Image Editing with Optimization (part II)

Jun-Yan Zhu
16-726, Spring 2022
Image Editing with Optimization

Desired output:
- stay close to the input.
- satisfy user’s constraint.

Photo: [Tao et al. 2014]
Image Editing with Optimization

Image Smoothing via L0 Gradient Minimization [Xu et al., SIGGRAPH Asia 2011]

$$\arg \min_{\hat{y}} \left\{ \| \hat{y} - x \| + \lambda C(\hat{y}) \right\}$$

L0 norm on image gradients (the total number of nonzero elements)
Things can get really bad

Image Warping

Image Composition

The lack of “safety wheels”
Adding the “safety wheels”

A desired output:
- stay close to the input.
- satisfy user’s constraint.
- Lie on the natural image manifold
Changing Variables

• Traditional method: Optimizing the image

\[ \hat{y}^* = \arg\min_{\hat{y}} \mathcal{L}(x, y, \hat{y}) \]

• New method: Optimizing the latent code

\[ z^* = \arg\min_{z} \mathcal{L}(x, y, G(z)) \]
Projecting and Editing an Image

original photo

projection on manifold

Project

different degree of image manipulation

Edit Transfer

Editing UI

transition between the original and edited projection
Projecting an Image into GAN Manifold

Input: real image $x$
Output: latent vector $z$

Optimization

$$z^* = \arg \min_z \mathcal{L}(G(z), x)$$

Inverting Network

$$z = E(x)$$
$$E = \arg \min_E \mathbb{E}_x \mathcal{L}(G(E(x)), x)$$

Hybrid Method

Use the network as initialization for the optimization problem
Manipulating the Latent Code

original photo

projection on manifold

Editing UI

transition between the original and edited projection

different degree of image manipulation

Project

Edit Transfer
Post-Processing

original photo

projection on manifold

Project

Editing UI

transition between the original and edited projection

different degree of image manipulation

Edit Transfer
Image Editing with GANs

• Step 1: Image Projection/Reconstruction

\[ z_0 = \arg \min_z \mathcal{L}(G(z), x) \]

• Step 2: Manipulating the latent code

\[ z_1 = z_0 + \Delta z \]

• Step 3: Generate the edited result

\[ G(z_1) \]
Image Projection with GANs
Baseline

• Baseline: Optimizing the latent code

\[ z^* = \arg \min_{\hat{z}} \mathcal{L}(G(\hat{z}; \theta), x) \]

\( z^* \) and \( z_0 \) are used interchangeably
Find the Differences...

Original image

GAN reconstructed image

- No microwave
- No lamp
- No chairs/table
- No window
- No stove
- No something there
- No plug
Baseline

• Baseline: Optimizing the latent code

\[ z^* = \arg \min_z \mathcal{L}(G(z; \theta), x) \]
How to Improve GANs Projection

• Baseline: Optimizing the latent code

\[ z^* = \arg\min_z \mathcal{L}(G(z; \theta), x) \]

• Generator fine-tuning:

\[ z^*, \theta^* = \arg\min_{z, \theta} \mathcal{L}(G(z; \theta), x) + R(\theta) \]
Generator Fine-tuning (Progressive GANs)

Original image

With $\mathbf{z}^*$

With $\mathbf{z}^*$ and $\mathbf{\theta}^*$


Inspired by Deep Image Prior [Ulyanov et al.] and Deep Internal learning [Shocher et al.]
Generator Fine-tuning (BigGAN)

Progressive Reconstruction
• First match semantics
• Then match color and textures

Exploiting Deep Generative Prior for Versatile Image Restoration and Manipulation
[Pan et al., ECCV 2020]
Generator Fine-tuning (BigGAN)

- **Colorization**
- **Inpainting**
- **Super-resolution**
- **Adversarial defense**

**Random jittering**
- Target: Sable
- Reconstruction: Sable
- Jittering effects: Sable

**Category transfer**
- Target: Sable
- Reconstruction: Sable
- Transfer to other categories: Sable

**Image morphing**
- Target A: Sable
- Reconstruction A: Sable
- Interpolation: Sable
- Reconstruction B: Sable
- Target B: Sable
How to Improve GANs Projection

• Baseline: Optimizing the latent code

\[ z^* = \arg\min_{z} L(G(z; \theta), x) \]

