image recognition training

A guide for training and testing darkyolo neural network

Written by: Linas Minkevičius

Company: Zoomtv.lt (TV Partneriai)

Table of Contents

[Purpose of a guide 4](#_Toc523146682)

[Introduction 4](#_Toc523146683)

[Yolodarknet setup 4](#_Toc523146684)

[Useful files and folders 4](#_Toc523146685)

[Yolo file tree 4](#_Toc523146686)

[Root folder 5](#_Toc523146687)

[backup folder 5](#_Toc523146688)

[darknet.exe 5](#_Toc523146689)

[chart.jpg 5](#_Toc523146690)

[yolo\_[insert your config name here].cfg 5](#_Toc523146691)

[filenames.txt 5](#_Toc523146692)

[data folder 5](#_Toc523146693)

[obj.data 5](#_Toc523146694)

[obj.names 6](#_Toc523146695)

[test.txt 6](#_Toc523146696)

[train.txt 6](#_Toc523146697)

[logoData folder 6](#_Toc523146698)

[Supportand main applications 6](#_Toc523146699)

[Database preparation 6](#_Toc523146700)

[Gather videos 7](#_Toc523146701)

[Crop videos to frames 7](#_Toc523146702)

[“FrameCut” project setup 7](#_Toc523146703)

[Cropping 8](#_Toc523146704)

[Marking objects 9](#_Toc523146705)

[Data parsing 12](#_Toc523146706)

[Pre-Training data validation 12](#_Toc523146707)

[Training preparation 13](#_Toc523146708)

[Training data preparation 13](#_Toc523146709)

[Filepath txt generator 13](#_Toc523146710)

[Configuration editing 14](#_Toc523146711)

[Training 14](#_Toc523146712)

[Post-Training validation 15](#_Toc523146713)

[Database maintainance 17](#_Toc523146714)

[Database version tracking 17](#_Toc523146715)

[Object removing 17](#_Toc523146716)

[Image augmentation 17](#_Toc523146717)

[No table of figures entries found. 19](#_Toc523146718)

[Conclusion 20](#_Toc523146719)

# Purpose of a guide

The purpose of this guide is to clear up all of the misunderstandings and communication errors when trying to train Darknet neural network with a guidance of a person.

This guide will explore things as:

* Preparing training program
* Preparing database
* Training
* Testing
* Maintaining database

IMPORTANT NOTICE: THIS GUIDE WILL BE HEAVILY DEPENDANT ON OUR PROJECT LOGOTRACKER, IT DOES NOT TELL YOU HOW TO OPERATE CMAKE OR HOW TO SET UP SYTEM FOR CMAKE PROJECT.

# Introduction

Let’s say you want your computer software to recognize certain thing or things in a video or an image. What do you need to accomplish that? You require a trained neural network. The latest technology we have is YOLODarknet. As I currently do not fully understand what is happening under the hood, I suggest you go and do some reading at <https://github.com/AlexeyAB/darknet>

Proceed to **Yolodarknet setup**.

# Yolodarknet setup

So you want to use Daknet neural network? You are lucky! There are some quite good instructions how to set up (and even use) written on <https://github.com/AlexeyAB/darknet>. Go there and do some reading. I can’t really help with this yet, as I only did this setup 1-2 times.

After setup, jump to **Database preparation**

# Useful files and folders

There are several files and folders inside yolo/x64 folder you will be using while preparing and training this network.

## Yolo file tree

* Yolo/x64 (Root folder)
  + Darknet.exe
  + Chart.jpg
  + yolov2\_logo\_detection.cfg
  + filenames.txt ?
  + backup
  + data
    - obj.data
    - obj.names
    - test.txt
    - train.txt
    - logoData

### Root folder

This folder is really important. It contains all of the executables and other locations you will need while using this trainer.

### backup folder

Folder containing all of the computed weights after training.

### darknet.exe

This is the main executable which is used to start training.

### chart.jpg

Picture generated from training graph. You can use this at presentations to show how training curve went by.

### yolo\_[insert your config name here].cfg

Another really important file. This one has most of the instructions trainer will be using. Such as learning rate, batch sizes, subdivisions, height, width layer count and etc. If you want to know more, I suggest do some more reading on git.

### filenames.txt

I am not sure about this yet. Although it contains all picture database filenames.

### data folder

Folder containing your database and information about objects you want to recognize.

### obj.data

File containing file paths to the object data. Also contains line to define how many objects there are.

