ATPESC 2021

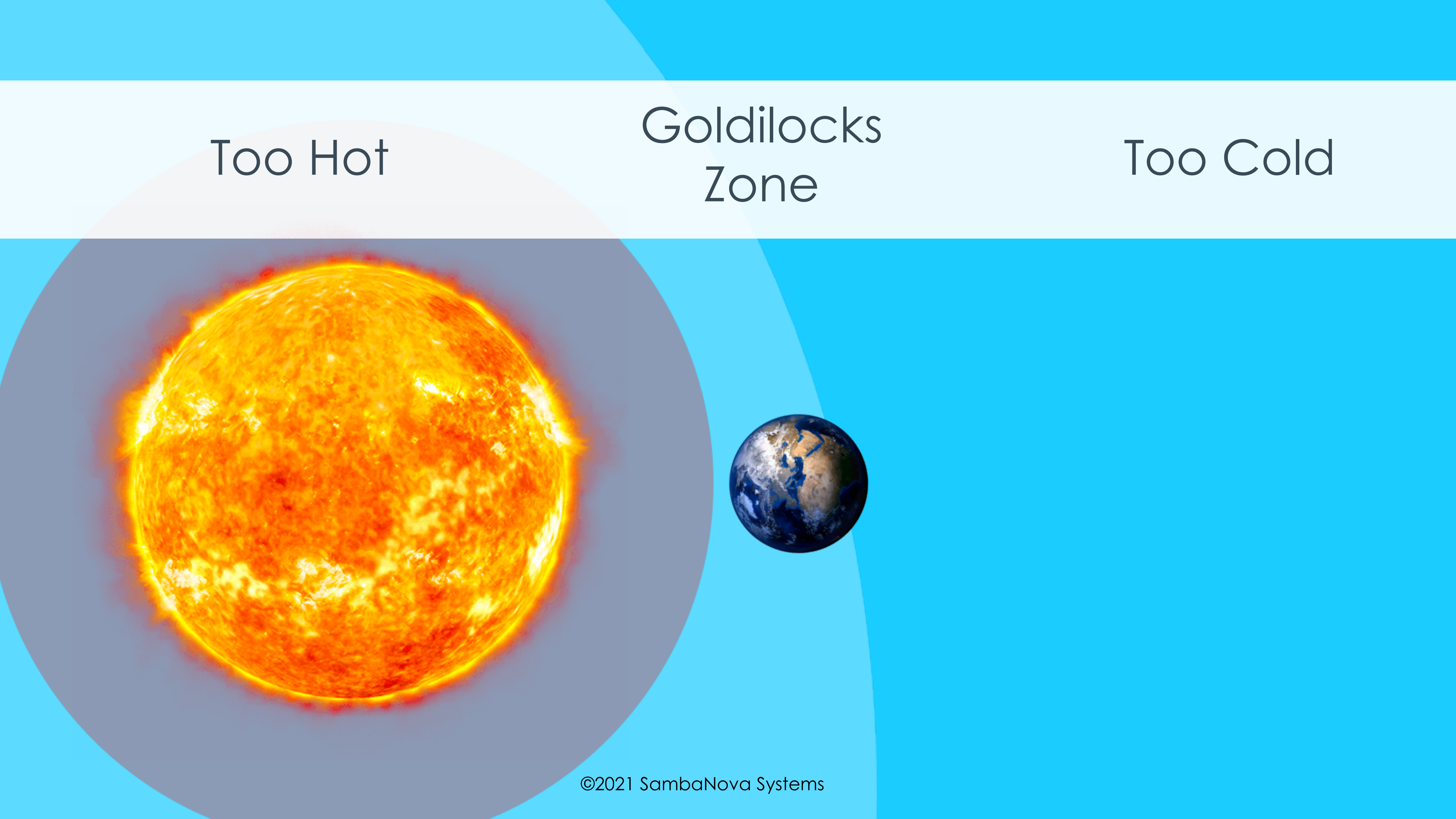
August 8, 2021

SW/HW Innovations in Emerging DL Training Systems

Urmish Thakker

Principal Engineer

SambaNova SYSTEMS

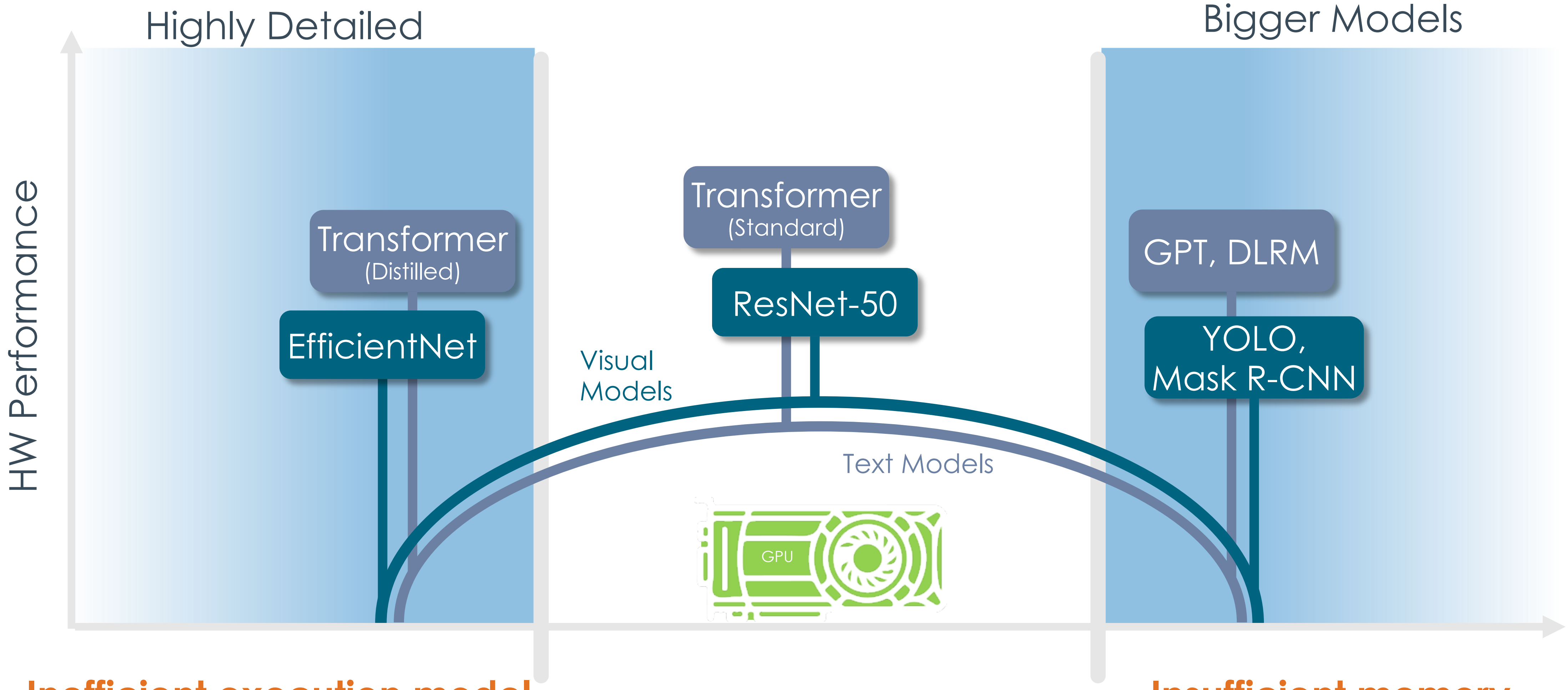


Too Hot

Goldilocks Zone

Too Cold

Yesterday's Goldilocks Zone is Constraining Progress

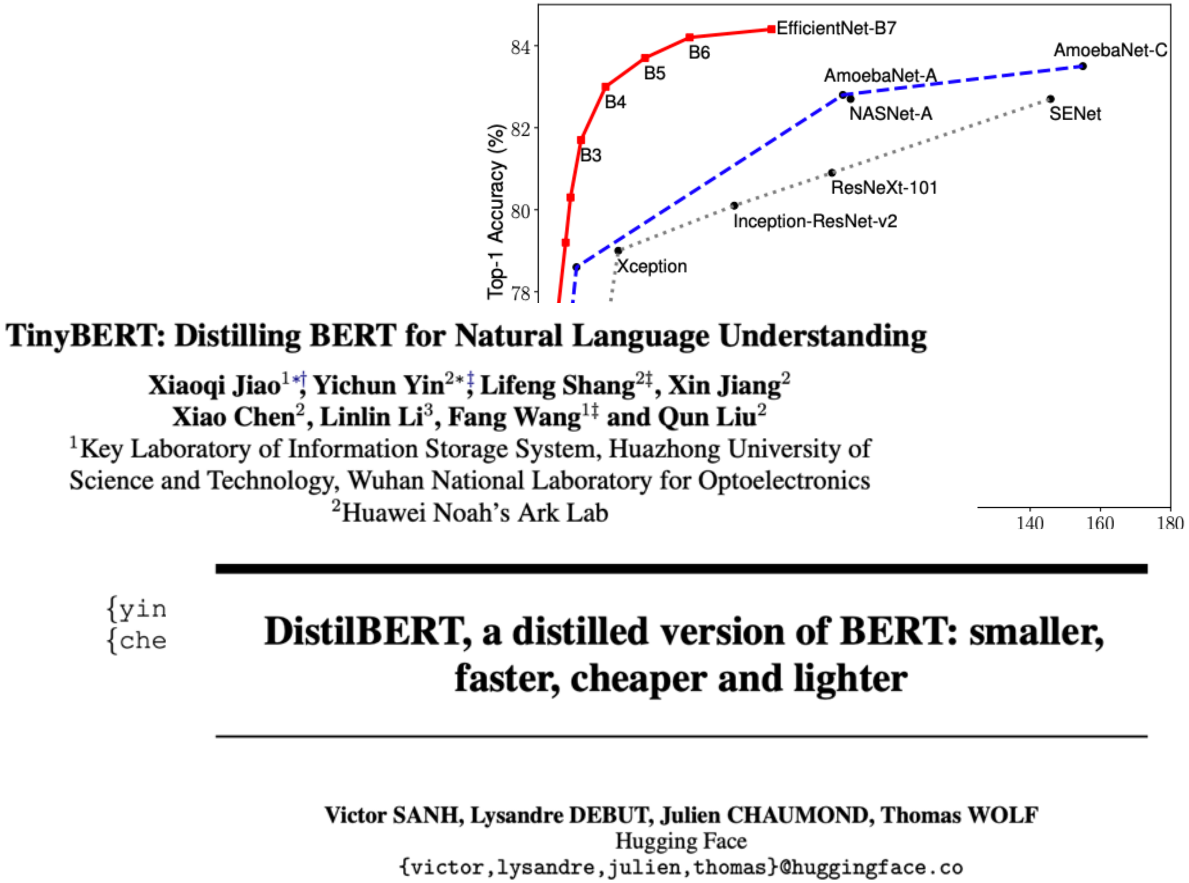


Ineffiicient execution model

Insufficient memory

Trend of SOTA Models

Highly detailed



TinyBERT: Distilling BERT for Natural Language Understanding

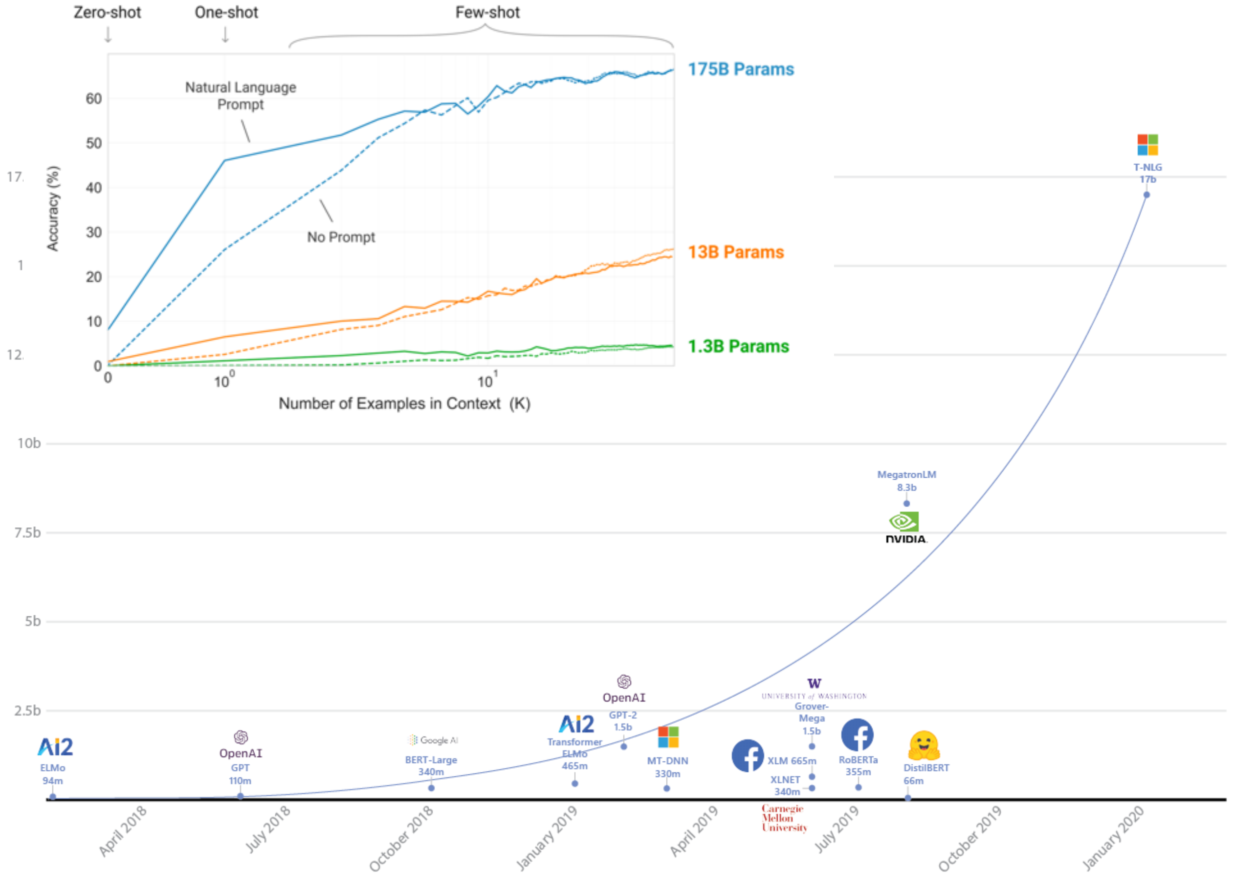
Xiaoqi Jiao[1*†], Yichun Yin[2*‡], Lifeng Shang[2‡], Xin Jiang[2]
Xiao Chen[2], Linlin Li[3], Fang Wang[1‡] and Qun Liu[2]
[1]Key Laboratory of Information Storage System, Huazhong University of Science and Technology, Wuhan National Laboratory for Optoelectronics
[2]Huawei Noah's Ark Lab

{yin
{che

DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter

Victor SANH, Lysandre DEBUT, Julien CHAUMOND, Thomas WOLF
Hugging Face
{victor,lysandre,julien,thomas}@huggingface.co

Bigger Models



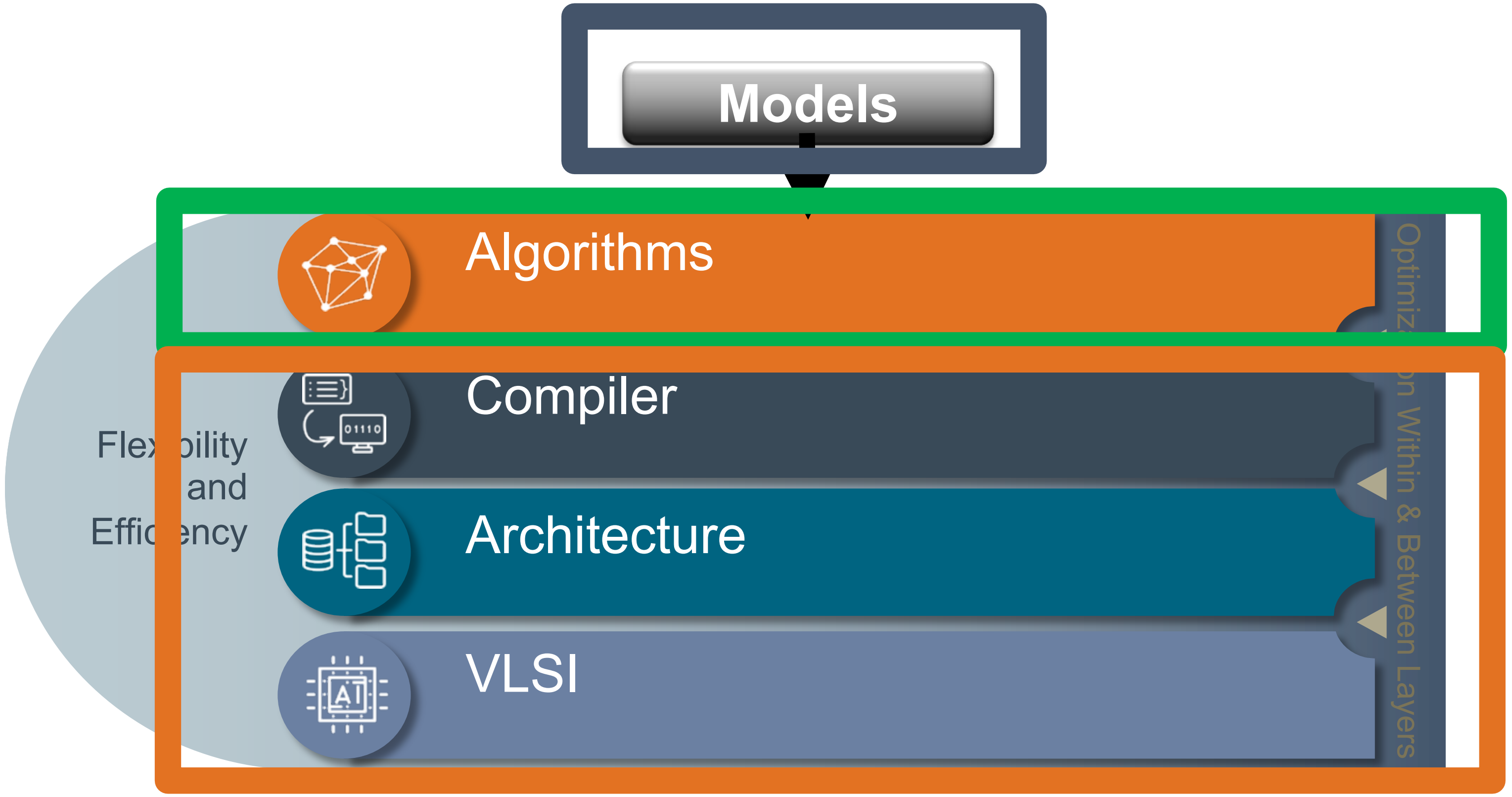
Our Mission

Shaping the next-generation ML / DL computing system to accelerate the full model spectrum

How do we break out of the Godilocks Zone?

