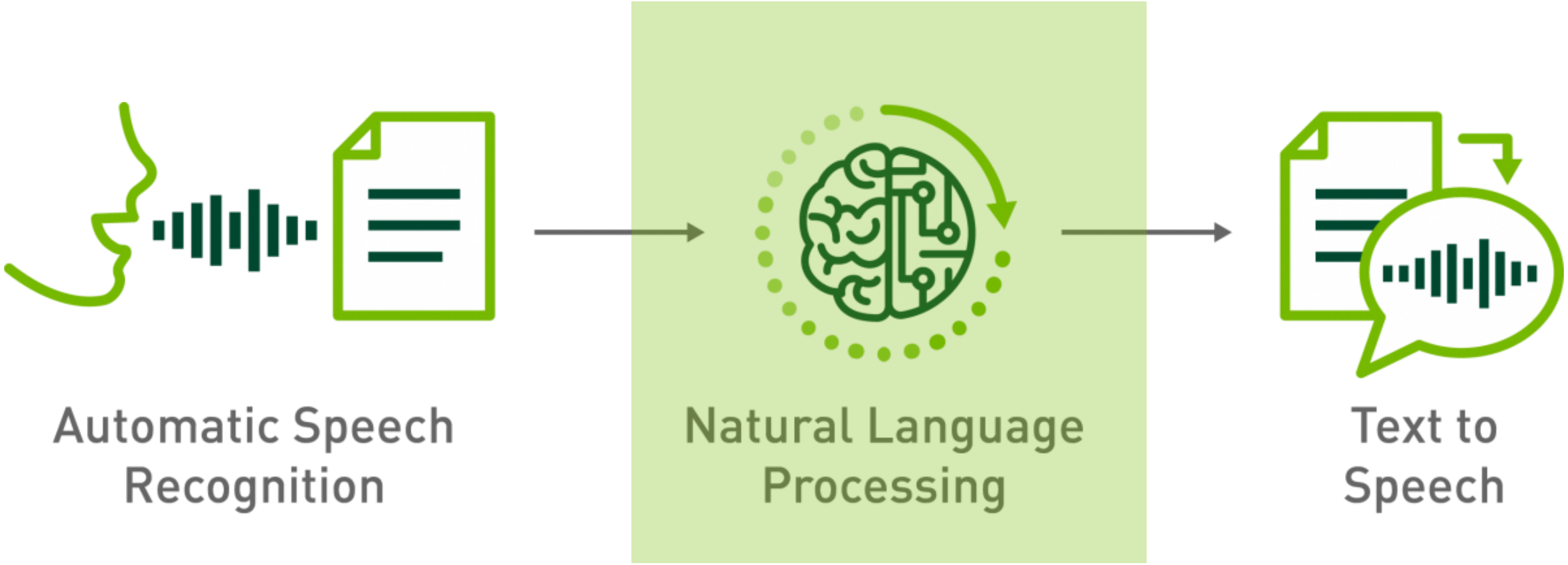
CHALLENGES OF BUILDING TRANSFORMERS BASED NATURAL LANGUAGE PROCESSING MODELS

Meriem Bendris, Solution Architect, NVIDIA

NATURAL LANGUAGE PROCESSING

The brain of Conversational AI



NATURAL LANGUAGE PROCESSING

- Machine Translation
- Question Answering
- Spell Checker Grammar Correction
- Information Retrieval
- Named Entity Recognition
- Characters Recognition
- Sentiment Analysis
- Topic Classification
- Automatic Speech Recognition
- Text summarization
- SPAM Filtering
- Virtual Assistant

LARGE LANGUAGE MODELS

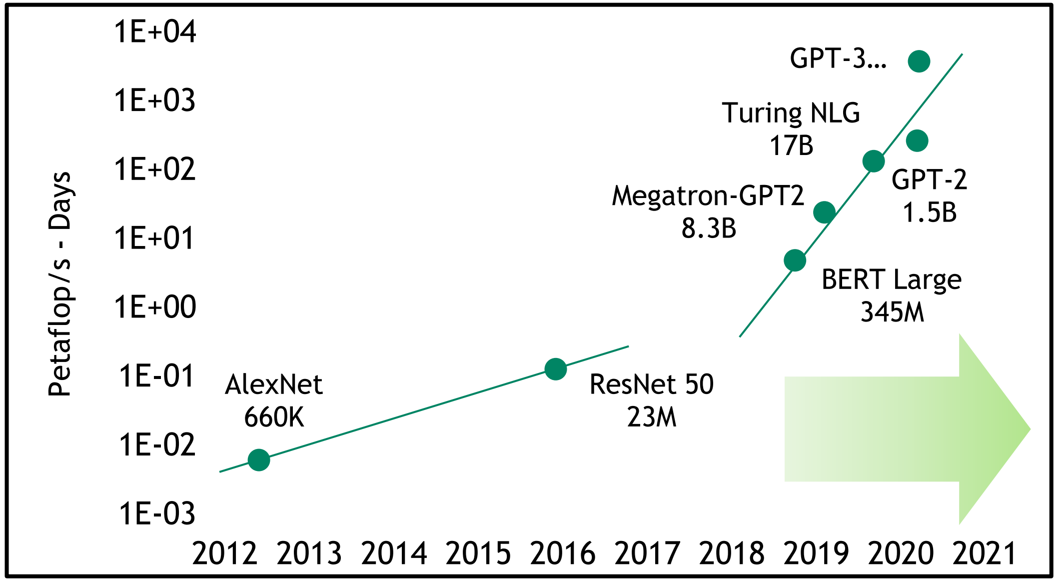
Exploding model complexity

Exploding model complexity:

- Large models, large datasets and large compute
- OpenAI GPT-3 175B* parameters:
  - **Compute: 4.5** ExaFLOPs / iteration (~95K iterations)
  - **Memory using FP16: 2.8TB** (700GB parameters + 700GB Gradients + 1400GB Optimizer state)

Challenges:

- Ability to efficiently collect and process large volumes of data
- Ability to efficiently train large models on large volumes of data
- Ability to cost effectively deploy large models



* Total parameters: 175B. Number of layers: 96, Batch size: 1536, Sequence length: 2048, Hidden layer size: 12288, Vocabulary size: 51200

AGENDA

Overview of Modern NLP Model

What are transformers, Success Reasons

Models Training

What are the challenges of building modern NLP models?

Distributed Training, Memory usage reduction

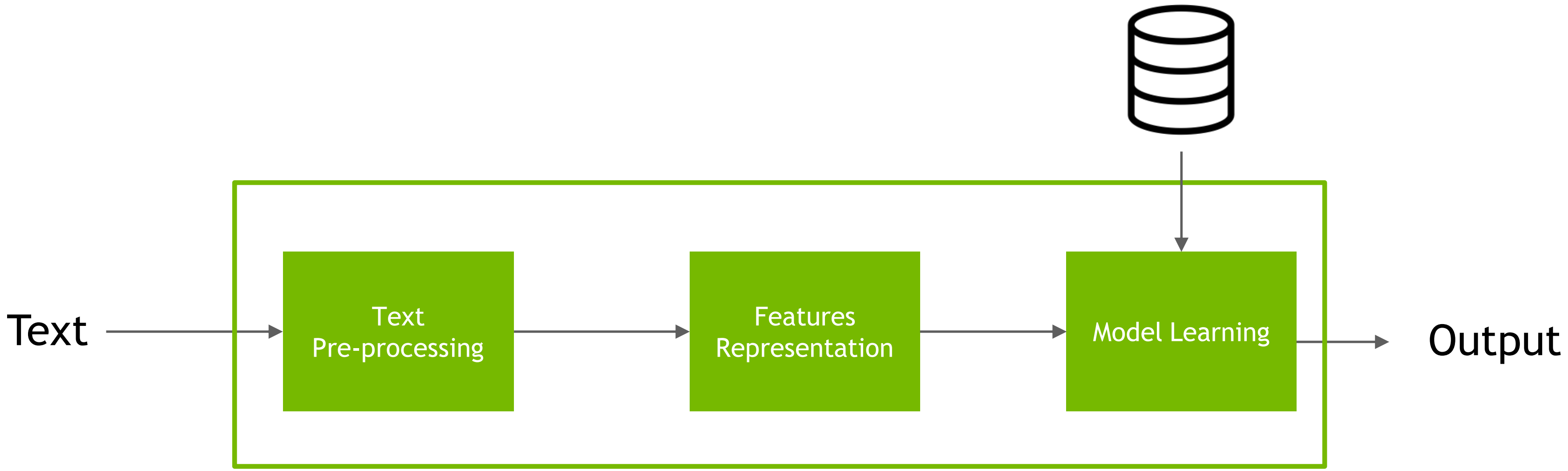
Model Deployment

What are the challenges of deploying NLP models?

Model Optimization and efficient Model Serving

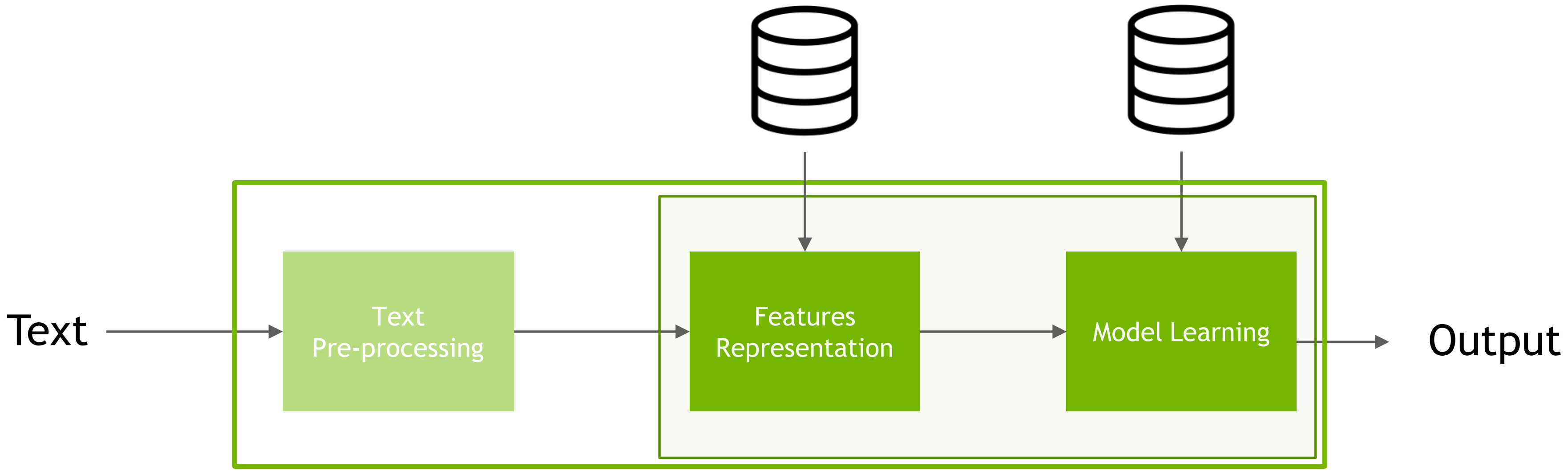
NATURAL LANGUAGE PROCESSING

Machine Learning



NATURAL LANGUAGE PROCESSING

Deep Learning Promise



NATURAL LANGUAGE PROCESSING

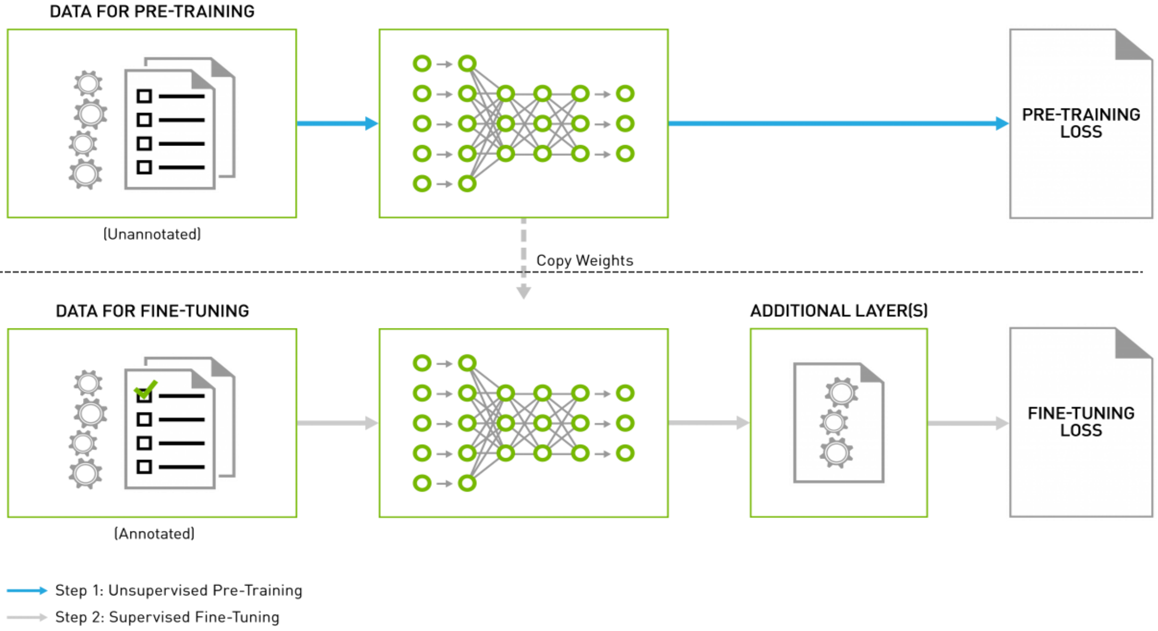
From Language Models to NLP downstream tasks

