







Introduction

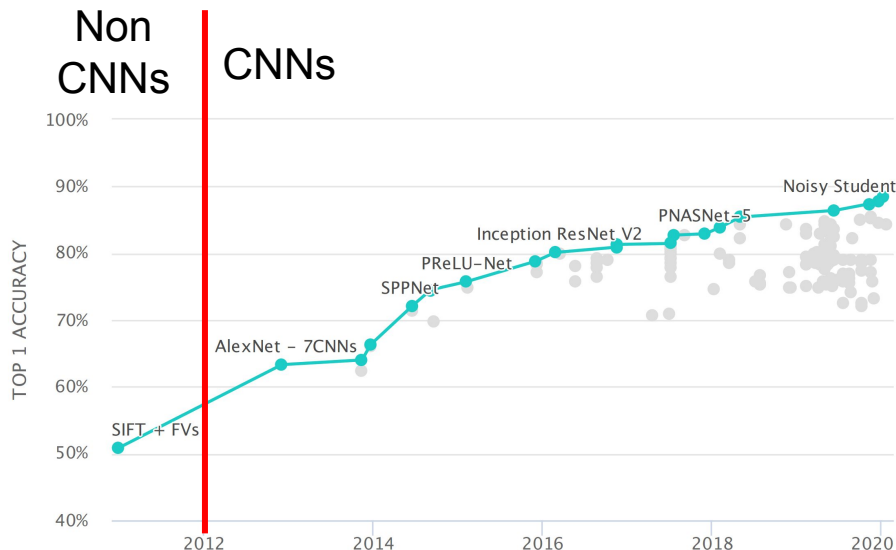
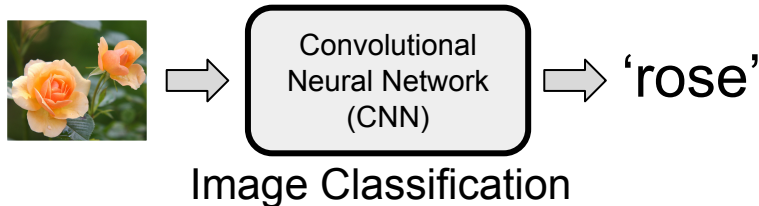
ECE-GY 9483/CSCI-GA 3033

Special Topics in Electrical Engineering EFFICIENT AI AND
HARDWARE ACCELERATOR DESIGN

Life is Powered by Deep Learning

- Deep Neural Networks (DNNs) have achieved state-of-the-art performance across a variety of domains

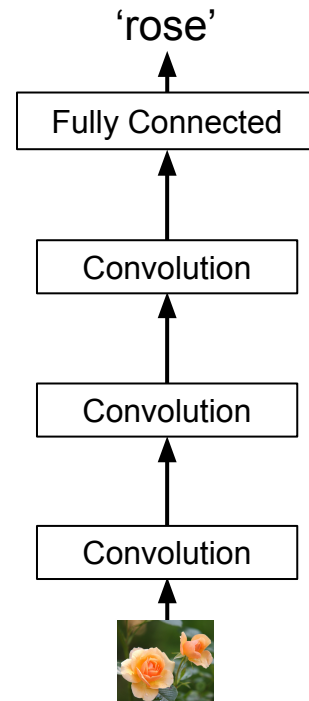
- Image Recognition 
- Video Processing 
- Natural Language Processing 
- Autonomous Driving 



- More desirable modern services are enabled by DNN

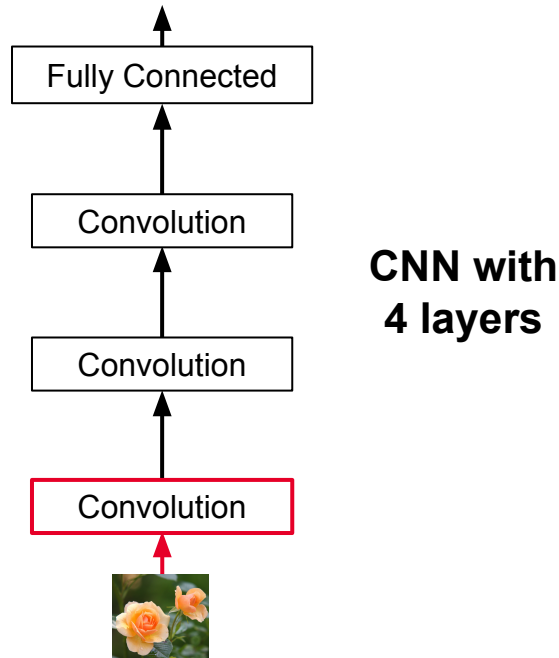
How Deep Neural Network is Executed?

- Use a Convolutional Neural Network (CNN) as an example
- This CNN contains four layers
 - 3 convolutional layers
 - 1 fully connected layer

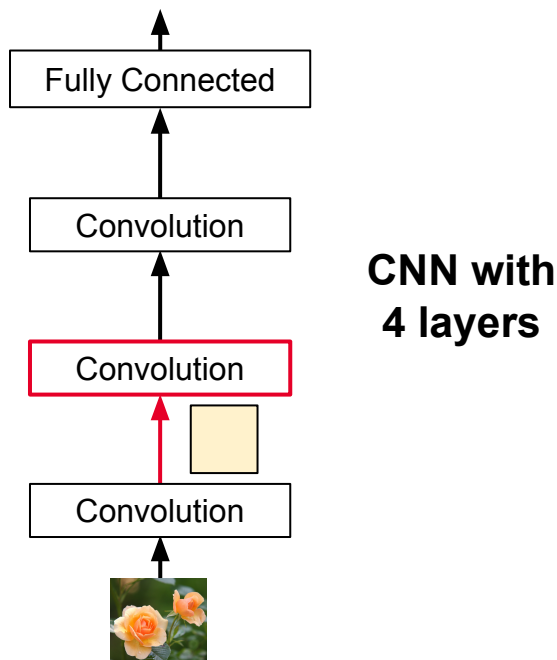


**CNN with
4 layers**

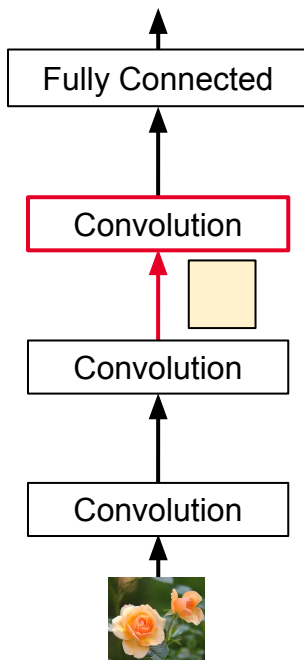
How Deep Neural Network is Executed?



How Deep Neural Network is Executed?

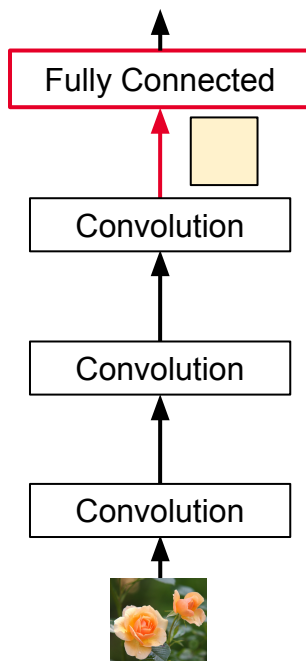


How Deep Neural Network is Executed?



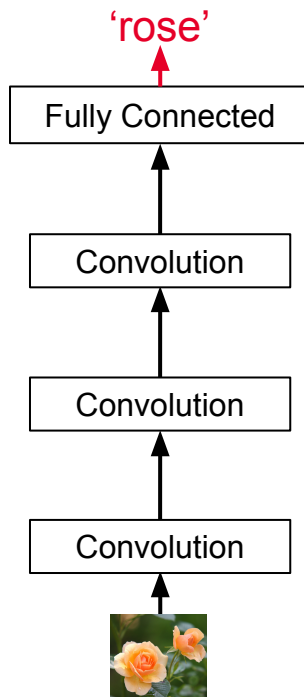
**CNN with
4 layers**

How Deep Neural Network is Executed?



**CNN with
4 layers**

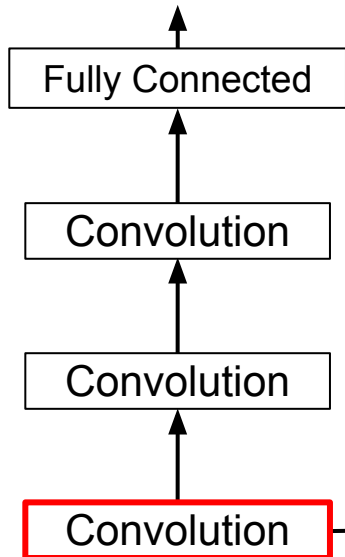
How Deep Neural Network is Executed?



**CNN with
4 layers**

DNN Execution: A Matrix View

Layer View



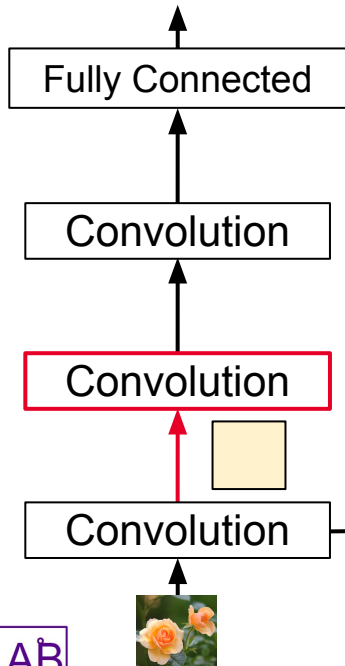
Matrix View



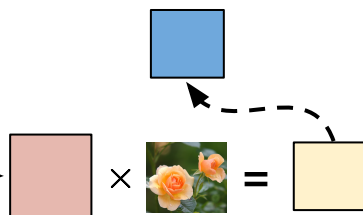
- Weight matrices are **learned** during training

DNN Execution: A Matrix View

Layer View



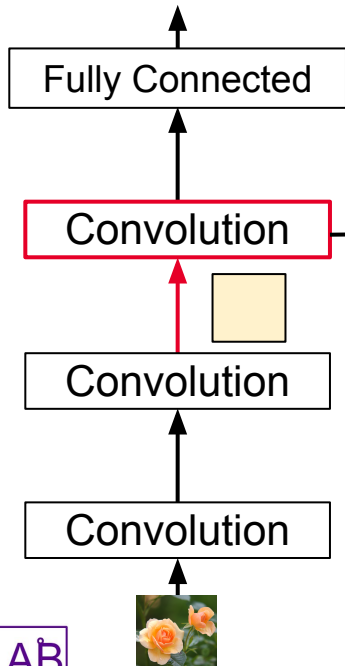
Matrix View



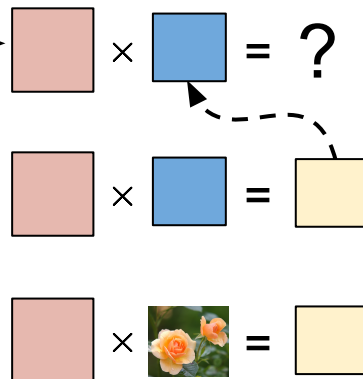
- Weight matrices are **learned** during training

