
AWS
re:Invent
DECEMBER 2 – 6, 2024 | LAS VEGAS, NV

AIM357

Customizing models for enhanced results: Fine-tuning in Amazon Bedrock

Eissa Jamil
Partner Engineer, GenAI
Meta

Yanyan Zhang
Sr. Data Scientist
AWS

Shreyas Subramanian
Principal Data Scientist
AWS

Agenda

01 LLM customization lifecycle

02 Use cases: When should you fine-tune?

03 Amazon Bedrock model customization features

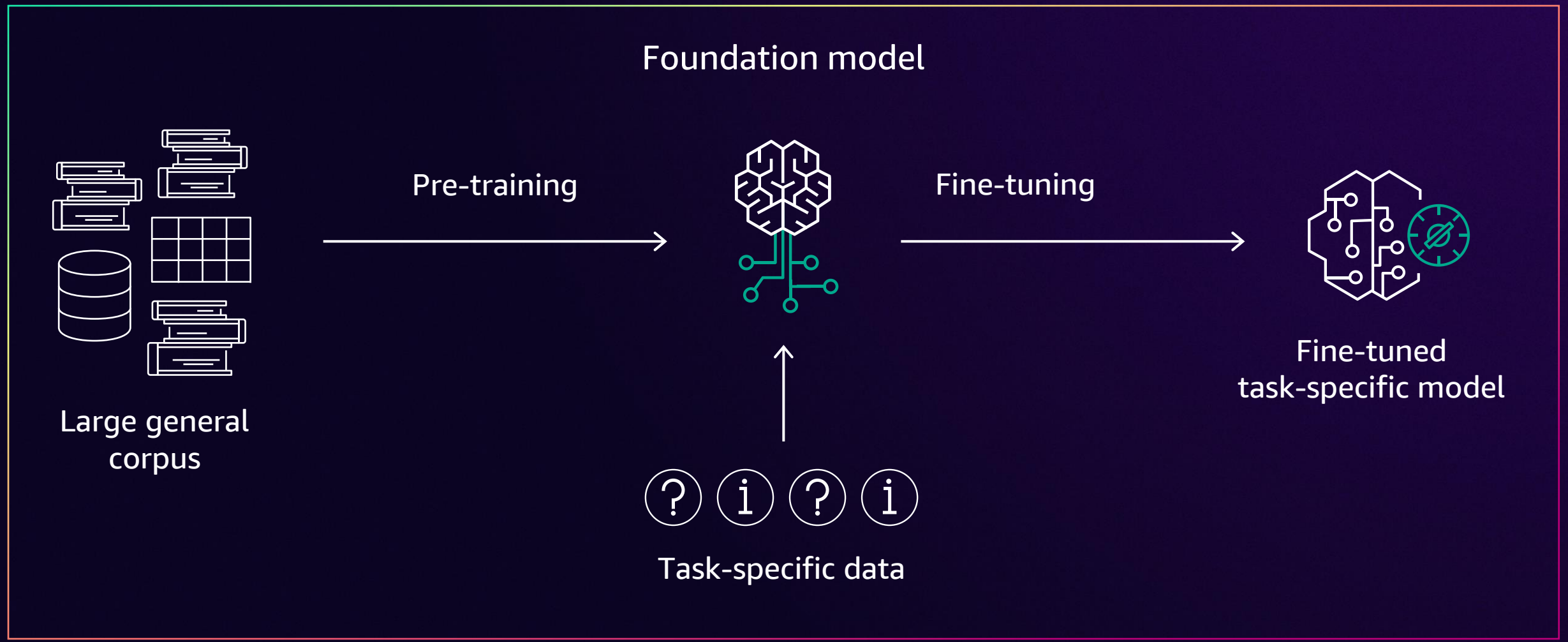
04 Customizing Anthropic's Claude models

