1 Introduction: Why Knowledge Distillation?

1.1 The computational problem

Modern large language models require enormous computational resources to train.
Training a frontier model can involve:

- Thousands of GPUs or specialized accelerators.
- Massive datasets containing trillions of tokens.
- Weeks or months of distributed computation.
- Expensive reinforcement learning and post-training.
- Significant engineering and infrastructure costs.

Even after training, deploying these models is expensive.
A large model may require:

- Multiple GPUs to serve a single model instance.
- Large amounts of memory for model weights and KV caches.
- Significant energy consumption.
- High inference latency.

This creates an interesting question:

> Can we use an expensive, highly capable model to teach a much smaller model how to perform many of the same tasks?

This is the fundamental motivation behind knowledge distillation.

1.2 The teacher-student analogy

Imagine an experienced professor teaching a graduate student.
The professor has:

- Years of accumulated knowledge.
- Experience solving difficult problems.
- An understanding of which approaches work.
- An ability to recognize subtle mistakes.

The student does not need to independently rediscover everything the professor knows.

Instead, the student observes the professor solving problems, studies the solutions, asks questions, and practices similar problems.

Eventually, the student can solve many problems independently.

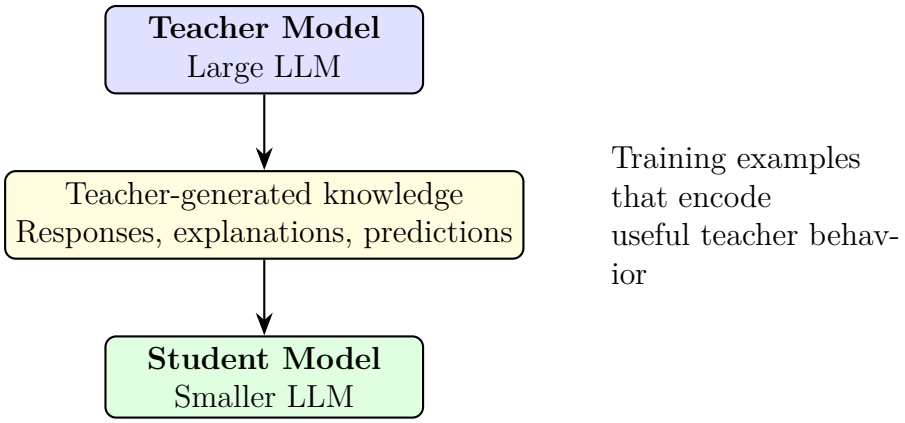
Knowledge distillation applies a similar idea to neural networks.

- **Teacher:** A large, capable model.
- **Student:** A smaller model.
- **Teaching material:** Predictions, explanations, generated examples, or intermediate representations.
- **Learning:** Training the student to reproduce useful aspects of the teacher's behavior.

The important distinction is that we are not necessarily transferring the teacher's parameters.

We are transferring information about how the teacher behaves.

1.3 The basic architecture



The teacher is generally already trained and frozen.

We use it to generate information that becomes training material for the student.

The student is then trained independently.

Important: The teacher and student do not have to share the same architecture, number of layers, or number of parameters.

However, some distillation techniques require compatible tokenizers or internal representations.

2 Foundations: Why Does Distillation Work?

2.1 Traditional knowledge distillation

Knowledge distillation predates modern LLMs.

A particularly influential formulation was introduced by Hinton, Vinyals, and Dean in 2015.
Consider a classification problem.
Suppose we have an image of a dog, and a model must classify it as one of several categories:

- Dog
- Cat
- Wolf
- Fox

A conventional training dataset might provide the following label:

$$y = [1, 0, 0, 0]$$

This tells the model that the correct answer is Dog.
But it does not tell the model anything about the relative similarity of the other categories.
For example, a wolf is more similar to a dog than a cat is.
A teacher model might produce:

$$p_T = \begin{bmatrix} 0.70 \\ 0.05 \\ 0.20 \\ 0.05 \end{bmatrix}$$

This distribution contains additional information.
The teacher is effectively saying:

- Dog is the most likely answer.
- Wolf is also reasonably plausible.
- Cat and Fox are less plausible.

