06 Can an AI Learn Inside Its Own World Model?

BY NILUSHANAN KULASINGHAM

13 MIN READ · INTERACTIVE

> A month of robot time becomes a day if the practice happens inside the model. That has been the pitch since Dyna, and Dreamer made it work. What the exchange rate costs, and why the fix for an agent that exploits its own dream is to make the dream worse on purpose.

The figures in this chapter are interactive. Press and drag them online at worldmodels101.com/chapters/dreaming.

A robot arm learning to pick something up will drop it a couple of million times first. Every drop costs something real: wall-clock seconds, and wear on the joints. Someone nearby has to reset the scene when the arm knocks the bin over. At one attempt a second, that is a month of doing nothing else.

The idea is simple enough to say in a sentence. Spend part of that month collecting experience, fit a model to it, and let the arm practise inside the model, where a drop costs only compute. Richard Sutton wrote that loop down in 1991 and called the agent Dyna. Sutton is a founder of reinforcement learning, where an agent learns by acting and being rewarded, and he co-wrote its standard textbook.

Dyna was tiny, a loop in a grid maze, and about thirty years early. Danijar Hafner's Dreamer, which runs the same loop with deep networks, arrived in 2019. The question in the title is whether the loop works, and the short answer is yes. The longer answer is about what it costs.

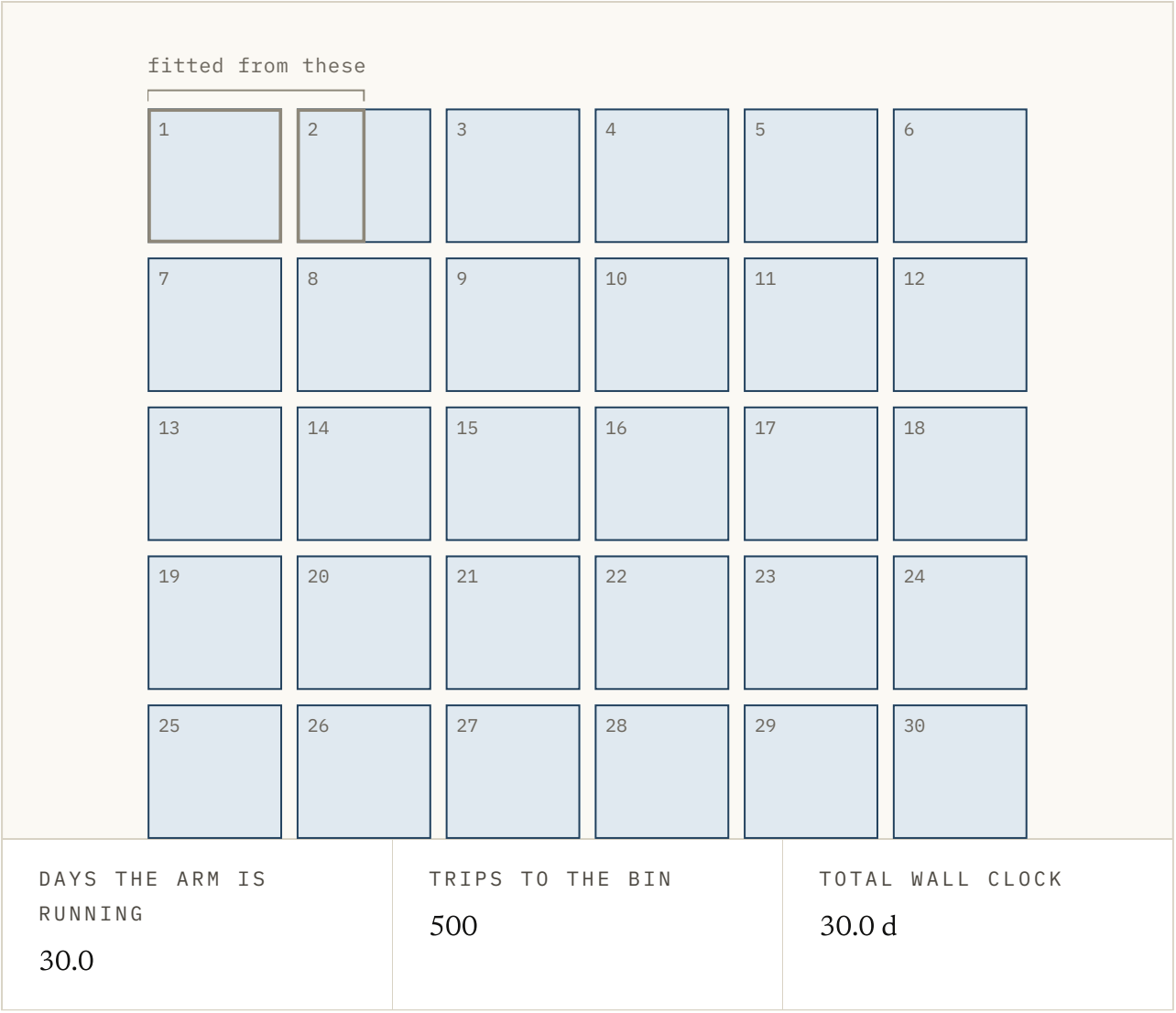


FIG. 6.1 The month from the first paragraph, drawn as thirty days of the arm dropping things. Slide the practice into the model and watch the days empty from the end backwards. Two things do not move: the day and a half at the start, because the model has to be fitted to something, and the fact that somebody makes every one of those trips to the bin. Nothing in here prices what an imagined step is worth, which is the next question. The costs are illustrative.

