ELR, realizando la media

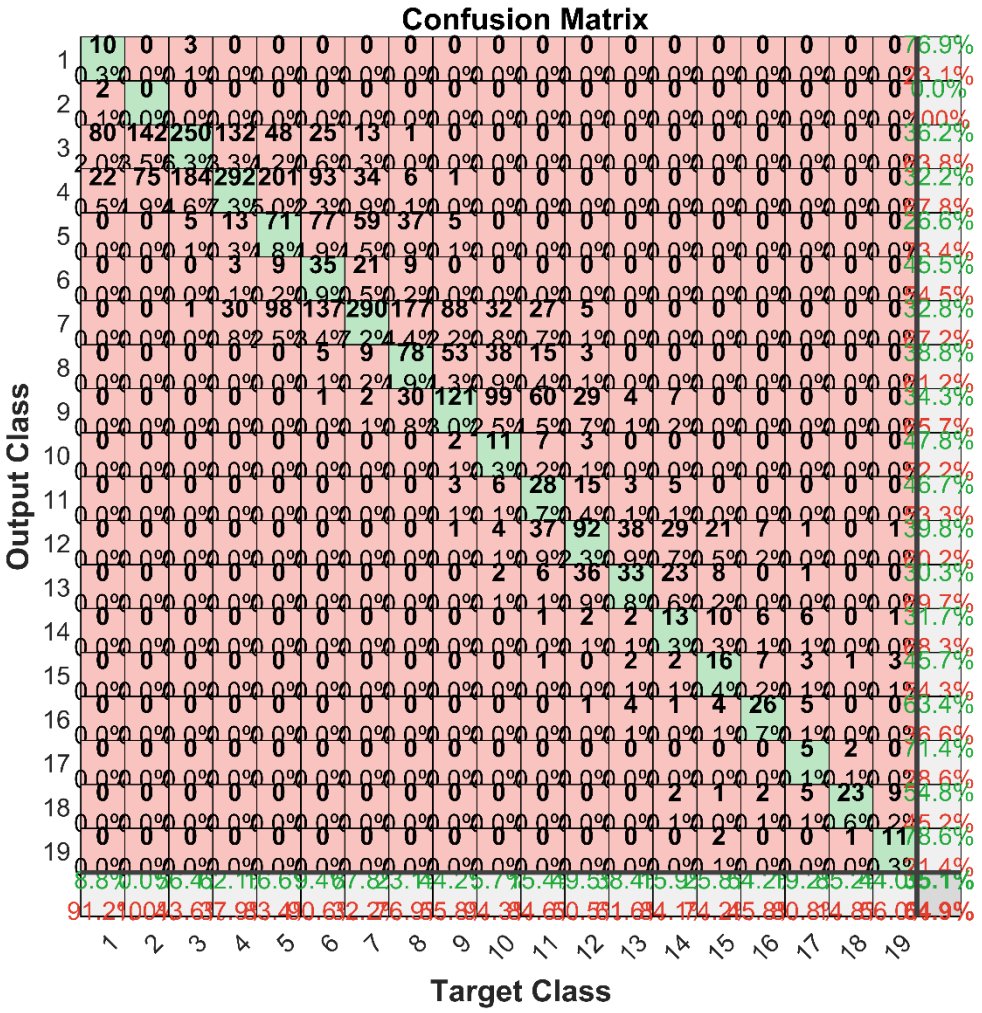

Figura 30: Matriz de confusión para ELR con el primer tipo de procesado, audio a audio realizando la media.

La precisión es del 35,1%.