Fundamental advances required at all layers of the SW/HW stack.

The SambaNova Systems Advantage


Models
Algorithms
Compiler
Architecture
VLSI
Flexibility and Efficiency
Optimization Within & Between Layers

Application innovations

High model accuracy

High compute efficiency

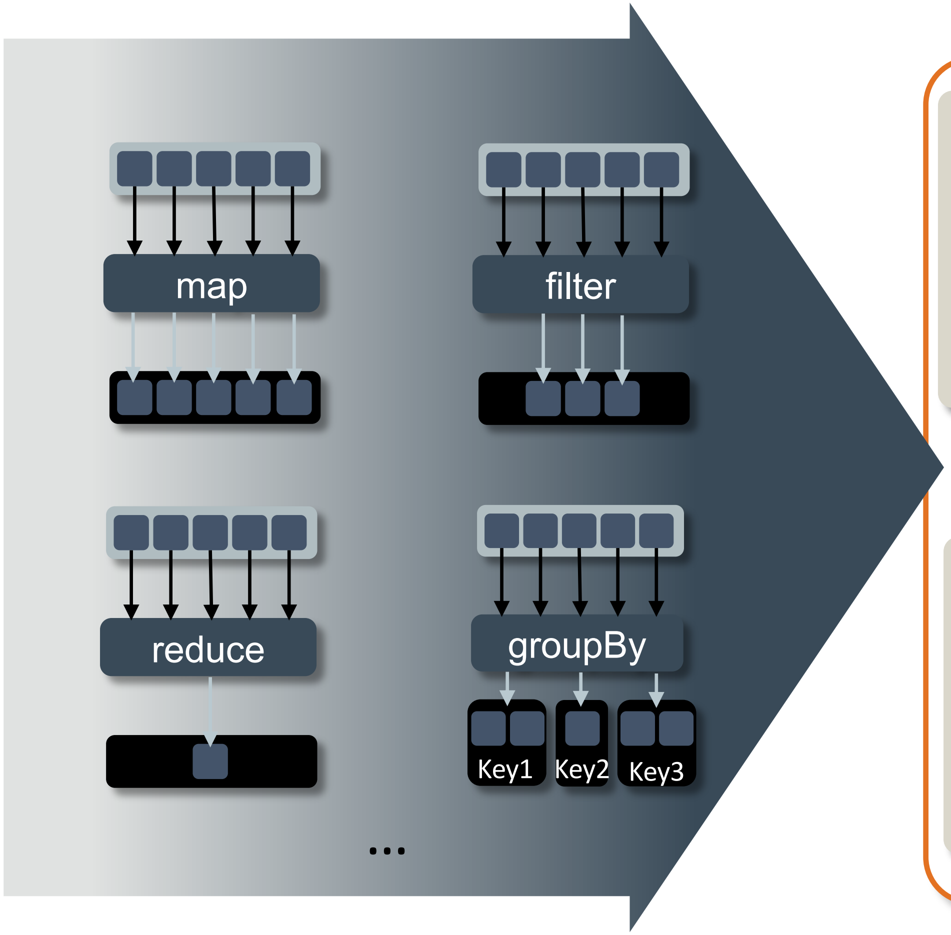
©2021 SambaNova Systems

Part 1.

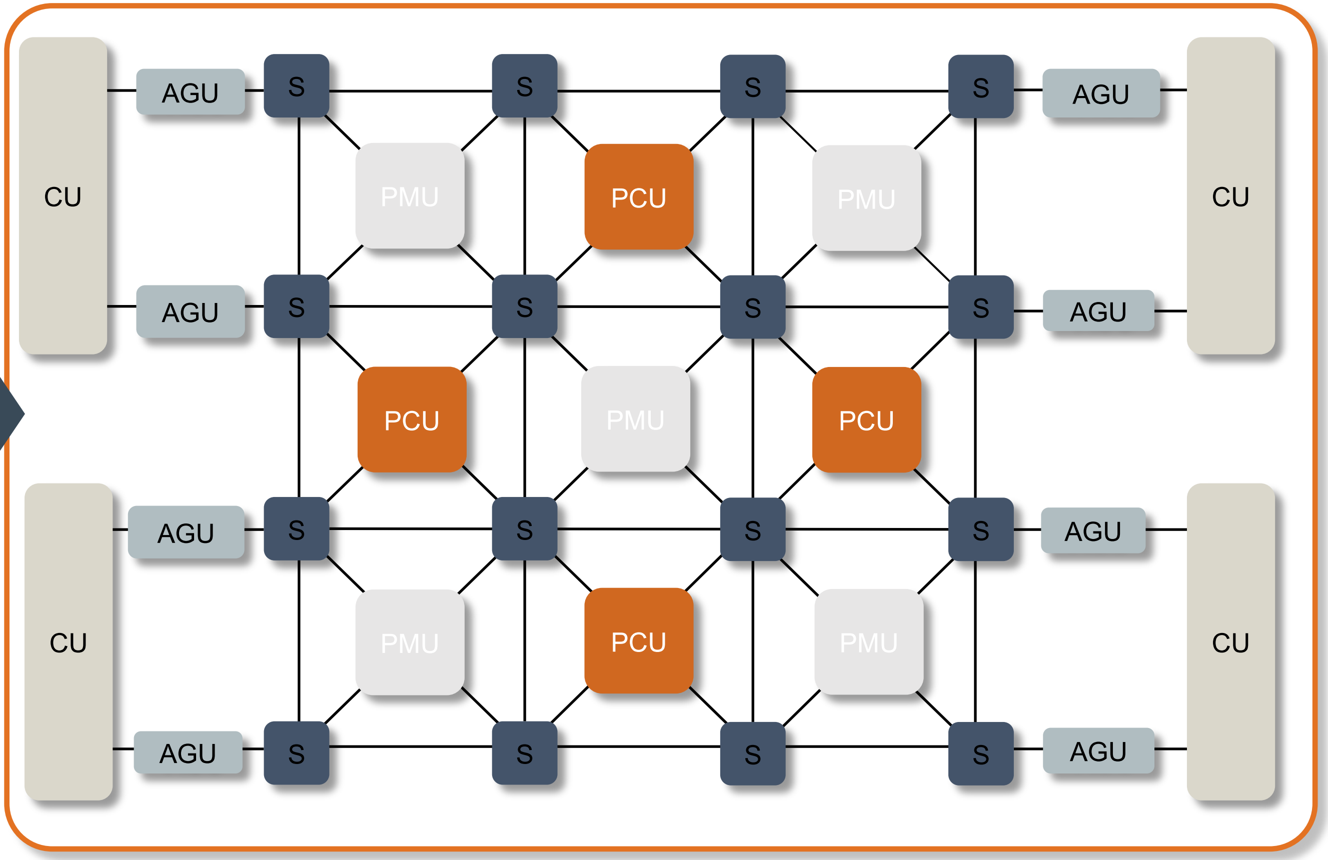
Enabling higher compute efficiency

Architecture: Reconfigurable Dataflow Unit (RDU)

Parallel Patterns

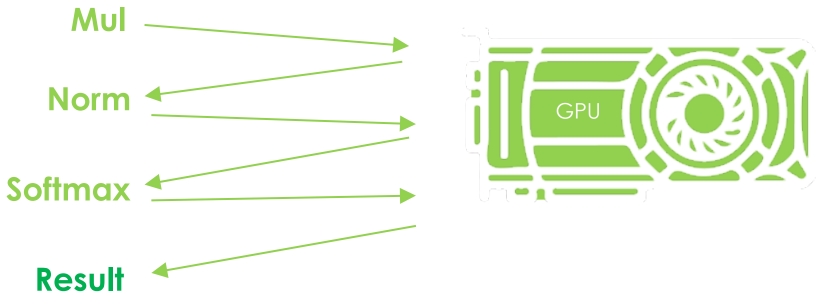


Array of reconfigurable compute, memory and communication

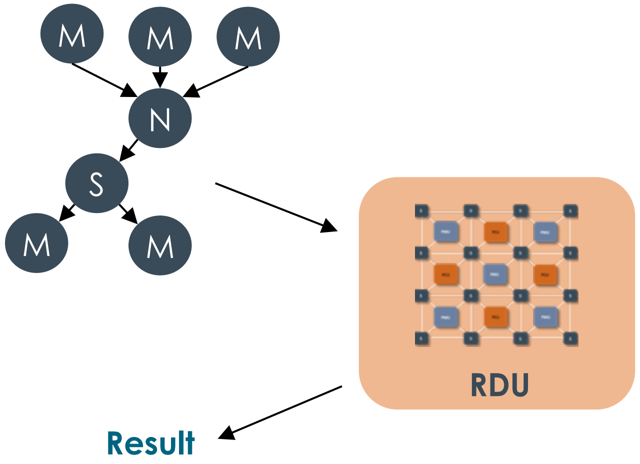


Spatial Dataflow Within an RDU

The old way: kernel-by-kernel

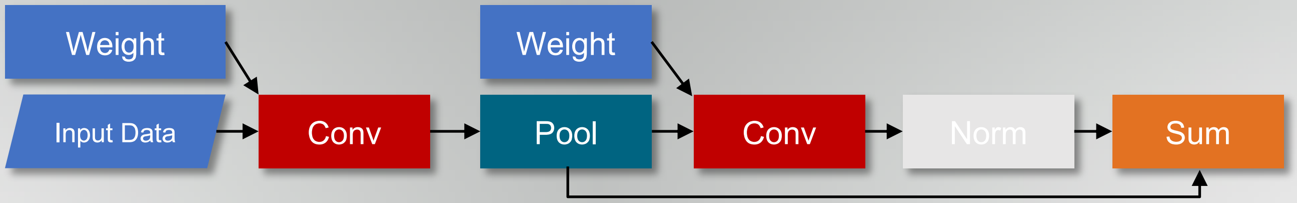


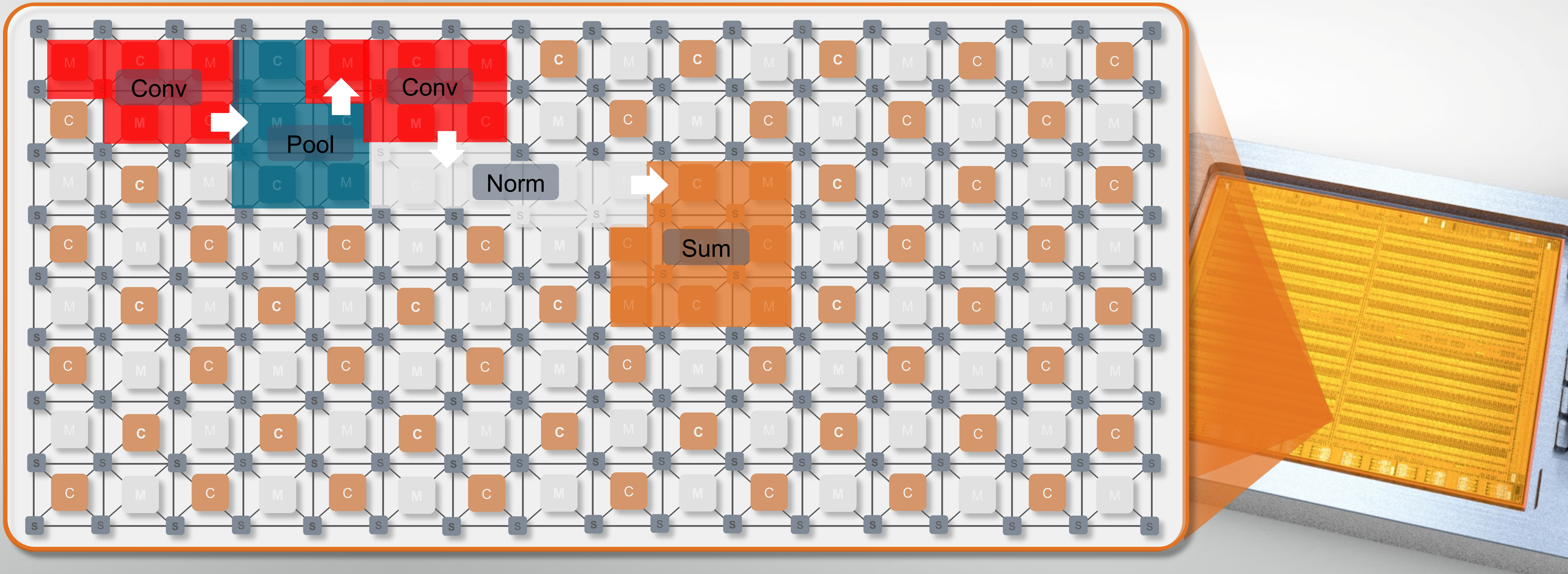
The Dataflow way: spatial



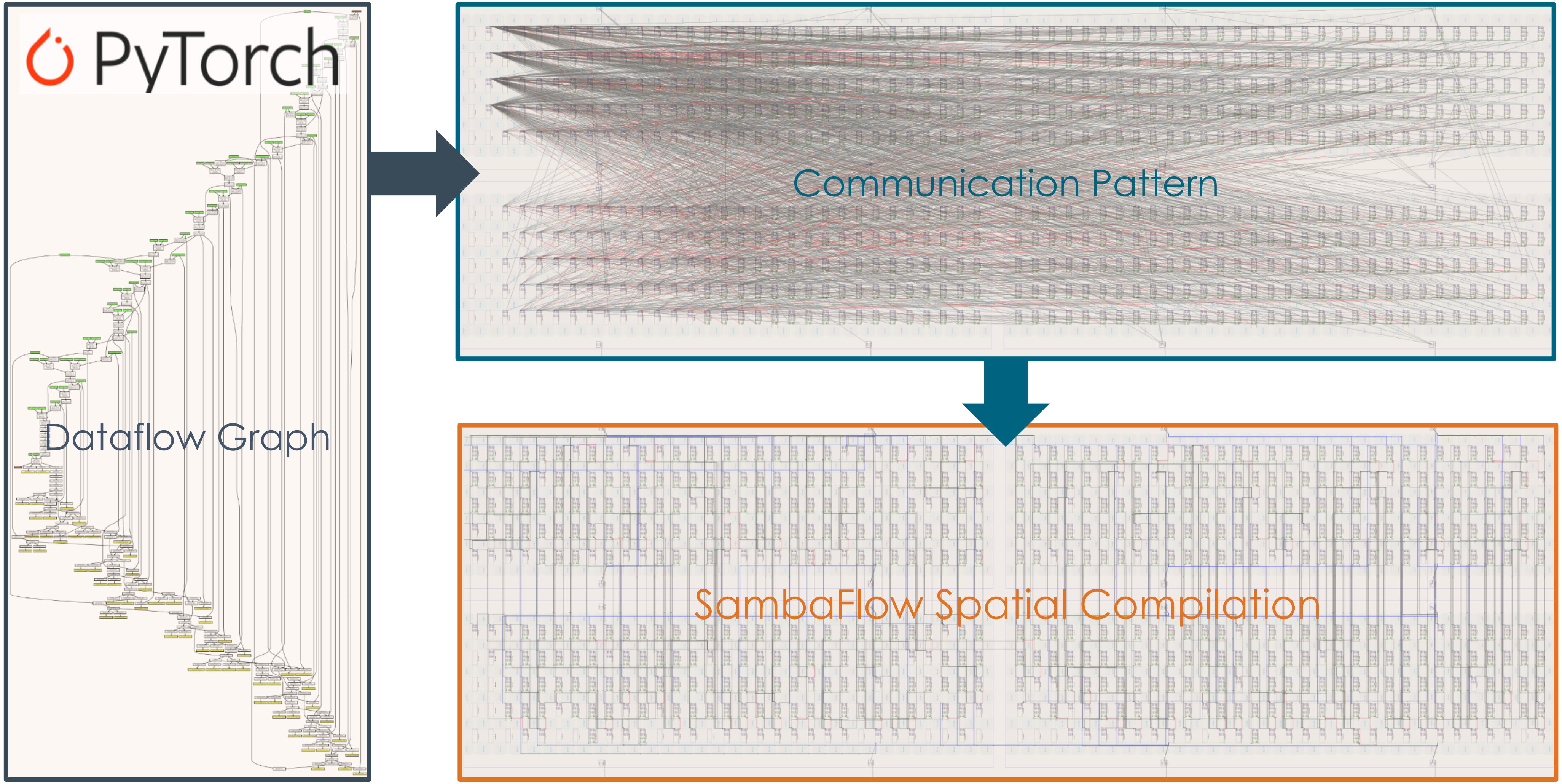
SambaFlow eliminates overhead and maximizes utilization