The student learns not only the correct answer, but also something about the teacher's understanding of the problem.
This additional information is sometimes called **dark knowledge**.

2.2 Analogy: Multiple-choice examination

Imagine asking students to identify an animal.
A student who answers:

> Dog.

has provided a correct answer.
Another student answers:

70% dog, 20% wolf, 5% cat, 5% fox.

The second answer reveals more about how the student thinks about the problem.
It communicates uncertainty and relationships between possible answers.
Distillation attempts to transfer some of this additional information.

2.3 Softmax temperature

A neural network produces logits, which are unnormalized scores.
The probabilities are obtained using softmax:

$$p_i = \frac{\exp(z_i)}{\sum_j \exp(z_j)}$$

Knowledge distillation often uses a temperature parameter:

$$p_i^{(T)} = \frac{\exp(z_i/T)}{\sum_j \exp(z_j/T)}$$

where:

- z_i is the logit for class i.
- T is the temperature.

When $T = 1$, we have the usual softmax.
When $T > 1$, the probability distribution becomes softer.
For example:

$$\text{Original: } [0.90, 0.08, 0.02]$$

A higher temperature might produce:

$$\text{Softened: } [0.60, 0.27, 0.13]$$

The less likely alternatives become more visible.
This allows the student to learn relationships that would otherwise be hidden by a highly confident prediction.

2.4 The distillation loss

Let:

- p_T be the teacher's probability distribution.
- p_S be the student's probability distribution.
- y be the ground-truth label.

The student can be trained using a combination of two losses:

$$\mathcal{L} = \alpha \mathcal{L}_{CE}(y, p_S) + (1 - \alpha) T^2 D_{KL}(p_T^{(T)} \| p_S^{(T)})$$

Here:

- Cross-entropy measures the student's error against the actual label.
- KL divergence measures the difference between the teacher's and student's probability distributions.
- T controls the softness of the distributions.
- α controls the relative importance of the original labels and teacher predictions.

The T^2 factor is conventionally included to compensate for the temperature-dependent scaling of gradients.

The student is therefore learning from two sources:

1. What the training data says is correct.
2. What the teacher believes is likely.

2.5 Why does this improve learning?

There are several complementary explanations.

1. More informative supervision

Hard labels provide one answer.

Teacher distributions provide relative preferences among many possible answers.

This can provide a richer training signal.

2. Better generalization

The teacher's outputs encode similarities and relationships between examples.

The student can learn smoother decision boundaries rather than memorizing isolated answers.

3. Regularization

The teacher constrains the student's behavior.

Instead of learning arbitrary solutions that fit the training data, the student is encouraged to approximate a more capable model.

4. Transfer of learned representations

The teacher has already learned useful abstractions from large-scale training.

The student can learn behaviors associated with those abstractions through examples, without independently discovering all of them from scratch.

Important qualification:

Distillation does not magically transfer knowledge that the student cannot represent.

The student has limited capacity.

It can approximate selected aspects of the teacher's behavior, but it cannot necessarily reproduce everything.

3 Knowledge Distillation Specifically for LLMs

The traditional classification example is useful, but LLMs present a more interesting problem.

An LLM does not simply choose one of a few classes.

It generates sequences of tokens.

For a sequence:

$$x_1, x_2, \ldots, x_n$$

the model estimates:

$$P(x_{t+1} \mid x_1, \ldots, x_t)$$

At every generation step, it predicts a probability distribution over a potentially enormous vocabulary.

Knowledge distillation for LLMs can therefore operate at several different levels.

3.1 Method 1: Response-based distillation

This is one of the simplest and most widely accessible approaches.

The teacher generates answers to a collection of prompts.

Those prompt-response pairs become training examples for the student.

For example:

```
Prompt:
Explain the difference between a database index
and a database primary key.

Teacher:
A primary key uniquely identifies a row.
An index is a data structure that accelerates
retrieval. A primary key often has an associated
unique index, but the concepts are different.

Student:
[Trained to reproduce the teacher's response]
```

The student is trained using ordinary supervised fine-tuning (SFT).

