

**Universidad de Antioquia**

**Maestría en ingeniería**



**Fundamentos de Deep Learning**

**Informe Proyecto del Curso**

**Estudiantes:**

**Ricardo Tangarife González    1045025315**

**Fabian Stiven Duque Duque    1026149866**

**Docentes:**

**Raul Ramos Pollan**

**Julian David Arias londoño**

**Semestre 2022/1**

## **Introducción.**

Como proyecto final del curso nos planteamos elaborar modelos iniciales de machine learning que apoyen nuestros trabajos de investigación de maestría los cuales están enmarcados dentro de un proyecto que busca dar apoyo al ámbito de la seguridad ciudadana, específicamente a través de señales acústicas y visuales.

Así, buscamos generar dos modelos, uno para la clasificación de dos eventos de sonido y un modelo de detección de peleas a través de videos, que apoyen el ámbito de la seguridad ciudadana.

## **Objetivos de Machine Learning.**

Objetivo 1: Clasificar los eventos sonoros de disparos y sirenas correspondientes a eventos que apoyan el ámbito de la seguridad ciudadana.

Objetivo 2: Identificar eventos de peleas a través de videos que puedan ser útiles en generar alertas como apoyo a la seguridad ciudadana.

Para la descripción de los procedimientos, metodologías y resultados obtenidos correspondientes a cada uno de los objetivos se divide en secciones respectivas a cada uno.

### **1. Clasificación de Eventos de Sonido.**

Para la clasificación de eventos sonoros se seleccionó un dataset público de eventos acústicos urbanos nombrado UrbanSound8k, este dataset contiene 10 clases diferentes de sonidos de las cuales se eligieron 2 de interés para la seguridad ciudadana, que fueron el sonido de disparos (*gun\_shot*) y el sonido de sirenas (*siren*).

#### **1.1 Notebook.**

La clasificación de los eventos de sonido seleccionados se realiza a través del notebook nombrado en la entrega como "01 - SoundClassification.ipynb".

En este notebook se enmarcan las diferentes etapas para la solución del problema, inicialmente se realiza la carga y exploración del dataset, seguidamente de la adecuación del dataset y selección de los sonidos de interés. A continuación, se toman algunas muestras del dataset para visualización de las mismas, para posteriormente realizar una extracción de las características del sonido mediante las cuales se enfoca el entrenamiento del modelo. Como paso siguiente se realiza la adecuación de los datos para la división en conjuntos de entrenamiento y test. A partir de esto se pasa hacia la construcción del modelo. Una vez definido el modelo, se realiza el proceso de entrenamiento y de allí pasando a las métricas de evaluación del mismo.

El modelo obtenido es almacenado en memoria y posteriormente es cargado para su uso, se realizan algunas predicciones de prueba con sonidos pregrabados del dataset y como últimas pruebas se utilizan funciones para grabar sonidos directamente y ser probados en el modelo.

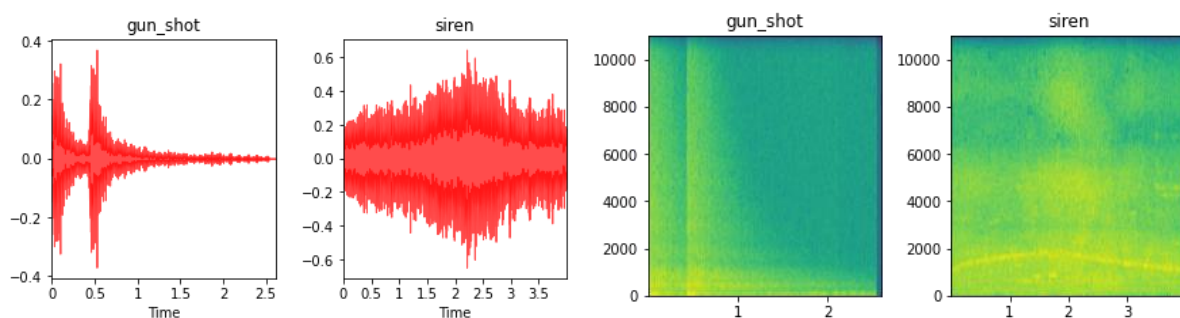
#### **1.2 Solución e Iteraciones.**

Como preprocesamiento inicialmente se seleccionan los sonidos de interés de la base de datos, pasando de tener 10 clases de la base de datos a únicamente las clases correspondientes a disparos y sirenas. En la figura 1 se visualiza algunos de los datos que contiene el dataframe.

