Tips for Best Training Results

📚 This guide explains how to produce the best mAP and training results with YOLOv5 🚀.

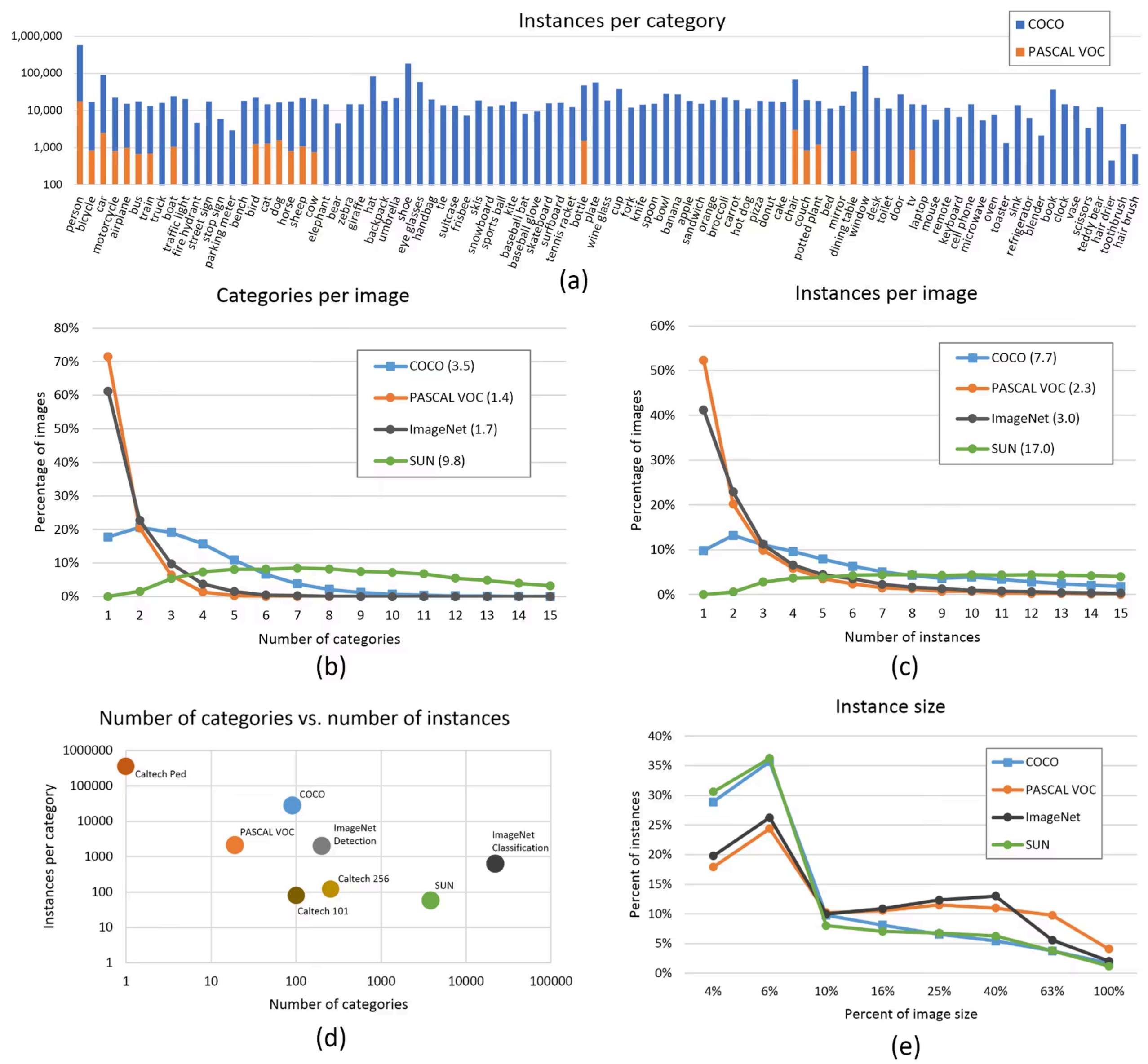
Most of the time good results can be obtained with no changes to the models or training settings, **provided your dataset is sufficiently large and well labelled**. If at first you don't get good results, there are steps you might be able to take to improve, but we always recommend users **first train with all default settings** before considering any changes. This helps establish a performance baseline and spot areas for improvement.

If you have questions about your training results **we recommend you provide the maximum amount of information possible** if you expect a helpful response, including results plots (train losses, val losses, P, R, mAP), PR curve, confusion matrix, training mosaics, test results and dataset statistics images such as labels.png. All of these are located in your `project/name` directory, typically `yolov5/runs/train/exp`.

We've put together a full guide for users looking to get the best results on their YOLOv5 trainings below.

