Drop the GAN: 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/

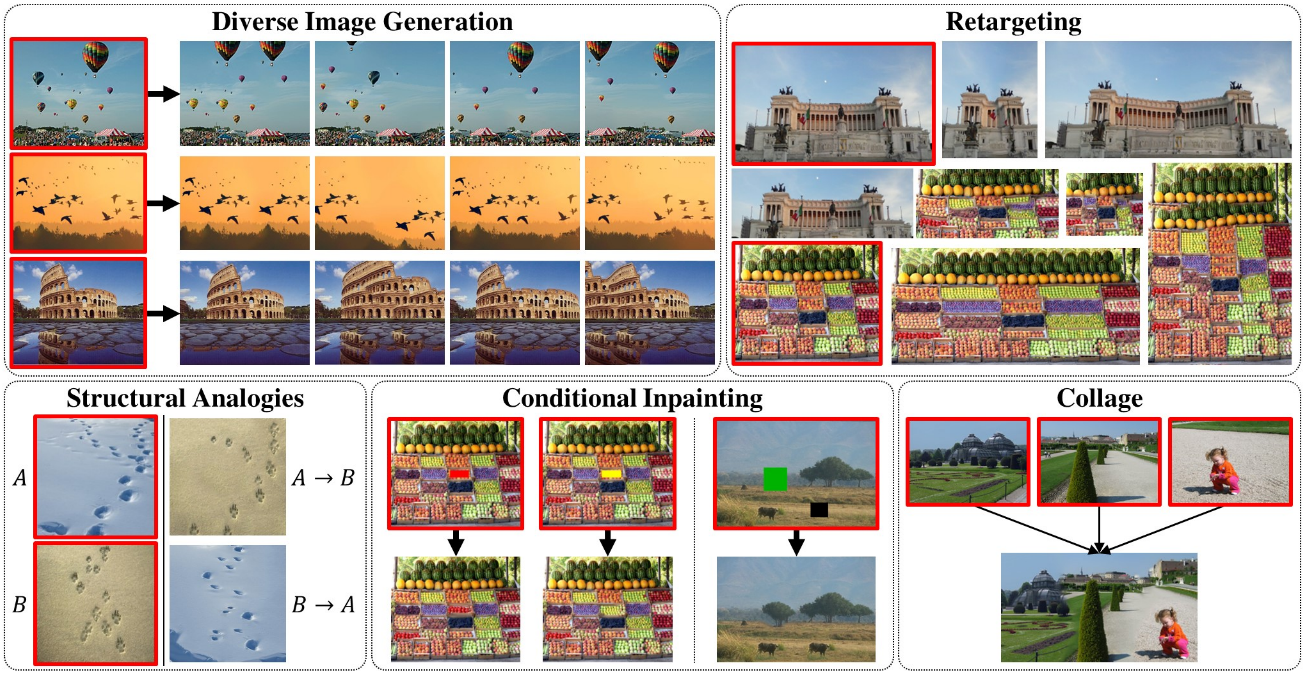

Figure 1: *Our simple unified framework covers a broad spectrum of single-image generative tasks, that usually require hours of training per image for GANs. Using patch nearest neighbors and a single source image, we can perform these tasks in a few seconds and with higher quality. We show here results obtained with our method for pivotal examples shown in SinGAN [25], InGAN [26], Structural analogies [3] and Bidirectional Similarity [28]. Additionally, we introduce novel applications such as Conditional-Inpainting. Input images marked in red.*

