Unsupervised Dynamic Learning with GANs

Yiding Jiang * Allan Zhou †

May 2017

1 Introduction

In reinforcement learning, learning dynamics is often a central challenge. Being able to predict the future state of the environment given the current state and the actions taken is useful when planning. In computer vision, unsupervised representation learning is also interest of on-going research. In the real world, this often means learning how physical interaction works– what happens when a robot moves its arm into an object? Video data presents an opportunity for agents to learn from existing video datasets and their own interactions as to how physical interaction works. By using video data in tandem with action data, an agent can learn a model of the world through action-conditional video prediction [1, 2].

There has been recent work developing video prediction models that explicitly model pixel motion that was released alongside a robot pushing dataset (containing both video and state-action data) [1]. That work introduces Dynamic Neural Advection (DNA) and Convolutional Dynamic Neural Advection (CDNA) to predict pixel motion. The former calculates each pixel of a predicted frame as an expectation of nearby pixels from the input frame under a distribution outputted by the neural network. The latter generates a small number of distributions each modeling the motion of an entire object instead of the motion of its individual pixels. These are each applied to the entire image, and the multiple outputs are composited together using learned masks. There has also been work using conditional Generative Adversarial Networks (GANs)[3, 4] to learn from images [5] and to approach video prediction, which has shown some success in removing the blur that arises from uncertainty in models trained on mean-squared error [6, 7, 8]. With GANs, two networks are trained. The generator network produces predictions and is trained to fool a discriminator network, which itself is trained to distinguish generated output from real data. In the conditional GAN setting, the generator and discriminator both receive input frames and actions (in the action-conditioned case).

In our work, we combine the techniques for explicitly modeling pixel motion and optic flows with the adversarial approach, and evaluate their effectiveness in learning physical interactions and dynamic of the agent. We introduce two separate models: a single-frame prediction model that recursively predict the future frame and a multi-frame prediction model that uses 3D convolutions to capture the time dependencies of frames. Following the GAN approach, we trained discriminator networks to distinguish generated video from real video, and the generator networks are trained on a combination of mean squared error to ground truth and adversarial loss. In order to learn action-conditional video prediction, the models use both history frames and state-action data as inputs. Demo of this project can be found at https://github.com/yidingjiang/Action_Conditioned_GAN_demo.

2 Models

2.1 Adversarial Loss

Both the single frame and multi-frame models use a generator network $\mathcal{G}$ and a discriminator network $\mathcal{D}$, trained adversarially. The discriminator was trained using the following loss with one-sided soft labeling,

*yjiang9603@berkeley.edu, 3rd year EECS undergrad
†allan.zhou@berkeley.edu, 3rd year EECS undergrad

which helps stabilize the unpredictable behavior of binary cross-entropy [9]:

$$L_{\mathcal{D}} = L_{bce}(\mathcal{D}(x, a, y), 0.9) + L_{bce}(\mathcal{D}(x, a, \mathcal{G}(x, a)), 0) \tag{1}$$

where x refers to the input frame(s), a refers to the state-action data, y refers to the ground truth next frame(s), and L_{bce} refers to binary cross-entropy. The generator's adversarial loss is thus:

$$L_{\mathcal{G},adv} = L_{bce}(\mathcal{D}(x, a, \mathcal{G}(x, a)), 1) \tag{2}$$

Similar to [6], we trained our models not purely on adversarial loss, but on a hybrid where adversarial loss was combined with 1-norm or 2-norm loss to y_i:

$$L_{\mathcal{G}} = \lambda_{adv} L_{\mathcal{G},adv} + \lambda_{lp} L_{\mathcal{G},lp} \tag{3}$$