DNN Execution: A Matrix View

Layer View

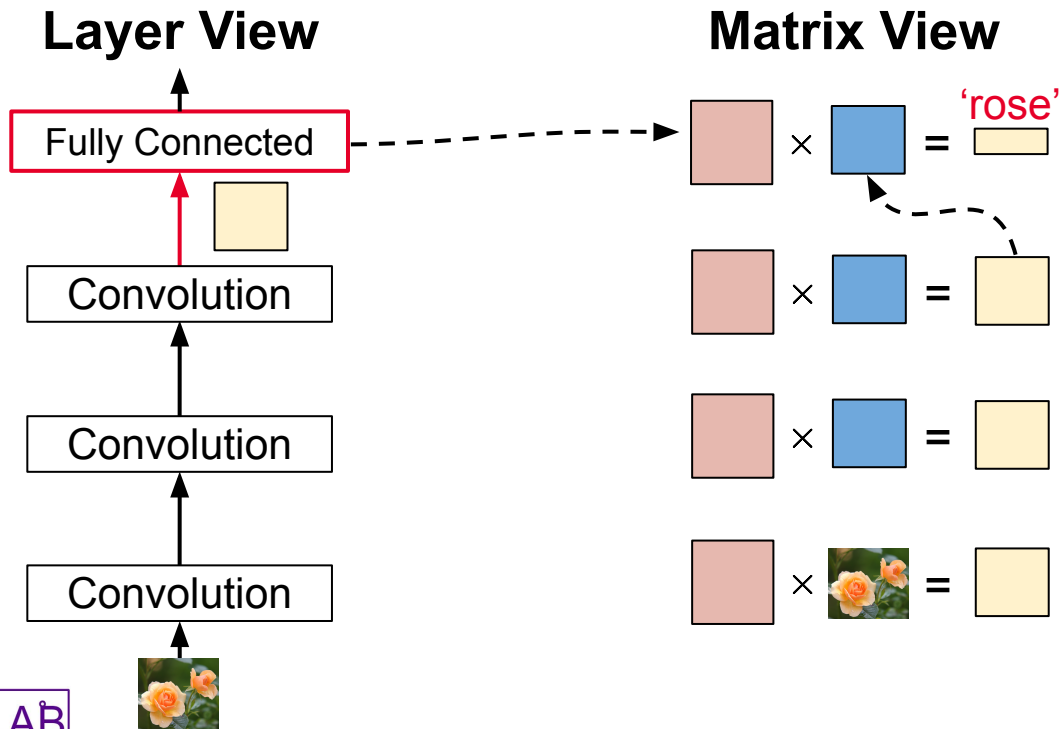


Matrix View



- Remaining layers follow this pattern.

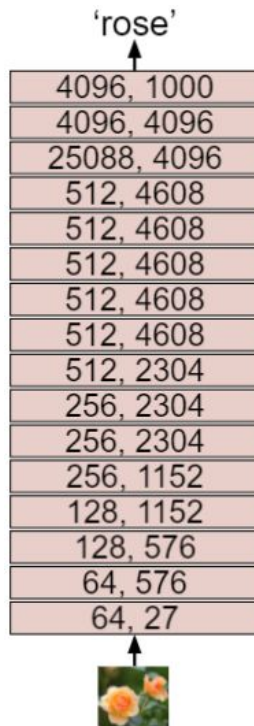
DNN Execution: A Matrix View



- Remaining layers follow this pattern

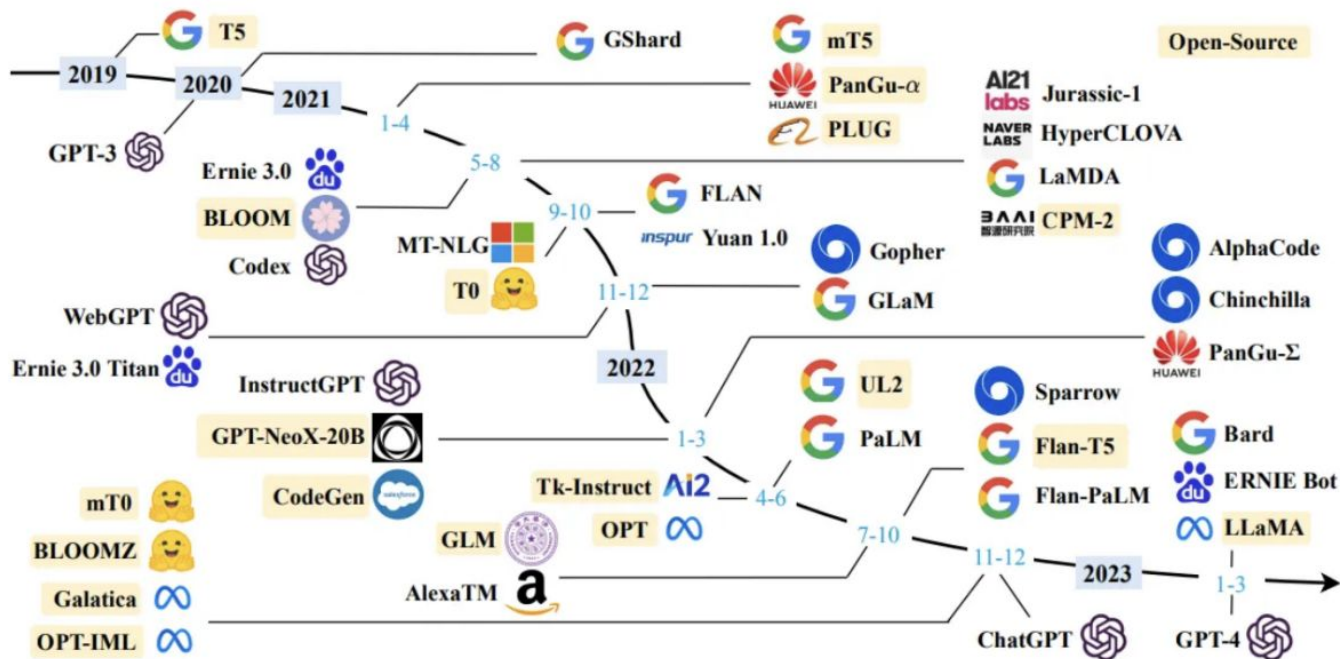
Deployment of DNN: Problems

- The majority of computation workloads for DNN inference involves a series of **matrix multiplications**.

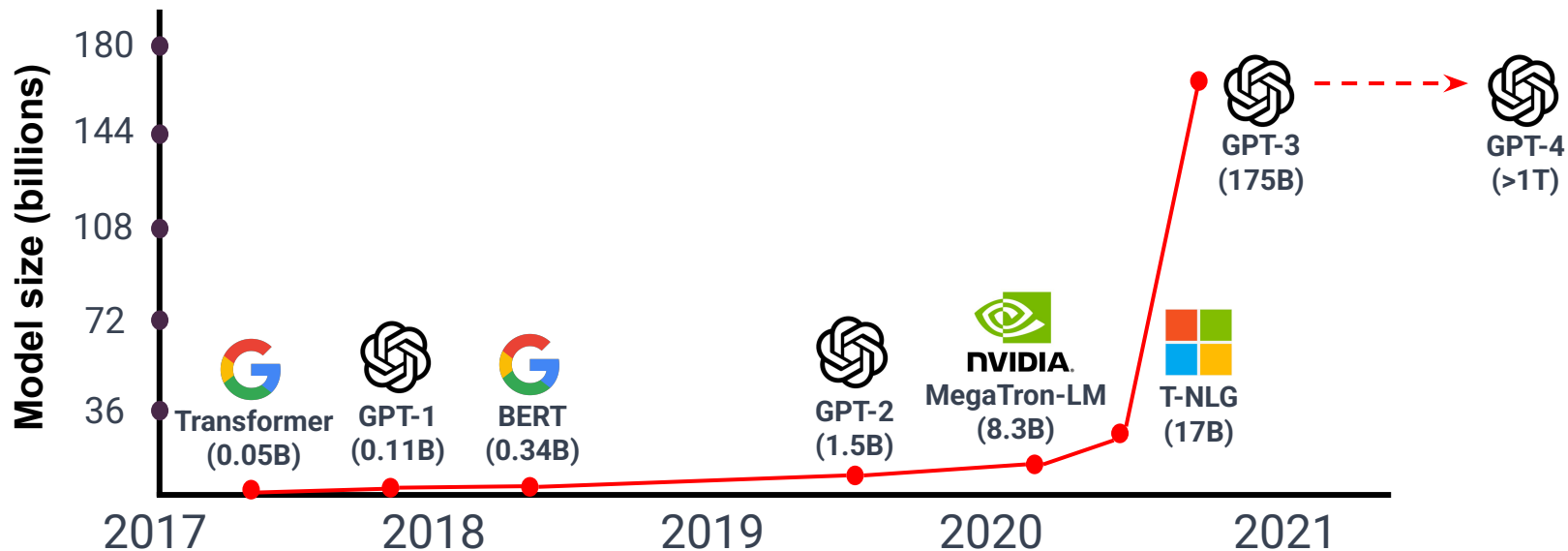


VGG-16 is a CNN with over 150M weights across 16 matrices

The Era of Large Models (LMs)



Cost of Large Models



- $1.4e^{12}$ FLOPs to execute GPT-2.

The Cost of Large Models

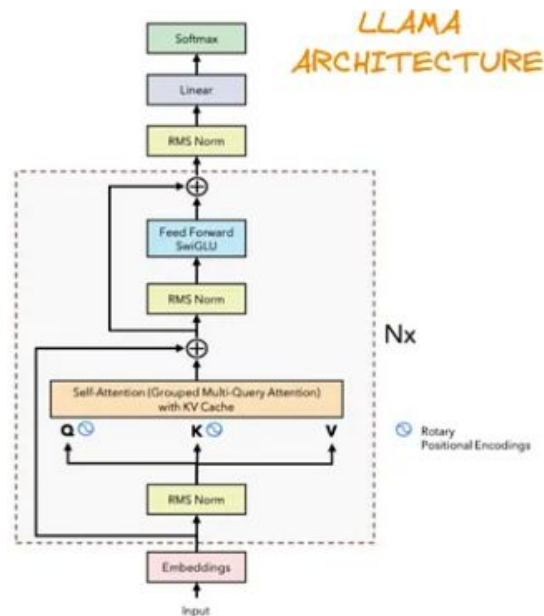


- Training GPT-4 required **25,000 A100 GPUs** over several weeks.
- **Cost:** Renting a single high-end GPU on cloud services like AWS can cost **\$3–\$5 per hour**. Training GPT-4 is estimated to cost **\$63-100 million** on cloud computing resources.

