Microsoft Azure

Unlocking business value with fine-tuning

Contents

Executive Summary

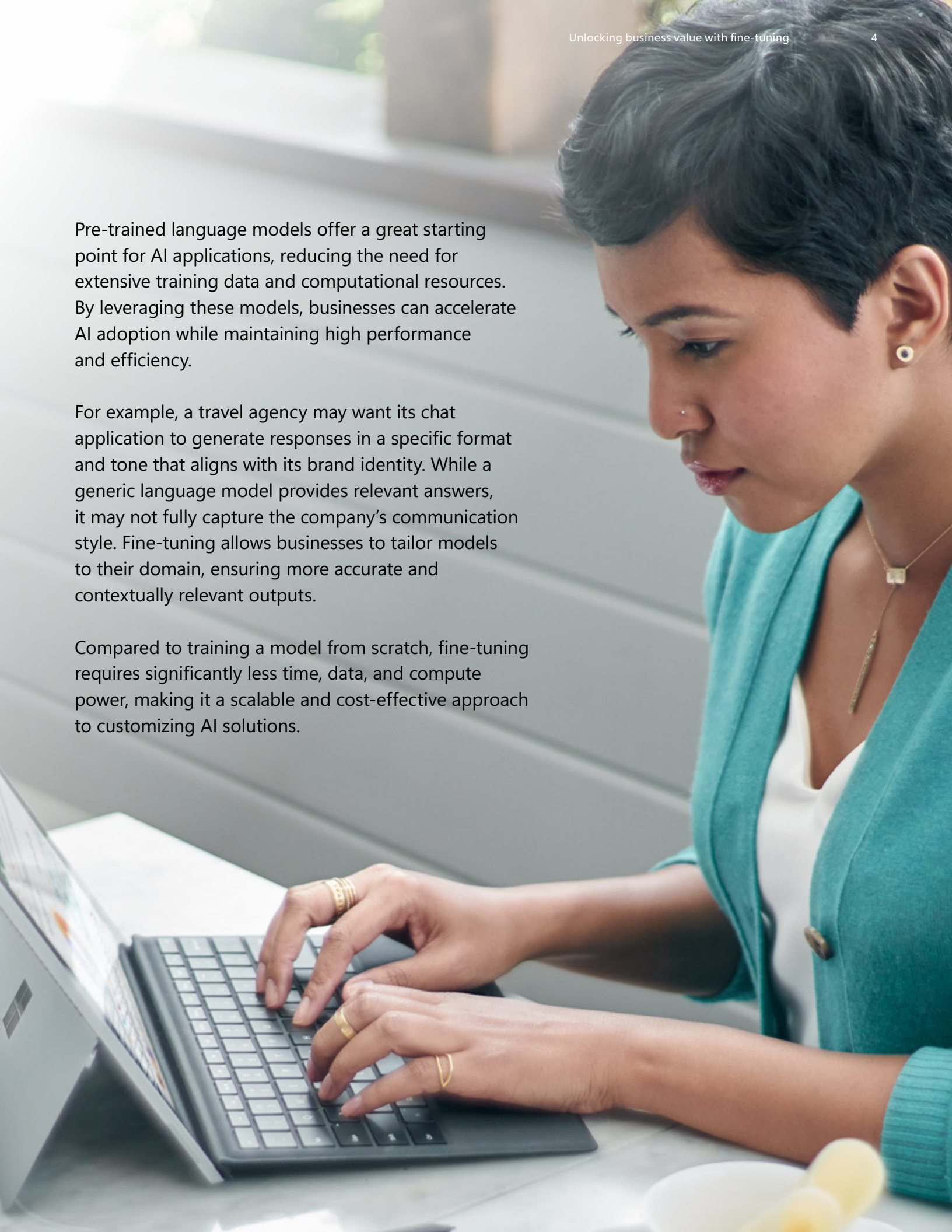
Fine-tuning pre-trained language models unlocks new levels of accuracy and efficiency for enterprise AI applications. By training models on domain-specific data, businesses can enhance contextual relevance, reduce token usage, and accelerate response times—delivering AI solutions that align more closely with their operational needs.

This approach provides significant advantages over using generic models or training from scratch. Fine-tuned models require shorter prompts, which can lower per-query costs depending on the use case while improving output quality.

A hybrid strategy combining fine-tuning with retrieval-augmented generation further enhances AI performance. By integrating real-time knowledge retrieval with model customization, enterprises can reduce hallucinations, refine brand-specific communication, and optimize structured outputs.

However, fine-tuning requires careful execution. High-quality, representative training data is essential, and hyperparameter tuning plays a critical role in avoiding overfitting. Best practices include starting with smaller models, iterating through experimental phases, and leveraging well-structured datasets.

For organizations looking to scale AI capabilities efficiently, fine-tuning presents a powerful solution—enabling specialized AI models that drive better business outcomes.



Pre-trained language models offer a great starting point for AI applications, reducing the need for extensive training data and computational resources. By leveraging these models, businesses can accelerate AI adoption while maintaining high performance and efficiency.

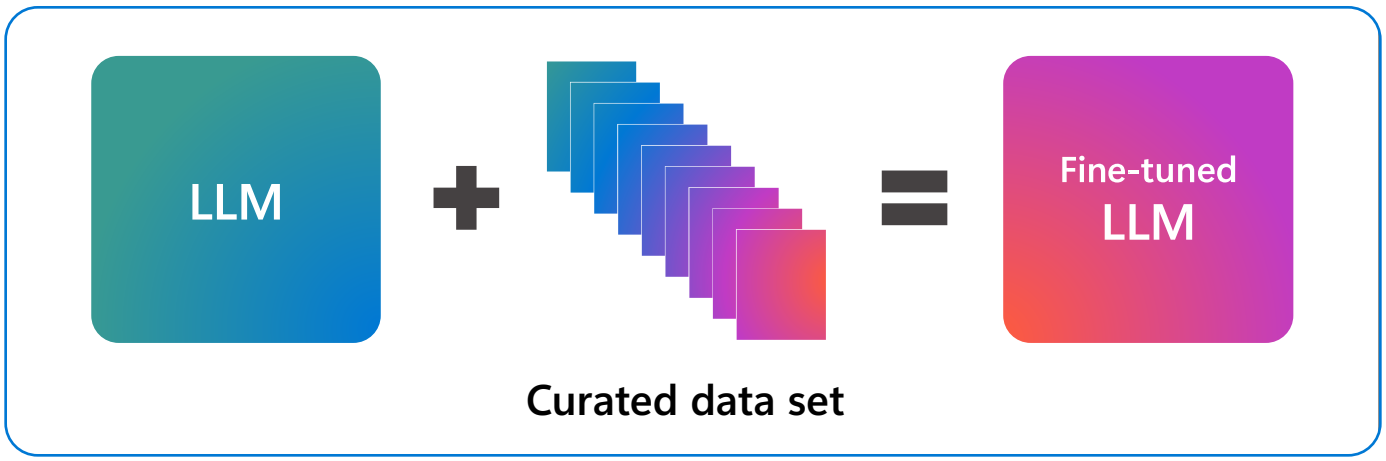
For example, a travel agency may want its chat application to generate responses in a specific format and tone that aligns with its brand identity. While a generic language model provides relevant answers, it may not fully capture the company's communication style. Fine-tuning allows businesses to tailor models to their domain, ensuring more accurate and contextually relevant outputs.

Compared to training a model from scratch, fine-tuning requires significantly less time, data, and compute power, making it a scalable and cost-effective approach to customizing AI solutions.

Why fine-tuning matters for Enterprises

Fine-tuning refines a pre-trained model by further training on a task-specific or new dataset. It leverages the knowledge gained by a model during its initial training on a large, diverse dataset and refines it to perform a specific task or improve its performance using a smaller dataset. This approach is often more efficient and effective than training a new model from scratch, especially for specialized tasks.

Fine-tuning is crucial for meeting customer-specific needs, as it allows organizations to adapt pre-trained models to their unique datasets and requirements. This customization enhances performance, reduces token costs, and ensures that AI solutions are aligned with business goals.



Key benefits of fine-tuning

Enhanced accuracy and relevance

Fine-tuning improves the model's performance on particular tasks by training it with task-specific data. This often results in more accurate and relevant high-quality outputs compared to using general prompts.

Unlike few-shot learning, where only a limited number of examples can be included in a prompt, fine-tuning allows you to train the model on an additional dataset. This helps the model learn more nuanced patterns and improves task performance.

Efficiency and potential cost savings

Fine-tuned models require shorter prompts because they have already been trained on relevant examples. This reduces the number of tokens needed in each request, which can lead to cost savings depending on the use case.

Since fine-tuned models need fewer examples in the prompt, they process requests faster, resulting in quicker response times.

Scalability and specialization

Fine-tuning leverages the extensive pre-training of language models and hones their capabilities for specific applications, making them more efficient and effective for targeted use cases.