Although we are not using Wasserstein loss, we found that weight clipping in the discriminator helped improve the GAN training process [10]. Further we define $L_{\mathcal{G},lp}$ as the p^{th} matrix norm of the difference between the generated frame and the ground truth:

$$L_{\mathcal{G},lp} = ||\mathcal{G}(x, a) - y||_p^p$$

2.2 Single-frame prediction

Prior work on video prediction conditioned on agent action has been based on auto-encoder style architectures [2] trained on Atari games, or convolutional LSTMs [1] and pixelCNNs [11], which have been tested on the robot pushing dataset. Our single frame prediction model aims to predict the next frame conditioned on the current frame, the state of the agent (robot arm) and the action applied during the current frame. This architecture can achieve multi-frame prediction by recursively feeding the output from one time step back into the generator in order to predict the next timestep. The rationale is that if an agent understands the relationship between different actions and their corresponding visual feedback, it should be able to infer the next time step without having to rely on previous experiences. This is under the assumptions that the movements of the objects in the video are relatively slow and, therefore, the momentum does not significantly impact the trajectories of objects in the scene.

In a similar light, we give the generator an auxiliary task to predict the state of the robot arm after applying the action, in addition to predicting the next frame. There are several major benefits to performing the auxiliary task. First, predicting the next state of the robot ensures that information about changing states is preserved over the transformation. Second, predicting the future states allows the network to exploit domain commonalities between two closely related tasks and to achieve better results for both. Lastly, since the multi-frame prediction is done by recursively transforming the output of the network, the dynamics that the network predicts can deviate from the ground truth. By using the state predicted by the network over the ground truth, we allow the network to use states that are consistent with its internal representation. This idea of multi-task learning has been used in a wide variety of settings with convnets [12, 13, 14, 15]. In practice, we observed that the auxiliary task makes the training more stable and produces better outputs.

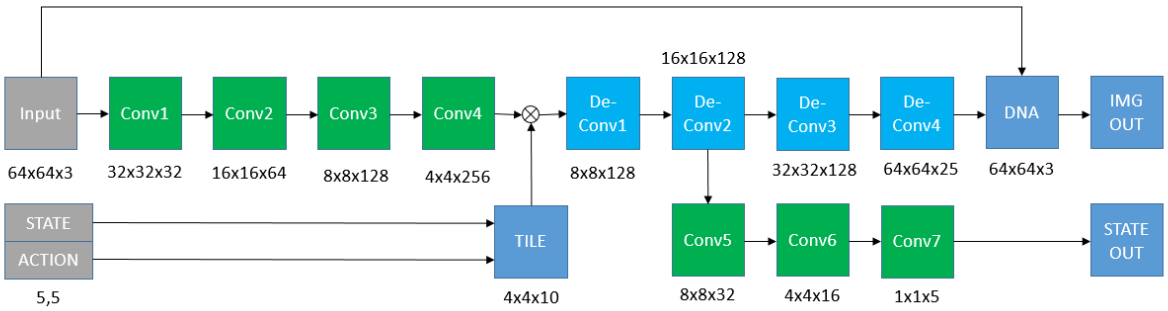
The architectures of the generator and the discriminator are shown in 1 and 2. The generator has an auto-encoder architecture where the actions and states are tiled and concatenated along the channels at the smallest feature map at convolution 4. At deconvolution 2, the architecture branches out and use a small 3-layer convolution network to predict the new state. The generator uses DNA: the output of the final Deconvolution layer is applied to the input frame to produce the transformed output.

In DNA, for every pixel of the predicted image, the generator outputs a distribution m over pixel locations in the input image, which is then used to transform the input image x into a prediction $\hat{y}$. Each distribution has limited spatial extent, say $k \times k$, and each pixel of the predicted image is computed as an expectation:

$$\hat{y}_{x,y} = \sum_{i=-k+1}^{k-1} \sum_{j=-k+1}^{k-1} m_{x,y}(i, j) x_{x-i,y-j} \tag{4}$$

