Action-Conditioned Visual Prediction for Robotic Manipulation

Anirudh Pai Ryan Cheng Junhua Ma

Abstract

We investigate the task of action-conditioned visual prediction in robotic manipulation: given a sequence of past RGB frames and associated low-level robot actions, can we predict the robot's future visual observations? This capability is crucial for model-based planning and sim-to-real transfer, enabling agents to reason over future states without environment interaction. Using the large-scale DROID dataset, we propose a generative flow matching model model that predicts future frames from a wrist-mounted camera viewpoint, conditioned on prior frames and joint-level actions. To urge the model to learn actions—rather than simply reproducing the last ground-truth frame in the context horizon—we explore several training strategies and evaluate their effectiveness through visual assessments of stability during auto-regressive rollout. In particular, we find that cascading UNets during training significantly improves inference-time stability. These findings highlight that modeling action-conditioned dynamics in the visual domain is feasible even in complex manipulation settings, paving the way for its integration into planning pipelines.

1. Introduction

The ability to reason about the consequences of actions is a hallmark of intelligent behavior—especially in robotics. While many recent advantages have focused on reasoning in the language space, action models must also learn to reason directly in the visual domain, enabling them to anticipate how their actions will affect the world around them to a greater degree of granularity. In this work, we investigate action-conditioned video prediction, the task of generating future observations autoregressively, conditioned on prior observations and actions. Such predictions would implicitly encode scene dynamics, object interactions, and contact events—without requiring explicit supervision.

However, modeling accurate and stable visual dynamics in robotic manipulation remains challenging. In real-world scenes, objects are diverse and oftentimes partially observable, leading to error compounding and collapse during long-horizon prediction often lead to drift or collapse.

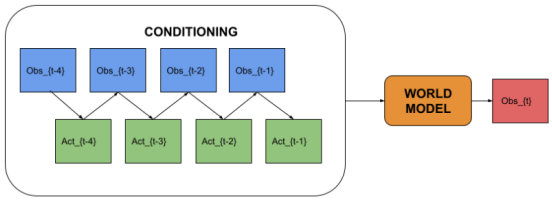

Figure 1. Overview of World Model Setup

Moreover, many existing approaches are developed and evaluated on synthetic datasets that target a narrow domain, limiting their generalization to real-world manipulation.

In this work, we attempt to address these gaps. More particularly, we use the DROID dataset, which provides thousands of Franka Emika episodes containing manipulation-heavy tasks. Each episode has synchronized high-resolution RGB observations, and joint-level action vectors, making this dataset ideal for our world modeling purposes. We develop a flow matching-based generative model (Figure 1) that predicts future camera observations conditioned on both visual context and robot joint velocities. Compared to diffusion models or other latent dynamics approaches, flow matching offers simpler and more stable training while still enabling flexible conditional generation.

Through extensive ablation, we propose *cascading U-Nets*, a method of training that facilitates more stability during inference. While our method does not yet constitute a complete world model for planning and control, it demonstrates a clear improvement over our traditional flow matching baseline, hinting that there is scope for more improvement in this direction.

2. Related Work

2.1. Latent World Models

Latent world models aim to learn compact internal representations of the environment dynamics to enable planning and control. A prominent example is **PlaNet** [3], which learns a stochastic latent dynamics model from images and performs planning in the latent space using model predictive control. Building on this, **Dreamer** [4] improves effi-

ciency by using imagined rollouts in the latent space to train both the value function and the policy, making it suitable for long-horizon reasoning and control from high-dimensional observations. However, both papers acknowledge the difficulty of achieving such planning with predictions made in the visual domain, which is an ongoing challenge today.

2.2. Pixel-Level World Models for Games

World models that operate directly in pixel space have also gained attention, especially in game environments. **SimVP** [2] proposes a simple yet effective fully convolutional architecture for video prediction, demonstrating strong performance on synthetic and natural video benchmarks.