Rapid Dataflow Compilation to RDU





SambaFlow Produces Highly Optimized Spatial Mappings



PyTorch

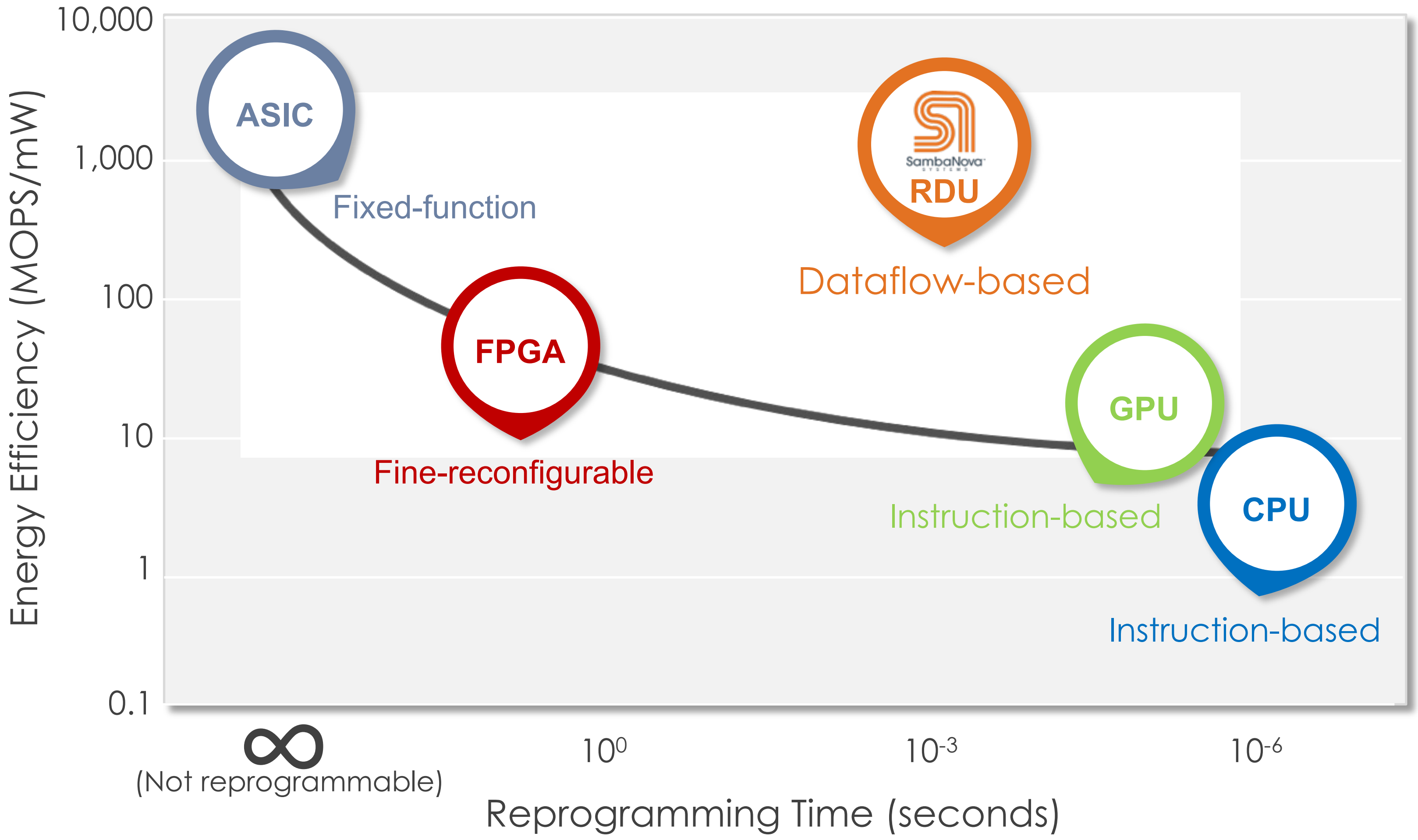
Dataflow Graph

Communication Pattern

SambaFlow Spatial Compilation

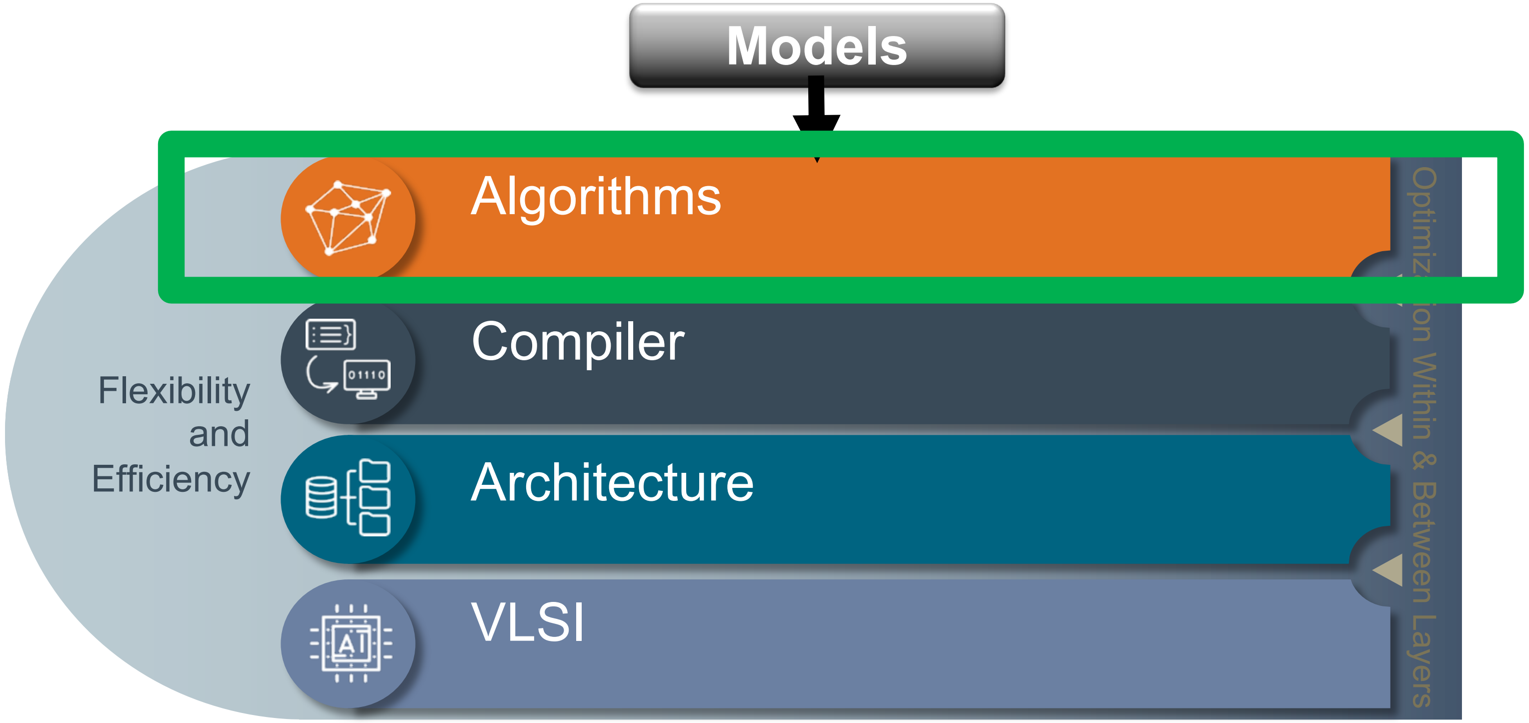
Uncompromised Programmability and Efficiency

Breaking out of the programmability vs. efficiency tradeoff curve



The SambaNova Systems Advantage

Achieve low time-to-accuracy

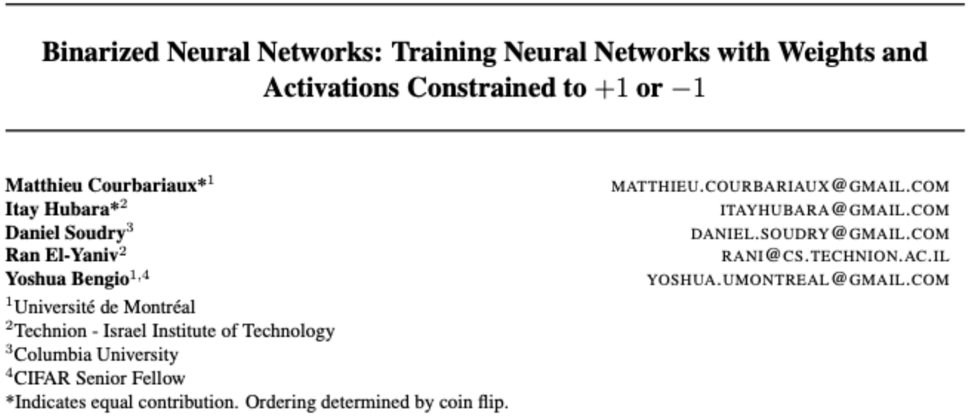

Models
Algorithms
Compiler
Architecture
VLSI
Flexibility and Efficiency
Optimization Within & Between Layers

High model accuracy

Part 2. High model accuracy:

+ Pure 16-bit FPU training
+ Asynchronous pipeline parallelization

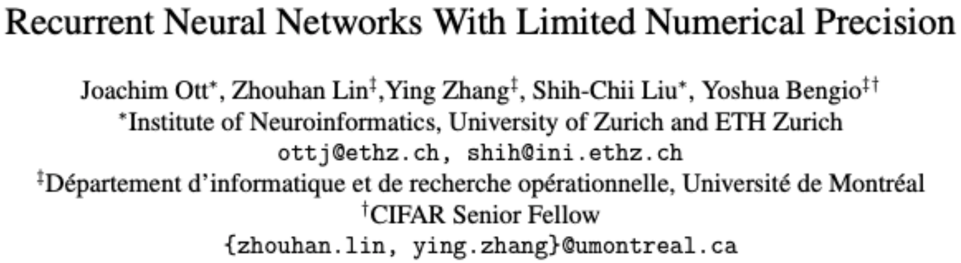
Low Precision (< 32-bit) Training



Binarized Neural Networks: Training Neural Networks with Weights and Activations Constrained to $+1$ or -1

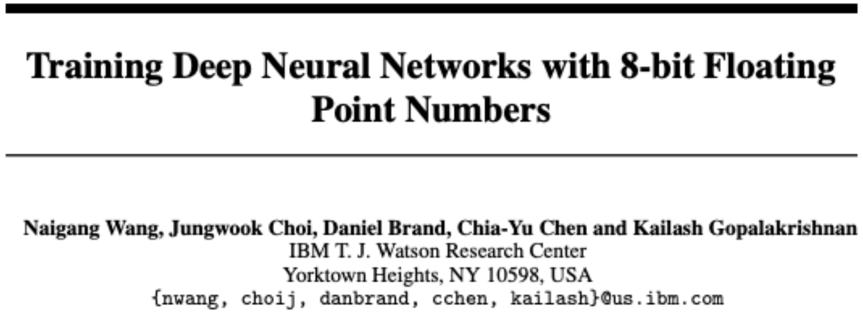
Matthieu Courbariaux*[1] MATTHIEU.COURBARIAUX@GMAIL.COM
Itay Hubara*[2] ITAYHUBARA@GMAIL.COM
Daniel Soudry[3] DANIEL.SOUDRY@GMAIL.COM
Ran El-Yaniv[2] RANI@CS.TECHNION.AC.IL
Yoshua Bengio[1,4] YOSHUA.UMONTREAL@GMAIL.COM

[1] Université de Montréal
[2] Technion - Israel Institute of Technology
[3] Columbia University
[4] CIFAR Senior Fellow
*Indicates equal contribution. Ordering determined by coin flip.



Recurrent Neural Networks With Limited Numerical Precision

Joachim Ott*, Zhouhan Lin‡, Ying Zhang‡, Shih-Chii Liu*, Yoshua Bengio‡†
*Institute of Neuroinformatics, University of Zurich and ETH Zurich
ottj@ethz.ch, shih@ini.ethz.ch
‡Département d'informatique et de recherche opérationnelle, Université de Montréal
†CIFAR Senior Fellow
{zhouhan.lin, ying.zhang}@umontreal.ca



Training Deep Neural Networks with 8-bit Floating Point Numbers

Naigang Wang, Jungwook Choi, Daniel Brand, Chia-Yu Chen and Kailash Gopalakrishnan
IBM T. J. Watson Research Center
Yorktown Heights, NY 10598, USA
{nwang, choij, danbrand, cchen, kailash}@us.ibm.com

Higher system efficiency, minimal impact on acc. for **specific models**

Efficiency of Low Precision Floating-point-units (16 vs. 32-bit)[1,2]

1.5X lower chip area

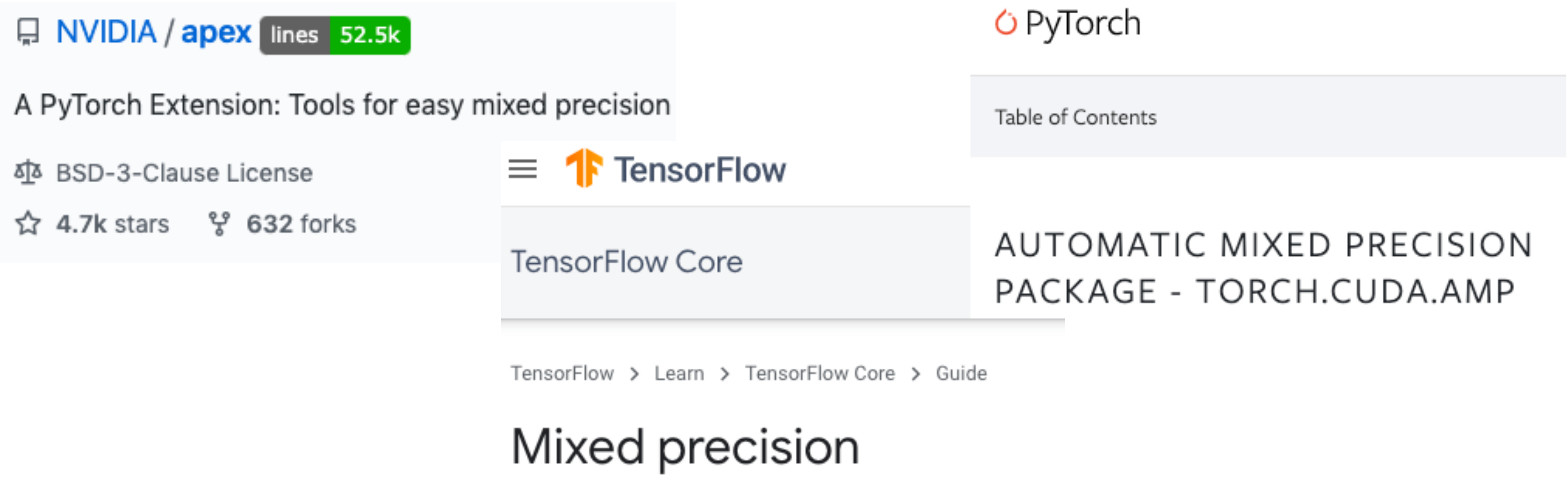
3X higher energy efficiency

1.5X higher throughput

[1] Horowitz. ISSCC 2014

[2] Galal et. al. ISCA 2013

Mixed Precision for *Generic* DL Training (16 + 32 bits FPU)



NVIDIA / apex lines 52.5k

A PyTorch Extension: Tools for easy mixed precision

BSD-3-Clause License

☆ 4.7k stars 632 forks

TensorFlow

TensorFlow Core

TensorFlow > Learn > TensorFlow Core > Guide

Mixed precision

PyTorch

Table of Contents

AUTOMATIC MIXED PRECISION PACKAGE - TORCH.CUDA.AMP

Illusion:

16-bit FPU alone is not enough to maximize model acc.

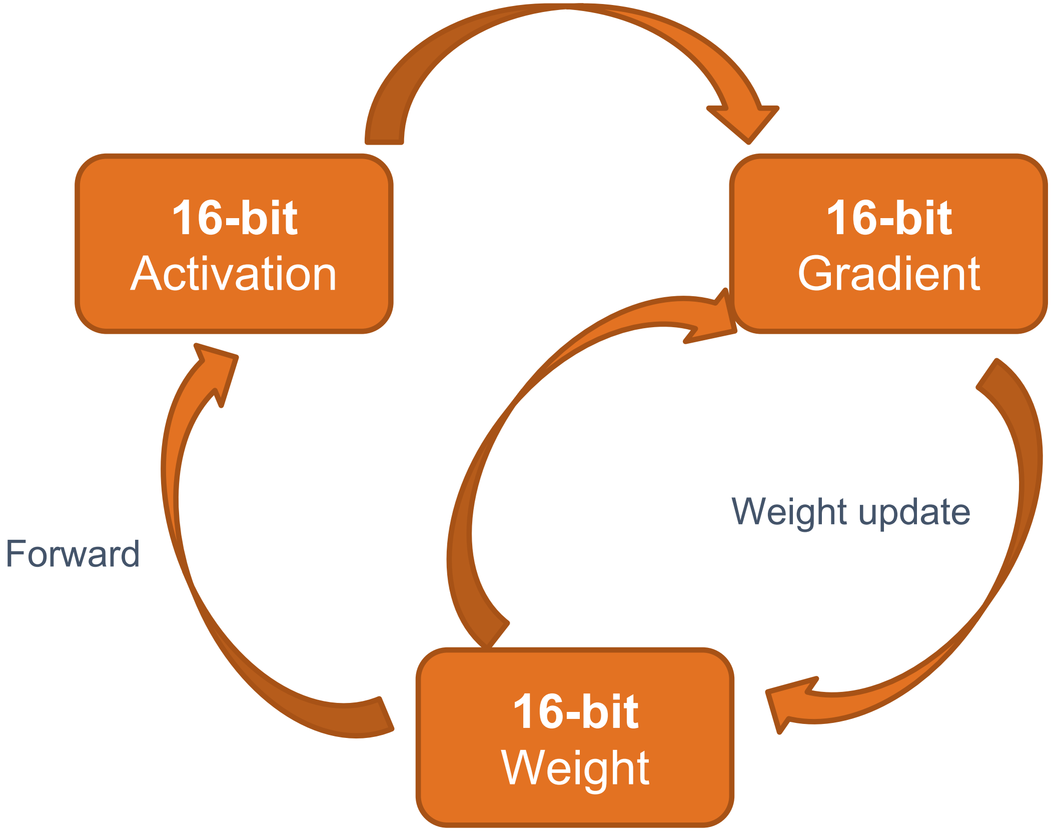
*Can we support **only 16-bit FPU** on accelerators*

&

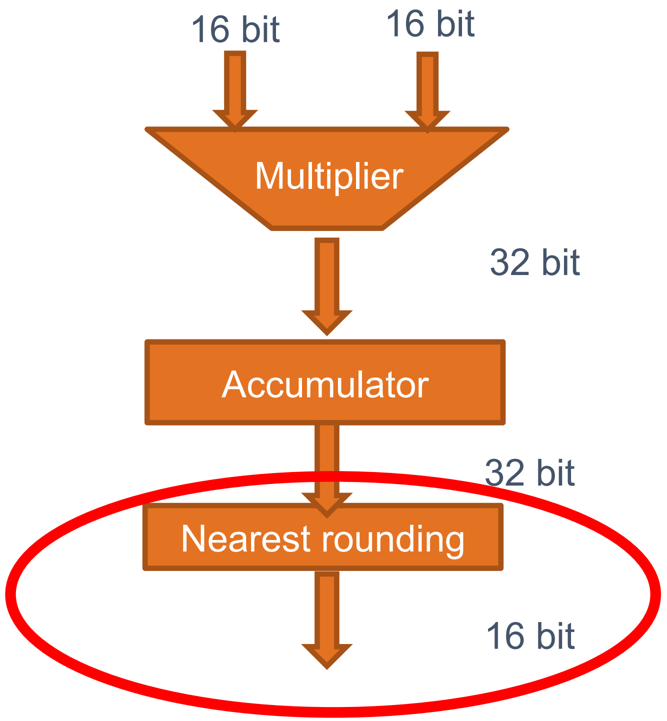
achieve model acc. matching 32-bit training?

