UNIVERSITY of HOUSTON
CULLEN COLLEGE of ENGINEERING

ANTS LAB

YOLObile: Real-Time Object Detection on Mobile Devices via Compression-Compilation Co-Design

Yuxuan Cai[1]*, Hongjia Li[1]*, Geng Yuan[1]*, Wei Niu[2] , Yanyu Li[1] , Xulong Tang[3] , Bin Ren[2] , Yanzhi Wang[1]

[1]Northeastern University, [2]Willian & Mary, [3]University of Pittsburgh

Presented by Rui Chen

AAAI 2021

Outline

- Motivation
- Framework Overview
  – Block-punched Pruning
  – Compiler-assisted Acceleration
  – Mobile GPU-CPU Collaborative Scheme
- Experimental Results

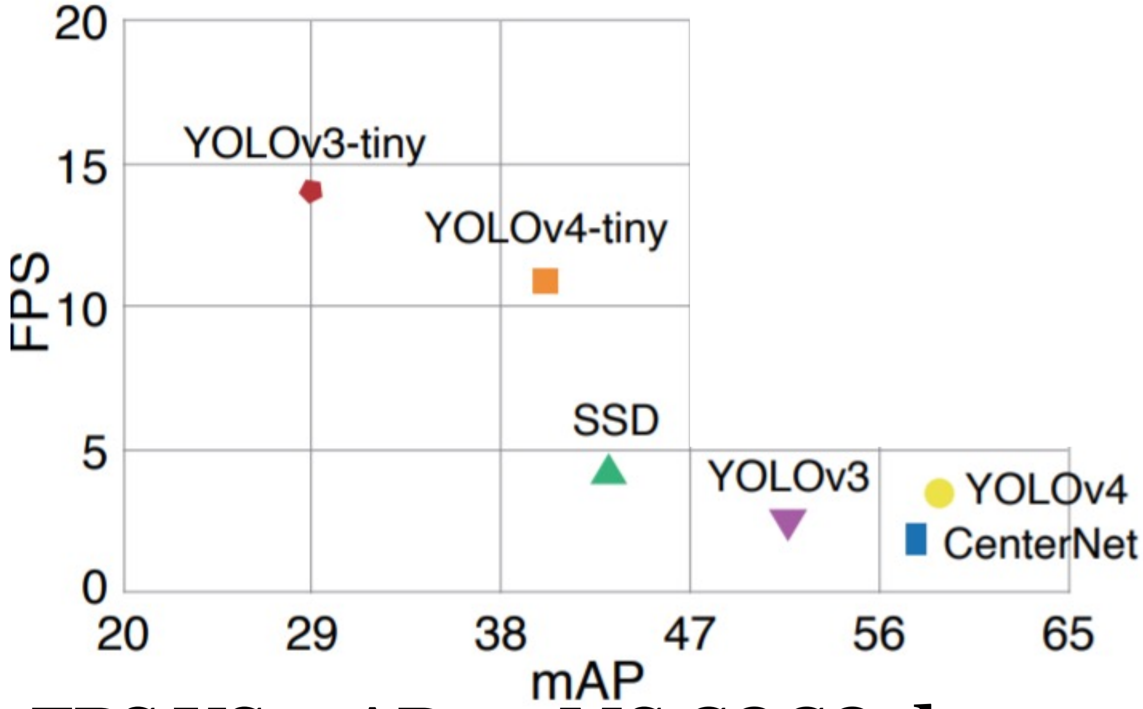
Background: Object Detection

- Autonomous Driving
- UAV Obstacle Avoidance
- Augmented Reality
- Robot Vision

Background: Object Detection

FPS: frames per second
mAP: mean average precision

- Executing DNNs inference on mobile device is still challenging
  - High computation and storage demands
  - Cannot achieve real-time performance with high accuracy