More recently, **DIAMOND** [1] introduces a diffusion-based world model that preserves high-fidelity visual details and improves downstream policy learning. DIAMOND is applied to both 2D environments like Atari and more complex 3D games such as CS:GO. To train its generative model. DIAMOND leverages the Elucidated Diffusion Model (EDM) framework [6], which uses optimal noise schedules, data augmentations, and loss rebalancing to stabilize and improve diffusion training. This setup enables DIAMOND to model pixel-level dynamics with greater precision, resulting in more sample-efficient and visually accurate world predictions.

2.3. World Models for Robotic Manipulation

Though much success has been seen in applying such world models to synthetic environments (like those in video games), robot manipulation presents unique challenges that make building world models much trickier. Contact dynamics, occlusions, and partial observability limit action representations from fully describing observation changes. To combat this, **Masked World Models** [10] use masked autoencoders to separately learn visual representations and latent dynamics, achieving improved sample efficiency in robotic control tasks.

While not explicitly a world model, **Gato** [9] demonstrates that a single transformer architecture can perform diverse tasks—ranging from vision and language to robotic control—highlighting the potential of unified architectures for generalist agents.

3. Method

3.1. UNet Diffusion Models

Denoising Diffusion Probabilistic Models (DDPMs) [5] emerged in 2020 as a powerful class of generative models that synthesize high-quality images by iteratively denoising samples from a noise distribution. Given a data distribution $p_{\text{data}}(x)$, diffusion models define a forward noising process that gradually perturbs a data sample x_0 into a Gaussian noise x_T through a sequence of latent variables $\{x_t\}_{t=1}^{T}$. Specifically,

$$x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t}\epsilon$$

and $\bar{\alpha}_t$ is the cumulative product of noise schedule parameters. The reverse process learns to invert this corruption by modeling the conditional distributions $p_\theta(x_{t-1} \mid x_t)$ using a neural network.

For our setup, we adopt a UNet model for our denoising network $f_\theta(x_t, t)$, which predicts the original clean signal x_0 for any given latent variable x_t. The UNet model consists of an encoder-decoder structure with skip connections that preserve spatial detail across different resolution levels. This architecture has been shown to be effective in capturing both global structure and fine-grained local details, which allows for high-fidelity generation.

To condition our UNet based on actions and noise timestep t, we use a feature-based linear modulation (FiLM) setup formulated as

$$h' = FC(a) * h + FC(t)$$

given actions a, timesteps t, UNet hidden embedding h and fully-connected layer FC. To condition our UNet with multiple past frames, we simply stack the frames and concatenate with the input noisy image x_t along the channel axis.

To improve the generated results, we also apply classifier-free guidance (CFG) during sampling which generates final prediction u with both a conditional prediction u_{cond} and unconditional prediction u_{uncond} based on the formulation

$$u = u_{uncond} + \gamma(u_{cond} - u_{uncond})$$

for some guidance scale γ. [8]

3.2. Flow Matching

However, initial experiments using DDPM proved to be unsuccessful, as training dynamics prevented fast convergence. Moreover, DPPMs often require a large number of timesteps during sample-time, which is a pitfall of applying them to world modeling; ideally, we are able to train a model that is able to sample efficiently, allowing it to be used in real-time simulations.

To combat this, we instead propose to use flow matching. We employ flow matching [7] to iteratively denoise images by learning a vector field that transitions noisy data toward a clean distribution. Given a noisy sample $x_0 \sim \mathcal{N}(0, 1)$ and a clean sample x_1 from the training data, the flow $u(x_0, t)$, representing the velocity of this transformation, is defined as the derivative of x_t with respect to time:

$$u(x_0, t) = \frac{d}{dt} x_t = x_1 - x_0.$$

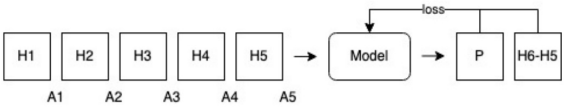

Figure 2. Overview of Single U-Net diffusion model setup for K=5, which conditions on past 5 frames H1 to H5 and actions A1 to A5, and predicts the difference between next and current frame H6-H5.

