Drop the GAN[1]: In Defense of Patches Nearest Neighbors as Single Image Generative Models

Niv Granot* Ben Feinstein* Assaf Shocher* Shai Bagon† Michal Irani*
*Dept. of Computer Science and Applied Math, The Weizmann Institute of Science †Weizmann Artificial Intelligence Center (WAIC)

Project Website: http://www.wisdom.weizmann.ac.il/~vision/gpnn/

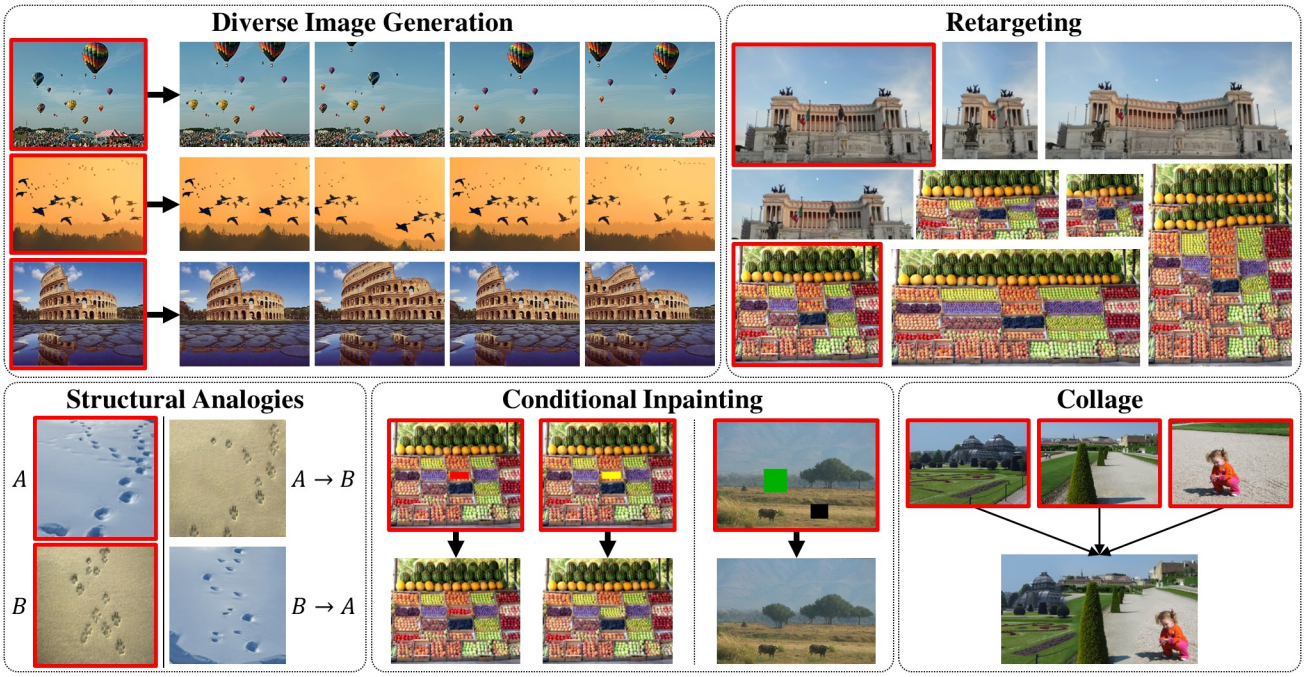

Figure 1. *Our simple unified framework covers a broad spectrum of single-image generative tasks, that usually require hours of training per image for GANs. Using patch nearest neighbors and a single source image, we can perform these tasks in a few seconds and with higher quality. We show here results obtained with our method for pivotal examples shown in SinGAN [27], InGAN [28], Structural analogies [3] and Bidirectional Similarity [30]. Additionally, we introduce novel applications such as Conditional-Inpainting. Input images marked in red.*

Abstract

Image manipulation dates back long before the deep learning era. The classical prevailing approaches were based on maximizing patch similarity between the input and generated output. Recently, single-image GANs were introduced as a superior and more sophisticated solution to image manipulation tasks. Moreover, they offered the opportunity not only to manipulate a given image, but also to generate a large and diverse set of different outputs from a single natural image. This gave rise to new tasks, which are considered "GAN-only". However, despite their impressiveness, single-image GANs require long training time (usually hours) for each image and each task and often suffer from visual artifacts. In this paper we revisit the classical patch-based methods, and show that – unlike previously believed – classical methods can be adapted to tackle these novel "GAN-only" tasks. Moreover, they do so ***better and faster than single-image GAN-based methods****. More specifically, we show that: (i) by introducing slight modifications, classical patch-based methods are able to unconditionally generate diverse images based on a single natural image; (ii) the generated output visual quality exceeds that of single-image GANs by a large margin (confirmed both quantitatively and qualitatively); (iii) they are orders of magnitude faster (runtime reduced from hours to seconds).*[2]

1. Introduction

Single-image generative models perform image synthesis and manipulation by capturing the patch distribution of a single image. Prior to the Deep-Learning revolution, the classical prevailing methods were based on optimizing the similarity of small patches between the input image and the generated output image. These *unsupervised* patch-based methods (e.g., [2, 6, 7, 25, 26, 30, 35]) gave rise to a wide variety of remarkable image synthesis and manipulation tasks, including image completion, texture synthesis, image summarization/retargeting, collages, image reshuffling, and more. Specifically, the Bidirectional similarity approach [2, 30] encourages the output image to contain only patches from the input image ("Visual Coherence"), and vice versa, the input should contain only patches from the output ("Visual Completeness"). Hence, no new artifacts are introduced in the output image and no critical information is lost either.

[1]While we could not resist this wordplay, we do acknowledge that single-image GANs have a few capabilities which cannot be realized by simple patch nearest-neighbors. We discuss these pros (and cons) in detail in Sec. 5. No GANs were harmed in the preparation of this paper... ☺.

[2]This project received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 788535), and the Carolito Stiftung. Dr Bagon is a Robin Chemers Neustein AI Fellow.

Recently, deep *Single-Image Generative Models* took the field of image manipulation by a storm. These models are a natural extension of "Deep Internal Learning" [9,11,29,32,37,38,43]. They train a fully convolutional GAN on a single input image, in a coarse-to-fine fashion, and thus capture its patch distribution at various scales. Then, the generator produces new images with similar patch distribution. They were initially used for "classical" tasks, such as image retargeting [28] and texture synthesis [4, 17, 39, 41]. Later, following externally trained GANs (e.g. [13]), they turned to solve new "GAN-only" (actually, "DL-only") generative tasks. SinGAN [27] showed the ability to unconditionally generate a large diversity of plausible images, based only on a single *natural* input image. Many other works followed SinGAN [3,5,14,15,21–23,34]. Most of these works solved tasks that were never considered with classical patch-based methods. Thus we refer to these tasks as "GAN-only".