The objective is generally token-level cross-entropy:

$$\mathcal{L}_{SFT} = -\sum_{t=1}^{n} \log P_S(y_t \mid x, y_{<t})$$

where:

- x is the prompt.

- y_t is the teacher-generated token.
- $y_{<t}$ represents previously generated tokens.

In effect, the teacher becomes a source of synthetic training data.

Analogy:

Instead of giving a student a textbook containing only questions and answers, we give the student a textbook written by an expert who has already solved thousands of problems.

The student learns by studying the solutions.

3.2 Method 2: Logit-based distillation

This approach is closer to traditional knowledge distillation.

Instead of receiving only the teacher's final answer, we obtain the teacher's probability distribution over the vocabulary at each generation step.

For example, suppose the teacher is generating:

```
The capital of France is ...
```

The teacher might assign the following probabilities to the next token:

Token	Probability
Paris	0.92
Lyon	0.03
London	0.02
Berlin	0.01
Other	0.02

The student receives more information than simply being told that the next token is Paris.

It learns that Paris is overwhelmingly preferred, while Lyon is a more plausible alternative than London.

The loss can be calculated at each token position:

$$\mathcal{L}_{logit} = T^2 \sum_t D_{KL}\left(P_T^{(T)}(\cdot \mid x, y_{<t}) \| P_S^{(T)}(\cdot \mid x, y_{<t})\right)$$

There are important practical complications.

- The teacher and student may use different tokenizers.
- Their vocabularies may have different sizes.
- Their token boundaries may not match.
- Access to the teacher's logits may be restricted.

For these reasons, response-based distillation is often more convenient when the teacher is a proprietary, black-box API.

3.3 Method 3: Chain-of-thought distillation

A more interesting approach is to teach the student not only the final answer, but also an explanation or intermediate reasoning trace.

Consider the following question:

> A store sells an item for \$120 after applying a 20% discount. What was its original price?

A teacher might generate:

1. Let the original price be x.
2. A 20% discount means the customer pays 80%.
3. Therefore, $0.8x = 120$.
4. Dividing both sides by 0.8 gives $x = 150$.
5. The original price was \$150.

A student trained only on final answers sees:

$$\text{Question} \longrightarrow 150$$

A student trained on reasoning examples sees:

$$\text{Question} \longrightarrow \text{Intermediate steps} \longrightarrow 150$$

This may help the student learn reusable problem-solving patterns.

It is particularly useful for:

- Mathematical reasoning.
- Programming.
- Logical inference.
- Multi-step questions.
- Structured problem solving.

However, an important distinction must be made.

A teacher's explanation is not necessarily a faithful record of its internal computation.

A generated rationale can be a useful teaching artifact without being a literal description of how the model arrived at its answer.

3.4 Method 4: Preference and reward distillation

We can also transfer preferences rather than simply copying individual answers.

Suppose the teacher is given two responses:

- Response A: Technically correct but poorly explained.
- Response B: Correct, clear, and well structured.

The teacher prefers B.

We can use the teacher to generate preference data:

$$(x, y_{\text{preferred}}, y_{\text{rejected}})$$

The student is trained to prefer responses similar to the teacher's preferred responses.

This can be used in preference optimization methods, including variants of Direct Preference Optimization (DPO).

The important idea is that we are transferring a judgment about response quality, rather than merely transferring a particular response.

3.5 Method 5: Multi-teacher distillation

There is no requirement that we use only one teacher.

Suppose we have three models:

- Teacher A: Strong mathematical reasoning.
- Teacher B: Strong programming ability.
- Teacher C: Strong natural-language understanding.

We can generate training examples from each teacher and combine them into one dataset.



This is useful because different teachers may have different strengths.

However, their outputs may also conflict.

We need methods to:

- Identify which teacher is more reliable for a particular task.

- Remove contradictory or low-quality examples.
- Balance contributions from different teachers.
- Avoid overrepresenting one teacher's style.