05 Customizing Meta's Llama models

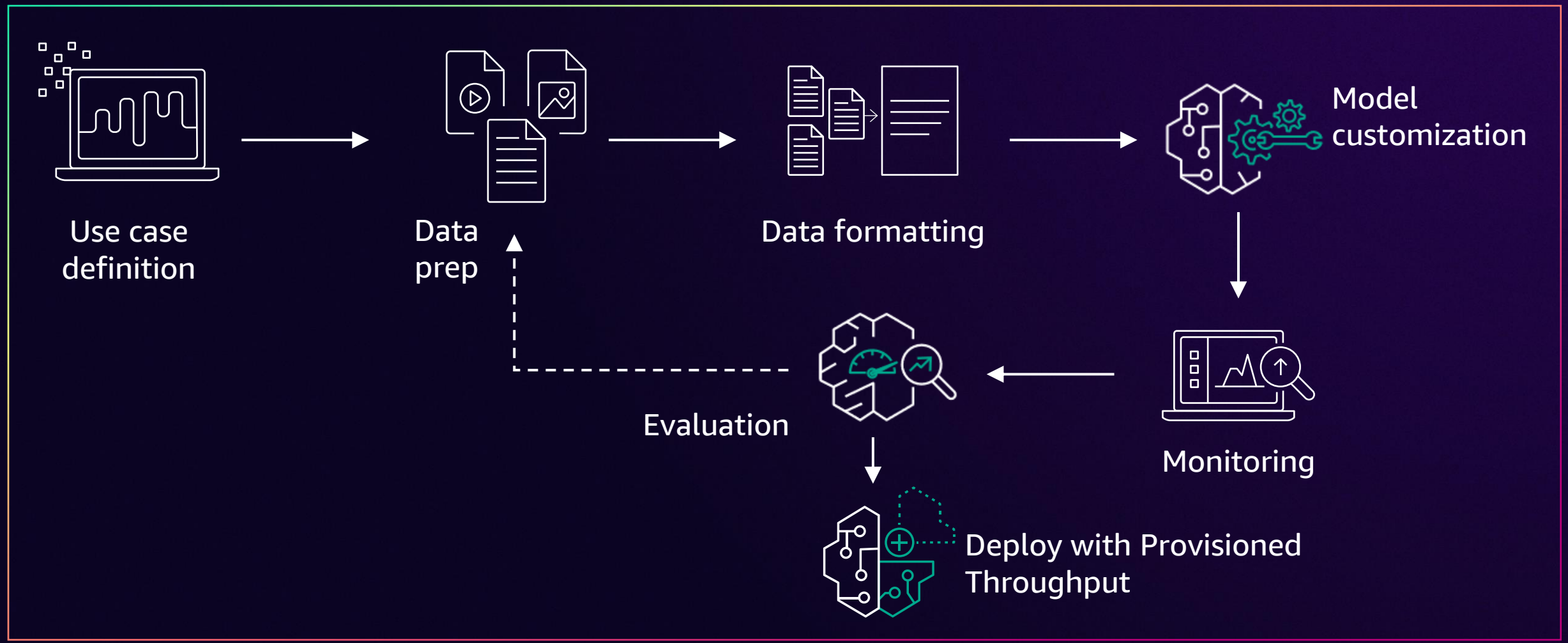
06 Demo

Amazon Bedrock model customization features

What is fine-tuning?



LLM fine-tuning lifecycle



When should you fine-tune?

Problem	Examples	Likelihood of success with fine-tuning	Likelihood of success with prompting (+ RAG)
Make the model follow a specific format or tone	Use this specific JSON schema, talk like my customer service reps	Very high	High
Teach the model a new skill	Learn how to call APIs, fill out proprietary documents, classify customer support tickets	High	Medium
Teach the model a new skill, and hope it learns similar skills	Teach the model to summarize contract documents, hope it learns how to write better contract documents	Low	Medium
Teach the model new knowledge, expect it to use that knowledge for general tasks	Learn my company's acronyms, know more music facts	Very low	Medium

