Rewriting a Deep Generative Model

David Bau[1], Steven Liu[1], Tongzhou Wang[1], Jun-Yan Zhu[2], Antonio Torralba[1]

MIT CSAIL[1] Adobe Research[2]

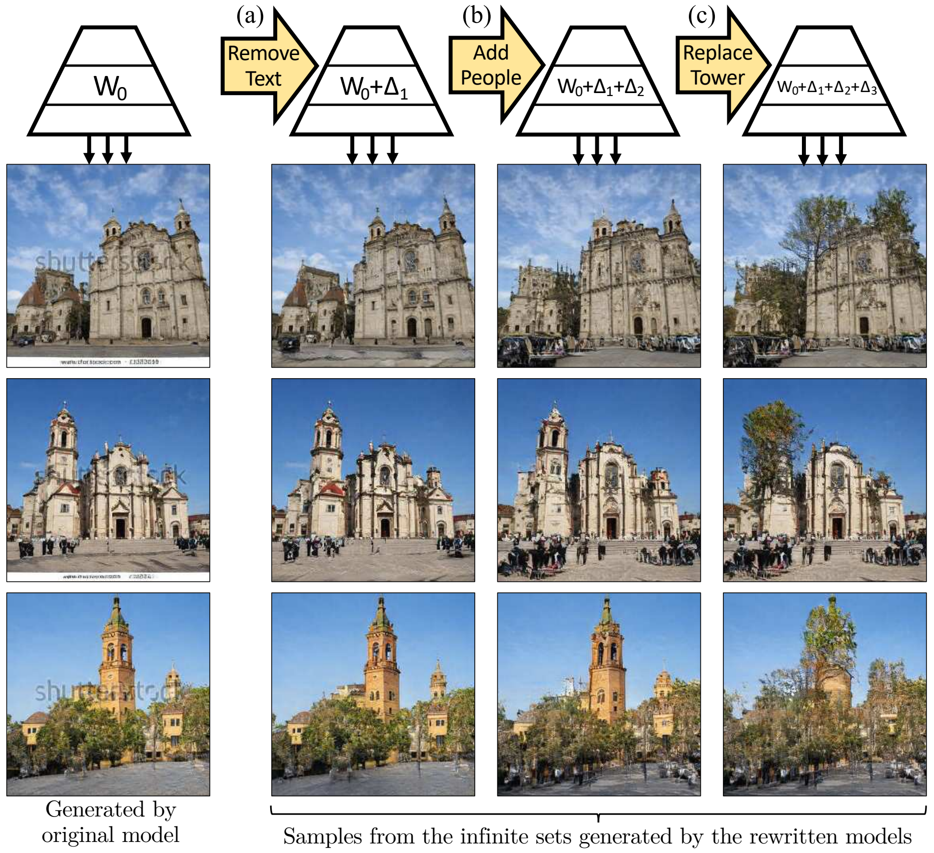
Abstract. A deep generative model such as a GAN learns to model a rich set of semantic and physical rules about the target distribution, but up to now, it has been obscure how such rules are encoded in the network, or how a rule could be changed. In this paper, we introduce a new problem setting: manipulation of specific rules encoded by a deep generative model. To address the problem, we propose a formulation in which the desired rule is changed by manipulating a layer of a deep network as a linear associative memory. We derive an algorithm for modifying one entry of the associative memory, and we demonstrate that several interesting structural rules can be located and modified within the layers of state-of-the-art generative models. We present a user interface to enable users to interactively change the rules of a generative model to achieve desired effects, and we show several proof-of-concept applications. Finally, results on multiple datasets demonstrate the advantage of our method against standard fine-tuning methods and edit transfer algorithms.

1 Introduction

We present the task of *model rewriting*, which aims to add, remove, and alter the semantic and physical rules of a pretrained deep network. While modern image editing tools achieve a user-specified goal by manipulating individual input images, we enable a user to synthesize an unbounded number of new images by editing a generative model to carry out modified rules.

For example in Figure 1, we apply a succession of rule changes to edit a StyleGANv2 model [40] pretrained on LSUN church scenes [77]. The first change removes watermark text patterns (a); the second adds crowds of people in front of buildings (b); the third replaces the rule for drawing tower tops with a rule that draws treetops (c), creating a fantastical effect of trees growing from towers. Because each of these modifications changes the generative model, every single change affects a whole category of images, removing all watermarks synthesized by the model, arranging people in front of many kinds of buildings, and creating tree-towers everywhere. The images shown are samples from an endless distribution.

But why is rewriting a deep generative model useful? A generative model enforces many rules and relationships within the generated images. From a purely scientific perspective, the ability to edit such a model provides insights about what the model has captured and how the model can generalize to unseen scenarios. At a practical level, deep generative models are increasingly useful for image and video synthesis [54,83,34,12]. In the future, entire image collections, videos,



(a)
Remove Text
(b)
Add People
(c)
Replace Tower
W_0
$W_0+\Delta_1$
$W_0+\Delta_1+\Delta_2$
$W_0+\Delta_1+\Delta_2+\Delta_3$
Generated by original model
Samples from the infinite sets generated by the rewritten models

Fig. 1: Rewriting the weights of a generator to change generative rules. Rules can be changed to (a) *remove* patterns such as watermarks; (b) *add* objects such as people; or (c) *replace* definitions such as making trees grow out of towers. Instead of editing individual images, our method edits the generator, so an infinite set of images can be potentially synthesized and manipulated using the altered rules.

or virtual worlds could potentially be produced by deep networks, and editing individual images or frames will be needlessly tedious. Instead, we would like to provide authoring tools for modifying the models themselves. With this capacity, a set of similar edits could be effortlessly *transferred* to many images at once.

A key question is how to edit a deep generative model. The computer vision community has become accustomed to training models using large data sets and expensive human annotations, but we wish to enable novice users to easily modify and customize a deep generative model *without* the training time, domain expertise, and computational cost of large-scale machine learning. In this paper, we present a new method that can locate and change a specific semantic relationship within a model. In particular, we show how to generalize the idea of a *linear associative memory* [45] to a nonlinear convolutional layer of a deep generator.

Each layer stores latent rules as a set of key-value relationships over hidden features. Our constrained optimization aims to add or edit one specific rule within the associative memory while preserving the existing semantic relationships in the model as much as possible. We achieve it by directly measuring and manipulating the model's internal structure, without requiring any new training data.

We use our method to create several visual editing effects, including the addition of new arrangements of objects in a scene, systematic removal of undesired output patterns, and global changes in the modeling of physical light. Our method is simple and fast, and it does not require a large set of annotations: a user can alter a learned rule by providing a single example of the new rule or a small handful of examples. We demonstrate a user interface for novice users to modify specific rules encoded in the layers of a GAN interactively. Finally, our quantitative experiments on several datasets demonstrate that our method outperforms several fine-tuning baselines as well as image-based edit transfer methods, regarding both photorealism and desirable effects. Our code, data, and user interface are available at our website.

