Universidad del País Vasco Euskal Herriko Unibertsitatea
BILBOKO INGENIARITZA ESKOLA
ESCUELA DE INGENIERÍA DE BILBAO

GRADO EN INGENIERÍA EN TECNOLOGIA DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

Evaluación de un sistema de Machine Listening para la clasificación de sonidos urbanos

Alumno/Alumna: <González, Gorostiaga, Peio>

Director/Directora (1): <Saratxaga, Couceiro, Ibon>

Curso: <2019-2020>

Fecha: <Bilbao, a 17 de julio de 2020>

Resumen

Las técnicas de *Deep Learning* o aprendizaje profundo cada vez se aplican más al mundo real, y están en continuo desarrollo en busca de nuevas utilidades. Son muy conocidas las técnicas utilizadas para la clasificación de imágenes basadas de redes neuronales convolucionales, pero no tanto para la clasificación acústica. El grupo de investigación AHOLAB ha desarrollado un modelo convolucional capaz de detectar y clasificar los eventos sonoros de un automóvil con el fin de alertar de posibles averías en él. AHOLAB tenía el deseo de ampliar las funcionalidades de su red neuronal y extenderla a nuevos campos de uso. Es por ello por lo que mediante este trabajo se ha reproducido la arquitectura desarrollada por el grupo de investigación y se ha aplicado a la clasificación de eventos urbanos. Para lograr el objetivo, durante este trabajo se han ido adecuando los hiperparámetros del modelo convolucional para que tenga la capacidad de clasificar eventos sonoros detectados en cualquier punto de una ciudad. Todo esto se ha realizado a la vez que se ha ido preparando una arquitectura similar para presentarla a un concurso llamado DCASE en el cual también se debían clasificar sonidos urbanos. Gracias al material ofrecido por la organización del concurso se ha podido formar una gran base de datos con la que alimentar y entrenar la red neuronal y se ha podido evaluar hasta qué punto de precisa puede llegar a ser.

Palabras clave: *Deep learning*, red neuronal convolucional, hiperparámetros, entrenar

Laburpena

Deep Learning edo ikaskuntza sakoneko teknikak gero eta gehiago aplikatzen dira mundu errealean, eta etengabe garatzen ari dira erabilera berrien bila. Oso ezagunak dira sare neuronal konboluzionaletan oinarritutako irudiak sailkatzeko erabiltzen diren teknikak, baina ez hainbeste sailkapen akustikorako. AHOLAB ikerketa-taldeak auto baten soinu-gertaerak detektatzeko eta sailkatzeko gai den eredu konboluzional bat garatu du, automobilak izan ditzakeen matxuren berri emateko. AHOLAB bere sare neuronalaren funtzionalitateak zabaldu eta erabilera eremu berrietara hedatzeko nahia zuen. Horregatik, ikerketa-taldeak garatutako arkitektura erreproduzitu da lan honen bidez, eta hiri-ekitaldien sailkapenari aplikatu zaio. Helburu hori lortzeko, lan honetan zehar konboluzio-ereduaren hiperparametroak egokitu dira, hiriguneetan antzemandako soinu-gertaerak sailkatzeko gaitasuna izan dezan. Guzti hau, DCASE izeneko lehiaketa batera aurkezteko antzeko arkitektura bat prestatuz joan da, non hiriko soinuak ere sailkatu behar ziren. Lehiaketaren antolatzaileek eskainitako materialari esker, sare neuronala elikatzeko eta entrenatzeko datu-base handi bat osatu ahal izan da, eta sistema zenbaterainokoa izan daitekeen ebaluatu ahal izan da.

Hitz gakoak: *Deep Learning*, sare neuronal konboluzional, hiperparametro, entrenamendua

Abstract

The techniques of Deep Learning are increasingly applied to the real world and are in continuous development in search of new uses. The techniques used for the classification of images based on convolutional neural networks are well known, but not so much for acoustic classification. The AHOLAB research group has developed a convolutional model capable of detecting and classifying the sound events of a car in order to warn of possible breakdowns in it. AHOLAB wanted to extend the functionalities of its neural network and extend it to new fields of use. That is why through this work the architecture developed by the research group has been reproduced and applied to the classification of urban events. To achieve this objective, during this work the hyperparameters of the convolutional model have been adapted so that it has the capacity to classify sound events detected in urban places. All this has been done at the same time that a similar architecture has been prepared to be presented to a contest called DCASE in which urban sounds had to be classified as well. Thanks to the material offered by the organization of the competition, a large database has been formed with which to feed and train the neural network and it has been possible to evaluate how accurate it can be.

Key words: Deep learning, convolutional neural network, hyperparameters, train

Índice

Lista de ilustraciones

Lista de tablas

Lista de acrónimos

BSR	Buzz, squeak and rattle
CPU	Central Processing Unit
CNN	Convolutional Neural Network
CRNN	Convolutional Recursive Neural Network
DDR4	Double Data Rate type four Synchronous Dynamic Random-Access Memory
DNN	Deep Neural Network
DP	Deep Learning
GNU	GNU is Not Unix
GPU	Graphics Processing unit
HDD	Hard Drive Disk
IA	Inteligencia Artifical
MFB	Mel filter bank
ML	Machine Learning
MLP	Multilayer Perceptron
SDD	Soft Drive Disk
SSH	Secure SHell
RAM	Random Access Memory
RNN	Recursive Neural Network
ReLU	Rectified Linear Units
TFG	Trabajo Fin de Grado

1. Introducción

El aprendizaje profundo o *Deep Learning* es un concepto que está en constante desarrollo y que en los últimos tiempos la evolución que ha sufrido dicha tecnología ha sido tremendamente rápida y variada. Progresivamente se va introduciendo en la vida cotidiana, en la mayoría de las ocasiones sin que la gente que no tiene conocimiento previo de esta tecnología se percate, pero cada vez va adquiriendo una importancia mayor. Ejemplo de esto, y de su uso actual, puede ser la orientación de anuncios y predicción de preferencias de los clientes en la web, la ayuda en la mejora en la comprensión de enfermedades y sus mutaciones, el análisis de imágenes médicas como radiografías y resonancias magnéticas, la localización de caras e identificación de emociones faciales, reconocimiento de voz, la clasificación de videos, etc.

El *Deep Learning*, como parte de la inteligencia artificial, es capaz de adquirir conocimientos humanos, y además se le suma la capacidad de aprender, utilizando redes neuronales profundas (en inglés conocidas *como Deep Neural Networks*). Pero hay un campo, el de la clasificación de sonidos ambientales, que no está tan avanzado como el resto, como puede ser la clasificación de imágenes. En los trabajos realizados anteriormente en el ámbito de clasificación de sonidos urbanos, destaca la gran utilización de los modelos convolucionales, y en la mayoría de los casos con resultados satisfactorios.

AHOLAB, grupo de investigación con el que se realiza el trabajo, en el año 2018 desarrolló un modelo convolucional capaz para detectar y clasificar eventos sonoros de un automóvil, y de esta forma alertar de posibles errores en él. Los resultados fueron positivos, por lo que AHOLAB pretende continuar desarrollando su arquitectura e implementarla en la clasificación de sonidos urbanos, siendo conscientes de que para ello deben hacer algunas modificaciones en los parámetros de configuración (también conocidos como hiperparámetros) del modelo si se quieren conseguir unos resultados aceptables.

En este proyecto se aplicará la arquitectura de AHOLAB a la clasificación de sonidos ambientales. Para lograr el objetivo, se hará un estudio detallado del modelo para luego hacer una réplica adaptada al nuevo campo de uso. Después, se entrenará el modelo haciendo algunos ajustes en los hiperparámetros, y cada vez que termine un entrenamiento se hará un análisis de los resultados para determinar cómo debe ser el siguiente proceso a lanzar. Además, se pretende tomar parte en el concurso DCASE, que consiste en clasificar sonidos urbanos, y que gracias al material ofrecido por la organización se podrá formar una amplia base de datos con la que alimentar el sistema, y al mismo tiempo se podrá evaluar la precisión de la nueva arquitectura. En resumen, en este proyecto se aplicará y evaluará el sistema de AHOLAB en la clasificación de sonidos urbanos

Mediante este documento, se detallan los objetivos a cumplir en el proyecto, la metodología seguida, el diseño aplicado y se analizan los resultados obtenidos.

2. Contexto

En este apartado se va a explicar cuál es la situación inicial de una manera global atendiendo a los avances y estudios realizados a nivel mundial relacionados con la tecnología usada en este trabajo. A continuación, se concretará la situación en la que se encontraba el grupo de investigación AHOLAB antes del comienzo del proyecto ya que como base se va utilizar su red neuronal. Por último, se justificarán las motivaciones que han hecho que se lleve a cabo este proyecto.

La clasificación automática del sonido ambiental es un área de investigación en crecimiento con numerosas aplicaciones en el mundo real. Si bien existe una gran cantidad de investigaciones en los campos de audio, como el habla y la música, el trabajo en la clasificación de los sonidos ambientales es relativamente escaso.

En los últimos años, las técnicas de aprendizaje automático y aprendizaje profundo se han iniciado en el campo del procesamiento de audio, tanto en analizar y procesar señales de voz como en otros muchos tipos de eventos sonoros. El aumento de uso de redes neuronales profundas, más conocidas como *Deep Neural Networks* (DNN) han ofrecido nuevas maneras de abordar los clásicos problemas de análisis y procesamiento de señales sonoras como la detección de sonido, reconocimiento de voz, clasificación sonora, síntesis… Estos avances en la IA han producido grandes mejoras en cuanto a rendimiento y resultados, y con esto han permitido a muchas aplicaciones de detección y procesado de audio romper sus limitaciones de uso para ser utilizadas por el gran público.

Las nuevas técnicas de *Deep Learning* o aprendizaje profundo tienen características muy provechosas para afrontar los complejos problemas que surgen en el mundo real: tienen la capacidad de trabajar con grandes cantidades de datos de entrada, son capaces de aplicar procesamiento no lineal, pueden extraer su propio conocimiento a partir de datos, permiten detectar patrones en condiciones de ruido muy desfavorables, se pueden adaptar a condiciones variables o adversas, etc. Como parte negativa de estas redes, está la dificultad de entrenamiento de las mismas ya que requieren una extensa base de datos y a su vez una capacidad computacional muy grande, ya que los cálculos a realizar tienen una complejidad muy elevada.

Dentro del *Deep Learning* y de las *Deep Neural Networks*, se encuentran las redes neuronales convolucionales, en inglés conocidas como *Convolutional Neural Networks* o CNN, las cuales están especializadas en el reconocimiento de imágenes y audios. Concretamente, las redes convolucionales están formadas por un numero de capas convolucionales y de capas *pooling* (capas de agrupación) encargadas en buscar patrones en la señal de entrada y reducir su tamaño. Por lo tanto, para el problema planteado en este trabajo, las CNN funcionarían mejor que los DNN genéricos, ya que captura implícitamente la estructura de las imágenes y los audios.

Pero por lo que a este trabajo respecta, que se va trabajar con una base datos que contiene archivos de audio, hay que mencionar que a las redes neuronales no se les suele alimentar directamente con señales sonoras, ya que supone un gran coste computacional. Como se ha comentado anteriormente, estas redes son capaces de trabajar con imágenes, por lo que en la mayoría de estudios realizados hasta el momento los audios han sufrido una conversión a imagen. En gran parte de esos estudios los audios han sido transformados a una de las dos siguientes opciones: la primera conversión que se puede aplicar es trazar la forma de onda directamente (esto requiere de otras arquitecturas de red que no son convolucionales), y la segunda conversión que se puede aplicar es lograr el espectrograma de la señal, y de esta forma se logra alimentar la CNN como si la entrada original de una imagen se tratase. Hay que destacar, que la técnica más utilizada es la del espectrograma ya que en diversas investigaciones se ha demostrado ser más efectiva.

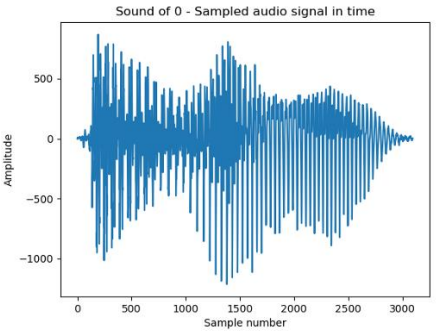

Ilustración 1: Amplitud pronunciación 0

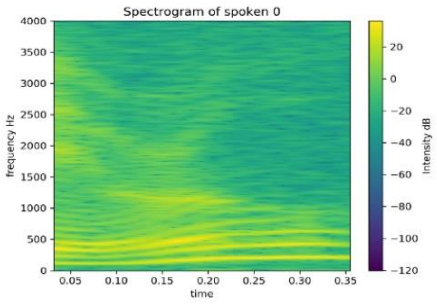

Ilustración 2: Espectrograma pronunciación 0

El grupo de investigación AHOLAB, con el cual se ha realizado el trabajo, ha desarrollado su propia arquitectura para aplicarla a la clasificación de audios. En 2018, en un trabajo realizado para la empresa automovilística Mercedes-Benz, desarrolló un modelo convolucional capaz de detectar y clasificar sonidos producidos en un coche. De esta forma, la compleja tarea de evaluación de posibles errores en un vehículo se convirtió en más sencilla desde el punto de vista humano, ya que el sistema se encargaba de detectar todos los sonidos relevantes, y de clasificarlos según de donde proviniese, y por último evaluar si se podría tratar de un posible defecto. En definitiva, gracias al sistema creado por AHOLAB la evaluación acústica se ha convertido en una fuente de información para la detección y el diagnostico de problemas de diseño, montaje o funcionamiento del vehículo para la empresa Mercedes.

2.1. Motivaciones

Tal y como se ha dicho en el apartado anterior, el modelo convolucional creado por AHOLAB ha demostrado su efectividad en la clasificación y detección de eventos sonoros de un automóvil. Pero al grupo de investigación se le originó la motivación de comprobar si su arquitectura era capaz de aclimatarse a nuevos campos de trabajo. Su intención era probarla en la clasificación de sonidos urbanos. Es evidente que una arquitectura calcada a la creada para Mercedes no obtendría los resultados deseados con los sonidos urbanos, por lo que el grupo de investigación pretende hacer un ajuste de los hiperparámetros del modelo para adecuarlo al nuevo espacio de trabajo. Además, se desea demostrar cómo aplicando técnicas de aprendizaje profundo a la clasificación de sonidos ambientales, centrándose específicamente en la identificación de sonidos urbanos, se pueden conseguir resultados positivos. Por todo esto, se ha visto la necesidad de poner en marcha este duro y bonito trabajo con el fin de cumplir con los objetivos y necesidades de AHOLAB.

3. Objetivos y alcance del trabajo

Una vez visto el punto de partida y conocido los antecedentes de este trabajo, a través de este apartado se pretende explicar de una forma breve, clara y concisa cuales son los objetivos de este proyecto. El objetivo principal del trabajo es aplicar el sistema desarrollado por AHOLAB a la clasificación de sonidos ambientales. Para lograr este objetivo principal se han planteado otros objetivos secundarios o específicos que sirven de apoyo para conseguir el objetivo principal.

3.1. Objetivo principal

- **Aplicar el modelo convolucional creado por AHOLAB a la clasificación de sonidos urbanos**: El objetivo es reproducir una arquitectura similar a la de AHOLAB para aplicarla a sonidos ambientales. Se plantea el análisis de si la arquitectura diseñada por AHOLAB para detectar los eventos sonoros dentro de un automóvil sirve también para clasificar los eventos sonoros de una ciudad de una forma óptima. Para ello, se buscará reproducir la arquitectura con una serie de variantes con las cuales se pretende mejorar los resultados, y luego esa misma arquitectura será puesta a prueba para analizar si los resultados son aceptables.

3.2. Objetivos específicos

- **Intentar superar la *Baseline* propuesta por el *Challenge* DCASE:** Se plantea el reto de comprobar si los resultados obtenidos en el objetivo principal son positivos. Para ello se tomará parte en un concurso llamado DCASE el cual se basa en clasificar los eventos sonoros urbanos. La organización del concurso ofrece un sistema ya desarrollado, y a su vez publican el porcentaje de precisión que tiene ese sistema, también conocido como *Baseline*. El reto es superar, o por lo menos acercarse, a los resultados conseguidos por la *Baseline* utilizando la arquitectura de AHOLAB.

- **Formar en modelos convolucionales:** Mediante este objetivo secundario se pretende formar y entender las redes convolucionales. Cada día estos modelos avanzan y desarrollan nuevas técnicas, por lo que en este trabajo se analizarán y aplicarán los avances más recientes. Concretamente, la formación se centrará en las redes convolucionales aplicadas a la clasificación de sonidos urbanos. A su vez, se analizará el sistema previamente desarrollado por AHOLAB para la clasificación de eventos sonoros

de un automóvil diseñado para la empresa Mercedes basado en redes convolucionales. Un buen estudio de este sistema es de vital ayuda para conseguir el objetivo principal del proyecto.

4. Beneficios que aporta el trabajo

En este apartado se valorarán los diferentes beneficios que puede llegar aportar este trabajo en diferentes ámbitos. Se van a destacar principalmente los beneficios técnicos, sociales y económicos, aunque este trabajo pueda llegar contribuir beneficios en otros muchos ámbitos.

4.1. Beneficios sociales

Con este trabajo se ayuda a la clasificación de ruidos y eventos sonoros sucedidos en una ciudad o en cualquier espacio ambiental. El sistema diseñado clasifica los ruidos en diferentes tipos de clases como podrían ser ruidos de coches, voces humanas, sonidos meteorológicos, animales… Las aplicaciones del *Machine Listening,* y que este sistema puede llegar a lograr, pueden ser múltiples, por ejemplo, en una ciudad puede ayudar al control de tráfico detectando si hay demasiado ruido de coche o puede advertir de posibles agresiones detectando los gritos del agredido. También puede ser de gran ayuda para las personas sordas en sus respectivas actividades diarias ofreciéndoles una asistencia de detección de posibles peligros mediante la acústica. Otro ámbito de uso es el hogar, siendo un complemento fundamental en el uso inteligente de los recursos pudiendo detectar posibles intrusos. En beneficio de las empresas, el sistema puede tener un uso como mantenimiento predictivo de la maquinaria. Además de los beneficios mencionados, a estos se les pueden sumar otros tantos, y en un futuro se le pueden dar nuevas utilidades sin tener que cambiar la arquitectura del sistema dependiendo las nuevas necesidades.

4.2. Beneficios económicos

En lo que a los beneficios económicos respecta, se podrían distribuir en dos subcategorías: los beneficios al grupo de investigación AHOLAB y los beneficios a otras empresas. Con este sistema, AHOLAB dispondría de un sistema de gran valor y en caso de estar interesados de sacarlo al mercado muchas empresas estarían dispuestas a adquirirlo, como ya se ha demostrado con otros proyectos anteriores desarrollados por AHOLAB.

Esas empresas, a su vez, pueden reducir costes de sus presupuestos valiéndose del sistema. El sistema no está limitado a una sola modalidad de uso por lo que cada empresa podría darle una funcionalidad diferente. Uno de los posibles usos que puede tener está relacionado con la seguridad, ya que si se instala en diferentes puntos de la empresa el sistema puede avisar de ruidos anómalos como voces humanas de posibles intrusos, y de esta forma la empresa podría reducir costes en personal de seguridad.