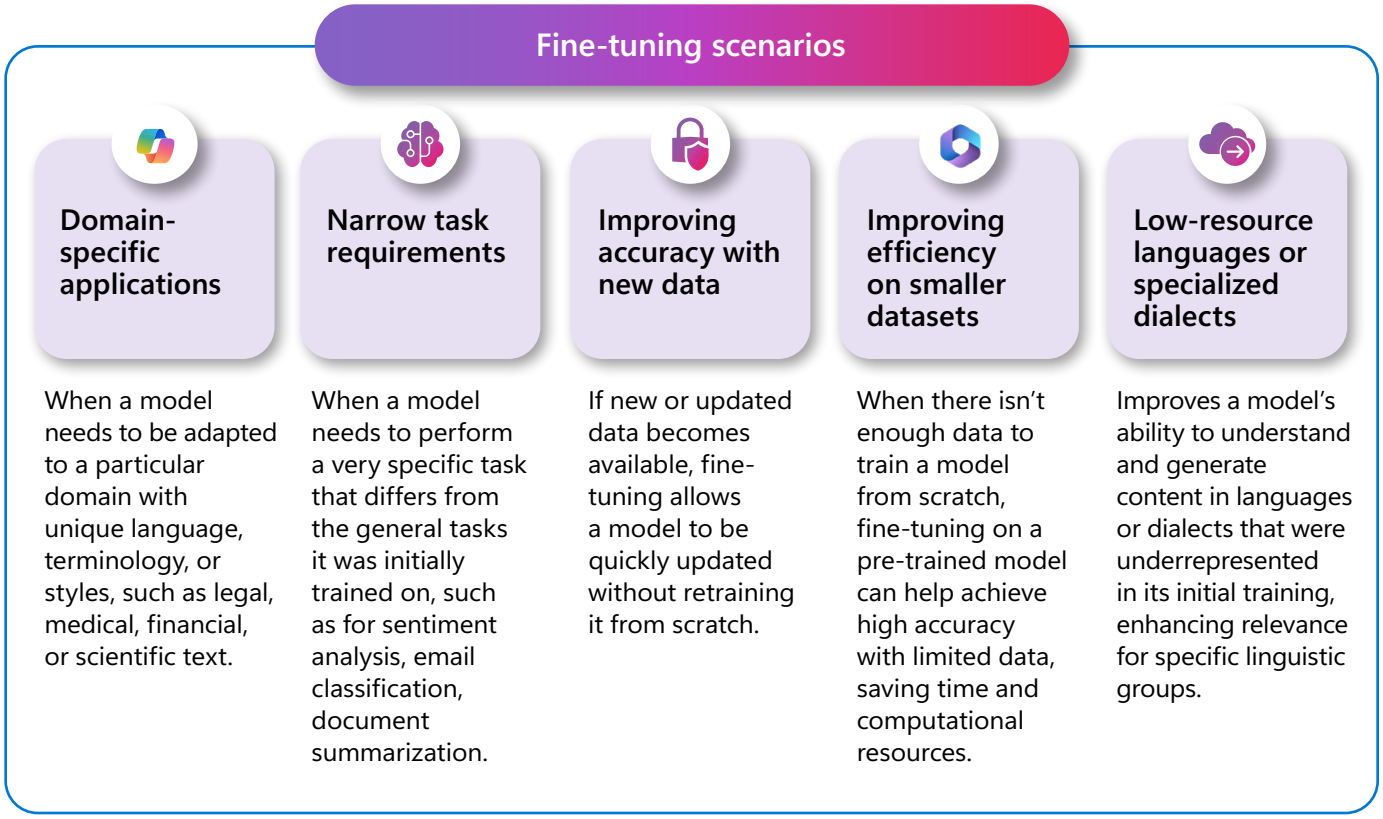
Fine-tuning smaller models can achieve performance levels comparable to larger, more expensive models for specific tasks. This approach reduces computational costs and increases speed, making it a cost-effective scalable solution for deploying AI in resource-constrained environments.

Enhanced performance and accuracy	Efficiency and cost savings	Scalability and specialization
Task-specific optimization Nuanced learning	Token optimization Lower latency requests	Domain-specific adaptation lightweight, high-performance models

Fine-tuning is a smart investment that optimizes model efficiency and can reduce operational costs over time. While it requires an initial training cost, it can lower per-query token usage depending on the use case, while also improving accuracy and response quality. By tailoring the model to your specific needs, you create a faster, potentially more cost-effective solution that delivers better outcomes while maintaining scalability.

When to fine-tune

Fine-tuning is suited for times when you have a small amount of data and want to improve the performance of your model.



Fine-tuning can be for different kinds of use cases - but they often fall into broader categories.

Reducing prompt engineering overhead: Many users begin with few-shot learning, appending examples of desired outputs to their system message. Over time, this can lead to increasingly long prompts, driving up token counts and latency. Fine tuning lets you embed these examples into the model by training on the expected outputs. This is particularly valuable in scenarios with numerous edge cases.

Modifying style and tone: Fine-tuning helps align model outputs with a desired style or tone, ensuring consistency in applications like customer service chatbots and brand-specific communication.

Generating outputs in specific formats or schemas: Models can be fine-tuned to produce outputs in specific formats or schemas, making them ideal for structured data generation, reports, or formatted responses.

Enhancing tool usage: While the chat completions API supports tool calling, listing many tools increases token usage and may lead to hallucinations. Fine-tuning with tool examples enhances accuracy and consistency, even without full tool definitions.

Enhancing retrieval-based performance: Combining fine-tuning with retrieval methods improves a model's ability to integrate external knowledge, perform complex tasks, and provide more accurate, context-aware responses. Fine-tuning trains the model to effectively use retrieved data while filtering out irrelevant information.

Optimizing for efficiency: Fine-tuning can also be used to transfer knowledge from a larger model to a smaller one, allowing the smaller model to achieve similar task performance with lower cost and latency. For example, production data from a high-performing model can be used to fine-tune a smaller, more efficient model. This approach helps scale AI solutions while maintaining quality and reducing computational overhead.

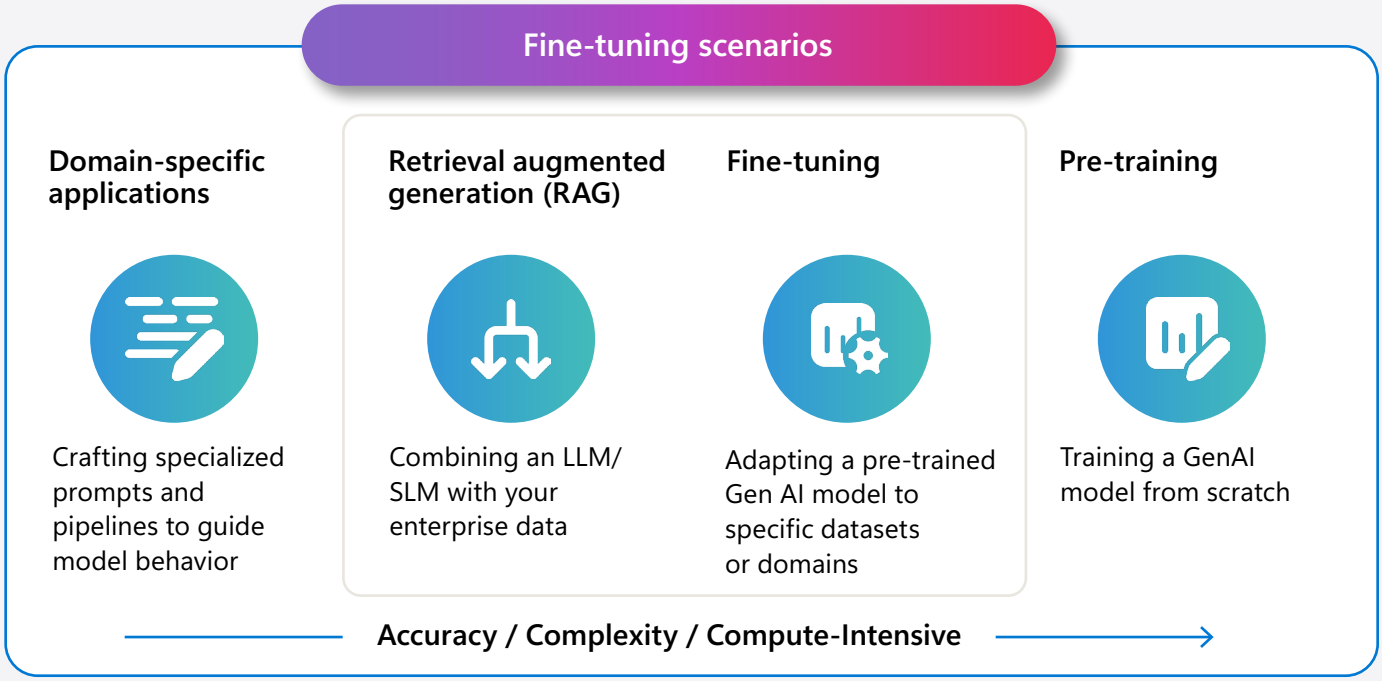
Optimizing models with prompt engineering, RAG, and fine-tuning

Optimizing generative AI models can be approached in multiple ways, depending on the complexity of the task and the level of customization required. The journey typically begins with prompt engineering, a straightforward method to refine model outputs without altering the underlying model. As needs become more sophisticated, techniques like retrieval-augmented generation (RAG) and fine-tuning provide greater control, enabling models to generate more accurate, context-aware, and domain-specific responses.

Prompt engineering is a quick and easy way to improve how the model acts, and what the model needs to know. It is a technique that involves designing prompts for natural language processing models. This process improves accuracy and relevancy in responses, optimizing the performance of the model.

When you want to improve the quality of the model even further, there are two common techniques that are used:

- **RAG** improves foundation model performance by retrieving data from external sources and incorporating it into a prompt. It allows businesses to achieve customized solutions while maintaining data relevance and optimizing costs. RAG is most commonly applied when you need the model's responses to be factual and grounded in specific data. For example, you want customers to ask questions about hotels that you're offering in your travel booking catalog.
- **Fine-tuning** retrains an existing foundation model using example data, resulting in a new "custom" model that has been optimized using the provided examples. It is especially useful when you want the model to follow a specific style, format, or tone in its responses.



