Distributed Training, Inferencing and Customization of Large Language Models

26th May 2023

Ettikan Kandasamy Karuppiah (Ph.D)
Director/Technologist, Asia Pacific South Region

Training & Deploying of Foundation Models are Challenging

Foundation models are neural networks trained on massive unlabeled datasets to handle a wide variety of tasks

- Mountains of Training Data
- Large-scale compute infrastructure for training & inferencing, costing $10 M+ in just cloud costs
- Complex techniques to train and deploy on large-scale infrastructure
- Deep technical expertise

Training & Deploying of GPT-3

Training

Train 300B tokens in days (A100) - BF16			
	800 GPUs (5x DGX SuperPod)	3x DGX SuperPod	1x DGX SuperPod
GPT-3: 126M	**0.07**	**0.12**	**0.37**
GPT-3: 5B	**0.8**	**1.3**	**3.9**
GPT-3: 20B	**3.6**	**6**	**18.1**
GPT-3: 40B	**6.6**	**10.9**	**32.8**
GPT-3: 175B	**28**	**46.7**	**140**

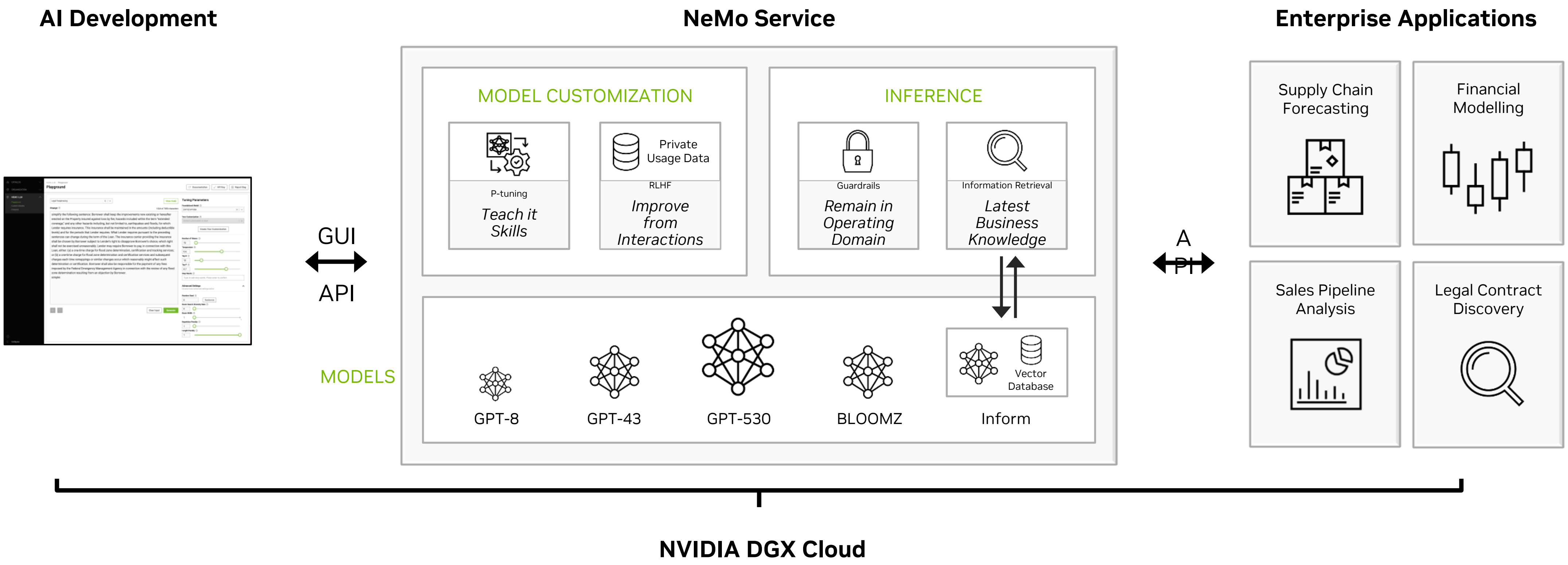
Inference

Estimated Inference Capacity					
GPT-3 Model Parameter Count	Precision	Input/Output Length (Tokens)	Batch Size	Estimated GPU Memory Size	Estimated # of A100 80GB
100M - 3B	FP16	60/20 200/200	1-256	200MB - 6GB	1
5B - 20B	FP16	60/20 200/200	1-256	10GB - 600GB	1-8
100B - 300B	FP16	60/20 200/200	1-256	200GB - 2TB	8-32 GPUs 1-4 Nodes
500B - 1T	FP16	60/20 200/200	1-256	1TB - 5TB	16-64 GPUs 2-8 Nodes

NeMo Service Introduction

NVIDIA NeMo Service

Enterprise Hyper-Personalization and At-Scale Deployment of Intelligent Large Language Models



Your Enterprise AI
Customize state-of-the-art pre-trained language models

Easily Develop & Connect Applications
GUI-based Playground and Scalable Cloud API

Deploy Anywhere
In the Service, Across Public Clouds, or On-Premises

Enterprise Support
Fully supported by NVIDIA AI Experts from Customization to Deployment At-Scale

Get Started with NeMo Service

Apply Now

Web Pages

- NVIDIA Generative AI Solutions
- NVIDIA NeMo Service

Blogs

- What are Large Language Models?
- What Are Large Language Models Used For?
- What are Foundation Models?
- How To Create A Custom Language Model?
- Adapting P-Tuning to Solve Non-English Downstream Tasks

GTC Sessions

- How to Build Generative AI for Enterprise Use-cases
- Leveraging Large Language Models for Generating Content
- Power Of Large Language Models: The Current State and Future Potential
- Generative AI Demystified

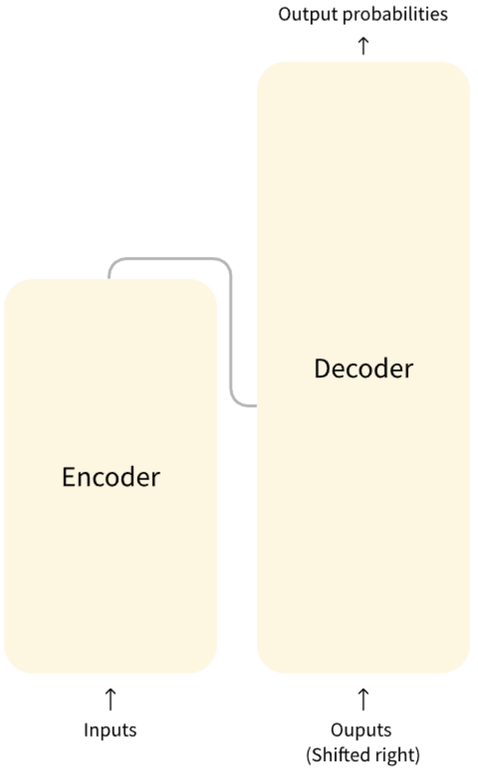
NeMo Framework – Deep Dive

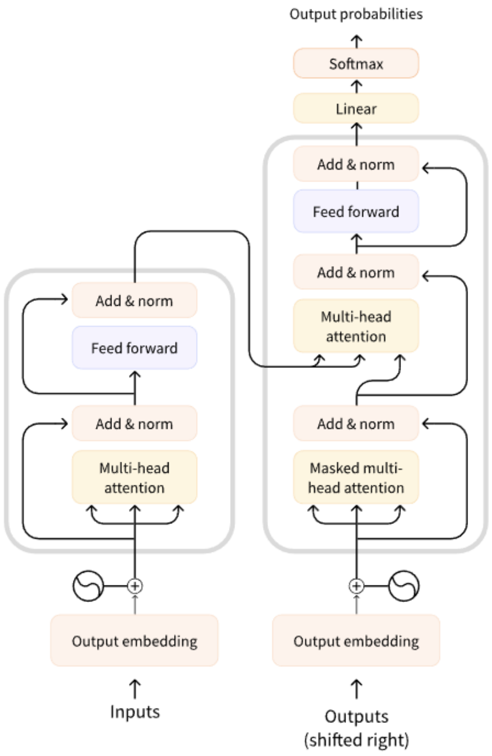
When Large-Language-Models Make Sense

	Traditional NLP Approach	Large Language Models
Requires labelled data	Yes	No
Parameters	100s of millions	Billions to trillions
Desired model capability	Specific (one model per task)	General (model can do many tasks)
Training frequency	Retrain frequently with task-specific training data	Never retrain, or retrain minimally

- Zero-Shot (or Few Shot Learning)
  - Painful & Impractical to get a large corpus of labelled data
- Models can learn new tasks
  - If you want models with "common sense" and can generalize well to new tasks
- A single model can serve all use-cases
  - At-scale you avoid costs and complexity of many models, saving cost in data curation, training, and managing deployment

ARCHITECTURE





- A **transformer** is a deep learning model that adopts the mechanism of self-attention, differentially weighting the significance of each part of the input data.
- Introduced in Attention Is All You Need
- Based on Encoder-Decoder Architecture, wherein encoder understands language, whilst decoder generates language

Transformers

The Next Wave of AI

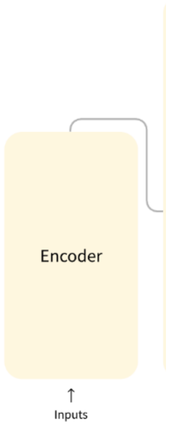
Encoders

For Understanding Language

Suited for task requiring an understanding of the full sentence, such as sentence classification, named entity recognition, and extractive question answering.