2 Related Work

Deep image manipulation. Image manipulation is a classic problem in computer vision, image processing, and computer graphics. Common operations include color transfer [62,50], image deformation [64,72], object cloning [59,11], and patch-based image synthesis [19,5,28]. Recently, thanks to rapid advances of deep generative models [25,42,30], learning-based image synthesis and editing methods have become widely-used tools in the community, enabling applications such as manipulating the semantics of an input scene [57,6,69], image colorization [80,33,49,82], photo stylization [24,36,53,51], image-to-image translation [34,83,9,70,52,31], and face editing and synthesis [23,55,60]. While our user interface is inspired by previous interactive systems, our goal is *not* to manipulate and synthesize a single image using deep models. Instead, our work aims to manipulate the structural rules of the model itself, creating an altered deep network that can produce countless new images following the modified rules.

Edit transfer and propagation. Edit transfer methods propagate pixel edits to corresponding locations in other images of the same object or adjacent frames in the same video [2,74,27,14,13,78,20]. These methods achieve impressive results but are limited in two ways. First, they can only transfer edits to images of the same instance, as image alignment between different instances is challenging. Second, the edits are often restricted to color transfer or object cloning. In contrast, our method can change context-sensitive rules that go beyond pixel correspondences (Section 5.3). In Section 5.1, we compare to an edit propagation method based on state-of-the-art alignment algorithm, Neural Best-Buddies [1].

Interactive machine learning systems aim to improve training through human interaction in labeling [15,21,65], or by allowing a user to to aid in the model optimization process via interactive feature selection [18,26,61,47] or model and hyperparameter selection [58,35]. Our work differs from these previous approaches because rather than asking for human help to attain a fixed objective, we enable a user to solve novel creative modeling tasks, given a pre-trained model. Model rewriting allows a user to create a network with new rules that go beyond the patterns present in the training data.

Transfer learning and model fine-tuning. Transfer learning adapts a learned model to unseen learning tasks, domains, and settings. Examples include domain adaptation [63], zero-shot or few-shot learning [68,48], model pre-training and feature learning [16,79,75], and meta-learning [8,4,22]. Our work differs because instead of extending the training process with more data or annotations, we enable the user to directly change the behavior of the existing model through a visual interface. Recently, several methods [71,67,6] propose to train or fine-tune an image generation model to a particular image for editing and enhancement applications. Our goal is different, as we aim to identify and change rules that can generalize to many different images instead of one.

3 Method

To rewrite the rules of a trained generative model, we allow users to specify a handful of model outputs that they wish to behave differently. Based on this objective, we optimize an update in model weights that generalizes the requested change. In Section 3, we derive and discuss this optimization. In Section 4, we present the user interface that allows the user to interactively define the objective and edit the model.

Section 3.1 formulates our objective on how to add or modify a specific rule while preserving existing rules. We then consider this objective for linear systems and connect it to a classic technique—associative memory [43,3,44] (Section 3.2); this perspective allows us to derive a simple update rule (Section 3.3). Finally, we apply the solution to the nonlinear case and derive our full algorithm (Section 3.4).

3.1 Objective: Changing a Rule with Minimal Collateral Damage

Given a pre-trained generator $G(z;\theta_0)$ with weights θ_0, we can synthesize multiple images $x_i = G(z_i;\theta_0)$, where each image is produced by a latent code z_i. Suppose we have manually created desired changes x_{*i} for those cases. We would like to find updated weights θ_1 that change a computational rule to match our target examples $x_{*i} \approx G(z_i;\theta_1)$, while minimizing interference with other behavior:

$$\theta_1 = \arg\min_{\theta} \mathcal{L}_{\mathsf{smooth}}(\theta) + \lambda \mathcal{L}_{\mathsf{constraint}}(\theta), \tag{1}$$

$$\mathcal{L}_{\mathsf{smooth}}(\theta) \triangleq \mathbb{E}_z \left[\ell(G(z;\theta_0), G(z;\theta))\right], \tag{2}$$

$$\mathcal{L}_{\mathsf{constraint}}(\theta) \triangleq \sum_i \ell(x_{*i}, G(z_i;\theta)). \tag{3}$$

A traditional solution to the above problem is to jointly optimize the weighted sum of $\mathcal{L}_{\mathsf{smooth}}$ and $\mathcal{L}_{\mathsf{constraint}}$ over θ, where $\ell(\cdot)$ is a distance metric that measures the perceptual distance between images [36,17,81]. Unfortunately, this standard approach does not produce a generalized rule within G, because the large number of parameters θ allow the generator to quickly overfit the appearance of the new examples without good generalization; we evaluate this approach in Section 5.

However, the idea becomes effective with two modifications: (1) instead of modifying all of θ, we reduce the degrees of freedom by modifying weights W at only one layer, and (2) for the objective function, we directly minimize distance in the output feature space of that same layer.

Given a layer L, we use k to denote the features computed by the first $L-1$ fixed layers of G, and then write $v = f(k; W_0)$ to denote the computation of layer L itself, with pretrained weights W_0. For each exemplar latent z_i, these layers produce features k_{*i} and $v_{*i} = f(k_{*i}; W_0)$. Now suppose, for each target example x_{*i}, the user has manually created a feature change v_{*i}. (A user interface to create target feature goals is discussed in Section 4.) Our objective becomes:

$$W_1 = \arg\min_W \mathcal{L}_{\mathsf{smooth}}(W) + \lambda \mathcal{L}_{\mathsf{constraint}}(W), \tag{4}$$

$$\mathcal{L}_{\mathsf{smooth}}(W) \triangleq \mathbb{E}_k \left[\, ||f(k; W_0) - f(k; W)||^2 \, \right], \tag{5}$$

$$\mathcal{L}_{\mathsf{constraint}}(W) \triangleq \sum_i ||v_{*i} - f(k_{*i}; W)||^2, \tag{6}$$

where $||\cdot||^2$ denotes the L2 loss. Even within one layer, the weights W contain many parameters. But the degrees of freedom can be further reduced to constrain the change to a specific direction that we will derive; this additional directional constraint will allow us to create a generalized change from a single (k_*, v_*) example. To understand the constraint, it is helpful to interpret a single convolutional layer as an associative memory, a classic idea that we briefly review next.

3.2 Viewing a Convolutional Layer as an Associative Memory

Any matrix W can be used as an associative memory [44] that stores a set of key-value pairs $\{(k_i, v_i)\}$ that can be retrieved by matrix multiplication:

$$v_i \approx W k_i. \tag{7}$$

The use of a matrix as a *linear associative memory* is a foundational idea in neural networks [43,3,44]. For example, if the keys $\{k_i\}$ form a set of mutually orthogonal unit-norm vectors, then an error-free memory can be created as

$$W_{\text{orth}} \triangleq \sum_i v_i {k_i}^T. \tag{8}$$

Since ${k_i}^T k_j = 0$ whenever $i \neq j$, all the irrelevant terms cancel when multiplying by k_j, and we have $W_{\text{orth}}\, k_j = v_j$. A new value can be stored by adding $v_* {k_*}^T$ to the matrix as long as k_* is chosen to be orthogonal to all the previous keys. This process can be used to store up to N associations in an $M \times N$ matrix.

