Action-Conditional Video Prediction with motion-equivariance regularizer

Ruohan Zhang ZHARU@UTEXAS.EDU
William Xie WXIE@CS.UTEXAS.EDU
Department of Computer Science, The University of Texas at Austin

Abstract

This paper studies action-conditional video frame prediction problem for Atari games. We adopt the deep convolution-action-deconvolution architecture in (Oh et al., 2015), and combine it with the equivariance contrastive loss described in (Jayaraman & Grauman, 2015) using a Siamese network. We construct a dataset from four Atari games using two different types of agents under a variety of conditions. We design three deep network based models and found that one of them perform comparably to the previous method. While our models do not show improvement over the original, we learned more about the characteristics of these networks in the process.

1. Introduction

This work explores frame prediction problem in the videos game domain. Given the the current and past frames and the most recent action input, we want to predict the immediate next frame. A model that can accurately predict future is an important component for planning-based control algorithms such as deep reinforcement learning (RL) and tree-search based algorithms. In many vision-based RL problem, e.g., Atari games (Bellemare et al., 2012a), the model for dynamics between the agent and the environment is often unavailable. Learning a predictive model could enable us to learn transition functions in the Markov decision process. This allows us to use model-based RL approaches to tackle these problems. In addition, planning-based algorithms such as Monte Carlo tree search can significantly improve the performance of RL algorithms as shown by the recent successes in applying these concepts in Go and Atari game domains (Silver et al., 2016; Guo et al., 2014).

Futhermore, the tradeoff between exploration and exploitation in RL algorithms affects the convergence speed (Sutton & Barto, 1998). A common way for an agent to do exploration is to select an action randomly. However, a learned predicative model can inform the agent the consequences of actions. Hence, if the agent has an explicit memory, it can choose to go to a state (image frame) which it has not visited before and performs an informed exploration (Oh et al., 2015). However, this technique only significantly improves the deep Q-learning algorithm in one out of five games that they tested. Whether this is due to the quality of prediction is still an unanswered question.

In this paper, we want to combine the techniques from (Oh et al., 2015) and (Jayaraman & Grauman, 2015) for frame prediction. Ohs model trains a convolutional neural networks (CNN) with LSTM layers (Hochreiter & Schmidhuber, 1997) that predict the next video frame of Atari games (Bellemare et al., 2012a), given the current and past frames and the current control action as inputs. We adopt the convolution-action-deconvolution architecture from their work as the basis of our model. However, their network requires requires 500,000 frames per game (9.25 hours of game play at 15Hz) for training. In domains other than video games, such as robot navigation, this amount of data is expensive to acquire. We hope to find a suitable regularize to make this approach more valuable for smaller datasets.

We hypothesize that learning motion equivariance representations will produce better video frame predictions. The goal is to improve the action-conditioned video frame prediction by incorporating motion equivariance constraints.

The work of (Jayaraman & Grauman, 2015) introduces the concept of learning equivariance representation in the latent space for videos and synchronized actions. They use a Siamese network to incorporate a contrastive loss (Bromley et al., 1993; Hadsell et al., 2006; Mobahi et al., 2009), which seeks to learn unique transformations from each actions. Their evaluation is on the scene recognition task with the contrastive loss serves as a regularizer. We want to extend this approach to the video frame prediction task.

This is not a direct transfer of the ego-motion concept to a different domain. Where as (Jayaraman & Grauman, 2015)

emphasizes the ego-motion equivariance features, the motions in Atari games are mostly not ego-centric. In addition, most games have many objects on the screen that are moving but are not directly controlled by the input actions. We want to answer the question of whether by applying motion equivariance constraints could help video prediction with the presence of aforementioned dynamic objects.

The rest of the paper is organized as follows. Section 2 presents related work. In section 3 we show our data collection procedure. Section 4 introduces our method. Section 5 presents experiments and results. Lastly, Section 6 concludes and discusses potential future work.