8.2.6 Primer tipo de procesado, ELR, realizando la mediana

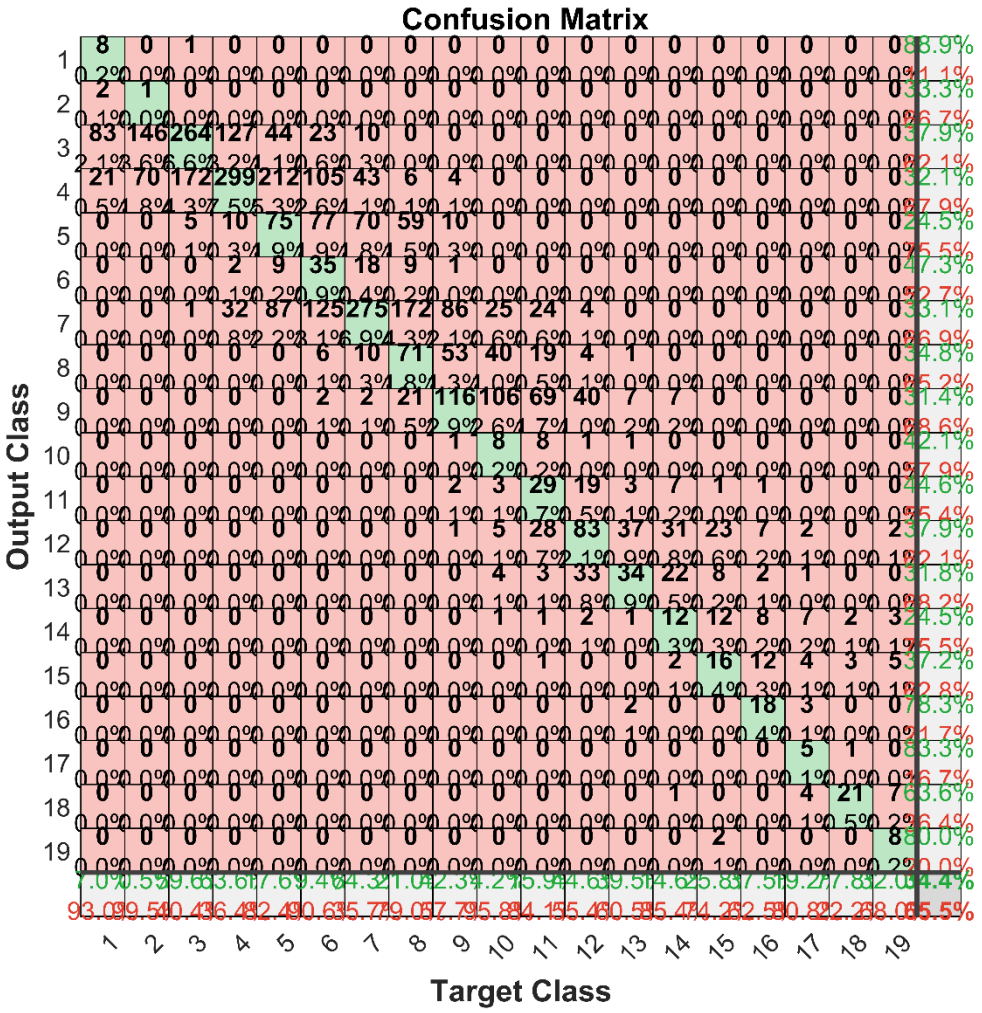

Figura 31: Matriz de confusión para ELR con el primer tipo de procesado, audio a audio realizando la mediana.

La precisión es del 34,4%.

