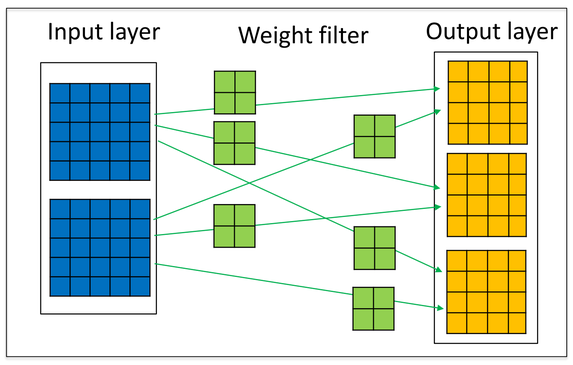
ooc_cuDNN : A Deep Learning Library Supporting CNNs over GPU Memory Capacity

Yuki Ito, Ryo Matsumiya, Toshio Endo (Tokyo Institute of Technology)

Background

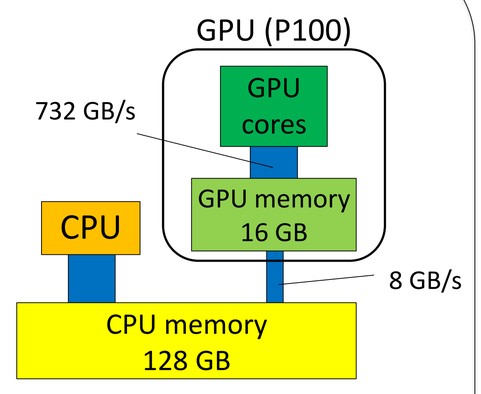
- Convolutional neural networks (CNNs) are used in many fields.
  - ➢ Image recognition, Image processing, speech recognition, etc…

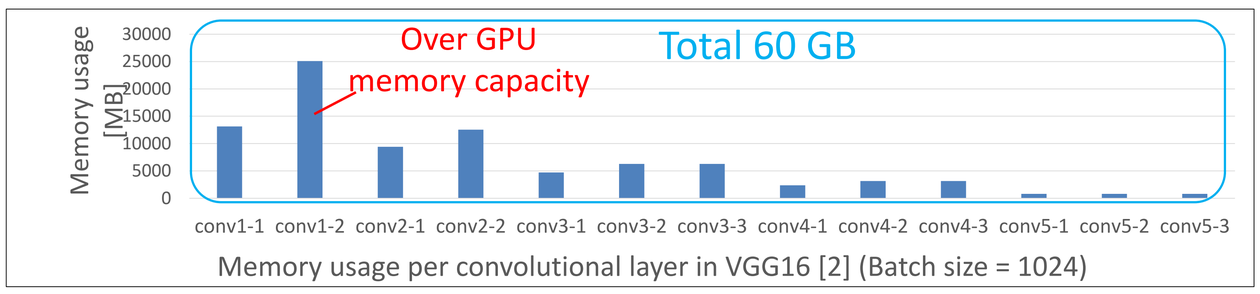


Convolution computation

- cuDNN [1] library can accelerate computation of CNNs
  - ➢ Developed by NVIDIA
  - ➢ Used by many deep learning frameworks
  - ➢ Use graphic processing units (GPUs) effectively

- ***Motivation***
  - ➢ It is hard for large scale CNNs to be computed using cuDNN
    - ❖ cuDNN can use GPU memory only
    - ❖ GPU memory capacity is limited
      - ✓ Even computation of one layer may run out of GPU memory