Supported Models

- BERT



Decoders

For Generative Models

Suited for tasks involving Text Generation

Supported Models:

- GPT-3

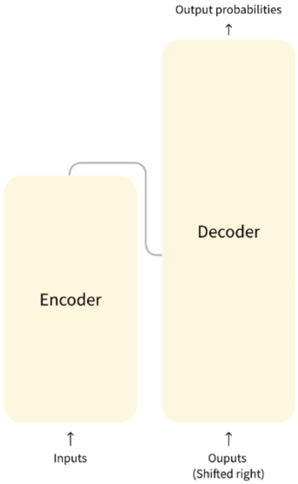
Encoder-Decoders

Sequence-to-Sequence

Suited for tasks around generating new sentences depending on a given input, such as summarization, translation, or generative question answering.

Supported Models:

- T5
- Multilingual – mT5

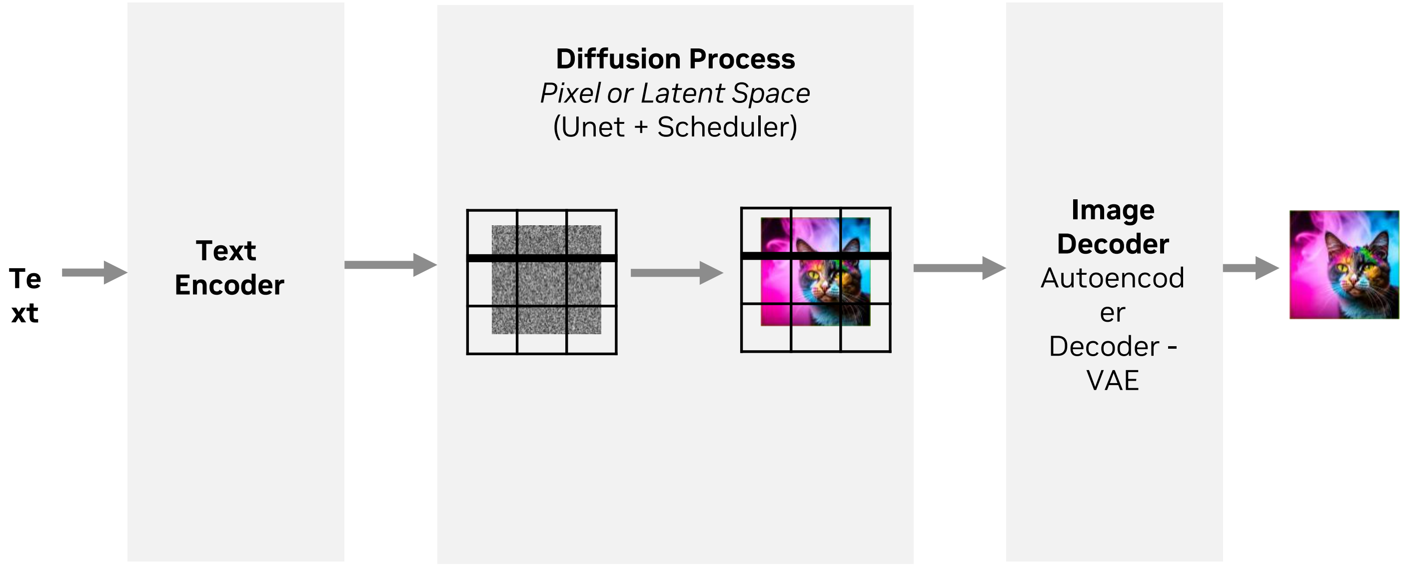


Supported Language Models

Support for Multi-Modal Models

Generative Image Models

Text to Image Generative Models



Supported Models In NeMo framework:

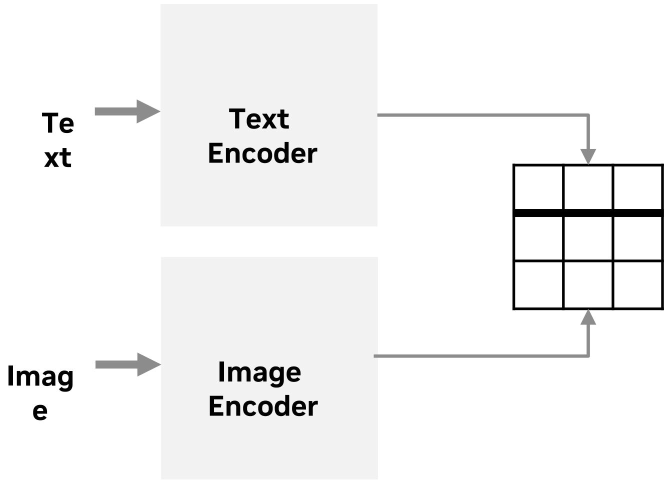
Diffusion in Latent Space: Stable Diffusion v1.5

Diffusion in Pixel Space: Imagen

Image-to-Image Models: Instruct-Pix2Pix (For editing images – No text encoder)

Discriminative

Suitable for Tasks Like Image Classification, Object Detection



Supported Models In NeMo framework:

Text-Encoder: Vision-Transformer

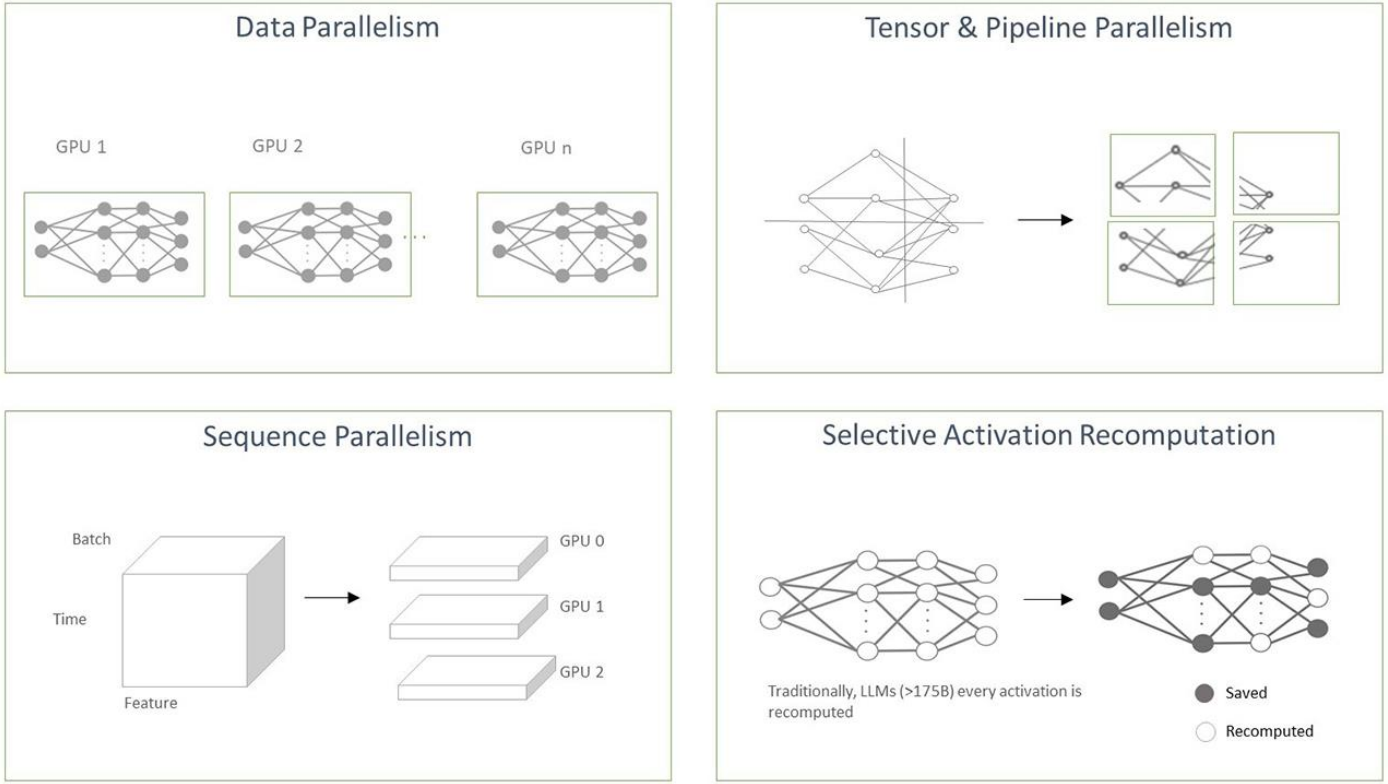
Multi-Modal: CLIP

Overall Model: ViT-CLIP

DISTRIBUTED TRAINING

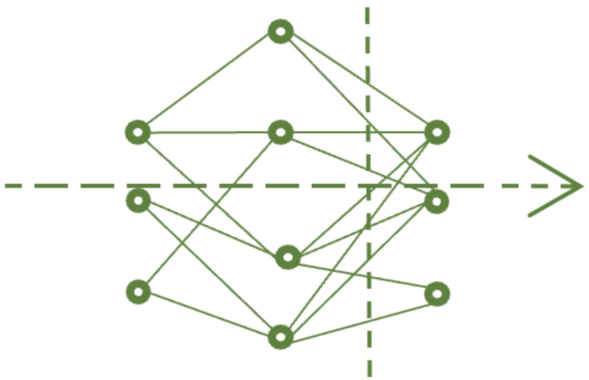
Overcoming Challenges of Training Foundation Model

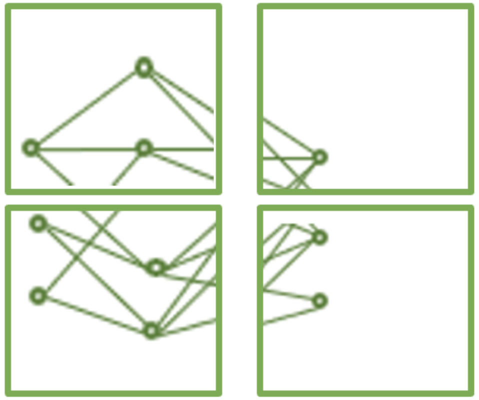
NeMo framework offers efficient algorithms to train large-scale models



