Computers & Graphics 98 (2021) 188–196

Contents lists available at ScienceDirect

Computers & Graphics

journal homepage: www.elsevier.com/locate/cag

Technical Section

Neural architecture search for deep image prior☆

Kary Ho[a], Andrew Gilbert[a,*], Hailin Jin[b], John Collomosse[a,b]

[a] CVSSP, University of Surrey, UK
[b] Creative Intelligence Lab, Adobe Research, USA

ARTICLE INFO

Article history:
Received 15 December 2020
Revised 7 May 2021
Accepted 29 May 2021
Available online 10 June 2021

Keywords:
Machine learning
Network architecture search
Super-resolution
Inpainting

ABSTRACT

We present a neural architecture search (NAS) technique to enhance image denoising, inpainting, and super-resolution tasks under the recently proposed Deep Image Prior (DIP). We show that evolutionary search can automatically optimize the encoder-decoder (E-D) structure and meta-parameters of the DIP network, which serves as a content-specific prior to regularize these single image restoration tasks. Our binary representation encodes the design space for an asymmetric E-D network that typically converges to yield a content-specific DIP within 10–20 generations using a population size of 500. The optimized architectures consistently improve upon the visual quality of classical DIP for a diverse range of photographic and artistic content.

1. Introduction

Many common image restoration tasks require the estimation of missing pixel data: denoising and artifact removal, inpainting, and super-resolution. Usually, this missing data is estimated from surrounding pixel data, under a smoothness prior. Recently it was shown that the architecture of a randomly initialized convolutional neural network (CNN) could serve as an effective prior, regularizing estimates for missing pixel data to fall within the manifold of natural images. This regularization technique referred to as the *Deep Image Prior* (DIP) [1], exploits both texture self-similarity within an image and the translation equivariance of CNNs to produce competitive results for the image restoration tasks mentioned above. However, the efficacy of DIP is dependent on the architecture of the CNN used; different content requires different CNN architectures for excellent performance and care over meta-parameter choices, e.g., filter sizes, channel depths, epoch count [2].

This paper contributes an evolutionary strategy to automatically search for the optimal CNN architecture and associated meta-parameters given a single input image. The core technical contribution is a genetic algorithm (GA) [3] for representing and optimizing a content-specific network architecture for use as the DIP regularizer in image restoration tasks. DIP depends upon a hand-designed CNN structure as a prior image reconstruction (as noted in the original DIP paper). Therefore, DIP is only useful with a human in the loop, optimising the CNN to get good results. We show that superior results are achieved through architecture search versus standard DIP backbones (or random architectures). We contribute a well-designed binary search space for NAS to automate this, making DIP practical for all 3 tasks originally proposed: in-painting, denoising, and upscaling, and finding architectures that outperform each of the hand-tuned examples for all images in the original DIP paper. Figure 1 contrasts the output of classic DIP [1] with our neural architecture search (NAS-DIP). Unlike image classification, to which NAS has been extensively applied [4–7], optimizing architecture encoder-decoder (E-D) networks for image reconstruction tasks is under-researched. Under DIP, an E-D network (whose architecture we seek) is overfitted to reconstruct the input image from a random noise field, acquiring a generative model of that image's structure. Parallel work in GANs [8] has shown that architectural design, particularly of the decoder, is critical to learning a sufficient generative model and that such architectures are content-specific.

Further, we demonstrate that – whilst the optimized architectures are content-specific – common DIP architectures are evolved for common visual styles of an image; for example, DIP for image reconstruction of painterly images shared similar backbones (shown later in Fig. 5). This unique insight enables clustering of visual style without supervision. It is exploited practically to speed-up the search via warm-start for known content types. For example, if an image is detected as a painting, then the search space can be initialized to a known good distribution for painterly styles. This is important, as a common challenge with NAS methods is that they are generally slow to converge.

☆ This article was recommended for publication by M Kim.

* Corresponding author.
E-mail address: a.gilbert@surrey.ac.uk (A. Gilbert).

https://doi.org/10.1016/j.cag.2021.05.013
0097-8493/

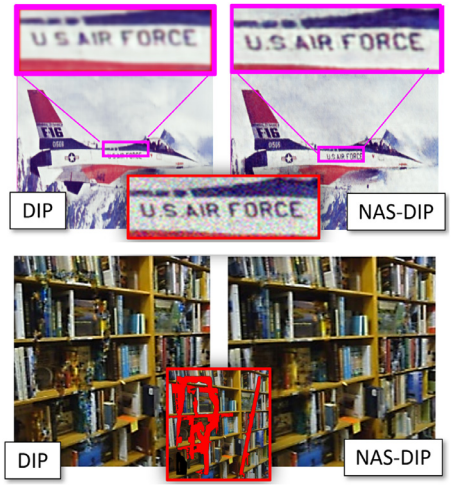

Fig. 1. Neural Architecture Search yields a content-specific deep image prior (NAS-DIP) to enhance DIP for image restoration tasks e.g. denoising (top) and inpainting (bot.).

Our novel technical contributions are as follows:

1. Representation and method for **evolutionary neural architecture search** of encoder-decoder architectures for DIP, leveraging a state-of-the-art perceptual metric to guide the optimization [9]. This enables **DIP to be applied fully automatically**, rather than relying on hand-tuning of the DIP architecture for each image (as [1,2]).
2. **State of the art DIP results** (PSNR/SSIM) for all 3 of the tasks (inpainting, denoising, up-scaling) proposed in the original DIP paper [1], beating the hand-optimized DIP architectures proposed [2].
3. **Demonstrating the content-*style*dependency of DIP architectures**, which opens an efficient route to the application of our work, initializing NAS using pre-identified distributions of architectures for particular content styles.

2. Related work

Neural architecture search (NAS) seeks to automate the design of deep neural network architectures through data-driven optimization [10], most commonly for image classification [5,11], but recently also object detection [11] and semantic segmentation [12]. NAS addresses a facet of the automated machine learning (AutoML) [13] problem, which more generally addresses hyper-parameter optimization and tuning of training meta-parameters.

