vDNN: Virtualized Deep Neural Networks for Scalable, Memory-Efficient Neural Network Design

Minsoo Rhu, Natalia Gimelshein, Jason Clemons, Arslan Zulfiqar, and Steve Keckler

NVIDIA

MICRO 2016

Motivation

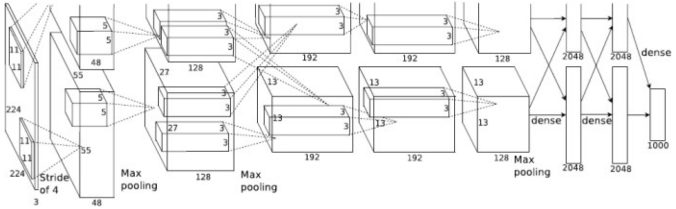
Trend: large and deep neural networks

Convolutional neural networks (CNNs)

Grown from 7 layers to 152 layers (between 2012 to 2015)

Recurrent neural networks (RNNs)

Employ 100s to 1000s of layers (when the recurrence is unrolled)

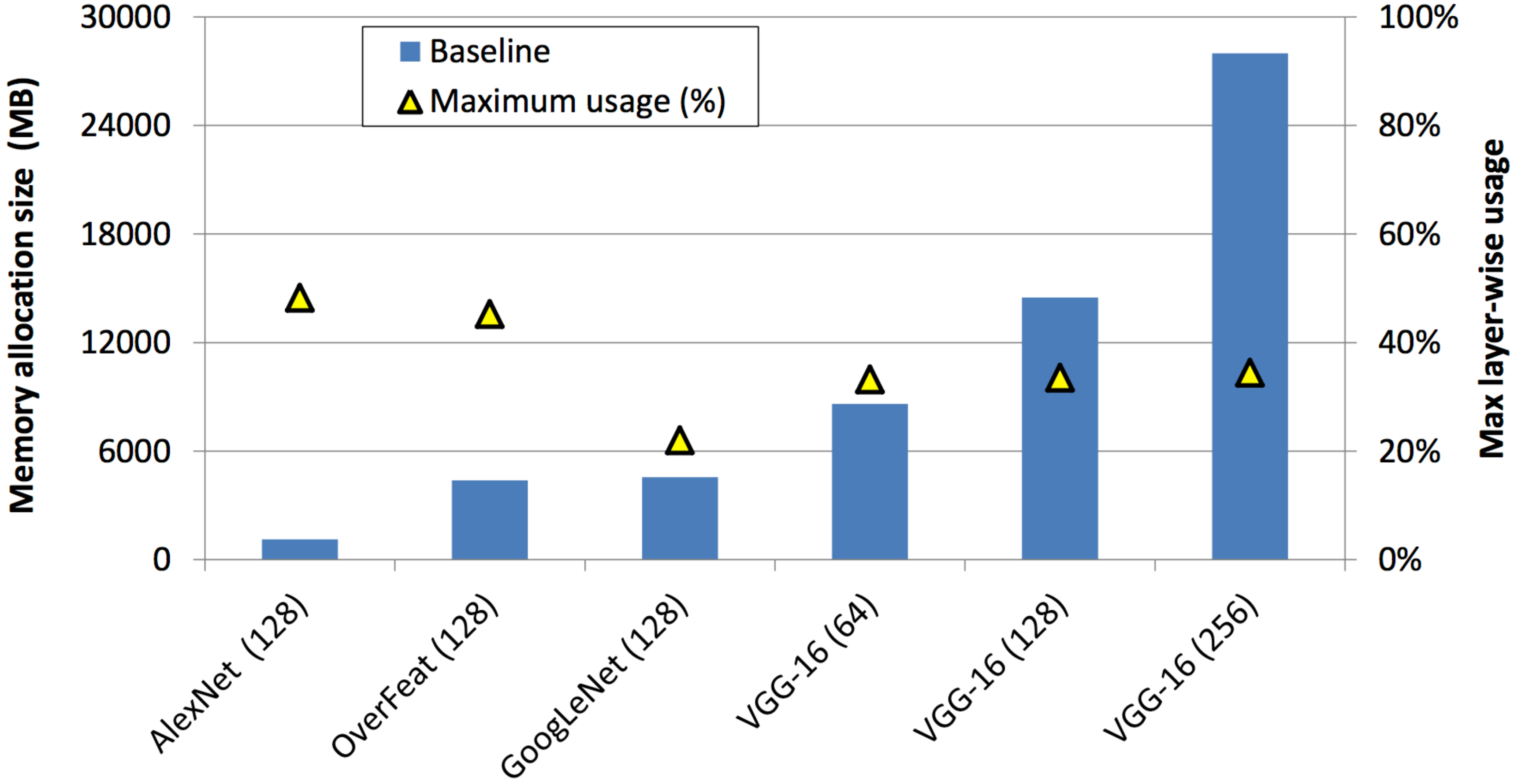


AlexNet (2012)
7 layers

ResNet (2015)
152 layers

Motivation

Challenges: deep networks require large GPU memory



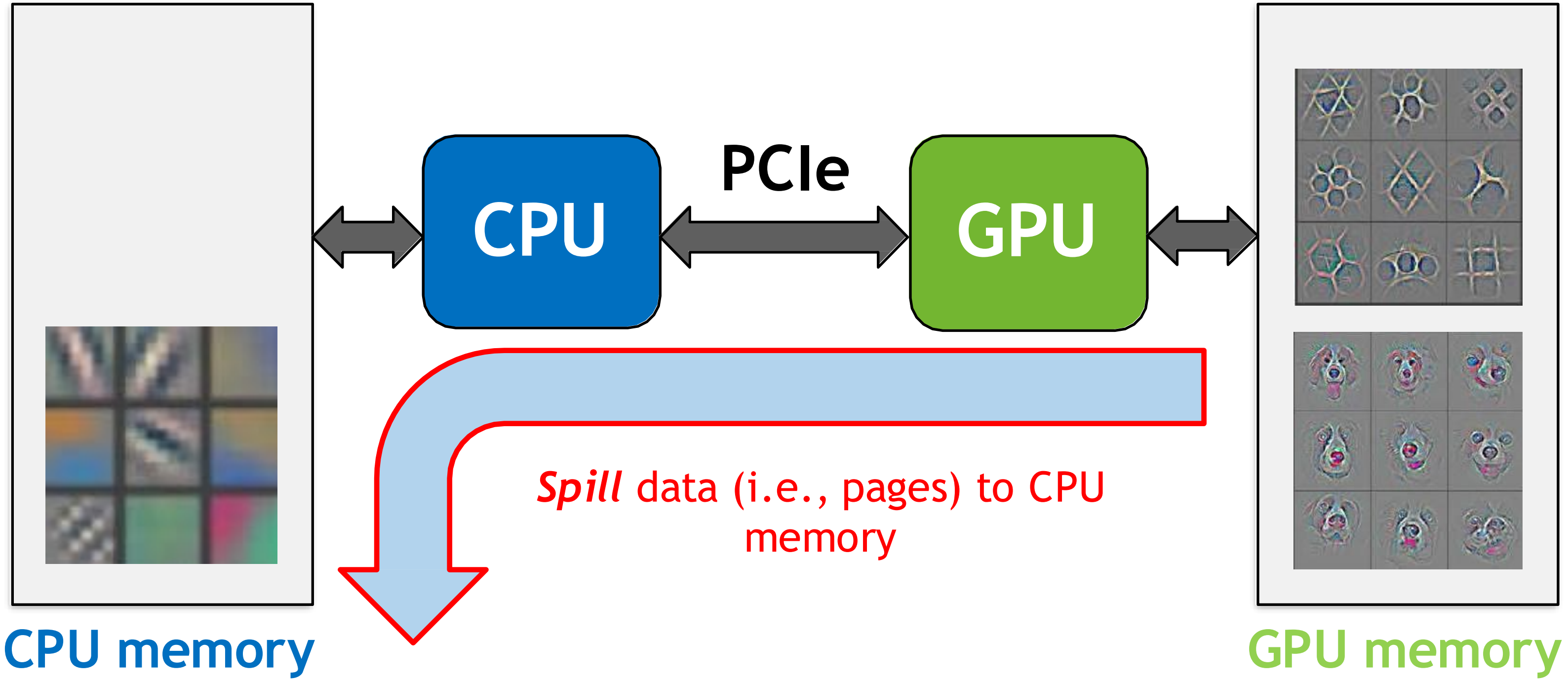
Motivation

Memory capacity bottleneck

- Use less desirable DNN architectures
  - Smaller number of layers
  - Smaller batch sizes
  - Less performant but more memory-efficient convolutional algorithms
- Parallelize the DNN across multiple GPUS
  - Data parallelism
  - Model parallelism
- Network compression
  - Network pruning
  - Quantization
  - Reduced precision

Motivation

Wait … what about CUDA UVM (**U**nified **V**irtual **M**emory) ?



< UVM page-migration from 10000 ft. >

Motivation

Wait … what about CUDA UVM (Unified Virtual Memory) ?

CPU-GPU page-migration in discrete GPU systems (via PCIe)

- 20 ~ 50 μs latency to bring in a single 4 KB page*
- PCIe bw. utilization is around 200 MB/sec (out of the 16 GB/sec under gen3)

Training deep neural networks incur 10s of GBs of memory allocations

- Performance bottlenecked by the throughput of CPU-GPU page-migration

* Zheng et al., “Towards High Performance Paged Memory for GPUs”, HPCA-2016

AGENDA

Why does training DNNs require large memory?

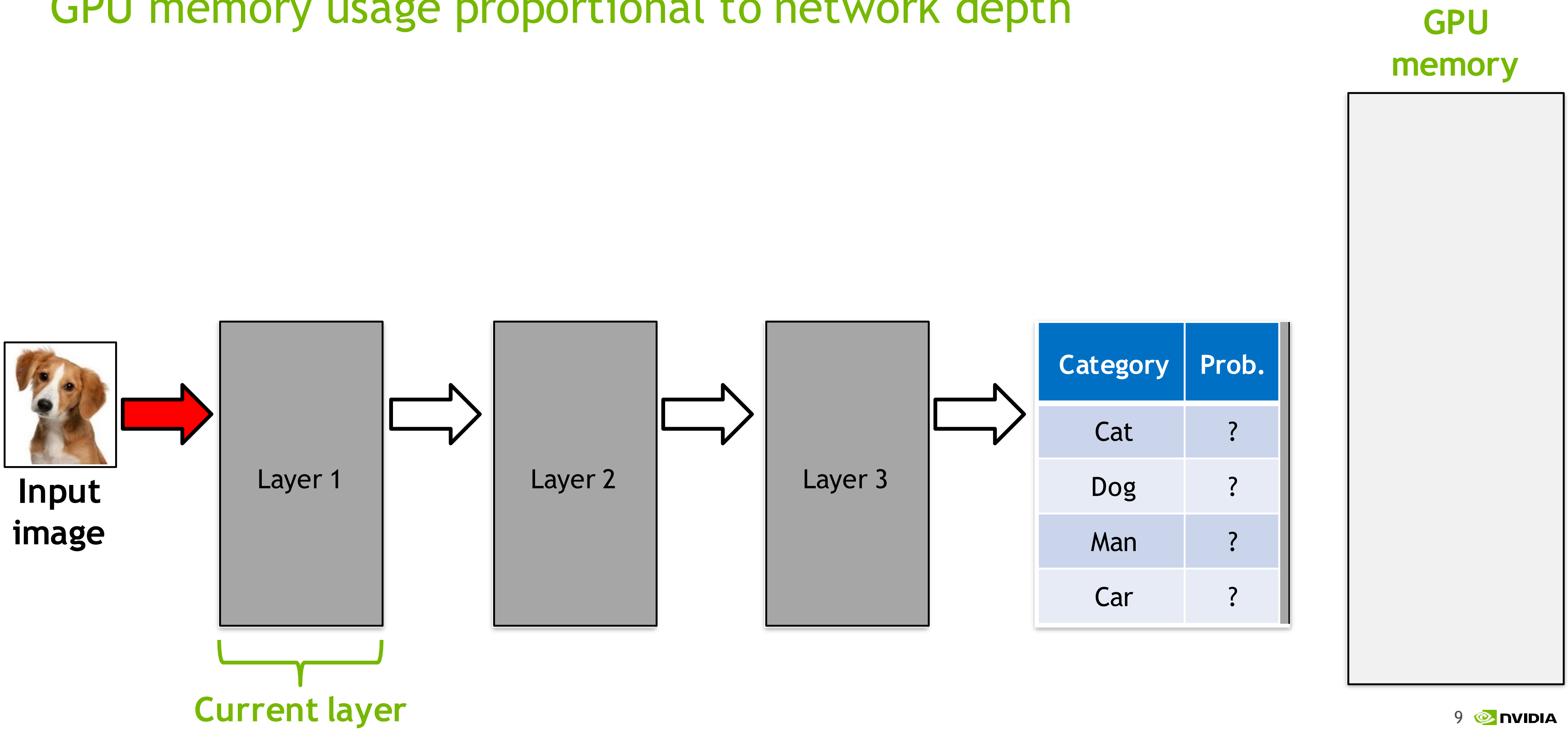
What is our proposed solution to this problem?

How good & effective is our proposal?

