TFGAN: IMPROVING CONDITIONING FOR TEXT-TO-VIDEO SYNTHESIS

Anonymous authors
Paper under double-blind review

ABSTRACT

Developing conditional generative models for text-to-video synthesis is an extremely challenging yet an important topic of research in machine learning. In this work, we address this problem by introducing Text-Filter conditioning Generative Adversarial Network (TFGAN), a GAN model with novel conditioning scheme that aids improving the text-video associations. With a combination of this conditioning scheme and a deep GAN architecture, TFGAN generates photo-realistic videos from text on very challenging real-world video datasets. In addition, we construct a benchmark synthetic dataset of moving shapes to systematically evaluate our conditioning scheme. Extensive experiments demonstrate that TFGAN significantly outperforms the existing approaches, and can also generate videos of novel categories not seen during training.

1 INTRODUCTION

Generative models have gained much interest in the research community over the last few years as they provide a promise for unsupervised representation learning. Generative Adversarial Networks (GANs) (Goodfellow et al., 2014) have been one of the most successful generative models till date. Following its introduction in 2014, significant progress has been made towards improving the stability, quality and the diversity of the generated images (Salimans et al., 2016; Karras et al., 2017). While GANs have been successful in the image domain, recent efforts have extended it to other modalities such as texts (Wang et al., 2018a), graphs (Wang et al., 2018b), etc.

In this work, we focus on the less studied domain of videos. Generating videos are much harder than images because the additional temporal dimension makes generated data extremely high dimensional, and the generated sequences must be both photo-realistically diverse and temporally consistent. We tackle the problem of text-conditioned video synthesis where the input is a text description and the goal is to synthesize a video corresponding to the input text. This problem has many potential applications, some of which include producing multimedia special effects, generating synthetic data for model-based Reinforcement Learning systems and domain adaptation, etc.

Two recent works that address the problem of text-conditioned video generation include Li et al. (2018) and Pan et al. (2017). Both these methods are variants of conditional GAN model applied to the video data. In spite of some successes, they have the following limitations: (1) They employ 3D transposed convolution layers in the generator network, which constrains them to only produce fixed-length videos. (2) Their models are trained on low-resolution videos - results are shown only at a 64×64 resolution. (3) Text conditioning is performed using a simple concatenation of video and text features in the discriminator: Such a conditioning scheme may perform well on certain datasets, but has difficulty in capturing rich video-text variations.

In this work, we aim to address all the concerns above. First, to model videos of varying length, we use a recurrent neural network in the latent space and employ a shared frame generator network similar to (Tulyakov et al., 2018). Second, we present a model for generating high-resolution videos by using a Resnet-style architecture in the generator and the discriminator network. Third, we propose a new multi-scale text-conditioning scheme based on convolutional filter generation to strengthen the associations between the conditioned text and the generated video. We call our model Text-Filter conditioning GAN (TFGAN). Finally, we construct a benchmark synthetic moving shapes dataset to extensively evaluate the effectiveness of the new conditioning scheme we proposed.

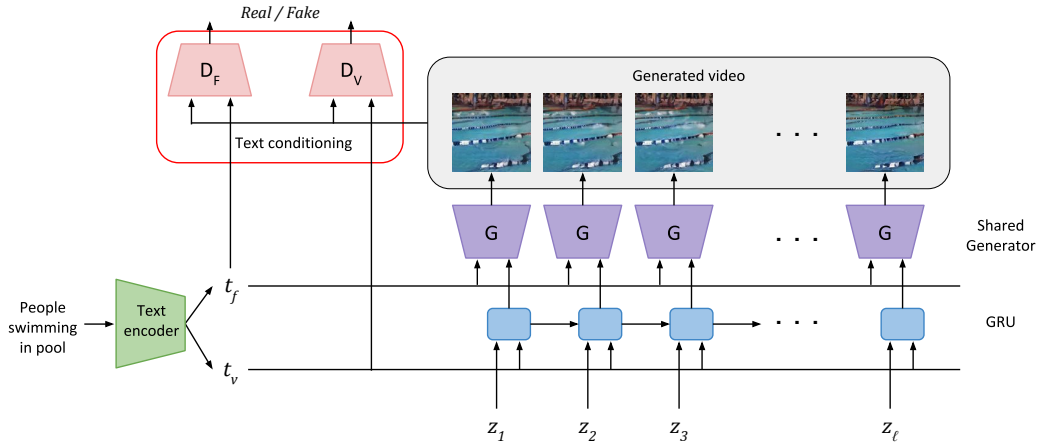


Figure 1: Our TFGAN framework. Text representations are extracted from the input text and passed to a GRU network to get a trajectory in the latent space. These latent vectors are fed to a shared frame generator to produce the video sequence. The generated videos are then passed to conditional discriminator networks. The box highlighted in red is where the conditioning is performed and is expanded in Fig. 2

In summary, our contributions in this work are as follows: (i) A new conditional GAN with an effective multi-scale text-conditioning scheme based on convolutional filter generation is proposed; (ii) A benchmark synthetic dataset for studying text conditioning in video generation is presented; (iii) Photo-realistic video synthesis is achieved using a deeper generator-discriminator architecture.

2 Related work

Two popular approaches to generative modeling include GANs (Goodfellow et al., 2014) and Variational Autoencoders(VAEs) (Kingma & Welling, 2014). GANs are formulated as a 2− player *minimax* game between a generator and a discriminator network, while VAEs are based on variational inference where a variational lower bound of observed data log-likelihood is optimized. Among the two approaches, GANs have generated significant interest as they have been shown to produce images of high sample fidelity and diversity (Karras et al., 2017).

A variant of GAN models are conditional GANs where the generator network is conditioned on input variables of interest. Such a conditioning input can be labels (Odena et al., 2017), attributes (Yan et al., 2016), text (Zhang et al., 2017; Xu et al., 2018) or even images (Zhu et al., 2017). We focus on text conditioning since it is relevant to this work. One of the first works to perform text-conditioned image synthesis is Reed et al. (2016). Their method was only shown to synthesize low-resolution images. To improve the resolution, Zhang et al. (2017) proposed stacking multiple GAN architectures, each producing images of increasing resolution. While the above two methods perform conditioning using the global text representation, Xu et al. (2018) adopts an attention mechanism to focus on fine-grained word-level representations to enable improved conditioning.

While image generation is a well studied problem, there has been very little progress in the domain of video generation. Vondrick et al. (2016) proposed a GAN architecture based on 3D convolutions to generate video sequences, but it can only generate fixed-length videos. Tulyakov et al. (2018) proposed using a recurrent neural network in the latent space to model videos of varying lengths. While these models are not designed to handle conditional video generation, Li et al. (2018) and Pan et al. (2017) perform text-conditioned video synthesis by using the sentence embedding as a conditional input. However, both of these conditional generative models are based on 3D convolutions, they can only produce fixed-length low-resolution videos. In this work, we address this issue by developing an architecture capable of producing high-resolution videos of varying length.