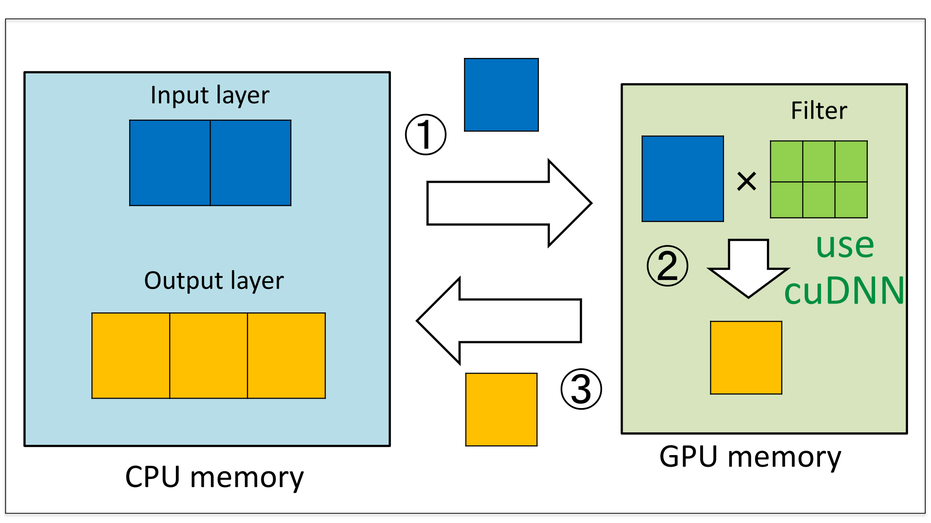




Memory usage per convolutional layer in VGG16 [2] (Batch size = 1024)

Our solution

- We designed and implemented ***ooc_cuDNN*** library.
- ***ooc_cuDNN* (out-of-core cuDNN)** supports large scale CNNs
  - ➢ Compatible with cuDNN
  - ➢ Enable to compute CNNs that exceed GPU memory capacity
    - ❖ Use both GPU and CPU memory
  - ➢ Divide layers and filters
    - ❖ Each layer (or filter) is put on GPU or CPU memory
  - ➢ Divided data are used for computation on GPU with cuDNN.
    - ❖ Swap data between CPU and GPU memory
    - ❖ Overlap CPU-GPU communication and computation



Use CPU and GPU memory

Optimization(1) : Auto-tuning division sizes

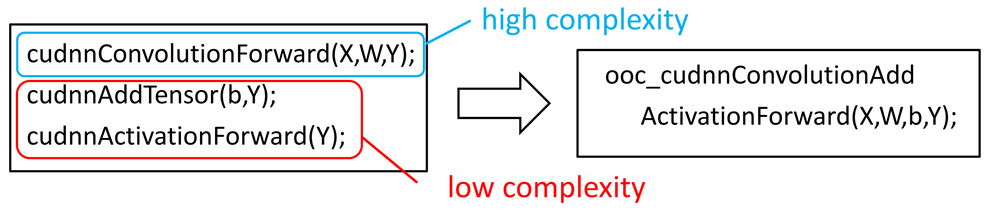
- Performance of *ooc_cuDNN* is affected by each division size.
  - ➢ Make performance model
    - ➢ Optimize division size based on the model.

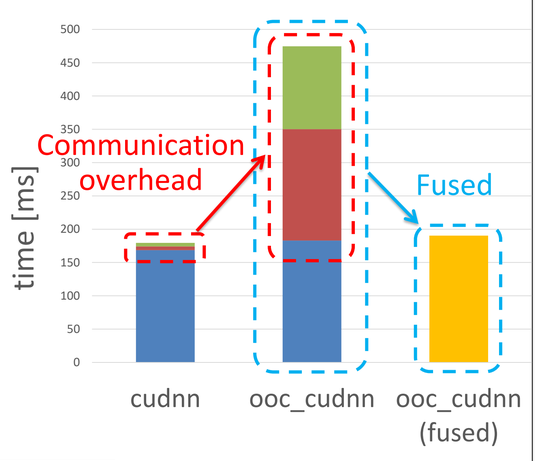
$$T_{conv} = t_{HtoD} + t_{conv} + t_{DtoH} + \left(\left\lceil \frac{c_x}{d\_c_x} \right\rceil - 1\right) \max(t_{HtoD}, t_{conv}) + \left(\left\lceil \frac{n}{d\_n} \right\rceil \left\lceil \frac{c_Y}{d\_c_Y} \right\rceil \left\lceil \frac{h_Y}{d\_h_Y} \right\rceil - 1\right) \max\left(\left\lceil \frac{c_x}{d\_c_x} \right\rceil t_{HtoD}, \left\lceil \frac{c_x}{d\_c_x} \right\rceil t_{conv}, t_{DtoH}\right)$$

Performance model of convolution

Optimization(2) : Fusion of computations

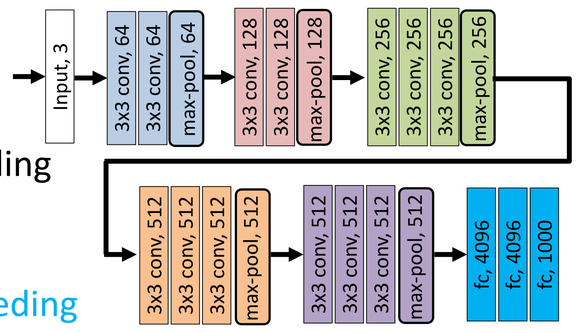
- Performance of low complexity computations is too low in *ooc_cuDNN*.
  - ➢ In those computations, communication can not be hidden completely.
  - ➢ Provide ***fused functions*** that perform high complexity computations and low complexity computations at once.