You should start by evaluating the performance of a base model with prompt engineering and/or RAG to get a baseline for performance.

Having a baseline for performance without fine-tuning is essential for knowing whether or not fine-tuning has improved model performance. Fine-tuning with bad data makes the base model worse, but without a baseline, it's hard to detect regressions.

You should be able to clearly articulate a specific use case for fine-tuning and identify the model you hope to fine-tune.

You can also use a combination of optimization strategies, like RAG and a fine-tuned model, to improve your language application.

	RAG	Fine-tuning
Access to latest information	Dynamically retrieves real-time data from external sources.	Requires retraining to incorporate new information.
Response accuracy and performance	Grounded in external documents, works well for fact-based or knowledge-driven queries.	Excels at context-aware responses, specific tasks, and specialized workflows.
Computational efficiency	Lower training costs, updates occur at the retrieval layer.	Higher training costs but optimized inference speed.
Latency	Can introduce higher response time due to retrieval steps.	Faster inference after fine-tuning, as no external retrieval is needed.
Cost considerations	Lower costs upfront, but retrieval API costs may increase with usage.	Higher initial investment but lower long-term costs for repetitive tasks.
Hallucination risk	Reduces hallucinations by retrieving factual data.	Can still hallucinate if fine-tuned on incomplete or biased data.

You are ready for fine-tuning if...

You have experience with Prompt Engineering and RAG approaches.

You can share specific challenges and lessons learned from alternative techniques that were attempted before fine-tuning.

You have conducted quantitative assessments of baseline performance, where possible. You have clear examples of previous optimization attempts, detailing what was tested and how performance was measured.

- You have identified limitations of the base model, such as:
- Inconsistent performance on edge cases
- Insufficient context window for few-shot learning
- High latency impacting real-time use cases

Hybrid approach with RAG and fine-tuning

The hybrid approach of using RAG and fine-tuning together, often referred to as retrieval augmented fine-tuning (RAFT), combines the strengths of both techniques to enhance the performance of LLMs.

Why adopt a hybrid approach

Combining RAG and fine-tuning, offer a powerful synergy that addresses the limitations of individual techniques. This involves fine-tuning a model with specific data to create a foundation of high accuracy, which can then be supplemented with RAG's up-to-date information retrieval. The result is highly accurate, relevant, and context-aware outputs.

Integrating RAG with fine-tuning creates a model that leverages both real-time knowledge retrieval and deep customization, offering:

Improved accuracy and relevance – Fine-tuning personalizes responses, while RAG ensures the AI accesses the latest information.

Reduced hallucinations – RAG reduces model-generated misinformation by grounding responses in real-world data.

Lower computational costs – Instead of retraining a model frequently, businesses can use RAG to inject fresh knowledge dynamically.

Better compliance and control – Fine-tuning refines model behavior, while RAG allows retrieval from vetted, enterprise-approved data sources.

When to use a hybrid approach

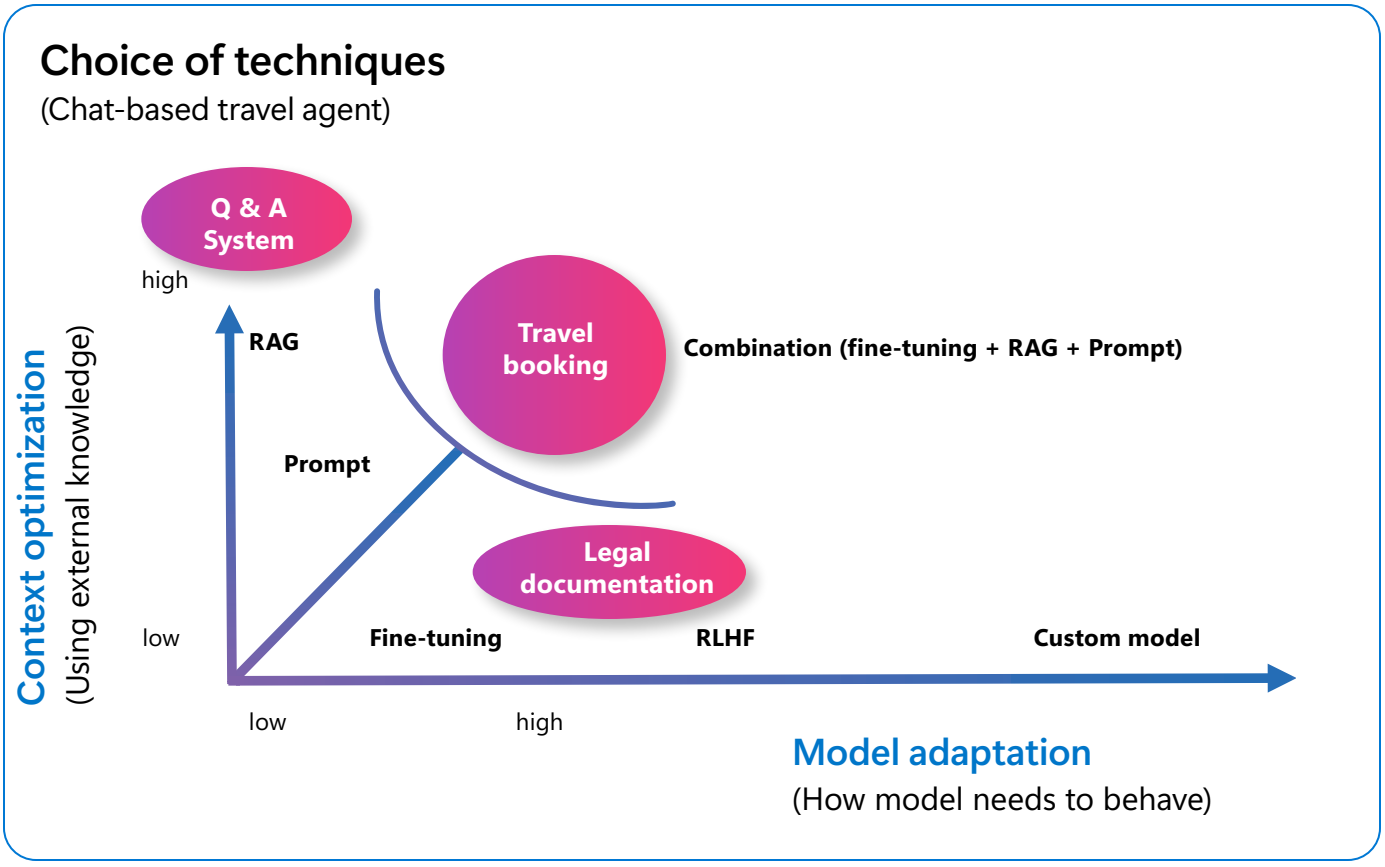
A study explored the trade-offs between RAG and fine-tuning for incorporating proprietary and domain-specific data into LLMs. Researchers developed a pipeline that extracted information from PDFs, generated question-answer pairs for fine-tuning, and evaluated performance using GPT-4.

Their findings on an agricultural dataset demonstrated how RAG and fine-tuning can complement each other. Fine-tuning improved accuracy by over 6 percentage points (p.p.), while RAG provided an additional 5 p.p. increase by dynamically retrieving relevant knowledge. In a specific experiment, the fine-tuned model improved answer similarity from 47% to 72%, leveraging knowledge across different geographic locations.

This study highlights that fine-tuning excels at embedding structured domain expertise into the model, while RAG enhances adaptability by incorporating real-time, external knowledge. Businesses can strategically combine both approaches to maximize accuracy, efficiency, and relevance based on their industry needs.

Here's another example:

A customer wants an LLM model to turn natural language questions into queries in a specific, nonstandard query language. The customer provides guidance in the prompt ("Always return GQL") and uses RAG to retrieve the database schema. However, the syntax isn't always correct and often fails for edge cases. The customer collects thousands of examples of natural language questions and the equivalent queries for the database, including cases where the model failed before. The customer then uses that data to fine-tune the model. Combining the newly fine-tuned model with the engineered prompt and retrieval brings the accuracy of the model outputs up to acceptable standards for use.



The fine-tuning journey

Before you start with fine-tuning

When starting out on your generative AI journey, we recommend you begin with prompt engineering and RAG to familiarize yourself with base models and its capabilities.

As you get comfortable and begin building your solution, it's important to understand where prompt engineering may fall short and that will help you realize if you should try fine-tuning.

- Is the base model failing on edge cases or exceptions?
- Is the base model inconsistent in producing outputs in the desired format?
- Is it difficult to fit enough examples in the context window to steer the model?
- Is the model's latency too high for your performance requirements?
- Is the base model becoming too expensive or inefficient due to long, complex prompts?