- Requires extensive experimentation to configure hyperparameters
- Needs state-of-the-art algorithms to process internet-scale data across an entire datacenter

Maximize GPU Utilization over InfiniBand and Minimum Latency within a Single Node





Pipeline (Inter-Layer) Parallelism

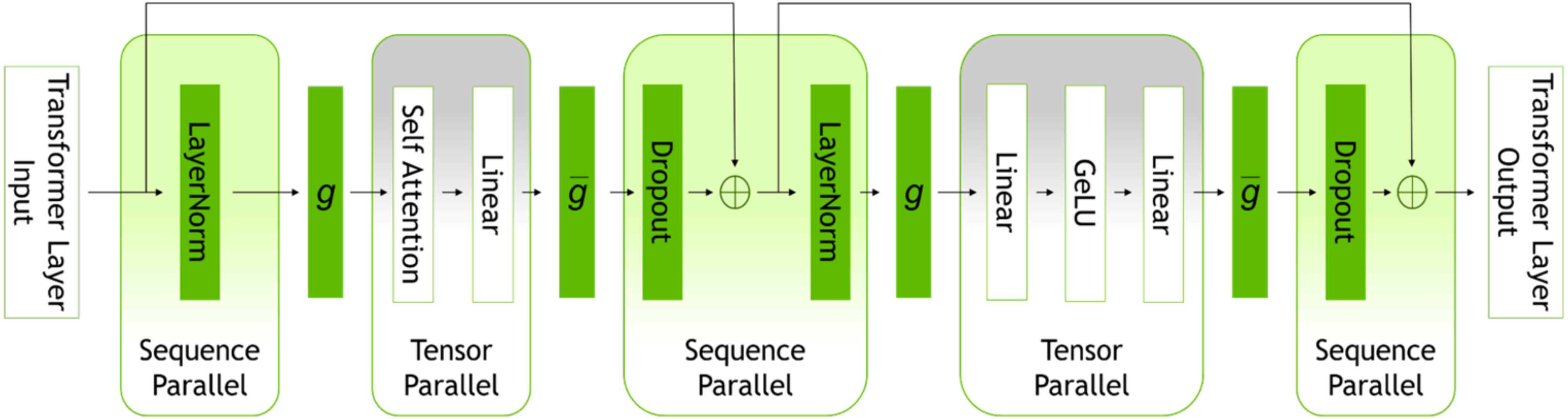
- Split contiguous sets of layers across multiple GPUs
- Layers 0,1,2 and layers 3,4,5 are on different GPUs

Tensor (Intra-Layer) Parallelism

- Split individual layers across multiple GPUs
- Devices compute different parts of Layers 0,1,2,3,4,5

Pipeline & Tensor Parallelism for Training

Training Models at Scale

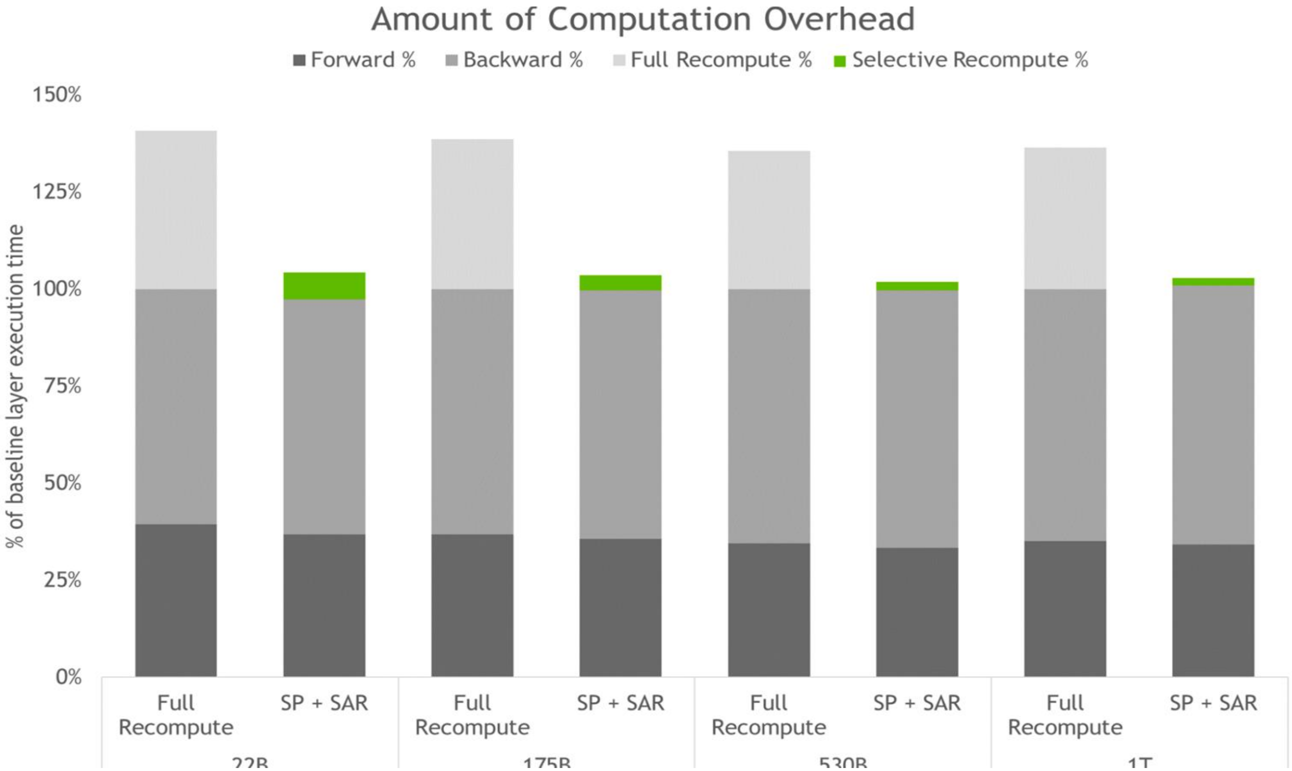


- Splits tensors across sequence dimension
- Reduce memory consumption of activation to reduce re-computation of activations during back-prop

Sequence Parallelism for Training

Increase throughput during back-propagation

Selective Activation Recomputation



- Choose activations to calculate based on compute-memory tradeoff
- Lower memory footprint of activations and increase throughput of network

Selective Activation Recomputation for Training

Distributed Training with Nemo

Example of Config

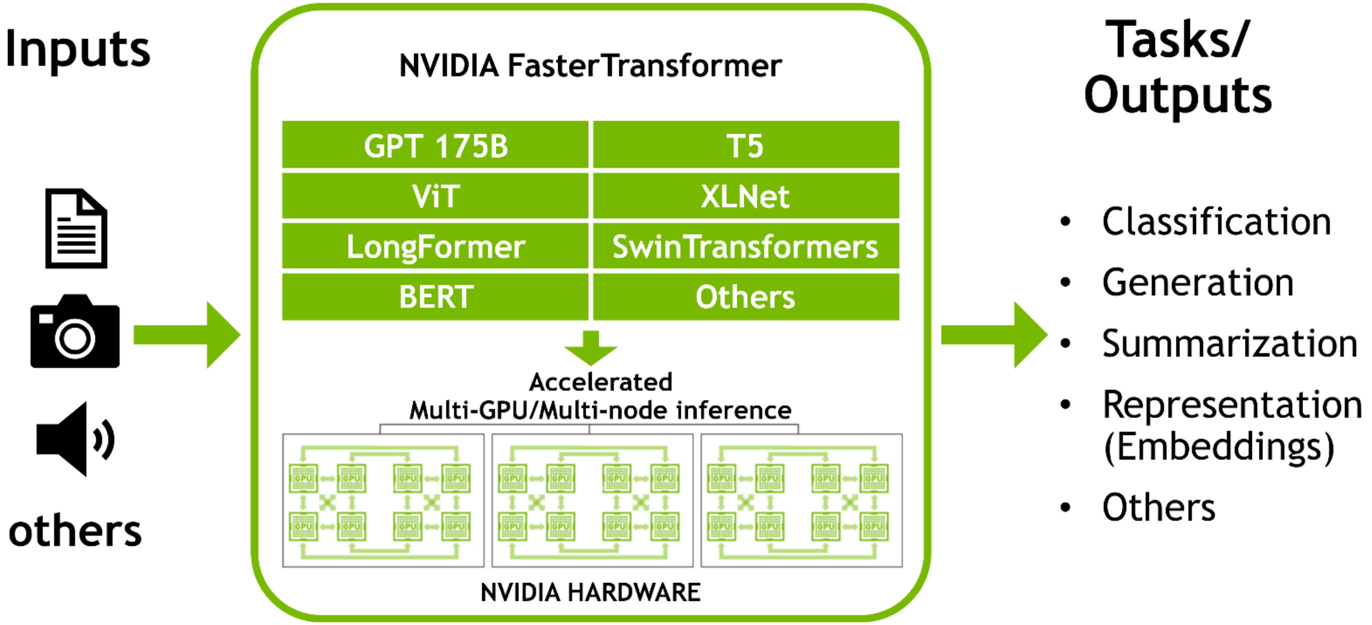
```
model:
.....
   tensor_model_parallel_size: 8
   pipeline_model_parallel_size: 16
......
 ## Activation Checkpointing
  activations_checkpoint_granularity: selective # 'selective' or 'full'
......

 ## Sequence Parallelism
  sequence_parallel: True
```