2. Related Work

Action conditioned video prediction There are many works on action-conditioned predictive models from reinforcement learning research, but they mostly address predictions for high-level states, e.g., locations and positions. Vision-based RL prediction problem was introduced by (Schmidhuber & Huber, 1991). Directly predicting the raw pixel image frames is a relatively new approach (Oh et al., 2015), with the success of deep neural networks. The work (Watter et al., 2015) tried to predict long-term image sequences for control problems using a variety of autoencoders. (Bellemare et al., 2013; 2014) attempt to predict images patches by its neighboring patches. (Lenz et al., 2015) uses a recurrent neural network to predict robot 3D coordinates, as a component of model predictive control algorithm.

The embodied (visuomotor) approach to vision problems One motivation of (Jayaraman & Grauman, 2015) is that visual feature learning can be tied with ego-motions so the consequences of motions will be encoded in the learned feature space. A contemporary and closely related work is (Agrawal et al., 2015). Their method uses ego-motion as an additional supervision signal for feature learning albeit in a more direct setting. Using a Siamese sytle CNN, their system learns the weights to predict the correct transformation given sequential image-motion pairs. Other works also investigate roles ego-motion can play when used in conjunction with ego-centric video streams. (Cohen & Welling, 2014) tries to learn independent, irreducible representation of an object's movement relative to the observer. (Konda & Memisevic, 2015) attempts to predict the underlying action using the visual information alone. Our project follows the same vein as these past works by using the image and the associated action constrained by equivariance to make the right prediction of the next frame. This task is very similar to (Agrawal et al., 2015) by using action as additional supervision from the ego-centric point-of-view.

The Siamese Network The Siamese network (Bromley et al., 1993; Hadsell et al., 2006; Mobahi et al., 2009) allows weight sharing among networks, and is trained using contrastive loss. This network has become a standard technique for deep learning and is applied to signature verification (Bromley et al., 1993), face verification (Taigman et al., 2014), speaker-specific information extraction (Chen & Salman, 2011), etc. The contrastive loss accounts for the negative training samples, and seeks to push learned representation apart from the positive training samples. In our case, the contrastive loss pushes transformations under different actions apart.

3. Data Collection

Due to a recent update of the Atari emulator with some bug fixes (Bellemare et al., 2012a), we do not reuse the data from (Oh et al., 2015). Instead, we collect our own by using two different types of agents on the Atari emulator. The first agent is a random agent, which chooses an action uniformly random from the action set. The second agent is an agent trained using a Monte Carlo Tree Search (MCTS) algorithm, namely the Upper Confidence Bound applied to trees (UCT) (Kocsis & Szepesvári, 2006). The Atari emulator is deterministic if the game seed is fixed, hence one can use UCT to obtain a sequence of moves and perform reasonably well in many of the Atari games. By combining data from both agents, we can obtain long episodes from UTC agent with frames deep in the game state space and cover more breath using the random agent. Ideally, a deep RL agent with ϵ-greedy policy should be used in the future. However, such agent would be time-consuming to train.

Figure 1 shows a sample frame from each game that we use to collect data. Table 3 summarizes characteristics of each game. Note that Battlezone is specifically chosen because its motion is mainly ego-centric. Different games can result in drastically different prediction accuracies in Ohs model. Having a variety of games help us to explore this issue further and see how game characteristics affect the performance with our model.

We record videos frames along with control actions. The data is assumed to be organized into a collection of frame-action pairs:

$$D = \{(x_1, a_1), \cdots, (x_N, a_N)\}_{t=1}^{N} \quad (1)$$

All games are recorded at 15Hz down sampling from the original 60Hz. The missing frames have the same action inputs to the emulator as the previous action. This allows adequate visual change to the output frame as the original games were designed to have slower human inputs in mind. A total of 50,000 frames are recorded for each game. In addition, we gathered data for Battlezone at 6Hz, 2Hz, 1Hz, and 0.5Hz as well.

Game	Freeway	Pong
Background	Fixed	Fixed
Object Entry	Yes	No
Object Leave	Yes	No
Viewpoint	top-down	top-down
Game	Space Invaders	Battlezone
Background	Fixed	Changing
Object Entrance	No	Yes
Object Exit	Yes	Yes
Viewpoint	top-down	1st person

Table 1. These are the game data characteristics for the 4 games. Object Entrance and Exit indicate that whether there are considerable amount of objects consistently enter/leave the game screens.

