WINS: Winograd Structured Pruning for Fast Winograd Convolution

Cheonjun Park[1], Hyunjae Oh[2], Mincheol Park[3], Hyunchan Moon[4], Minsik Kim[2],
Suhyun Kim[5], Myung Kuk Yoon[6], Won Woo Ro[2]

[1]Hankuk University of Foreign Studies, [2]Yonsei University, [3]SAIT (Samsung),
[4]LG Electronics, [5]Kyung Hee University, [6]Ewha Womans University

1. Background : Winograd Convolution

- **Efficient Winograd convolution on GPU**
  - **Winograd Conv. Problem 1** : Low computational intensity
  - **Winograd Conv. Problem 2** : Frequent memory access overhead of EWMM
  - GPU convert and process Winograd convolution's EWMM and channel-wise summation to BGEMM operations.
  - BGEMM maintains a structured form to maximize data reuse on the GPU.
  - EC2B (conversion from EWMM and Channel-wise summation to BGEMM).

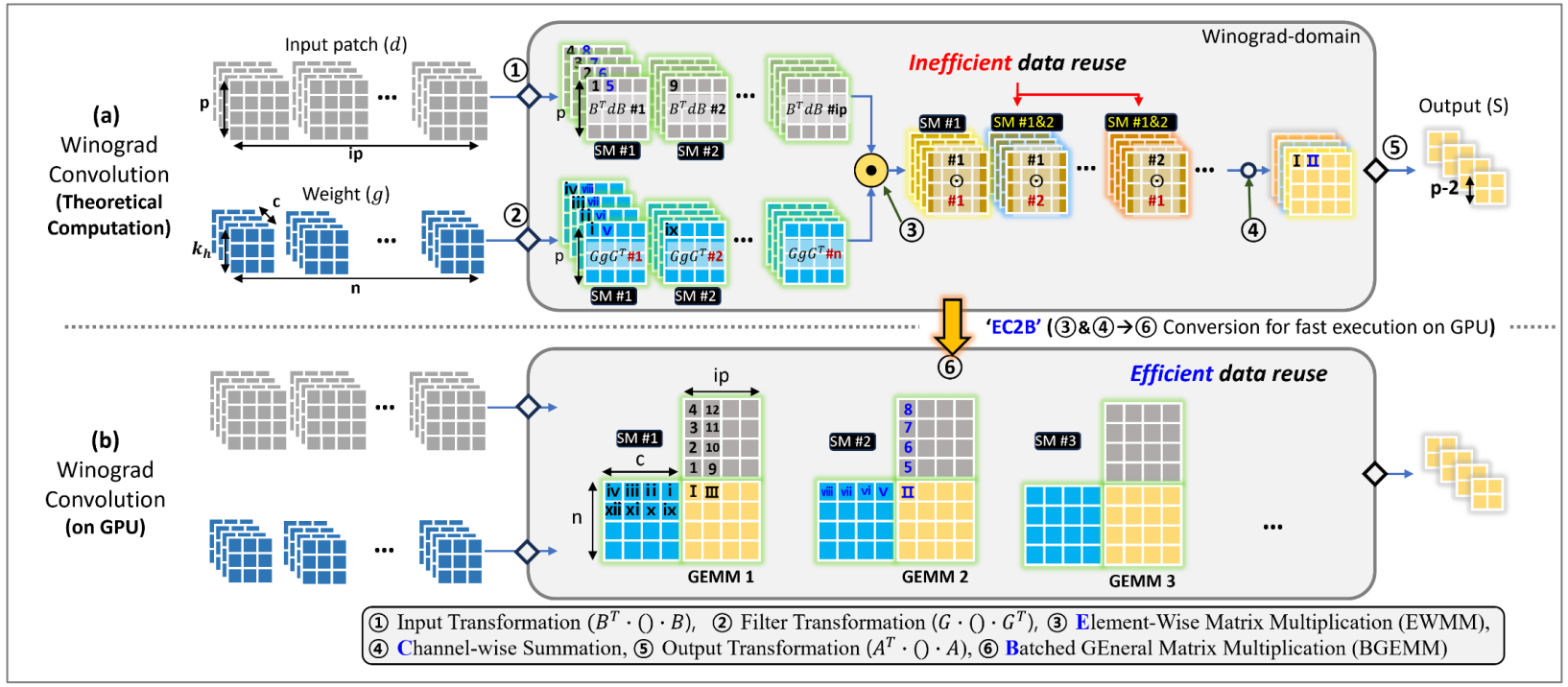

