Generative visual manipulation on the natural image manifold

指導教授: 莊仁輝
研究助理: 何智輝

Jun-Yan Zhu, Philipp Krähenbühl, Eli Shechtman and Alexei A. Efros.
"Generative Visual Manipulation on the Natural Image Manifold",
in European Conference on Computer Vision (ECCV). 2016.

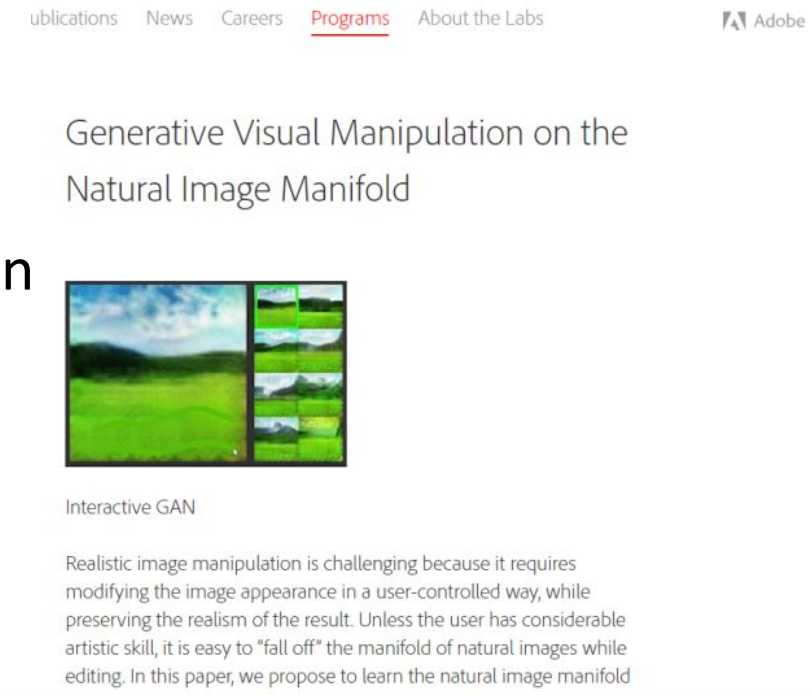
2017/03/14

Outline

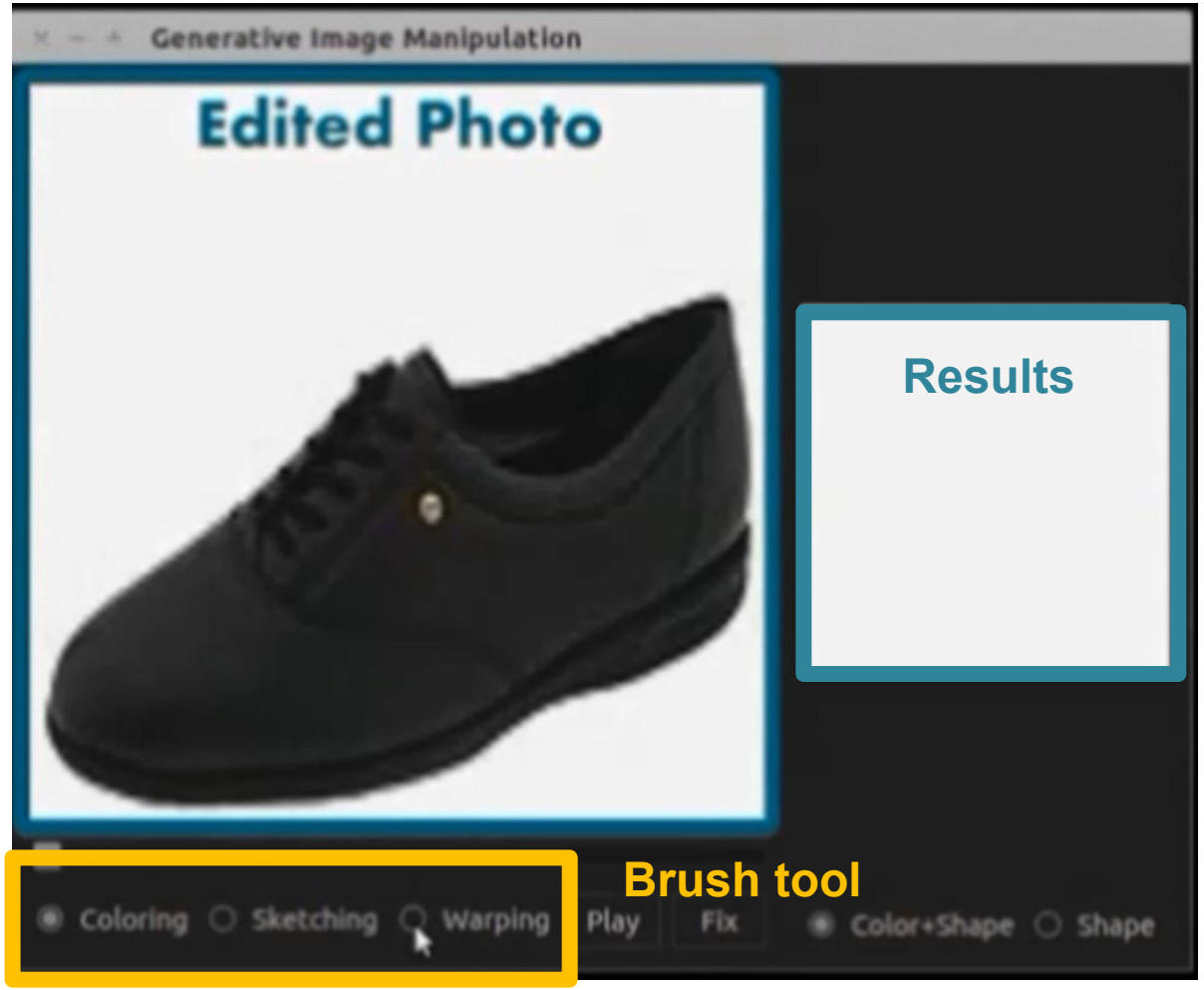
- Introduction
- Generative adversarial neural network (GAN)
- Learning natural image manifold
- Approach
  a) Projecting an real image onto approximated manifold
  b) Manipulating the latent vector
  c) Gradient descent update
  d) Edit transfer

