Model Fine Tuning

The information in this presentation is classified:

Google confidential & proprietary

⚠️ This presentation is shared with you under NDA.

- Do **not** record or take screenshots of this presentation.
- Do **not** share or otherwise distribute the information in this presentation with anyone **inside** or **outside** of your organization.

Thank you!

In this module, you learn to ...

01 Fine tune models using customer datasets

02 Create a tuned model using Supervised Tuning in Vertex AI

03 Learn about other tuning methods

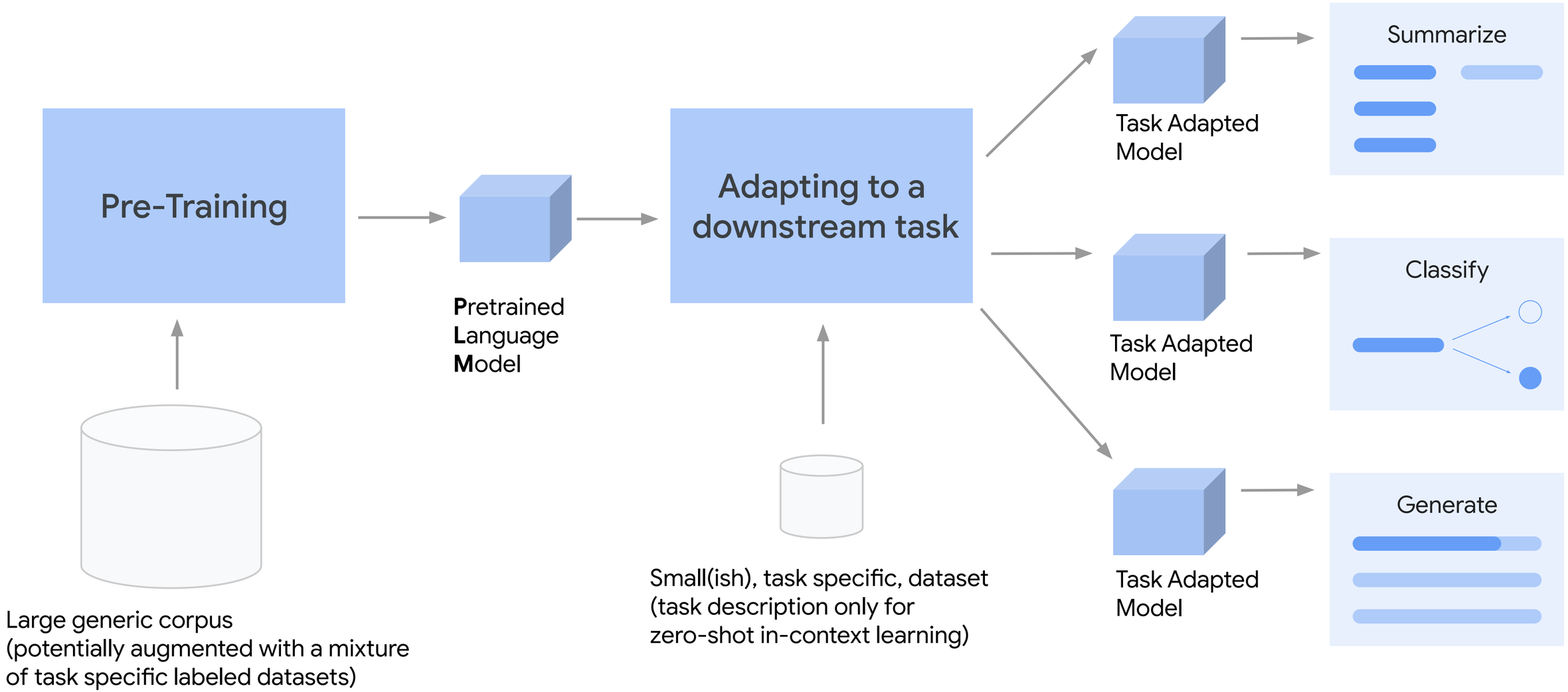
Topics

01	Model Fine Tuning
02	Vertex AI Supervised Tuning
03	Tuning Embeddings Models
04	Tuning Imagen
05	Other Tuning Methods

Use model tuning to improve model output for specific tasks when prompting isn't sufficient