Q. Why does training DNNs incur such high GPU memory usage?

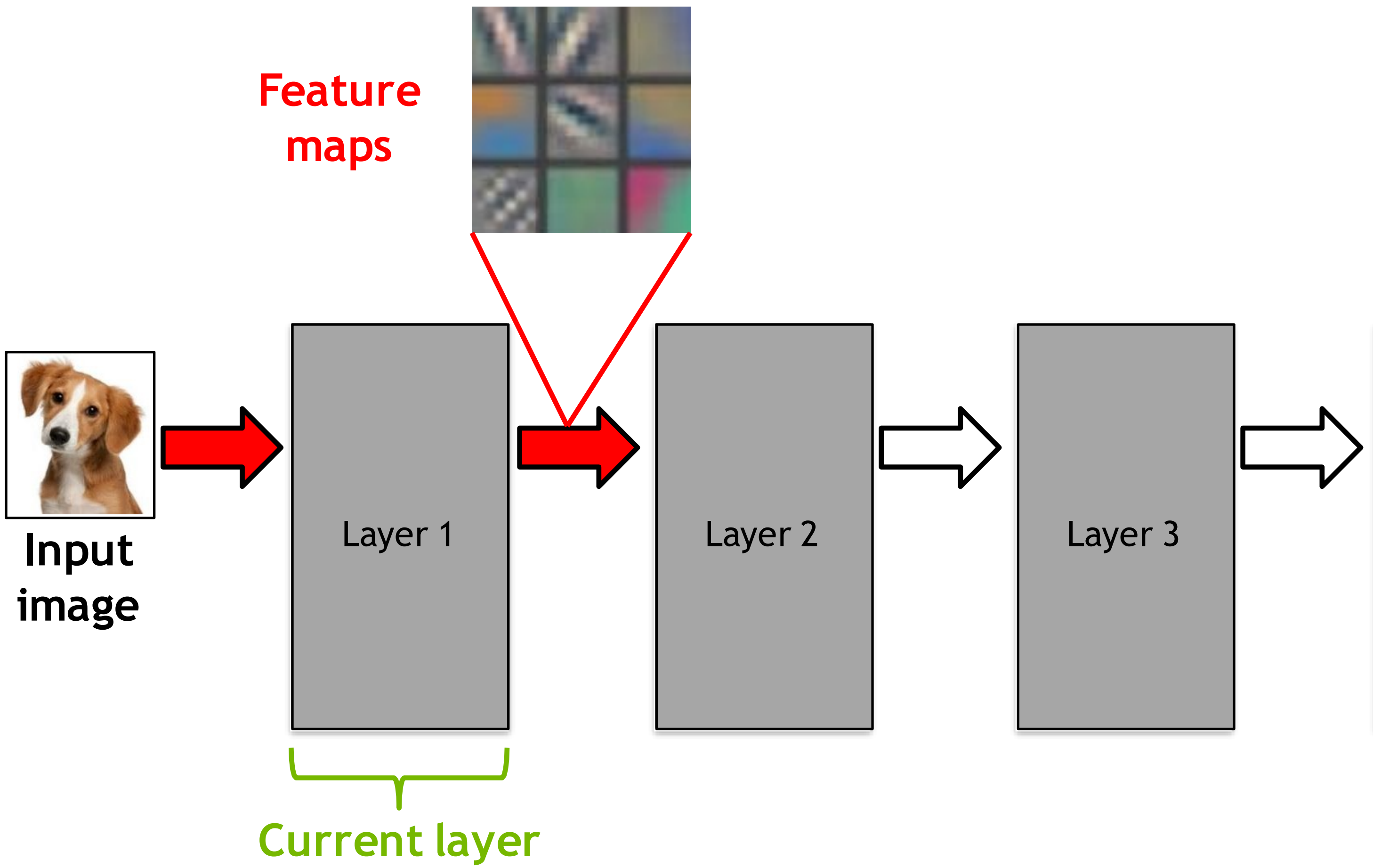
The Problem

GPU memory usage proportional to network depth



The Problem

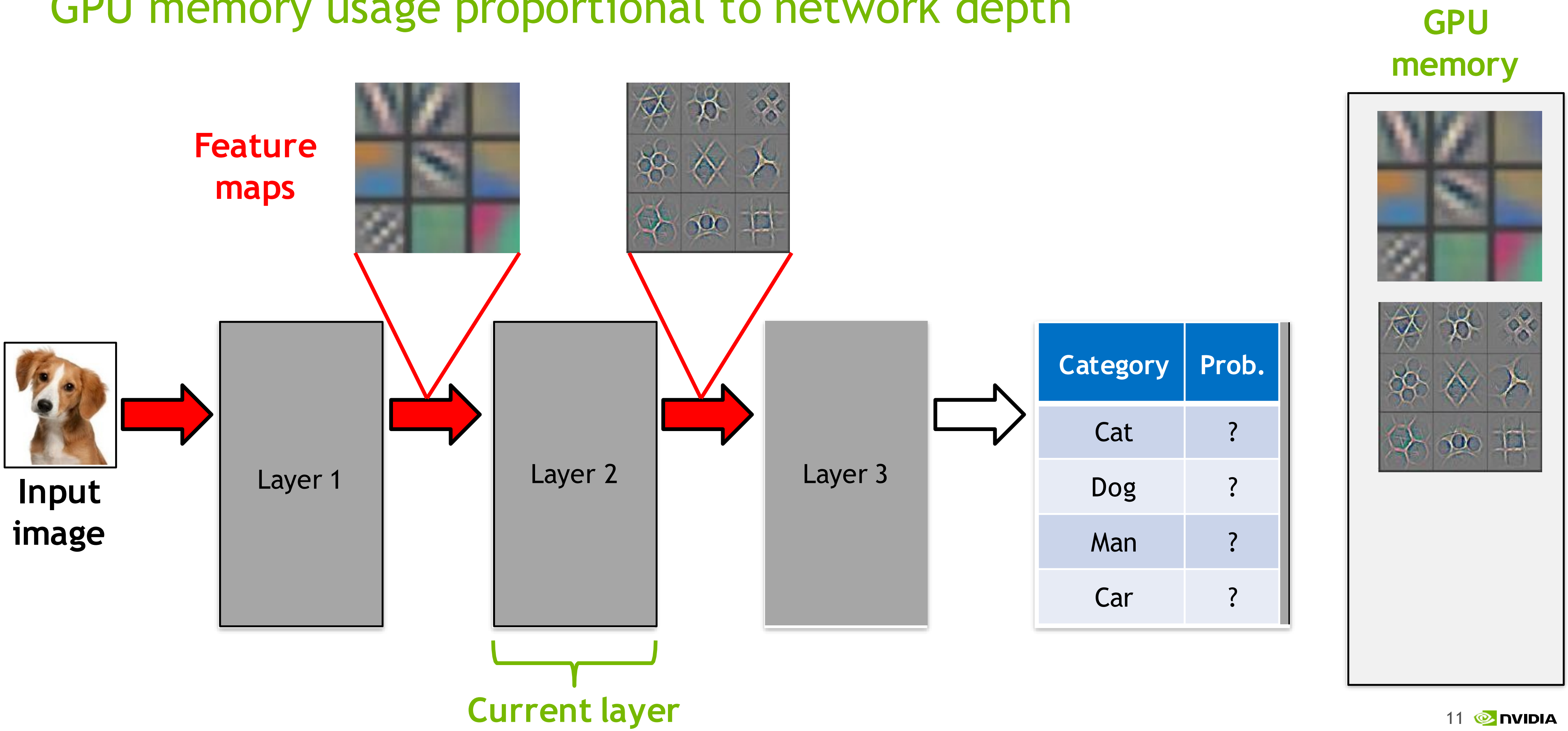
GPU memory usage proportional to network depth



Category	Prob.
Cat	?
Dog	?
Man	?
Car	?

