SlimYOLOv3:Narrower, Faster and Better for Real-Time UAV Applications (2019) 최종정리

Abstarct

- UAV의 카메라 시점에 의한 컴퓨터 비젼에 관심이 많아짐
- 하지만 UAV에서의 object detection을 실시간으로 처리하는 작업은 어려움
- 마지막에는 `channel scaling factor`에 L1 Regulation을 적용시켜서 convolution 계층의
  channel-level 희소성(sparsity)을 적용시키고 ‘slim’한 object detectors을 얻기 위해서 덜 중요한 feature channel을 제거함
- fewer trainable parameters와 FLOPS를 original YOLOv3와 비교하여 제시함
- SlimYolov3를 VisDrone2018-Det benchmark dataset 으로 평가
- unpruned 놈과 비교했을 때 FLOPS가 90.8%줄어들었고, parameter크기가 92% 줄었고
  running time이 2배 더 빠르다.

다른 pruning 비율에 대한 실험 결과는 더 작은 구조의 SlimYOLOv3가 기존 YOLOv3보다 더 효율적이고 빠르다고 일관적으로 확인해준다.

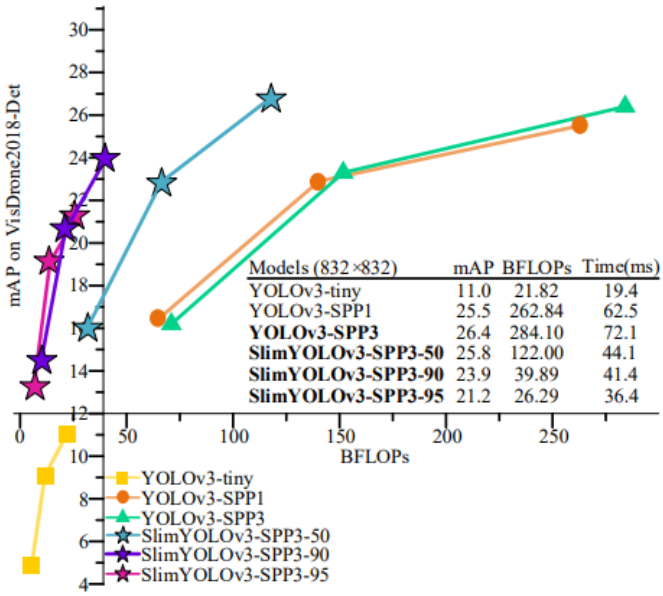


Figure 1. Billion floating point operations (BFLOPs) versus accuracy (mAP) on *VisDrone2018-Det* benchmark dataset. Enabled by channel pruning, our SlimYOLOv3-SPP3 achieves comparable detection accuracy as YOLOv3 but only requires the equivalent floating point operations as YOLOv3-tiny. Such performance is very competitive in drone applications. Details are given in §5.

Introduction

challengin problems

(1) 어떻게 다양한 변화를 다룰것인가 (항공 사진에서 객체의 시각적 외형에 대한)

(variation : illumination, view, small sizes and ration)

(2) 제한된 메모리와 computing power을 가진 기기에 어떻게 deploy할지

(3) detection 정확도와 실시간 요구에 대한 균형을 어떻게 맞출지

주요 인기있는 네트워크 구조

- 수동적(manually)으로 설계되었고,
- 그 구조들에서 각 구성요소의 중요성은 학습되기 전에 결정될 수 없음
- Training 과정에서 network는 학습가능한 계층의 weigth를 조절해가면서 각 component의 중요성을 학습할 수 있음

- 결과적을 network안의 몇몇 connection과 computations들은 불필요하거나 중요하지 않는 요소들이 될 수있고 그러므로 성능에서 큰 저하 없이 제거될 수 있다.