DISTRIBUTED INFERENCE

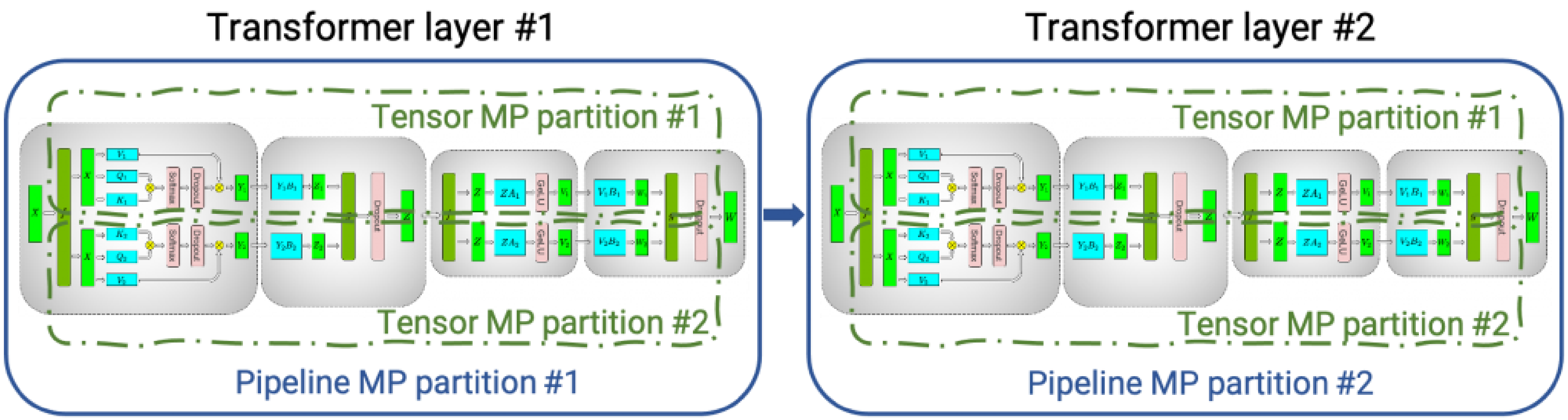
DISTRIBUTED INFERENCE WITH FASTERTRANSFORMER

- Accelerated engine for the inference of transformer-based models
- Leverage highly optimized cuBLAS, cuBLASLt , and cuSPARSELt libraries.
- Highly optimize transformer blocks.
  - Layer fusion
  - GEMM autotuning
  - Quantization
- Distributed inference with MNMG.
  - Usage of MPI and NCCL



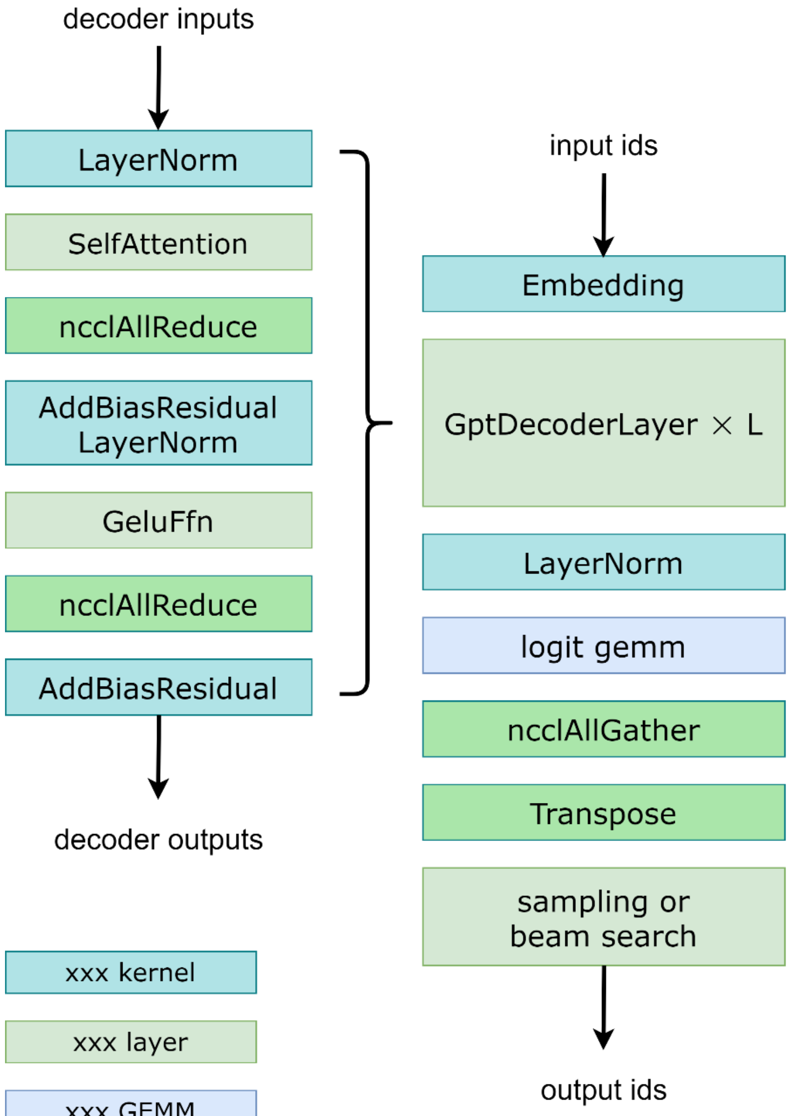
DISTRIBUTED INFERENCE WITH FASTERTRANSFORMER

