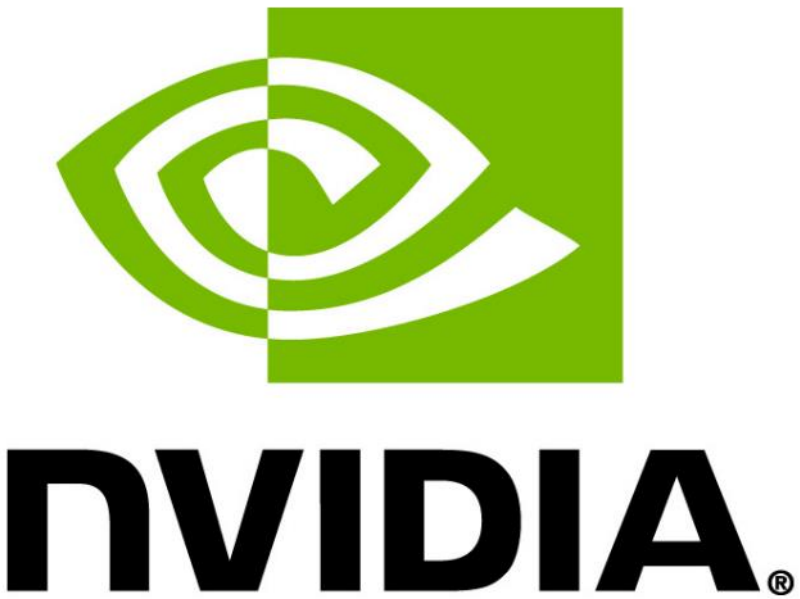
DEEP LEARNING の最新事例および TRAININGの現状

井﨑　武士，ディープラーニング部　部長
成瀬　彰，シニア ディベロッパー テクノロジー エンジニア

NVIDIA



創業1993年

共同創立者兼CEO ジェンスン・フアン
（Jen-Hsun Huang）

1999年 NASDAQに上場（NVDA）

1999年にGPUを発明
その後の累計出荷台数は10億個以上

2015年度の売上高は46億8,000万ドル

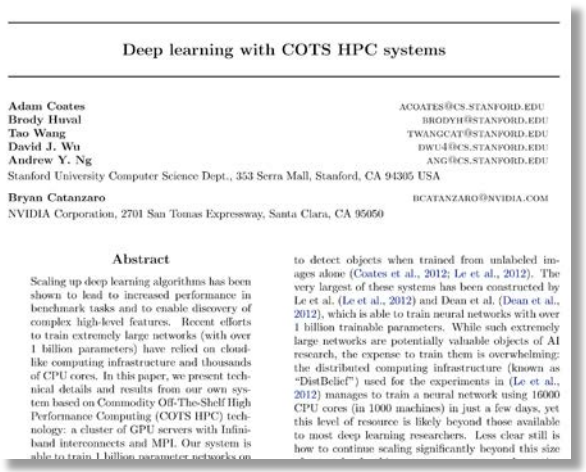
社員は世界全体で9,100人

約7,300件の特許を保有

本社は米国カリフォルニア州サンタクララ

DEEP LEARNING の最新事例

人を超えた性能に



Deep learning with COTS HPC systems

Adam Coates ACOATES@CS.STANFORD.EDU
Brody Huval BRODYH@STANFORD.EDU
Tao Wang TWANGCAT@STANFORD.EDU
David J. Wu DWU4@CS.STANFORD.EDU
Andrew Y. Ng ANG@CS.STANFORD.EDU
Stanford University Computer Science Dept., 353 Serra Mall, Stanford, CA 94305 USA

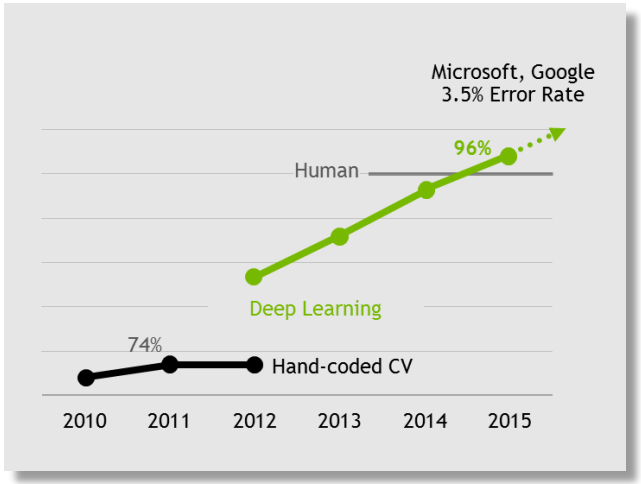
Bryan Catanzaro BCATANZARO@NVIDIA.COM
NVIDIA Corporation, 2701 San Tomas Expressway, Santa Clara, CA 95050

Abstract

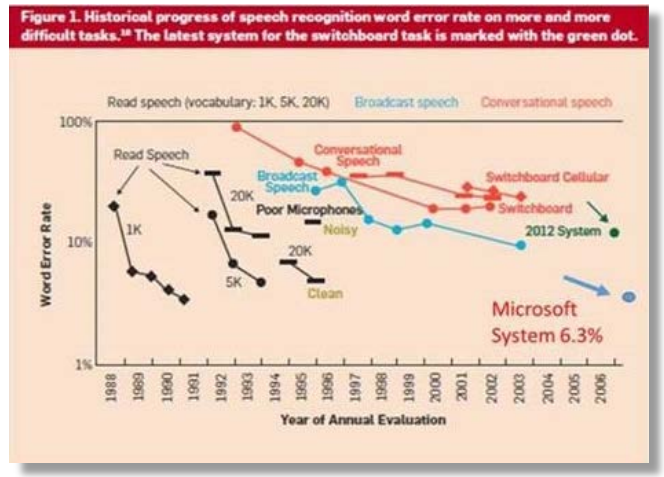
Scaling up deep learning algorithms has been shown to lead to increased performance in benchmark tasks and to enable discovery of complex high-level features. Recent efforts to train extremely large networks (with over 1 billion parameters) have relied on cloud-like computing infrastructure and thousands of CPU cores. In this paper, we present technical details and results from our own system based on Commodity Off-The-Shelf High Performance Computing (COTS HPC) technology: a cluster of GPU servers with Infiniband interconnects and MPI. Our system is able to train 1 billion parameter networks on

to detect objects when trained from unlabeled images alone (Coates et al., 2012; Le et al., 2012). The very largest of these systems has been constructed by Le et al. (Le et al., 2012) and Dean et al. (Dean et al., 2012), which is able to train neural networks with over 1 billion trainable parameters. While such extremely large networks are potentially valuable objects of AI research, the expense to train them is overwhelming: the distributed computing infrastructure (known as "DistBelief") used for the experiments in (Le et al., 2012) manages to train a neural network using 16000 CPU cores (in 1000 machines) in just a few days, yet this level of resource is likely beyond those available to most deep learning researchers. Less clear still is how to continue scaling significantly beyond this size

2012: Deep Learning 研究者がGPUを適用し始める



2015: ImageNet — Deep Learning は画像認識で人の認識率を超えた

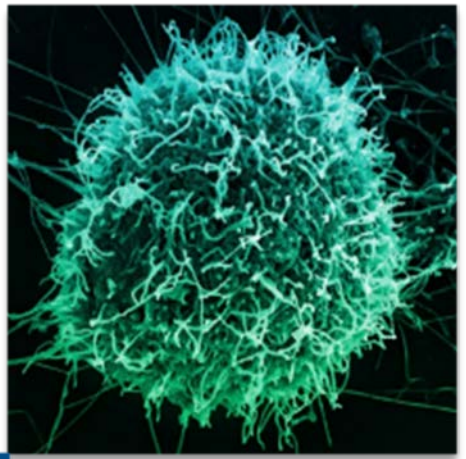


2016: Microsoft は音声認識で人を超えた認識率を達成

飛躍的に進化するディープラーニング

1秒間に600万人の
顔認識を行える
Deep Face

肺がんの検出率が
読影診断医の
1.5倍

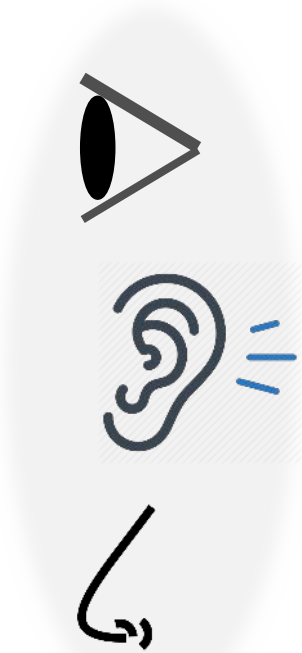


エボラウイルスの
感染力を低減する
新しい薬の候補を
発見

周囲の雑音・ノイズ下
での音声認識率を
飛躍的に改善

世界チャンピオンの
イ・セドル棋士を
Deep Mindの
AlphaGo が下す

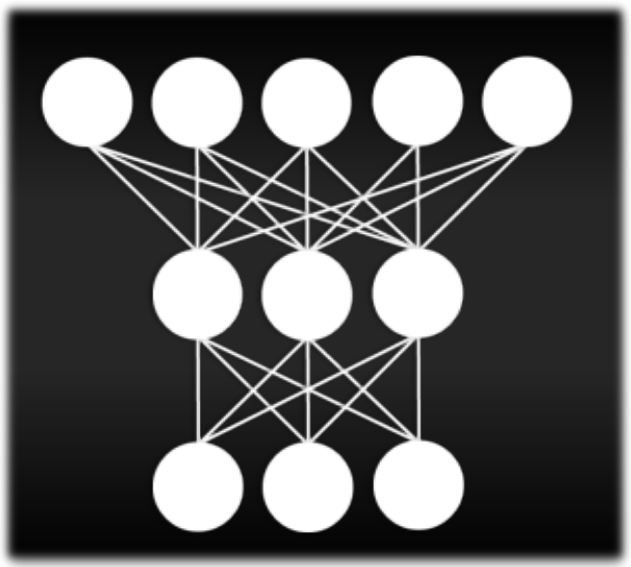
ディープラーニングとは



知覚情報

100億~140億の
神経細胞(ニューロン)

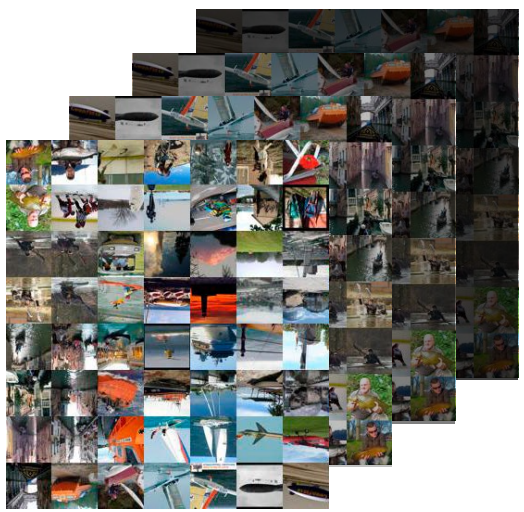
認識・判断



ニューラルネットワーク

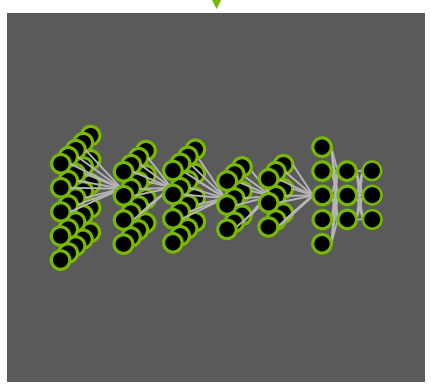
新たなコンピューティングモデル

自らコーディングするソフトウェア



学習アルゴリズム

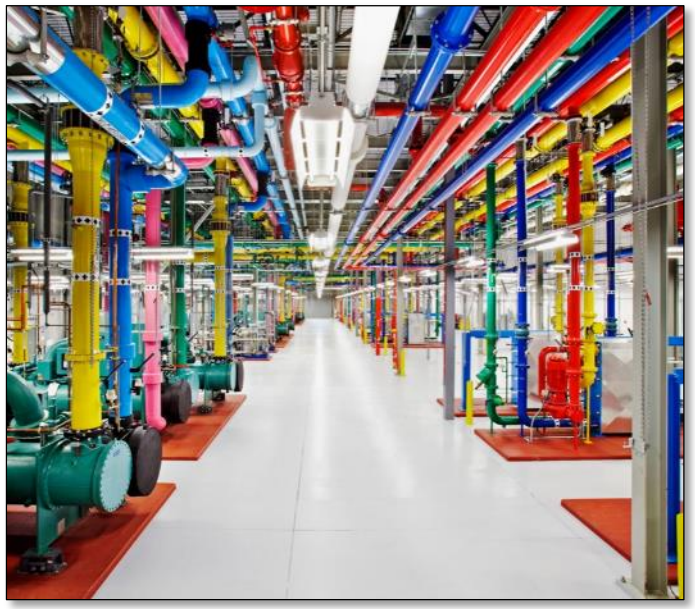
“エクサFLOPS”



“little girl is eating piece of cake"

生産性の向上 - 活用事例

FANUC:バラ積み部品ピッキング
８時間の学習で、90%の精度を
実現

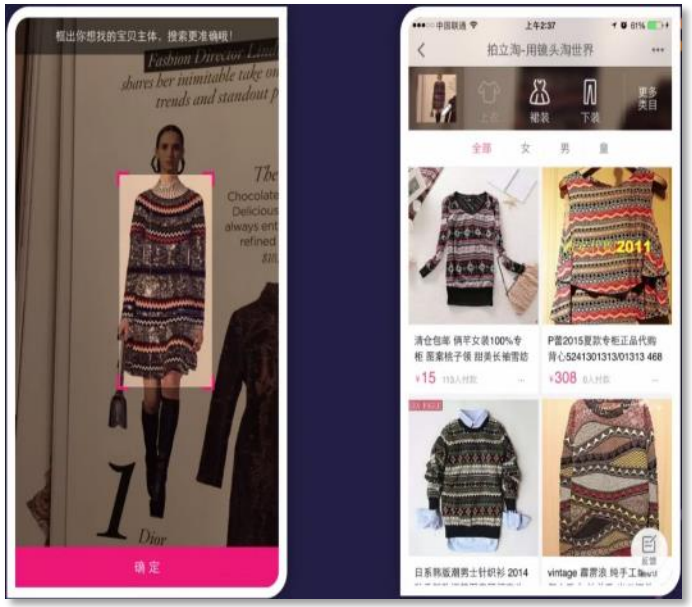


Google: AIによりデータセンターの
消費電力を数100億円規模で
低減

選択的な給水や除草により
化学薬品の使用を９０％
低減

便利な生活 - 活用事例

WEpod:大学と近隣の駅間を連絡する自動運転電気自動車が登場



Alibaba: スマートフォンで撮った写真から同一商品、類似商品を検索

Sharp: お掃除ロボットに音声認識を搭載。ハンズフリーで掃除を開始

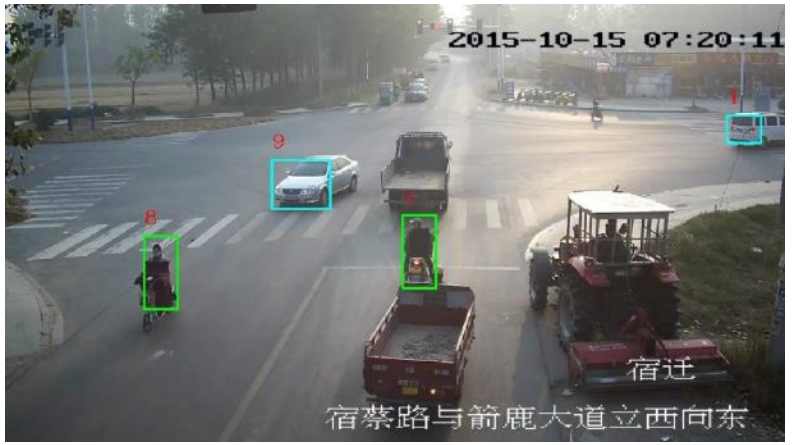
安心・安全な生活 - 活用事例

Paypal:不正決済の検出の誤報率が50%低減

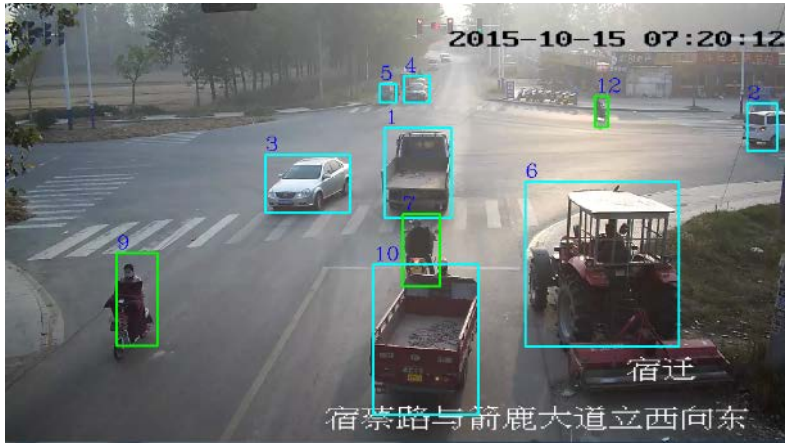
herta Security:スマート監視カメラにより空港やショッピングモールの公共安全を向上

vRad:CTスキャン画像により、潜在的に頭蓋内の出血の可能性が高い箇所を特定し、予防

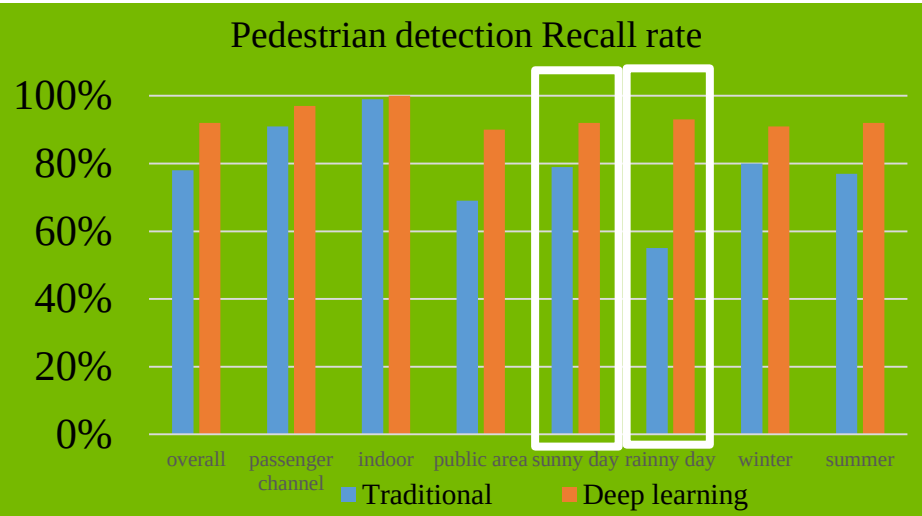
監視カメラ

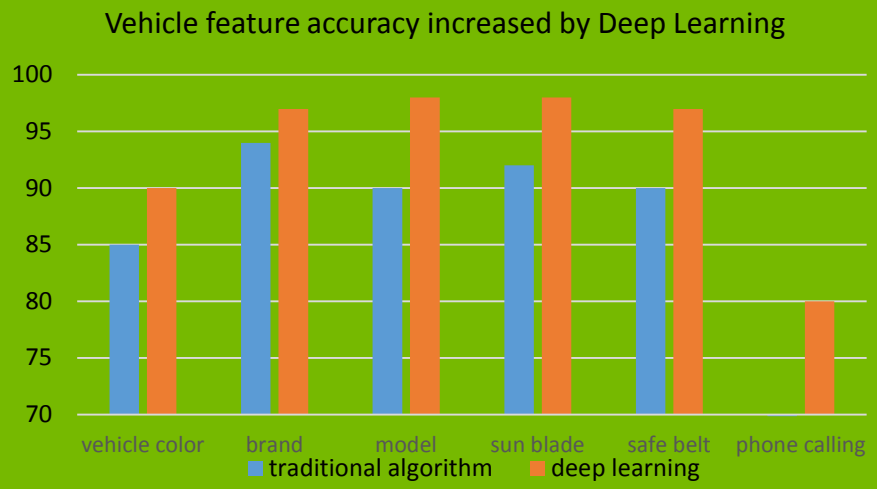


従来のアルゴリズム

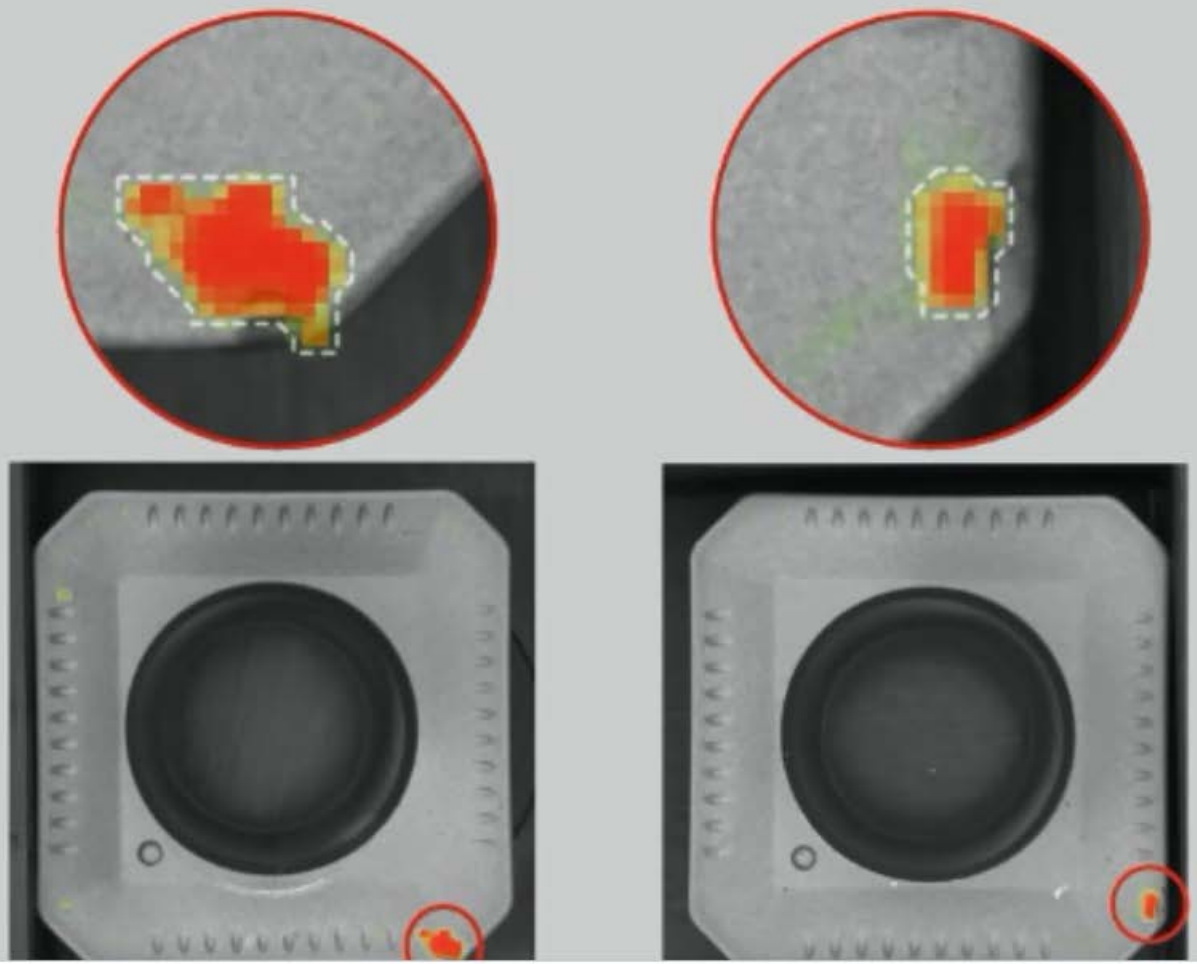


ディープラーニング





MEDICAL INSERTS | TEXTILE | CUTTING TOOLS | PRINTING | WELDING | PRE-FORM | LOGISTIC CENTER | WATCH INDUSTRY



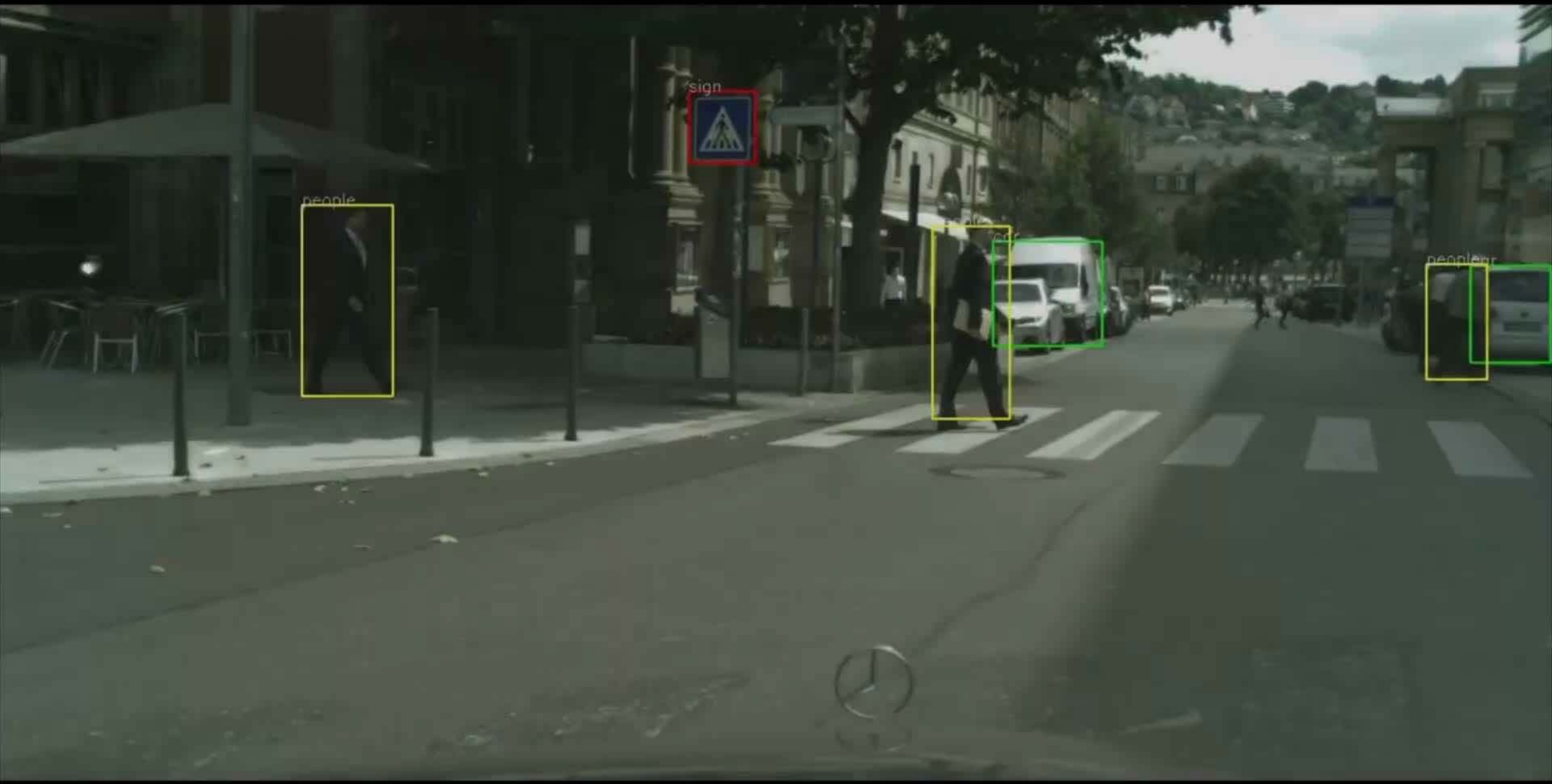
VIDI Suite

Industrial image analysis software

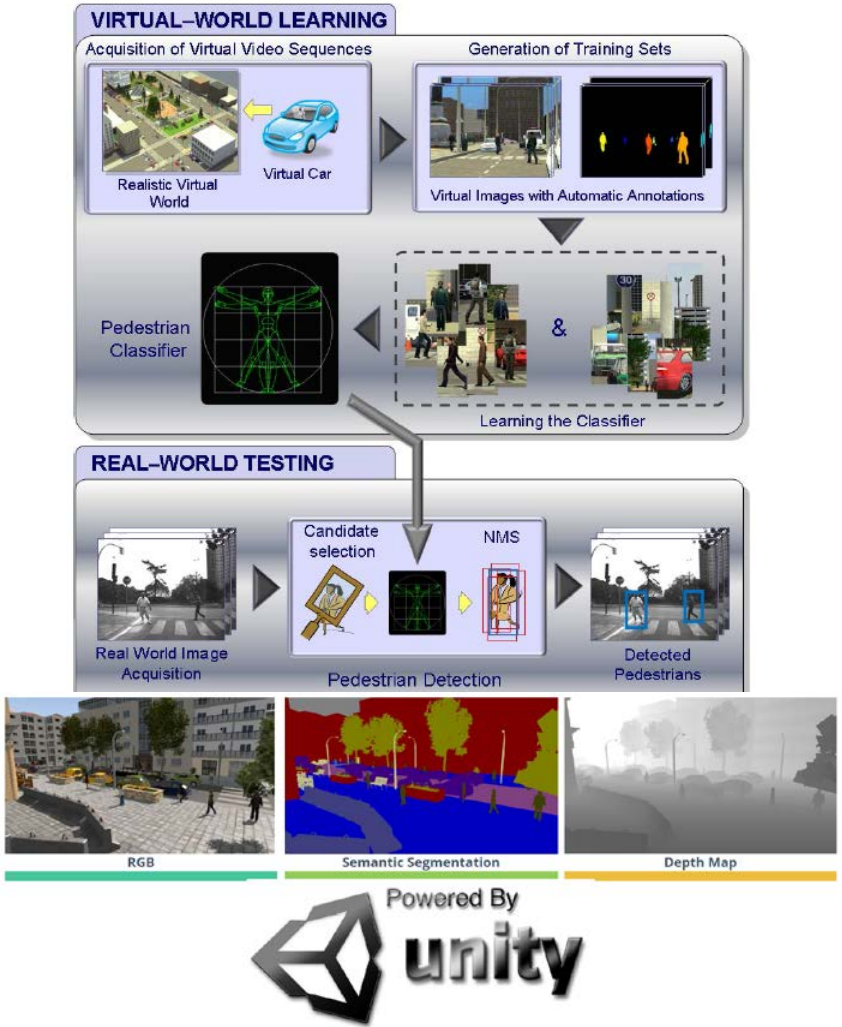
Automated inspection & classification

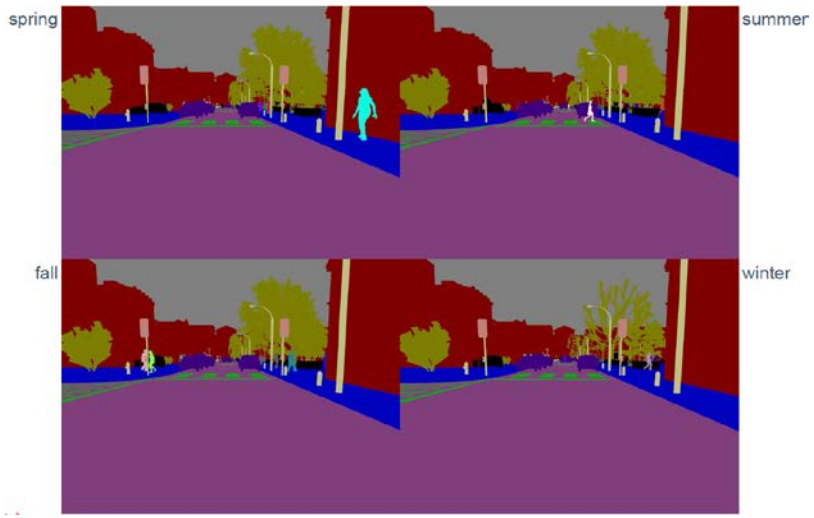
Human-like
Self-learning
Powerful

VIDI blue
VIDI red
VIDI green

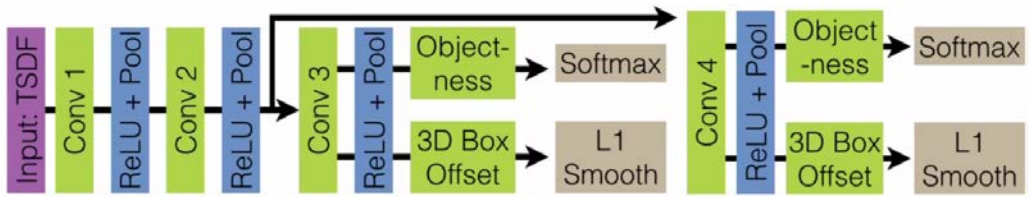

sign
people

バルセロナ自治大学/仮想世界を用いたセグメンテーション



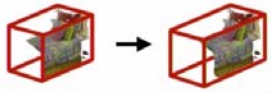


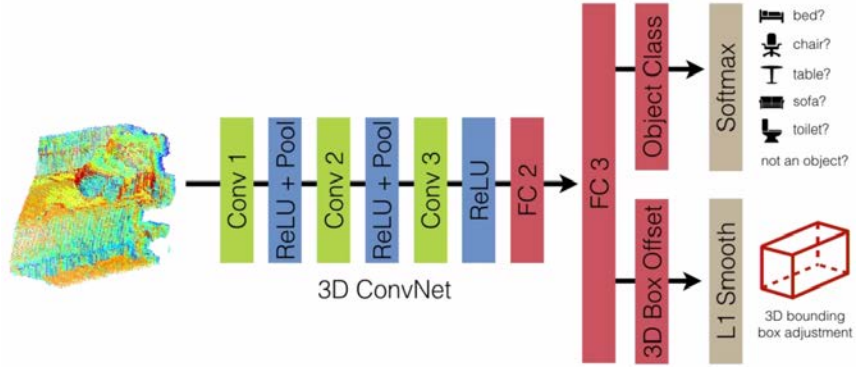
PRINCETON UNIVERSITY / 3D OBJECT RECOGNITION

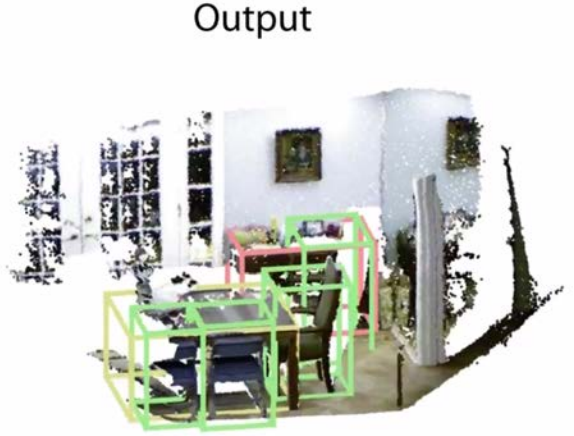


- **Objectness**: Is there an object in this box?

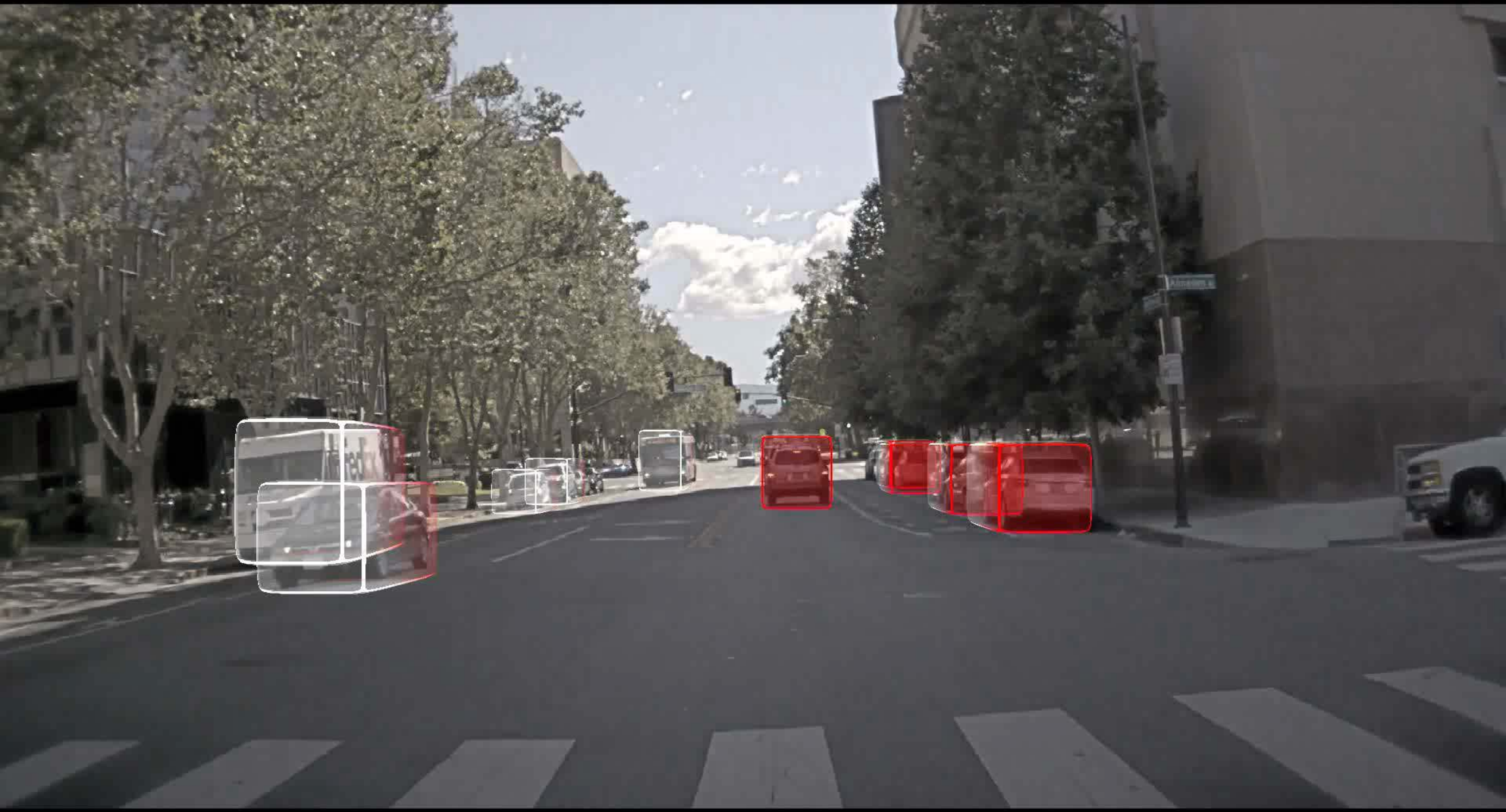
- **3D Box Offset**: How to adjust the 3D bounding box to better fit the object?



