SNIA DEVELOPER CONFERENCE
SDC 25
By Developers FOR Developers

Hyatt Regency Santa Clara, CA
September 15-17, 2025

Storage for AI 102

A further perspective of AI Workloads and how they affect storage planning

Curtis Ballard
SNIA Technical Council

www.sniadeveloper.org

Speaker

Curtis Ballard

SNIA Technical Council

HPE

Bio: SNIA Technical Council

SDC 25

What this presentation IS

Builds On
Storage for AI 101*

High Level
Overview of AI use of storage

Thinking Food
for other AI SDC talks

* https://www.snia.org/sniadeveloper/session/18369 (SDC Regional 2025)
https://www.snia.org/sniadeveloper/session/18471 (SDC 2024)

What Kind of Storage?

- File storage?
- Object storage?
- Block storage?

- Direct attached storage?
- External (SAN/NAS) storage?

- **It Depends**
  - This presentation will cover general characteristics of AI workloads, from the perspective of the “storage”.
  - The specific storage implementation for any given workload has lots of choice points that need deeper analysis.

Data is the Food for AI

AI: Runs On

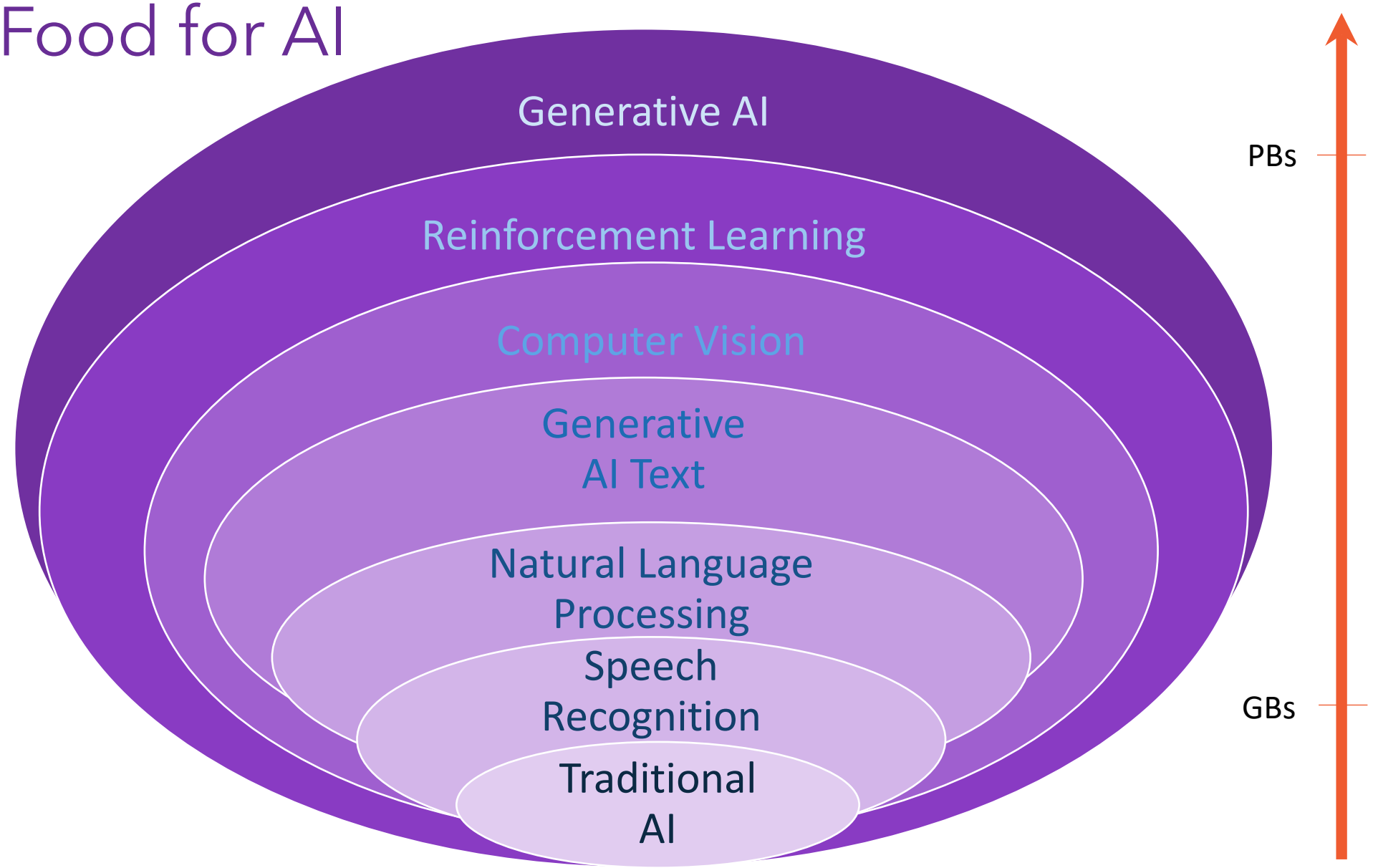
GPUs and CPUs

but

- Eats Data!

and data

- Eats Storage!



Data is the Food for AI

AI: Runs On

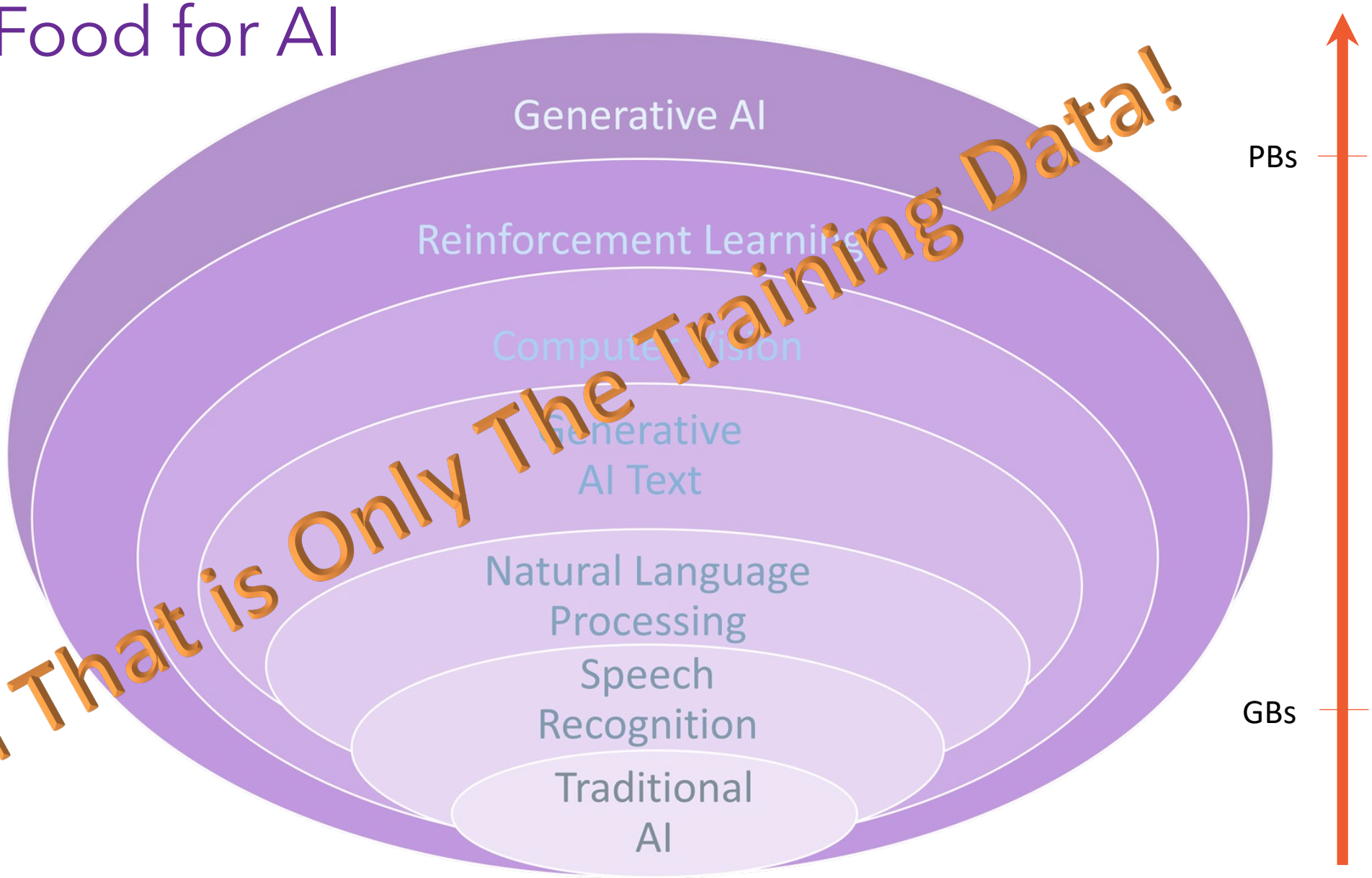
GPUs and CPUs

but

- Eats Data!

