TDC: Towards Extremely Efficient CNNs on GPUs via Hardware-Aware Tucker Decomposition

Lizhi Xiang, Miao Yin, Chengming Zhang, Aravind Sukumaran-Rajam, P. Sadayappan, Bo Yuan, Dingwen Tao

THE UNIVERSITY OF UTAH

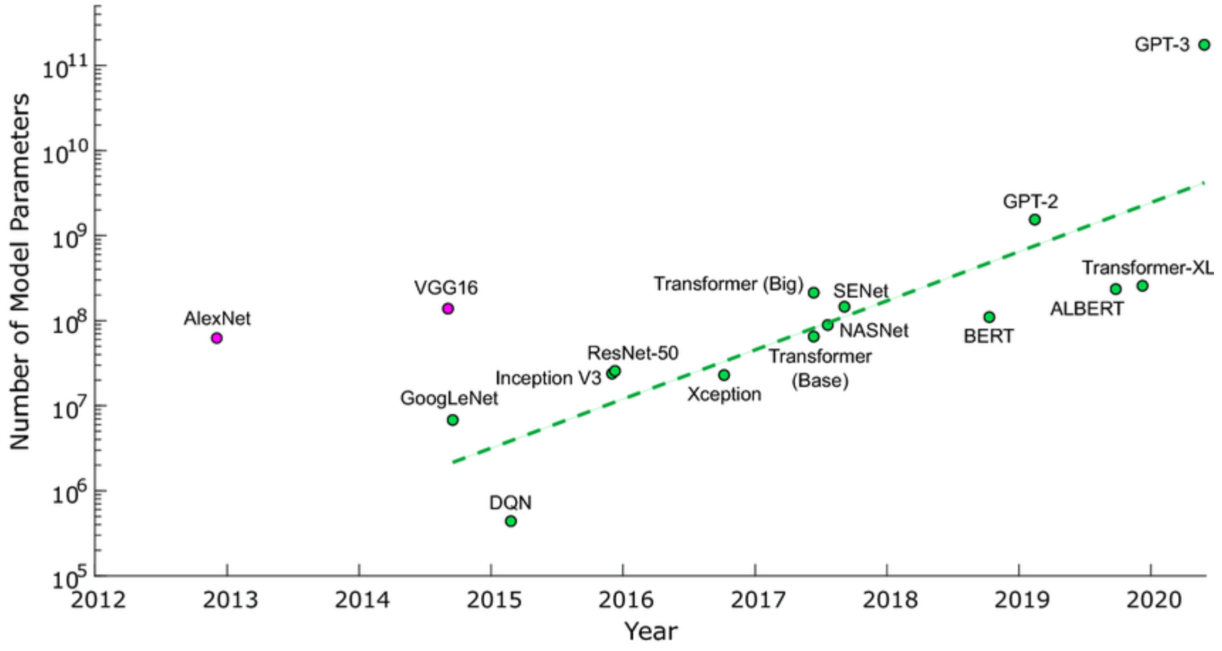
INDIANA UNIVERSITY

RUTGERS THE STATE UNIVERSITY OF NEW JERSEY

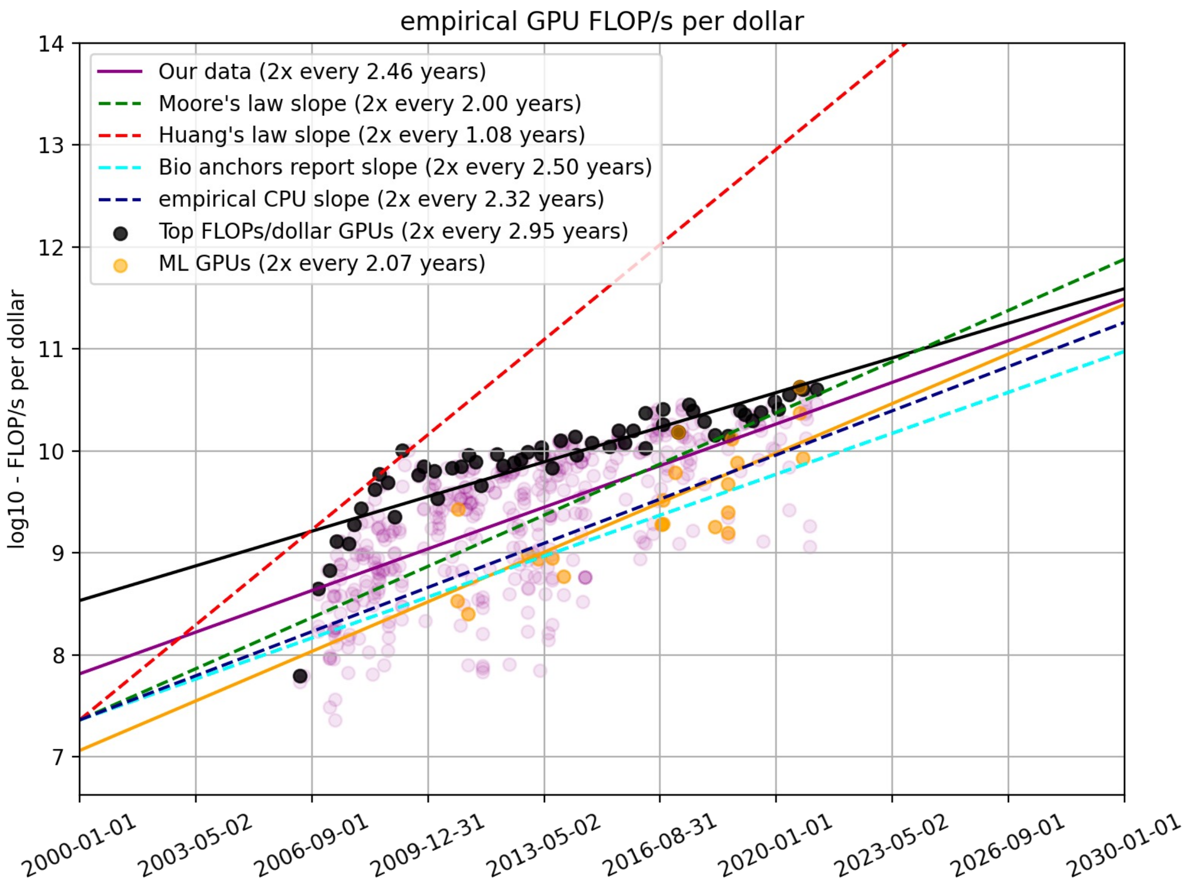
Meta

AI is Everywhere

