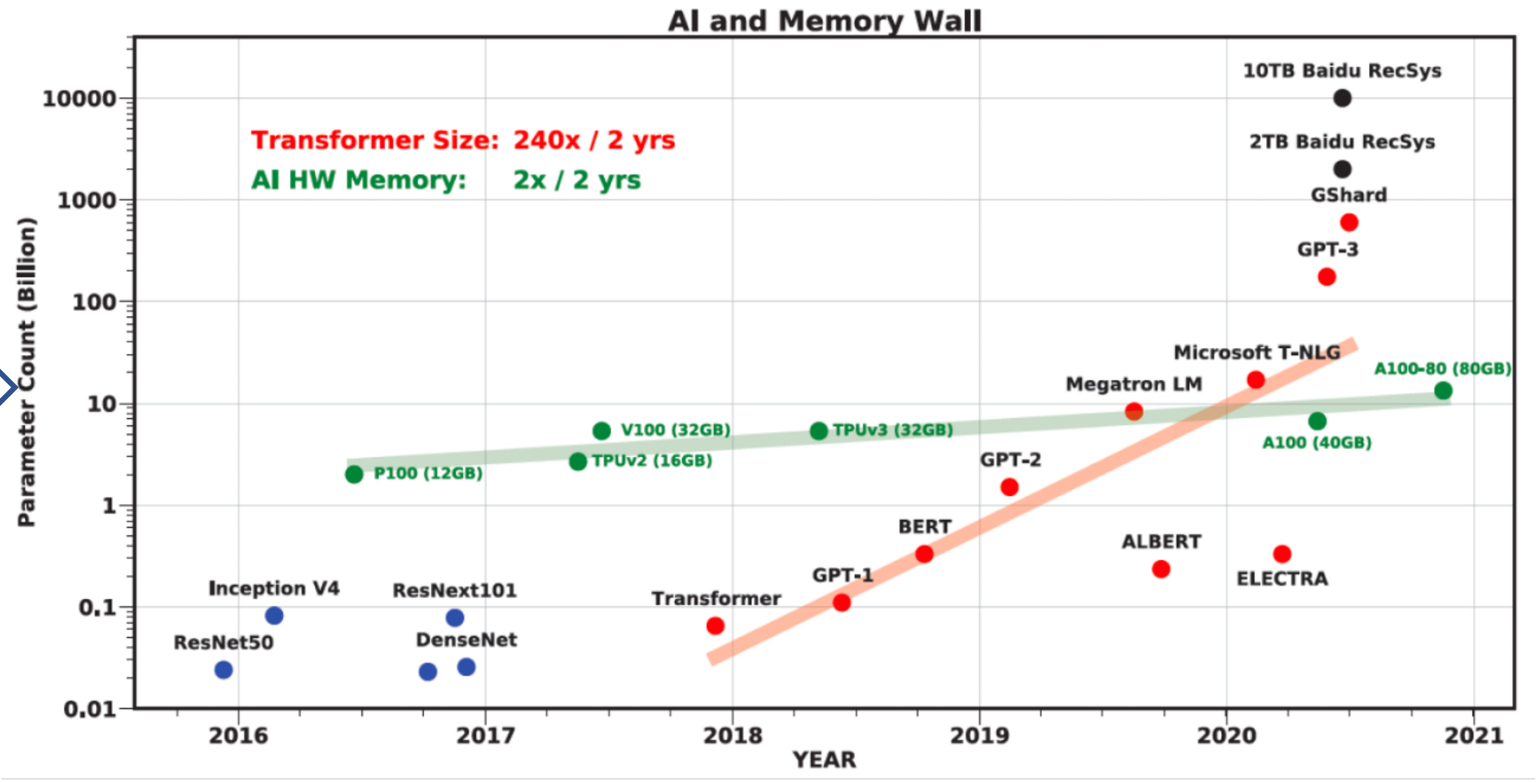
CSwap: A Self-Tuning Compression Framework for Accelerating Tensor Swapping in GPUs

Ping Chen[*] , Shuibing He[*], Xuechen Zhang[#], Shuaiben Chen[*]
Peiyi Hong[*], Yanlong Yin[$], Xian-He Sun[+], Gang Chen[*]

* 浙江大学 Zhejiang University
# Washington State University
$ Zhejiang Lab 之江实验室
+ Illinois Tech

