

# Supplementary Materials

Kangyeol Kim<sup>\*1,7</sup>  
[kangyeolk@kaist.ac.kr](mailto:kangyeolk@kaist.ac.kr)

Sunghyun Park<sup>\*†1,2</sup>  
[psh01087@kaist.ac.kr](mailto:psh01087@kaist.ac.kr)

Junsoo Lee<sup>3</sup>  
[junsoolee93@webtoonscorp.com](mailto:junsoolee93@webtoonscorp.com)

Joonseok Lee<sup>4,5</sup>  
[joonseok2010@gmail.com](mailto:joonseok2010@gmail.com)

Sookyung Kim<sup>6</sup>  
[sookim@parc.com](mailto:sookim@parc.com)

Jaegul Choo<sup>1</sup>  
[jchoo@kaist.ac.kr](mailto:jchoo@kaist.ac.kr)

Edward Choi<sup>1</sup>  
[edwardchoi@kaist.ac.kr](mailto:edwardchoi@kaist.ac.kr)

<sup>1</sup> KAIST AI  
 Korea

<sup>2</sup> Kakao Enterprise  
 Korea

<sup>3</sup> Naver Webtoon  
 Korea

<sup>4</sup> Seoul National University  
 Korea

<sup>5</sup> Google Research  
 United States

<sup>6</sup> Xerox Palo Alto Research Center  
 United States

<sup>7</sup> Letsur Inc.  
 Korea

## 1 Background on ODE Formulation

The essence of neural ODEs [5] lies in employing a neural network  $f$  to estimate the vector field of the latent state  $\mathbf{h}$ . Formally,  $\mathbf{h}$  in the continuous domain  $T \in \mathcal{T}$  can be represented as

$$\frac{d\mathbf{h}(t)}{dt} = f_{\theta}(\mathbf{h}(t), t), \quad \mathbf{h}(T) = \mathbf{h}(0) + \int_0^T f_{\theta}(\mathbf{h}(t), t) dt,$$

where  $\theta$  is a set of trainable parameters of  $f$ . Regarding time-series modeling,  $\mathcal{T}$  corresponds to the continuous time domain, thereby allowing the model to characterize the latent state over the continuously evolving time. In our work, the neural ODE is used to model the continuous-time dynamics of the keypoint sequence, which conveys successive geometric information on the object movement.

## 2 IMPLEMENTATION DETAILS

### 2.1 Network Architecture

We provide the architecture details of MODE-GAN, which consists of two stages. All architectures are described for the purpose of generating  $64 \times 64$  videos of 16 timesteps. The architectures of the motion generator and the discriminator are shown in Tables 1 and 2,

\* indicates equal contribution.

† This work was done during an internship at Kakao Enterprise.

© 2021. The copyright of this document resides with its authors.

It may be distributed unchanged freely in print or electronic forms.

Part	Layer	Activation	Output Shape
$z_m$	-	-	128
Function $P$	Linear	ReLU	128
	Linear	ReLU	128
	Linear	ReLU	128
	Linear	-	128
ODE Solver $f_\theta$	Linear	Tanh	128
	Linear	-	128
Function $Q$	Linear	ReLU	128
	Linear	ReLU	128
	Linear	-	2×K
Heatmaps $\mathcal{H}_{1:T}$	-	-	T×K×64×64

Table 1: Motion generator architecture of our model.

Part	Layer	Activation	Output Shape
$z_a$	-	-	128
FC	Linear	LeakyReLU	128
	Linear	LeakyReLU	128
	Linear	LeakyReLU	128
	Linear	LeakyReLU	8192
Generator $G$	$M_t^1$	-	512×4×4
	Upsample	-	512×8×8
	Comp. Block	-	512×8×8
	Upsample	-	512×16×16
	Comp. Block	-	256×16×16
	Upsample	-	256×32×32
	Comp. Block	-	128×32×32
	Upsample	-	128×64×64
	Comp. Block	-	64×64×64
	Conv3×3	Tanh	3×64×64

Table 3: Video generator architecture of our model.

respectively. Also, the architectures of the video generator and discriminator are shown in Tables 3 and 4, respectively.