3.6 Method 6: Self-distillation

A particularly interesting variation is self-distillation.

A model can act as its own teacher.

For example:

1. A model generates answers to selected prompts.
2. Its answers are evaluated or filtered.
3. The resulting examples are used to fine-tune the model or another version of it.
4. The improved model generates additional examples.

This creates an iterative process.

$$M_0 \rightarrow D_0 \rightarrow M_1 \rightarrow D_1 \rightarrow M_2$$

Self-distillation can improve task-specific behavior, but it is not guaranteed to produce continuous improvement.

If the model repeatedly reinforces its own mistakes, the process can become self-reinforcing.

4 Why Does Distillation Reduce Computational Costs?

4.1 The separation between training and inference

We should distinguish two different computational costs.

Training cost:

The cost of creating the student model.

Inference cost:

The cost of using the trained student model to answer questions.

Distillation does not necessarily make the original training process cheap.

In fact, generating millions of examples from a large teacher can be expensive.

The main advantage is that the expensive teacher work can be performed once, while the resulting student can be used many times.

4.2 Analogy: Manufacturing a specialized machine

Imagine a factory that needs to perform one million operations.

Option A: Hire an expert engineer to perform every operation.

Option B: Pay the engineer to design and teach a machine how to perform the operation, then let the machine perform it repeatedly.

The machine has an upfront cost, but its marginal operating cost is much lower.

This is the economic motivation behind distillation.

4.3 Model size and inference

Consider two models:

$$M_T = 70 \text{ billion parameters}$$

$$M_S = 7 \text{ billion parameters}$$

The student has one tenth as many parameters.

If both use the same numerical precision, the approximate weight storage is:

$$\text{Memory} = \text{Parameters} \times \text{Bytes per parameter}$$

At 16-bit precision:

$$M_T \approx 140 \text{ GB}$$

$$M_S \approx 14 \text{ GB}$$

These figures represent weight storage only, not total serving memory.

The student may therefore:

- Fit on fewer GPUs.
- Run on less expensive hardware.
- Support more concurrent users.
- Have lower inference latency.
- Consume less energy.
- Be deployable on local or edge devices.

Actual speed improvements depend on architecture, hardware, quantization, batch size, and memory bandwidth.

A tenfold reduction in parameters does not guarantee a tenfold reduction in latency.

4.4 The break-even calculation

Suppose:

- C_D is the total cost of distillation.
- C_T is the cost per teacher inference.
- C_S is the cost per student inference.
- N is the number of future inference requests.

Using the teacher directly costs:

$$C_{\text{teacher}} = NC_T$$

Distilling and using the student costs:

$$C_{\text{student}} = C_D + NC_S$$

Distillation becomes economically advantageous when:

$$C_D + NC_S < NC_T$$

Rearranging:

$$N > \frac{C_D}{C_T - C_S}$$

This is the break-even number of inference requests.

The larger the deployment volume, the more attractive distillation becomes.

Important observation:

Distillation is particularly attractive for applications where the same kinds of tasks are performed repeatedly.

For example:

- Customer support.
- Document classification.
- Code assistance.
- Data extraction.
- Domain-specific question answering.

A small model trained for a specific purpose may be much more economical than repeatedly calling a general frontier model.

5 The Long-Tail Problem: What Gets Lost?

This is one of the most important limitations of knowledge distillation.

A student may perform remarkably well on common tasks while performing substantially worse on rare or unusual tasks.

This is often described as a long-tail problem.

5.1 What is the long tail?

Consider the distribution of questions submitted to a general-purpose assistant.

