

# Predicting the Future with Transformational States

Andrew Jaegle<sup>1</sup>, Oleh Rybkin<sup>1</sup>, Konstantinos G. Derpanis<sup>2</sup>,  
and Kostas Daniilidis<sup>1</sup>

<sup>1</sup>University of Pennsylvania, <sup>2</sup>Ryerson University  
ajaegle@upenn.edu, oleh@cis.upenn.edu,  
kosta@scs.ryerson.ca, kostas@cis.upenn.edu

**Abstract.** An intelligent observer looks at the world and sees not only what is, but what is moving and what can be moved. In other words, the observer sees how the present state of the world can transform in the future. We propose a model that predicts future images by learning to represent the present state and its transformation given only a sequence of images. To do so, we introduce an architecture with a latent state composed of two components designed to capture (i) the present image state and (ii) the transformation between present and future states, respectively. We couple this latent state with a recurrent neural network (RNN) core that predicts future frames by transforming past states into future states by applying the accumulated state transformation with a learned operator. We describe how this model can be integrated into an encoder-decoder convolutional neural network (CNN) architecture that uses weighted residual connections to integrate representations of the past with representations of the future. Qualitatively, our approach generates image sequences that are stable and capture realistic motion over multiple predicted frames, without requiring adversarial training. Quantitatively, our method achieves prediction results comparable to state-of-the-art results on standard image prediction benchmarks (Moving MNIST, KTH, and UCF101).

**Keywords:** Image prediction, motion, sequence modeling

## 1 Introduction

Humans and other animals are able to reason about the future state of the world given visual observations of the present. Even as infants, humans can use images to make informed predictions of how objects and agents will move and act in the future [1]. A large body of evidence from the neural and cognitive sciences suggests that humans build predictive models of the world and use the resulting predictions to guide action and to learn better ways of engaging with the world

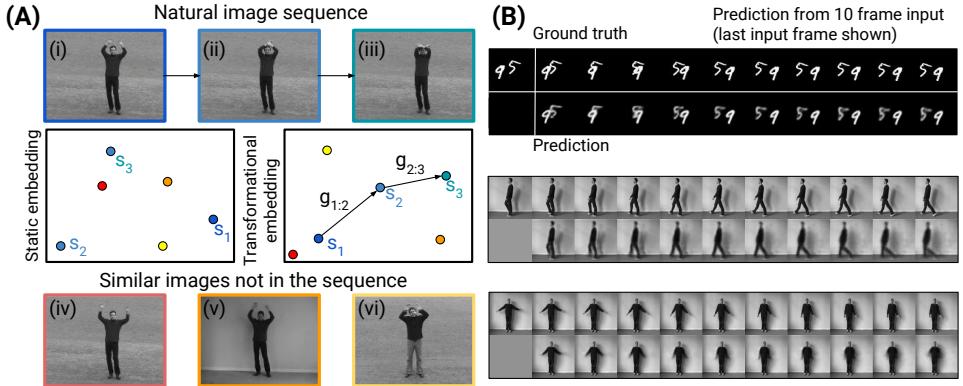


Fig. 1: (A) This work is motivated by the observation that image transformations may be more easily modeled by a network that learns to transform latent states rather than transform or associate pixel intensities. By learning to model states,  $s$ , along with transformations between states,  $g$ , the RNN is encouraged to model sequences not by memorizing arbitrary transitions between certain images (static embedding) but by reshaping the embedding so that natural state transformations are predictable (transformational embedding). Figure best viewed in color. (B) Sample predictions of our model on sequences from the Moving MNIST and KTH datasets. We show only the last of 10 input images for visualization purposes. Our model produces good image predictions using only pixel-wise reconstruction losses.

[2]. The world is filled with image sequences, and it is clear that intelligent agents might use the rich dynamics of visual stimuli to guide learning. But how agents should learn to predict effectively remains an important computational problem.

The focus of this paper is the prediction of future images given a sequence of past images. Image prediction offers a general approach to tackling the challenge of state prediction in vision because it is not tied to a specific task or representation. By predicting images instead of task-dependent representations such as labels or segmentations, the agent gains more flexibility in how it uses information about the future. From this perspective, image prediction offers a unique opportunity for unsupervised visual representation learning, as image-level predictions can be used as a learning signal even in the absence of well-defined tasks or task-conditional reward signals.

Motion prediction is at the heart of future prediction in temporally contiguous video. Over short periods of time, scenes in the real world contain a slowly changing context and set of objects. Future frames can largely be predicted by modeling the motion of objects and the scene: that is, by transforming the current state into future states. Our work is motivated by the observation that learning states that can be transformed to produce future states may lead to

representations that are easier to predict, as illustrated in Figure 1(A). Other methods that use motion for prediction typically rely on the assumption that image transformations can be modeled with local, piecewise translational motion. However, such methods struggle with scenes containing flexible objects like human bodies (e.g. the one shown in Figure 1(A)), due to the self-occlusion and non-rigid deformations that such objects introduce. Here, we propose a model that makes predictions by transforming latent representations, and which can reason about transformations that are more complex than simple pixel translations.

To predict realistic images, a model of sequence transformations must also capture the appearance and texture of the scene as it transforms. This includes the content of image regions that appear or become dis-occluded over time. A model must capture the appearance of the foreground and background to paint in details of image regions that are revealed as the objects and scene move. A useful future image prediction model needs to model both this static state and its transformations to ensure that individual frames are realistic and that objects and the scene move realistically. Here, we show how to integrate weighted residual connections into our network to produce good models of background texture and other static scene content.

We propose to predict a latent state representation that encodes both the current state of the scene and its transformation and that can be decoded to produce future images. Our method learns representations of states and transformations that are stable and sufficiently rich to produce multiple future frames without re-encoding estimated frames or repeatedly copying pixels from the input sequence. Our architecture learns to capture naturalistic motion in a variety of settings (synthetic and real) with minimal modifications. Our model achieves quantitative results competitive with the state of the art without assuming a static background (or stabilized preprocessing), without being constrained to directly copying or translating pixels from input frames, and without adversarial learning.

Our technical contributions are as follows:

- a novel RNN core formulation with a partitioned representation of latent states and transformations;
- a weighted, temporal residual connection that enables stably reconciling features across multiple time steps without re-encoding images;
- an encoder-decoder architecture that can be stably trained for good end-to-end image prediction without an adversarial loss.

## 2 Related work

There is a growing interesting in predicting future imagery conditioned on past observations. This body of work leverages large, unlabeled video datasets to learn to make predictions. Prior work has explored a variety of aspects of the problem. Here, we present an overview of prior work organized by the level of abstraction

of the target output, the generative process, the structure of the latent representation, and the loss function guiding learning. To further aid interpretation of the present work, we compare the specific design choices of a range of recently proposed models in Supplementary Table 1.

**Prediction targets** At the prediction output level, a variety of representational levels have been targeted. At the highest level, several works have considered predicting semantic segmentation of frames [3], deep visual image representations of frames [4], human pose [5,6,7] and human actions [8,9]. Others have considered mid-level representation outputs, such as optical flow [10,11]. At the lowest and most general level, a growing body of research has explored predicting the pixel intensities of future frames [12,13,14,15,16,17,18,19,20,21,22,23]. In this paper, we propose a method to predict frame-wise pixel intensities by recurrently transforming image representations into the future.

**Prediction and transformation** A key differentiating aspect between prior work is the generative process. Inspired by encoder-decoder language models (e.g. [24]), [12,13] consider a recurrent network that encodes an input sequence into a fixed length vector and a subsequent recurrent network that decodes the vector to progressively generate predicted frames. Others have considered a more direct approach that predicts future frames from observed frames using a convolutional neural network (CNN) [14,3]. Several other works have proposed copying or applying simple transformations to past pixels to generate image frame predictions [25,26,18,17,27]. In contrast, we predict future images by transforming and decoding the latent space, rather than by directly predicting future frames or by copying or transforming past pixels.

