Diverse Generation from a Single Video Made Possible

Niv Haim*, Ben Feinstein*, Niv Granot, Assaf Shocher, Shai Bagon, Tali Dekel and Michal Irani
Weizmann Institute of Science, Rehovot, Israel

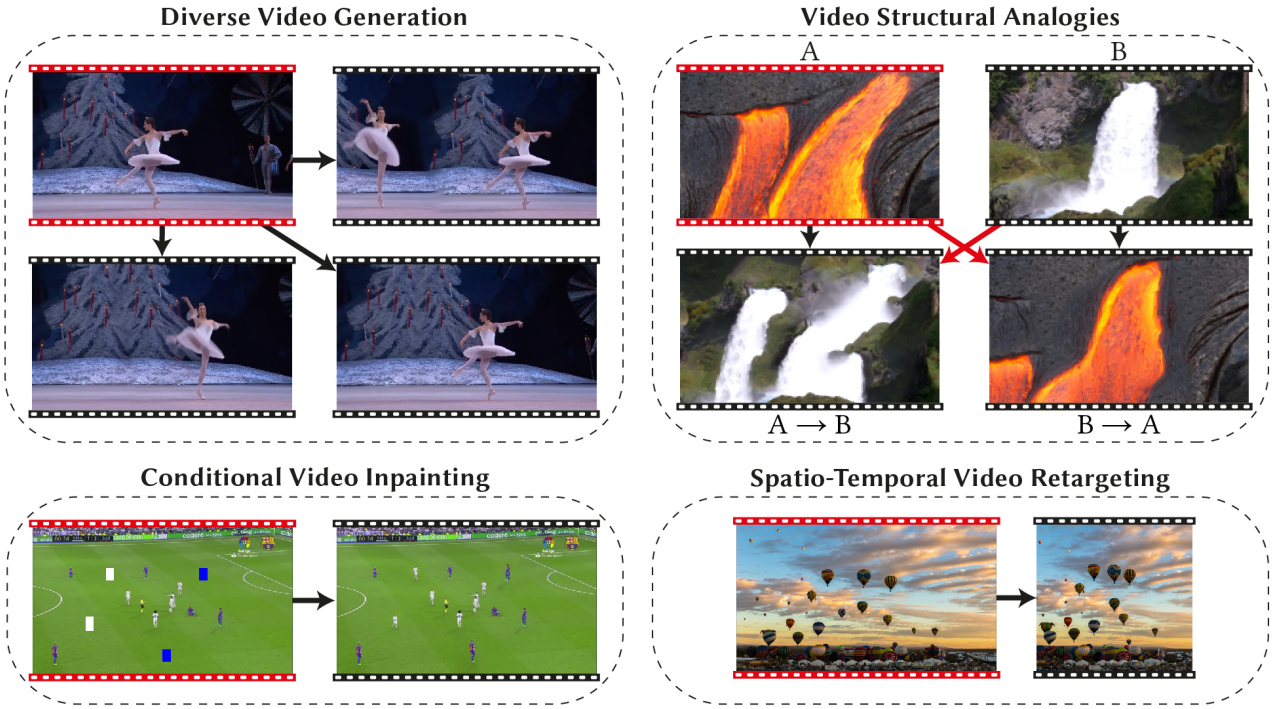

Figure 1. We present fast and practical video generation from a single natural video, which generates *diverse* high-quality video (top left), but also gives rise to additional tasks, such as spatial and temporal retargeting (bottom right), video analogies (top right), conditional video inpainting (bottom left) and more. No pre-training takes place, and the whole process lasts seconds. Input videos marked in red. Actual videos are in the supplementary material.

Abstract

Most advanced video generation and manipulation methods train on a large collection of videos. As such, they are restricted to the types of video dynamics they train on. To overcome this limitation, GANs trained on a single video were recently proposed. While these provide more flexibility to a wide variety of video dynamics, they require days to train on a single tiny input video, rendering them impractical. In this paper we present a fast and practical method for video generation and manipulation from a single natural video, which generates diverse *high-quality video outputs within seconds (for benchmark videos). Our method can be further applied to Full-HD video clips within minutes. Our approach is inspired by a recent advanced patch-nearest-neighbor based approach [16], which was shown to significantly outperform single-image GANs, both in run-time and in visual quality. Here we generalize this approach from images to videos, by casting classical space-time patch-based methods as a new* generative video model. *We adapt the generative image patch nearest neighbor approach to efficiently cope with the huge number of space-time patches in a single video. Our method generates more realistic and higher quality results than single-video GANs (confirmed by quantitative and qualitative evaluations). Moreover, it is disproportionally faster (runtime reduced from several days to seconds). Other than diverse video generation, we demonstrate several other challenging video applications, including spatio-temporal video retargeting, video structural analogies and conditional video-inpainting. Project page:* https://nivha.github.io/vgpnn

*Equal contribution

1. Introduction

Generation and editing of natural videos remain challenging, mainly due to their large dimensionality and the enormous space of motion they span. Therefore, the tremendous recent advances in image generation cannot be readily applied to the video domain. The most common modern framework is training generative models on a large collection of videos. Such methods (e.g., [1, 47]) can typically produce high quality results only for a limited class of videos such as face expressions (e.g., [41]) or robot hand movement (e.g., [10]), and cannot be applied to arbitrary videos. On the other side of the spectrum, *single* video generative models have recently emerged [3, 17]. These methods learn a generative video model from a single input video, in an unsupervised manner. As such, they learn the distribution of space-time patches within the single input video and are then able to generate a plethora of new videos with the same patch distribution. However, single video GANs suffer from some heavy disadvantages - it takes days to train for each input video, thus making them applicable to only very small resolution and video length. They are also prone to optimization issues such as mode collapse, which often lead to poor visual quality and noticeable visual artifacts. These shortcomings render existing single-video methods impractical and unscalable.

Synthesizing and editing video content by modeling the distribution of space-time patches in a single video dates back to classical, pre-deep learning methods. These classical methods are based on optimization over space-time patches, which extended 2D patch-based methods to cope with videos, and demonstrated impressive results for various applications such as video summarization, video completion, and video texture synthesis. However, due to the intense computation required for working with space-time patches within a single video, these methods require long runtime and have been limited to low resolution and short videos.

We present a fast and practical method for video generation and manipulation from a single video, which generates *diverse* high-quality video outputs within seconds/minutes (depending on the video size), and can be applied to high resolution (Full-HD) videos. Our approach is inspired by the recent generative patch nearest neighbors approach, GPNN [16], which was shown to significantly outperform single-image GANs, both in run-time and in visual quality. Here, we generalize this approach from images to videos, by casting classical space-time patch-based methods as a new *generative video model*. That is, given a single input video, our Video-Based Generative Patch Nearest Neighbors (**VGPNN**) method can generate many random novel video outputs, all of which match the distribution of space-time patches of the original video.

Adapting GPNN to the realm of videos includes efficiently coping with the huge number of space-time patches in a single video, and also utilizing the temporal information in videos. On the computational aspect, a naïve extension of GPNN from images to videos is prohibitive in both run time and memory consumption. To this end we extend PatchMatch [6], an efficient approximate nearest-neighbor search method, to search in a globally-weighted patch similarity space (we call our extension *WeightedPatchMatch*). To our surprise, the approximated nearest neighbor field resulting from PatchMatch could still produce high quality results, while reducing generation run time by many orders of magnitude. Thus, VGPNN makes diverse video generation from a single video realistically possible for the first time.