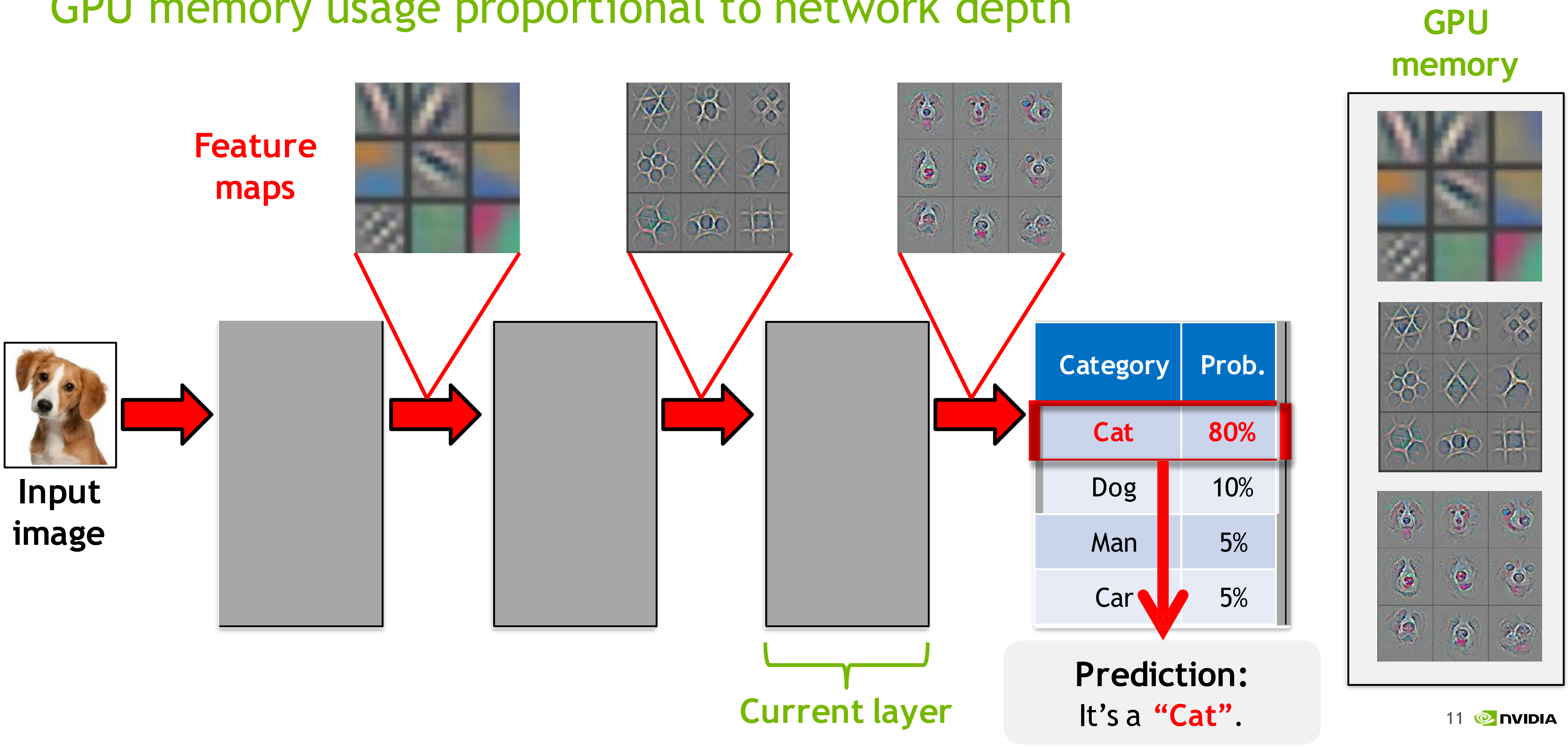
The Problem

GPU memory usage proportional to network depth



The Problem

GPU memory usage proportional to network depth

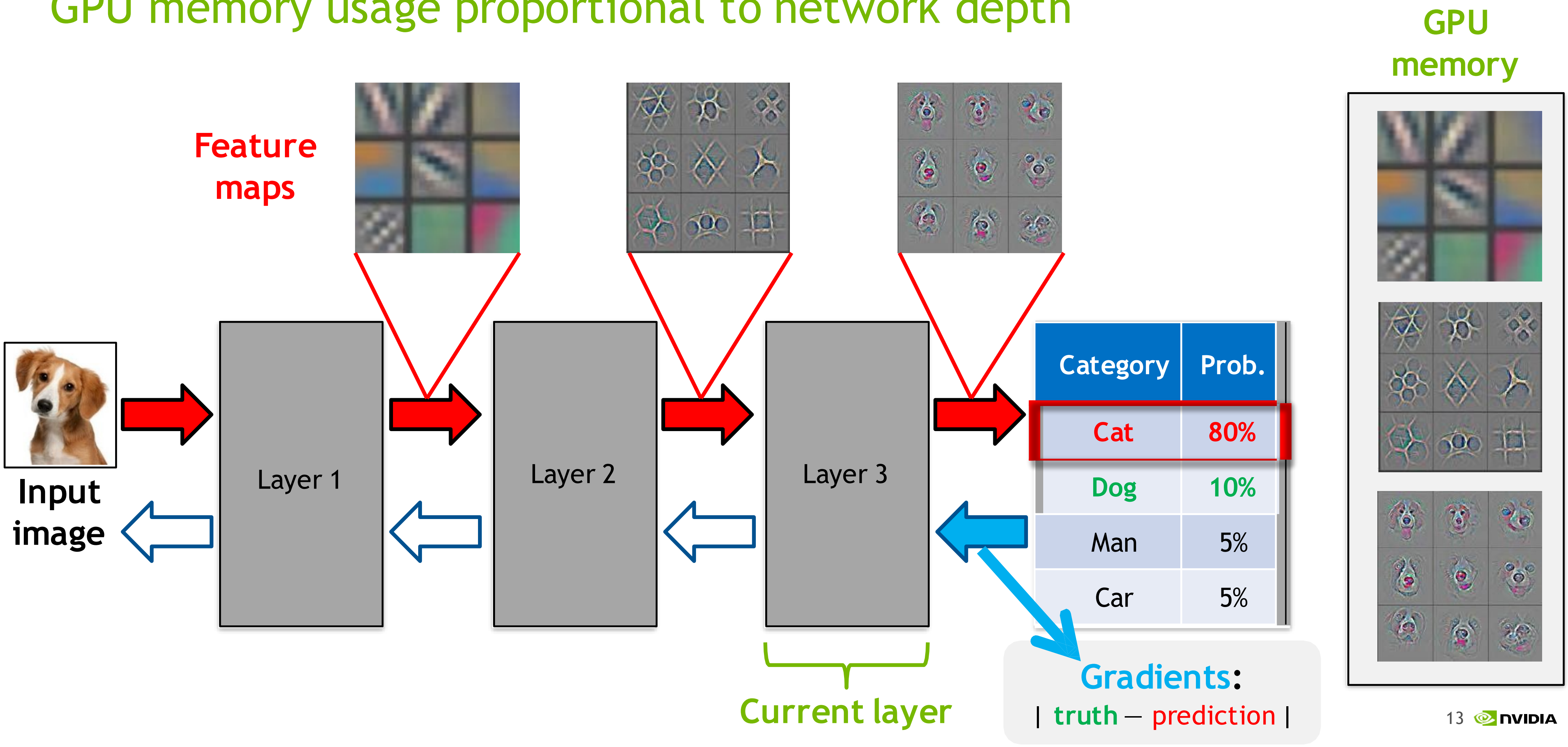


Category	Prob.
Cat	80%
Dog	10%
Man	5%
Car	5%

Now we go backwards: *backward* *propagation*

The Problem

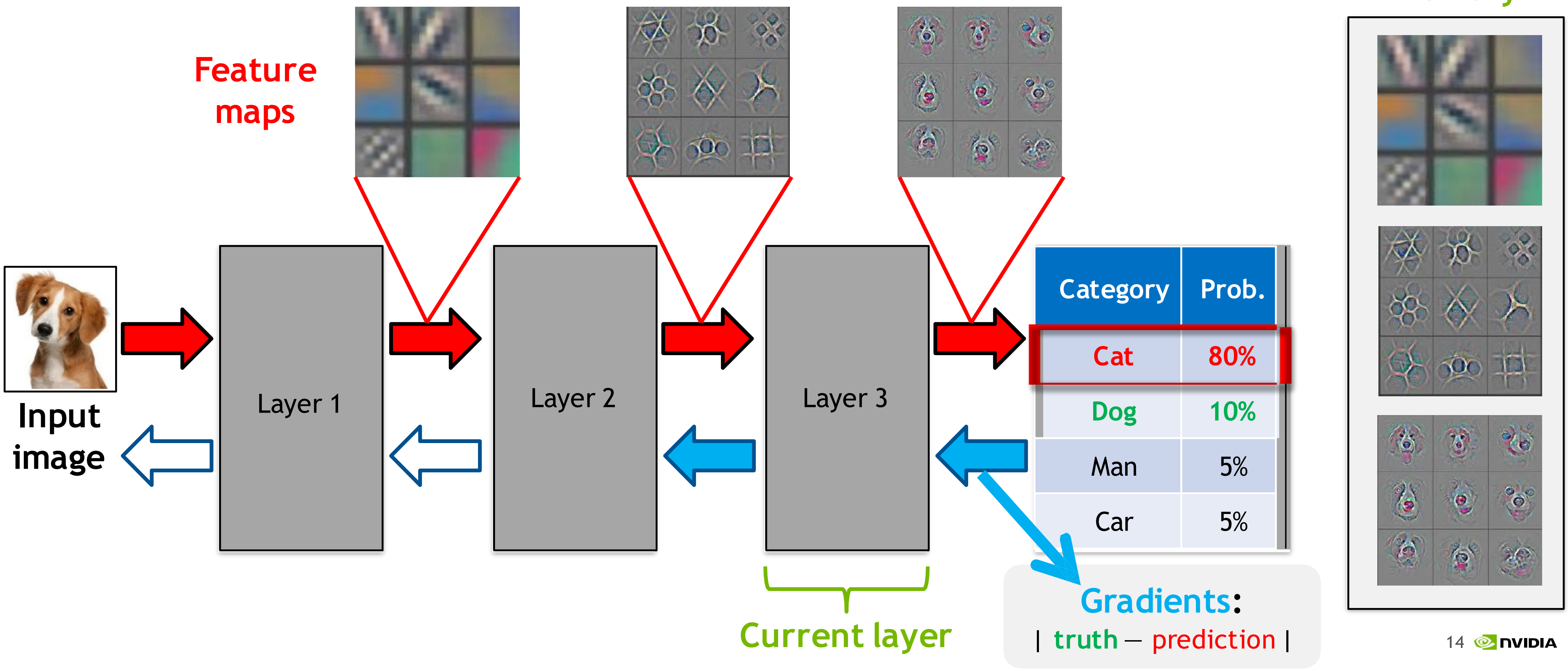
GPU memory usage proportional to network depth



The Problem

GPU memory usage proportional to network depth

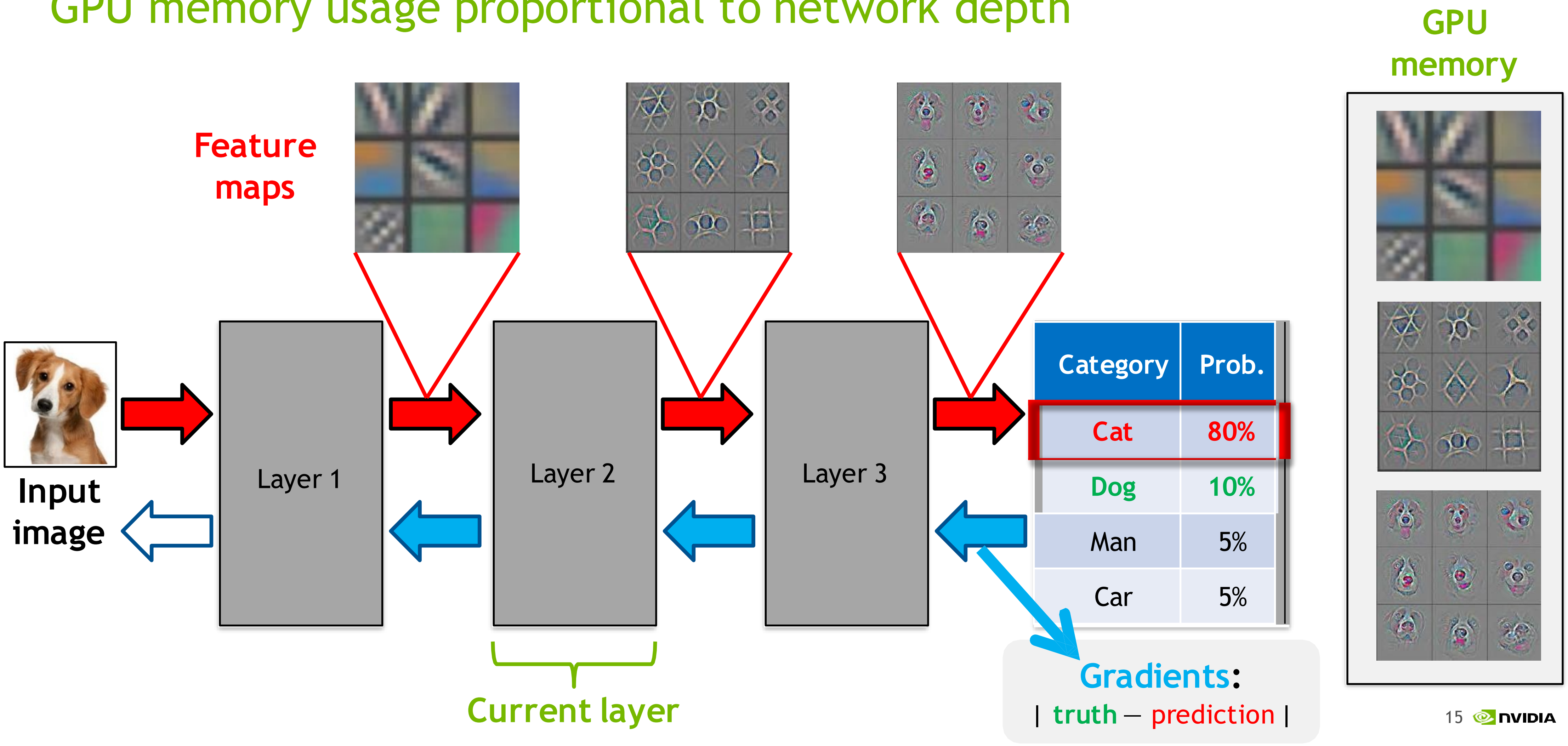
GPU memory



Category	Prob.
Cat	80%
Dog	10%
Man	5%
Car	5%

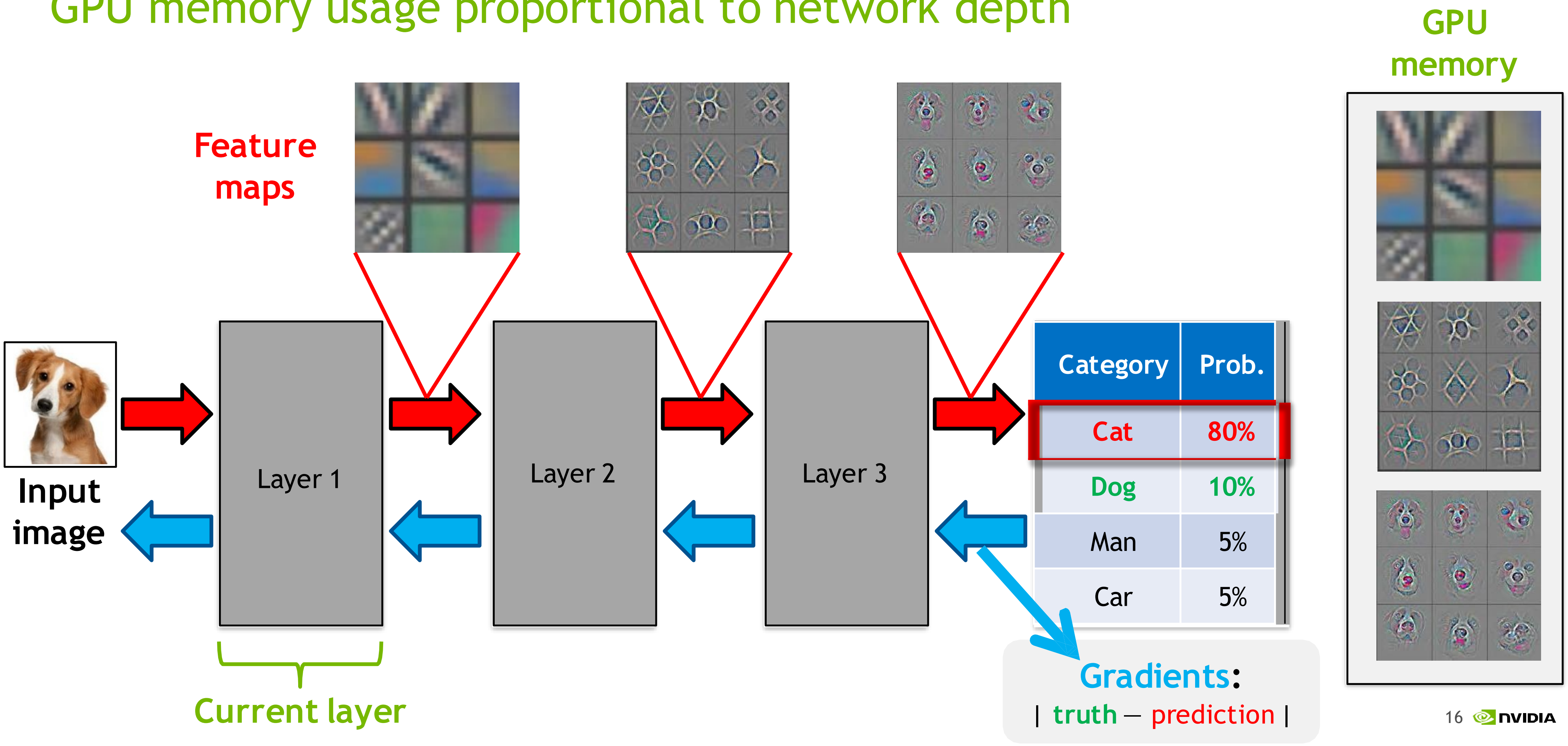
The Problem

GPU memory usage proportional to network depth



The Problem

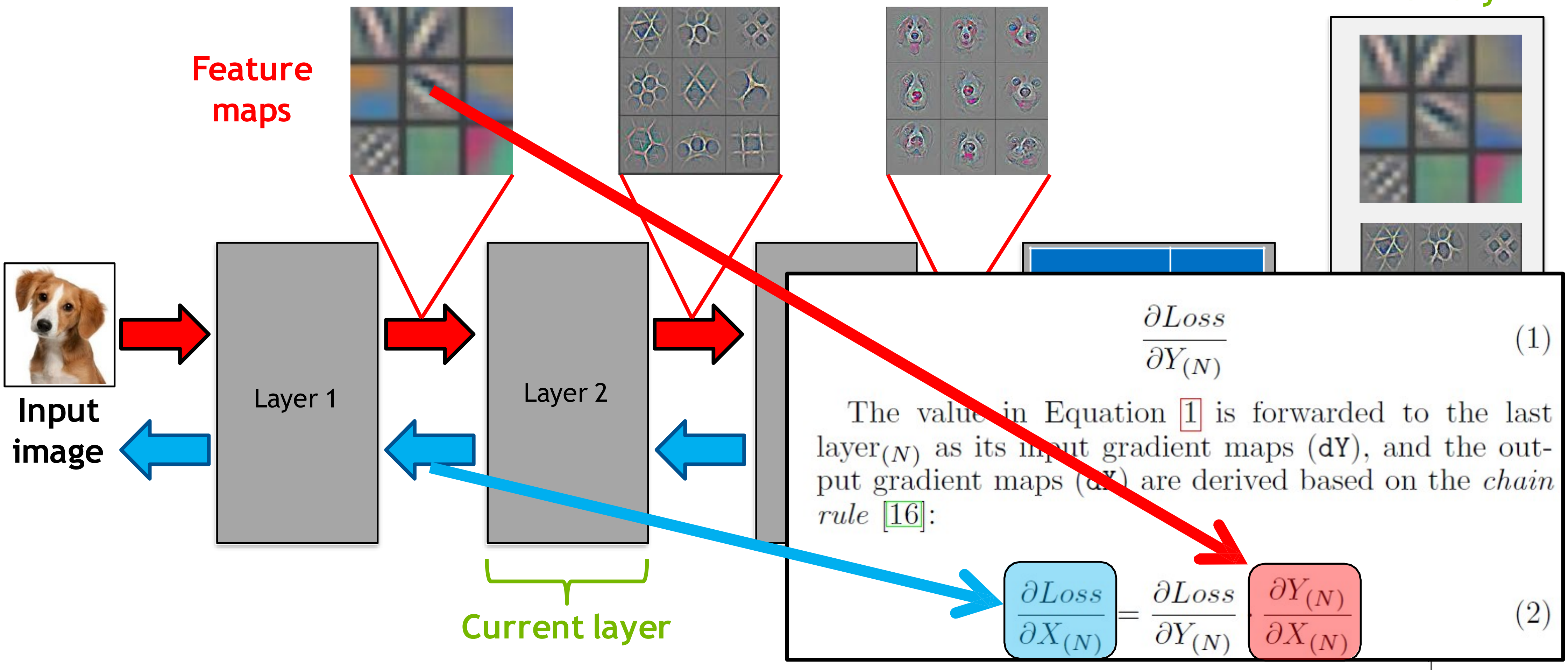
GPU memory usage proportional to network depth