The UNet model $u_\theta(x_t, t, o, a)$ is trained to approximate this flow, minimizing the objective:

$$\mathcal{L} = \mathbb{E}_{x_t,t,o,a}\left[\|(x_1 - x_0) - u_\theta(x_t, t, o, a)\|^2\right].$$

This approach leads to efficient pathing for the UNet model, leading to better results.

3.3. Alternate Parameterization: Difference Prediction

Further ablations revealed that the above parameterization is sub-optimal—the training dynamics urge the model to blindly reproduce the last ground-truth observation, rather than truly learning the effect of actions in the visual space. We re-parameterize the model, therefore, to predict the *difference* between the last ground-truth frame and the next observation. The idea is that computing a loss between the difference (though in theory, should not make a difference), will help influence training dynamics to emphasize action conditioning. Thus, we update our objective: x_1 and x_0, instead of being clean and noised versions of samples from our dataset, are now clean and noised versions of the difference between adjacent observations from the same episode.

$$\mathcal{L} = \mathbb{E}_{x_t,t,o,a}\left[\|(x_1 - x_0) - u_\theta(x_t, t, o, a)\|^2\right].$$

A diagram of this architecture is shown in Figure 2.

We show later that this parameterization leads to greater inference-time stability, especially when we evaluate the model on auto-regressive sampling over longer horizons.

3.4. Cascaded UNet

As a final experiment, we also try to *cascade* many UNets into a single pipeline (Figure 3). The idea behind this methodology is that the UNet at each subsequent layer will be trained on outputs of the UNet that came before it. In this setup, the final UNet in the cascade is trained mostly on model outputs, and does not see much ground-truth information during training. As a result, we found sample-time rollouts to be far more stable, since the model learns to .

3.5. Transfer Learning

We also experiment with using a pretrained backbone UNet for the cascaded UNets setting, with a ResNet 34 encoder

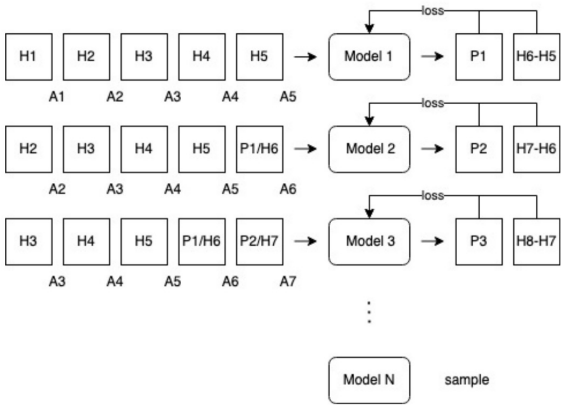

Figure 3. Overview of Cascaded U-Net diffusion model setup for K=5, where later models conditions on predicted frames from earlier models. The last model is used for sampling.

and pretrained on ImageNet. We modify the architecture with Feature-wise Linear Modulation (FiLM) layers to incorporate the time and action conditioning right after the bottleneck in the UNet architecture [8]. We also try freezing the pretrained encoder layers for the first 5 epochs out of 50, but do not note any significant changes in performance from doing so.

3.6. Dataset

We use the **DROID** dataset, which contains a large number of manipulation trajectories collected from the Franka 7DoF robot arm. Each trajectory contains high-resolution frames captured from three different camera angles, and is annotated with joint-level DoF positions, velocities, and actions. Two samples from the dataset are shown in 4

When training on observations from the two exterior angles, it was hard to accurately evaluate the ability of our model to learn actions; the difference between adjacent observations is so small that even models outputting the last ground-truth frame results in trajectories close to the ground truth during inference. In this project, therefore, we focus some of our attention on learning actions from the perspective of the wrist (*wrist_image_left*), as this contains the most background optical flow as well as one of the exterior perspectives of the robot, (*exterior_image_1_left*, which offers the most scope for evaluating different training setups.

