Introduction to Inference

Micah Villmow, Principal TensorRT Engineer

Hotchips Tutorial on ML-Inference 8/27/2023

Agenda

- Inference Introduction
- Ecosystem
- Optimization
- Execution

Inference Introduction

What Is Inference?

"Inference is **a conclusion reached on the basis of evidence and reasoning**[1]"

— Oxford Dictionary

Conclusion:

The output of the network

Evidence:

Prior knowledge is the trained weights

Current knowledge is the input activations

Reasoning:

The reasoning is implicit in the network

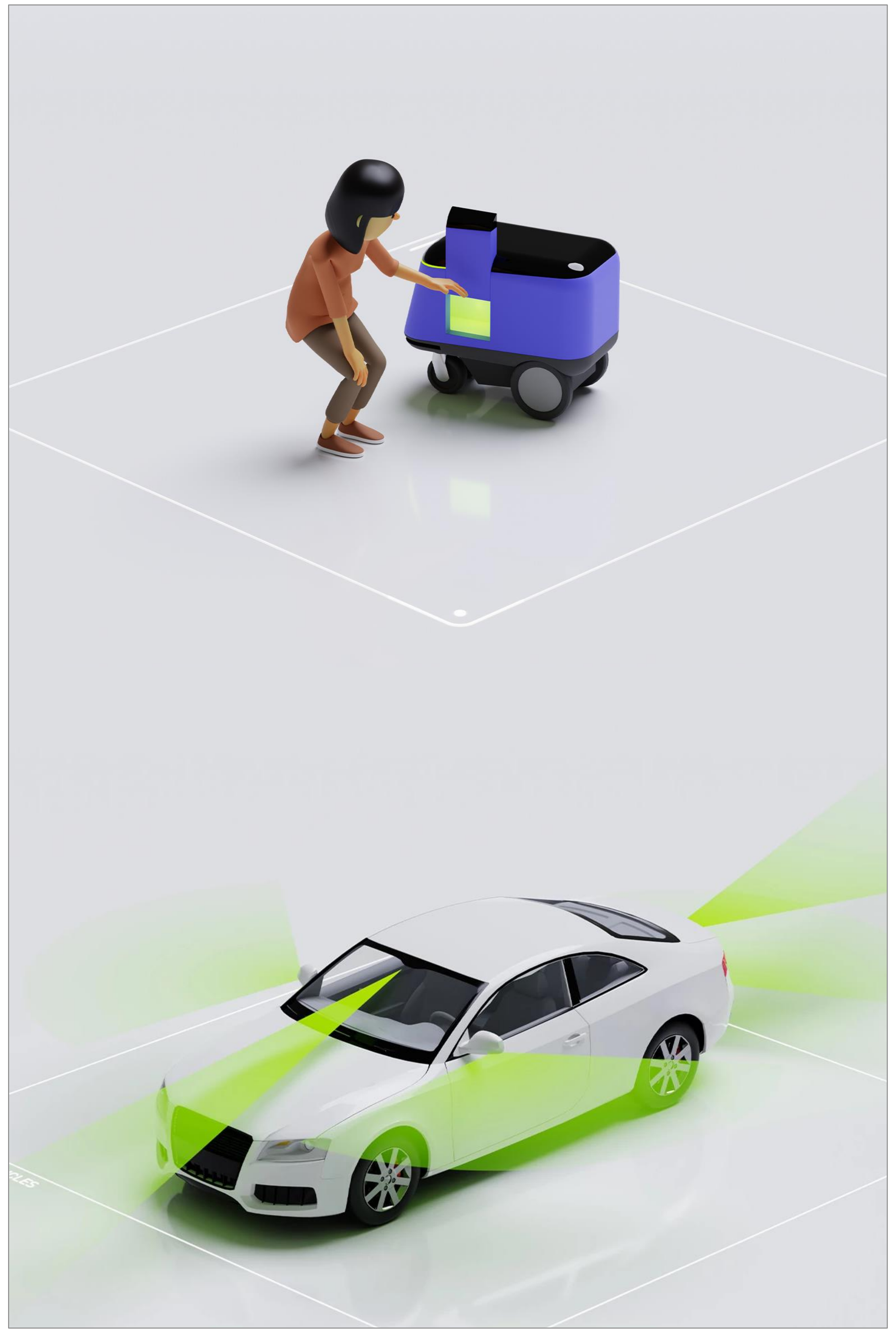
[1] *https://www.oed.com/search/dictionary/?scope=Entries&q=inference*

Difference From Training

- No backward passes.
- Weights are read-only.
- Activations do not need to be stored.
- Dataset is unknown.
- The data can be, but is not required to be, normalized.
- Optimize for different metrics vs throughput for training.

AI Inference Drives The Modern Applications

Demand for fast, easy inference deployment greater than ever



Computer Vision

Conversational AI

Recommender System

Fraud Detection

Ecosystem

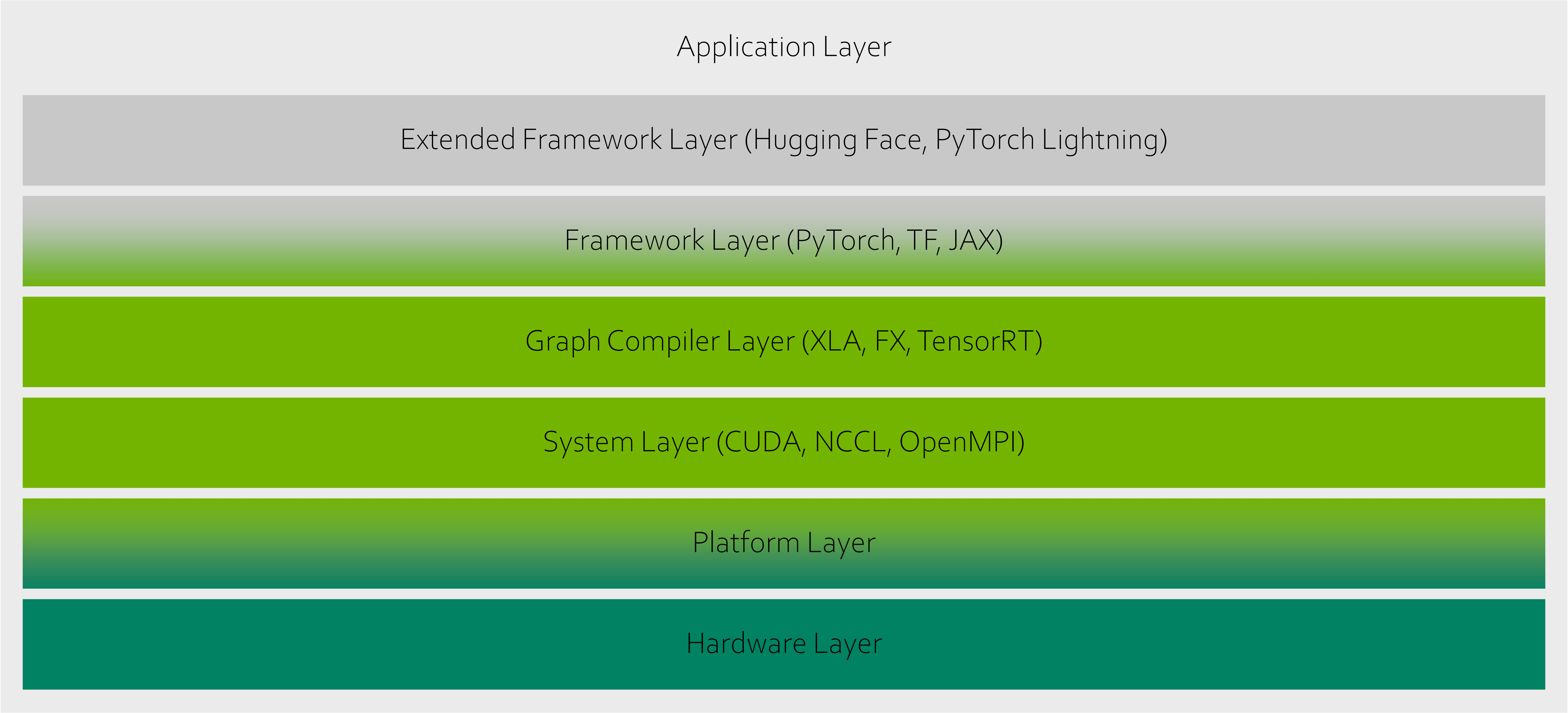
Hardware Ecosystem

Embedded Automotive Data Center Edge Gaming Professional

VPU CPU DPU TPU GPU SOC WaferScale

Inference Software Ecosystem

A simplified model



Application Layer

Extended Framework Layer (Hugging Face, PyTorch Lightning)

Framework Layer (PyTorch, TF, JAX)

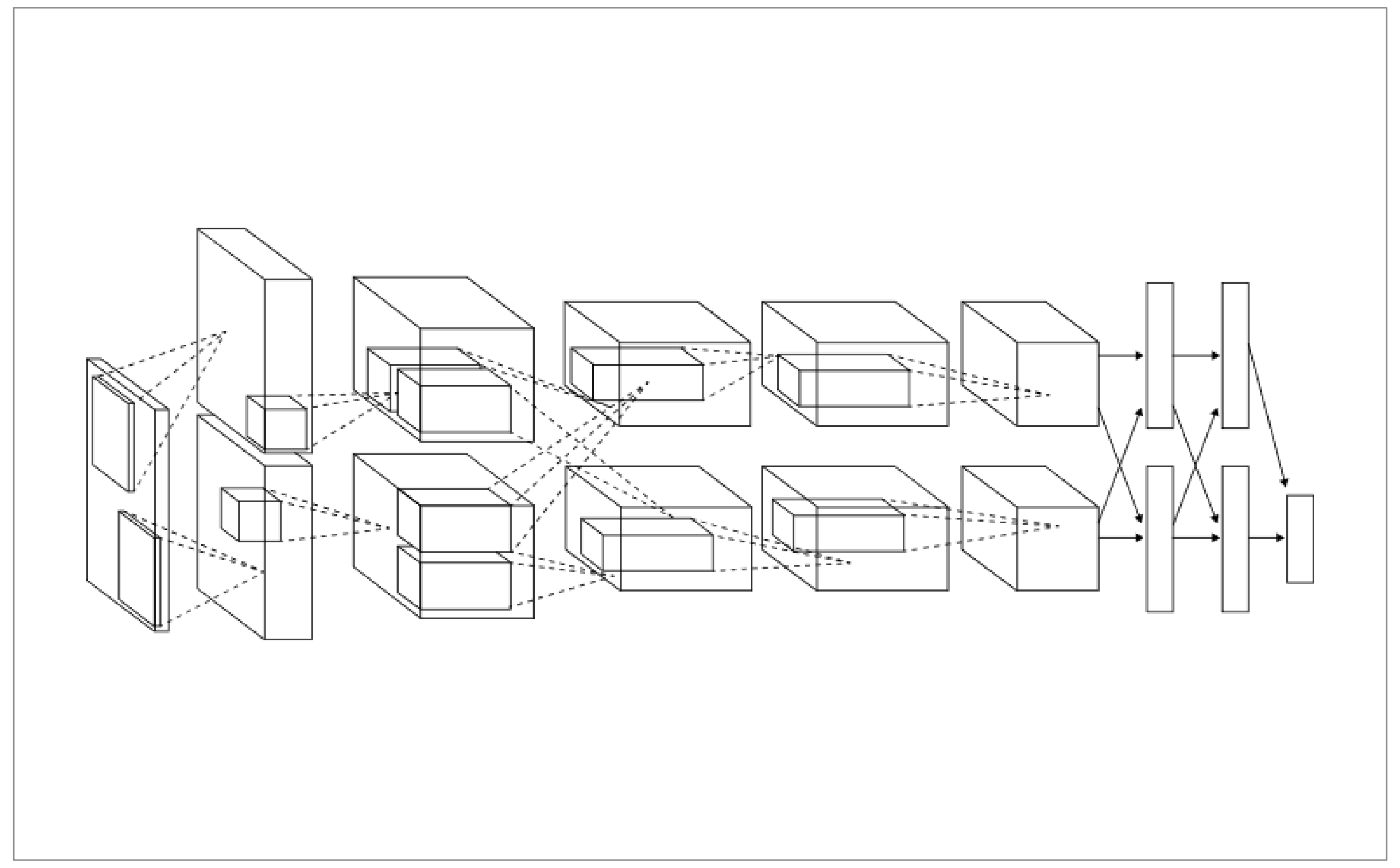
Graph Compiler Layer (XLA, FX, TensorRT)

System Layer (CUDA, NCCL, OpenMPI)