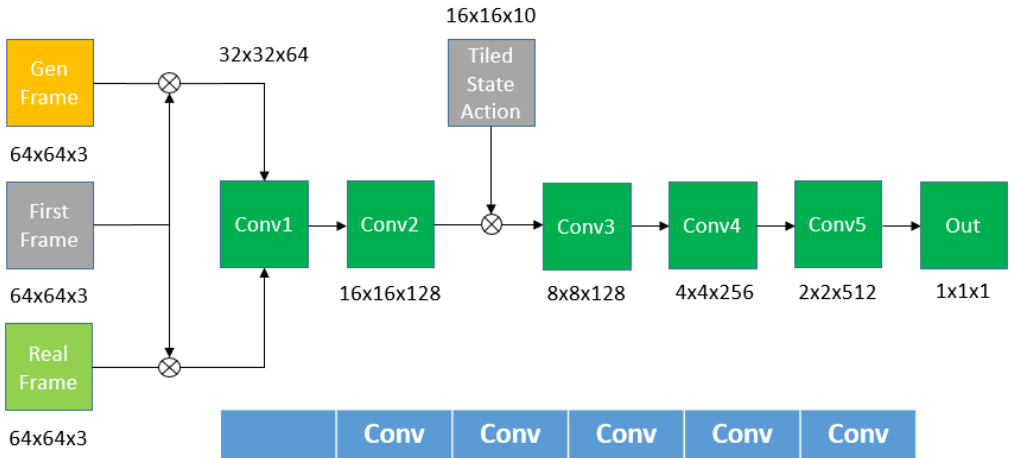
In the single frame model, we normalize through softmax similar to [16] to maximize cross-entropy and encourage sparsity in the distribution.

The discriminator takes in the generated frame and the real frame and stacks them along the channel of the input frame. The discriminator also has access to the initial state and actions at convolution 2 so it can determine the likelihood of the two consecutive frames.



	Conv 1	Conv 2	Conv 3	Conv 4	Conv 5	Conv 6	Conv 7	Deconv 1	Deconv 2	Deconv 3	Deconv 4
Kernel	5x5	5x5	5x5	5x5	3x3	3x3	4x4	5x5	5x5	5x5	5x5
Stride	2x2	2x2	2x2	2x2	2x2	2x2	1x1	2x2	2x2	2x2	2x2

Figure 1: The single-frame action-conditional DNA **generator** architecture with ReLU activation and Batchnorm at all hidden layers



	Conv 1	Conv 2	Conv 3	Conv 4	Conv 5
Kernel	5x5	5x5	5x5	5x5	2x2
Stride	2x2	2x2	2x2	2x2	1x1

Figure 2: The single-frame action-conditional **discriminator** architecture with LeakyReLU activation and Batchnorm at all hidden layers

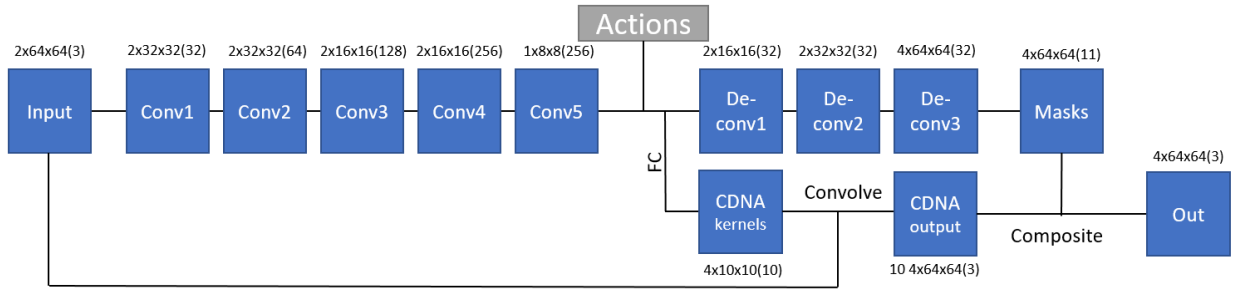
2.3 Multi-frame prediction

Prior work on adversarial video prediction that explicitly models motion [8] uses 3D convolutions to input and output multiple frames at once and uses a DNA-like approach to model the motion of pixels. We implemented the architecture from that work (the DNA model) with action conditions by concatenating tiled state-action data with the output of the 5th convolution layer of the generator along the channel dimension. We also reduced the size of the output from a 12 timestep prediction to a 4 time-step prediction.

In addition, we developed a similar architecture with 3D convolutions that uses the CDNA approach to model the motion of objects rather than just using individual pixels. In the CDNA model, instead of outputting the distribution for every output pixel like it does in DNA, the generator outputs a few distributions which are each applied to the entire input image, and the resulting transformed images are then composited together by learned masks. This takes advantage of the fact that the pixels of rigid objects tend to move together, and thus learning a distribution for every pixel is unnecessary.