DNN Model Trend vs GPU Development



Ref: https://www.researchgate.net/publication/349044689_Freely_scalable_and_reconfigurable_optical_hardware_for_deep_learning

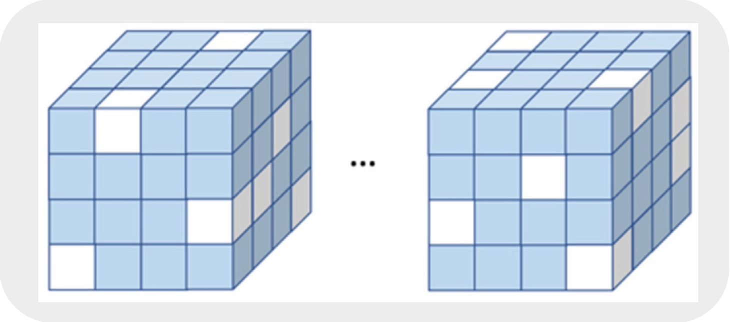


Ref: https://epochai.org/blog/trends-in-gpu-price-performance

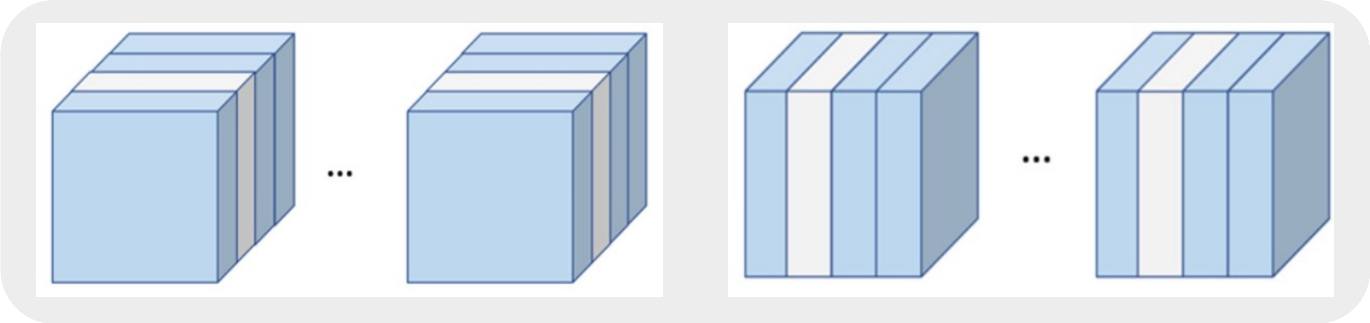
> The development of GPU is significantly behind the expanding speed of DNN model size