### obj.names

Can you guess what it is? No? It is obviously a file containing object names.

### test.txt

File containing all of the image paths of the database. Used for testing?

### train.txt

File containing all of the image paths of the database. Used for training.

### logoData folder

Folder containing your database of pictures and txt files with the same name as they. Those files tell where and what object is in that picture. If it is empty, there is none. You will have to visit this folder pretty damn often…

## Supportand main applications

This project also uses our own applications to prepare data, test it and manage final product.

Currently it is located inside [D:\LogoBackend\](file:///D:\LogoBackend\src)

If you want to contribute, ask for access to Bitbucket repo called “LogoBackend” from IT administrator (Mindaugas Butkus).

# Database preparation

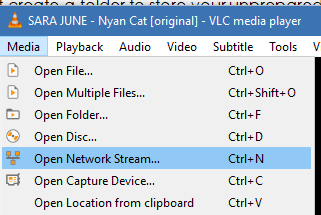
Preparing the database is essential. If I tell you to give me some runite ore and a bottle of aggression potion, what would you do? Most likely shrug your shoulders because you have no clue what they are. But if I told you that it was from a certain game, you could answer me that you don’t play it or do not have those items. It shows that with some information from the past, you can give me some defined answer. Same goes for the neural networks as they are based on how human nerves work. If you show it enough of images with your required object in it and point where it is in a picture and where it is not, after some time it could give you a defined answer of a confidence of an object in a picture.

But I see you asking – where can I get those pictures? And to answer that – it depends on the objects and scenarios you will be using. There is a whole field of science of preparing databases. I am no pro at it. All I can tell is about our current example of using basketball games videos to extract images for our logo detection.

\*Drums rolling\*

It is time for our actual first step!

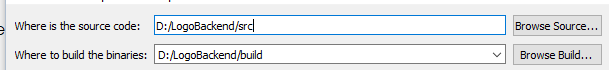
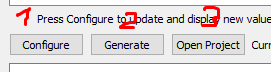
## Gather videos

* First create a folder to store your unprepared database somewhere nearby Root folder.
* Make or find videos containing your object, and some videos which don’t. Make sure those videos are as high quality as possible. We do not want to play “what can you see in this ink spot over 100 m away”. Same goes for the objects – make sure someone could easily recognize them without context. They should be fairly big (at least 50x50).
* Rename videos so they would not contain spaces and foreign symbols.
* It is wise to crop long videos to pieces containing your objects. You can use VLC media player recording button.
* You can also rip youtube videos with VLC. Just right click on video, copy video url, open VLC and : 

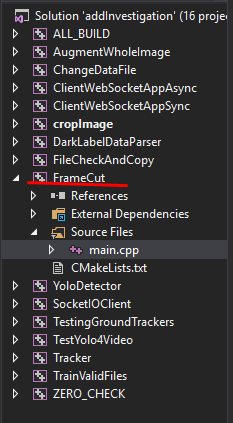
## Crop videos to frames

This step will use our application called “FrameCut”. Before that, we need to set it up. Same process will go for other applications as well.

## “FrameCut” project setup

1. Add < add\_subdirectory(FrameCut) > to [D:\LogoBackend\src\CMakeLists.txt](file:///D:\LogoBackend\src\CMakeLists.txt)
2. Open CMake
3. Type in this: 
4. Press on these: 

It should open the project solution with projects defined in <file:///D:\LogoBackend\src\CMakeLists.txt> .



If you get some errors, it is recommended to contact Rytis Augustauskas for assistance. Or you could do it yourself, but be aware that you can mess everything up.

1. Build ALL\_BUILD with both debug and release versions.
2. Open “FrameCut” project bin folder. [D:\LogoBackend\build\FrameCut\Debug](file:///D:\LogoBackend\build\FrameCut\Debug)

