PROGRAMMATIC VIDEO PREDICTION USING LARGE LANGUAGE MODELS

Hao Tang* **Kevin Ellis**
Department of Computer Science
Cornell University
Ithaca, NY, USA
{ht383,kellis}@cornell.edu

Suhas Lohit **Michael J. Jones** **Moitreya Chatterjee**
Mitsubishi Electric Research Laboratories (MERL)
Cambridge, MA, USA
{slohit,mjones,chatterjee}@merl.com

ABSTRACT

The task of estimating the world model by describing the dynamics of a real world process assumes immense importance for anticipating and preparing for future outcomes and finds wide-spread use in applications such as video surveillance, robotics, autonomous driving, *etc.* This task entails synthesizing plausible visual futures, given a few frames of a video – necessary to set the visual context for the synthesis. Towards this end, different from end-to-end deep learning based approaches for video frame prediction, we propose ProgGen – which undertakes the task of video frame prediction by synthesizing computer programs which represent the dynamics of the video using a set of neuro-symbolic, human-interpretable set of states (one per frame) by leveraging the inductive biases of Large (Vision) Language Models (LLM/VLM). In particular, ProgGen utilizes LLM/VLM to synthesize computer programs to: (i) estimate the states of the video, given the visual context (*i.e.*the frames); (ii) predict the states corresponding to future time steps by estimating the transition dynamics; (iii) render the predicted states as visual RGB-frames. Empirical evaluations reveal that our proposed method outperforms competing techniques at the task of video frame prediction in two challenging environments: (i) PhyWorld and (ii) Cart Pole. Additionally, ProgGen permits counter-factual reasoning and editability, attesting to its effectiveness and generalizability. [1]

1 INTRODUCTION

The task of extrapolating the dynamics of a real world process into the future is an essential one and necessitates an understanding of the world model governing the dynamics of the process. As artificial intelligence systems become ubiquitous and get deployed to work side-by-side with humans, for tasks such as video surveillance Wang et al. (2018), autonomous driving Jain et al. (2015), robotics Cherian et al. (2024), it becomes increasingly more important that they be capable of reasoning about the dynamics of real world phenomena, given a visual context, and synthesize plausible futures.

With the recent success of deep neural networks, the research community has resorted to deploying such models for the task of video frame prediction Babaeizadeh et al. (2017); Denton & Fergus (2018); Chatterjee et al. (2021). Models developed for this task are capable of generating the future frames of a video, given a few past frames – in order to set the visual context for the generation. Lately, diffusion models Blattmann et al. (2023); Höppe et al. (2022) have shown the most promise for this task. These models have an encoder-decoder architecture. In the encoding stage, a small magnitude of noise (usually sampled from a Gaussian distribution) is iteratively added to a sample drawn from the training data, potentially until the real data is indistinguishable from pure (Gaussian) noise. The goal of the decoding phase then, is to iteratively de-noise the sample to eventually recover the original data. Despite their success, the widespread adoption of such approaches is limited primarily by their reliance on copious amounts of training data and their lack of interpretability.

*Work done during an internship at MERL.

[1]The code is available at: https://github.com/metro-smiles/ProgGen

Moreover, such models are only effective when the data on which the inference is to be performed is largely similar to the one on which they have been trained.

Existing physics-based approaches Wu et al. (2015); Liu et al. (2024) for the task of video frame generation/prediction, seek to estimate physical properties of the objects that are present in the video, such as their mass, position, *etc.* either using perception modules or by having users provide initialization of those attributes. However, such approaches either pre-define these attributes or pre-define the world model governing the dynamics of the objects in the video, limiting their applicability.

In this work, we mitigate the aforementioned challenges by developing a novel, physics-grounded, neuro-symbolic, program synthesis-based video prediction model that leverages the inductive biases of Large Language/Vision-Language Models (LLM/VLM) for improved sample efficiency, generalizability, and interpretability. Additionally, our formulation permits user-guided editability and counter-factual reasoning on the video frames which are to be generated as well. At its core, our proposed method called ProgGen, represents the dynamics of a video using a set of neuro-symbolic states, each defined by a set of human-interpretable, physics-grounded attributes, such as position or velocity of an object in the video. To enable such a representation, ProgGen employs a novel, programmatic perception pipeline (denoted by $\mathcal{P}$) to estimate these states from the visual context of a video and another program (denoted by $\mathcal{D}$) to learn the dynamics, so as to be able to extend the state estimates into the future, using a VLM. A programmatic rendering pipeline (denoted by $\mathcal{R}$), also generated by a VLM, then enables ProgGen to decode these predicted states into visually plausible RGB-frames. We leverage pre-trained deep-neural segmentation and tracking models (such as Grounded-Segment Anything (Ren et al., 2024)) to facilitate the perception process needed to estimate the states and adopt a symbolic, physical simulator (Feronato, 2012) to simulate the objects' dynamics in the pixel space. Moreover, the human-interpretable nature of the estimated latent states, lends them to human manipulation allowing for editing or counter-factual reasoning on the videos. Figure 1 presents an overview of the ProgGen pipeline.

In order to empirically validate the efficacy of ProgGen, we present empirical evaluations on synthetic environments – PhyWorld Kang et al. (2024) and Cart Pole gym. Aided by the inductive biases of the LLM/VLM, our method learns a customized video prediction model with no more than 10 training samples and performs competitively to the state-of-the-art diffusion models that are trained with upto three million samples. Moreover, our model generalizes much better to out-of-distribution inputs than the diffusion models can.

In summary, the contributions of our work, are as follows:

- We develop a novel, programmatic video prediction model, called ProgGen, leveraging the inductive biases of LLM/VLM in order to do away with the need for large-scale training data.
- ProgGen undertakes the video prediction task by representing the dynamics of the video using a set of neuro-symbolic, human-interpretable set of states.
- Empirical evaluations reveal that ProgGen outperforms competing approaches at the task of video frame prediction in two challenging environments: (i) PhyWorld (ii) Cart Pole.
- We show that ProgGen permits counter-factual reasoning, editability, and interpretable video generation attesting to its effectiveness and generalizability for video prediction tasks.