## 2.2 Evaluation Metric

We evaluate all video generation models via Frechet Inception Distance (FID) [10, 11], measuring the distance between two sets of videos based on their embeddings from a pre-trained feature extractor [12]. Specifically, the video FID [13] is computed as

$$\|\mu - \tilde{\mu}\|^2 + \text{Tr}(\Sigma + \tilde{\Sigma} - 2\sqrt{\Sigma\tilde{\Sigma}}), \quad (1)$$

where  $\mu$ ,  $\tilde{\mu}$ ,  $\Sigma$ , and  $\tilde{\Sigma}$  represent the mean and covariance matrix of real and fake feature maps across all video frames, respectively. Lower FID means the distribution between the feature vectors of real and fake are more similar, hence more realistic fake samples.

Part	Layer	Activation	Output Shape
Heatmap $\mathcal{H}_t$	-	-	K×64×64
Discriminator $D_{fr}^{(l)}$	Conv	LeakyReLU	32×32×32
	Conv	LeakyReLU	64×16×16
	Conv	LeakyReLU	128×8×8
	Conv	LeakyReLU	256×4×4
	Conv	LeakyReLU	256×1×1
Heatmaps $\mathcal{H}_{1:T}$	-	-	1
	-	-	T×K×64×64
Discriminator $D_{sq}^{(l)}$	Conv	LeakyReLU	T×32×32×32
	Conv	LeakyReLU	T/2×32×32×32
	Conv	LeakyReLU	T/2×64×16×16
	Conv	LeakyReLU	T/4×64×16×16
	Conv	LeakyReLU	T/4×128×8×8
	Conv	LeakyReLU	T/8×128×8×8
	Conv	LeakyReLU	T/8×256×4×4
	Conv	LeakyReLU	T/16×256×4×4
	Conv	LeakyReLU	T/16×256×1×1
	Conv	-	1

Table 2: Motion discriminator architecture of our model.

Part	Layer	Activation	Output Shape
Frame $v_t$	-	-	3×64×64
Discriminator $D_{fr}^{(l)}$	Conv	LeakyReLU	64×32×32
	Conv	LeakyReLU	128×16×16
	Conv	LeakyReLU	256×8×8
	Conv	LeakyReLU	512×4×4
	Conv	LeakyReLU	512×1×1
Video $v_{1:T}$	-	-	1
	-	-	T×3×64×64
Discriminator $D_{sq}^{(l)}$	Conv	LeakyReLU	T×64×32×32
	Conv	LeakyReLU	T/2×64×32×32
	Conv	LeakyReLU	T/2×128×16×16
	Conv	LeakyReLU	T/4×128×16×16
	Conv	LeakyReLU	T/4×256×8×8
	Conv	LeakyReLU	T/8×256×8×8
	Conv	LeakyReLU	T/8×512×4×4
	Conv	LeakyReLU	T/16×512×4×4
	Conv	LeakyReLU	T/16×512×1×1
	Conv	-	1

Table 4: Video discriminator architecture of our model.

