Xp-GAN: Unsupervised Multi-object Controllable Video Generation

1st Bahman Rouhani  
Faculty of Computer Engineering  
Amirkabi University of Technology  
Tehran, Iran  
brouhani@aut.ac.ir

2nd Mohammad Rahmati  
Faculty of Computer Engineering  
Amirkabi University of Technology  
Tehran, Iran  
rahmati@aut.ac.ir

Abstract—Video Generation is a relatively new and yet popular subject in machine learning due to its vast variety of potential applications and its numerous challenges. Current methods in Video Generation provide the user with little or no control over the exact specification of how the objects in the generated video are to be moved and located at each frame, that is, the user can’t explicitly control how each object in the video should move. In this paper we propose a novel method that allows the user to move any number of objects of a single initial frame just by drawing bounding boxes over those objects and then moving those boxes in the desired path. Our model utilizes two Autoencoders to fully decompose the motion and content information in a video and achieves results comparable to well-known baseline and state of the art methods.

Index Terms—Generative Adversarial Networks, GANs, Autoencoders, AEs, Video Generation, Controllable Video Generation.

I. INTRODUCTION

With the introduction of Generative Adversarial Networks [1], the tasks of image generation and consequently video generation have seen great progress [2] [3] [4] [5] [6].

Despite all these progress however, a very desirable feature is lacking. The current video generation methods offer little to no control over how the generated video is to behave, that is, there are currently few models that allow the user to choose how the elements of a video should move and behave in the generated video. This is a desired feature since it opens a large area of applications for video generation. For example one can use video generation to create high quality animations. Some methods allow the user to implicitly control the final video by conditioning the output video on some input e.g. the initial frame [7], the initial and final frame [8] or a text input [9] or changing the style of a video with another [10] [11]. These methods have some shortcomings. A text input or few frames are too implicit; the user can’t specify every move and detail by text or a small number of frames. Specifying initial and final frame will not guarantee what path the video takes to get from initial frame to the final frame. Style transfer models [10] [12] [13] allow the user to change the style of one video with another, e.g. turn a video to its semantic map or vice versa. However creating a video semantic map video is still a computationally expensive task on its own.

Recently GameGAN [14] was introduced which simulates a game environment, allowing the user to control the agents actions with keyboard. While GameGAN needs labeled data, a newer method called CADDY [15] allows unsupervised playable video generation where the user controls the main object in the scene by pressing keys. However these methods are still not fully controllable. GameGAN can’t be used for general purpose video generation since it requires labeled data. Moreover, both of these methods only allow the user to control one object (game agent or the main moving object in the scene). Finally both GameGAN and CADDY only allow the already observed actions to take place, in other words, the object can only move in a way that the network has seen before. These problems vastly limit the ability of model in generating arbitrary videos.

We propose a model that allows the unsupervised generation of video where user is allowed to control multiple objects and move them in novel ways. The user has control over objects through a schematic motion reference video where each object in the final video is represented by a simple rectangle, and the movement of rectangles dictate how the objects in the final video should move. We specify what the objects are, how they look and what environment they are in by providing a single content reference image. In the first frame of our motion reference video, every rectangle is located on the same pixels as its corresponding object so the network knows which object each rectangle refers to. Similarly in the generated video, the first frame looks like the reference content image, and the objects move according to their corresponding rectangle in the motion reference video. A general schema of the inputs and output can be seen in figure [1].

It should be noted that generating the motion reference video has almost no computational cost, since it only consists of a blank background and a number of rectangles. This input can be easily generated with a few lines of code, or indeed via a user-friendly interface by which user can drag each object in its desired path. Alternatively, the video can be set to be controlled via keyboard as in GameGAN and CADDY.

We utilize two Auto-Encoders to decompose content and motion information and encode them in a latent space. A Generator is then trained to generate videos based on these information. We train this generator both adversarial loss
related works.

Section III describes our method with respect to their references.

Section IV the implementation and outlining the overall structure of our model before going into details of the model. In section V the implementation details are discussed. We report the performance of Xp-GAN and draw conclusions in sections VI and VII respectively.

