Towards Practical Single-shot Motion Synthesis

Konstantinos Roditakis Spyridon Thermos Nikolaos Zioulis

Moverse

{kostas,spiros,nick}@moverse.ai

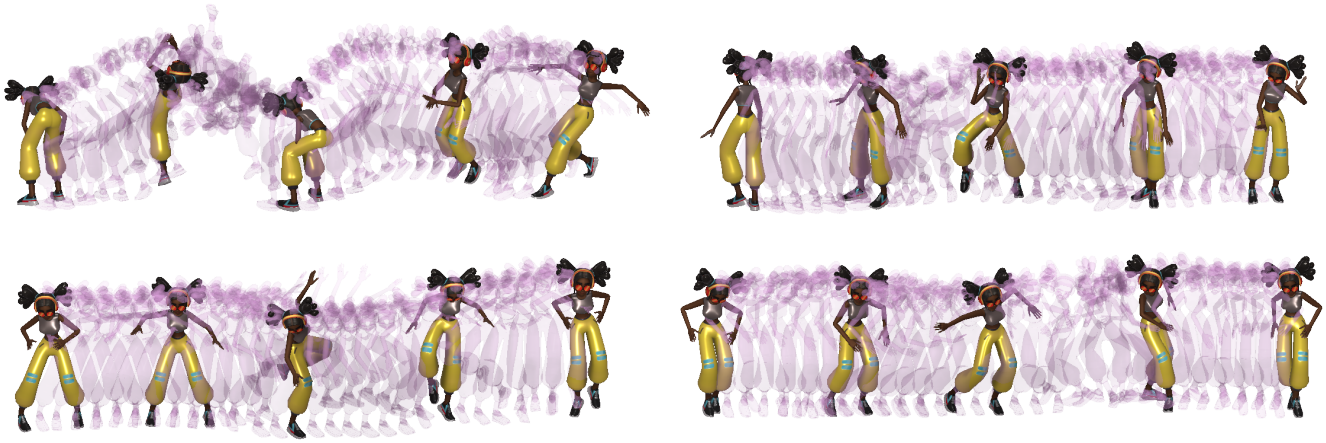

Figure 1. Given a single motion sequence our improved GAN learns to generate motion variations in minutes using mini-batch training and transfer learning, without compromising the quality or the diversity of synthesized motion. Here we generate variations of the Mixamo sequences (raw-major order): a) "breakdance freezes" , b) "dancing", c) "swing dancing, and d) "salsa dancing" .

Abstract

Despite the recent advances in the so-called "cold start" generation from text prompts, their needs in data and computing resources, as well as the ambiguities around intellectual property and privacy concerns pose certain counterarguments for their utility. An interesting and relatively unexplored alternative has been the introduction of unconditional synthesis from a single sample, which has led to interesting generative applications. In this paper we focus on single-shot motion generation and more specifically on accelerating the training time of a Generative Adversarial Network (GAN). In particular, we tackle the challenge of GAN's equilibrium collapse when using mini-batch training by carefully annealing the weights of the loss functions that prevent mode collapse. Additionally, we perform statistical analysis in the generator and discriminator models to identify correlations between training stages and enable transfer learning. Our improved GAN achieves competitive quality and diversity on the Mixamo benchmark when compared to the original GAN architecture and a single-shot diffusion model, while being up to $\times 6.8$ *faster in training time from the former and* $\times 1.75$ *from the latter. Finally, we demonstrate the ability of our improved GAN to mix and compose motion with a single forward pass. Code available at* https://moverseai.github.io/single-shot

1. Introduction

Since the advent of the Large Language Models (LLMs), the so-called "cold-start" generation has attracted impressive attention as a viable path to artificial general intelligence. In fact, LLMs have demonstrated proficiency in various domains [27, 45] transforming text information to images, scenes, and more recently to human pose [9] and motion [21] even coupled with denoising diffusion model [16] variants [22, 42]. However, the said models require massive computational resources and are trained on vast amounts of annotated data, which can include personal and sensitive information. Their lack of interpretability and explainability [49] poses a certain risk that they may inadvertently memorize and reproduce this sensitive data during generation, which raises ethical and intellectual property barriers.

Inference, or using a pre-trained LLM for generating text, also demands significant computational resources. LLMs may reflect and potentially amplify biases present in the training data, leading to biased or unfair outputs. Addressing bias requires careful curation of training data and model design, which can be resource-intensive and challenging.

An interesting alternative to cold-start generation and its challenges are the single sample generative models, which can serve as powerful editing tools as they provide a good balance between pluralism and context preservation. Pioneered in the domain of images, they have been used to remap [38], composite and edit [34] images, increase their resolution [37], and also generate [20] and expand [50] textures. Even though they are very important as they can help overcome intellectual property and data privacy/sensitivity issues, they still remain a relatively unexplored topic. In the context of 3D content, they are even more important as – compared to text, images, and/or video – 3D data are more challenging to acquire in quantity. Specifically for 3D motion, two variants have been recently introduced, GANimator [28] and SinMDM [33] that represent the two dominant classes of approaching this task, namely using generative adversarial networks (GANs) [11] and denoising diffusion probabilistic models (DDPMs) [17]. While both are hyperparameter and architecture sensitive approaches, the latter (SinMDM) was shown to be faster to train than the former (GANimator), as well as support more applications without re-training. On the other hand, GANimator relies on a single forward pass, and thus, exhibits much faster inference performance compared to the iterative nature of diffusion. Content editing applications need to support interactive workflows (*i.e.* real-time inference) but at the same time the workflows are on-demand, which also imposes constraints on the set up time (*i.e.* fast training).

In this work we focus on improving the training time of GANimator that already delivers real-time inference, taking a step towards the practical realization of single motion generative editing. We find that a major limitation of single sample GANs compared to DDPMs is the lack of mini-batch training. The latter is a challenge for single sample training, especially in the unconditional case, a task that needs to balance adversarial training with a latent anchoring objective. Further, we show that the hierarchical nature of GANimator is an unexplored trait that can be exploited to improve training time and realize more editing applications in a unified manner and without retraining.

Summarizing, our contribution is two-fold:

- We study the challenges for mini-batch and hierarchical training in the single sample GAN regime and show increase its performance by a factor of 10 through combining mini-batch training and cross-stage transfer learning. Our GANimator variant trains faster than SinMDM and simultaneously offers real-time inference performance.
- We show how we can realize diverse motion compositing in a single forward pass by exploiting the hierarchical nature of GANimator, expanding its application domain.

