Stochastic Image-to-Video Synthesis using cINNs
Supplemental Material

Michael Dorkenwald[1] Timo Milbich[1] Andreas Blattmann[1] Robin Rombach[1]
Konstantinos G. Derpanis[2,3,4*] Björn Ommer[1*]
[1]IWR/HCI, Heidelberg University, Germany [2]Department of Computer Science, Ryerson University, Canada
[3]Vector Institute for AI, Canada [4]Samsung AI Centre Toronto, Canada

Contents

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 page[1].** For each video, multiple cycles are shown (indicated left-bottom) as well as the corresponding video playback rate in frames-per-second (FPS) (right-bottom). The file structure of our provided video material is as follows:

```
supplemental_material_222
    |
    +--A1-Landscape
    |
    +--A2-iPER
    |
    +--A3-DTDB
    |
    +--A4-BAIR
    |
    +--A5-Controllable_Video_Synthesis
    |
    +--A6-Failure_Cases
```

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 the competing methods. Both MDGAN [32] and DTVNet [33] produce blurry videos when using the officially provided pretrained weights and code from the respective webpages. 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, x_0. It can be seen that our approach produces diverse samples capturing a broad range of motion directions, as well as speeds. Moreover, we demonstrate

*Indicates equal supervision.
[1]`https://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 64×64, 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 motion which is most evident for the clouds texture. However, their method suffers from blurring due to optimization using an L2 pixel loss. Similar to the comparisons on the Landscape dataset (Sec. A.1), AL [6] has problems with learning and reproducing the motion of dynamic textures exhibiting rapid motion changes, such as fire. This is explained by the susceptibility of optical flow to inaccuracies when capturing very fast motion, as well as dynamic patterns outside the scope of optical flow, e.g., flicker. Moreover, in the clouds examples (last row) AL wrongly sets the landscape into motion. Our model, on the other hand, produces sharp video sequences with realistic looking motions for *all* textures.

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 η, 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 $\tilde{X}$ to a random starting frame x_0. To this end, we extract the residual representation $\tilde{\nu}$ of $\tilde{X}_0$ by first obtaining its video representation $\tilde{z} = q(z|\tilde{X})$ and corresponding residual $\tilde{\nu} = \mathcal{T}_\theta^{-1}(\tilde{z}; \tilde{x}_0)$ with $\tilde{x}_0$ being the starting frame of $\tilde{X}$. We use $\tilde{\nu}$ 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 $\tilde{z} = q_\phi(z|\tilde{X})$ followed by extracting the corresponding residual information $\tilde{\nu} = \mathcal{T}_\theta^{-1}(\tilde{z}; \tilde{x}_0, \tilde{\eta})$. 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 ν 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 $q_\phi(z|X)$ follows the structure of a 3D ResNet [10] using GroupNorm [30] as a normalization layer. Two convolutions with a kernel size of 4×4 are used to obtain an one-dimensional latent representation for representing the mean μ and log variance $\log \sigma^2$. During training, we sample from $q_\phi(z|X)$ using the the reparametrization trick [16, 23].

Decoder. The decoder $p_\psi(X|x_0, z)$ 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

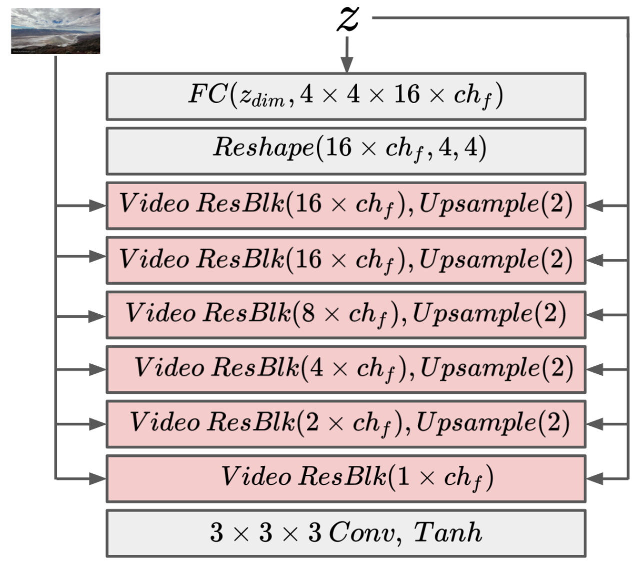

Figure 1. Overview of the decoder structure.

feature map. The hyperparameters λ and λ_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 1×1 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, $\mathcal{T}_\theta$, 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].