Compression Techniques

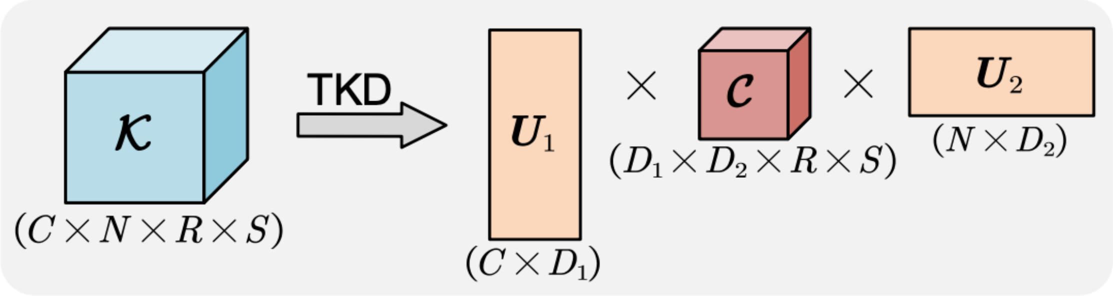
Unstructured Pruning



Structured Pruning

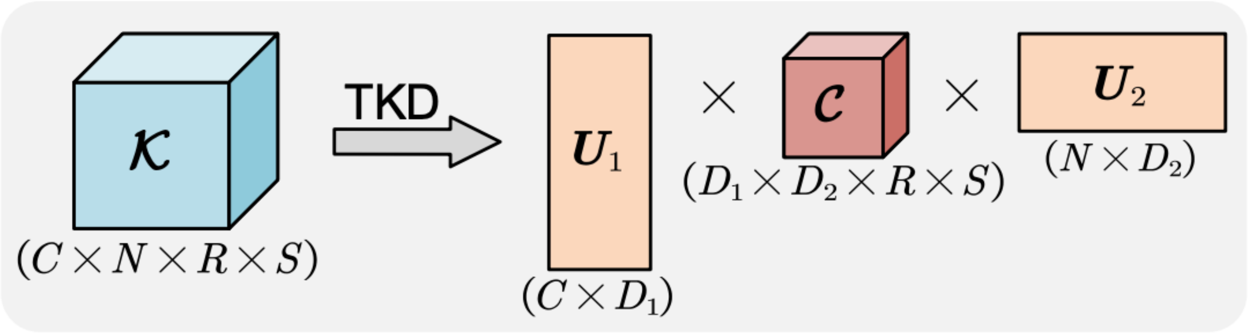


Tensor Decomposition



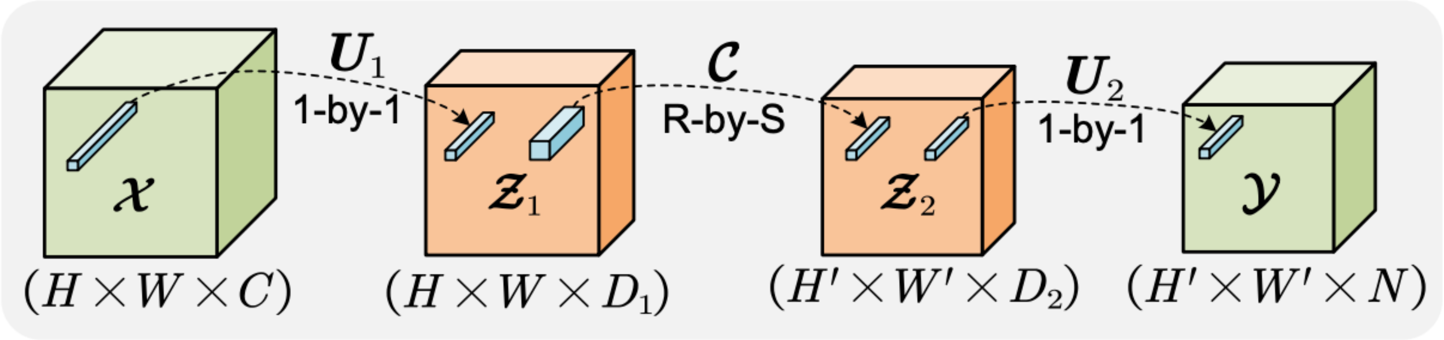
Tucker Decomposition (TKD)

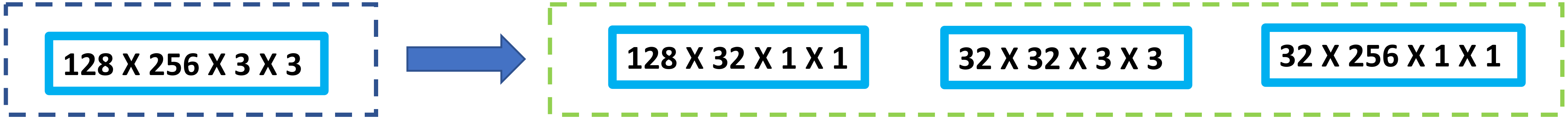
> Original kernel is decomposed into three kernels:



- Avoid complex data structure
- Able to keep the spatial information
- Adjust D1 and D2 to control the entire computational cost under a target budget

> Tucker-format convolution (The original convolution is transformed to three small convolutions):





Discrepancy in Practice

01

Hard to train TKD compressed models

02

Lack of software-aware TKD convolution algorithms for CNN acceleration

03

Lack of performance-driven frameworks for highly efficient and accurate CNN inference on GPUs

Optimized Training

> Challenges for training tucker-format models:

- Directly training Tucker-format models from scratch
  - Limited capacity -> **accuracy degradation**
- Initializing Tucker-format models from uncompressed models
  - Approximation error -> **accuracy degradation**

> Why Alternating Direction Method of Multipliers (ADMM)?

- Impose low-rankness corresponding to hardware performance
- Significantly preserve task accuracy

Accuracy comparison between directly training and our ADMM-based compression for ResNet-20 on CIFAR-10:

Method	**Top-1 (%)**	**FLOPs↓**
Baseline	91.25	N/A
Direct Compression	87.41	60%
ADMM-based	91.02	60%

Optimized Training(ADMM-based Training)

Training objective:

$$\min_{\mathcal{W}} \ \ell(\mathcal{K}), \ \text{s.t.} \ \mathrm{rank}(\mathcal{K}) \le \mathcal{P}_{\text{device}}.$$

Hardware budget

$$\Downarrow$$

$$\min_{\mathcal{W}} \ \ell(\mathcal{K}), \ \text{s.t.} \ \mathrm{rank}(\mathcal{K}) \le [D_1^*, D_2^*],$$