Our VGPNN results outperform single-video GANs, across all measures. Most of all, the runtime is incredibly reduced, from the impractical length of roughly 8 days to about 18 seconds for a $13 \times 144 \times 256$ video on a Quadro RTX 8000 GPU. This makes single video generative models practical for the first time. Beyond our main task of diverse, unconditional generation of videos, our framework gives rise to various classical and new video synthesis and manipulation applications including: spatio-temporal video retargeting (e.g., video extension & video "summarization"), video structural analogies, sketch-to-video, *conditional* video-inpainting, and more, all produced with the same unified framework. Some applications are schematically illustrated in Figs. 1 and 2. Full videos and many more results can be found in the supplementary material.

Our contributions are therefore several fold:

- VGPNN is the first single-video generative model that is scalable to high resolution (spatial or temporal).
- We cast classical space-time patch nearest neighbor methods as a *generative* video model.
- We non-trivially adapt the patch-nearest-neighbor approach to video, making it affordable and practical. In particular, we present *WeightedPatchMatch*, an extension of PatchMatch, that can handle per-patch global information.
- Our approach can be used to a variety of new video applications, all within a single unified framework. These include diverse generation, spatio-temporal retargeting, sketch-to-video, and video structural analogies.

2. Related work

Synthesis and manipulation of videos is mostly done by generative models trained on large collections of videos. The most common approach for unconditional video generative models is an extension of GAN [15] to videos; [1,41,47,51]. There are also many GAN based methods for video-to-video translation [27,48–50]. Another approach is by using GANs for images but predicting paths in the latent

Figure 2. **Diverse Single Video Generation:** Given an input video (red), VGPNN is able to generate similarly looking videos (blue) capturing both appearance of objects as well as their dynamics. Note the high quality of our generated videos. Please watch the full resolution videos in the supplementary material.

space [25,34]. A very common technique is considering the video as a sequence of frames and autoregressively predicting the next frame each time [2, 4, 5, 10, 12, 43–45]. All of these require lots of data to pre-train on, and can only generate videos from the same domain they were trained on.

Some video editing applications were demonstrated before deep-learning gained popularity. One such application is video retargeting or summarization [18,24,31,33,39,53]. [35] considered 3D video textures and were able to synthesize them by statistical modeling. A similar, modern and very impressive approach by [21] combined this rational with deep networks. Some classical methods were based on optimizing similarity of space-time patches [39, 52]. These were extensions of approaches mainly used for images [11, 39]. As such, while performing reasonably on images in terms of runtime and memory, their extension to video is hardly scalable (hours for editing a single video).

Within the field of "Deep Internal Learning" [13, 14, 38, 42, 54, 55, 58], Deep Single-Image Generative Models have emerged. These models train a GAN on a single image, in an unsupervised manner, and have shown to produce impressive results for image synthesis and manipulation tasks. Being fully convolutional, these single-image GANs learn the patch distribution of the single input image, and are then able to generate a plethora of new images with the same patch distribution. These include SinGAN [36] for generating a *large diversity* of different image instances – all sampled from the input patch distribution, InGAN [37] for flexible image retargeting, Structural-Analogies [7], texture synthesis [8, 22, 56, 57], and more [9, 17, 20, 26, 28, 29, 46]. Single image GANs suffer from long training time per image, artifacts and mode collapse.

Recently, this approach was extended to single-video GANs. In the video domain, the main goal is to generate novel instances but with the same content as in the reference video. [3] extends [36] to video in a straightforward manner, by employing 3D (space-time) convolutions. [17] Combines Variational Auto Encoders for the coarse scales, thus preventing mode collapse, with GANs for the fine scale to achieve improved quality. The single-input GANs shortcomings are significantly amplified in the video variants. The runtime is disproportionately increased (to several days per small input video) and the quality is reduced. Many unrealistic artifacts usually appear. Moreover, these methods only handle very short low-resolution videos and are unscalable. We argue that training for days, for each input and each task, only for short low-resolution videos can not be considered as a real solution.

A new practical solution in the image domain was recently introduced by [16]. They observed that "good-old" patch-based methods are superior in many aspects to single-image GANs. They suggested to reconsider simple patch nearest neighbors again, but cast them as a generative model that when given a single image can generate many diverse outputs. This approach significantly outperforms the GAN based methods, as well as older classical patch-based methods. The quality of the images is much better and the runtime is reduced dramatically.

3. Method

Our goal is to make diverse video generation based on a single video possible. By "diverse" we mean 3 things: (i) diverse output samples from a single input video; (ii) diverse natural dynamics (i.e., not restricted to a specific training video dataset, or specific type of video dynamics); (iii) diverse set of applications within a single unified computational framework.

In order to capture both spatial and temporal information of a single video, VGPNN starts by building a spatio-temporal pyramid and operates coarse-to-fine to capture the internal statistics of the input video at multiple scales (Fig. 3). Formally, given a source video x, we construct a spatio-temporal pyramid $\{x_0, \ldots, x_N\}$, where $x_n = x{\downarrow_{r^n}}$ is a downscaled version of x by factor r^n, both in space and in time (with $r > 1$; in our current implementation $r = \frac{6}{5}$; downscaling via cubic interpolation in all 3 dimensions).

The output video is then generated in a progressive growing manner (as is common with generative models e.g., [17,23,36]). Namely, VGPNN is a sequence of **VPNN** layers, each operates in its own scale.

At the coarsest scale, the input to the first VPNN layer is an *initial coarse guess* of the output video. This is created by adding random Gaussian noise z_N to x_N (see Fig. 3). The noise z_N promotes high diversity in the generated output samples from the single input. The global structure (e.g., a head is above the body) and global motion (e.g., humans walk forward), is prompted by x_N, where such structure and motion can be captured by *small space-time* patches (in this work we used $(3 \times 7 \times 7)$ patches, where 3 is in the temporal dimension). Other choices of the initial coarse-scale guess lead to a variety of other applications (presented in Sec. 5).

The coarsest-scale y_N of the output video (output of the coarsest-scale VPNN layer) is generated by replacing each space-time patch of the initial coarse guess $x_N + z_N$ with its nearest neighbor patch from the corresponding coarse input x_N. The resulting patches are then folded to a video, by choosing at each space-time pixel location the median of all suggestions from neighboring patches.

At each subsequent scale, the input to the corresponding VPNN layer is the bicubically *upscaled* output of the previous VPNN layer ($y_{n+1}\uparrow$). The output y_n is then generated by replacing each space-time patch with its nearest neighbor patch from the corresponding input scale x_n (using same patch-size as before, now capturing finer details; see also QKV scheme below). This way, the output y_n in

each scale is similar in structure and in motion to the initial guess, but contains the same space-time patch statistics of the corresponding input scale x_n. Finally, the resulting patches are folded to a video.

To further improve the quality and sharpness of the generated output at each pyramid scale (y_n), we iterate several times through the current scale VPNN layer, each time using the current output y_n as input to the next iteration (similar to the EM-like approach employed in many patch-based works [6, 16, 39, 52]).

Noise initialization To better adjust this randomness to video data, we found that temporal consistency is best preserved in the generated output video when the noise is randomized for each *spatial* location, but is then *replicated through the temporal dimension*.

QKV scheme The straightforward way to compare two patches is by using their RGB values. While this works well in most cases, we found out that in several cases it is better to compare patches in other search spaces. To this end we use a *QKV scheme* (query, key, value) at each level (similar to GPNN [16]).

We denote the aforementioned two inputs to the VPNN layer as follows: the upscaled output of previous layer ($y_{n+1}\uparrow$) denoted as Q, and the corresponding level from the pyramid of the original video (x_n) denoted as V. We now define K as follows: each patch Q_i in Q search for its nearest neighbor K_j in K. Then, Q_i will be replaced with V_j (the patch in V corresponding to K_j).