3 METHOD

We first provide a formal description of the problem being address. We are given access to n data points $\{(\mathbf{v}_i, \mathbf{t}_i)\}_{i=1}^{n}$ sampled from an underlying joint distribution $p(\mathbf{v}, \mathbf{t})$ in the video-sentence space. Here, each $\mathbf{v}_i \in \mathbb{R}^{T \times C \times W \times H}$ is a video clip and $\mathbf{t}_i$ is a sentence description. We are interested in learning a model capable of sampling from the unknown conditional distribution $p(\mathbf{v}|\mathbf{t})$. Similar to conditional GANs, we formulate the problem as learning a transformation function $G(\mathbf{z}, \mathbf{t})$ from a known prior distribution $P_{\mathbf{z}}(\mathbf{z})$ and the conditional input variable $\mathbf{t}$ to the unknown conditional distribution $p(\mathbf{v}|\mathbf{t})$. The function G is optimized using an adversarial training procedure.

3.1 MODEL FRAMEWORK

The framework of our proposed model is shown in Fig. 1. The text description $\mathbf{t}$ is passed to a text encoder T to get a frame-level representation $\mathbf{t}_f$ and a video-level representation $\mathbf{t}_v$. Here, $\mathbf{t}_f$ is a representation common to all frames, and contains frame-level information like background, objects, etc. from the text. The video representation $\mathbf{t}_v$ extracts the temporal information such as actions, object motion, etc. The text representation along with a sequence of noise vectors $\{\mathbf{z}_i\}_{i=1}^{l}$ are passed to a recurrent neural network to produce a trajectory in the latent space. Here, l denotes the number of frames in the video sequence. These sequence of latent vectors are then passed to a shared frame generator model G to produce the video sequence.

The generated video is then fed to two discriminator models - D_F and D_V. D_F is a frame-level discriminator that classifies if the individual frames in the video are real/fake, whereas the video discriminator D_F is trained to classify the entire video as real/fake. The discriminator models D_F and D_V also take the text encoding $\mathbf{t}_f$ and $\mathbf{t}_v$ respectively as inputs so as to enforce text-conditioning.

3.2 TEXT-FILTER CONDITIONING

To build strong conditional models, it becomes important to learn good video-text associations in the GAN model. A standard technique is to sample negative $(\mathbf{v}, \mathbf{t})$ pairs (wrong associations) and train it as fake class, while the correct $(\mathbf{v}, \mathbf{t})$ pairs are trained as real class in the discriminator network. Since the generator is updated using the gradients from the discriminator network, it becomes important to effectively fuse the video and text representations in the discriminator so as to make the generator condition well on the text. Previous methods (Li et al., 2018; Pan et al., 2017) use a simple concatenation of text and video features as the feature fusion strategy. We found that this simple strategy produces poor conditioned models in datasets where there are rich text-video variations (refer to Section. 4 for more details).

Our proposed model Text-Filter conditioning GAN (TFGAN) focuses on improving text conditioning. In TFGAN, we employ a scheme based on generating convolutional filters from the text features. This scheme, which we call Text-Filter conditioning, is shown in Fig. 2. Let us first divide the discriminator network D (which can be D_F or D_V) into multiple sub-networks $\{D^{(i)}\}_{i=1}^{m}$ so that $D(\mathbf{x}) = D^{(m)} \circ D^{(m-1)} \circ \dots D^{(1)}(\mathbf{x})$. These sub-networks can be as small as a single layer, or can be a cascade of multiple layers. Let $\mathbf{d}^{(i)}$ denote the output of the i^{th} sub-network of the discriminator. From the text features, we generate a set of convolution filters $\{\mathbf{f}_i\}_{i=1}^{m}$. Each filter $\mathbf{f}_i$ is now convolved with the discriminator response $\mathbf{d}^{(i)}$, and this convolved output is passed through additional convolutions after which they are pooled to get a single video-text representation. We use this pooled feature vector to classify the $(\mathbf{v}, \mathbf{t})$ pairs as real or fake. Because the generated convolutional filters $\{\mathbf{f}_i\}$ are applied to discriminator sub-network outputs $\{\mathbf{d}^{(i)}\}$ from different layers, the resulting text conditioning effectively imposes semantic constraints extracted from input texts to the generated individual frames and video clips at different feature abstraction levels.

3.3 TRAINING ALGORITHM

The discriminator model D and the generator model G are trained using an adversarial game as done in the standard conditional GANs. However, since we employ deep Resnet-style architectures for our $G - D$ networks, it was important to stabilize the GAN training. We use the regularizer as proposed in Mescheder et al. (2018) where the norm of the discriminator gradients are penalized. With this regularizer, our optimization objective can be expressed as

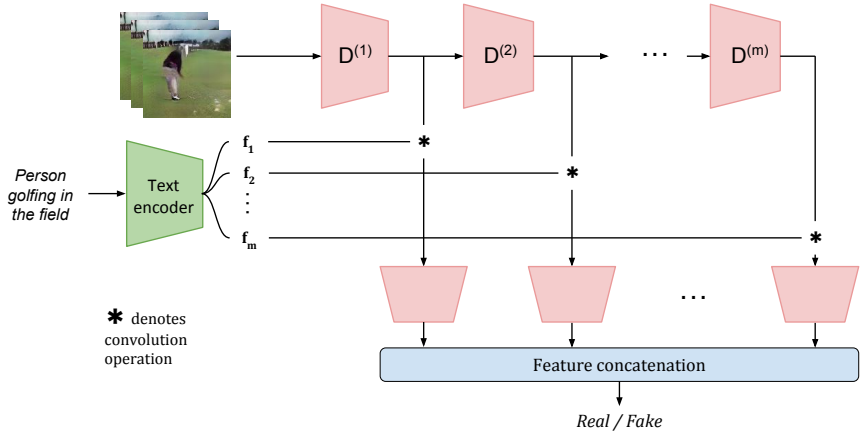


Figure 2: Illustration of our Text-Filter conditioning strategy. From the text features, we extract a set of convolution filters $\{\mathbf{f}_i\}_{i=1}^{m}$ and apply the filter $\mathbf{f}_i$ to the output $\mathbf{d}^{(i)}$ of sub-network $\mathbf{D}^{(i)}$. These responses are then passed to additional convolutional layers before they are pooled to get a single video-text representation that is used to classify the input $(\mathbf{v}, \mathbf{t})$ pair as real/fake.

$$L_{real} = \mathbb{E}_{(\mathbf{v},\mathbf{t})\sim p_{data,real}}[\log(D(\mathbf{v}, T(\mathbf{t}))) + \frac{\gamma}{2}\|\nabla D(\mathbf{v}, \mathbf{t})\|^2] \tag{1}$$

$$L_{fake} = \frac{1}{2}[\mathbb{E}_{(\mathbf{v},\mathbf{t})\sim p_{data,fake}} \log(1 - D(\mathbf{v}, T(\mathbf{t}))) + \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}} \log(1 - D(G(\mathbf{z}, T(\mathbf{t})), T(\mathbf{t})))] \tag{2}$$

$$\min_G \max_D \quad L_{real} + L_{fake} \tag{3}$$

The text encoder T is optimized as follows

$$\max_T L_T = \mathbb{E}_{(\mathbf{v},\mathbf{t})\sim p_{data,real}} \log(D(\mathbf{v}, T(\mathbf{t}))) + \mathbb{E}_{(\mathbf{v},\mathbf{t})\sim p_{data,fake}} \log(1 - D(\mathbf{v}, T(\mathbf{t}))) \tag{4}$$

In the above set of equations, $p_{data,real}$ denotes the real data distribution with correct video-text correspondences, whereas $p_{data,fake}$ refers to the distribution with incorrect video-text correspondences. Note that we have two discriminator networks - D_F, D_V in our models, and the above equations have to be repeated for both models. Eq.1-4 are optimized by alternating between the minimization and maximization problems as done in standard GAN. Please refer to Appendix A for the detailed training algorithm.

