Rewriting a Deep Generative Model

David Bau, Steven Liu, Tongzhou Wang, Jun-Yan Zhu, Antonio Torralba
MIT CSAIL, Adobe

ECCV’20 Oral

Presented by Eungyeup Kim

Vision Seminar
01 SEP 2020

Motivation

It has been obscure how the GANs encode the target distribution.

- A deep generative models such as GANs learn to model a rich set of semantic and physical rules about the target distribution.

- However, it has been unclear how such rules are encoded in the network, or how a rule could be changed.

- The ability to locate or edit those rules provides insights about what the model has captured and how the model can generalize to unseen scenarios.

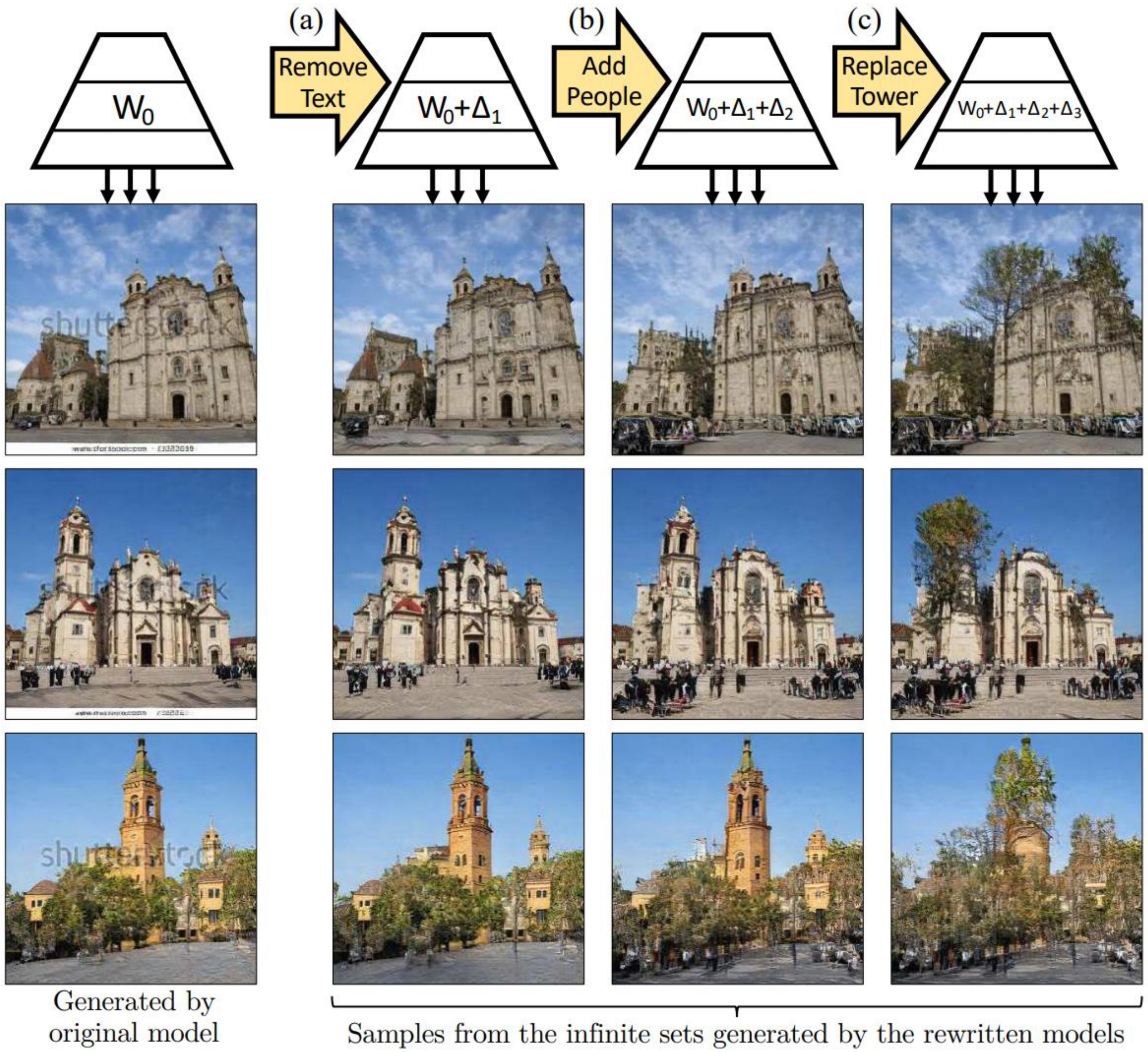
- We might want to create new data that never existed before. (Just like photoshop)

