초거대 언어모델 분산학습

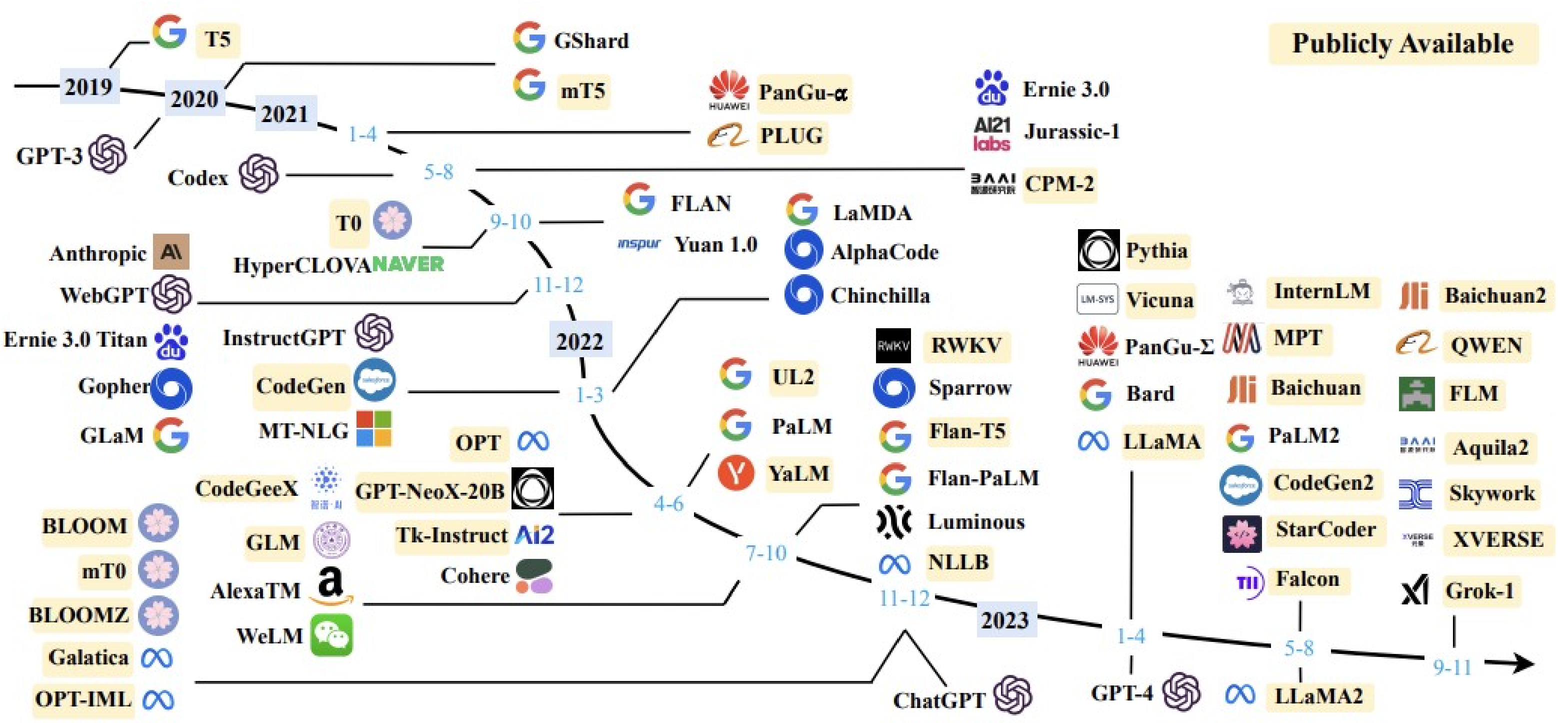
구건모
2024.

목차

- **Introduction**
    - Large Language Model (LLM)
    - What is LLMOps
    - When Do We Need to Train LLM
    - Challenges in Training LLM
- **Distributed Training**
    - Tensor Parallelism
    - Pipeline Parallelism
    - Data Parallelism
    - Sharded Data Parallelism

Large Language Model (LLM)

- 크다는 것은 상대적이고 시간이 지나면서 절대적인 기준이 달라짐
- 2018 년에는 BERT(340M) [1] 도 large language model 이었음

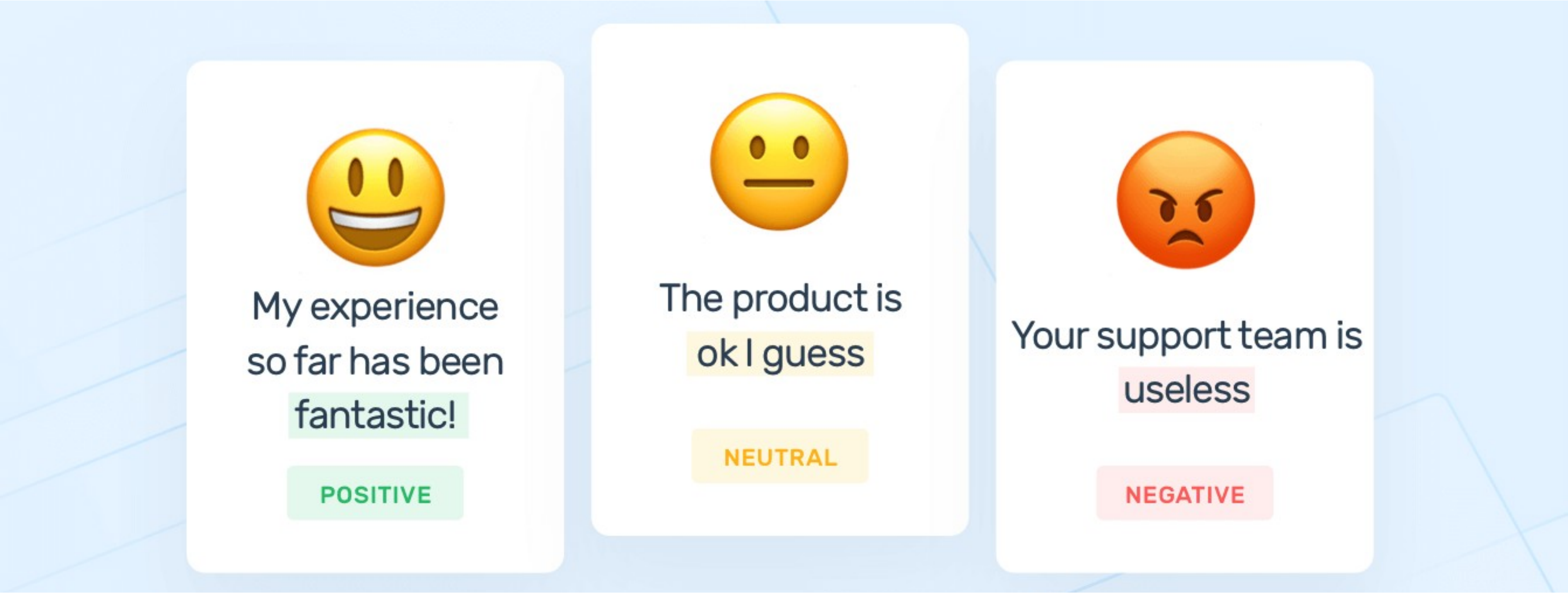


A timeline of existing large language models (having a size larger than 10B) in recent years [2].

[1] Devlin et al., “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”, ACL 2019
[2] Zhao et al., “A survey of Large Language Models”

Large Language Model (LLM)

- LLM 이전에는 학습 데이터를 만들 때 사람의 노력이 많이 필요했음
  - Sentiment analysis: 문장이 주어졌을 때 긍정 / 중립 / 부정인지 사람이 분류



My experience so far has been fantastic!

The product is ok I guess

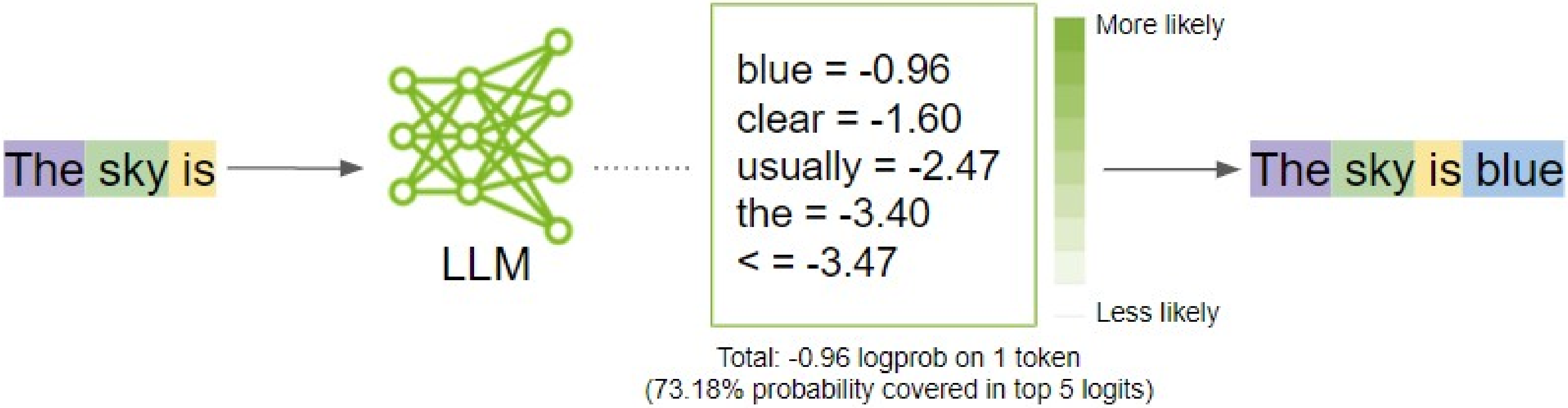
Your support team is useless

Sentiment analysis [1].