Push most of the practice inside and you should see the month come down to about a day and a half. That has been the sales pitch from Dyna to Dreamer, and the saving is real. It holds only while an imagined step stays useful enough to count.

The short answer is yes, and Dreamer is the evidence

Learning inside a model is the biggest practical argument for building one, and Dreamer is the evidence. Danijar Hafner built it as a Toronto PhD student working inside Google Brain, a year after his PlaNet had planned by search inside a learned model. Dreamer dropped the search and trained a policy (the part that picks the action. It is what is being trained here; the model is not) instead.

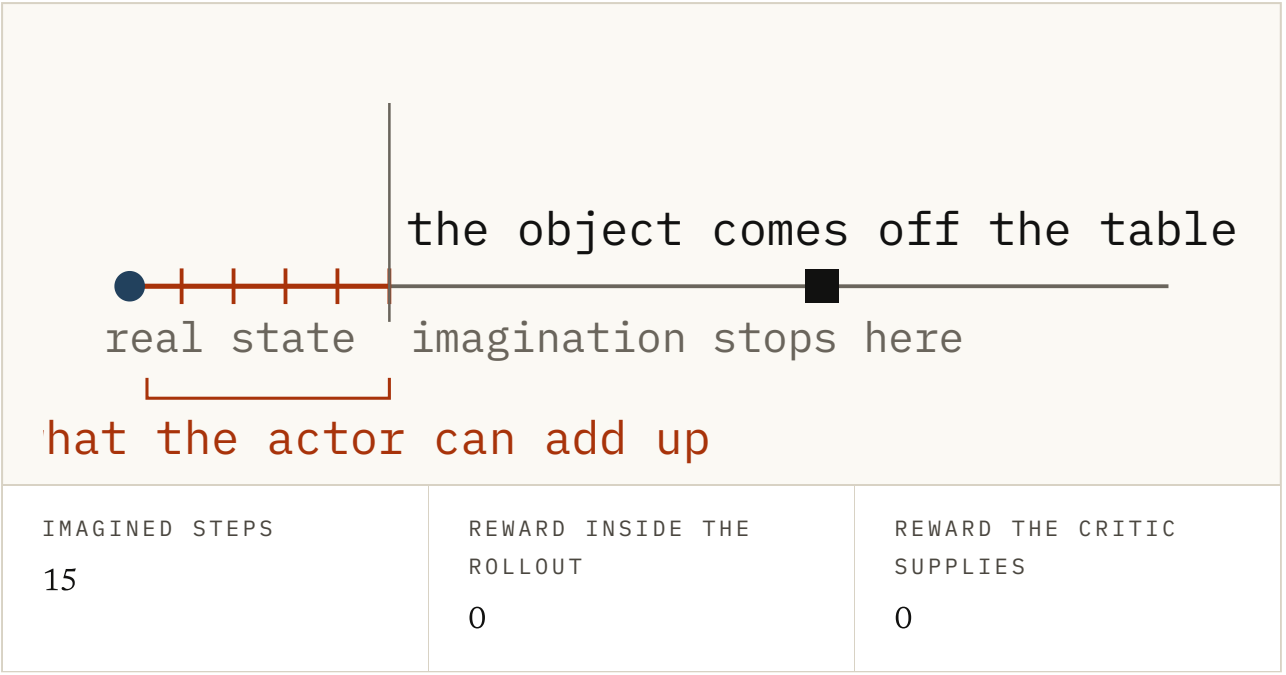


FIG. 6.2 The actor learns from one imagined rollout, and the rollout stops long before the arm gets the object off the table. Drag the reward further out and watch it leave the imagined stretch, so there is nothing left for the actor to add up. Then switch the critic on: it hands the actor one number standing for everything after the last imagined step. That number is a guess, and no part of it came from the world. The lengths are illustrative.

Dream to Control, the 2019 paper, learned its behaviour from long imagined rollouts. Neither the actor nor the critic ever touched the environment while it learned. The actor is the policy under another name. The critic is a second network that guesses how much reward lies ahead.

DreamerV2 followed in 2020 as *Mastering Atari with Discrete World Models*. The discrete part is a state made of multiple-choice answers rather than dials. On the Atari games it started beating agents that learn straight from the environment.

DreamerV3 arrived in *Nature* in 2025 as *Mastering diverse control tasks through world models*. It ran with one set of settings across more than 150 tasks. It was the first agent to collect diamonds in Minecraft without human data, which takes a long chain of steps in order. That is the strongest answer I know of to whether learning in imagination generalises.