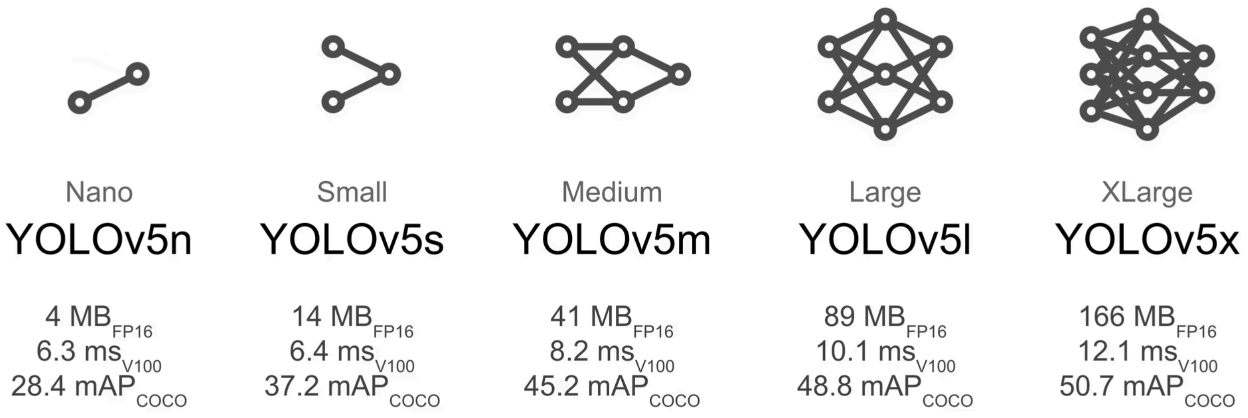
Dataset

- **Images per class.** ≥ 1500 images per class recommended
- **Instances per class.** ≥ 10000 instances (labeled objects) per class recommended
- **Image variety.** Must be representative of deployed environment. For real-world use cases we recommend images from different times of day, different seasons, different weather, different lighting, different angles, different sources (scraped online, collected locally, different cameras) etc.
- **Label consistency.** All instances of all classes in all images must be labelled. Partial labelling will not work.
- **Label accuracy.** Labels must closely enclose each object. No space should exist between an object and it's bounding box. No objects should be missing a label.
- **Label verification.** View `train_batch*.jpg` on train start to verify your labels appear correct, i.e. see example mosaic.
- **Background images.** Background images are images with no objects that are added to a dataset to reduce False Positives (FP). We recommend about 0-10% background images to help reduce FPs (COCO has 1000 background images for reference, 1% of the total). No labels are required for background images.



Model Selection

Larger models like YOLOv5x and YOLOv5x6 will produce better results in nearly all cases, but have more parameters, require more CUDA memory to train, and are slower to run. For **mobile** deployments we recommend YOLOv5s/m, for **cloud** deployments we recommend YOLOv5l/x. See our README table for a full comparison of all models.



- **Start from Pretrained weights.** Recommended for small to medium-sized datasets (i.e. VOC, VisDrone, GlobalWheat). Pass the name of the model to the `--weights` argument. Models download automatically from the latest YOLOv5 release.

```
python train.py --data custom.yaml --weights yolov5s.pt
                                             yolov5m.pt
                                             yolov5l.pt
                                             yolov5x.pt
                                             custom_pretrained.pt
```

- **Start from Scratch.** Recommended for large datasets (i.e. COCO, Objects365, OIv6). Pass the model architecture YAML you are interested in, along with an empty `--weights ''` argument:

```
python train.py --data custom.yaml --weights '' --cfg yolov5s.yaml
                                                      yolov5m.yaml
                                                      yolov5l.yaml
                                                      yolov5x.yaml
```

Training Settings

Before modifying anything, **first train with default settings to establish a performance baseline**. A full list of train.py settings can be found in the train.py argparser.

- **Epochs.** Start with 300 epochs. If this overfits early then you can reduce epochs. If overfitting does not occur after 300 epochs, train longer, i.e. 600, 1200 etc. epochs.

- **Image size.** COCO trains at native resolution of `--img 640`, though due to the high amount of small objects in the dataset it can benefit from training at higher resolutions such as `--img 1280`. If there are many small objects then custom datasets will benefit from training at native or higher resolution. Best inference results are obtained at the same `--img` as the training was run at, i.e. if you train at `--img 1280` you should also test and detect at `--img 1280`.
- **Batch size.** Use the largest `--batch-size` that your hardware allows for. Small batch sizes produce poor batchnorm statistics and should be avoided.
- **Hyperparameters.** Default hyperparameters are in hyp.scratch-low.yaml. We recommend you train with default hyperparameters first before thinking of modifying any. In general, increasing augmentation hyperparameters will reduce and delay overfitting, allowing for longer trainings and higher final mAP. Reduction in loss component gain hyperparameters like `hyp['obj']` will help reduce overfitting in those specific loss components. For an automated method of optimizing these hyperparameters, see our Hyperparameter Evolution Tutorial.

Further Reading

If you'd like to know more, a good place to start is Karpathy's 'Recipe for Training Neural Networks', which has great ideas for training that apply broadly across all ML domains: https://karpathy.github.io/2019/04/25/recipe/

Good luck 🍀 and let us know if you have any other questions!

📅 Created 10 months ago ✏️ Updated 3 days ago