YOLOv4 - 8 Tactics to Build a Better Model

The YOLO v4 model is currently one of the best architectures to use to train a custom object detector, and the capabilities of the Darknet repository are vast. In this post, we discuss ten advanced tactics in YOLO v4 so you can build the best object detection model from your custom dataset.

Note: this discussion assumes that you have already trained YOLO v4.

YOLO v4 Advanced Tactics RoadMap:

1. ***Gather More Data***

   We will look at 5 such ways of collecting data for training your custom model that solves your problem.

- Publicly available open labelled datasets.
  eg : ImageNet , COCO , Google's Open Image etc.
- **Scraping the Web (which i used for this Project)**
- Taking Photographs.
- Data Augmentation.
- Data Generation (Synthetic Data by GANs)

2. ***Image Preprocessing and Augmentation***
   Gathering and labeling more data is costly. Luckily, there are several ways to improve the scope and size of your training set through augmentation.
   Here is a list of tools that you can use for annotating images:

- MakeSense.AI **(which i used for this Project)**
- LabelImg
- VGG image annotator
- LabelMe
- Scalable
- RectLabel

3. Image Input Resolution Size

The input resolution determines the number of pixels that will be passed into the model to learn and predict from. A large pixel resolution improves accuracy, but trades off with slower training and inference time. Larger pixel resolution can help your model detect small objects.

4. When to Use Pretrained Weights

These weights have been pretrained on the COCO dataset, which includes common objects like people, bikes, and cars. It is generally a good idea to start from pretrained weights, especially if you believe your objects are similar to the objects in COCO.

5. Choosing a Model Size and Architecture

If inference speed is your end goal, the smaller model may be of interest. And it may be possible to pick up accuracy drops by improving your dataset.
Shifting from YOLOv4 to YOLOv4-tiny is a matter of model configuration.

6. Picking Up From a Previous Training Run

Be sure that the weights were trained with the same configuration file and trained under the same version of the Darknet framework!
Just like pretrained weights, you can pick up from previous training runs and apply your model's learnings from other datasets to new datasets.

7. Choosing the Best Model after Training

Darknet model weights are saved in the `backup` folder and automatically save every 1000 iterations. Darknet will automatically save the best model for you with extension `__best.weights`. Darknet will also save the last weights from your training run with extension `_last.weights`.

8. Track Your Model Evaluations

As you are working through various tweaks in your dataset and in YOLOv4 modeling, it can be useful to track your results, so you can see what combination of techniques works best on your dataset.
Darknet will output a mean average precision score, which is the primary metric to track as you decide which model is working best on your data. Another important metric to consider is the speed of inference.

Reference : https://www.youtube.com/watch?v=GUw6at85un8