resumenes:

Transcripción musical mediante redes neuronales profundas:Autor

Jorge Donis del Álamo

Tutor/es

José Javier Valero Mas

Departamento de Lenguajes y Sistemas Informáticos

Jorge Calvo Zaragoza

Departamento de Lenguajes y Sistemas Informáticos

Primero se realiza un preprocesado de la señal de audio que consiste un downsampling a 22kHz y la posterior transformación a tiempo-frecuencia mediante la STFT[4]. Después se proceden a calcular distintas características.

Por una parte se calcula el Patrón Espectral (SP). Éste codifica el contenido tímbrico de la canción[5]. El SP se calcula ordenando (por bloques) la intensidad de las frecuencias. A partir del SP se calcula el DSP (Delta Spectral Pattern), que simplemente mide la variabilidad de los timbres. De manera análoga, se calcula, a partir del DSP el VDSP (Variance Delta Spectral Pattern).

Para codificar la estructura rítmica, se usa el Patrón Logarítmico de Fluctuación (LFP). Éste intenta medir la periodicidad de las frecuencias. Por ejemplo, en un compás simple de 4, con un tempo andante (80 negras por minuto), cada 1,33 segundos se podrá observar un aumento de intensidad en todas las frecuencias (en concreto, las frecuencias de las notas de los instrumentos como el bajo).

Después se calcula el Patrón de Correlación (CP). Éste es una matriz cuadrada en la que se mide la correlación entre cada una de las distintas frecuencias. Por ejemplo, si la batería suele tocar el hi-hat junto al bombo, se verá una relación entre esas frecuencias (lo cual es una característica representativa de esa canción).

Finalmente se calcula el Patrón de Contraste Espectral (SCP). Simplemente mide (por bloques) la distancia entre la intensidad de la frecuencia más intensa y la intensidad de la frecuencia menos intensa. En general, esto es

una métrica de la harmonicity[6].

En la Figura 1.2 podemos ver una representación de todas estas características:

Posteriormente se concatenan todas las características en un vector de 9448 dimensiones. Este vector es el que caracteriza a la canción. Dos canciones similares tendrán vectores similares.

El vector se puede usar directamente para la clasificación en géneros. En este artículo, se emplea una red neuronal para ello. La entrada es el propio vector y las salidas son las distintas categorías.

También se pueden comparar vectores para ver su similitud. Para ello se usa la distancia de Manhattan[7]..

1.4 Music Source Separation

Music Source Separation (MSS) es el campo de la MIR que consiste en la descompo- sición de una canción en sus correspondientes tracks[9] o incluso instrumentos. La Figura 1.3 muestra el resultado al cual se trata de llegar con Music Source Separation (MSS).

El primer paso es, como suele darse en MIR, representar de la señal en el plano tiempo- frecuencia (espectrograma). Después, existen varios métodos para hacer la separación.

Uno de ellos es la Non-negative Matrix Factorization (NMF). La entrada al problema es el espectrograma, que en definitiva es una matriz de intensidades de frecuencias $S_{t,n}$ donde t es el número de timesteps[10] y n es el número de distintas frecuencias. El problema que se intenta resolver es obtener dos matrices $D_{n,k}$ (diccionario) y $A_{k,t}$ (matriz de activación) tales que $S = DA$[11]. k vendría a ser el tamaño del diccionario. En general, se busca una $k < n$. Cada columna de I es un vector de k plantillas espectrales (cada plantilla, de longitud f). Tras la factorización, se puede apreciar en A los instantes t durante los cuales se activa cada una de las plantillas espectrales. En la Figura 1.4 se observa la factorización claramente. Si escogemos una k igual al número de notas (o notas MIDI), entonces estaremos haciendo AMT (ver Sección 1.6).

Con esto, ya tendríamos los intervalos temporales durante los cuales ciertas frecuencias se activan con una cierta intensidad. Si k = 5, tendríamos la separación para los tracks: vocals, bass, drums, guitar y others. Finalmente, queda extraer las frecuencias necesarias del espectrograma original. Una vez tenemos las frecuencias para cada track aisladas, queda realizar la transformación inversa frecuencia-tiempo para volver a obtener una señal de sonido.

Cabe decir que este proceso puede brindar unos mejores resultados si la canción ya está separada en formato estéreo, ya que la percusión y el bajo suelen encontrarse en el canal izquierdo y el resto de instrumentos en el derecho.

Las aplicaciones del MSS son evidentes. Cuando no se tiene acceso a los tracks originales[12], se pueden obtener de manera automática. Éstos pueden ser usados simplemente para can- tar karaoke, por entretenimiento. También puede servir en ámbitos educativos para enseñar canto. Otro ejemplo de aplicación sería aligerar la carga de un sistema de recomendación

musical

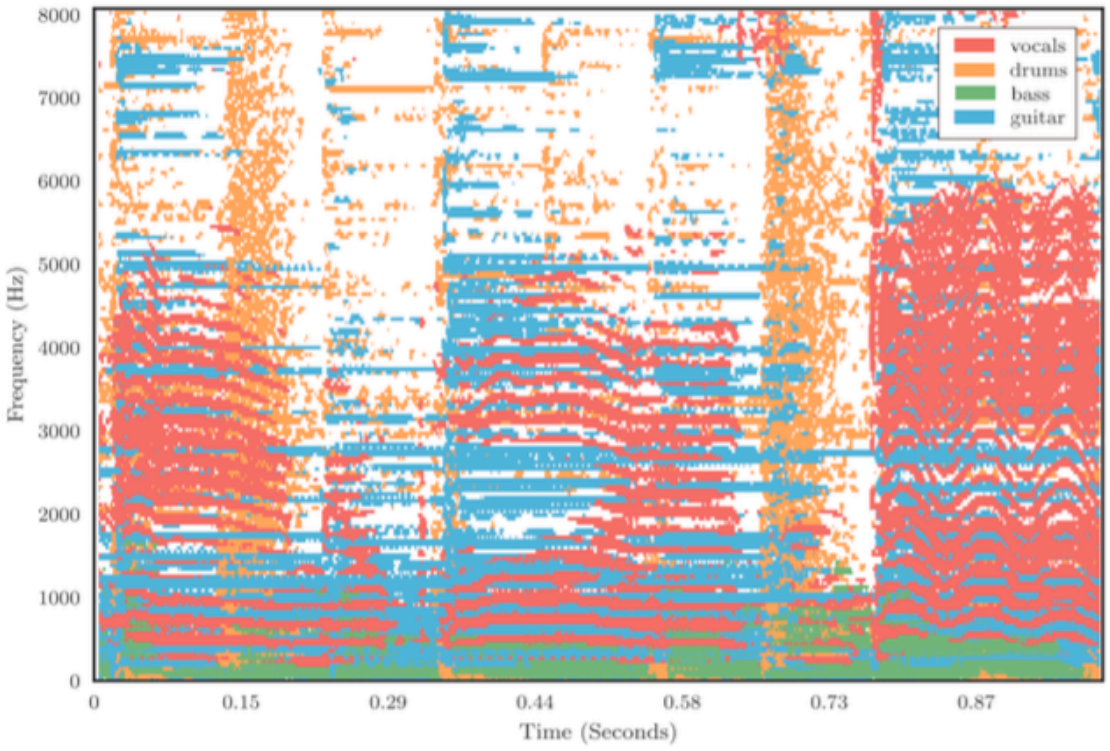


Figura 1.3: Descomposición de un espectrograma en sus correspondientes tracks (Cano y cols., 2019).

ELECCION DE LA VENTANA :

La elección de la función de ventana no es una tarea trivial (“Understanding FFT’s and Windowing”, s.f.). Cada función tiene sus características. Ciertas funciones nos permitirán resolver distintas problemáticas. En cualquier caso, hay que analizar el contenido frecuencial de la onda.

• Si la señal contiene interferencias de frecuencias muy distantes a la frecuencia de inte- rés, se usará una ventana en la que los extremos decrezcan rápidamente. La ventana de Hann es un ejemplo de este tipo de ventana.

• Si la señal contiene interferencias de frecuencias próximas, se usará una ventana en la que los extremos decrezcan lentamente. Por ejemplo, la ventana de Hamming.

• Si existen dos señales con frecuencias muy próximas, es interesante usar un función

de ventana cuyo pico central se muy estrecho. Por ejemplo, una ventana Gaussiana

$$w[n] = \exp(-1(n-N/2)2). 2\ \sigma N/2$$

• Si la amplitud de la frecuencia es muy pronunciada pero no es muy precisa (ocupa varias frecuencias), entonces se ha de escoger una función de ventana con un poco central amplio. Por ejemeplo, la ventana Plank-taper.

• Si el espectro frecuencial es denso (o plano), es mejor usar la ventana uniforme. La ventana uniforme es equivalente a no usar ninguna ventana.

La ventana de Hann es una buena elección en el 95% de los casos (“Understanding FFT’s and Windowing”, s.f.). Tanto la ventana de Hann como la Hamming son sinusoides y esto las dota de un pico central ancho que captura muy bien la frecuencia original.

Con esto, el proceso de Short-Time Fourier Transform (STFT) quedaría resumido por la Figura 2.9. La entrada al problema es la señal compleja. La salida es el espectro tiempo-frecuencia. Los parámetros que podemos establecer son:

• Función de ventana w[]. Explicada anteriormente, sirve para suavizar los extremos. Multiplicaremos la onda compleja en el intervalo deseado por este

valor.

• Número de muestras por bloque n. Como explicábamos antes, aplicaremos DFT sobre unas pequeñas porciones de la señal original. Estos bloques o frames están com- puestos por n muestras cada uno. A partir de ahora, en general, al tratar de valores discretos, ya no usaremos el tiempo como magnitud, sino el número de muestras[10]. En caso de que el número total de muestras de la señal no sea divisible por este valor n, se suele emplear zero-padding[11] para que cada frame llegue hasta este valor.

• Tamaño de ventana M. Es el número de muestras que serán multiplicadas por la función de ventana. El análisis de este algoritmo se vuelve muy complejo si M/= n. Sin embargo, M puede ser menor que n. Cuando multiplicamos menos muestras por la función de ventana que el tamaño del frame, el resto valdrán 0.

• Hop-length R. Este parámetro nos permite describir cómo queremos que se solapen los frames. R es el número de muestras entre el comienzo de cada frame. R siempre es menor que n si queremos que haya solape. Cuando R = n, no hay ningún solape.

Por esto, cada vez que hemos empleado la DFT, podemos emplear un algoritmo mucho más eficiente conocido como Fast Fourier Transform (FFT). Este algoritmo no repite k veces el mismo sumatorio, porque hace uso de una matriz en la cual va guardando resultados previamente calculados (programación dinámica iterativa).

Lo más relevante de la fórmula anterior es que toma dos parámetros: la frecuencia k y el frame m. De la misma manera que la DFT tiene dos dimensiones: frecuencia e intensidad; la STFT tiene tres dimensiones: frecuencia, tiempo[12] e intensidad. Estas tres dimensiones son comúnmente representadas mediante dos ejes y un color, dando lugar al famoso espectrograma (ver Figura 2.11).

4.1 Red Neuronal

La entrada es la imagen espectrograma tiempo-frecuencia de la muestra de audio (ver Figura 2.11). En nuestro caso, el corpus[3] cuenta únicamente con ficheros MIDI. El primer paso es usar algún instrumento virtual para generar las señales de audio. En este caso, con la herramienta timidity generamos los fichero con codificación WAV usando un piano virtual con una tasa de

muestreo de 44.1KHz. Después procedemos a calcular la Short-Time Fourier Transform (STFT) con hop_length de 512 muestras y window_length de 2048 muestras. La función de ventana es la de Hann. A la hora de graficar el espectrograma usamos una escala lineal en el eje Y. El resultado final se puede ver en la Figura 4.2. El único post-procesado que se hace de esta imagen es un escalado (la altura tiene que ser fija) y una reducción a un único canal de color (blanco y negro).

Sobre esta imagen aplicaremos filtros de convolución (ver Figura 4.3). Los filtros (o ker- nels) de convolución son matrices que son aplicadas sobre la imagen. Cada píxel se convierte en una combinación lineal de sus píxeles adyacentes, en la que los coeficientes son los valores de la matriz de convolución. Estos coeficientes son aprendidos durante el backpropagation. Cabe establecer el número de filtros y el tamaño del kernel.

Tras cada filtro de convolución se aplica un pooling. El pooling es sencillamente un agrupa- miento de los píxeles. Normalmente se escoge el valor máximo de cada kernel, pero también se puede aplicar la media o incluso el valor mínimo. Cabe destacar que cuando se aplica una capa de pooling se reduce la dimensionalidad.

Tras aplicar varias capas de convolución y pooling, en definitiva, la red está aprendiendo características sobre la "forma" de la imagen. Y con las capas de pooling se está comprimiendo esta información. Al final, se puede entender que la imagen original ha sido segmentada y sobre cada segmento se ha obtenido un vector de características.

Sobre cada vector de características se aplica una capa recurrente. En este caso dos redes Long Short-Term Memory (LSTM). Las redes recurrentes son redes cuya entrada depende de su propia salida (ver Figura 4.4). Son empleadas con datos secuenciales de longitud indeterminada, como es el caso de las señales de audio, o las imágenes[4]. En este caso, sirve para introducir una dimensión temporal al problema de aprendizaje. De hecho, emplean el algoritmo de backpropagation through time para minimizar la función de error. En definitiva, las redes recurrentes tienen un estado interno que es un alias para la memoria. Una red Long Short-Term Memory (LSTM), cuando analiza el segmento (frame) t_4 tiene en cuenta las salida de la red t_3, que a su vez depende de la t_2... Además, se puede conectar la salida de la red del frame siguiente a la propia, con lo que se obtiene una red LSTM bidireccional. Éstas tienen memoria tanto de los frames anteriores

como de los siguientes.

Finalmente, se usa una capa densa normal para proporcionar la salida softmax para cada categoría. Las categorías son el ground-truth, es decir, las etiquetas válidas. Son la transcrip- ción a notación simbólica. En este caso hemos acabado escogiendo una notación apodada semantic (ver Figura 4.5). Esta notación tiene un total de 1781 tokens distintos. Hay que tener en cuenta que se tienen que codificar todas las notas con todas las alteraciones y duraciones posibles. Además de los silencios, los compases y las claves. Para agilizar el proceso de traducción entre un string como clef-G2 y el índice del token correspondiente (387, en este caso), se emplea un clase auxiliar SemanticTranslator (ver código 4.1). Esta clase tiene dos estructuras de datos, un array y un diccionario. El array tiene como claves los tokens (strings) y como valor el índice. De esta manera, usando una tabla hash se puede traducir en tiempo constante O(1). En el caso del array, el índice es el número de token y el valor es el propio token. Estas dos estructuras de datos están serializadas mediante pickle y guardadas en memoria secundaria.

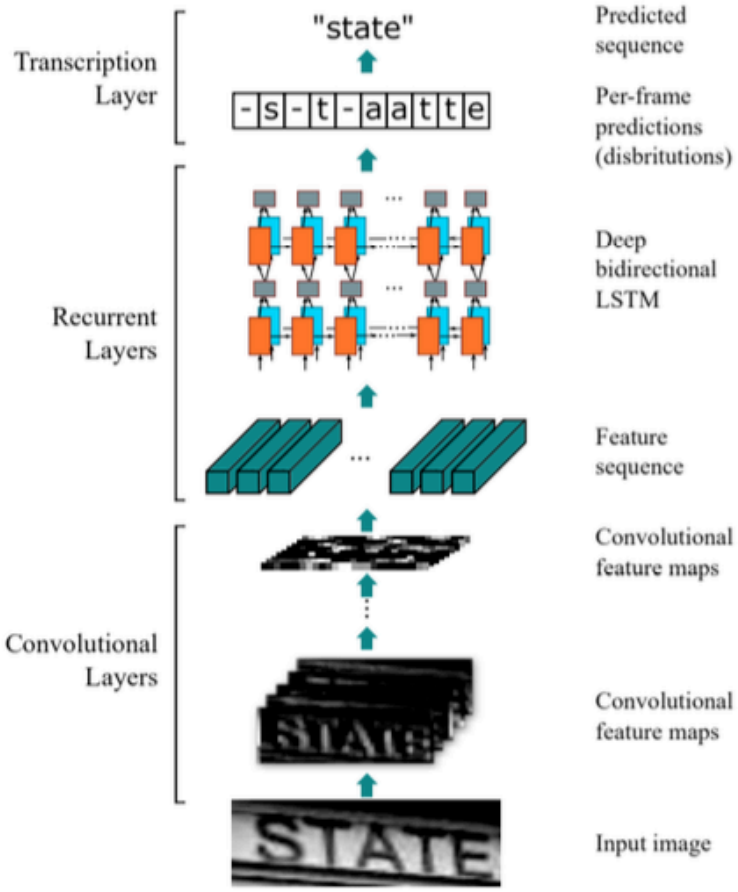


Figura 4.1: Arquitectura de una red CRNN (Convolutional Recurrent Neural Network).

