Stochastic Image-to-Video Synthesis using cINNs
Supplemental Material

Michael Dorkenwald1  Timo Milbich1  Andreas Blattmann1  Robin Rombach1
Konstantinos G. Derpanis2,3,4*  Björn Ommer1*

1IWR/HCI, Heidelberg University, Germany  2Department of Computer Science, Ryerson University, Canada
3Vector Institute for AI, Canada  4Samsung AI Centre Toronto, Canada

Contents

A Additional Visualizations  1
  A.1. Landscape . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1
  A.2. iPER . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 2
  A.3. DTDB . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 2
  A.4. BAIR . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 2
  A.5. Controllable Video Synthesis . . . . . . . . . . . . . . . . . . . . . . . . . 2
  A.6. Failure Cases . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3
B Implementation Details  3
  B.1. Network Details . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3
  B.2. Training Details . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4
C Evaluation Details  4
  C.1. Diversity Metric . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4
  C.2. Evaluation Protocol . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 5
  C.3. Dynamic Texture FVD (DTFVD) . . . . . . . . . . . . . . . . . . . . . . . . 5

A. Additional Visualizations

For each of our experiments conducted in the main paper, we provide additional video material, consisting of 17 videos in total. To further highlight the benefits of our proposed framework, in the course of our supplemental video material, we compare to five approaches. Due to the collective large size of the videos, the supplemental with the corresponding videos is provided on our project page1.

We next discuss the video material for each experiment individually. Each subsection matches its corresponding file (e.g., ‘A.1.Landscape’ corresponds to ‘...--A1-Landscape’) which contains the discussed video sequences.

A.1. Landscape

For the Landscape dataset [32], we provide the corresponding video (Landscape_samples.mp4) to the samples depicted in Fig. 3 in the main paper. Additionally, we show a qualitative comparison to previous work, i.e., AL [6], DTVNet [33], and MDGAN [32] in Landscape_comparison.mp4, with ‘GT’ denoting the ground-truth. We clearly observe that our model synthesizes more appealing and realistic video sequences compared to the competing methods. While AL produces decent animations in the presence of small motion, when animating fast motions, however, warping artifacts are present, cf. e.g., row 3. These artifacts become even more evident when AL is applied to DTDB (Sec. A.3). In contrast, our method produces realistic looking results in the case of both small and large motions. Next, we evaluate the diversity of the generated samples in Landscape_diversity.mp4. The video contains multiple future progressions for a given starting frame, x0. It can be seen that our approach produces diverse samples capturing a broad range of motion directions, as well as speeds. Moreover, we demonstrate

supplemental_material_222
  |  ++--A1-Landscape
  |  ++--A2-iPER

*Indicates equal supervision.
1https://bit.ly/3dg90fV
in Landscape_longer_duration.mp4 the capability of our model to synthesize longer sequences (48 frames) by sequentially applying our model on the last frame of the previously predicted video sequence.

A.2. iPER

For the iPER dataset [18], we provide the corresponding video (iPER_samples.mp4) to the samples depicted in Fig. 4 in the main paper. We further provide a qualitative comparison to the best performing method IVRNN [3] on iPER in iPER_comparison.mp4 with ‘GT’ denoting the ground-truth. Our method produces more natural motions, e.g., row 3, compared to [3]. Note, that both methods suffer from artifacts due to the low image resolution of $64 \times 64$, such as vanishing hands in motion.

A.3. DTDB

For each dynamic texture from DTDB [9] used in our main paper, we provide examples (Clouds.mp4, Fire.mp4, vegetation.mp4, Waterfall.mp4) for stochastic image-to-video synthesis for random starting frames, $x_0$, comparing our proposed approach to AL [6] and DG [31]. As described in the main paper, DG [31] is directly optimized on test samples, thus overfitting directly to the test distribution. Consequently, we observe that their generations almost perfectly reproduce the ground-truth. Our method produces more natural motions, e.g., row 3, compared to [3]. Note, that both methods suffer from artifacts due to the low image resolution of $64 \times 64$, such as vanishing hands in motion.

A.4. BAIR

In BAIR_comparison.mp4, we provide a qualitative comparison to a strong baseline, IVRNN [3], on the BAIR dataset [5]. While both approaches are able to render the robot’s end effector and the visible environment well, we observe significant differences when it comes to the effector interacting with or occluding background objects. An example of this difficulty can be seen when interacting with the object in the middle of the scene in row 2. IVRNN is unable to depict the object structure and texture during the interaction which results in heavy blur due to averaging over all possible future states. In contrast, this interaction looks much more natural in the video sequence predicted by our model (also row 2). Moreover, the last row (back of the scene, right) illustrates a problem of IVRNN which sometimes occurs in the presence of object occlusions. Specifically, the object which is occluded at the beginning is eventually revealed and is synthesized as a blurry texture, by that, averaging over all possible realizations. Again, our model does not suffer from this problem and correctly handles object occlusions. Additionally, BAIR_diversity.mp4 qualitatively illustrates the prediction diversity of our model by animating a fixed starting frame $x_0$ multiple times. Again, ‘GT’ denotes ground-truth. Our model synthesizes diverse samples by broadly covering motions in the $x, y,$ and $z$ directions.