Some tasks occur very frequently:

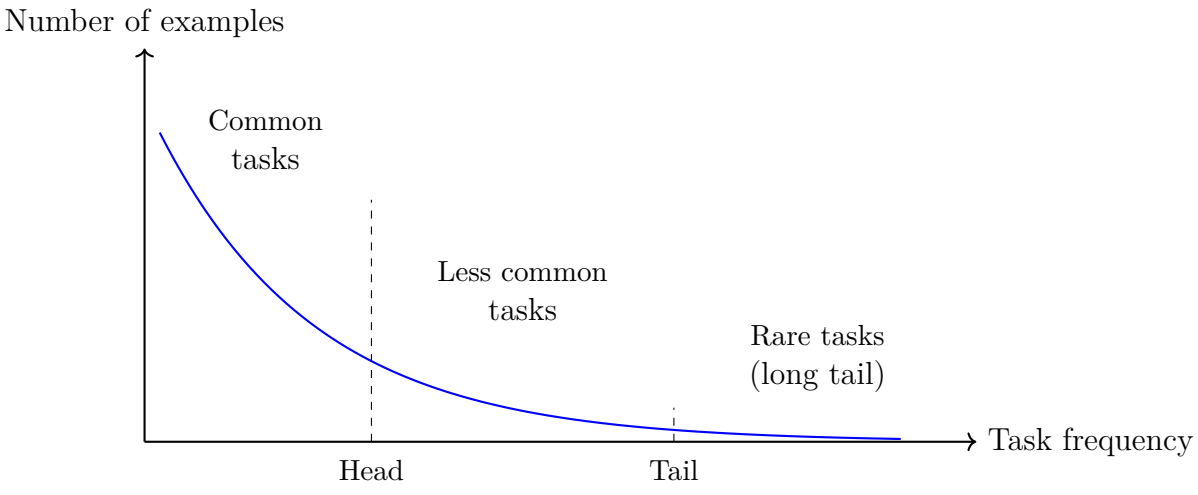
- Summarizing text.
- Answering general knowledge questions.
- Writing simple emails.
- Explaining common programming concepts.

Other tasks are much less frequent:

- Debugging an obscure compiler problem.
- Understanding an unusual legal clause.
- Reasoning about an uncommon mathematical structure.
- Diagnosing an unusual interaction between software systems.
- Solving a highly specialized scientific problem.

These rare tasks form the long tail of the distribution.

5.2 Visualizing the long tail



Most training examples are likely to represent common tasks.

If the student learns primarily from these examples, it may reproduce the teacher's common capabilities but fail to capture its rare capabilities.

5.3 Analogy: A medical textbook

Imagine a medical student studying from a textbook that contains:

- 10,000 examples of common illnesses.
- 100 examples of uncommon illnesses.
- 2 examples of extremely rare conditions.

The student becomes very good at diagnosing common illnesses.

However, the student may never develop the experience necessary to recognize rare conditions.

The student has learned the most frequently represented patterns, not necessarily the full breadth of the expert's knowledge.

The same phenomenon can occur in LLM distillation.

5.4 Why the teacher's capabilities may not transfer

Consider a teacher with the following capabilities:

Capability	Teacher	Student
Common questions	Excellent	Excellent
Basic programming	Excellent	Excellent
Advanced mathematics	Excellent	Good
Rare programming errors	Excellent	Weak
Unusual scientific problems	Excellent	Poor
Specialized reasoning	Excellent	Variable

The student may appear almost as capable as the teacher on ordinary benchmarks.

Yet it may fail on tasks that the teacher handles easily.

This is especially problematic because common benchmarks often measure performance on a limited collection of well-represented tasks.

5.5 The sampling problem

Suppose a teacher can answer 1,000 different categories of questions.

We generate one million training examples.

However, our prompt generator samples tasks according to their real-world frequency.

If 90% of questions belong to 20 common categories, the student receives relatively little information about the remaining 980 categories.

Even if the teacher is excellent at all 1,000 categories, the student only sees a limited sample of its behavior.

We can represent the training distribution as:

$$x \sim P_{\text{train}}(x)$$

But the deployment distribution may be different:

$$x \sim P_{\text{deploy}}(x)$$

The student minimizes expected training loss:

$$\min_{\theta_S} \mathbb{E}_{x \sim P_{\text{train}}} \left[\mathcal{L}(f_S(x), f_T(x)) \right]$$

If:

$$P_{\text{train}} \neq P_{\text{deploy}}$$

then good training performance does not guarantee good deployment performance.

