

#### Máster en Big Data

## Asignatura: Casos de analítica Curso: 2021/22

## Fecha: 13/06/2022

Apellidos: García Layunta Nombre: Xavier Login xavier.glayunta

Apellidos: Gómez Fernández Nombre: David Login david.gfernandez

**Caso 3: UrbanSound 8K**

En este caso vamos a clasificar eventos acústicos de la ciudad de Nueva York. La contaminación acústica en entornos urbanos es un problema en auge que las administraciones públicas quieren solventar. Por esto, desde la NYU se ha confeccionado un dataset con más de 8000 ficheros de audio de 10 fuentes de ruido distintas para el desarrollo de modelos de clasificación automática (https://urbansounddataset.weebly.com/urbansound8k.html). Con estos datos, se os propone que extraigáis características acústicas de la señal (el espectrograma) i que utilicéis técnicas de deep learning para la clasificación de eventos acústicos. Explicitad y detallad todos los pasos hechos para responder a cada pregunta.

Primera Parte: Análisis Cuantitativo.

El archivo UrbanSound8k.csv contiene información de metadatos de cada archivo de audio del conjunto de datos. Esto incluye:

***slice\_file\_name:*** El nombre del archivo de audio. El nombre toma el siguiente formato: [fsID]-[classID]-[occurrenceID]-[sliceID].wav, donde:

[fsID] = el ID de Freesound de la grabación de la que se tomó este extracto (slice)

[classID] = un identificador numérico de la clase de sonido (se explica más abajo este identificador)

[occurrenceID] = un identificador numérico para distinguir diferentes "sucesos" del sonido dentro de la grabación original

[sliceID] = un identificador numérico para distinguir diferentes porciones tomadas del mismo “audio"

ejemplo: filename = '/content/drive/MyDrive/Colab Notebooks/UrbanSound8K/audio/fold1/46918-5-0-0.wav'

***fsID***: (identificador de la grabación origen del extracto). El ID de Freesound de la grabación de la que se tomó este extracto (slice), como se ha visto anteriormente

**start:** La hora de inicio del segmento en la grabación original de Freesound

***end:*** La hora de finalización del corte en la grabación original de Freesound

***salience:*** Una calificación (subjetiva) de prominencia del sonido: 1 = primer plano, 2 = fondo.

***fold:*** El número de carpeta (de 1 a 10) en el que está almacenado el archivo.

***classID***: identificador de clase explicado anteriormente. Se trata de un identificador numérico por cada una de las clases de sonido, con estas concordancias:

0 = aire acondicionado

1 = bocina de automóvil

2 = niños jugando

3 = ladrido de perro

4 = perforación

5 = motor al ralentí

6 = disparo de pistola

7 = martillo neumático

8 = sirena

9 = música callejera

***class:*** El nombre de la clase (como se ha visto en el apartado anterior): aire acondicionado, bocina de automóvil, niños jugando, ladrido de perro, perforación, motor en ralentí, disparo, martillo neumático, sirena y música callejera.

* 1. **Primer examen preliminar del dataset. ¿Qué clases componen dataset?**

Como se ha explicado en el punto anterior, con la explicación del metadato que viene asociado a la misma fuente de datos, vemos que hay 10 tipos de clase de sonido:

0 = aire acondicionado

1 = bocina de automóvil

2 = niños jugando

3 = ladrido de perro

4 = perforación

5 = motor al ralentí

6 = disparo de pistola

7 = martillo neumático

8 = sirena

9 = música callejera

Además, vienen organizados en 10 carpetas diferentes (fold 1 a fold 10) sin que haya ninguna correspondencia entre las clases y el número de carpeta en el que están guardados.

A continuación, se muestran alguna de las características previas que hemos estudiado del dataset, de cara plantear el resto de las cuestiones del ejercicio.

|  |
| --- |
| Gráfico, Gráfico de barras  Descripción generada automáticamente |
| Imagen: Nº de audios por tipo. |

Vemos como de todas las clases el número de audios es muy similar, excepto para las clases “*car\_horn*” y “*gun\_shot*” que son menos de la mitad.

Por otro lado, ploteamos el resto de columnas del dataset, por si hubiera alguna característica interesante.

|  |
| --- |
| Gráfico, Gráfico de rectángulos  Descripción generada automáticamente |
| Imagen: Duración de los audios en cada carpeta |

Donde ya se aprecia que los audios están homogéneamente distribuidos por todas las capetas, y de nuevo, que hay dos clases de las que el número de muestras es menor.