Fig.1 Overview of Winograd convolution: Conversion from (a) theoretical computation to (b) GPU-accelerated execution.

2. Related Work

- **Filter Pruning (FP)**
  - Filter pruning removes rows (filter pruning) in converted weight matrix using im2col [3] for multiple 3D filter tensors.
  - Spatial-Winograd pruning is the appliance of filter pruning on Winograd convolution, the pruning unit size is a 3D filter size.
- **Winograd Weight Pruning (WWP)**
  - WWP removes redundant values of Winograd-transformed matrices in a unit of an element to reduce the matrix size of Winograd convolution.
  - Sparse Winograd CNN introduces the first attempt at pruning and re-training method in the Winograd domain.

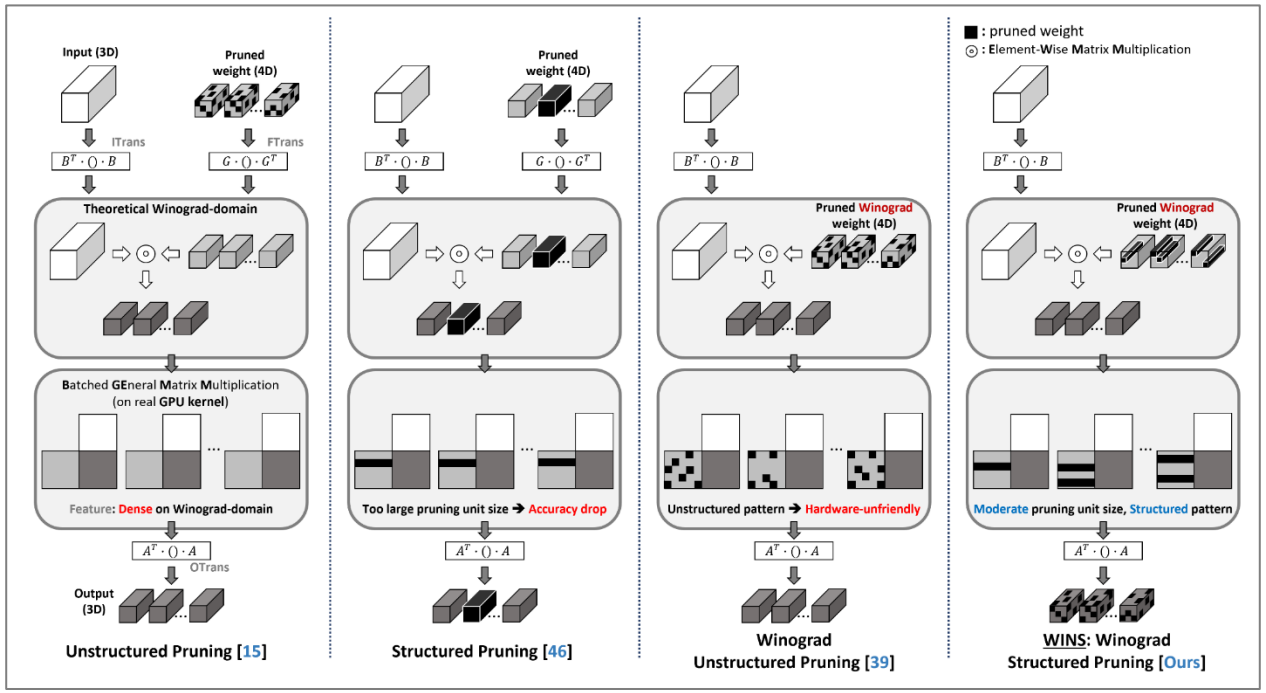

Fig.2 Overview of previous pruning methods and the proposed WINS for Winograd convolution.