4.3. Beneficios técnicos

En cuanto a los beneficios técnicos que este trabajo ofrece, son muy variados. Desde hace un tiempo tanto la Inteligencia Artificial como el *Deep Learning* han sufrido grandes avances. Hasta el momento la mayoría de trabajos realizados usando la tecnología *Deep Learning* se han basado en la clasificación de imágenes, pero mediante este trabajo se pretende demostrar que su uso también se puede extender a la clasificación de audios, lo cual es un beneficio de cara al futuro ya que este trabajo puede servir de base para otros, tanto para el grupo de investigación AHOLAB como para otras empresas.

Por otro lado, se va hacer una investigación sobre las redes convolucionales y su aplicación a la clasificación de sonidos urbanos. Es un área muy poco explorada por lo que un correcto análisis de los resultados servirá para futuros trabajos sobre estas redes en la aplicación a sonidos ambientales y otras aplicaciones.

5. Estado del arte

Los trabajos realizados anteriormente en la clasificación de sonidos ambientales no abundan. En esta sección se va tratar de seleccionar y resumir los trabajos más importantes, sobre todo los que tengan relación con las redes neuronales convolucionales, ya que en nuestro proyecto se trabajará con ellas. Hay que destacar que la clasificación de sonido ambiental tiene una larga historia tal y como se cuenta en Chachada and Kuo[1], pero en nuestro caso nos centraremos sobre todo en el uso de redes neuronales convolucionales en la clasificación de entornos sonoros. En este apartado también se expondrá el *Challenge* DCASE, un concurso basado en *Urban Sound Clasification* en cual tomaremos parte.

5.1. Historia de las redes neuronales

Los orígenes de las redes neuronales se remontan a 1943, año en el que W. McCulloch y W. Pitts crearon un primer modelo de redes neuronales basándose en las matemáticas y algoritmos denominados lógica de umbral. Poco después de sus avances, Donald Hebb explicó a grandes rasgos el aprendizaje neuronal, conociéndose su ley como la "regla de Hebb"[2], la cual es precursora de las técnicas de aprendizaje por parte de redes neuronales en la actualidad.

Posteriormente, en 1956, se produjo en Dartmouth la primera conferencia en la que se habló de la capacidad de la computación de simular el aprendizaje de los cerebros biológicos. Al poco de esto (1960) llegó la primera aplicación de las redes a problemas reales: La eliminación de ecos en líneas telefónicas mediante el Adaline (Adaptative 4 Linear Elements) y el Madaline (Multiple Adaptative Linear Elements).

Sin embargo, en 1969, con la publicación del libro "Perceptrons: An introduction to Computational Geometry" por parte de Marvin Minsky y Seymour Papert, el interés por las redes neuronales cayó repentinamente dado que en el libro se demostraron importantes deficiencias en los modelos neuronales artificiales que se habían desarrollado hasta ese momento. A pesar de dicha caída la investigación no se frenó, destacando en 1977 el trabajo realizado por James Anderson y su extensión "Brain-State-in-a-Box" [3], la cual permitió modelar funciones de mayor complejidad.

El resurgimiento de las redes neuronales se produjo en 1985 con la publicación por parte de John Hopfield del libro "Computación neuronal de decisiones en problemas de optimización"[4], un año más tarde, en 1986, David Rumelhart introdujo en su versión actual el algoritmo de propagación hacia atrás, lo cual provocó un nuevo panorama mucho más alentador en las investigaciones y desarrollo de redes neuronales.

Desde entonces observamos numerosos esfuerzos en el campo de las redes neuronales, produciéndose avances tanto en software como en hardware.

En cuanto a las redes convolucionales, permiten utilizar las neuronas artificiales para aprovechar la estructura local de los datos de entrada bidimensionales, por ejemplo, una imagen. Cada neurona, en vez de estar conectada a todas las entradas provenientes de la capa previa, se limita sólo a procesar una pequeña parte del espacio de entrada, un bloque de NxM pixeles, por ejemplo), lo que se denomina el campo receptivo. Esta unidad convolucional, también llamada kernel o filtro, se mueve por toda la entrada, con lo que se obtiene un mapa de características. Así podemos tener un filtro convolucional que detecte un determinado patrón en la entrada bidimensional (por ejemplo, un sector de arco) y el mapa de características que obtenemos tras aplicar la CNN nos indicará si ese patrón aparece en alguna parte de los datos de entrada. En las CNNs se utilizan múltiples filtros para buscar diferentes patrones. Además, pueden apilarse CNNs de forma que los filtros funcionen jerárquicamente.

Las redes neuronales convolucionales han visto su uso extendido en los últimos años. Además, sus aplicaciones son variadas y abarcan clasificación de imágenes, detección de objetos, segmentación de los mismos, etc. En el campo de la clasificación de imágenes se ha avanzado enormemente desde la introducción en 2012 del modelo “AlexNet”. Desde aquel modelo, que conseguía un 57% de precisión, se ha avanzado a un escenario en el que los principales modelos se mueven en torno al 75% de precisión en el *dataset* de ImageNet[5].

5.2. Clasificación de sonido ambiental

La clasificación de sonidos ambientales mediante redes neuronales ha sido un campo al que le ha costado desarrollarse. Hasta hace unos años, los trabajos relativos a este campo no abundaban, pero hay que decir, que cada vez se están abriendo más investigaciones que relacionan los modelos convolucionales y la clasificación de sonido ambiental.

La clasificación de ruido urbano es un campo actualmente en crecimiento. Crear una base de datos es complicado, por lo que lo habitual es apoyarse en bases de datos ya creadas. La base de datos gratuita más extensa de audio etiquetado de ruido urbano es *UrbanSound*, recopilada por los grupos de investigación *Music and Audio Research Laboratory y Center for Urban Science and Progress* de la universidad de Nueva York. Proponen una taxonomía bastante completa a la hora de clasificar sonidos del entorno. Sin embargo, la base de datos contiene únicamente 10 clases.

En este apartado se van a mencionar los trabajos más destacables hechos hasta la fecha en relación a la clasificación de sonido ambiental.

5.2.1. Çakir, 2014

La tesis[6] plantea por primera vez el uso de una red neuronal profunda completamente conectada *feed-forward*, aplicada al problema de clasificación multietiqueta, también llamado clasificación polifónica, en el que varias clases pueden solaparse en el tiempo en una señal. Çakir compara como parámetros de entrada MFCCs, espectro filtrado mel y el logaritmo del espectro filtrado mel, plantea 20 clases de sonido diferentes y obtiene una precisión en el etiquetado en torno al 65% considerando bloques de un segundo.

5.2.2. Piczak, 2015

Piczak propone[7] un clasificador de fragmentos cortos de sonidos ambientales basado en una red neuronal convolucional o CNN. Las redes convolucionales se usan mucho en el ámbito del reconocimiento de imagen, pero también en habla y en música han tenido mucho éxito. Piczak propone una arquitectura basada en dos capas convolucionales con sendas etapas de *max-pooling,* para reducir la dimensionalidad de la salida de ambas capas, tras las cuales se utilizan dos capas de redes neuronales completamente conectadas seguidas de una capa de salida con tantas salidas como clases hay en el problema (es un clasificador categórico). El autor utiliza dos bases de datos de sonido ambiente (ESC-5-10 y UrbanSound8K), equilibradas y etiquetadas manualmente, y segmenta sus grabaciones en tramas más pequeñas, cuyo mel-espectrograma escalado logarítmicamente alimenta a la red. Los resultados, con 10 clases diferentes de sonido son de una precisión del 73,7%, bastante aceptable teniendo en cuenta que los datos disponibles no son muy abundantes. Este sistema ha tenido un desarrollo posterior por Piczak[8] que presenta su sistema en la competición de detección y clasificación de escenas y eventos acústicos DCASE [9]. Piczak introduce variaciones en la arquitectura original, básicamente incrementando hasta 200 el número de filtros mel de los parámetros de entrada, y obtiene resultados algo mejores, pero no está claro si los resultados presentan cierto sobreentrenamiento.

5.2.3. Salamon and Bello, 2015

Piczak compara su sistema con un clasificador no supervisado basado en un *clustering* o agrupamiento mediante k-medias esféricas y un algoritmo de clasificación de bosques aleatorios (*random forests*), propuesto por Salamon and Bello,[10], que poco después publican también su propio sistema basado en redes neuronales convolucionales[11], en este caso en una tarea supervisada utilizando la base de datos UrbanSound8K. También investigan técnicas de aumento

de datos para lograr generar más datos de entrada para mejorar el entrenamiento de las redes. Con su sistema logran una precisión cercana al 80%.

5.2.4. Cakir, 2017

Çakir también ha proseguido su investigación en la detección de eventos sonoros polifónicos[12], proponiendo una arquitectura de redes neuronales convolucionales junto con capas recursivas (RNN), que dotan de "memoria" a la red. Su red CRNN (*convolutional recursive neural network*) se evalúa con 4 conjuntos de datos mejorando los resultados de redes CNN, RNN y otros clasificadores, aunque su sistema depende fuertemente de la cantidad de datos anotados disponibles.

Por otro lado, el problema de la escasez de datos anotados es omnipresente, y por eso se buscan también métodos no supervisados[13]. En el ámbito de la clasificación multietiqueta, proponen utilizar lo que llaman *de-noising autoencoders*, en su versión simétrica y asimétrica para obtener parámetros basados en datos à partir de parámetros MFB.

5.3. Análisis *Challenge* DCASE

La organización de DCASE ofrece una gran variedad de concursos en los que poder tomar parte, cada uno con una diferentes caracteristicas. Los *Challenge* se ordenan del 1 al 6 divididos en diferentes tareas o *task*, como puede ser la clasificación de escenas acústicas o el automatizado de subtítulos para audio.

Atendiendo a la arquitectura del modelo de AHOLAB y a nuestros conocimientos previos, se ha decidido tomar parte del *task* número 1. Esta tarea es la que más se ajusta a la arquitectura de AHOLAB y a el objetivo principal de este proyecto, aplicar el modelo convolucional de AHOLAB a la clasificación de escenas acústicas.

La tarea número uno se divide en otras dos, cada una con una serie de variantes que la diferencia de la otra. Para este proyecto se decidió que en un principio se iba tomar parte en los dos, teniendo la posibilidad de trabajar más en uno que en otro o pudiendo abandonar alguno en mitad de la investigación. A continuación, se va aclarar en que consiste cada uno de ellos.

5.3.1. Subtask 1A: Clasificación de escenas acústicas con múltiples dispositivos

Se basa en la clasificación de sonidos ambientales provenientes de los datos recogidos por múltiples dispositivos (reales y simulados).

Esta subtarea se refiere al problema básico de la clasificación de escenas acústicas, en el que se requiere clasificar una grabación de audio de prueba en una de las diez clases conocidas como escenas acústicas. Esta tarea apunta a las propiedades de generalización de los sistemas en varios dispositivos diferentes, y utilizará datos de audio grabados y simulados en una variedad de dispositivos.

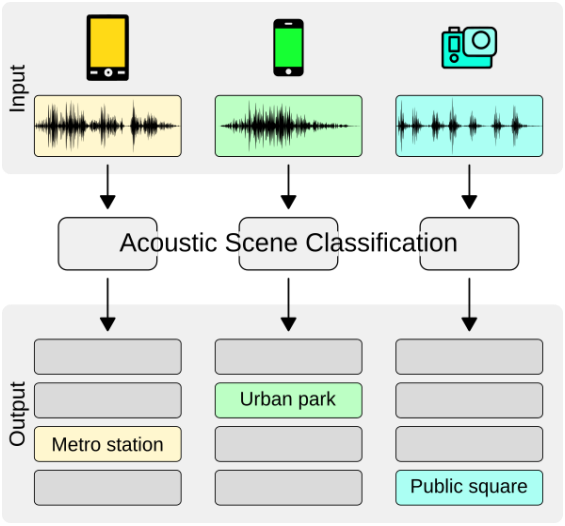

Ilustración 3: Task1a

5.3.1.1. Base de datos

El conjunto de datos contiene grabaciones de 12 ciudades europeas en 10 escenas acústicas diferentes utilizando 4 dispositivos diferentes. Además, se crearon datos sintéticos para 11 dispositivos móviles basados en las grabaciones originales. Las escenas acústicas y las ciudades son las siguientes

Escenas	Lugares
Aeropuerto	Ámsterdam
Centro comercial cubierto	Barcelona
Calle peatonal	Helsinki
Plaza pública	Lisboa
Estación de metro	Londres
Calle con nivel medio de tráfico	Lyon
Viajar en tranvía	Madrid

Viajando en autobús	Milán
Viajando en un metro subterráneo	Praga
Parque urbano	Paris
	Estocolmo
	Viena

Tabla 1:Base de datos task1a

5.3.1.1.1. Datos de desarrollo

El conjunto de desarrollo contiene datos de 10 ciudades y 9 dispositivos: 3 dispositivos reales (A, B, C) y 6 dispositivos simulados (S1-S6). La cantidad total de audio en el conjunto de desarrollo es de 64 horas. El conjunto de datos se proporciona con una división de entrenamiento y prueba en la que se incluye el 70% de los datos para cada dispositivo para el entrenamiento, y el 30% para la prueba.

El identificador o nombre de cada audio sigue un criterio, que luego a la hora de evaluar debe ser el mismo. El criterio es el siguiente:

[scene label]-[city]-[location id]-[segment id]-[device id].wav

5.3.2. Subtask1B: clasificación de escenas acústicas de baja complejidad

Esta subtarea se refiere a la clasificación de audio en tres clases principales: interior, exterior y transporte. La tarea apunta a soluciones de baja complejidad para el problema de clasificación en términos del tamaño del modelo, y utiliza audio grabado con un solo dispositivo.

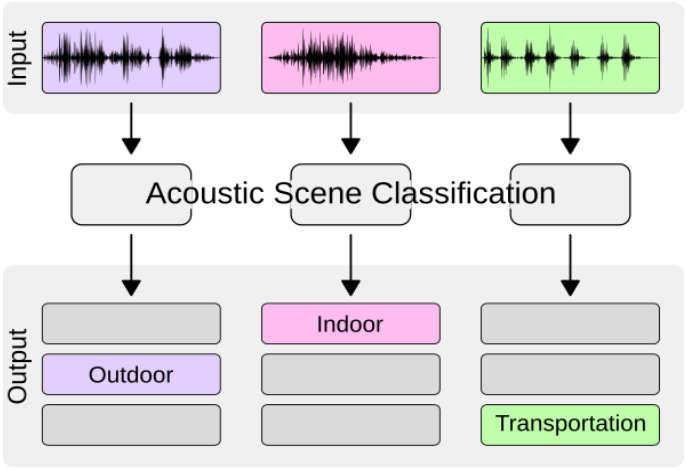

Ilustración 4: Task1b

5.3.2.1. Base de datos

El conjunto de datos contiene grabaciones de 12 ciudades europeas en 10 escenas acústicas diferentes. Las 10 escenas acústicas se agrupan en tres clases principales de la siguiente manera:

Indoor	Outdoor	Transportation
Aeropuerto	Calle peatonal	Viajar en autobús
Centro comercial cubierto	Plaza pública	Viajar en tranvía
Estación de metro	Calle con tráfico medio	Viajar en metro
	Parque urbano	

Tabla 2: Base de datos task1b

Este conjunto de datos contiene datos grabados con un solo dispositivo. El audio se proporciona en formato binaural de 48 kHz y 24 bits.

5.3.2.2. Datos de desarrollo

El conjunto de desarrollo contiene datos de 10 ciudades. La cantidad total de audio en el conjunto de desarrollo es de 40 horas. El identificador de ubicación se puede encontrar en el archivo de metadatos proporcionado en el conjunto de datos o en los nombres de los archivos de audio:

[scene label]-[city]-[location id]-[segment id]-[device id].wav

En este mismo formato se deben de enviar los resultados en caso de tomar parte en este *Challenge*.

5.3.2.3. Requisitos del sistema

Para este segundo *Challenge* del *task* 1 DCASE obliga a cumplir una serie de requisitos mínimos para poder participar en el concurso. La complejidad del clasificador para esta configuración está limitada al tamaño de 500 KB para los parámetros distintos de cero. Esto se traduce en parámetros de 128K cuando se usa float32, que a menudo es el tipo de datos predeterminado. Para aplicar el límite estrictamente al tamaño del clasificador, el recuento de parámetros excluirá la primera capa activa de la red si esta capa es una capa de extracción de características. Las capas no utilizadas en la etapa de clasificación, como las capas de normalización por lotes, también se omiten del cálculo del tamaño del modelo. Si el sistema usa incrustaciones (por ejemplo, VGGish, OpenL3 o EdgeL3), la red utilizada para generar las incrustaciones cuenta en el número de parámetros.

Se debe proporcionar información completa sobre el tamaño del modelo en el informe técnico que debe enviar junto a los resultados para tomar parte en el concurso.

5.3.3. Reglas

Para todos los *Challenge* de DCASE hay unas normas comunes que se deben cumplir para poder participar en él.

- Se permite el uso de datos externos.
- Se permite la manipulación de los datos de capacitación y desarrollo proporcionados (por ejemplo, mezclando datos muestreados de un pdf o utilizando técnicas como el cambio de tono o el estiramiento de tiempo).
- Los participantes no pueden emitir juicios subjetivos de los datos de evaluación ni anotarlos.
- La decisión de clasificación debe hacerse de forma independiente para cada muestra de prueba.

5.3.4. *Baseline* o sistema de referencia

La *Baseline* es un sistema que ofrece DCASE por cada subtarea publicada, en la que se da una posible solución al problema. La *Baseline* proporciona un enfoque de vanguardia para la

clasificación en cada subtarea. El sistema de línea de base se basa en "dcase_util" (una caja de herramientas) y tiene toda la funcionalidad necesaria para el manejo de conjunto de datos, extracción de características acústicas, almacenamiento y acceso, capacitación y almacenamiento de modelos acústicos y evaluación. La estructura modular del sistema permite a los participantes modificar el sistema según sus necesidades.

A continuación, se van a especificar los resultados obtenidos de la *Baseline* para los datos de validación:

5.3.4.1. Resultados task1a

ESCENA	PRECISION	PERDIDAS
Airport	45%	1,615
Bus	62,9%	0,964
Metro	53,5%	1,281
Metro station	53%	1,298
Park	71,3%	1,022
Public square	44,9%	1,633
Shopping mall	48,3%	1,482
Street, pedestrian	29,8%	2,277
Street, traffic	79,9%	0,731
Tram	52,2%	1,350
Average	**54,1%**	**1,365**

Tabla 3: Resultados Baseline task1a

5.3.4.2. Resultados task1b

ESCENA	PRECISION	PERDIDAS
Indoor	82%	1,680
Outdoor	8,5%	0,365
Transportation	91,5%	0,282
Average	**87,3%**	**1,365**

Tabla 4: Resultados Baseline task1

6. Análisis de alternativas

En este capítulo del documento se van a seleccionar diferentes alternativas para los recursos necesarios para el desarrollo del proyecto. Por cada recurso, se realizará un análisis exhaustivo del mismo analizando las necesidades, y luego se enumerarán y expondrán las diferentes alternativas como posibles soluciones. Por último, siguiendo unos criterios, también explicados previamente, se decidirá cuál es la mejor alternativa para cubrir la necesidad planteada al inicio. Atendiendo a las necesidades del proyecto, se realizarán los siguientes análisis: en qué plataforma es más adecuado desarrollar y ejecutar la red neuronal, con qué datos completar la base de datos y cómo implementar la arquitectura de AHOLAB para la clasificación de sonidos ambientales

6.1. Plataforma de desarrollo y ejecución de la red neuronal

Anteriormente se ha comentado que en este trabajo se participará en el *Challenge* DCASE, y para ello se necesitará instalar su sistema. El nuevo entorno de trabajo que se debe crear es muy pesado ya que en él se deben descargar las bases de datos de DCASE, los archivos de la *Baseline* de DCASE y software específico para poder trabajar con redes neuronales como Python, TensorFlow o Keras. Además, estos programas deben ser instalados, o en caso de que ya estén instalados deberán ser actualizados en una concreta versión. El peso estimado del nuevo entorno oscila entre 250GB y 300GB. Para esto, como solución, se crearán nuevos entornos Conda.