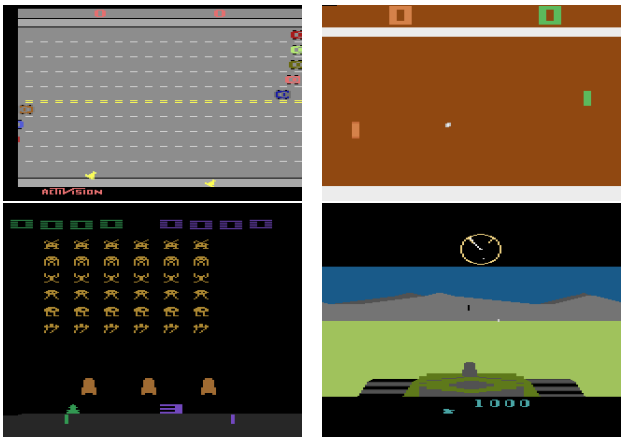

Figure 1. Four Atari games we use. Top left: game Freeway. The object being controlled is the chicken (yellow one on the bottom left). Top right: Pong. The controlled object is the green peddle. Bottom left: Space Invaders. The controlled object is the green spaceship at the bottom. Bottom right: Battlezone. The game is ego-centric.

4. Method

Since the actions in Atari emulator are discrete, we will not need to cluster ego-motion patterns as (Jayaraman & Grauman, 2015) did. We modify the feedforward CNN structure used by (Oh et al., 2015). The original network is shown in Figure 2.

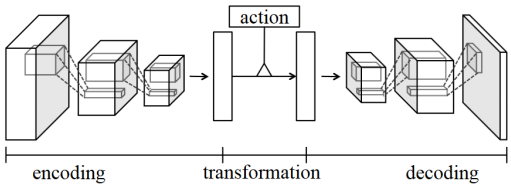

Figure 2. The feedforward only deep CNN structure from (Oh et al., 2015).

The input frame(s) is fed into several convolution layers to get a latent space representation. Then, the action, represented by a one-hot vector, is transformed to the same dimension and combined with the input through point-wise multiplication. A mirror collection of deconvolution layers follows to decode the signal back to the pixel space. The resulting output is the prediction of the next frame. Their loss function is defined as the Euclidean distance between the predicted and actual frame:

$$\mathcal{L}_p(\theta) = \sum_t ||\hat{X}_t - X_t||_2^2 \tag{2}$$

where $\hat{X}$ is the predicted frame and X is the actual frame. θ are convolution and deconvolution network parameters to be learned. Our model combines the above Euclidean loss with a contrastive loss:, so the overall loss is:

$$\mathcal{L}(\theta, M) = \mathcal{L}_p + \lambda \mathcal{L}_c(\theta, M) \tag{3}$$

where λ is a regularizing constant that needs to be carefully tuned. The contrastive loss $\mathcal{L}_c(\theta, M)$ is of the form similar to the one described in Equation (5) and (6) of (Jayaraman & Grauman, 2015):

$$\mathcal{L}_c(\theta, M) = \sum_i d_a(X_t, X_{t+1}, a_t) \tag{4}$$

where M_a is the *equivariance map* matrix for action a and assumed to be linear. $a_{i,t}$ is the action i taken at time t. Raw sequential frames X_t and X_{t+1} are transformed to the latent feature space through convolution layers. The contrastive loss for action a that progresses X_t to X_{t+1} is defined as follows:

$$\begin{aligned} d_a(X_t, X_{t+1}, a_t) = \mathbb{1}(a_t = a) \\ ||M_a conv(X_t) - conv(X_{t+1})||_2^2 \\ + \mathbb{1}(a_t \neq a) \\ max(\delta - ||M_a conv(X_t), conv(X_{t+1})||_2^2, 0) \end{aligned} \tag{5}$$

Intuitively, optimization of the loss function bring X_t and X_{t+1} together in the latent feature space given the correct action transformation. For all other action transformations, they are pushed apart by a fixed margin δ. Since Equation 3 combines two loss together, we jointly optimize for θ, M. The network will try to obtain a high prediction quality, meanwhile attempts to ensure that different actions will map the same input frame into different representations in the latent space. The network architecture that implements the loss functions in Equations 3, 4, and 5 is shown in Figure 3. The parameters for convolution layers are shared forming a Siamese network.