Deep Neural Network is Popular



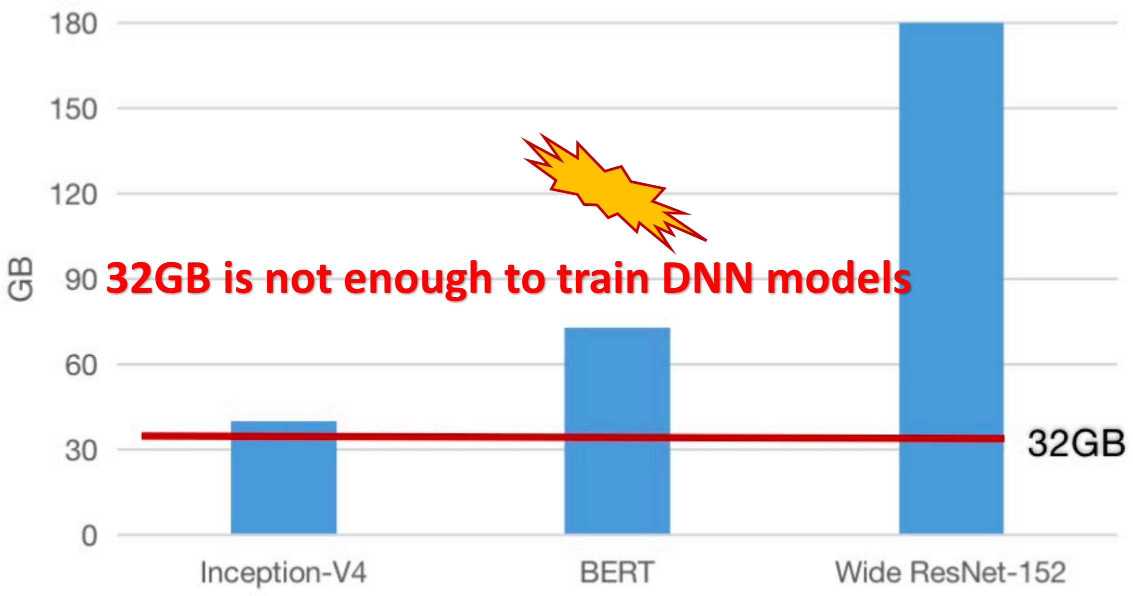
Explosive DNN Model Size

The Shortage of GPU Memory

https://www.microway.com/hpc-tech-tips/nvidia-tesla-v100-price-analysis/

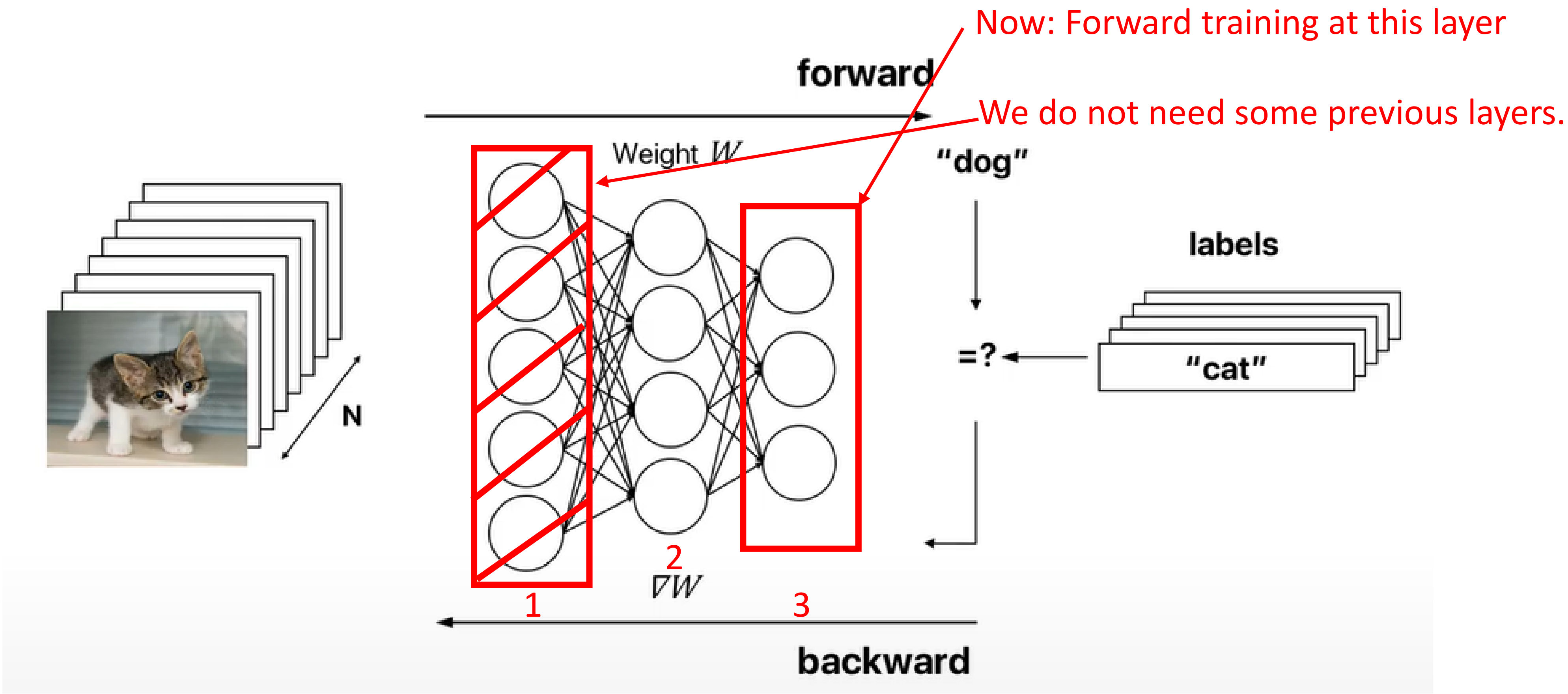
Cost $10000

GPU	Memory/GB	Bandwidth	Tensor core	TFLOPS
V100(SXM2)	32 HBM2	900	640	15.7
TITAN RTX	24 GDDR6	672	576	16.3
P100(SXM2)	16 HBM2	732	NA	10.6
TITAN V	12 HBM2	652.8	640	15
RTX 2080Ti	11 GDDR6	616	544	13.4
RTX 2080	8 GDDR6	448	368	10.1
RTX 2070	8 GDDR6	448	288	7.5
TITAN Xp	12 GDDR5X	547.7	NA	12
RTX 1080Ti	11 GDDR5X	484	NA	11.3
TITAN X	12 GDDR5	336.5	NA	11
GTX 1080	8 GDDR5X	484	NA	8.9
RTX 1070Ti	8 GDDR5	256	NA	8.1
RTX 1070	8 GDDR5	256	NA	6.5
RTX 1060	6 GDDR5	256	NA	4.4



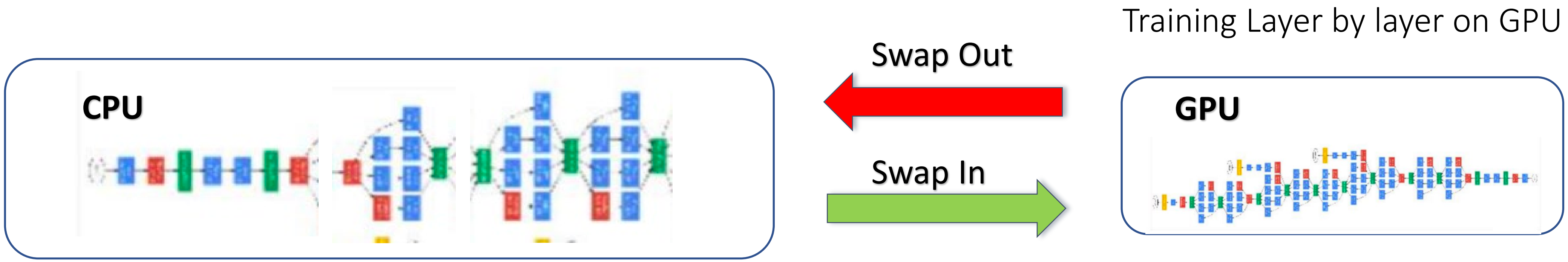
Current GPU cannot support DNN training because of GPU memory shortage

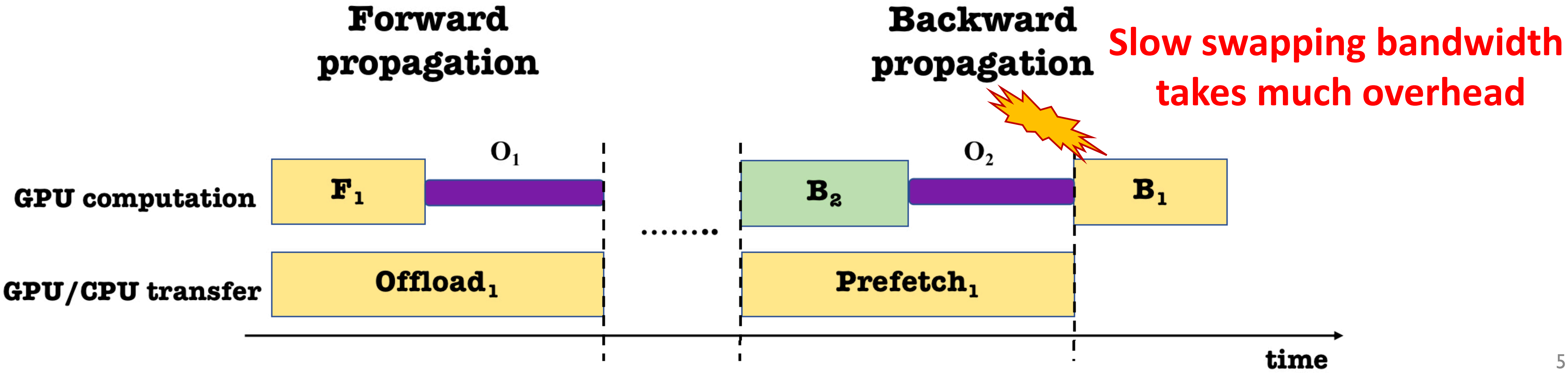
The Background of Deep Neural Network



During Layer-N training procedure, GPU can only visit the tensors which have dependency with Layer-N

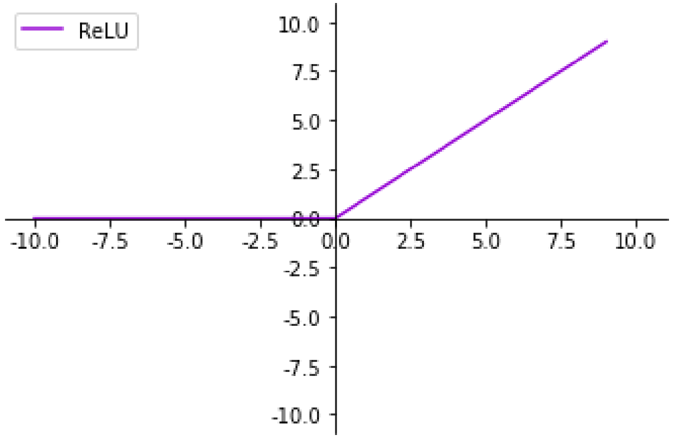
The GPU–CPU Swapping Solution





Rhu, M., Gimelshein, N., Clemons, J., Zulfiqar, A., & Keckler, S. W. (2016). VDNN: Virtualized deep neural networks for scalable, memory-efficient neural network design. *MICRO, 2016-Decem*.

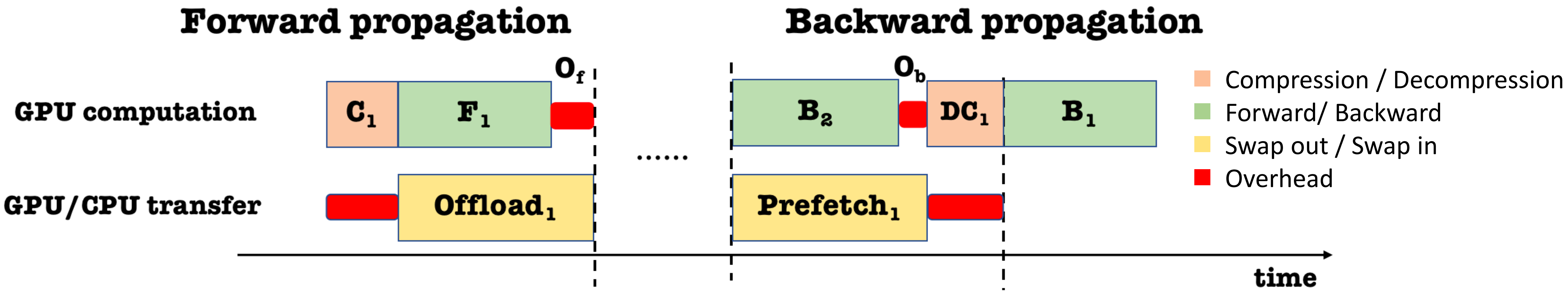
The Swapping with Compression Solution



ReLU Layers => Tensor Sparsity

Compressing all sparse tensors (after-ReLU layers) before swapping out and **decompress** them after swapping in.