This scheme helps in several cases. First, patches in the *upscaled* generated output from the previous scale are blurry. Seeking their nearest neighbors in x_n (whose patches are sharp) often results in improper matches. This is mitigated by using a "key video" K, created by saptio-temporally upscaling the previous-level video from the pyramid, $K = x_{n+1}\uparrow^r$, having similar degree of blur/degradation as in the query video Q. After finding its match in the similarly-degraded set K, each patch Q_i is then replaced with a patch V_j (corresponding to K_j), which is crisp and sharp (undegraded).

If keys and values were not separated, patches from blurry and motion-aliased video (Q) would look for nearest neighbors in a video which does not suffer from similar degradation (V). This may lead to a blurry and temporally-inconsistent result. Since temporal interpolation causes severe artifacts (such as motion aliasing), using the query-key-value scheme has a significant effect on our video results, even greater than on images. (note that this choice of K is done in the first iteration of each scale of VGPNN).

Second, this scheme is also used in other applications (see Sec. 5) to better utilize the temporal information in videos when matching between patches.

WeightedPatchMatch A key component in VGPNN is *WeightedPatchMatch*, an extension of PatchMatch [6] to handle globally-dependent information (weights) in its objective. The original PatchMatch algorithm provides an approximate solution to the nearest-neighbor field (NNF) between two sets. For natural images/videos, the large correlations between adjacent pixels allow for a quick convergence to a very good approximated solution.

WeightedPatchMatch uses the same propagation and random search steps as in the original PatchMatch (using the "jump flood" scheme [32]), and introduces weights to the distance metric. Let S be the source ("query") video, T be the target ("key") video, D the metric function between two patches and W the *weight* function. Let $\mathbf{p} = (t, x, y)$ be a position in the source video and f be the NNF. In each iteration, *WeightedPatchMatch* solves for:

$$f(\mathbf{p}) = \arg\min_{\mathbf{v}} W(\mathbf{p}+\mathbf{v}) \cdot D(S(\mathbf{p}), T(\mathbf{p}+\mathbf{v})) \quad (1)$$

Where $\mathbf{v} = (t', x', y')$ are possible NNF candidates, such as the NNF at the current location $f(t, x, y)$ or at a neighbor location $f(t, x-1, y)$ in the propagation step.

Implemented on GPU using PyTorch [30], our *WeightedPatchMatch* has time complexity of $O(n \times d)$ and $O(n)$ additional memory (where n is the video size and d is the patch size). Compared to GPNN, with time complexity of $O(n^2 \times d)$ and memory footprint $O(n \times d)$, our algorithm is significantly faster and requires far less memory. Run times comparison of VGPNN vs. GPNN is shown in Fig. 6.

Completeness score GPNN [16] introduced the normalized similarity score (Eq. 2) that encourages *visual completeness* while maintaining *visual coherence* * [39]. The similarity score between a query patch Q_i and a key patch K_j is defined as follows:

$$S(Q_i, K_j) := \frac{1}{\alpha + \min_\ell D(Q_\ell, K_j)} D(Q_i, K_j) \ , \quad (2)$$

where $D = MSE$, and α controls the degree of completeness (smaller α encourages more completeness). S is essentially a weighted version of D, whose weights depends globally on K and Q.

In its original form, PatchMatch supports complicated metrics in its nearest neighbor search but does not receive additional globally-dependent information as input (e.g., weights). Therefore, it requires modification in order to minimize the objective in Eq. 2 which uses knowledge of

*"Visual completeness" is obtained when all the patches in the input can be found in the generated output (hence no critical information was lost). "Visual coherence" is obtained when the generated output contains only patches from the input (hence no new undesired artifacts were introduced in the output). While coherence is always desired, completeness is required only in some applications (e.g., in visual summarization).

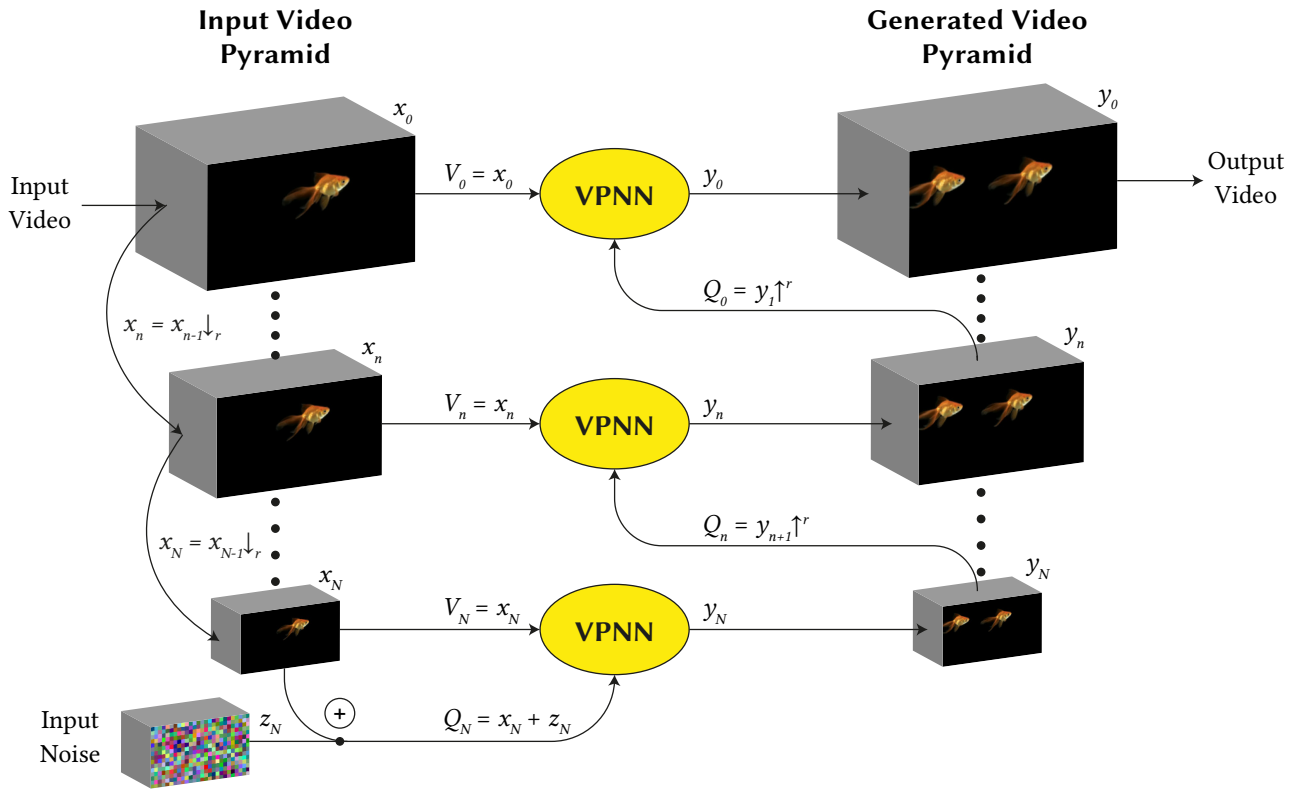

Figure 3. **VGPNN Architecture:** Given a single source video x_0, a spatio-temporal pyramid is constructed and an output video y_0 is generated coarse-to-fine. At each scale, VPNN module (see Fig. 4) is applied to transfer an initial guess Q_n to the output y_n which shares the same space-time patch distribution as the input x_n. At the coarsest scale, noise is injected to induce spatial and temporal randomness.

global information from all patches (the similarity score from Q_i to K_j depends on all other query patches). To this end we use our *WeightedPatchMatch*, using per-key weights $W_j = {}^{1}/_{(\alpha+\min_\ell D(Q_\ell,K_j))}$.

