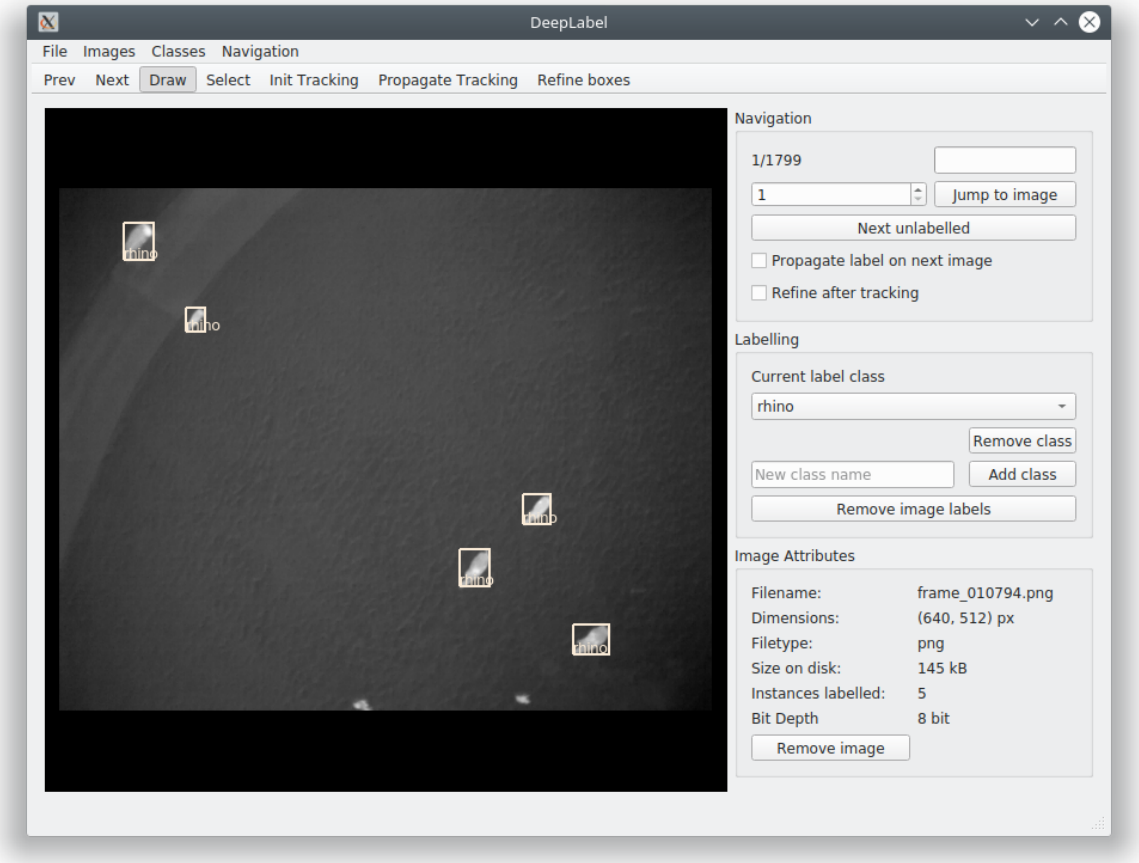
TUTORIAL

DEEPLABEL



V 1.0

SUMMARY

INTRO

DeepLabel is a cross-platform tool for annotating images with labelled bounding boxes. A typical use-case for the program is labelling ground truth data for object-detection machine learning applications. DeepLabel runs as a standalone app and compiles on Windows, Linux and Mac.

Deeplabel also supports running inference using state-of-the-art object detection models like Faster-RCNN and YOLOv4. With support out-of-the-box for CUDA, you can quickly label an entire dataset using an existing model.

DeepLabel can be used via GUI or via command line for automated processing.