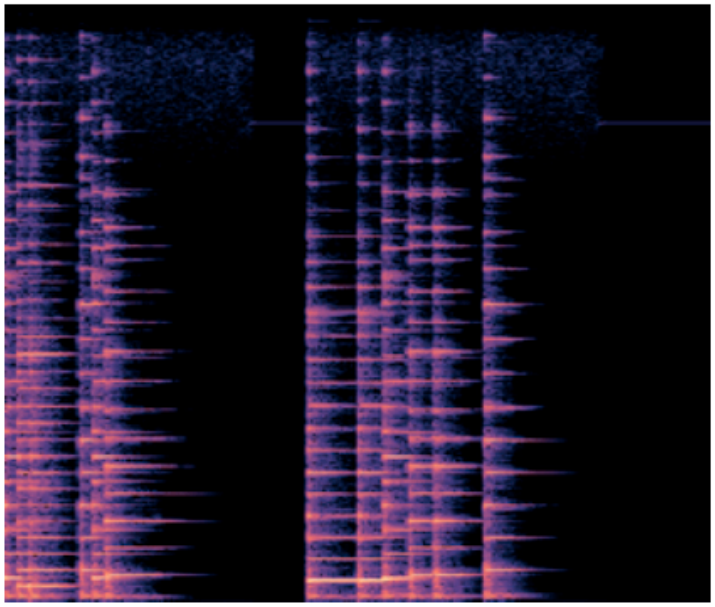


Figura 4.2: Ejemplo de espectrograma sintético. Se puede apreciar cómo apenas existen frecuencias fuera de los parciales. Es decir, hay poca *inharmonicity* (al tratarse de un instrumento sintético).

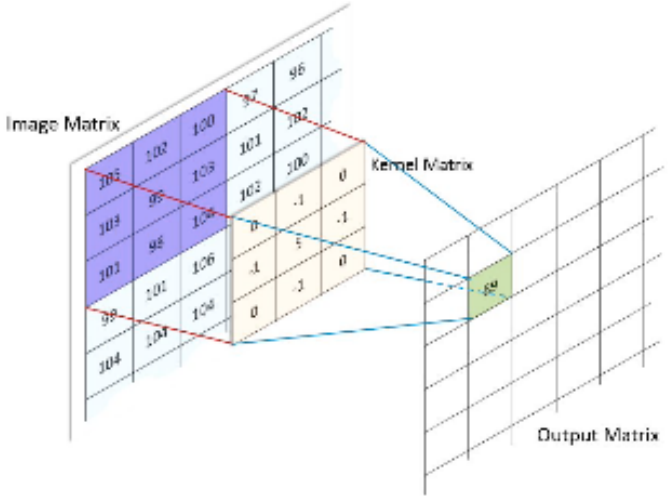


Figura 4.3: Ejemplo de aplicación de un *kernel* de convolución.

Con esto, la salida de la CRNN tendrá 1781 dimensiones para cada frame t_i. Cada frame se corresponderá con las características de una región rectangular de los píxeles del espectrograma. El tamaño de esta región depende del factor de reducción. Éste depende a su vez del número de capas de pooling y del tamaño de las mismas[5]. En la arquitectura que se presenta

se usan espectrogramas de es 256 píxeles de alto y de ancho variable. En la última capa de convolución se emplean 128 kernels. Con esto cada frame t_i tiene asociado un vector de $256/2^3 * 128 * 2^3 = 4096$ características que se corresponden a una región de de 8 píxeles de ancho por 256 de alto. Esta es la entrada a cada nodo de la LSTM. La tabla 4.1 muestra la arquitectura en detalle. Ésta incluye capas de dropout para mejor generalización y capas de batch normalization para acelerar el aprendizaje.

A la salida de la Convolutional Recurrent Neural Network (CRNN) tenemos un vector de predicciones softmax para cada token. A continuación explicaremos la métrica de error empleada para que el modelo apCambio de espectrograma

El primer cambio que pensamos que podría mejorar la transcripción fue el espectrograma. La Figura 5.2 muestra los cambios realizados. En concreto:

1. Se cambia la altura de la imagen de 256 a 192 píxeles (perdiendo resolución).

2. Se establece un umbral mínimo de intensidad de -70 (valor de intensidad de píxel propio de matplotlib). Si no se supera este umbral, el píxel pasa a ser negro. También se establece un umbral máximo de 8.

3. Se reduce el rango de frecuencias. Para abarcar las 88 notas del piano virtual, se usa un rango de entre 27.5Hz (A0) y 3520Hz (A8).

4. Se cambia a un hop_length de 128 samples.

5. Se establece un tamaño de ventana y un número de muestras por bloque (n) de 1024
(ambos).

6. Se emplea una escala de Mel en la dimensión frecuencial.

La escala de Mel (de la palabra melodía) es una escala perceptiva de alturas basada en experimentos con oyentes. En estos experimentos se intenta establecer una escala tal que todas las frecuencias suenen equidistantes a oídos humanos. Existen varias escalas de Mel (según los experimentos). La fórmula empleada por librosa para pasar de f hercios (Hz) a m mels es la siguiente:

$$m=2595\log_{10}(1+\ ^{f}\ )\ 700$$

Podemos apreciar como la escala es logarítmica. A primera vista, parece que la Subfigura 5.2b presenta varias ventajas frente a la Subfigura 5.2a. Por un lado, se usan menos fre- cuencias. Esto ayuda a eliminar información que podemos considerar como redundante. Cabe recordar que estamos tratando con transcripción de músicas monofónicas. Es decir, sólo va a existir una frecuencia fundamental en cada instante. El espectrograma de la derecha cla- ramente muestra como se eliminan muchos harmónicos que son irrelevantes. Por otro lado, esta escala de Mel nos ayuda a resaltar las frecuencias más graves. Esto es idóneo, ya que las frecuencias más graves son las fundamentales, que son las que determinan la altura

6 Conclusiones

El modelo de transcripción presentado muestra claramente que es posible traducir una señal de audio en una notación simbólica musical. Por simplicidad, en este caso hemos trabajado con música monofónica, pero se puede extraer que este modelo, con ciertos ajustes, podría brindar buenos resultados para música polifónica.

Además, se demuestra que una única red neuronal puede, en un solo paso, transcribir audio en notación simbólica (modelo end-to-end). Esto es gracias al método de entrenamiento CTC (Connectionist Temporal Classification). Podemos concluir que este método es, en general, una buena herramienta para resolver problemas de etiquetado de secuencias, como Optical Character Recognition (OCR)[1], Automatic Speech Recognition (ASR) o Handwritten Text Recognition (HTR).

Más en detalle, podemos comprobar que el espectro frecuencial es una buena codifi- cación de la información musical para Automatic Music Transcription (AMT). En concreto, la Short-Time Fourier Transform

(STFT) y el uso de imágenes de espectrogramas parecen ser una buena representación gráfica de las muestras de audio.

Aún con todo, se desprenden algunas conclusiones aparentemente ilógicas de los experimen- tos. Parece sorprendente que la corrección a posteriori de las barras de compás no mejore el error de validación. Aún más sospechoso es que el uso de una escala de Mel no brinde mejores resultados. Sin embargo, el funcionamiento de las redes neuronales no deja de ser un tanto obscuro y en ocasiones es difícil extraer conclusiones lógicas cuando se trabaja con ellas[2].

Finalmente me gustaría recalcar un par de aspectos sobre el Music Information Retrieval (MIR). Este es un campo de investigación plenamente activo (aunque algo reducido). La investigación sigue activa precisamente porque aún estamos muy lejos de llegar a resultados prácticos. Sin embargo, quisiera destacar la relevancia y utilidad que puede llegar a tener la MIR en la educación artística y en la música en general. Sin duda, un futuro en el que las máquinas compusieran música o ayudaran a componerla sería apasionante.

DE ESTE TRABAJO PODEMOS UTILIZAR LAS EXPLICACIONES DE LAS REDES NEURONALES USADAS,

ME GUSTARIA IMPLEMENTAR LO DE LA SEPARACION EN TRACKS, ESTO PODRIA SERVIR PARA VER SI PASARON UN AUTO Y UNA MOTO JUNTAS?

— —
— —

Trabajo Fin de Grado

RECOLECCIÓN Y

RECONOCIMIENTO DE EVENTOS

DOMÉSTICOS MEDIANTE SENSORES DE SONIDO EN IOT

Grado en Ingeniería InformáticaEscuela Politécnica Superior de Jaén

Alumno: José Manuel Vílchez Chiachío

El enfoque basado en el aprendizaje profundo más común para

la clasificación de sonidos es convertir el archivo de audio en una imagen y luego usar una red neuronal para procesar la imagen. Mostafa et all **[56]** realiza la clasificación de la música utilizando redes neuronales probabilísticas con resultados satisfactorios. La mayoría de los enfoques de clasificación sólidos utilizan el reconocimiento de patrones supervisado. Sin embargo, Zhang y Schuller **[57]** expresan el problema de que el etiquetado manual de conjuntos de datos es muy costoso y recomiendan el aprendizaje semi-supervisado como una mejor solución. McLoughlin et **[58]** afirma que la clasificación del sonido en entornos ruidosos realistas es un desafío y proponen una red neuronal profunda como una solución viable. Piczak y Zhang et **[59]** transmiten la idea de que las redes neuronales convolucionales tienen las mejores tasas de precisión en el análisis de espectrograma.

Según lo resumido por Chachada et all **[60]**, hay tres formas amplias de procesar sonidos ambientales con fines de clasificación:

1. Framing-baseddondelasseñalesdeaudioseseparanenframesusando una ventana de Hamming. Luego, las características se extraen de cada frame y se clasifican por separado.

2. Procesamiento basado en sub-framing donde los frames se subdividen aún más y cada frame se clasifica en base a la votación mayoritaria de los sub- frames.

3. Procesamiento secuencial donde las señales de audio se dividen en segmentos de típicamente 30 ms con un 50% de superposición. El clasificador luego clasifica las características extraídas de estos segmentos.

EN ESTE TRABAJO HAY RESULTADOS MUY IMPRESIONANTES!

DE AQUI SACO LA IDEA DE TRABAJAR CON LA VENTANA

DE HAMMING.

Detección de vehículos en circulación mediante visión artificial y redes neuronales convolucionales

Detection of vehicles in circulation
by artificial vision and convolutional neural networks

Fernando Pérez Gutiérrez

MÁSTER EN INGENIERÍA INFORMÁTICA. FACULTAD DE INFORMÁTICA UNIVERSIDAD COMPLUTENSE DE MADRID

ACA DETECTAN VEHICULOS PERO POR IMAGENES NO POR AUDIOS

TRABAJO FIN DE GRADO

PREDICCIÓN DE SONIDOS USANDO HARDWARE

Autor: Blanca Lluch Ponce Director: Marc Pomar Torres

Madrid, 2020

3.2 Arquitectura VGGish

VGGish es una red neuronal creada con TensorFlow formada por 11 capas como se muestra en la figura 13. Estas 11 capas tienen en común varios parámetros. Entre ellos los pesos y los sesgos iniciales. Estos dos parámetros son de los mas importantes en cuanto al ritmo de aprendizaje. Al pasar un vector de salida como entrada de la siguiente capa, se le aplican a ese vector los pesos, y se pasan por una capa de activación junto con el sesgo. [37] Para los pesos se aplica un inicializador que genera valores con una distribución normal con una desviación estándar de 0.01. Para el sesgo se aplica un inicializador de ceros. Además de estos dos parámetros también tienen en común la capa de activación nombrada antes que en este caso se utiliza la capa ReLu (Rectified Linear Units). Esta consiste en aplicar los pesos y el sesgo a la salida de la capa anterior y pasar los valores como entrada a la siguiente capa, modificándolos únicamente si son valores negativos, en este caso estos valores los pondría a cero.

Entre estas 11 capas, aparecen la capa de convolución 2D, Operación de agrupación máxima 2D, repetición, aplanar y capa de conexión. Las capas de convolución 2D (conv2d) tienen un tamaño de *kernel* de 3x3, de relleno (*padding*) [?]*SAME*[?] y de zancada *(stride) 1* . El relleno [?]*SAME*[?] se utiliza para añadir ceros en los márgenes y no variar las dimensiones del vector de entrada. El efecto que tiene la zancada es sobre como aplicar el *kernel* al vector, 1 es el predeterminado. Con esta capa se genera un mapa de características, por medio de operaciones de productos escalares y sumas entre el *kernel* y los valores que este filtrando en ese momento.

Las capas de operación de agrupación máxima (max_pool2d) tienen un tamaño de *kernel* 2x2, un relleno igual que la capa de convolución y una zancada de 2. Con esta capa se elimina la influencia que pueda llegar a tener la posición de cada característica de la entrada. Consiste en resumir los datos en el valor más activo y presente entre todos los de la entrada. [38]

Las capas de repetición (*repeat)*, aplican la misma capa con los mismos argumentos tantas veces como se le indique, en este caso 2 veces. La capa de aplanar (*flatten)* consiste en aplanar la entrada sin variar el tamaño de lote (*batch size*). El tamaño de lote consiste en el numero de elementos del *dataset* que se entrenan a la vez. El

numero de iteraciones es la cantidad de lotes necesarios para completar el entrenamiento de la red con todo el *dataset*. En el caso de VGGish el *batch size* es 100. Por último, la capa de conexión (*fully_connected)* crea una matriz de pesos que multiplica por la entrada y reduce la salida al numero de características que ha de tener la salida.

En el caso de querer conectar esta red a otra para clasificar los audios sería necesario añadir una capa de activación no lineal entre las dos redes neuronales conectadas. Se utiliza una capa de activación no lineal porque produce un mayor cambio entre la salida de una red y la entrada de otra que es esencial para que la red modele y aprenda con información más compleja. [39]

ESTO SE PODRIA IMPLEMENTAR

– –
– –
– –
– –
– – – – – – – – – – – – – – – – – –

Trabajo Fin de Grado
Grado en Ingeniería de las Tecnologías de Telecomunicación

Clasificación automática de sonidos utilizando aprendizaje máquina

Autor: Patricio Rodríguez Ramírez Tutor: Francisco José Simois Tirado

2.3. Según las características del sonido

Todas estas citas anteriores tienen como punto en común su arquitectura basada en redes neuronales. A continuación, vamos a introducir otros trabajos centrándonos en las características del audio que le pasamos al clasificador. Este punto también lo desarrollaremos en el apartado siguiente.

Uno de los primeros trabajos en el área del reconocimiento de audio fue el de Aucouturier et al. [27] (2007). En él los autores extraían los Coeficientes Cepstrales en la Frecuencia de Mel (MFCCs) a los audios y utilizaban un modelo mixto gaussiano (GMM) para clasificar. Su trabajo dio muy buenos resultados. Sin embargo, el trabajo de Lagrange et al. [28] (2015) demostró que el resultado estaba sesgado a un set de datos con muy poca variabilidad. Lagrange et al. aplicaron nuevos *datasets* con mayor variabilidad al modelo y los resultados obtenidos no fueron tan excepcionales.

Una opción es calcular una variedad de características al sonido. Eronen et al. [29] (2006) calcularon los MFCCs, la tasa de cruces por cero, el ancho de banda, etcétera. Para la clasificación emplearon el modelo oculto de Márkov (HMM). Geiger et al. [30] (2013) emplearon 13 características cepstrales, 35 características espectrales, 6 características de energía y 3 características relacionadas con la voz. Para la clasificación usaron una máquina de soporte de vectores (SVM).

Phan et al. [31] (2016) emplearon como características los Coeficientes Cepstrales en la Frecuencia Gammatone (GFCCs) y como clasificador una máquina de soporte de vectores. Por su parte, Roma et al. [32] (2013) emplean como características los coeficientes cepstrales en la frecuencia de mel más el cálculo de parámetros de recurrencia en los audios.