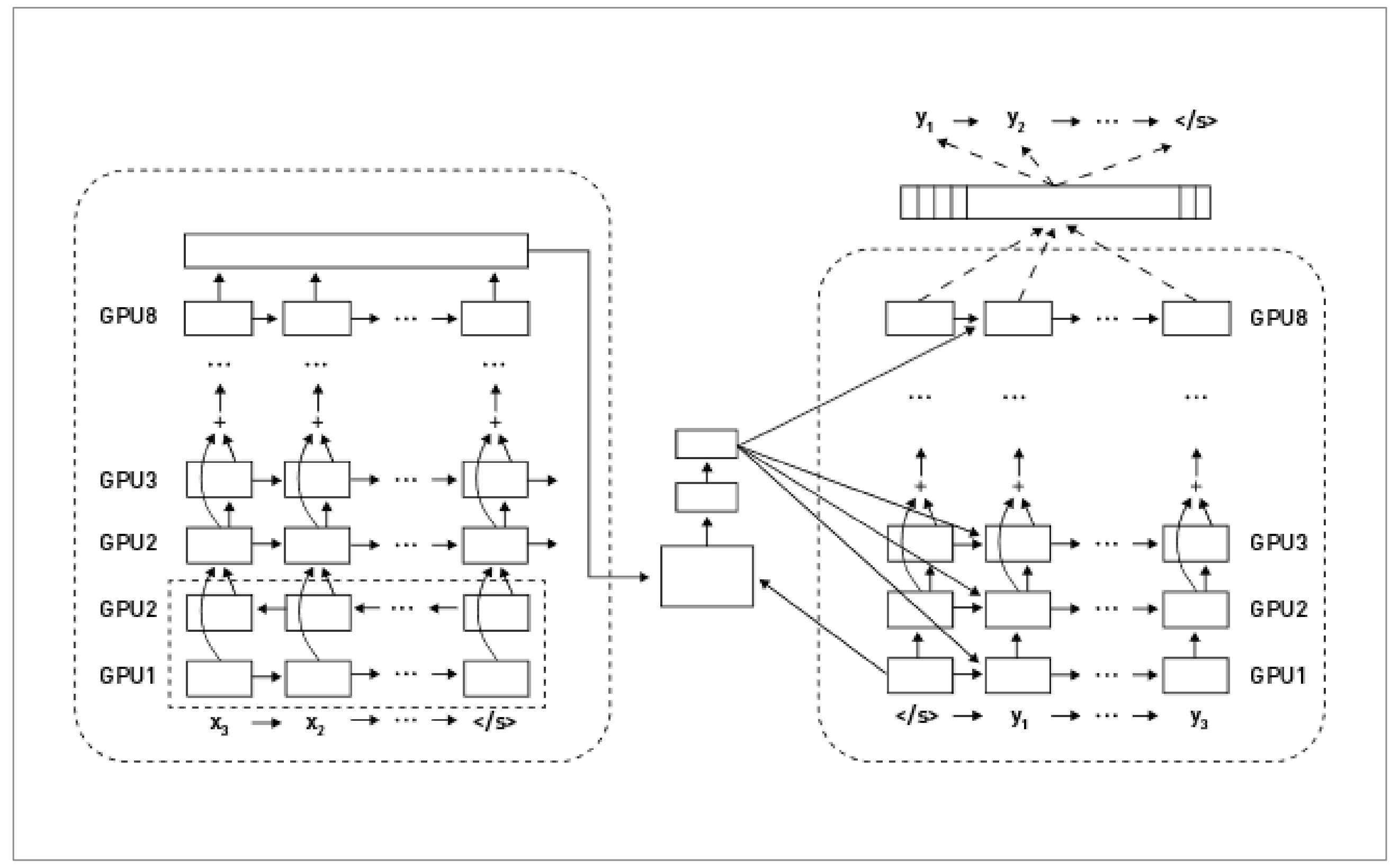
Platform Layer

Hardware Layer

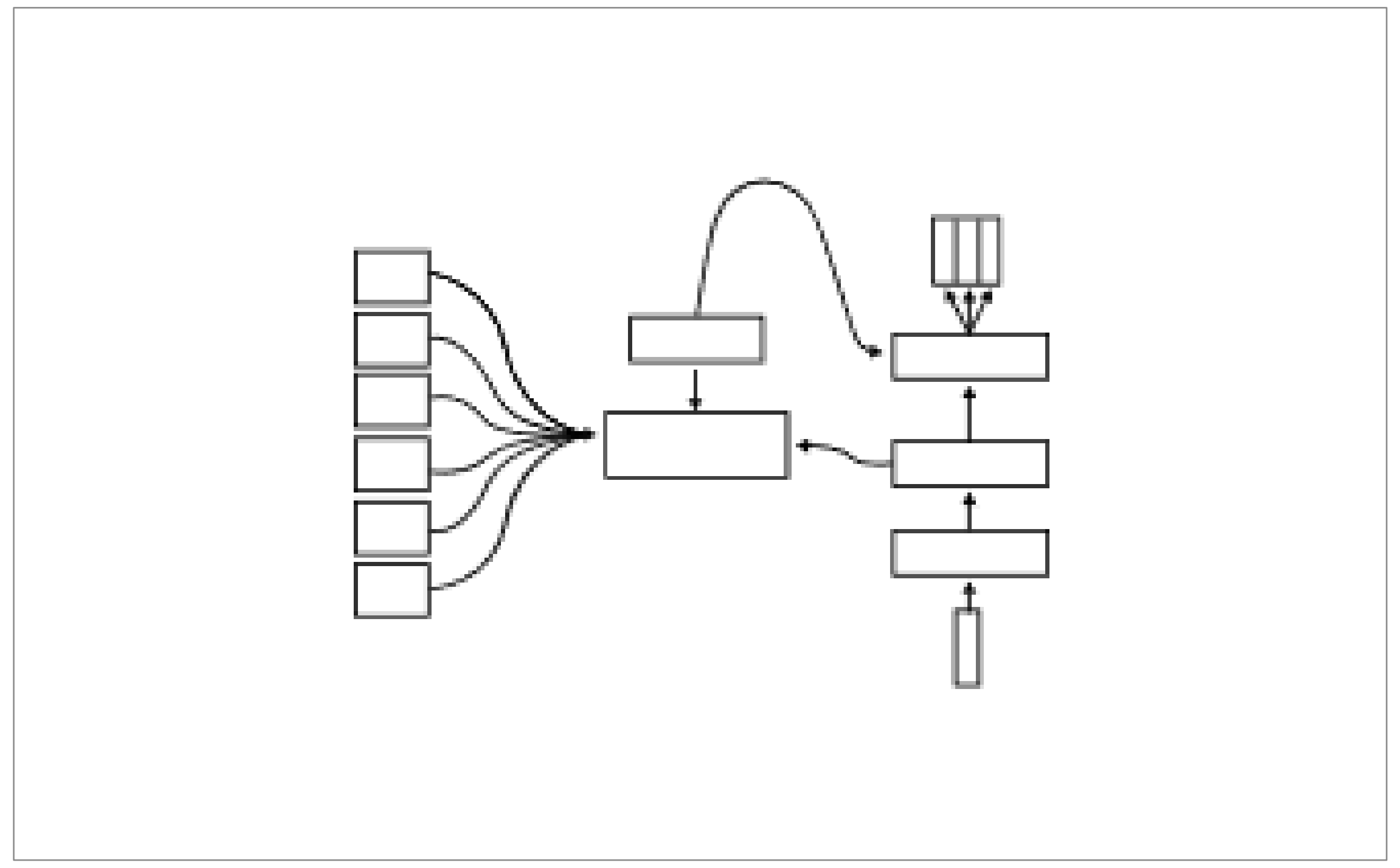
Cambrian Model Explosion



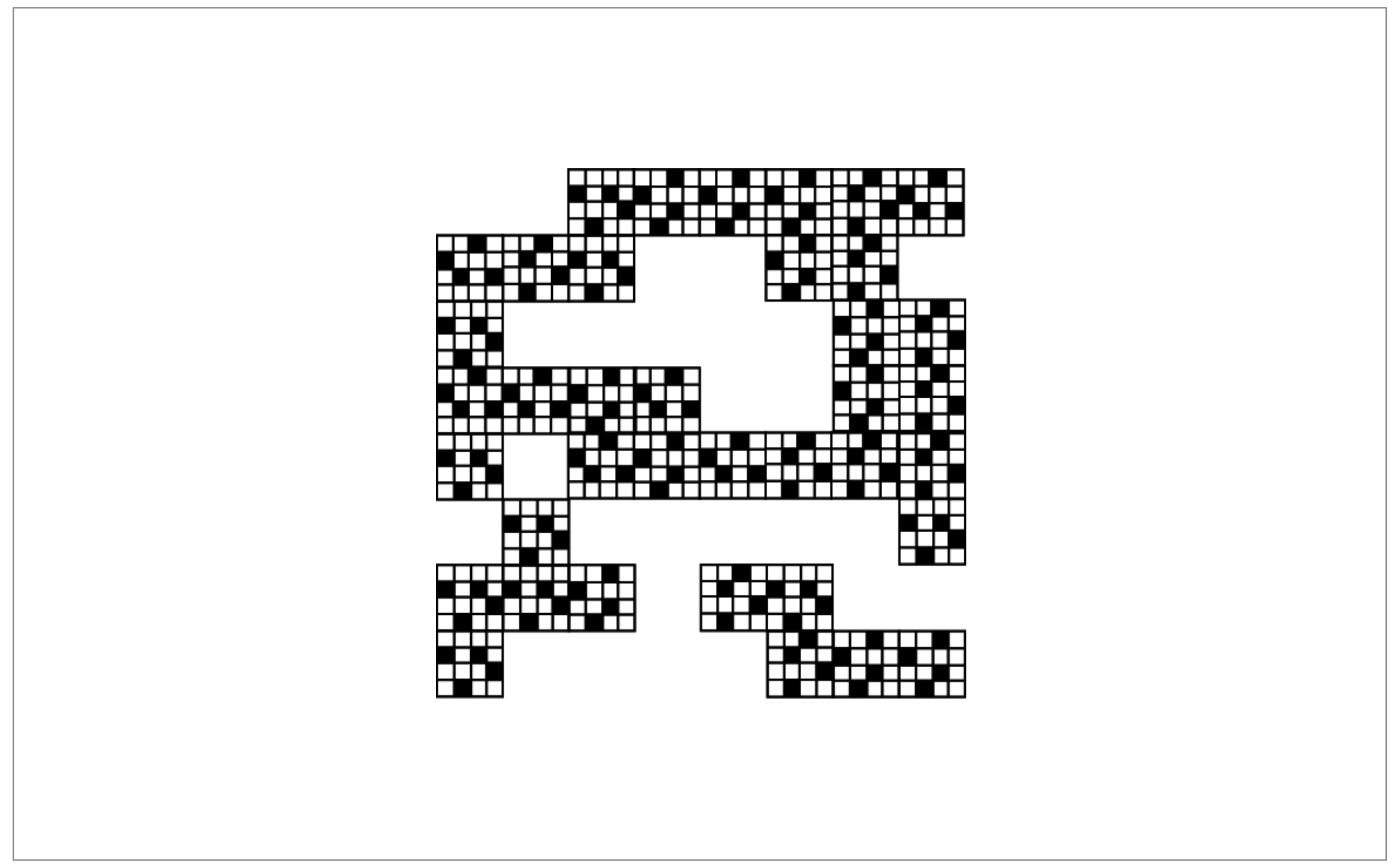
CNNs



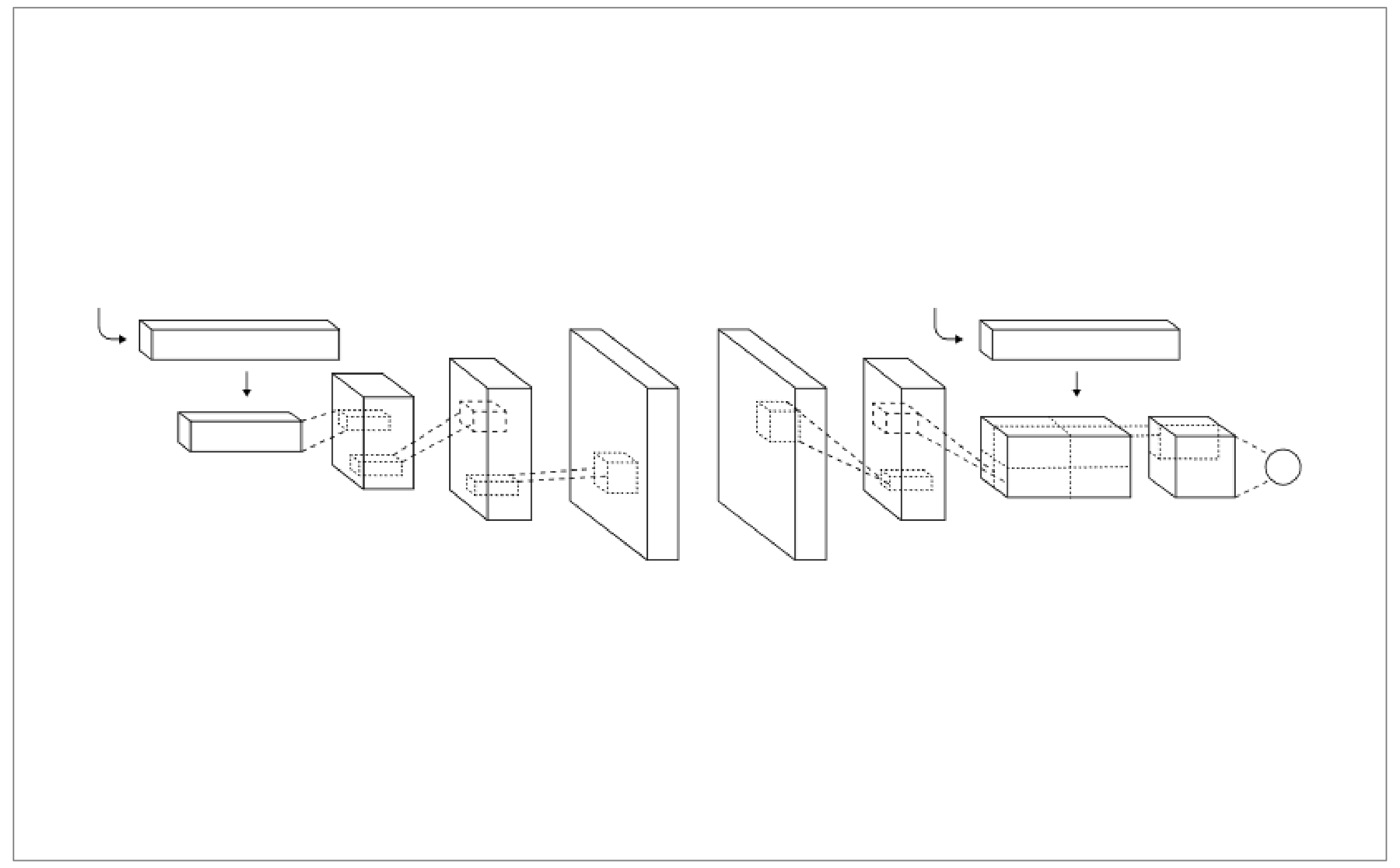
RNNs



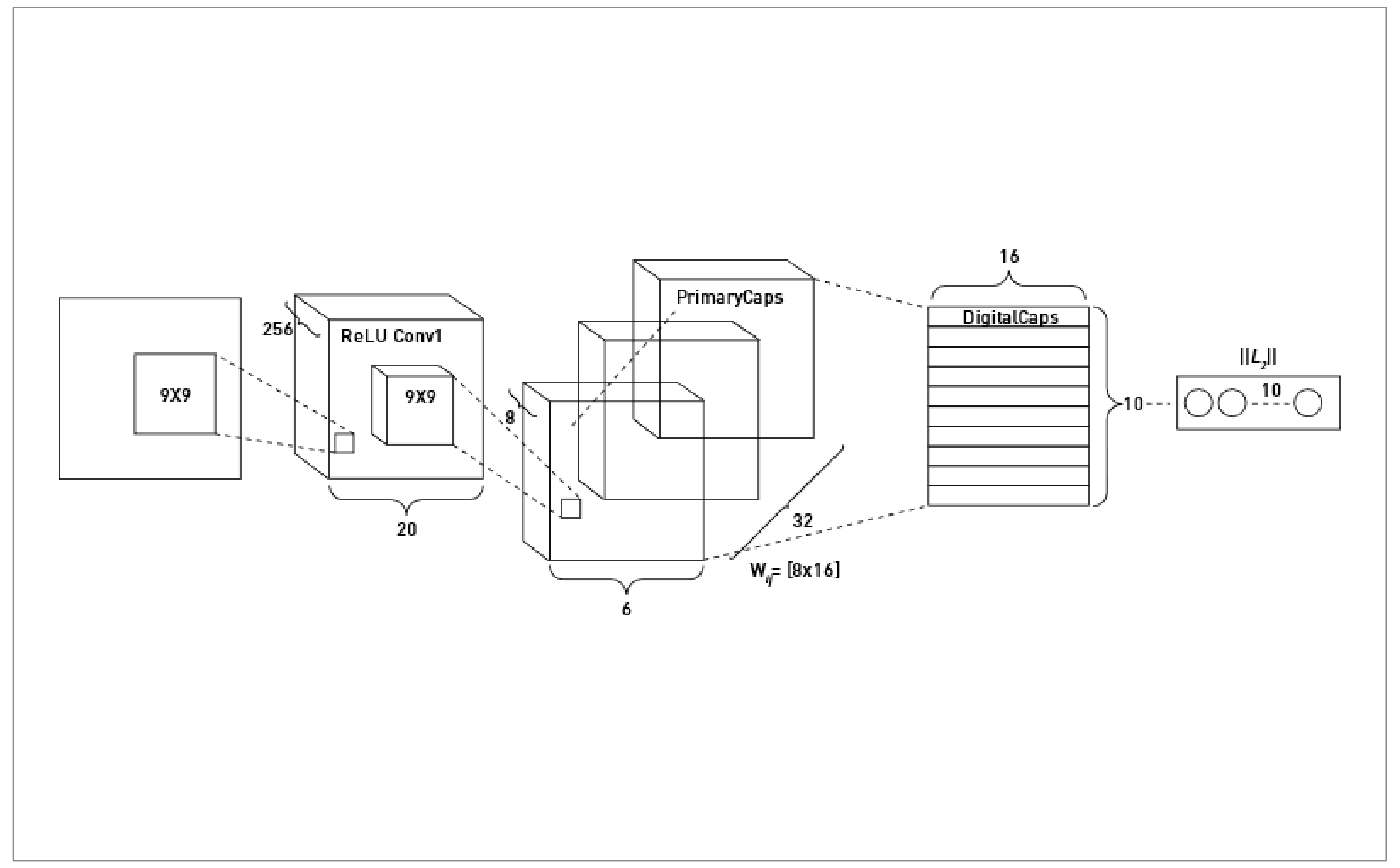
Transformers



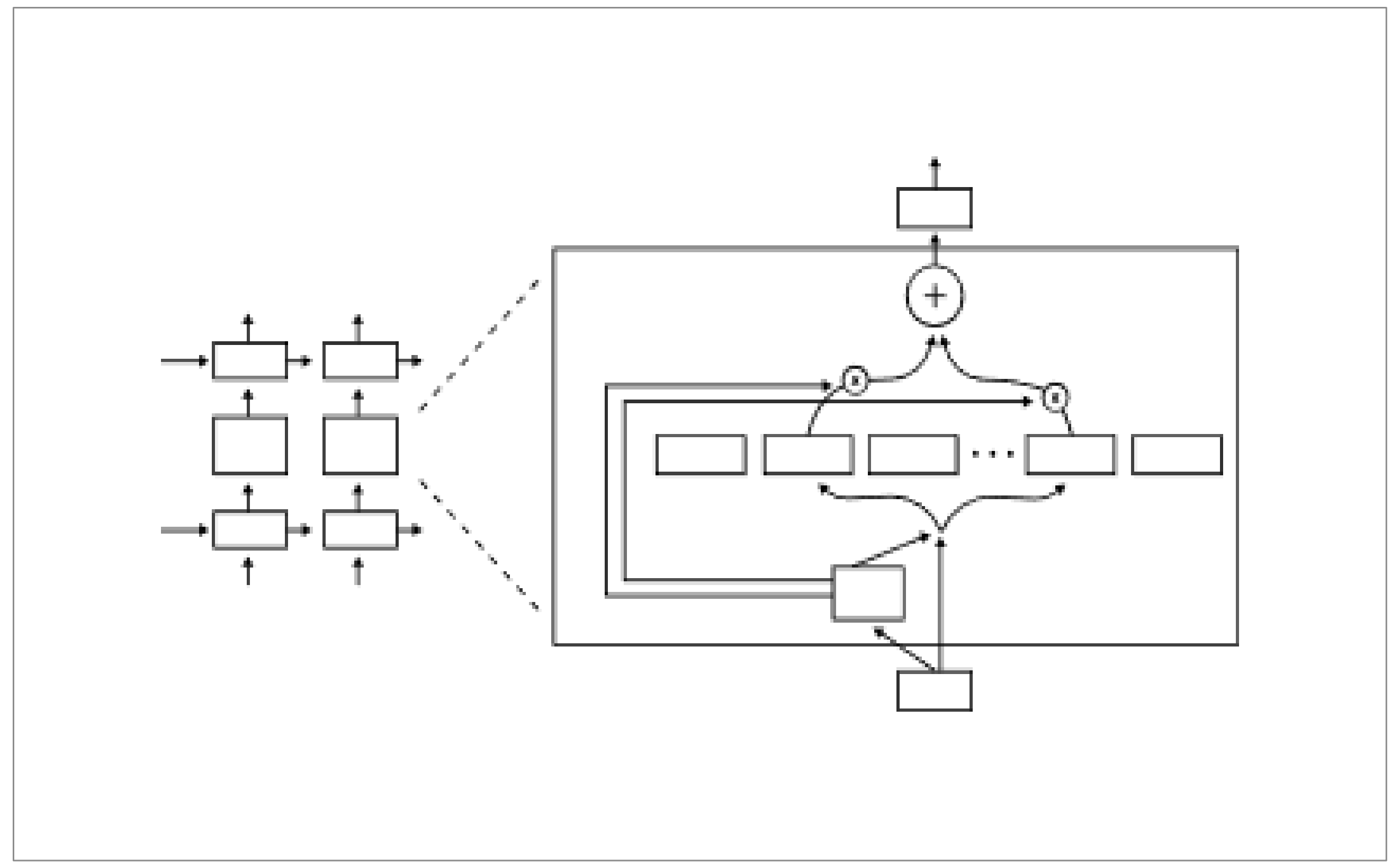
Sparse LSTMs



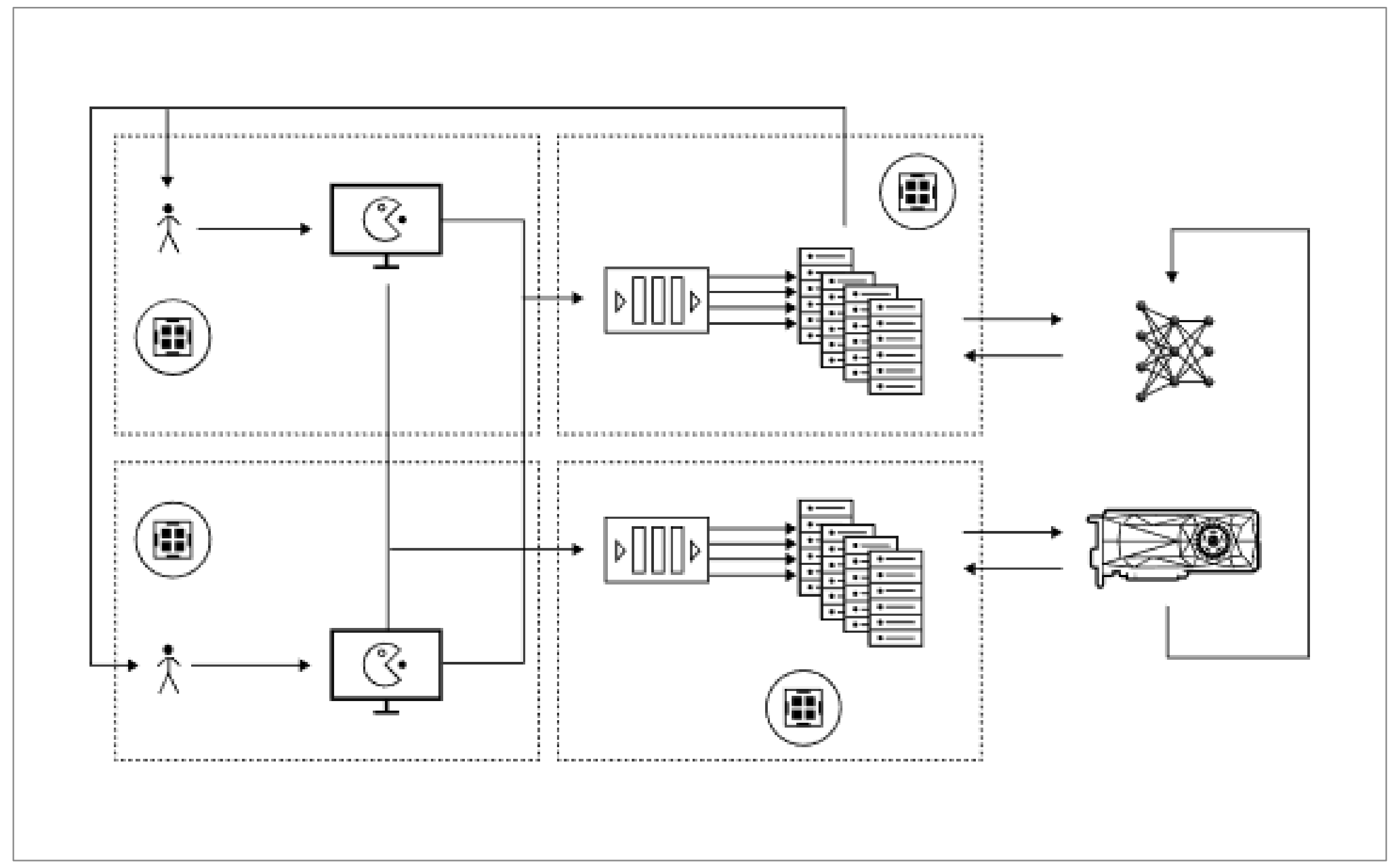
GANs



Capsule Nets



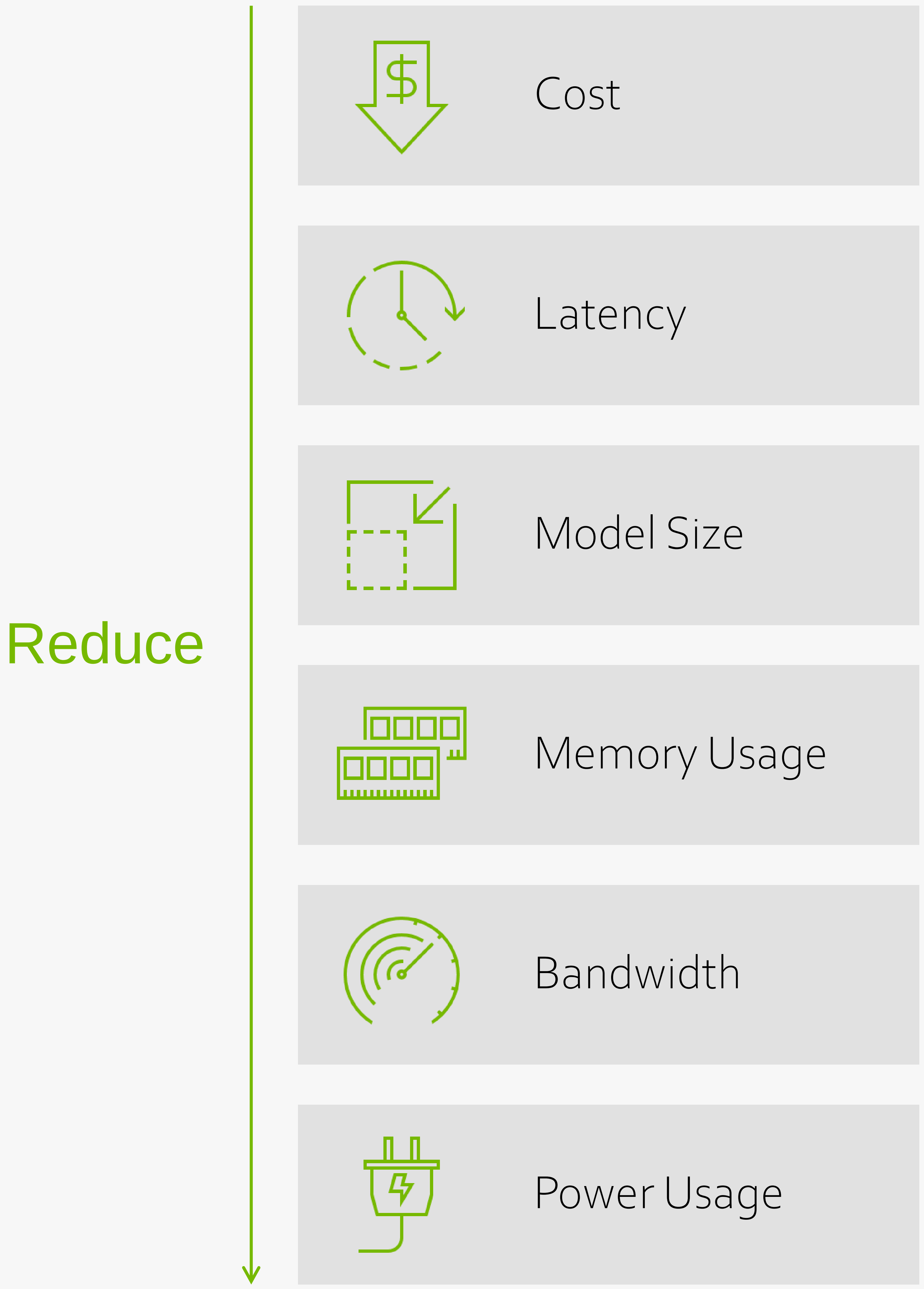
Large Language Models



Reinforcement Learning

Optimizations

Why Optimize?



Reduce

- Cost
- Latency
- Model Size
- Memory Usage
- Bandwidth
- Power Usage

Precision

What is the best format?

FP Format	Exponent*	Significand+
IEEE-754 64bit	11	52
IEEE-754 32bit	8	23
TensorFloat-32 19bit	8	10
IEEE-754 16bit	5	10
BFloat 16bit	8	7
FP8E4M3	4	3
FP8E5M2	5	2

Integer Formats	
Int64	Int32
Int16	Int8
Int4	Int2
Uint64	Uint32
Uint16	Uint8
Uint4	Uint2
