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

Suraj Nair†, Silvio Savarese†, Chelsea Finn†
†Stanford University
Email: surajn@stanford.edu
sites.google.com/stanford.edu/gap

A. Related Work

Recent years have seen impressive results from reinforcement learning [58] applied to challenging problems such as video games [41, 46], Go [56], and robotics [38, 47, 34]. However, the dependence on large quantities of labeled data can limit the applicability of these methods in the real world. One approach is to leverage self-supervision, where an agent only uses data that it can collect autonomously.

Self-Supervised Reinforcement Learning: Self-supervised reinforcement learning explores how RL can leverage data which the agent can collect autonomously to learn meaningful behaviors, without dependence on task specific reward labels, with promising results on tasks such as robotic grasping and object re-positioning [50, 16, 68]. One approach to self-supervised RL has been combining goal-conditioned policy learning [33, 54, 9] with goal re-labeling [4] or sampling goals [44, 43]. While there are numerous ways to leverage self-supervised data, ranging from learning distance metrics [67, 29], generative models over the state space [35, 18, 17, 39, 45], and representations [60], one of the most heavily utilized techniques is learning the dynamics of the environment [64, 20, 1].

Model-Based Reinforcement Learning: Learning a model of the dynamics of the environment and using it to complete tasks has been a well studied approach to solving reinforcement learning problems, either through planning with the model [11, 64, 40, 6, 8, 2, 28, 42] or optimizing a policy in the model [51, 25, 70, 37, 32, 63, 27, 24, 7]. Numerous works have explored how these methods might leverage deep neural networks to extend to high dimensional problem settings, such as images. One technique has been to learn large video prediction models [20, 5, 15, 16, 49, 36, 61, 65], however model under-fitting remains an issue for these approaches [10]. Similarly, many works have explored learning low dimensional latent representations of high dimensional states [64, 14, 69, 28, 35, 31, 62, 37, 23] and learning the dynamics in the latent space. Unlike these works, we aim to make the problem easier by encouraging the network to predict only task-relevant quantities, while also changing the objective, and hence the distribution of prediction errors, in a task-driven way. This allows the prediction problem to be more directly connected to the downstream use-case of task-driven planning.

Addressing Model Errors: Other works have also studied the problem of model error and exploitation. Approaches such as ensembles [8, 59] have been leveraged to measure uncertainty in model predictions. Similarly, Janner et al. [32] explore only leveraging the learned model over finite horizons where it has accurate predictions and Levine et al. [38] use local models. Exploration techniques can also be used to collect more data where the model is uncertain [48].

Most similar to our proposed approach are techniques which explicitly change the models objective to optimize for performance on downstream tasks. [55, 30] explore only predicting future reward to learn a latent space in which they learn dynamics, Freeman et al. [22] learn a model with the objective of having a policy achieve high reward from training in it, and Amos et al. [3], Srinivas et al. [57] embed a model/planner inside a neural network. Similarly, Farahmand et al. [19], D'Oro et al. [13] explore how model training can be re-weighted using value functions or policy gradients to emphasize task specific performance. Unlike these works, which depend heavily on task-specific supervision, our proposed approach can be learned on purely self-supervised data, and generalize to unseen tasks.

B. Method Implementation Details

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

1) Implementing Goal-Aware Prediction

We implement GAP with a latent dynamics model. Given a dataset of trajectories $[\tau_1, ..., \tau_N]$, we sample sequences of states $[(s_1, a_1), ..., (s_T)]$ where we re-label goal for the trajectory as $s_g = s_T$.

The GAP model consists of three components, (1) an encoder $f_{enc}(z_t|s_t, s_g; \theta_{enc})$ that encodes the state s_t and goal s_g into a latent space z_t, (2) a decoder $f_{dec}(s_g - s_t|z_t; \theta_{dec})$ that decodes samples from the latent distribution into $s_g - s_t$, and (3) a forward dynamics model in the latent space $f_{dyn}(z_{t+1}|z_t, a_t; \theta_{dyn})$ which learns to predict the future latent distribution over z_{t+1} from z_t and action a_t. In our experiments we work in the setting where states are images, so $f_{enc}(z_t|s_t, s_g)$ and $f_{dec}(s_g - s_t|z_t)$ are convolutional neural networks, and $f_{dyn}(z_{t+1}|z_t, a_t)$ is a fully-connected network. The full set of parameters $\theta = \{\theta_{enc}, \theta_{dec}, \theta_{dyn}\}$

are jointly optimized. Exact architecture and training details for all modules can be found in the supplement. Following prior works [21, 20, 2], we train for multi-step prediction. More specifically, given $s_t, a_{t:t+H}, s_g$, the model is trained to reconstruct $(s_g - s_t), ..., (s_g - s_{t+H})$.