Introduction

- The system is an editing tool that allows realistic manipulation of photos in real-time.
- It is a data driven approach to edit image, including
  - changing the shape and color
  - transforming one image to another
  - generating a new image from scratch
- Application: online shopping scenario, image manipulation



Introduction

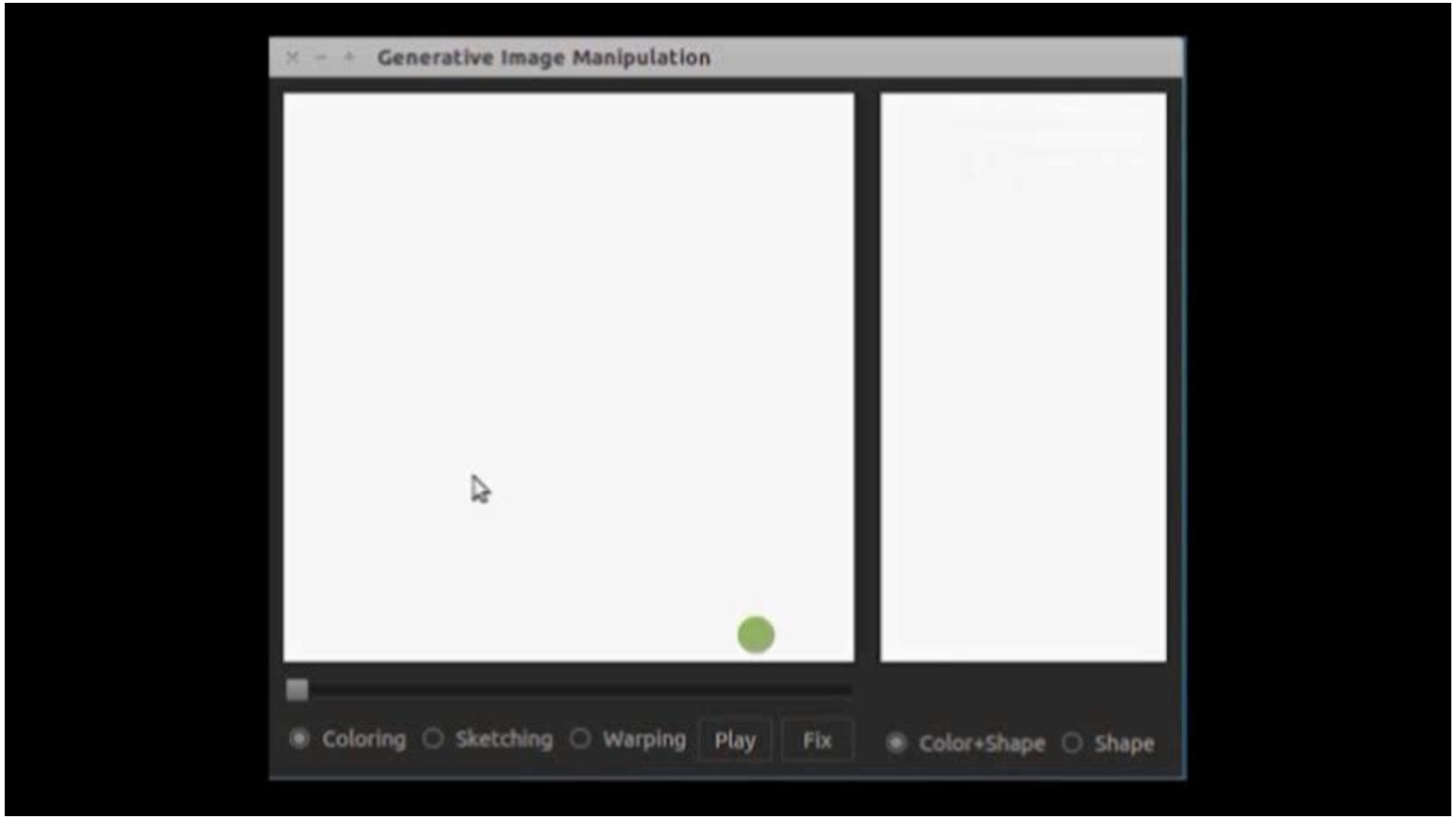


Coloring

Sketching

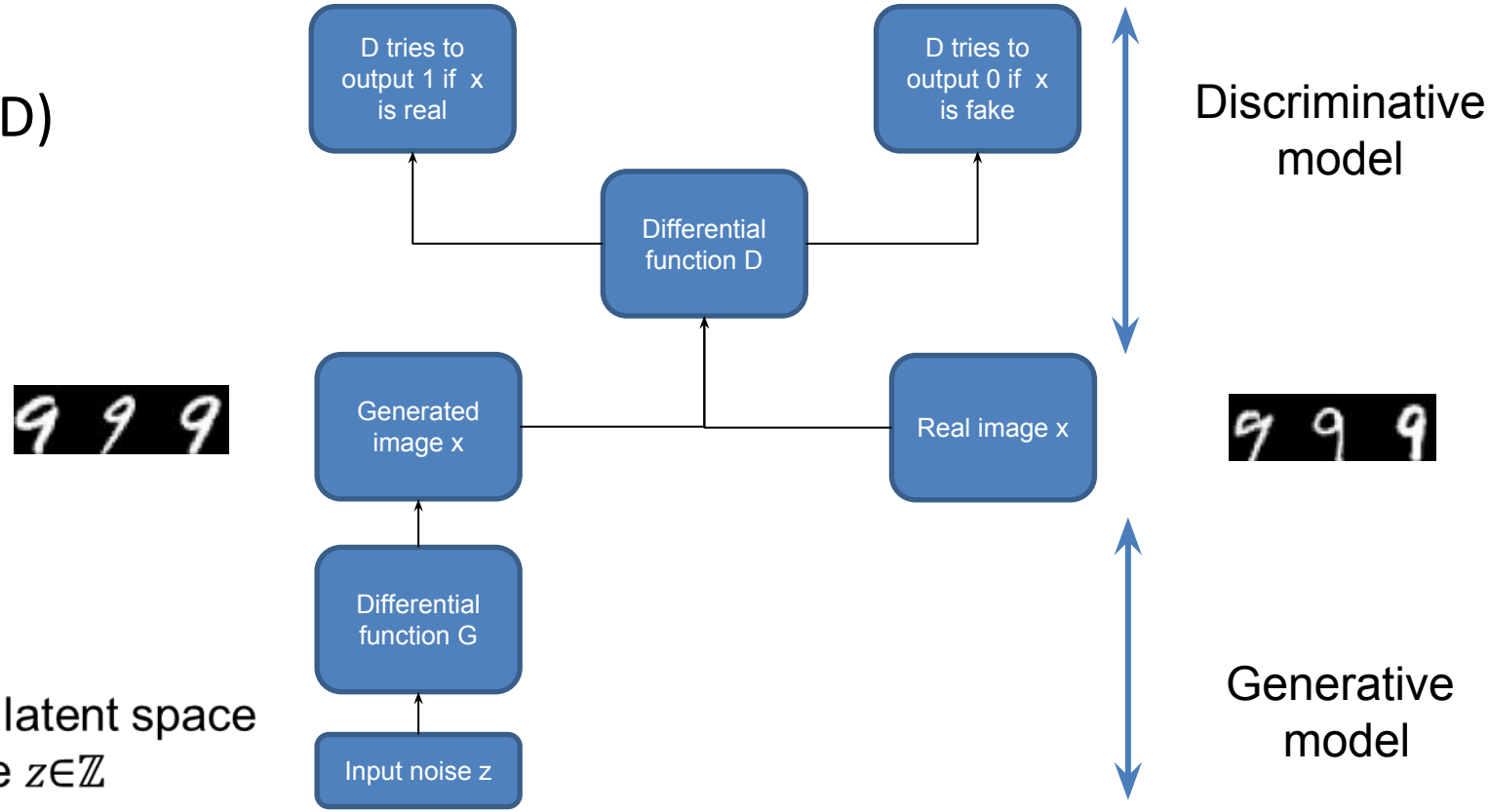
Warping

Introduction



GAN

- Train lots of images belongs to the same class on GAN
- A GAN model consists of two networks
  - Generative model (G)
  - Discriminative model (D)



- $\mathbb{Z}$ denotes a d-dimensional latent space
- z is a random vector, where $z \in \mathbb{Z}$

GAN

- Generative model (G):
  1. Goal : captures the real data distribution and produces a fake image that looks real
  2. Input : a random vector z
  3. Output : a generated image $G(z)$
  4. Objective: $\min\{\log(1 - D(G(z)))\}$