Uint1	

** Exponent determines the range*
+ Significand size determines precision

Layouts

What is the best representation?

Scalar Format

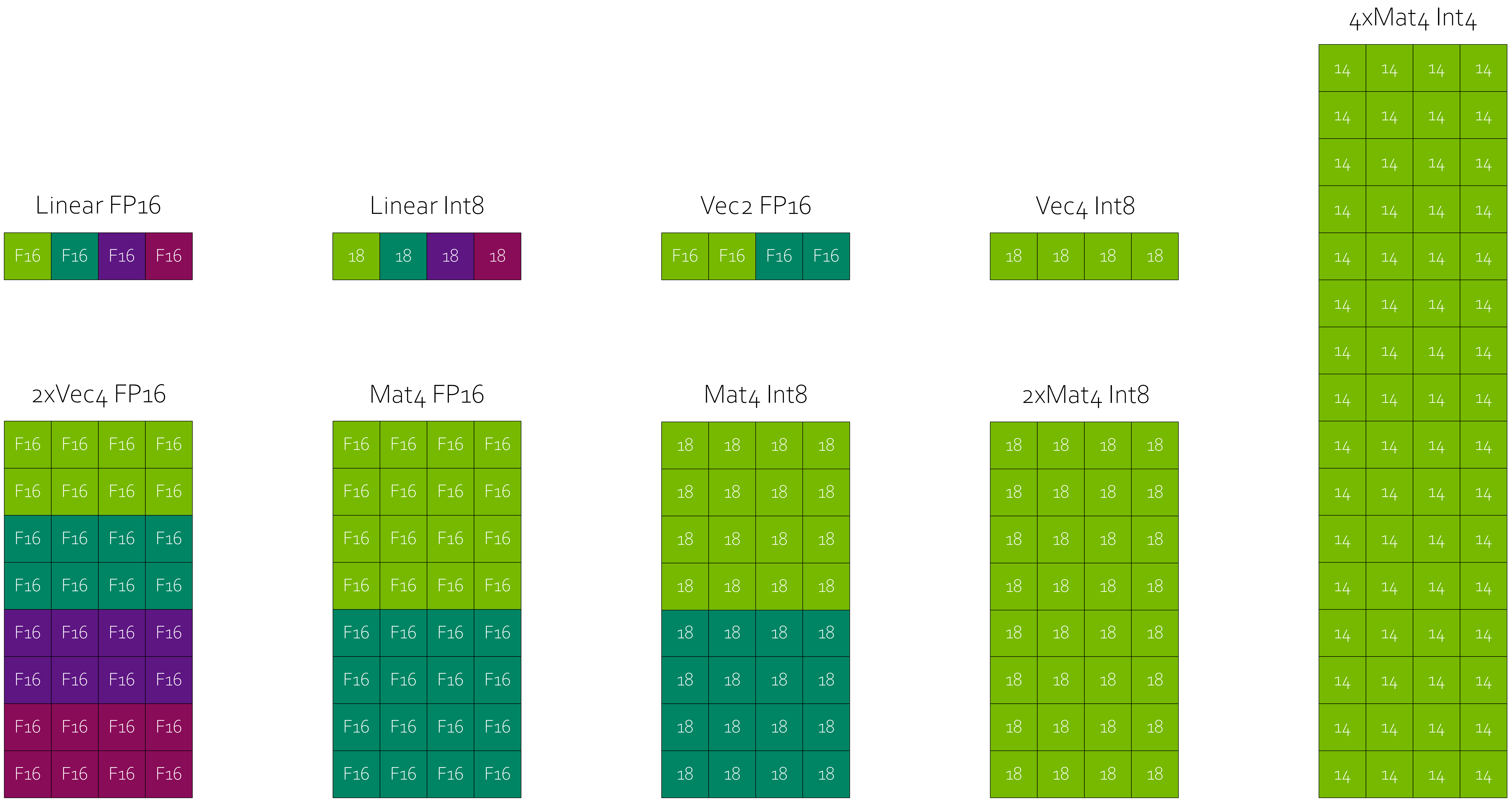
2-wide Vector Format

4-wide Vector Format

8-wide Vector Format

Memory — Formats

Combination of Precision and Layout



Linear FP16

F16	F16	F16	F16

Linear Int8

18	18	18	18

Vec2 FP16

F16	F16	F16	F16

Vec4 Int8

18	18	18	18

4xMat4 Int4

14	14	14	14
14	14	14	14
14	14	14	14
14	14	14	14
14	14	14	14
14	14	14	14
14	14	14	14
14	14	14	14
14	14	14	14
14	14	14	14
14	14	14	14
14	14	14	14
14	14	14	14
14	14	14	14
14	14	14	14
14	14	14	14

2xVec4 FP16

F16	F16	F16	F16
F16	F16	F16	F16
F16	F16	F16	F16
F16	F16	F16	F16
F16	F16	F16	F16
F16	F16	F16	F16
F16	F16	F16	F16
F16	F16	F16	F16

Mat4 FP16

F16	F16	F16	F16
F16	F16	F16	F16
F16	F16	F16	F16
F16	F16	F16	F16
F16	F16	F16	F16
F16	F16	F16	F16
F16	F16	F16	F16
F16	F16	F16	F16

