The Illustrated DeepSeek-R1

A recipe for reasoning LLMs

JAY ALAMMAR
JAN 28, 2025

[Draft post, updates to come, please let me know if you have any suggestions or feedback here or on Bluesky or X/Twitter]

DeepSeek-R1 is the latest resounding beat in the steady drumroll of AI progress. For the ML R&D community, it is a major release for reasons including:

. It is an open weights model with smaller, distilled versions and
. It shares and reflects upon a training method to reproduce a reasoning model like OpenAI O1.

In this post, we'll see how it was built.

Contents:

3- Creating reasoning models with large-scale reinforcement learning (RL)

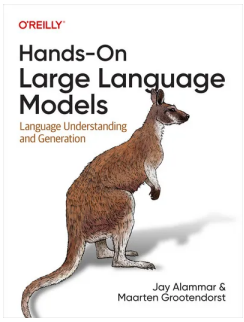
3.1- Large-Scale Reasoning-Oriented Reinforcement Learning (R1-Zero)

3.2- Creating SFT reasoning data with the interim reasoning model

3.3- General RL training phase

Architecture

Most of the foundational knowledge you need to understand how such a model works is available in our book, Hands-On Large Language Models.



Official website of the book. You can order the book on Amazon.
All code is uploaded to GitHub.

Recap: How LLMs are trained

Just like most existing LLMs, DeepSeek-R1 generates one token at a time, except it excels at solving math and reasoning problems because it is able to spend more time processing a problem through the process of generating thinking tokens that explain its chain of thought.

The following figure, from Chapter 12 of our book shows the general recipe of creating a high-quality LLM over three steps:

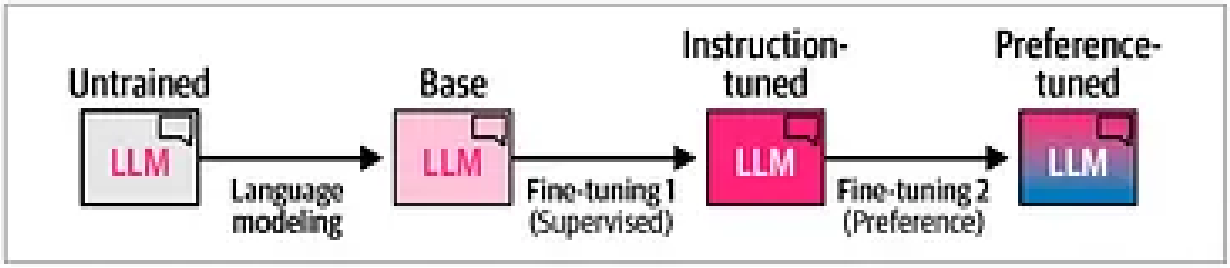


Figure 12-3. The three steps of creating a high-quality LLM.

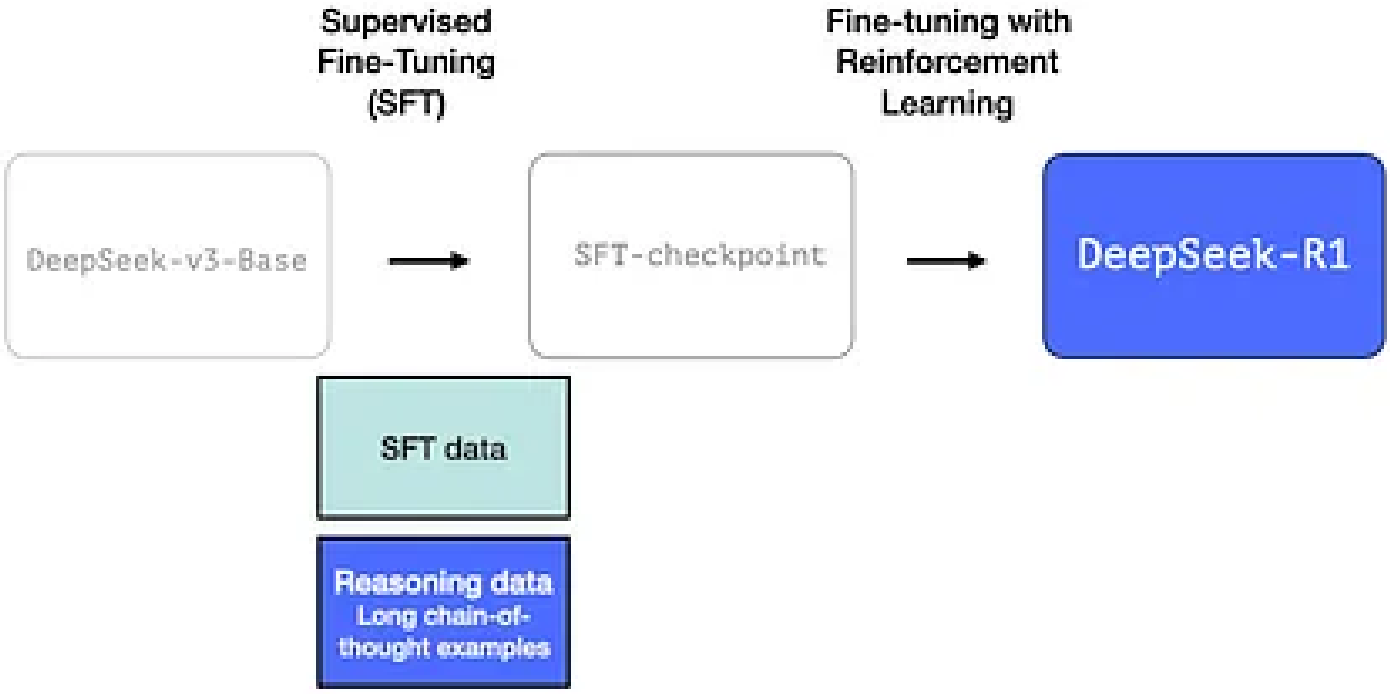
1) The language modeling step where we train the model to predict the next word using a massive amount of web data. This step results in a base model.
2) a supervised fine-tuning step that makes the model more useful in following instructions and answering questions. This step results in an instruction tuned model or a supervised fine - tuning / SFT model.
3) and finally a preference tuning step which further polishes its behaviors and aligns to human preferences, resulting in the final preference-tuned LLM which you interact with on playgrounds and apps.

DeepSeek-R1 Training Recipe

DeepSeek-R1 follows this general recipe. The details of that first step come from a previous paper for the DeepSeek-V3 model. R1 uses the *base* model (not the final DeepSeek-v3 model) from that previous paper, and still goes through an SFT and preference tuning steps, but the details of how it does them are what's different.

There are three special things to highlight in the R1 creation process.

