SOLUCIÓN IOT PARA DETECCIÓN DE SONIDOS MEDIOAMBIENTALES MEDIANTE APRENDIZAJE PROFUNDO / IOT SOLUTION FOR ENVIRONMENTAL SOUNDS DETECTION THROUGH DEEP LEARNING

TRABAJO FIN DE MÁSTER
CURSO 2019-2020

AUTORA
CLARA ALESSANDRA JUSTINO LIMACO

DIRECTOR
GONZALO PAJARES MARTINSANZ

CONVOCATORIA: JULIO 2020
CALIFICACIÓN: 9 (SOBRESALIENTE)

MÁSTER EN INTERNET DE LAS COSAS
FACULTAD DE INFORMÁTICA
UNIVERSIDAD COMPLUTENSE DE MADRID

SOLUCIÓN IOT PARA DETECCIÓN DE SONIDOS MEDIOAMBIENTALES MEDIANTE APRENDIZAJE PROFUNDO / IOT SOLUTION FOR ENVIRONMENTAL SOUNDS DETECTION THROUGH DEEP LEARNING

TRABAJO DE FIN DE MÁSTER EN INTERNET DE LAS COSAS
DEPARTAMENTO DE INGENIERÍA DE SOFTWARE E INTELIGENCIA ARTIFICAL

AUTORA
CLARA ALESSANDRA JUSTINO LIMACO

DIRECTOR
GONZALO PAJARES MARTINSANZ

CONVOCATORIA: JULIO 2020
CALIFICACIÓN: 9 (SOBRESALIENTE)

MÁSTER EN INTERNET DE LAS COSAS
FACULTAD DE INFORMÁTICA
UNIVERSIDAD COMPLUTENSE DE MADRID

2 DE JULIO DEL 2020

AGRADECIMIENTOS

A mis padres y a mi hermana gracias por el inmenso apoyo y cariño que me dan en todo momento. Gracias a mi tutor Gonzalo Pajares por ayudarme, apoyarme y guiarme en el desarrollo del proyecto.

RESUMEN

El trabajo plantea una solución conceptual para detectar e identificar sonidos mediante la aplicación de técnicas inteligentes basadas en Redes Neuronales Convolucionales (aprendizaje profundo) bajo el paradigma de Internet de las Cosas (IoT). El objetivo viene motivado por la necesidad de detectar sonidos de motosierras para prevenir la tala indiscriminada en zonas forestales como fin último. No obstante, la propuesta es aplicable a cualquier entorno donde la monitorización del ruido sea el objetivo principal.

Para ello se proporciona una solución integrada, formada por una serie de módulos para captura, procesamiento y transmisión de datos e información mediante los dispositivos físicos, repositorios y algoritmos correspondientes. Más específicamente, se dispone de un dispositivo móvil de captura de audios, simulado en este caso por audio clips de una duración determinada. Se dispone de un computador donde se procesan los audios, bien en local o en remoto, generando los correspondientes espectrogramas a partir de los audios, que se pasan a un modelo específico de red neuronal convolucional con sus correspondientes capas para el entrenamiento y posterior clasificación en las tres categorías de clases utilizadas, una de ellas la de los sonidos de motosierra.

Al servidor remoto tienen acceso tanto el dispositivo móvil como el computador, para llevar a cabo los procesamientos requeridos; si bien, el primero carece de capacidad para procesos computacionalmente costosos, tal como ocurre con el entrenamiento de la red, por lo que esta fase sólo es posible realizarle en el computador, eso sí, bien en remoto o en local. El modelo de red y los parámetros obtenidos durante el entrenamiento se almacena convenientemente en remoto, con acceso del dispositivo móvil, que realiza el proceso de clasificación. Los resultados de la clasificación se envían a la nube, donde se almacenan para su posterior visualización con fines de monitorización, a la vez que se levantan las correspondientes alarmas de aviso cuando se detectan las acciones de tala no controlada a través de una cuenta Twitter y, por tanto, accesible desde cualquier dispositivo con esta capacidad.

Los resultados obtenidos permiten validar el modelo desde el punto de vista conceptual previsto, tanto desde el punto de vista de los módulos de procesamiento inteligente como del planteamiento dentro del paradigma IoT.

Palabras clave

Reconocimiento de sonidos, aprendizaje profundo, Internet de las Cosas, motosierra

ABSTRACT

The work proposes a conceptual solution to detect and identify sounds by applying intelligent techniques based on Convolutional Neural Networks (deep learning) under the Internet of Things (IoT) paradigm. The objective is motivated by the need to detect the sounds of chainsaws to prevent indiscriminate felling in forest areas as the goal. However, the proposal is applicable to any environment where noise monitoring is the main objective.

To do that, an integrated solution is provided, consisting of several modules, including capturing, processing and data transmission and information through the corresponding physical devices, repositories and algorithms. More specifically, a mobile device captures audios, simulated in this case by audio clips of a specified duration. There is a computer where the audios are processed, either locally or remotely, generating the corresponding spectrograms from the audios, which are fed to a specific model of convolutional neural network with its corresponding layers for training and subsequent classification in the three categories of classes used, one of which is chainsaw sounds.

Both, the mobile device and the computer can get access to the remote server, to carry out the required processing; although the former lacks the ability for computationally expensive processes, such as the ones occurred during the network training, so this phase can only be performed on the computer, however, either remotely or locally. The network model and the parameters obtained during the training phase are conveniently stored remotely, with access from the mobile device, which performs the classification process. The classification results are sent to the cloud, where they are stored for later visualization and for monitoring purposes, while the corresponding warning alarms are raised when uncontrolled deforestation actions are detected through a Twitter account and therefore, accessible from any device with this capacity.

The results derived from the previous processes allow validating the conceptual model, both from the point of view of the intelligent process, based on the convolutional neural network and the proposed approach within the IoT paradigm.

Keywords

Sound recognition, deep learning, Internet of Things, chainsaw

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

Capítulo 1 - Introducción

1.1 Preliminares

Diariamente, las personas están sujetas a una variedad de ruidos de amplitud variable, estas fuentes de ruido incluyen aquellas relacionadas con viajes, por ejemplo, trenes subterráneos, motocicletas, motores de aviones y ruido de viento, por citar sólo algunos y a su vez con la ocupación de una persona, por ejemplo, equipos de fábrica, motosierras, taladros neumáticos, cortadoras de césped, segadoras, entre otras (Anderson, 2015). Los animales también producen ruidos para diversas funciones biológicas, como defender territorios, atraer parejas, disuadir a los depredadores, navegar, encontrar comida y mantener contacto con los miembros de su grupo social (Darras y col., 2016).

En los últimos años el número de soluciones de monitoreo acústico ha ido incrementando, producto de las distintas soluciones que se pueden plantear con solo analizar los diferentes ruidos que podemos encontrar en el medio ambiente.

Estas soluciones vienen desde el control para medir la contaminación por ruido en las ciudades como una prioridad y a partir de ese compromiso avanzar hacia la construcción de entornos acústicos más saludables (Gallo y col., 2018), determinar la riqueza de fauna y construir índices de biodiversidad estudiando así las interacciones y dinámicas sociales de las mismas (Darras y col., 2016) y la elaboración de alarmas a partir de los eventos de reproducción de ruidos que impliquen actividades ilegales. Entre algunas de las soluciones existentes destacan las siguientes:

a) Rainforest Connection

El proyecto RFCx (Rainforest Connection, 2020) crea sistemas de monitoreo acústico para la deforestación ilegal en tiempo real. Los guardaparques son los primeros en ser informados, el guardaparques más capacitado logrará detener la tala ilegal. Con análisis predictivo RFCx ahora puede predecir con 3 minutos de anticipación cuándo y dónde es probable que ocurra un evento con intervención de un ruido de motosierra. Una aplicación móvil alerta a los guardabosques para que puedan aparecer antes de que los registradores puedan comenzar. La predicción se basa en sonidos históricos, capturados por teléfonos inteligentes antiguos, que transmiten los sonidos de la selva tropical a un modelo de aprendizaje automático para detectar estos eventos.

b) Low-cost IoT noise monitoring network in Torino

La ciudad de Torino decidió enfrentar el problema de contaminación acústica por actividades de vida nocturna mediante un enfoque integrado y estableció una red de monitoreo de ruido, bajo el paradigma de IoT (Internet of Things), de bajo costo utilizando varios teléfonos inteligentes Android en el distrito de San Salvario, donde existe una gran cantidad de restaurantes, bares, pubs y clubes que atraen cada fin de semana miles de personas. La Agencia Ambiental Regional (ARPA Piemonte) desarrolló una aplicación para el procesamiento de señales y la transmisión de datos y, después de un conjunto completo de pruebas de laboratorio para evaluaciones de incertidumbre y la definición de un procedimiento de calibración, comenzó un despliegue in situ en el verano de 2016. Cada hora los datos se analizaron después de haber sido recopilados durante meses en tiempo real por una plataforma regional de datos abiertos IoT, lo que condujo a un primer mapa de exposición para el ruido de ocio y un espectro detallado anual de niveles de ruido por hora. Este enfoque basado en datos ha sido elegido para impactar en la agenda política, en una participación positiva más fuerte de los ciudadanos en los grupos focales, en el compromiso de los empresarios locales y en la conciencia de los clientes, para comenzar una mejor adaptación de los eventos al aire libre a la vida en la ciudad. La red de monitoreo de ruido permitió medir los efectos de las nuevas regulaciones en el verano de 2017 apoyando la evaluación del efecto general de las nuevas acciones para la reducción de ruido previstas para 2018, en el marco del Proyecto Monica Horizon 2020 (Gallo y col., 2018).

Los ejemplos anteriores son sólo dos aspectos a considerar en relación a diferentes aplicaciones relacionadas con el análisis y monitorización del ruido en diversos ámbitos, principalmente en entornos de exterior. Además, gracias al auge de las tecnologías sensoriales y los desarrollos en el ámbito de IoT es clara la tendencia incremental a proporcionar soluciones de este tipo, donde la inteligencia artificial está llamada a jugar un papel esencial. Por añadir algún ejemplo más a los anteriores, y ya específicos en IoT cabe señalar los siguientes:

- La empresa ENVIRA proporciona soluciones inteligentes que permiten determinar los niveles de contaminación acústica, para que las administraciones puedan conocer los análisis en tiempo real, a la vez que optar por soluciones que puedan paliar sus efectos (ENVIRA, 2020).
- INTELKIA también proporciona soluciones para combatir la contaminación por ruido con soluciones integrables en plataformas en el ámbito de las Smart Cities (INTELKIA, 2020).

Teniendo en cuenta lo anterior, el presente trabajo también se orienta en el análisis de ruido mediante una aplicación inteligente basada en redes neuronales convolucionales, a la vez

que se establecen los mecanismos necesarios para establecer los accesos correspondientes a la nube dentro del paradigma IoT. El trabajo se orienta específicamente en la identificación de sonidos de motosierra, con fines de prevención de actividades ilegales en hábitats forestales, si bien la misma es aplicable a cualquier tipo de sonidos.

1.2 Objetivos

El objetivo general que se propone en el presente trabajo es diseñar un sistema IoT, como una solución de concepto factible para la identificación de sonidos en entornos naturales. Obviamente, dada su naturaleza no resulta posible su despliegue en un entorno real, así pues, los datos objeto de análisis se obtienen a partir de audios convenientemente seleccionados. De esta manera, se realiza una propuesta que proporciona una solución para su posterior implantación en entornos reales, sin más que sustituir los citados datos proporcionados por un dispositivo equipado con un micrófono con capacidad de generar audios en tiempo real.

Como objetivos específicos se plantean los siguientes,

- Diseño de un procedimiento de preparación de los datos procedentes de los audios disponibles.
- Desarrollo de un procedimiento para extracción de espectrogramas de la señal de audio, involucrando técnicas de análisis de la señal y su transformación al espectro de frecuencia mediante la transformada de Fourier.
- Diseño a nivel local de una red neuronal, que acepta como entrada los espectrogramas, considerados como imágenes espectrales, en el sentido indicado.
- Configuración de la plataforma remota (Matlab Drive) para el entrenamiento y ejecución de la red, con acceso desde un dispositivo móvil.
- Diseño y configuración de la plataforma remota, en la nube, para almacenamiento y envío de alarmas mediante Twitter.
- Integración de los módulos que implementan los procesos anteriores.
- Análisis y valoración de los resultados.

El plan de trabajo previsto tiene como base los objetivos específicos reseñados, contemplando el mismo número de fases que objetivos, y con una planificación equilibrada por semanas a lo largo del tiempo de desarrollo del proyecto.

Por otra parte, los propios objetivos determinan la aportación específica al proyecto como una solución integrada.

1.3 Motivaciones y contribuciones

Una vez establecido el planteamiento inicial y los objetivos a conseguir, la motivación del presente proyecto es doble. Por un lado, la necesidad de desarrollar un sistema de reconocimiento de sonidos en general y por otro lado la identificación de sonidos, en este caso de motosierra, en un entorno natural para detectar actividades ilegales, tales como la tala de incontrolada de árboles en bosques protegidos. Todo ello mediante la aplicación de técnicas inteligentes, concretamente en el ámbito del aprendizaje profundo utilizando Redes Neuronales Convolucionales (RNC) y bajo el paradigma IoT.

Como se ha indicado previamente, se proporciona una solución de concepto, que permite determinar su viabilidad para su implantación en entornos reales, sin más que adaptar los dispositivos de captura de datos y las comunicaciones para la transmisión de los mismos o los resultados del procesamiento en su caso. Esto contribuirá, sin duda, al sostenimiento medioambiental por la identificación de las mencionadas actividades, a la vez que establece las bases para su despliegue en cualquier entorno, de exterior o interior, cuya finalidad sea la identificación de sonidos o niveles de ruido con la finalidad que se establezca en cada caso.

Bajo estas premisas, las contribuciones específicas realizadas en el presente proyecto se sintetizan como sigue:

1. Análisis de viabilidad del modelo a implementar, definiendo la estructura de la aplicación, los dispositivos de captura de datos y herramientas de procesamiento a utilizar.
2. Análisis de la estructura del modelo y estudio de la viabilidad respecto a la transmisión y procesamiento de los datos desde su entrada al sistema hasta su llegada a la nube. Se consideran las interfaces y las posibilidades de conexión a través de internet.
3. Estudio y análisis de métodos de procesamiento de señales de audio, mediante la generación de espectrogramas, que se procesan como imágenes y constituyen los datos de entrada al modelo de la red neuronal.
4. Estudio y adaptación de modelos de arquitecturas de redes neuronales convolucionales en el ámbito del procesamiento de imágenes con el fin de procesar los espectrogramas, considerados como imágenes.
5. Generación de datos de audio, tanto para el entrenamiento de la red, como para la clasificación de los mismos tras el entrenamiento.
6. Diseño del modelo de almacenamiento y procesamiento en la nube, teniendo en cuenta que se utilizan plataformas remotas, como se verá posteriormente (Drive y ThingSpeak).

7. Implementación de los distintos módulos que componen el sistema e integración de los mismos para configurar una única plataforma, con acceso a dispositivos sensoriales.
8. Validación de los módulos y del modelo integrado.

1.4 Organización de la memoria

La memoria se ha organizado en cinco capítulos, con el siguiente contenido en los capítulos restantes. El capítulo dos describe la metodología aplicada para el desarrollo de la solución. El capítulo tres describe el diseño de la solución, la cual comprende la arquitectura de la solución y el procesamiento de datos. En el capítulo cuatro se presenta el análisis de los resultados obtenidos. Finalmente, en el capítulo cinco se presentan las conclusiones y el trabajo futuro que se podría efectuar.

Capítulo 2 - Descripción de la metodología aplicada

Tal y como se describe en los objetivos, se plantea el desarrollo de una solución de concepto en el ámbito del IoT. Con tal propósito, se plantea el desarrollo de las técnicas que se describen a continuación, y que incluyen: a) Segmentación de la señal de audio; b) Reconocimiento de señales de audio mediante técnicas de aprendizaje profundo basadas en RNC.

2.1 Segmentación de la señal de audio

a) Transformada de Fourier

Dada una secuencia discreta de entrada $\{x_k\}$, la Transformada Discreta de Fourier (*TDF*) se define tal y como se expresa en la ecuación siguiente,

$$X(w) = \sum_{k=-\infty}^{\infty} x_k e^{-jwk} \tag{2.1}$$

Se trata de una señal periódica de periodo *2p*, que además permite definir la Transformada (discreta) Inversa de Fourier (TIF) como sigue,

$$x_k = \frac{1}{2\pi} \int_{-\pi}^{\pi} X(w) e^{jwk} dw \tag{2.2}$$

En un sistema dado, la transformada de Fourier permite derivar la relación existente entre una entrada $\{x_k\}$ proporcionada al sistema y la salida $\{y_k\}$ según (2.3).

$$Y(w) = G(w)X(w) \tag{2.3}$$

De esta forma, conocida la respuesta en frecuencia del sistema y la transformada discreta de la secuencia de entrada, se calcula la transformada de la secuencia y se determina la salida mediante (2.3), aplicándose posteriormente la transformada inversa según la ecuación (2.2).

Por ejemplo, considérese una respuesta en frecuencia dada por la ecuación (2.4),

$$G(w) = \sum_{0}^{2} h_k e^{-jwk} = 0.15 + 0.5e^{-jw} + 0.75e^{-j2w} \tag{2.4}$$

y supóngase la secuencia de entrada $\{x_k\} = \{1,1, 0, 0, ...\}$. En primer lugar, se calcula la TDF de la secuencia de entrada,

$$X(w)=\sum_{k=0}^{1} x_k e^{-jwk} = 1+e^{-jw} \tag{2.5}$$

Aplicando la ecuación (2.3) se tiene,

$$\begin{aligned} Y(w) &= \left(0.15+0.5e^{-jw}+0.75e^{-j2w}\right)\left(1+e^{-jw}\right) = \\ & 0.15+0.65e^{-jw}+1.25e^{-2jw}+0.75e^{-3jw} \end{aligned} \tag{2.6}$$

Siendo en este caso, la secuencia de salida $\{y_k\}$ = {0.15, 0.65, 1.25, 0.75}. La idea que subyace tras la transformada de Fourier estriba en el hecho de la descomposición de una determinada señal en otras señales de distinta frecuencia de forma que la suma de éstas genere la primera. En la figura 2-1 se muestra la composición de dos señales sinusoidales y por tanto cómo la señal compuesta se puede descomponer en las otras dos.

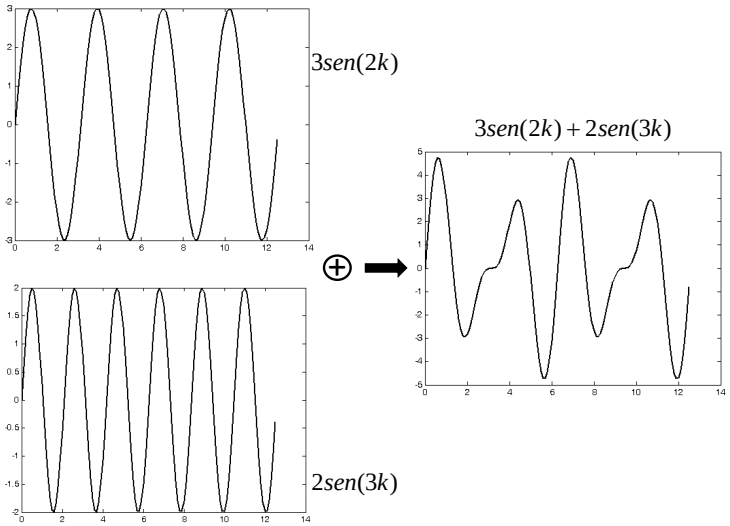


Figura 2-1 Composición de señales sinusoidales de distinta frecuencia

Teniendo en cuenta la ecuación (2.1) y considerando que el número de muestras de la ventana o parte de la señal que se va a analizar es *M* y asumiendo que $w=2\pi f=2\pi T^{-1}$ la *TDF* puede expresarse como,

$$X(u)=\frac{1}{M}\sum_{k=0}^{M-1} x_k e^{-j2\pi uk/M} \quad \text{con u = 0,1, 2, ..., M-1} \tag{2.7}$$

El concepto de dominio de la frecuencia se deriva directamente de la fórmula de Euler, esto es,

$$e^{j\theta}=\cos\theta+j\sin\theta \tag{2.8}$$

con lo cual la ecuación (2.7) se puede reescribir teniendo en cuenta lo anterior y que *cos(θ)=cos(θ)*,

$$X(u) = \frac{1}{M}\sum_{k=0}^{M-1} x_k \left[\cos 2\pi uk/M - j\sin 2\pi uk/M\right] \quad (2.9)$$

con u = 0,1, 2, …, M-1. De esta forma, se puede observar que cada término de la transformada de Fourier (esto es, el valor de *X(u)* para cada valor de u) se compone de la suma de todos los valores de x_k. En definitiva, los valores de x_k se multiplican por senos y cosenos de varias frecuencias. Los valores de *u* o dominio sobre el que *X(u)* varía se denomina apropiadamente dominio de frecuencia, ya que *u* determina la frecuencia de las componentes de la transformada. Los valores x_k de la señal también afectan a las citadas frecuencias, si bien la contribución de todas ellas es la misma para cada valor de *u*. Cada uno de los *M* términos de *X(u)* se denomina componente de frecuencia de la transformada.

