SinGAN-GIF: Learning a Generative Video Model from a Single GIF

Rajat Arora
University of California, Davis
arora.rajat428@gmail.com

Yong Jae Lee
University of California, Davis
yongjaelee@ucdavis.edu

Abstract

We propose SinGAN-GIF, an extension of the image-based SinGAN [27] to GIFs or short video snippets. Our method learns the distribution of both the image patches in the GIF as well as their motion patterns. We do so by using a pyramid of 3D and 2D convolutional networks to model temporal information while reducing model parameters and training time, along with an image and a video discriminator. SinGAN-GIF can generate similar looking video samples for natural scenes at different spatial resolutions or temporal frame rates, and can be extended to other video applications like video editing, super resolution, and motion transfer. The project page, with supplementary video results, is: `https://rajat95.github.io/singan-gif/`

1. Introduction

Generative Adversarial Networks (GANs) have come a long way since they were first proposed in 2014 [9]. GANs can now generate high fidelity images, particularly when constrained to a specific class like cars, faces, etc. [2, 17]. They have been used for various image processing applications including super resolution, image editing, and style transfer [5, 19, 41].

Recent work showed that GANs can be trained using only a *single* image [27, 28]. By learning the distribution of patches in the single training image, these methods can generate diverse high quality samples that contain semantically similar visual content but in different configurations and structures. Since structures are more repetitive in textured images [42], these methods typically perform better on natural scenes.

In this work, we propose SinGAN-GIF, an extension of SinGAN [27] to GIFs (or short video clips). Our approach takes a single GIF as input, and learns the spatio-temporal patch distributions. Once trained, it can generate new GIFs that contain the same semantic content of the input training GIF but with different spatio-temporal structures and configurations (e.g., different spatial aspect ratios, different temporal speeds). See Figure 1.

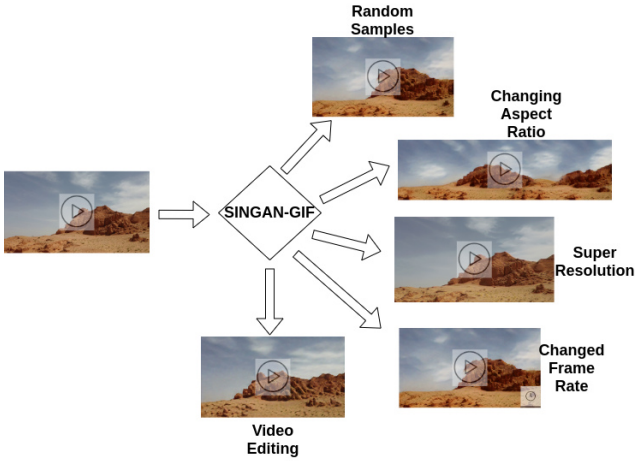

Figure 1. Given a single GIF (short video clip) as input, our model SinGAN-GIF, learns a generator to generate random samples that capture variations of the same visual content. SinGAN-GIF can generate samples at any aspect ratio, perform super-resolution, change the temporal frame rate, and be used for video editing applications.

There are some key challenges that make the extension of SinGAN to the video domain non-trivial. Despite tremendous progress in image generation, generating videos is still a largely open problem. The main difficulties are increased data complexity with the addition of the temporal dimension, and typically huge computation and memory requirements that hinder direct extensions of image generation techniques to the video domain. Despite these challenges, a GIF or short video snippet is much simpler than regular videos in complexity and demands far less computational resources. Furthermore, GIFs often contain simple, repetitive patterns, especially if they pertain to natural scenes. Therefore, by focusing on GIFs, we can circumvent many of the usual difficulties of video generation.

Our network architecture builds upon SinGAN [27] as a baseline and extends it to videos. SinGAN's network consists of a pyramid of fully convolutional GANs, each re-

sponsible for capturing the patch distribution at different resolutions. We replace the 2D spatial convolutions with 3D spatio-temporal convolutions, use two sets of discriminators – one spatial (2D) and one spatio-temporal (3D), and to improve color fidelity, we apply a color statistic matching loss. To alleviate large computation cost and training time, we perform channel-wise separable 3D convolutions and perform only 2D spatial convolutions at larger resolutions in the pyramid, once the coarse spatio-temporal structure is modeled at lower resolutions. We show that our model can generate diverse variations of a single GIF with different aspect ratios and resolutions especially on natural scenes (similar to single image generation models), which contain repetitive and smooth spatio-temporal patterns. We also demonstrate video specific applications including slomo, extrapolation, and motion transfer.

2. Related Work

We summarize related work in generative video models and single image generation models.

2.1. Generative Video Models

Although videos are essentially only a sequence of frames, extending image generation techniques to videos continues to be a challenging problem. Works that generate motion can broadly grouped into two types: (1) unconditional in which a video is generated from a noise vector, and (2) conditional in which the a video is generated conditioned on a specific, often user supplied, input.