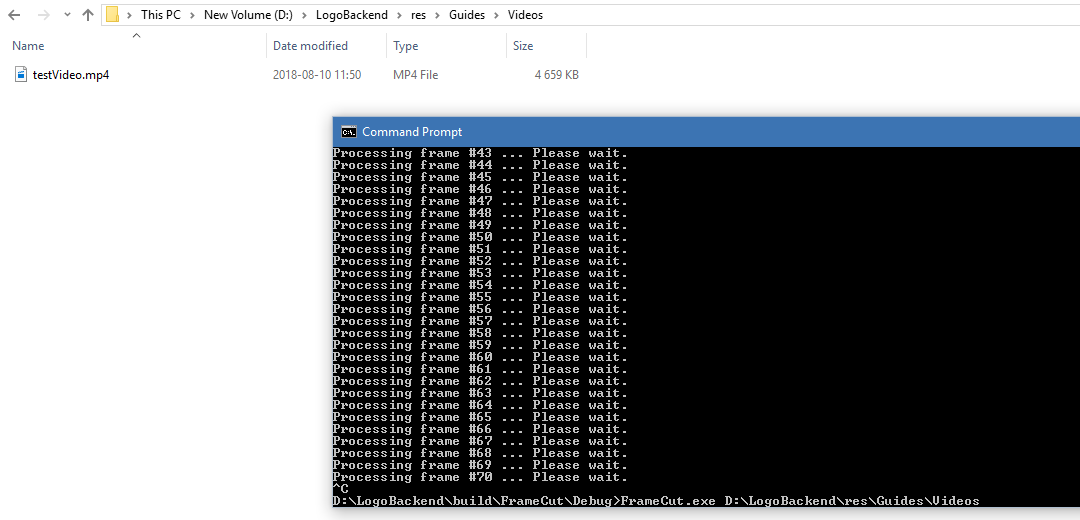
Congrats! You have finished setting up “FrameCut” application. Proceed to next step.

## Cropping

1. Open CMD.
2. Navigate to [D:\LogoBackend\build\FrameCut\Debug](file:///D:\LogoBackend\build\FrameCut\Debug). To make it easy:

**D:**

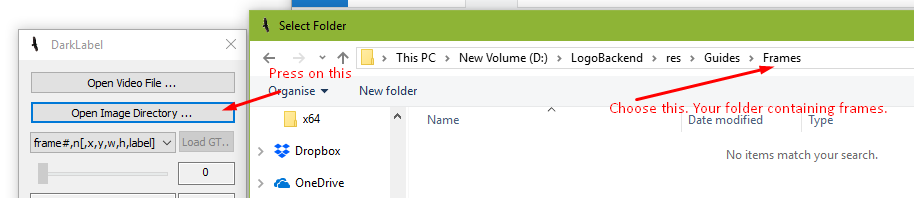
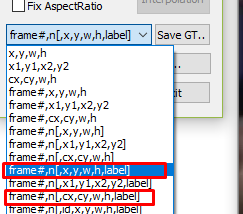
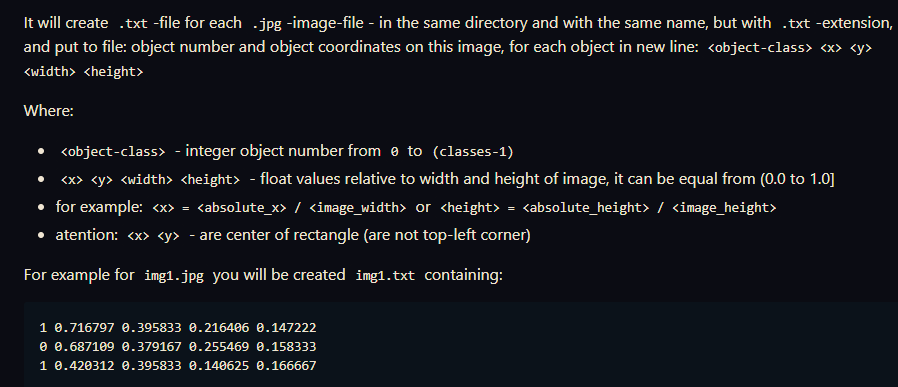
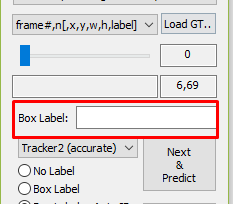
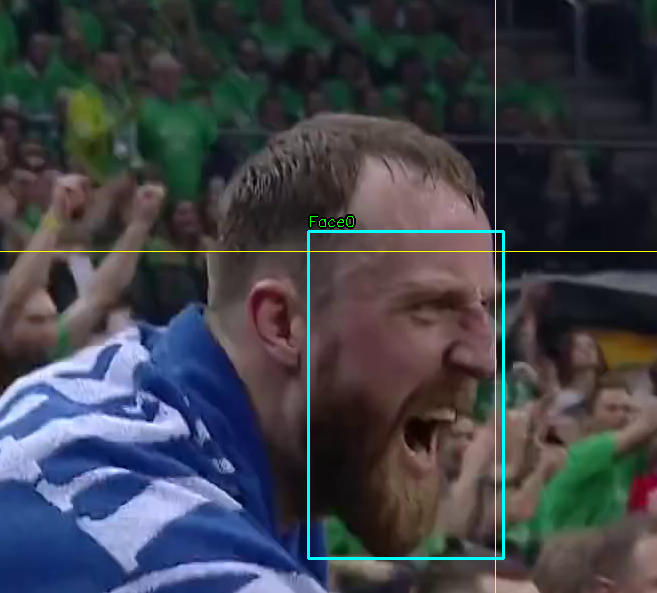
**Cd** [**D:\LogoBackend\build\FrameCut\Debug**](file:///D:\LogoBackend\build\FrameCut\Debug)