Note that W_j includes a minimization term. Computing it exhaustively, as done in GPNN, would prohibit the extention to videos. Since this term is essentially the nearest neighbor of K_j in Q, we solve the NNF using PatchMatch, thus obtaining an approximation of W_j (as stated in Eq. 2). As apparent from our results, we do not suffer loss in quality or lack of completeness, which might have arisen from the approximated solution.

The complete flow of VPNN is described in Fig. 4.

4. Experimental Results

We evaluate and compare the performance of our main application – random/diverse video generation from a single input video. Figs. 1 and 2 illustrate that our method can take a single video and generate new diverse videos sharing the same space-time patch distribution. One can see how from a single video of a dancer, our method is able to generate diverse outputs both spatially (number of dancers and their locations) and temporally (generated dancers are not synced). Or, how from a single video of traffic junction (second video in Fig. 2), VGPNN generates new videos such that the entire scene changes, yet the output is visually pleasing (both in space and in time). Our efficient method is capable of generating diverse videos of high spatial resolution and of significant temporal extent within *minutes*. **Please refer to the supplementary material to view the full resolution videos (360p, and an example of Full-HD), and many more examples.**

Comparison to other video generation methods: We further compare our method to recently published methods of diverse video generation from single video: HP-VAE-GAN [17] and SinGAN-GIF [3]. We show that our results are both qualitatively and quantitatively superior while reducing the runtime by a factor of $\sim 35,000$ (from 8 days training on one video to 18 seconds for new generated video). Since SinGAN-GIF did not make their code available, and the training time of HP-VAE-GAN for a single video (of size $13 \times 144 \times 256$) is roughly 8 days, we are only able to compare to videos published by these methods. HP-VAE-GAN dataset comprises of 10 input videos with 13 frames each, and of spatial resolution of 144×256 pixels. SinGAN-GIF dataset has 5 input videos with maximal

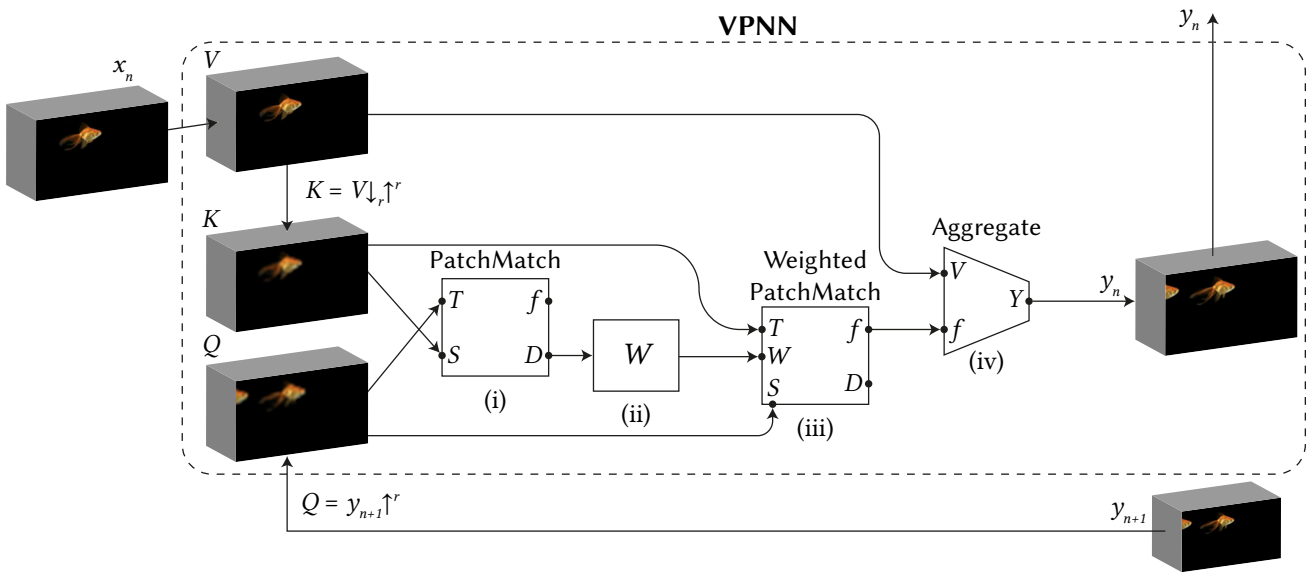

Figure 4. **VPNN Module:** At each scale, the VPNN modules generates an output y_n, which is similar in structure to the upscaled output from coarser scale, $y_{n+1}\uparrow^r$, and has similar patch distribution as x_n. Using PatchMatch, and our per-key weighted version Weighted-PatchMatch, VPNN is able to construct y_n efficiently, and maintain visual completeness if necessary, even for extremely large inputs. PatchMatch receives a source video S and a target video T and returns the nearest neighbor field (NNF) f (location of nearest target patch for each source patch) and the distance map D (distance between each source patch and its nearest neighbor target patch). WeightedPatch-Match receives, in addition, W, the per-key weights. The Aggregate component receives a video V and an NNF f and reconstruct and output video Y.

resolution of 168×298 pixels and 8-16 frames.

Qualitative comparison: A side-by-side comparison of random generation of videos can be viewed in the supplementary material. Our generated samples are more spatially and temporally coherent as well as of higher visual quality. Generating new videos using the space-time patches of the original input video, rather than regressing output RGB values, give rise to high quality outputs. Fig. 5 highlights this important feature of our method by comparing the visual quality of representative generated frames of our method to frames generated by HP-VAE-GAN and SinGAN-GIF.

Quantitative comparison: On the one hand, randomly generated samples should resemble (both in appearance and in dynamics) the input video. On the other hand, these samples should be diverse. Therefore, we measure patch statistics similarity between a generated output and the input using the Single-Video-FID (SVFID) proposed by [17]: it computes the Fréchet distance between the statistics of the generated output and the original video using pre-computed C3D features [40]. Lower SVFID is better. Additionally, we measure how diverse our different samples are, using an adaptation of the *diversity* index proposed by [36] for images: Given a source video, standard deviation of each video "pixel" (3D RGB element in the video, converted to grayscale) is calculated across all generations, and then averaged across all pixels. Finally, it is normalized by the std of pixels in the input video.

For each input video in HP-VAE-GAN and SinGAN-GIF datasets we generated the same number of random sample as publicly available (10 generations for each video in HP-VAE-GAN dataset, and 6 generations for each video in SinGAN-GIF dataset), and compared their SVFID and diversity. You can find all videos (inputs and generated ones) in the supplementary material. Table 1 reports SVFID and diversity scores of our generated videos compared to the random samples generated by either HP-VAE-GAN or SinGAN-GIF. Our generated samples bear more substantial similarity to the input videos (indicated by lower SVFID) while exhibiting significant diversity.