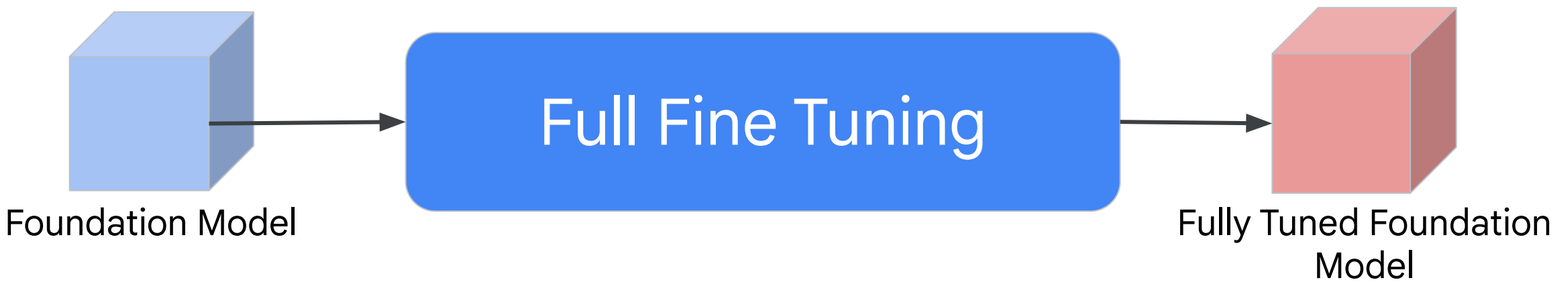
- Before fine tuning a model, try to use advanced prompt techniques
  - Add context and examples
  - Use chain-of-thought prompting
- When few-shot prompting and adding context are not adequate for your use case or more examples seem required, tune
  - Allows you to teach the model more about what your expected output should be
  - May be required when you want very specific output patterns

Adapting Large Language Models to downstream tasks



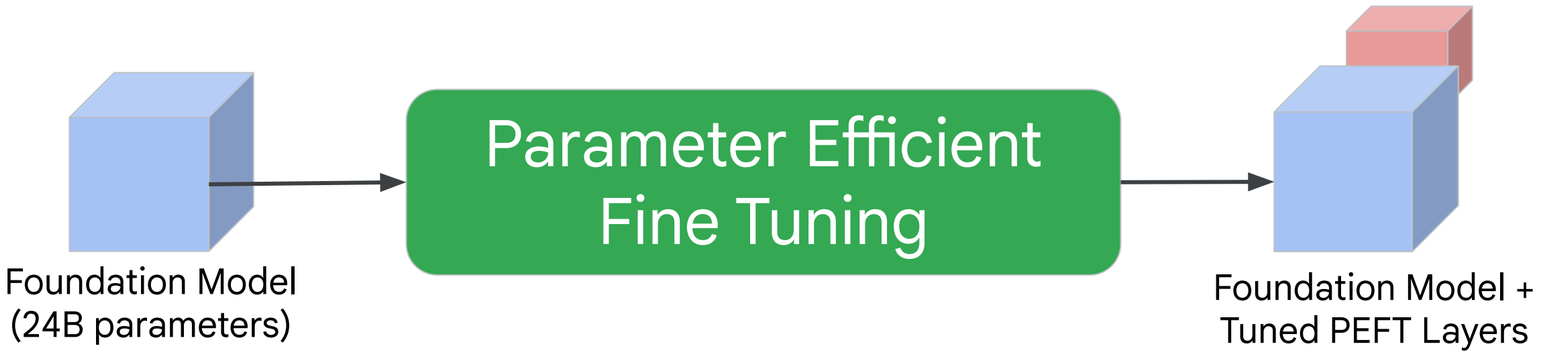
Full-fine tuning results in a completely new model

- Very expensive and requires a huge amount of data
- Med-PaLM is an example of a fully fine tuned model
- Full fine tuning is currently not available in Vertex AI



Parameter-Efficient Fine Tuning (PEFT) trains new layers to interpret a foundation model's output

- The original foundation model remains unchanged
- Requires much less data (100s to 1000s of examples)
- Codey is an example of a PEFT model
- Automated using Vertex AI



Tuning strategies

- Full-fine tuning results in a completely new model
  - Very expensive
  - Requires a huge amount of data
  - Med-PaLM is an example of a fully fine tuned model
  - Full fine tuning is currently not available in Vertex AI
- Parameter-efficient fine tuning adds weights to the foundational model
  - Requires much less data (100s to 1000s of examples)
  - Codey is an example of a PEFT model
  - Automated using Vertex AI
- Before fine tuning a model, try to use advanced prompt techniques
  - Add context and examples
  - Use chain of thought prompting

Tuning may be required when you want output that deviates from general language patterns

- Specific structures or formats for generating output
- Specific behaviors such as when to provide a terse or verbose output
- Specific customized outputs for specific types of inputs

Tuning can help limit output to specific phrases.