2 Related Works

End-to-end Models for Video Prediction: Recent years have seen an explosion of works that use end-to-end deep learning-based methods for video frame prediction in unconstrained environments, with or without text prompts, trained on datasets with thousands of videos Lu et al. (2023); Nikankin et al. (2022); Sun et al. (2023); Chatterjee et al. (2021); Yu et al. (2023); He et al. (2022); Ho et al. (2022). Most of these models adopt a diffusion model-based pipeline. Some prominent examples include methods like Stable Video Diffusion Blattmann et al. (2023), RaMViD Höppe et al. (2022), and TI2V-Zero Ni et al. (2024) that allow feeding a few initial frames of the video to the diffusion model in order to generate plausible future frames of the videos. These models currently tend to generate very short videos ($\sim$ 1 minute long), before losing temporal consistency and do not allow for easy and interpretable control over the video generation process. Furthermore, they do not implicitly

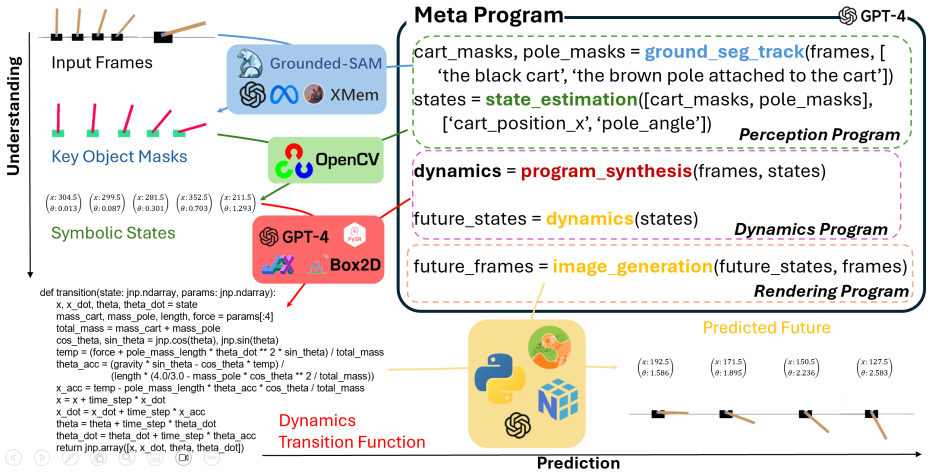

Figure 1: Overview of ProgGen showing the meta program that solves the video prediction task by: (i) generating a program for perception and estimation of object states ($\mathcal{P}$), (ii) estimating the world model (*i.e.*the dynamics of the video) program synthetically ($\mathcal{D}$), and (iii) rendering future frames using a rendering program ($\mathcal{R}$). The figure also illustrates the output of these steps for the Cart Pole setting, including the principal objects that have been discovered (such as "the black cart"), their states, the generated program for estimating the dynamics and the predicted frames of the video.

learn the physics of the world accurately, as shown by Kang *et al.* Kang et al. (2024) where the diffusion models are unable to learn world models even in very simple settings. Our proposed method attempts to address these challenges by adopting a program synthetic approach for the task, representing the dynamics of the video by a set of neuro-symbolic, human interpretable states.

Program Generation and Physics-based Approaches for Video Prediction: Unlike end-to-end methods, program generation and physics-based methods adopt a neuro-symbolic approach in order to learn the dynamics of a system.In the case of Galileo Wu et al. (2015), a Markov-Chain Monte Carlo (MCMC)-based sampling method is adopted to infer a set of physical properties of the objects in the video, such as mass, position, friction, *etc.* A 3D physics engine is then used to simulate the future frames of a video using these estimated quantities. Differently, our proposed method, uses a program synthetic approach to infer the physical properties by leveraging LLM/VLMs. DreamCoder Ellis et al. (2023) uses a wake-sleep algorithm to synthesize programs from a small set of input examples and has been shown to learn natural concepts that are applicable in a variety of applications including drawing simple graphics. Similar to our work, WorldCoder Tang et al. (2024) uses an LLM to generate programs that can be used to simulate the environment. However, these approaches only work with environments with discrete states and is not directly applicable to cases like ours where continuous parameters (like object properties) have to be estimated. PhysGen Liu et al. (2024) takes an image and attributes (like force/torque) as input, to generate the dynamics of objects by using a strong perception module and a physics engine. However, this method only deals with rigid bodies and the world model is not learned by observing videos, unlike in our work. Concurrent to our work, LLMPhy Cherian et al. (2024) was proposed which also uses an LLM to generate code in order to predict the physical properties of the objects in a video. However, what these properties are, is pre-determined and so is the nature of motion (such as quadratic, linear, *etc.*), while in our case both the attributes and the dynamics are automatically determined.Symbolic reasoning (SR) methods like PySR Cranmer (2023) that use evolutionary optimization and LLM-SR Shojaee et al. (2024) learn dynamical equations from data. However, unlike these approaches, ProgGen provides a complete, program-synthetic pipeline that incorporates perception, dynamics, and generation into a single interpretable system.

3 PRELIMINARIES

A video $V = \{f_0, f_1, f_2, \ldots, f_T\}$ with $T+1$ frames, can be thought of as a collection of $T+1$ random variables f_i, each denoting a RGB frame of the video, *i.e.*$f_i \in \mathbb{R}^{H \times W \times 3}$, where H and

W denote the height and width of each frame. Given a set of $F+1$ initial frames of the video ($0 \leq F < T$), to set the visual context, the task of video frame prediction entails forecasting the remaining frames, f_{F+1} onwards through f_T. Addressing this task requires designing a model p_θ, parameterized by a set of parameters θ, which takes as input the first $F+1$ frames of an unseen video and predicts the remaining frames. This can be represented as:

$$\{f_i\}_{i=F+1}^{T} = p_\theta(\{f_j\}_{j=0}^{F}; \{\psi_j\}_{j=0}^{F}),$$

where ψ_j denotes a possible latent state of the model (one for every predicted frame), in case its design leverages such a formulation (such as in the case of recurrent neural models).