	slice_file_name	fsID	start	end	saliency	fold	classID	class
106	102305-6-0-0.wav	102305	0.0	2.61161	1	1	6	gun_shot
114	102853-8-0-0.wav	102853	0.0	4.00000	2	7	8	siren
115	102853-8-0-1.wav	102853	0.5	4.50000	2	7	8	siren
116	102853-8-0-2.wav	102853	1.0	5.00000	2	7	8	siren
117	102853-8-0-3.wav	102853	1.5	5.50000	2	7	8	siren

**Figura 1.** Dataframe con los datos de interés.

Una vez definidos los datos a procesar, se realiza una visualización gráfica de algunas muestras de los datos, desplegando los sonidos de las clases su forma de onda en el tiempo y espectrograma de frecuencia, en la figura 2 se visualización estas salidas.



**Figura 2.** Visualización de muestras en el tiempo y la frecuencia.

Para tener una mejor adaptación del modelo a los datos, se decidió utilizar las características del sonido en el espectro de la frecuencia, ya que al tratarse en forma de imágenes los datos a procesar, las redes neuronales convolucionales permitían explotar su capacidad de abstracción de información de las mismas.

Así, se realiza la extracción de características mediante los coeficientes cepstrales en las frecuencias de Mel, que permiten la representación de la información en bandas de frecuencia presentadas en una escala logarítmica, llevando a tener la información del sonido de una forma comprimida y pudiéndose representar mediante una imagen.

Para esto se utilizó la librería librosa útil en el procesamiento de sonidos y análisis de audio. En la figura 3 se visualiza un dataframe conformado por las características extraídas para cada dato de la base de datos y su clase correspondiente.

	feature	class_label
0	[[-119.95263, -98.58099, -102.46894, -113.9573...	6
1	[[-212.37451, -203.63791, -200.84283, -208.838...	8
2	[[-196.08969, -200.47655, -211.0078, -213.0787...	8
3	[[-187.98851, -198.36465, -217.52213, -221.166...	8
4	[[-224.6572, -225.77936, -235.45703, -238.4739...	8

**Figura 3.** Dataframe con las características MFCC de cada muestra.

Una vez obtenidas las características se procedió a la división del dataset para los datos de entrenamiento y pruebas, dejando un 20% para pruebas. Y así continuar con la definición del modelo.

En este caso se definió un modelo basado en capas convolucionales de 2 dimensiones mediante las cuales se puedan extraer esos patrones de características de cada una de las muestras de las clases, representadas en los valores de intensidad de los coeficientes ceptrales en frecuencia de Mel. Así, el modelo definido como secuencial consta de 4 capas convolucionales de 2 dimensiones cada una de ellas con un maxpooling de tamaño 2x2 y a su vez un dropout de 0.2 para regularizar el entrenamiento. Después se agrega una capa de averagepooling global que nos permita contener la información en un vector para posteriormente llevarse a la capa flatten y poderse realizar finalmente la función de activación softmax en la última capa densa del número de clases. En la figura 4 se presenta la definición del modelo para mayor entendimiento.

```
model = Sequential()
model.add(Conv2D(filters=16, kernel_size=2, input_shape=(num_rows, num_columns, num_channels), activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.2))

model.add(Conv2D(filters=32, kernel_size=2, activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.2))

model.add(Conv2D(filters=64, kernel_size=2, activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.2))

model.add(Conv2D(filters=128, kernel_size=2, activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.2))
model.add(GlobalAveragePooling2D())
model.add(Flatten())
model.add(Dense(num_labels, activation='softmax'))
```

**Figura 4.** Definición del modelo de redes neuronales.

Finalmente se compila el modelo con una función de pérdida de cros entropía categórica y con optimizador Adam.