II. RELATED WORKS

A. Video Generation

Vondrick et al. [6] introduce VGAN which generates videos through two streams, one that generates a background and one that generates a foreground. These two streams are then combined by a mask that applies appropriate weights to foreground and background pixels. While VGAN maps a vector $z$ in the latent space to a background and foreground video, TGAN [5] first uses a temporal generator to map a vector $z$ in latent space to a sequence of vectors in an intermediate space representing a sequence of frames which are then fed to an image generator network to generate the corresponding frames.

This approach still ultimately maps each video to a vector in the initial latent space, therefore Tulyakov et al. [4] introduced MoCoGAN which considers each frame a vector in space and assumes videos are generated by traversing through these vectors with arbitrary speeds. Moreover, MoCoGAN learns to generate a video based on decomposed content and motion vectors. This architecture also introduces two discriminators: a video discriminator which is trained to distinguish between real and fake videos and an image discriminator which focuses on distinguishing between real and fake single frames subsampled from the video. DVDGAN [3] also uses a temporal and a separate spatial discriminator. However to make these networks more scalable, the temporal network is trained on downsampled videos.

Kahembwe et al. [16] argue that the 3D kernels used in video discriminators (namely on those of VGAN and MoCoGAN) results in higher curvature in the loss landscape, rendering the optimization harder and less stable. With this in mind LDVDGANs are introduced which decompose 3D convolutions into 2D convolutions followed by 1D temporal convolution layers. This significantly reduces the parameter count in the model whilst stabilizing the training procedure.

B. Conditional and Constrained Video Generation

A sub-class of video generation is condition or constrained video generation in which the generated video is conditioned on some input other than the noise. This input can be category [4], text [17] [9], or another video [10] [11] [13], or some input images [7] [8].

A popular task in conditional video generation is video translation in which the input is another video and the output is usually the same video in a different domain, style or setting [10] [11] [13]. Cycle-GAN [12] uses two generators to transfer a video from original domain $a$ to a domain $b$. The generator $G_b$ transfers each image $I_a$ from domain $a$ to domain $b$ and generator $G_a$ then translates the same image $G_b(I_a)$ back to the original domain $a$. The cycle consistency loss is the difference between the reconstructed image $G_a(G_b(I_a))$ and the original image $I_a$. Recycle-GAN [10] also uses cycle consistency, however this method additionally uses the temporal information to transfer videos. Each frame $x_{t}^{a}$ in domain $a$ and time $t$ is first translated to another domain in the same fashion as Cycle-GAN, then a prediction network $P_y$ generates the next frame based on this translated frame, and the predicted frame is transferred back to the original domain. The loss function here is the distance between frame $x_{t+1}^{a}$ and its reconstructed $G_a(P_y(G_b(x_{t}^{a})))$. Building on the same idea, MoCycleGAN [13] uses optical flow between two consecutive frames two predict the next frame in the second domain, while also directly warping the separating between two consecutive frames and comparing them.

Other tasks involve different conditional inputs. Yitong et al. [17] use GANs and Variational Autoencoders (VAEs) to generate a video from text. Marwah et al. [9] use a VAE along with two soft attention mechanism (one for short-term and one for long-term) two solve the same problem. Given a single image, ImagineFlow network [7] can generate a video starting with that image. ImagineFlow uses a network to predict and generate a flow for the given image and then uses the final frame to predict the backward flow, leading back to the original image and ensuring bi-directional consistency. Wang et al [8] define and solve the task of point to point video generation.
where given an initial and target frame, the model generates a video starting from the former and ending in latter.

C. Controllable Video Generation

An emerging task in video generation is to explicitly control the movement of object or objects in the video through users instructions. For example controlling how the agent in the video should move by pressing different keys or by moving a corresponding object in the desired path. Note that this is different from video to video translation in that the user has direct and arbitrary control over the movement of objects in the video, while in video translation the user can replace the movement of a video with that of another realistic video which is computationally costly to produce.