**Factored representations** Another line of recent work has approached the problem of prediction by factoring the representation of the latent information or shaping the latent to learn properties useful for prediction. Vondrick et al. [15] factor the generative process into separate foreground and background streams that are combined to create the final video. Goroshin et al. [28] train a linearized latent space so that future prediction can be treated as linear interpolation. Several works [21,29] have considered predicting human pose and then conditioning image generation on the predicted pose. These models can achieve impressive results, but they assume latents with known structure (i.e. 2D poses) and are thus limited to human-focused imagery. Our method assumes only that learned representations can be transformed by a learned operator and is not restricted to settings where the latent space can be explicitly labeled.

**Predicting with motion and content** Most similar to our work are two approaches that factor the latent information to capture scene appearance and dynamics [30,16]. Denton et al. [30] learn separate representations of content and pose and predict future frames by fixing the content and estimating the future pose. This model produces very stable predictions, but it assumes content does not change over a sequence. This limits its applicability to scenes with camera motion and dynamic content. Our method does not assume fixed scenes, but instead learns a representation of state and motion that is designed to accommodate a variety of transformations while still preserving image structure.

Villegas et al. [16] learn representations of content and motion by feeding two networks with images and difference images, respectively. They train their method with an adversarial loss and need to re-encode predictions to generate more than one future frame. While their method incorporates motion into the representation by splitting the input into images and difference images and directly predicting next frames, our method learns to represent both states and transformations from frames and learns motion by applying transformations to states. Our method produces good results without adversarial training or image re-encoding.

**Loss functions** Previous work has proposed to improve future prediction by designing loss functions to guide learning to better solutions. While earlier work uses standard pixel-wise reconstruction losses like the mean-squared error (MSE) or binary cross entropy (BCE) loss function [13], more recent work (e.g., [14,17,31,32]) often incorporates some form of generative adversarial network (GAN) model [33], either alone or in conjunction with a reconstruction loss, such as MSE. While GANs can produce crisp predictions, they are notoriously hard to train and model convergence is difficult to evaluate [34]. In this paper, we demonstrate that a simple MSE loss is capable of generating good predictions when paired with an appropriately structured architecture.

### 3 Technical approach

In future prediction, we are given a sequence of  $T$  images  $\{I_1, \dots, I_T\}$  and want to produce the most likely sequence of  $K$  future images  $\{I_{T+1}, \dots, I_{T+K}\}$ . We seek to do so by capturing how the structure of the image transforms over time.

Images are high dimensional but the pixel space dimension does not reflect the intrinsic dimensionality of the scene. For example, a  $64 \times 64$  image of two translating digits lives in a pixel space of the same dimensionality as a  $64 \times 64$  image of a walking person. However, the latter image contains scene content with many more degrees of freedom so its intrinsic dimensionality is higher. Similarly, a  $128 \times 128$  and a  $64 \times 64$  image of the same walking person both depict content with the same degrees of freedom (up to appearance details lost in downsampling), but with very different pixel dimensions. When we predict images, we must predict pixels, but we seek to do so by modeling the transformation of the images' content.

Accordingly, we model the instantaneous state of the scene at time  $t$  using a latent variable  $s_t$ . Because we do not know the state of future frames, we seek to transform past latent variables  $\{s_1, \dots, s_T\}$  to estimate the future latents  $\{s_{T+1}, \dots, s_{T+K}\}$ . Future latents depend on previously estimated future latents, so we model this estimation process with a function  $f$ , such that the estimate at time  $k$ , where  $1 \leq k \leq K$ , is given by

$$\hat{s}_t = f(\{s_1, \dots, s_T, \hat{s}_{T+1}, \dots, \hat{s}_{T+k-1}\}). \quad (1)$$

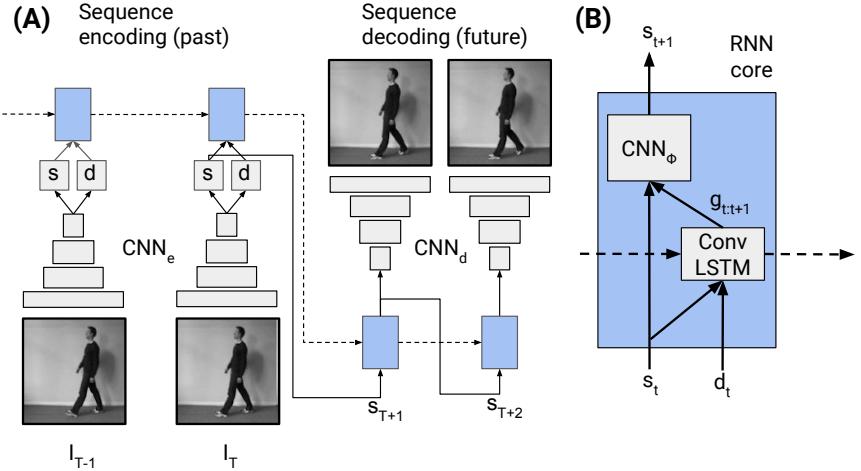


Fig. 2: Architecture overview. (A) Our model uses an encoder-decoder sequence-to-sequence architecture with a factorized latent that captures the image state,  $s$ , and transformation,  $d$ . Residual connections are omitted for clarity; see the text and Figure 3 for details. (B) Future states are transformed from past states using an RNN core that accumulates the transformation estimate  $g$  with a ConvLSTM and applies it to the recursively estimated state  $s$  with an operator  $\text{CNN}_\phi$ .

In the context of image prediction, such a function is typically parameterized with a recurrent neural network (RNN) applied to the output of the encoder of an encoder-decoder architecture [13]:

$$\hat{I}_{T+k} = \text{CNN}_d(\hat{s}_{T+k}) \quad (2)$$

$$\hat{s}_{T+k} = \text{RNN}(\{s_1, \dots, s_T, \hat{s}_{T+1}, \dots, \hat{s}_{T+k-1}\}) \quad (3)$$

$$s_t = \text{CNN}_e(I_t), \quad (4)$$

where  $\hat{s}_t$  is the estimated latent state at time  $t$ ,  $\hat{I}_t$  is the estimated image at time  $t$ , and  $\text{CNN}_e$  and  $\text{CNN}_d$  are encoder and decoder CNNs, respectively. This is illustrated in Figure 2(A).

While such structures can in principle learn to model arbitrary transformations [35], these models often struggle to produce realistic image transformations. In practice, these models may learn to memorize transformations as arbitrary mappings from state to state (as illustrated schematically in Figure 1(A)) rather than representing transformations as predictable, generalizable mappings like those that characterize the natural transformations between world states.

We now describe how we encourage the network’s latent state to learn more predictable mappings by jointly learning representations of state and transformations.

### 3.1 Transformational states

To encourage an encoder-decoder structure to learn to model predictable latent space transformations, we introduce an additional latent variable  $d_t$  to capture the evidence available for estimating the transformation from each input image. The output of the encoder CNN at each time step can then be written as

$$s_t, d_t = \text{CNN}_e(I_t). \quad (5)$$

Next, we describe how we encourage the network to exploit the factorization in (5) by wiring the network so that transformational states  $d_t$  cannot directly predict images but must act by transforming states  $s_t$ .

We want the  $d_t$  to capture all information that is available from  $I_t$  about the transformation from state  $s_t$  to  $s_{t+1}$ . Let us call this transformation  $g_{t:t+1}$ . While individual images provide some information about how the world will move, in general they are insufficient to model the full transformation from  $t$  to  $t+1$  and to later points in the future.

To see this, consider the person in image (i) of Figure 1(A). From this world state, transformations in time are unlikely to produce image (v), which shows the same person in a different scene, or image (vi), which shows a different person in a similar position. However, the person in image (i) might move his arms closer together (producing image (ii)) or further apart (producing image (iv)). Given image (i) and (ii), however, image (iii) becomes much more likely than image (iv).

That is, the transformation that can be estimated from a single image ( $d_t$ ) will not in general be equivalent to the true state transformation ( $g_{t:t+1}$ ). However, some information about the transformational state of the world is observable from a single image, and we can arrive at better estimates by integrating transformation estimates over time. Accordingly, we use an RNN to incorporate the history of instantaneous transformation estimates  $d_t$  and the accompanying states  $s_t$  to obtain a better estimate of the transformation  $g_{t:t+1}$  acting on  $s_t$ :

$$g_{t:t+1} = \text{RNN}(\{[d_1, s_1], \dots, [d_t, s_t]\}). \quad (6)$$

We then model the action of this transformation on latent states as

$$s_{t+1} = \Phi(g_{t:t+1}, s_t), \quad (7)$$

where  $\Phi$  is an operator that transforms  $s_t$  by applying  $g_{t:t+1}$ . We parameterize  $\Phi$  with a three-layer CNN (with no recurrence), and we parameterize the RNN with a three-layer convolutional long short-term memory (convLSTM) model. We show the full recurrent core, including the CNN operator and transformation RNN in Figure 2(B).