Not Optimal



Rhu, M., O'Connor, M., Chatterjee, N., Pool, J., Kwon, Y., & Keckler, S. W. (2018). Compressing DMA Engine: Leveraging Activation Sparsity for Training Deep Neural Networks. *HPCA'18*

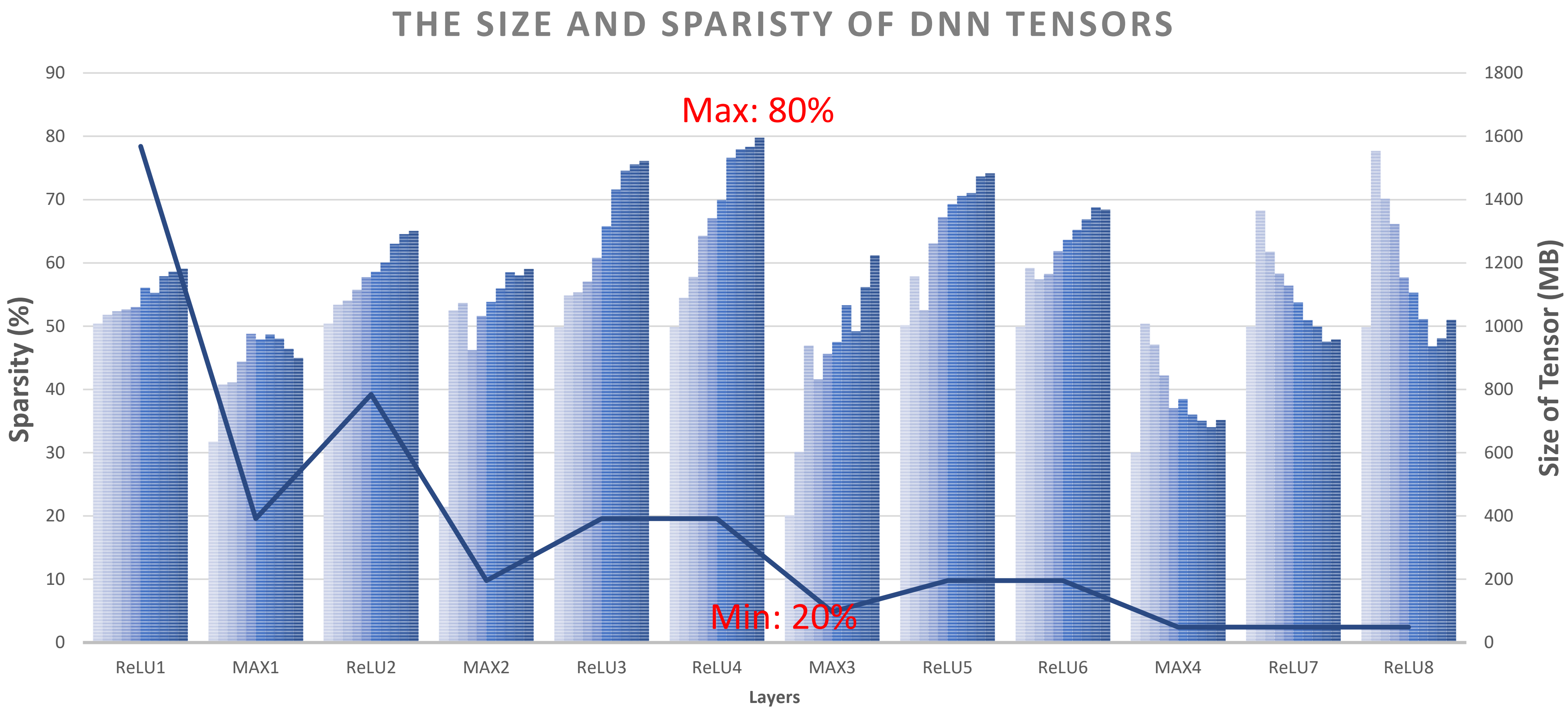
Related Works (Swapping and Compression)

Technique	Compression	Compression unit/location	Portability	Compression Optimization
vDNN[MICRO’16]	✗	N/A	✓	N/A
cDMA [HPCA’18]	✓	GPU	✗	✗
vDNN++ [IPDPS’19]	✓	CPU	✓	✗
CSwap [CLUSTER’21]	✓	GPU	✓	✓

Compression Optimization (Tensor Selection): ✗ means compressing all sparse tensors without optimization or not.

Rhu, M., Gimelshein, N., Clemons, J., Zulfiqar, A., & Keckler, S. W. (2016). VDNN: Virtualized deep neural networks for scalable, memory-efficient neural network design. *MICRO’16*.
Rhu, M., O’Connor, M., Chatterjee, N., Pool, J., Kwon, Y., & Keckler, S. W. (2018). Compressing DMA Engine: Leveraging Activation Sparsity for Training Deep Neural Networks. *HPCA’18*
Shriram, S. B., Garg, A., & Kulkarni, P. (2019). Dynamic memory management for GPU-based training of deep neural networks. *IPDPS’2019*

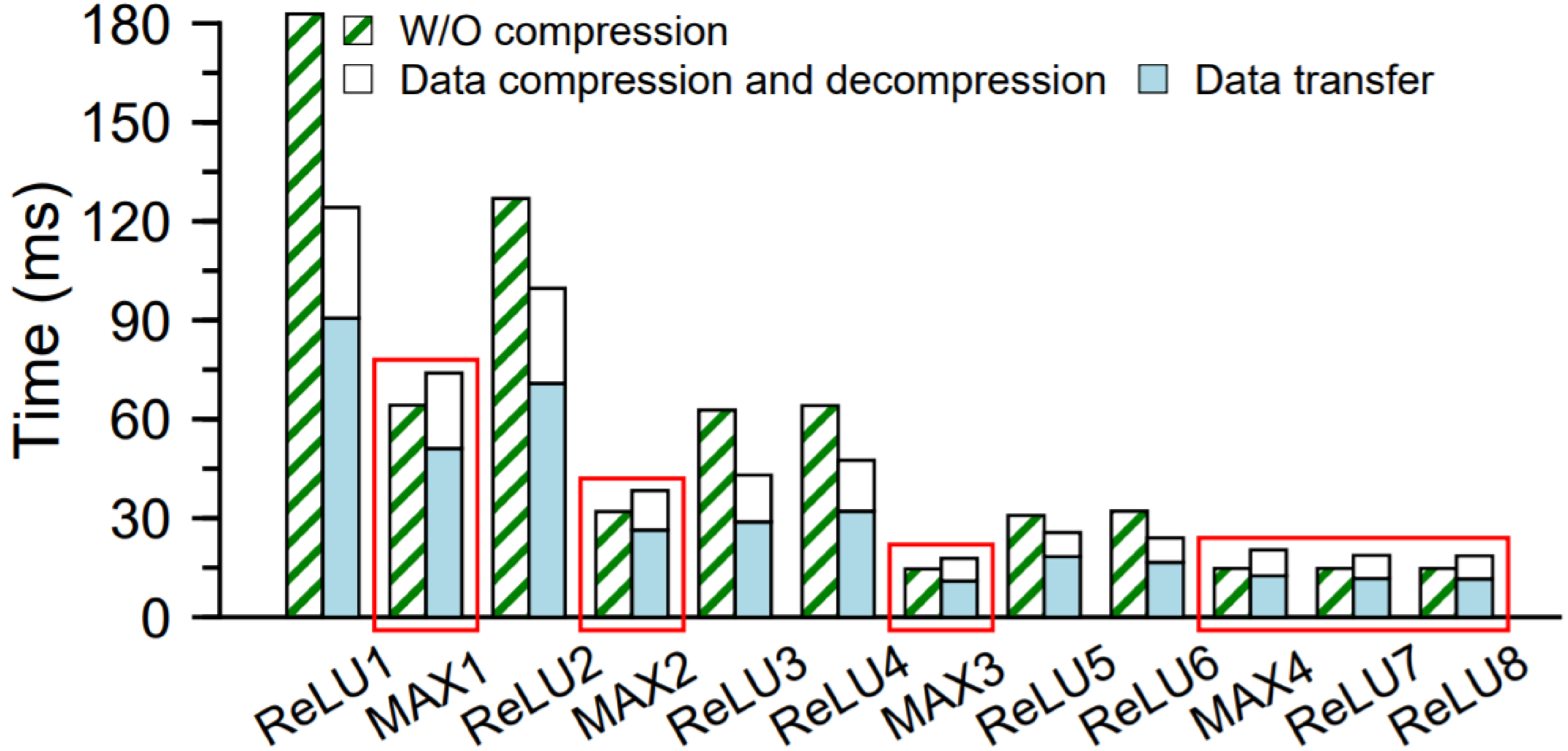
Observation 1: Changing Sparsity of Tensors



Some DNN tensors sparsity changes constantly during training the tensor size changes across layers .