3. Proposed Method: WINS

- **Winograd Structured Pruning (WINS)**
  - (a) Motivation 1: Large pruning unit size problem of FP
  - (b) Motivation 2: Irregular data problem of WWP
  - WINS removes weight parameters vector by vector from each sub-GEMM weight matrix.
  - WINS's pruning unit size ($R^{1\times c}$) of the WINS is 16 (p^2) times smaller than that ($p^2 \times R^{1\times c}$) of the filter pruning.

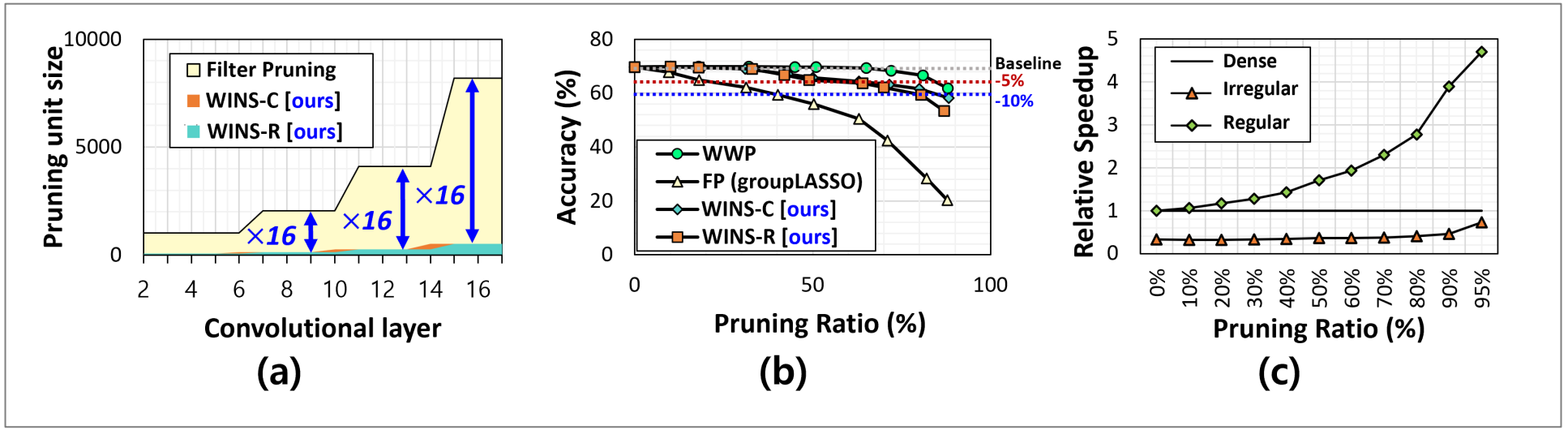

Fig.3 (a) Comparison of pruning unit size on ResNet-18. (b) Comparison of accuracy with respect to pruning ratio for four pruning methods of ResNet-18 on ImageNet. The models were fine-tuned for one epoch. (c) Comparison of GEMM speedups by pruning between irregular data (cuSPARSE) and regular data (cuBLAS). The speedups are measured on the 6th convolution layer of VGG-16.

