From Chain-of-Thought to Chain-of-State

Why capable models must predict the world back

Dan Kondratyuk · rekursiv.ai

CVPR 2026 Tutorial · From Perception to Simulation · Denver

When I ask, “what is a *world model*?” to different people, I always get a different answer.

Video world model

Genie 3, LingBot World

Video world model

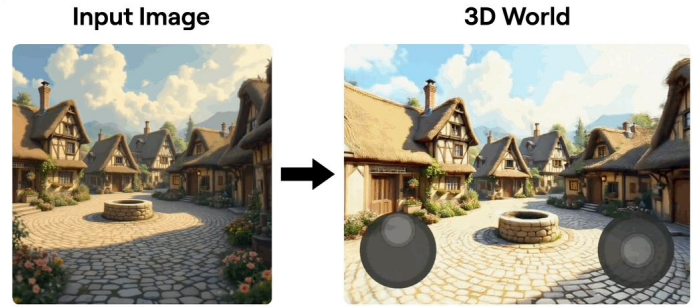
Robotics world model

1X

Video world model

Robotics world model

3D world model

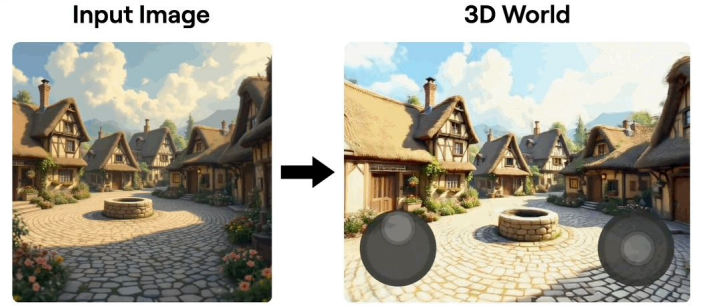


Marble

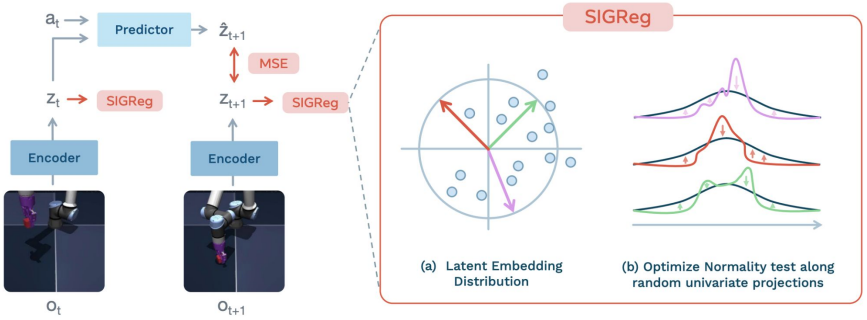
Video world model

Robotics world model

3D world model



Latent world model



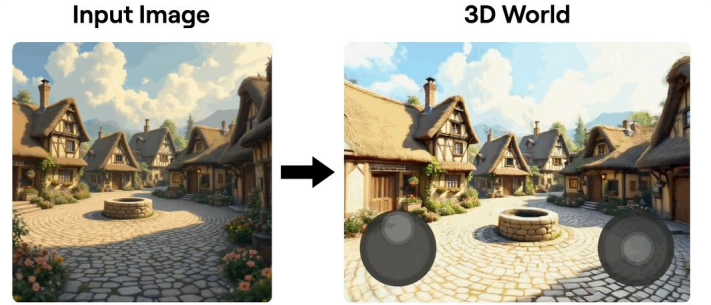
LeWorldModel

Video world model

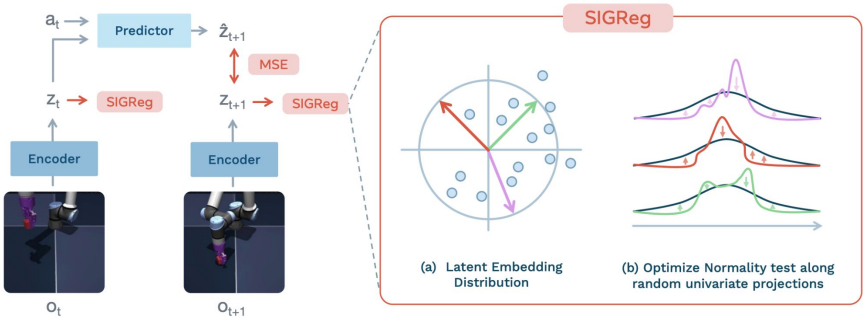
Robotics world model

What isn’t a world model?

3D world model



Latent world model



Video world model

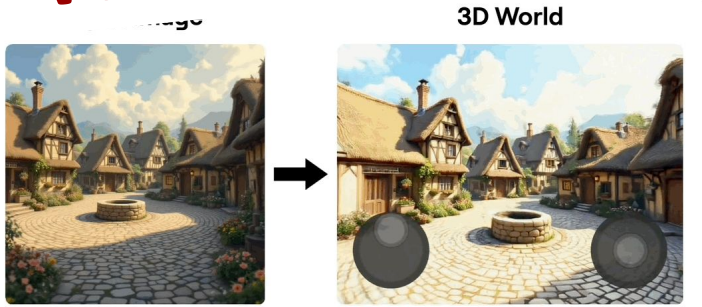
Robotics world model

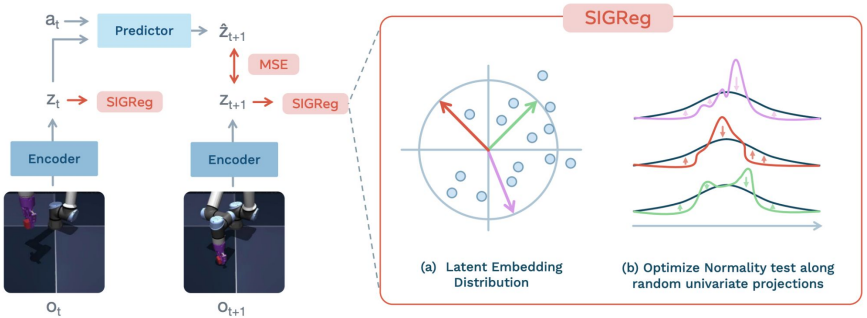
What isn't a world model?

What puts the *world* in world model?

3

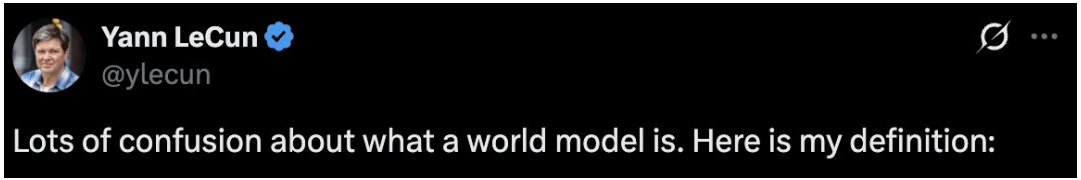
Latent world model

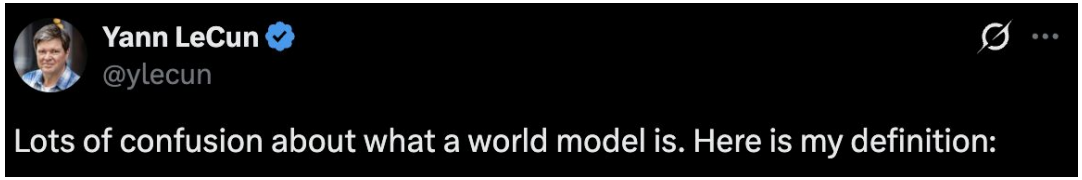





Natasha Malpani
@natashamalpani
world models are a sexy misnomer.
@ylecun , @nvidia , and @drfeifei
are all building world models. they are not building the same thing.