We next describe how we integrate skip connections into the model to encourage long-term stability and fidelity of image production while the state is undergoing transformations.

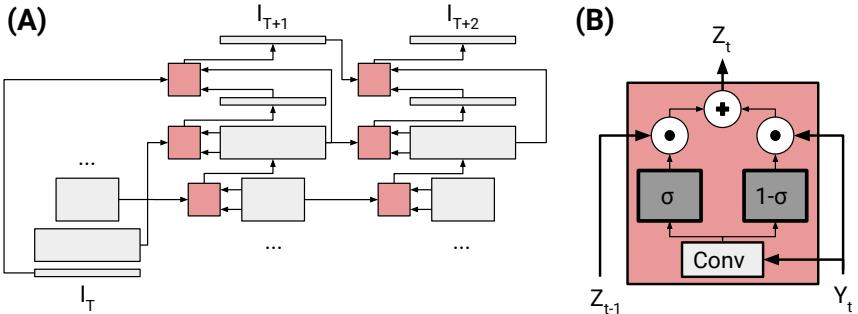


Fig. 3: Weighted residual connections. (A) To produce high quality images at multiple time steps in the future without re-encoding images, we use a residual connection scheme designed to gradually alter image content from the last observed input image. Residual connections connect the encoder at time  $T$  (last input) to the decoder at time  $T + 1$  (first output). At subsequent times, the decoder inherits information about the past from the decoder at the previous time only. The network has this connectivity pattern at every layer: we show only two decoder layers and the output image for easier visualization. (B) We use a retinotopic weighting scheme to allow each layer of a decoding network to selectively incorporate skipped input from the past. Weights and feature maps at time  $t$  are functions of the predicted latent state  $\hat{s}_t$  at time  $t$ .

### 3.2 Weighted residual connections

Recent works [30,16,26] have found that skip connections from encoder to decoder networks are essential for producing high quality image outputs, especially for capturing high-frequency information of textures and background. However, when encoded images are in the past and decoded images are in the future, this paradigm is limited in several ways. First, future encoder states cannot be used as a source of skipped image information without first re-encoding estimated images. This may lead to difficulties in CNN/RNN training because of mismatched statistics between true and estimated frames. Second, skipping from past states to future ones can introduce artifacts when static features are copied as if nothing in the scene has changed. This can result in ghosting artifacts that are difficult for the network to learn to correct.

We introduce a mechanism for passing information forward from the encoder state at the last input time step to the decoder at future time steps without re-encoding predictions and without repeatedly copying activations from the past (Figure 3). Instead of copying activations from the encoder to the decoder at all future times, we connect the encoder at the last input time step only to the decoder at the first prediction time step. Subsequent decoder time steps take the activations of the decoder at the previous time step and re-weight them. This configuration allows features to flow forward in time from the last input

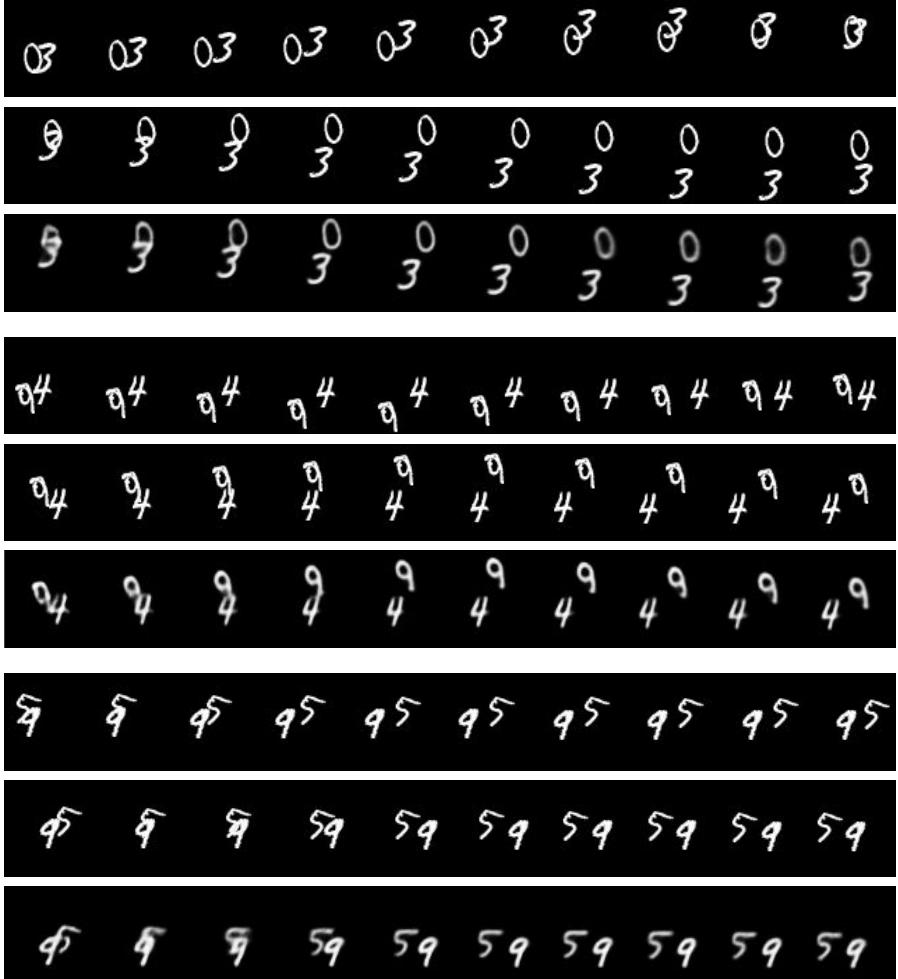


Fig. 4: Example sequences on Moving MNIST. For all three examples, the first row shows the input sequence (past), the second row shows the ground truth future, and the third row shows the predicted sequence. Our model is able to stably predict digits over multiple timesteps, even when digits overlap for multiple frames.

time step, while allowing features to change as necessary to reflect motion and without requiring images to be re-encoded.

The initial feature map  $Y_t^l$  output by layer  $l$  of the decoder network at time  $t$  is combined with skipped output  $Z_{t-1}^l$  from the previous time step in the form of a weighted residual connection:

$$Z_t^l = (1 - \sigma(W^l)) \odot Y_t^l + \sigma(W) \odot Z_{t-1}^l, \quad (8)$$

Model	average, 10 predicted frames	first frame prediction
ConvLSTM [37]	367.2	-
Encoder-Decoder LSTM [13]	341.2	-
Dynamic Filter Networks [25]	285.2	-
Spatiotemporal Autoencoder [27]	-	179.8
Video Pixel Networks [19]	87.6	-
Video Ladder Networks [23]	187.7	-
Ours	210.1	172.4

Table 1: Comparison of binary cross-entropy (BCE) results (nats/frame) on the Moving MNIST test set. Lower scores indicate better performance.

where  $\odot$  denotes element-wise multiplication.  $Z_t^l$  is the final output of the network at layer  $l$  at time  $t$ .  $W_t^l$  (a weight map) is the output of a  $1 \times 1$  convolution with  $Y_t^l$  as input. We output one weight value for each spatial position of the feature map and broadcast the weight to all channels to perform the elementwise multiplication. This weighting strategy introduces only  $K^l + 1$  parameters per layer, where  $K^l$  is the number of channels in  $Y^l$ .

For the first prediction time step, the skip inputs  $Z_{t-1}^l$  come from the layer of the encoder network at the last input time step with matching spatial dimension. Otherwise, they come from the corresponding layer in the decoder at time  $t - 1$ . The weighting scheme is shown in Figure 3(A). This configuration is similar to the one introduced in [36], applied at each time step.

When paired with our network architecture, this skip configuration allows us to estimate future images without re-encoding estimated images into the encoder CNN. Because subsequent time steps inherit the activations of the previous decoder state, and do not directly copy the states of the last encoder (as in e.g. [38]), we observed that these networks trained more quickly and resulted in fewer ghosting artifacts.