Examples of failure with the base model and prompt engineering can help you identify the data to collect for fine-tuning and establish a performance baseline that you can evaluate and compare your fine-tuned model against. **Having a baseline for performance without fine-tuning is essential for knowing whether or not fine-tuning improves model performance.**

Executing your fine-tuning project

At a high level, fine tuning requires you to:

(1) Prepare and upload training data,

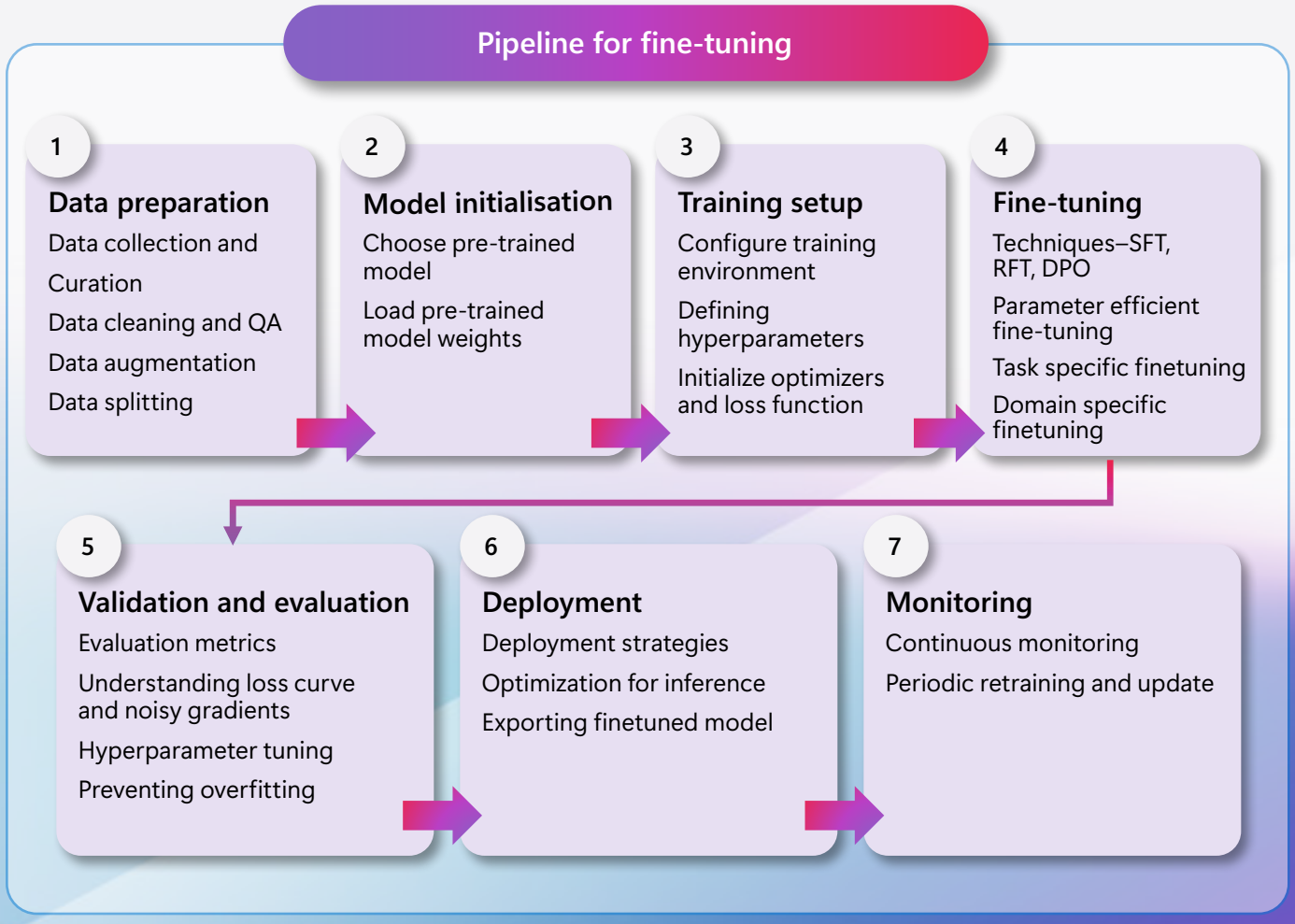
(2) Train a new fine-tuned model,

(3) Evaluate your newly trained model,

(4) Deploy that model for inferencing, and

(5) Use the fine-tuned model in your application

It's important to call out that fine-tuning is heavily dependent on the quality of data that you can provide. It's best practice to provide hundreds, if not thousands, of training examples to be successful and get your desired results.



Data preparation

What data are you going to use for fine-tuning?

The fine-tuning process begins by selecting a pretrained model and preparing a relevant dataset tailored to the target task. This dataset should reflect the kind of inputs the model will see in deployment.

For example, if the goal is to fine-tune a model for sentiment analysis, the dataset would include labeled text examples categorized by sentiment (positive, negative, neutral).

The model is then retrained on this dataset, adjusting its parameters to better align with the new task. This retraining process usually requires fewer computational resources compared to training a model from scratch, as it builds upon the existing capabilities.

Even with a great use case, fine-tuning is only as good as the quality of the data that you're able to provide. Different models will require different data volumes, but you often need to provide fairly large quantities of high-quality curated data. Another important point is even with high quality data if your data isn't in the necessary format for fine-tuning, you'll need to commit engineering resources in order to properly format the data.

You are ready for fine-tuning if...

You have identified relevant datasets for fine-tuning.

You have formatted the dataset appropriately for training.

You have curated the dataset to ensure high quality.

To fine-tune a model for chat or question answering, your training dataset should reflect the types of interactions the model will handle.
Here are some key elements to include in your dataset:

- **Prompts and responses: Each entry should contain a prompt (e.g., a user question) and a corresponding response (e.g., the model's reply).**
- **Contextual information:** For multi-turn conversations, include previous exchanges to help the model understand context and maintain coherence.
- **Diverse examples:** Cover a range of topics and scenarios to improve generalization and robustness.
- **Human-generated responses:** Use responses written by humans to teach the model how to generate natural and accurate replies.
- **Formatting:** Use a clear structure to separate prompts and responses. For example, \n\n###\n\n and ensure the delimiter doesn't appear in the content.

Best Practices for data preparation

The more training examples you have, the better. Fine tuning jobs will not proceed without at least 10 training examples, but such a small number isn't enough to noticeably influence model responses. It is best practice to provide hundreds, if not thousands, of training examples to be successful. 100 good-quality examples are better than 1000 poor examples.

In general, doubling the dataset size can lead to a linear increase in model quality. But keep in mind, low quality examples can negatively impact performance. If you train the model on a large amount of internal data, without first pruning the dataset for only the highest quality examples you could end up with a model that performs much worse than expected.

Best practices for data labelling

Accurate and consistent labelling is crucial for training the model. Follow these best practices:

- **Ensure data diversity:** Include all the typical variations such as document formats (digital vs. scanned), layout differences, varying table sizes, and optional fields**.**
- **Define fields clearly:** Use semantically meaningful field names (e.g., effective_date), especially for custom models, and follow consistent naming conventions like Pascal or camel case.
- **Maintain label consistency:** Ensure uniform labeling across documents, particularly for repeated values.
- **Split your data:** Separate training and validation sets to evaluate the model on unseen data and avoid overfitting.
- **Label at scale:** Aim for at least 50 labeled documents per class, where applicable.
- **Combine automation and review:** Use AI-generated labels to accelerate the process, focusing manual effort on complex or critical fields.

Model selection

Selecting the right model for fine-tuning is a critical decision that impacts performance, efficiency, and cost. Before making a choice, it is essential to clearly define the task and establish the desired performance metrics. A well-defined task ensures that the selected model aligns with specific requirements, optimizing effort and resources.

Best practices for model selection

Choose models based on domain specificity and use case.
Start by evaluating industry-standard models for general capabilities, then assess models fine-tuned for your specific use case. If your task requires deep domain expertise, selecting a model tailored to your industry can improve accuracy and efficiency while reducing the need for extensive fine-tuning.

Assess model performance on leaderboards
Review benchmark leaderboards to evaluate how pre-trained models perform on relevant tasks. Focus on key metrics such as accuracy, coherence, latency, and domain-specific benchmarks to identify a strong foundation for fine-tuning.

Experiment with model playgrounds
Utilize interactive testing environments to assess the base model's performance on real-world use cases. By adjusting prompts, temperature, and other parameters, you can identify performance gaps before investing in fine-tuning.

Weigh trade-offs between model size, complexity, cost, and performance
Larger models may offer superior accuracy but come with higher computational costs and latency. Consider the balance between efficiency and precision based on your deployment needs.

Training and evaluation
Fine-tuning isn't merely a matter of retraining on a new dataset; it also involves careful consideration of various hyperparameters and techniques to balance accuracy and generalization. A key risk is overfitting, where a model becomes too narrowly adapted to **t**raining data, reducing its effectiveness on unseen inputs. To mitigate overfitting and optimize performance, fine-tuning requires adjusting parameters such as learning rate, regularization, batch size, number of epochs, and seed settings.