⇒ This paper presents the task of *model rewriting: add, remove, and alter the rules* of a pretrained deep network.

Introduction

Model Rewriting

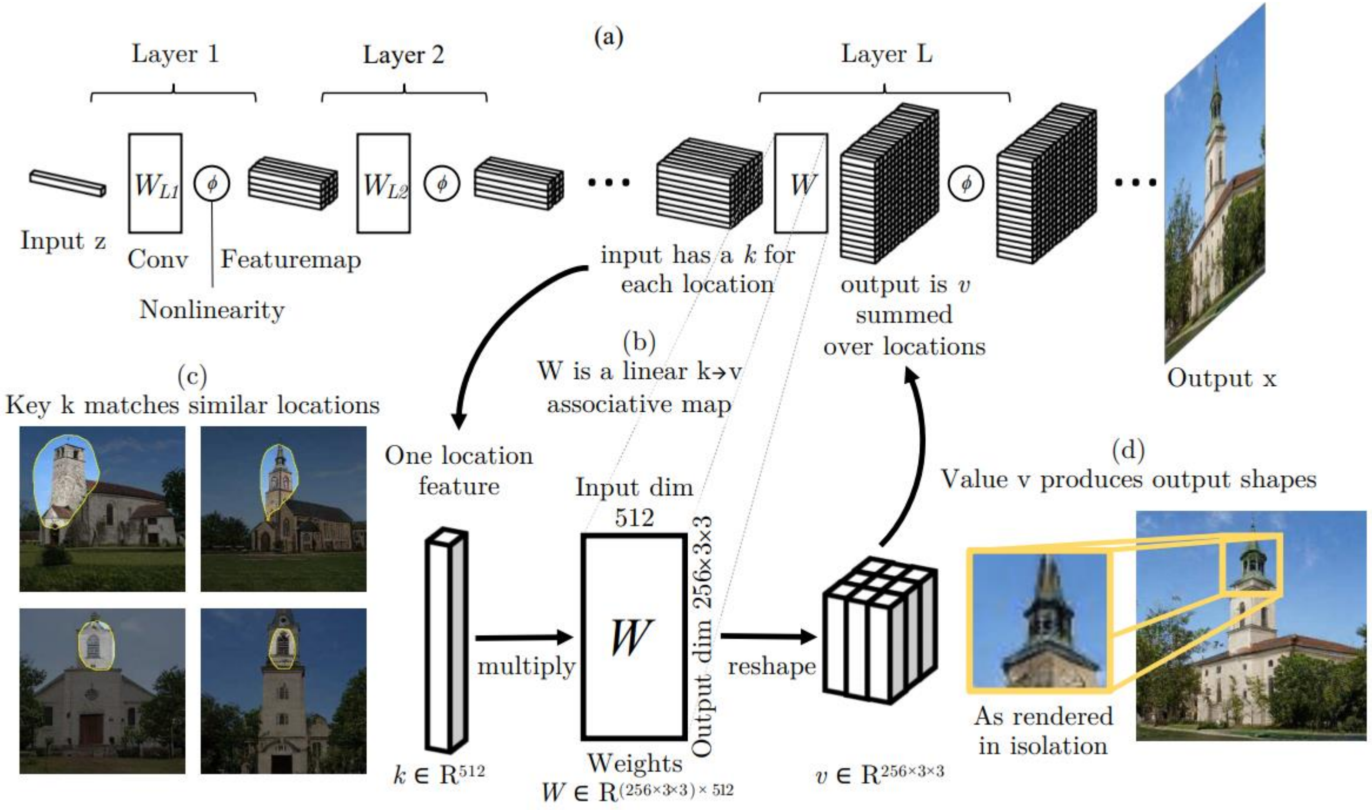
- This method changes a set of desired rules by manipulating a layer of a deep network as a linear associative memory.
- By optimizing a layer with user-provided keys and values, the model can be customized without the large-scale of training time and computational cost.
- By adding, removing and changing the rules encoded in the model, totally unseen images can be generated by the user's intention.