Abstract

Single image generative models perform synthesis and manipulation tasks by capturing the distribution of patches within a single image. The classical (pre Deep Learning) prevailing approaches for these tasks are based on an optimization process that maximizes patch similarity between the input and generated output. Recently, however, Single Image GANs were introduced both as a superior solution for such manipulation tasks, but also for remarkable novel generative tasks. Despite their impressiveness, single image GANs require long training time (usually hours) for each image and each task. They often suffer from artifacts and are prone to optimization issues such as mode collapse. In this paper, we show that all of these tasks can be performed without any training, within several seconds, in a unified, surprisingly simple framework. We revisit and cast the "good-old" patch-based methods into a novel optimization-free framework. We start with an initial coarse guess, and then simply refine the details coarse-to-fine using patch-nearest-neighbor search. This allows generating random novel images better and much faster than GANs. We further demonstrate a wide range of applications, such as image editing and reshuffling, retargeting to different sizes, structural analogies, image collage and a newly introduced task of conditional inpainting. Not only is our method faster ($\times 10^3$-$\times 10^4$ than a GAN), it produces superior results (confirmed by quantitative and qualitative evaluation), less artifacts and more realistic global structure than any of the previous approaches (whether GAN-based or classical patch-based).

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., [7, 28, 2, 23, 6, 24]) 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 [28, 2] 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.

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" [27, 9, 35, 40, 34, 11, 30]. They train a GAN on a single image, in an unsupervised way, and have shown to produce impressive generative diversity of results, as well as notable new generative 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 [25] for generating a *large diversity* of different image instances – all sampled from the input patch distribution, InGAN [26] for flexible image retargeting, Structural-Analogies [3], texture synthesis [15, 4, 38, 36], and more [13, 20, 21, 12, 32, 19, 5].

However, despite their remarkable capabilities (both diverse generated outputs and diverse new tasks), single-image GANs come with several heavy penalties compared to their simpler classical patch-based counterparts: (i) They require very 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.

Hence, while the course of history has taken the field of image synthesis, from the patch search methods to powerful GANs, it turns out that "good-old" patch-based methods are superior in many aspects. In this paper we suggest to reconsider simple patch nearest neighbors again, *but with a new twist*, which allows to inject new Single Image GANs-like generative capabilities into nearest-neighbor patch-based methods, thus obtaining *the best of both worlds*. We further analyze and characterize the pros and cons of these 2 approaches (patch nearest-neighbors vs. single-image GANs) in Sec. 5.

We observe that the generative *diversity* of single-image GAN methods (which classical methods lack), stems primarily from their *unconditional* input at coarser image scales. We further observe that the main source of the above-mentioned drawbacks (slowness, mode-collapse, and poorer visual quality compared to classical methods) stems from the generative (GAN-based) module. We therefore suggest to "drop the GAN" [1], and replace this module with a simple (upgraded) Patch Nearest-Neighbor (PNN) module, while maintaining the unconditional nature of GAN-based methods. This gives rise to a simple new *generative* patch-based algorithm, which we call *Generative Patch Nearest-Neighbor* (**GPNN**). Its noise input at coarse-levels yields *diverse image generation*, of much higher quality and significantly faster than single-image GANs (with similar diversity). This is verified via extensive evaluations (Sec. 3).

GPNN can perform the new generative tasks of single-image GANs, as well as the old classical tasks, in a single unified framework, without any training, in an optimization-free manner, within a few seconds. Unlike single-image GAN-based methods, GPNN is very fast ($\times 10^3$-$\times 10^4$ faster), and produces superior visual results (confirmed by extensive quantitative and qualitative evaluation). Unlike classical patch-based methods, GPNN enjoys the non-deterministic nature of GANs, their large diversity of possible outputs, and new generative tasks/capabilities.

We demonstrate a wide range of applications of GPNN: first and foremost *diverse image generation*, but also image editing, image retargeting, structural analogies, image collages, and a newly introduced task of "conditional inpainting". We show that GPNN produces results which are either comparable or of higher quality than any of the previous approaches (whether GAN-based or classical patch-based). 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 can be cast as a generative model, which substantially outperforms modern single-image GANs – both in quality and in speed .
- We introduce such a casting – GPNN, a new generative patch-based algorithm, as an alternative to single-image GANs, which provides a unified framework for a large variety of applications.
- We analyze and discuss the *inherent pros & cons* of the modern GAN-based approaches vs. classical Patch-based approaches. We experimentally characterize the extent