[1] https://monkeylearn.com/sentiment-analysis/

Large Language Model (LLM)

- LLM 은 human labeling 이 필요 없는 한 가지 task 를 학습해서 놀라울 정도로 다양한 task 를 수행할 수 있음
  - Next token prediction: 주어진 토큰들의 바로 다음에 나타날 토큰 예측
  - Raw 텍스트 데이터에서 학습 데이터를 만들 수 있음
    - The sky is blue: (The → sky), (The sky → is), (The sky is → blue)
    - 인터넷에 모델이 학습 할 수 있는 초대규모 텍스트 데이터 존재

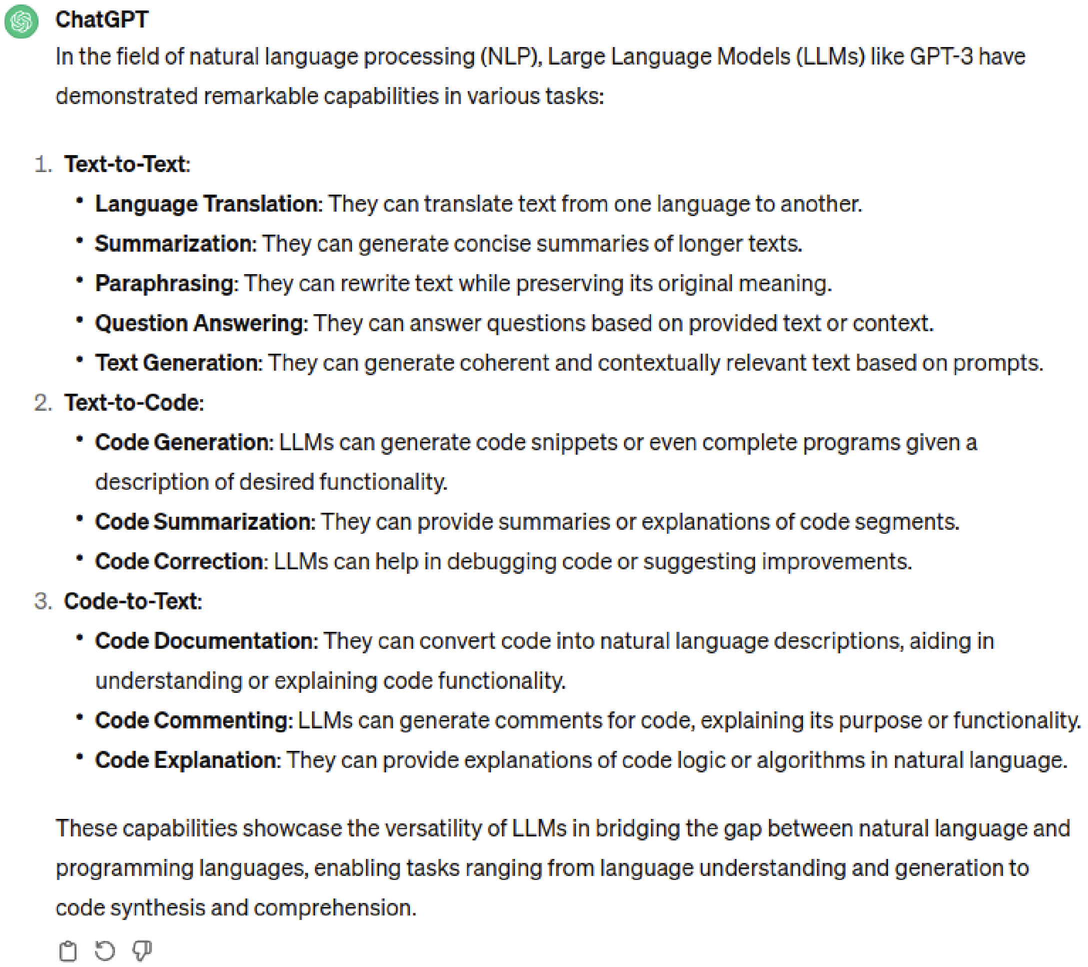


General working flow of an LLM predicting the next word [1].

[1] https://developer.nvidia.com/blog/how-to-get-better-outputs-from-your-large-language-model/

Large Language Model (LLM)

- 하나의 모델로 다양한 task 를 할 수 있음



ChatGPT

In the field of natural language processing (NLP), Large Language Models (LLMs) like GPT-3 have demonstrated remarkable capabilities in various tasks:

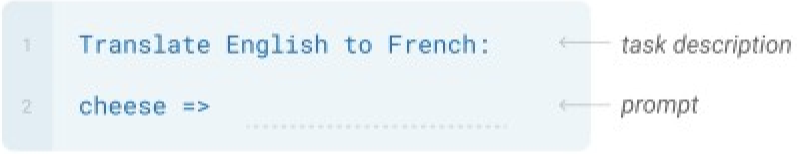
1. **Text-to-Text**:
   - **Language Translation**: They can translate text from one language to another.
   - **Summarization**: They can generate concise summaries of longer texts.
   - **Paraphrasing**: They can rewrite text while preserving its original meaning.
   - **Question Answering**: They can answer questions based on provided text or context.
   - **Text Generation**: They can generate coherent and contextually relevant text based on prompts.
2. **Text-to-Code**:
   - **Code Generation**: LLMs can generate code snippets or even complete programs given a description of desired functionality.
   - **Code Summarization**: They can provide summaries or explanations of code segments.
   - **Code Correction**: LLMs can help in debugging code or suggesting improvements.
3. **Code-to-Text**:
   - **Code Documentation**: They can convert code into natural language descriptions, aiding in understanding or explaining code functionality.
   - **Code Commenting**: LLMs can generate comments for code, explaining its purpose or functionality.
   - **Code Explanation**: They can provide explanations of code logic or algorithms in natural language.

These capabilities showcase the versatility of LLMs in bridging the gap between natural language and programming languages, enabling tasks ranging from language understanding and generation to code synthesis and comprehension.

Large Language Model (LLM)

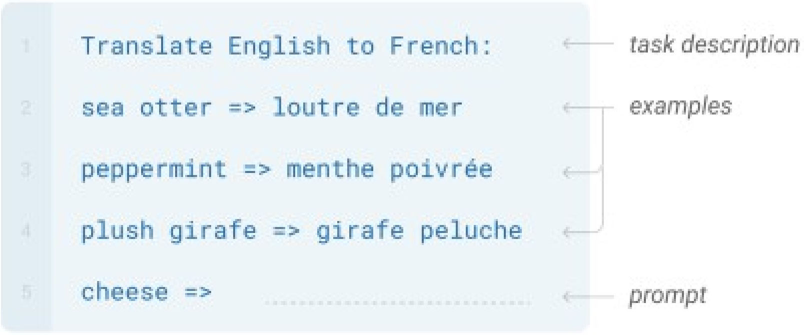
- In-context learning 을 통해 다양한 task 를 수행할 수 있음
- Next token prediction 으로 학습된 모델은 성능이 크기에 비례함