User study: We further conducted a user study evaluation using Amazon Mechanical Turk (AMT). For each dataset we asked 100 turkers to judge between our and the other method's generated sample (both samples generated from the same input video), which sample is better in terms of sharpness, natural looking and coherence. The 5^{th} column in Table 1 shows the percentage of users who favored one method over the other. While there is a significant user preference towards our method on the videos generated from HP-VAE-GAN dataset, the results are not that clear-cut for the SinGAN-GIF dataset. This might be due to the some-

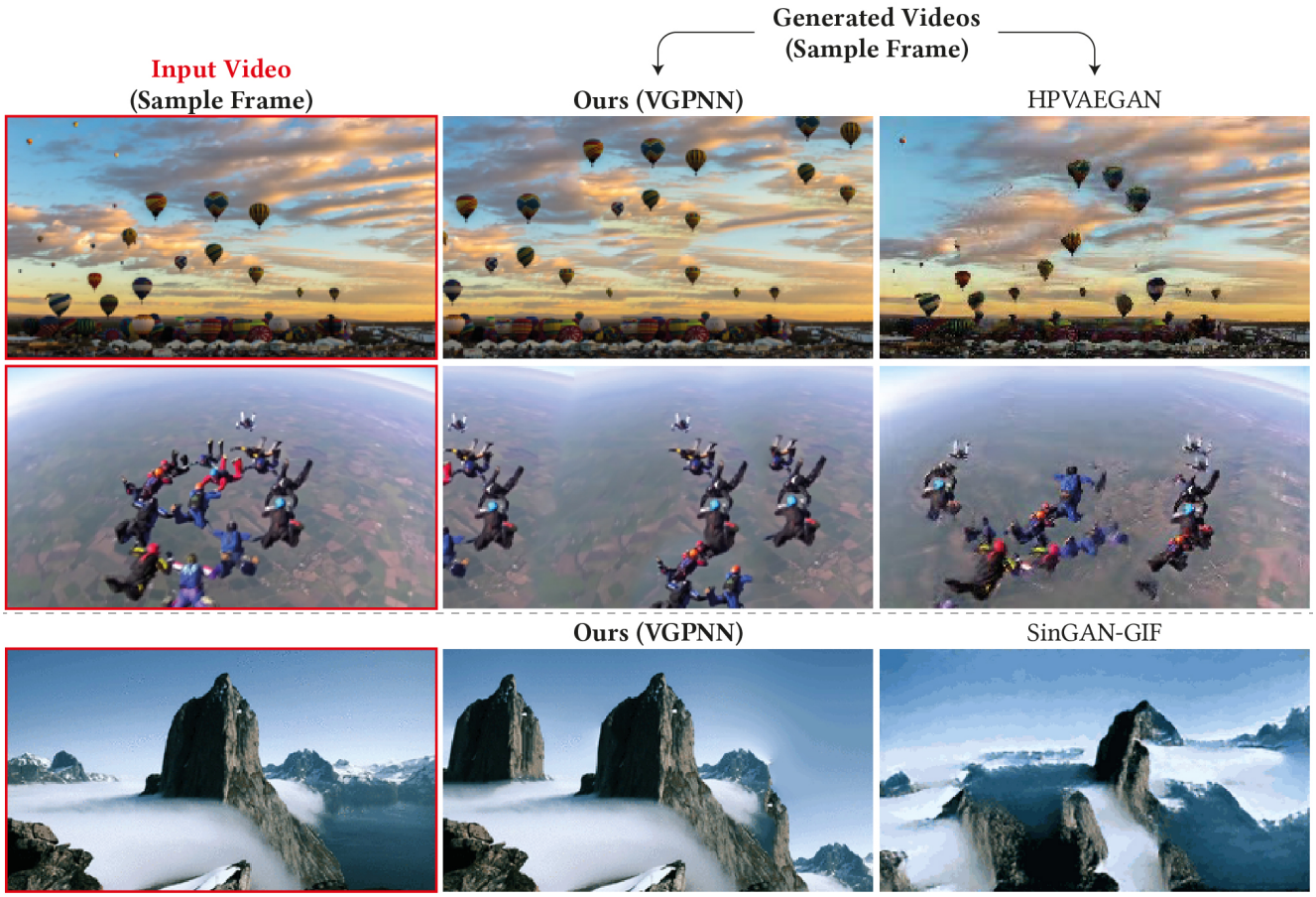

Figure 5. **Comparing the visual quality of generated frames:** Our generated frames (blue) are not only sharper than those of HP-VAE-GAN or SinGAN-GIF, but also exhibit more coherent and plausible arrangements of the scenes. Note the artefacts in the generated frames of both HP-VAE-GAN and SinGAN-GIF. Please refer to the supplementary material for the actual videos and more comparisons.

what restricted nature of the videos in that particular dataset.

Reducing running times: As discussed in sec. 3, VGPNN is significantly more efficient than GPNN due to the use of efficient WeightedPatchMatch algorithm for nearest neighbors search, rather than exhaustive search as in GPNN. This change has decreased the memory footprint significantly, making the generation of high-resolution videos (including Full-HD 1080p) possible, while significantly shortening the runtime. A comparison of the generation time of VGPNN, GPNN* and HP-VAE-GAN can be seen in Fig. 6.

5. Applications

Generating random samples of a single video is beneficial not only as an academic concept but also as a practical goal. This section demonstrates several applications whose goal is to manipulate a given video and uses VGPNN as a video-specific prior.

Spatial retargeting: The goal of video spatial retargeting is to change the spatial dimensions of a video without distorting its visual contents (e.g., fit a portrait video to a wide screen display). Video spatial retargeting can be performed in a very similar manner to video generation, but instead of injecting noise at the coarsest scale, Q_N, we gradually resize Q_n towards the target size. For an input video with input spatial dimensions of $h \times w$ and target size of $h \cdot s_h \times w \cdot s_w$ we upscale Q_n to scale $s_h^{(N-n)/N} \times s_w^{(N-n)/N}$ of the corresponding K_n. Note that the scales of V_n are unchanged, hence no distortion is introduced to the patches reconstructing the retargeted video. As can be seen in Fig. 7, the result preserves the original size and aspect ratio of objects in the video while keeping the overall appearance coherent even though the aspect ratio is significantly different. The dynamics and motions in the videos are also preserved, as can be seen in the videos in the supplementary material. For instance, the balloons are not "squashed" but are rather packed more compactly in the sky, more members were added to the choir, instead of stretching them. Never-

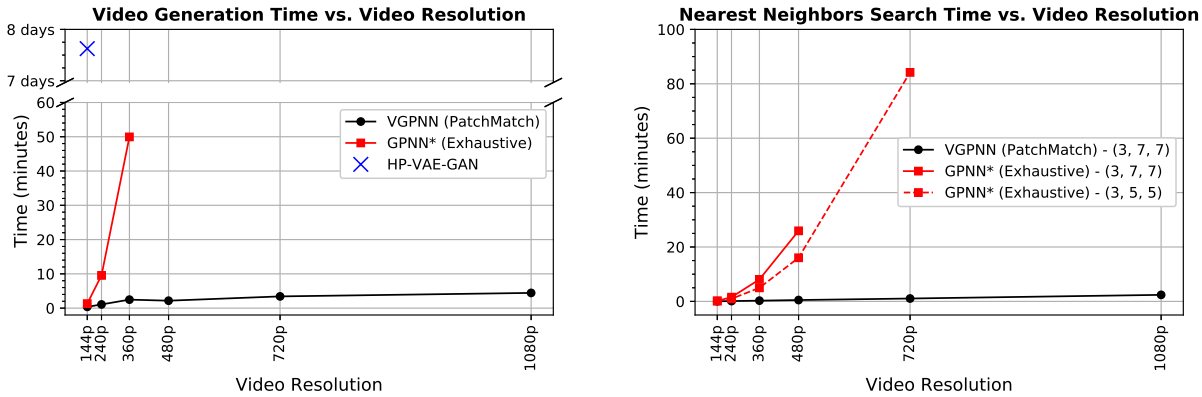

Figure 6. **Left:** Video generation time of VGPNN vs GPNN* (a trivial extension of GPNN [16] from 2D to 3D video data) and HP-VAE-GAN [17] on Nvidia Quadro RTX 8000 (48GB). The x-axis scale is the number of pixels in a 13 frames video. Note that the y-axis is cut, and that HP-VAE-GAN takes ~ 7.5 days to complete (compared to minutes in VGPNN) on the same resolution. **Right:** A detailed comparison between nearest neighbors search time with PatchMatch (VGPNN) and with exhaustive search (GPNN*). GPNN* exceeds GPU memory at medium resolution (480p) with the original patch size $(3, 7, 7)$. The dashed line with smaller patch size $(3, 5, 5)$ is intended to show the quadratic trend with more data points.