Language Model Pretraining:

- **Large unlabelled dataset**
- **Self-Supervised**

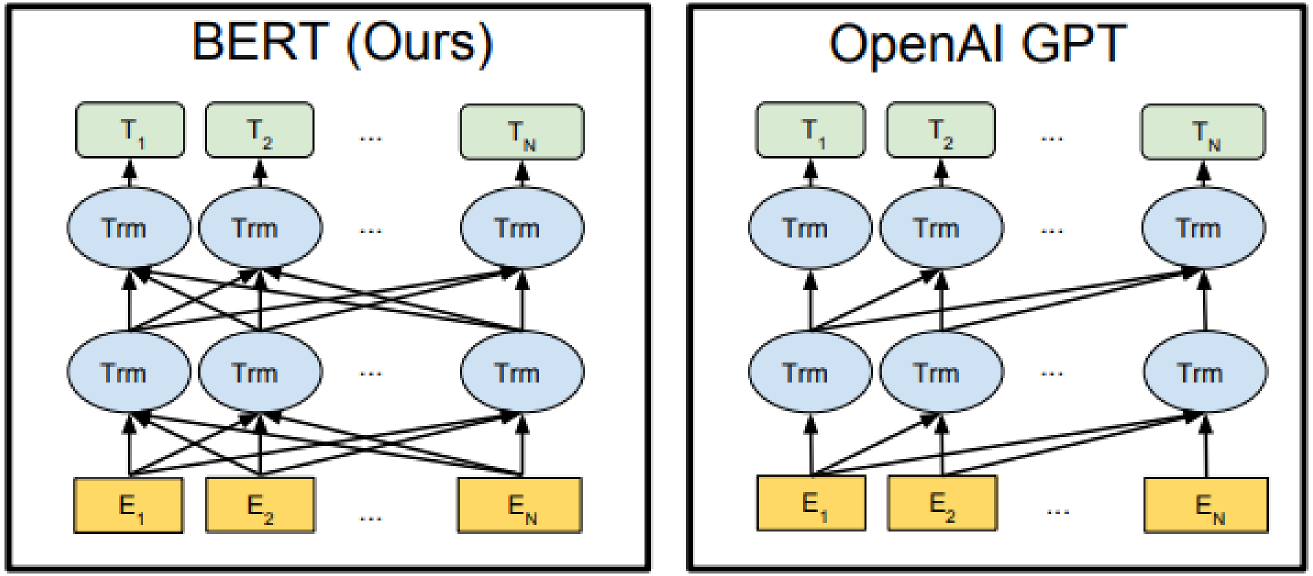
Finetuning on downstream NLP tasks:

- **Annotated dataset**
- **Additional layers**



NATURAL LANGUAGE PROCESSING

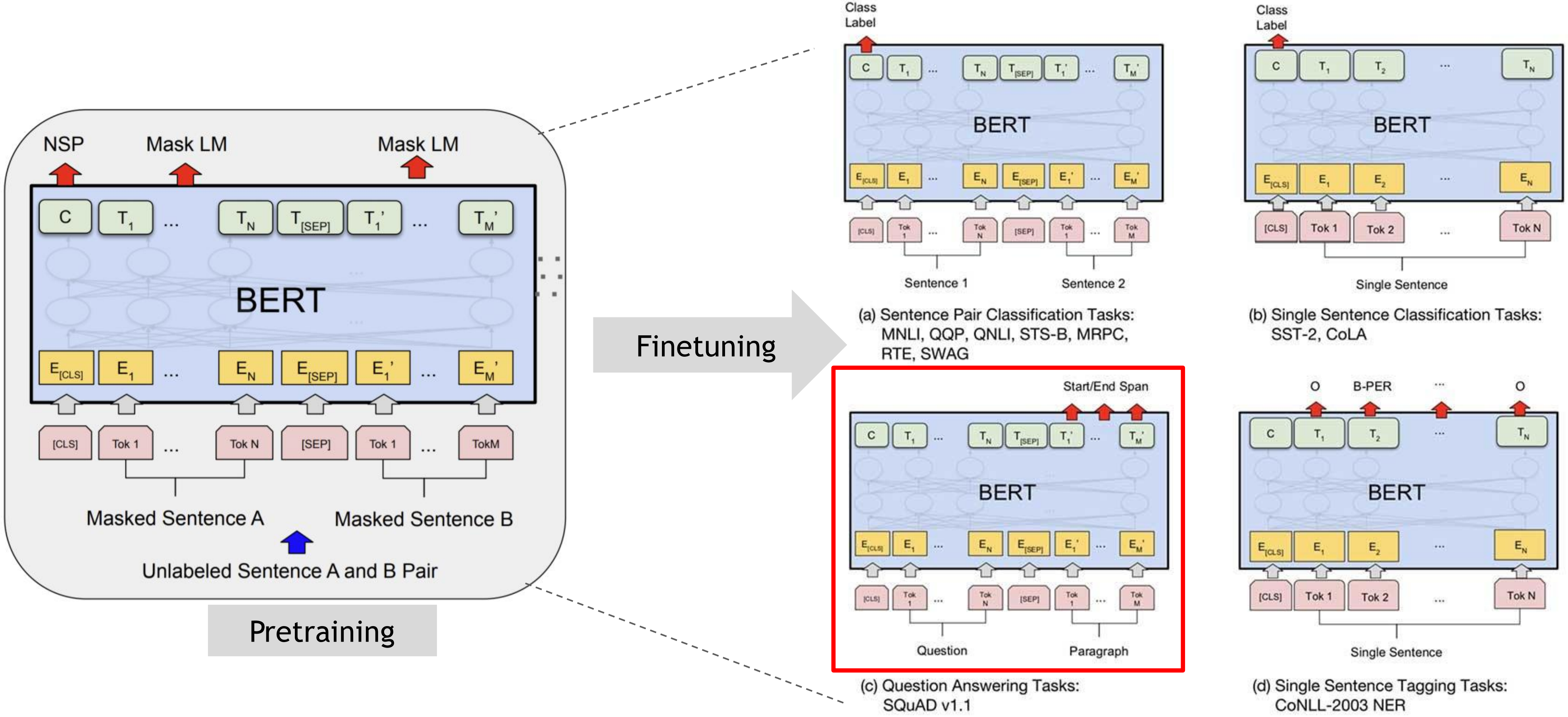
Language Models Examples



Jacob Devlin Ming-Wei Chang Kenton Lee Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. (2018) arxiv:1810.04805

NATURAL LANGUAGE PROCESSING

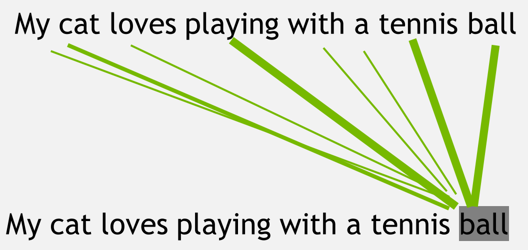
From Language Models to NLP downstream tasks with BERT



Jacob Devlin Ming-Wei Chang Kenton Lee Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. (2018) arxiv:1810.04805