Unconditional methods usually use GANs [9] to learn a mapping between a noise vector and a data sample in the true distribution. Existing video generation work split the noise vector to model background and foreground separately and then stitch them together [34], split the generator into temporal and image parts and use a temporal generator to predict a sequence of noise vectors for the image generation submodule [26], use RNNs to recursively obtain motion codes while using per-frame and video-level discriminators [33], use multiple streams to model motion and appearance [35], or propose the sliced Wasserstein loss for video generation [38]. Recent work uses a BigGAN [2] like architecture along with temporal and spatial downsampling for the discriminators to achieve state of the art generation results on the Kinetics-600 dataset [6].

An overwhelming majority of works in video generation are conditional in nature in the sense that their output is conditioned on previous frames (video prediction [1, 3, 20, 30, 36, 37]; video in-betweening [21]) rather than predicting new videos from a single noise vector. For video prediction, existing work use VAEs to estimate a posterior on the input frames to stochastically predict future frames [1, 20], add a residual to the noise vector to sequentially predict future frames [8], propose video pixel networks to directly model the joint distribution of raw RGB pixels [16], or propose latent video transformers that model dynamics in latent space and thereby reduce computational complexity [25]. Other lines of work instead deal with fixed domains like cityscapes, clouds, and time-lapses [7, 23, 24]. However, these methods are either limited to low spatial resolution or model simpler optical flow instead of predicting in a more complex RGB space. Moreover, many of the above models operate at the frame-level and fuse the frame-level information to form a representation for videos, whereas we focus on directly modeling videos as a whole.

2.2. Single Image Generation Models

Single Image generative models work on the principle that patches repeat internally across the image at various scales and can be used to create new images that largely maintain a global structure while retaining finer textures. [29, 40] successfully trained such models for specific tasks like super-resolution, texture expansion, etc. InGAN [28] used a generative model to learn the internal distribution of image patches using a multi-scale hierarchical discriminator, but focused mainly on changing aspect ratio. SinGAN [27] successfully trained a multi-purpose generative model that generates realistic random samples from a single image and demonstrated numerous applications including harmonization, super-resolution, image editing, etc.

While we are still a long way from generic realistic video generation, our work focuses on generating a single GIF (i.e., a short video clip). This problem is much simpler in complexity compared to modeling a large video dataset. Furthermore, since existing work have shown to be able to model the internal distribution of image patches well, we demonstrate that they can be extended to generate realistic video samples for video editing applications, especially for natural scenes.

3. Approach

We first briefly describe SinGAN [27] which was proposed for images, and then describe in detail the modifications that we made to extend it to video data.

3.1. Background: SinGAN [27]

SinGAN consists of a series of generators $(G_1, G_2, ..., G_n)$ operating on an image pyramid $(x_1, x_2, ..., x_n)$ where each successive training sample x_i's spatial dimensions are upsampled by a factor of r ($r > 1$) over its coarser counterpart x_{i-1}. Each generator G_i focuses on modeling the distribution of patches at the scale corresponding to x_i via adversarial training [9]. At the coarsest scale, generator G_1 take a Gaussian noise map z_1 as input and produces image sample $\tilde{x_1}$:

$$\tilde{x}_1 = G_1(z_1). \tag{1}$$

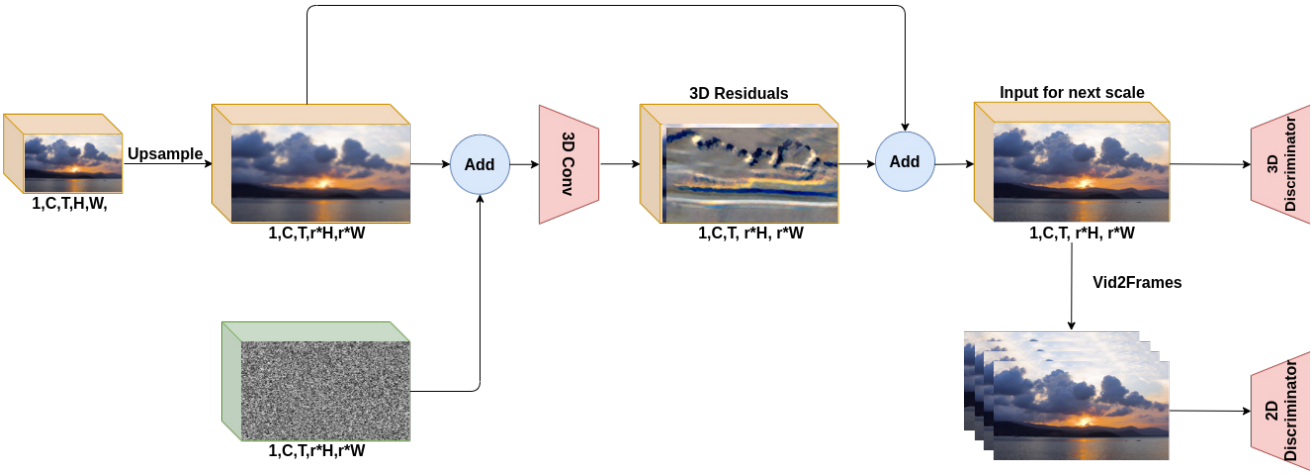

Figure 2. Training pipeline for SinGAN-GIF's 3D convolutional network. Given a video snippet at scale n with spatial dimensions H and W, channel dimension C, and temporal dimension T, we first upsample it by a factor of r. This upsampled video is added to a randomly sampled Gaussian noise map and passed through a 3D CNN to estimate a residual value to refine the upsampled video output. It is trained with adversarial losses from two discriminators working on the entire snippet and individual frames, respectively. This refined output is used as input for the next scale where the process repeats.