Metafóricamente hablando, la transformada de Fourier puede compararse con un prisma óptico que separa la luz en sus componentes de color primarios según su longitud de onda (frecuencia). De este modo, la transformada de Fourier puede verse como un prisma matemático que separa una señal en sus diferentes componentes de frecuencia, según el contenido de las propias frecuencias.

b) Procesamiento de la señal mediante ventanas

En lo que respecta al reconocimiento de sonidos, el tratamiento de la señal conlleva normalmente el análisis de ésta mediante distintos métodos de procesamiento, entre ellos destacan los procedimientos basados en ventanas. El proceso consiste en definir una secuencia de duración limitada en el tiempo denominada ventana (McLoughlin 2009; Pajares 2017). Dicha secuencia está formada por un número determinado de coeficientes, que se desplaza a lo largo de la señal digital, haciéndola coincidir por tramos sobre dicha señal. La ventana posee un ancho dado, *L*, pudiéndose desplazar con un cierto solapamiento *a*. La figura 2-2 muestra un esquema ilustrativo relativo al procedimiento mencionado. En efecto, se inicia el proceso en un determinado punto, donde la citada ventana se solapa sucesivamente sobre la señal coincidiendo en la parte definida por el parámetro *a* entre los desplazamientos sucesivos. En cada posición se realizan las operaciones pertinentes entre los coeficientes de la señal abarcados por la ventana y los propios coeficientes de ésta. Los desplazamientos vienen determinados por el ancho de la señal, la cual se identifica mediante la variable temporal *t*, que en el caso de la figura 2-2 se representa sobre el eje de abscisas.

El uso de ventanas tiene su justificación en hechos evidentes que pueden surgir en el análisis de determinadas situaciones. Supóngase que se ha captado un ruido durante un cierto periodo de tiempo, pudiendo ocurrir que en un momento dado aparece un ruido espurio procedente de cualquier fenómeno, tal como el paso de un vehículo. Es evidente que, si se analiza el conjunto de la señal, el valor medio de la misma se incrementará por la aparición de cualquiera de los eventos señalados. Sin embargo, mediante un análisis por tramos, dicho valor sólo se verá incrementado en el tramo donde se haya producido el evento. Por tanto, la señal contiene dos eventos significativos: el ruido de la motosierra con intensidad normal y la parte correspondiente a la zona de alta intensidad. Resulta evidente cómo la señal cambia bruscamente del estado del ruido normal al del vehículo. Esto se puede ver como un cambio de estado entre los sucesivos tramos de la señal. En estas situaciones no tiene mucho sentido considerar el promedio de intensidad de toda la señal sino la intensidad por segmentos. Bajo estos conceptos subyace la principal idea del análisis de la señal por tramos, proceso que en la literatura especializada se conoce como *short-term*.

De este modo, supóngase la señal discreta *x(n)* con *n = 0,…,N*. Durante el procesamiento por ventanas el foco de atención se pone en una pequeña parte de dicha señal, de forma que se realiza la multiplicación de las muestras de la señal con los coeficientes de la ventana que aparecen solapados y por tanto coincidiendo temporalmente.

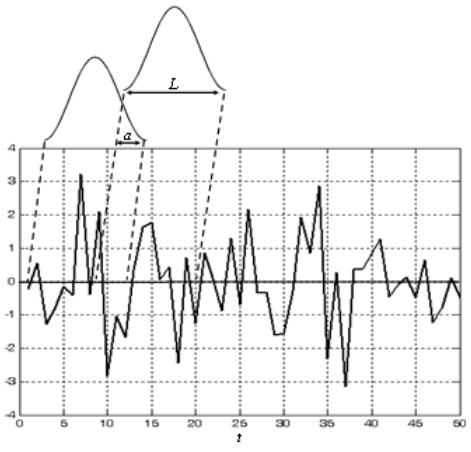


Figura 2-2 Ventanas de procesamiento desplazándose a lo largo de una señal unidimensional bajo análisis

Para la definición de las mencionadas ventanas se utilizan diferentes funciones, destacando las siguientes, por ser de amplia utilización en reconocimiento de señales de audio:

- *Ventana rectangular:*
  $V(n) = 1$ en el intervalo de duración del bloque $(0 \leq n \leq L-1)$
  $V(n) = 0$ en el resto de la señal
- *Ventana de Hamming:*
  $V(n) = 0.54 - 0.46\cos(2\pi n/L)$ en el intervalo $(0 \leq n \leq L-1)$
  $V(n) = 0$ en el resto de la señal
- *Ventana de Hanning:*
  $V(n) = 0.55-0.5\cos(2\pi n/L)$ en el intervalo $(0 \leq n \leq L-1)$
  $V(n) = 0$ en el resto de la señal
- *Ventana Gaussiana:*
  $V(n) = e^{-n^2/\sigma^2}$ en el intervalo $(0 \leq n \leq L-1)$
  $V(n) = 0$ en el resto de la señal

En el caso de la ventana Gaussiana el parámetro *s* representa la desviación estándar, de suerte que cuanto mayor es su valor tanto mayor será la apertura de la ventana y por consiguiente mayor será también el número de coeficientes de la señal que quedan bajo la citada ventana.

En la figura 2-3 se muestran, de forma solapada, las gráficas correspondientes a las cuatro ventanas definidas previamente. Una característica común de todas ellas es que sus anchos son controlables mediante el parámetro *L*. Este parámetro resulta ser esencial ya que muchas propiedades de los elementos de la señal van a depender del ancho de la ventana. En efecto, supóngase el caso en el que la ventana ocupa toda la señal. El resultado del análisis será diferente en el supuesto de que la ventana sea de menor ancho y sólo ocupe el correspondiente a un sonido determinado. En el primer caso se analiza la señal al completo y de una sola vez mientras que en el segundo se obtiene resultados diferentes para cada tramo de la señal analizada.

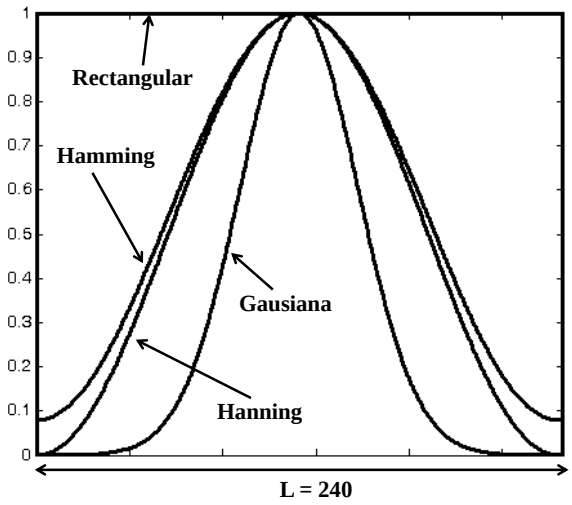


Figura 2-3 Diferentes formas de ventanas para análisis de señales de ruido

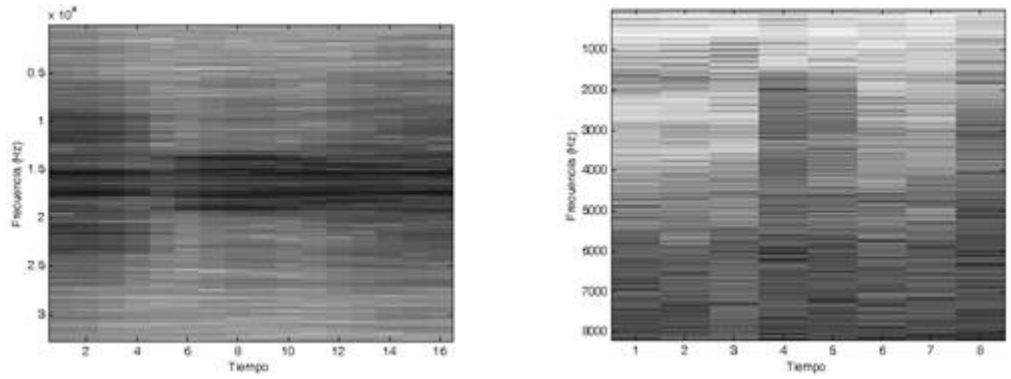
c) Espectrogramas

La transformada de Fourier se puede aplicar de forma que, dada una señal, ésta se descomponga en tramos y sobre cada tramo se aplique una operación basada en procesamiento con ventanas. Así, sobre cada uno de estos tramos se aplica la *TDF* para un determinado rango de frecuencias obteniendo los coeficientes de la propia transformada para cada tramo bajo análisis, que constituye lo que se conoce como espectrograma de la señal (Giannakopoulos y Pikrakis, 2014).

Es decir, tomando como referencia la ecuación (2.3), ésta indica que para cada valor de frecuencia *w*, se obtiene una respuesta diferente del filtrado. Es lo que se conoce como banco de filtros. Esto permite obtener sobre una misma señal de audio distintas respuestas para cada frecuencia, de forma que en su conjunto constituyen el espectrograma completo.

En la figura 2-4(*a*) se muestra un espectrograma de frecuencias con los valores escalados en el rango [0,1] y mostrados en distintas tonalidades, también conocido como sonograma, que corresponde al análisis de una determinada señal, la cual se ha dividido en 16 tramos (eje horizontal). Sobre ella se ha aplicado una operación basada en la ventana de tipo Gaussiano y se han obtenido los correspondientes coeficientes para el rango de frecuencias mostrado en el eje vertical que varía desde 0 a 32768 Hz. En la figura 2-4(*b*) se muestra el correspondiente espectrograma o sonograma para la misma señal en la que el número de tramos analizados es 8, habiéndose utilizado una ventana de Hamming con un solapamiento del 50% entre ventanas consecutivas para cada tramo y un rango de frecuencias de 0 a 8000Hz.

Un aspecto importante a destacar respecto de los mencionados estereogramas es su dimensionalidad, que es del tipo *M*×*N*×1, siendo respectivamente las dimensiones ancho, alto y profundidad. Esto define las características de la capa de entrada a la red neuronal, como se indica posteriormente.



(a) (b)

Figura 2-4 Espectrogramas (sonogramas) mediante la TDF: (a) 16 tramos; (b) 8 tramos

En la figura 2-5 se muestran sendas señales de audio correspondientes a dos tipos de sonidos diferentes, concretamente uno correspondiente a motosierra (a) y otro a un sonido desconocido con ruido de fondo. En la parte inferior de dichas figuras se muestran los correspondientes espectrogramas en escala de colores según la magnitud espectral, donde los valores con magnitud alta se muestran en tonalidades de color más intensas y viceversa. En el caso del audio correspondiente a la motosierra presenta características espectrales bien diferenciadas con respecto a las que aparecen en el caso del sonido desconocido. Estas características, son las que permiten a la red ajustar sus pesos en función del tipo de sonido procesado. En los ejemplos mostrados el número de bandas sobre los que se computa el espectrograma es de 40, para tramos de duración de 1 segundo y teniendo en cuenta que dado que la frecuencia (en Hertzios) de muestreo de las señales es de 16000 Hz, el número de muestras es de 16000, como aparece el eje de abscisas en la parte superior izquierda. En el eje de ordenadas de estas mismas figuras se muestra la magnitud de la señal. Respecto de los espectrogramas mostrados en la parte inferior, indicar que en el eje de abscisas se representan los 40 tramos (parte central de la figura) en los que se divide la señal de duración 1s. En el eje de ordenadas, se representan los valores obtenidos por aplicación del filtrado con distintas frecuencias (w), o banco de filtros, para cada uno de los 40 tramos, siendo en este caso también de 40 frecuencias, y por tanto w = 40.

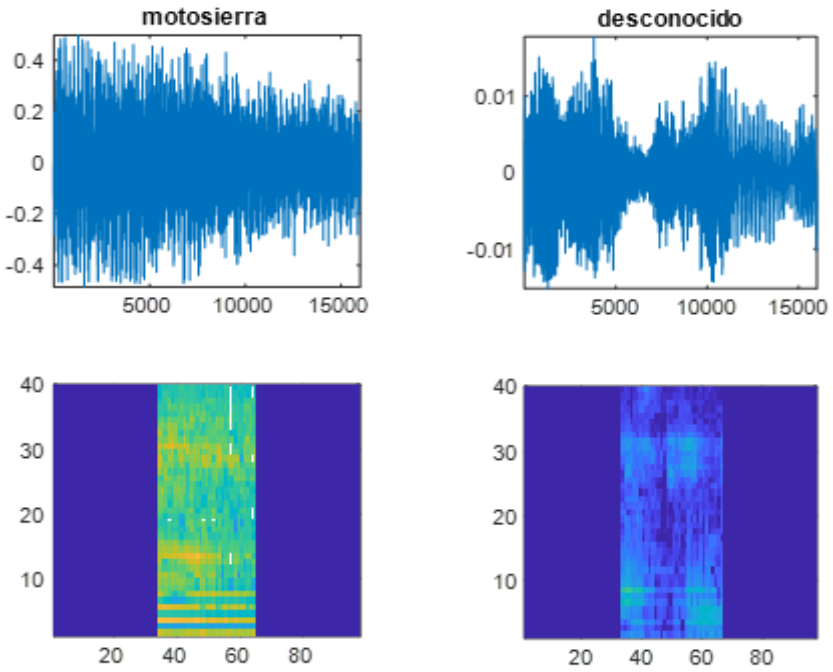


Figura 2-5 Espectrogramas (sonogramas) mediante la TDF con 40 tramos

2.2 Reconocimiento de señales de audio mediante técnicas de aprendizaje profundo basadas en Redes Neuronales Convolucionales

Una Red Neuronal Convolucional (RNC) es un tipo específico de red neuronal, cuyo fundamento son las denominadas capas de convolución, de ahí su nombre. El objetivo, consiste en aprender determinados valores o pesos en los denominados núcleos de convolución, como se verá posteriormente, teniendo en cuenta los datos de entrada, en este caso espectrogramas, obtenidos a partir de las señales de audio como se describe previamente y proporcionados durante la fase que se conoce como de *entrenamiento*.

Como se ha indicado previamente, los espectrogramas se estructuran con sus tres dimensiones que los definen y que, bajo esta perspectiva, pueden considerarse como imágenes de gris con un único canal, definido por su dimensión de profundidad, y sus correspondientes ancho y alto definidos por *M* y *N*. Por tanto, de aquí en adelante, y desde el punto de vista de definición del modelo de red, los espectrogramas se asimilan a una imagen mono-canal.

En el ámbito del tratamiento de imágenes, las RNC están ampliamente implantadas, por poner sólo un ejemplo, considérese el modelo AlexNet (ImageNet, 2019; Russakovsky y col., 2015; Krizhevsky y col., 2012), que ganó la competición ILSVRC-2012 con una tasa de error de un 15.3% frente al segundo competidor que obtuvo un 26.2%. Este tipo de modelos se caracteriza por estar previamente entrenados (modelo pre-entrenado) con un ingente número de imágenes, siendo del orden de más de doce millones, y para clasificar 1000 tipos de objetos diferentes. En este sentido, cuando se utiliza su arquitectura como modelo, resulta habitual aprovechar los pesos que posee el modelo original pre-entrenado, y realizar un rediseño de las últimas capas de salida para adaptar la red a la aplicación concreta sobre la que se aplica. Por ejemplo, si en esta aplicación concreta se trata de identificar cuatro objetos, la última capa de salida sólo poseerá 4 neuronas, en lugar de las 1000 del modelo original. Una vez hecho esto, la red se re-entrena para ajustar sus pesos a las nuevas imágenes, que constituyen la base de aplicación del modelo. Esto es lo que en términos científicos se conoce como transferencia de aprendizaje.

En el caso de la aplicación de reconocimiento de sonidos, que se plantea en este trabajo, no es posible realizar la mencionada transferencia de aprendizaje, entre otras razones porque en este caso no se trata de reconocer imágenes propiamente dichas, sino espectrogramas asimilados a imágenes. Desde el punto de vista de diseño del modelo, tampoco es factible dado que las entradas de los modelos pre-entrenados son imágenes multicanal, en lugar de mono-canal como es el caso del presente trabajo. Por tanto, en el modelo de red que se plantea en este trabajo se utiliza la arquitectura del modelo AlexNet por su buen rendimiento, sin ningún tipo de transferencia de aprendizaje, lo que supone realizar el entrenamiento desde el inicio y por tanto con los pesos de la red con valores aleatorios, que se ajustan en función de los espectrogramas que se proporcionen como datos de entrenamiento.

En cualquier caso, los modelos RNC poseen un determinado número de capas, dependiendo de la arquitectura elegida en su definición. En cada capa se aplican operaciones de convolución con núcleos de distintos tamaños según la nomenclatura *m* x *m*. Los indicadores *K, p* y *s* que aparecen en las diferentes capas, hacen referencia respectivamente al número de filtros (con su correspondiente núcleo de convolución) aplicados, al número de ceros añadidos en las filas y columnas externas del espectrograma (*padding*) y unidades de desplazamiento del núcleo (*stride*), también sobre el espectrograma. La entrada es un espectrograma de dimensión 40x98x1 y el modelo de red que se plantea es el que se muestra en la figura 2-6, definido en MathWorks (2019). Se trata de un modelo de 24 capas con las características que se describen a continuación y que comprenden 5 de convolución, 5 de normalización, 5 *ReLU*, 4 *Maxpool*, 1 *Dropout*, 1 totalmente conectada (fc), 1 *Softmax* haciendo un total de 22, que junto con las dos

correspondientes a la entrada y salida hacen el total de las 24. En el modelo mostrado en la figura 2-6 tras cada operación, y por consiguiente tras cada conjunto de operaciones, se obtiene un mapa de características como un volumen 3-D hasta llegar a la salida para el número de clases en cuestión, siendo tres según la arquitectura de la red mostrada, dado que en este trabajo son tres las clases que se manejan para la clasificación de los audios.