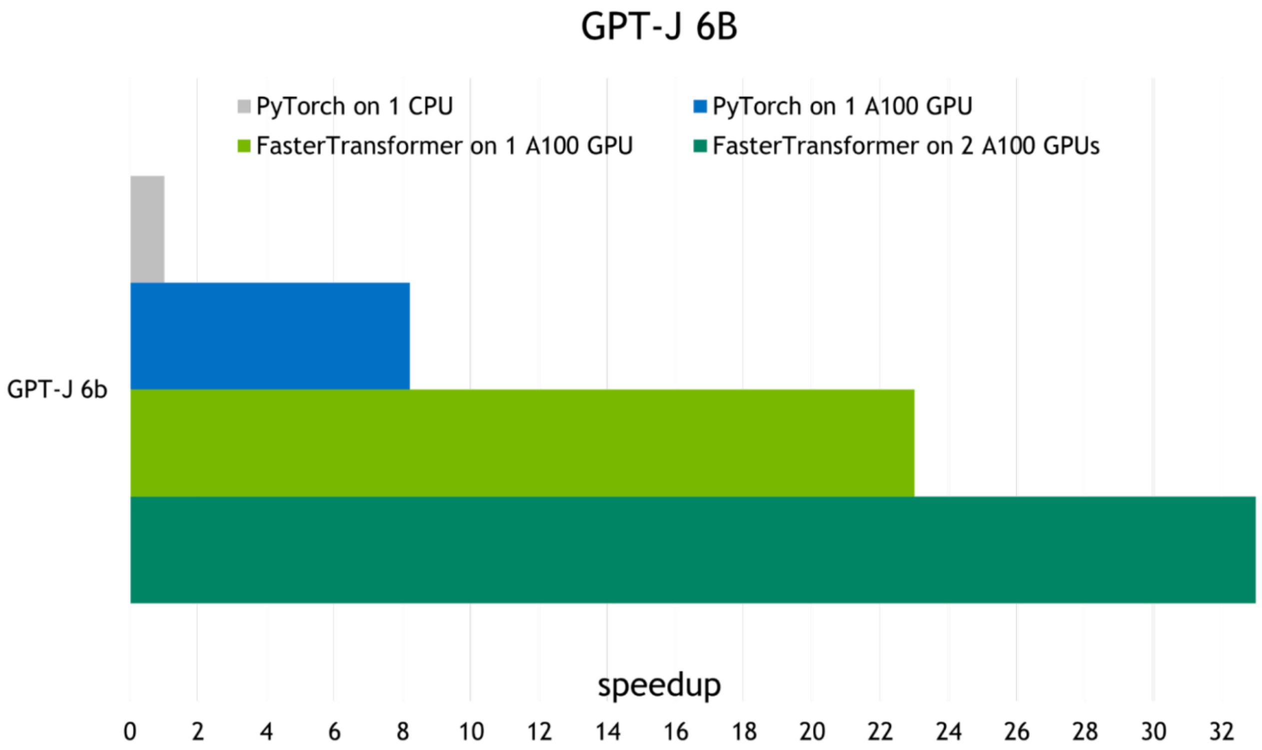
MNMG



DISTRIBUTED INFERENCE WITH FASTERTRANSFORMER

GPT with optimize transformer blocks

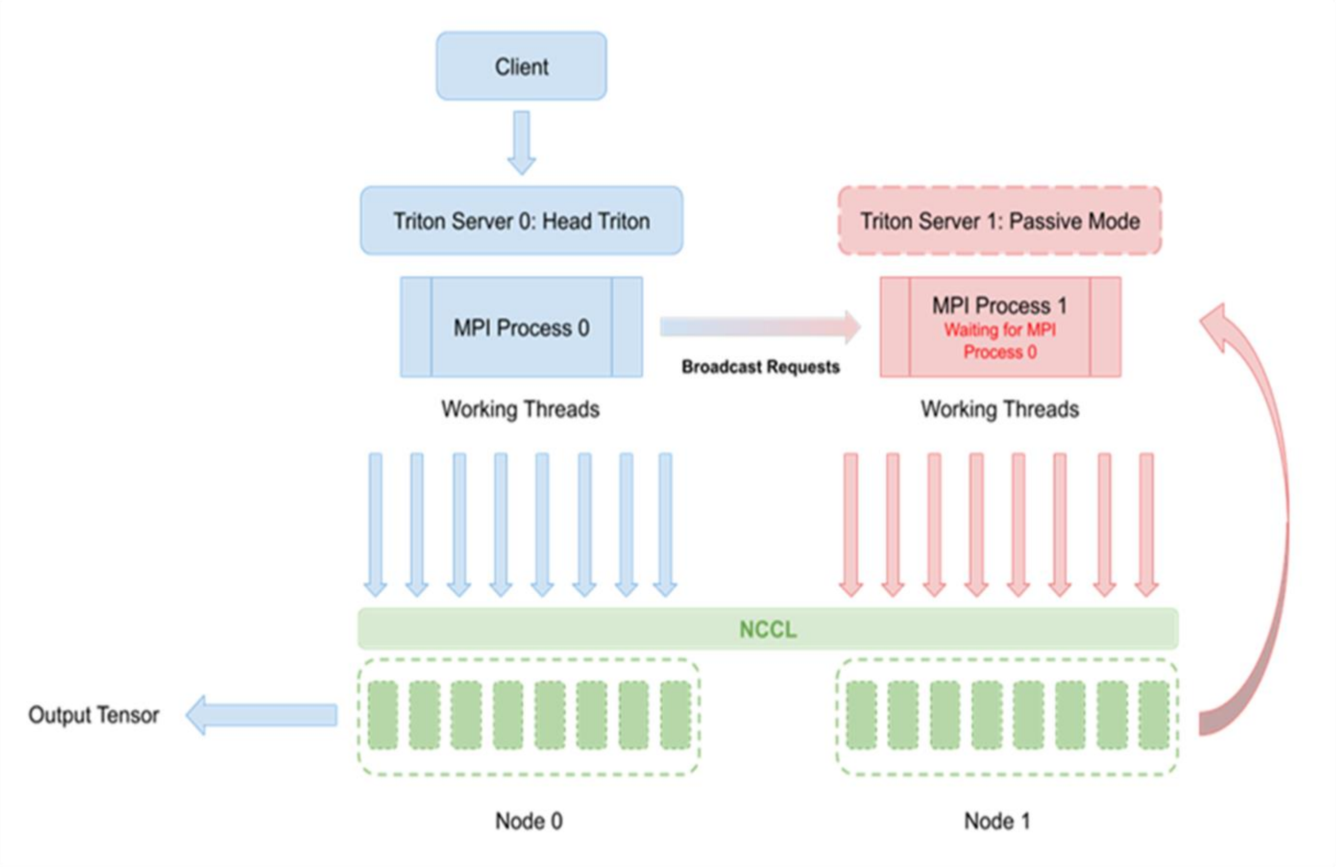




DISTRIBUTED INFERENCE WITH FASTERTRANSFORMER

Serve giant transformer models and accelerate inference

- Optimize kernels to accelerate inference for encoder/decoder layers of transformer models
- Integrated as a backend in Triton Inference Server
- Uses tensor/pipeline parallelism for multi-GPU, multi-node inference
- Uses MPI and NCCL to enable inter/intra node communication
- Supports BERT, GPT, T5, ViT and Swin-T style models
- Megatron, HuggingFace and ONNX converters provided

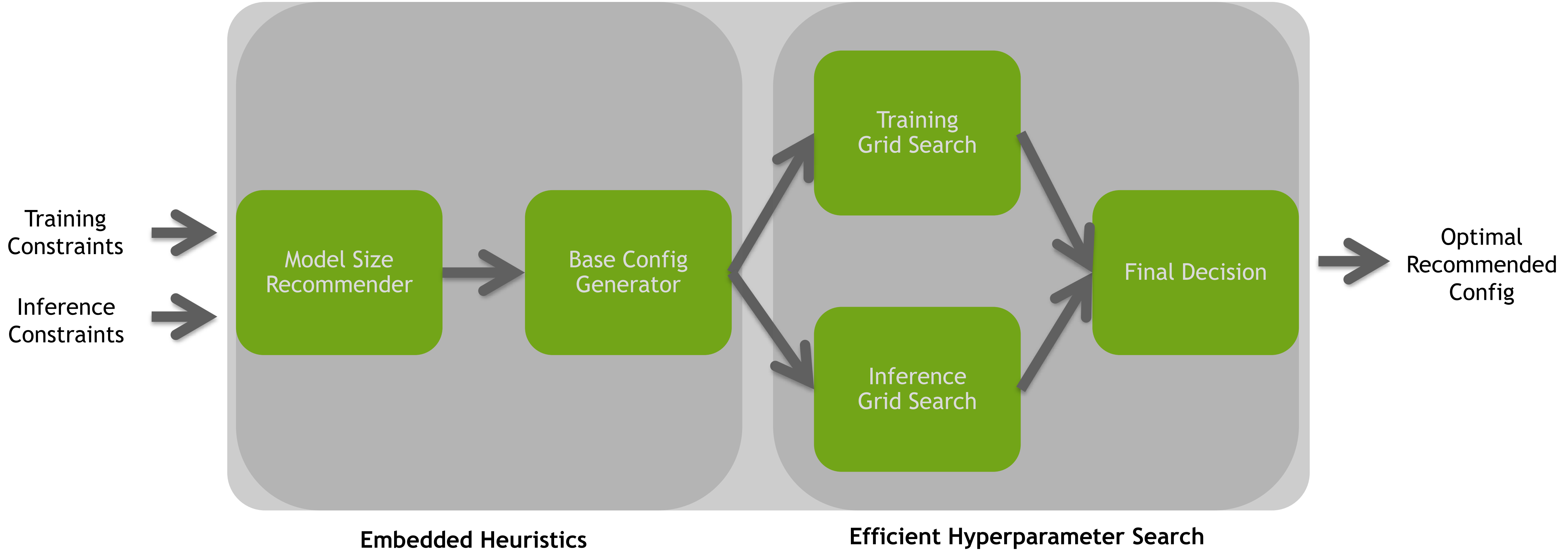


DISTRIBUTED INFERENCE WITH NEMO

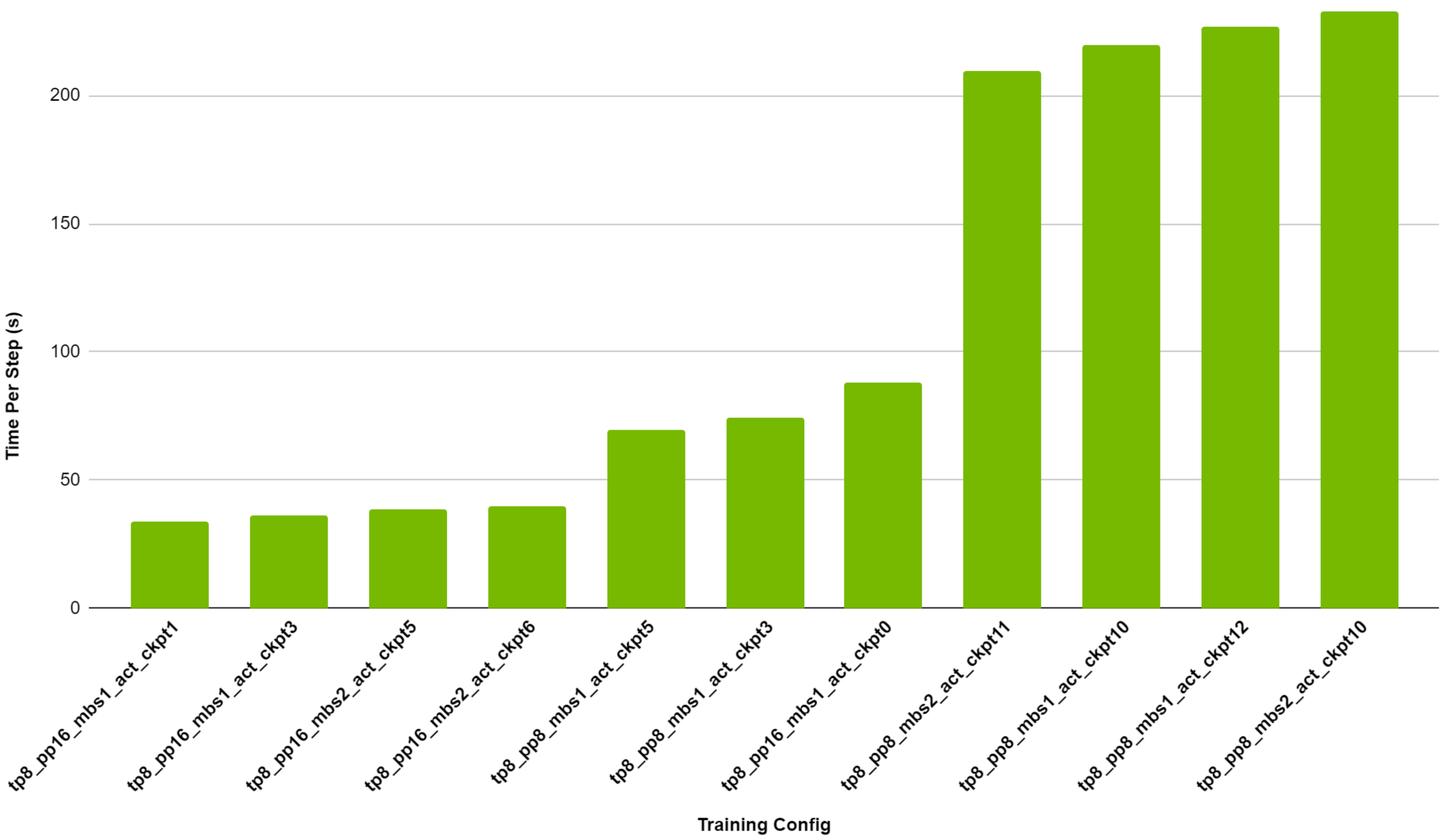
```
python3 FasterTransformer/examples/pytorch/gpt/utils/nemo_ckpt_convert.py \
        --in-file /checkpoints/nemo_gpt1.3B_fp16.nemo \
        --infer-gpu-num 1 \
        --saved-dir /model_repository/gpt3_1.3b \
        --weight-data-type fp16 \
        --load-checkpoints-to-cpu 0
……
python3  /export_scripts/prepare_triton_model_config.py \
        --model-train-name gpt3_1.3b \
        --template-path /opt/bignlp/fastertransformer_backend/all_models/gpt/fastertransformer/config.pbtxt \
        --ft-checkpoint /model_repository/gpt3_1.3b/1-gpu \
        --config-path /model_repository/gpt3_1.3b/config.pbtxt \
        --max-batch-size 256 \
        --pipeline-model-parallel-size 1 \
        --tensor-model-parallel-size 1 \
        --data-type bf16'
```