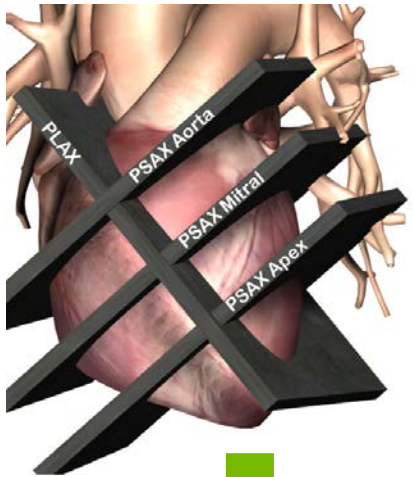




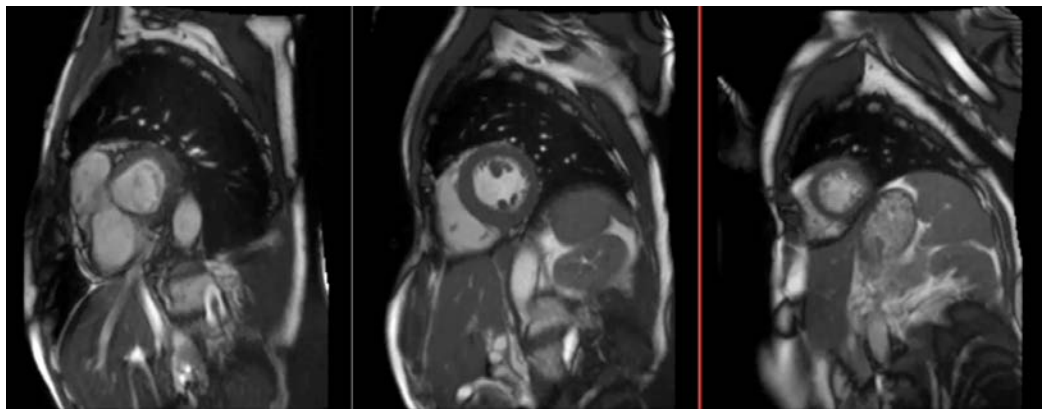
	Algorithm	Input						mAP
3D Non-Deep Learning	Sliding Shapes	Depth	33.5	29	34.5	33.8	67.3	39.6
2D Deep Learning	Depth-RCNN (segment)	Depth	71	18.2	49.6	30.4	63.4	46.5
	Depth-RCNN (segment)	RGB-D	74.7	18.6	50.3	28.6	69.7	48.4
	Depth-RCNN (CAD fit)	Depth	72.7	47.5	54.6	40.6	72.7	57.6
	Depth-RCNN (CAD fit)	RGB-D	73.4	44.2	57.2	33.4	84.5	58.5
3D Deep Learning	Ours	Depth	83.0	58.8	68.6	49.5	79.2	67.8
	Ours	RGB-D	**84.7**	**61.1**	**70.5**	**55.4**	**89.9**	**72.3**



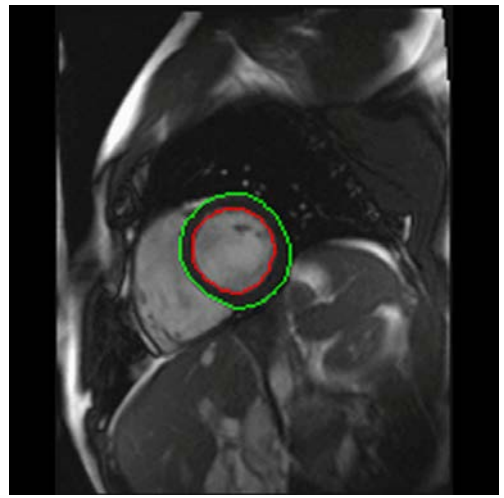
ディープラーニングによる駆出率の計測



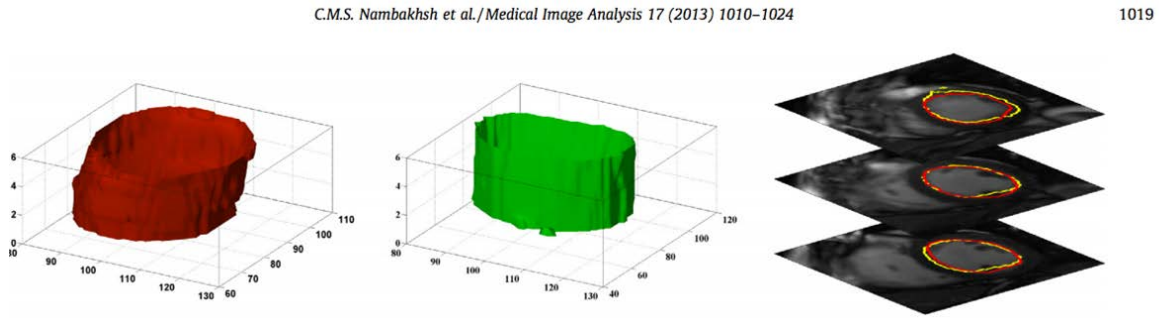
MRI imaging



Manual annotation



Software volume estimate



NIH

Booz | Allen | Hamilton

INTELLIGENT VOICE/CNNを用いた方言分類

NIST LRE Competition

6言語、20方言

アラビア語(エジプト、イラク、レバノン、
マグレビ、標準語)

中国語(広東、北京、上海、台湾)

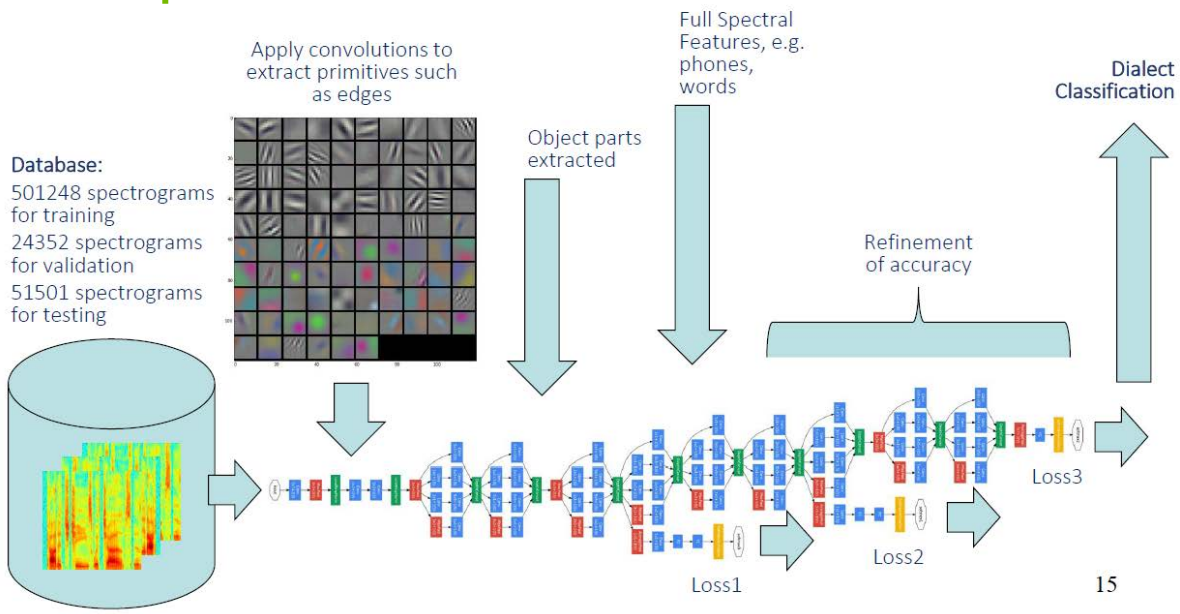
英語(英国、米国、インド)

フランス語(西アフリカ、ハイチ)

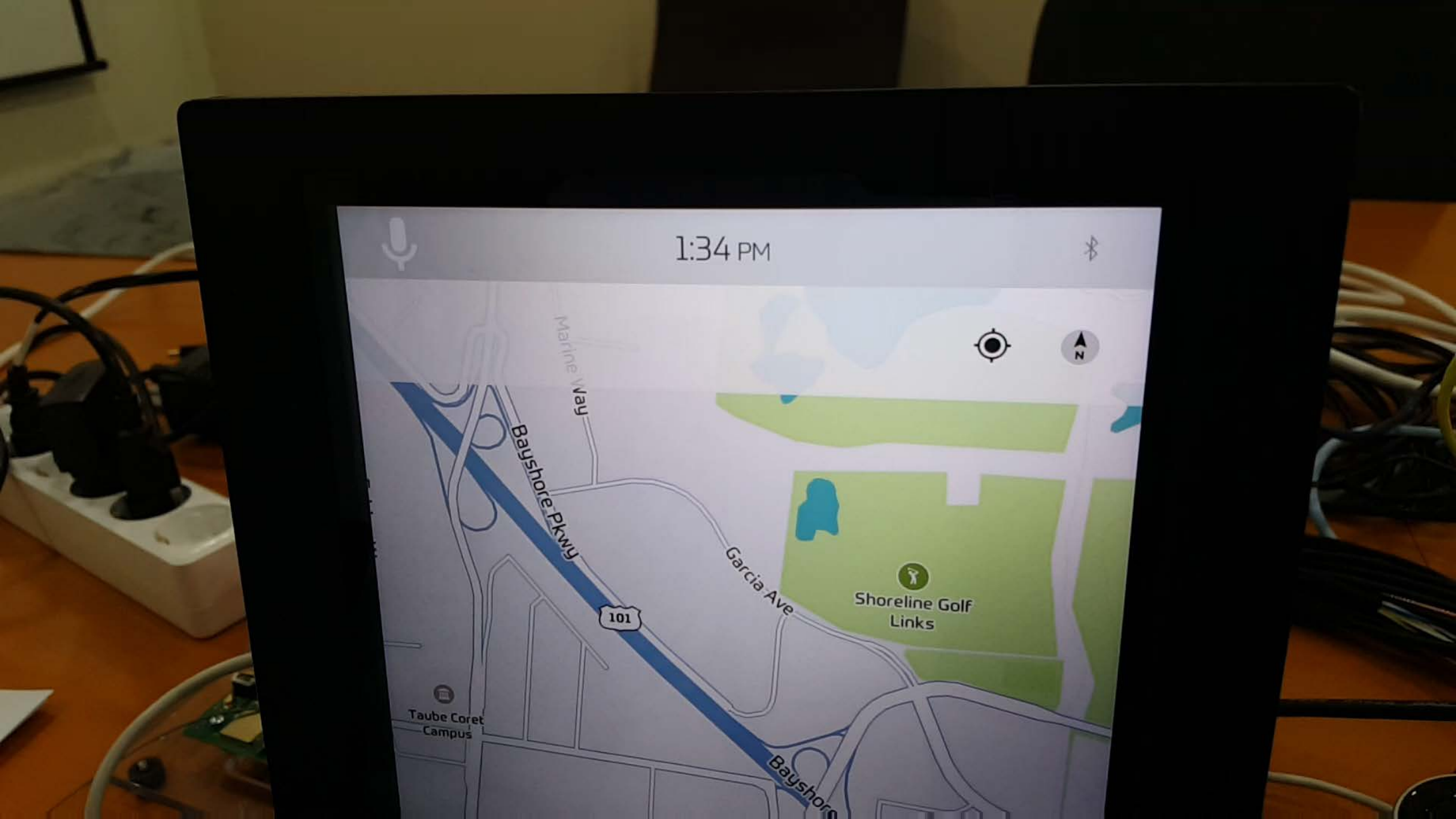
リベリア語 （カリブスペイン、ヨーロッパスペイン
ラテンアメリカスペイン、ブラジルポルトガル）

スラブ語(ポーランド、ロシア)

500時間以上のスピーチデータ



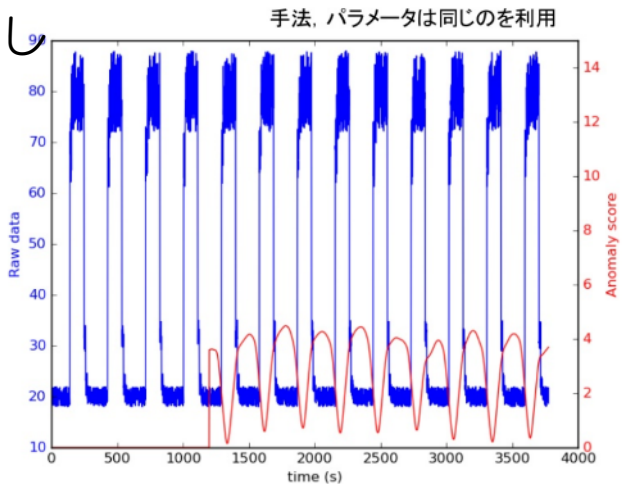
- Accuracy – 83.99 (Top-1), 98.89% (Top-5)

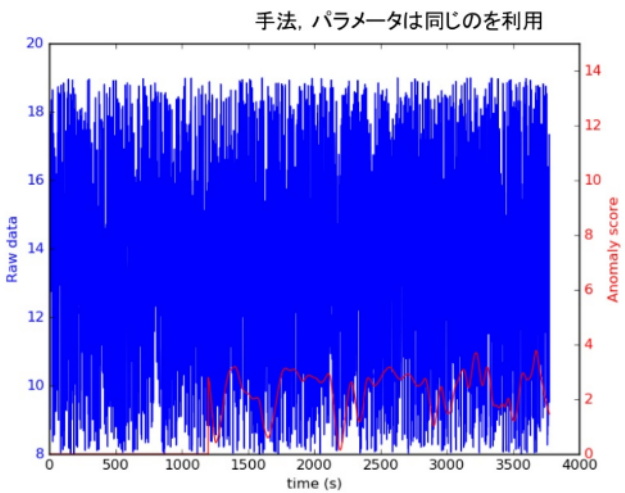

1:34 PM
Marine Way
Bayshore Pkwy
101
Garcia Ave
Shoreline Golf Links
Taube Coret Campus
N

異常検知

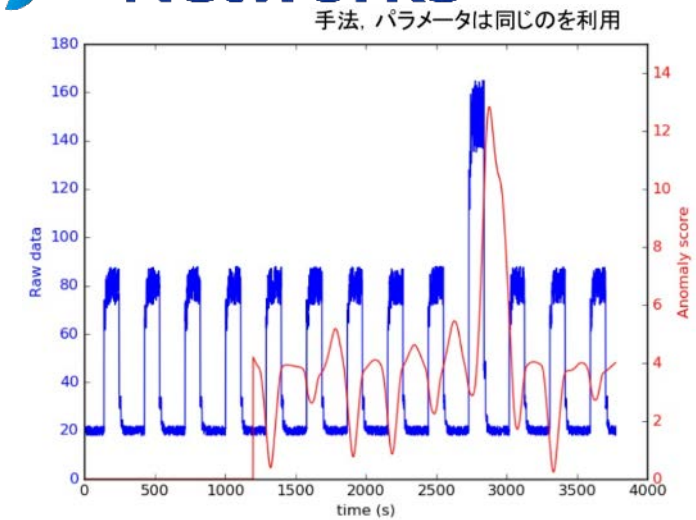
Preferred Networks

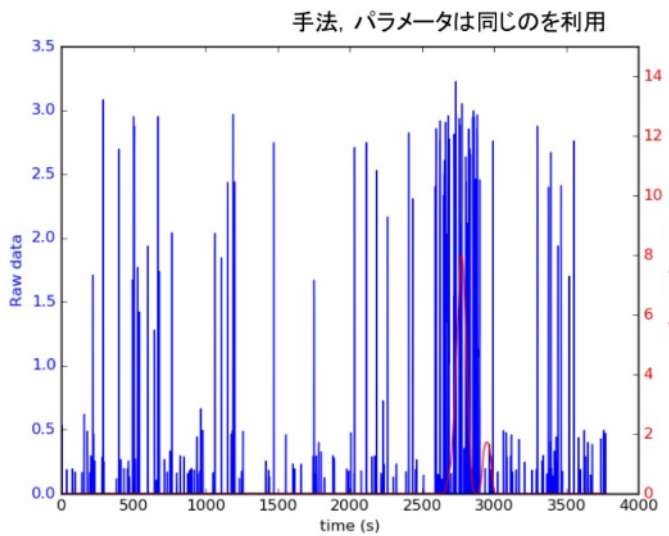
異常なし

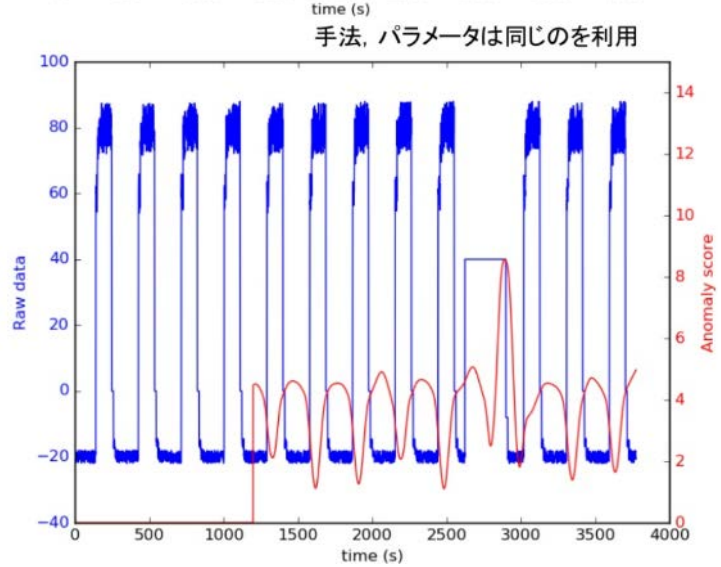




異常あり



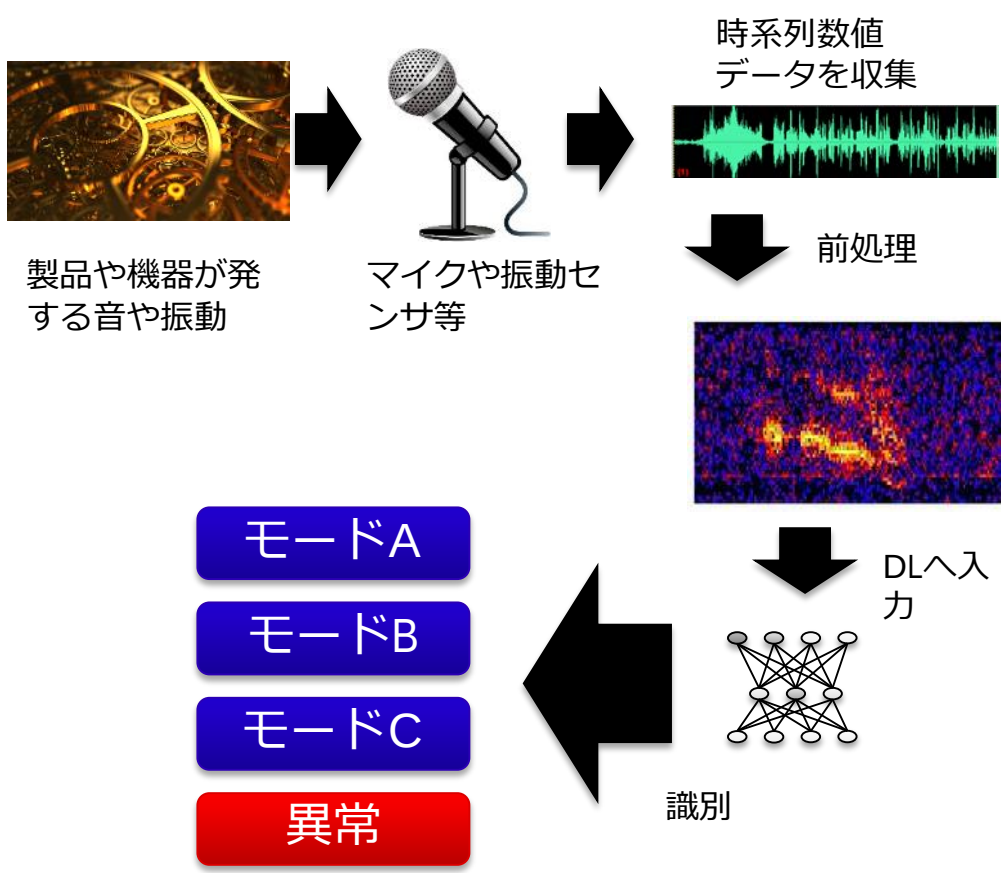




異音検知

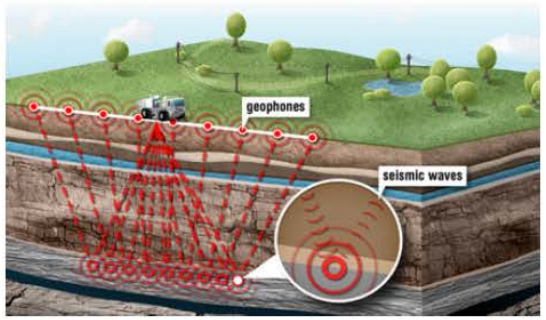
組み立て済みの機器の検品時に、機器の動作音をベテランが判断して異常検知している。属人性が高くスケールが困難である。長い経験により習得した技能であり、継承が困難。高齢化によって高周波成分の聞き取り能力が低下＝＞検知性能の低下が疑われるが、客観的な判断手段がないため対応が難しい。

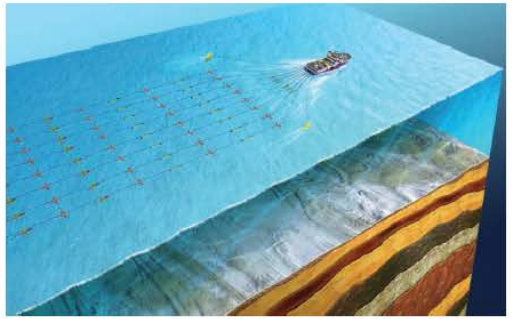
DL技術を応用し、音響認識技術を考案。ベテランが検知できなかった異常（後日故障により異常検査で判明）を検知することに成功。

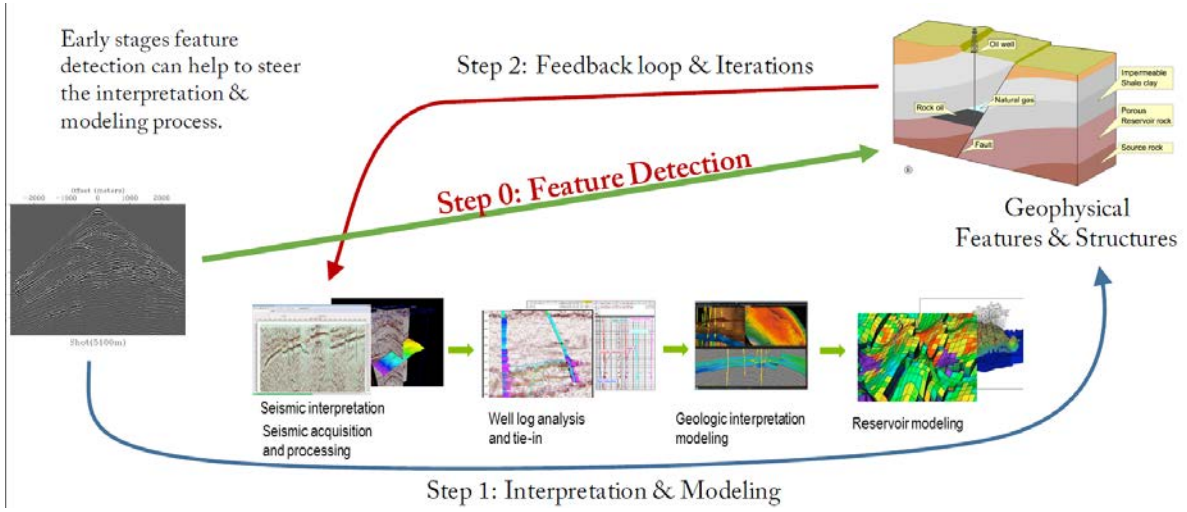


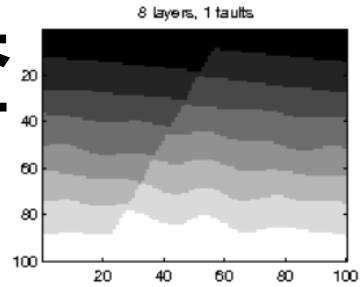
MIT/弹性波探查

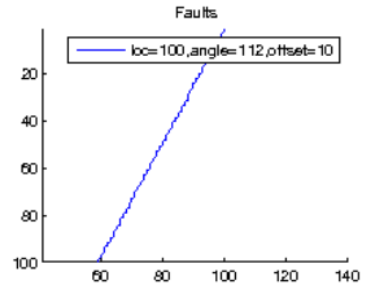
Data acquisition, on/off shore.

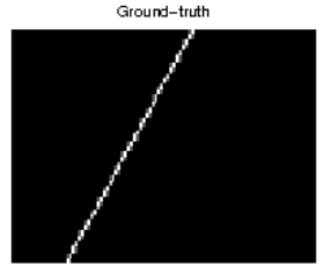


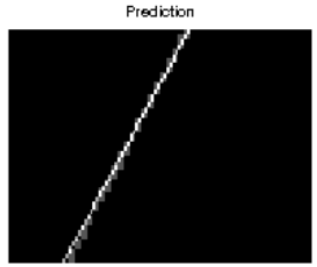


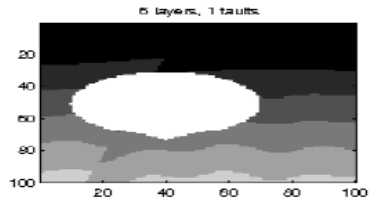


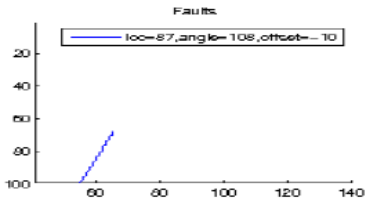






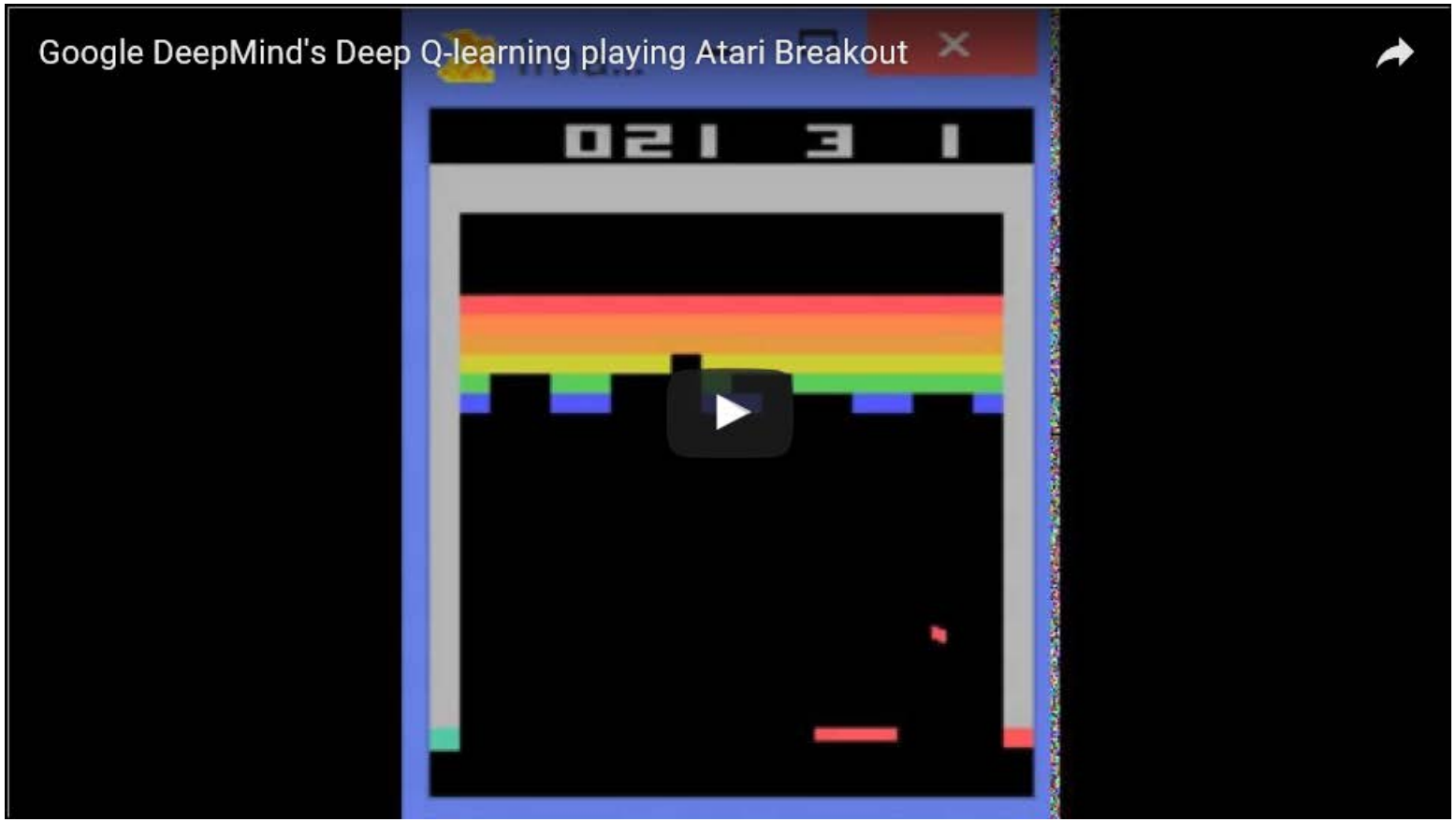






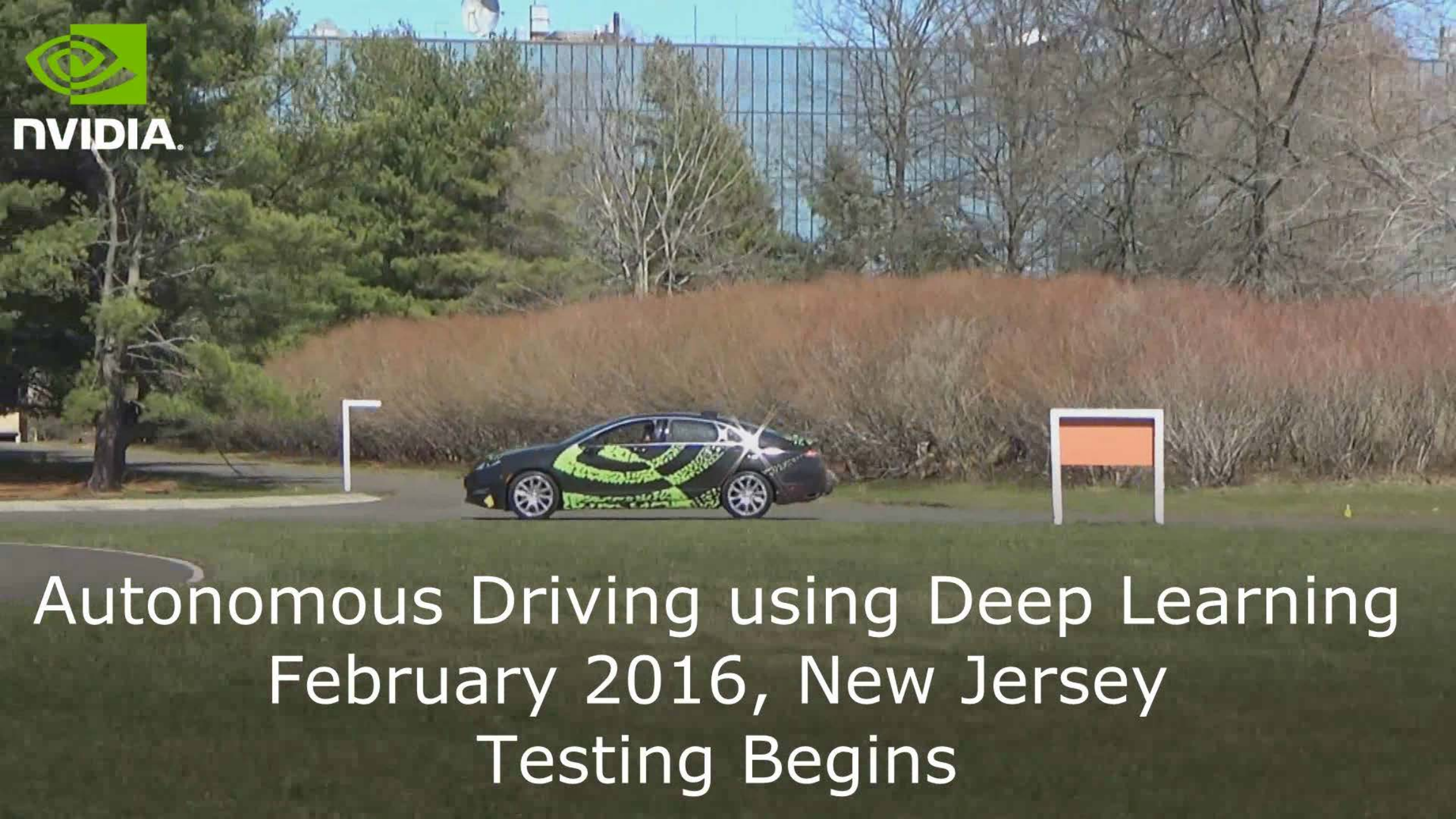
DEEP REINFORCEMENT LEARNING

Taking action from visual input



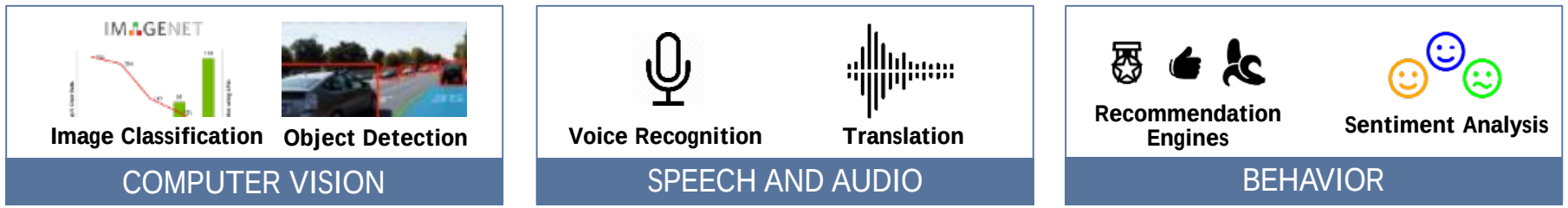
https://youtu.be/V1eYniJ0Rnk

Google DeepMind


NVIDIA
Autonomous Driving using Deep Learning
February 2016, New Jersey
Testing Begins

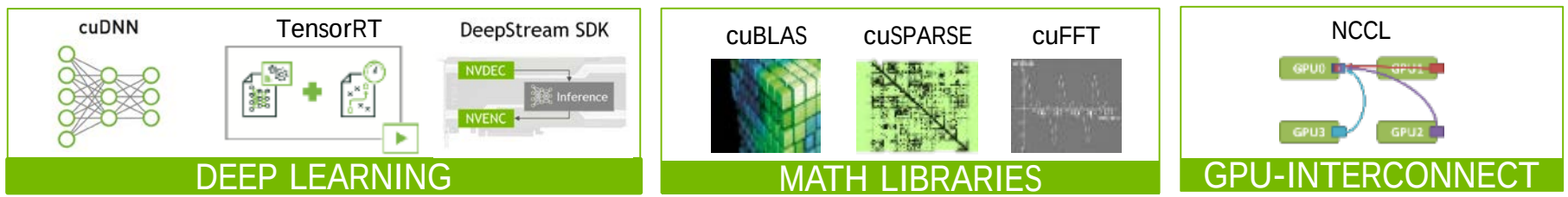
エヌビディア ディープラーニング プラットフォーム

APPLICATIONS



FRAMEWORKS

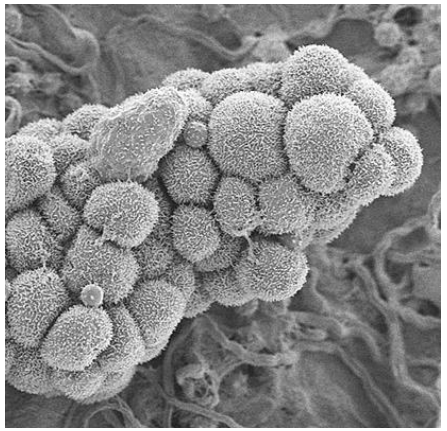
DEEP LEARNING SDK

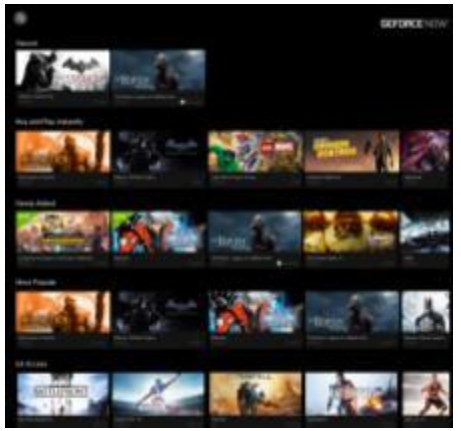


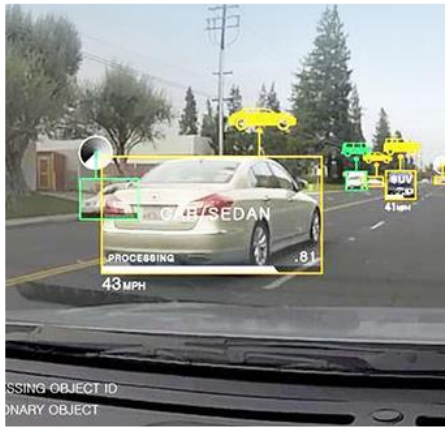
GPU PLATFORM

Deep Learningの最新事例および Trainingの現状 (後半)

Deep Learning Everywhere







Internet & Cloud

Image Classification
Speech Recognition
Language Translation
Language Processing
Sentiment Analysis
Recommendation

Medicine & Biology

Cancer Cell Detection
Diabetic Grading
Drug Discovery

Media & Entertainment

Video Captioning
Video Search
Real Time Translation

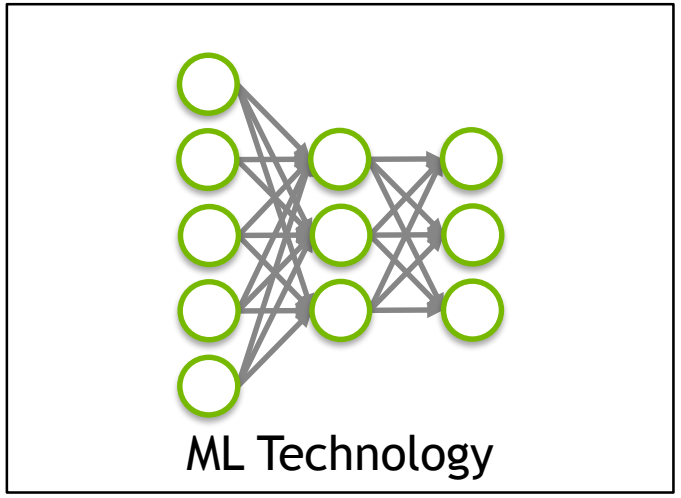
Security & Defense

Face Detection
Video Surveillance
Satellite Imagery

Autonomous Machines

Pedestrian Detection
Lane Tracking
Recognize Traffic Sign

3 Key Things to Enable Deep Learning

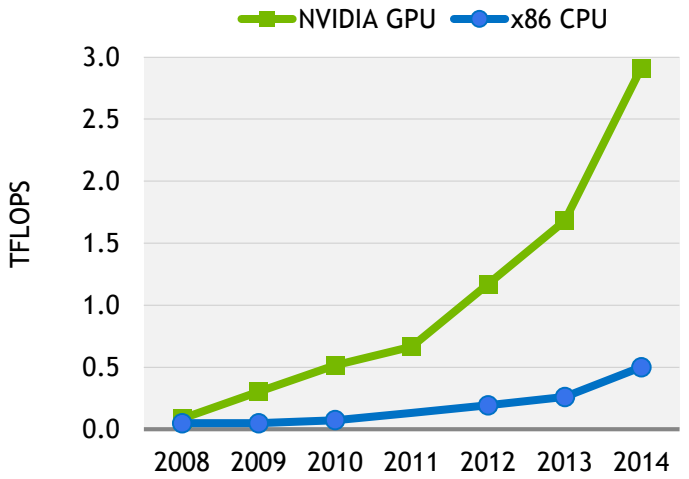


TORCH	CAFFE
NYU facebook	Berkeley
THEANO	MATCONVNET
Université de Montréal	University of Oxford
MOCHA.JL	PURINE
Massachusetts Institute of Technology	NUS, Carnegie Mellon University
MINERVA	MXNET*
NYU, Tsinghua University	University of Washington, Carnegie Mellon University

facebook 350 million images uploaded per day

Walmart 2.5 trillion transactions per hour

YouTube 100 hour video uploaded per minute

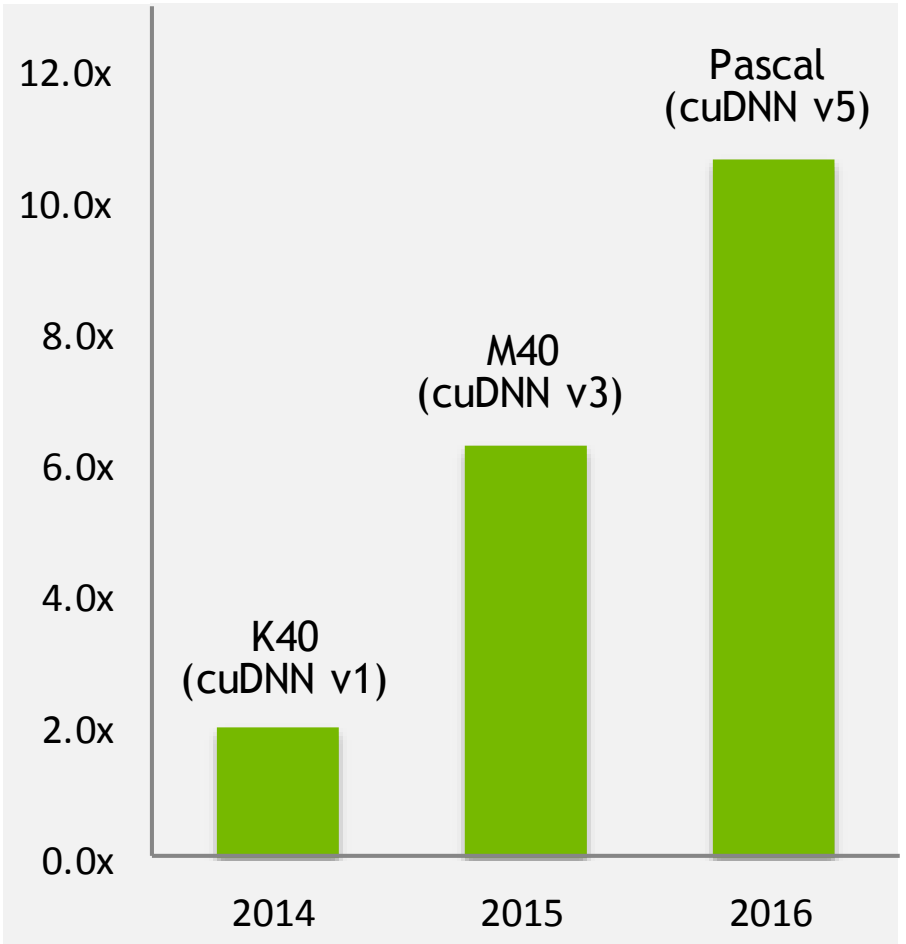


Kepler → Maxwell → Pascal

Focus on DL

- Improve training speed
  - More than 10x in 3 years
- Eenhance GPU architecture for DL
- Optimize software for DL
  - cuDNN, cuBLAs, …

AlexNet training throughput based on 20 iterations, CPU: 1x E5-2680v3 12 Core 2.5GHz.