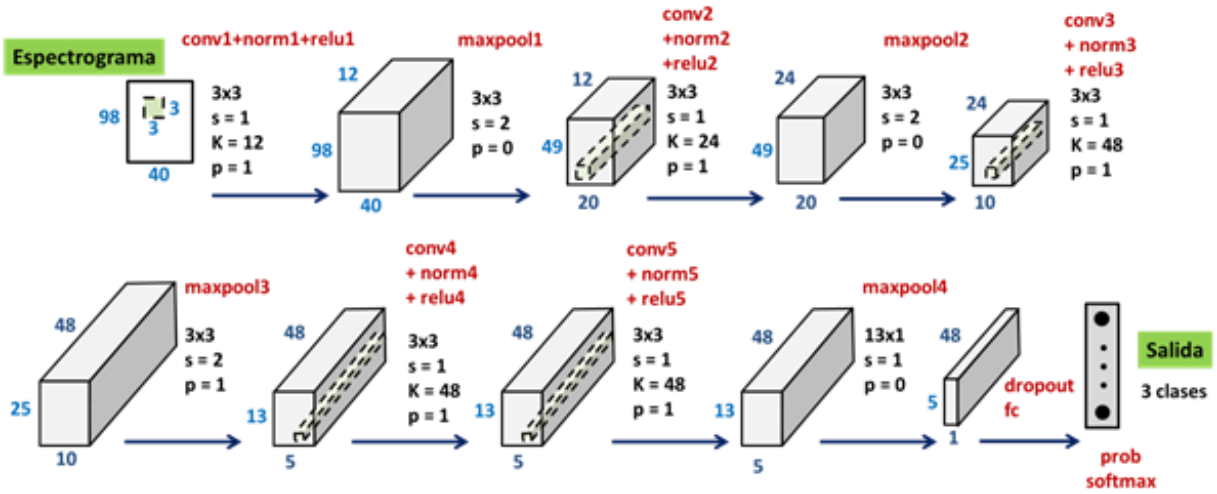


Figura 2-6 Modelo de Red Neuronal Convolucional con 24 capas

Las diferentes operaciones involucradas en el modelo anterior se describen a continuación:

Convolución (*conv*), esta operación matemática combina dos funciones para generar una tercera, consiste básicamente en superponer una función sobre otra modificada para obtener una serie de características.

$$S(i,j) = (I * K)(i,j) \tag{2.10}$$

donde el argumento I se conoce como entrada (*input*), que es un vector o matriz multidimensional. El segundo argumento K, conocido como filtro (*kernel*) de convolución, es una matriz de pesos que se obtienen durante el proceso de aprendizaje. Ambas entradas son conocidas como tensores. La salida S se define generalmente como mapa de características (*feature map*). Los núcleos de convolución se muestran gráficamente en la figura 2-6 dentro de los mapas de características, mientras que sus dimensiones se identifican en la parte superior de cada anotación, que en el modelo presentado son de 3x3. Por otra parte, el símbolo K indica el número de núcleos de convolución utilizados en cada caso. Así por ejemplo para realizar la primera convolución se utiliza K = 12, lo que origina el volumen de características a la salida de

esta misma dimensión. Lo mismo ocurre para el resto de convoluciones con valores de K de 24 y 48 en varias capas.

La operación de convolución en las RNC, como su propio nombre indica, es un elemento clave, de ahí su importancia en ellas. Dado que en el presente trabajo se utilizan espectrogramas de los sonidos, que se tratan como imágenes de entrada a la RNC, y por tanto con dimensión 2-D, a continuación se ilustra gráficamente el concepto de convolución con solapamiento total del núcleo y obteniendo una imagen resultante, cuyos bordes externos se quedan sin valores, obteniendo una imagen de menor dimensión que la original, en el caso de la figura se pasa de tener dimensión 6×7 a 4×5. No obstante, si se desea obtener una imagen de la misma dimensión lo que hay que hacer es ampliar la imagen original en dos filas (arriba y abajo) y dos columnas (izquierda y derecha) con ceros, procesándola con el núcleo solapado. Esta operación es lo que se conoce como *zero-padding* y generalmente se indica en términos de implementación como '*same*'. Por el contrario, si no se realiza tal operación, es decir, no se añaden ceros, en implementación se indica como '*valid*', en referencia a este último caso, más adelante se indica que la operación es sin *zero-padding*.

Por otra parte, en las convoluciones, el campo receptivo se define como la región de entrada que contribuye a la salida generada por el filtro. En la figura 2-7 se muestran sendos campos receptivos de la imagen I que contribuyen a las salidas P y Q generadas por el filtro K.

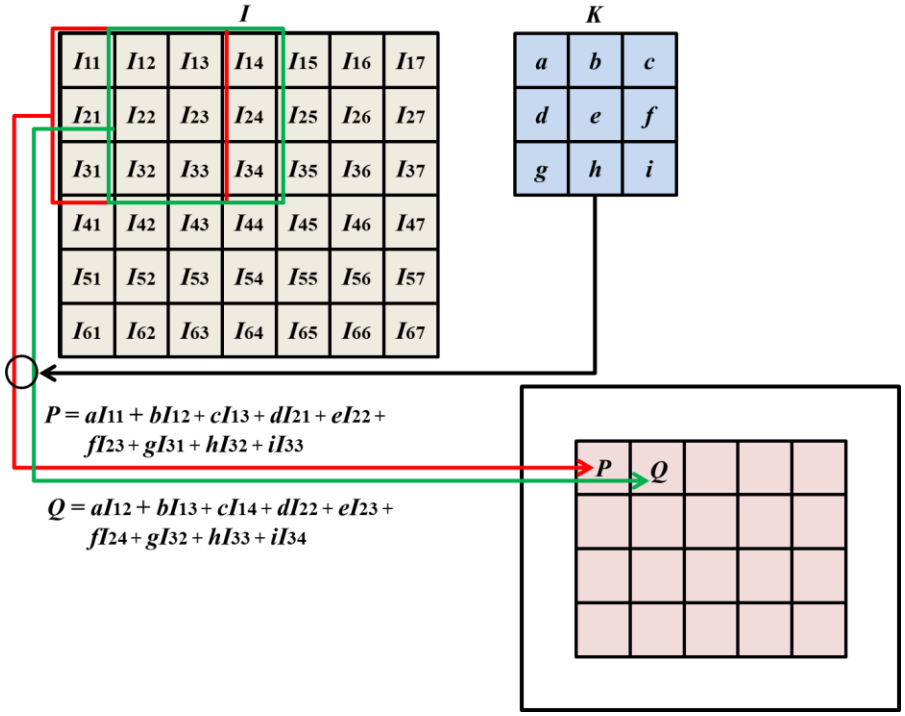


Figura 2-7 Ejemplo de convolución 2D

Como se ha indicado previamente, la convolución de la figura 2-7 es de tipo 2-D. También podría ser de tipo 3-D, de forma que el núcleo sería un cuboide que se desplaza a través de las dimensiones alto, ancho y alto del mapa de características. Por ejemplo, si se tiene una imagen RGB y por tanto con tres canales para la capa de entrada se utilizan núcleos de tipo cuboide y por tanto con tres dimensiones. Esto puede generalizarse a convoluciones de tipo N-D cuando el número de dimensiones es N.

En la figura anterior, los desplazamientos del núcleo son de una posición a la siguiente, pero en lugar de desplazarse una posición de izquierda a derecha y de arriba abajo, el desplazamiento podría ser de más de una unidad, es lo que se conoce como *stride*, que también interviene en el cómputo de la dimensión de salida de la imagen, como se explica más adelante.

Así pues, la colección de núcleos que definen una convolución discreta tiene una forma que se corresponde a alguna permutación del tipo siguiente ($n, m, k_1, \ldots, k_N$), donde:

n ≡ número de mapas de características de salida,

m ≡ número de mapas de características de entrada,

kj ≡ dimensión del núcleo a lo largo del eje j.

La dimensión o_j de una capa de convolución a lo largo del eje j tiene las siguientes propiedades,

i_j ≡ dimensión de entrada a lo largo del eje j,

k_j ≡ dimensión del núcleo a lo largo del eje j.

s_j ≡ distancia entre dos posiciones consecutivas del núcleo (*stride*) a lo largo del eje j.

p_j ≡ número de ceros concatenados al comienzo y al final del eje j (*zero-padding*).

La figura 2-8 muestra un ejemplo ilustrativo sobre el proceso de convolución bi-dimensional (N = 2), con $i_1 = i_2 = 5$ como tamaño 5×5 de la imagen de entrada (I) y dimensión 3×3 ($k_1 = k_2 = 3$) del núcleo de convolución (K), con desplazamiento (*stride*) de uno en ambas direcciones ($s_1 = s_2 = 1$) y cero *padding* ($p_1 = p_2 = 0$). Generalmente, suele ser habitual que los valores de i, k, s y p en todos los ejes tengan el mismo valor.

I

3	2	1	4	2
5	1	3	1	3
2	1	5	3	2
2	4	3	4	0
1	3	2	2	3

K

1	2	3
2	4	1
2	3	1

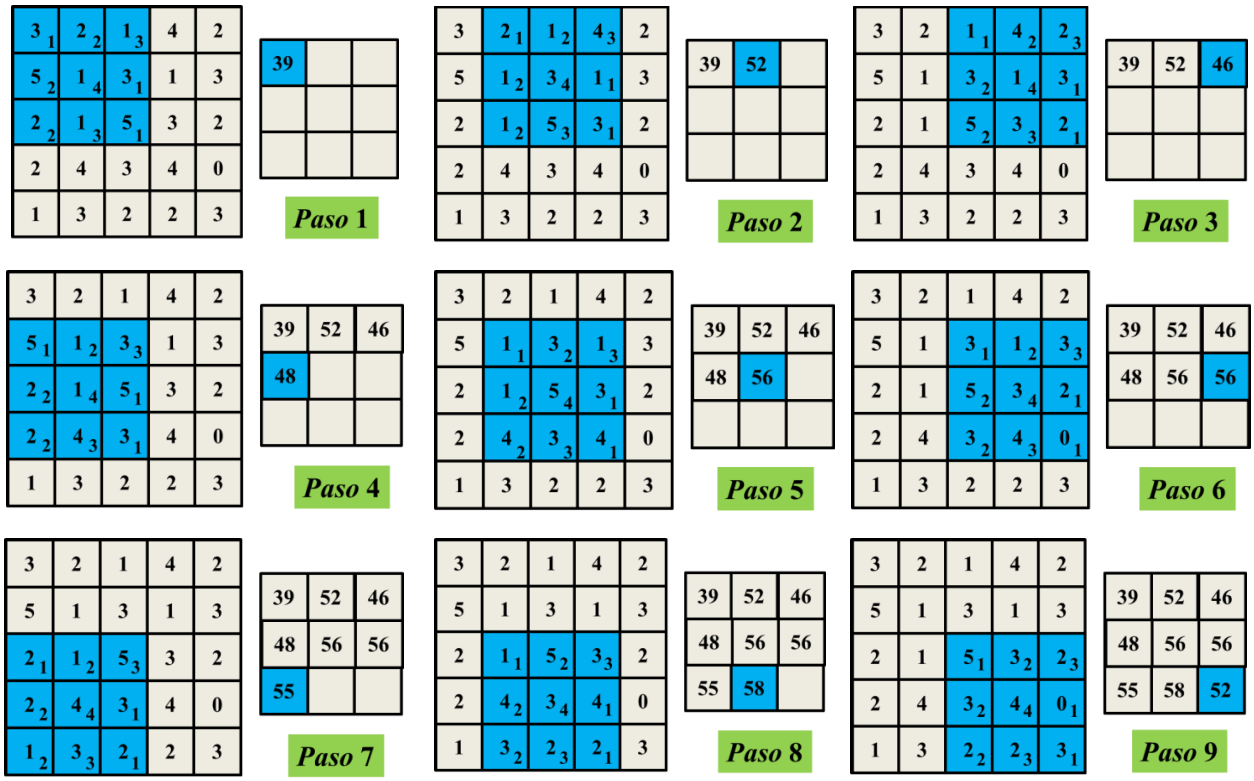
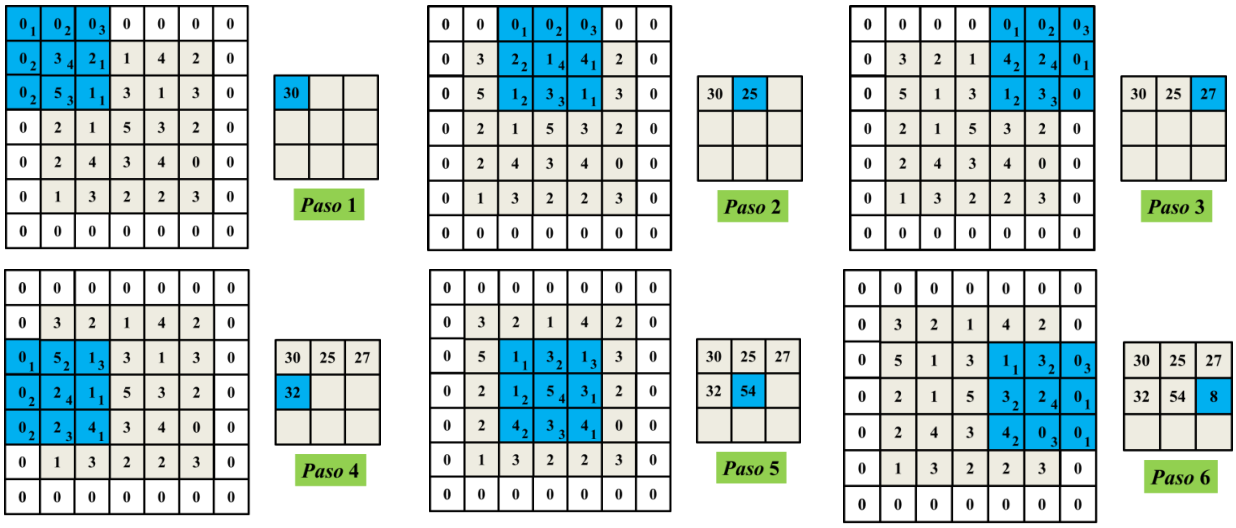


Figura 2-8 Ejemplo de una convolución discreta con N=2, i_1= i_2=5, k_1=k_2=3, s_1=s_2=1, p_1=p_2=0

La figura 2-9 muestra un ejemplo ilustrativo sobre el proceso de convolución bi-dimensional (N = 2), con $i_1 = i_2 = 5$ como tamaño 5×5 de la misma imagen de entrada (I) y dimensión 3×3 ($k_1 = k_2 = 3$) del mismo núcleo de convolución (K), con desplazamiento (*stride*) de dos en ambas direcciones ($s_1 = s_2 = 2$) y una extensión de *padding* ($p_1 = p_2 = 1$).



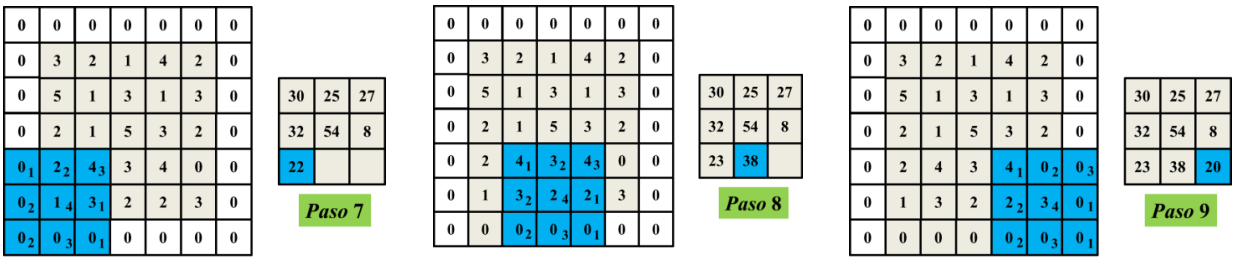


Figura 2-9 Ejemplo de una convolución discreta con N=2, i_1=i_2=5, k_1=k_2=3, s_1=s_2=2, p_1=p_2=1

ReLU (*relu*), es una función de activación, conocida como *Rectified Linear Unit,* de forma que, por cada resultado generado por la convolución, la rectifica linealmente devolviendo 0 para valores negativos y el propio valor de entrada para valores positivos, definida como f(x) = max(0, x), con x siendo la entrada a la neurona, y cuya representación gráfica se muestra en la figura 2-10.

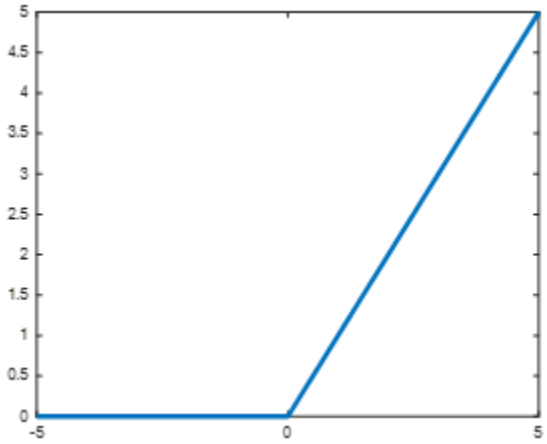


Figura 2-10 Función ReLU

Normalización (*norm*), la operación de normalización local sirve para acelerar la convergencia durante el proceso de aprendizaje. Actualmente se usa la normalización por lotes en lugar de la de respuesta local, la función se define como sigue,

$$b_{x,y}^{i} = a_{x,y}^{i} \Big/ \left(k + \alpha \sum_{j=max(0,i-n/2)}^{min(N-1,i+n/2)} \left(a_{x,y}^{i} \right)^2 \right)^{\beta} \qquad (2.11)$$

donde N es el número de filtros de la capa dada, $a_{x,y}^{i}$ es la actividad de la neurona y k, α, n, β son hiperparametros. En Krizhevsky (2012) se proponen los siguientes valores para los parámetros k = 2, n = 5, $\alpha = 10^{-4}$, β = 0.75, siendo éstos los utilizados en el modelo de arquitectura mostrado en la figura 2-6.

Pooling (*pool*), la función de *pooling* modifica la capa de salida agrupando en una porción más pequeña las características, de tal forma que dada una pequeña traslación en la entrada no afecte a los valores de las salidas. Se define como se indica en la ecuación (2.12), teniendo en cuenta que las variables involucradas representan la dimensión de entrada (i), el tamaño del núcleo (*kernel*) de convolución o de *pooling* (k, en ambos casos), el desplazamiento del núcleo a lo largo de la imagen (s, *stride*). Los corchetes representan la función *floor* (mayor entero menor que al argumento de entrada) y en ocasiones *ceil* (menor entero mayor que el argumento de entrada).

$$o = \left[\frac{i + 2p - k}{s} \right] + 1 \tag{2.12}$$

Considerando por ejemplo la primera convolución (conv1) donde la entrada es 90x40, con p =1, s =1 y con tamaño del núcleo de convolución de 3x3, y por tanto, k = 3 en ambas dimensiones ancho y alto. Las salidas según la ecuación (8), resultan ser [(98+2-3)/1]+1 = 98 y [(40+2-3)/1]+1 = 40 para las dimensiones alto y ancho respectivamente del espectrograma de entrada, lo cual, junto con la aplicación de un conjunto de K = 12 núcleos de filtros produce un volumen para el mapa de características de 90x40x12. Las dimensiones de este volumen no se ven afectadas por las operaciones subsiguientes de normalización (*norm1*) y ReLU (*relu1*). Continuando con el ejemplo de aplicación, considérese la operación *maxpool1*, en este caso p=1, s = 2 y k = 3, donde k en esta operación se refiere a la dimensión de extensión de la operación de agrupamiento (*pooling*), que en este caso resulta para ambas dimensiones alto y ancho aplicando la operación *ceil*: [(98+0-3)/2]+1 = 49 y [(40+0-3)/2]+1 = 20. Para el resto de operaciones se aplican criterios similares. Únicamente, reseñar que, en el caso del modelo mostrado, la cuarta operación de *maxpooling* difiere ligeramente del resto, en el sentido de que para su realización se aplica una dimensionalidad de 13x1 sobre las 48 capas, resultando así un volumen para el mapa de características a la salida de 5x1x48.

La operación de *pooling* es otra de las operaciones esenciales en las RNC, que se aplica para reducir la dimensionalidad de un determinando mapa de características, de forma que en cada paso, la posición de la ventana se actualiza de acuerdo a los desplazamientos (*strides*). Cuando se sitúa la ventana en las proximidades de los píxeles de borde algunos de los elementos de la ventana pueden quedar fuera de los elementos de entrada. Como en el caso de la convolución, para obtener los valores de las regiones de borde, la entrada puede extenderse con valores de cero, lo que previamente se ha denominado *zero-padding*. Si bien, puede optarse por no aplicar este tipo de extensión en cuyo caso se dice que la operación se realiza sin *zero-padding*. Por tanto, al igual que en el caso de la convolución y desde el punto de vista

de la implementación, cuando se aplica *zero-padding* se suele indicar como '*same*' y en caso contrario como '*valid*'. En determinadas representaciones gráficas de capas de red, esta última operación se suele indicar con *p*=0 o equivalentemente con el símbolo 'V' de '*valid*'. Por el contrario, en el primer caso, se indica exactamente el valor de *p* sin más especificación de simbología y por tanto, rara vez aparece 'S' refiriéndose a '*same*'. En la figura 2-11 se muestra un ejemplo ilustrativo de *max pooling* sin *zero padding* (V, *p* = 0) y *s* = 2. En este caso la ventana se deslaza de dos en dos posiciones en horizontal y vertical a partir del inicio, si bien al ser una operación de tipo V, la columna más a la derecha no interviene dado que las ventanas no se puedes solapar sobre ellas.