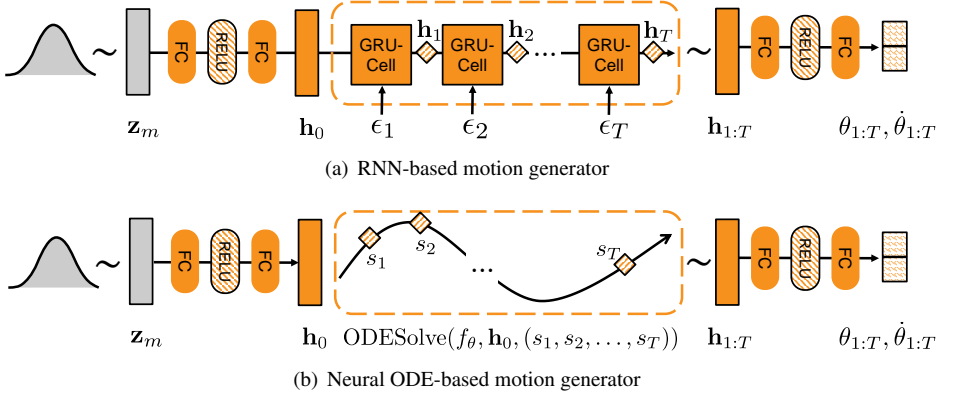


Figure 1: Comparison of RNN and neural ODE-based architectures for pendulum experiment. The only difference lies in the intermediate neural networks that generates the hidden states  $\mathbf{h}_{1:T}$ . For the RNN input, a Gaussian noise  $\epsilon_t$  is used for each timesteps  $t = 1, 2, \dots, T$ .

## 2.3 Adversarial Objectives

**Adversarial Losses at Stage I.** We employ two discriminators  $D_{\text{fr}}^{(\text{I})}, D_{\text{sq}}^{(\text{I})}$ , where each receives Gaussian heatmaps at an individual frame and sequence level, respectively.  $D_{\text{fr}}^{(\text{I})}$  plays a role in enhancing the generated keypoints quality better in single frame-level and  $D_{\text{sq}}^{(\text{I})}$  encourages the generated keypoints sequences to be naturally evolving.

$$\begin{aligned} \mathcal{L}_{\text{adv}}^{(\text{I})} = & \mathbb{E}_{\mathcal{H}_t, \hat{\mathcal{H}}_t} [\log D_{\text{fr}}^{(\text{I})}(\mathcal{H}_t) + \log(1 - D_{\text{fr}}^{(\text{I})}(\hat{\mathcal{H}}_t))] \\ & + \mathbb{E}_{\mathcal{H}_{1:T}, \hat{\mathcal{H}}_{1:T}} [\log D_{\text{sq}}^{(\text{I})}(\mathcal{H}_{1:T}) + \log(1 - D_{\text{sq}}^{(\text{I})}(\hat{\mathcal{H}}_{1:T}))]. \end{aligned} \quad (2)$$

In particular, we employ the WGAN-GP [15] loss as the adversarial loss.

**Adversarial Losses at Stage II.** We use adversarial losses to achieve both goals: (1) the image discriminator loss for generating realistic frames and (2) the video discriminator loss for retaining motion information, where we use the WGAN-GP [15] loss. To this end, we use two motion conditional adversarial losses:  $D_{\text{fr}}^{(\text{II})}, D_{\text{sq}}^{(\text{II})}$  encourages the model to generate realistic frames and retaining motion information.

$$\begin{aligned} \mathcal{L}_{\text{adv}}^{(\text{II})} = & \mathbb{E}_{\mathbf{v}_t, \hat{\mathbf{v}}_t, \mathcal{H}_t} [\log D_{\text{fr}}^{(\text{II})}([\mathbf{v}_t; \mathcal{H}_t]) + \log(1 - D_{\text{fr}}^{(\text{II})}([\hat{\mathbf{v}}_t; \mathcal{H}_t))] \\ & + \mathbb{E}_{\mathbf{v}_{1:T}, \hat{\mathbf{v}}_{1:T}, \mathcal{H}_{1:T}} [\log D_{\text{sq}}^{(\text{II})}([\mathbf{v}_{1:T}; \mathcal{H}_{1:T}]) + \log(1 - D_{\text{sq}}^{(\text{II})}([\hat{\mathbf{v}}_{1:T}; \mathcal{H}_{1:T}))], \end{aligned} \quad (3)$$

where  $[\cdot; \cdot]$  denotes concatenation. We adopt the WGAN-GP [15] loss similar to stage I.