Dataset	**Method**	**SVFID** $\downarrow$ [17]	**Diversity** [36]	**Head-on competition** [%] $\uparrow$ (User study)	**Runtime** $\downarrow$
HP-VAE-GAN	HP-VAE-GAN	0.0081	0.4049	32.16±1.77	7.625 days (658980 secs)
	VGPNN (Ours)	**0.0072**	0.4482	**67.84**±1.77	**18** secs
SinGAN-GIF	SinGAN-GIF	0.0119	0.8593	49.42±3.27	Unpublished
	VGPNN (Ours)	**0.0058**	0.5955	**50.57**± 3.27	**10** secs

Table 1. **Quantitative Evaluation:** We compared quality (SVFID) and diversity of randomly generated samples on HP-VAE-GAN and SinGAN-GIF datasets. Our method produces diverse samples that are resemble better (low SVFID) the input video. Furthermore, in a user study, users tend to score our generated videos as better compared to HP-VAE-GAN or SinGAN-GIF. See text for details.

theless, the motion of the balloons or the sway of the choir members are preserved. Unfortunately, we are unable to compare our results to those of [33, 39, 53]: these methods did not make their implementation available, and re-implementing requires elaborate engineering and tweaking of hyper-parameters.

Temporal retargeting: Similar to spatial retargeting, one can change the *temporal* extent of a video to create a temporal summary or an extension of the original video. Fig. 1 illustrates this application. Again, this is done by gradually changing the number of frames in Q_n, while leaving V_n unchanged. Several examples of video summarization can be found in the supplementary material. Note how the *speeds* of the different actions are maintained similar to the original long video, however a more compact representation of the action is generated when creating a summary. Moreover, we can, in a similar manner, extend the temporal duration of a video creating longer dynamics while preserving the speed of the individual actions. See examples in the supplementary material, where, e.g., the choreography of the ballet dancer is longer, but the pace of the dance motions remains the same.

Video conditional inpainting: Similar to conditional image inpainting [16], in this task an input video with some occluded space-time volume is received, and the missing part should be completed based on crude color cues placed by the user in the occluded space. These crude cues are used to steer the way the missing part is filled by VGPNN. This video with the masked space-time volume and the uniform color cues is the input x. The initial guess y_{N+1} is set to be a downscaled version of x by r^N. The number of pyramid levels N is chosen such that the occluded part in the downscaled video is roughly the size of a single patch. At the coarsest level the masked part is coherently reconstructed using other space-time patches adhering to the coarse color cue placed in the occluded region. In finer levels, details and dynamic elements are added. Fig. 8 shows how choosing different color cues results with VGPNN completing different elements in those regions. For instance, a blue mask steers the completion to a player from Barcelona while a

Figure 7. **Spatial Video Retargeting:** the input video (red) is retargeted to different sizes and aspect ratios, while maintaining object sizes, aspect ratios and composition. Note that when the frame was extended in width, more singers were added to each row, and when the frame was extended in height, additional rows of singers were added to the choir. Full videos and additional examples can be found in the supplementary material.

white mask triggers VGPNN to add a player to Real. The input and inpainted videos can be seen in the supplementary material.

Video analogies: Inspired by Image Analogies [19] and [7] we propose the task of video analogies, where we transfer the dynamic behavior from a content video x^A using the dynamic elements of a style video x^B. This is achieved by creating two spatio-temporal pyramids $x^A_{0..N}$ and $x^B_{0..N}$, setting $Q_N = x^A_N$, and at each step $(K_n, V_n) = (x^B_n \downarrow\uparrow, x^B_n)$. This way we are able to capture the global dynamic behavior depicted by the coarse spatio-temporal scale x^A_N of the content video A, while taking the fine details from the patches of B. In Fig. 1 we show an example of video analogies between two videos, with two results: one takes style from A and content from B and vice versa.

Video analogies can also be extended to "sketch to video": by providing a dynamic "sketch" of an action we can use VGPNN to generate an analogy between the sketch and a style video. Fig. 9 illustrates this application, the actual sketches and resulting videos can be found in the supplementary material. Note how the building blocks, along with their local motions, were taken from the style video, while the overall dynamic behavior of the sketch input is maintained.

Limitations: VGPNN is only concerned with video patches at multiple scales, and therefore it lacks any "global understanding" of the scene or any high-level semantic information. Hence, when a video contains global camera motion of a rigid scene, as in Fig. 10 (left), VGPNN often generates local deformations to the rigid scene (highlighted by red and green). Note that despite the high visual quality of every generated frame on itself, when viewing the generated *dynamics* the result is perceived as unrealistic. Additionally, VGPNN has no high-level semantic information on the contents of the videos. Consequently, it often fails to generate plausible complex semantic objects such as dancing people (Fig. 10 right).

6. Conclusion

In this work, we demonstrated how using simple patch-based methods allows us to perform random video generation from a single video and carry out challenging manipulations such as spatio-temporal retargeting, inpainting, and analogies. Basing our method on learning-free efficient mechanisms allows us to perform these tasks, for the first time, in close to real-time. Moreover, generating the output video by directly combining space-time patches from the input video, as opposed to indirectly regressing RGB values, substantially increases the visual quality of our outputs. This game-changing reduction in run time and the improvement of output visual quality turn these applications from mere academic concepts to valuable tools that can actually be used by practitioners.

7. Acknowledgements

This project received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement

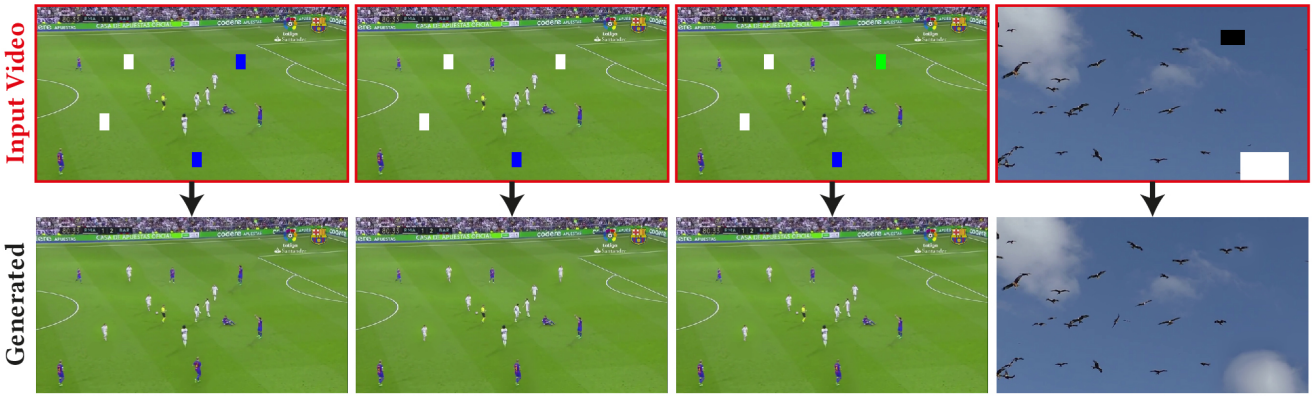

Figure 8. **Conditional Video Inpainting:** the input video (red) includes regions masked by the user to be inpainted by VGPNN. The completion output is conditioned on the crude color cue provided, for each masked region, by the user. Blue cues result with Barcelona players, while white ones steers VGPNN to inpaint with Real players. Note that for the birds video (right) white cue results with additional clouds in the sky. This completion persists through the video dynamics, please see full videos in the supplementary material.