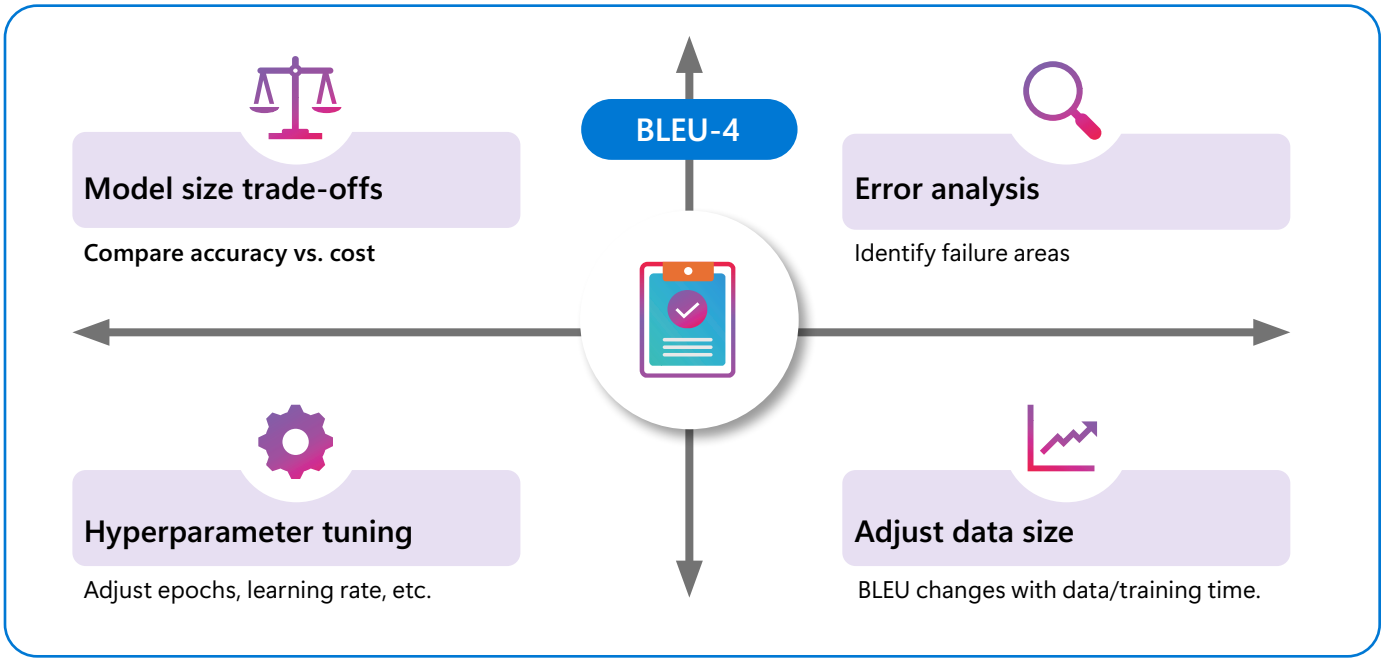
Best practices for training

- Experiment with different hyperparameters to improve model performance.
- Consider freezing early layers responsible for general language understanding and fine-tune only the later, task-specific layers. This preserves foundational knowledge while enabling domain-specific adaptation.
- Align training data with real-world usage patterns. For example, if 90% of queries are about Question A, mirror that ratio in your dataset to optimize for high-frequency scenarios while still covering edge cases.
- Establish clear training goals and evaluation metrics to monitor fine-tuning effectiveness.

Use evaluations in fine-tuning

You should have clearly defined goals for what success with fine-tuning looks like. Ideally, these should go beyond qualitative measures and include quantitative metrics, such as using a holdout validation set, conducting user acceptance testing, or A/B tests comparing the fine-tuned model to the base model.

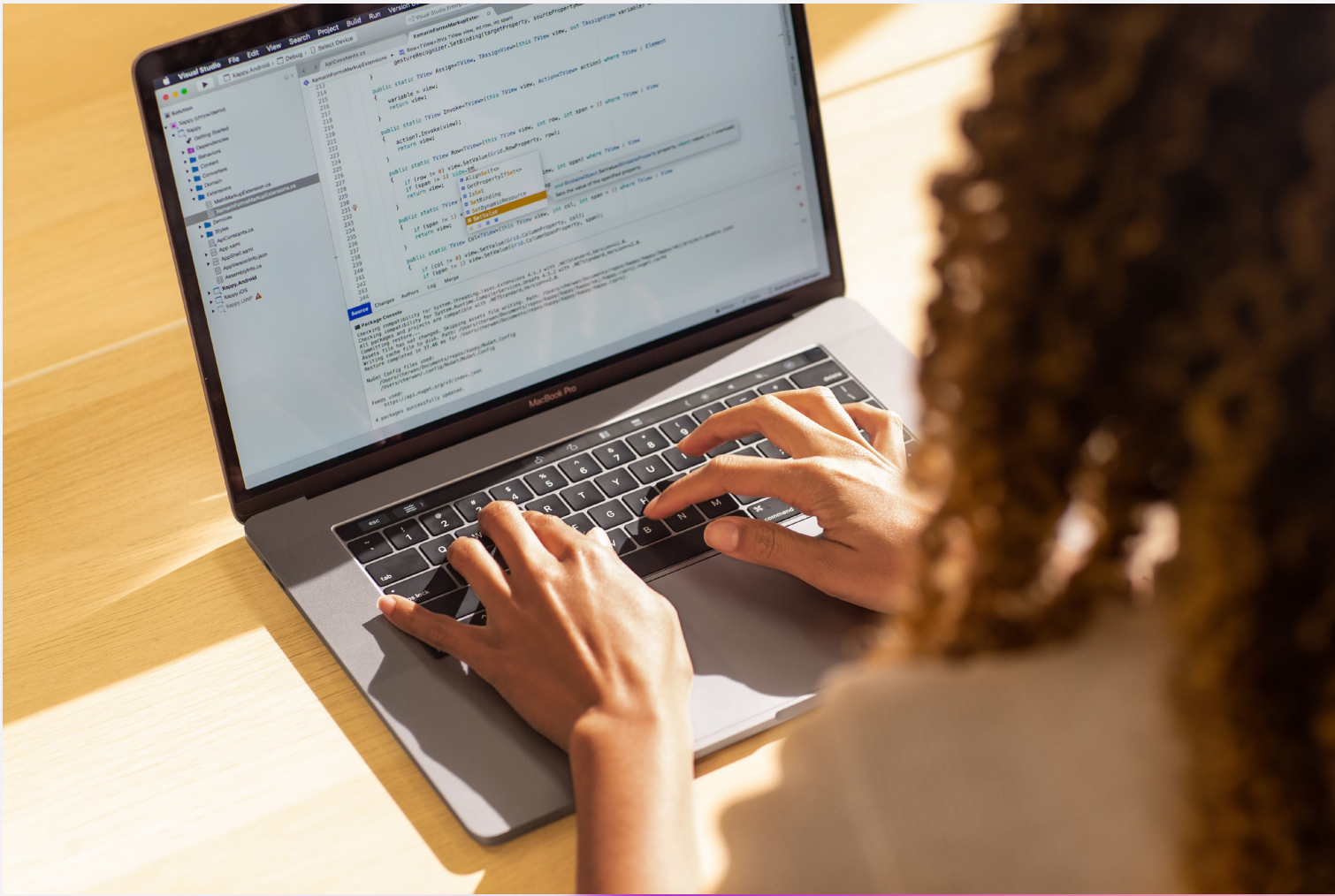
Model training can be guided by metrics. For example, BLEU-4 was used to evaluate training when fine-tuning a model to generate chest X-Ray reports, as seen in this paper.



Use intermediate checkpoints for better model selection. Save checkpoints at regular intervals (e.g., every few epochs) and evaluate their performance. In some cases, an intermediate checkpoint may outperform the final model, allowing you to select the best version rather than relying solely on the last trained iteration.

Deployment and monitoring

- Choose a suitable deployment infrastructure, such as cloud-based platforms or on-premises servers.
- Continuously monitor the model's performance and make necessary adjustments to ensure optimal performance.
- Consider regional deployment needs and latency requirements to meet enterprise SLAs. Implement security guardrails, such as private links, encryption, and access controls, to protect sensitive data and maintain compliance with organizational policies.



Use cases

Turning natural language into a query language is just one use-case where you can 'show not tell' the model how to behave. Custom model can be built for a wide range of business use cases, including:

Business use cases

Customer service

Generate personalized responses to customer inquiries by leveraging relevant product information, customer history, and external knowledge bases.

By quickly identifying root causes and providing accurate solutions, hybrid models can enhance customer satisfaction and reduce resolution time.

Content generation

Generate high-quality content, such as product descriptions, marketing copy, and social media posts, by combining factual information with creative writing skills.

By ensuring alignment with brand guidelines, hybrid models help maintain consistent brand voice and messaging.

Document summarization

Generate accurate and concise summaries of long documents by leveraging external knowledge bases and fine-tuned language models.

Hybrid models can extract key information, making it easier for users to quickly find what they need.

Knowledge management

Help users find relevant information from large knowledge bases by understanding their queries and retrieving the most relevant documents.

Hybrid models can also be used to create knowledge graphs that visualize and analyze complex information.

The following industry use cases illustrate how fine-tuned AI models can address domain-specific challenges, enhance operational efficiency, and deliver tailored value across sectors.

Healthcare and life sciences

Diagnostic Support
Customize AI models using proprietary medical research, clinical guidelines, and patient data.

Clinical Documentation
Fine-tune AI for medical transcription, documentation automation, and physician assistance.

Finance

Personalized Advisory
Automate financial insights by tailoring recommendations based on historical and real-time data.

Strategic Alignment
Ensure AI-driven investment analysis adheres to internal strategies and risk policies

Legal services

Contract Intelligence
Fine-tune AI models to generate contract summaries, legal briefings, and compliance documents with strict adherence to legal language.

Content Consistency
Maintain brand voice across marketing, corporate communications, and technical documentation

Agriculture

Crop Health Monitoring
Fine-tune models to analyze leaf images and sensor data to detect diseases and recommend targeted treatments.