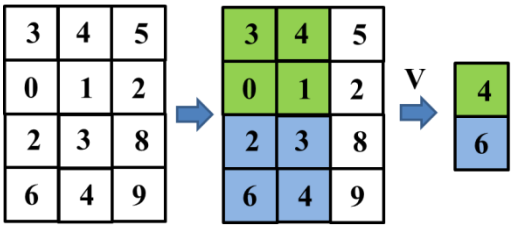


Figura 2-11 Max pooling sin cero padding (V) y s = 2

En la figura 2-12 se muestra también la misma operación, si bien ahora con *zero padding* (S, *p* = 1 en horizontal) y *s* = 2. En este caso la ventana en su desplazamiento en saltos de dos en dos ya puede incorporar a la columna de la derecha.

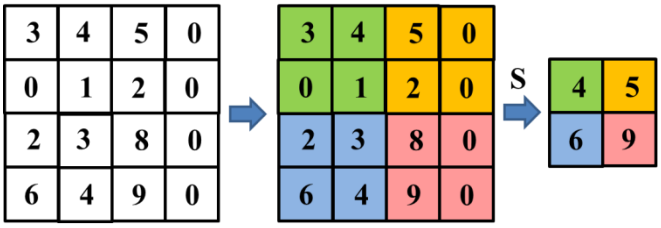


Figura 2-12 Max pooling con cero padding (S) y s = 2

Dropout (*drop*), sirve para no sobresaturar la red neuronal generando soluciones de polinomios de grado demasiado alto. Para ello se propone anular determinado tipo de neuronas de forma aleatoria mediante un hiperparámetro que indique la probabilidad de supervivencia de cada neurona. Las operaciones de este tipo aparecen indicadas como *drop*, correspondiendo en ambos casos al 20% de las neuronas anuladas, las cuales se eliminan de forma aleatoria teniendo en cuenta dicha restricción. Conviene señalar que esta operación es esenciala en los modelos RNC con el fin de evitar los sobreajustes (Srivastava y col., 2014; Krizhevsky, 2012). Como se ha indicaso previamente, la selección de las unidades a anular se realiza de forma aleatoria. Así, la red resultante es la que mantiene las unidades que sobreviven al dropout. No obstante, en lugar de eliminar neuronas, también cabe la posibilidad de

mantenerlas asignándoles un hipeparámetro de forma aleatoria que es en realidad una probabilidad *p* de supervivencia durante la fase de entrenamiento, que se utiliza para disminuir la influencia del peso en la fase de test. En la figura 2-13 se muestra un ejemplo gráfico de nodos cancelados (en rojo) y con el peso de activación $w_{(5,6)}$ atenuado por la probabilidad *p* durante la fase de test.

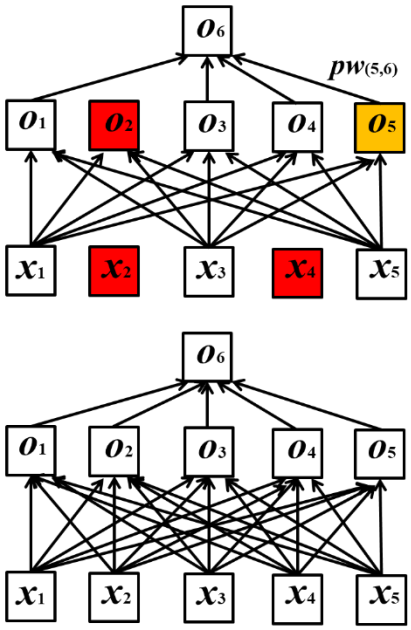


Figura 2-13 Dropout

En cualquier caso, el dropout puede aplicarse tras la salidad de una determinada capa como por ejemplo, tras un pooling como ocurre en el ejemplo gráfico mostrado en la figura 2-14.

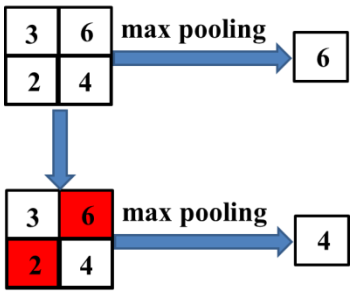


Figura 2-14 Dropout y max pooling

Softmax, se aplica la función exponencial a cada elemento del vector de entrada y se normalizan sus valores por la suma de las exponenciales, lo que asegura que los valores del vector de salida se sitúen en el rango [0,1], proporcionando así la probabilidad de pertenencia a cada

una de las clases para una imagen de entrada dada. Su función se define según la siguiente expresión.

$$softmax(\boldsymbol{x})_i = \frac{exp\,(x_i)}{\sum_{j=1}^{n} exp\,(x_j)} \tag{2.13}$$

Gradiente descendente. Técnica de minimización, que se usa también para el ajuste de los pesos de la red durante el proceso de retro-propagación. La función por minimizar se conoce como función objetivo o criterio y cuando se está minimizando, se le denomina también *cost function*, *loss function* o *error function* tal y como se indica en Goodfellow y col. (2016). Así pues, el problema consiste en minimizar una función objetivo que tiene la forma,

$$J(w)=\frac{1}{n}\sum_{i=1}^{n} J_i(w) \tag{2.14}$$

donde el parámetro w que minimiza J(w) debe estimarse. J_i se asocia con la i-ésima observación en el conjunto de datos utilizados para el entrenamiento (ajuste). El término estocástico proviene del hecho relativo a la selección de las muestras (observaciones) para el ajuste de forma aleatoria. $J_i(w)$ es el valor de la *loss function* en el i-ésimo ejemplo y J(w) es el riesgo empírico.

El gradiente descendente se usa para minimizar esta función de forma iterativa, con iteraciones t,

$$w(t+1)=w(t)-\varepsilon\frac{1}{n}\sum_{i=1}^{n}\nabla J_i(w) \tag{2.15}$$

De esta forma iterativa el método recorre el conjunto de entrenamiento y realiza la actualización anterior para cada muestra de entrenamiento. Se pueden realizar varios pasos sobre el conjunto de entrenamiento hasta que el algoritmo converja. Si se hace esto, los datos se pueden seleccionar aleatoriamente (criterio estocástico) en cada paso para evitar ciclos. Estas muestras así seleccionadas constituyen lo que se denomina *batch*. Las implementaciones típicas pueden usar una razón de aprendizaje adaptativa para que el algoritmo converja. En pseudocódigo, el método de gradiente descendente estocástico es como sigue,

1. Elegir un vector inicial de parámetros w (puede ser aleatoriamente) y razón de aprendizaje ε.
2. Repetir hasta que se consigue un mínimo aproximado
   2.1 Seleccionar aleatoriamente ejemplos en el conjunto de entrenamiento
   2.2 Para i = 1, 2, …, n, hacer

$$w(t+1)=w(t)-\varepsilon\nabla J_i(w)$$

Ejemplo: Supóngase que se quiere ajustar una línea recta y = w_1 + w_2x a partir de un conjunto de observaciones (x_1, x_2, …, x_n) con sus correspondientes respuestas estimadas $(\hat{y}_1, \hat{y}_2, ..., \hat{y}_n)$ utilizando mínimos cuadrados. La función objetivo a minimizar es,

$$J(w)=\sum_{i=1}^{n} J_i(w) = \sum_{i=1}^{n} \left(\hat{y}_i - y_i\right)^2 = \sum_{i=1}^{n} \left(w_1 + w_2 x_i - y_i\right)^2 \tag{2.16}$$

Uno de los métodos de optimización es el conocido como estimación del momento adaptativo (*adam, adaptive moment estimation*) descrito en Kingma y Ba (2014), que es justamente el que se utiliza en el modelo propuesto en el presente trabajo, definido como sigue,

$$\begin{aligned} m(t) &= \beta_1 m(t-1) + \left(1-\beta_1\right)\nabla J_i(w(t-1)) \\ v(t) &= \beta_2 v(t-1) + \left(1-\beta_2\right)\left(\nabla J_i(w(t-1))\right)^2 \\ w(t) &= w(t-1) - \frac{\alpha m(t-1)}{\sqrt{v(t-1)} + \varepsilon} \end{aligned} \tag{2.17}$$

Todos los procedimientos basados en RNC, consisten en extraer determinadas características de los espectrogramas mediante las operaciones definidas previamente, de forma que se obtienen los valores que definen los núcleos de convolución. Estas características pueden ser componentes de alta o baja frecuencia subyacentes en los espectrogramas. En las primeras capas se usan las funciones de convolución, rectificación lineal para destacar las características importantes y normalización para acelerar la convergencia. Para conectar cada neurona con su capa oculta se utiliza una matriz de pesos, que se modificará según la red neuronal vaya "aprendiendo", y un sesgo, a este par se le conoce como filtro con su correspondiente núcleo. Cada núcleo se genera en función de las características de la imagen a destacar. Seguidamente se usa una capa de *pooling* para crear una versión condensada de las características derivadas de la información recogida por la capa convolucional, cabe destacar que existirán tantas capas de convolución como de agrupación.

Tras varias etapas de convolución-agrupación se llega a las conexiones de tipo "*Fully Connected*" o totalmente conectadas, identificadas como fc, concretamente en el modelo de red propuesto se corresponde con la penúltima capa conectada a la salida. Por último, se aplica la función *softmax* definida previamente para obtener la probabilidad para cada una de las clases que queramos analizar. (Revisar matlab)

Una vez definido el modelo de red, se encuentra disponible para su entrenamiento. De suerte que en el caso del presente trabajo se establecen tres clases, que determinan los tres tipos

de sonido a identificar por lo que existirá como resultado de la red, después de la operación *softmax*, un vector cuyos valores corresponden a esas tres clases [y0, y1, y2]. En el caso del modelo presentado los pesos se inician con valores aleatorios en el rango [-1,1]. A modo de ejemplo, tomando como referencia la primera capa de convolución (conv1), donde se aplican 12 filtros, cada uno de dimensión 3x3, el primer filtro toma los siguientes valores iniciales durante uno de los entrenamientos realizados.

$$w(:,:,1) = 10^{-4}\begin{bmatrix} 84 & 108 & -122 \\ -23 & 21 & -114 \\ 75 & 18 & -143 \end{bmatrix}$$

El objetivo, una vez inicializados los pesos es ajustar (minimizar) la función de pérdida durante el entrenamiento, de forma que una salida deseada, por ejemplo [1,0,0], que correspondería a la plataforma se corresponda con la que se obtiene para una entrada dada. Este reajuste se realiza mediante retro-propagación del error aplicando la técnica del gradiente descendente.

La red debe reentrenarse con el mayor número de ejemplos (espectrogramas) disponibles para conseguir el mejor mínimo posible de la mencionada función de pérdida.

Capítulo 3 - Diseño de la aplicación

Una vez establecidas las bases teóricas que sustentan la aplicación, a continuación, se describe el modelo arquitectónico conceptual propuesto, indicando los módulos y recursos utilizados en cada caso. Se indican los flujos de datos entre módulos, así como los correspondientes interfaces. Se describe también la configuración de los módulos para el procesamiento de datos. Bajo esta consideración, hay que tener en cuenta que el despliegue en un entorno real queda condicionado a su verificación conceptual.

3.1 Arquitectura del sistema

El modelo de arquitectura que se plantea es el que se muestra en el esquema general de la figura 3-1, todo ello bajo cobertura internet. El dispositivo móvil permite capturar los datos, en este caso secuencias de sonido (audio clips), para almacenamiento y análisis de los mismos en la nube, a través de Matlab Drive (2020), de aquí en adelante identificado simplemente como Drive. Si bien en este caso, dadas las características del diseño y su orientación hacia una solución de concepto, los datos se encuentran previamente almacenados en el repositorio del propio Drive, de forma que cuando se conecta el móvil a él se realiza la pertinente sincronización, quedando disponibles tanto los datos almacenados como los programas para su procesamiento. La sincronización también se establece entre el Computador y Drive.

Como se puede apreciar en la mencionada figura, existen conexiones entre todos los elementos que componen la arquitectura bajo la requerida cobertura internet. También se proporciona un acceso específico a ThingSpeak (2020) como plataforma base en el ámbito de IoT proporcionada por MathWorks (2020). Las flechas de doble sentido determinan el flujo de datos entre los distintos componentes en ambas direcciones.

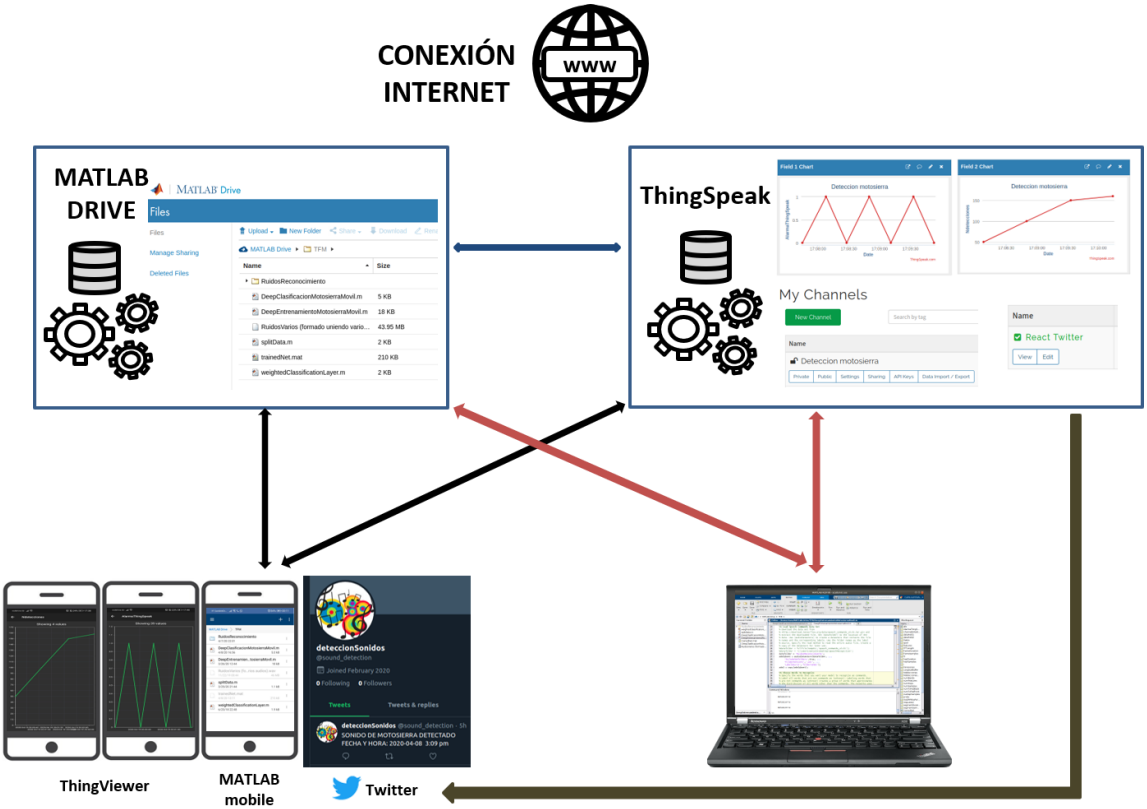


Figura 3-1 Arquitectura del sistema

Así, pues los elementos que componen la arquitectura del sistema son:

a) **Dispositivo Móvil:** elemento principal para la captura de datos y conexión vía internet con Drive y ThingSpeak. Recibe mensajes vía Twitter en el caso de producirse eventos sospechosos cuando se detecta un determinado tipo de ruido, en este caso de motosierra.

b) **Matlab Drive:** plataforma de almacenamiento y procesamiento de datos en la nube con los programas alojados en ella, y con las conexiones indicadas.

c) **ThingSpeak**: plataforma IoT en la nube para almacenar los datos relacionados a los sonidos reconocidos en Matlab hasta la ejecución de creación de tweets, tal y como se describe posteriormente en la descripción del flujo de datos.

d) **Computador**: plataforma para la realización de procesos con alta carga computacional, no soportados por el lanzamiento de procesos desde el móvil, ya que éste tiene limitado el tiempo de proceso a 240 segundos como máximo. Esta limitación conlleva implícito el hecho de que el entrenamiento de la red sólo es posible en el computador, y no en el dispositivo móvil, a pesar de que ambos tienen acceso al Drive sincronizado entre ambos.

Como se deduce del esquema de la figura 3-1, los accesos a Drive y ThingSpeak se pueden realizar tanto desde el dispositivo móvil como desde el computador, de forma que éstos pueden cargar datos en ellos, realizar procesos y solicitar la descarga de datos resultantes de dichos procesos. Además, conviene señalar que el dispositivo móvil accede a Drive a través de la correspondiente App (Matlab Mobile, 2020) disponible en Google Play y App Store según el dispositivo y su sistema operativo utilizado, que por otra parte, debe encontrarse instalada en dicho dispositivo. Como quiera Drive es un espacio asignado a una cuenta personal institucional de The Mathworks, con acceso restringido por usuario, los accesos tanto desde el móvil como desde el computador se realizan al mismo y único espacio, encontrándose sincronizado de forma que los dispositivos tengan accesos a los mismos datos y procedimientos. A este respecto conviene señal que aunque tanto Drive como ThingSpeak son ambas plataformas remotas, es ésta última la que se encuentra configurada según el paradigma IoT. Finalmente, indicar que la visualización por cualquier usuario de los canales (siempre que éstos sean públicos) y los datos allí alojados se realiza a través de la correspondiente página ThingSpeakWeb (2020) o la aplicación ThingView (2020) desde el móvil, conociendo el identificador de usuario o una etiqueta asociada al canal. La posibilidad de visualización de los datos en canales públicos permite que cualquier interesado, tanto a nivel particular como institucional, pueda conocer los eventos producidos por el sistema, facilitando la monitorización de los eventos correspondientes y su ubicación temporal.

El flujo y procesamiento de datos que se propone es el siguiente, cuyos detalles se describen en el Apéndice:

a) En Drive se almacena un audio sintético generado entremezclando secuencias de ruidos de motosierra y otros, tales como ruido de tráfico o en ciudad o de fondo en bosques para diferenciarlos de los primeros que constituyen el interés esencial, y que simulan la captura de datos directa por el computador. Además se almacenan convenientemente los datos para el entrenamiento de la RNC.
b) Desde el Computador se realiza el entrenamiento de la RNC mediante el acceso a Drive. Tras el entrenamiento, en Drive permanece almacenado el modelo de red con los pesos de las capas actualizados tras el entrenamiento.
c) Desde el dispositivo móvil se solicita a través de Drive, la clasificación del audio sintético antes mencionado para su análisis. Cuando se detecta un tipo de ruido, que se identifica como de motosierra, se incrementa un contador, reportando a ThingSpeak tal evento.
d) En ThingSpeak se almacena un indicador de la accion de deteccion y el número de detecciones de los sonidos de motosierra identificados. Tambien se levanta la alarma que

envia un aviso vía Twitter, el cual se recibe en todos los dispositivos que cuentan con acceso a la cuenta habilitada en Twitter.

