# FORKING — Q&A Reading Script

Shaoyang Guo (Sean) | Host: Cecile | 2026-10-21

Question wording is frozen — identical to the list sent to the host. Answers are the round-11 prepared versions (Q5 includes the single-epoch damage addition; Q8 opens with the AI/RSI acceleration context and closes with the science-of-AI prototype / ArchitectureIQ).

---

## Main questions

### 1. When did you first realize that forking was more than just ordinary overfitting?

Honestly, it was too special to miss. I was drawn in the first time I saw it. I had never seen overfitting in language-model pretraining before, and I had never seen curves split into such a fork-like shape. My first reaction was: this must be a bug, probably a training-validation mismatch. But it was not a bug. Over-encoding is a good trick, and the effect was real.

It is worth noting how the work came together. The phenomenon was discovered by Danni Yu, and first studied by Shanbin Yu. I joined partway through. At that point, there was still no satisfying explanation for it. I believed I could give a simple model, and I did. That is something I am proud of.

***Follow-up: Which observation was hardest to explain?***

The power law. We spent most of our time explaining the gap-frequency power law. If this were plain finite-sample overfitting, you would expect an exponent of minus one half. But we measured around 0.25 for bigrams and 0.33 for trigrams.

At first I guessed it was caused by the training dynamics. Then I guessed it comes from the natural distribution of language. In the end, it really is the language. This is the Good-Turing distribution law: after an n-gram context has been sampled f times, the probability that the next token is still unseen is proportional to f to the minus alpha. And that alpha matches our 0.25 and 0.33 very well.

Following this key, I realized we should separate the unseen continuations from the seen ones. That led to the crowding-out picture: within a context's own subspace, the seen tokens squeeze the probability of the unseen ones. And from that, you can derive an approximately exponential law for how the gap grows with the number of epochs.

## 2. Could you explain the dog-and-cat example again in simple terms?

Sure. Imagine the phrase "there is a" appears only twice in a small training sample, and both times it is followed by "dog."

For this example, suppose "dog" and "cat" are equally likely in the wider world. Our small sample just happened to contain two dogs and no cats.

The model may have seen "cat" many times elsewhere. But it has never seen "cat" after this particular phrase.

Now we replay the same two examples. Every time, the model is rewarded for predicting "dog." More probability goes to "dog," and less is left for alternatives like "cat."

So when it encounters a new example saying "there is a cat," it can perform worse, even though it has become better at predicting its training examples.

The key point is that repeating the same examples adds no new evidence. The model becomes more confident without becoming better informed.

***Follow-up: How can a familiar word still be a "novel continuation"?***

"Cat" is a familiar word. What is missing is the combination: this particular phrase followed by this particular word.

So "novel" is relative to the context. The model knows the word and has seen the phrase, but the training sample has never shown it that combination.

## 3. Does this happen only with your particular training settings, or more generally?

The phenomenon was robust across the settings we tested. The optimizer and learning rate can change how quickly or how strongly Forking appears, but it was not confined to one particular configuration.

When we combined over-encoding with repeated training on limited data, the effect kept showing up.

## 4. What does this mean in practice, for fine-tuning and reinforcement learning?

My first recommendation is to freeze the over-encoding tables when fine-tuning on a small, heavily repeated dataset. That would also be my starting point for RL.

But freezing is not a complete solution. The context-specific representations are already there, and the rest of the model can keep adapting to them. So some Forking can remain even after the tables stop changing.

Second, in supervised fine-tuning, I would be careful about repeating similar patterns too many times. A dataset can contain many examples while still offering very little variety.

Those are the practical precautions I would start with, while checking whether the model continues to improve on new examples.

## 5. Does this really have an impact on production-scale models?

My honest answer: probably yes, but it is hard to see from the outside.

So far, DeepSeek V4.1 and Qwen3.8 Flash Next have not reported this phenomenon. I suspect two reasons. First, in the phases where the n-gram tables are trainable, they mostly do single-epoch training. Second, as the Engram authors themselves have said, their tables show no sign of having scaled to the boundary; there is still room to scale. And they are indeed scaling: the October release of DeepSeek V4.1 Flash carries a notably larger n-gram table than the September release, going from roughly 150 billion parameters in the table to roughly 250 billion. We are now running experiments to find how far a DeepSeek-scale model's table has to be scaled before this problem appears.

But note that in real pre-training, some specialized data does get repeated. Those are exactly the places where this failure mode can show up.

Even if you never do multi-epoch training, forking still deserves some attention. At the very least, at the end of training, measure whether there is a gap between the loss on tokens the model has already seen and the loss on tokens it has not. Even in one-epoch training, even if you never see a fork in the curves, that does not mean your model's performance is undamaged. Production-scale models also take a small amount of damage: it eats away part of the gains from over-encoding.