Data Collection and Model Training: In our *self-supervised* setting, data collection simply corresponds to rolling out a random exploration policy in the environment. Specifically, we sample uniformly from the agent's action space, and collect 2000 episodes, each of length 50, for a total of 100,000 frames of data.

During training, sub-trajectories of length 30 time steps are sampled from the data set, with the last timestep labeled as the goal $s_g = s_{30}$. Depending on the current value of H, loss is computed over H step predictions starting from states $s_{t:(t+H)}$. We use a curriculum when training all models, where H starts at 0, and is incremented by 1 every 50,000 training iterations. All models are trained to convergence, for about $300,000$ iterations on the same dataset.

Planning with GAP: For all trained models, when given a new goal at test time s_g, we plan using model predictive control (MPC) in the latent space of the model. Specifically, both the current state s_t and s_g are encoded into their respective latent spaces z_t and z_g (Algorithm 1, Line 3).

Algorithm 1 Latent MPC$(f_{enc}, f_{dyn}, s_t, s_g)$

1: Let $D = 1000, D^* = 10, H = 15$
2: Receive current state s_t and goal state s_g
3: Encode $z_t \sim f_{enc}(s_t, s_g), z_g \sim f_{enc}(s_g, s_g)$
4: Initialize $N(\mu, \sigma^2) = \mathcal{N}(0, 1)$
5: Let cost function $C(z_i, z_j) = ||z_i - z_j||_2^2$
6: **while** iterations ≤ 3 **do**
7: $\quad a^1_{t:H}, ..., a^D_{t:H} \sim N(\mu, \sigma^2)$
8: $\quad z^1_{t+1:t+H}, ..., z^D_{t+1:t+H} \sim f_{dyn}(z_t, a^1_{t:H}, ..., a^D_{t:H})$
9: $\quad \hat{c}^1, ..., \hat{c}^D = [\sum_{h=1}^{H} \mathcal{C}(z^1_{t+h}, z_g), ..., \sum_{h=1}^{H} \mathcal{C}(z^D_{t+h}, z_g)]$
10: $\quad a_{sorted} = Sort([a^1_{t:H}, ..., a^D_{t:H}])$ by $\hat{c}$
11: $\quad$ Refit μ, σ^2 to $a_{sorted}[1 : D^*]$
12: **end while**
13: Return $\hat{c}_{sorted}[1]$, $a_{sorted}[1]$

Then using the model $f_{dyn}(z_{t+1}|z_t, a_t)$, the agent plans a sequence of H actions to minimize cost $\sum_{h=0}^{H} ||z_g - z_{t+h}||_2^2$ (Algorithm 1, Lines 4-11). Following prior works [20, 28], we use the cross-entropy method [52] as the planning optimizer. Finally, the best sequence of actions is returned and executed in the environment (Algorithm 1, Line 13).

While executing the plan, our model re-plans every H timesteps. That is, it starts at state s_t, uses Latent MPC (Algorithm 1) to first plan a sequence of H actions, executes them in the environment resulting in a state s_{t+H}, then re-plans an additional H actions, and executes them resulting in a final state s_T. Success is computed based the difference between s_T and s_g.

2) 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 [44] 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 [26] from RLkit, namely 2 layer MLPs of size 256.

SVG+GAP: In all SVG [12] 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.

3) 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 [15] 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.