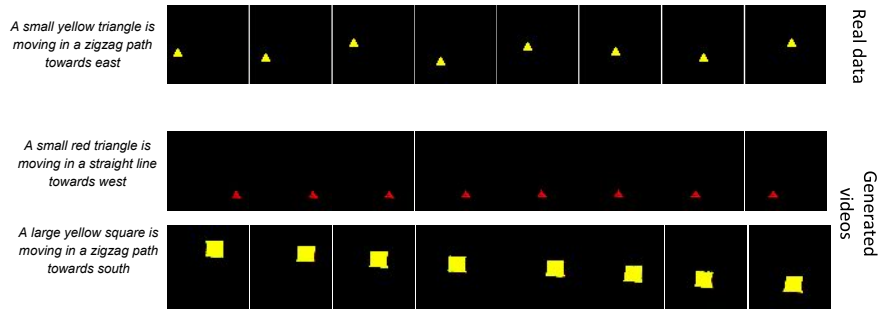
4 Experiments

This section discusses the experimental validation of our TFGAN model. We first describe a benchmark synthetic dataset we created for the task of text-to-video generation, and use it to better analyze our system. Then, we show results on a challenging real-world video dataset - the Kinetics human action video dataset (Kay et al., 2017). Finally, we show how our method can be extended to the task of text-to-image synthesis and show results on the CUB birds dataset (Welinder et al., 2010).

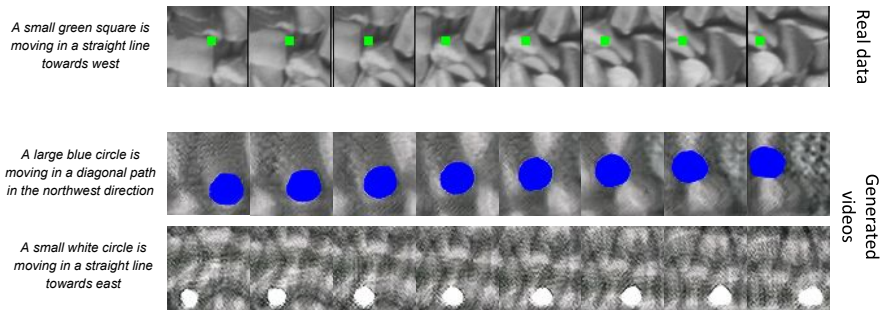
4.1 Moving Shapes Dataset

4.1.1 Dataset creation

To better understand the task of text-to-video synthesis, we created a dataset of moving shapes where a shape moves along a trajectory as described by the corresponding text description. This synthetic dataset has 5 control parameters: shape type, size, color, motion type and motion direction. Of these, the first three parameters are frame-level attributes while the last two are temporal attributes. The set of possible values each parameter can take is listed in Appendix B. The combination of



(a) Shapes-v1 dataset



(b) Shapes-v2 dataset

Figure 3: Samples from the Moving Shapes Dataset: In each set of images, the top row corresponds to the original video and the bottom two rows correspond to the video generated by our TFGAN model (Better viewed in color).

all parameters results in 360 unique parameter configurations. Some samples from this dataset are shown in Fig. 3a. We call this dataset *Shapes-v1 dataset*.

The above dataset contains videos with static background (all black). While this is a reasonable assumption to make, it is hardly true in practice as many videos have dynamic backgrounds. So, we create a second dataset called *Shapes-v2 dataset* which is a version of Moving Shapes dataset with dynamic backgrounds. To generate the background, we choose images from the Kylberg Texture Dataset (Kylberg, 2011) and sample a sequence of patches corresponding to a random trajectory. Each patch in this sequence forms the background of an individual frame in the video. These background textures are blended with the moving object resulting in videos as shown in Fig. 3b. This dataset is much more challenging than the *Shapes-v1 dataset* as the generative models should learn to ground the text description to the moving object but not to the randomly moving background. Both these datasets were created with videos containing 16 frames at a 64×64 frame resolution.

Table 1: Attribute classification accuracy (in %) on *Shapes* dataset

Method	Shape	Color	Size	Motion	Direction
Shapes-v1 dataset					
Feature concat	70.18	99.23	83.69	96.83	99.00
Feature concat multiscale D	65.25	99.91	74.31	99.12	99.11
Text-filter conditioning	**97.66**	**99.99**	**98.60**	**99.40**	**100.00**
Shapes-v2 dataset					
Feature concat	64.49	99.98	81.46	97.40	99.40
Feature concat multiscale D	56.12	99.71	65.15	97.20	98.00
Text-filter conditioning	**88.52**	**99.98**	**94.46**	**100.00**	**99.80**

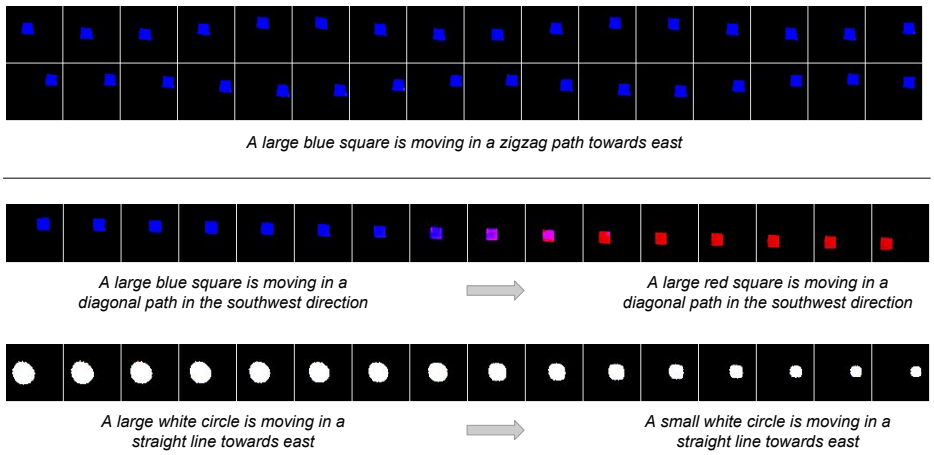


Figure 4: Exploratory experiments on *Shapes-v1* dataset. The image on the top shows the long sequence experiment where we generate 32-length sequence from a model trained for 16 frames. The top row of this video are the first 16 frames and the bottom row corresponds to the next 16. The images on the bottom illustrate the interpolation experiments where we generate a video corresponding to a smooth transition between two input sentences.