Method	Landscape	Fire	Vegetation	Waterfall	Clouds
AL[6]	4.53	0.36	0.30	0.80	1.22
Ours	**5.21**	**1.42**	**1.21**	**1.11**	**1.51**

Table 1. Diversity scores based on the I3D [27] trained on DTDB [9]. The average difference between ground-truth samples are a factor of ~ 1000 smaller for the I3D [27] network trained on DTDB [9] as the one trained Kinetics [13]. For presentation purposes, the numbers in the table have been multiplied by a factor of 1000.

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

$$\begin{aligned}\mathcal{L}_{p_\psi,q_\phi} =& \mathbb{E}_{\substack{X\sim p_X(X)\\ z\sim q_\phi(z|X)}}\Big[\lambda[\| X - p_\psi(X|x_0,z) \|_1 \\ &+ \ell^\phi(X, p_\psi(X|x_0,z))] - \mathcal{D}_T(p_\psi(X|x_0,z)) \\ &- \mathcal{D}_S(p_\psi(X|x_0,z)) + \lambda_F \ell_F(X, p_\psi(X|x_0,z))\Big] \\ &+ \beta D_{\mathrm{KL}}(q_\phi(z|X)||q(z)) \ , \end{aligned} \tag{1}$$

where ℓ_F denotes the feature matching loss [29] to stabilize the training.

The loss objective for the temporal discriminator can be written as

$$\begin{aligned}\mathcal{L}_{\mathcal{D}_T} =& \mathbb{E}_{X\sim p_X(X)}\left[\rho(1-\mathcal{D}_T(X)) + \lambda_{GP} \| \nabla\mathcal{D}_T(X) \|_2^2\right] \\ &+ \mathbb{E}_{\substack{X\sim p_X(X)\\ z\sim q_\phi(z|X)}}[\rho(1+\mathcal{D}_T(p_\psi(X|x_0,z))], \end{aligned} \tag{2}$$

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

For the spatial discriminator, the objective can be formulated as

$$\begin{aligned}\mathcal{L}_{\mathcal{D}_S} =& \mathbb{E}_{X\sim p_X(X)}[\rho(1-\mathcal{D}_S(X)] \\ &+ \mathbb{E}_{\substack{X\sim p_X(X)\\ z\sim q_\phi(z|X)}}[\rho(1+\mathcal{D}_S(p_\psi(X|x_0,z))]. \end{aligned} \tag{3}$$

The overall loss objective can be summarized as

$$\mathcal{L} = \mathcal{L}_{p_\psi,q_\phi} + \mathcal{L}_{\mathcal{D}_T} + \mathcal{L}_{\mathcal{D}_S}. \tag{4}$$

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 β 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 ν_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, $\mathcal{T}_\theta$, was trained using Adam [14] with a learning rate of $1\cdot 10^{-5}$.

Method	VGG Cosine	VGG MSE	I3D MSE
SAVP[†,3] [17]	0.000	0.00	0.01
SRVP[3] [7]	0.040	0.34	1.01
IVRNN[3] [3]	0.023	0.23	0.57
Ours	0.042	0.58	1.76

Table 2. Comparison of different diversity metrics on iPER [18]. [†] SAVP experienced mode collapse due to training instabilities originating from the two involved discriminators. The VGG based feature extractors have been pretrained on ImageNet [25]. The I3D feature extractor has been pretrained on Kinetics [2]. [3] denotes models trained using the official code from their corresponding webpages.

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[2]. 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 64×64 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 progression 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. As explained in the main paper, our model is trained on a sequence length of 16 but applied two times by using the last predicted frame as input for the next prediction. For diversity, we again generate five future progressions for each sequence of the 918 evaluation sequences and use the same procedure described for BAIR.