Selected ranks according to practical runtime of our kernel

$$\Downarrow$$

$$\min_{\mathcal{K},\widehat{\mathcal{K}}\in\mathcal{Q}} \ \max_{\mathcal{M}} \ \ell(\mathcal{K}) + \frac{\rho}{2}\|\mathcal{K} - \widehat{\mathcal{K}} + \mathcal{M}\|_F^2 - \frac{\rho}{2}\|\mathcal{M}\|_F^2,$$

where $\mathcal{Q} = \{\widehat{\mathcal{K}} | \mathrm{rank}(\widehat{\mathcal{K}}) \le [D_1^*, D_2^*]\}$

Training steps:

$$\mathcal{K} \leftarrow \mathcal{K} - \alpha\left(\frac{\partial \ell(\mathcal{K})}{\partial \mathcal{K}} + \rho(\mathcal{K} - \widehat{\mathcal{K}} + \mathcal{M})\right),$$

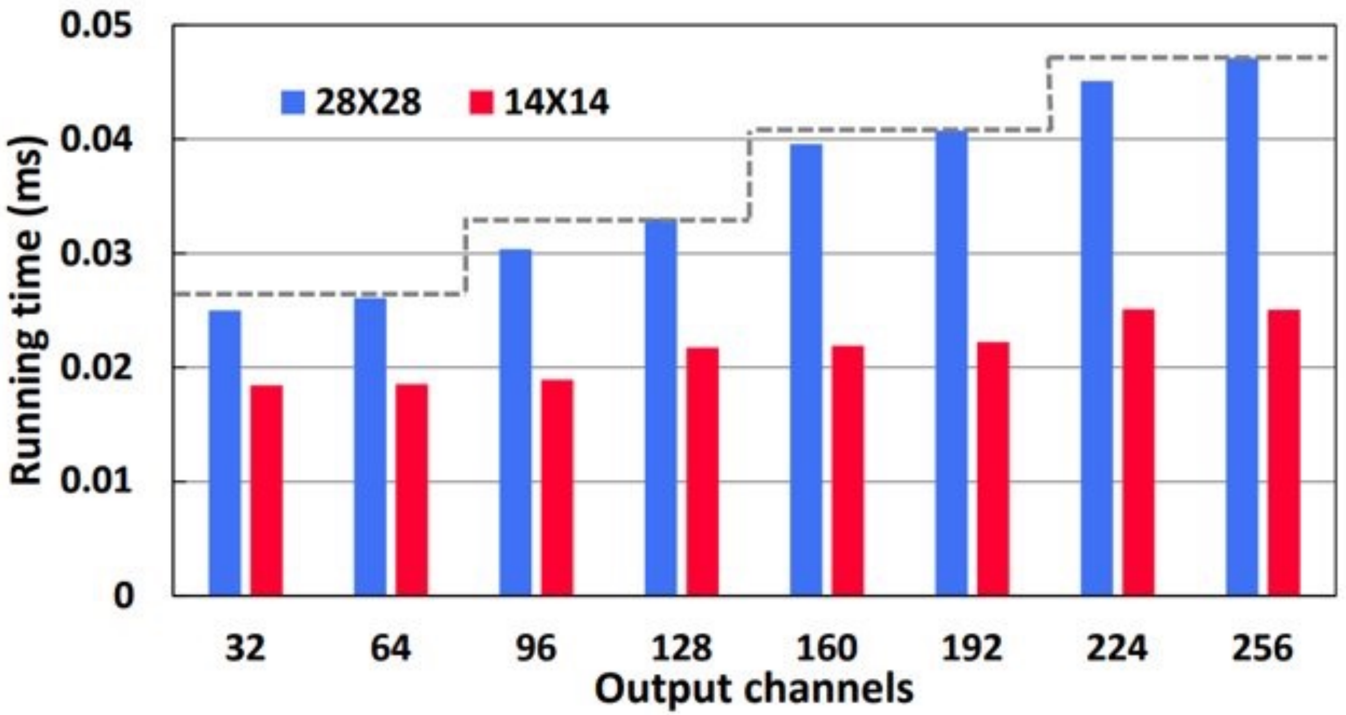
$$\widehat{\mathcal{K}} \leftarrow \mathbf{proj}(\mathcal{K} + \mathcal{M})$$

