AI Link Collection Report - 2025-07-08

Generated on July 08, 2025 at 11:01 AM

Total Items: 1

Report Statistics	
Total Links Processed	1
Generation Date	2025-07-08
Generation Time	11:01:46

■ Link Collection Details

1. The Era of Exploration

■ https://yidingjiang.github.io/blog/post/exploration/

■ Full Article Content:

Large language models are the unintended byproduct of about three decades worth of freely accessible human text online. Ilya Sutskever compared this reservoir of information to fossil fuel, abundant but ultimately finite. Some studies suggest that, at current token■consumption rates, frontier labs could exhaust the highest■quality English web text well before the decade ends. Even if those projections prove overly pessimistic, one fact is clear: today's models consume data far faster than humans can produce it. Large language models are the unintended byproduct of about three decades worth of freely accessible human text online. Ilya Sutskever compared this reservoir of information to fossil fuel, abundant but ultimately finite. Some studies suggest that, at current token■consumption rates, frontier labs could exhaust the highest■quality English web text well before the decade ends. Even if those projections prove overly pessimistic, one fact is clear: today's models consume data far faster than humans can produce it. David Silver and Richard Sutton call this coming phase the "Era of Experience," where meaningful progress will depend on data that learning agents generate for themselves. In this post, I want to build on their statement further: the bottleneck is not having just any experience but collecting the right kind of experience that benefits learning. The next wave of AI progress will hinge less on stacking parameters and more on exploration, the process of acquiring new and informative experience. David Silver and Richard Sutton call this coming phase the " Era of Experience," where meaningful progress will depend on data that learning agents generate for themselves. In this post, I want to build on their statement further: the bottleneck is not having just experience but collecting the kind of experience that benefits learning. The next wave of AI progress will hinge less on stacking parameters and more on exploration, the process of acquiring new and informative experience. To talk about experience collection, we must also ask what it costs to collect them. Scaling is, in the end, a question of resources – compute cycles, synthetic■data generation, data curation pipelines, human oversight, any expenditure that creates learning signal. For simplicity, I'll fold all of these costs into a single bookkeeping unit I call flops. Strictly speaking, a flop is one floating■point operation, but the term has become a lingua franca for "how much effort did this system consume?" I'm co■opting it here not for its engineering precision but because it gives us a common abstract currency. My discussion depends only on relative spend, not on the particular mix of silicon, data, or human time. Treat flops as shorthand for "whatever scarce resource constrains scale.". To talk about experience collection, we must also ask what it costs to collect them. Scaling is, in the end, a question of resources – compute cycles, synthetic■data generation, data curation pipelines, human oversight, any expenditure that creates learning signal. For simplicity, I'll fold all of these costs into a single bookkeeping unit I call. Strictly speaking, a flop is one floating■point operation, but the term has become a lingua franca for "how much effort did this system consume?" I'm co■opting it here not for its engineering precision but because it gives us a common abstract currency. My discussion depends only on relative spend, not on the particular mix of silicon, data, or human time. Treat flops as shorthand for "whatever scarce resource constrains scale." In the sections that follow, I'll lay out a handful of observations and connect ideas that usually appear in different contexts. Exploration is most often used in the context of reinforcement learning (RL), but I will also use "exploration" in a broader sense – much wider than its usual role in RL – because every data-driven system has to decide which experiences to collect before it can learn from them. This usage of exploration is also inspired by my friend Minqi's excellent article "General intelligence requires rethinking exploration.". In the sections that follow, I'll lay out a handful of observations and connect ideas that usually appear in different