4.1.2 QUANTITATIVE EVALUATION

An important advantage of creating the synthetic dataset is that it provides a framework for quantitative evaluation of the text-conditioning. First, we train five attribute classifiers (shape, size, color, motion and direction classifiers) on the real data using the ground truth attributes (we have access to ground-truth attributes as we stored them while creating the dataset). We then use these trained attribute classifiers to verify if the attributes of the generated videos correspond to those described by the input text in the test set. For each text description in the test set, we generate the video using our TFGAN model and measure the attribute prediction accuracy. Higher this accuracy, better conditioned is our GAN model.

We experiment with the following models: (1) FeatCat: a conditional GAN model trained using simple text-video feature concatenation in the discriminator network (2) FeatCat branchingD: conditional GAN model with branching D structure where responses at intermediate layers of D are pooled, and this pooled feature is concatenated with text embedding. This model is essentially TFGAN but without performing the convolutions from text-filters, and (3) TFGAN with Text-Filter conditioning. The architecture and hyper-parameter details are described in the Appendix C. Some sample generations of our Text-Filter conditioned GAN model is shown in Fig. 3a and 3b

Table 1 reports the quantitative evaluation of the three conditional GAN models on *Shapes* dataset. We observe that TFGAN with text-filter conditioning achieves the best performance among the three models on both the datasets. An important observation to note is that using a branching architecture in the discriminator network alone does not improve the text conditioning. This shows that the effectiveness of our method comes not from the branching architecture, but in how text conditioning is applied (using convolutions) at multiple layers of the discriminator network.

4.1.3 EXPLORATORY EXPERIMENTS

In this section, we report some exploratory experiments we perform on the Shapes dataset.

Sentence interpolation In this experiment, we depict conditional interpolation whereby frames in a video transition corresponding to the interpolation between two sentence descriptions. Let S_1 and S_2 denote the two sentences that are interpolated, and $(\mathbf{t}_f^{S_1}, \mathbf{t}_v^{S_1})$ and $(\mathbf{t}_f^{S_2}, \mathbf{t}_v^{S_2})$ denote their corresponding feature representation obtained by passing through the text encoder T. For each frame to be generated, the corresponding conditioning feature is obtained by a linear interpolation between these two representations:

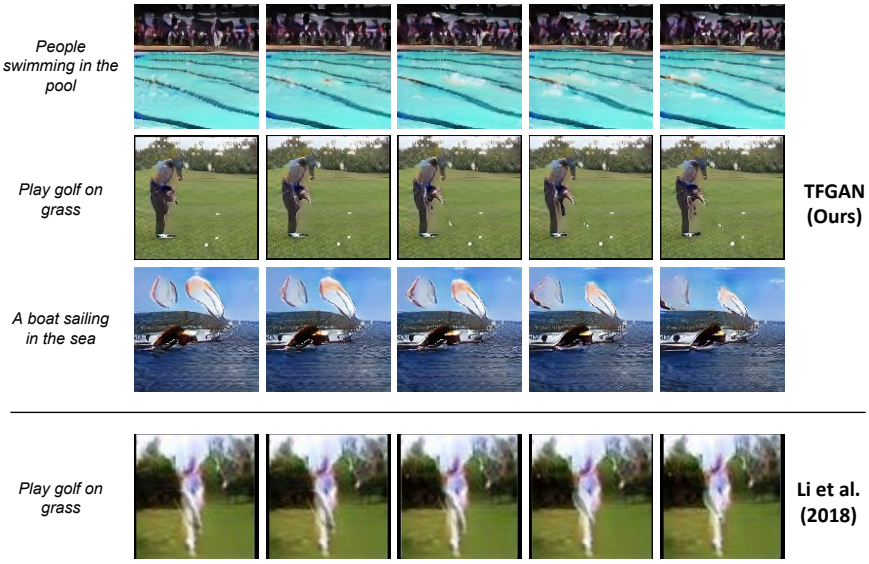


Figure 5: Sample generations from models trained on Kinetics dataset

$$(t_f^i, t_v^i) = (1-\alpha)(t_f^{S1}, t_v^{S1}) + \alpha(t_f^{S2}, t_v^{S2})$$

Instead of using a fixed text representation (t_f, t_v) as conditioning argument to all the frames in the Generator network, we use (t_f^i, t_v^i) as input to the frame i. The resulting interpolated videos are shown in Fig. 4. We observe that we are able to obtain smooth transitions. When interpolating between the blue square and the red square, we obtain some intermediate frames with pink shade. Interestingly, none of the samples in the dataset contain pink color. In the second figure, we observe a smooth decrease in the object size while the object continues to move in the specified trajectory.

Generating novel categories To characterize if the model has learned to generalize and not naively memorize the dataset, this experiment aims to study the ability of our TFGAN model to produce videos not seen during training. Of the 360 unique parameter configurations in the Shapes dataset, we randomly hold out n configurations from the training set. After training the model on this training set, we feed the text descriptions from the held-out n configurations and measure the attribute classification accuracy in this set. In this experiment, we choose $n = 20$. The results are reported in Table 2. We observe that our model achieves good accuracy and this illustrates the ability of our method to generalize.

Table 2: Attribute classification acc. on novel categories (*Shapes-v1*)

Attribute	Accuracy (in %)
Shape	96.21
Color	99.78
Size	98.77
Motion	96.23
Direction	99.42

Long Sequence Generation One of the benefits of using a RNN-based GAN model is that it allows us to model variable-length video sequences. To demonstrate this, we perform an experiment where we train our TFGAN model on 16-length video sequences and generate 32-length sequences. This can be performed easily as we could potentially generate a latent trajectory of any length using the RNN model in the latent space, and the videos are generated using a shared generated acting on this latent trajectory. Fig. 4 shows the output of one such 32-length sequence generated. We observe that the model is able to clearly perform the zig-zag motion beyond 16 frames.