We incorporate a similar weighted residual scheme to directly skip the last input image to future timesteps. As with feature maps, for all times  $t > T + 1$  we skip the image from the previous timestep  $t - 1$  instead of the last input image  $I_T$ . Directly skipping the final input image to later time steps resulted in lower quality outputs (see Supplemental Figures 2 and 3 for examples). We also observed that the residual connection works best when the weighting is applied after the tanh nonlinearity in both images. Weighting before the output nonlinearity led to saturated images at later prediction time steps.

## 4 Experiments

### 4.1 Datasets

We performed experiments on three datasets: a standard synthetic dataset, Moving MNIST [13], and two real world datasets, KTH actions [39] and UCF101 [40]. Moving MNIST is a dataset of synthetic videos, with an arbitrarily large training

Model	PSNR		SSIM	
	at time 1	average over 10 frames	at time 1	average over 10 frames
ConvLSTM [16]	33.8	27.6	0.95	0.84
MCNet [16]	33.8	28.2	0.95	0.86
Ours	34.8	29	0.95	0.86

Table 2: Comparison of frame prediction results on the KTH test set. Higher scores indicate better performance.

Model	PSNR	SSIM
EpicFlow [41]	29.1	0.91
NextFlow [42]	29.9	-
BeyondMSE [14]	28.2	0.89
DVF [18]	29.6	0.92
Dual Motion GAN [20]	30.5	0.94
Ours	28.3	0.88

Table 3: Comparison of next frame prediction results on the UCF101 test set (split 1). Higher scores indicate better performance.

set (videos are generated procedurally) and a test set of 10,000 videos. Each video has an image resolution of  $64 \times 64$  and is 20 frames in length. The videos capture two digits moving in random directions and with random velocities. KTH consists of 2391 videos capturing six human actions: boxing, hand clapping, hand waving, jogging, running, and walking. As is standard practice in prior work on frame prediction using KTH, we convert the images to  $128 \times 128$  prior to processing. All sequences contain scenes with relatively homogeneous backgrounds. The scenes were captured with a static camera at 25 frames per second. UCF101 contains 13,320 YouTube videos capturing 101 human actions. As done in previous evaluations using UCF101, we convert the images to  $256 \times 256$  prior to processing. Notably, many UCF101 videos contain spatial and temporal (i.e. duplicate frames) artifacts due to compression.

## 4.2 Architecture and training details

The Moving MNIST and KTH networks were trained to predict 10 frames given 10 input frames and UCF101 networks were trained to predict 1 frame given 2 input frames (to allow us to compare to the compendium of state-of-the-art results in [20]). On all datasets probed, we trained the network end-to-end using an average pixel-wise reconstruction loss between the estimated sequences and ground truth future sequences. We use an MSE loss for KTH and UCF101 and a BCE loss for Moving MNIST. All networks were trained using stochastic gradient descent (SGD) with momentum. We used a starting learning rate of 1 on KTH and UCF101 and 10 on Moving MNIST. We decayed learning rates by a

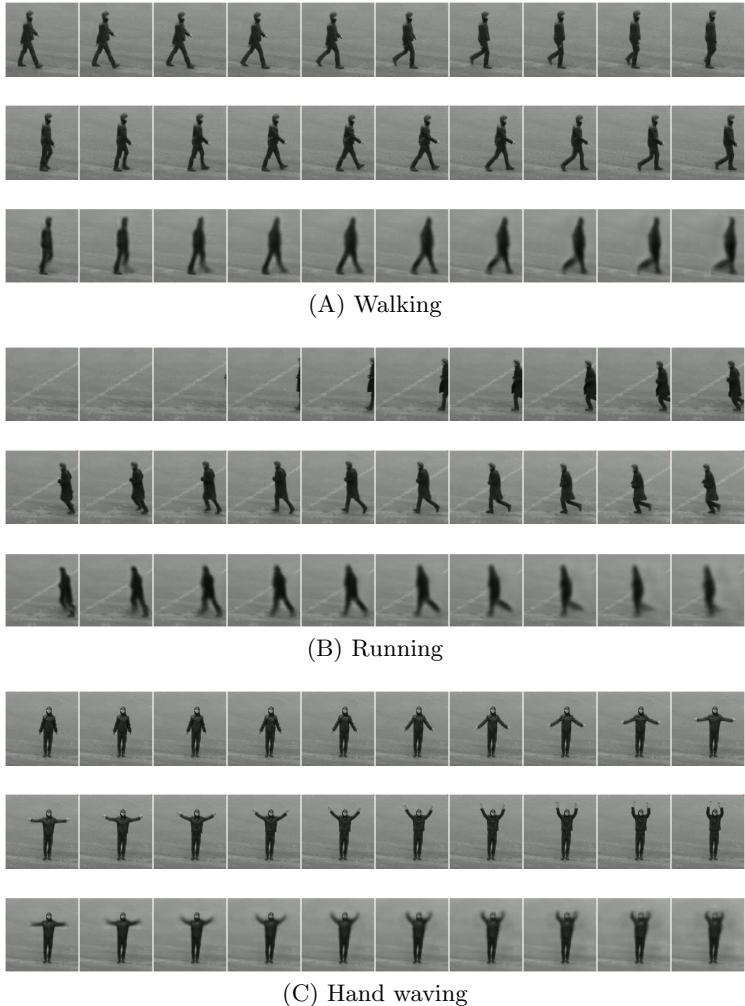


Fig. 5: Example sequences on KTH. For all three examples, the first row shows the input sequence (past), the second row shows the ground truth future, and the third row shows the predicted sequence. The model produces faithful motion in a variety of settings and is able to paint in the background after dis-occlusion.

factor of 10 every time the validation loss reached a plateau, until convergence. We used a momentum value of  $\beta = 0.5$  in all cases. We used a weight decay of  $1 \times 10^{-4}$  for encoder and decoder weights on all networks, and we included dropout with a rate of 0.5 in all hidden layers of encoder networks on UCF101 and KTH.

We used horizontal mirroring and random cropping for data augmentation on both UCF101 and KTH datasets. We trained on Moving MNIST with 50

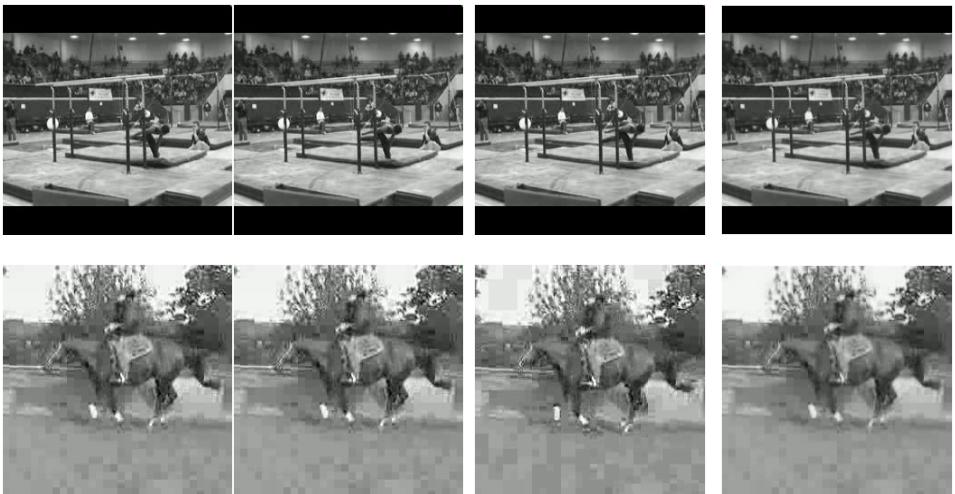


Fig. 6: Example sequences on UCF101. For each example, we show two frames from the past followed by the ground truth third frame and the third frame predicted by the model from the first two images.

sequences per batch, on KTH with 20 sequences per batch, and on UCF101 with 10 sequences per batch.