NATURAL LANGUAGE PROCESSING

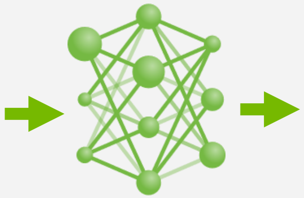
Success Reasons



Attention Mechanism

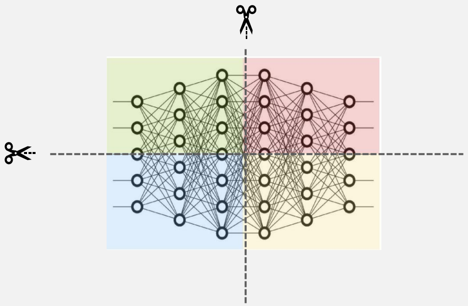
Large Datasets

[MASK]
The ~~cat~~ is playing in the garden



The cat is playing in the garden

Self-Supervised Learning

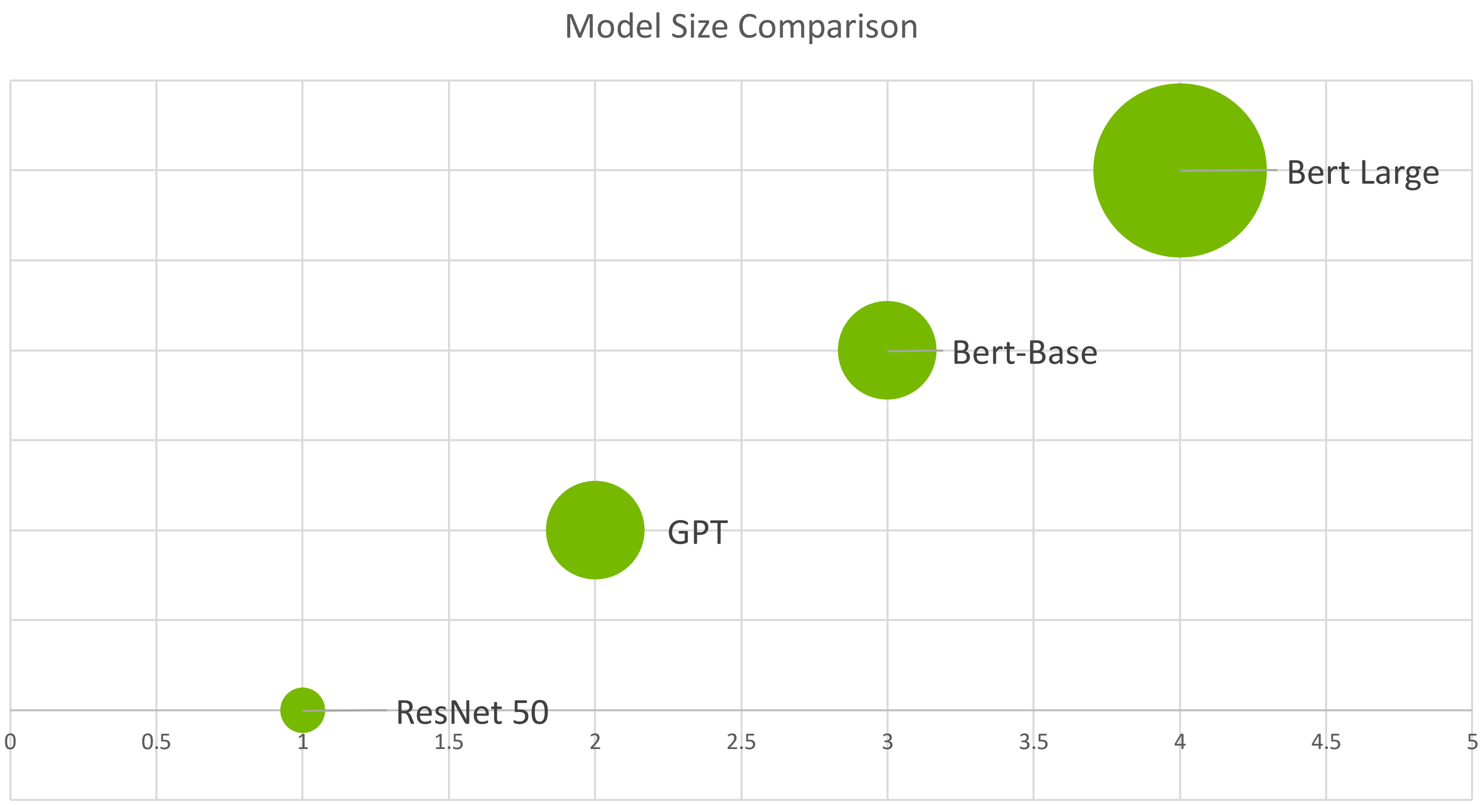


Distributed Training

Hardware Acceleration

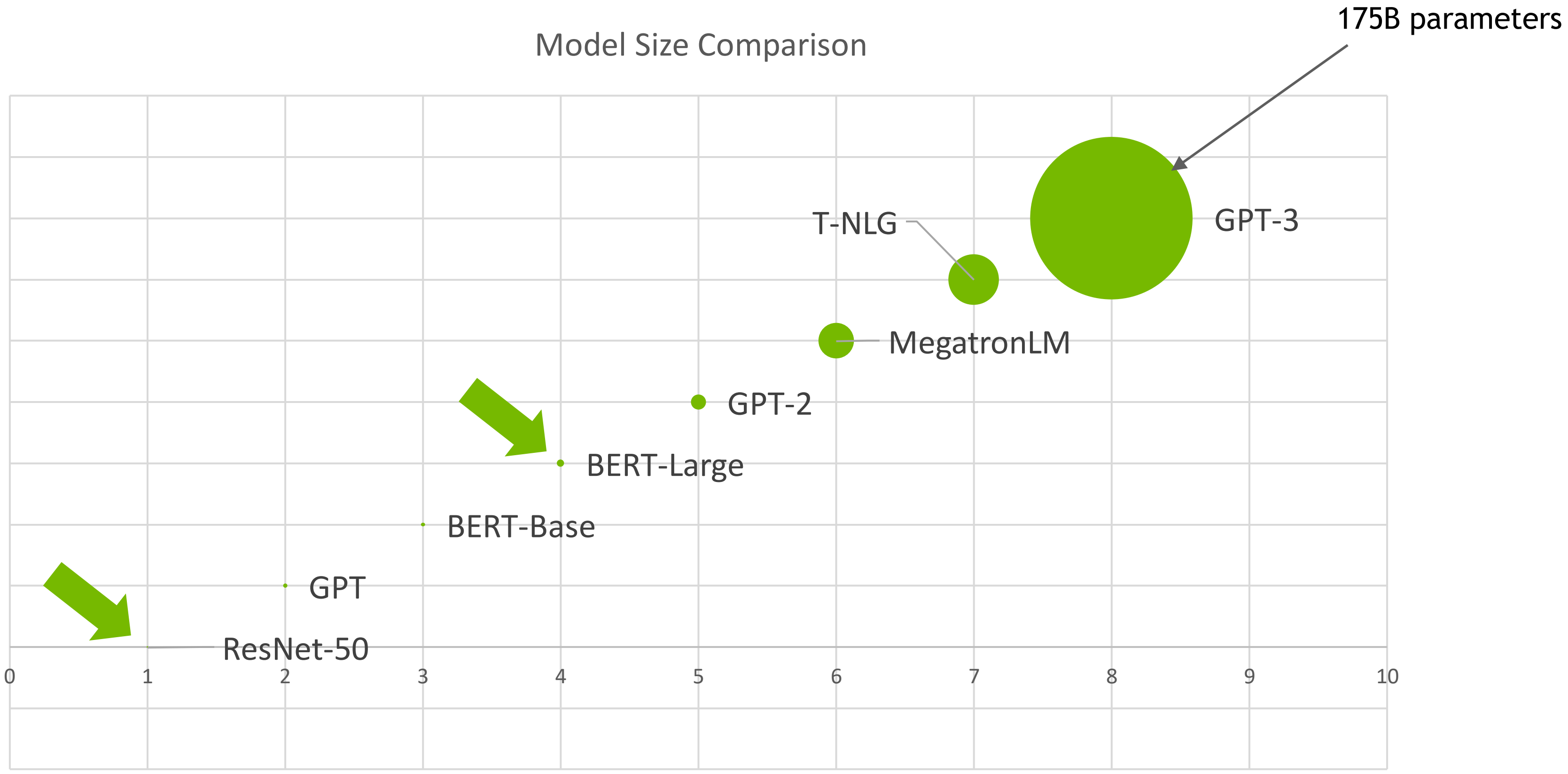
NATURAL LANGUAGE PROCESSING

Going Bigger



NATURAL LANGUAGE PROCESSING

Going Bigger



CHALLENGE OF GOING BIGGER

Large Neural Networks are big

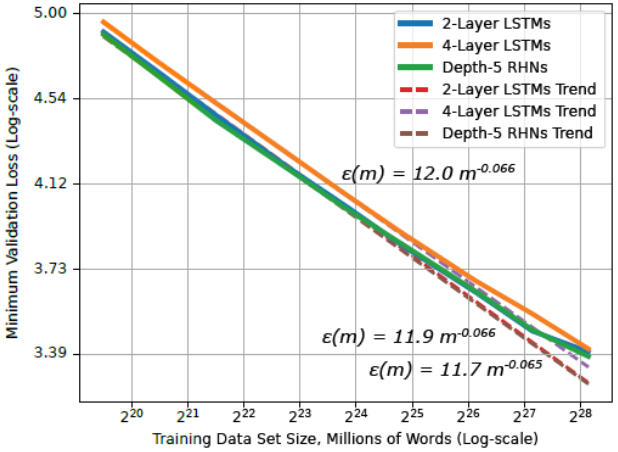
Consider 1 billion parameters model in **FP16** and do the math:

- Data representation: Weights and Gradients in FP16
- Adam optimizer: Store 12 bytes per weight in FP16

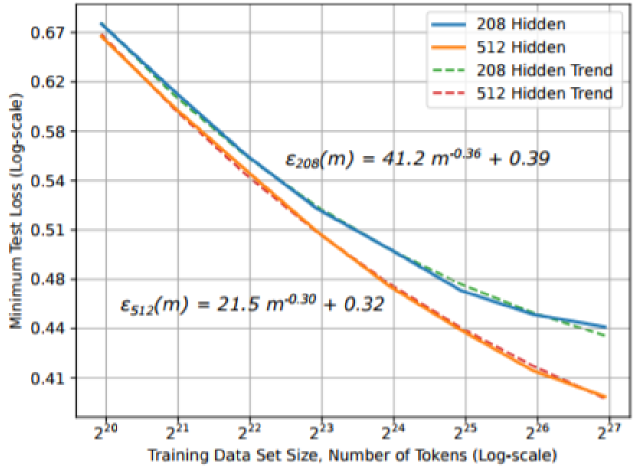
$$\underbrace{10^9}_{\text{1 billion parameters}} * \left(\underbrace{2B}_{\text{2 bytes per weight}} + \underbrace{2B}_{\text{2 bytes per gradient}} + \underbrace{12B}_{\text{12 bytes per optimizer state}} \right) = \mathbf{14.90GB}$$

CHALLENGE OF GOING BIGGER

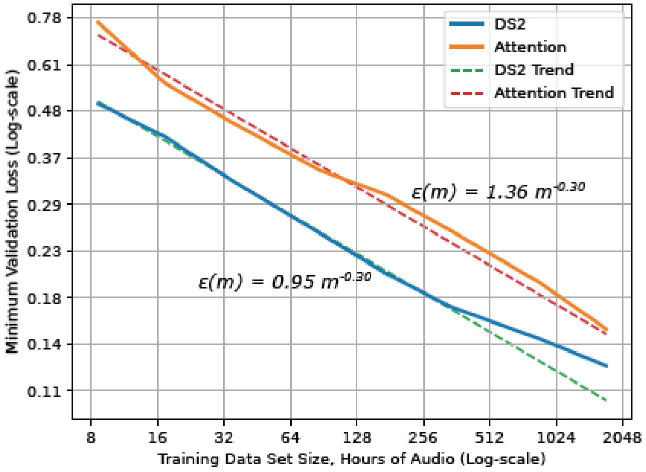
Large Neural Networks require Large Datasets



Word Language models



Machine Translation



Speech Models

Hestness, J., et al. Deep Learning Scaling is Predictable, Empirically.2017, arXiv preprint arXiv:1712.00409.

AGENDA

Overview of Modern NLP Model

What are transformers, Success Reasons

Models Training

What are the challenges of building modern NLP models?

Distributed Training, Memory usage reduction

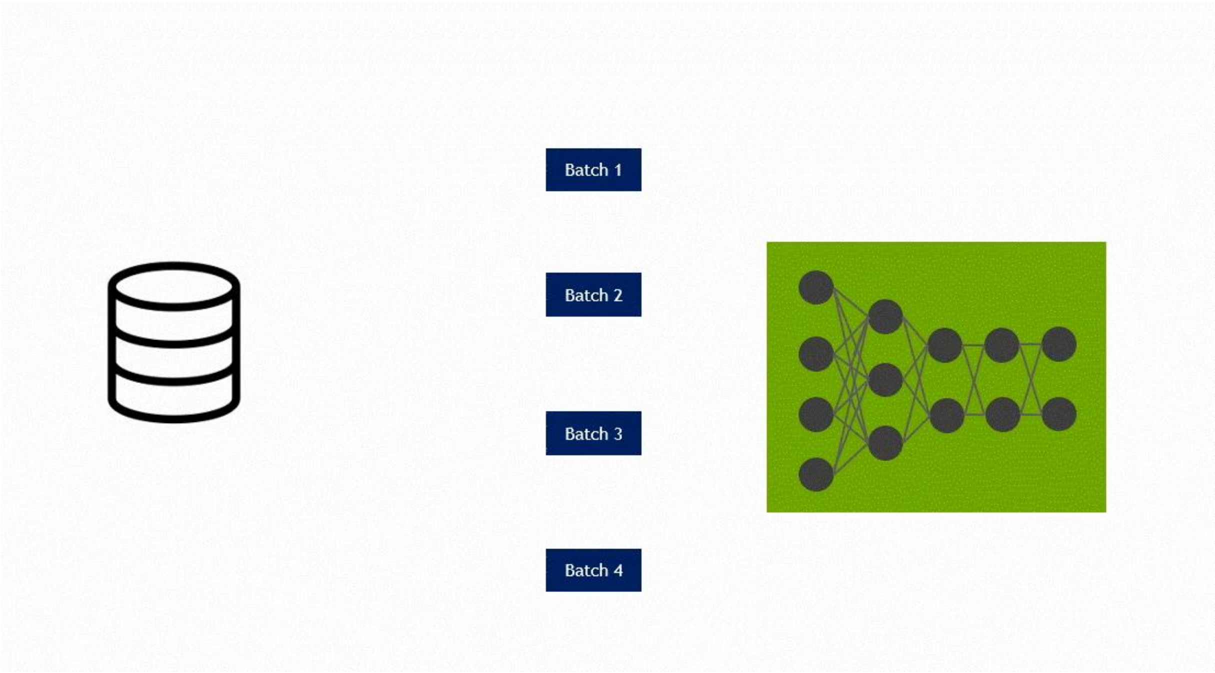
Model Deployment

What are the challenges of deploying NLP models?

Model Optimization and efficient Model Serving

DISTRIBUTED TRAINING ACROSS MACHINES

Training across 1-GPU



Stochastic Gradient Decent

Repeat n Epochs

Batch 1:
- *Forward, Backward*
- *Parameters update*

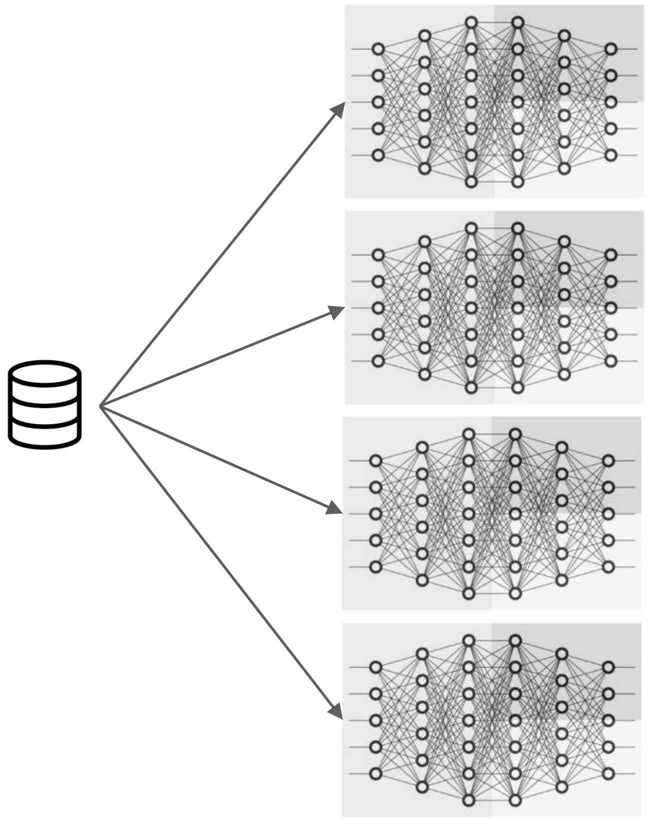
Batch 2:
- *Forward, Backward*
- *Parameters update*

…

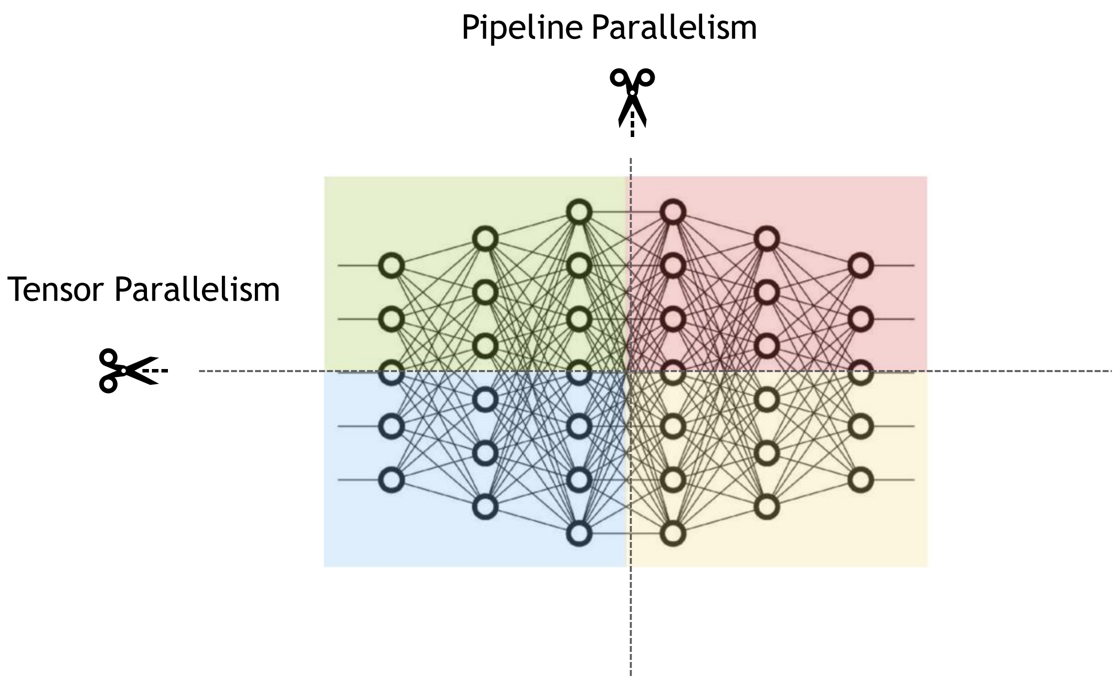
Batch k:
- *Forward, Backward*
- *Parameters update*