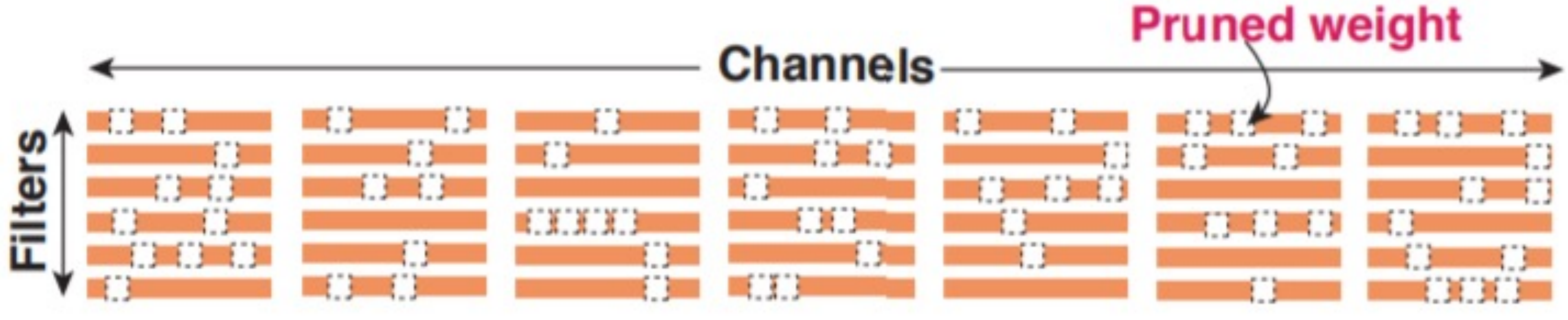
FPS VS mAP on MS COCO dataset

ANTS LAB

UNIVERSITY of HOUSTON
CULLEN COLLEGE of ENGINEERING

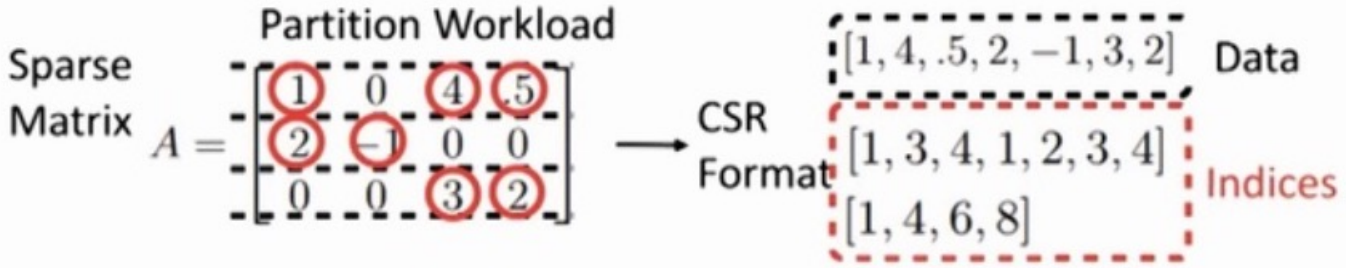
Background: Pruning

- **Non-structured** weight pruning: arbitrary weight can be pruned



Model weights in one layer

- Limited actual development
  - Indices are required for sparse format – **speed degradation** in GPU/CPU



ANTS LAB

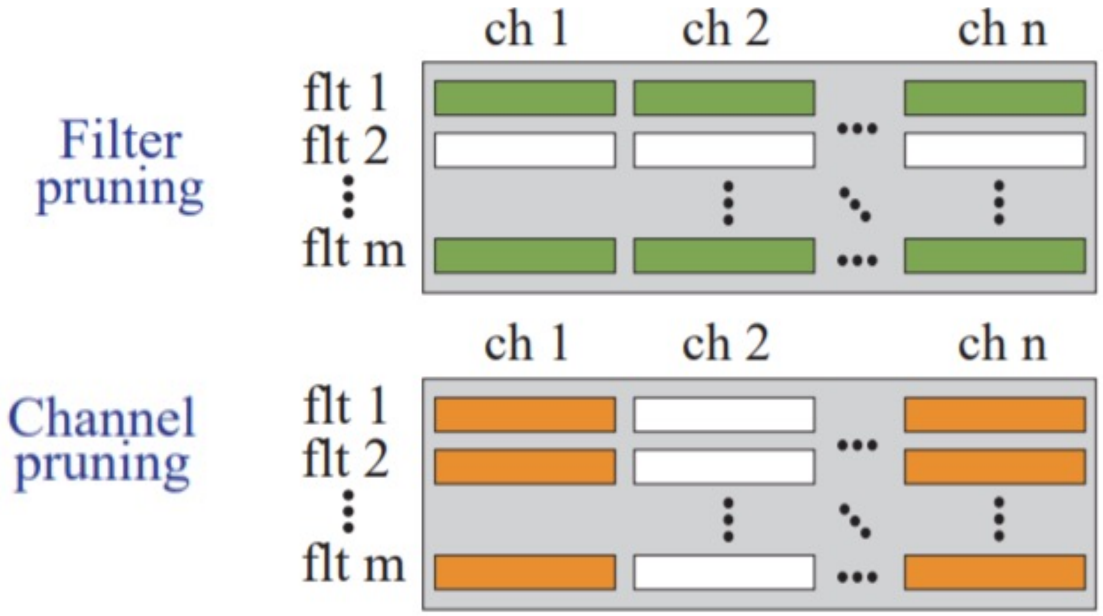
UNIVERSITY of HOUSTON
CULLEN COLLEGE of ENGINEERING

Background: Pruning

- **Structured** weight pruning
  - Hardware friendly
  - **Large Accuracy loss** due to coarse granularity

$$I * W = O$$

$$I \in R^{\{a\times b\times n\}}, W \in R^{\{k\times k\times n\times m\}}, O \in R^{\{c\times d\times m\}}$$