More information

https://github.com/jveitchmichaelis/deeplabel

Download release (windows)

https://github.com/jveitchmichaelis/deeplabel/releases/tag/0.16.1

UTLN Existing Database

https://github.com/valentinbarchasz/Yolo-Dataset-Project

You will find under ImagesDatasets/robocup-MSL-dataset/ :

- Folders of images (Nubot Dataset + UTLN)
- Output folder containing exported data from DataSet_3C_Robot-Ballon-But_Set_NuBot.lbldb for Darknet Yolo
- a list of .lbldb files. (each .lbldb represent a dataset)

WorkFlow

DeepLabel was built with convenience in mind. Image locations, classes and labels are stored in a local sqlite database (called a *project*, in the application). When a label is added or removed, this is immediately reflected in the database.

A typical workflow for DeepLabel is:

1. Create a new project database
2. Add images, or import an existing project in a variety of common ML formats
3. Load in a class list, or manually add classes
4. Label/inspect the images
5. Export data in the desired format

1. Create a New Project Database

The first step to annotate image, is to create a new Database file (.lbldb)

To do this, run deeplabel.exe

Then click “File”->”NewProject”

Select a location where to save the database.

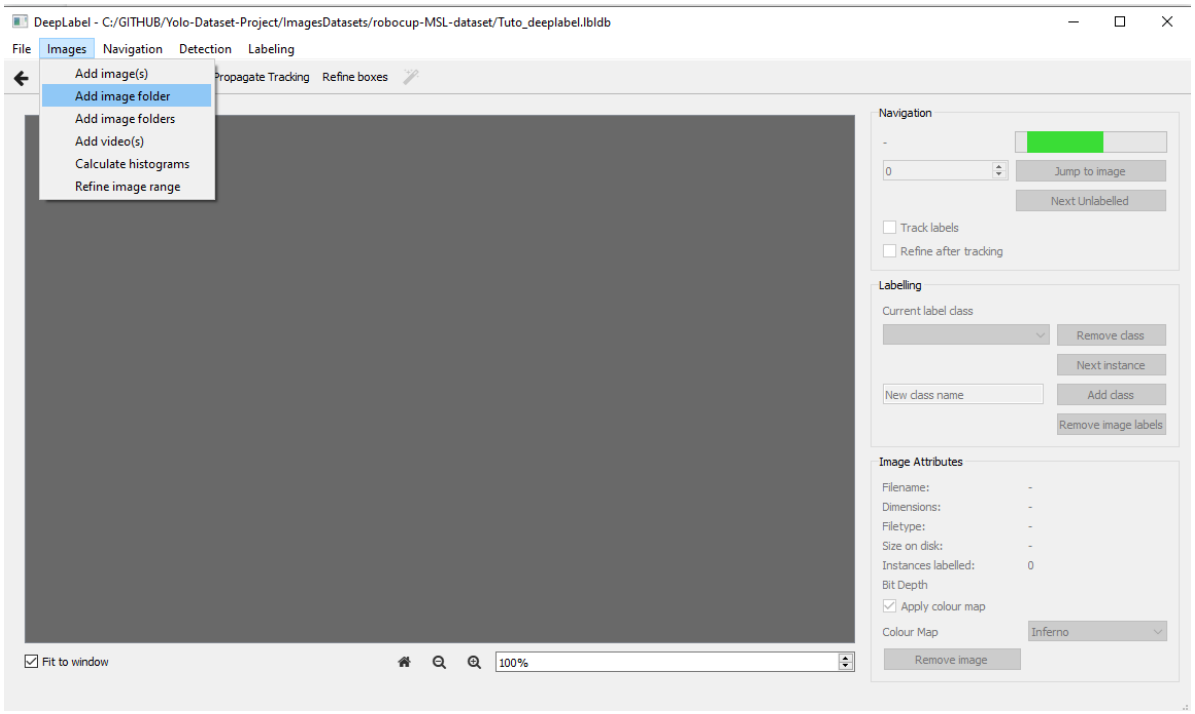
<u>KEEP IN MIND:</u> that the path to labeled images is saved as a <u>relative path</u> in the database. So once the database is created, avoid moving/renaming images folder, or change path of the labelled images.

2. Add images to database

Once your database is created, you may want to import images and start labeling them.