2. Prior Art

Data-driven motion generation. Synthesizing human motion is a long standing problem with some of the early approaches being based on statistical modelling [8], exemplar [3, 4] or graph walk [24] based composition, and learning the distribution of novel constructs like motion textons [29]. Given larger datasets and modern learning techniques it is now possible to generate motion from text [6, 32, 42], offer a GPT-like interface for motion [21] and use said models to perform editing tasks [5, 36]. More general motion synthesis and editing frameworks have also been presented [18] due to the wider availability of large scale motion datasets. Still, large dataset acquisition is challenging, especially when considering text prompt annotations. It also comes with barriers related to the sensitivity of the data either from a personal or creative point of view.

Single-shot generation. Simple sample generative models are a promising alternative to overcome these barriers as they train specialized models that generate variations of a specific sample only. InGAN [38] first showed that it is possible to train a conditional GAN model on a single image for the task of remapping, with follow-up works focusing on texture generation and editing [7, 20, 26]. Following a progressive learning scheme across multiple stages, each operating on a different scale, SinGAN [34] is the first unconditional GAN trained on a single image. Crucially it relies on a patch-based discriminator [19, 26] restricting its receptive field, which is coupled with a reconstruction objective that anchors its latent space protect the model training from a mode collapse.

Nonetheless, this increases the training time, which – when considering the single sample context – is a big obstacle for practical application use. Different variants followed, with ExSinGAN [47] using external priors to improve structural and semantic performance of the model, while ConSinGAN [15] focused on improving the training time by training stages in parallel and reducing their number. To improve the preservation of sample's context, OneShotGAN [40] introduced a dual discriminator to supervise both the global context, as well as the patch-based layout. PetsGAN [48] improves training time by leveraging external priors whereas HP-VAE-GAN [14] opts for a hybrid VAE-GAN scheme to enable single video generation. The challenge of quickly training single sample generative models mostly stems from the nature of the adversarial game, and thus, novel approaches that reformulate the task to a reconstruction [44] or nearest neighbor retrieval [12] one manage to greatly accelerate training time, but at the expense of generation variance. Using diffusion mod-

els, like SinDDM [25], is another alternative to reducing training time due to mini-batch training. The interaction between the added reconstruction objective with the adversarial game is difficult to balance, and is the reason why single sample GANs are typically trained with a batch size of one as training destabilizes when increasing the batch size.

Single-shot 3D content generation. More recently, scarce efforts have been made to demonstrate single sample generation to 3D content. Sin3DM [43] trains a diffusion model on a single sample, and leverages and intermediate latent representation to overcome memory and runtime performance issues. SinGRAF [39] bridges a neural rendering representation to accomplish single sample variation generation of specific 3D scenes. For 3D motion single sample generation there exist two approaches, GANimator [28] and SinMDM [33], using adversarial training and denoising diffusion respectively. GANimator consists of 7 stages of skeleton-aware convolutions [1] forming 4 pyramid levels (2-2-2-1 stages) emulating the pyramidal design of SinGAN in the temporal domain, where each stage learns to generate a sequence of different length (up to the input's one). Changing the core of the aforementioned works, Raab *et al*. [33] present a motion diffusion model that learns to generate variations of a single motion sequence. SinMDM follows the structure of the UNet-based diffusion model presented in [30] but with a significant detail; they add shift-invariant local attention layers [2] to decrease the receptive field of UNet and avoid overfitting in the single sample training scenario. GANimator is slower to train but significantly faster when generating samples, whereas SinMDM exploits mini-batch training to reduce training time but requires an iterative diffusion process at inference. Further, SinMDM was shown to support more applications without requiring retraining, an important advantage from a practical point of view.

3. Efficient Novel Motion Synthesis

In this section we present our findings for improving the performance of single-shot motion synthesis in more detail, starting with enabling mini-batch training, followed by transfer learning between stages.

3.1. Background

We present the background of GANimator [28] and formalize the notations to set the stage for our improvements in the GAN training process.

Data representation. Li *et al*. [28] form a motion representation $\mathcal{M}_T \equiv \mathbb{R}^{T\times(JQ+C+3)}$, where T indicates the number of frames, J is the number of skeleton joints, $Q = 6$ corresponds to the 6D rotation representation of the joints, and C indicates the foot contact labels followed by the 3D representation of the root joint including the x- and z-axis velocity and the y-axis position.

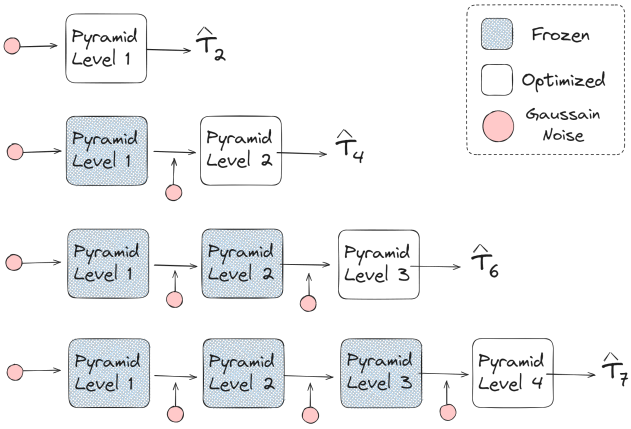

Figure 2. A schematic representation of the GANimator [28] gradual training architecture. From top to bottom, each pyramid level is learning to generate motion features at time scale ($\hat{T}_*$). When a pyramid level is trained it serves as a frozen feature extractor for the next level. Note that the last pyramid level (L_4) consists of only one $\{G(\cdot), D(\cdot)\}$ pair.

GAN architecture. GANimator follows a coarse-to-fine motion feature learning approach, with S stages of generators $G(\cdot)$ and discriminators $D(\cdot)$ pairs. The GAN model does not train all stages in an end-to-end manner, but follows a gradual learning approach, as shown in Fig. 2. The stages are groups of pairs of $\{G(\cdot), D(\cdot)\}$ forming 4 pyramid levels L_* - except for L_4 that contains only S_7. Each level's stage learns to generate the motion features of increasing temporal resolution. For the rest of the paper we use the subscript to denote stage levels and the superscript to denote the pyramid levels (*e.g*. $\hat{T}_2^1$ corresponds to the generated motion features from the second $G(\cdot)$ of the first pyramid level). Each generator $G(\cdot)$ and discriminator $D(\cdot)$ consists of 4 skeleton-aware convolutions [1] followed by leaky ReLU activations (except for the final convolution layer). As depicted in Fig. 3, G_1^1 is responsible for learning the mapping between the sampled noise and the motion representation denoted as $\hat{T}$+1, *i.e*. $\hat{T}_1^1 = G_1^1(z_1)$, while the rest $G^*_{\{2,\dots,S\}}(\cdot)$ form a hierarchical auto-regressive process that progressively upsamples the generated sequence:

$$\hat{T}_i^\ell = G_i^\ell(\hat{T}_{i-1}^\ell, z_i),\ i \in \{2,\dots,S\},\ \ell \in \{1,\dots,L\}, \quad (1)$$

where z_i is sampled from an i.i.d. normal distribution $\mathcal{N}(0, I)$ and multiplied with an decreasing amplitude σ_i.