## 2.4 Details of Pendulum Experiments

**Dataset Generation.** We used 1,000 simulated pendulum trajectories with fixed gravity force  $g$  of  $9.81 \text{ m/s}^2$ . The damping factor  $B$ , length of pendulum  $L$ , and mass of bob  $M$  were each sampled from the Gaussian distribution with respective means of 0.2, 1.0, 1.0 and unit variance. For trajectory simulation, Runge-Kutta was used to solve the pendulum equation (first order differential equations) described below:

$$\begin{bmatrix} \frac{d\theta}{dt} \\ \frac{d\dot{\theta}}{dt} \end{bmatrix} = \begin{bmatrix} \dot{\theta} \\ -\left(\frac{g}{L}\right) \cdot \sin(\theta) - \left(\frac{B}{M}\right) \cdot \dot{\theta} \end{bmatrix},$$

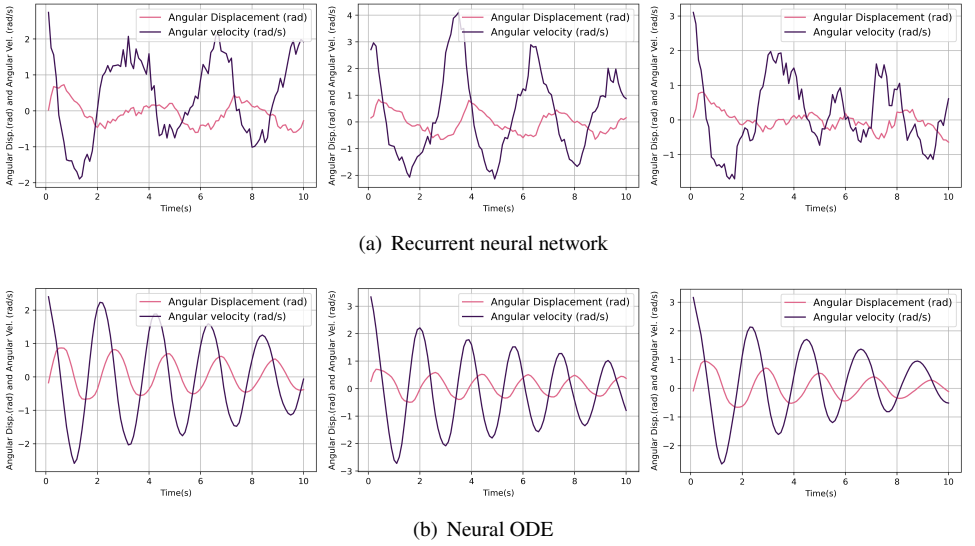


Figure 3: Visualization of generated  $\theta, \dot{\theta}$  using two different motion generators employing (a) RNN and (b) Neural ODE

where  $dt$  is a time unit fixed to 0.1 during data generation.

**Comparison of RNN and ODE Motion Generator.** Fig. 1 shows the detailed architectures of RNN and neural ODE-based pendulum dynamics generator. Both generators are trained using 1,000 simulated pendulum trajectories. After training, the generators should mimic the plausible dynamics of a pendulum by estimating two physical variables  $\theta$  and  $\dot{\theta}$ .

## 2.5 Dataset Preprocessing

Fig. 2 shows the chosen facial keypoints (2, 9, 16, 20, 25, 38, 42, 45, 47, 49, 52, 55, 58th) for *MUG* [10] and *UvA-NEMO* [11]. The facial keypoint selections refer to previous work [8]. We crop and resize to  $64 \times 64$  pixels for both datasets.

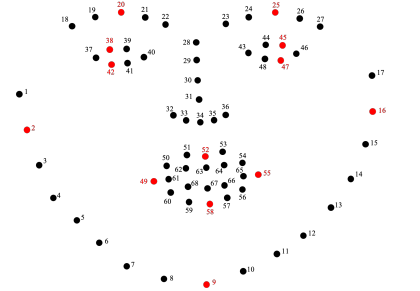


Figure 2: The 68 2D landmarks used by facial landmarks detector model [10]. We use 13 keypoints among them (red).

## 3 ADDITIONAL EXPERIMENTS

In the following section, we provide additional experiments as follows:

- Analysis on the dynamics of each video dataset.
- Qualitative examples from pendulum experiments using both recurrent neural network and neural ODE.
- Qualitative examples of sampled keypoints sequence from our motion generator.



- Qualitative comparison results against unconditional video generation baselines using facial expression datasets.
- Qualitative results of appearance and motion manipulation on *UvA-NEMO* and *MUG*.
- Qualitative and quantitative evaluations on dynamic frame rate.

