Actividades adicionales

Para esta práctica requerirá recursos computacionales mayores que con todos los modelos vistos. Recomendamos que utilice algún servicio online que provea GPUs para realizar las pruebas.

**Ejercicio 1**. Clasificación de imágenes de radiografías de tórax (carpeta pneumonia)  
  
El conjunto de datos contiene imágenes de pacientes con radiografías normales y con neumonía. Las imágenes de neumonía contienen tanto pacientes con neumonía “normal” como con neumonía por COVID19. Su objetivo es entrenar un modelo con el F-Score (o F-measure o F1) más alto en el conjunto de test.

1. Entrene un modelo que obtenga al menos un f-score de 0.97 en test.
2. El conjunto de datos está desbalanceado, así que deberá compensar para eso utilizando pesos por clase (parámetro **class\_weight** del método **fit**). Utilice los pesos por clase para mejorar el F-score.
3. Utilice un modelo pre-entrenado como MobileNet o ResNet18 para aumentar el desempeño.
4. Diseñe una estrategia de aumentación de datos para aumentar el desempeño de las red. Estudie las imágenes de entrada y considere qué transformaciones pueden ser útiles para ese dominio.

**Ejercicio 2**. Clasificación de muestras de audio (carpeta Perros y gatos).

Este dataset contiene audios (archivos wav) de ladridos de perros (clase 0) y maullidos de gatos (clase 1) de distintas duraciones. El objetivo es utilizar las muestras de *train* para entrenar un modelo el cual permita conseguir el f-score más alto en el conjunto de *test*.

Se provee un script de ejemplo de cómo leer los archivos de audio y prepararlos para conformar el dataset a utilizar en la fase de entrenamiento. Un archivo de audio puede verse como un vector unidimensional de valores de la señal de audio, donde la cantidad de valores de este vector depende de varios factores como la frecuencia de muestreo y la longitud del audio.

Para normalizar la cantidad de entradas puede intentar varias pruebas:

1. a partir de un archivo de audio tomar varias porciones del vector, todas de la misma longitud.
2. extraer características espectrales de los audios a través de distintas utilidades que se detallan en el notebook asociado.
3. En el caso de generar más de un vector de features por archivo de audio, luego deberá tener en cuenta que para el testeo deberá hacer lo mismo y como con cada feature del mismo vector podría tener distintas respuestas del modelo, deberá implementar un mecanismo de votación para determinar la clase a la cual corresponde cada audio testeado.

**Ejercicio 3**. Clasificación de textos (carpeta Pizza request).

Este dataset contiene pedidos (texto escrito en lenguaje natural en inglés) de usuarios solicitando se les regale una pizza. Los pedidos los hacen los usuarios de una red social a otros usuarios de la misma red social explicando el motivo por el cual deberían regalarle una pizza. El dataset posee además del pedido original por el usuario si el regalo fue realizado (clase 1) o no (clase 0). El objetivo es utilizar las muestras de *train* para entrenar un modelo el cual permita conseguir el f-score más alto en el conjunto de *test*.

Se provee un script de ejemplo de cómo procesar los pedidos de los usuarios y prepararlos para conformar el dataset a utilizar en la fase de entrenamiento.

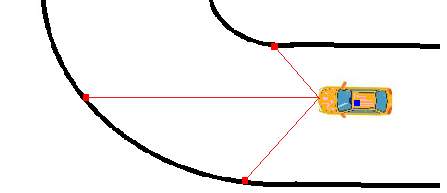
Para normalizar la cantidad de entradas puede realizar lo que desee: filtrado de stop-words, lematización, stemming, etc.

**Ejercicio 4**. Simulación de conducción de un vehículo (carpeta Conduccion de auto).

En este ejercicio no hay un dataset, sino que debe crearse utilizando el simulador provisto. Para ello deben establecerse la cantidad y dirección de los sensores de proximidad del vehículo.

El valor de 0 indica el frente del auto, un valor negativo indica la rotación del sensor hacia la izquierda y un valor positivo indica la rotación hacia la derecha. Ejemplo con tres sensores:

sensores = [-45, 0, 45]



Para la generación del dataset se debe ejecutar la función conduccionManual, la cual recibe los sensores definidos y la imagen con el circuito que se debe simular:

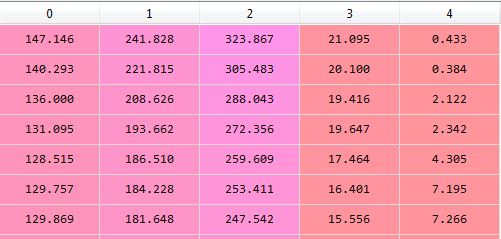
simulador.conduccionManual(sensores, 'Circuito1.jpg')

Los circuitos son imágenes en blanco y negro, donde los bordes negros indican las “paredes” del circuito, el cual se debe evitar chocar. Los sensores buscan las paredes y dan una medida de distancia del sensor a la pared.

Para simular el movimiento del vehículo, primero se debe hacer click dentro del circuito, donde se desea iniciar la simulación (tener en cuenta que el vehículo siempre aparecerá “mirando” a la izquierda. Luego el vehículo seguirá el puntero del mouse (hay que moverlo continuamente para generar los eventos de movimiento; no funciona dejando el mouse quieto). Por cada movimiento se generará una fila con las distancia detectadas por todos los sensores, junto con la rotación del vehículo (si se detecta que se debe girar para seguir al mouse) y el “paso” que se da hacia adelante. Por cada movimiento se genera una muestra de sensores-respuestas\_del\_vehículo. Así, al finalizar la simulación (haciendo click) se contará con un dataset de la forma:

<sensor 1, sensor 2, …., sensor n, avance\_vehículo, rotación\_vehículo>

Ejemplo con tres sensores:



El dataset se puede obtener con la función obtenerDatosDeSensores:

dataset = simulador.obtenerDatosDeSensores()

Con ese dataset se debe entrenar un modelo ya que el objetivo es que el vehículo aprenda a conducir solo. Esto es, el vehículo sensará las distancias y consultará con el modelo para saber cuánto avanzar y cuánto girar en cada “paso” de la simulación.

Luego de entrenado el modelo se debe proveer de una función que simulará al conductor, la cual recibirá la distancias censadas por todos los sensores (variables independientes) y se debe devolver dos valores (un valor que indique cuanto avanzar y otro que indique cuanto girar (variables a predecir))

def fConductor(distancias):

# consulta con el modelo entrenado

return (cuanto\_avanzar, cuanto\_girar)

Para testear lo aprendido por le modelo se debe invocar a la función conduccionAutonoma, la cual debe recibir los sensores, el circuito y la función que simula al conductor:

simulador.conduccionAutonoma(sensores, circuito, fConductor)

Se provee un script de ejemplo de cómo obtener el dataset y de como simular la conducción.

ACLARACIÓN: el script no se debe ejecutar de manera completa. Primero se debe ejecutar la parte de generación de datos, luego el entrenamiento del modelo y finalmente el testing de conducción.

El objetivo es tratar de que el vehículo de la mayor cantidad de “pasos” por el circuito sin chocarse con las paredes del mismo.