Losses. The model is trained with multiple losses for securing an equilibrium in the adversarial game between generators and discriminators, having the constraint that there is only a single sample to train the model. Both $G(\cdot)$ and $D(\cdot)$ are supervised with the Wasserstein variant from [13]:

$$\mathcal{L}_{adv} = \mathbb{E}_{\hat{T}_i\sim\mathbb{P}_{G_i}}[D_i(\hat{T}_i)] - D_i(T_i) + \lambda_{gp}\mathbb{E}_{\tilde{T}_i\sim\mathbb{P}_{G_i}}\left[\left(||\nabla D_i(\tilde{T}_i)||_2 - 1\right)^2\right],$$

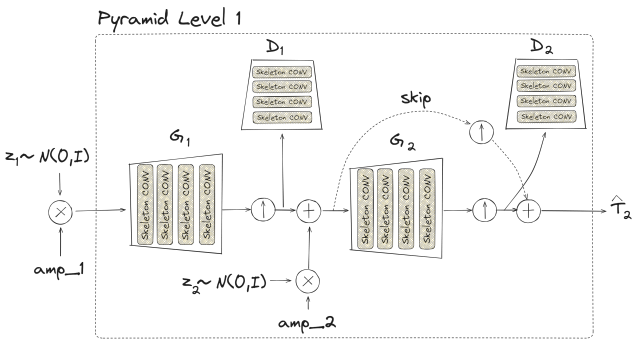

Figure 3. GANimator's pyramid level 1 in detail: G_1 is responsible for learning the mapping between sampled noise $z_1 \sim \mathcal{N}(0, I)$ (multiplied by a predefined amplitude) and the motion representation T, which is then upsampled ($\hat{T}_1$) and sent to D_1; before G_2 we add extra noise to the predicted motion features $\hat{T}1$ to force the model to learn variations of the input sample.

where $\mathbb{P}$ denotes a learned distribution and $\tilde{T}_i = \alpha\hat{T}_i + (1-\alpha)T_i$ is a linear combination of the generated and ground truth motion features, respectively. Note that the gradient penalty regularization in Eq. 3.1 enforces Lipschitz continuity and stabilizes the training. To prevent mode collapse due to the single-sample training the $G(\cdot)$ are additionally supervised by an $L1$ reconstruction loss:

$$\mathcal{L}_{rec} = ||G_i(\hat{T}_{i-1}, z_i^*) - T_i||_1, \tag{2}$$

where z_i^* denotes a predefined noise code (at different levels of amplitude $i \in \{1, \ldots, S\}$) that is set to approximate the reconstruction of T. Lastly, a regularization loss is added to supervise the contact of the predefined set of joints, denoted as $\mathcal{C}$, with the ground:

$$\mathcal{L}_{con} = \frac{1}{T|\mathcal{C}|} \sum_{j \in \{\mathcal{C}\}} \sum_{t=1}^{T} ||\mathcal{V}^{t,j}||_2^2 \cdot \mathrm{S}(\mathcal{C}^{t,j}), \tag{3}$$

where S denotes the skewed Sigmoid function that forces the output to be almost binary, and $\mathcal{V}$ is the velocity computed for the joints of $\mathcal{C}$ using forward kinematics.

Although each loss operates on a different part of the GAN training process, we can summarize them as:

$$\mathcal{L} = \lambda_{adv}\mathcal{L}_{adv} + \lambda_{rec}\mathcal{L}_{rec} + \lambda_{con}\mathcal{L}_{con}, \tag{4}$$

where $\lambda_{adv}, \lambda_{rec}, \lambda_{con}$ as well as λ_{GP} from Eq. 3.1 are responsible for weighting the contribution of each loss to the adversarial game of single-sample motion learning. Li *et al*. [28] discuss the contribution of each loss to the final result, while our focus is concentrated in the balancing between the losses.

3.2. Mini-batch Training

One major advantage of the single-shot diffusion models vs single-shot GANs is the exploitation of mini-batch training.

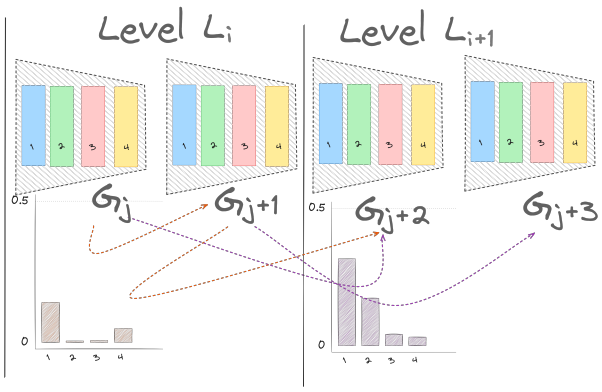

Figure 4. Representation similarities across stages and levels. Each generator (G_j) exhibits low representation similarity scores with the following generator (G_{j+1}) (orange dashed arrows & plots). Yet, we find that the corresponding stages' generators ($G_j \longleftrightarrow G_{j+2}$) across levels ($L_i$) exhibit higher similarity scores for the early layers only (1 & 2 – purple dashed arrows & plots). Transferring the trained generators' (G_j) layers' weights to the next level's (L_{i+1}) corresponding stages' generators (G_{j+2}) before training them, improves convergence rate.