*Figure: We evaluate ReLU output tensors in VGG16 on ImagNet. 50 epochs.

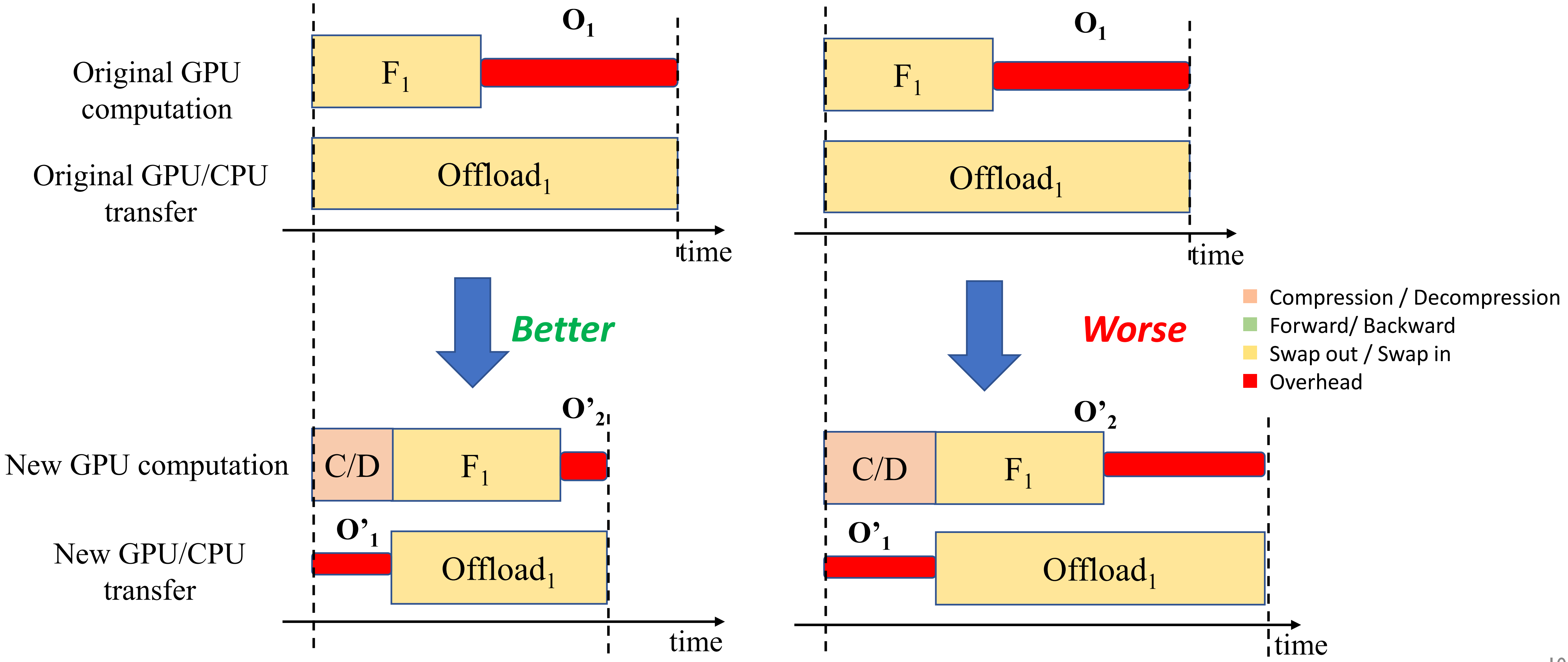
Observation 2: Ineffectiveness of Compressing all Tensors



Some DNN tensors are unworthy being compressed.

*Figure: We evaluate on VGG16 across ImagNet.

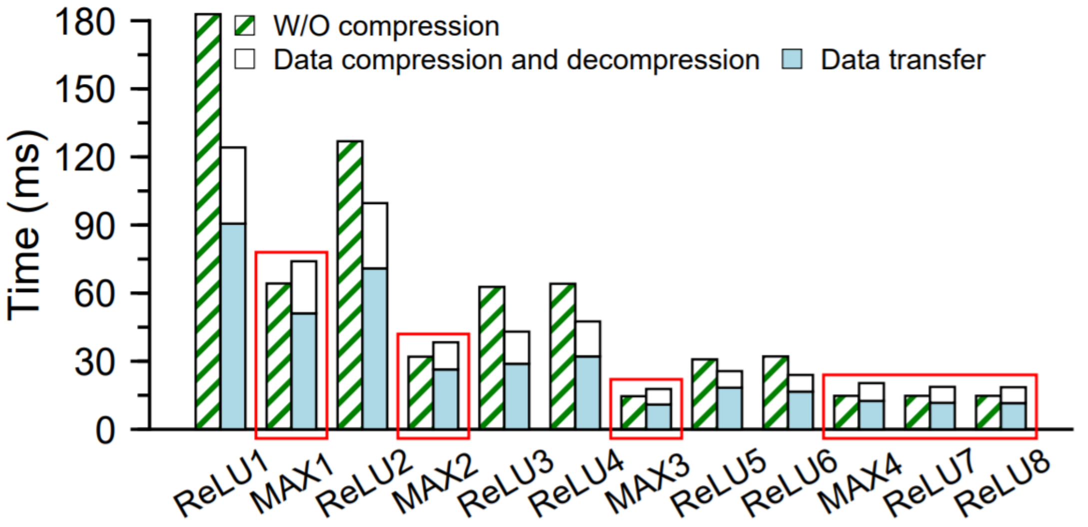
Observation 2: Ineffectiveness of Compressing all Tensors



Objectives of CSWAP

Software-level framework (Portability) + **Self-tuning** + **Selective compression**

CSWAP



Policy

With compression:

- ReLU[1-6]

Without compression:

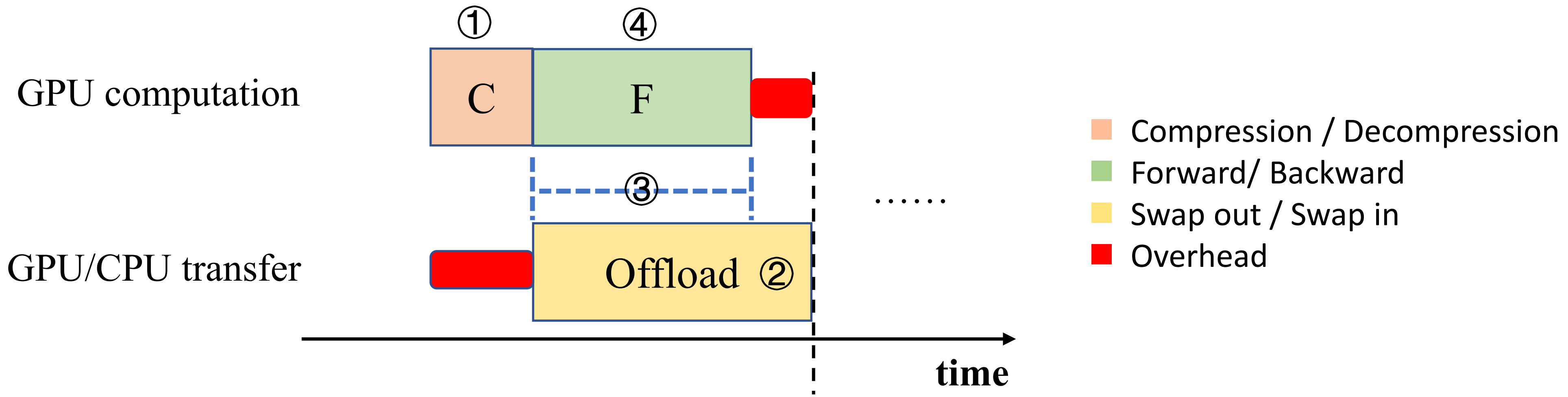
- MAX[1-4], ReLU[7-8]

Challenges of CSWAP

Challenges 1 : How to determine the compression policy for a sparse tensor?

- Different **sparsity** ①②;
- Different **sizes** ①②;
- Different **overlap time** ③;
- Different **forward and backward time** ④.

These metrics influence the overall training time.