Pure 16-bit (BFloat16) FPU Training

Data Flows

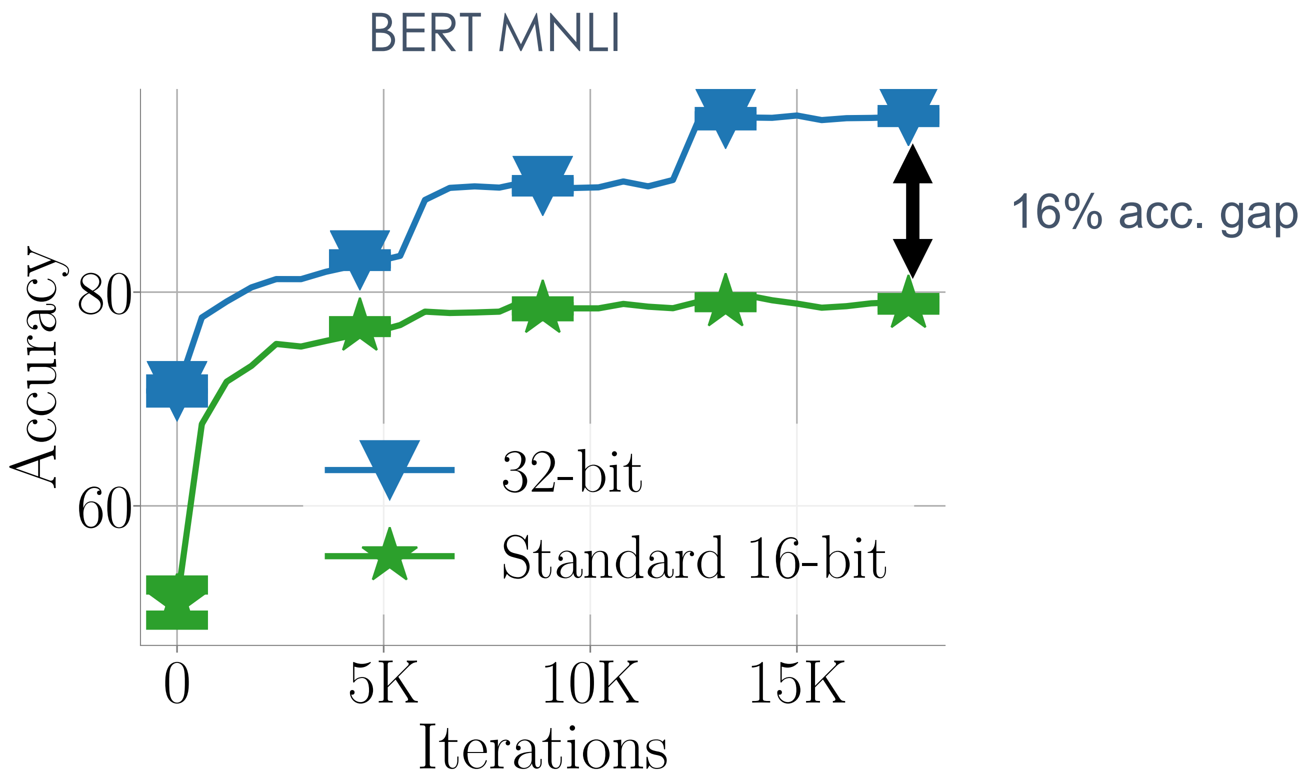


Multiply-Accumulation Units



Primary source of numerical error

The Accuracy Challenge



Standard 16-bit FPU training degrades model accuracy

The Devil: Nearest Rounding(NR) for Model Weight Updates

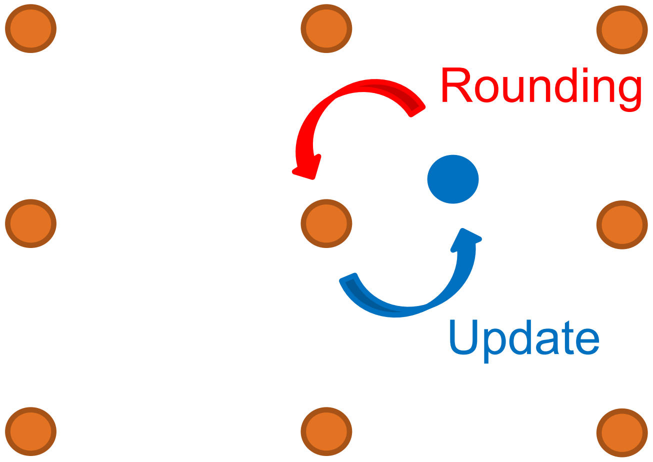
$$\boldsymbol{w}_{t+1} = \boldsymbol{Q}\left(\boldsymbol{w}_t - \alpha \nabla f_{\sigma(t)}(\boldsymbol{w}_t)\right)$$

Model weight

Nearest Rounding

Minibatch

Model weights halt when updates becomes small



The Devil: Nearest Rounding (NR) for Model Weight Updates

Theory sketch for least-squares regression

Machine precision

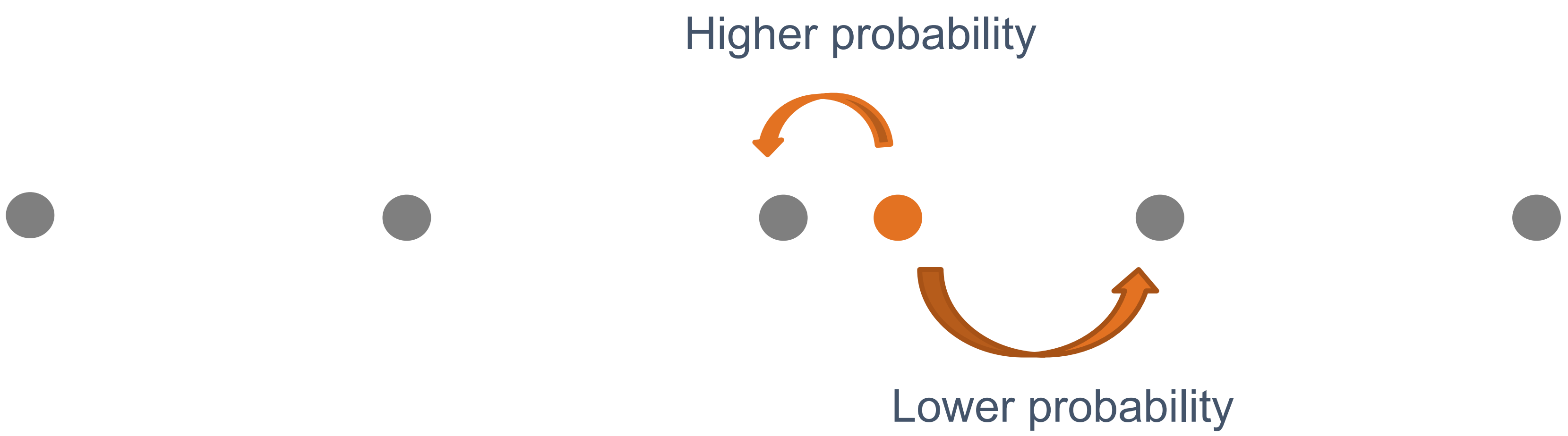
$$\left\|\boldsymbol{w}_t - \boldsymbol{w}^*\right\| \geq \mathcal{O}\left(\epsilon \cdot \min_j \left|w_j^*\right|\right)$$

Optimal solution

j-th dim of the optimal solution

Inaccurate weight update fundamentally degrades convergence

Stochastic Rounding to the Rescue

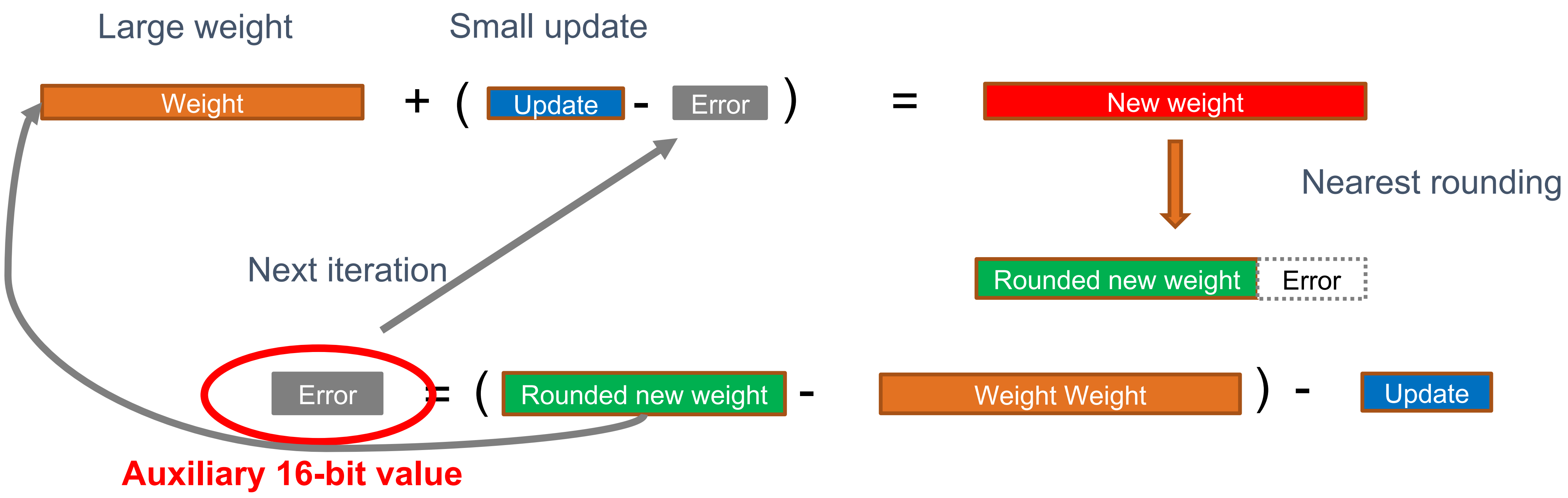


Intuition

The expectation of unbiased estimates is as accurate as weights w/o rounding

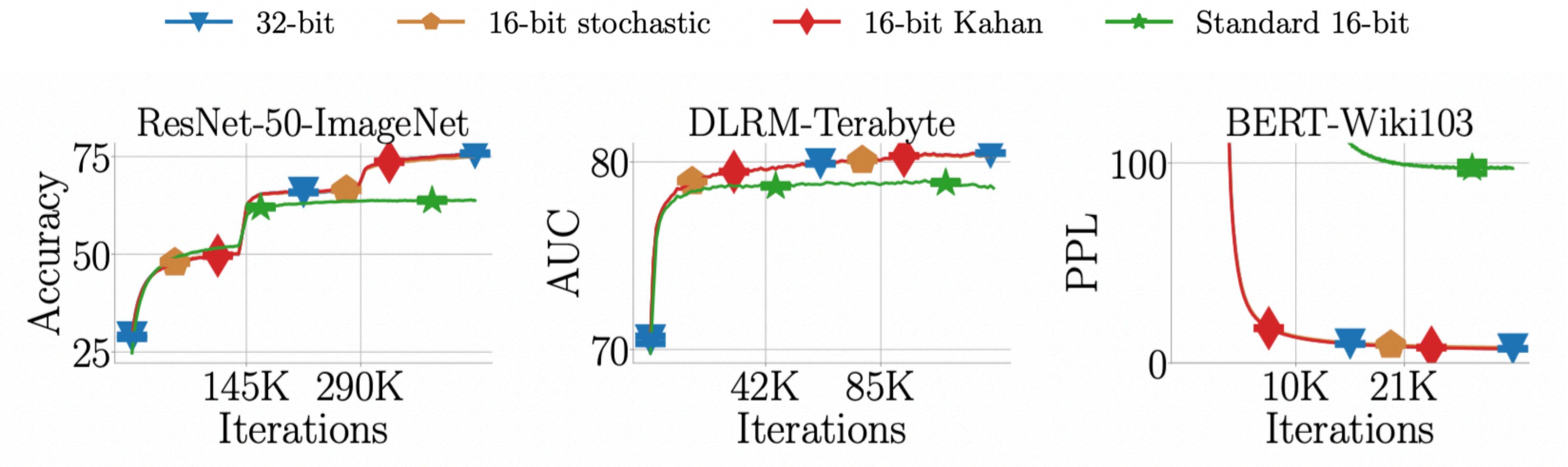
Kahan Summation as Alternative Enhancement

Auxiliary 16-bit values to track and correct weight update errors from NR



Experiment:

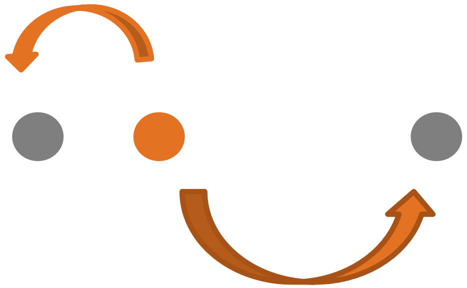
Pure 16-bit training can match 32-bit training in model acc.