Lastly, we focus on a small subset of the data named **Droid 100**, which contains a relatively small number of trajectories compared to the original, full dataset. We do this due to computational constraints; training on the full dataset simply took too long to train for the scope of this project (though it is definitely an avenue for future exploration).

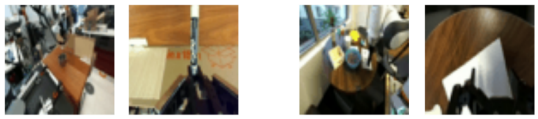

Figure 4. Two samples from Droid 100. Displaying two viewpoints for each sample.

4. Experiments

4.1. Traditional Flow Matching Setup

To address initial convergence-related issues with a DDPM setup, we first attempted to use traditional flow matching. As shown in Figure 6, flow matching converges significantly faster and exhibits less overfitting compared to DDPM. The loss curves and qualitative comparisons between ground-truth and sampled observations (Figure 5) initially indicated strong performance. However, further evaluation revealed that the model failed to effectively capture action-conditioned dynamics, highlighting a key limitation in this setup.

When visualizing auto-regressive rollouts, we observed that the model repeatedly output its most recent frame—effectively copying its own previous prediction during inference. This led to static observations that degraded into noise as the rollout horizon increased (Figure 7).

To further evaluate the model's reliance on actions, we zeroed out the action vectors during inference. Surprisingly, the model still reconstructed frames with similar fidelity, confirming that it had largely ignored the action inputs during training.

4.2. Flow Matching With Difference Prediction

To address this limitation, we re-parameterized the learning objective as detailed in Section 3.3, and observed a marked improvement in the model's reliance on action conditioning. Specifically, we computed the loss between the model's output and a noised version of the difference between consecutive frames, rather than the full observation itself. This encouraged the model to focus on frame-to-frame changes driven by actions, rather than reconstructing static content. Figure 9 shows the results during inference-time, as well as inference results from using a pretrained backbone and cascaded flow matching. Although the model is still quite unstable, it is now able to output non-static trajectories, and learn a limited form of action dynamics in the real world.

4.3. Cascaded UNet

Finally, we perform a variety of ablations by cascading UNets. Preliminary results suggest that increasing the number of models tends to lead to increase in stability, up to the length of the horizon window (increasing the number of models beyond this would not have any added benefit).

We show some of our inference-time visualizations in Figure 9. As displayed in the image, the model is clearly able to maintain image consistency for a far higher number of frames, suggesting that exposure to model output during training is vital. We also visualize some predictions of