Natasha Malpani
@natashamalpani
world models are a sexy misnomer.
@ylecun , @nvidia , and @drfeifei
are all building world models. they are not building the same thing.


Yann LeCun
@ylecun
Lots of confusion about what a world model is. Here is my definition:



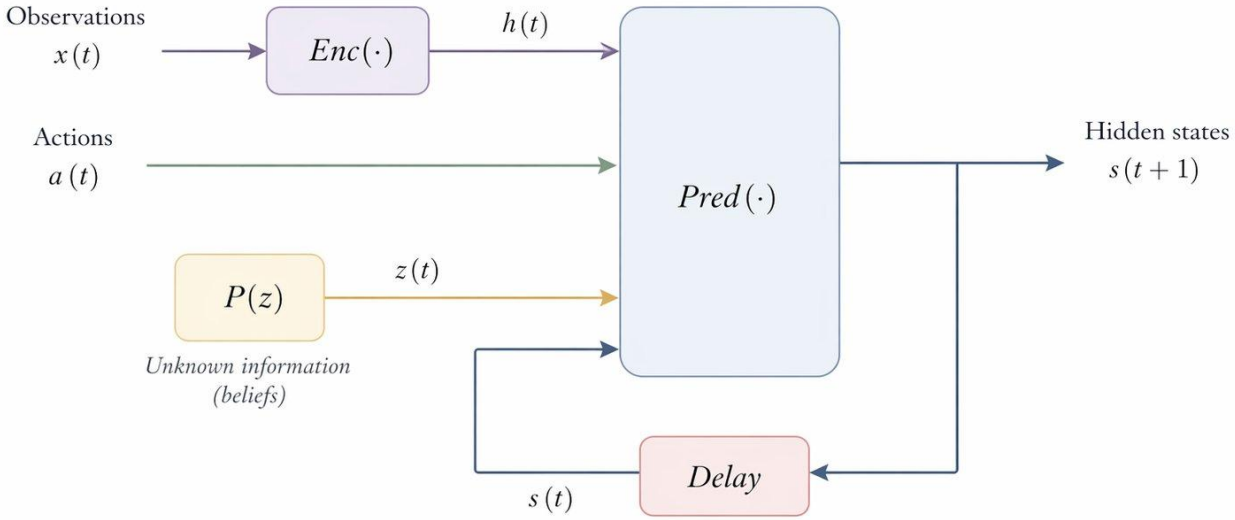


Figure 1. Block diagram of the world model. The encoder $Enc(\cdot)$ maps observations $x(t)$ to latent embeddings $h(t)$. The predictor $Pred(\cdot)$ estimates the next hidden state $s(t+1)$ using the latent embedding $h(t)$, actions $a(t)$, latent variables $z(t)$ from $P(z)$ (unknown information or beliefs), and the previous hidden state $s(t)$ provided through a unit delay.

Given:

x(t): observation

s(t): previous state estimate of the world

a(t): action proposal

z(t): latent variable proposal

A world model computes:

h(t) = Enc(x(t)), the representation

s(t+1) = Pred(h(t), s(t), z(t), a(t)), the prediction

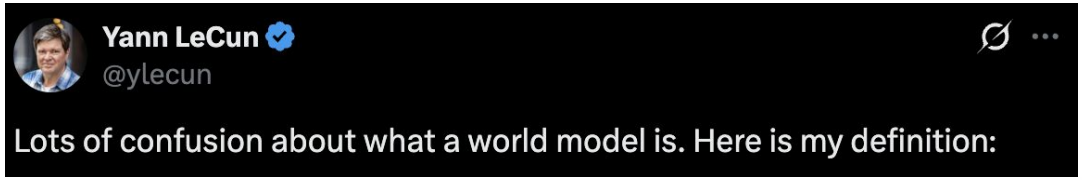
Where:

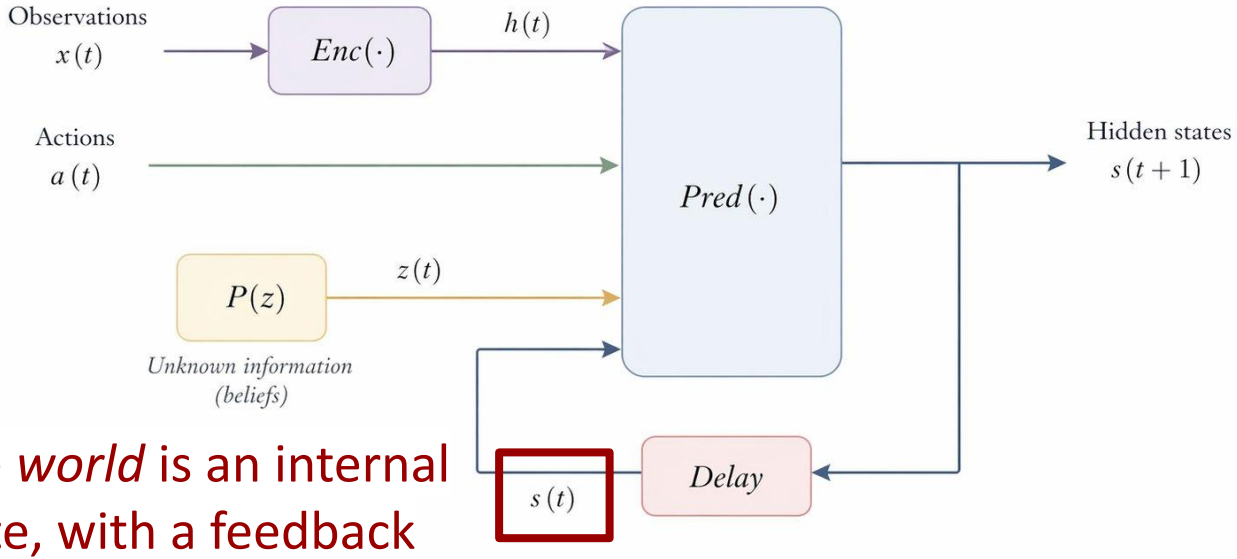
Enc(.): (trainable) encoder

Pred(.): (trainable) hidden state function

z(t): prior beliefs over plausible predictions

Dec(s(t)): (not shown) optional view/transformation of the hidden state





The *world* is an internal state, with a feedback loop

Figure 1. Block diagram of the world model. The encoder $Enc(\cdot)$ maps observations $x(t)$ to latent embeddings $h(t)$. The predictor $Pred(\cdot)$ estimates the next hidden state $s(t+1)$ using the latent embedding $h(t)$, actions $a(t)$, latent variables $z(t)$ from $P(z)$ (unknown information or beliefs), and the previous hidden state $s(t)$ provided through a unit delay.