Despite their remarkable capabilities, single-image GANs come with several heavy penalties compared to their simpler classical patch-based counterparts: (i) They require long training time (usually hours) for each input image and each task (as opposed to fast patch-based methods [2]). (ii) They are prone to optimization issues such as mode collapse. (iii) They often produce poorer visual quality than the classical patch-based methods (i.e., lack of "Coherence" [30]).

In this paper we show that, with some proper modern adaptations, "good-old" patch-based methods can surprisingly solve most of these previously considered "GAN-only" tasks. In fact, while the course of history has taken the research community from classical patch-based methods to powerful GANs, we claim that classical methods are superior to single-image GANs in many aspects. We demonstrate this claim through an adapted patch-based method (GPNN - Generative Patch Nearest Neighbors), that generates diverse images with comparable or better visual quality, and orders of magnitude faster than GANs (Sec. 3).

More specifically, GPNN emulates SinGAN, but "drops the GAN" and replaces it with a simple patch-nearest-neighbors (PNN) module. This results in a simple *generative* patch-based algorithm, which is a natural extension of classical methods. Unlike the older patch-based methods, GPNN exhibits the non-deterministic nature of GANs, their large diversity of possible outputs, and the ability to solve new generative tasks. Unlike GAN-based methods, GPNN enjoys classical methods benefits: it does not require any training, thus is very fast ($\times 10^3$-$\times 10^4$ faster than GANs). Moreover, it uses only *real* patches from the input image, hence produces results with higher visual quality than single-image GANs.

This modern adaptation of classical-methods is able to perform both the new generative "GAN-only" tasks, as well as the old classical tasks, all in a single unified framework, and with high quality results (Sec. 4). We demonstrate a wide range of applications: first and foremost *diverse image generation from a single natural image*, but also image editing, image retargeting, structural analogies, image collages, and a newly introduced task of "conditional inpainting". Finally, we analyze and characterize the pros and cons of these two approaches (patch nearest-neighbors vs. single-image GANs) in Sec. 5. We hope GPNN will serve as a new baseline for single image generation/manipulation tasks. Our contributions are therefore several fold:

- We show that "good old" patch nearest neighbor approaches, when cast correctly, can solve tasks which were previously considered "GAN-only" (such as unconditional and diverse image generation). We introduce such a casting – GPNN, a generative patch-based algorithm, which provides a unified framework for a large variety of tasks.
- We show that such a casting of classical methods produces random novel images with *higher quality* and *much faster* (3 orders of magnitude) than single image GANs.
- We analyze and discuss the *inherent pros & cons* of the modern single image GAN-based approaches vs. classical patch-based approaches. We further experimentally examine an existing hypothesis that GANs are "fancy nearest neighbor extractors". (Spoiler: our conclusion is "No").

2. GPNN: Modern Adaptation of Classical Methods

Our goal is to efficiently adapt classical patch-based methods as diverse single-image generative models. To this end, we consider principles from both Bidirectional Similarity [30] and SinGAN [27], to get the best of both worlds.

Similarly to both, and to many other works (e.g. [7, 26, 28, 35]), we follow a coarse-to-fine strategy in the generation process (Fig. 2(a)), i.e. at each scale n:

$$y_n = G(x_n, \tilde{y}_{n+1}) \tag{1}$$

where x_n is the input image x, downsampled by r^n (for $r>1$). $\tilde{y}_{n+1}$ is an initial guess, the upsampled output from the previous $(n+1)$'th scale. Whereas SinGAN uses a GAN at each scale (as G), our adapted classical method (GPNN) uses a *non-parametric* Patch Nearest Neighbors (PNN) module (Fig. 2(b)) to enforce similarity of patch statistics be-

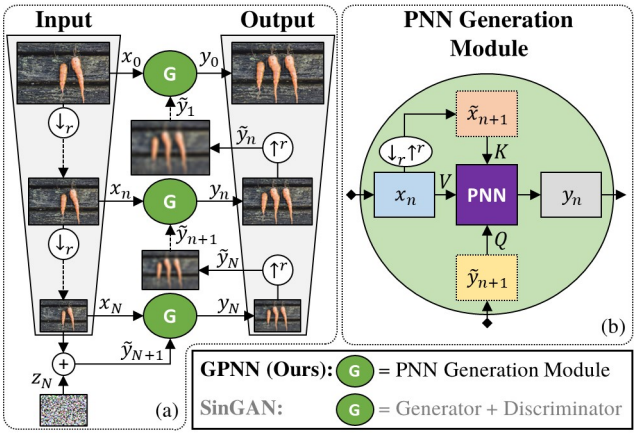

Figure 2. **The GPNN method.** *GPNN's multi-scale architecture is very similar to that of SinGAN [27]: Each scale consists of a single image generator G, that generates diverse outputs y_n with similar patch distribution as the source x_n. The generation module G (a GAN in [27]), is replaced here with a non-parametric PNN Generation Module. The coarsest level input is injected with noise.*