→ 이러한 추정으로 `model pruning` 방식은 모델을 단순화시키는데 사용됨

▼ 논문

> A typical deep learning pipeline briefly involves designing network structures, fine-tuning hyper-parameters, training and evaluating network. The majority of popular network structures (e.g., ResNet and DenseNet) are designed manually, in which the importance of each component cannot be determined before training. During the training process, network can learn the importance of each component through adjusting the weights in trainable layers automatically. Consequently, some connections and computations in the network become redundant or non-critical and hence can be removed without significant degradation in performance [17]. Based on this assumption, many model pruning methods have been designed recently to simplify deep models and facilitate the deployment of deep models in practical applications. Channel pruning is a coarse-grained but effective approach, and more importantly,

Channel Pruning

- pruned model에 단지 구성 파일들 안의 대응되는 channel(or filter)의 갯수를 수정함으로써
  적용하기 편리함

fine-tuning

- 그 후에 잠재적 일시적 저하를 보충하기 위해서 pruned models에서 수행됨

→ 전문가에 의해 수동적으로 설계된 deep object detectors는 고유의 불필요한 중복성이 feature channels에 존재한다고 생각해서 parameter 크기와 FLOPS를 channel pruning으로 줄이려고 함

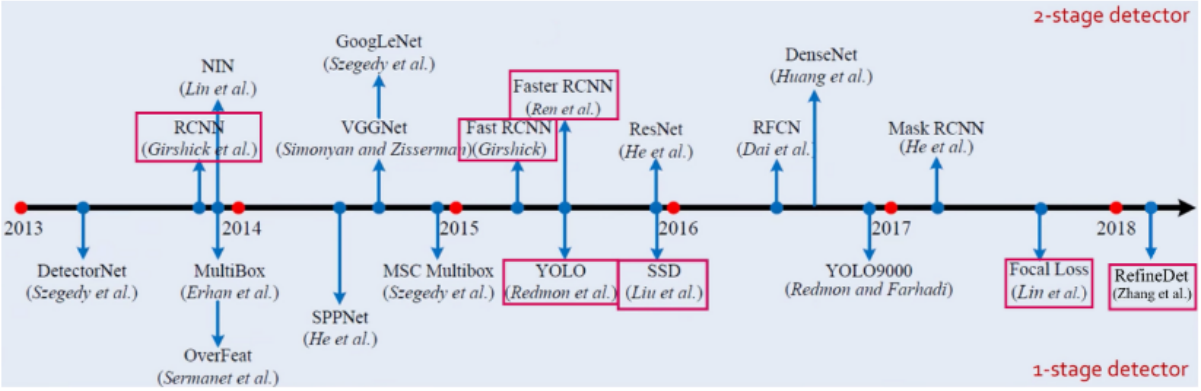
▼ 논문

> practical applications. Channel pruning is a coarse-grained but effective approach, and more importantly, it is convenient to implement the pruned models just by modifying the number of corresponding channel (or filter) in configuration files. A fine-tuning operation is subsequently performed on pruned models to compensate potentially temporary degradation. We empirically argue that deep object detectors designed by experts manually might exist inherent redundancy in feature channels, thus making it possible to reduce parameter size and FLOPs through channel pruning.

2. Related Work

2.1 Deep Object Detector

발전되면서 two-stage detectors와 single-stage detector로 구분됨



2-stage Detector

- Regional Proposal과 Classification이 순차적으로 이뤄짐
- Regional Proposal 진행 후 Classification 진행
- 비교적 느리지만 정확도가 높음

Regional Proposal

- 물체가 있을만한 영역을 빠르게 찾아내는 알고리즘
- ex) Selective search : 비슷한 질감, 색, 강도를 갖는 인접 픽셀로 구성된 다양한 크기의 window를 생성)
- object의 위치를 찾는 localization 문제

(기존의 sliding window의 방식은 모든 영역을 window로 탐색)

1-stage Detector

- Regional Proposal과 Classification이 동싱 이뤄짐
- ex) YOLO, SSD 계열
- 비교적 빠르지만 정확도가 낮음

2.2 Model pruning

존재하는 model compression(모델 경량화) 방식은

1. **model pruning**
2. knowledge distillation
3. parameter quantization
4. dynamic computation

Model Pruning

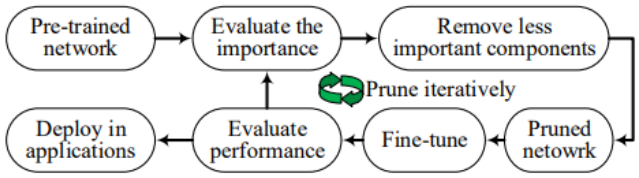


Figure 2. A representative procedure of incremental model pruning. There exists four iterative steps: (1) evaluating importance of each component in a pre-trained deep model; (2) removing the components that are less important to model inference; (3) fine-tuning pruned model to compensate potentially temporary degradation in performance; (4) evaluating the fine-tuned model to determine whether pruned model is suitable for deployment. An incremental pruning strategy is preferred to prevent over-pruning.