Here is why it is easy to miss. If you train for only one epoch, there are actually two training losses. One is the loss on tokens you have already seen; the other is the online training loss, on the fresh tokens in each new batch. That online training loss is strictly equal to the validation loss, which is exactly why, in first-epoch pretraining, we do not need to add a same-distribution validation loss: the two coincide. So the damage does not announce itself as a fork. What you may notice instead is that the loss descends a little more slowly than expected. If we could completely solve forking, it should descend somewhat faster.

And here is a simple check: fix a set of examples the model has already trained on, and look at their loss at the end of training. You will find it has indeed dropped significantly, well below the loss on fresh tokens. That gap between trained and untrained tokens is what to watch for, even when no fork is visible.

And my understanding is that the Engram authors at DeepSeek, along with foundation-model researchers at other large companies, have actually seen this: in production-scale experiments the failure mode does appear, but it is diluted in the overall averages, so a single aggregate curve will not reveal it. They may well have treated it as a bug at first, just like I did when I first saw the curves. In fact, some companies have already reached out to me about collaborating on this problem. But I am no longer inside a company now, so I cannot access IP-sensitive information.

## 6. What is the next experiment you most want to run?

Once I felt we had explained the mechanism, I honestly planned to take a break.

But recently, people working on foundation models came to me and said this problem urgently needs a solution. So I decided to keep going.

My next step is a scaling study: how does the effect change as we increase the model size, the memory capacity, and the amount of data? I want to understand its importance at production scale.

Then comes the harder part: how to prevent Forking while keeping the benefits of over-encoding.

These experiments are running now. I do not currently have enough compute for a large production-scale study, so I will have to be clever about the experimental design.

***Follow-up: How will you know whether a fix really works, without losing the benefits of the memory?***

Under the same training budget, I would want the model with memory to retain an advantage over a comparable model without it.

Then I would check whether that advantage survives further training on smaller datasets. Does the model still perform better on new examples, while the replay gap stays under control?

A smaller gap is encouraging, but the final model also needs to remain useful. That is the comparison I care about.

## 7. You said your ambition is to establish the science of AI. What exactly does that mean?

For me, SAI starts with a standard of evidence.

My name on Xiaohongshu, also known as the Red Book, is actually "Be Critical." That probably tells you something about how I read papers.

I often feel that a paper's conclusions go beyond what its experiments establish. Or the conditions under which a conclusion holds are left unclear. We may have a description of what happened, but still lack an explanation of why it happened or when we should expect it to happen again.

What I want is a clear chain of reasoning between claims. We should be able to say which steps are exact, which involve approximations, and which depend on measurements.

I do not think we have to derive everything from pure mathematics. If we insisted on that, we would only ever be able to compute multi-layer linear networks. And I believe what the physicist Anderson said: more is different. You cannot describe complex phenomena with pure mathematics alone. Instead, at intermediate levels, usable laws emerge that decouple from the microscopic details: the ideal gas law, Newton's laws. You do not need to compute every atom.

We can build what I think of as intermediate first principles: relationships between quantities that describe the system as a whole, which we can isolate, study separately, and connect to make further predictions.

Let me give a simple example.

Suppose we can derive an exact relationship: A equals B plus C. We cannot calculate C analytically, but we can measure it. If C stays small enough, across the conditions we care about, then A is approximately equal to B.

That is a useful conclusion. We have an explicit approximation, a measurable error, and a range of conditions where we have checked that it works.

But there is another requirement. I want to change B in a controlled way and check whether A responds as predicted, while the correction term C remains small.

If I only measure A and B in one particular experiment and find that they happen to match, that is not enough. I want the relationship to help me predict what happens when I intervene. That is where the causal part becomes important.

Then we can use these tested relationships as building blocks. One supports another, and we

can reason about the system while keeping track of our assumptions and approximations.

That is the kind of scientific understanding I would like SAI to establish.

***Follow-up: How do you want to develop this idea over the next few years?***

I am still developing the framework. I have a direction I want to pursue, but I would not present it as a finished theory.

Over the next few years, I want to make these standards more concrete through actual research problems. What kinds of conclusions can our experiments establish? Which assumptions do they depend on? How do we measure the errors in our approximations, and recognize when a relationship stops working?

Forking is one example through which I am exploring these questions.

More broadly, I want to understand what we can reliably know about machine learning, how we can justify that knowledge, and where the limits of that knowledge are.

For me, establishing the science of AI includes being precise about those limits.

## 8. What is your view on innovation in language-model architectures?

With the development of AI, and of RSI, recursive self-improvement, architecture research has markedly accelerated over the past year. Over-encoding, sparse MoE, mHC, attention residuals, DeepSeek's MLA... new ideas keep coming one after another.

I am optimistic about it. A good architectural idea can change how effectively a model uses its computation, its memory, or its training data.

N-gram memory is an interesting example because it gives us a relatively inexpensive way to add capacity. Finding this failure mode does not make that idea less worth exploring. It gives us a clearer problem to solve.