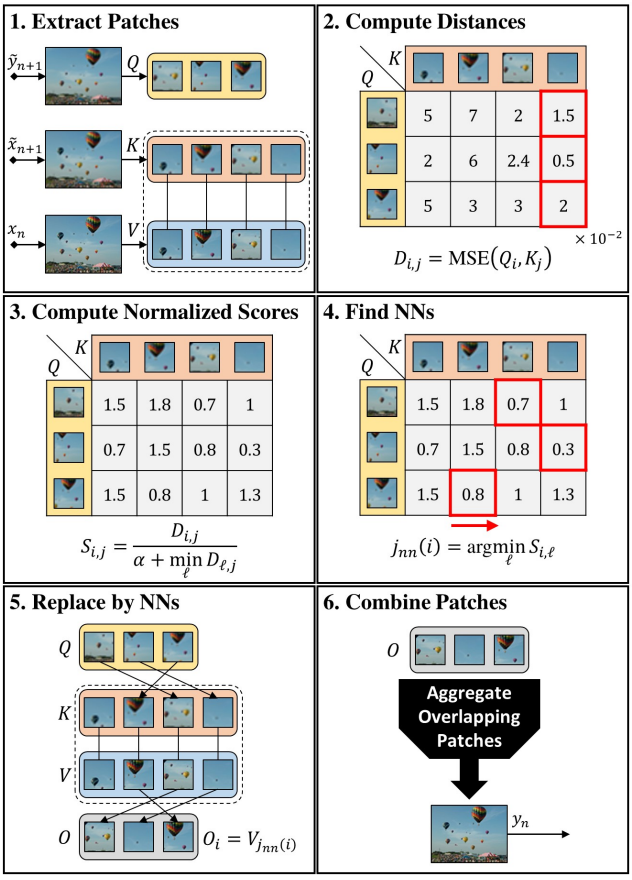

Figure 3. **Algorithmic steps of PNN.**

tween the source image and the generated output. The PNN module (similarly to the bidirectional similarity [30] core unit) promotes visual coherence and completeness using patch-nearest-neighbors. Moreover, as opposed to classical methods, noise is injected to coarse levels to induce randomness (similarly to SinGAN). This, and other modifications, allow to turn classical patch-based methods into a generative algorithm, which is fast and high-quality.

2.1. Patch Nearest Neighbor Generation Module

The goal of the PNN Generation Module is, at each scale n, to generate a new image y_n, based on an initial guess image $\tilde{y}_{n+1}$ and a source image x_n, such that the structure would be similar to that of $\tilde{y}_{n+1}$'s and the internal statistics would match that of x_n. To achieve that, PNN replaces each patch in $\tilde{y}_{n+1}$ with its nearest neighbor in x_n. While this approach is similar to that of [30], GPNN introduces two modifications to [30]'s core patch-similarity module (in addition to the unconditional input at coarse scales): (i) A *Query-Key-Value patch search strategy*, which improves the visual quality of the generated output; and (ii) A new *normalized patch-similarity measure*, which ensures visual completeness in an *optimization-free* manner. These differences are detailed below.

While classical methods look for nearest patch, and directly use its value, we claim that it may be suboptimal in the coarse-to-fine scheme. The initial guess is usually upscaled from coarser level, hence somewhat blurry. However, the source image patches are sharp. Therefore, blurry/smooth patches might be unintentionally favored.

To overcome this problem, PNN uses a Query-Key-Value scheme (see Figs. 2(b), 3, similarly to [33]) where the lookup patch and replacement patch are different. For each query patch Q_i in the initial guess $\tilde{y}_{n+1}$, we find its closest *key* patch K_j in a similarly-blurry version of the source image $\tilde{x}_{n+1} = x_{n+1}\uparrow^r = (x_n\downarrow_r)\uparrow^r$, but replace it with its corresponding *value* patch V_j from the sharp source image at that scale, x_n. Key and value patches are trivially paired (have the same pixel coordinates). That way, blurry *query* patches are compared with blurry *key* patches, but are replaced with sharp *value* patches.

Another difference regards the metric used to find nearest-neighbors. In some applications (e.g. image retargeting), it is essential to ensure that no visual data from the input is lost in the generated output. This was defined by [30] as visual *completeness*. In [30], completeness is enforced using an iterative optimization process over the pixels. InGAN [28] uses a cycle-consistency loss for this purpose (which comes at the expense of lost diversity). PNN enforces completeness using a *normalized patch similarity score*, which replaces the common L_2-metric. This similarity score is used to find patch-nearest-neighbors, and favors *key* patches that are not well-represented in the *query*, practically encouraging visual completeness in an optimization-free manner, as detailed in step 3 of the algorithm summary below.

PNN consists of 6 main algorithmic steps (numbered in Fig. 3):

1. **Extract patches:** Following the Query-Key-Value scheme, fully overlapping $p\times p$ patches are extracted from the source image x_n (as V), the initial guess $\tilde{y}_{n+1}$ (as Q), and the similarly-blurry image $\tilde{x}_{n+1}$ (as K). The only exception is the coarsest scale (n=N), where we use the value patches as keys (K=V). Keys and values are paired (patches from the same location have the same index in both pools).
2. **Compute Distances Matrix:** $D_{i,j} \leftarrow MSE(Q_i, K_j)$. We utilize the parallel nature of computing D, and run it on GPU for speed.
3. **Compute Normalized Scores:** PNN uses the following

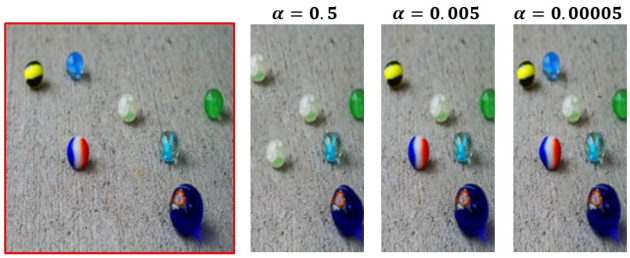

Figure 4. **Completeness – the role of α:** *Retargeting with different α values (input in red). For large α some input marbles are missing or duplicated in the output. As α decreases, completeness is further enforced. When $\alpha = 5\cdot 10^{-5}$, all original marbles are kept.*

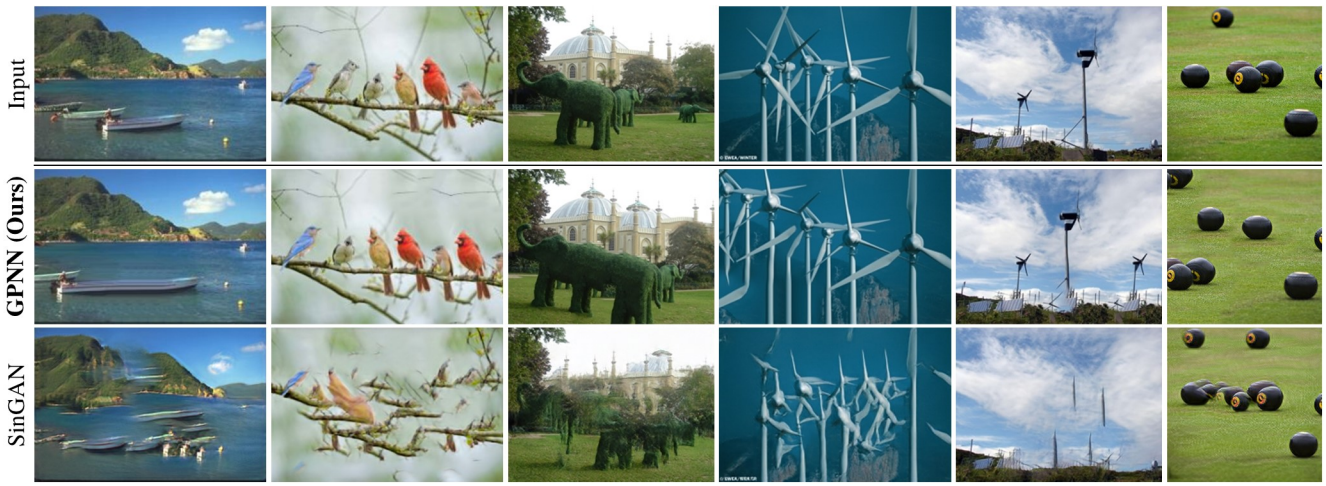

Figure 5. **Random Instance Generation Comparison:** *(Please zoom in) Images generated by our method are compared with images generated by SinGAN [27]. Images generated by GPNN (2nd row) look very realistic, whereas SinGAN produces many artifacts (3rd row).*