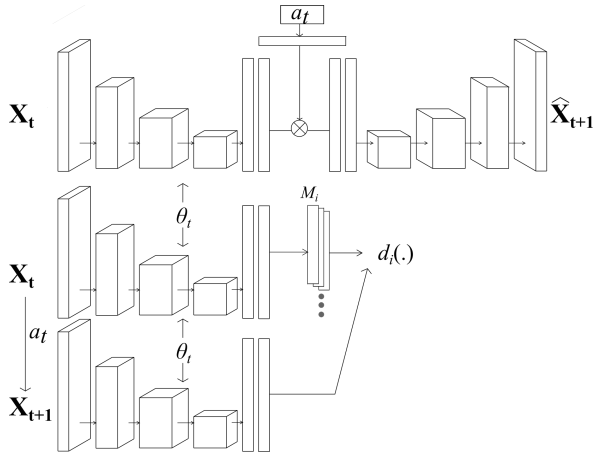

Figure 3. Model 1 architecture. The convolution layers are shared and hence the network is a Siamese network. X_t is the input frame(s). a_t is the current action. X_{t+1} is the next frame. $\hat{X}_{t+1}$ is the prediction. M_is are the equivariance maps.

In addition to the previously described architecture, we designed another two novel variations for the same prediction task. The intuition is that instead of applying the contrastive loss in latent feature space, one can apply it in the reconstructed pixel space. i.e., given different actions, we want the network to produce different next frames. The new loss function is then just:

$$\mathcal{L}(\theta) = \mathcal{L}_c(\theta) = \sum_i d_a(\hat{X}_{a_i,t+1}, X_{t+1}, a_{i,t}) \quad (6)$$

since the contrastive loss counts for the prediction loss in the positive mode. And the new contrastive loss is hence:

$$d_a(\hat{X}_{a_i,t+1}, X_{t+1}, a_i) = \mathbb{1}(a_i = a)||\hat{X}_{a_i,t+1} - X_{t+1}||_2^2 + \mathbb{1}(a_i \neq a)max(\delta - ||\hat{X}_{a_i,t+1} - X_{t+1}||_2^2, 0) \quad (7)$$

Our second model is shown in Figure 4, where the all the actions are combined with the input. Weights from the deconvolution layers are shared between the networks. The contrastive loss pushes away predictions by the wrong actions.

A slightly different architecture that implements the same loss function is shown in Figure 5. The main difference is that no action is used in the latent space and deconvolution weights are not shared. Instead, each action learns its own deconvolution layers.

For all networks, the input X_t can have multiple frames. (Oh et al., 2015) finds that using frames from the past can help predictions. Hence, X_t can be a stack of the current frame and immediate previous frames. We use both 1 and 4 frame stacks are input for our experiments.

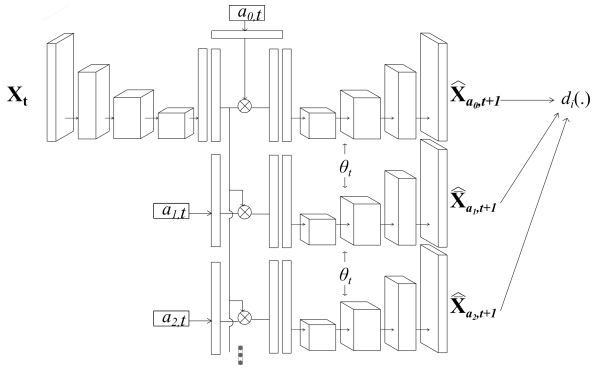

Figure 4. Model 2 architecture. The deconvolution layers are both shared. Different actions are used for transformation.

