FuzuliYolov3 Documentation

Folders and Their Usage

In the root there are some folders to hold related data. These folders are:

models:

All model files should be in here. To train a new model you should create a new folder under models and the new model folder should have a cfg file (name is not important), obj.names, train.txt and validate.txt. There must be only one cfg file in the model folder. Other files will be created after training. After training your model folder will look like this:

```
.
|-- somename.cfg
|-- best.pt
|-- best.weights
|-- data.data
|-- last.pt
|-- obj.names
|-- results.png
|-- results.txt
|-- train.shapes
|-- train.txt
|-- validate.shapes
`-- validate.txt
```

model_detect_outputs:

For each detection some folders will be created in this folder. In the model_detect_output there will be folders with model names and those folders will keep specific detection data folders with the name of modelname_imgsizeparameter. Also detection logs are created here.

model_test_outputs:

This folder keeps the logs of tests. File names are like modelname_logs_imgsizeparameter.txt .

test_files:

All test files like test videos, test images, test.data files etc. can be stored here.

pruned_models:

If you prune a model or draw the graphs of some pruned models their data will be created in this folder.

img_size_tests:

There is a feature to test models for a range of img-size parameters and then see the graph of precision, recall, mAP and F1. After using this feature the graphs and related data files will be created in this folder.

yolov3:

The original yolo files are stored here. They cloned as “git clone -b archive https://github.com/ultralytics/yolov3.git”.

yolov3-channel-and-layer-pruning:

This is a repo to prune models. It was cloned from https://github.com/tanluren/yolov3-channel-and-layer-pruning.

scripts:

In this folder there are only 2 files you will be interested in, config.cfg and general.py. Whatever you want to run you should edit the config.cfg and just run general.py. That's it! This is the usage of the FuzuliYolov3.

config.cfg file can contain whitespaces or comments. But do not change the first line of file.

Configuration file Usage

I will look over block by block. Some parameters take boolean value. You should write those in camel case form which Python uses for bools. Otherwise it will give an error.

[general]:

train = True
test = False
detect = True
prune = False
img-size-test = True

If the parameter is True, then the related script will run. The order of run is as given in the cfg file.

[train]:

model-path = Path of model

calculate_anchors = If this is True, anchors will calculate with train.txt
anchor-custer-num = If calculate_anchors is True, this number of anchors will be calculated
anchor-width = If calculate_anchors is True, this width value will be use to calculate
anchor-height = If calculate_anchors is True, this height value will be use to calculate

create-data-file = If this is True, automatically changes train, valid, names paths and classes parameters in the data file. If the data file does not exist, it will be created.
epochs = 30
batch-size = 8

img-size = Default is 320 640. Images are resizing to train with given size
weights = Switch resume to True if you give this. It is using for continuing to train with pre-trained model. Give the path of weight if you need
name = Don't change this parameter. It changes the name of weights and results files.
adam = Adam optimizer option

device = Empty quotes uses cuda if available. Other options are 'cpu' or 0,1,2 etc. to specify gpu
single-cls = If you are training a single class object detection model switch this to True.
freeze-layers = Freeze non-output layers
multi-scale = adjust (67%% - 150%%) img_size every 10 batches
rect = rectangular training
resume = resume training frome last.pt
nosave = Only save final checkpoint
notest = Only test final epoch
evolve = Evolve hyperparameters
bucket = gsutil bucket (I don't know what is this)
cache-images = Cache images for faster training

cfg = Don't give this. cfg file path will be found automatically
data = Don't give this. .data file will created and used automatically

[test]:
model-path = Path of model to find cfg, data, pt files
save-logs = If true the output won't print on sysout, will save as a txt file
batch-size = For testing it should be 1. Don't change
img-size = Images will resize with given parameter to a square
task = Must be 'test'. Don't change it.
data = test .data file path

device = Empty quotes uses cuda if available. Other options are 'cpu' or 0,1,2 etc. to specify gpu
single-cls = If you trained a single class object detection model switch this to True. If it is true the activation function of the final layer is sigmoid, otherwise softmax.
save-json = save a cocoapi-compatible JSON results file
augment = Augments images
conf-thres = confidence threshold
iou-thres = intersection of union threshold

cfg = Don't give this. cfg file path will be found automatically
weights = Don't give this. .weights file will created and used automatically

[detect]:
model-path = Path of model to find cfg, data, pt files
source = detection will use these data. image folder path or video path

