

## Programació per a la ciència de dades - PAC4

En aquest Notebook trobareu un exercici que suposa la quarta activitat d'avaluació continuada (PAC) de l'assignatura. Aquesta PAC intenta presentar-vos un petit projecte on heu de resoldre diferents exercicis, que englobarà molts dels conceptes coberts durant l'assignatura.

L'objectiu d'aquest exercici serà desenvolupar un **paquet de Python**, fora de l'entorn de Notebooks, que ens permeti resoldre el problema donat. Treballareu en arxius plans `.py`. Aquest paquet haurà d'incloure el corresponent codi organitzat lògicament (separat en mòduls, organitzats per funcionalitat), la documentació de codi (*docstrings*) i tests. A més, s'hauran d'incloure els corresponents arxius de documentació d'alt nivell (`README`), així com els arxius de llicència i dependències (`requirements.txt`) tal i com s'explica a la teoria. Fer un `setup.py` és opcional, de la mateixa forma que incloure un informe (en format `.pdf`) amb el resum dels resultats de la pràctica, però si es fa es valorarà positivament de cara a la nota de la pràctica i del curs.

Se'ns demana que implementem un paquet (mòdul) de Python que sigui capaç de realitzar una anàlisi d'imatges de diferents ciutats europees preses entre 2015 i 2019. Per una banda tindrem les imatges i per altre els objectes presents en aquestes imatges i la seva posició dins de la imatge.

## Enunciat:

Ens han encarregat analitzar imatges de carrers de diferents ciutats europees, per un projecte relacionat amb *smart-cities*. Per tal de començar el projecte, tenim un datatset d'imatges de tres ciutats on les imatges han estat filmades des d'un cotxe des de diferents punts de la ciutat. El data set complet el podeu trobar [aquí](https://www.cityscapes-dataset.com) (<https://www.cityscapes-dataset.com>). Junt amb les imatges ens han donat també uns fitxers de text on podem trobar els tipus d'objecte que hi ha i les posicions per cada un d'ells, de tal manera que tenim **un fitxer de text per a cada imatge**. Ens indiquen que aquesta informació s'ha extret utilitzant [YOLOv5](https://docs.ultralytics.com) (<https://docs.ultralytics.com>), (YOLO, *You Only Look Once*) que és un algoritme basat en xarxes convolucionals molt potent per a la detecció d'objectes en temps real.

En aquesta PAC haureu de treballar amb aquests fitxers per tal d'analitzar les imatges i extreure'n conclusions sense haver de mirar cadascuna de les imatges. Les dades les teniu al fitxer **dataset.tar.gz** on hi trobareu:

- **images:** Carpeta que conté totes les imatges. Fixeu-vos que en el nom del fitxer hi surt la ciutat i la data en la que la fotografia ha estat presa.
- **labels:** En aquesta carpeta trobareu els arxius `.txt` amb el mateix nom base que la imatge a la que correspon. A cada arxiu hi hauran tantes línies com objectes trobats a la imatge. Per a cada objecte trobareu 6 columnes amb la següent informació:

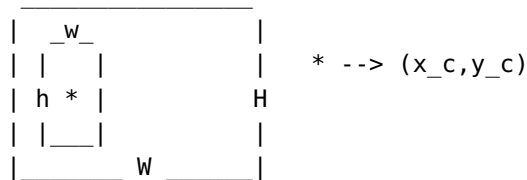
\* **\*\*identificador de l'objecte\*\***: enter entre 0 i 80, que són la quantitat d'objectes que contempla el model de YOLO (us donem una relació entre l'identificador i el tipus d'objete en un altre arxiu més avall)

\* **\*\*coordenades objecte  $x_c^n, y_c^n, w^n, h^n$ \*\***: La posició de l'objecte detectat es defineix per la *\*bounding box\**, que és el rectangle que conté l'objecte. Aquest ve definit per 4 coordenades, en aquest cas les coordenades són el valor central  $(x_c^n, y_c^n)$ , l'amplada i l'alçada  $(w^n, h^n)$ . YOLO dona tots aquests valors normalitzats, per tant es trobaran entre 0 i 1, és a dir, les coordenades horitzontals van dividides per l'amplada total de la imatge i les coordenades verticals per l'alçada total de la imatge:

$$x_c^n = \frac{x_c}{W}, y_c^n = \frac{y_c}{H}$$

$$w^n = \frac{w}{W}, h^n = \frac{h}{H}$$

on  $x_c, y_c$  és el valor central de la imatge en pixels,  $w$  i  $h$  son l'amplada i l'alçada de la *bounding box* i  $W$  i  $H$  son l'amplada i l'alçada de l'imatge total. Ho podeu veure al següent esquema:



- **confiança de la detecció**: En la última columna tenim la probabilitat que dona el model de YOLO de que la posició de l'objecte sigui la correcta.

```
9 0.760986 0.140137 0.0229492 0.104492 0.285246
58 0.960693 0.693359 0.0786133 0.210938 0.293333
9 0.928955 0.0634766 0.0405273 0.0996094 0.332471
9 0.908691 0.059082 0.0791016 0.114258 0.374223
9 0.801514 0.254395 0.0336914 0.135742 0.390878
9 0.887451 0.0537109 0.0395508 0.107422 0.554214
9 0.243896 0.267578 0.0209961 0.109375 0.591291
2 0.438232 0.438965 0.0541992 0.0478516 0.740896
2 0.753662 0.459473 0.0825195 0.100586 0.745214
2 0.530273 0.453613 0.0576172 0.0771484 0.814936
2 0.384766 0.450195 0.0634766 0.107422 0.829835
...
```

Per exemple la primera línia de l'arxiu que us presentem a dalt, tenim un objecte amb identificador igual a 9, amb coordenades normalitzades  $x_c=0.760986$ ,  $y_c=0.140137$ ,  $w=0.0229492$   $h=0.104492$  i probabilitat de que estigui ben detectar igual a 0.285246.

- **class\_name.txt**: Fitxer on trobareu la relació entre l'identificador de l'objecte i el nom.

# Presentació dels resultats: Per a fer l'entrega més fàcil i més homogènia us aconsellem que organitzeu el codi, de tal manera que **des del fitxer principal retorni les respostes del que se us demana a la PAC** cridant les funcions que haureu definit en mòduls. Per això, a cada exercici, us indicarem el format que heu de tenir de cada resposta. Si trobeu més pràctic fer-ho d'una altra manera, cal que estigui ben indicat al README per poder-ho executar sense problema. Us recordem que al README també heu d'indicar com executar els tests i comprovar la cobertura d'aquests.

## Control i revisió del dataset:

Quan comencem a treballar en un projecte d'anàlisi de dades, una pràctica important és assegurar-nos que les dades són correctes abans de posar-nos a fer cap tipus d'anàlisi. En altres paraules, cal fer una anàlisi exploratori inicial. Per fer això us proposem que feu:

### Exercici 1.

Llegir tots els fitxers i ajuntar-los en un dataframe amb les columnes que cregueu interessants per tal de resoldre la PAC.

### Exercici 1.2.

En aquest cas tenim poques imatges, però en un cas realista es podrien tenir moltíssimes imatges per analitzar. Si tinguessiu milers d'arxius amb més d'un Gb cada arxiu com ho farieu? (no cal implementar la solució, però sí justificar-la)

**Mostreu per pantalla** les primeres files del dataframe i contesteu a la pregunta 1.2 amb un print.

### Exercici 2.

De vegades, ens trobem que les dades estan corrompudes, tant pot ser per errors humans com per algun *bug* del codi. Per tal de detectar si hi ha algun fitxer no vàlid, creeu una funció `check_yolo` que prengui com a input un fitxer i retorni si té el format YOLO o no. Comproveu el nombre de columnes i les característiques de cadascuna per tal si es compleix estrictament el format descrit en l'enunciat. Si detectem fitxers que continguin alguna línia o valor incompatible amb un possible fitxer YOLO els eliminarem del dataset i treballarem sense ells durant la resta de pràctica.

Crideu la funció desde el codi principal i **Mostreu per pantalla** una llista del nom dels arxius que no segueixen el format, i si fos el cas mostrar missatge de que no se n'ha trobat cap.

### Exercici 3.

També ens agradaria comprovar que les prediccions són correctes, la manera més senzilla seria visualitzar-ho. Per això mirarem que els objectes detectats corresponen a la imatge dibuixant el rectangle sobre la imatge. Podeu fer servir funcions de la llibreria `matplotlib.pyplot` tant per carregar les imatges (`imread`), generar el rectangle (`patches.Rectangle`) i visualitzar el resultat (`imshow` i `add_patch`).

Per fer aquest exercici haureu de fer un canvi de coordenades, ja que la manera de definir el rectangle de YOLO i el `patches.Rectangle` és diferent.

YOLO (unnormalized)	<code>patches.Rectangle</code>
<pre>+-----width-----+                             height      *                         (x,y)                                  +-----+-----+-----+</pre>	<pre>+-----width-----+                             height  (x,y) *-----+-----+</pre>

Recordeu "desnormalitzar" coordenades YOLO perquè estiguin en número de píxels. Per això necessiteu la mida original de les imatges que és  $W = 2048$ ,  $H = 1024$ .

Per visualitzar la imatge podeu seguir aquest [exemple \(https://www.adamsmith.haus/python/answers/how-to-draw-a-rectangle-on-an-image-in-python\)](https://www.adamsmith.haus/python/answers/how-to-draw-a-rectangle-on-an-image-in-python)

Comprova que les *bounding boxes* estan englobant els objectes amb la primera foto (tenint en compte ordre lexicogràfic) de cada ciutat.