* 1. **Mostrad en una tabla u histograma el número de ficheros totales y tiempo por clase de cada fold del dataset.**

|  |  |
| --- | --- |
| Texto  Descripción generada automáticamente |  |
| Tabla: Nº de audios de cada clase | Tabla: Duración total agrupada por cada clase de audio |
|  | |
| Imagen: Duración de los audios por tipo de dato. | |
|  | |
| Imagen: Duración de los audios en cada carpeta | |

Podemos observar que el número de muestras es homogéneo para todas las clases exceptuando “*car\_horn*” (claxon) y “*gun\_shot*” (disparo) que son considerablemente menores que el resto (1000 para todas ella excepto para “sirena” que se queda en 929).

La duración de las muestras por carpetas está muy balanceada, todas ellas tienen un reparto de duración de las muestras total muy equilibrado.

La mayor parte de las muestras duran 4 segundos, aunque hay (pocas) muestras repartidas entre <0 y <4 seg. Esto mismo se da por clases (todas las clases ocurre esto) y por carpetas.

* 1. **Ahora sumad el número de ficheros y de tiempo por clase de todo el dataset y razonad si éste está balanceado**

|  |  |
| --- | --- |
|  |  |
| Tabla: Nº de audios de cada clase | Tabla: total de la duración de los audios de cada clase |

Vemos que tanto que el número de muestras por clase, como la suma de las duraciones de los audios para cada clase se cumple lo mismo. Es homogéneo (está balanceado) excepto para las clases “*car\_horn*” (claxon) y “*gun\_shot*” (disparo) donde se observa que son considerablemente menores que el resto de las clases que sí están balanceadas.

Segunda Parte: Extracción de Características y Análisis Cualitativo.

Al trabajar con señales de audio, es muy frecuente que un primer paso sea extraer características frecuenciales (frecuencia máxima, frecuencia mínima, MFCC, …), temporales (ZCR, envolvente acústica, …) o espectro-temporales (espectrograma, espectrograma mel, …) de las señales. Para solventar este caso, vamos a utilizar los espectrogramas de la señal como características para poder clasificarlas. Podéis utilizar o bien los espectrogramas o bien los espectrogramas mel de la señal.

Para esta parte hemos realizado el tratamiento de los datasets de partida y la programación en Python necesaria para poder contestar a las preguntas.

* 1. **Mostrad un espectrograma o espectrograma mel de cada clase del dataset.**

|  |
| --- |
|  |
| Representación frecuencial por clase |

|  |
| --- |
| La fachada de un local comercial  Descripción generada automáticamente con confianza baja |
| Espectrograma por clase |
| Estante de madera  Descripción generada automáticamente con confianza baja |
| Espectrograma “Mel” por clase |

* 1. **¿Qué rango frecuencial tiene el espectrograma? ¿Y temporal? ¿Cómo vais a abordar el hecho de que hay ficheros de audio en el dataset con distintas duraciones?**

Una vez obtenidos los espectrogramas es fácil obtener esos datos:

-El Rango temporal, es entre 0 y 4 seg.

-En frecuencia, varía entre lo 0 y los -80 dB.

|  |
| --- |
| Interfaz de usuario gráfica, Texto, Aplicación  Descripción generada automáticamente |
| Imagen: Rangos frecuencial y temporal del espectrograma |

Utilizando la función de librosa: *librosa.load* (audio), es decir, dándole un audio, devuelve los valores “y” y “sr” (muestras y sapmple rate (muestras/ seg)).

La duración del auido se obtiene como y/sr. Se realiza el cálculo y obtenemos que para que el audio sea de 4seg, el valor de y (el número de muestras, teniendo en cuenta la frecuencia de muestreo) ha de ser y=88200.

Se realiza la programación necesaria en el código, concatenando los audios más cortos sobre sí mismos, para obtener audios de 4 seg, en todos los casos.

|  |
| --- |
|  |
| Imagen: Concatenación de audios, para que todos sean de 4 seg. |

* 1. **Generad los espectrogramas para todo el dataset y guardároslo en disco. Se recomienda el uso de ficheros HDF5. Tened en cuenta que para la clasificación vamos a utilizar redes neuronales que esperan entradas de 224x224**

Para resolver este problema, y tras documentarnos, hemos decidido partir del espectograma-mel para resolver este y el resto de las cuestiones.

Partimos del tipo de representación de los audios posibles:

-**Espectrograma:** Es la transformación más conocida. Se basa en la aplicación de la transformada Fourier (FFT) en cada instante de tiempo.

-**Logaritmo del espectrograma en el eje** y. Se basa en la idea de dar más resolución a las frecuencias bajas

-**Espectograma** de Mel: Transformación similar a calcular el logaritmo del espectrograma, dando más peso a las que oye el odio humano.