Para este modelo las diferentes iteraciones que se realizaron fueron a través de la variación del número de filtros empleados para cada capa convolucional, de igual manera variando el porcentaje de dropout en cada capa de salida. Otro de los parámetros que fueron considerados a través de las iteraciones era el número de épocas de entrenamiento del modelo, esto con el fin de obtener mejores resultados.

### 1.3 Resultados.

Posterior al entrenamiento del modelo se obtuvo unos buenos resultados al evaluar el modelo. Inicialmente evaluando la métrica de accuracy en training y en testing se obtuvieron los resultados mostrados en la figura 5.

```
Training Accuracy:  0.9856045842170715
Testing Accuracy:   0.9961685538291931
```

**Figura 5.** Medición de Accuracy en el modelo.

Para corroborar los resultados se evaluaron otras métricas en el modelo, que dan información de qué tan robusto del modelo. Estos resultados se presentan en la figura 6.

```
Confusion Matrix :
```

```
[[ 75  0]
 [ 1 185]]
```

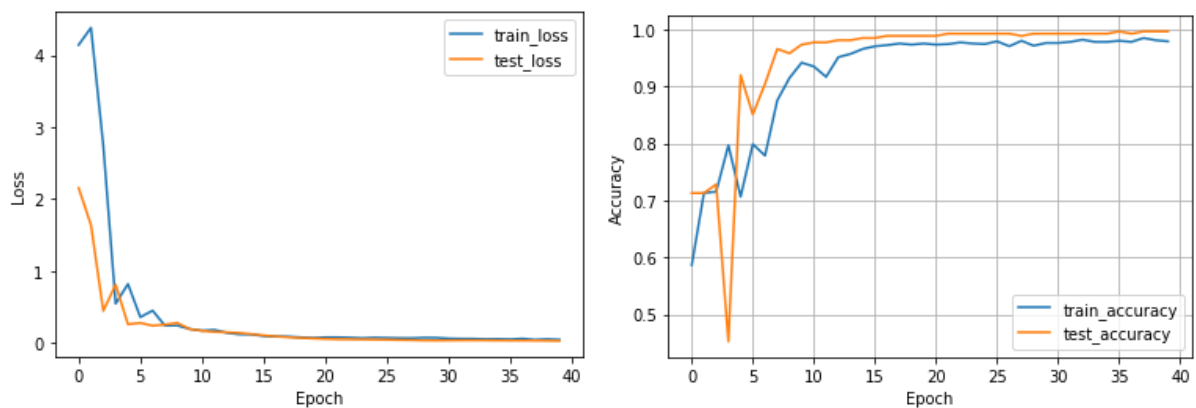
```
Classification Report :
```

	precision	recall	f1-score	support
0	0.99	1.00	0.99	75
1	1.00	0.99	1.00	186
accuracy			1.00	261
macro avg	0.99	1.00	1.00	261
weighted avg	1.00	1.00	1.00	261

**Figura 6.** Métricas evaluación del modelo.

Estos resultados nos dan evidencia de una buena clasificación en testing del modelo, se puede ver una mayor clasificación en el número de muestras tomadas para la clase 1 correspondiente al sonido de sirenas, lo cual puede mostrar una tendencia de clasificación mayor hacia esa clase.

Para ver un poco del proceso de entrenamiento se grafica el histórico de la perdida y el accuracy del modelo en train y test. Son mostrados en la figura 7.



**Figura 7.** Histórico de perdida y accuracy del modelo.


Finalmente para realizar unas pruebas con el modelo resultante se definió una función de predicción en la cual se recibe la dirección del audio que se desea clasificar, este debe tener una longitud de máximo 3 segundos, con este parámetro se extraen las características del sonido para ser pasadas al modelo predictivo con la forma requerida que se definió, una vez dada la predicción se hace pasar por un valor umbral para tener una certeza de las probabilidades de salida de las clases y así imprimir si la clase predicha con cierto grado de confiabilidad.