• Generator fine-tuning:

\[ z^*, \theta^* = \arg\min_{z, \theta} L(G(z; \theta), x) + R(\theta) \]

• Optimizing intermediate features

\[ w^*_+ = \arg\min_{w_+} L(g(w_+), x) \]
Using Different Layers

Optimizing the latent code

\[ z^* = \arg \min_z \mathcal{L}(G(z), x) \]

Optimizing the style code

\[ w^* = \arg \min_w \mathcal{L}(g(w), x) \]

Optimizing the extended style code

\[ w_+^* = \arg \min_{w_+} \mathcal{L}(g(w_+), x) \]

Image2StyleGAN [Abdal et al., 2019], StyleGAN2 [Karras et al., 2019]
Using Different Layers: w Space

Input

Reconstruction
Using Different Layers: w+ Space

All the results are reconstructed via the StyleGAN Face model.

Image2StyleGAN [Abdal et al., 2019]
How to Improve GANs Projection

• Baseline: Optimizing the latent code

\[ z^* = \arg \min_z \mathcal{L}(G(z; \theta), x) \]

• Generator fine-tuning:

\[ z^*, \theta^* = \arg \min_{z, \theta} \mathcal{L}(G(z; \theta), x) + R(\theta) \]

• Optimizing intermediate features

\[ w^*_+ = \arg \min_{w+} \mathcal{L}(g(w_+), x) \]
Generator Fine-tuning with w+ Space

Pivotal Tuning for Latent-based Editing of Real Images [Roich et al., 2021]
How to Improve GANs Projection

• Baseline: Optimizing the latent code

\[ z^* = \arg \min_{z} \mathcal{L}(G(z; \theta), x) \]

• Training an encoder E(x). Advantages?
  – Faster inference
  – More reliable initialization

• Encoder design depends on
  – Generator architecture.
  – Which latent space: z, w, w+.
  – Pre-trained network weights.
Encoding in Style: a StyleGAN Encoder for Image-to-Image Translation
[Richardson et al., CVPR 2021]
Example: An StyleGAN Encoder
Debugging GANs Projection (HW5)

\[ z_0 = \arg \min_z \mathcal{L}(G(z), x) \]

• What can go wrong?
  – Generator: G (cannot generate the image or too deep)
  – Reconstruction loss: L (not a good image distance)
  – Optimization method: SGD, ADAM (local minimum)
    (1) use a more advanced solver: e.g., L-BFGS (Quasi-Newton)
    (2) train an encoder to initialize the latent code. E(x)

• Debugging steps:
  – Reconstruct a generated image
  – Reconstruct a training set real image
  – Reconstruct a validation/test set real image
  – Reconstruct an in-the-wild image (e.g., Internet photo, camera roll)
Reconstruction ≠ Editing

Interpolations between two images

Image2StyleGAN [Abdal et al., 2019]
Image Editing with GANs

• Step 1: Image Projection/Reconstruction

\[ z^*, \theta^* = \arg \min_{z, \theta} \mathcal{L}(G(z; \theta), x) + R(\theta) \]

• Step 2: Manipulating the latent code

\[ z_1 = z_0 + \Delta z \]

• Step 3: Generate the edited result

\[ G(z_1) \]
Manipulating Latent code/layer
(computing directions offline)
Compute $\Delta z$

Step 1: annotate images (manually or via a pre-trained classifier)
Step 2: compute directions

DCGAN [Radford et al. 2016]
Manipulating Latent code/layer (PCA directions)
GANSpace: Discovering PCA directions

First find compute potential directions (PCA), then name them
GANSpace: Discovering PCA directions

- z: latent codes.
- y: intermediate features.
- v: PCA direction in feature space.
- u: PCA direction in latent space.