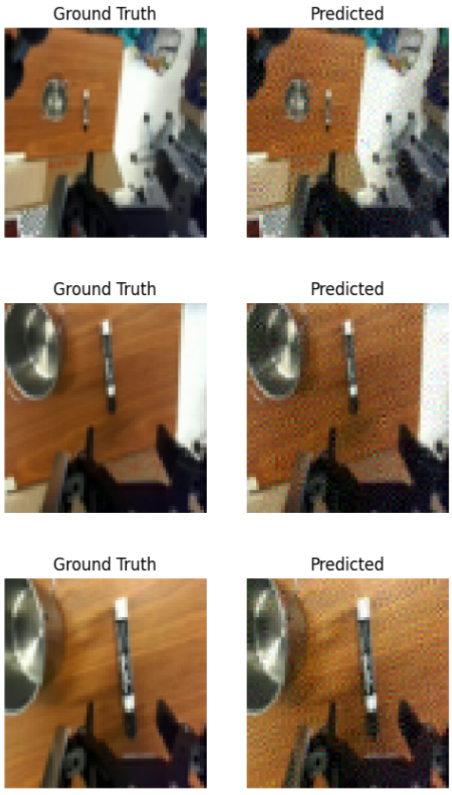

Figure 5. Traditional FM: Ground-Truth vs Predicted

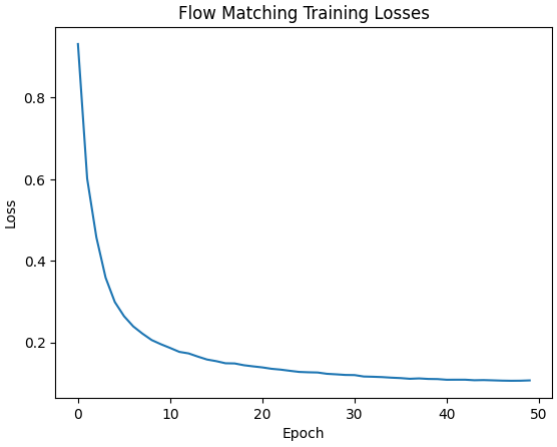

Figure 6. Traditional FM Training Losses

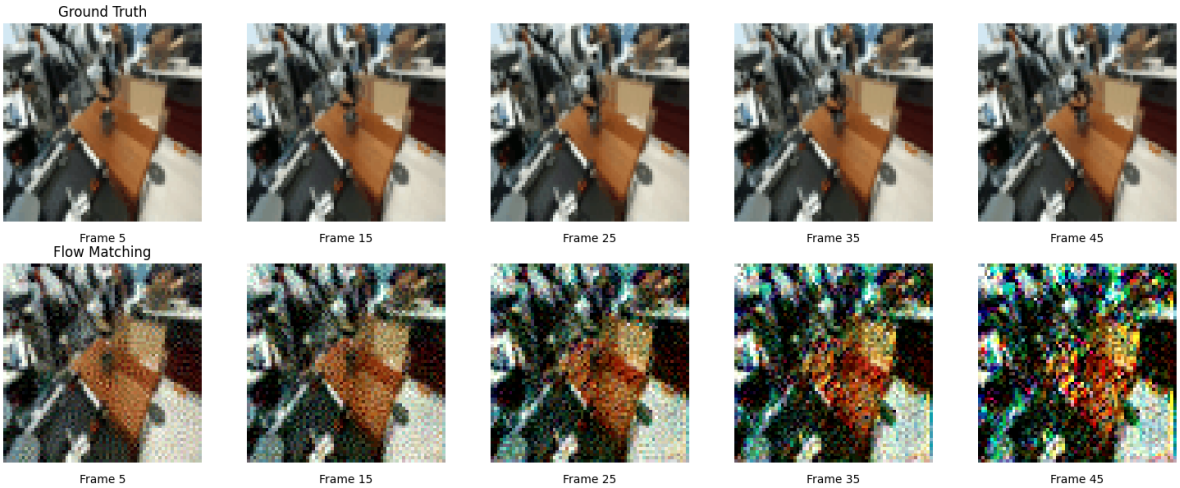

Figure 7. Traditional FM Rollouts. Note the gripper in the Ground Truth is moving downwards, but is stationary in the Flow Matching rollout.

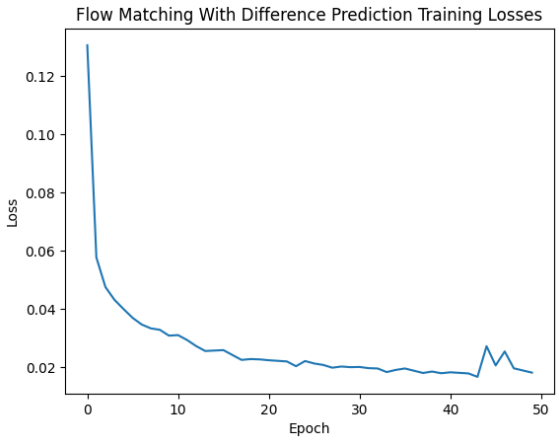

Figure 8. FM Difference Prediction Training Losses

camera frames on the gripper in figure 11. We can see that the model does seem to predict some of the motion of the gripper as it moves away from the table.

The training curve for our best cascaded UNet model is plotted in Figure 10. The UNet has a hidden dimension of 256 and is trained on a horizon of 10 frames and 5 cascaded models. Interestingly, the training losses for the cascaded UNets (and the Flow Matching Difference Prediction) are generally higher than that of single UNets, but the rollouts are far more stable, presumably because the model is more generalized and robust to slightly out-of-distribution images.