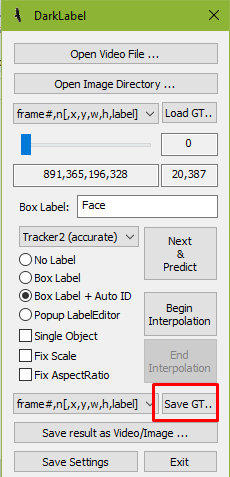
1. Press TAB. It should automatically fill line with FrameCut.exe
2. Paste in path to folder containing all of your videos. And videos only! 
3. Press ENTER and wait.
4. All of the frames will be found in [D:\LogoBackend\build\FrameCut\Debug](file:///D:\LogoBackend\build\FrameCut\Debug). Take them to some other folder. And continue to next step.

## Marking objects

As we prepared frames to work with, it is time to start to label them. Basically make several pictures with encircled objects.

To start, first we need to get marking software <http://darkpgmr.tistory.com/16> . Follow instructions to set it up. When installed, you MUST copy *opencv\_ffmpeg320.dll* (Found in OpenCV library) into DarkLabelv\_part1 if you want to mark straight to video and not to frames.

1. Open DarkLabel.exe
2. 
3. You can choose any of these formats.  Preferably first one. Some conversions will need to be made anyway, as openCV requires that rectangle corner would be top left, and DarkYolo takes center of the rectangle. ** You should take note on what kind of format you are using.
4. Read through manual of the program for hotkeys.
5. If frame contains your object:
   1. Type in the name of your object into . **ALWAYS MAKE SURE THAT YOU TYPE IN THE NAME IN YOUR OBJECT LIST. MESSING THIS UP WILL MAKE YOU LOSE A LOT OF TIME. THIS NAME WILL BE USED IN TRAINING.**
   2. Mark your object with a mouse. Try to be as accurate as you can. 
   3. If the frame does not contain any more objects, press ENTER, or Next&Predict.
   4. Adjust if needed the bounding boxes of your objects and add missing ones if needed.
   5. Continue until the end of the frames.

It is a good idea to keep saving after every 100 frames or so, as the program is not stable. It may crash and you would lose a lot of progress. To do that just press . If the program crashes, or you just want to continue where you have left off, press on “Load GT…” while selected your frame format.

When you have finished marking everything, save, and name this file something like “yourvideotitle\_final\_gt.txt” .

Proceed to next step.

## Data parsing

This program does not generate the right format on its own. It also does not generate separate txt files as it is required for DarkYolo. To do that, we need to use yet another program written by us. It is called “DarkLabelDataParser”. Follow **“FrameCut” project setup** to set it up.

1. Open [D:\LogoBackend\build\DarkLabelDataParser\Debug](file:///D:\LogoBackend\build\DarkLabelDataParser\Debug) in file explorer
2. Open CMD

**D:**

**Cd D:\LogoBackend\build\DarkLabelDataParser\Debug**

1. Press TAB
2. . Enter these parameters. dataPath – Location of your frames. Press ENTER.
3. Go to [D:\LogoBackend\build\DarkLabelDataParser\Debug](file:///D:\LogoBackend\build\DarkLabelDataParser\Debug) and paste all of the generated txt files inside your frames folder. There should be equal amounts of txt files to jpg files.

Proceed to the next step.

## Pre-Training data validation

This step is really important. Details are crucial. We will crop our objects from frames, and check if our frames look correct. We will be using “cropImage” project, so go on and follow **“FrameCut” project setup** to set it up.