This is a form of distribution shift.

5.6 Capacity limitations

Even with a perfectly balanced dataset, the student may not have enough capacity to reproduce the teacher.

Consider:

$$\text{Teacher capacity} \gg \text{Student capacity}$$

The student must approximate a much more complex function using fewer parameters.

Some capabilities will be preserved, while others will be lost.

This is similar to compressing a high-resolution image into a small thumbnail.

The overall subject remains recognizable, but small details disappear.

The lost details may be precisely the unusual features that matter in specialized situations.

5.7 Why average performance can be misleading

Suppose a student performs as follows:

Task category	Teacher	Student
Common tasks	95%	94%
Uncommon tasks	90%	75%
Rare tasks	85%	35%

A benchmark dominated by common tasks may report excellent overall performance.

But the student has lost a substantial amount of performance on rare tasks.

This raises an important question:

> Are we measuring the average intelligence of the model, or are we measuring its ability to handle the full range of situations that might arise?

Both measurements matter.

5.8 How can we address the long-tail problem?

Several approaches can help.

1. Deliberate prompt diversity

Do not generate training prompts only from common user interactions.

Include:

- Rare technical problems.
- Difficult edge cases.
- Unusual combinations of constraints.
- Adversarial examples.
- Specialized domain questions.

2. Importance sampling

Oversample rare but important categories.

Instead of sampling purely according to frequency, use a modified distribution:

$$P_{\text{sample}}(x) \propto P_{\text{natural}}(x)w(x)$$

where $w(x)$ increases the probability of selecting important or underrepresented examples.

3. Difficulty-based sampling

Generate increasingly difficult problems.

For example:

1. Basic programming questions.
2. Questions involving several programming concepts.
3. Questions involving unusual interactions.
4. Problems requiring multiple independent reasoning steps.

4. Multiple teachers

Use specialized teachers to cover different areas.

A general-purpose model may be supplemented with teachers specializing in mathematics, programming, science, and other domains.

5. Evaluation on rare cases

Create a dedicated evaluation set containing uncommon and difficult examples.

Measure not only average accuracy, but also:

- Worst-case performance.
- Performance on rare categories.
- Robustness to unusual inputs.
- Calibration and uncertainty.

6. Retrieval and tools

Not every capability needs to be stored in the student's parameters.

A smaller model can use external retrieval systems, specialized tools, or other models to handle unusual problems.

This is an important architectural alternative: rather than distilling every capability into one small model, we can build a system of specialized components.

6 Additional Limitations and Failure Modes

The long tail is not the only problem.

6.1 Teacher errors become student training data

Suppose the teacher confidently provides an incorrect answer.

If the student is trained to imitate that answer, the mistake becomes part of its training data.

This can create systematic errors.

The student is not necessarily learning the truth.

It is learning the teacher's behavior.

Therefore, teacher-generated data should ideally be validated through:

- Ground-truth labels.
- Independent verification.
- Multiple teacher agreement.
- Executable tests for generated code.
- Domain-specific validation rules.

6.2 Style imitation versus capability transfer

A student may learn to sound like the teacher without actually learning the underlying capability.

For example, a teacher may produce a convincing mathematical explanation.

The student may learn the style of that explanation without acquiring reliable mathematical reasoning.

This distinction is important:

$$\text{Imitating an explanation} \neq \text{Learning to solve the problem}$$

One way to test this is to evaluate the student on new problems that were not included in the teacher's training examples.

6.3 Mode collapse and reduced diversity

If the student is trained repeatedly on a narrow collection of teacher-generated responses, its outputs may become less diverse.

For example, suppose the teacher can explain a concept in several different ways.

If the dataset contains only one preferred explanation, the student may become overly dependent on that style.

This can reduce flexibility.

6.4 Catastrophic forgetting

Fine-tuning a pretrained student on a narrow synthetic dataset can damage previously learned capabilities.

For example, a general-purpose model fine-tuned extensively on programming examples may become better at programming but worse at other tasks.

This is related to catastrophic forgetting.