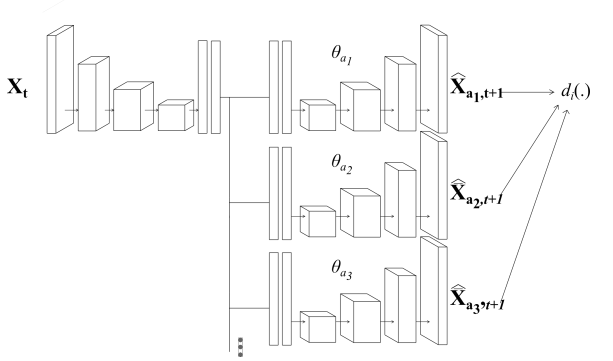

Figure 5. Model 3 architecture. Each action has its own deconvolution layers.

All our models are implemented using the Caffe deep learning library (Jia et al., 2014). All networks are trained using Adam solver (Kingma & Ba, 2014) with default parameters. The baseline learning rate ranges from $1e{-}4$ to $1e{-}6$. The learning rate decays by steps with the step size varying depending on experimental conditions. The minimum distance δ in contrastive loss ranges from 0.01 to 1. The regularizing constant λ ranges from 0.01 to 0.1.

5. Experiments and Results

First, we replicate the model in Figure 2 (Oh et al., 2015), which we will refer to as Oh's model. We train another baseline that is not action-conditional, i.e., we do not feed action into the latent space for the architecture in Ohs model. We will refer this as the no-act baseline.

5.1. Results

Unfortunately, the networks in Figure 4 and 5 do not converge to a satisfactory loss magnitude even after extensive modification to both margin distance δ and learning rate. Figure 6 shows the loss function over iterations. The output of both network resembles that of the no-act baseline.

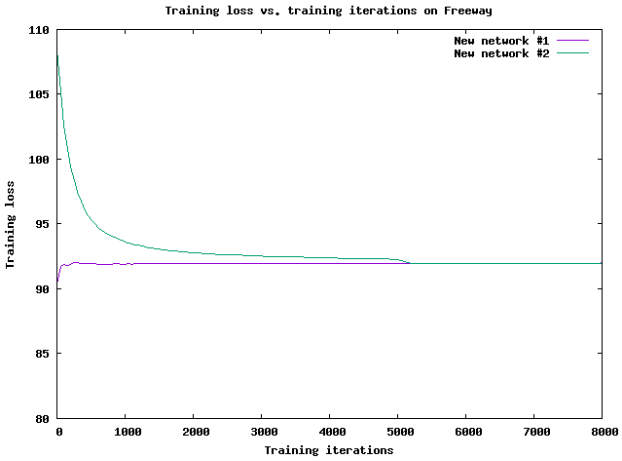

Figure 6. The training loss for our newly proposed network architectures never converge.

We will discuss the potential cause of this phenomenon in the Section 6.

We first show some qualitative comparisons between prediction results of no-act baseline, our model 1 (Figure 3), Oh's model, and the ground truth for each game. We experiment with dataset sizes of 1k, 2k, 10k, and 100k, where training and testing data are evenly split. Figures 7, 8, 9, and 10 show results of the models trained on 2k images. Videos of the results can be in the following links: youtu.be/AMl5mNzwP0c (1 frame) and youtu.be/SApB-rWibhg (4 frame).

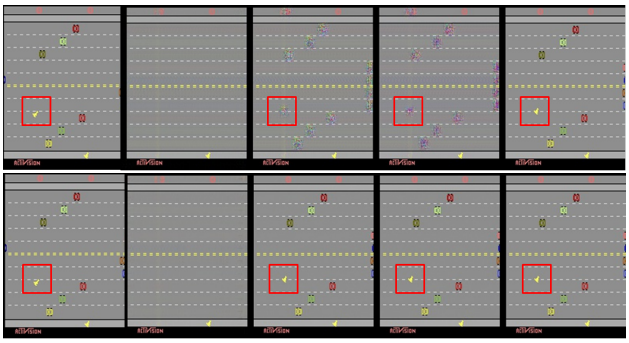

Figure 7. Prediction results for game Freeway. The action here is UP. Top row: 1 frame input. Bottom row: 4 frames input. Each row from left to right: current frame; no-act baseline prediction; our model's prediction; Oh's model's prediction; and the ground truth. The red boxes highlight the object being controlled, and they are placed at the same position to help visualize the consequences of the action. The results for the rest of the games will be presented in the same manner.