Efficient AI: An Emerging Area

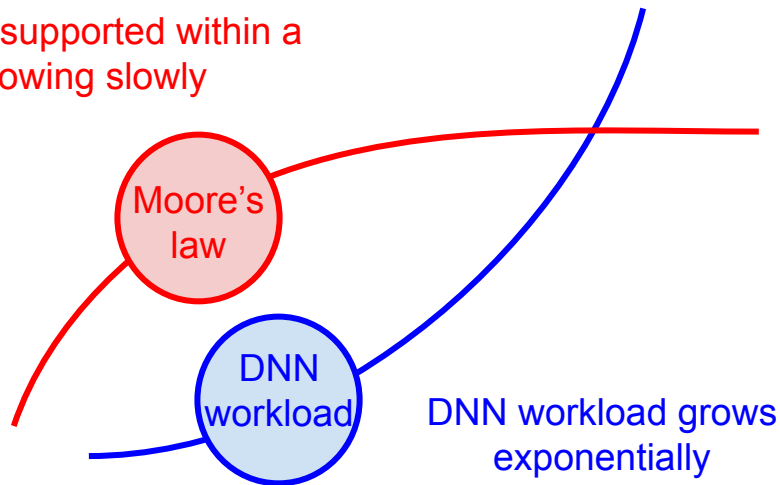
Model Size	FP16	FP8	INT4
8B	16 GB	8 GB	4 GB
70B	140 GB	70 GB	35 GB
405B LLaMA 3.1	810 GB	405 GB	203 GB

Design more aggressive and efficient AI model is of paramount importance



Efficient AI: An Emerging Area

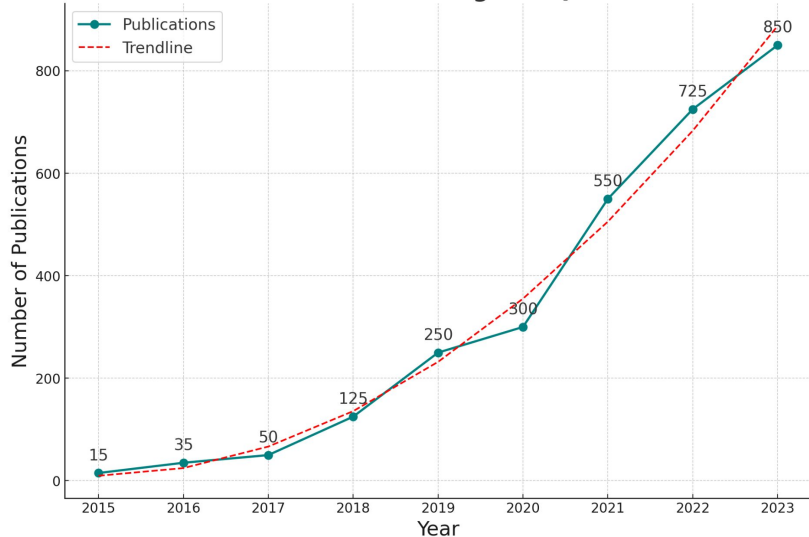
The amount of compute supported within a hardware unit is growing slowly



How to reduce the compute while maintaining a good DNN accuracy?

Efficient AI: An Emerging Area

Research Publications on DNN Pruning and Quantization (2015-2023)



- Efficient AI has become one of the most popular areas in AI community.
- The recent emergence of large models has further heightened the need for efficient AI.

Efficient AI: An Emerging Area

Infra Hardware TF

Sunnyvale, CA + 2 more

M Visiting Researcher - AI Accelerators
Meta
Harrisburg, PA • via Monster
3 days ago Full-time

Hardware System

Menlo Park, CA AI

IBM Research Scientist-AI Accelerator Design
IBM
Yorktown Heights, NY • via Karkidi
\$ 120K-190K a year Full-time Health insurance Dental insurance Paid time off

Silicon Hardware

Sunnyvale, CA + 2 more

B Machine Learning Engineer - Efficient Machine Learning
Bose Corporation, U.S.A
Anywhere • via Workday
4 days ago Work from home Full-time

Visiting Research

Menlo Park, CA AI

A Machine Learning/AI Engineer
Advanced Micro Devices, Inc
Boxborough, MA • via AMD Careers
Full-time

Director, AI Resea

London, UK AI Res

A Sr Machine Learning Engineer, AI Software Solutions
Advanced Micro Devices, Inc
Fishkill, NY • via Monster
Full-time

Jc

TATA CONSULTANCY SERVICES tcs Artificial Intelligence Engineer
Tata Consultancy Services
Malvern, PA • via LinkedIn
21 hours ago \$ 130K-160K a year Full-time

Accelerator Design

Karkidi
time Health insurance Paid time off Dental insurance

Engineering/ AI accelerator compiler and Runtime

IS
gree.

anager, AI Compiler

ome Full-time Health insurance

ation Engineer for Intel AI Accelerator

nce Paid time off
sient practical experience.

ems ML - Frameworks / Compilers / Kernels

reers Jobs

ems ML - Frameworks / Compilers / Kernels

iter

AI Tech Startups/Unicorns

nomtek


Fast Multi On-D AI

ARTIFICIAL INTELLIGENCE

Emergent AI

Deploy custom AI locally for you without internet

Explore Octopus



Cerebras and D

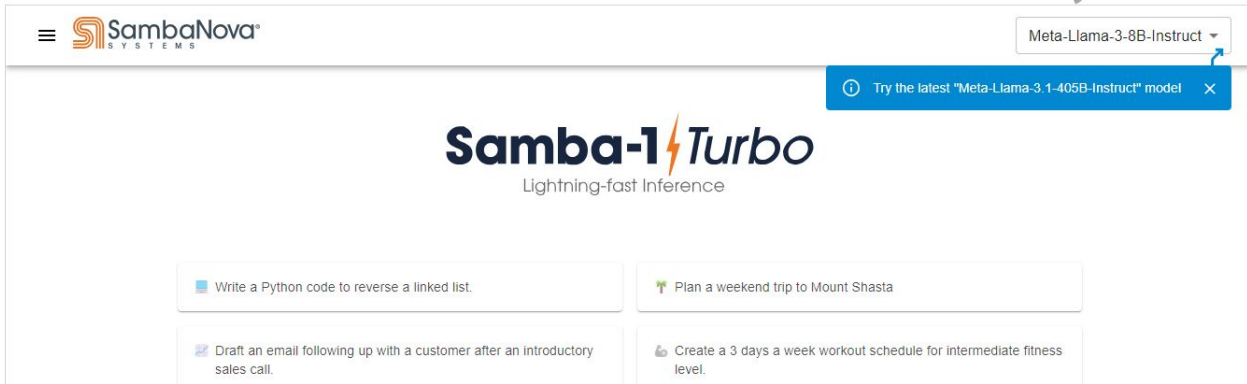
Rev


Cerebra

Unleash the Fastest Private Enterprise GPT

World record performance with Llama 3.1 405b on the only generative AI platform with full accuracy.

Try it now!



 SambaNova[®]
SYSTEMS

Meta-Llama-3-8B-Instruct

Try the latest "Meta-Llama-3.1-405B-Instruct" model

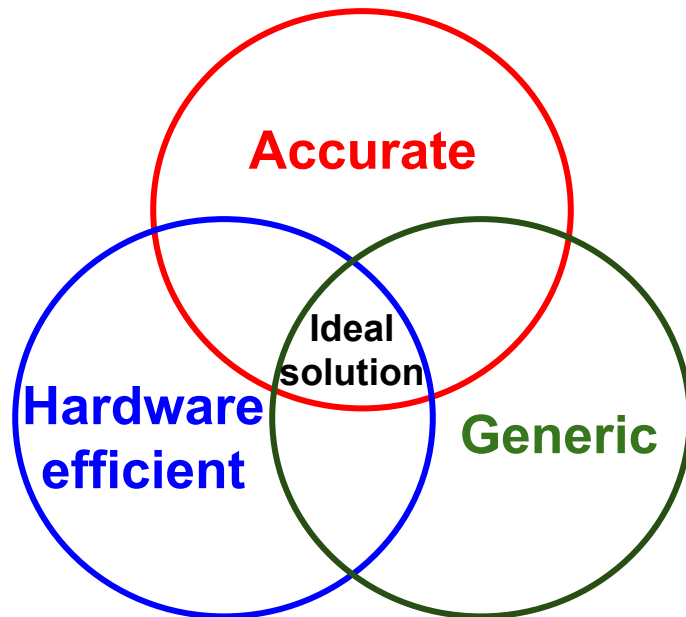
Samba-1 Turbo

Lightning-fast Inference

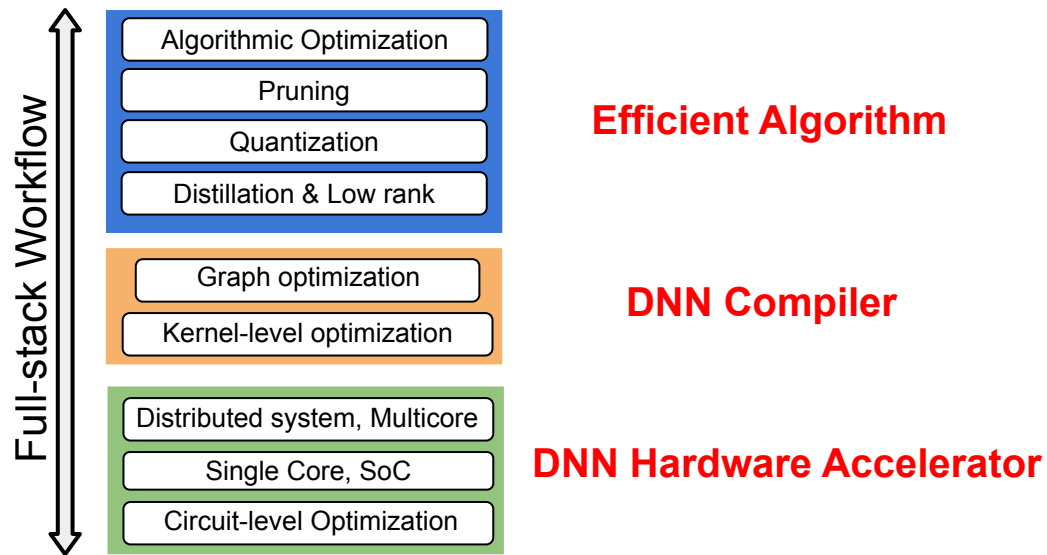
- Write a Python code to reverse a linked list.
- Plan a weekend trip to Mount Shasta
- Draft an email following up with a customer after an introductory sales call.
- Create a 3 days a week workout schedule for intermediate fitness level.