Early NAS leveraged Bayesian optimization for MLP networks [14], and was extended to CNNs [15] for CIFAR-10. In their seminal work, Zoph and Le [16] applied reinforcement learning (RL) to construct image classification networks via an action space, tokenizing actions into an RNN-synthesised string, with reward, driven by validation data accuracy. The initially high computational overhead (800GPUs/4 weeks) was reduced while further enhancing accuracy. For example, by exploring alternative policies such as proximal policy optimization [17] or Q-learning [18], RL approaches over RNN now scale to contemporary datasets, e.g. NASNet for ImageNet classification [11,19] and was recently explored for GAN over CIFAR-10 [20]. Cai et al. [21] similarly tokenizes the architecture but explore the solution space via sequential transformation of the string via function-preserving mutation operations on an LSTM-learned embedding of the string. Our work also encodes architecture as an (in our case, binary) string but optimizes via an evolutionary strategy rather than training a sequential model under RL. **Evolutionary strategies** for network architecture optimization were first explored for MLPs in the early nineties [22]. Genetic algorithms (GAs) were used to optimize both the architecture and weights [23,24], rather than rely upon back-prop to reduce the GA evaluation bottleneck; however, this is not practical for contemporary CNNs. While selection and population culling strategies [4,5,7,25] have been explored to reduce computational costs in developing high performing image classification networks over ImageNet. Similarly, a Super-Net based one-shot NAS [26] uses a multi-path sampling strategy with rejection to greedily filter the weak paths.

Our work contributes to the much sparser body of research optimizing E-D networks [27,28]. It is unique in that we explore image restoration architectures via GA optimization, and as such, our architecture representation differs from prior work, including very recent, contemporaneous work [29].

Single image restoration has been extensively studied in classical vision and deep learning, where priors are prescribed or learned from representative data. A common prior to texture synthesis is the Markov Random Field (MRF), in which the pairwise term encodes spatial coherence. Several inpainting works exploit MRF formulations of this kind [30–32], including methods that source patches from the input [33] or multiple external [34,35] images, or use random propagation of a few good matched patches [36]. Patch self-similarity within single images has also been exploited for single image super-resolution [37] and denoising. The DIP [1] (and its video extension [38]) exploit translation equivariance of CNNs to learn and transfer patches within the receptive field. Very recently, single image GAN [39] has been proposed to learn a multi-resolution model of appearance, and DIP has been applied to image layer decomposition [40]. Our work targets the use cases for DIP proposed within the original paper [1], namely super-resolution, denoising, and region inpainting. Generative Adversarial Networks (GANs) are more widely used for inpainting and super-resolution [41,42] by learning structure and appearance from a representative corpus image data [43], in some cases explicitly maintaining both local and global consistency through independent models [44]. Our approach and the DIP approach differs in that we do not train a network to perform a specific task. Instead, we use an untrained (randomly initialized) network to perform the tasks by overfitting such a network to a single image under a task-specific loss using neural architecture search.

3. Architecture search for DIP

The core principle of DIP is to learn a generative CNN G_θ (where θ are the learned network parameters e.g. weights) to reconstruct x from a noise field $\mathcal{N}$ of identical height and width to x, with pixels drawn from a *uniform* random distribution. Ulyanov [1] propose a symmetric encoder-decoder network architecture with skip connections for G_θ, comprising five pairs of (up-)convolutional layers with varying architectures depending on the image restoration application (denoising, inpainting or super-resolution) and the image content. A reconstruction loss is applied to learn G_θ for an image x:

$$\theta^* = \arg\min_\theta \theta \|G_\theta(\mathcal{N}) - x\|_2^2. \quad (1)$$

Our core contribution is to optimize not only for θ but also for architecture G using a genetic algorithm (GA), guided via a perceptual quality metric [9], as now described. For clarity, we use 'epoch' to refer to a network architecture training cycle proposed by NAS. We use 'generation' to refer to an iteration of the genetic algorithm (GA) used to discover those architectures.

3.1. Network representation

We encode the space of encoder-decoder (E-D) architectures from which to sample G as a constant length binary sequence,

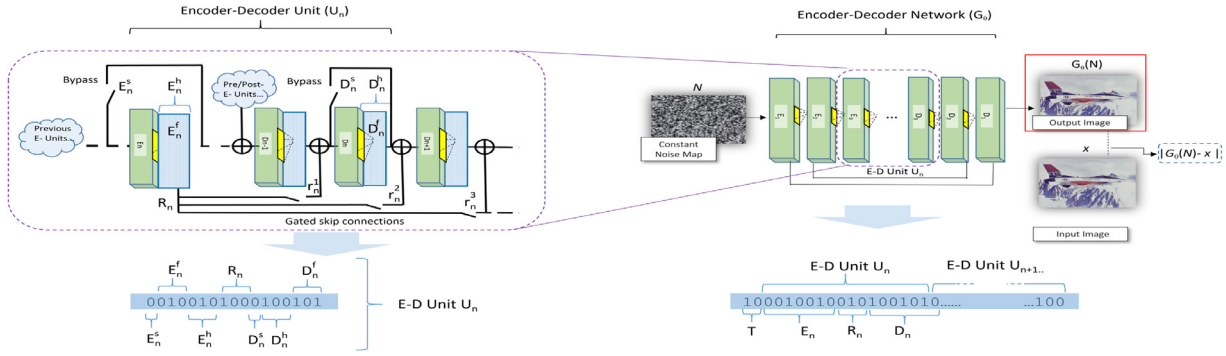

Fig. 2. Architecture search space of NAS-DIP(-T). The Encoder-Decoder (E-D) network G (right) is formed of several E-D Units (U_n) each with an Encoder E_n) and Decoder D_n paired stage (zoomed, left) represented each by 7 bits plus an additional $4N$ bits R_n to encode gated skip connections from E_n to other decoder blocks in the network. Optionally the training epoch count T is encoded (Section 3.1). Under DIP images are reconstructed from constant noise field N by optimizing to find weights θ thus overfitting the network to input image x under reconstruction loss e.g. here for denoising (Eq. (3)).

representing N paired encoder-decoder units $U = \{U_1, ..., U_N\}$. Following [1], G is a fully convolutional E-D network and optimize for the connectivity and meta-parameters of the convolutional layers. A given unit U_n comprises encoder E_n and decoder D_n convolutional stages denoted E_n and D_n respectively each of which requires 7 bits to encode its parameter tuple. Unit U_n requires a total $14 + 4N$ bits to encode, as an additional $4N$-bit block for the unit, encodes a tuple specifying the configuration of skip connections from its encoder stage to each of the decoder stages i.e both within itself and other units. Thus, the total binary representation for an architecture in neural architecture search for DIP (**NAS-DIP**) is $N(14 + 4N)$. For our experiments, we use $N = 6$ but note that the effective number of encoder or decoder stages varies according to skip connections. Figure 2 illustrates the organization of the E-D units (right) and the enlarged detailed architecture of a single E and resultant 3 D units (left). Each unit U_n comprises the following binary representation, where super-scripts indicate elements of the parameter tuple:

$\mathrm{E}_n^s \in [0, 1]$ (1 bit) a binary indicator of whether the encoder stage of unit U_n is skipped (bypassed). $\mathrm{E}_n^f \in [0, 7]$ (3 bits) encoding filter size $f = 2E_n^f + 1$ learned by the convolutional encoder. $\mathrm{E}_n^h \in [0, 7]$ (3 bits) encoding number of filters $h = 2^{E_n^h - 1}$ and so channels output by the encoder stage. $\mathrm{D}_n^s \in [0, 1]$ (1 bit) a binary indicator of whether the decoder stage of unit U_n is skipped (bypassed). $\mathrm{D}_n^f \in [0, 7]$ (3 bits) encoding filter size $f = 2D_n^f + 1$ learned by the up-convolutional decoder stage. $\mathrm{D}_n^h \in [0, 7]$ (3 bits) encoding number of filters $h = 2^{D_n^h - 1}$ and so channels output by the decoder stage. $\mathrm{R}_n \in \mathbb{B}^{4N}$ (4N bits) encodes gated skip connections as $[\rho_n^1, ..., \rho_n^N]$; each 4-bit group $\rho_n^i \in [0, 15]$ determines whether gated skip path r_n^i connects from E_n to D_i and if so, how many filters/channels (i.e skip gate is open if $r_n^i = 0$).

NAS-DIP-T. We explore a variant of the representation encoding maximum epoch count $T = 500 * (2^t - 1)$ via two additional bits coding for t and thus a representation length for NAS-DIP-T of $N(14 + 4N) + 2$.

Symmetric NAS-DIP. We also explore a compact variant of the representation that forces $E_n = D_n$ for all parameters, forcing a symmetric E-D architecture to be learned and requiring only 10 bits to encode U_n and compare to asymmetric architectures in Section 4.2.

3.2. Evolutionary optimization

DIP provides an architecture specific prior that regularises the reconstruction of a given source image x from a uniform random noise field $\mathcal{N}$. Fixing N constant, we use a genetic algorithm (GA) to search architecture space for the optimal architecture G^* to recover a 'restored' e.g. denoised, in-painted or upscaled version $\hat{x}$ of that source image:

$$\hat{x} = G^*_{\theta^*}(\mathcal{N}). \tag{2}$$

GAs simulate the process of natural selection by breeding successive generations of individuals through the processes of crossover, fitness-proportionate reproduction, and mutation. In our implementation, such individuals are network configurations encoded via our binary representation. Figure 3 illustrates NAS-DIP convergence for inpainting task on BAM! [45]. The left illustrating the Input (top) and converged (bottom) result at generation 13, together with samples of the top, middle and bottom performing architecture results.

Individuals are evaluated by running a pre-specified DIP image restoration task using the encoded architecture. We consider restoration tasks (denoising, inpainting, super-resolution) in which an ideal $\hat{x}$ is unknown (is sought) and so a proxy must guide the GA search. In lieu of this ground truth we employ a trained perceptual measure (Section 3.2.1) to assess the visual quality generated by any candidate architecture by training $G^*_\theta(N)$ via backpropagation to minimize a task specific reconstruction loss:

$$\mathcal{L}_{de-noise}(x; G) = \min_\theta ||G_\theta(\mathcal{N}) - x||_2^2. \tag{3}$$

$$\mathcal{L}_{in-paint}(x; G) = \min_\theta ||M(G_\theta(\mathcal{N})) - M(x)||_2^2. \tag{4}$$

$$\mathcal{L}_{upscale}(x; G) = \min_\theta ||D(G_\theta(\mathcal{N})) - x||_2^2. \tag{5}$$

Where D is a bi-linear downsampling operator reducing its target to the size of x and $M(.)$ is a masking operator that returns zero within the region to be in-painted.

We now describe a single iteration of the GA search, which is repeated until the improvements gained over the last few generations are marginal (the change in both average and maximum population fitness over a sliding window fall below a threshold).

3.2.1. Population sampling and fitness

We initialize a population of $K = 500$ possible network configuration solutions uniformly sampling $\mathbb{B}^{N(14+4N)}$ to seed initial architectures $\mathcal{G} = \{G_1, G_2, ..., G_K\}$. The visual quality of $\hat{x}$ under each architecture is assessed via the pairwise perceptual measure (LPIPS score) [9] against the source image as a proxy for individual fitness. We explored several scores from the GAN literature including Inception Score [46] and Frechet Inception Distance [47] but found pair-wise LPIPS to improve convergence of NAS-DIP.

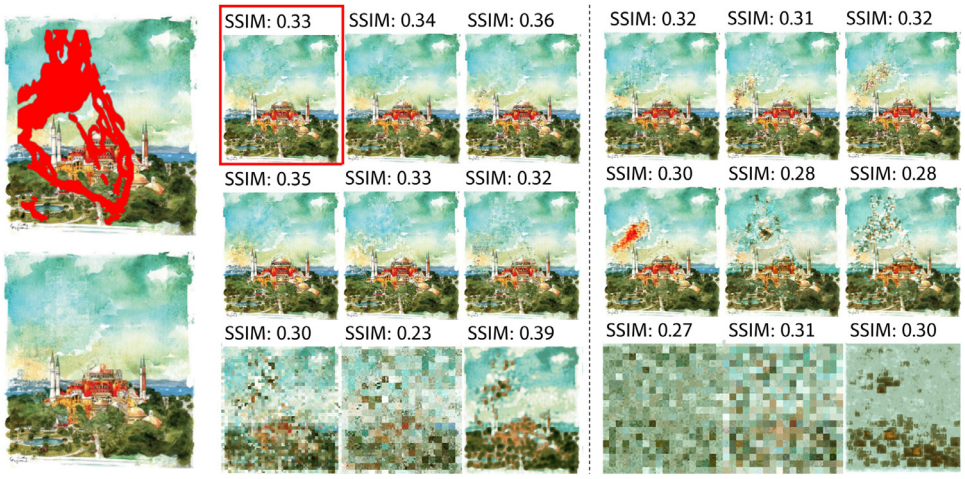

Fig. 3. NAS-DIP convergence for inpainting task on BAM! [45]. Left: Input (top) and converged (bottom) result at generation 13. Right: Sampling the top, middle and bottom performing architectures (shown in respective rows) from a population of 500 architectures, at the final (1st, middle) and final (13th, rightmost) generations. Inset scores: Fitness selection is driven using LPIPS [9]; evaluation by PSNR/SSIM.