4. Enhanced Method 1: WINS-B (Balanced)

- **Motivation:** **Imbalance overhead problem of WINS.**
  - WINS exhibits an imbalanced character istic attributed to the different PR in each of the 16 (p2) sub-GEMMs.
  - This imbalance leads to an under-throughput on GPUs, as each sub-GEMM is assigned to an SM for independent execution.
- **Method: WINS-B ensures that all sub-GEMMs have equal PR and employ a specific order for pruning to achieve this equal PR across all sub-GEMMs.**

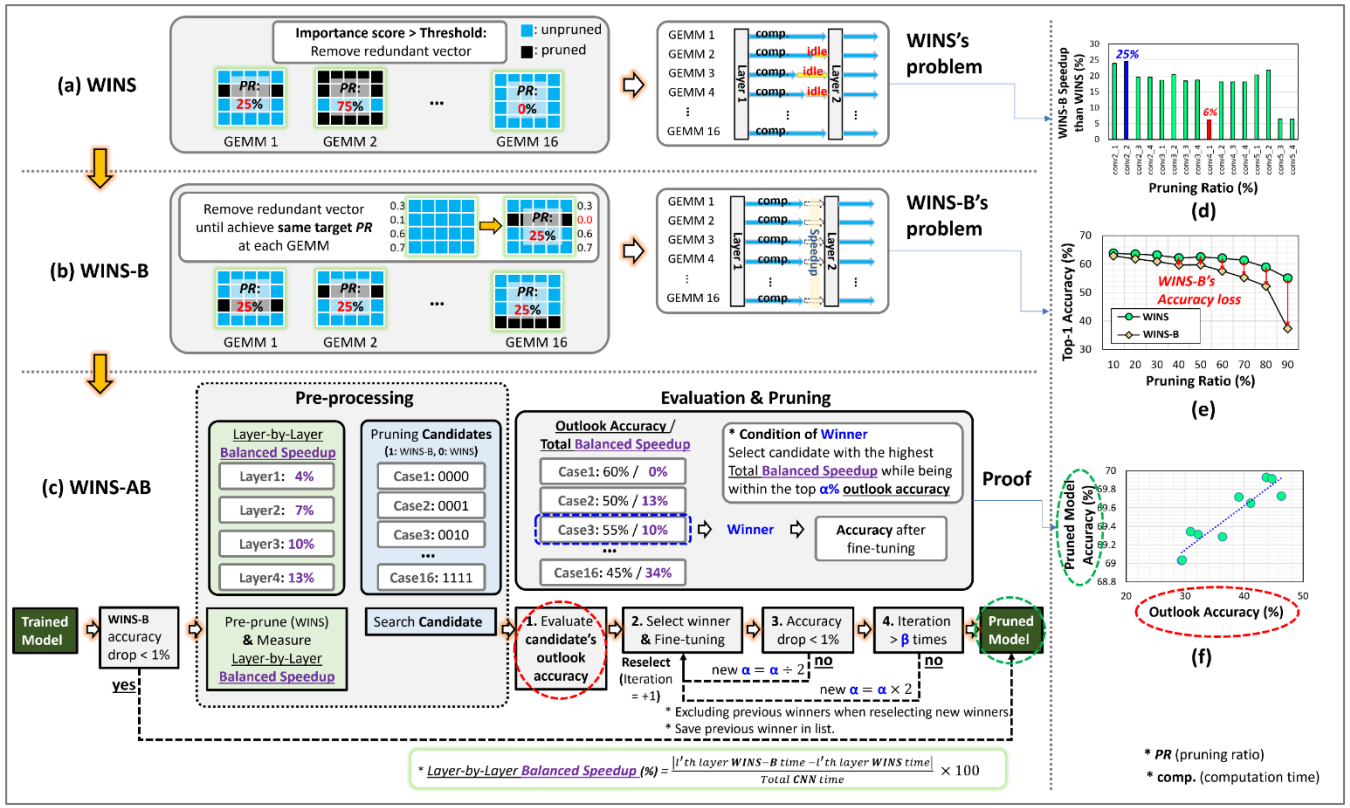

Fig.4 Overview of the three proposed methods: (a) WINS, (b) WINS-B, and (c) WINS-AB. ResNet-18 on ImageNet. (d) Comparison of performance improvements in layers obtained from applying balancing on WINS. (e) Comparison of Top-1 accuracy between WINS and WINS-B. The fine-tuning epoch is one. (f) Correlation of outlook accuracy (100 iterations) and fine-tuning accuracy (40 epochs).