contexts. Exploration is most often used in the context of reinforcement learning (RL), but I will also use “exploration” in a broader sense – much wider than its usual role in RL – because every data-driven system has to decide which experiences to collect before it can learn from them. This usage of exploration is also inspired by my friend ’s excellent article “ General intelligence requires rethinking exploration The rest of the post is organized as the following: first, how pre■training inadvertently solved a part of the exploration problem, second, why better exploration translates into better generalization, and finally, where we should spend the next hundred thousand GPU■years. The rest of the post is organized as the following: first, how pre■training inadvertently solved a part of the exploration problem, second, why better exploration translates into better generalization, and finally, where we should spend the next hundred thousand GPU■years. Pretraining is exploration. Pretraining is exploration The standard LLM pipeline is to first pretrain a large model on next-token prediction with a large amount of text and then finetune the model with RL to achieve some desired objectives. Without large-scale pretraining, the RL step would struggle to make any progress. This contrast suggests that pretraining has accomplished something that is difficult for tabula rasa RL (i.e., from scratch). The standard LLM pipeline is to first pretrain a large model on next-token prediction with a large amount of text and then finetune the model with RL to achieve some desired objectives. Without large-scale pretraining, the RL step would struggle to make any progress. This contrast suggests that pretraining has accomplished something that is difficult for tabula rasa RL (i.e., from scratch). A seemingly contradictory and widely observed trend in recent research is that smaller models can demonstrate significantly improved reasoning abilities once distilled using the chain-of-thought generated by larger, more capable models. Some interpret this as evidence that large scale is not a prerequisite for effective reasoning. In my opinion, this conclusion is misguided. The question we should ask is: if model capacity is not the bottleneck for reasoning, why do small models need to distill from a larger model at all? A seemingly contradictory and widely observed trend in recent research is that smaller models can demonstrate significantly improved reasoning abilities once distilled using the chain-of-thought generated by larger, more capable models. Some interpret this as evidence that large scale is not a prerequisite for effective reasoning. In my opinion, this conclusion is misguided. The question we should ask is: if model capacity is not the bottleneck for reasoning, why do small models need to distill from a larger model at all? A compelling explanation for both observations is that the immense cost of pretraining is effectively paying a massive, upfront “exploration tax.” By themselves, models with no pretraining or smaller pretrained models have a much harder time reliably exploring the solution space and discovering good solutions on their own1. The pretraining stage pays this tax by spending vast amounts of compute on diverse data to learn a rich sampling distribution under which the correct continuations are likely. Distillation, in turn, is a mechanism for letting a smaller model inherit that payment, bootstrapping its exploration capabilities from the massive investment made in the larger model. A compelling explanation for both observations is that the immense cost of pretraining is effectively paying a massive, upfront “ exploration tax.” By themselves, models with no pretraining or smaller pretrained models have a much harder time reliably exploring the solution space and discovering good solutions on their own. The pretraining stage pays this tax by spending vast amounts of compute on diverse data to learn a rich sampling distribution under which the correct continuations are likely. Distillation, in turn, is a mechanism for letting a smaller model inherit that payment, bootstrapping its exploration capabilities from the massive investment made in the larger model. Why is this pre-paid exploration so important? In its most general form, the RL loop looks something like this:. Why is this pre-paid exploration so important? In its most general form, the RL loop looks something like this: • Exploration. The agent generates some randomized exploration trajectories. Exploration. The agent generates some randomized exploration trajectories. • Reinforce. The good trajectories are up-weighted and the bad ones are down-weighted. Reinforce. The good trajectories are up-weighted and the bad ones are down-weighted. For this learning loop to be effective, the agent must be capable of generating at least a minimal number of “good” trajectories during its exploration phase. This concept is sometimes called coverage in RL. In LLMs, this exploration is typically achieved through sampling from the model’s autoregressive output distribution. Under this exploration scheme, the correct solutions need to already be likely in the naive sampling distribution. If a lower■capacity model rarely stumbles on a valid solution by random sampling, it would have nothing useful to reinforce. For this learning loop to be effective, the agent must be