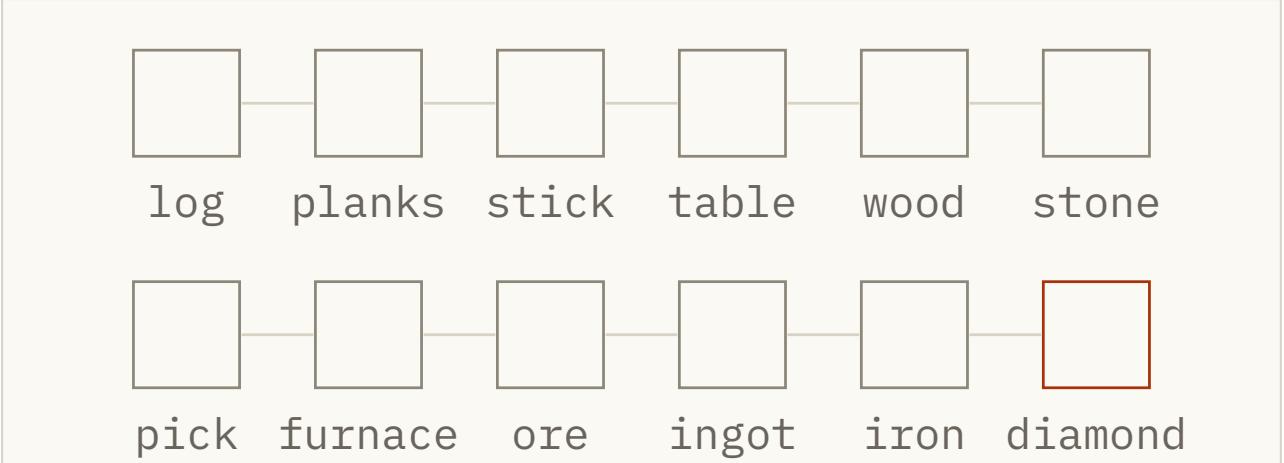


FIG. 6.3 The diamond is twelve steps in order, and DreamerV3 got there without being shown how. Drag the chance of getting one link right, then press Try a run and watch where the chain breaks. Nine times in ten per link still fails about seven runs in ten, which is why finishing the chain is worth more as evidence than any single task. The per link odds are illustrative; the twelve links are the game's.

So systems like these get competent on a fraction of the world contact that learning directly would need. On a real robot that fraction decides whether learning is possible at all. The month of dropped objects is what the dynasty was built to shorten. Now let's look at what the loop is doing, because that is where the cost hides.

The loop has to go round twice

Dyna's loop has four steps, and it repeats. Act in the world for a while and keep what happened. Fit a model to it. Train the policy inside that model, which costs compute rather than robot contact, and then take the behaviour back out.

The behaviour that comes back out collects better experience than the last lap's did. That makes the next model better. The second lap is the part to watch in the figure below.

> The loop going round again is the part the summaries drop. A model fitted to the flailing of an untrained agent is only good where an untrained agent goes. It only improves once a better policy goes somewhere new.

