Goal-Aware Prediction: Learning to Model What Matters Supplementary Material

1. Method Implementation Details

In this section we go over implementation details for our method as well as our comparisons.

1.1. Architecture Details

Block/Door Domain: All comparisons leverage a nearly identical architecture, and are trained on an Nvidia 2080 RTX. In the block pushing domain input observations are [64,64, 6] in the case of our model (GAP), as well as the ablations, and [64,64, 3] in the case of Standard.

All use an encoder f_{enc} with convolutional layers (channels, kernel size, stride): [(32, 4, 2), (32, 3,1), (64, 4, 2), (64, 3,1), (128, 4, 2), (128, 3,1), (256, 4, 2), (256, 3,1)] followed by fully connected layers of size [512, 2 $\times L$] where L is the size of the latent space (mean and variance). All layers except the final are followed by ReLU activation.

The decoder f_{dec} takes a sample from the latent space of size L, then is fed through fully connected layers [128, 128, 128], followed by de-convolutional layers (channels, kernel size, stride): [(128, 5, 2), (64, 5, 2), (32, 6, 2), (3, 6,2)]. All layers except the final are followed bu ReLu activation, except the last layer which is a Sigmoid in the case of Standard, and GAP (-Residual).

For all models the dynamics model f_{dyn} are a fully connected network with layers [128, 128, 128, L], followed by ReLU activation except the final layer.

The inverse model baseline utilizes the same f_{enc} and f_{dyn} as above, but f_{dec} is instead a fully connected network of size [128, 128, action size] where action size is 4 (corresponding to delta x,y, z motion and gripper control). All layers except the final are followed by ReLU activation.

Lastly, the RIG (Nair et al., 2018) baseline uses a VAE with identical f_{enc} and f_{dec} to the standard approach above, except learns a policy in the latent space. The policy architecture used is the default SAC (Haarnoja et al., 2018) from RLkit, namely 2 layer MLPs of size 256.

. Correspondence to: Suraj Nair <surajn@stanford.edu>.

Proceedings of the 37^{th} International Conference on Machine Learning, Online, PMLR 119, 2020.

SVG+GAP: In all SVG (Denton and Fergus, 2018) based experiments on real robot data, the architecture used is identical to the SVG architecture as described in official repo[1] with the VGG encoder/decoder. All BAIR dataset experiments take as input sequences of 2 frames and predict 10 frames, while all RoboNet experiments take as input 2 frames and predict 20 frames. The latent dimension is 64, and the encoder output dimension is 128. All models are trained with batch size 32.

1.2. Training Details

Block/Door Domain: We collect a dataset of 2,000 trajectories, each 50 timesteps with a random policy. All models are trained on this dataset to convergence for roughly 300,000 iterations. All models are trained with learning rate of 1e-4, and batch size 32.

The RIG baseline is trained using the default SAC example parameters in RLkit, for an additional 3 million steps.

BAIR Robot Dataset: We train on the BAIR Robot Dataset (Ebert et al., 2017) as done in the original SVG paper, except with action conditioning.

RoboNet: We train on the subset of the RoboNet dataset which considers only the sawyer arm and the front facing camera view, and use a random 80/20 train test split.

1.3. Task/Evaluation Details

Tasks. All tasks are defined in a Mujoco simulation built off the Meta-World environments (Yu et al., 2019). In Task 1, the agent must push a single block to a target position, as specified by a goal image. The task involves either pushing the pink, green, or blue block. Success if defined as being within 0.08 of the goal position. In Task 2 the agent must push 2 blocks, specifically the green and blue block to their respective goal positions, again indicated by a goal image. Success is determined as both blocks being within 0.1 of their respective goal positions. In Tasks 3 and 4 the agent must close or open a door, as specified by a goal image, where success is definged as being within $\pi/6$ radians of the goal angle.

Evaluation. During all control experiments, evaluation is done using model predictive control with the latent space

[1] https://github.com/edenton/svg

dynamics model. Specifically, we do latent MPC as described in Algorithm 1, specifically by planning 15 actions, executing them in the environment, then planning 15 more actions and executing them. Each stage of planning uses the cross entropy method, specifically sampling 1000 action sequences, sorting them by the mean latent distance cost to the goal, refitting to the top 10, and repeating 3 times, before selecting the total lowest cost action.

2. Additional Results

In Figure 1 we present additional examples of combining SVG with GAP on the BAIR dataset. Again we consider the goal image to be the last state in the trajectory. We see that using GAP leads to more effectively capturing task relevant objects, highlighted in red.

References

E. L. Denton and R. Fergus. Stochastic video generation with a learned prior. In *International Conference on Machine Learning*, 2018.

F. Ebert, C. Finn, A. X. Lee, and S. Levine. Self-supervised visual planning with temporal skip connections. *CoRR*, abs/1710.05268, 2017.

T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *ArXiv*, abs/1801.01290, 2018.