Possible mitigations include:

- Mixing original training examples with synthetic data.
- Using smaller learning rates.
- Limiting the number of fine-tuning epochs.
- Using regularization.
- Evaluating general capabilities throughout training.

6.5 The student may learn teacher-specific biases

A student can inherit:

- Incorrect assumptions.
- Cultural or linguistic biases.
- Unwanted response patterns.
- Particular refusal behaviors.
- Limitations of the teacher's training data.

Distillation does not automatically improve safety, fairness, or factual accuracy.

7 The Controversy: Is Distillation Legitimate?

Knowledge distillation is a standard and legitimate machine learning technique.

However, its use with proprietary frontier LLMs has created significant controversy.

The central question is:

> If a company spends billions of dollars developing a powerful model, should another company be allowed to systematically query that model and use its responses to train a competing model?

There are several different perspectives.

7.1 Why companies use distillation

A company developing an LLM may use its own large model to generate synthetic training data for a smaller model.

This is a natural and useful approach.

For example:

1. Train a large model.
2. Use it to generate millions of high-quality examples.
3. Filter and validate those examples.
4. Fine-tune a smaller model.
5. Deploy the smaller model at a lower cost.

This allows a company to offer:

- Faster models.
- Cheaper models.
- Specialized models.
- Models that can operate locally.

This is generally not controversial when the company owns the teacher model and has the right to use its outputs for this purpose.

7.2 Distillation using another company's model

Now consider a different scenario.

Company A spends substantial resources training a frontier model.

Company B obtains access to that model through an API.

Company B systematically generates millions of prompts and collects the model's responses.

It then uses these responses to train its own model.

The resulting model may reproduce some of Company A's capabilities without Company B having incurred the same original development costs.

This raises several questions.

1. Is the process permitted by the API's terms of service?
2. Does it violate intellectual property rights?
3. Does it constitute unfair competition?
4. Should model outputs be treated as protected intellectual property?
5. Is there a difference between learning from publicly available information and systematically extracting a model's behavior?

These questions do not all have the same legal answer.

The fact that an API permits a user to obtain an answer does not necessarily mean it permits unlimited systematic extraction for training a competing model.

Conversely, the technical ability to learn from outputs does not, by itself, establish that every such use is unlawful.

The applicable contracts, jurisdiction, copyright rules, and other legal doctrines matter.

7.3 The DeepSeek controversy

One prominent example concerns DeepSeek.

The development of DeepSeek's reasoning models generated significant discussion about whether frontier models had been used as teachers.

OpenAI and other AI companies have made allegations that certain organizations used systematic, large-scale querying of proprietary models to extract capabilities.

In February 2026, Anthropic also publicly described what it characterized as industrial-scale distillation campaigns targeting Claude.

The company reported that three AI laboratories, including DeepSeek, Moonshot, and MiniMax, generated over 16 million exchanges through approximately 24,000 fraudulent accounts.

According to Anthropic, the targeted capabilities included:

- Agentic reasoning.
- Tool use.

- Programming.
- Reasoning traces.

OpenAI has also publicly described efforts to detect and disrupt unauthorized distillation activity.

These are company-reported allegations and should be distinguished from independently adjudicated findings.

It is important for students to understand that the technical existence of distillation is not itself evidence of misconduct.

The controversy concerns how the teacher was accessed, what outputs were collected, what restrictions applied, and how the resulting model was developed.

7.4 The safety controversy

There is another concern beyond intellectual property.

Suppose a frontier model has undergone extensive safety training.

It may have:

- Refusal mechanisms.
- Safety classifiers.
- Restrictions on dangerous instructions.
- Monitoring and abuse detection.
- Extensive red-team testing.

Another organization could potentially train a student to reproduce useful reasoning capabilities without reproducing the same safety infrastructure.

The result might be a smaller model with strong technical capabilities but weaker safeguards.

This creates an important distinction:

$$\text{Capability transfer} \neq \text{Safety transfer}$$

A model can inherit useful capabilities without inheriting the processes, evaluations, and safeguards that made the teacher suitable for deployment.