NEMO HYPERPARAMETER TOOL

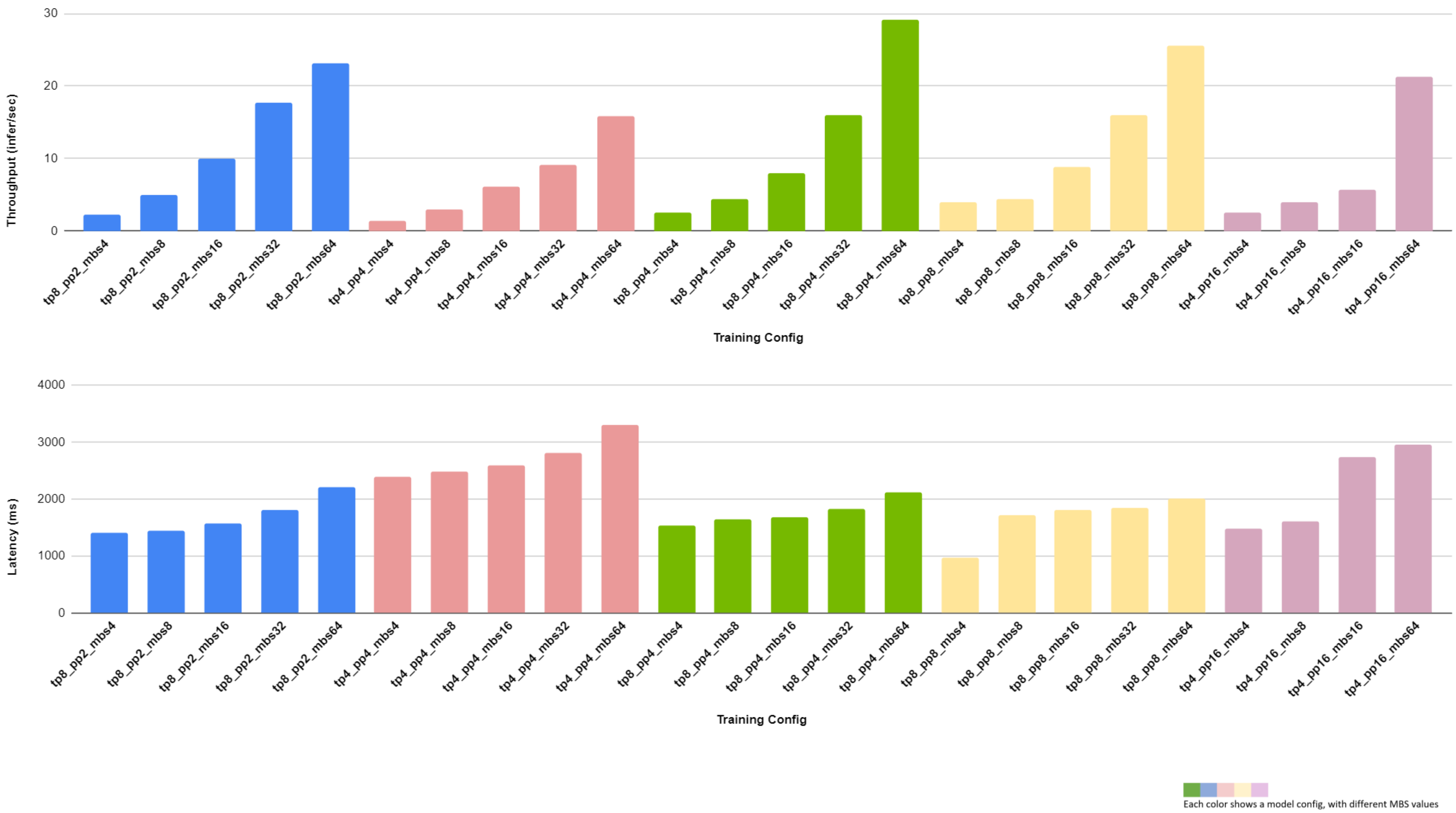
EFFICIENT HYPERPARAMETER SEARCH WITH EMBEDDED HEURISTICS



175B GPT-3 MODEL: 6.85X TRAINING SPEEDUP



INFERENCE 175B GPT-3 MODEL: OPTIMIZING THROUGHPUT AND LATENCY



NVIDIA

QUICK ITERATION FOR FASTER EXPERIMENTATION AND RESEARCH

- Easy to use and flexible
- Multiple models supported: GPT3, T5, mT5
- Decides the model size based on hardware constraints
- Baseline configuration using heuristics for any model size
    - Learning rate, weight initialization, optimizer, weight decay, dropout, data type, global batch size…
- Best training and inference configurations found quickly
- Go from zero to optimal configuration
- Know the inference latency/throughput before train the model

PROMPT LEARNING

LLMS ARE KNOWLEDGEABLE FOR GENERAL QUESTIONS

Zero-Shot

"What is the yellow part in an egg?" →

LLM

→ "This is the part that suspended in the center of the egg."

CUSTOMIZATION IS REQUIRED FOR BUSINESS-SPECIFIC TASKS

"What is the yellow part in an egg?

LLM

Nutrition Chatbot

"The yellow part in an egg is the yolk. It contains fat, cholesterol, and protein."

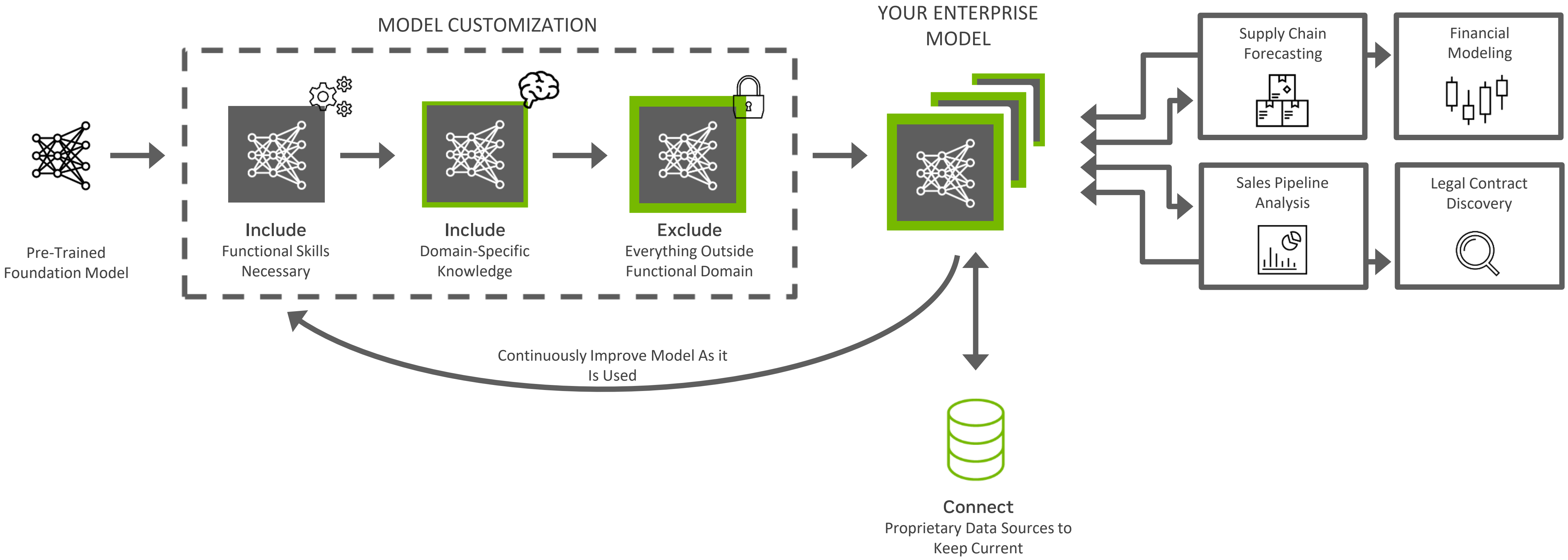
Prenatal Chatbot

"The yellow part in an egg is rich in choline, which is important for fetal brain development"