4 Proposed Method

Typically, the aforementioned predictive models are instantiated via parameter-intensive deep neural network-based models Blattmann et al. (2023); Ni et al. (2024), which are plagued by two key challenges: (i) such models often require large volumes of in-domain training data and (ii) the lack of interpretability of the prediction process. In order to address these key challenges, in this work, we adopt a physics-grounded, neuro-symbolic approach for the task of video frame prediction. In broad strokes, we train our proposed approach called ProgGen, to estimate a set of neuro-symbolic states (each with its own physics-grounded, human-interpretable attributes such as position of an object, its velocity, *etc.*), which best explains the phenomena observed in frames f_{F+1} through f_T, while maintaining consistency with the frames f_0 through f_F. This process is realized by estimating three Python programs, derived from a large Vision-Language Model (VLM), like GPT-4 Achiam et al. (2023). The first of these is called the *Perception Program* ($\mathcal{P}$), which maps any RGB-frame $\{f_i\}_{i=0}^{T}$ to a state ($\{s_i\}_{i=0}^{T}$). A state is a collection of physics-grounded, human-interpretable attributes, such as position of an object, angle between an object's parts, *etc.* $\mathcal{P}$ automatically determines which attributes are the most applicable for a given video. The second program, called the *Dynamics Estimation Program* ($\mathcal{D}$), seeks to forecast future physics-grounded states given a past set of states. The final program, called the *Rendering Program* ($\mathcal{R}$), maps these states ($\{s_i\}_{i=0}^{T}$) back to the RGB-frames. The inference pipeline of ProgGen operates by having $\mathcal{P}$ derive the state-representation of the seen frames in a video ($\{f_i\}_{i=0}^{F}$), followed by $\mathcal{D}$ forecasting the set of neuro-symbolic states corresponding to the unseen frames ($\{s_i\}_{i=F+1}^{T}$), given the state-representation of the seen frames in the video ($\{s_i\}_{i=0}^{F}$). Finally $\mathcal{R}$ decodes the predicted states, corresponding to the unseen frames, in order to synthesize the predictions for the corresponding RGB-frames ($\{\hat{f}_i\}_{i=F+1}^{T}$). Algorithm 1 presents an outline of the inference regime of ProgGen while Figure 1 presents an overview.

4.1 Neuro-Symbolic Video Understanding and Prediction

At a high-level ProgGen, subsumes two tasks: (i) *Video Understanding:* Entails estimating the neuro-symbolic states, corresponding to the seen frames, f_0 through f_F of a video, using $\mathcal{P}$, such that when these states are decoded into RGB-frames, using $\mathcal{R}$, they maximize the likelihood of the seen frames. (ii) *Video Prediction:* Wherein, $\mathcal{D}$ and a handful of trainable parameters θ, are used to learn the video dynamics and forecast the states corresponding to the unseen frames, f_{F+1} through f_T of a video. This is followed by the use of $\mathcal{R}$ to decode these states to RGB-frames. During training, both the video understanding and video prediction steps can leverage all $T+1$ frames of a video in the training set to determine the $\mathcal{P}$, $\mathcal{D}$, θ, and $\mathcal{R}$ which best explain the said videos. For inference on any video, rather than directly predicting the unseen frames, $f_{F+1:T}$, given the seen frames $f_{0:F}$, as is common in conventional encoder-decoder based frame prediction approaches Denton & Fergus (2018); Chatterjee et al. (2021), we instead use $\mathcal{P}$ to first encode the seen frames into states, and then use $\mathcal{D}$, θ, and $\mathcal{R}$, to predict and render the unseen frames. Mathematically, the perception step can be represented by the following notation:

$$s_i \sim p\left(\cdot | f_i; \mathcal{P}\right); \forall i \in \{0, \ldots, F\}$$

While the prediction step can be described, using the following two-step procedure:

$$s_i \sim h\left(\cdot | s_{i-1}; \mathcal{D}, \theta\right); \forall i \in \{F+1, \ldots, T\}$$

$$f_i \sim r\left(\cdot | s_i; \mathcal{R}\right); \forall i \in \{F+1, \ldots, T\}$$

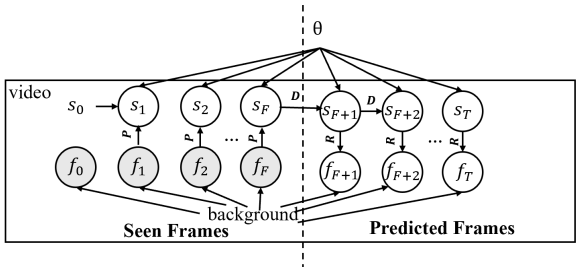

Figure 2: Plate diagram depicting the graphical model of ProgGen, during inference. The first $F + 1$ frames are used to set the visual conditioning while the subsequent frames are predicted.

4.2 Programs and States

Unlike conventional encoder-decoder based approaches Ni et al. (2024); Blattmann et al. (2023), the latent states (s_i) in ProgGen are symbolic, human-interpretable, and carry semantic meaning. In particular, they are described by physics-based, human-interpretable attributes, such as *position* of an object, its *velocity*, *etc.* The design of ProgGen, allows for $\mathcal{P}$ to figure out what attributes might be the most fitting to represent the states. Such a capability allows for much more flexibility of representation, thereby enhancing the generalizability of ProgGen. $\mathcal{P}$ uses state-of-the-art visual reasoning tools like Grounded-SAM Ren et al. (2024) for detecting and localizing the objects in the video and XMem Cheng & Schwing (2022) for ensuring that the detections are consistent over the frames. Likewise, $\mathcal{D}$ and θ allow for flexibility in representing the dynamics of the video, with θ capturing the global parameters in the environment, such as the gravity.