**Analysis on Dynamics of Datasets.** To analyze the dynamics of each dataset, we includes the video examples of the four training datasets : *Weizmann Action*, *KTH Action*, *MUG*, and *UvA-NEMO*. Due to different frame rates and velocity of human motions in each dataset, the degree of the movement between the adjacent frames varies as shown in Fig. 4. We find out that Human action datasets (*i.e.*, *Weizmannn Action* and *KTH Action*) have relatively large movements compared to facial expression datasets (*i.e.* *MUG* and *UvA-NEMO*). As a result, our model and baselines display more significant movements in the human action videos than in the facial expression videos.

**Examples of Pendulum Experiment.** Fig. 3 shows additional examples of the generated pendulum dynamics from the RNN and the neural ODE based motion generators (See Fig. 1 for detailed architecture). As shown in Fig. 3, the RNN-based motion generator fails to generate smooth dynamics (a). In contrast, our ODE-based motion generator successfully simulates the pendulum dynamics (b).

**Examples of Sampled Motion.** Fig. 5 illustrates the examples of the generated keypoints sequences from our motion generator trained on *Weizmann Action* (purple) or *KTH Action* dataset (blue). Through these examples, we demonstrate that our motion generator has the capability in producing diverse and fluid motions as a form of a sequence of evolving keypoints.

**Comparison with baselines using Facial Expression Datasets.** Fig. 6 presents qualitative comparison of MODE-GAN with VGAN [14], TGAN-v2 [9], MoCoGAN [15] and G<sup>3</sup>AN [16] on two facial expression datasets: *UvA-NEMO* and *MUG*. MODE-GAN shows competitive results compared to the baseline models while most generated videos contain relatively marginal movement than human action cases due to the restricted movement characteristics of facial datasets.

**Examples of Motion and Appearance Manipulation.** As shown in Fig. 7-10, we further demonstrate the ability of MODE-GAN to separately learn the appearance and motion representations. We show the examples of manipulating appearance and motion noise vectors in our video generator trained on *MUG* and *UvA-NEMO* dataset.

**Qualitative Evaluation on Dynamic Frame Rate.** One major strength of MODE-GAN which makes it distinguished with other baselines is its ability to generate videos in continuous-time domain  $\mathcal{T}$ . In other words, MODE-GAN can generate realistic videos in arbitrary frame rates regardless of the frame rate of the training samples, by simply integrating (*i.e.* generating keypoints sequences) over the desired timesteps. (*e.g.*, MODE-GAN can generate video frames at 0.17 or 0.81 second by training with 1.0-second time intervals). In other words, MODE-GAN can flexibly generate a video at an arbitrary frame rate other than the one used at training (*e.g.*, generating video frames at 0.17 or 0.81 second). In this experiment, we compare the quality of Weizmann Action videos generated in various frame rates ( $\{16, 18, 20, 22, 24\}$  FPS) by MODE-GAN and a baseline (MoCoGAN) both trained at 16 FPS. While MODE-GAN learns motion dynamics based on fixed timesteps  $\{s_1, s_2, \dots, s_T\} \subset \mathcal{T}$ , where  $0 < s_1 < s_2 < \dots < s_T$  during training, it can generate videos in arbitrary frame rates by simply integrating over the desired timesteps during inference. In this experiment, we compare the quality of videos generated in various frame rates,

both by MODE-GAN and a baseline, namely MoCoGAN. We define A continuous-time video generation problem aims to generate video frames  $\hat{\mathbf{v}}_{1:L}$  for another set of timesteps  $\mathcal{M} \equiv \{m_1, m_2, \dots, m_L\} \subset \mathcal{T}$ . As a matter of fact, this task can reduce to discrete-time video generation problem as previously tackled if we set  $\mathcal{M} = \mathcal{S}$ . Furthermore, We evaluate the video quality given the arbitrary length query timesteps  $\mathcal{M}$  to demonstrate the capability on synthesizing a video at continuous-time domain. As MoCoGAN produces motion codes via its RNN component only for each 1/16 second, we linearly interpolate two adjacent motion codes from the RNN to obtain the motion representation at unseen timesteps.<sup>1 2</sup> Fig. 11 shows a comparison between synthetic videos generated by MoCoGAN and MODE-GAN at 20 FPS. While MoCoGAN generates frames that looks like a mixture of multiple images, MODE-GAN generates distinct frames, showing the advantage of the ODE-based approach in handling continuous time. These results demonstrate that simply interpolating different motion representations to generate in-between frames yields sub-optimal outcomes (*i.e.* mixed frames at the pixel level).