Mat4 Int8

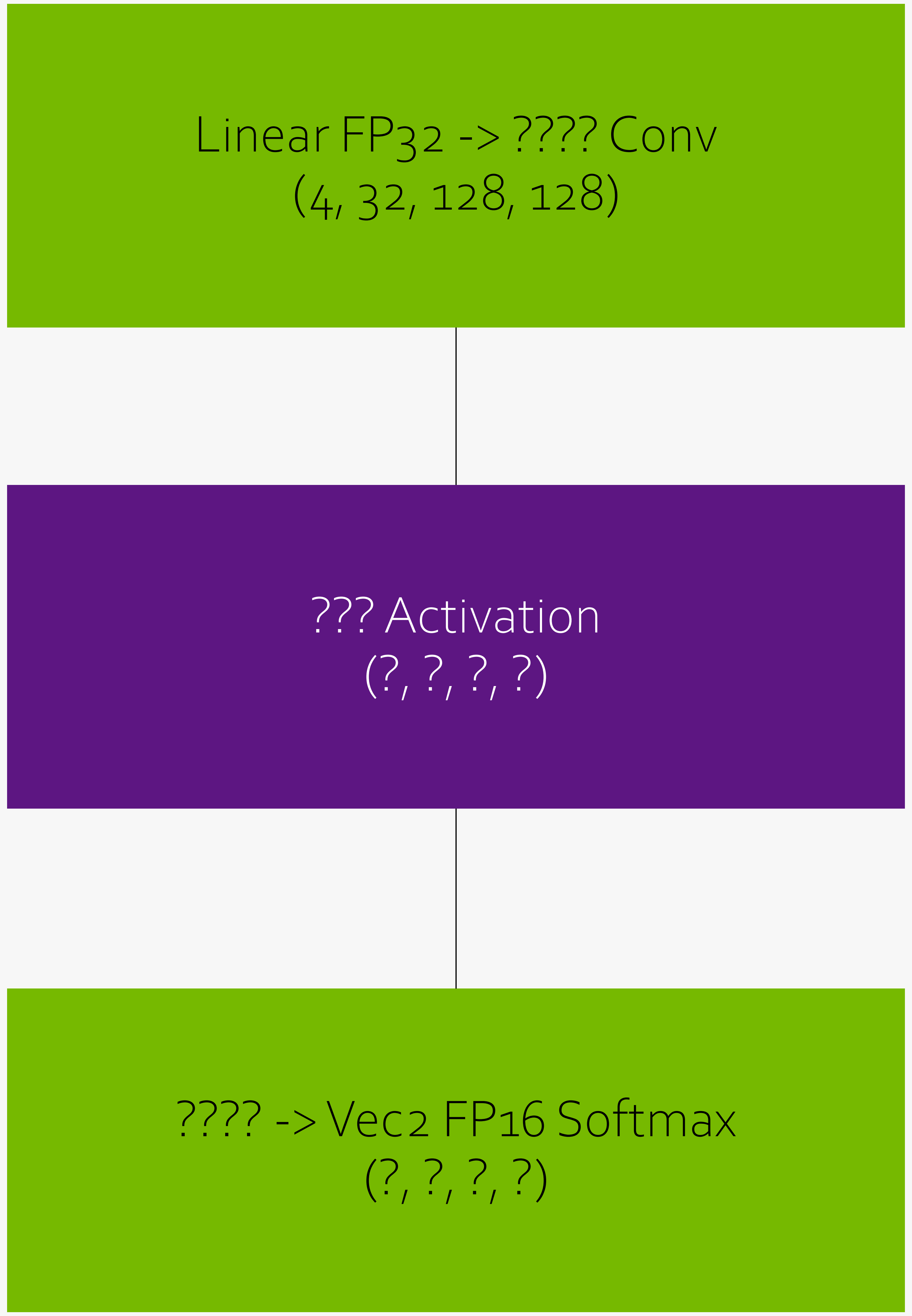
18	18	18	18
18	18	18	18
18	18	18	18
18	18	18	18
18	18	18	18
18	18	18	18
18	18	18	18
18	18	18	18

2xMat4 Int8

18	18	18	18
18	18	18	18
18	18	18	18
18	18	18	18
18	18	18	18
18	18	18	18
18	18	18	18
18	18	18	18

Type and Shape inference

- What are the input and output types of the purple layer?
- What are the output dimensions of the Softmax?



Dynamic — Dimensions

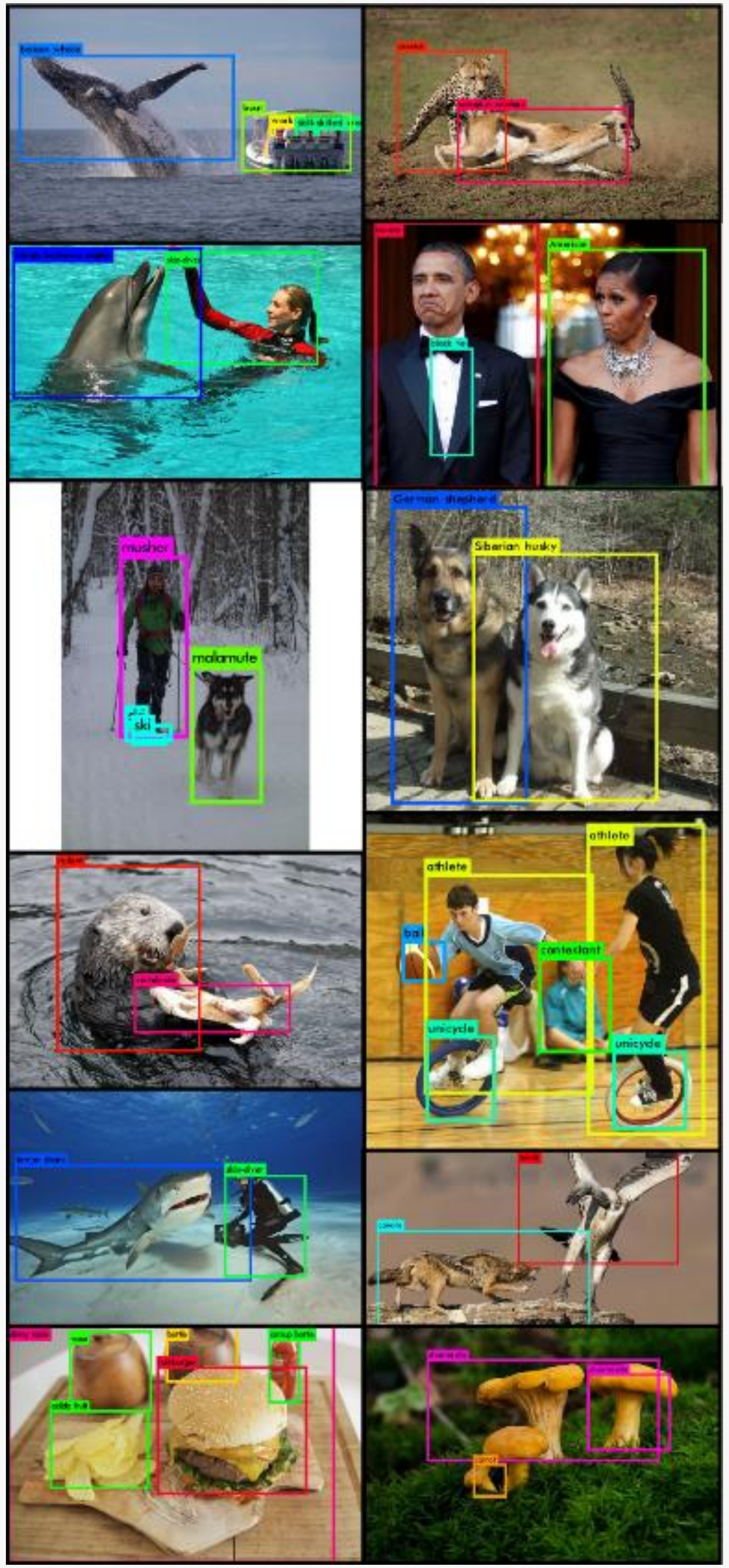
Variable sized inputs for a network

- How many video formats are there?
- Image sizes?
- Sentence lengths?
- Dictionary sizes?
- Object counts?

5:4	4:3	3:2	16:10	16:9
SXGA 1280x1024	QVGA 320x240	NTSC 720x480	CGA 320x200	WVGA 854x480
QSXGA 2560x2048	VGA 640x280	1152x768	WSXGA+ 1680x1050	HD 720 1280x720
	PAL 768x576	1280x854	WUXGA 1920x1200	HD 1080 1920x1080
	SVGA 800x600	1440x960	WQXGA 2560x1600	
	XGA 1024x768			
	1280x960			
	SXGA+ 1400x1050			
	UXGA 1600x1200			
	QXGA 2048x1536			

Dynamic
Data Dependent Shapes

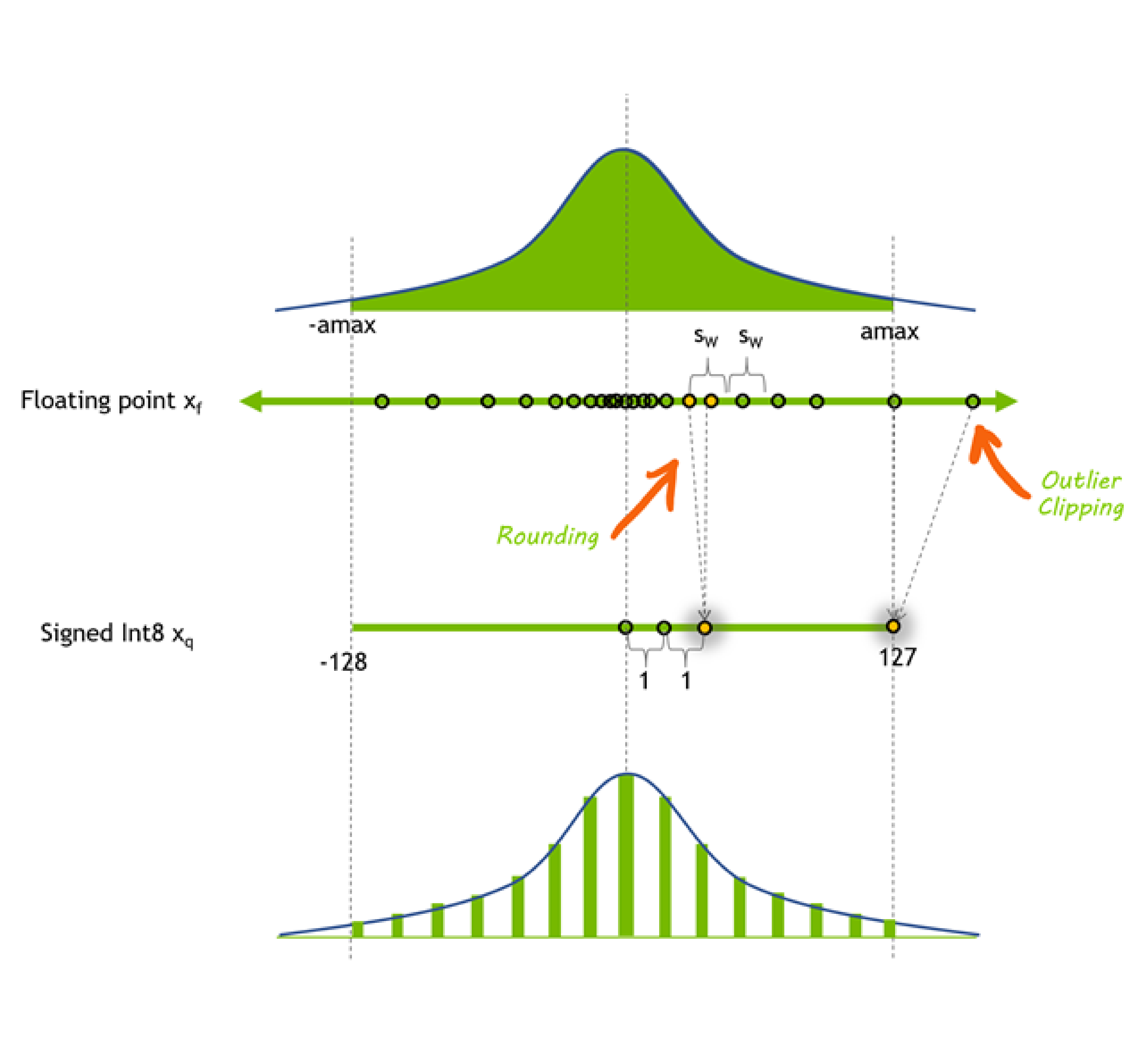
- How many objects are in each image?



Quantization

Do almost the same with less

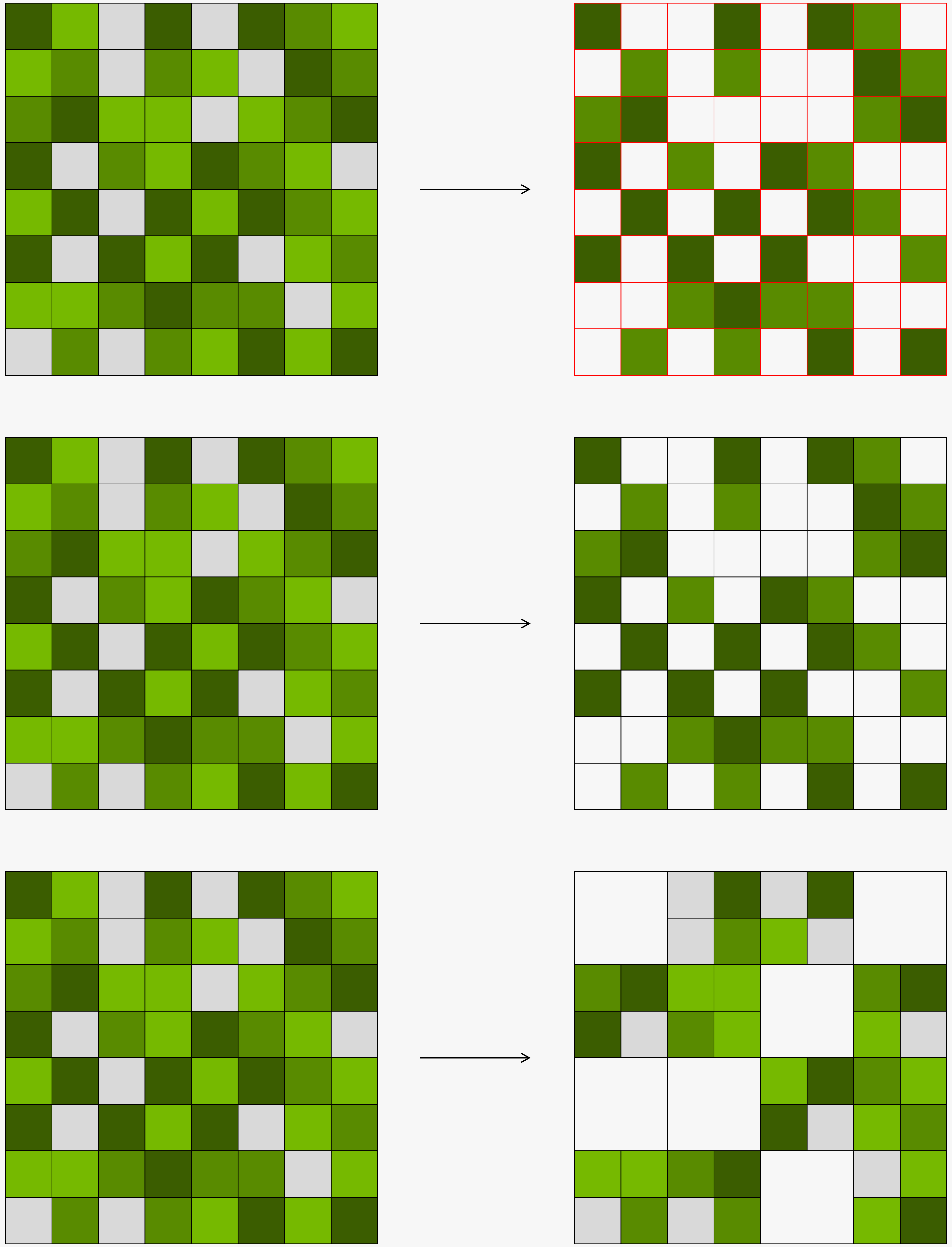
- Decreases latency and storage
- Balances between truncation and discretization