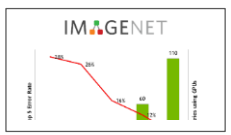
Deep Learning SDK

All Leading DL frameworks using NVIDIA DL SDK

COMPUTER VISION

OBJECT DETECTION

IMAGE CLASSIFICATION



SPEECH & AUDIO

VOICE RECOGNITION

LANGUAGE TRANSLATION

NATURAL LANGUAGE PROCESSING

RECOMMENDATION ENGINES

SENTIMENT ANALYSIS

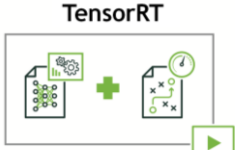
DL Frameworks

Caffe
Chainer
DL4J
Deeplearning4j
Mocha.jl
julia
K
KERAS
Microsoft
CNTK
mxnet
Purine
TensorFlow
torch
MatConvNet
MINERVA
OpenDeep
Pylearn2
theano

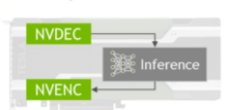
NVIDIA DEEP LEARNING SDK

cuDNN

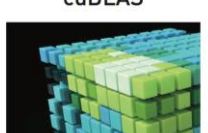
TensorRT



DeepStream SDK



cuBLAS



cuSPARSE

NCCL

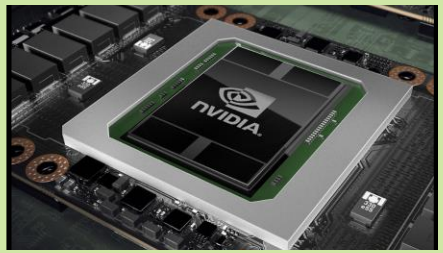
developer.nvidia.com/deep-learning-software

TESLA P100: Pascal GPU

TESLA P100: The Engine for Deep Learning

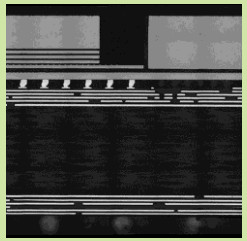
New GPU Architecture to Enable the World's Fastest Compute Node

Pascal Architecture



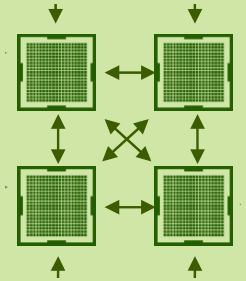
Highest Compute Performance

CoWoS HBM2



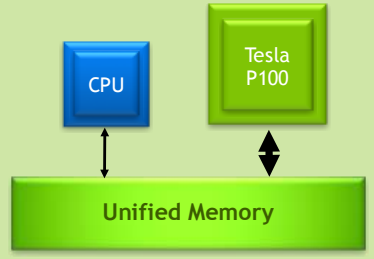
Unifying Compute & Memory in Single Package

NVLink

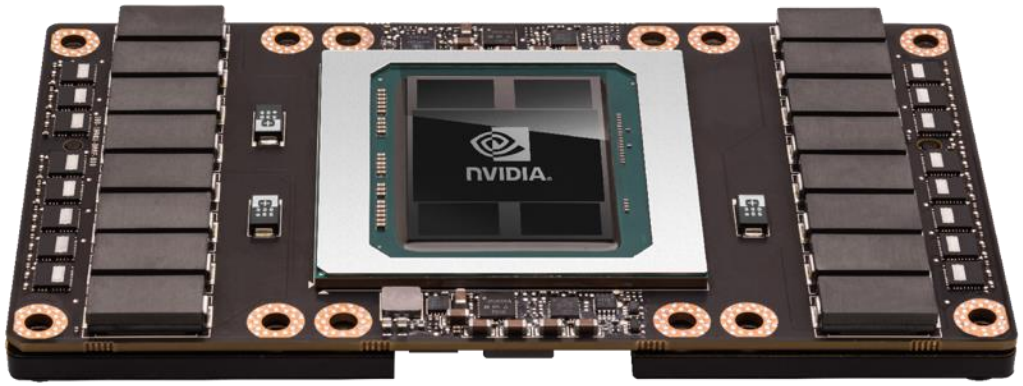


GPU Interconnect for Maximum Scalability

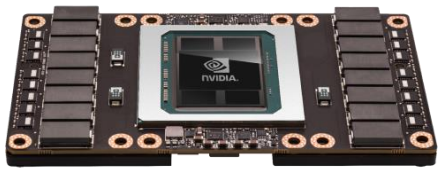
Page Migration Engine



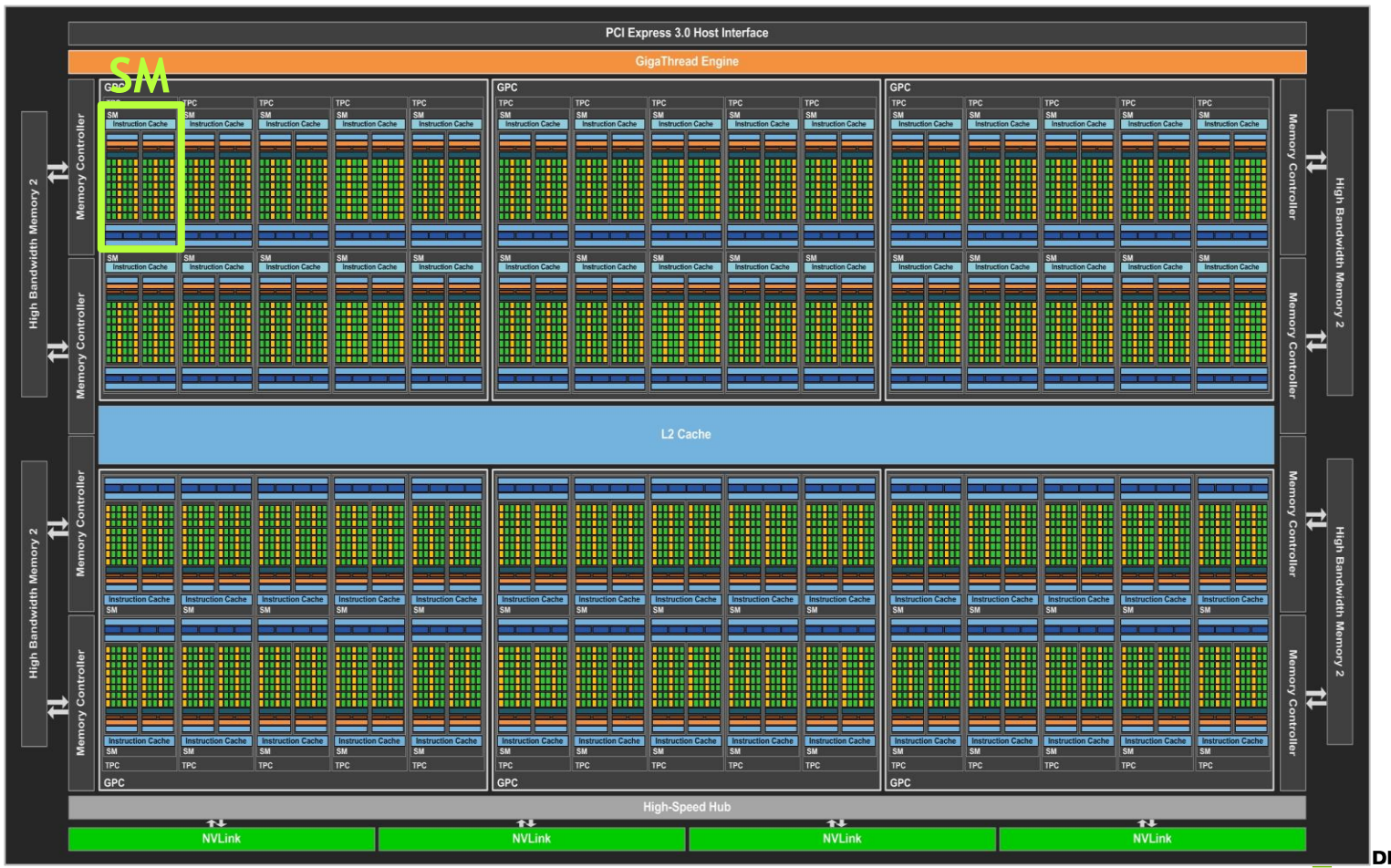
Simple Parallel Programming with Virtually Unlimited Memory



TESLA P100 (GP100)



3584 CUDA cores (56 SMs x 64 CUDA cores/SM)



Variable Precision

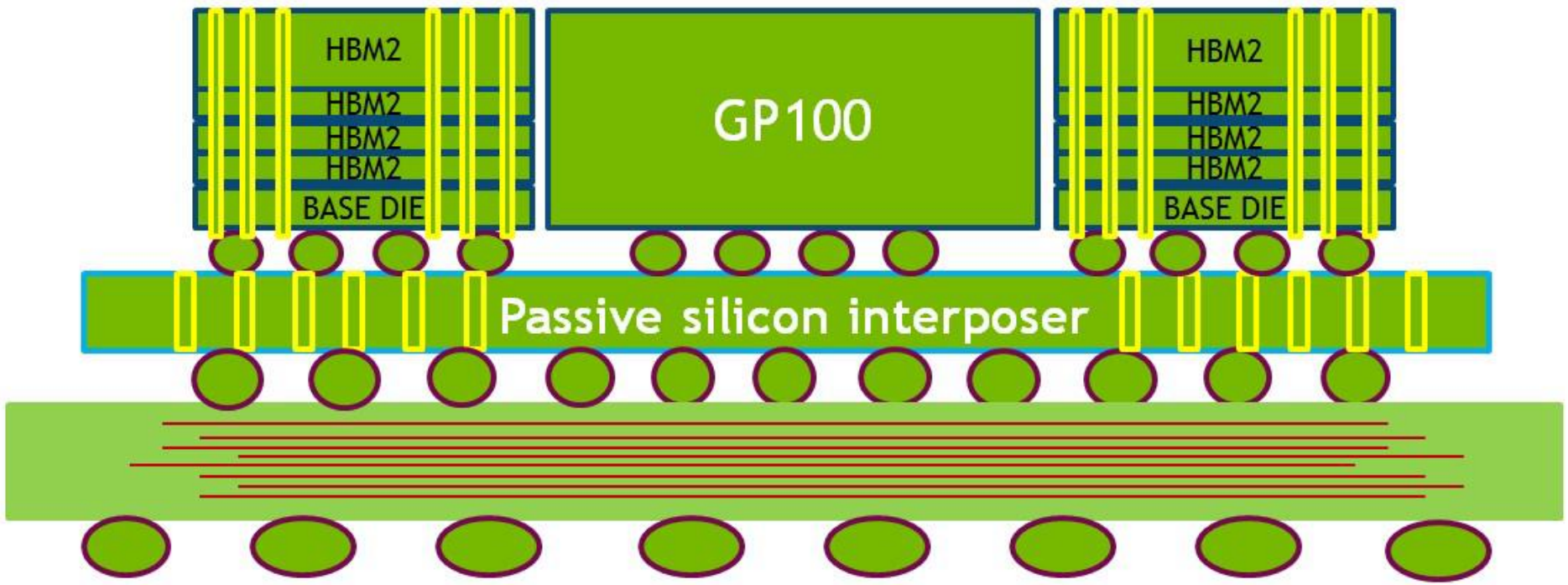
- FP64: 5.3 TF
- FP32: 10.6 TF
- FP16: 21.2 TF

HBM2

- 4096 bit width
- 16 GB
- 732 GB/s

HBM2: 732GB/s memory bandwidth

16GB, native ECC, 4K-bits interface

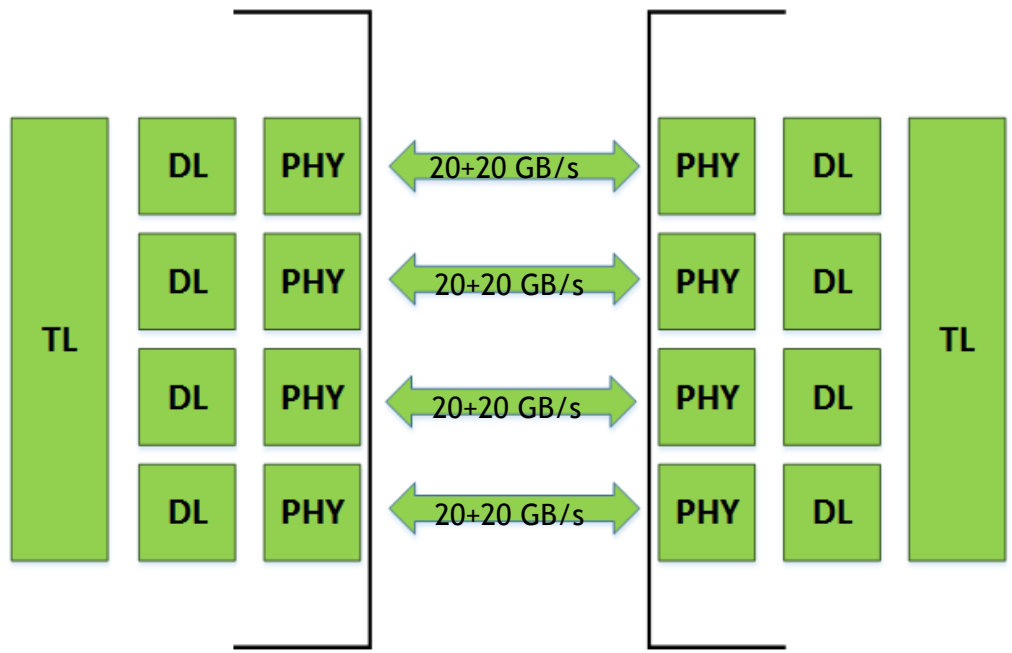


NVLINK

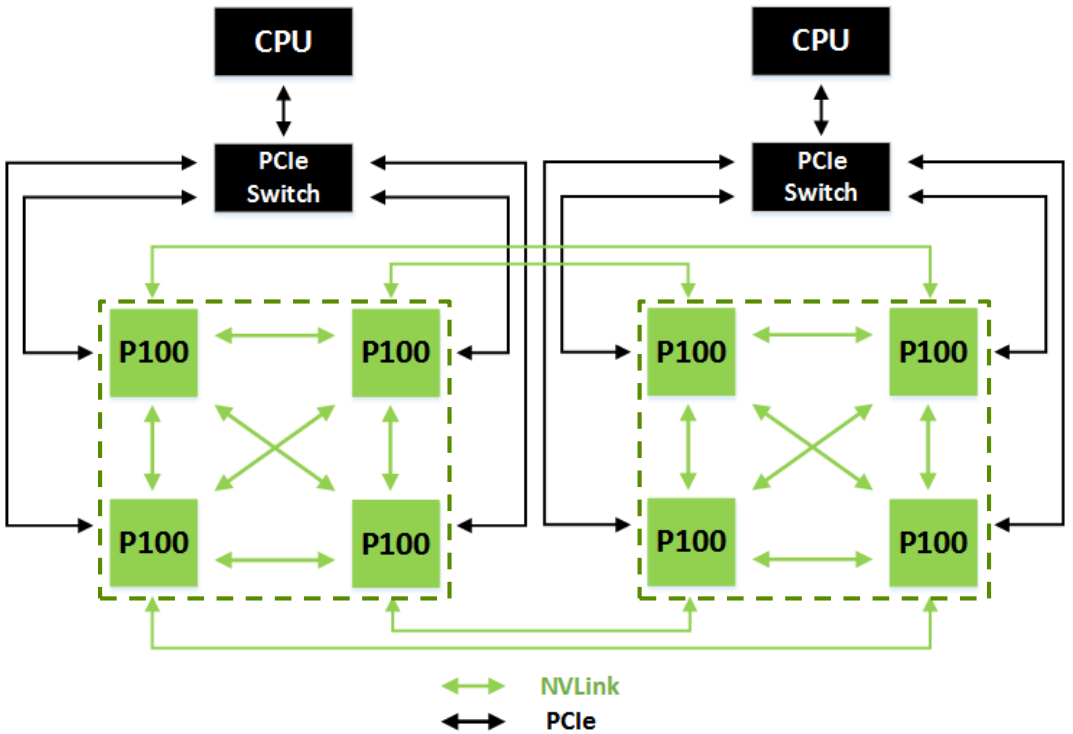
High speed interconnect

- P100 supports 4 NVLinks
  - 20+20 GB/s per link
  - Up to 94% bandwidth efficiency
- Links can be ganged for higher bandwidth
- Supports read/writes/atomics to peer GPU
- Supports read/write access to NVLink-enabled CPU

NVLink
on Tesla P100



NVLINK GPU CLUSTER: DGX-1

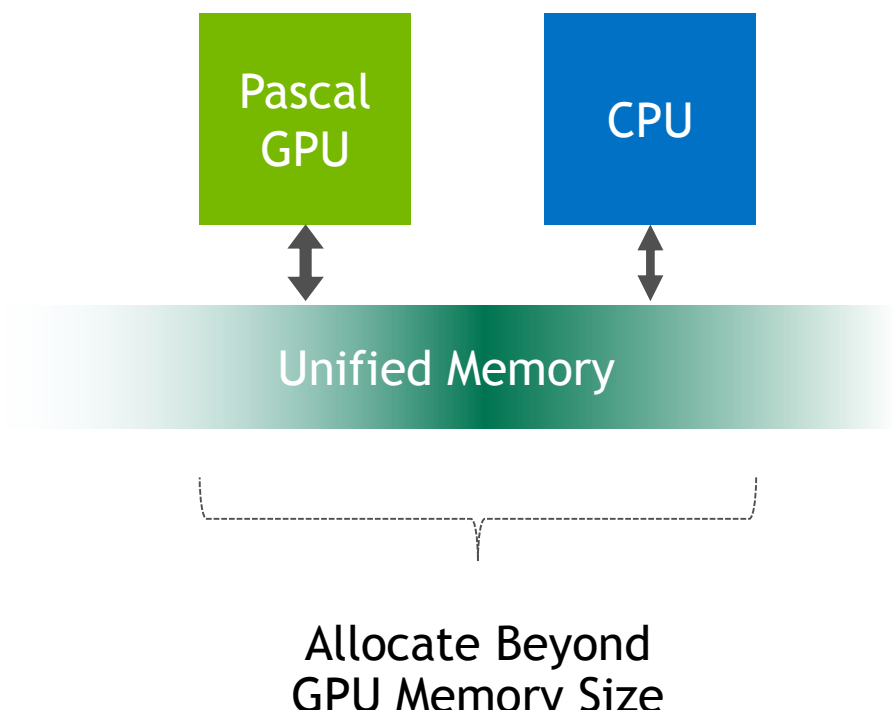


- Two gropus with fully connected 4 GPUs
  - Each GPU is connected to one GPU of peer group
- Load/store access to Peer GPU Memory
- High speed copy engines for bulk data copy
- No hardware routing

Unified Memory

Pascal Page Migration Engine

Pascal and CUDA 8

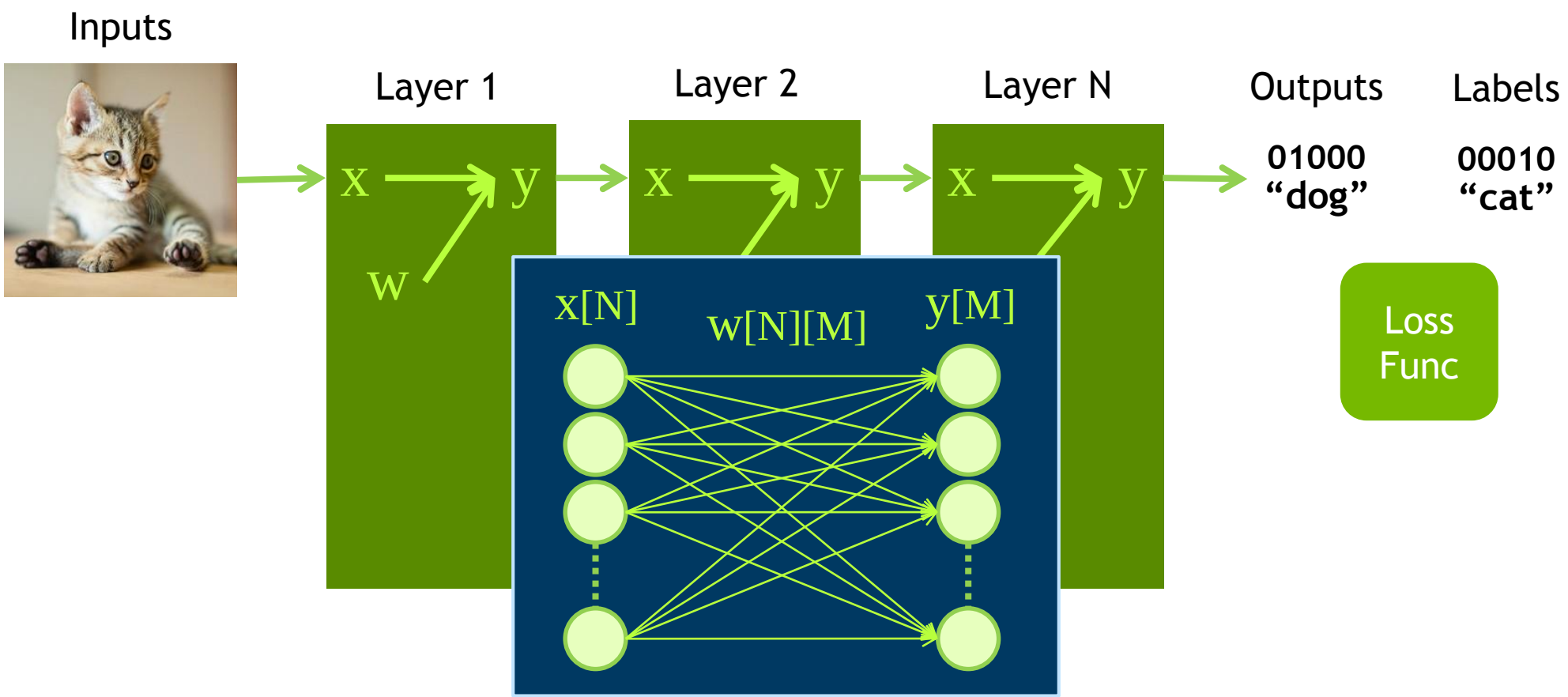


Enable Large Data Models	Oversubscribe GPU memory Allocate up to system memory size
Simpler Data Accesss	CPU/GPU Data coherence Unified memory atomic operations
Tune Unified Memory Performance	Usage hints via cudaMemAdvise API Explicit prefetching API

Deep Learning: Training

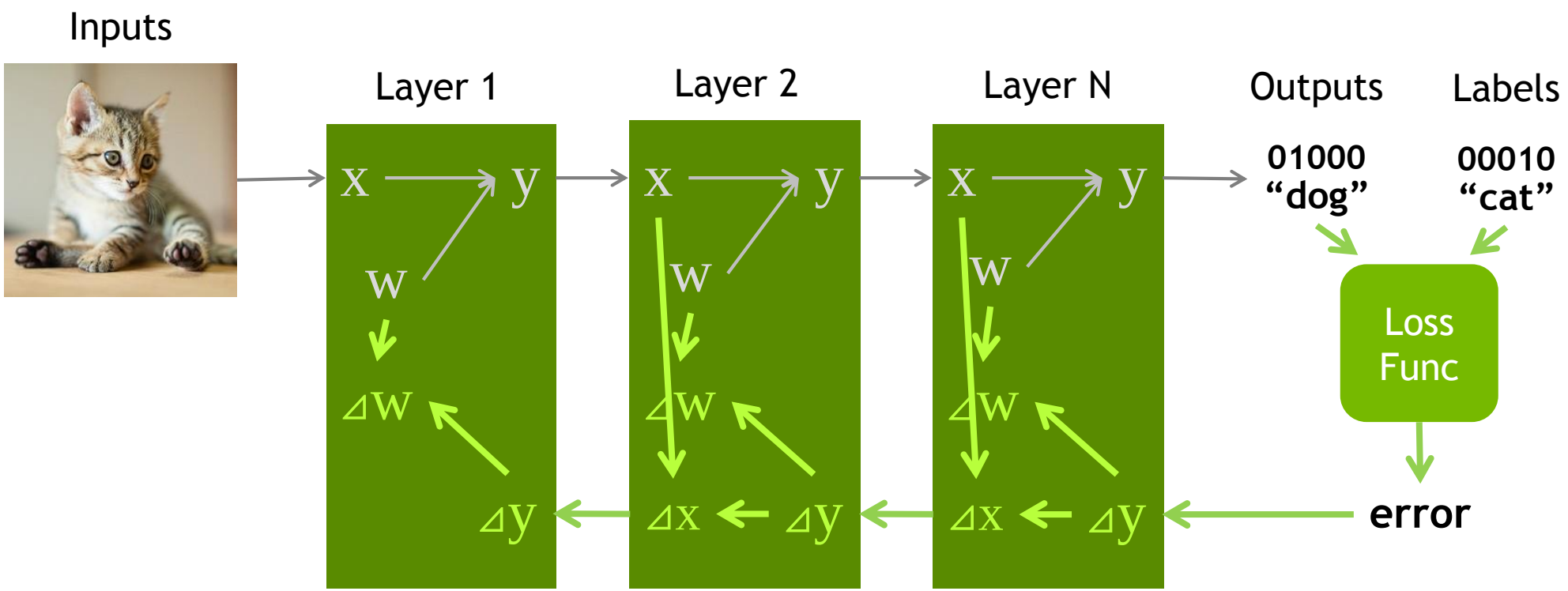
Process of training

Forward Propagation



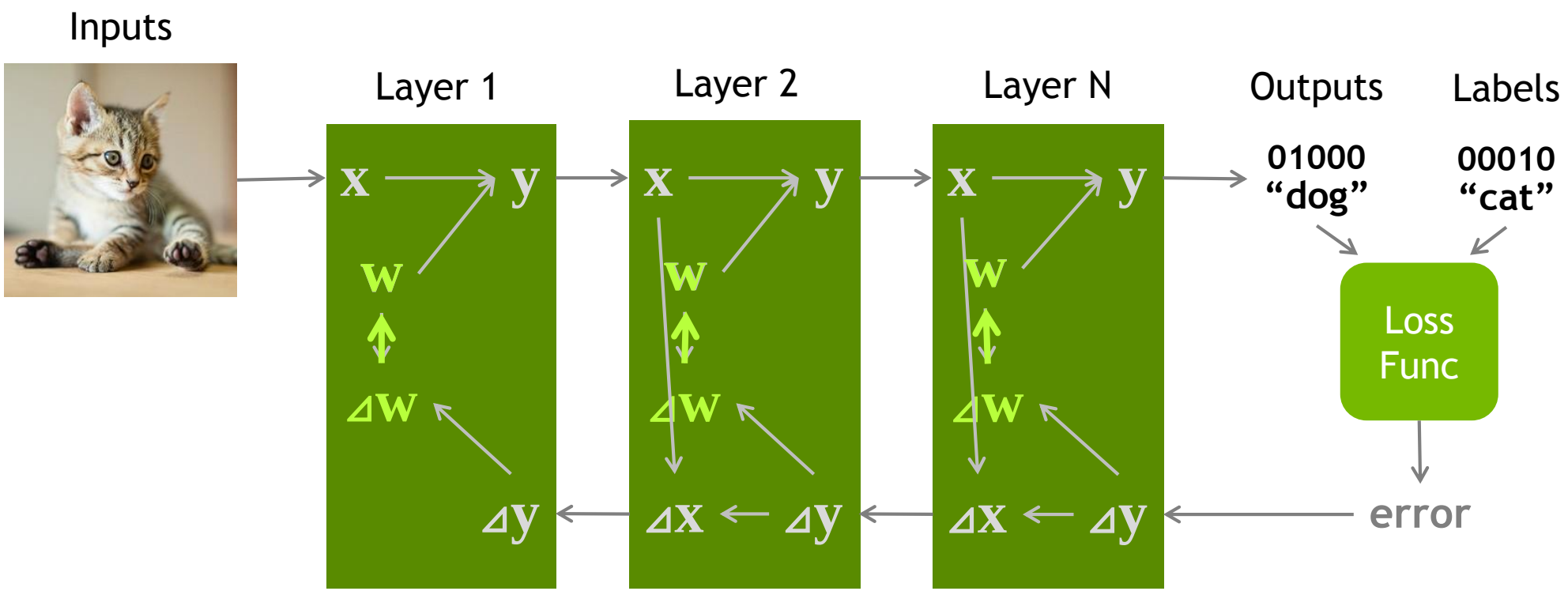
Process of training

Backward Propagation



Process of training

Update Weights

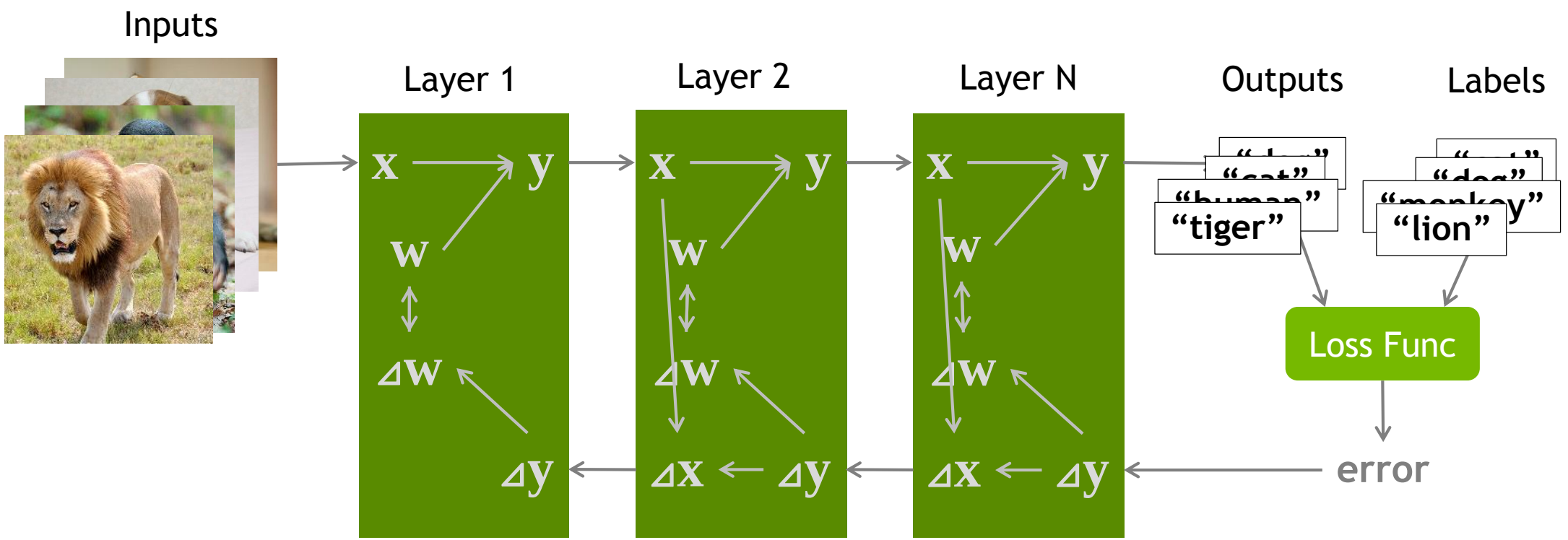


Mini Batch

Learn many inputs together for speed-up

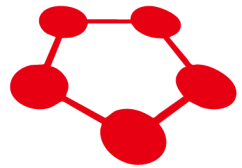
One by one: Memory bandwidth bound

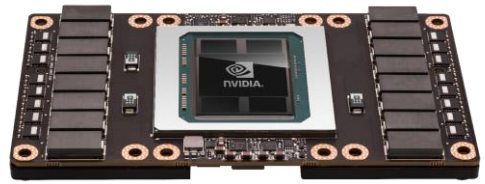
Mini batch: Compute bound



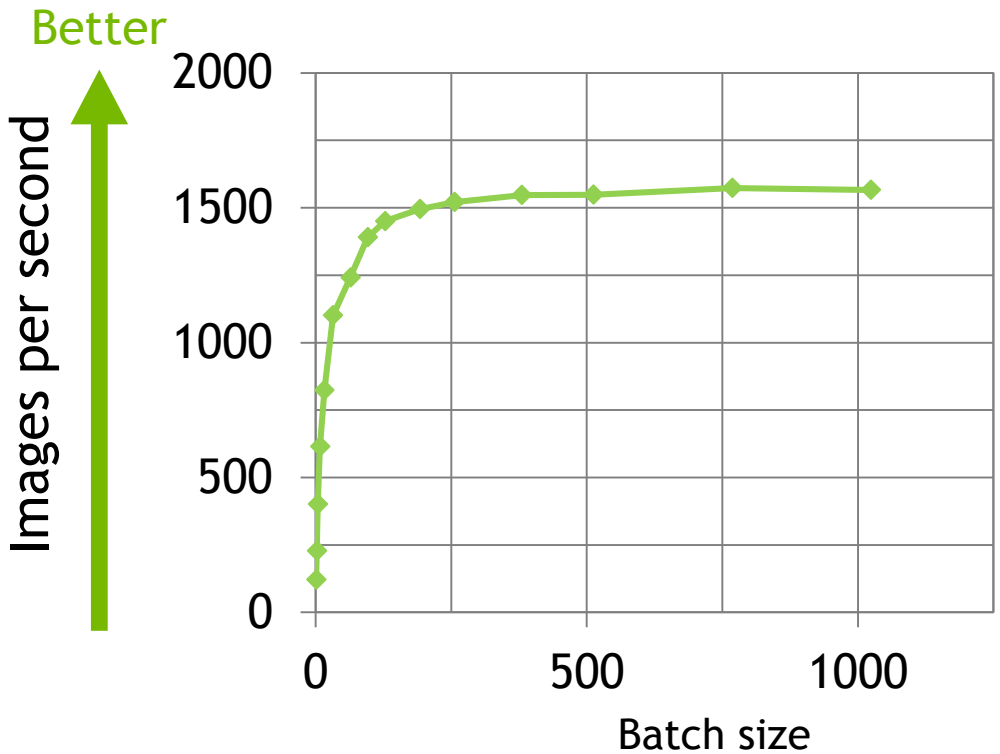
Batch Size and Performance

TESLA P100, Chainer



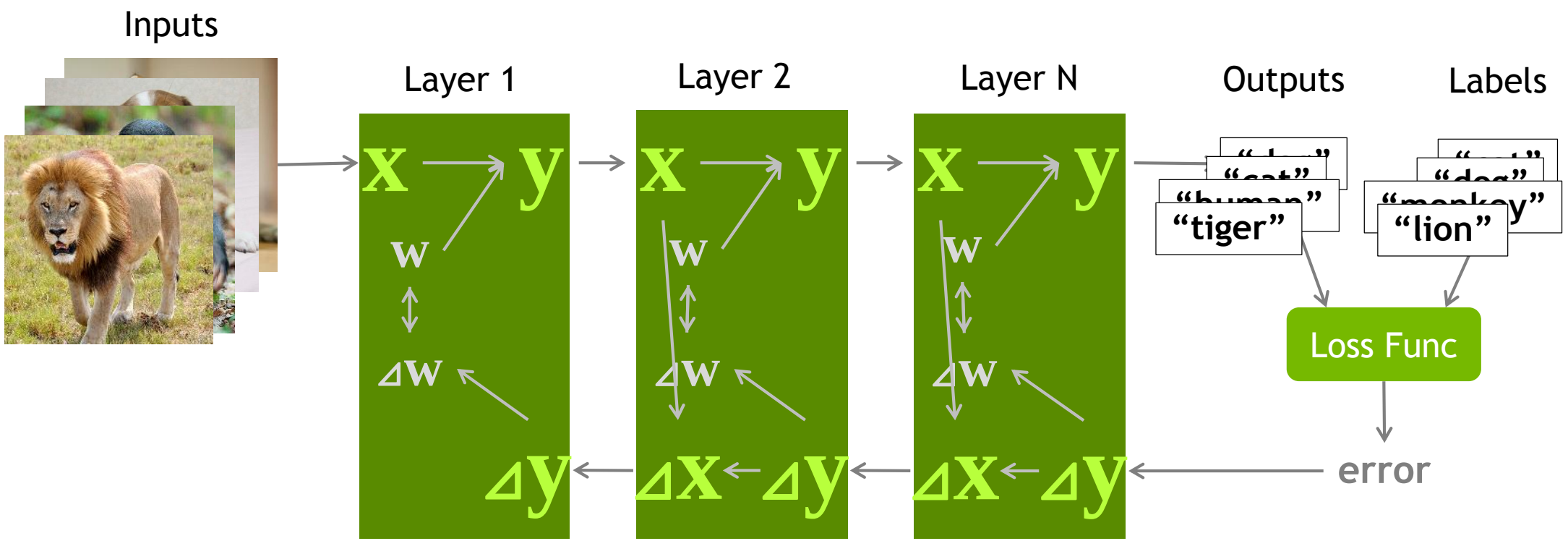


AlexNet (7 layers)
2012



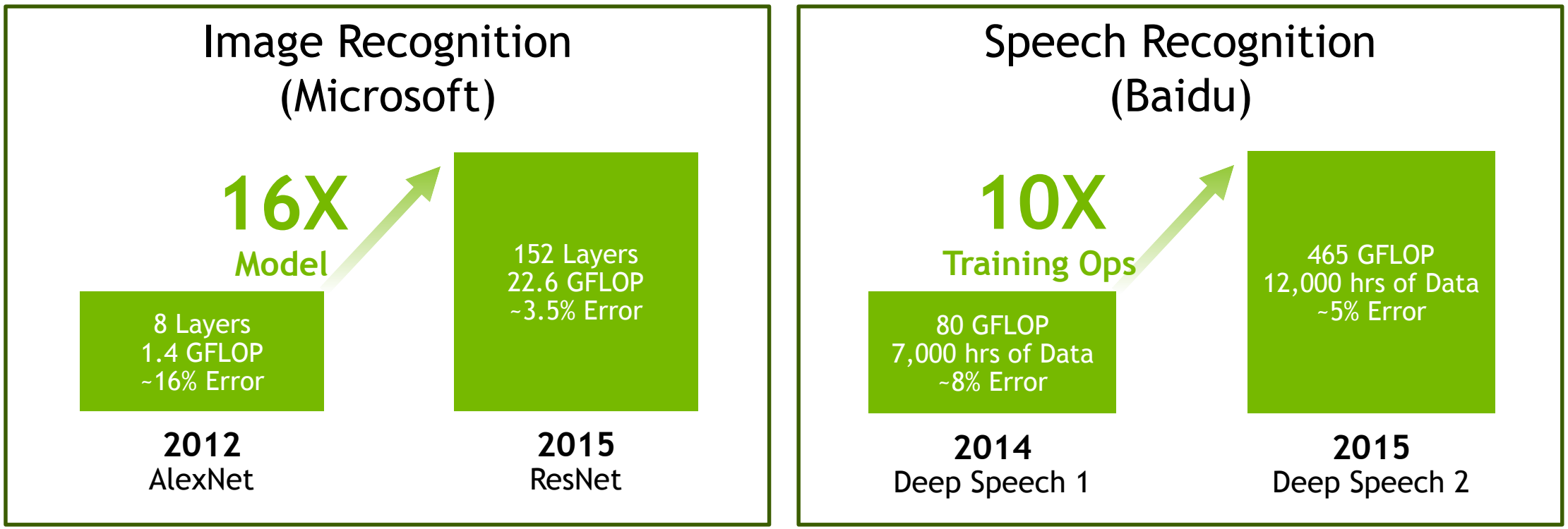
Mini Batch

More memory is necessary



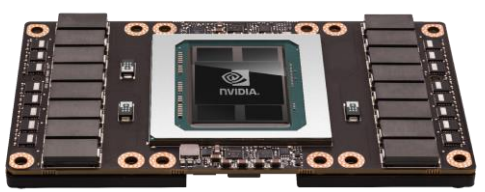
Deeper Model and Bigger Data for Higher Accuracy

More Computing Power is Required

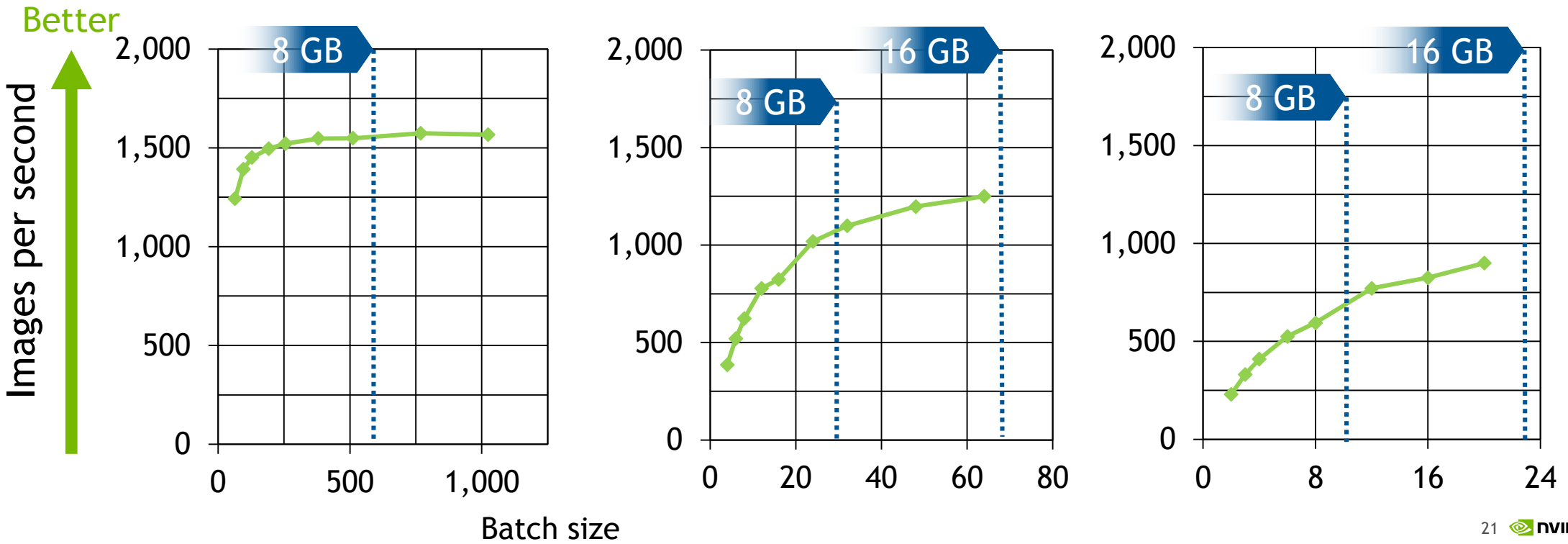


Batch Size and Performance

TESLA P100, Chainer



AlexNet (7 layers) 2012 | VGG-D (16 layers) 2014 | ResNet (152 layers) 2015



How can we accelerate DL training more?