and data

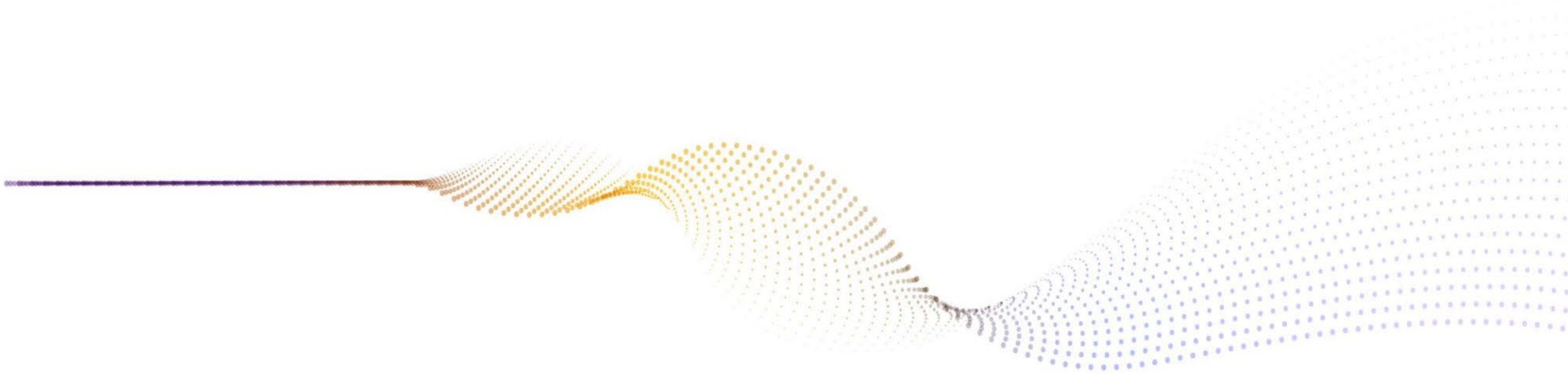
- Eats Storage!



Storage is the Foundation for AI

Unique storage requirements for AI depend on a solid foundation

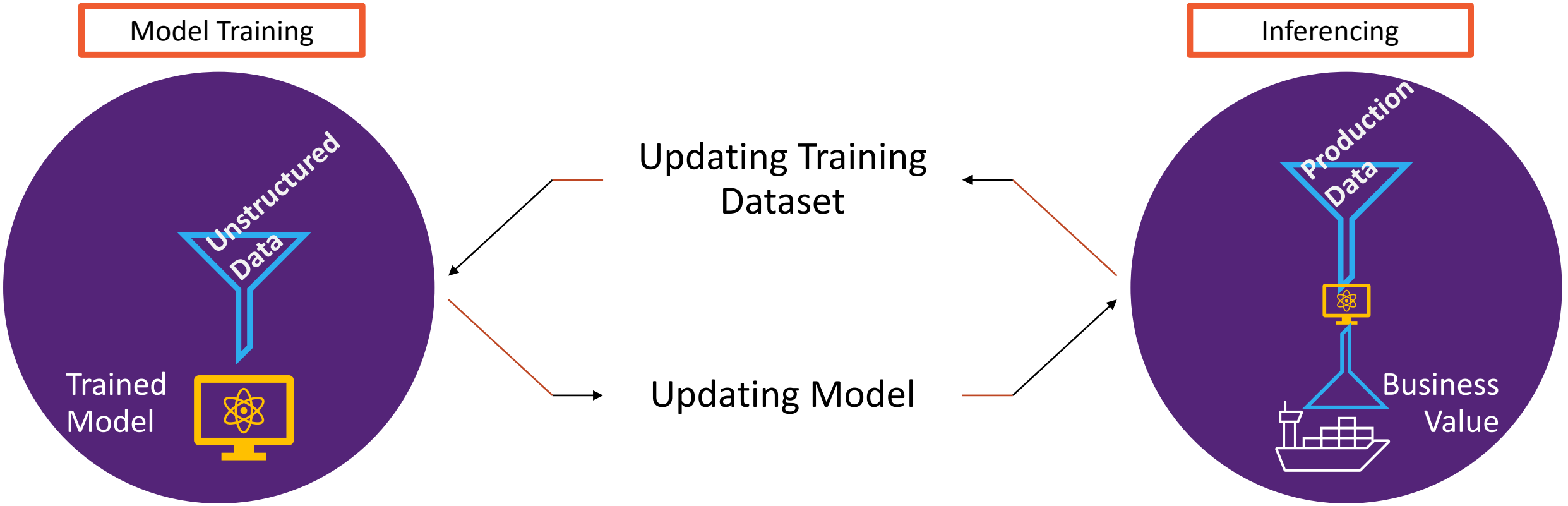
- Huge amounts of data
- Extensive data manipulation
- Multi-phase workload with different data types, sizes, accesses
- Widely variable performance and capacity for different phases
- Highly parallel operations in some phases