Para realizar pruebas con datos nuevos se implementan dentro del notebook una función para grabar y guardar sonido directamente, por tanto, se realiza la prueba de dos sonidos y los resultados al llamar la función de predicción son los presentados en la figura 8. En el primer caso fue grabado un sonido de disparo y fue erróneamente clasificado, pero no fue predicha la clase gracias al umbral definido en la función de predicción. En el segundo caso fue grabado un sonido de sirena y fue correctamente clasificado.

```
#Grabar porción de audio directamente desde el colab
pathaudio3 = recordAndSave(sec=3,path='/content/drive/MyDrive/Colab Notebooks/Urban Sound Detection/sonidosPrueba/prueba2.wav')

#Se predice el audio que fue grabado anteriormente
predict(pathaudio3)

/usr/local/lib/python3.7/dist-packages/librosa/core/audio.py:165: UserWarning: PySoundFile failed. Trying audioread instead.
  warnings.warn("PySoundFile failed. Trying audioread instead.")
[5.8423918e-08 1.0000000e+00]
1
Class predicted : siren
```



**Figura 8.** Predicción de nuevos sonidos.

## 2. Clasificación de Eventos de Riñas Callejeras.

En este caso para la clasificación de eventos de riñas callejeras se seleccionó un dataset público de eventos de riñas urbanas y situaciones normales de la vida cotidiana nombrado UBI-Fights, este dataset fue muy complejo para extraer información relevante pues no contaba con una estandarización que facilitara el entrenamiento de modelos de aprendizaje automático, ya que los videos eran mezclados de diferentes fuentes, perspectivas de cámaras, duraciones de los videos y focalización de la imagen.

### 2.1 Notebook.

La clasificación de los eventos de sonido seleccionados se realiza a través de los notebooks nombrados en la entrega como "02 - VideosClassificationResNet50.ipynb" y "03 - VideosClassificationInceptionV3.ipynb".

En estos notebooks se enmarcan dos soluciones que planteamos para la clasificación de los videos, una de las primeras pruebas fue la trabajada en el notebook 03-VideosClassificationInceptionV3.ipynb, en esta implementamos la extracción de características de los videos para cada uno de los frames mediante el modelo entrenado InceptionV3, una vez extraídas las características de los frames se construyó un modelo basado en dos capas recurrentes GRU y una capa densa, con este modelo no se alcanzó a tener un entrenamiento del modelo con buenos resultados, las predicciones aún eran muy aleatorias por lo tanto se construyó la segunda versión para el clasificador de videos, ahora con el notebook nombrado 02 - VideosClassificationResNet50.ipynb. Este último fue seleccionado como modelo con mejores resultados para la clasificación de las clases de pelea o normal en los videos.

### 2.2 Solución e Iteraciones.

Es así como en el notebook seleccionado 02 - VideosClassificationResNet50.ipynb inicialmente se prepararon imágenes que correspondían a los frames de cada video para el entrenamiento y test, etiquetándolas de acuerdo a su clase de pertenencia. Una vez preparadas los frames de entrenamiento con el tamaño y etiquetado, se realiza un augmentation de cada imagen a través del imageDataGenerator de keras para obtener mejores resultados en el entrenamiento. El modelo se construye a partir del modelo base ResNet50 que nos permita extraer las características de los diferentes frames y a partir de allí se vectoriza mediante la capa flatten y se agrega una capa densa seguida con un dropout de 0.5 de regularización y finalmente la capa densa de salidas con el número de neuronas correspondiente a las etiquetas en este caso 2. El modelo base ResNet50 es utilizado como transfer learning y sus capas son configuradas como no entrenables en el modelo, en la figura 9 se observa la construcción del modelo.

```

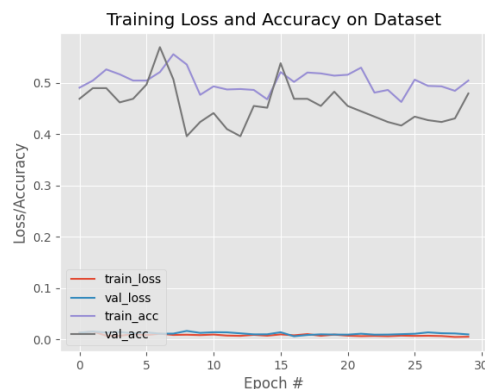
baseModel = ResNet50(weights="imagenet", include_top=False,
    input_tensor=Input(shape=(224, 224, 3)))
headModel = baseModel.output
headModel = AveragePooling2D(pool_size=(7, 7))(headModel)
headModel = Flatten(name="flatten")(headModel)
headModel = Dense(512, activation="relu")(headModel)
headModel = Dropout(0.5)(headModel)
headModel = Dense(len(lbl.classes_), activation="softmax")(headModel)
model = Model(inputs=baseModel.input, outputs=headModel)
for layer in baseModel.layers:
    layer.trainable = False