Una característica de entrada al sistema de clasificación puede ser el histograma de gradientes orientados (HOG). Bisot et al. [33] (2015) obtienen el histograma de gradientes orientados del espectrograma logarítmico de frecuencia del extracto de audio. Utilizan un clasificador basado en una máquina de soporte de vectores. Rakotomamonjya et al. [34] (2015) obtienen el histograma de gradientes orientados del espectrograma de la transformada de Q constante (CQT). Emplean varios sets de datos para su experimentación, con distintos clasificadores y resultados.

Cauchi et al. [35] (2013) desarrollaron un modelo en base a la factorización no negativa de matrices (NMF) para establecer las similitudes de los audios grabados en estaciones de tren. Bisot et al. [36] (2016) también emplearon la factorización no negativa de matrices para su problema de clasificación de sonidos ambientales.

Benetos et al. [37] (2012) usaron una variación del análisis de componentes principales (PCA) para extraer las características de los audios y entrenar su modelo. Lee et al. [38] (2013) utilizan una máquina estricta de Boltzmann dispersa (RBM) para obtener las características de entrada. Salamon et al. [39] (2015) calcularon los espectrogramas de mel, después aplicaron análisis de componentes principales y finalmente emplearon el algoritmo *k-means*

esférico (SKM) para extraer las características. También estudiaron este procedimiento de manera no supervisada. [40] (2015).

5

6 Estado del arte

Todos los trabajos citados nos han demostrado que existen muchas maneras de abordar las tareas de clasificación de audio ambiental. Para finalizar este extenso capítulo sobre el estado del arte nos vamos a centrar en el trabajo de Salamon et al., que han experimentado diversos métodos con un set de datos de sonidos ambientales.

En el trabajo [41] (2014) los autores presentan la taxonomía de los sonidos urbanos y crean el *dataset* con el que van a trabajar. Las características de entrada al sistema son los coeficientes cepstrales a la frecuencia de mel y el clasificador se basa en una máquina de soporte de vectores. En el trabajo [40] (2015) visto con anterioridad emplean como entrada al sistema el resultado del algoritmo *k-means* esférico aplicado a los espectrogramas. El clasificador se basa en el algoritmo de árboles aleatorios. En el trabajo [39] (2015) también emplean como entrada la salida del algoritmo *k-means* esférico aplicado a los espectrogramas. El clasificador se basa en una máquina de soporte de vectores. Finalmente en el trabajo [42] (2017) emplean como entrada al sistema los espectrogramas de los extractos de audio y el clasificador es una red neuronal convolucional profunda.

Los resultados de sus trabajos van en progresiva mejora. Mientras que el primer trabajo se considera la base y alcanza una precisión de más del 68%, los dos siguientes con el algoritmo *k-means* esférico alcanzan precisiones del 74% y 75%. La primera implementación con la red neuronal convolucional les deparó un resultado del 73% de precisión. Sin embargo, al aplicar un algoritmo de aumento de datos la precisión alcanzó cerca del 79%. Finalmente implementaron el modelo de Piczak [6] (2015) con la misma base de datos. El resultado de este modelo fue del 73%.

En la Figura 2-2 el punto rojo indica el resultado medio del modelo y las barras nos indican la desviación estándar de la medida. Fijándonos en la arquitectura basada en red neuronal convolucional vemos como, en el caso de no aplicar el algoritmo de aumento de datos (zona a la izquierda de la línea de puntos), la precisión media es menor que en los modelos que emplean *k-means* esférico. Sin embargo, la desviación estándar de la precisión es menor en el caso de la CNN ya que sus barras tienen un menor tamaño en comparación con los modelos que emplean SKM.

En el caso de aplicar un algoritmo de aumento de datos (zona a la derecha de la línea de puntos) vemos como la precisión media de la CNN aumenta con respecto al modelo SKM. A pesar de esto, la desviación estándar de la medida es mayor en el caso de la red neuronal convolucional, ya que sus barras ocupan un mayor tamaño en comparación con el modelo que emplea *k-means* esférico.

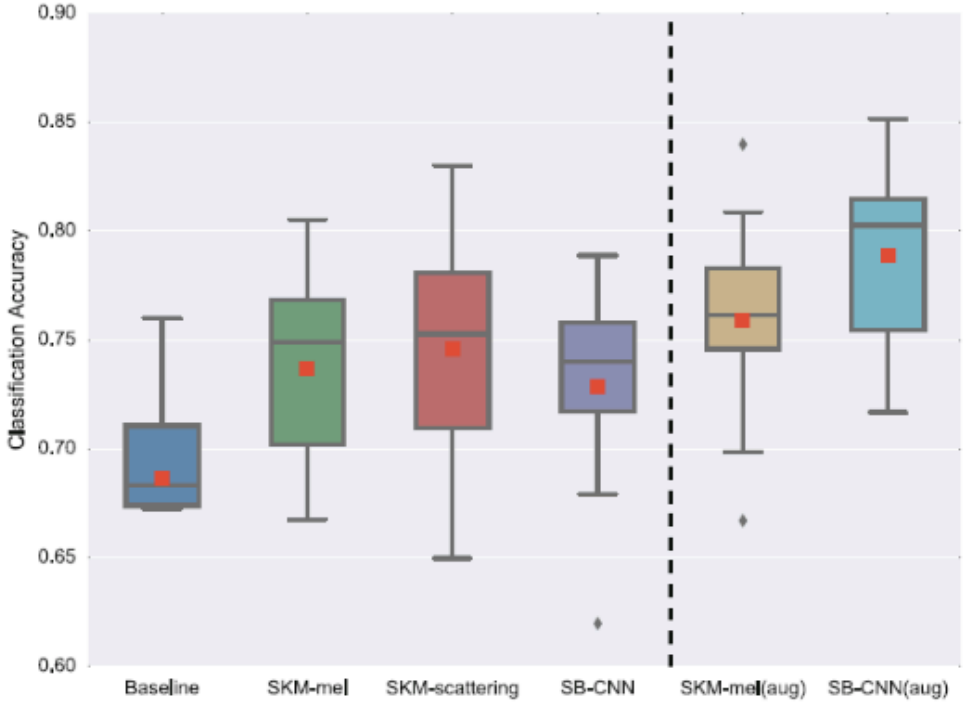


Figura 2-2. Resultados de precisión de los distintos modelos de Salamon et al.

3.1.3 Espectrograma

Para obtener una representación tiempo-frecuencia tenemos que hacer la DFT de la señal cada cierto tiempo. En el eje de abscisas representamos el tiempo transcurrido y en el eje de ordenadas las frecuencias. El nivel de intensidad de esa frecuencia se representa en el espectrograma, donde las zonas más oscuras indican mayor nivel de decibelios y las zonas más claras menor nivel. En la Figura 3-3 tenemos un ejemplo de espectrograma.

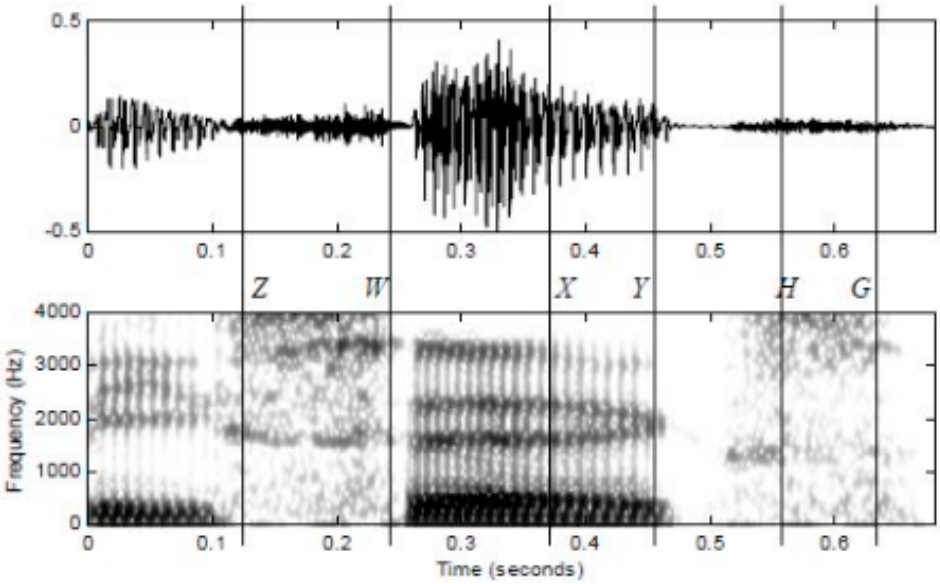


Figura 3-3. Audio con su espectrograma

Este espectrograma representa el eje de ordenadas, es decir, las frecuencias, de manera lineal. Sin embargo, a veces queremos obtener información en ciertas bandas de frecuencia. Para ello hacemos pasar el resultado de la FFT por un banco de filtros. Estos filtros pueden estar distribuidos de diversas maneras.

La escala de Mel está relacionada con la percepción auditiva humana. Estudios como el de Stevens et al. [44] (1937) observan que el oído humano actúa como un filtro y que concentra su actividad en ciertas partes del espectro de frecuencia. Esta característica se aproxima por el banco de filtros de Mel (Figura 3-4), donde existen más filtros y con bandas más estrechas en las zonas de baja frecuencia que en las zonas de alta frecuencia, donde los filtros son pocos y de banda ancha.

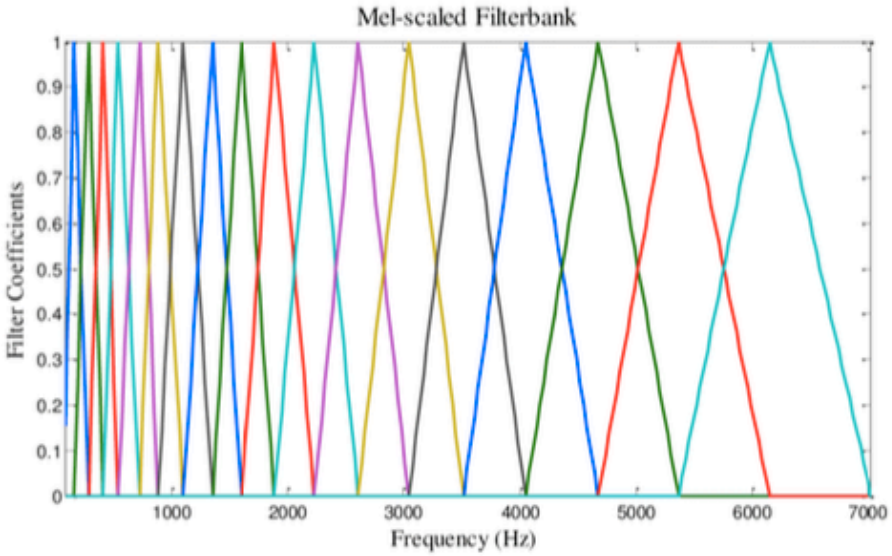


Figura 3-4. Banco de filtros de Mel

Este procedimiento lo siguen distintos tipos de escala, como las bandas críticas (Zwicker et al. [45] (1980)) o la transformada *Constant-Q* (Brown et al. [46] (1991)). En este último caso la escala de frecuencias no está uniformemente espaciada, sino que su distribución es geométrica. Otro ejemplo son los filtros *Gammatone* (Patterson et al. [47] (1992)). Estos filtros son más anchos que los de Mel, pero están distribuidos de manera parecida: en las frecuencias bajas tenemos más filtros que en las frecuencias altas. Se muestran en la Figura 3-5.

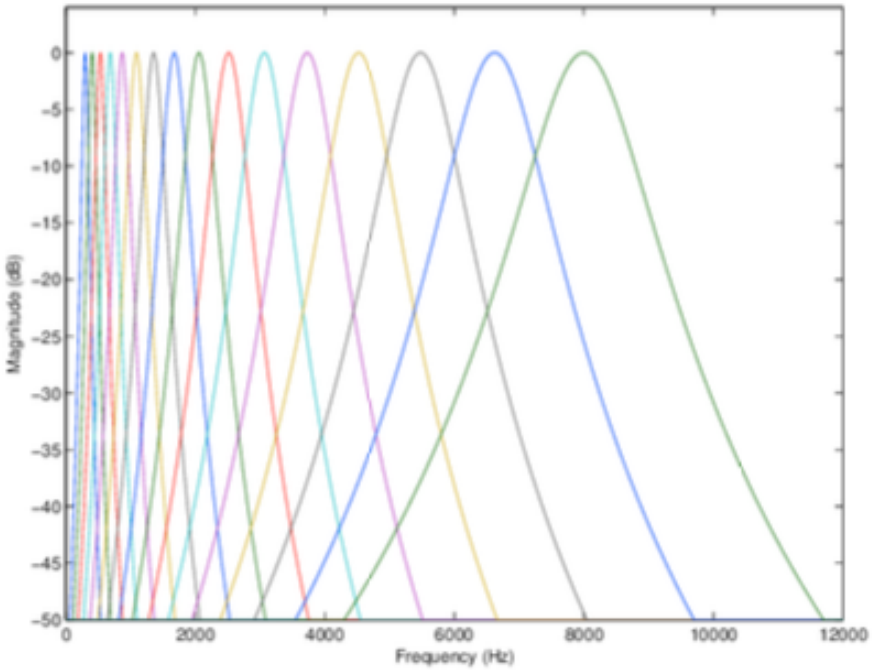


Figura 3-5. Banco de filtros *Gammatone*

Una vez elegido el filtro tendremos un resultado distinto en el espectrograma. En la Figura 3-6 se muestra (a) la forma de onda del audio, (b) el espectrograma de frecuencia lineal, (c) el espectrograma de frecuencia de Mel y (d) el espectrograma de la transformada *Constant-Q*.

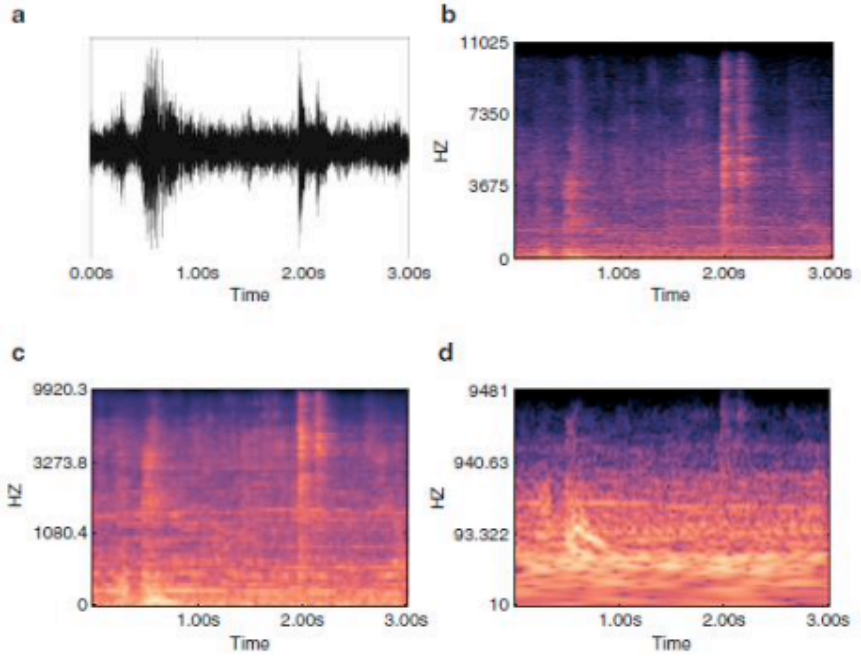


Figura 3-6. Distintos ejemplos de espectrogramas

Estas representaciones visuales del audio pueden ser utilizadas como datos de entrada a nuestro clasificador. Sin embargo, como vimos en el capítulo anterior se suele aplicar un paso más y se calculan unos valores numéricos, llamados coeficientes, para que sean estos los que se pasen como entrada al sistema y no la imagen del espectrograma completo.

Los coeficientes más frecuentes a la hora de trabajar con audio son los coeficientes cepstrales a la frecuencia de Mel (MFCC) (Peltonen et al. [48] (2002), Valero et al. [49] (2012)). Para computar estos coeficientes tenemos que calcular el logaritmo de cada ventana de espectrograma a las que le hicimos la DFT, como vimos anteriormente. Seguidamente a cada ventana le aplicamos la Transformada Discreta del Coseno (Discrete Cosine Transform, DCT). Los valores de las amplitudes resultantes son los denominados coeficientes cepstrales a la frecuencia de Mel.

Normalmente se almacenan los 13 primeros MFCC de cada extracto de espectrograma, ya que los siguientes no contienen información relevante sobre el audio. El conjunto de vectores de MFCC de todas las ventanas del audio es el que se pasa como parámetro de entrada al modelo. En este caso los datos de entrada del modelo serían menos pesados si los comparamos computacionalmente con un espectrograma completo de un audio. A su vez existen más alternativas en cuanto al cálculo de coeficientes se trata, como por ejemplo los coeficientes cepstrales *Gammatone* (GFCC) (Phan et al.