Farmer Advisory Support
Fine-tune AI assistants to provide tailored guidance on best practices, government policies, and market trends, with support for multiple languages and dialects

Fine-tuning techniques

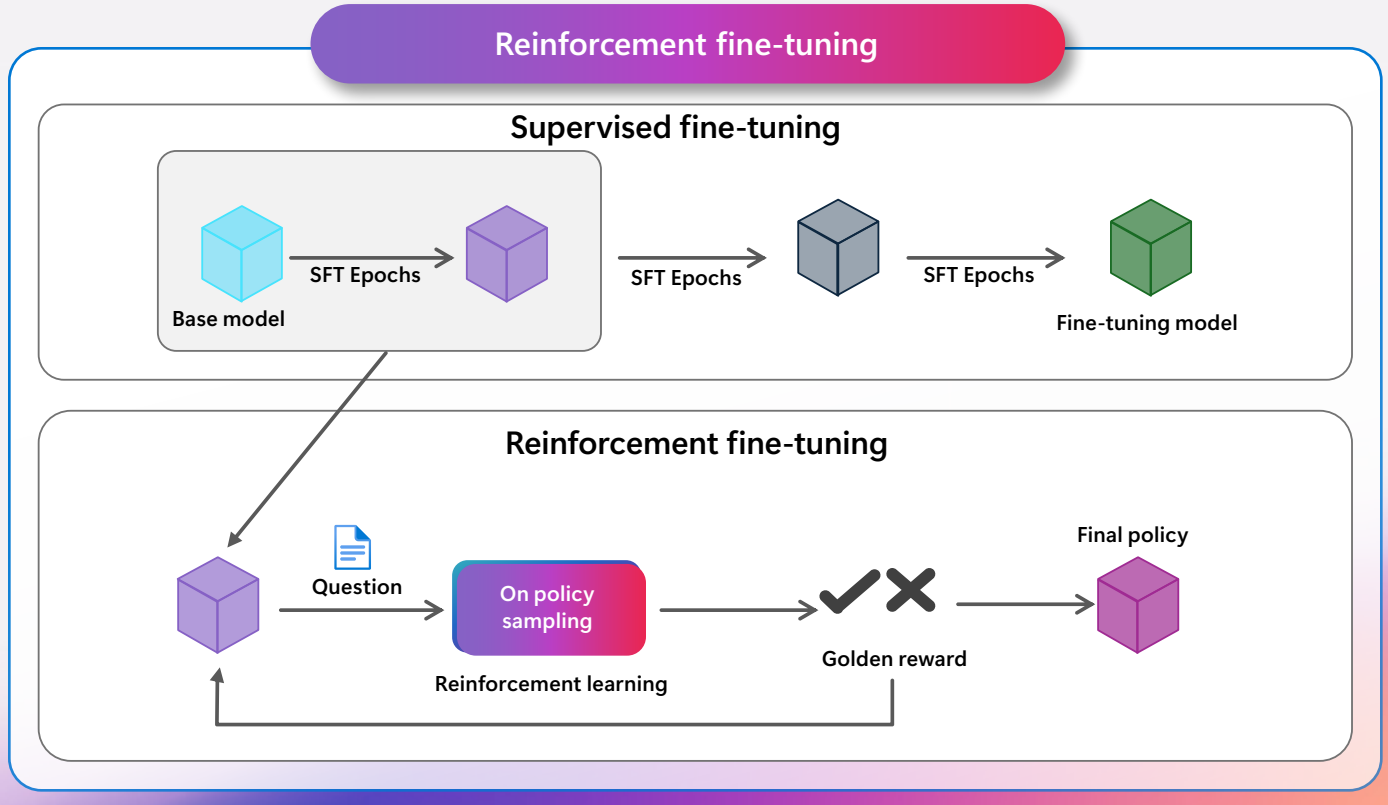
Supervised fine-tuning

Supervised fine tuning allows you to provide custom data (prompt/completion or conversational chat, depending on the model) to teach the base model new skills. This process involves further training the model on a high-quality labelled dataset, where each data point is associated with the correct output or answer. The goal is to enhance the model's performance on a particular task by adjusting its parameters based on the labelled data.

Reinforcement fine-tuning

Reinforcement fine-tuning is a model customization technique, particularly beneficial for optimizing model behavior in highly complex or dynamic environments, enabling the model to learn and adapt through iterative feedback and decision-making.

For example, financial services providers can optimize the model for faster, more accurate risk assessments or personalized investment advice. In healthcare and pharmaceuticals, o3-mini can be tailored to accelerate drug discovery, enabling more efficient data analysis, hypothesis generation, and identification of promising compounds.

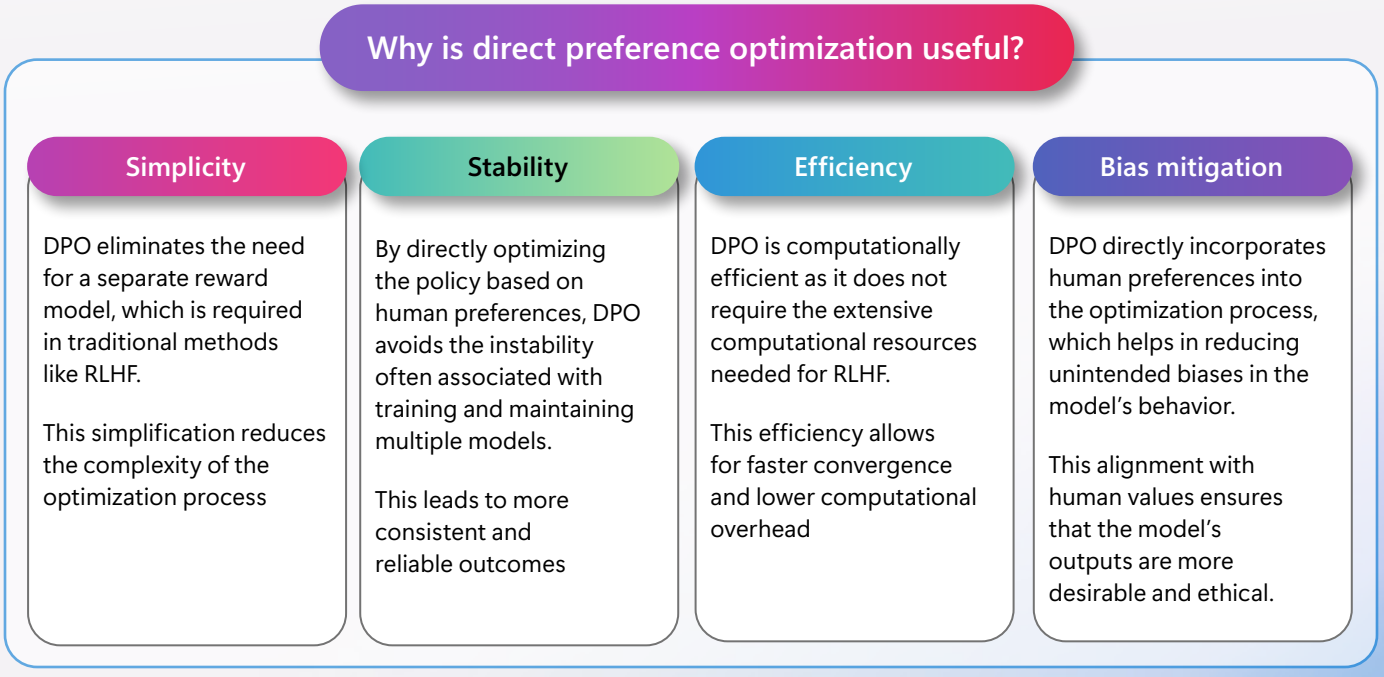


Direct preference optimization

Direct Preference Optimization (DPO) is another new alignment technique for large language models, designed to adjust model weights based on human preferences. Unlike Reinforcement Learning from Human Feedback (RLHF), DPO does not require fitting a reward model and uses binary preferences for training. This method is computationally lighter and faster, making it equally effective at alignment while being more efficient. DPO is especially useful in scenarios where subjective elements like tone, style, or specific content preferences are important.

Why is DPO useful?

DPO is especially useful in scenarios where there's no clear-cut correct answer, and subjective elements like tone, style, or specific content preferences are important. This approach also enables the model to learn from both positive examples (what's considered correct or ideal) and negative examples (what's less desired or incorrect), often leveraging "thumbs up" and "thumbs down" feedback data to refine responses based on user preferences.



DPO is believed to be a technique that will make it easier for customers to generate high-quality training data sets. While many customers struggle to generate sufficient large data sets for supervised fine-tuning, they often have preference data already collected based on user logs, A/B tests, or smaller manual annotation efforts.

Overall, DPO offers a streamlined, stable, and efficient alternative to traditional methods, making it a promising approach for fine-tuning language models to better align with human expectations and values.

Model optimization using distillation

Model distillation empowers developers to use the outputs of large, complex models to fine-tune smaller, more efficient ones. This technique allows the smaller models to perform just as well on specific tasks, all while significantly cutting down on both cost and latency.

Distillation refers to the process of using a large, general purpose teacher model to train a smaller student model to perform well at a specific task.