The quantitative results in average mean squared error (MSE) for each model can be found in Table 2. Note that using 1 frame input, the MSE for no-act can be better than

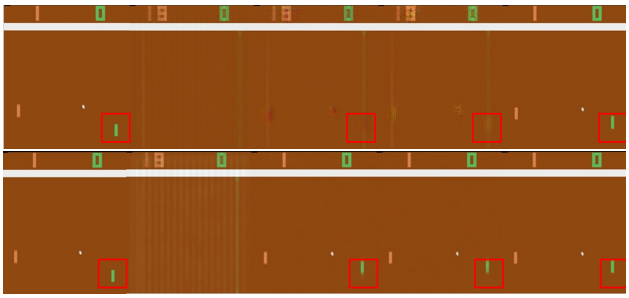

Figure 8. Prediction results for Pong. The action here is UP.

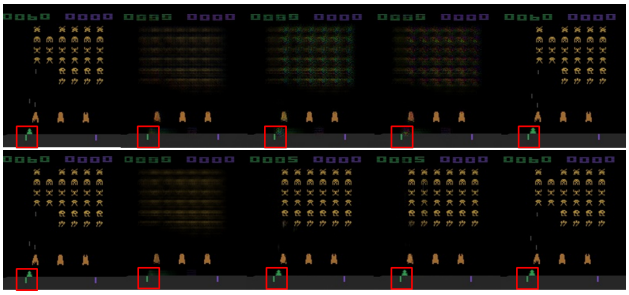

Figure 9. Prediction results for Space Invaders. The action here is RIGHT.

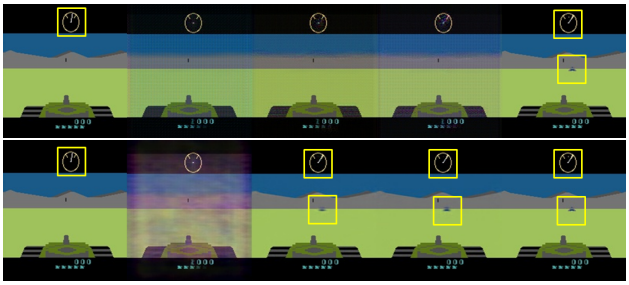

Figure 10. Prediction results for game Battle Zone. The action here is RIGHT. Yellow boxes highlight the major visual differences after taking the action.

Oh's model and our model. However, when we visualize the results, no moving objects from no-act prediction is visible. Both Ohs and our models captures blurry regression of where the object could be. This might be because the predicted blurs do to retain the original colors of the moving objects well. MSE simply did not fully capture this these features.

We then hypothesize that maybe our tasks are not difficult enough for Oh's model. Hence we choose a game like Battle Zone and use data collected at much lower frame rates at 6Hz, 2Hz, 1Hz, and 0.5Hz. Now a single action would change the scene drastically. Lower frequency makes the prediction more challenging and degrades the performance for all methods. Table 3 shows the average MSE results. Other than for 0.5hz where our model fails to converge,

	1 frame			4 frame		
	No action	Oh et al.	Ours	No action	Oh et al.	Ours
Freeway	**9.77** $\pm$ 0.17	15.64 $\pm$ 0.53	15.16 $\pm$ 0.55	4.20 $\pm$ 0.22	**0.49** $\pm$ 0.05	0.57 $\pm$ 0.08
Battle Zone	**47.33** $\pm$ 9.94	54.02 $\pm$ 9.56	57.07 $\pm$ 6.88	54.85 $\pm$ 9.14	**13.51** $\pm$ 21.05	16.83 $\pm$ 20.89
Space Invaders	26.54 $\pm$ 0.96	25.90 $\pm$ 1.03	**25.07** $\pm$ 0.94	24.67 $\pm$ 1.06	8.16 $\pm$ 4.00	**7.88** $\pm$ 5.48
Pong	15.86 $\pm$ 0.67	**5.62** $\pm$ 0.43	6.51 $\pm$ 0.60	20.93 $\pm$ 1.28	0.92 $\pm$ 0.16	**0.86** $\pm$ 0.17

Table 2. The average mean square error across the test set.