-**Escalograma**: Transformación basada en Wavelets en lugar de Fourier.

-**MFCC**s: Coeﬁcientes Cepstrales en las Frecuencias de Mel.

Para revolver el ejercicio utilizamos la función *librosa* que nos permite obtener esos parámetros. Así podríamos obtener:

|  |
| --- |
|  |
| Imagen: Rangos frecuencial y temporal del espectrograma |

Se observa que llamando a la función *librosa.x*, y dándole los valores “y” y “sr” (mestras y sapmple rate (muestras/ seg)), obtenidos mediante *librosa.load* de un adio .wav, obtenemos diferentes parámetros que pueden ser usados como *features* para una clasificación posterior.

Además, esta función devuelve los parámetros: *chorma\_stft*, *choma\_cq* y *chroma\_cens*, aunque nosotros no los utilizaremos, si no que haremos uso del espectograma-Mel para contestar a las preguntas, y más tarde, por comparación de resultados, realizaremos el mismo proceso utilizando los MFCCs.

Se muestran, a continuación, esto dos parámetros visualmente.

|  |  |
| --- | --- |
|  |  |
| Imagen: Mel espectograma | Imagen: Coeficientes MFFCs |

Dicho esto, se ha programado el código para obtener las features necesarias para resolver el problema desde varios puntos de vista.

Método 1: Método propuesto en el enunciado. Obteniendo como features el espectograma de cada audio. Reduciéndolo a una matriz 224x224, para que sirva de alimentación a una Red neuronal de clasificación de imágenes.

Método 2: Obtener como features el espectograma de cada audio, pero no tratándolo como una matriz, sino como un array, y además construyendo nuestra propia red neuronal de clasificación “a medida”, teniendo en cuanta las dimensiones de nuestras features, y las 10 clases que queremos clasificar.

Método 3: Igual que en el método 2, pero utilizando como features los coeficientes MFFCs, en lugar del espectograma.

Se muestra a continuación, de manera gráfica y a modo resumen, cómo se ha planteado la extracción de features en cada método y los resultados obtenidos.

|  |
| --- |
|  |
| Imagen: Extracción de features y del classID. Método 1 |
|  |
| Imagen: Extracción de features y del classID. Método 2 |
|  |
| Imagen: Extracción de features y del classID. Método 3 |

|  |  |
| --- | --- |
| Interfaz de usuario gráfica  Descripción generada automáticamente |  |
| Imagen: Resultados obtenidos (previo al uso de una red neuronal). Método 1 | Imagen: Resultados obtenidos (previo al uso de una red neuronal). Método 2 |
|  |  |
| Imagen: Resultados obtenidos (previo al uso de una red neuronal). Método 3 |  |

Tercera Parte: Análisis Predictivo.

Ahora vais a clasificar los espectrogramas. Para hacerlo, debéis utilizar el método de *fine-tuning* utilizando una red neuronal profunda. Debéis tener en cuenta que típicamente estas redes han sido entrenadas con imágenes con canales R, G y B, pero que vuestros espectrogramas únicamente nos proporcionan información en cuanto a variaciones temporales (eje horizontal), variaciones frecuenciales (eje vertical) e intensidad. Por lo tanto, no tiene sentido utilizar el color del *plot* del espectrograma como característica. Siendo así, una posible solución es utilizar el espectrograma replicado tres veces para tener las tres capas (R, G y B) que espera como entrada la red neuronal. Os recomendamos que utilicéis la librería *pytorch* para solventar el problema. Podéis utilizar el siguiente enlace como referencia: <https://pytorch.org/tutorials/beginner/finetuning_torchvision_models_tutorial.html>

* 1. **Seleccionad como mínimo una red neuronal y explicad su arquitectura (número de capas, estructura de cada capa, etc.).**

Siguiendo el código se ha usado la siguiente nomenclatura:

Método 1: Código programado en clase del que no hemos hecho uso.

Método 2: Red neuronal programada, utilizando como features el espectograma Mel.

Método 3: Red neuronal programada, utilizando como features los coeficientes MFFCs.

A partir de ahora, sólo haremos mención a los métodos 2 y 3, porque son de los que hemos obtenido valores satisfactorios.

Método 2:

Se plantea una arquitectura, compuesta por:

* Capa Convolucional, CNN 2D con 64 unidades y activación *relu.*
* Capa MaxPool2D con ventana 2\*2.
* Capa Convolucional l, CNN 2D con 128 unidades y activación *relu.*
* Capa MaxPool2D con ventana 2\*2.
* Capa “Dense” DL con 1024 unidades y activación *relu.*
* Capa “Dense” DL de 10 unidades con activación *softmax*.
* Capa de salida de 10 nodos, que coinciden con el número de clasificaciones posibles.