8.2.7 Segundo tipo de procesado, ELR, realizando la media

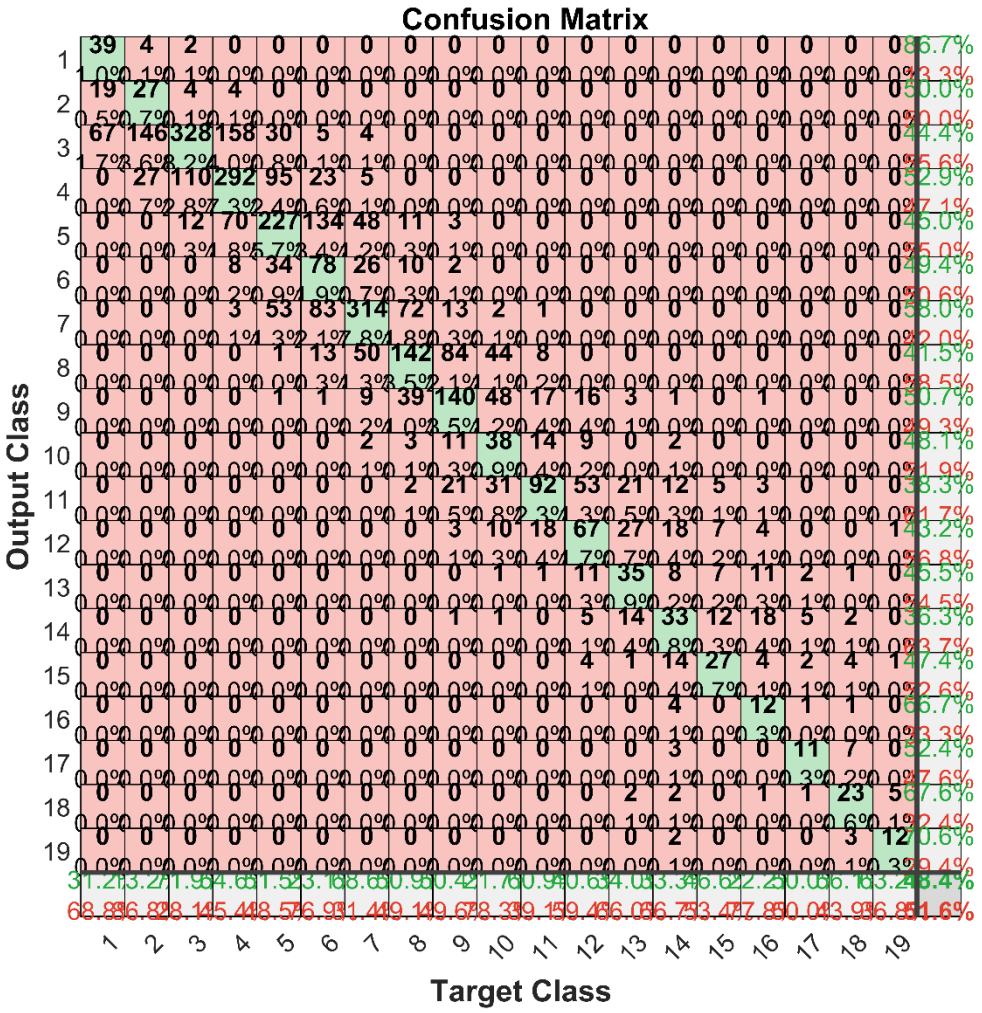

Figura 32: Matriz de confusión para ELR con el segundo tipo de procesado, audio a audio realizando la media.

La precisión es del 48,4%.