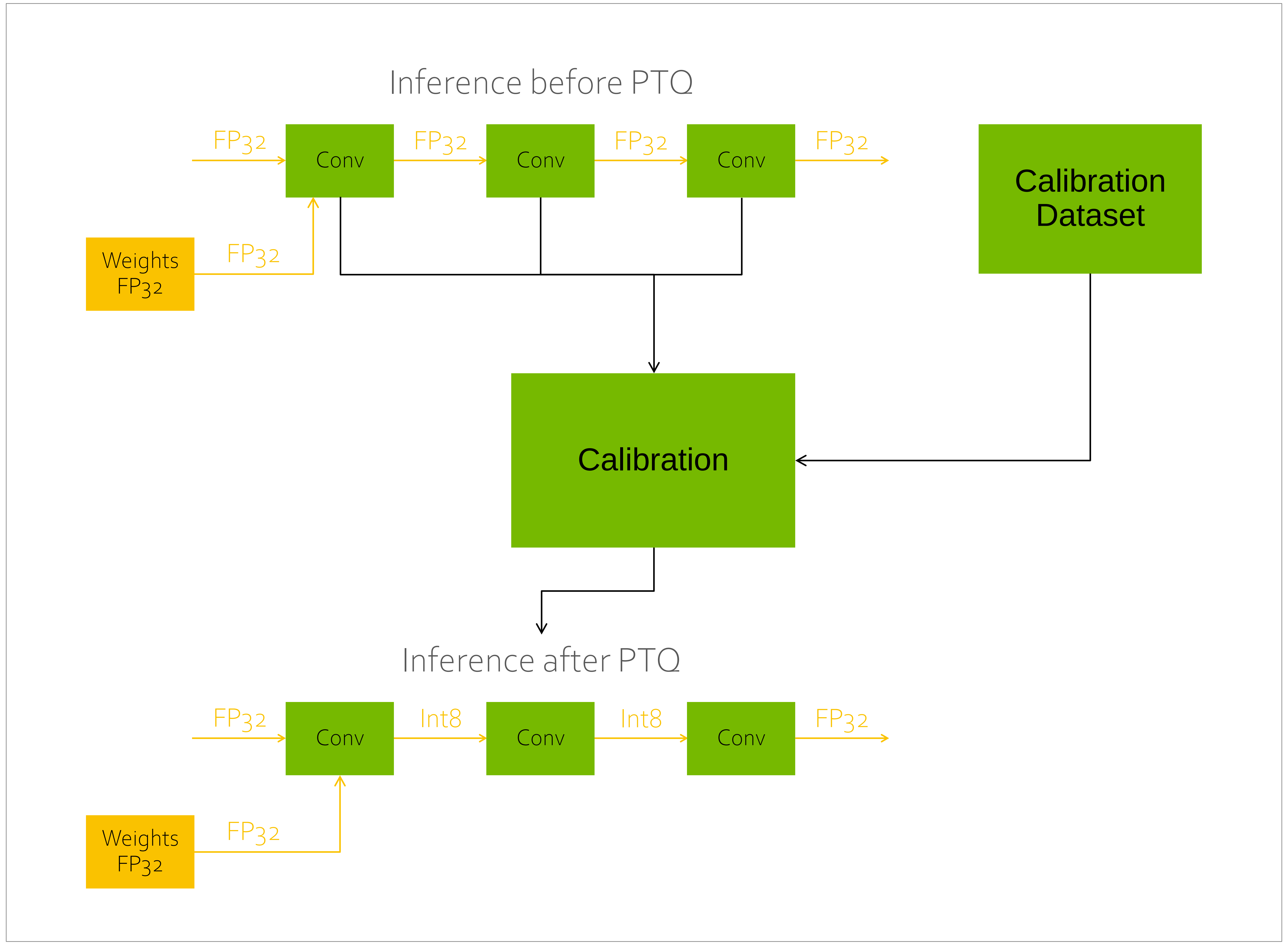
Activations — Sparsity

Types of sparsity

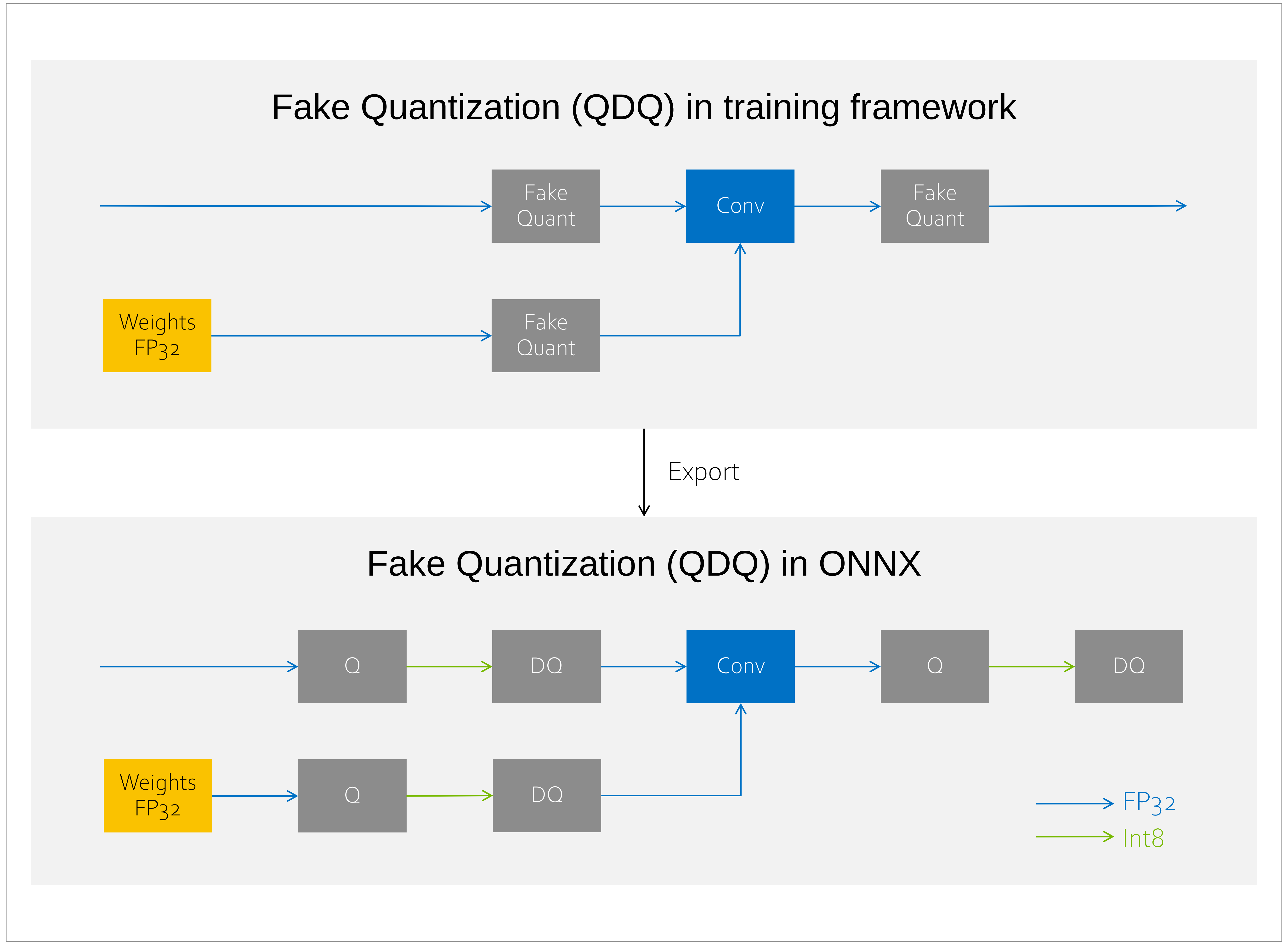
- Generated as part of training or via post training optimization
- Sparsity in inference can improve performance and reduce size of weights
- Structured — The number of weights that are zero is same every N values
  - N:M — N zeros every M elements
  - Block — Various blocks are sparse
- Unstructured - Percentage of memory that is 0
  - 10% — 1 out of 10 elements over entire memory block is a zero



Quantization Methods



Post Training Qauntization

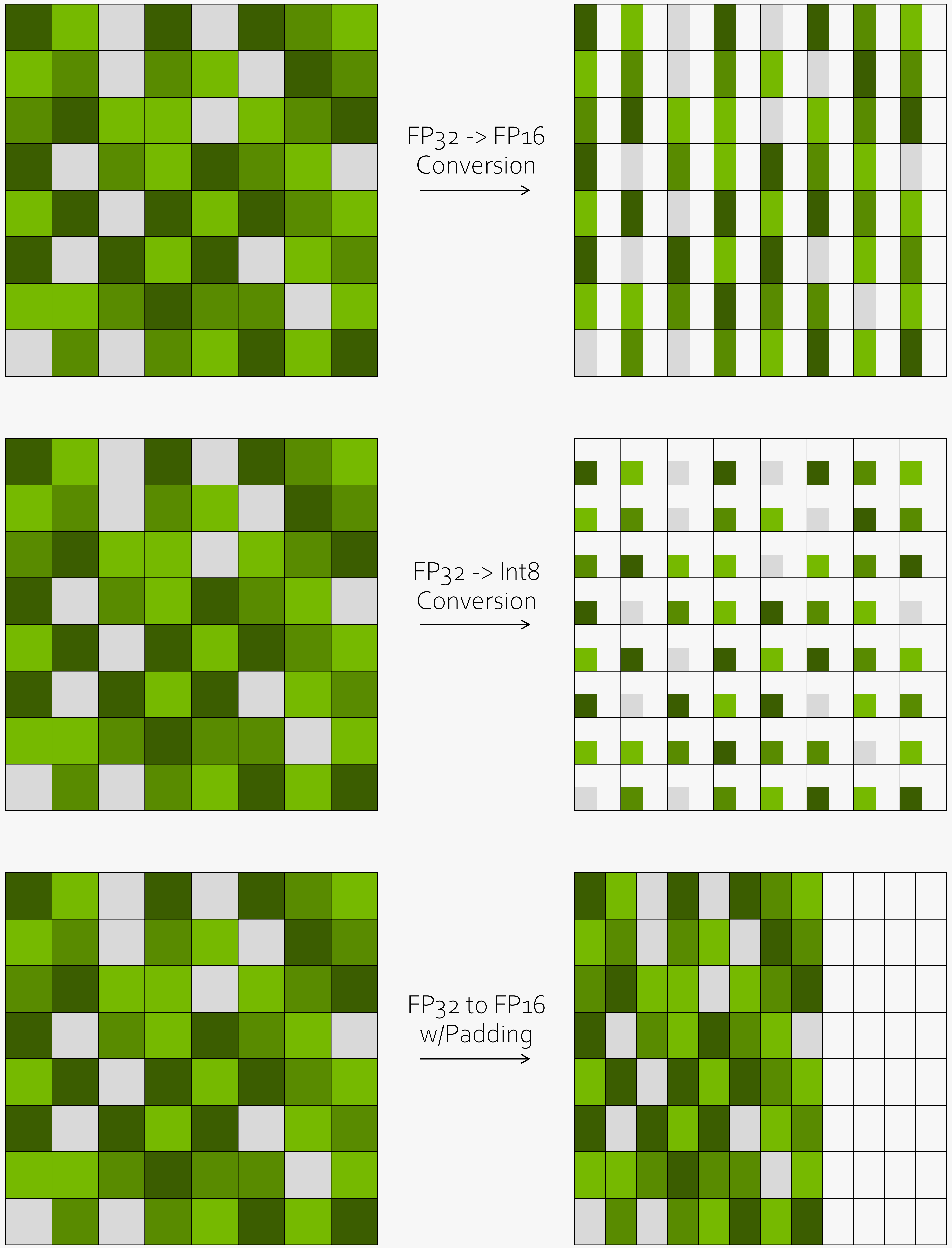


Quantization Aware Training

Weight Optimizations

Transforming weights to different types

- Conversion to different data types lowers memory requirements
- Conversion to different formats allows efficient algorithms



FP32 -> FP16 Conversion

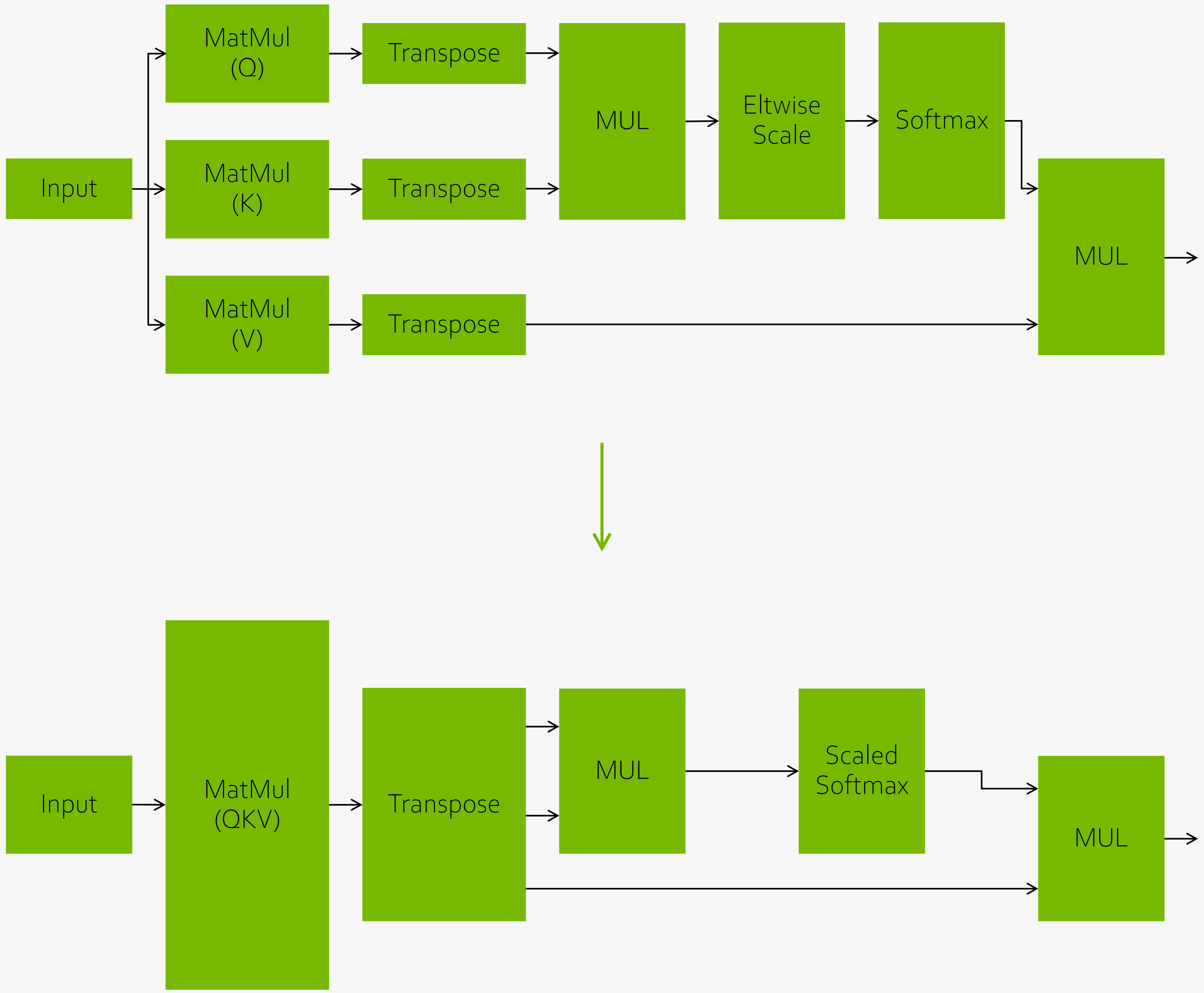
FP32 -> Int8 Conversion

FP32 to FP16 w/Padding

Layer Fusion

Optimizes use of GPU memory and bandwidth by fusing nodes in a kernel

- Combines nodes into a single node, making single kernel execution.
- Significantly reduces number of layers to compute, resulting in faster performance.
- Eliminates unnecessary memory traffic by not spilling to memory.



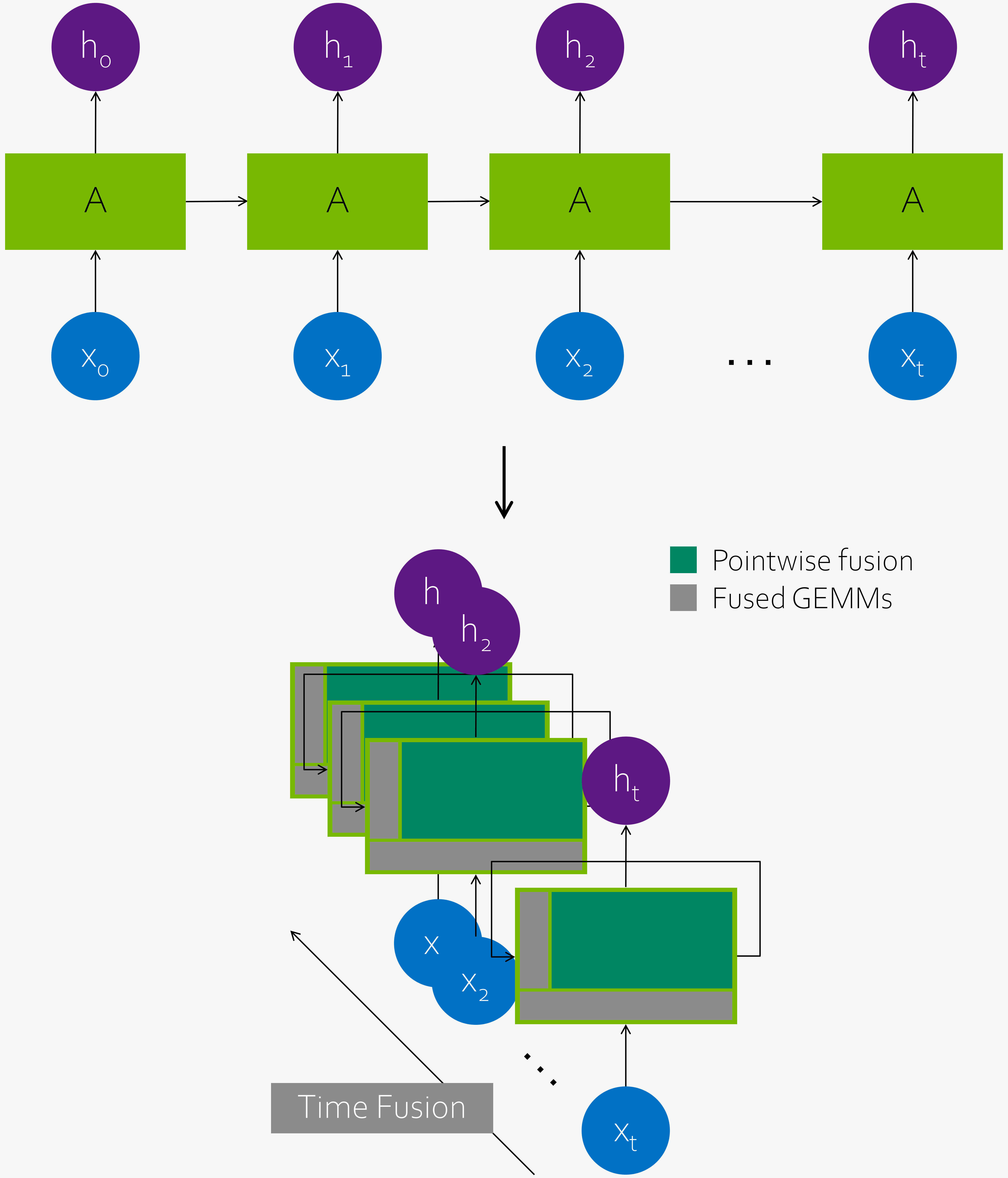
developer.nvidia.com/tensorrt

Time Fusion

Optimizes recurrent neural networks over time steps with dynamically generated kernels