En la execució del codi presenteu la **visualització d'aquestes tres imatges amb els contorns dels objectes detectats**. Us deixem aquí una imatge d'exemple:





## Anàlisi de dades:

### Exercici 4.

Per aquesta part treballarem només amb els objectes que tinguin una **confiança major a 0.4** i només amb els fitxers que hagin passat el test de **candidats a YOLO**.

#### Exercici 4.1

Trobar i representar gràficament la distribució d'objectes en tot el dataset. És a dir, volem saber per cada objecte (per cada instància diferents) quants objectes s'han detectat. **Mostreu per pantalla** els identificadors i noms dels 5 objectes que apareixen més cops en tot el dataset i quants cops hi apareixen. **Mostreu una gràfica de tipus barres** amb el número total d'objectes de cada classe.

#### Exercici 4.2.

A l'apartat anterior hem trobat els 5 objectes més populars al dataset. D'aquests 5 objectes més populars ens preguntem si segueixen algun patró en la seva aparició a les imatges. És a dir, com es distribueix la seva freqüència d'aparició en una imatge (si sempre surt 3 cops a cada imatge on apareix, o el més freqüent és que surti 5 cops i el menys és que no surti, etc).

**Mostreu en un sol gràfic** les distribucions per objecte trobades per tal de comparar-les (feu servir histogrames normalitzats)

En altres paraules, volem codificar la següent informació:

L'objecte A surt 3 cops en la imatge 1.

L'objecte A surt 1 cop i objecte B surt 2 cops en la imatge 2.

L'objecte A surt 3 cops i l'objecte C surt 1 cop a la imatge 3.

#### Exercici 4.3

Quin és el nombre mig d'objectes totals per imatge? **Mostrar per pantalla** el resultat degudament explicat i formatat.

#### Exercici 4.4

Volem saber quins són els tres elements més populars per imatge. Per definir els elements més populars per imatge hem creat una casuística a seguir (que es detalla al final de l'enunciat). Us demanem:

a) **Crear una funció** que donat el dataframe us retorni un diccionari ordenat segons popularitat de l'objecte on com a claus tinguem tots els possibles labels o objectes que apareixen al dataset amb el que hem treballat des de l'exercici 3. En altres paraules, volem comptar per cada objecte quants cops ha estat entre els més populars de cada imatge. **Mostrar per pantalla** el diccionari ordenat obtingut aplicant la funció que acabem de definir al nostre dataset.

b) Coincideixen els tres elements més populars per imatge amb els més populars dins del dataset? Si responeu que no: Expliqueu perquè pot passar això. Si responeu que sí: Doneu un exemple d'un possible cas on això no passaria. **Respondre en un print** per pantalla.

---

**Nota:** Casuística per triar donada una imatge quins són els seus objectes més populars:

1. Si en té **menys de 3 objectes o exactament 3 objectes** diferents agafarem com a populars tots aquells que apareixen.
2. Si té **més de 3 objectes diferents**:
3. A. Si la freqüència més alta apareix en més de tres objectes, agafarem tots aquests objectes com a els

més populars (en aquest cas podem agafar més de tres objectes). Això inclou el cas en que tots els objectes tinguin la mateixa popularitat.

*Exemples (4 gats, 4 gossos, 4 ànecs, 4 rates, 1 colom) -> objectes més populars: (gat, gos, ànec, rata) o (1 gat, 1 gos, 1 ànec, 1 rata, 1 colom) -> objectes més populars: (gat, gos, ànec, rata, colom)*

2.2) Si no hi ha empat de popularitat entre el 3r i 4t objecte més popular agafarem els 3 objectes més populars de la imatge

2.3) Si tenim empats en popularitat entre els 3 més populars i/o el 3r i 4t més populars farem el següent:

2.3.1) Si no hi ha empat de popularitat entre el 3r i 4t objecte més popular agafarem els tres objectes més populars.

*Exemples (4 gats, 4 gossos, 4 ànecs, 2 rates, 1 colom) -> objectes més populars: (gat, gos, ànec) o (4 gats, 4 gossos, 3 ànecs, 1 rates, 1 colom) -> objectes més populars: (gat, gos, ànec)*

2.3.2) Si l'empat de popularitat es produeix entre el 3r i 4t més populars agafarem només els dos objectes més populars.

*Exemple (4 gats, 4 gossos, 3 ànecs, 3 rates, 1 colom) -> objectes més populars: (gat, gos)*

2.3.3) Si l'empat de popularitat es produeix entre el 2n i 3r més populars agafarem només l'objecte més popular.