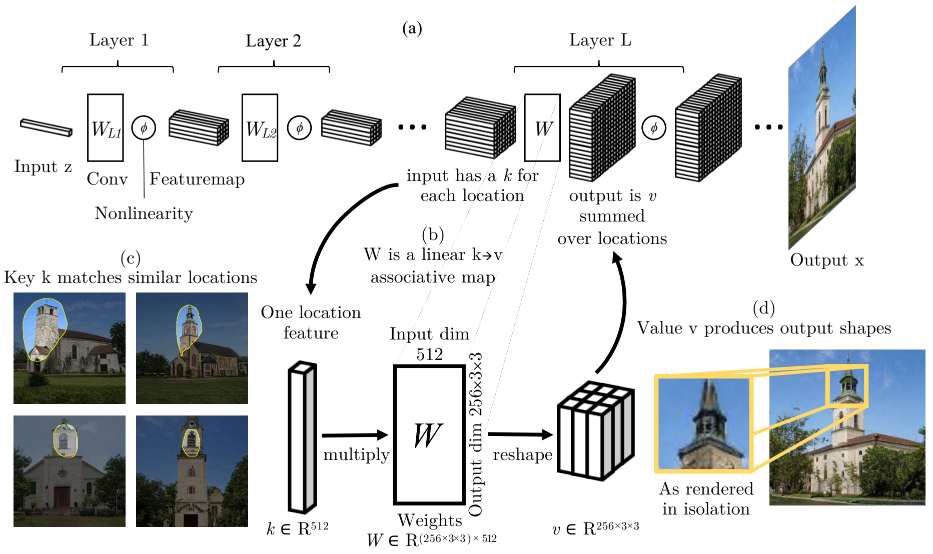

(a)
Layer 1
Layer 2
Layer L
W_{L1}
ϕ
W_{L2}
W
Input z
Conv
Featuremap
Nonlinearity
input has a k for each location
output is v summed over locations
Output x
(b)
W is a linear k→v associative map
(c)
Key k matches similar locations
One location feature
Input dim 512
Output dim 256x3x3
multiply
reshape
$k \in R^{512}$
Weights $W \in R^{(256\times3\times3)\times512}$
$v \in R^{256\times3\times3}$
(d)
Value v produces output shapes
As rendered in isolation

Fig. 2: (a) A generator consists of a sequence of layers; we focus on one particular layer L. (b) The convolutional weights W serve an associative memory, mapping keys k to values v. The keys are single-location input features, and the values are patterns of output features. (c) A key will tend to match semantically similar contexts in different images. Shown are locations of generated images that have features that match a specific k closely. (d) A value renders shapes in a small region. Here the effect of a value v is visualized by rendering features at one location alone, with features at other locations set to zero. Image examples are taken from a StyleGANv2 model trained on LSUN outdoor church scenes.

Figure 2 views the weights of one convolutional layer in a generator as an associative memory. Instead of thinking of the layer as a collection of convolutional filtering operations, we can think of the layer as a memory that associates keys to values. Here each key k is a single-location feature vector. The key is useful because, in our trained generator, the same key will match many semantically similar locations across different images, as shown in Figure 2c. Associated with each key, the map stores an output value v that will render an arrangement of output shapes. This output can be visualized directly by rendering the features in isolation from neighboring locations, as shown in Figure 2d.

For example, consider a layer that transforms a 512-channel featuremap into a 256-channel featuremap using a 3×3 convolutional kernel; the weights form a $256 \times 512 \times 3 \times 3$ tensor. For each key $k \in \mathbb{R}^{512}$, our layer will recall a value $v \in \mathbb{R}^{256\times3\times3} = \mathbb{R}^{2304}$ representing a 3×3 output pattern of 256-channel features, flattened to a vector, as $v = Wk$. Our interpretation of the layer as an associative memory does not change the computation: the tensor is simply reshaped and treated as a dense rectangular matrix $W \in \mathbb{R}^{(256\times3\times3)\times512}$, whose job is to map keys $k \in \mathbb{R}^{512}$ to values $v \in \mathbb{R}^{2304}$, via Eqn. 7.

Arbitrary Nonorthogonal Keys. In classic work, Kohonen [45] observed that an associative memory can support more than N nonorthogonal keys $\{k_i\}$ if instead of requiring exact equality $v_i = Wk_i$, we choose W_0 to minimize error:

$$W_0 \triangleq \arg\min_W \sum_i ||v_i - Wk_i||^2. \tag{9}$$

To simplify notation, let us assume a finite set of pairs $\{(k_i, v_i)\}$ and collect keys and values into matrices K and V whose i-th column is the i-th key or value:

$$K \triangleq [k_1|k_2|\cdots|k_i|\cdots], \tag{10}$$

$$V \triangleq [v_1|v_2|\cdots|v_i|\cdots]. \tag{11}$$

The minimization (Eqn. 9) is the standard linear least-squares problem. A unique minimal solution can be found by solving for W_0 using the normal equation $W_0KK^T = VK^T$, or equivalently by using the pseudoinverse $W_0 = VK^+$.

3.3 Updating W to Insert a New Value

Now, departing from Kohonen [45], we ask how to modify W_0. Suppose we wish to overwrite a single key to assign a new value $k_* \to v_*$ provided by the user. After this modification, our new matrix W_1 should satisfy two conditions:

$$W_1 = \arg\min_W ||V - WK||^2, \tag{12}$$

$$\text{subject to } v_* = W_1 k_*. \tag{13}$$

That is, it should store the new value; and it should continue to minimize error in all the previously stored values. This forms a constrained linear least-squares (CLS) problem which can be solved exactly as $W_1KK^T = VK^T + \varLambda {k_*}^T$, where the vector $\varLambda \in \mathbb{R}^m$ is determined by solving the linear system with the constraint in Eqn. 13 (see Appendix B). Because W_0 satisfies the normal equations, we can expand VK^T in the CLS solution and simplify:

$$W_1KK^T = W_0KK^T + \varLambda {k_*}^T \tag{14}$$

$$W_1 = W_0 + \varLambda(C^{-1}k_*)^T \tag{15}$$

Above, we have written $C \triangleq KK^T$ as the second moment statistics. (C is symmetric; if K has zero mean, C is the covariance.) Now Eqn. 15 has a simple form. Since $\varLambda \in \mathbb{R}^m$ and $(C^{-1}k_*)^T \in \mathbb{R}^n$ are simple vectors, the update $\varLambda(C^{-1}k_*)^T$ is a rank-one matrix with rows all multiples of the vector $(C^{-1}k_*)^T$.

Eqn. 15 is interesting for two reasons. First, it shows that enforcing the user's requested mapping $k_* \to v_*$ transforms the soft error minimization objective (12) into the hard constraint that the weights be updated in a particular straight-line direction $C^{-1}k_*$. Second, it reveals that the update direction is determined only by the overall key statistics and the specific targeted key k_*. The covariance C is a model constant that can be pre-computed and cached, and the update direction is determined by the key *regardless of any stored value.* Only $\varLambda$, which specifies the magnitude of each row change, depends on the target value v_*.