Efficient AI: An Emerging Area

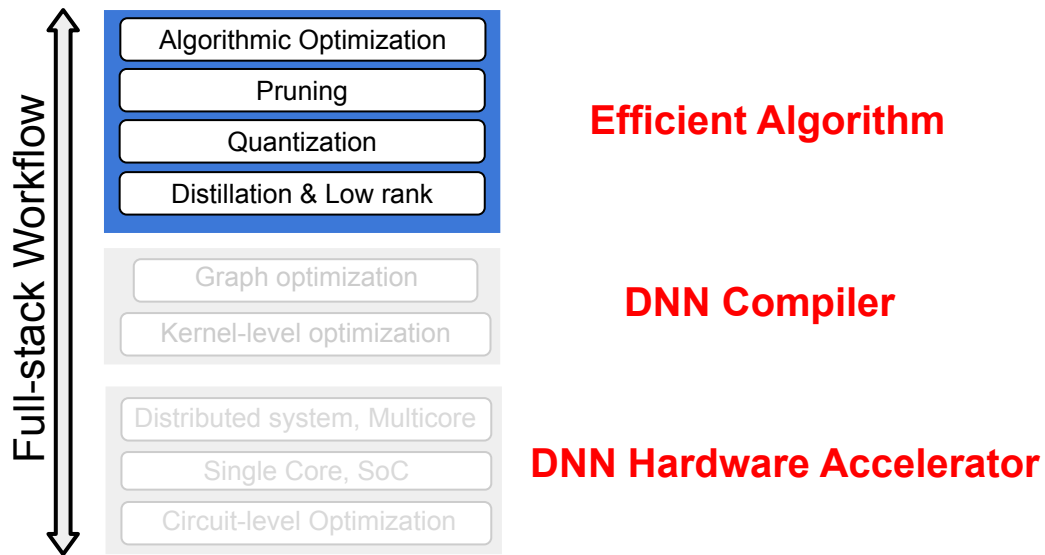
Challenges



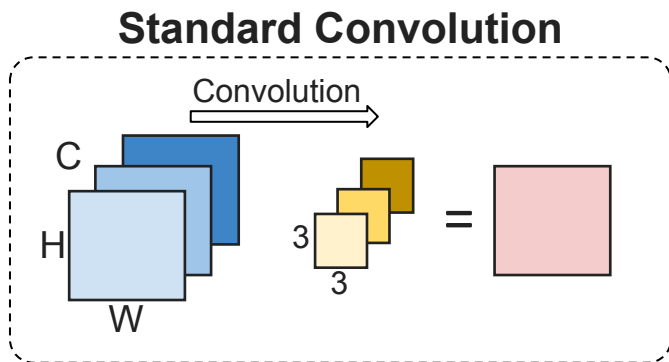
Efficient AI: Full-stack Workflow



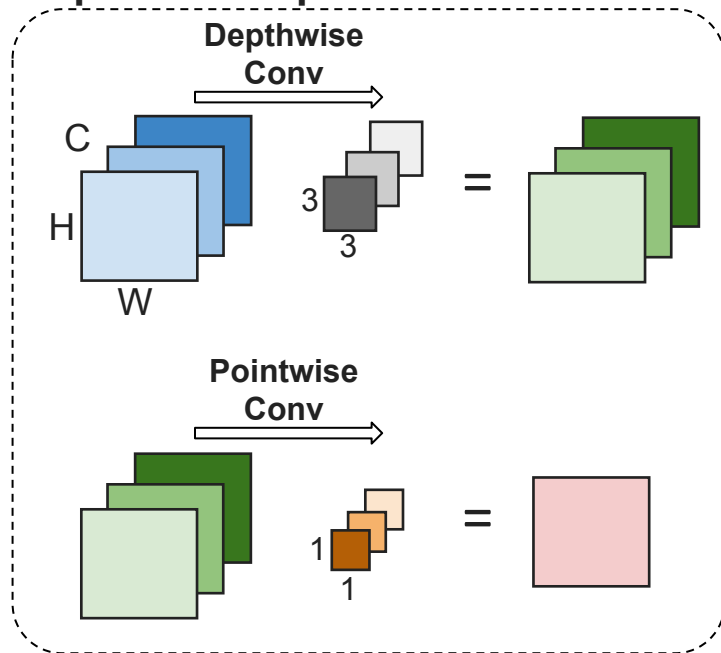
Efficient AI: Full-stack Workflow



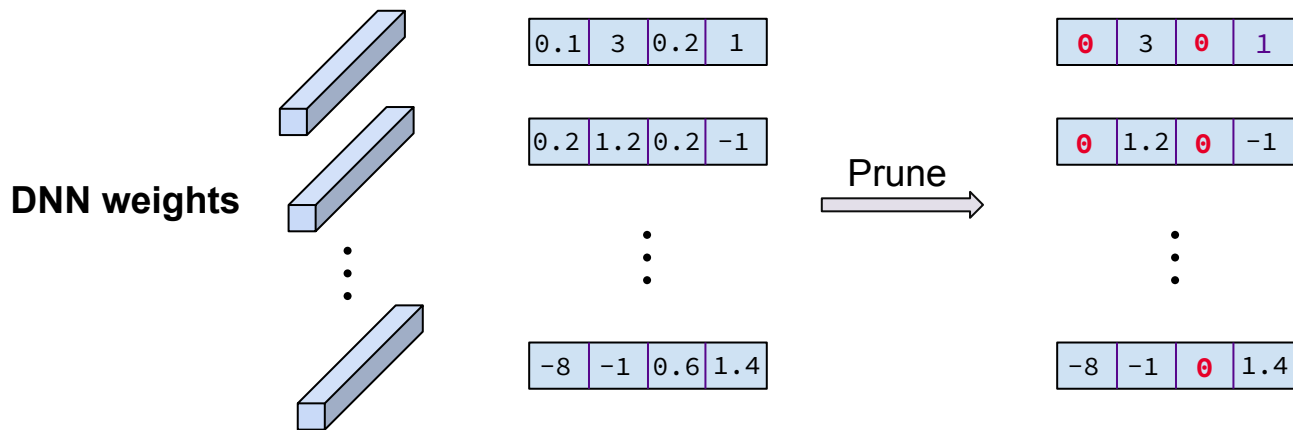
Algorithmic Optimization



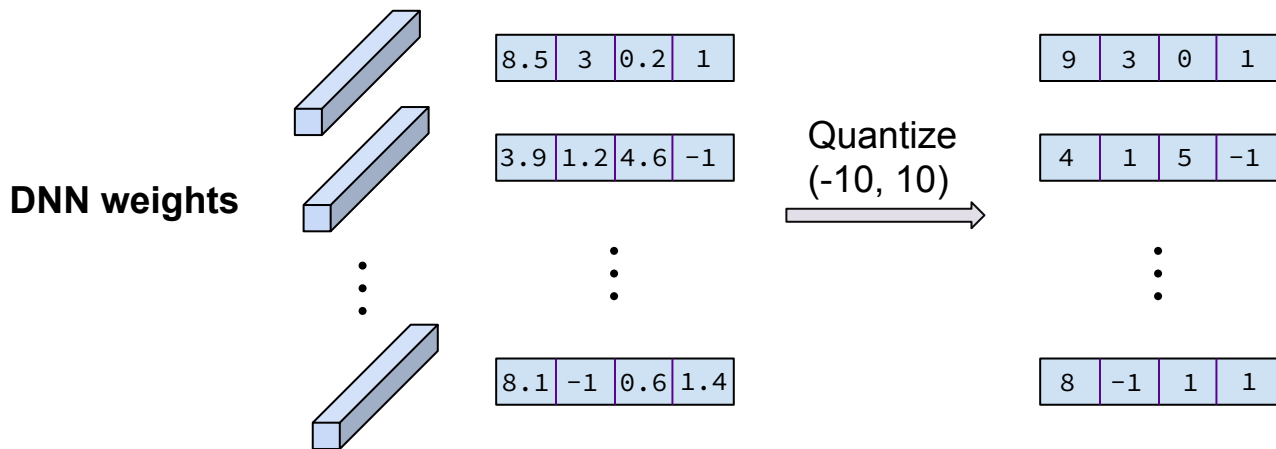
Depthwise Separable Convolution



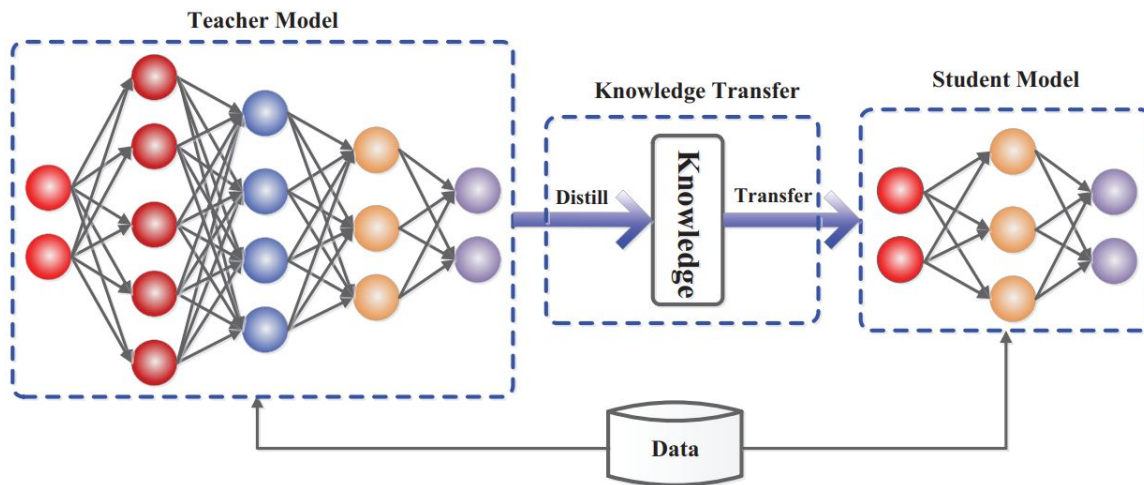
Efficient DNN Algorithm: Pruning



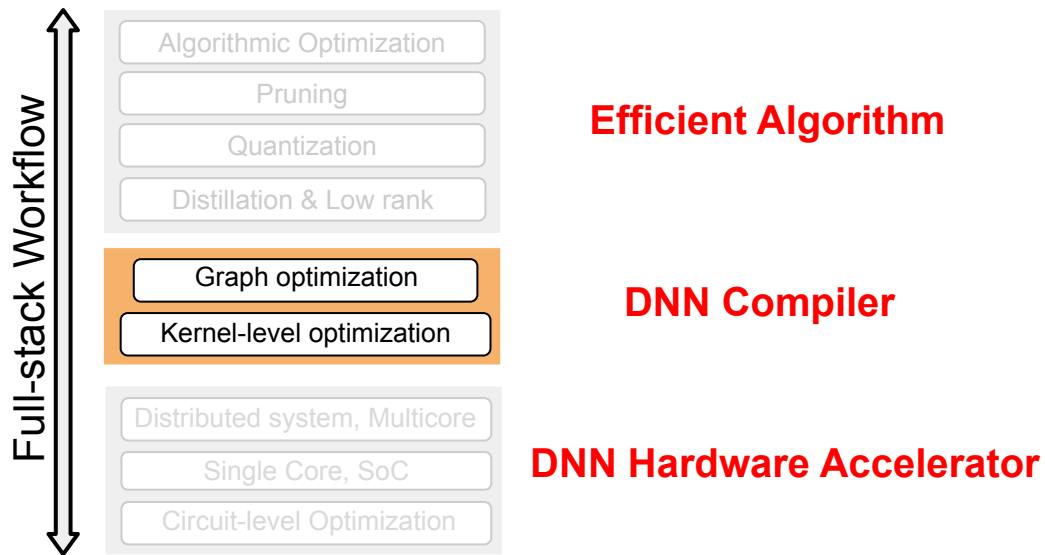
Efficient DNN Algorithm: Quantization



Knowledge Distillation

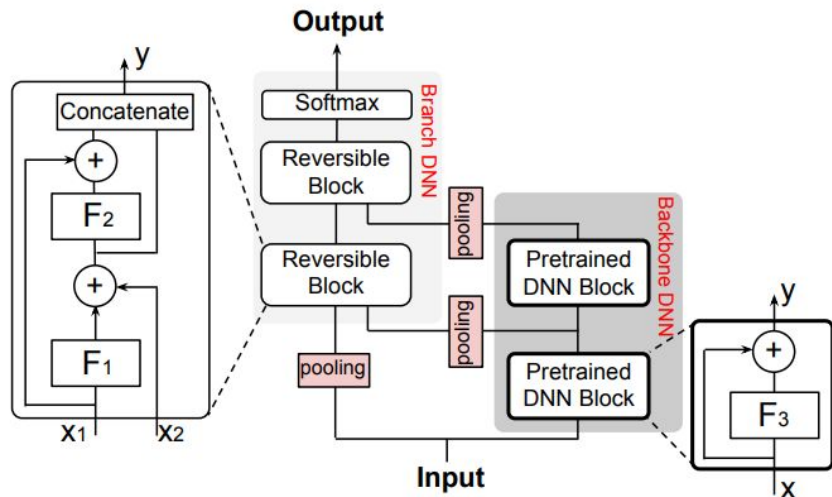


Efficient AI: Full-stack Workflow

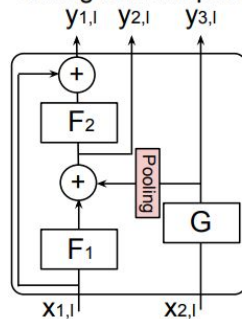


Graph Level Optimization

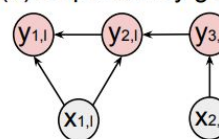
CAMEL Training



(a) Computation during forward pass



(b) Dependency graph

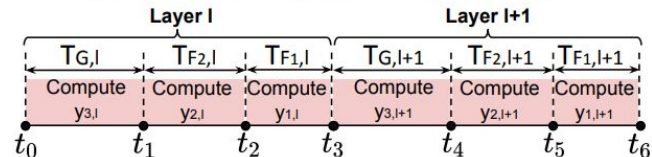


(c) Pseudo instruction

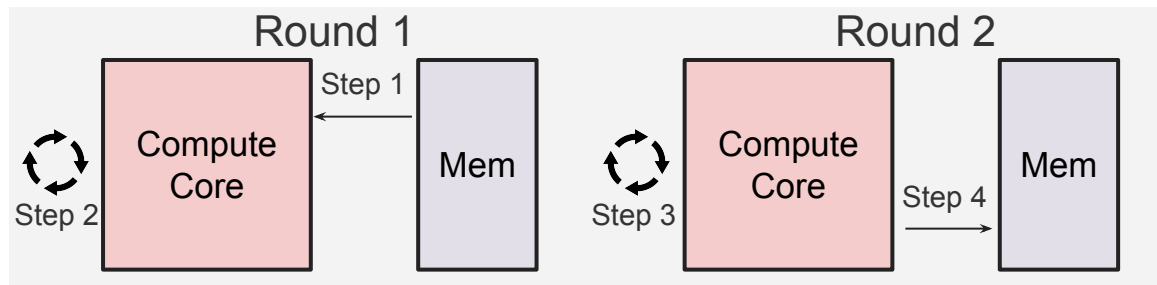
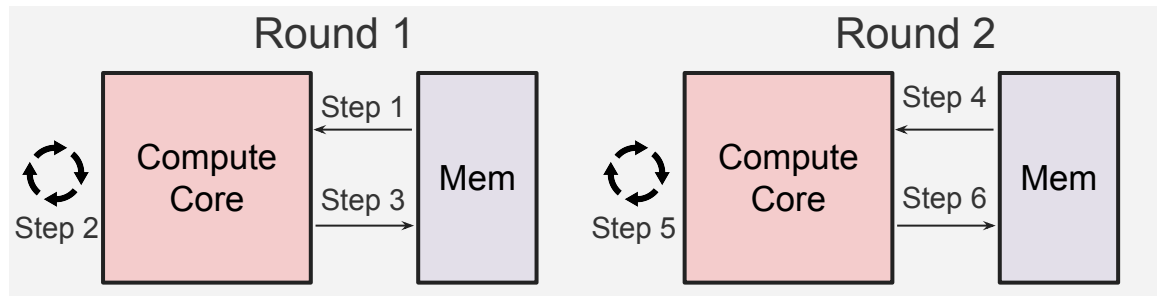
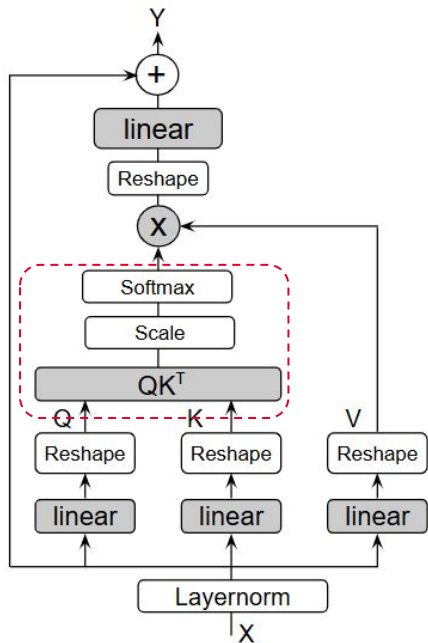
```

Compute  $y_{3,i} = G(x_{2,i})$ 
Overwrite  $x_{2,i}$  with  $y_{3,i}$ 
Compute  $y_{2,i} = \text{Pool}(y_{3,i}) + F_1(x_{1,i})$ 
Save  $y_{2,i}$ 
Compute  $y_{1,i} = x_{1,i} + F_2(y_{2,i})$ 
Overwrite  $x_{1,i}$  with  $y_{1,i}$ 
    
```

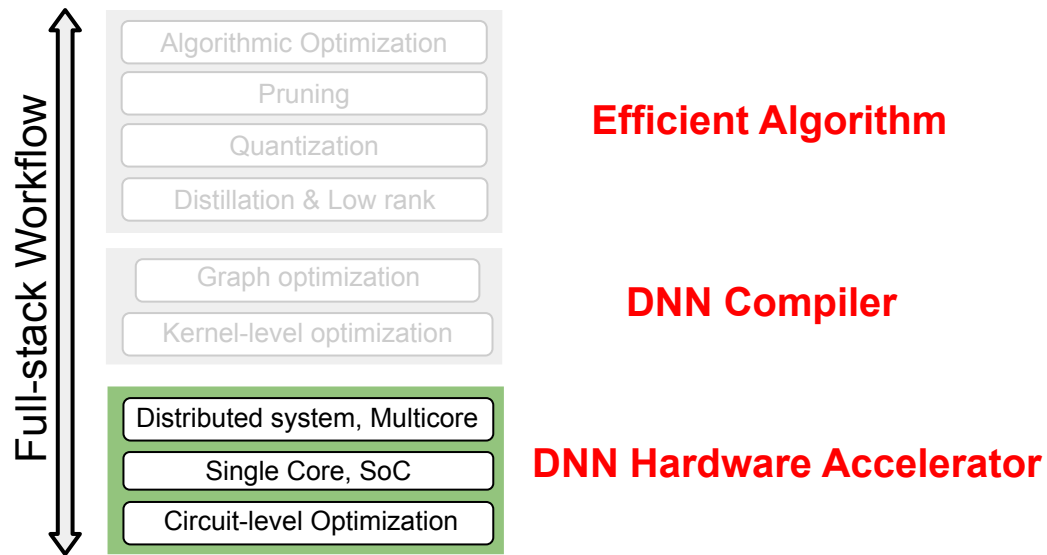
(d) Computation pattern of forward pass



Kernel Level Optimization



Efficient AI: Full-stack Workflow



Hardware Support for DNN