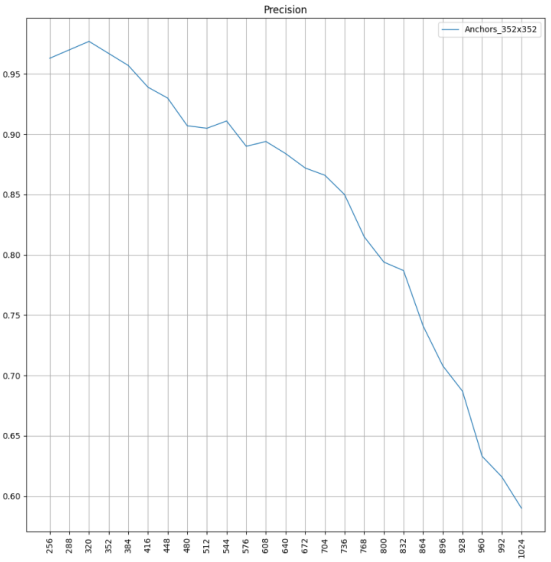
img-size = Images will resize with the given parameter. Letterbox method is using for resizing.
save-logs = If true the output won't print on sysout, will save as a txt file

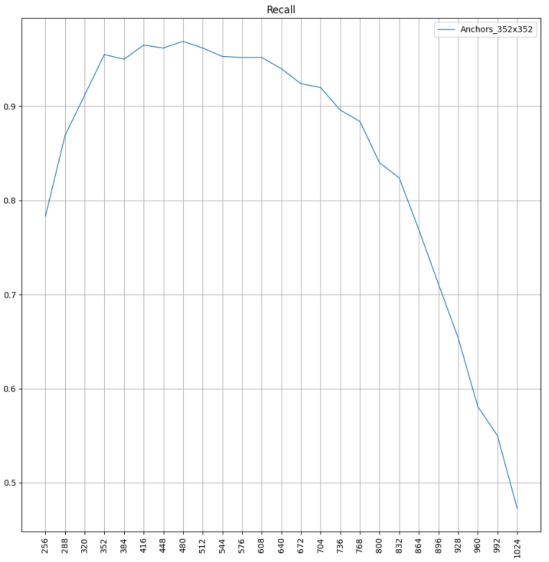
save-txt = founded objects bounding boxes are saving to a txt file
device = Empty quotes uses cuda if available. Other options are 'cpu' or 0,1,2 etc. to specify gpu
conf-thres = confidence threshold
iou-thres = intersection of union threshold
fourcc = Video format, don't change
half = half precision FP16 inference
view-img = Display results
agnostic-nms = Class agnostic-nms (I don't know what it is)
augment = Augments images

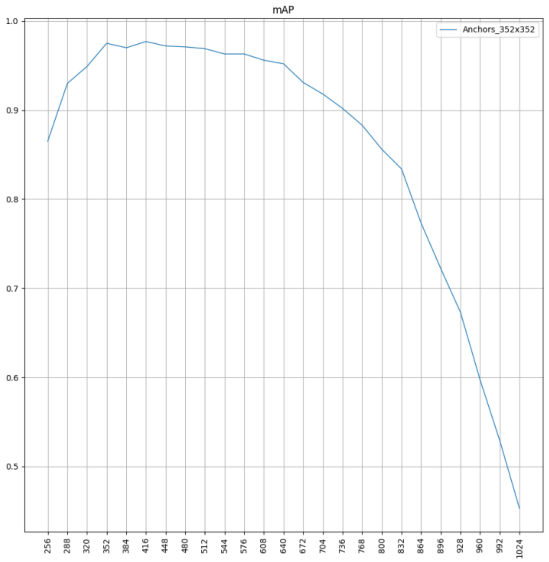
cfg = Don't give this. cfg file path will be found automatically
weights = Don't give this. weights file path will be found automatically
names = Don't give this. names file path will be found automatically
output = Don't give this. Defaultly script will create a folder as Yolov3_Archive/model_outputs/{modelnum}/output. Attention!! Files under this path will be removed first and then will create new files. Also it will save output logs (if save-logs = True) in Yolov3_Archive/model_outputs/{modelnum}/logs.txt