The CDNA generator architecture is outlined in Figure 3. It takes 2 frames of history as input and outputs predictions for the next 4 timesteps. Similar to the CDNA architecture of [1], the network produces 10 CDNA kernels with dimension 4x10x10 (time, width, height), which are applied via convolution over the entire 2nd input image to produce 10 transformed outputs. Three deconvolution layers then produce the 10 masks used to composite the transformed outputs together (there is an 11th mask for the input image, allowing the network to directly copy pixels from the input). The masks are all softmax normalized such that the sum of the masks is 1 for every pixel. The composition happens as pointwise multiplication between each mask and its corresponding transformed output, and the results are summed together. As in the DNA generator, state-action data is tiled in the spatial dimensions and then concatenated with the output of the 5th convolution layer.

The discriminator network is the same for both the DNA and the CDNA implementations: a 5 layer convnet similar to the first 5 layers of the CDNA generator, with scalar output and sigmoid activation. The state-action data is tiled and concatenated along the channel dimension of the input. All hidden layers of both networks use ReLU activation, and are batch normalized. [17].



	Conv1	Conv2	Conv3	Conv4	Conv5	Deconv1	Deconv2	Deconv3	Mask
Filter	1x3x3	1x3x3	1x3x3	1x3x3	2x3x3	2x3x3	1x3x3	2x3x3	1x1x1
Stride	1x2x2	1x1x1	1x2x2	1x1x1	2x2x2	2x2x2	1x2x2	2x2x2	1x1x1

Figure 3: The multi-frame action-conditional CDNA **generator** architecture. Each layer is a 3D convolution/deconvolution with 'same' padding, with output shape labeled above in **time** x **width** x **height** (# channels) order. The filter sizes and strides are specified in the table below. ReLU activations are used at every layer, except after the "Masks" layer, which uses a softmax activation. Batch normalization is applied after every layer. The **discriminator** architecture is a 5-layer convnet similar to the first 5 layers of the generator.

3 Experiments

3.1 Dataset

We use Google Brain's robot pushing dataset for all experiments [1]. This dataset contains both video and state-action data of robots completing pushing tasks. The video data is down-sampled to 64x64 spatial resolution with area interpolation, since all of our models work at that resolution. The models are trained on the "push_train" group from this dataset and tested on "push_testseen" group from this dataset.

3.2 Implementation

All models are implemented in Tensorflow [18] and use the Adam optimizer[19].

For the single frame prediction model, we used a learning rate of .001 in both the generator and the discriminator, and set $\lambda_{adv} = 1, \lambda_{lp} = 0.05$. We also set the order of $L_{\mathcal{G},lp}$ to 1. In addition, we penalized the generator with the l_2 difference between the predicted state and the actual state with $\lambda_{state} = 1$. The single frame model is trained for more than 30000 iterations on random frames chosen between the 6^{th} and 20^{th} frames, with every other frame being skipped. Experimentally, we found that this setting gives an appropriate difference between frames to train.

For the multi-frame prediction models, we used a learning rate of .0002 in both the generator and the discriminator, and set $\lambda_{adv} = \lambda_{lp} = 1$. For the $L_{\mathcal{G},lp}$ loss we used $p = 2$. Each multi-frame model was trained for 5000 iterations on the first 6 timesteps of video (2 input frames, 4 output frames). Note that although the DNA model as originally proposed in [8] was trained with purely adversarial loss, we used a combined loss here (which we found worked better with this dataset).

4 Results

All models were tested on the same test data drawn from the "push_testseen" group of the robot push dataset. Due to how they were trained, the single-frame architecture is tested with input starting 6 timesteps with 1 frame skip into the video, while the multi-frame architecture is tested with input at the beginning of the video.

4.1 Metrics

To evaluate the performance of our model, we adapted the Structural Similarity Index (SSIM) and Peak Signal to Noise Ratio (PSNR) used in [1]. Quantitative results are plotted in Figure 4, using SSIM and PSNR as metrics, with the identity (returning the input as the prediction) plotted for comparison. Predictions are made multiple timesteps into the future by recursively feeding the output from the previous timestep back in as input to the model.