A. V. Nair, V. Pong, M. Dalal, S. Bahl, S. Lin, and S. Levine. Visual reinforcement learning with imagined goals. In *Advances in Neural Information Processing Systems*, pages 9191–9200, 2018.

T. Yu, D. Quillen, Z. He, R. R. Julian, K. Hausman, C. Finn, and S. Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. *ArXiv*, abs/1910.10897, 2019.

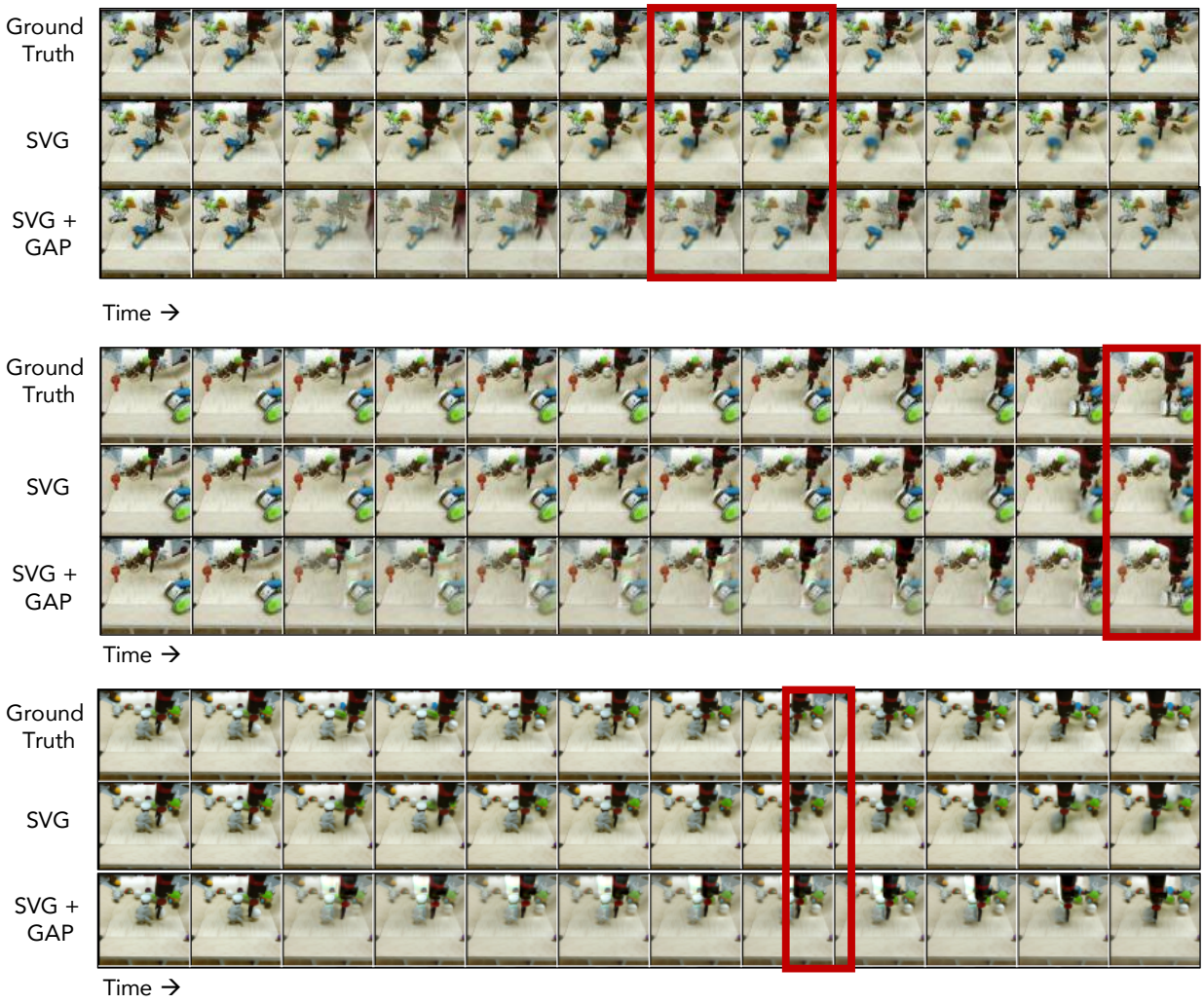

Figure 1: **GAP+SVG Video Prediction (BAIR Robot Dataset):** We present additional qualitative examples of action-conditioned SVG with and without GAP on the BAIR robot dataset, predicting on goal-reaching trajectories. Note, in the GAP predictions the goal is added back to the predicted goal-state residual. In this case the goal is the rightmost frame. In the top example, we see that GAP more effectively models the water gun, while normal SVG blurs it out. In the middle example, we see that by using the goal information, GAP is able to more effectively model the jar. In the bottom example we see that SVG blurs out a background object, while GAP does not.