Además, el entrenamiento de un modelo convolucional necesita de una gran capacidad computacional. Esos cálculos, al ser muy pesados, se realizan mediante tarjetas gráficas, por lo tanto, la maquina donde quede instalada la *Baseline* necesita disponer de una buena GPU.

Esta situación causa la problemática de donde es más correcto instalar ejecutar la red neuronal, y ante esto se plantean dos soluciones posibles: la primera es instalarla en el ordenador personal y la segunda en el servidor de AHOLAB.

- **Ordenador personal:** un Huawei Matebook D, en el que se podría trabajar desde casa accediendo a la red del hogar que dispone de 600MB de fibra simétrica. En lo que a los aspectos técnicos respecta, contiene una memoria HDD de 1TB y otra SSD de un 1TB. La RAM es de 6GB y tiene una tarjeta gráfica del modelo NVIDIA GeForce MX150. Incluso pudiendo instalar la *Baseline,* no está claro que la tarjeta gráfica pueda entrenar el modelo.
  En la siguiente tabla se pueden observar de una forma ordenada los aspectos técnicos más relevantes:

Nombre	Memoria	RAM	Tarjeta gráfica	Distribución	Procesador
Huawei Matebook D	1 x HD 1TB, 1 x SSD 1TB GB	1 x 8GB RAM	1 x NVIDIA GeForce 6021MB	Linux o Windows	1 x Intel Core i7

Tabla 5: Caracteristicas ordenador personal

- **Servidor AHOLAB:** el servidor está instalado en el Centro de Datos de Leioa por lo que está conectado a la red de Internet de la universidad, y para acceder a él desde fuera es necesario abrir una conexión SSH. La distribución usada por el servidor es GNU/Linux en su versión Ubuntu. En cuanto a los datos técnicos, tiene una memoria HDD de 8TB, 500GB de SSD y dispone de RAM de 128GB de tipo DDR4. Las tarjetas gráficas, uno de los aspectos más importantes, son 4 Nvidia TITAN RTX. El servidor es de uso compartido, por lo que al mismo tiempo hay más usuarios trabajando en él. Mediante la siguiente tabla se resume los datos técnicos:

Nombre	Memoria	RAM	Tarjeta gráfica	Distribución	Procesador
T10G-SP Dual Xeon Scalable HPC 10 GPU	1 x HD 8TB, 1 x SSD 500 GB	4 x 32GB DDR4	4 x NVIDIA Titan RTX 24GB	1 x Distribución GNU/Linux Fedora, Ubuntu	1 x Intel Xeon Silver 2,1GHz

Tabla 6: Caracteristicas servidor AHOLAB

Una vez expuestas las dos alternativas posibles, se van a mencionar los criterios de elección con el valor de importancia de cada uno representado en porcentaje:

- **Memoria (20%):** La memoria libre que disponga la máquina que se va utilizar es de vital importancia ya que como se ha mencionado anteriormente el sistema de DCASE es muy pesado y se necesita una gran capacidad de memoria. Además, durante los entrenamientos de los modelos convolucionales se crean una serie da archivos temporales que necesitan de más memoria.
- **Capacidad de GPU (60%):** Los modelos convolucionales necesitan ser entrenados, y para cada entrenamiento se necesita una capacidad computacional muy elevada para realizar los cálculos necesarios. Para ello se utilizan las tarjetas gráficas, y de esta forma se libera de carga de trabajo a la CPU. Es importante tener un GPU potente ya que esto puede hacer que disminuya considerablemente el tiempo de entrenamiento de los modelos convolucionales.

- **Conexión a internet (10%):** A la hora de descargar datos de internet o actualizar algún programa es necesario una buena conexión a internet, sin fallos y con una velocidad bastante alta ya que son necesarios descargar una cantidad muy elevada de datos.
- **Comodidad (10%):** La comodidad del usuario también es un factor a valorar, aunque en menor medida que los dos primeros. Si el usuario se encuentra a gusto en su puesto de trabajo se aumentará su rendimiento. Esto sucede cuando el trabajador está más familiarizado con un sistema que con otro.

Atendiendo a estos criterios se van a calificar las dos alternativas planteadas con una nota del 0 al 10, y la alternativa que consiga la nota más alta será la solución para el problema. Para conseguir la nota final se ponderarán por su porcentaje cada una de las notas parciales.

	Memoria (20%)	GPU (60%)	Conexión a internet (10%)	Comodidad (10%)	Resultado final
Ordenador personal	5	7	9	9	7
Servidor AHOLAB	9	9	8	7	8.7

Tabla 7: Análisis instalación sistema DCASE

Como se puede ver en la Tabla 7 , una vez finalizado el análisis de donde instalar la *Baseline* la mejor opción es instalarla en servidor de AHOLAB ya que dispone de una capacidad de memoria mucho mayor y de cuatro tarjetas gráficas.

6.2. Base de datos a utilizar

En esta ocasión se plantea la problemática de como completar la base de datos con los audios suficientes. El entrenamiento de un modelo convolucional requiere de una robusta base de datos para alimentarla. Debe de estar formada por miles de audios en un mismo formato (lo más común es .wav), con una misma duración y con una misma frecuencia de muestreo. Además, el etiquetado de los archivos debe seguir un patrón para que el sistema pueda interpretarlos y de esta forma entrenar y aprender.

Como solución a este problema se han encontrado las siguientes dos posibles soluciones:

- **Base de datos de DCASE únicamente:** DCASE ofrece una base de datos ya creada para el concurso. En esta alternativa se plantea utilizar exclusivamente esa base de datos

para entrenar el sistema. Esto facilitaría la creación de la base de datos, pero puede suceder que se quede corta para entrenar lo necesario el modelo.

- **Utilizar otras bases de datos ya creadas:** Mediante esta alternativa se plantea la posibilidad de sumarle nuevas bases de datos a la ya ofrecida por DCASE. En la web hay disponibles diversas bases de datos ya creadas, como puede ser la mencionada anteriormente *Urban Sound*. Al fusionar más de una base de datos, se conseguirá un mejor entrenamiento para el sistema, pero esto ocasionará una carga computacional mayor. Además, al juntar las dos o más bases de datos se deberá resolver el problema de etiquetado y hay que preparar los scripts para que puedan acceder a la nueva base de datos y puedan extraer los *features*, esto ocasiona un trabajo de programación mayor.

Una vez explicadas las soluciones posibles, se va proceder a detallar cuales van a ser los criterios de evaluación para escoger la mejor solución para el problema. Cada criterio tendrá un peso diferente a la hora de escoger la mejor solución.

- **Cantidad de material (40%):** mediante este parámetro se mide la cantidad de material del que se dispone para entrenar el modelo.
- **Sencillez (40%):** este parámetro mide la sencillez de utilización de la base datos. Aquí se valoran tanto el tiempo requerido como la complejidad a la hora de montar la base de datos.
- **Manejo (20%):** Este parámetro mide como es el manejo de los audios una vez creada la base de datos, por ejemplo, la gestión del etiquetado o hacer una modificación en ella.

Atendiendo a estos criterios se van a calificar las dos alternativas planteadas con una nota del 0 al 10, y la alternativa que consiga la nota más alta será la solución para el problema. Para conseguir la nota final se ponderarán por su porcentaje cada una de las notas parciales.

	Cantidad de material (40%)	Sencillez (40%)	Manejo (20%)	Resultado final
Base de datos de DCASE únicamente	7	10	7	8,2
Utilizar otras bases de datos ya creadas:	10	4	8	7,2

Tabla 8: Análisis creación base de datos

Atendiendo a la Tabla 8, la mejor opción es descargar la base de datos ya creada por DCASE ya que es el método más sencillo y rápido. Además, aunque tenga un número de muestras limitado, con ella es más que suficiente para poder entrenar el modelo convolucional.

6.3. Método de implementación de la arquitectura

Mediante este apartado se va resolver el problema de como reproducir la arquitectura de AHOLAB. El objetivo es crear un modelo convolucional similar y para ello se plantea dos alternativas para llevarlo a cabo:

- **Usar el software provisto por el modelo de referencia de DCASE:** la organización del concurso ofrece un sistema ya creado. El sistema está construido de tal forma que permite cambiar la arquitectura de la red neuronal en cierta medida, y de esa manera se nos permite implementar el modelo de AHOLAB usando los scripts proporcionados por DCASE.
- **Implementación completa de la red neuronal:** la implementación se haría directamente sobre Python y TensorFlow. Se debe crear todo el software necesario para extraer los parámetros de base de datos según su estructura. Además, se debe implementar la red en TensorFlow y todos los procesos de entrenamiento y evaluación.

A continuación, se van a detallar los criterios con los que se tomará la decisión, dándoles a cada uno de ellos una relevancia diferente:

- **Rapidez (30%):** este parámetro evalúa la rapidez con la que se puede hacer un cambio en la arquitectura.
- **Sencillez (50%):** mediante este parámetro se tiene en cuenta la complejidad de implementar la nueva arquitectura.
- **Realista (20%):** medirá hasta qué punto la nueva arquitectura se puede parecer a la inicial planteada por AHOLAB.

Atendiendo a estos criterios se van a calificar las tres alternativas planteadas con una nota del 0 al 10, y la alternativa que consiga la nota más alta será la solución para el problema. Para conseguir la nota final se ponderarán por su porcentaje cada una de las notas parciales.

	Rapidez (30%)	Sencillez (50%)	Realista (20%)	Resultado final
Usar el software provisto por el modelo de referencia de DCASE	9	8	7	8,1
Implementación completa de la red neuronal	7	6	9	6,9

Tabla 9: Análisis scripts

En definitiva, la mejor alternativa de las dos es usar los scripts de DCASE, ya que tienen una sencilla instalación y no requiere hacer ningún retoque en la arquitectura para poder ejecutarla. Además, a la hora de hacer esos cambios se pueden realizar de forma rápida simplemente cambiando el valor del hiperparámetro.

7. Análisis de riesgos

Durante el transcurso del proyecto pueden surgir una serie de problemas que se pueden prever de antemano. Mediante este apartado se van a estudiar los riesgos que pueden darse durante la elaboración de este trabajo. En primer lugar, se detectarán los riesgos potenciales del trabajo y una vez identificados, se analizarán uno a uno detallando la probabilidad de que ocurra y, en caso de ocurrir, su impacto en el proyecto. Para finalizar, se preparará un plan de actuación en contra cada uno de esos riesgos.

7.1. Identificación de riesgos

Para empezar, se van a identificar los posibles riesgos pueden surgir explicando en qué situación pueden darse:

- **Averías en el servidor con la Escuela cerrada:** el servidor de AHOLAB se encuentra situado dentro del Centro de Proceso de Datos de Leioa por lo que en caso de ocurrir un fallo mecánico en él o un fallo de software que requiera el reinicio del servidor se necesita por acceder al centro. Por lo que si la escuela se encuentra cerrada por una situación adversa como puede ser una pandemia sería imposible reparar el problema y el servidor quedaría deshabilitado.
- **Ocupación del servidor:** 4 son las GPU de las que dispone el servidor. Pero en él hay muchos usuarios que tienen la necesidad de usar las GPU día a día y lanzan procesos de una duración muy larga. Es posible que existan situaciones en las que nos conectemos al servidor y no tengamos donde lanzar nuestros procesos, y de esta forma se retrasa el trabajo.
- **Publicación del material del *Challenge* DCASE:** DCASE publica los *Challenge* y las bases dos meses y medio antes de que termine el plazo del concurso. Durante esos dos meses está el periodo de exámenes de la universidad, en los que tanto el director del trabajo como el alumno que lo realiza tienen que tomar parte. Por lo que en las fechas en las que se debería invertir más tiempo en el desarrollo del proyecto, es cuando de menos tiempo se dispone.

7.2. Análisis de riesgos

Una vez identificados todos los riesgos, se va proceder a examinar cada uno de ellos calculando la probabilidad que hay de que sucedan, y en caso de suceder que, impacto tendría en el proyecto:

- **Averías en el servidor con la Escuela cerrada:** la probabilidad de que ocurra es muy escasa ya que tendría que suceder una situación muy extraordinaria que impidiese entrar a la Escuela, como puede ser una pandemia o un incendio. Pero en caso de que sucediese, las consecuencias serían muy graves ya que en él se guardan todos los datos y es la herramienta de trabajo.
- **Ocupación del servidor:** este problema puede surgir fácilmente, y no únicamente una vez, sino que en repetidas ocasiones. Además, su repercusión en el proyecto es bastante elevada, y en caso de repetirse de continuo podría llegar a ser un problema muy grave.
- **Publicación del material del *Challenge* DCASE:** este problema aparecerá casi seguro durante el proyecto, ya que no depende de nosotros las fechas de publicación del concurso. La repercusión que tiene en el trabajo es bastante alta ya que influye en posibles retrasos o no conseguir los resultados esperados

	IMPACTO				
PROBABILIDAD	Muy baja (0,05)	Baja (0,1)	Moderada (0,2)	Alta (0,4)	Muy alta(0,8)
Rara (0,1)	0,005	0,01	0,02	0,04	Averías en el servidor con la Escuela cerrada (0,08)
Complicada (0,3)	0,015	(0,03)	0,06	0,12	0,24
Posible (0,5)	0,025	0,05	0,1	0,2	0,4
Puede suceder (0,7)	0,035	0,07	0,14	Ocupación del servidor(0,28)	0,56
Casi seguro (0,9)	0,045	0,09	0,18	0,36	Publicación del material del Challenge DCASE(0,72)

Tabla 10: Análisis de riesgos

7.3. Respuesta ante los riesgos

Por último, y una vez analizado los riesgos en la Tabla 10, se va proceder a estudiar cómo hacer frente a esos problemas. Siguiendo unos criterios se van a clasificar los riesgos y a través de esa clasificación se preparará una respuesta diferente para cada situación. De la siguiente manera se han clasificado los riesgos de este proyecto:

- **Aceptar el riesgo:** las consecuencias del riesgo se aceptan; y en caso de suceder se decidirá como hacerles frente. No se cambiará el plan inicial del proyecto.
    - Ocupación del servidor: En caso de conectarse al servidor, y no poder lanzar un proceso porque están todas las GPU ocupadas, se buscará otro que hacer, como puede ser hacer un análisis de los resultados, hasta que se libere alguna.
- **Controlar el riesgo:** se acepta el riesgo, pero hay que controlarlo de cerca. En caso de suceder hay que tener una alternativa preparada.
    - Averías en el servidor con la Escuela cerrada: cuando se detecte que puede suceder este problema porque la Escuela está cerrada por algún motivo extraordinario, se puede ir instalando la *Baseline* en el ordenador local. Así, en caso de errores en el servidor se tendrá una alternativa con la que trabajar.
- **Evitar el riesgo:** Consiste en modificar el plan del proyecto para eliminar el riesgo o proteger un objetivo del proyecto o asegurar que no se verá afectado.
    - Publicación del material del Challenge DCASE: Para no tener que concentrar todo el trabajo en 2 meses y medio, antes se trabajará con las tareas del *Challlenge* de 2019. De esta forma, cuando se publiquen las bases de 2020, se tendrá parte del trabajo adelantado.

8. Diseño

Como se ha mencionado en los apartados anteriores del documento, el modelo convolucional desarrollado por AHOLAB es la base de este proyecto y por lo tanto se seguirá su diseño. En esta sección, se analizará como está estructurada la arquitectura, y seguido se estudiará un script ofrecido por DCASE para poder hacer el ajuste de los hiperparametros.

8.1. Redes convolucionales

Tal y como se ha dicho anteriormente el modelo fue desarrollado para clasificar los eventos sonoros provenientes de un automóvil, por lo que esos sonidos se podrían incluir dentro en los ruidos BSR (*Buzz, squeak and rattle*). En su momento, AHOLAB no encontró grandes trabajos en los que se relacionasen las redes neuronales y ruidos BSR, y por lo tanto, tuvo que apostar por una arquitectura nueva. Finalmente, se decantaron por las redes neuronales convolucionales (CNN) ya que han demostrado un gran rendimiento en el procesamiento de señales de audio y una gran robustez frente al ruido, como demostraron Piczak y Salomon and Bello en cuyos trabajos se basaron en crear un nuevo modelo.

Las arquitecturas de CNNs más habituales están formadas por diferentes capas: una capa de entrada, un grupo de capas convolucionales y capas *pooling* (capas de reducción de muestreo) en diferentes combinaciones, una serie reducida de capas ocultas de neuronas completamente conectadas y una capa de salida.

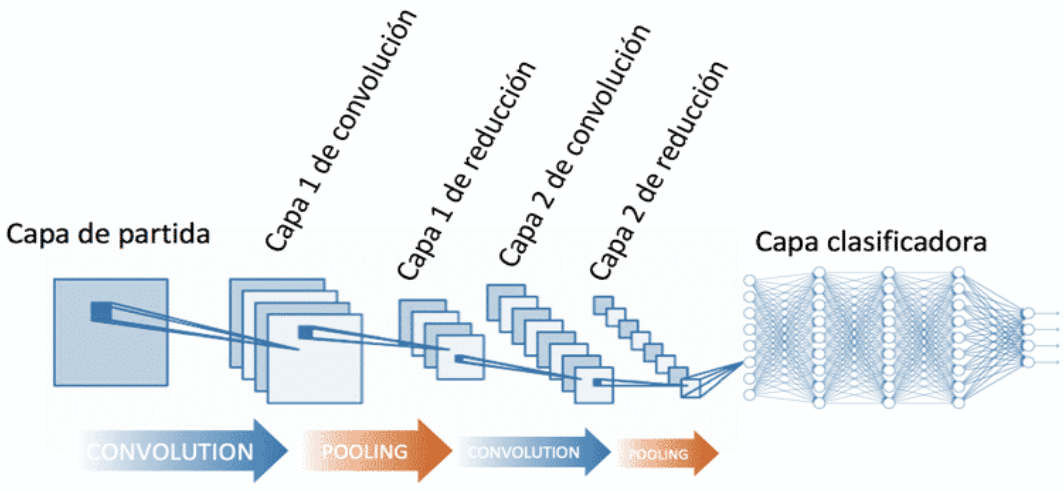

Ilustración 5: Arquitectura red neuronal convolucional

Las neuronas artificiales o unidades de una capa convolucional no están conectadas a todos los datos de la entrada a la vez, sino que solo procesa una pequeña parte de la entrada bidimensional, que se denomina campo receptivo de la unidad convolucional. Los pesos que esta unidad aplica a cada uno de sus valores del campo receptivo forman un filtro que se aplica a toda la entrada bidimensional, como podría ser el espectrograma de una señal. De esta forma, y tras aplicar el filtro a toda la entrada bidimensional, se consigue un mapa igual que indica cuanto de parecida es la entrada en esa zona con el patrón buscado por el filtro: se conoce como *feature map.* La red convolucional no aplica un único filtro, sino que una gran cantidad de ellos.