Skip the computing and storage of P feature maps in the $l+1$ layer

Reduce the computing and loading of P feature maps from the l layer

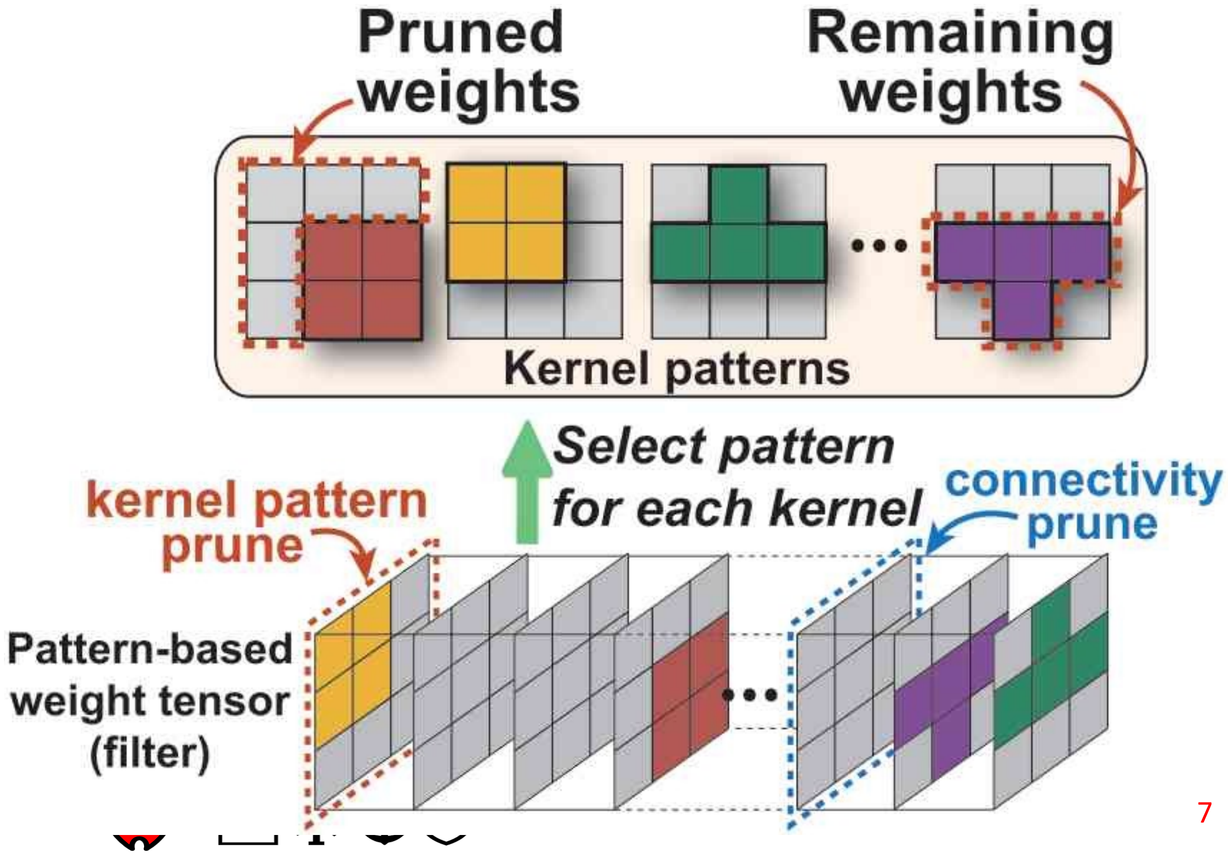
Ch: channel
Flt: filter

Background: Pruning

mAP: mean average precision

- **Pattern-based** weight pruning[1]

It is inspired by Sparse Convolution Pattern (using Gaussian and Laplace filters).



Pro: Accurate and efficient

Cons: Only applicable to 3x3 conv layers

YOLOv4:

21% mAP loss with 5x compression, not acceptable

[1]PCONV: Ma et al. "The Missing but Desirable Sparsity in DNN Weight Pruning for Real-Time Execution on Mobile Devices", AAAI,2020