1- Long chains of reasoning SFT Data

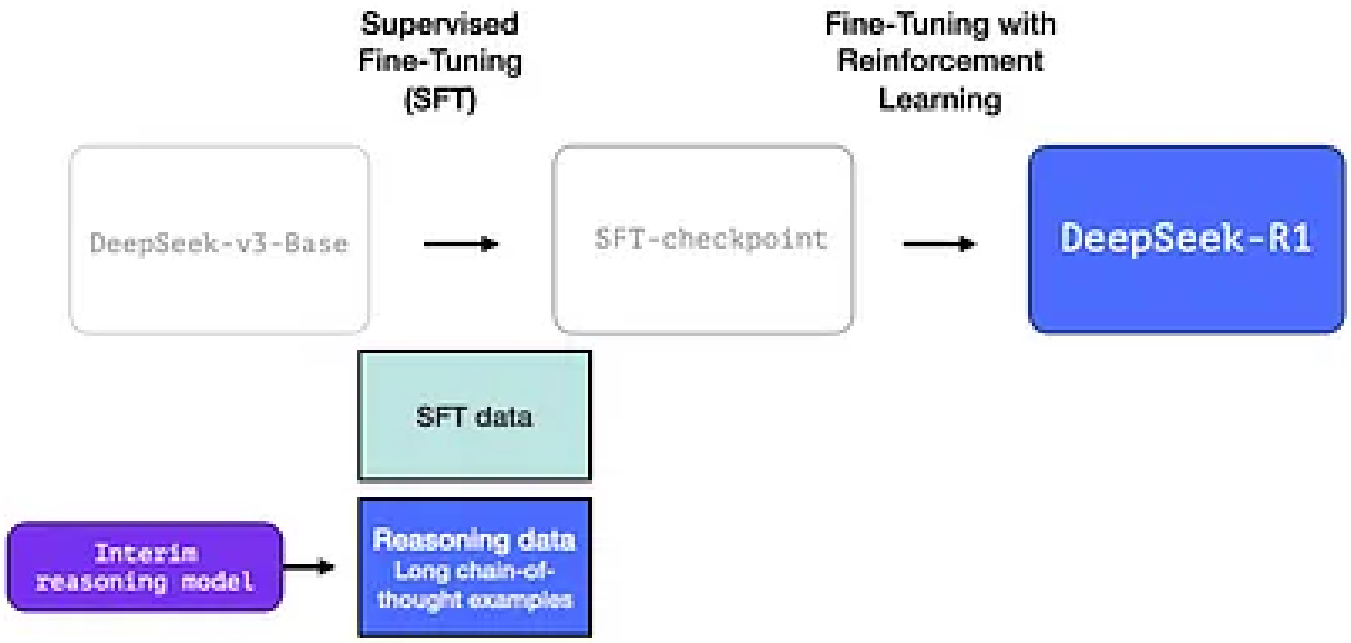


This is a large number of long chain-of-thought reasoning examples (600,000 of them). These are very hard to come by and very expensive to label with humans at this scale. Which is why the process to create them is the second special thing to highlight

2- An interim high-quality reasoning LLM (but worse at non-reasoning tasks).

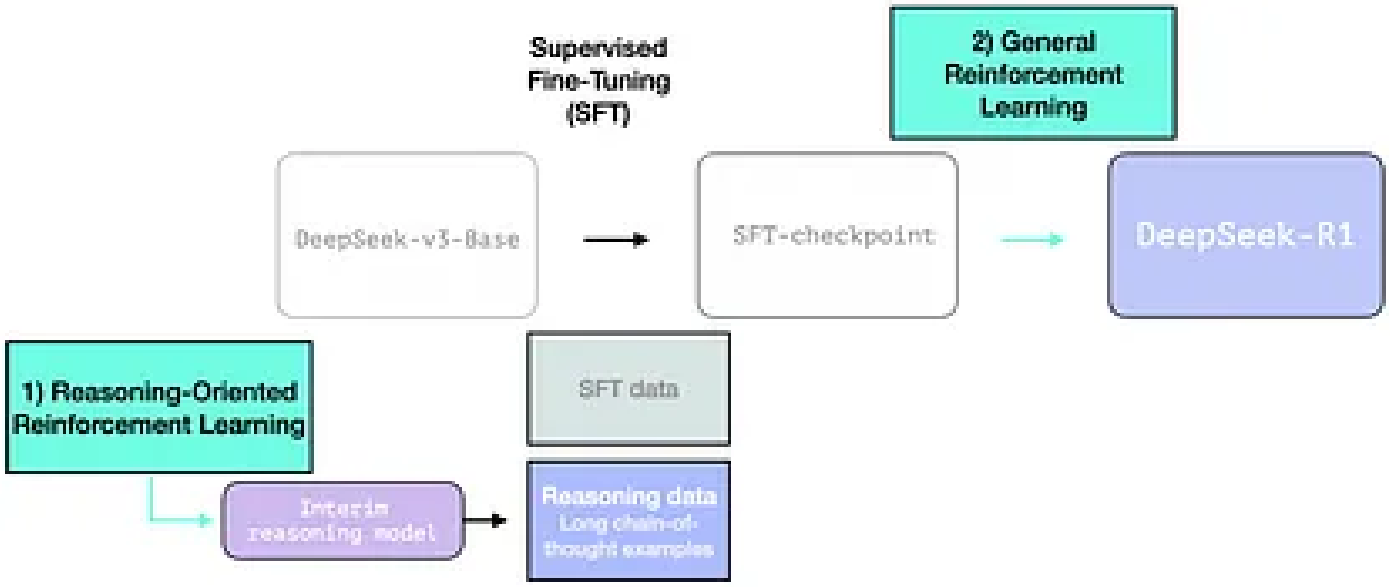
This data is created by a precursor to R1, an unnamed sibling which specializes in reasoning. This sibling is inspired by a third model called *R1-Zero* (that we'll discuss shortly). It is significant not because it's a great LLM to use, but because creating it required so little labeled data alongside large-scale reinforcement learning resulting in a model that excels at solving reasoning problems.

The outputs of this unnamed specialist reasoning model can then be used to train a more general model that can also do other, non-reasoning tasks, to the level users expect from an LLM.



3- Creating reasoning models with large-scale reinforcement learning (RL)

This happens in two steps:



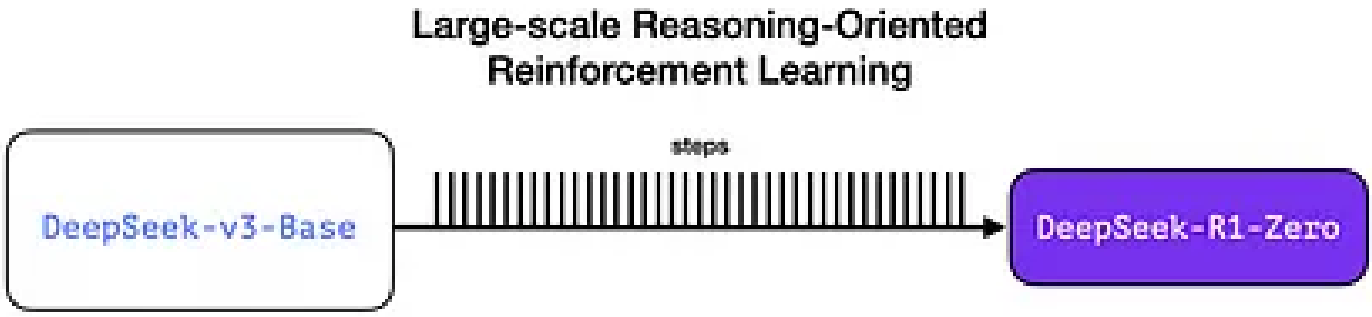
3.1 Large-Scale Reasoning-Oriented Reinforcement Learning (R1-Zero)

Here, RL is used to create the interim reasoning model. The model is then used to generate the SFT reasoning examples. But what makes creating this model possible is an earlier experiment creating an earlier model called *DeepSeek-R1-Zero*.