Zero-shot

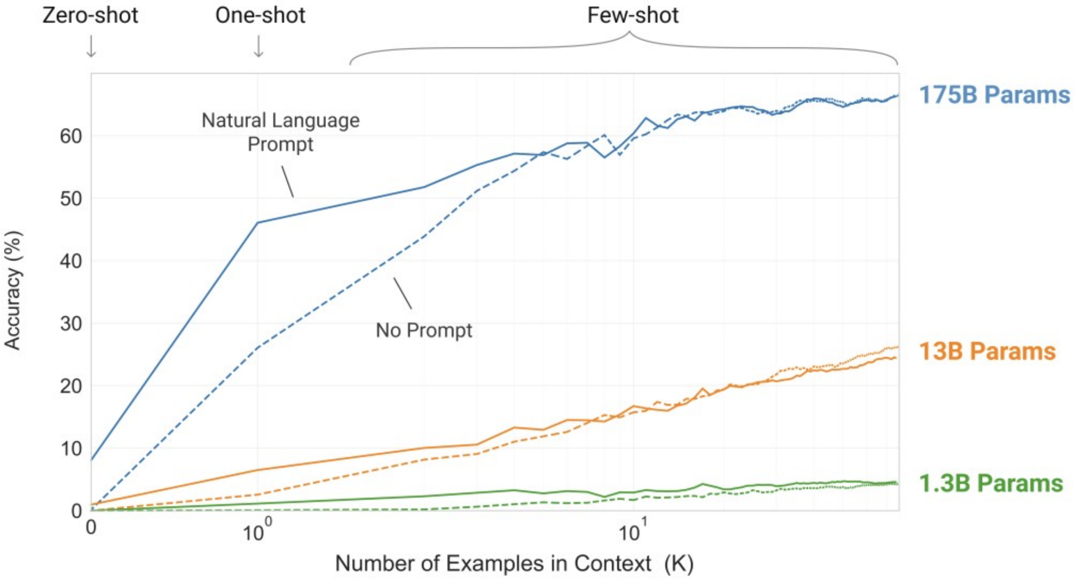


```
Translate English to French:
cheese =>
```

Few-shot



```
Translate English to French:
sea otter => loutre de mer
peppermint => menthe poivrée
plush girafe => girafe peluche
cheese =>
```

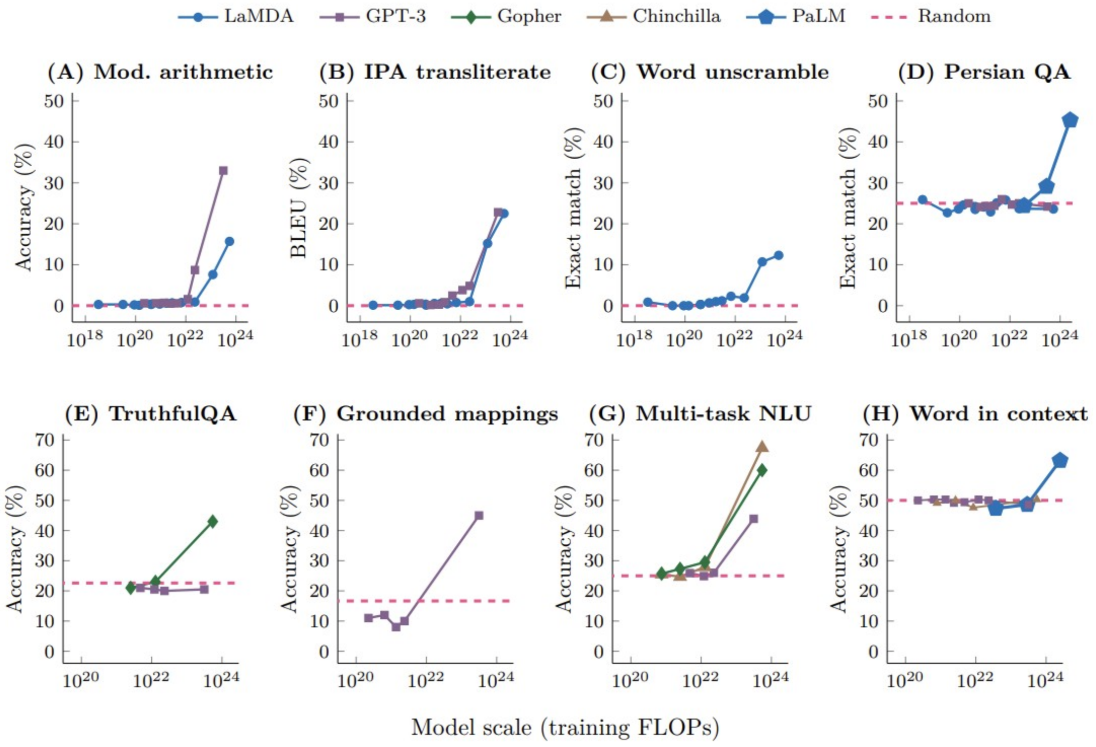


Larger models make increasingly efficient use of in-context information [1].

[1] Brown et al., “Language Models are Few-Short Learners”

Large Language Model (LLM)

- Emergent ability 에 대한 환상
- 모델 크기를 늘리면 특별한 능력이 생겨서 성능이 갑자기 좋아진다 ?

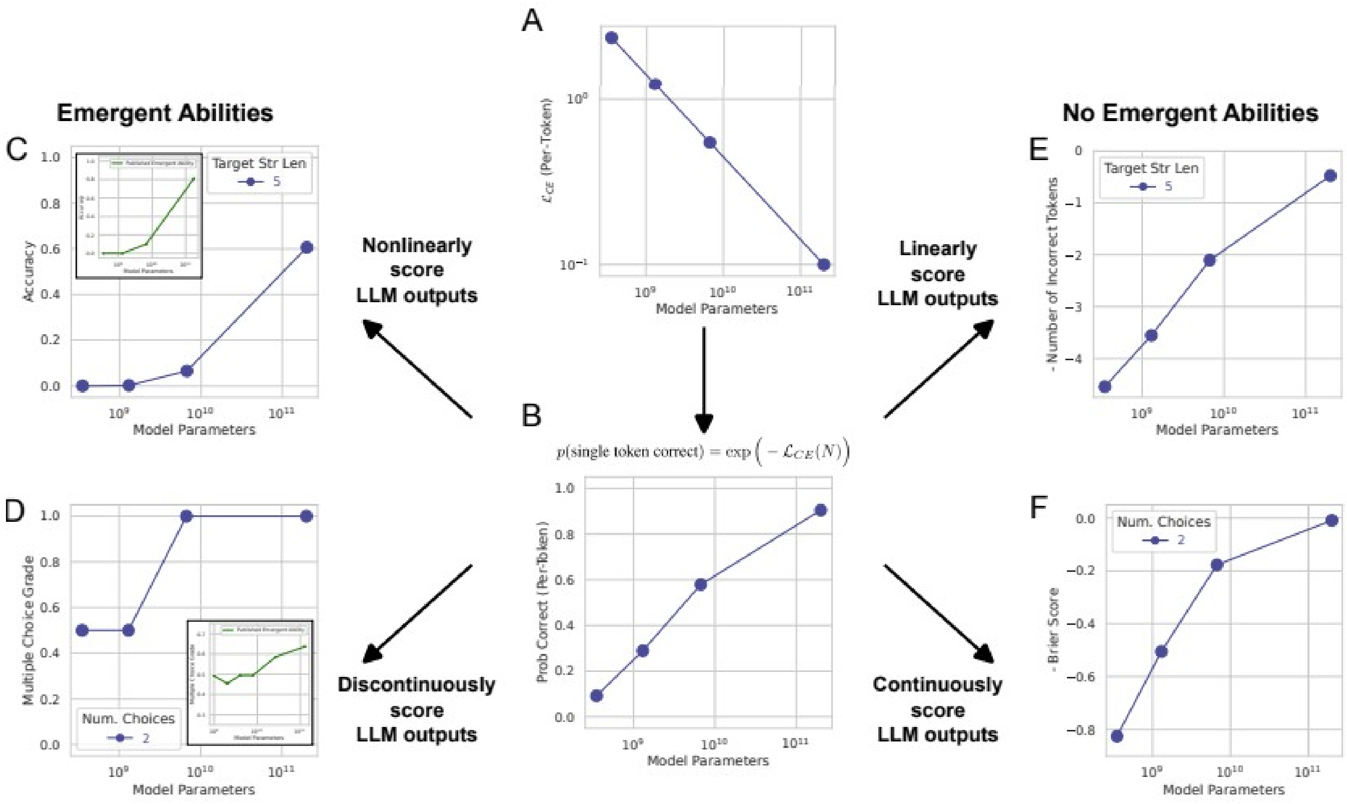


Eight examples of emergence in the few-shot prompting setting [1].