Contribution

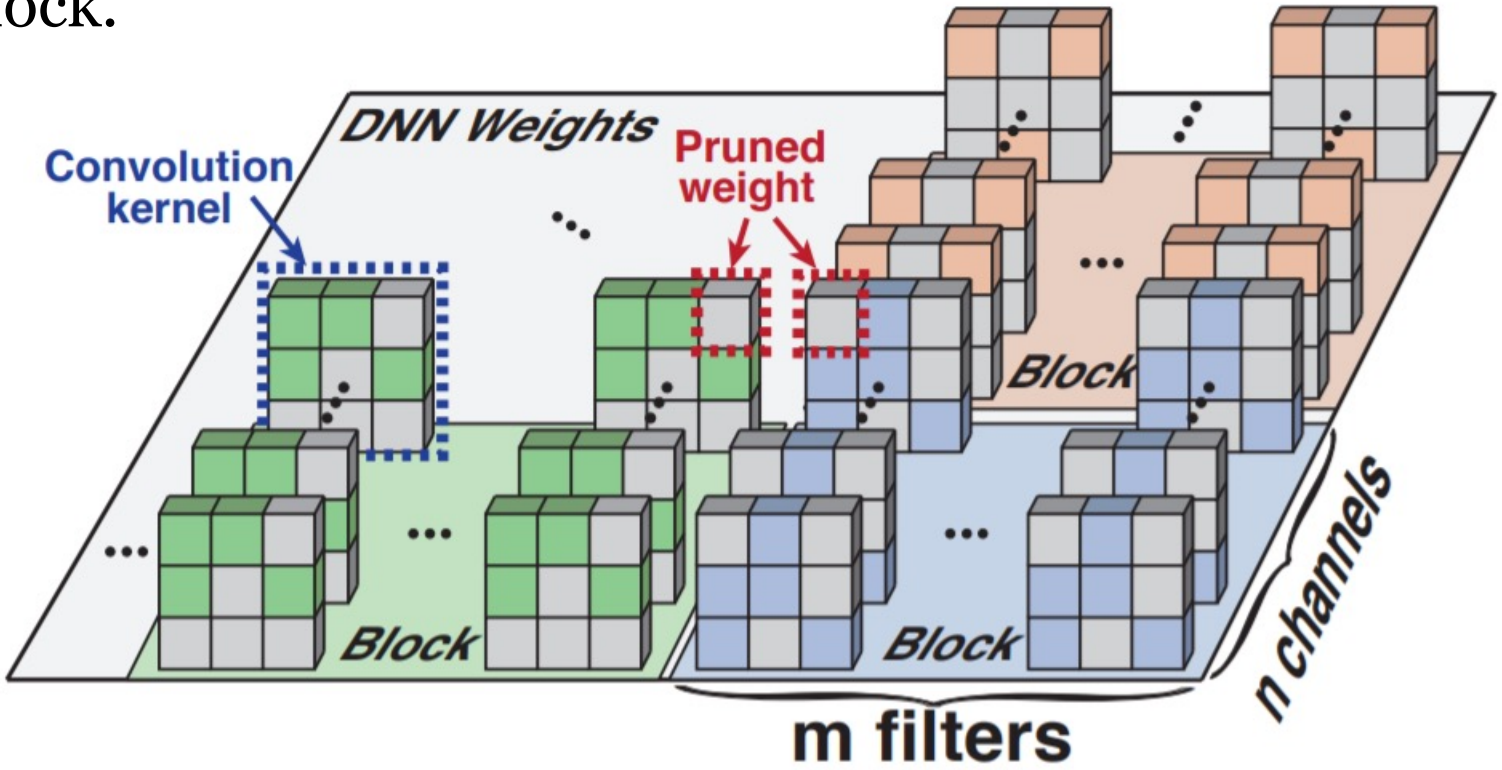
- Propose "Block-Punched Pruning" that can be applied on different types of layers
- Propose an efficient computation method for further accelerating the DNN inference speed of object detection tasks.

Outline

- Motivation
- Framework Overview
  - Block-punched Pruning
  - Compiler-assisted Acceleration
  - Mobile GPU-CPU Collaborative Scheme
- Experimental Results