model pruning 방식에서 deep models들로부터 제거되는 components는

독립적인 neural connection 혹은 네트워크 구조들 자체가 될 수 있음

Weight Pruning

- small weights로 덜 중요한 연결을 제거함(prune)
- 개념적으로 이해하기 쉽지만, pruned model을 기억하고(store) 빠르게 하는것이 어려움

→ 생성된 **불규칙한 네트워크 구조** 때문

- 기술적으로 weight pruning은 특별한 소프트웨어 라이브러리 혹은 그에 맞는 하드웨어가 pruned model을 지원하도록 설계되지 않는다면, 실제 applications에 적합하지 않을 수 있음

▼ 논문

structures [17][24]. Weight pruning methods prune the less important connections with small weights. It is conceptually easy to understand, but it is hard to store the pruned model and speed up due to the generated irregular network architecture. Technically, weight pruning might not be suitable for practical applications unless special software library or dedicated hardware is designed to support the pruned model. Unlike weight pruning, structured pruning is more likely to produce regular and tractable network

Structured pruning(channel pruning)

- Weight pruning과 달리 structured pruning이 **더 규칙적이고 다루기 쉬운 네트워크 구조**를 생성할 가능성이 높음
- structured pruning을 위한 structured unimportance를 얻기 위해서 연구원들은 structured sparsity learning을 포함하고 있는 structured sparsity regularization을 가진 `sparsity training` 과 `channel-wise scaling factors의 sparsity` 에 의지함
- network slimming - 효율적인 channel pruning방식

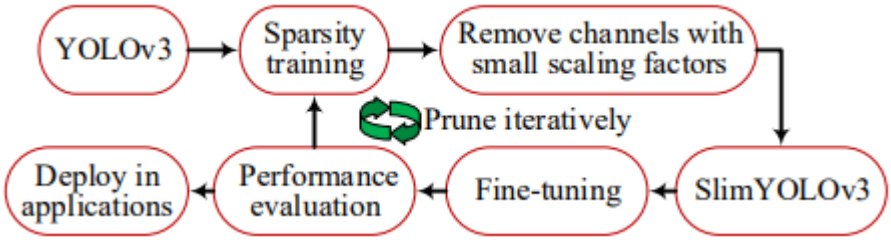
network slimming

- `batch normalization`의 `scaling factors` 을 `channel-wise scaling factors`로서 채택함
- 이 `scaling factors` 에 `L1 regularization`을 사용하여 networks를 학습시킴
- hardware과 software에 한정되지 않고 pruned models를 실행하기 편함

▼ 논문

model. Unlike weight pruning, structured pruning is more likely to produce regular and tractable network architectures. To obtain structured unimportance for structured pruning, researchers resort to sparsity training with structured sparsity regularization involving structured sparsity learning [29] and sparsity on channel-wise scaling factors [17][24]. Liu et al. [24] proposed a simple but effective channel pruning approach called network slimming. They directly adopted the scaling factors in batch normalization (BN) layers as channel-wise scaling factors and trained networks with L1 regularization on these scaling factors to obtain channel-wise sparsity. Channel pruning is a coarse-grained but effective approach, and more importantly, it is convenient to implement the pruned models without the requirements of dedicated hardware or software. They applied network slimming methods to prune CNN-based image classifiers and notably reduced both model size and computing operations. In this paper, we follow Liu's work and extend it to be a coarse-grained method of neural architecture search for efficient deep object detectors.

3. SlimYOLOv3



- channel pruning을 YOLOv3에 적용하여 SlimYOLOv3을 얻음
- 최소한의 수정으로 deep feature을 풍부하게(enrich)하기 위해서 spatial pyramid pooling (SPP) module을 YOLOv3에 적용한 YOLOv3-SPP3가 있음

YOLOv3-SPP3

- SPP 모듈은 kernel size가 각각 1x1, 5x5, 9x9, 13x13 의 크기인 4개의 병렬 maxpool 계층으로
  구성되어 있음
- SPP 모듈은 다른 receptive 영역을 가진 multiscale deep features을 추출하고 feature maps의 channel dimension크기로 연결하여 이들을 통합할 수 있음
- 같은 layer 내에서 얻은 multiscale features은 적은 computation cost로
  YOLOv3의 detection 정확도를 더 개선시킬것이라고 기대함
- SPP 모듈에서 소개 된 additional feature channels은 추가적인 FLOPS를 줄이거나 refined 할 수 있음
- SPP module을 YOLOv3의 각 detection header 앞의 5번째와 6번째 convolution 계층 사이에 넣어 통합함