Model	AIME 2024		MATH-500	GPQA Diamond	LiveCode Bench	CodeForces
	pass@1	cons@64	pass@1	pass@1	pass@1	rating
OpenAI-o1-mini	63.6	80.0	90.0	60.0	53.8	1820
OpenAI-o1-0912	74.4	83.3	94.8	77.3	63.4	1843
DeepSeek-R1-Zero	71.0	86.7	95.9	73.3	50.0	1444

Table 2 | Comparison of DeepSeek-R1-Zero and OpenAI o1 models on reasoning-related benchmarks.

R1-Zero is special because it is able to excel at reasoning tasks without having a labeled SFT training set. Its training goes directly from a pre-trained base model through a RL training process (no SFT step). It does this so well that it's competitive with o1.



This is significant because data has always been the fuel for ML model capability. How can this model depart from that history? This points to two things:

1- Modern base models have crossed a certain threshold of quality and capability (this base model was trained on 14.8 trillion high-quality tokens).

2- Reasoning problems, in contrast to general chat or writing requests, can be automatically verified or labeled. Let's show this with an example. This can be a prompt/question that is a part of this RL training step:

Write python code that takes a list of numbers, returns them in a sorted order, but also adds 42 at the start.

A question like this lends itself to many ways of automatic verification. Say we present this this to the model being trained, and it generates a completion:

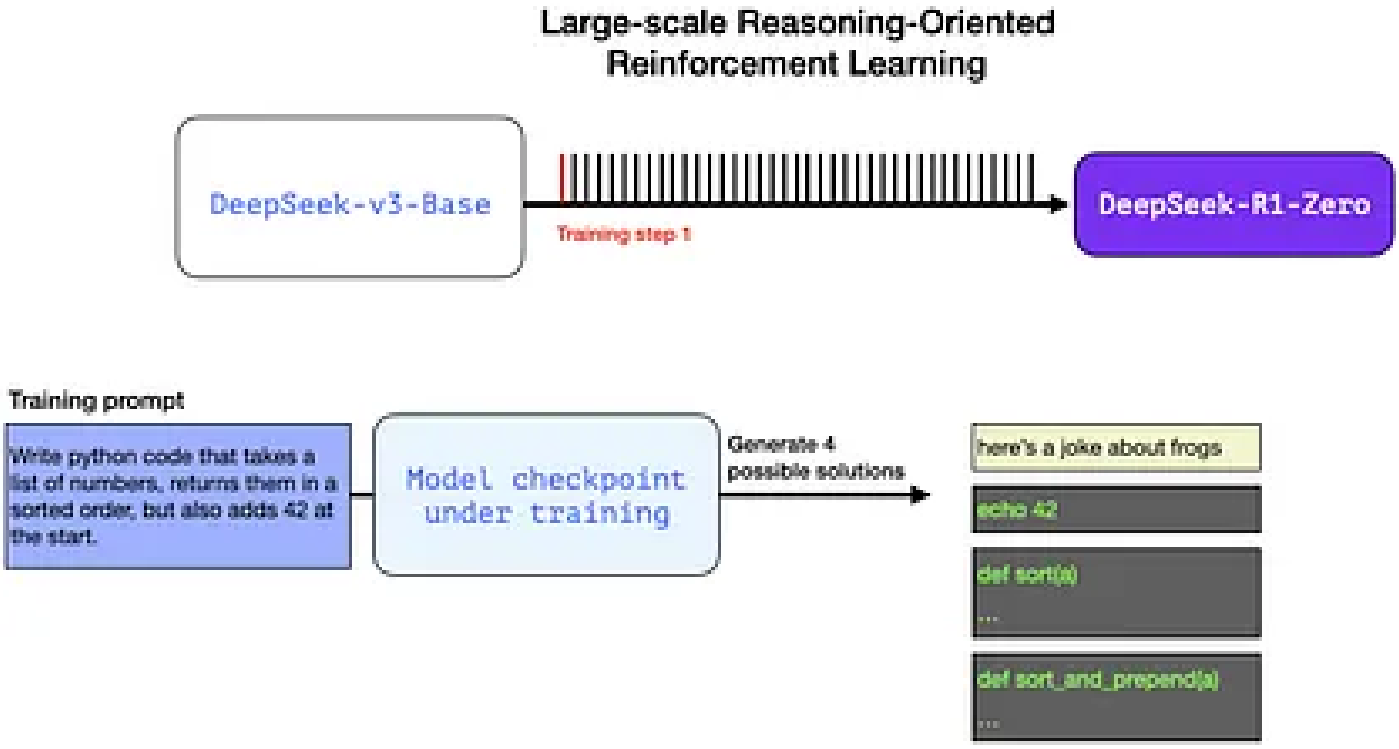
A software linter can check if the completion is proper python code or not

We can execute the python code to see if it even runs

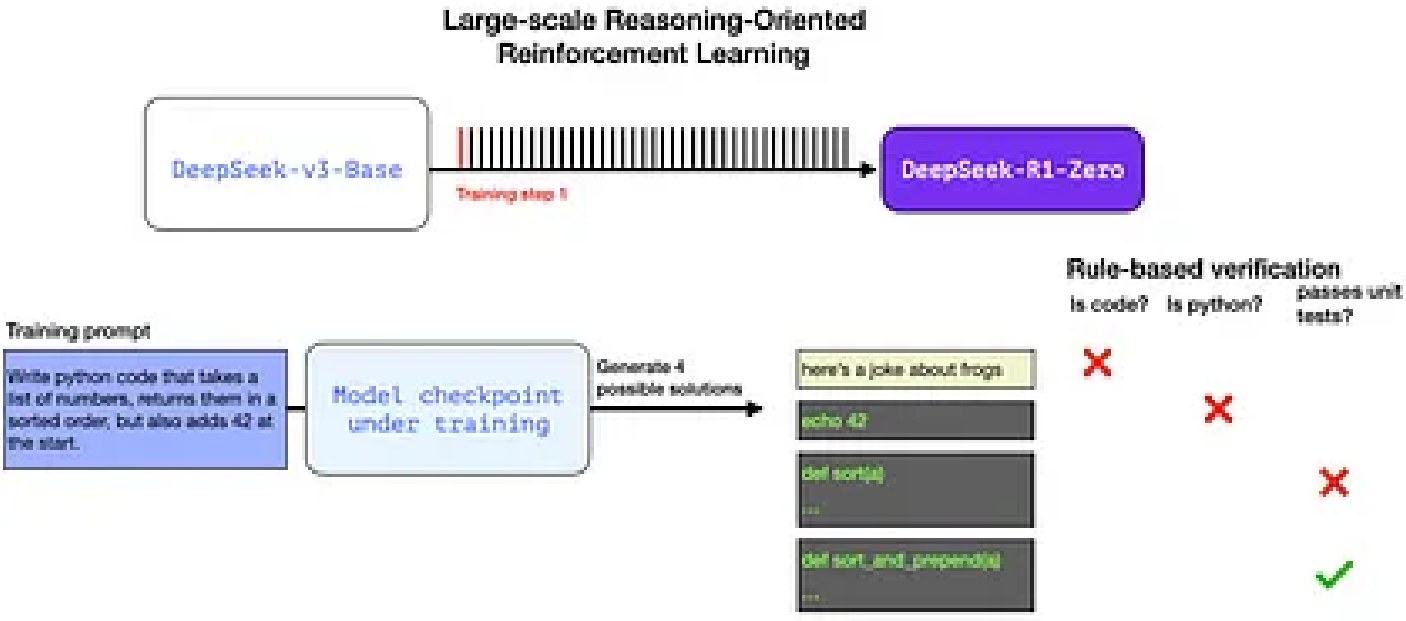
Other modern coding LLMs can create unit tests to verify the desired behavior (without being reasoning experts themselves).