3.2 Configuración para el procesamiento de datos

Una vez definida la arquitectura, los datos constituyen una parte esencial del sistema. A continuación, se describen los distintos procesos llevados a cabo sobe los mismos, desde su captura por el dispositivo correspondiente, en este caso el móvil, hasta su almacenamiento y tratamiento en la nube. Así, se considera en primer lugar la generación de audios tanto para el entrenamiento como para las fases de validación y test. Con los audios obtenidos, se generan los correspondientes espectrogramas para muestras de audio convenientemente estructuradas. Una vez caracterizados los datos en forma de espectrogramas, quedan disponibles como entrada a la red neuronal convolucional. Ésta se configura con la arquitectura mostrada en la figura 2-6, siendo necesario definir los parámetros involucrados durante el proceso de entrenamiento. Los resultados del procesamiento de los datos se reciben en la nube, en este caso ThingSpeak, siendo necesaria su configuración.

a) Dispositivo móvil para captura y preparación de los datos

Como se ha indicado previamente, en el presente trabajo se plantea una solución de concepto, lo que implica que no se capturan datos reales en tiempo real, siendo necesario acudir a bases de datos específicas. Para el caso de los sonidos de motosierra se han tomado audios correspondientes a diferentes modelos y fabricantes de motosierra.

Con los datos disponibles, y dado que es necesario entrenar la RNC para su entrenamiento, los datos, en un primer momento, se estructuran de forma que se establecen tres categorías, una para el entrenamiento propiamente dicho, otra para la validación del entrenamiento y una tercera para los test con datos extra.

b) Generación de espectrogramas

Las señales de audio se preparan de forma que se define la duración de cada secuencia en segundos (D), estableciéndose en un segundo (1s). Por otra parte, el clip de audio completo se divide en tramos, cuyo valor se identifica como duración del tramo, que según la figura 2-2 se corresponde con el ancho de la ventana (*L*) expresado también en segundos, y establecido en 0.025s, lo que da lugar a 40 tramos en el intervalo D. También, con referencia a la misma figura 2-2, se define el ancho de solapamiento entre ventanas (*a*) que en este caso se fija en 0.01s. La frecuencia de la señal de audio (*fs*), se mide en Hertzios

(s^{-1}), de forma que para el presente trabajo se establece en *fs* = $48*10^3$. Se utiliza la ventana de *Hanning* con el valor de *L* indicado previamente. Por otra parte, la longitud de la transformada de Fourier o número de puntos de la transformada considerada es de 1024, que es el parámetro M en la ecuación. En la siguiente tabla se recopilan los mismos valores indicados previamente.

Por otra parte, el número de muestras de audio para cada tramo aparecen en la tabla 3-1 donde aparece el tipo de parámetro, junto con su duración o extensión en tiempo y en número de muestras, resultantes de multiplicar el valor del parámetro (P) por la frecuencia (fs).

Parámetros (P)	Tiempo en segundos (s)	Número de muestras P*fs
D	1.0	48000
L	0.025	1200
a	0.010	480
L-a (solapamiento)	0.015	720

Tabla 3-1 Parámetros para generar los espectrogramas

Respecto de los datos, resulta habitual realizar un proceso que se conoce técnicamente como aumento de datos (*data augmentation*), que en el caso del reconocimiento de audios consiste en la generación audios de ruido aleatorio, como el aplicado en el presente trabajo, que se incorporan al conjunto de datos de entrenamiento. Para que no sea excesivamente invasivo y domine el proceso de aprendizaje, también suele ser común añadir un número de audios que represente un cierto porcentaje con respecto a las muestras de entrenamiento disponibles, fijado en este caso en un 10% del total de audios disponibles.

c) Modelo de la RNC y definición de parámetros de entrenamiento

Como se ha indicado previamente, el modelo de red utilizado en el presente trabajo es el de las 24 capas definidas en la figura 2-6. Se han diseñado otros modelos añadiendo capas adicionales, concretamente añadiendo un módulo cada vez comprendiendo las 4 capas del tipo convX, normX, reluX y maxpoolX, donde X indica el lugar de inserción, de forma que si X = 2, estas 4 capas se insertan entre los actuales módulos con X = 1 y X = 2, desplazando el resto de los módulos. Las mencionadas inserciones se han realizado tanto entre las capas maxpool1 y conv2, así como entre las capas maxpool2 y conv3. En cualquiera de los dos casos los resultados, en cuanto al desempeño final, no han mejorado los del modelo propuesto, antes bien, se han incrementado los tiempos de entrenamiento y

ligeramente los de clasificación, por lo que se ha optado finalmente por el modelo propuesto en la mencionada figura 2-6.

Por otra parte, y desde el punto de vista del proceso de entrenamiento, es necesario establecer los parámetros que se indican a continuación, cuyos valores se describen en el capítulo siguiente:

a) ***Epochs* e iteraciones**: define los pasos para procesar el conjunto de muestras de entrenamiento. En cada paso se puede seleccionar un determinado subconjunto del total de muestras disponibles, si bien en todos los casos del mismo tamaño. Para cada *epoch* se define un determinado número de iteraciones.
b) **Batch *size***: define la cantidad de audios que se utilizan en cada iteración.
c) **Razón de aprendizaje (*learning rate*)**: determina la rapidez con la que la red aprende según el método de actualización de los parámetros que constituyen el objetivo del aprendizaje.
d) **Método de optimización**: define el método utilizado para realizar la minimización de la función de coste.

Además, durante el proceso de entrenamiento es necesario definir tres conjuntos de audios, a saber: a) *Entrenamiento,* b) *Validación* y c) *Test*.

El conjunto de Entrenamiento sirve para alimentar la red, y el conjunto de Validación en realidad puede considerarse como parte del conjunto de entrenamiento, ya que se utiliza para ajustar los hiperparámetros de las capas que definen el modelo. Por lo general, este último se usa para verificar la correcta evolución del proceso de entrenamiento con los parámetros establecidos y para evitar el sobreajuste de los pesos. Así, por ejemplo, si la red está entrenada solamente con un conjunto de entrenamiento, es muy probable que obtenga un 100% de precisión y con un excesivo sobreajuste, obteniendo un rendimiento muy pobre en el conjunto de prueba.

Para medir el comportamiento del modelo de red ajustado se utiliza lo que se conoce como precisión en la validación (*validation accuracy*). Por consiguiente, el conjunto de validación, que es independiente del conjunto de entrenamiento, se utiliza para el ajuste de parámetros y medir la precisión. Por otro lado, el conjunto de Test sirve para determinar el rendimiento del modelo, una vez que el entrenamiento ha finalizado.

Una vez completado el entrenamiento, tanto el modelo de red como los pesos de las capas ajustados (aprendidos) se almacenan para su uso en la fase de identificación o clasificación de los sonidos

d) Configuración de ThingSpeak

Previamente creada la cuenta correspondiente de ThingSpeak, para almacenar los datos relacionados a las detecciones de sonidos se crea el canal de nombre "Detección motosierra" con 2 campos, Field1 con nombre AlarmaThingSpeak el cual almacena el número "1" si Drive detecta el sonido de la motosierra o el número "0" de caso contrario. Además, Field2 lleva el nombre Ndetecciones el cual almacena la cantidad de veces que se detectó el sonido de la motosierra, tal y como aparece en la figura 3-2.

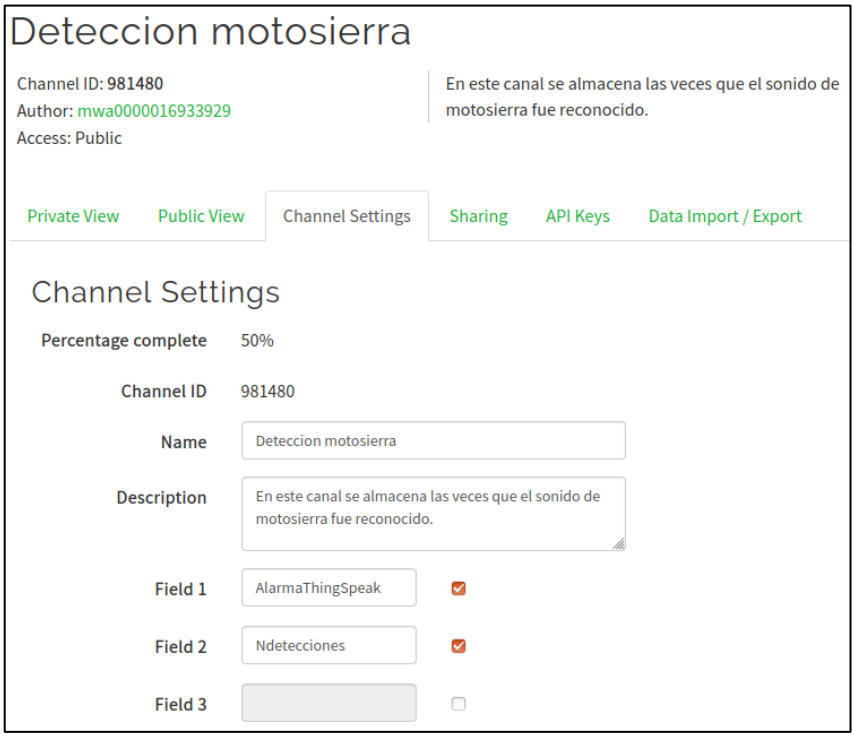


Figura 3-2 Configuración de canal en ThingSpeak

Se hace uso de la opción ThingTweet para enlazar la cuenta de Twitter @sound_detection, creada para la publicación de avisos cuando se detecta una alta posibilidad de tala ilegal por la persistencia en la detección de sonidos de motosierra. Como se observa en la Figura 3-3, es necesario haber iniciado la sesión en la cuenta Twitter además de autorizar en ThingTweet la generación de tweets, en dicha cuenta, activando la opción "Link Twitter Account".

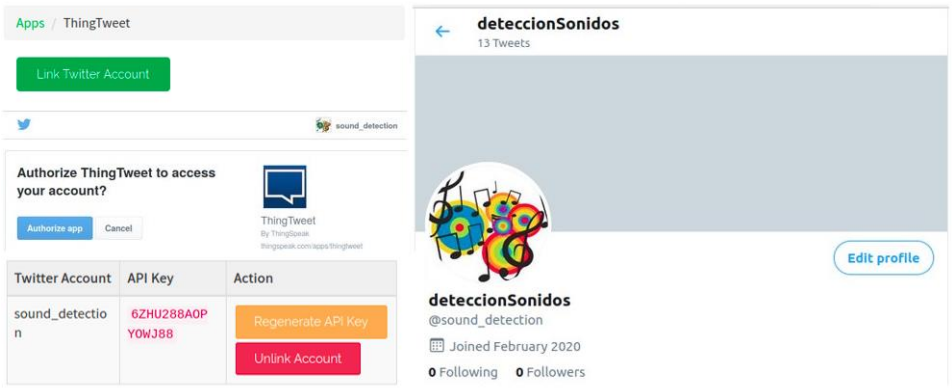


Figura 3-3 Configuración de ThingTweet

En ThingSpeak se crea una aplicación de tipo "React", la cual realiza una acción especificada cuando se cumple una determinada condición. Tal como se puede observar en la Figura 3-4, la aplicación React para este módulo con nombre "React Twitter", se encarga de verificar cada inserción en el campo Field1: AlarmaThingSpeak. De darse el caso que el valor de inserción contiene el digito "1" se genera el tweet con el siguiente contenido: "SONIDO DE MOTOSIERRA DETECTADO FECHA Y HORA: %%datetime%%", donde datetime representa la fecha y hora en la que se detectó dicha inserción.

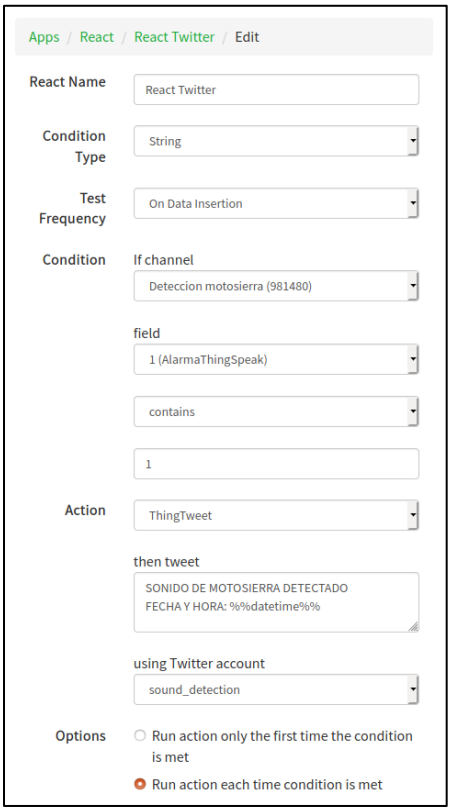


Figura 3-4 Configuración de React

Capítulo 4 - Análisis de resultados

Para el análisis de los resultados se describen, en primer lugar, los recursos utilizados, donde se describen tanto la naturaleza de los datos como las herramientas. En segundo lugar, se analiza la generación de los espectrogramas a partir de los audios, que constituyen las entradas a la RNC tanto en la fase de entrenamiento como en la de identificación, donde se evalúan los resultados. Finalmente, se muestran los resultados relativos a las operaciones realizadas en ThingSpeak y los mensajes de Twitter.

4.1 Recursos y herramientas

Para el presente trabajo, el conjunto de siete audios utilizados para los experimentos se ha obtenido del repositorio Free Sound Effects (2020), que se encuentran disponibles para uso no comercial en el formato de compresión MP3. Las características de los mismos se reflejan en la tabla 4-1. En la primera columna se indica el nombre del fichero del audio. En la segunda columna aparece el nombre con el que se identifica en el repositorio. En la tercera columna el tamaño del fichero en Mb. En la cuarta columna aparece la duración en segundos del audio correspondiente. La quinta columna aparece la frecuencia de muestreo de la señal, expresada en hertzios (Hz), esto es, el número de muestras tomadas por segundo. Finalmente, en la sexta columna aparece el número de muestras total de la señal, que se obtiene multiplicando la frecuencia de muestreo por la duración de la misma.

Fichero de Audio	Nombre del audio	Tamaño en Mb	Duración en segundos	Frecuencia de muestreo de la señal (Hz)	Número de muestras
chainsaw-01.mp3	Chainsaw 1	1.5	37.7280	48000	1810944
chainsaw-02.mp3	Chainsaw 2	2.4	60.7680	48000	2916864
chainsaw-03.mp3	Chainsaw 3	1.3	33.0480	48000	1586304
chainsaw-04.mp3	Chainsaw 4	1.4	33.9360	48000	1628928
chainsaw-05.mp3	Chainsaw 5	0.681	16.9920	48000	815616
chainsaw-06.mp3	Chainsaw 6	1.4	35.4960	48000	1703808
chainsaw-07.mp3	Chainsaw 7	1.6	38.8800	48000	1866240

Tabla 4-1 Características de los datos de audio

A partir de los ficheros anteriores, para generar las muestras de entrenamiento, validación y test, se toma cada uno de los ficheros en la tabla 4-1 de los cuales se extraen audio clips conteniendo 16000 muestras, generando así 770 nuevos audios con nombre motosierra_x.wav,

donde x es un número de la secuencia del nuevo fichero generado. Los audios destinados para las categorías de validación y prueba se especifican en archivos de texto plano con nombres: validation_list.txt con 77 audios y testing_list.txt con 79 audios, los destinados para el entrenamiento serán los no incluidos en los archivos de texto. Conviene reseñar en este momento, que aunque el número de audios utilizados para el entrenamiento es relativamente pequeño comparado con el número de imágenes que utiliza el modelo AlexNet en su versión pre-entrenada, en el presente trabajo éste es suficiente para verificar y comprobar el comportamiento del modelo de red propuesto bajo la perspectiva de la solución de concepto que se plantea bajo el paradigma IoT.

Para el procesamiento y tratamientos de los datos se emplea la herramienta Matlab (MathWorks, 2020) con los toolboxes de *Deep Learning*, *Signal Processing*, *Audio* y *ThingSpeak support*. Como herramientas para acceso a los servidores y procesamiento de datos en la nube se utilizan MathDrive (2020) y la pataforma ThingSpeak (2020). El computador empleado para el procesamiento corresponde a un procesador Intel Core i7 3.0 GHz (9th *generation*) con 16GB de RAM y Ubuntu 18 como sistema operativo de 64-bits.

El código, junto con los datos de procesamiento y configuración, así como el repositorio donde se ubican, se describen en el Apéndice, junto con las instrucciones necesarias para su ejecución

4.2 Generación de espectrogramas

A cada espectrograma se le asigna una de las tres etiquetas "motosierra", "fondo" o "desconocido" las cuáles serán los indicadores de la clasificación que posteriormente se efectúe a partir del modelo entrenado. En definitiva, representan las tres categorías en las que se clasifican los audios, y por tanto las tres salidas que proporciona la Red, construyendo el objetivo del entrenamiento de la RNC. Para preparar los datos para el entrenamiento se generan, mediante el proceso descrito en la sección 3.2, los espectrogramas correspondientes para las categorías mencionadas. Se puede observar en la figura 4-1 el espectrograma 20 del conjunto de entrenamiento identificado como XTrain y del conjunto de validación (XValidation), donde las dimensiones horizontales representan las 40 bandas de frecuencia y la dimensión vertical el tiempo en segundos.

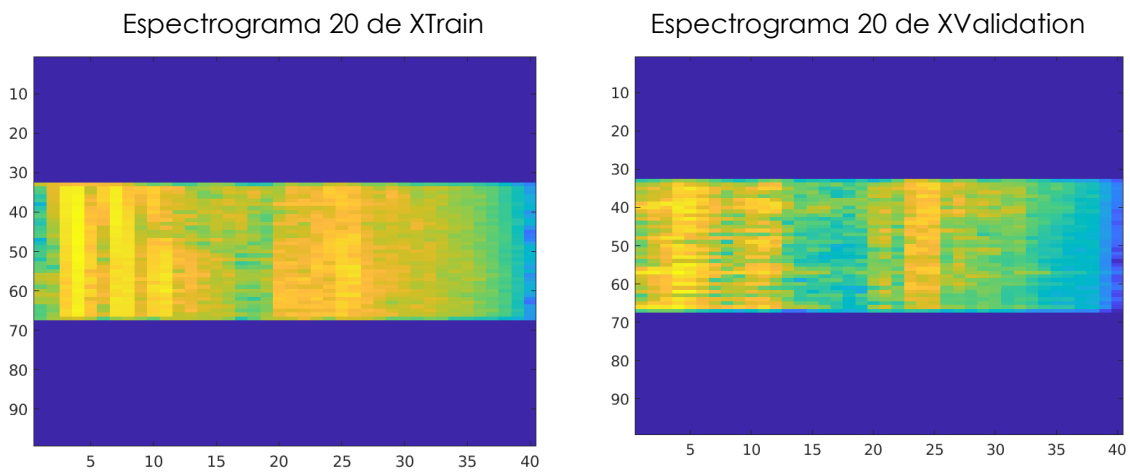


Figura 4-1 Distribución de muestras de entrenamiento y validación

La red es capaz, no solo de reconocer los sonidos provenientes de la motosierra, sino también de detectar si la entrada contiene silencio o ruido de fondo. Se usan diferentes audios para crear muestras de audio clips, con un segundo de duración, de ruido de fondo, que se generan aleatoriamente con valores en el rango (0,1). La figura 4-2 muestra la distribución del número de espectrogramas para las categorías de entrenamiento y validación. También se muestran los involucrados para la clasificación de sonidos desconocidos y los usados para agregar sonidos de fondo. Como puede observarse en ambos casos, el número de muestras en el caso de la motosierra es superior a los de las otras dos, en proporciones aproximadas de 2/3 y 1/3 de desconocidos y fondo frente a motosierra para el entrenamiento y de 3/5 y 3/8 en el caso validación. En consecuencia, el número de muestras correspondientes a la categoría de motosierra es superior a las otras dos. Conviene reseñar que sobre las muestras involucradas en los procesos de aprendizaje (entrenamiento, validación y test) es habitual aplicar técnicas conocidas como validación cruzada (k-fold cross-validation) (Duda y col., 2001), de forma que del total de audios disponibles en cada iteración, se toman k para validación y el resto para entrenamiento. El proceso se repite k veces, seleccionando en cada iteración un conjunto de validación diferente de forma que el resto, también diferente, se utiliza para entrenamiento. Tras las k iteraciones se calcula el error promedio con los diferentes modelos entrenados, que determina la precisión y el error final. Aunque en el caso de las RNC este proceso es también aplicable, no obstante, en el presente trabajo, no se ha considerado, ya que los lotes que componen las diferentes epochs se procesan intercambiando las muestras en cada iteración, compensando en parte el efecto de validación cruzada, además de que la mejora de la RNC no se establece como un objetivo preferente, dado que el trabajo busca principlamente verificar la validez de una solución integrada IoT.