Distillation is of particular interest for several reasons:

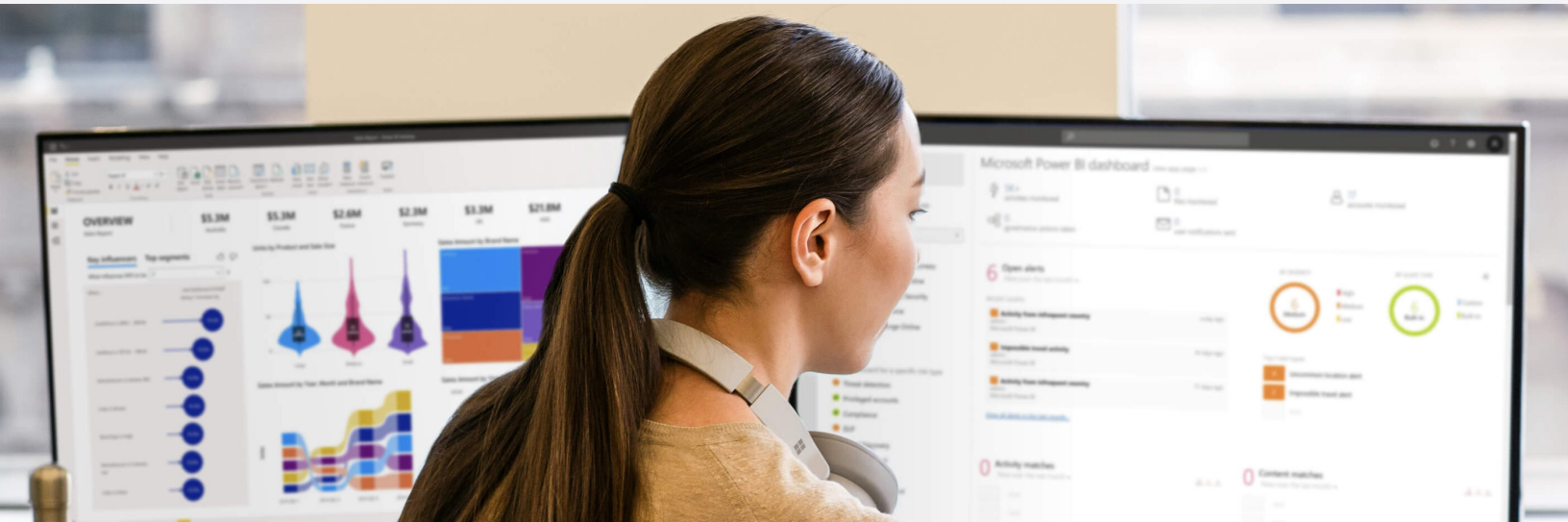
- reduce the costs and latency
- improve performance.
- operate in resource-constrained environments

Let's work together on a distillation user scenario: distilling from production for a news sentiment use case.

Imagine a company using an AI-powered platform to monitor news in real-time, tracking sentiment around its brand, products, and industry trends. Behind the scenes, the large model is analyzing news articles to detect positive, negative, or neutral sentiment.

As the number of news sources and updates grows, so do the platform's operating costs and with each new data source, processing times begin to slow. Now, imagine if the platform could maintain this large model's intelligence and accuracy while reducing both costs and response times.

Model distillation to the rescue! We capture model's real-time interactions with news articles, building a rich dataset of responses. With this data, we can distil large model's power into a smaller, faster model that delivers high-quality answers at a fraction of the cost.



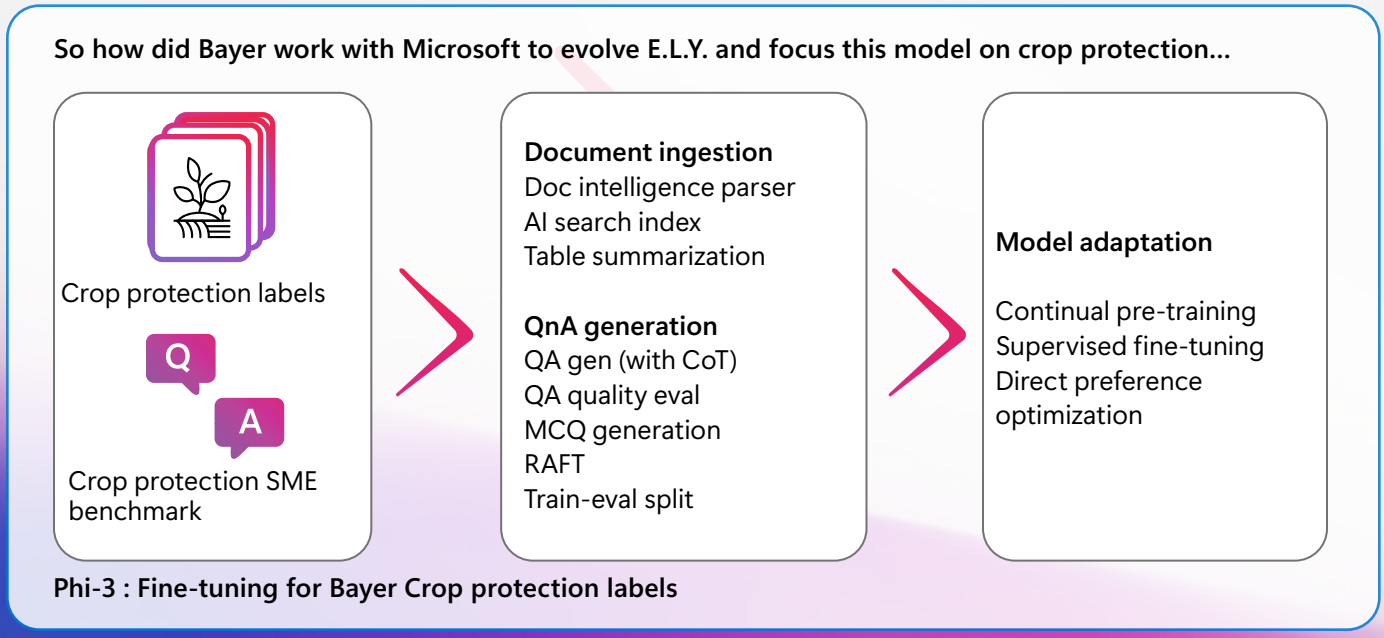
Challenges and limitations of fine-tuning

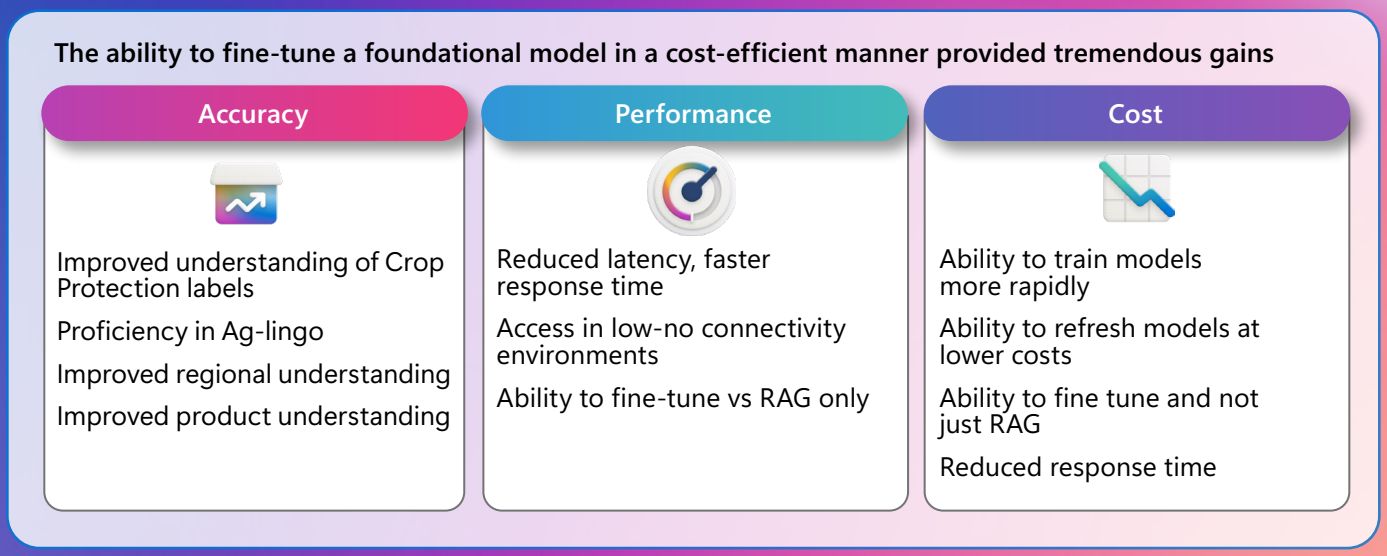
Fine-tuning large language models scan be a powerful technique to adapt them to specific domains and tasks. However, fine-tuning also comes with some challenges and disadvantages that need to be considered before applying it to a real-world problem. Below are a few of these challenges and disadvantages.

- Fine-tuning requires high-quality, sufficiently large, and representative training data matching the target domain and task. Quality data is relevant, accurate, consistent, and diverse enough to cover the possible scenarios and variations the model will encounter in the real world. Poor-quality or unrepresentative data leads to over-fitting, under-fitting, or bias in the fine-tuned model, which harms its generalization and robustness.
- Fine-tuning large language models means extra costs associated with training and hosting the custom model.
- Formatting input/output pairs used to fine-tune a large language model can be crucial to its performance and usability.
- Fine-tuning may need to be repeated whenever the data is updated, or when an updated base model is released. This involves monitoring and updating regularly.
- Fine-tuning is a repetitive task (trial and error) so, the hyperparameters need to be carefully set. Fine-tuning requires much experimentation and testing to find the best combination of hyperparameters and settings to achieve desired performance and quality.

Enhancing expertise with fine-tuning at Bayer

Bayer leveraged fine-tuning to enhance the accuracy and efficiency of its agronomy-focused AI assistant, E.L.Y. Initially, Bayer implemented RAG on GPT-3.5 to improve responses related to Bayer's commercial agricultural products, achieving a 42% improvement in accuracy. However, as usage scaled, challenges emerged—slower response times, higher costs, and limited scalability. To address this, Bayer collaborated with Microsoft to fine-tune a specialized model, focusing on crop protection label understanding. The fine-tuning process involved continual pre-training, supervised fine-tuning, and direct preference optimization, supported by real-world prompt templates, training sets, and proprietary QA benchmarks. By integrating domain expertise and optimizing model adaptation, Bayer significantly improved response precision, reduced latency, and ensured the model could handle complex agronomic queries, including unit conversions and application rates, ultimately delivering a faster, and effective solution.