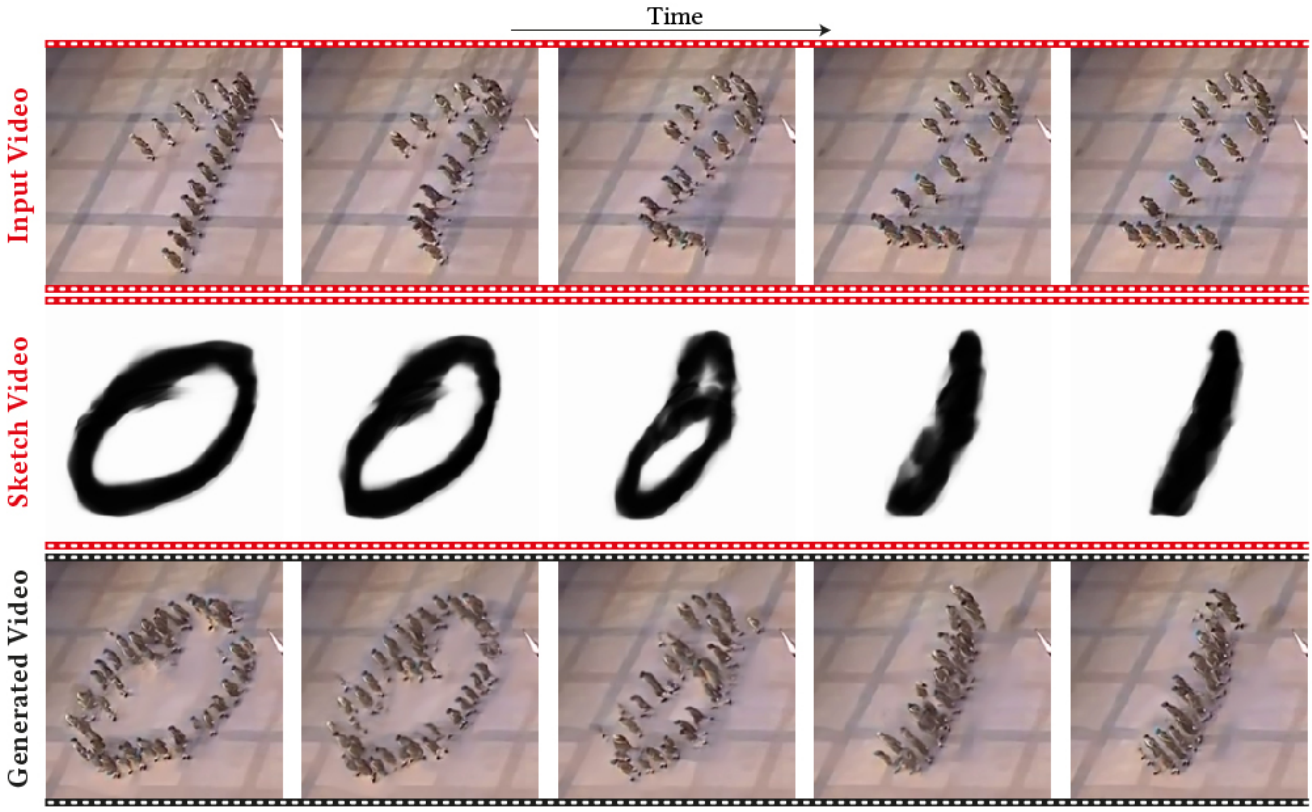

Figure 9. **Sketch to Video:** Given an input video of parading soldiers (top) and a crude binary sketch video of morphing MNIST digits (middle) VGPNN is able to generate a new video with the soldiers parading to form the sketched shape (bottom). Full videos and additional examples can be found in the supplementary material.

No 788535), from the D. Dan and Betty Kahn Foundation, and from the Israel Science Foundation (grant 2303/20). Dr. Bagon is a Robin Chemers Neustein AI Fellow.

References

[1] Sandra Aigner and Marco Körner. Futuregan: Anticipating the future frames of video sequences using spatio-temporal 3d convolutions in progressively growing gans.

Figure 10. **Limitations & Failures:** VGPNN has no notion of global 3D rigidity or semantic meaning of objects/actions. Therefore, it may sometimes introduce non-rigid deformations to rigid scenes undergoing global camera motion (left). Each frame in the generated output looks very good on its own, and the motion varies smoothly and pleasantly over time. However, while all space-time patches are faithful to the original video, their overall composition may sometimes produce unrealistic (non-rigid) scenes undergoing 3D camera motion. Furthermore, VGPNN might sometimes fall into undesired local minima and create semantically meaningless moving object parts and undesired visual artefacts (right). Example videos of such failure cases can be found in the supplementary material.

arXiv preprint arXiv:1810.01325, 2018. 2

[2] Emre Aksan and Otmar Hilliges. Stcn: Stochastic temporal convolutional networks. *arXiv preprint arXiv:1902.06568*, 2019. 4

[3] Rajat Arora and Yong Jae Lee. Singan-gif: Learning a generative video model from a single gif. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1310–1319, 2021. 2, 4, 6

[4] Mohammad Babaeizadeh, Chelsea Finn, Dumitru Erhan, Roy H Campbell, and Sergey Levine. Stochastic variational video prediction. *arXiv preprint arXiv:1710.11252*, 2017. 4

[5] Nicolas Ballas, Li Yao, Chris Pal, and Aaron Courville. Delving deeper into convolutional networks for learning video representations. *arXiv preprint arXiv:1511.06432*, 2015. 4

[6] Connelly Barnes, Eli Shechtman, Adam Finkelstein, and Dan B Goldman. Patchmatch: A randomized correspondence algorithm for structural image editing. *ACM Trans. Graph.*, 28(3):24, 2009. 2, 5

[7] Sagie Benaim, Ron Mokady, Amit Bermano, and L Wolf. Structural analogy from a single image pair. In *Computer Graphics Forum*, volume 40, pages 249–265. Wiley Online Library, 2021. 4, 10

[8] Urs Bergmann, Nikolay Jetchev, and Roland Vollgraf. Learning texture manifolds with the periodic spatial GAN. *arXiv preprint arXiv:1705.06566*, 2017. 4

[9] Jinshu Chen, Qihui Xu, Qi Kang, and MengChu Zhou. Mogan: Morphologic-structure-aware generative learning from a single image. *arXiv preprint arXiv:2103.02997*, 2021. 4

[10] Emily Denton and Rob Fergus. Stochastic video generation with a learned prior. In *International Conference on Machine Learning*, pages 1174–1183. PMLR, 2018. 2, 4

[11] Alexei A Efros and Thomas K Leung. Texture synthesis by non-parametric sampling. In *Proceedings of the seventh IEEE international conference on computer vision*, volume 2, pages 1033–1038. IEEE, 1999. 4

[12] Jean-Yves Franceschi, Edouard Delasalles, Mickaël Chen, Sylvain Lamprier, and Patrick Gallinari. Stochastic latent residual video prediction. In *International Conference on Machine Learning*, pages 3233–3246. PMLR, 2020. 4

[13] Yosef Gandelsman, Assaf Shocher, and Michal Irani. " double-dip": Unsupervised image decomposition via coupled deep-image-priors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11026–11035, 2019. 4

[14] Pallabi Ghosh, Vibhav Vineet, Larry S Davis, Abhinav Shrivastava, Sudipta Sinha, and Neel Joshi. Depth completion using a view-constrained deep prior. In *International Conference on 3D Vision (3DV)*, pages 723–733. IEEE, 2020. 4

[15] Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *arXiv preprint arXiv:1406.2661*, 2014. 2