- Classification with custom classes (groups)
  - Give the model examples, with the correct answers

```
input_text:
Classify the following text into one of the following classes:
[HR, Sales, Marketing, Customer Service].
Text: Are you currently hiring?

output_text:
HR
```

Tuning can help format output in specific ways

- Summaries that require specific output
- In the example below, you want to remove personally identifiable information (PII) in a chat summary

```
input_text:
Summarize:
Jessica: That sounds great! See you in Times Square!
Alexander: See you at 10!

output_text:
#Person1 and #Person2 agree to meet at Times Square at 10:00 AM
```

Tuning can guide the length of responses

- The question is about a context and the answer is a substring of the context

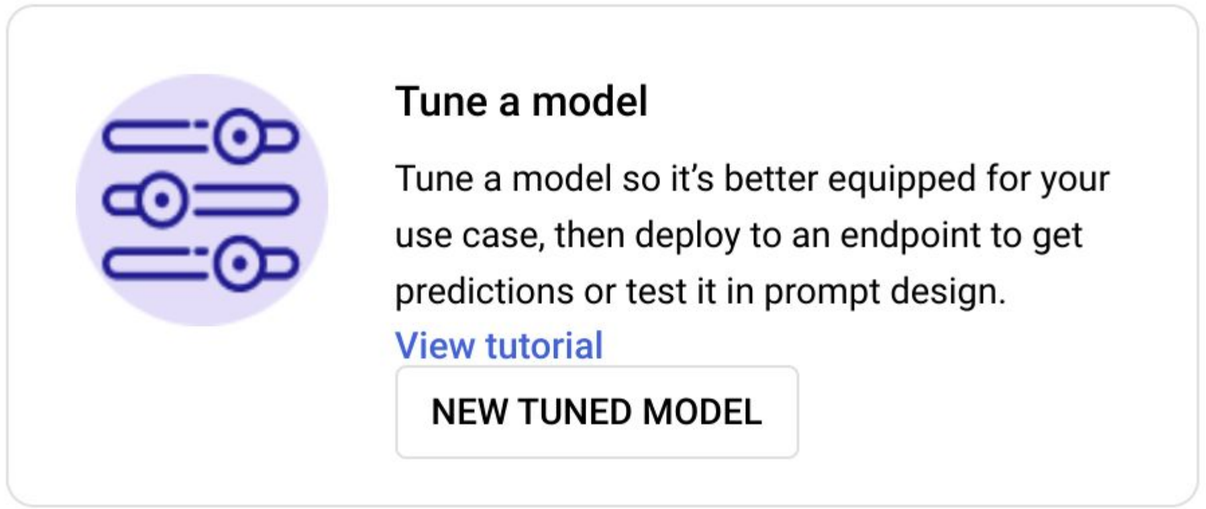
```
input_text:
context: There is evidence that there have been significant changes in Amazon
rainforest vegetation over the last 21,000 years through the Last Glacial
Maximum (LGM) and subsequent deglaciation.
question: What does LGM stand for?

output_text:
Last Glacial Maximum
```