*Exemple (4 gats, 3 gossos, 3 ànecs, 3 rates, 1 colom) -> objecte més popular: (gat)*

---

## Exercici 5.

Representa gràficament i **mostra per pantalla** el número de cotxes per any, per cada ciutat. **Representa-ho en una sola gràfica** on apareguin els resultats de totes les ciutats.

## Exercici 6.

Pregunta oberta: En el dataset de **zurich** hi ha hagut una mica de descontrol i s'han afegit algunes imatges que no són de la ciutat. Dissenya una funció que sigui capaç d'identificar les imatges que no pertanyin a la ciutat de la forma més automatitzada possible. És un **exercici lliure** on no hi ha una única manera de resoldre'l. Es valorarà la capacitat de trobar totes les imatges infiltrades i la creativitat de la resposta. Raoneu la vostra resposta i expliqueu per què ho heu fet així. **Mostreu per pantalla**, el nom del fitxer corresponent a les imatges detectades com intruses i què heu fet per trobar-les. \n",

## Exercici 7.

Guarda tota la informació en un fitxer .csv amb capçalera nom imatge, número de cotxes, número de semàfors, número de persones, ciutat, any, i si pertany o no a una ciutat (és a dir, si no és una imatge detectada com intrusa). **Mostra per pantalla**, degudament formatat, el nom de l'arxiu generat i on s'ha guardat.

Observeu que heu de generar codi que permeti **representar els resultats de l'exercici 3, 4 i 5 gràficament i per pantalla (exercicis 1,2, 4, 5, 6 i 7)**.

El codi haurà d'estar ordenat, eviteu que el fitxer principal tingui massa codi, degudament incloent-hi documentació de funcions, i correctament testejat usant la llibreria `unittest`. Els testos proporcionats hauran de donar cobertura com a mínim al 50% de la funcionalitat proposada.

## Cobertura dels testos

El mesurament de la cobertura dels testos s'utilitza per avaluar l'eficàcia dels testos proposats. En particular, serveix per determinar la qualitat dels testos desenvolupats i per determinar les parts crítiques del codi que no han estat testejades. Per tal de mesurar aquest valor, proposem l'ús de l'eina `Coverage.py` (<https://coverage.readthedocs.io/en/coverage-5.3/>). A la documentació, podreu trobar [com instal·lar-la](https://coverage.readthedocs.io/en/coverage-5.3/install.html#install) (<https://coverage.readthedocs.io/en/coverage-5.3/install.html#install>) i [com fer-la servir](#)

## Criteris de correcció

Aquesta PAC es valorarà seguint els criteris següents:

- **Funcionalitat** (5.75 punts): Es valorarà que el codi implementi correctament el que demana l'enunciat.
  - Exercici 1 (0.25 punts)
  - Exercici 2 (0.75 punts)
  - Exercici 3 (1 punt)
  - Exercici 4 (1.75 punts)
  - Exercici 5 (0.5 punts)
  - Exercici 6 (1 punt)
  - Exercici 7 (0.5 punts)
- **Documentació** (0.5 punts): Totes les funcions dels exercicis d'aquesta PAC hauran d'estar correctament documentades utilitzant docstrings (en el format que preferiu).
- **Modularitat** (1 punt): Es valorarà la modularitat del codi (tant l'organització del codi en fitxers com la creació de funcions).
- **Estil** (0.5 punts): El codi ha de seguir la guia d'estil de Python (PEP8), exceptuant els casos on fer-ho compliqui la llegibilitat del codi.
- **Tests** (1.25 punts): El codi ha de contenir una o diverses *suïtes* de testos que permetin comprovar el bon funcionament de les funcions implementades, obtenint un mínim del 50% de cobertura.
- **Requeriments** (0.5 punts): Hi haurà d'haver un fitxer de requeriments que llisti (només) les llibreries necessàries per a executar el codi.
- **README i llicència** (0.5 punts): Es valorarà la creació d'un fitxer de README, que presenti el projecte i expliqui com executar-lo, així com la inclusió de la llicència sota la qual es distribueix el codi (podeu triar la que vulgueu).

## Important

**Nota 1:** De la mateixa manera que en les PACs anteriors, els criteris transversals es valoraran de manera proporcional a la part de la funcionalitat implementada.

Per exemple, si el codi només implementa la meitat de la funcionalitat demanada, i la documentació d'aquesta part és perfecta, aleshores la puntuació corresponent a la part de documentació seria de 0.25.

**Nota 2:** És imprescindible que el paquet que lliureu s'executi correctament a la màquina virtual, i que el fitxer de README que inclogueu expliqui clarament com s'ha d'executar el vostre codi per tal de generar les gràfiques resultants de l'anàlisi i tots els resultats, a més de com executar els tests i comprovar la cobertura.

**Nota 3:** Lliureu el paquet com a un únic arxiu .zip al Registre d'Avaluació Continua. **El codi de Python haurà d'estar escrit en fitxers plans de Python.**