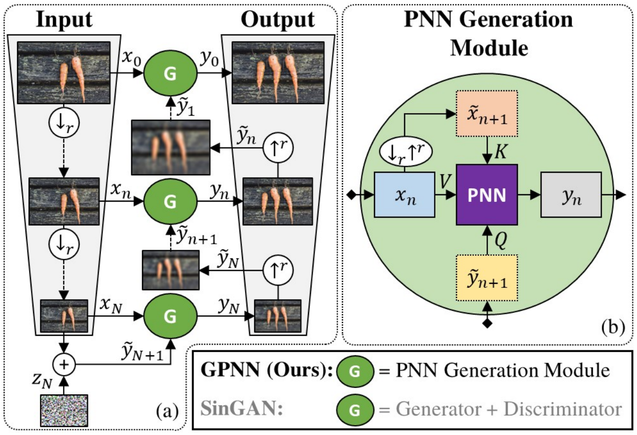

Figure 2: **The GPNN method.** *GPNN's multi-scale architecture is very similar to that of SinGAN [25]: Each scale consists of a single image generator G, that generates diverse outputs y_n with similar patch distribution as the source x_n. The generation module G (a GAN in [25]), is replaced here with a non-parametric PNN Generation Module. The coarsest level input is injected with noise.*

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

to which single-image GANs perform nearest-neighbor extraction behind the scenes.

2. Method

Our goal is to efficiently cast patch nearest neighbor search as a diverse single-image generative model. To achieve that, GPNN uses a multi-scale architecture with *noise injected input* (Fig. 2(a); Sec. 2.1), similarly to SinGAN [25]. However, in each scale, GPNN uses a *non-parametric* Patch Nearest Neighbor (PNN) Generation Module (Fig. 2(b), Sec. 2.2), as opposed to a full-scale GAN in SinGAN. PNN generates new images with similar patch distribution as the source image *at that scale*.

2.1. Multi-scale Architecture

It was previously recognized (e.g., [25, 28, 26, 24]), that different information is captured in each scale of an image – from global arrangement at coarser scales, to textures and fine details at finer scales. To capture details from all scales, GPNN has a coarse-to-fine architecture (illustrated in Fig. 2(a)). Given a source image x, it builds a pyramid $\{x_0, \dots, x_N\}$, where x_n is x downscaled by a factor r^n (for $r{>}1$; in our current implementation $r{=}\frac{4}{3}$). GPNN uses the same patch size $p \times p$ in all scales ($p{=}7$ in our implementation). Similarly to [25], the depth of the pyramid N is chosen such that p is approximately half of the image height. At scale n, a new image y_n is generated by PNN Generation Module (see Fig. 2(b); Sec. 2.2), using *real* patches from the source image at that scale x_n, guided by $\tilde{y}_{n+1}$, the initial guess. PNN enforces similarity between the internal statistics of the output image and source image at each scale.

In the coarsest level, the initial guess is the source image injected with noise (similarly to [8]), $\tilde{y}_{N+1} = x_N + z_N$, where $z_N \sim \mathcal{N}(0, \sigma^2)$. The coarsest scale defines the arrangement of objects in the image. Injecting noise at that scale makes the nearest-neighbors search nearly random (the mean of the patches remains the same in expectation), hence induces diversity in the global arrangement, 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 are the basis for a wide variety of additional applications we present in Sec. 4.

In finer scales, the initial guess is the upscaled output of the coarser level, $\tilde{y}_{n+1} = y_{n+1}\uparrow^r$. The output at each scale is a refinement of the coarser scale output. Hence, the final output $y = y_0$ shares the internal statistics of x at all scales.

2.2. Patch Nearest Neighbors Generation Module

The goal of the PNN Generation Module is 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 patches from the initial guess $\tilde{y}_{n+1}$ with patches from the source image x_n.