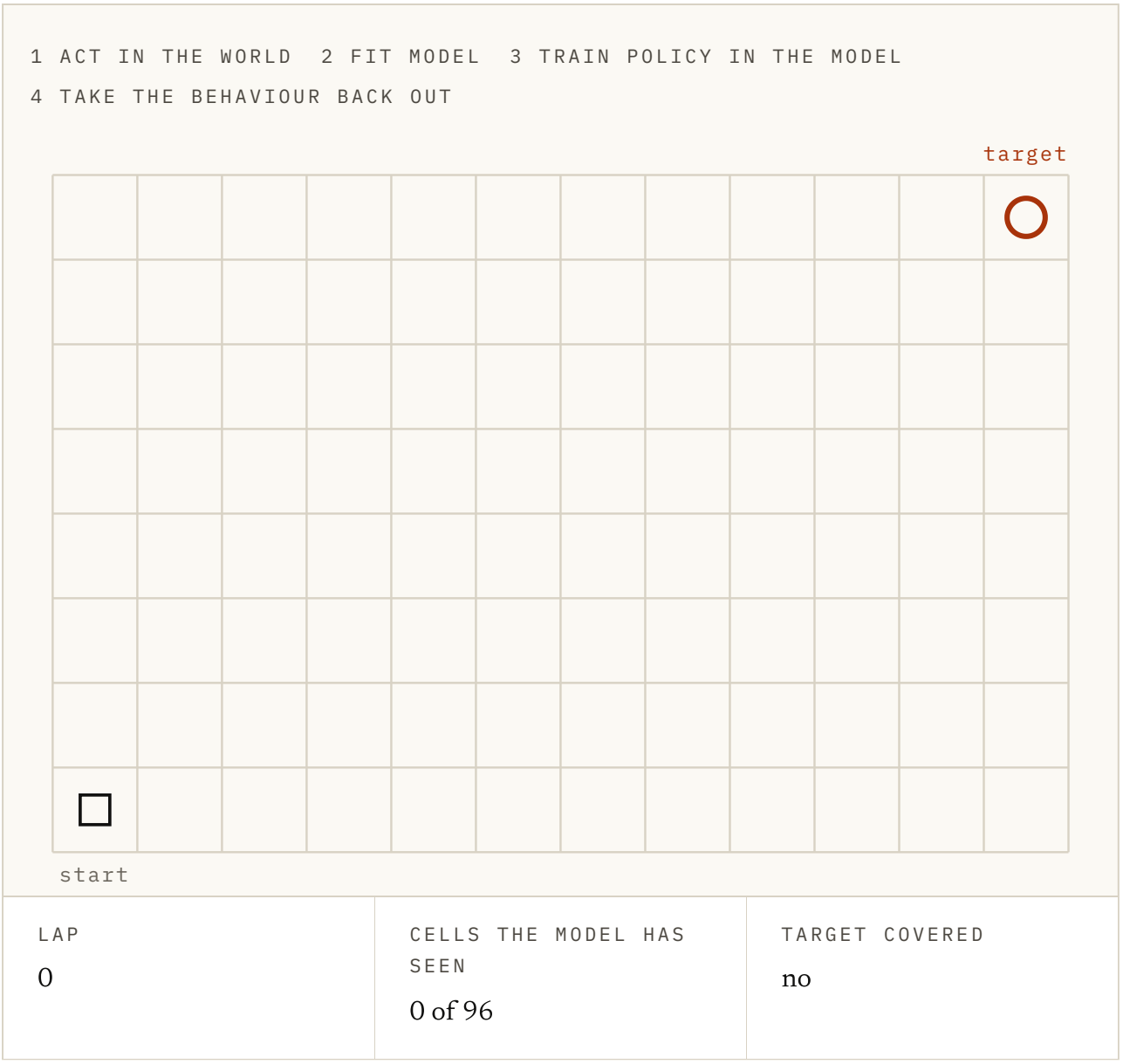


FIG. 6.4 A small grid world with a target in the far corner. Press Run a lap. The first policy wanders at random, so the model only learns the cells near the start, which are the shaded ones, and a policy trained inside it has no idea where the target is. Press it again. The better policy goes further, the shaded region grows, and after a lap or two the model finally covers the target. Then press Reset and try turning off Improve the policy: the shaded region stops growing, lap after lap. Illustrative throughout.

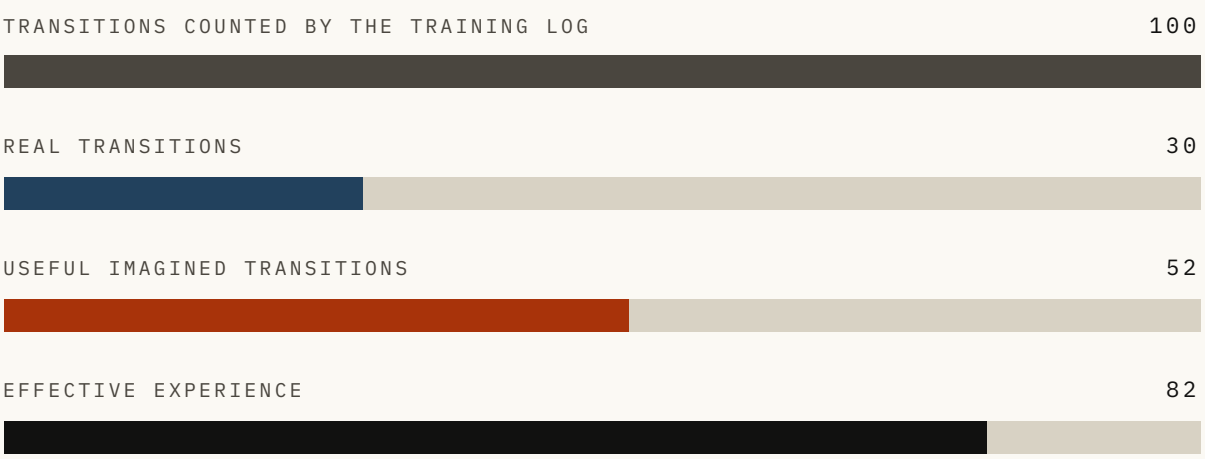
The first model is fitted to whatever a useless policy bumped into. It knows the floor and nothing about the target. The second lap fixes that, because the better policy goes to new places and brings back the data the first model was missing.

That is why Figure 6.1 keeps a day and a half locked. Imagined steps are worth having only while the model can supply them accurately, and it can only do that where it has been. A network for a robot's world has to guess about the rest.

Imagined experience is a different currency

A training log counts one real transition and one model-generated transition as two rows. A transition is one step of experience: an action and what it led to. The learner cannot count them the same.

Imagined experience carries the model's blind spots. The discount compounds with every step a rollout, a chain of imagined transitions, takes from a real state. I will call this the exchange rate.