4.2 KINETICS DATASET

To demonstrate the practical relevance of our approach, we perform experiments on real-world video datasets. We use the dataset proposed in Li et al. (2018) for this purpose. This dataset

Figure 6: Sample generations of Text2img synthesis from our model trained on CUB-birds dataset

contains videos of human actions, and was curated from YouTube and Kinetics human action video dataset Kay et al. (2017). The dataset contains the following action classes - biking in snow, playing hockey, jogging, playing soccer ball, playing football, kite surfing, playing golf, swimming, sailing and water skiing. This is an extremely challenging dataset for the task of video generation due to the following reasons: (1) videos are extremely diverse, and there are a lot of variations within the video, and (2) some videos have low-resolution and poor-quality video frames. Some sample videos from the dataset are shown in Fig. 5

The results of training our TFGAN model on the Kinetics dataset are shown in Fig. 5. We observe that our model is able to produce videos of much higher quality than the comparison method (Li et al., 2018). We are able to generate fine motions like golf swing, while Li et al. (2018) produces a blobby region. Also, we train the model to produce 128×128 resolution videos, while the method in Li et al. (2018) was trained only on 64×64 videos. As done in Li et al. (2018), we report a simplified version of inception score whereby a video classification model is trained on the real data, and the accuracy on generated data is reported. We report the performance on the following five categories as done in Li et al. (2018): kite surfing, playing golf, biking in snow, sailing, swimming and water skiing. As can be seen from Table. 3, our methods achieves significantly higher accuracy than the method in Li et al. (2018). In-set refers to the performance obtained on the test set of real videos, thus serves as an upper bound. We report additional results, architecture and hyper-parameter details in Appendix C.

Table 3: Classification accuracy

Method	Acc. (%)
In-set	78.1
T2V(Li et al. (2018))	42.6
Ours	53.2

4.3 Text to Image Generation

Text-to-image generation is a relatively easier problem than text-to-video generation due to the absence of temporal constraints. Although the focus of this paper is on text-to-video synthesis, our framework is flexible and can be trivially extended to the problem of text-to-image synthesis. This can be accomplished by removing the video-level discriminator D_V and the RNN network in the latent space. We train our GAN model with Text-Filter conditioning on the CUB-Birds dataset Welinder et al. (2010), a benchmark dataset for text-to-image generation. Some of the samples from the generated images are shown in Fig. 6. We observe that our model is able to produce photo-realistic images. We also report Inception score as a quantitative metric. As can be seen from Table. 4, our method achieves higher inception scores than the comparison methods.

Table 4: Inception Score on CUB-Birds dataset

Method	Inception Score
StackGAN	3.7 ± 0.04
StackGAN v-2	3.82 ± 0.06
Ours	4.12 ± 0.18

5 Conclusion

In this work, we address the problem of generating videos conditioned on text. We propose a novel text-conditioning framework whereby conditioning is performed using convolution operations on image feature maps with filters generated from text. To better understand the text conditioning, we construct a synthetic dataset and show that our conditioning scheme achieves superior performance compared to other techniques. Finally, by using deeper architectures in the discriminator and generator networks, we generate photo-realistic videos on the challenging Kinetics dataset.

REFERENCES

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger (eds.), *Advances in Neural Information Processing Systems 27*, pp. 2672–2680. Curran Associates, Inc., 2014. URL http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pp. 770–778. IEEE Computer Society, 2016.

Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *CoRR*, abs/1710.10196, 2017. URL http://arxiv.org/abs/1710.10196.

Will Kay, João Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, Mustafa Suleyman, and Andrew Zisserman. The kinetics human action video dataset. *CoRR*, abs/1705.06950, 2017.

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

Gustaf Kylberg. The kylberg texture dataset v. 1.0. External report (Blue series) 35, Centre for Image Analysis, Swedish University of Agricultural Sciences and Uppsala University, Uppsala, Sweden, September 2011. URL http://www.cb.uu.se/~gustaf/texture/.

Yitong Li, Martin Renqiang Min, Dinghan Shen, David E. Carlson, and Lawrence Carin. Video generation from text. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, Louisiana, USA, February 2-7, 2018*, 2018.

Lars Mescheder, Sebastian Nowozin, and Andreas Geiger. Which training methods for gans do actually converge? In *International Conference on Machine Learning (ICML)*, 2018.

Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional image synthesis with auxiliary classifier GANs. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 2642–2651, 2017.

Yingwei Pan, Zhaofan Qiu, Ting Yao, Houqiang Li, and Tao Mei. To create what you tell: Generating videos from captions. October 2017. URL https://www.microsoft.com/en-us/research/publication/create-tell-generating-videos-captions/.

Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. Generative adversarial text to image synthesis. In *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pp. 1060–1069, New York, New York, USA, 20–22 Jun 2016. PMLR.

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, Xi Chen, and Xi Chen. Improved techniques for training gans. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 29*, pp. 2234–2242. Curran Associates, Inc., 2016. URL http://papers.nips.cc/paper/6125-improved-techniques-for-training-gans.pdf.

Sergey Tulyakov, Ming-Yu Liu, Xiaodong Yang, and Jan Kautz. Mocogan: Decomposing motion and content for video generation. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. Generating videos with scene dynamics. In *Advances in Neural Information Processing Systems 29*, pp. 613–621. Curran Associates, Inc., 2016. URL http://papers.nips.cc/paper/6194-generating-videos-with-scene-dynamics.pdf.

Heng Wang, Zengchang Qin, and Tao Wan. Text generation based on generative adversarial nets with latent variables. In *Advances in Knowledge Discovery and Data Mining - 22nd Pacific-Asia Conference, PAKDD 2018, Melbourne, VIC, Australia, June 3-6, 2018, Proceedings, Part II*, 2018a.

Hongwei Wang, Jia Wang, Jialin Wang, Miao Zhao, Weinan Zhang, Fuzheng Zhang, Xing Xie, and Minyi Guo. Graphgan: Graph representation learning with generative adversarial nets. In *AAAI*. AAAI Press, 2018b.

P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona. Caltech-UCSD Birds 200. Technical Report CNS-TR-2010-001, California Institute of Technology, 2010.

Tao Xu, Pengchuan Zhang, Qiuyuan Huang, Han Zhang, Zhe Gan, Xiaolei Huang, and Xiaodong He. Attngan: Fine-grained text to image generation with attentional generative adversarial networks. 2018.

Xinchen Yan, Jimei Yang, Kihyuk Sohn, and Honglak Lee. Attribute2image: Conditional image generation from visual attributes. In *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part IV*, 2016.

Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, and Dimitris Metaxas. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. In *ICCV*, 2017.

Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, 2017.