similarity score to encourage visual *completeness*:

$$S_{i,j} = \frac{D_{i,j}}{\alpha + \min_\ell D_{\ell,j}} \tag{2}$$

Intuitively, when a key patch K_j is missing from the queries, the normalization term (denominator) is large, resulting in smaller $S_{i,j}$ (thus increasing the chance of K_j to be chosen). On the other hand, when a key patch K_j appears among the queries, the denominator tends to α, hence $S_{i,j} \propto D_{i,j}$. The parameter α thus serves as a knob to control the degree of completeness, where small α encourages completeness, and $\alpha \gg 1$ is essentially the same as using MSE.

4. **Find NNs:** For each query patch Q_i, we find the index of its closest *key* patch, i.e. $j_{nn}(i) = \text{argmin}_\ell S_{i,\ell}$.
5. **Replace by NNs:** Each query patch Q_i is replaced with the value of its nearest neighbor, $V_{j_{nn}(i)}$. The output is denoted as O_i.
6. **Combine Patches:** Overlapping patches are combined into an image. Pixels in multiple overlapping patches are aggregated using a Gaussian weighted-mean [12].

Note that combining very different overlapping patches may cause inconsistencies and artifacts (lack of coherence). To mitigate these inconsistencies, PNN is applied T times at each scale (as in [30]). In the first iteration, the initial guess is the upscaled output from previous scale. In further iterations, the current output is inserted as initial guess.

Obtaining Diverse Outputs: We recognize that SinGAN's primary source of diversity is its coarsest level, where they use an *unconditional* GAN, as opposed to a more restricted *conditional* GAN in other scales. This makes sense, since small patches in the coarsest scale capture the global properties of objects in the image. Following that, GPNN injects noise in the coarsest scale (as in [8]). Specifically, the initial guess in the coarsest scale is defined as $\tilde{y}_{N+1} = x_N + z_N$, where $z_N \sim \mathcal{N}(0, \sigma^2)$. This makes the nearest-neighbors search nearly random (only the mean of the patches remains the same in expectation), thus induces diversity in the global structures, yet the PNN maintains coherent outputs. The use of different noise maps z_N is the basis for the diverse image generation presented in Sec. 3, whereas different choices for the initial guess $\tilde{y}_{N+1}$ are the basis for a wide variety of additional applications we present in Sec. 4.

Enforcing completeness: Fig. 4 Shows the effect of parameter α on the completeness of the output. As $\alpha \to 0$, $\min_l D_{l,j}$ dominates the denominator of the score S in Eq. (2). $S_{i,j}$ is thus large for well represented key patches K_j (since $\min_l D_{l,j} \approx 0$), and small for missing key patches (as $\min_l D_{l,j}$ is large). Since we take the minimum over S to find the target patch (step 4 of our algorithm), missing key patches are more likely to be chosen (promoting completeness – see rightmost image in Fig. 4). In contrast, when $\alpha \gg 1$, the denominator becomes a constant $\sim\alpha$ (since $\forall i, j \ D_{i,j} \leq 1$), hence the closest patch is chosen based on MSE only (completeness not enforced). Parameter α thus sets the desired level of completeness (illustrated in Fig. 4).

Runtime: A key advantage of GPNN over GAN-based methods (e.g. [3, 27, 28]) is its runtime. While GAN-based methods require a long training phase (hours *per image*), GPNN uses a non-parametric generator which needs no training. Thus, SinGAN takes $\sim$1 *hour* to generate a 180×250 sized image, whereas GPNN does so in 2 *seconds* (see Tab. 1). Further details are provided in project webpage.

3. Results & Evaluation

We evaluate and compare the performance of GPNN to SinGAN [27] on the main application of *random image generation* based on a single natural image. We first follow the *exact same* evaluation procedure of SinGAN [27], with the same data. We then add more measures, more tests and introduce more data. We show substantial supremacy in Visual-Quality and Realism with a large margin, both quantitatively and qualitatively, over all measures and datasets. The complete set of results can be found in the project webpage. Runtime of GPNN is shown to be $\times 10^3$ faster. To be fair, a trained SinGAN can be used to generate many variations of the same image. However, it would require generation

Dataset	Method	SIFID ↓ [27]	NIQE ↓ [24]	Confusion-Paired [%] ↑ Time Limit	Confusion-Paired [%] ↑ No Time Limit	Confusion-Unpaired [%] ↑ Time Limit	Confusion-Unpaired [%] ↑ No Time Limit	Realism competition [%] ↑ GPNN vs. SinGAN	Diversity [27]	Runtime ↓ [sec]
Places50 [27, 40]	SinGAN (N)	0.085	5.240	21.5±1.5	22.5±2.5	42.9±0.9	35.0±2.0	28.1±2.2	0.5	3888.0
	GPNN (σ=1.25)	**0.071**	**5.049**	**44.7**±1.7	**38.7**±2.0	**47.6**±1.5	**45.8**±1.6	**71.9**±2.2	0.5	**2.1**
	SinGAN ($N-1$)	0.051	5.235	30.5±1.5	28.0±2.6	**47**±0.8	33.9±1.9	32.8±2.3	0.35	3888.0
	GPNN (σ=0.85)	**0.044**	**5.037**	**47**±1.6	**42.6**±1.7	**47**±1.4	**45.9**±1.6	**67.2**±2.3	0.35	**2.1**
SIGD16	SinGAN (N)	0.133	6.79	28.0±3.3	12.0±2.3	35.9±2.7	39.9±2.7	41.7±3.3	0.49	3888.0
	GPNN (σ=0.75)	**0.07**	**6.38**	**46.6**±2.6	**43.3**±2.7	**46.3**±2.6	**46.8**±2.4	**59.3**±3.3	0.52	**2.1**