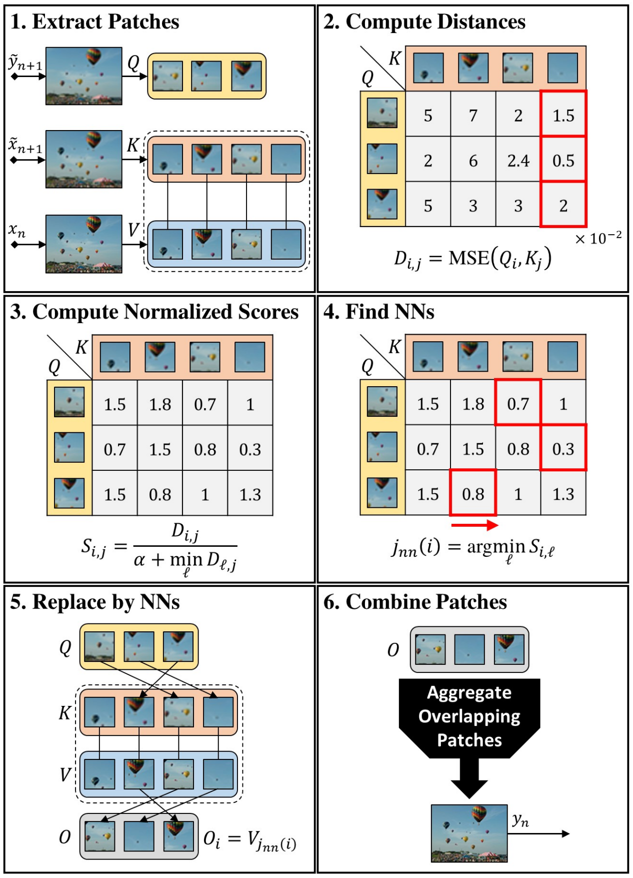

Figure 3: **Algorithmic steps of PNN.**

While this coarse-to-fine refinement strategy bears resemblance to that of classical patch-based methods, GPNN introduces 2 major differences (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.

Classical patch-based methods use a Query-Reference scheme, where the query is the initial guess and the reference is the source image. Each *query* patch (from the initial guess image) is replaced, or optimized to get closer to, a *reference* patch (from the source image). This encourages similarity between the internal statistics of the output and the source. However, that scheme may fail when there is a significant distribution shift between query and reference patches. For example, when the query patches are blurry (due to upscaling the initial guess from a coarser resolution), they might be matched to blurry reference patches.

To overcome this problem, PNN uses a Query-Key-Value scheme (see Figs. 2(b), 3, similarly to [31]). Instead of comparing the query patch to a *reference* patch and replacing it by that same patch (as done in classical methods), here the lookup patch and replacement patch are different. For each query patch Q_i in $\tilde{y}_{n+1}$, we find its closest *key* patch K_j in a (blurry) upscaled version of the reference image $\tilde{x}_{n+1} =$