The remaining generators focus on refining and adding missed details to the output of the generators preceding them. This is done by up-sampling the output of the preceding generator and adding a Gaussian noise map to predict a residual image, which when added to the input upsampled image results in a more detailed image. Residuals are estimated at each stage instead of directly generating the image, so that the generators do not disregard the output from the coarser generators. This process can be written as:

$$\tilde{x}_n = \uparrow \tilde{x}_{n-1} + G_n(z_n, \uparrow \tilde{x}_{n-1}). \quad (2)$$

Along with random samples, SinGAN also preserves a series of noise maps $z_{opt} = (z^*, 0, 0, ..., 0)$, which are used to reconstruct the original image back via a reconstruction loss. z^* is a noise map that is drawn once and kept fixed during training. Having a pre-decided z_{opt} enables the model to generate the original image back, which is necessary for image manipulation applications that directly edit the original image.

3.2. SinGAN-GIF

A simple approach to extend SinGAN to videos is to use dual discriminators, with one of them distinguishing generated samples from real ones at the image-level while the other focuses on the temporal information at the video-level, similar to prior video generation work [6, 33, 35]. In accordance with SinGAN, we maintain a pyramid of generators each of which operates on a successively upscaled version of the input video; i.e., given a pyramid of n generators from 1 to n, each generator operates on a version of the original input scaled by a factor r^i. The output from the i^{th} generator is upsampled and added to a noise map before being fed to the next generator, as shown in Fig. 2. This ensures that successive generators consider both the noise as well as the previous generation for the new output.

Each generator is associated with two discriminators: one of which operates on each of the output frames individually, while the other works on the entire video snippet as a whole. Ideally, only a single video discriminator should be able to check for both spatial and temporal inconsistencies but empirically we found high frequency noise artifacts to be frequent when using only the video discriminator.

Another difference in training procedure when working with longer GIF sequences of e.g., $T = 32$ frames, is that all frames may not fit in GPU memory within a single forward pass especially for higher resolutions. This means that we need to have different z_{opt}'s (needed for the reconstruction loss) for different chunks of the same video clip since we can only pass e.g., $t = 8$ frames at a time. We solve this by simply pre-setting a $T \times h \times w$ noise map for the entire snippet but clip a $t \times h \times w$ portion out of it, corresponding to the frames currently being processed through the network (where h and w are height and width at a specific scale). We notice this simple approach gives reasonably accurate reconstructions.

While this simple approach achieves qualitatively good results, it is quite slow to train even at a low spatial resolution of about 160x160 and a low temporal resolution of 8 frames on a NVIDIA 1080ti GPU. The reason is in large part due to naive 3D convolutions having many more parameters than an analogous 2D convolutional network.

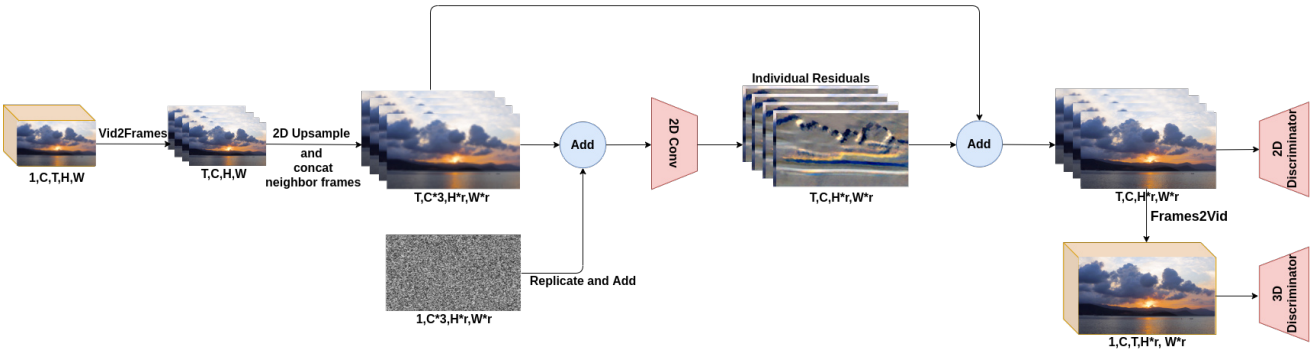

Figure 3. Training pipeline for SinGAN-GIF's 2D convolutional network. At a given scale n, frames are first extracted from a video snippet obtained from the last scale $n-1$. The frames are then upsampled and appended to their temporal neighbors along the channel dimension and noise is added. They are then passed through a 2D CNN which predicts residuals to refine each input frame. The refined frames are individually passed through an image discriminator and also converted back to a video and passed through a video discriminator.

3.3. Towards a More Lightweight Architecture

We make the following adjustments to speed up training at higher resolutions.