APPENDIX A TRAINING ALGORITHM

Section. 3 discusses the various loss terms we use in our TFGAN model. Equation 1 gives the expressions for the loss terms L_{real}, L_{fake} and L_T. Denoting $L^F_{real}, L^F_{fake}, L^F_T$ as the losses for frame-level discriminator D_F and $L^V_{real}, L^V_{fake}, L^V_T$ as the losses for video-level discriminator D_V, the training algorithm we use is mentioned in Alg. 1

Algorithm 1 Training algorithm

Require: θ: Parameters of G, ϕ_F: Parameters of D_F, ϕ_V: Parameters of D_V, ϕ_T: Parameters of T
Require: N_{iter}: number of training iterations
Require: α: Learning rate, N_b: batch size
for t in $1 : N_{iter}$ **do**
  Sample N_b real samples with correct video-text correspondence $\{(\mathbf{v}^r_i, \mathbf{t}^r_i)\}_{i=1}^{N_b}$
  Sample N_b real samples with incorrect video-text correspondence $\{(\mathbf{v}^f_i, \mathbf{t}^f_i)\}_{i=1}^{N_b}$
  Update D_F: $\phi_F = \phi_F + \alpha\nabla[L^F_{real} + L^F_{fake}]$
  Update D_V: $\phi_V = \phi_V + \alpha\nabla[L^V_{real} + L^V_{fake}]$
  Update G: $\theta = \theta - \alpha\nabla[L^F_{real} + L^F_{fake} + L^V_{real} + L^V_{fake}]$
  Update T: $\phi_T = \phi_T + \alpha\nabla[L^F_T + L^F_T + L^V_T + L^V_T]$
end for

APPENDIX B DATASET GENERATION - SHAPES DATASET

Both *Shapes-v1* and *Shapes-v2* dataset contain videos with objects moving along a specific trajectory. There are 5 control parameters - shape, size and color of object, type and direction of motion. Table 5 lists the possible values each parameter can take. The (shape, color, size) tuple describes the structure of the object, while (motion type, direction) tuple dictates how the object moves in the video. For straight line and zig-zag motion, the object could move in north, south, west and east direction, while for diagonal motion, the possible directions include north-west, north-east, south-west and south-east. The zig-zag motion was generated using a sinusoidal function.

APPENDIX C ARCHITECTURE AND HYPER-PARAMETERS

We first define some basic architecture blocks:

- ResnetBlock(x, y): x + Conv2D($x \rightarrow y$, kernel=3, str=2, pad=1) if $x == y$, else Shortcut($x \rightarrow y$) + Conv(x$-$ >y, 3×3, str=2, pad=1). Here, Shortcut($x \rightarrow y$) is the 1×1 convolution that is maps from x filters to y filters. This is a simplified resnet block that was proposed in He et al. (2016).
- ResnetBlock3D(x, y): x + Conv3D($x \rightarrow y$, kernel=3, str=2, pad=1) if $x == y$, else Shortcut($x \rightarrow y$) + Conv3D(x$-$ >y, 3×3, str=2, pad=1)
- Text-Img feat concat: Can be any feature concatenation technique. We use Text-Filt conditioning as discussed in Sec. 3

C.1 TEXT-FILTER CONDITIONING

In all our experiments, we used the following architecture for Text-Filter conditioning. The discriminator (both D_V and D_F) were divided into three sub-networks ($m = 3$). In Moving Shapes experiments, the sub-network $D^{(0)}$ is the network till Resnet-1 block, $D^{(1)}$ is the sub-network from Resnet-2 block till the end of Renset-3 block and $D^{(3)}$ forms the rest of the network. For the Kinetics experiment, we the sub-network $D^{(0)}$ is the network till Renset-11 block, $D^{(1)}$ spans from Renset-20 block to Resnet-31 block, and $D^{(3)}$ forms the rest of the network.

We first take the output of $D^{(0)}$ and apply $1D$ convolution to bring the number of filters to 8. So, if the output of $D^{(0)}$ was a $n \times k \times w$ map, this transformation will bring it down to $8 \times k \times w$. Let the text embedding (which is obtained by passing through the text encoder) be a $d-$ dimensional vector. We first apply a $FC(d \rightarrow 5.5.8.8)$ to this embedding and reshape it to $8 \times 8 \times 5 \times 5$ filter. This filter is then convolved with the transfomed outputs of D^0. These resulting convolved feature maps are passed through two conv-2D layers with Avg-Pool and then reshaped to 128 dimensional vector. The same procedure is done for $D^{(1)}$. For $D^{(2)}$, we just take the output and pass it through a fully connected layer to get 128 dimensional vector. These three 128 dimensional vectors are concatenated to get one resulting vector which classifies if the input is real or fake. The exact sample procedure is repeated for Video discriminators, only difference being the size of Text-Filters: 3D filters of size $3 \times 5 \times 5$ is used instead of 5×5 filter . So, the FC applied on text embedding will be $FC(d \rightarrow 5.5.3.8.8)$

C.2 SHAPES DATASET

For *Shapes-v1* and *Shapes-v2* datasets, we used an architecture based on ResnetBlock as shown in Tables. 6, 7 and 8. For the text encoder, we first obtained the GloVE embeddings of individual words, then applied a 1D-CNN based network with the following network architecture:

Conv1D(512, kernel=3) $\rightarrow$ ReLU $\rightarrow$ MaxPool(2) $\rightarrow$ Conv1D(512, kernel=3) $\rightarrow$ ReLU $\rightarrow$ MaxPool(2) $\rightarrow$ Conv1D(256)

1D-CNN was sufficient in this case as most sentences were of rougly similar lenghts. The inputs were zero padded to making every sentence have the dimension. We tried using a LSTM model and it gave similar performance as 1D CNN.

Table 5: Simulation parameters

Atribute	Set of values
Shape	{ Square, Circle, Triangle }
Color	{ Red, Blue, Green, White, Yellow }
Size	{ Small, Large }
Motion type	{ Straight Line, Diagonal, Zig-zag }
Direction	{ North, South, East, West } for st.line and zig-zag motion { North-east, North-west, South-east, South-west } for diagonal motion

C.3 KINETICS DATASET

The architectures used in Kinetics dataset are shown in Tables. 9, 11 and 10. For the text encoder, we used the same architecture as that of Shapes dataset.

APPENDIX D ADDITIONAL RESULTS

D.1 KINETICS DATASET

Some additional results on the Kinetics dataset are shown in Fig. 7. We observe that we are able to generate videos of high quality. To illustrate the variations that occur within a class, we generate multiple videos of the same text description. Figure. 8 shows one such example for swimming class. We find that out model is capable of generating diverse predictions.

Table 6: Generator architecture - Shapes dataset

Layer	Output size	Filter
Input	768	-
Fully connected	1024	$768 \rightarrow 1024$
Reshape	$256 \times 2 \times 2$	-
ResnetBlock-0	$256 \times 2 \times 2$	$256 \rightarrow 256$
Upsample	$256 \times 4 \times 4$	-
ResnetBlock-1	$128 \times 4 \times 4$	$256 \rightarrow 128$
Upsample	$128 \times 8 \times 8$	-
ResnetBlock-2	$128 \times 8 \times 8$	$128 \rightarrow 128$
Upsample	$128 \times 16 \times 16$	-
ResnetBlock-3	$128 \times 16 \times 16$	$128 \rightarrow 128$
Upsample	$128 \times 32 \times 32$	-
ResnetBlock-4	$64 \times 32 \times 32$	$128 \rightarrow 64$
Upsample	$64 \times 64 \times 64$	-
ResnetBlock-5	$32 \times 64 \times 64$	$64 \rightarrow 32$
Conv2D	$3 \times 64 \times 64$	$32 \rightarrow 3$
Tanh	$3 \times 64 \times 64$	-