Architectures that produce higher quality outputs are more likely to be selected for the next generation of solutions. Population or network configuration diversity is encouraged via the process's stochasticity and a random mutation into the offspring genome. We apply elitism; the bottom 5% network configurations or population is culled, and the top 5% pass unperturbed to the next generation – the fittest individual in successive generations is thus at least as fit as those in the past. The middle 90% is used to produce the remainder of the next generation. Two network configurations or individuals are selected stochastically with a bias to fitness $p(G_i) = f(G_i)/\sum_{j=1}^{K} f(G_j)$, and bred via cross-over and mutation (Section 3.2.2) to produce a novel offspring for the successive generation. This process repeats with replacement until the population count equals the previous.

3.2.2. Cross-over and mutation

Individuals are bred via genetic crossover; given two constant length binary genomes of form $G\{(E_1, R_1, D_1), ..., (E_N, R_N, D_N)\} \in \mathbb{B}^{N(14+4N)}$, a splice point $S = [1, N]$ is randomly selected such that units from a pair of parents A and B are combined via copying of units $U_{N<S}$ from A and $U_{N\geq S}$ from B. Such cross-over could generate syntactically invalid genomes, e.g., due to tensor size incompatibilities between units in the genome. During the evaluation, an invalid architecture evaluates to zero, prohibiting its selection for subsequent generations.

Population diversity is encouraged via the stochasticity of selection and introducing a random mutation into the offspring genome. Each bit within the offspring is subject to random flip with low probability p_m; we trade-off this rate against convergence in Section 4.2.2.

3.2.3. Efficient deployment

Architectures discovered by NAS-DIP are visualized in Fig. 5, using t-SNE projection in the architecture space ($\mathbb{B}^{N(14+4N)}$) for a solution population at convergence (generation 20) for a representative inpainting task for which symmetric E-D network was sought. This grouping within the style genre classes demonstrates that the NAS-DIP is able to inpaint within a broad range of styles. The evolutionary optimization process typically converges within 20 epochs, resulting in more robust candidate architectures within the search space.

We observe that similar kinds of visual content repeatedly result in similar clusters of architecture emerging from the NAS; Fig. 5 visualizes 80 best performing networks for inpainting BAM! artworks of identical content (flowers) but exhibiting eight different artistic styles. Each image has been run through NAS-DIP under loss Eq. (4). This result reinforces our main scientific contribution of automatic discovery of CNN architectures for DIP via evolutionary NAS. However also presents a practical route to engineering an efficient deployment of our technique, which, like all NAS, requires lengthy search times (in the order of a few hours) to find the best architectures for a given image and reconstruction task.

We cache the results of NAS-DIP for images of several visual styles (we use the 8 BAM! style classes defined in [35]). We use this distribution of architectures to seed the initial population rather than random sampling, based on the visual content being reconstructed. Following [45] we train a ResNet50 classifier with mean Average Precision of (98.6, 97.3, 100.0, 98.5, 97.1, 89.5, 98.6, 97.4) for classes (3D, comic, graphite, oil, pen ink, photo, vector art, watercolor) respectively to decide on the relevant initial distribution of architectures, which can then be fine-tuned using just one epoch of NAS evolution in just 1-2 min; representative output is included in Fig. 6.

4. Experiments and discussion

We evaluate the proposed neural architecture search technique for DIP (NAS-DIP) for each of the three image restoration tasks proposed in DIP [1]: image inpainting, super-resolution, and denoising.

4.1. Datasets

NAS-DIP works by optimising over a single image, for the datasets, we evaluate over a number of public datasets: 1) Places2 [48]; a dataset of photos commonly used for inpainting (test partition sampled as [44]); 2) Behance Artistic Media (BAM!) [49]; a dataset of 8 media styles, test partition as [35]; 3) 9 denoising images and 11 inpainting images both from[1] as used by in the DIP work [1]; 4) Set14 [50] dataset for 4x Super resolution also used in the DIP work [1]. the dataset of images used to evaluate the original DIP algorithm. Where baseline comparison is made to images from the latter, the specific network architecture is replicated according to Ulyanov et al. [2] and the authors' public implementation. Results are quantified via three objective metrics; PSNR (as in DIP) and structural similarity (SSIM) [51] are

[1] http://www.cs.tut.fi/~foi/GCF-BM3D/index.html#ref_results

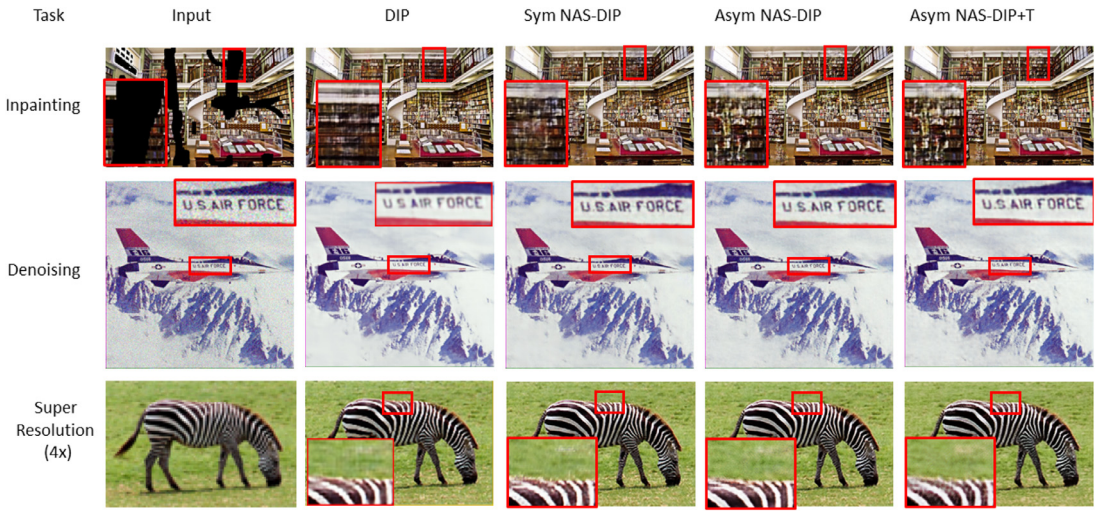

Fig. 4. Evaluation of proposed NAS-DIP (and variants) vs classical DIP [1], for inpainting, denoising and 4× super-resolution on the Library, Plane and Zebra images of the *DIP* dataset.