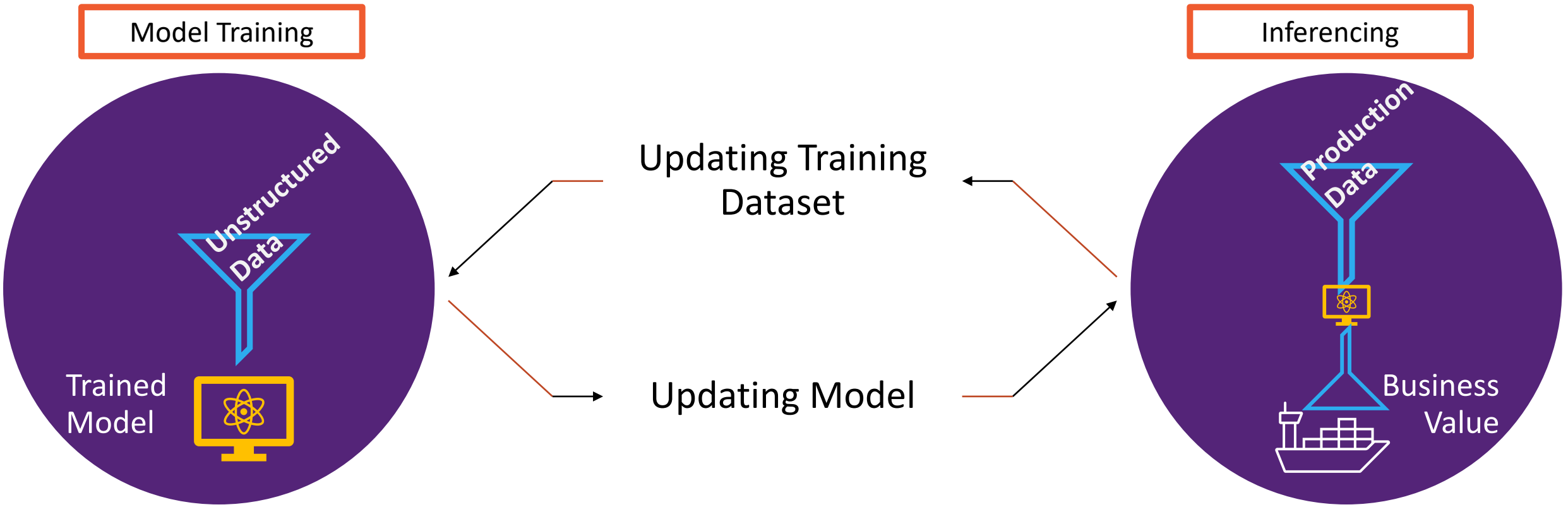
A Quick Intro to AI use of Storage

The Two Sides of AI: Training and Inferencing

The Two Sides of AI: Training and Inferencing

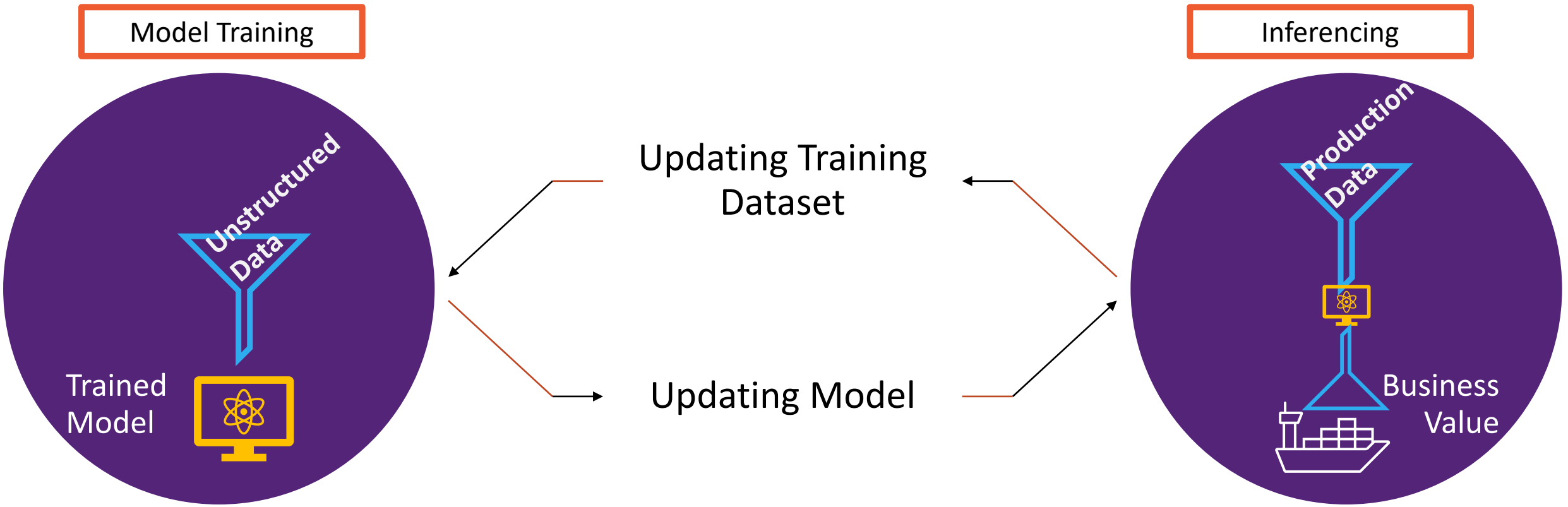


The Two Sides of AI: Training and Inferencing



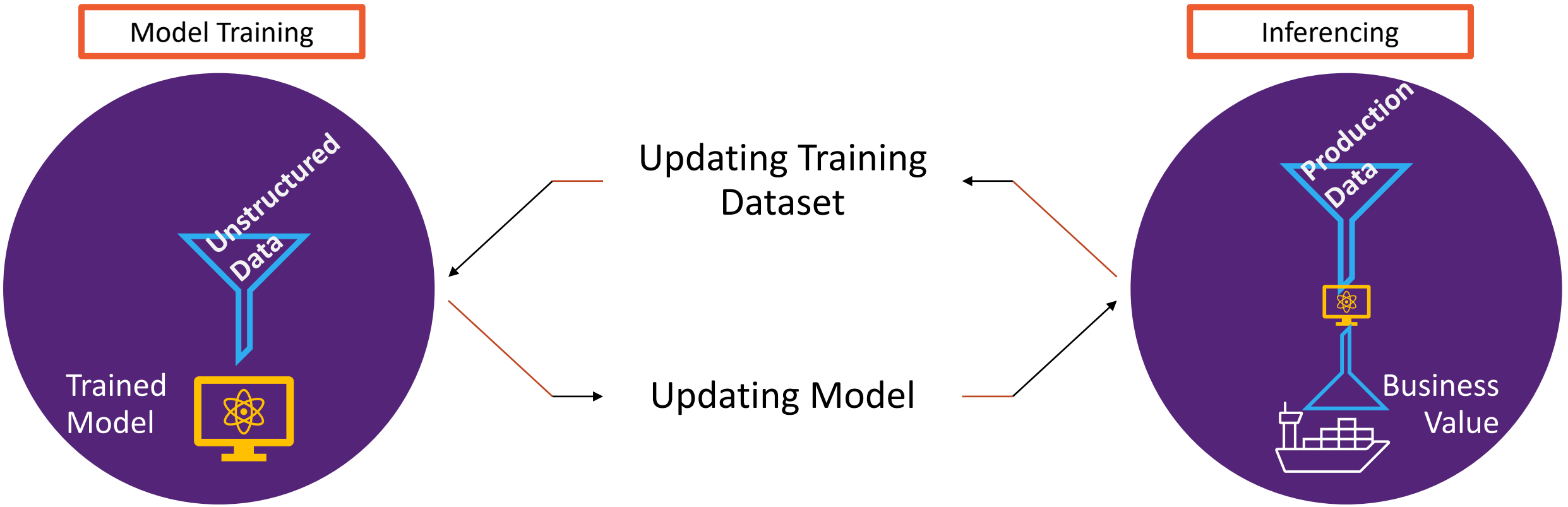
But

The Two Sides of AI: Training and Inferencing

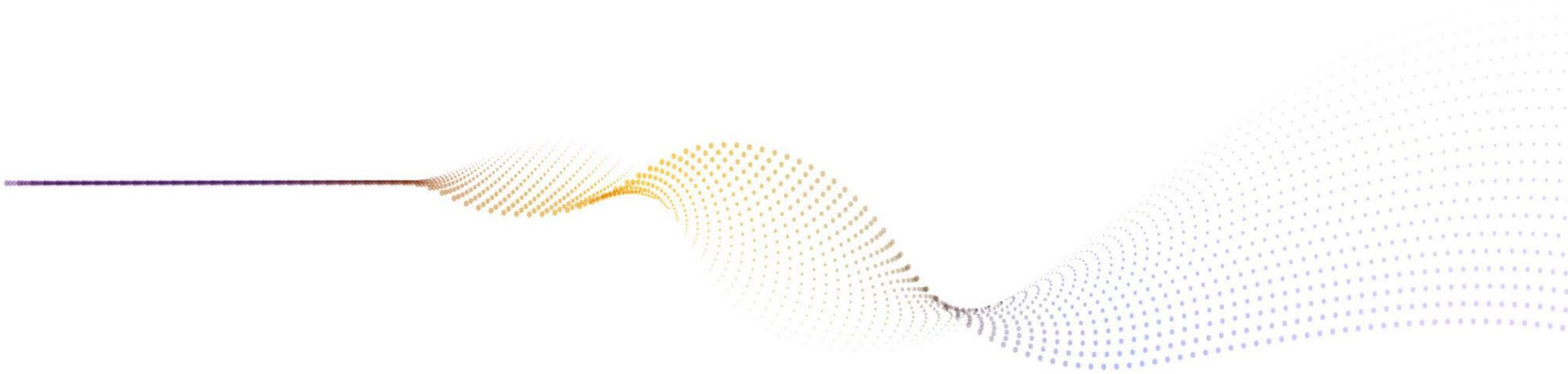


Several phases support these

The Two Sides of AI: Training and Inferencing