[50] (2016), Valero et al. [51] (2012)).

Cross-validation

La validación cruzada tiene el mismo objetivo que las capas de *Dropout* y *BatchNormalization*, evitar el sobreajuste. Métodos de *cross-validation* hay muchos, pero todos se basan en la misma idea. Hacen particiones del *dataset* en entrenamiento y test y entrena el modelo, todas las veces necesarias hasta que todas las muestras de la base de datos hayan sido tomadas como test. Así nos aseguramos de que la precisión del modelo no depende de la partición aleatoria de datos que hayamos hecho, ya que todos los datos van a formar parte del entrenamiento y del test.

Uno de los algoritmos más comunes para implementar validación cruzada es el llamado *k-fold* (Krishni [65] (2018)). Consiste en dividir la base de datos en *k* lotes de datos. El entrenamiento se realizará con *k*-1 lotes de datos y el test se hará con un lote de datos solo. Se guardan los resultados de precisión y error de esa evaluación

y se repite el entrenamiento con un lote de test distinto y los restantes para entrenamiento.

El algoritmo *k-fold* no tiene en cuenta el número de muestras de cada clase que existan en la base de datos total. Nosotros buscamos que la separación en datos de test y entrenamiento se haga de la manera más homogénea posible. Para ello se implementa el algoritmo *stratified k-fold*, que usaremos en nuestros experimentos. Este método sí tiene en cuenta el número de muestras totales de cada clase y las separa según el número de subdivisiones que le hayamos especificado. Un ejemplo visual lo podemos encontrar en la Figura 3-18.

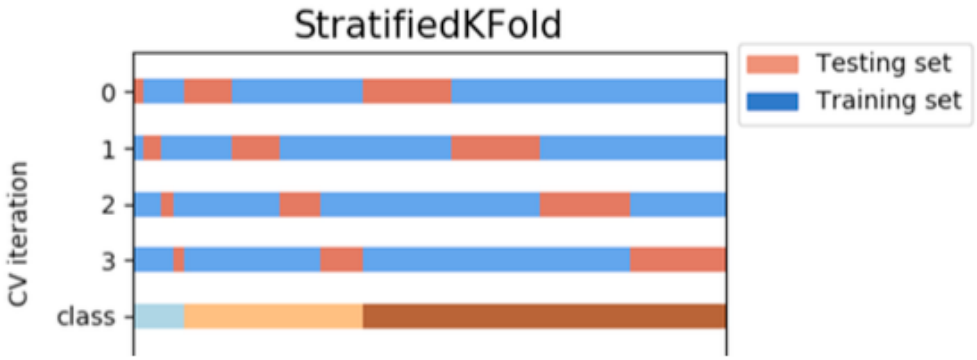


Figura 3-18. Ejemplo de validación cruzada para cuatro divisiones

4.3.3 Arquitectura propuesta

Nuestro modelo cuenta con tres capas convolucionales 2D, todas con la función de activación *ReLu* y las dos primeras cuentan con una capa de *Max Pooling* 2D a su salida. Los filtros de estas capas son de tamaño 5x5, la primera capa convolucional cuenta con 24 filtros y las dos últimas con 48 filtros. Las capas de *Max Pooling* tienen un filtro de 4x2.

Tras esta primera mitad de arquitectura le sigue una capa *Flatten* seguida de una capa de *Dropout*. Continúa el modelo con una capa densamente conectada de 64 neuronas, también seguida de una capa de *Dropout* y con la función de activación *ReLu*. Finalmente se encuentra la capa de salida con 7 neuronas y función de activación *softmax*. En la Figura 4-2 mostramos un esquema visual de nuestro primer modelo.

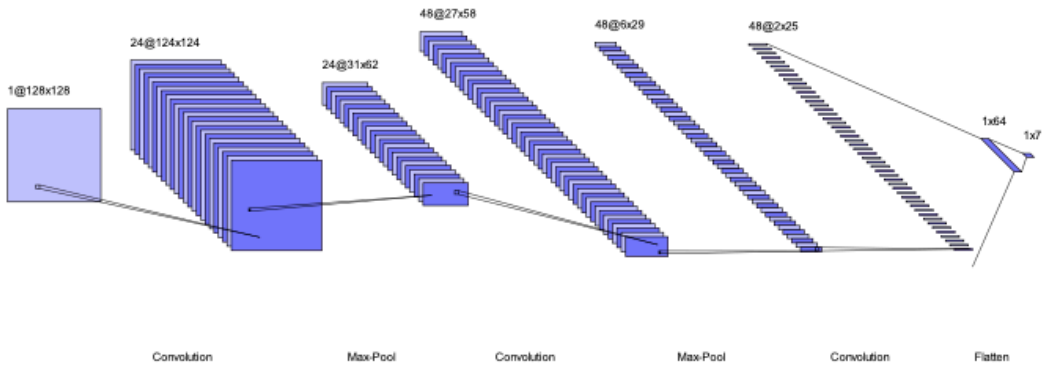


Figura 4-2. Esquema del primer modelo

Finalmente decidimos los parámetros para compilar el modelo. Como función de pérdida, también llamada función de coste, elegimos `categorical_crossentropy` ya que estamos utilizando la función de pérdida *cross-entropy* para nuestro problema de clasificación con varias clases. La fórmula para calcular la pérdida en un clasificador con *C* clases la mostramos en la Ecuación 4-1.

$$CE = -\sum_{i}^{C} y_i * \log y_i^{pred}$$

Ecuación 4-1. Función de pérdida *cross-entropy*

Para calcular la pérdida *cross-entropy* o *CE* tenemos: el valor de la clase real que le pasamos al modelo, y_i, que será 1 si el audio pertenece a esa clase y 0

en los demás casos; y el valor de la clase predicha por el modelo, y_i^{pred}, que nos lo proporcionará la capa *softmax*.

Esta capa devuelve a su salida la probabilidad de que el audio pertenezca a las distintas clases del modelo. En nuestro caso la capa *softmax* posee siete neuronas, una por cada clase (C=7), y para cada audio a clasificar esta capa mostrará en cada neurona la probabilidad de que pertenezca a esa clase. Todas las probabilidades mostradas por la capa final suman la unidad. La clase que el modelo intuya como correcta tendrá un valor cercano a 1 en la neurona correspondiente y las demás un valor cercano a 0.

Volviendo al cálculo del error, solo nos interesa averiguar *CE* para la clase a la que pertenece el audio, ya que el valor de y_i en la multiplicación será 1 y el resto 0. Se puede comprobar que cuando el valor de y_i^{pred} es cercano a 1, *CE* es cercano a 0.

Como función de reducción del coste, también llamado optimizador, elegimos Adam con un *learning rate* de 0.0001. Por último, decidimos que se mida la precisión del modelo. Esto nos devolverá en cada iteración del entrenamiento la precisión del entrenamiento y validación y la pérdida en el entrenamiento y la validación. Mostramos la arquitectura en el Código 4-5.

Para la definición matemática de precisión podemos tomar como ejemplo el clasificador binario descrito en el capítulo anterior, cuya matriz de confusión la mostramos en la Tabla 3-1. La precisión es la relación entre los datos correctamente clasificados por el modelo y el total de datos, tanto correctos como incorrectos (Ecuación 4-2).

Se trata de un porcentaje que puede resultar engañoso si no se tiene en cuenta otras medidas, como es el caso de la pérdida. La pérdida, como hemos visto antes, no se trata de un porcentaje. Su valor es la suma de los errores que el modelo ha cometido al clasificar y es la medida que automáticamente el modelo intenta minimizar con la función reducción de coste y el algoritmo de *Backpropagation*.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Ecuación 4-2. Precisión para un clasificador binario

4.4. Implementación del segundo modelo

Durante nuestra investigación para realizar el capítulo del estado del arte en

esta memoria pudimos comprobar como muchas de las arquitecturas que se proponían en las investigaciones tenían cuatro capas convolucionales en vez de tres. También nos dimos cuenta de este detalle en los modelos propuestos para la competición de *Kaggle*, de los cuales muchos de ellos consiguieron buena puntuación en el reto.

Por eso decidimos implementar un segundo modelo con una arquitectura nueva. Nos basamos en el trabajo del

usuario de *Kaggle* Razar [62] (2018) para desarrollar esta arquitectura. Los pasos del preprocesamiento de los datos son los mismos que para el primer modelo, lo único que alteramos es la arquitectura.

Como hemos comentado, este modelo tiene cuatro capas convolucionales 2D, cada una con 32 filtros de tamaño 4x10 y seguidas una capa de *Batch Normalization*, una función de activación *ReLu* y una capa de *Max Pooling* 2D con filtros 2x2. Continúa con una capa *Flatten*, otra densamente conectada de 64 neuronas, otra capa de *Batch Normalization* y la función de activación *ReLu*. Finalmente tenemos la capa de salida como en el modelo anterior, con 7 neuronas y la función de activación *softmax*. Como en el caso del modelo anterior mostramos un esquema visual de esta arquitectura en la Figura 4-3. Finalmente empleamos los mismos parámetros del modelo anterior para la compilación de este modelo (Código 4-6).

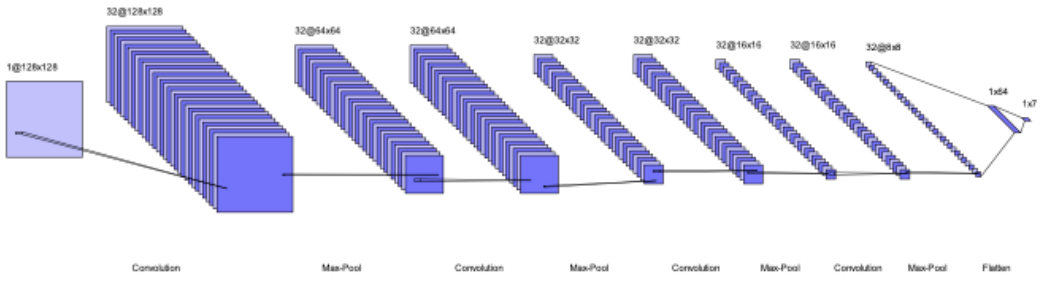


Figura 4-3. Esquema del segundo modelo

5.1.2 Matriz de confusión

El entrenamiento nos dejó una copia de seguridad del modelo con los parámetros que hacían el error de validación mínimo. Concretamente la mejor combinación de valores de pesos y sesgos se dio en la *epoch* número 73. Se consiguió un error de validación de **0.28129** y una precisión en la

validación de **0.9141**.

Estos resultados en el entrenamiento nos dejaron una buena impresión. En primer lugar, la curva del error de validación no se separa apenas de la curva del error de entrenamiento, lo cual significa que se está realizando un buen ajuste. En segundo lugar, el error de validación alcanza un valor mínimo cercano a los valores de pérdida de validación y test de los diversos estudios vistos en el capítulo del estado del arte, alrededor de 0.2.

A continuación, nos dispusimos a cargar el mejor modelo guardado para evaluarlo con los datos de test (Código 5-2). Estos datos han estado apartados durante todo el entrenamiento, por lo tanto, va a ser la primera vez que nuestra red neuronal clasifique estos datos.

Esta evaluación de los datos de test nos proporcionó un resultado en la precisión del **0.9172** y un error del **0.2471**. Como vemos el valor de pérdida del test se acerca aún más al valor del estado del arte. Tras estos resultados en la evaluación nos disponemos a visualizar la matriz de confusión del modelo, empleando para ello la librería importada de *scikit-learn* y la función que previamente definimos.

La precisión del modelo, anteriormente calculada, nos indica el porcentaje de evaluaciones correctas sobre el total de evaluaciones hechas por el modelo. Esta métrica es bastante significativa a la hora de visualizar globalmente el rendimiento del modelo, pero a veces es interesante investigar sobre las predicciones que realiza el modelo en cada clase.

Para ello calculamos la matriz de confusión de nuestro modelo. En primer lugar, se visualiza la matriz de

confusión con todas las muestras de test que se les han pasado al modelo (Tabla 5-1). En segundo lugar, normalizamos los resultados al número de muestras de test verdaderas por cada clase (Tabla 5-2). Así conseguimos visualizar un resultado en tanto por uno dependiendo de las muestras de test observadas en cada clase.

Como vemos nuestro modelo ha clasificado de manera idónea un 90% de las muestras de cada clase. Podemos destacar las 11 muestras de la clase *drilling* que han sido clasificadas por nuestro modelo como la clase *jackhammer*. Ambos sonidos son parecidos, ruidos agudos de maquinaria, por lo que encontramos lógico la clasificación incorrecta de estas muestras. Sin embargo, apenas suponen el 9% de las muestras de la clase

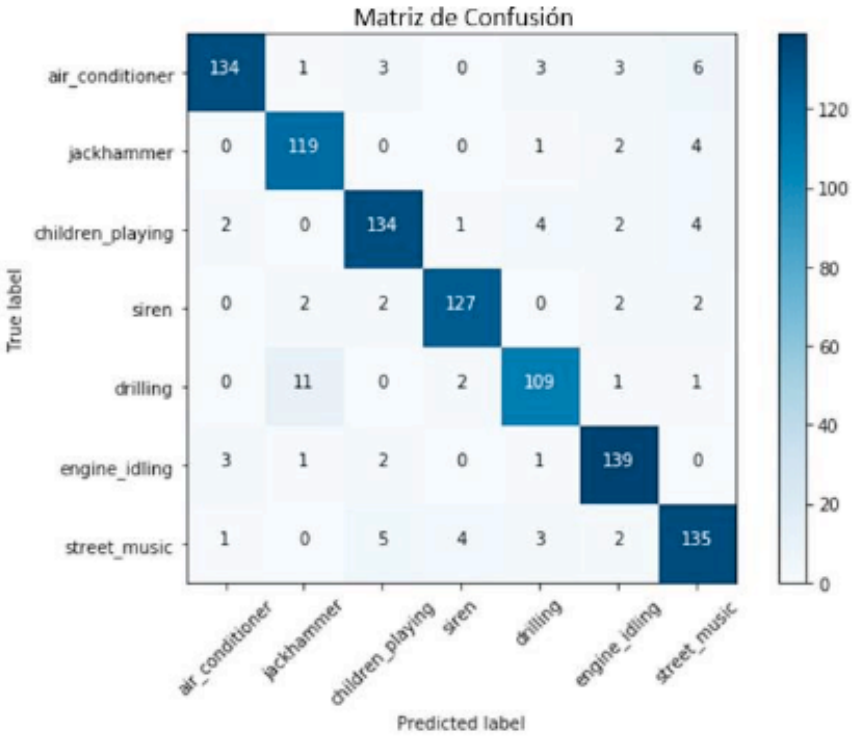


Tabla 5-1. Matriz de confusión del primer modelo

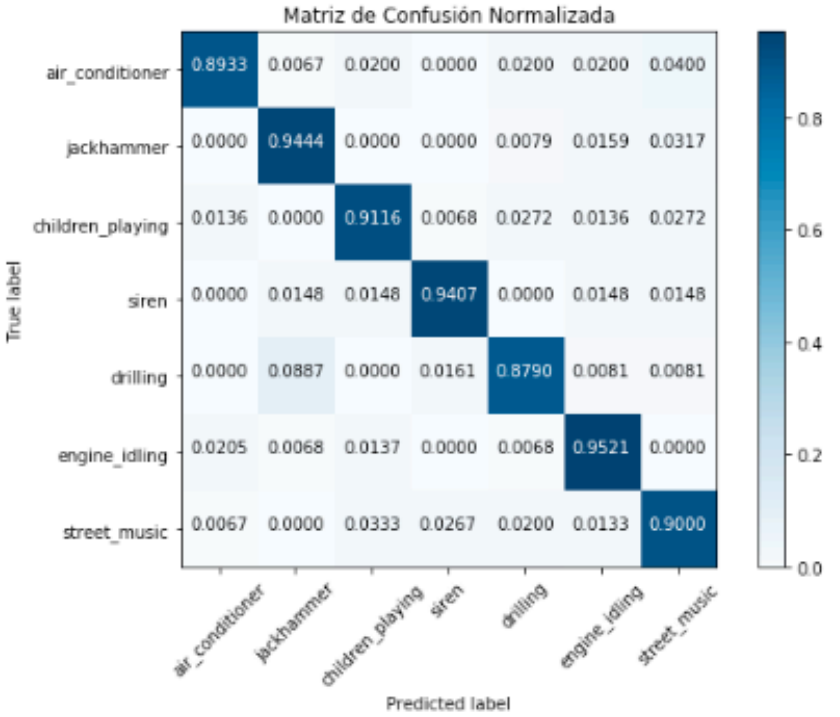


Tabla 5-2. Matriz de confusión normalizada del primer modelo

ME GUSTA LA FORMA DE MOSTRAR LAS ARQUITECTURA DE LOS MODELOS, COMO LOS EXPLICA, TAMBIEN ME GUSTARIA IMPLEMENTAR LA MATRIZ DE

CONFUSION.