[1] Wei et al., “Emergent Abilities of Large Language Models”, TMLR 2022

Large Language Model (LLM)

- 성능이 급격하게 변하는 것은 평가지표가 불연속적 / 비선형이기 때문

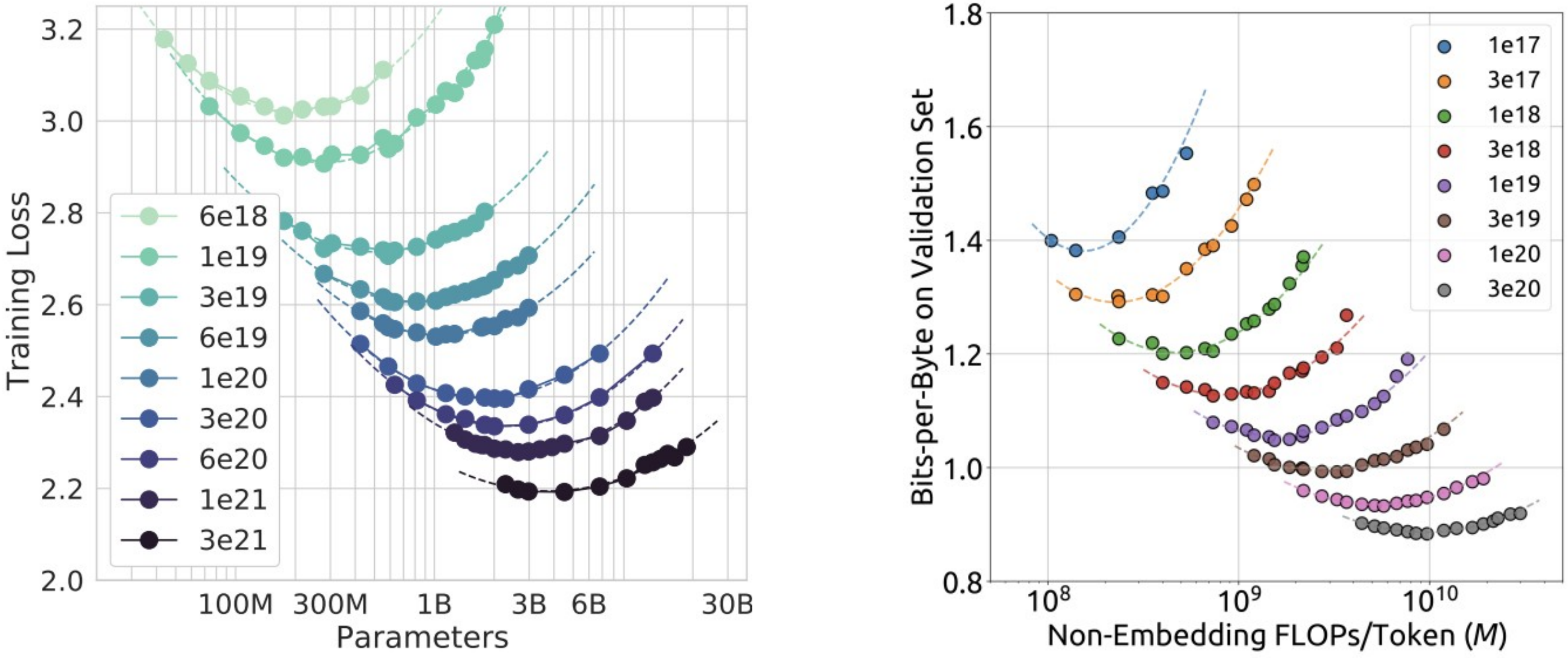


Emergent abilities of large language models are created by the researcher’s chosen metrics, not unpredictable changes in model behavior with scale [1].

[1] Schaeffer, Miranda, and Koyejo, “Are Emergent Abilities of Large Language Models a Mirage?”, NeurIPS 2023

Large Language Model (LLM)

- 여전히 모델의 크기가 커지면 성능이 높아지는 것은 사실임
- 모델과 데이터의 크기에 따라 성능을 어느정도 예측할 수 있음



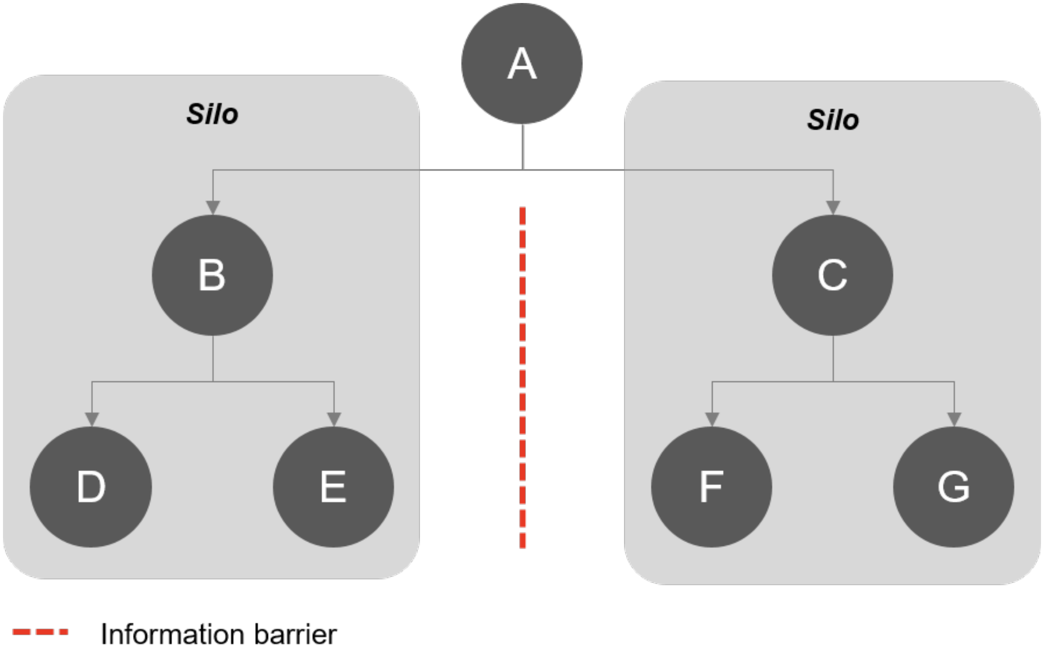
IsoFLOP surve of Chinchilla (left) [1] and DeepSeek LLM (right) [2].

[1] Hoffmann et al., “Training Compute-Optimal Large Language Models”, NeurIPS 2022
[2] Bi et al., “DeepSeek LLM: Scaling Open-Source Language Models with Longtermism”

What is LLMOps

Information Silo

- 곡물 저장 창고처럼 각 조직에 정보가 머무르고 공유되지 않는 현상



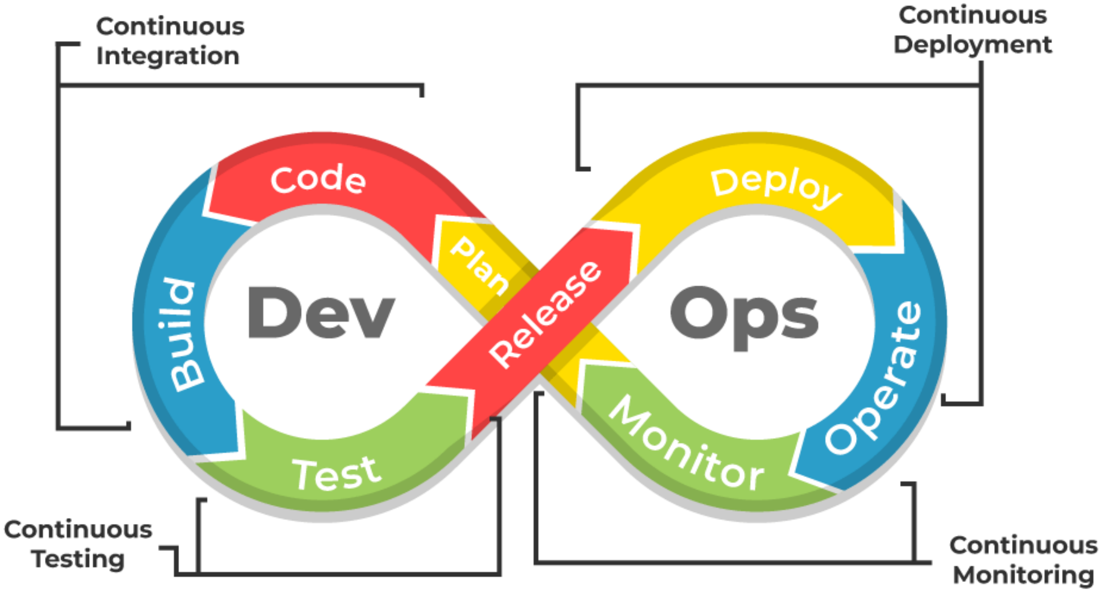
Grain silo (left) [1] and information silo (right) [2].