	D thinks G(z) is real	D thinks G(z) is fake
Objective value		

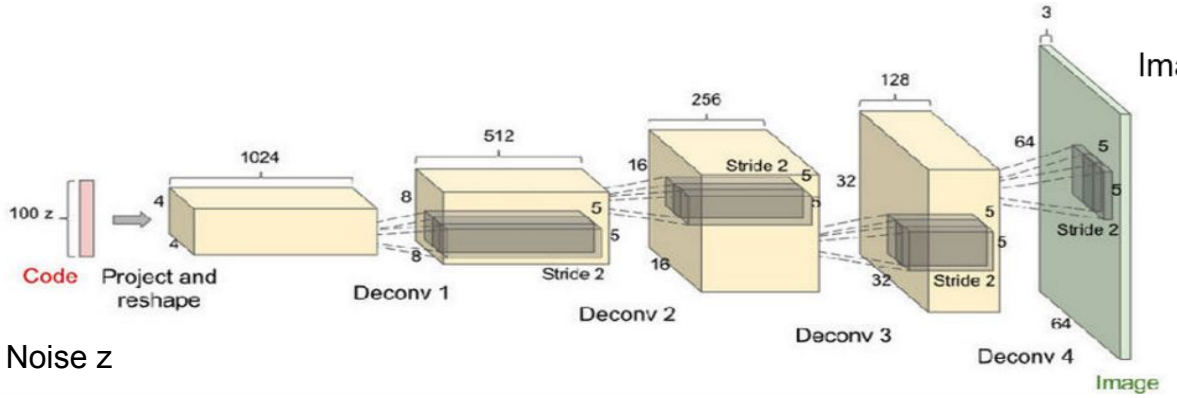


Image x

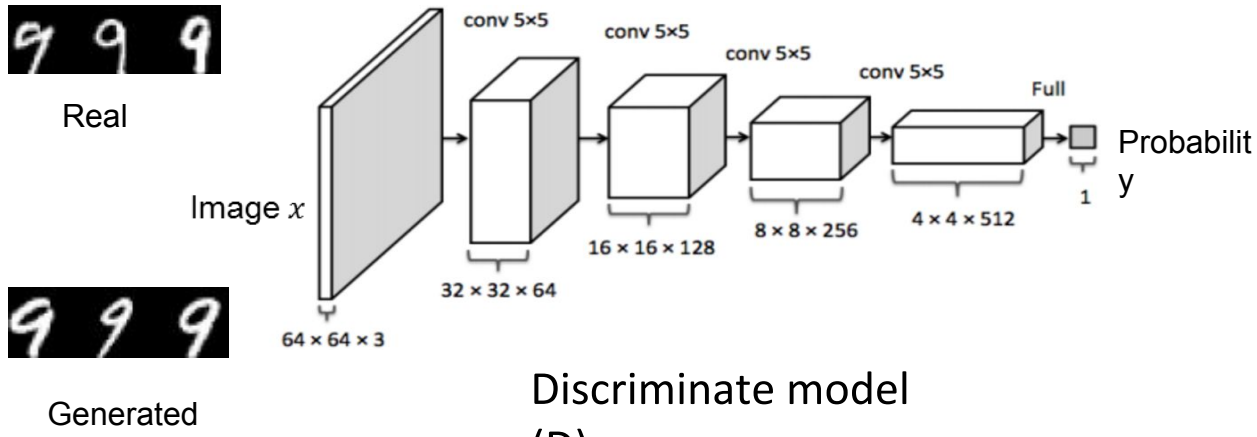
Generated

Generative model (G)

GAN

- Discriminative model (D)
  1. Goal : distinguishes between a real image x^R or a fake image $G(z)$
  2. Input : a real image x^R or a fake image $G(z)$
  3. Output : a probability whether the image is real or fake
  4. Objective: $\max\{\log(D(x^R)) + \log(1 - D(G(z)))\}$

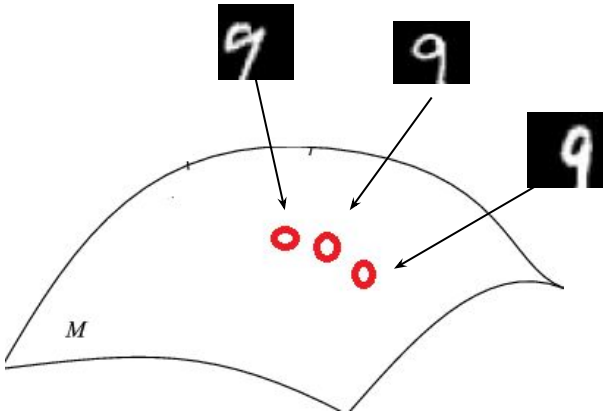
	D thinks G(z) is real and x is fake	D thinks G(z) is fake and x is real
Objective value		



