Model selection

Copy page

Choose the best model for performance and cost.

Choosing the right model, whether GPT-4o or a smaller option like GPT-4o-mini, requires balancing **accuracy**, **latency**, and **cost**. This guide explains key principles to help you make informed decisions, along with a practical example.

Core principles

The principles for model selection are simple:

Optimize for accuracy first: Optimize for accuracy until you hit your accuracy target.

Optimize for cost and latency second: Then aim to maintain accuracy with the cheapest, fastest model possible.

1. Focus on accuracy first

Begin by setting a clear accuracy goal for your use case, where you're clear on the accuracy that would be "good enough" for this use case to go to production. You can accomplish this through:

Setting a clear accuracy target: Identify what your target accuracy statistic is going to be.

For example, 90% of customer service calls need to be triaged correctly at the first interaction.

Developing an evaluation dataset: Create a dataset that allows you to measure the model's performance against these goals.

To extend the example above, capture 100 interaction examples where we have what the user asked for, what the LLM triaged them to, what the correct triage should be, and whether this was correct or not.

Using the most powerful model to optimize: Start with the most capable model available to achieve your accuracy targets. Log all responses so we can use them for distillation of a smaller model.

Use retrieval-augmented generation to optimize for accuracy

Use fine-tuning to optimize for consistency and behavior

During this process, collect prompt and completion pairs for use in evaluations, few-shot learning, or fine-tuning. This practice, known as **prompt baking**, helps you produce high-quality examples for future use.

For more methods and tools here, see our Accuracy Optimization Guide.

Setting a realistic accuracy target

Calculate a realistic accuracy target by evaluating the financial impact of model decisions. For example, in a fake news classification scenario:

> **Correctly classified news:** If the model classifies it correctly, it saves you the cost of a human reviewing it - let's assume **$50**.
>
> **Incorrectly classified news:** If it falsely classifies a safe article or misses a fake news article, it may trigger a review process and possible complaint, which might cost us **$300**.

Our news classification example would need **85.8%** accuracy to cover costs, so targeting 90% or more ensures an overall return on investment. Use these calculations to set an effective accuracy target based on your specific cost structures.

2. Optimize cost and latency

Cost and latency are considered secondary because if the model can't hit your accuracy target then these concerns are moot. However, once you've got a model that works for your use case, you can take one of two approaches:

> **Compare with a smaller model zero- or few-shot:** Swap out the model for a smaller, cheaper one and test whether it maintains accuracy at the lower cost and latency point.
>
> **Model distillation:** Fine-tune a smaller model using the data gathered during accuracy optimization.

Cost and latency are typically interconnected; reducing tokens and requests generally leads to faster processing.

The main strategies to consider here are:

> **Reduce requests:** Limit the number of necessary requests to complete tasks.
>
> **Minimize tokens:** Lower the number of input tokens and optimize for shorter model outputs.
>
> **Select a smaller model:** Use models that balance reduced costs and latency with maintained accuracy.

To dive deeper into these, please refer to our guide on latency optimization.

Exceptions to the rule

Clear exceptions exist for these principles. If your use case is extremely cost or latency sensitive, establish thresholds for these metrics before beginning your testing, then remove the models that exceed those from consideration. Once benchmarks are set, these guidelines will help you refine model accuracy within your constraints.

Practical example

To demonstrate these principles, we'll develop a fake news classifier with the following target metrics:

Accuracy: Achieve 90% correct classification

Cost: Spend less than $5 per 1,000 articles

Latency: Maintain processing time under 2 seconds per article

Experiments

We ran three experiments to reach our goal:

1. **Zero-shot:** Used `GPT-4o` with a basic prompt for 1,000 records, but missed the accuracy target.
2. **Few-shot learning:** Included 5 few-shot examples, meeting the accuracy target but exceeding cost due to more prompt tokens.
3. **Fine-tuned model:** Fine-tuned `GPT-4o-mini` with 1,000 labeled examples, meeting all targets with similar latency and accuracy but significantly lower costs.

ID	METHOD	ACCURACY	ACCURACY TARGET	COST	COST TARGET	AVG. LATENCY	LATENCY TARGET
1	gpt-4o zero-shot	84.5%		$1.72		< 1s	
2	gpt-4o few-shot (n=5)	91.5%	✓	$11.92		< 1s	✓
3	gpt-4o-mini fine-tuned w/ 1000 examples	91.5%	✓	$0.21	✓	< 1s	✓

Conclusion

By switching from `gpt-4o` to `gpt-4o-mini` with fine-tuning, we achieved **equivalent performance for less than 2%** of the cost, using only 1,000 labeled examples.

This process is important - you often can't jump right to fine-tuning because you don't know whether fine-tuning is the right tool for the optimization you need, or you don't have enough labeled examples. Use `gpt-4o` to achieve your accuracy targets, and curate a good training set - then go for a smaller, more efficient model with fine-tuning.