[1] https://www.moylangrainsilos.com/
[2] https://en.wikipedia.org/wiki/Information_silo

What is LLMOps

DevOps

- Development and operations
- 개발과 운영을 자동화하기 위한 다양한 노력을 포함하는 광범위한 개념
- 한 번 배포한 후에도 지속적으로 개발 및 재배포가 필요함



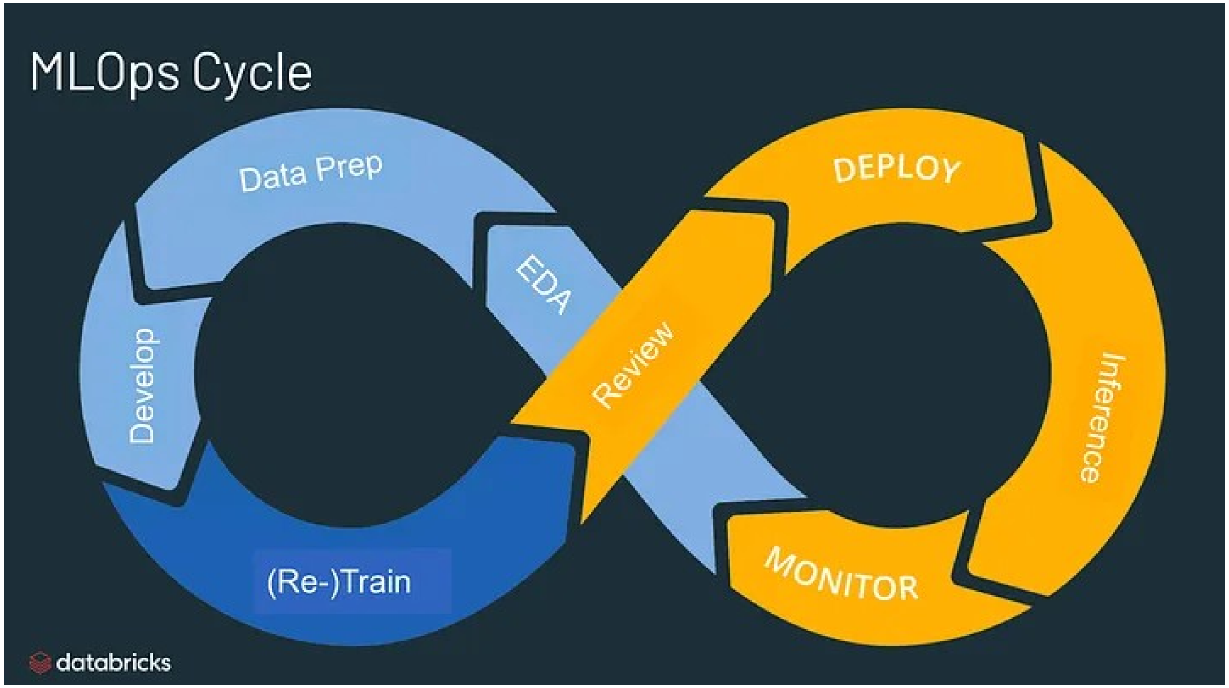
DevOps Tutorial [1].

[1] https://www.geeksforgeeks.org/devops-tutorial/

What is LLMOps

MLOps

- Machine learning operations
- 모델을 학습한 후 시간이 지나면 재학습 필요



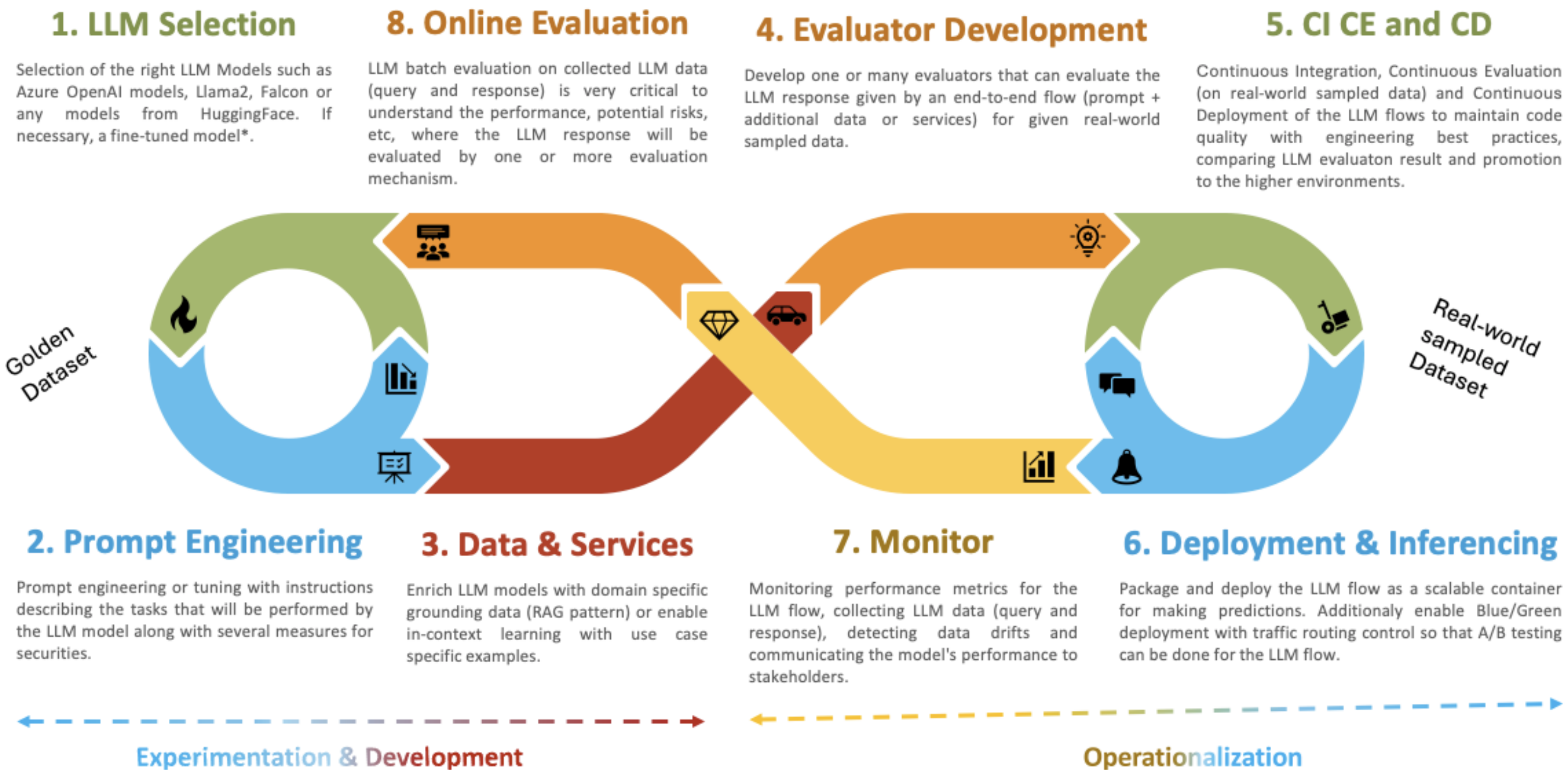
MLOps cycle [1].

[1] https://www.databricks.com/glossary/mlops

What is LLMOps

LLMOps

- MLOps for LLM



1. LLM Selection

Selection of the right LLM Models such as Azure OpenAI models, Llama2, Falcon or any models from HuggingFace. If necessary, a fine-tuned model*.

8. Online Evaluation

LLM batch evaluation on collected LLM data (query and response) is very critical to understand the performance, potential risks, etc, where the LLM response will be evaluated by one or more evaluation mechanism.

4. Evaluator Development

Develop one or many evaluators that can evaluate the LLM response given by an end-to-end flow (prompt + additional data or services) for given real-world sampled data.

5. CI CE and CD

Continuous Integration, Continuous Evaluation (on real-world sampled data) and Continuous Deployment of the LLM flows to maintain code quality with engineering best practices, comparing LLM evaluaton result and promotion to the higher environments.

2. Prompt Engineering

Prompt engineering or tuning with instructions describing the tasks that will be performed by the LLM model along with several measures for securities.

3. Data & Services

Enrich LLM models with domain specific grounding data (RAG pattern) or enable in-context learning with use case specific examples.

7. Monitor

Monitoring performance metrics for the LLM flow, collecting LLM data (query and response), detecting data drifts and communicating the model's performance to stakeholders.