The convolutional architectures used on all three datasets are based on the DCGAN architecture [43]. Each layer of the decoder except for the input layer contains the same number of channels as the corresponding layer of the encoder architecture. Because the decoder does not take the transformational latent as input, the decoder input is of size  $4 \times 4 \times N_s$ , while the encoder output is of size  $4 \times 4 \times (N_s + N_d)$ , where  $N_s$  is the number of channels in the state latent  $s$  and  $N_d$  is the number of channels in the transformational latent  $d$ . In all architectures used here,  $N_s = N_d$ . We did not perform hyperparameter search for the values of  $N_s$  and  $N_d$  or the architectures used for encoders and decoder CNNs, and it is likely that better results can be obtained using optimized settings.

The architectures we use on Moving MNIST, KTH, and UCF101 differ only in the number of layers and the number of filters per layer in the encoder and decoder CNNs. Architecture depths were chosen so that the spatial size the encoder output (and decoder input) was  $4 \times 4$ . We specify full architectures in the supplemental material (supplemental section 2). We will make the model code and trained models available upon paper acceptance.

### 4.3 Evaluation

It is difficult to quantitatively evaluate prediction results because reconstruction errors and other measures do not generally fully capture the perceptual quality of reconstructed images [44,45,14]. Nonetheless, quantitative evaluations can give

a reasonable indication of the average quality of a method when seen alongside the qualitative results the method produces.

We evaluate our methods using the error measures most commonly used in the literature: binary cross entropy for Moving MNIST [13] and peak signal to noise ratio (PSNR) and Structural Similarity (SSIM) [45] for KTH and UCF101. We evaluate SSIM using a window of 7x7 pixels with uniform weighting (the same parameters as [30]).

We show quantitative results for Moving MNIST in Table 1, KTH in Table 2, and UCF in Table 3. In all cases, our results are competitive with state of the art. Because of the large number of architectures and loss configurations in the literature, it is infeasible to thoroughly test all architecture and loss configurations. We report results based on the numbers used in the literature. To aid interpretation of our results in the context of the sequence prediction literature, we include a table comparing the different architectural and loss configurations in the supplement (Supplementary Table 1).

We show sample qualitative results on the three datasets in Figure 4 (Moving MNIST), Figure 5 (KTH), and Figure 6 (UCF). Our method produces reasonable results with good motion in many of settings in these three datasets. The output of dynamic models are difficult to evaluate based on static images alone, and consequently the results of our method are best understood by examining the videos on the project website ([https://daniilidis-group.github.io/transformational\\_states](https://daniilidis-group.github.io/transformational_states)). To aid interpretation of our results, we also show failure cases on KTH in Figure 7. Additionally, we show prediction results produced by models with network ablations in the supplement: ablations on Moving MNIST are shown in Supplementary Figure 1 and ablations on KTH are shown in Supplementary Figures 2 and 3. Ablation results are shown on random sequences from the test data in all cases for fair comparison.

## 5 Summary

We have described a model for predicting sequences of future images using an architecture that learns latent states and their transformations to future states. We show how to couple this architecture with weighted residual connections from past to future time steps to produce images that are stable after recursive transformations. The resulting network can be trained to predict reasonable results on synthetic and real datasets without requiring direct pixel copying or a GAN. Our model produces good qualitative results and achieves quantitative results comparable to state-of-the-art on several image prediction datasets.

**Acknowledgments** We thank Kenneth Chaney and Nikos Kolotouros for computing support, Stephen Phillips for helpful comments, and the members of the Daniilidis group and the vision community at Penn for many fruitful discussions. We are grateful for support through the following grants: NSF-DGE-0966142 (IGERT), ARL RCTA W911NF-10-2-0016, and ONR N00014-17-1-2093. K.G.D. is supported by a Canadian NSERC Discovery grant.

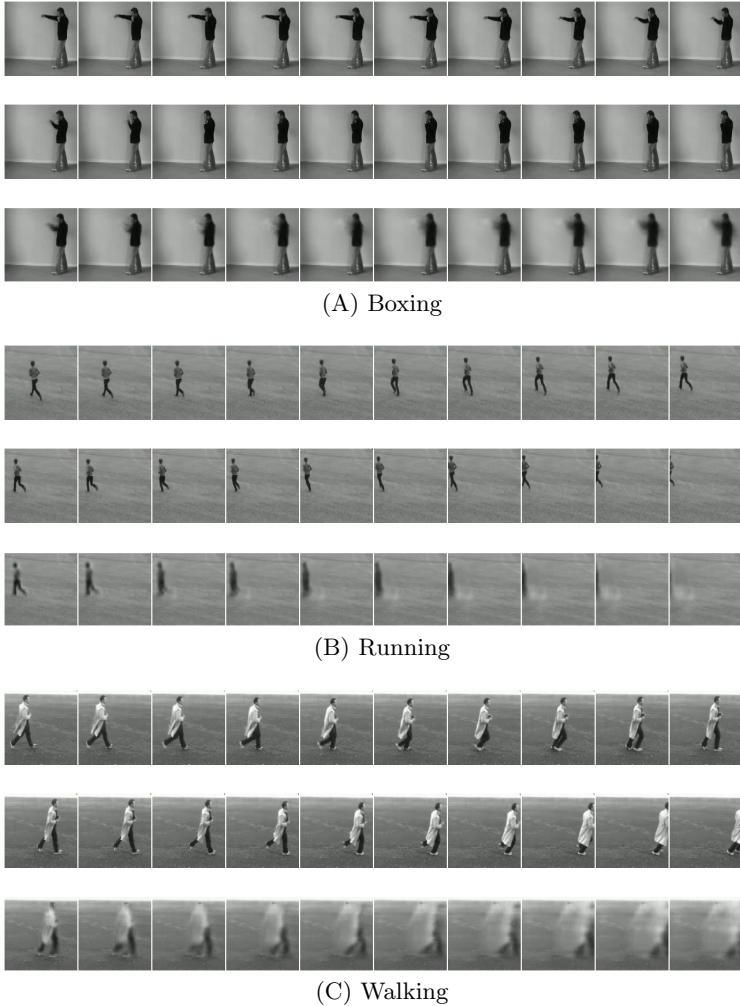


Fig. 7: Example failure cases on KTH. (A) The model outputs a blurry motion sequence that does not correspond to the ground truth. (B) The model fails to correctly predict motion or paint in the background when the moving object occupies only a small part of the image. (C) The model fails to correctly paint in the background after the foreground moves, leading to ghosting artifacts.

## References

1. Spelke, E., Phillips, A., Woodward, A.: Infants knowledge of object motion and human action. *Causal Cognition: A Multidisciplinary Debate* (1996)
2. Bubic, A., Von Cramon, D.Y., Schubotz, R.: Prediction, cognition and the brain. *Frontiers in Human Neuroscience* **4** (2010)
3. Luc, P., Neverova, N., Couprie, C., Verbeek, J., LeCun, Y.: Predicting deeper into the future of semantic segmentation. In: *ICCV*. (2017) 648–657
4. Vondrick, C., Pirsiavash, H., Torralba, A.: Anticipating visual representations from unlabeled video. In: *CVPR*. (2016) 98–106
5. Fragkiadaki, K., Levine, S., Felsen, P., Malik, J.: Recurrent network models for human dynamics. In: *ICCV*. (2015) 4346–4354
6. Bütepage, J., Black, M.J., Krägic, D., Kjellström, H.: Deep representation learning for human motion prediction and classification. In: *CVPR*. (2017) 1591–1599
7. Martinez, J., Black, M.J., Romero, J.: On human motion prediction using recurrent neural networks. In: *CVPR*. (2017) 4674–4683
8. Nguyen, M.H., la Torre, F.D.: Max-margin early event detectors. *IJCV* **107**(2) (2014) 191–202
9. Kitani, K.M., Ziebart, B.D., Bagnell, J.A., Hebert, M.: Activity forecasting. In: *ECCV*. (2012) 201–214
10. Yuen, J., Torralba, A.: A data-driven approach for event prediction. In: *ECCV*. (2010) 707–720
11. Walker, J., Gupta, A., Hebert, M.: Dense optical flow prediction from a static image. In: *ICCV*. (2015) 2443–2451
12. Ranzato, M., Szlam, A., Bruna, J., Mathieu, M., Collobert, R., Chopra, S.: Video (language) modeling: a baseline for generative models of natural videos. *arXiv e-Prints* (2014)
13. Srivastava, N., Mansimov, E., Salakhutdinov, R.: Unsupervised learning of video representations using LSTMs. In: *ICML*. (2015) 843–852
14. Mathieu, M., Couprie, C., LeCun, Y.: Deep multi-scale video prediction beyond mean square error. In: *ICLR*. (2016)
15. Vondrick, C., Pirsiavash, H., Torralba, A.: Generating videos with scene dynamics. In: *NIPS*. (2016) 613–621
16. Villegas, R., Yang, J., Hong, S., Lin, X., Lee, H.: Decomposing motion and content for natural video sequence prediction. In: *ICLR*. (2017)
17. Vondrick, C., Torralba, A.: Generating the future with adversarial transformers. In: *CVPR*. (2017) 2992–3000
18. Liu, Z., Yeh, R.A., Tang, X., Liu, Y., Agarwala, A.: Video frame synthesis using deep voxel flow. In: *ICCV*. (2017) 4473–4481
19. Kalchbrenner, N., van den Oord, A., Simonyan, K., Danihelka, I., Vinyals, O., Graves, A., Kavukcuoglu, K.: Video pixel networks. In: *ICML*. (2017) 1771–1779
20. Liang, X., Lee, L., Dai, W., Xing, E.P.: Dual motion GAN for future-flow embedded video prediction. In: *ICCV*. (2017) 1762–1770
21. Walker, J., Marino, K., Gupta, A., Hebert, M.: The pose knows: Video forecasting by generating pose futures. In: *ICCV*. (2017) 3352–3361
22. Lotter, W., Kreiman, G., Cox, D.D.: Deep predictive coding networks for video prediction and unsupervised learning. In: *ICLR*. (2017)
23. Cricri, F., Ni, X., Honkala, M., Aksu, E., Gabbouj, M.: Video Ladder Networks. *ArXiv e-prints* (December 2016)