3.3.1 Channel-wise separable 3D Convolution

Channelwise 3D convolutions split a $k \times h \times w$ convolution into a $1 \times h \times w$ spatial convolution and a $k \times 1 \times 1$ temporal convolution [13, 12]. This not only reduces the number of learnable parameters but also makes training faster and more stable without any major impact on the quality of results. We use separable 3D convolutions for the 3D generator. We attempted to use separable convolutions for the discriminator as well but we found that it leads to highly unstable training so we use regular 3D convolutions instead.

3.3.2 3D-2D Hybrid Pyramid Scheme

We notice that beyond a certain spatial resolution, the temporal context from all frames is not necessary and the context from the immediate temporal neighboring frames is more important. Therefore, to further improve training speed, we start with 3D convolutions for the initial few scales but later switch to 2D convolutions. These 2D convolutions take an upsampled frame from the previous scale as well as its preceding and proceeding neighboring frames appended along the channel dimension. Just like the 3D generator, the output is passed through both image and video discriminators. Furthermore, to ensure each individual frame forms part of the same video sequence, we set the sampled additive noise z to be the same for all individual frames. A detailed description of this step is shown in Fig. 3.

With this change, combined with separable convolutions, we notice a decrease in training time of approximately $\sim$20% at 256x256 resolution.

3.4. Training Procedure

Each generator is trained sequentially moving from coarser scales to finer scales, and we only estimate residuals when moving towards finer scales. Once a generator is trained, it is kept fixed while the next generator is optimized. Each generator is trained using adversarial losses from both video and image discriminators, a reconstruction loss, and a color statistics matching loss:

$$\begin{aligned} L_{G_n} = \min_{G_n} \max_{D_{n\_img}} L_{adv\_img}(G_n, D_{n\_img}) + \\ \alpha \cdot \min_{G_n} \max_{D_{n\_vid}} L_{adv\_vid}(G_n, D_{n\_vid}) + \\ \beta \cdot L_{rec}(G_n) + \gamma \cdot L_{stat}(G_n), \end{aligned} \quad (3)$$

where G_n, D_{n_vid}, D_{n_img} denote the generator, video discriminator, and image discriminator at the n'th scale of generation, respectively. L_{adv_vid} and L_{adv_img} are the adversarial losses for video and image discriminators, respectively, while L_{rec} and L_{stat} are reconstruction and color statistics matching losses, respectively. We explain each loss in detail next.

3.4.1 Adversarial Losses

Each pair of generator and corresponding discriminators are trained using the WGAN-GP [10] loss as it is shown to increase stability:

$$\begin{aligned} L = \underbrace{\mathbb{E}_{\tilde{x} \sim P_g}[D(\tilde{x})] - \mathbb{E}_{x \sim P_r}[D(x)]}_{CriticLoss} + \\ \underbrace{\lambda \cdot \mathbb{E}_{\tilde{x} \sim P_g}([\|\nabla D(\tilde{x})\|_2 - 1])^2}_{GradientPenalty} \end{aligned} \quad (4)$$

We train the discriminators by averaging over image patches and video volumes for images and videos, respectively, similar to patchGAN [14].

Figure 4. Effect of adding color statistics loss. First Row: Original 4 consecutive frames of a given video. Second Row: Random consecutive frames generated without Color Statistics Loss. Third Row: Random consecutive frames generated with Color Statistics Loss. It is clear that images in second row seem to not have the same color scheme of the original video. But after adding the statistics loss the color matches the input frames faithfully while also being sufficiently distinct from the input video in terms of layout.

3.4.2 Reconstruction Loss

In order to make it possible to edit the original frames, we need to be able to generate training frames back from generators in a controllable way. To ensure this, we choose a set of z's prior to training (as explained earlier) which must map back to the original frames. Thus at a scale n, the reconstruction loss can be stated as:

$$L_{rec} = \|(G_n(z_{opt}^{n-1}, \uparrow \tilde{x}_{n-1}^{rec}) + \tilde{x}_{n-1}^{rec}) - x_n\|_1 \quad (5)$$

where x_n is the original snippet at the n'th scale while $\tilde{x}_{n-1}^{rec}$ is the generated sample and z_{opt}^{n-1} is the pre-determined noise map at the preceding $n-1$ scale.

Similar to SinGAN [27], we use the reconstruction loss to determine σ for the Gaussian noise map of the next finer scale. However, since we are using a set of frames of a video instead of a single image, we use the average RMSE between all frames of the reconstructed video and the input video to determine σ_n for scale $n+1$. Using reconstruction error to estimate σ helps the network estimate the extent of details to be added since a higher RMSE would mean more significant changes are required for improvement as compared to a lower RMSE.

3.4.3 Color Statistics Loss

While SinGAN [27] uses only the adversarial and reconstruction losses, for videos we see that with just these losses the generated samples do not match the color distribution of the training video frames; see Fig 4. We hypothesize that small errors in estimating residual images leads to small divergences from the original distribution which over multiple scales leads to a very noticeable difference. This issue was also reported in [39] and was resolved by matching the mean and covariance of color channels of generated images at successive spatial resolutions. Since we are training our model on a single video, we simply match the mean and variance of each color channel of each frame of the generated video to the input video:

$$L_{stat} = \mathbb{E}_{\tilde{x}\sim P_g, x\sim P_r, i\sim\{r,g,b\}}[\|\mu(\tilde{x}^i) - \mu(x^i)\|_2 + \|Cov(\tilde{x}^i) - Cov(x^i)\|_2] \quad (6)$$

where μ is mean, cov is covariance, P_r is real distribution, P_g is generated distribution, x is the set of training frames, and $\tilde{x}$ denotes generated frames.