We could not reproduce the result of [1] and [11] due to computational limitations as both LSTMs and PixelCNNs take very long times to train. In [11], they also use negative log likelihood, which GANs do not explicitly model. Furthermore, the number of frames being predicted is also different, so the comparison will not be meaningful; however, we believe that visually, our approach has to some extent addressed the issue of blur in [1]. With regards to PixelCNN, although [11] generates perfectly realistic samples, its sampling speed prevents it from scaling, whereas our feed forward model generates samples at a much faster speed.

We also observed that the qualitative improvement in the samples is not necessarily reflected in the quantitative metrics. This may be due to the inability of existing metrics (e.g. SSIM and PSNR) to capture the problems that generative models are trying to address. Metrics like Inception score also cannot be used due to the similarities between the generated frame and the real frame. We suspect that this is because these metrics do not accurately capture the way that our visual system compares images. One theoretically ideal metric could be Earth Mover Distance or Wasserstein Distance, but its large complexity makes it computationally infeasible. This can be tested when faster methods for the EMD become available.

4.2 Single-frame predictions

Some samples from the single-frame model are shown in Figures 5 and 6, where they are compared to the ground truth. We have also included the model output when given the same image input but conditioned on double the action input or a negated action input (roughly, twice as much action or reversed action). Although we did not train the network on these actions, we observed that the network is actually able to correctly interpret the meaning behind these actions and to generate the predictions accordingly on the held-out data set. This is particularly obvious in the case of reversed action. For the 2x action, although the effect is not always as straight-forward, it is clear that the difference in action has effects on the generated future.

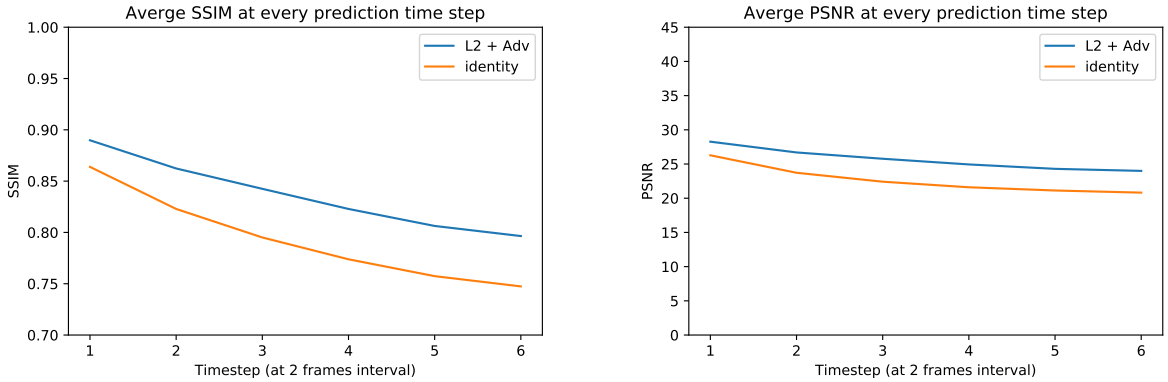


Figure 4: Structural similarity index (SSIM) and Peak Signal to Noise Ratio (PSNR) of the single-frame model's predictions on test data with the ground truth, averaged for the prediction timestep. "Identity" refers to taking the input frame and using it as the prediction.