1. Open [D:\LogoBackend\build\cropImage\Debug](file:///D:\LogoBackend\build\cropImage\Debug) .
2. Open your frames folder.
3. Create new folder structure in your frames folder
   1. cropedImg
      1. object1 – These names should be exact as in your object list.
      2. object2
4. Open CMD

**D:**

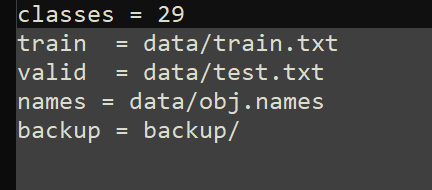
**Cd D:\LogoBackend\build\cropImage\Debug**

1. Press TAB
2. Copy path to your frames folder. Don‘t forget to add „/“ to the end of your path.
3. Press ENTER twice and wait.
4. After processing all frames, you will see how many objects were cropped. 
5. Open cropedImg folder in your frames folder. Now the real “fun” begins. Open each folder and slowly check each picture of your object. If it looks odd (too obscured, very blurry, wrong object at all) write the video name and frame ID.
6. Go to **Marking objects,** and fix your mistakes. Repeat all of those steps again until you (and probably someone else) are sure that everything seems fine.
7. Copy your data to database additions folder. See **Database version tracking.**

Proceed to next step. And consider using **Image augmentation**

# Training preparation

## Training data preparation

1. Backup your current database from **logoData folder** and **backup folder.** See **Database version tracking.**
2. Copy your current additions frames folder (the one containing jpgs and txts) into **logoData folder**.
3. Edit **obj.data** to match the number of objects you want to train. 
4. Edit **obj.names** by inserting or removing new objects. BE AWARE OF TYPOS.
5. Now we need to prepare two files containing all of the image paths.

### Filepath txt generator

We will be using application called “TrainValidFiles”, so follow **“FrameCut” project setup** to set it up.

1. Open [D:\LogoBackend\build\TrainValidFiles\Debug](file:///D:\LogoBackend\build\TrainValidFiles\Debug) .
2. Open CMD

**D:**

**Cd D:\LogoBackend\build\TrainValidFiles\Debug**

1. Enter t and press TAB
2. Copy path to your logoData folder. Don‘t forget to add „/“ to the end of your path. And it MUST be named „logoData“.
3. Press ENTER and wait patiently. This may take a couple of minutes.
4. You should see a list of all objects in your database. 
5. Paste train.txt and test.txt from [D:\LogoBackend\build\TrainValidFiles\Debug](file:///D:\LogoBackend\build\TrainValidFiles\Debug) to **data folder.**

### Configuration editing

Before starting training, you should consider adjusting **yolo\_[insert your config name here].cfg to** your own needs. This requires quite a bit of knowledge what are you doing, so I suggest reading about it on <https://github.com/AlexeyAB/darknet>

Otherwise leave our defined settings.

Once finished, go to **Training.**

# Training

To begin training, open [D:\Rytis\yolo\x64](file:///D:\Rytis\yolo\x64) , or your own DarknetYolo folder.