3.5. Implementation Details

Training all pairs of generators and discriminators using Adam [18] with a learning rate of 5e-4 for roughly 8000 iterations produces decent results for all GIFs, although the optimal hyperparameters can vary for individual GIFs. We start with a batch size of 16 (or total number of frames if less than 16) and gradually reduce to 8. We find that 3D convolution for the coarsest 4 scales followed by 2D convolutions for the rest of the finer scales give the best results. The values for α, β, γ in Eq. 3 are determined empirically to be 5, 50, and 2 respectively. The smallest spatial dimension at the coarsest scale is set at 25 pixels while $r = 4/3$ is the scaling factor. The gradient penalty term for both discriminators is set to 0.1.

4. Results

With SinGAN-GIF, we can train GIFs with a spatial resolution of 256x256 using a single Titan XP GPU. We experiment with a variety of GIFs to show the flexibility and generality of our approach. The training videos belong to diverse natural scenes like cloudy sky, waterfall, desert, mountains, sunset, seashore, nightime-timelapse, river, and flock of birds. Each GIF consists of 8-32 frames.

In all the results (see Fig. 5 and supplementary videos in the project page), we see that our method not only captures the textures and relative positions of the objects in the videos, but also their associated motion. Furthermore, despite training with just 8 frames during a single forward pass, our generation need not be limited to just an 8 frame

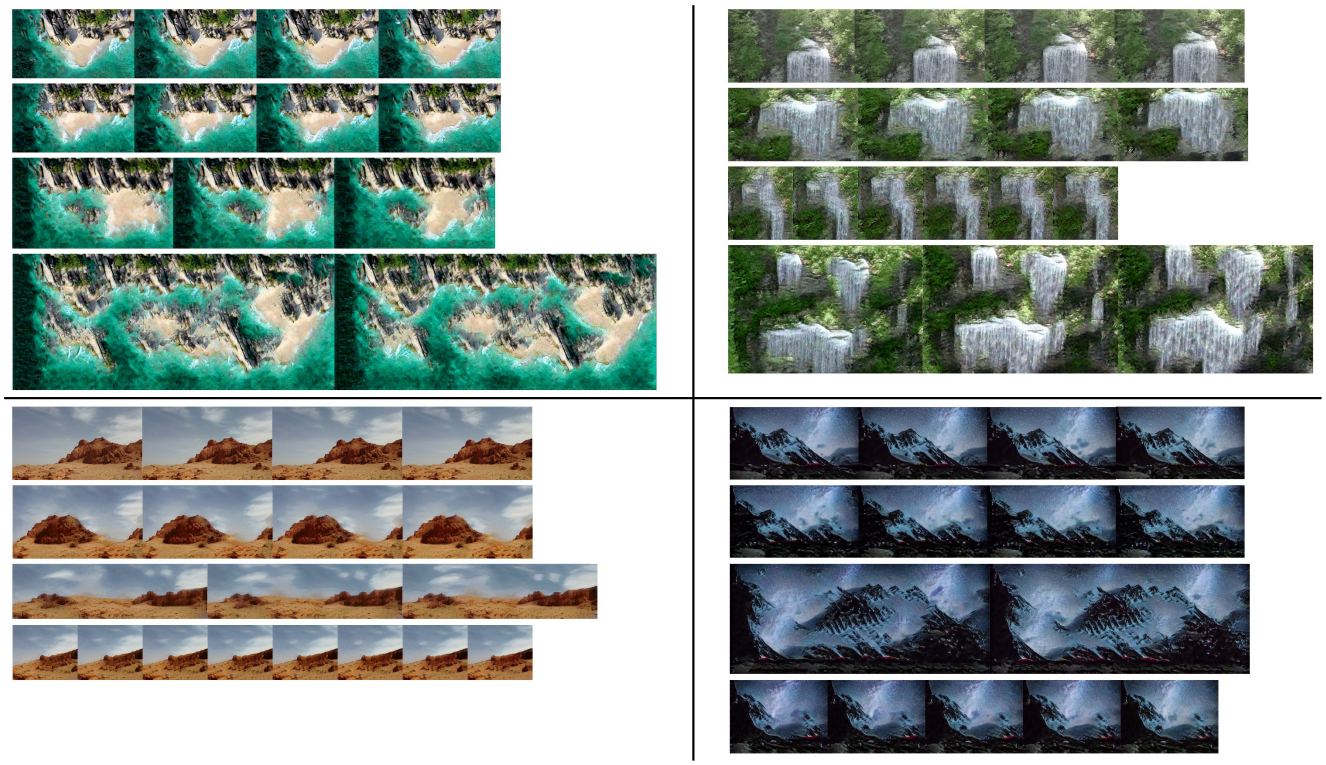

Figure 5. Row1: Original training frames. Row2-4: Frames from videos generated with different aspect ratios. Link to generated GIFs in supplementary.

video as the network is able to utilize the context to generate meaningful videos of up to 32 frames.