Table 1
Per-dataset performance of NAS-DIP (asymmetric) versus DIP, over the dataset proposed in DIP, also BAM!, Places2.

Dataset	Task	PSNR ↑		LPIPS ↓		SSIM ↑	
		DIP	NAS-DIP	DIP	NAS-DIP	DIP	NAS-DIP
BAM! [45]	inpainting	16.8	**19.76**	0.42	**0.25**	0.32	**0.62**
Places2 [44]	inpainting	12.4	**15.43**	0.37	**0.27**	0.67	**0.90**
11 Images[1] [1]	inpainting	29.92	**30.40**	0.11	**0.07**	0.84	**0.89**
9 Images[1] [1]	denoising	29.22	**30.42**	0.09	**0.07**	0.87	**0.91**
Set14 [50]	super-res x4	27.00	**28.45**	0.15	**0.11**	0.87	**0.91**

Table 2
Detailed quantitative comparison of visual quality under three image restoration tasks for the DIP dataset. Comparison between the architecture presented in the original DIP for each image, with the best architecture found under each of three variants of the proposed method: NAS-DIP, NAS-DIP-T, and NAS-DIP constrained to symmetric E-D (Section 3.1). Quality metrics: LPIPS [9] (used in NAS objective); and both PSNR and SSIM [51] as external metrics.

Task	Image	PSNR ↑				LPIPS ↓				SSIM ↑			
		DIP	Sym NAS-DIP	Asym NAS-DIP	Asym NAS-DIP-T	DIP	Sym NAS-DIP	Asym NAS-DIP	Asym NAS-DIP-T	DIP	Sym NAS-DIP	Asym NAS-DIP	Asym NAS-DIP-T
inpainting	Vase	18.3	29.8	29.2	**30.2**	0.76	0.02	**0.01**	0.02	0.48	0.95	**0.96**	0.95
	Library	19.4	19.7	**20.4**	20.0	0.15	0.10	**0.09**	0.12	0.83	0.81	**0.84**	0.83
	Face	33.4	34.2	**36.0**	35.9	0.078	0.03	**0.01**	0.04	0.95	0.95	**0.96**	0.95
denoising	F16	23.1	25.8	25.7	**25.9**	0.15	0.096	0.09	**0.07**	0.85	0.90	0.92	**0.94**
	Snail	12.0	12.6	12.2	**12.7**	0.11	0.12	0.09	**0.06**	0.61	0.52	0.74	**0.84**
super-res	Zebra 4x	25.7	**26.2**	25.6	26.0	0.19	**0.13**	0.14	0.14	0.67	0.51	**0.75**	0.72
	Zebra 8x	20.6	**22.6**	21.96	**22.6**	0.57	0.29	**0.19**	0.20	0.28	0.34	**0.48**	0.47

used to evaluate against ground truth, and the perceptual metric (LPIPS) [9] used as fitness score in the GA. Note that during training, we optimise over the LPIPS metric as the fitness score of candidate networks and therefore the main performance metrics are PSNR and SSIM, which are not used within the optimisation process.

4.1.1. Training details

We implement NAS-DIP in Tensorflow, using ADAM and a learning rate of 10^{-3} for all experiments. For NAS-DIP, the epoch count is 2000; otherwise, this and other metaparameters are searched via the optimization. To alleviate the evaluation bottleneck in NAS (which takes, on average, 2-3 min per architecture proposal), we distribute the evaluation step over 16 Nvidia Titan-X GPUs capable of each evaluating two proposals concurrently for an average NAS-DIP search time of 3-4 h total (for an average of 10–20 generations to convergence). Our DIP implementation extends the authors' public code for DIP [1]. Reproducibility is critical for NAS-DIP optimization; the same genome must yield the same perceptual score for a given source image within the generations. In addition to fixing all random seeds (e.g., for batch sampling and fixing the initial noise field), we take additional technical steps to avoid non-determinism in cuDNN through the avoidance of atomic add operations in image padding present in original DIP code. If different initial seeds were used, there was little difference found in the final results, as generally, the populations converged within the 20 generations to the same/similar results.

4.2. Network representation

We evaluate three variants of the architecture representation proposed in Section 3.1: NAS-DIP, NAS-DIP-T (in which epoch count is also optimized), and a constrained version of NAS-DIP, forcing a symmetric E-D network. For all experiments, $N = 5$ enabling E-D networks of up to 10 (up-)convolutional layers with gated skip connections. Performance is evaluated using PSNR for comparison with original DIP [1] and SSIM and the LPIPS score used to guide NAS is also reported. Table 2 provides direct comparison on im-

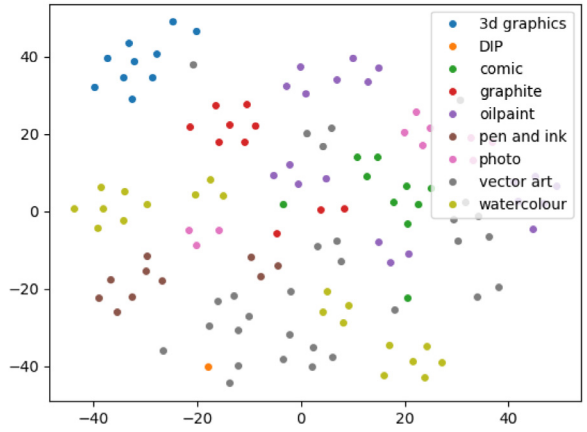

Fig. 5. NAS-DIP content specific architecture discovery: t-sne visualization of discovered architectures in $\mathbb{B}^{N(14+4N)}$ inpainting artwork from BAM! [45] (7 BAM styles + DIP photograph); common visual styles yield a common best performing architecture under NAS-DIP.

ages from Ulyanov et al. [1]; visual output and convergence graphs are shown in Fig. 4. Following random initialization, relative fitness gains of up to 30% are observed after 10–20 generations, after which performance converges to values above the DIP baseline [1] in all cases.