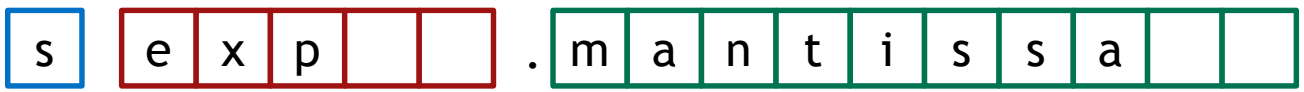
Deeper model, Bigger data

- Memory size
- Less precision
  - FP16
- Multi-GPU
  - Data Parallelism
- Out-of-core
  - Very deep model with large batch size

FP16

[TESLA P100] FP16: 16-bit floating point

IEEE 754 compliant

- 16-bit 

  - sign:1-bit, exponent:5-bits, fraction:10-bit
- Dynamic range: 2^{40}
  - Normalized values: $2^{-14} \sim 65504$ ($\approx 2^{16}$)
  - Subnormal at full speed

USE CASES

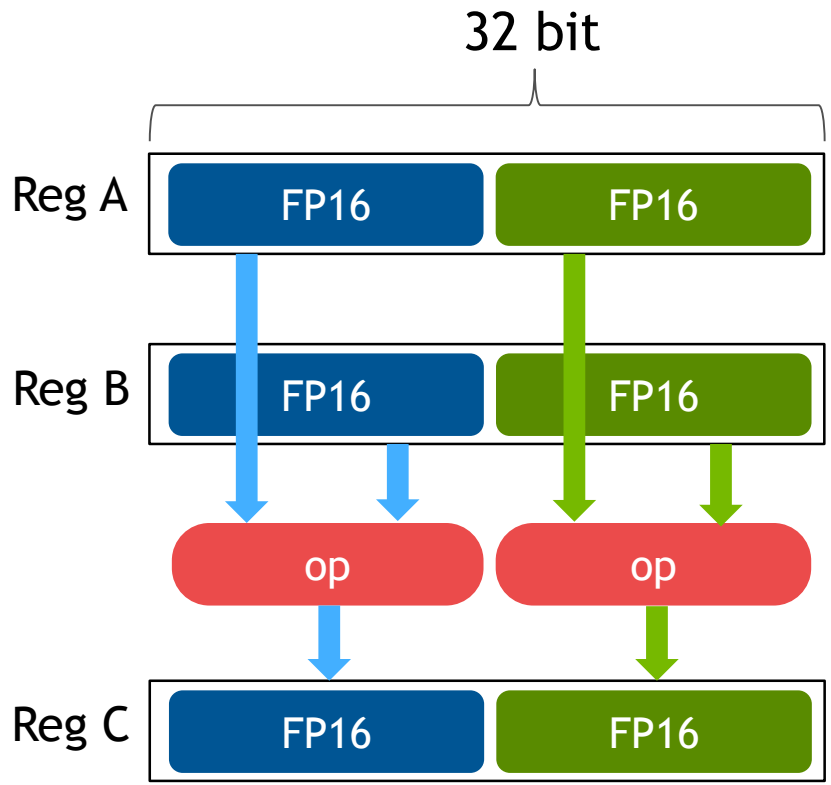
Deep Learning

Image Processing

Sensor Data

[TESLA P100] FP16: 16-bit floating point

- 2-way SIMD
  - 2x FP16 in a 32-bit register
  - **Memory footprint: Half of FP32**
- Instructions
  - Add, Sub, Mul
  - Fma (Fused Multiply and Add)
- 4 arithmetics per cycle
  - **2x faster than FP32**



cuda_fp16.h cuBLAS cuDNN

Multi-GPU training

Accelerate DL training with multi-GPU

- Model Parallelism
  - Partition model across GPUs
  - Not flexible nor scalable
- Asynchronous Data Parallelism
  - Partition data across GPUs
  - Update weights asynchronously, send its gradients to parameter server, and get the latest weights from it.
  - Parameter server is needed
- Synchronous Data Parallelism

Synchrouns Data Parallelism

Copy Model, Assigne different data

Inputs

Layer 1

Layer 2

Layer N

Labels

GPU 1

W

W

W

“cat”

LossFunc

GPU 2

W

W

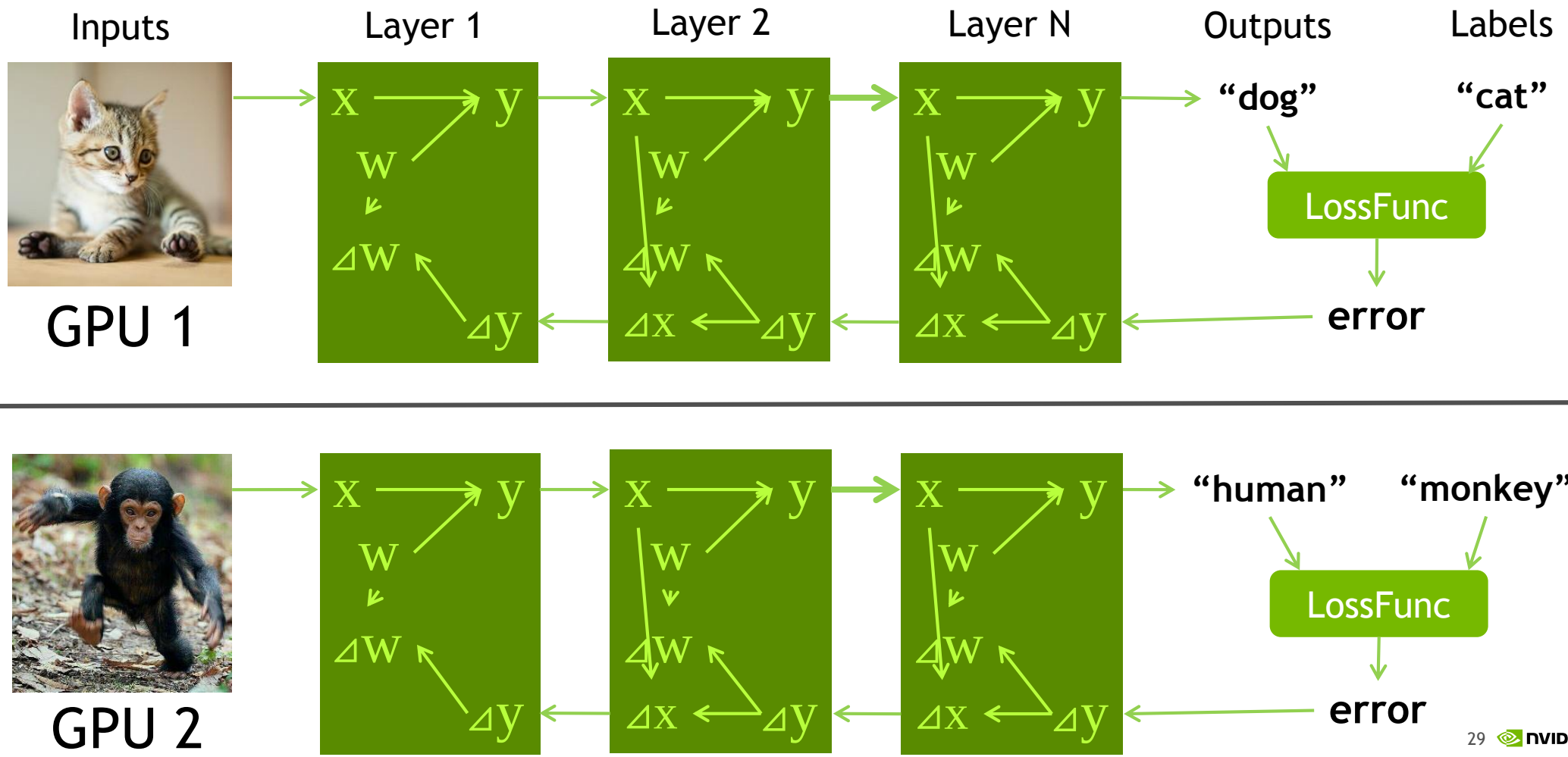
W

“monkey”

LossFunc

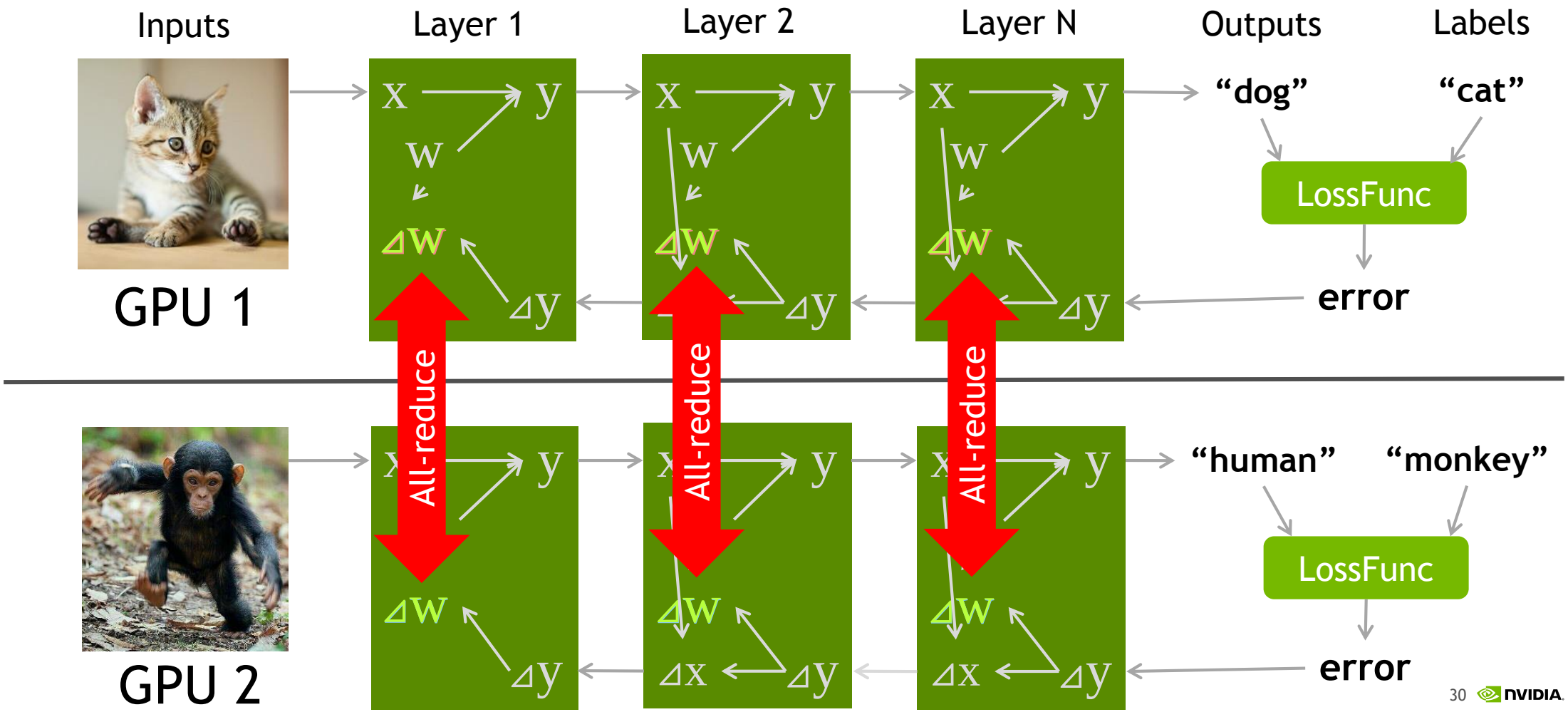
Synchrouns Data Parallelism

Forward & Backward Independently



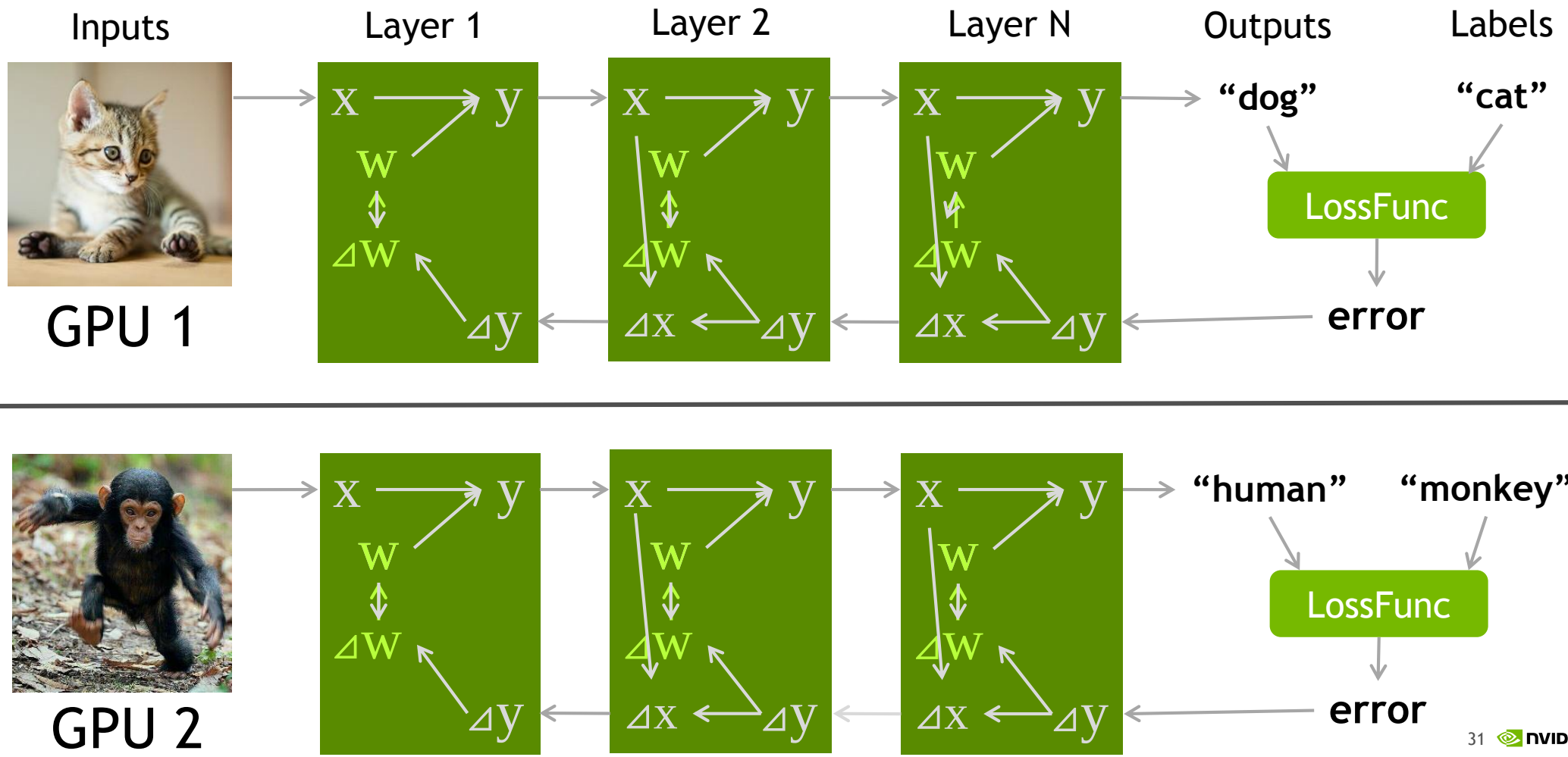
Synchrouns Data Parallelism

Combine ⊿w over multi-GPU



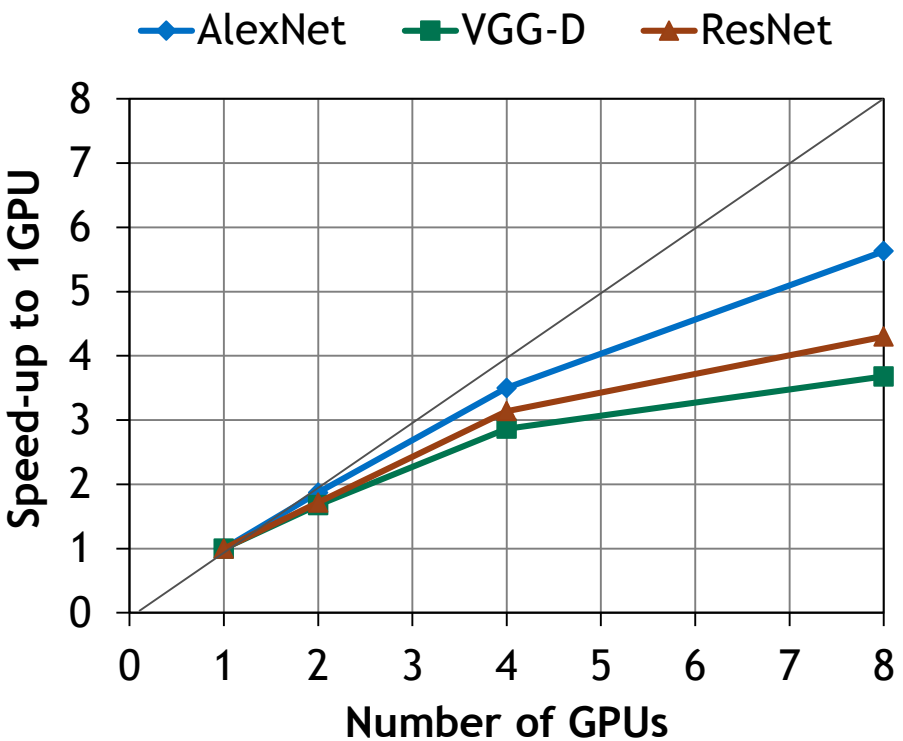
Synchrouns Data Parallelism

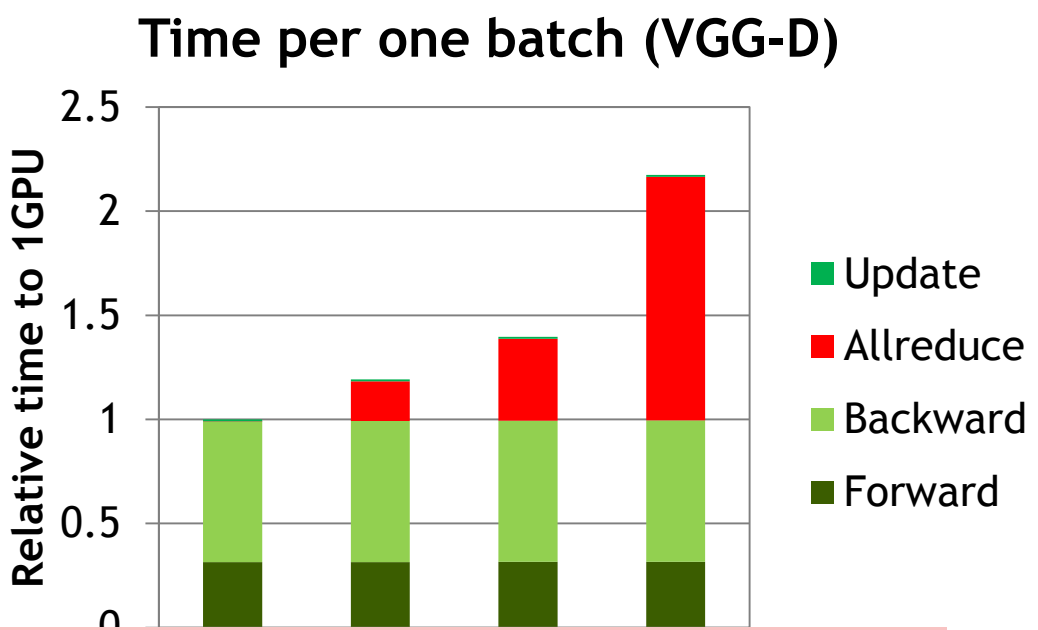
Update Weights Independently



Multi-GPU performance

NVIDIA DGX-1, Chainer with multi-process patch

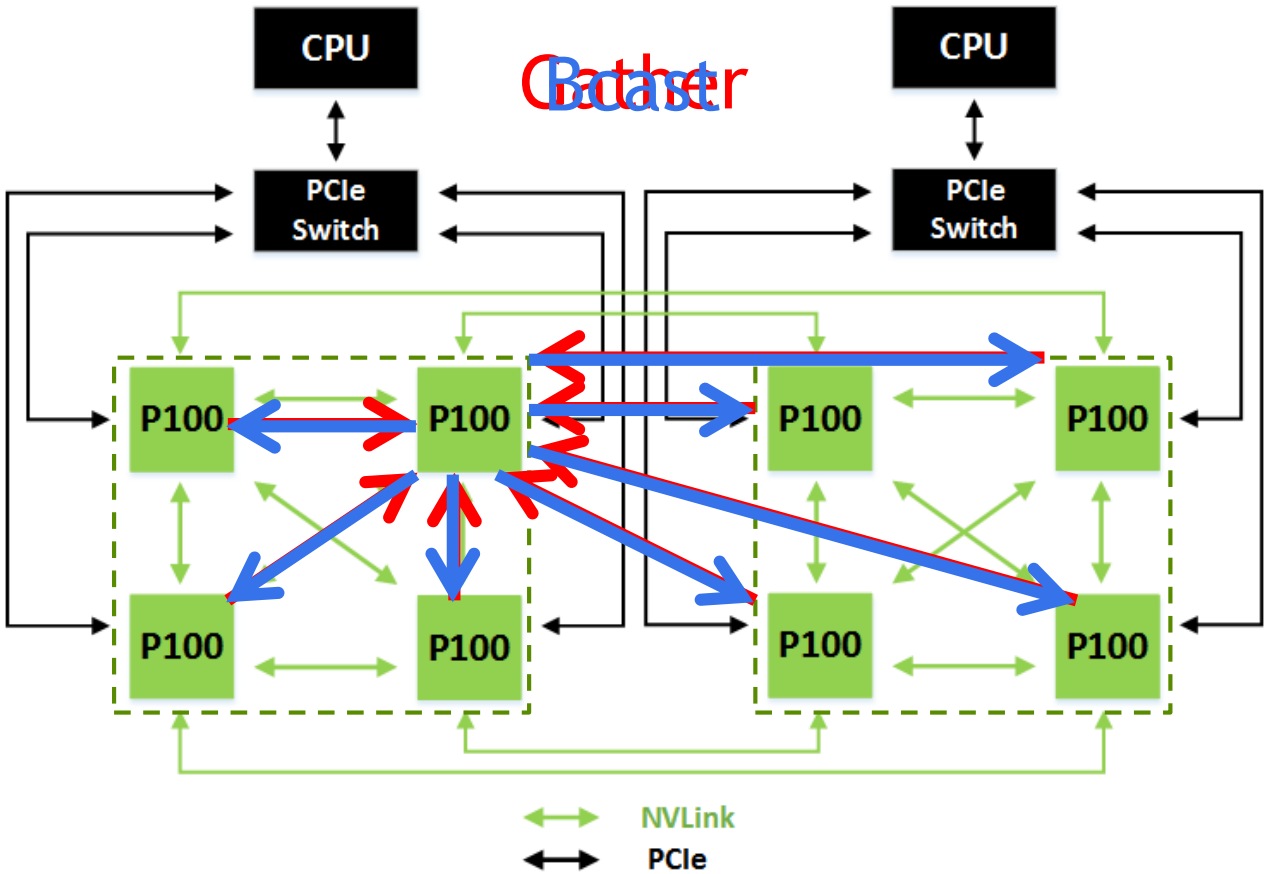




[Batch size per GPU] AlexNet:768, VGG-D:32, ResNet:12

DGX-1's NVLink is not well utilized.
Chainer's all-reduce implementation is naïve "gather and broadcat".

Allreduce: gather & broadcast



- Easy to implement, but naiva and not bandwidth optimal
- Best algorithm may change depending on topology
- Not practical for DL framework to implement several algorithms

NCCL

Optimized primitives for collective multi-GPU communication

- Accelerated collectives for multi-GPU machine
  - Similar to MPI collectives
  - Single-node, Any number of GPUs
    - PCIe, QPI and NVLINK
  - Multi-thread and Multi-process
  - Out-of-place and In-place

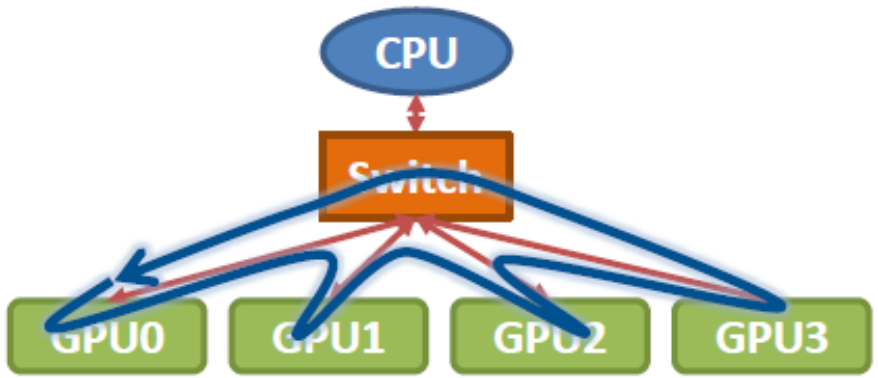
- Supported Collectives
  - Broadcast
  - All-gather
  - Reduce
  - All-reduce
  - Reduce-scatter

NCCL Implementation

Ring Algorithm

Most collectives amenable to bandwidth-optimal implementation on rings, and many topologyies can be interpreted as one or more rings [P. Patarasuk and X. Yuan]

- 1 CPU and 4 GPUs (PCIe)

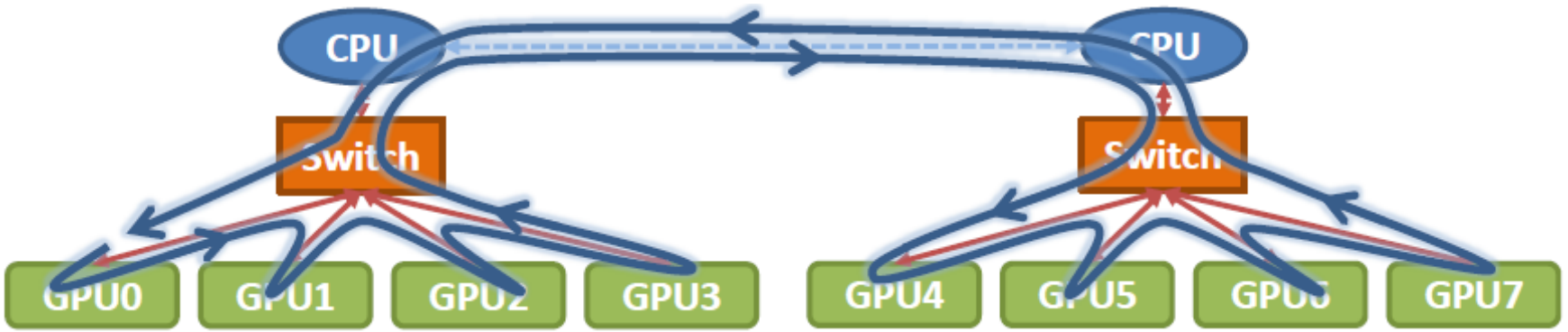


NCCL Implementation

Ring Algorithm

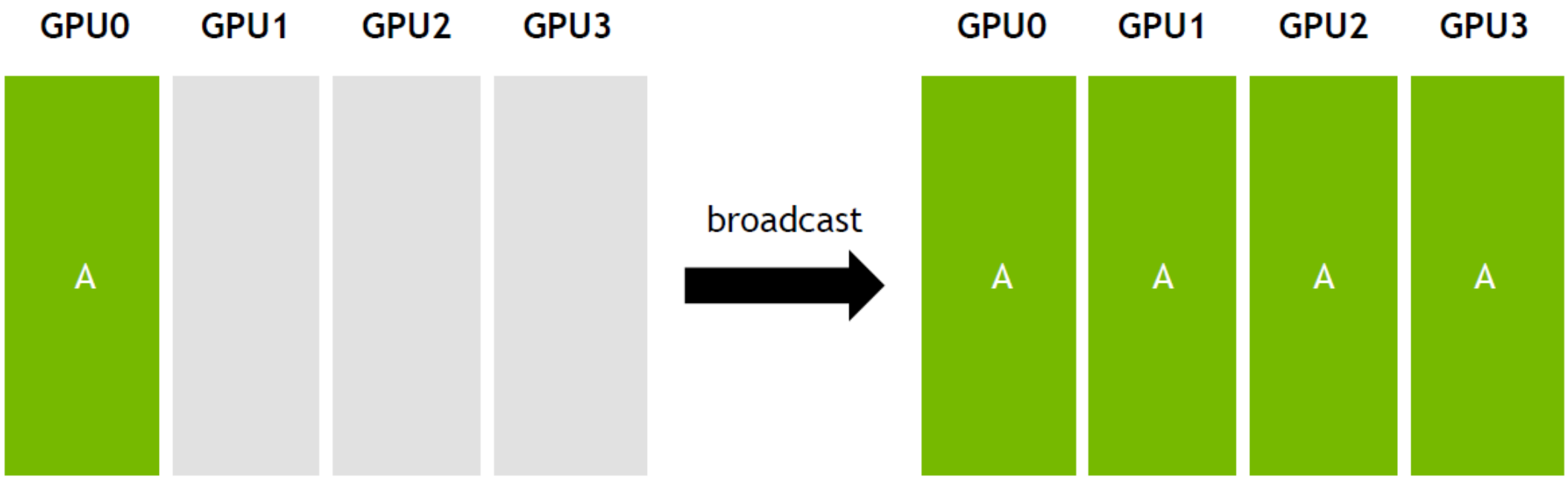
Most collectives amenable to bandwidth-optimal implementation on rings, and many topologyies can be interpreted as one or more rings [P. Patarasuk and X. Yuan]

- 2 CPUs and 8 GPUs (QPI and PCIe)