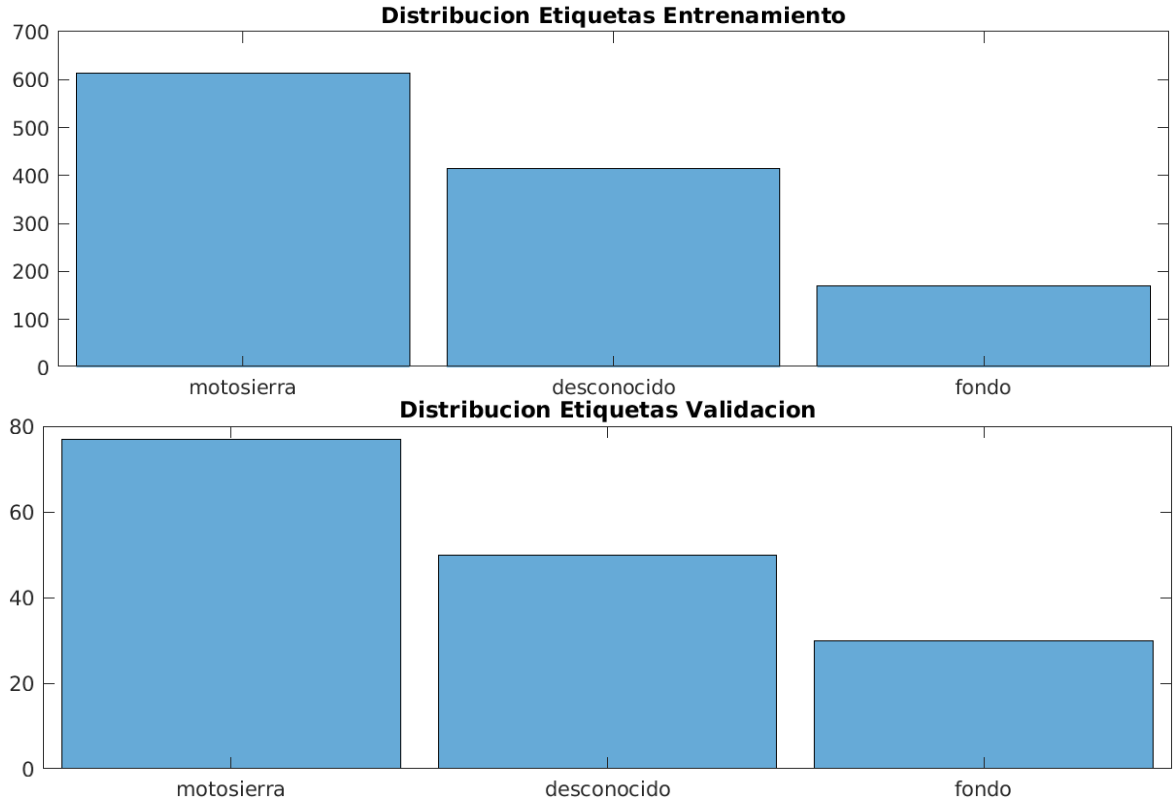


Figura 4-2 Distribución de muestras de entrenamiento y validación

4.3 Entrenamiento de la RNC

Con los espectrogramas previamente generados se lleva a cabo el entrenamiento teniendo en cuenta el modelo de la RNC visto en la figura 2-6. La figura 4-3 muestra una etapa intermedia del proceso de entrenamiento cuando se han realizado 11 iteraciones de 225, esto es un 8,4% del total. El proceso completo consta de 25 *epochs* con 9 iteraciones cada una, hasta completar el total de 225 iteraciones, que se corresponde al máximo. A medida que se progresa se muestran los resultados correspondientes a la precisión y al ajuste relativo a la función de coste o *loss function*. No se ha fijado ningún límite del tipo *patience*, que está fijado a infinito. Cuando se fija este parámetro a un valor determinado, lo que realmente se indica es que el proceso se detenga después de que tras el número especificado de *epochs* no se consigan mejoras significativas o relevantes en la precisión.

En la figura 4-4 se puede observar aun cuando se han finalizado todos los epochs no se alcanza totalmente una precisión del 100%, por lo que sería necesario fijar más *epochs* para este

conjunto de datos (Entrenamiento y Validación), si bien se ha preferido mantener los valores indicados para mostrar este efecto.

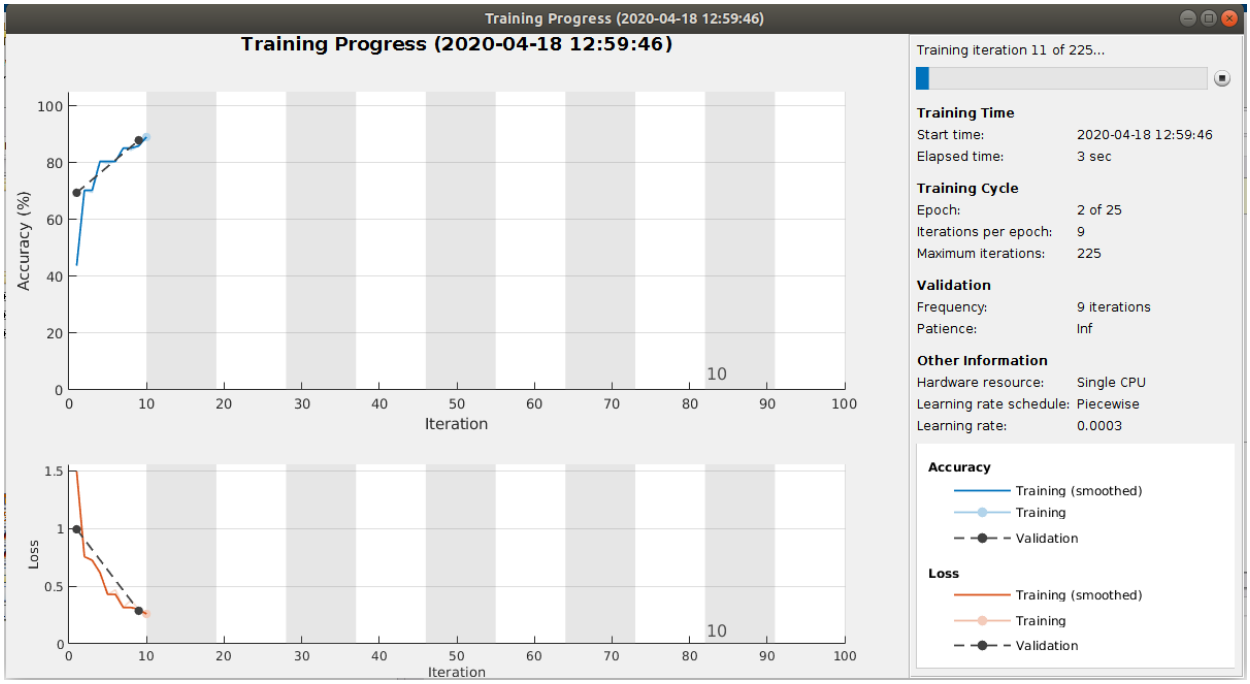


Figura 4-3 Progreso del proceso de entrenamiento de la RNC

El proceso de entrenamiento se completa según se indica en la figura 4-4, observándose una precisión en cuanto a validación de las muestras de aproximadamente el 98%, que se ha mantenido constante a lo largo del proceso desde que se estabilizó en la primera *epoch* lo que permite concluir que el modelo utilizado resulta ciertamente eficiente.

Con relación a esta misma figura se observa que el tiempo total empleado en el proceso ha sido de 53 segundos. Se trata de un tiempo razonable bajo las consideraciones especificadas. Si bien, dada la evolución del proceso, si en lugar de 25 *epochs* se utilizan 40, se consiguen los resultados mostrados en la figura 4-5, alcanzando una precisión del 98.73% en un minuto y 24 segundos, que es un porcentaje apropiado en un tiempo no tan superior al caso anterior.

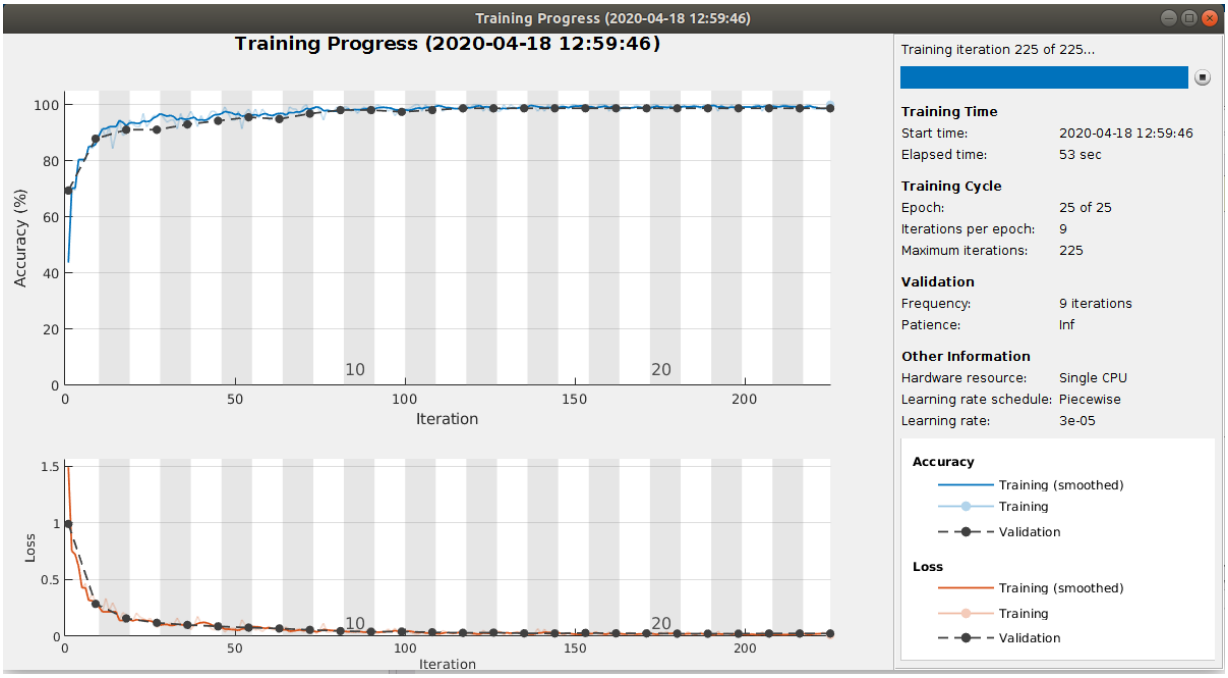


Figura 4-4 Finalización del proceso de entrenamiento de la RNC con 25 epochs

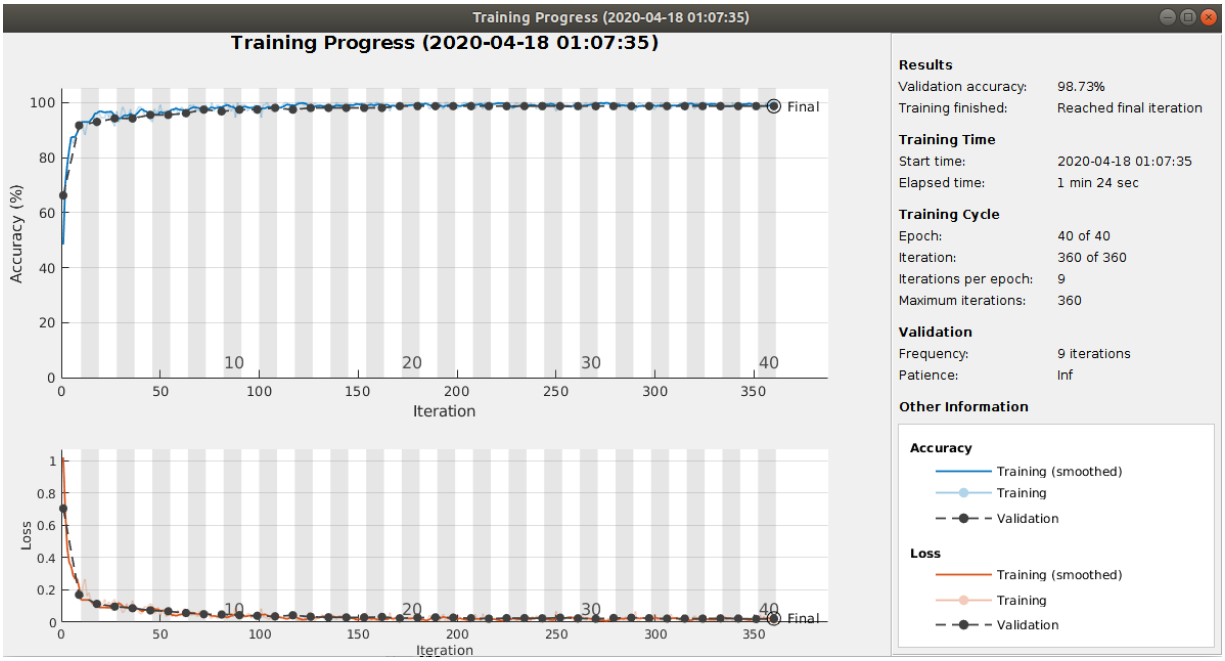


Figura 4-5 Finalización del proceso de entrenamiento de la RNC con 40 epochs

Una vez realizado el entrenamiento, los pesos en las distintas capas han sido convenientemente ajustados. A modo de ejemplo, considerando la primera capa de convolución (*conv1*), de acuerdo con el esquema de la figura 2-6, en la primera capa de convolución se aplican 12 filtros (K = 12), todos ellos de dimensión 3x3, obteniéndose el correspondiente mapa de características, de dimensión 98x40x12, es decir, un conjunto de 12

matrices de salida de dimensión 98x40. Los valores de los filtros 1 y 12 en la mencionada capa conv1 son los siguientes:

$$w_1 = \begin{bmatrix} -0.0044 & 0.0056 & -0.0051 \\ 0.0144 & 0.0154 & -0.0032 \\ -0.0113 & -0.0158 & 0.0029 \end{bmatrix} \qquad w_{12} = \begin{bmatrix} 0.0015 & 0.0054 & -0.0109 \\ 0.0067 & 0.0033 & -0.0047 \\ -0.0038 & -0.0089 & 0.0188 \end{bmatrix}$$

En la figura 4-6(a) se muestra un espectrograma, sobre el que se aplican los dos filtros anteriores, dando como resultado los espectrogramas filtrados que se muestran en (b) y (c) respectivamente, en ambos casos con las dimensiones especificadas de 98x40.

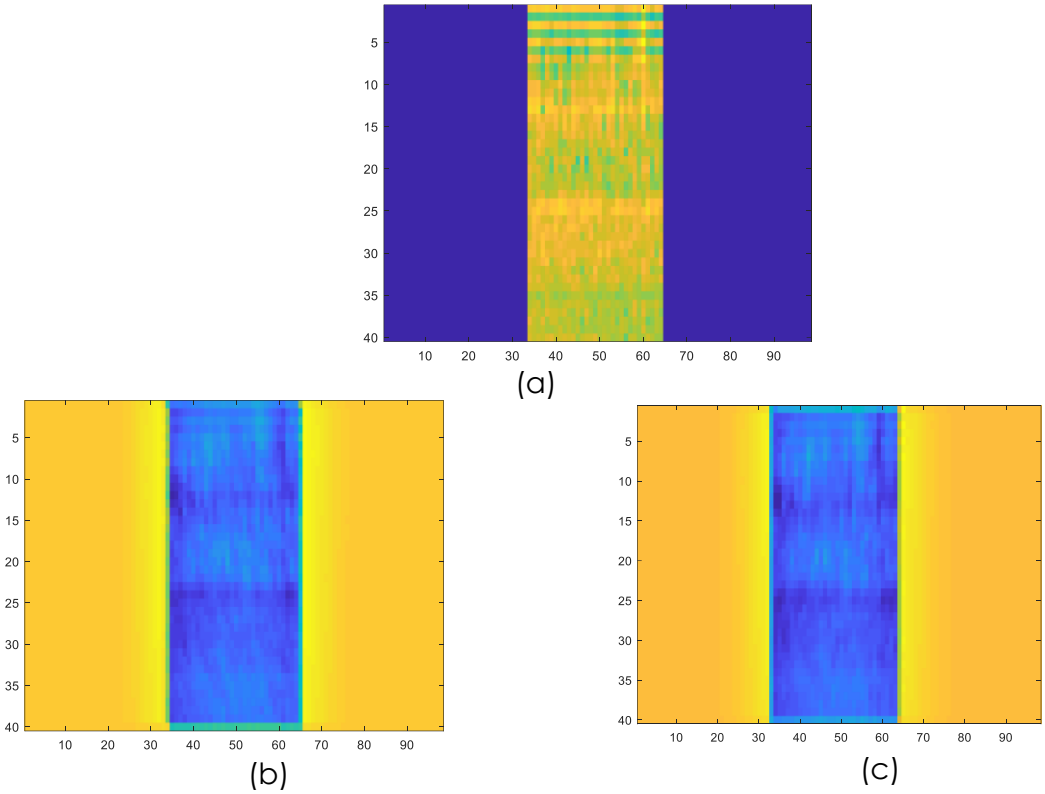


Figura 4-6 Capa de convolución (conv1): (a) Espectrograma; (b) Resultado de aplicar el filtro 1; (c) Resultado de aplicar el filtro 12.

Para ilustrar de nuevo el comportamiento de los filtros, en la figura 4-7 se muestra el resultado obtenido tras el paso del espectrograma por la primera capa *Maxpool* (*maxpool1*), tal y como aparece en la mencionada figura 3.6, en sus canales 1 y 12, teniendo en cuenta, en este caso, que el tamaño del mapa de características es de dimensión 49x20x12. Concretamente, en (a) se muestra el resultado en el canal 1 y en (b) en el canal 12.

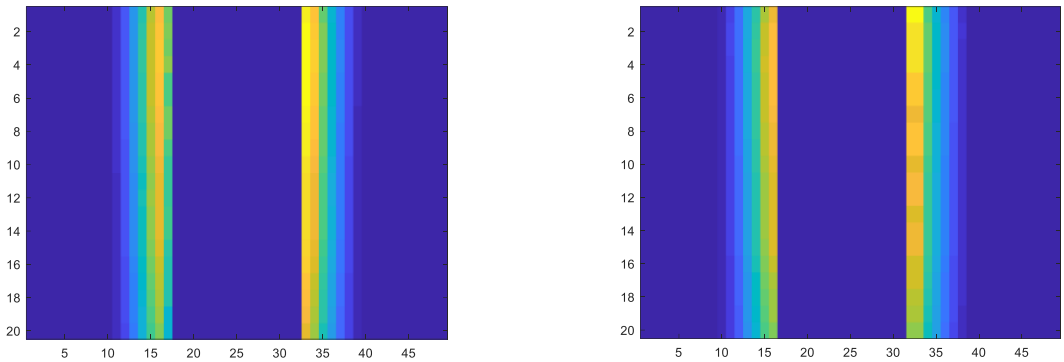


Figura 4-7 Capa de Maxpool (maxpool1): (a) Resultado obtenido en el canal 12

Finalmente, en la figura 4-8 se muestra la matriz de confusión obtenida tras el proceso de entrenamiento, donde las filas de las dos columnas hacia la derecha muestran los porcentajes de observaciones clasificadas correcta e incorrectamente para cada clase predicha y las columnas de las últimas dos filas inferiores muestran los porcentajes de observaciones clasificadas correcta e incorrectamente para cada clase verdadera. A partir de los datos mostrados en la figura 4-8, se observan los porcentajes de las observaciones, donde los datos clasificados correctamente alcanzan porcentajes más que aceptables en relación al tipo de datos utilizados y teniendo en cuenta que algunos se han generado de forma sintética. En efecto, los porcentajes de observaciones clasificados correctamente llegan al 94.8% para motosierra y al 100% para las otras dos clases. Con relación a la predicción de las clases los porcentajes son del 100% para motosierra y del 93.8% y 96.2% para las otras dos clases, fondo y desconocido respectivamente, en todos los casos son también suficientemente aceptables en relación con los datos utilizados.