3.4 Generalize to a Nonlinear Neural Layer

In practice, even a single network block contains several non-linear components such as a biases, ReLU, normalization, and style modulation. Below, we generalize our procedure to the nonlinear case where the solution to W_1 cannot be calculated in a closed form. We first define our update direction:

$$d \triangleq C^{-1} k_*. \tag{16}$$

Then suppose we have a non-linear neural layer $f(k; W)$ which follows the linear operation W with additional nonlinear steps. Since the form of Eqn. 15 is sensitive to the rowspace of W and insensitive to the column space, we can use the same rank-one update form to constrain the optimization of $f(k_*; W) \approx v_*$.

Therefore, in our experiments, when we update a layer to insert a new key $k_* \rightarrow v_*$, we begin with the existing W_0, and we perform an optimization over the rank-one subspace defined by the row vector d^T from Eqn. 16. That is, in the nonlinear case, we update W_1 by solving the following optimization:

$$\Lambda_1 = \underset{\Lambda \in \mathbb{R}^M}{\arg\min} \, ||v_* - f(k_*; W_0 + \Lambda \, d^T)||. \tag{17}$$

Once Λ_1 is computed, we update the weight as $W_1 = W_0 + \Lambda_1 d^T$.

Our desired insertion may correspond to a change of more than one key at once, particularly if our desired target output forms a feature map patch V_* larger than a single convolutional kernel, i.e., if we wish to have $V_* = f(K_*; W_1)$ where K_* and V_* cover many pixels. To alter S keys at once, we can define the allowable deltas as lying within the low-rank space spanned by the $N \times S$ matrix D_S containing multiple update directions $d_i = C^{-1} K_{*i}$, indicating which entries of the associative map we wish to change.

$$\Lambda_S = \underset{\Lambda \in \mathbb{R}^{M \times S}}{\arg\min} \, ||V_* - f(K_*; W_0 + \Lambda \, {D_S}^T)||, \tag{18}$$

$$\text{where } D_S \triangleq [d_1 | d_2 | \cdots | d_i | \cdots | d_S] \,. \tag{19}$$

We can then update the layer weights using $W_S = W_0 + \Lambda_S {D_S}^T$. The change can be made more specific by reducing the rank of D_S; details are discussed Appendix D. To directly connect this solution to our original objective (Eqn. 6), we note that the constrained optimization can be solved using projected gradient descent. That is, we relax Eqn. 18 and use optimization to minimize $\arg\min_W ||V_* - f(K_*; W)||$; then, to impose the constraint, after each optimization step, project W into into the subspace $W_0 + \Lambda_S {D_S}^T$.

4 User Interface

To make model rewriting intuitive for a novice user, we build a user interface that provides a three-step rewriting process: *Copy*, *Paste*, and *Context*.

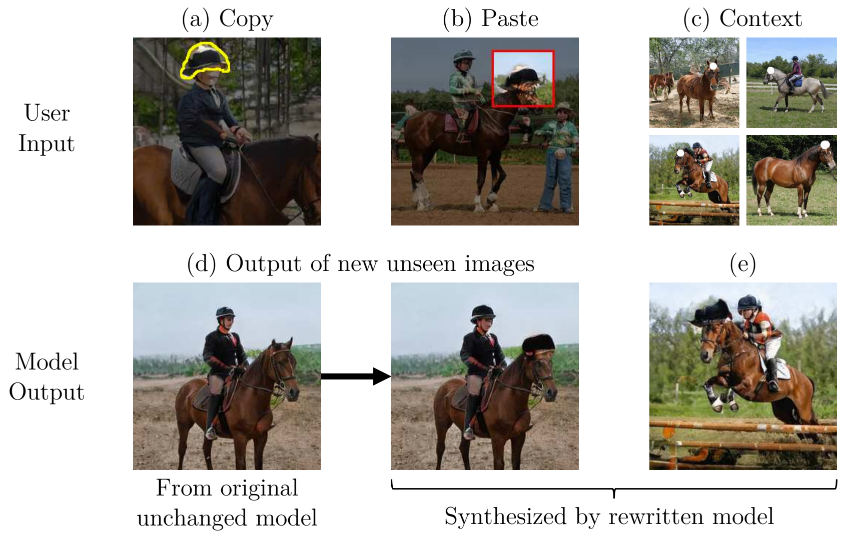

(a) Copy
(b) Paste
(c) Context
User Input
(d) Output of new unseen images
(e)
Model Output
From original unchanged model
Synthesized by rewritten model

Fig. 3: The *Copy-Paste-Context* interface for rewriting a model. (a) **Copy:** the user uses a brush to select a region containing an interesting object or shape, defining the target value V_*. (b) **Paste:** The user positions and pastes the copied object into a single target image. This specifies the $K_* \to V_*$ pair constraint. (c) **Context:** To control generalization, the user selects target regions in several images. This establishes the updated direction d for the associative memory. (d) The edit is applied to the model, not a specific image, so newly generated images will always have hats on top of horse heads. (e) The change has generalized to a variety of different types of horses and poses (see more in Appendix A).

Copy and Paste allow the user to copy an object from one generated image to another. The user browses through a collection of generated images and highlights an area of interest to copy; then selects a generated target image and location for pasting the object. For example, in Figure 3a, the user selects a helmet worn by a rider and then pastes it in Figure 3b on a horse's head.

Our method downsamples the user's copied region to the resolution of layer L and gathers the copied features as the target value V_*. Because we wish to change not just one image, but the model rules themselves, we treat the pasted image as a new rule $K_* \to V_*$ associating the layer $L-1$ features K_* of the target image with the newly copied layer L values V_* that will govern the new appearance.

Context Selection allows a user to specify how this change will be generalized, by pointing out a handful of similar regions that should be changed. For example, in Figure 3b, the user has selected heads of different horses.

We collect the layer $L-1$ features at the location of the context selections as a set of relevant K that are used to determine the weight update direction d via Eqn. 16. Generalization improves when we allow the user to select several context regions to specify the update direction (see Table 1); in Figure 3, the four examples are used to create a single d. Appendix D discusses this rank reduction.

Applying one rule change on a StyleGANv2 model requires about eight seconds on a single Titan GTX GPU. Please check out the demo video of our interface.

5 Results

We test model rewriting with three editing effects. First, we add new objects into the model, comparing results to several baseline methods. Then, we use our technique to erase objects using a low-rank change; we test this method on the challenging watermark removal task. Finally, we invert a rule for a physical relationship between bright windows and reflections in a model.