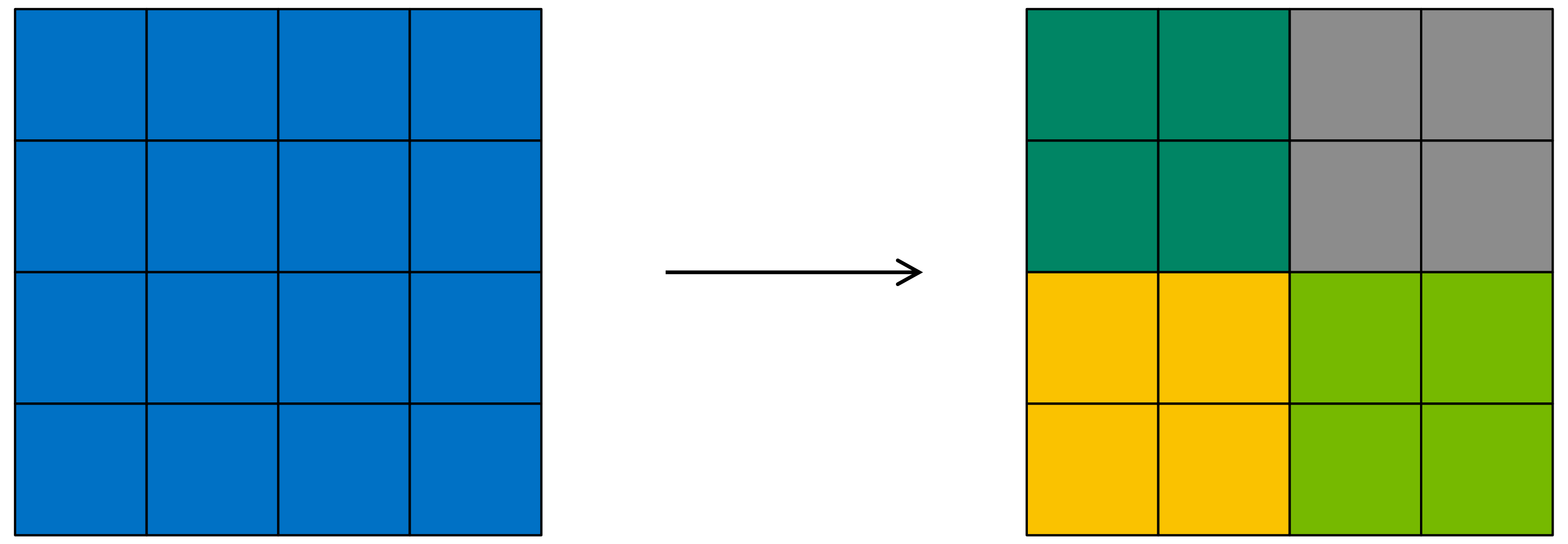
- Recurrent neural network optimizations
- Deploy highly optimized ASR and TTS
- Compiler fuses pointwise ops, fuses GEMMs and compute efficiently across time steps

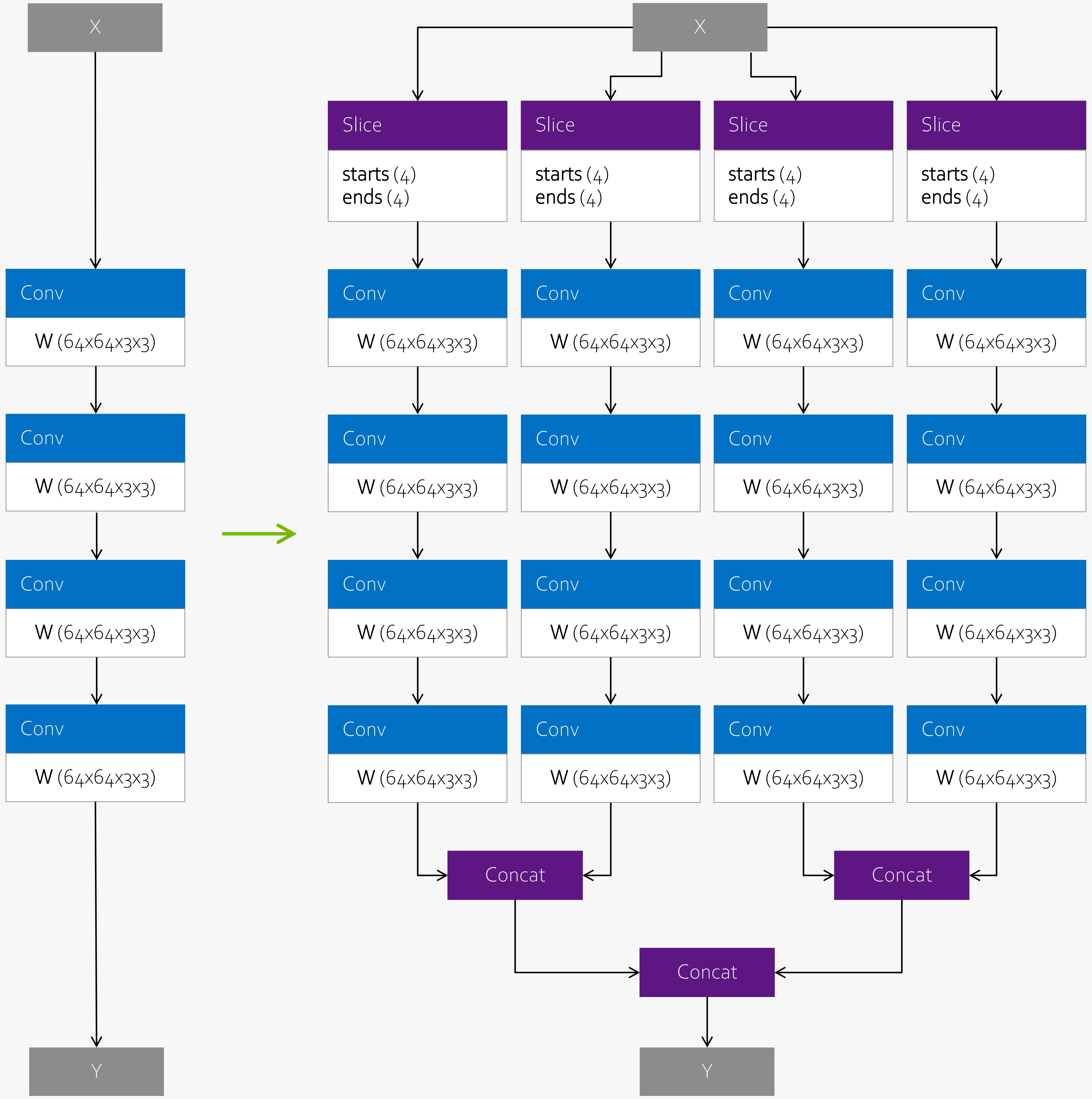
developer.nvidia.com/tensorrt



Memory — Tiling

- Segment a graph to get better computation locality
- Fits more of the graph computation into L1/L2



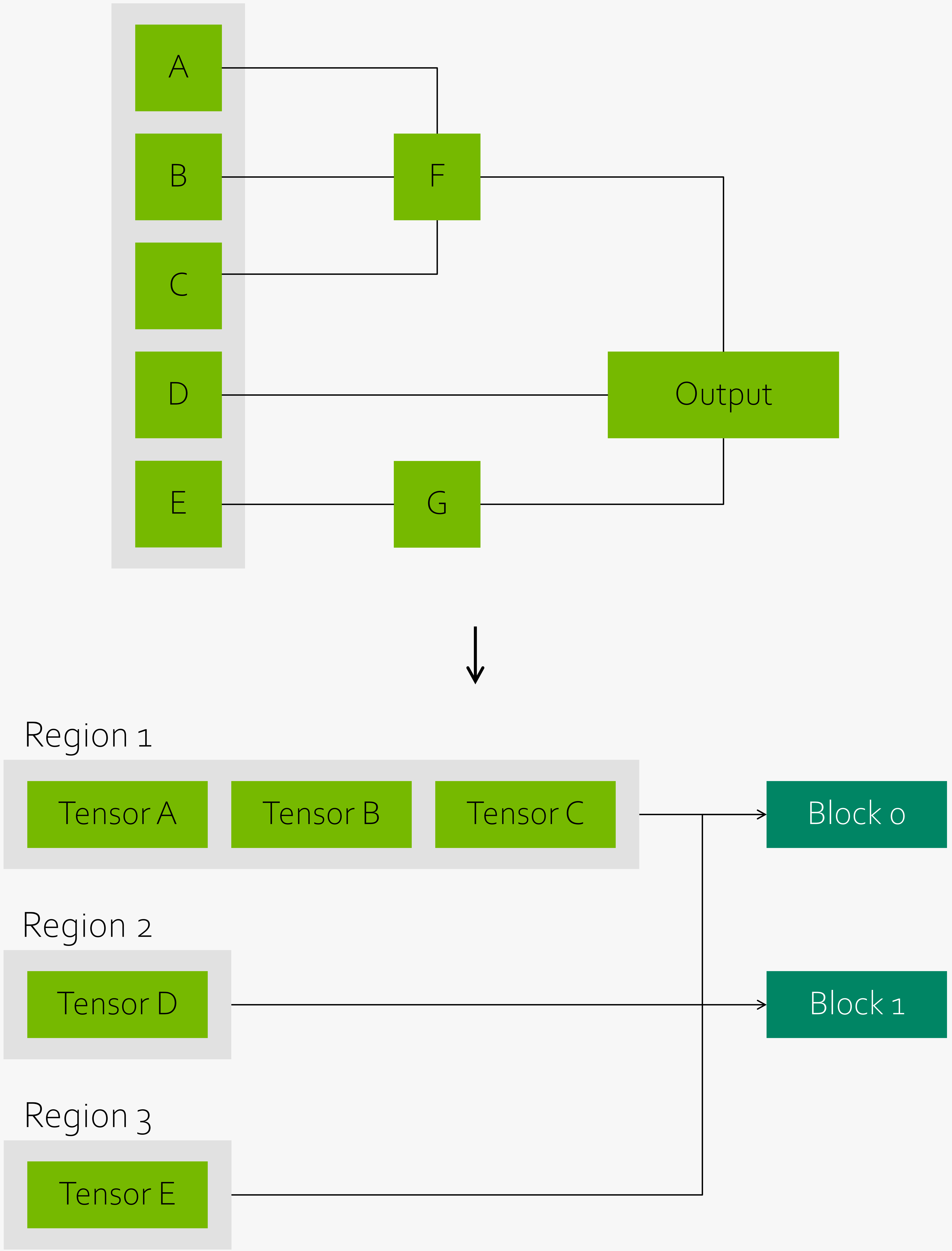


Memory Scheduling

Minimizes memory footprint and reuses memory for tensors efficiently

- Reduces memory footprint and improves memory re-use
- Two tensors with disjoint lifetimes can share the same memory
- Becomes an instance of the “dynamic storage allocation problem”
- Similar to traditional register allocation

developer.nvidia.com/tensorrt

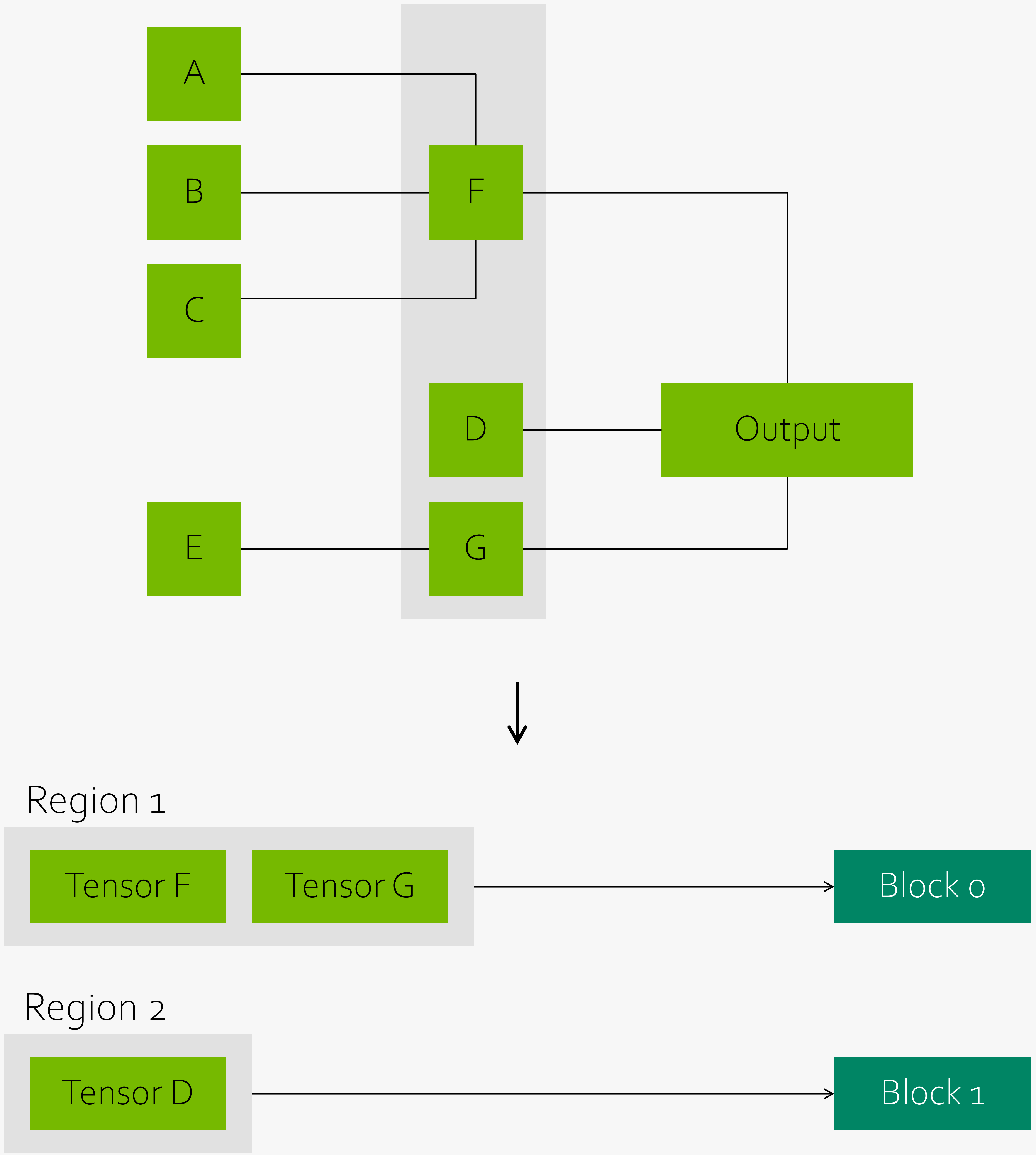


Memory Scheduling

Minimizes memory footprint and reuses memory for tensors efficiently

- Reduces memory footprint and improves memory re-use
- Two tensors with disjoint lifetimes can share the same memory
- Becomes an instance of the “dynamic storage allocation problem”
- Similar to traditional register allocation