Backgrounds

Viewing a Convolutional Layer as an Associative Memory

- In this paper, a convolutional layer is viewed as a memory in which many independent feature mappings are memorized.
- This is appropriate especially when the layer representations in the neighbors are disentangled from one another.



- Each key k denotes a single-location feature vector.
- A convolutional layer W stores an output value v corresponding to the k. **(b)**
- The same key will match many semantically similar locations across different images. **(c)**
- v renders shapes in a small region. **(d)**

Methods

Objective (Linear)

A traditional solution to update the model weights for generating desired images x_{*i} given z_i is as below:

$$\theta_1 = arg\min_{\theta} L_{smooth}(\theta) + \lambda L_{constraint}(\theta),$$
$$L_{smooth}(\theta) = E_z[l(G(z;\theta_0), G(z;\theta)],$$
$$L_{constraint}(\theta) = \sum_i l\big(x_{*i}, G(z_i;\theta)\big),$$

where $G(z;\theta_0)$ represents a pre-trained generator, l a distance metric and x_{*i} an image containing desired changes. However, as the number of parameter θ is large, the generator quickly overfits to the new examples x_{*i} without good regularization. (..?)

$$W_1 = arg\min_{W} L_{smooth}(W) + \lambda L_{constraint}(W),$$
$$L_{smooth}(W) = E_k[\|f(k;W_0) - f(k;W)\|^2],$$
$$L_{constraint}(W) = \sum_i \|v_{*i} - f(k_{*i};W)\|^2,$$

Where given a layer L ,f denotes a computation of L itself, k an input representation vector and v an output representation vector.

Methods

Objective (Linear)

As pretrained generator with layer L and its parameter W_0,

$$W_0 = arg\min_{W} \sum_{i} \|v_i - Wk_i\|^2,$$
$$K = [k_1, k_2, \dots, k_i, \dots],$$
$$V = [v_1, v_2, \dots, v_i, \dots].$$

By using normal equation, we can obtain W_0 as below:

$$W_0 KK^T = VK^T$$

Suppose we wish to overwrite a single key to assign a new value $k_* \rightarrow v_*$ provided by the user. Then, a new parameter W_1 can be achieved as below:

$$W1 = arg\min_{W} \|V - WK\|^2,$$
$$subject\ to\ v_* = W_1 k_*.$$

By solving this constrained linear least-squares, we can obtain W_1 as below:

$$W_1 KK^T = W_0 KK^T + \Lambda k_*^T,$$
$$W_1 = W_0 + \Lambda (C^{-1} k_*)^T,$$

Where $C = KK^T$(covariance).

Methods

Objective (Linear)

$$W_1 K K^T = W_0 K K^T + \Lambda k_*^T,$$
$$W_1 = W_0 + \Lambda (C^{-1} k_*)^T.$$

The update $\Lambda (C^{-1} k_*)^T$ is a rank-one matrix with rows all multiples of the vector $(C^{-1} k_*)^T$ or d^T.

1. The update is done in a particular straight-line direction $(C^{-1} k_*)^T$.
2. The update direction is determined only by the overall key statistics (covariance) and the specific targeted key k_*.
3. Only Λ depends on the target value v_*.

We can obtain the W_1 in a closed form.

Objective (Non-Linear)

However, what if f is a non-linear function, which is very common in our neural network setting..?

$$\Lambda_1 = arg \min_{\Lambda} \sum_i \| v_* - f(k_*; W_0 + \Lambda d^T) \|^2,$$

Therefore, we can update the layer weights using $W_1 = W_0 + \Lambda_1 \mathrm{d}^{\mathrm{T}}$.