Given:

x(t): observation

s(t): previous state estimate of the world

a(t): action proposal

z(t): latent variable proposal

A world model computes:

h(t) = Enc(x(t)), the representation

s(t+1) = Pred(h(t), s(t), z(t), a(t)), the prediction

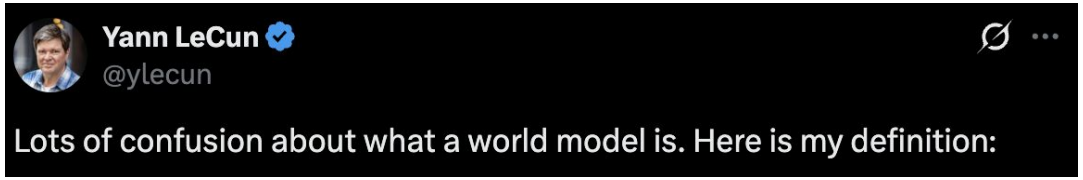
Where:

Enc(.): (trainable) encoder

Pred(.): (trainable) hidden state function

z(t): prior beliefs over plausible predictions

Dec(s(t)): (not shown) optional view/transformation of the hidden state



Given:

x(t): observation

s(t): previous state estimate of the world

a(t): action proposal

z(t): latent variable proposal

A world model computes:

h(t) = Enc(x(t)), the representation

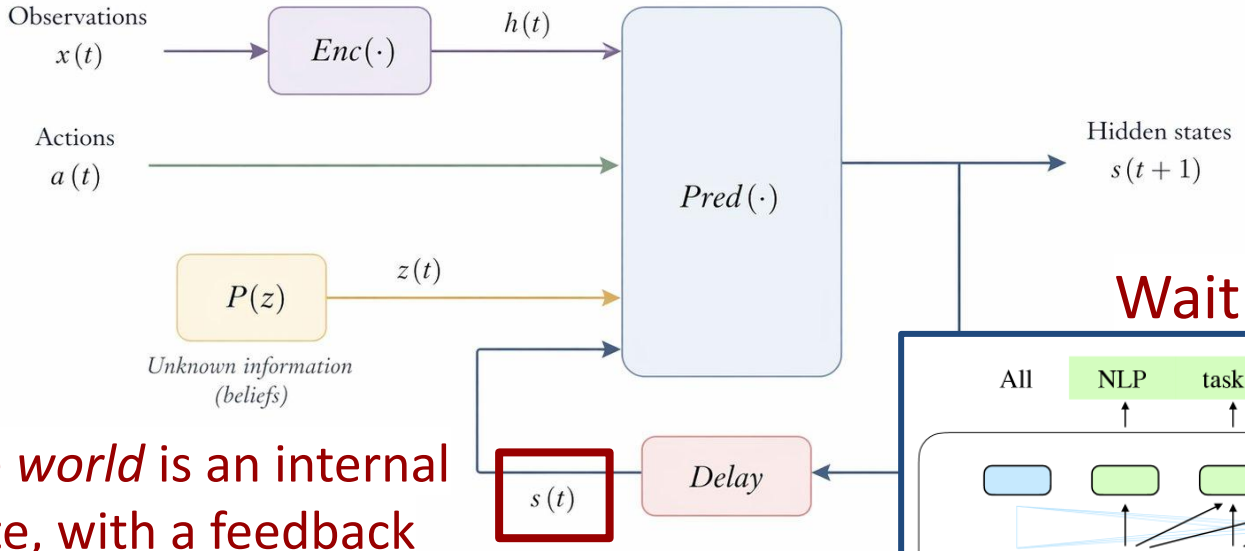
s(t+1) = Pred(h(t), s(t), z(t), a(t)), the prediction

Where:

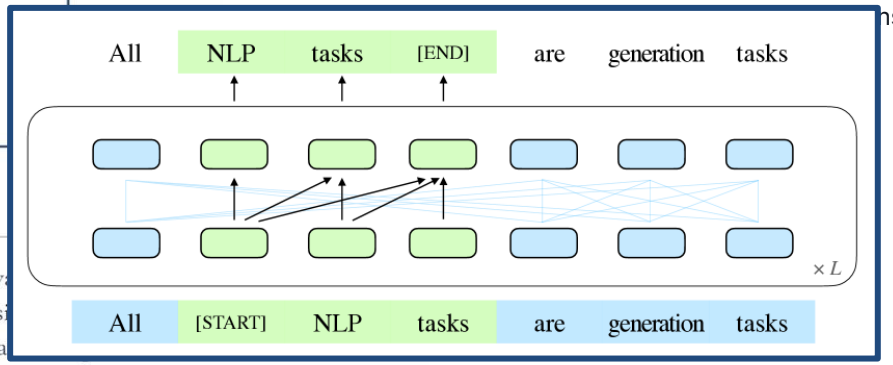
Enc(.): (trainable) encoder

den state function

ns



Wait a second…



The *world* is an internal state, with a feedback loop

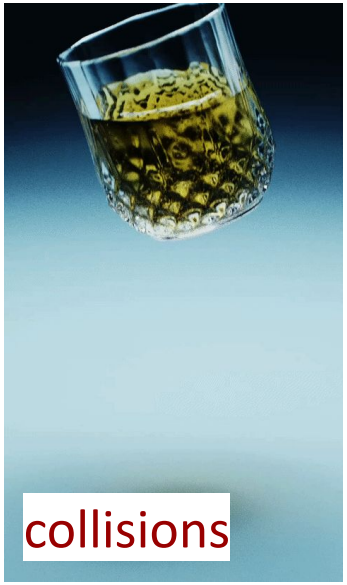
igure 1. **Block diagram of the world model.** The encoder $Enc(\cdot)$ maps observa
embeddings $h(t)$. The predictor $Pred(\cdot)$ estimates the next hidden state $s(t+1)$ usi
$h(t)$, actions $a(t)$, latent variables $z(t)$ from $P(z)$ (unknown information or beliefs), a
state $s(t)$ provided through a unit delay.

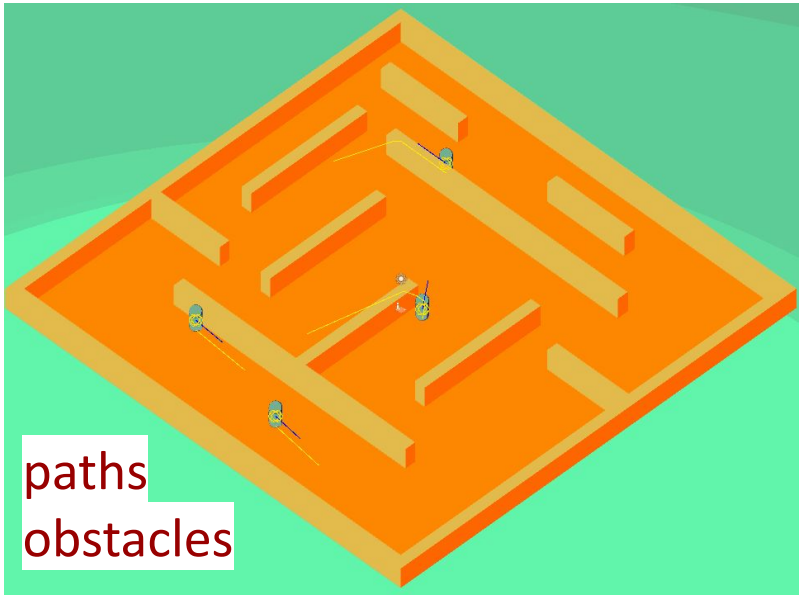
Key idea: world models are *simulators* of an *environment*

States are a way of remembering different parts of the world

Analogous to chain-of-thought, world models are *chain-of-state*

States can be explicit representations or implicitly learned





What's the ultimate form of a world model?

What's the *ultimate form* of a world model?