Truncated-HOSVD that truncates the smallest singular values

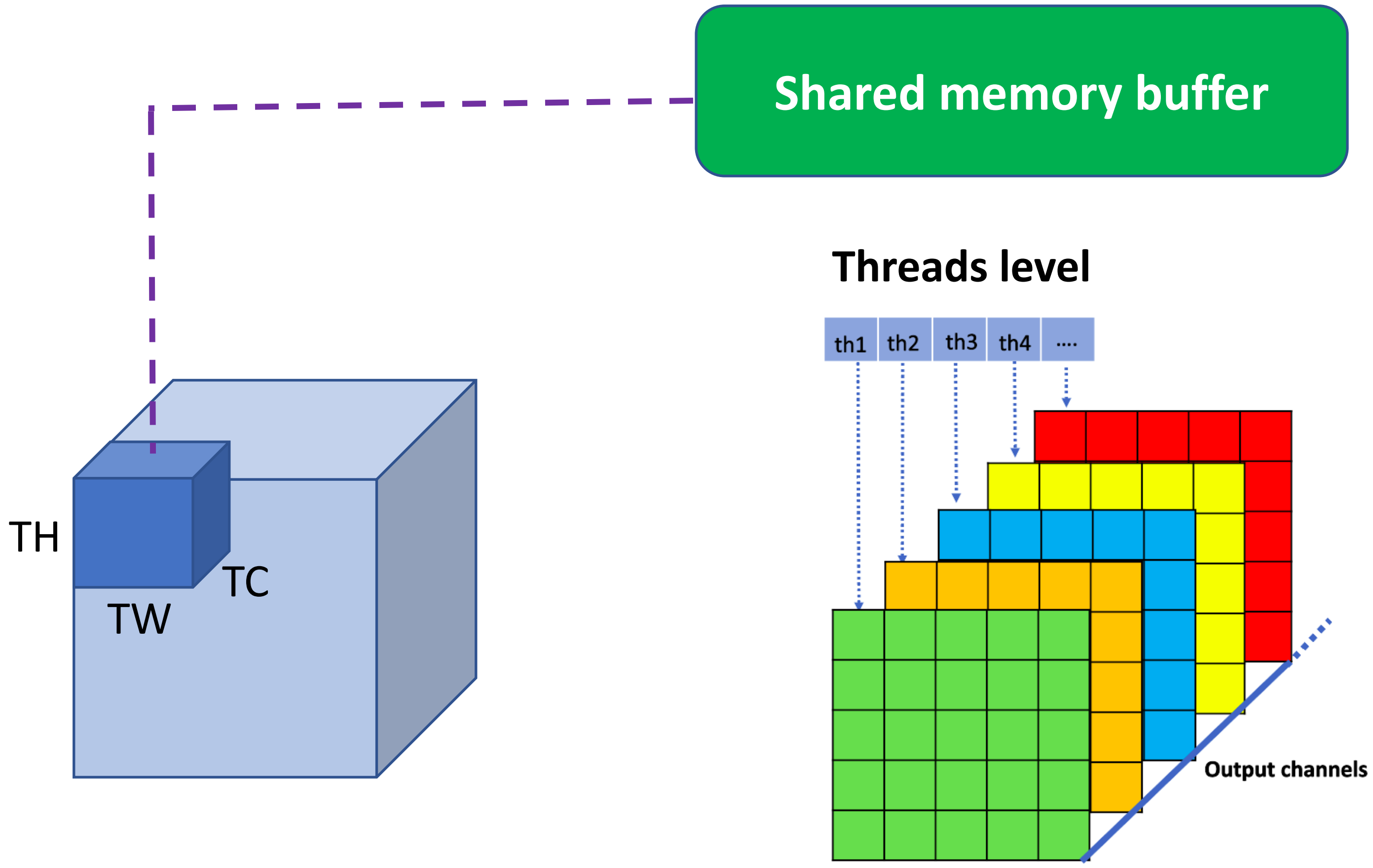
TDC: Convolution Kernel Design

Hard to translate flops reduction to actual performance improvement.