Experiments

User Interface

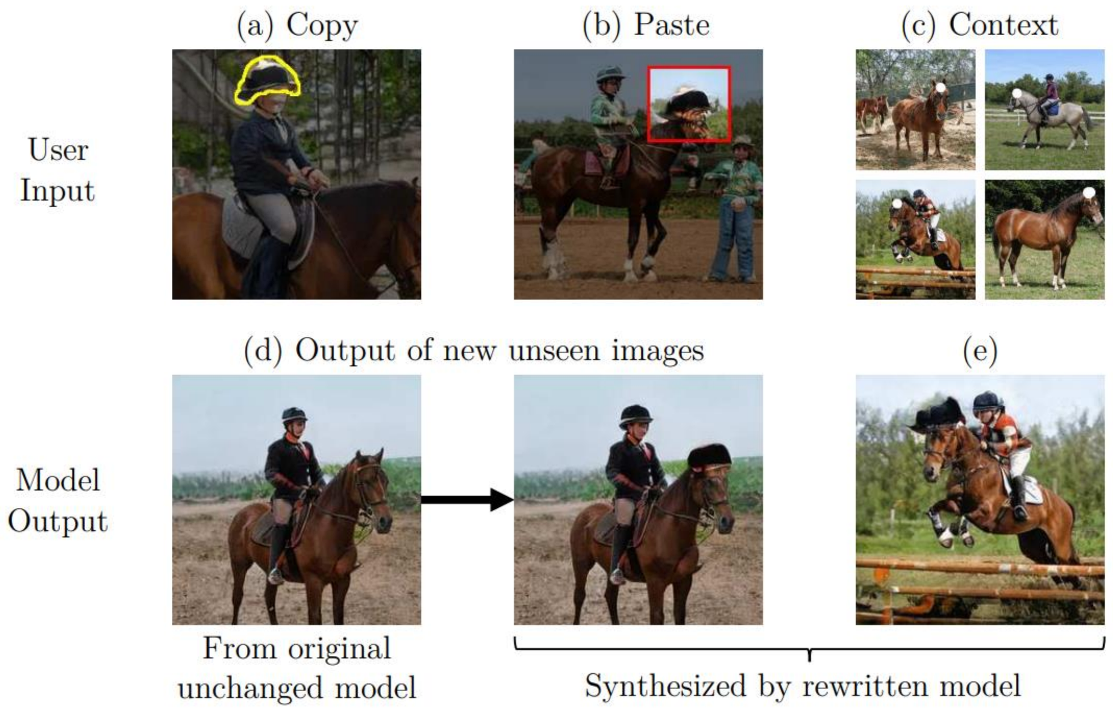
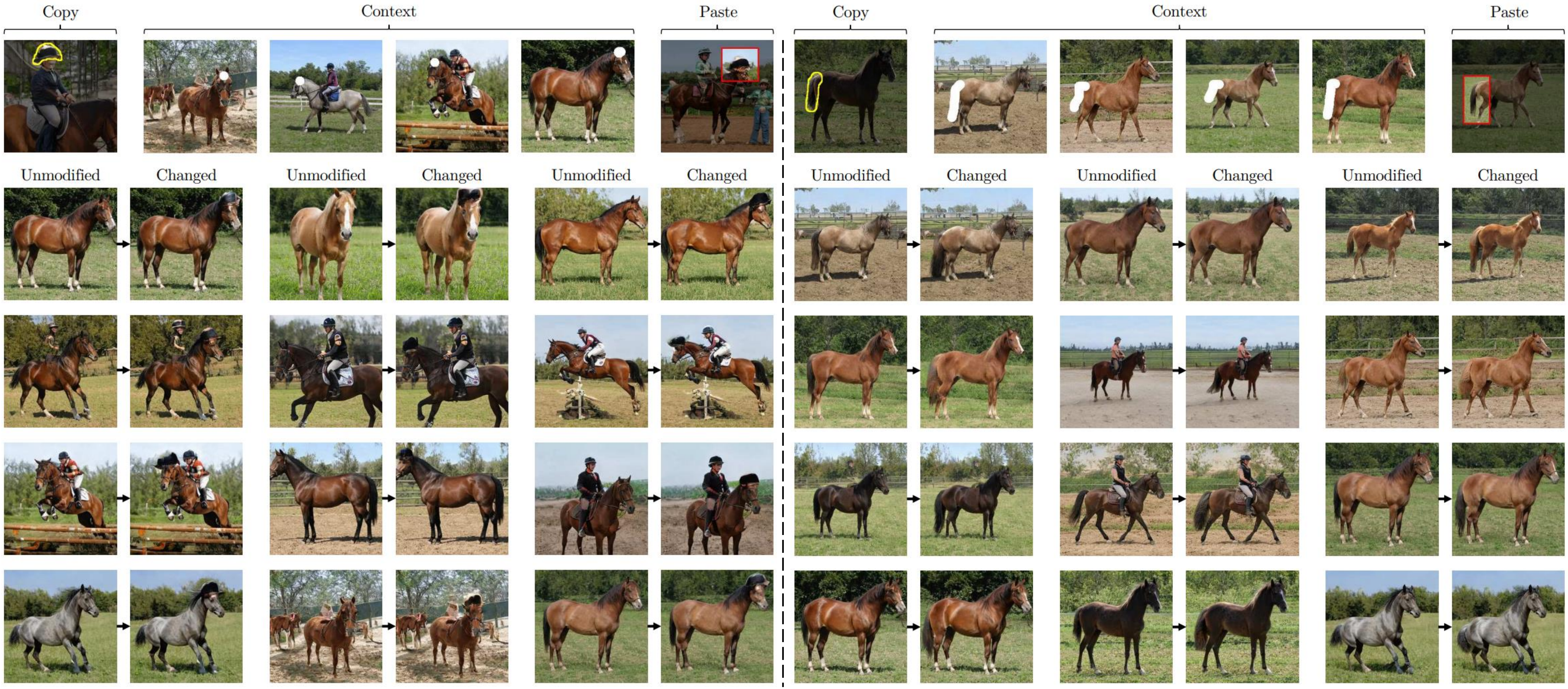