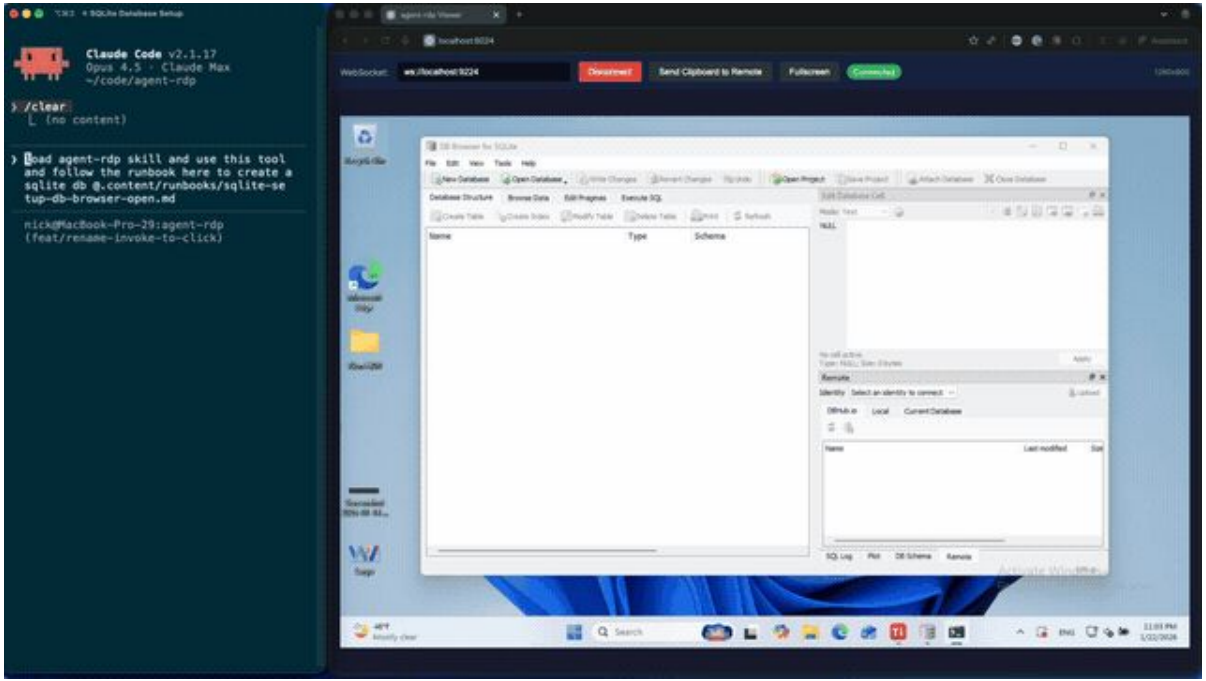
A thing which is infinitely interactive

Key property: informationally scale-invariant

- I click on anything and can zoom in. Then zoom in on that.
- I can split anything in half. And again.
- I can go in; I can go out.

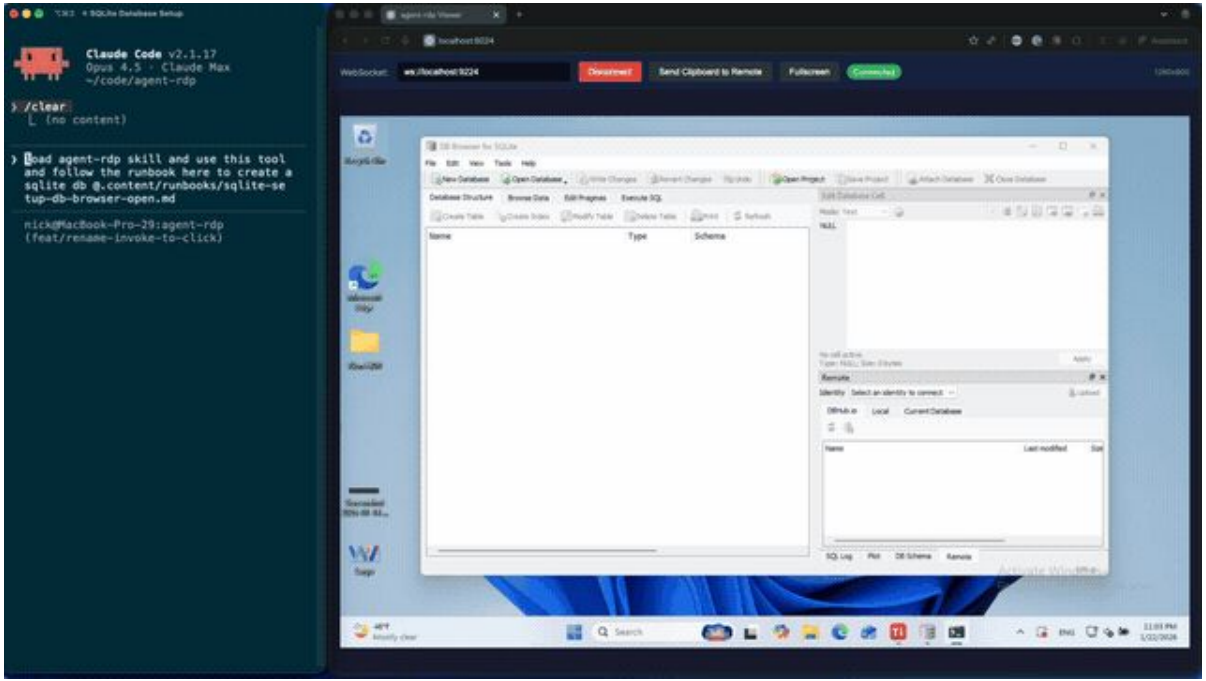
The endgame of world models: *simulate an operating system*?

Simulate any program inside a neural network



Why does the world need this?

- It’s a Universe Simulator (do anything)
- It’s expensive and slow to do things in real life
- Counterfactuals: What would the world be like if Henry Ford never born?

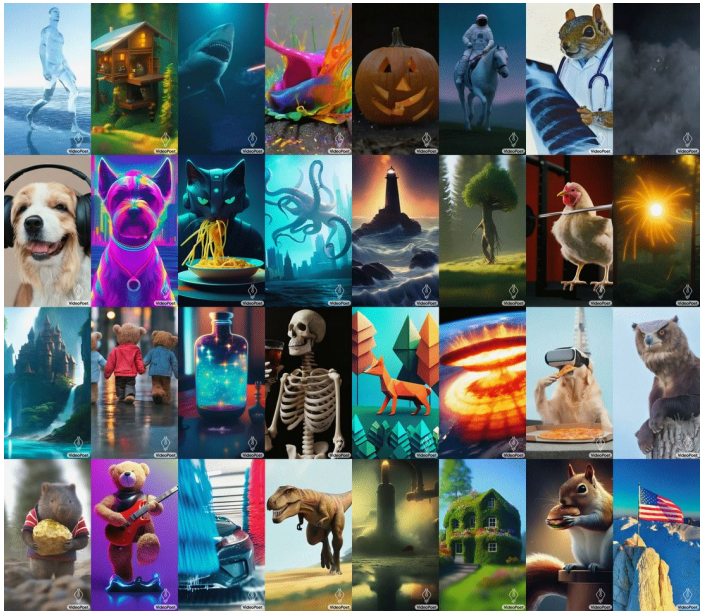


How do we get there?

VideoPoet: the “Video-First Foundation Model”

“what i cannot create i do not understand” — Richard Feynman

Our bet: if we learn to generate videos, than we can understand them



VideoPoet: the “Video-First Foundation Model”

“what i cannot create i do not understand” — Richard Feynman

❌ **Our bet:** if we learn to generate videos, than we can understand them

- By training on unsupervised videos, **we get neither good understanding nor generation**

April-May 2023

VideoPoet: the “Video-First Foundation Model”

“what i cannot create i do not understand” — Richard Feynman

❌ **Our bet:** if we learn to generate videos, than we can learn to understand them

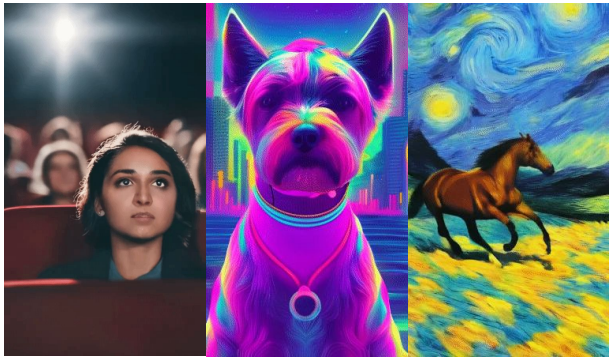
- By training on unsupervised videos, **we get neither good understanding nor generation**

April-May 2023

August 2023

September 2023

December 2023



Where did this change in improvement come from?