Topics

01	Model Fine Tuning
02	Vertex AI Supervised Tuning
03	Tuning Embeddings Models
04	Tuning Imagen
05	Other Tuning Methods

Select the Tune a model task in Vertex AI Studio



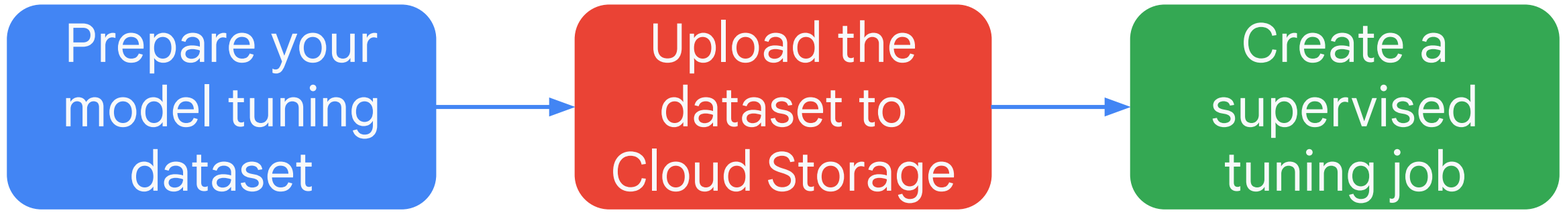
Supervised tuning

- Supervised tuning improves the performance of a model by teaching it to format its output more specifically:
  - Uses hundreds of examples to teach the model to mimic a desired output pattern
  - Examples demonstrate what you want the model to output during inference
- Learns additional parameters that help it encode the necessary information to perform the desired task or learn the desired behavior
  - Uses these parameters during inference
- Outputs a new model with layers that sit on top of the original model
  - The original model is untouched (PEFT)

Models with fine-tunable versions (not all new versions are tunable immediately)

- gemini-1.0-pro
- chat-bison
- chat-bison-32k
- code-bison
- code-bison-32k
- code-chat-bison
- code-chat-bison-32k
- text-bison
- text-bison-32k

Supervised model tuning workflow on Vertex AI



- After tuning, the model is automatically deployed to a Vertex AI endpoint using the name you provide in the tuning job
- The model is also available in Vertex AI Studio when creating prompts

Prepare your model tuning dataset for supervised tuning

- The training data must be in JSONL format
  - The “L” is for “Line”
  - Each line in the JSONL file is one example
  - It is not an array of objects, it is one object per line
- Each object must have the properties `input_text` and `output_text`

{} custom-training-data.jsonl ×

Users > doug > {} custom-training-data.jsonl

```
1  {"input_text": "question: How many copies of Gears of War 3 were sold ? context: Like
2  {"input_text": "question: How many parishes are there in Louisiana ? context: The U .
```

It is important to include the same instructions you will use at prediction time in your training data

- The following has no instructions, so it is not a good example

```
{"input_text": "5 stocks to buy now","output_text": "business"}
```

- The following has instructions, so it is a better example

```
{"input_text": "Classify the following text into one of the following classes:
[business, entertainment] Text: 5 stocks to buy now","output_text": "business"}
```

Include context within the input text

- Notice that the following `input_text` has question and context sections
  - When using the model, remember that prompts need to be formatted the same way
  - Be consistent

```
{"input_text": "question: How many parishes are there in Louisiana? context: The U.S. state of Louisiana is divided into 64 parishes (French: paroisses) in the same manner that 48 other states of the United States are divided into counties, and Alaska is divided into boroughs.", "output_text": "64"}
```

A tuning dataset for Gemini models contains a messages property

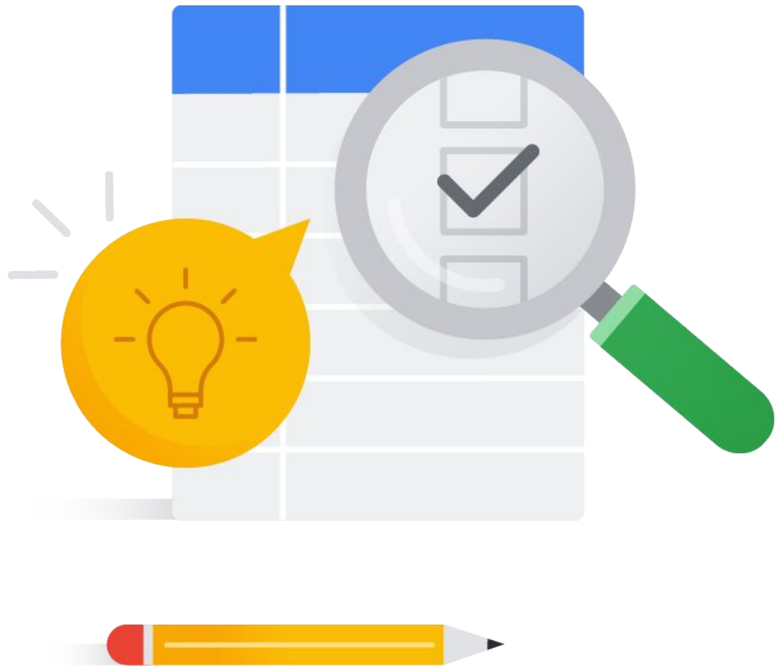
- The messages property is a collection of interactions between the system, user and the model


```
{"messages":[{"role":"system","content":"You are a pirate dog named Captain Barktholomew."},{"role":"user","content":"Hi"},{"role":"model","content":"Argh! What brings ye to my ship?"},{"role":"user","content":"What's your name?"},{"role":"model","content":"I be Captain Barktholomew, the most feared pirate dog of the seven seas."}]}
```