Reconocimiento de Voz usando Redes Neuronales Artificiales Backpropagation y Coeficientes LPC

Luis. A. Cruz-Beltrán[1] and Marco. A. Acevedo-Mosqueda[1]

[1] SEPI-Telecomunicaciones ESIME IPN Unidad Profesional “Adolfo López Mateos". Col. Lindavista, 07738, México. D. F
lcruzb06@ipn.mx, macevedo@ipn.mx

4.2 Preprocesamiento

El objetivo de esta etapa es acondicionar la señal de entrada para que esta pueda ser procesada por la RNA, primero acotamos la señal de voz eliminando la parte inicial y final de la misma, que solo representan ruido para obtener la señal de voz a la cual le aplicaremos las Wavelets, como se ve en la Figura 4, se toma la subseñal *a[n]* correspondiente a las bajas frecuencias de la señal de voz donde se localiza la mayor cantidad de energía de la misma, despreciándose la subseñal *b[n]* que corresponde a las altas frecuencias ya que es donde se encuentra la mayor cantidad de ruido de la señal (ruido ambiental y el ruido del canal telefónico). Obteniendo así una señal de voz compacta y filtrada con respecto a la original. Posteriormente se normaliza la señal de voz resultante, para finalmente extraer los coeficientes LPC de la señal, que servirán para el diseño de los patrones de entrenamiento de la RNA.

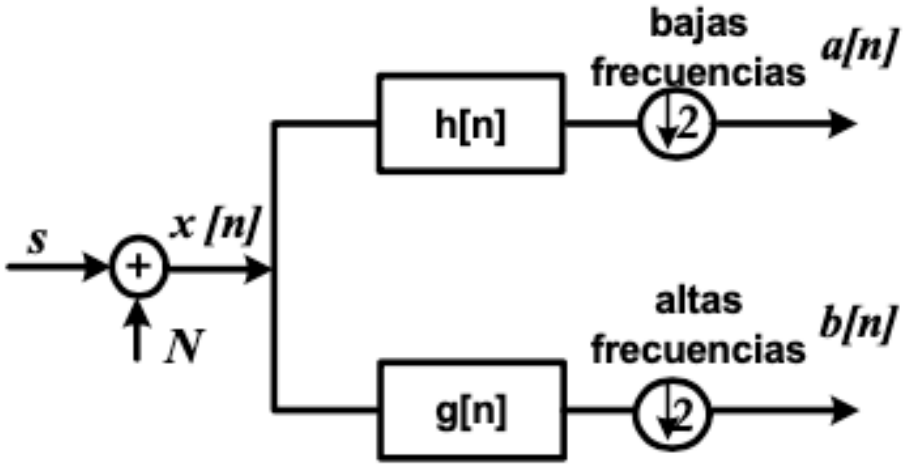


Fig. 4. Estructura de la Wavelet.

La etapa del preprocesamiento de la señal de voz consiste de los siguientes pasos, los

cuales se observan en la Figura 2.

Wavelets: Toda señal de voz en la Naturaleza se encuentra afectada por ruido, y la señal de voz del canal telefónico no es la excepción. Por tal motivo se emplean las Wavelets para reducir este efecto. En este trabajo se propone emplear tres tipos de Wavelets las cuales son Haar, Coiflet y Daubechies. Observando que la mejor de ellas es la wavelet Daubechies debido a que, presenta el mayor porcentaje de compactación de energía de la subseñal $a[n]$ para cada uno de los veinticinco archivos. Esto permite eliminar la subseñal $b[n]$ de altas frecuencias.

Normalización: La normalización consiste en ajustar todos los parámetros a una sola escala para que al momento de ser utilizados por la RNA no causen problemas de estabilidad, en este caso la escala empleada se encuentra dada por los parámetros de la función de activación de la RNA que es una tangente bipolar sigmoidal y trabaja con valores de [-1,1], por lo tanto cada uno de los 25 archivos que previamente fueron compactados y filtrados por medio de las Wavelets es normalizado a esta escala, como se observa en (10), donde los datos que se quieren normalizar se encuentran dentro del vector $x[i]$, con i=$1,...,n$. El procedimiento a seguir es el siguiente:

. a) Se calcula la media (u) y la desviación estándar (σ) del vector $x[n]$.

. b) Senormalizanlosdatossegúnlarelación:

$$x\&[n] = \frac{x[n]-\mu}{\sigma} \quad (10)$$

c) Se calculan el máximo y el mínimo del vector $x\&[n]$, se divide por el de mayor valor absoluto y los datos normalizados caen dentro del intervalo *[-1,1]*.

IMPLMENTAN UNA SEPARACION DE ALTAS Y BAJAS FRECUENCIAS PARA DIFERENCIAR DEL RUIDO AMBIENTAL ESTO TAMBIEN PODRIAMOS IMPLEMENTAR

_ _

UNIVERSIDAD POLITÉCNICA DE MADRID ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA Y DISEÑO INDUSTRIAL Grado en Ingeniería Electrónica Industrial y Automática

TRABAJO FIN DE GRADO

Análisis de la contaminación acústica mediante Inteligencia Artificial

Autor: Álvaro Vellella Ramos

Tutora:

Raquel Cedazo León Departamento de Ingeniería Eléctrica, Electrónica, Automática y Física Aplicada

Madrid, septiembre 2022

Capítulo 3 FUNDAMENTOS

TEÓRICOS

3.1. CONVOLUCIÓN

La convolución es la operación matemática que permite a las CNN extraer las características de una imagen. Desde el punto de vista digital, una imagen es una matriz de dos dimensiones en la que cada elemento está formado por uno o varios bytes, dependiendo del número de canales por el que esté compuesta la imagen. Por ejemplo, en el caso de una imagen a color el número de canales será de 3: un byte para el color rojo, otro para el verde y otro para el azul, mientras que una imagen en escala de grises tendrá únicamente un canal.

La operación de convolución discreta es una transformación en la que el valor del píxel resultante es una combinación lineal de los valores de los píxeles vecinos en la imagen. Esta transformación se puede definir por dos matrices: la imagen y la matriz de coeficientes, también conocida como máscara de convolución, filtro de convolución o kernel, la cual define los pesos que se aplicarán a cada píxel de la imagen y sus píxeles vecino (Figura 3-1). Los filtros de convolución tendrán unas dimensiones mucho menores a las de la imagen, tomando normalmente valores impares de 3x3, 5x5 o 7x7 para anclar el centro del filtro a cada píxel de la imagen.

Como resultado de este proceso iterativo se obtiene una imagen de menores dimensiones a la original, cuyas características se ven realzadas dependiendo del tipo de filtro utilizado.

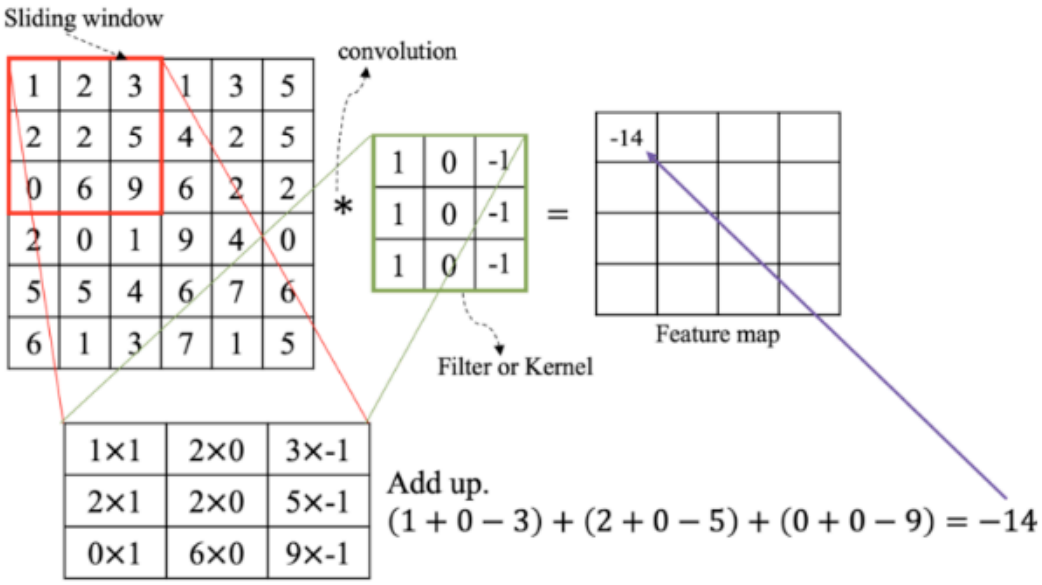


Figura 3-1. Operación de convolución discreta [49].

3.2. REDES NEURONALES CONVOLUCIONALES

Las CNN explotan la operación de convolución en imágenes para extraer las características de las imágenes de entrada. Estas redes tienen una o varias capas convolucionales compuestas por un número determinado de filtros convolucionales. Cada filtro lleva a cabo la operación de convolución sobre la imagen de entrada. Al principio, los pesos de cada filtro tendrán unos valores de inicialización o aleatorios que, a medida que se entrena la red, convergerán en los valores óptimos que permitan detectar las distintas características que permiten clasificar la imagen adecuadamente.

Cada filtro se especializará en detectar una característica concreta y, a medida que se añadan capas convolucionales, las características detectadas por cada capa superior tendrán cada vez una mayor complejidad, ya que utilizan las características más simples extraídas en las capas anteriores.

Normalmente, las capas convolucionales se intercalan con capas de agrupación que reducen la dimensionalidad de las imágenes de salida. Estas capas agrupan los valores de varios píxeles contiguos en un

único píxel siguiendo una norma determinada. Una de las más utilizadas son las capas *max-pooling* o de agrupación máxima, las cuales mantienen el valor máximo de un conjunto de píxeles en el nuevo píxel. A medida que se reduce las dimensiones de las matrices de características, se pueden añadir más filtros en la siguiente capa convolucional.

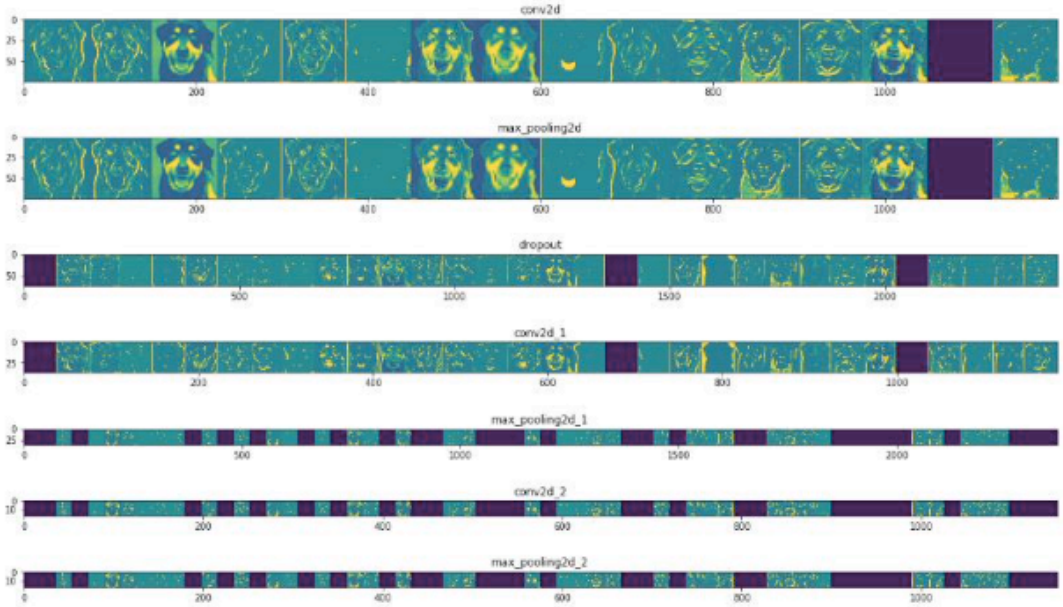


Figura 3-2. Extracción de características de una CNN [50].

En la Figura 3-2 se puede observar el proceso de extracción de características de una CNN a través de las capas de convolución y agrupación. Mientras que algunos filtros se especializan en detectar bordes, otros lo hacen para detectar ciertas texturas o contrastes, obteniendo las representaciones de las distintas características que componen la imagen. A medida que se aplican las capas de agrupación, las dimensiones de la imagen se reducen.

Una vez extraídas todas las matrices de características, éstas se aplanan en un único vector mediante una capa de aplanamiento o *flatten*, o mediante una capa de agrupación global, la cual agrupa todos los píxeles de las imágenes de características en un único valor, como puede ser el valor máximo.

Finalmente, se añaden una o varias capas de neuronas completamente conectadas para llevar a cabo la tarea de clasificación (Figura 2-2). Se llaman así debido a que cada neurona está conectada a todas las

salidas de la capa anterior. La salida de cada neurona es el resultado de la suma ponderada de todas sus entradas. La ponderación de cada entrada viene definida por el peso que se le asigna a cada conexión (Figura 3-3). Además de esto, cada neurona tendrá también un término independiente conocido como *bias* o sesgo, actuando como un modelo de regresión lineal. Al resultado de esta operación se le aplica una función no lineal, llamada función de activación, que evita que todo el conjunto de neuronas colapse como si de una única neurona se tratase (debido a que la suma de varias deformaciones lineales es equivalente a una única deformación lineal). A medida que se entrena la red, las neuronas van ajustando estos valores para obtener las salidas que permitan su clasificación final.

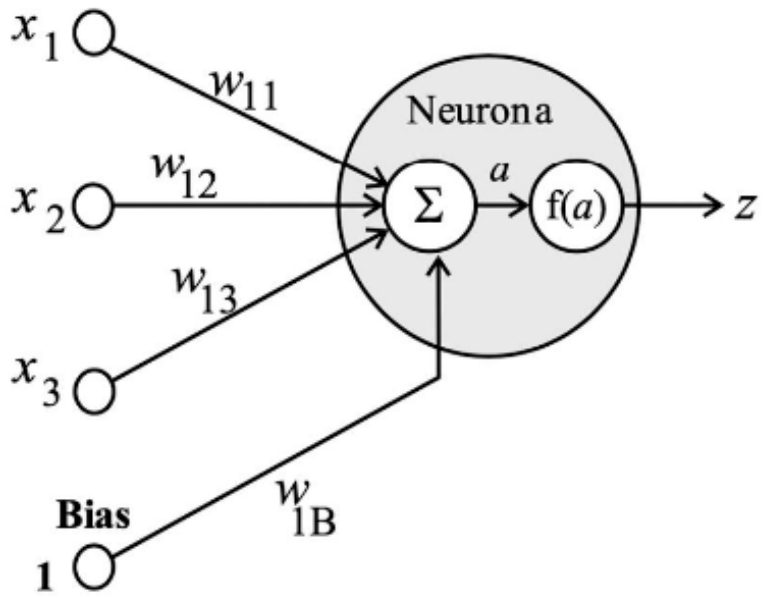


Figura 3-3. Neurona artificial [51].

La última capa neuronal de la red será la capa de salida, la cual debe tener el mismo número de neuronas que clases. La salida de cada neurona será la predicción de la red para su respectiva clase.

3.2.1. ENTRENAMIENTO: PARÁMETROS E HIPERPARÁMETROS

Además de la arquitectura, hay otros dos conjuntos de elementos que definen la configuración de una red neuronal, estos son los parámetros

e hiperparámetros.

Los parámetros de una red se corresponden con aquellos valores que definen el comportamiento de ésta. Se pueden diferenciar dos tipos de parámetros: los que son entrenables, como son los pesos y sesgos de las neuronas o los pesos de las máscaras de convolución, y los no entrenables, como algunos valores que se utilizan en capas de normalización[12]. Los parámetros entrenables son aquellos cuyo valor se modifica o ajusta durante el entrenamiento, mientras que los no entrenables se mantienen constantes una vez calculados. Estos parámetros se pueden contar en miles para redes simples, millones e incluso miles de millones para los modelos más complejos.