VideoPoet: the “Video-First Foundation Model”

“what i cannot create i do not understand” — Richard Feynman

❌ **Our bet:** if we learn to generate videos, than we can learn to understand them

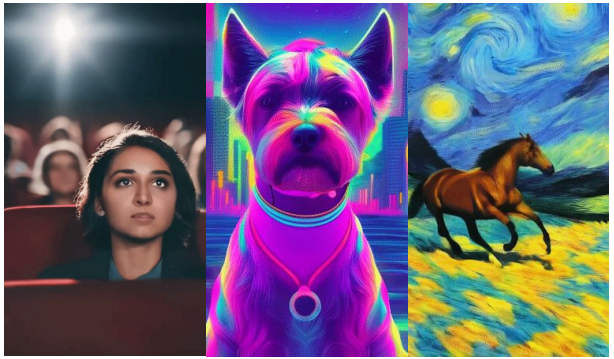
- By training on unsupervised videos, **we get neither good understanding nor generation**

April-May 2023

August 2023

September 2023

December 2023



Where did this change in improvement come from?

The answer is **language**

Additional Problems in World Modeling

Problem 1: the exposure bias problem

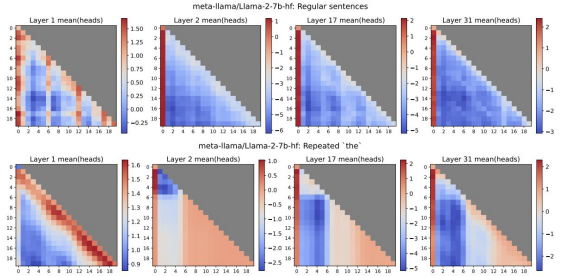
After a while, states start to degrade

- This happens both in video models and LLMs
- Often caused by attention sinks in transformers
  - But can also be caused by RoPE
- Solutions involve closing the out-of-distribution train-inference gap (e.g., Self-Forcing)

Google DeepMind

Interpreting the Repeated Token Phenomenon in Large Language Models

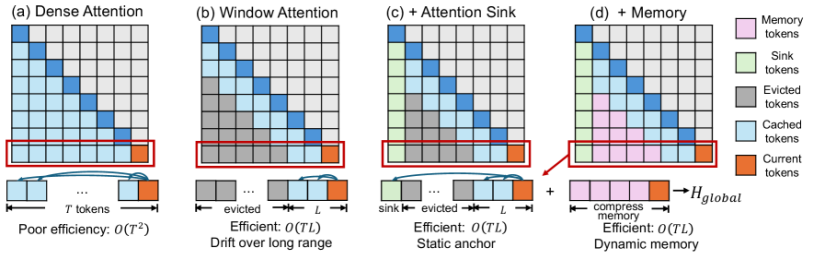
Itay Yona[1], Ilia Shumailov[1], Jamie Hayes[1], Federico Barbero[2] and Yossi Gandelsman[3]
[1]Google DeepMind, [2]University of Oxford, [3]UC Berkeley



1s

5s

CausVid

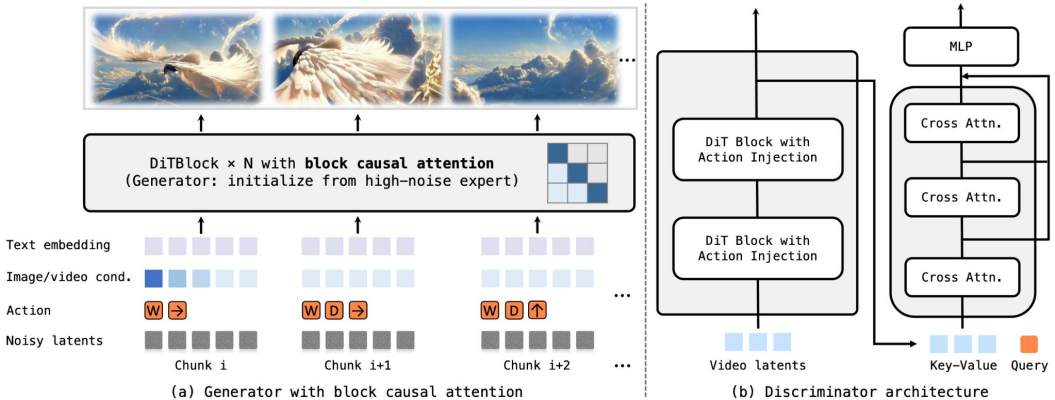


VideoSSM

Problem 2: The long horizon challenge

How to keep track of very long context? What if states are very large?

- In video generation, **this is brutal**: a 1080p@30fps 60-second video using a 4×16×16 tokenizer will use ~3.6M tokens!
  - How can we ever hope to keep track of hour-long sessions?



https://github.com/robbyant/lingbot-world

Problem 2: The long horizon challenge

How to keep track of very long context? What if states are very large?

- Data compression: encode more data in fewer tokens
- Sparse/linear attention: compress/prune states in the past
- Hierarchical representations: represent data/state at different granularities
- Multi-view: different features of the same environment

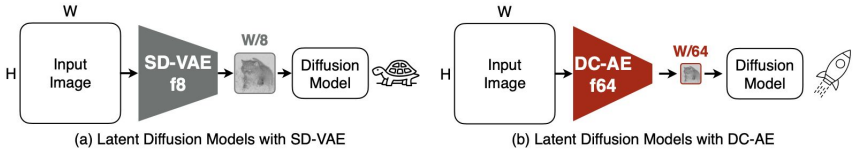
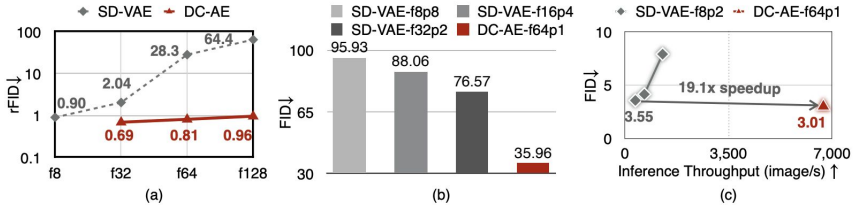
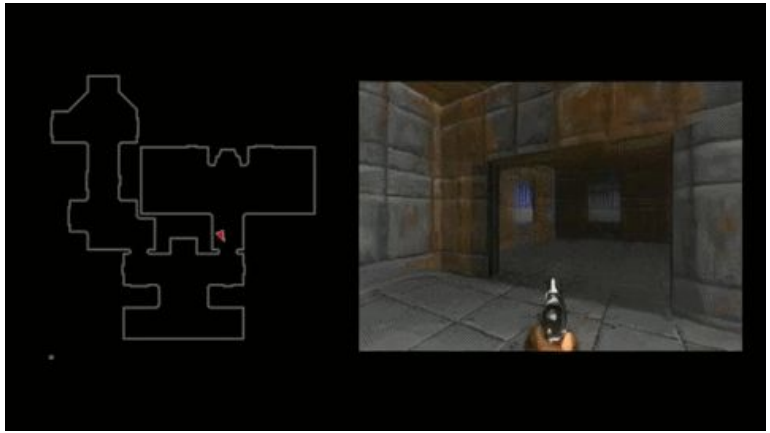


Figure 1: DC-AE accelerates diffusion models by increasing autoencoder's spatial compression ratio.