```

**Figura 9.** Definición del modelo de redes neuronales.

Por las complejidades que presentaba el dataset y demora en entrenamiento del modelo con todo el dataset, se preparó un subset de entrenamiento y test con los videos recortados de las partes de interés y con corta duración, así pudiéndose llevar a cabo un mejor entrenamiento del modelo.

Sin embargo, el entrenamiento de modelos para clasificación de video requiere mucho poder computacional y espacio de memoria disponible, factores con los que no se contaba desde el computador personal con el que realizamos los modelos, por tanto, no se realizaron muchas épocas de entrenamiento del modelo, llevando a pocas precisiones de clasificación. En la figura 10 se representa la imagen del histórico de entrenamiento del modelo para 30 épocas, como se puede ver su mejoría a lo largo de las épocas es poca, no logra extraer características importantes que logren diferenciar de manera precisa las clases de cada video en este caso a través de los frames.



**Figura 10.** Histórico de pérdida y accuracy del modelo.

Para contrastar los resultados se evaluaron otras métricas en el modelo, que dan información de qué tan robusto del modelo. Estos resultados se presentan en la figura 11.

Confusion Matrix :

```

[[101  81]
 [ 38  76]]

```

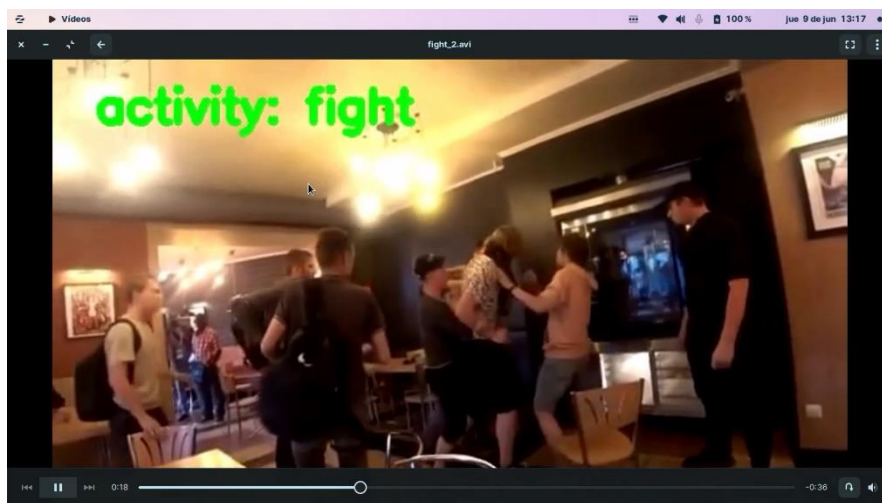
Classification Report :

	precision	recall	f1-score	support
0	0.73	0.55	0.63	182
1	0.48	0.67	0.56	114
accuracy			0.60	296
macro avg	0.61	0.61	0.60	296
weighted avg	0.63	0.60	0.60	296

**Figura 11.** Métricas evaluación del modelo.

Estos resultados nos dan evidencia de una media clasificación en testing del modelo, no se obtiene resultados sobresalientes, pero permiten tener un indicio de la clasificación de las clases correspondientes a cada frame de video. En este caso aún se podría entrenar durante más tiempo y con más muestras el modelo para verificar la obtención de una mejor clasificación.