Los hiperparámetros de una red son los parámetros ajustables que permiten controlar el proceso de entrenamiento del modelo. Por ejemplo, en una CNN algunos de estos hiperparámetros pueden ser el número de capas convolucionales de la red, la cantidad o las dimensiones de los kernel, el tipo de agrupación que se aplica, el número de neuronas ocultas, etc. La correcta elección de estos hiperparámetros influirá de manera significativa en el rendimiento de la red. Desafortunadamente, el proceso de optimación de la configuración de estos hiperparámetros suele ser manual y muy costoso computacionalmente, ya que requiere de prueba y error.

Además de los hiperparámetros que definen la arquitectura de la red, también están los que configuran el proceso de entrenamiento, como son las funciones de pérdida, optimización y activación, la tasa de aprendizaje o el tamaño del lote.

El entrenamiento de una CNN se basa en el aprendizaje supervisado. Es decir, la red necesita un conjunto de datos etiquetados representativo que le permita evaluar sus predicciones para reajustar sus parámetros y conseguir así resultados más precisos. Las funciones de pérdida o de coste son las encargadas de evaluar la desviación entre las predicciones de la red para cada entrada y los valores reales. El error obtenido se tendrá en cuenta para la autocorrección de los parámetros que tuvieron influencia en esa desviación. Algunas de las funciones de pérdida más comunes son el error cuadrático medio, el

error absoluto o el error absoluto escalar.

En tareas de clasificación multiclase en las que una entrada sólo puede pertenecer a una de muchas categorías posibles, es común utilizar la función de pérdida de entropía cruzada categórica (*categorical cross-entropy*). Esta función está diseñada para cuantificar la diferencia entre dos funciones de probabilidad, por lo que es adecuada para aquellos casos en los que, aunque la predicción ha sido incorrecta, la probabilidad obtenida se ha quedado cerca del valor real. Además, su derivada es sencilla, lo que facilita los cálculos computacionales y mejora el rendimiento del entrenamiento.

El objetivo del entrenamiento es minimizar la función de coste encontrando los parámetros entrenables adecuados y asegurando, al mismo tiempo, una buena generalización. El reajuste de estos parámetros se lleva a cabo mediante un algoritmo numérico llamado *backpropagation*. La función de optimización es la encargada de generar valores de los parámetros cada vez mejores. Su funcionamiento se basa en calcular el gradiente de la función de coste (derivada parcial) por cada parámetro de la red. Como lo que se quiere es minimizar el error, los parámetros se modificarán en la dirección negativa del gradiente. De cara a agilizar la convergencia de la función de coste hacia su mínimo, el vector de gradiente se multiplica por un factor denominado tasa de aprendizaje (*Learning Rate*) [52].

El descenso de gradiente es un algoritmo iterativo, que comienza desde un punto aleatorio en una función y viaja por su pendiente en pequeños pasos hasta que alcanza un mínimo. Este algoritmo es útil en los casos en los que no se pueden encontrar los puntos óptimos al igualar la pendiente de la función a 0, como es el caso de las funciones de coste en redes neuronales. Sin embargo, tiene la limitación de que puede converger en un mínimo local y no absoluto.

El cálculo de la derivada parcial de la función de coste respecto de cada parámetro de la red para cada entrada del conjunto de datos es inviable debido a la enorme cantidad de ambos. La función Stochastic Gradient Descent (SGD) limita el cálculo de la derivada a tan solo

una observación aleatoria por cada iteración, aunque existen algunas variaciones como mini-batch SGD que seleccionan varias observaciones en vez de una.

La función *Momentum* guarda un registro de la media de los antiguos vectores de descenso del gradiente para acelerar el descenso en aquellas direcciones que son similares a las anteriores.

Las funciones AdaGrad (*Adaptive Gradient Algorithm*) y RMSProp (*Root Mean Square Propagation*) adaptan la tasa de aprendizaje de la red a cada parámetro. En el caso de AdaGrad, las tasas de aprendizaje para cada parámetro se calculan con la raíz cuadrada del sumatorio de los valores anteriores al cuadrado, mientras que RMSProp utiliza una media ponderada exponencial.

Por último, el algoritmo Adam (*Adaptive moment estimation*) es una combinación de las funciones de AdaGrad y RMSProp. Adam mantiene una tasa de aprendizaje para cada parámetro y, además de calcular RMSProp, cada factor de entrenamiento también se ve afectado por la media del *momentum* del gradiente.

El entrenamiento de una red neuronal es un proceso iterativo en el que cada dato del conjunto de entrenamiento pasa varias veces por este proceso de optimización. Se conoce como épocas o *epoch* al número de veces que todo el conjunto de datos pasa por este proceso. Cuando el conjunto de datos es muy grande, se suele dividir en lotes de menor tamaño, también conocido como *batch*. Una iteración sería cada vez que un *batch* pasa por el proceso de optimización. Tanto el número de *epochs* que se entrena la red, como el tamaño del *batch* empleado se consideran también hiperparámetros.

Otro hiperparámetro del que ya se ha hablado en el apartado anterior son las funciones de activación. Estas funciones se corresponden con las deformaciones no lineales que se aplican a las salidas de cada neurona. Su elección tiene un gran impacto en la capacidad de aprendizaje de la red neuronal. En general, una buena función de activación cumple las siguientes características [53]:

- Fuga de gradiente: como se ha visto anteriormente, las redes

neuronales se entrenan utilizando el descenso de gradiente y el algoritmo de *backpropagation*, lo que quiere decir que el gradiente de cada capa afecta a la capa anterior. Si el gradiente de la función de activación es cercano a cero, perjudicará el entrenamiento de la red neuronal, pues todos los parámetros de las capas anteriores no se verán casi afectados por la función de optimización.

- Centrado en cero: la función de activación debe ser simétrica en cero, de esta manera, los gradientes no se desplazan hacía una dirección particular.

- Gasto computacional: las funciones de activación se aplican después de cada capa y deben calcularse millones de veces, por lo que una buena función de activación debe ser económica computacionalmente.

• Diferenciable: el cálculo del descenso de gradiente requiere de la derivación de la función de activación, luego necesariamente tienen que ser diferenciables.

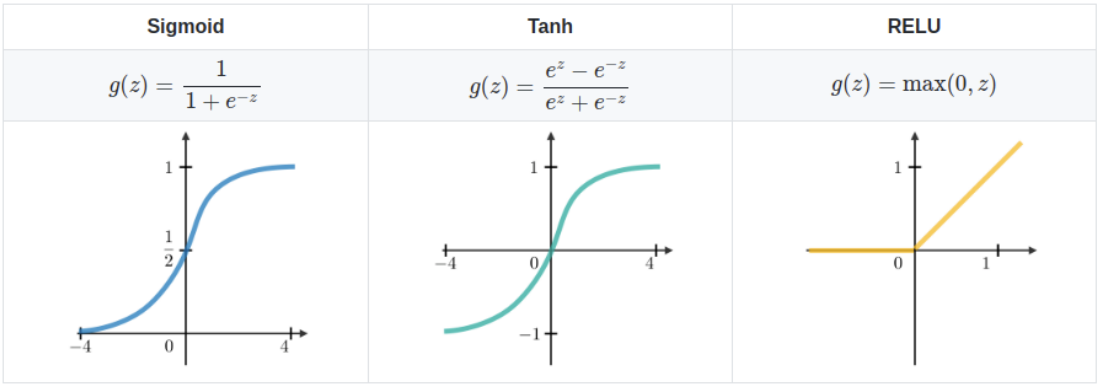
Las funciones de activación Sigmoide (*Sigmoid*) y Tangente hiperbólica (Tanh) se utilizan para clasificadores binarios. Un inconveniente de estas funciones es que, para valores muy pequeños o demasiado grandes, la derivada converge hacia 0, lo que conlleva un problema de fuga de gradiente. Además, no son muy económicas computacionalmente debido a las operaciones exponenciales.

La diferencia entre estas dos funciones es que Tanh está centrada en cero y tiene un rango de entre -1 y 1, mientras que la función sigmoide está centrada en 0,5 y devuelve valores normalizados entre 0 y 1 (Figura 3-4).

Figura 3-4. Funciones de activación Sigmoide, Tanh y ReLU [54].

La función de activación más popular en el aprendizaje profundo es la ReLU (*Rectified Linear Units*). Esta función descarta cualquier valor negativo y mantiene los positivos. Tiene una mejor propagación del gradiente en comparación con las anteriores. Además, es invariante en escala y es una función muy rápida de calcular. Sin embargo, no está centrada en 0 ni tampoco es diferenciable en 0, aunque sí en el resto de valores. Otro inconveniente que tiene es que, al no tener un límite superior, la salida se puede hacer excesivamente grande, dejando a estos nodos inutilizables.

Finalmente, para clasificaciones multiclase es habitual utilizar la función de activación *Softmax*. Esta función transforma las salidas de una capa neuronal en forma de probabilidades, de manera que el sumatorio de todas las probabilidades de las salidas es 1. Se utiliza en la última capa de clasificación.



3.3. SOBREAJUSTE

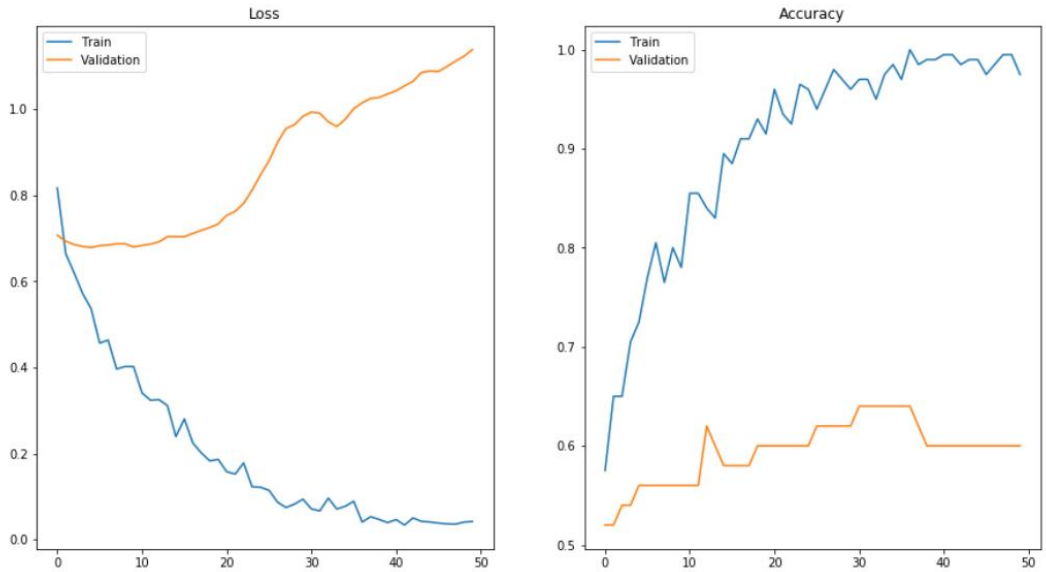
Cada vez que se entrena una red neuronal, existe el riesgo de que los parámetros de la red se ajusten demasiado a los datos del entrenamiento. Un buen modelo es aquel que consigue generalizar de manera adecuada la información proporcionada por los datos del conjunto de entrenamiento, de manera que sea capaz de hacer predicciones precisas cuando se introducen nuevos datos que no ha visto antes.

Normalmente, se utilizan dos conjuntos de datos a la hora de entrenar

un modelo, estos son el conjunto de entrenamiento y el de validación. El conjunto de entrenamiento contiene los datos con los que se va a entrenar la red, mientras que el conjunto de validación se utiliza únicamente para que el modelo haga predicciones, sin reajustar los parámetros de la red. Los datos de estos conjuntos deben ser independientes y nunca deben mezclarse, ya que validar la red con un dato con el que ha sido entrenada no proporciona una valoración objetiva de su rendimiento.

La manera de evaluar el rendimiento una red neuronal es mediante las curvas de aprendizaje (Figura 3-5). Éstas se componen de la tasa de acierto de las predicciones de la red (*accuracy*) y del valor obtenido de la función de pérdida para cada conjunto (*loss*). **Figura 3-5.** Efecto del sobreajuste en las curvas de aprendizaje [55].

Cuando un modelo es demasiado simple, es posible que no tenga los recursos necesarios para ajustarse a los datos de entrenamiento y hacer predicciones precisas. En estos casos, por más que se entrene la red, aunque las curvas de aprendizaje para ambos conjuntos coincidan, ésta no consigue alcanzar una buena precisión para ninguno de los conjuntos, quedándose estancada en un valor intermedio, al igual que la curva de la función de pérdida. Cuando sucede esto, es lo que se conoce como subajuste.



Normalmente, este problema se soluciona aumentando la complejidad

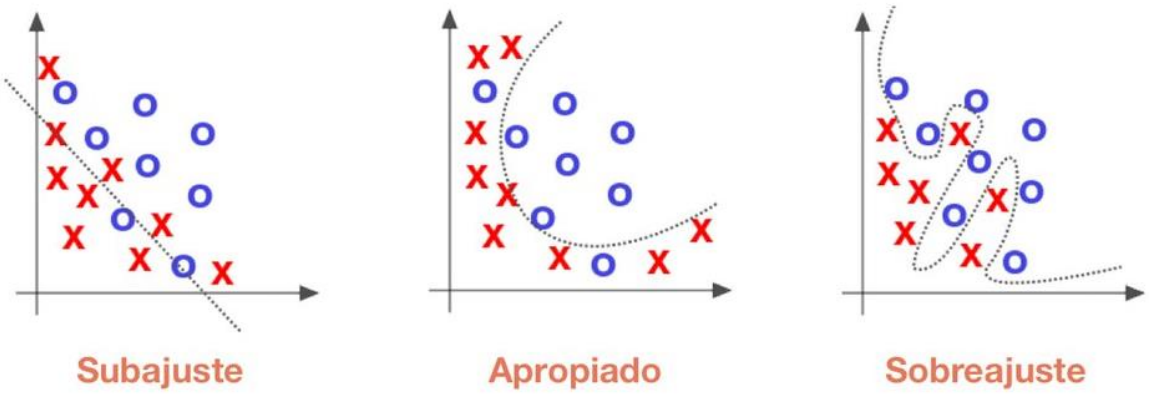
de la red. Es decir, añadiendo un mayor número de filtros, neuronas o capas.

No se debe confundir el subajuste con aquellos casos en los que la red no está aprendiendo y obtiene predicciones aleatorias. Cuando hay un problema de subajuste, la red aprende hasta que sus limitaciones lo permiten, y ambas curvas convergen en un valor distinto a la probabilidad de acertar lanzando predicciones aleatorias, mientras que cuando la red no está aprendiendo, los valores de la curva de precisión siempre estarán en torno al valor de dicha probabilidad.

Por el contrario, cuando la tasa de aciertos para el conjunto de entrenamiento es considerablemente mayor que la del conjunto de validación (Figura 3-6), es una señal de que el aprendizaje de la red se está ajustando demasiado a los datos del entrenamiento y no está siendo capaz de generalizar las características de estos datos para hacer predicciones precisas sobre otros nuevos. Este problema es lo que se conoce como sobreajuste. En la Figura 3-6 se muestra un ejemplo de los problemas de subajuste y sobreajuste para un modelo sencillo de clasificación.

Figura 3-6. Subajuste y sobreajuste de un modelo de clasificación [56].

Los problemas de sobreajuste es algo común a la hora de entrenar una red neuronal. En algunos casos se puede solucionar reduciendo la complejidad de la red, aunque la forma más efectiva de evitarlo es tener un conjunto de entrenamiento lo más amplio y representativo posible. Pero si por lo contrario se cuenta con un conjunto de datos limitado, existen diversas estrategias que permiten generalizar el aprendizaje de la red, como el aumento de datos (*data augmentation*) o las técnicas de regularización.



37

. 3.4. CARACTERÍSTICAS DEL SONIDO

El sonido es una onda mecánica longitudinal que se propaga a través de un medio elástico, como es el aire. Se trata de un transporte de energía sin transporte de materia.

Se produce cuando un cuerpo vibra y transmite dichas vibraciones al medio circundante en forma de ondas sonoras. Éstas se desplazan de forma expansiva a una velocidad determinada que depende de las condiciones del medio de propagación (en el aire, en condiciones normales de presión y temperatura, es de aproximadamente 340 m/s.), y pueden ser absorbidas o rebotar en los distintos tipos de superficies que se encuentren a su paso, logrando diferentes efectos de eco o de distorsión [59].