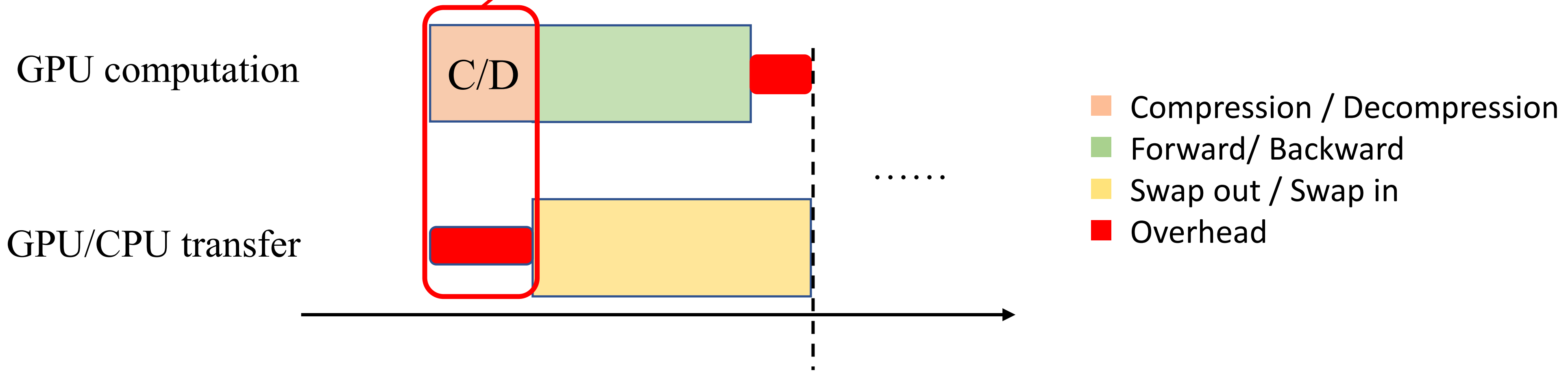


Challenges of CSWAP

Challenges 2 : How to predict the (de)compression time?

- ➢ Without (de)compression time, we cannot make decisions.

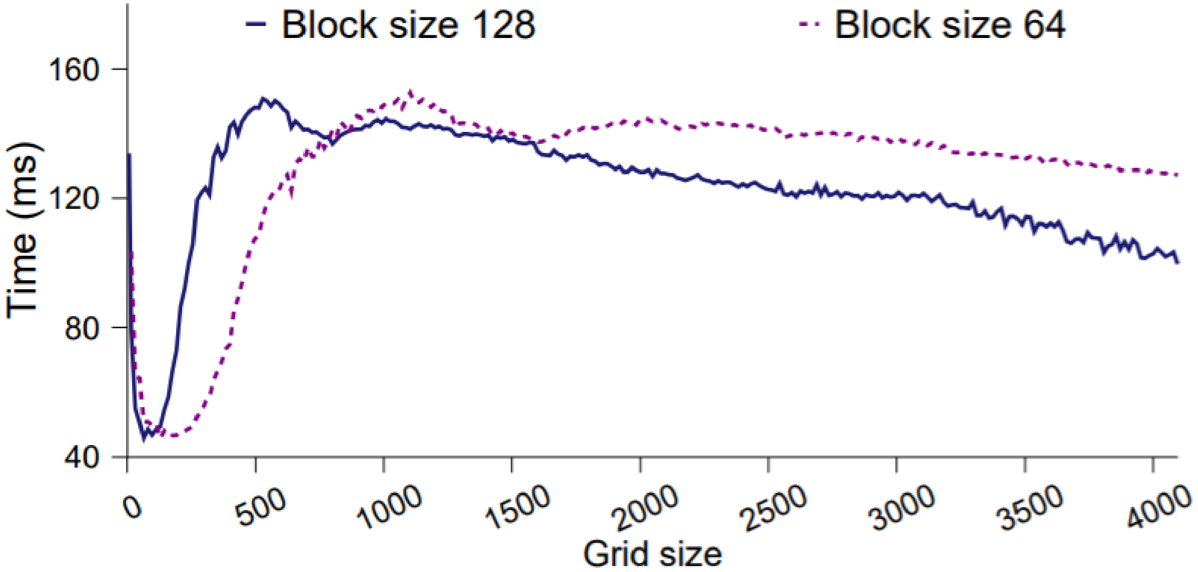
Important but uneasy to know



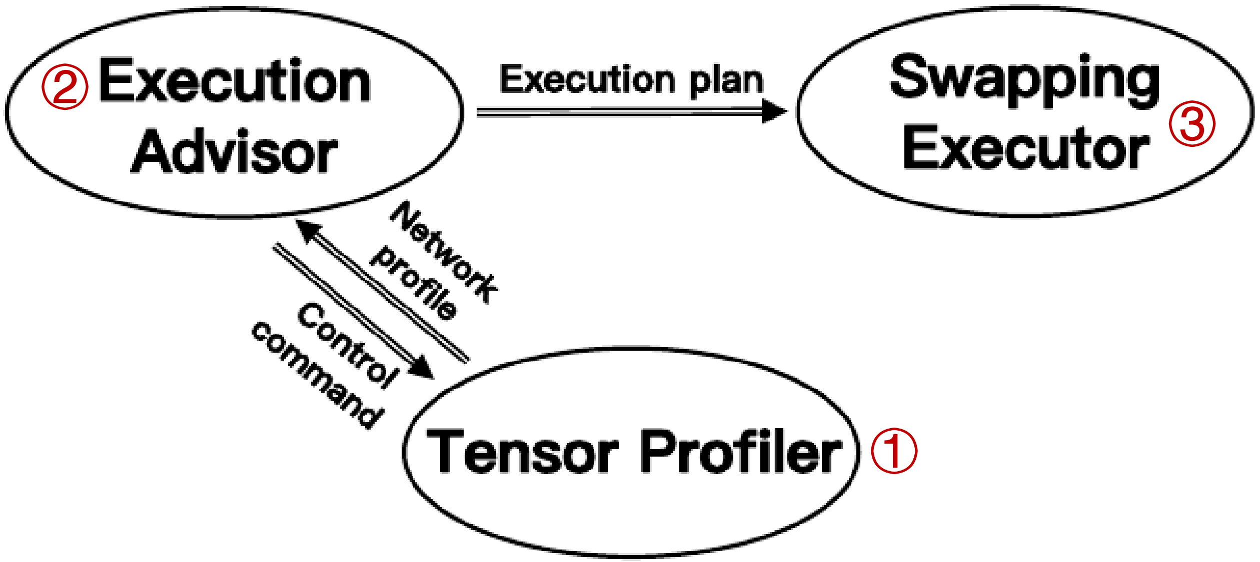
Challenges of CSWAP

Challenges 3 : the compression/decompression algorithm performance varies severely with different GPU settings.

- **Super parameters** : GPU has Grid size and Block size.
- Bruce force search (Grid search) needs hours.



Overview of CSWAP



① ***The tensor profiler:*** *Collecting* tensor sparsity, size, and execution time of layers.
② ***Execution Advisor***: Making *policy,* includes compression decision and GPU settings for (de)compression operations.
③ ***Swapping Executor***: DNN training.

1. Determining Cost-Effectiveness of Tensor Compression

- We compare the swapping cost with compression ***T*** with the swapping cost without compression ***T'***
  - ***T' > T => compression***
  - ***T'< T => no compression***

$$T' = \max(\frac{Size^t}{BW_{d2h}} - Hidden_f^t, 0) + \max(\frac{Size^t}{BW_{h2d}} - Hidden_b^t, 0) \quad (1)$$

$$T = Time_c^t + Time_{dc}^t + O_f + O_b \quad (2)$$

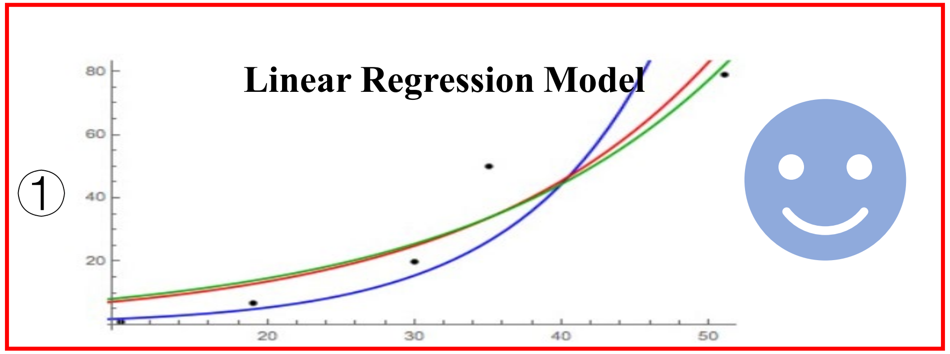
$$O_f = \max(\frac{Size^t \times (1 - Sparsity^t)}{BW_{d2h}} - Hidden_f^t, 0) \quad (3)$$

$$O_b = \max(\frac{Size^t \times (1 - Sparsity^t)}{BW_{h2d}} - Hidden_b^t, 0) \quad (4)$$