- GPU is better than CPU in terms of throughput for both Neural Network training and inference.
 - GPU leverages the highly parallelized architecture of its computing units to handle computational intensive operations.
- However, GPU:
 - General purpose, although much more specific than CPU.
 - Still not fast and power-efficient enough.
 - Does not support advanced efficient DNN algorithm.



NVIDIA

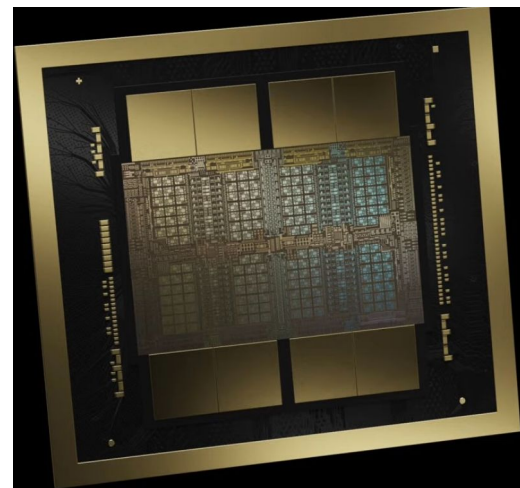
Chip size	814 mm ²
On-chip memory	~50MB
Total memory	~96GB HBM
Cores	16,896 FP32 + 528 Tensor
Precision	FP16/FP8/INT8
Memory bandwidth	0.003 Petabytes/sec



NVIDIA H100

NVIDIA

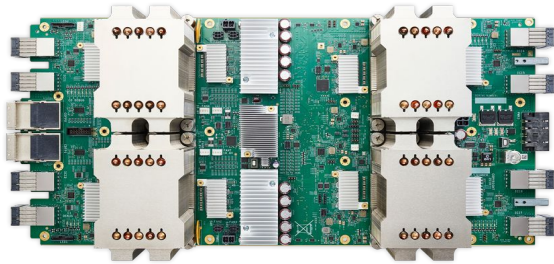
Chip size	-
On-chip memory	-
Total memory	192GB HBM
Cores	-
Precision	FP16/FP8/FP4/INT8
Memory bandwidth	8 Terabytes/sec



NVIDIA Blackwell

Hardware Support for DNN

- ASIC-based implementations have been recently explored to accelerate the DNN inference.
 - Google's TPU, Apple's Neural Engine, Cerebras AI chip, ...
- FPGA-based accelerators for DNN inference have been recently developed.
 - Has good programmability and flexibility
 - Short development cycles
 - Can be used as a benchmark before implementing on ASIC



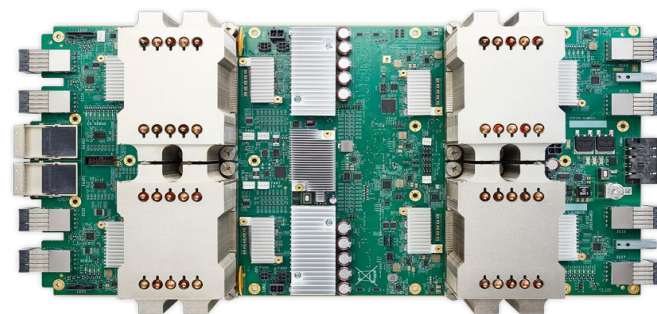
Tensor Processing Unit (Google)



Alveo Accelerator Card (Xilinx)

Google

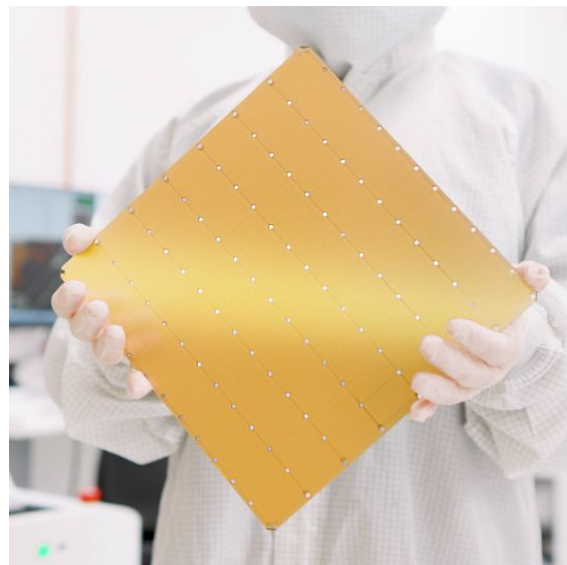
Chip size	790 mm ²
On-chip memory	112 MB
Total memory	32GB HBM
Precision	BF16/INT8
Memory bandwidth	1640 TB/sec



TPU v6 (Trillium)