DISTRIBUTED TRAINING ACROSS MACHINES

Strategies



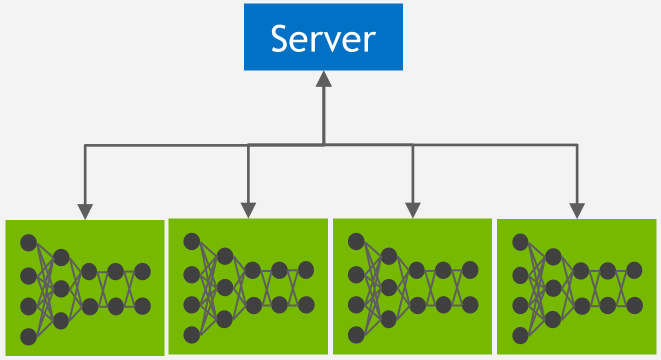
Advantage: Training Speed up
Cost: Gradients exchange



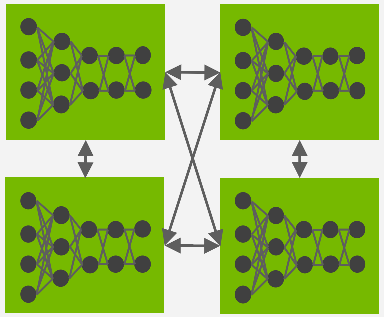
Advantage: Training Bigger Models
Cost: Features Maps exchange

DISTRIBUTED TRAINING ACROSS MACHINES

Data Parallel – Gradient Exchange Strategies



Centralized



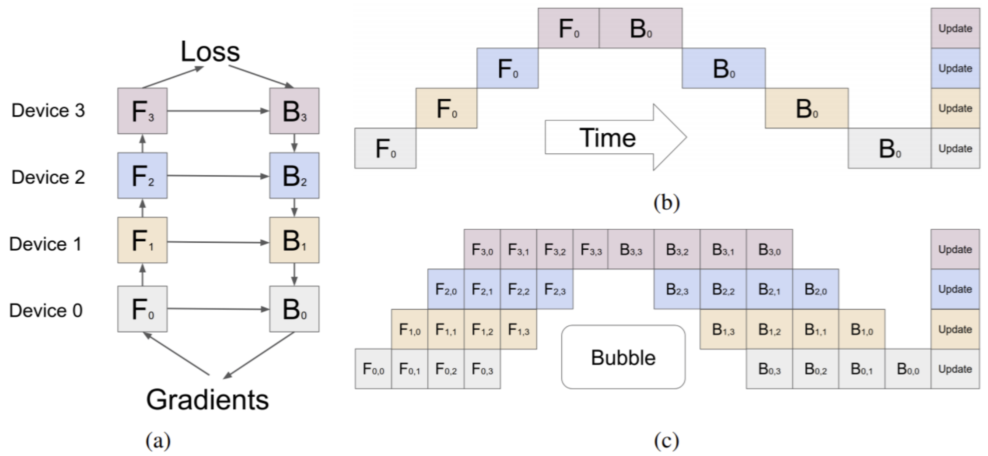
Decentralized

DISTRIBUTED TRAINING ACROSS MACHINES

Micro Batch Pipeline Parallelism

GPipe

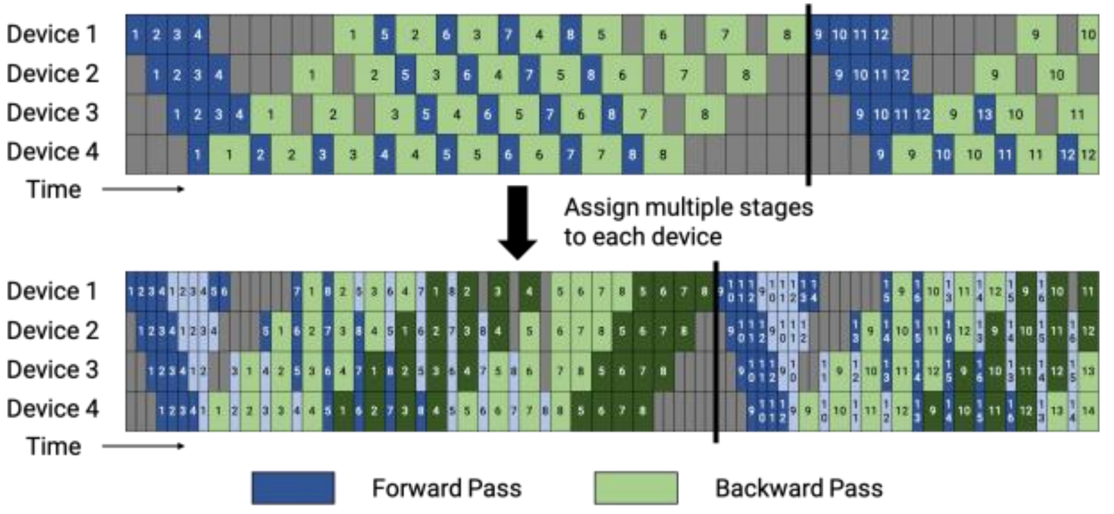
Easy Scaling with Micro-Batch Pipeline Parallelism



[Yanping Huang et al. GPipe: Easy Scaling with Micro-Batch Pipeline Parallelism, Advances in Neural Information Processing Systems 32 (NeurIPS 2019)]

Interleaved Pipeline

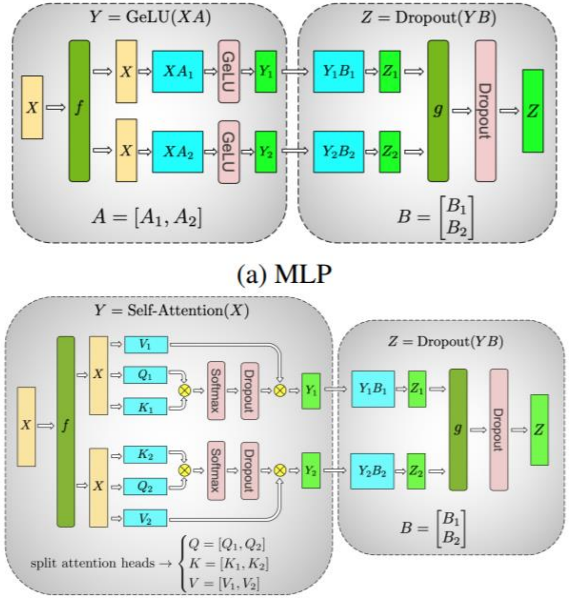
Reduce pipeline Bubble with more communication



https://github.com/NVIDIA/Megatron-LM

DISTRIBUTED TRAINING ACROSS MACHINES

Model Distribution – Tensor Slicing Strategy



(b) Self-Attention

Figure 3. Blocks of Transformer with Model Parallelism. f and g are conjugate. f is an identity operator in the forward pass and all reduce in the backward pass while g is an all reduce in the forward pass and identity in the backward pass.

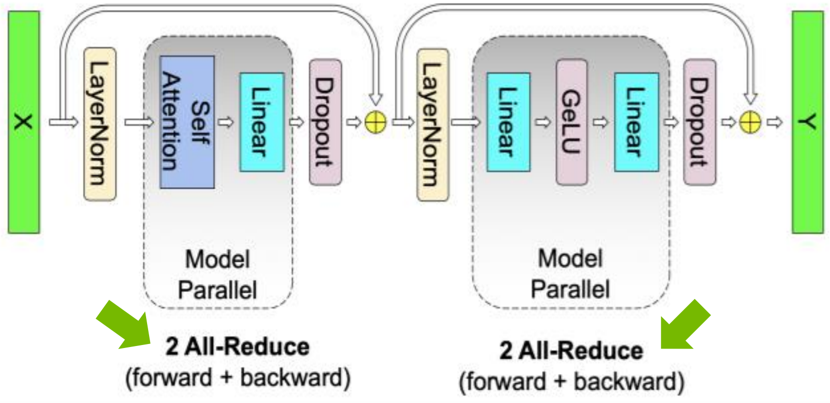


Figure 4. Communication operations in a transformer layer. There are 4 total communication operations in the forward and backward pass of a single model parallel transformer layer.

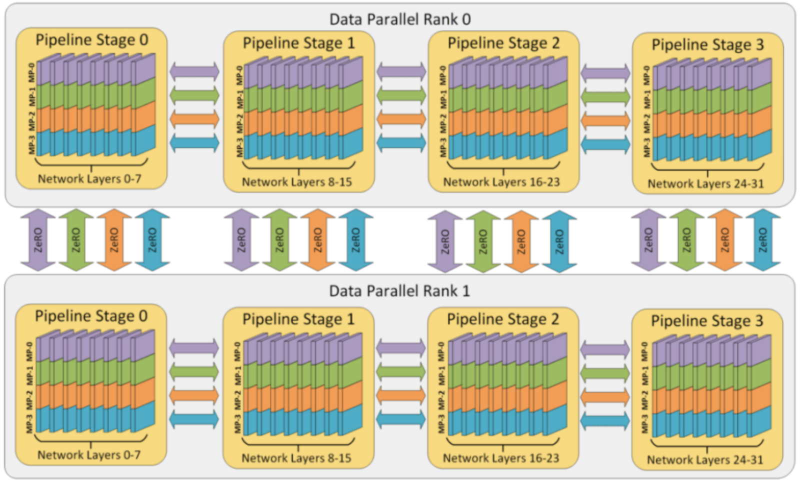
Shoeybi, M., Patwary, M., Puri, R., LeGresley, P., Casper, J., & Catanzaro, B. (2019). Megatron-lm: Training multi-billion parameter language models using gpu model parallelism.
https://github.com/NVIDIA/Megatron-LM

DISTRIBUTED TRAINING ACROSS MACHINES

Model Distribution – Hybrid Strategy - DeepSpeed

DeepSpeed

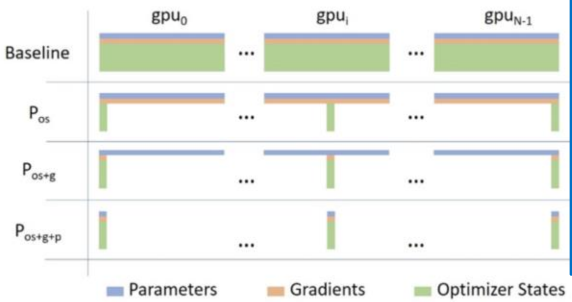
Data Parallel (ZeRO Redundancy Optimizer) | Tensor Slicing Megatron | Model Pipelining



Example 3D parallelism with 32 workers

ZeRO Redundancy Optimizer

Maintain data parallelism communication volume and reduce memory footprint



	Memory Consumption: Formulation	Memory Consumption: Specific Example (K=12 Ψ=7.5B N_d=64)	Comm Volume
Baseline	$(2+2+K)*\Psi$	120GB	1x
P_{os}	$2\Psi + 2\Psi + \frac{K*\Psi}{N_d}$	31.4GB	1x
P_{os+g}	$2\Psi + \frac{(2+K)*\Psi}{N_d}$	16.6GB	1x
P_{os+g+p}	$\frac{(2+2+K)*\Psi}{N_d}$	1.9GB	1.5x

Figure 1: Memory savings and communication volume for the three stages of ZeRO compared with standard data parallel baseline. In the memory consumption formula, Ψ refers to the number of parameters in a model and K is the optimizer specific constant term. As a specific example, we show the memory consumption for a 7.5B parameter model using Adam optimizer where K=12 on 64 GPUs. We also show the communication volume of ZeRO relative to the baseline.