In order to quantitatively evaluate the quality of our generated GIFs, we calculate the Single-GIF FID (SGFID) score, which is a direct extension of the Single-Image FID score [27]. Specifically, for each frame of the generated GIF, we take the conv layer features right before the second pooling layer of the Inception Network [31] (i.e., one feature vector per spatial location in the corresponding feature map per frame). The SGFID is the FID [11] between the statistics of those features in the real video (across all frames) and those features in the generated sample.

We compare our scores against a simple modification of SinGAN where instead of training with just a single image we train it on all frames of a video together by passing a randomly selected frame through the discriminator during each forward pass. As can be seen from Table 1, our output has significantly lower SGFID scores indicating that our method models the patches in the original frames better than trivially training SinGAN with all the frames of a given GIF.

Finally, we also perform qualitative user studies using Amazon Mechanical Turk. Workers have access to each video for 5 seconds and can play it twice before making a decision. We asked mechanical turkers to identify whether a video was real or fake. We generated 10 samples each for 9 videos (a total of 90 generated samples), and for each sample had 3 turkers evaluate. We took the majority vote from the 3 evaluaters to filter out noise. 44 out of 90 generated samples were identified as real, and the remaining 46 were identified as fake. This shows that our generated samples, while not perfect, are reasonably real looking.

Model	SGFID score
SinGAN-GIF	0.37
SinGAN (All Frames)	0.89

Table 1. SGFID scores averaged over 90 GIFs (9 training GIFs, each with 10 generations). Each generated video has the same number of frames as the training GIF.

5. Applications

We next demonstrate a number of video editing applications using SinGAN-GIF. Our goal here is to demonstrate the feasibility of our single architecture in performing a variety of diverse tasks, but we note that there are task specific methods like [15] for video slo-mo and [32] for super-resolution, which outperform our approach.

5.1. Changing Aspect Ratios

Figure 5 shows how our model is able to produce frames that are variations of the original video in different spatial

Figure 6. Row1: Frames edited using SinGAN-GIF. Row2: Seam introduced from naive translation of the frames to the left.

resolutions. Importantly, the temporal relationship between consecutive frames also makes sense. This application can have real world use cases where one may need similar looking videos to fit to different screen-sizes or for data augmentation to get a variety of videos from a single data point. Since our model is fully convolutional, different spatial resolutions can be easily achieved by changing the shape of the input Gaussian noise maps.

5.2. Video Slo-Mo

We can also modify SinGAN-GIF slightly to obtain slower versions of an input GIF. For this, we model input frames as a path in latent space i.e., we assume the first and last frames come from two different Gaussian distributions centered at -1 and +1, respectively. The rest of the frames can then be sampled using the formula $(n/N) * z1 + ((N - n)/N) * z2$, where n is the current frame while N is the total number of frames we want to generate. Sampling frames like this can be thought of as a timer which can help the network distinguish between the frame rate of the output video. This though poses a problem since we get patches from the input GIF only at a fixed rate.

To work around this problem, while the generator is trained to generate frames at a variable frame rate, we subsample the generated video back to the frame rate of the original training GIF before passing it to the discriminators. For example, in order to generate videos at half speed, during each forward pass we will generate twice as many frames as the original video, and select every alternate frame and pass it to the discriminator so that it can be compared against the original clip. All the frames are still individually passed to the image discriminator to make sure each frame looks realistic as well. The results of this method can be seen in the supplementary.

5.3. Video Lengthening

Since our model learns the inherent pattern of motion and the correct spatial locations of objects in relation to the overall scene, we can train our model on a small clip of just 8 frames but generate realistic videos which are longer up to 32 frames. This is because motions are often repetitive especially in natural scenes, and neighboring frames provide enough context to the model to reliably extend the motion using the patterns it has learned from just the 8 input frames of the GIF. The results can be found in the supplementary.

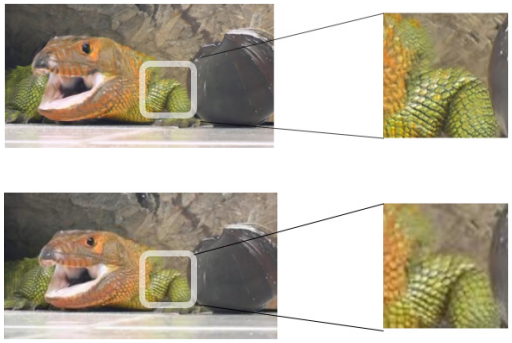

Figure 7. Top: Result for 3x upsampling using the finest scale of SinGAN-GIF. Bottom: Result for 3x upsampling using bilinear interpolation. One can see sharper details in the top figure. Link to the complete super-resolution video result can be found in the supplementary.