**Quantitative Evaluation on Dynamic Frame Rate.** Fig. 12 shows the video FID scores of the videos generated at increasing frame rates (from 16 to 24) using the two models trained at 16 FPS. Based on the the monotonic increase of video FID scores by both models, it is evident that the denser the frame rate, the more challenging the video generation becomes. However, the steeper slope of MoCoGAN than MODE-GAN clearly indicates that MODE-GAN is quite robust to the increasing FPS. In other words, MODE-GAN is more comfortable than MoCoGAN in generating videos in continuous-time domain, even at a higher frame rate than the training frame rate.

---

<sup>1</sup>Unlike MODE-GAN and MoCoGAN, other baselines cannot generate or interpolate motion features at arbitrary timesteps.

<sup>2</sup>We take MoCoGAN as a representative RNN-based model for generating motion features at arbitrary timesteps. Note that G<sup>3</sup> AN cannot generate or interpolate motion features at arbitrary timesteps.



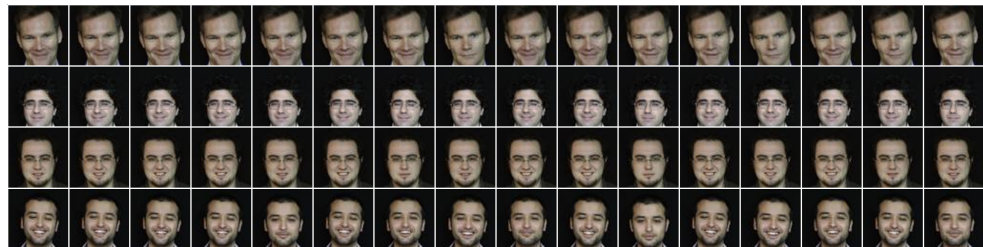
(a) Weizmann Action



(b) KTH Action



(c) MUG



(d) UvA-NEMO

Figure 4: Video samples from each dataset.

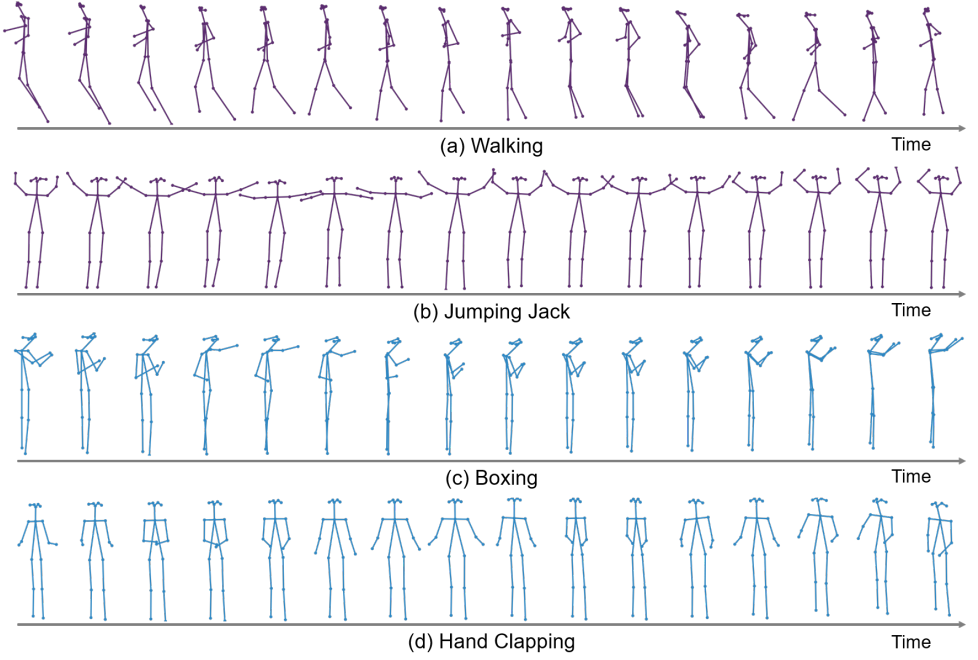


Figure 5: Examples of generated keypoints sequences from our motion generator.