4.4. Pre-Trained Backbone

Finally, we try our cascaded UNet implementation with a pre-trained backbone (specifically, a ResNet 34 encoder) as well as the single Flow Matching Difference Predictor. We can see from Figure 9 that using a pretrained UNet (a ResNet 34 encoder trained on ImageNet) does make a small difference in the robustness of the model to veering out of distribution in the cascaded flow matching model but not in the single UNet difference predictor. This is probably because the greater sample efficiency was more beneficial to training the multiple models in the cascaded UNets by making sure each already had some representation of the. natural image space. This extension did not result in any great increase in quality compared to the Cascaded UNet of the previous section, however. Training losses and an autoregressive frame-by-frame comparison are shown in Figures 14 and 9. We believe training for more epochs would result in better results using the pre-trained backbone, and would like to try this in a future iteration of this project.

5. Conclusion

In this project, we explored action-conditioned visual prediction for robotic manipulation using generative flow matching. By conditioning on past visual observations and low-dimensional action data (joint-level), we were able to architect a world model that is able to stay stable during sample-time for several frames at a time. Our experiments on the DROID dataset showed that navie flow matching often leads to the model ignoring the action conditioning. Reparameterizing our UNet to predicting the difference between adjacent observations mitigated this effect, improving the training dynamics to urge more apparent action-conditioning. Finally, we introduce a cascaded UNet architecture to improve inference-time stability during autoregressive rollout. By optimizing a model that sees mostly model outputs during train-time, we show that we can in-