24. Sutskever, I., Vinyals, O., Le, Q.V.: Sequence to sequence learning with neural networks. In: NIPS. (2014) 3104–3112
25. Brabandere, B.D., Jia, X., Tuytelaars, T., Gool, L.V.: Dynamic filter networks. In: NIPS. (2016) 667–675
26. Finn, C., Goodfellow, I.J., Levine, S.: Unsupervised learning for physical interaction through video prediction. In: NIPS. (2016) 64–72
27. Pătrăucean, V., Handa, A., Cipolla, R.: Spatio-temporal video autoencoder with differentiable memory. In: ICLR Workshop. (2016)
28. Goroshin, R., Mathieu, M., LeCun, Y.: Learning to linearize under uncertainty. In: NIPS. (2015) 1234–1242
29. Villegas, R., Yang, J., Zou, Y., Sohn, S., Lin, X., Lee, H.: Learning to generate long-term future via hierarchical prediction. In: ICML. (2017) 3560–3569
30. Denton, E.L., Birodkar, V.: Unsupervised learning of disentangled representations from video. In: NIPS. (2017) 4417–4426
31. Lu, C., Hirsch, M., Schölkopf, B.: Flexible spatio-temporal networks for video prediction. In: CVPR. (2017) 2137–2145
32. Zeng, K., Shen, W.B., Huang, D., Sun, M., Niebles, J.C.: Visual forecasting by imitating dynamics in natural sequences. In: ICCV. (2017) 3018–3027
33. Goodfellow, I.J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A.C., Bengio, Y.: Generative adversarial nets. In: NIPS. (2014) 2672–2680
34. Goodfellow, I.J.: NIPS 2016 tutorial: Generative adversarial networks. (June 2017)
35. Siegelmann, H.T., Sontag, E.D.: On the computational power of neural nets. *J. Comput. Syst. Sci.* **50**(1) (1995) 132–150
36. Srivastava, R.K., Greff, K., Schmidhuber, J.: Highway networks. In: ICML Deep Learning Workshop. (2015)
37. Shi, X., Chen, Z., Wang, H., Yeung, D., Wong, W., Woo, W.: Convolutional LSTM network: A machine learning approach for precipitation nowcasting. In: NIPS. (2015) 802–810
38. Denton, E., Fergus, R.: Stochastic Video Generation with a Learned Prior. ArXiv e-prints (February 2018)
39. Schüldt, C., Laptev, I., Caputo, B.: Recognizing human actions: A local SVM approach. In: ICPR. (2004) 32–36
40. Soomro, K., Zamir, A.R., Shah, M.: UCF101: A dataset of 101 human actions classes from videos in the wild. ArXiv e-prints (Nov 2012)
41. Revaud, J., Weinzaepfel, P., Harchaoui, Z., Schmid, C.: Epicflow: Edge-preserving interpolation of correspondences for optical flow. In: CVPR. (2015) 1164–1172
42. Sedaghat, N.: Next-flow: Hybrid multi-tasking with next-frame prediction to boost optical-flow estimation in the wild. arXiv e-Prints (Dec 2016)
43. Radford, A., Metz, L., Chintala, S.: Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. In: ICLR. (2016)
44. Theis, L., van den Oord, A., Bethge, M.: A note on the evaluation of generative models. In: ICLR. (2016)
45. Wang, Z., Bovik, A.C., Sheikh, H.R., Simoncelli, E.P.: Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing* **13**(4) (April 2004) 600–612
46. Ebert, F., Finn, C., Lee, A.X., Levine, S.: Self-supervised visual planning with temporal skip connections. In: CoRL. (2017)

# Supplemental material: Predicting the Future with Transformational States

## 1 Video results

In videos included on the project website ([https://daniilidis-group.github.io/transformational\\_states](https://daniilidis-group.github.io/transformational_states)), we visualize prediction results from our model on a large number of sequences from Moving MNIST, KTH, and UCF101. Videos are chosen randomly from the three datasets. In all cases, we show the full sequence given as input to the network (10 frames for Moving MNIST and KTH, 2 frames for UCF101) and the ground truth future sequence along with the network prediction (10 frames for Moving MNIST and KTH, 1 frame for UCF101). Videos are looped and the frames predicted by the network are highlighted in green for clarity.

## 2 Network architectures

Full architectures for CNN encoders, CNN decoders, and the two components of the full RNN core ( $\text{CNN}_\phi$  and RNN) are given for the three datasets below. Parameters for convolutional (Conv), transposed convolutional (TransposedConv), and convolutional LSTM (ConvLSTM) layers are specified as {input feature map spatial dimensions, filter size, number of filters, convolution stride}. Other network elements either have no parameters (tanh and sigmoid activation functions) or always use the default Tensorflow parameter settings (batch norm (BN), leaky rectified linear unit (LReLU)).

Network components are wired together as in Figure 2. Weighted residual connections are used identically in the KTH and UCF architectures. The Moving MNIST architecture omits the residual connection to the output image, but is otherwise identical. Note that CNN encoders and  $\text{CNN}_\phi$  components include an output tanh nonlinearity to make it easier for the network to match their output distributions to that of a ConvLSTM.

**Moving MNIST:****Encoder**

Conv $\{64 \times 64, 4 \times 4, 64, 2\} \rightarrow \text{LReLU} \rightarrow$   
 Conv $\{32 \times 32, 4 \times 4, 64, 2\} \rightarrow \text{BN} \rightarrow \text{LReLU} \rightarrow$   
 Conv $\{16 \times 16, 4 \times 4, 96, 2\} \rightarrow \text{BN} \rightarrow \text{LReLU} \rightarrow$   
 Conv $\{8 \times 8, 4 \times 4, 96, 2\} \rightarrow \text{BN} \rightarrow \text{LReLU} \rightarrow$   
 Conv $\{4 \times 4, 4 \times 4, 128, 1\} \rightarrow \text{BN} \rightarrow \tanh$

**Decoder**

TransposedConv $\{4 \times 4, 4 \times 4, 96, 2\} \rightarrow \text{BN} \rightarrow \text{LReLU} \rightarrow$   
 TransposedConv $\{8 \times 8, 4 \times 4, 96, 2\} \rightarrow \text{BN} \rightarrow \text{LReLU} \rightarrow$   
 TransposedConv $\{16 \times 16, 4 \times 4, 64, 2\} \rightarrow \text{BN} \rightarrow \text{LReLU} \rightarrow$   
 TransposedConv $\{32 \times 32, 4 \times 4, 64, 2\} \rightarrow \text{BN} \rightarrow \text{LReLU} \rightarrow$   
 TransposedConv $\{64 \times 64, 4 \times 4, 1, 1\} \rightarrow \text{sigmoid}$