DTDB [9]. We create an evaluation set by randomly sampling five sequences of length 16 from each test video resulting in between 90 and 385 test sequences depending on the texture. Based on these sequences, the 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.

[2]

https://github.com/edouardelasalles/srvp

https://github.com/facebookresearch/improved_vrnn

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

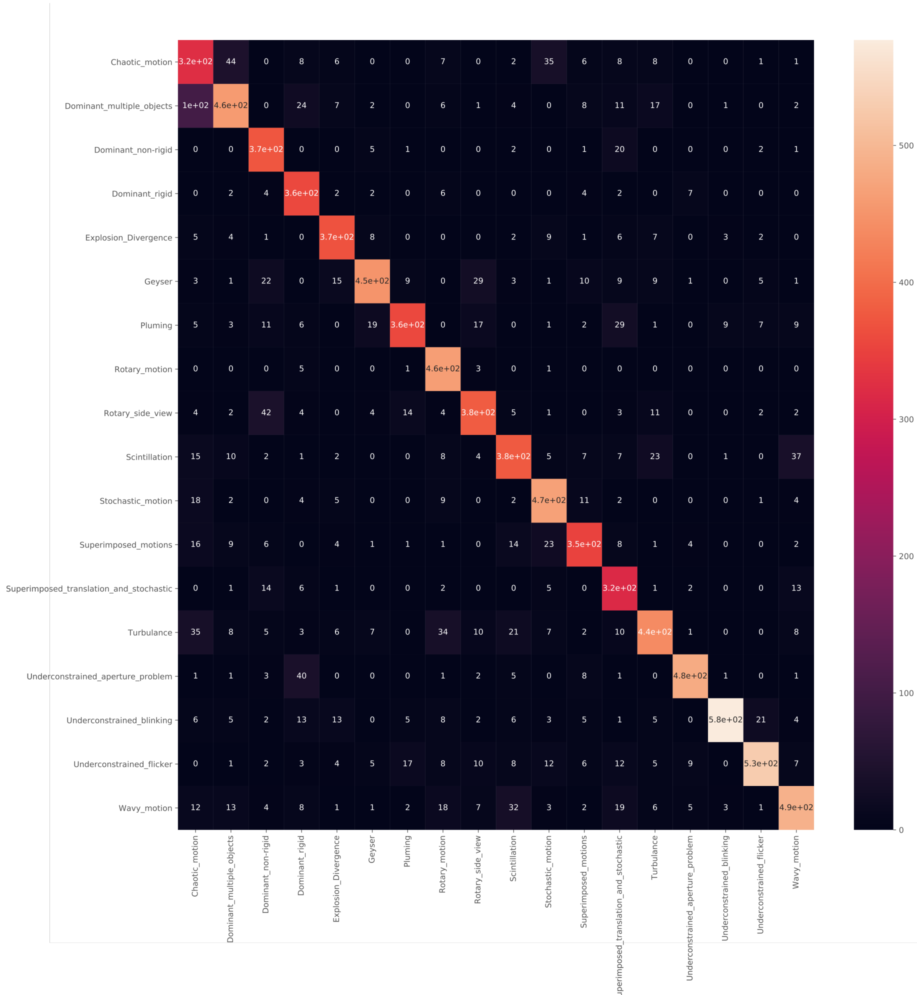

Figure 2. Confusion matrix on the test set of DTDB [9] computed from our DTFVD backbone model.

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

[1] Lynton Ardizzone, Carsten Lüth, Jakob Kruse, Carsten Rother, and Ullrich Köthe. Guided image generation with conditional invertible neural networks. *CoRR*, 2019. 3

[2] João Carreira and Andrew Zisserman. Quo vadis, action recognition? A new model and the kinetics dataset. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4724–4733, 2017. 4

[3] Lluís Castrejón, Nicolas Ballas, and Aaron C. Courville. Improved conditional vrnns for video prediction. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 7607–7616, 2019. 2, 4

[4] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real NVP. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017. 3

[5] Frederik Ebert, Chelsea Finn, Alex X. Lee, and Sergey Levine. Self-supervised visual planning with temporal skip connections. In *Conference on Robot Learning (CoRL)*, pages 344–356, 2017. 2, 4, 5