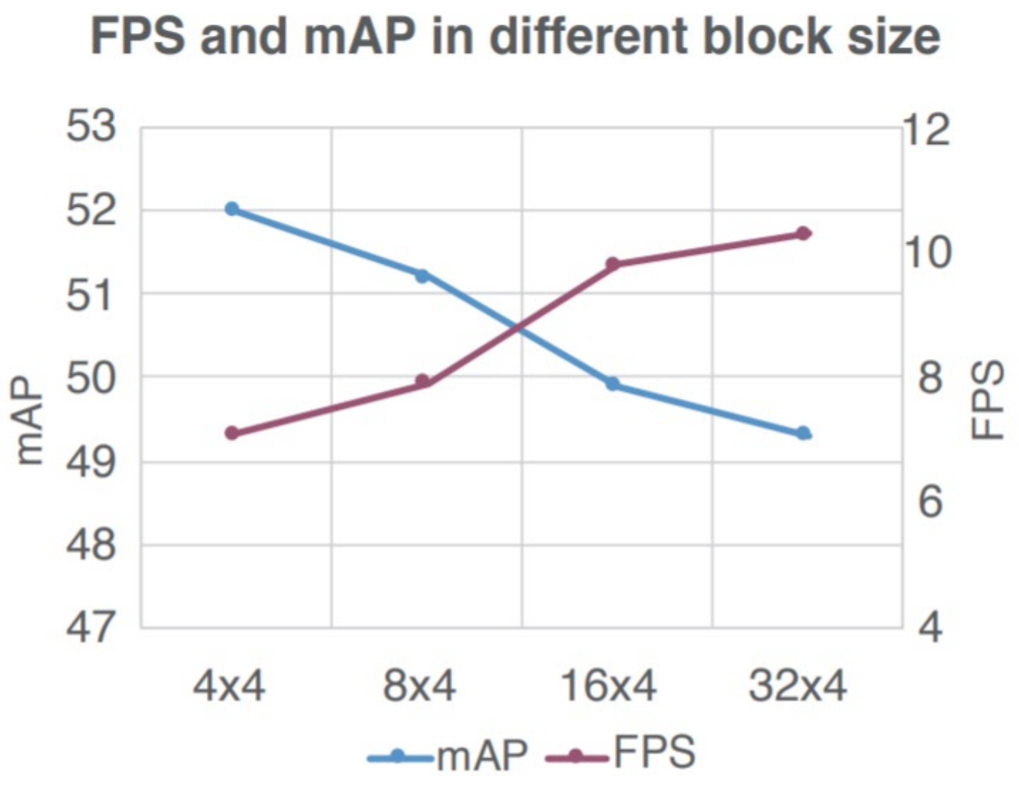
Block-Punched Pruning

- Divided the 4d weight matrix into blocks
- The weights to be pruned will punch through the same location of all kernels within a block.



Block-Punched Pruning

- Block size?
  - #channels: equals to the length of the vector registers of each treading in the mobile CPU/GPU
  - #filters: the trade-off between accuracy and hardware acceleration (multithreading)
- Larger block size, better leverage the hardware parallelism, more accuracy loss



ANTS LAB
Ai

UNIVERSITY of HOUSTON
CULLEN COLLEGE of ENGINEERING

Reweighted Regularization Pruning

- How to prune?

Controls the trade-off between accuracy and sparsity

Regularization term

Pruning problem: $\underset{\boldsymbol{W},\boldsymbol{b}}{\text{minimize}} \quad f(\boldsymbol{W};\boldsymbol{b}) + \boxed{\lambda} \sum_{i=1}^{N} \boxed{R(\boldsymbol{\alpha}_i^{(t)}, \boldsymbol{W}_i),}$

- Reweighted group lasso[1] regularization $\boldsymbol{W}_i = [\boldsymbol{W}_{i1}, \boldsymbol{W}_{i2}, ..., \boldsymbol{W}_{iK}]$, $\boldsymbol{W}_{ij} \in \mathbb{R}^{g_i m \times g_i n}$.

$$R(\boldsymbol{\alpha}_i^{(t)}, \boldsymbol{W}_i) = \sum_{j=1}^{K} \sum_{h=1}^{g_m^i} \sum_{w=1}^{g_n^i} \|\alpha_{ijn}^{(t)} \circ [\boldsymbol{W}_{ij}]_{h,w}\|_F^2,$$

Penalize the weight with small values

$$\alpha_{ijn}^{(t)} = \frac{1}{\|[\boldsymbol{W}_{ij}]_{h,w}^{t-1}\|_F^2 + \epsilon}.$$

- Use pre-trained model and retrain for 3-4 iterations

[1]Candès et al. “Enhancing Sparsity by Reweighted l1 Minimization” Journal of Fourier Analysis & Applications, 2008