**RNN**

ConvLSTM $\{4 \times 4, 3 \times 3, 64, 1\} \rightarrow$   
 ConvLSTM $\{4 \times 4, 3 \times 3, 64, 1\} \rightarrow$   
 ConvLSTM $\{4 \times 4, 3 \times 3, 64, 1\}$

**CNN $_{\phi}$** 

Conv $\{4 \times 4, 4 \times 4, 64, 1\} \rightarrow \text{LReLU} \rightarrow$   
 Conv $\{4 \times 4, 4 \times 4, 64, 1\} \rightarrow \text{LReLU} \rightarrow$   
 Conv $\{4 \times 4, 4 \times 4, 64, 1\} \rightarrow \tanh$

**KTH:****Encoder**

Conv $\{128 \times 128, 4 \times 4, 64, 2\} \rightarrow \text{LReLU} \rightarrow$   
 Conv $\{64 \times 64, 4 \times 4, 128, 2\} \rightarrow \text{BN} \rightarrow \text{LReLU} \rightarrow$   
 Conv $\{32 \times 32, 4 \times 4, 256, 2\} \rightarrow \text{BN} \rightarrow \text{LReLU} \rightarrow$   
 Conv $\{16 \times 16, 4 \times 4, 512, 2\} \rightarrow \text{BN} \rightarrow \text{LReLU} \rightarrow$   
 Conv $\{8 \times 8, 4 \times 4, 512, 2\} \rightarrow \text{BN} \rightarrow \text{LReLU} \rightarrow$   
 Conv $\{4 \times 4, 4 \times 4, 256, 1\} \rightarrow \text{BN} \rightarrow \tanh$

**Decoder**

TransposedConv $\{4 \times 4, 4 \times 4, 512, 2\} \rightarrow \text{BN} \rightarrow \text{LReLU} \rightarrow$   
 TransposedConv $\{8 \times 8, 4 \times 4, 512, 2\} \rightarrow \text{BN} \rightarrow \text{LReLU} \rightarrow$   
 TransposedConv $\{16 \times 16, 4 \times 4, 256, 2\} \rightarrow \text{BN} \rightarrow \text{LReLU} \rightarrow$   
 TransposedConv $\{32 \times 32, 4 \times 4, 128, 2\} \rightarrow \text{BN} \rightarrow \text{LReLU} \rightarrow$   
 TransposedConv $\{64 \times 64, 4 \times 4, 64, 2\} \rightarrow \text{BN} \rightarrow \text{LReLU} \rightarrow$   
 TransposedConv $\{128 \times 128, 4 \times 4, 1, 1\} \rightarrow \tanh$

**RNN**

ConvLSTM $\{4 \times 4, 3 \times 3, 128, 1\} \rightarrow$   
 ConvLSTM $\{4 \times 4, 3 \times 3, 128, 1\} \rightarrow$   
 ConvLSTM $\{4 \times 4, 3 \times 3, 128, 1\}$

**CNN $_{\phi}$** 

Conv $\{4 \times 4, 4 \times 4, 128, 1\} \rightarrow \text{LReLU} \rightarrow$   
 Conv $\{4 \times 4, 4 \times 4, 128, 1\} \rightarrow \text{LReLU} \rightarrow$   
 Conv $\{4 \times 4, 4 \times 4, 128, 1\} \rightarrow \tanh$

**UCF:****Encoder**

$\text{Conv}\{256 \times 256, 4 \times 4, 64, 2\} \rightarrow \text{LReLU} \rightarrow$   
 $\text{Conv}\{128 \times 128, 4 \times 4, 128, 2\} \rightarrow \text{BN} \rightarrow \text{LReLU} \rightarrow$   
 $\text{Conv}\{64 \times 64, 4 \times 4, 256, 2\} \rightarrow \text{BN} \rightarrow \text{LReLU} \rightarrow$   
 $\text{Conv}\{32 \times 32, 4 \times 4, 256, 2\} \rightarrow \text{BN} \rightarrow \text{LReLU} \rightarrow$   
 $\text{Conv}\{16 \times 16, 4 \times 4, 512, 2\} \rightarrow \text{BN} \rightarrow \text{LReLU} \rightarrow$   
 $\text{Conv}\{8 \times 8, 4 \times 4, 512, 2\} \rightarrow \text{BN} \rightarrow \text{LReLU} \rightarrow$   
 $\text{Conv}\{4 \times 4, 4 \times 4, 512, 1\} \rightarrow \text{BN} \rightarrow \tanh$

**Decoder**

$\text{TransposedConv}\{4 \times 4, 4 \times 4, 512, 2\} \rightarrow \text{BN} \rightarrow \text{LReLU} \rightarrow$   
 $\text{TransposedConv}\{8 \times 8, 4 \times 4, 512, 2\} \rightarrow \text{BN} \rightarrow \text{LReLU} \rightarrow$   
 $\text{TransposedConv}\{16 \times 16, 4 \times 4, 256, 2\} \rightarrow \text{BN} \rightarrow \text{LReLU} \rightarrow$   
 $\text{TransposedConv}\{32 \times 32, 4 \times 4, 256, 2\} \rightarrow \text{BN} \rightarrow \text{LReLU} \rightarrow$   
 $\text{TransposedConv}\{64 \times 64, 4 \times 4, 128, 2\} \rightarrow \text{BN} \rightarrow \text{LReLU} \rightarrow$   
 $\text{TransposedConv}\{128 \times 128, 4 \times 4, 64, 2\} \rightarrow \text{BN} \rightarrow \text{LReLU} \rightarrow$   
 $\text{TransposedConv}\{256 \times 256, 4 \times 4, 1, 2\} \rightarrow \tanh$

**RNN**

$\text{ConvLSTM}\{4 \times 4, 3 \times 3, 256, 1\} \rightarrow$   
 $\text{ConvLSTM}\{4 \times 4, 3 \times 3, 256, 1\} \rightarrow$   
 $\text{ConvLSTM}\{4 \times 4, 3 \times 3, 256, 1\}$

**CNN<sub>φ</sub>**

$\text{Conv}\{4 \times 4, 3 \times 3, 256, 1\} \rightarrow \text{LReLU} \rightarrow$   
 $\text{Conv}\{4 \times 4, 3 \times 3, 256, 1\} \rightarrow \text{LReLU} \rightarrow$   
 $\text{Conv}\{4 \times 4, 3 \times 3, 256, 1\} \rightarrow \tanh$

### 3 Comparison to other prediction models

In Supplementary Table 1, we compare details of the architecture and training configurations used in various recently proposed architectures for future prediction. Most notably, we produce future predictions without re-encoding predicted images as input for the encoder network, without directly copying from the input sequence, and without using GANs at any point in network training. The gradient difference loss (GDL) is defined in [14].

We strongly encourage the reader to investigate the cited papers for more details: this table is intended only as a road map to the very interesting and growing literature on future prediction.

### 4 Ablation studies

Here, we present qualitative results from ablations of the proposed architecture on Moving MNIST (Supplemental Figure 1) and KTH (Supplemental Figures 2 and 3). On Moving MNIST, we show the results of training the model with and without weighted residual connections. Moving MNIST images are fairly simple,

Table 1: Comparison of sequence prediction model components and training configurations.