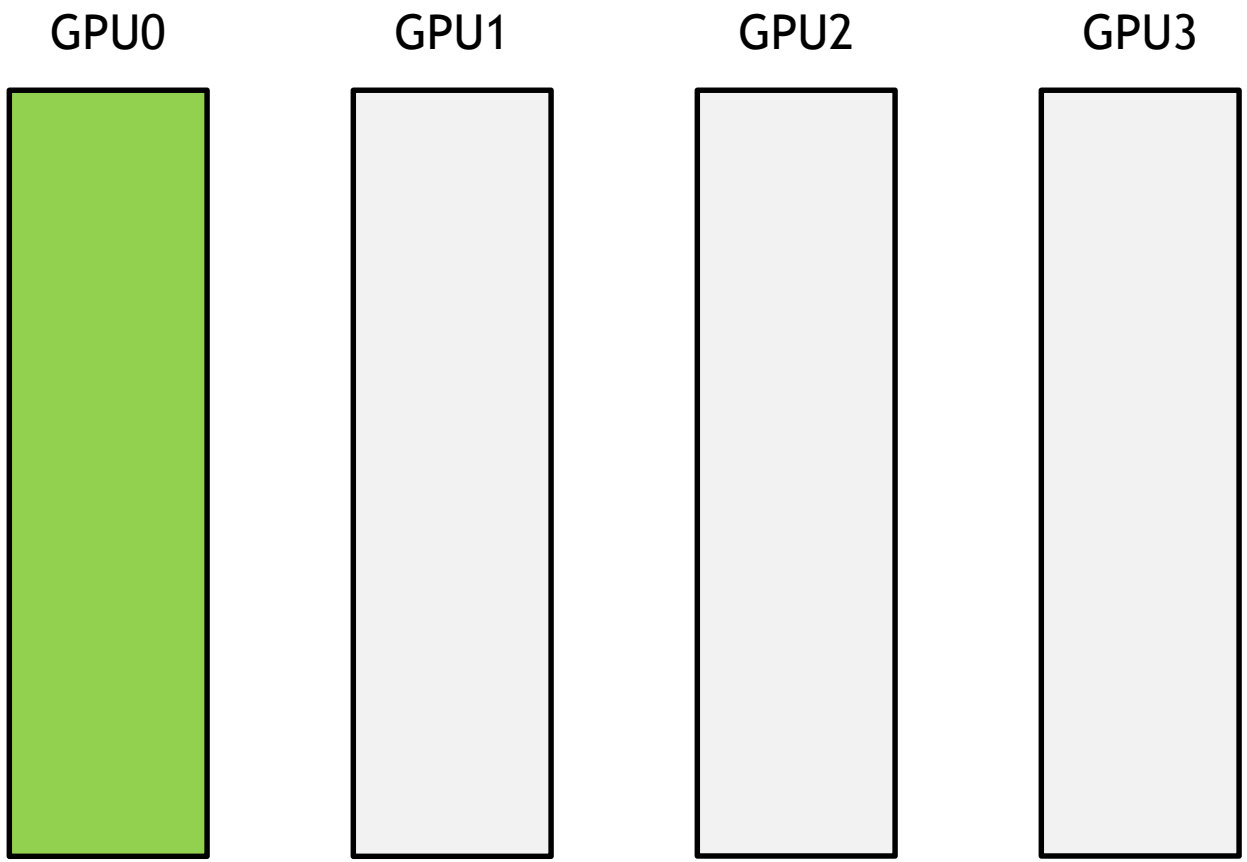
Broadcast

Deliver sender's data to all



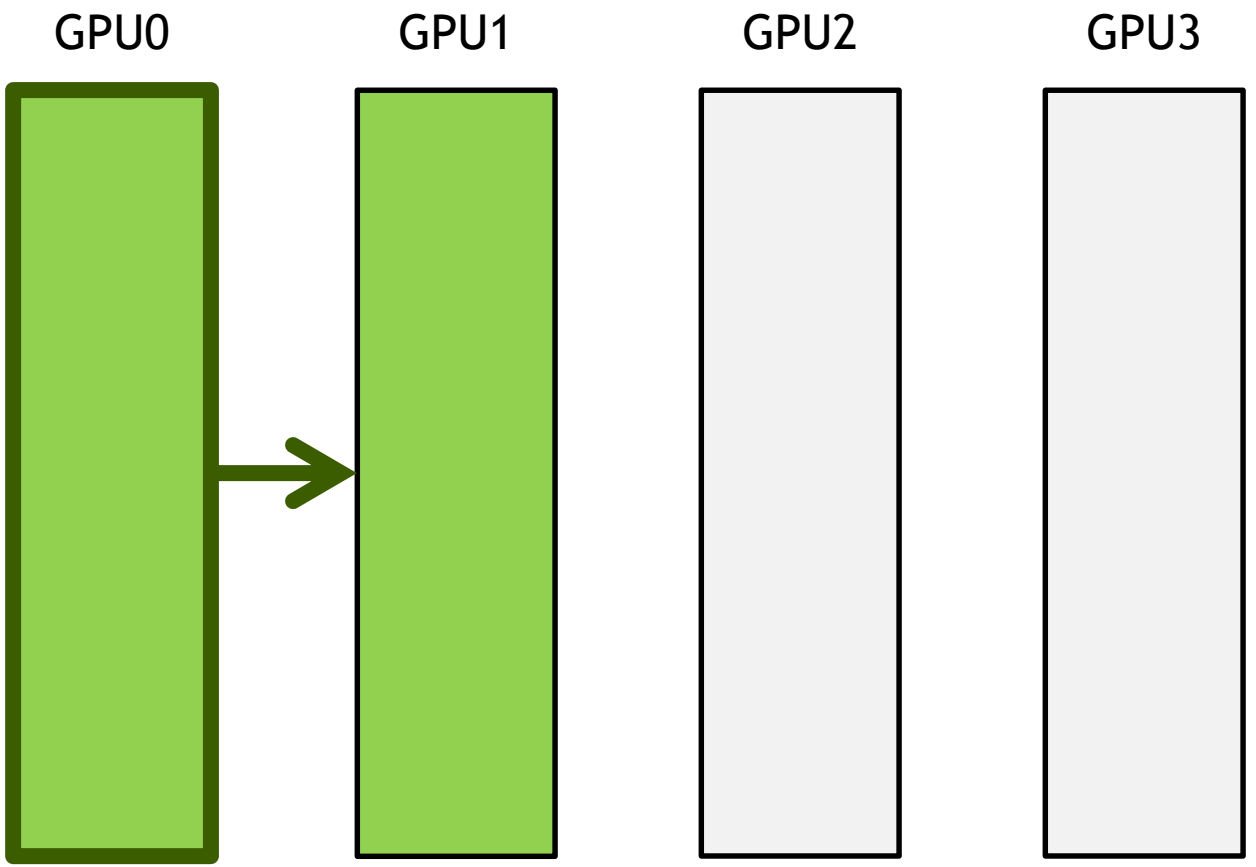
Broadcast

With unidirectional ring



Broadcast

With unidirectional ring



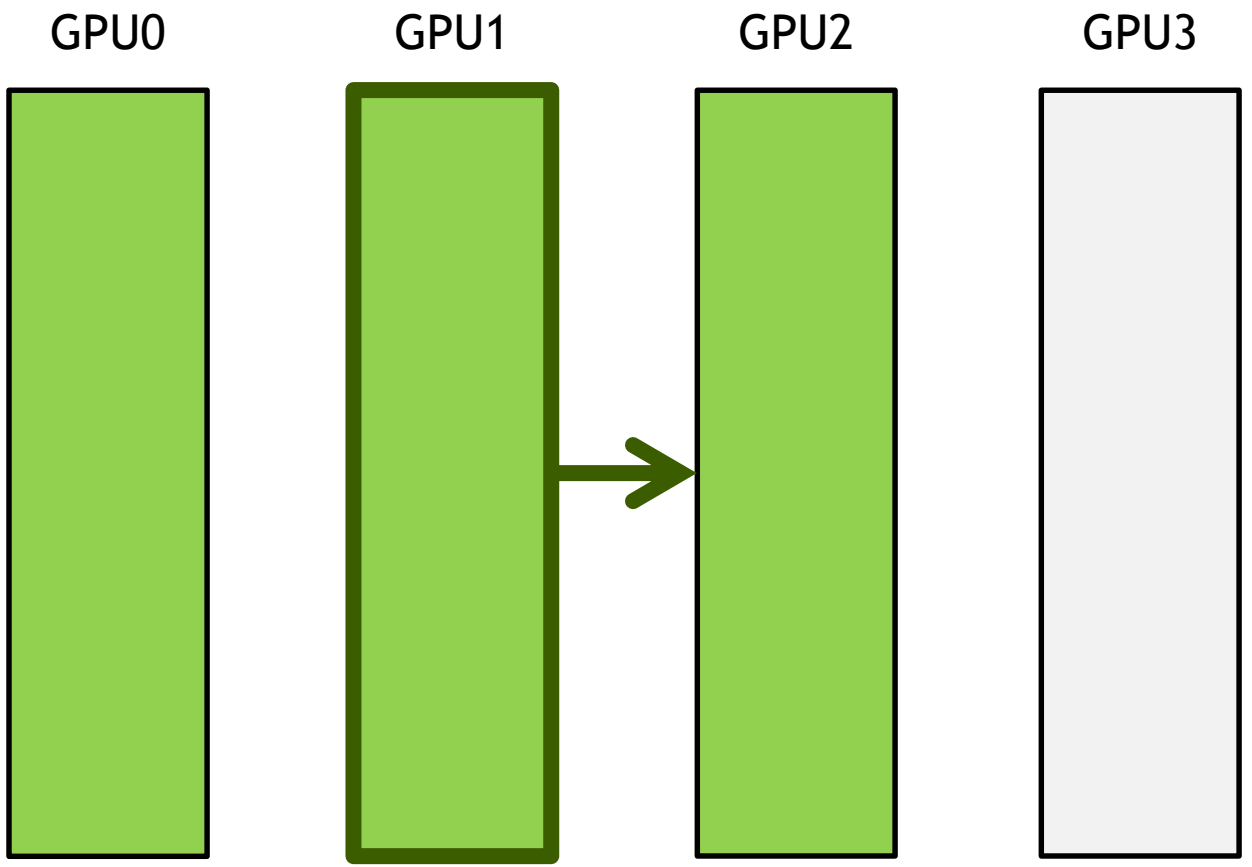
Step 1: $\Delta t = N/B$

N: bytes to broadcast

B: bandwidth of each link

Broadcast

With unidirectional ring



Step 1: $\Delta t = N/B$

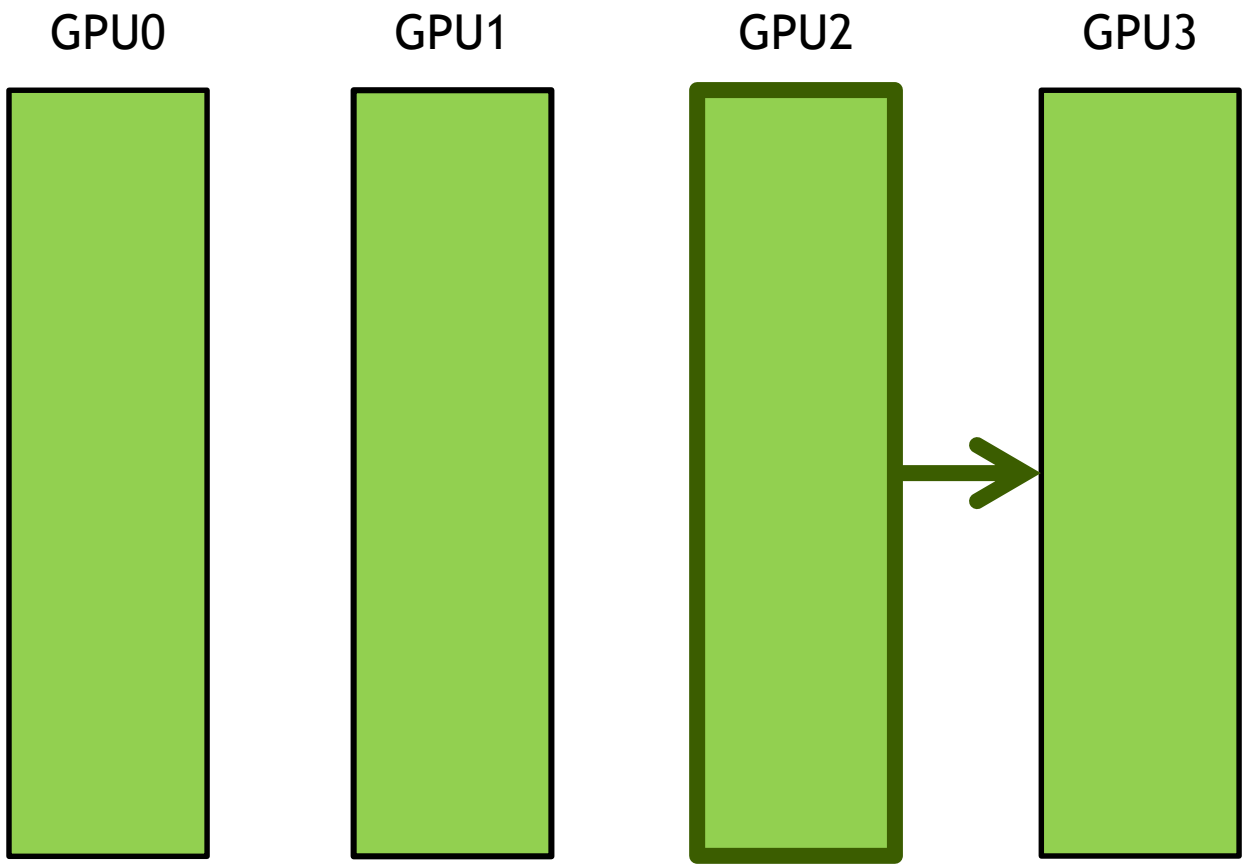
Step 2: $\Delta t = N/B$

N: bytes to broadcast

B: bandwidth of each link

Broadcast

With unidirectional ring



Step 1: $\Delta t = N/B$

Step 2: $\Delta t = N/B$

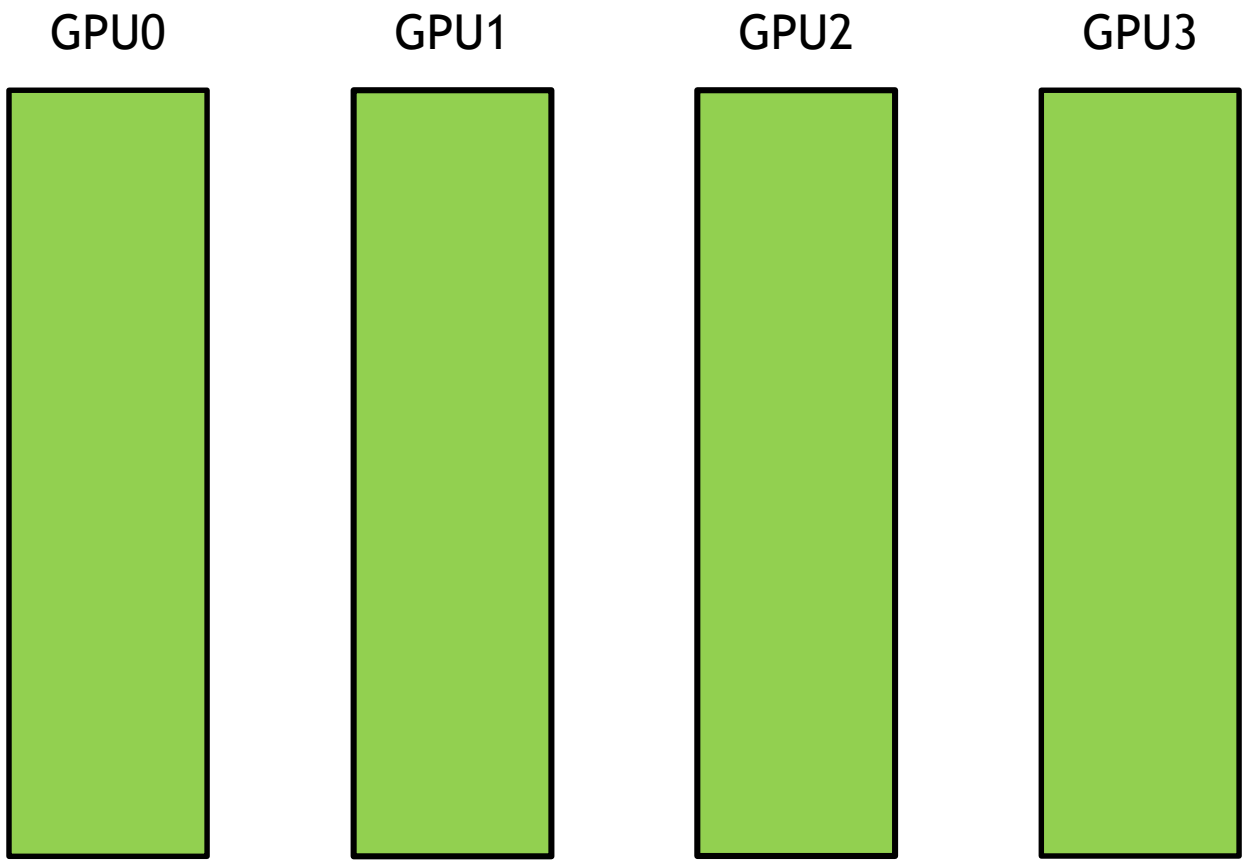
Step 3: $\Delta t = N/B$

N: bytes to broadcast

B: bandwidth of each link

Broadcast

With unidirectional ring



Step 1: $\Delta t = N/B$

Step 2: $\Delta t = N/B$

Step 3: $\Delta t = N/B$

Total time: $(K-1)N/B$

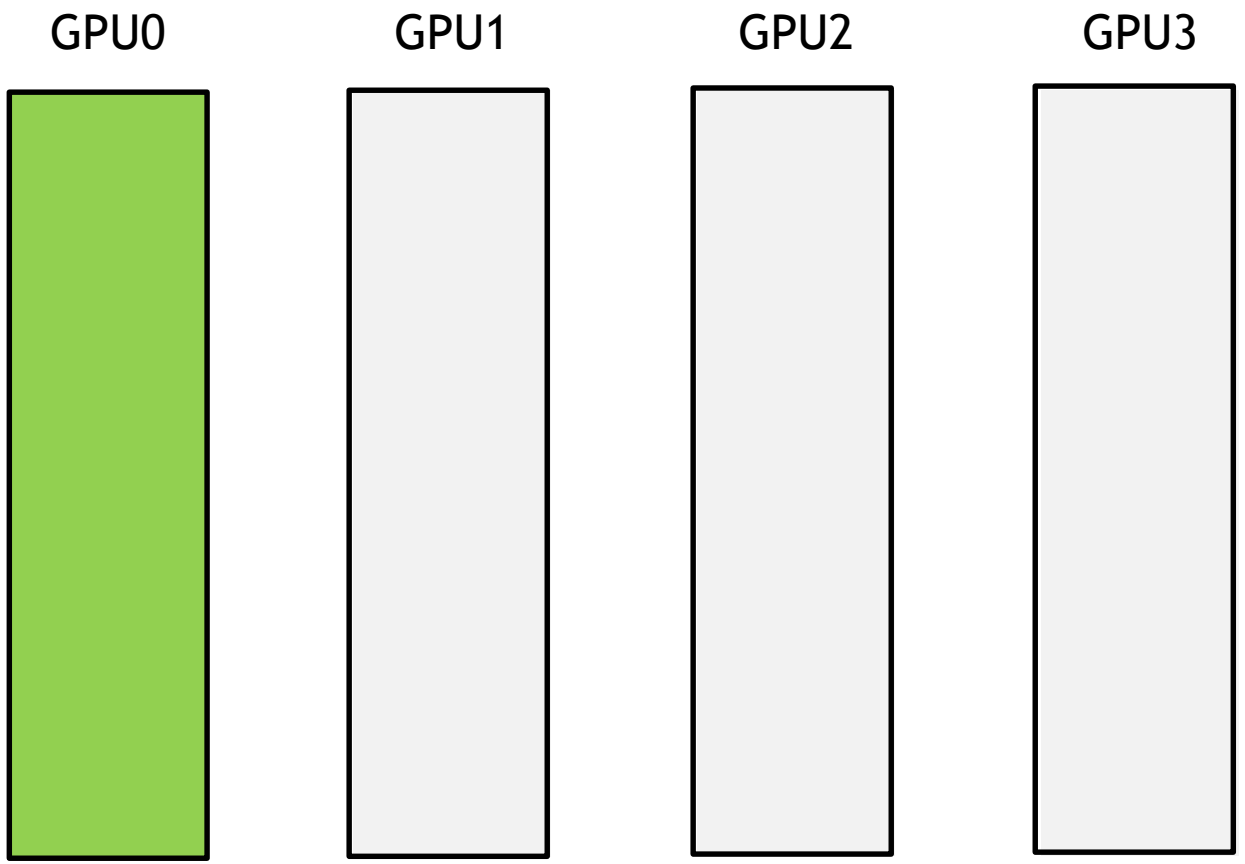
N: bytes to broadcast

B: bandwidth of each link

K: number of GPUs

Broadcast

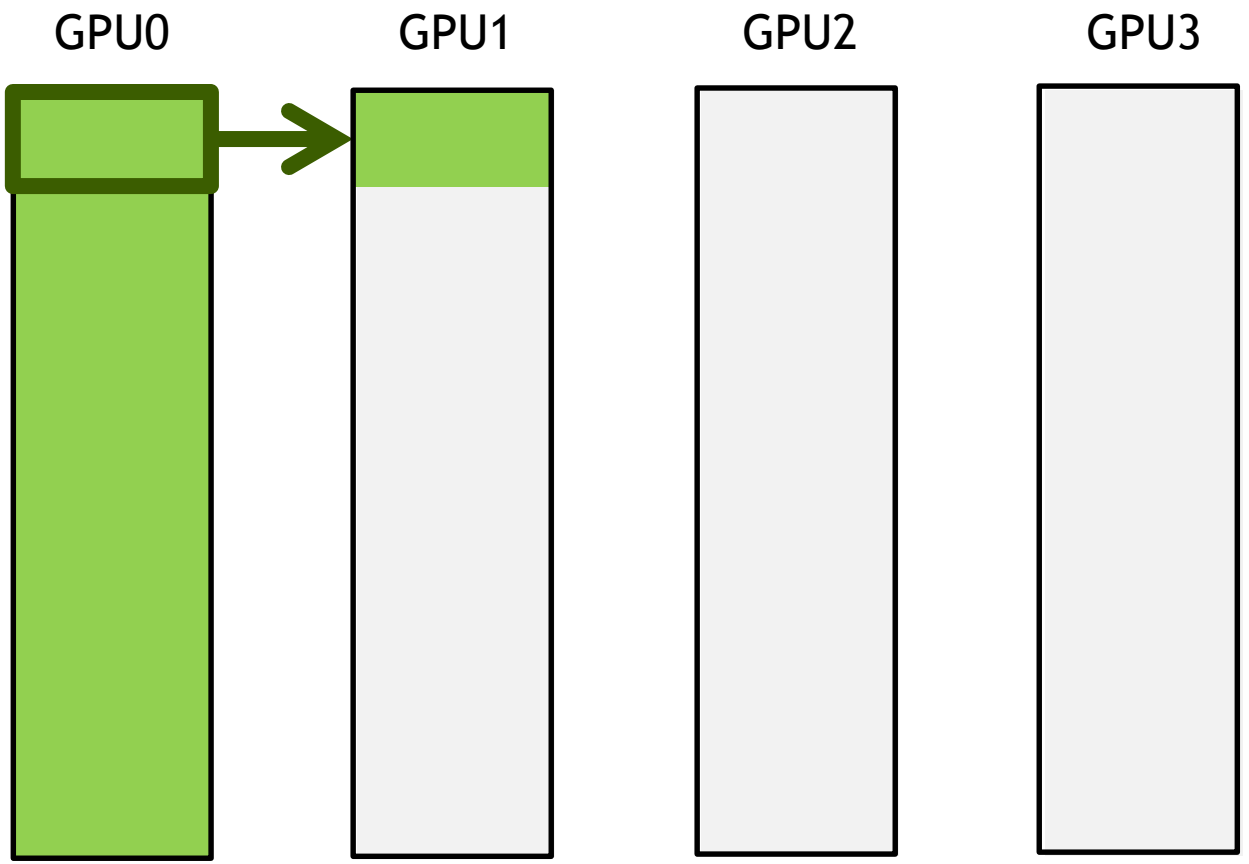
With unidirectional ring



Split data into S messages

Broadcast

With unidirectional ring

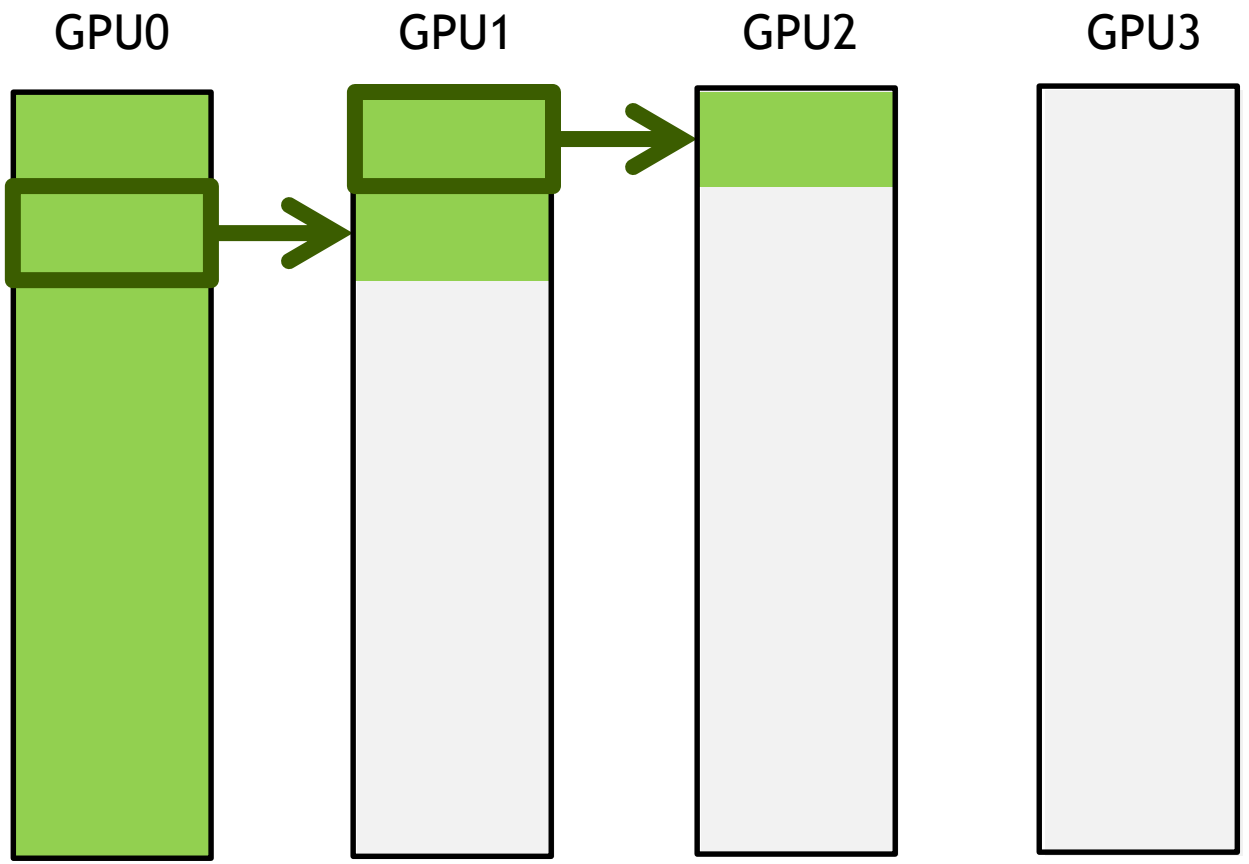


Split data into S messages

Step 1: $\Delta t = N/BS$

Broadcast

With unidirectional ring



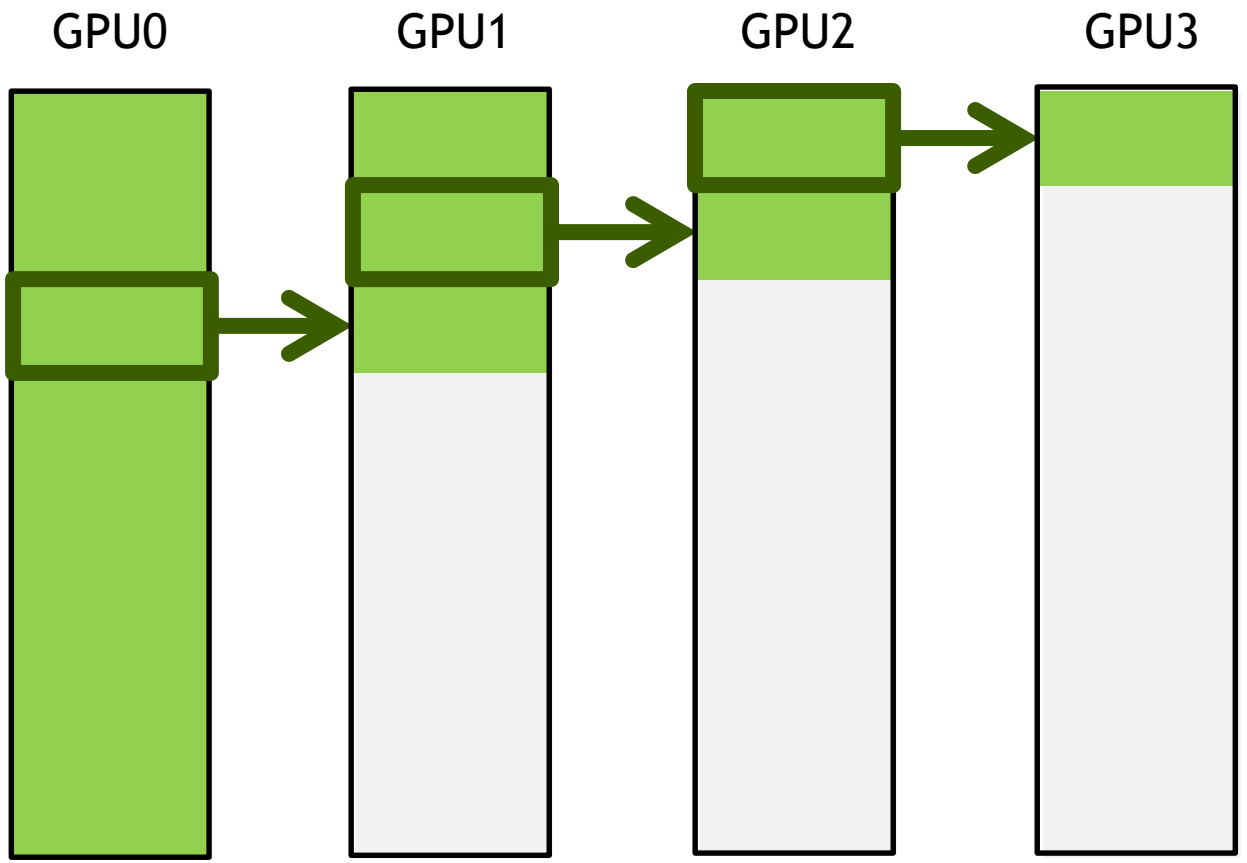
Split data into S messages

Step 1: $\Delta t = N/BS$

Step 2: $\Delta t = N/BS$

Broadcast

With unidirectional ring



Split data into S messages

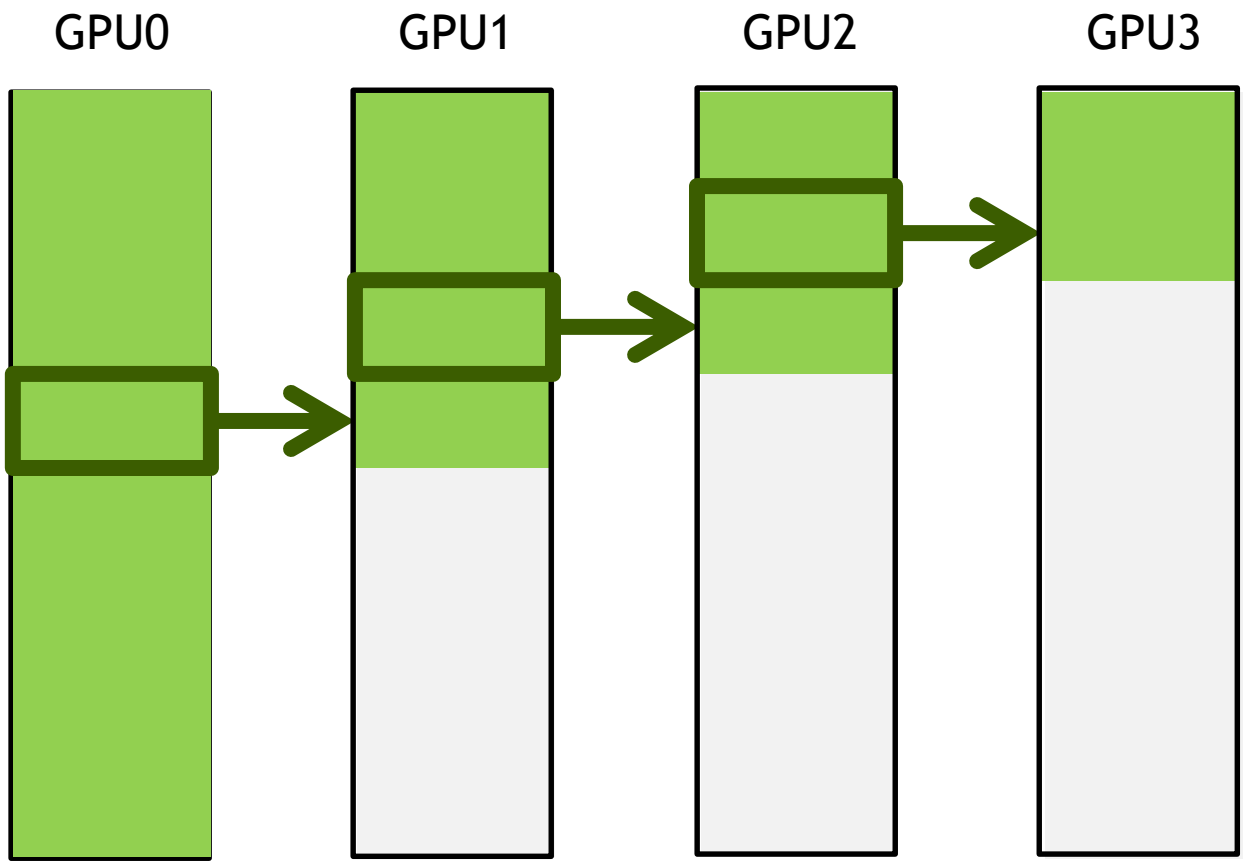
Step 1: $\Delta t = N/BS$

Step 2: $\Delta t = N/BS$

Step 3: $\Delta t = N/BS$

Broadcast

With unidirectional ring



Split data into S messages

Step 1: $\Delta t = N/BS$

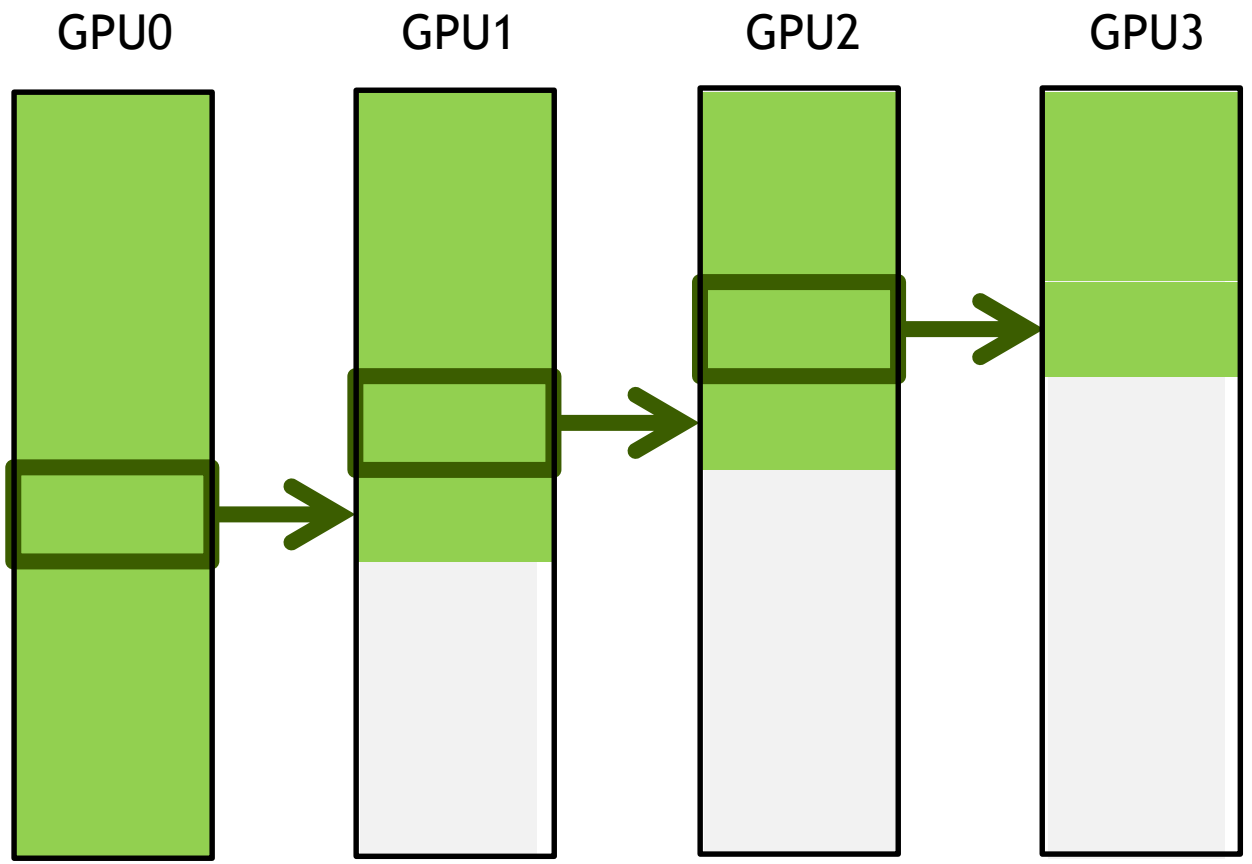
Step 2: $\Delta t = N/BS$

Step 3: $\Delta t = N/BS$

Step 4: $\Delta t = N/BS$

Broadcast

With unidirectional ring



Split data into S messages

Step 1: $\Delta t = N/BS$

Step 2: $\Delta t = N/BS$

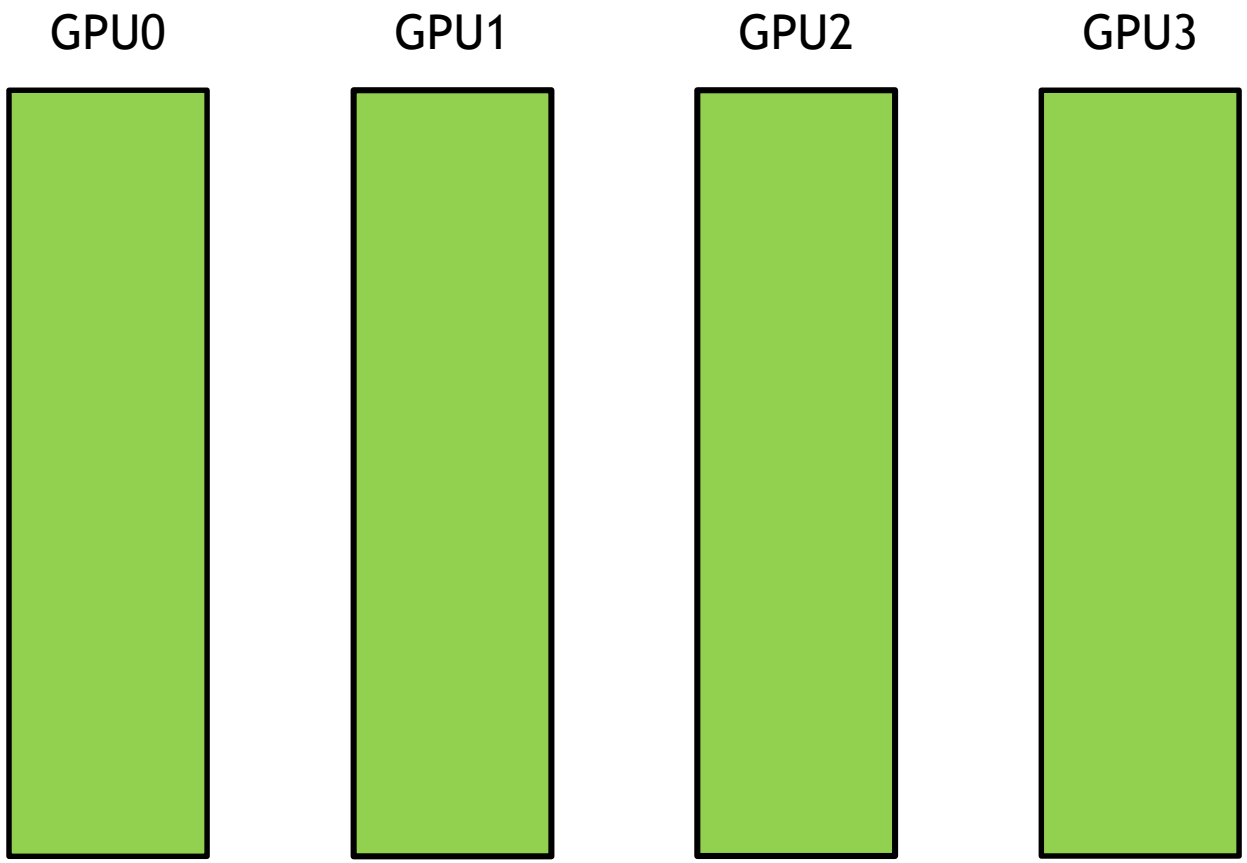
Step 3: $\Delta t = N/BS$

Step 4: $\Delta t = N/BS$

Step 5: $\Delta t = N/BS$

Broadcast

With unidirectional ring



Split data into S messages

Step 1: $\Delta t = N/BS$

Step 2: $\Delta t = N/BS$

Step 3: $\Delta t = N/BS$

Step 4: $\Delta t = N/BS$

Step 4: $\Delta t = N/BS$

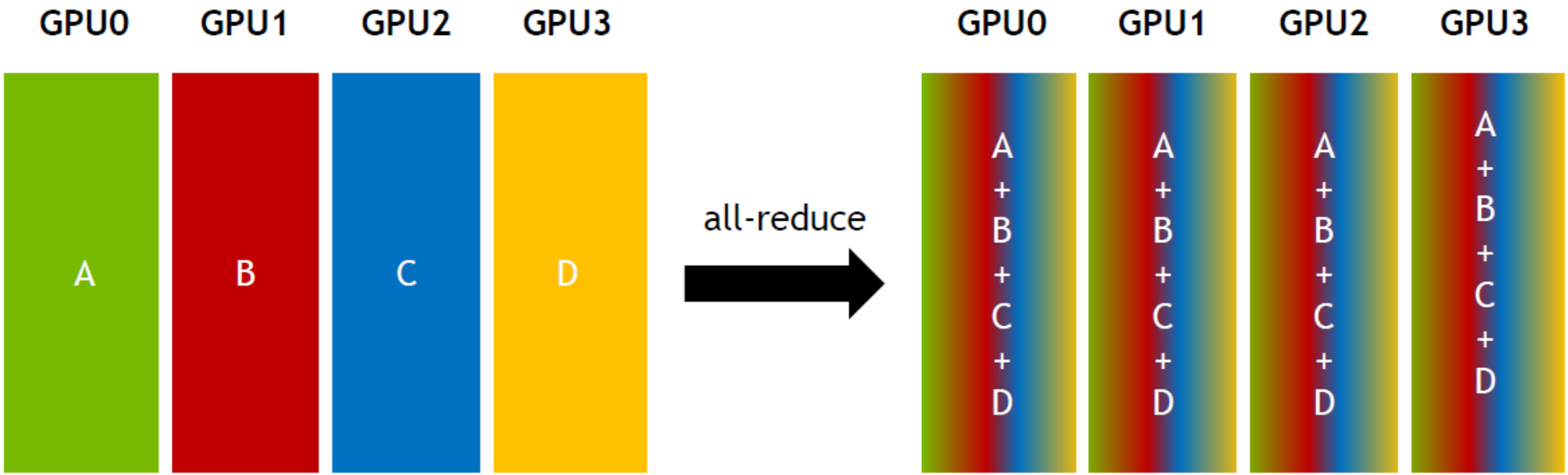
...