To do this, clicon “Images” menu and select the desired option (add image(s), add Image folder, add Image folders).

In this tutorial, we will choose “add Image folder”



Once done, we can see the number of images imported in the database, and we can preview them.

You can repeat the operation, and add multiple folders to the database.

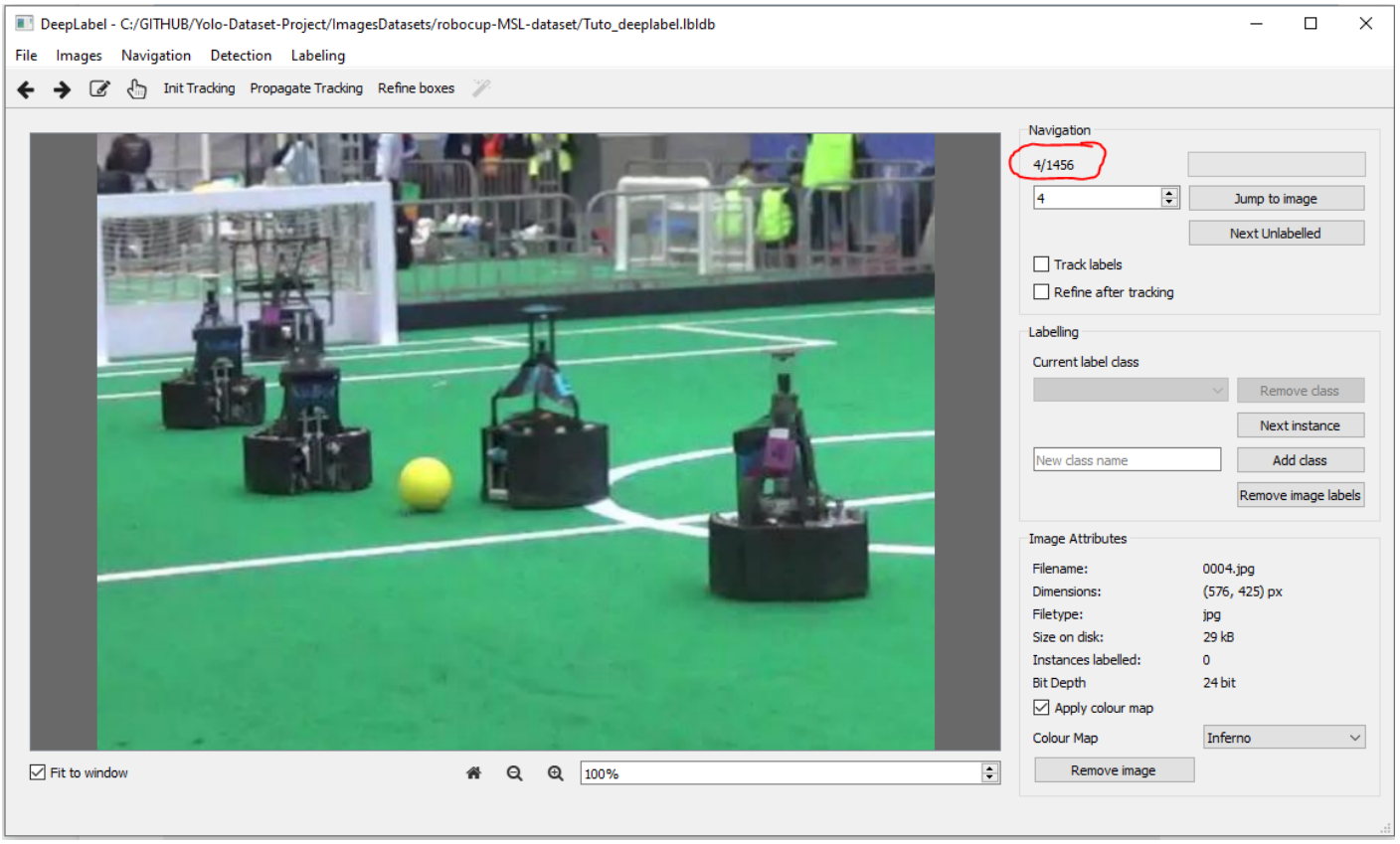


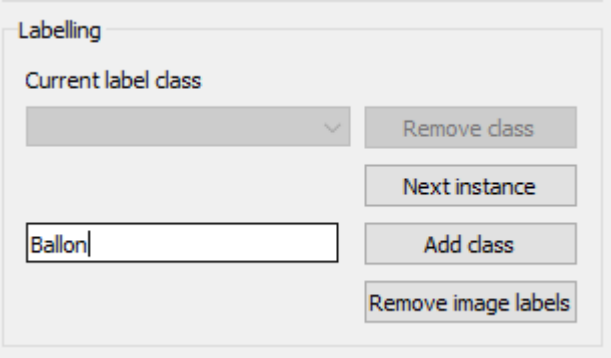
Image paths in the database are stored relative to the database location. This means you can easily copy over files to another system, provided you keep the relative structure of the files.