capable of generating at least a minimal number of “good” trajectories during its exploration phase. This concept is sometimes called coverage in RL. In LLMs, this exploration is typically achieved through sampling from the model’s autoregressive output distribution. Under this exploration scheme, the correct solutions need to already be likely in the naive sampling distribution. If a lower■capacity model rarely stumbles on a valid solution by random sampling, it would have nothing useful to reinforce. Exploration without any prior information is a very difficult process. Even in the simplest tabular RL setting, where every situation (a state) and every action can be listed out in a table, theory says learning needs a lot of trials. A well known lower-bound on the sample complexity on the number of episodes for tabular RL is $\Omega(\frac{SAH^2}{\epsilon^2})$ (Dann & Brunskill, 2015), where S is the size of the state space, A is the size of the action space, H is the horizon, and ϵ is the “distance” to the best possible solution. This means that the minimum number of episodes grows linearly with the number of state-action pairs, and quadratically with the horizon. For LLMs, the state space now includes every possible text prefix, and the action space is any next token, both of which are very large. Without any prior information, RL in this setting would be practically impossible. Exploration without any prior information is a very difficult process. Even in the simplest tabular RL setting, where every situation (a state) and every action can be listed out in a table, theory says learning needs a lot of trials. A well known lower-bound on the sample complexity on the number of episodes for tabular RL is $\Omega(\frac{SAH^2}{\epsilon^2})$ (Dann & Brunskill, 2015), where S is the size of the state space, A is the size of the action space, H is the horizon, and ϵ is the “distance” to the best possible solution. This means that the minimum number of episodes grows linearly with the number of state-action pairs, and quadratically with the horizon. For LLMs, the state space now includes every possible text prefix, and the action space is any next token, both of which are very large. Without any prior information, RL in this setting would be practically impossible. Up to now, the hard work of exploration has been largely done by pretraining and learning a better prior from which to sample trajectories. However, this also means that the types of trajectories the model can sample naively are heavily constrained by the prior. To progress further, we must figure out how to move beyond the prior. Up to now, the hard work of exploration has been largely done by pretraining and learning a better prior from which to sample trajectories. However, this also means that the types of trajectories the model can sample naively are heavily constrained by the prior. To progress further, we must figure out how to move beyond the prior. Exploration helps generalization. Exploration helps generalization Historically, RL research has focused on solving a single environment at a time, like Atari or MuJoCo. This is equivalent to training and testing on the same datapoint. However, how well a given model does in the single environment does not say much about how well the model can handle truly novel situations. Machine learning is ultimately about generalization: for many hard problems, if we know them in advance, we can engineer a bespoke solution. What matters is succeeding on problems we haven’t seen or even anticipated. Historically, RL research has focused on solving a single environment at a time, like Atari or MuJoCo. This is equivalent to training and testing on the same datapoint. However, how well a given model does in the single environment does not say much about how well the model can handle truly novel situations. Machine learning is ultimately about generalization: for many hard problems, if we know them in advance, we can engineer a bespoke solution. What matters is succeeding on problems we haven’t seen or even anticipated. The generalization performance of RL is critical for language models. During training, an LLM sees only a finite set of prompts, yet at deployment it must handle arbitrary user queries that could be very different from the ones seen during training. Notably, current LLMs excel at tasks with a verifiable reward (e.g., coding puzzles or formal proofs) because the correctness can be easily checked. The tougher question is to generalize these capabilities to fuzzier domains (e.g., generate a research report or write a novel) where feedback is sparse or ambiguous, and large-scale training and data collection are difficult. The generalization performance of RL is critical for language models. During training, an LLM sees only a finite set of prompts, yet at deployment it must handle arbitrary user queries that could be very different from the ones seen during training. Notably, current LLMs excel at tasks with a verifiable reward (e.g., coding puzzles or formal proofs) because the correctness can be easily checked. The tougher question is to generalize these capabilities to fuzzier domains (e.g., generate a research report or write a novel) where feedback is sparse or ambiguous, and large-scale training and data collection are difficult. What are some options we have