	Uses skip connections or copies past?	Re-encodes images to generate predictions after $t=T+1$ ?	Uses LSTMs?	Uses additional labels or training?	Uses GANs?	Loss
BeyondMSE [14]	Uses multi-scale Laplacian pyramid on full sequence (re-)encoding	Yes	No	No	GAN on predicted images	MSE, GDL, GAN
MCNet [16]	Skips from previous frame and difference image re-encoding	Re-encodes images and re-computes difference images	ConvLSTM on difference image encoding	No	GAN on predicted images	MSE, GDL, GAN
DRNet [30]	Copies content vector from last time step	No	LSTM on input sequence embedding	Encoder output trained to disentangle content from pose, content to remain static over a sequence	GAN to disentangle content and pose	Two-stage training: (1) GAN, (2) MSE
SVG-LP [38]	Skips from last input frame encoding	Yes	LSTM on encoder output and LSTM on learned prior	No	No	MSE, KL divergence between model and learned prior
SNA [46]	Skips from previous frame re-encoding and from last input frame	Yes	ConvLSTM layers throughout encoder and decoder	Uses control state and action as additional input	No	MSE
Dual Motion GAN [20]	No	Yes	ConvLSTM on encoder output	Trains network to predict current and future optical flow	GAN on predicted images and predicted current and future flow	GAN, VAE KL divergence
DVF [18]	No	Yes	No	No	No	L1, total variation losses
PredNet [22]	No	Yes	ConvLSTMs throughout architectures	No	No	Predictive coding L1 loss
Adversarial Transformer [17]	Predicted images given as interpolation of last input image pixels	No, last input image directly transformed	No	No	Uses conditional GAN on predicted sequences	GAN
Ours	Masked residual from previous decoder state	No	ConvLSTM on encoder output	No	No	MSE

so residual connections do not lead to as large an improvement in performance as on datasets with real-world image statistics.

On KTH, we compare the full model against models trained (i) with a ConvLSTM core instead of the full RNN core described in the main paper, (ii) using residual connections directly from the last input time step instead of the previous time step (i.e. the decoder at time  $t = T + k$  receives skip connections from the encoder at time  $t = T$  instead of the decoder at time  $t = T + k - 1$ ), (iii) with no skip or residual connections. The full model best captures image motion while also leading to better background in-painting.

All models were trained with the hyperparameters used to train the model described in the paper. On KTH, this produced good results for all models including residual connections. We have seen qualitatively better motion on the model without residuals using different hyperparameters, but the results shown here are representative of the difference between models. Including residual connections led to dramatically better results on background prediction, but the model without residual connections appears to model motion reasonably well in some cases.

Fig. 1: Comparison of Moving MNIST results on architectures with and without residual connections. The model labeled “no residuals” has no skip or residual connections of any kind. The model labeled “w/ residuals” is the full model described in the paper. The model without weighted residual connections produces good predictions, but including these connections produces crisper results, especially at early prediction time steps. Both architectures reliably capture digit identity, even after the digits overlap.

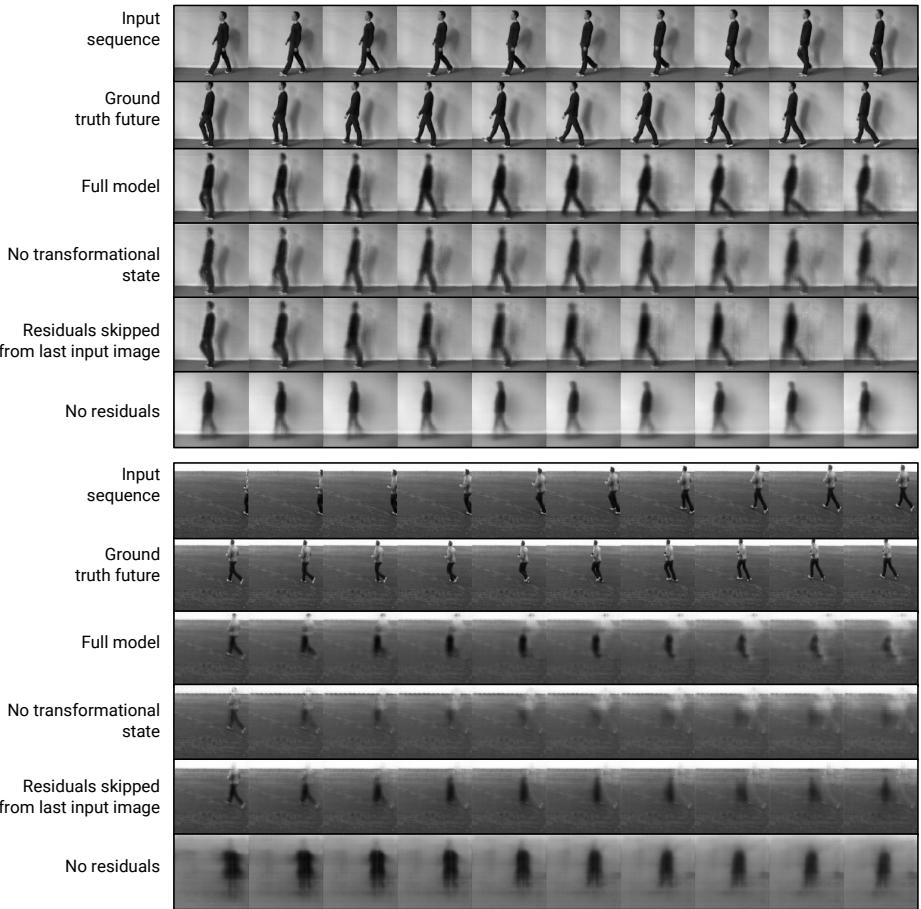


Fig. 2: Comparison of KTH results on models with architectural ablations. (i) “No transformational state”: the RNN core omits the  $\text{CNN}_\phi$  and includes only ConvLSTM components. (ii) “Residuals skipped from last input image”: each decoder directly receives residual input from the encoder at the last input time step ( $t = T$ ) instead of the previous decoder time step. Weighted residuals are still used. (iii) “No residuals”: no residual or skip connections of any kind are used. The second sequence shown here is very challenging for all models. While not perfect, the full model produces better motion (notice the motion of the legs) and less prominent ghosting artifacts than ablations.

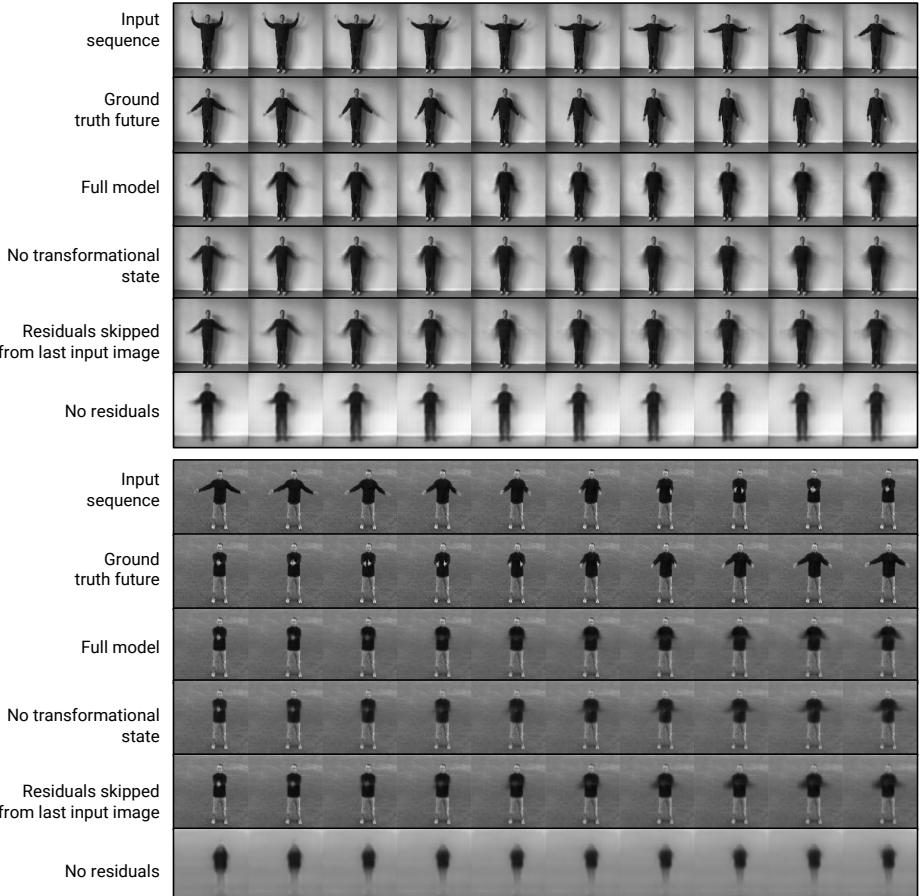


Fig. 3: Additional comparisons of KTH results on models with architectural ablations. See Supplemental Figure 2 caption for explanation of ablations.