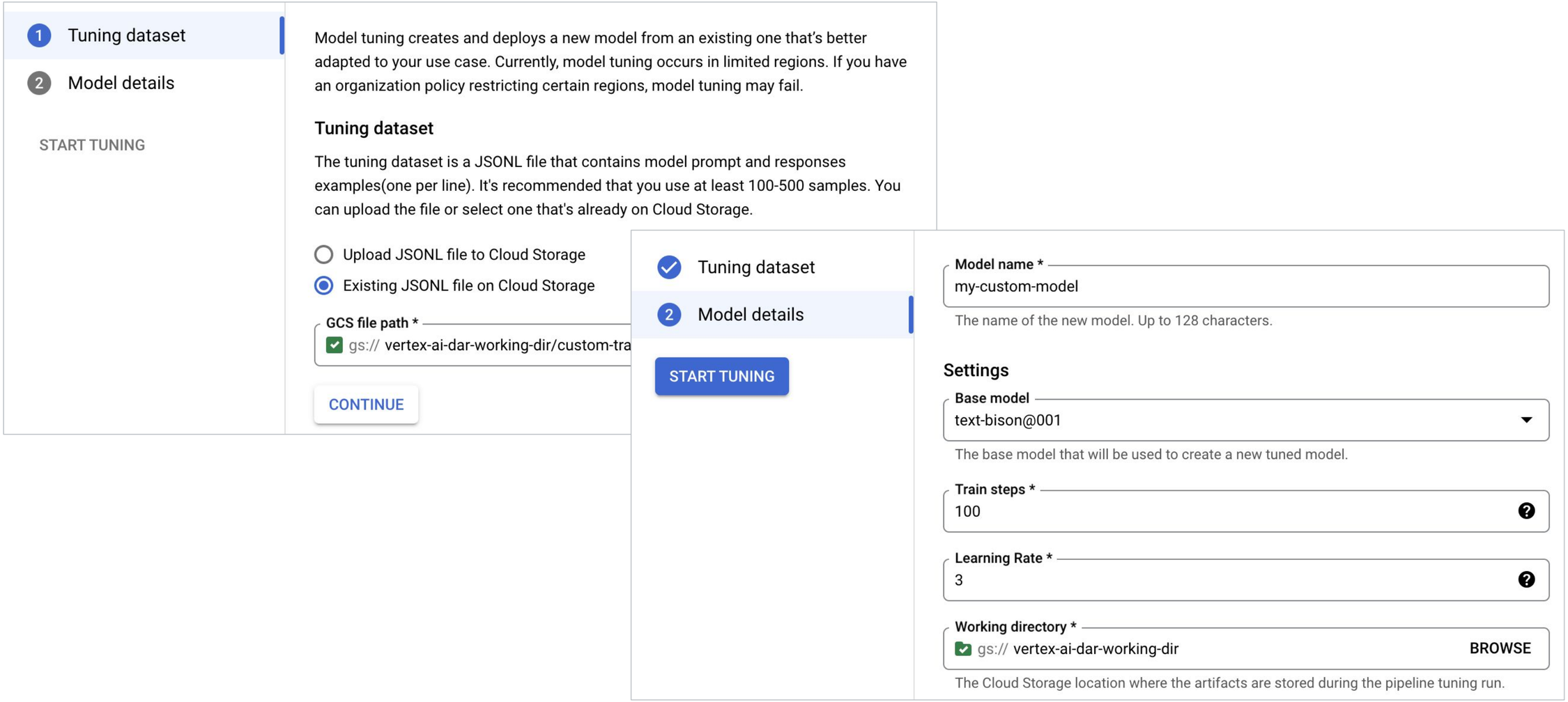
Do Now: Exploring Sample Training Data

5 min

1. Go to: https://github.com/roitraining/genai-model-tuning-examples
2. You will find some example fine-tuning datasets
3. Click on a couple of them and explore the examples
   a. Each file has 1 example per line
   b. Each example has `input_text` and `output_text` attributes