Las capas convolucionales suelen llevar asociadas capas de reducción de muestreo, también llamadas capas *pooling.* Estas capas toman conjuntos de adyacentes del *feature map* producido por la capa convolucional y lo fusionan en una única salida según diferentes criterios: media, valor máximo, votación, etc.

Esta es la secuencia habitual de una red convolucional:

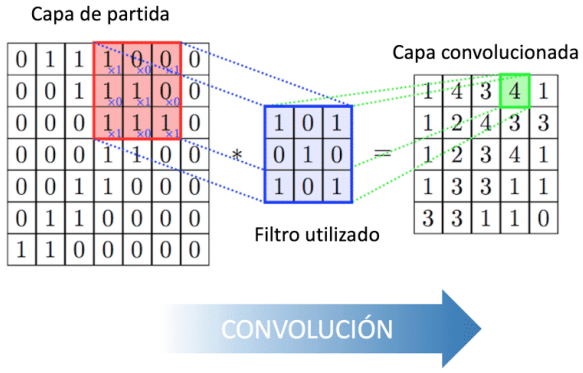

Ilustración 7: Convolución

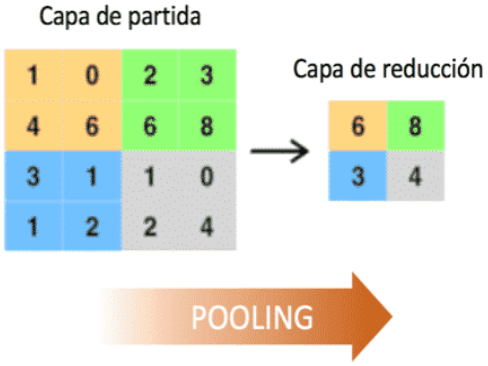

Ilustración 6: Pooling

AHOLAB para su arquitectura propuso algunos cambios respecto a la arquitectura habitual de las CNN.

8.2. Arquitectura original deAHOLAB

Uno de los cambios más significativos es la dimensionalidad de los filtros utilizados, ya que normalmente las unidades convolucionales procesan entradas bidimensionales y producen salidas, mapas de características, también bidimensionales.

Sin embargo, en el caso de las señales de audio los patrones en frecuencia no son tan repetibles como en el tiempo, por lo que no tiene sentido deslizar el filtro o *kernel* en el eje de

la frecuencia, y es más lógico deslizarlo únicamente en el eje del tiempo. De esta forma los filtros ocupan todo el ancho espectral y se mueven sólo en el eje temporal, así logrando un mapa de características unidimensional.

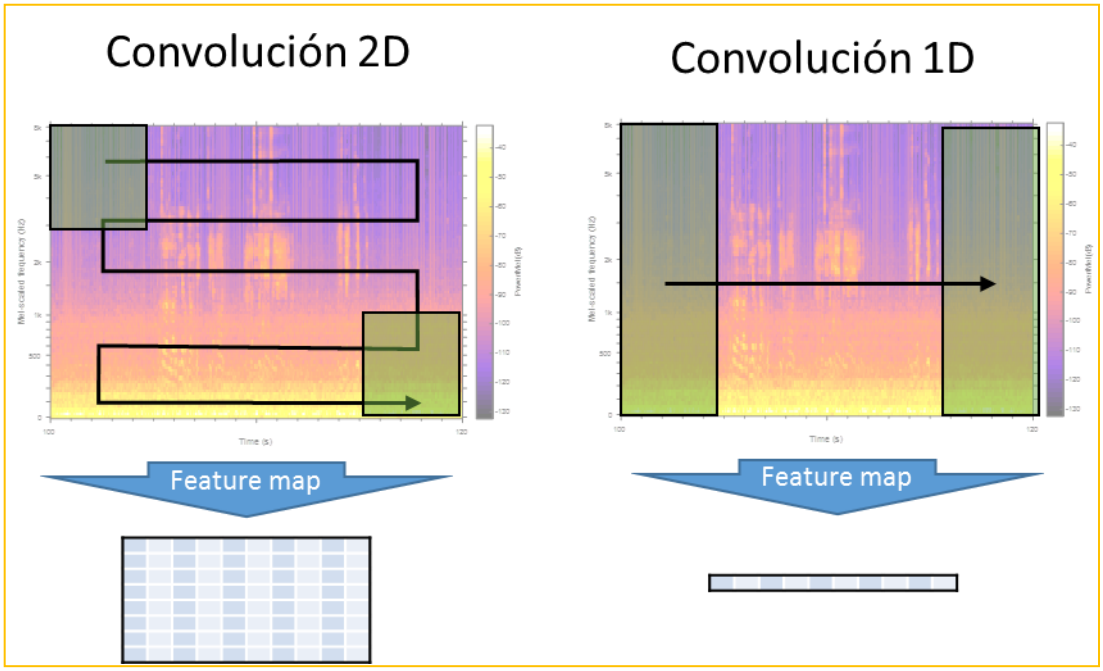

Ilustración 8: Convolución1D vs convolución2D

La arquitectura de AHOLAB consta de 2 capas convolucionales con sus correspondientes capas de reducción, dos capas completamente conectadas y una capa de salida. En los siguientes puntos se detallan lo más importante de la arquitectura:

- **1ª capa convolucional:** consta de 80 filtros, sus dimensiones son (nMx6), donde n es el número de canales n-aurales, M el número de valores en el eje de frecuencia de la entrada y 6 corresponde con el eje temporal. En todos los casos el desplazamiento de los filtros es de 1x1 y no se aplica el filtrado en los bordes (*padding*).
- **1ª capa de reducción de muestro o *pooling*:** se basa en el seleccionar el valor máximo (*max-pool*) de los valores que entran en un rectángulo de 1 x 3. Así la salida tras esta capa es unidimensional, en el eje temporal. El rectángulo de *pooling* se desplaza a saltos de 3 columnas, es decir, en el eje temporal, un *stride* de (1x3).
- **2ª capa convolucional:** como la primera, consta de 80 filtros de dimensiones (1x3) y desplazamiento (1x1).

- **2ª capa de reducción de muestro o *pooling*:** hace max-pooling de nuevo, esta vez con ventanas de (1x3) y desplazamiento de (1x3). Tras esta capa, la salida se reordena para pasar de su forma matricial a una lineal que permita alimentar las capas completamente conectadas que ese encuentran justo después.
- **Dos capas de neuronas completamente conectadas:** tiene 2000 neuronas cada capa. A estas capas se les aplica la técnica de *dropout* o descarte aleatorio con un 50% de probabilidad en la fase de entrenamiento.
- **Capa de salida:** es una capa con tantas neuronas como clases se estén considerando en el clasificador. Se aplica la función *Softmax* para obtener la probabilidad de cada clase a partir de los *scores* no normalizados de las neuronas de salida.

En la Ilustración 9 se puede observar un esquema general de la arquitectura descrita, aunque en este caso todas las entradas son monoaurales:

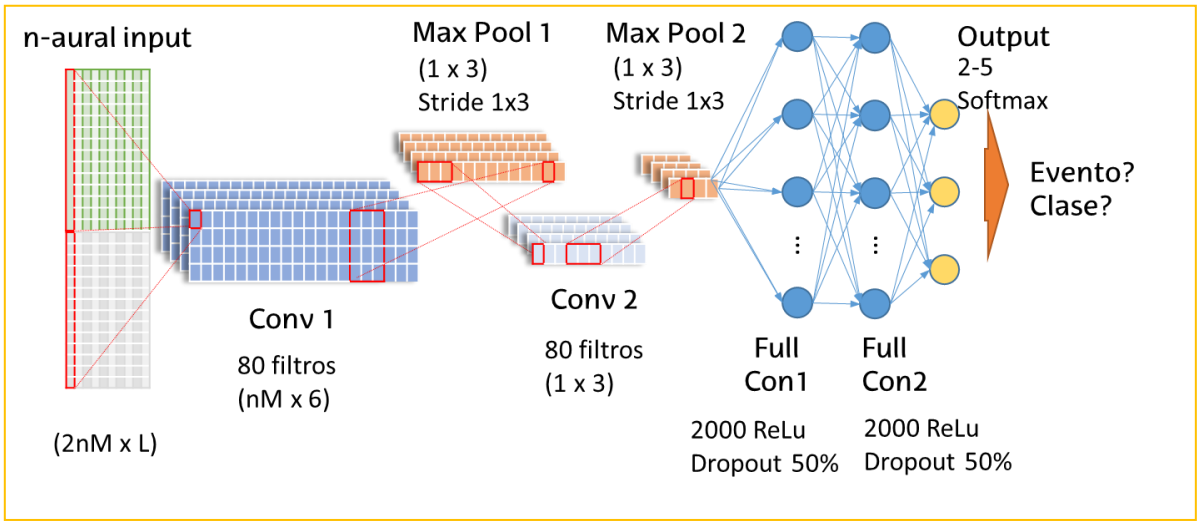

Ilustración 9: Arquitectura AHOLAB

En las capas convolucionales, la función de activación se encarga de devolver una salida a partir de un valor de entrada, normalmente el conjunto de valores de salida en un rango determinado como (0,1) o (-1,1). Se buscan funciones que las derivadas sean simples, para minimizar con ello el coste computacional. En el caso de AHOLAB, como función de activación no lineal se usó la función ReLU (*Rectified Linear Units*), que se obtiene según la ecuación:

$$f(x) = max\{0, x\}$$

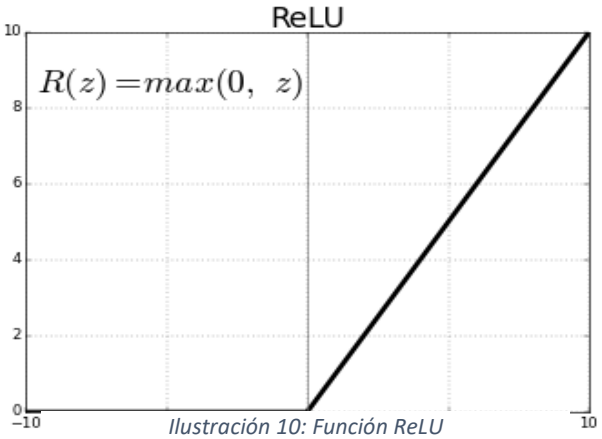

Ilustración 10: Función ReLU

La función ReLU tiene algunas características beneficiosas: no está acotada, se comporta bien con imágenes, buen desempeño en redes convolucionales… Además, presenta algunas ventajas sobre las funciones de activación tradicionales como la tanh (tangente hiperbólica) o sigmoide: cálculos más rápidos, mejor propagación de gradiente, etc.

8.2.1. Datos de entrada de la red

El sistema se alimenta de fragmentos cortos de audio y proporciona una salida para cada uno de ellos. La larga señal de entrada se fragmenta en tramas más cortas y fáciles de analizar. Para fragmentar las grabaciones el modelo de AHOLAB utiliza un enventanado rectangular de una longitud de entre 0,5s y 1,5s. En los casos de los segmentos donde no hay evento el enventanado tiene un solapamiento del 50%. Sin embargo, en los segmentos que corresponden a eventos el solapamiento aumenta al 75% lo que permite obtener más fragmentos.

Hasta el momento, las redes neuronales que procesan audio se suelen alimentar con parámetros extraídos de la señal, aunque últimamente se han propuesto arquitecturas de redes neuronales capaces de ser alimentadas con audio directamente.

Como se ha señalado anteriormente, estos parámetros son espectro-temporales. Pudiendo ser una de las tres siguientes opciones:

- Espectrograma: representación del tiempo contra la frecuencia. Los colores representan la intensidad en cada punto.
- Log-mel-espectrograma: sigue la misma lógica que el espectrograma, pero después de conseguir el espectrograma se usan unos filtros para seleccionar las frecuencias al alcance del oído humano.
- MFCC: son coeficientes para la representación del habla basados en la percepción auditiva humana.

AHOLAB se decantó por usar log-mel ya que es el método que mejor se adapta a los eventos sonoros que se debían clasificar.

En cuanto a las frecuencias de muestreo de la señal, tanto las señales originalmente grabadas en los automóviles por AHOLAB como las ofrecidas por DCASE están grabadas a 48KHz. Por lo que AHOLAB se propuso reducir esa frecuencia de muestro ya que los principales ruidos de interés se encuentran en las frecuencias bajas, pero por otro lado tiene gran peligro reducir el ancho de banda en demasía ya que se puede llegar perder información importante de alta frecuencia. Al final, tras una serie de pruebas, AHOLAB determinó que la frecuencia de muestreo óptima era 12KHz.

En cuanto a la extracción de parámetros, como se acaba de decir, se utilizó log-mel con las siguientes características: un enventanado de 1024 puntos (85 ms a fs=12000Hz), y se realiza este enventanado, el *framerate*, cada 20 ms aprox. (256 muestras a fs=12000Hz), y se selecciona el valor de su amplitud en dBs. Una vez conseguido el espectrograma se aplican unos 60 filtros conocidos como mel, esos filtros son triangulares de banda de paso creciente con la frecuencia, que siguen la escala mel la cual tiene en cuenta las características perceptuales del oído humano. De esta forma, AHOLAB consigue unas imágenes, que este caso son espectrogramas mel, con los que poder alimentar la red convolucional.

Por último y para terminar con el análisis del modelo de AHOLAB, hay que destacar que la base de datos de AHOLAB se divide en tres grandes conjuntos: el conjunto de entrenamiento, con el que se entrena el modelo, el de validación, para validar los ajustes y determinar el óptimo, y de test, que es con el que evalúa el modelo finalmente. El reparto típico suele ser de un 60-80% de los datos para el entrenamiento, 10-20% para validación y otro tanto para test.

8.3. Análisis de los scripts de la tarea 1 e implementación de la arquitectura de AHOLAB

Una vez examinado el modelo convolucional de AHOLAB, se va reproducir uno similar mediante los scripts de DCASE. Para ello, lo único que se debe modificar es un script llamado "task1x.yaml" que se descarga en la instalación de la *Baseline,* en el cual están guardados el valor de los parámetros e hiperparametros de la arquitectura. En los siguientes apartados se va explicar los parámetros más importantes del modelo y los valores que deben tomar para recrear el modelo de AHOLAB. Por último, se expondrán los resultados conseguidos después de entrenar el sistema.

8.3.1. Características acústicas

DCASE ofrece tres métodos posibles para la extracción de parámetros: openL3, edgeL3 y mel. Obviamente, el método que usaremos para la nueva arquitectura será mel, ya que es el que utiliza AHOLAB en modelo original. Entrando en detalle de las características mel, la *Baseline* viene por defecto con un salto de ventana de 0,02 segundos y con un tamaño de 0,04 segundos. Además, tiene 40 filtros mel por defecto. Para realizar la implementación, se cambiará el salto de ventana a 0,25s y el tamaño a 0,05s para que entre dentro de las características del modelo, y en cuanto al número de filtros mel se pondrán 60.

De esta forma, en las siguientes dos imágenes se puede observar los cambios realizados de la *Baseline* a la nueva arquitectura:

```
mel:
  spectrogram_type: magnitude
  hop_length_seconds: 0.025
  win_length_seconds: 0.05
  window_type: hamming_asymmetric
  n_mels: 60
  n_fft: 1024
  fmin: 0
  fmax: 24000
  htk: false
  normalize_mel_bands: false
```

Ilustración 11: parámetros de extracción AHOLAB

Respecto a la frecuencia de muestreo, se dejará la que trae por defecto DCASE que son 48KHz, ya que se adapta mejor que los 12KHz que propone AHOLAB, porque las frecuencias importantes a capturar en los sonidos urbanos son más elevadas que la de los automóviles.

8.3.2. Modelo de aprendizaje

Como método de aprendizaje se puede elegir entre dos en la *Baseline* de DCASE: por un lado, MLP y por otro CNN. MLP (Multi Layer Perceptron) es un método de aprendizaje automático que se ha quedado anticuado respecto a las CNN en el trabajo de clasificación de imágenes. MLP solía aplicarse a la visión artificial con una serie de capas completamente conectadas donde cada perceptrón está conectado con cualquier otro perceptrón, y esto trae consigo desventajas como pueden ser que el número de parámetros totales puede crecer muy alto porque hay redundancia en dimensiones altas o que no tiene en cuenta la información espacial ya que toma vectores aplanados como entradas.

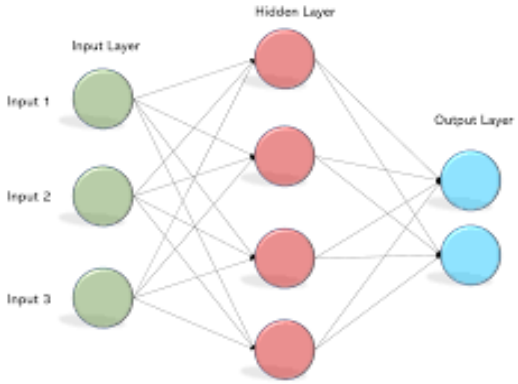

Ilustración 12: Red MLP

Sin duda alguna el método elegido para el aprendizaje son las CNN ya que es el más eficiente y encima el que usa AHOLAB en su arquitectura.

La estructura de la arquitectura de la red neuronal convolucional ofrecida por el concurso es muy parecida a la desarrollada por AHOLAB. Consta de dos capas convolucionales, cada una de ellas seguida por una de reducción o *pooling.* La salida de las capas anteriores se convierte en un vector unidimensional y sirve de entrada de una capa totalmente conectada de 100 neuronas. Por último, dispone de una capa de salida que tiene como activación la función *softmax*. Hay que mencionar que las capas de convolución de la red son bidimensionales.

8.3.3. Capas convolucionales

Las dos capas convolucionales son prácticamente iguales por lo que los cambios a realizar serán los mismos en una que en otra. Por un lado, la función de activación es ReLU por lo que no habrá que cambiarla. Por otro lado, el tamaño del *kernel* o de los filtros es 7, y esto hace que las capas del modelo de DCASE sean bidimensionales, por lo tanto, se debe cambiar el tamaño del kernel. Teniendo en cuenta que los filtros mel aplicados en la extracción de datos han sido 60, se aplicará un tamaño de filtros de 60x6, ya que es lo más similar al modelo de AHOLAB.

En cuanto a los bordes, existe la posibilidad de aplicar o no aplicar *padding*. Aunque la arquitectura de AHOLAB no aplique *padding* o relleno en los bordes, en la réplica si se va utilizar

ya que de esta forma nos permite poder utilizar el mismo tamaño de *kernel* para las dos capas. En lo que respecta al número de filtros aplicados, serán 80 en cada capa.

8.3.4. Capas de reducción o pooling

Las dos capas de reducción, que están situadas a la salida de cada capa convolucional, son capas de reducción por máximo. Es decir, del tamaño de ventana seleccionado para hacer la reducción, se escoge el valor máximo. El tamaño de los filtros será 1x3 y el *stride* también 1x3. El *stride* representa el salto a dar después de aplicar el filtro en una zona.

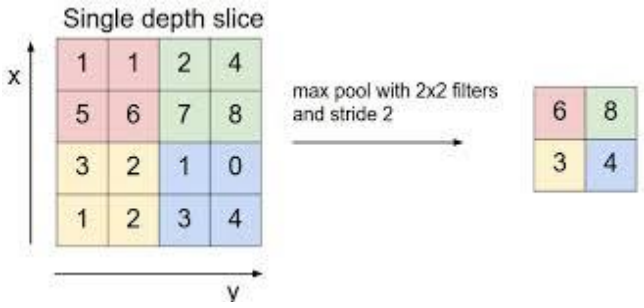

Ilustración 13: Max-Pooling

En la imagen de la derecha se puede observar cómo se realiza la reducción por valor máximo o más conocido en inglés como *Max-Pooling*.