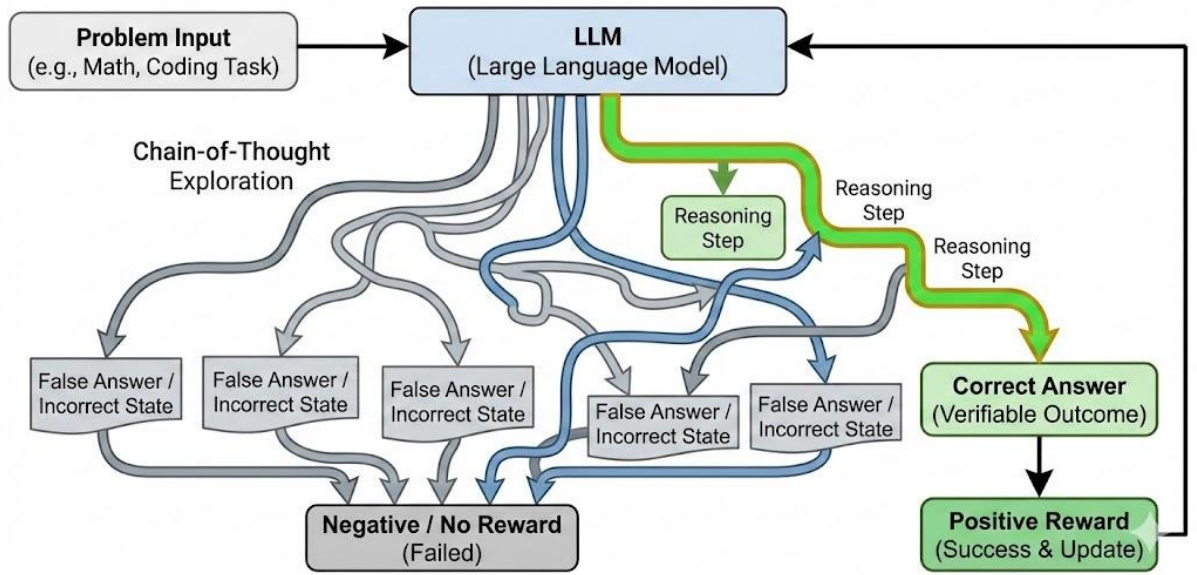
https://github.com/mit-han-lab/efficientvit

https://ryanpo.com/multigen/

Problem 3: Introducing planning and reasoning

Many current world models are surface-level: they only predict the *next* state

To get more abstract reasoning, we need to think *many steps ahead*
RLVR is a great application of this!



LLMs as World Models

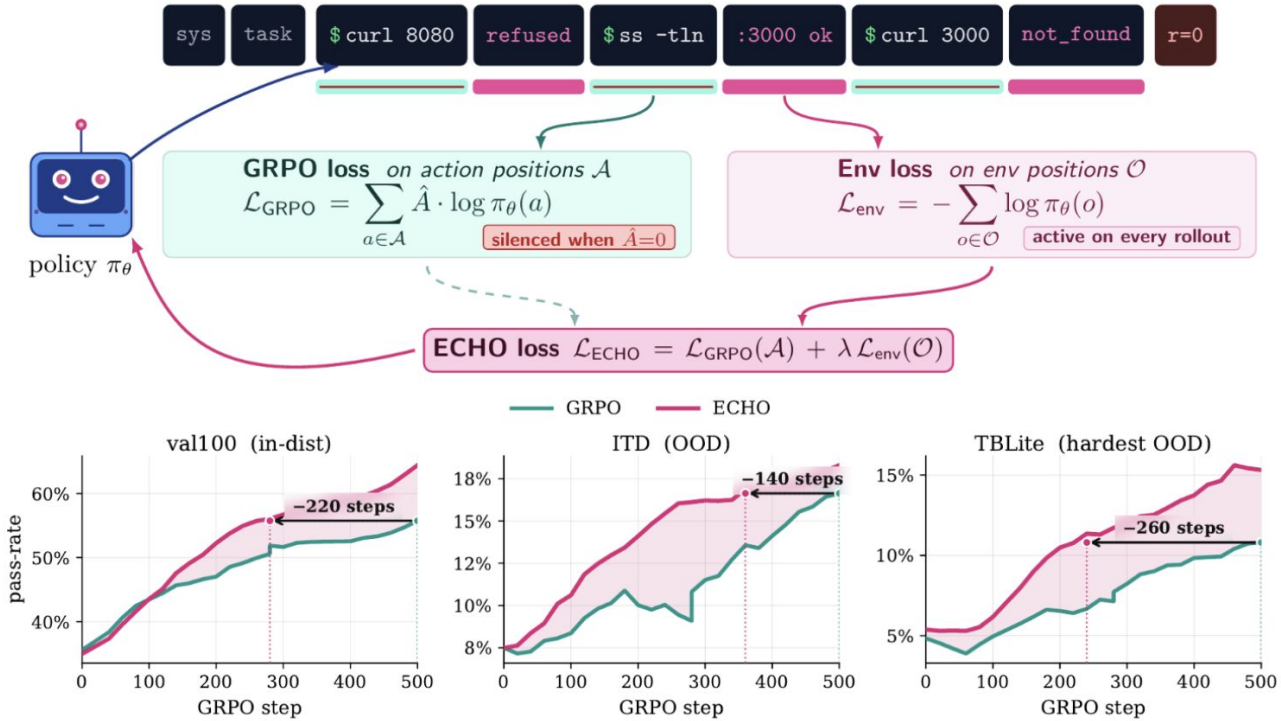
Example: **terminal simulation**

ECHO: Terminal Agents Learn World Models for Free

Vaishnavi Shrivastava Ahmed Awadallah Dimitris Papailiopoulos
Microsoft Research

Code: https://github.com/microsoft/echo-rl

Predicting terminal output improves terminal usage accuracy. Why?



But is this the best way?

If we want an OS, how to simulate programs in neural networks?

The Halting Problem: you can’t know in advance the outcome of every program

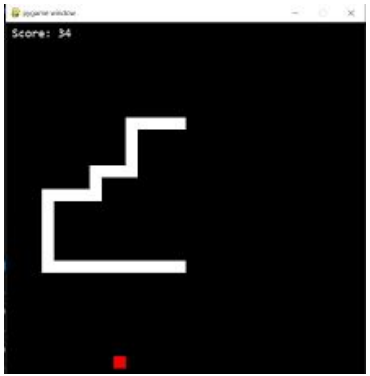
To some degree, this is already being approximated with coding agents (but with limited bandwidth). Soon we may also see this for visual models

Code

```
if event.type == pygame.KEYDOWN:
    # Stop the snake going into itself
    # if we are moving down dont move up etc..
    if event.key == pygame.K_UP:
        if mainGame.snake.direction.y != 1:
            mainGame.snake.direction = Vector2(0,-1)
    if event.key == pygame.K_DOWN:
        if mainGame.snake.direction.y != -1:
            mainGame.snake.direction = Vector2(0,1)
    if event.key == pygame.K_LEFT:
        if mainGame.snake.direction.x != 1:
            mainGame.snake.direction = Vector2(-1,0)
    if event.key == pygame.K_RIGHT:
        if mainGame.snake.direction.x != -1:
            mainGame.snake.direction = Vector2(1,0)
```