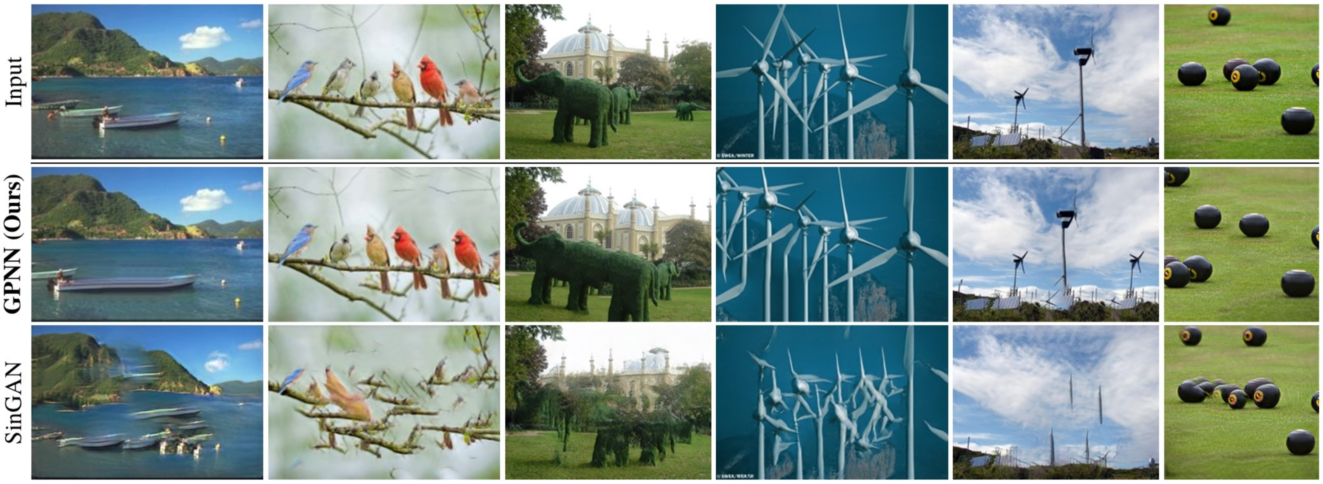

Figure 4: **Random Instance Generation Comparison:** *(Please zoom in) Images generated by our method are compared with images generated by SinGAN [25]. Images generated by GPNN (2nd row) look very realistic, whereas SinGAN produces many artifacts (3rd row).*

$x_{n+1}\uparrow^r = (x_n\downarrow_r)\uparrow^r$, and replace it by 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 [28] as visual *completeness*. In [28], completeness is enforced using an iterative optimization process over the pixels. InGAN [26] 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:** PNN receives a sharp source image x_n and an initial guess $\tilde{y}_{n+1}$ (which is an upscaled version of the generated output from the previous scale, hence somewhat blurry). Patches from the initial guess are extracted into the *query* pool of patches (denoted as Q). Nearest neighbors of the blurry query patches are searched in the similarly-blurry image obtained by upscaling the coarser source image, $\tilde{x}_{n+1}$ (Fig. 2(b)). Its patches are denoted as the *key* pool of patches (K). The corresponding sharp patches are then extracted from the sharp source image x_n, and are denoted as the *value* pool of patches (V). The only exception is the coarsest image scale (n=N), where we use the value patches as keys ($K = V$). The keys and values are ordered in the same way to maintain correspondences (patches from the same location have the same index in both pools). Patches are overlapping, so the same pixel can appear in multiple patches.

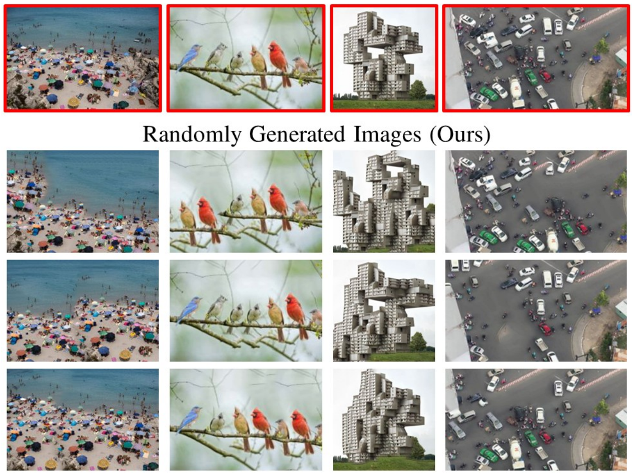

Figure 5: **Diverse Image Generation:** *(Please zoom in) Random images produced by GPNN from a single input (marked in red).*

2. **Compute Distances:** The MSE distance between each query patch Q_i and each key patch K_j is computed, and stored in the distances matrix $D_{i,j}$. We utilize the parallel nature of computing $D_{i,j}$, and run it on GPU for speed.

3. **Compute Normalized Scores:** To encourage visual completeness, PNN uses a similarity score that favors key patches that are missing in the queries. This increases their chance to be chosen and appear in the output, and thereby improve *Completeness*. The score normalizes the distance with a per-key factor:

$$S_{i,j} = \frac{D_{i,j}}{\alpha + \min_\ell D_{\ell,j}} \qquad (1)$$

Intuitively, when a key patch K_j is missing in the queries, the normalization term would be large and the score would be smaller. On the other hand, when a key patch appears in the queries, the normalization factor would get closer to α. The parameter α is used 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.