Figure 2 presents a plate diagram of our proposed framework. Each frame f_i of a video is derived from its underlying latent state s_i, which in turn is influenced by the previous states $s_{0:i-1}$. For completeness, we also assume a random variable, $background$, to be capturing the details of the static background environment of the video.[2] The program $\mathcal{P}$ captures a disentangled structure of the scene in a video, such as how many objects there are, if two objects are connected, if an object is dynamic or static, *etc.* The dynamics of a video are captured by the evolution of the states, following the rules specified by the program, $\mathcal{D}$ and the global parameters, θ. In particular, $\mathcal{D}$ lays out the laws of physics that principally explain the dynamics in the video, such as laws of inertia, laws of conservation of energy and momentum *etc.* The program $\mathcal{R}$ is responsible for mapping the states of the predicted frames to the RGB-domain, using a physics-based simulator.

We seek to leverage the effective program generation capacity of large Language/Large Vision-Language Models (LLM/VLM) Achiam et al. (2023) to propose hypotheses for $\mathcal{P}$, $\mathcal{D}$, and $\mathcal{R}$, given the frames of a training video, $f_{0:T}$. To aid the VLM in this search, we design Affordance Rules (ARLs) which describe certain constraints governing the dynamics of the environment to aid the simulation of the dynamics in the pixel space. For instance, the ARL for a rigid body motion environment setting could be that affine transforms capture the motion of objects in the RGB-frame. This significantly reduces the number of candidate hypotheses programs which fit the training data, thereby making the searching task by the LLM/VLM, more efficient.

4.3 Training ProgGen

Training ProgGen entails estimating the programs $\mathcal{P}$, $\mathcal{D}$, and $\mathcal{R}$, as well as a handful of global parameters θ in order to maximize the likelihood of the frames in each of the N training videos. For each video, this can be captured by the following probabilistic formulation:

$$\mathcal{L}(f_{0:T}; \mathcal{P}, \mathcal{D}, \theta, \mathcal{R}) := p(f_{0:T}|s_0; \mathcal{P}, \mathcal{D}, \theta, \mathcal{R})$$
$$= \int_{s_{0:T}} p(f_T|f_{0:T-1}, s_0; \mathcal{P}, \mathcal{D}, \theta, \mathcal{R}) p(f_{0:T-1}|s_0; \mathcal{P}, \mathcal{D}, \theta, \mathcal{R}) \mathrm{d}s_t$$

[2]For the sake of brevity, we ignore this random variable, which can be introduced into the formulation without loss of generalizability.

$$= \int_{s_{0:T}} p(f_t|s_t)p(s_t|s_{t-1})p(s_{t-1}|f_{0:t-1})p(f_{0:t-1}|s_0;\mathcal{P},\mathcal{D},\theta,\mathcal{R})\mathrm{d}s_t$$

$$= \int_{s_{1:T}} \prod_{t=1}^{T} r(f_t|s_t;\mathcal{R})h(s_t|s_{t-1};\mathcal{D},\theta)p(s_{t-1}|f_{t-1};\mathcal{P})\mathrm{d}s_t$$

The training objective may then be represented as follows:

$$\mathcal{P},\mathcal{D},\theta,\mathcal{R} = \arg\max_{\widetilde{\mathcal{P}},\widetilde{\mathcal{D}},\widetilde{\theta},\widetilde{\mathcal{R}}} \mathcal{L}(f_{0:T};\widetilde{\mathcal{P}},\widetilde{\mathcal{D}},\widetilde{\theta},\widetilde{\mathcal{R}}) \tag{1}$$

In order to aid interpretability and for ease of design, we assume $r(f_t|s_t;\mathcal{R}) \sim \mathcal{N}(f_t,\Sigma)$, where Σ is an isotropic constant. Since maximizing the likelihood is equivalent to maximizing its logarithmic variant, this results in the following optimization setup, over the training set:

$$\log \mathcal{L}(f_{0:T};\mathcal{P},\mathcal{D},\theta,\mathcal{R}) \propto \sum_{i=1}^{N}\sum_{t=0}^{T} ||\hat{f}_t^i - f_t^i||^2, \tag{2}$$

where $\hat{f}_t^i$ denotes the estimate of the t^{th}-frame by ProgGen of the i^{th} video while f_t denotes the true frame in the training video.

Two-stage Training Optimization: Foundation models such as large language models (LLMs) or visual language models (VLMs) have shown impressive abilities to synthesize programs Tang et al. (2024) and analyze the discrete structures of the scene Ren et al. (2024). However, they are not as effective in estimating continuous parameters Tang et al. (2024). We thus introduce a two-stage pipeline to train ProgGen. In the first stage, the programs, $\mathcal{P}$, $\mathcal{D}$, and $\mathcal{R}$ are estimated, while the trainable continuous parameters θ are estimated in the next. To enable the learning of the continuous parameters, the LLMs are prompted to synthesize programs with placeholders for the continuous parameters, θ: $prog \sim q_{\text{LLM}}(\cdot|f_{0:T})$. The continuous parameters are then learned by maximizing the likelihood of the seen frames of a given video, as shown in Eq. 1. For non-differentiable programs, we use black-box optimizers such as the Powell's method Powell (2006) to learn the parameters. If the Affordance Rules requires LLMs/VLMs to generate differentiable programs, e.g., using JAX, we then adopt higher-order optimizers such as L-BFGS Fei et al. (2014) for more efficient optimization. These optimizations are executed till convergence or until a maximum number of iterations, whichever occurs earlier.