	No action	Oh et al.	Ours
Battle Zone 15 hz	54.85 $\pm$ 9.14	**13.51** $\pm$ 21.05	16.83 $\pm$ 20.89
Battle Zone 6 hz	93.16 $\pm$ 1.69	14.17 $\pm$ 21.18	**14.02** $\pm$ 21.60
Battle Zone 2 hz	64.67 $\pm$ 3.68	**14.18** $\pm$ 20.44	15.64 $\pm$ 19.57
Battle Zone 1 hz	70.47 $\pm$ 2.05	**15.02** $\pm$ 22.04	15.68 $\pm$ 22.90
Battle Zone 0.5 hz	67.63 $\pm$ 2.27	**18.08** $\pm$ 23.34	42.15 $\pm$ 9.40

Table 3. The average mean square error for Battle Zone at different frame rates with 4 frame input.

Ohs model still performs slightly better on average. However, the gap becomes narrower than before. Video of the results can be found here: youtu.be/-DYt_G0GJBE.

The main conclusions for our experiments are: first, prediction with 1 frame as input is not sufficient to reproduce the image but nonetheless captures the moving objects. Using a stack of 4 frames significantly improves prediction accuracy. Second, our model 1 performs comparable to Oh's model, where both outperform the no-act baseline. No-act baseline shows its inability to capture any moving objects, which confirms the importance of action inputs in video prediction tasks. Surprisingly, even with much smaller amount of data (original dataset size 500k), Oh's model can still perform really well. In addition, we have several observations through visualization. The 4-frame version can accurately predict score changes in Pong, since the scores occupy a large region in Pong. Both our model and Oh's model cannot accurately predict small objects, such as the ball in Pong and the bullets in Space Invaders. In addition, both models can predict object entrance, such as the enemy's tank in Battlezone. But they do not handle object exits very well, such as the aliens in Space Invaders.

6. Conclusion and Future Work

Our new models do not show advantage over the ones described in (Oh et al., 2015) and they are more difficult to train. In addition, two of our networks do not converge well. Perhaps Atari games might not be the most suitable domain to apply these models due to the conflicting goals for prediction loss and contrastive loss. The contrastive loss tries to push representations or predictions for each action apart and apply structure to the underlying encoding. However, this distance is often very small in Freeway, Pong, Space Invaders and even Battle Zone as the changes resulting from the actions are limited. The conflict arises because the prediction loss attempts to get the most of the pixels correct, meanwhile contrastive loss exaggerates the small differences resulted from actions. We hypothesize that a dataset where actions will result in big changes in pixel space, e.g., ego-motion dataset KITTI (Geiger et al., 2012; 2013), or other robot navigation datasets, will be more suitable.

For our future work, given more computational resource and time, we first want to perform more experiments in a systematic way, e.g., exhaustive search on hyperparameters and do MSE comparison on all combinations, varying dataset sizes, try on some other games. Doing so will give us more convincing results that could explain the feasibility as well as characteristics of the models.

One issue with our model and Oh's model is that they both fail to capture small objects such as the bullets in Space Invader and Battle Zone or the ball in Pong. However, these objects are important for control tasks because they correlate highly with rewards. Reward-conditional video prediction may be tackled by introducing another prior based on reward. This would highlight the objects or regions that are tied to reward signals.

Another potential future work is to distinguish objects controlled by the actions from those who are not. This is already a problem already mentioned by (Oh et al., 2015; Bellemare et al., 2012b) in the Atari domain. In the current vision-based control literature, the input state space is often the raw pixel space. To tell apart controlled versus uncontrolled objects is a challenging problem. It is still unclear how we can predict controlled objects directly from our model.

References

Agrawal, Pulkit, Carreira, Joao, and Malik, Jitendra. Learning to see by moving. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 37–45, 2015.

Bellemare, Marc, Veness, Joel, and Bowling, Michael. Bayesian learning of recursively factored environments. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pp. 1211–1219, 2013.

Bellemare, Marc, Veness, Joel, and Talvitie, Erik. Skip context tree switching. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pp. 1458–1466, 2014.

Bellemare, Marc G, Naddaf, Yavar, Veness, Joel, and Bowling, Michael. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 2012a.