In fact, finding the equilibrium in the adversarial game of a single-shot GAN is challenging and the mode collapse is a common result when trying to set the batch size larger than 1. As discussed in [35], in a data-driven adversarial game mini-batching helps the discriminator to understand when the generator produces samples of very low variation and avoid mode collapse due to this side information. In single-shot generation, the outputs of the generator are by definition minor variations of the single input sample, while the narrow receptive field of the patch-based discriminator and the reconstruction loss $\mathcal{L}_{rec}$ are responsible for preventing the mode collapse. Naturally, increasing the batch size counters the intuition that the narrow receptive field of the discriminator will prevent mode collapse, while the reconstruction loss tends to reduce the coverage by forcing the average of the batch to be similar to the input sequence. Since mini-batching critically improves the training time, we performed an ablation study on the weights of $\mathcal{L}_{rec}$ and $\mathcal{L}_{adv}$ in a quest for finding the correct combination that preserves the equilibrium in the adversarial game. However, note that this is not a straight-forward weight tuning process since each loss operates on different parts of the training process (*i.e*. different optimizer). Starting from the original weight values $\lambda_{adv} = 1$ and $\lambda_{rec} = 50$ we choose a stage-based linear annealing of: a) λ_{adv}, b) λ_{rec}, c) both λ_{adv} and λ_{rec}. As shown in Table 1, we achieve the best results with (c) when we boost $\mathcal{L}_{adv}$ in the early stages where mapping from sampled noise to motion representation takes place, while the later stages are dominated by the reconstruction loss to ensure that no rare mini-motions are omitted.

Table 1. Ablation study of the model parameters for the GANimator [28] architecture. The demonstrated results correspond to motion "salsa dance" of the Mixamo benchmark. Although the GAN variant with batch size 24 (Abl #10) is the fastest to train, Abl #9 exhibits the best trade-off between quality/diversity and train time.

	Quality & Diversity						Model Parameters					Performance
	Coverage (↑)	Global Div. (↑)	Local Div. (↑)	SIFID (↓)	Inter Div. (↑)	Intra Div. (↓)	BS	Iters	l_adv	l_rec	TL	Train Time (↓)
Baseline	1.00	1.12	1.05	3.47	1.67	1.72	1	105k	1	10		5h30m
Abl #1	1.00	0.98	0.92	3.61	1.19	1.90	1	210k	1	10		10h21m
Abl #2	1.00	0.86	0.81	4.49	0.73	2.37	1	105k	5	10		5h30m
Abl #3	1.00	0.71	0.67	3.41	0.47	1.87	1	210k	5	10		10h21m
Abl #4	0.96	1.51	1.41	4.66	1.80	2.48	16	105k	5	10		0h52m
Abl #5	1.00	1.31	1.22	3.36	1.72	2.31	16	210k	5	10		1h47m
Abl #6	1.00	1.26	1.18	3.99	1.99	2.26	16	210k	5	100		1h47m
Abl #7	1.00	1.34	1.27	4.59	2.17	2.70	16	[210, 210, 105, 52.5]	5	100	y	0h42m
Abl #8	1.00	1.43	1.34	4.20	1.95	2.23	16	210k	[5.0,5.0,2.5,1.0]	[50,75,100,100]		1h47m
Abl #9	**1.00**	**1.41**	**1.32**	**3.93**	**1.93**	**2.30**	**16**	**[210, 210, 105, 70]**	**[5.0,5.0,2.5,1.0]**	**[50,75,100,100]**	**y**	**0h48m**
Abl #10	0.96	1.58	1.48	4.98	2.07	2.32	24	105k	5	10		0h29m
Abl #11	0.99	1.43	1.33	5.43	1.76	2.38	24	210k	5	100		0h57m

3.3. Cross-Stage Transfer Learning

In addition to increasing convergence via larger mini-batch training, the hierarchical structure of single sample GANs can be exploited to improve convergence rates. ConSinGAN [15] trained multiple stages in parallel, and re-used the discriminator from the previous training stage to improve performance by transferring its weights to initialize the next stage discriminator. GANimator [28] already trains two stages in the same pyramid level but does not perform discriminator transfer across stages or pyramid levels, even though the same discriminator is used – in terms of architecture and capacity, similar to ConSinGAN and in contrast to SinGAN that increases discriminator capacity at each stage.

Curiously, re-using the generator weights does not lead to improved performance. To study this we turn to studying the neural network representations using finer grained layer-wise analysis. It has been shown that the similarity of representations across layers can be measured despite the higher dimensionality of the representations [23]. We perform linear centered kernel alignment (CKA) for all stage combinations across all pyramid levels.

Our results, visualized in Fig. 4, show that despite the hierarchical approach using the same model and capacity for each stage S, most layers lie at low similarity scores. Yet we find that the early layer representations across the corresponding stages of each pyramid level, exhibit higher levels of similarity. Contrary to ConSinGAN that outputs features at each level apart from the last, GANimator reconstructs the output motion at each level. Therefore the early convolutional layers operate on similar features and, as indicated by their CKA scores, extract similar representations, whereas the latter layers apply the level's motion motifs, style and details.

Based on this analysis, we design a generator transfer learning scheme across levels and stages where each generator's (G_j^i) early layer weights are initialized from the early layer weights of the corresponding generator of the previous level (G_{j-2}^{i-1}).

This ensures that each next training stage is closer to the converged state, thus allowing us to reduce the number of iterations significantly, further boosting single sample GAN training time.

4. Experiments

Our experimental section is split into the quantitative analysis, discussing the metrics of the literature and comparing our improved approach with the state-of-the-art (SoTA), and the qualitative analysis that consists of new applications for single-shot GAN-based generation.

4.1. Quantitative analysis

Fist, we briefly report the metrics used in the literature for assessing the quality and the diversity of single-sample generation, present our implementation details, and discuss our results compared to similar models [28, 33].

Metrics. For a fair comparison we use the metrics presented in [28, 33], which try to measure the local and global diversity of the generated sequences, as well as their quality in terms of plausibility and coverage. We give a brief description for each metric, commenting on its usefulness.

Quality: The combination of the coverage and the plausibility expressed as distance from a distribution form a robust pair for understanding how realistic is the generated motion. Li *et al.* [28] consider a temporal window T_w of the input sequence covered if its distance from its nearest neighbor Q_w in the generated sequence is less than a predefined threshold ϵ; on the other hand, SinMDM adopts the Fréchet Inception Distance (FID) variant from [34], which uses the deep features of an earlier convolutional layer of the Inception Network [41] in order to compute the FID statistics between the input and the generated sequences. As noted in [33], coverage has been experimentally shown to be sensitive, thus we choose to interpret it jointly with the single sample FID (SiFID) to truly describe quality.