Symbol	Meaning	Profiling
$Size^t$	size of tensor t	one time
BW_{h2d}	effective PCIe bandwidth from CPU to GPU	one time
BW_{d2h}	effective PCIe bandwidth from GPU to CPU	one time
$Hidden_f^t$	overlapped swapping latency in forward propagation of tensor t	one time
$Hidden_b^t$	overlapped swapping latency in backward propagation of tensor t	one time
$Sparsity^t$	sparsity of tensor t	epoch
$Time_c^t$	compression time of tensor t	offline
$Time_{dc}^t$	decompression time of tensor t	offline

2. Prediction of (De)compression Time

①



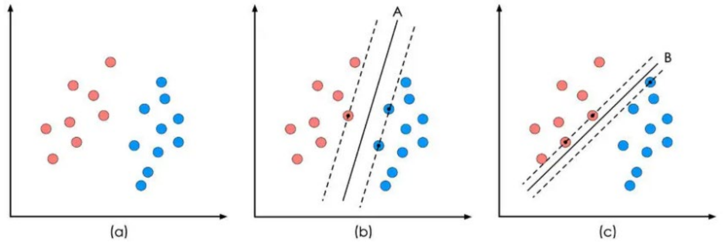
Compression or Decompression time

Size
Sparsity

Compression Algorithm

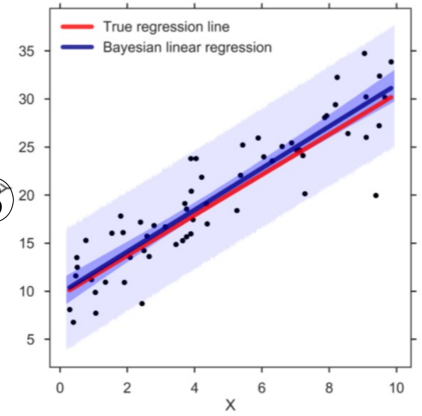
SVM

②



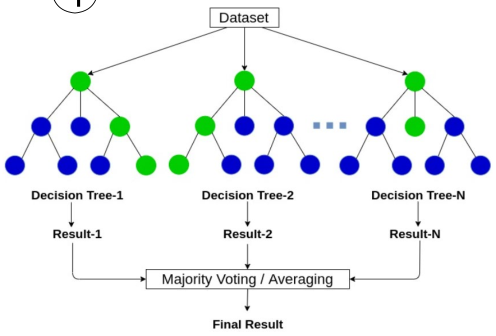
Bayesian Regression

③



Decision Tree

④



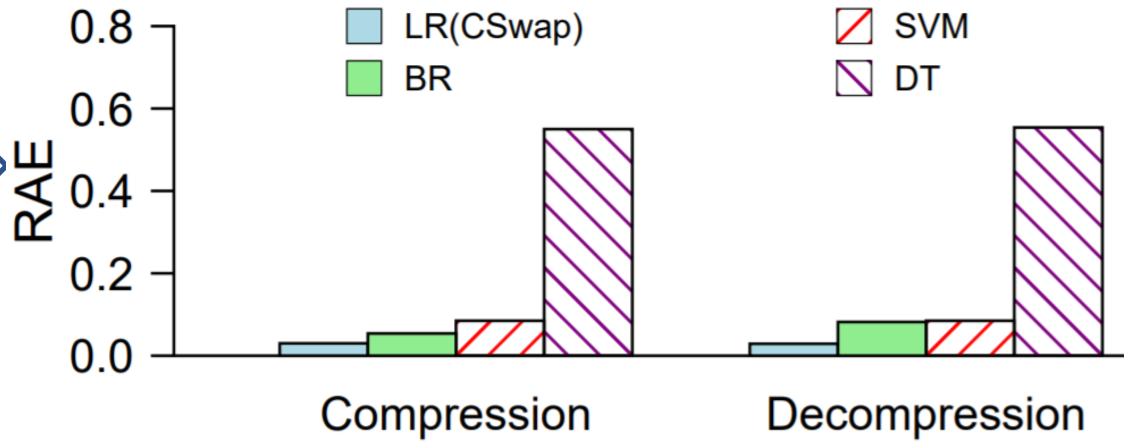


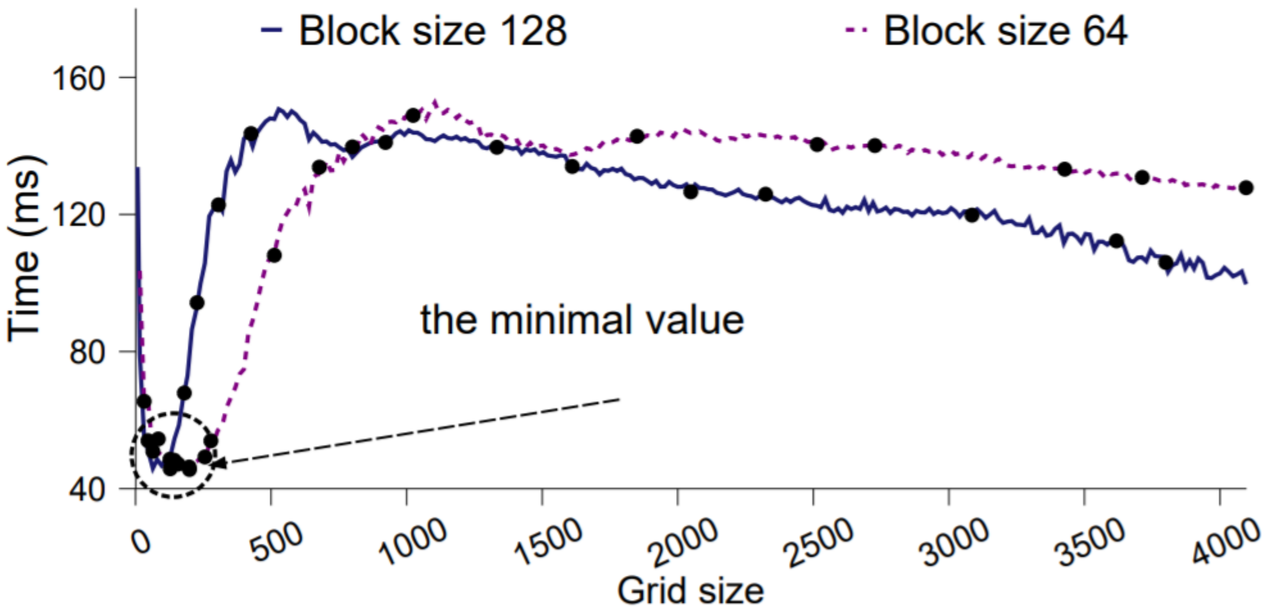
Figure: The accuracy of (de)compression time predication using LR, BR, SVM, and DT models.

3. Setting GPU Parameters for Compression Kernels

Algorithm 1 BO search algorithm for choosing GPU parameters for (de)compression kernels

Require: s_1: the number of initial samples; s_2: the times of attempts to find the optimal solution;

$bayes\_opt \leftarrow$ new $bayes\_opt()$ ▷ Create a CSWAP BO search engine
$\boldsymbol{D} \leftarrow \varnothing$ ▷ Dataset of previously observed samples
for i = 1, 2, ..., s_1 **do**
  $g \leftarrow random(0..4096)$ ▷ g denotes grid size
  $b \leftarrow random(64,128)$ ▷ Set block size as 64 or 128
  $p \leftarrow$ ¡g,b¿
  $y \leftarrow bayes\_opt.exec(p)$ ▷ obtain sum of $Time_c^t$ and $Time_{dc}^t$
  $\boldsymbol{D}.append(p,y)$ ▷ Add the new sample to $\boldsymbol{D}$
end for
$bayes\_opt.update(\boldsymbol{D})$ ▷ estimate posterior distribution and acquisition function
for i = 1, 2, ..., s_2 **do**
  $p \leftarrow bayes\_opt.select()$ ▷ select the next point to search
  $y \leftarrow bayes\_opt.exec(p)$
  $\boldsymbol{D}.append(p,y)$
  $bayes\_opt.update(\boldsymbol{D})$
end for
return $bayes\_opt.optimize(\boldsymbol{D})$ ▷ return an optimal point



Explore & Exploit => Fast and jump minimum point

Hours to near 1 minutes

Bayesian Optimization

Experimental Setting

- Platform1:
  - 2.60 GHz Intel(R) Xeon(R) Gold 6126 CPU
  - NVIDIA Tesla V100 GPU with 32 GB GPU memory