- **Irregular convolution shape.**
- **Compute resource under-utilization.**

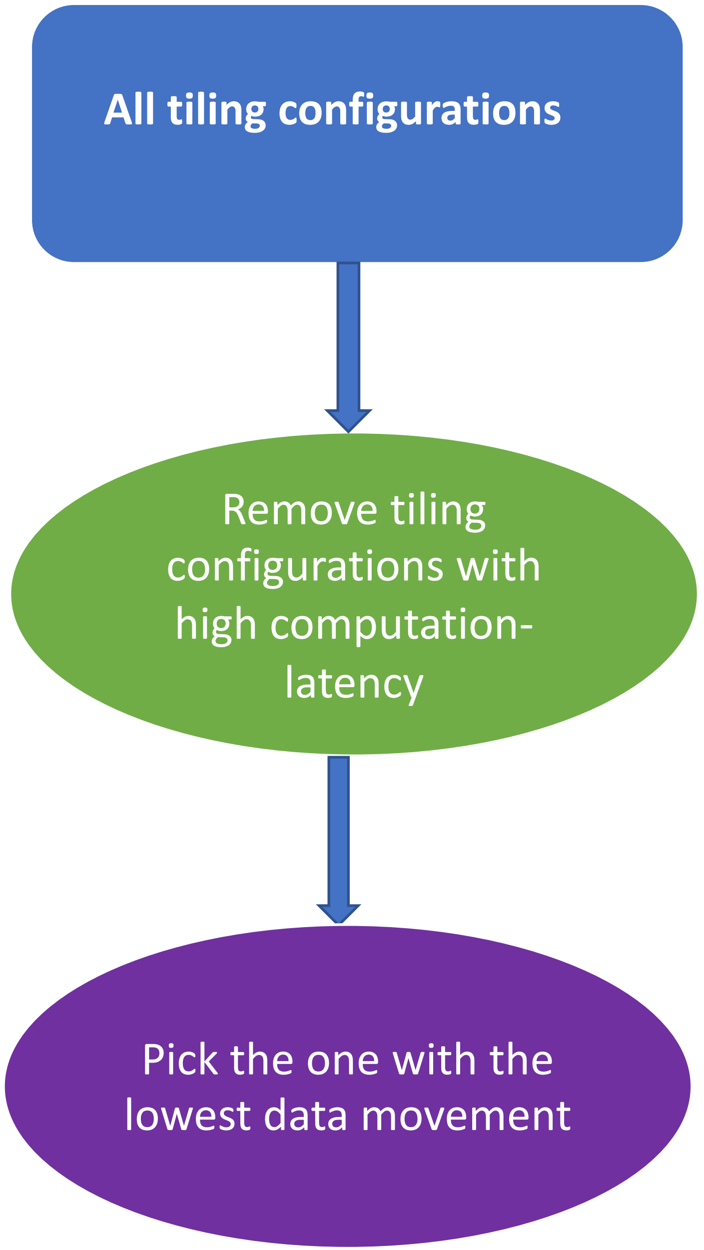


TDC: Convolution Kernel Design




```
//Input: Input tensor X, Conv kernel K
//Output: Output tensor Y
shared input_tile[TC][(TH+R-1)*(TW+S-1)]
float temp_result[TH][TW], kernel[R][S]
unsigned int tile_tc_id = blockId/(H/TH * W/TW)
unsigned int tile_id = blockId%(H/TH * W/TW)
unsigned int tile_h_id = tile_id/(W/TW)
unsigned int tile_w_id = tile_id%(W/TW)
unsigned int output_n = threadIdx.x
//copy tiled input tensor from global to shared
copy(input_tile, X)
syncthreads() //synchronize all threads in a thread block
for c = 0 to TC:
  copy(kernel, K,n,c+tile_tc_id*TC)
  for (v,h,w) in (input_tile):
    for r = 0 to R
      for s = 0 to S
        y_out = h - r
        x_out = w - s
        if y_out<0 or x_out< 0 or y_out>TH or x_out>TW:
          continue
        result = v * kernel[r][s]
        temp_result[y_out*TW+x_out] += result
// Write the output back to memory
for th to TH:
  for tw to TW:
    y = tile_id/(W/TW)*TH+th
    x = tile_id%(W/TW)*TW+tw
    atomicAdd(Y[H*W*N+y*W*N+x*N+n],temp_result[th*TW+tw])
```

Analytical Modeling



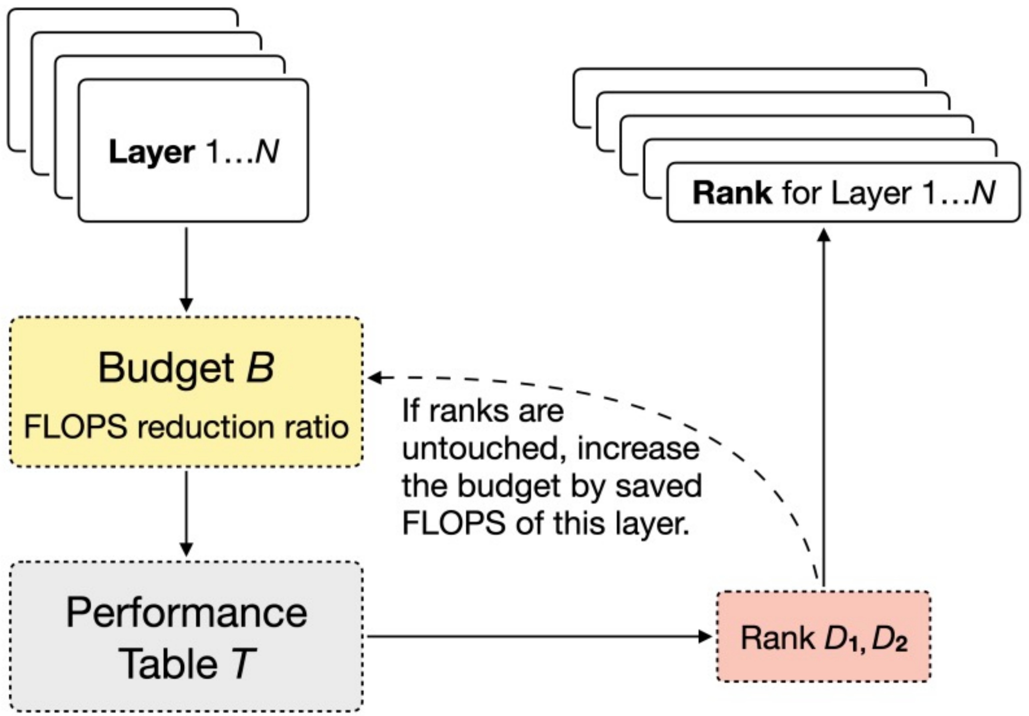
- $num\_blks = H/TH \times W/TW \times C/TC$
- $waves = ceil\ ((max\_ths \times occupancy)/(num\_blks \times blk\_dim))$
- $blks\_wave = (max\_ths \times occupancy)/blk\_dim$
- $flops\_blk = 2 \times TH \times TW \times TC \times N$
- $flops\_wave = blks\_wave \times flops\_blk$
- $time\_wave = flops\_wave/(comp\_thr \times occupancy)$
- $estimated\_time = time\_wave \times waves$

Hardware-aware Rank Determination

> Importance of rank D1 and D2:

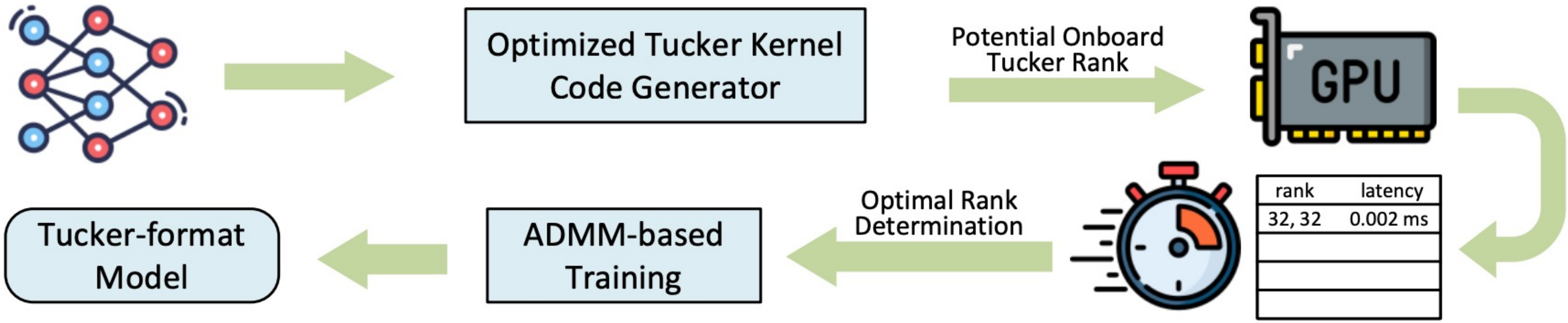
- Task accuracy
- Practical speedup
- Overall computational cost

> Proposed rank search strategy:



TDC Framework

> Overview of our TDC framework for generating TKD-compressed CNN models with high-performance inference code on GPUs:



Experiments

Accuracy table

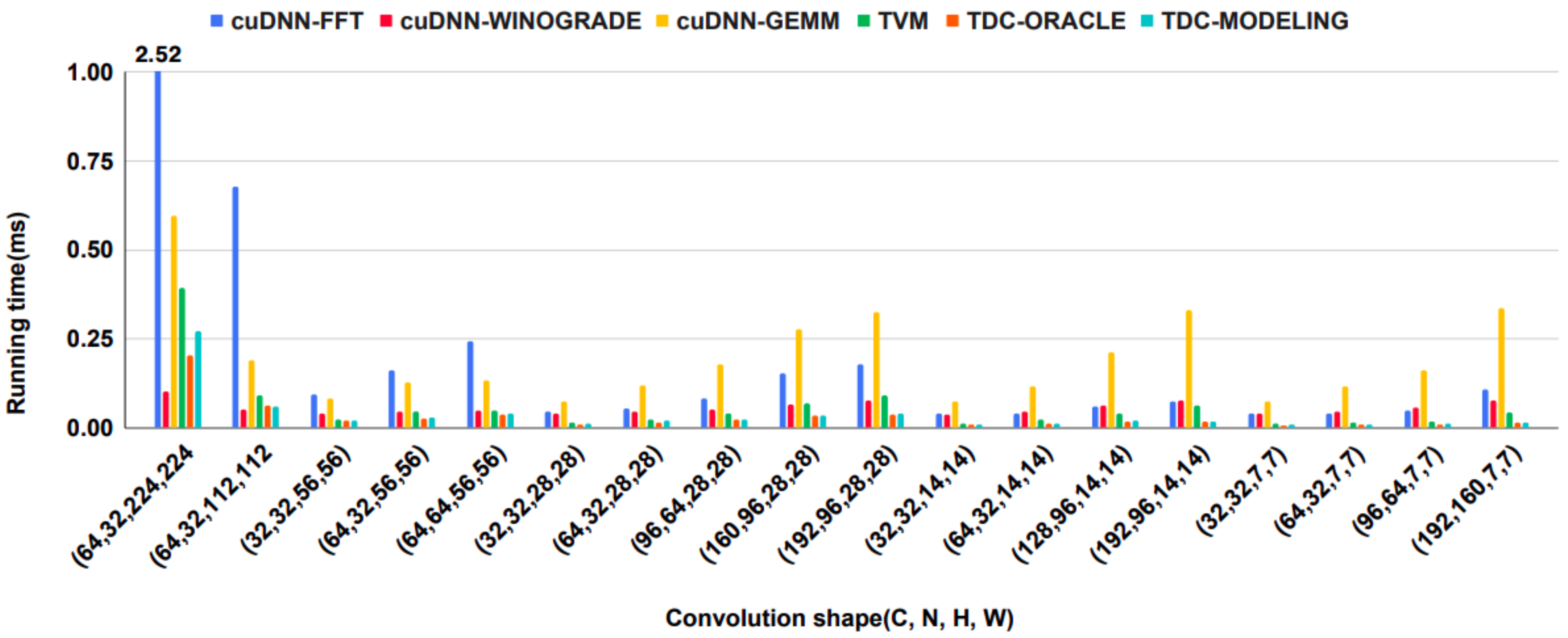
Model		Compression Method	Top-1/Drop (%)	FLOPs↓
ResNet-18	Original [14]	No compr.	69.75/-0.00	N/A
	FPGM [16]	Pruning	68.41/-1.34	42%
	DSA [27]	Pruning	68.61/-1.14	40%
	SCOP [37]	Pruning	68.62/-1.13	45%
	TRP [40]	MD	65.51/-4.24	60%
	Stable [33]	CPD	69.06/-0.69	65%
	Opt. TT [42]	TTD	69.29/-0.46	60%
	Std. TKD [19]	TKD	66.65/-3.10	60%
	MUSCO [13]	TKD	69.28/-0.47	58%
	TDC	TKD	69.70/-0.05	63%
ResNet-50	Original [14]	No compr.	76.13/-0.00	N/A
	FPGM [16]	Pruning	75.59/-0.54	42%
	HRank [24]	Pruning	74.98/-1.15	44%
	TDC	TKD	77.46/+1.33	40%
	Stable [33]	CPD	74.66/-1.47	60%
	TDC	TKD	76.42/+0.29	60%
VGG-16	Original [14]	No compr.	71.59/-0.00	N/A
	CC [22]	MD	68.81/-2.78	50%
	TDC	TKD	71.62/+0.03	80%
DN-1	Original [14]	No compr.	74.43/-0.00	N/A
	TDC	TKD	76.33/+1.90	10%
DN-2	Original [14]	No compr.	76.88/-0.00	N/A
	TDC	TKD	76.92/+0.04	10%