And Every Phase Uses Storage Differently

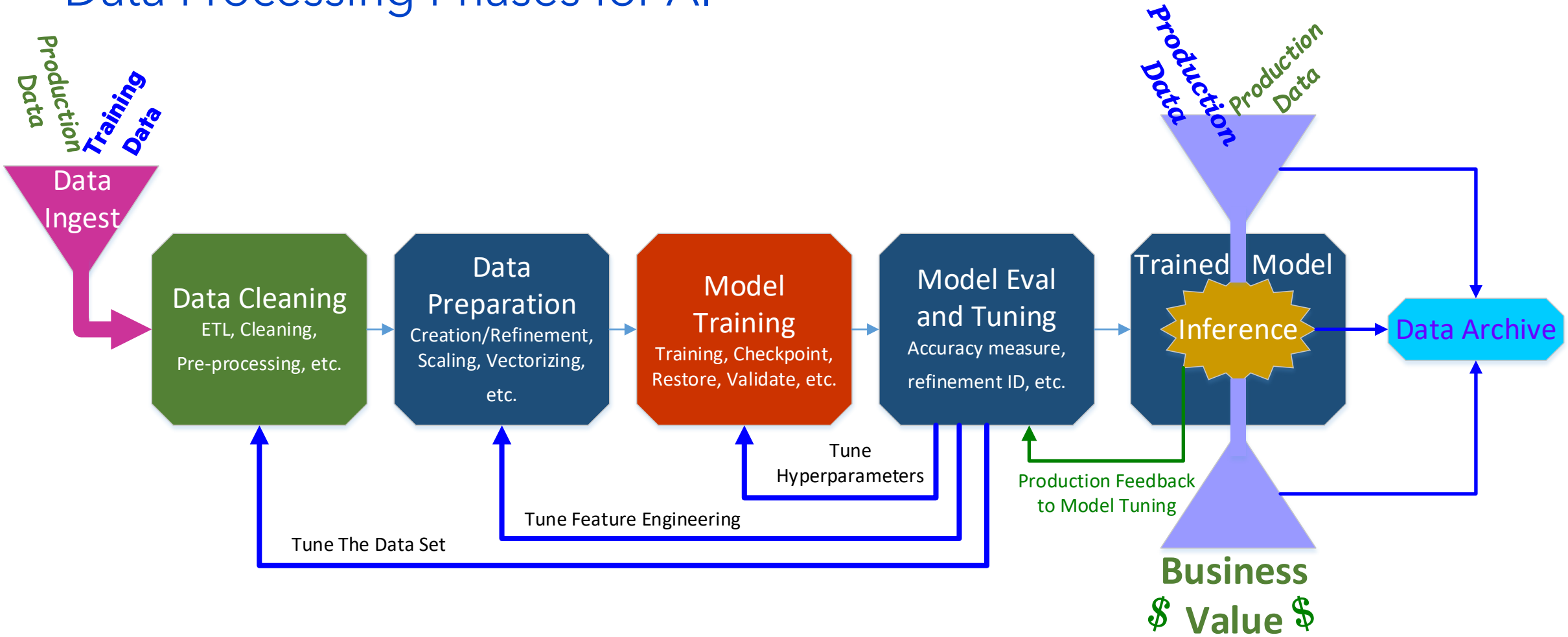


What Does An AI Pipeline Look Like

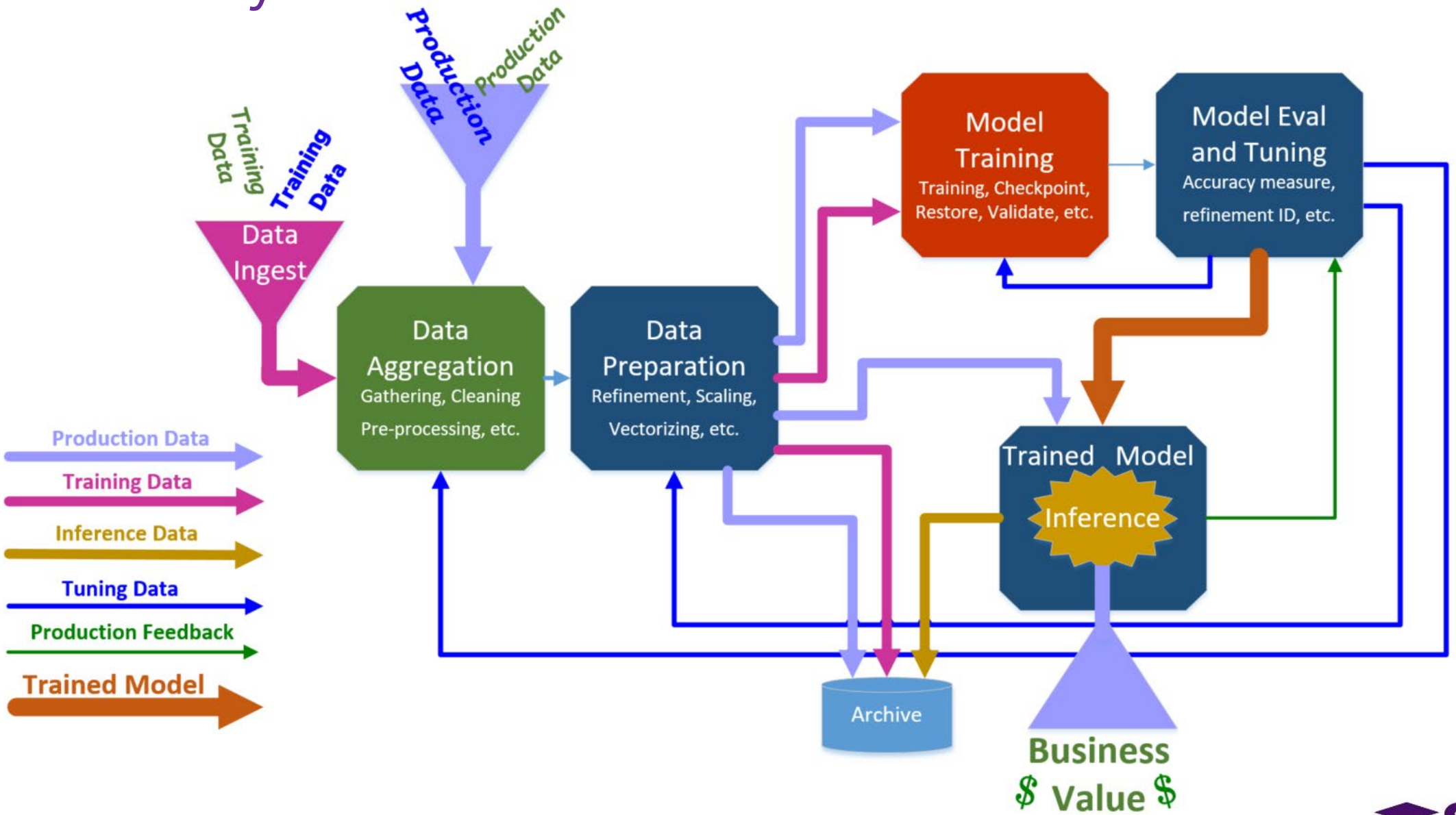
From a storage perspective

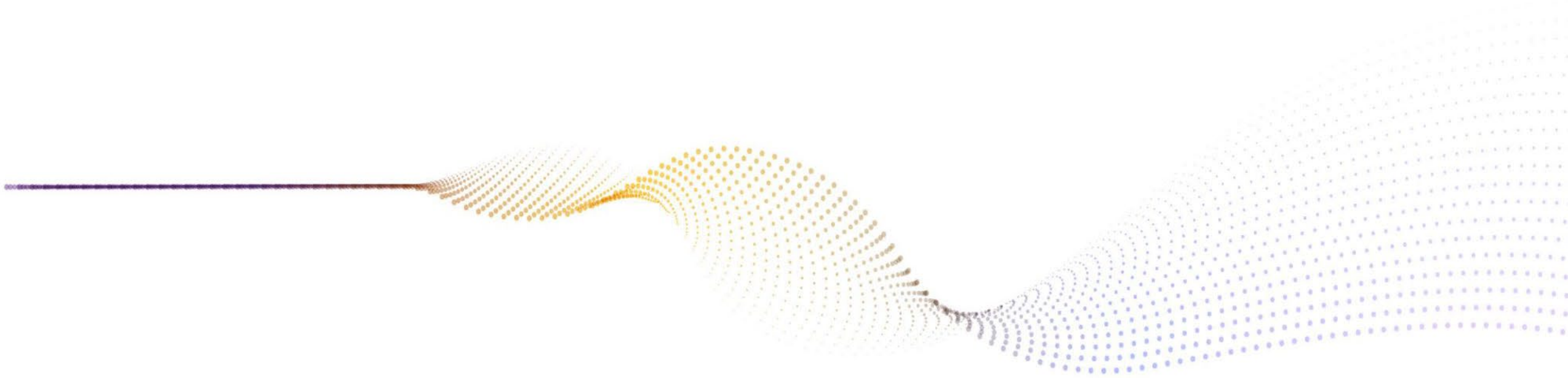
A Simplified View

Data Processing Phases for AI



But it really looks more like



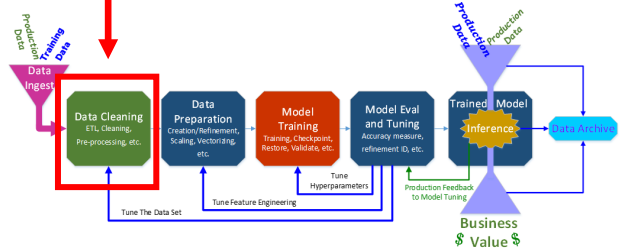


Overview of Data Processing Phases for AI

What happens in these phases?