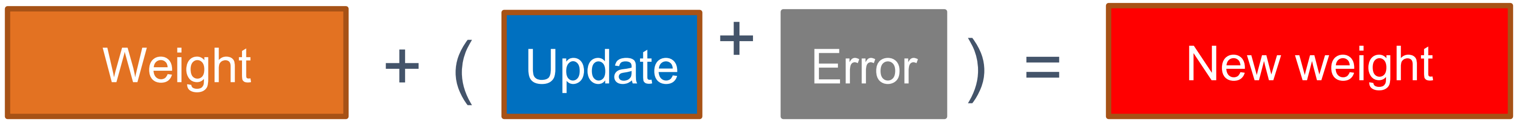
In submission ‘20

Summary

With support for



&



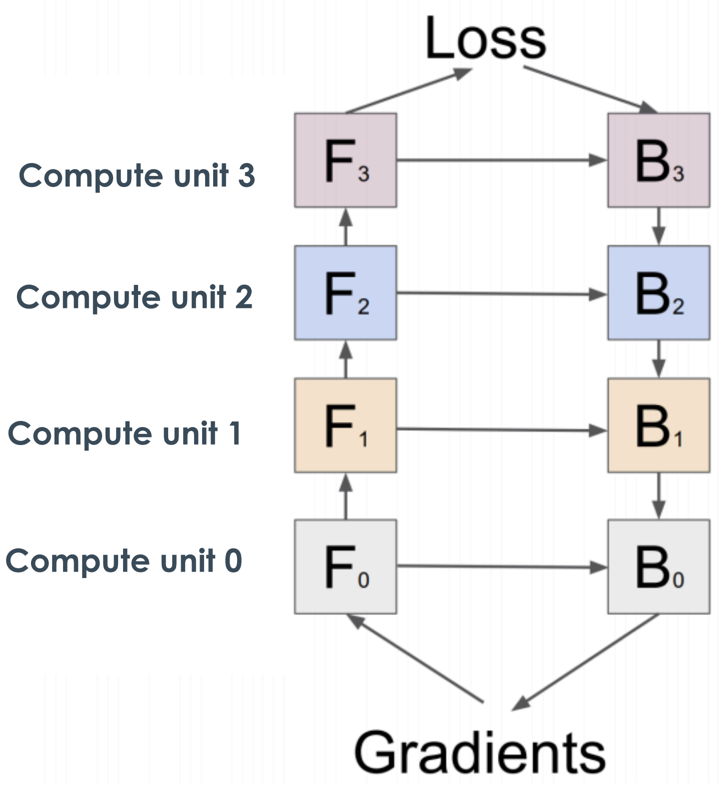
Weight + (Update + Error) = New weight

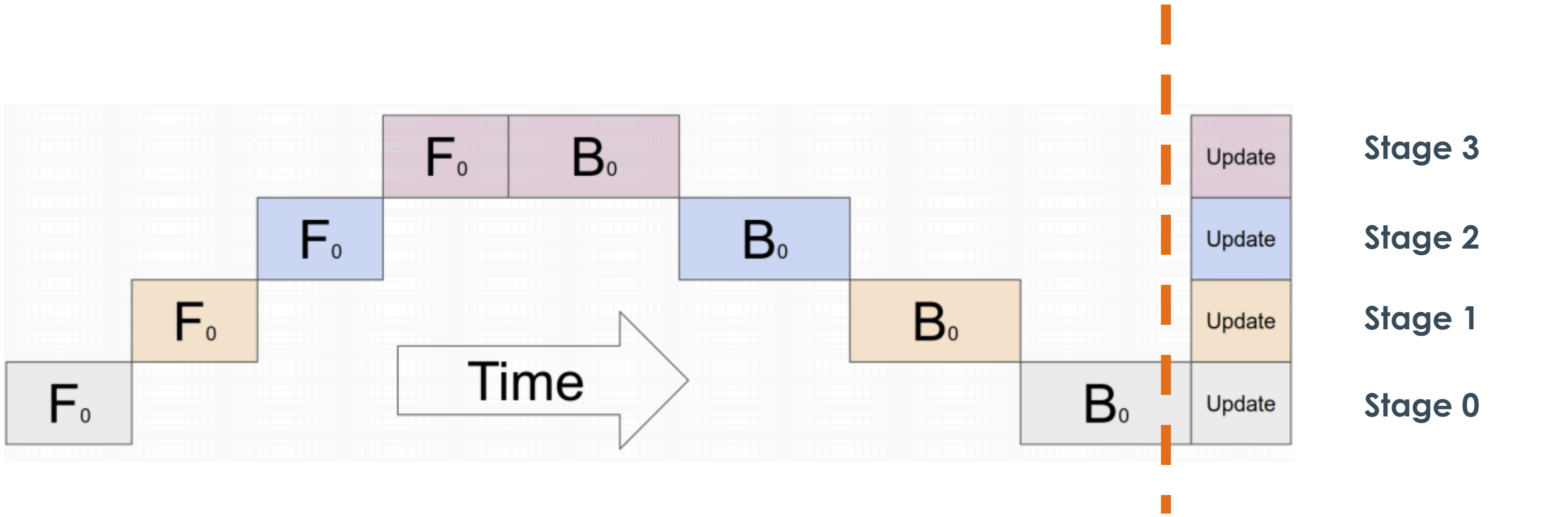
Stochastic rounding

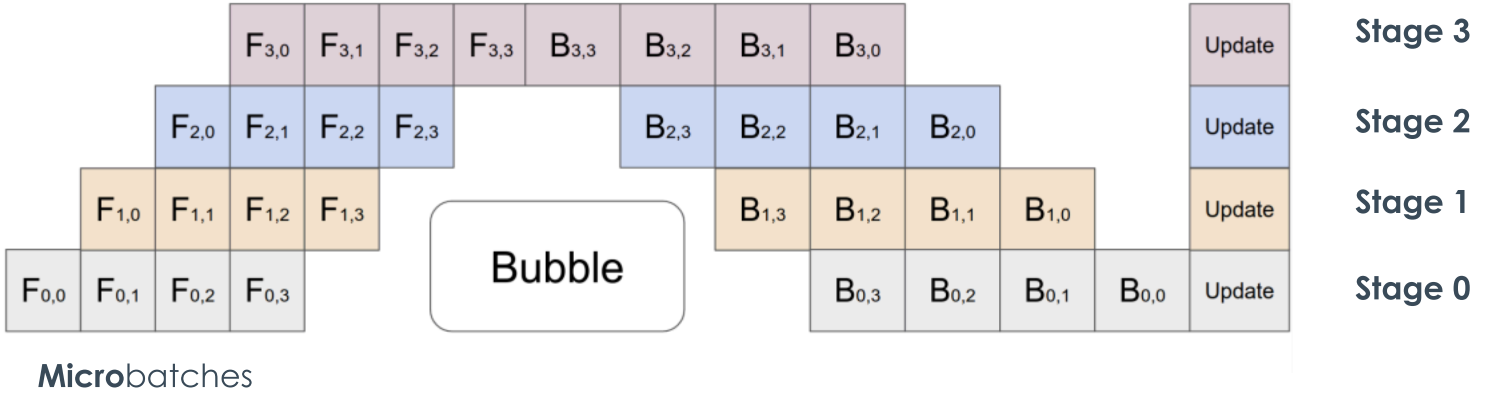
Kahan summation

Accelerators with only 16-bit compute units can match acc. of 32-bit training

Model (Pipeline) Parallelism






$F_{3,0}$ $F_{3,1}$ $F_{3,2}$ $F_{3,3}$ $B_{3,3}$ $B_{3,2}$ $B_{3,1}$ $B_{3,0}$
Update
Stage 3
$F_{2,0}$ $F_{2,1}$ $F_{2,2}$ $F_{2,3}$ $B_{2,3}$ $B_{2,2}$ $B_{2,1}$ $B_{2,0}$
Update
Stage 2
$F_{1,0}$ $F_{1,1}$ $F_{1,2}$ $F_{1,3}$ $B_{1,3}$ $B_{1,2}$ $B_{1,1}$ $B_{1,0}$
Update
Stage 1
$F_{0,0}$ $F_{0,1}$ $F_{0,2}$ $F_{0,3}$ $B_{0,3}$ $B_{0,2}$ $B_{0,1}$ $B_{0,0}$
Update
Stage 0
Bubble
Microbatches

1. Huang et. Neurips 2019

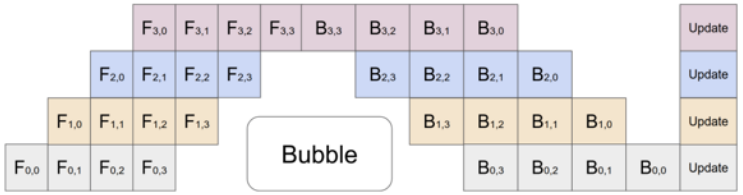
Model (Pipeline) Parallelism: Are we there yet?

Conventional processor pipeline

# of pipeline stages ↑ Throughput ↑

Model training pipeline with synchronization barrier

# of pipeline stages ↑ Utilization ↓

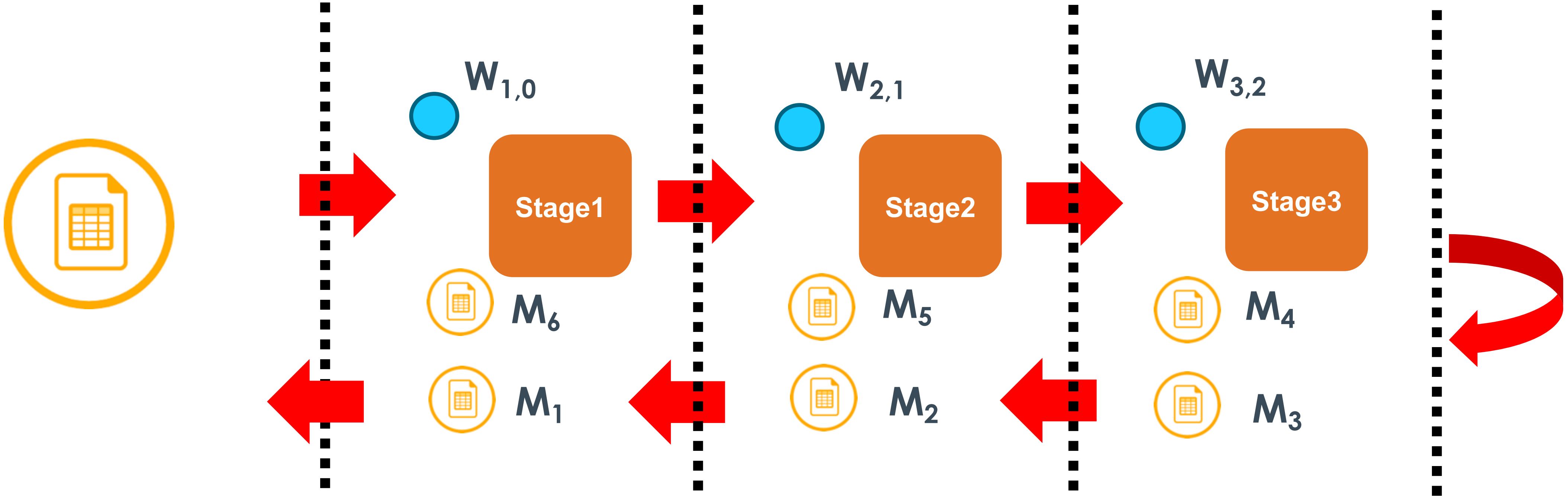


$F_{3,0}$ $F_{3,1}$ $F_{3,2}$ $F_{3,3}$ $B_{3,3}$ $B_{3,2}$ $B_{3,1}$ $B_{3,0}$ Update
$F_{2,0}$ $F_{2,1}$ $F_{2,2}$ $F_{2,3}$ $B_{2,3}$ $B_{2,2}$ $B_{2,1}$ $B_{2,0}$ Update
$F_{1,0}$ $F_{1,1}$ $F_{1,2}$ $F_{1,3}$ $B_{1,3}$ $B_{1,2}$ $B_{1,1}$ $B_{1,0}$ Update
$F_{0,0}$ $F_{0,1}$ $F_{0,2}$ $F_{0,3}$ Bubble $B_{0,3}$ $B_{0,2}$ $B_{0,1}$ $B_{0,0}$ Update

How much utilization do we really need to sacrifice?

Async. Pipeline Parallelism Steady State

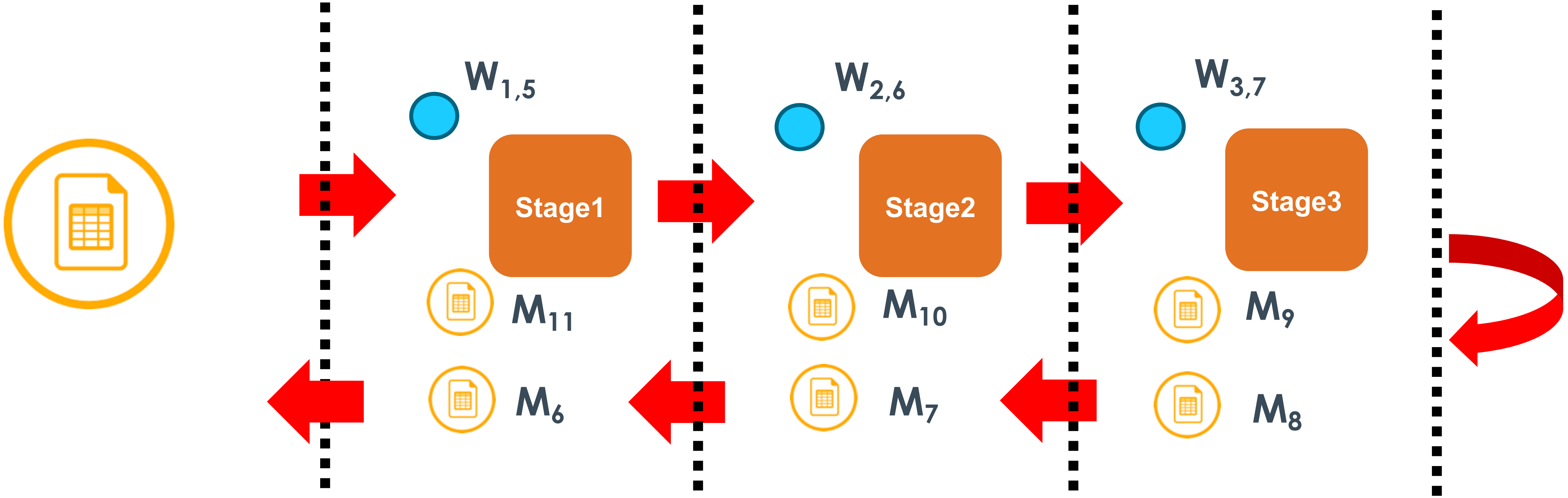
$W_{i,j}$ Stage i weight after j-th update M_i i-th minibatch



Goal: No hardware sacrifices!

Async. Pipeline Parallelism Steady State

$\mathbf{W_{i,j}}$ Stage i weight after j-th update $\mathbf{M_i}$ i-th minibatch

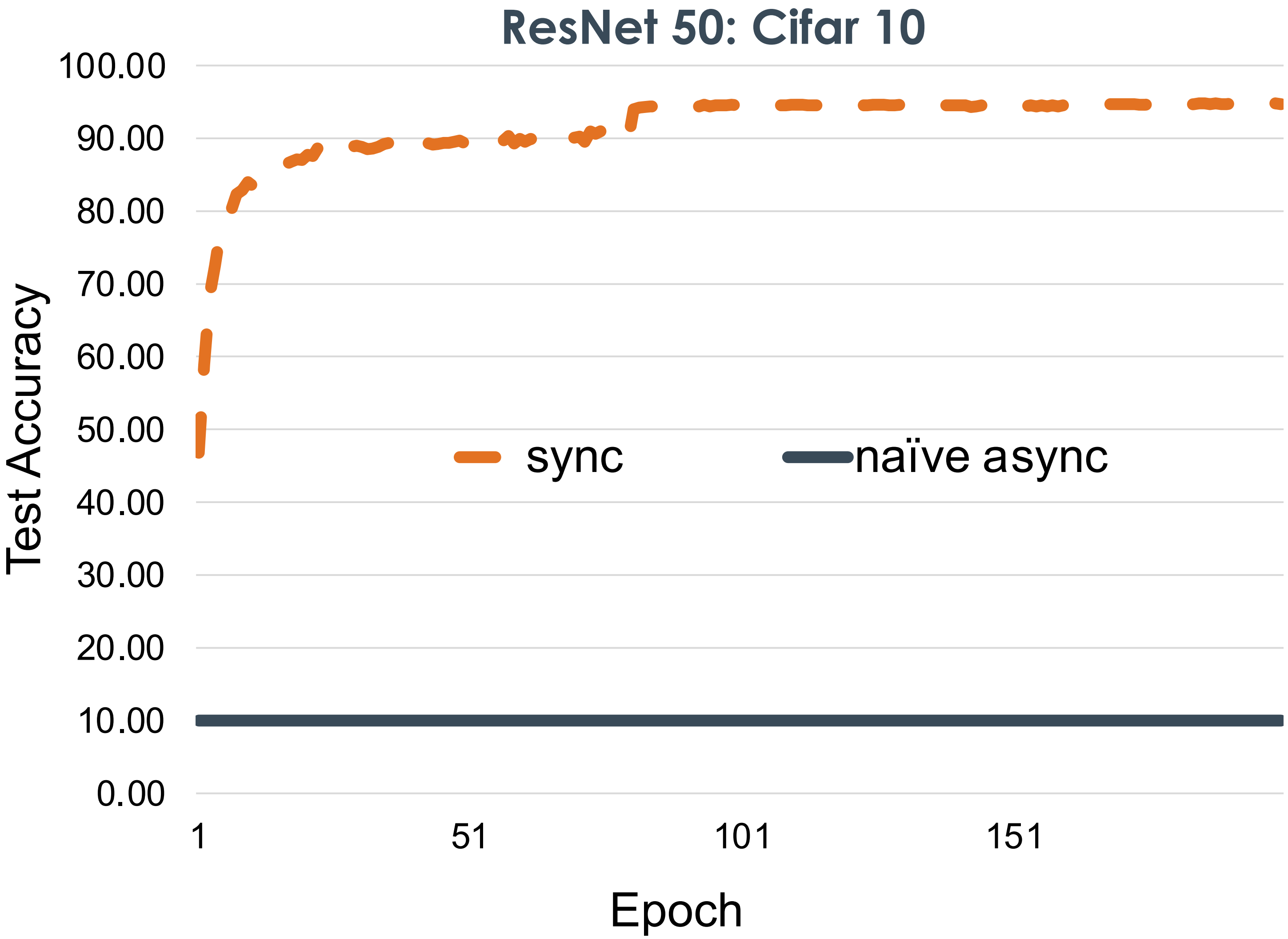


$\mathbf{M_6}$ uses $\mathbf{W_{1,0}}$ for forward and $\mathbf{W_{1,5}}$ for backward: delay = 5

$\mathbf{M_6}$ uses $\mathbf{W_{3,4}}$ for forward and $\mathbf{W_{3,5}}$ for backward: delay = 1

Panic: Introduces different **asynchrony** (delays) at different stages.

Houston, we have a problem.



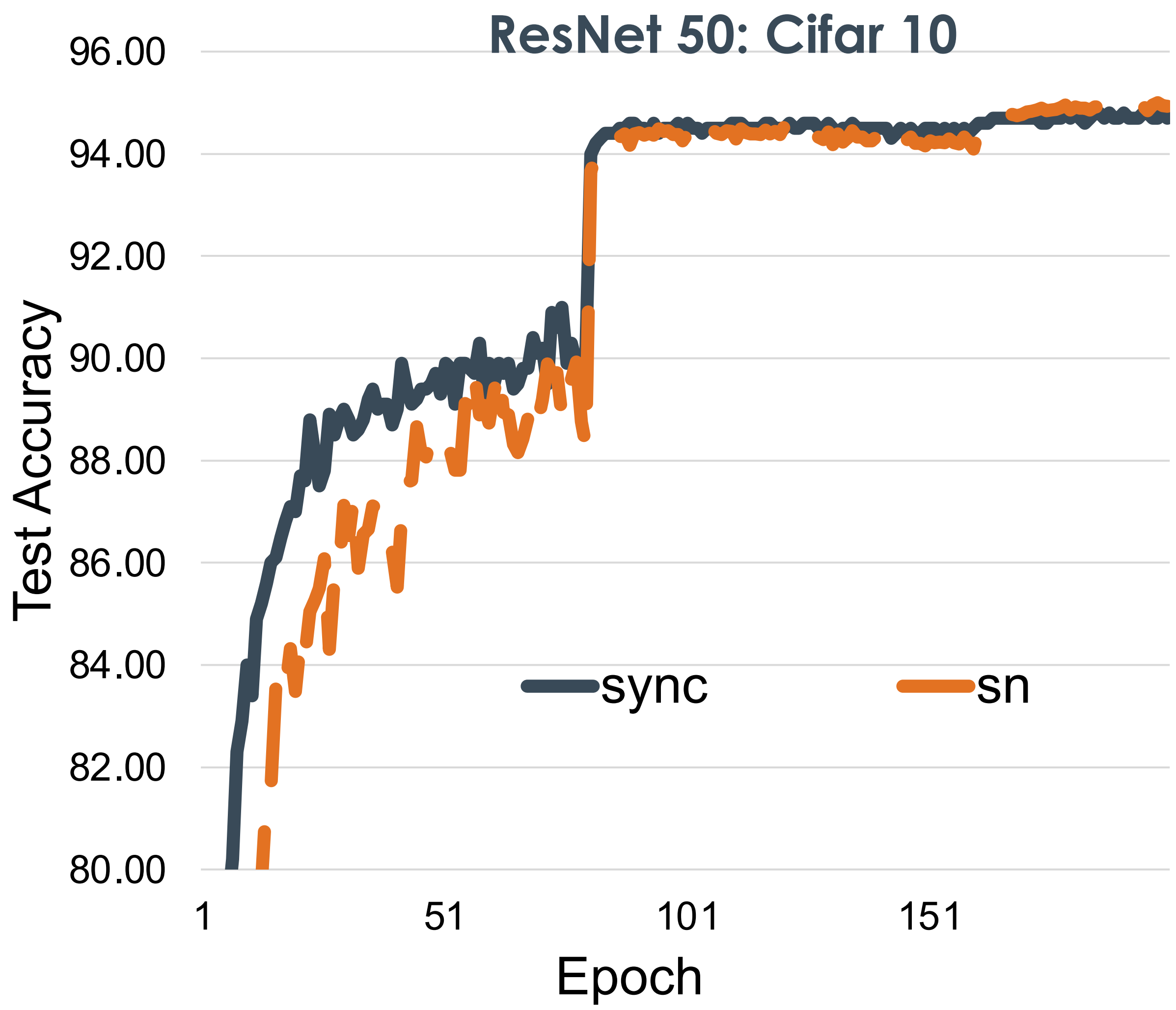
Key Insight: Scale your learning rate proportional to the delay.