8.3.5. Capas totalmente conectadas

El modelo proporcionado por DCASE utiliza una única capa totalmente conectada con 100 neuronas y *dropout* de 0,3. Para simular la arquitectura de AHOLAB se debe sumar una capa totalmente conectada y además se van a implementar 2000 neuronas por en cada capa. El *dropout* también es un hiperparametro muy importante de esta capa ya que representa el porcentaje de las neuronas que se desconectan en cada iteración de forma aleatoria y de esta manera se les obliga a las neuronas cercanas a no depender tanto de las neuronas desactivadas y se consigue que el sistema no dependa de un número reducido de neuronas que en caso de fallar podría causar grandes errores en los resultados. Además, con un dropout adecuado se puede reducir el *overfitting* que implica que en caso de que en la fase de evaluación se introduzca una clase nueva que no se encontraba en la base de datos de entrenamiento, el sistema sepa cómo actuar

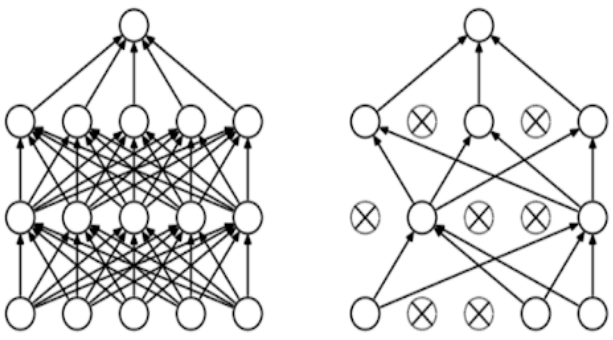

Ilustración 14: Dropout

y como clasificarla, ya que en caso de haber overfitting el sistema no sabría cómo actuar ante una nueva clase y la clasificaría erróneamente.

8.3.6. Capa de salida

Esta última capa, la capa de salida y la encargada de clasificar las muestras en distintas clases, tiene como función de activación *softmax* que conecta contra la capa de salida final la cantidad de neuronas correspondientes al número de clases que estemos clasificando. La función *softmax* se encarga de pasar la probabilidad (entre 0 y 1) a las neuronas de salida.

En cuanto al porcentaje de la base de datos para entrenar el sistema y para validarlos se repartirá de la siguiente manera: el 30% para entrenar y el 70% para validar. Por otro lado, la tasa de aprendizaje o *learning-rate* será de 0,00001 aunque por defecto en DCASE venga 0,001. El mencionado *learning-rate* no debe ser muy grande ya que esto haría que el proceso de aprendizaje fuese demasiado rápido. El número de *epoch* utilizados en el entrenamiento serán 200, de esta forma se realizará el mismo proceso 200 veces.

Resultados

En las siguientes tablas se va mostrar los resultados obtenidos usando la arquitectura de AHOLAB en la clasificación de sonidos urbanos. Se ha aplicado la misma arquitectura tanto para *el task1a* como para el *task1b.* También se van a representar gráficamente las pérdidas ocasionadas en cada *epoch,* de esta forma se va poder determinar si las *epoch* aplicadas son suficientes o son demasiadas.

ESCENA	PRECISIÓN	PÉRDIDAS
Airport	38,4%	4,229
Bus	35%	3,803
Metro	45,1%	4,489
Metro_station	23,6%	7,176
Park	58,6%	6,017
Public_square	20,5%	7,38
Shopping mall	46,1%	3,378

Street pedestrian	39,1%	4,381
Street traffic	68,7%	2,233
Tram	38,7%	4,267
Average	**41,4%**	**4,736**
Resultado Baseline	**54,1%**	**1,365**

Tabla 11: Resultados task1a AHOLAB

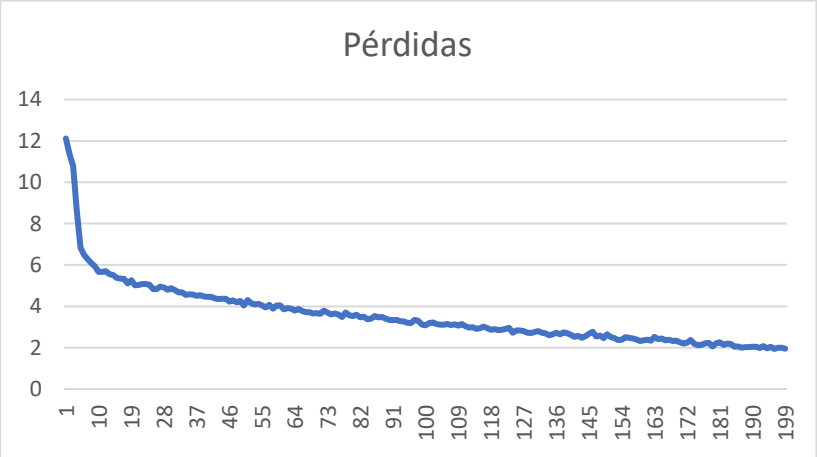

Ilustración 15: Pérdidas task1a arquitectura AHOLAB

SCENE	ACCURACY	LOGLOSS
Indoor	72,7%	5,793
Outdoor	88,7%	2,739
Transportat	89,6%	1,609
Average	**83,7%**	**3,339**
Resultado Baseline	**87,3%**	**1,365**

Tabla 12: Resultados task1b AHOLAB

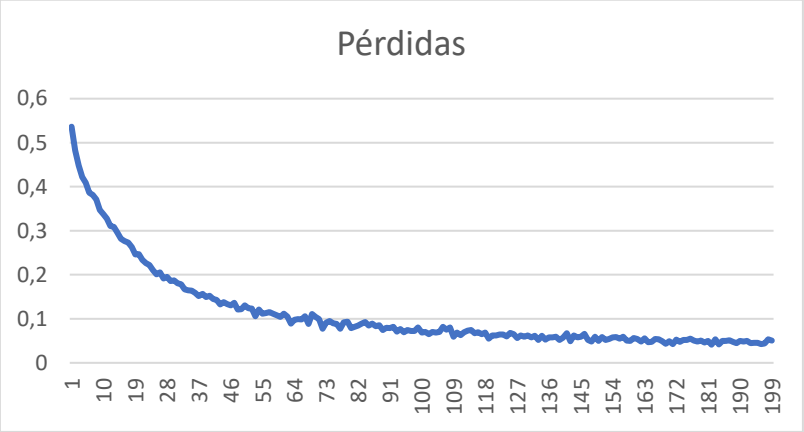

Ilustración 16:Pérdidas task1b arquitectura AHOLAB

Atendiendo a los resultados obtenidos, hay una conclusión que salta a la vista: la arquitectura desarrollada no supera a la *Baseline* en la clasificación de sonidos ambientales. Otra deducción es que la diferencia entre los resultados obtenidos entre el task1a y el task1b son considerables. Por un lado, los resultados logrados para el task1b no se alejan en demasía de los de la *Baseline,* estando únicamente cuatro puntos por debajo, y esto hace pensar que haciendo algún ajuste en los hiperparámtros se puede llegar a obtener resultados satisfactorios. Asimismo, el 83,7% de precisión conseguido ya es un resultado más que aceptable, aunque se debe tener en cuenta que la complejidad del *task1b* es mucho más reducida. Por otro lado, en el *task1a* no se han obtenido los resultados esperados quedando muy lejos de los de la *Baseline*, logrando una pobre precisión del 41,4%.

Después de analizar los dos resultados, se tomó la decisión de centrar los esfuerzos en conseguir avances en el *task1b*, y una vez logrados aplicarlos en el *task1a.* Analizando los resultados también se concluye que no son necesarias 200 *epoch* para entrenar el sistema ya que a partir de las 130-140 las pérdidas son mínimas y constantes.

8.4. Software necesario

Para trabajar con redes neuronales convolucionales es necesario un software específico. Además, para poder ejecutar la *Baseline* ofrecida por DCASE es necesario instalar cada programa en su correcta versión ya que si no podrían surgir problemas de compilación. El trabajo se va realizar en el servidor que tiene instalado AHOLAB en Leioa, pero dicho servidor es utilizado por más usuarios al mismo tiempo, y dispone del software necesario para poder realizar el trabajo. El problema es que la versiones no coinciden con las requeridas por DCASE y no es posible actualizarlas ya que los demás usuarios del servidor están trabajando con dichas versiones de software.

Ante esa situación, se ha decidido escoger como solución la creación de un nuevo entorno Conda. Conda simplemente es un gestor de paquetes, y a su vez un sistema de gestión de entornos de código abierto que está escrito en el lenguaje de programación Python, pero puede gestionar proyectos que contengan código escrito en otros lenguajes, así como proyectos multilenguaje.

Ilustración 17: Conda

Una vez creado un nuevo entorno Conda, se pueden volver a instalar todo el software requerido por DCASE en su versión correcta, ya que Conda será la encargada de hacer la gestión de las versiones.

Los siguientes programas son los requeridos para poder ejecutar la arquitectura de DCASE, y de esta forma también para poder ejecutar la nuestra.

- **Python (versión 3.7):** es un lenguaje de programación mundialmente conocido. Se trata de un lenguaje de programación multiparadigma, ya que soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional. Es un lenguaje interpretado, dinámico y multiplataforma. Python es un arma muy utilizada en las redes neuronales ya que contiene librerías que permite trabajar fácilmente con ellas.

Ilustración 18: Python

- **Numpy:** es una extensión de Python, que le agrega mayor soporte para vectores y matrices, constituyendo una biblioteca de funciones matemáticas de alto nivel para operar con esos vectores o matrices. Proporciona un objeto de matriz multidimensional de alto rendimiento y herramientas para trabajar con estas matrices. Esta librería es de vital importancia en los modelos convolucionales ya que la mayoría del material de trabajo suelen ser vectores y matrices.

Ilustración 19: NumPy

- **TensorFlow (versión 1.14):** TensorFlow facilita la creación modelos de aprendizaje automático. Es una biblioteca de código abierto para aprendizaje automático a través de un rango de tareas, y desarrollado por Google para satisfacer sus necesidades de sistemas capaces de construir y entrenar redes neuronales para detectar y descifrar patrones y correlaciones, análogos al aprendizaje y razonamiento usados por los humanos. TensorFlow puede correr en múltiple CPUs y GPUs, lo cual nos facilita el trabajo ya que AHOLAB dispone de un servidor con 4 GPU.

Ilustración 20: Tensorflow

- **Cudatoolkit:** el kit de herramientas de NVIDIA proporciona un entorno de desarrollo para crear aplicaciones aceleradas por GPU de alto rendimiento. Con CUDA Toolkit, se puede desarrollar, optimizar e implementar sus aplicaciones en sistemas integrados acelerados por GPU.

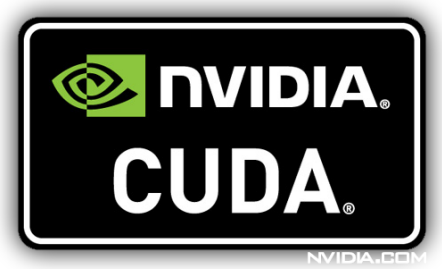

Ilustración 21: Cuda

- **CuDNN:** la biblioteca de red neuronal profunda NVIDIA CUDA (cuDNN) es una biblioteca acelerada por GPU de primitivas para redes neuronales profundas. cuDNN proporciona implementaciones altamente ajustadas para rutinas estándar como convolución hacia adelante y hacia atrás, agrupación, normalización y capas de activación. Este programa nos viene muy bien para trabajar con nuestro modelo convolucional.
- **LibROSA:** LibROSA es un paquete de Python para análisis de música y audio. Proporciona los bloques de construcción necesarios para crear sistemas de recuperación de información musical. Contiene las siguientes funcionalidades que nos ayudan tratar y procesar los audios guardados en la base de datos: estimar tempo y detectar eventos de ritmo, extracción y visualización de características, generación de banco de filtros, calcular representaciones en espectrogramas…

Ilustración 22: Librosa

- **Absl-py (versión 0.9**): es una colección de código de biblioteca de Python para construir aplicaciones de Python. El código se recopila de la propia base de código Python de Google, y ha sido ampliamente probado y utilizado en producción.
- **Dcase_util (versión 0.2.12):** esta colección de utilidades es para la detección y clasificación de escenas y eventos acústicos. Estas utilidades se crearon originalmente para los sistemas de referencia DCASE *Challenge* y se agrupan en una biblioteca independiente para permitir su reutilización en otros proyectos de investigación. El objetivo principal de las utilidades es simplificar el código de investigación, hacerlo más legible y más fácil de mantener.
- **Sed_eval:** proporciona herramientas para evaluar los sistemas de clasificación de escenas acústicas.

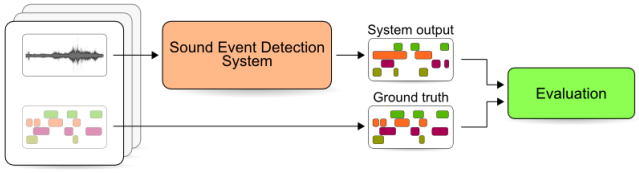

Ilustración 23: Sed_eval

9. Experimentos y análisis de los resultados

En esta sección se van a resumir los resultados obtenidos en cada entrenamiento con el objetivo de mejorar los resultados logrados con la arquitectura de AHOLAB. Para ello, se ha ido iterando el valor de cada hiperparámetro para buscar cuando es máximo. Por cada entrenamiento lanzado se ha cambiado el valor de un único hiperparámetro y de esta forma, poco a poco, se ha ido consiguiendo una sustancial mejora de los resultados. El entrenamiento se ha realizado tanto para el *task1a* como para el *task1b*, pero hay que recalcar que hemos centrado la mayoría de los esfuerzos y de los recursos computacionales en mejorar el *task1b,* ya que los resultados obtenidos con la arquitectura de AHOLAB en el *task1b* han sido más satisfactorios que los logrados en el *task1a*. Por lo que la metodología a seguir en cuanto al *task1a* y al *task1b* ha sido la siguiente: en el *task1b* se han hecho las pruebas para buscar el máximo de cada parámetro y una vez encontrado se ha implementado la mejora en el *task1a* para comprobar si ahí también sucedía.

De esta forma, en los siguientes subapartados se van a poder ver los resultados obtenidos en la búsqueda del mejor valor de cada hiperparámetro. Cada subapartado será destinado al análisis de un hiperparámetro y se mencionará la arquitectura previa utilizada para realizar la mejora, se mostrarán los resultados obtenidos y al final se elegirá el mejor valor. Solo se van a mostrar los resultados más significativos y más relevantes respecto al resultado final, esto quiere decir que además de los resultados publicados aquí, se han realizado muchas más pruebas.

El primer cambio que afectará a todos las pruebas realizadas será bajar el número de *epoch* con el objetivo de reducir el tiempo de entrenamiento. Como se ha podido observar en los resultados para la arquitectura de AHOLAB a partir del *epoch* 130 las pérdidas son mínimas y constantes, por lo que se ha decidido que para los siguientes entrenamientos el valor de las *epoch* sea 130.

9.1. Resultados Task1b

Este ha sido el *task* con el que más hemos trabajado y en el que más pruebas se han realizado. Para cada análisis se ha escogido un hiperparámetro a estudiar, y manteniendo la otra parte de la arquitectura igual, se ha ido cambiando el valor de ese hiperparámetro en busca del mejor resultado. En los siguientes subapartados se mostrarán los resultados de los hiperparámetros más importantes.

En el anexo1 está publicado el mejor resultado de cada hiperparámetro más detalladamente, se representa mediante una tabla cual es porcentaje de acierto en cada caso para cada clase. Los hiperparámetros con los que se van a realizar las pruebas son los siguientes: número de filtros mel, tamaño del kernel, número de filtros de las capas convolucionales,

cantidad de *units* por capas, tamaño enventanado y salto enventanado, número de capas convolucionales y número de capas totalmente conectadas.

9.1.1. Número de filtros mel

El número de filtros mel a aplicar es muy importante ya que implica cuanto se va parecer la señal de entrada al oído humano. Después de haber aplicado 60 se va a probar con los típicos valores de 80, 128 y 256 ya que son los más utilizados en la mayoría de los trabajos. Aunque se hará una prueba con un valor bajo para demostrar que bajar la cantidad no es eficiente.

Para que el modelo siga siendo unidimensional cada vez que se modifique el número de filtros mel a aplicar, también se debe cambiar el tamaño del *kernel*. Para ello el primer valor del *kernel* siempre debe ser igual que la cantidad de mel que se ha utilizado, es decir, en nuestro caso serán 30x6, 80x6, 128x6 y 255x6.

Número de mel	30	60	80	128	256
Precisión	79,2%	83,7%	83,2%	82,7%	80,4%

Tabla 13: Resultados filtros mel

Arquitectura utilizada	
Tamaño enventanado(s)	0,04
Tamaño salto enventanado(s)	0,02
Numero de capas convolucion	2
Número de filtros Conv. 1	80
Número de filtros Conv. 2	80
Pooling	1 x 3
Capas totalmente conectadas	2
Neuronas por capa	2000
Dropout	0,5
Epoch	130
Learning-rate	0,00001

Tabla 14: Arquitectura para el análisis de filtros mel

Ilustración 24: Gráfica filtros mel

Después de analizar los resultados obtenidos, se puede decir que el número de filtros mel más apropiados para simular el oido humana en este caso son 60, un valor que entra dentro de los utilizados por AHOLAB. Como se puede observar en la gráfica, cuanto mayor es la cantidad de filtros mel aplicados menor es el rendimiento del sistema. También se demuestra que al bajar el número de filtros la precisión baja considerablemente.

9.1.2. Tamaño del kernel

Como se ha dicho anteriormente el tamaño del *kernel* siempre debe tener relación directa con el número de filtros mel. Pero tras unas pruebas realizadas junto a Itxasne Díaz, investigadora del grupo AHOLAB, nos hemos encontrado con que reduciendo mínimamente el tamaño del *kernel* se puede lograr una mejora de los resultados.

En la prueba anterior se ha determinado que el número de filtros a aplicar deben de ser 60 por lo que los valores con los que vamos a probar serán muy cercanos a los 60 originales, y después vamos a trasladar los resultados a una gráfica para determinar mediante la curva cuál es el valor del *kernel* con el que se hacen máximos los resultados. Por otro lado, como prueba de que reducir el tamaño del *kernel* en demasía no sirve se ha probado con un valor muy inferior a 60. Los valores con los que se va probar van a ser los siguientes: 30x6 58x6, 57x6, 56x6 y 55x6.