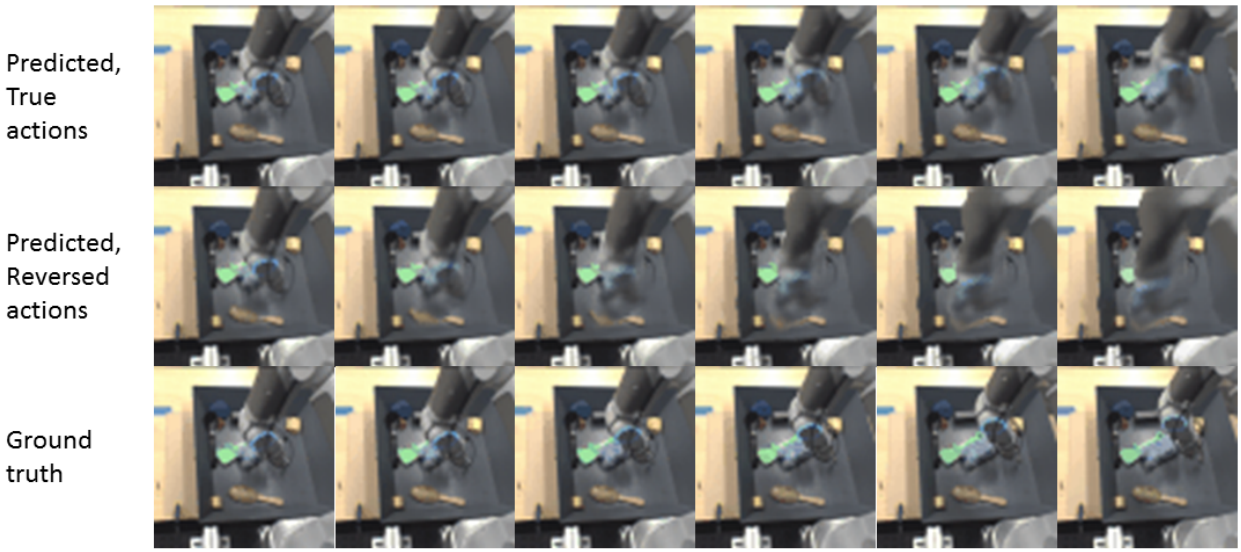


Figure 5: Single frame output, including with action input reversed. Notice the arm moves in the opposite direction compared to when the action input is not reversed (true). More sample GIF's can be found here.

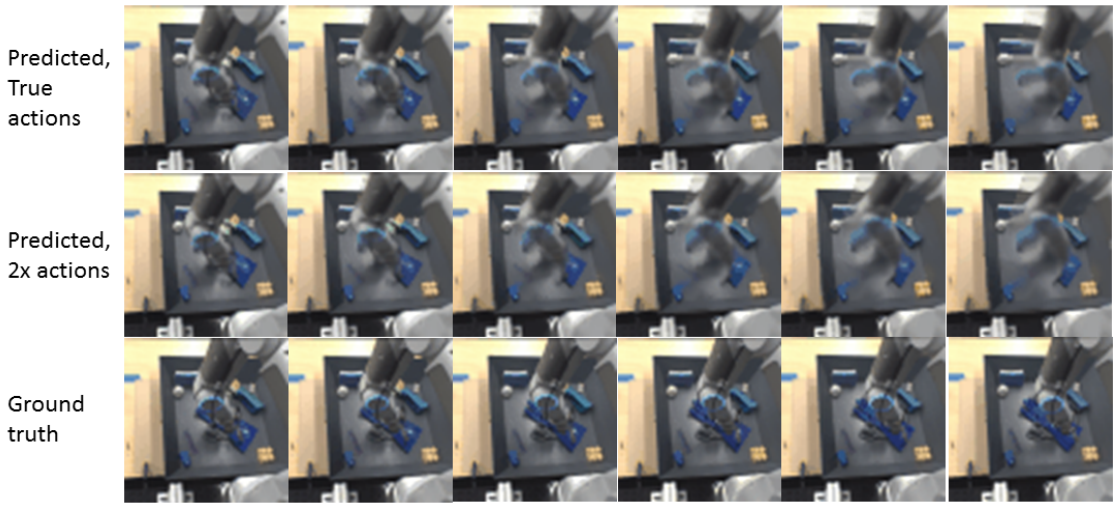


Figure 6: Single frame output, including with action input doubled. The effect of doubling the action is not immediately obvious, but the output is different compared to with true action input. More sample GIF's can be found here.