4) Task/Evaluation Details

Tasks. All tasks are defined in a Mujoco simulation built off the Meta-World environments [66]. In Task 1, the agent must push a single block to a target position, as specified by a

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

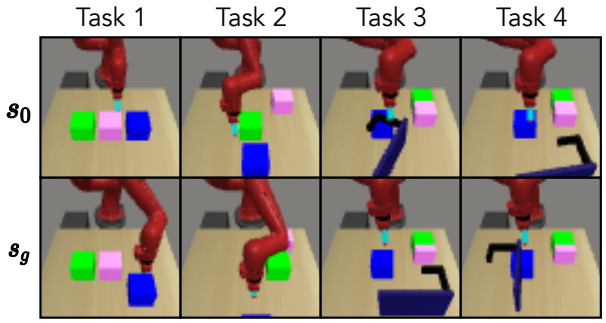

Fig. 1. **Evaluation Tasks:** Sample initial & goal states for each of the simulated manipulation tasks. Tasks involve manipulating blocks or a door, with the task specified by a goal image.

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 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.

5) Experimental Comparisons

Comparisons: We compare to several model variants in our experiments. **GAP** is our approach of learning dynamics in a latent space conditioned on the current state and goal, and reconstructing the residual between the current state and goal state. **GAP (-Goal Cond)** is an ablation of GAP that does not use goal conditioning. Instead of conditioning on the goal and predicting the residual to the goal, it is conditioned on the initial state, and predicts the residual to the initial state. This is representative of prior works (e.g. Nagabandi et al. [42]) that predict residuals for model-based RL. **GAP (-Residual)** is another ablation of GAP that is conditioned on the goal but maintains the standard reconstruction objective instead of the residual. This is similar to prior work on goal conditioned video prediction [53]. **Standard** refers to a standard latent dynamics model, representative of approaches such as PlaNet [28], but without reward prediction since we are in the self-supervised setting.

When studying task performance, we also compare to two alternative self-supervised reinforcement learning approaches. First, we compare to an **Inverse Model**, which is a latent dynamics model where the latent space is learned via an action prediction loss (instead of image reconstruction), as done in Pathak et al. [48]. Second, we compare to a model-free approach: reinforcement learning with imagined goals (**RIG**) [44], where we train a VAE on the same pre-collected dataset as the other models, then train a policy in the latent space of the VAE to reach goals sampled from the VAE. Further implementation details can be found in the supplement.

C. Additional Results

1) Theorem 3.1 Proof

Proof: For the specified policy, violating ϵ-optimality will only occur if the cost of the best action sequence $a^1_{1:T}$ is overestimated or if the cost of a sub-optimal action sequence $(i \mid c^*_i > c^*_1 + \epsilon)$ is underestimated. Thus, let us define the "worst case" cost predictions as the ones for which c^*_1 is most overestimated and $c^*_i \quad \forall i \mid c^*_i > c^*_1 + \epsilon$ are most underestimated (while still satisfying Equations 2 and 3). Concretely we write the worst case cost estimates as

$$\tilde{c}_i := \min \hat{c}_i \quad \forall i \mid c^*_i > c^*_1 + \epsilon$$

$$\tilde{c}_1 := \max \hat{c}_1$$

s.t. Eq. 2 and 3 hold. We will now show that $\tilde{c_1} < \tilde{c_i} \quad \forall i \mid c^*_i > c^*_1 + \epsilon$. First, since $\tilde{c_i}$ satisfies Eq. 3, we have that

$$\tilde{c_i} > c^*_i - (c^*_i - c^*_1) + \epsilon$$

Similarly, since $\tilde{c_1}$ satisfies Eq. 2, we have that

$$\tilde{c_1} < c^*_1 + \epsilon$$

Substituting, we see that

$$\tilde{c_i} > c^*_i - (c^*_i - c^*_1) + \epsilon = c^*_1 + \epsilon > \tilde{c_1} \quad \forall i \mid c^*_i > c^*_1 + \epsilon \quad (1)$$

Hence even in the worst case, Equations 2 and 3 ensure that $\hat{c_i} > \hat{c_1} \quad \forall i \mid c^*_i > c^*_1 + \epsilon$, and thus no action sequence i for which $c^*_i > c^*_1 + \epsilon$ will be selected and the policy will remain ϵ-optimal. Note that action sequences besides $i = 1$ for which $c^*_i \leq c^*_1 + \epsilon$ costs are unbounded, as it is ok for them to be significantly underestimated since selecting them still allows the policy to be ϵ-optimal. ■

2) Verifying Theorem 3.1 Experimentally

We now verify the above analysis through a controlled study of how prediction error affects task performance. To do so, we will use the true model of an environment and true cost of an action sequence for planning, but will artificially add noise to the cost/model predictions to generate model error.