Also see “Editing in Style: Uncovering the Local Semantics of GANs”, Collins et al., CVPR 2020
“Closed-Form Factorization of Latent Semantics in GANs”, Shen and Zhou. CVPR 2021
GANSpace: Discovering PCA directions
Manipulating Latent code/layer (offline optimization)
Given a pre-defined function $\text{edit}$ and a pre-trained generator $G$

**Linear case:**

$$(w \text{ is a vector)}$$

$$\arg \min_w \mathbb{E}_{z, \alpha} [\mathcal{L}(G(z + \alpha w), \text{edit}(G(z), \alpha))]$$

**Non-linear case:**

$$(f \text{ is a function)}$$

apply it $n$ times

$$\arg \min_f \mathbb{E}_{z, n} [||G(f^n(z)) - \text{edit}(G(z), n\epsilon)||]$$
Offline optimization

Requirement: A known **edit** function (e.g., shift, zoom, rotate)

[Jahanian*, Chai*, Isola. 2020]
CLIP-guided Directions

\[
\arg\min_{\mathcal{W}+} D_{\text{CLIP}}(G(w), t) + \lambda_{L2} \|w - w_s\|_2 + \lambda_{ID} L_{ID}(w)
\]

Output is close to the text  
Output is close to the original latent  
Output is close to input

StyleCLIP: Text-Driven Manipulation of StyleGAN Imagery [Patashnik et al., ICCV 2021]
**Input:** an image and a caption.

**Output:** similarity between the text embedding and the image embedding

---

**StyleCLIP:** Text-Driven Manipulation of StyleGAN Imagery [Patashnik et al., ICCV 2021]
CLIP-guided Directions

\[ \arg \min_{w \in \mathcal{W}^+} D_{\text{CLIP}}(G(w), t) + \lambda_{L2} \| w - w_s \|_2^2 + \lambda_{ID} \mathcal{L}_{ID}(w) \]

Output is close to the text  
Close to the original latent  
Output is close to input
GAN Inversion Demo
Manipulating network weights
Image Editing with GANs

• Step 1: Image Projection/Reconstruction

\[ z_0, \theta_0 = \arg \min_{z, \theta} \mathcal{L}(G(z; \theta), x) \]

• Step 2: Manipulating the network weights

\[ \theta_1 = \theta_0 + \Delta \theta \]

• Step 3: Generate the edited result

\[ G(z_0; \theta_1) \]
Understanding a Generator

Each layer consists of a set of neurons.

Each step:
Increases spatial resolution

Input: random vector $z$

What happens inside?

Examine each neuron

Output image

Progressive GANs [Karras et al. 2018]
Which neurons **correlate** to an object class?

Which neurons **correlate** to an object class?

Which neurons **correlate** to an object class?

Which neurons *correlate* to an object class?
Which neurons **correlate** to an object class?

Church samples

Tree Neuron

Dome Neuron
Which neurons correlate to an object class?

Dining room samples

252 out of 512 neurons are correlated to objects, part, and materials
Which neurons **correlate** to an object class?

Living room model

- Layout
- Object and parts
- Edges, textures, local structure
Which neurons **cause** an object class?

Which neurons **cause** an object class?

Which neurons **cause** an object class?

Which neurons **cause** an object class?

- **Single neuron**
  - Featuremap
  - Turn off the neuron
  - Turn on the neuron

- **Generated image**
  - Image with neuron off
  - Image with neuron on

- **Average causal effect**
Which neurons **cause** an object class?
Object-Scene Relationships

Turn off person neurons
Object-Scene Relationships

Turn off \texttt{window} neurons
Object-Scene Relationships

Turn off table neurons
Object-Scene Relationships

Turn off chair neurons
Object-Scene Relationships

Ablating Conference Room Generator Units

- person
- window
- curtain
- table
- chair
Object-Scene Relationships

Yellow box: highlight every location where we can insert doors
Object-Scene Relationships

Yellow box: highlight every location where we can insert doors
Object-Scene Relationships

Where Can a Door Go?

Average Causal Effect

- window
- brick
- building
- tree
- sky

67
Debugging and Improving Models

Turning off neurons with artifacts

Example artifact-causing neurons

Neuron #63

Neuron #231
Interactive Painting

Select a feature brush & strength and enjoy painting:

- tree
- grass
- door
- sky
- cloud
- brick
dome

draw remove
undo reset

Online Demo
Interactive Painting

Select a feature brush & strength and enjoy painting:

- tree
- grass
- door
- sky
- cloud
- brick
- dome

Online Demo

Manipulating a Real Photo

Original image + edits  
Editing with $\hat{z}$  
Editing with $\hat{z}$ and $\hat{\theta}$

Manipulating a Real Photo

Input image  Remove chairs  Output result

Manipulating a Real Photo

Input image  Add windows  Output result

Manipulating a Real Photo via GAN Dissection

Input image  Restyle trees for spring  Restyle trees for autumn