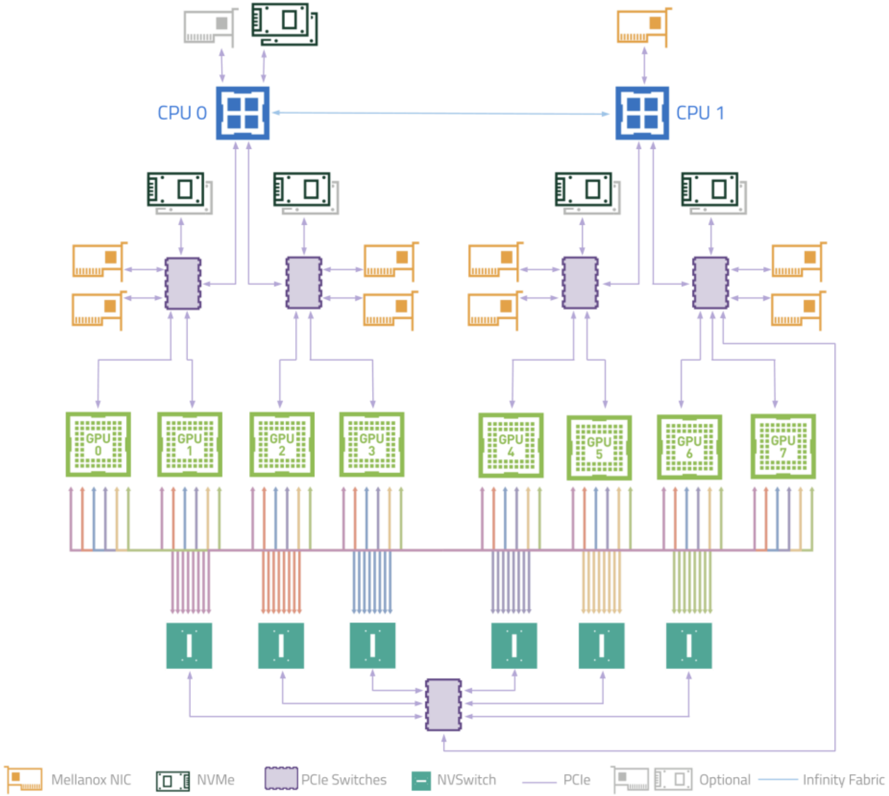
https://www.deepspeed.ai/

DISTRIBUTED TRAINING OPTIMIZATION

DISTRIBUTED TRAINING ACROSS MACHINES

Hardware Topology

NVIDIA DGX A100 System



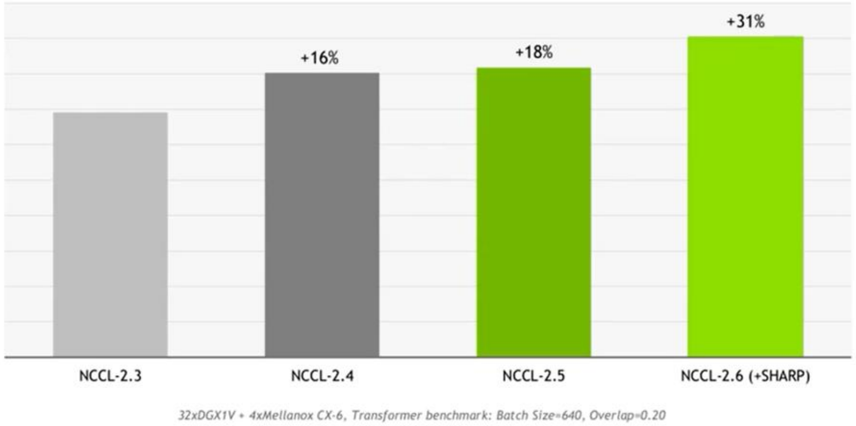
175B Transformer ~ 34 days on a 1024 A100
1T Transformer ~ 84 days on 3072 A100

GRADIENT EXCHANGE CHALLENGES

Optimized Inter GPU Communication - NCCL

NVIDIA Collective Communications Library

- Automatic topology detection
- Graph search for the optimal set of rings and trees with the highest bandwidth and lowest latency over PCIe and NVLink high-speed interconnects within a node and over NVIDIA Mellanox Network across nodes
- Provide routines such as all-gather, all-reduce, broadcast, reduce, reduce-scatter, point-to-point send and receive
- Integrated within several Deep Learning frameworks such as Caffe2, MxNet, PyTorch



Transformer

The Training speedup takes NCCL 2.3 as a reference.

https://developer.nvidia.com/nccl

DISTRIBUTED TRAINING ACROSS MACHINES

Large Batch Size

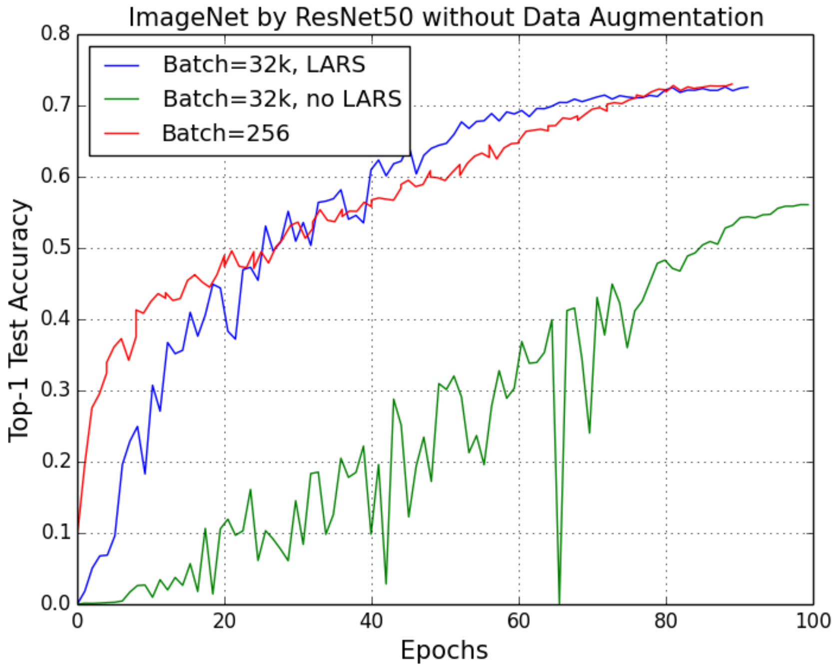
Compensate the batch size increase by a learning rate increase

Large batch size lead to noisy gradients

Trust Ratio based optimizers: Instability when the ratio is too high

$$\lambda^l = \eta \times \frac{||w^l||}{||\nabla L(w^l)||}$$

- Layer-wise Adaptive Rate Scaling – LARS
- Layer-wise Adaptive Moments optimizer for Batch training - LAMB
- NVLAMB



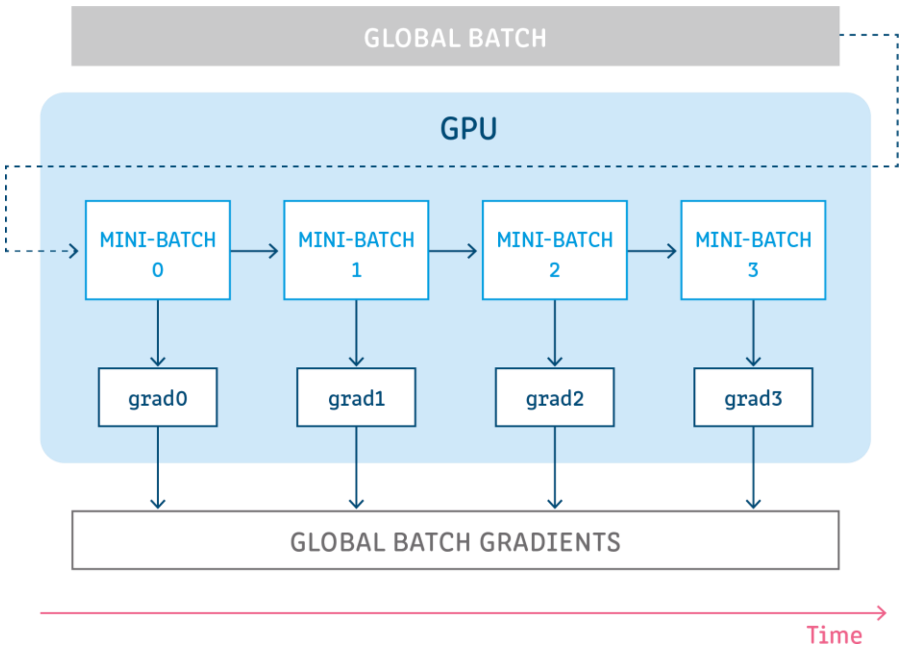
[Boris Ginsburg, Igor Gitman, Yang You. Large Batch Training of Convolutional Networks with Layer-wise Adaptive Rate Scaling. ICLR 2018]
[Yang You et al. LARGE BATCH OPTIMIZATION FOR DEEP LEARNING: TRAINING BERT IN 76 MINUTES. ICLR 2020]

DISTRIBUTED TRAINING ACROSS MACHINES

Large Batch Size - Gradient accumulation:

Batch size is bounded by the GPU memory

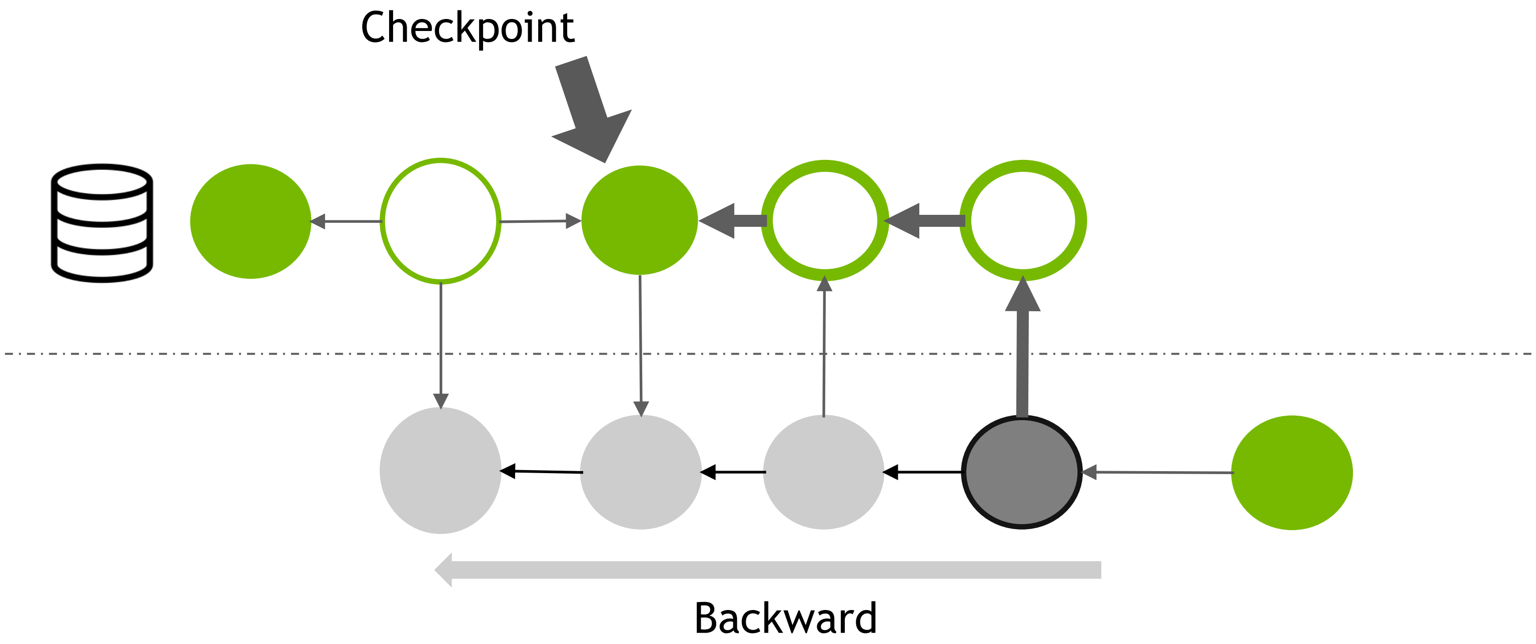
Gradient accumulation: Split the batch into several mini-batches that will be run sequentially



https://towardsdatascience.com/what-is-gradient-accumulation-in-deep-learning-ec034122cfa

TRAINING MEMORY REDUCTION

Gradient-Checkpointing



https://github.com/cybertronai/gradient-checkpointing

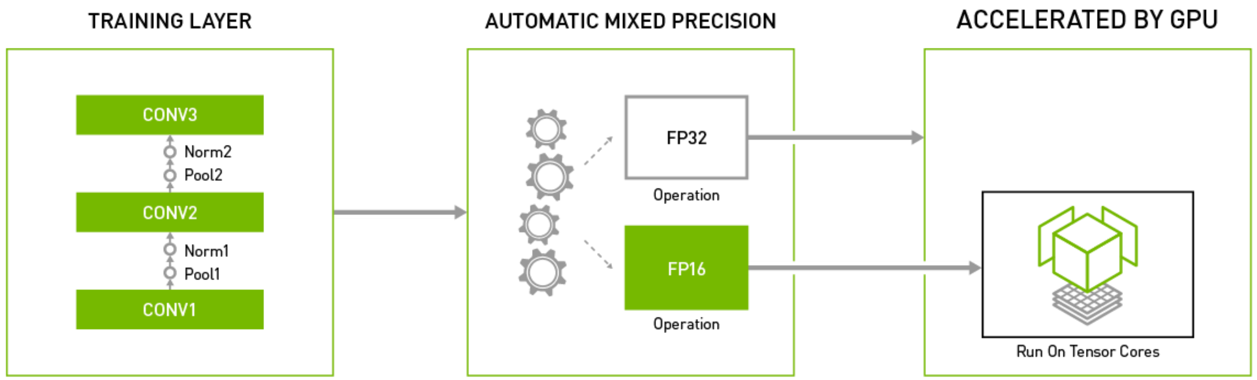
TRAINING MEMORY BANDWIDTH REDUCTION

Automatic Mixed Precision

Use different numerical precisions in a computational method.