GameGAN [14] aims to simulate a game environment using GANs. The architecture consists of a dynamic module responsible for learning the game logic, a rendering module for generating high quality frames and an external memory module for saving the game state. The framework is trained on video-game footage and the keys player pressed between each two frames.

CADDY (Clustering for Action Decomposition and Discovery) [15] solves the problem of controllable video generation in an unsupervised manner. Specifically, an encoder is used to extract frame representation. An action model uses the frame representations to predict the action taking place and a recurrent model uses the frame representations along with the predicted action to predict the next state. Finally the predicted states are fed to a decoder to reconstruct the frame. In this way the network learns the action and state space simultaneously.

III. PROPOSED MODEL

A. Problem Definition

Let us denote a realistic video of length $n$ with $V = \{f_1, ..., f_n\}$ with $f_t$ being the frame at time $t$. We show the distribution of all such realistic videos with $\mathbb{P}_V$. Supposing there are $m$ objects present in the video, we show all their locations in each frame by $L = \{[l^1, ..., l^m], ..., [l^1, ..., l^m]\}_n$ where $l^i$ denotes the location of $i$th object and $\{l^1, ..., l^m\}_t$ denotes $t$th frame. Finally, consider a function $\mathcal{H}(\cdot)$ which given a video, will output the location of its objects (i.e. an object detection and tracking algorithm):

$$\mathcal{H}(V) = \{[l^1, ..., l^m], ..., [l^1, ..., l^m]\}_n. \tag{1}$$

Given an initial frame $f_1$ and a set of location instructions $L$ we would like to generate the video $\tilde{V}$ such that its probability of belonging to $\mathbb{P}_V$ is maximized. We would also like the locations of objects in the generated video, $\mathcal{H}(\tilde{V})$, to match $L$. Thus we define our goal as finding $\tilde{V}$ such that:

$$\max_{\tilde{V}} \mathbb{P}_V(\tilde{V}) \ s.t \ \tilde{V} = \{\tilde{f}_1, ..., \tilde{f}_n\}, \ \tilde{f}_1 \simeq f_1, \ \mathcal{H}(\tilde{V}) \simeq L \tag{2}$$

B. Overall pipeline

We use a pre-trained object tracking model $\mathcal{H}$ to create motion reference videos $V_M$ from regular videos $V$. All the pixels inside bounding boxes where objects where detected are set to 1 (or white) and all other pixels are masked with 0 (black). Some sample frames of motion reference videos are shown at figure 2 below their matching original frames.

**Fig. 2.** Two frames from training video (top), the corresponding frames in the motion reference video (middle) and the product of the original frame and the motion reference frame (bottom). When original frames from $V$ are multiplied by corresponding frames from motion reference video $V_M$, the background is masked but the objects are preserved since their pixels in $V_M$ are set to 1.

We then employ a Variational Autoencoder (VAE) called Content VAE to extract content information from a single image (hereafter referred to as the content reference image) and another VAE called Motion VAE to extract motion information from the a schematic video (called motion reference video). We denote the encoder for our Content and Motion VAEs as $E_c(\cdot)$ and $E_m(\cdot)$ respectively. The two VAEs are trained separately to reconstruct the motion reference video and reference content image and each use their own specific loss functions. We will discuss these details on sections III-C and III-D respectively.

Our Generator consists of two sub-modules: a Decoder $D(\cdot)$, responsible for learning for reconstructing a rough sketch of the final video and a Super Resolution Module $SR(\cdot)$ which helps the enhancement of quality and consistency in the videos. After training the two VAEs, for each video $V$ in dataset we feed its motion reference video $V_M$ to the Motion Encoder and its first frame $f_1$ to the Content Encoder. The extracted motion information $M$ and content information $C$ are then concatenated and fed to the Decoder which is trained to reconstruct the video. We can show these operations as
This can also be explained as the network “denoising” of reconstructing the image so the network fails to exit this point in solution space is far from the optimal point relatively low (and close to optimal) loss. It should be noted a blank white or black image since this will result in a the network converges to the local minima of reconstructing a black or white reconstructed image when one set of the regions are small enough to be treated as noise.