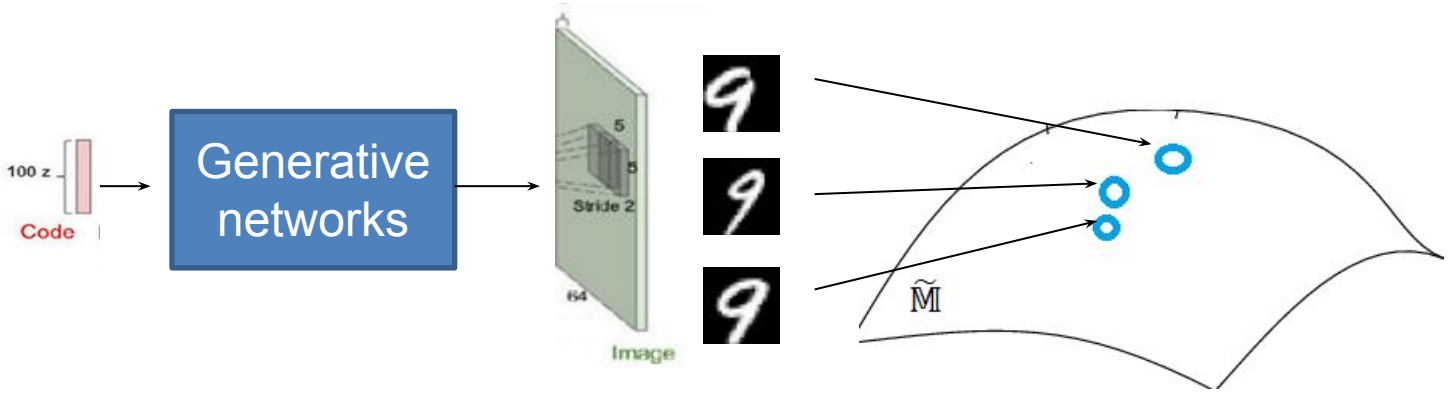
Discriminate model (D)

Learning natural image manifold

- Obtain a generative network G and a discriminative network D
- Assume all natural images lie on low dimension real image manifold $\mathbb{M}$ with distance function $\mathcal{S}(x_1, x_2)$, where $x_1, x_2 \in \mathbb{M}$
- Define approximated image manifild $\tilde{\mathbb{M}} = \{\mathrm{G}(\mathrm{z}) | \, z \in \mathbb{Z}\}$



Real image manifold $\mathbb{M}$

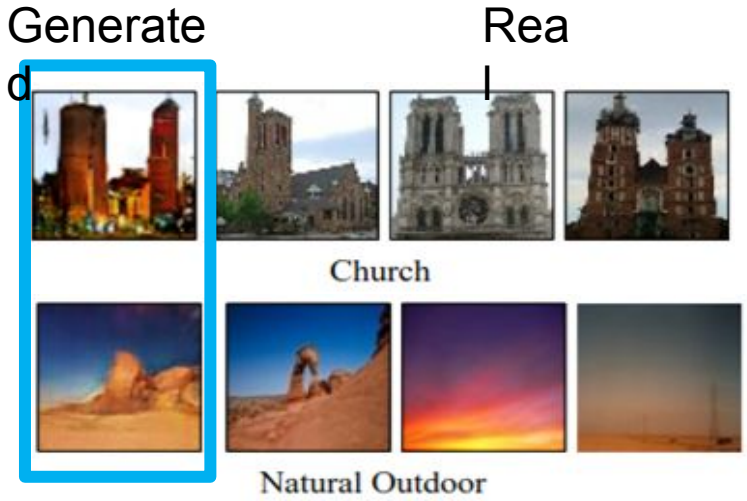


Approximated image manifild $\tilde{\mathbb{M}}$

Learning natural image manifold

- View GAN as a manifold approximation, because GAN produces high quality samples.
- A well-trained GAN should not memorize the training data.

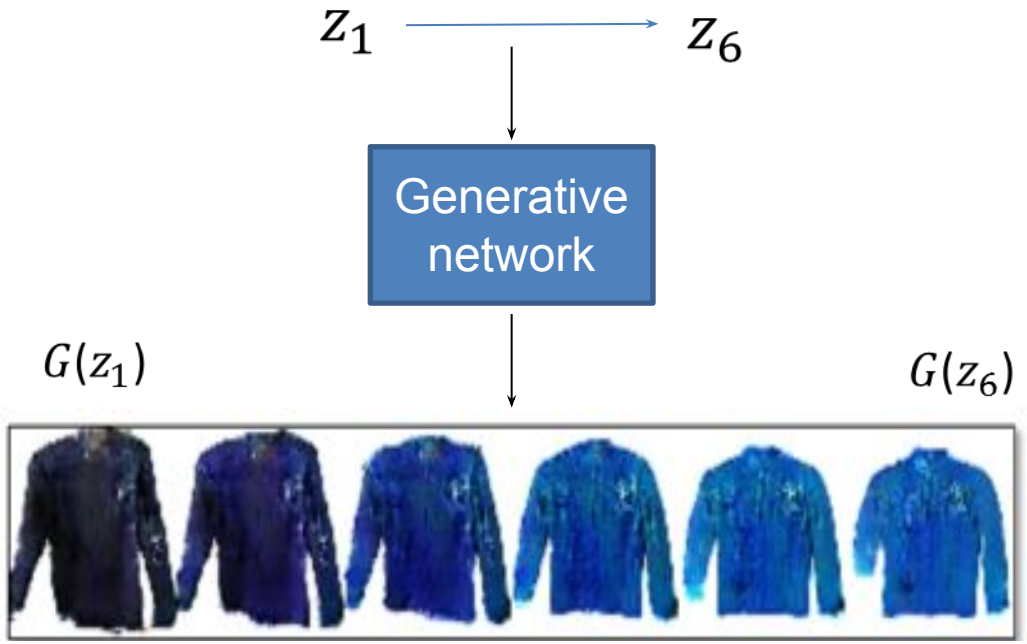
(a) Randomly generated samples from a GAN, trained on shirts dataset.