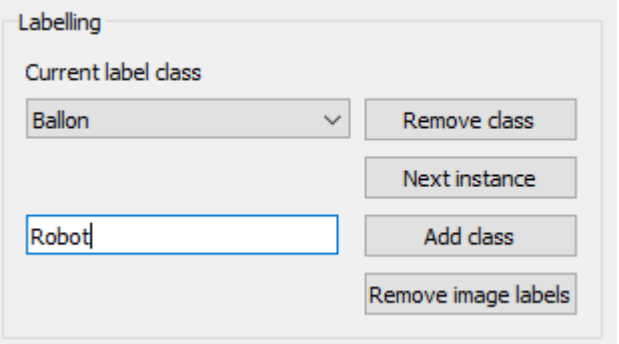
3. Adding Classes to the database

To add a class to the database, go on the right side of GUI, in “Labeling” box, clic on the textbox “*New class name”* and type the name of the class to add.

Then click on the “Add class” Button.

In this example we will create a “Ballon” class and a “Robot” class.





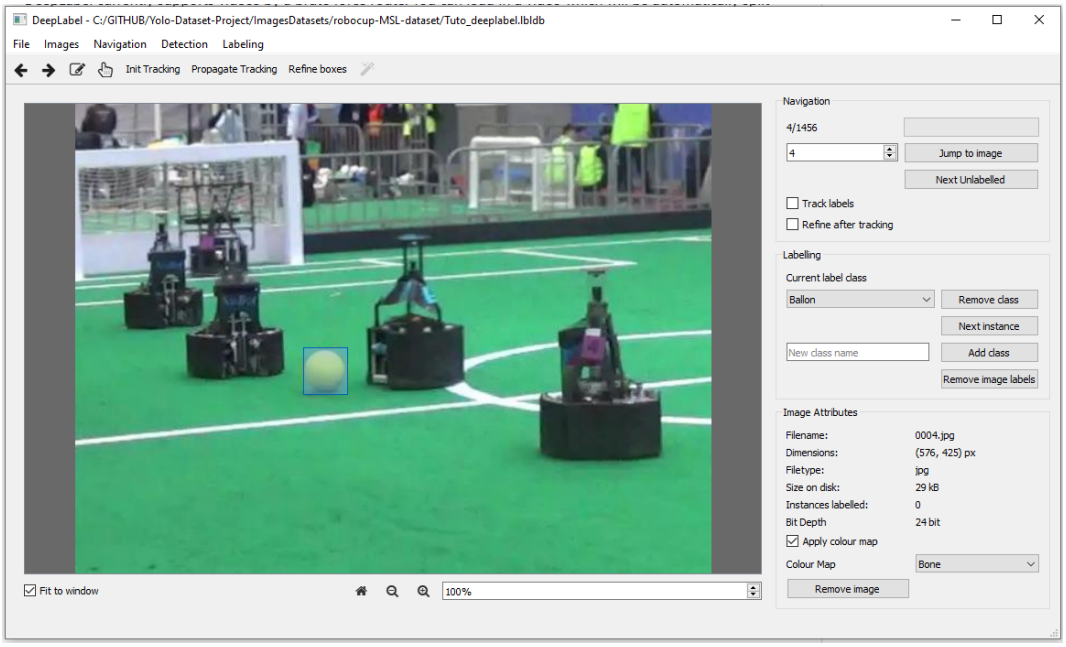
You can also delete a class (for the entire database), or remove image labels (for the current image).