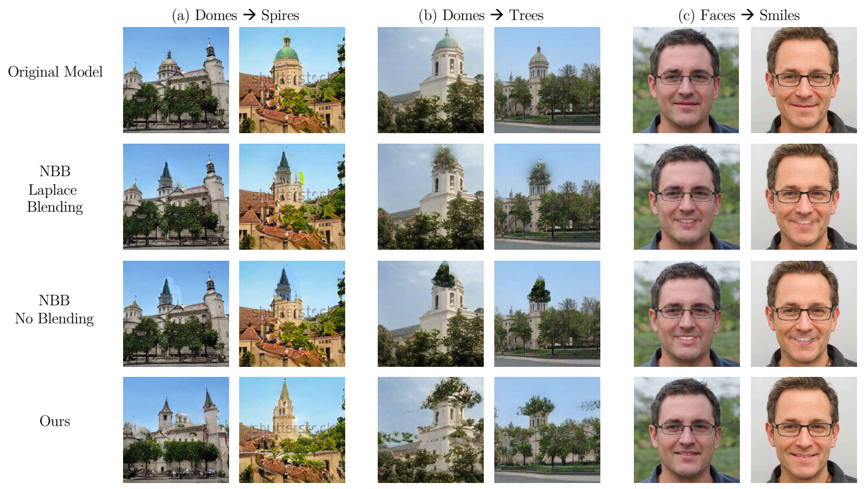


Fig. 4: Adding and replacing objects in three different settings. (a) Replacing domes with an angular peaked spire causes peaked spires to be used throughout the model. (b) Replacing domes with trees can generate images unlike any seen in a training set. (c) Replacing closed lips with an open-mouth smile produces realistic open-mouth smiles. For each case, we show the images generated by an unchanged model, then the edit propagation results, with and without blending. Our method is shown in the last row.

5.1 Putting objects into a new context

Here we test our method on several specific model modifications. In a church generator, the model edits change the shape of domes to spires, and change the domes to trees, and in a face generator, we add open-mouth smiles. Examples of all the edits are shown in Figure 4.

Quantitative Evaluation. In Tables 1 and 2,we compare the results to several baselines. We compare our method to the naive approach of fine-tuning all weights

	% smiling images ↑	LPIPS (masked) ↓	% more realistic than ours ↑
Our method (projected gradient descent)	84.37	**0.04**	–
With direct optimization of Λ	87.44	0.14	43.0
With single-image direction constraint	82.12	0.05	47.3
With single-layer, no direction constraint	90.94	0.30	6.8
Finetuning all weights	85.78	0.40	8.7
NBB + Direct copying	94.81	0.32	9.8
NBB + Laplace blending	**93.51**	0.32	8.6
Unmodified model	78.37	–	50.9

Table 1: Editing a StyleGANv2 [40] FFHQ [39] model to produce smiling faces in $n = 10,000$ images. To quantify the efficacy of the change, we show the percentage of smiling faces among the modified images, and we report the LPIPS distance on masked images to quantify undesired changes. For realism, workers make $n = 1,000$ pairwise judgements comparing images from other methods to ours.

	Dome → Spire			Dome → Tree	
	% `dome` pixels correctly modified ↑	LPIPS (masked) ↓	% more realistic than ours ↑	% `dome` pixels correctly modified ↑	LPIPS (masked) ↓
Our method (projected gradient descent)	**92.03**	**0.02**	–	48.65	**0.03**
With direct optimization of Λ	80.03	0.10	53.7	**59.43**	0.13
With single-image direction constraint	90.14	0.04	48.8	39.72	0.03
With single-layer, no direction constraint	80.69	0.29	38.1	41.32	0.45
Finetuning all weights	41.16	0.36	27.1	10.16	0.31
NBB + Direct copying	69.99	0.08	8.9	46.44	0.09
NBB + Laplace blending	69.63	0.08	12.2	31.18	0.09
Unmodified model	–	–	63.8	–	–

Table 2: We edit a StyleGANv2 [40] LSUN church [77] model to replace domes with spires/trees in $n = 10,000$ images. To quantify efficacy, we show the percentage of `dome` category pixels changed to the target category, determined by a segmenter [73]. To quantify undesired changes, we report LPIPS distance between edited and unchanged images, in non-dome regions. For realism, workers make $n = 1,000$ pairwise judgements comparing images from other methods to ours.

according to Eqn. 3, as well as the method of optimizing all the weights of a layer without constraining the direction of the change, as in Eqn. 6, and to a state-of-the-art image alignment algorithm, *Neural Best-Buddies* (NBB [1]), which is used to propagate an edit across a set of similar images by compositing pixels according to identified sparse correspondences. To transfer an edit from a target image, we use NBB and Moving Least Squares [64] to compute a dense correspondence between the source image we would like to edit and the original target image. We use this dense correspondence field to warp the masked target into the source image. We test both direct copying and Laplace blending.

For each setting, we measure the efficacy of the edits on a sample of 10, 000 generated images, and we also quantify the undesired changes made by each method. For the smiling edit, we measure efficacy by counting images classified as smiling by an attribute classifier [66], and we also quantify changes made in the images outside the mouth region by masking lips using a face segmentation model [84] and using LPIPS [81] to quantify changes. For the dome edits, we

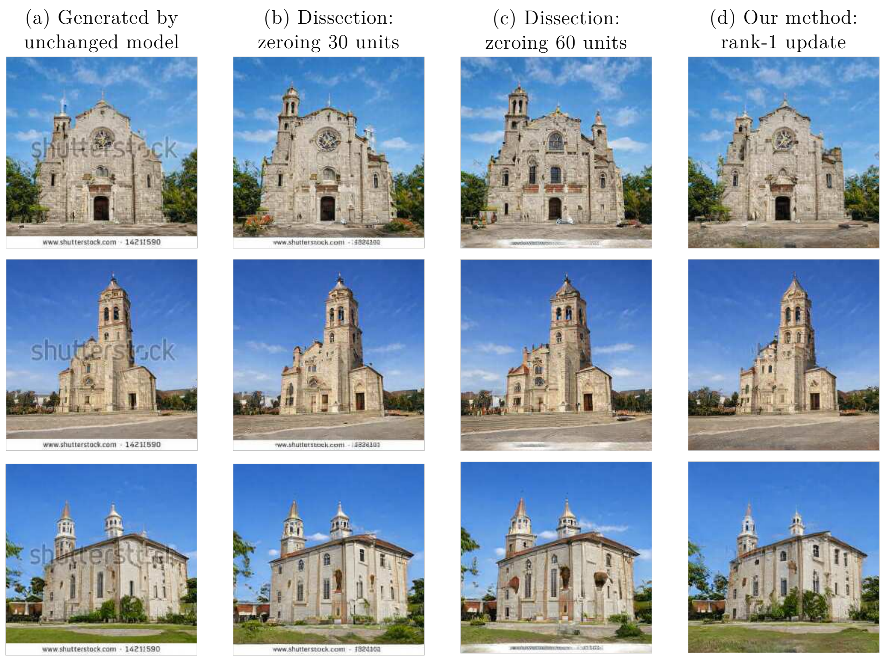

(a) Generated by unchanged model
(b) Dissection: zeroing 30 units
(c) Dissection: zeroing 60 units
(d) Our method: rank-1 update

Fig. 5: Removing watermarks from StyleGANv2 [40] LSUN church [77] model. (a) Many images generated by this model include transparent watermarks in the center or text on the bottom. (b) Using GAN Dissection [7] to zero 30 text-specific units removes middle but not bottom text cleanly. (c) Removing 60 units does not fully remove text, and distorts other aspects of the image. (b) Applying our method to create a rank-1 change erases both middle and bottom text cleanly.