for training a generalizable model? A recurring theme of deep learning is that data diversity drives robust generalization. Exploration directly controls the diversity of the data. In supervised learning, a labeled example reveals all its details in a single forward pass2 so the only way to increase data diversity is to collect more data. In RL, by contrast, each interaction only exposes a narrow slice of the environment. As a result, the agent must gather a sufficiently varied set of trajectories to build a representative picture. If the trajectories collected lack diversity (e.g., naive random sampling), the policy can overfit to that narrow slice and stumble even within the very same environment. What are some options we have for training a generalizable model? A recurring theme of deep learning is that data diversity drives robust generalization. Exploration directly controls the diversity of the data. In supervised learning, a labeled example reveals all its details in a single forward pass so the only way to increase data diversity is to collect more data. In RL, by contrast, each interaction only exposes a narrow slice of the environment. As a result, the agent must gather a sufficiently varied set of trajectories to build a representative picture. If the trajectories collected lack diversity (e.g., naive random sampling), the policy can overfit to that narrow slice and stumble even within the very same environment. This problem compounds when there are multiple environments. A popular RL generalization benchmark is Procgen, which is a collection of Atari-like games that have procedurally generated environments, so each game in principle contains “infinitely” many environments. The objective is to train on a fixed number of environments for a fixed number of steps and generalize to completely unseen environments3. This problem compounds when there are multiple environments. A popular RL generalization benchmark is Procgen, which is a collection of Atari-like games that have procedurally generated environments, so each game in principle contains “infinitely” many environments. The objective is to train on a fixed number of environments for a fixed number of steps and generalize to completely unseen environments Many existing approaches for this benchmark treat the problem as a representation learning problem and apply regularization techniques adapted from supervised learning (e.g., dropout or data augmentation). These help, but they overlook exploration, one of the most important structural components of RL. Since the agents collect their own data, they can improve generalization by changing exploration. In a previous work, my coauthors and I showed that pairing an existing RL algorithm with a stronger exploration strategy can double its generalization performance on Procgen without explicit regularization. In a more recent work, we found that better exploration also lets the model leverage more expressive model architectures and computational resources, and generalize better on Procgen as the result4. Many existing approaches for this benchmark treat the problem as a representation learning problem and apply regularization techniques adapted from supervised learning (e.g., dropout or data augmentation). These help, but they overlook exploration, one of the most important structural components of RL. Since the agents collect their own data, they can improve generalization by changing exploration. In a previous work, my coauthors and I showed that pairing an existing RL algorithm with a stronger exploration strategy can double its generalization performance on Procgen without explicit regularization. In a more recent work, we found that better exploration also lets the model leverage more expressive model architectures and computational resources, and generalize better on Procgen as the result While Procgen is certaintly not as difficult and complex as the problems LLMs are trained to solve today, the overall problem structure is essentially the same – the RL agent is trained on a finite set of problems and tested on new problems at test time without further training. The way we do exploration with LLMs today is fairly simple, typically limited to sampling from the model’s autoregressive distribution with tweaks to temperature or entropy bonus, so there is a large design space for potentially better exploration approaches. Admittedly, there have not been many successful examples in this direction. This could be because it is a very hard problem, it is not flop-efficient enough to be practical, or we just haven’t tried hard enough. However, if Procgen-style exploration gains do translate, we’re leaving efficiency – and perhaps entirely new capabilities – on the table. The next section discusses where we might look first. While Procgen is certaintly not as difficult and complex as the problems LLMs are trained to solve today, the overall problem structure is essentially the same – the RL agent is trained on a finite set of problems and tested on new problems at test time without further training. The way we do exploration with LLMs today is fairly simple, typically limited to sampling from the model’s autoregressive distribution with tweaks to temperature or entropy bonus, so there is a large design space for potentially better