Performing operations in half-precision format while storing minimal information in single-precision to retain as much information as possible in critical parts of the network

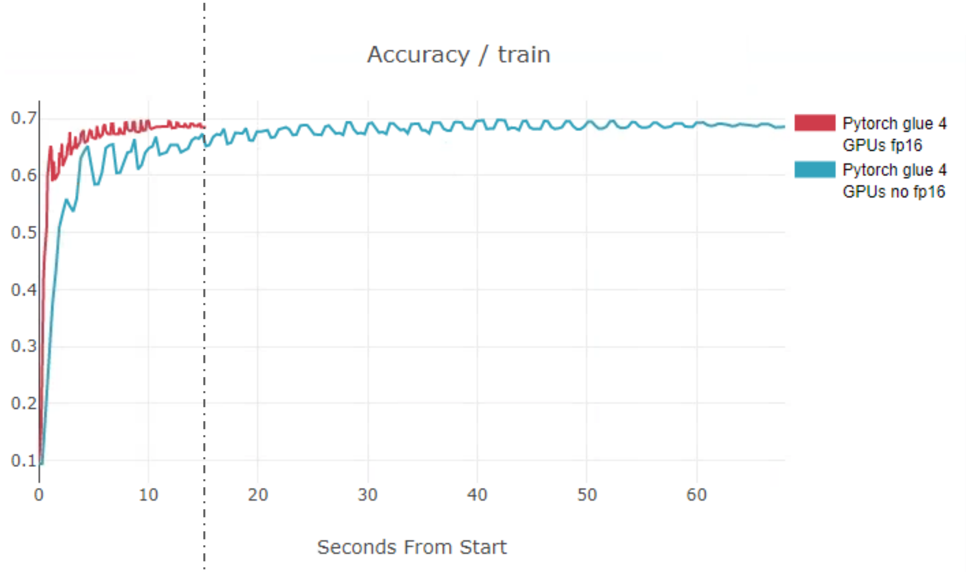
Tensor cores: up to 3x overall speedup on the most arithmetically intense model architectures.



How to train using mixed precision, see the Mixed Precision Training paper and Training With Mixed Precision documentation.

TRAINING MEMORY BANDWIDTH REDUCTION

Automatic Mixed Precision – APEX library



1 Epoch BERT finetuning for Sentiment Analysis
With/without fp16 (all other arguments are identical)
DGX-1 4 V100 16G, pytorch:20.06-py3, BS=32, max len=128,
warmup= 0.1, dynamic loss scaling
code available for reproduction

```
Iteration: 98% 205/209 [01:08<00:01, 3.03it/s]A
Iteration: 98% 205/209 [01:09<00:01, 3.02it/s]A
Iteration: 98% 205/209 [01:08<00:01, 3.02it/s]A
Iteration: 98% 205/209 [01:09<00:01, 3.03it/s]A
Iteration: 99% 206/209 [01:09<00:00, 3.04it/s]A
Iteration: 99% 206/209 [01:08<00:00, 3.03it/s]A
Iteration: 99% 206/209 [01:08<00:00, 3.04it/s]A
Iteration: 99% 206/209 [01:10<00:00, 3.03it/s]A
Iteration: 99% 207/209 [01:08<00:00, 3.03it/s]A
Iteration: 99% 207/209 [01:09<00:00, 3.03it/s]A
Iteration: 99% 207/209 [01:08<00:00, 3.02it/s]A
Iteration: 99% 207/209 [01:10<00:00, 3.03it/s]A
Iteration: 100% 208/209 [01:09<00:00, 3.04it/s]A
Iteration: 100% 208/209 [01:10<00:00, 3.02it/s]A
Iteration: 100% 208/209 [01:08<00:00, 3.02it/s]A
Iteration: 100% 208/209 [01:10<00:00, 3.03it/s]A
```

Train without fp16

```
Iteration: 95% 199/209 [00:15<00:00, 13.97it/s]A
Iteration: 95% 199/209 [00:16<00:00, 13.97it/s]A
Iteration: 96% 201/209 [00:15<00:00, 13.94it/s]
Iteration: 96% 201/209 [00:15<00:00, 13.93it/s]A
Iteration: 96% 201/209 [00:16<00:00, 13.96it/s]A
Iteration: 96% 201/209 [00:16<00:00, 13.82it/s]A
Iteration: 97% 203/209 [00:15<00:00, 13.85it/s]A
Iteration: 97% 203/209 [00:16<00:00, 13.87it/s]A
Iteration: 97% 203/209 [00:17<00:00, 13.82it/s]A
Iteration: 98% 205/209 [00:15<00:00, 13.80it/s]AA
Iteration: 98% 205/209 [00:17<00:00, 13.82it/s]AA
Iteration: 99% 207/209 [00:16<00:00, 13.84it/s]A
Iteration: 99% 207/209 [00:17<00:00, 13.84it/s]A
Iteration: 100% 209/209 [00:16<00:00, 14.21it/s]A
Iteration: 100% 209/209 [00:15<00:00, 14.18it/s]AA
```

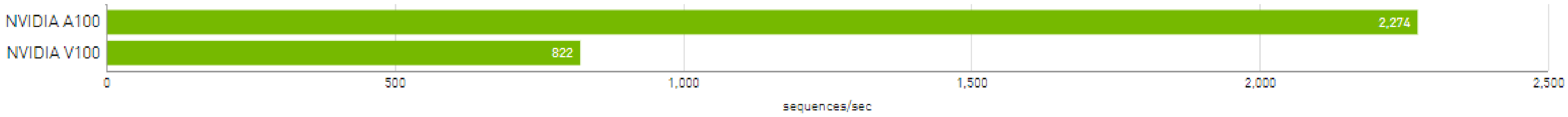
Train with fp16

BERT LARGE PRETRAINING

Encapsulate Best Practices

Training Natural Language Processing

BERT Pre-Training Throughput



DGX-A100 server w/ 8x NVIDIA A100 on PyTorch | DGX-1 server w/ 8x NVIDIA V100 on PyTorch (2/3)Phase 1 and (1/3)Phase 2 | Precision: FP16 for A100 and Mixed for V100 | Sequence Length for Phase 1= 128 and Phase 2 = 512

NVIDIA A100 BERT Training Benchmarks

Framework	Network	Throughput	GPU	Server	Container	Precision	Batch Size	Dataset	GPU Version
PyTorch	BERT Pre-Training	2,274 sequences/sec	8x A100	DGX-A100	-	FP16	-	Wikipedia+BookCorpus	A100 SXM4-40GB

DGX-A100 server w/ 8x NVIDIA A100 on PyTorch (2/3)Phase 1 and (1/3)Phase 2 | Sequence Length for Phase 1 = 128 and Phase 2 = 512

https://developer.nvidia.com/deep-learning-performance-training-inference

AGENDA

Overview of Modern NLP Model

What are transformers, Success Reasons

Models Training

What are the challenges of building modern NLP models?

Distributed Training, Memory usage reduction

Model Deployment

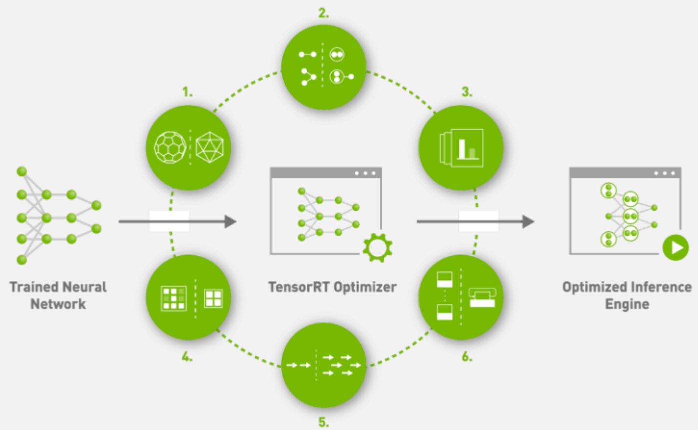
What are the challenges of deploying NLP models?

Model Optimization and efficient Model Serving

DEPLOYMENT CHALLENGES

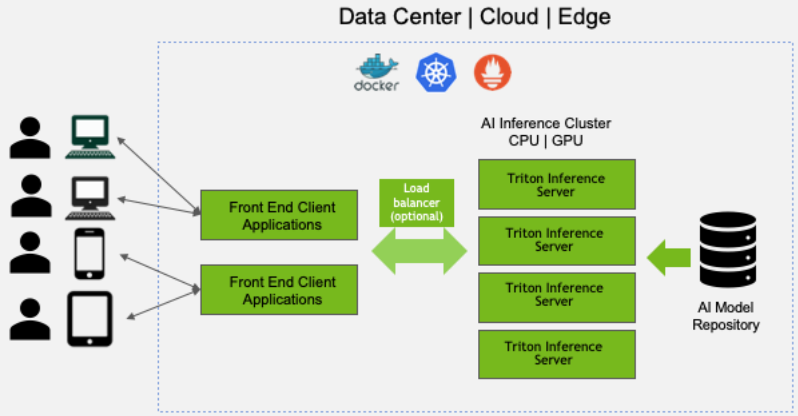
Reduce latency & Maximize throughput

Model Optimization



NVIDIA TensorRT

Model Serving



Triton Inference Server

Model Optimization

NVIDIA TensorRT

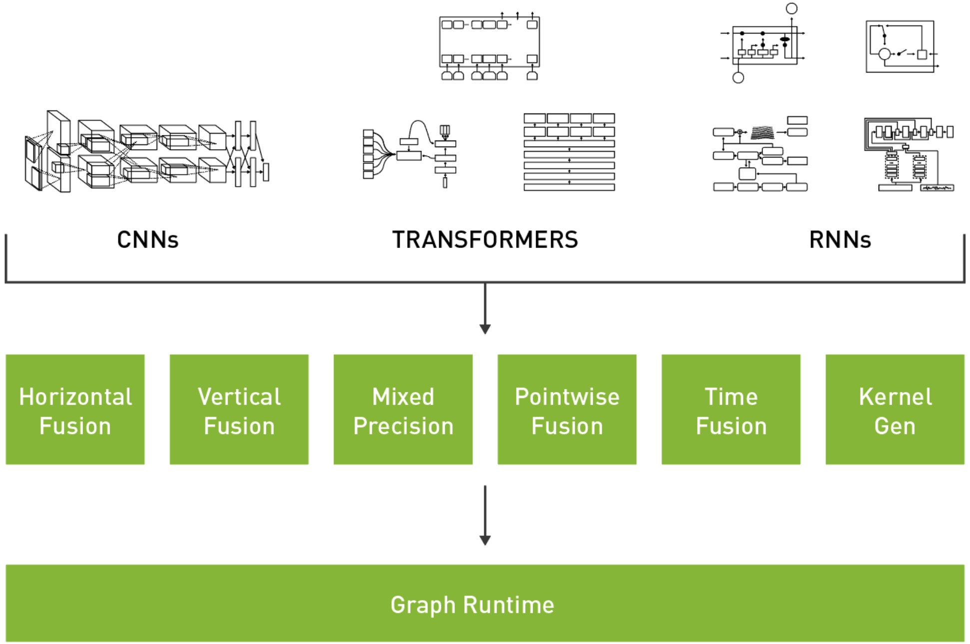
Weight & Activation Precision Calibration: Maximizes throughput by quantizing models to INT8 while preserving accuracy

Layer & Tensor Fusion: Optimizes use of GPU memory and bandwidth by fusing nodes in a kernel

Kernel Auto-Tuning: Selects best data layers and algorithms based on target GPU platform

Dynamic Tensor Memory: Minimizes memory footprint and re-uses memory for tensors efficiently

Time Fusion: Optimizes recurrent neural networks over time steps with dynamically generated kernels

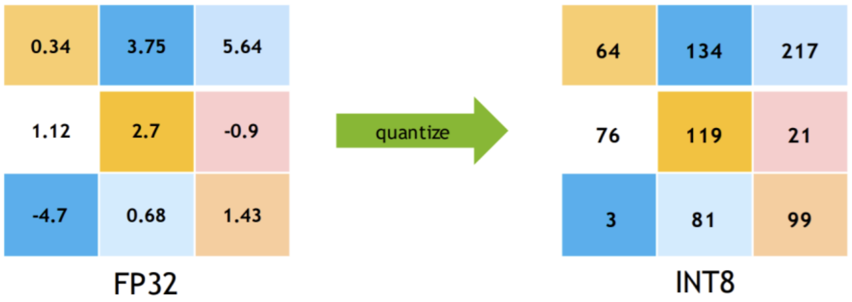


https://developer.nvidia.com/tensorrt

MODEL OPTIMIZATION

TensorRT - Quantization

Convert continuous values to discrete set of values using linear/non-linear scaling techniques.