6. Deployment & Inferencing

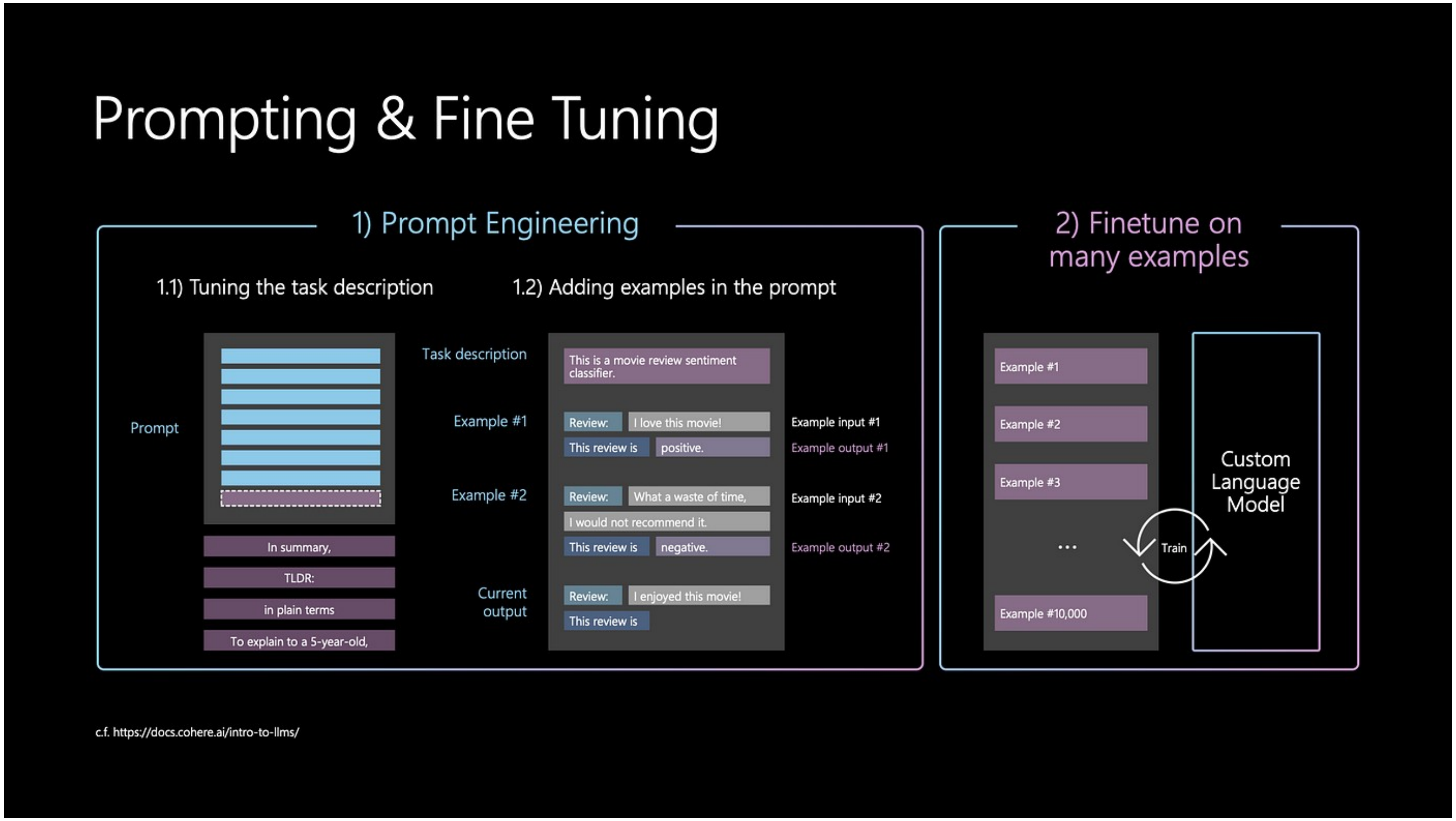
Package and deploy the LLM flow as a scalable container for making predictions. Additionaly enable Blue/Green deployment with traffic routing control so that A/B testing can be done for the LLM flow.

LLMOps loop diagram [1]

[1] https://www.linkedin.com/pulse/llmops-approach-early-thoughts-prabal-deb-26eqc/

When Do We Need to Train LLM

- 비용이 적게드는 방법부터 시도하는 것이 좋음



Prompting & Fine Tuning

1) Prompt Engineering

1.1) Tuning the task description

Prompt

In summary,

TLDR:

in plain terms

To explain to a 5-year-old,

1.2) Adding examples in the prompt

Task description	This is a movie review sentiment classifier.	
Example #1	Review: I love this movie!	Example input #1
	This review is positive.	Example output #1
Example #2	Review: What a waste of time, I would not recommend it.	Example input #2
	This review is negative.	Example output #2
Current output	Review: I enjoyed this movie!	
	This review is	

2) Finetune on many examples

Example #1

Example #2

Example #3

...

Example #10,000

Train

Custom Language Model

c.f. https://docs.cohere.ai/intro-to-llms/

In-context learning and fine-tuning [1]

[1] https://rpradeepmenon.medium.com/mastering-generative-ai-interactions-a-guide-to-in-context-learning-and-fine-tuning-9ee620c76246

Challenges in Training LLM

- 메모리 [1]
  - 2 bytes per parameter for the weights
  - 4 bytes per parameter for optimizer state (Adam)
  - 2 bytes per parameter for gradients
  - 7B model = 8 * 7e9 = 56e9 = 52GiB
- 시간 [1]
  - FLOPs = 6 * N * D, where N is #params and D is data size
  - D = 20 * N [2]
  - 7B model = 6 * 7e9 * 20 * 7e9 = 5.88e21
  - A100 FLOP/s = 312e12
  - Time = 5.88e21 / 312e12 = 1.88e7 sec = 218 days

[1] Weights & Biases, Training and Fine-tuning Large Language Models (LLMs)
[2] Hoffmann et al., “Training Compute-Optimal Large Language Models”, NeurIPS 2022

Distributed Training

Matrix Multiplication

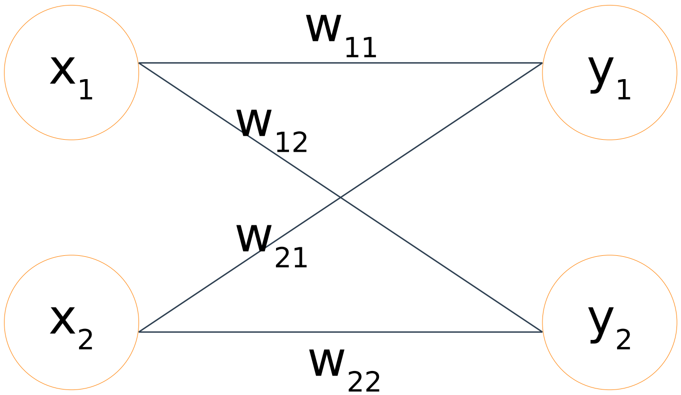
- 행렬 A의 i번째 행의 원소들과 행렬 B의 j번째 열의 원소들을 짝지어 곱한 후 더한 값이 AB의 i행 j열의 원소가 됨

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \times \begin{bmatrix} x & y \\ z & w \end{bmatrix} = \begin{bmatrix} ax + bz & ay + bw \\ cx + dz & cy + dw \end{bmatrix}$$

Distributed Training

Linear Layer

- 입력의 모든 뉴런과 출력의 모든 뉴런이 가중치를 가진 간선으로 연결됨
- 출력 뉴런의 값은 입력 뉴런과 가중치의 행렬곱으로 표현 가능



$= x_1 * w_{11} + x_2 * w_{21}$

$= x_1 * w_{12} + x_2 * w_{22}$

$$\begin{bmatrix} x_1 & x_2 \end{bmatrix} \times \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{bmatrix} = \begin{bmatrix} x_1 * w_{11} + x_2 * w_{21} \\ x_1 * w_{12} + x_2 * w_{22} \end{bmatrix}$$

Distributed Training

Tensor Parallelism

- Tensor 는 matrix 를 차원에 대해 일반화한 것
- 각 GPU 가 matrix multiplication 을 나누어 계산함

Single GPU

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \times \begin{bmatrix} x & y \\ z & w \end{bmatrix} = \begin{bmatrix} ax + bz & ay + bw \\ cx + dz & cy + dw \end{bmatrix}$$

Multiple GPUs

GPU 0

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \times \begin{bmatrix} x \\ z \end{bmatrix} = \begin{bmatrix} ax + bz \\ cx + dz \end{bmatrix}$$

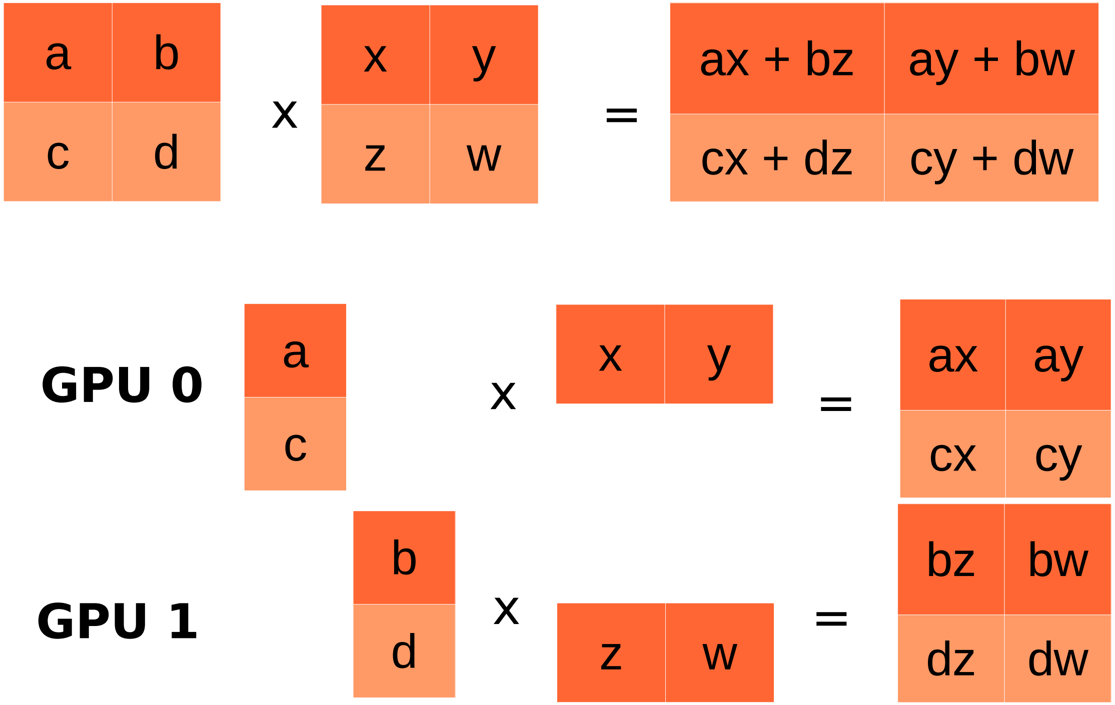
GPU 1

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \times \begin{bmatrix} y \\ w \end{bmatrix} = \begin{bmatrix} ay + bw \\ cy + dw \end{bmatrix}$$