Tamaño del *kernel*	30x6	58x6	57x6	56x6	55x6
Precisión	77,3%	83,2%	83,1%	85,1%	83,9%

Tabla 15: Resultados tamaño kernel

Arquitectura utilizada	
Tamaño enventanado(s)	0,04
Tamaño salto enventanado(s)	0,02
Número de flitros mel	60
Numero de capas convolucion	2
Número de filtros Conv. 1	80
Número de filtros Conv. 2	80
Pooling	1 x 3
Capas totalmente conectadas	2
Neuronas por capa	2000
Dropout	0,5
Epoch	130
Learning-rate	0,00001

Tabla 16: Arquitectura para el análisis del tamaño del kernel

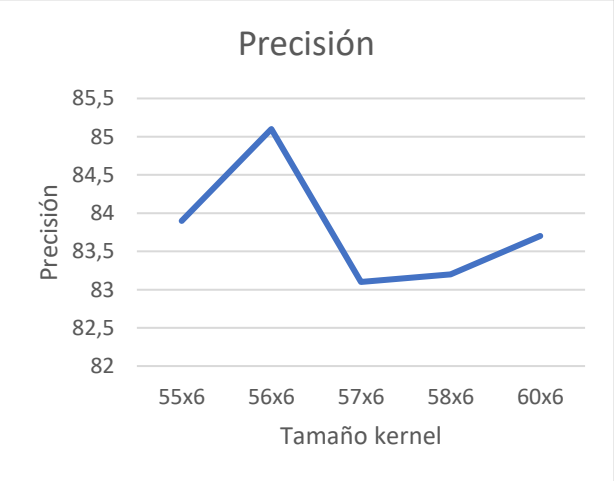

Ilustración 25: Gráfica tamaño de kernel

En la gráfica de precisión se puede observar que hemos detectado un máximo cuando el tamaño del kernel es 56x6. La hipótesis de esta mejora es la siguiente: con un filtro de 60x6 que abarca todo el espectrograma solo se buscan los patrones en una colocación concreta y en caso de que el patrón este colocado un poco mas arriba o mas abajo el fitro no es capaz de detectarlo. Esto es muy común en los espectrogramas, ya que los patrones a buscar pueden fluctuar un poco arriba o un poco abajo. Sin embargo, un filtro demasiado pequeño impide que los patrones grandes puedan ser detectados con claridad.

9.1.3. Número de filtros de las capas convolucionales

Para las pruebas anteriormente realizadas las dos capas convolucionales eran exactamente iguales. En esta se plantea la alternativa de cambiar el número de filtros de cada capa, teniendo en cada una de ellas una cantidad diferente. Lo más común es que cada capa tenga entre 32 y 512 filtros diferentes. La diferencia entre esos dos dígitos nos abre un abanico de combinaciones demasiado amplio, por lo que nos hemos visto en la obligación de acotarlas. Los resultados siempre han sido mejores cuando la primera capa convolucional ha tenido más filtros que la segunda y encima tras una serie de pruebas hemos observado como al aplicar más de 80 filtros los resultados empiezan a decrecer. Por lo que tras una serie de pruebas con las siguientes combinaciones estos son los resultados obtenidos:

Filtros Conv1	80	80	75	64	80
Filtros Conv2	32	64	50	32	80
Precisión	83,4%	85,2%	84,5%	85%	85,1%

Tabla 17: Resultados filtros

Arquitectura utilizada	
Tamaño enventanado(s)	0,04
Tamaño salto enventanado(s)	0,02
Número de flitros mel	60
Numero de capas convolucion	2
Tamaño kernel	56x6
Pooling	1 x 3
Capas totalmente conectadas	2
Neuronas por capa	2000
Dropout	0,5
Epoch	130
Learning-rate	0,00001

Tabla 18: Arquitectura para el análisis de número de filtros

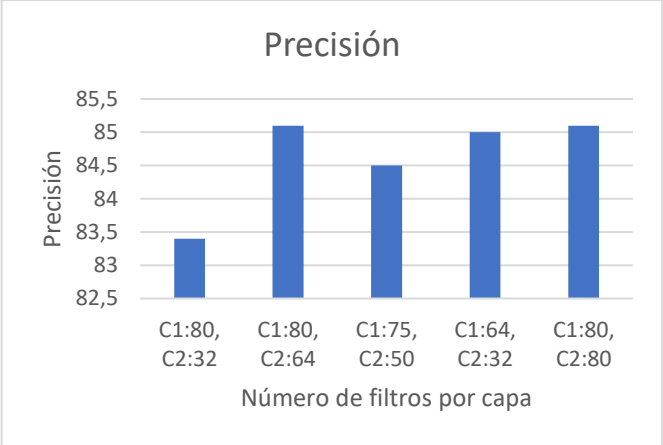

Ilustración 26: Gráfica número de filtros

Después de muchas pruebas e innumerables combinaciones de filtros, hemos determinado que la mejor arquitectura es que la primera capa convolucional tenga 80 filtros y la segunda 64 filtros. Es cierto que la precisión es prácticamente igual que con el modelo de AHOLAB (80 en la primera y 80 en la segunda) pero en precisiones tan altas como el 85,2% una sola decima de mejora hace que se cambien parámetros del modelo.

9.1.4. Número de units por capa

Encontrar el número de neuronas ideal es una tarea muy compleja. En esta arquitectura hemos estado aplicando 2000 por cada capa totalmente conectada, pero una cantidad tan elevada podría acarrear *overfitting* al sistema. Por lo que se ha intentado hacer unas pruebas con menos neuronas por capa para estudiar si nuestro sistema sufría *overfitting* y en caso que se diese, poder encontrar una cantidad más óptima de neuronas.

Units	3000	2000	1500	1000	500	100
Precisión	80,5%	85,2%	85,3	83,4%	83,7%	85,4%

Tabla 19: Resultados número de neuronas

Arquitectura utilizada	
Tamaño enventanado(s)	0,04
Tamaño salto enventanado(s)	0,02
Número de flitros mel	60
Numero de capas convolucion	2
Tamaño kernel	56x6
Número de filtros Conv. 1	80
Número de filtros Conv. 1	64
Pooling	1 x 3
Capas totalmente conectadas	2
Dropout	0,5
Epoch	130
Learning-rate	0,00001

Tabla 20: Arquitectura para el análisis de número de neuronas

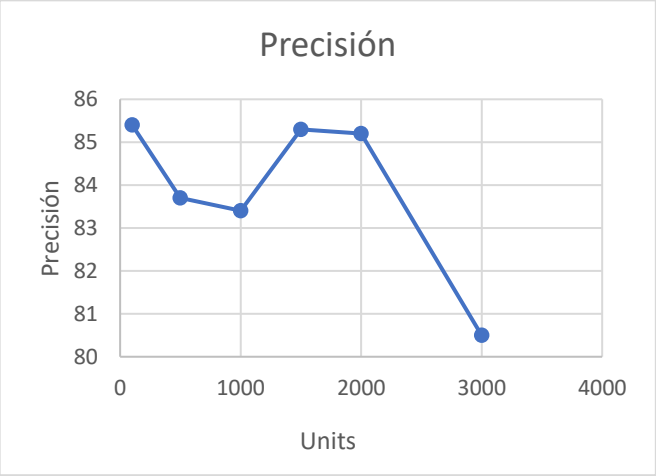

Ilustración 27: Gráfica neuronas

Como consecuencia de las pruebas realizadas se puede concluir que con 2000 neuronas la red neuronal no sufre *overfitting.* Además, con 1500 *units* se encuentra un máximo de rendimiento, pero esto sucede también con 100 *units*. Por lo que hemos determinado aplicar 100 neuronas por cada capa, ya que de esta forma se puede reducir enormemente el coste computacional para obtener los mismos resultados, aunque nos desviemos de la arquitectura de AHOLAB original.

Otra conclusión que se puede sacar observando la gráfica de los resultados es que a partir de las 2000 neuronas el sistema sí que sufre el efecto *overfitting*.

9.1.5. Tamaño enventanado y tamaño salto enventanado

El tamaño del enventanado es de suma importancia ya que representa cuanta cantidad de información se va analizar por imagen. Un enventanado de mucho tiempo podría reunir demasiada información y de esta forma dificultar el encuentro de patrones. Sin embargo, un enventanado muy pequeño puede confundir a la red en la búsqueda de patrones.

El salto de enventanado también tiene su importancia, un salto de enventanado apropiado propicia que no se pierda nada de información y que el procesamiento de los datos sea fluido al mismo tiempo. Un enventanado demasiado grande produce pérdida de información y uno demasiado pequeño una carga computacional grande.

Para las pruebas se seguirá el criterio de que hay un solapamiento mínimo del 50% pero se realizará alguna prueba con un solapamiento superior e inferior.

Tamaño enventanado	0,02	0,04	0,04	0,4
Tamaño salto enventanado	0,01	0,01	0,03	0,2
Precisión	87%	84,6%	84,9%	85,2%

Tabla 21: Resultados enventanado

Arquitectura utilizada	
Número de flitros mel	60
Numero de capas convolución	2
Tamaño kernel	56x6
Número de filtros Conv. 1	80
Número de filtros Conv. 1	64
Pooling	1 x 3
Capas totalmente conectadas	2
Neuronas por capa	100
Dropout	0,5
Epoch	130
Learning-rate	0,00001

Tabla 22: Arquitectura para el análisis de tamaño del enventanado

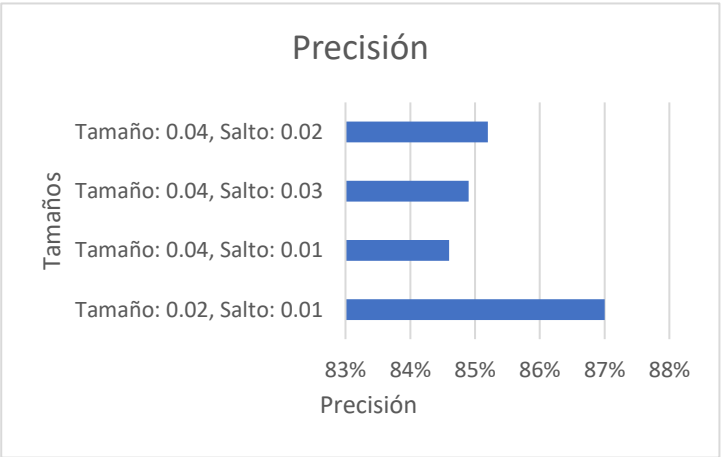

Ilustración 28: Gráfica tamaño enventanado

Haciendo un correcto manejo del enventanado se obtiene una considerable mejora de los resultados. Como se puede observar en la gráfica de precisión, ni aumentando ni reduciendo el solapamiento se consigue un aumento en los resultados. Sin embargo, si se reduce tanto el tamaño del enventanado como el del salto la mejoría es notoria.

9.1.6. Dropout

El *dropout* de cada capa totalmente conectada es otro de los hiperparámetros a analizar. Aunque se entienda que el ideal es el 50%, se van hacer pruebas con un porcentaje mayor y con uno menor, 70% y 30% respectivamente.

Dropout	30%	50%	70%
Precisión	85,3%	87%	81,2%

Tabla 23: Tabla resultados dropout

Arquitectura utilizada	
Número de flitros mel	60
Numero de capas convolucion	2
Tamaño kernel	56x6
Número de filtros Conv. 1	80
Número de filtros Conv. 2	64
Pooling	1 x 3
Capas totalmente conectadas	2
Neuronas por capa	100
Dropout	0,5
Epoch	130
Learning-rate	0,00001

Tabla 24: Arquitectura para el análisis de dropout

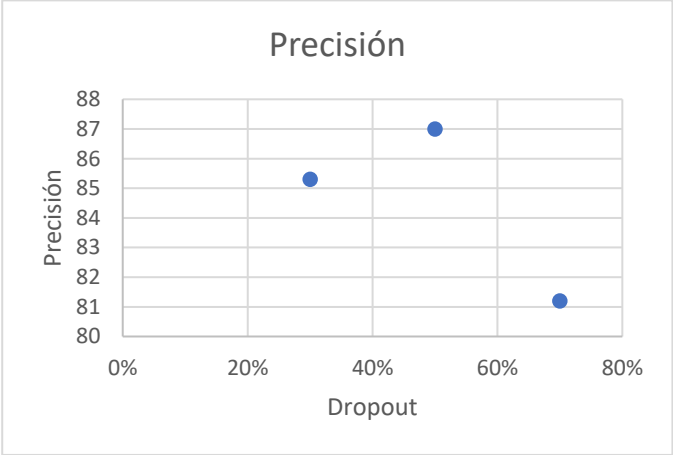

Ilustración 29: Gráfica dropout

Como muestran los resultados de esta prueba, el valor del *dropout* debe ser 0,5, es decir que en cada *epoch* se desconecten la mitad de las neuronas. En la gráfica se puede observar claramente que tanto aumentando como reduciendo el valor del *dropout*, los resultados empeoran. Esto puede suceder porque al desconectar demasiadas neuronas el sistema no sepa en clase clasificar cada muestra o porque desconectando pocas neuronas el sistema sea demasiado dependiente de unas determinadas neuronas.

9.1.7. Número de capas convolucionales

En la arquitectura desarrollada por AHOLAB se aplicaron dos capas convolucionales, y esta cantidad es la más común en los modelos convolucionales. Entendiendo que reduciendo el número de dos capas a una, los resultados empeorarían, se va probar la alternativa de aumentar la cantidad de capas. Se van hacer la prueba de implementar 3, 4, 5 y hasta 6 capas convolucionales, cada una seguida de su respectiva capa de reducción. La estructura de las capas añadidas serán idénticas a la primera.

Número de capas convolucionales	2	3	4	5	6
Precisión	87%	85,7%	87,2%	87,8%	86,8%

Tabla 25: Resultados número de capas convolucionales

Arquitectura utilizada	
Tamaño enventanado(s)	0,02
Tamaño salto enventanado(s)	0,01
Número de flitros mel	60
Tamaño kernel	56x6
Número de filtros Conv. 1,3,4,5	80
Número de filtros Conv. 1	64
Pooling	1 x 3
Capas totalmente conectadas	2
Neuronas por capa	100
Dropout	0,5
Epoch	130
Learning-rate	0,00001

Tabla 26: Arquitectura para el análisis de número de capas convolucionales

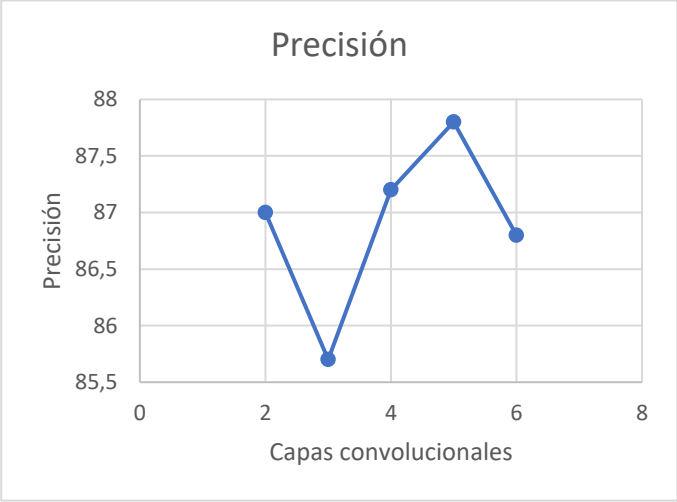

Ilustración 30: Gráfica número de capas convolucionales

Aumentando el número de capas convolucionales se consigue la mayor mejora de resultados, logrando un 87,8% de precisión y de esta forma superando a la *Baseline* de DCASE. Gracias a insertar más capas convolucionales el sistema consigue hacer un análisis más profundo de cada espectrograma y de esta forma pasarle a la capa totalmente conectada una información más precisa. Aunque hay que mencionar que incluir demasiadas capas (6 capas) puede repercutir negativamente ya el sistema puede tener un exceso de información y no ser capaz de clasificar correctamente.

9.1.8. Número de capas totalmente conectadas

En cuanto al número de capas totalmente conectadas, en la mayoría de las arquitecturas publicadas hasta el momento se han usado 2, ya que con este número se era posible conseguir buenos resultados y además reducir la cantidad de parámetros. Pero, aun así, se va asumir la gran cantidad de parámetros y de carga computacional añadiendo más capas totalmente conectadas. Se va hacer la prueba con 3 y con 4.

Número de capas totalmente conectadas	2	3	4
Precisión	87,8%	85,9%	85%

Tabla 27: Resultados número de capas conectadas

Arquitectura utilizada	
Tamaño enventanado(s)	0,04
Tamaño salto enventanado(s)	0,02
Número de flitros mel	60
Numero de capas convolucion	2
Número de filtros Conv. 1,3,4,5	80
Número de filtros Conv. 2	64
Pooling	1 x 3
Neuronas por capa	100
Dropout	0,5
Epoch	130
Learning-rate	0,00001

Tabla 28: Arquitectura para el análisis de número de capas totalmente conectadas

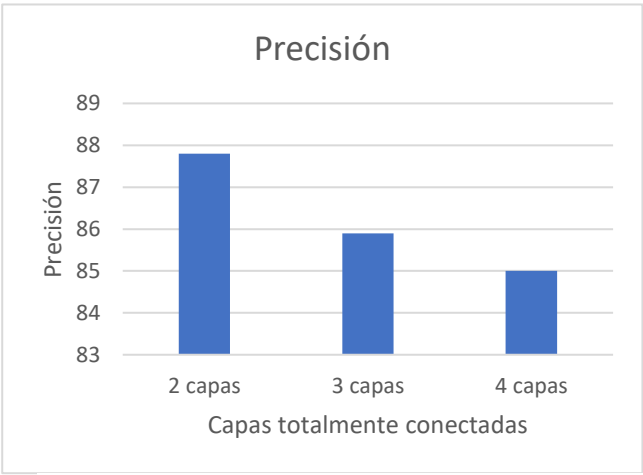

Ilustración 31: Gráfica capas totalmente conectadas

Claramente se puede deducir que el número idóneo de capas totalmente conectadas es dos y que a partir de ahí el rendimiento del sistema baja. Esto puede ser causa del *overfitting,* ya que se introducen más neuronas repartidas en más capas.

9.1.9. Cross validation

Una vez finalizado el proceso de entrenamiento y obtenido una arquitectura con la cual se han conseguido los mejores resultados, hemos procedido a verificar que los resultados logrados con esa arquitectura son fiables, y para ello se ha aplicado un método llamado *cross validation.* Este método consiste en utilizar la misma arquitectura cinco veces para entrenar y evaluar el sistema, pero en cada intento se usa una parte distinta de la base de datos para el entrenamiento y otra distinta para la evaluación. En nuestro caso, la base de datos se divide en el 30% para validar y en 70% para entrenar, pero utilizando cross validation en cada intento, los datos reservados en el 30% de evaluación cambian aleatoriamente. De esta forma se consigue comprobar que la arquitectura es fiable y que no sirve para clasificar solamente un determinado grupo de archivos.

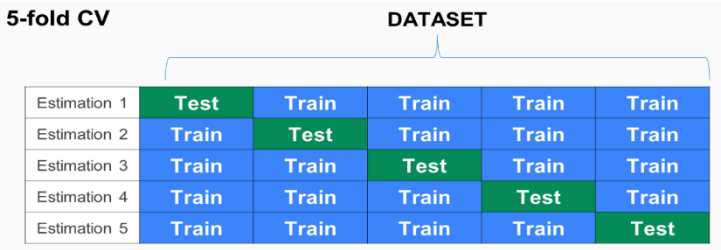

Ilustración 32: Cross validation

En la siguiente tabla se puede observar los resultados obtenidos en los 5 intentos realizados:

Intento 1	Intento 2	Intento 3	Intento 4	Intento 4	Media
87,8%	88,1%	87,8%	87,5%	87,6%	87,7 ± 0,3%

Tabla 29: Resultados finales task1b

En la Tabla 30 se puede ver con detalle la precisión para cada clase.