Surrogate Loss for Efficient Optimization: The typical pixel-level training loss, shown in Eq. 2, while being differentiable and easy to optimize is not sensitive to the accuracy of the reconstructed frames. For instance, if in a scene a large, white background occupies most of the pixels, Eq. 2 yields a low loss even if a completely blank frame is predicted. Moreover, the computational cost to run $\mathcal{R}$ for every frame in the training video is expensive. As a fix to this, we independently try to achieve estimates for the attributes listed by ProgGen, for defining the state of the video. Such attributes could include positions of objects, angles, *etc.* and could be independently estimated using tools such as Segment Anything Model (SAM) Ren et al. (2024) and appropriate functions from OpenCV Bradski et al. (2000). Along the lines of Eq. 2, the likelihood estimation task for each training set video, then becomes:

$$\mathcal{L}(s_{0:T};\mathcal{P},\mathcal{D},\theta) := q(s_{0:T}|f_{0:T};\mathcal{P},\mathcal{D},\theta) \tag{3}$$

$$\propto \sum_{t=0}^{T} ||\hat{s}_t - s_t||^2, \tag{4}$$

where $\hat{s}_t$ denotes the estimate of the t^{th}-state by ProgGen while s_t denotes the true state value.

Algorithm 1 Inference with ProgGen

Require: Video $f_{0:F}$, s_0, $\mathcal{P}$, $\mathcal{D}$, θ, $\mathcal{R}$
Ensure: Generated video $f_{F+1:T}$
1: **for** $t = 1$ to F **do**
2: $\quad s_t \leftarrow \mathcal{P}(f_t, \theta)$
3: **end for**
4: **for** $t = F + 1$ to T **do**
5: $\quad s_t \leftarrow \mathcal{D}(s_{t-1}), \theta$
6: $\quad f_t \leftarrow \mathcal{R}(s_t)$
7: **end for**
8: **return** $f_{F+1:T}$

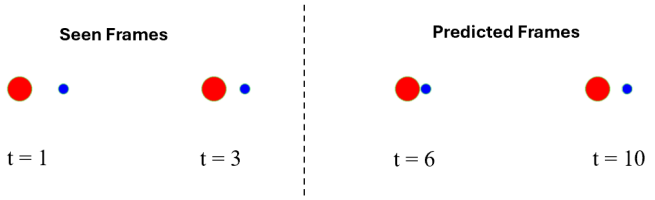

Figure 3: Qualitative visualization of frames predicted by our method for the out of domain, two ball collision case.

5 Experiments

5.1 Experimental Setup

Environments: We conduct experiments in the following two challenging simulation environments:

(a) Phyworld-OOD: The Phyworld environment Kang et al. (2024) simulates the kinematic states for various physical scenarios, such as with balls moving in a confined space and/or colliding with each other. We conduct experiments with videos collected from this environment under two settings: (i) *Uniform:* Here there is only one ball that moves horizontally with a constant velocity in the space. The law of inertia guides the motion of the ball; (ii) *Collision:* Here there are two balls, of different sizes (*i.e.* radii), moving horizontally at different velocities and colliding with each other. The law of conservation of energy and momentum guides the collision process and the velocities after the collision. We generate videos under the aforementioned settings by varying the radii and the velocities of the balls. Our programmatic video prediction method works by estimating the positions of the ball to define the states. The videos contain a total of 20 frames, with the first 3 frames being used for conditioning (*i.e.* seen frames). To assess the generalizability of competing approaches, we conduct evaluations under two settings, per prior work Kang et al. (2024): (i) *In-distribution*: Here the videos in the test set have ball velocities similar to those in the training; (ii) *Out-of-distribution*: Here the test set videos contain ball velocities much higher than those in the training set. We train all baseline models with varying training set sizes, viz. 30k and 3M videos under the aforementioned settings. For ProgGen, on the other hand, we use only 10 videos to train for the parameters, θ.

(b) Cart Pole: We also conduct experiments in the gym-Cart Pole gym environment, with a constant action – pulling the cart to the left, while the pole rotates in a clock-wise direction. The videos are generated by altering the pole's initial angle, the velocity of the cart, and the angular velocity of the pole. A programmatic approach to video frame prediction needs to automatically determine that these are the attributes to be estimated and undertake the prediction by extrapolating the dynamics using these attributes. In this setting, we test the generalizability of competing approaches by adopting a zero-shot setting where there are no training videos and the models are asked to generate the next ten frames of a video, given the first 10 frames.

Baselines: In order to assess the benefits of ProgGen at the task of video frame prediction, we choose the following competitive, state-of-the-art baselines drawn from two broad streams of research in this field: (i) From among the data-driven video frame prediction type of models, we use diffusion-

Model	Training dataset size	uniform-motion-iid (↓)	uniform-motion-ood (↓)	collision-iid (↓)	collision-ood (↓)
DiT-small	30K	0.0221	0.4349	0.0267	0.1873
DiT-big	30K	0.0166	0.4330	0.0302	0.2411
DiT-large	30K	0.0150	0.3783	0.0240	0.2700
DiT-small	3M	0.0149	0.2875	0.0227	0.1525
DiT-big	3M	<u>0.0138</u>	0.3583	<u>0.0181</u>	0.2106
DiT-large	3M	**0.0124**	0.4270	**0.0153**	0.1613
Galileo Wu et al. (2015)	30K	0.0176	0.0187	0.0502	0.0544
Galileo Wu et al. (2015)	3M	0.0163	<u>0.0173</u>	0.0342	<u>0.0428</u>
ProgGen (Ours)	10	0.0147	**0.0150**	0.0227	**0.0241**

Table 1: Velocity prediction accuracy for the Phyworld Uniform Ball Motion and Ball Collision settings for both in (iid) and out of distribution (ood) settings. Best results are shown in **bold**, second best in <u>underline</u>.