4. Label/Inspect Images

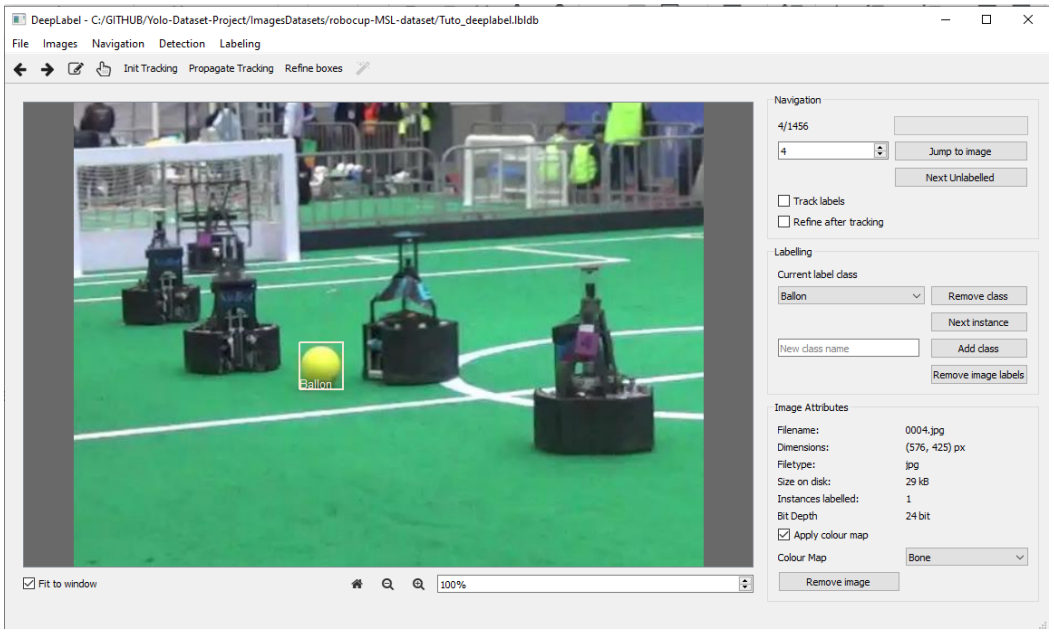
Draw Mode:

In **draw** mode, you can click to define the corners of a bounding box rectangle.