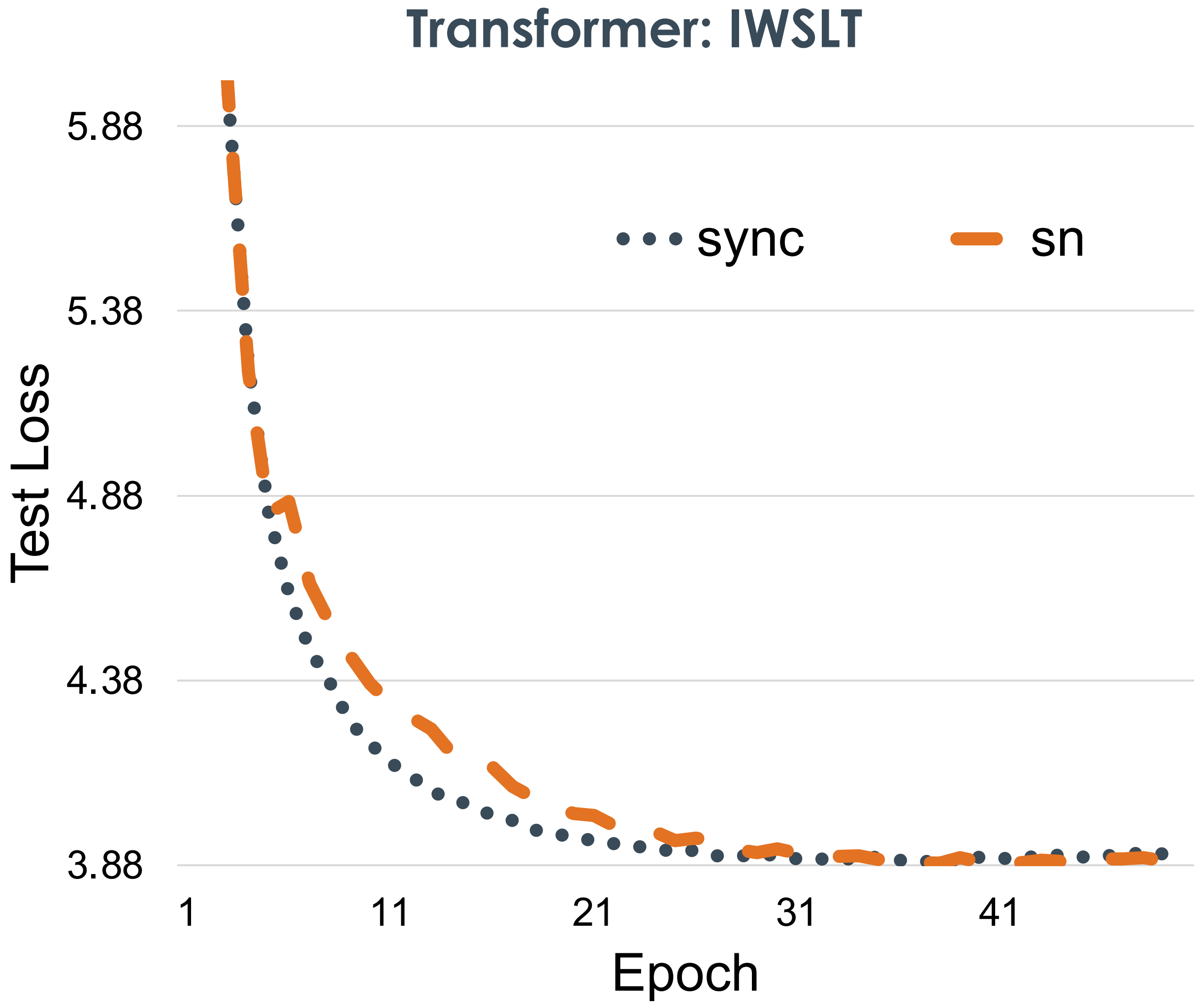
$$\alpha = \min\left(\alpha_{\text{sync}}, \frac{C}{\tau_i}\right)$$

Chris De Sa

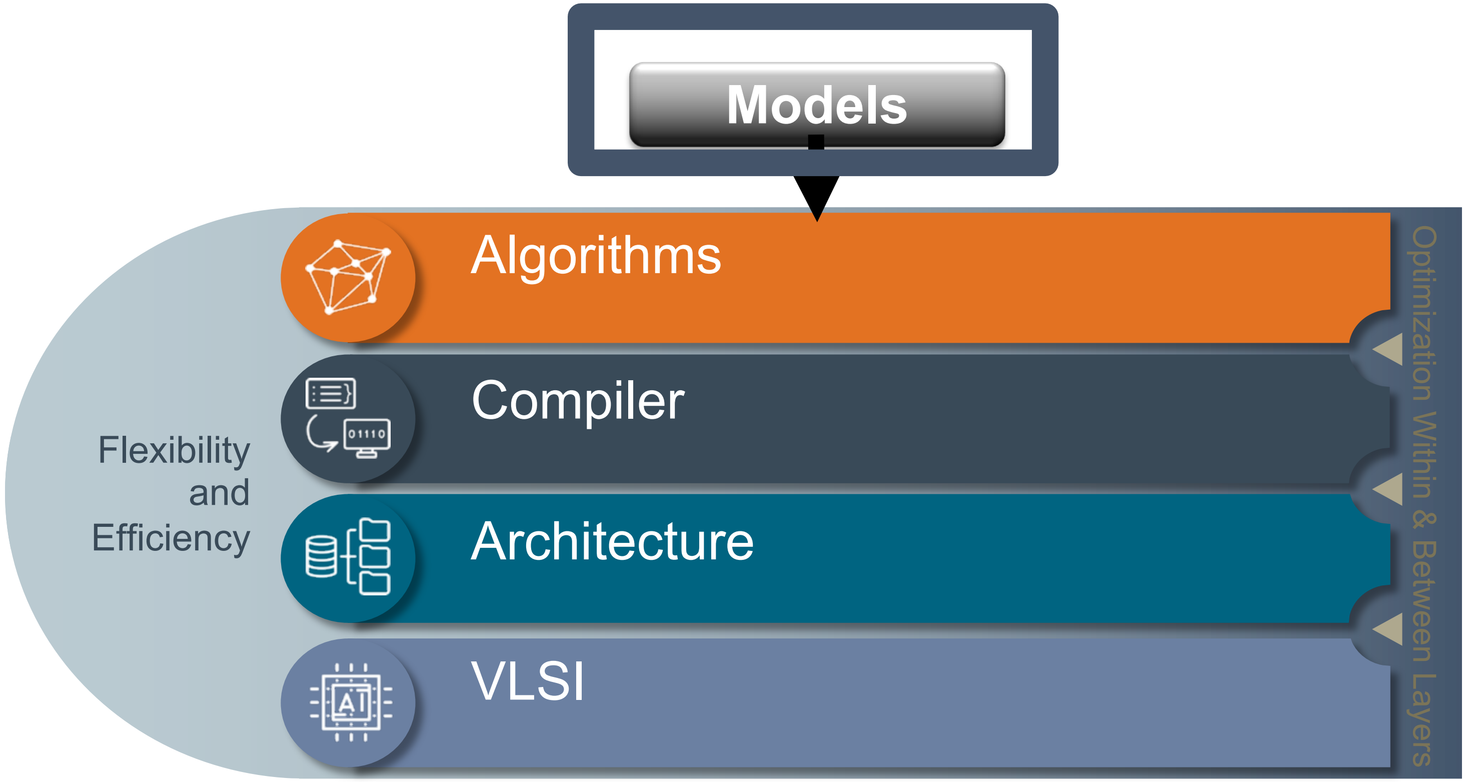
PipeMare: MLSys '20

Maximize efficiency with no accuracy compromise





The SambaNova Systems Advantage


Models
Algorithms
Compiler
Architecture
VLSI
Flexibility and Efficiency
Optimization Within & Between Layers

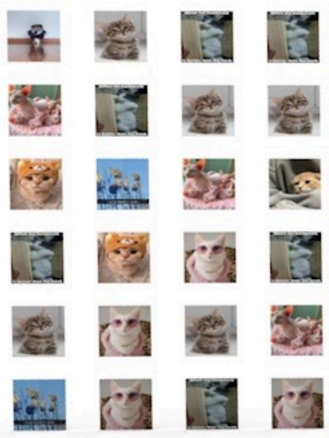
Application innovations

Part 3. Model Innovations:

Powered by our architecture and algorithm

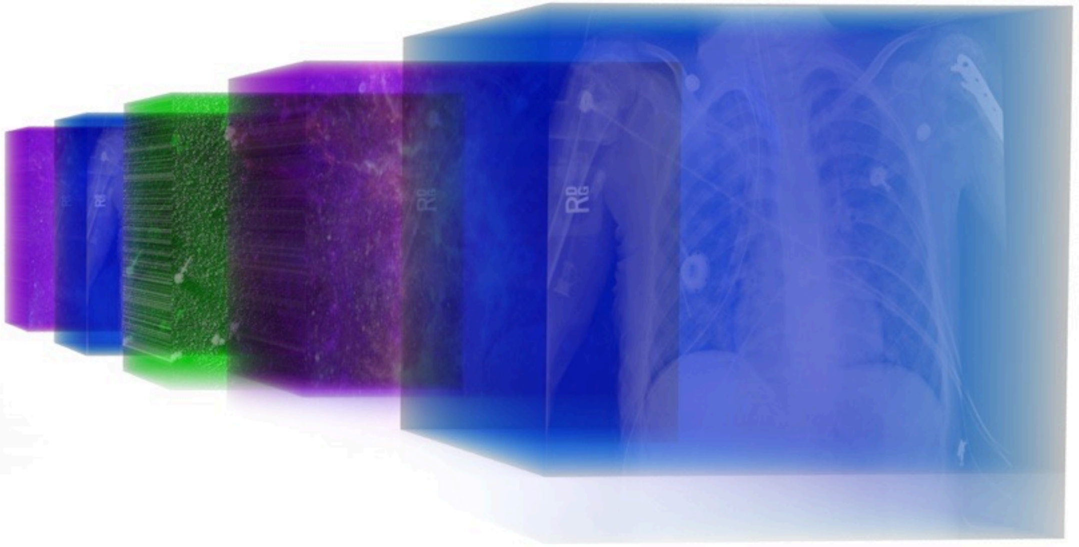
Computer Vison

Evolution of high-resolution Deep Learning



Low-resolution
(e.g. cats)

4k images
(e.g. Autonomous driving)



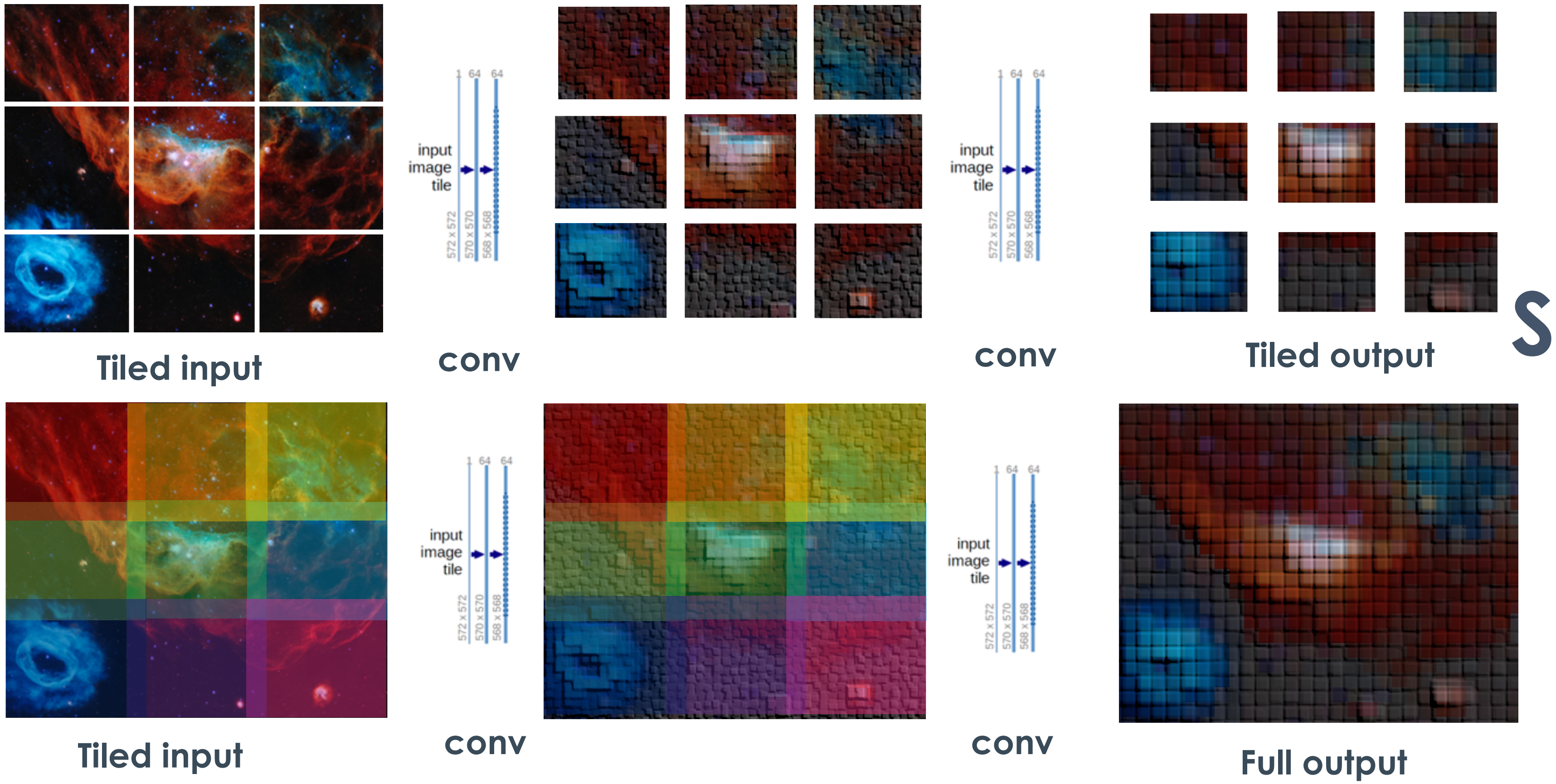
50k x 50k
(e.g. astronomy,
medical imaging, virus, …)

No Compromise High-Res Segmentation

Classic:
chop image into sub-images

Loses information in output!