(b) The difference between generated image and real images shows that GAN does not memorize the real image

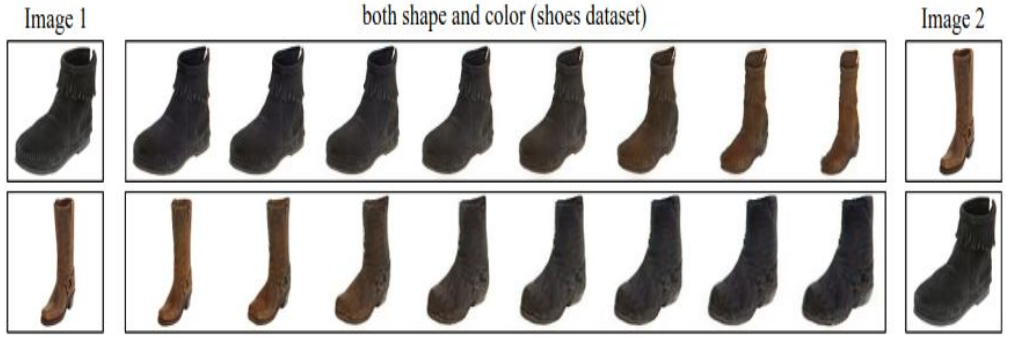
Learning natural image manifold

- View GAN as a manifold approximation, because the Euclidean distance in the $\mathbb{Z}$ corresponds to perceptual similarity
- Therefore, we approximate $\mathcal{S}(G(z_1),G(z_2)) \approx \|z_1 - z_2\|^2$



Fix z_1 and z_6 and interpolate z_2 to z_5 in between.
The generated images show perceptual similarity.

Learning natural image manifold



Approach

Real image manifold $\mathbb{M}$

(a) original photo

(e) different degree of image manipulation

Project

Edit Transfer

Approximated image manifild $\widetilde{\mathbb{M}}$

(b) projection on manifold

(c) Editing UI

(d) smooth transition between the original and edited projection

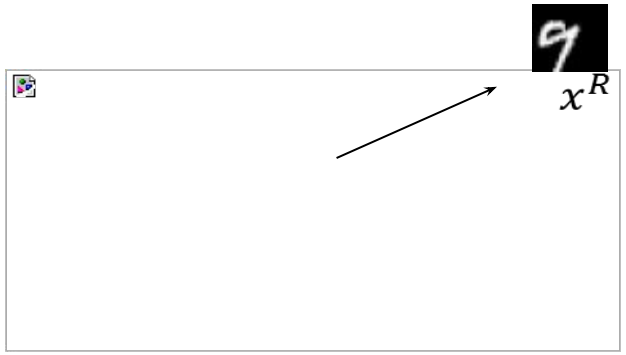
Projecting an real image onto approximated manifold

- A real photo x^R is projected to approximated manifold $\tilde{\mathbb{M}}$
- Find a generated image $x^* \in \tilde{\mathbb{M}}$ close to x^R
- $x^* = arg \min_{x \in \tilde{\mathbb{M}}} \mathcal{L}(x, x^R) \rightarrow z^* = arg \min_{z \in \mathbb{Z}} \mathcal{L}(G(z), x^R)$
- $\mathcal{L}$ is a distance metric