4.3 Multi-frame predictions

Some qualitative results from the multi-frame models are presented in Figure 8, showing the 4th prediction timestep for different models compared to the ground truth. Figure 8 also shows the result of training the CDNA model solely on Mean Squared error (CDNA L2), or $\lambda_{adv} = 0$. When combined with adversarial loss (Hybrid), there is less blurring in the outputs of both models (DNA and CDNA). However, they both still introduce other artifacts. For example, the DNA model tends to break the arm. Quantitative results, again using Peak Signal to Noise Ration (PSNR) and Structural Similarity Index (SSIM), are presented in Figure 7. The performance of the identity (taking the last input frame and using it as the prediction) is shown for comparison. The DNA model performs the best in both metrics. The CDNA model has a similar performance to that of the identity, though this may partly be because the predictions are limited to the first 6 timesteps, which tend to have less motion.

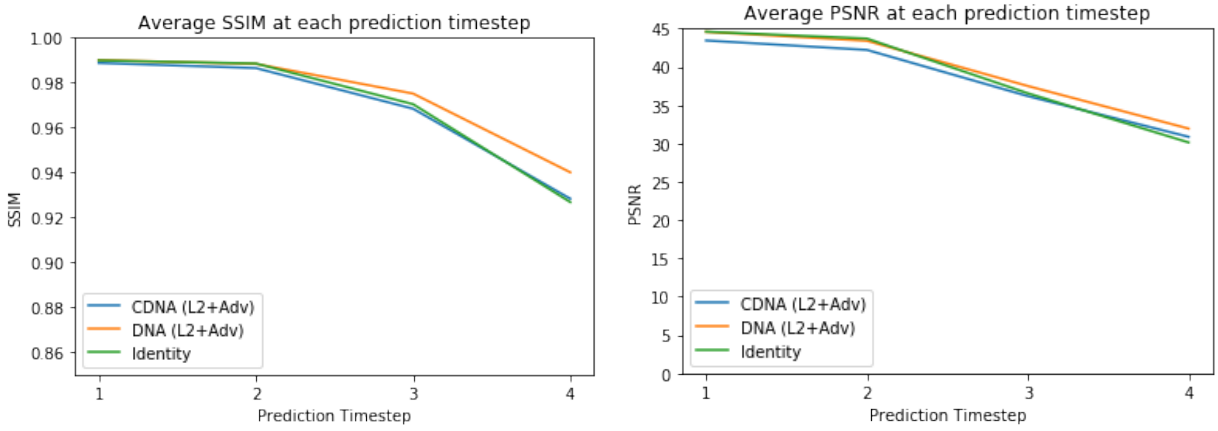


Figure 7: Structural similarity index (SSIM) and Peak Signal to Noise Ratio (PSNR) of each multi-frame model's predictions on test data with the ground truth, averaged for the prediction timestep (thus t=4 means 4 timesteps after the 2nd input frame). The "Identity" simply refers to taking the 2nd input frame and using it as the prediction.