Table 1. **Quantitative Evaluation**. *We evaluate our results over two datasets: The Place50 images used in the evaluation of [27], and our new SIGD dataset (see text). We use a variety of measures: NIQE (unpaired image quality assessment) [24], SIFID - single image FID [27], and human evaluations through an extensive user-study (see text). We repeat the evaluation for multiple diversity levels (measured as proposed by [27]). The table shows GPNN outperforms SinGAN by a large margin in every aspect: Visual quality, Realism, and Runtime.*

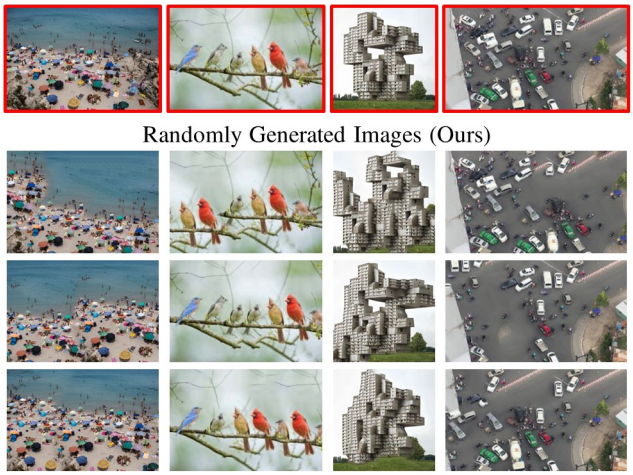

Figure 6. **Diverse Image Generation:** *(Please zoom in) Random images produced by GPNN from a single input (marked in red).*

of 1800 random output images from the same input image, in order for the runtime of SinGAN and GPNN to match (assuming 1hr for training SinGAN with 0 time for inference, compared to 2 seconds for GPNN).

Data: We evaluate our performance on two datasets. The first is the set of 50 images used in SinGAN [27] for their evaluation (50 images from Places365 dataset [40]). The second is a small, but very different dataset we introduce, Single Image Generation Dataset (SIGD) – a set of 16 images that well exemplify a variety of different important aspects of single image generation tasks (visual aspects not represented in the structural Places images). These include: 7 images extracted from figures in the SinGAN paper [27], 2 images from the Structural Analogies paper [3] and 7 more we collected from online sources. These SIGD images are characterized by having many more conceivable versions per image than Places images, hence are more suited for comparing the quality of random image generation.

Visual Results: Fig. 6 shows GPNN results for diverse image generation, which highlight 3 characteristics of GPNN: (i) *Visual quality*: The results are sharp looking with almost no artifacts (please zoom-in). (ii) *Realistic structure*: The generated images look real, the structures make sense (iii) *Diversity*: GPNN produces high diversity of results. This diversity can be observed in the number of objects and their location (birds, cars, beach), but also in their size (building), and the global arrangement (shore shape, branches). This is achieved while maintaining characteristics (i) and (ii). Fig. 1 (top-left) further shows results of GPNN on SinGAN's pivotal examples in their paper [27]. Fig. 5 shows a visual comparison of GPNN to SinGAN. Images generated by GPNN look very realistic, whereas SinGAN often produces artifacts/structures that make no sense (note that birds and branches generated by GPNN look very realistic despite the different ordering of the birds on the branch, while SinGAN doesn't maintain realistic structures).

Quantitative evaluation: Table. 1 shows quantitative comparison between GPNN and SinGAN. All generated images are found in the project webpage. We use SIFID [27] to measure the distance between the source and the generated image patch distributions, as well as NIQE [24] for reference-free quality evaluation. GPNN has much greater flexibility in choosing the degree of diversity (by tuning the input noise). However, for fair comparison, we adjusted the input noise level in GPNN so that its diversity level matches that of SinGAN's results. Diversity is measured as proposed by SinGAN: pixelwise STD over 50 generated images. On SinGAN's places50 dataset we achieve clear superiority in both measures (SIFID & NIQE), for both levels of diversity used in SinGAN. The margin is even larger on the SIGD dataset.

Qualitative Evaluation – Extensive User-Study: Table 1 displays the results of our user-study, conducted using Amazon Mechanical Turk platform. Our surveys composed of: 2 setups (paired & unpaired) × 2 datasets (Places50 [27] & SIGD) × multiple diversity levels × 2 temporal modes (Time limit & No time limit). Altogether, these resulted in *27 different surveys, each answered by 50 human raters*. The number of questions in each survey equals the number of images in the dataset. Results are summarized in Table 1.

paired / unpaired setups: In the paired setup, the ground-truth image and a generated image are shown side-by-side in random order. The rater is asked to determine which one is real. In the unpaired setup, a single image is shown (real or generated), and the rater has to decide whether it is real or fake. In both setups we report the percent of trials the rater was "fooled" (hence, the highest expected score is 50%). The above setups were applied separately to GPNN and SinGAN. In addition, we also ran a *Realism competition* where the rater has to decide which image looks more realistic, in a paired survey of GPNN-vs-SinGAN (here the highest possible score is 100%).

Time limit / No time limit: We first followed the time-limited setup of SinGAN's user study [27], which flashed each image

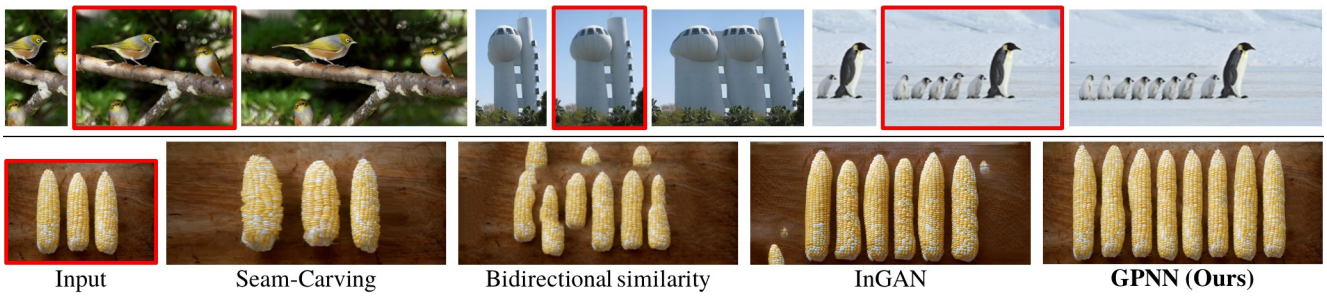

Figure 7. **Retargeting:** *(Please zoom in) Top rows show retargeted images by our method. Patch distribution is kept when retargeting to various target shapes. Bottom row shows comparison with previous patch-based [1], [30] and GAN-based [28] methods.*

for only 1 second ("Time limit"). We argue that this limit makes it hard for raters to notice differences, resulting in a strong bias towards 50% for any method. We therefore repeat the study also with unrestricted time ("No time limit").