Consider a 2 dimensional navigation task, where the agent is initialized at $s_0 = [0.5, 0.5]$ and is randomly assigned a goal $s_g \in [0, 1]^2$. Assume we have access to the underlying model of the environment, and cost defined as $\mathcal{C}(s_t, s_g) = ||s_t - s_g||_2$. We can run the policy described in Theorem 3.1, specifically sampling $N = 100$ action sequences, and selecting the one with lowest predicted cost, where we consider 2 cases: (1) predicted cost is using the true model, but with noise α added to the true cost $\hat{c_i} = c^*_i + \alpha$ of some subset of action sequences, and (2) predicted cost is true cost, but with noise α added to the model predictions $s_{t+1} = \bar{s}_{t+1} + \alpha$ where $\bar{s}_{t+1} \sim p(s_{t+1}|s_t, a_t)$ of some subset of action sequences. The

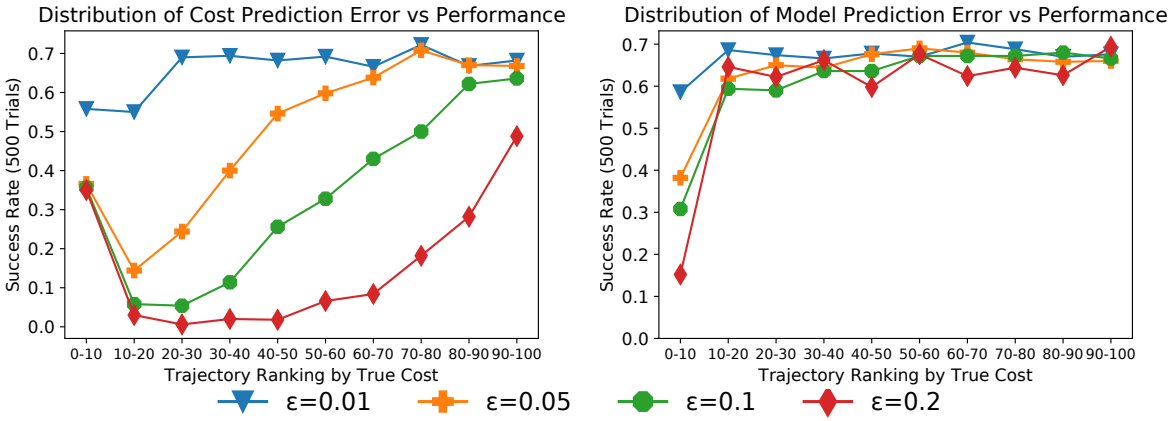

Fig. 2. **Distribution of model errors vs. performance**: We validate how the distribution of model errors affects performance on a simple 2D navigation domain, by adding noise to cost predictions **(left)** or model predictions **(right)**. We add varying amounts of noise with magnitude up to ε to the predictions of the 10 lowest true cost trajectories **(0-10)** to the 10 highest true cost trajectories **(90-100)**. We observe that adding noise to low true cost trajectories dramatically reduces performance, while adding noise to the high true cost trajectories has no nearly no impact on performance.

first case relates directly to Theorem 3.1, while the second case relates to what we can control when training a self-supervised dynamics model. When selectively adding noise, we will use uniform noise $\alpha \sim \mathcal{U}(-\varepsilon, \varepsilon)$. We specifically study the difference in task performance when adding noise α to model predictions for the first 10% of trajectories with lowest true cost, the second 10% lowest true cost trajectories, etc., up to the 10% of trajectories with highest true cost. Here "true cost" refers to the cost of the action sequence under the true model and cost function without noise. For each noise augmented model we measure the task performance, specifically the success rate (reaching within 0.1 of the goal), over 500 random trials.