Escena	Precisión	Pérdidas
Indoor	83,5%	0,856
Outdoor	90,6%	0,51
Transportat	89,2%	0,75
----------	----------	----------
Precisión total	87,8%	0,691

Tabla 30: Mejor resultado task1b detallado

9.1.10. Análisis de los resultados del task1b

En resumen, se ha encontrado una arquitectura que ha sido capaz de superar los resultados obtenido por la *Baseline* (87,3%) con las siguientes características:

Arquitectura utilizada	
Tamaño enventanado(s)	0,04
Tamaño salto enventanado(s)	0,02
Número de flitros mel	60
Numero de capas convolucion	2
Número de filtros Conv. 1,3,4,5	80
Número de filtros Conv. 2	64
Pooling	1 x 3
Capas totalmente conectadas	2
Neuronas por capa	100
Dropout	0,5
Epoch	130
Learning-rate	0,00001

Tabla 31: Arquitectura task1b final

El modelo presenta algunas variantes respecto al desarrollado por AHOLAB como pueden ser el aumento de capas convolucionales o la reducción del enventanado. Pero gracias a estos ajustes en los hiperparámetros se ha conseguido mejorar los resultados que se consiguieron aplicando directamente la arquitectura de AHOLAB (83,7%)

También hay que mencionar que esta arquitectura no cumple con uno de los requisitos del task1b del concurso de DCASE que es que el sistema debe pesar menos de 500KB, y nuestro modelo pesa 33,4MB, una cantidad demasiado alta.

9.2. Resultados Task1a

Tal y como se ha comentado en este mismo apartado, para el *task1a* se ha aplicado menos tiempo y menos entrenamientos. Por lo que para la mejora del *task1a* nos hemos basado en el *task1b,* ya que, si en uno mejoraba, probabamos el cambio en el otro. Esto no implica que el ajuste de un hiperparámetro que mejore el task1b tenga que mejorar obligatoriamente el *task1a*. En cada *Challenge* se tienen que clasificar diferentes tipos de sonidos urbanos por lo que en la arquitectura de cada uno puede haber modificaciones.

En las siguientes tablas se va mostrar la arquitectura y el mejor resultado obtenido para el task1a, pero en caso de querer ver detalladamente el resultado, pudiendo observar la precisión de cada dispositivo para cada clase puedes ir al anexo 2. Como sucede en el *task1b*, para validar que el resultado final es fiable se ha aplicado la técnica de cross-validation.

Arquitectura utilizada	
Tamaño enventanado(s)	0,04
Tamaño salto enventanado(s)	0,02
Número de flitros mel	60
Numero de capas convolucion	2
Número de filtros Conv. 1,3,4,5	80
Número de filtros Conv. 2	64
Pooling	1 x 3
Capas totalmente conectadas	2
Neuronas por capa	100
Dropout	0,5
Epoch	130
Learning-rate	0,00001

Tabla 32: Arquitectura task1a

Intento 1	Intento 2	Intento 3	Intento 4	Intento 5	Media
43,1%	43,6%	43,5%	43,5%	43,4%	43,4 ± 0,2%

Tabla 33: Resultados finales task1a

9.2.1. Análisis resultados task1a

Esta segunda tarea claramente no ha conseguido los objetivos propuestos. El resultado final (43,4%) es demasiado bajo para una clasificación de este tipo. Además, ni siquiera hemos conseguido superar a la *Baseline* ofrecida por DCASE, quedando casi 10 puntos por debajo de ella. Aunque, teniendo en cuenta el tiempo invertido en esta tarea los resultados van acordes al esfuerzo realizado

En un futuro, se podría continuar trabajando con esta tarea, ya que como ha quedado demostrado, cambiando algunos hiperparámetros se pueden obtener mejores resultados.

10. Planificación del proyecto

Mediante este apartado se va a describir cual ha sido la planificación del proyecto para la realización del mismo. El proyecto se ha dividido en diferentes paquetes de trabajo, y en cada uno de ellos ha habido una serie de tareas a realizar. Para elaborar una correcta planificación hay que tener en cuenta la fecha de inicio y la fecha final del proyecto, que en este caso son 1 de octubre de 2019 y 21 de julio de 2020 respectivamente.

También se deben mencionar a las personas que han participado en este largo proyecto de una manera u otra, en mayor o en menor medida. Sin la ayuda de ellos probablemente hubiese sido muy complicado llevar a cabo el proyecto.

- Ibon Saratxaga: director del TFG, ingeniero en telecomunicaciones y profesor de la Escuela de Ingeniería de Bilbao. Su función principal ha sido la de dirigir el trabajo y ayudar con los pasos a seguir durante el trabajo.
- Itxasne Díez: ingeniera en telecomunicaciones y participante del grupo de investigación AHOLAB. Conoce perfectamente el modelo convolucional y ha sido de vital ayuda en su implementación.

Participante	Abreviatura	Función
Peio Gonzalez	PG	Ingeniero junior
Ibon Saratxaga	IS	Ingeniero senior
Itxasne Díez	ID	Ingeniera senior

Tabla 34: Participantes en el proyecto

A continuación, se va a describir el trabajo realizado en cada paquete y en cada una de sus tareas. Se detallará la fecha de inicio y la de final, la duración tanto en días como en horas y los participantes de cada paquete. Además, se especificará los responsables y la duración (en días y horas) de las tareas de cada paquete. Hay que tener en cuenta que los paquetes se irán realizando secuencialmente, es decir, hasta que no concluya el anterior no se comenzará con el siguiente. En cuanto a las tareas, algunas se realizarán secuencialmente y otras en paralelo (siempre y cuando las tareas realizadas en paralelo pertenezcan al mismo paquete)

PT1: Definición del proyecto

La premisa de este primer paquete de trabajo es definir cuáles van a ser los objetivos a cumplir con este trabajo, para ello se hará un análisis de los *Challenge* disponibles en DCASE. A parte, pero en paralelo, se realizará un análisis exhaustivo del sistema previamente desarrollado por AHOLAB. De esta forma, al término de este primer paquete se tendrá el suficiente conocimiento sobres redes neuronales y, en concreto, sobre el modelo de AHOLAB como para poder comenzar con el proyecto

Fecha de inicio	1 de octubre de 2019
Fecha final	31 de octubre de 2019
Duración (días)	30
Carga de trabajo (horas)	50
Participantes	PG, IS

Tabla 35: Datos PT1

En este paquete de trabajo las dos últimas tareas, A102 y A103, se realizarán en paralelo, pero no comenzarán hasta que no finalice la primera, A101.

A101: Descripción de especificaciones básicas

El objetivo de esta tarea es especificar brevemente cómo será el funcionamiento del proyecto. Es decir, se explicarán cuáles van a ser los objetivos del proyecto y qué es lo que el proyecto representa de una forma breve pero concreta.

A102: Análisis del *Challenge* DCASE

A través de esta tarea se analizarán los diferentes *task* o *Challenge* que propone DCASE. Para ello, se revisará cada uno detalladamente entendiendo las características de cada uno. En esta tarea se escogerá o se escogerán los diferentes *task* con los que se trabajará durante el TFG. El hecho de escoger un *task* en esta tarea no implica que al final se deba presentar a concurso o que se deba trabajar sobre él hasta el final del proyecto, por lo que si surgiese algún problema durante el trabajo se podría decidir abandonar un *task*. Pero es de suma importancia elegir los *task* que mejor se adapten a la arquitectura diseñada por AHOLAB y a la vez que esos *task* tengan

la dificultad apropiada. Para realizar esta tarea se revisará la página oficial de DCASE donde aparecen detalladamente todas las características de los diferentes *task*.

A103: Estudio sobre redes convolucionales y del sistema de AHOLAB

Al final de esta tarea se deberá haber aprendido y entendido la arquitectura creada y utilizada por AHOLAB. Para ello se deberá leer el "Creación de una herramienta de detección y clasificación de sonidos para vehículos"[14] presentado por el director del TFG, Ibon Saratxaga, en el que se explica minuciosamente el sistema de detección y clasificación de eventos sonoros creado por AHOLAB. A su vez, se realizará una investigación sobre las redes neuronales convolucionales con el fin de poder aplicar eso conocimientos en este proyecto.

Tarea	Responsable(s)	Duración (días)	Carga de trabajo (horas)
A101	PG, IS	7	5 PG, 5 IS
A102	IS	17	10 IS
A103	PG	17	30 PG

Tabla 36: Tareas PT1

PT2: Instalación de la *Baseline* DCASE2019

Con este paquete se busca la correcta instalación de la *Baseline* de DCASE para 2019. Se instalará el de 2019 y no el de 2020, que es el que se va utilizar para entrenar el sistema de cara al *Challenge*, porque para la fecha en la que se realiza este paquete aún no está publicado la *Baseline* de 2020 por parte de DCASE. Por lo tanto, se instalará el de 2019 hasta que esté disponible el de 2020. Para esto se deberá crear un nuevo entorno Conda en los servidores de AHOLAB y después instalar los programas necesarios. Por último, se ejecutará la *Baseline* para comprobar que la instalación se ha realizado correctamente.

Otro de los objetivos a cumplir por medio de este paquete es el aprendizaje de utilización de los servidores de AHOLAB, como por ejemplo el funcionamiento y distribución de las 4 GPU o el linkeamiento de los discos duros.

En lo que al orden de las tareas respecta, en este paquete se realizarán todas las tareas secuencialmente

Fecha de inicio	1 de noviembre de 2019
Fecha final	22 de noviembre de 2019
Duración (días)	20
Carga de trabajo (horas)	25
Participantes	PG, IS

Tabla 37: Datos PT2

A201: Creación de un entorno Conda y familiarización con el servidor

En esta primera tarea del paquete de trabajo 2 se creará un nuevo entorno Conda dentro del usuario correspondiente. Para ello se seguirán una serie de instrucciones que ofrece DCASE. A parte se iniciará un proceso para familiarizarse con el método de uso del servidor.

A202: Instalación del software necesario

Una vez creado un nuevo entorno Conda, se pueden empezar a instalar los diferentes softwares necesarios para poder ejecutar la *Baseline*. En la página web de DCASE se especifica claramente que software es necesario y en cuál de sus versiones, siendo esto último algo muy importante ya que instalar una versión demasiado vieja o nueva podría causar errores en la ejecución de la *Baseline*. De esta manera se conseguirá un entorno Conda en el cual habrá todo lo necesario para luego trabajar sobre ella sin ningún tipo de problema. Existe la posibilidad de crear más entornos Conda si fuese necesario durante esta fase, de esta manera se podría tener diferentes versiones de software en cada uno.

A203: Ejecución de la *Baseline*

Cuando se termine la tarea anterior y se disponga de un entorno Conda en perfectas condiciones se procederá a instalar la o las *Baseline* de los *task* que se hayan elegido para trabajar. También se instalarán las bases de datos que contienen los audios en formato .wav, y como el *Challenge* de 2019 ya ha finalizado se pueden instalar tanto el paquete de entrenamiento como el de evaluación. Una vez instaladas las *Baseline*, se ejecutarán y se comprobará que los resultados son los esperados, para ello se debe consultar la página oficial de DCASE donde vienen publicados los resultados de la *Baseline* de cada *task*. En caso de que los resultados sean correctos se comenzará con el siguiente paquete de traba y en caso contrario se deberá revisar la tarea número dos de este paquete.

Tarea	Responsable(s)	Duración (días)	Carga de trabajo (horas)
A201	PG, IS	5	5 PG, 5 IS
A202	PG	10	10 PG
A203	PG	5	5 PG

Tabla 38: Tareas PT2

PT3: Entrenamiento del Sistema DCASE2019

Como se ha explicado anteriormente, mediante este trabajo se busca la implementación del sistema desarrollado por AHOLAB en la clasificación de eventos sonoros recogidos en una ciudad. El objetivo de este paquete es intentar llevar la arquitectura de AHOLAB a cada uno de los *task* con los que se va trabajar. Por eso es muy importante haber estudiado previamente como es la arquitectura propuesta por AHOLAB. Además, una vez trasladada la arquitectura a los *task*, se va buscar una mejora de los resultados. Para ello, se irán haciendo modificaciones en algunos de los hiperparámetros de la arquitectura para buscar donde es máxima la precisión de la misma. Al final de cada entrenamiento se tendrá que realizar un análisis de los resultados para decidir si el entrenamiento ha sido satisfactorio.

Fecha de inicio	20 de noviembre de 2019
Fecha final	28 de febrero de 2020
Duración (días)	72
Carga de trabajo (horas)	75
Participantes	PG, IS

Tabla 39: Datos PT3

En este paquete, como sucede con el anterior, todas las tereas se realizarán secuencialmente.

A301: Implementación de la arquitectura

En esta primera tarea del paquete se irá implementando poco a poco la arquitectura de AHOLAB en el task1b de 2019. La razón para realizarlo poco a poco y no todo de golpe es que se va buscar si en algún momento, sin que esté totalmente implementada, se encuentra un máximo de precisión en los resultados y poder tomar ese sistema como referencia.

Se comenzará modificando las redes convolucionales cambiando el tamaño de los filtros y el *kernel*. Luego, se convertirán en redes unidimensionales y cambiarán una serie de sus hiperparámetros. A continuación, se modificando otros apartados como las capas de *pooling*, las capas totalmente conectadas, el *dropout*…

A302: Busca de mejoras

Una vez implementada la arquitectura y sabiendo con que sistema es máxima la precisión, se buscará la mejora de los resultados. Para ello se irá modificando la arquitectura, pero siempre sin alejarse de la desarrollada por AHOLAB. Después de cada entrenamiento se hará un análisis de los resultados para determinar si los cambios realizados en la arquitectura para ese entrenamiento han sido correctos y decidir que hiperparámetros añadir, cambiar o eliminar antes de lanzar la siguiente ejecución.

A303: Simulación de evaluación

En las dos tareas anteriores se ha trabajado con la parte de la base de datos correspondiente al entrenamiento. Pero para terminar con este paquete se hará una simulación de cómo se tendría que ejecutar el sistema para mandar los resultados al *Challenge*. Para ello debe escoger el sistema donde los resultados hayan sido mejores, y volver a ejecutarlo en modo evaluación y los resultados recogerlos en un archivo .csv. Este último documento es el que se debería enviar al *Challenge*.

Tarea	Responsable(s)	Duración (días)	Carga de trabajo (horas)
A301	PG, IS	30	25 PG, 5 IS
A302	PG, IS	35	35 PG,

			5 IS
A303	PG	7	5 PG

Tabla 40: Tareas PT3

PT4: Instalación del sistema DCASE2020

El 1 de marzo de 2020 DCASE publica en su página web las *Challenge* propuestos para 2020. Estos *Challenge* no variarán mucho de los de 2019 pero se deben repetir todos los pasos anteriores. Lo primero de todo será analizar cada *task* para decidir en principio en cuales se va a participar. Luego se instalará un nuevo entorno Conda y sus correspondientes programas. Para acabar se ejecutarán las *Baseline* de los *task* elegidos.

Fecha de inicio	1 de marzo de 2020
Fecha final	20 de marzo de 2020
Duración (días)	20
Carga de trabajo (horas)	25
Participantes	PG, IS

Tabla 41: Datos PT4

En lo que al orden de las tareas de este paquete respecta, se realizarán todas en modo secuencial, una tras otra.

A401: Elección de los *task*

Como se realizó en la segunda tarea del primer paquete de trabajo, se analizarán los diferentes *task* propuestos por la organización del concurso, y se buscará uno o unos que se adapten a la dificultad que se busca y que se adapte también a la arquitectura de AHOLAB.

A402: Creación de nuevo entorno Conda y su configuración

En esta tarea se creará un nuevo entorno Conda con el que se trabajará de aquí en adelante. La creación de este nuevo entorno es necesario ya que hay algunas modificaciones de las versiones de algunos programas respecto al año pasado, así como *keras*, *tensorflow* o *phyton*.

Como sucedió en el de 2019, en el nuevo entorno Conda hay que instalar los programas exigidos por el concurso en su versión correcta. Los programas y sus respectivas versiones vienen detallados en la página oficial del *Challenge*.

A403: Ejecución de las *Baseline*

Para finalizar con este paquete de trabajo, se ejecutarán las *Baseline* de los *task* elegidos. Para ello, previamente se deben haber instalado los sistemas de los *task* y haber descargado las bases de datos. En este caso, en 2020, únicamente se podrá instalar la parte de la base de datos correspondiente al entrenamiento o al desarrollo, ya que la parte de evaluación no estará publicada cuando se realice esta tarea. Para finalizar con la tarea y el paquete se comprobará que los resultados obtenidos en las *Baseline* son los publicados en la web de DCASE. Si no es así, se deberá repasar los pasos anteriores en busca del error cometido.

Tarea	Responsable(s)	Duración (días)	Carga de trabajo (horas)
A401	IS	5	5 IS
A402	PG, IS	10	10 PG, 5 IS
A403	PG	5	5 PG

Tabla 42: Tareas PT4

PT5: Entrenamiento del Sistema DCASE2020

Unos de los objetivos principales de este paquete de trabajo es la implementación del sistema desarrollado por AHOLAB en los *task* elegidos para trabajar, como ya sucedió en el paquete tres con el *Challenge* de 2019. A su vez, se buscará la mejora de resultados mediante el entrenamiento y las modificaciones de algunos parámetros. En este paquete se tiene la ventaja de que ya se disponen de los resultados conseguidos en 2019 y basándose en ellos ayudará a conseguir más rápido una mejora de los resultados.

Es posible que no se pueda implementar en su totalidad el sistema de AHOLAB ya que las características del *Challenge* lo impiden. Para la implementación del sistema contaremos con la gran ayuda de la anteriormente mencionada Itxasne Díez.

Fecha de inicio	20 de marzo de 2020
Fecha final	1 de junio de 2020
Duración (días)	71
Carga de trabajo (horas)	75
Participantes	PG, IS, ID

Tabla 43: Datos PT5

A501: Implementación de la arquitectura basándose en los análisis

Como ya sucedió en la implementación de la arquitectura para 2019, los cambios no se realizarán todos de golpe, sino que poco a poco. En este caso, como se disponen de los resultados obtenidos para 2019, la implementación se irá realizando basándose en esos resultados. Por eso es necesario hacer un análisis de los resultados conseguidos en el de 2019. Una vez analizado los resultados, la implementación se hará progresivamente buscando donde los resultados fueron mejores en 2019, ya que todo hace indicar que serán también en 2020. Para realizar la implementación, en esta tarea se contará con la ayuda de Itxasne. ya que ella conoce al detalle la arquitectura.

A502: Entrenamiento del sistema

En esta tarea se buscará la mejora del sistema y si es posible mejorar los resultados que ofrece la *Baseline* en cada *task*. Como referencia se tomará el mejor resultado obtenido anteriormente y a partir de ahí se irán haciendo pruebas con diferentes sistemas. Entre las diferentes pruebas que se realizarán están el aumento de capas totalmente conectadas, el aumento o disminución de neuronas y filtros, aplicar o no aplicar *padding*… Pero siempre hay que tener en cuenta las limitaciones que tiene el *Challenge* y que el nuevo sistema no se debe alejar demasiado del propuesto por AHOLAB ya que el objetivo principal es comprobar que el sistema de AHOLAB sirve para clasificar ruidos o sonidos de urbanos. Como ya sucedió en el entrenamiento para el *Challenge* de 2019, el final de cada entrenamiento requiere un análisis de los resultados logrados para determinar los hiperparámetros a cambiar para el siguiente

Tarea	Responsable(s)	Duración (días)	Carga de trabajo (horas)
A501	PG, IS, ID	30	20 PG, 10 IS, 5 ID
A502	PG, IS	41	25 PG, 10 IS

Tabla 44: Tarea PT5

PT6: Análisis de los resultados

Este paquete de trabajo no tendrá ninguna tarea adicional asignada. El objetivo de este paquete es hacer un análisis de los resultados obtenidos y sacar las conclusiones. Los diferentes resultados logrados gracias a los entrenamientos se ordenarán, guardarán y archivarán siguiendo una lógica.