Table 7: Frame Discriminator architecture - Shapes dataset

Layer	Output size	Filter
Input	$3 \times 64 \times 64$	-
Conv2D	$32 \times 64 \times 64$	$3 \rightarrow 32$
ResnetBlock-0	$64 \times 64 \times 64$	$32 \rightarrow 64$
AvgPool	$64 \times 32 \times 32$	-
ResnetBlock-1	$128 \times 32 \times 32$	$64 \rightarrow 128$
AvgPool	$128 \times 16 \times 16$	-
ResnetBlock-2	$128 \times 16 \times 16$	$128 \rightarrow 128$
AvgPool	$128 \times 8 \times 8$	-
ResnetBlock-3	$128 \times 8 \times 8$	$128 \rightarrow 128$
AvgPool	$128 \times 4 \times 4$	-
ResnetBlock-4	$256 \times 4 \times 4$	$128 \rightarrow 256$
AvgPool	$256 \times 2 \times 2$	-
ResnetBlock-5	$256 \times 2 \times 2$	$256 \rightarrow 256$
Reshape	256.2.2	-
Text-Img feat concat	n_{fused}	-
Fully connected	$n_{fused} \rightarrow 1$	$n_{fused} \rightarrow 1$

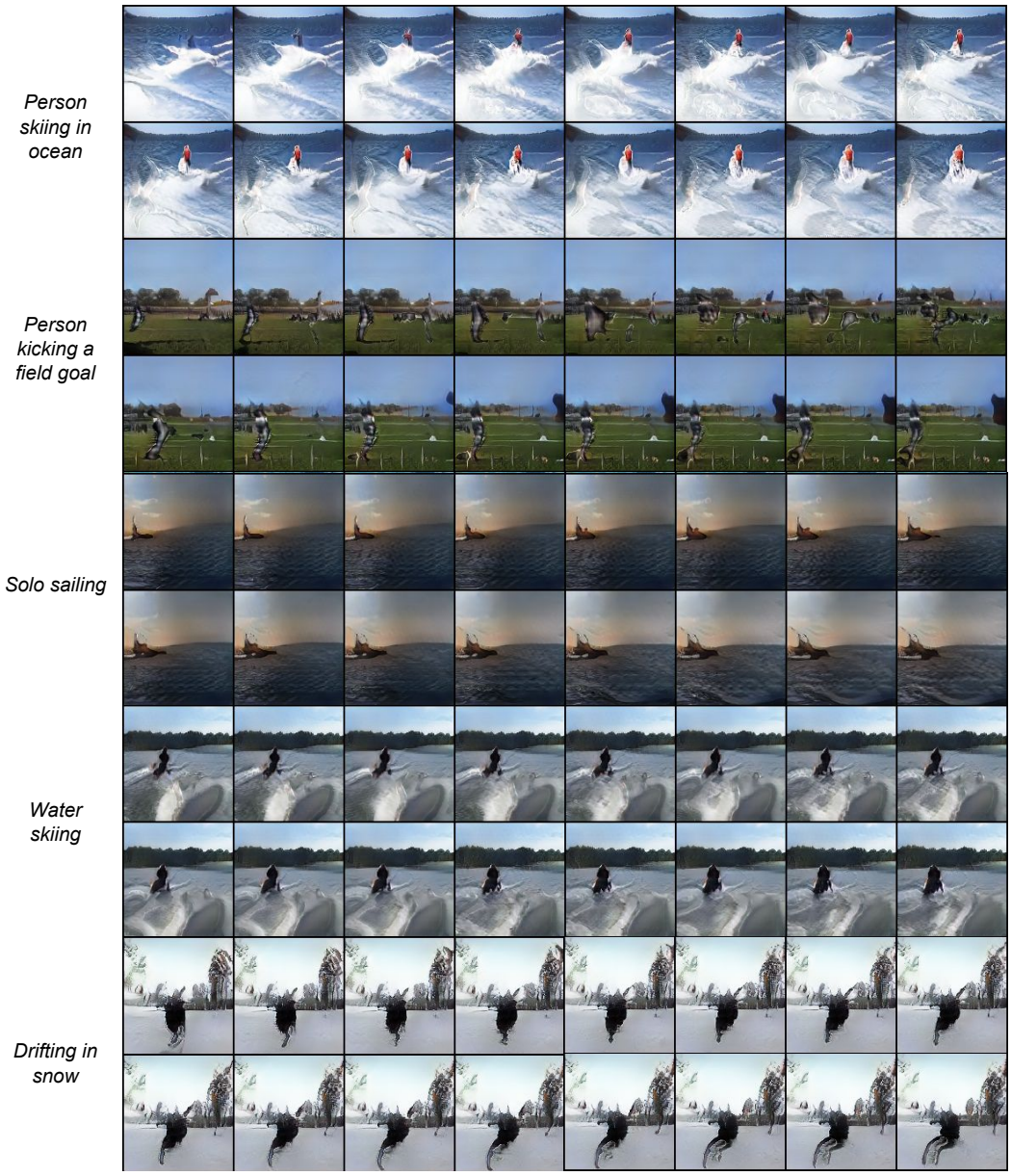


Figure 7: Additional results on Kinetics dataset

D.2 EPIC-KITCHENS DATASET

In this experiment, we consider Epic-Kitchens dataset (Kay et al., 2017) which is a ego-centric dataset containing videos of people cooking in a kitchen. The videos were recorded using a high-definition head mounted camera. The dataset is annotated with text descriptions of step-by-step instructions of a cooking recipe with the corresponding timestamp as people perform the action. We extracted clips of 16 frames containing the following action classes: 'take', 'cut', 'dicing', 'pour', 'stir', 'wash', 'grate', 'rinse', 'put'. This resulted in 4793 (video, text) pairs. We trained our TFGAN model on this dataset. The generations we obtained are shown in Fig. 9. We observe that our model is able to produce good videos: In the first video, we observe the action of a person adding the cherry

Table 8: Video Discriminator architecture - Shapes dataset

Layer	Output size	Filter
Input	$3 \times 16 \times 64 \times 64$	-
Conv3D	$32 \times 16 \times 64 \times 64$	$3 \rightarrow 32$
ResnetBlock3D-0	$64 \times 16 \times 64 \times 64$	$32 \rightarrow 64$
AvgPool3D	$64 \times 8 \times 32 \times 32$	-
ResnetBlock3D-1	$128 \times 8 \times 32 \times 32$	$64 \rightarrow 128$
AvgPool3D-spatial	$128 \times 8 \times 16 \times 16$	-
ResnetBlock3D-2	$128 \times 8 \times 16 \times 16$	$128 \rightarrow 128$
AvgPool3D	$128 \times 4 \times 8 \times 8$	-
ResnetBlock3D-3	$128 \times 4 \times 8 \times 8$	$128 \rightarrow 128$
AvgPool3D	$128 \times 2 \times 4 \times 4$	-
ResnetBlock3D-4	$256 \times 2 \times 4 \times 4$	$128 \rightarrow 256$
AvgPool3D	$256 \times 1 \times 2 \times 2$	-
ResnetBlock3D-5	$256 \times 1 \times 2 \times 2$	$256 \rightarrow 256$
Reshape	256.2.2	-
Text-Img feat concat	n_{fused}	-
Fully connected	$n_{fused} \rightarrow 1$	$n_{fused} \rightarrow 1$