Video World Model

Game

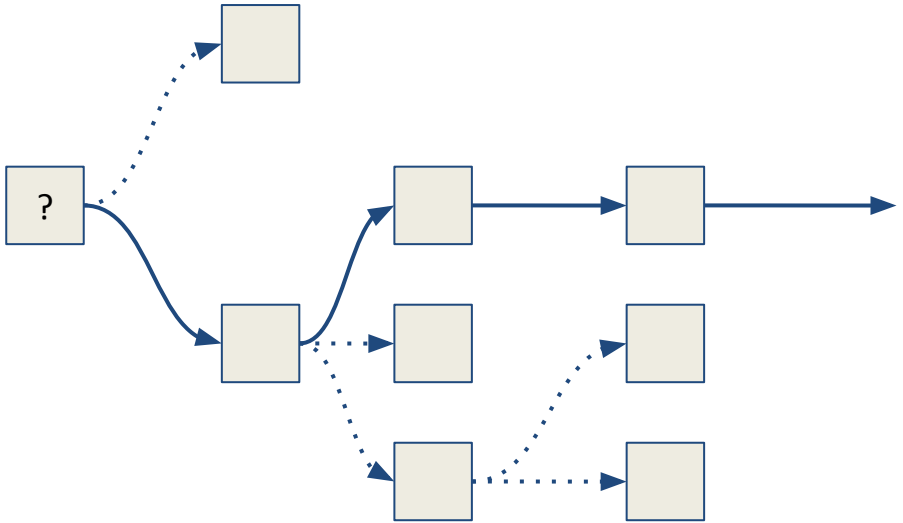


The Scientic Method as a World

The cost of a single paper: possibly millions of $$$

Ideas are expensive to test, because you can't tell if an idea is a good one

It's a long chain of states to get from point A to point B



VideoPoet: A Large Language Model for Zero-Shot Video Generation

Dan Kondratyuk [*1] Lijun Yu [*1 2] Xiuye Gu [*1] José Lezama [*1] Jonathan Huang [*1] Grant Schindler [1]
Rachel Hornung [1] Vighnesh Birodkar [1] Jimmy Yan [1] Ming-Chang Chiu [1] Krishna Somandepalli [1]
Hassan Akbari [1] Yair Alon [1] Yong Cheng [1] Josh Dillon [1] Agrim Gupta [1] Meera Hahn [1] Anja Hauth [1]
David Hendon [1] Alonso Martinez [1] David Minnen [1] Mikhail Sirotenko [1] Kihyuk Sohn [1] Xuan Yang [1]
Hartwig Adam [1] Ming-Hsuan Yang [1] Irfan Essa [1] Huisheng Wang [1] David A. Ross [1] Bryan Seybold [*1]
Lu Jiang [*1 2]

Abstract

We present VideoPoet, a model for synthesizing high-quality videos from a large variety of conditioning signals. VideoPoet employs a decoder-only transformer architecture that processes multimodal inputs – including images, videos, text, and audio. The training protocol follows that of Large Language Models (LLMs), consisting of two stages: pretraining and task-specific adaptation. During pretraining, VideoPoet incorporates a mixture of multimodal generative objectives within an autoregressive Transformer framework. The pretrained LLM serves as a foundation that is adapted to a range of video generation tasks. We present results demonstrating the model's state-of-the-art capabilities in zero-shot video generation, specifically highlighting the generation of high-fidelity motions. Project page: https://sites.research.google/videopoet/.

1. Introduction

Recently, there has been a surge of generative video models capable of a variety of video creation tasks. These include text-to-video (Zhang et al., 2023a; Singer et al., 2022), image-to-video (Yu et al., 2023c), video-to-video stylization (Chen et al., 2023b; Chai et al., 2023; Voleti et al., 2022), and video editing (Ceylan et al., 2023; Wang et al., 2023b; Geyer et al., 2023) among other video applications. Most existing models employ diffusion-based methods for video generation. These video models typically start with a pretrained image model, such as Stable Diffusion (Rombach et al., 2022; Podell et al., 2023), that produces high-fidelity images for individual frames, and then fine-tune the model to improve temporal consistency across video frames.

While Large Language Models (LLMs) are commonly used as foundation models across various modalities including language (Brown et al., 2020), code (Li et al., 2023; OpenAI, 2023), audio (Rubenstein et al., 2023), speech (Agostinelli et al., 2023), and robotics (Driess et al., 2023; Zitkovich et al., 2023), the diffusion model remains the predominant approach for video generation. Although early research has demonstrated the effectiveness of LLMs in text-to-image generation, *e.g.*, DALL-E (Ramesh et al., 2022), Parti (Yu et al., 2022) and (Ding et al., 2021), and text-to-video, *e.g.*, CogVideo (Hong et al., 2022)), language models have not reached a level of quality on par with video diffusion models in tasks like text-to-video generation as shown in previous studies (Nash et al., 2022; Villegas et al., 2022). In contrast to training exclusively for text-to-video tasks, the generative model of LLMs in the language domain emphasizes a large pretraining stage to learn a foundation (Bommasani et al., 2021) by examining pretraining tasks that extend beyond text-to-video generation.

A notable advantage of employing LLMs in video generation lies in the ease of integrating existing LLM frameworks. This integration allows for reusing LLM infrastructure and leverages the optimizations our community has developed over many years for LLMs, including optimizations in learning recipes for model scaling (Brown et al., 2020; Chowdhery et al., 2022), training and inference infrastructure (Du et al., 2022), hardware, among other advancements. This couples with their flexibility in encoding many diverse tasks in the same model (Raffel et al., 2020), which stands in contrast to most diffusion models where architectural changes and adapter modules are the dominant approach used to adapt the model to more diverse tasks (Zhang et al., 2023b).

In this paper, we exploit language models for video generation, following the canonical training protocols of LLMs in the language domain. We introduce *VideoPoet*, a language

[*]Equal contribution [1]Google [2]Carnegie Mellon University. Correspondence to: Lijun Yu <lijuny@google.com>, Jonathan Huang <jonathanhuang@google.com>, David Ross <dross@google.com>, Bryan Seybold <seybold@google.com>, Lu Jiang <roadjiang@gmail.com>.

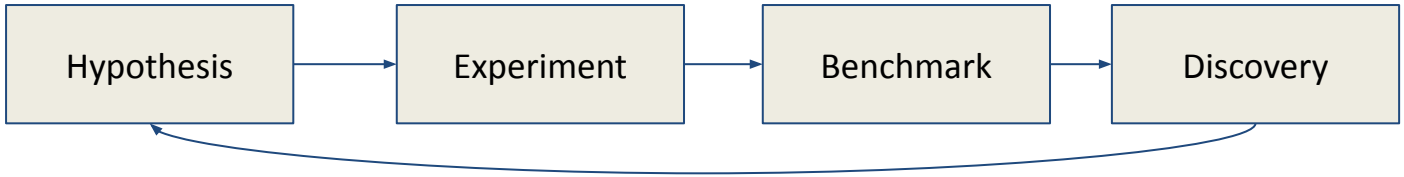
Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

How to get there: *simulate science*

rekursiv.ai's goal: autonomous ML research with AI Scientists

How do we know if an idea is any good?