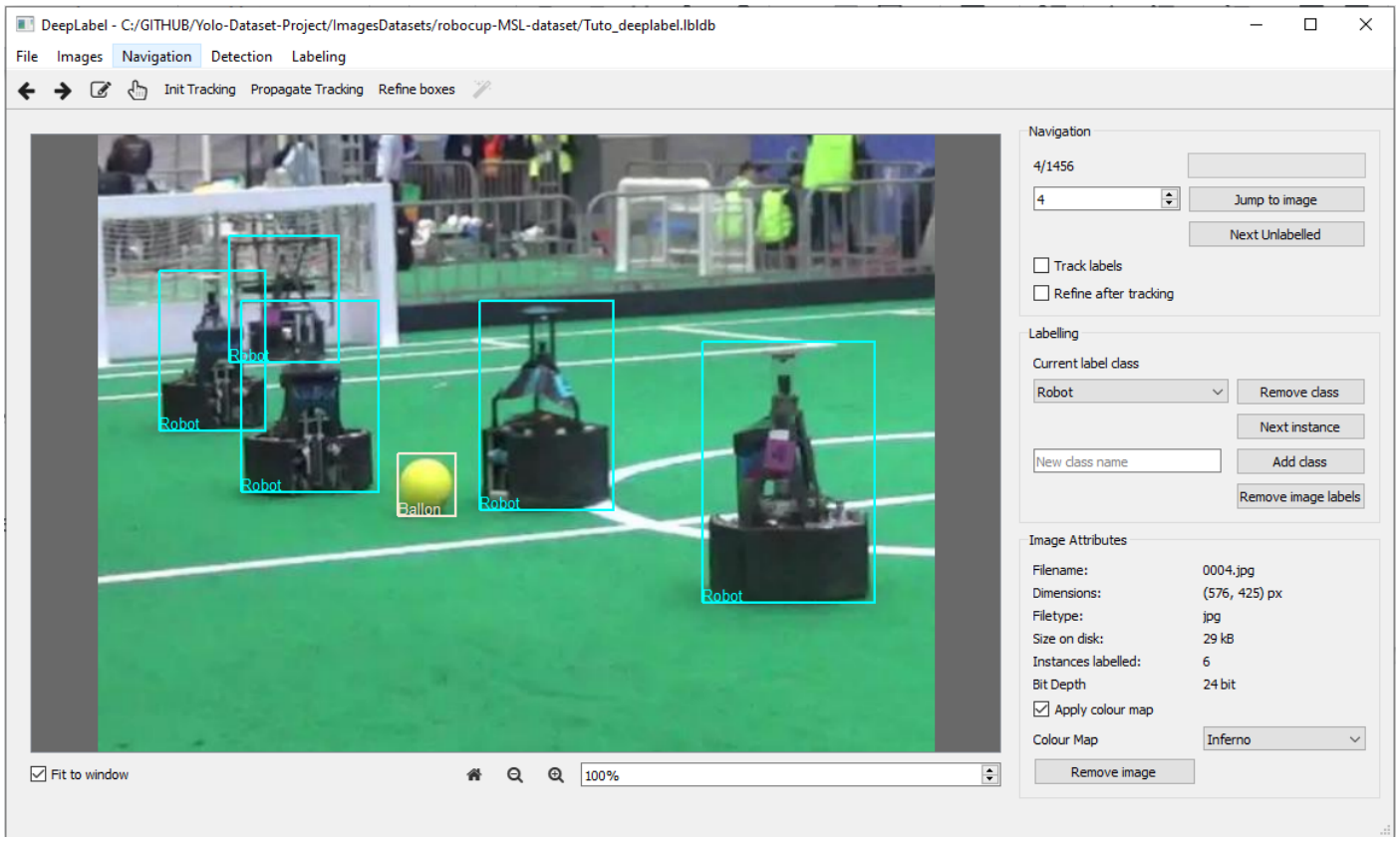
To label an image, go in draw mode (click on the ) and simply draw a rectangle on the object you want to annotate (with the currently selected class), then press the “Space Bar” key.



You should see a label on your bounding box:



Repeat this step for many objects you want to label, and for every image of the dataset.



Select Mode:

If you need to delete a label, switch to **select** mode. Click on a rectangle, it will highlight green, then hit delete or backspace to remove it.

5. Export Dataset

Currently you can export in:

- KITTI (e.g. for Nvidia DIGITS)
- Darknet for YOLO
- Pascal VOC
- COCO (experimental)
- Google Cloud Platform (e.g. for AutoML)
- TFRecord (for the Tensorflow Object Detection library)
    - Note this uses protobuf directly and there is *no* dependency on Tensorflow. I believe this is one of the few implementations of TFRecord writing in c++.
- Video (experimental, command line only)

Deeplabel treats your data as "golden" and does not make any attempt to modify it directly. This is a safe approach to avoid accidental corruption of a dataset that you spent months collating. As such, when you export labels, a copy of your data will be created with associated label files. For example, KITTI requires frames to be numerically labelled. In the future, augmentation may also be added, which is another reason to **not** modify your existing images.