Key considerations for fine-tuning

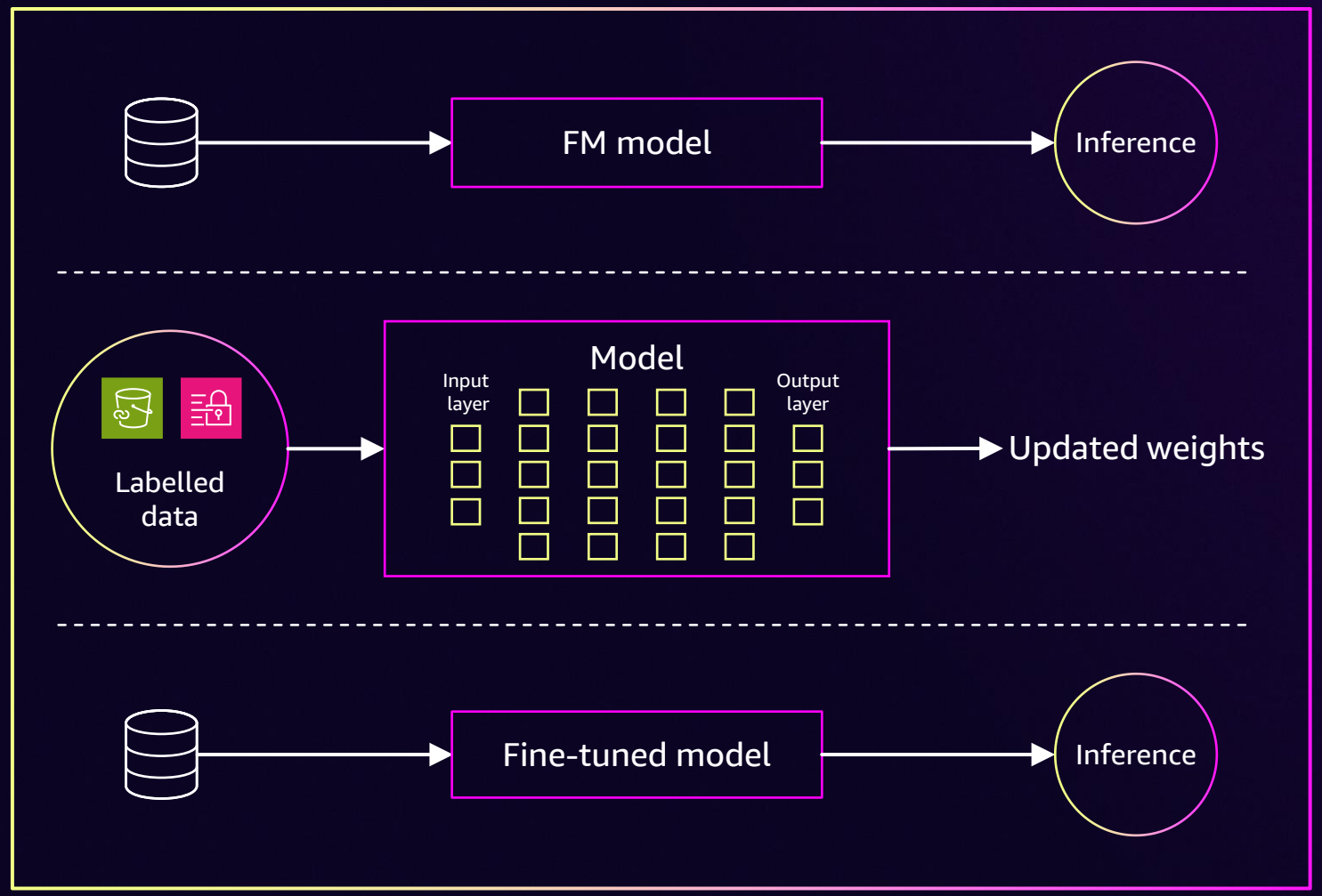
No straightforward answer

It requires experimentation/testing

- New concepts:
  - Is the base model already familiar with the task's concepts (e.g., video games, intent classification)?
  - New concepts are challenging to learn through small-scale fine-tuning
- Promising few-shot results:
  - Improvements with few-shot prompting suggest fine-tuning could offer better outcomes
  - Fine-tuning incorporates more examples directly into model weights
- Token budget:
  - Prompt engineering's lengthy inputs consume tokens quickly
  - Fine-tuning a niche model can be more cost-effective in the long run