Now we go backwards: *backward* ***propagation***

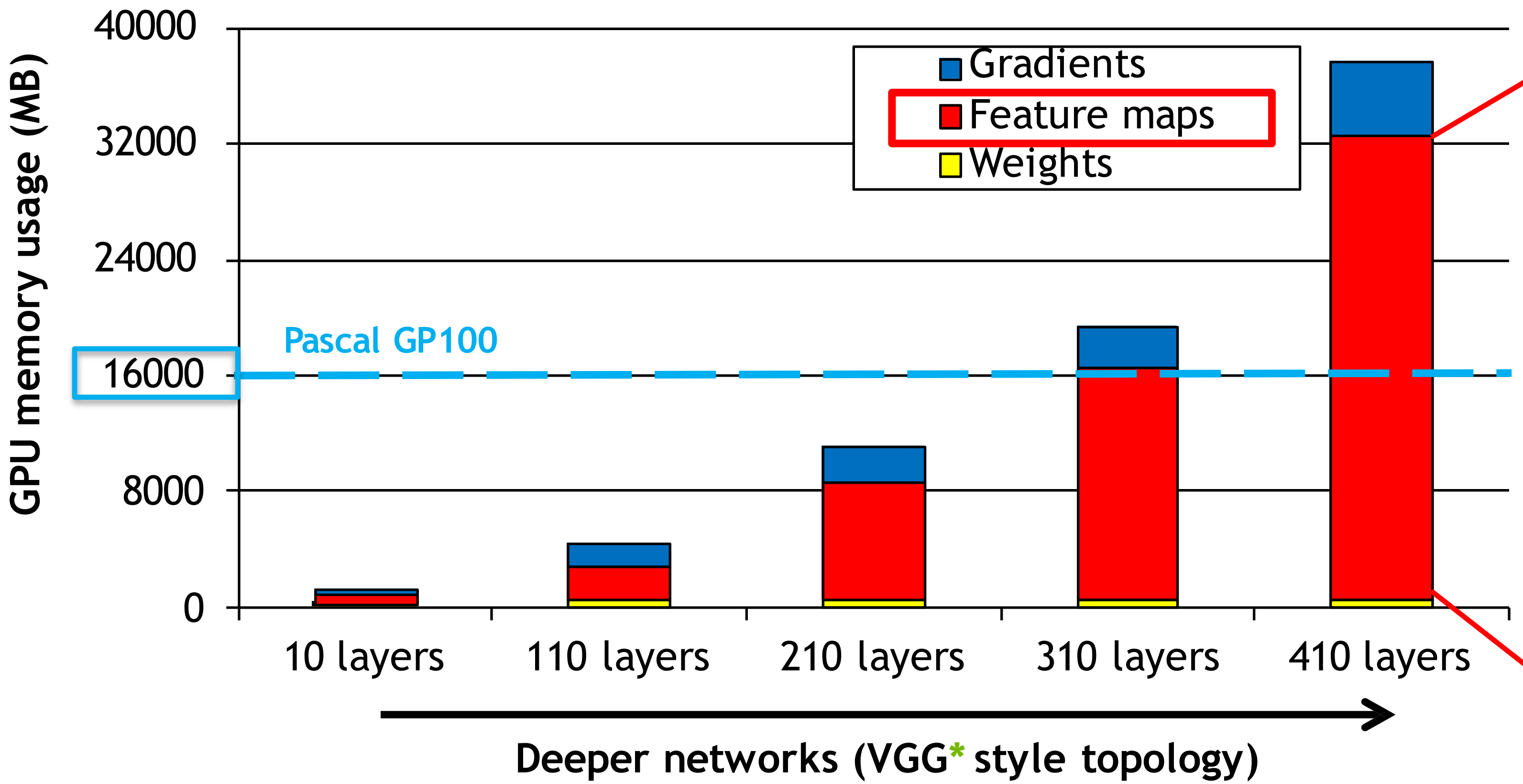
The Problem

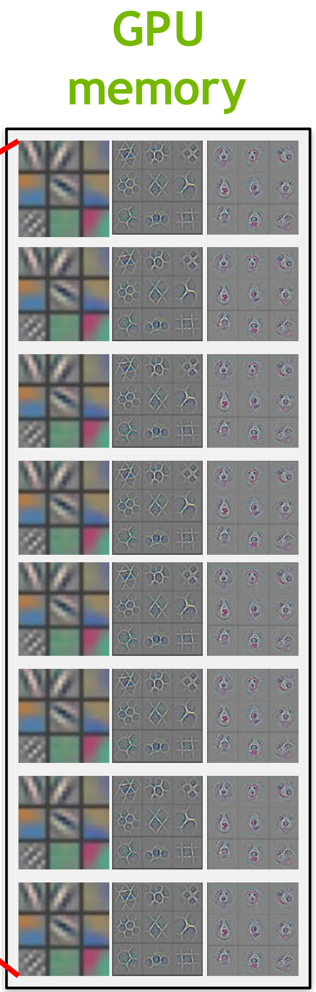
GPU memory usage proportional to network depth



The Problem

GPU memory usage proportional to network depth

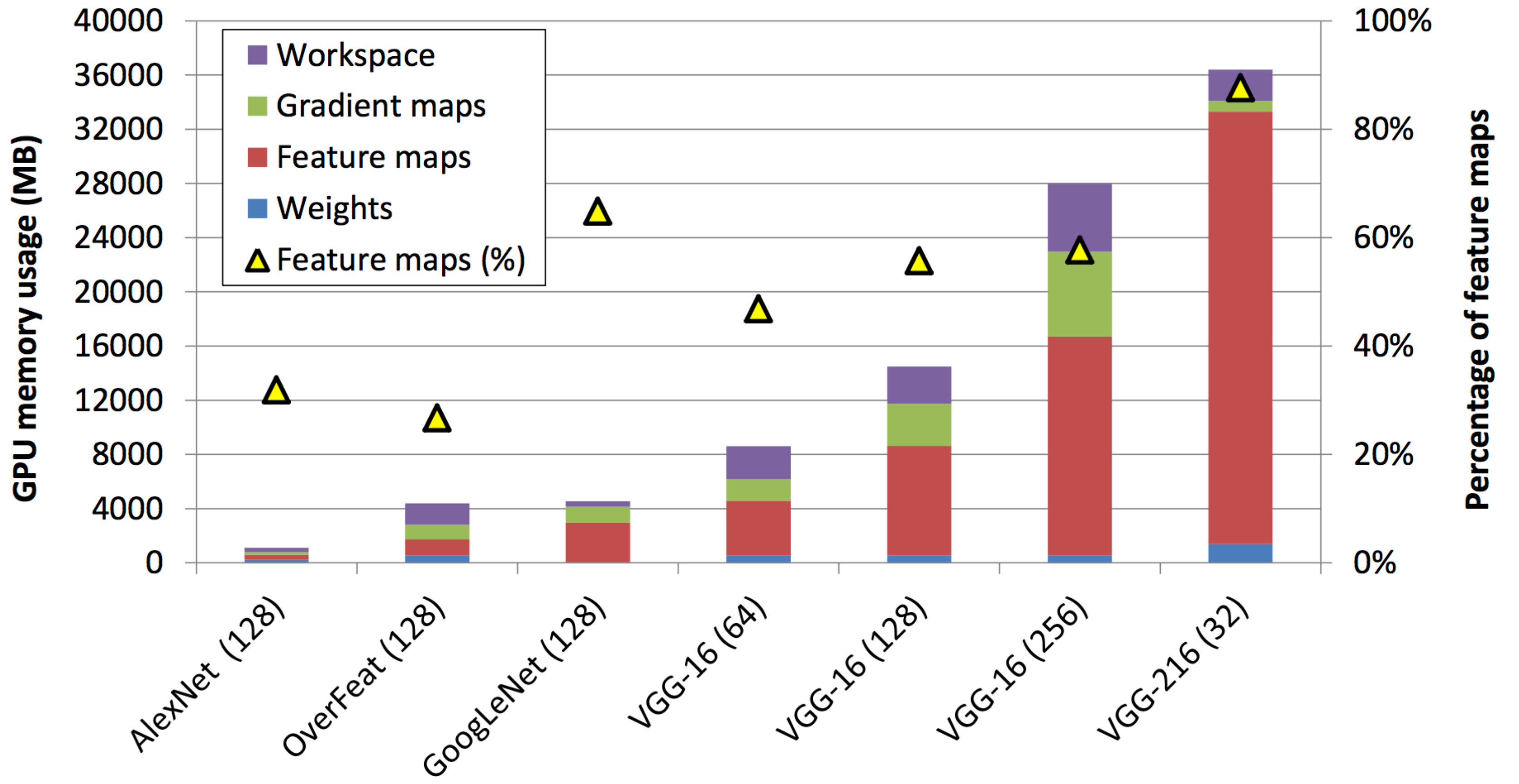




* Simonyan and Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition", ICLR-2015

The Problem

GPU memory usage proportional to network depth



NVIDIA

Our solution: virtualized DNN (vDNN)

Virtualized DNN (vDNN)

What is it?

CPU-side runtime memory manager tailored for DNNs

Functionality:

- *Virtualize* DNN memory usage across "***both***" CPU and GPU memory
- GPU memory acts as a fast ***cache*** for current layer's memory usage

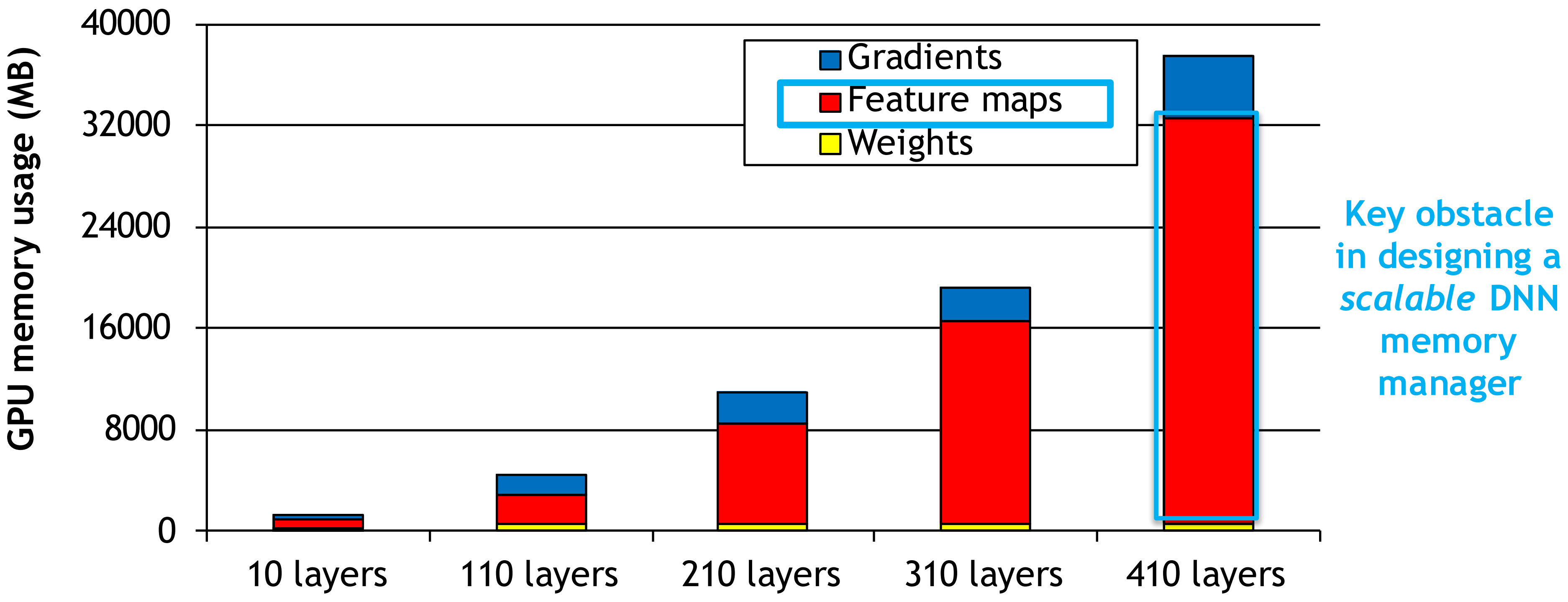
Virtualized DNN (vDNN)

Design principle

Exploits the following observations for performance optimizations

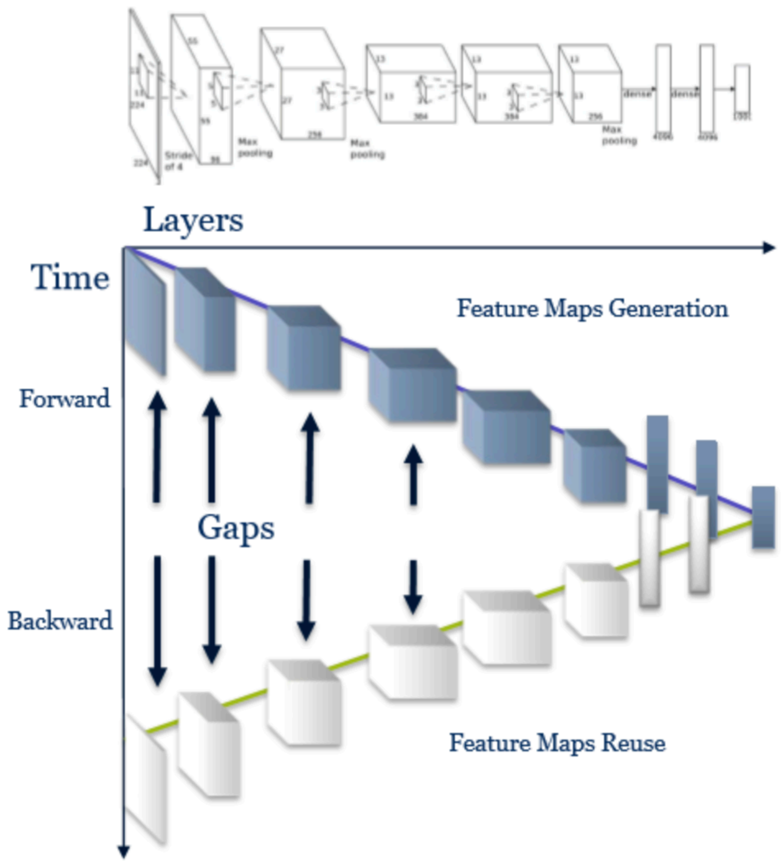
Virtualized DNN (vDNN)