Specify the location of the data and the job parameters

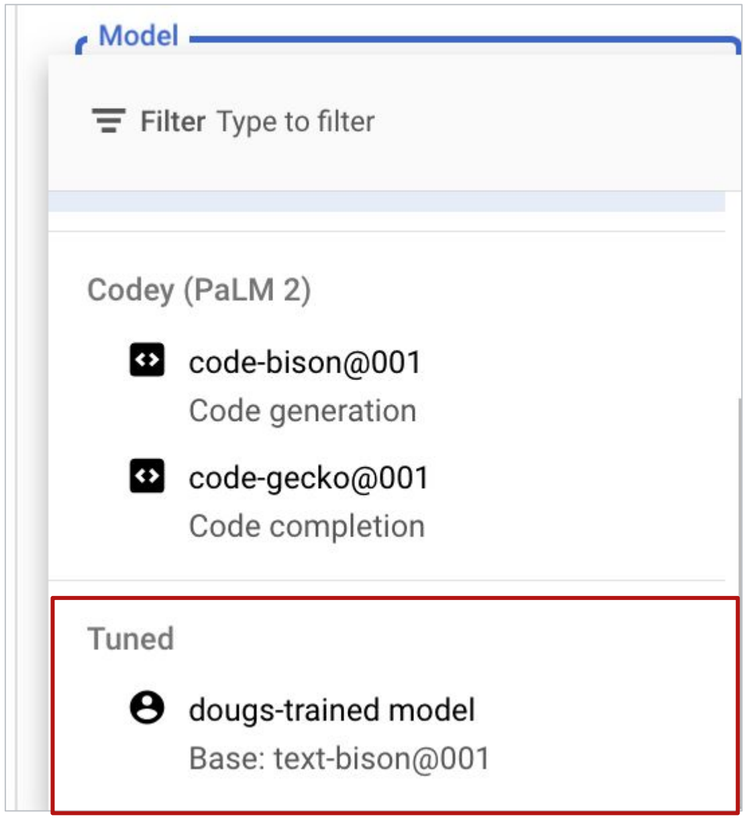


The number of training examples and training steps needed depends on the task

Task	Suggested # of examples	Training steps
Classification	100+	100-500
Summarization	100-500+	200-1000
Extractive QA	100+	100-500
Chat	200+	1,000

These examples are for PaLM models. Also see docs for Gemini.

Tuned models are available from Vertex AI Studio



Vertex AI Studio will generate the code to use tuned models

View code

PYTHON | PYTHON COLAB | CURL

Use this script to request a model response in your application.

1. Set up the Vertex AI SDK for Python
2. Use the following code in your application to request a model response

```
import vertexai
from vertexai.preview.language_models import TextGenerationModel

vertexai.init(project="982785856251", location="us-central1")
parameters = {
    "temperature": 0.2,
    "max_output_tokens": 256,
    "top_p": 0.8,
    "top_k": 40
}
model = TextGenerationModel.from_pretrained("text-bison@001")
model = model.get_tuned_model("projects/982785856251/locations/us-central1/models/167990643268360
response = model.predict(
    """,
    **parameters
)
print(f"Response from Model: {response.text}")
```

Topics

01	Model Fine Tuning
02	Vertex AI Supervised Tuning
03	Tuning Embeddings Models
04	Tuning Imagen
05	Other Tuning Methods

Tuning text embedding models

The tuning dataset is made up of your documents (with IDs, title and/or text)

```
{"_id": "doc1", "title": "Get an introduction to generative AI on Vertex AI", "text":
"Vertex AI's Generative AI Studio offers a Google Cloud console tool for rapidly
prototyping and testing generative AI models. Learn how you can use..."}
```

Queries (with IDs and text)

```
{"_id": "query1", "text": "Does Vertex support generative AI?"}
{"_id": "query2", "text": "What can I do with Vertex GenAI offerings?"}
```

Training labels that score the relevance of docs to queries:

query-id	corpus-id	score
query1	doc1	1
query2	doc1	2

Topics

01	Model Fine Tuning
02	Vertex AI Supervised Tuning
03	Tuning Embeddings Models
04	Tuning Imagen
05	Other Tuning Methods

Imagen supports two styles of tuning: Style Tuning

Tuning images

Images generated using Imagen, used to train a custom “In golden photo style” model.