5.4. Video Seamless Composition

SinGAN-GIF can be used to create translated or locally edited versions of input videos that are free of obvious seams. Our method is able to achieve this by passing downsampled locally edited frames to a coarse generator (say n, typically 3) and running it through the rest of the finer generators from $\{n + 1, ..., N\}$. That is, we skip the first n generators, and instead input the locally edited frames directly to the $n + 1$'th generator. Also, unlike SinGAN, we notice a single pass through the set of generators may not fix all the artifacts. Thus, the output from one pass through the generators is downsampled again and the process is re-

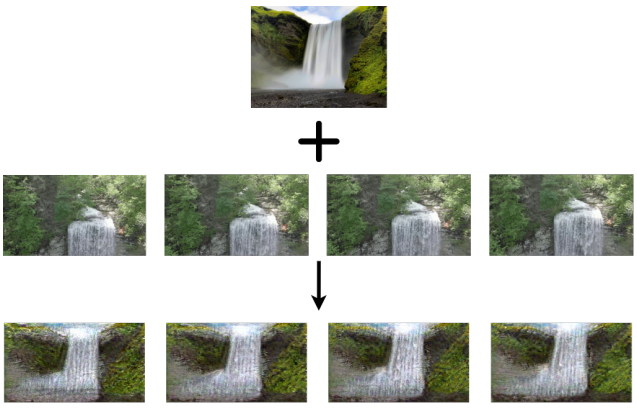

Figure 8. Row 1: A random image of a waterfall semantically similar to the training input. Row 2: Training input GIF of a waterfall. Row 3: Animated input image. Link to complete result video can be found in the supplementary.

peated. A small sequence of frames edited like this is shown in Figure 6.

5.5. Video Super-Resolution

It has been shown that patches in a given image tend to recur repeatedly over multiple scales [42]. This fact coupled with the intrinsic design of our model enables us to perform video super resolution by iteratively refining the input frames through the finest generator, which adds the most high frequency details. This is done in a manner similar to what was proposed in SinGAN. To upsample frames by a factor of k, we set the pyramid scale factor to $\sqrt[n]{k}$, where $k \in \mathbb{N}$ and the reconstruction error is given a higher weight of 125. A result is shown in Figure 7.

5.6. Motion Transfer

We also explore animating a single image which looks semantically similar to a training GIF. We replicate the image 8 times along the temporal dimension and pass it to the second coarsest generator. (Since the coarsest generator only takes a noise map as input, we cannot start at the coarsest level.) While this is not perfect since a random image will not exactly match the patches' patterns in the training video snippet, our method is still able to transfer motion reasonably well to correct regions and produce a reasonable output despite this being a very hard task. See Fig. 8.

6. Limitations

Since 3D convolutions are the building blocks for our architecture, memory is a bottleneck that limits us to train with smaller frame lengths and spatial resolutions despite the modifications in Section 3.3. Mixed precision training [22] or gradient checkpointing [4] are possible ways to overcome this limitation.

Figure 9. Top: Single frame of the training GIF. Bottom: Single frame from a random generated sample. Although the random sample follows the distribution of patches in the training frame, due to the absence of any high-level knowledge of the object present, the output does not make semantic sense.

Another limitation arises from the fact that a single video does not provide enough semantic information required to model rich real world dynamics. Thus, it can produce unnatural frames in the presence of a salient object as shown in Fig. 9. But an interesting thing to note here is that despite the unnatural appearance of frames, each part of the frame does move in a predictable way indicating our model successfully links motion to their corresponding 'entity'.

7. Conclusion

In this paper, we presented an approach to train deep generative networks on a single GIF (or short video clip). We showed that our model, SinGAN-GIF, can generate similar looking samples at different resolutions or frame rates. We also showed how SinGAN-GIF can be extended to other applications like video editing, super resolution, and motion transfer. We achieved this by extending the image-based SinGAN model, and using a combination of 3D and 2D convolutions to model temporal information while reducing model parameters.

Drawbacks include long training times and limited spatial resolutions due to memory. While generating random diverse videos is still a very open and challenging problem, by limiting the training data to a single GIF, we can generate realistic video samples for natural scenes and can model both the patch distribution and their associated motion successfully. We believe our work is an encouraging step in unconditional video generation research.

Acknowledgements We thank the reviewers for their helpful comments. This work was supported in part by NSF IIS-1748387, ARO YIP W911NF17-1-0410, and Adobe Data Science Research Award.

References

[1] Mohammad Babaeizadeh, Chelsea Finn, Dumitru Erhan, Roy H Campbell, and Sergey Levine. Stochastic variational video prediction. *arXiv preprint arXiv:1710.11252*, 2017.

[2] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018.

[3] Wonmin Byeon, Qin Wang, Rupesh Kumar Srivastava, and Petros Koumoutsakos. Contextvp: Fully context-aware video prediction. In *ECCV*, 2018.

[4] Tianqi Chen, Bing Xu, Chiyuan Zhang, and Carlos Guestrin. Training deep nets with sublinear memory cost. *arXiv preprint arXiv:1604.06174*, 2016.

[5] Yu Cheng, Zhe Gan, Yitong Li, Jingjing Liu, and Jianfeng Gao. Sequential attention gan for interactive image editing via dialogue. *arXiv preprint arXiv:1812.08352*, 2018.

[6] Aidan Clark, Jeff Donahue, and Karen Simonyan. Adversarial video generation on complex datasets. *arXiv*, 2019.