Model	MAE (↓)	PSNR (↑)	LPIPS (↓) (AlexNet)	LPIPS (↓) (VGGNet)
TI2V-Zero Ni et al. (2024)	0.011	22.01	5.9e-5	0.0038
Stable Video Diffusion-v1 Blattmann et al. (2023)	0.033	18.37	2.0e-4	0.0046
ProgGen (Ours)	**0.003**	**30.59**	**1.8e-5**	**2.8e-4**

Table 2: Frame prediction performance of competing methods on the Cart Pole environment. Best results shown in **bold**.

model based methods, such as TI2V-Zero Ni et al. (2024), Stable Video Diffusion-v1 Blattmann et al. (2023) for the CartPole experiments and following Kang *et al.* Kang et al. (2024), use DiT Kang et al. (2024) (which follows the same architecture as Sora Brooks et al. (2024)) for our experiments on PhyWorld. For DiT, we use the three variants for the PhyWorld dataset, as reported in Kang *et al.* Kang et al. (2024): the DiT-small, DiT-big, and DiT-large, with 22.5M, 89.5M, and 310M parameters, respectively. For TI2V-Zero and Stable Diffusion, we use their pre-trained models available from the web. (ii) From among the physics-based frame prediction approaches, we choose Galileo Wu et al. (2015) for our experiments on PhyWorld. Since the Cart Pole setting does not use any training data, we ignore this baseline for the Cart Pole experiments.

Evaluation Metrics: For <u>Phyworld</u>, Kang *et al.* Kang et al. (2024) uses an error measure based on the estimated velocities of the one or more balls, in every frame, when compared to that used for the synthesis of the videos to assess the quality of prediction. The per-frame velocity is computed as:

$$v_t^i = x_t^i - x_{t-1}^i; \forall t \in \{F+1, \ldots, T\}, \forall i \in \{1, \ldots, B\}$$

where v_t^i is the velocity of the i^{th} ball in the t^{th} frame, while x_t^i denotes the pixel location of the center of the ball in the t^{th} frame and B is the total number of balls in the environment. The final performance measure is an average of this error over all the balls. For our experiments, we stick to this evaluation measure as well.

<u>Cart Pole:</u> We evaluate the qualities of the generated videos in the Cart Pole environment by comparing them with the ground-truth videos using metrics that assess the quality of reconstructed RGB-frames and reporting the mean performance, including mean-absolute error (MAE), PSNR Channappayya et al. (2008), and LPIPS Zhang et al. (2018).

5.2 RESULTS

PhyWorld: As shown in Table 1, our model significantly outperforms the state-of-the-art diffusion models (*i.e.*DIT-small, DIT-big, DIT-large), trained on 30k training videos, even though our method is only trained on 10 videos. In particular, the velocity error measures for ProgGen are an order of magnitude smaller for the out-of-distribution (ood) case when compared to the diffusion-based baseline models. Further, even when the diffusion models are trained with $100\times$ the number of training videos, *i.e.*3M, our proposed method still performs comparably. Moreover, we observe that

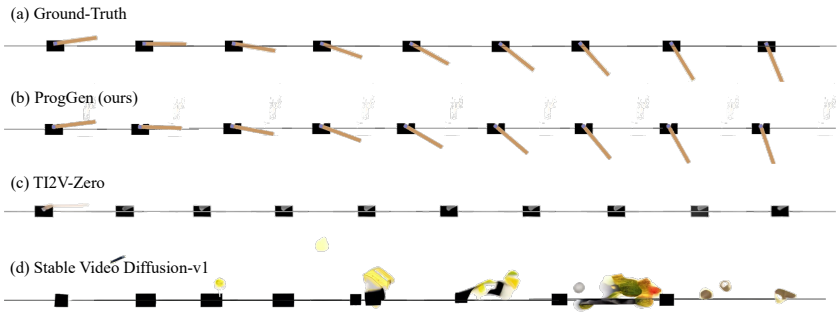

Figure 4: Qualitative visualization of frame prediction results between competing methods on the Cart Pole environment.

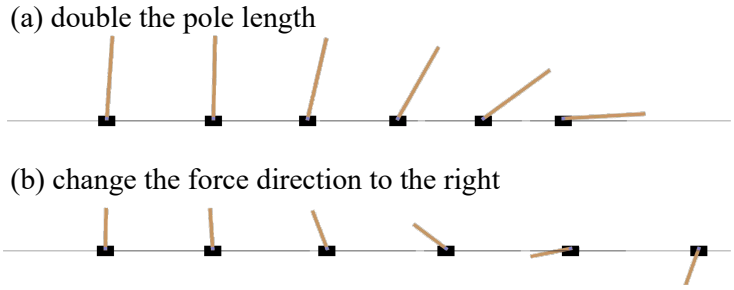

Figure 5: Frame prediction results for our method upon: (a) doubling the pole length; (b) switching the direction of motion the cart.

even physics-grounded models such as Galileo Wu et al. (2015), underperform ProgGen under all settings. This is perhaps because of the constraints imposed by Galileo in assuming a Gaussian distribution on the state transition function. We surmise that the flexibility obtained by modeling the video prediction task programmatically helps in capturing the dynamics of the video at the semantic level, *i.e.* in describing the world model in terms of objects and their motion rather than in terms of changes at the pixel level. Additionally, the global parameters θ accommodate for variance within the type of motion structure captured in the programs allowing ProgGen to fit to a wide range of videos.

Figure 3 visualizes the frames generated by our method for the two ball, out-of-distribution collision case. We see that our method is successful at following the physical laws as specified by the programs and can generate visually plausible frames. We also provide ablation results for training the DIT models with 300k training videos in the appendix.

Cart Pole: To assess the generalizability of ProgGen, we evaluate our method against competing baselines on the gym-Cart Pole environment. As shown in Table 2, our method outperforms all competing baselines, across all measures, by significant margins in this environment. This is evident from the qualitative results in Figure 4, as well. Furthermore, as shown in Figure 5, ProgGen can accommodate counter-factual reasoning effectively, such as if the pole length were to be doubled or the direction of motion of the cart were to be reversed, and still generate plausible frames.

6 Conclusions

In this work, we propose ProgGen, a program synthetic approach to video frame prediction which relies on the inductive biases of LLM/VLM to construct world models of videos, represented as a sequence of neuro-symbolic, human-interpretable states. Our proposed approach, while still largely modeling the video in an object-centric fashion, can outperform state of the art diffusion models for the task across multiple challenging environments, while utilizing only a fraction of the training data.

