Assignment for Image Recognition and Object Detection

In the final assignment of the "Visión por Computador" course, you will solve different problems using deep models.

The goals of the assignment are:

- Develop proficiency in using Tensorflow/Keras for training Neural Nets (NNs).
- Put into practice acquired knowledge to optimize the parameters and architecture of a *feed-forward Neural Net* (ffNN), in the context of an image recognition problem.
- Put into practice NNs specially conceived for analysing images. Design and optimize the parameters of a *Convolutional Neural Net* (CNN) to deal with previous image classification task.
- Train popular architectures from scratch (e.g., GoogLeNet, VGG, ResNet, ...), and compare the results with the ones provided by their pre-trained versions using *transfer learning.*
- Learn how to modify previous classification NNs for detecting objects. Build an object detector based on a sliding window, single-stage, or two-stage strategy.

Database

xView (http://xviewdataset.org/) is a large publicly available object detection data set, with approximately 1 million objects across 60 categories. It contains manually annotated images from different scenes around the world, acquired using the WorldView-3 satellite at 0.3m ground sample distance. There are 846 annotated images in total. For this practice, we divide these annotations into 761 and 85 images for training and testing respectively.

From now on, we discard most categories selecting the 12 most represented classes. As a result, we will only use **9809 and 1089 images for training and testing** respectively for the image detection task. The resulting images are resized to 600x600. Follow the link below to download the detection data set "xview_detection": https://drive.upm.es/s/XEBO3UVzyEI1M13

For image recognition, we crop previous images using their annotated bounding boxes to extract a subset of objects of interest. In this way, we collected **21377 and 2635 objects for training and testing** respectively. The resulting images are resized to 224x224. Follow the link below to download the classification benchmark "xview_recognition": https://drive.upm.es/s/4oNHlRFEd71HXp4

Annotations for train are available for download, whereas test annotations will remain private. Test evaluation can be done through Codabench benchmark platform. Only one member of each group must submit to each competition a zip file containing the test results generated in the json file.

Practical Assignment

Image Recognition

Design and train Neural Nets (NNs) to deal with the "xview_recognition" benchmark. Compare results obtained using ffNNs and CNNs. You can start your work from the sample notebook provided below, and improve its performance (see ffNN_example.ipynb).

To address this problem, you must decide on:

- Number of layers and number of units in each layer of your NN.
- Optimization algorithm and parameters to train the network.
- Check the evolution of these parameters during the optimization by using a validation subset and decide when to stop training.

It is also worth mentioning that there is a severe class imbalance. This difference in class frequencies affects the overall accuracy and should be considered when training the model.

Category	Count	Frequency
Cargo plane	635	2.97%
Helicopter	70	0.32%
Small car	4290	20.06%
Bus	2155	10.08%
Truck	2746	12.84%
Motorboat	1069	5.00%
Fishing vessel	706	3.30%
Dump truck	1236	5.78%
Excavator	789	3.69%
Building	4689	21.93%
Storage tank	1469	6.87%
Shipping container	1523	7.12%

Table 1. Class frequencies in our "xview_recognition" train subset.

Compare your architecture and results with other popular architectures (e.g., GoogLeNet, VGG, ResNet, etc.). Additionally, train some of these popular architectures from scratch, and compare the results with the ones provided by their pre-trained versions using transfer learning.

Small car | Building | Cargo plane | Truck | Fishing vessel

Figure 1. Sample objects of different categories acquired from the "xview_recognition" benchmark.

The Image Recognition challenge is properly hosted on: https://www.codabench.org/competitions/4278
Participants will have to login on Codabench using the UPM institutional email (@alumnos.upm.es)

Object Detection

In this task, you will take image classification to the next level, by recognizing multiple objects of different classes within a single image. Build a CNN to deal with the "xview_detection" benchmark.

To address this problem, you must choose one of the following options:

- Implement a sliding window strategy to process the whole images, and then train a classifier that determines whether each window includes or not an object of interest. In this way, you can use previous image classification model to infer the object category.
- Build a single-stage object detection model (e.g., YOLO, SSD, RetinaNet, etc.).
- Build a two-stage object detection model (e.g., Faster R-CNN, R-FCN, etc.).

We recommend the use of KerasCV API which includes single- and two- stage detection implementations. Note that the computational requirements needed to train an object detector are too much higher than those required for the image recognition task.