SambaNova:
Full image processing



83.1%

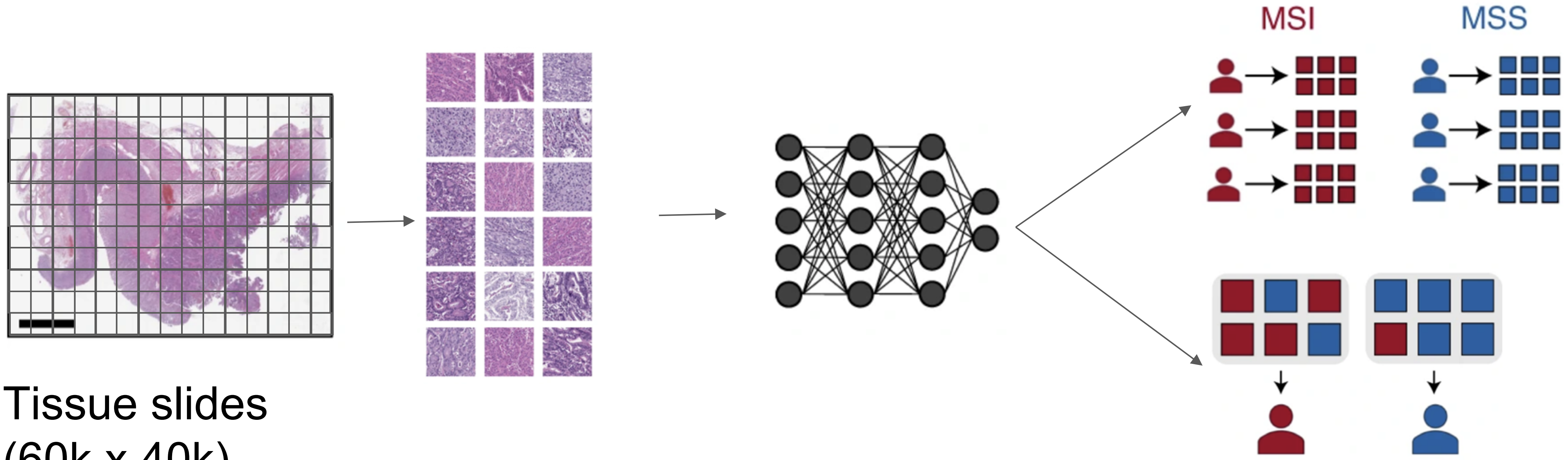
SOTA IOU

89.6%

Training w/o information loss from full-image processing

High-Res Pathology with Slide-level Label (TCGA)

Train with Patch label = slide label

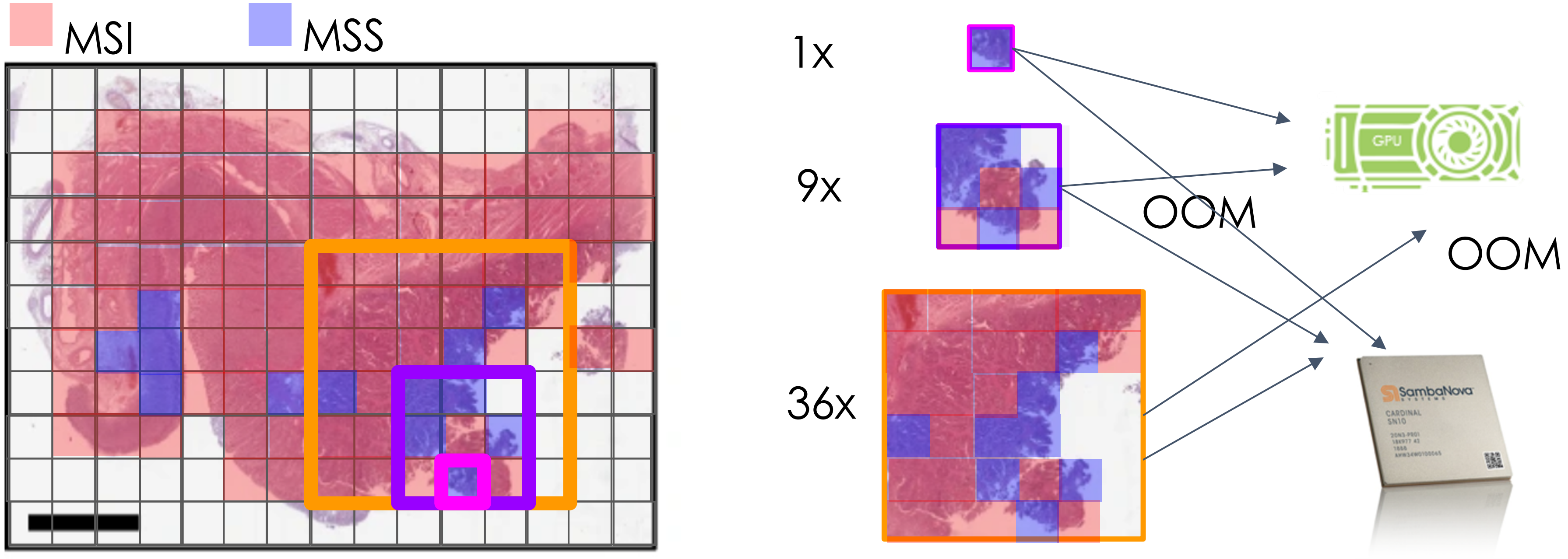


Tissue slides (60k x 40k)

Voting at slide level

Noisy patches limits model accuracy

High-Res Pathology with Slide-level label (TCGA)



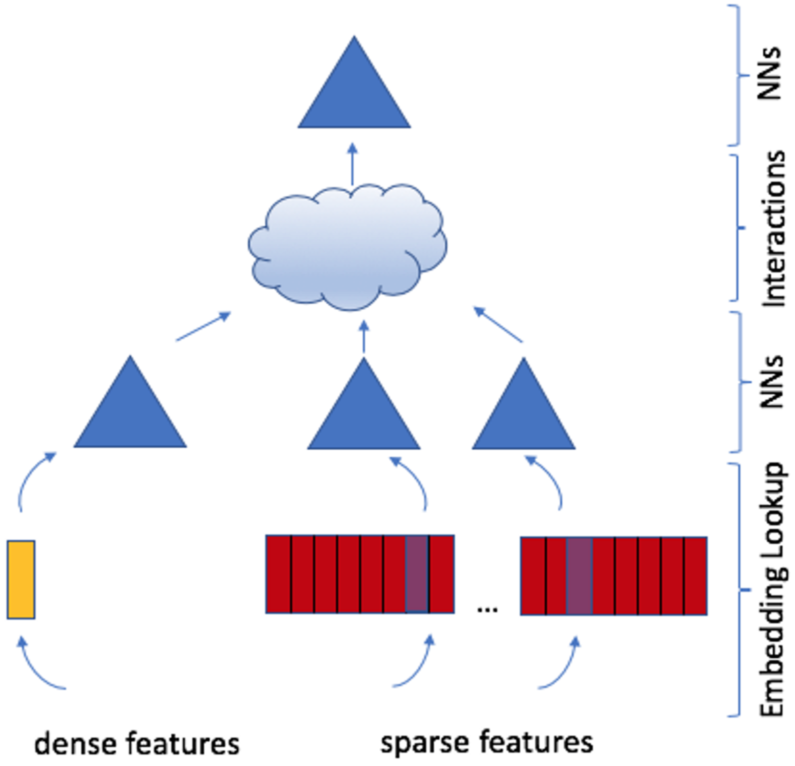
16X larger patches → 6 Pt higher AUC

Recommender Models

The backbone of many internet services

Entertainment	Social Media	E-Commerce	Consumer Services
	f	Alibaba.com®	lyft
hulu	in	amazon	mastercard
NETFLIX	p	ebay™	yelp

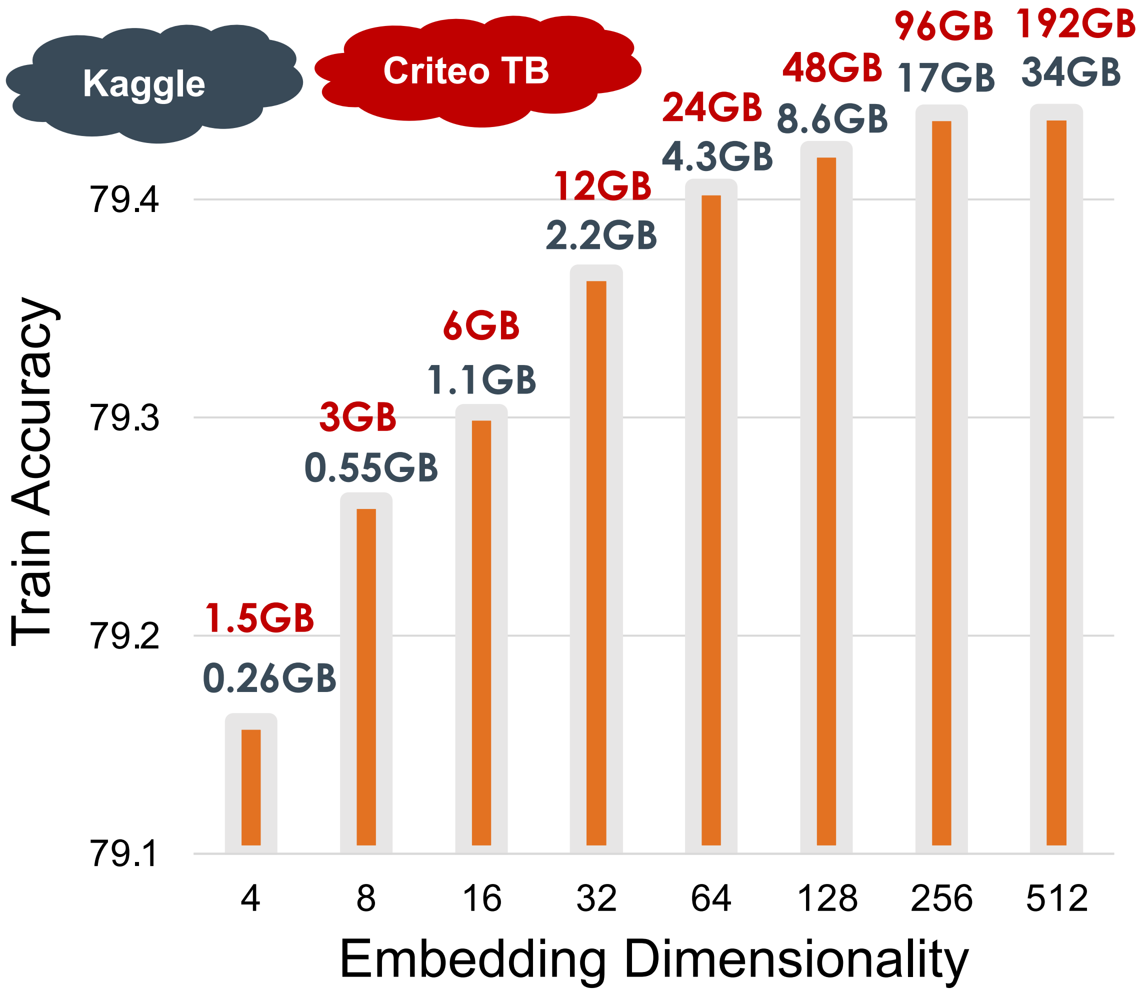
Recommender systems



Key common component: Sparse embedding feature

Recommender systems

More embedding features, more accuracy



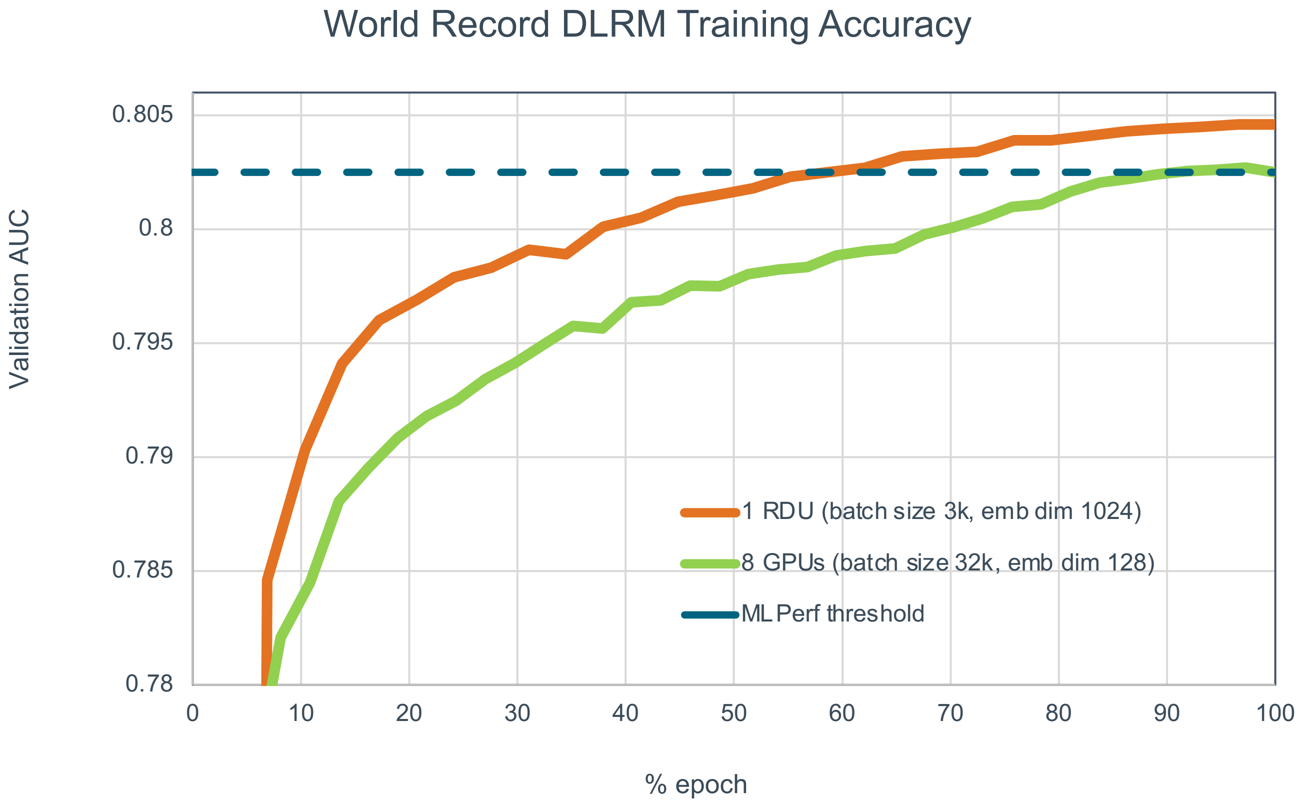
State-of-the-art accuracy on DLRM

80.46%

SOTA Accuracy

33%

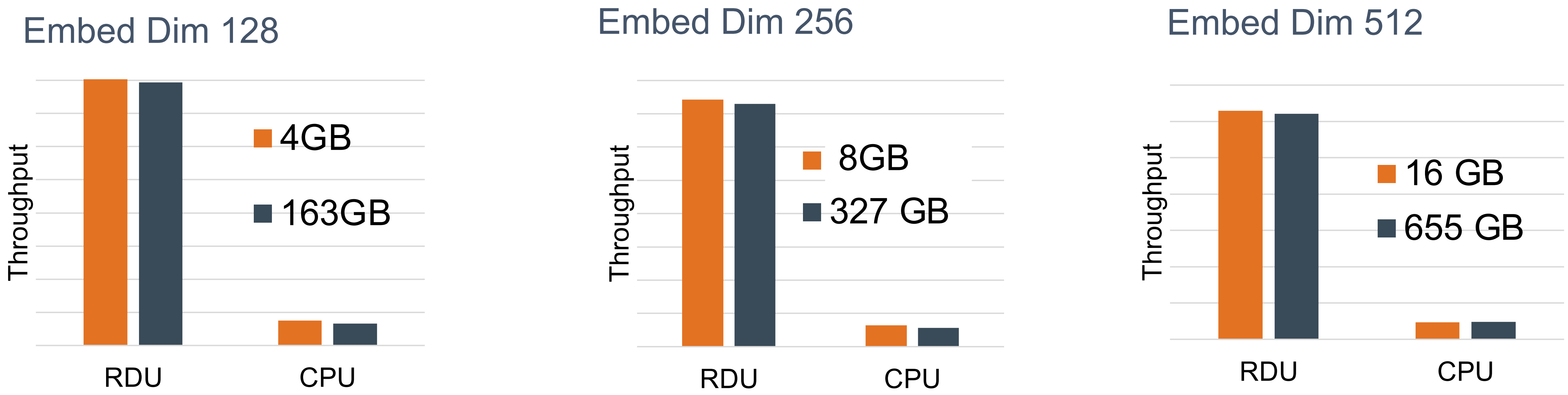
Faster Step-to-accuracy



Bigger isn't always better…but it is sometimes.

Training Performance

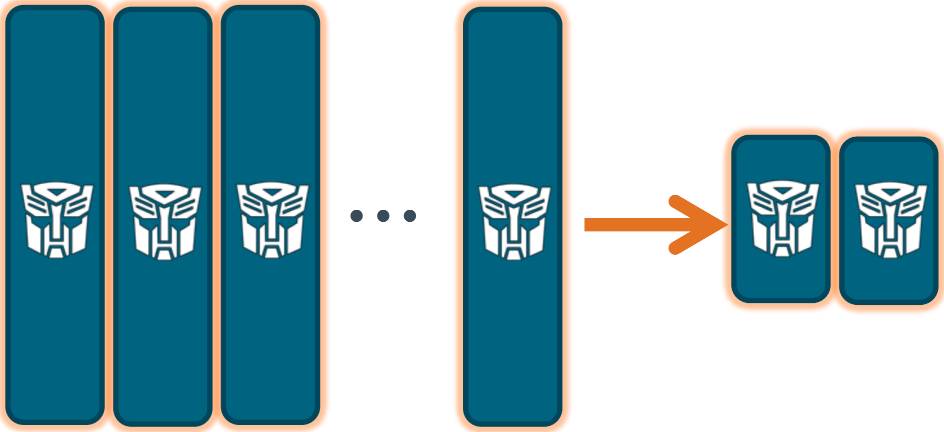
r5d.metal (CPU, FP32)



SambaNova scales to training massive recommender models

Natural Language Processing

Breakthrough efficiency in NLP model online deployment



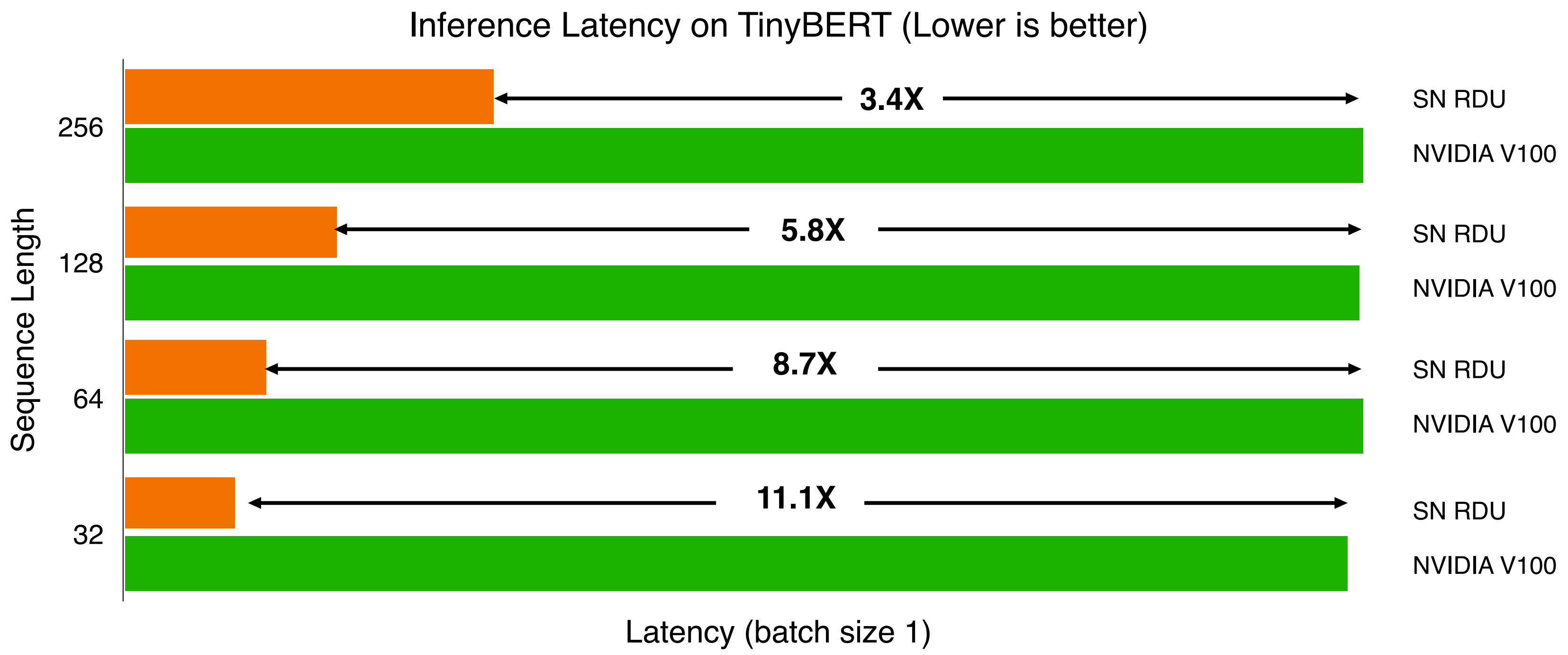
Distilled tiny Bert model

+

Siri, what is the whether in SF?