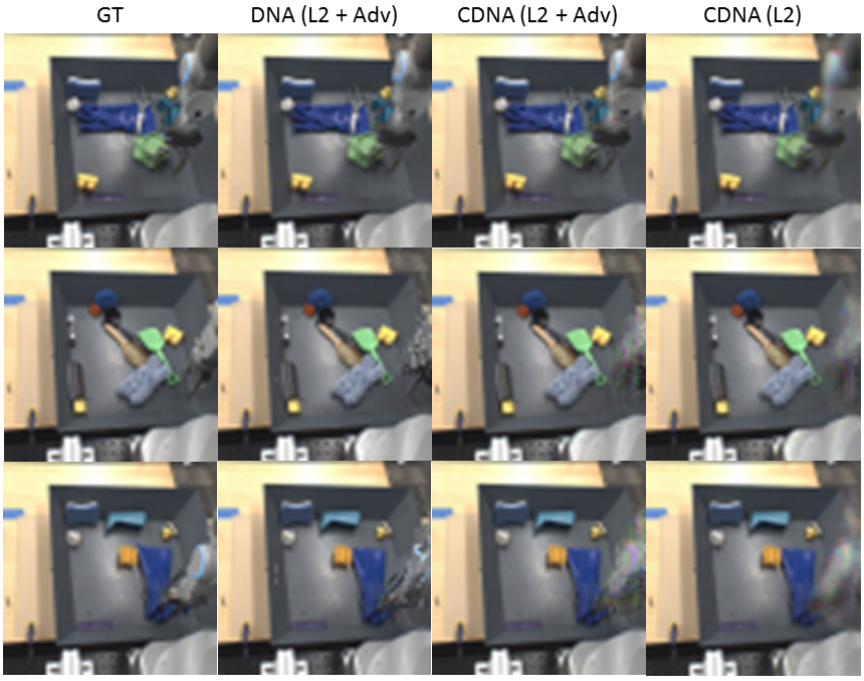


Figure 8: A few not-randomly selected example predictions for qualitative comparison of the multi-frame models on test input. The 4th (final) prediction timestep is shown here, with ground truth (GT) on the left. The DNA architecture is an action-conditioned implementation of the architecture proposed in [8]. The CDNA (L2) model is purely trained on mean-squared error. Notice that the models trained with hybrid loss do not exhibit the same blurring effect as the L2 model does. However, they introduce some of their own artifacts as well.

5 Conclusion

In this project, we experiment with multiple action-conditional video prediction models, adapting and combining ideas from prior work in this area. In particular, we focus on how architectures that explicitly model motion, which can be helpful when trying to learn physical interactions from video data, can benefit from the GAN approach. We test these models on a physical interaction video dataset and make qualitative and quantitative comparisons. We designed the single frame architecture in order to capture an understanding of the dynamics and the multi-frame model to capture temporal differences. We observed that our model generates sharper predictions compared to a l_2 based method.

Due to time and computational constraints, the multi-frame architectures in particular were trained for relatively few iterations and could benefit from longer training times. The auxiliary task of robot arm state prediction could be useful for the multi-frame architecture, as it was in the single-frame architecture. In fact, we believe that other video prediction tasks can also benefit from such techniques. Additionally, the single-frame architecture may benefit from object level motion learning through CDNA rather than just pixel level motion learning from DNA. These provide interesting directions of future investigation in learning physical interaction through video prediction by means of generative models.

References

[1] Chelsea Finn, Ian Goodfellow, and Sergey Levine. Unsupervised learning for physical interaction through video prediction. In *Advances In Neural Information Processing Systems*, pages 64–72, 2016.

[2] Junhyuk Oh, Xiaoxiao Guo, Honglak Lee, Richard L Lewis, and Satinder Singh. Action-conditional video prediction using deep networks in atari games. In *Advances in Neural Information Processing Systems*, pages 2863–2871, 2015.

[3] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[4] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[5] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[6] Michael Mathieu, Camille Couprie, and Yann LeCun. Deep multi-scale video prediction beyond mean square error. *arXiv preprint arXiv:1511.05440*, 2015.

[7] Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. Generating videos with scene dynamics. In *Advances In Neural Information Processing Systems*, pages 613–621, 2016.

[8] Carl Vondrick and Antonio Torralba. Generating the future with adversarial transformers. In *CVPR 2017*, 2017.

[9] Ian Goodfellow. Nips 2016 tutorial: Generative adversarial networks. *arXiv preprint arXiv:1701.00160*, 2016.

[10] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.

[11] Nal Kalchbrenner, Aaron van den Oord, Karen Simonyan, Ivo Danihelka, Oriol Vinyals, Alex Graves, and Koray Kavukcuoglu. Video pixel networks. *arXiv preprint arXiv:1610.00527*, 2016.

[12] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. *CoRR*, abs/1310.1531, 2013.

[13] David Eigen and Rob Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2650–2658, 2015.

[14] Junho Yim, Heechul Jung, ByungIn Yoo, Changkyu Choi, Dusik Park, and Junmo Kim. Rotating your face using multi-task deep neural network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 676–684, 2015.

[15] Tianzhu Zhang, Bernard Ghanem, Si Liu, and Narendra Ahuja. Robust visual tracking via structured multi-task sparse learning. *International journal of computer vision*, 101(2):367–383, 2013.

[16] Bert De Brabandere, Xu Jia, Tinne Tuytelaars, and Luc V Gool. Dynamic filter networks. In *Advances In Neural Information Processing Systems*, pages 667–675, 2016.

[17] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[18] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

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