Results: GPNN scores significantly higher than SinGAN in all setups (Table 1). Moreover, in the unlimited-time surveys (when the human rater had more time to observe the images), SinGAN's ability to "fool" the rater significantly drops, especially in the unpaired case. In contrast, having no time-limit had a very small effect on GPNN's confusion rate. In all surveys GPNN got results very close to chance level (50%). The fact that an observer with unlimited time can rarely distinguish between real and GPNN generated images, implies high realism of the generated results. Finally, in the direct GPNN-vs-SinGAN survey (unlimited time), GPNN's results were selected as more realistic than SinGAN's for most images in all surveys.

4. Additional Applications

In addition to diverse image generation, GPNN gives rise to many other applications, both "classical" and "GAN-only", all within a *single unified framework*. The different applications are obtained simply by modifying a few basic parameters in GPNN, such as the pyramid depth N, the initial guess at the coarsest scale $\tilde{y}_{N+1}$, and the choice of the hyper-parameter α in Eq. 2. We next describe each application, along with its design choices.

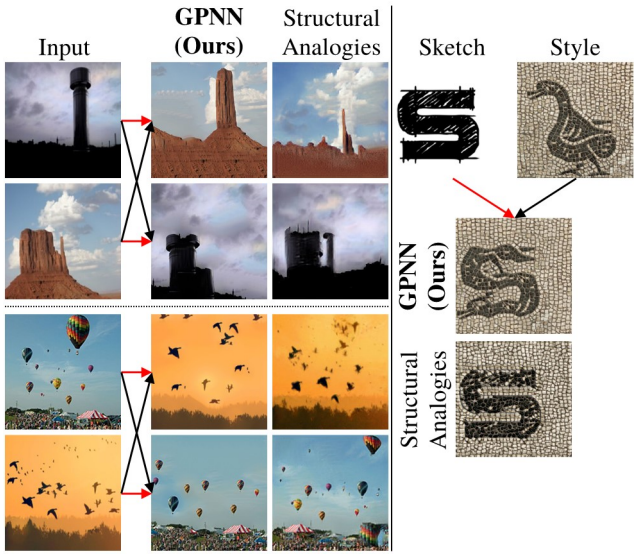

Figure 8. **Structural Analogies:** *(Please zoom in) Red arrow indicates the 'structure', while black arrow indicates the 'source' (defining the patch distribution to match). GPNN is compared to [3]. GPNN can also generate sketch-to-image instances (right).*

Retargeting: The goal is to resize the single source image to a target size (smaller or larger; possibly of different aspect ratio), but *maintain the patch distribution of the source image* (i.e., maintain the size, shape and aspect-ratio of all small elements of the source image) [2, 28, 30]. GPNN starts by naively resizing the input image to the target size, then downscale it by a factor of r^N. This is injected as the initial guess $\tilde{y}_{N+1}$. As we wish to retain as much visual information as possible from the source image, α in Eq. 2 is set to a small value (e.g., $\alpha = .005$), thus promoting "Completeness". To get better final results, the described process is done gradually (similarly to [30]), i.e. resizing in a few small steps. Retargeting results can be seen in Fig. 7 and 1. Fig. 7 further compares the performance of our method with that of [1, 28, 30]. GPNN produces results which are more realistic, and with less artifacts.

Image-to-image and Structural Analogies: We demonstrate image-to-image translations of several types. There exist many approaches and various goals and brandings for image-to-image translations; Style-transfer, domain-transfer, structural-analogies [8, 10, 16, 18–20, 31, 42]. Given two input images A and B, we wish to create an image with the patch distribution of A, but which is structurally aligned with B. Namely, a new image where all objects are located in the same locations as in B, but with the visual content of A. For that, GPNN sets the source image x to be A. The initial guess $\tilde{y}_{N+1}$, is chosen as B downscaled by r^N. This guess sets the overall structure and location of objects, while GPNN ensure that the output has similar patch distribution to A. The output should contain as much visual data from A (e.g., in the bottom-left pair in Fig. 8, it is desired that many types of balloons will appear in the output), hence we set α in Eq. 2 to be small (e.g., $\alpha = .005$). Finally, to refine the output, it is downscaled again by r^N and re-injected to GPNN. Results can be found in Fig. 8. Our method indeed creates new objects located as in B, while the patch distribution is similar to that of A as desired. GPNN works well also for sketch-to-image instances as shown. Compared to the GAN-based method of [3], our results suffer from less artifacts, and are more reliable to the style of the source image A (e.g., the "S" image in Fig. 8). In addition to providing

GPNN (Ours) Bidirectional Similarity

Figure 9. **Collage:** *Multiple input images are seamlessly combined into a single coherent output, maintaining visual information from all inputs. Note the higher quality of GPNN compared to Bidiectional-Similarity [30]. The 3 input images are found in Fig.1.*

superior results, GPNN is also several orders of magnitude faster than [3].

Conditional Inpainting: Similarly to the well studied inpainting task, in this task an input image with some occluded part is received. However, in our suggested conditional version, in addition to regular image completion, the user can further steer the way the missing part is filled. This is obtained by the user marking the image region to be completed, with a region of *uniform color of choice*, which is the "steering direction" (e.g., blue to fill sky, green to guide the completion toward grass, etc.). This will be the source image x. Note that differently from common image editing, GPNN does not "see" patches which originated in the occluded area. The initial guess $\tilde{y}_{N+1}$ is set to be a downscaled version of x by r^N. PNN applied at the coarsest level replaces the masked part coherently with respect to the chosen color. In finer levels, details and textures are added. In this task, completeness is not required, hence a large α is set in Eq. 2. Fig. 1 shows that the choice of different colors for the same masked region indeed affects the outcome, while maintaining coherent and visually appealing images.

Image Collage: This task, previously demonstrated in [30], aims to *seamlessly* merge a set of n input images $\{x^i\}_{i=1}^n$ to a single output image, so that no information/patch from the input images is lost in the output. We create the initial guess $\tilde{y}_{N+1}$ by first naively concatenating the input images, and downscaling this concatenation by r^N. Then, we use the same design as for retargeting, with a single change - GPNN extracts patches from all the source images (rather than from a single source image in retargeting). Fig. 9 shows a collage produced by GPNN, on an example taken from [30]. Compared with [30], our results are sharper and more faithful to the inputs.