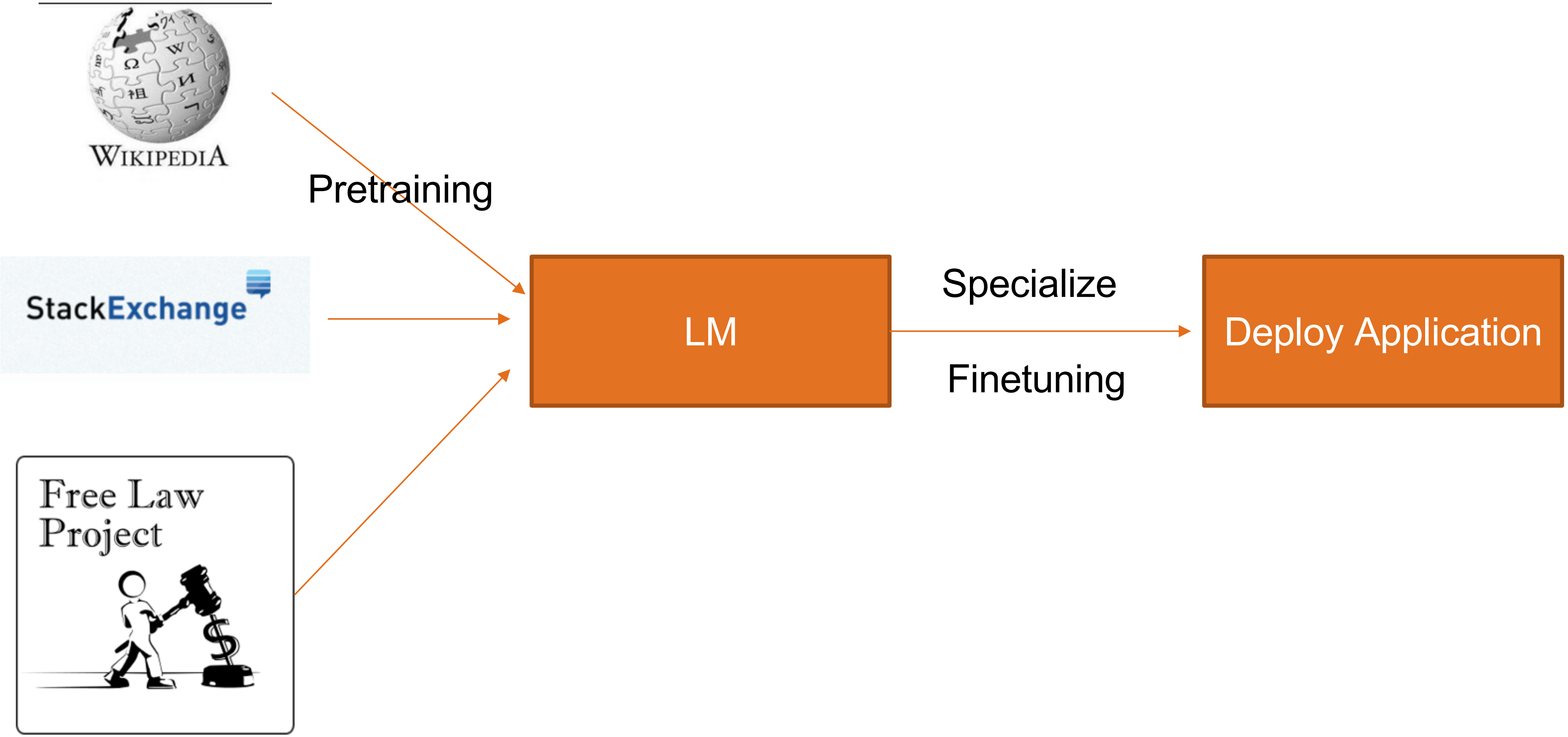
Short sequence input

Breakthrough Efficiency in NLP Model Online Deployment

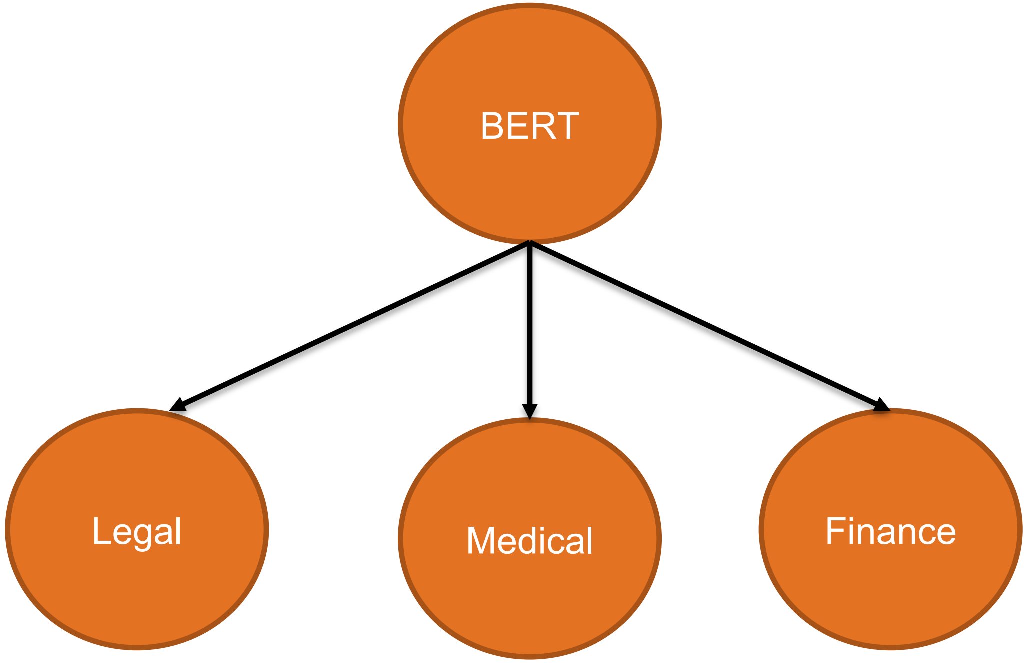


Enable up to 11X speedup for online training and inference

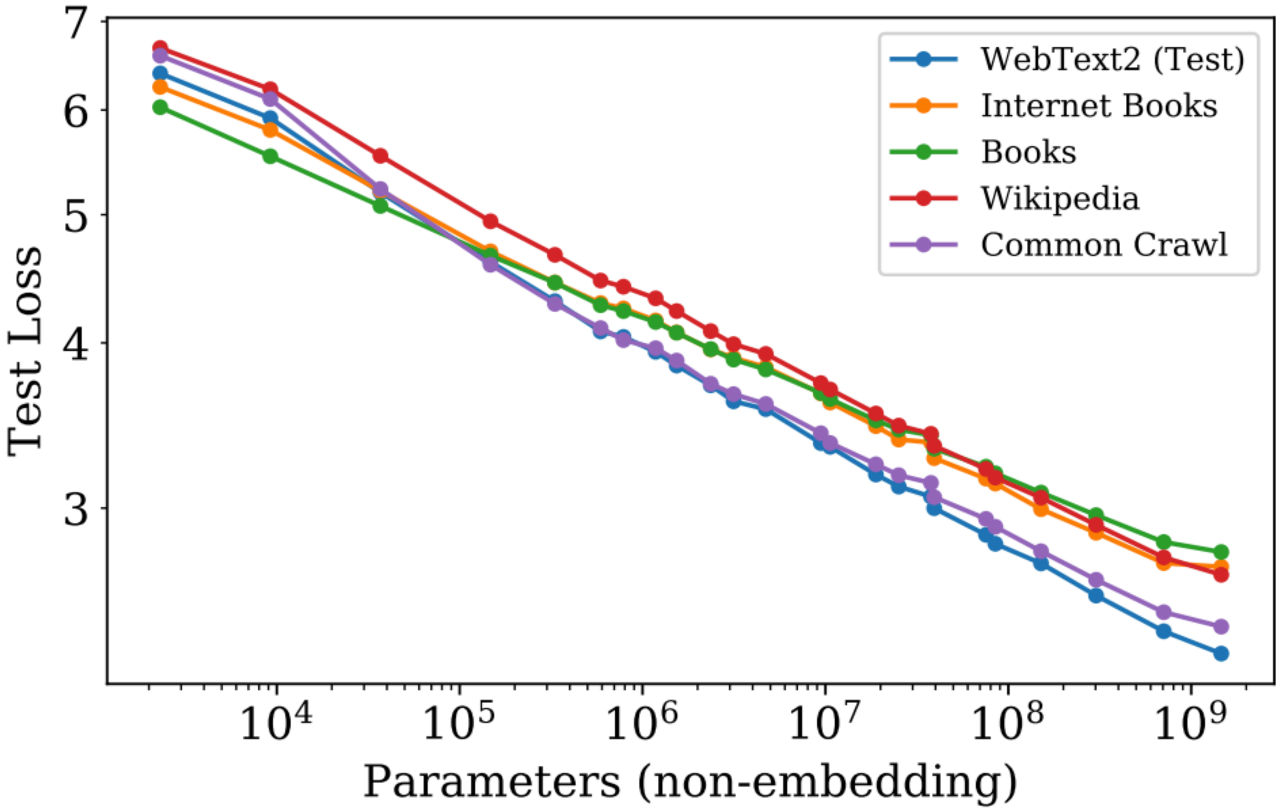
Pretraining and Finetuning



Domain Adaptation

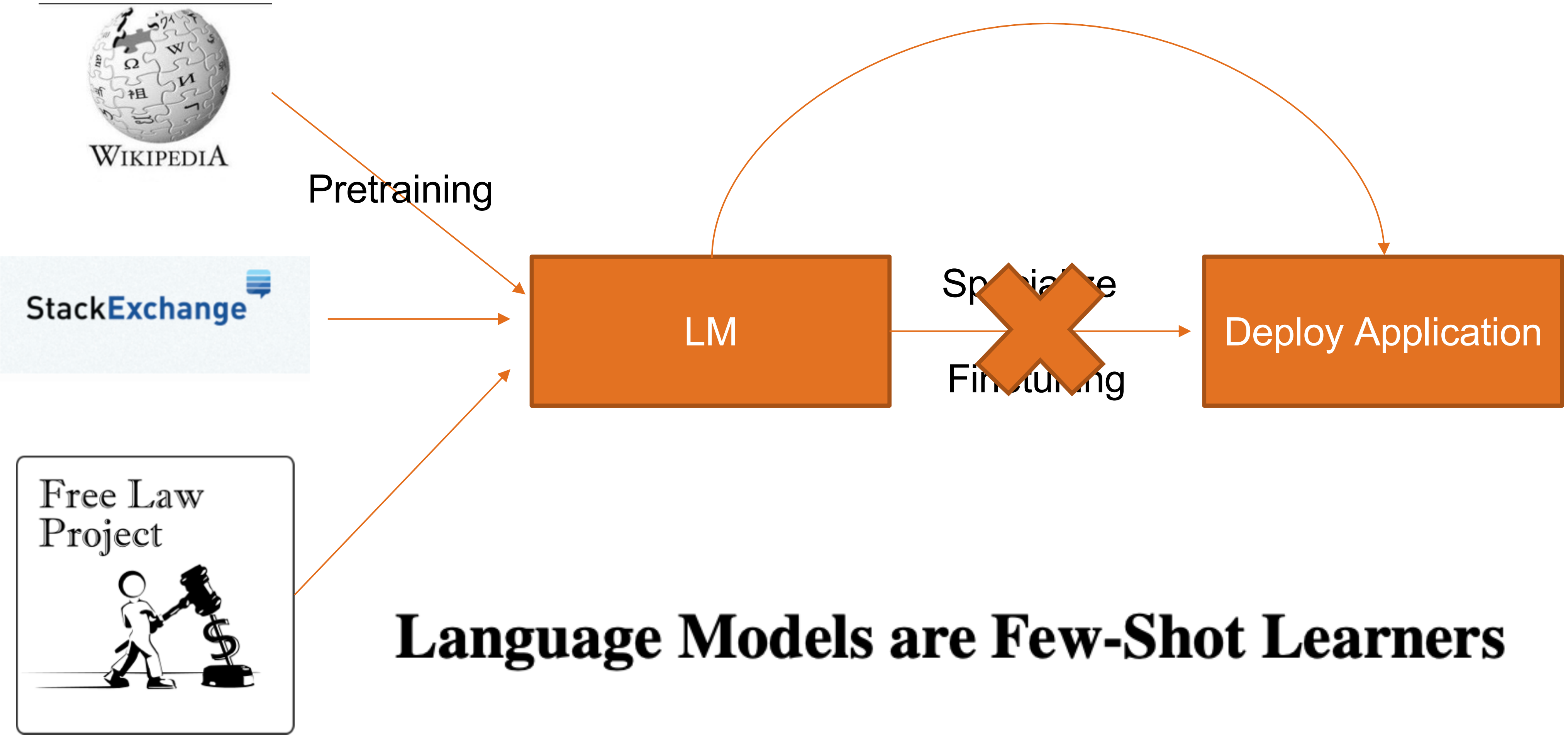


Scaling Laws for Neural Language Models



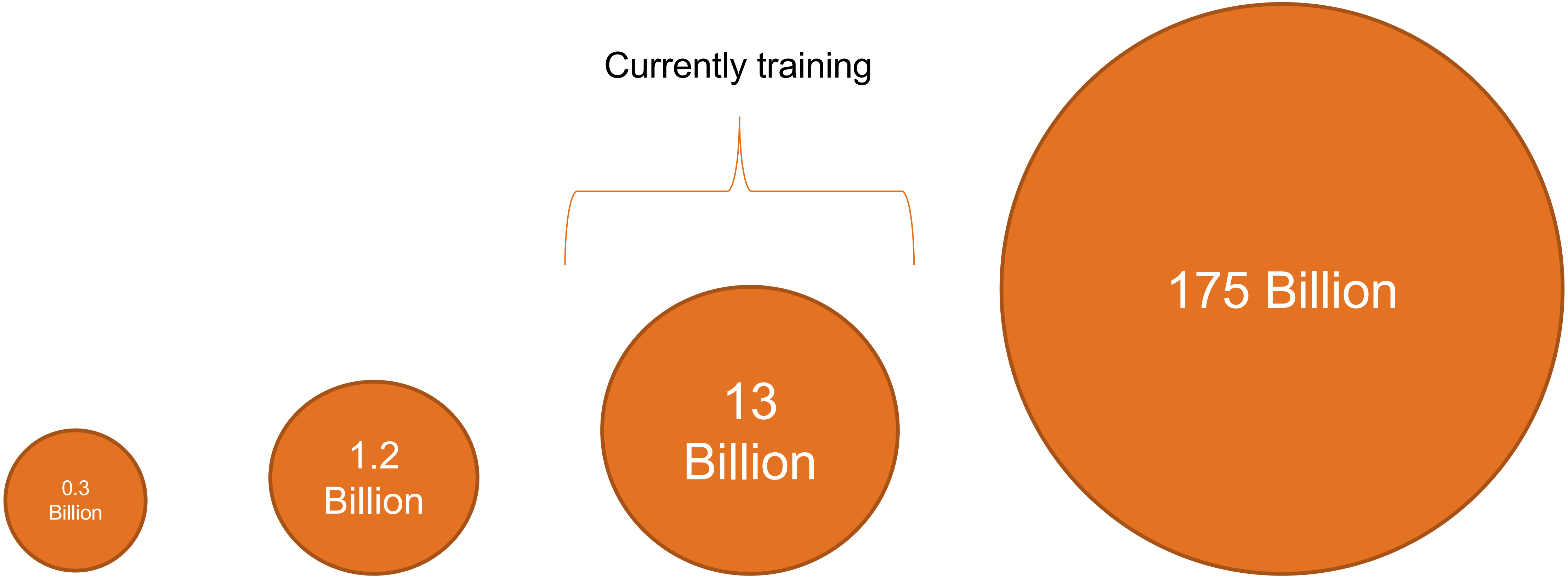
Application accuracy improves as the size of the language model increases

Pretraining and Finetuning



Language Models are Few-Shot Learners

GPT Family



Will start training next month

Currently training

0.3 Billion

1.2 Billion

13 Billion

175 Billion

BERT

GPT2-XL

GPT3-13b

GPT3

sambanova.ai

sambanova-systems

@SambaNovaAI

SambaNovaAI

SambaNova®
SYSTEMS