We can go even one step further and measure execution time and make the training process prefer more performant solutions over other solutions — even if they're correct python programs that solve the issue.

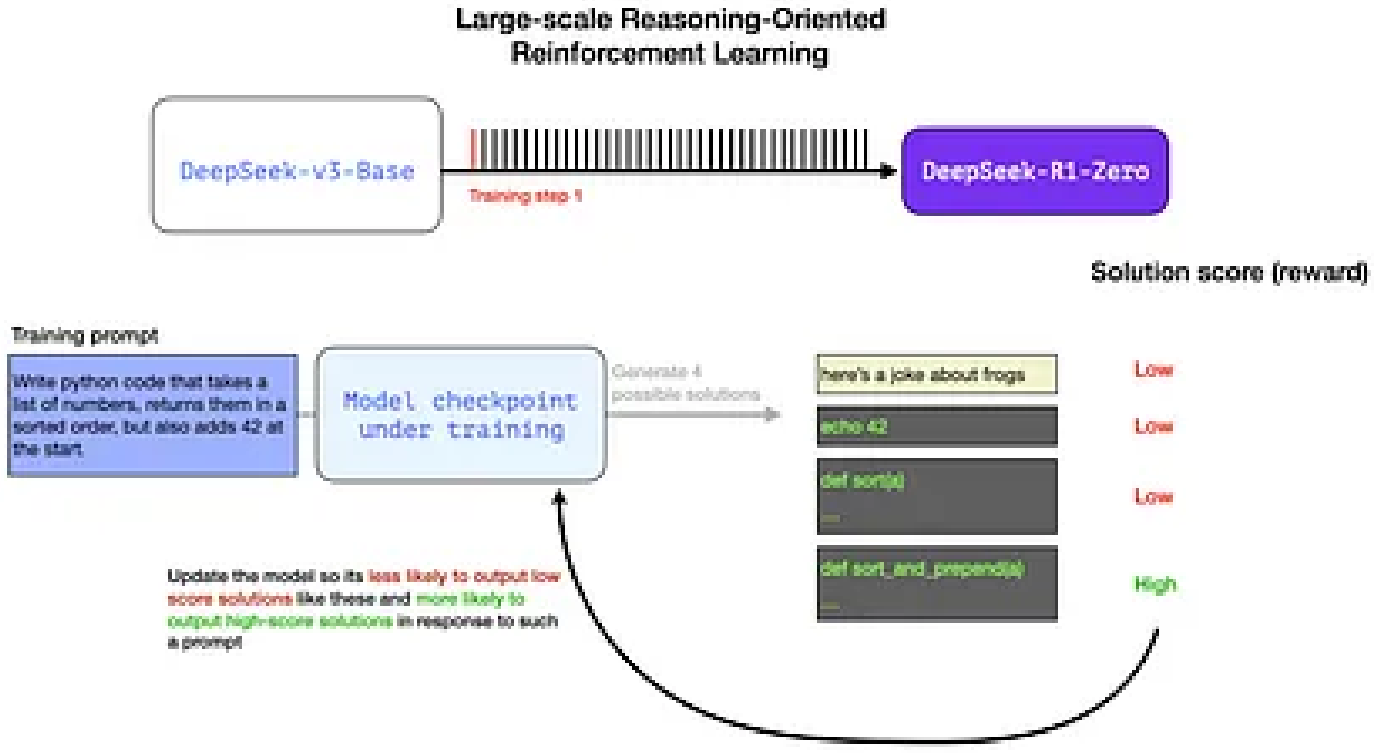
We can present a question like this to the model in a training step, and generate multiple possible solutions.



We can automatically check (with no human intervention) and see that the first completion is not even code. The second one is indeed python code but does not solve the problem. The third is a possible solution, but fails the unit tests, and the forth is a correct solution.



These are all signals that can be directly use to improve the model. This is of course done over many examples (in mini-batches) and over successive training steps.



These reward signals and model updates are how the model continues improving on tasks over the RL training process as seen in Figure 2 from the paper.

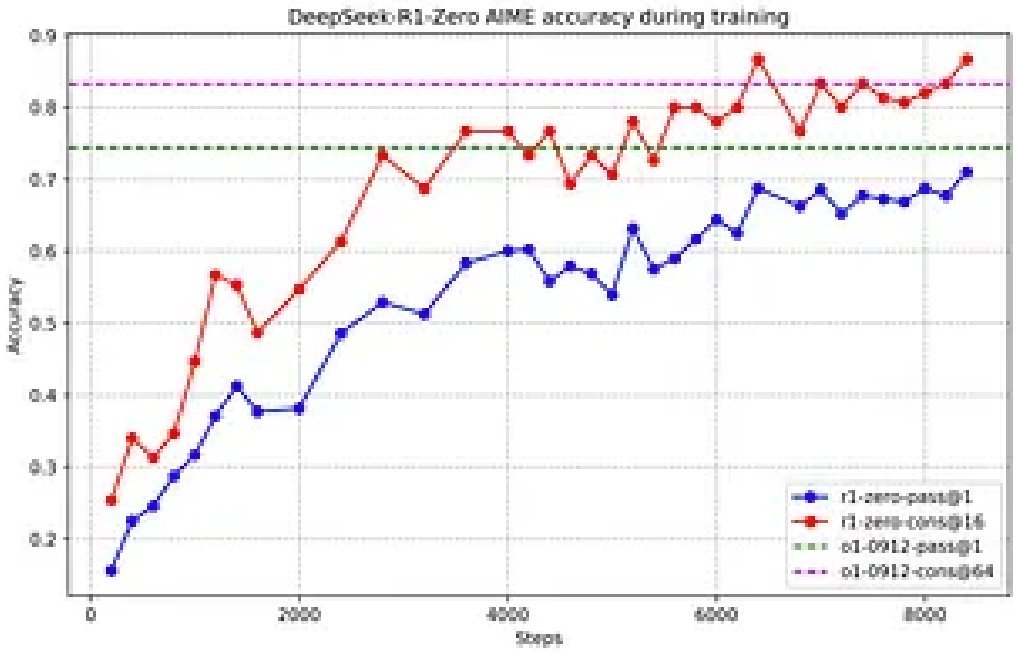


Figure 2 | AIME accuracy of DeepSeek-R1-Zero during training. For each question, we sample 16 responses and calculate the overall average accuracy to ensure a stable evaluation.