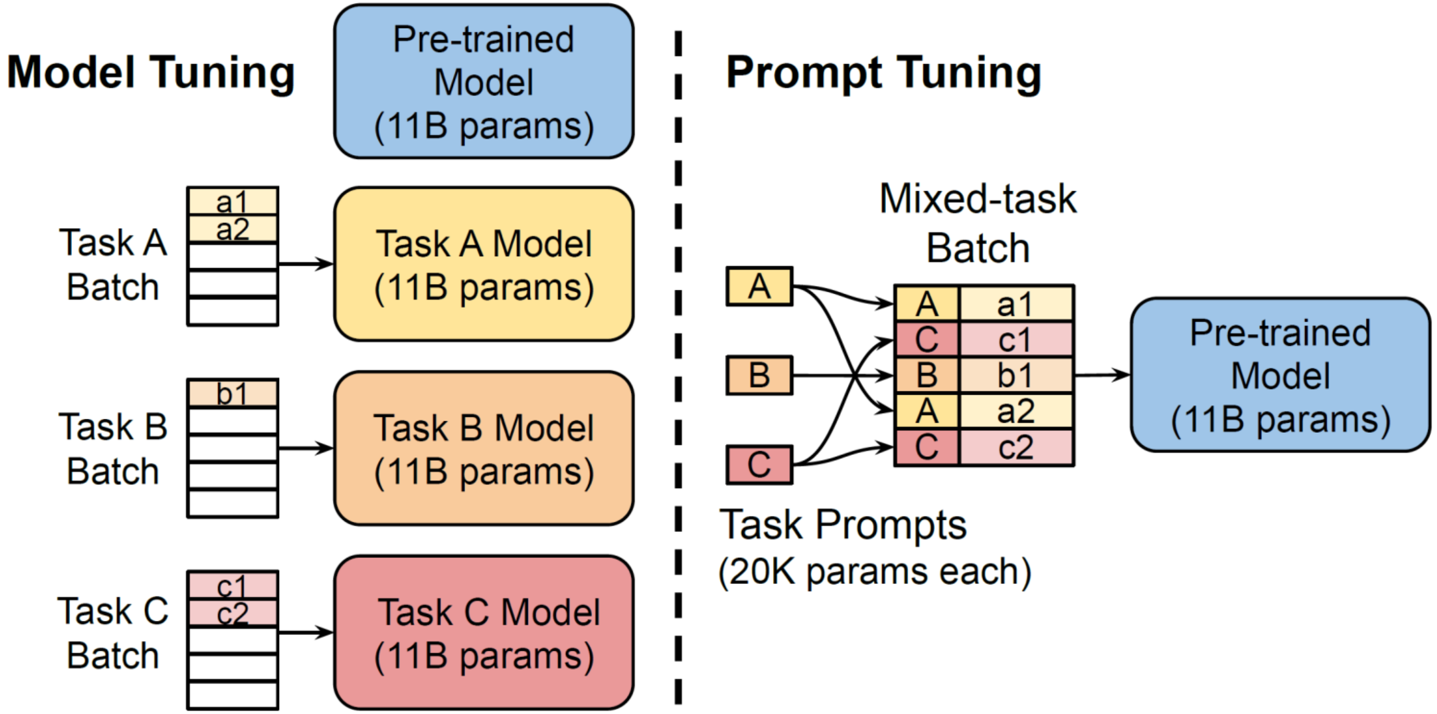
Culinary Chatbot

"The yellow part in an egg is used to fortify sauces and salad dressings, and to emulsify rich, fatty, ingredients like oil and butter"

OVERCOMING CHALLENGES OF USING FOUNDATION MODEL



Prompt tuning



One model, multiple prompts, multiple tasks.

Lester, B., Al-Rfou, R., & Constant, N. (2021). The power of scale for parameter-efficient prompt tuning.

P-tunning

GPT Understands, Too

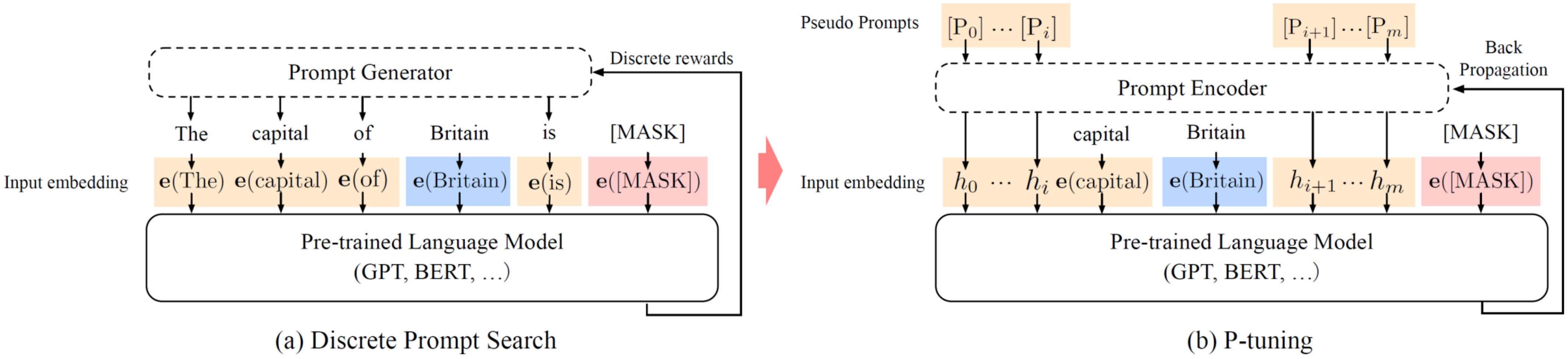
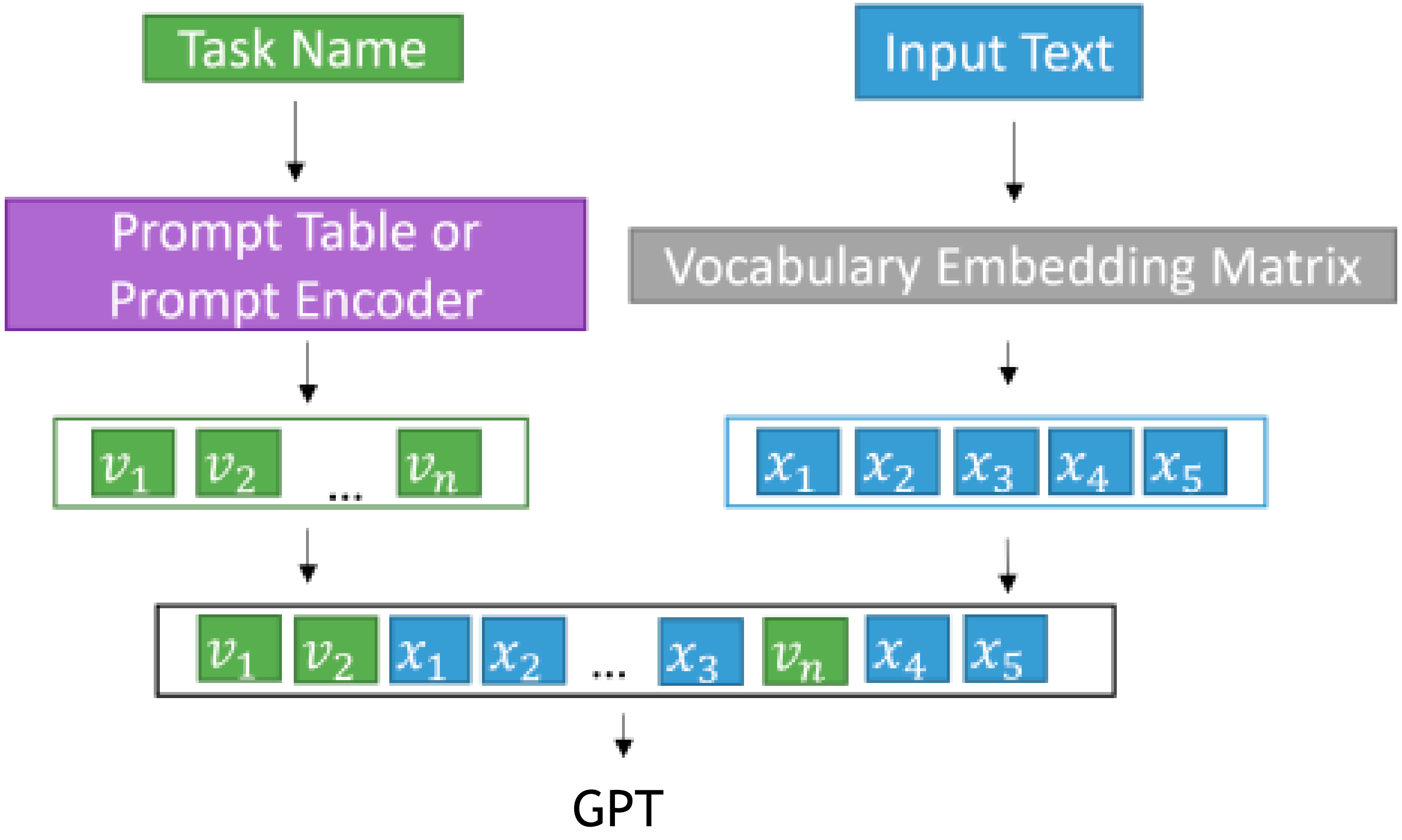


Figure 2. An example of prompt search for "The capital of Britain is [MASK]". Given the context (blue zone, "Britain") and target (red zone, "[MASK]"), the orange zone refer to the prompt tokens. In (a), the prompt generator only receives discrete rewards; on the contrary, in (b) the pseudo prompts and prompt encoder can be optimized in a differentiable way. Sometimes, adding few task-related anchor tokens (such as "capital" in (b)) will bring further improvement.