Image Editing: In image editing/reshuffling [27, 30], one makes a change in the image (move objects, add new objects, change locations, etc.), and the goal is to seamlessly blend the change in the output image. We use the unedited original image as the source image x, and a downscaled version of the edited image by r^N as the initial guess $\tilde{y}_{N+1}$. Completeness is not required in this task (e.g., if the edit removes an object), thus we set α in Eq. 2 to be large. Similarly to inpainting, the area around and inside the edited region is "corrected" by our algorithm to achieve coherence. Noise may be added at the coarsest level, to allow for different coherent solutions given a single input. Fig. 10 shows our editing results compared to SinGAN's [27]. Our results tend to be less blurry (especially visible around the edited region).

5. GANs vs. Patch Nearest-Neighbors: Pros & Cons

The experiments in Secs. 3 and 4 show striking superiority of GPNN compared to single-image GANs, both in visual-quality and in run-time (while having comparable diversity). This section first analyzes the source of these surprising *inherent advantages* of simple classical patch-based methods (exemplified through GPNN). Nevertheless, single-image GANs have several significant capabilities which *cannot be realized* by simple patch nearest-neighbor methods. Hence, despite the title of our paper, you may not always want to "Drop the GAN"... These *inherent limitations* of classical patch-based methods (GPNN included) are also discussed.

Advantages

The advantages of Patch-based methods over GANs stem primarily from one basic fundamental difference: Single-image GANs *implicitly learn* the patch-distribution of a single image, whereas classical Patch-based approaches *explicitly maintain* the entire patch-distribution (the image itself), and directly access it via patch nearest-neighbor search. This fundamental difference yields the following advantages:

Visual Quality: An output image produced by patch nearest-neighbor search, is *composed of original image patches* pulled out directly from the input image. In contrast, in GANs the output is *synthesized via an optimization process*. GPNN thus produces images whose patches are more faithful to the original input patches than GANs. This yields sharper outputs (almost as sharp as the input), with fewer undesired visual artifacts (please zoom in on Fig. 5, 10 to compare).

Runtime: Since no training takes place, the runtime of patch-based methods reduces *from hours to seconds* compared to GANs (Table 1). Moreover, since nearest-neighbor search can be done independently and in parallel for different image patches, this naturally leverages GPU computing.

Visual Completeness: While GANs are trained to produce patches of high likelihood (thus encouraging output *Coherence* to some extent), no mechanism enforces *Completeness* (i.e., encourage all patches of the input image to appear in the output image). Lack of Completeness is further intensified by the natural tendency of GANs to suffer from mode collapse. In contrast, classical patch-based methods can explicitly enforce Completeness, e.g., by optimizing *Bidirectional patch similarity* between the input and output image [30], or in an optimization-free manner by using GPNN's *patch-specific* normalized score (Eq. (2)). InGAN [28] has successfully introduced Completeness into GANs, by training a *conditional* single-image GAN via an encoder-encoder scheme with a reconstruction loss. However, this came with a price: lack of diversity in the generated outputs. In contrast, GPNN is able to promote both Completeness & Coherence, as well as large output diversity. There is an *inherent trade-off between the Completeness and Diversity*. The α parameter in Eq. (2) thus provides a "knob" to control the degree of desired Completeness in GPNN (according to the image/application at hand). Despite this new flexibility of

GPNN compared to GANs, all experiments in Table 1 were performed with a fixed α (for fairness).

Visual Coherence (realistic image structures): The iterative nearest-neighbor search of classical patch-based methods prevents forming adjacency of output patches that are not found adjacent in the input image (for any image scale). This tends to generate realistic looking structures in the output. In contrast, such tendency for coherence is only weakly enforced in GANs. Proximity between unrelated patches may emerge in the generated output, since the generator is fully convolutional with limited receptive field. The generator typically generates mitigating pixels, hopefully with high likelihood. This often results in incoherent non-realistic image structures and artifacts that do not exist in the input image. See examples in Fig. 5 and in the project webpage.

Controlling diversity vs. global structure: There is a natural trade-off between high output diversity and preserving global image structure. The magnitude σ of the noise added in GPNN to the coarsest scale of the input image, provides a simple user-friendly "knob" to control the degree of desired output diversity. It further yields a natural *continuum* between large diversity (high noise) and global structural fidelity (low noise). GANs, on the other hand, do not hold any mechanism for controlling the preservation of global structure (although there are some inductive biases that tend to preserve global structure implicitly [36]). While GANs may support a few discrete levels of diversity (e.g., [27] demonstrates 2 diversity levels), this diversity is not adjustable.

Limitations

Patch Generalization: Classical patch-based methods use a *discrete* patches distribution. GANs on the other hand learn a *continuous* distribution. GANs can therefore generate novel patches with high likelihood from the learned distribution. This capability is lacking in patch-based methods. Note, however, that this lack of generalization is only local – i.e., restricted only to tiny 7x7 patches. By combining the original image patches in very different ways, patch-based methods (like GPNN) obtain a very large diversity of new global structures, that tend to make much more visual sense than those generated by SinGAN, as confirmed by our extensive evaluations (Tab. 1, project page ,Fig. 6). Local patch-level-generalization can sometimes be advantageous (e.g., image harmonization [27]), but may also be disadvantageous, as it frequently generates undesired visual artifacts (Fig. 5).

Continuous output generation: Neural networks are continuous functions. Small change in the latent input causes a small divergence in the generated output. This enables latent space interpolation and other smooth manipulations, such as single image animation [27], or smooth resizing animation [28]. In contrast, nearest neighbor search is discrete in nature. This prevents naively performing continuous interpolation or animation in classical patch-based methods.

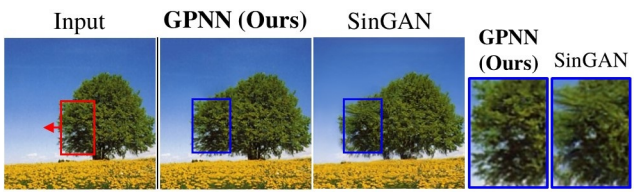

Figure 10. **Image Editing:** *A naively edited image is injected to our pyramid of PNNs. Compared with the results of [27].*

Mapping to patches vs. Mapping to pixels: In classical nearest-neighbor methods (including GPNN), the nearest-neighbor search maximizes the quality of the extracted patches, but not the quality of the final output pixels. The formation of the output image typically involves heuristic averaging of overlapping patches. This may introduce some local blurriness in patch-based methods. GAN discriminators also judge output patches in the size of their receptive field. However, since the generators receive pixel-based gradients, they can *optimize directly for each output pixel*.