Diversity: [28] and [33] compute the local and global diversity of the generated sequence following different approaches; Li *et al.* [28] use high-level features (*i.e.* rotation angles) to compute distances either from the nearest

Table 2. Quality, diversity and performance comparison between our improved GAN, GANimator [28] and SinMDM [33]. We present the average results on the Mixamo benchmark. The presented train and inference times are measured on a NVIDIA RTX 3060 GPU on that "salsa dance" with character "Joe" sample from Mixamo. Following [33] we compute the harmonic mean of the 6 metrics to provide a balanced overall result for each experiment.

	Quality & Diversity							Performance	
	Coverage (↑)	Global Div. (↑)	Local Div. (↑)	SIFID (↓)	Inter Div. (↑)	Intra Div. (↓)	Harm. Mean (↑)	Train Time (↓)	Inf. Time (↓)
[28]	0.95	1.02	0.96	1.15	1.49	2.12	0.50	5h30m	**5.2ms**
[33]	0.94	1.42	1.00	1.08	1.43	1.93	0.58	1h24m	5000ms
Ours	0.89	1.46	1.35	1.99	1.75	1.57	0.52	**0h48m**	**5.2ms**

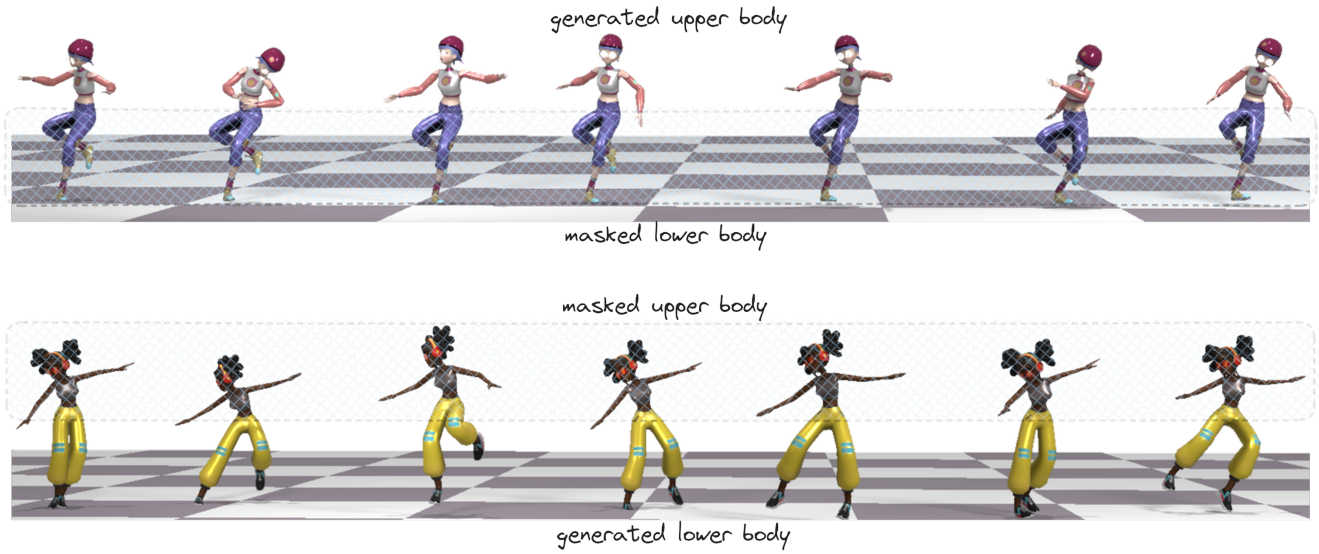

Figure 5. Body-part composition: (top) snapshots of 7 generated "swing dancing" motion variants with the lower body masked (shaded area) to preserve the original sequence pose, while the upper body is randomly generated; (bottom) snapshots of 7 generated "swing dancing" motion variants with the upper body masked to preserve the original sequence pose, while the lower body is randomly generated. Note that the root joint is considered as part of the lower body, thus the top samples retain the global orientation of the original motion.

neighbors of the input sequence, while Raab *et al*. [33] use embeddings of motion features from a pre-trained motion encoder for computing the corresponding distances. From our experiments, we confirm the superiority of deep features over raw input features [46] in interpreting the inter-diversity and the intra-diversity of the generated sequences. However, we run our evaluation using all presented metrics, as well as the harmonic mean from [33] that attempts to describe both quality and diversity with one value.

Data. We use the Mixamo[1] sequences presented in [33] to evaluate our improved GAN against the GANimator and the SinMDM in terms of training time, inference time, quality and diversity using the aforementioned metrics.

Implementation details. We use the provided PyTorch [31] implementations for GANimator and SinMDM and adopt the pretrained motion encoder from the SinMDM codebase to extract the motion embeddings for SiFID and inter/intra diversity metrics computation. All experiments are conducted on a NVIDIA RTX 3060 GPU.

[1] https://www.mixamo.com

Results. Table 1 details the performed ablation study for improving the training performance of GANimator by validating the two presented techniques, *i.e*. mini-batch training with equilibrium preservation and transfer learning between pyramid levels. As shown in the results, increasing the batch size significantly improves the training time, but should be accompanied with an increase of training iterations, λ_{adv}, and λ_{rec}. However, larger batch sizes seem to hurt the $G(\cdot) - D(\cdot)$ adversarial game and leads to inferior performance despite decreasing the training time. After having improved training time and demonstrated similar performance with the baseline, we move to exploit the correlations between pyramid levels. From Table 1, we can see that the combination of cross-stage transfer for the generators $G(\cdot)$, and annealing λ_{adv} and λ_{rec} leads to the best results, while exhibiting the best improvement in training time. As a next step, we compare our best model with GANimator [28] and SinMDM [33]. From the results in Table 2, we conclude that our improved GAN achieves slightly better results that its baseline [28] when tested on the Mixamo

