# Ejercicios Prácticos – Python SSr (computer vision, deep learning)

**Autor: Nicolás Aldecoa Rodrigo**

Para resolver los siguientes ejercicios usted puede valerse de cualquier recurso que considere útil.   
Se aceptan entregas parciales en caso de que no haya podido completar toda la ejercitación.  
Recomendamos leer bien en detalle las consignas, citar sus fuentes y probar las implementaciones.   
La cantidad de líneas esperadas para cada ejercicio no tiene en cuenta comentarios ni líneas en blanco, pero usted tiene la libertad de comentar y dar formato al código de la manera que considere apropiada.

**Manejo de paquetes y ambientes virtuales**

1. Crear un ambiente virtual llamado “EntreConsultas” en [Anaconda](https://www.anaconda.com/) o python3-venv en linux, con Python 3.7 e instalar las siguientes librerías: Numpy (1.16.3), OpenCV (4.4.0). Una vez finalizada la instalación, utilizar el intérprete de python en modo interactivo para verificar que ambas librerías puedan ser importadas y que posean la versión indicada. **Listar los comandos que debió ejecutar y adjuntar una captura de pantalla del output de la verificación usando el intérprete donde pueda verse que está trabajando en el ambiente virtual previamente creado.**

**Operaciones Numéricas y Visión por Computadora**

1. **Libs permitidas: numpy únicamente. Líneas de código esperadas: menos de 10.**  
   En EntreConsultas están utilizando una CNN para hacer encoding de rostros humanos en vectores de 128 features. Cada feature es un número de punto flotante de 32 bit de precisión.   
   Dispone de una matriz de encodings conocidos en forma de un arreglo de numpy de dimensiones (100, 128) donde cada fila corresponde con un rostro distinto y cada identidad se encuentra indexada en su base de datos con un ID igual al índice de la fila que ocupa en la matriz. Puede simularlo utilizando la siguiente sentencia:   
   *knownIDs = np.random.rand(100, 128).astype(np.float32)*  
   Para determinar si dos encodings son un match, se calcula la distancia euclidea entre los vectores y si esta resulta menor o igual a un valor de corte previamente determinado (tolerancia), se considera que corresponden al mismo rostro. El encoding de un rostro puede ser simulado con:  
   *face\_encoding =* *np.random.rand(128).astype(np.float32)*  
   **Implemente utilizando métodos de numpy una función “best\_match”** que reciba como argumentos:

* la matriz de encodings conocidos (*known\_faces: np.ndarray*)
* el encoding de un rostro (*face: np.ndarray*)
* un valor de tolerancia (*tolerance: float*)

y devuelva:

* el índice del encoding conocido con menor distancia euclidea a *face* (*idx: int*)
* el valor de distancia obtenido (*dist: float*)
* True si se trata de un match o False en caso contrario (*match: bool)*

**Recomendación:** las operaciones sobre arreglos de numpy y sus reglas de broadcasting permiten resolver este problema en muy pocas líneas de código y sin la necesidad de utilizar ciclos que harían la solución mucho menos eficiente.

1. **Libs permitidas: numpy, cv2, os, glob, argparse. Líneas de código esperadas: menos de 25.**

En EntreConsultas planean entrenar un modelo de detección de objetos (Faster R-CNN) con un dataset de logos y anuncios de marcas específicas que pueden aparecer durante la transmisión de un partido de fútbol. El caso de uso consiste en desenfocar (blurear) todos los objetos detectados en colecciones de fotografías tomadas durante un partido para su posterior publicación en una página web.  
En la documentación que recibió de R&D le aseguran que a futuro usted podrá importar una clase “*BrandDetector”* e instanciarla para utilizar su método público “*detect\_all”*. Este método recibe como argumento una **imagen RGB** y devuelve una lista de tuplas con las coordenadas de las bounding boxes de todos los objetos detectados:  
[ ((x1, y1), (x2,y2)), ((x1, y1), (x2,y2)), ((x1, y1), (x2,y2)), … ], donde (x1, y1) son 2 valores enteros correspondientes a las coordenadas del vértice superior izquierdo del rectángulo y (x2, y2) el vértice inferior derecho. Si ningún objeto es detectado, el método devuelve una lista vacía.  
**Usted debe diseñar un script para probar el método que:**

* Reciba como argumento por consola la ruta a un directorio.
* Abra una a una todas las imágenes contenidas en el mismo (.jpg, .jpeg o .png).
* Detecte en cada una los logos o anuncios asumiendo que tiene acceso a la clase BrandDetector anteriormente definida.
* Genere una nueva imagen donde se haya aplicado un blur/desenfoque intenso a todas las áreas rectangulares detectadas (el resto de la imagen debe quedar intacta).
* Guarde cada imagen resultante bajo el mismo nombre en el directorio “./blurred”.

**Recomendaciones:**Consulte la documentación de opencv para decidir qué método y parámetros utilizar para que las áreas desenfocadas sean ilegibles en la imagen resultante.  
Investigue sobre el uso de máscaras en OpenCV, existen algoritmos muy simples para manipular ciertas áreas de una imagen dejando el resto intacto.  
Cree una clase para simular BrandDetector con un método detect\_all que reciba como input una imagen (arreglo tridimensional de numpy) y devuelva una lista de bounding boxes hardcodeadas para poder probar que su script funcione.

**Modelado de Datos y Deep Learning**

1. Un cliente desea optimizar su línea de producción de juguetes parlantes. Existen dos clases de defectos que estos pueden tener: de apariencia (fallos en la pintura, abolladuras, y otros daños que pueden apreciarse visualmente) y de funcionamiento (el audio que produce al estar encendido sale completamente distorsionado).  
   Hoy en día tiene empleados dedicados a examinar cada juguete individualmente para descartar los que tienen fallos, pero considera que es un gasto excesivo en personal y está buscando cómo automatizar el proceso. El cliente asegura tener en su depósito miles de juguetes descartados en el último bimestre por distintos defectos esperando a ser reciclados, y cuenta con una producción diaria de juguetes sin defectos que está en el orden de los cientos.  
   EntreConsultas está elaborando una propuesta que consiste en colocar 2 videocámaras y un micrófono en la línea de producción para que por cada juguete que pase se capture video y

audio durante 2 segundos, y en base a esos datos un algoritmo prediga si debe ser descartado.

**Haga un diagrama simple del pipeline que propondría** para obtener un output binario (aceptado/rechazado) suponiendo que su input por cada juguete serán 2 secuencias de 60 imágenes a color y 2 segundos de audio. **Especifique qué tipo de modelos/arquitecturas recomendaría** utilizar y cómo se acoplarían. **Haga un breve listado de los requerimientos que haría al cliente** para crear un dataset que le permita entrenar sus modelos y evaluar el algoritmo final.