[6] Yuki Endo, Yoshihiro Kanamori, and Shigeru Kuriyama. Animating landscape: self-supervised learning of decoupled motion and appearance for single-image video synthesis. *ACM Transactions on Graphics*, pages 175:1–175:19, 2019. 1, 2, 4, 5

[7] Jean-Yves Franceschi, Edouard Delasalles, Mickaël Chen, Sylvain Lamprier, and Patrick Gallinari. Stochastic latent residual video prediction. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 3233–3246, 2020. 4

[8] Ishaan Gulrajani, Faruk Ahmed, Martín Arjovsky, Vincent Dumoulin, and Aaron C. Courville. Improved training of wasserstein gans. In *Neural Information Processing Systems (NeurIPS)*, pages 5767–5777, 2017. 4

[9] Isma Hadji and Richard P. Wildes. A new large scale dynamic texture dataset with application to convnet understanding. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 334–351, 2018. 2, 3, 4, 5, 6

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. 3

[11] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5967–5976, 2017. 3

[12] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4401–4410, 2019. 3

[13] Will Kay, João Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, Mustafa Suleyman, and Andrew Zisserman. The Kinetics human action video dataset. *CoRR*, 2017. 4, 5

[14] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015. 4

[15] Diederik P. Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *Neural Information Processing Systems (NeurIPS)*, pages 10236–10245, 2018. 3

[16] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2014. 3

[17] Alex X. Lee, Richard Zhang, Frederik Ebert, Pieter Abbeel, Chelsea Finn, and Sergey Levine. Stochastic adversarial video prediction. *CoRR*, 2018. 4

[18] Wen Liu, Zhixin Piao, Jie Min, Wenhan Luo, Lin Ma, and Shenghua Gao. Liquid warping GAN: A unified framework for human motion imitation, appearance transfer and novel view synthesis. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 5903–5912, 2019. 2, 3, 4, 5

[19] Lars M. Mescheder, Andreas Geiger, and Sebastian Nowozin. Which training methods for gans do actually converge? In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 3478–3487, 2018. 4

[20] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018. 3

[21] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic image synthesis with spatially-adaptive normalization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2337–2346, 2019. 3

[22] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Neural Information Processing Systems (NeurIPS)*, pages 8024–8035. 2019. 3

[23] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 1278–1286, 2014. 3

[24] Robin Rombach, Patrick Esser, and Björn Ommer. Making sense of cnns: Interpreting deep representations and their invariances with inns. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 647–664, 2020. 3

[25] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael S. Bernstein, Alexander C. Berg, and Fei-Fei Li. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision (IJCV)*, 115(3), 2015. 4

[26] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015. 4, 5

[27] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2818–2826, 2016. 4, 5

[28] Thomas Unterthiner, Sjoerd van Steenkiste, Karol Kurach, Raphaël Marinier, Marcin Michalski, and Sylvain Gelly. Towards accurate generative models of video: A new metric & challenges. *CoRR*, 2018. 5

[29] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional GANs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8798–8807, 2018. 4

[30] Yuxin Wu and Kaiming He. Group normalization. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 3–19, 2018. 3

[31] Jianwen Xie, Ruiqi Gao, Zilong Zheng, Song-Chun Zhu, and Ying Nian Wu. Learning dynamic generator model by alternating back-propagation through time. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 5498–5507, 2019. 2

[32] Wei Xiong, Wenhan Luo, Lin Ma, Wei Liu, and Jiebo Luo. Learning to generate time-lapse videos using multi-stage dynamic generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2364–2373, 2018. 1, 3, 5

[33] Jiangning Zhang, Chao Xu, Liang Liu, Mengmeng Wang, Xia Wu, Yong Liu, and Yunliang Jiang. Dtvnet: Dynamic time-lapse video generation via single still image. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 300–315, 2020. 1

[34] Jun-Yan Zhu, Richard Zhang, Deepak Pathak, Trevor Darrell, Alexei A. Efros, Oliver Wang, and Eli Shechtman. Toward multimodal image-to-image translation. In *Neural Information Processing Systems (NeurIPS)*, pages 465–476, 2017. 4