Relative to fp32 math

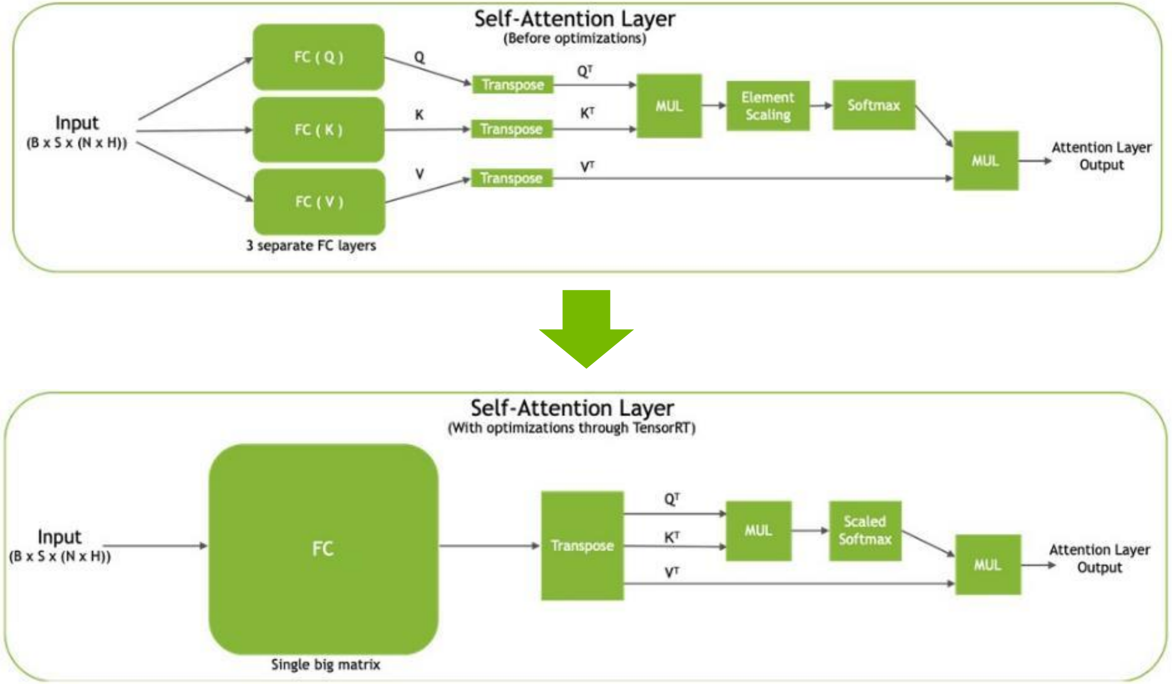
Input Type	Accumulation Type	Relative math throughput	Bandwidth savings
FP16	FP16	8x	2x
INT8	INT32	16x	4x
INT4	INT32	32x	8x
INT1	INT32	128x	32x

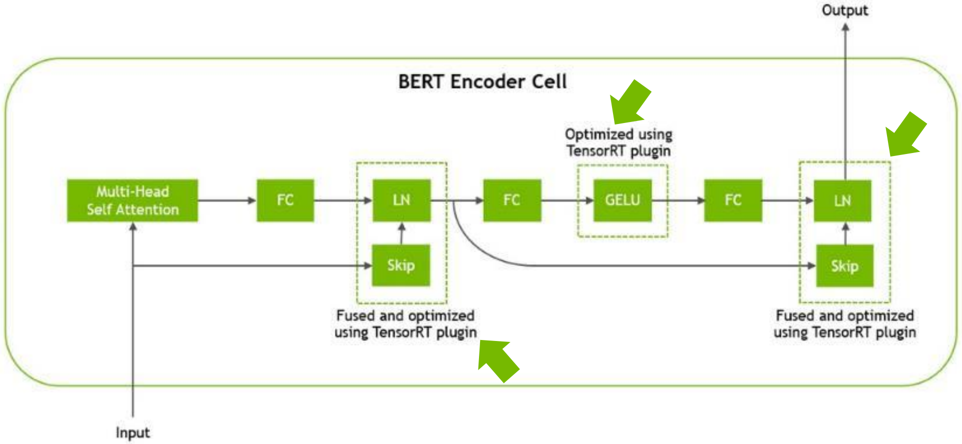
Bert large uncased	FP32	Int8 (GeLU10)	Rel Err %
MRPC	0.855	0.843	0.70%
SQuAD 1.1 (F1)	91.01	90.40	0.67%

Learn more: DEPLOYING QUANTIZATION-AWARE TRAINED NETWORKS USING TENSORRT SESSION

MODEL OPTIMIZATION

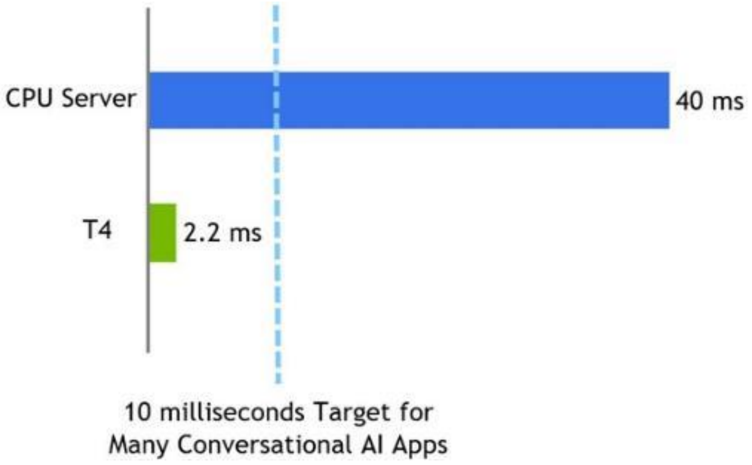
TensorRT - Transformer Layers Fusion





MODEL OPTIMIZATION

NVIDIA TensorRT



Using a Tesla T4 GPU, BERT optimized with TensorRT can perform inference in 2.2 ms for a QA task similar to available in SQuAD with batch size =1 and sequence length = 128.

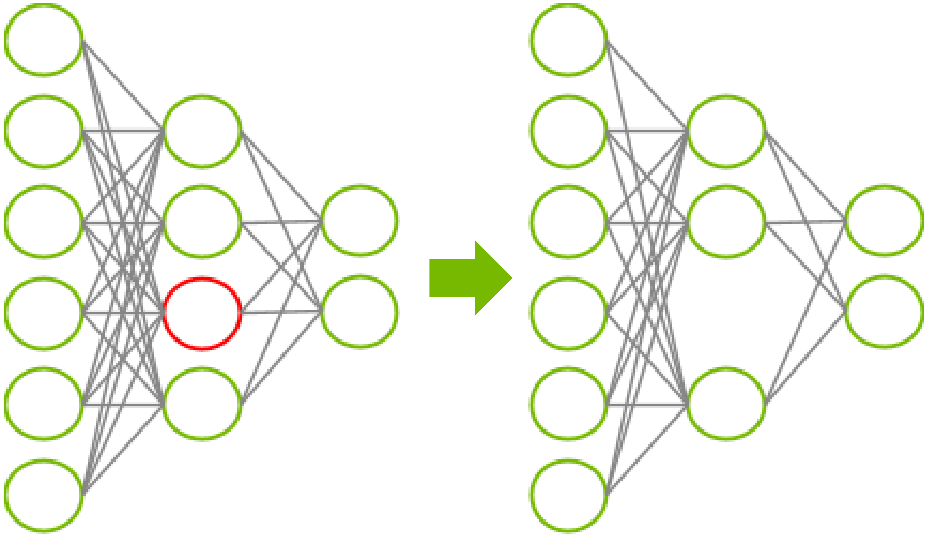
MODEL OPTIMIZATION

Pruning

Reduce the complexity of neural networks by
Removing Unnecessary Connections

- Reduce memory bandwidth
- Reduce memory footprint
- Accelerate the compute

Maintain accuracy of the original unpruned network



6 inputs, 6 neurons (including 2 outputs), 32 connections

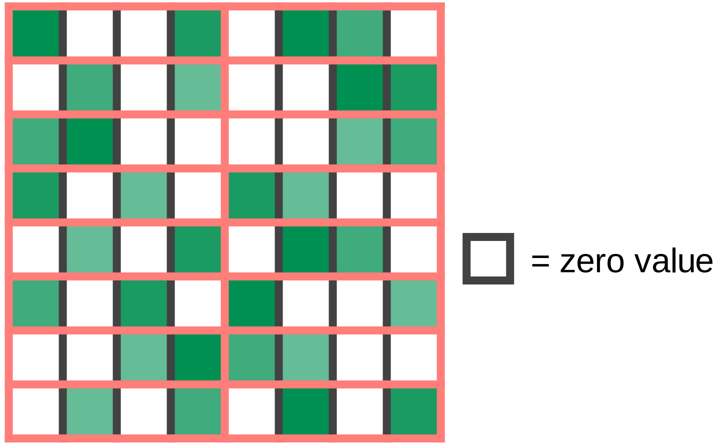
6 inputs, 5 neurons (including 2 outputs), 24 connections

MODEL OPTIMIZATION

Structured Sparsity in A100

Fine-grained structured sparsity for Tensor Cores

- 50% fine-grained sparsity
- **2:4 pattern:** 2 values out of each contiguous block of 4 must be 0



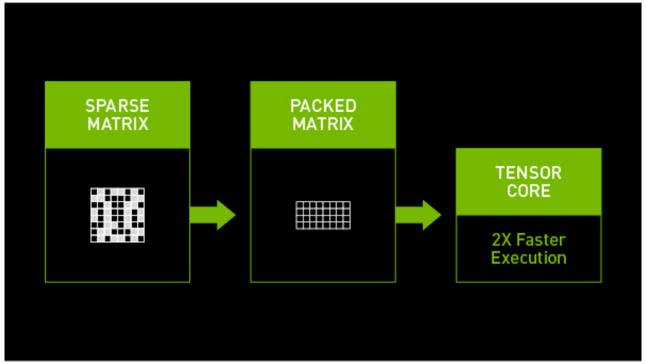
2:4 structured-sparse matrix

Accuracy:

- Medium sparsity level (50%), fine-grained
- Training: a recipe shown to work across tasks and networks

Speedup:

- Specialized Tensor Core support for sparse math
- Structured: lends itself to efficient memory utilization



MODEL OPTIMIZATION

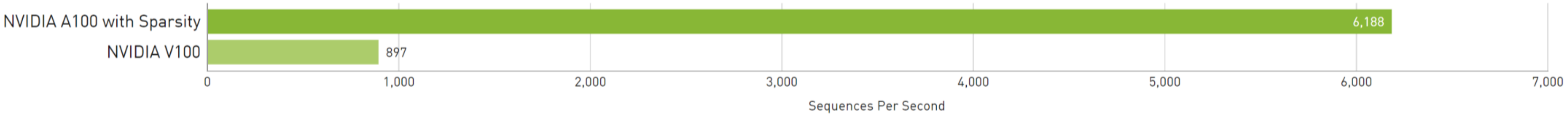
Sparsity on BERT for SQUAD Q&A

NVIDIA A100 BERT Inference Benchmarks

Network	Network Type	Batch Size	Throughput	Efficiency	Latency (ms)	GPU	Server	Container	Precision	Dataset	Framework	GPU Version
BERT-Large with Sparsity	Attention	94	6,188 sequences/sec	-	-	1x A100	DGX-A100	-	INT8	SQuaD v1.1	-	A100 SXM4-40GB

A100 with 7 MIG instances of 1g.5gb | Sequence length=128 | Efficiency based on board power
Containers with a hyphen indicates a pre-release container

BERT Inference Throughput



DGX-A100 server w/ 1x NVIDIA A100 with 7 MIG instances of 1g.5gb | Batch Size = 94 | Precision: INT8 | Sequence Length = 128
DGX-1 server w/ 1x NVIDIA V100 | TensorRT 7.1 | Batch Size = 256 | Precision: Mixed | Sequence Length = 128

https://developer.nvidia.com/deep-learning-performance-training-inference

EFFICIENT DEPLOYMENT

Production Data Center Inference Server

MAXIMIZE THROUGHPUT

Handle the maximum number of users at a time

MINIMIZE LATENCY

Customer experience depends on response time

ZERO DOWNTIME

Deploy updates without disrupting your service

MINIMIZE COST

Choosing the right hardware for the job and ensuring high levels of utilization

EFFICIENT DEPLOYMENT

Triton Takes Care of Plumbing To Deploy Models for Inference

Multiple Frameworks

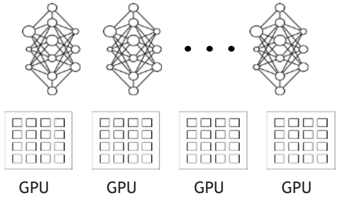
TensorFlow TensorRT

PYTORCH Custom ONNX RUNTIME

All Major Framework Backends For Flexibility & Consistency

Standard HTTP/gRPC Communication

Concurrent Execution



Automatically Runs Multiple Models Concurrently On One Or More GPUs To Maximize Utilization

Different Types of Queries

Real time Batch Stream Ensemble

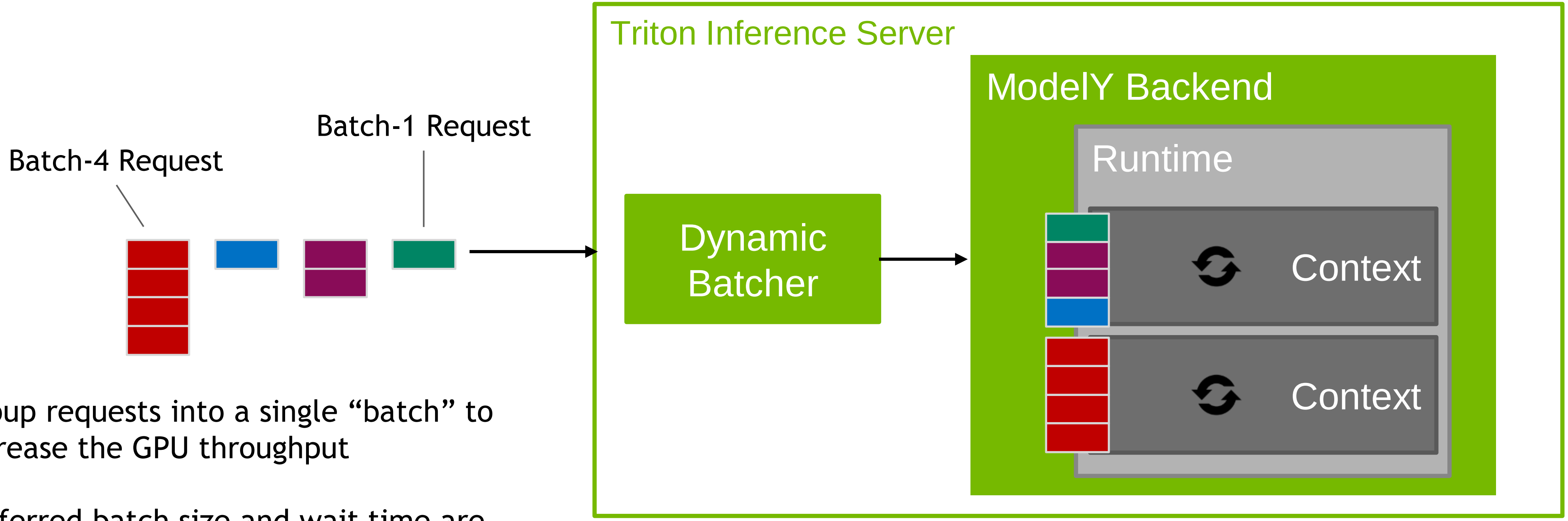
Supports Different Types Of Inference Queries For Different Use Cases