La capa de salida con *drop probability* de 0,2.

* Optimizador de Adam con función de pérdida probabilidad categorical\_crossentropy
* Se han utilizado 90 epochs

Método 3:

Se plantea una arquitectura, compuesta por:

* Capa de entrada con 40 nodos, ya que la función MFCC de extracción de características nos devuelve un conjunto de datos de 1×40
* Capas ocultas de 256 nodos, estas capas tendrán una capa densa con una función de activación de tipo ReLu, (se ha demostrado que esta función de activación funciona bien en redes neuronales).
* Aplicaremos un valor de Dropout del 50% en nuestras dos primeras capas. Buscando como resultado una red que sea capaz de responder mejor a la generalización y que sea menos probable que se produzca sobre-ajuste en los datos de entrenamiento.
* Capa de salida de 10 nodos, que coinciden con el número de clasificaciones posibles.

La activación es para nuestra capa de salida una función softmax. Softmax hace que la salida sume 1, por lo que la salida puede interpretarse como probabilidades. El modelo hará su predicción según la opción que tenga la mayor probabilidad.

|  |  |
| --- | --- |
| Tabla  Descripción generada automáticamente |  |
| Imagen: Red Neuronal utilizada. Método 2 | Imagen: Red Neuronal utilizada. Método 3 |

* 1. **Entrenad el modelo para clasificar el dataset UrbanSound 8K**[[1]](#footnote-1)**.**

De forma gráfica:

|  |
| --- |
| Interfaz de usuario gráfica  Descripción generada automáticamente con confianza media |
| Imagen: Red Neuronal utilizada. Método 2 |
| Interfaz de usuario gráfica, Texto, Aplicación, Word  Descripción generada automáticamente |
| Imagen: Red Neuronal utilizada. Método 3 |

* 1. **Explicad qué valores de hiperparámetros principales habéis utilizado (batch size, learning rate, optimizer, número de epochs) para obtener los resultados.**

**Batch size:** Para el método con espectograma-Mel: El modelo contaba con 90 epochs,

Para el método con MFCCs: Hemos utilizado 100 epochs, pero con bastantes menos se hubieran conseguido unos resultados equivalentes.

**Learning rate:** Para el método con espectograma-Mel: El modelo contaba con 90 epochs,

Para el método con MFCCs: Hemos utilizado 100 epochs, pero con bastantes menos se hubieran conseguido unos resultados equivalentes.

**Optimizador:** Se ha utilizado Adam, en ambos casos.

**Número de epochs:** Para el método con espectograma-Mel: El modelo contaba con 90 epochs.

Para el método con MFCCs: Hemos utilizado 100 epochs, pero con bastantes menos se hubieran conseguido unos resultados equivalentes.

Como función de pérdida utilizamos categorical\_crossentropy. Esta es una de las opción más común para la clasificación

* 1. **Mostrad las gráficas de accuracy de entrenamiento y validación del modelo. Opcionalmente, también podéis mostrar las gráficas de loss.**

|  |
| --- |
|  |
| Imagen: Duración de los audios por tipo de dato. Método 2 |
|  |
| Imagen: Duración de los audios en cada carpeta. Método 2 |

|  |
| --- |
|  |
| Imagen: Duración de los audios por tipo de dato. Método 3 |
|  |
| Imagen: Duración de los audios en cada carpeta. Método 3 |

|  |  |
| --- | --- |
|  |  |
| Imagen: Precisión Método 2 | Imagen: Precisión Método 3 |

Obtenemos una “*accuracy*” (precisión) de más de 0,86 en el método del espectograma y de más del 0.88 en el método con parámetros HFCCs, para los datos de prueba. Lo que se traduciría en un 86 y un 88 por ciento de los casos respectivamente.

* 1. **Mostrad la matriz de confusión obtenida tras la clasificación. Razonad qué clases clasifica mejor o peor.**

Se ha programado el código y ploteado la matriz con los resultados para los métodos 2 y 3. Gráficamente, los datos obtenidos.

|  |  |
| --- | --- |
|  |  |
| Imagen: Matriz de confusión Método 2 | Imagen: Matriz de confusión Método 3 |

Vemos que la clasificación de ambas redes es bastante buena para todas las clases excepto para las clases “*car\_horn*” (claxon) y “*gun\_shot*” (disparo) como era de espera, debido al menor número de muestras. En general los resultados de ambas redes son equiparables.

1. [↑](#footnote-ref-1)