Total time:

$(S+k-2)N/BS \rightarrow N/B$

All-reduce

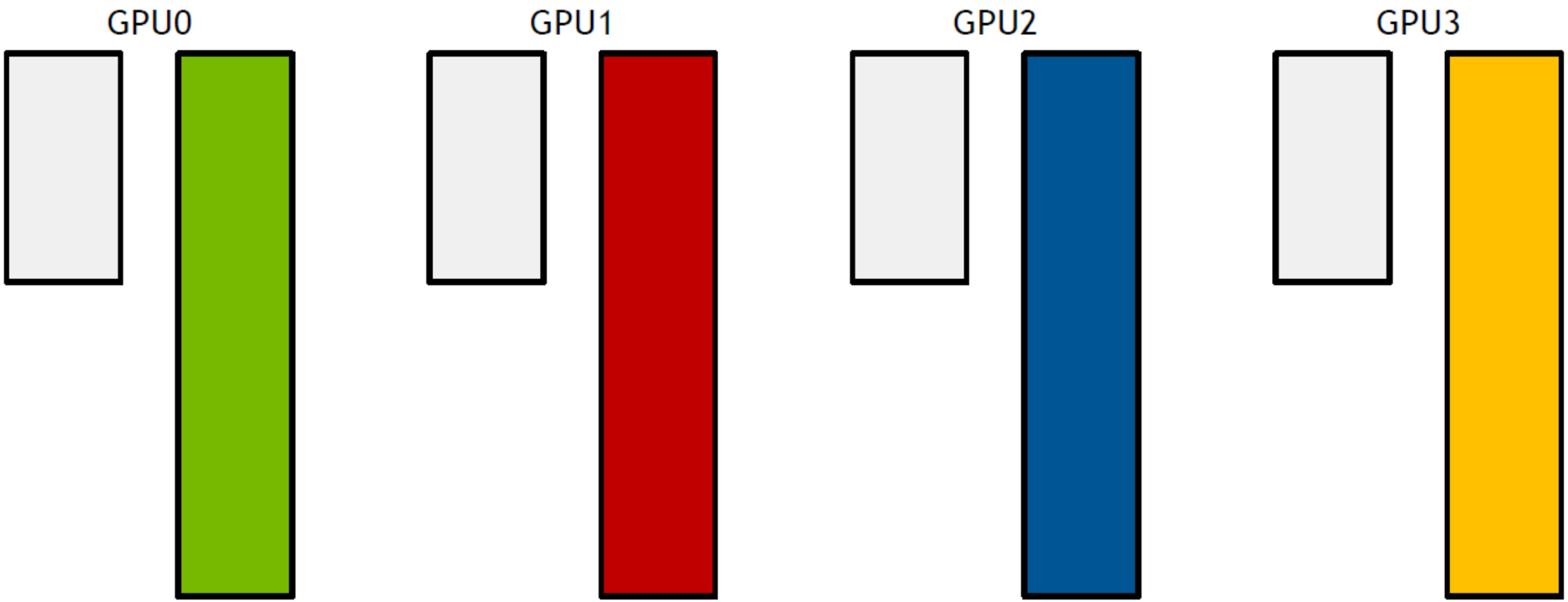
Combine data from all, deliver the result to all



All-reduce

With unidirectional ring

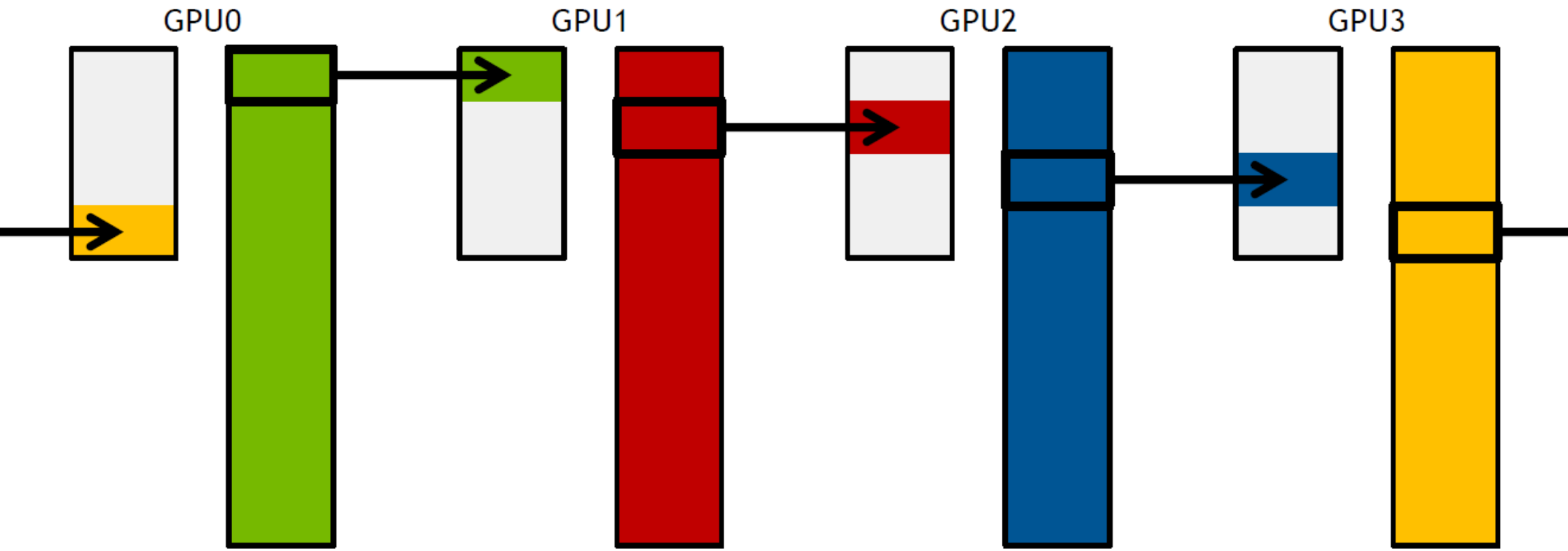
Chunk: 1
Step:



All-reduce

With unidirectional ring

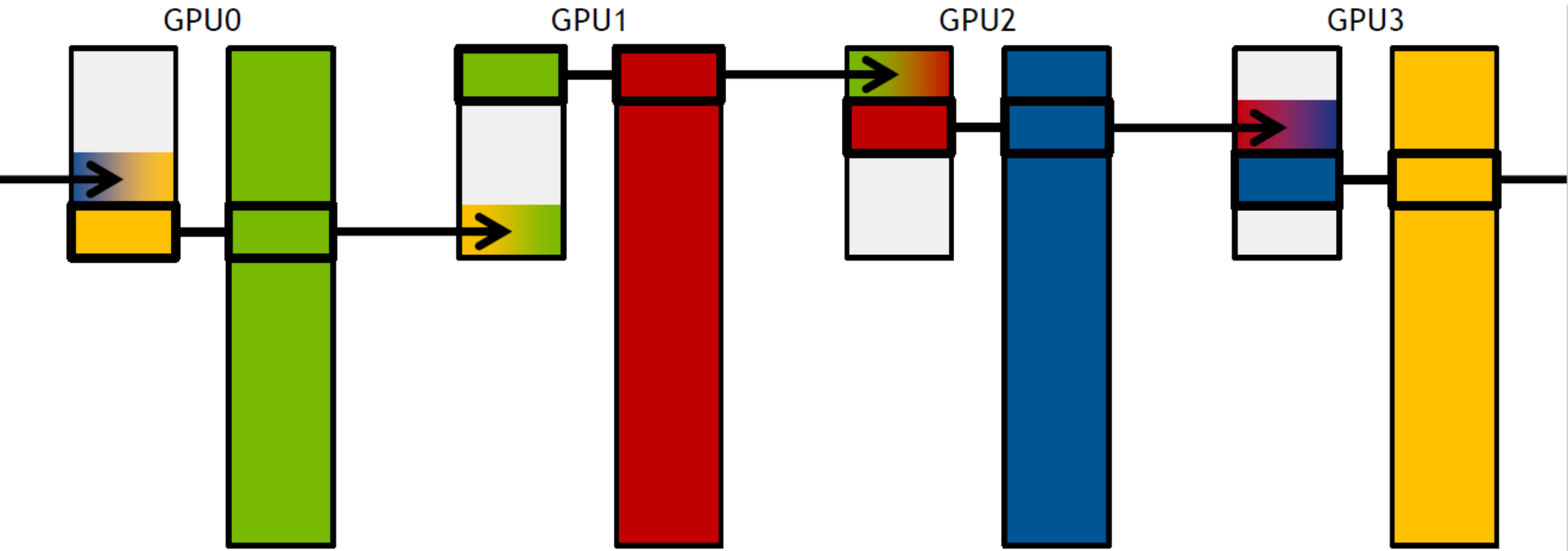
Chunk: 1
Step: 1



All-reduce

With unidirectional ring

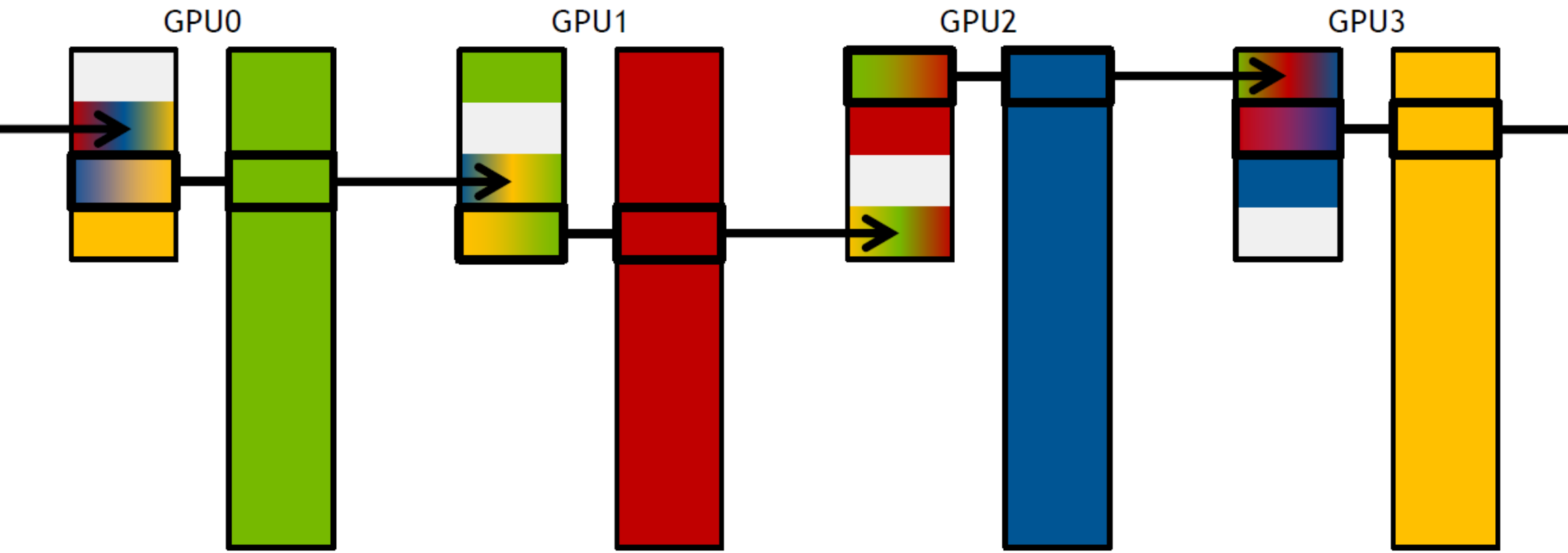
Chunk: 1
Step: 2



All-reduce

With unidirectional ring

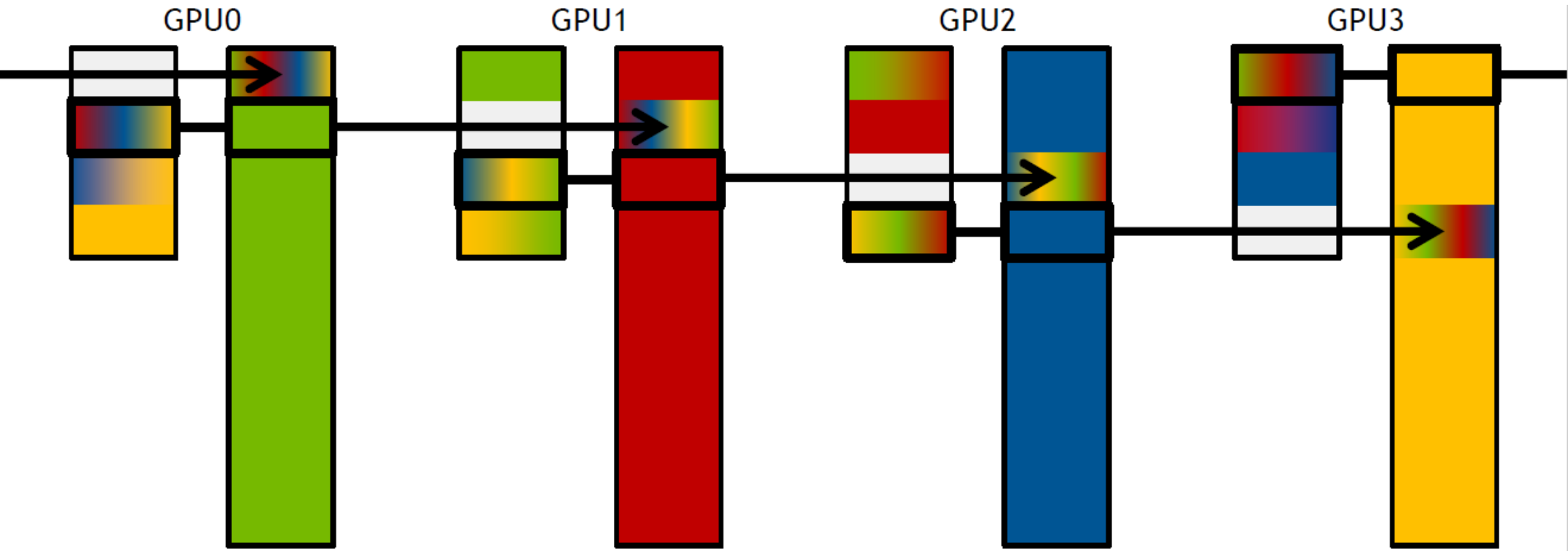
Chunk: 1
Step: 3



All-reduce

With unidirectional ring

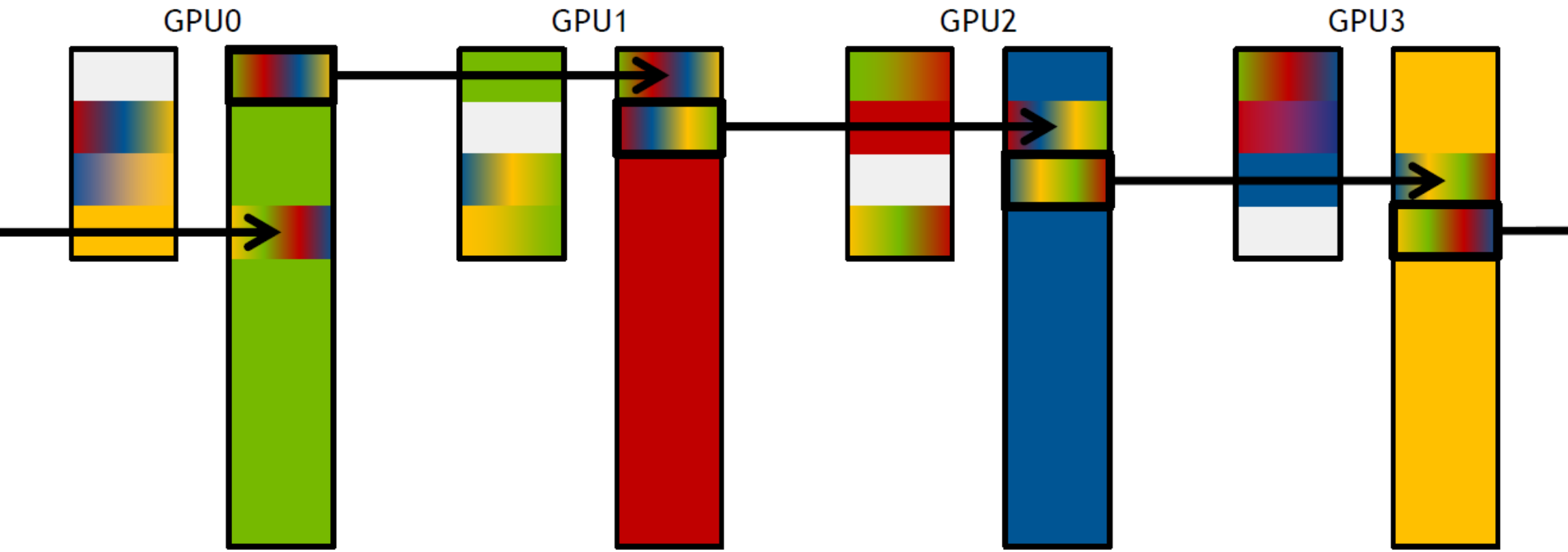
Chunk: 1
Step: 4



All-reduce

With unidirectional ring

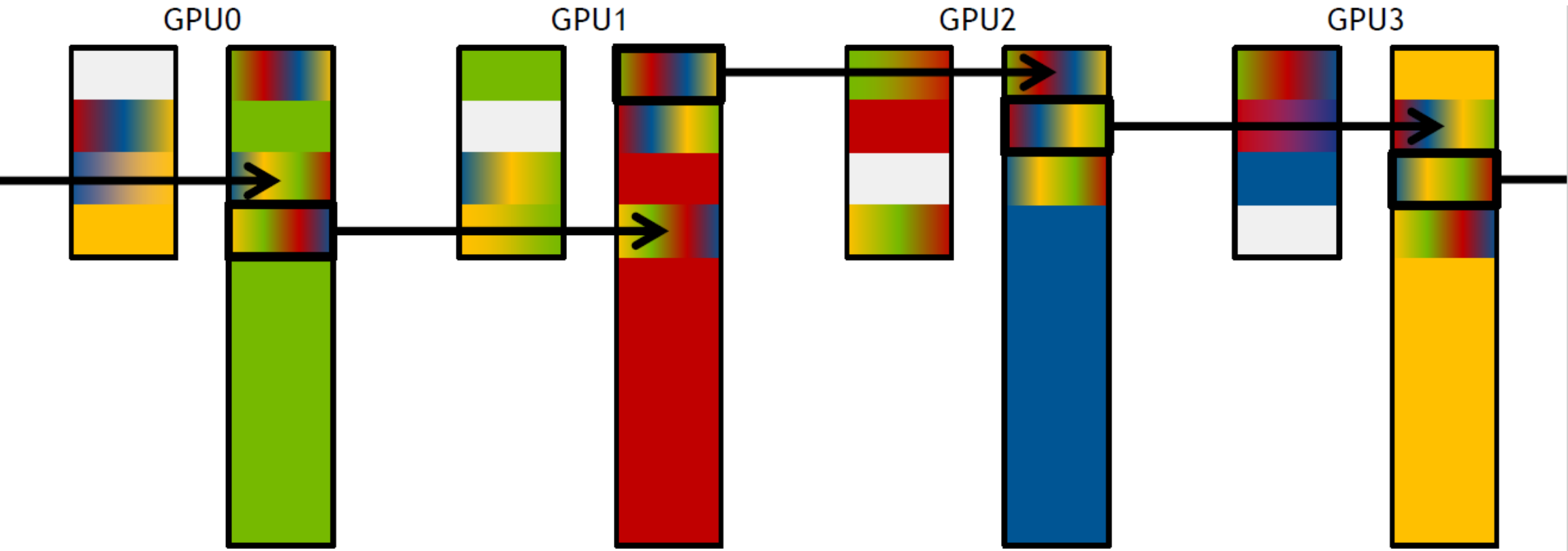
Chunk: 1
Step: 5



All-reduce

With unidirectional ring

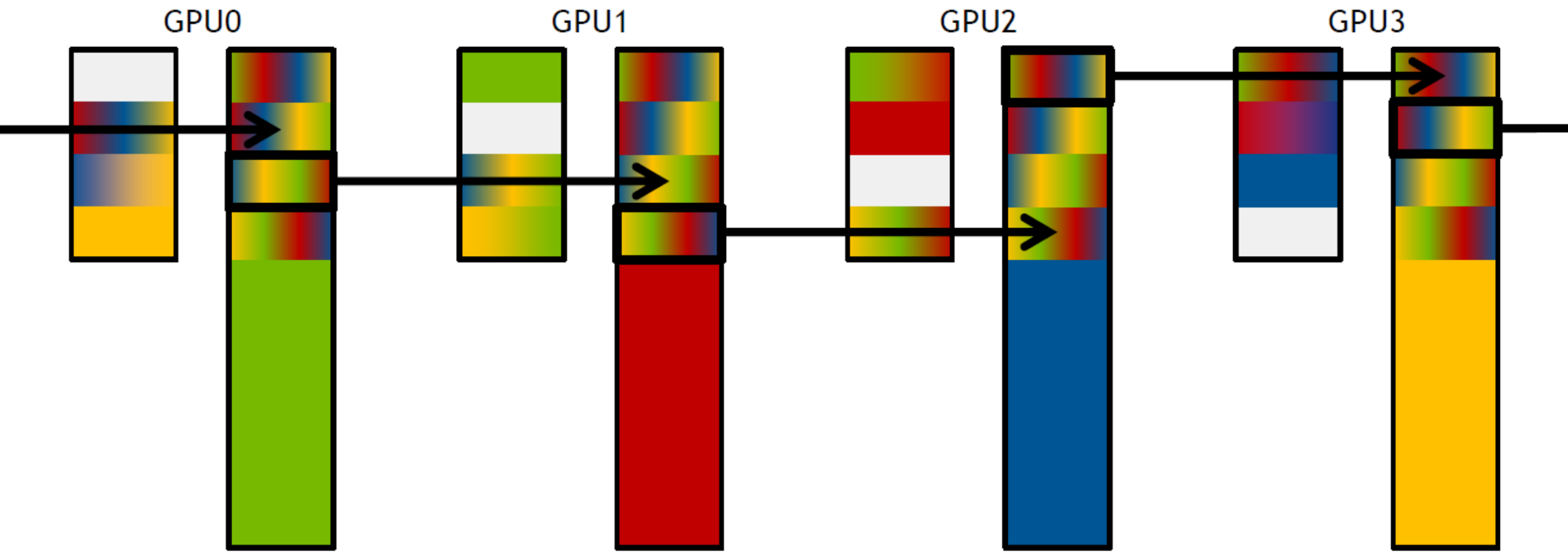
Chunk: 1
Step: 6



All-reduce

With unidirectional ring

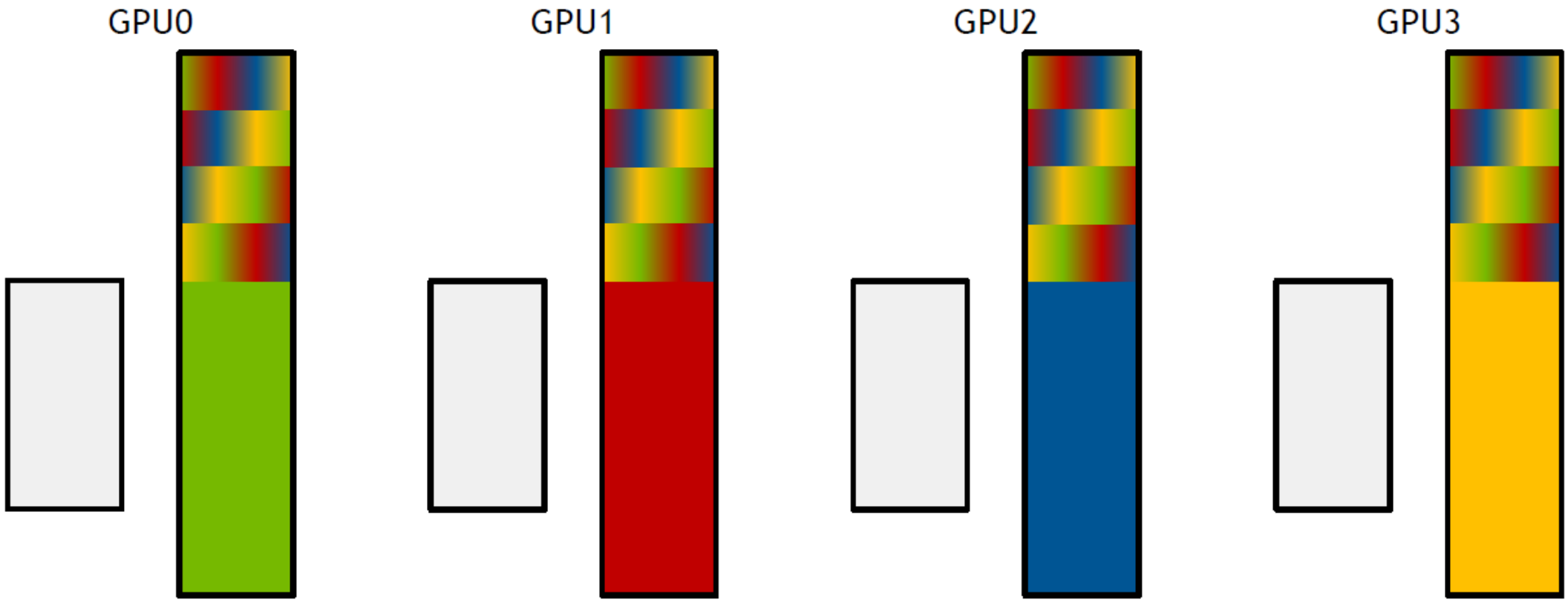
Chunk: 1
Step: 7



All-reduce

With unidirectional ring

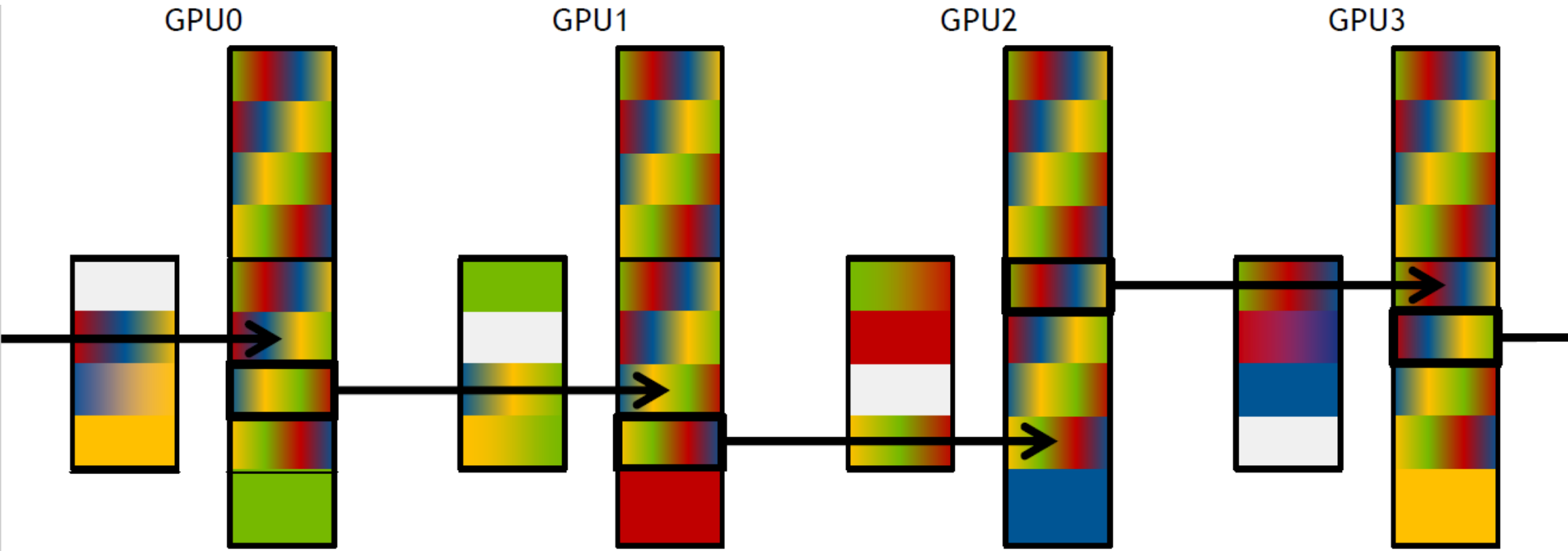
Chunk: 2
Step:



All-reduce

With unidirectional ring

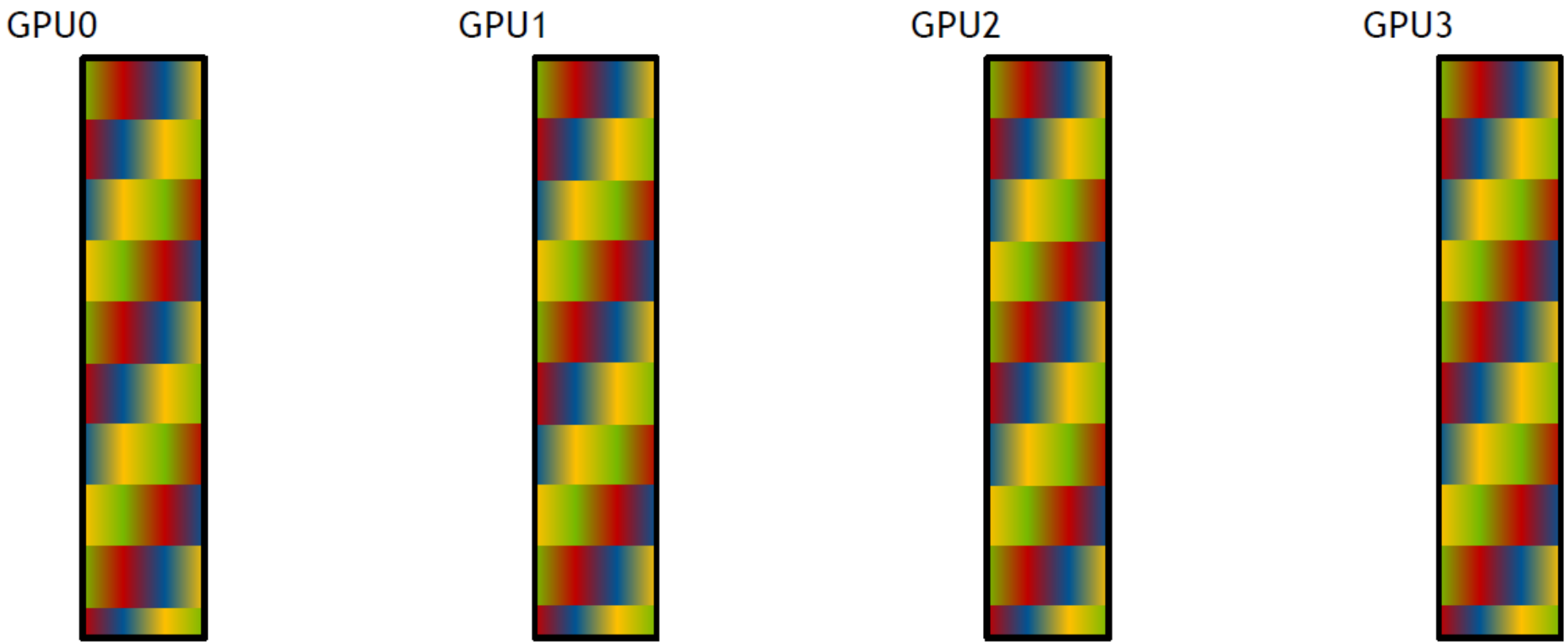
Chunk: 2
Step: 7



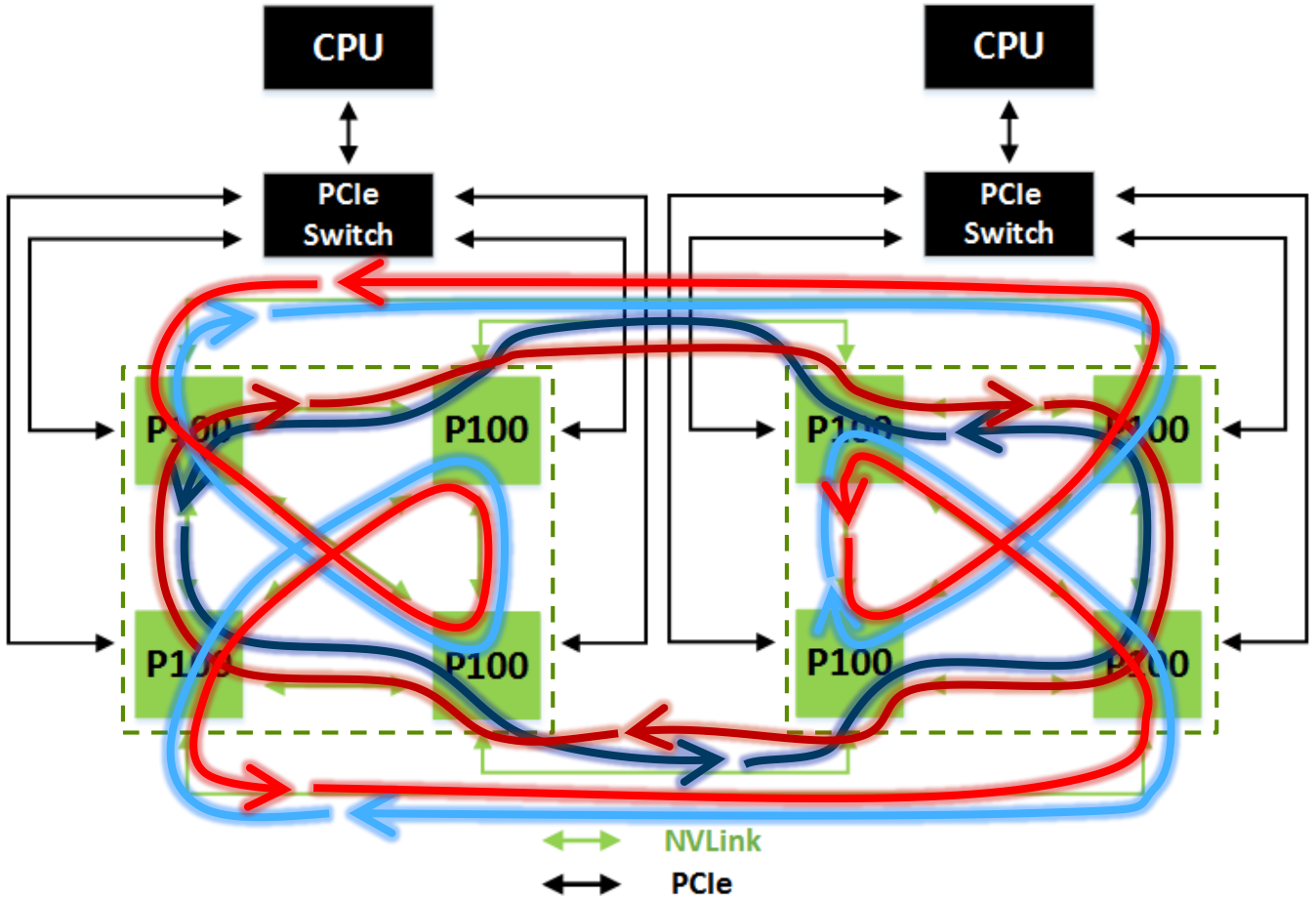
All-reduce

With unidirectional ring

Done



Apply NCCL to DGX-1

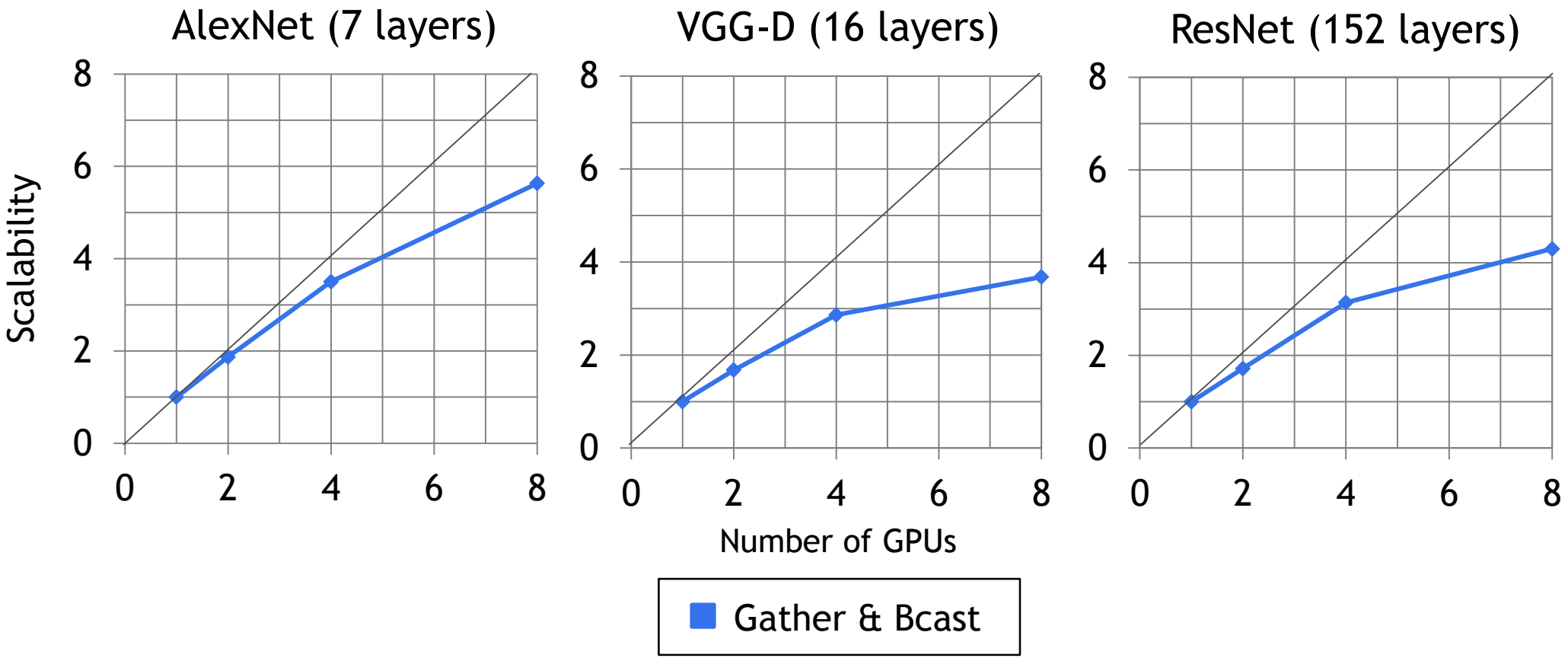


4 unidirectional rings

Peak: 80 GB/s
(20 GB/s per ring)

Multi-GPU performance w/o NCCL

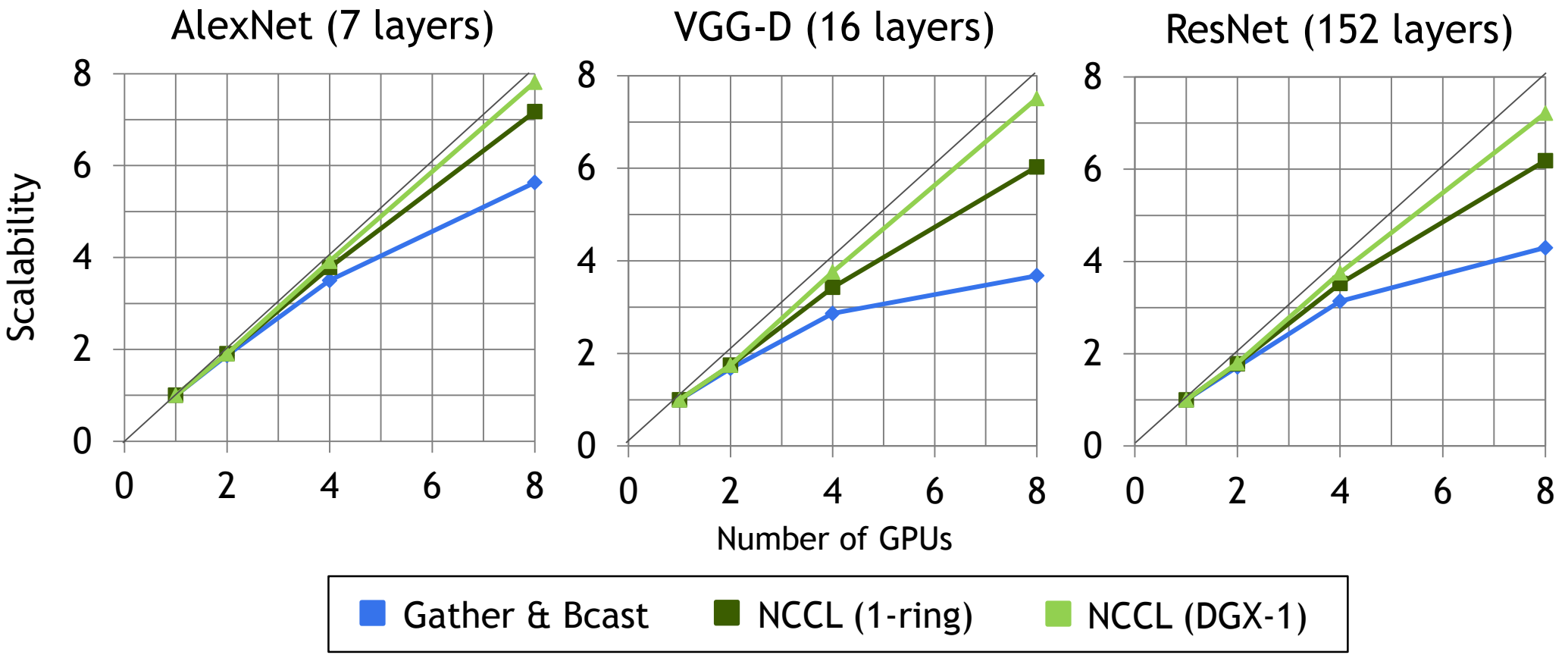
NVIDIA DGX-1, Chainer with multi-process patch



[Batch size per GPU] AlexNet:768, VGG-D:32, ResNet:12

Multi-GPU performance with NCCL

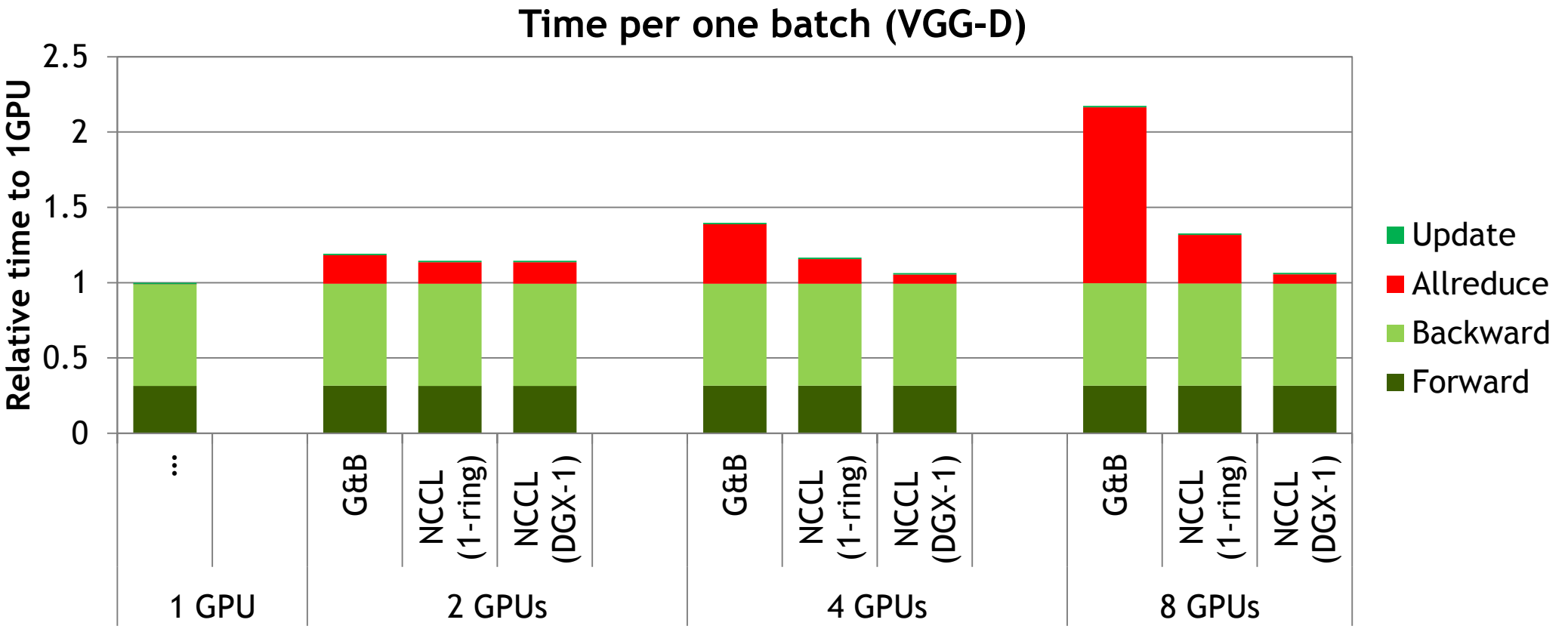
NVIDIA DGX-1, Chainer with NCCL patch



[Batch size per GPU] AlexNet:768, VGG-D:32, ResNet:12

Multi-GPU performance with NCCL

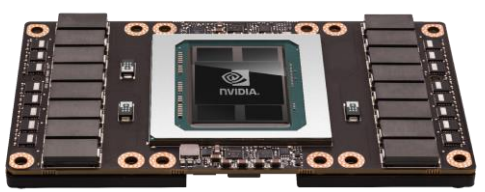
NVIDIA DGX-1, Chainer 1.17.0 with NCCL patch