Dataset	Method	SIFID ↓	NIQE ↓	Confusion-Paired [%] ↑		Confusion-Unpaired [%] ↑		Realism competition [%] ↑	Diversity	Runtime ↓
		[25]	[22]	Time Limit	No Time Limit	Time Limit	No Time Limit	GPNN vs. SinGAN	[25]	[sec]
Places50 [37, 25]	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 [25], and our new SIGD dataset (see text). We use a variety of measures: NIQE (unpaired image quality assessment) [22], SIFID - single image FID [25], and human evaluations through an extensive user-study (see text). We repeat the evaluation for multiple diversity levels (measured as proposed by [25]). The table shows GPNN outperforms SinGAN by a large margin in every aspect: Visual quality, Realism, and Runtime.*

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:** We replace each query patch Q_i 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 that appear in multiple overlapping patches are aggregated using a gaussian weighted-mean.

Note that combining very different overlapping patches may cause inconsistencies and artifacts. To mitigate these inconsistencies, PNN is applied T times at each scale (in our implementation T=10). In the first iteration, the initial guess is as explained above. In further iterations, the previous output (without upscaling) is used as initial guess.

Runtime: A key advantage of GPNN over GAN-based methods (e.g., SinGAN [25], InGAN [26], Structural Analogies [3]) is its short 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 about 1 *hour* to generate a 180×250 sized image, whereas GPNN does so in 2 *seconds* (see Table 1).

3. Results & Evaluation

We evaluate and compare the performance of GPNN to SinGAN [25] on the main application of *random image generation*. We first follow the exact same evaluation procedure of SinGAN [25], 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 supplementary material. Runtime of GPNN is shown to be 3 orders of magnitude faster.

Data: We evaluate our performance on 2 datasets. The first is the set of 50 images used in SinGAN [25] for their evaluation (50 images from Places365 dataset [37]). The second is a new benchmark 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 [25], 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 by different methods.

Visual Results: Fig. 5 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 (e.g., diverse architectures of the building), while maintaining the above 2 characteristics. Fig. 1 (top-left) further shows results of GPNN on SinGAN's pivotal examples in their paper [25]. Fig. 4 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 supplementary material. We use the SIFID measure [25] to measure the distance between the source and the generated image patch distributions, as well as NIQE [22] 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 [25] & 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-

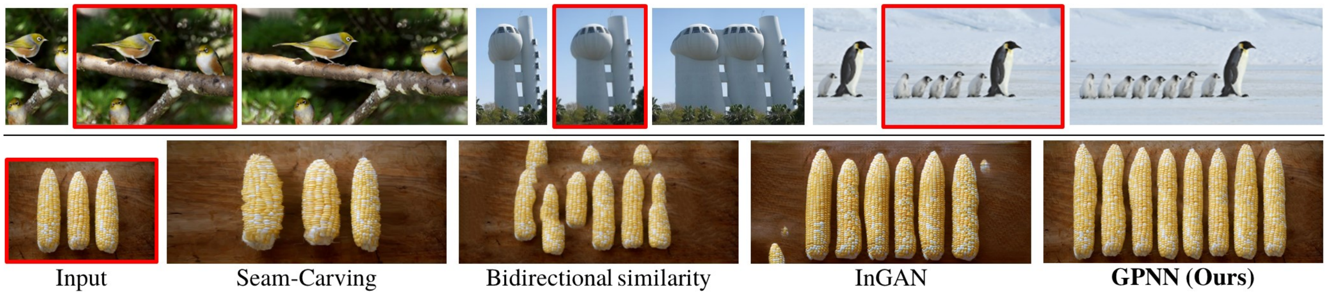

Figure 6: **Retargeting:** *(Please zoom in) Top rows show retargeted images by our method. Patch distribution is kept when retargeting to various target shapes. Bottom row shows comparison with previous patch-based [1],[28] and GAN-based [26] methods.*