A.5. Controllable Video Synthesis

In this section, we present qualitative experiments for the following controlled video prediction task: controlled image-to-video synthesis, motion transfer, and controlled video-to-video synthesis.

Controlled image-to-video synthesis. The video Endpoint_BAIR.mp4 illustrates several image-to-video generations while controlling $\eta = (x, y, z)$, the 3D end effector position, similar to Fig. 6 in our main paper. It shows that, while in each example the effector approximately stops at the provided end position (end frame of GT), its movements between the starting and end frame, which are inferred by the sampled residual representations $\nu \sim q(\nu)$, exhibit significantly varying and natural progressions. Moreover, in Direction_Clouds1.mp4 we provide additional video examples for controlling the direction of cloud movements with $\eta$, similar to Fig. 7 in our main paper. We observe that our model renders crisp future progressions (row 2-5) of a given starting frame $x_0$, while following our provided movement control (top row).

Motion transfer. Next, we analyze the application of our model for the task of directly transferring a query motion extracted from a given landscape video $X$ to a random starting frame $x_0$. To this end, we extract the residual representation $\hat{\nu}$ of $X_0$ by first obtaining its video representation $\hat{z} = q(z|X)$ and corresponding residual $\hat{\nu} = T_{\theta}^{-1}(\hat{z}; \hat{x}_0)$ with $\hat{x}_0$ being the starting frame of $\hat{X}$. We use $\hat{\nu}$ to animate the starting frame $x_0$. Transfer_Landscape.mp4 shows that our model accurately transfers the query motion, e.g., as the corresponding direction and speed of the clouds, to the target landscape images (rows 1-3, left-to-right).

Controlled video-to-video synthesis. In controlled video-to-video synthesis, we explicitly adjust the initial factor $\tilde{\eta}$ of an observed video sequence $\tilde{X}$. To this end, we first obtain its video representation $\tilde{z} = q_{\tilde{\theta}}(z|\tilde{X})$ followed by extracting the corresponding residual information $\tilde{\nu} = T_{\theta}^{-1}(\tilde{z}; \tilde{x}_0, \tilde{\eta})$. Subsequently, to generate the video sequence depicting our controlled adjustment of $\tilde{X}$, we simply choose a new value $\tilde{\eta} = \tilde{\eta}^*$ and perform the image-
to-sequence inference process. This can be seen in the video Direction_Clouds2.mp4 using cloud video sequences from DTDB [9]. In each example (second row), the motion direction of the query video (leftmost) is adjusted by the provided control (top row). To highlight that the residual representations \( \nu \) in these cases actually correspond to the query video, we additionally animate the initial image of the query videos by sampling a new residual representation \( \nu \sim q(\nu) \) and apply the same controls (bottom rows). We observe that, while the directions of the synthesized videos are identical, their speeds are significantly different, as desired. In the case of video-to-video synthesis, the movement speed remains the same, in contrast to the image-to-video case, where the movement speed has changed due to the changed residual representation.

A.6. Failure Cases

We highlight here two types of failure cases we observed which are visualized in the video Failure_cases.mp4:

- When the starting frame depicts a complex posture (e.g., folded arms or a leg in the air) on iPER [18] the model has difficulty synthesizing realistic continuations.
- While the Landscape dataset [32] mainly covers naturally progressing cloud motions, there is also a small subset of fast timelapse videos. Due to the underrepresentation of such examples in the dataset, our model struggles to correctly capture fast paced timelapse data without explicitly resorting to data-balancing techniques during training.

B. Implementation Details

Here, we provide a detailed overview of our network architecture as well as the training procedure. The PyTorch [22] implementation of our framework is available on our project page.

B.1. Network Details

**Encoder.** The encoder \( q_\theta(z|X) \) follows the structure of a 3D ResNet [10] using GroupNorm [30] as a normalization layer. Two convolutions with a kernel size of \( 4 \times 4 \) are used to obtain an one-dimensional latent representation for representing the mean \( \mu \) and log variance \( \log \sigma^2 \). During training, we sample from \( q_\theta(z|X) \) using the the reparametrization trick [16, 23].