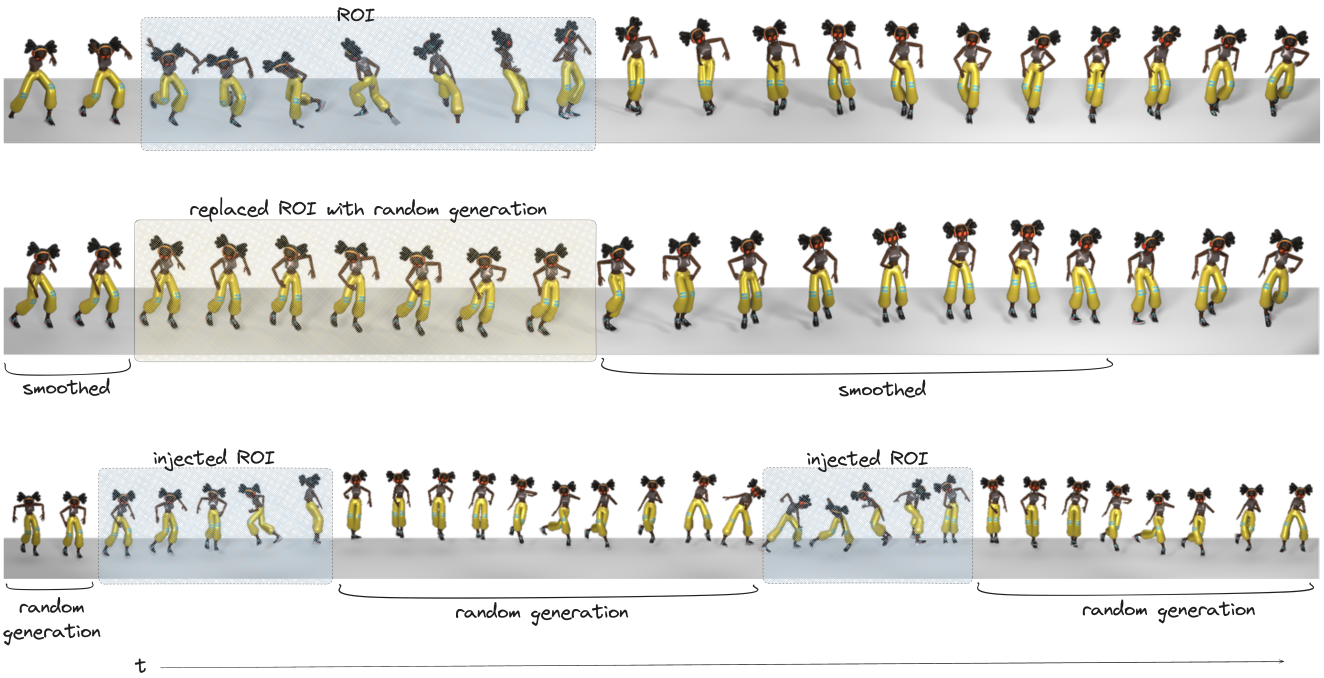

Figure 6. Full-body motion composition example: (top) we select the region of interest (ROI), *i.e.* a clock-wise spin, from the "salsa dancing" sequence of the Mixamo corpus; (middle) a variat of the original sequence is generated with the selected ROI removed and the temporal space being "filled" with motion features generated from noise; (bottom) a new "salsa dancing" variant is composed with 2 spins being placed at user-selected time steps - the non-ROI region are generated by sampled noise. We depict more frames of the composed "salsa" variant (bottom) to demonstrate the 2 spins in the same row.

dataset, while also approaching SinMDM [33]. However, the results showcase that our GAN exhibits a significant improvement in training time compared to both SoTA (almost $\sim\times7$ and $\sim\times2$ respectively), while being extremely faster in inference time compared to the latter ($\sim\times1000$).

4.2. Applications

Apart from the training time performance increase, we introduce new applications that can be performed with a single-shot GAN-based model, such as the body-part and full-body motion composition, in addition to showcasing results on applications introduced by GANimator, like motion re-styling and crowd generation. Note that contrary to [28] we focus solely on applications that do not need re-training, as they pose the main challenge and exploit the superiority of GANs over other approaches in terms of performance.

Body-part motion composition. As detailed in Sec. 3.1, the motion feature $\mathcal{M}_T$ of each input sequence T includes a 6D representation about each skeleton joint. This allow us to define body binary masks M for the upper and the lower body, which can be applied on the motion features during inference and force the masked area to retain its original values, while the rest of the body's movement will be generated based on randomly sampled noise. As depicted in Fig. 5 (top) we use the Mixamo motion "swing dancing" as input sequence T and we choose to keep the lower-body unaltered, while generating alternative - but natural - versions of the upper-body. To achieve that, we use the $G(\cdot)$ trained with this sequence; the hierarchical structure of the model allows us to choose at which level L_* to apply the predefined mask M^{lb} that will keep the lower-body (lb) unaltered. We choose to apply M^{lb} at L_2 as it leads to the smoothest blending of the body parts. Fig. 5 (bottom) demonstrates the application of M^{ub} to the upper-body (ub) of the same motion sample, which preserves its pose despite the generated global rotation and lower body pose.

Full-body motion composition. Assuming a reference motion T of arbitrary length, we consider the following options: a) remove mini-clips of T and use the GAN to "inpaint" them with generated content, and b) select one (or more) mini-clip(s) from T and compose a new motion with the mini-clip(s) placed at the temporal spot(s) of interest. To perform this options, we use a binary mask $M \in \mathbb{B}^{T\times(JQ+C+3)}$ applied to the whole motion feature and not on specific joint-related features. As depicted in Fig. 6, we use M to select the region of interest (ROI) of the "salsa dance" sequence, *i.e.* M values are ones for the frames that correspond to a clock-wise spin. Then, for option (a) we remove the ROI and inpaint the missing part of

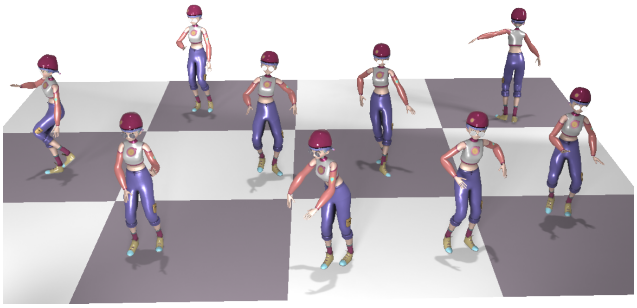

Figure 7. An exampled of crowd generation using the model training on the "dancing" Mixamo benchmark sample.

the motion as:

$$T^{inpaint} = (M \odot \hat{T_2^1}) \oplus (\tilde{M} \odot \downarrow T^{salsa}), \qquad (5)$$

where $\tilde{M}$ is the inverse of binary mask M and $\hat{T_2^1} = G_2^1(G_1^1(z_1), z_2)$ is the generated motion that is "inpainted" in the downsampled ($\downarrow$) salsa sequence. The result of the "inpainting" is presented in the middle of Fig. 6. For option (b) we use the ROI as a standalone mini-clip T^{ROI} which we downsample to the L_2 input level and concatenate a predefined numbers of ROIs with generated motion features from L_1 level. For example, as depicted in Fig. 6(bottom), two T^{ROI} - each representing a spin - are concatenated with $\hat{T_2^1}$ generated by generators trained with the "salsa dance" sample. The two spins are smoothly blended into the generated salsa dance sequence in the desired time steps.