Compiler-assisted Acceleration

- Block Reorder
  - Group the filters with the same structure

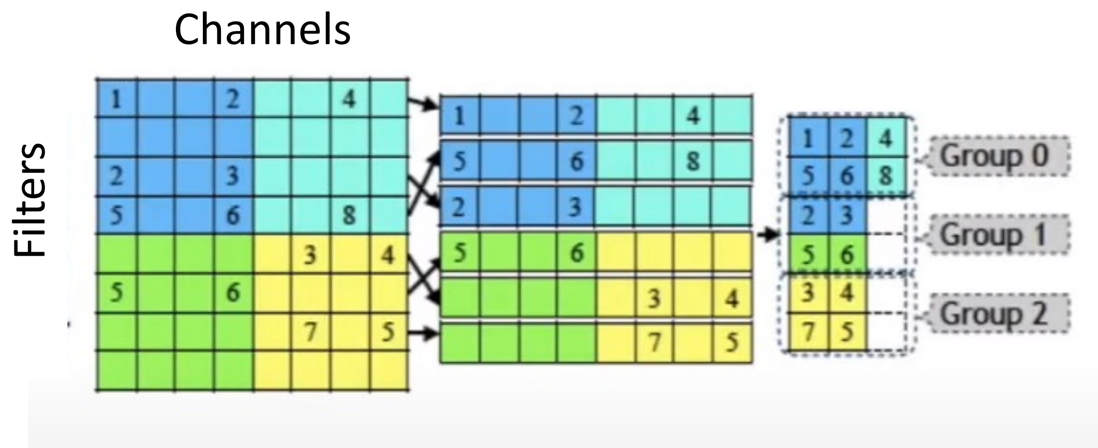


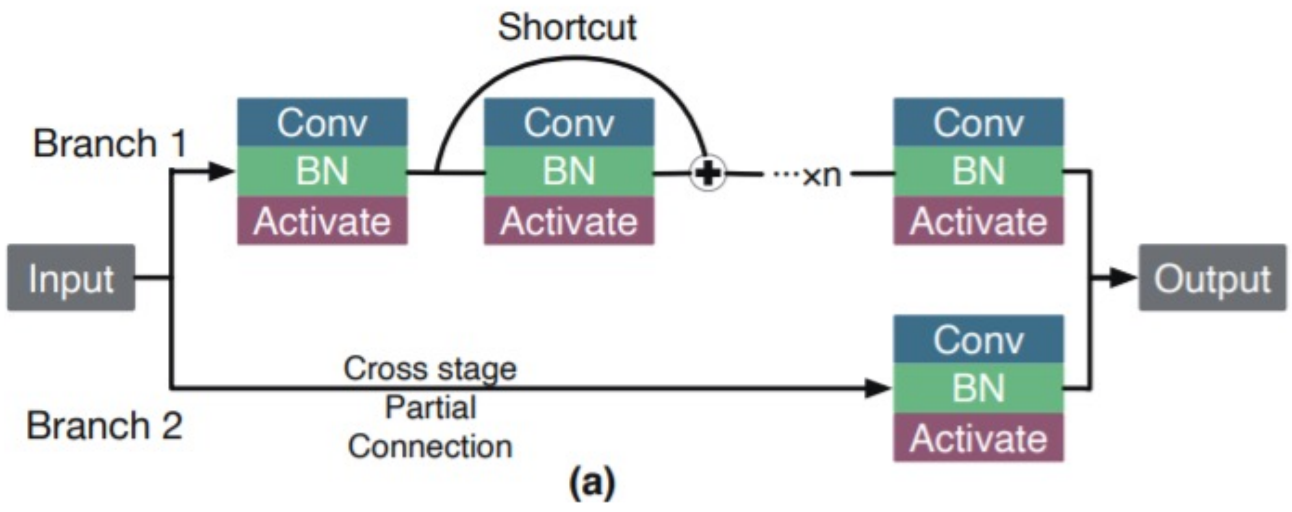
Figure. Block matrix reorder

- It can be efficiently applied on the commercial mobile GPU/CPU, no special hardware requirement.

ANTS LAB

UNIVERSITY of HOUSTON
CULLEN COLLEGE of ENGINEERING

Mobile CPU-GPU Collaborative Scheme



Yolov4

- Off-line device selection based on latency

$$\begin{cases} T_{par} = max\{t_{g1}\ ,\ t_{c2} + \tau\} \\ T_{ser} = t_{g1} + t_{g2} \end{cases}$$

select the optimal executing device for Branch 2

Outline

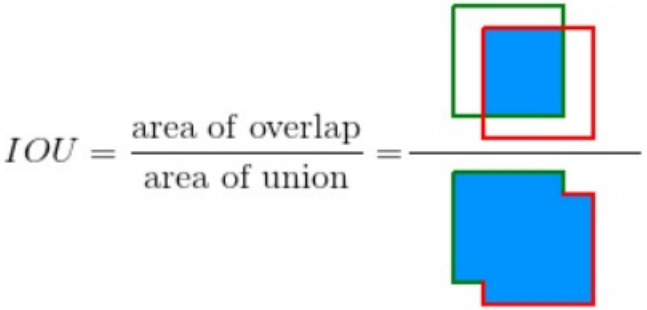
- Motivation
- Framework Overview
  - Block-punched Pruning
  - Compiler-assisted Acceleration
  - Mobile GPU-CPU Collaborative Scheme
- Experimental Results

ANTS LAB

UNIVERSITY of HOUSTON
CULLEN COLLEGE of ENGINEERING

Experiment Setup

- Nvidia Docker+Pytorch
- Training cost: 4xRTX2080Ti, 5 days
- Test device: Samsung Galaxy S20
- Manually designed layerwise prune ratio



- MS COCO dataset: 320x320 input image
- mAP: mean Average Precision under Intersection Over Union (IoU) 0.5 for multi-labels.
- AP[0.5:0.95]: Average precision under IoU 0.5 to 0.95