Count of visible watermarks	middle	bottom
Zeroing 30 units (GAN Dissection)	**0**	6
Zeroing 60 units (GAN Dissection)	**0**	4
Rank-1 update (our method)	**0**	**0**
Unmodified model	64	26

Table 3: Visible watermark text produced by StyleGANv2 church model in $n = 1000$ images, without modification, with sets of units zeroed (using the method of GAN Dissection), and using our method to apply a rank-one update.

measure how many dome pixels are judged to be changed to non-domes by a segmentation model [73], and we measure undesired changes outside dome areas using LPIPS. We also conduct a user study where users are asked to compare the realism of our edited output to the same image edited using baseline methods. We find that our method produces more realistic outputs that are more narrowly targeted than the baseline methods. For the smile edit, our method is not as aggressive as baseline methods at introducing smiles, but for the dome edits, our method is more effective than baseline methods at executing the change. Our metrics are further discussed in Appendix C.

5.2 Removing undesired features

Here we test our method on the removal of undesired features. Figure 5a shows several examples of images output by a pre-trained StyleGANv2 church model. This model occasionally synthesizes images with text overlaid in the middle and the bottom resembling stock-photo watermarks in the training set.

The GAN Dissection study [7] has shown that some objects can be removed from a generator by zeroing the units that best match those objects. To find these units, we annotated the middle and bottom text regions in ten generated images, and we identified a set of 60 units that are most highly correlated with features in these regions. Zeroing the most correlated 30 units removes some of the text, but leaves much bottom text unremoved, as shown in Figure 5b. Zeroing all 60 units reduces more of the bottom text but begins to alter the main content of the images, as shown in Figure 5c.

For our method, we use the ten user-annotated images as a context to create a rank-one constraint direction d for updating the model, and as an optimization target $K_* \to V_*$, we use one successfully removed watermark from the setting shown in Figure 5b. Since our method applies a narrow rank-1 change constraint, it would be expected to produce a loose approximation of the rank-30 change in the training example. Yet we find that it has instead improved specificity and generalization of watermark removal, removing both middle and bottom text cleanly while introducing few changes in the main content of the image. We repeat the process for 1000 images and tabulate the results in Table 3.

5.3 Changing contextual rules

In this experiment, we find and alter a rule that determines the illumination interactions between two objects at different locations in an image.

State-of-the-art generative models learn to enforce many relationships between distant objects. For example, it has been observed [6] that a kitchen-scene Progressive GAN model [38] enforces a relationship between windows on walls and specular reflections on tables. When windows are added to a wall, reflections will be added to shiny tabletops, and vice-versa, as illustrated in the first row of Figure 6. Thus the model contains a rule that approximates the physical propagation of light in a scene.

In the following experiment, we identified an update direction that allows us to change this model of light reflections. Instead of specifying an objective that copies an object from one context to another, we used a similar tool to specify a $K_* \to V_*$ objective that swaps bright tabletop reflections with dim reflections on a set of 15 pairs of scenes that are identical other than the presence or absence of bright windows. To identify a rank-one change direction d, we used projected gradient descent, as described in Section 3.4, using SVD to limit the change to rank one during optimization. The results are shown in the second row of Figure 6. The modified model differs from the original only in a single update direction of a single layer, but it inverts the relationship between windows and reflections: when windows are added, reflections are reduced, and vice-versa.

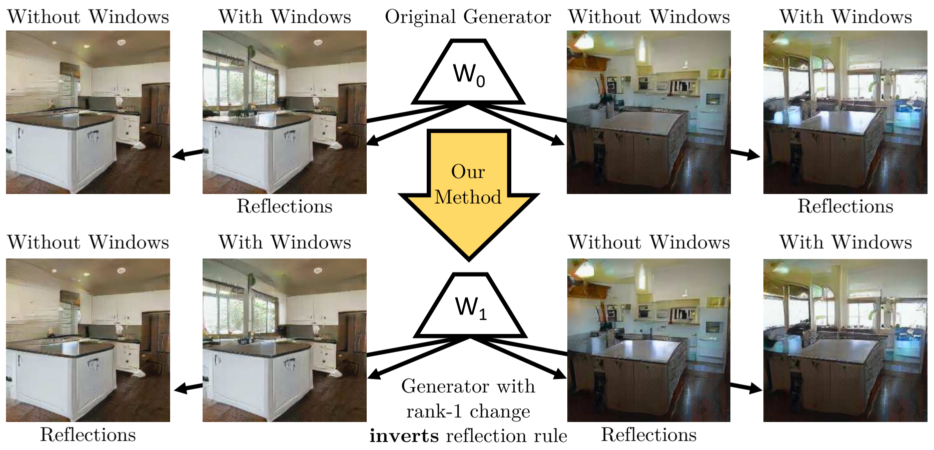


Fig. 6: Inverting a single semantic rule within a model. At the top row, a Progressive GAN [38] trained on LSUN kitchens [77] links windows to reflections: when windows are added by manipulating intermediate features identified by GAN Dissection [7], reflections appear on the table. In the bottom row, one rule has been changed within the model to *invert* the relationship between windows and reflections. Now adding windows *decreases* reflections and vice-versa.

6 Discussion

Machine learning requires data, so how can we create effective models for data that do not yet exist? Thanks to the rich internal structure of recent GANs, in this paper, we have found it feasible to create such models by rewriting the rules within existing networks. Although we may never have seen a tree sprouting from a tower, our network contains rules for both trees and towers, and we can easily create a model that connects those compositional rules to synthesize an endless distribution of images containing the new combination.

The development of sophisticated generative models beyond the image domain, such as the GPT-3 language model [10] and WaveNet for audio synthesis [56], means that it will be increasingly attractive to rewrite rules within other types of models as well. After training on vast datasets, large-scale deep networks have proven to be capable of representing an extensive range of different styles, sentiments, and topics. Model rewriting provides an avenue for using this structure as a rich medium for creating novel kinds of content, behavior, and interaction.

Acknowledgements. We thank Jonas Wulff, Hendrik Strobelt, Aaron Hertzman, Taesung Park, William Peebles, Gerald Sussman, and William T. Freeman for their vision, encouragement, and many valuable discussions. We are grateful for the support of DARPA XAI FA8750-18-C-0004, DARPA SAIL-ON HR0011-20-C-0022, NSF 1524817 on Advancing Visual Recognition with Feature Visualizations, NSF BIGDATA 1447476, and a hardware donation from NVIDIA.