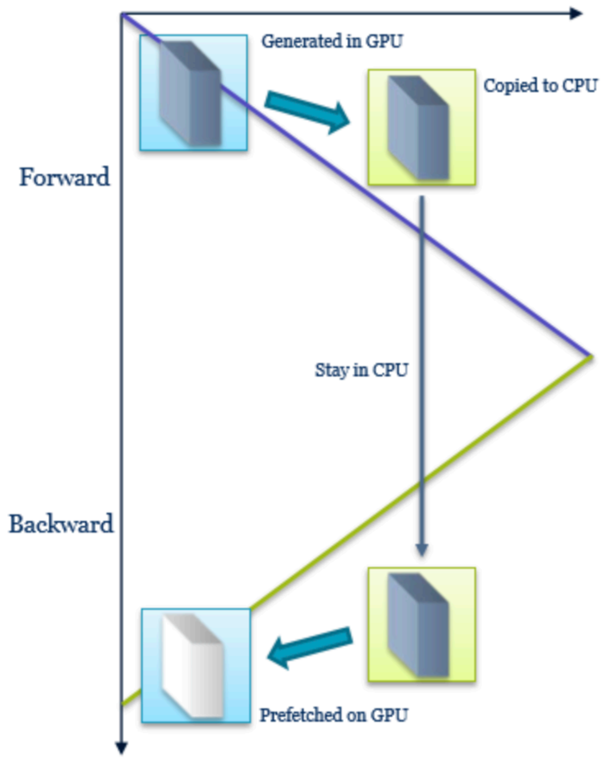
Key observations



Observation #1: feature maps dominate memory usage

Virtualized DNN (vDNN)

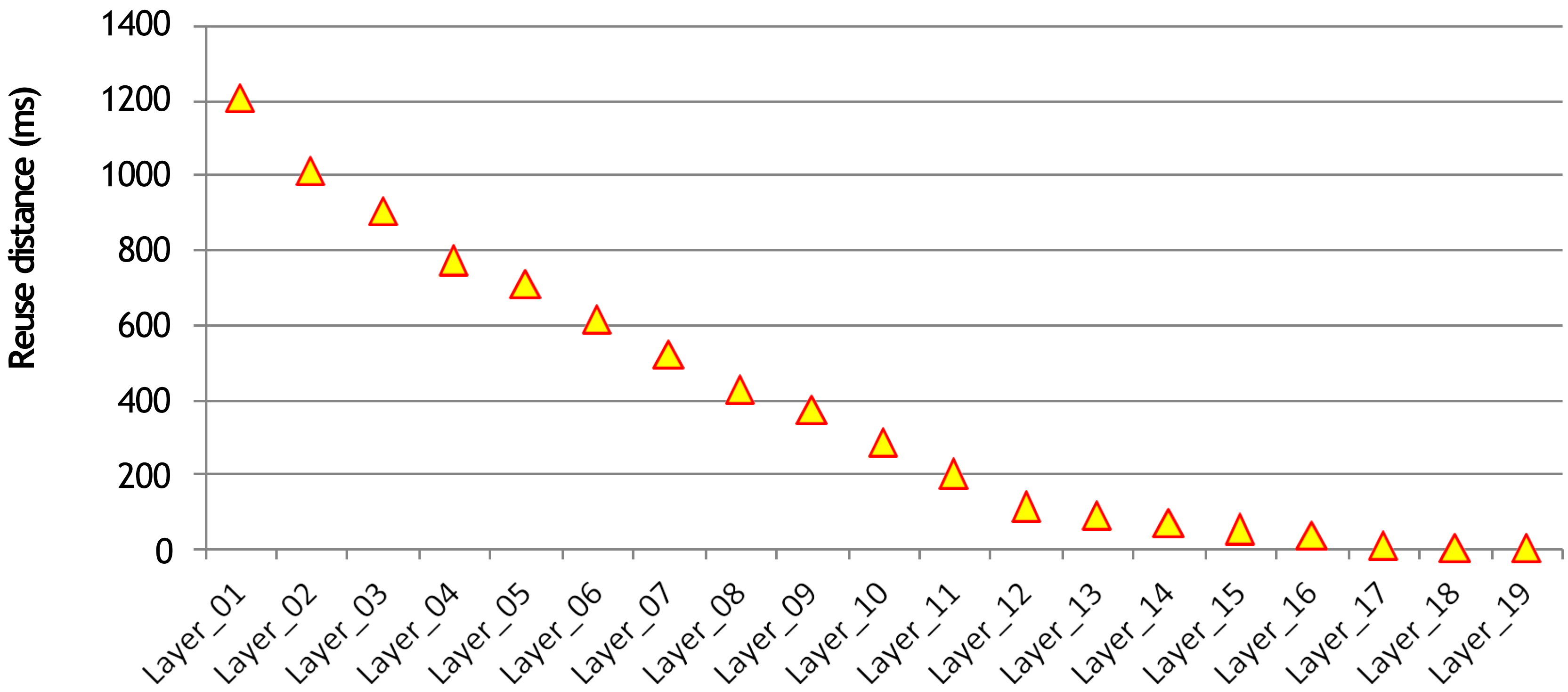




Observation #2: long reuse distance of feature maps

Virtualized DNN (vDNN)

Key observations

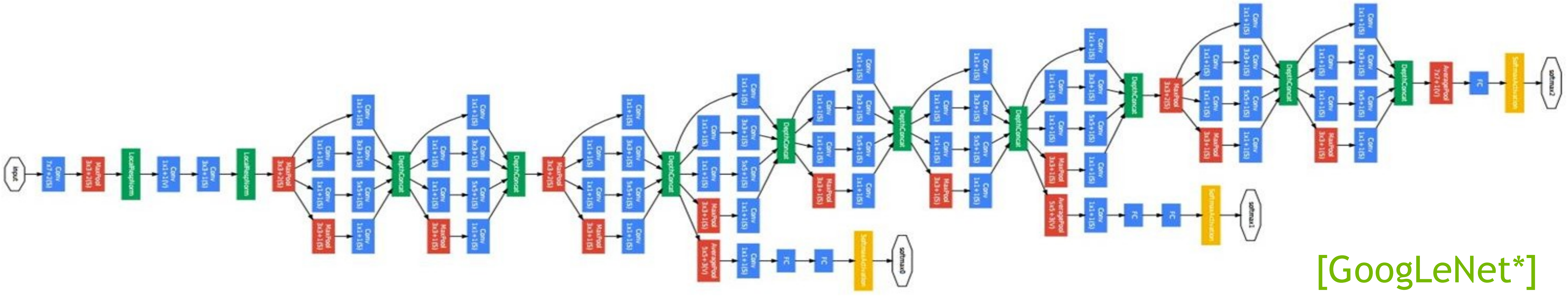


Observation #2: long reuse distance of feature maps

Virtualized DNN (vDNN)

Key observations

Exploits the following observations for performance optimizations



[GoogLeNet*]

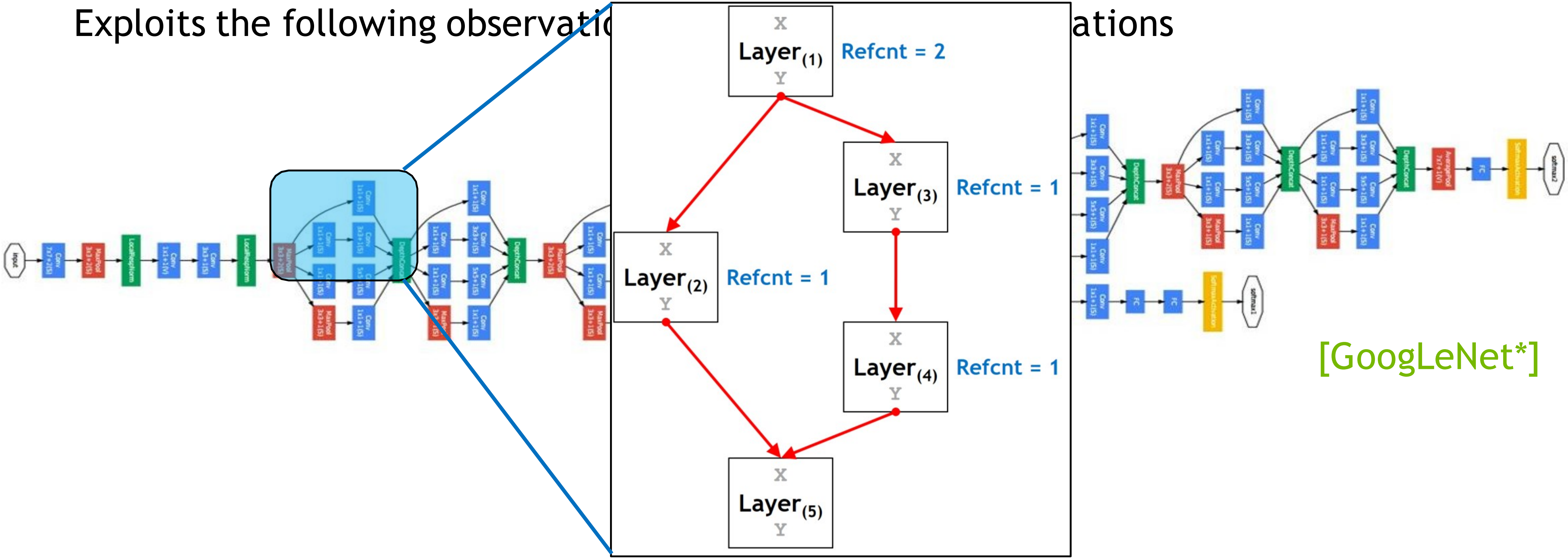
Observation #3: DNN computation dataflow = DAG (direct acyclic graph)

* Szegedy et al., “Going Deeper With Convolutions”, CVPR-2015

Virtualized DNN (vDNN)

Key observations

Exploits the following observati ations



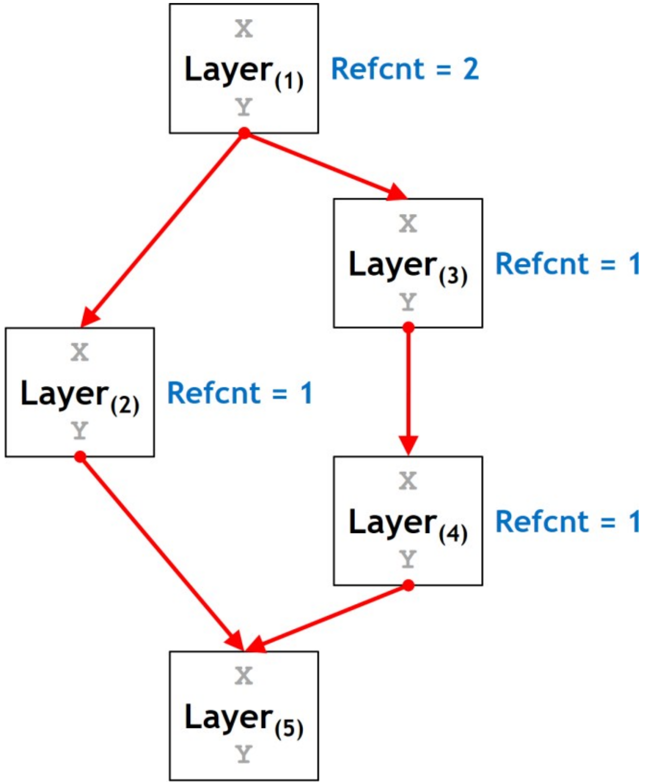
[GoogLeNet*]

Observation #3: DNN computation dataflow = DAG (direct acyclic graph)

* Szegedy et al., "Going Deeper With Convolutions", CVPR-2015

Virtualized DNN (vDNN)

Key observations

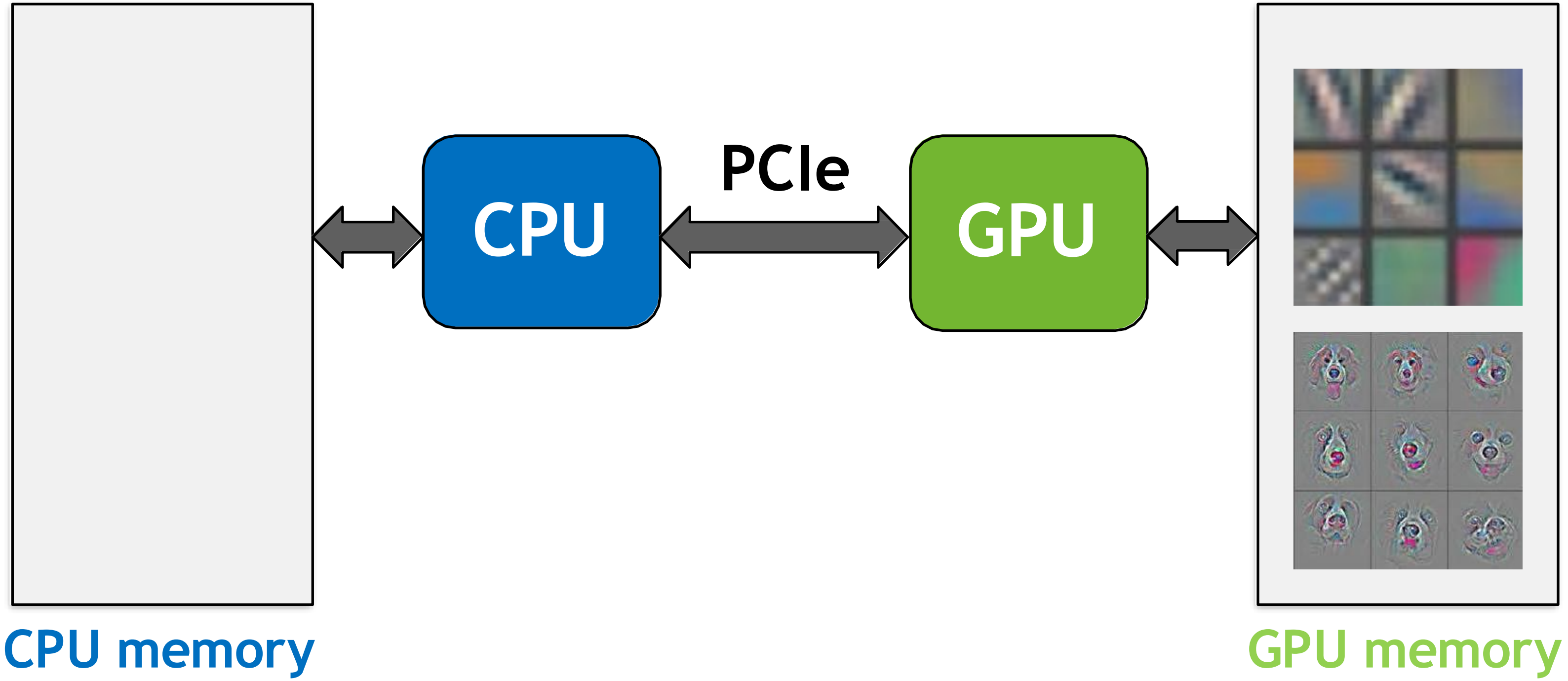


Refcnt: number of consumer layers of the current layer's output feature maps

Key idea) vDNN leverages the data dependencies of the feature maps revealed through the DAG to schedule intelligent CPU offload/prefetch operations.