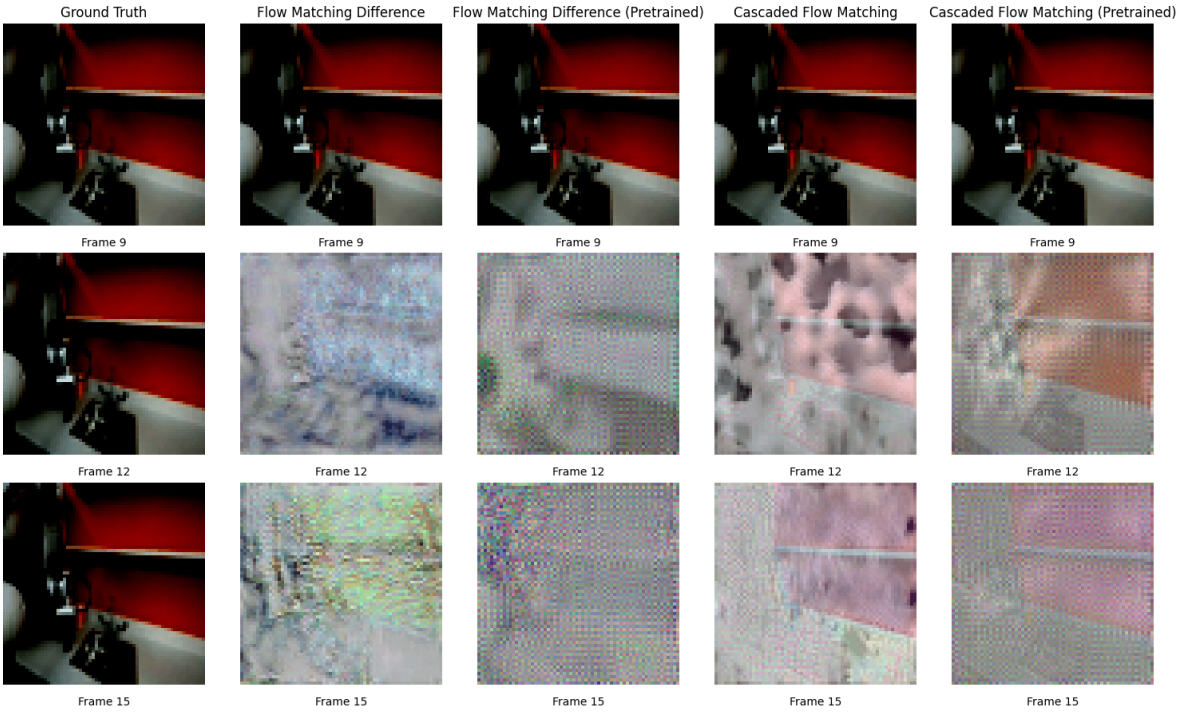

Figure 9. Comparison Rollouts of Flow Matching Difference and Cascaded Flow Matching with and without a pretrained backbone. From this figure it's hard to see (see the website for GIFs), but each of these models is able to predict the motion of the gripper.

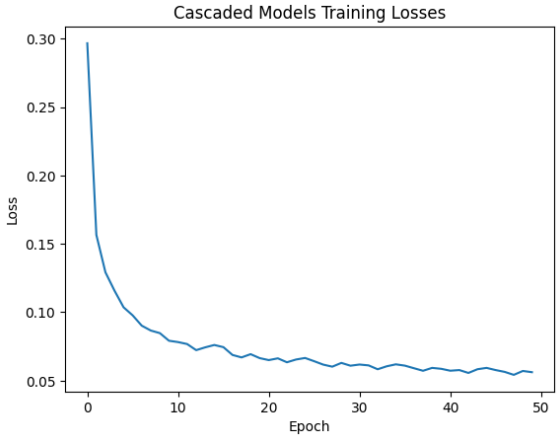

Figure 10. Cascaded FM Training Losses

crease robustness to auto-regressive noise. While our approach remains limited in rollout horizon and data scale, we believe that it is a promising foundation for integrating visual prediction models into model-based planning and sim-to-real pipelines.

6. Supplemental Material

GitHub: Link
Website: Link

The website contains supplemental visuals that cannot be displayed directly on PDF (i.e, GIFs and other animated visuals).

References

[1] Eloi Alonso, Adam Jelley, Vincent Micheli, Anssi Kanervisto, Amos Storkey, Tim Pearce, and François Fleuret. Diffusion for world modeling: Visual details matter in atari. In *NeurIPS*, 2024. Spotlight. 2

[2] Zhangyang Gao, Cheng Tan, Lirong Wu, and Stan Z. Li. Simvp: Simpler yet better video prediction. In *CVPR*, 2022. 2

[3] Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. *arXiv preprint arXiv:1811.04551*, 2019. 1

[4] Danijar Hafner, Timothy Lillicrap, Mohammad Norouzi, and Jimmy Ba. Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*, 2020. 1

[5] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, pages 6840–6851. Curran Associates, Inc., 2020. 2

[6] Tero Karras, Miika Aittala, and Timo Aila. Elucidating the design space of diffusion-based generative models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. 2