Figure 6: Qualitative comparison with baselines on facial expression datasets.



Figure 7: Qualitative results of our method on the *MUG* dataset. Every three row has the same  $\mathbf{z}_a$  (*i.e.* appearance noise vector) with different  $\mathbf{z}_m$  (*i.e.* motion noise vector).



Figure 8: Qualitative results of our method on the *MUG* dataset. Every three row has same  $\mathbf{z}_m$  (*i.e.* motion noise vector) with different  $\mathbf{z}_a$  (*i.e.* appearance noise vector).



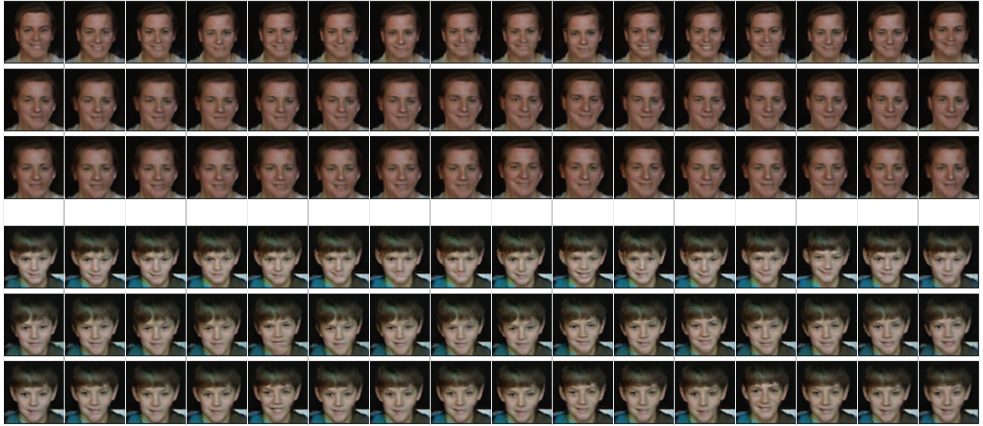


Figure 9: Qualitative results of our method on the *UvA-NEMO* dataset. Every three row has same  $\mathbf{z}_a$  (i.e. appearance noise vector) with different  $\mathbf{z}_m$  (i.e. motion noise vector).

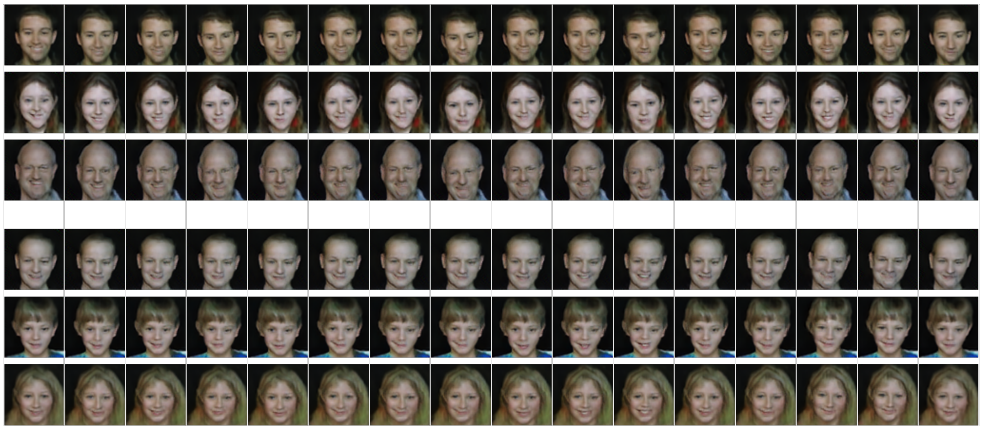


Figure 10: Qualitative results of our method on the *UvA-NEMO* dataset. Every three row has same  $\mathbf{z}_m$  (i.e. motion noise vector) with different  $\mathbf{z}_a$  (i.e. appearance noise vector).

