Training YOLO v3 for Objects Detection with Custom Data

Joining datasets

Joining datasets Labelled and Custom

In order to have opportunity to train two datasets, *Labelled* and *Downloaded Custom*, together in Darknet framework, it is needed to *join* them properly, *create* needed files and *update* annotations.

Firstly, create folder with name *Labelled-Custom* for joined dataset and copy in there all images from *Labelled* and *Downloaded Custom* datasets. You should have folder with images from both datasets, like following:

Labelled-Custom/
        image001.jpg
        image002.jpeg
                …

Pay attention! Copy only images from two datasets that have annotations. For example, if in *Labelled* dataset the images labelled partly, copy only images that have annotation *txt* files next to them. Or, finish annotating before.

After all images were collected in one folder, it's time to prepare files needed for training in *Darknet framework.*

These files are:
- joined_data.data
- classes.names
- train.txt
- test.txt

Five lines inside *joined_data.data* are:

- classes = 4
- train = /home/my_name/**train.txt**
- valid = /home/my_name/**test.txt**
- names = /home/my_name/**classes.names**
- backup = backup

First line specifies number of classes, namely, number of unique objects from two datasets that *YOLO v3* will be trained on, and that will be used for detection after training.

Second line specifies full path to the file *train.txt* that in turn consists of full paths to the images for training. The same is true for **third line** with difference that images are used for validation during training.

Fourth line specifies full path to the file *classes.names* that has unique and non-repeated names of objects for joined dataset.

Fifth line specifies folder where trained weights will be saved.

Download Py files into Joined-Labelled-Custom

Create a folder with name *Joined-Labelled-Custom* to keep everything organized. Download *Py* files from *Resources* and copy them to this folder. You should have following:

- *Joined-Labelled-Custom/*
    - *getting-full-path.py*
    - *joined-train-and-test-txt-files.py*
    - *joined-files-data-and-names.py*

Getting full paths

Before creating needed files to train in *Darknet framework*, it is needed to find *absolute* or *full paths* to the directories with *Labelled* dataset, *Downloaded Custom* dataset and just created folder with joined images *Labelled-Custom*:

- Copy and paste *Py* file ***getting-full-path.py*** to the folders with *Labelled* dataset, *Downloaded Custom* dataset and *Labelled-Custom*
- Open *Terminal* (or *Anaconda Prompt*) and activate your *Python v3* environment and go to the one of the needed directories. You can list all available sub-directories in the current directory by using following command in *Terminal* (or *Anaconda Prompt*):
  ```
  dir
  ```
  It will show all sub-directories you can go in. Go inside needed directory by using following command in *Terminal* (or *Anaconda Prompt*):
  ```
  cd Downloads/Labelled-Custom
  ```
  (yours should be different)
- Run following command in *Terminal* (or *Anaconda Prompt*):
  ```
  python3 getting-full-path.py
  ```
  or:
  ```
  python getting-full-path.py
  ```

- Repeat these steps to all three directories. You should get full paths like following (yours should be different):
  - */home/my_name/Downloads/video-to-annotate*
  - */home/my_name/OIDv4_Toolkit/OID/Dataset/train/Car_Bicycle_wheel_Bus*
  - */home/my_name/Downloads/Labelled-Custom*
- Open *Py* file ***joined-train-and-test-txt-files.py*** and *Py* file ***joined-files-data-and-name.py*** in your *Programming Environment (PyCharm or any other you use)* and assign to the following variables found full paths:
  - `full_path_to_labelled_images = ‘’`
  - `full_path_to_downloaded_images = ‘’`
  - `full_path_to_joined_images = ‘’`

Creating files train.txt and test.txt

When full path was found, it is time for creating files *train.txt* and *test.txt*:

- Open *Py* file ***joined-train-and-test-txt-files.py*** in your *Programming Environment (PyCharm or any other you use)*
- Run the code
- Open folder with joined images *Labelled-Custom* and check if *txt* files were created

Creating files joined_data.data and classes.names

Next, it is time for creating files *custom_data.data* and *classes.names*:

- Open *Py* file ***joined-files-data-and-name.py*** in your *Programming Environment (PyCharm or any other you use)*
- Run the code
- Open folder with joined images *Labelled-Custom* and check if files were created

Verify annotations by LabelIMG

After joining two datasets and updating annotations, it is possible to check that joining were made correctly, namely, updating of classes’ numbers.

- Open folder with joined images *Labelled-Custom*
- Create one more *txt* file with name ***classes.txt*** (use any text editor like *notepad* or other) and copy in it all lines with classes’ names from other file **classes.names** that has joined and unique classes’ names (yours can be different):
  car
  bicycle wheel
  bus
  motorbike
- Save changes and close the file ***classes.txt***

- Open *Terminal* (or *Anaconda Prompt*) and activate *environment* in which you installed *LabelIMG* tool
- Launch *LabelIMG* by one of the following command (depending on the way you chose for installation):
  `labelImg` (if pip was used)
  `python3 labelImg.py` (in other cases)
  `python labelImg.py` (in other cases)
- Go to *File* --> *Reset all* (it should close *LabelIMG*)
- Launch *LabelIMG* again
- Click on button *Open Dir* and navigate to the folder with images, annotations in *txt* files and just created file *classes.txt (Labelled-Custom)*
- By using *Next* and *Previous*, check if classes' names correctly describe bounding boxes with appropriate objects