exploration approaches. Admittedly, there have not been many successful examples in this direction. This could be because it is a very hard problem, it is not flop-efficient enough to be practical, or we just haven't tried hard enough. However, if Procgen-style exploration gains do translate, we're leaving efficiency – and perhaps entirely new capabilities – on the table. The next section discusses where we might look first. Two axes of scaling exploration. Two axes of scaling exploration Exploration, in the broad sense I'm using here, is deciding what data the learner will see. That decision happens on two distinct axes: Exploration, in the broad sense I'm using here, is deciding what data the learner will see. That decision happens on two distinct axes: • World sampling – deciding where to learn. World here refers to a particular problem that needs to be solved. In supervised learning (or unsupervised pretraining), this axis covers data collection, synthetic generation and curation: gathering and filtering raw documents, images, or code, each of which corresponds to a "world". In RL, this corresponds to designing or generating environments, such as a single math puzzle or a coding problem. We can even arrange the worlds into curricula. In both cases, world sampling is fundamentally about what "data points" the learner is allowed to see. This also decides the limit on all the information any agent can possibly learn. World sampling – deciding where to learn. World here refers to a particular problem that needs to be solved. In supervised learning (or unsupervised pretraining), this axis covers data collection, synthetic generation and curation: gathering and filtering raw documents, images, or code, each of which corresponds to a "world". In RL, this corresponds to designing or generating environments, such as a single math puzzle or a coding problem. We can even arrange the worlds into curricula. In both cases, world sampling is fundamentally about what "data points" the learner is allowed to see. This also decides the limit on all the information any agent can possibly learn. • Path sampling – deciding how to gather data inside a world. This step is unique to RL. Once a world is chosen, the agent still has to pick which trajectories to collect: random walks, curiosity■driven policies, tree search, tool-use, etc. Different path■sampling strategies can have different computation cost and produce very different training distributions even when the underlying world is identical. In short, path sampling is about what the learner "wants" to see. Path sampling – deciding how to gather data inside a world. This step is unique to RL. Once a world is chosen, the agent still has to pick which trajectories to collect: random walks, curiosity■driven policies, tree search, tool-use, etc. Different path■sampling strategies can have different computation cost and produce very different training distributions even when the underlying world is identical. In short, path sampling is about what the learner "wants" to see. In supervised learning or unsupervised pretraining, the second axis incurs a constant cost because a single forward (and backward) pass has access to all the information each data point contains (e.g., cross-entropy loss). Because there is no obvious way to "dig deeper" inside a single example (other than make the models larger), the exploration cost lives almost entirely on the first axis – world sampling. The flops can either go into acquire new worlds (e.g., new data points) or processing existing worlds (e.g., curation and synthetic data). In supervised learning or unsupervised pretraining, the second axis incurs a constant cost because a single forward (and backward) pass has access to all the information each data point contains (e.g., cross-entropy loss). Because there is no obvious way to "dig deeper" inside a single example (other than make the models larger), the exploration cost lives almost entirely on the first axis – world sampling. The flops can either go into acquire new worlds (e.g., new data points) or processing existing worlds (e.g., curation and synthetic data). In contrast, RL has much more flexibility in the second axis (in addition to the first axis). Because most random trajectories reveal little information about the ideal behavior, the information density (useful bits per flop) in RL is far lower than in supervised learning or pretraining. If we naïvely sample trajectory, we risk wasting flops on noise. So it's imporant to be judicious about how we spend our flops5. There are also more options for spending flops to explore within each world. For example, we can either sample more trajectories from a single environment or we can spend more flops thinking about how to sample the next trajectory to discover high-value states and actions. In contrast, RL has much more flexibility in the second axis (in addition to the first axis). Because most random trajectories reveal little information about the ideal behavior, the information density (useful bits per flop) in RL is far lower than in supervised learning or pretraining. If we naïvely sample trajectory, we risk wasting flops on noise. So it's imporant to be judicious about how we spend our flops. There are also more options for spending flops to explore within each world. For example, we can either sample more trajectories from a single environment or

we can spend more flops thinking about how to sample the next trajectory to discover high-value states and actions. For most, if not all, machine learning problems, the high-level goal can be understood as maximizing information per flop. For that purpose, these two levers form a trade-off curve. If one spends too much resources on world sampling and not enough on path sampling, the agent may not extract any meaningful experience from the sampled worlds. Vice versa, if one spends too much resources on a small set of worlds, the agent could overfit to the training worlds and would not learn generalizable behavior that transfer across worlds. The ideal scenario happens somewhere in between where the resources are divided between sampling new worlds and running algorithms (i.e., better than random sampling) that can extract more information from a single world. For most, if not all, machine learning problems, the high-level goal can be understood as maximizing information per flop. For that purpose, these two levers form a trade-off curve. If one spends too much resources on world sampling and not enough on path sampling, the agent may not extract any meaningful experience from the sampled worlds. Vice versa, if one spends too much resources on a small set of worlds, the agent could overfit to the training worlds and would not learn generalizable behavior that transfer across worlds. The ideal scenario happens somewhere in between where the resources are divided between sampling new worlds and running algorithms (i.e., better than random sampling) that can extract more information from a single world. If you are familiar with scaling laws, what I just described sounds a lot like the Chinchilla scaling laws but the two axes correspond to the compute used for different types of sampling rather than parameters and data. At each performance level, one should to be able to trace out an isoperformance curve where the x-axis and y-axis are the compute put into interacting with any given environment and the compute given to the environments, whether it is for generating the environment or for running the environment (e.g., a generative verifier with CoT). If you are familiar with scaling laws, what I just described sounds a lot like the Chinchilla scaling laws but the two axes correspond to the compute used for different types of sampling rather than parameters and data. At each performance level, one should to be able to trace out an isoperformance curve where the x-axis and y-axis are the compute put into interacting with any given environment and the compute given to the environments, whether it is for generating the environment or for running the environment (e.g., a generative verifier with CoT). Of the two axes, path sampling is a relatively well-defined problem. A principled approach for doing exploration within an environment is to reduce the model's uncertainty6. Many existing approaches for exploration have very strong sample complexity but they tend to be prohibitively expensive. Nonetheless, there is arguably a well-defined objective for path sampling and the main obstacle is to figure out a computationally efficient approximation. On the other hand, it is much less clear what the objective is for world sampling. One appealing idea is open-ended learning but even open-ended learning requires defining the universe of all environments (i.e., environment specs) or a subjective observer that judges whether an outcome is "interesting". Of the two axes, path sampling is a relatively well-defined problem. A principled approach for doing exploration within an environment is to reduce the model's uncertainty. Many existing approaches for exploration have very strong sample complexity but they tend to be prohibitively expensive. Nonetheless, there is arguably a well-defined objective for path sampling and the main obstacle is to figure out a computationally efficient approximation. On the other hand, it is much less clear what the objective is for world sampling. One appealing idea is open-ended learning but even open-ended learning requires defining the universe of all environments (i.e., environment specs) or a subjective observer that judges whether an outcome is "interesting". What objective should world sampling optimize? The unfortunate reality (or fortunate, depending on your perspective) is that the space of environments is infinite, but our resources are finite. If we want to do something useful, then we must express some preference over the environments. I suspect that the problem of designing environments will eventually become similar to selecting pretraining data. It will be hard to say exactly why one environment will help another environment and we will need a lot of them. In other words, there may not be a single clean and nice objective for designing environment specs. What objective should world sampling optimize? The unfortunate reality (or fortunate, depending on your perspective) is that the space of environments is infinite, but our resources are finite. If we want to do something useful, then we must express some preference over the environments. I suspect that the problem of designing environments will eventually become similar to selecting pretraining data. It will be hard to say exactly why one