What might the storage workload look like?

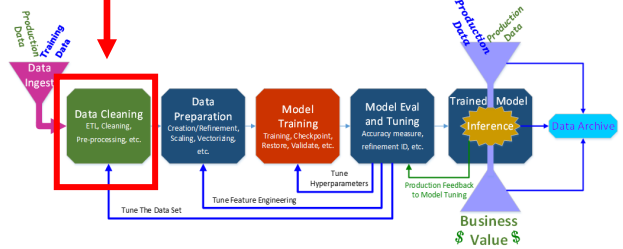
Data Aggregation



Good AI Results Requires Good Data

- Identify the data and data sources
  - Usually multi-modal today: documents, pictures, video, etc.
  - Many potential sources: logs, databases, commercial data sets, etc.
  - Carefully consider where your businesses unique information lies
- Data Ingest/ETL
  - Connectors to data pipelines for the identified data
- Clean and preprocess
  - Duplicate detect/remove
  - Outlier evaluation/removal
  - Incomplete data detect/repair/remove
  - Normalize

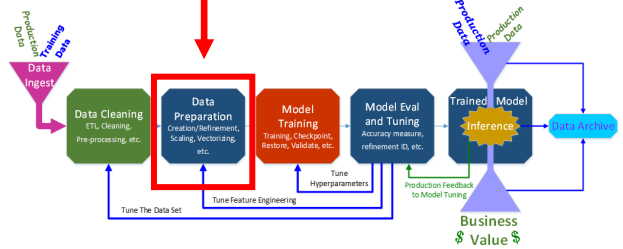
Data Aggregation Workload



- Data Ingest patterns vary – what does yours look like?
  - Continuous, relatively consistent data input can be sequentialized
  - Seemingly random data may be able to be sequentialized
- Capacity – data often collected in a data lake
  - Saving the right data frequently means saving a lot of data
  - Information not saved often can't be recaptured
- Data Access patterns
  - Random reads across data sets for cleaning and preprocessing
  - Random writes of prepared data
- Performance – As fast as is affordable at the right capacity

Sequential Random

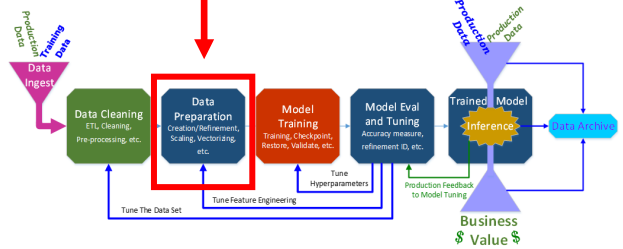
Data Preparation



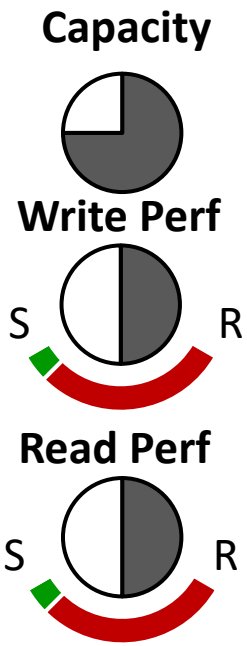
Turning good data into AI Food

- Data Scientists serve as translators
  - Raw data → Food for AI (Numbers)
  - Exploring the data – identifying patterns, outliers, relationships, etc.
  - Splitting data for training and testing
  - Feature extraction – converting key features into consumable nuggets
- Data transformation – converting data types (Vectorizing)
- Often highly parallel
- Domain knowledge is required

Data Preparation Workload

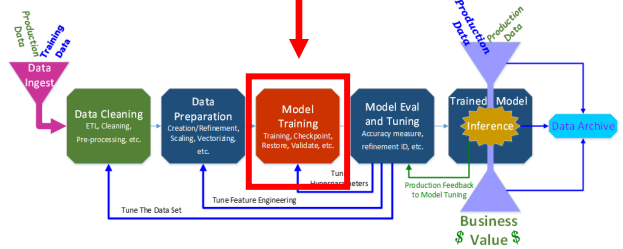


- Data Ingest – Data from the aggregation phase
  - Ideally data preparation and aggregation use the same storage
  - Data access patterns are different
- Capacity
  - Often uses same data lake as data aggregation
  - Data has been cleaned so capacity needs are somewhat less
- Data Access Patterns
  - Highly random reads selecting key data
  - Highly random writes of prepared data



Sequential Random

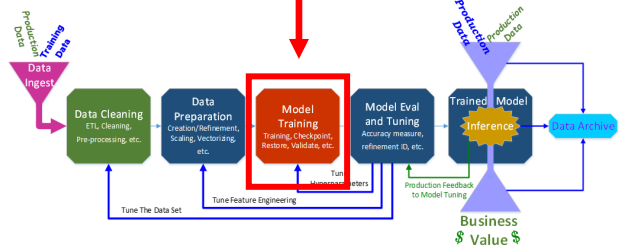
Model Training



Storage affects the efficiency of your model training

- Two key interactions with external storage
- Loading memory for training
  - Highly specialized small models may fit in GPU memory
  - Most models need to periodically load pieces of the model from storage
- Checkpointing
  - A point-in-time backup of the model
  - Essential for large models with long training duration
- Multi-GPU Concurrency
  - Most models split across GPUs with memory loading and checkpointing in parallel

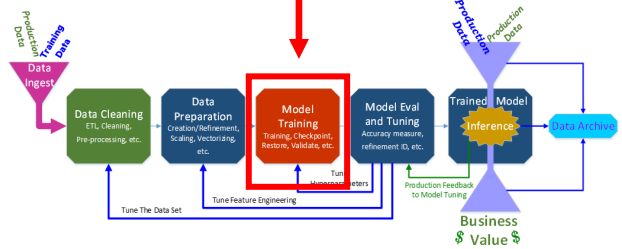
Data Loading for Model Training



- Data needs formatted properly to match the GPU
  - Tools like PyTorch and TensorFlow
  - GPU vendor tools
- Format data for multi-GPU training
  - Several different types of parallelism
- Many worker threads loading multiple GPUs in parallel
- Multiple models for loading data into GPUs
  - Traditional CPU driven
  - RDMA accelerated directly into GPU memory
  - GPU Initiated *
- SNIA Webinar: Critical Role of Storage in Optimizing AI Training Workloads Webinar
  - Also discusses our next topic, Checkpointing

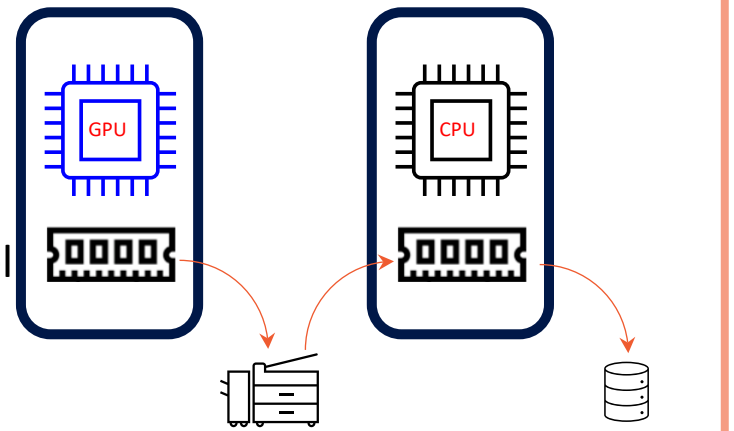
* https://arxiv.org/abs/2203.04910 - BaM System Architecture paper