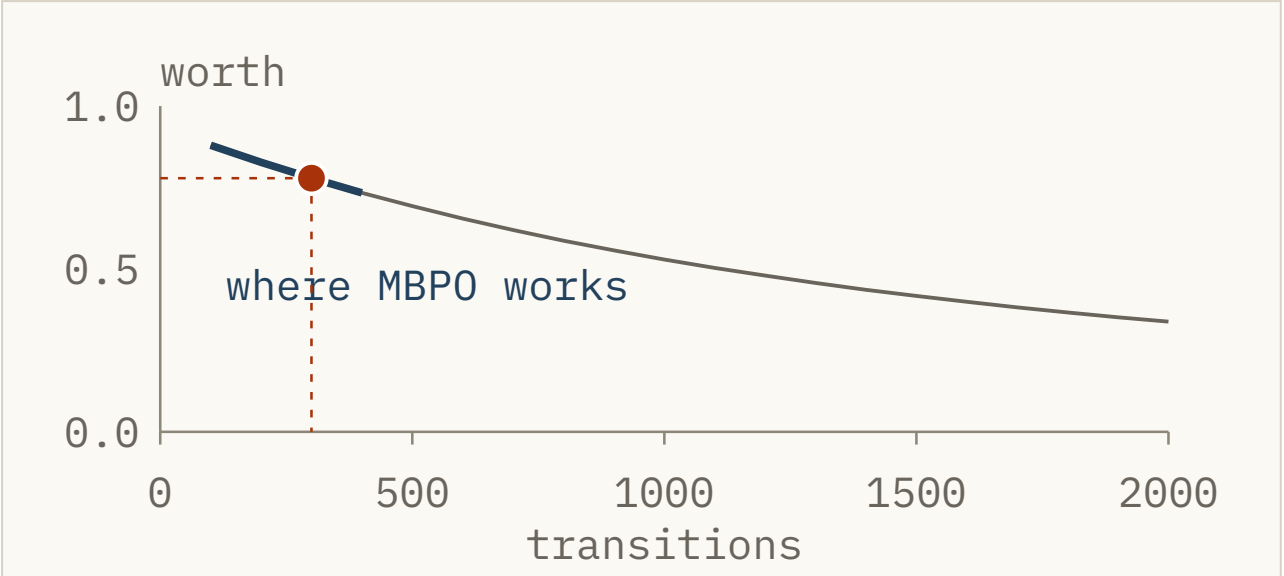
The model is accurate enough here that most of the synthetic batch still carries signal.

SYNTHETIC RELIABILITY	WORLD-CONTACT COST	MODEL COMPUTE COST
75%	30	5.6

FIG. 6.5 Push up the imagined share, then push up the model's per-step error. On paper the batch still holds 100 transitions. Watch the effective experience bar, which is what the learner is really getting, fall as you do. The discount is for illustration rather than an estimate. What it exposes is the assumption that generated and observed rows trade at par.

Michael Janner and colleagues priced it in 2019. Their paper, *When to Trust Your Model*, introduced MBPO, which trains a policy on imagined rollouts only a few steps long. Each starts from a real state drawn from the replay buffer, the store of real experience kept from acting.

That trades volume for trust. More imagination helps only while each step is priced by how far it has travelled from evidence.



TRANSITIONS IN THE BATCH	AVERAGE WORTH OF A ROW	EFFECTIVE TRANSITIONS
300	0.78	234

FIG. 6.6 One hundred real states from the replay buffer, and one choice about how far to imagine from each. Drag the rollout length and watch the dot travel down the curve: the batch gets bigger, and the share of it sitting near evidence gets smaller. MBPO works in the corner at the top left, a step or two out from every real state. Watch the third cell as you go, because that is the only one that counts. The trust figures are illustrative; the trade is the paper's.

The fireball policy

A policy trained inside a model is scored by that model. The model is all it can see, so it cannot tell a good action from one that only looks good to its examiner. It will find the second kind, because those are the cheapest points on offer.

David Ha and Jürgen Schmidhuber met this head-on in 2018, in *World Models*. Schmidhuber had argued for learned world models since 1990 and co-invented the LSTM, the memory network behind a decade of speech recognition. Ha, then at Google Brain, published it as an interactive web page.

Their agent was a tiny controller trained entirely inside a dream of a Doom level where the player dodges fireballs. The controller found a way of moving that stopped the dream producing fireballs at all. Let's see what that looks like.