Best practices for fine-tuning

Here are some best practices that can help improve the efficiency and effectiveness of fine-tuning LLMs for various applications:

Try different data formats: Depending on the task, different data formats can have different impacts on the model's performance. For example, for a classification task, you can use a format that separates the prompt and the completion with a special token, such as {"prompt": "Paris##\n", "completion": " city\n###\n"}. Be sure to use formats suitable for your application.

Collect a large, high-quality dataset: LLMs are data-hungry and can benefit from having more diverse and representative data to fine-tune on. However, collecting and annotating large datasets can be costly and time-consuming. Therefore, you can also use synthetic data generation techniques to increase the size and variety of your dataset. However, you should also ensure that the synthetic data is relevant and consistent with your task and domain. Also ensure that it does not introduce noise or bias to the model.

Try fine-tuning subsets first: To assess the value of getting more data, you can fine-tune models on subsets of your current dataset to see how performance scales with dataset size. This fine-tuning can help you estimate the learning curve of your model and decide whether adding more data is worth the effort and cost. You can also compare the performance of your model with the pre-trained model or a baseline. This comparison shows how much improvement you can achieve with fine-tuning.

Experiment with hyperparameters: Iteratively adjust hyperparameters to optimize the model performance. Hyperparameters, such as the learning rate, the batch size and the number of epochs, can have significant effect on the model's performance. Therefore, you should experiment with different values and combinations of hyperparameters to find the best ones for your task and dataset.

Start with a smaller model: A common mistake is assuming that your application needs the newest, biggest, most expensive model. Especially for simpler tasks, start with smaller models and only try larger models if needed.

Select models based on domain needs: Start with industry-standard models before considering fine-tuned versions for specific use cases. Use benchmark leaderboards to assess performance and test real-world scenarios in model playgrounds. Balance accuracy, cost, and efficiency to ensure the best fit for your deployment

Fine-tuning with Microsoft Foundry

As enterprises increasingly adopt generative AI for domain-specific tasks, the ability to tailor foundational models to unique datasets becomes critical. Foundry enables organizations to fine-tune large language and multimodal models with precision, scalability, and enterprise-grade security, bridging the gap between general-purpose AI and specialized business needs.

Foundry supports the fine-tuning of variety of pre-trained models, including Azure OpenAI models, lightweight Phi family and third-party models, offering flexibility across use cases. It provides a streamlined fine-tuning workflow that can be executed with minimal configuration.

Types of fine-tuning

Foundry supports a range of fine-tuning methods: supervised fine tuning (SFT), direct preference optimization (DPO) and Reinforcement Fine-tuning (RFT).

Supervised Fine-tuning trains a model using prompt-response pairs, demonstrating the desired way it should respond to specific inputs. This method helps customize the model to align with your unique use case. Foundry offers SFT for third-party models as well, enabling greater flexibility in customizing models to specific business needs.

Preference Fine-uning goes a step further by optimizing the model based on human preferences. By training with ranked response examples, DPO helps refine outputs to better match user expectations.

Reinforcement Fine-tuning enables customization of models using dozens to thousands of high-quality tasks. By grading the model's responses with provided reference answers, this technique reinforces how the model reasons through similar problems and improves its accuracy on specific tasks in that domain.

These techniques can be combined for greater customization. Start with SFT to build a model tailored to your tasks, ensuring high-quality and representative data. Then, apply DPO to fine-tune responses through comparative adjustments, aligning them even more closely with user preferences.

Model Distillation in Foundry

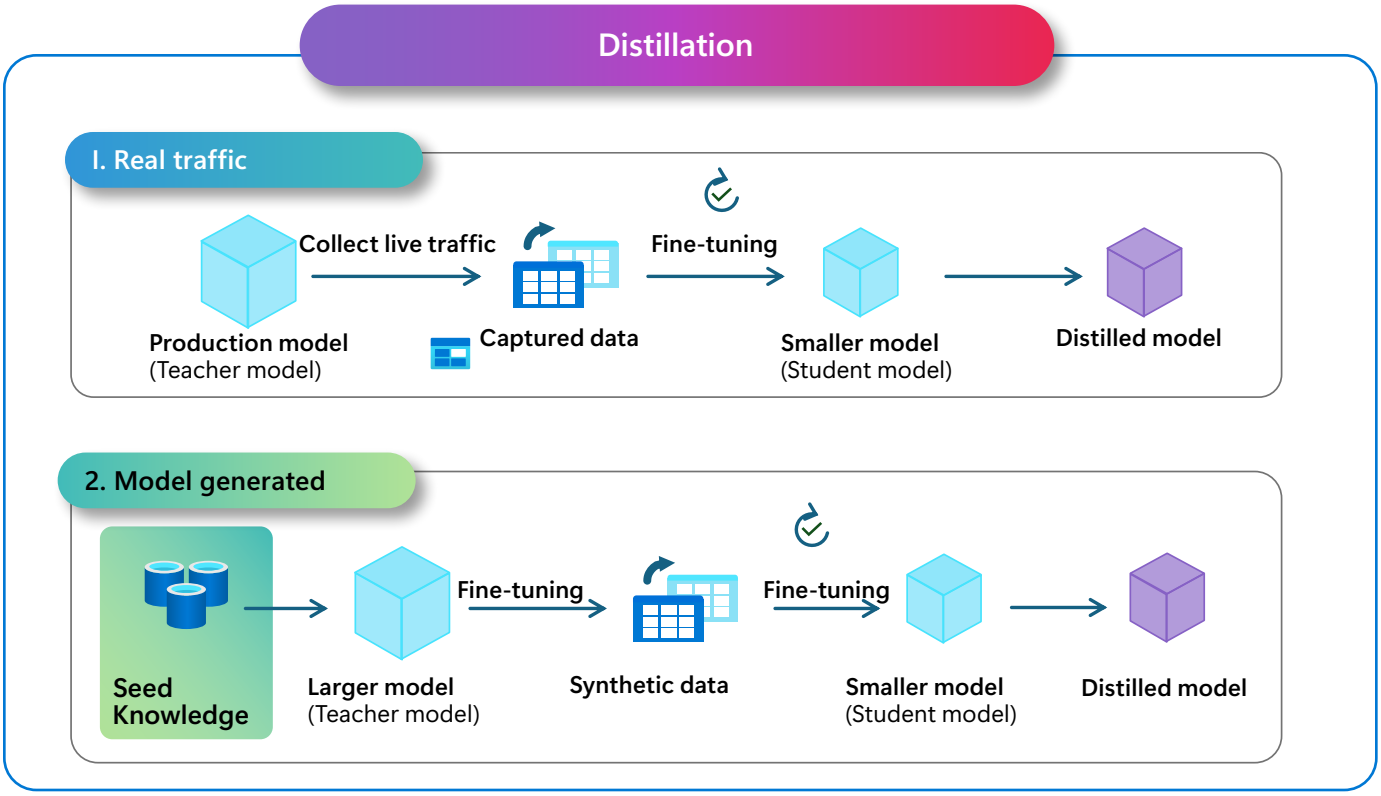
Foundry provides support for distillation, allowing you to efficiently train student models.

The key steps in knowledge distillation include:

1. Use the pretrained teacher model to produce outputs (soft labels or logits) for the original dataset.
2. Leverage the teacher's predictions, alongside the original dataset, to guide the student model in learning to replicate the teacher's behavior more efficiently.

Model distillation is supported for both Azure OpenAI Service models and select open source third party models available in the model catalog.

Azure OpenAI Service enables efficient model distillation by leveraging Stored Completions, Evaluation, and Fine-Tuning to optimize models for specific use cases.



With Stored Completions, users can capture input-output pairs from models like GPT-4o, creating high-quality datasets based on real production data. These datasets can then be used for both evaluation and fine-tuning.

Azure OpenAI Evaluation provides a streamlined framework to assess model performance, utilizing Stored Completions or existing datasets to measure effectiveness across specific tasks.

Finally, these insights seamlessly integrate into the fine-tuning process, ensuring that models are continuously refined and optimized based on real-world usage, improving accuracy, efficiency, and task-specific performance.

Continual fine-tuning

Update a fine-tuned model with new data or continue from where you left off. Keep most parameters the same, but consider reducing the learning rate.

Vision fine-tuning

Fine-tuning GPT-4o with vision capabilities is streamlined, allowing training with diverse image data, including graphs and charts. The input format remains consistent as JSONL, ensuring seamless integration. Additionally, there is no charge for rejected jobs, with helpful guidance provided on how to resolve issues for successful training.

Azure ensures responsible AI practices by automatically filtering out faces, captchas, and abusive content from training datasets, preventing these data rows from being used in model training.

Built in safety

Azure provides strong defences through Prompt Shields, jailbreak risk detection, and comprehensive safety evaluations. Automatic safety assessments include sampling training data for harmful content and running simulated adversarial conversations with fine-tuned models. Ensuring privacy and security, evaluations are conducted in dedicated, customer-specific private workspaces, with endpoints located in the same geography as the Azure resource.

Fine-tuning remains a critical capability for adapting foundation models to domain-specific tasks, enhancing accuracy, and reducing latency in production scenarios. By applying best practices, such as selecting the appropriate fine-tuning method, leveraging high-quality domain data, and using evaluation loops to validate performance, organizations can accelerate time to value while maintaining model integrity.

Foundry simplifies this process through an integrated platform that offers support for multiple fine-tuning techniques, seamless access to Microsoft and open-source models, and built-in tools for model monitoring, deployment, and governance. Together, these features enable enterprise teams to efficiently operationalize AI while aligning with strategic business goals.

Begin your journey with Foundry

Microsoft Azure