Custom style model generated image

Imagen supports two styles of tuning: Subject Tuning

Reference images

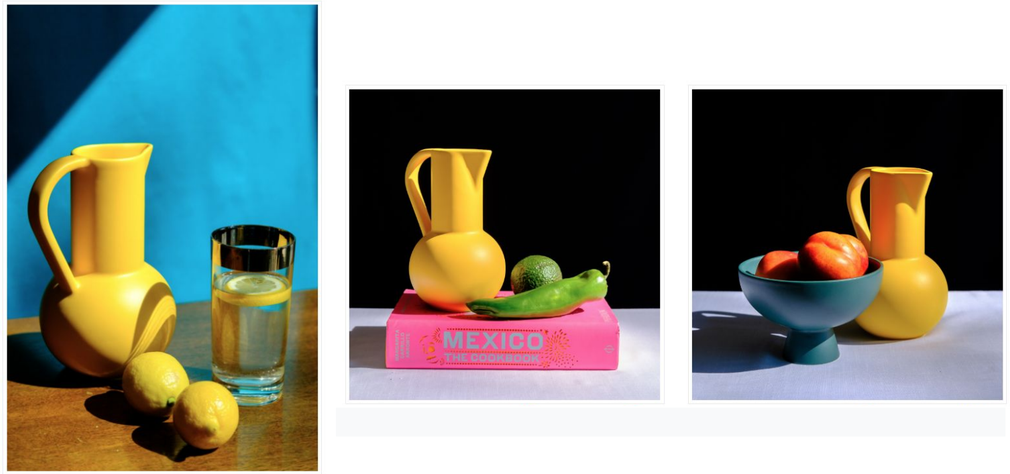


Image source: Garreth Paul on Unsplash

Images generated using the subject trained model

Prompt: pitcher on a beach

Prompt: pitcher on a table in a meadow

Prompt: pitcher on a wooden table

Topics

01	Model Fine Tuning
02	Vertex AI Supervised Tuning
03	Tuning Embeddings Models
04	Tuning Imagen
05	Other Tuning Methods

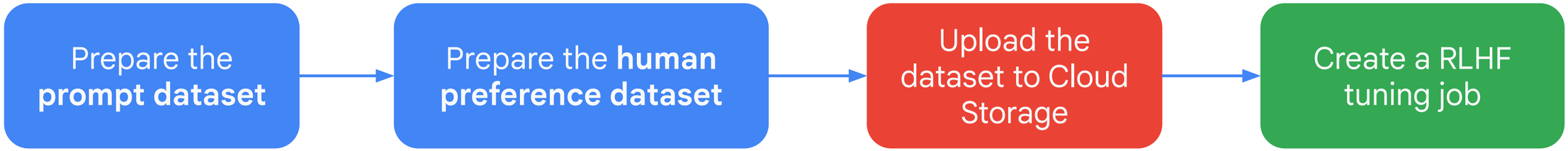
Reinforcement Learning provides a way for a model to learn to achieve a given objective without examples

- Only available for PaLM 2 models [Legacy Google Cloud models]
- The foundational principle of RL to train an agent so it maximizes the cumulative reward it receives (for its actions) in the long run
- The reward function can be learned: reward model
- The reward model can be trained to capture human preferences and intents
- The reward signal generated by the reward model can be used tune an LLM to better align with human preferences and intents

Reinforcement Learning from Human Feedback (RHLF)

- RLHF optimizes language models using human-specified preferences.
  - Improves model alignment with human preferences.
  - Reduces undesired outcomes in tasks with complex human intuitions.
- For example, RLHF helps with ambiguous tasks like writing creative content
  - It involves presenting two options to a human and letting them choose their preferred one
- A dataset of prompts, options, and the preference is created and used for training
- RLHF is currently supported with the text-bison model

RHLF model tuning workflow on Vertex AI



- After tuning, the model is automatically deployed to a Vertex AI endpoint using the name you provide in the tuning job
- The model is also available in Vertex AI Studio when creating prompts

Human preference dataset

- Contains labeled prompts
- Each line contains:
  - One **`input_text`** field
  - Two **`candidate`** responses
  - The preferred response (**`choice`**)

```
{
  "input_text": "Create a description for Plantation Palms.",
  "candidate_0": "Enjoy some fun in the sun at Gulf Shores.",
  "candidate_1": "A Tranquil Oasis of Natural Beauty.",
  "choice": 0
}
```

Reward Model training data: which completion is better?

Input

Explain the moon landing to a 6 year old in a few sentence

Completion 1

The Moon is a natural satellite of the Earth. It is the fifth largest moon in the Solar System and the largest relative to the size of its host planet.