Corresponding with the improvement of this capability is the length of the generated response, where the model generates more thinking tokens to process the problem.

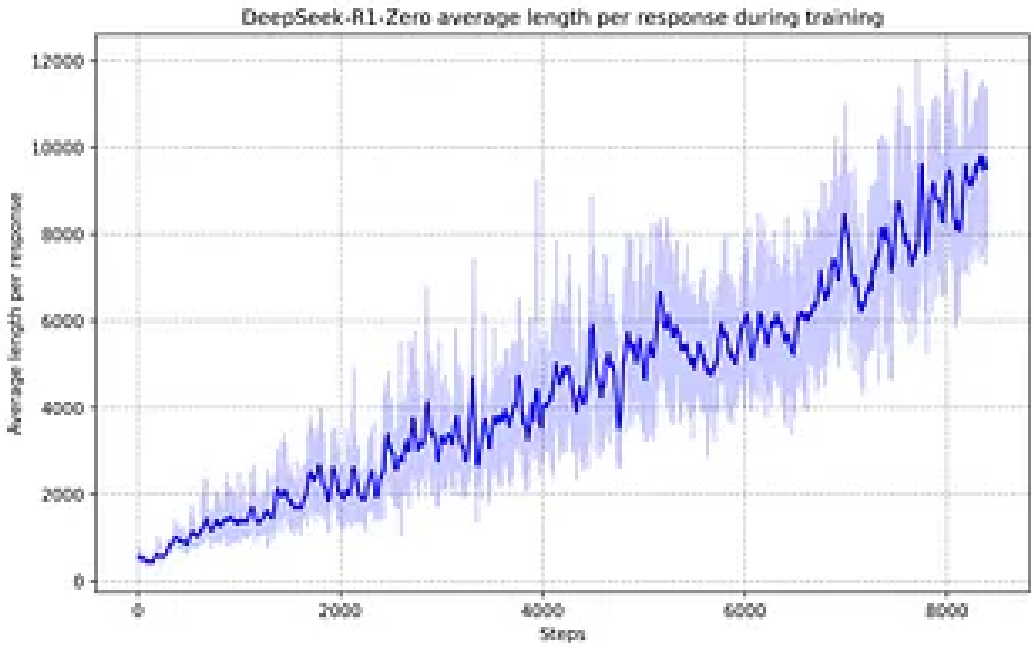
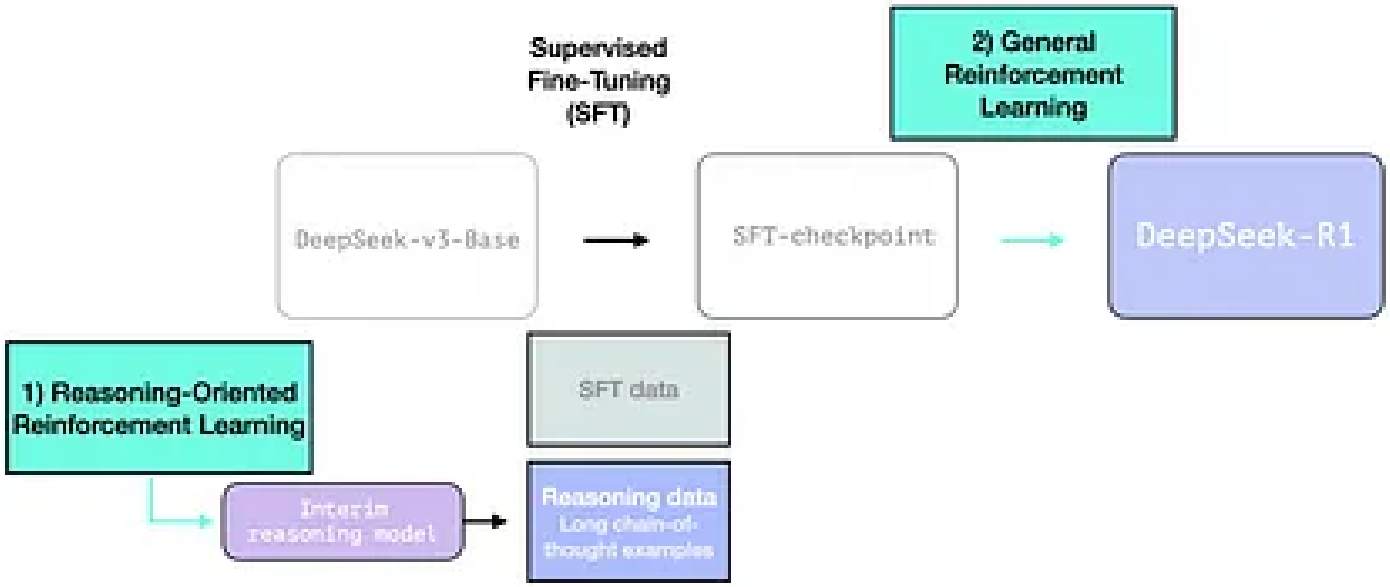


Figure 3 | The average response length of DeepSeek-R1-Zero on the training set during the RL process. DeepSeek-R1-Zero naturally learns to solve reasoning tasks with more thinking time.

This process is useful, but the R1-Zero model, despite scoring high on these reasoning problems, confronts other issues that make it less usable than desired.

Although DeepSeek-R1-Zero exhibits strong reasoning capabilities and autonomously develops unexpected and powerful reasoning behaviors, it faces several issues. For instance, DeepSeek-R1-Zero struggles with challenges like poor readability, and language mixing.