environment will help another environment and we will need a lot of them. In other words, there may not be a single clean and nice objective for designing environment specs. The more likely scenario (probably already happening) is that everyone will start designing specs within their own expertise or domain of interest. When we have enough “human-approved” and “useful” specs, maybe we can try to learn some common principles and eventually automate the process by learning from them, much like how pretraining data is selected today. It would be inconvenient if we need the same amount of environments as pretraining data to achieve the same level of generality for decision making, but there are some preliminary evidence that this may not be the case. In a recent work, we found that a fairly small number of environments is enough to train an agent capable of general exploration and decision making in entirely out-of-distribution environments. Furthermore, using existing LLMs could also greatly accelerate the design process. The more likely scenario (probably already happening) is that everyone will start designing specs within their own expertise or domain of interest. When we have enough “human-approved” and “useful” specs, maybe we can try to learn some common principles and eventually automate the process by learning from them, much like how pretraining data is selected today. It would be inconvenient if we need the same amount of environments as pretraining data to achieve the same level of generality for decision making, but there are some preliminary evidence that this may not be the case. In a recent work, we found that a fairly small number of environments is enough to train an agent capable of general exploration and decision making in entirely out-of-distribution environments. Furthermore, using existing LLMs could also greatly accelerate the design process. Of course, these are all very high-level musings and how one exactly scales these two axes is much less obvious than scaling pretraining. However, if we were able to figure out a reliable way to introduce scale into world sampling, and a more intelligent way of path sampling, we should be able to see isoperformance curves that bends inwards towards the origin (maybe they won’t be as smooth). This style of scaling laws would teach us the best way to allocate computation resource between the environments and the agent. Of course, these are all very high-level musings and how one exactly scales these two axes is much less obvious than scaling pretraining. However, if we were able to figure out a reliable way to introduce scale into world sampling, and a more intelligent way of path sampling, we should be able to see isoperformance curves that bends inwards towards the origin (maybe they won’t be as smooth). This style of scaling laws would teach us the best way to allocate computation resource between the environments and the agent. Final thoughts. Final thoughts I could keep unfolding more tangents – better curiosity objectives, open-endedness, meta■exploration that learns how to explore – but I think it is more important to make the high-level point clear. I could keep unfolding more tangents – better curiosity objectives open-endedness meta■exploration that learns how to explore – but I think it is more important to make the high-level point clear. Existing paradigms of scaling have been incredibly effective but all paradigms will eventually saturate. The question is where to pour the next orders of magnitude of compute. I’ve argued that exploration – both world sampling and path sampling – offers a promising direction. We don’t yet know the right scaling laws, the right environment generators, or the right exploration objectives, but intuitively they should be possible. The coming years will decide whether exploration can stretch our flops further on top of the existing paradigms. I think the bet is worth making. Existing paradigms of scaling have been incredibly effective but all paradigms will eventually saturate. The question is where to pour the next orders of magnitude of compute. I’ve argued that exploration – both world sampling and path sampling – offers a promising direction. We don’t yet know the right scaling laws, the right environment generators, or the right exploration objectives, but intuitively they should be possible. The coming years will decide whether exploration can stretch our flops further on top of the existing paradigms. I think the bet is worth making. Acknowledgement. Acknowledgement Many thanks to Allan Zhou, Sam Sokota, Minqi Jiang, Ellie Haber, Alex Robey, Swaminathan Gurumurthy, Kevin Li, Calvin Luo, Abitha Thankaraj, and Zico Kolter for their feedback and discussion on the draft. Many thanks to Allan Zhou, Sam Sokota, Minqi Jiang, Ellie Haber, Alex Robey, Swaminathan Gurumurthy, Kevin Li, Calvin Luo, Abitha Thankaraj, and Zico Kolter for their feedback and discussion on the draft. • A valid alternative possibility is that the RL optimization objective does not work well with smaller models, but this is likely not the case because before LLMs most successful applications of RL involved very small models. ■ A valid alternative possibility is that the RL optimization objective does not work well with smaller models, but this is