The following notebook provides the sample notebook of object detection results and detection metrics (see detector_example.ipynb).

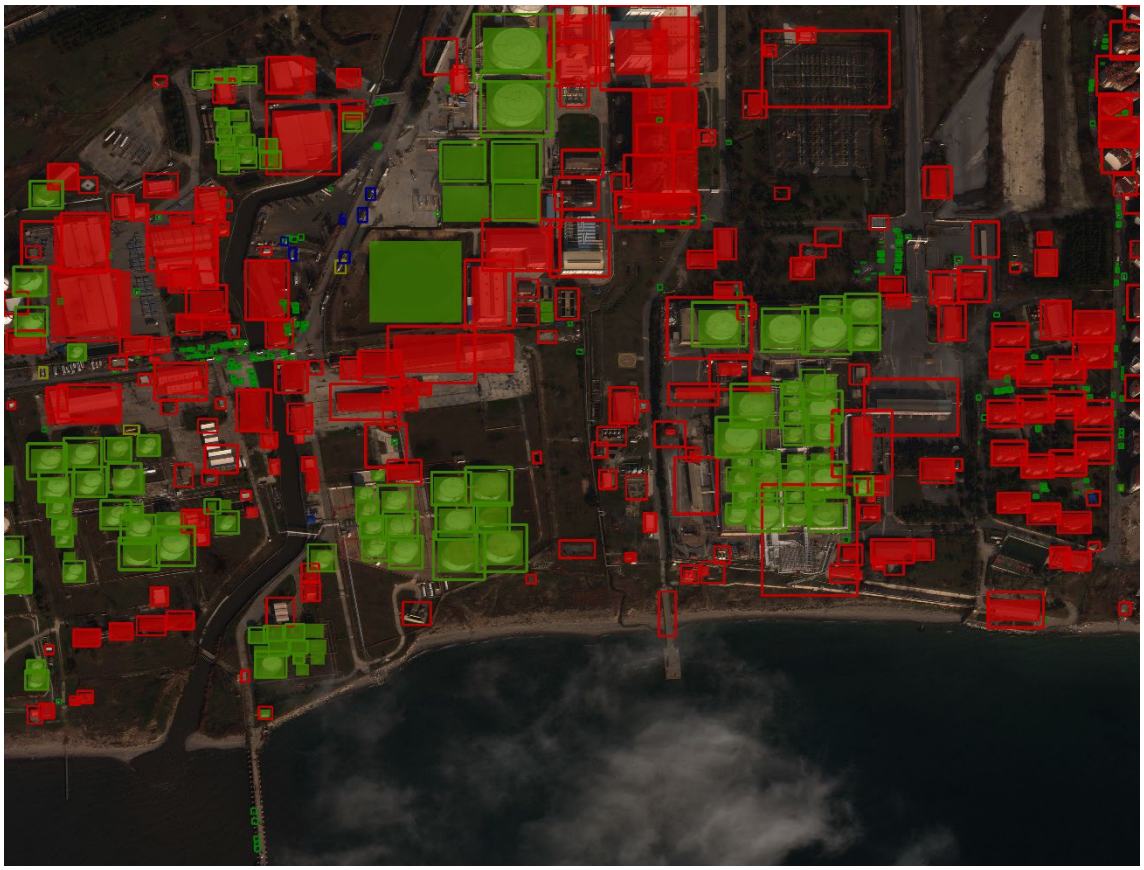

Figure 2. Sample image from the "xview_detection" test subset. The bounding boxes and contours represent predictions and annotations respectively, using a different colour for each category.

The Object Detection challenge is properly hosted on: https://www.codabench.org/competitions/XXXX
Participants will have to login on Codabench using the UPM institutional email (@alumnos.upm.es)

Presentation of results

You must prepare a **report (.pdf)** describing:

- The problems and data sets (briefly).
- The process that you have followed to reach your final solution on both detection and classification tasks, including your intermediate results. You must discuss and compare these results properly.
- Final network architectures, including optimization algorithm, regularization methods (dropout, data augmentation, etc…), number of layers / parameters, and performance obtained with your model on the testing data set, including the plots of the evolution of losses and accuracy performances.

In the submission via Moodle, attach your **Jupyter Notebook file (.ipynb)** with the results of computations attached to the code which generated them.

The assignment must be done in **groups of 3 students maximum**. Each team must submit one submission before **Tuesday, January 14th, 23:55h.**