Limitations: Adapting ProgGen to work in more real world settings might require a richer set of attribute descriptions which might be challenging for current LLM/VLM to estimate.

REFERENCES

Open ai gym. https://gymnasium.farama.org/index.html. Accessed: 2025-02-10.

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Mohammad Babaeizadeh, Chelsea Finn, Dumitru Erhan, Roy H Campbell, and Sergey Levine. Stochastic variational video prediction. *arXiv preprint arXiv:1710.11252*, 2017.

Andreas Blattmann, Tim Dockhorn, Sumith Kulal, Daniel Mendelevitch, Maciej Kilian, Dominik Lorenz, Yam Levi, Zion English, Vikram Voleti, Adam Letts, et al. Stable video diffusion: Scaling latent video diffusion models to large datasets. *arXiv preprint arXiv:2311.15127*, 2023.

Gary Bradski, Adrian Kaehler, et al. Opencv. *Dr. Dobb's journal of software tools*, 3(2), 2000.

Tim Brooks, Bill Peebles, Connor Holmes, Will DePue, Yufei Guo, Li Jing, David Schnurr, Joe Taylor, Troy Luhman, Eric Luhman, Clarence Ng, Ricky Wang, and Aditya Ramesh. Video generation models as world simulators. 2024. URL https://openai.com/research/video-generation-models-as-world-simulators.

Sumohana S Channappayya, Alan Conrad Bovik, and Robert W Heath. Rate bounds on ssim index of quantized images. *IEEE Transactions on Image Processing*, 17(9):1624–1639, 2008.

Moitreya Chatterjee, Narendra Ahuja, and Anoop Cherian. A hierarchical variational neural uncertainty model for stochastic video prediction. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 9751–9761, 2021.

Ho Kei Cheng and Alexander G Schwing. Xmem: Long-term video object segmentation with an atkinson-shiffrin memory model. In *European Conference on Computer Vision*, pp. 640–658. Springer, 2022.

Anoop Cherian, Radu Corcodel, Siddarth Jain, and Diego Romeres. Llmphy: Complex physical reasoning using large language models and world models. *arXiv preprint arXiv:2411.08027*, 2024.

Miles Cranmer. Interpretable machine learning for science with pysr and symbolicregression. jl. *arXiv preprint arXiv:2305.01582*, 2023.

Emily Denton and Rob Fergus. Stochastic video generation with a learned prior. In *International conference on machine learning*, pp. 1174–1183. PMLR, 2018.

Kevin Ellis, Lionel Wong, Maxwell Nye, Mathias Sable-Meyer, Luc Cary, Lore Anaya Pozo, Luke Hewitt, Armando Solar-Lezama, and Joshua B Tenenbaum. Dreamcoder: growing generalizable, interpretable knowledge with wake–sleep bayesian program learning. *Philosophical Transactions of the Royal Society A*, 381(2251):20220050, 2023.

Yun Fei, Guodong Rong, Bin Wang, and Wenping Wang. Parallel l-bfgs-b algorithm on gpu. *Computers & graphics*, 40:1–9, 2014.

Emanuele Feronato. *Box2d for Flash Games*. Packt Publishing Ltd, 2012.

Yingqing He, Tianyu Yang, Yong Zhang, Ying Shan, and Qifeng Chen. Latent video diffusion models for high-fidelity video generation with arbitrary lengths. *arXiv preprint arXiv:2211.13221*, 2(3):4, 2022.

Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J Fleet. Video diffusion models. *Advances in Neural Information Processing Systems*, 35:8633–8646, 2022.

Tobias Höppe, Arash Mehrjou, Stefan Bauer, Didrik Nielsen, and Andrea Dittadi. Diffusion models for video prediction and infilling. *arXiv preprint arXiv:2206.07696*, 2022.

Ashesh Jain, Hema S Koppula, Bharad Raghavan, Shane Soh, and Ashutosh Saxena. Car that knows before you do: Anticipating maneuvers via learning temporal driving models. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3182–3190, 2015.

Bingyi Kang, Yang Yue, Rui Lu, Zhijie Lin, Yang Zhao, Kaixin Wang, Gao Huang, and Jiashi Feng. How far is video generation from world model: A physical law perspective. *arXiv preprint arXiv:2411.02385*, 2024.

Shaowei Liu, Zhongzheng Ren, Saurabh Gupta, and Shenlong Wang. Physgen: Rigid-body physics-grounded image-to-video generation. In *European Conference on Computer Vision*, pp. 360–378. Springer, 2024.

Haoyu Lu, Guoxing Yang, Nanyi Fei, Yuqi Huo, Zhiwu Lu, Ping Luo, and Mingyu Ding. Vdt: An empirical study on video diffusion with transformers. *arXiv preprint arXiv:2305.13311*, 3(5):9, 2023.

Haomiao Ni, Bernhard Egger, Suhas Lohit, Anoop Cherian, Ye Wang, Toshiaki Koike-Akino, Sharon X Huang, and Tim K Marks. Ti2v-zero: Zero-shot image conditioning for text-to-video diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9015–9025, 2024.

Yaniv Nikankin, Niv Haim, and Michal Irani. Sinfusion: Training diffusion models on a single image or video. *arXiv preprint arXiv:2211.11743*, 2022.

Michael JD Powell. A fast algorithm for nonlinearly constrained optimization calculations. In *Numerical Analysis: Proceedings of the Biennial Conference Held at Dundee, June 28–July 1, 1977*, pp. 144–157. Springer, 2006.

Tianhe Ren, Shilong Liu, Ailing Zeng, Jing Lin, Kunchang Li, He Cao, Jiayu Chen, Xinyu Huang, Yukang Chen, Feng Yan, et al. Grounded sam: Assembling open-world models for diverse visual tasks. *arXiv preprint arXiv:2401.14159*, 2024.

Parshin Shojaee, Kazem Meidani, Shashank Gupta, Amir Barati Farimani, and Chandan K Reddy. Llm-sr: Scientific equation discovery via programming with large language models. *arXiv preprint arXiv:2404.18400*, 2024.