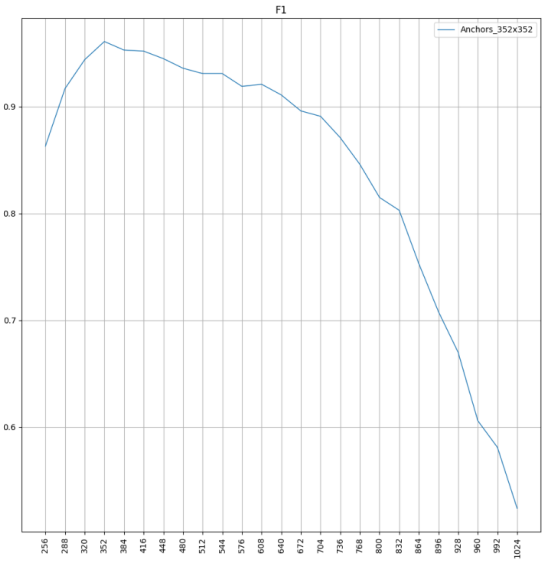
[prune]:
prune-step = Prune percentage increase each loop with given parameter
prune-start = Minimum prune percentage
prune-finish = Maximum prune percentage
prune-base-model = base model path to prune it
prune-graph-base-models = Model names to graph those. If you give a different model from the base model, be sure it exists.
prune-img-size = Images resizing to given size as a square
prune-only-draw-graph = If you already have pruned models and just

[img-size-test]:
img-size-test-graph-base-models = Model names to graph those. If you give a different model from the base model, be sure it exists. Also be sure all of them have same min-max-step values
img-size-test-base-model = base model to test it
img-size-test-min = Minimum image size to test
img-size-test-max = Maximum image size to test
img-size-test-step = Image size will increase each loop with given parameter
img-size-test-only-draw-graph = False
Image Size Test Example Output:

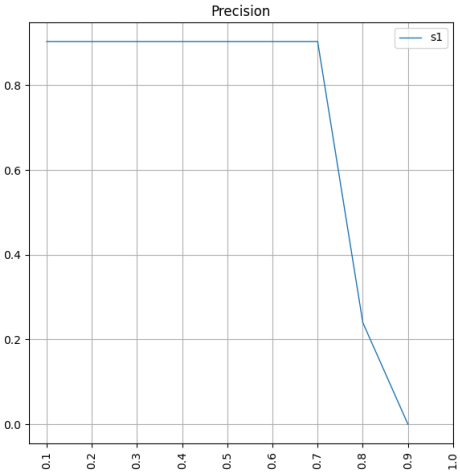


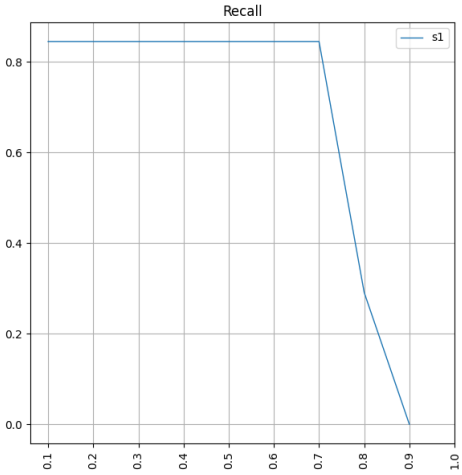


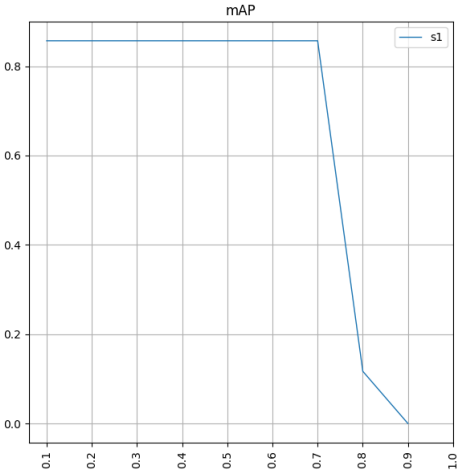


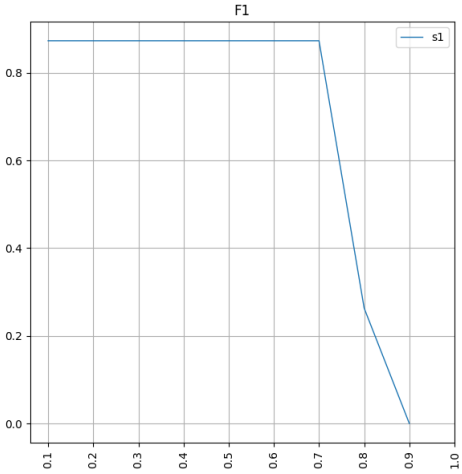


Pruning Example Output:


Precision
s1
0.0
0.2
0.4
0.6
0.8
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0


Recall
s1
0.0
0.2
0.4
0.6
0.8
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0


mAP
s1
0.0
0.2
0.4
0.6
0.8
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0


F1
s1
0.0
0.2
0.4
0.6
0.8
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0