Checkpointing

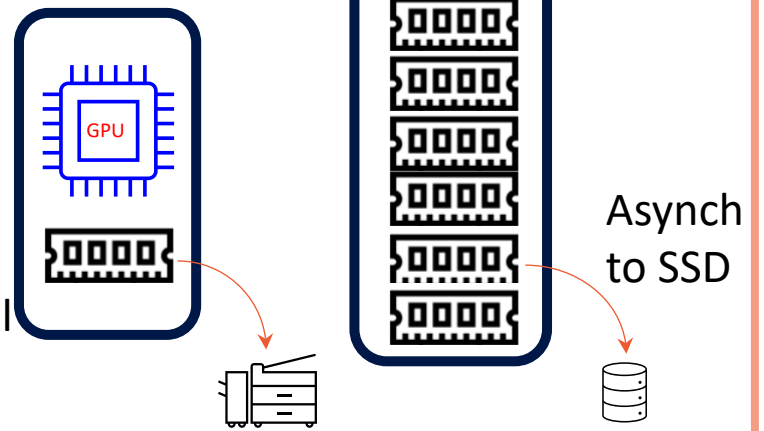


- Constantly evolving for improved efficiency
- Often asynchronous today to minimize GPU stall time*
  - GPU usually paused to snap model state from memory
  - SDC session Tuesday with Pratik Mishra ‡
- Storage perf often drives checkpoint frequency
  - More frequent checkpoints reduce cost of restarts

Synchronous
- State copied to CPU memory
- CPU writes model state to SSD
- Training restarts after copy to SSD

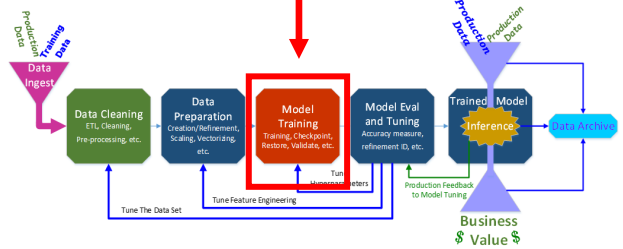


Async
- State copied to CPU memory
- Training restarts
- CPU copies model state to SSD later



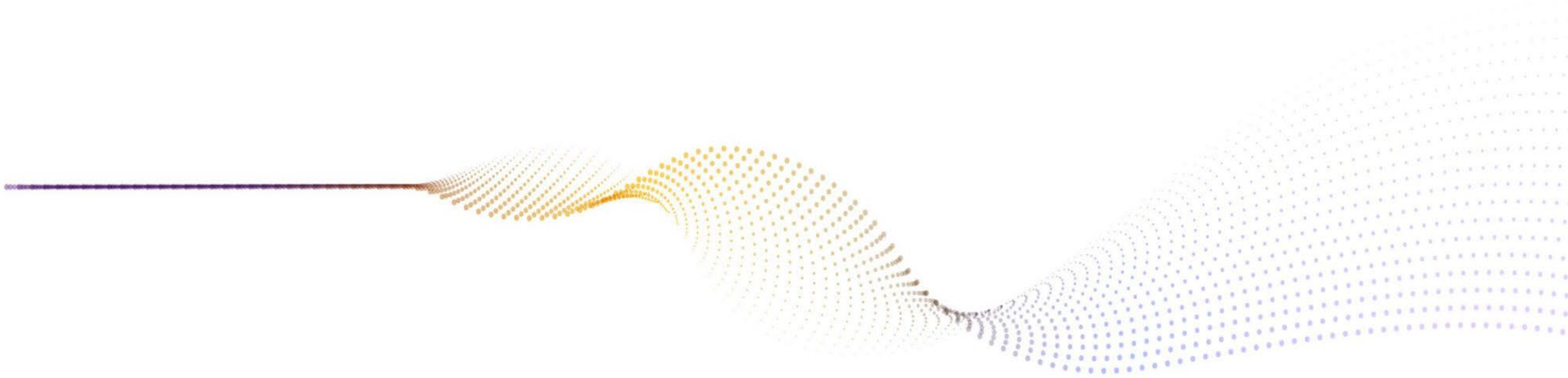
* https://pytorch.org/blog/6x-faster-async-checkpointing/
‡ https://www.snia.org/sniadeveloper/session/19315

Model Training Workload



- Data Loading
  - Often includes preprocessing with data transforms and/or augment
  - Many parallel threads to read/process for next training batch
  - Usually file, high performance object gaining popularity
  - Frequently random samples, each read sequentially
- Checkpointing
  - Files are written sequentially
  - May be multiple sequential writes in parallel
  - Restore - high seq read, many parallel reads to multiple GPUs
  - Storage performance determined by save/restore time goals

Sequential Random

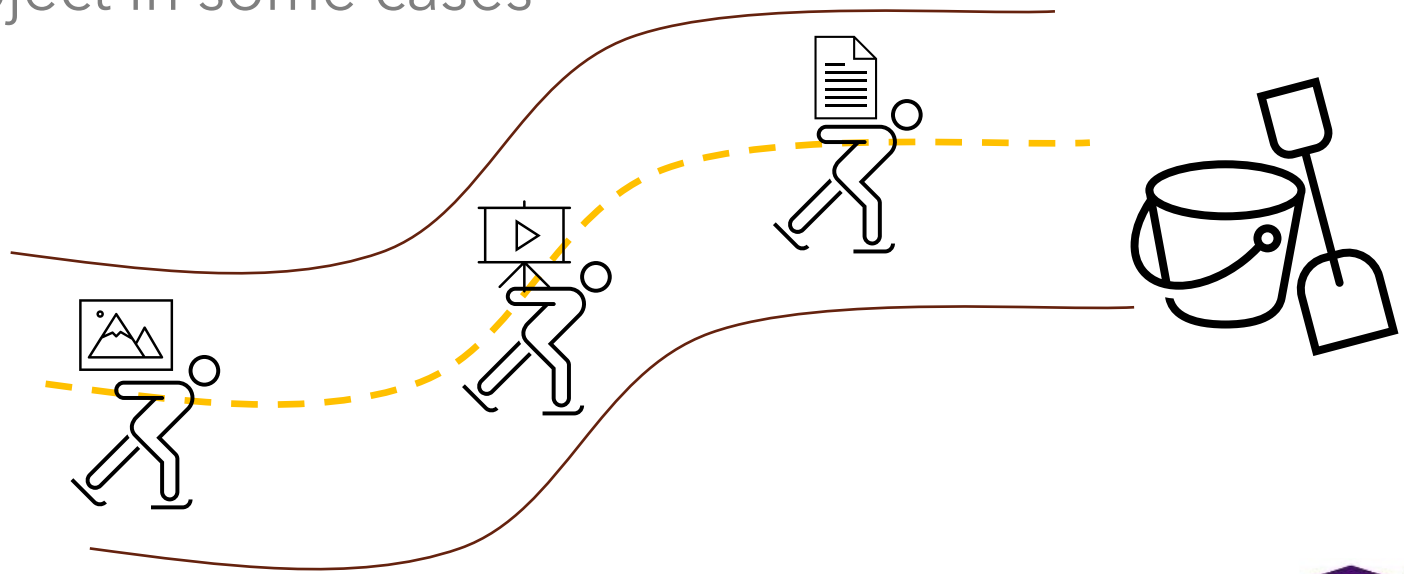


Further Thinking About Model Training

Fast Object for AI

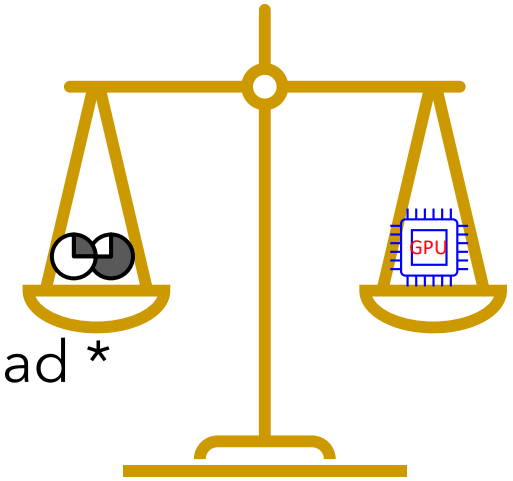
- Object storage is convenient but often has higher latency than file
- File has been more popular than object for AI workloads
- Fast object has been gaining in popularity
  - SSD based object in most cases
  - RDMA accelerated object in some cases