Sparcity training

- 모델의 channel-wise sparsity는 channel pruning과
  후에 제거될 잠재력이 있는 덜 중요한 channel들의 수를 설명하는데 도움이 됨
- channel pruning을 사용하기 위해서 `scaling factors`을 각 channel마다 할당함
- `scaling factors`의 절댓값이 channel의 중요도를 나타냄(denote)
- detection header를 제외하고 BN layer는 YOLOv3의 각 convolution layer 후 적용

▼ 논문

> **Sparsity training**. Channel-wise sparsity of deep models is helpful to channel pruning and describes the number of less important channels that are potential to be removed afterwards. To facilitate channel pruning, we assign a scaling factor for each channel, where the absolute values of scaling factors denote channel importance. Specifically, except for detection headers, a BN layer to accelerate convergence and improve generalization follows each convolutional layer in YOLOv3. BN layer normalize convolutional features using mini-batch statics, which is formulated as formula (1).

BN Layer

$$y = \gamma \times \frac{x - \bar{x}}{\sqrt{\sigma^2 + \varepsilon}} + \beta \quad (1)$$

- γ : `trainable scale factors`
- β : bias
- σ^2 : mini-batch안의 input features의 variance(변화량)
- $\bar{x}$: mini-batch안의 input features의 mean(평균)

BN Layer들 안의 trainable한 `scale factors`은 channel의 중요도를 나타나는데 사용됨

L1 normarlization

- 중요하지 않는 channel들로 부터 중요한 channel을 효과적으로 구분하기 위해서, γ(trainable scale factor)에 대해서 L1 regularization을 부과함으로써 channel-wise sparsity training을 수행함

$$L = loss_{yolo} + \alpha \sum_{\gamma \in \Gamma} f(\gamma) \quad (2)$$

- $f(\gamma) = abs(\gamma)$: L1-norm
- α : penalty factor (두 loss 사이의 간격을 균형맞추는 요소)

Channel pruning

- $\hat{\gamma}$: gloab threshold - feature channel이 제거될지 말지 결정
- pruning ratio를 통제하기 위해서$\hat{\gamma}$ 는 모든 $abs(\gamma)$에 대해서 n-th percentile로 설정됨
- π : safety threshold
  - convolution 계층에서의 over-pruning을 예방하고 network연결의 완전성을 유지하기 위함
  - 특정 layer의 모든 $abs(\gamma)$의 k-th percentile로 층 마다 설정
- $\hat{\gamma}$ 의 최소와 π 보다 작은 scaling factors를 가진 feature channels를 제거
- maxpool layer과 upsample layer은 channel수와 연관이 없기 때문에 pruning 과정에서 버림
  (안쓴다는 의미인듯)

fine-tuning

- channel pruning 이후 잠재적 일시적 저하를 보상하기 위해서 pruned models에 수행되도록
  제안됨

Iteratively pruning

- over-pruning을 방지하기 위해 incremental pruning

4. Experiments

4.1 Datasets

VisDrone2018-Det dataset consists of 7,019 static images captured by drone platforms in different
places at different height

4.2 Models

- YOLOv3-SPP1 : 수정된 YOLOv3로 첫 번 째 detector header의 앞에 SPP module이 1개
  - YOLOv3보다 COCO 데이터셋에 대해서 detection accuracy가 좋음
- YOLOv3-SPP3 : YOLOv3의 3개의 detector header 앞에 5th와 6th convolution 사이 3개의 SPP module을 통합(incorporating)하여 구현됨
- SlimYOLOv3
  - channel pruning module의 global threshold $\hat{\gamma}$를 모든 $abs(\gamma)$의 50%, 90%, 95%의 pruning ration를 각각 적용한 3개의 모델을 구현
  - local safety threshold π는 90% 적용한 모델의 각 layer에서 경험적으로 적용 (단일 layer에 unpruned된 channel을 최소 10% 유지하기 위해)

4.3 Training

Sparsity training

- 100epochs 의 학습진행
- penalty factor α에 대해 0.01, 0.001, 0.0001 세 개의 다른 값에 대해 실험

▼ 논문

4.3. Training

Normal training. Following the default configurations in Darknet [16], we train YOLOv3-tiny, YOLOv3 and YOLOv3-SPP3 using SGD with the momentum of 0.9 and weight decay of 0.0005. We use an initial learning rate of 0.001 that is decayed by a factor of 10 at the iteration step of 70000 and 100000. We set the maximum training iteration as 120200 and use mini-batch size of 64. We set the size of input image as 416 for YOLOv3-tiny and 608 for YOLOv3 and YOLOv3-SPP3. Multiscale training is enabled by randomly rescaling the sizes of input images. We initialize the backbone networks of these three models with the weights pre-trained on ImageNet [4].