Virtualized DNN (vDNN)

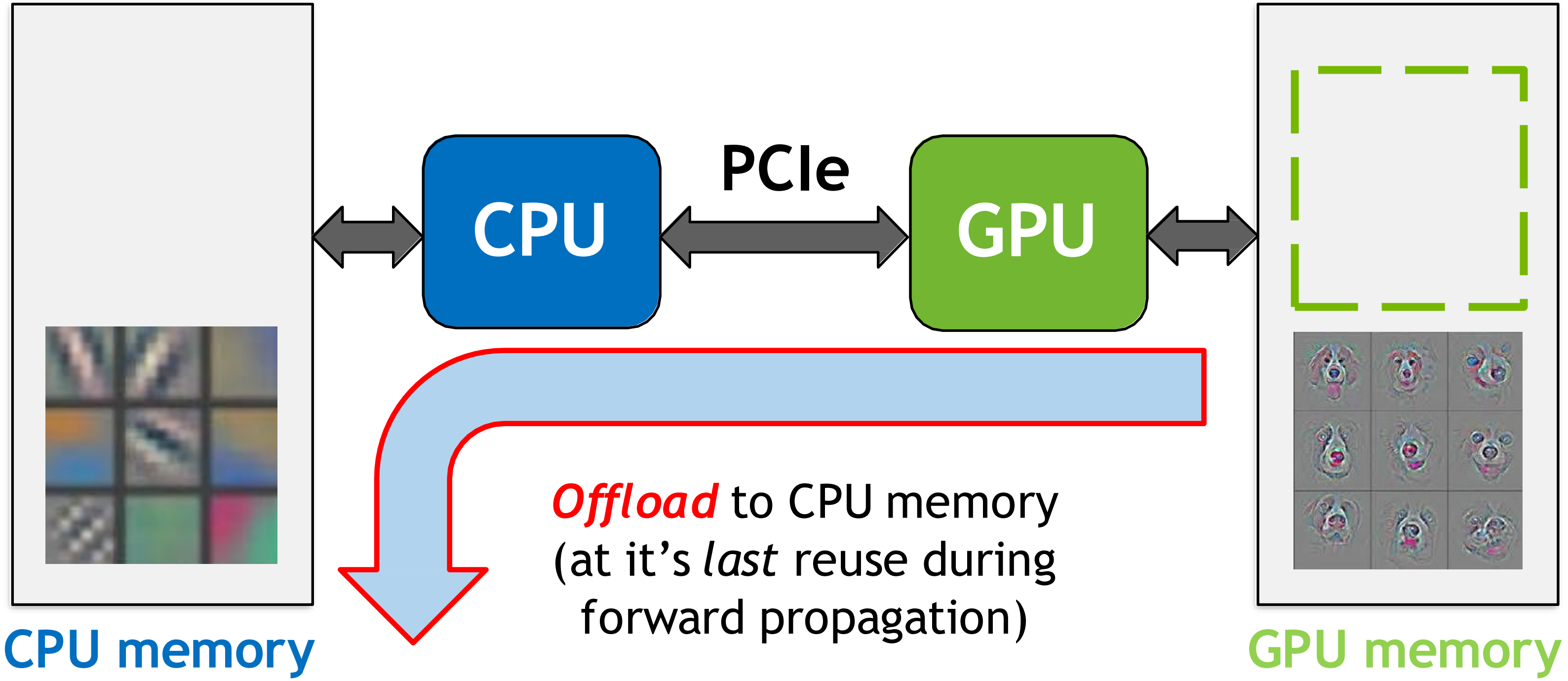
Offloading feature maps to CPU memory



Virtualized DNN (vDNN)

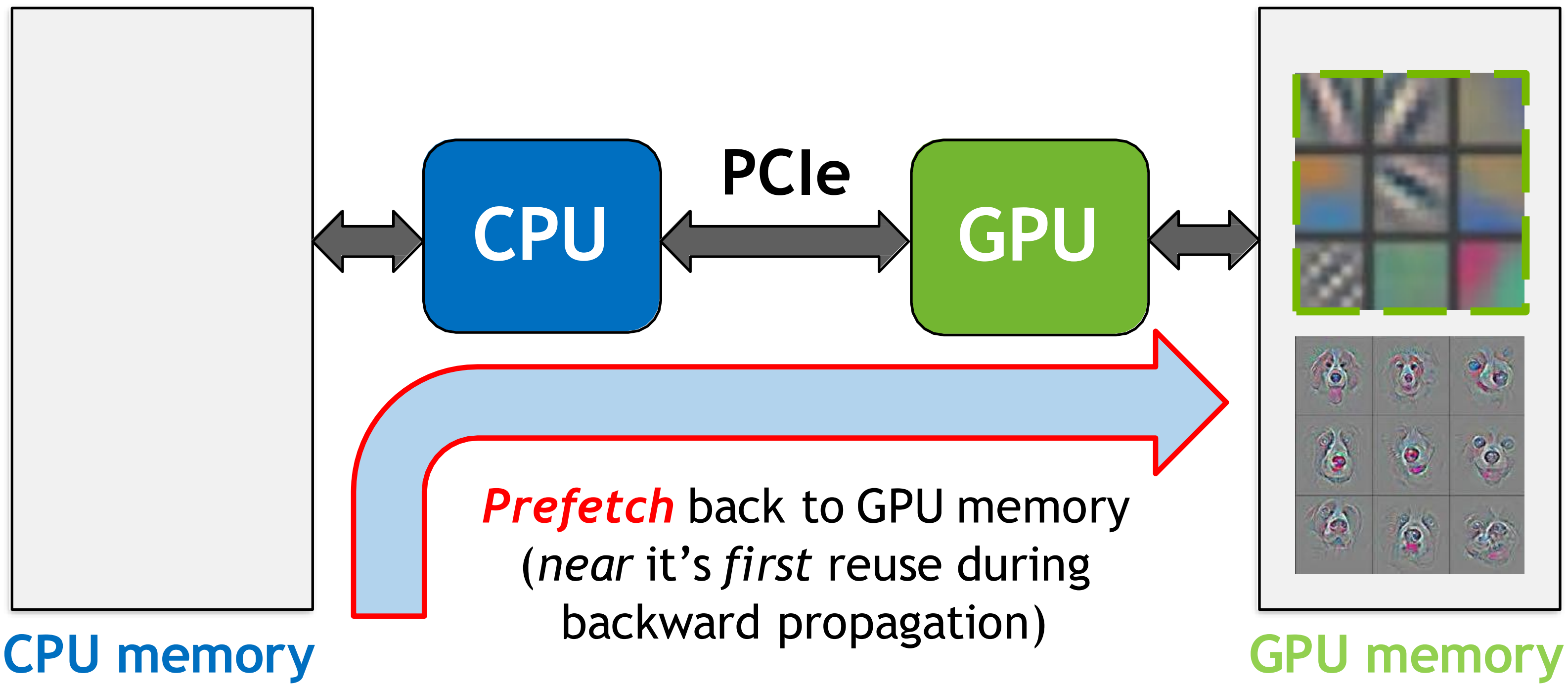
Offloading feature maps to CPU memory

Free up space for future allocations

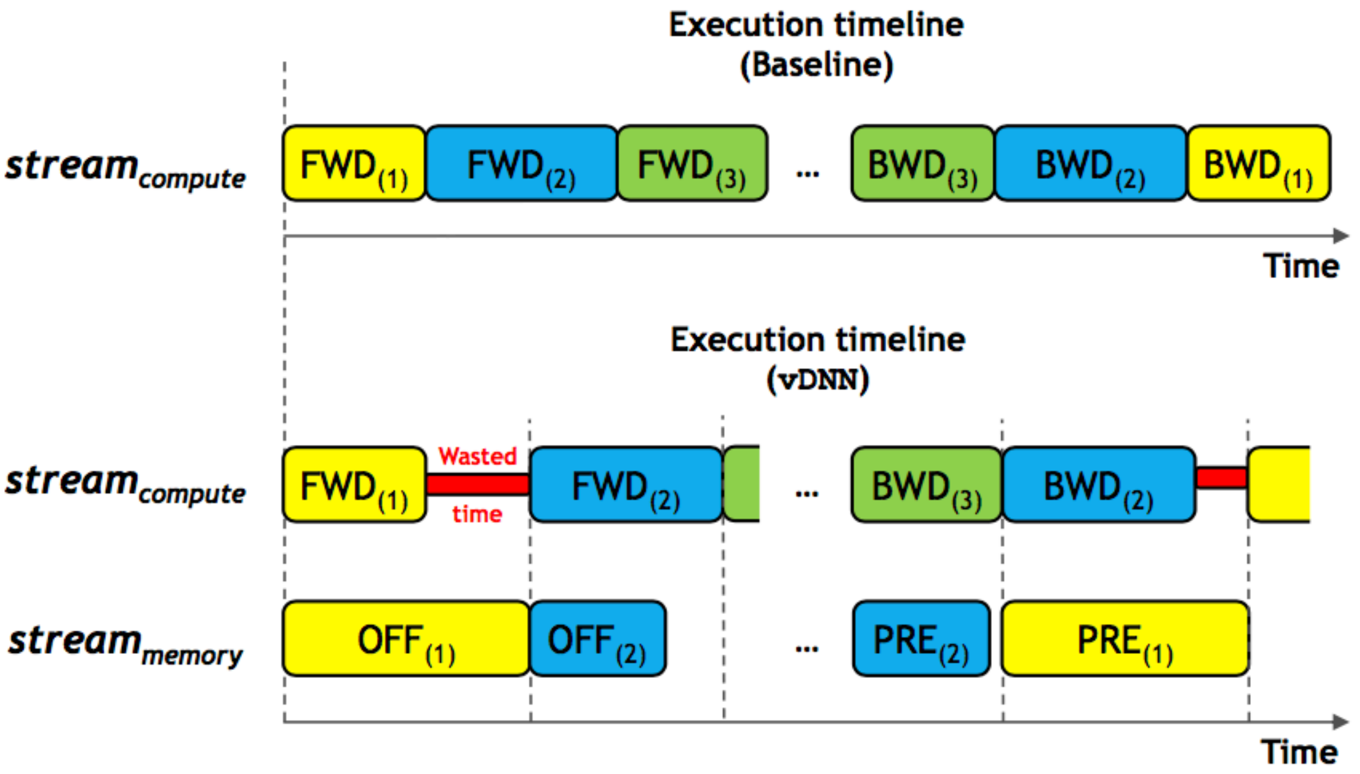


Virtualized DNN (vDNN)