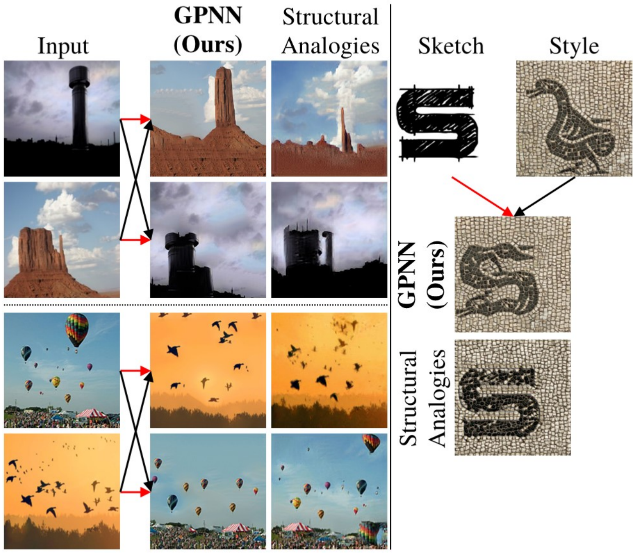

Figure 7: **Structural Analogies:** *(Please zoom in) Red arrow indicates the 'structure', while black arrow indicates the 'source' (defining the patch distribution to match). GPNN is compared to [3]. GPNN can also generate sketch-to-image instances (right).*

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 [25], which flashed each image for only 1 second ("Time limit"). We argue that $1sec$ makes it hard for raters to notice differences, resulting in a strong bias towards chance (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 (old and new), 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 level $\tilde{y}_{N+1}$, and the choice of the hyper-parameter α in Eq. 1. We next describe each application, along with its design choices.

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) [26, 28, 2]. GPNN starts by naively resizing the input image to the target size, and downscale it by a factor of r^N, this is inserted as the initial guess $\tilde{y}_{N+1}$. The pyramid depth N might change according to the objects size in the image, but it's usually chosen such that the smaller axis of the image at the coarsest pyramid level is roughly ×4 GPNN's patch size p. For retargeting, we wish to retain as much visual information as possible from the source image, and hence α in Eq. 1 is set to a small value (e.g., α = .005), thus promoting "Completeness". To get better final results, the described process is done gradually (similarly to [28]), i.e. resizing in a few small steps. Results for retargeting can be seen in Fig. 6 and 1. Fig. 6 further compares the performance of our method with that of [26, 28, 1]. 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 [29, 10, 17, 39, 16, 14, 18, 8]. 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. Our 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 pyramid depth N may change between pairs of images according to the change in object sizes. The output should contain as much visual data from A (e.g., in the bottom-left pair in Fig. 7, it is desired that many types of balloons will appear in the output), hence we set α in Eq. 1 to be small (e.g., α = .005). Finally, for refinement of the output, the output is downscaled by r^N and re-inserted to GPNN. Results can be found in Fig. 7. Our method creates new objects at the locations of objects of image B, while the output image seems to be from the distribution of image 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. 7). In addition to providing 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, and the missing part should be reconstructed. 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 GPNN gets the mask of the occluded part, hence does not use patches which originated from there. The initial guess $\tilde{y}_{N+1}$ 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 image is roughly $p \times p$ (the size of a single patch). 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. 1. Fig. 1, show 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 [28], 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 ("Completeness"), yet, no new visual artifacts that did not exist in the inputs are introduced in the output ("Coherence"). We create the initial guess $\tilde{y}_{N+1}$ by first naively concatenating the input images. 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. 8 shows a collage produced by GPNN, on an example taken from [28]. Compared with [28], our results are sharper and more faithful to the inputs.

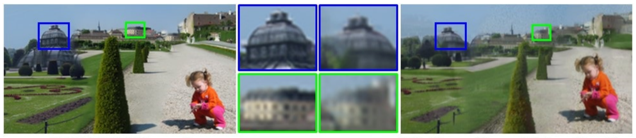

Figure 8: **Collage:** *Multiple input images are seamlessly combined into a single coherent output, maintaining visual information from all inputs. Note the higher quality of GPNN compared to Bidiectional-Similarity [28]. The 3 input images are found in Fig.1.*

Image Editing: In image editing/reshuffling [28, 25], 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. 1 to be large. The depth N of pyramid is set such that the edited region covers roughly the size of a single patch ($p \times p$) in the coarsest scale. 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. 9 shows our editing results compared to SinGAN's [25]. 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*.