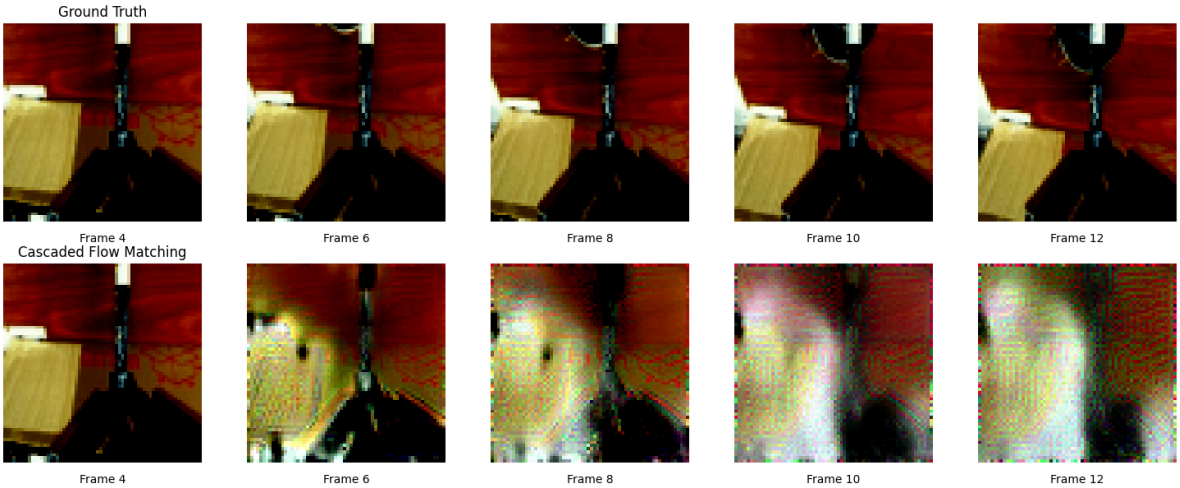

Figure 11. Cascaded UNet Rollout with Gripper Camera. The gripper is moving away from the table after picking up the pen.

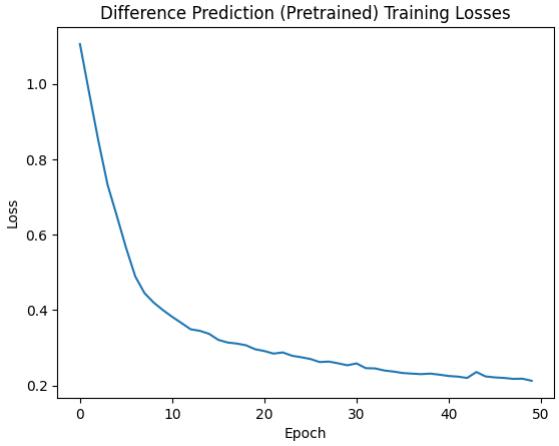

Figure 12. Pretrained FM Difference Prediction Training Losses

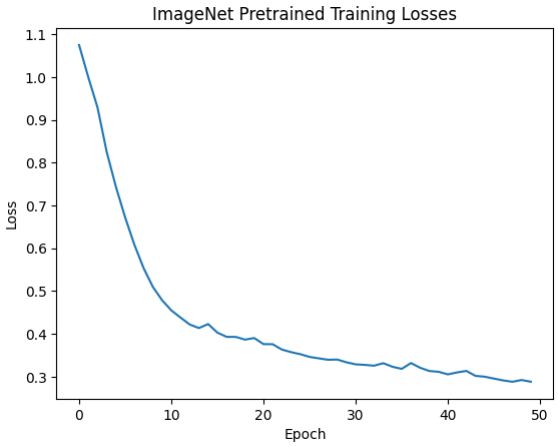

Figure 13. Pretrained Cascaded UNet Prediction Training Losses

[7] Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations*, 2023. 2

[8] Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI conference on artificial intelligence*, 2018. 2, 3

[9] Scott Reed, Konrad Zolna, Emilio Parisotto, Alexander Novikov, et al. A generalist agent. *TMLR*, 2022. 2

[10] Younggyo Seo, Danijar Hafner, Hao Liu, Fangchen Liu, Stephen James, Kimin Lee, and Pieter Abbeel. Masked world models for visual control. In *CoRL*, 2022. 2

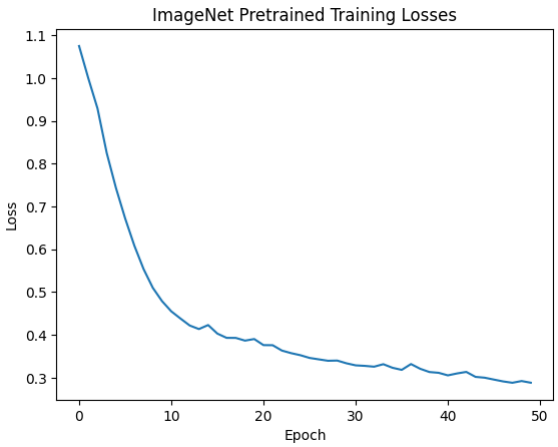

Figure 14. Pretrained Backbone w/ Cascaded UNet Training Losses