Sparsity training. We perform sparsity training for YOLOv3-SPP3 for 100 epochs. Three different values of penalty factor α, i.e., 0.01, 0.001 and 0.0001, are used in our experiments. The remaining hyper-parameters of sparsity training are same as normal training.

5. Results and Discussion

SPP Module의 효과

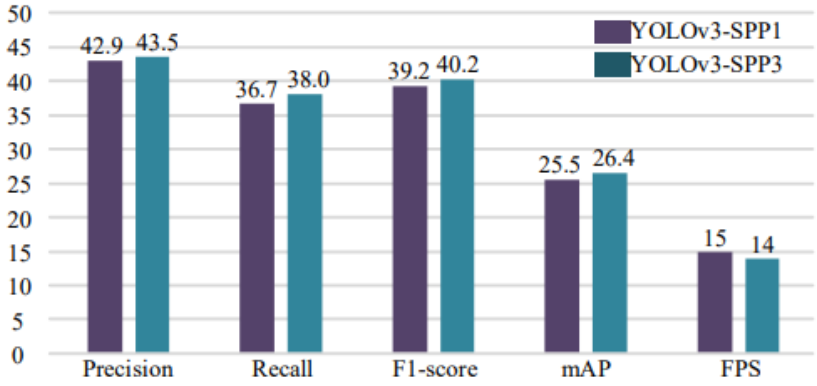


Figure 5: Performance comparison of YOLOv3-SPP1 and YOLOv3-SPP3 with input size of 832 ×832.

Table 1. Evaluation results of baseline models and pruned models.

Model	Input size	Precision	Recall	F1-score	mAP	BFLOPS	FPS	Inference time (ms)	Parameters	Volume
YOLOv3-tiny	416	19.5	10.5	13.1	4.9	5.46	134	7.5	8.7M	33.1MB
	608	24.1	16.8	19.1	9.1	11.65	80	12.5		
	832	23.4	20.1	21.0	11.0	21.82	52	19.4		
YOLOv3-SPP1	416	39	24.5	29.5	16.5	65.71	46	21.7	62.6M	239MB
	608	44.2	32.4	36.9	22.9	140.36	26	38.6		
	832	42.9	36.7	39.2	25.5	262.84	15	67.9		
YOLOv3-SPP3	416	36.6	24.9	29.1	16.2	71.03	40	24.8	63.9M	243MB
	608	43.7	33.7	37.6	23.3	151.72	23	43.1		
	832	43.5	38	40.2	26.4	284.10	14	72.1		
SlimYOLOv3-SPP3-50	416	39.2	23.5	28.7	15.7	30.51	67	14.9	20.8M	79.6MB
	608	45.6	32.1	37.1	22.6	65.17	39	25.6		
	832	45.9	36	39.8	25.8	122	23	44.1		
SlimYOLOv3-SPP3-90	416	32.2	21.6	24.4	14.5	9.97	67	14.8	8.0M	30.6MB
	608	37.9	30.0	32.0	20.6	21.3	40	25.1		
	832	**36.9**	**33.8**	**34.0**	**23.9**	**39.89**	**24**	**41.4**		
SlimYOLOv3-SPP3-95	416	33.8	20.1	22.9	13.3	6.57	72	13.8	5.1M	19.4MB
	608	37.3	28.2	30.1	19.1	14.04	41	24.1		
	832	36.1	31.6	32.2	21.2	26.29	28	36.4		

- 416x416, 608x608에 대해서 SPP3와 SPP1을 비교했을 때 detection performance가 성능차이가 거의 안났지만 더 큰 input인 832x832일 경우에 mAP와 F1-score에 대해서 차이가 outperforms할 수 있는것을 확인할 수 있음

→ **SPP 모듈**이 **고해상도 이미지안**에서의 다른 크기의 receptive fields를 통해서 **유용한 multiscale deep feature를 detectors가 추출하도록 도와준**다는 것임