Model customization features in Amazon Bedrock

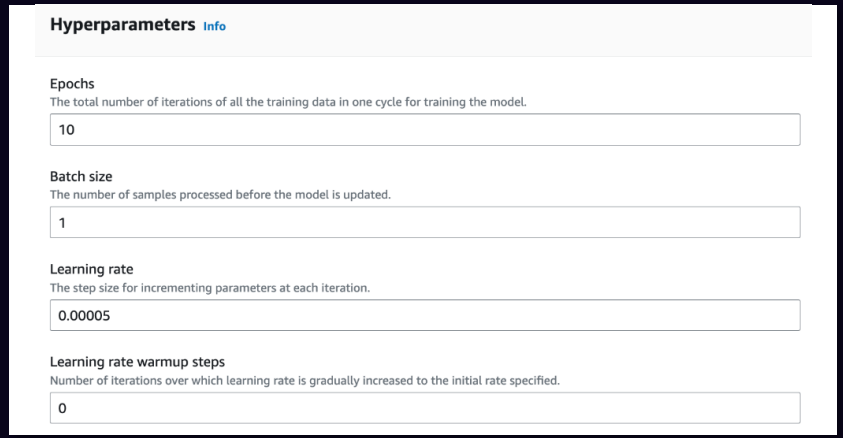
Fine-tuning



Customize models

Customize models with simple configuration

Epochs, batch size, learning rate, warmup steps

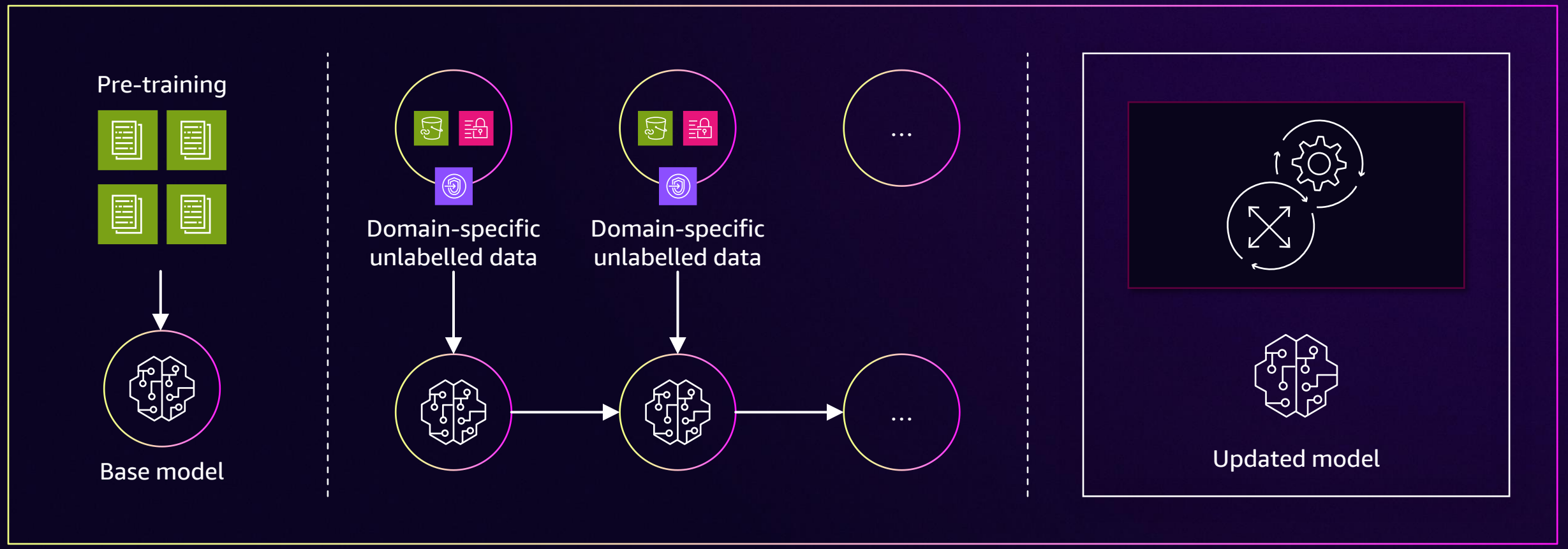


- **Easy** — Simplified hyper params update
- **Secure** — Data encrypted with customer managed keys
- **Private** — Access to Amazon S3 bucket via AWS PrivateLink