Fig. 3: The *Copy-Paste-Context* interface for rewriting a model. (a) **Copy:** the user uses a brush to select a region containing an interesting object or shape, defining the target value V_*. (b) **Paste:** The user positions and pastes the copied object into a single target image. This specifies 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).

Experiments

Putting objects into a new context



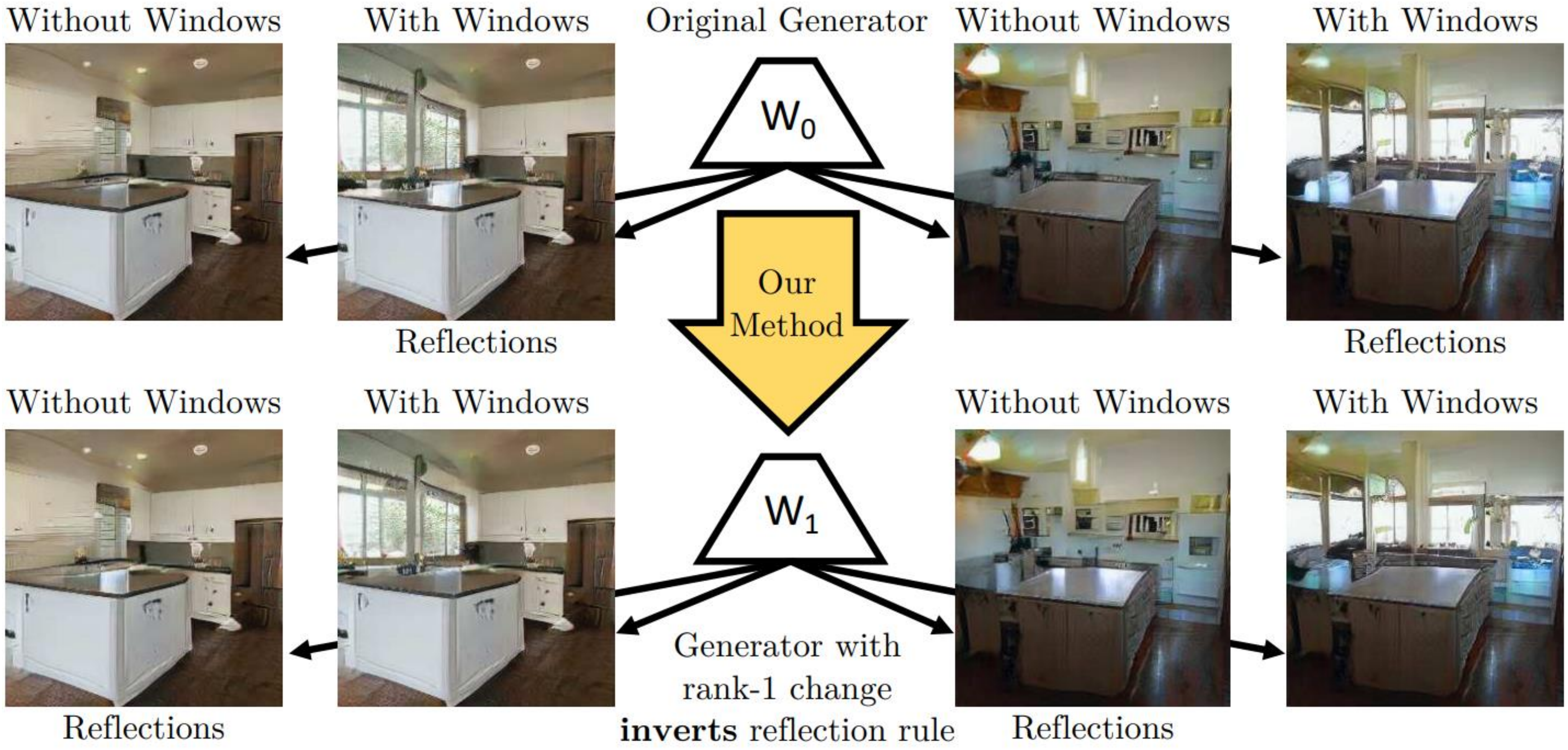
Experiments

Removing undesired features

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

Experiments

Changing contextual rules



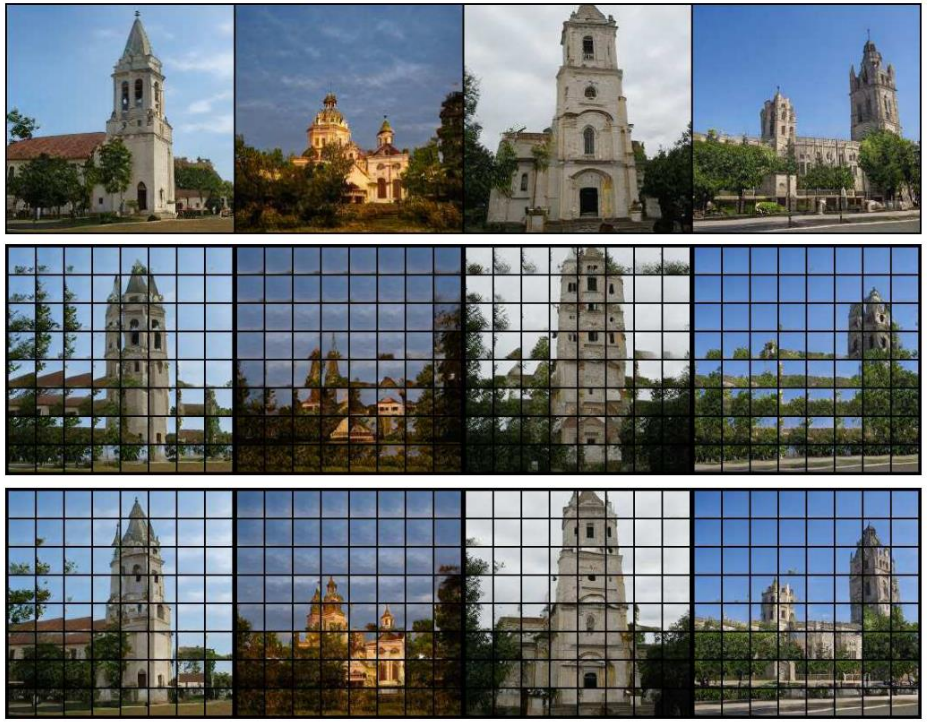
Experiments

Which layer do we have to choose?