7.5 An analogy: Learning from a competitor

Imagine two pharmaceutical companies.

Company A spends billions developing a new drug and conducting extensive research.

Company B obtains the results of Company A's experiments and uses them to accelerate development of its own drug.

There are legitimate and illegitimate ways to learn from another company's research.

Public scientific publications are one thing.

Unauthorized access to proprietary experiments is another.

The same distinction is relevant to model distillation.

The difficult question is where to draw the boundary between legitimate learning and unauthorized extraction of proprietary capabilities.

7.6 A broader philosophical question

There is also a more fundamental question:

Should intelligence be treated as a proprietary asset?

Human beings learn by observing other people.

A student can learn from a professor.

A scientist can reproduce another scientist's results.

A programmer can study software behavior and implement a similar system.

Machine learning itself is based on extracting patterns from data.

At what point does learning from another system become impermissible copying?

This is not merely a technical question.

It involves economics, intellectual property, competition, and the philosophy of knowledge.

8 Putting Everything Together

8.1 What knowledge distillation actually accomplishes

Knowledge distillation allows a smaller model to learn from a more capable model.

The teacher can provide:

- Final answers.
- Token probability distributions.
- Intermediate representations.
- Explanations and reasoning examples.
- Preferences and evaluations.
- Synthetic training data.

The student uses these signals to approximate selected aspects of the teacher's behavior.

8.2 The main trade-offs

Dimension	Benefit	Limitation
Model size	Smaller models require less memory.	Reduced representational capacity.
Inference	Lower serving costs and potentially lower latency.	Student may be less accurate on difficult tasks.
Training data	Teacher can generate large amounts of useful data.	Synthetic data may contain systematic errors.
Reasoning	Explanations can transfer useful problem-solving patterns.	Explanations may not represent actual internal reasoning.
Generalization	Teacher outputs can provide richer supervision.	Rare tasks may be underrepresented.
Deployment	Models can run on cheaper hardware.	Safety and reliability must be evaluated independently.
Competition	Accelerates research and model development.	May raise intellectual property and contractual issues.

8.3 An important distinction: Distillation versus compression

These terms are related but not identical.

Model compression is the broader goal of reducing the computational or storage requirements of a model.

Examples include:

- Quantization.
- Pruning.
- Weight sharing.
- Low-rank approximation.

Knowledge distillation transfers information from one model to another through training.

It can be used as a model compression technique, but it can also be used to create specialized models, transfer capabilities, or improve training data.

For example, a student can be trained using a teacher without necessarily being dramatically smaller.

Similarly, a quantized model can be compressed without using a teacher at all.

8.4 The future: Smaller specialized models

One interesting possibility is that the future of AI will not consist exclusively of increasingly large general-purpose models.

Instead, we may see systems composed of:

- A relatively small general-purpose model.
- Specialized models for mathematics or coding.
- Models optimized for particular industries.
- Retrieval systems for external knowledge.
- Larger models called only when necessary.

Distillation makes it possible to transfer useful capabilities into specialized components.

This could produce more economical and efficient AI architectures.

9 Conclusion

Knowledge distillation is one of the important techniques for making modern AI systems more efficient and accessible.

Its central insight is simple:

> We do not always need to independently train a smaller model to discover everything that a larger model has already learned. We can use the larger model's behavior as a source of supervision.

However, there are several important qualifications:

1. Distillation transfers behavior, not necessarily the teacher's internal knowledge or representations.
2. A student can approximate a teacher on common tasks while losing important capabilities on rare and difficult problems.
3. Teacher-generated data must be evaluated: errors, biases, and limitations can propagate.
4. Distillation can substantially reduce inference costs, but generating the training data can itself be expensive.
5. Capability transfer does not guarantee safety transfer.
6. The use of proprietary models as teachers introduces difficult questions about intellectual property, competition, and acceptable use.

The most important practical lesson is that a successful distillation project is not simply about creating a smaller model.

It is about deciding which capabilities must be preserved, which examples are necessary to teach them, and how to verify that the resulting model behaves reliably across both common and unusual situations.