- SPP module이 추가되면서 trainable parameter과 FLOPS는 증가되지만 무시할 수 있는 수치

Sparsity training의 효과

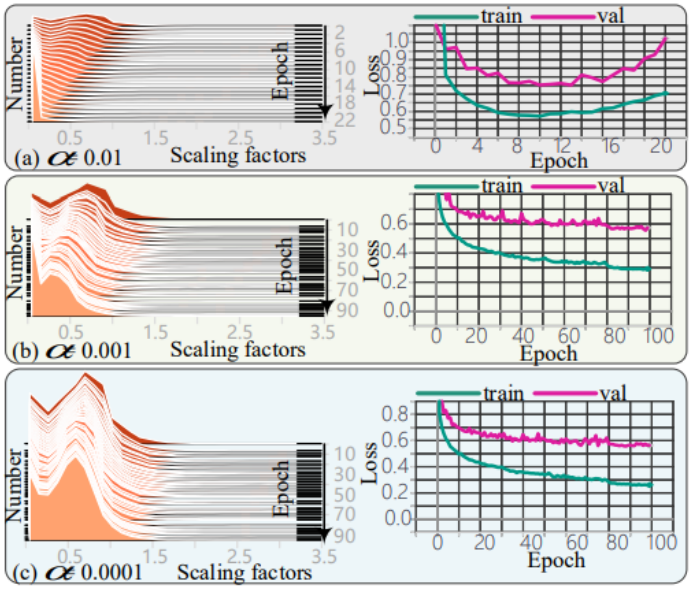


Figure 7. Histogram statistics of scaling factors in all BN layers (left) and loss curve of training and validation sets (right) during sparsity training of YOLOv3-SPP3 with three different values of α including 0.01, 0.001 and 0.0001. In (a), we terminate the sparsity training early when the model get stuck in underfitting.

→ Scaling factor의 분포의 변화와 YOLOv3-SPP3의 sparsity training 동안의 loss 그래프

- 학습 과정에서 상대적으로 작은 `scaling factors`의 수가 증가하는 반면 큰 `scaling factors`의 수는 감소함
- sparsity 학습은 `scaling factors` 값을 효율적으로 줄일 수 있고 convolution layers의 feature channel을 부족하게 만듬
- $\alpha = 0.01$인 경우 `scaling factors`을 너무 급하게 감소시켜서 모델이 underfitting으로 되버림
- 실험에서는 $\alpha = 0.001$ 로 진행

▼ 논문

Effect of sparsity training. During the sparsity training, we compute the histogram of scaling factors (absolute value) in all BN layers of YOLOv3-SPP3 to monitor change in the distribution of scaling factors. We visualize these histograms as well as the loss curves of training and validation sets in Figure 7. With the training progress, the number of smaller scaling factors increases while the number of larger factors decreases. Sparsity training is able to effectively reduce the scaling factors and thus make the feature channels of convolutional layers in YOLOv3-SPP3 sparse. However, sparsity training with a larger penalty factor, i.e., α = 0.01, make the scaling factors decay so aggressive that models start failing with underfitting. In our experiments, we use the YOLOv3-SPP3 model trained with penalty factor α = 0.0001 to perform channel pruning.

channel pruning의 효과

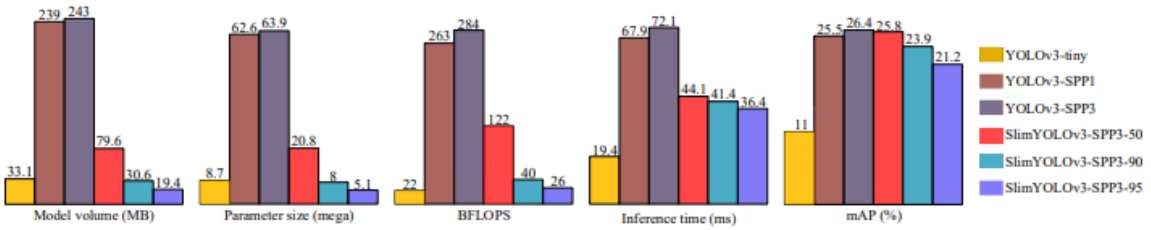


Figure 6. Comparison of baseline models and our SlimYOLOv3 models in model volume, parameter size, BLOPs, inference time and mAP score when input size is 832×832.

- FLOPS와 parameter size를 줄임