We see in Figure 2 that for multiple values of noise ε, when adding noise to the better (lower true cost) trajectories we see a significant drop in task performance, while when adding noise to the worse (higher true cost) trajectories task performance remains relatively unchanged (except for the case with very large ε). In particular, we notice that when adding noise to cost predictions, performance scales almost linearly as we add noise to worse trajectories. Note there is one exception to this trend: if we add noise only to the top 10% of trajectories, performance is not optimal, but reasonable because the best few trajectories will occasionally be assigned a lower cost under the noise model.

In the case of model error, we see a much steeper increase in performance, where adding model error to the best 10 trajectories significantly hurts performance, while adding to the others does not. This is because, in this environment, noise added to model predictions generally makes the cost of those predictions worse; so if no noise is added to the best trajectories, the best action sequence is still likely to be selected. The exact relationship between model prediction error and cost prediction error depends on the domain and task. But, we can see that in both cases in Figure 2, the conclusion from Theorem 3.1 holds true: accuracy on good action sequences matters much more than accuracy on bad action sequences.

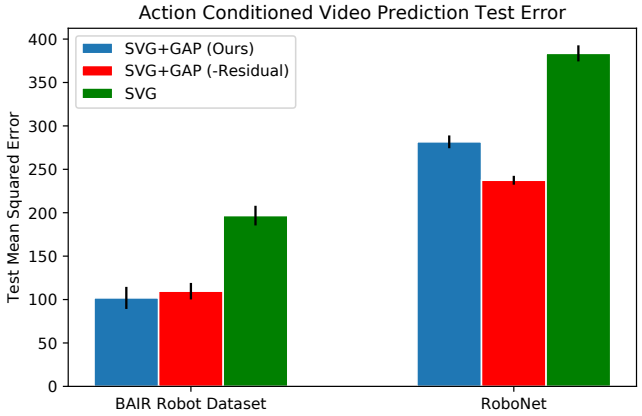

Fig. 3. **Distribution of Model Errors (Real Robot Data):** We examine the model error of SVG combined with GAP on **unseen, goal-reaching trajectories** from two real robot datasets (the BAIR Dataset [15] and the RoboNet Dataset [10]). We see that action-conditioned SVG combined with GAP has lower prediction error on the goal reaching trajectories than standard action-conditioned SVG. We observe that the GAP ablation which also conditions on the goals, but predicts residuals is equally effective in this setting.

3) Additional Real Robot Results

We compare the prediction error of SVG to SVG+GAP on goal reaching trajectories (Figure 3) from real robot datasets, namely the BAIR Robot Dataset [15] and the RoboNet Dataset [10]. We see that action-conditioned SVG combined with GAP as well as the ablation without residual prediction both have lower prediction error on goal reaching trajectories than standard action-conditioned SVG.

REFERENCES

[1] Pulkit Agrawal, Ashvin V Nair, Pieter Abbeel, Jitendra Malik, and Sergey Levine. Learning to poke by poking: Experiential learning of intuitive physics. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing*

Systems 29, pages 5074–5082. Curran Associates, Inc., 2016.

[2] Brandon Amos, Laurent Dinh, Serkan Cabi, Thomas Rothörl, Alistair Muldal, Tom Erez, Yuval Tassa, Nando de Freitas, and Misha Denil. Learning awareness models. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=r1HhRfWRZ.

[3] Brandon Amos, Ivan Jimenez, Jacob Sacks, Byron Boots, and J Zico Kolter. Differentiable mpc for end-to-end planning and control. In *Advances in Neural Information Processing Systems*, pages 8289–8300, 2018.

[4] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. *CoRR*, abs/1707.01495, 2017.

[5] Mohammad Babaeizadeh, Chelsea Finn, Dumitru Erhan, Roy H. Campbell, and Sergey Levine. Stochastic variational video prediction. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=rk49Mg-CW.

[6] Ershad Banijamali, Rui Shu, Mohammad Ghavamzadeh, Hung Hai Bui, and Ali Ghodsi. Robust locally-linear controllable embedding. *ArXiv*, abs/1710.05373, 2017.

[7] Arunkumar Byravan, Jost Tobias Springenberg, Abbas Abdolmaleki, Roland Hafner, Michael Neunert, Thomas Lampe, Noah Siegel, Nicolas Manfred Otto Heess, and Martin A. Riedmiller. Imagined value gradients: Model-based policy optimization with transferable latent dynamics models. *ArXiv*, abs/1910.04142, 2019.