- SDC session: Accelerating Object Storage for AI/ML with S3 RDMA
- SDC BoF Tuesday

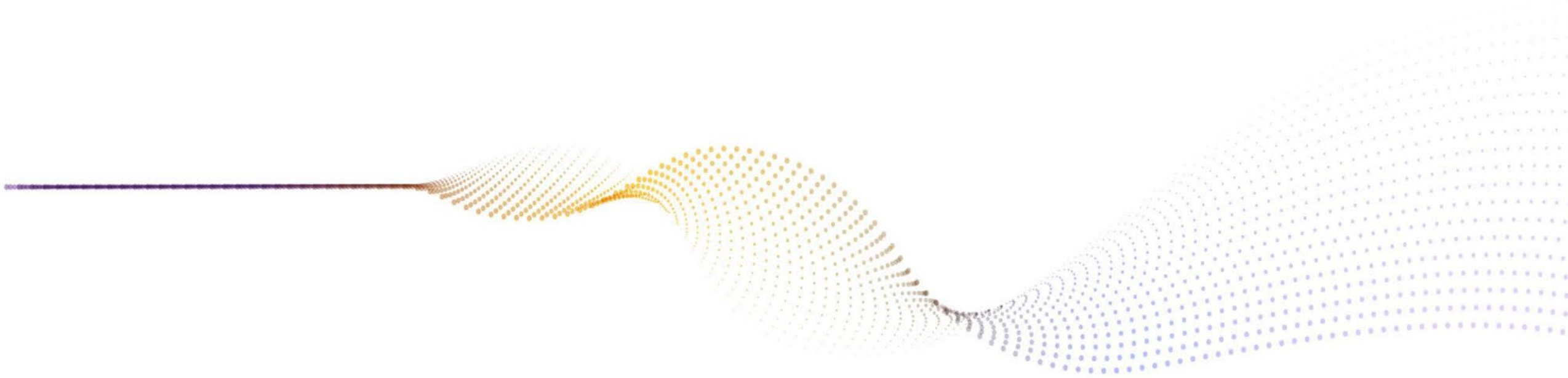


Model Training – General Storage Planning

- GPUs drive the cost – maximizing GPU utilization optimizes investment
- Design for a balanced architecture
  - Balance storage performance with GPU requirements
- Consider data sources
  - May require both file and object access
- If known training workloads – match storage performance to workload *
  - AI GPU benchmarks can show peak performance for various models
  - MLCommons MLPerf Training benchmarks is a good source
  - Determine size of training examples
  - Multiply throughput and size to estimate required read bandwidth
- For general purpose training may need to support GPU max read speed
  - Can be up to 1GB/s per GPU for high end GPUs today, increasing regularly



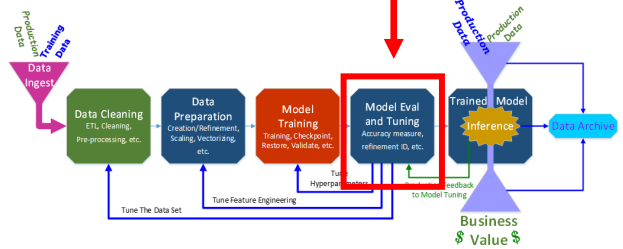
* https://www.snia.org/educational-library/storage-requirements-ai-2024



And Back To Workload Overview

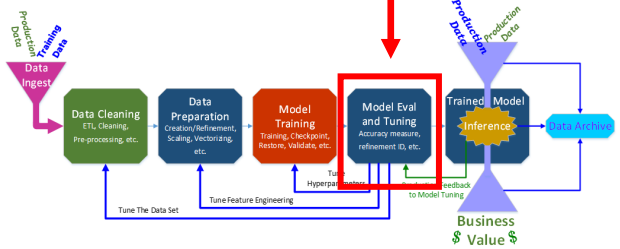
Programming already in progress interrupted for the short AI training break

Model Evaluation and Tuning



- Often considered to be part of the training
  - Can be part of successive training refinement or tune/retrain with new data
- Multiple possible goals
  - Improve accuracy, domain specific focus, bias or style
- May require new data aligned with tuning goals
- Multiple techniques
  - Adjust then retrain entire model
  - Fine tune specific layers
- Measuring how well the results of the model match expectations
  - Accuracy – how often is it correct?
  - Precision/Recall – roughly a measure of how often wrong vs right
  - Measures such as F1 Score and AUC-ROC (area under the curve/receiver operating characteristics

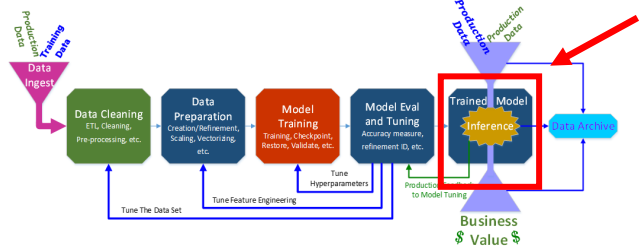
Evaluation and Tuning Workload



- Random reads for model evaluation
  - Compare test/validate datasets
  - Evaluate metrics
- Random read/write updating the hyperparameters
  - Usually stored in a structured format like JSON or YAML
- Usually lower capacity requirements
  - Evaluating results, adjusting parameters – not the full dataset

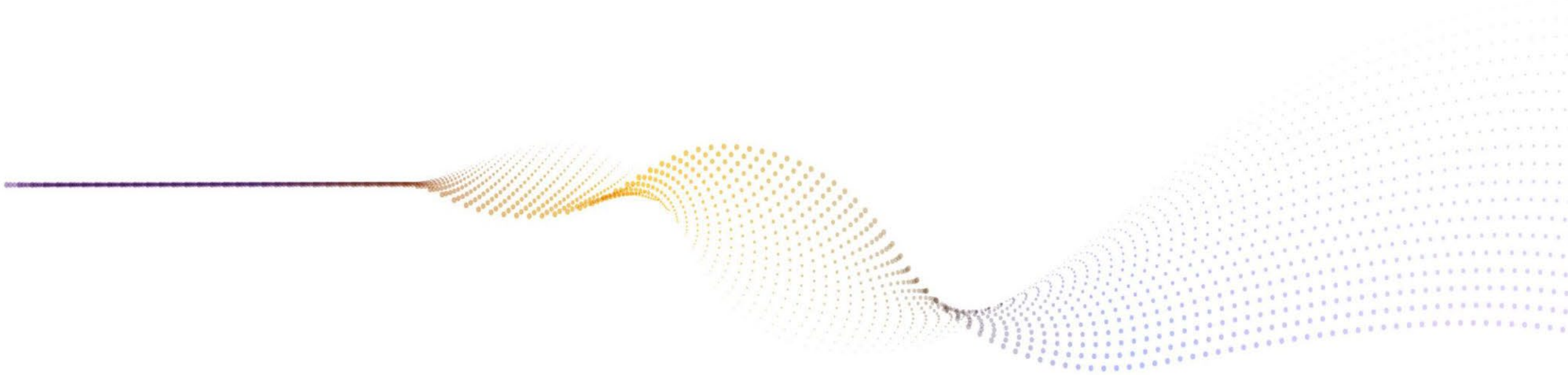
Sequential Random

Inference



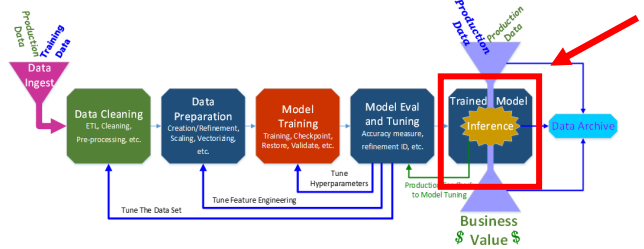
Where most business value is created!