1. Open CMD
2. Paste in:

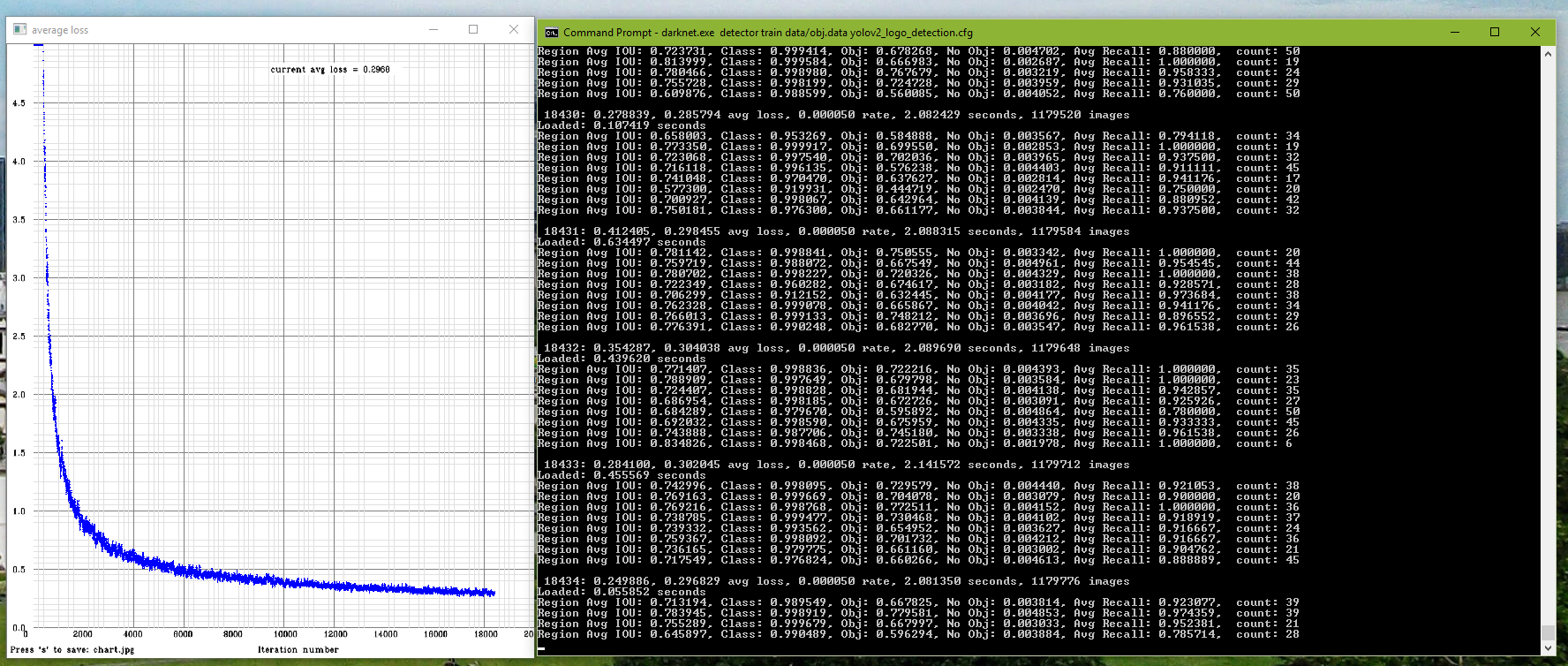
D:

Cd [D:\Rytis\yolo\x64](file:///D:\Rytis\yolo\x64)

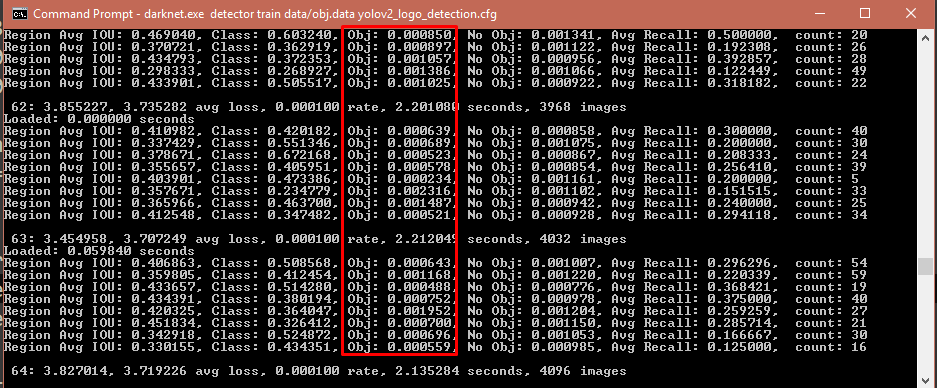
1. Type in “dark” and press TAB
2. Enter the parameters after “train”. [file containing folder paths for database] [configuration file] [OPTIONAL weights you want to start training from]

darknet.exe detector train data/obj.data yolo-obj.cfg yolo-obj\_2000.weights

1. Press enter. WARNING this step takes a really long time and will take most of your GPU resources. Make sure you won’t need it before continuing.

You should see this kind of view: 

At the start, you will not see that curve.

You shouldn’t let this column reach zero. 

# Post-Training validation

We will be using “TestYolo4Video” project, follow **“FrameCut” project setup** to set it up.

This application will calculate probabilities of objects in the video and display it. It will also generate output video. While testing outputs, you are free to organize them like you want.