Fecha de inicio	1 de junio de 2020
Fecha final	15 de junio de 2020
Duración (días)	15
Carga de trabajo (horas)	25
Participantes	PG

Tabla 45: Datos PT7

PT7: Gestión del proyecto

En este último paquete se realizará la documentación del proyecto. Se elaborará una memoria que recoja todo lo realizado desde el comienzo del trabajo hasta el final del mismo. En él se explicará los objetivos del trabajo, las investigaciones y avances realizados durante el proyecto, y para acabar se redactarán una serie de conclusiones

Fecha de inicio	15 de junio de 2020

Fecha final	16 de julio de 2020
Duración (días)	31
Carga de trabajo (horas)	200
Participantes	PG, IS

Tabla 46: Datos PT7

A701: Escritura del documento

Esta tarea es la más larga y costosa del paquete, en la que se debe redactar el documento que luego se debe enviar. En él se recogerá de una forma clara y detallada todo lo realizado durante los meses de investigación. Se aclarará cual es la situación inicial, que objetivos se buscan mediante este proyecto y que resultados se han obtenido, realizando un análisis de ellos. La memoria deberá contener una serie de campos exigidos para que la misma sea aprobada.

PT8: Reuniones de seguimiento

Esta tarea se ha realizado durante todo proyecto desde su comienzo. Han sido unas reuniones semanales en las que el director del proyecto, Ibon Saratxaga, ha ido haciendo un control y seguimiento de lo que se ha ido realizando. A la conclusión de la parte práctica y comienzo de la redacción del documento final, en las reuniones se ha hablado sobre que mejorar en el documento y cuáles deben ser los pasos a seguir.

Fecha de inicio	1 de octubre de 2019
Fecha final	16 de julio de 2020
Duración (días)	9 meses
Carga de trabajo (horas)	50
Participantes	PG, IS

Tabla 47: Datos PT8

Diagrama de GANTT

	Task Name	Comienzo	Fin	Nombres de los recursos
1	**Reuniones**	**mié 02/10/19**	**mié 15/07/20**	
44	**Definicion del proyecto**	**mar 01/10/19**	**jue 31/10/19**	
45	Descripción de especifica	mar 01/10/19	mié 09/10/19	PG;IS
46	Análisis de Challenge DCA	jue 10/10/19	jue 31/10/19	IS
47	Estudio del sistema de AH	jue 10/10/19	jue 31/10/19	PG
48	**Instalacione del Baseline D**	**vie 01/11/19**	**vie 22/11/19**	
49	Creación de un entorno C	vie 01/11/19	mié 06/11/19	IS;PG
50	Instalacion del software r	jue 07/11/19	lun 18/11/19	PG
51	Ejecución de la Baseline	mar 19/11/19	vie 22/11/19	PG
52	**Entrenamiento del Sistema**	**lun 25/11/19**	**vie 28/02/20**	
53	Implementación de la arq	lun 25/11/19	jue 02/01/20	IS;PG
54	Busca de mejores	vie 03/01/20	mié 19/02/20	IS;PG
55	Simulación de evaluación	jue 20/02/20	vie 28/02/20	PG
56	**Instalación del sistema DC**	**lun 02/03/20**	**jue 19/03/20**	
57	Elección de los task	lun 02/03/20	mié 04/03/20	IS
58	Creación de un entorno C	jue 05/03/20	vie 13/03/20	IS;PG
59	Ejecución de los Baseline	lun 16/03/20	jue 19/03/20	PG
60	**Entrenamiento del Sistema**	**vie 20/03/20**	**vie 29/05/20**	
61	Implementación de la arq	vie 20/03/20	lun 27/04/20	IS;PG;ID
62	Entrenamiento del sistem	mar 28/04/20	vie 29/05/20	IS;PG
63	Análisis de los resultados	lun 01/06/20	lun 15/06/20	PG
64	**Gestión del proyecto**	**mar 16/06/20**	**mar 21/07/20**	
65	Escritura del documento	mar 16/06/20	mar 21/07/20	IS;PG

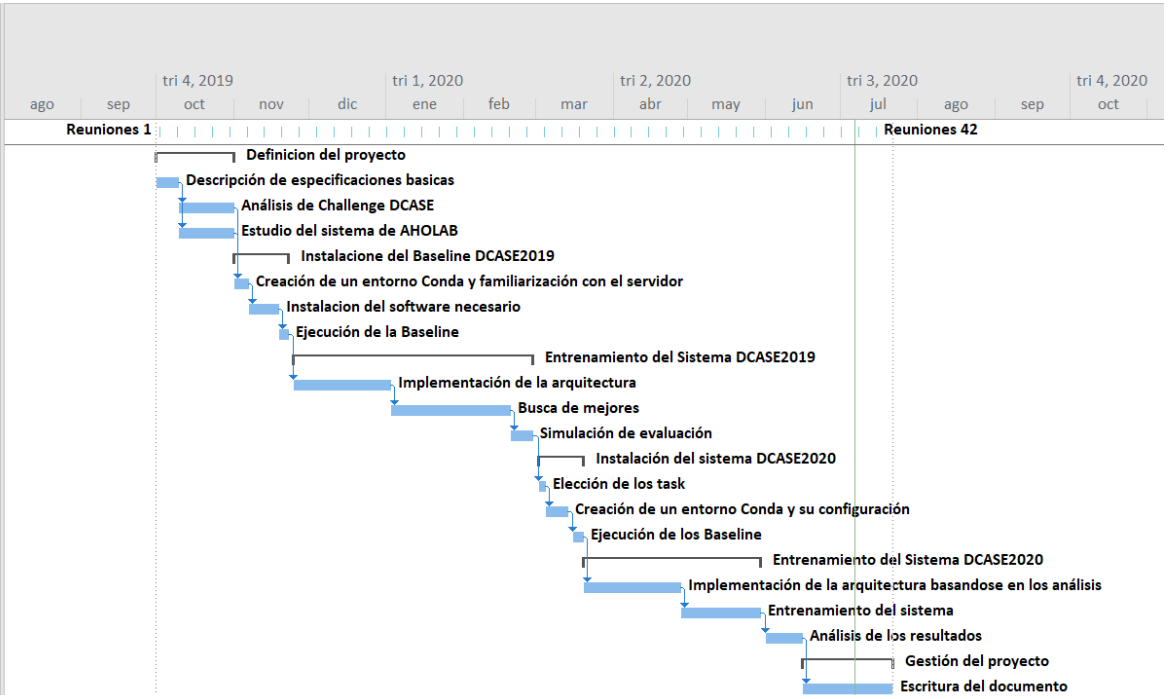

Ilustración 33: Diagrama de GANTT

11. Presupuesto

En este capítulo se gestionará el aspecto económico del proyecto. Para realizar un correcto presupuesto hay que tener en cuenta los diferentes gastos que ha habido durante el proyecto, desde recursos humanos hasta costos materiales. El presupuesto se va desgranar en diferentes secciones y la suma de cada una de ellas dará el coste total del proyecto.

Recursos humanos

En la sección de recursos humanos del presupuesto se van a tener en cuenta las personas que han trabajado en el proyecto. En el proyecto han tomado parte tres personas, dos ingenieros senior, Ibon Saratxaga e Itxasne Díez, y un ingeniero junior, Peio Gonzalez. En la siguiente tabla esta detallado el cálculo total del valor los recursos humanos, teniendo en cuenta el número de horas que ha tomado parte en el proyecto y la tasa horaria de cada uno.

	Puesto	Tasa horaria($\frac{€}{h}$)	Número de horas(h)	Total (€)
IS	Ingeniero senior	50	95	4750
ID	Ingeniera senior	50	10	500
PG	Ingeniero junior	20	440	8800
Total				14050

Tabla 48: Balance de recursos humanos

Amortizaciones

Las amortizaciones tienen en cuenta los recursos materiales que se han usado en el proyecto y que a su vez tienen una duración de vida larga. Dentro de estos recursos se pueden incluir tanto maquinaria, equipamientos, programas *software* como *hardware* especifico. Para este proyecto se han utilizado dos tipos de equipamiento que requieren ser amortizado: el ordenador portátil y el servidor de AHOLAB. Este servidor dispone de una gran capacidad de memoria y de cuatro tarjetas gráficas, por lo que no hemos sido los únicos en utilizarlo durante el transcurso del proyecto. La vida útil del servidor es de aproximadamente 10 años y se compró

por un precio de 17.200 euros. El ordenador portátil utilizado tiene una vida útil de 7 años con un coste inicial de 700 euros.

En cuanto al software utilizado, todo es de libre uso y por lo tanto gratuito, por lo que no lo incluiremos en este apartado.

	Vida útil(años)	Coste (€)	Tiempo utilizado (meses)	Utilización simultanea	Total (€)
Servidor AHOLAB	15	17.200	10	4	238
Ordenador portátil	7	700	10	1	166
Total					404

Tabla 49: Amortizaciones del proyecto

Gastos

En este apartado se incluyen las compras realizadas durante el proyecto y que han sido de uso exclusivo del proyecto, por lo que no se pueden incluir en la sección de amortizaciones. La única compra que se ha realizado para este proyecto ha sido una licencia Office de medio año para hacer una gestión más cómoda del trabajo.

	Valor (€)
Licencia Office	46
Total	46

Tabla 50: Gastos del proyecto

Otros

Para finalizar el presupuesto, se va añadir un último concepto al presupuesto llamado costes no directos que va ser el 10% de los costes directos (la suma de los anteriores apartados). A través de este apartado se representará el coste del consumo eléctrico de los aparatos, el mantenimiento de las instalaciones, la conexión a internet…

Por último, en la siguiente tabla se puede ver detallado apartado por apartado cada coste del proyecto y el valor final del mismo:

Concepto	Valor (€)
Recursos humanos	14050
Amortización	404
Gastos	56
Costes directos	14500
Costes indirectos	1450
Total	15950

Tabla 51: Presupuesto

12. Conclusiones

Mediante este documento se han podido observar los procedimientos para realizar el proyecto, que se puede dividir en dos grandes secciones. La primera relacionada con la implementación de la arquitectura de AHOLAB a la clasificación de sonidos urbanos, y la segunda con el entrenamiento y la busca de mejora del sistema. Al termino de cada apartado se han recogido los resultados obtenidos en él, para luego hacer un análisis detallado, y terminar sacando unas conclusiones sobre ellos.

En la primera parte del proyecto, en la que se debía implementar la arquitectura de AHOLAB a la clasificación de sonidos ambientales, los resultados han sido positivos en lo que a la tarea 1b respecta, ya que la precisión del sistema no ha quedado muy lejos de la marcada por la *Baseline* de DCASE. La parte negativa han sido los resultados obtenidos para la tarea 1a, quedando muy lejos de los de DCASE. Pero, de todas maneras, estos resultados han demostrado que el modelo convolucional de AHOLAB sirve para clasificar sonidos urbanos, aunque necesite un ajuste de los hiperparámetros para adaptarse al nuevo campo de trabajo.

En la segunda parte del proyecto, se han hecho modificaciones a la arquitectura original con el objetivo de superar al modelo de DCASE. Tras una infinidad de pruebas, se consiguió un sistema capaz de clasificar los sonidos ambientales mejor que el del concurso, logrando una precisión del 87,7%. Esto quiere decir que se ha conseguido aplicar el modelo convolucional de AHOLAB a la clasificación de sonidos ambientales logrando unos resultados satisfactorios, aunque la arquitectura no sea totalmente idéntica. Por su parte, en el *task1a* no se ha conseguido superar la *Baseline,* pero haciendo un ajuste de los hiperpárametros hemos logrado acercarnos a ella.

En cuanto a la participación en el *Challenge*, se ha decidido no tomar parte en él, ya que en la tarea 1a no se ha conseguido el resultado esperado y en la tarea 1b, aunque se ha superado la *Baseline,* el modelo no reunía los requisitos para poder ser presentado (el tamaño del modelo era de 33,4MB).

En definitiva, gracias a este proyecto se ha conseguido una formación más profunda en el mundo del aprendizaje automático y se ha podido evolucionar con el modelo de AHOLAB aplicándolo a nuevos entornos. En un futuro, se podría continuar con el desarrollo de esta arquitectura, mejorándola para obtener una precisión mayor o modificándola para implementarla en otros campos de trabajo.

13. Anexos

Anexo1: Resultados detallados task1b

En este primer anexo se va poder ver con detalle el mejor resultado obtenido en el análisis de cada hiperparametro para el task1b. Se va a especificar el porcentaje de acierto para cada clase.

60mel

Escena	Precisión	Pérdidas
Indoor	72,7%	5,793
Outdoor	88,7%	2,739
Transportat	89,6%	1,609
----------	----------	----------
Precisión total	83,7%	3,339

Tabla 52: Resultados detallados con 60 mel

Kernel: 56x6

Escena	Precisión	Pérdidas
Indoor	78,3%	7,095
Outdoor	85,5%	4,297
Transportat	91,6%	2,669
----------	----------	----------
Precisión total	85,1%	4,818

Tabla 53: Resultados detallados con kernel 56x6

Filtros: 80 y 65

Escena	Precisión	Pérdidas
Indoor	77.7%	6,929
Outdoor	88,5%	4,297
Transportat	91,6%	2,669
----------	----------	----------
Precisión total	85,2%	4,818

Tabla 54: Resultados con filtros 80 y 64

Units: 100

Escena	Precisión	Pérdidas
Indoor	79.7%	1,42
Outdoor	88,6%	0,582
Transportat	87,9%	1,182
----------	----------	----------
Precisión total	85,4%	1,026

Tabla 55: Resultados con 100 units

Enventanado: 0,2 – Salto enventanado: 0,1 – dropout: 0,5

Escena	Precisión	Pérdidas
Indoor	82,2%	3,112
Outdoor	89,3%	2,277
Transportat	89,5%	0,778
----------	----------	----------
Precisión total	87%	2,076

Tabla 56: Resultados con envnetanado reducido y dropout 0,5

Capas convolucionales: 5 – Capas totalmente conectadas: 2

Escena	Precisión	Pérdidas
Indoor	83,5%	0,856
Outdoor	90,6%	0,51
Transportat	89,2%	0,75
----------	----------	----------
Precisión total	87,8%	0,691

Tabla 57: Resultados 5 capas convolucionales y 2 totalmente conectadas

Peso del sistema:	33,34MB

Tabla 58: Peso del sistema final

Anexo 2: Resultado detallado task1a

En este anexo se va poder observar más detalladamente el mejor resultado obtenido para el task1a. En la tabla aparece la precisión que ha tenido cada dispositivo (tanto los reales como los simulados) para cada clase.

Escena	Precisión	A	B	C	S1	S2	S3	S4	S5	S6	Pérdidas
----------	------	-----	-----	-----	-----	-----	-----	-----	-----	-----	-------
airport	28.6%	57.6	27.3	60.6	27.3	36.4	27.3	12.1	9.1	0.0	5.279
bus	28.6%	54.5	30.3	33.3	21.2	33.3	30.3	21.2	18.2	15.2	5.027
metro	57.9%	75.8	60.6	51.5	45.5	57.6	54.5	69.7	51.5	54.5	1.776
metro_station	36.0%	42.4	51.5	21.2	39.4	30.3	39.4	30.3	33.3	36.4	2.950
park	53.9%	87.9	84.8	97.0	42.4	60.6	48.5	18.2	42.4	3.0	3.117
public_square	21.9%	42.4	36.4	30.3	18.2	18.2	30.3	6.1	3.0	12.1	5.163
shopping_mall	52.9%	57.6	63.6	57.6	54.5	63.6	42.4	39.4	42.4	54.5	2.439
street_pede..	37.0%	51.5	51.5	51.5	39.4	51.5	30.3	15.2	15.2	27.3	4.513
street_traf..	67.3%	90.9	69.7	75.8	75.8	63.6	69.7	54.5	48.5	57.6	2.101
tram	46.5%	60.6	45.5	51.5	54.5	72.7	51.5	30.3	30.3	21.2	2.997
----------	------	-----	-----	-----	-----	-----	-----	-----	-----	-----	-------
Precisión total	43.1%	62.1	52.1	53.0	41.8	48.8	42.4	29.7	29.4	28.2	3.536

Tabla 59: Resultado final task1a

14. Bibliografia

[1] S. Chachada and C.-C. J. Kuo, "Environmental sound recognition: A survey," *2013 Asia-Pacific Signal Inf. Process. Assoc. Annu. Summit Conf.*, no. March 2016, pp. 1–9, 2013, doi: 10.1109/APSIPA.2013.6694338.

[2] D. O. Hebb, *The Organization of Behavior: A Neuropsychological Theory*. .

[3] J. A. Anderson, J. W. Silverstein, S. A. Ritz, and R. S. Jones, "Distinctive features, categorical perception, and probability learning: Some applications of a neural model," *Psychol. Rev.*, vol. 84, no. 5, pp. 413–451, Sep. 1977, doi: 10.1037/0033-295X.84.5.413.

[4] J. J. Hopfield and D. W. Tank, "'Neural' computation of decisions in optimization problems," *Biol. Cybern.*, vol. 52, no. 3, pp. 141–152, 1985, doi: 10.1007/BF00339943.

[5] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks." Accessed: Jul. 15, 2020. [Online]. Available: http://code.google.com/p/cuda-convnet/.

[6] E. Çakir, "Multilabel Sound Event Classification with Neural Networks," Tampere University of Technology, 2014.

[7] K. J. Piczak, "Environmental Sound Classification with Convolutional Neural Networks," *2015 IEEE Int. Work. Mach. Learn. Signal Process.*, 2015.

[8] K. J. Piczak, "The Details That Matter: Frequency Resolution of Spectrograms in Acoustic Scene Classification," in *DCASE 2017-Workshop on Detection and Classification of Acoustic Scenes and Events*, 2017, no. November, Accessed: Nov. 22, 2017. [Online]. Available: http://www.cs.tut.fi/sgn/arg/dcase2017/documents/challenge_technical_reports/DCASE2017_Piczak_208.pdf.

[9] D. Stowell, D. Giannoulis, E. Benetos, M. Lagrange, and M. D. Plumbley, "Detection and Classification of Acoustic Scenes and Events," *IEEE Trans. Multimed.*, vol. 17, no. 10, pp. 1733–1746, Oct. 2015, doi: 10.1109/TMM.2015.2428998.

[10] J. Salamon and J. P. Bello, "Unsupervised feature learning for urban sound classification," in *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, Apr. 2015, vol. 2015-Augus, pp. 171–175, doi: 10.1109/ICASSP.2015.7177954.

[11] J. Salamon and J. P. Bello, "Deep Convolutional Neural Networks and Data Augmentation for Environmental Sound Classification," *IEEE Signal Process. Lett.*, vol. 24, no. 3, pp. 279–283, Aug. 2017, doi: 10.1109/LSP.2017.2657381.

[12] E. Cakir, G. Parascandolo, T. Heittola, H. Huttunen, and T. Virtanen, "Convolutional Recurrent Neural Networks for Polyphonic Sound Event Detection," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 25, no. 6, pp. 1291–1303, Jun. 2017, doi:

10.1109/TASLP.2017.2690575.

[13] Y. Xu *et al.*, "Unsupervised Feature Learning Based on Deep Models for Environmental Audio Tagging," 2016, doi: 10.1109/TASLP.2017.2690563.

[14] I. S. Couceiro, P. Agregado, and E. H. Unibertsitatea, "Creación de una herramienta de detección y clasificación de sonidos para vehículos," pp. 1–68, 2019.