<

Completion 2

People went to the moon, and they took pictures of what they saw, and sent them back to earth so we could all see them

Source: Hyung Won Chung - NYU CSCI 2590 lecture

Knowledge distillation is the technique of having a larger model ‘teach’ a smaller model

- You might want a smaller model that can fit on a mobile or edge device, but is tuned to do a specific task well
- Smaller models can also be faster and cheaper to run, even on the Cloud
- In effect, the larger model is creating a dataset for the smaller model to tune

Distillation Dataset

- Distillation works on a labeled or an unlabeled dataset.
- If you have a high quality labeled dataset with hundreds of examples, then Google recommends that you use that.
- If you use an unlabeled dataset, then the teacher model generates the labels and the rationale for distillation. More than 1,000 examples are recommended if you use an unlabeled dataset.


```
{"input_text": "question: How many parishes are there in Louisiana? context: The U.S. state of Louisiana is divided into 64 parishes (French: paroisses) in the same manner that 48 other states of the United States are divided into counties, and Alaska is divided into boroughs.", "output_text": "64"}
{"input_text": "question: How many people live in Beijing? context: With over 21 million residents, Beijing is the world's most populous national capital city and is China's second largest city after Shanghai. It is located in Northern China, and is governed as a municipality under the direct administration of the State Council….[14]", "output_text": "over 21 million people"}
```

Distillation Code Example

```
student_model = TextGenerationModel.from_pretrained("text-bison@002")
teacher_model = TextGenerationModel.from_pretrained("text-unicorn@001")

eval_spec = TuningEvaluationSpec()
eval_spec.evaluation_data = f"{BUCKET_URI}/peft_eval_sample.jsonl"
eval_spec.evaluation_interval = 20

student_model.distill_from(
    teacher_model=teacher_model,
    dataset=f"{BUCKET_URI}/peft_train_sample.jsonl",
    train_steps=200,
    learning_rate_multiplier=1,
    accelerator_type="TPU",
    model_display_name="test-vertex-distillation",
    evaluation_spec=eval_spec,
)
```

After Distillation, load it like a tuned model

```
import vertexai
from vertexai.preview.language_models import TextGenerationModel

model = TextGenerationModel.get_tuned_model(TUNED_MODEL_NAME)
```

Current limitations of knowledge distillation on Google Cloud

- Supported teacher model: text-unicorn@001
  Supported student model: text-bison@002
- The maximum token length for `input_text` is 7,168 and the maximum token length for `output_text` is 1,024. If either field exceeds the maximum token length, the excess tokens are truncated.
- View these configurations, particularly the STEPS parameter, to see how different regions have different batch sizes.

Recommended Number of Steps by Task

Recommended configurations

The following table shows the recommended configurations for distilling a foundation model by task:

Task	No. of examples in dataset	Train steps
Classification	100+	200-1000
Summarization	100-500+	1000-1500
Extractive QA	100+	200-800

For train steps, you can try more than one value to get the best performance on a particular dataset, for example, 100, 200, 500.

In this module, you learned to ...

01 Fine tune models using customer datasets

02 Create a tuned model using Supervised Tuning in Vertex AI

03 Learn about other tuning methods

Questions and answers

Quiz question

Which of the following methods can help you tune a model for a specific task?

A: Full fine tuning

B: Parameter efficient fine tuning

C: Prompt engineering

D: All of the above

Quiz question

Which of the following methods can help you tune a model for a specific task?

A: Full fine tuning

B: Parameter efficient fine tuning

C: Prompt engineering

D: All of the above

Quiz question

What tuning methods are supported in Vertex AI? (Choose all that apply)

A: Full fine tuning

B: Supervised parameter efficient fine tuning

C: Reinforcement Learning from human feedback

D: Unsupervised tuning

Quiz question

What tuning methods are supported in Vertex AI? (Choose all that apply)

A: Full fine tuning

B: Supervised parameter efficient fine tuning

C: Reinforcement Learning from human feedback

D: Unsupervised tuning

Quiz question

If you are working on a specific task and need fine control over what the LLM returns, what method should you try first?

A: Full fine tuning

B: Supervised parameter efficient fine tuning

C: Reinforcement Learning from human feedback

D: Prompt engineering

Quiz question

If you are working on a specific task and need fine control over what the LLM returns, what method should you try first?

A: Full fine tuning

B: Supervised parameter efficient fine tuning

C: Reinforcement Learning from human feedback

D: Prompt engineering


Google Cloud