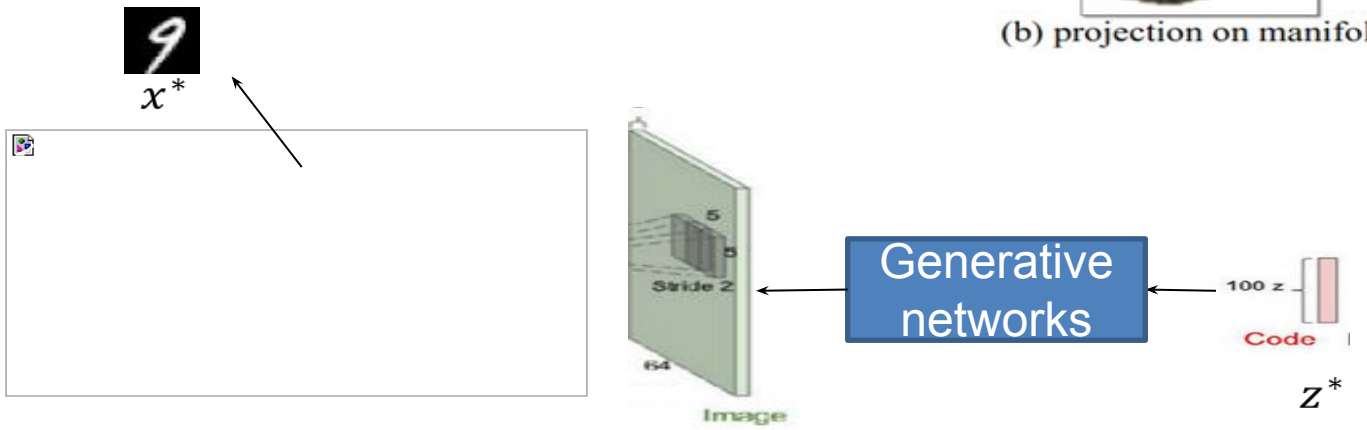
(a) original photo

Project

(b) projection on manifold



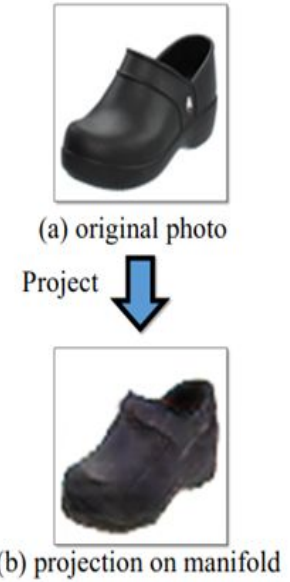
Real image manifold $\mathbb{M}$

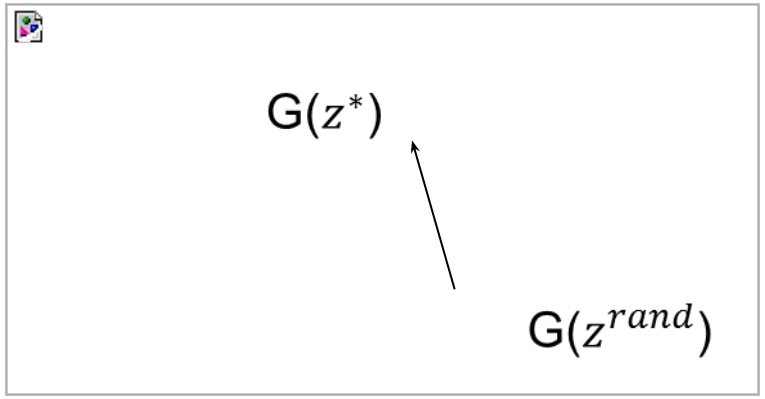


Approximated image manifild $\tilde{\mathbb{M}}$

Projecting an real image onto approximated manifold

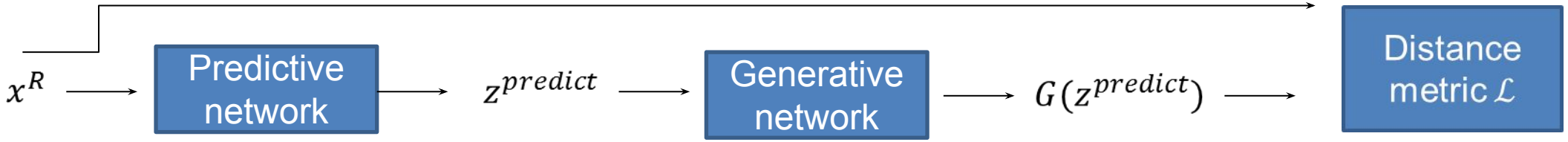
- $\mathcal{L}$ is a distance metric, defined as $\mathcal{L}(x_1, x_2) = \|C(x_1) - C(x_2)\|^2$
- C is a feature extractor, which is the weighted combination of raw pixels and conv4 features from AlexNet trained on ImageNet
- $z^* = arg\min_{z \in \mathbb{Z}} \mathcal{L}(G(z), x^R) \rightarrow z^* = arg\min_{z \in \mathbb{Z}} \|C(G(z)) - C(x^R)\|^2$
- Too difficult to optimize $C(G(z))$ in real time via random initialization of z





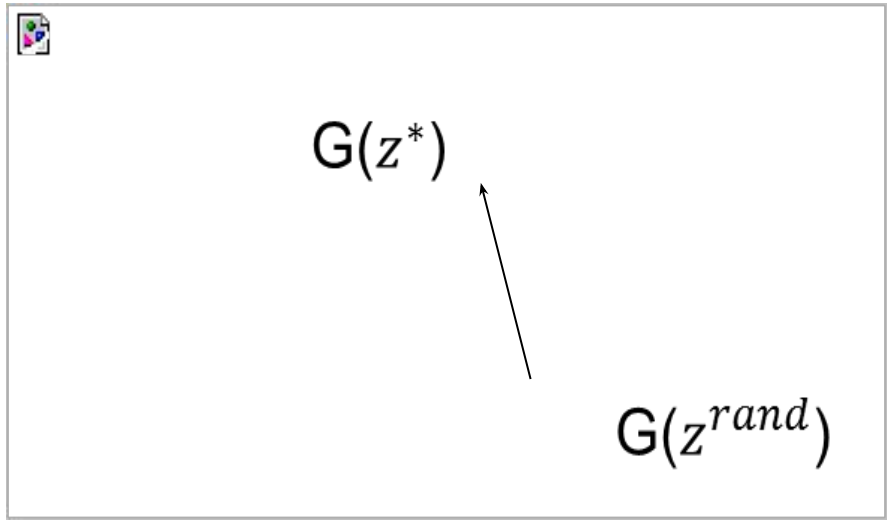
Projecting an real image onto approximated manifold