Distributed Training

Tensor Parallelism

- Tensor 는 matrix 를 차원에 대해 일반화한 것
- 각 GPU 가 matrix multiplication 을 나누어 계산함



Single GPU

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} x \begin{bmatrix} x & y \\ z & w \end{bmatrix} = \begin{bmatrix} ax + bz & ay + bw \\ cx + dz & cy + dw \end{bmatrix}$$

Multiple GPUs

GPU 0

$$\begin{bmatrix} a \\ c \end{bmatrix} x \begin{bmatrix} x & y \end{bmatrix} = \begin{bmatrix} ax & ay \\ cx & cy \end{bmatrix}$$

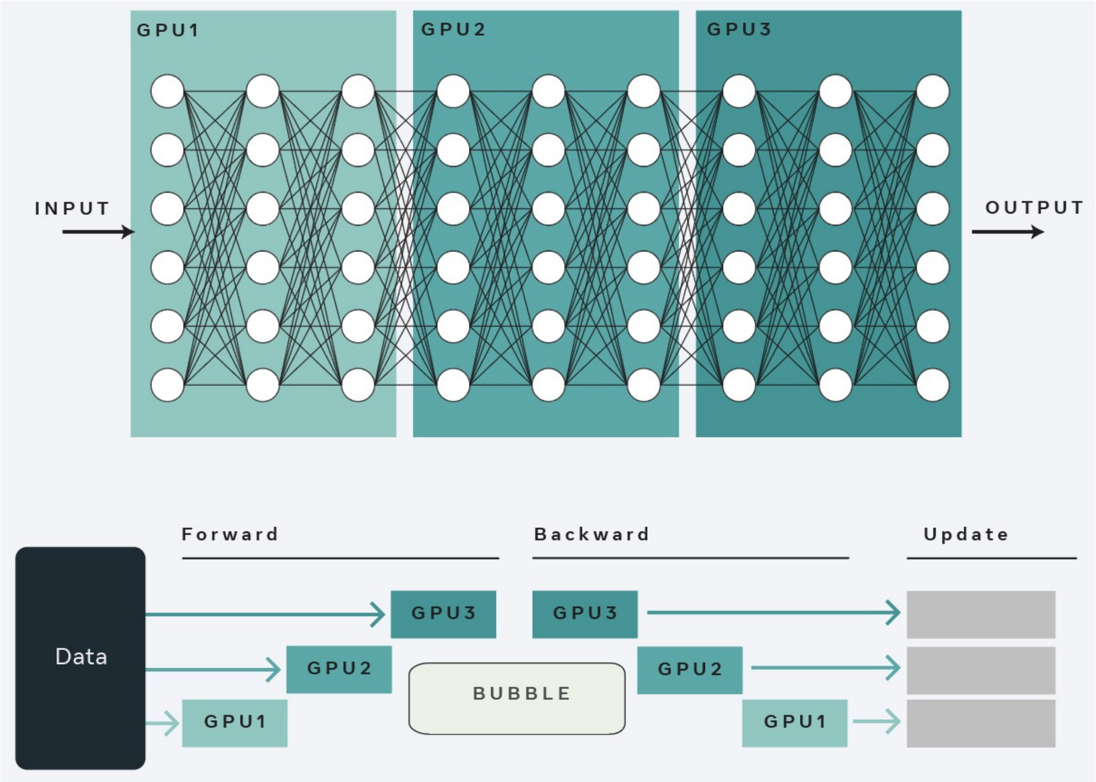
GPU 1

$$\begin{bmatrix} b \\ d \end{bmatrix} x \begin{bmatrix} z & w \end{bmatrix} = \begin{bmatrix} bz & bw \\ dz & dw \end{bmatrix}$$

Distributed Training

Pipeline Parallelism

- 한 단계의 출력이 다음 단계의 입력으로 이어지는 형태의 구조



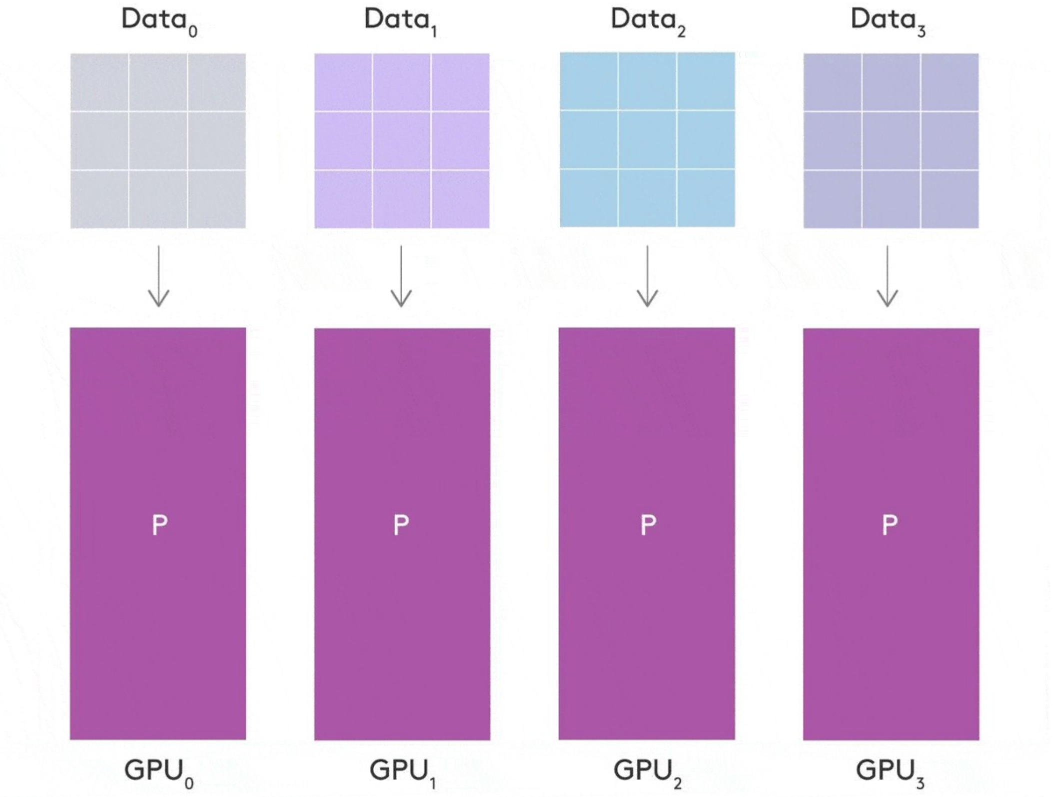
Pipeline parallelism [1]

[1] https://fairscale.readthedocs.io/en/latest/deep_dive/pipeline_parallelism.html