Another challenge arises from the speed of objects in the videos. Specifically, if the objects move too slowly, the network can achieve a low loss by just reconstructing the same frame for the whole video. In this case the network will learn to encode the information pertaining the location of objects but not their movement.

To overcome the first challenge, we generate a pixel-wise weight matrix for each motion reference video, where the foreground and background pixels are weighed such that the overall sum of both sets will be ultimately equal. We first reconstruct the video, then count the number of foreground and background pixels in both original and reconstructed video. We consider a pixel as foreground if its labeled as foreground in either original or reconstructed image since in our dataset background pixels outnumber those of foreground. Consider the motion reference video M and its reconstruction \( \tilde{M} \), a function \( N_{fg}(\cdot) \) which takes as input a video and returns the number of foreground pixels and \( |M| \) as the total number of pixels in \( M \) and denote logical “OR” operation with \( \lor \), we then calculate the weight matrix \( W \) as

\[
W_{bg} = \frac{N_{fg}(M) + N_{fg}(\tilde{M}) + \epsilon}{2 \times |M|},
\]

\[
W_{fg} = \frac{2 \times |M| - N_{fg}(M) + N_{fg}}{2 \times |M|},
\]

\[
W(i, j) = \begin{cases} W_{fg} & \text{if } M_{fg}(i, j) = 1 \\ W_{bg} & \text{if } M_{fg}(i, j) = 0 \end{cases}
\]

To address the second challenge, we calculate the difference between each frame and its previous frame. To be exact, we calculate the ”exclusive or” (XOR) of each frame and its previous frame except for the first frame. This operation finds the pixels that have changed in the new frame. We then increase the weight of these pixels to a factor \( \lambda \). We find that a good value for \( \lambda \) can be twice the calculated weight for foreground (or the maximum of the in general). For the first frame of each video, we do not increase weight of any pixels. We construct a difference weight matrix in this way as stated in equation 9 where \( \oplus \) denotes the XOR operation, \( M^{1:n} \) denotes the frames \( i \) to \( j \) inclusive and \( 0 \) is a zero matrix the same size as a single frame, which we concatenate to \( M_{diff} \) so it has the same dimensions and shape as \( W \) in equation 8.

\[
M_{diff} = M^{1:n-1} \oplus M^{2:n}, \quad W_{diff} = (0, M_{diff} \times \lambda)
\]

The Motion Weighed loss function (denoted with \( L_{MW} \)) for our Motion VAE is then calculated as in equation 10 where \( \odot \) is pixel-wise multiplication and the \( || \) denotes absolute value (not norm).

\[
L_{MW} = E_m(\hat{V}_M), 
\]

\[
C = E_c(f_1),
\]

\[
\tilde{V} = D((L, C)),
\]

\[
\hat{V} = SR(\tilde{V}, z)
\]

The Decoder \( D \) is trained as a decoder on reconstruction loss:

\[
L_D = |D((L, C)) - V|_1
\]

Finally we train the Decoder along with the Super Resolution module in our Generator and the Discriminator similar to Wasserstein-GAN with gradient penalty [13].

C. Motion VAE

The Motion VAE has the task of reconstructing the motion reference video. We use a DCNN (Deep Convolutional Neural Net) with 3D-convolution for this network.