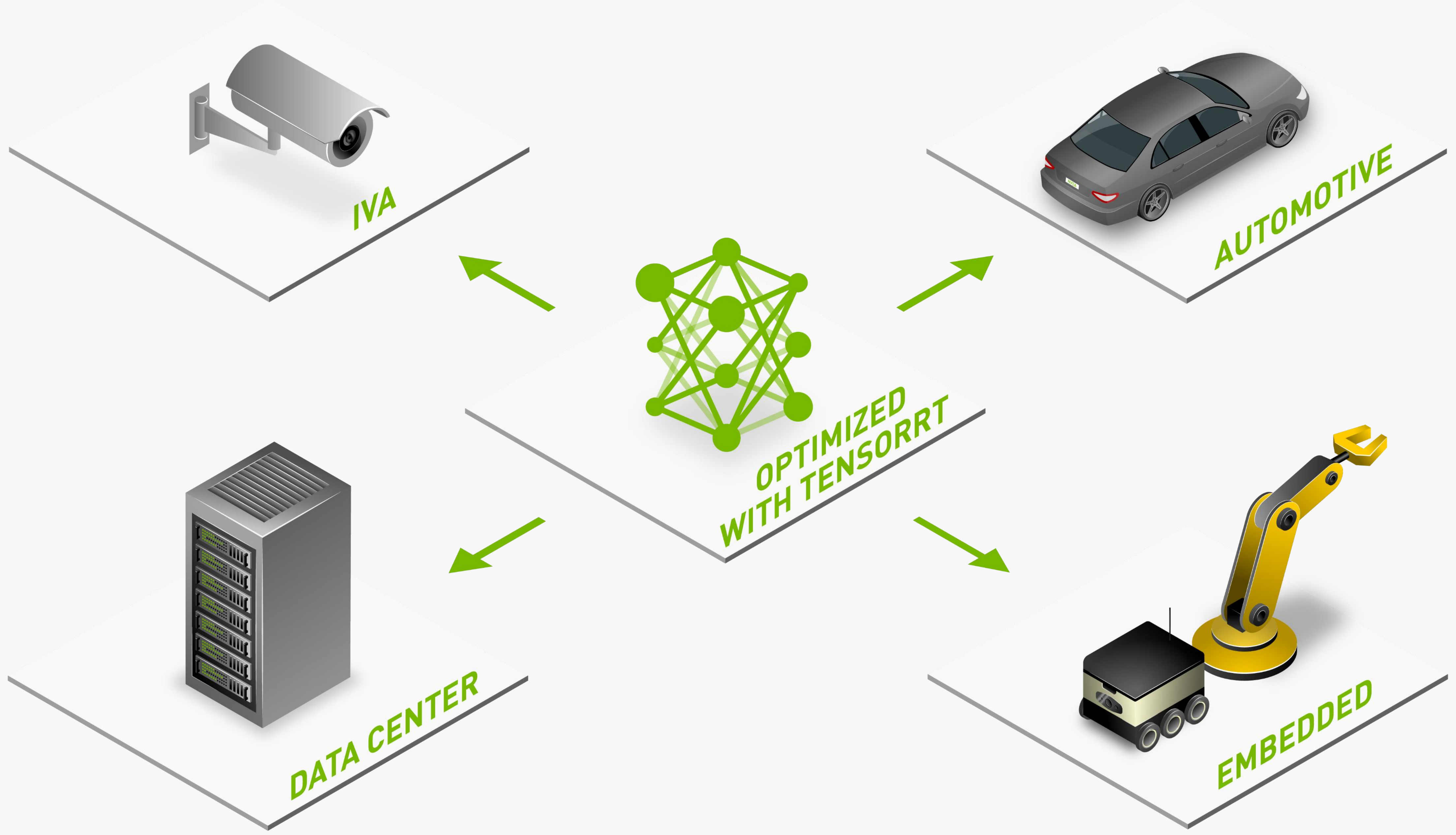
developer.nvidia.com/tensorrt



Kernel Selection

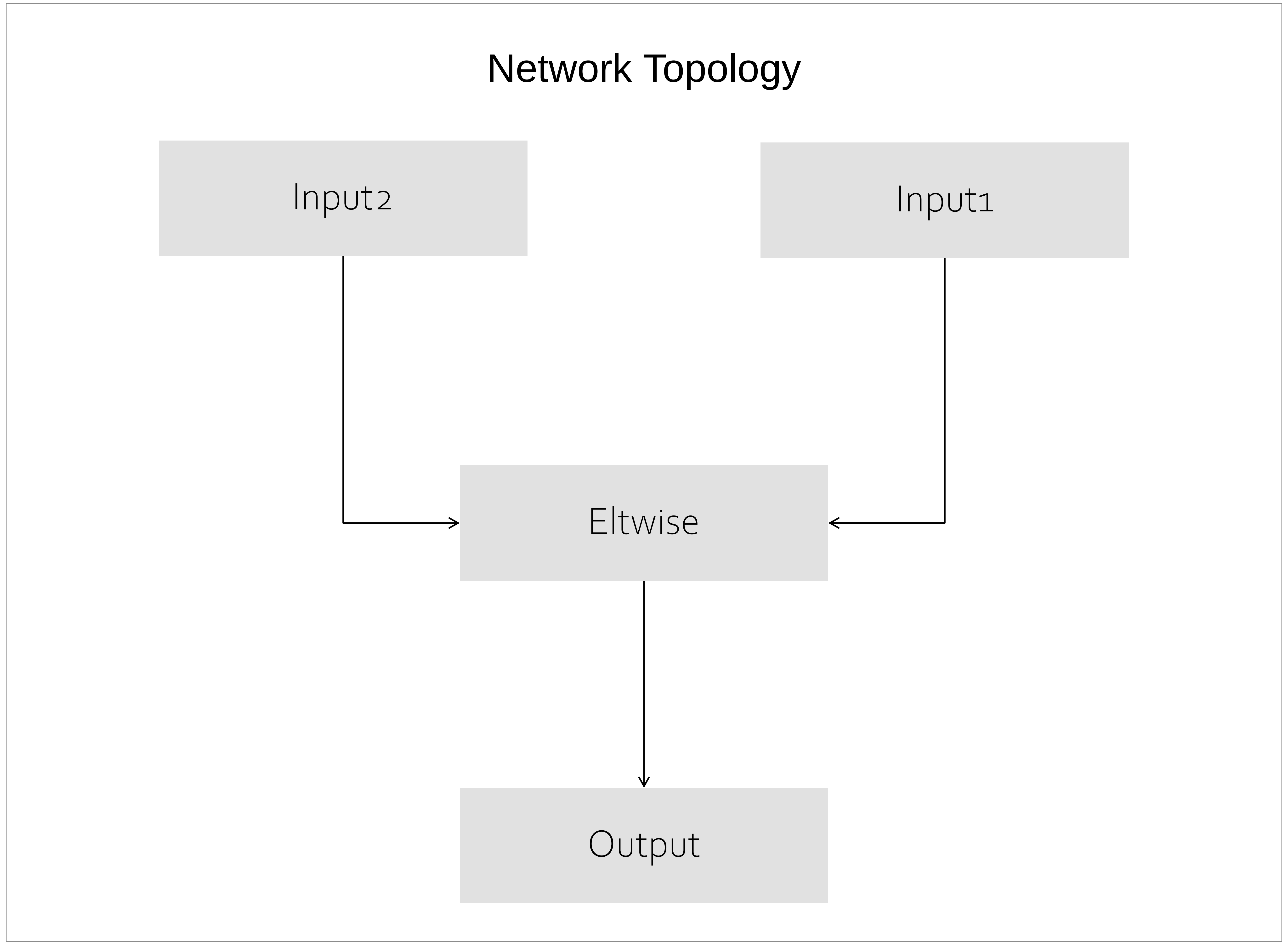
Selects best data layers and algorithms based on the target platform

- Specialized kernels optimized for every operation
- Combination of static and dynamically generated kernels
- Kernel selection uses timing information to choose combination of formats, precisions, and implementations that minimizes a network property
- Strives for best performance for specific deployment platform and specific neural network



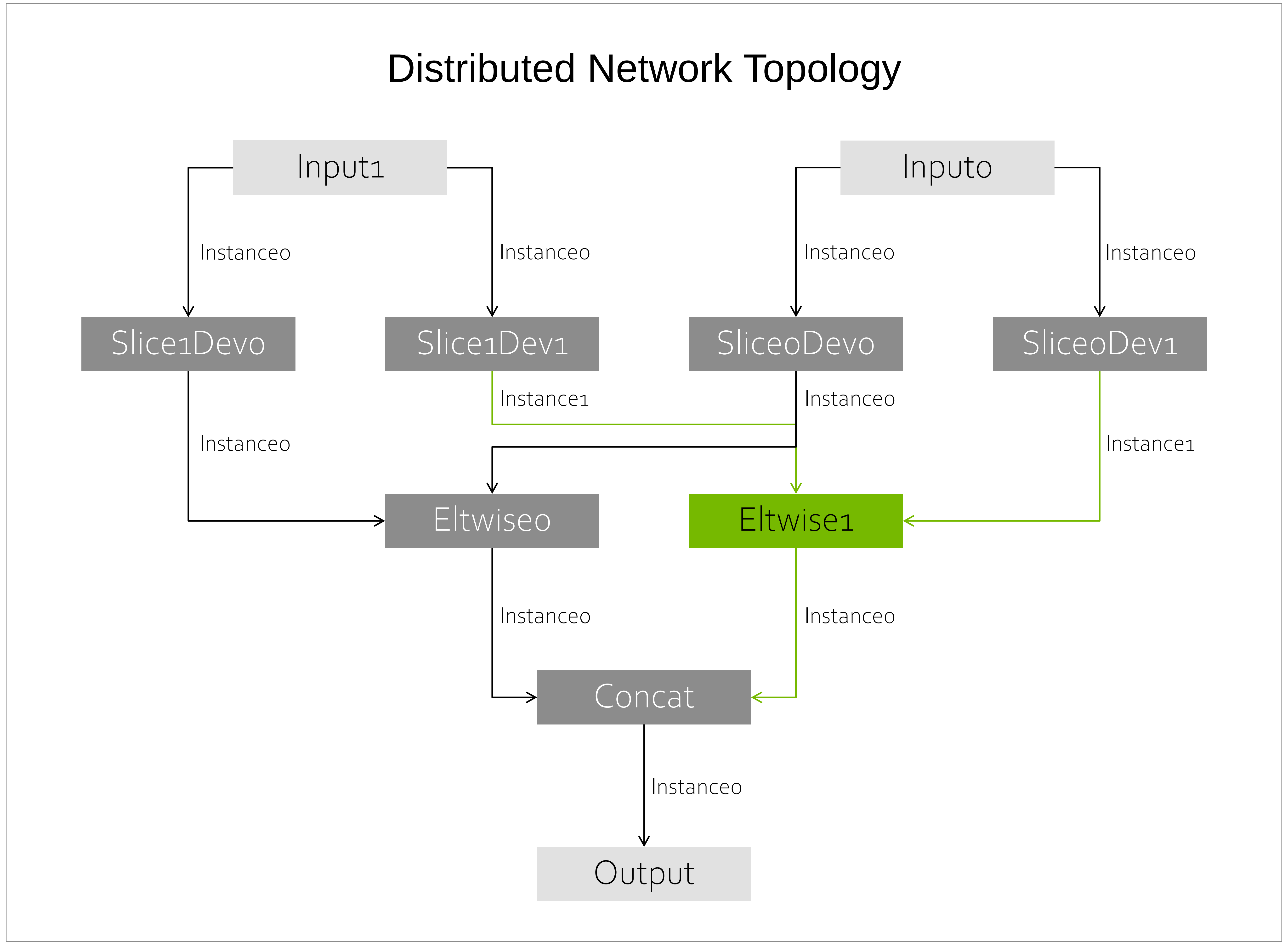
Multi-Device Segmentation

How to run large language models?



Network Topology

Single Device Model



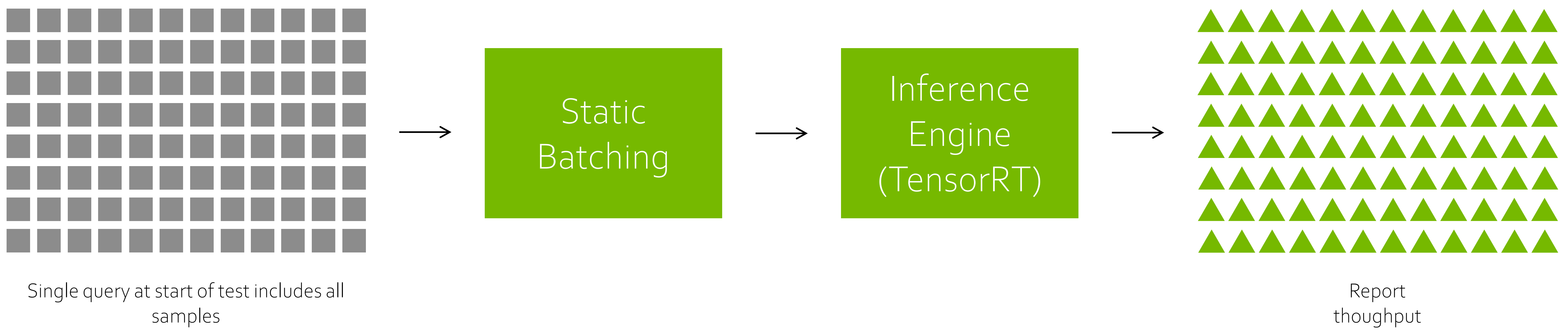
Distributed Network Topology

Multi Device Model

Execution

Execution Modes

Offline



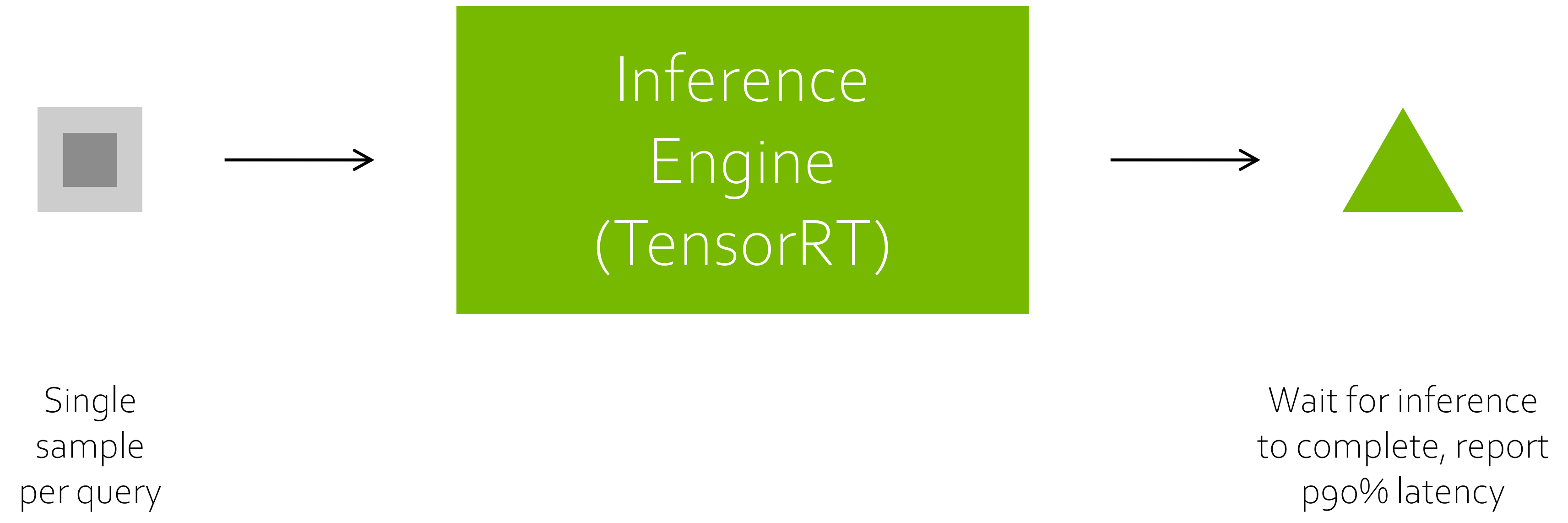
Static Batching

Inference Engine (TensorRT)

Single query at start of test includes all samples

Report thoughput

Single Stream

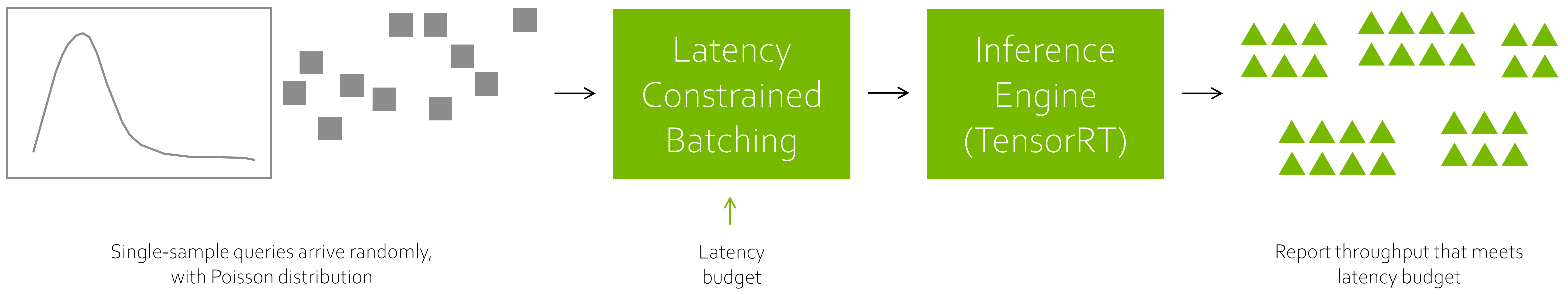


Inference Engine (TensorRT)