References

1. Aberman, K., Liao, J., Shi, M., Lischinski, D., Chen, B., Cohen-Or, D.: Neural best-buddies: Sparse cross-domain correspondence. ACM TOG **37**(4), 69 (2018) 3, 11
2. An, X., Pellacini, F.: Appprop: all-pairs appearance-space edit propagation. ACM TOG **27**(3), 40 (2008) 3
3. Anderson, J.A.: A simple neural network generating an interactive memory. Mathematical biosciences **14**(3-4), 197–220 (1972) 4, 5
4. Andrychowicz, M., Denil, M., Gomez, S., Hoffman, M.W., Pfau, D., Schaul, T., Shillingford, B., De Freitas, N.: Learning to learn by gradient descent by gradient descent. In: NeurIPS. pp. 3981–3989 (2016) 4
5. Barnes, C., Shechtman, E., Finkelstein, A., Goldman, D.B.: Patchmatch: A randomized correspondence algorithm for structural image editing. ACM TOG **28**(3), 24 (2009) 3
6. Bau, D., Strobelt, H., Peebles, W., Wulff, J., Zhou, B., Zhu, J., Torralba, A.: Semantic photo manipulation with a generative image prior. ACM TOG **38**(4) (2019) 3, 4, 13
7. Bau, D., Zhu, J.Y., Strobelt, H., Bolei, Z., Tenenbaum, J.B., Freeman, W.T., Torralba, A.: Gan dissection: Visualizing and understanding generative adversarial networks. In: ICLR (2019) 12, 13, 14, 27
8. Bengio, S., Bengio, Y., Cloutier, J., Gecsei, J.: On the optimization of a synaptic learning rule. In: Optimality in Artificial and Biological Neural Networks. pp. 6–8. Univ. of Texas (1992) 4
9. Bousmalis, K., Silberman, N., Dohan, D., Erhan, D., Krishnan, D.: Unsupervised pixel-level domain adaptation with generative adversarial networks. In: CVPR (2017) 3
10. Brown, T.B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al.: Language models are few-shot learners. arXiv preprint arXiv:2005.14165 (2020) 14
11. Burt, P., Adelson, E.: The laplacian pyramid as a compact image code. IEEE Transactions on communications **31**(4), 532–540 (1983) 3
12. Chan, C., Ginosar, S., Zhou, T., Efros, A.A.: Everybody dance now. In: ICCV (2019) 1
13. Chen, X., Zou, D., Li, J., Cao, X., Zhao, Q., Zhang, H.: Sparse dictionary learning for edit propagation of high-resolution images. In: CVPR (2014) 3
14. Chen, X., Zou, D., Zhao, Q., Tan, P.: Manifold preserving edit propagation. ACM TOG **31**(6), 1–7 (2012) 3
15. Cohn, D., Atlas, L., Ladner, R.: Improving generalization with active learning. Machine learning **15**(2), 201–221 (1994) 3
16. Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., Darrell, T.: Decaf: A deep convolutional activation feature for generic visual recognition. In: ICML (2014) 4
17. Dosovitskiy, A., Brox, T.: Generating images with perceptual similarity metrics based on deep networks. In: NeurIPS (2016) 5
18. Dy, J.G., Brodley, C.E.: Visualization and interactive feature selection for unsupervised data. In: SIGKDD. pp. 360–364 (2000) 3
19. Efros, A.A., Freeman, W.T.: Image quilting for texture synthesis and transfer. In: SIGGRAPH. ACM (2001) 3

20. Endo, Y., Iizuka, S., Kanamori, Y., Mitani, J.: Deepprop: Extracting deep features from a single image for edit propagation. Computer Graphics Forum **35**(2), 189–201 (2016) 3
21. Fails, J.A., Olsen Jr, D.R.: Interactive machine learning. In: ACM IUI. pp. 39–45 (2003) 3
22. Finn, C., Abbeel, P., Levine, S.: Model-agnostic meta-learning for fast adaptation of deep networks. In: ICML. pp. 1126–1135. JMLR. org (2017) 4
23. Fried, O., Tewari, A., Zollhöfer, M., Finkelstein, A., Shechtman, E., Goldman, D.B., Genova, K., Jin, Z., Theobalt, C., Agrawala, M.: Text-based editing of talking-head video. ACM TOG **38**(4), 1–14 (2019) 3
24. Gatys, L.A., Ecker, A.S., Bethge, M.: Image style transfer using convolutional neural networks. In: CVPR (2016) 3
25. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: NeurIPS (2014) 3
26. Guo, D.: Coordinating computational and visual approaches for interactive feature selection and multivariate clustering. Information Visualization **2**(4), 232–246 (2003) 3
27. Hasinoff, S.W., Jóźwiak, M., Durand, F., Freeman, W.T.: Search-and-replace editing for personal photo collections. In: 2010 IEEE International Conference on Computational Photography (ICCP). pp. 1–8. IEEE (2010) 3
28. Hertzmann, A., Jacobs, C.E., Oliver, N., Curless, B., Salesin, D.H.: Image analogies. In: SIGGRAPH (2001) 3
29. 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: NeurIPS (2017) 28
30. Hinton, G.E., Osindero, S., Teh, Y.W.: A fast learning algorithm for deep belief nets. Neural Computation **18**, 1527–1554 (2006) 3
31. Huang, X., Liu, M.Y., Belongie, S., Kautz, J.: Multimodal unsupervised image-to-image translation. ECCV (2018) 3
32. Huh, M., Zhang, R., Zhu, J.Y., Paris, S., Hertzmann, A.: Transforming and projecting images to class-conditional generative networks. In: ECCV (2020) 22
33. Iizuka, S., Simo-Serra, E., Ishikawa, H.: Let there be Color!: Joint End-to-end Learning of Global and Local Image Priors for Automatic Image Colorization with Simultaneous Classification. ACM TOG **35**(4) (2016) 3
34. Isola, P., Zhu, J.Y., Zhou, T., Efros, A.A.: Image-to-image translation with conditional adversarial networks. In: CVPR (2017) 1, 3
35. Jiang, B., Canny, J.: Interactive machine learning via a gpu-accelerated toolkit. In: ACM IUI. pp. 535–546 (2017) 4
36. Johnson, J., Alahi, A., Fei-Fei, L.: Perceptual losses for real-time style transfer and super-resolution. In: ECCV (2016) 3, 5
37. Karras, T.: FFHQ dataset. `https://github.com/NVlabs/ffhq-dataset` (2019) 20
38. Karras, T., Aila, T., Laine, S., Lehtinen, J.: Progressive growing of gans for improved quality, stability, and variation. In: ICLR (2018) 13, 14
39. Karras, T., Laine, S., Aila, T.: A style-based generator architecture for generative adversarial networks. In: CVPR (2019) 11, 27
40. Karras, T., Laine, S., Aittala, M., Hellsten, J., Lehtinen, J., Aila, T.: Analyzing and improving the image quality of stylegan. In: CVPR (2020) 1, 11, 12, 21, 26
41. Kingma, D., Ba, J.: Adam: A method for stochastic optimization. In: ICLR (2015) 26, 27
42. Kingma, D.P., Welling, M.: Auto-encoding variational bayes. ICLR (2014) 3