En el aire, el fenómeno de propagación se debe a la puesta en vibración de las moléculas próximas al elemento vibrante, que a su vez transmiten el movimiento a las moléculas vecinas, y así sucesivamente. La vibración de las moléculas de aire provoca una variación de la presión atmosférica, es decir, el paso de una onda sonora por el aire produce una onda de presión. Esta variación de la presión se denomina presión acústica

39

o presión sonora, y se define como la diferencia entre la presión instantánea y la presión atmosférica en un instante dado [60] (Figura 3-7).

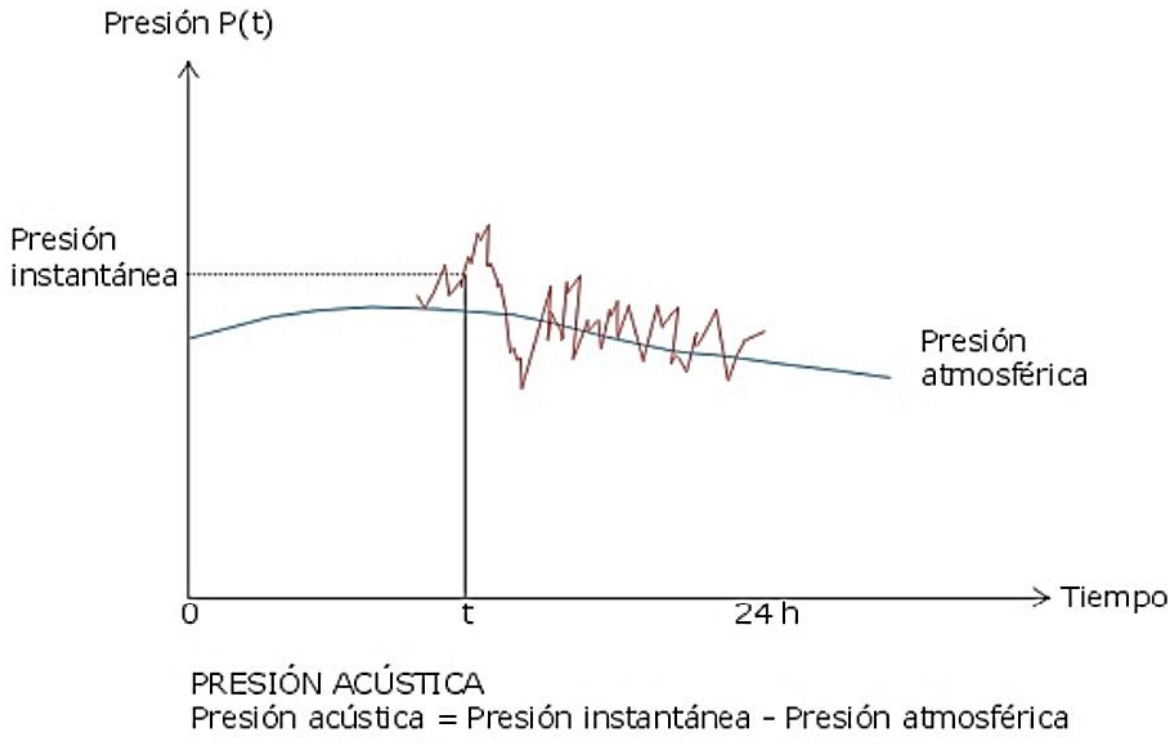
Figura 3-7. Presión acústica [60].

Al igual que cualquier onda en física, el sonido tiene tres características fundamentales: amplitud, frecuencia y composición armónica, lo que en acústica se denomina: intensidad, tono y timbre, respectivamente.

• **Intensidad**: es lo que se conoce como volumen. Hace referencia a la amplitud de la onda sonora y se relaciona con la cantidad de energía transmitida. Se mide en decibelios (dB).

• **Tono**: hace referencia a la frecuencia de la onda sonora. Es el número de oscilaciones por segundo y se mide en hercios (Hz). Los tonos graves se relacionan con las frecuencias bajas y los agudos, con las altas. El ser humano es capaz de escuchar sonidos con tonos comprendidos entre los 20 y 20.000 Hz

• **Timbre**: es la cualidad que permite distinguir dos sonidos de igual frecuencia e intensidad emitidos por distintas fuentes. El sonido, normalmente, no es una única onda de una frecuencia concreta, sino que está compuesto por varias ondas de frecuencias distintas superpuestas. La onda principal se le conoce como fundamental, y el resto de las ondas acopladas tienen el nombre de armónicos [38].



3.4.1. NIVELES SONOROS: EL DECIBELIO

Las presiones acústicas a las cuales es sensible el oído humano varían en un intervalo muy grande. El umbral inferior de la audición humana es de 2 · 10-5 Pa, mientras que el umbral máximo es de alrededor de 20 Pa. Además, el comportamiento del oído humano se asemeja más a una función logarítmica que a una lineal [60]. Es por estos motivos por lo que resulta más conveniente utilizar una escala logarítmica en lugar de una lineal para medir el nivel de presión sonora.

El nivel de presión sonora L se define por la siguiente expresión:

$$p^2\ p\ L_p = 10 \cdot \log_{p^2} = 20 \cdot \log_{p}$$

oo

Figura 3-8. Nivel de presión sonora.

Donde p es la presión acústica de la onda sonora y p_o es el valor de referencia. El nivel de presión sonora L se expresa en decibelios (dB).

Generalmente, se toma el valor de la presión acústica que representa el umbral inferior de la audición humana ($2 \cdot 10^{-5}$ Pa) como valor de referencia. Se habla entonces de decibelios de nivel de presión sonora (Sound Pressure Level, dB SPL). Sin embargo, para sistemas digitales, este valor de referencia es generalmente el valor máximo disponible que puede recoger el dispositivo. Se habla de decibelios a escala completa (Full Scale, dBFS).

En el caso de un micrófono digital, la máxima intensidad sonora que puede medir se corresponderá con los 0 dBFS. Por lo tanto, aquellas intensidades que superen este umbral se registrarán también como 0 dBFS, y las inferiores a él tomarán valores negativos, siendo los más alejados del cero los menos intensos.

En la escala SPL, el sonido audible más bajo se correspondería con un valor de 0 dB SPL, un aumento del doble de la energía de la onda

sonora se traduciría en un incremento del nivel de presión sonora de 3 dB, y si ésta aumenta en un factor de 10, sería un incremento de 10 dB en la escala logarítmica.

Para hacerse una idea de los valores de esta escala, la intensidad del sonido percibida en una biblioteca se correspondería a un valor de 40 dB aproximadamente, una conversación normal rondaría los 60 dB, un restaurante ruidoso, en torno a 90 dB, el interior de una discoteca estaría a unos 110 dB y la explosión de un globo, 150 dB.

Sin embargo, la intensidad percibida de las ondas acústicas emitidas por la fuente sonora se va atenuando a medida que aumenta la distancia a ésta. En el caso de una propagación esférica desde una fuente puntual, como puede ser un disparo o el ladrido

de un perro, al doblar la distancia, el nivel de presión sonora disminuye en 6 dB, y en una propagación cilíndrica desde una fuente lineal, como puede ser una carretera, doblar la distancia supone una pérdida de 3 dB [60].

Además de la distancia, hay otros factores que también atenúan la intensidad de la onda sonora, como la absorción del aire. Debido a que el aire no es un gas de densidad homogénea, ni está en absoluto reposo, existe una atenuación debida a la transformación de parte de la energía acústica en calor. Esta atenuación depende de la frecuencia del sonido, de la temperatura y de la humedad del aire. Cuanto mayor es la frecuencia, mayor es la atenuación experimentada [60].

3.4.2. EXTRACCIÓN DE CARACTERÍSTICAS DEL SONIDO

La señal digital de audio se obtiene de la discretización de la onda de presión sonora descrita en los apartados anteriores a una determinada frecuencia de muestreo. El valor de cada muestra se corresponde con la suma de las amplitudes de todos los armónicos que componen la onda en ese instante y, aunque esta representación del sonido tiene algunas ventajas para su tratamiento digital, no aporta demasiada

información de las características del sonido.

El conjunto de datos que se empleará para entrenar la red, así como las muestras de sonido que se tomarán posteriormente, tienen un formato de archivo de audio (WAV). Sin embargo, el modelo de CNN que se empleará en este proyecto está diseñado para recibir imágenes como entrada. Esto hace que se requiera de un método que permita extraer las características de la señal de audio y representarlas en una imagen que la red pueda interpretar, esto es, un espectrograma.

Los espectrogramas son una representación visual de las variaciones de frecuencia e intensidad del sonido a lo largo de un periodo de tiempo. Existen diferentes técnicas de extracción y representación las de características del sonido, una de ellas y la que se utilizará en este proyecto es el espectrograma de log-Mel (Figura 3-9).

Para descomponer la señal en cada una de las frecuencias individuales y amplitud que la conforman se debe aplicar la transformada de Fourier [61]. Además, como en un audio hay señales sonoras que varían con el tiempo, se debe aplicar la transformada rápida de Fourier (*Fast Fourier Transform*, FFT), que recoge los espectros de la señal aplicando la transformada de Fourier en segmentos muy cortos del audio superpuestos. De esta manera, se convierte la señal del dominio del tiempo al dominio de la frecuencia, obteniendo lo que se conoce como espectro.

La escala de Mel interpreta las frecuencias de manera similar a como lo hace el oído humano. Tiene como unidad un tono, tal que distancias iguales entre tonos suenan igual de distantes para una persona. Un espectrograma de Mel, por tanto, no es más

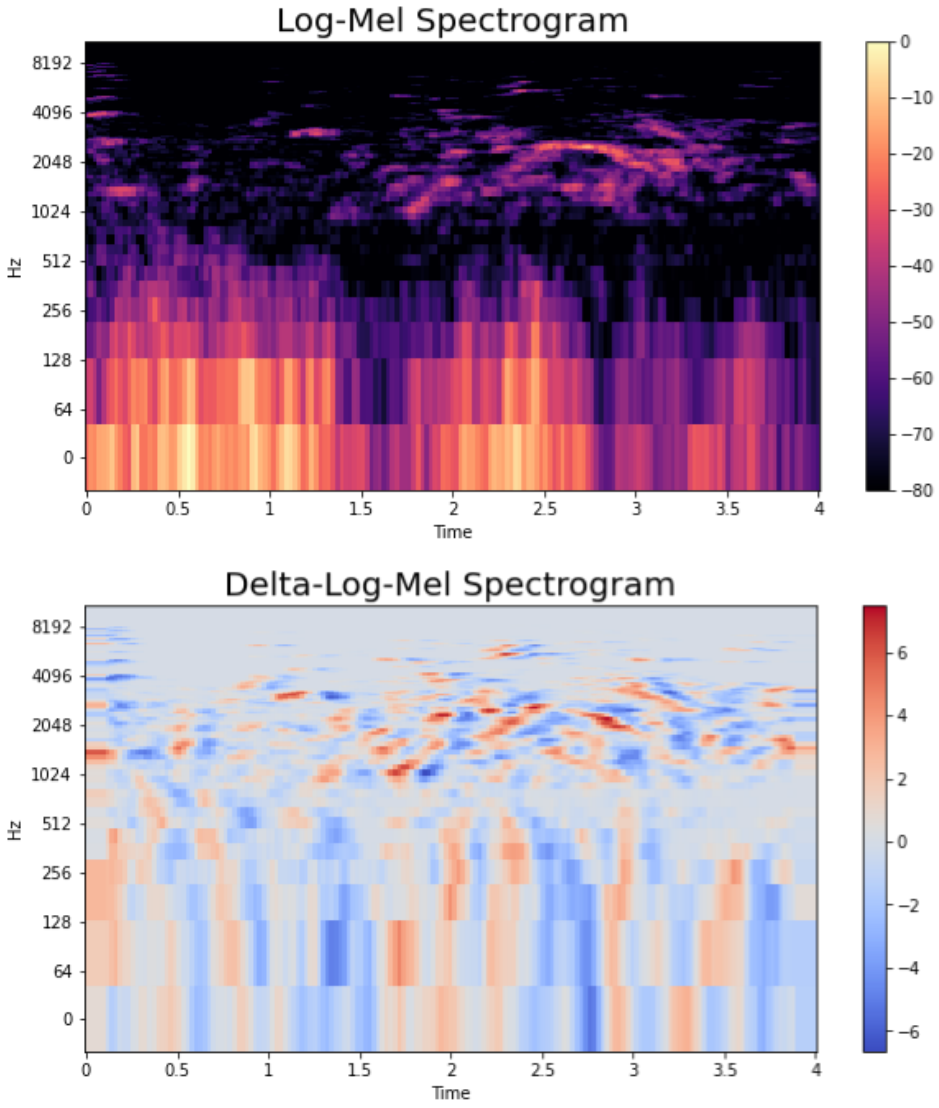
que el espectro obtenido de aplicar la FFT a la onda de audio y representado en la escala de Mel.

Para obtener la representación del espectrograma en esta escala, se debe aplicar un determinado número de filtros de Mel. Dependiendo del número de filtros aplicados se obtendrá un determinado número de bandas, lo que está directamente relacionado con la resolución de la

matriz obtenida. Si además se representan las intensidades del sonido en escala logarítmica (dB), se obtiene lo que se conoce como espectrograma de log-Mel. En la Figura 3-9 se muestra un espectrograma de Log-Mel de 128 bandas.

Figura 3-9. Espectrograma log-Mel de 128 bandas.

Figura 3-10. Espectrograma delta-log-Mel de 128 bandas.



También existen otras variaciones del espectrograma de Mel, como pueden ser el delta-log-Mel o el delta-delta-log-Mel. Estos espectrogramas representan la dinámica del espectrograma de log-Mel, es decir, son una diferenciación de éste. En la Figura 3- 10 se

puede ver un espectrograma delta-log-Mel de 128 bandas.

ESTA MUY BIEN HECHO EL FUNDAMENTO TEORICO.

_____
_____
_____
_____
_____

Clasificación de sonidos ambientales usando la transformada wavelet continua y redes neuronales convolucionales

Francisco J. Mondragón, Héctor M. Pérez-Meana*, Gustavo Calderón, y Jonathan Jiménez

Escuela Superior de Ingeniería Mecánica y Eléctrica Culhuacan, SEPI, Avenida Santa Ana 1000, San Francisco Culhuacan, Culhuacan CTM V, Coyoacán, 04440 CDMX, México. (Correo-e: fmondragon1200@alumnio.ipn.mx; hmperezm@ipn.mx; jjimeneza@alumno.ipn.mx; gus_auza@hotmail.com)

*Autor a quien debe ser dirigida la correspondencia.

Recibido Sep. 1, 2020; Aceptado Oct. 27, 2020; Versión final Dic. 23, 2020, Publicado Abr. 2021

Resumen

Este artículo propone un esquema en el cual inicialmente se obtiene una representación tiempo-frecuencia usando la transformada wavelet continua (CWT), la cual tiene una resolución logarítmica en el plano de la frecuencia similar a la del sistema auditivo humano. El desarrollo de este tipo de sistemas para la clasificación de sonidos ambientales ha sido un tópico de amplia investigación debido a sus aplicaciones en diversos campos de la ciencia e ingeniería. Al igual que otros esquemas de clasificación, estos se basan en la extracción de parámetros característicos, los cuales se insertan en la etapa de clasificación. La CWT se inserta en una red neuronal profunda, para llevar a cabo el proceso de clasificación. Los resultados obtenidos, usando bases de datos de sonidos ambientales tales como, ESC-50, TUT Acoustic Scene y SONAM-50, demuestran que el esquema propuesto proporciona un funcionamiento superior al de otros esquemas previamente propuestos.

Palabras clave: reconocimiento sonidos ambientales; red neuronal profunda; transformada wavelet continua; espectrograma

INTRODUCCIÓN

La clasificación de sonidos ambientales (ESC), por sus siglas en Ingles "Environmental Sound Classification", es un área de investigación en reciente crecimiento, debido al crucial rol que tienen los sonidos en nuestra interacción con el entorno. Por lo tanto, es fundamental para el éxito de la inteligencia artificial que los robots o las computadoras puedan comprender los sonidos, en forma similar a como lo hacen los humanos. El Desarrollo de tecnologías que ayuden en tareas como, sistemas para el monitoreo de cuartos inteligentes, mejorar la navegación autónoma (Chu et al., 2009), determinación de especies de aves y mamíferos basado en los sonidos que producen (Abber, 2020; Potamitis, 2014; Xie y Zhu, 2019), clasificación de sonidos en ayudas auditivas (Alexandre et al., 2007), indexación y recuperación de contenido multimedia (Tong et al., 2014), entre otras ha sido la motivación para el desarrollo de sistemas de ESC.