Rendered output

Rendered layer 6 patches

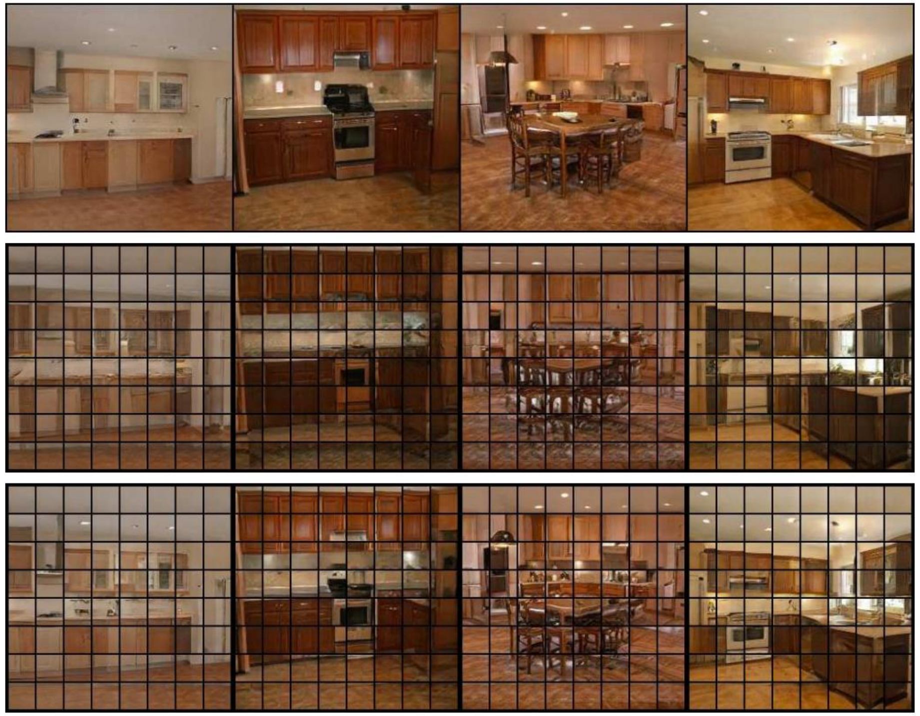
Rendered layer 8 patches

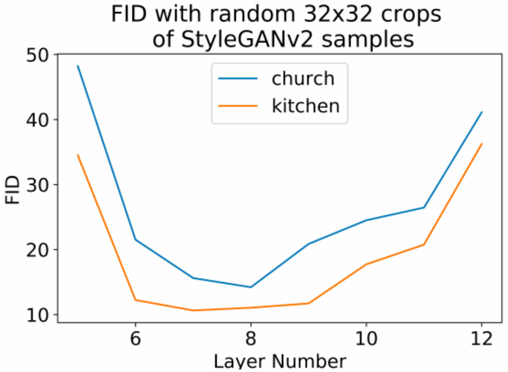


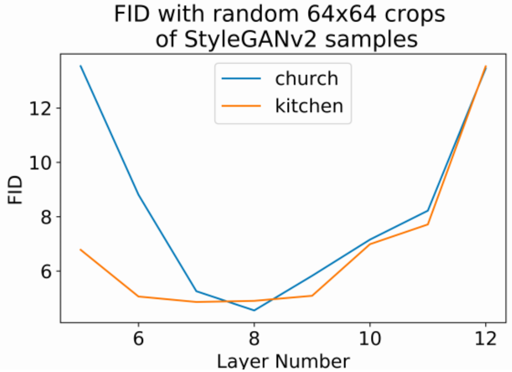
Rendered output

Rendered layer 6 patches

Rendered layer 8 patches







Thank you