5. Enhanced Method 2: WINS-AB (Adaptive)

- **Motivation: Accuracy drop of WINS-B.**
  - The accuracy of WINS-B exhibits a disadvantage compared to relying on WINS alone.
- **Method: WINS-AB introduce an adaptive framework that applies WINS-B or WINS selectively, simultaneously considering evaluation accuracy and speedup due to workload balance.**
  - [1] Pre-processing -> [2] Evaluation & Pruning (Outlook Accuracy)

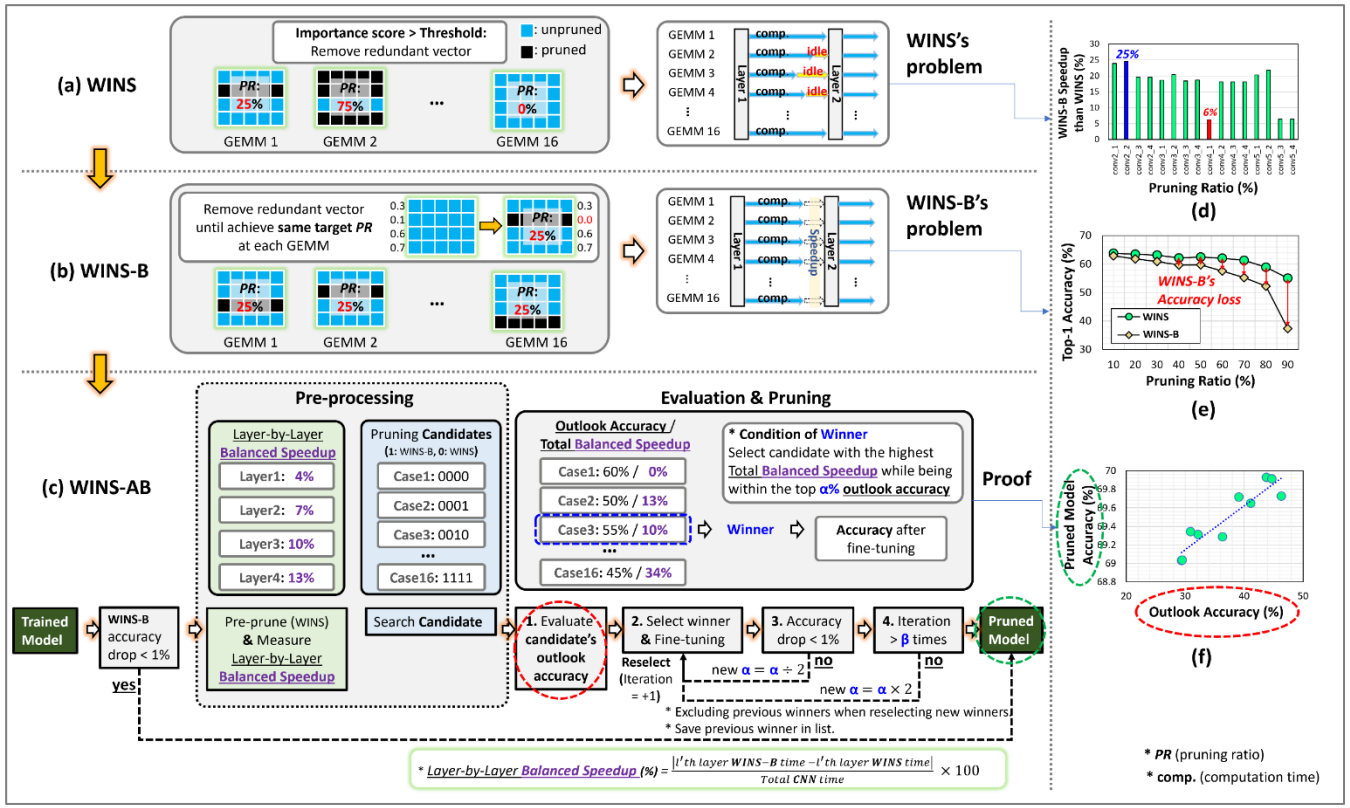

Fig.4 Overview of the three proposed methods: (a) WINS, (b) WINS-B, and (c) WINS-AB. ResNet-18 on ImageNet. (d) Comparison of performance improvements in layers obtained from applying balancing on WINS. (e) Comparison of Top-1 accuracy between WINS and WINS-B. The fine-tuning epoch is one. (f) Correlation of outlook accuracy (100 iterations) and fine-tuning accuracy (40 epochs).

6. Experiments

- **In ResNet-18 and VGG-16, within 1.5% accuracy drop, WINS-AB is 1.84× and 1.27× faster than FP.**
- **WINS-AB maintains a structural form, so WINS-AB is 2.01× faster than Winograd convolution when PR is 84%.**
- **On ResNet-50, WINS-AB achieves 1.82% higher accuracy and has 1.47 times faster inference time, even with up to15.5% more PR than Torque.**

Table.1 Comparison with FP on ImageNet. The GEMM operation with the unpruned weight parameters of WINS-AB and input values is accelerated using customized CUTLASS API. This symbol (⋆) means `prune using BCBP at 1×1 convolution layer'. This symbol (‡) means `no prune at 1×1convolution layer'. GAP is the Top-1 accuracy difference between baseline and pruned model.