R1 is meant to be a more usable model. So instead of relying completely on the RL process, it is used in two places as we've mentioned earlier in this section:
1- creating an interim reasoning model to generate SFT data points
2- Training the R1 model to improve on reasoning and non-reasoning problems (using other types of verifiers)

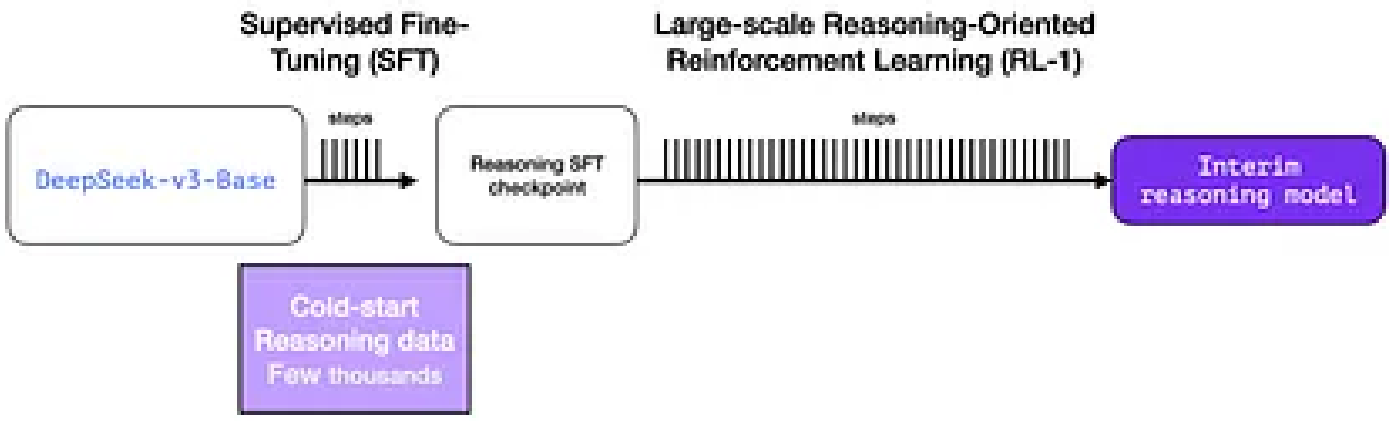


3.2 Creating SFT reasoning data with the interim reasoning model

To make the interim reasoning model more useful, it goes through an supervised fine-tuning (SFT) training step on a few thousand examples of reasoning problems (some of which are generated and filtered from R1-Zero). The paper refers to this as cold start data”

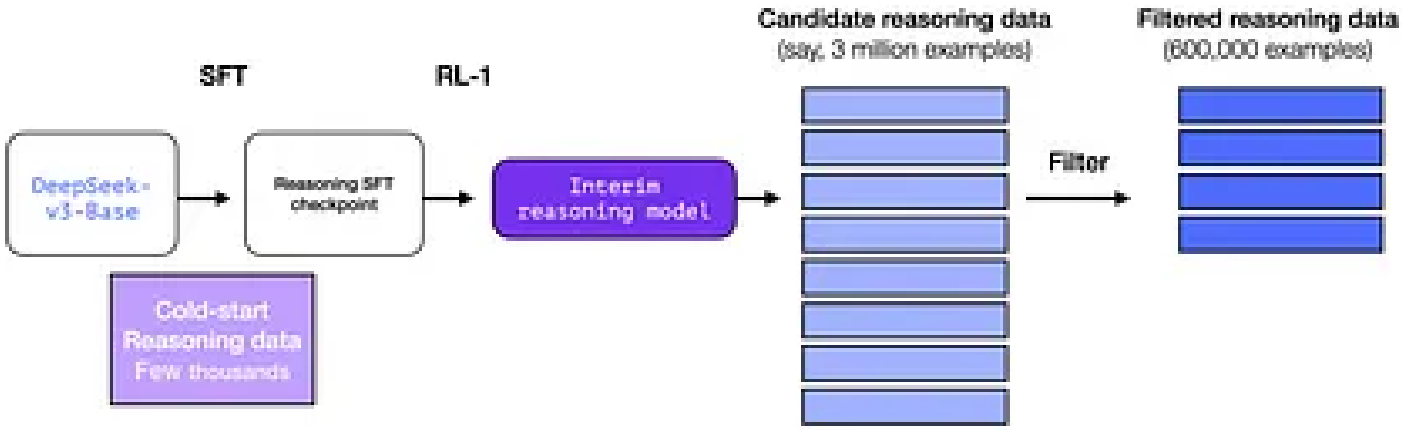
2.3.1. Cold Start

Unlike DeepSeek-R1-Zero, to prevent the early unstable cold start phase of RL training from the base model, for DeepSeek-R1 we construct and collect a small amount of long CoT data to fine-tune the model as the initial RL actor. To collect such data, we have explored several approaches: using few-shot prompting with a long CoT as an example, directly prompting models to generate detailed answers with reflection and verification, gathering DeepSeek-R1-Zero outputs in a readable format, and refining the results through post-processing by human annotators.



But wait, if we have this data, then why are we relying on the RL process? It’s because of the scale of the data. This dataset might be 5,000 examples (which is possible to source), but to

train R1, 600,000 examples were needed. This interim model bridges that gap and allows to synthetically generate that extremely valuable data.



If you're new to the concept of Supervised Fine-Tuning (SFT), that is the process that presents the model with training examples in the form of prompt and correct completion. This figure from chapter 12 shows a couple of SFT training examples:

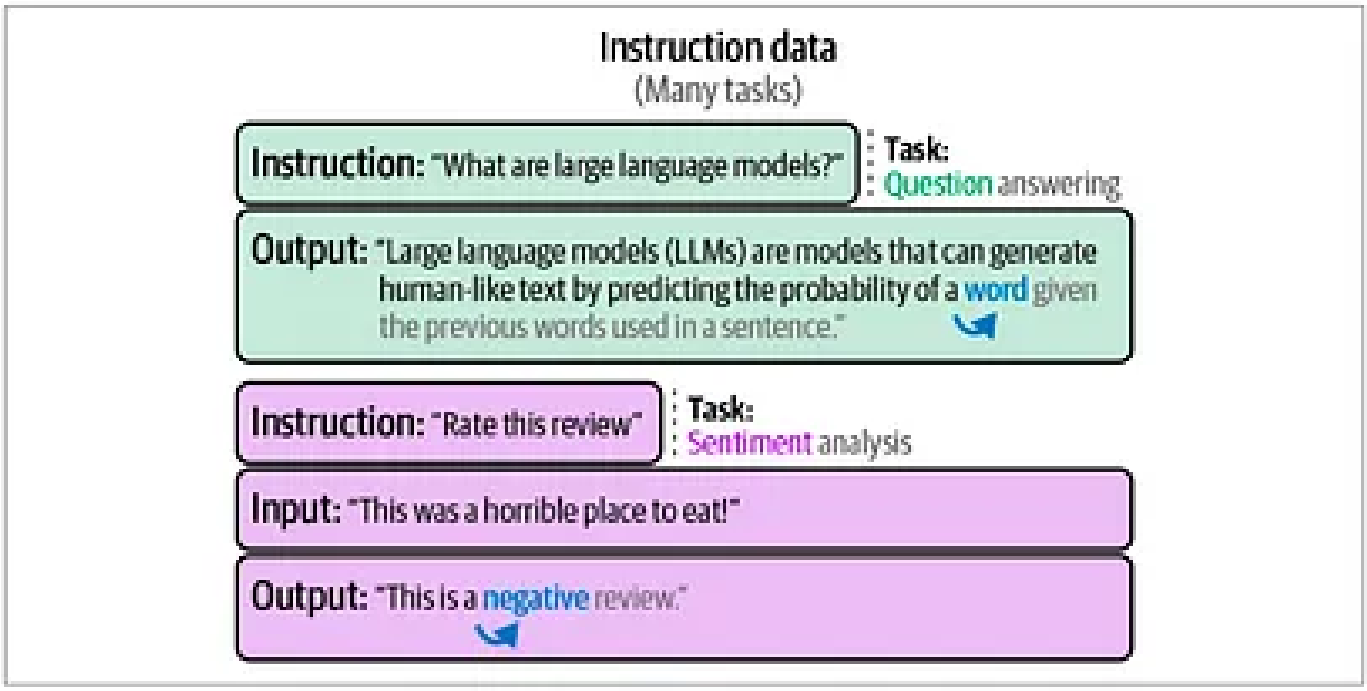
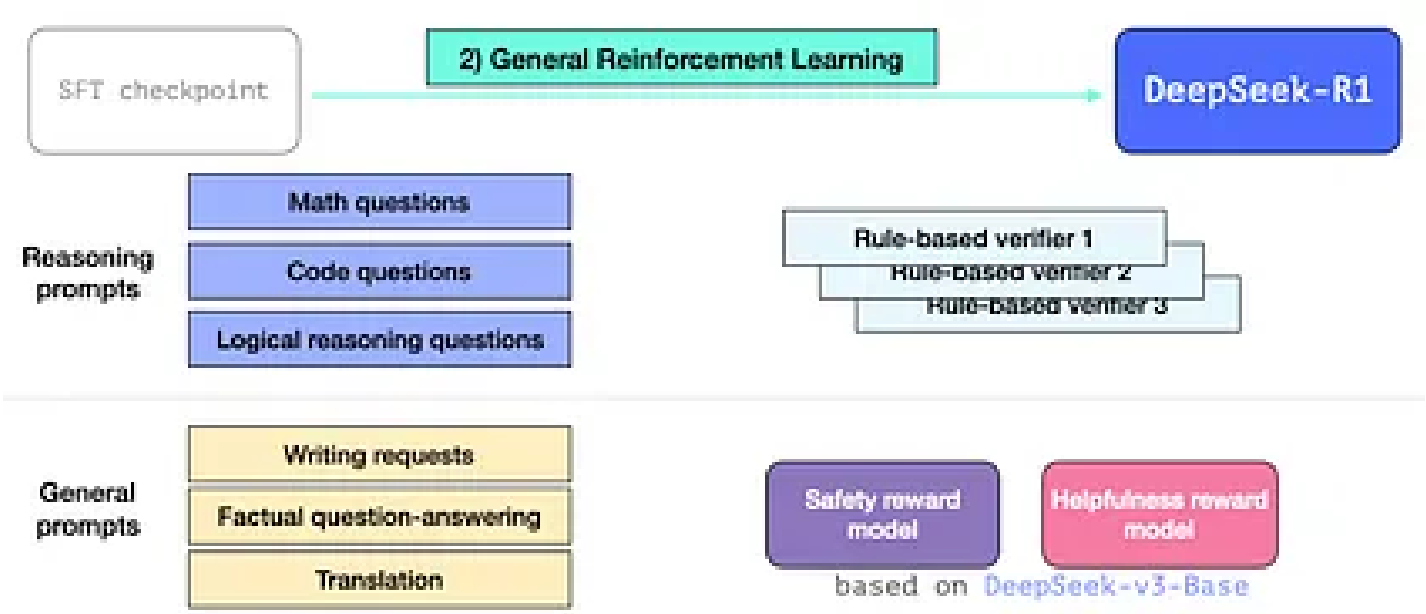


Figure 12-7. Instruction data with instructions by a user and corresponding answers. The instructions can contain many different tasks.

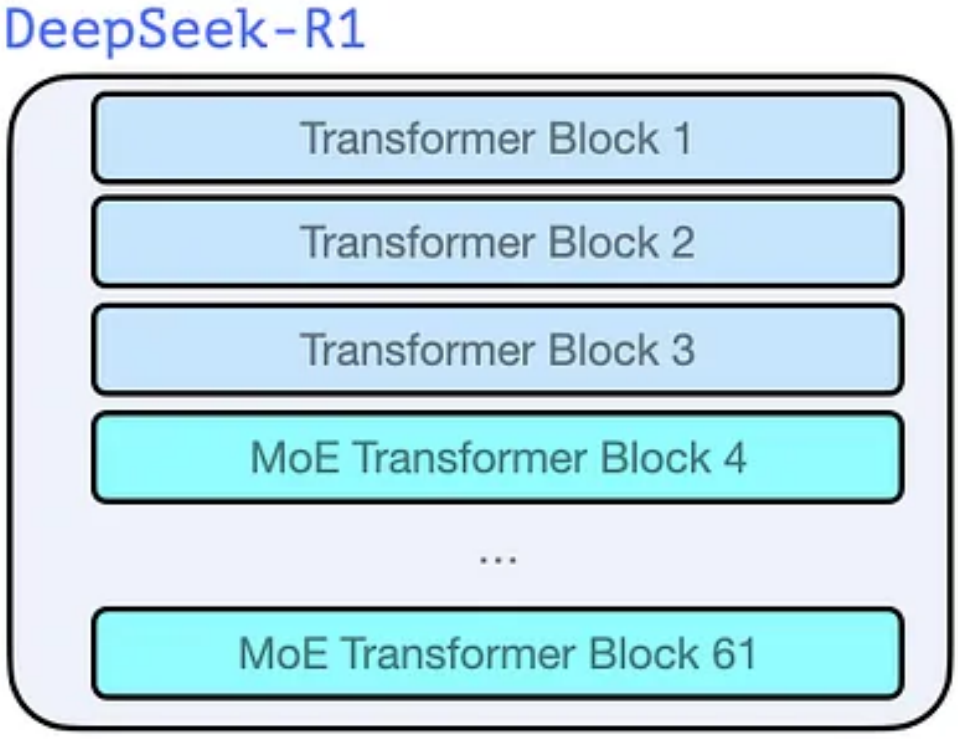
3.3 General RL training phase

This enables R1 to excel at reasoning as well as other non-reasoning tasks. The process is similar to the the RL process we've seen before. But since it extends to non-reasoning applications, it utilizes a helpfulnes and a safety reward model (not unlike the Llama models) for prompts that belong to these applications.