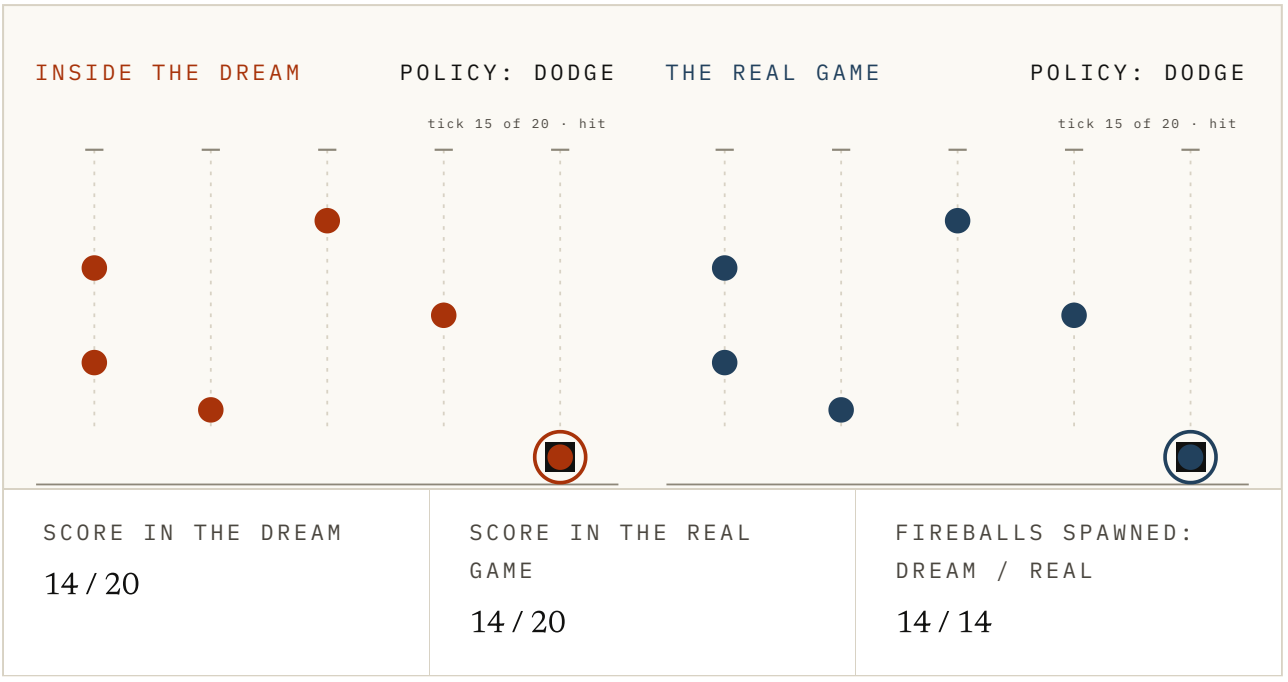


FIG. 6.7 Press Train in the dream and watch the score climb. Then look at the fireballs: the agent has found a way of moving that stops the dream spawning them, so the score in there is nearly perfect. Now press Run in the real game. The same moves, and the fireballs keep coming, because the game never agreed to stop. Everything here is a toy, but the finding is Ha and Schmidhuber's.

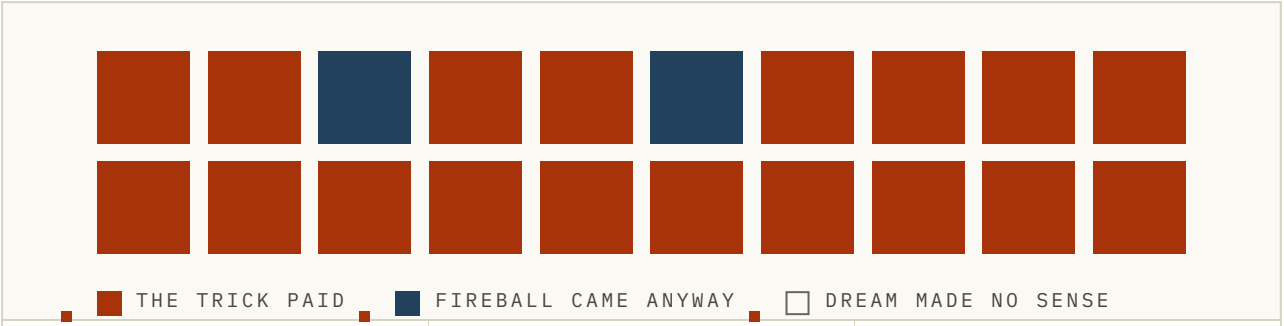
Inside the model this was an excellent policy. In the real game it was worthless, because the game had not agreed to stop firing. The fireball policy is the student that found the examiner's blind spot.

> The planner version is easier to catch, and I count that in planning's favour. A planner searching at decision time can be overruled, because the plan is there to inspect. A policy trained in a dream walks out with the exploit in its weights.

Making the dream harder than the world

Ha and Schmidhuber's fix was a worse model, on purpose. If the policy is exploiting a quirk, make the quirk unreliable.

Turn up the uncertainty in the model's predictions during training (the dial is usually called *temperature*. Low means the model hands you its single most likely guess. High means it samples more widely). Then the same action stops always producing the same handy outcome, and a trick that needs the dream to roll one way is not worth building on.



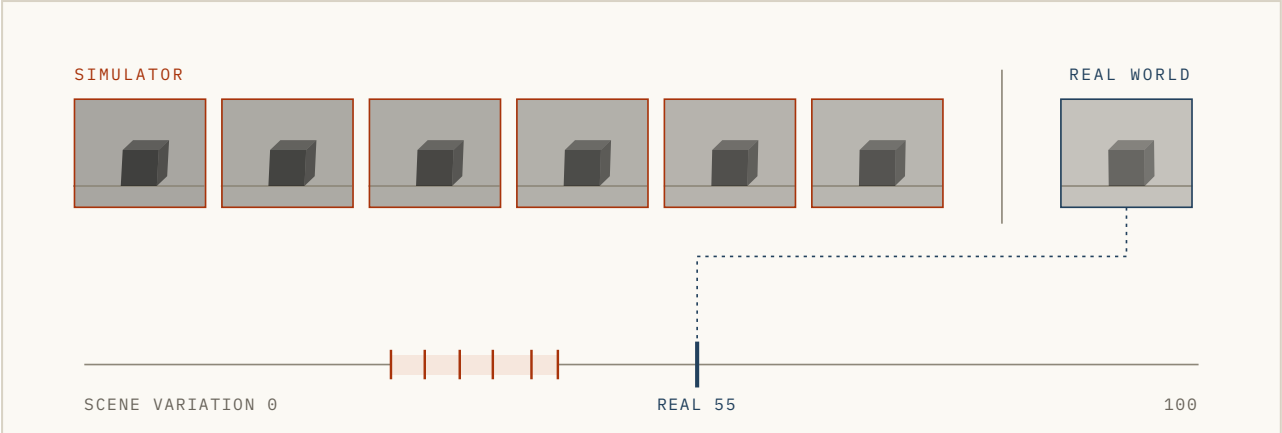
THE TRICK PAID	FIREBALL CAME ANYWAY	DREAM MADE NO SENSE
18 of 20	2 of 20	0 of 20