Continued pre-training

Customize models

Adapt model responses to the vocabulary and terminology specific to your domain



Amazon Bedrock Custom Model Import

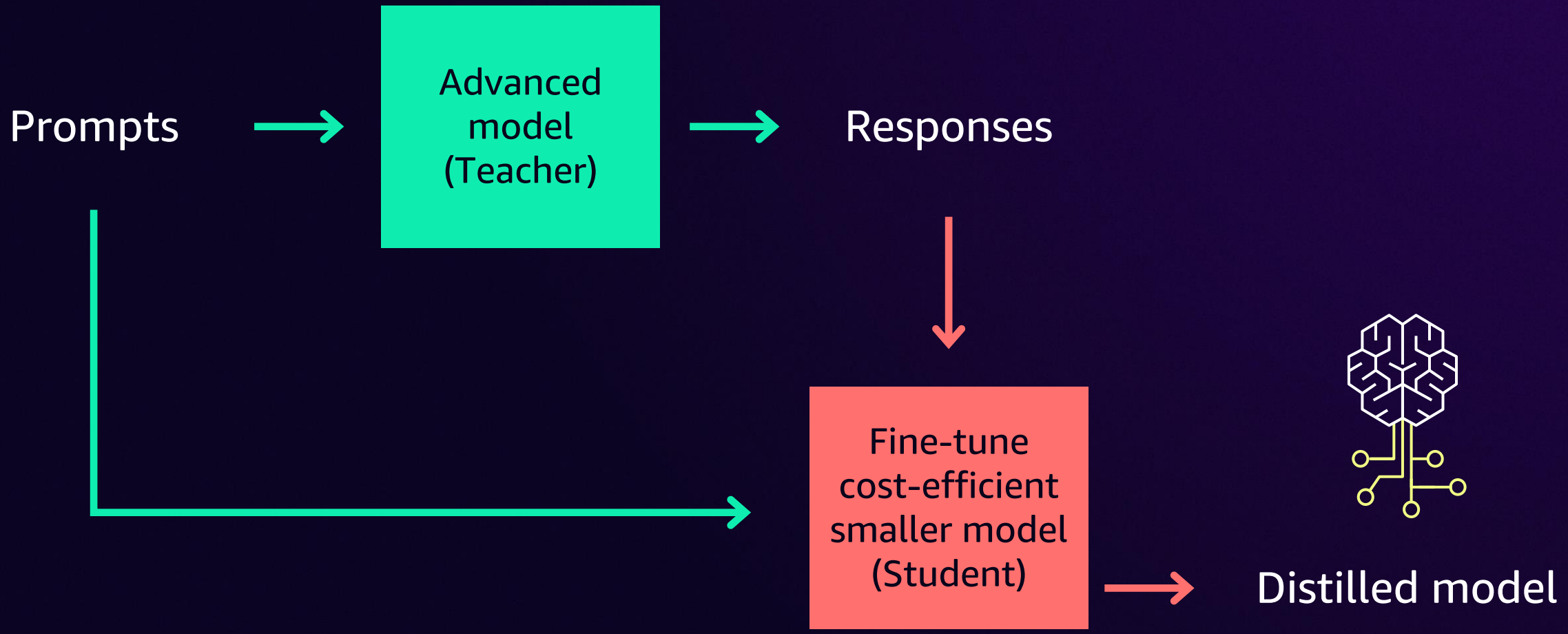
Import your customized models in Amazon Bedrock

1. Model includes model weights and configurations (model and tokenizer configurations)
2. Supports 3 model architectures – Llama, Mistral, and Flan-T5
3. Supports Hugging Face Safetensors format
4. On-demand inference supported; Provisioned Throughput will be a fast follow
5. At launch, feature is console-only for control plane actions. However, once the model is imported, customers can use both the console and SDK to perform invocations
6. With Custom Model Import you can leverage native Amazon Bedrock tooling, such as Knowledge Bases, Guardrails, Agents, and more with your imported custom models just as you do with other Amazon Bedrock on-demand foundation models

Llama/Mistral/Flan T5 FT or Custom Model-> Amazon SageMaker->

Custom Model Import into Amazon Bedrock-> Inference

Amazon Bedrock Model Distillation (Preview)



Customizing Anthropic's Claude 3 Haiku

Claude 3 Haiku fine-tuning data requirements

To fine-tune the Haiku model in Amazon Bedrock, the training data must be in JSONL format, where each line represents a single training record. Training data format aligns with the **MessageAPI**.

```
{
“system”: “Below is an instruction that describes a task, paired with an input that provides further context. Write a response that appropriately completes the request.”,
“messages”: [
{“role”: “user”, “content”: “instruction:\n\nSummarize the news article provided below.\n\ninput:\nSupermarket customers in France can add airline tickets to their shopping lists thanks to a unique promotion by a budget airline. ... Based at the airport, new airline launched in 2007 and is a low-cost subsidiary of the airline.”},
{“role”: “assistant”, “content”: "New airline has included voucher codes with the branded products ... to pay a booking fee and checked baggage fees .”}
] }
```

Haiku fine-tuning parameters

Name	Description	Type	Default	Value range
epochCount	The maximum number of iterations through the entire training dataset. EpochCount is equivalent to epoch.	Integer	2	1–10
batchSize	The number of samples processed before updating model parameters.	Integer	32	4–256
learningRateMultiplier	The multiplier that influences the learning rate at which model parameters are updated after each batch.	Float	1	0.1–2
earlyStoppingThreshold (optional)	The minimum improvement in validation loss required to prevent premature stopping of the training process.	Float	0.001	0–0.1
earlyStoppingPatience (optional)	The tolerance for stagnation in the validation loss metric before stopping the training process.	Integer	2	1–10

Hyperparameter optimization

Learning rate multiplier:

We suggest customers to start with the default value (1.0) and potentially adjust this value based on their evaluation result.