What interests me is how a new component changes the learning process. What becomes easier to learn? How does it affect the sharing of information across examples?

I would also like to see architectural ideas studied across different stages of training. A component may behave differently during broad pretraining and during adaptation to a smaller dataset.

For me, the most exciting outcome is an architectural improvement that also teaches us something about how learning works.

And this connects to something I am working on right now: a science of AI prototype, an idea proposed by Ziming Liu. We have defined a set of standards for SAI, and we use a self-evolving system to run experiments and produce knowledge. This is related to our other project, ArchitectureIQ. And we have already produced some knowledge that humans did not have before.

## 9. Do we need a complete theory of AI before we can improve it?

No, I do not think we need to wait for a complete theory.

Experiments often come first. We try an idea, something surprising happens, and that gives us a question worth explaining. That is a perfectly reasonable way for research to move forward.

But understanding becomes especially valuable when we want to go beyond the experiment where an idea was discovered. We might change the amount of data, train for longer, or adapt the model to a new task. A useful explanation helps us decide what to expect.

Let me give a concrete example, from what I am working on right now. When researchers at large companies try to fix forking, their first ideas are things like tuning the weight decay or the dimensions of the memory table. But the core of the problem is low-frequency n-gram contexts, and that is easy to miss. After hearing this talk, it may sound obvious. It was my initial intuition too, that the bias of low-frequency tokens drives forking; but the full picture turned out to be less simple, and at the time it was not obvious even to my collaborators.

Once we have the picture, targeted fixes follow. One is masking low-frequency contexts at retrieval, which works well. Another is applying appropriate dropout. And once we understand the vectors behind over-encoding, we can go further: merging related contexts, treating it as a VQ-VAE-style optimization problem, so that phrases like "there is a" and "that is a" land in the same slot. That folds related tokens into a shared subspace and reduces the number of rare contexts. I do not have a clean experiment for that last idea yet; it may become the next paper.

I see it as a cycle. Experiments give us phenomena; explanations suggest new experiments; those experiments reveal where the explanation works and where it needs improvement.

So I am comfortable with useful engineering coming before a full theory. I would like the two to keep informing each other.

## 10. Could AI itself help us discover the science of AI?

Yes, and I find that possibility very exciting.

I would like AI to help throughout the research process: proposing hypotheses, implementing experiments, organizing results, and noticing patterns that deserve closer attention.

The part I am especially interested in is choosing the next experiment. Given several possible explanations, can an AI identify a test whose result would meaningfully change what we believe?

That would be a valuable step toward AI doing scientific research.

It also connects to my interest in self-evolving AI. I would like a system to become better at identifying its own weaknesses, forming explanations, and testing improvements.

The standard should still be scientific. Its explanations need to make predictions, and its proposed improvements need to survive careful evaluation. Producing more experimental results is useful; learning how to ask better questions would be even more interesting to me.

# Backup questions

## 11. How do you know when an explanation is convincing?

I ask what the explanation would lead me to expect before I run the next experiment.

It is often possible to tell several plausible stories about the same result. The useful question is whether those stories predict different outcomes when we change something.

Then we can design an experiment that separates them.

I also find it convincing when one fairly simple idea explains several observations that initially seemed unrelated. In Forking, that meant connecting repetition, rare contexts, and the behavior of unseen continuations.

But a good explanation still has boundaries. I would be willing to revise it if we found a well-controlled result that contradicted an important prediction. That would be useful information.

For me, confidence comes from surviving attempts to distinguish the explanation from its alternatives.

## 12. Can researchers with limited compute still do useful AI research?

Yes, especially if they can ask a question that a carefully chosen experiment can answer.

More compute lets you explore more possibilities and test larger systems. But a large run is not automatically informative about why something happened.

Sometimes the most useful experiment is a small one where you keep most things fixed and change the single part that separates two explanations. A clear result there can guide much more expensive work.

That is how I am thinking about the next stage of this project. I want smaller experiments to identify the relationships that matter, then use the larger tests we can afford to check the most important predictions.

Of course, production-scale claims eventually need evidence at an appropriate scale. I would not assume that a small-model result automatically transfers.

Limited compute makes choosing the question and the comparison especially important.

## 13. What makes a research problem worth pursuing for you?

I am drawn to something surprising that I can turn into a clear question.

An unusual result becomes especially interesting when it has some regularity: it repeats, it changes predictably with a condition, or it seems to conflict with how we usually think about the system.

Then I ask whether a simple explanation might connect the observations, and whether there is an experiment that could test it.

Forking had those qualities for me. The curve looked unusual, but there was also structure behind it: the timing of replay, the role of context frequency, and the effect of the memory module.

I also like problems where understanding can eventually inform design. If explaining a failure helps us build a better model, the scientific and practical motivations reinforce each other.

That combination of curiosity, a testable question, and a possible practical consequence is very appealing to me.