Bellemare, Marc G, Veness, Joel, and Bowling, Michael. Investigating contingency awareness using atari 2600 games. In *AAAI*, 2012b.

Bromley, Jane, Bentz, James W, Bottou, Léon, Guyon, Isabelle, LeCun, Yann, Moore, Cliff, Säckinger, Eduard, and Shah, Roopak. Signature verification using a siamese time delay neural network. *International Journal of Pattern Recognition and Artificial Intelligence*, 7 (04):669–688, 1993.

Chen, Ke and Salman, Ahmad. Extracting speaker-specific information with a regularized siamese deep network. In *Advances in Neural Information Processing Systems*, pp. 298–306, 2011.

Cohen, Taco S and Welling, Max. Transformation properties of learned visual representations. *arXiv preprint arXiv:1412.7659*, 2014.

Geiger, Andreas, Lenz, Philip, and Urtasun, Raquel. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pp. 3354–3361. IEEE, 2012.

Geiger, Andreas, Lenz, Philip, Stiller, Christoph, and Urtasun, Raquel. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, pp. 0278364913491297, 2013.

Guo, Xiaoxiao, Singh, Satinder, Lee, Honglak, Lewis, Richard L, and Wang, Xiaoshi. Deep learning for real-time atari game play using offline monte-carlo tree search planning. In *Advances in Neural Information Processing Systems*, pp. 3338–3346, 2014.

Hadsell, Raia, Chopra, Sumit, and LeCun, Yann. Dimensionality reduction by learning an invariant mapping. In *Computer vision and pattern recognition, 2006 IEEE computer society conference on*, volume 2, pp. 1735–1742. IEEE, 2006.

Hochreiter, Sepp and Schmidhuber, Jürgen. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Jayaraman, Dinesh and Grauman, Kristen. Learning image representations tied to ego-motion. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1413–1421, 2015.

Jia, Yangqing, Shelhamer, Evan, Donahue, Jeff, Karayev, Sergey, Long, Jonathan, Girshick, Ross, Guadarrama, Sergio, and Darrell, Trevor. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

Kingma, Diederik and Ba, Jimmy. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kocsis, Levente and Szepesvári, Csaba. Bandit based monte-carlo planning. In *Machine Learning: ECML 2006*, pp. 282–293. Springer, 2006.

Konda, Kishore and Memisevic, Roland. Learning visual odometry with a convolutional network. In *International Conference on Computer Vision Theory and Applications*. Citeseer, 2015.

Lenz, Ian, Knepper, Ross, and Saxena, Ashutosh. Deepmpc: Learning deep latent features for model predictive control. In *Robotics Science and Systems (RSS)*, 2015.

Mobahi, Hossein, Collobert, Ronan, and Weston, Jason. Deep learning from temporal coherence in video. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 737–744. ACM, 2009.

Oh, Junhyuk, Guo, Xiaoxiao, Lee, Honglak, Lewis, Richard L, and Singh, Satinder. Action-conditional video prediction using deep networks in atari games. In *Advances in Neural Information Processing Systems*, pp. 2845–2853, 2015.

Schmidhuber, Juergen and Huber, Rudolf. Learning to generate artificial fovea trajectories for target detection. *International Journal of Neural Systems*, 2(01n02):125–134, 1991.

Silver, David, Huang, Aja, Maddison, Chris J, Guez, Arthur, Sifre, Laurent, van den Driessche, George,

Schrittwieser, Julian, Antonoglou, Ioannis, Panneershelvam, Veda, Lanctot, Marc, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

Sutton, Richard S and Barto, Andrew G. *Introduction to reinforcement learning*, volume 135. MIT Press Cambridge, 1998.

Taigman, Yaniv, Yang, Ming, Ranzato, Marc'Aurelio, and Wolf, Lior. Deepface: Closing the gap to human-level performance in face verification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1701–1708, 2014.

Watter, Manuel, Springenberg, Jost, Boedecker, Joschka, and Riedmiller, Martin. Embed to control: A locally linear latent dynamics model for control from raw images. In *Advances in Neural Information Processing Systems*, pp. 2728–2736, 2015.