Cerebras AI Chips

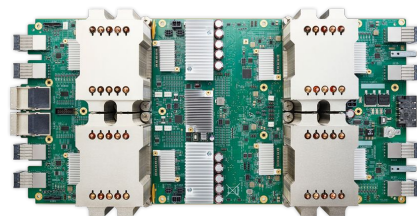
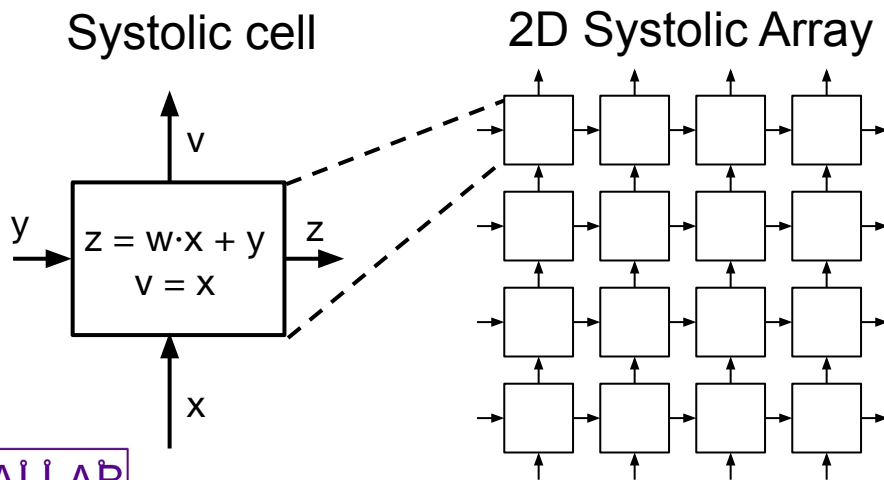
Chip size	46,225 mm ²
On-chip memory	44 GB
Cores	900,000
Memory bandwidth	21 Petabytes/sec



Cerebras CS-3

Systolic Array

- Kung and Leiserson, "Systolic Arrays for VLSI," 1978 and Kung, "Why systolic architectures?" 1982
- 2D grid of multiplier-accumulators (MACs) for matrix multiplication
- Used by Google TPU for deep learning (2017), etc



TPU (Google)

Bit-serial Low-precision Multiplier

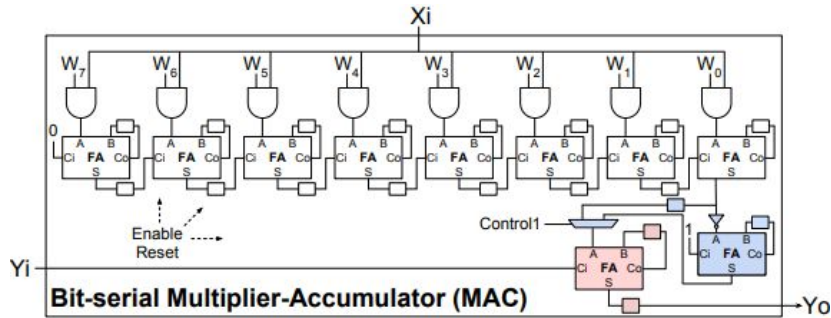
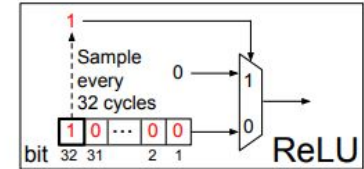
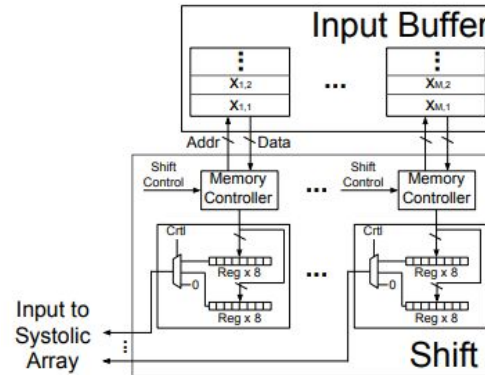
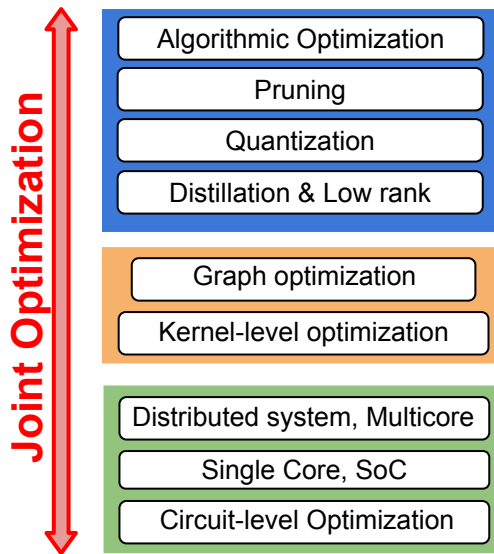


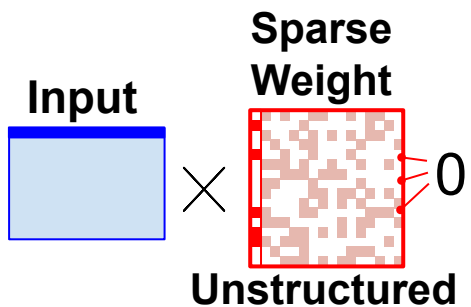
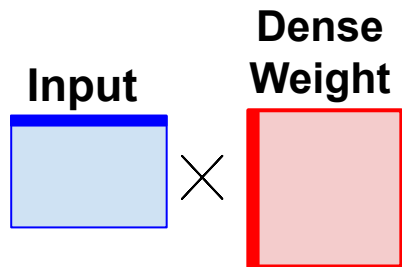
Figure 7: Bit-serial multiplier-accumulator (MAC).



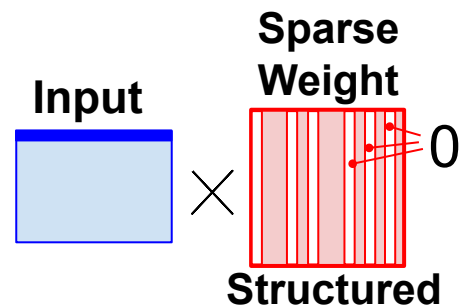
Why We Need Codesign?



Why We Need Codesign?



High accuracy
low hardware efficiency

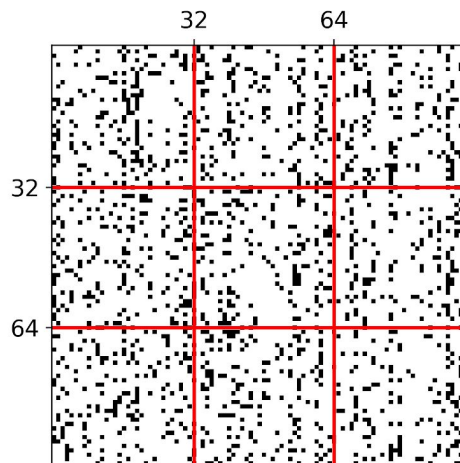


Low accuracy
high hardware efficiency

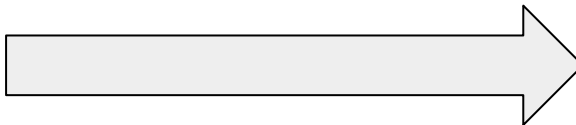
Hardware architecture needs to be considered when designing efficient DNN.

Column Combining

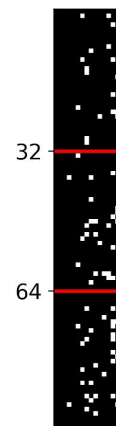
Sparse
Weight Matrix



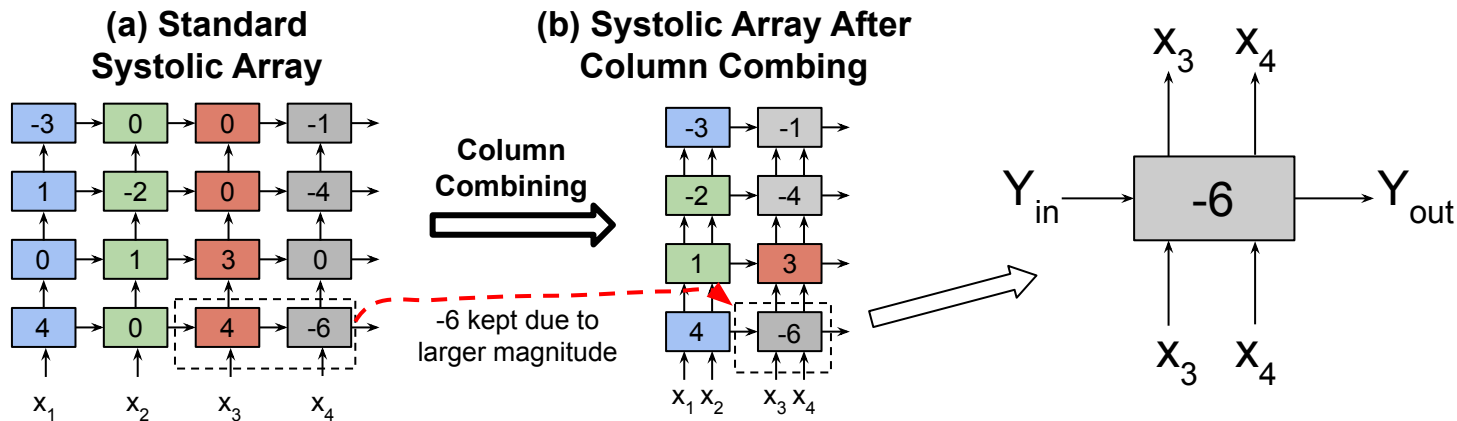
Column Combining
8x reduction in size



Packed Format in
Systolic Array

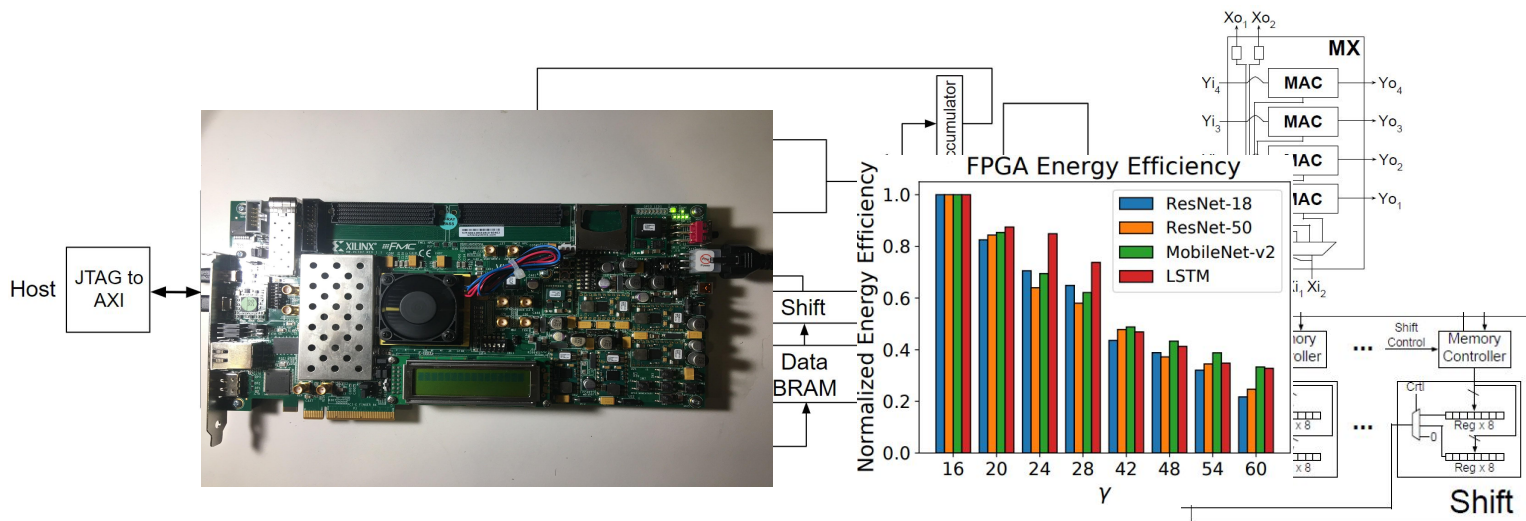


Column Combining



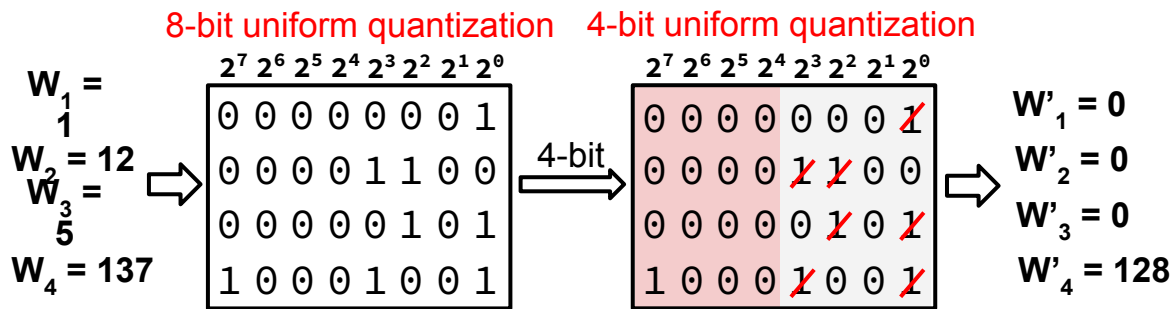
- Column combining can greatly increase the utilization efficiency of the systolic array
- Recently, Nvidia A100 GPU adopts a similar idea to support the balanced structured sparsity on their GPU