Don't just imagine the future, test it! Run simulations of simulations

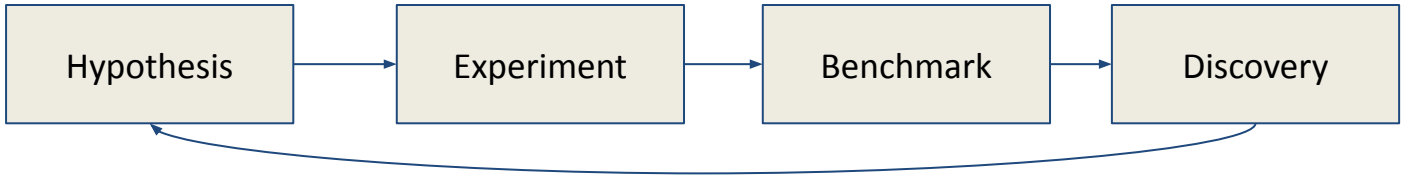


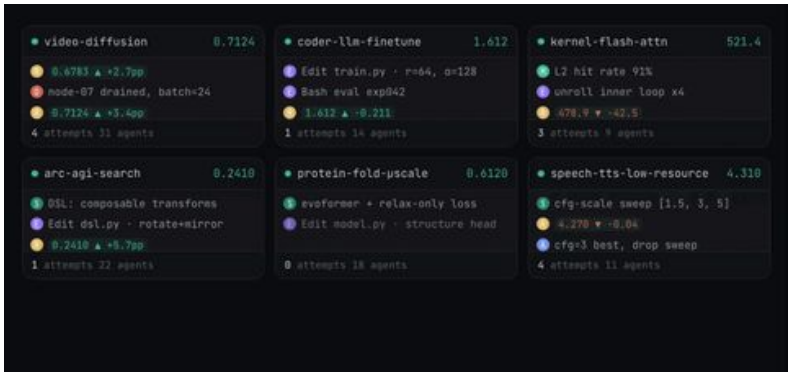
A secondary benefit: simulate the science of science

ML research is hard, and most ideas and experiments fail

Therefore, building autonomous systems that do research can improve this success rate

We’re seeing real productivity gains from deploying swarms of agents to rapidly iterate 24/7



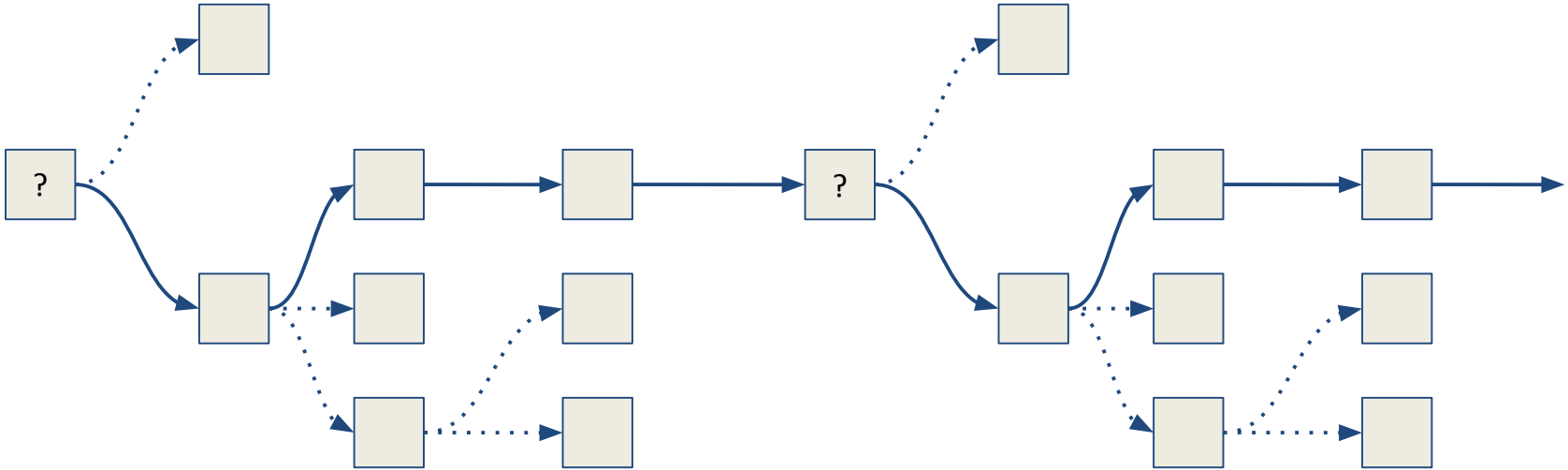


But we don’t get this for free

Current agents do science poorly, because they can’t easily connect hypotheses to results

You have to build lots of guardrails to keep them on track

More research is necessary to learn how to make it work better

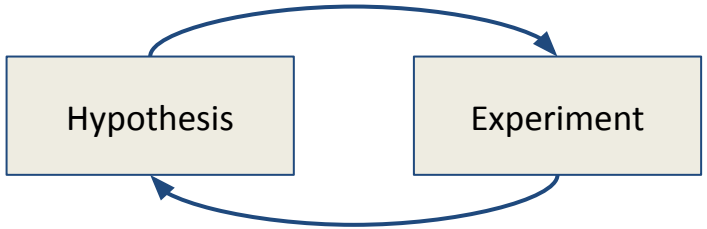


What should we do?

The ultimate form of a world model looks like an operating system

Thinking ahead, we can anticipate what problems we need to solve to get there

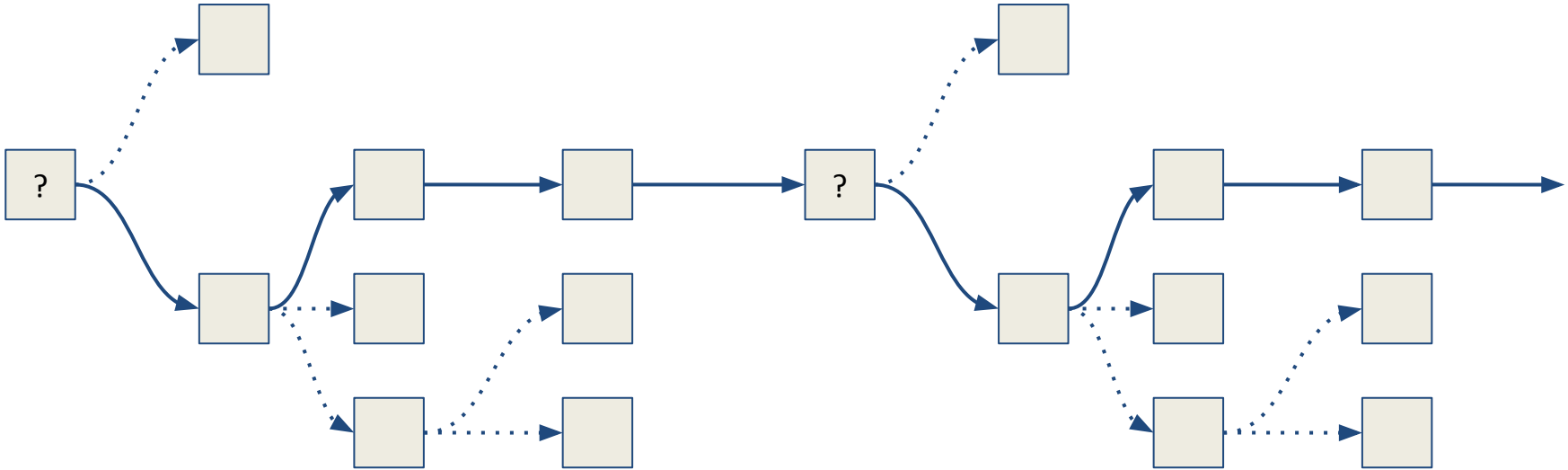
Automating the science will get us us there quicker



Automating the Scientific Method

If we automate this process, we create a simulated world in which everything is *testable* with an experiment

The next generation of world models can imagine future states by putting them to the test, accumulating knowledge like a scientist



How to get there: *simulate science*

Why is experimentation important?

The Halting Problem: you can't know in advance the outcome of every program