Are GANs "Fancy Nearest Neighbors Extractors"?

It has been argued that GANs are only elaborated machinery for nearest neighbors retrieval. In *single-image* GANs, the "dataset" is small enough (patches from a single image), providing an excellent opportunity to quantitatively examine this claim. Fig. 11 shows two generated random instances of the same input image (red), once using SinGAN (green) and once using GPNN (purple). We measured the distance between the generated patches to their nearest neighbors in the input image and plotted the histograms of these distances (RMSE). We repeated this experiment twice for two different input images. In both cases, SinGAN generated novel patches that do not have close nearest-neighbors in the input image. That is, GANs are capable of generating new local samples (in this case - small patches) beyond nearest neighbors. However, this capability of GANs comes with a price, its newly generated instances suffer from blur and artifacts (*please zoom in*). In contrast, GPNN generated samples have higher fidelity to the input (the histogram is peaked at zero), resulting with higher output quality. While it has a limited ability to generate novel patches, it does show large diversity in the output images.

6. Conclusion

We observe that classical patch-based methods can solve tasks previously perceived as "GAN-only". Specifically, we show that with some modern adaptation, classical methods have the ability to produce a plethora of new images based on a single *natural* image. While such a transition (from powerful GANs to classical patch-based methods) might seem unnatural, we show that the task at hand greatly benefits from that. Our extensive evaluations show that classical methods perform this task with higher-quality, and orders of magnitude faster than single-image GANs.

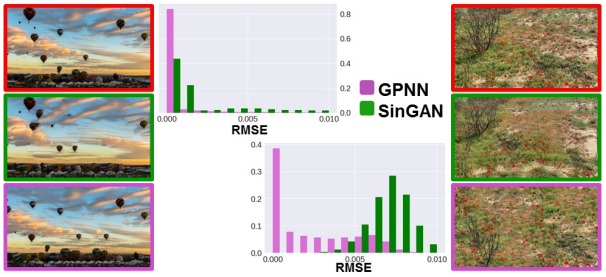

Figure 11. **Are GANs "Fancy Nearest Neighbors Extractors"?** *Input image (red), GPNN generated (purple), SinGAN generated (green). Please see text for details.*

References

[1] Shai Avidan and Ariel Shamir. Seam carving for content-aware image resizing. In *ACM SIGGRAPH 2007 papers*, pages 10–es. 2007. 6

[2] 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. 1, 2, 6

[3] Sagie Benaim, Ron Mokady, Amit Bermano, and L Wolf. Structural analogy from a single image pair. In *Computer Graphics Forum*. Wiley Online Library, 2020. 1, 2, 4, 5, 6, 7

[4] Urs Bergmann, Nikolay Jetchev, and Roland Vollgraf. Learning texture manifolds with the periodic spatial GAN. *arXiv preprint arXiv:1705.06566*, 2017. 2

[5] 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. 2

[6] Tali Dekel, Tomer Michaeli, Michal Irani, and William T Freeman. Revealing and modifying non-local variations in a single image. *ACM Transactions on Graphics (TOG)*, 34(6):1–11, 2015. 1

[7] 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. 1, 2

[8] Michael Elad and Peyman Milanfar. Style transfer via texture synthesis. *IEEE Transactions on Image Processing*, 26(5):2338–2351, May 2017. 4, 6

[9] 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 (CVPR)*, June 2019. 2

[10] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2414–2423, 2016. 6

[11] 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. 2

[12] Daniel Glasner, Shai Bagon, and Michal Irani. Super-resolution from a single image. *2009 IEEE 12th International Conference on Computer Vision*, pages 349–356, 2009. 4

[13] 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

[14] 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

[15] 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. 2

[16] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017. 6

[17] Nikolay Jetchev, Urs Bergmann, and Roland Vollgraf. Texture synthesis with spatial generative adversarial networks. *arXiv preprint arXiv:1611.08207*, 2016. 2

[18] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European conference on computer vision*, pages 694–711. Springer, 2016. 6

[19] Nicholas Kolkin, Jason Salavon, and Gregory Shakhnarovich. Style transfer by relaxed optimal transport and self-similarity. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10051–10060, 2019. 6

[20] Jing Liao, Yuan Yao, Lu Yuan, Gang Hua, and Sing Bing Kang. Visual attribute transfer through deep image analogy. *arXiv preprint arXiv:1705.01088*, 2017. 6

[21] 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. 2

[22] 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. 2

[23] 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. 2

[24] Anish Mittal, Rajiv Soundararajan, and Alan C Bovik. Making a "completely blind" image quality analyzer. *IEEE Signal processing letters*, 20(3):209–212, 2012. 5

[25] Y. Pritch, E. Kav-Venaki, and S. Peleg. Shift-map image editing. In *International Conference on Computer Vision (ICCV)*, 2009. 1

[26] Yi Ren, Yaniv Romano, and Michael Elad. Example-based image synthesis via randomized patch-matching, 2016. 1, 2

[27] 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. 1, 2, 4, 5, 7, 8

[28] Assaf Shocher, Shai Bagon, Phillip Isola, and Michal Irani. InGAN: Capturing and remapping the "DNA" of a natural image. In *arXiv*, 2019. 1, 2, 3, 4, 6, 7, 8

[29] 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 (CVPR)*, June 2018. 2

[30] 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. 1, 2, 3, 4, 6, 7

[31] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Improved texture networks: Maximizing quality and diversity in feed-forward stylization and texture synthesis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6924–6932, 2017. 6

[32] 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. 2

[33] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017. 3

[34] Yael Vinker, Eliahu Horwitz, Nir Zabari, and Yedid Hoshen. Deep single image manipulation. *arXiv preprint arXiv:2007.01289*, 2020. 2

[35] Yonatan Wexler, Eli Shechtman, and Michal Irani. Space-time completion of video. *IEEE Transactions on pattern analysis and machine intelligence*, 29(3):463–476, 2007. 1, 2

[36] Rui Xu, Xintao Wang, Kai Chen, Bolei Zhou, and Chen Change Loy. Positional encoding as spatial inductive bias in GANs. *arXiv preprint arXiv:2012.05217*, 2020. 8

[37] 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. 2

[38] 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. 2

[39] 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. 2

[40] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Antonio Torralba, and Aude Oliva. Places: An image database for deep scene understanding, 2016. 5

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

[42] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017. 6

[43] 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. 2