Prefetching feature maps back into GPU memory



vDNN Memory Transfer Policy



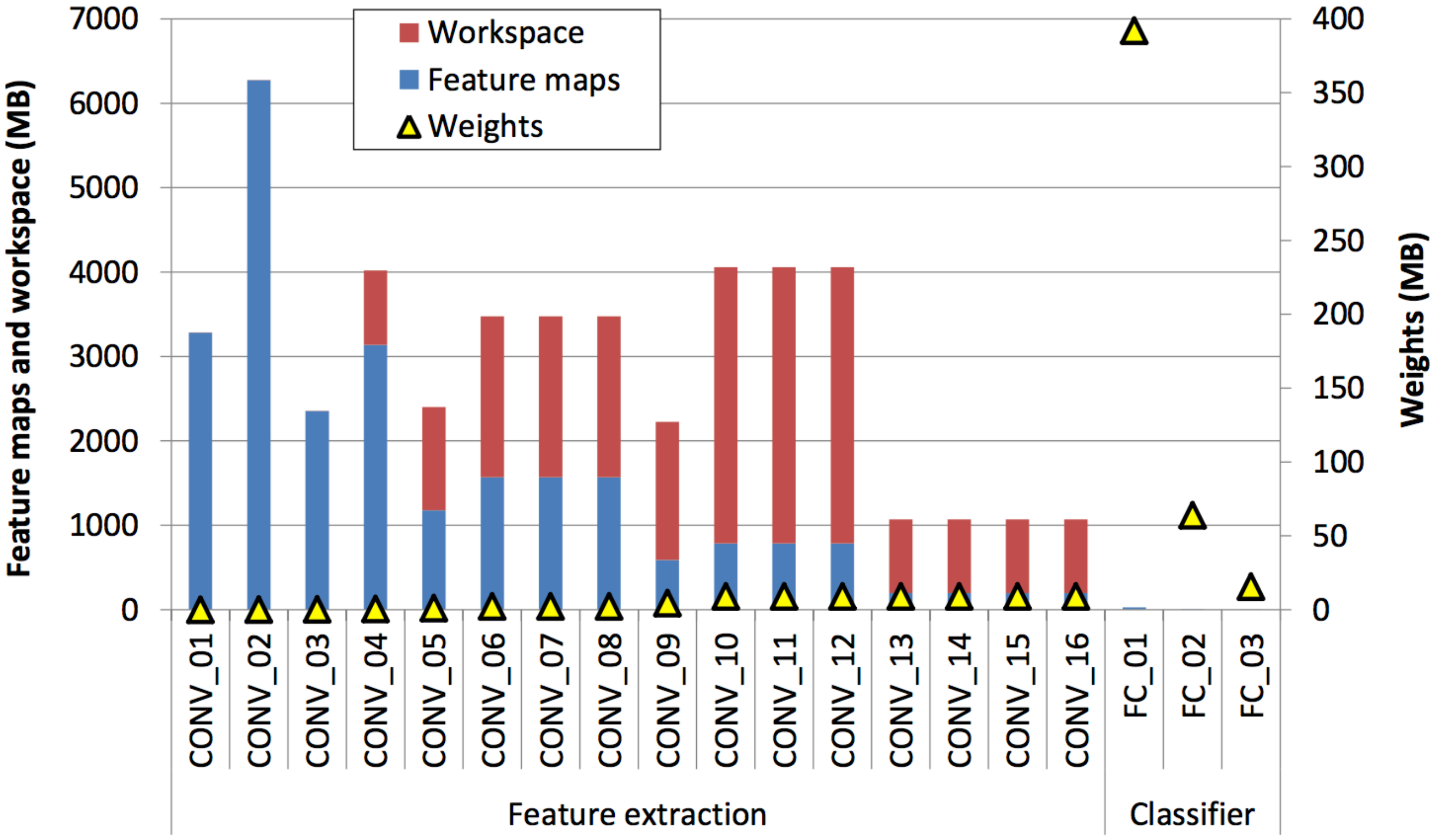
NVIDIA

vDNN Memory Transfer Policy

Different Convolution Algorithms in cuDNN 4.0

- IMPLICIT_GEMM
- PRECOMP_GEMM
- GEMM
- DIRECT
- FFT
- FFT_TILING
- WINOGRAD
- WINOGRAD_NONFUSED

vDNN Memory Transfer Policy

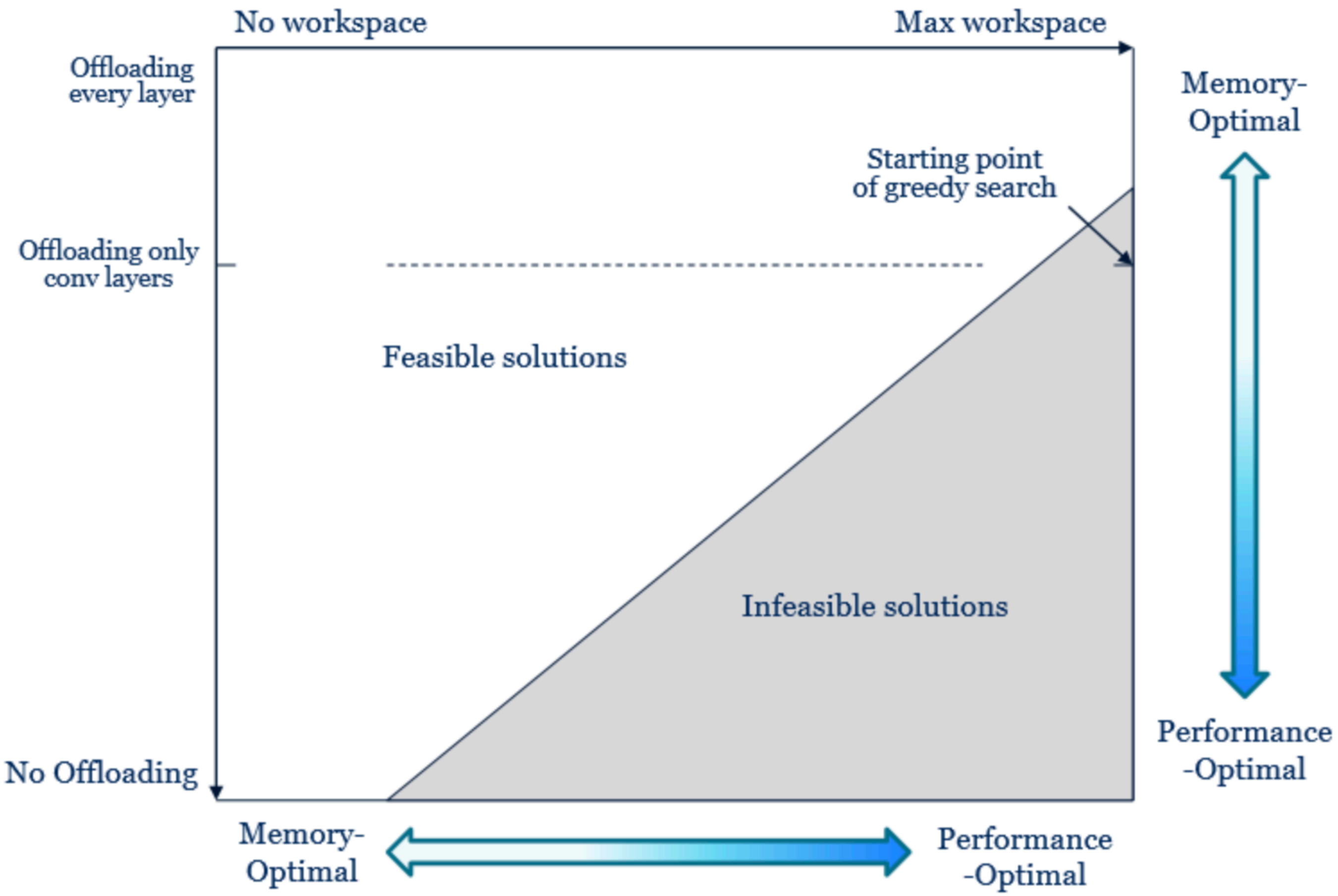


NVIDIA

vDNN Memory Transfer Policy

Tradeoff: Time & Space

Whether a layer should be offloaded/prefetched or not, and what convolution algorithm should we choose.



NVIDIA

vDNN Memory Transfer Policy

- **Static vDNN**
  - vDNN-all + memory-optimal-conv
  - vDNN-all + performance-optimal-conv
  - vDNN-conv + memory-optimal-conv
  - vDNN-conv + performance-optimal-conv

- **Dynamic vDNN**
  1. Started from vDNN-all + memory-optimal-conv
  2. If passed, then no-offload + performance-optimal-conv
  3. If failed, then
     1. vDNN-conv + performance-optimal-conv
     2. vDNN-all + performance-optimal-conv
  4. If failed, then tries to locally reduce a layer's memory usage, greedy search for a global optimum state in terms of trainability and performance.
     1. vDNN-conv + greedy-optimal-conv
     2. vDNN-all + greedy-optimal-conv

How good is vDNN?

Evaluation Methodology

Compute node configuration

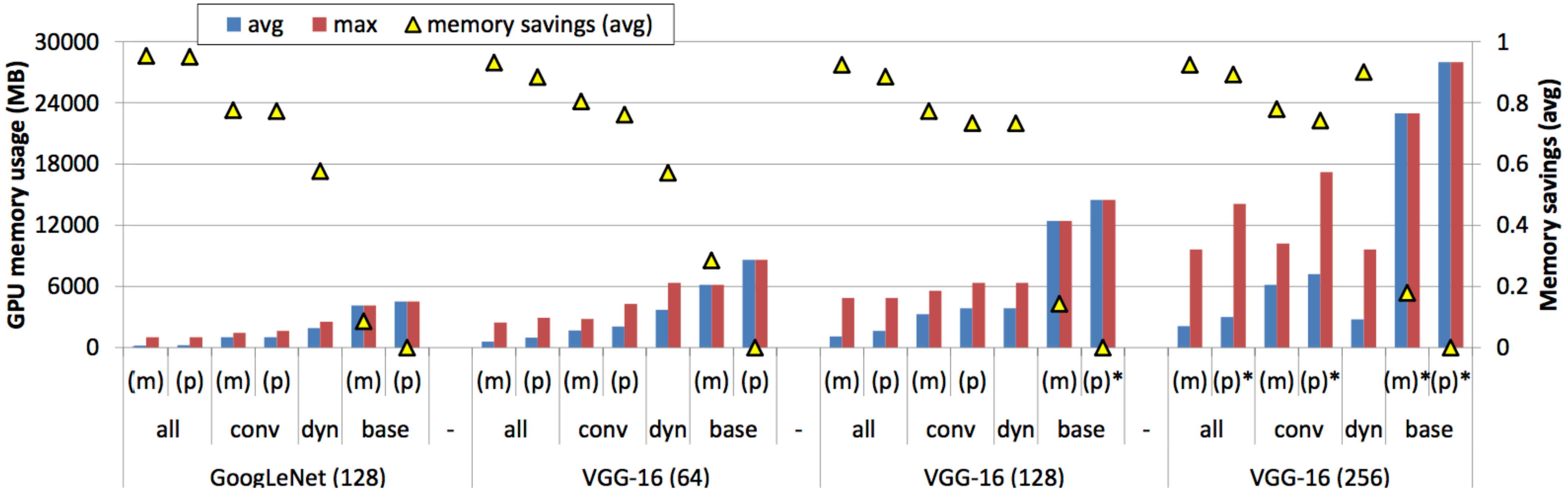
CPU: Intel i7-5930K + 64 GB DDR4 memory

GPU: Maxwell Titan X + 12 GB GDDR5 memory

Can allocate data up to **(64+12) GB**

PCIe: 16 GB/sec data transfer bandwidth (gen3)