Performance evaluation

Approach	Input Size	backbone	#Weights	#FLOPs	mAP	AP@[.5:.95]	FPS
CenterNet-DLA (Duan et al.)	512	DLA34	16.9M	52.58G	57.1	39.2	1.9
CornerNet-Squeeze (Law et al.)	511	-	31.77M	150.15G	-	34.4	0.3
SSD (Liu et al.)	300	VGG16	26.29M	62.8G	43.1	25.1	4.2
MobileNetv1-SSDLite (Sandler et al.)	300	MobileNetv1	4.31M	2.30G	-	22.2	49
MobileNetv2-SSDLite (Sandler et al.)	300	MobileNetv2	3.38M	1.36G	-	22.1	41
Tiny-DSOD (Li et al.)	300	-	1.15M	1.12G	40.4	23.2	-
YOLOv4 (Bochkovskiy, Wang, and Liao)	320	CSPDarknet53	64.36M	35.5G	57.3	38.2	3.5
YOLO-Lite (Huang, Pedoeem, and Chen)	224	-	0.6M	1.0G	-	12.26	36
YOLOv3-tiny (Redmon and Farhadi)	320	Tiny Darknet	8.85M	3.3G	29	14	14
YOLOv4-tiny (Bochkovskiy, Wang, and Liao)	320	Tiny Darknet	6.06M	4.11G	40.2	-	11
YOLObile (GPU only)	320	CSPDarknet53	4.59M	3.95G	**49**	**31.6**	**17**
YOLObile (GPU&CPU)	320	CSPDarknet53	4.59M	3.95G	**49**	**31.6**	**19.1**

Table 2: Accuracy (mAP) and speed (FPS) comparison with other object detection approaches.

Performance evaluation

- All the results are evaluated under our compiler optimization

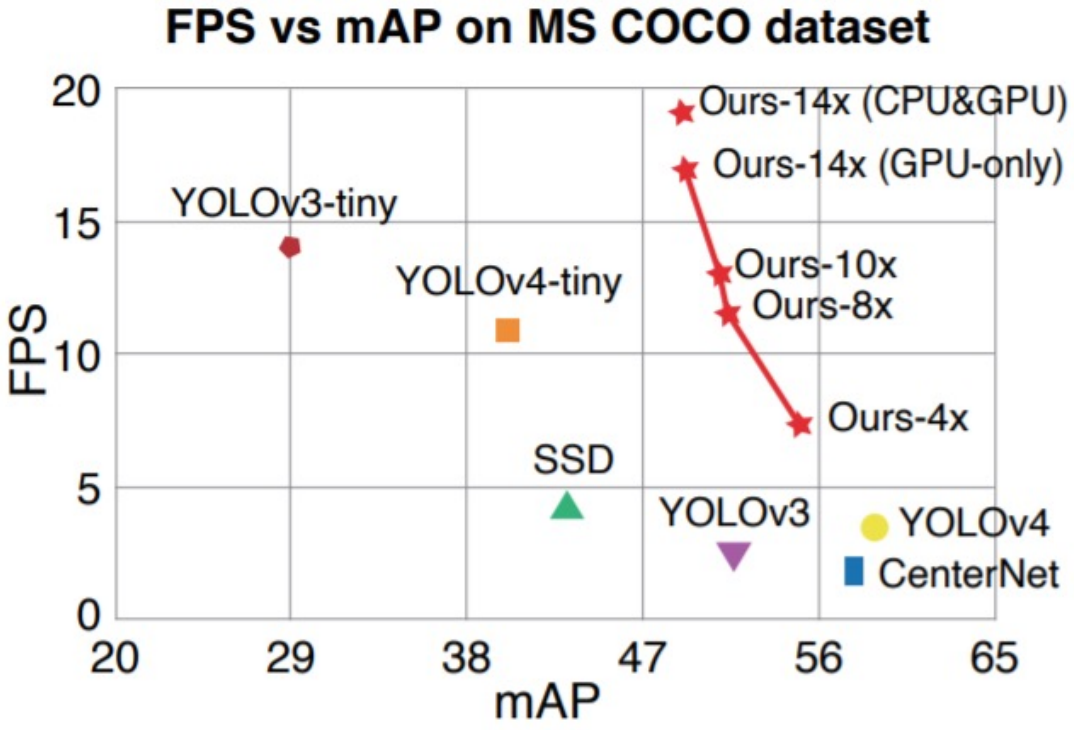


Figure 6: The accuracy (mAP) and speed (FPS) comparison of YOLObile under different compression rate and different approaches.

Performance evaluation

#Weights	#Weights Comp. Rate	#FLOPs	mAP	AP@[.5:.95]	FPS
64.36M	1×	35.8G	57.3	38.2	3.5
16.11M	3.99×	10.48G	55.1	36.5	7.3
8.04M	8.09×	6.33G	51.4	33.3	11.5
6.37M	10.1×	5.48G	50.9	32.8	13
4.59M	14.02×	3.95G	49	31.9	17

Table 1: Accuracy and speed under different compression rates.