For both inpainting and denoising tasks, asymmetric networks are found that outperform any symmetric network, including the networks published in the original DIP for those images and tasks. For super-resolution, the symmetric network is found to exceed asymmetric networks and constraining the genome in this way aids in discovering a performant network. In all cases, it was unhelpful to optimize for epoch count T, despite prior observations on the importance of tuning T in DIP [1]. Table 1 broadens the experiment for NAS-DIP, averaging performance for three datasets: BAM! with 80 randomly sampled images, ten from each of the eight styles; Places2 with a 50 image subset included in [44]; DIP with 7 images used in [1]. For all 3 tasks and all 3 datasets, NAS-DIP discovers networks outperforming classic DIP [1]. Additional results of the denoising and 8x super-resolution can be seen in Fig. 8

Fig. 6. In-painted results (source/mask left, output right) of 7 BAM styles + DIP photograph for proposed NAS-DIP content specific architecture discovery.

4.2.1. Perceptual user inpainting study

We conducted a study via Amazon Mechanical Turk (AMT) to compare NAS-DIP performance versus the classic DIP baseline. Eighty random images sampled from BAM! (10 per 8 styles) were displayed next to each result in a random arrangement presented to 5 participants, yielding 400 annotations. Participants were asked to 'identify the highest quality image' without sight of the ground truth. A consensus threshold of 3 out of 5 (majority vote) was used to disregard results that failed to reach consensus. Our approach significantly outperforms the existing baselines DIP with results echoing Table 1 trends, with 80.9% preference for NAS-DIP.

4.2.2. Evaluating mutation rate

Population elitism requires a moderate mutation rate $p(r)$ to ensure population diversity and so convergence, yet raised too

Fig. 7. Result of varying mutation (bit flip) rate $p(r)$; improved visual quality (left, zoom recommended) is obtained with a mutation rate of $p(r) = 0.05$ equating to an average of 4 bit flips per offspring. Convergence graph for each image (right).

Fig. 8. Example denoising and super-resolution results for.

high convergence occurs at a lower quality level. We evaluated $p_m = \{0.01, 0.02, 0.05, 0.10\}$ observing that for all tasks, a bit flip probability of 5% (i.e. an average of 4 bit flips per offspring for $N = 5$) encourages convergence of the highest fitness value. This correlates to the highest visual quality according to both PSNR and SSIM external metrics. Figure 7 provides representation visual output alongside convergence graphs. In general, convergence is achieved in 10–20 generations taking a few hours on a single GPU for a population $K = 500$.

4.3. Content aware inpainting

To provide a qualitative performance of the Content-Aware inpainting, we compare our proposed approach against several current baselines: PatchMatch [36], a recent GAN inpainting work that uses millions of training images [44], and the Style-aware inpainting [35]. We compare against the same baselines using scenic photographs in Places2, Fig 10 presents a visual comparison over the

(a) Source & Mask (red) (b) PatchMatch (c) ImgComp (d) Style-aware (e) Proposed

Fig. 10. Qualitative comparison of NAS-DIP inpainting vs. 3 (trained) baselines; PatchMatch [36], ImgComp [44], Style-aware [35]. Zoom to view.

Table 3
SSIM result comparison between NAS-DIP and baselines over Places2 results in [44]. SSIM, higher is better.

	Method				
	IM[52]	PM [36]	CE [53]	GL [44]	Asym NAS-DIP
SSIM	0.76	0.88	0.74	0.89	**0.90**

image set included in [44] with the same mask regions, and the quantitative results are presented in Table 3. while the Places2 line in Table 1 indicates the quantitative performance of the approach on the Places2 dataset images.

4.4. Discovered architectures

The distribution of architectures discovered by NAS-DIP for a single representative image is visualized in Fig. 11. Using t-SNE projection in the architecture space ($\mathbb{B}^{N(14+4N)}$) for a solution population at convergence (generation 20) for a representative super-resolution task for which symmetric E-D network was sought. Distinct clusters of more robust candidate architectures have emerged with distinct skip activations and channel widths, color coding indicates fitness. This shows image quality to be conditioned on a complex mix of these parameters and not solely, for example, on network capacity (maximum channel width). While population at convergence is multi-modal, with no single converged network configuration identified.

We evaluate multiple tasks for a single given image. Figure 9 shows best and worst performing architectures, comparing how characteristics of architectures differ for the three different tasks of inpainting, super-resolution and denoising (note for fairness of comparison, we fix the E-D network to symmetric forcing the same layer count $N = 5$ for each task. Note that higher capacity networks are learned for inpainting. All successful networks smoothly increase/decrease channel count with layer depth versus the more irregular arrangement of worse perform-

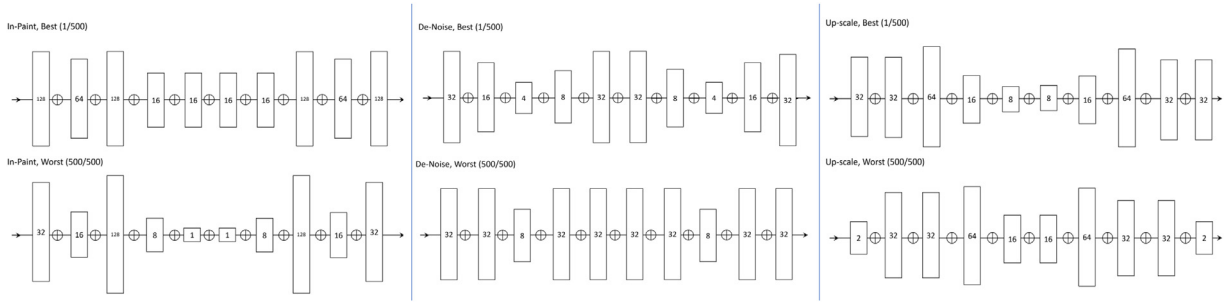

Fig. 9. Comparing structure and depth of best vs. worst discovered architectures at convergence for a single image (fixed $N = 5$ symmetric E/D layer representation for fair comparison) for each of the 3 tasks. Architecture space ($\mathbb{B}^{N(14+4N)}$).