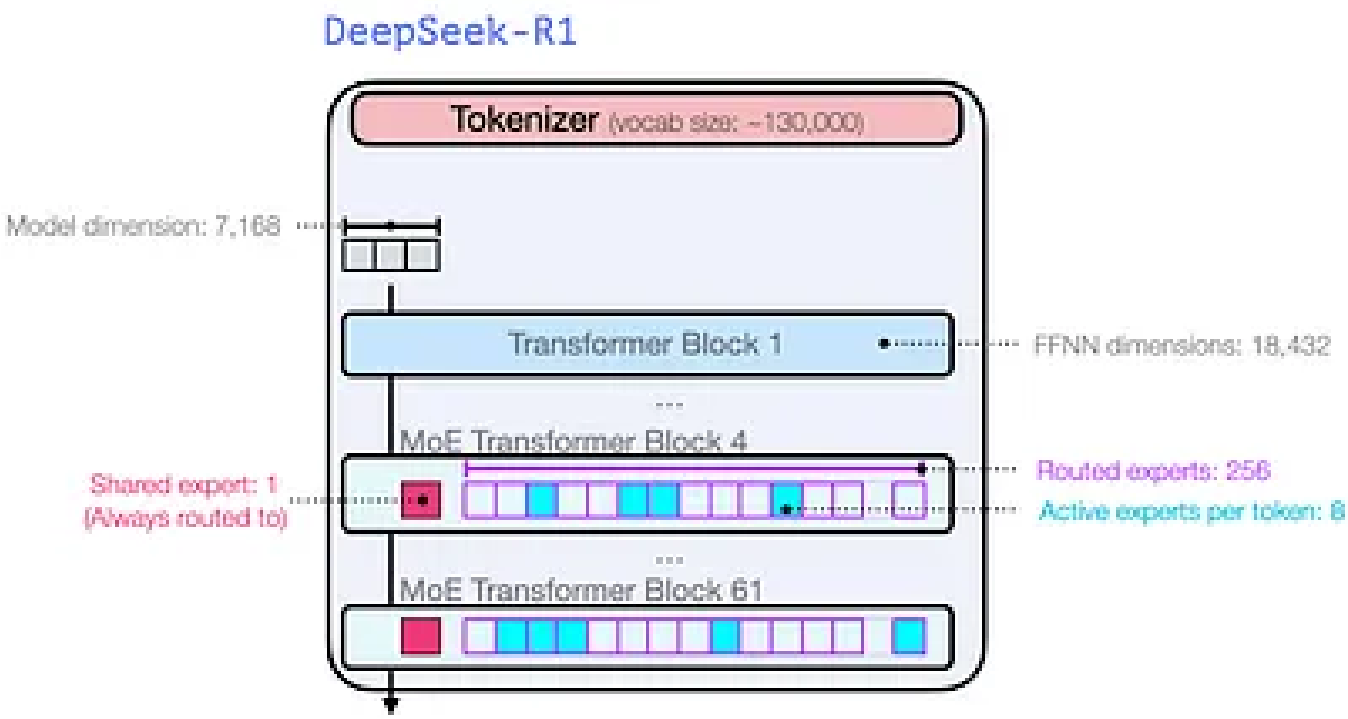


Architecture

Just like previous models from the dawn of GPT2 and GPT 3, DeepSeek-R1 is a stack of Transformer decoder blocks. It's made up 61 of them. The first three are dense, but the rest are mixture-of-experts layers (See my co-author Maarten's incredible intro guide here: A Visual Guide to Mixture of Experts (MoE)).



In terms of model dimension size and other hyperparameters, they look like this:



More details about the model architecture are presented in their two earlier papers:

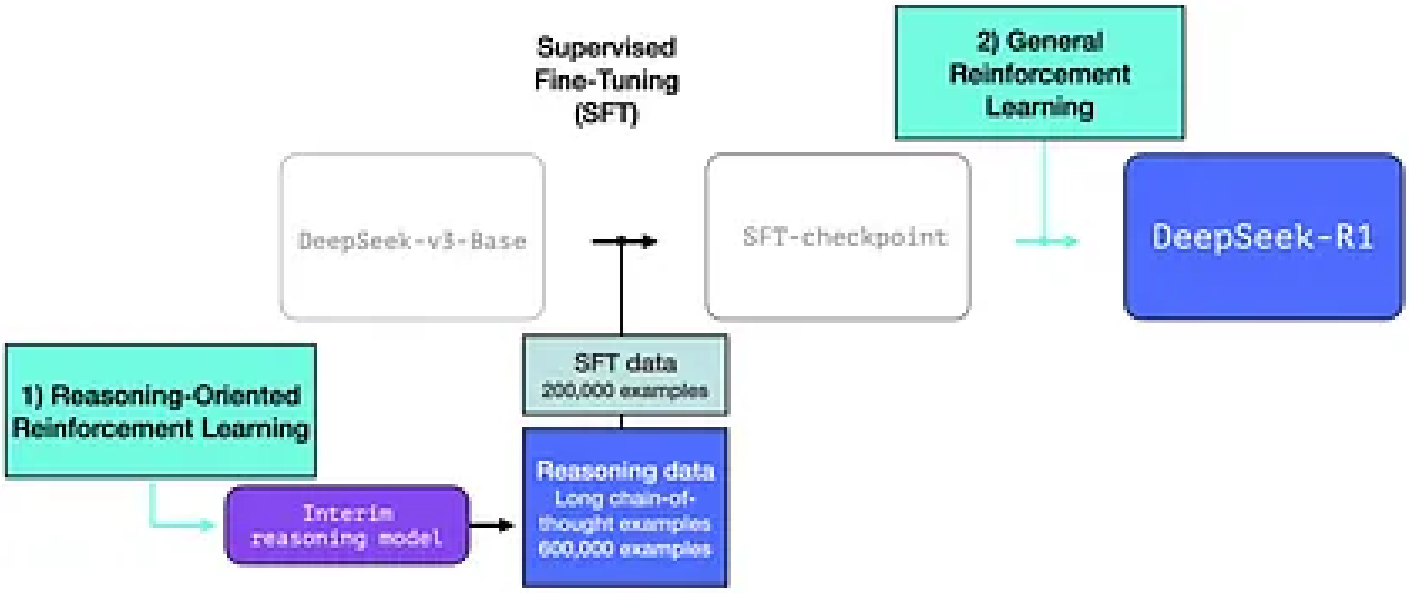
DeepSeek-V3 Technical Report

DeepSeekMoE: Towards Ultimate Expert Specialization in

Mixture-of-Experts Language Models

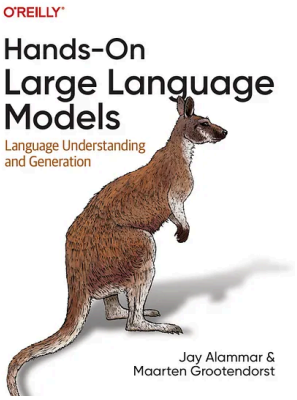
Conclusion

With this, you should now have the main intuitions to wrap your head around the DeepSeek-R1 model.



If you felt needed a little more foundational information to understand this post, I'd suggest you pick up a copy of Hands-On Large Language Models or read it online on O'Reilly and

check it out on Github.



Other suggested resources are:

DeepSeek R1's recipe to replicate o1 and the future of reasoning LMs by Nathan Lambert

A Visual Guide to Mixture of Experts (MoE) by Maarten Grootendorst

Sasha Rush's YouTube video Speculations on Test-Time Scaling (o1)

Yannis Kilcher's DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models (Paper Explained)

Open R1 is the HuggingFace project to openly reproduce DeepSeek-R1

Putting RL back in RLHF