- Platform2:
  - 2.10 GHz Intel(R) Xeon(R) Gold 5218R CPUs
  - RTX 2080Ti GPU, and 11 GB GPU memory

- Workloads and datasets
  - NN: AlexNet , Plain20 , VGG16 , MobileNet, ResNet and SqueezeNet (6)
  - Dataset: CIFAR10, ImageNet (2)

- Baselines
  - vDNN[1] , vDNN++[2], and cDMA[3]

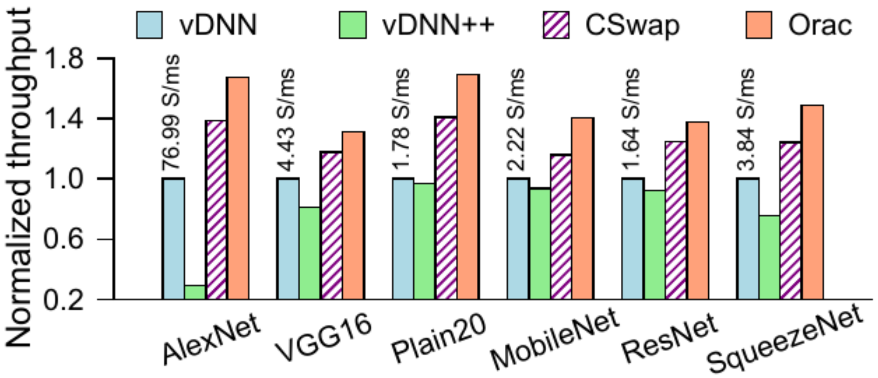
[1]"VDNN: Virtualized Deep Neural Networks for Scalable, Memory Efficient Neural Network Design," in *Proceedings of the Annual International Symposium on Microarchitecture (MICRO)*

[2]"Dynamic memory management for GPU-based training of deep neural networks," in Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS)

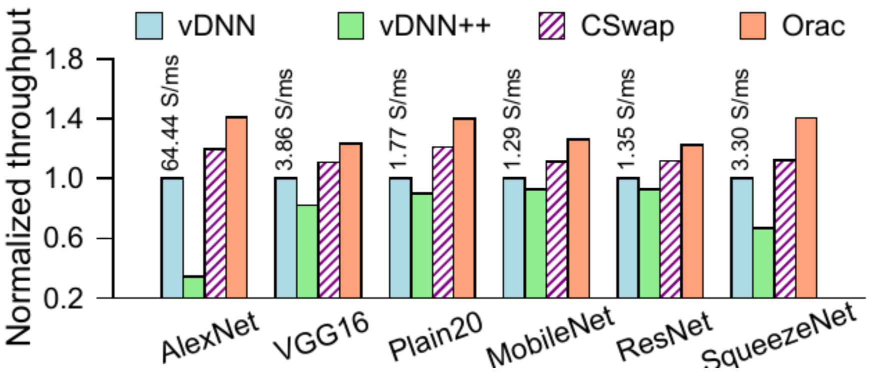
[3] "Compressing DMA Engine: Leveraging Activation Sparsity for Training Deep Neural Networks," in Proceedings of the International Symposium on High-Performance Computer Architecture (HPCA)

Eval 1: Overall Performance

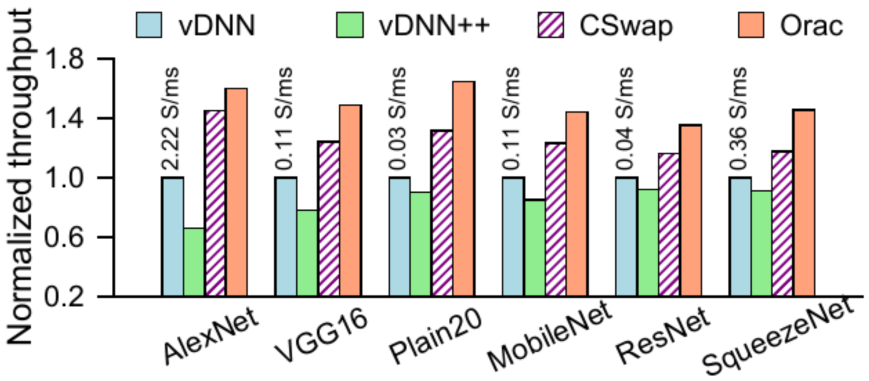
CIFAR10-V100



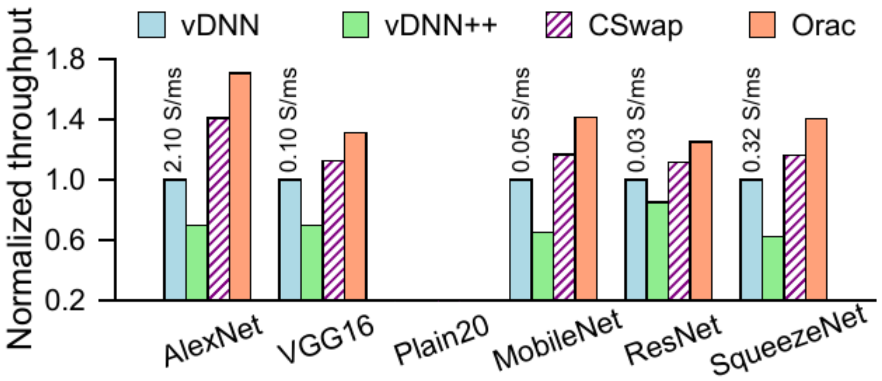
CIFAR10-2080Ti



ImageNet-V100

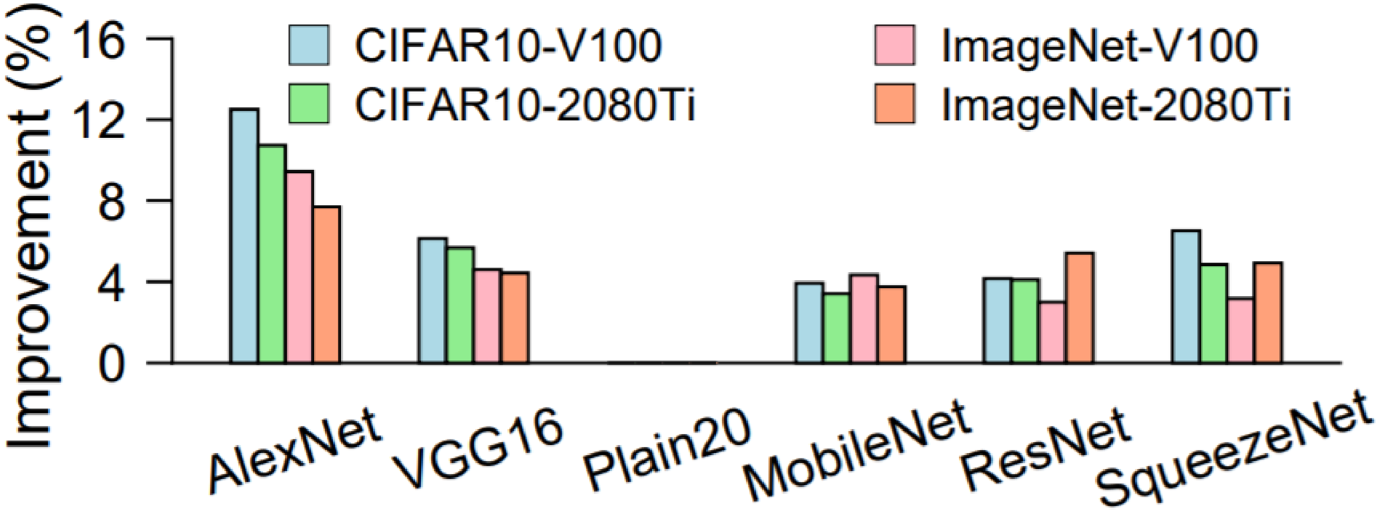


ImageNet-2080Ti



CSWAP outperforms vDNN and vDNN++ by 25% and 190% on average

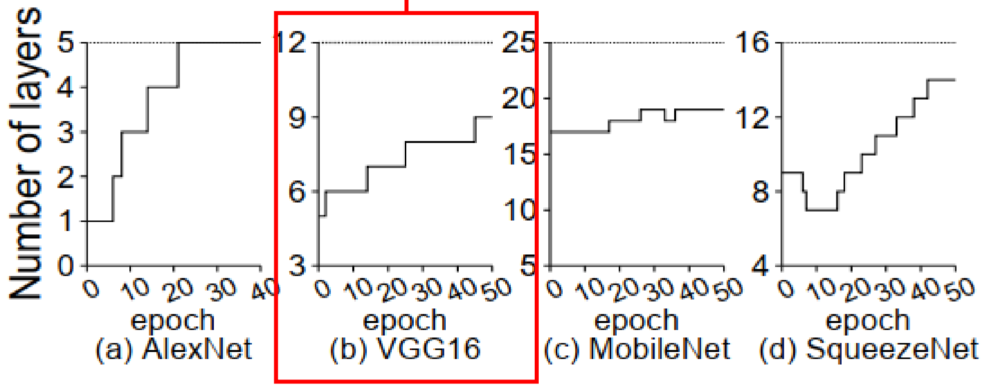
Eval 2: Effectiveness of Dynamic Tensor Compression