43. Kohonen, T.: Correlation matrix memories. IEEE transactions on computers **100**(4), 353–359 (1972) 4, 5
44. Kohonen, T.: Associative memory: A system-theoretical approach, vol. 17. Springer Science & Business Media (2012) 4, 5
45. Kohonen, T., Ruohonen, M.: Representation of associated data by matrix operators. IEEE Transactions on Computers **100**(7), 701–702 (1973) 2, 7
46. Kokiopoulou, E., Chen, J., Saad, Y.: Trace optimization and eigenproblems in dimension reduction methods. Numerical Linear Algebra with Applications **18**(3), 565–602 (2011) 26
47. Krause, J., Perer, A., Bertini, E.: Infuse: interactive feature selection for predictive modeling of high dimensional data. IEEE transactions on visualization and computer graphics **20**(12), 1614–1623 (2014) 3
48. Lake, B.M., Salakhutdinov, R., Tenenbaum, J.B.: Human-level concept learning through probabilistic program induction. Science **350**(6266), 1332–1338 (2015) 4
49. Larsson, G., Maire, M., Shakhnarovich, G.: Learning representations for automatic colorization. In: ECCV (2016) 3
50. Levin, A., Lischinski, D., Weiss, Y.: Colorization using optimization. ACM TOG **23**(3), 689–694 (2004) 3
51. Liao, J., Yao, Y., Yuan, L., Hua, G., Kang, S.B.: Visual attribute transfer through deep image analogy. arXiv preprint arXiv:1705.01088 (2017) 3
52. Liu, M.Y., Breuel, T., Kautz, J.: Unsupervised image-to-image translation networks. In: NeurIPS (2017) 3
53. Luan, F., Paris, S., Shechtman, E., Bala, K.: Deep photo style transfer. In: CVPR. pp. 4990–4998 (2017) 3
54. Mathieu, M., Couprie, C., LeCun, Y.: Deep multi-scale video prediction beyond mean square error. In: ICLR (2016) 1
55. Nagano, K., Seo, J., Xing, J., Wei, L., Li, Z., Saito, S., Agarwal, A., Fursund, J., Li, H., Roberts, R., et al.: pagan: real-time avatars using dynamic textures. In: SIGGRAPH Asia. p. 258 (2018) 3
56. Oord, A.v.d., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., Kavukcuoglu, K.: Wavenet: A generative model for raw audio. arXiv preprint arXiv:1609.03499 (2016) 14
57. Park, T., Liu, M.Y., Wang, T.C., Zhu, J.Y.: Semantic image synthesis with spatially-adaptive normalization. In: CVPR (2019) 3
58. Patel, K., Drucker, S.M., Fogarty, J., Kapoor, A., Tan, D.S.: Using multiple models to understand data. In: IJCAI (2011) 4
59. Pérez, P., Gangnet, M., Blake, A.: Poisson image editing. In: SIGGRAPH. pp. 313–318 (2003) 3
60. Portenier, T., Hu, Q., Szabó, A., Bigdeli, S.A., Favaro, P., Zwicker, M.: Faceshop: Deep sketch-based face image editing. ACM Transactions on Graphics (TOG) **37**(4), 99:1–99:13 (Jul 2018) 3
61. Raghavan, H., Madani, O., Jones, R.: Active learning with feedback on features and instances. JMLR **7**(Aug), 1655–1686 (2006) 3
62. Reinhard, E., Adhikhmin, M., Gooch, B., Shirley, P.: Color transfer between images. IEEE Computer graphics and applications **21**(5), 34–41 (2001) 3
63. Saenko, K., Kulis, B., Fritz, M., Darrell, T.: Adapting visual category models to new domains. In: ECCV. pp. 213–226. Springer (2010) 4
64. Schaefer, S., McPhail, T., Warren, J.: Image deformation using moving least squares. ACM TOG **25**(3), 533540 (Jul 2006) 3, 11
65. Settles, B., Craven, M.: An analysis of active learning strategies for sequence labeling tasks. In: EMNLP. pp. 1070–1079 (2008) 3

66. Sharma, A., Foroosh, H.: Slim-cnn: A light-weight cnn for face attribute prediction. arXiv preprint arXiv:1907.02157 (2019) 11, 22
67. Shocher, A., Cohen, N., Irani, M.: zero-shot super-resolution using deep internal learning. In: CVPR. pp. 3118–3126 (2018) 4
68. Socher, R., Ganjoo, M., Manning, C.D., Ng, A.: Zero-shot learning through cross-modal transfer. In: NeurIPS (2013) 4
69. Suzuki, R., Koyama, M., Miyato, T., Yonetsuji, T., Zhu, H.: Spatially controllable image synthesis with internal representation collaging. arXiv preprint arXiv:1811.10153 (2018) 3
70. Taigman, Y., Polyak, A., Wolf, L.: Unsupervised cross-domain image generation. In: ICLR (2017) 3
71. Ulyanov, D., Vedaldi, A., Lempitsky, V.: Deep image prior. In: CVPR (2018) 4
72. Wolberg, G.: Digital image warping. IEEE computer society press (1990) 3
73. Xiao, T., Liu, Y., Zhou, B., Jiang, Y., Sun, J.: Unified perceptual parsing for scene understanding. In: ECCV (2018) 11, 12, 22
74. Xu, K., Li, Y., Ju, T., Hu, S.M., Liu, T.Q.: Efficient affinity-based edit propagation using kd tree. ACM TOG **28**(5), 1–6 (2009) 3
75. Yosinski, J., Clune, J., Bengio, Y., Lipson, H.: How transferable are features in deep neural networks? In: NeurIPS. pp. 3320–3328 (2014) 4
76. Yu, C., Wang, J., Peng, C., Gao, C., Yu, G., Sang, N.: Bisenet: Bilateral segmentation network for real-time semantic segmentation. In: ECCV. pp. 325–341 (2018) 22
77. Yu, F., Seff, A., Zhang, Y., Song, S., Funkhouser, T., Xiao, J.: Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. arXiv preprint arXiv:1506.03365 (2015) 1, 11, 12, 14, 20, 26
78. Yücer, K., Jacobson, A., Hornung, A., Sorkine, O.: Transfusive image manipulation. ACM TOG **31**(6), 1–9 (2012) 3
79. Zeiler, M.D., Fergus, R.: Visualizing and understanding convolutional networks. In: ECCV (2014) 4
80. Zhang, R., Isola, P., Efros, A.A.: Colorful image colorization. In: ECCV (2016) 3
81. Zhang, R., Isola, P., Efros, A.A., Shechtman, E., Wang, O.: The unreasonable effectiveness of deep features as a perceptual metric. In: CVPR (2018) 5, 11, 21
82. Zhang, R., Zhu, J.Y., Isola, P., Geng, X., Lin, A.S., Yu, T., Efros, A.A.: Real-time user-guided image colorization with learned deep priors. ACM TOG **9**(4) (2017) 3
83. Zhu, J.Y., Park, T., Isola, P., Efros, A.A.: Unpaired image-to-image translation using cycle-consistent adversarial networks. In: ICCV (2017) 1, 3
84. ZLL: Face-parsing pytorch. `https://github.com/zllrunning/face-parsing.PyTorch` (2019) 11, 22