[8] Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *Advances in Neural Information Processing Systems*, pages 4754–4765, 2018.

[9] Felipe Codevilla, Matthias Müller, Alexey Dosovitskiy, Antonio López, and Vladlen Koltun. End-to-end driving via conditional imitation learning. *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–9, 2017.

[10] Sudeep Dasari, Frederik Ebert, Stephen Tian, Suraj Nair, Bernadette Bucher, Karl Schmeckpeper, Surender Singh, Sergey Levine, and Chelsea Finn. Robonet: Large-scale multi-robot learning. *ArXiv*, abs/1910.11215, 2019.

[11] Marc Deisenroth and Carl E Rasmussen. Pilco: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on machine learning (ICML-11)*, pages 465–472, 2011.

[12] Emily L. Denton and Rob Fergus. Stochastic video generation with a learned prior. In *International Conference on Machine Learning*, 2018.

[13] Pierluca D'Oro, Alberto Maria Metelli, Andrea Tirinzoni, Matteo Papini, and Marcello Restelli. Gradient-aware model-based policy search. *ArXiv*, abs/1909.04115, 2019.

[14] Alexey Dosovitskiy and Vladlen Koltun. Learning to act by predicting the future. *ArXiv*, abs/1611.01779, 2016.

[15] Frederik Ebert, Chelsea Finn, Alex X. Lee, and Sergey Levine. Self-supervised visual planning with temporal skip connections. *CoRR*, abs/1710.05268, 2017.

[16] Frederik Ebert, Chelsea Finn, Sudeep Dasari, Annie Xie, Alex X. Lee, and Sergey Levine. Visual foresight: Model-based deep reinforcement learning for vision-based robotic control. *CoRR*, abs/1812.00568, 2018.

[17] Ben Eysenbach, Russ R Salakhutdinov, and Sergey Levine. Search on the replay buffer: Bridging planning and reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 15246–15257, 2019.

[18] Kuan Fang, Yuke Zhu, Animesh Garg, Silvio Savarese, and Li Fei-Fei. Dynamics learning with cascaded variational inference for multi-step manipulation. *ArXiv*, abs/1910.13395, 2019.

[19] Amir-Massoud Farahmand, Andre Barreto, and Daniel Nikovski. Value-Aware Loss Function for Model-based Reinforcement Learning. In Aarti Singh and Jerry Zhu, editors, *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pages 1486–1494, Fort Lauderdale, FL, USA, 20–22 Apr 2017. PMLR. URL http://proceedings.mlr.press/v54/farahmand17a.html.

[20] Chelsea Finn and Sergey Levine. Deep visual foresight for planning robot motion. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2786–2793. IEEE, 2017.

[21] Chelsea Finn, Ian J. Goodfellow, and Sergey Levine. Unsupervised learning for physical interaction through video prediction. In *NIPS*, 2016.

[22] Daniel Freeman, David Ha, and Luke Metz. Learning to predict without looking ahead: World models without forward prediction. In *Advances in Neural Information Processing Systems*, pages 5379–5390, 2019.

[23] Carles Gelada, Saurabh Kumar, Jacob Buckman, Ofir Nachum, and Marc G. Bellemare. Deepmdp: Learning continuous latent space models for representation learning. *CoRR*, abs/1906.02736, 2019. URL http://arxiv.org/abs/1906.02736.

[24] Karol Gregor, Danilo Jimenez Rezende, Frédéric Besse, Yan Wu, Hamza Merzic, and Aaron van den Oord. Shaping belief states with generative environment models for rl. In *NeurIPS*, 2019.

[25] David Ha and Jürgen Schmidhuber. World models. *ArXiv*, abs/1803.10122, 2018.

[26] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *ArXiv*, abs/1801.01290, 2018.

[27] Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*, 2019.

[28] Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben

Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *International Conference on Machine Learning*, pages 2555–2565, 2019.

[29] Kristian Hartikainen, Xinyang Geng, Tuomas Haarnoja, and Sergey Levine. Dynamical distance learning for semi-supervised and unsupervised skill discovery. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=H1lmhaVtvr.