1. Open [D:\LogoBackend\build\TestYolo4Video\Debug](file:///D:\LogoBackend\build\TestYolo4Video\Debug)
2. Open CMD
3. Paste in:

D:

Cd [D:\LogoBackend\build\TestYolo4Video\Debug](file:///D:\LogoBackend\build\TestYolo4Video\Debug)

1. Paste in the video to [D:\LogoBackend\build\TestYolo4Video\Debug](file:///D:\LogoBackend\build\TestYolo4Video\Debug) folder you want to test on.
2. Paste in **yolo\_[insert your config name here].cfg obj.names** and weights from **backup folder.**
3. If there is no “cfg.json” in this directory, create one. It should look like :

{

"DarkNet": {

"detectorName": "amazing",

"cfg": "yolov2\_logo\_detection.cfg",

"weights": "yolov2\_logo\_detection\_19600.weights",

"objectNamesPath": "obj.names"

},

"Video": {

"videoFilePath": "Euroleague Playoffs Game 3 Žalgiris Olympiacos LT 2018 04 24.mp4"

},

"Pictures": {

"pictureDirectory": "HereWeCanFindALotOfPictures"

}

}

1. Type in “Test” and press TAB
2. Then just simply press ENTER.
3. The detector will load your weights, and start detecting. You can stop at any time by pressing CTRL+C inside console. It will generate “results.avi” and “results.txt”.

My suggestion when you start testing, is to write down any problems while it is still detecting, to save up some time. Those problems could be:

* Does not detect at all
* Is very jittery
* Mixes up objects
* Detects trash
* Has hard time following moving objects
* And many more

1. Repeat the testing with weights from 16k-20k range.
2. Analyze results and make your conclusion. Is it good enough? If yes, which weights worked the best? If you are unsatisfied with the results, then it means that you need to change some things in database or change configuration. Consult **Database maintainance** for ways to improve your database and solve couple of problems.

# Database maintainance

## Database version tracking

It is wise to try keeping several versions of your database and tracking its changes. Although the bigger your database is the bigger file sizes will get. You could use your own way of tracking changes, but I suggest using our current method.

Go to [D:\LogoBackend\res\Database](file:///D:\LogoBackend\res\Database) . This is the place where it is saved and tracked.

Under “Current” folder there should be current database. You are free to make changes (such as removal of certain elements) and test it.

In “Backup” folder you should save database and weights before making additions. ALWAYS create a new folder with date of the change. Save this folder frequently to the cloud.

“Additions” folder is used for… Can you guess what? – Additions to the database. ALWAYS create a new folder with date of the change. In this folder have GT file from DarkLabel, Frame folder containing both jpg and txt files, and README file containing description of what you added.

## Object removing

Sometimes you can see from results that some object may be interfering with others or is not needed at all. Then you need to remove it from your database.

Do a backup **Database version tracking.** Use Pre-Training data validation step on your database. Find bad logo selections, and copy them to a different folder. Follow **“FrameCut” project setup** to set up “BadFrameRemover” project.

1. Open [D:\LogoBackend\build\BadFrameRemover\Debug](file:///D:\LogoBackend\build\BadFrameRemover\Debug) in file explorer
2. Open CMD

**D:**

**Cd D:\LogoBackend\build\BadFrameRemover Debug**

1. Press TAB
2. . Enter logoPath – Location of your bad logos; databasePath. Press ENTER.

This will remove all those logos from the database. Be warned at without a backup, this is a point of no return. Also you will lose WHOLE frame. So other logos will be lost too.

No go to **Training preparation.**

## Image augmentation

To increase the size of our database we can use augmentation. This can solve various problems as different lighting, noises, movement and many more. It just depends on what kind of augmentations you are willing to apply to your current database. Although it is better if you apply augmentations to additions and not database itself as constant augmentations to whole database will make several images which are corrupted.

WARNING THIS PART IS NOT USER FRIENDLY YET. Requires change of code. Change folder locations for augmented data paths and rebuild that project. Change which augmentations you want to use. DEPRECATED

1. Open [D:\LogoBackend\build\AugmentWholeImage\Debug](file:///D:\LogoBackend\build\AugmentWholeImage\Debug)
2. Open CMD
3. Paste in:

D:

Cd [D:\LogoBackend\build\AugmentWholeImage\Debug](file:///D:\LogoBackend\build\AugmentWholeImage\Debug)

1. Press TAB.
2. Enter path of your data to augment
3. Enter path where augmented date will be saved
4. Press ENTER
5. Follow further instructions
6. Wait really patiently, as it may take several minutes. You can see progress bar.
7. Do a backup **Database version tracking.**
8. Paste in all those new images to your **logoData folder.**
9. Retrain **Training preparation.**

# Conclusion

Now you know how to train neural network for image recognition using our provided tools.