FPGA Accelerator



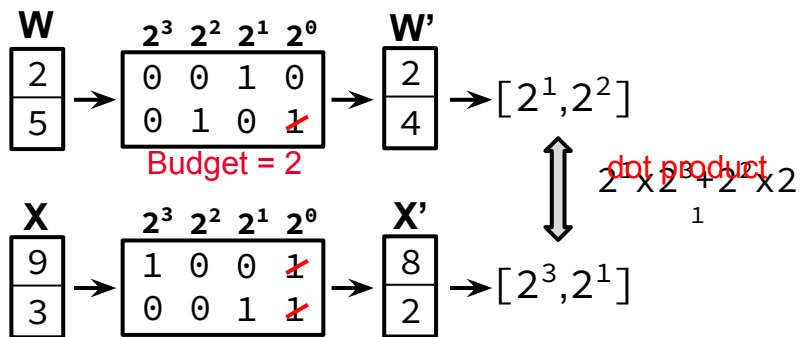
Kung, H. T., Bradley McDanel, and Sai Qian Zhang. "Packing sparse convolutional neural networks for efficient systolic array implementations: Column combining under joint optimization." *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*. 2019. 46

Term Quantization



- Low-precision quantization leads to significant quantization error.
- Both weights and input activation are highly biased in values.

Term Quantization



4-bit uniform quantization

	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
0	0	0	0	0	0	0	0	1
0	0	0	0	1	1	0	0	0
0	0	0	0	0	1	0	1	0
1	0	0	0	1	0	0	0	1

$W'_1 = 0$
 $W'_2 = 0$
 $W'_3 = 0$
 $W'_4 = 128$

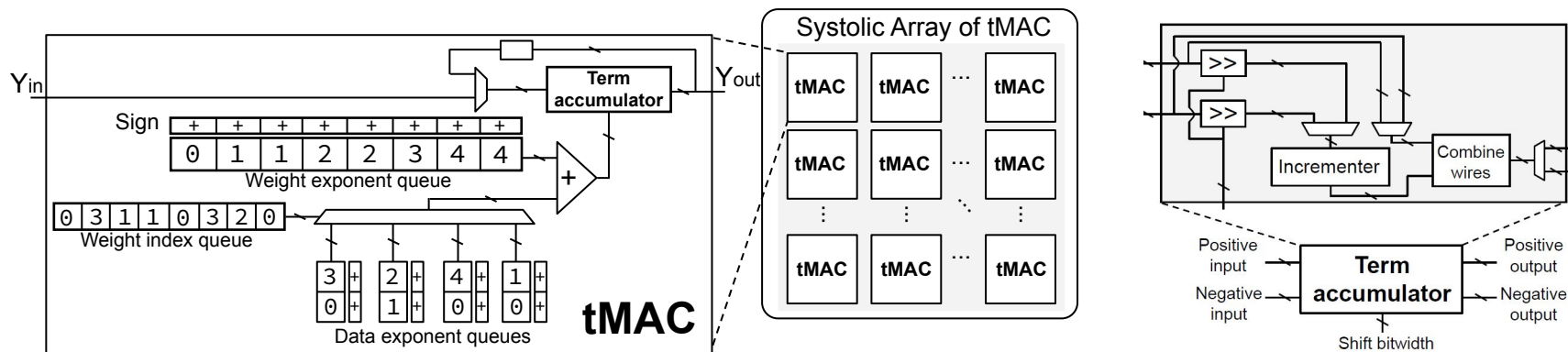
TQ with a budget = 4

	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
0	0	0	0	0	0	0	0	1
0	0	0	0	1	1	0	0	0
0	0	0	0	0	1	0	1	0
1	0	0	0	1	0	0	0	1

$W'_1 = 0$
 $W'_2 = 12$
 $W'_3 = 0$
 $W'_4 = 136$

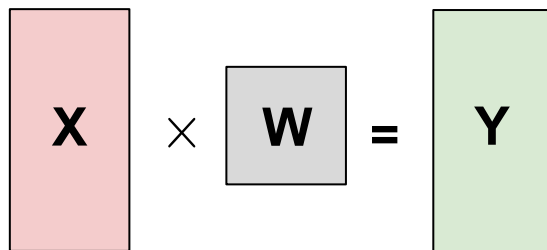
- We can control the term-level computations by setting a **group term budget**.
- For a group of values, we rank and remove the small terms based on this budget.

Term Quantization: Accelerator Design



- We propose the term MAC (tMAC) for the efficient implementation of TQ.
- A tMAC processes all term-pair multiplications across a group of weight and data values.
- Each term is represented by their corresponding exponent (2-3 bits).
- The term accumulation can be implemented using half adders.

Efficient DNN Training: Forward Pass


$$\mathbf{X} \times \mathbf{W} = \mathbf{Y}$$

X: input maps

W: weight filters

Y: output maps

- The convolutional operations during the forward propagation can be converted into matrix multiplications

Efficient DNN Training: Backward Pass

Input data gradient
Computation

$$\nabla Y \times W^T = \nabla X$$

X: input maps
 ∇X : input gradient

Weight gradient
Computation

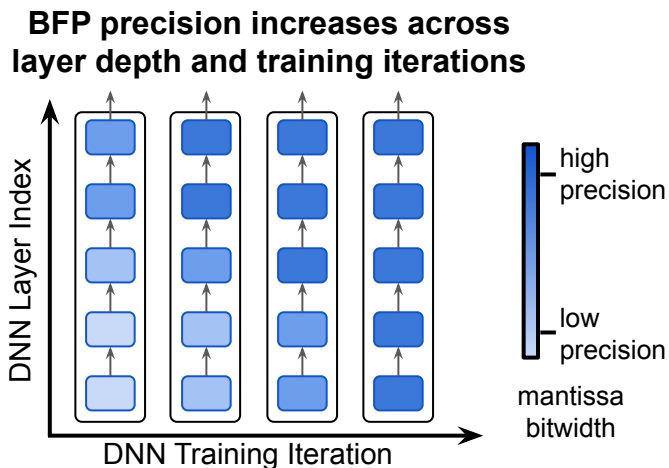
$$X^T \times \nabla Y = \nabla W$$

W: weight filters
 ∇W : weight gradient

Y: output maps
 ∇Y : output gradient

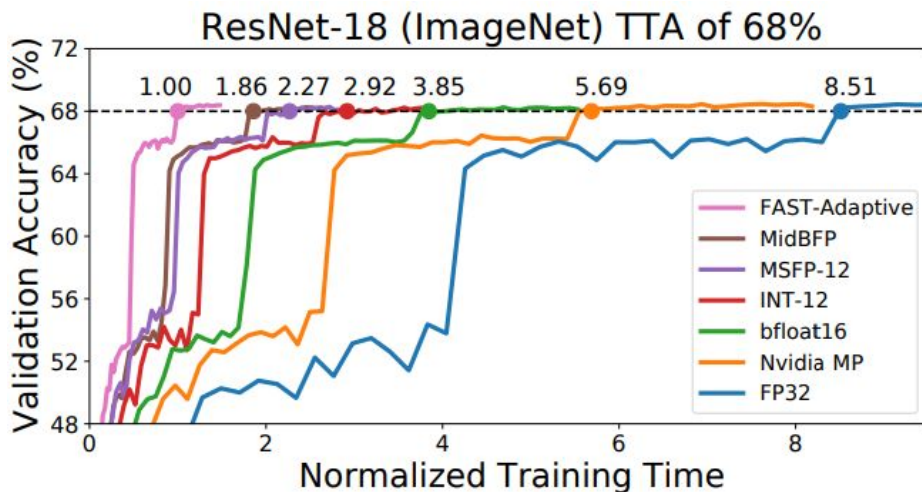
- DNN backward propagation involves two matrix multiplications

Efficient DNN Training: FAST Algorithm



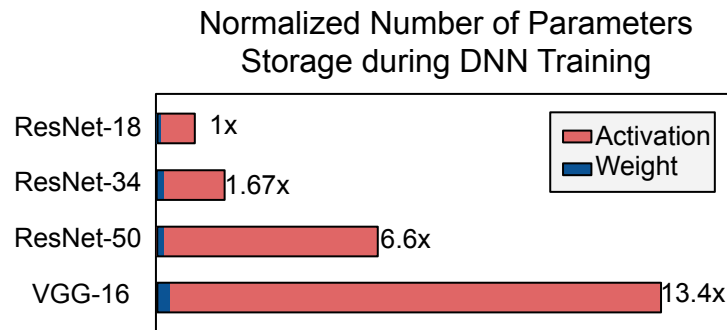
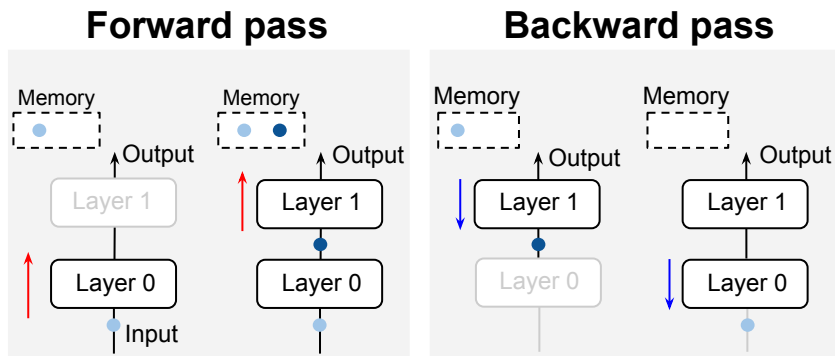
- We name this approach **FAST** (Fast First, Accurate Second Training)
- We linearly increase the training precision across both layer depth and training iterations