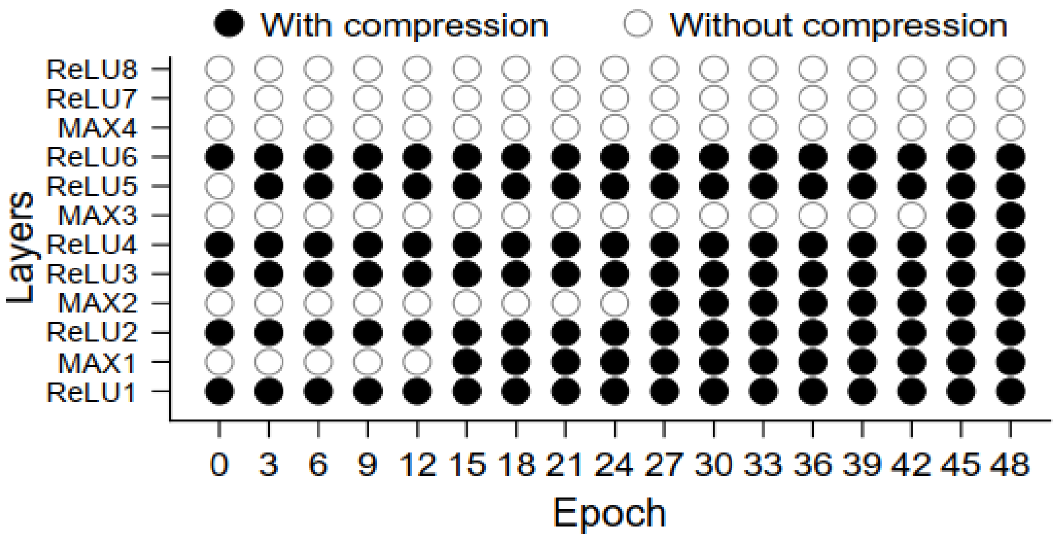


Performance improvement of CSwap over the static compression (SC) scheme.

CSWAP can improve the performance by 5.5% and 5.1% on average compared to cDMA.

Eval 3: Effectiveness of Dynamic Tensor Compression





DNN training details using CSWAP

Thanks for your attention!

* 浙江大学 Zhejiang University

WASHINGTON STATE UNIVERSITY

$ ZHEJIANG LAB 之江实验室

+ ILLINOIS TECH

Appendix

Model	ReLu layers	All layers	Ratio
AlexNet	7	21	33%
VGG19	16	38	42%
SqueezeNet	26	57	46%
MobileNet	27	83	33%
GoogleNet	64	205	31%

Appendix-1: ReLU layers

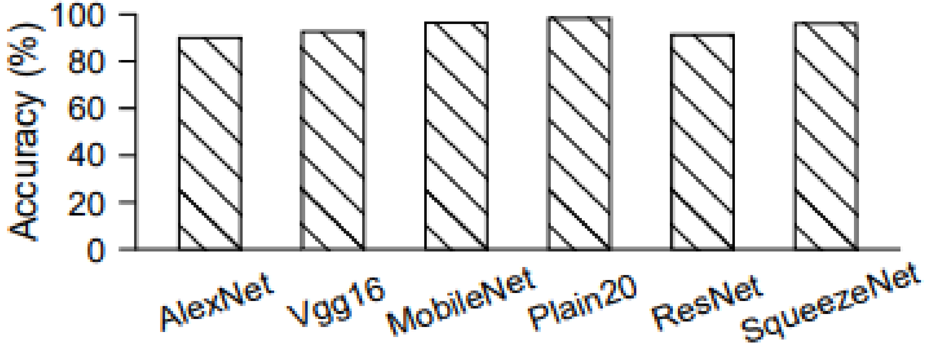


Figure 12: The compression decision accuracy based on the LR model.

Appendix

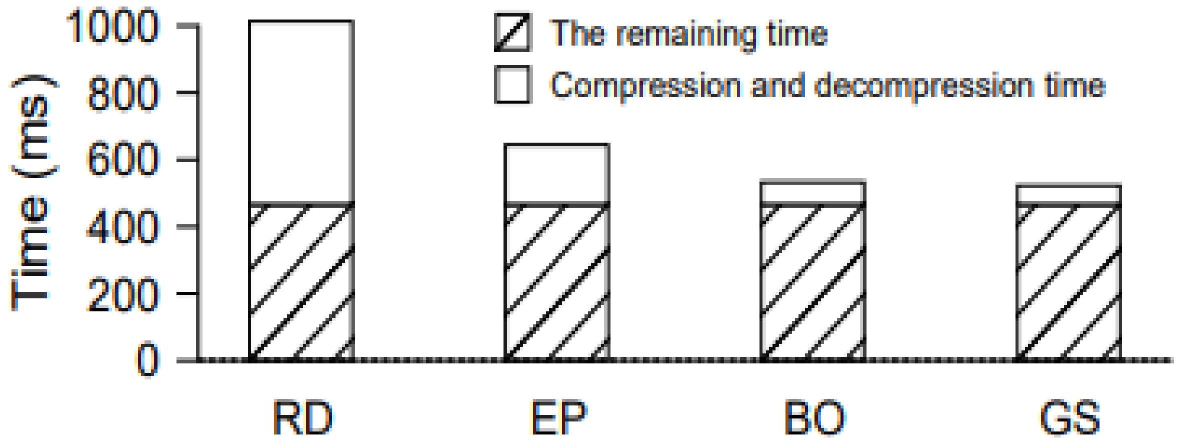


Figure 13: The average training time of VGG16 for one iteration. RD: random search, EP: expert knowledge, BO: CSwap BO search, and GS: grid search.

Appendix

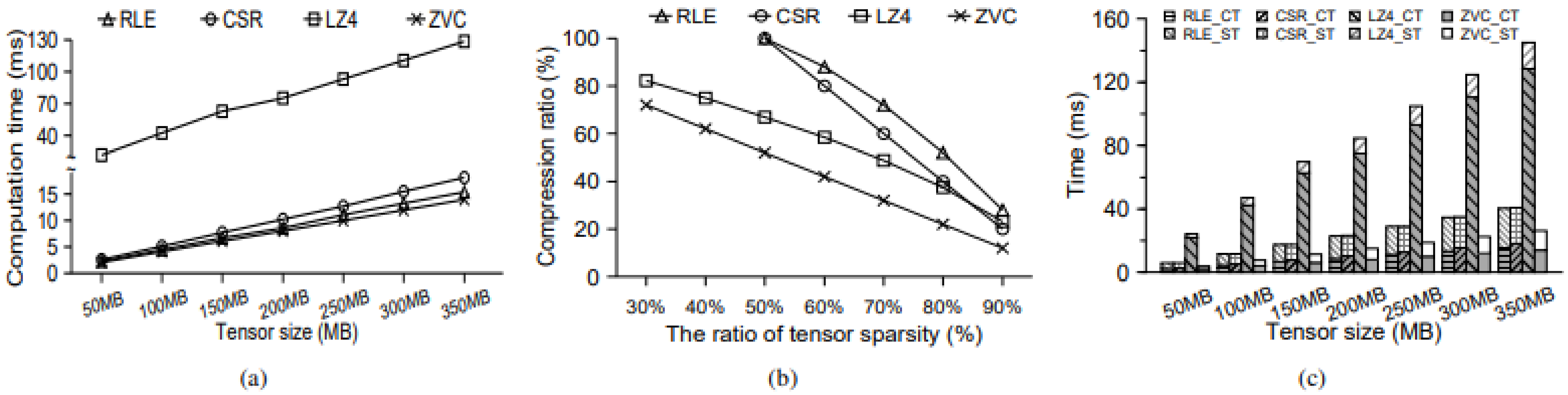


Figure 11: (a) Computation time of the compression algorithms with the tensor sparsity of 60%. (b) The compression ratio with the tensor size of 50 MB. (c) Tensor swapping time. X_CT and X_ST denote the computation time and data swaping time using the compression algorithm X.