8.2.8 Segundo tipo de procesado, ELR, realizando la mediana

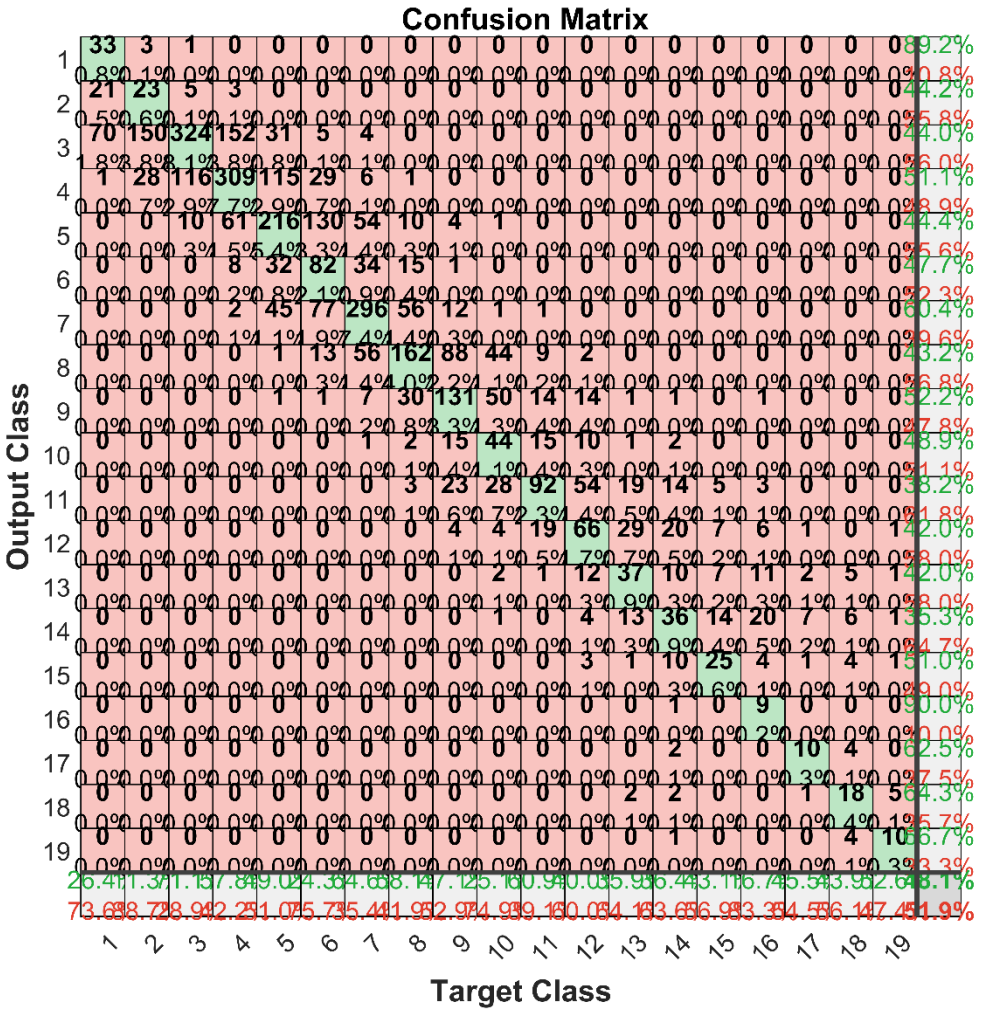

Figura 33: Matriz de confusión para ELR con el segundo tipo de procesado, audio a audio realizando la mediana.

La precisión es del 48,1%.

Los mejores resultados generales se consiguen con el segundo tipo de procesado y realizando la media. En ellos, la probabilidad de que un audio pertenezca a la clase de ELR que indica el perceptrón es de un 48,4%.

Utilizando el primer tipo de procesado los resultados son también ligeramente mejores con la media.

8.3 Error absoluto medio

Podría darse el caso de que un perceptrón acierte la clase con una probabilidad elevada pero que cometiera grandes errores cuando no. Si en lugar de predecir la clase correcta n predijera la $n+1$, sería menos grave que si predijera la $n+4$. Si bien esto se puede visualizar en la matriz de confusión, no es del todo automático; en especial en el caso del ELR en el que existen muchas clases.

Así, junto a la precisión, se evaluará también el error absoluto medio de cada perceptrón como parámetro adicional para determinar cuál es mejor.

Nótese que el error inherente de clasificar los valores reales de RT y ELR para que se ajusten a las clases no se tendrá en cuenta. Además, para realizar los cálculos, se asumirán como valores de RT y ELR los centrales del intervalo correspondiente a su clase. Por ejemplo, si la clase 1 de RT cubre el intervalo de 0,3 a 0,4 segundos, se tomará RT = 0,35 segundos.

8.4 Resumen de resultados

En las tablas 4 y 5 se encuentran los resultados principales. Se observa que tanto para estimar el RT como para estimar el ELR, el mejor perceptrón es el que utiliza el procesado de la modulación temporal. Además, realizar la media de las muestras en la salida, pertenecientes a un mismo audio resulta también lo más beneficioso para ambos.

Tipo de procesado	**Tipo de datos a la salida**	**Precisión (%)**	**Error Absoluto Medio (s)**
Procesado 1	Muestra a muestra	49,5	0,06
Procesado 1	Media	67,3	0,03
Procesado 1	Mediana	67,6	0,03
Procesado 2	Muestra a muestra	55,7	0,05
Procesado 2	Media	75,8	0,02
Procesado 2	Mediana	75,1	0,03

Tabla 4: Resumen de resultados para RT.

Tipo de procesado	**Tipo de datos a la salida**	**Precisión (%)**	**Error Absoluto Medio (dB)**
Procesado 1	Muestra a muestra	25,1	1,65
Procesado 1	Media	35,1	1,04
Procesado 1	Mediana	34,4	1,09
Procesado 2	Muestra a muestra	31,2	1,45
Procesado 2	Media	48,4	0,73
Procesado 2	Mediana	48,1	0,75

Tabla 5: Resumen de resultados para ELR.

9. Conclusiones

Este trabajo ha supuesto una primera aproximación al uso de redes neuronales para estimar parámetros acústicos de un audio grabado bajo determinadas condiciones de reverberación. Se ha conseguido entrenar un perceptrón multicapa que es capaz de predecir el RT y el ELR, aproximándose a los valores reales con cierta precisión. El mayor avance radica en que no se tengan que realizar mediciones *in situ* de la sala donde se realiza la grabación, con lo que se facilita su implementación en un abanico amplio de casos prácticos.