[7] Yuki Endo, Yoshihiro Kanamori, and Shigeru Kuriyama. Animating landscape: self-supervised learning of decoupled motion and appearance for single-image video synthesis. *arXiv preprint arXiv:1910.07192*, 2019.

[8] Jean-Yves Franceschi, Edouard Delasalles, Mickaël Chen, Sylvain Lamprier, and Patrick Gallinari. Stochastic latent residual video prediction. *arXiv preprint arXiv:2002.09219*, 2020.

[9] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NeurIPS*, 2014.

[10] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In *NeurIPS*, 2017.

[11] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *NeurIPS*, 2017.

[12] Rui Hou, Chen Chen, Rahul Sukthankar, and Mubarak Shah. An efficient 3d cnn for action/object segmentation in video. *arXiv preprint arXiv:1907.08895*, 2019.

[13] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[14] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *CVPR*, 2017.

[15] Huaizu Jiang, Deqing Sun, Varun Jampani, Ming-Hsuan Yang, Erik Learned-Miller, and Jan Kautz. Super slomo: High quality estimation of multiple intermediate frames for video interpolation. In *CVPR*, pages 9000–9008, 2018.

[16] Nal Kalchbrenner, Aäron Oord, Karen Simonyan, Ivo Danihelka, Oriol Vinyals, Alex Graves, and Koray Kavukcuoglu. Video pixel networks. In *ICML*, 2017.

[17] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *CVPR*, 2019.

[18] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[19] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. In *CVPR*, 2017.

[20] Alex X Lee, Richard Zhang, Frederik Ebert, Pieter Abbeel, Chelsea Finn, and Sergey Levine. Stochastic adversarial video prediction. *arXiv preprint arXiv:1804.01523*, 2018.

[21] Yunpeng Li, Dominik Roblek, and Marco Tagliasacchi. From here to there: Video inbetweening using direct 3d convolutions. *arXiv preprint arXiv:1905.10240*, 2019.

[22] Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, and Hao Wu. Mixed precision training. In *ICLR*, 2018.

[23] Seonghyeon Nam, Chongyang Ma, Menglei Chai, William Brendel, Ning Xu, and Seon Joo Kim. End-to-end time-lapse video synthesis from a single outdoor image. In *CVPR*, 2019.

[24] Junting Pan, Chengyu Wang, Xu Jia, Jing Shao, Lu Sheng, Junjie Yan, and Xiaogang Wang. Video generation from single semantic label map. *arXiv preprint arXiv:1903.04480*, 2019.

[25] Ruslan Rakhimov, Denis Volkhonskiy, Alexey Artemov, Denis Zorin, and Evgeny Burnaev. Latent video transformer. *arXiv preprint arXiv:2006.10704*, 2020.

[26] Masaki Saito, Eiichi Matsumoto, and Shunta Saito. Temporal generative adversarial nets with singular value clipping. In *ICCV*, 2017.

[27] Tamar Rott Shaham, Tali Dekel, and Tomer Michaeli. Singan: Learning a generative model from a single natural image. In *ICCV*, 2019.

[28] Assaf Shocher, Shai Bagon, Phillip Isola, and Michal Irani. Ingan: Capturing and remapping the" dna" of a natural image. *arXiv preprint arXiv:1812.00231*, 2018.

[29] Assaf Shocher, Nadav Cohen, and Michal Irani. "zero-shot" super-resolution using deep internal learning. In *CVPR*, 2018.

[30] Nitish Srivastava, Elman Mansimov, and Ruslan Salakhudinov. Unsupervised learning of video representations using lstms. In *ICML*, 2015.

[31] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *CVPR*, 2015.

[32] Yapeng Tian, Yulun Zhang, Yun Fu, and Chenliang Xu. Tdan: Temporally-deformable alignment network for video super-resolution. In *CVPR*, pages 3360–3369, 2020.

[33] Sergey Tulyakov, Ming-Yu Liu, Xiaodong Yang, and Jan Kautz. Mocogan: Decomposing motion and content for video generation. In *CVPR*, 2018.

[34] Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. Generating videos with scene dynamics. In *NeurIPS*, 2016.

[35] Yaohui Wang, Piotr Bilinski, Francois Bremond, and Antitza Dantcheva. This video does not exist. disentangling motion and appearance for video generation. *arXiv preprint arXiv:1912.05523*, 2019.

[36] Yunbo Wang, Lu Jiang, Ming-Hsuan Yang, Li-Jia Li, Mingsheng Long, and Li Fei-Fei. Eidetic 3d lstm: A model for video prediction and beyond. In *ICLR*, 2018.

[37] Henglai Wei, Xiaochuan Yin, and Penghong Lin. Novel video prediction for large-scale scene using optical flow. *arXiv preprint arXiv:1805.12243*, 2018.

[38] Jiqing Wu, Zhiwu Huang, Dinesh Acharya, Wen Li, Janine Thoma, Danda Pani Paudel, and Luc Van Gool. Sliced wasserstein generative models. In *CVPR*, 2019.

[39] 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.

[40] 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.

[41] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *ICCV*, 2017.

[42] Maria Zontak and Michal Irani. Internal statistics of a single natural image. In *CVPR*, 2011.