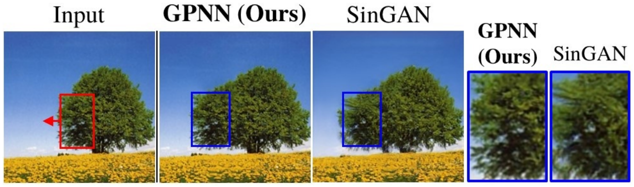

Figure 9: **Image Editing:** *A naively edited image is injected to our pyramid of PNNs. Compared with the results of [25].*

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. 4, 9 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 [28], or in an optimization-free manner by using GPNN's *patch-specific* normalized score (Eq. (1)). InGAN [26] 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. (1) 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. 4 and in Supplementary-Material.

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 [33]). While GANs may support a few discrete levels of diversity (e.g., [25] demonstrates 2 diversity levels), this diversity is not adjustable.

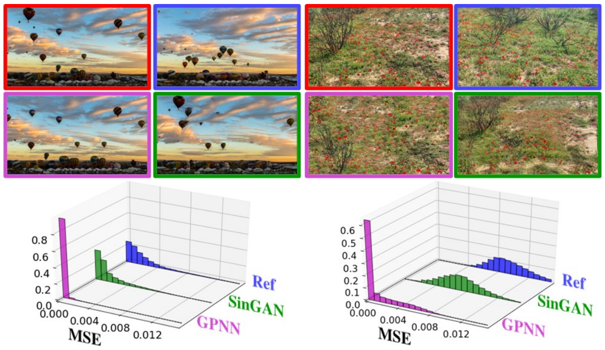

Figure 10: **Are GANs "Fancy Nearest Neighbors Extractors"?** *Input image (red), reference (blue), GPNN generated (purple), SinGAN generated (green). Please see text for details.*

Limitations

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. Such generalization can be advantageous (e.g., image harmonization [25]), but may also be disadvantageous, as it frequently generates undesired artifacts (see above).

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 [25], or smooth resizing animation [26]. In contrast, nearest neighbor search is discrete in nature. This prevents naively performing continuous interpolation or animation in classical patch-based methods.

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 (a single image), providing an excellent opportunity to quantitatively examine this claim. Fig. 10 shows two generated random instances of the same input image (red), once using SinGAN (green) and once using GPNN (purple). We also added a reference image (blue) of the same scene taken a slightly different position/time representing an “ideally” new generated instance. We measured the distance between the generated patches to their nearest neighbors in the input image and plotted the histograms of these distances (MSE). We repeated this experiment twice for 2 different input images. It is easy to see that the GAN-generated patch distribution behaves more like the reference distribution than the patch nearest neighbor generated distribution. That is, GANs are capable of generating new samples 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 has higher fidelity to the input (the histogram is peaked at zero), resulting with higher output quality, but at a limited ability to generate novel patches.

6. Acknowledgements

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). Dr Bagon is a Robin Chemers Neustein AI Fellow.

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, 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

[8] Michael Elad and Peyman Milanfar. Style transfer via texture synthesis. *IEEE Transactions on Image Processing*, 26(5):2338–2351, May 2017. 3, 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] 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

[13] 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

[14] 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

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

[16] 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

[17] 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

[18] 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

[19] 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

[20] 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

[21] 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

[22] 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

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

[24] Yi Ren, Yaniv Romano, and Michael Elad. Example-based image synthesis via randomized patch-matching, 2016. 1, 3

[25] 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, 3, 4, 5, 6, 7, 8

[26] 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, 5, 6, 8

[27] 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

[28] 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, 3, 4, 6, 7, 8

[29] 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

[30] 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

[31] 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

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

[33] 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

[34] 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

[35] 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

[36] 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

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

[38] 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

[39] 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

[40] 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