In a general setting, there is no guaranteed bound on the ratio of number of foreground to background pixels which means one can be significantly greater than the other. This presents a challenge to the training of the network. When number of black and white pixels are significantly different, the network converges to the local minima of reconstructing a blank white or black image since this will result in a relatively low (and close to optimal) loss. It should be noted that this point in solution space is far from the optimal point of reconstructing the image so the network fails to exit this state. This can also be explained as the network “denoising” the video. Denoising is a well-known feature of Autoencoders [19] [20] [21] [20]. This feature results in the unwanted blank black or white reconstructed image when one set of the regions are small enough to be treated as noise.
Figure 4 shows a very simple view of this network with two reconstructed samples.

\[ L_{MW} = \text{Mean}(|M - \hat{M}| \odot [W_{bg} + W_{diff}]) \]  \hspace{1cm} (10)

**D. Content VAE**

When generating our final video, we want the location and motion of objects to be exclusively dependent on the motion encoding provided by our Motion VAE. That is to say, when generating videos with fixed motion encoding, any change in the content encoding should not affect the location of the objects. To ensure this, we would like our content encoding to contain as little motion or location information as possible. A naive approach is to reconstruct a single frame to encode the content information since a single frame does not contain any information about how the items in the scene will move. This frame will, however, contain information about where the objects are initially located.

To train our Content VAE to exclusively encode content information, we leverage the motion reference videos. To be specific, we modify the established \( L_1 \) reconstruction loss used for training of Autoencoders so that the network will not be penalized greatly on where the objects are reconstructed. We define Content Weight Loss (denoted with \( L_{CW} \)) as in equation (11) where \( \odot \) is pixel-wise multiplication and \( M_1 \) is the first frame in the motion reference video and the \( || \) denotes the absolute value rather than norm.

\[ L_{CW} = \text{Mean}((\hat{f} - f) \odot M_1) \]  \hspace{1cm} (11)

By multiplying the loss with the corresponding frame in motion reference video, we effectively mask the loss function on the pixels where no object was present. This in turn allows the network to reproduce objects not just in the exact location the object resides, but also on the neighboring regions on which there is no other object. Figure 5 shows two examples of how this loss works. The reconstructed image on itself is chaotic and makes no sense to human eye since the objects have been reproduced with no location restraint. After applying the mask however it is visible that the objects (soccer players in this case) are fully and correctly reconstructed.

In practice, here we use a more delicate type of masking instead of using the motion reference video. This is because in cases where the objects are relatively small and the camera is fixed the VAE tends to only reconstruct the background and not the foreground objects. Masking the background with motion reference video somewhat alleviates this problem. When the objects are small though, the network can still learn to only reconstruct the background since the bounding boxes around objects still contain more of background rather than the objects. We explain how these more exact masks are generated in section [V].

**E. Generator**

Our Generator module consists of two sub-modules, a Decoder which is tasked with generating a rough sketch of the video and a Super Resolution module which will help the Generator to generate better quality videos. This is intended to help stabilize and accelerate the learning process in GANs. Reconstructing a set of videos is an inherently simpler task rather than learning to generate novel and new videos. Therefore we first train our system to simply learn reconstructing some videos. After the Decoder learns to generate the general structure and overall view of video, we use adversarial learning to improve the quality of this images and also allow for generation of new images.

In the first stage of training, only the Decoder network is trained. We feed a single frame to our Content Encoder and a motion reference video to our Motion Encoder, the encoding from these two networks is then concatenated and
fed to the decoder, which is trained to reconstruct the original video. When training the Super Resolution, in contrast to the previous stage, we do not use encodings from Motion and Content Encoder but rather sample these two vectors from a normal distribution. In this way we ensure that our network will learn to generate new videos. Figure 6 shows the pipeline for Generator module.

Fig. 6. The Generator module. Our Decoder is first trained to produce a rough sketch of the video. The whole generator is then trained in an adversarial manner.

IV. IMPLEMENTATION DETAILS

As mentioned in section III-D we use a more exact and delicate masks for the training of our Content VAE. These masks cover almost all of the background and none of the foreground. The reason for using this type of mask, as explained more extensively in section III-D, is that when presented with images of small objects, the VAEs only learn to reconstruct the background of images. In these cases we use this aspect of VAEs to our advantage. We train an ordinary Autoencoder on these images (with the regular $L_1$ loss) which only learns to generate the background. We then use this AE to subtract and remove the acquired background from the images. The original frame $f$ and the background of that frame $f_{bg}$ are subtracted and the pixels where the absolute value of the results is higher than a threshold are set to 1, and the remaining pixels are set to 0. Figure 7 shows this process.