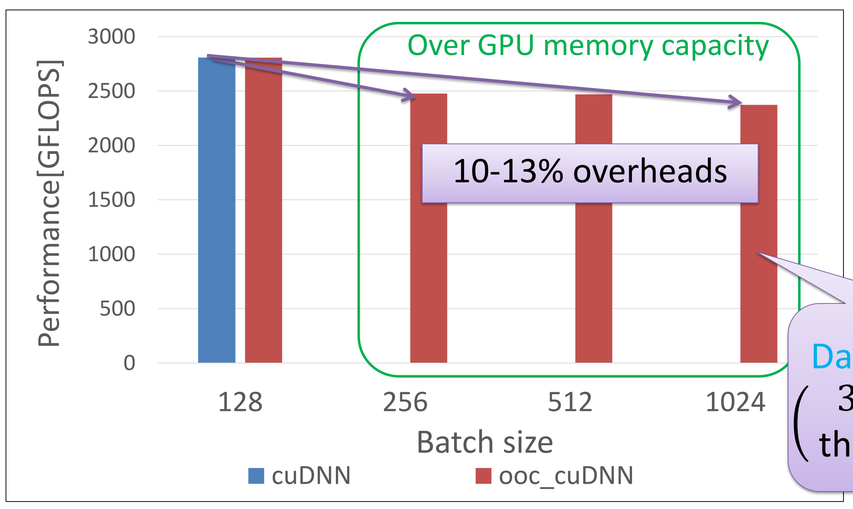


Evaluation

- Apply *ooc_cuDNN* to CNN application
  - ➢ Forward and Backward of VGG16[2]
  - ➢ The required memory size increases according to batch size.
- Experiment with Tesla P100
  - ➢ *ooc_cuDNN* enables to compute CNN exceeding GPU memory capacity.

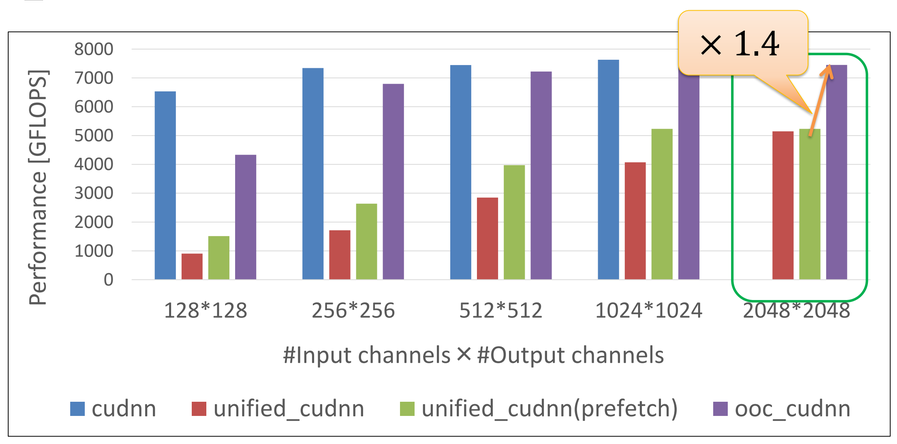


VGG16 network structure [2]



Comparison with Unified Memory

- *ooc_cuDNN* calls cudaMemcpy() explicitly.
- Recent CUDA has new mechanism named **Unified Memory**.
  - ➢ Address space of CPU and GPU are unified.
    - ❖ Data exceeding GPU memory capacity are supported by swapping mechanism.
  - ➢ Software prefetch can improve performance.
- We implemented ***unified_cuDNN*** (cuDNN with Unified Memory).
  - ➢ Perform ***Convolution*** with P100.
  - ➢ *ooc_cuDNN is* > ×1.4 faster.



Future work

- Optimization considering the entire CNN
  - ➢ Which data should be put on CPU memory?
  - ➢ Which computation should be fused?
- Co-design with existing deep learning frameworks
  - ➢ e.g. TensorFlow, Caffe2
- Support distributed computation

[1] NVIDIA Cooperation, NVIDIA cuDNN https://developer.nvidia.com/cudnn
[2] Karen Simonyan et al. Very Deep Convolutional Networks for Large-Scale Image Recognition, ICLR, 2015

This work is supported by JST-CREST, "Software Technology that Deals with Deeper Memory Hierarchy in Post-petascale Era"