Figura 4-8 Matriz de confusión obtenida durante el proceso de entrenamiento

4.4 Identificación de espectrogramas mediante RNC

Para la prueba de clasificación del modelo se hace uso del audio "RuidosVarios (formado uniendo varios audios)" que se encuentra almacenado en Drive, la clasificación se puede ejecutar desde un dispositivo móvil o desde el computador. El audio contenido en el mencionado fichero es de tipo wav, con una duración de 480 segundos, con una frecuencia de muestreo de 48000 Hz, exactamente como las muestras utilizadas para el entrenamiento. Por tanto, el número de muestras totales es de 23040000. Se extraen tramos del audio y se generan los espectrogramas respectivos. A partir del modelo previamente entrenado se clasifica cada espectrograma en "desconocido" o "motosierra", como los mostrados en la figura 4-9. Donde cada espectrograma clasificado como "motosierra" aumentará el conteo de detecciones. Los espectrogramas en este caso se generan seleccionando muestras con 16000 valores, que se extraen de la señal de audio mencionada, lo cual significa que se utilizan en total de 1440 espectrogramas (23040000/16000). El fichero de audio completo se genera con 1440 clips, cada uno de ellos con 16000 muestras. Dichos audios son distintos a los utilizados durante el entrenamiento. El 65% corresponden a sonidos de motosierra y el resto a desconocido. Posteriormente, se entremezclan aleatoriamente para generar el fichero de audio completo "RuidosVarios" encadenándose uno tras otro. Esto significa que no se realiza ningún proceso de mezcla de sonidos ni se aplican variaciones de la frecuencia o se introduce ruido para generar distorsiones en los audios que lo generan. La distorsión de las señales sería algo normal en los procesos donde el aprendizaje constituye el objetivo principal. Si bien en este caso, al tratarse de una solución conceptual en IoT, el aprendizaje forma parte de la solución integrada, sin poseer un carácter finalista, razón por la que no se realizan las mezclas de sonidos con fines de distorsión, que en cualquier caso, habría que tener en cuenta el tipo y magnitud de las señales distorsionadas."

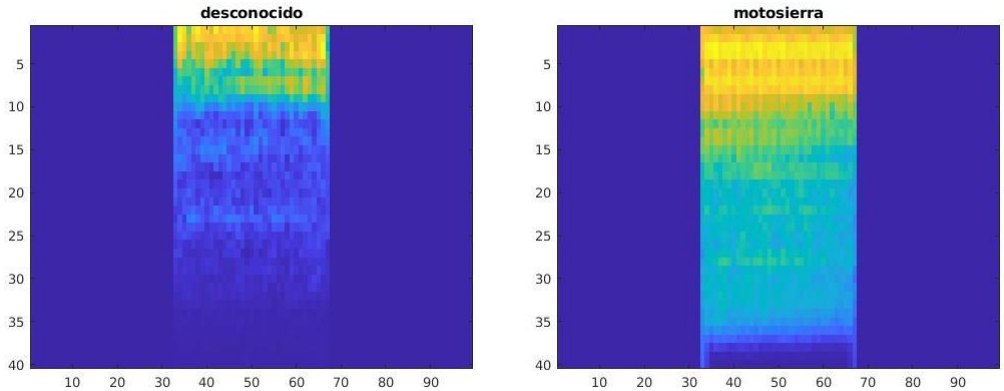


Figura 4-9 Espectrogramas usados para la clasificación

Con los 1440 audios disponibles, extraídos según se describe en la sección 4.1, se ha determinado, con los datos de validación, que el sistema es capaz de reconocer el 94.8% como motosierra, lo que representa un porcentaje aceptable desde el punto de vista de efectividad del sistema desarrollado. El procedimiento para determinar si un audio se considera como motosierra o no, que es lo que realmente interesa en este trabajo, es un experto humano que escucha el audio y determina su identificación o no como motosierra, para ello se tiene en cuenta que si lo que se escucha tiene una suficiente duración identificable como sonido de motosierra, éste se considera como tal y en aquellas otras muestras en las que el experto no es capaz de identificarlo como motosierra, bien por su escaso contenido de muestras de motosierra o por un exceso de ruido o por cualquier otra causa, éste no se clasifica como de motosierra. Esta es la razón por la que a la hora de generar el fichero de audio completo no es necesario llevar a cabo ningún etiquetado de los audios que generan dicho fichero.

Como colofón al análisis de espectrogramas conviene señalar que la RNC está sólo entrenada para el reconocimiento de tres tipos de sonidos (motosierra, desconocido y fondo). Esto significa que cualquier sonido que se proporcione al sistema se va a clasificar en una de las tres categorías indicadas, proporcionando la correspondiente probabilidad de clase a través de la capa o función *softmax* de forma que la suma total de probabilidades sea la unidad. Esto no significa que el reconocimiento del sonido sea el correcto. En efecto, en el contexto donde se pretende implantar el sistema IoT que se presenta en este trabajo, esto es la detección de talas ilegales en bosques con motosierras, puede convenir identificar también el sonido de vehículos a motor, tales como motocicletas o coches todo terreno, ya que cabe la posibilidad de que los autores de esas actividades ilegales se desplacen en ellos, de forma que su detección previa o temprana evitaría incluso la necesidad de la detección de la motosierra. Para ello, es necesario suministrar a la red nuevas muestras de entrenamiento con esas categorías, y naturalmente llevar a cabo el correspondiente re-entrenamiento, teniendo en cuenta que en este caso, a las tres categorías iniciales se añadirían estas dos nuevas, haciendo un total de cinco.

Al efectuar un re-entrenamiento se hizo uso de audios correspondientes a sonidos de motores de carros y motocicletas obtenidas del repositorio FreeSound (2020), las características de los mismos se reflejan en la tabla 4-2.

Fichero de Audio	Nombre del audio	Tamaño en Mb	Duración en segundos	Frecuencia de muestreo de la señal (Hz)	Número de muestras
carro-01.wav	61281__robinhood76__00441-fast-car-1	1.3	7.5094	44100	331166
carro-02.wav	160442_henrique85__start-engine	3.5	20.0726	44100	885202

carro-03.wav	345925__1histori__car-engine	9.5	24.6246	48000	1181981
carro-04.wav	376788__juanmower__carshow-cars-accelerate-crowd-002	32.7	111.7200	48000	5362560
carro-05.wav	392986__gaulish27__01-motor-2-170512-1016	5.2	39.0110	44100	1720384
motocicleta-01.flac	192661__ondrosik__motorcycles	7.5	9.4293	48000	452608
motocicleta-02.wav	236986_soundskeep__motorcycles	51.7	85.4571	44100	3768657
motocicleta-03.flac	264773__cdrk__motorcycle-passing-by	1.8	176.6250	48000	8478000
motocicleta-04.wav	462324_davidjuradoh_motorcycle-01	4.4	16.4054	44100	723476

Tabla 4-2 Características de los datos en los nuevos audios

A partir de dichos ficheros se vuelven a generar las muestras de entrenamiento, validación y test, se emplea el proceso previamente mencionado obteniendo los espectrogramas correspondientes a las nuevas categorías, carro y motocicleta, dos de ellos se muestra en la figura 4-10.

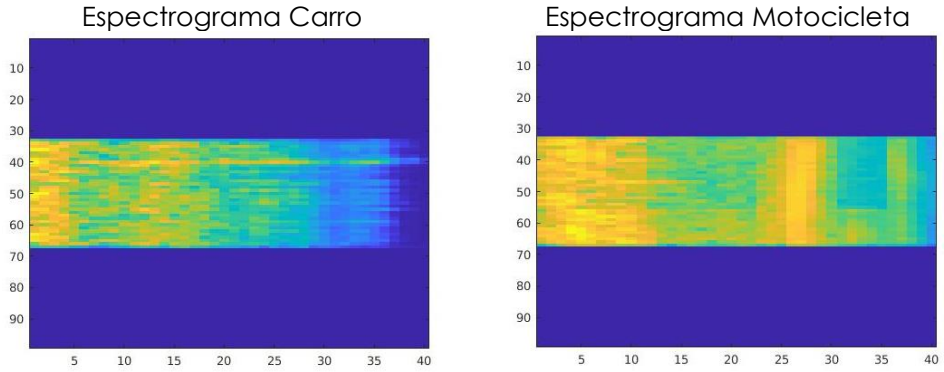


Figura 4-10 Espectrogramas de las nuevas categorías

Se obtiene la nueva distribución de las etiquetas para el entrenamiento y validación, los cuales incluyen las nuevas categorías generando así 529 nuevos audios de carros y 335 de motocicletas. Los audios destinados para las categorías de validación y prueba se actualizan en los ficheros previamente indicados: validation_list.txt y testing_list.txt. La nueva distribución que se muestra en la figura 4-11, corresponde a los 34 audios usados para validación, 36 para test y el resto para entrenamiento para la categoría carro; en el caso de motocicletas son 50 para validación, 52 para test y el resto para entrenamiento. Por las mismas razones indicadas previamente, no se aplica validación cruzada.

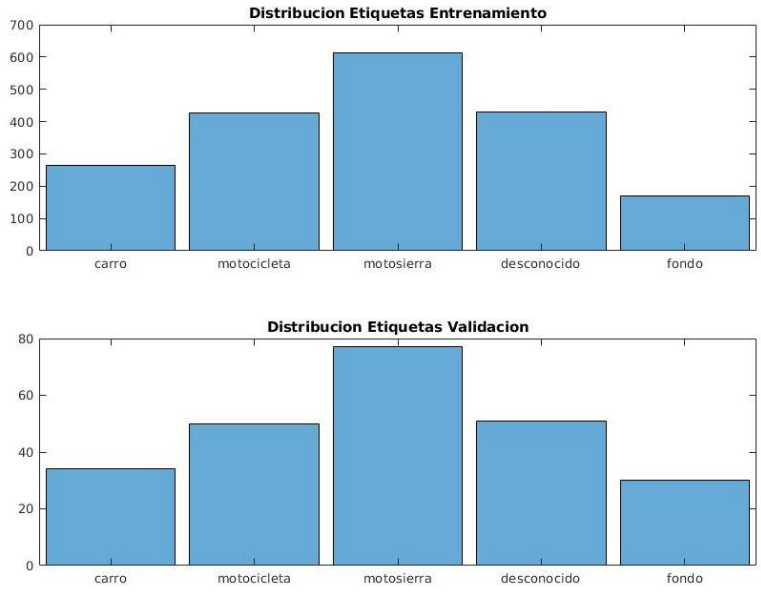


Figura 4-11 Distribución de muestras de entrenamiento y validación

Se procedió a re-entrenar a la red y para este proceso se completa según se indica en la figura 4-12, observándose una precisión en cuanto a validación de las muestras del 97.55% que permite concluir que el modelo utilizado continua resultando ciertamente eficiente para las nuevas categorías agregadas.

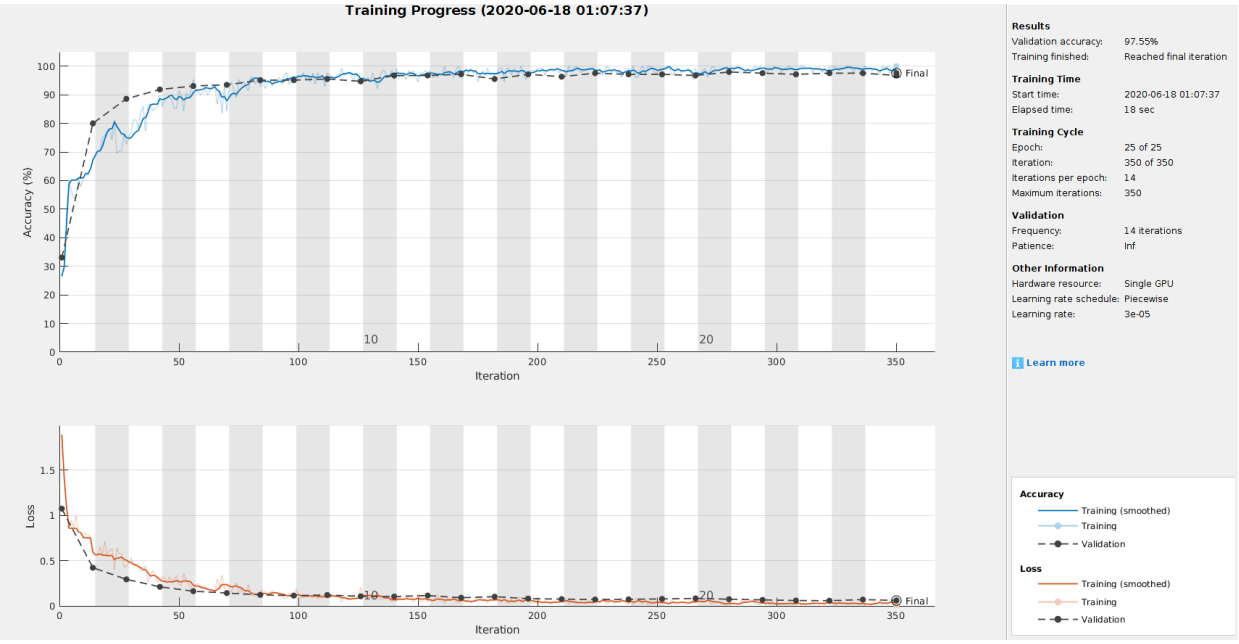


Figura 4-12 Finalización del proceso de entrenamiento de la RNC para 5 categorías

En la figura 4-13 se muestra la matriz de confusión obtenida tras el proceso de re-entrenamiento y a partir de los datos mostrados se determinan los porcentajes de las

observaciones, donde los datos clasificados correctamente continúan alcanzando porcentajes más que aceptables en relación al tipo de datos utilizados. En efecto, los porcentajes de observaciones clasificados correctamente llegan al 97.4% para motosierra, 91.2% para carros y 98% para motocicletas. Con relación a la predicción de las clases los porcentajes son del 98.7% para motosierras, 100% para carros y 94.2% para motocicletas.

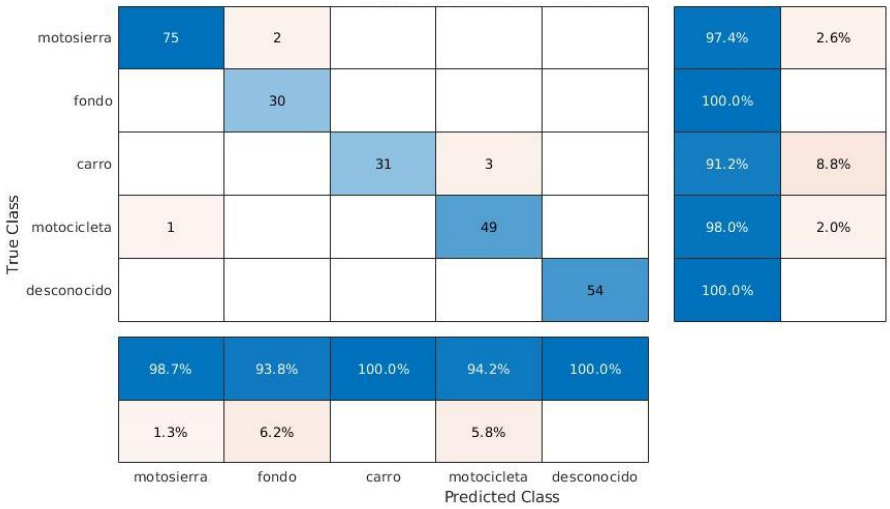


Figura 4-13 Matriz de confusión obtenida durante el proceso de re-entrenamiento

Con estas nuevas categorías, el modelo es capaz de diferenciar el sonido de los motores de carros y motocicletas, no clasificándolos como motosierras.

4.5 ThingSpeak y Twitter

Mientras se efectúa la clasificación para el audio propuesto, cada 50 detecciones del sonido de motosierra se realiza el envío del valor 1 al campo de AlarmaThingSpeak en la plataforma ThingSpeak y a su vez el número de detecciones en la que se obtuvo. En la figura 4-14 se muestran las veces en que la alarma toma el valor de 1 así como las detecciones almacenadas.

Como se ha mencionado previamente, la visualización de los datos en canales públicos, como es el caso, se puede realizar bien a través de un acceso web directo o mediante la App ThingView, en ambos casos mediante la identificación de usuario propietario de los canales públicos.

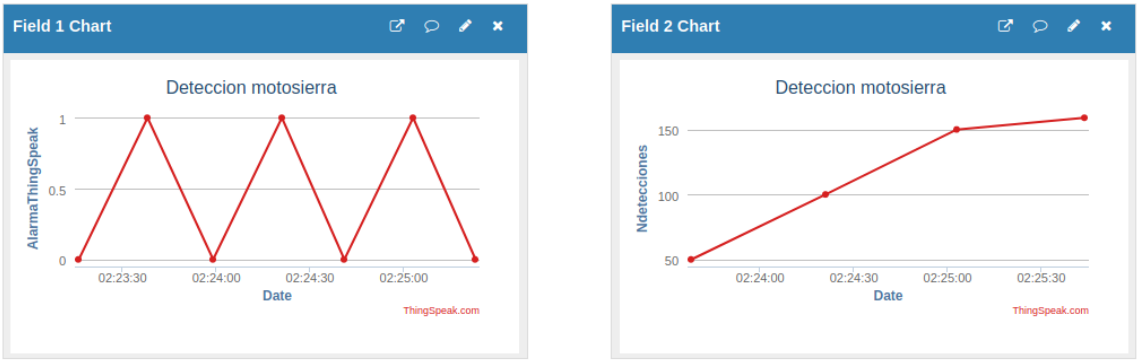


Figura 4-14 Representación de los datos en los campos del canal

Debido al React previamente configurado, en la figura 4-15 se evidencia su última ejecución al haber recibido el valor de 1 en el field AlarmaThingSpeak.

Figura 4-15 Última ejecución del React Twitter

Finalmente, la cuenta de twitter @sound_detection muestra la generación de los 3 tweets generados por control React debido a las inserciones del valor 1 para la alarma de detección de motosierra. Como se definió en la sección de descripción de ThingSpeak, el mensaje que se muestra es "SONIDO DE MOTOSIERRA DETECTADO" acompañado de la fecha y hora en que sucedió el evento, tal y como se aparece en la figura 4-16.

Figura 4-16 Perfil de la cuenta twitter @sound_detection

Capítulo 5 - Conclusiones y trabajo futuro

5.1 Conclusiones

En el presente trabajo se presenta una solución conceptual para identificación de sonidos mediante la aplicación de técnicas inteligentes basadas en aprendizaje profundo en un contexto IoT. La idea se plantea con fines de atajar la deforestación de bosques.

En el trabajo se proporciona una solución integrada, consistente en una serie de módulos que conforman una estructura completa, verificando su viabilidad en el contexto planteado y con posibilidades de ampliación e implantación a otros entornos donde el reconocimiento de sonidos constituya el núcleo central y la publicación de resultados en la nube sea también primordial para que los mismos puedan ser consultados en remoto, sin más requisitos que una conexión adecuada y con unos dispositivos de uso común.

Específicamente, el desarrollo planteado contiene los siguientes elementos y estructuras:

a) Captura y preparación de datos de audio.
b) Desarrollo de un procedimiento para la generación de espectrogramas como elementos de entrada a la red neuronal.
c) Diseño de una Red Neuronal Convolucional, convenientemente entrenada con los datos preparados, y con el fin de identificar audios, principalmente de sonidos de motosierra.
d) Diseño y configuración de un sistema de procesamiento remoto utilizando plataformas específicas como Drive y ThingSpeak, con envío de avisos y alarmas vía Twitter.