[16] Niv Granot, Ben Feinstein, Assaf Shocher, Shai Bagon, and Michal Irani. Drop the gan: In defense of patches nearest neighbors as single image generative models. *arXiv preprint arXiv:2103.15545*, 2021. 1, 2, 4, 5, 9

[17] Shir Gur, Sagie Benaim, and Lior Wolf. Hierarchical patch vae-gan: Generating diverse videos from a single sample. *arXiv preprint arXiv:2006.12226*, 2020. 2, 4, 6, 7, 9

[18] Michael Gygli, Helmut Grabner, Hayko Riemenschneider, and Luc Van Gool. Creating summaries from user videos. In *European conference on computer vision*, pages 505–520. Springer, 2014. 4

[19] Aaron Hertzmann, Charles E Jacobs, Nuria Oliver, Brian Curless, and David H Salesin. Image analogies. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 327–340, 2001. 10

[20] Tobias Hinz, Matthew Fisher, Oliver Wang, and Stefan Wermter. Improved techniques for training single-image gans. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1300–1309, 2021. 4

[21] Aleksander Holynski, Brian Curless, Steven M Seitz, and Richard Szeliski. Animating pictures with eulerian motion fields. *arXiv preprint arXiv:2011.15128*, 2020. 4

[22] Nikolay Jetchev, Urs Bergmann, and Roland Vollgraf. Texture synthesis with spatial generative adversarial networks. *arXiv preprint arXiv:1611.08207*, 2016. 4

[23] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017. 4

[24] Philipp Krähenbühl, Manuel Lang, Alexander Hornung, and Markus Gross. A system for retargeting of streaming video. In *ACM SIGGRAPH Asia 2009 papers*, 2009. 4

[25] Alex X Lee, Richard Zhang, Frederik Ebert, Pieter Abbeel, Chelsea Finn, and Sergey Levine. Stochastic adversarial video prediction. *arXiv preprint arXiv:1804.01523*, 2018. 4

[26] Jianxin Lin, Yingxue Pang, Yingce Xia, Zhibo Chen, and Jiebo Luo. TuiGAN: Learning versatile image-to-image translation with two unpaired images. In *European Conference on Computer Vision*, pages 18–35. Springer, 2020. 4

[27] Arun Mallya, Ting-Chun Wang, Karan Sapra, and Ming-Yu Liu. World-consistent video-to-video synthesis. *arXiv preprint arXiv:2007.08509*, 2020. 2

[28] Indra Deep Mastan and Shanmuganathan Raman. Dcil: Deep contextual internal learning for image restoration and image retargeting. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2366–2375, 2020. 4

[29] Indra Deep Mastan and Shanmuganathan Raman. DeepCFL: Deep contextual features learning from a single image. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2897–2906, 2021. 4

[30] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. 5

[31] Alex Rav-Acha, Yael Pritch, and Shmuel Peleg. Making a long video short: Dynamic video synopsis. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 1, pages 435–441. IEEE, 2006. 4

[32] Guodong Rong and Tiow-Seng Tan. Jump flooding in gpu with applications to voronoi diagram and distance transform. In *Proceedings of the 2006 symposium on Interactive 3D graphics and games*, pages 109–116, 2006. 5

[33] Michael Rubinstein, Ariel Shamir, and Shai Avidan. Improved seam carving for video retargeting. *ACM transactions on graphics (TOG)*, 27(3):1–9, 2008. 4, 9

[34] Masaki Saito, Eiichi Matsumoto, and Shunta Saito. Temporal generative adversarial nets with singular value clipping. In *Proceedings of the IEEE international conference on computer vision*, pages 2830–2839, 2017. 4

[35] Arno Schödl, Richard Szeliski, David H Salesin, and Irfan Essa. Video textures. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 489–498, 2000. 4

[36] Tamar Rott Shaham, Tali Dekel, and Tomer Michaeli. Singan: Learning a generative model from a single natural image. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4570–4580, 2019. 4, 7, 9

[37] 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. 4

[38] Assaf Shocher, Nadav Cohen, and Michal Irani. "zero-shot" super-resolution using deep internal learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3118–3126, 2018. 4

[39] Denis Simakov, Yaron Caspi, Eli Shechtman, and Michal Irani. Summarizing visual data using bidirectional similarity. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2008. 4, 5, 9

[40] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 4489–4497, 2015. 7

[41] Sergey Tulyakov, Ming-Yu Liu, Xiaodong Yang, and Jan Kautz. Mocogan: Decomposing motion and content for video generation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1526–1535, 2018. 2

[42] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Deep image prior. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9446–9454, 2018. 4

[43] Ruben Villegas, Dumitru Erhan, Honglak Lee, et al. Hierarchical long-term video prediction without supervision. In *International Conference on Machine Learning*, pages 6038–6046. PMLR, 2018. 4

[44] Ruben Villegas, Arkanath Pathak, Harini Kannan, Dumitru Erhan, Quoc V Le, and Honglak Lee. High fidelity video prediction with large stochastic recurrent neural networks. *arXiv preprint arXiv:1911.01655*, 2019. 4

[45] Ruben Villegas, Jimei Yang, Seunghoon Hong, Xunyu Lin, and Honglak Lee. Decomposing motion and content for natural video sequence prediction. *arXiv preprint arXiv:1706.08033*, 2017. 4

[46] Yael Vinker, Eliahu Horwitz, Nir Zabari, and Yedid Hoshen. Deep single image manipulation. *arXiv preprint arXiv:2007.01289*, 2020. 4

[47] Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. Generating videos with scene dynamics. *arXiv preprint arXiv:1609.02612*, 2016. 2

[48] Ting-Chun Wang, Ming-Yu Liu, Andrew Tao, Guilin Liu, Jan Kautz, and Bryan Catanzaro. Few-shot video-to-video synthesis. *arXiv preprint arXiv:1910.12713*, 2019. 2

[49] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Guilin Liu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. Video-to-video synthesis. *arXiv preprint arXiv:1808.06601*, 2018. 2

[50] Yaohui Wang, Piotr Bilinski, Francois Bremond, and Antitza Dantcheva. Imaginator: Conditional spatio-temporal gan for video generation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1160–1169, 2020. 2

[51] Yaohui Wang, Francois Bremond, and Antitza Dantcheva. Inmodegan: Interpretable motion decomposition generative adversarial network for video generation. *arXiv preprint arXiv:2101.03049*, 2021. 2

[52] Yonatan Wexler, Eli Shechtman, and Michal Irani. Space-time video completion. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, volume 1, pages I–I. IEEE, 2004. 4, 5

[53] Lior Wolf, Moshe Guttmann, and Daniel Cohen-Or. Non-homogeneous content-driven video-retargeting. In *Proceedings of the Eleventh IEEE International Conference on Computer Vision (ICCV)*, 2007. 4, 9

[54] Haotian Zhang, Long Mai, Ning Xu, Zhaowen Wang, John Collomosse, and Hailin Jin. An internal learning approach to video inpainting. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2720–2729, 2019. 4

[55] Lin Zhang, Lijun Zhang, Xiao Liu, Ying Shen, Shaoming Zhang, and Shengjie Zhao. Zero-shot restoration of back-lit images using deep internal learning. In *Proceedings of the 27th ACM International Conference on Multimedia*, pages 1623–1631, 2019. 4

[56] Xin Zhao, Lin Wang, Jifeng Guo, Bo Yang, Junteng Zheng, and Fanqi Li. Solid texture synthesis using generative adversarial networks. *arXiv preprint arXiv:2102.03973*, 2021. 4

[57] 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. 4

[58] Liad Pollak Zuckerman, Eyal Naor, George Pisha, Shai Bagon, and Michal Irani. Across scales and across dimensions: Temporal super-resolution using deep internal learning. In *European Conference on Computer Vision*, pages 52–68. Springer, 2020. 4