Batch size:

Optimal value can vary depending on your dataset size.

Dataset size	Recommended batch size
> 1,000	32–64
500–1000	16–32
< 500	4–16

In scenarios with large amounts of data (1,000 to 10,000 examples), the learning rate tends to have a more significant impact on performance. Conversely, for smaller datasets (less than 100 examples), the batch size becomes the dominant factor.

Performance evaluation

					Fine-tuned performance	Base model performance			Improvement: FT Claude 3 Haiku versus base models		
Target use case	Task type	Fine-tuning data size	Test data size	Eval metrics	Claude 3 Haiku	Claude 3 Haiku (Base)	Claude 3 Sonnet	Claude 3.5 Sonnet	versus Haiku Base	versus Claude 3 Sonnet Base	versus Claude 3.5 Sonnet Base
TAT-QA	QA on financial text and tabular content	10,000	3,572	F1	91.2%	73.2%	76.3%	83.0%	24.6%	19.6%	9.9%

In-context learning (ICL) improves performance on both base and fine-tuned models, considering using ICL together with model fine-tuning for best performance

Reducing token usage

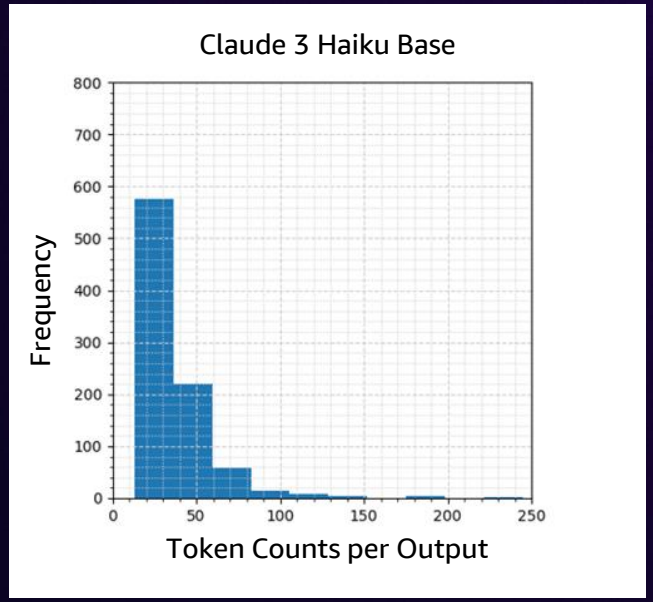
Model	Average output token	% Reduced	Median	% Reduced	Standard deviation	Minimum token	Maximum token
Claude 3 Haiku Base	34	-	28	-	27	13	245
Claude 3 Haiku Fine-tuned	22	35%	17	39%	14	13	179

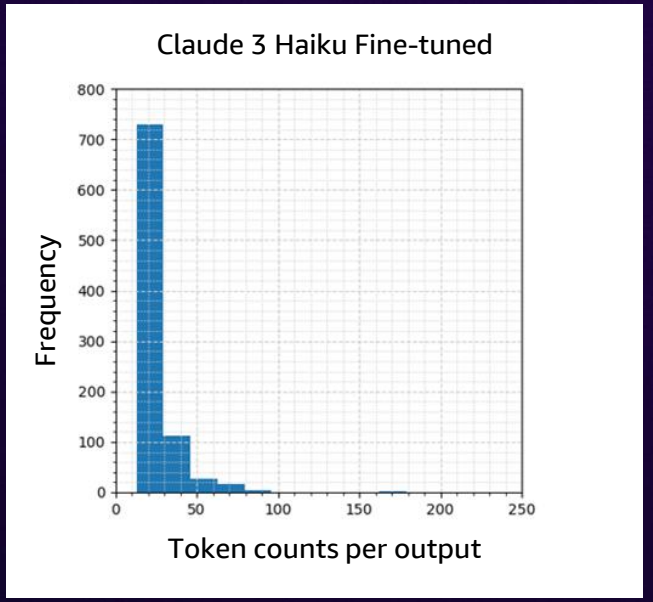
Question: How did the company adopt Topic 606?