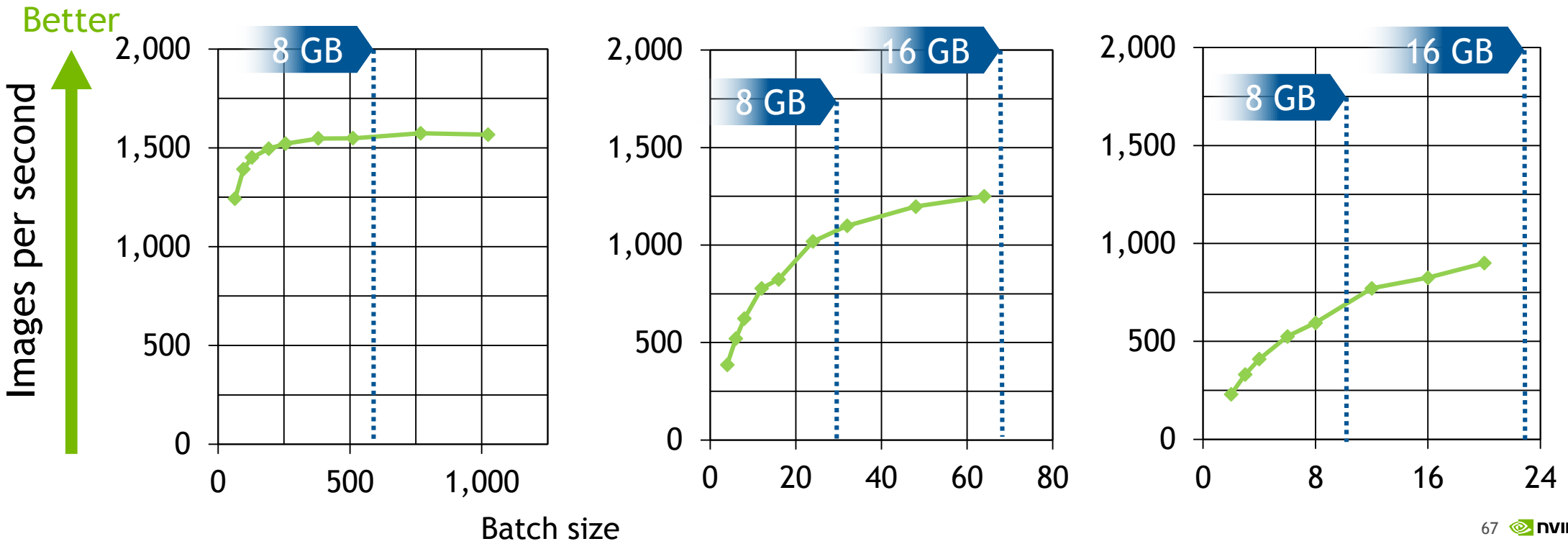
Out-of-core Training

Batch Size and Performance

TESLA P100, Chainer

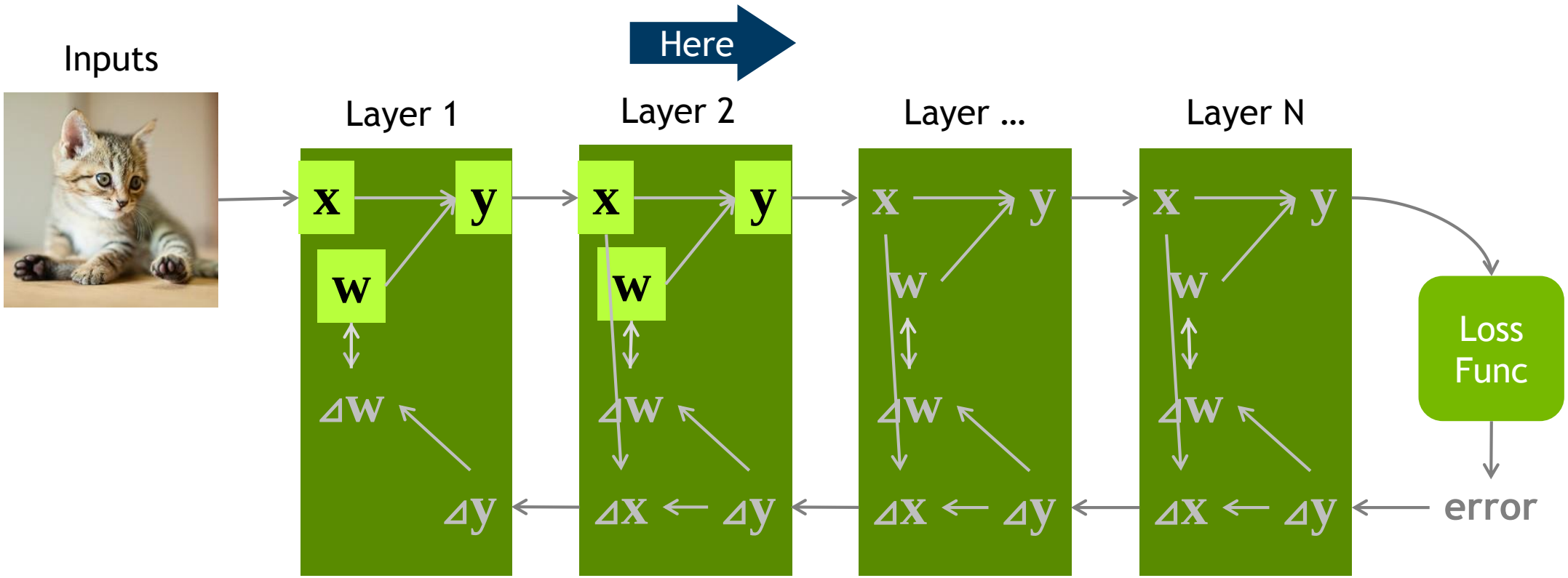


AlexNet (7 layers) 2012 | VGG-D (16 layers) 2014 | ResNet (152 layers) 2015



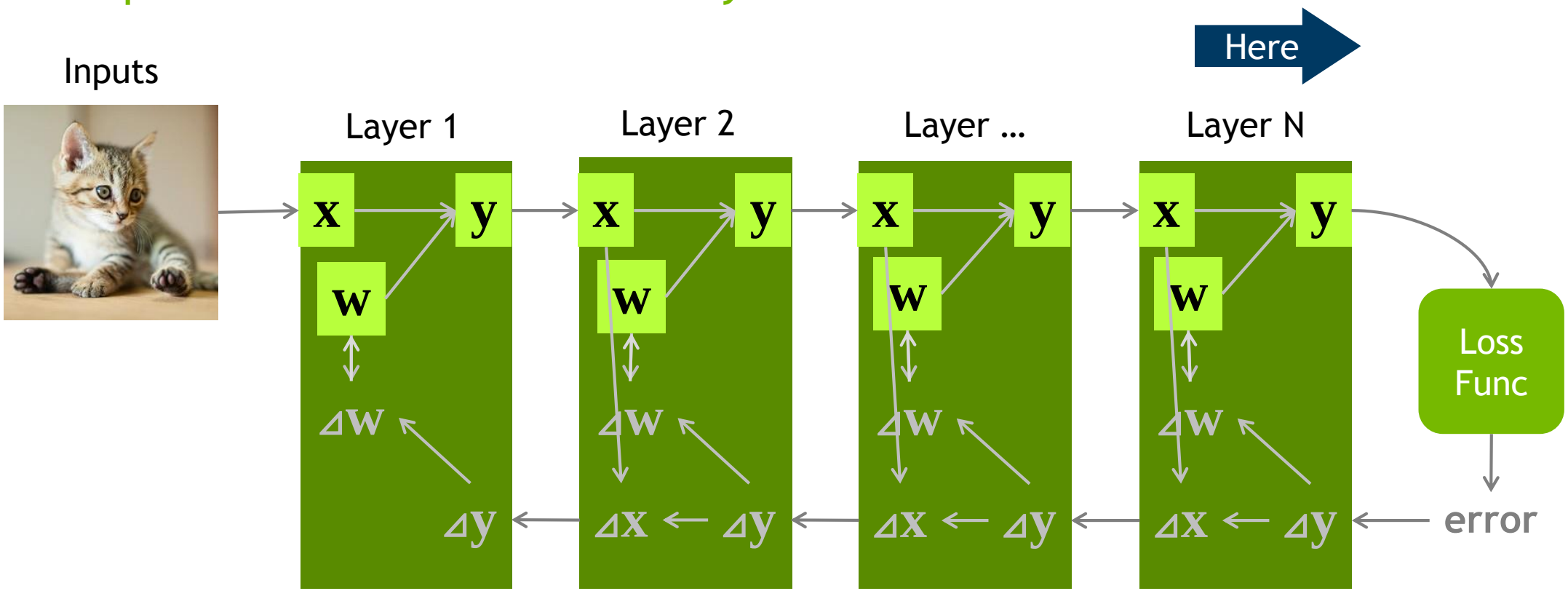
Network wide memory allocation

Keep whole data on GPU memory



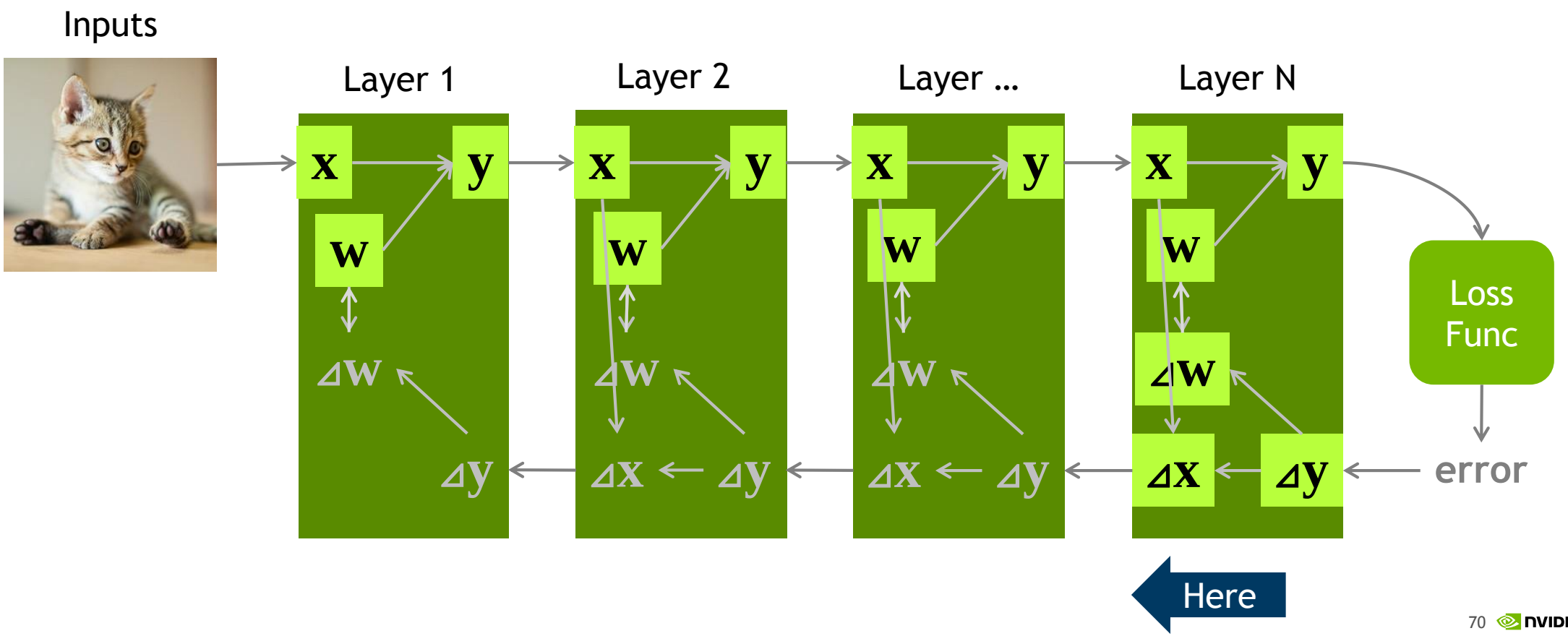
Network wide memory allocation

Keep whole data on GPU memory



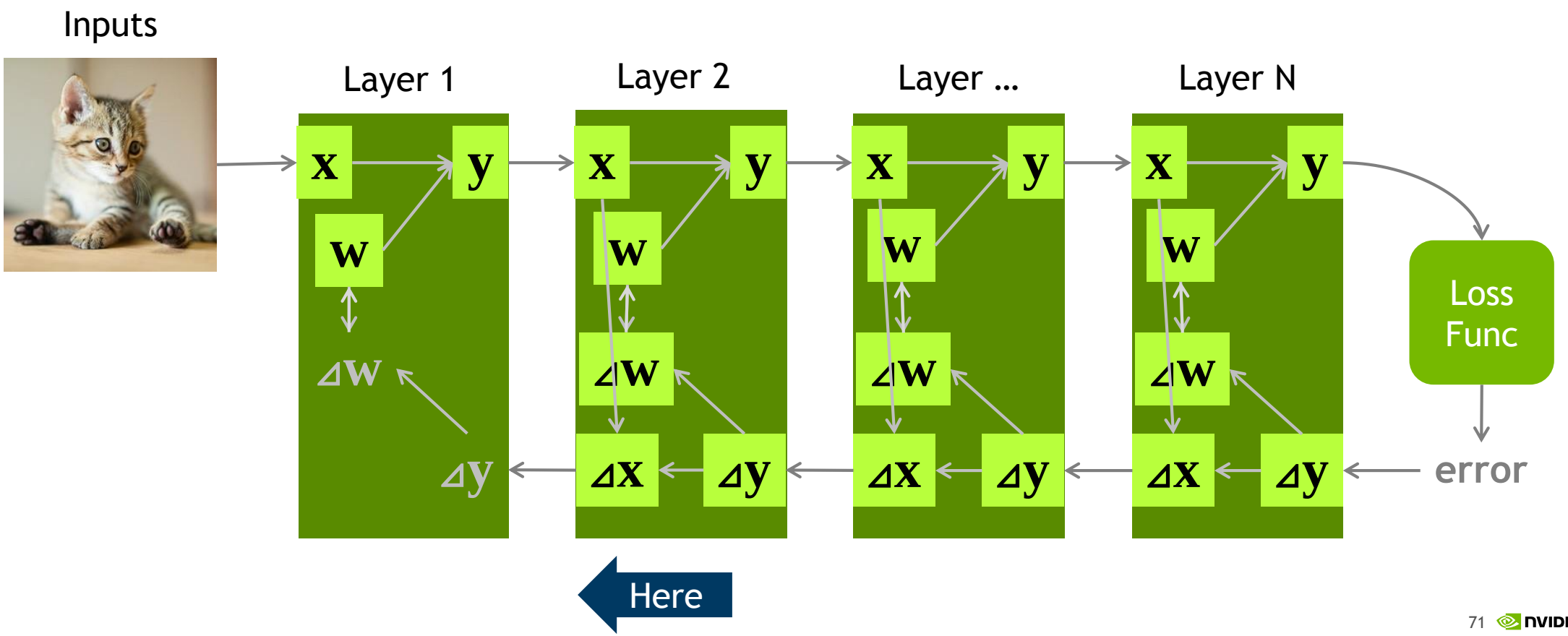
Network wide memory allocation

Keep whole data on GPU memory



Network wide memory allocation

Keep whole data on GPU memory

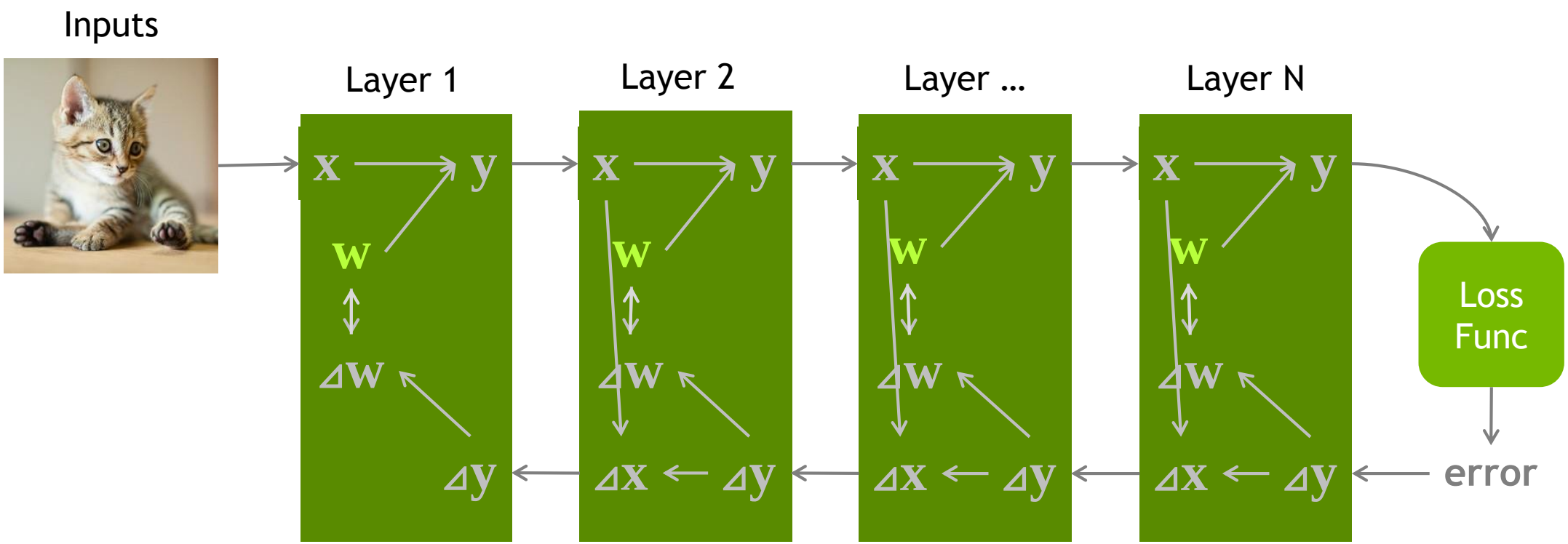


Out-of-core training

- Why do we need to keep whole data on GPU?
- Can we swap-out data to CPU memory that is not used soon?
  - CPU memory is much larger than GPU memory