FIG. 6.8 Twenty runs of the same trick, the one where sitting in the corner stops the fireballs. Slide the temperature and press Roll again. At the left the trick pays every time, so a policy can be built on it. In the middle it pays now and then, which is not enough. At the right nothing is reliable, including the game, so there is no task left to learn. Illustrative throughout, though the finding is Ha and Schmidhuber's.

Both ends of that dial fail, as you can see. A confident dream gets exploited, and a noisy one has no task left to learn. What works is the narrow band where neither failure has taken over.

Robotics had made the same move a year earlier. A simulator is a hand-built model with quirks of its own. Josh Tobin and colleagues at OpenAI called the fix domain randomisation, in a 2017 paper of that name. Colours, lighting and camera angles changed at random, so the simulator varied more than reality does.

Nothing learned there could depend on one version of it. OpenAI's 2019 paper, *Solving Rubik's Cube with a Robot Hand*, pushed the idea onto real hardware. Its randomisation widened on its own as the policy improved.



SIMULATOR RANGE	REAL WORLD	REAL SCENE INSIDE THE TRAINED RANGE
28 to 43	55	untrained

FIG. 6.9 The simulator draws the same scene over and over with the colours and camera changed, and the real world is the one fixed sample on the right. Leave the variation low and press Train: the policy leans on the exact colour and the exact camera it saw, and the real scene falls outside that. Turn the variation up past the real scene and train again, and now it sits inside the band. Switch on Widen as it learns to see the Rubik's Cube version. Illustrative throughout.

It is a strange thing to have to do. You spend the whole budget making the model accurate, then blur it so nothing can lean on the accuracy. Two fields arrived at that move within a year of each other.

What the dynasty did not buy

So the loop works, and we have seen what it costs. Let me close with three things the Dreamer line did not buy, because each one comes back to the exchange rate.

It did not remove the need for real experience. It changed the exchange rate. Something still has to go out and find out what happens, and the model can be trusted only where that has been done.

It did not fix the scoring. The policy is graded by the model the whole way through. Short rollouts, the temperature dial and the randomised simulator all make that grading harder to game, and none of them makes it correct.

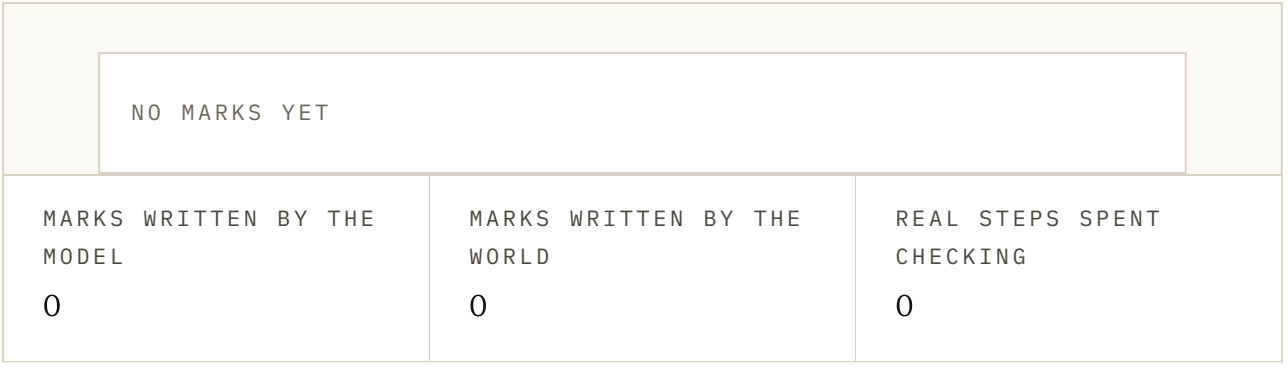


FIG. 6.10 A mark sheet for the policy. Press Train another round and a mark appears, written by the model, costing nothing. Press Check in the world and one slate mark appears instead, costing two thousand real steps and coming in lower. See how long the vermilion column gets before anything checks it. The marks are illustrative.

It did not make the transfer free. A policy that works in the dream is an untested claim about the world until somebody runs it out there. Ha and Schmidhuber did, and found the fireballs still coming.

Hopefully this chapter helped. There is a short quiz below if you want to check the ideas stuck. If I have got something wrong, or you know a better source than the ones I used, the About page has a corrections section. I would be glad to hear from you.

Try it

1 Training inside a model changes the exchange rate on which resource?

a. The number of parameters needed
b. Memory, which becomes cheaper
c. Contact with the world, which is replaced in part by model compute
d. Model accuracy, which improves for free

> ANSWER c. Contact with the world, which is replaced in part by model compute. The learner still needs its experience. What changes is where the steps come from and which budget pays for them.