Since we want our network to learn the shape of the objects, we leave out some of the background around the margins of foreground. This is done by applying a Gaussian kernel to the resulting mask to widen the white regions of the masks. For our dataset we use a 10 kernel.

V. EXPERIMENTS

We train our method on Alfheim dataset [22] consisting of footage of soccer games captured on 3 different cameras. We also train RecycleGAN [10], MocoGAN [4] and ldvdGAN [16] on this dataset and measure the Frechet Inception Distance (FID) [23] for all these methods. We train MocoGAN and ldvdGAN as conventional GANs with no conditional input. To train RecycleGAN we consider the original videos of Alfheim dataset as the domain $A$ and their motion reference videos as domain $B$. This network is then asked to apply style transfer to motion reference videos, converting them to realistic videos. We use random vectors sampled from a Normal distribution two generate videos with Xp-GAN rather than with encoding from Motion and Content AE. This is to ensure a fair comparison since Xp-GAN might perform better when the encodings are extracted from real videos.

For each method we produce 15 sets of 300 videos and compute FID for each set. We use these sets to a confidence interval for our results using bootstrapping. These results are shown in table V.

Overall we find that our method is able to generate fully controllable videos from a single initial reference frame and a simple low complexity schematic video while maintaining relative quality to state of the art methods. Our experiments show that the model is able to move the objects in the scene according to the reference motion video as long as the movements are within bounds of natural movement. Figures 8 and 9 show examples of generated videos.

We also test our model on motion reference videos with un-natural movements, that is, movements that differ highly with what the network has seen on dataset e.g. all the soccer players moving exactly in a single unusual direction. We find that our model does slightly drift in these cases, which might be resolved by using larger datasets.
better approach would be to make use of all the information and therefore infer how to generate its corresponding object. A rectangle corresponds to which object based on their location in theory, the model should be able to distinguish which to have a greater capacity for learning and generalizing, since generate videos with multiple objects provided it was modified are of the same type. The same model might be able to objects of different types as at the moment, all moving object and then training the whole network adversarially. we ease the training process by first training and Autoencoder baseline models in quality on the selected datasets. Moreover, our method allows the user to select an arbitrary number of objects and move them in a specified path by placing a controllable setting than the current state of the art methods.

Our method can be improved through adding support for objects of different types as at the moment, all moving object provided by the object detection algorithm and encode the object category into the rectangle instead of the value 1 for all the pixels.

Our proposed method was applied to baseline and simple AE and GAN models which results in average quality and performance due to our limitations in processing power. Another improvement to the current model would be applying the same workflow to more complicated GAN models with higher number of parameters which usually yields better quality and allows for higher resolution.

The model could also benefit from more frame count or ideally generating videos with an arbitrary number of frames.

We believe this model can be built upon and used widely in day to day applications after few improvements. We speculate that developing models that learn to generate videos more specifically and explicitly would be a natural next step for the research in video generation.

VI. CONCLUSIONS AND FUTURE WORK

We introduce a novel method to generate videos in a more controllable setting than the current state of the art methods. Our model allows the user to select an arbitrary number of objects and move them in a specified path by placing a rectangle on them and moving the rectangles in the desired path. Our method is able to produce results comparable to baseline models in quality on the selected datasets. Moreover, we ease the training process by first training and Autoencoder and then training the whole network adversarially.