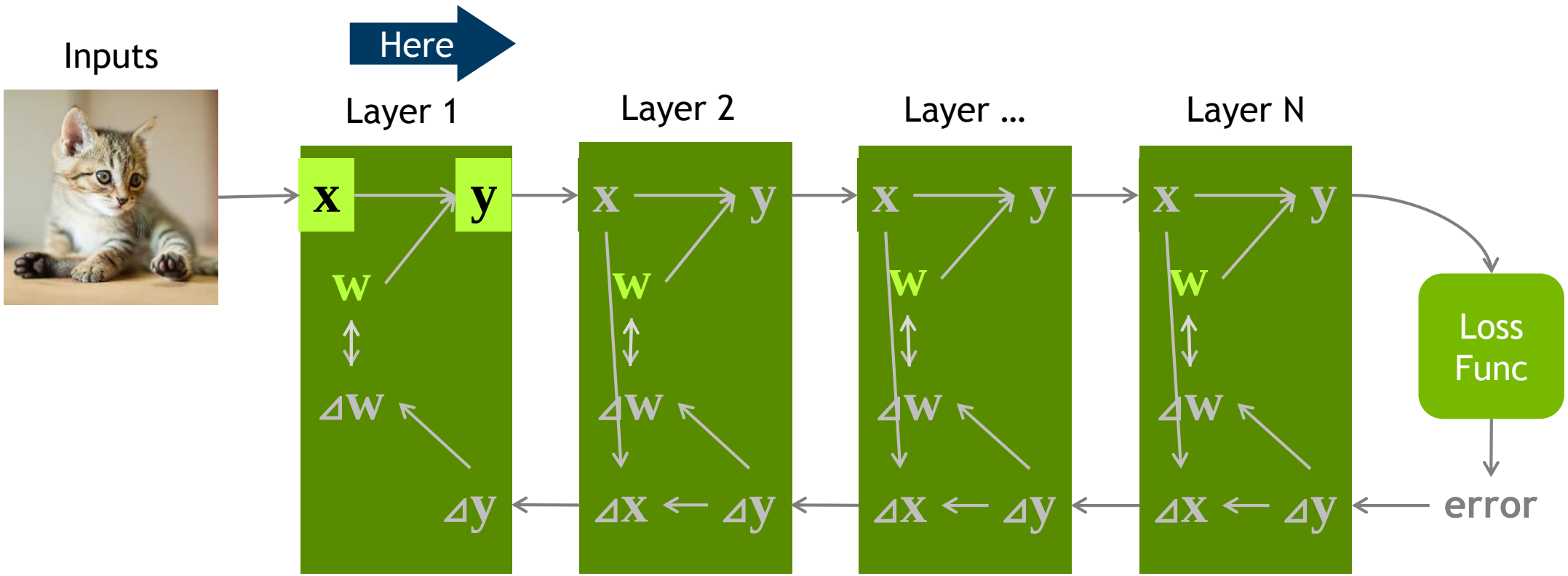
Layer wise memory allocation

Swap-out/in data to/from CPU memory



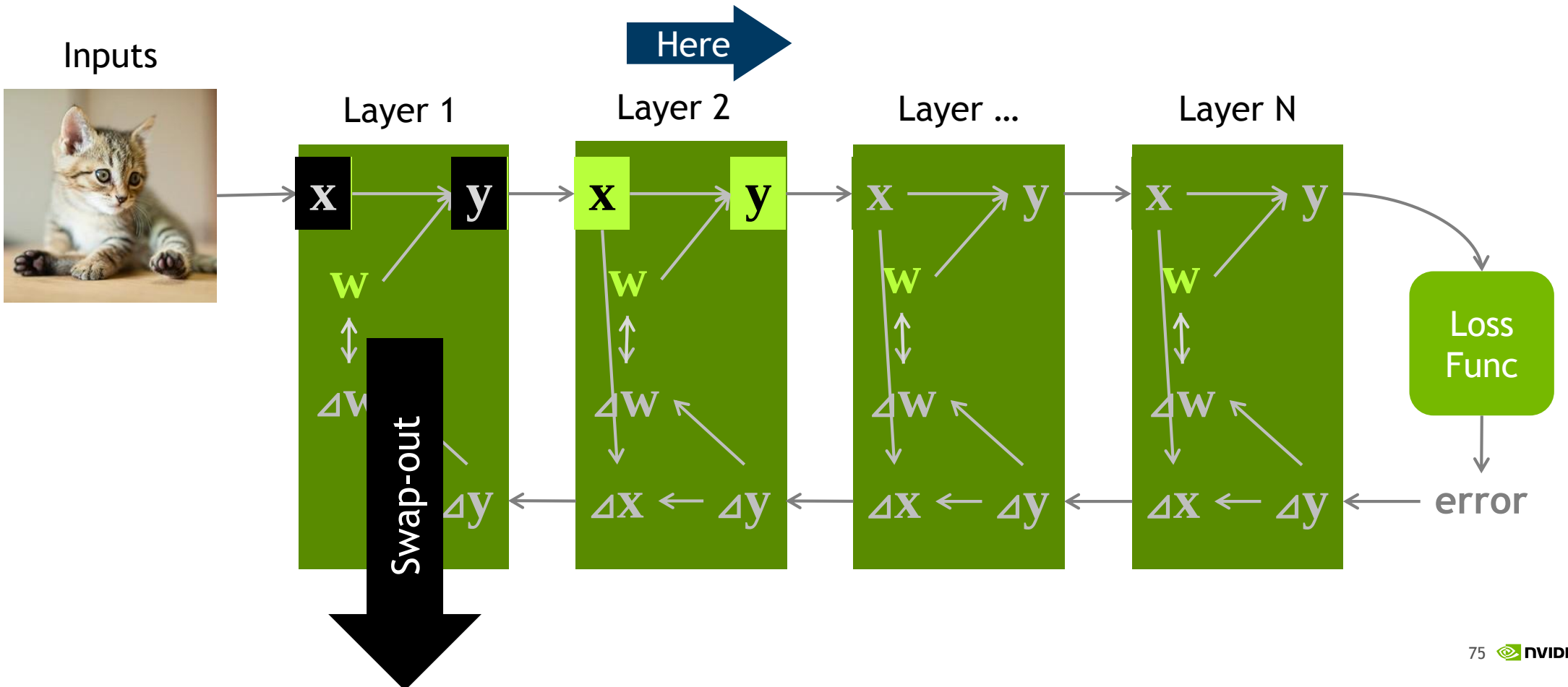
Layer wise memory allocation

Swap-out/in data to/from CPU memory



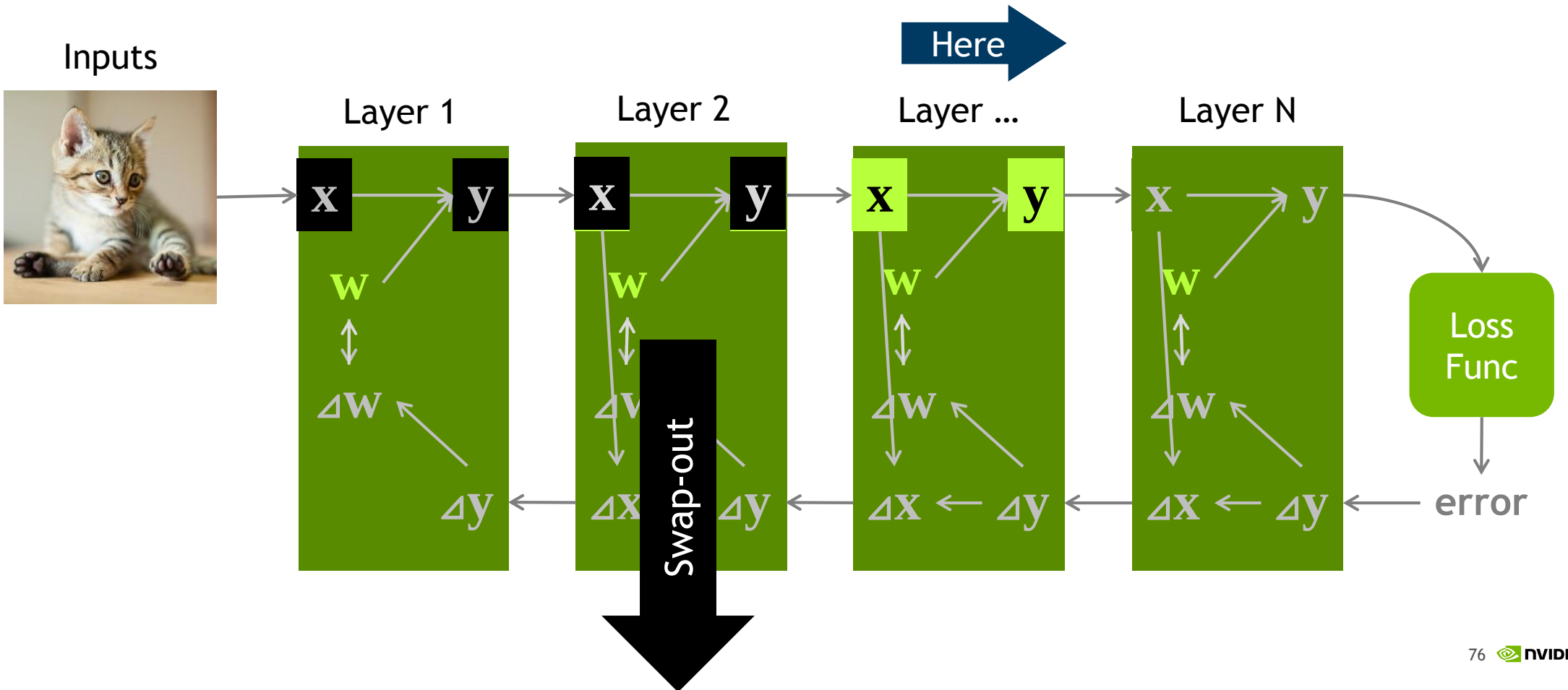
Layer wise memory allocation

Swap-out/in data to/from CPU memory



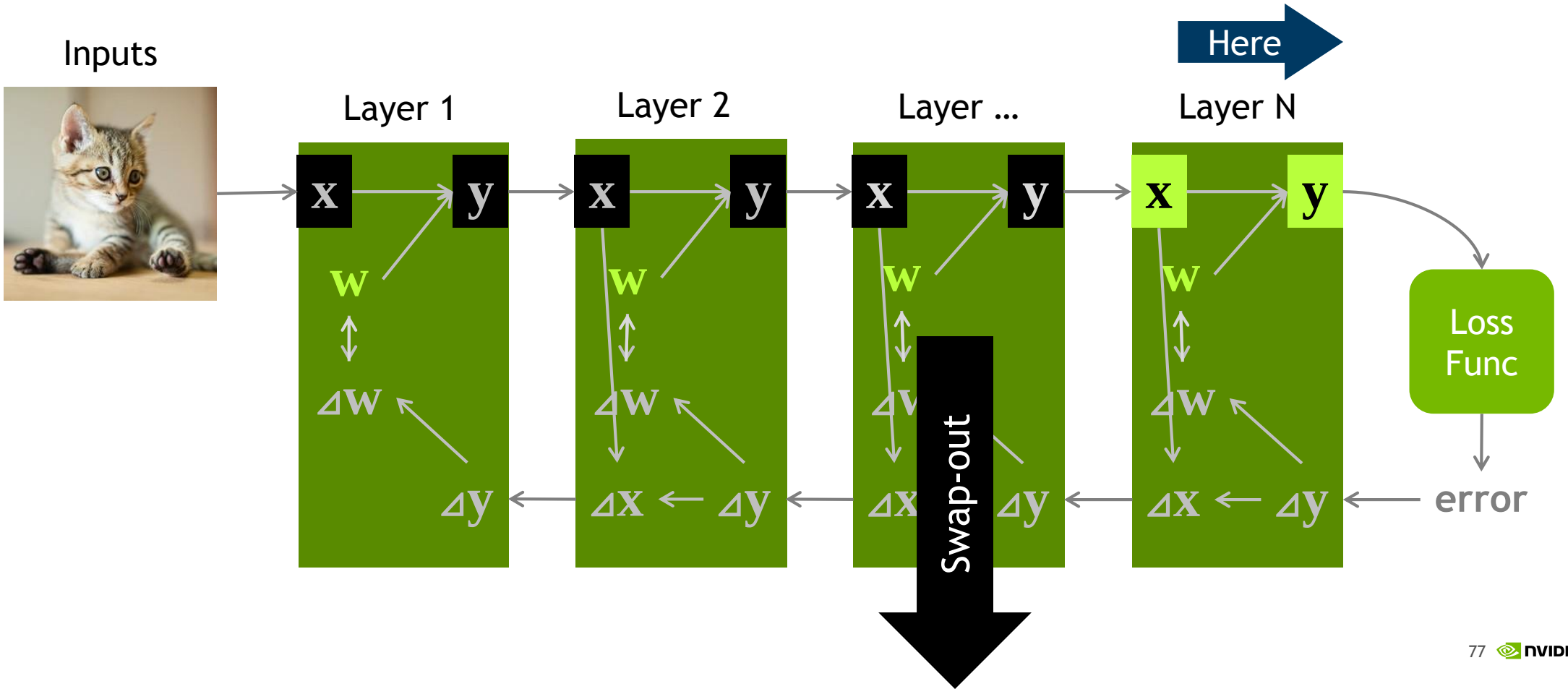
Layer wise memory allocation

Swap-out/in data to/from CPU memory



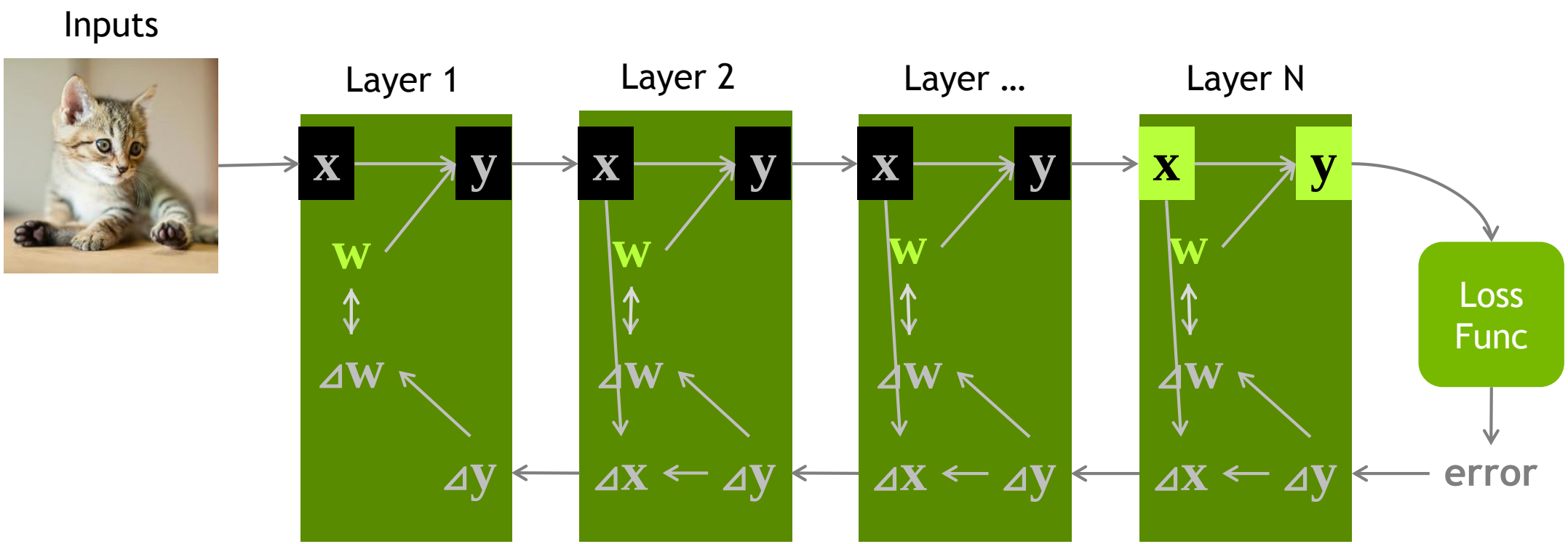
Layer wise memory allocation

Swap-out/in data to/from CPU memory



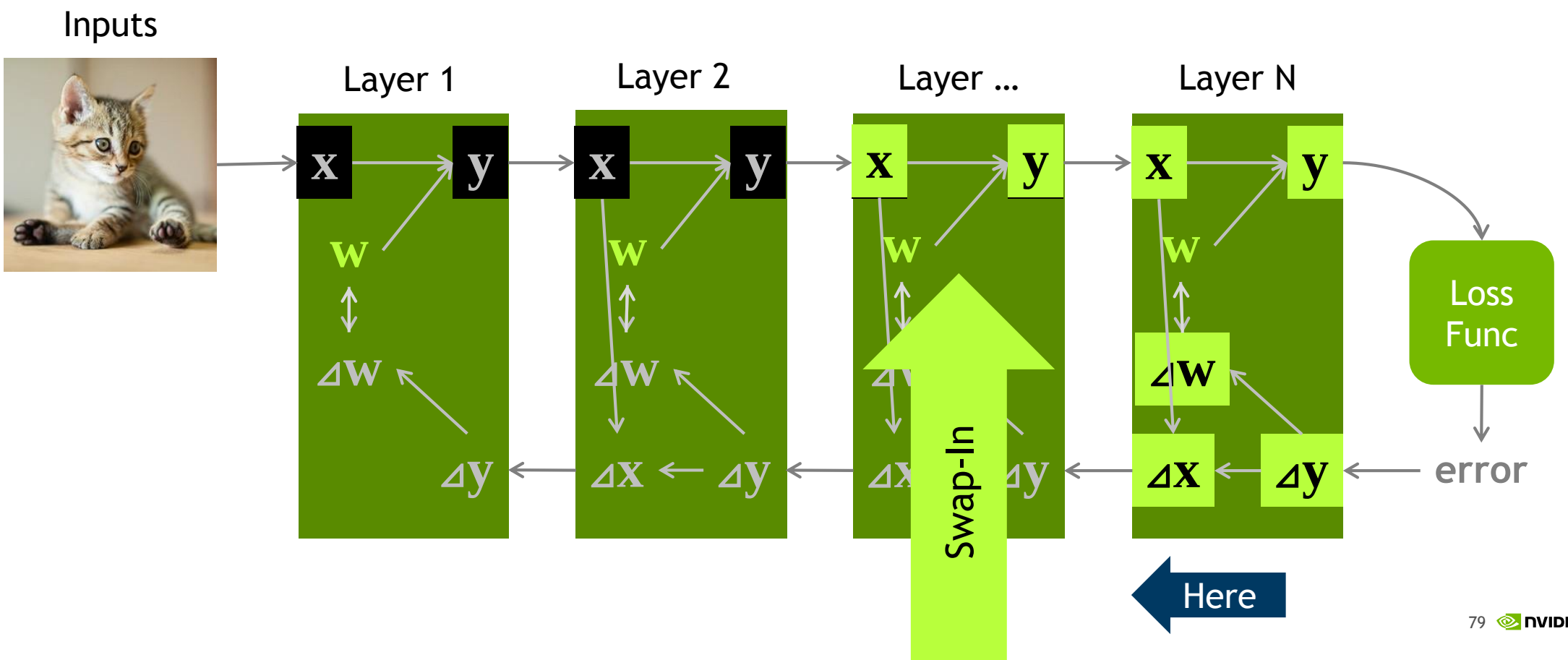
Layer wise memory allocation

Swap-out/in data to/from CPU memory



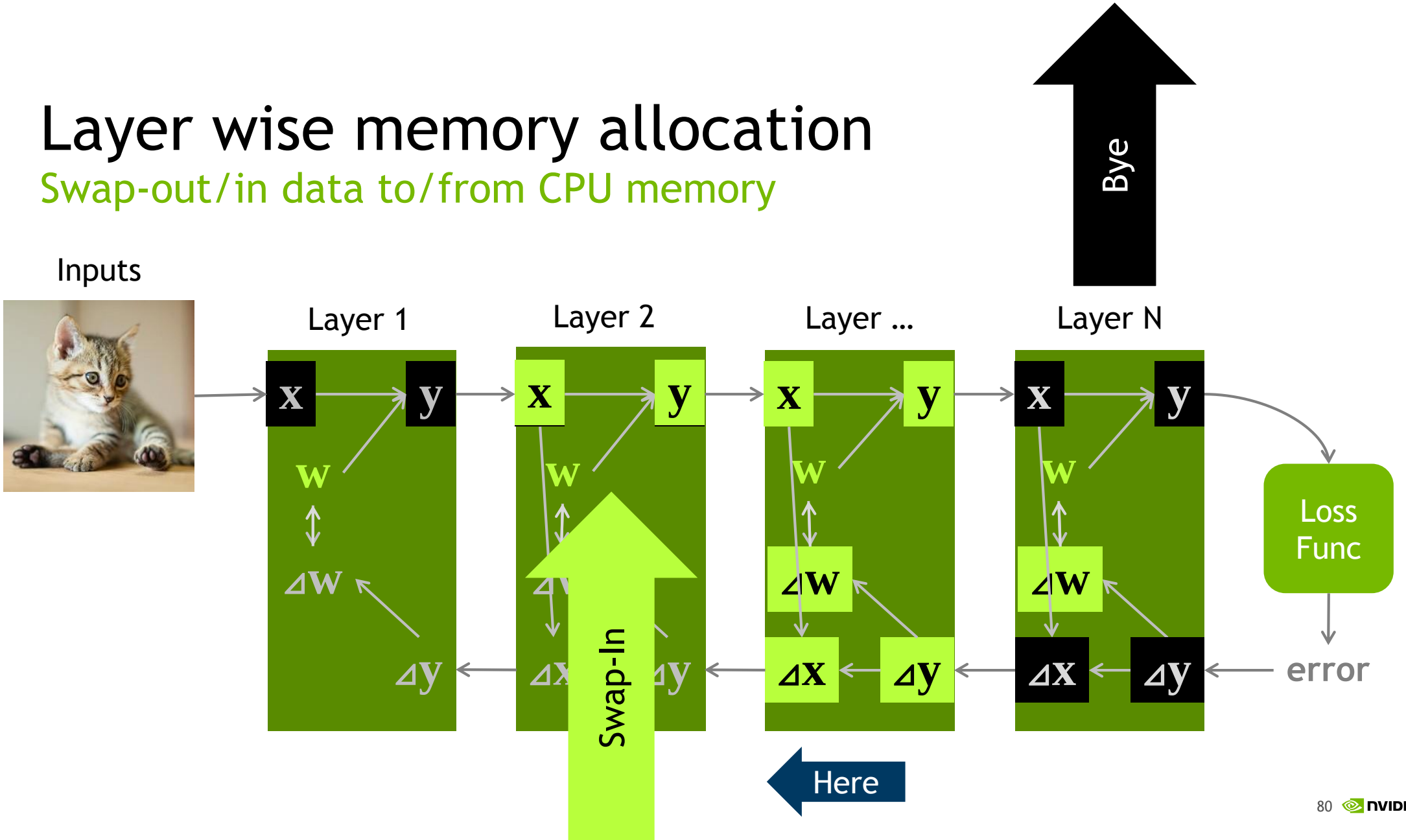
Layer wise memory allocation

Swap-out/in data to/from CPU memory



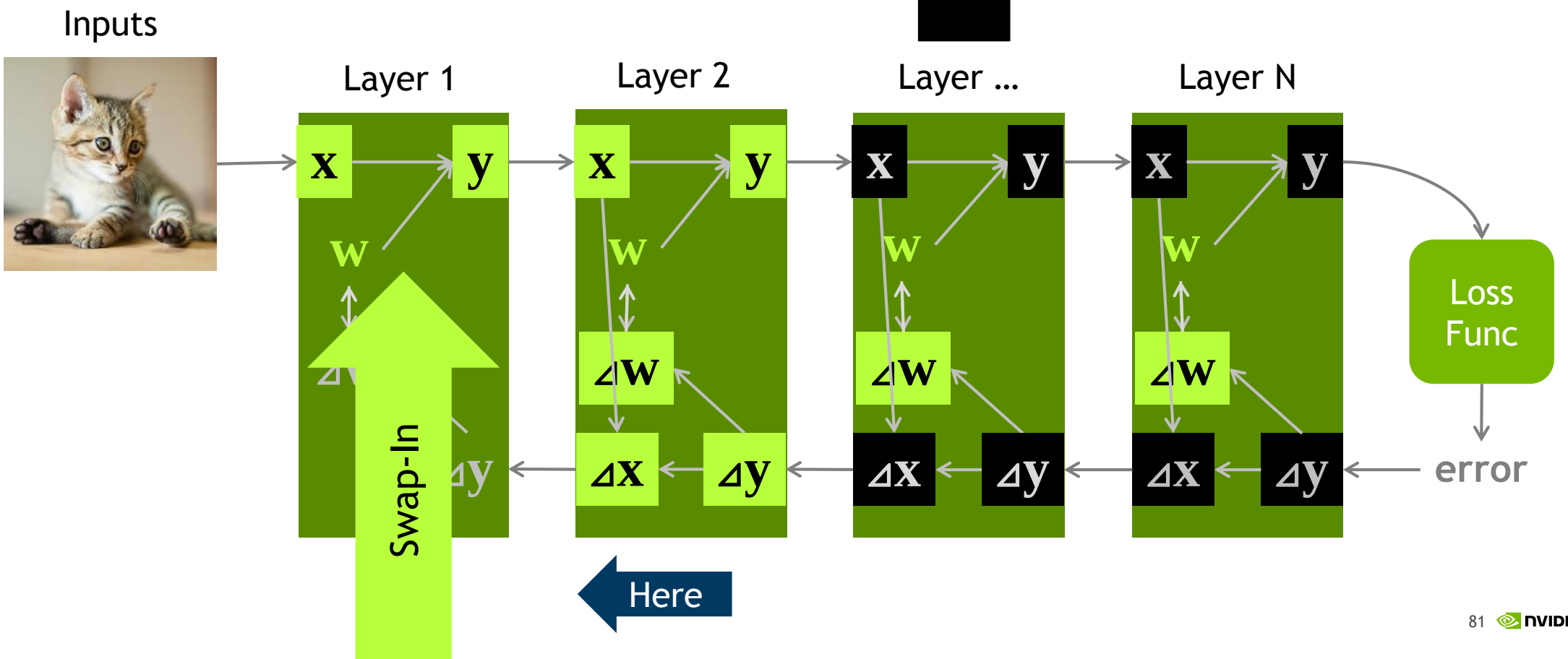
Layer wise memory allocation

Swap-out/in data to/from CPU memory



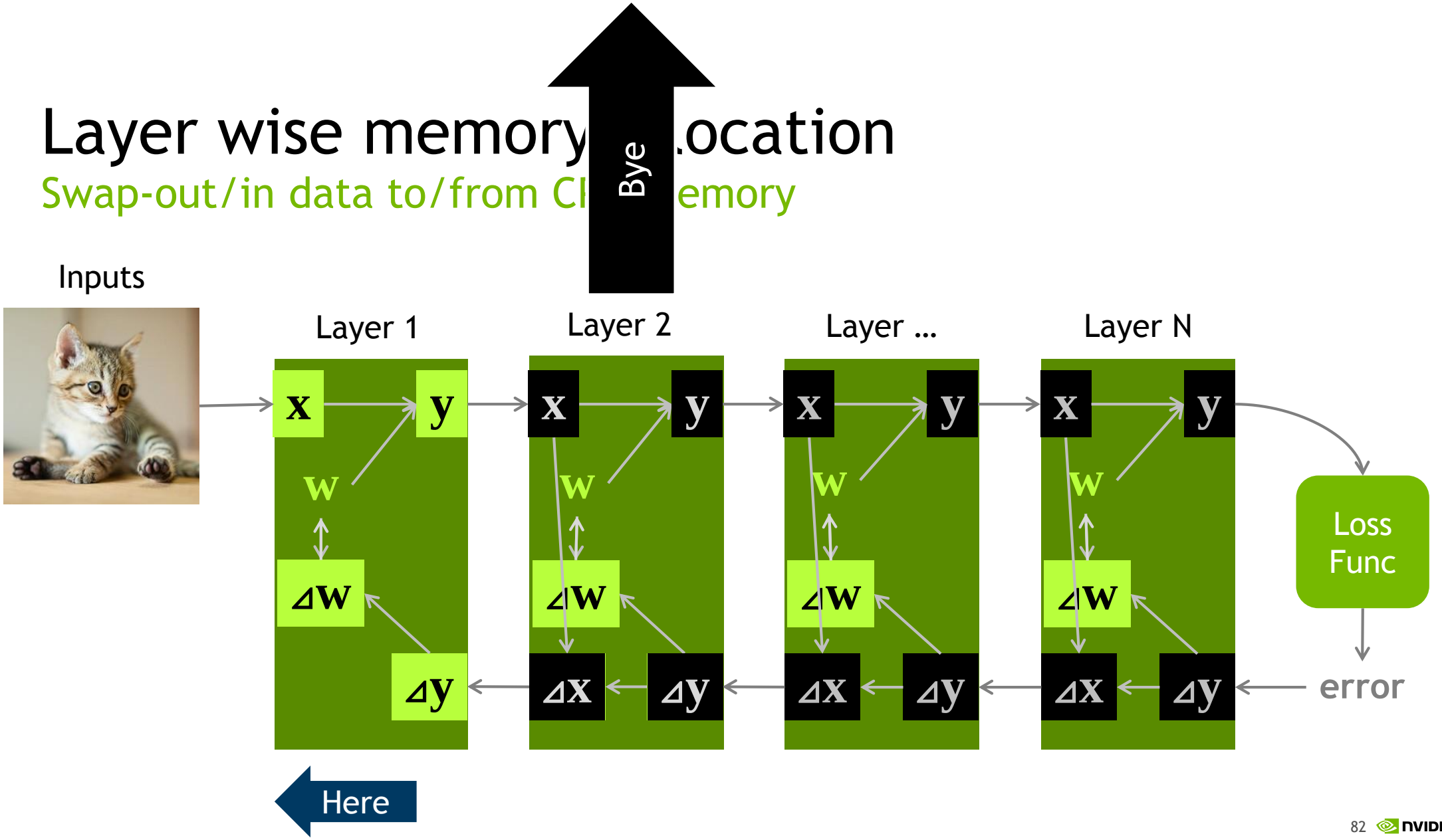
Layer wise memory allocatio

Swap-out/in data to/from CPU memory



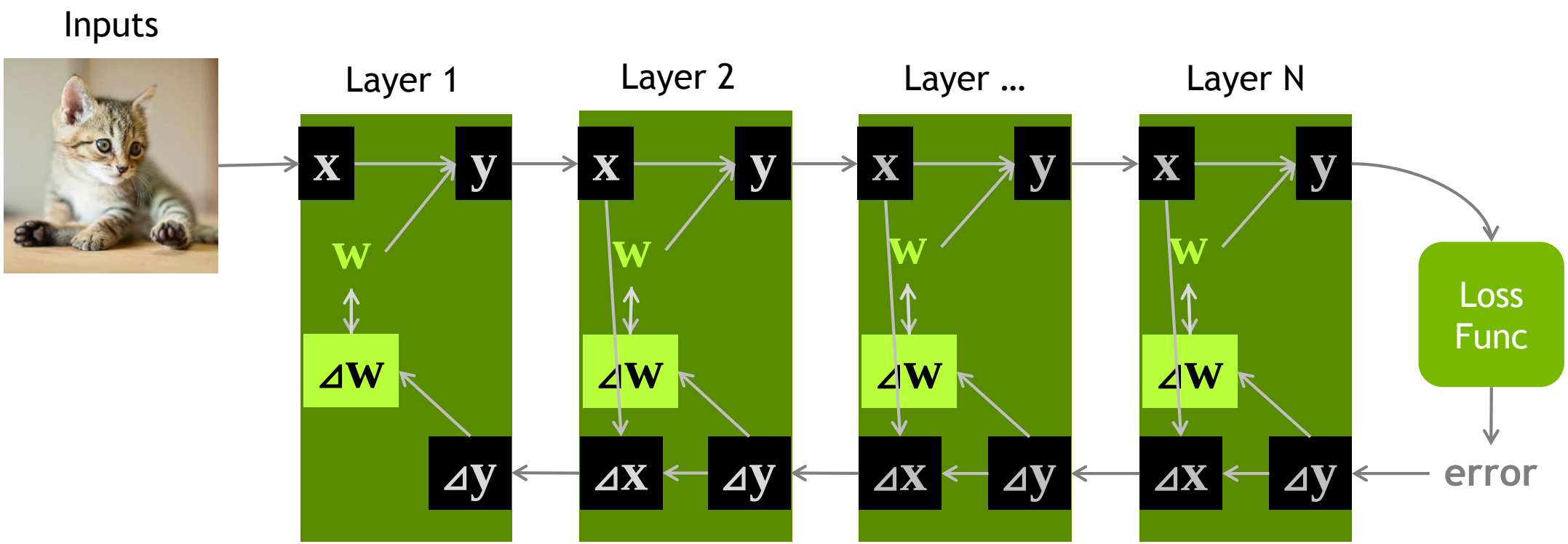
Layer wise memory .ocation

Swap-out/in data to/from CI emory

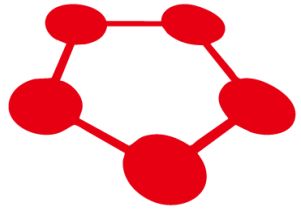


Layer wise memory allocation

Swap-out/in data to/from CPU memory



Out-of-core trial on Chainer



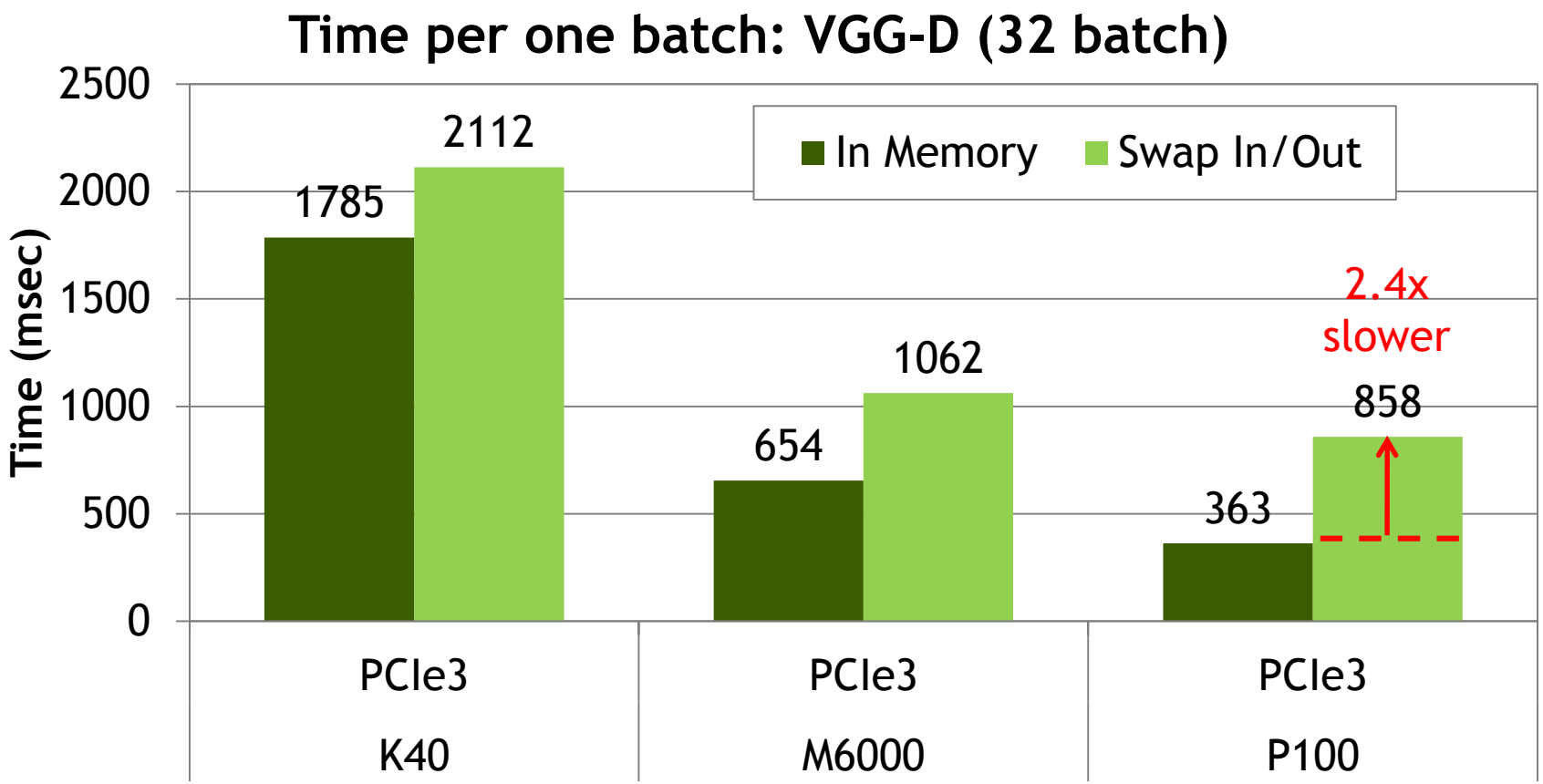
Chainer + Out-of-core patch (*)

- Add some features for Out-of-core training to Chainer
  - Swap-out data that is not use for the time being (Forward)
  - Swap-in data that will be used next (Backward)
  - Kernel and data transfers are fully overlapped.

(*) : https://github.com/anaruse/chainer/tree/test.out-of-core

Out-of-core: performance comparision

Chainer + Out-of-core patch (*)



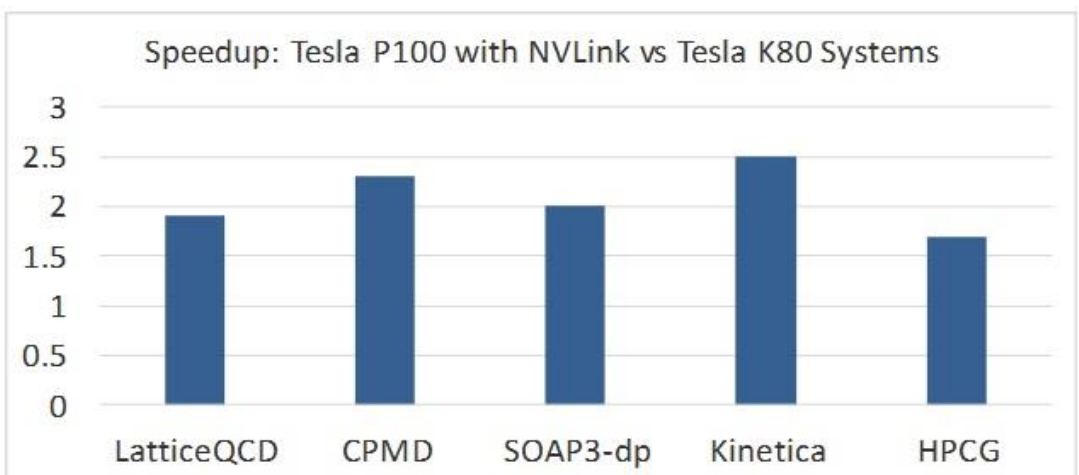
PCIe3 is too slow for P100

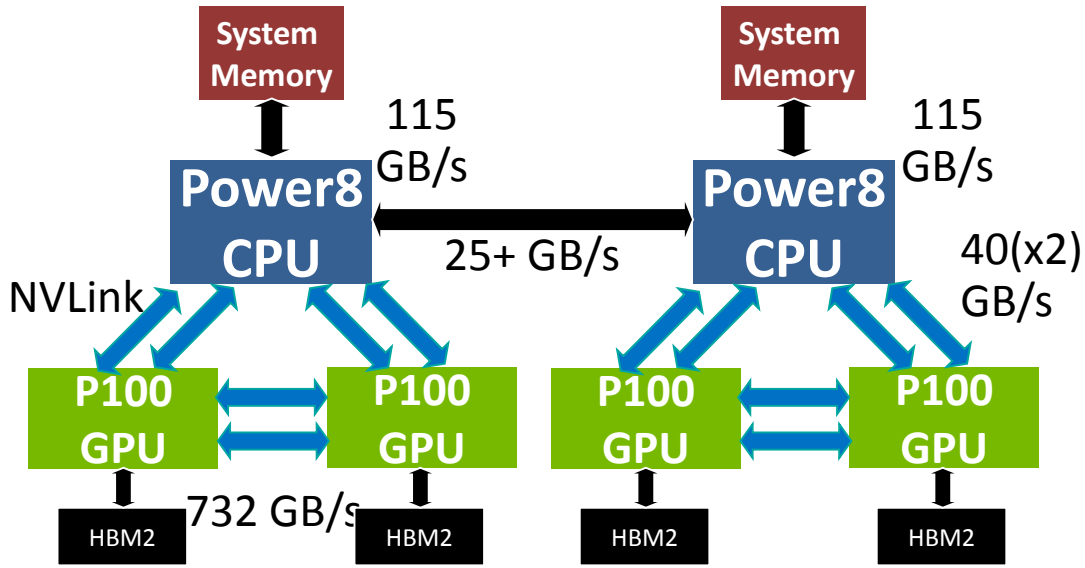
(*) : https://github.com/anaruse/chainer/tree/test.out-of-core

IBM Minsky: 4 Pascal GPUs NVLink attach to 2 Power8 CPUs

NVLink between CPUs and GPUs enables fast memory access to large data sets in system memory

Two NVLink connections between each GPU and CPU-GPU leads to faster data exchange

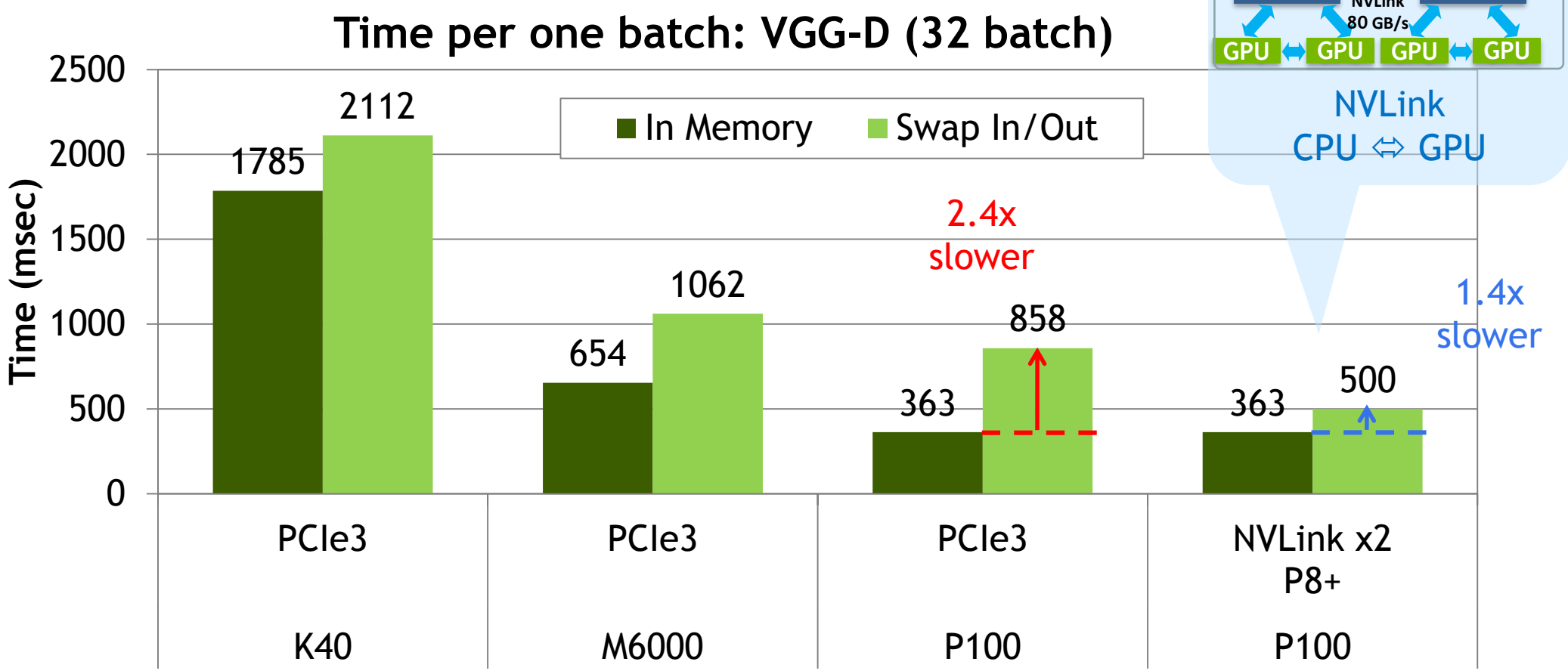




https://www.ibm.com/blogs/systems/ibm-nvidia-present-nvlink-server-youve-waiting/

Out-of-core: performance comparision

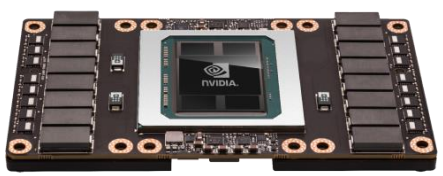
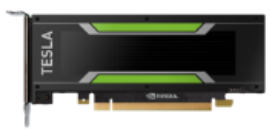
Chainer + Out-of-core patch (*)



(*) : https://github.com/anaruse/chainer/tree/test.out-of-core

Inference

Pascal GPUs (TESLA)

TESLA P100 (SXM2)	TESLA P100 (PCI)	TESLA P40	TESLA P4
			
HPC and Deep Learning GP100	HPC and Deep Learning GP100	Deep Learning GP102	Deep Learning (Inference) GP104

CC6.0

CC6.1

Two Pascals

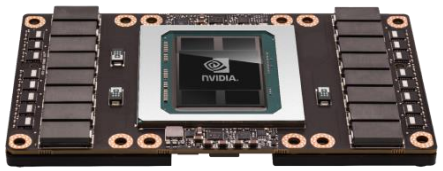
GP100 (CC 6.0)

56 SMs x 64 CUDA cores

GP102 (CC 6.1)

30 SMs x 128 CUDA cores

TESLA P100 (GP100, CC 6.0)



3584 CUDA Cores

FP64: 5.3 TF

FP32: 10.6 TF

FP16: 21.2 TF

INT8: …

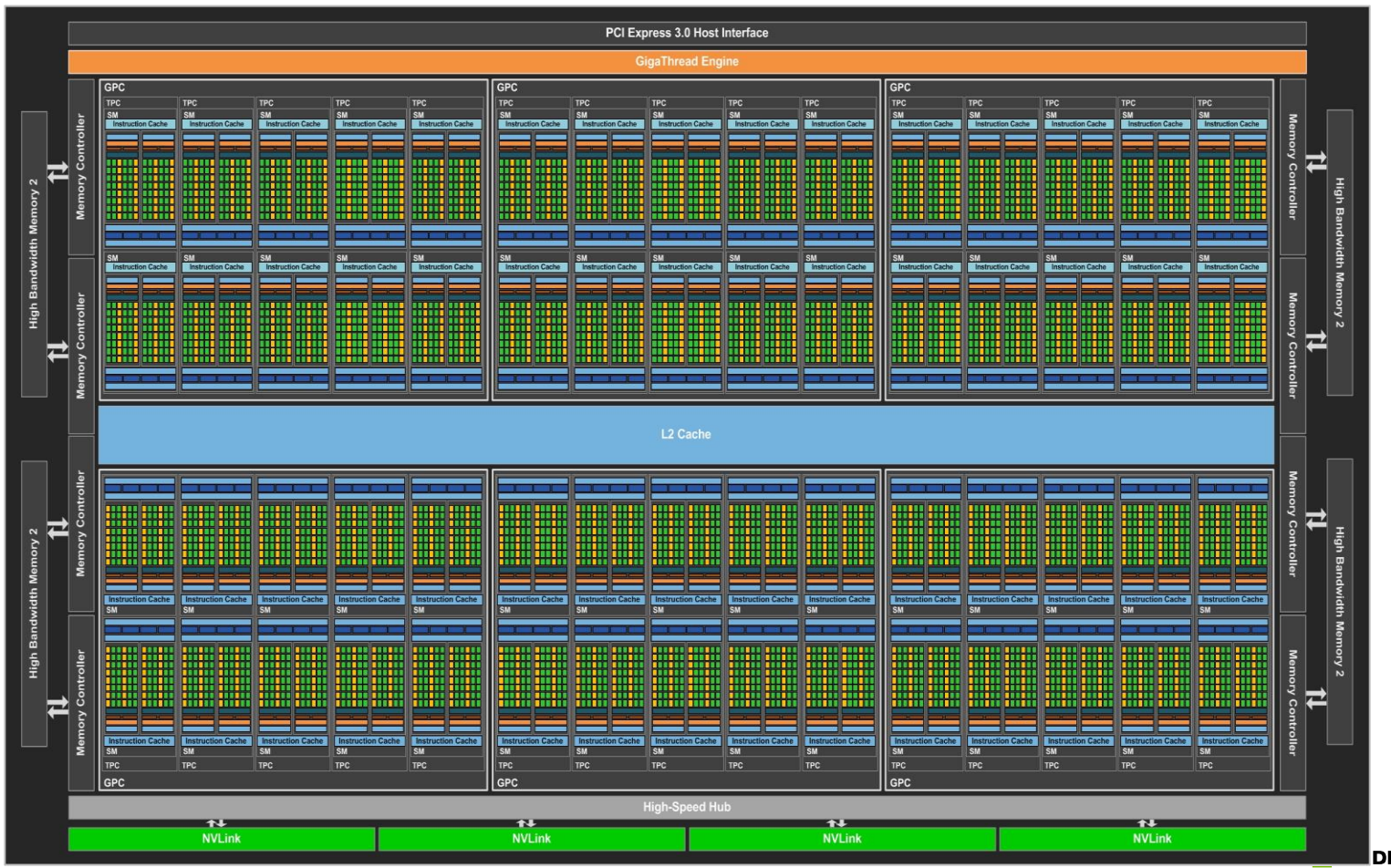
HBM2

4096 bit width

16 GB

732 GB/s

56 SMs x 64 CUDA cores



TESLA P40 (GP102, CC 6.1)

3840 CUDA Cores

FP64: …

FP32: 11.8 TF

FP16: …

INT8: 47.0 Top

GDDR5X

384 bit width

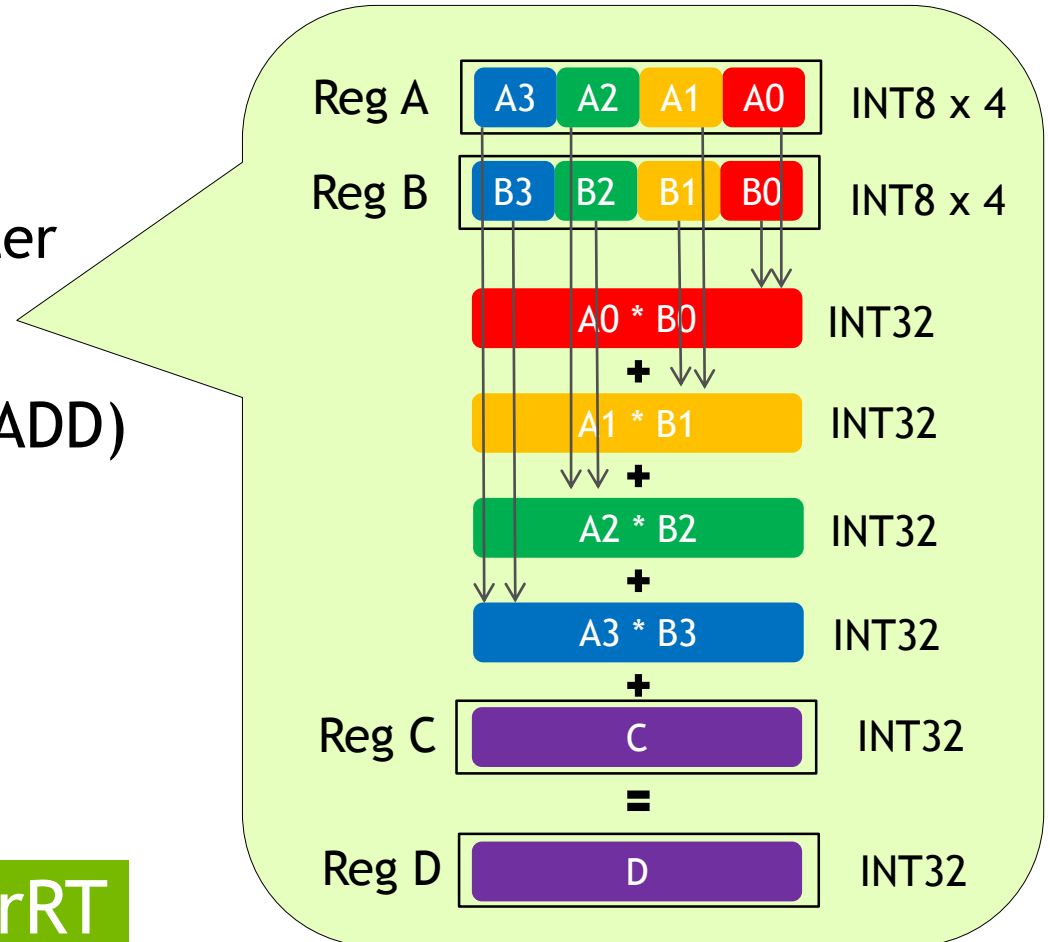
24 GB

347 GB/s

30 SMs x 128 CUDA cores

[CC 6.1] New Instruction for INT8: dp4a

- 4-way INT8 vector dot + INT32
  - four 8-bit integer in a 32-bit register
  - (A0*B0 + A1*B1 + A2*B2 + A3*B3) + C
- 8 arithmetics per cycle (4 MUL + 4 ADD)
- 4x faster than FP32



USE CASE
Deep Learning (inference)

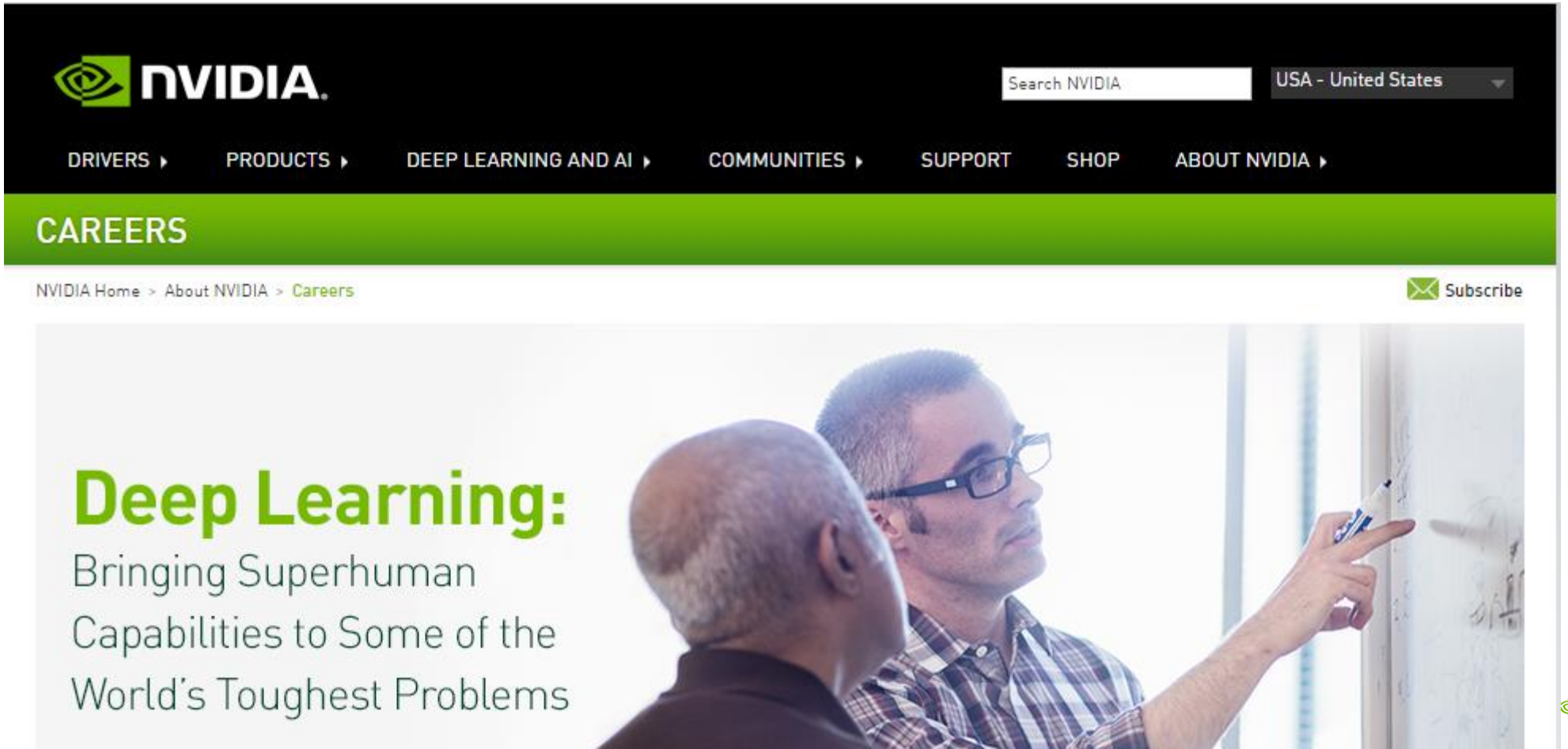
tensorRT

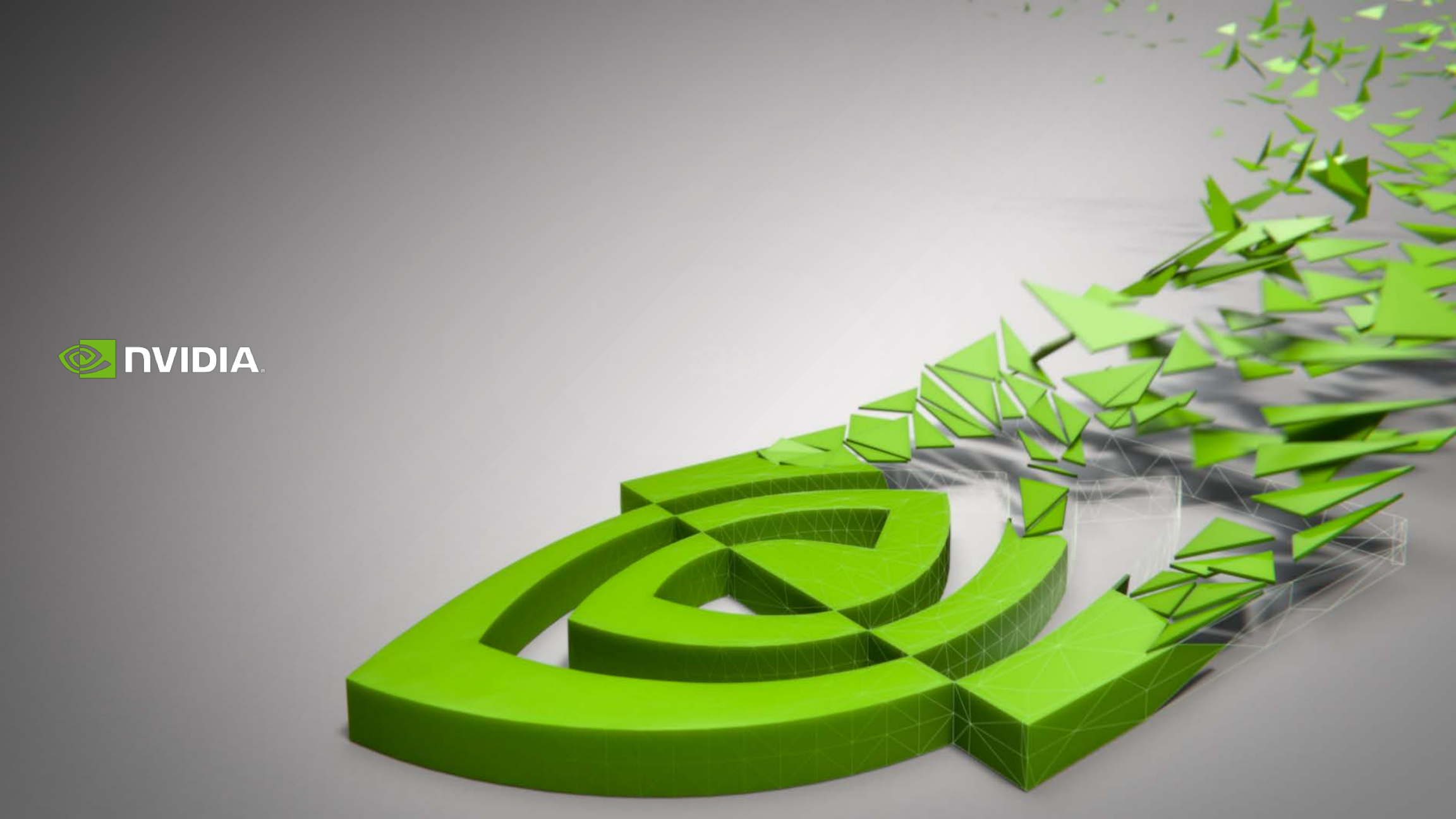
Summary and Futre

- Summary
  - Pascal: The engine for DL/AI
  - FP16, NVLINK, NCCL, Out-of-core for training
  - INT8 for inference
- Future
  - Volta: Next generation GPU architecutre
  - We continue to enhance GPU for DL/AI

We’re hiring engineers!

http://www.nvidia.com/object/careers.html




NVIDIA