Finalmente, para este modelo la predicción se realizó de tal manera que permite identificar utilizando los frames de cada video si pertenece a una de las clases o a otra, una vez identificada la clase perteneciente del frame se etiqueta y se agrega la etiqueta de manera visual al video. A partir de estos frames etiquetados se genera un nuevo video etiquetado visualmente con la predicción correspondiente según los frames predichos. Así como salida se entrega un nuevo video etiquetado según la clase correspondiente. Un pantallazo de un frame del video de salida etiquetado es el presentado en la figura 12, en donde se aprecia la etiqueta de actividad como fight (pelea) para un video donde pertenece a la clase de pelea.



**Figura 12.** Video de salida de predicción etiquetado.

## Disponibilidad de Datasets.

### UrbanSound8k:

Para el entrenamiento del modelo de predicción acústico se seleccionaron los sonidos del dataset UrbanSound8k.

El dataset UrbanSound8k es un dataset público con 8730 sonidos etiquetados, utilizado en diferentes tareas abiertas en kaggle, su tamaño en disco es aproximadamente 7GB y los archivos de sonido están en formato .wav; la distribución de clases es la siguiente:

0 = air_conditioner = 873	1 = car_horn = 873
2 = children_playing = 873	3 = dog_bark = 873
4 = drilling = 873	5 = engine_idling = 873
6 = gun_shot = 873	7 = jackhammer = 873
8 = siren = 873	9 = street_music = 873

Para acceder y descargar este dataset se debe dirigir a la página oficial de los contribuidores alojada en la siguiente url:

<https://urbansounddataset.weebly.com/urbansound8k.html>



## UBI-Fights:

UBI-Fights es un conjunto de datos a gran escala de 80 horas de video completamente anotado a nivel de cuadro. Consta de 1000 videos, donde 216 videos contienen un evento de lucha y 784 son situaciones normales de la vida diaria, la dimensión 640 x 360 píxeles y la velocidad de fotogramas es de 30 fps y el tamaño en disco es cercano a 7.6 GB.

El conjunto de datos cuenta con las siguientes anotaciones:

- Fight: F\_id\_environment\_camera\_color.mp4
- Normal: N\_id\_environment\_camera\_color.mp4
- Environment: Indoor (0) / Outdoor (1);
- Camera: Fixed (0) / Rotated (1) / Movable (2);
- Color: RGB (0) / Grayscale (1);

Para acceder y descargar este dataset se debe dirigir a la página oficial de los contribuidores alojada en la siguiente url:

<http://socia-lab.di.ubi.pt/EventDetection/>

## Referencias.

Fundamentos de Deep Learning. (2022). Retrieved 6 June 2022, from <https://rramosp.github.io/2021.deeplearning/intro.html>

UrbanSound8K. (2022). Retrieved 6 June 2022, from <https://urbansounddataset.weebly.com/urbansound8k.html>

UrbanSound8K Kaggle. (2022). Retrieved 6 June 2022, from <https://www.kaggle.com/datasets/chrisfilo/urbansound8k>

UBI Event Detection. (2022). Retrieved 6 June 2022, from <http://socia-lab.di.ubi.pt/EventDetection>

Iterative weak/self-supervised classification framework for abnormal events detection. (2022). Retrieved 6 June 2022, from [https://www.di.ubi.pt/~hugomcp/doc/Events\\_PRL.pdf](https://www.di.ubi.pt/~hugomcp/doc/Events_PRL.pdf)

Selección de métricas para los modelos de aprendizaje automático | Fayrix. (2022). Retrieved 6 June 2022, from [https://fayrix.com/machine-learning-metrics\\_es](https://fayrix.com/machine-learning-metrics_es)