Efficient DNN Training: FAST Algorithm

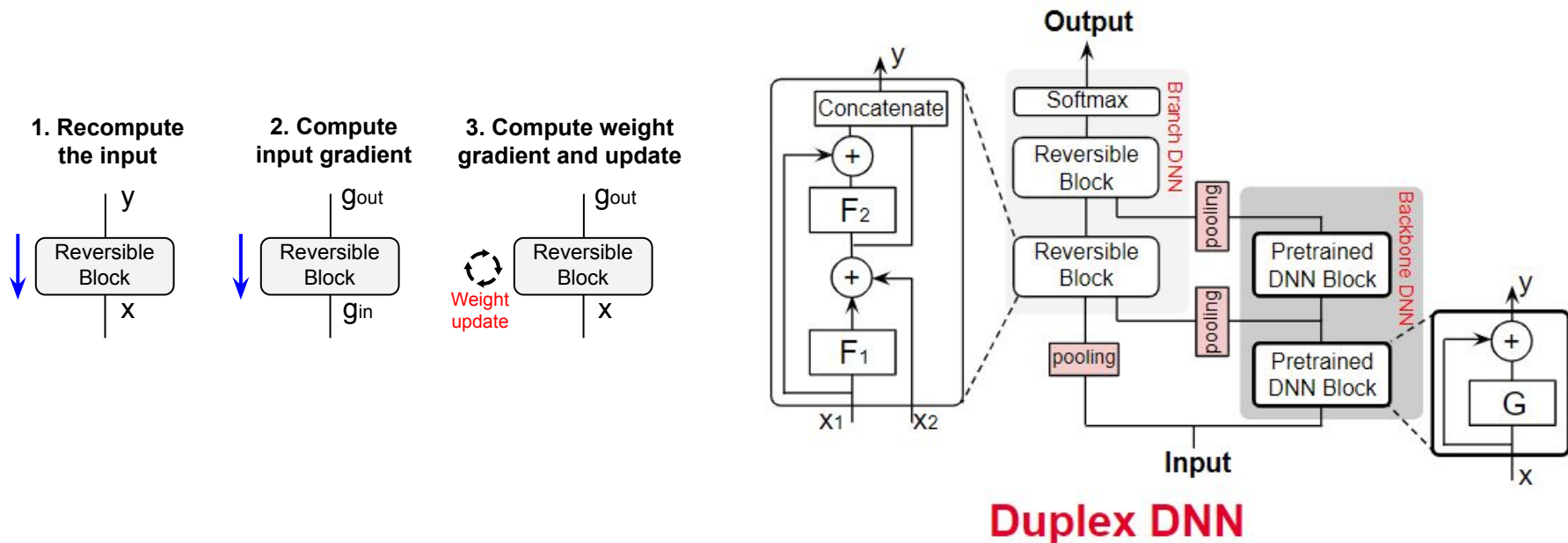


- We use Time-to-Accuracy (TTA) as the evaluation metric to compare different approaches
- Our FAST approach achieves the lowest TTA across all the numeric formats

Reversible DNN for Efficient On-chip Learning

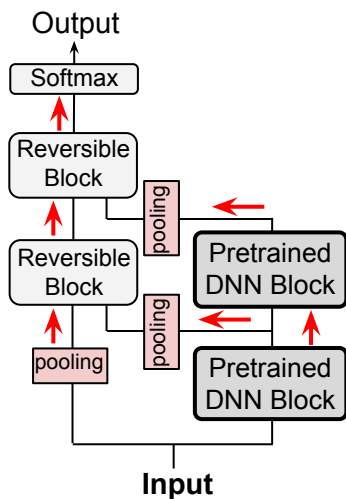


Reversible DNN for Efficient On-chip Learning

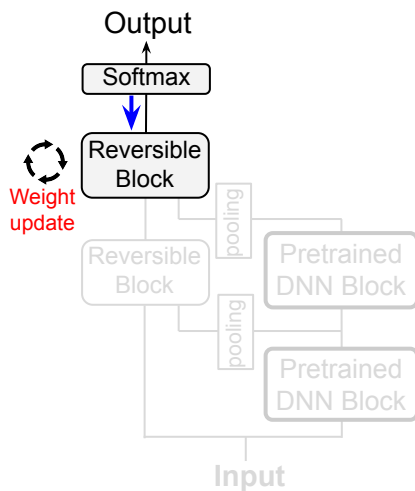


Reversible DNN for Efficient On-chip Learning

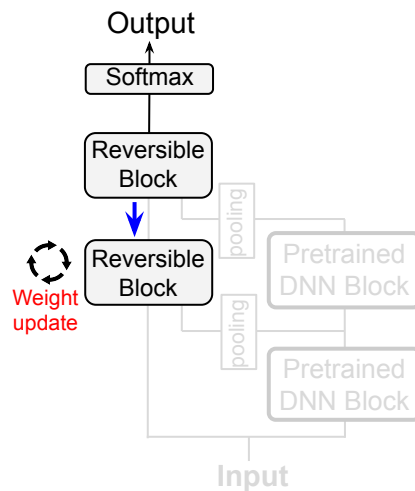
— Forward pass — Backward pass



Forward Pass

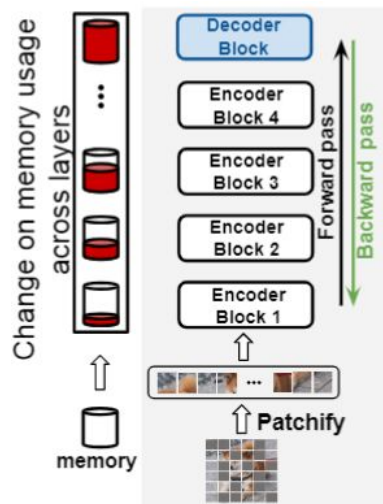


Backward Pass
(Phase 1)

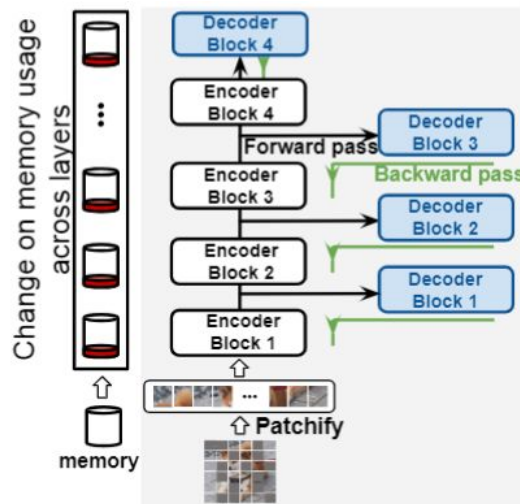


Backward Pass
(Phase 2)

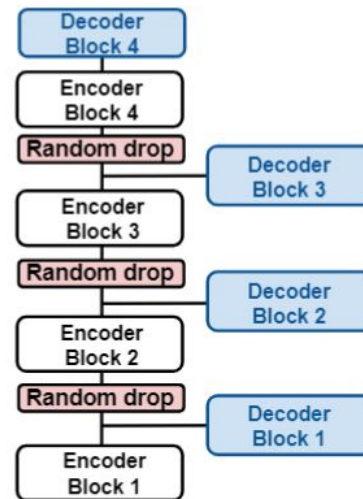
Efficient Deep Self-Supervised Learning



(a) Conventional MIM

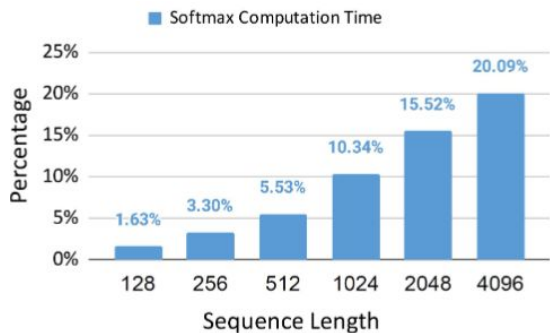


(b) BIM

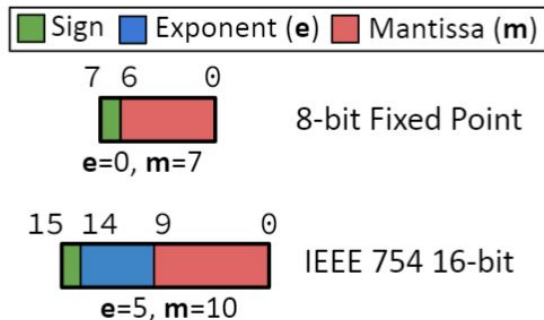


(c) BIM with incremental masking

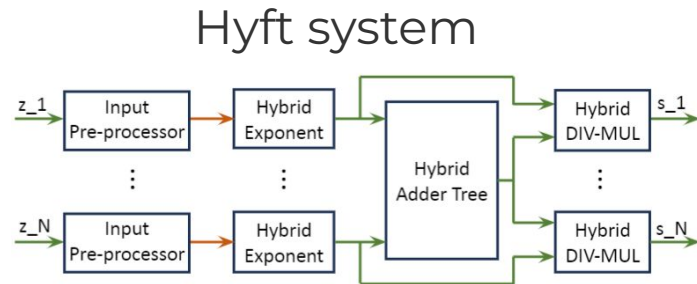
Softmax Acceleration in Large Models



(a)

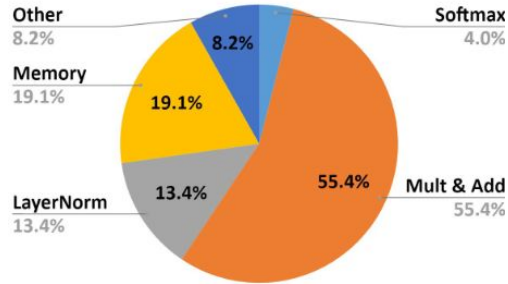
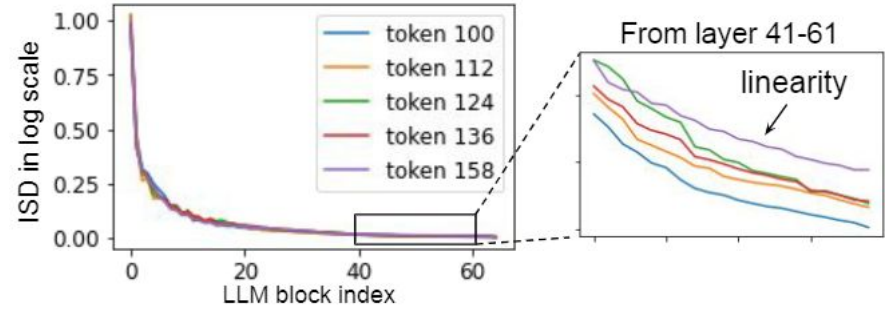
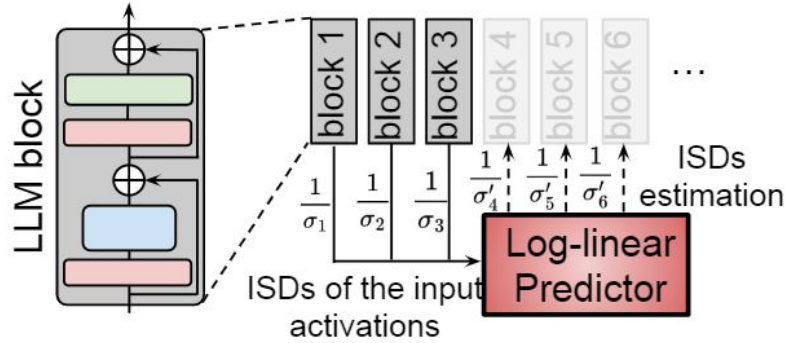


(b)



(c)

Normalization Acceleration in Large Models



- We adopt the principles of algorithm and hardware co-design, introducing a holistic normalization accelerating method.
- We leverage on the strong correlation observed in normalization statistics across consecutive layers, enabling the bypassing of normalization computation through the estimation of statistics