**Decoder.** The decoder \( p_\varphi(X|x_0, z) \) consists of \( n = 6 \) video residual blocks, with each block followed by nearest-neighbor upsampling to upscale the feature map in space and time (except the last one). This structure is illustrated in Fig. 1. The video representation, \( z \), is inserted into the generator using a fully connected layer matching the initial feature map. The hyperparameters \( \lambda \) and \( \lambda_F \) are both set to 10. The channel factor, \( ch_f \), defines the number of channels and by that, the depth of the model. For BAIR and iPER, we set \( ch_f \) to 64, otherwise we set it to 32. Depending on the dataset, time length, and resolution, the last two up-scaling layers needs to be adjusted. The video representation \( z \) is inserted to the decoder using a fully connected layer matching the initial feature map. We use GroupNorm [30] in SPADE [21] and instance normalization in the ADAIN [12] layer. If the input and the output channels do not match, a \( 1 \times 1 \) convolution is used to adjust the channel dimensions. For matching the output channels, we use a 3D convolution followed by a Tanh activation function. Moreover, spectral norm [20] is used in the decoder.

**Bijective Transformation.** The bijective transformation, \( T_\theta \), is realized as a normalizing flow consisting of a stacked sequence of \( n_f \) invertible neural networks (INNs) operating on the video representation, \( z \). We use \( n_f = 20 \) invertible blocks for all datasets except for BAIR where we set \( n_f = 40 \). Each block consists of actnorm [15], affine coupling layers [4], and fixed shuffling layers, following previous work [24]. Each affine coupling layer is parameterized by two fully connected layers. In every affine coupling layer, we additionally insert the conditioning information following previous work [1, 24]. The feature representation for the starting frame \( x_0 \) is obtained by a pretrained Autoencoder optimized for reconstructing images.

**Discriminators.** For the static discriminator, a patch discriminator [11] is used and for the temporal discriminator a 3D ResNet [10].
B.2. Training Details

The loss objective for the generative model of a video sequence \( X = [x_1, \ldots, x_T] \sim p_X(X) \in \mathbb{R}^{d_x} \) with the corresponding starting frame \( x_0 \in \mathbb{R}^{d_x} \) and a video representation \( z \sim q_\phi(z|X) \in \mathbb{R}^{d_z} \) can be written as

\[
L_{P_\psi, Q_\phi} = \mathbb{E}_{X \sim p_X(X)} \mathbb{E}_{z \sim q_\phi(z|X)} \left[ \lambda \| X - p_\psi(X|x_0, z) \|_1 + \ell_F(X, p_\psi(X|x_0, z)) - D_S(p_\psi(X|x_0, z)) + \lambda_F \ell_F(X, p_\psi(X|x_0, z)) \right] + \beta D_{KL}(q_\phi(z|X)||q(z)) ,
\]

where \( \ell_F \) denotes the feature matching loss [29] to stabilize the training.

The loss objective for the temporal discriminator can be written as

\[
L_{D_T} = \mathbb{E}_{X \sim p_X(X)} \left[ \rho(1 - D_T(X)) + \lambda_{GP} \| \nabla D_T(X) \|_2^2 \right] + \mathbb{E}_{X \sim p_X(X)} \left[ \rho(1 + D_T(p_\psi(X|x_0, z))) \right],
\]

where \( \| \nabla D_T(X) \|_2^2 \) denotes the gradient penalty [19, 8] to stabilize the discriminator training and \( \rho \) the ReLU activation function. The weighting factor \( \lambda_{GP} \) was set to 10.

For the spatial discriminator, the objective can be formulated as

\[
L_{D_S} = \mathbb{E}_{X \sim p_X(X)} [\rho(1 - D_S(X))] + \mathbb{E}_{X \sim p_X(X)} [\rho(1 + D_S(p_\psi(X|x_0, z)))] .
\]

The overall loss objective can be summarized as

\[
L = L_{P_\psi, Q_\phi} + L_{D_T} + L_{D_S} .
\]

Our video synthesis model is trained using Adam [14] with a learning rate of \( 2 \cdot 10^{-4} \), \( \beta_1 = 0.5, \beta_2 = 0.9 \), weight decay of \( 10^{-5} \), and exponential learning rate decay. The dimension of \( z \) is set to \( d_z = 128 \) for all datasets except for iPER, where it is set to 64. The weighting term \( \beta \) of the Kullback-Leibler divergence loss \( D_{KL} \) is set to \( \beta = 1 \cdot 10^{-6} \). For the controllable video synthesis task, we discretize the conditioning \( \nu_1 \) to one-hot vectors. For the 3D end effector position, the \( z \) axis is discretized into 16 bins and the \( x \) and \( y \) axes into 32 bins. For the clouds, the motion direction is discretized into 36 bins. The 3D end effector position was provided by [5] and for the clouds [9] we manually labelled the direction. The normalizing flow, \( T_\theta \), was trained using Adam [14] with a learning rate of \( 1 \cdot 10^{-5} \).