Mingzhen Sun, Weining Wang, Xinxin Zhu, and Jing Liu. Moso: Decomposing motion, scene and object for video prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 18727–18737, 2023.

Hao Tang, Darren Key, and Kevin Ellis. Worldcoder, a model-based llm agent: Building world models by writing code and interacting with the environment. *arXiv preprint arXiv:2402.12275*, 2024.

Gang Wang, Bo Li, Yongfei Zhang, and Jinhui Yang. Background modeling and referencing for moving cameras-captured surveillance video coding in hevc. *IEEE Transactions on Multimedia*, 20(11):2921–2934, 2018.

Jiajun Wu, Ilker Yildirim, Joseph J Lim, Bill Freeman, and Josh Tenenbaum. Galileo: Perceiving physical object properties by integrating a physics engine with deep learning. *Advances in neural information processing systems*, 28, 2015.

Sihyun Yu, Kihyuk Sohn, Subin Kim, and Jinwoo Shin. Video probabilistic diffusion models in projected latent space. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 18456–18466, 2023.

Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 586–595, 2018.

7 APPENDIX

7.1 MODEL PERFORMANCE WITH VARYING TRAINING SET SIZES

Model	Training dataset size	uniform-motion-iid (↓)	uniform-motion-ood (↓)	collision-iid (↓)	collision-ood (↓)
DiT-small	30K	0.0221	0.4349	0.0267	0.1873
DiT-big	30K	0.0166	0.4330	0.0302	0.2411
DiT-large	30K	0.0150	0.3783	0.0240	0.2700
DiT-small	300K	0.0184	0.2929	0.0241	0.2000
DiT-big	300K	0.0166	0.4330	0.0193	0.1663
DiT-large	300K	0.0140	0.3847	0.0158	0.1725
DiT-small	3M	0.0149	0.2875	0.0227	0.1525
DiT-big	3M	0.0138	0.3583	0.0181	0.2106
DiT-large	3M	0.0124	0.4270	0.0153	0.1613
Galileo Wu et al. (2015)	30K	0.0176	0.0187	0.0502	0.0544
Galileo Wu et al. (2015)	3M	0.0163	0.0173	0.0342	0.0428
Ours	1	0.0192	0.0185	0.4805	0.2879
Ours	10	0.0187	0.0176	0.0385	0.0418
Ours	100	0.0175	0.0164	0.0340	0.0321

Table 3: Velocity prediction accuracy for the Phyworld Uniform Ball Motion and Ball Collision settings for both in (iid) and out of distribution (ood) settings.

In Table 3, we report model performances with varying training set sizes for the PhyWorld Kang et al. (2024) environment. We train the competing baseline models with 30k, 300k, or 3M videos, while we train our proposed approach with merely 1, 10 or 100 training videos. As we see from the results, our proposed method (ProgGen) trained with 10 or 100 videos, outperforms all competing methods for the challenging, out of the distribution (ood) motion setting. Noticeably, our model trained with just 1 video is also competitive in the ood setting. Even for the in distribution setting (iid), our model is competitive with respect to the competing baselines.

7.2 ILLUSTRATIVE PROMPTS AND CODES

Below, we present as an illustrative example of the prompt used in order for ProgGen to generate the *Dynamics Estimation Program* $\mathcal{D}$, using a VLM, for the Cart Pole environment:

"The environment you're in is the classic CartPole environment. This is a well-known problem in reinforcement learning where a pole is attached by an un-actuated joint to a cart, which moves along a frictionless track. The goal is to keep the pole upright by moving the cart left or right. The state of the system is represented by a tuple consisting of: 1. cart_position: The position of the cart on the track. 2. cart_velocity: The velocity of the cart. 3. pole_angle: The angle of the pole with the vertical axis. 4. pole_angular_velocity: The angular velocity of the pole. Given this setup, the dynamics of the environment are governed by the physics of the system, specifically the equations of motion for the cart and the pole. The force applied to the cart (left or right) will affect these state variables over time. We'll use some constants (which will be optimized later) such as gravity, mass of the cart and pole, length of the pole, and the time step for the discretization of the dynamics."

The above prompt leads to the following synthesized program:

```
def transition(state: jnp.ndarray, params: jnp.ndarray):
    x, x_dot, theta, theta_dot = state
    mass_cart, mass_pole, length, force = params[:4]
    total_mass = mass_cart + mass_pole
    cos_theta, sin_theta = jnp.cos(theta), jnp.sin(theta)
    temp = (force + pole_mass_length * theta_dot ** 2 * sin_theta) / total_mass
    theta_acc = (gravity * sin_theta - cos_theta * temp) /
                        (length * (4.0/3.0 - mass_pole * cos_theta ** 2 / total_mass))
    x_acc = temp - pole_mass_length * theta_acc * cos_theta / total_mass
    x = x + time_step * x_dot
    x_dot = x_dot + time_step * x_acc
    theta = theta + time_step * theta_dot
    theta_dot = theta_dot + time_step * theta_acc
    return jnp.array([x, x_dot, theta, theta_dot])
```

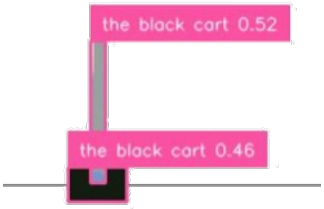

Figure 6: Grounded-SAM Ren et al. (2024) incorrectly labels the pole as ”the black cart”.

7.3 FAILURE CASES

Even though ProgGen achieves state-of-the-art frame prediction performance in our experiments, it does falter sometimes. In Figure 6, we show one such case. Here, the detections produced by Grounded SAM Ren et al. (2024) are incorrect. In particular, even the “pole” is detected as a “black cart” with high confidence. In order to correct for such erroneous detections, we leverage GPT-4v Achiam et al. (2023) to verify the detections in order to improve the effectiveness and reliability of the perception but we observe GPT-4v’s perception module to also be noisy at times resulting in inaccurate estimation of attributes, for such cases.