Ablation Study

Pruning Scheme	#Weights	#Weights Comp. Rate	mAP	FPS
Not Prune	64.36M	1×	57.3	3.5
Unstructured	8.04M	8.09×	53.9	6.4
Structured	8.04M	8.09×	38.6	12
Ours	8.04M	8.09×	51.4	11.5

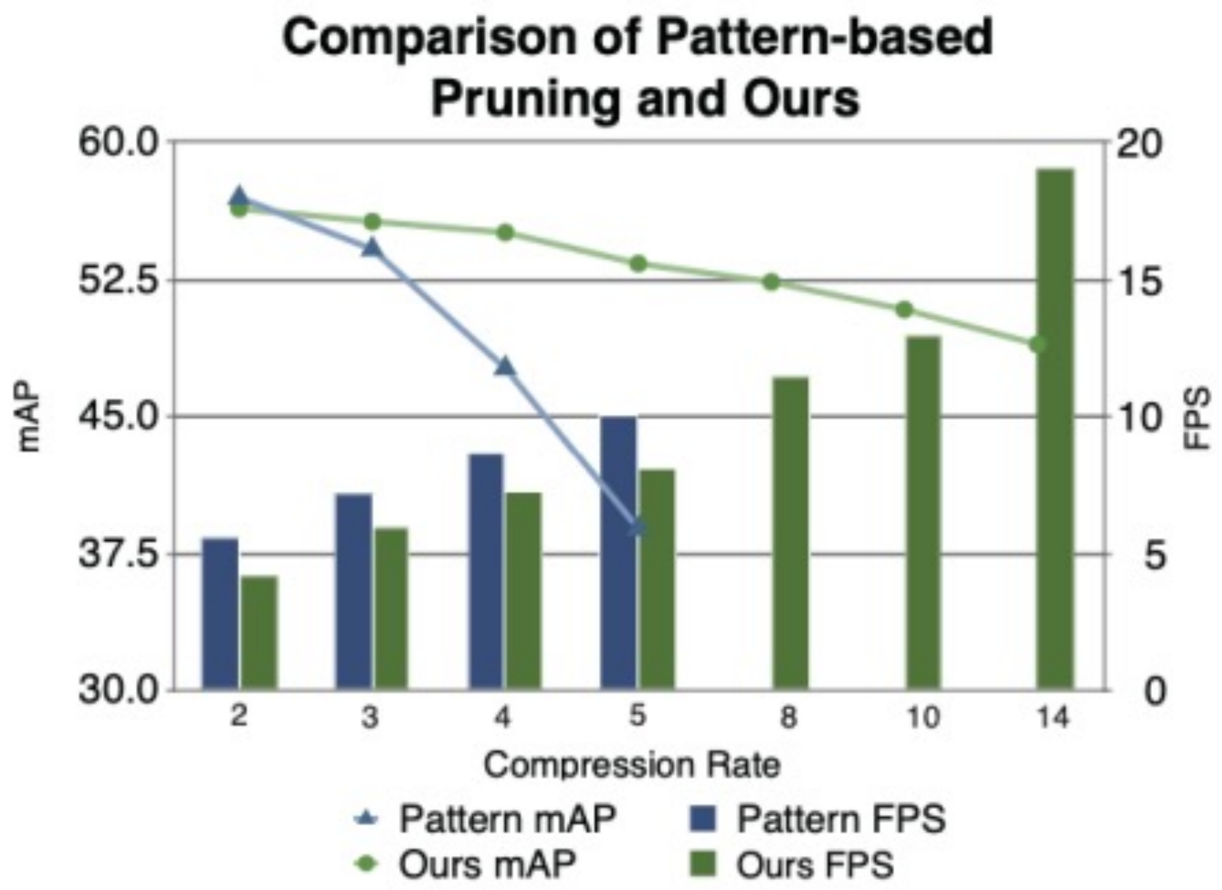
Table 3: Comparison of different pruning schemes.

Compared with structured pruning and unstructured pruning, our block-punched pruning scheme achieves both high accuracy and fast inference speed.

ANTS LAB

UNIVERSITY of HOUSTON
CULLEN COLLEGE of ENGINEERING

Ablation Study



- Block-punched method achieves a small accuracy drop when the compression rate is high (>5).
- When the compression rate is 5, the Pattern-based method results in a sharp drop down of the curve.

Conclusion

- YOLObile, a real-time object detection framework on mobile devices via compression-compilation co-design
  - Block-Punched Pruning
  - Reweighted Regularization Pruning Algorithm
  - Compiler-assisted Acceleration
  - Mobile GPU-CPU Collaborative scheme

THANK YOU

UNIVERSITY of HOUSTON | ENGINEERING