Algunos enfoques utilizados en la tarea de ESC se basan en características como la Transformada de Fourier Discreta (DFT), coeficientes cepstrales de las frecuencias Mel (MFCC) (Chu, et al., 2009; Salamon y Bello, 2014), coeficientes cepstrales de frecuencias gammaton (GFCC), características estadísticas y combinación de estas. La representación de los sonidos basada en el sistema de audición humana tiene un gran interés en el desarrollo de características para realizar ESC. Los modelos auditivos se basan en algoritmos matemáticos que intentan imitar el procesamiento de audición humana diseñados a partir de experimentos psicofísicos y fisiológicos.

El análisis wavelet (Martínez et al., 2018; Jiménez et al., 2018) se está convirtiendo en una herramienta matemática habitual en el estudio de señales no estacionarias como lo son la mayoría de los sonidos. Esta herramienta realiza el análisis de una señal usando versiones escaladas y trasladadas de una función base llamada función wavelet madre [?](t). Hay dos diferentes tipos de transformada wavelet: la Transformada Wavelet Discreta (DWT) y la Transformada Wavelet Continua (CWT). La principal diferencia entre ambas transformadas es la forma en la cual el parámetro de escalamiento es discretizado. La CWT discretiza más fielmente que la DWT. La diferencia es que mientras que en la CWT normalmente se determina alguna base que es una potencia fraccionaria de dos, es decir $2^{j/v}$ con j=1, 2, 3, ..., n donde el parámetro v es conocido como el número de voces por octava, ya que para poder incrementar la escala en una octava se necesitan v escalas intermedias, mientras que en la DWT el parámetro de escalamiento siempre se discretiza con potencias enteras de dos, esto es 2^j con j=1 ,2, 3, ..., n por lo que el número de voces por octava es siempre 1, por lo tanto la CWT dependiendo del valor de v nos proporciona una mayor resolución en frecuencia de la señal en análisis. Sin embargo, esto también aumenta la cantidad de cálculo requerido. La CWT se puede

considerar como un banco de filtros que tienen subbandas de frecuencia espaciadas de manera logarítmica similar al sistema auditivo humano, por lo cual provee una representación tiempo frecuencia con una resolución logarítmica en el eje de la frecuencia, lo que proporciona una mejor representación para la inspección visual, como se muestra en la figura 1.

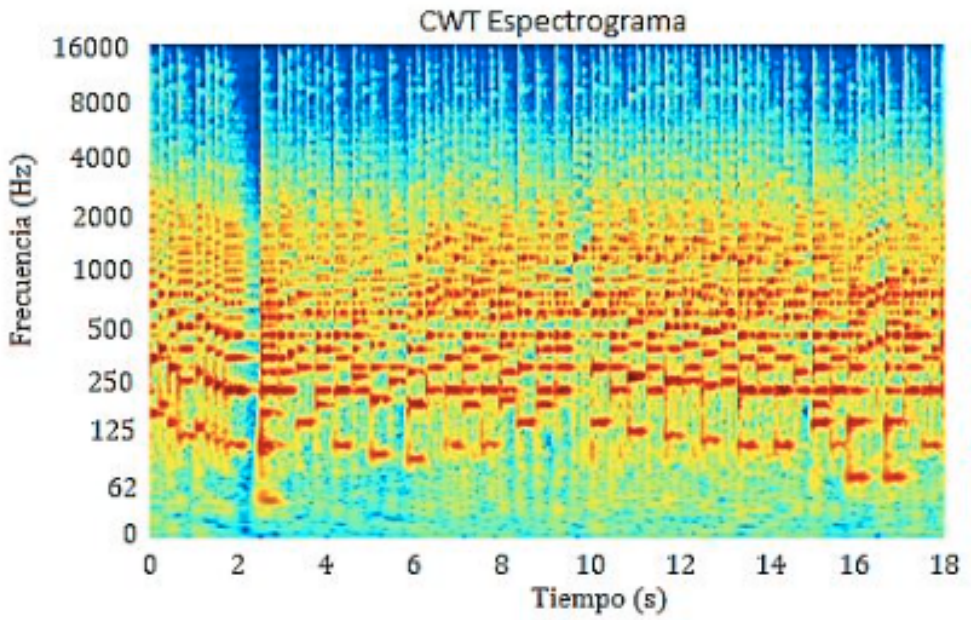


Fig. 1: CWT Espectrograma de una grabación de piano

METODO PROPUESTO

El sistema propuesto para la clasificación de sonidos ambientales e identificación de senarios acústicos se muestra en la figura 8.

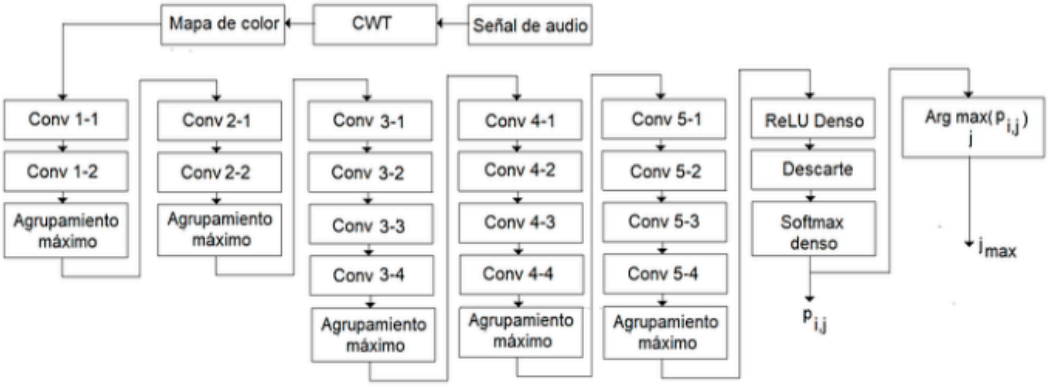


Fig. 8. Sistema propuesto para la clasificación de sonidos y escenarios ambientales

Bases de Datos

Las bases de datos empleadas para llevar a cabo la evaluación del sistema propuesto son: a) ESC-50 (Piczak, 2015b), la cual contiene 2000 archivos de audio de sonidos ambientales con 5 segundos de duración cada uno, en formato WAV, con frecuencia de muestreo de 44,100 Hz. Está dividida en

50 clases con 40 muestras por cada clase. Esta base de datos incluye sonidos tales como vocalizaciones de animales, sonidos provocados por el agua, paisajes sonoros naturales, sonidos no vocalizados emitidos por humanos, sonidos domésticos y sonidos urbanos. b) La segunda base de datos desarrollada por nosotros, llamada SONAM- 50, contiene 1200 grabaciones de diferentes longitudes de tiempo, en formato WAV, con frecuencia de muestreo de 32,000 Hz. Está basada en sonidos usados en el estudio de similitud y categorización de sonidos ambientales reportado por Gygi (Gygi et al., 2007). Esta base de datos tiene 50 clases de sonidos ambientales tales como: sonidos producidos por maquinas, sonidos de varias condiciones climáticas, sonidos humanos no vocalizados, vocalizaciones de animales y sonidos generados por actividades humanas. Cada uno con 24 muestras por cada clase. Como preparación para el procesamiento de estas bases de datos se aplicó una compuerta de ruido, usando como umbral -30 dB para la máxima amplitud de la grabación antes de saturar y

se normalizaron las grabaciones a un nivel de -3 dB del nivel máximo en la grabación. c) La tercera base usada, la TUT Acoustic Scene (Mesaros A. et al. 2017) la cual consiste en grabaciones de varios escenarios acústicos con distintas locaciones de grabación, para cada locación se realizaron grabaciones de 3 a 5 minutos y después fueron divididos en archivos de 10 segundos de duración, estos tienen una frecuencia de muestreo de 44100 Hz, 24 bits de resolución y estereofónica. Contiene 15 distintos escenarios acústicos con 312 muestras por cada escenario, estos escenarios los conforma grabaciones hechas en algún transporte (autobús, automóvil, tren, tranvía), en el interior de un recinto (cafetería, tienda, casa, biblioteca, estación de metro, oficina) y en el exterior (ciudad, camino forestal, playa, residencial, parque).

CONCLUSIONES

Este artículo propone un algoritmo para la clasificación o reconocimiento de eventos acústicos o sonidos ambientales basado en la transformada wavelet continua y redes neuronales profundas. Los resultados experimentales, (figura 9) muestran las funciones wavelet madre log-normal y Morlet, proporcionan los CWT espectrogramas más apropiados para caracterizar las señales de audio. Se evaluaron además diversas estructuras de redes DNN y de la Tabla 2 se puede observar que, de las redes evaluadas, la red VGG-19 es la más adecuada para realizar la tarea de ESC. Así mismo, dado que el mapa de color de los espectrogramas tiene un rol importante, se llevó a cabo una evaluación de varios de ellos,

obteniéndose que los mapas de color Viridis, Jet y Gray podrían ser adecuados para crear los CWT espectrogramas usado para clasificar los sonidos. Así mismo, se encontró que una duración de 2.5s parece adecuada para llevar a cabo el reconocimiento, al igual que una frecuencia de muestreo de aproximadamente 22kHz. Finalmente se analizó el número de voces por octava que proporcionan el mejor funcionamiento del sistema propuesto, encontrándose que el incremento del número de voces por octava impacta de manera importante en la precisión al implementar ESC. Una vez determinados los parámetros del sistema se evaluó su funcionamiento cuando se requiere detectar tanto eventos acústicos, como sonidos ambientales. Los resultados experimentales obtenidos muestran que cuando el algoritmo propuesto se emplea para reconocer eventos acústicos, éste presenta una tasa de reconocimiento del 85.75% usando la base de datos ESC-50, 89.26% usando las bases de datos ESC-50 y 90.85% usando la base de datos TUT Acoustic scene, respectivamente, la cual es superior al reconocimiento proporcionado por otros esquemas previamente reportados en la literatura, basados en la transformada wavelet y redes DNN; además de ser muy cercano a otro esquema basado en máquinas de Boltzman restringidas (RBM), como se muestra en la Tabla 7. Finalmente se observa que, el sistema propuesto presenta una tasa de reconocimiento de 80% cuando se emplea para el reconocimiento de sonidos ambientales usando la base de datos ESC-50 y 87.6% de reconocimiento cuando se emplea la base de datos SONAM-50.

ES MUY INTERESANTE LO QUE LOGRAN CON LA TRANSFORMADA DE WAVELET PARECE SER LA MEJOR PARA RUIDOS AMBIENTALES, HABRIA QUE PORBAR CON SU MODELO PROPUESTO, Y ME GUSTA COMO EXPLICA LA BASE DE DATOS, NOSOTROS DEBEMOS PONER LA TOMA DE MUESTRA CON EL CELULAR ESPECIFICACIONES Y COMO CORTAMOS LOS AUDIOS Y SUS ESPECIFICACIONES

SISTEMA DE PREDICCION DE RUIDO URBANO MEDIANTE REDES NEURONALES TESIS DOCTORAL NATALIA GENARO GARCIA

TIEMPO E INTERVALOS

Las medidas objetivas de los niveles de sonido son una parte indispensable de cualquier programa de protección contra el ruido ambiental. Los niveles de ruido ambiental varían espacial y cronológicamente, ya que el ruido es a menudo impulsivo o puede contener tonos puros. Además, las molestias procedentes de fuentes de ruido externas – ladridos de perro, vuelo de aviones, niños jugando – deben tratarse de formas diferentes.

El resultado de una evaluación de un ruido nunca es una simple cifra como 77 dB. Es el valor de los parámetros o indicadores específicos obtenidos bajo unas condiciones conocidas y bien documentadas.

Se eligen diferentes intervalos de tiempo dependiendo de los objetivos perseguidos y de las características de la circulación. En la elección de los períodos de medida existen varias tendencias:

- Encontrar las horas de mayor tráfico y medir para obtener el valor medio de ese período.
- Medir durante el tiempo correspondiente al paso de, al menos, un cierto número de vehículos ligeros y/o pesados y considerar los resultados obtenidos como la energía sonora característica de la carretera.
- Medir durante largos períodos (más de 24 horas).

Lo ideal es medir el ruido durante el intervalo temporal de referencia completo. Como regla general, los períodos de medición deben ser tan largos como sea necesario para conseguir un buen conocimiento de la evolución del ruido durante un día, una semana o una estación, teniendo en cuenta las condiciones atmosféricas de la zona. A veces por determinadas razones se utilizan períodos más largos tales como

mediciones de un mes y un año. En dichos casos para obtener un historial de tiempo de los niveles de ruido, se utiliza un registro de valores obtenidos cada segundo, minuto o cuarto de hora. Sin embargo, las mediciones de larga duración pueden ser caras y difíciles de gestionar. Normalmente esas evaluaciones se basarán en tomar mediciones de muestras representativas y con ellas, extrapolar una visión completa y general.

De todo esto se extrae que el tiempo de medición no tiene límite. Sin embargo, como deben realizarse con frecuencia estudios y evaluaciones en cortos períodos de tiempo, la mayoría de las normas nacionales han fijado mínimos intervalos de tiempo para medir, que dependerán del propósito de la evaluación y suele estar entre quince minutos y una hora. En las mediciones para este estudio se han hecho durante un tiempo de entre 25 y 70 minutos, dependiendo del *tiempo de estabilización*, que ha oscilado entre 5 y 53 minutos. El tiempo de estabilización es el tiempo a partir del cual no tiene sentido seguir midiendo puesto que no van a obtenerse datos diferentes del nivel de L_{Aeq}, ya que se considera que se ha estabilizado. Sobre el estudio del tiempo mínimo de medición y del tiempo de estabilización se han publicado varios trabajos entre los que se destacan[8] y [151].

VARIABLES QUE SE MIDEN

Ya en el capítulo I se dio una definición de los elementos que influyen en el nivel de presión sonora y/o que han sido utilizados por algunos de los métodos analizados en el capítulo I. Son los siguientes:

- Caudal de vehículos.
- Velocidad de los vehículos.
- Distancia desde la fuente de ruido al oyente.
- Ángulo de visión.
- Tipo de vía.
- Anchura de la vía.
- Altura de la vía.
- Pendiente de la vía.
- Tipo de flujo del tráfico.
- Tipo de pavimento.

Dado que la ciudad de Granada es especialmente ruidosa y de una fisonomía muy particular, los expertos en ruido ambiental han recomendado tener en cuenta una serie de consideraciones especiales para el estudio:

- Se distinguirá entre el número de carriles ascendente y descendente, así como el estado del pavimento.
- Además de la anchura de la vía, se tendrá en cuenta la anchura de la calzada.
- Dada la gran influencia de la pendiente de la vía en la producción de ruido por parte de los automóviles, se diferenciará entre el caudal de vehículos ascendente y descendente para vehículos ligeros, pesados y motocicletas. Esto implicará 6 variables que engloban el caudal de vehículos en general.

- Se tendrá en cuenta la aparición de eventos anómalos, separados en: eventos anómalos relacionados con el tráfico y eventos anómalos no relacionados con el tráfico. Por otro lado, se tendrá en cuenta el número de vehículos con sirena observados durante la toma. Los eventos anómalos son eventos sonoros cortos y de gran intensidad, como se describió en la definición del modelo lineal multivariante granadino en el capítulo I.
- Las condiciones del entorno son también importantes, teniendo en cuenta la presencia de obras y el tipo de entorno (comercial y de ocio o no).
- Se considerará el período del día en el que la toma fue efectuada (día o noche), así como el tiempo de estabilización (descrito en el apartado anterior, Tiempo e Intervalos).
- Se descarta el ángulo de visión, dado que su influencia es despreciable.

DIFIERE DE LO QUE ESTAMOS HACIENDO, PERO ES INTERESANTE VER LO INTERVALOS DE TIEMPO Y LAS VARIABLES QUE MIDEN PARA PODER CLASIFICAR EL SONIDO, NOSOTROS DEBERIAMOS AGREGAR LA VELOCIDAD DE LOS VEHICULOS?, EL LUGAR EN DONDE ME UBICO SIEMPRE PARA GRABAR LOS VIDEOS HAY UN REDUCTOR DE VELOCIDAD, EN PRINCIPIO TODPS LOS AUTOS QUE GRABE DEBERIAN TENER APROXIMADAMENTE LA MISMA VELOCIDAD.

— —
— —
— —
— —
— — — — — — — — — — — — — — — — — —