Accuracy summary

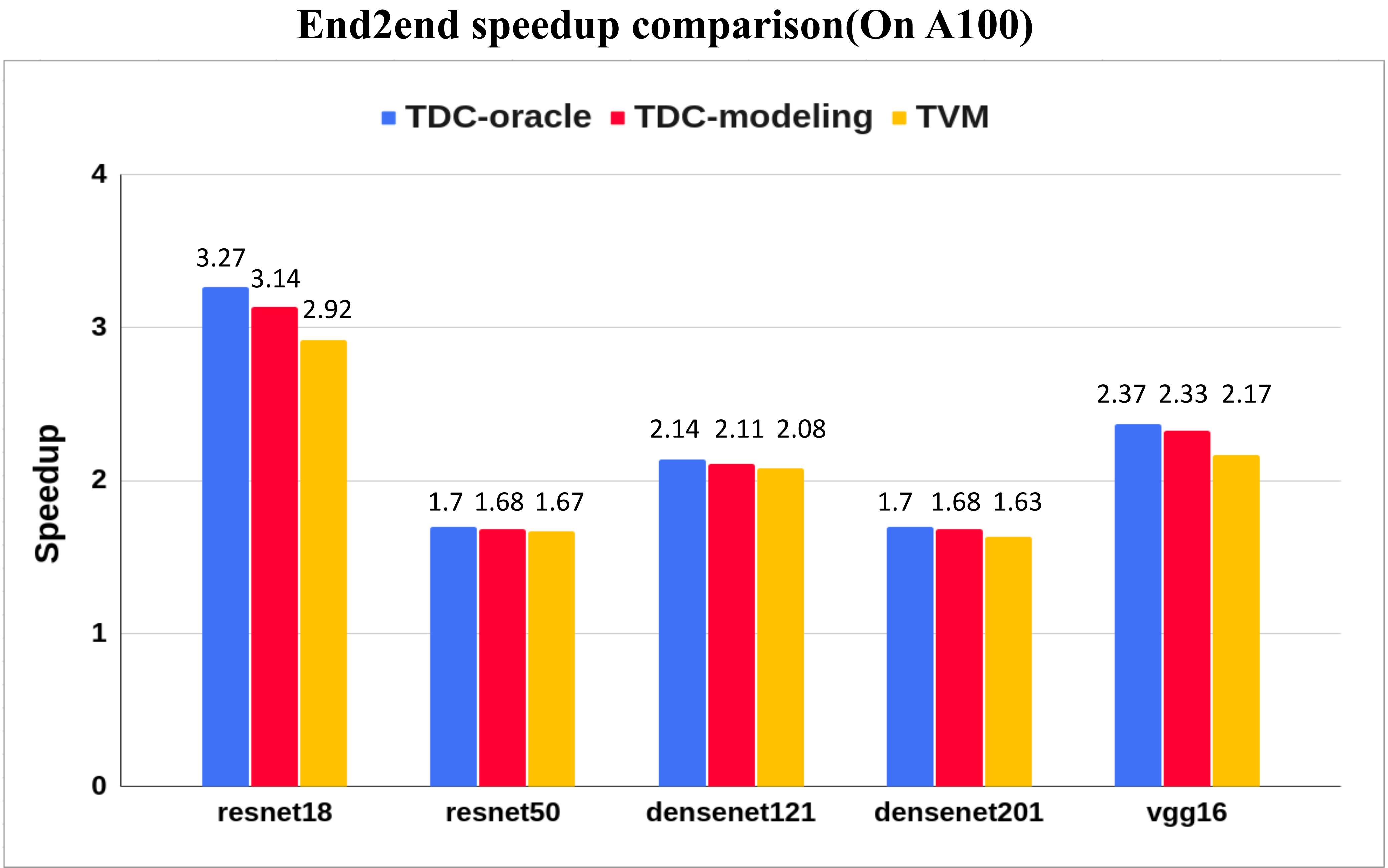
- 0.05% accuracy loss on Resnet-18
- 0.29% accuracy increment on Resnet-50
- 0.03% accuracy increment on Vgg-16
- 1.90% accuracy increment on Densenet-121
- 0.04% accuracy increment on Densenet-201

Experiments

Layer-wise TDC kernel performance evaluation(On A100)



Experiments



Thank you!

Any questions are welcome

Contacts: Lizhi Xiang: u0814474@umail.utah.edu
Yinmiao: miao.yin@rutgers.edu