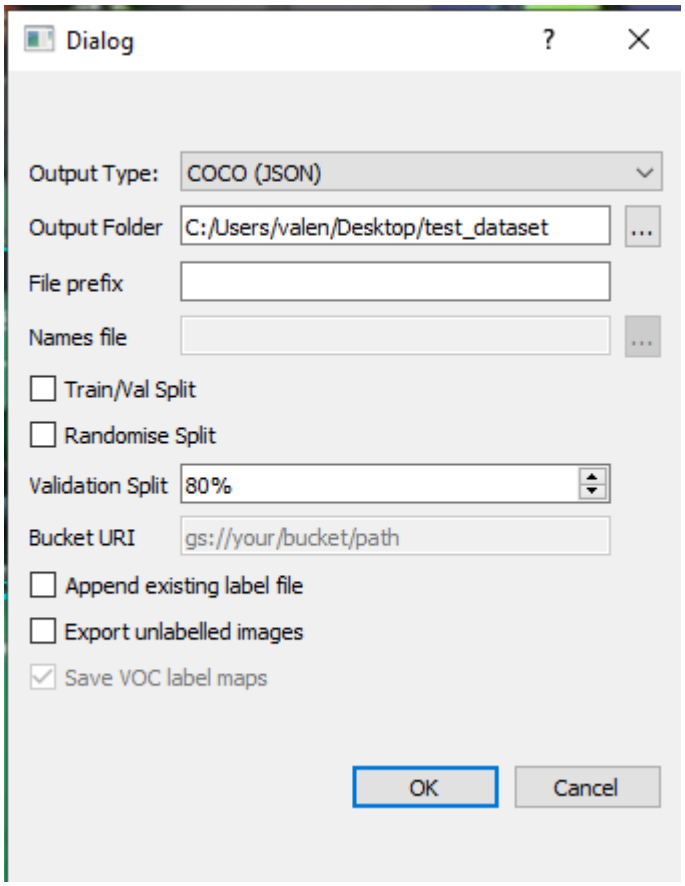
When exporting to darknet, you should specify an existing "names" file so that your output labels have consistent class IDs. Similarly, when you export to Pascal VOC, you have the option of exporting a label map file which maps class IDs to labels. This file is quite easy to generate yourself (and you may already have it). The format is:

```
{

item {

  name: some_class

  id: 1

  displayname: some_class

}

}
```

DeepLabel can automatically split your data into train and validation sets, you can choose what fraction to use (you can set 0% or 100% if you just want a test or validation set).

If you want to export your dataset for Darknet Yolo, you can use the GUI to do so.

“File”->”Export labels”



Else, if you want to export your dataset in the **TFRecord** format (used for tensorflow, …) you **have to** use the command line.

<u>Exemple:</u>

deeplabel.exe export -i Tuto_deeplabel.lbdlb -f tfrecord -o ./output_dataset/ -s 0.2

Command Line Interface

Deeplabel has a convenient command line interface to facilitate import and export of data:

(base) PS C:\Users\Josh> deeplabel.exe -h

Usage: deeplabel.exe [*options*] *mode*

Options Description

Options	Description
-?, -h, --help	Displays help on commandline options.
--help-all	Displays help including Qt specific options.
-v, --version	Displays version information
-f, --format	export format, can be: (kitti, darknet, gcp, voc, coco, mot , birdsai, tfrecord)
-o, --output <folder path>	Output folder
-i, --input <file path>	label database
-s, --split <percentage>	validation split percentage
--no-subfolders	export directly to specified folder
--prefix <prefix>	filename prefix
-n, --names <file path>	names file
--bucket	GCP bucket
--local	use local paths for GCP export
--export-map	export label.pbtxt file
--shuffle	shuffle images when splitting
--append-labels	append to label files
--export-unlabelled	export images without labels
--images <images>	import image path/folder
--annotations <annotations>	import annotation path/folder

--import-unlabelled	import images without labels
--overwrite	overwrite existing databases
--records <images>	mask for TF Records (*wildcard)

Mode Description

Mode	Description
import	used to import label in database
export	used to export dataset

Keyboard Shorcuts

Shortkey	Description
left	Go to previous image
right	Go to next image
Ctrl+left	Step 10 images next
Ctrl+right	Step 10 images previous
Space	Validate annotation box