Existe una correlación negativa entre el RT y el ELR. Además, a la hora de generar las RIRs, resulta complejo conseguir valores altos de ELR en salas de dimensiones pequeñas.

El procesado que más beneficia el entrenamiento del perceptrón y que, consecuentemente, consigue mejores resultados, es el que emplea filtros de modulación temporal.

En cuanto al número de capas ocultas y neuronas del perceptrón, se ha concluido que 128 neuronas en una única capa oculta es la solución más eficiente. Emplear dos capas no mejora el rendimiento. Emplear 256 neuronas lo mejora ligeramente pero a costa de aumentar de forma desproporcionada el tiempo de entrenamiento.

Por último, cabe destacar que el perceptrón mejora sus predicciones de forma muy significativa si se aumenta el tiempo de audio considerado para generar un resultado. Es decir, utilizando la media de las probabilidades correspondientes a múltiples muestras se consiguen mejores resultados que si se toma una única muestra.

10. Líneas futuras

Existen muchos puntos en los que se podría continuar investigando para perfeccionar el modelo y poder implementarlo en aplicaciones reales. Algunas de estas mejoras y aplicaciones se mencionan a continuación:

- Ampliar el número de clases de salida para el parámetro ELR y, sobre todo, para el RT: con el desarrollo actual únicamente se pueden estimar valores de ELR de 1 a 19 decibelios y valores de RT de 0,3 a 0,68 segundos. Este rango no incluye todos valores que se podrían dar en la realidad de forma habitual. Habría por tanto que ampliar la base de datos y volver a entrenar los perceptrones.

- Añadir ruido a la base de datos: los perceptrones no están preparados para recibir audios ruidosos ya que no han sido entrenados para ello. Habría que añadir diversos tipos de ruido comunes a los audios de la base de datos y volver a entrenar los perceptrones.

- Introducir audios grabados en salas reales a la base de datos: todos los audios empleados han sido grabados en un entorno sin reverberación, y esta se ha simulado de forma artificial. Sería interesante contar con audios grabados en salas reales de los que conozcamos sus valores de RT y ELR. De esta forma se aumentaría la diversidad de los datos.

- Investigar otros tipos de procesado: se ha comprobado que el tipo de procesado tiene una gran influencia en el rendimiento de los perceptrones, por lo que interesaría evaluar más tipos, como por ejemplo, extrayendo las características con los Coeficientes Cepstrales en las Frecuencias de Mel.

- Entrenar un perceptrón que estime el RT y el ELR a la vez: como se vio en la Figura 6, existe una correlación negativa entre el RT y el ELR. Por ello, sería interesante investigar la creación de un perceptrón que estime de forma conjunta estos parámetros en lugar de tener dos diferentes.

- Implementarlo en una aplicación real para disminuir la reverberación: un ejemplo donde se podría utilizar el perceptrón es en una aplicación de grabado de audio con el ordenador. Se captaría el primer segundo de la grabación para procesarlo e introducírselo al perceptrón. Este estimaría los valores de RT y ELR que permitirían emplear un algoritmo de atenuación de la reverberación que se nutriera de estos parámetros. Así, una vez se tuvieran estos parámetros se podría aplicar el algoritmo al resto de la grabación, si es para tiempo real, o a la grabación completa, si es para almacenarla, de forma que se mejoraría la calidad del audio.

  Con el fin de mejorar la estimación de los parámetros, también se podrían captar más segundos del audio para introducir al perceptrón, en función de la importancia que suponga tener su estimación antes o después.

11. Bibliografía

[1] M. Valente, H. Hosford-Dunn, R.J. Roeser (2008). Audiology. Thieme. pp. 425–426.

[2] P.A. Naylor, N. D. Gaubitch (2010). Speech Dereverberation. Springer.

[3] E. Habets (2010). Speech Dereverberation Using Statistical Reverberation Models. En: P.A. Naylor, N.D. Gaubitch. Speech Dereverberation.

[4] N. Gaubitch, P.A. Naylor (2010). Subband Inversion of Multichannel Acoustic Systems. En: P.A. Naylor, N.D. Gaubitch. Speech Dereverberation.

[5] R.F. de Mello, M.A. Ponti (2018). Machine Learning, A Practical Approach on the Statistical Learning Theory. Springer.

[6] M.E. Celebi, K. Aydin (2016). Unsupervised Learning Algorithms. Springer.