C. Evaluation Details

C.1. Diversity Metric

Besides synthesis quality, diversity is the main criteria we use to evaluate and compare stochastic video synthesis approaches. The assessment of diversity is typically based on measures utilizing feature representations of pretrained models [17, 34]. For instance, SAVP [17] uses a VGG network [26] trained for classification on ImageNet [25] to yield frame-wise representations of video sequences. Based on these representations, videos are compared based on their frame-wise differences measured using a given distance metric. The guiding intuition is that more diverse sample sets should exhibit larger feature differences on average. To this end, SAVP [17] uses the Cosine distance. We argue that this evaluation distance has a major drawback: the Cosine distance only measures the angle between feature vectors, thus discarding crucial information represented by the vector norms. For instance, two data points may lie approximately on a line (i.e., a Cosine distance of 0) but still are located far from each other. Hence, diversity is measured based on incomplete information.

To circumvent this issue, we propose to replace the Cosine distance with the Euclidean distance which also takes the magnitude of a vector into account. Moreover, to explicitly capture temporal information, we also investigate replacing the frame-based VGG feature extractor with an I3D model [27] which directly yields representations that capture the appearance and dynamics of the entire video sequence. Tab. 2 compares the discussed diversity measures. It can be seen that independent of the diversity measure, the order of the approaches is the same. We employ both VGG MSE and I3D MSE measures in our experiments. Note
that the I3D feature extractors have been trained on similar datasets as the videos to be evaluated, i.e., Kinetics [13] for human motion [18] and DTDB [9] for Landscape [32]. Moreover, we report the missing diversity scores based on the I3D [27] from the main paper on Landscape [6] and DTDB [9] in Tab. 1.

C.2. Evaluation Protocol

For comparisons on each dataset, we use the reported numbers from the corresponding paper, where possible, otherwise we use pretrained models or train models from scratch using the code from the official webpage. Here, we list the evaluation protocol for each dataset.

**BAIR [5].** We follow the standard protocol [28] for computing the FVD score by evaluating videos on a sequence length of 16 on a resolution of $64 \times 64$ using all 256 test videos. Diversity is measured by predicting five future progression given the starting frames from all 256 test sequences and computing the Euclidean distance in the VGG-16 [26] as well as in the I3D [27] feature space between the corresponding generated videos.

**iPER [18].** For evaluating the FVD score, we use 1000 randomly sampled sequences from the test set as well as the corresponding generations. Note, for a fair comparison, we concatenate the last conditioning frame to the generated rather than all conditioning frames since previous work condition on up to eight frames. This results in a sequence of length 17 for computing the FVD score. For computing the diversity, we predict five future progressions for each of the 1000 test sequences and measure the diversity based on that.

**Landscape [32].** We create an evaluation set by randomly sampling six times sequences of length 32 from each test video with length over 32 resulting in 918 videos. Based on these sequences, FVD, DTFVD, LPIPS, and FID are computed. This evaluation procedure is the same for each texture. We train one model for AL [6] as well as for our approach on each texture. For diversity, we again generate five future progressions for each sequence of the evaluation set and use the same procedure described for BAIR.

C.3. Dynamic Texture FVD (DTFVD)

In Sec. 4.3 of our main paper, we introduced a dedicated FVD metric for the domain of dynamics textures, the Dynamic Texture Fréchet Video Distance (DTFVD). To this end, we trained a network on DTDB [9] for the task of dynamic texture classification. The motivation behind introducing DTFVD is to provide an additional metric which is sensitive to the types of appearances and dynamics encapsulated by dynamic textures, rather than human action-related motions, as captured by FVD. For the DTFVD network, we use the same architecture as used for the FVD model, i.e., an I3D network [27]. At convergence (cf. Fig. 3), the DTFVD model achieved 81.7% training accuracy, while achieving 84.0% test accuracy, thus indicating that the model yields well generalizing features capturing the appearance and dynamics in DTDB. A similar conclusion can be drawn by looking at the confusion matrix in Fig. 2 computed for the test set of DTDB, which shows a dominant diagonal structure. Note, we used dropout with a probability of $p = 0.5$ to avoid overfitting, which explains why the classification performance is higher on the test set than on the training set. To evaluate sequences with lengths of 16 as well as 32 we train two separate networks.

https://github.com/edouardelasalles/srvp
https://github.com/facebookresearch/improved_vrn
https://github.com/alexlee-gk/video_prediction
https://github.com/jianwen-xie/Dynamic_generator
https://github.com/zilongzheng/STGConvNet
https://github.com/endo-yuki-t/Animating-Landscape
https://github.com/zhangzjn/DTVNet
https://github.com/weixiong-ur/mdgan
Figure 2. Confusion matrix on the test set of DTDB [9] computed from our DTFVD backbone model.

Figure 3. Training and validation loss while optimizing our DTFVD backbone network on a sequence length of 32. Similar accuracy on both dataset splits indicate a well-generalizing model.
References