Crowd generation & motion expansion. Single-shot learning enables the generation of motions with common low-frequency features, *i.e.* same motion base, and small variations in the high-level features. This means that by sampling multiple codes from a Gaussian distribution $\mathcal{N}(0, I)$, we can generate a crowd performing similar motions. An example of crowd generation is depicted in Fig. 7. Another straight-forward application is the motion expansion. Since the skeleton-aware convolutions can be applied to a motion feature of arbitrary size, we can concatenate generated features $\hat{T_4^2}$ to the downsampled original motion features T_4^2 at the temporal dimension and use them as input to the corresponding $G_{\{2,\dots,S\}}$.

Re-styling. This has been the most discussed application in single-shot generation as it is relatively trivial for the image domain, however applying style on a motion is challenging. In the image domain, transferring style is the process of applying texture encoded information on image content (*e.g.* applying a style of another artist on a painting as in [10]). In the motion domain, style transfer is realized as applying high-frequency details on low-frequency features that correspond to a certain motion. This means that one cannot apply a dancing style from a stationary (*i.e.* minor translation) motion to a walking one with single-shot generation. However, even with some restrictions, re-styling a motion in real-time is still valuable and leads to interesting results, as the example in Fig. 8. To re-style motion T_x with style from motion T_y, we use the generators G_i^y with $i \in 2, \dots S$, *i.e.* the stages that learn the high-level features (style) of T_y with a downsampled version of T_x that corresponds to the content (T_x^C). Note that the temporal downsampling process operates as low-pass filter, encoding the coarsest features of T_x as its content.

Figure 8. Restyling "house dancing" (top) using a generator trained on a "salsa dancing" Mixamo benchmark sample (bottom). Hands pose and knees proximity are characteristics of the "salsa dancing" style.

5. Conclusion

In this work we investigate the performance of the single-sample generation GANs and address its key challenges. Building on prior work for learning motion generation from a single sample, we propose an loss weight annealing technique for enabling mini-batch training without compromising the adversarial equilibrium. To further minimize the required training iterations we propose a certain cross-stage weight initialization process based on a statistical analysis that exposes correlations between GAN stages. Overall, having similar performance in quality and diversity as anchor, our GAN improves GANimator and SinMDM train time and achieves impressive results in real-time applications without the need for re-training. Next steps include the integration of prior knowledge to further speed up training performance without compromising quality or diversity, as well as the investigation for a more specialized metric that will combine coverage with quality to be able to indicate when we sacrifice tail mini-motions for generated motion smoothness or variation.

References

[1] Kfir Aberman, Peizhuo Li, Dani Lischinski, Olga Sorkine-Hornung, Daniel Cohen-Or, and Baoquan Chen. Skeleton-aware networks for deep motion retargeting. *ACM Trans. Graph. (TOG)*, 39(4):62, 2020. 3

[2] Moab Arar, Ariel Shamir, and Amit H. Bermano. Learned queries for efficient local attention. In *Proc. IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, pages 10841–10852, 2022. 3

[3] Okan Arikan and David A Forsyth. Interactive motion generation from examples. *ACM Trans. Graph. (TOG)*, 21(3): 483–490, 2002. 2

[4] Okan Arikan, David A Forsyth, and James F O'Brien. Motion synthesis from annotations. In *Proc. ACM SIGGRAPH*, pages 402–408. 2003. 2

[5] Nikos Athanasiou, Mathis Petrovich, Michael J Black, and Gül Varol. TEACH: Temporal action composition for 3D humans. In *Proc. Int. Conf. 3D Vis. (3DV)*, pages 414–423, 2022. 2

[6] Samaneh Azadi, Akbar Shah, Thomas Hayes, Devi Parikh, and Sonal Gupta. Make-an-animation: Large-scale text-conditional 3D human motion generation. In *Proc. IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, pages 15039–15048, 2023. 2

[7] Urs Bergmann, Nikolay Jetchev, and Roland Vollgraf. Learning texture manifolds with the periodic spatial GAN. In *Proc. Int. Conf. Mach. Learn. (ICML)*, pages 469–477, 2017. 2

[8] Richard Bowden. Learning statistical models of human motion. In *Proc. IEEE Conf. Comput. Vis. Pattern Recog. Worksh. (CVPRW)*, 2000. 2

[9] Yao Feng, Jing Lin, Sai Kumar Dwivedi, Yu Sun, Priyanka Patel, and Michael J. Black. ChatPose: Chatting about 3D human pose. In *Proc. IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, 2024. 1

[10] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *Proc. IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, pages 2414–2423, 2016. 8

[11] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020. 2

[12] Niv Granot, Ben Feinstein, Assaf Shocher, Shai Bagon, and Michal Irani. Drop the GAN: In defense of patches nearest neighbors as single image generative models. In *Proc. IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, pages 13460–13469, 2022. 2

[13] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of Wasserstein GANs. In *Adv. Neural Inform. Process. Syst.*, 2017. 3

[14] Shir Gur, Sagie Benaim, and Lior Wolf. Hierarchical patch VAE-GAN: Generating diverse videos from a single sample. *Adv. Neural Inform. Process. Syst.*, 33:16761–16772, 2020. 2

[15] Tobias Hinz, Matthew Fisher, Oliver Wang, and Stefan Wermter. Improved techniques for training single-image GANs. In *Proc. IEEE Win. Conf. on App. Comput. Vis. (WACV)*, pages 1300–1309, 2021. 2, 5

[16] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Adv. Neural Inform. Process. Syst.*, page 6840–6851, 2020. 1

[17] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Adv. Neural Inform. Process. Syst.*, 33:6840–6851, 2020. 2

[18] Daniel Holden, Jun Saito, and Taku Komura. A deep learning framework for character motion synthesis and editing. *ACM Trans. Graph. (TOG)*, 35(4):1–11, 2016. 2

[19] Phillip Isola, JunYan Zhu, Tinghui Zhou, and Alexei A. Efros. Learned queries for efficient local attention. In *Proc. IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, pages 1125–1134, 2017. 2