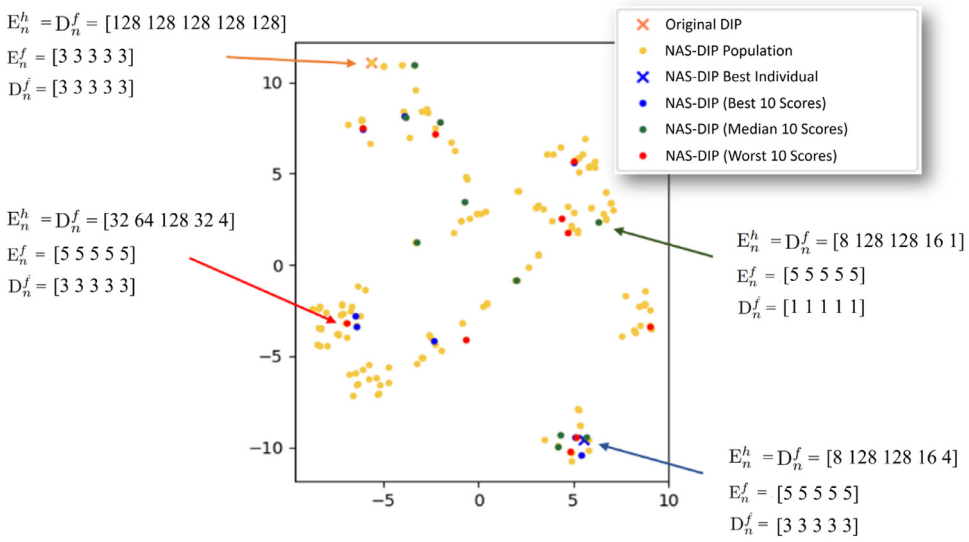

Fig. 11. t-SNE visualizations in architecture space ($\mathbb{B}^{N(14+4N)}$) for NAS-DIP, $N = 5$ symmetric network: Population at convergence is multi-modal (super-resolution).

ing networks. Compared to the default architectures in classic DIP [2] there is an increase in skip connections and diversity of channel depths that would be hard to tune manually.

Next, we explore multiple images for a single given task. Figure 5 visualizes, via t-SNE, the best performing architectures in $B^{N(14+4N)}$ for inpainting 80 images, sampled from BAM! [45]. The images contain similar content (flowers) but exhibit different styles (10 each of 8 different styles). Each image has been run through NAS-DIP under loss Eq. (4); representative output is included in Fig. 6. The plot reveals style-specific clusters; pen-and-ink, graphite sketches, 3D graphics renderings, comics, and vector artwork all form distinct groups while others, e.g., watercolor, form several clusters. We conclude that images that share a common visual style exhibit commonality in network architecture necessary to perform image reconstruction well. This suggests a future application for NAS-DIP beyond NAS to unsupervised clustering of style.

5. Conclusion

We reported neural architecture search (NAS) for image reconstruction under the recently proposed DIP [1], learning a content-specific prior for a given source image in the form of an encoder-decoder (E-D) network architecture. Following the success of evolutionary search techniques for image classification networks [4,5] we leveraged a genetic algorithm to search a binary architecture space. We demonstrated its efficacy for image denoising, inpainting, and super-resolution. For the latter case, we observed a constrained version of our genome yielding symmetric networks exceeded that of asymmetric networks, which benefited the other two tasks. In all cases, we observed the discovered networks' performance to significantly exceed classical DIP and the potential for content-specific architectures beyond image restoration to unsupervised style clustering. Future work could pursue explore further generalizations beyond fully convolutional E-Ds to incorporate pooling and normalization strategies.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

CRediT authorship contribution statement

Kary Ho: Conceptualization, Methodology, Software, Data curation, Writing - original draft. **Andrew Gilbert:** Conceptualization, Methodology, Software, Writing - original draft, Writing - review & editing, Supervision. **Hailin Jin:** Conceptualization, Methodology, Supervision. **John Collomosse:** Conceptualization, Methodology, Writing - review & editing.

Acknowledgments

The work was supported by Adobe Research.

References

[1] Ulyanov D, Vedaldi A, Lempitsky V. Deep image prior. In: The IEEE conference on computer vision and pattern recognition (CVPR); 2018.
[2] Ulyanov D, Vedaldi A, Lempitsky V. Deep image prior. Intl J Comput Vis (IJCV) 2019.
[3] de Jong K. Learning with genetic algorithms. J Mach Learn 1988;3:121–38.
[4] Real E, Moore S, Selle A, Saxena S, Suematsu Y, Le QV, et al. Large-scale evolution of image classifiers. In: Proc. ICLR; 2017.
[5] Real E, Aggarwal A, Huang Y, Le QV. Aging evolution for image classifier architecture search. In: Proc. AAAI; 2019.
[6] Liu C, Zoph B, Neumann M, Shelns J, Hua W, Li L, et al. Progressive neural architecture search. In: Proc. ECCV; 2018.
[7] Elsken T, Metzen J, Hutter F. Efficient multi-objective neural architecture search via lamarckian evolution. In: Proc. ICLR.
[8] Wojna Z, Ferrari V, Guadarrama S, Silberman N, Chen L, Fathi A, et al. The devil is in the decoder: Classification, regression and GANs. Intl J Comput Vis (IJCV) 2019;127:1694–706.
[9] Zhang R, Isola P, Efros A, Shechtman E, Wang O. The unreasonable effectiveness of deep features as a perceptual metric. In: Proc. CVPR; 2018.
[10] Elsken T, Metzen JH, Hutter F. Neural architecture search: a survey. J Mach Learn Res (JMLR) 2019b;20:1–21.
[11] Zoph B, Vasudevan V, Shlens J, Le QV. Learning transferrable architectures for scalable image recognition. In: Proc. CVPR; 2018.
[12] Chen L, Collins M, Zhu Y, Papandreou G, Zoph B, Schroff F, et al. Searching for efficient multi-scale architectures for dense image prediction. In: Proc. NeurIPS; 2018. p. 8713–24.
[13] Automated machine learning: methods, systems, challenges. Hutter F, Kotthoff L, Vanschoren J, editors. Springer; 2019. In press, available at http://automl.org/book.
[14] Bergstra J, Yamins D, Cox D. Making a science of model search: hyper-parameter optimization in hundreds of dimensions for vision architectures. In: Proc. ICML; 2013.
[15] Domhan T, Springenberg JT, Hutter F. Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves. In: Proc. IJCAI; 2015.
[16] Zoph B, Le QV. Neural architecture search with reinforcement learning. In: Proc. ICLR; 2017.