Los módulos anteriores se han integrado convenientemente, verificando el correcto funcionamiento de todos ellos en su conjunto y por separado, en ambos casos con resultados satisfactorios.

5.2 Trabajo futuro

Una vez desarrollado el modelo conceptual, se han identificado las siguientes posibles mejoras de futuro, que se sintetizan como sigue:

a) Implantación del sistema en entornos reales, dando el salto de la solución conceptual a la realidad.

b) Estudio de las posibilidades de adaptación a otros entornos donde la identificación de sonidos sea la clave, incluyendo sistemas para monitorización del ruido ambiental, tanto en entornos de interior como de exterior.
c) En el caso de monitorización del ruido ambiental a lo largo del tiempo, se puede pensar en la implementación de modelos de predicción, bien realizando los procesamientos en la nube a través de Matlab Drive o desarrollando aplicaciones de procesamiento en el propio ThingSpeak. De esta forma, si se dispone de datos suficientes, es posible desarrollar modelos de estimación autorregresivos o de tipo Long Short-Term Memory (LSTM) con redes neuronales recurrentes (Hochreiter y Schmidhuber, 1997), en este último caso también encuadradas en el paradigma del aprendizaje profundo.
d) Estudio de otros modelos arquitectónicos de redes neuronales convolucionales con distinto número de capas, e incluso añadiendo más capas de salida en función de otros tipos de audios y ruidos.
e) Crear una base de datos de audios propia para contextualizar los ruidos según diferentes entornos.

Chapter - Introduction

Preliminares

Everyday people are exposed to different amplitude noise, these noise sources include transportation e.g. trains, motorcycles, airplanes engines, air noise and at the same time a person-controlled machinery such as chainsaws, drills, industry equipment, tires, lawn mowers, mowers (Anderson, 2015). Animals also produced for different biological purposes, like guarding territory, drawn mates, dissuade predators, navigation, finding food and keeping contact with different social group members.

In this past year the amount of acoustic monitoring solutions has increased due to the many different solutions one can implement to monitor the distinct noises we can find around nature.

These solutions come from the priority to control to measure the contamination in cities and from this commitment we move along to building healthier acoustic surroundings (Gllo y col. 2018), determining wildlife richness and building biodiversity index studying the interactions and social dynamics from both of them (Darras y col. 2016) and the making of alarms from noise reproduction events that imply illegal activities, such as:

Rainforest Connection

The RFCx proyect creates acoustic monitoring systems to real time illegal deforestation. First park wardens are informed and the best will stop the illegal cut down. With the RFCx predictive analysis you can now predict with 3 minutes in advance when and where the chainsaw event will probably occur. A phone app will alert park wardens so they can show up before the recorders can begin. The prediction is based on historical sounds captured by old smartphones that transmit the tropical jungle sounds to an automatic learning model to detect these events (Rainforest Connection,2020).

Low Cost Iot Noise monitoring network in Torino.

Torino city decided to face the nightlife activities noise contamination problem with an integrated approach and established a low cost IoT monitoring network using Android smartphones in San Salvario district, where the amount of restaurant, bars, pubs and clubs attract thousands of people every weekend. The Regional Environmental Agency (ARPA Piemonte) developed an app to signal processing, data transmission and after a complete group of lab tests

for uncertainty evaluations and the definition of a calibration process, the deployment started in situ on the summer of 2006. Every hour data was analyzed after being collected for months in real time on the IoT OpenData platform, which led to a first exposure map to leisure noise and an annual detailed spectrum of noise level per hour.

This approach based on data has been chosen to impact in the political agenda, in a more positive participation of the focal group community, in the commitment of the local businesses and the client's awareness, to begin a better adaptation of open-air activities into daily life. The noise monitoring network allowed to measure the effects of new summer 2017 regulations and the evaluation of the general effect of the new measures to noise reduction predicted to 2018, in the Proyecto Monica Horizon 2020 framework (Gallo y col., 2018).

The examples above are just two ways to consider when it comes to the different applications related to Analysis and noise monitoring in different scenarios mainly in exterior environments. Also, thanks to the sensory technologies and IoT development is clear the increased tendency to bring this kind of solutions, where artificial intelligence is the protagonist. To add some examples besides the ones above specifically about IoT:

- ENVIRA Company gives intelligence solutions that allow to determine the acoustic contamination levels so administrations can learn the real time analysis and pick solutions that can diminish their effects. (ENIRA, 2020).

- INTELKIA also give solutions to fight against noise contamination with integrated solutions in platforms for Smart Cities (INTELKIA, 2020).

Considering all the information above, our present work is also oriented to the analysis of noise through a smart application based on convolutional neural networks at the same time establishing the necessary mechanisms to set the access to the cloud within the IoT paradigm. Thi work is oriented specifically to identify chainsaw sounds for prevention purposes due to illegal activities in forest habitats, this can be applied to any different sounds also.

Objectives

The general objective proposed in this work is to design an IoT system, as a feasible concept solution for the identification of sounds in natural environments. Obviously, given its nature, it is not possible to deploy it in a real environment, so the data under analysis is obtained from suitably selected audios. In this way, a proposal is made that provides a solution for its subsequent implementation in real environments, simply replacing the aforementioned data provided by a device equipped with a microphone capable of generating audio in real time.

As specific objectives the following are proposed,

- Design of a procedure for preparing the data from the available audios.
- Development of a procedure for extracting spectrograms from the audio signal, involving analysis techniques of the signal and its transformation to the frequency spectrum by Fourier transform.
- Design at the local level of a neural network, which accepts as input the spectrograms, considered as spectral images, in the indicated sense.
- Configuration of the remote platform (Matlab Drive) for network training and execution, with access from a mobile device.
- Design and configuration of the remote platform, in the cloud, for the storage and sending of alarms through Twitter.
- Integration of the modules that implement the previous processes.
- Analysis and evaluation of the results.

The planned work is based on the specific objectives outlined, contemplating the same number of phases as objectives, and with balanced planning for weeks throughout the project development time.

On the other hand, the objectives themselves determine the specific contribution to the project as an integrated solution.

Motivations and contributions

Once the initial approach and the objectives to be achieved have been established, the motivation for this project is twofold. On the one hand, the need to develop a sound recognition system in general and, on the other hand, the identification of sounds, in this case a chainsaw, in a natural environment to detect illegal activities, such as the uncontrolled cutting of trees in forests. protected. All this through the application of intelligent techniques, specifically in the field of deep learning using Convolutional Neural Networks (CNN) and under the IoT paradigm.

As previously indicated, a concept solution is provided, which allows determining its feasibility for its implementation in real environments, simply by adapting the data capture devices and the communications for the transmission of the same or the results of the processing in their case. This will undoubtedly contribute to environmental sustainability by identifying the aforementioned activities, while establishing the basis for their deployment in any environment, outdoor or indoor, the purpose of which is to identify sounds or noise levels in order that is established in each case.

Under these premises, the specific contributions made in this project are summarized as follows:

1. Feasibility analysis of the model to be implemented, defining the application structure, the data capture devices and processing tools to be used.
2. Analysis of the model structure and feasibility study regarding the transmission and processing of data from its entry into the system until its arrival in the cloud. Interfaces and connection possibilities through the internet are considered.
3. Study and analysis of audio signal processing methods, through the generation of spectrograms, which are processed as images and constitute the input data to the neural network model.
4. Study and adaptation of convolutional neural network architectures models in the field of image processing in order to process spectrograms, considered as images.
5. Generation of audio data, both for network training and for classifying them after training.
6. Design of the storage and processing model in the cloud, taking into account that remote platforms are used, as will be seen later (Drive and ThingSpeak).
7. Implementation of the different modules that make up the system and their integration to configure a single platform, with access to sensory devices.
8. Validation of the modules and the integrated model.

Organization of work

The work has been organized into five chapters, with the following content in the remaining chapters. Chapter two describes the methodology applied to develop the solution. Chapter three describes the design of the solution, which includes the architecture of the solution and data processing. Chapter four presents the analysis of the results obtained. Finally, chapter five presents the conclusions and future work that could be done.

Chapter - Conclusions and future work

Conclusions

In this work, it is presented a conceptual solution for the Identification of sounds by the application of smart techniques based on deep learning in an IoT context. The idea is proposed to stop deforestation.

This work provides an integrated solution that consists in a series of modules that shape a complete structure, verifying its viability in the already set context and with possibilities to expand and implement to other environments where sound recognition is the central core along with the result publication on the cloud so these ones can be accessed remotely, with no further requirements than just an adequate connection and common use devices.

Specifically, the proposed development contains these next elements and structures

a) Capture and preparation of audio data
b) Development of a spectrogram generation procedure as input elements of the neural network.
c) Design of a Convolutional Neural Network conveniently trained with prepared data to identify audios mainly chainsaw sounds.
d) Design and configuration of a remote processing system using specifically platforms such as Drive and ThingSpeak, with a notification and alarm sending via Twitter.

The previous modules have been integrated conveniently, verifying the right functioning of all of them as a whole and separately, in both cases the results are satisfactory.

Future work

Once the conceptual model was developed, the following possible future improvements have been identified, which are summarized as follows:

a) Implementation of the system in real environments, making the leap from conceptual solution to reality.

b) Study of the possibilities of adaptation to other environments where the identification sounds is the key, including systems for monitoring environmental noise, both indoors and outdoors.

c) In the case of monitoring environmental noise over time, you can think about implementing prediction models, either by carrying out the processing in the cloud through Matlab Drive or by developing processing applications in ThingSpeak itself. In this way, if sufficient data is available, it is possible to develop autoregressive or Long Short-Term Memory (LSTM) estimation models with recurrent neural networks (Hochreiter and Schmidhuber, 1997), in the latter case also framed in the paradigm of deep learning.

d) Study of other architectural models of convolutional neural networks with different number of layers, and even adding more output layers depending on other types of audio and noise.

e) Create your own audio database to contextualise the noises according to different environments.

BIBLIOGRAFÍA

1. Anderson J. (2015) Intelligent ambient sound monitoring system U.S. Patent No. 9191744 B2.
2. Darras, K., Pütz, P., Fahrurrozi, Rembold, K.,Tscharntke, T.(2016) Measuring sound detection spaces for acoustic animal sampling and monitoring, Biological Conservation 201:29–37
3. Duda, R.O, Hart, P.E., Stork, D.G. (2001). Pattern Classification. John Wiley and Sons, NY, USA.
4. ENVIRA Monitorización de ruido (2020). https://enviraiot.es/monitorizacion-de-ruido/ (accedido Marzo 2020).
5. FreeSound (2020). Disponible on-line: www.freesound.org (accedido Junio 2020).
6. Free Sound Effects (2020). Disponible on-line: www.freesoundeffects.com (accedido Diciembre 2019).
7. Gallo, E., Ciarlo, E., Santa, M., Sposato, E. Fogola, J., Grasso, D. (2018). Analysis of leisure noise levels and assessmentof policies impact in San Salvario district, Torino (Italy), by low-cost IoT noise monitoring network. Euronoise, Crete (Greece), 27-31 May;
8. Giannakopoulos, T., Pikrakis, A. (2014). Introduction to Audio Analysis: a Matlab Approach, Elsevier, Oxford.
9. Goodfellow, I., Bengio, Y., and Courville, A. (2016). Deep learning. MIT Press. USA. Disponible on-line: https://www.deeplearningbook.org/ (accedido Enero 2020)
10. Hochreiter, S., Schmidhuber, J. (1997). Long short-term memory. Neural Computation, 9(8), 1735–1780.
11. ImageNet (2020). Disponible on-line: http://www.image-net.org (accedido Marzo 2020).
12. INTELKIA IN_NOISE – Solución IoT para la monitorización del ruido (2020). http://www.intelkia.com/productos/in-noise-solucion-iot-monitorizacion-ruido/ (accedido Marzo 2020)
13. Kingma, D.P., Ba, J.L. (2014). Adam: A method for stochastic optimization. arXiv:1412.6980v9.

14. Krizhevsky, A., Sutskever, I., Hinton, G.E. (2012) ImageNet Classification with Deep Convolutional Neural Networks. In Proc. 25th Int. Conf. on Neural Information Processing Systems (NIPS'12), vol. 1, pp. 1097-1105.
15. Mathworks (2020). Speech Command Recognition Using Deep Learning. Disponible on-line: https://es.mathworks.com/help/deeplearning/ug/deep-learning-speech-recognition.html (accedido Marzo 2020).
16. Matlab Drive (2020) Disponible on-line: https://es.mathworks.com/products/matlab-drive.html (accedido Marzo 2020).
17. Matlab Mobile (2020). Disponible on-line: https://es.mathworks.com/products/matlab-mobile.html (accedido Marzo, 2020).
18. McLoughlin, I. (2009) Applied Speech and Audio Processing With MATLAB Examples, Cambridge University Press, Cambridge, UK.
19. Orozco Medina, M. G., & González, A. E. (2015). La importancia del control de la contaminación por ruido en las ciudades. Redalyc, 129-136. Obtenido de https://www.redalyc.org/html/467/46750925006
20. Pajares, G. (2017) Análisis y Reconocimiento de Voz. RC-Libros, Madrid.
21. Rainforest Connection (2020) Disponible on-line: https://rfcx.org (accedido Marzo 2020).
22. Russakovsky, O., Deng, J., Su, H. Krause, J., Satheesh, S., Ma, S. Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C., Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. International Journal of Computer Vision (IJCV). 115(3), 211–252.
23. Srivastava, N., G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. Journal of Machine Learning Research, 15, 1929-1958
24. ThingSpeak (2020). Thing Speak for IoT Projects. Disponible on-line: https://thingspeak.com/. (accedido Marzo 2020).
25. ThingSpeakWeb (2020). Disponible on-line: https://thingspeak.com/channels/public (accedido Marzo, 2020).
26. ThingView (2020). ThingSpeak Viewer. Disponible on-line: https://play.google.com/store/apps/details?id=com.cinetica_tech.thingview&hl=es (accedido Marzo, 2020)

APÉNDICES

El código, junto con los datos de procesamiento y configuración se ubican en el repositorio siguiente: https://github.com/cjustinol/TFMiot. Tras su descarga y descompresión se genera el directorio raíz *TFMiot-master* con el contenido de datos y ficheros de configuración que se describen en la sección A. En la sección B se describen los programas que conforman la aplicación. Finalmente, en la sección C se describe el procedimiento para la ejecución de los mencionados programas.

A. Datos y ficheros de configuración

a) La carpeta *RuidosReconocimiento*, bajo el directorio raíz, contiene las siguientes subcarpetas, con los ficheros necesarios para realizar el entrenamiento, cuyos nombres identifican las clases de ruido utilizadas:

- *motosierra*, con 770 audios
- *trafico*, con 526 audios
- *television*, con 515 audios
- *ruido_fondo*, con 6 audios, que son parte del repositorio Google Speech Commands data set, bajo Creative Commons BY 4.0 license, disponible en http://download.tensorflow.org/data/speech_commands_v0.02.tar.gz.

La carpeta *motosierra* contiene los audio clips clave para el entrenamiento. Las carpetas *tráfico* y *televisión* contienen audios propios para la diferenciación de distintos tipos de sonidos en clasificación, captados a partir de sonidos de ruidos medioambientales en escenarios de tráfico y con una televisión de fondo, mientras que la carpeta *ruidos_fondo* contiene audios provenientes del repositorio mencionado, que fue creado por los equipos de TensorFlow y AIY (https://aiyprojects.withgoogle.com/) conteniendo 65.000 clips de un segundo de duración y que sirven para introducir perturbaciones en los sistemas de reconocimiento de sonidos, en este caso de voz.

b) En la misma carpeta *RuidosReconocimiento* se ubican los ficheros de texto *validation_list.txt* y *testing_list.txt* que contienen los ficheros a utilizar para Validación y Test, durante la fase de entrenamiento. El resto de los ficheros disponibles y no incluidos en estos listados son los que se utilizan para el entrenamiento (*Training*).

c) En la carpeta raíz se ubica el fichero *RuidosVarios (formado uniendo varios audios).wav*, utilizado para el proceso de clasificación y descrito en profundidad en la sección 4.4.

d) En la carpeta raíz se ubica también el fichero *trainedNet.net*, que contiene un modelo entrenado, con los pesos aprendidos.

B. Programas en Matlab

Los programas en código Matlab, ubicados en el directorio raíz son los siguientes:

a) DeepEntrenamientoMotosierraMovil.m, con las siguientes funcionalidades:

- Organización del almacenamiento de los audio-clips para entrenamiento, validación y test. Utiliza la función la función contenida en *splitData.m*.
- Definición de las categorías de las clases, que se estructuran en *motosierra,* para los audios contenidos en la subcarpeta del mismo nombre y *fondo* (audios de las subcarpetas *tráfico* y *television*). Se añaden los audios de la carpeta *ruido_fondo* para introducir el mencionado ruido adicional con los audios contenidos en la misma.
- Generación de los espectrogramas de los audios contenidos en las carpetas anteriores, mediante la aplicación de la transformada discreta de Fourier.
- Normalización de los valores de los espectrogramas al rango de valores [-1,1], para favorecer que todos los datos contribuyan por igual en el proceso de aprendizaje.
- Visualización de algunos de los espectrogramas generados, y de la distribución del número de espectrogramas asignados para entrenamiento, validación y test.
- Definición de la arquitectura de la RCN, con la estructura de capas de la figura 2-6. Entre ellas se crea la capa final de clasificación mediante la función definida en el fichero *weightedClassificationLayer.m*.
- Definición de los parámetros del proceso de entrenamiento de la red, incluyendo: método de optimización, razón de aprendizaje inicial, número de *epochs*, dimensión de los lotes o factor de reducción de la razón de aprendizaje.
- Inicio del proceso de entrenamiento, de forma que tras su finalización se obtiene el modelo de red con los pesos ajustados, de acuerdo con los audios de entrenamiento suministrados. El fichero *trainedNet.net* es uno de los modelos obtenidos, que se almacena convenientemente para la fase de clasificación.
- Realización de la fase de test y visualización de los resultados obtenidos con los audios de test asignados previamente.

b) DeepClasificacionMotosierraMovil.m, con las siguientes funcionalidades:

- Definición de las claves de acceso al canal de ThingSpeak y configuración de las variables para acceso de escritura en la nube.

- Carga del modelo RCN entrenado en el paso previo, contenido en el fichero *trainedNet.net*.
- Carga del fichero *RuidosVarios (formado uniendo varios audios).wav*
- Extracción de las fracciones de audio a partir de este fichero, con los cuales de generan los correspondientes espectrogramas, por aplicación de la transformada discreta de Fourier con el mismo criterio y procedimiento que el aplicado en la fase de entrenamiento, además de la correspondiente normalización de los valores de dichos espectrogramas.
- Clasificación de cada espectrograma con al RCN.
- Acceso a ThingSpeak para reportar los resultados de la clasificación. Esta acción genera la actualización de los datos en los canales de ThingSpeak con el envío del correspondiente Tweet, si procede.

Una vez finalizado este proceso, los datos actualizados en los canales de ThingSpeak quedan disponibles para su visualización a través de la web o de la App ThingView desde un dispositivo móvil.

C. Procedimiento de ejecución

El proceso para la ejecución de los programas definidos previamente se realiza como sigue:

- Instalación de los programas y modelo de la red en Drive, además del fichero *RuidosVarios (formado uniendo varios audios).wav*, para que éste pueda ser accedido desde el dispositivo móvil. Si el proceso de clasificación se realiza en el computador, este fichero puede ubicarse en cualquier carpeta a la que tenga acceso el computador.
- Ejecución del programa *DeepEntrenamientoMotosierraMovil.m* desde el Drive o proporcionar la vía de acceso a dicho fichero.
- Ejecución del programa *DeepClasificationMotosierraMovil.m* desde el Drive o proporcionar la vía de acceso al modelo generado en el paso anterior.
- Visualizar los datos almacenados en ThingSpeak mediante acceso web o la App ThingView instalada en el dispositivo móvil.
- De haber sido reconocido algún sonido de motosierra visualizar el tweet correspondiente.