[30] Aaron Havens, Yi Ouyang, Prabhat Nagarajan, and Yasuhiro Fujita. Learning latent state spaces for planning through reward prediction, 2020. URL https://openreview.net/forum?id=ByxJjlHKwr.

[31] Brian Ichter and Marco Pavone. Robot motion planning in learned latent spaces. *IEEE Robotics and Automation Letters*, 4(3):2407–2414, 2019.

[32] Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. In *NeurIPS*, 2019.

[33] Leslie Pack Kaelbling. Learning to achieve goals. In *IJCAI*, pages 1094–1098, 1993.

[34] Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, and Sergey Levine. Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation. *arxiv:Preprint*, 2018.

[35] Thanard Kurutach, Aviv Tamar, Ge Yang, Stuart J Russell, and Pieter Abbeel. Learning plannable representations with causal infogan. In *Advances in Neural Information Processing Systems*, pages 8733–8744, 2018.

[36] Alex X. Lee, Richard Zhang, Frederik Ebert, Pieter Abbeel, Chelsea Finn, and Sergey Levine. Stochastic adversarial video prediction. *CoRR*, abs/1804.01523, 2018.

[37] Alex X. Lee, Anusha Nagabandi, Pieter Abbeel, and Sergey Levine. Stochastic latent actor-critic: Deep reinforcement learning with a latent variable model. *ArXiv*, abs/1907.00953, 2019.

[38] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016.

[39] Kara Liu, Thanard Kurutach, Pieter Abbeel, and Aviv Tamar. Hallucinative topological memory for zero-shot visual planning, 2020. URL https://openreview.net/forum?id=BkgF4kSFPB.

[40] Rowan McAllister and Carl Edward Rasmussen. Improving pilco with bayesian neural network dynamics models. 2016.

[41] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin A. Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen. King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518:529–533, 2015.

[42] Anusha Nagabandi, Kurt Konoglie, Sergey Levine, and Vikash Kumar. Deep dynamics models for learning dexterous manipulation. *ArXiv*, abs/1909.11652, 2019.

[43] Ashvin Nair, Shikhar Bahl, Alexander Khazatsky, Vitchyr Pong, Glen Berseth, and Sergey Levine. Contextual imagined goals for self-supervised robotic learning. *ArXiv*, abs/1910.11670, 2019.

[44] Ashvin V Nair, Vitchyr Pong, Murtaza Dalal, Shikhar Bahl, Steven Lin, and Sergey Levine. Visual reinforcement learning with imagined goals. In *Advances in Neural Information Processing Systems*, pages 9191–9200, 2018.

[45] Suraj Nair and Chelsea Finn. Hierarchical foresight: Self-supervised learning of long-horizon tasks via visual subgoal generation. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=H1gzR2VKDH.

[46] OpenAI. Openai five. https://blog.openai.com/openai-five/, 2018.

[47] OpenAI, Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Józefowicz, Bob McGrew, Jakub W. Pachocki, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, Jonas Schneider, Szymon Sidor, Josh Tobin, Peter Welinder, Lilian Weng, and Wojciech Zaremba. Learning dexterous in-hand manipulation. *CoRR*, abs/1808.00177, 2018. URL http://arxiv.org/abs/1808.00177.

[48] Deepak Pathak, Pulkit Agrawal, Alexei A. Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 488–489, 2017.

[49] Chris Paxton, Yotam Barnoy, Kapil D. Katyal, Raman Arora, and Gregory D. Hager. Visual robot task planning. *CoRR*, abs/1804.00062, 2018.

[50] Lerrel Pinto and Abhinav Gupta. Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours. In *2016 IEEE international conference on robotics and automation (ICRA)*, pages 3406–3413. IEEE, 2016.

[51] Sébastien Racanière, Theophane Weber, David P. Reichert, Lars Buesing, Arthur Guez, Danilo Jimenez Rezende, Adrià Puigdomènech Badia, Oriol Vinyals, Nicolas Manfred Otto Heess, Yujia Li, Razvan Pascanu, Peter W. Battaglia, Demis Hassabis, David Silver, and Daan Wierstra. Imagination-augmented agents for deep reinforcement learning. *ArXiv*, abs/1707.06203, 2017.