Table 9: Generator architecture - Kinetics dataset

Layer	Output size	Filter
Input	768	-
Fully connected	8192	$768 \rightarrow 8192$
Reshape	$512 \times 4 \times 4$	-
ResnetBlock-00	$512 \times 4 \times 4$	$512 \rightarrow 512$
ResnetBlock-01	$512 \times 4 \times 4$	$512 \rightarrow 512$
Upsample	$512 \times 8 \times 8$	-
ResnetBlock-10	$512 \times 8 \times 8$	$512 \rightarrow 512$
ResnetBlock-11	$512 \times 8 \times 8$	$512 \rightarrow 512$
Upsample	$512 \times 16 \times 16$	-
ResnetBlock-20	$256 \times 16 \times 16$	$512 \rightarrow 256$
ResnetBlock-21	$256 \times 16 \times 16$	$256 \rightarrow 256$
Upsample	$256 \times 32 \times 32$	-
ResnetBlock-30	$128 \times 32 \times 32$	$256 \rightarrow 128$
ResnetBlock-31	$128 \times 32 \times 32$	$128 \rightarrow 128$
Upsample	$128 \times 64 \times 64$	-
ResnetBlock-40	$64 \times 64 \times 64$	$128 \rightarrow 64$
ResnetBlock-41	$64 \times 64 \times 64$	$64 \rightarrow 64$
Upsample	$64 \times 128 \times 128$	-
ResnetBlock-50	$32 \times 128 \times 128$	$64 \rightarrow 32$
ResnetBlock-51	$32 \times 128 \times 128$	$32 \rightarrow 32$
Conv2D	$3 \times 128 \times 128$	$32 \rightarrow 3$
Tanh	$3 \times 128 \times 128$	-

tomato to the bowl and taking the hands off. The images of water splashing and spinach placed in the side can be seen from the third video.

Table 10: Frame Discriminator architecture - Kinetics dataset

Layer	Output size	Filter
Input	$3 \times 128 \times 128$	-
Conv2D	$32 \times 128 \times 128$	$3 \rightarrow 32$
ResnetBlock-00	$32 \times 128 \times 128$	$32 \rightarrow 32$
ResnetBlock-01	$64 \times 128 \times 128$	$32 \rightarrow 64$
AvgPool	$64 \times 64 \times 64$	-
ResnetBlock-10	$64 \times 64 \times 64$	$64 \rightarrow 64$
ResnetBlock-11	$128 \times 64 \times 64$	$64 \rightarrow 128$
AvgPool	$128 \times 32 \times 32$	-
ResnetBlock-20	$128 \times 32 \times 32$	$128 \rightarrow 128$
ResnetBlock-21	$256 \times 32 \times 32$	$128 \rightarrow 256$
AvgPool	$256 \times 16 \times 16$	-
ResnetBlock-30	$256 \times 16 \times 16$	$256 \rightarrow 256$
ResnetBlock-31	$512 \times 16 \times 16$	$256 \rightarrow 512$
AvgPool	$512 \times 8 \times 8$	-
ResnetBlock-40	$512 \times 8 \times 8$	$512 \rightarrow 512$
ResnetBlock-41	$512 \times 8 \times 8$	$512 \rightarrow 512$
AvgPool	$512 \times 4 \times 4$	-
ResnetBlock-50	$512 \times 4 \times 4$	$512 \rightarrow 512$
ResnetBlock-51	$512 \times 4 \times 4$	$512 \rightarrow 512$
Reshape	512.4.4	-
Text-Img feat concat	n_{fused}	-
Fully connected	$n_{fused} \rightarrow 1$	$n_{fused} \rightarrow 1$

Table 11: Video Discriminator architecture - Kinetics dataset

Layer	Output size	Filter
Input	$3 \times 16 \times 128 \times 128$	-
Conv3D	$32 \times 16 \times 128 \times 128$	$3 \rightarrow 32$
ResnetBlock3D-00	$32 \times 16 \times 128 \times 128$	$32 \rightarrow 32$
ResnetBlock3D-01	$64 \times 16 \times 128 \times 128$	$32 \rightarrow 64$
AvgPool3D	$64 \times 8 \times 64 \times 64$	-
ResnetBlock3D-10	$64 \times 8 \times 64 \times 64$	$64 \rightarrow 64$
ResnetBlock3D-11	$128 \times 8 \times 64 \times 64$	$64 \rightarrow 128$
AvgPool3D	$128 \times 4 \times 32 \times 32$	-
ResnetBlock3D-20	$128 \times 4 \times 32 \times 32$	$128 \rightarrow 128$
ResnetBlock3D-21	$256 \times 4 \times 32 \times 32$	$128 \rightarrow 256$
AvgPool3D	$256 \times 2 \times 16 \times 16$	-
ResnetBlock3D-30	$256 \times 2 \times 16 \times 16$	$256 \rightarrow 256$
ResnetBlock3D-31	$512 \times 2 \times 16 \times 16$	$256 \rightarrow 512$
AvgPool3D-spatial	$512 \times 2 \times 8 \times 8$	-
ResnetBlock3D-40	$512 \times 2 \times 8 \times 8$	$512 \rightarrow 512$
ResnetBlock3D-41	$512 \times 2 \times 8 \times 8$	$512 \rightarrow 512$
AvgPool3D	$512 \times 1 \times 4 \times 4$	-
ResnetBlock3D-50	$512 \times 1 \times 4 \times 4$	$512 \rightarrow 512$
ResnetBlock3D-51	$512 \times 1 \times 4 \times 4$	$512 \rightarrow 512$
Reshape	512.4.4	-
Text-Img feat concat	n_{fused}	-
Fully connected	$n_{fused} \rightarrow 1$	$n_{fused} \rightarrow 1$

Figure 8: Variations wihin a category - Swimming class

Figure 9: Results on Epic-Kitchens dataset

D.3 TEXT-TO-IMAGE GENERATION

Some additional results on CUB-Birds dataset are shown in Fig. 10. We see that our model can generate photo-realistic images with good variations.

Table 12: Hyper-parameter details for Shapes and Kinetics experiments

Parameter	Config
Batch Size	8 videos
Optimizer G	Adam
Learning rate G	0.0001
Adam params G: (β_1, β_2)	$(0.0, 0.99)$
Optimizer D_F	Adam
Learning rate D_F	0.0001
Adam params D_F: (β_1, β_2)	$(0.0, 0.99)$
Regularization γ for D_F	10.0
Optimizer D_V	Adam
Learning rate D_V	0.0001
Adam params D_V: (β_1, β_2)	$(0.0, 0.99)$
Regularization γ for D_V	10.0

Figure 10: Sample generations from text2img model trained on CUB-Birds dataset