Method	Pruning Ratio (%)	Pruned FLOPs (%)	Top-1 Accuracy (%)			Inference time	Speed up
			Baseline	Pruned	GAP		
ResNet-18 (**baseline**)	-	-	69.70	-	-	3.16ms	1.00×
UDSP [9]	20.0	-	69.76	69.48	-0.28	2.51ms	1.26×
SOSP [46]	39.1	-	69.80	69.60	-0.20	1.88ms	1.68×
WINS-AB [**ours**]	40.1	-	69.70	69.90	**0.20**	1.87ms	**1.69**×
FPGM [20]	41.8	-	70.20	68.40	-1.80	1.85ms	1.71×
PFP [35]	43.8	-	69.70	67.30	-2.40	1.83ms	1.73×
WHC [2]	-	41.8	69.80	68.50	-1.30	-	-
RLAL [8]	-	50.0	69.80	69.00	-0.80	-	-
WINS-AB [**ours**]	84.0	75.1	69.70	69.00	**-0.70**	1.12ms	**2.82**×
ResNet-50 (**baseline**)	-	-	76.10	-	-	12.53ms	1.00×
UDSP [9]	25.0	-	76.13	76.51	0.38	9.74ms	1.29×
ATO [59]	-	61.7	76.13	76.07	-0.06	-	-
APIB [13]	58.0	62.0	76.15	75.37	-0.78	6.06ms	2.07×
Torque [14]	64.5	-	76.07	74.58	-1.49	5.34ms	2.35×
WINS-AB* [**ours**]	80.0	69.5	76.10	76.43	**0.33**	3.61ms	**3.47**×
SqueezeNet-1.0 (**baseline**)	-	-	58.20	-	-	1.81ms	1.00×
ℓ_2-norm [32]	9.7	-	58.20	55.30	-2.90	1.61ms	1.12×
WINS-AB [**ours**]	28.1	-	58.20	57.00	-1.20	1.48ms	1.23×
DenseNet-121 (**baseline**)	-	-	74.60	-	-	9.21ms	1.00×
C-SGD [16]	29.9	-	74.50	73.70	-0.80	8.10ms	1.14×
LWP [67]	35.2	-	74.60	72.80	-1.80	6.92ms	1.33×
WINS-AB‡ [**ours**]	85.0	-	74.60	74.20	**-0.40**	5.55ms	**1.66**×


Thank you