[17] Schulman J., Wolski F., Dhariwal P., Radford A., Klimov O.. Proximal policy optimization algorithms. 2017. arXiv preprint arXiv:1707.06347.
[18] Baker B, Gupta O, Naik N, Raskar R. Designing neural network architectures using reinforcement learning. In: Proc. ICLR; 2017.
[19] Negrinho R., Gordon G.. DeepArchitect: automatically designing and training deep architectures. 2017. arXiv preprint arXiv:1704.08792.
[20] Gong X, Chang S, Jiang Y, Wang Z. AutoGAN: neural architecture search for generative adversarial networks. In: Proc. ICCV; 2019.
[21] Cai H, Chen T, Zhang W, Yu Y, Wang J. Efficient architecture search by network transformation. In: Proc. AAAI; 2018.
[22] Miller GF, Todd PM, Hedge SU. Designing neural networks using genetic algorithms. In: Proc. Intl. Conf. on Genetic Algorithms; 1989.
[23] Angeline P, Saunders G, Pollack J. An evolutionary algorithm that constructs recurrent neural networks. IEEE Trans Neural Netw 1994;5(1):54–65.
[24] Stanley K, Miikkulainen R. Evolving neural networks through augmenting topologies. Evol Comput 2002;10:99–127.
[25] Hu Y., Wu X., He R.. TF-NAS: rethinking three search freedoms of latency-constrained differentiable neural architecture search. arXiv preprint arXiv:200805314 2020;.
[26] You S, Huang T, Yang M, Wang F, Qian C, Zhang C. GreedyNAS: towards fast one-shot NAS with greedy supernet. In: CVPR; 2020.
[27] Saikia T, Marrakchi Y, Zela A, Hutter F, Brox T. AutoDispNet: improving disparity estimation with automl. In: ICCV; 2019.
[28] Suganuma M, Ozay M, Okatani T. Exploiting the potential of standard convolutional autoencoders for image restoration by evolutionary search. In: ICML; 2018.
[29] Chen Y-C, Gao C, Robb E, Huang J-B. NAS-DIP: learning deep image prior with neural architecture search. In: ECCV; 2020.
[30] Kwatra V, Schodl A, Essa I, Turk G, Bobick A. Graphcut textures: Image and video synthesis using graph cuts. ACM Trans Graphics 2003;3(22):277–86.
[31] He K, Sun J. Statistics of patch offsets for image completion. In: Euro. Conf. on Comp. Vision (ECCV); 2012.
[32] Liu Y, Caselles V. Exemplar-based image inpainting using multiscale graph cuts. IEEE Trans Image Process 2013:1699–711.
[33] Efros A, Leung T. Texture Synthesis by non-parametric sampling. In: Proc. Intl. Conf. on Computer Vision (ICCV); 1999.
[34] Hays J, Efros AA. Scene completion using millions of photographs. In: ACM Transactions on Graphics (TOG). ACM; 2007.
[35] Gilbert A, Collomosse J, Jin H, Price B. Disentangling structure and aesthetics for content-aware image completion. In: Proc. CVPR; 2018.
[36] Barnes C, Shechtman E, Finkelstein A, Goldman D. PatchMatch: a randomized correspondence algorithm for structural image editing. In: Proc. ACM SIGGRAPH; 2009.
[37] Glasner D, Bagon S, Irani M. Super-resolution from a single image. In: Intl. conference on computer vision (ICCV); 2009.
[38] Zhang H, Mai L, KJin H, Wang Z, Xu N, Collomosse J. Video inpainting: an internal learning approach. In: Proc. ICCV; 2019.
[39] Shaham TR, Dekel T, Michaeli T. SinGAN: learning a generative model from a single natural image. In: Proc. ICCV; 2019.
[40] Gandelsman Y, Shocher A, Irani M. Double-dip: unsupervised image decomposition via coupled deep-image-priors. In: Proc. CVPR. IEEE; 2018.
[41] Zhang H, Xu T, Li H, Zhang S, Wang X, Huang S, et al. StackGAN++: realistic image synthesis with stacked generative adversarial networks. IEEE Trans PAMI 2017;41(8):1947–62.
[42] Wang T-C, Liu M-Y, Zhu J-Y, Tao A, Kautz J, Catanzaro B. High-resolution image synthesis and semantic manipulation with conditional GANs. In: Proc. CVPR 2018; 2018.
[43] Yeh R., Chen C., Lim T.Y., Hasegawa-Johnson M., Do M.N.. Semantic image inpainting with perceptual and contextual losses. arXiv preprint arXiv:160707539 2016;.
[44] Iizuka S, Simo-Serra E, Ishikawa H. Globally and locally consistent image completion. ACM Trans Graphics (Proc of SIGGRAPH 2017) 2017;36(4):107:1–107:14.
[45] Wilber M.J., Fang C., Jin H., Hertzmann A., Collomosse J., Belongie S.. Bam! the behance artistic media dataset for recognition beyond photography. arXiv preprint arXiv:170408614 2017a;.
[46] Salimans T, Goodfellow I, Zaremba W, Cheung V, Radford A, Chen X. Improved techniques for training GANs. In: Proc. NeurIPS; 2016. p. 2234–42.
[47] Heusel M, Ramsauer H, Unterthiner T, Nessler B, Hochreiter S. GANs trained by a two time-scale update rule converge to a local NASH equilibrium.. In: Proc. NeurIPS; 2017. p. 6629–40.
[48] Zhou B., Khosla A., Lapedriza A., Torralba A., Oliva A.. Places: an image database for deep scene understanding. arXiv preprint arXiv:161002055 2016;.
[49] Wilber M, Fang C, Jin H, Hertzmann A, Collomosse J, Belongie S. Bam! the behance artistic media dataset for recognition beyond photography. In: Proc. ICCV.
[50] Zeyde R, Elad M, Protter M. On single image scale-up using sparse-representations. In: International conference on curves and surfaces. Springer; 2010. p. 711–30.
[51] Wang Z, Bovik AC, Sheikh HR, Simoncelli EP. Image quality assessment: from error visibility to structural similarity. IEEE Trans Image Process 2004;13(4):600–12.
[52] Darabi S, Shechtman E, Barnes C, Goldman DB, Sen P. Image melding: combining inconsistent images using patch-based synthesis.. In: ACM TOG; 2012.
[53] Pathak D, Krähenbühl P, Donahue J, Darrell T, Efros A. Context encoders: Feature learning by inpainting. In: CVPR; 2016.