PP- Yolo

PP-YOLO does not introduce a new way of designing object detection models, however, it provides a valuable case study of which tricks known from a wider field of deep learning work well in the context of one-stage object detectors.

5 reasons of improvement of pp-yolo over yolov4

Batch size:

In order to improve the model performance, the authors of the paper experimented with a batch size of 192. Training the neural network with a larger batch size leads to a boost in the accuracy of the model by stabilizing the training process.

Exponential moving average:

Exponential moving average algorithms are quite common in the financial analytics domain. Using a similar methodology, the authors improved the performance of the model by smoothing out the weights during the training phase. This process is also very memory efficient since at every update step only the value of the smoothing factor needs to be kept in memory. This value was set to 0.9998 in the paper.

Dropblock:

One way to improve the accuracy of the model and ensure that the model does not overfit to the training data is to use regularization techniques like adding noise or using a dropout layer. Dropout layers randomly drop or ignore some nodes while training the neural network. This process can be made even more robust by dropping neurons in a continuous region as opposed to doing it independently. This ensures that the model can generalize well on unseen data.

Iou loss:

Neural networks have a loss function to make the network learn and adapt. In the original YOLOv3 paper only the L1 loss was used and adapted for estimating the bounding box coordinates. IoU or Intersection over union indicates what percentage of the predicted box coordinates overlaps with the ground-truth bounding box coordinates. The authors from the BAIDU research team experimented with both types of losses in the training process. The authors of the paper decided to stick with the basic version of the IoU loss since it gives good results in estimating the mAP metric.

pre-trained models:

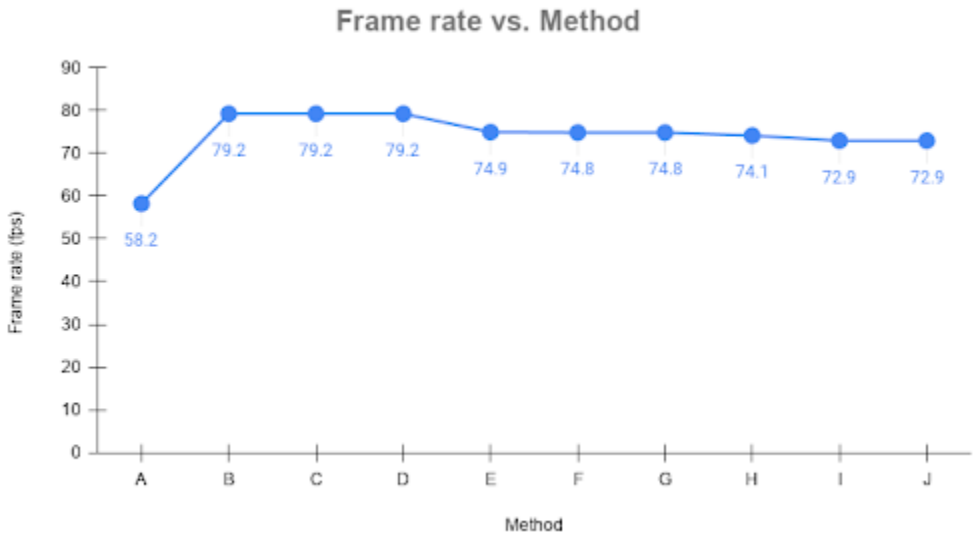
Using pre-trained models and transferring the knowledge learned from one domain to another domain is a common trick to improve the speed of training the model and the effectiveness of the model. The authors used a better set of pre-trained weights for the

backbone architecture which allowed them to improve the final detection accuracy even more.

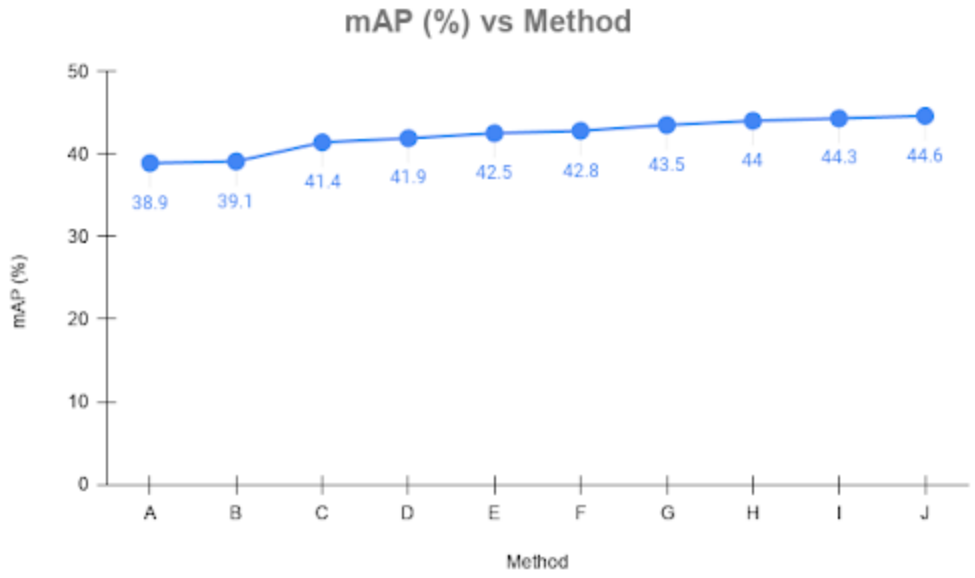
How above methods impact the following:

1. Frame rate
2. Mean average precisions
3. Interference time

Frame rate:
Computer vision applications using edge hardware (as opposed to outsourcing the computational heavy lifting to the cloud) need to have a very low latency in processing the individual frames extracted from a live video stream. Changing the backbone architecture from Darknet-53 to ResNet leads to a significant increase in the number of frames that can be processed in a second. The other configurations applied later do slow the algorithm slightly, however, it remains significantly faster than the benchmark.



Mean average precisions:
Let us now see how the mAP is affected by the change in the base configuration.



As you can see there is a 5.7% increase (that is a 14.6% relative improvement!) in the mAP score from the base configuration to the sophisticated configuration (denoted by J).

Interference time:
Building models and optimizing them to run as fast as possible requires a fair bit of engineering and tweaking. As you can see in the plot below, the time it takes for the model to make the predictions is drastically reduced from 17.2 ms with the original darknet architecture to 13.7 ms with ResNet as a backbone architecture and the other configurations applied.