- Inference is the process where an AI model is used to "Infer" information based on your inputs and the model's training
- Trained models develop pattern recognition techniques which are then applied during inference for tasks like:
  - Make conclusions from the data
  - Use learned patterns/techniques to manipulate the data
  - Generate recommendations
- Storage access patterns have some themes but also vary



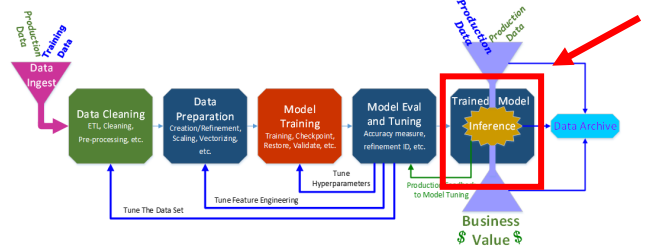
Quick Intro to Inference

Simple Inference



- Input prompt is 'tokenized' – (converted to numbers representing the input)
- Tokens are fed into the neural network
- Computations performed on the tokens using the training and information in the selected AI model
  - Knowledge applied is embedded in training weights
- Output tokens are generated
- Results stored with random writes
- "Context" tracks information like: prompts, instructions, prior results
- Example storage size: GPT-3 standard context window, 2048 tokens

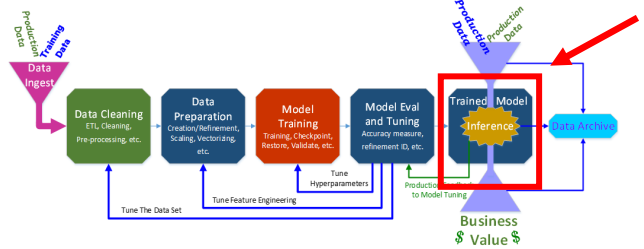
Simple Inference



Mostly memory

- Input prompt is 'tokenized' – (converted to numbers representing the input)
- Tokens are fed into the neural network
- Computations performed on the tokens using the training and information in the selected AI model
  - Knowledge applied is embedded in training weights
- Output tokens are generated
- Results stored with random writes
- "Context" tracks information like: prompts, instructions, prior results
- Example storage size: GPT-3 standard context window, 2048 tokens

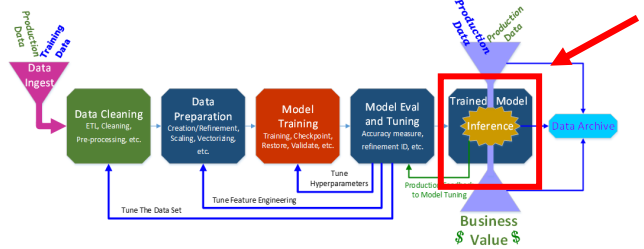
Simple Inference



Mostly memory

- Input prompt is 'tokenized' – (converted to numbers representing the input)
- Tokens are fed into the neural network
- Computations performed on the tokens using the training and information in the selected AI model
  - Knowledge applied is embedded in training weights
- Output tokens are generated
- Results stored with random writes
- "Context" tracks information like: prompts, instructions, prior results
- Example storage size: GPT-3 standard context window, 2048 tokens
- ***Most inference has moved significantly beyond this***

RAG



Retrieval Augmented Generation

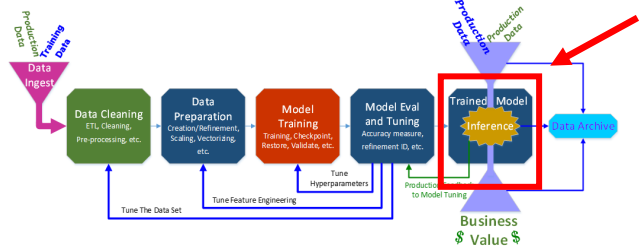
- Retrieves information from an associated db to extend knowledge
- Requires data prep similar to training data prep
  - Data for augmentation is 'vectorized' – converted to numeric representation
  - Data is tokenized then embedded into vectors of a few KB
- During RAG the prompt is used to query a set of similar vectors
  - Original prompt and "similar" vectors are sent to the AI model
  - Extends the model to include new information without retraining
- Stored in the context window
- Newer models growing context windows – 128,000 std GPT-5 tiers
- Size of results written to storage is also growing

And Many More Types of Inference

- Thought models – Chain/Tree/Graph
- Multi-agent models
- Reinforcement learning
- Deep memory, or long term memory models

- Frequently combined with RAG
  - RAG + Knowledge Graphs
  - RAG + Reinforcement Learning
  - RAG + The Kitchen Sink (Metaphorical, anything you can think of)

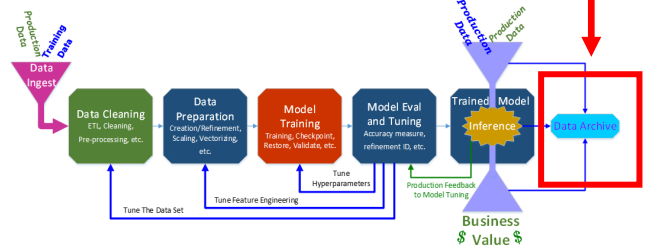
Inference Workload

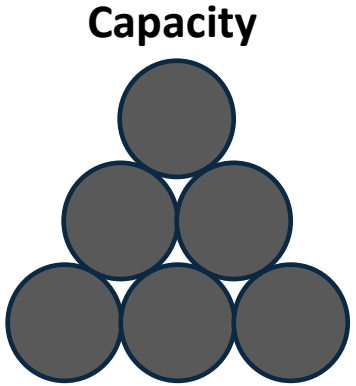


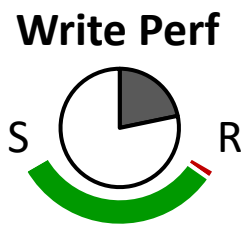
- Workload has variations depending on type of inference
  - Tends to be random with more reads than writes
- Read workload usually random reads
  - Reading input prompts
  - Reading augmentation data for RAG
  - Reading saved context
- Write workload
  - Saving output
  - Saving context
- KV cache – popular model for storing context

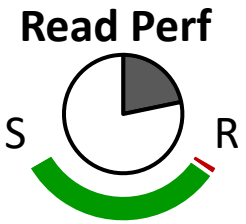
Sequential Random

And Don’t Forget! - Archive



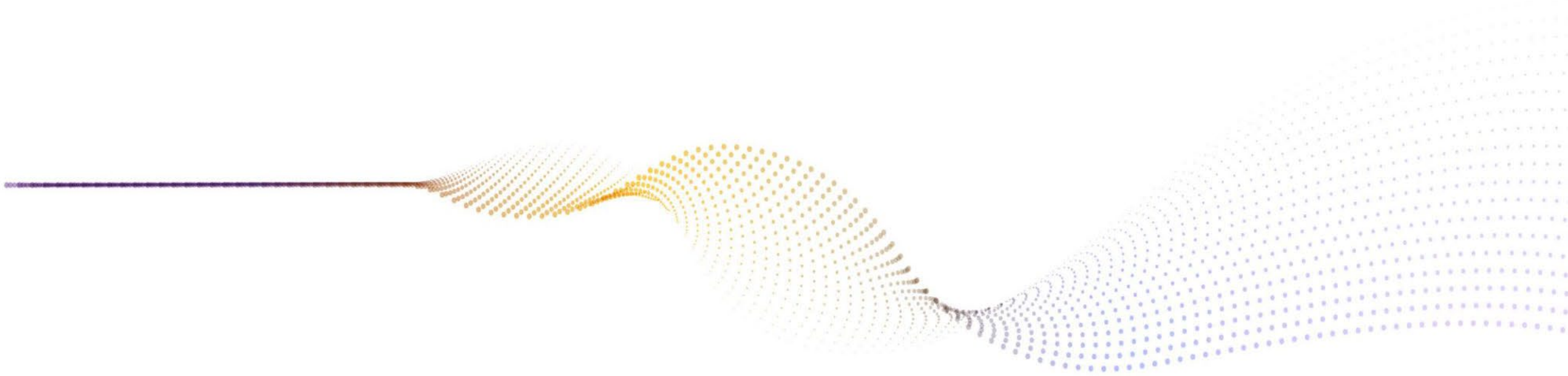






- Often overlooked, not core AI, but important AI storage
- Mandated by regulations for some AI applications
- Similar, but not traditional “archive”
  - Archived data may be brought back for training or new insights
- Performance needs vary but “just fast enough”
- No accepted terminology, maybe “Cold Storage”
- Continually growing data set
- Requires low cost and low carbon footprint storage
  - Opportunity for zero power storage such as DNA and Optical

Sequential Random



Thank you for attending!

Please remember to rate this session. You get access the presentations at
http://sniadeveloper.org/conference