[7] The Mathworks, Inc. Redes neuronales. Consultado en junio de 2019 en https://es.mathworks.com/discovery/neural-network.html

[8] J. Brownlee, Crash Course on Multi-Layer Perceptron Neural Networks. Consultado en junio de 2019 en https://machinelearningmastery.com/neural-networks-crash-course/

[9] U. Walkarn. A Quick Introduction to Neural Networks. Consultado en junio de 2019 en https://ujjwalkarn.me/2016/08/09/quick-intro-neural-networks/

[10] The Mathworks, Inc. Función de transferencia sigmoide. Consultado en junio de 2019 en https://es.mathworks.com/help/deeplearning/ref/logsig.html

[11] The Mathworks, Inc. Función de transferencia de unidades lineales rectificadas. Consultado en junio de 2019 en https://es.mathworks.com/help/deeplearning/ref/poslin.html

[12] The Mathworks, Inc. Función de transferencia *softmax*. Consultado en junio de 2019 en https://es.mathworks.com/help/deeplearning/ref/softmax.html

[13] The Mathworks, Inc. Choose a Multilayer Neural Network Training Function. Consultado en junio de 2019 en https://es.mathworks.com/help/deeplearning/ug/choose-a-multilayer-neural-network-training-function.html

[14] M.F. Meiller (1993). Neural Networks, Vol. 6, pp. 525-533.

[15] The Mathworks, Inc. Crossentropy. Consultado en junio de 2019 en https://es.mathworks.com/help/deeplearning/ref/crossentropy.html

[16] J.D. McCaffrey. Why you should use cross-entropy error instead of classification error or mean squared error for neural network classifier training. Consultado en junio de 2019 en https://jamesmccaffrey.wordpress.com/2013/11/05/why-you-should-use-cross-entropy-error-instead-of-classification-error-or-mean-squared-error-for-neural-network-classifier-training/

[17] J.S. Garofolo et al. TIMIT Acoustic-Phonetic Continuous Speech Corpus LDC93S1. Web Download. Philadelphia: Linguistic Data Consortium, 1993.

[18] J.B. Allen, D.A. Berkley. “Image method for efficiently simulating small-room acoustics”, *Journal Acoustic Society of America*, 65(4), p. 943, Abril 1979.

[19] E. Habets, “Room Impulse Response Generator”, Internal Report, 1-17, 2006.

[20] F. Xiong, S. Goetze, B. Kollmeier, B. T. Meyer, “Exploring Auditory-Inspired Acoustic Features for Room Acoustic Parameter Estimation From Monaural Speech”, *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 26, no. 10, pp. 1809-1820, 2018.

[21] D. Wang, G.J. Brown (2006). Computational Auditory Scene Analysis: Principles, Algorithms, and Applications. Wiley.

[22] S. Nandhini A. Shenbagavalli, "Voiced/Unvoiced Detection using Short Term Processing", *International Journal of Computer Applications*, vol. 4(2), pp. 39-43, ICIIECS-2014.

[23] F. Xiong et al, "Front-end technologies for robust ASR in reverberant environments—Spectral enhancement-based dereverberation and auditory modulation filterbank features", *EURASIP J. Adv. Signal Process*, vol. 2015, no. 70, pp. 1–18, 2015.

[24] The Mathworks, Inc. Choose Neural Network Input-Output Processing Funtions. Consultado en junio de 2019 en https://es.mathworks.com/help/deeplearning/ug/choose-neural-network-input-output-processing-functions.html

[25] T. Dau, B. Kollmeier, A. Kohlrausch, "Modeling auditory processing of amplitude modulation. I. Detection and masking with narrow-band carriers", *J. Acoust. Soc. Amer*., vol. 102, no. 5, pp. 2892–2905, 1997.

[26] Acoustics—Measurement of Room Acoustic Parameters, the International Organization for Standardization (ISO) Std. 3382, Agosto 2009.

[27] P. Kendrick, T.J. Cox, F.F. Li, Y. Zhang, J.A. Chambers, "Monaural room acoustic parameters from music and speech", *J. Acoust. Soc. Amer*., vol. 124, no. 1, pp. 278–287, 2008.

[28] V. Nair, G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines", *Proc. 27th Int. Conf. Mach. Learn.*, pp. 807–814, Junio 2010.