Our method can be improved through adding support for objects of different types as at the moment, all moving object are of the same type. The same model might be able to generate videos with multiple objects provided it was modified to have a greater capacity for learning and generalizing, since in theory, the model should be able to distinguish which rectangle corresponds to which object based on their location and therefore infer how to generate its corresponding object. A better approach would be to make use of all the information

REFERENCES

[1] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial networks,” arXiv preprint arXiv:1406.2661, 2014.
[2] J. Pan, C. Wang, X. Jia, J. Shao, L. Sheng, J. Yan, and X. Wang, “Video generation from single semantic label map,” in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 3733–3742.
[3] A. Clark, J. Donahue, and K. Simonyan, “Adversarial video generation on complex datasets,” arXiv preprint arXiv:1907.06571, 2019.
[4] S. Tulyakov, M.-Y. Liu, X. Yang, and J. Kautz, “Mocogan: Decomposing motion and content for video generation,” in 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). IEEE Computer Society, 2018, pp. 1526–1535.
[5] M. Saito, E. Matsumoto, and S. Saito, “Temporal generative adversarial nets with singular value clipping,” in 2017 IEEE International Conference on Computer Vision (ICCV). IEEE, 2017, pp. 2849–2858.
[6] C. Vandrick, H. Pirsiavash, and A. Torralba, “Generating videos with scene dynamics,” in Proceedings of the 30th International Conference on Neural Information Processing Systems, 2016, pp. 613–621.
[7] L. Sheng, J. Pan, J. Guo, J. Shao, X. Wang, and C. C. Loy, “Unsupervised bi-directional flow-based video generation from one snapshot,” arXiv preprint arXiv:1903.00913, 2019.
[8] T.-H. Wang, Y.-C. Cheng, C. H. Lin, H.-T. Chen, and M. Sun, “Point-to-point video generation,” in Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 10.491–10.500.
[9] T. Marwah, G. Mittal, and V. N. Balasubramanian, “Attentive semantic video generation using captions,” in Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 1426–1434.
[10] A. Bansal, S. Ma, D. Ramanan, and Y. Sheikh, “Recycle-gan: Unsupervised video retargeting,” in European Conference on Computer Vision. Springer, 2018, pp. 122–138.
[11] C. Chan, S. Ginosar, T. Zhou, and A. A. Efros, “Everybody dance now,” in Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 5933–5942.
[12] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, “Unpaired image-to-image translation using cycle-consistent adversarial networks,” in Proceedings of the IEEE International Conference on Computer Vision (ICCV), Oct 2017.
[13] Y. Chen, Y. Pan, T. Yao, X. Tian, and T. Mei, “Moecycle-gan: Unpaired video-to-video translation,” in Proceedings of the 27th ACM International Conference on Multimedia, 2019, pp. 647–655.
[14] S. W. Kim, Y. Zhou, J. Philion, A. Torralba, and S. Felder, “Learning to simulate dynamic environments with gamegan,” in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 1231–1240.
[15] W. Menapace, S. Lathuilière, S. Tulyakov, A. Siarohin, and E. Ricci, “Playable video generation,” in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 10061–10070.
[16] E. Kahembwe and S. Ramamoorthy, “Lower dimensional kernels for video discriminators,” *Neural Networks*, vol. 132, pp. 506–520, 2020.
[17] Y. Li, M. Min, D. Shen, D. Carlson, and L. Carin, “Video generation from text,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.
[18] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville, “Improved training of wasserstein gans,” in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017, pp. 5769–5779.
[19] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, “Extracting and composing robust features with denoising autoencoders,” in *Proceedings of the 25th international conference on Machine learning*, 2008, pp. 1096–1103.
[20] A. Ashfahani, M. Pratama, E. Lughofer, and Y.-S. Ong, “Devdan: Deep evolving denoising autoencoder,” *Neurocomputing*, vol. 390, pp. 297–314, 2020.
[21] A. Majumdar, “Blind denoising autoencoder,” *IEEE transactions on neural networks and learning systems*, vol. 30, no. 1, pp. 312–317, 2018.
[22] S. A. Pettersen, D. Johansen, H. Johansen, V. Berg-Johansen, V. R. Gaddam, A. Mortensen, R. Langseth, C. Griwodz, H. K. Stensland, and P. Halvorsen, “Soccer video and player position dataset,” in *Proceedings of the 5th ACM Multimedia Systems Conference*, 2014, pp. 18–23.
[23] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, “Gans trained by a two time-scale update rule converge to a local nash equilibrium,” *Advances in neural information processing systems*, vol. 30, 2017.