[20] Nikolay Jetchev, Urs Bergmann, and Roland Vollgraf. Texture synthesis with spatial generative adversarial networks. workshop on adversarial training. In *Adv. Neural Inform. Process. Syst. Worksh.*, 2016. 2

[21] Biao Jiang, Xin Chen, Wen Liu, Jingyi Yu, Gang Yu, and Tao Chen. MotionGPT: Human motion as a foreign language. pages 20067–20079, 2023. 1, 2

[22] Jihoon Kim, Jiseob Kim, and Sungjoon Choi. FLAME: Free-form language-based motion synthesis & editing. pages 8255–8263, 2023. 1

[23] Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In *Proc. Int. Conf. Mach. Learn. (ICML)*, pages 3519–3529, 2019. 5

[24] Lucas Kovar, Michael Gleicher, and Frédéric Pighin. Motion graphs. In *Seminal Graphics Papers: Pushing the Boundaries, Volume 2*, pages 723–732. 2023. 2

[25] Vladimir Kulikov, Shahar Yadin, Matan Kleiner, and Tomer Michaeli. SinDDM: A single image denoising diffusion model. In *Proc. Int. Conf. Mach. Learn. (ICML)*, pages 17920–17930, 2023. 3

[26] Chuan Li and Michael Wand. Precomputed real-time texture synthesis with Markovian generative adversarial networks. In *Proc. Eur. Conf. Comput. Vis. (ICCV)*, pages 702–716, 2016. 2

[27] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. BLIP-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *Proc. Int. Conf. Mach. Learn. (ICML)*, pages 19730–19742, 2023. 1

[28] Peizhuo Li, Kfir Aberman, Zihan Zhang, Rana Hanocka, and Olga Sorkine-Hornung. GANimator: Neural motion synthesis from a single sequence. *ACM Trans. Graph. (TOG)*, 41 (4):138, 2022. 2, 3, 4, 5, 6, 7

[29] Yan Li, Tianshu Wang, and Heung-Yeung Shum. Motion texture: A two-level statistical model for character motion synthesis. In *Proc. Annual Conf. on Comp. Graph. and Inter. Tech. (CGIT)*, pages 465–472, 2002. 2

[30] Alexander Quinn Nichol, Prafulla Dhariwal, Aditya Ramesh, Pranav Shyam, Pamela Mishkin, Bob McGrew, Ilya Sutskever, and Mark Chen. GLIDE: Towards photorealistic image generation and editing with text-guided diffusion

models. In *Proc. Int. Conf. Mach. Learn. (ICML)*, pages 16784–16804. 3

[31] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. PyTorch: An imperative style, high-performance deep learning library. *Adv. Neural Inform. Process. Syst.*, 32, 2019. 6

[32] Mathis Petrovich, Michael J Black, and Gül Varol. TEMOS: Generating diverse human motions from textual descriptions. In *Proc. Eur. Conf. Comput. Vis. (ICCV)*, pages 480–497, 2022. 2

[33] Sigal Raab, Inbal Leibovitch, Guy Tevet, Moab Arar, Amit H Bermano, and Daniel Cohen-Or. Single motion diffusion. In *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2024. 2, 3, 5, 6, 7

[34] Tamar Rott Shaham, Tali Dekel, and Tomer Michaeli. SinGAN: Learning a generative model from a single natural image. In *Proc. Int. Conf. Comput. Vis. (ICCV)*, 2019. 2, 5

[35] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, Xi Chen, and Xi Chen. Improved techniques for training GANs. In *Adv. Neural Inform. Process. Syst.*, 2016. 4

[36] Yoni Shafir, Guy Tevet, Roy Kapon, and Amit Haim Bermano. Human motion diffusion as a generative prior. In *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2023. 2

[37] Assaf Shocher, Nadav Cohen, and Michal Irani. "Zero-shot" super-resolution using deep internal learning. In *Proc. IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, pages 3118–3126, 2018. 2

[38] Assaf Shocher, Shai Bagon, Phillip Isola, and Michal Irani. InGAN: Capturing and retargeting the "DNA"" of a natural image. In *Proc. Int. Conf. Comput. Vis. (ICCV)*, pages 4491–4500, 2019. 2

[39] Minjung Son, Jeong Joon Park, Leonidas Guibas, and Gordon Wetzstein. SinGRAF: Learning a 3D generative radiance field for a single scene. In *Proc. IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, pages 8507–8517, 2023. 3

[40] Vadim Sushko, Jurgen Gall, and Anna Khoreva. One-shot GAN: Learning to generate samples from single images and videos. In *Proc. IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, pages 2596–2600, 2021. 2

[41] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proc. IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, pages 1–9, 2015. 5

[42] Guy Tevet, Sigal Raab, Brian Gordon, Yoni Shafir, Daniel Cohen-or, and Amit Haim Bermano. Human motion diffusion model. In *Proc. Int. Conf. Learn. Represent. (ICLR)*. 1, 2

[43] Rundi Wu, Ruoshi Liu, Carl Vondrick, and Changxi Zheng. Sin3DM: Learning a diffusion model from a single 3D textured shape. In *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2023. 3

[44] Jihyeong Yoo and Qifeng Chen. SinIR: Efficient general image manipulation with single image reconstruction. In *Proc. Int. Conf. Mach. Learn. (ICML)*, pages 12040–12050, 2021. 2

[45] Hang Zhang, Xin Li, and Lidong Bing. Video-LLaMa: An instruction-tuned audio-visual language model for video understanding. 2023. 1

[46] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proc. IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, pages 586–595, 2018. 6

[47] ZiCheng Zhang, CongYing Han, and TianDe Guo. ExSinGAN: Learning an explainable generative model from a single image. *arXiv preprint arXiv:2105.07350*, 2021. 2

[48] Zicheng Zhang, Yinglu Liu, Congying Han, Hailin Shi, Tiande Guo, and Bowen Zhou. PetsGAN: Rethinking priors for single image generation. In *Proc. AAAI*, pages 3408–3416, 2022. 2

[49] Haiyan Zhao, Hanjie Chen, Fan Yang, Ninghao Liu, Huiqi Deng, Hengyi Cai, Shuaiqiang Wang, Dawei Yin, and Mengnan Du. Explainability for large language models: A survey. *ACM Trans. Intell. Syst. Technol.*, 15(2), 2024. 1

[50] Yang Zhou, Zhen Zhu, Xiang Bai, Dani Lischinski, Daniel Cohen-Or, and Hui Huang. Non-stationary texture synthesis by adversarial expansion. *arXiv preprint arXiv:1805.04487*, 2018. 2