Single sample per query

Wait for inference to complete, report p90% latency

Server



Latency Constrained Batching

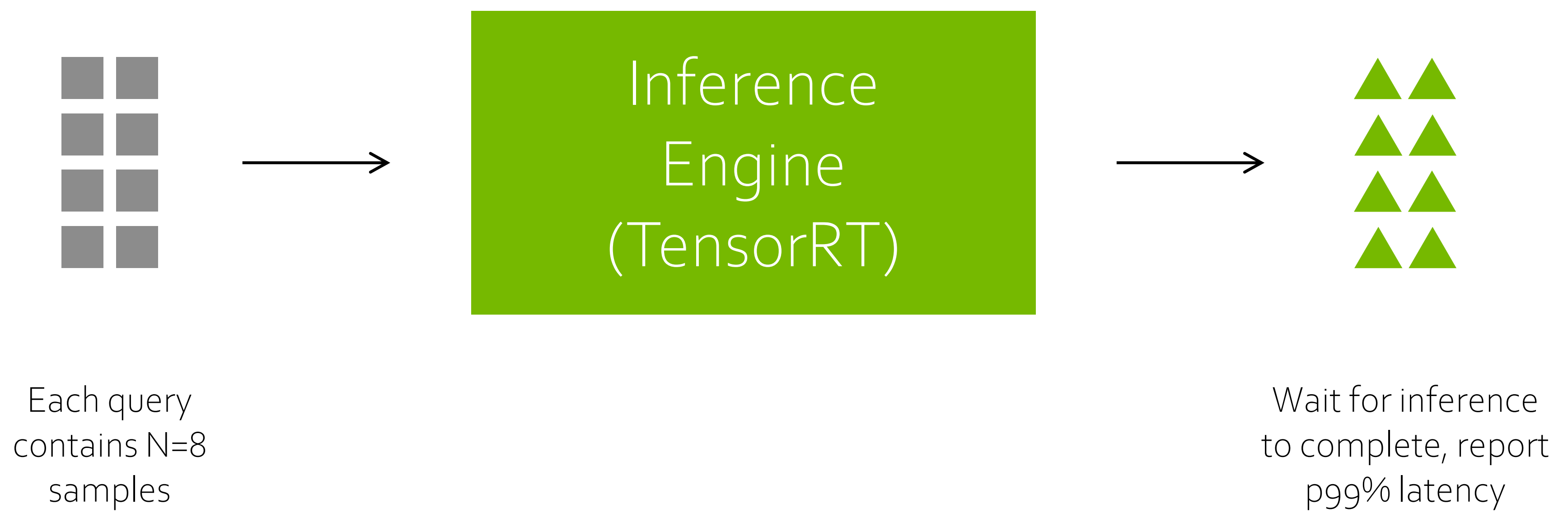
Inference Engine (TensorRT)

Single-sample queries arrive randomly, with Poisson distribution

Latency budget

Report throughput that meets latency budget

Multi Stream



Inference Engine (TensorRT)

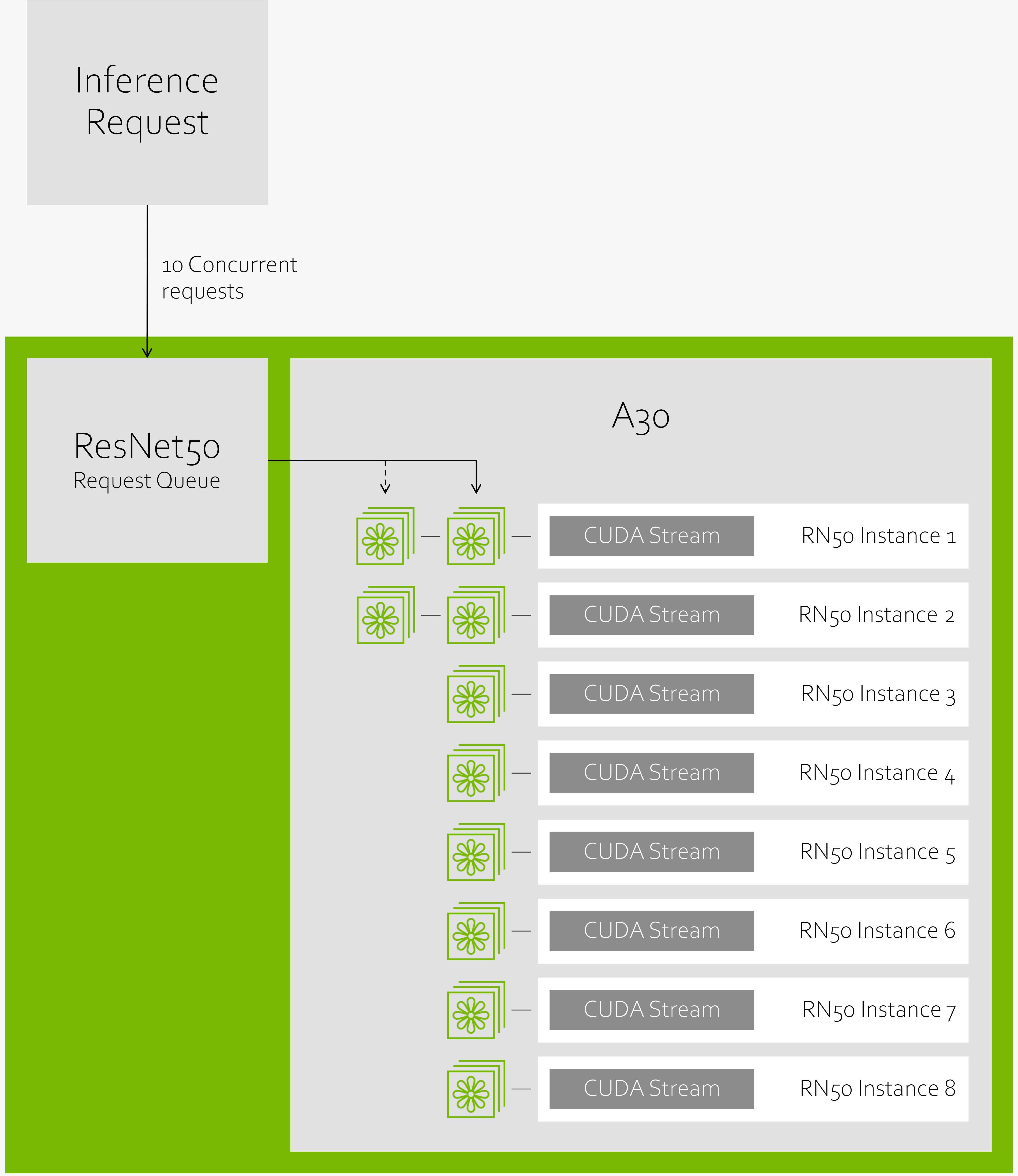
Each query contains N=8 samples

Wait for inference to complete, report p99% latency

Multi-Stream Concurrent Execution

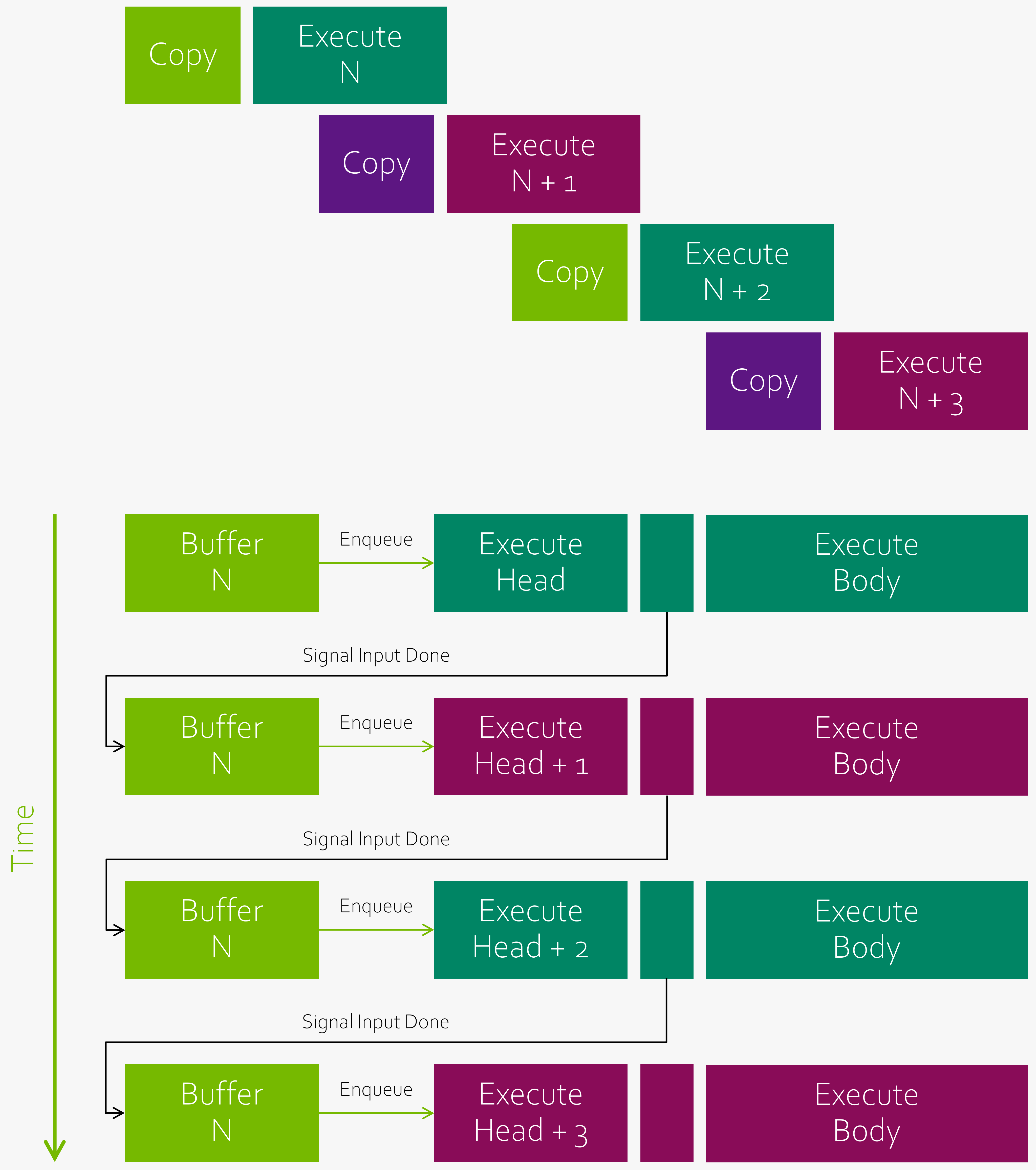
Uses a scalable design to process multiple input streams in parallel

- Better performance and improved utilization through multi-stream concurrent execution



Activations — Async Buffering

- Relies on multiple buffers to pipeline execution
- Requires notification of when a buffer is ready
- Allows pipelining of execution and copying inputs for next iteration



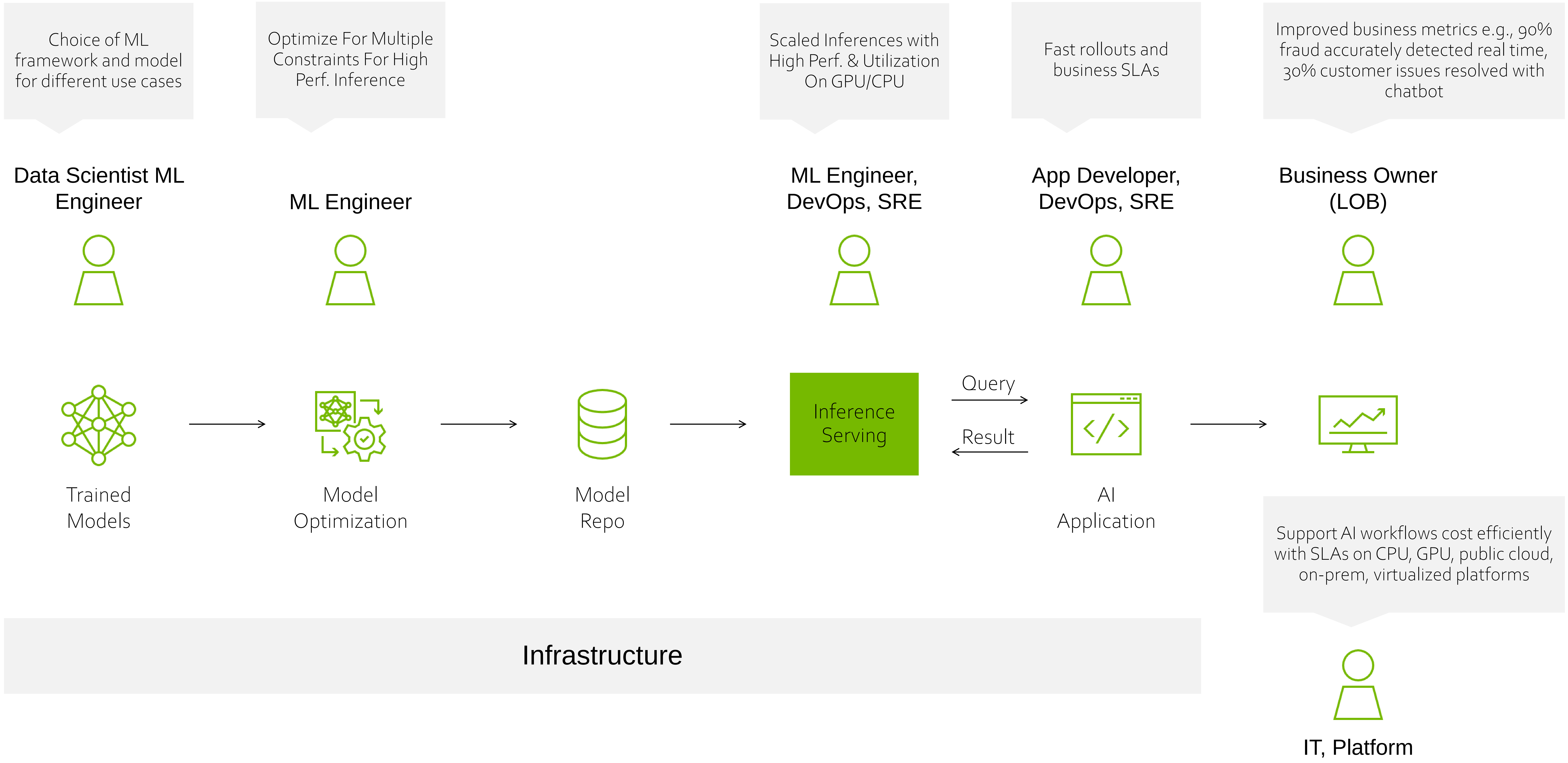
Delivering High Performance Across Frameworks

NVIDIA Triton's architecture

Model Analyzer
Model Orchestration
Multiple Client Applications
Python/C++ Client Library
Query
Result
Python/C++ Client Library
Query
Result
Python/C++ Client Library
Query
Result
Standard HTTP/gRPC
Or
In-Process API
(directly integrate into client app via C or Java API)
Dynamic Batching (Real time, Batch, Stream)
Multiple GPU & CPU Backends
Custom
ONNX
OpenVINO
PyTorch
TensorFlow
TensorRT
Per Model Scheduler Queues
...
Flexible Model Loading (All, Selective)
Many active models
Model Repository
Utilization, Throughput, Latency Metrics
Metrics
Kubernetes, Prometheus
GPU
CPU

AI Inference Workflow

Collaboration between multiple teams



Choice of ML framework and model for different use cases

Data Scientist ML Engineer

Optimize For Multiple Constraints For High Perf. Inference

ML Engineer

Scaled Inferences with High Perf. & Utilization On GPU/CPU

ML Engineer, DevOps, SRE

Fast rollouts and business SLAs

App Developer, DevOps, SRE

Improved business metrics e.g., 90% fraud accurately detected real time, 30% customer issues resolved with chatbot

Business Owner (LOB)

Trained Models

Model Optimization

Model Repo

Inference Serving

Query

Result

AI Application

Support AI workflows cost efficiently with SLAs on CPU, GPU, public cloud, on-prem, virtualized platforms

IT, Platform

Infrastructure

NVIDIA