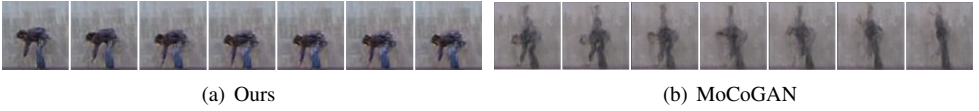


Figure 11: 20-FPS videos generated by two models.

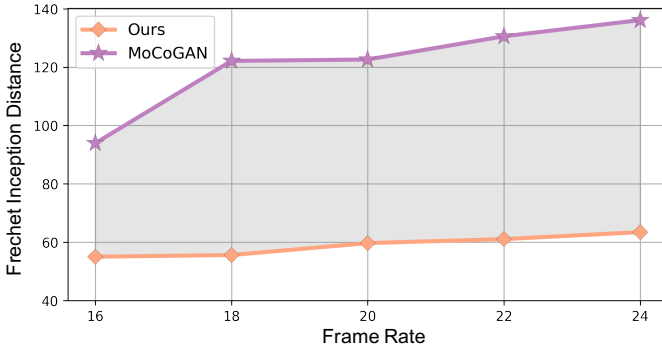


Figure 12: Video FID scores with varied frame rates on Weizmann Action. MODE-GAN consistently outperforms MoCoGAN, showing marginal performance drop to generate at higher frame rate.

## References

- [1] Niki Aifanti, Christos Papachristou, and Anastasios Delopoulos. The MUG facial expression database. In *International Workshop on Image Analysis for Multimedia Interactive Services*, 2010.
- [2] Adrian Bulat and Georgios Tzimiropoulos. How far are we from solving the 2D & 3D face alignment problem? (and a dataset of 230,000 3D facial landmarks). In *Proc. of the IEEE International Conference on Computer Vision (ICCV)*, 2017.
- [3] Tian Qi Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In *Advances in Neural Information Processing Systems (NIPS)*, 2018.
- [4] Hamdi Dibeklioglu, Albert Ali Salah, and Theo Gevers. Are you really smiling at me? spontaneous versus posed enjoyment smiles. In *Proc. of the European Conference on Computer Vision (ECCV)*, 2012.
- [5] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of Wasserstein GANs. In *Advances in Neural Information Processing Systems (NIPS)*, 2017.
- [6] Kensho Hara, Hirokatsu Kataoka, and Yutaka Satoh. Can spatiotemporal 3d cnns retrace the history of 2d cnns and imagenet? In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6546–6555, 2018.
- [7] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems (NIPS)*, 2017.

- [8] Yunseok Jang, Gunhee Kim, and Yale Song. Video prediction with appearance and motion conditions. *arXiv preprint arXiv:1807.02635*, 2018.
- [9] Masaki Saito, Shunta Saito, Masanori Koyama, and Sosuke Kobayashi. Train sparsely, generate densely: Memory-efficient unsupervised training of high-resolution temporal GAN. *International Journal of Computer Vision (IJCV)*, 2020.
- [10] Sergey Tulyakov, Ming-Yu Liu, Xiaodong Yang, and Jan Kautz. MoCoGAN: Decomposing motion and content for video generation. In *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.
- [11] Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. Generating videos with scene dynamics. In *Advances in Neural Information Processing Systems (NIPS)*, 2016.
- [12] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Guilin Liu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. Video-to-video synthesis. In *Advances in Neural Information Processing Systems (NIPS)*, 2018.
- [13] Yaohui Wang, Piotr Bilinski, Francois Bremond, and Antitza Dantcheva. G<sup>3</sup>AN: Disentangling appearance and motion for video generation. In *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.