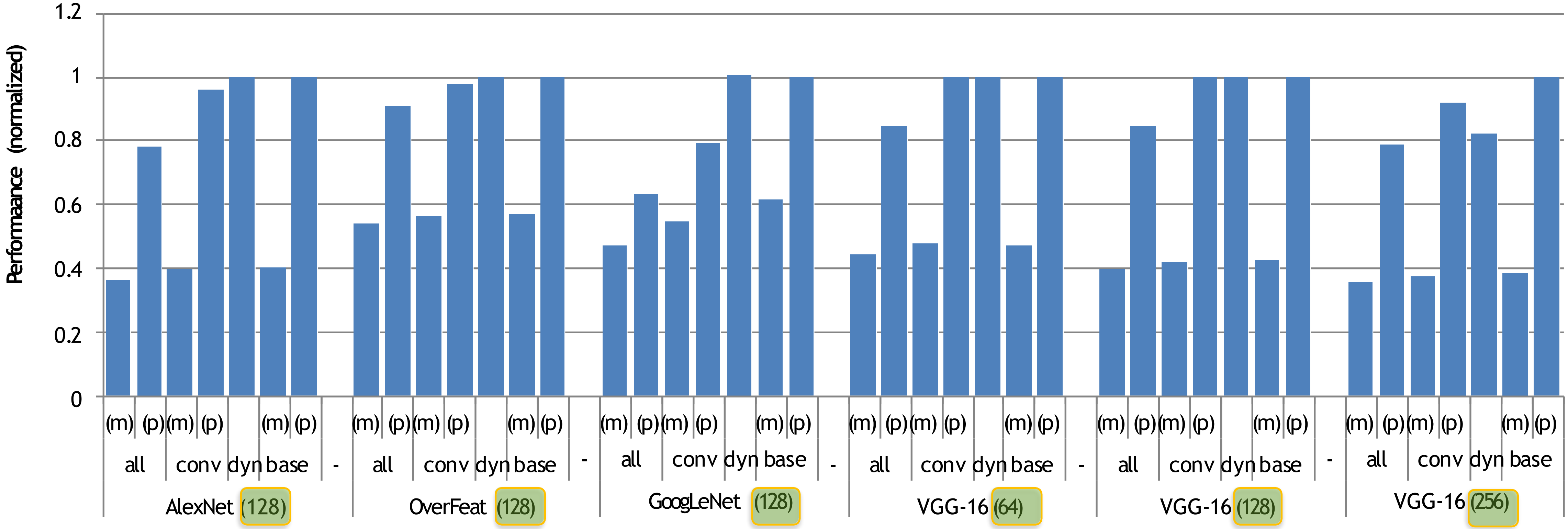
Memory usage



NVIDIA

Performance

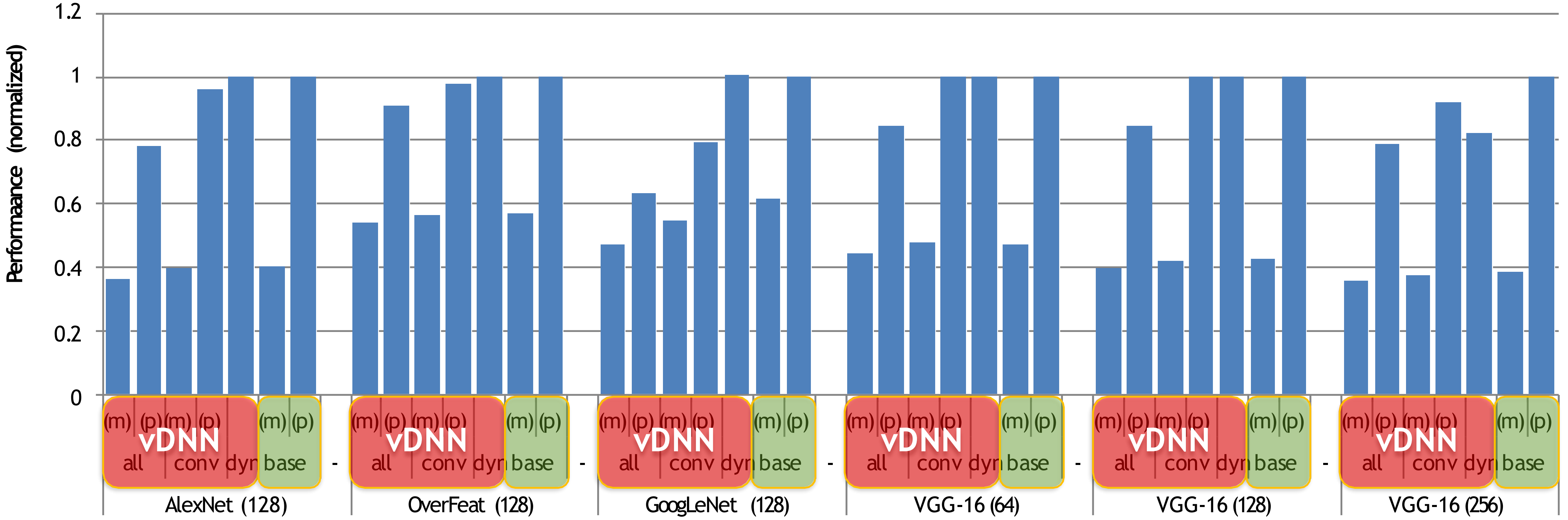
Higher is better



: mini-batch size used to train the target network

Performance

Higher is better

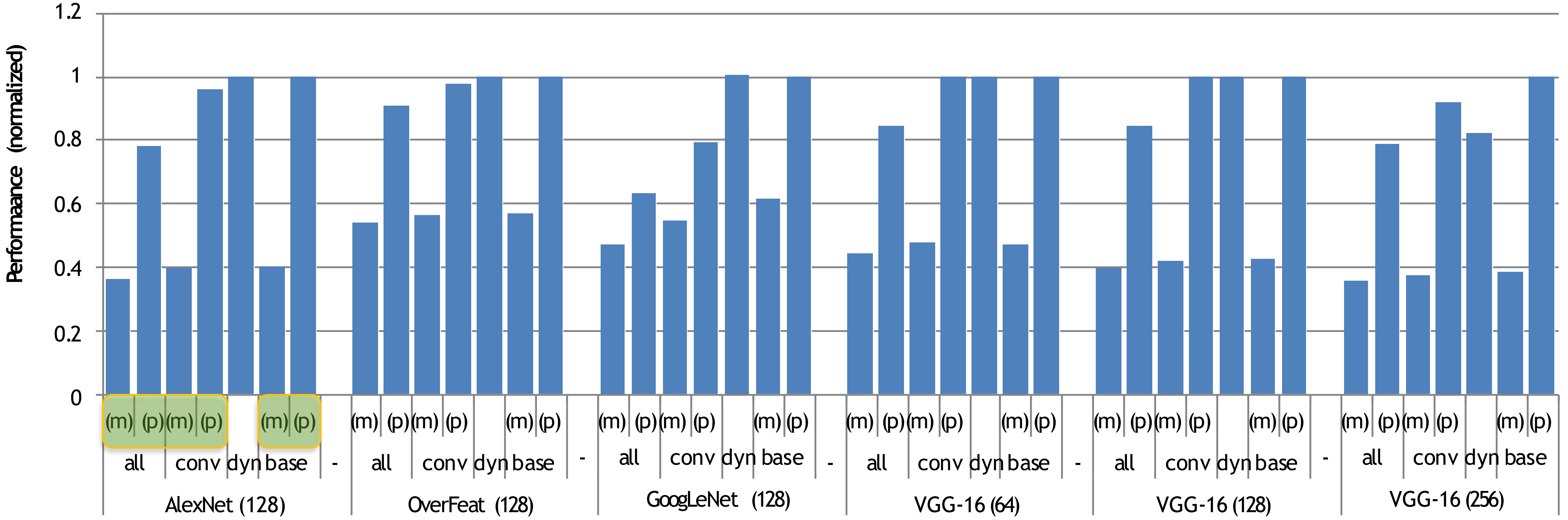


: vDNN (with different offload/prefetch policies, all / conv / dyn)

: Baseline

Performance

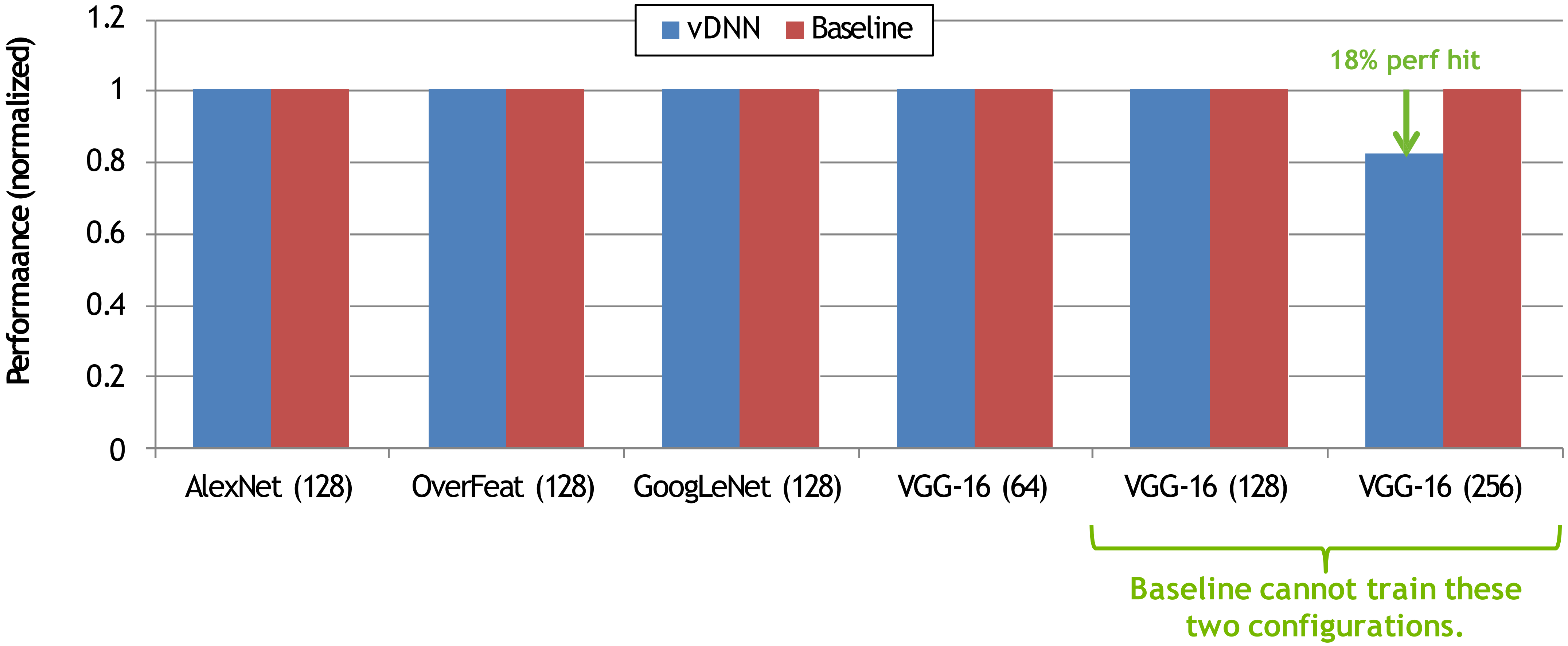
Higher is better



: convolutional algorithm chosen in cuDNN (v4), **(m)**: memory-optimal algo, **(p)**: perf-optimal algo

Performance

Higher is better



DRAM bandwidth utilization

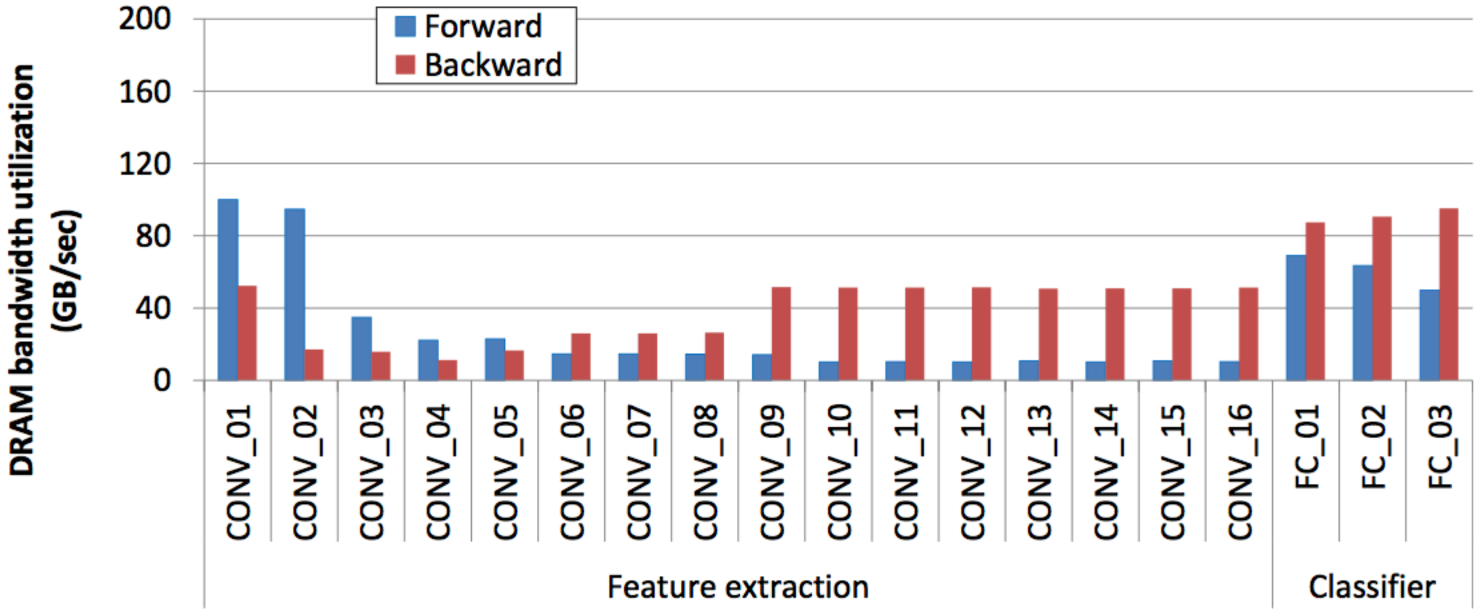
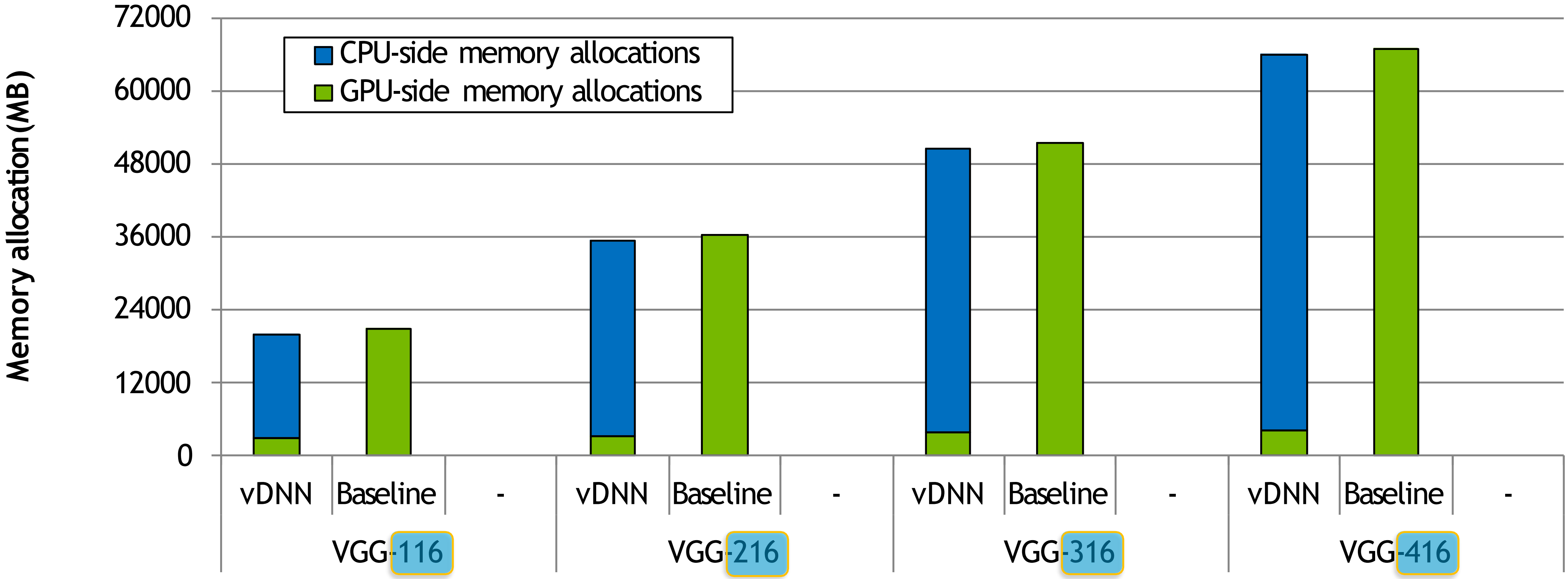


Fig. 13: Maximum DRAM bandwidth utilization for each CONV layer's forward and backward propagation.

Scalability of vDNN

Testing the trainability of vDNN with *extremely* deep networks



: number of convolutional layers

Conclusion

vDNN is a scalable, performant virtual memory solution for DNNs

GPU memory capacity bottleneck is an important problem in the ML research space

- Page-migration VM solutions incur high overhead due to OS service requests
- PCIe bw. utilization becomes extremely low (200 MB/sec)

vDNN is an application-aware/software-level direct memory management solution

- Leverages the DAG dataflow for intelligent data movements across CPU-GPU
- Maximally utilizes PCIe bandwidth (12.8 GB/sec)

Conclusion

vDNN is a scalable, performant virtual memory solution for DNNs

Reduce the average GPU memory usage:

AlexNet 89%

OverFeat 91%

GoogleNet 95%

VGG-16 90% with only 18% theoretical performance loss

Q&A