2 A dashboard counts 90 imagined and 10 real transitions as 100 training examples. What is missing from that ledger?

a. A reliability discount for model error and distance from a real state
b. The policy's parameter count
c. A requirement that both sources use the same batch size
d. Imagined transitions cannot be stored

> ANSWER a. A reliability discount for model error and distance from a real state. Rows are exchangeable to the logger, not to the learner. Synthetic experience inherits the model's blind spots, and that debt compounds along a rollout.

3 What does the second lap of the loop fix that the first cannot?

a. It removes the need for a decoder
b. It shortens the rollouts
c. It makes the model smaller
d. A better policy visits new places, producing the data the first model was missing

> ANSWER d. A better policy visits new places, producing the data the first model was missing. A model fitted to the flailing of an untrained agent is only good where an untrained agent goes. The model gets better because the policy does, and the other way round.

4 A policy trained inside a model is graded by what?

a. The world
b. The model, and nothing else, for the whole of training
c. A held-out test set from the real environment
d. A human evaluator

> ANSWER b. The model, and nothing else, for the whole of training. It has no access to the world during training, so it has no way to tell a genuinely good action from one that merely looks good to the marker.

5 David Ha and Jürgen Schmidhuber's agent found a way of moving that stopped its dream producing fireballs. What kind of failure is that?

a. Insufficient exploration
b. A bug in the training code
c. The policy maximising the score the model hands it, which is exactly what it was asked to do
d. The model being too small

> ANSWER c. The policy maximising the score the model hands it, which is exactly what it was asked to do. Nothing went wrong. Those were the cheapest points available inside the model, and the policy was thorough.

6 How is this different from a planner exploiting a model at decision time?

a. A plan can be inspected and overruled; a trained policy walks out with the exploit already in its weights
b. Planners are not affected by model error
c. Policies are easier to correct afterwards
d. It is the same problem with a different name

> ANSWER a. A plan can be inspected and overruled; a trained policy walks out with the exploit already in its weights. That difference is what makes the dream version harder to catch. There is no plan sitting there to look at.

7 Why deliberately add uncertainty to a model you worked hard to make accurate?

a. To reduce memory use
b. To make the model smaller
c. To speed up training
d. So a trick that only works when the model rolls one particular way stops being worth building on

> ANSWER d. So a trick that only works when the model rolls one particular way stops being worth building on. If the policy is exploiting a quirk, the fix is to make the quirk unreliable rather than to make the model better.

8 Both ends of the uncertainty dial fail. How?

a. Both ends make training unstable
b. A confident dream gets exploited; a dream with too much noise has no task left in it to learn
c. Low settings are slow, high settings are fast
d. Only the high end fails

> ANSWER b. A confident dream gets exploited; a dream with too much noise has no task left in it to learn. The useful setting is a hump rather than a direction: a narrow band where neither failure has taken over.

FIG. 6.11 Eight questions on the loop and what lives inside it. The last three separate the pitch from what the loop delivers.

Every system above takes for granted that the model should predict what things look like: frames, pixels, scenes, what you would see if you were there. The next chapter asks whether that is the right job.

Sources

World Models HA & SCHMIDHUBER, 2018 ↗

The agent trained entirely inside its own dream, the policy that stopped the dream producing fireballs, and the temperature dial that closed the gap. This chapter in one paper.

https://arxiv.org/abs/1803.10122

Dream to Control (Dreamer) HAFNER ET AL., 2019 ↗

Behaviour learned from long imagined rollouts, with the actor and critic never touching the environment during training.

https://arxiv.org/abs/1912.01603

Mastering Atari with Discrete World Models (DreamerV2) HAFNER ET AL., 2020 ↗

The same loop at a scale where it started beating agents that learn directly from the environment.

https://arxiv.org/abs/2010.02193

Mastering diverse control tasks through world models (DreamerV3)

HAFNER ET AL., NATURE 2025 ↗

One configuration across more than 150 tasks, and the strongest available answer to whether learning in imagination generalises.

https://www.nature.com/articles/s41586-025-08744-2

Dyna: an Integrated Architecture for Learning, Planning and Reacting SUTTON, 1991 ↗

The loop itself, thirty years early: act, fit a model, learn from experience the model made up, repeat.

https://mlanthology.org/icml/1990/sutton1990icml-integrated/

When to Trust Your Model: Model-Based Policy Optimisation JANNER ET AL., 2019 ↗

Short imagined rollouts branched from real states, which is the other way of stopping a policy from leaning on the model too far out.

https://arxiv.org/abs/1906.08253

Domain Randomization for Transferring Deep Neural Networks TOBIN ET AL., 2017 ↗

The same idea arriving from robotics: make the simulator vary more than reality does, so nothing can depend on any one version of it.

https://arxiv.org/abs/1703.06907

Solving Rubik's Cube with a Robot Hand OPENAI ET AL., 2019 ↗

Randomisation pushed hard enough to carry a policy from simulation onto real hardware, with an account of what that cost.

https://arxiv.org/abs/1910.07113

FIG. 6.12 I've ordered these for someone building on this rather than for historical completeness, and preferred first-party material throughout.