Dynamic Batching

Dynamic Batching Maximizes Throughput Under Latency Constraint

EFFICIENT DEPLOYMENT

DYNAMIC BATCHING SCHEDULER



Group requests into a single “batch” to increase the GPU throughput

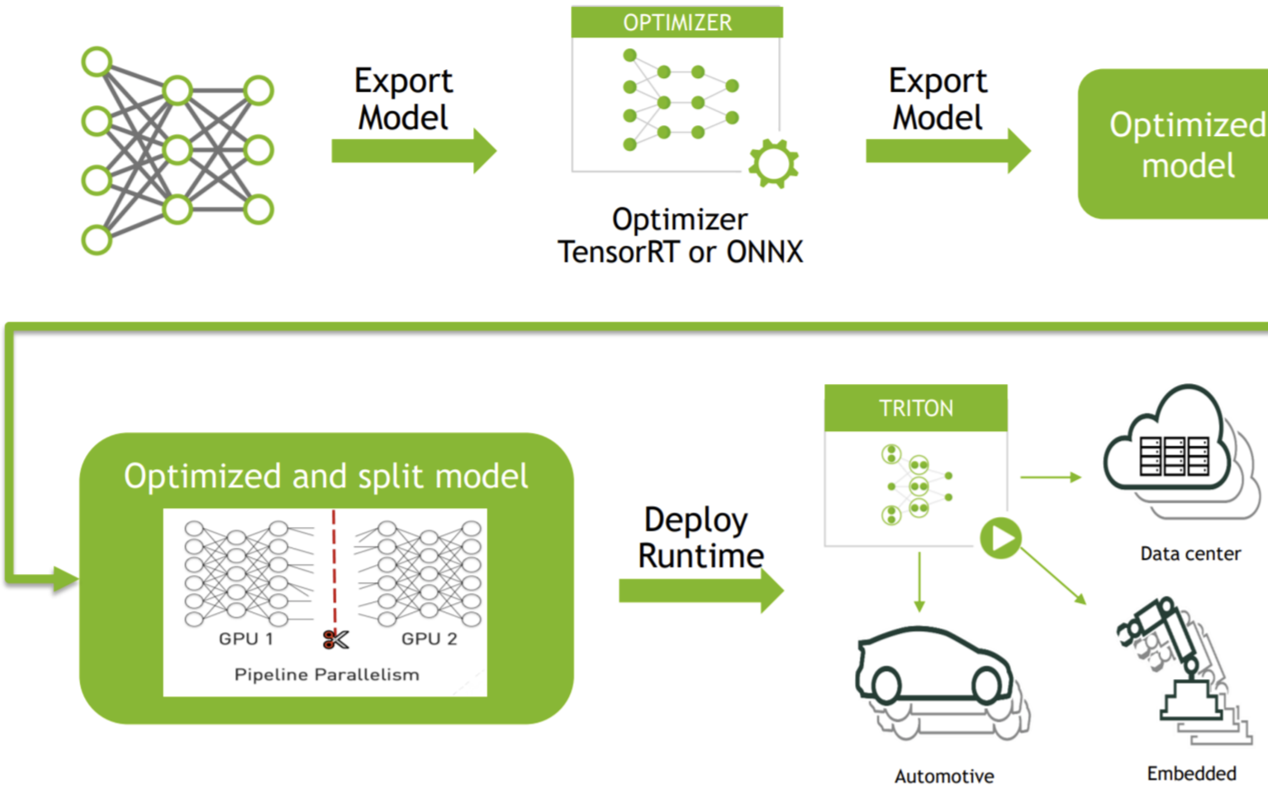
Preferred batch size and wait time are configuration options.

Assume 4 gives best utilization in this example.

```
max_batch_size: 8
dynamic_batching {
   preferred_batch_size: [ 4, 8 ]
   max_queue_delay_microseconds: 100
}
```

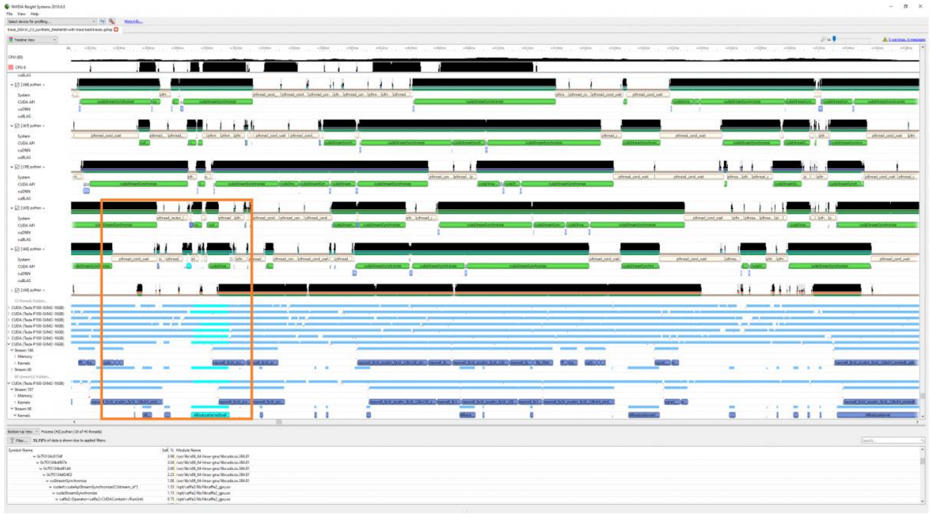
EFFICIENT DEPLOYMENT

Megatron GPT-3 large model inference with Triton Inference Server and ONNX runtime



DEEP LEARNING PROFILER

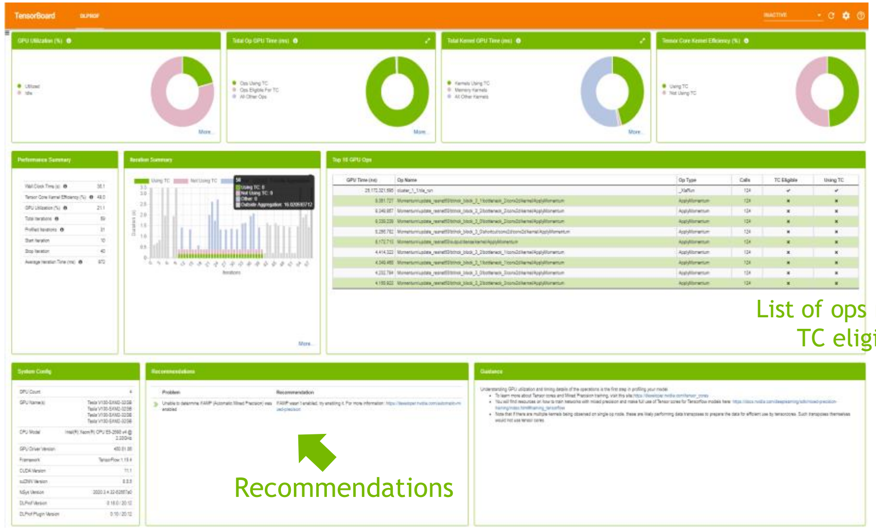
NVIDIA Nsight Systems



```
nsys profile --trace=cuda,cudnn,cublas,osrt,nvtx --delay=60 python my_dnn_script.py
```

Tensorboard-plugin

% GPU Utilization



List of ops running on GPUs, TC eligible, using TC

Recommendations

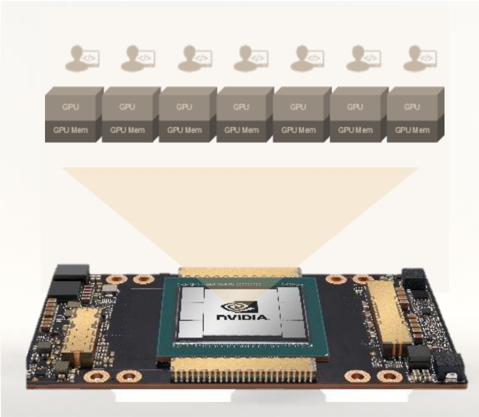
https://developer.nvidia.com/nsight-systems
https://docs.nvidia.com/deeplearning/frameworks/tensorboard-plugin-user-guide/

PERSPECTIVES & TAKEAWAY

ENABLING LARGE SCALE MODELS

NVIDIA Solutions

- **Training:** Megatron-LM, Distributed Training, Optimizer (LARC, LAMB), Automatic Mixed Precision, NGC
- **Inference:** TensorRT, Structured Sparsity, Triton Inference Server
- **Reference architecture:** A100, DGX A100, DGX SuperPOD



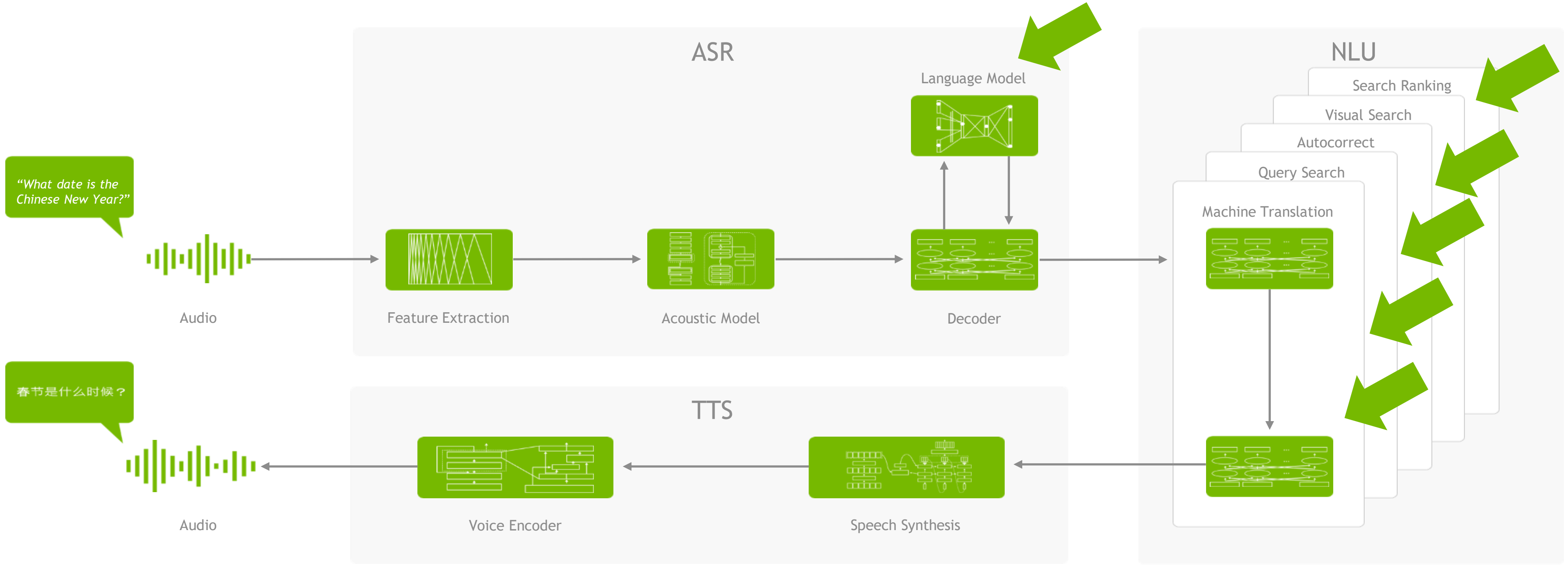
A100 With MIG

DGX A100

DGX SUPERPOD

PERSPECTIVES & TAKEAWAY

NLP in Real Applications



NVIDIA RIVA

Fully Accelerated Framework for Multimodal Conversational AI Services

Pre-trained Models

NVIDIA GPU CLOUD

Customize

Transfer Learning Toolkit (TLT)

NeMo

Deploy with Riva

Dialog Manager

Computer Vision

Language Understanding

Speech Synthesis

Speech Recognition

Audio

Multi-speaker
Domain specific
Transcription

Available in Riva 1.0 Beta

Available in future version

https://developer.nvidia.com/riva

NVIDIA

THANK YOU!

Q&A