Prompt Learning with Nemo

Using Both Prompt and P-Tuning



Prompt Learning with Nemo

Example of Prompt Tuning Config

```
language_model_path: models/megatron_125M_gpt.nemo

existing_tasks: []

new_tasks: ["sentiment", "intent_and_slot"]

task_templates:

- taskname: "sentiment"

  prompt_template: "<|VIRTUAL_PROMPT_0|> {sentence} sentiment: {label}"

  total_virtual_tokens: 100

  virtual_token_splits: [100]

  truncate_field: null

  answer_only_loss: False

- taskname: "intent_and_slot"

  prompt_template: "<|VIRTUAL_PROMPT_0|> Predict intent and slot <|VIRTUAL_PROMPT_1|> :\n{utterance}{label}"

  total_virtual_tokens: 100

  virtual_token_splits: [80, 20]

  truncate_field: null

  answer_only_loss: True

  answer_field: "label"
```

```
prompt_tuning:

  new_prompt_init_methods: ["text", "text"]

  new_prompt_init_text: ["financial sentiment analysis ", "intent and slot classification"]
```

Prompt Learning with Nemo

Example of P-Tuning Config

```
model:
 language_model_path: models/megatron_125M_gpt.nemo
  existing_tasks: ["sentiment", "intent_and_slot"]
  new_tasks: ["squad"]
```

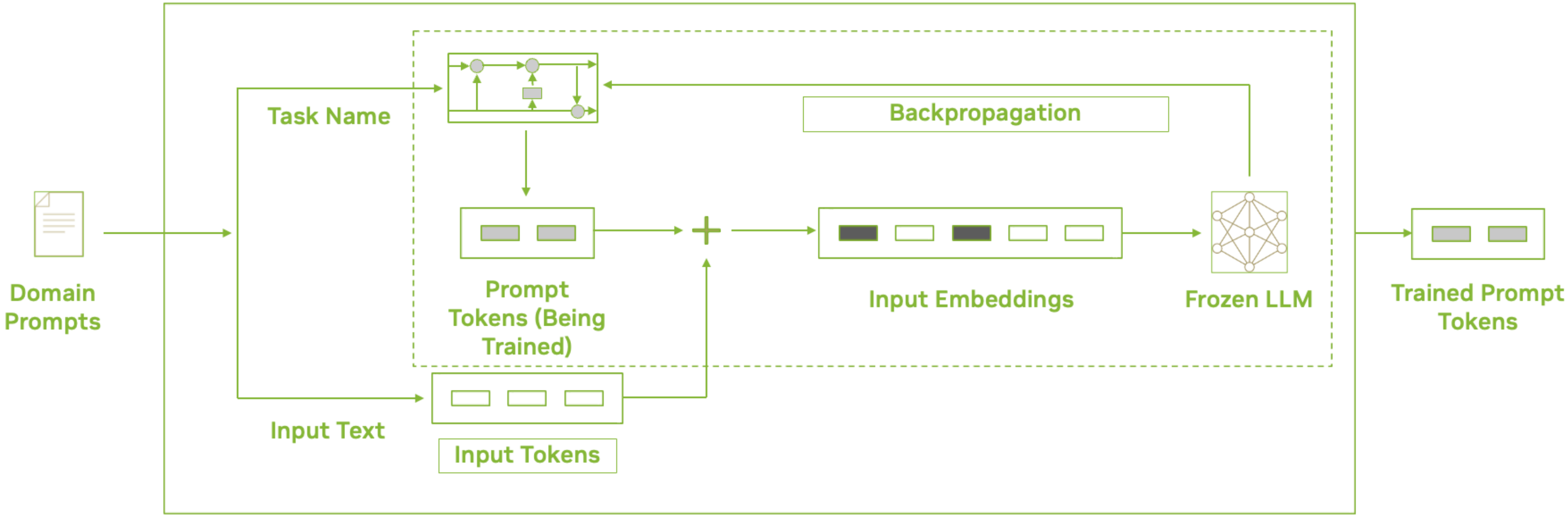
```
p_tuning:
 dropout: 0.0
 num_layers: 2
```

```
task_templates:
  - taskname: "sentiment"
    prompt_template: "<|VIRTUAL_PROMPT_0|> {sentence} sentiment: {label}"
    total_virtual_tokens: 100
    virtual_token_splits: [100]
    truncate_field: nulltruncate_field: context
    answer_only_loss: False
```

```
- taskname: "squad“
prompt_template: "<|VIRTUAL_PROMPT_0|> Answer the question
from the context {question} {context} Answer: {answer}“
total_virtual_tokens: 9
virtual_token_splits: [9]
answer_only_loss: True
answer_field: "answer"
```

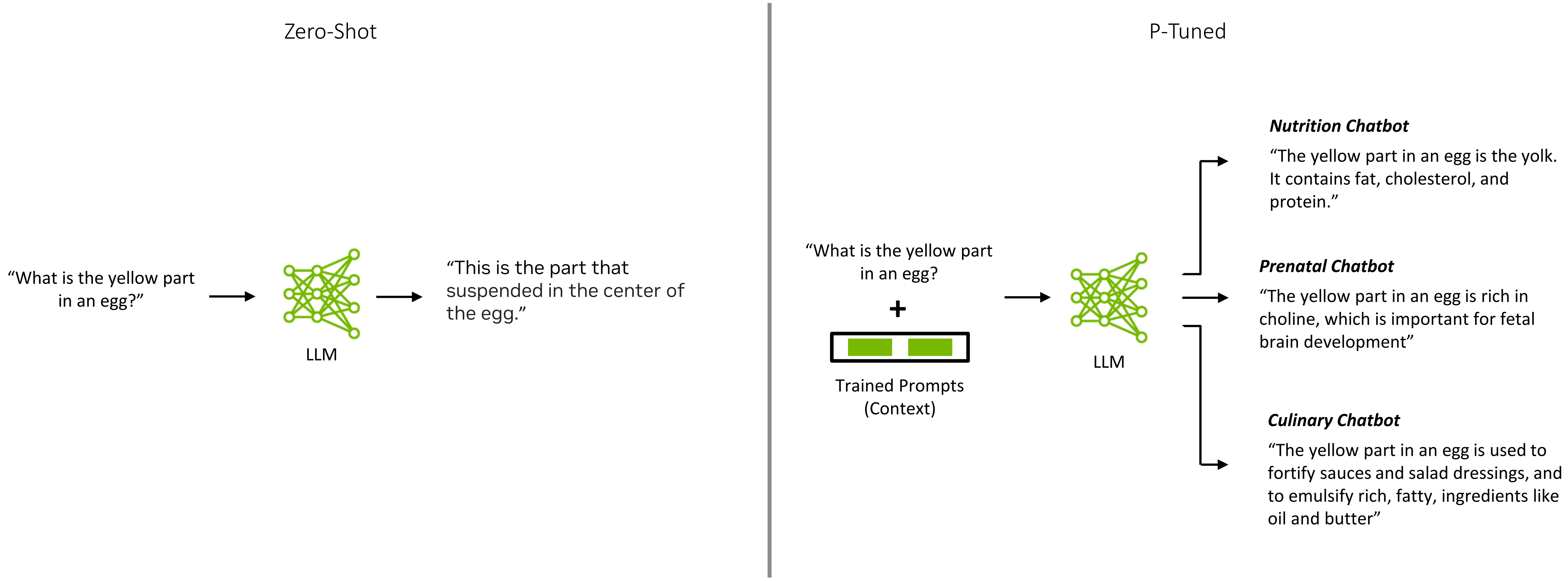
```
  - taskname: "intent_and_slot"
    prompt_template: "<|VIRTUAL_PROMPT_0|> Predict intent and slot <|VIRTUAL_PROMPT_1|>
:\n{utterance}{label}"
    total_virtual_tokens: 100
    virtual_token_splits: [80, 20]
    truncate_field: null
    answer_only_loss: True
    answer_field: "label"
```

PROMPT LEARNING WITH NEMO



- Freeze foundational model, and learn the prompt tokens using a supervised learning approach
- Get high accuracy for specific use-cases with just 100s of samples

CUSTOMIZATION IS REQUIRED FOR BUSINESS-SPECIFIC TASKS



Resources

- DEVBLOGS
- Adapting P-Tuning to Solve Non-English Downstream Tasks
- How to Create a Custom Language Model

- TUTORIALS
- Prompt Learning
- Multitask_Prompt_and_Ptuning

- GTC sessions
- Efficient At-Scale Training and Deployment of Large Language Models – GTC Session
- Hyperparameter Tool GTC Session

- Register here
- Find out more here
- NVIDIA Brings Large Language AI Models to Enterprises Worldwide | NVIDIA Newsroom

DEVBLOGS and VIDEOS:

- Adapting P-Tuning to Solve Non-English Downstream Tasks
- NVIDIA AI Platform Delivers Big Gains for Large Language Models
- Efficient At-Scale Training and Deployment of Large Language Models – GTC Session
- Hyperparameter Tool GTC Session
- Using DeepSpeed and Megatron to Train Megatron-Turing NLG 530B, the World's Largest and Most Powerful Generative Language Model | NVIDIA Developer Blog

CUSTOMER STORIES:
The King's Swedish: AI Rewrites the Book in Scandinavia
eBook Asset

No Hang Ups With Hangul: KT Trains Smart Speakers, Customer Call Centers With NVIDIA AI

Resources

Get Started

Customers Using NeMo Framework Today

Korean Language Models Powering:

1. AI Contact Center - Cloud-based solution handling 100K calls/day without human intervention, reducing consultation times by 15 seconds.
2. Providing home assistant functions through IPTV, serving 8 Million families

AI SWEDEN

Accelerated NLP industry applications in Sweden by making the power of a 100-billion-parameter model for Nordic languages easily accessible to the Nordic ecosystem.

Improved downstream NLP tasks, like sentiment analysis, dialogue, and translation, by training custom Large Language Models using NeMo framework.