Distributed Training

Data Parallelism

- 각 GPU 에 전체 모델이 올라가고 데이터만 분산됨



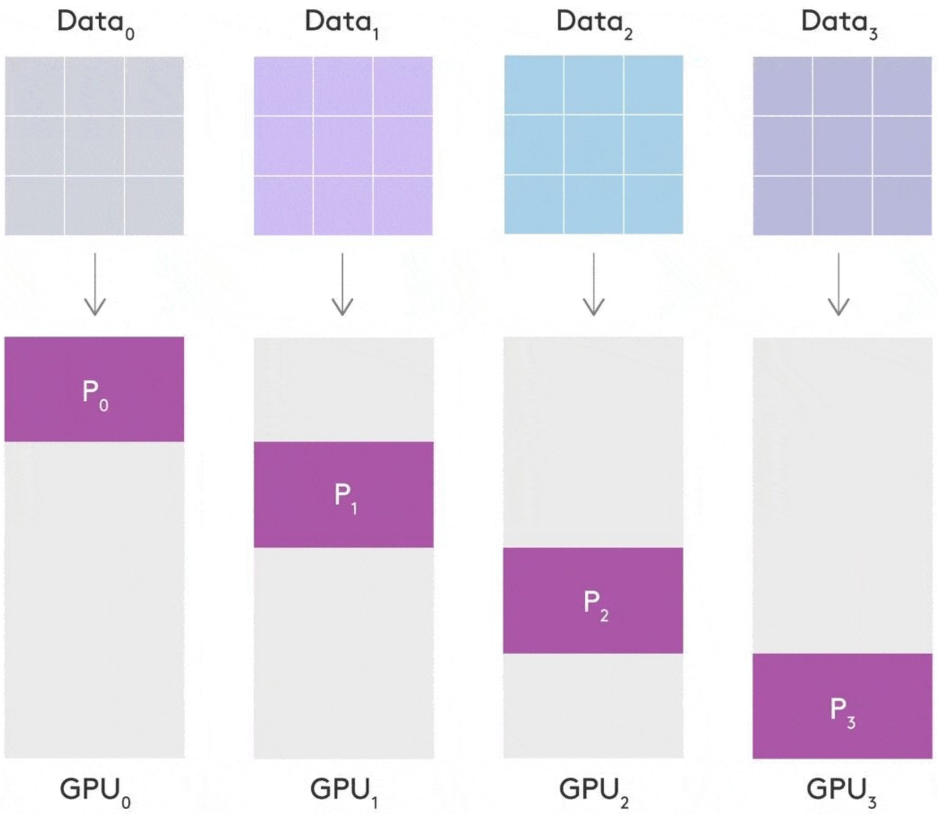
Data parallelism [1]

[1] https://seannaren.medium.com/introducing-pytorch-lightning-sharded-train-sota-models-with-half-the-memory-7bcc8b4484f2

Distributed Training

Sharded Data Parallelism

- 각 GPU 에 모델과 데이터가 모두 분산됨



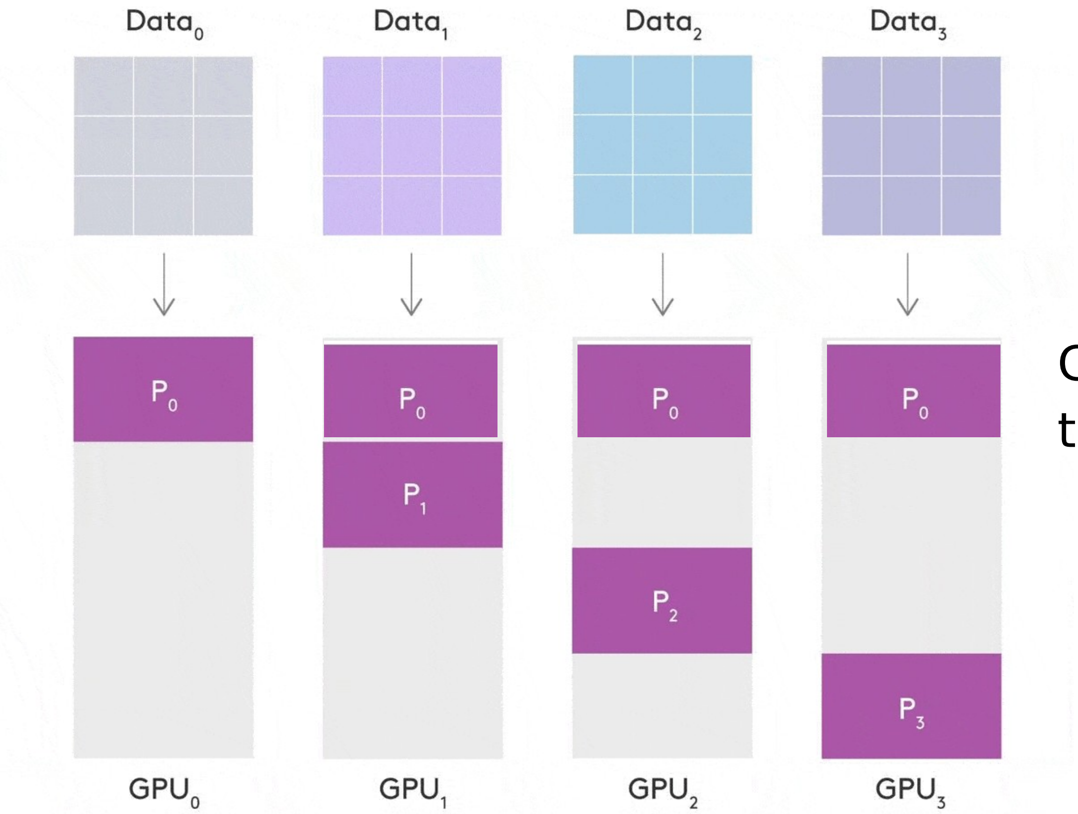
Sharded data parallelism [1]

[1] https://seannaren.medium.com/introducing-pytorch-lightning-sharded-train-sota-models-with-half-the-memory-7bcc8b4484f2

Distributed Training

Sharded Data Parallelism

- 각 GPU 에 모델과 데이터가 모두 분산됨



Copy P_0 from GPU 0 to GPUs 1, 2, 3

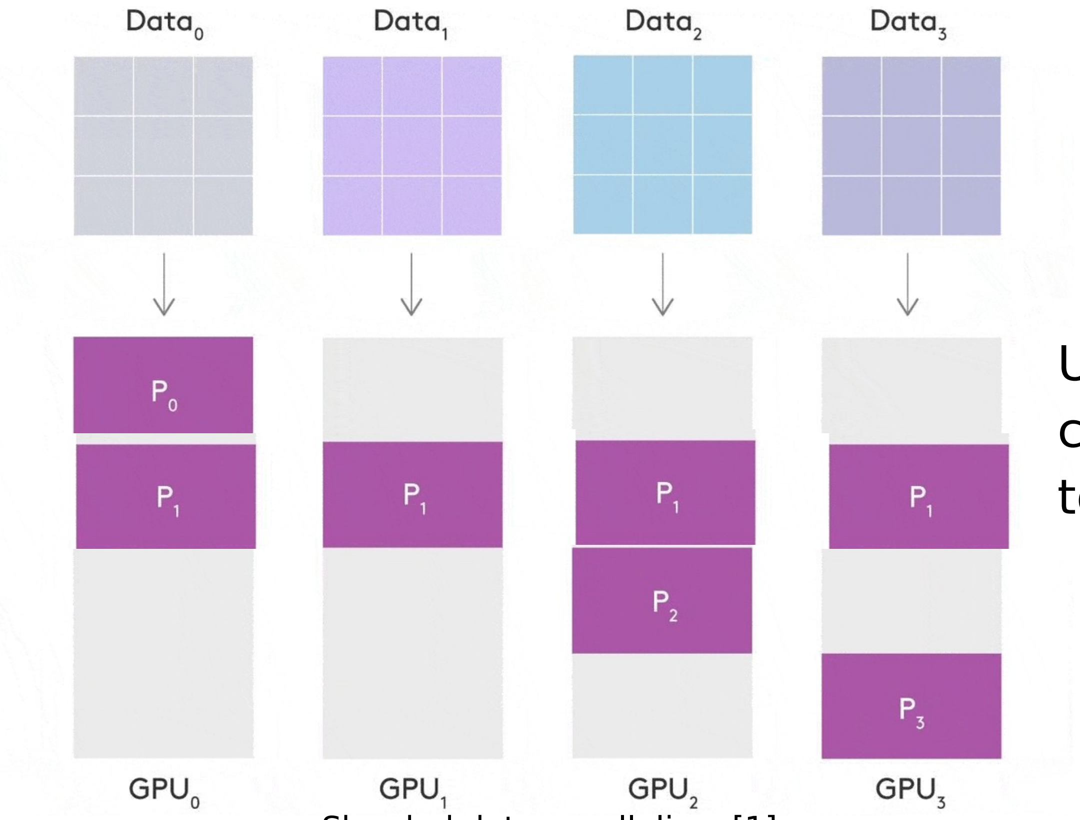
Sharded data parallelism [1]

[1] https://seannaren.medium.com/introducing-pytorch-lightning-sharded-train-sota-models-with-half-the-memory-7bcc8b4484f2

Distributed Training

Sharded Data Parallelism

- 각 GPU 에 모델과 데이터가 모두 분산됨



Unload P_0 and copy P_1 from GPU 1 to GPUs 0, 2, 3

Sharded data parallelism [1]

[1] https://seannaren.medium.com/introducing-pytorch-lightning-sharded-train-sota-models-with-half-the-memory-7bcc8b4484f2

Conclusion

Distributed Training

- Tensor Parallelism: 메모리↓ , 시간 ↓
- Pipeline Parallelism: 메모리↓
- Data Parallelism: 시간↓
- Sharded Data Parallelism: 메모리↓ , 시간↓