likely not the case because before LLMs most successful applications of RL involved very small models. ■. A valid alternative possibility is that the RL optimization objective does not work well with smaller models, but this is likely not the case because before LLMs most successful applications of RL involved very small models. • This doesn't mean the model can full exploit this information because the model can be limited in its computation power. It just means that if it wants to, that information is fully available. ■ This doesn't mean the model can full exploit this information because the model can be limited in its computation power. It just means that if it wants to, that information is fully available. ■. This doesn't mean the model can full exploit this information because the model can be limited in its computation power. It just means that if it wants to, that information is fully available. • For generalization to be tractable, we must assume that a "good enough" policy exists for all environments. This is analogous to assuming there is small or no label noise in supervised learning. ■ For generalization to be tractable, we must assume that a "good enough" policy exists for all environments. This is analogous to assuming there is small or no label noise in supervised learning. ■. For generalization to be tractable, we must assume that a "good enough" policy exists for all environments. This is analogous to assuming there is small or no label noise in supervised learning. • At the time of writing, I believe this sets a new state-of-the-art performance on the "25M easy" benchmark of ProcGen. ■ At the time of writing, I believe this sets a new state-of-the-art performance on the "25M easy" benchmark of ProcGen. ■. At the time of writing, I believe this sets a new state-of-the-art performance on the "25M easy" benchmark of ProcGen. • Interestingly, for many problems such as Atari, random sampling works reasonably well. I think that says much more about the environments than the exploration method itself. ■ Interestingly, for many problems such as Atari, random sampling works reasonably well. I think that says much more about the environments than the exploration method itself. ■. Interestingly, for many problems such as Atari, random sampling works reasonably well. I think that says much more about the environments than the exploration method itself. • There is a wide family of RL algorithms under the names of posterior sampling or information-directed sampling that try to direct the exploration to reduce the model's uncertainty, but they are generally too expensive to be done exactly at the scale of LLMs. Various approximations exist but they have not been widely used for LLMs to the best of my knowledge. ■ There is a wide family of RL algorithms under the names of posterior sampling or information-directed sampling that try to direct the exploration to reduce the model's uncertainty, but they are generally too expensive to be done exactly at the scale of LLMs. Various approximations exist but they have not been widely used for LLMs to the best of my knowledge. ■. There is a wide family of RL algorithms under the names of posterior sampling information-directed sampling that try to direct the exploration to reduce the model's uncertainty, but they are generally too expensive to be done exactly at the scale of LLMs. Various approximations exist but they have not been widely used for LLMs to the best of my knowledge.

■ **Shared by:** Sai
■ **Shared on:** Jul 07, 2025 at 11:21 AM
■ **Domain:** yidingjiang.github.io
■ **Length:** 6,325 words
■ **Processed:** Jul 08, 2025 at 11:01 AM