[52] R Rubinstein and D Kroese. *The Cross-Entropy Method: A Unified Approach to Combinatorial Optimization, Monte-Carlo Simulation and Machine Learning*. 01 2004.

[53] Oleh Rybkin, Karl Pertsch, Frederik Ebert, Dinesh Jayaraman, Chelsea Finn, and Sergey Levine. Goal-

conditioned video prediction, 2020. URL https://openreview.net/forum?id=B1g79grKPr.

[54] Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. Universal value function approximators. In *International Conference on Machine Learning*, 2015.

[55] Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, Timothy P. Lillicrap, and David Silver. Mastering atari, go, chess and shogi by planning with a learned model. *ArXiv*, abs/1911.08265, 2019.

[56] David Silver, Aja Huang, Christopher J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529:484–503, 2016. URL http://www.nature.com/nature/journal/v529/n7587/full/nature16961.html.

[57] Aravind Srinivas, Allan Jabri, Pieter Abbeel, Sergey Levine, and Chelsea Finn. Universal planning networks. *ArXiv*, abs/1804.00645, 2018.

[58] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018. URL http://incompleteideas.net/book/the-book-2nd.html.

[59] Brijen Thananjeyan, Ashwin Balakrishna, Ugo Rosolia, Felix Li, Rowan McAllister, Joseph E. Gonzalez, Sergey Levine, Francesco Borrelli, and Ken Goldberg. Safety augmented value estimation from demonstrations (saved): Safe deep model-based rl for sparse cost robotic tasks. 2019.

[60] Rishi Veerapaneni, John D. Co-Reyes, Michael Chang, Michael Janner, Chelsea Finn, Jiajun Wu, Joshua B. Tenenbaum, and Sergey Levine. Entity abstraction in visual model-based reinforcement learning. *ArXiv*, abs/1910.12827, 2019.

[61] Ruben Villegas, Arkanath Pathak, Harini Kannan, Dumitru Erhan, Quoc Le, and Honglak Lee. High fidelity video prediction with large stochastic recurrent neural networks. 11 2019.

[62] Angelina Wang, Thanard Kurutach, Kara Liu, Pieter Abbeel, and Aviv Tamar. Learning robotic manipulation through visual planning and acting. *CoRR*, abs/1905.04411, 2019.

[63] Tingwu Wang and Jimmy Ba. Exploring model-based planning with policy networks. *ArXiv*, abs/1906.08649, 2019.

[64] Manuel Watter, Jost Tobias Springenberg, Joschka Boedecker, and Martin A. Riedmiller. Embed to control: A locally linear latent dynamics model for control from raw images. *CoRR*, abs/1506.07365, 2015.

[65] Annie Xie, Frederik Ebert, Sergey Levine, and Chelsea Finn. Improvisation through physical understanding: Using novel objects as tools with visual foresight. *CoRR*, abs/1904.05538, 2019.

[66] Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan R Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. *ArXiv*, abs/1910.10897, 2019.

[67] Tianhe Yu, Gleb Shevchuk, Dorsa Sadigh, and Chelsea Finn. Unsupervised visuomotor control through distributional planning networks. *CoRR*, abs/1902.05542, 2019.

[68] Andy Zeng, Shuran Song, Stefan Welker, Johnny Lee, Alberto Rodriguez, and Thomas A. Funkhouser. Learning synergies between pushing and grasping with self-supervised deep reinforcement learning. *CoRR*, abs/1803.09956, 2018.

[69] Marvin Zhang, Sharad Vikram, Laura Smith, Pieter Abbeel, Matthew J. Johnson, and Sergey Levine. Solar: Deep structured latent representations for model-based reinforcement learning. *ArXiv*, abs/1808.09105, 2018.

[70] Łukasz Kaiser, Mohammad Babaeizadeh, Piotr Miłos, Błażej Osiński, Roy H Campbell, Konrad Czechowski, Dumitru Erhan, Chelsea Finn, Piotr Kozakowski, Sergey Levine, Afroz Mohiuddin, Ryan Sepassi, George Tucker, and Henryk Michalewski. Model based reinforcement learning for atari. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=S1xCPJHtDB.