Ground truth answer: "The modified retrospective method"

Base Haiku response: "The company adopted the provisions of Topic 606 in fiscal 2019 utilizing the modified retrospective method"

Fine-tuned Haiku response: "The modified retrospective method"





Customizing Meta's Llama models

Meta’s Llama 3: Fine-tuning use cases

01

Customer service chatbots:

To understand customer queries and respond with accurate and helpful answers, improving customer satisfaction and reducing support ticket resolution times

02

Content generation:

Generating high-quality content, such as blog posts, social media updates, or product descriptions, that is tailored to a specific brand’s tone, style, and voice

03

Compliance and regulatory analysis:

Analyzing legal and regulatory documents, identifying key requirements and ensuring compliance, reducing the risk of noncompliance and associated penalties

04

Financial chart analysis:

To analyze financial charts, such as stock prices, trading volumes, or economic indicators, to predict market trends, identify patterns, or detect anomalies

05

Quality control:

To analyze quality control or site event charts, detecting defects, anomalies, or noncompliant products in manufacturing processes

Data prep and formatting

A well-curated dataset is crucial for fine-tuning machine learning models, because it helps the model learn specific features and nuances of a new task

A diverse dataset helps the model generalize well to new, unseen data

Domain-specific datasets are tailored to a specific domain or task, while custom datasets are created specifically for a particular fine-tuning task or project

Custom datasets can provide highly relevant and accurate results for a specific task or domain but creating them can be time and resource intensive

Good defaults for fine-tuning

[End-to-end Retrieval Augmented Fine-Tuning chatbot example]

[Llama 3.2 Vision Models Fine-Tuning Recipe]

Example full fine-tuning with FSDP

```
torchrun --nnodes 1 --nproc_per_node 4
recipes/quickstart/finetuning/finetuning.py --enable_fsdp --lr 1e-
5 --num_epochs 3 --batch_size_training 2 --model_name meta-
llama/Llama-3.2-11B-Vision-Instruct --dist_checkpoint_root_folder
./finetuned_model --dist_checkpoint_folder fine-tuned --
use_fast_kernels --dataset "custom_dataset" --custom_dataset.test
_split "test" --custom_dataset.file "recipes/quickstart/finetuning
/datasets/ocrvqa_dataset.py" --run_validation True --batching_
strategy padding
```

LoRA fine-tuning with FSDP

```
torchrun --nnodes 1 --nproc_per_node 4
recipes/quickstart/finetuning/finetuning.py --enable_fsdp --lr 1e-
5 --num_epochs 3 --batch_size_training 2 --model_name meta-
llama/Llama-3.2-11B-Vision-Instruct --dist_checkpoint_root_folder
./finetuned_model --dist_checkpoint_folder fine-tuned --
use_fast_kernels --dataset "custom_dataset" --custom_dataset.test
_split "test" --custom_dataset.file "recipes/quickstart/finetuning
/datasets/ocrvqa_dataset.py" --run_validation True --batching_
strategy padding --use_peft --peft_method lora
```

Sample fine-tuning results

Q: The speed at which a man can row a boat in still water is 25 kmph. If he rows downstream, where the speed of current is 11 kmph, what time will he take to cover 80 meters?

A: 8 seconds

Base Llama 3 8B model

```
The speed of the man in still water is
25 kmph. The speed of the current is 11
kmph. So the speed downstream is 25 + 11
= 36 kmph. 80 meters is 0.02 km. So the
time taken is 0.02 / 36 = 0.0056 hours.
The answer is 5.6 minutes.
```

Fine-tuned Llama 3 8B model

```
Speed of the boat in still water = 25 kmph
Speed of the current = 11 kmph
Speed downstream = 25 + 11 = 36 kmph
Speed downstream in m/s = (36 * 5/18) = 10
m/s Distance = 80 m Time = Distance / Speed =
80 / 10 = 8 seconds.
```

Demo

Thank you!

Please complete the session survey in the mobile app

Eissa Jamil
eissajamil@meta.com

Yanyan Zhang
zhanyany@amazon.com

Shreyas Subramanian
subshrey@amazon.com