- Instead of random initialization, predict the z
- Train a feed forward neural network P and predict z from a x^R.
- $\theta_p{}^* = arg\min_{\theta_p} \sum_n \mathcal{L}(G(P(x_n{}^R, \theta_p)), x^R)$, where $x_n{}^R$ denotes the n-the image in the dataset
- Fix G throughout the training
- The network architecture of P is equivalent to D

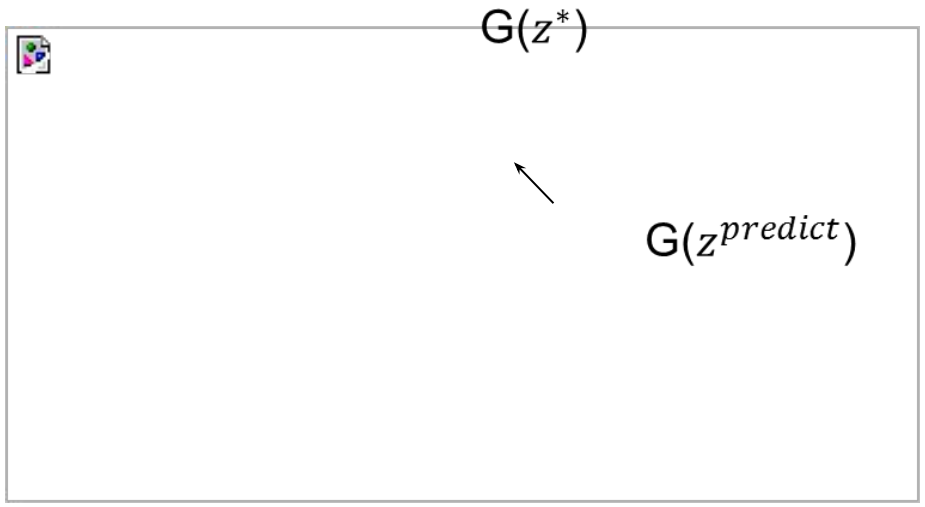


Projecting an real image onto approximated manifold

- The system optimizes $z^* = arg \min_{z \in \mathbb{Z}} \|C(G(z)) - C(x^R)\|^2$ starting from $z^{predict}$



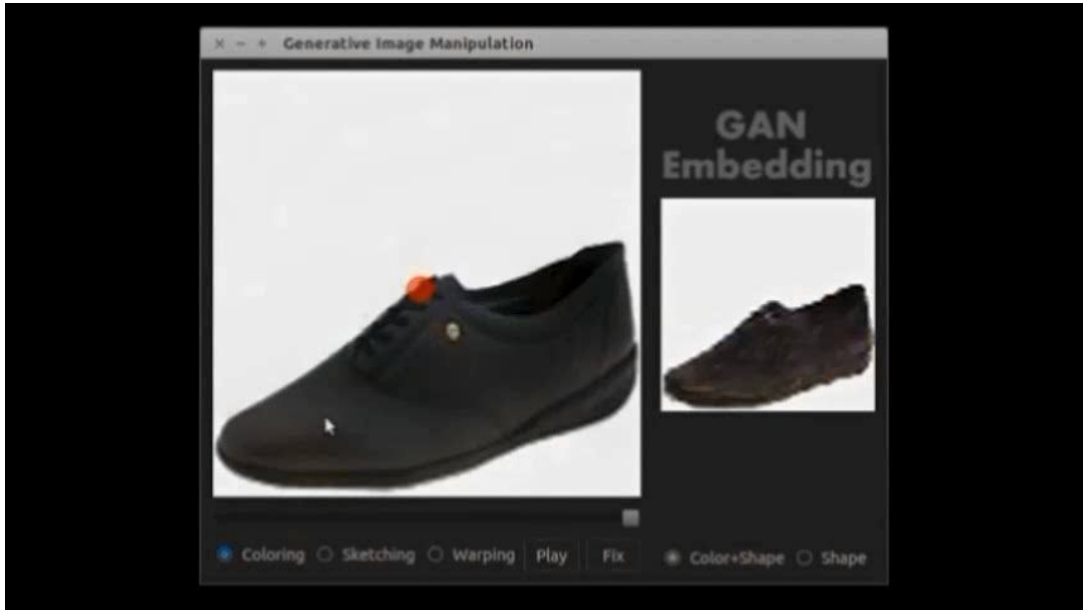
starting from z^{rand}



starting from $z^{predict}$

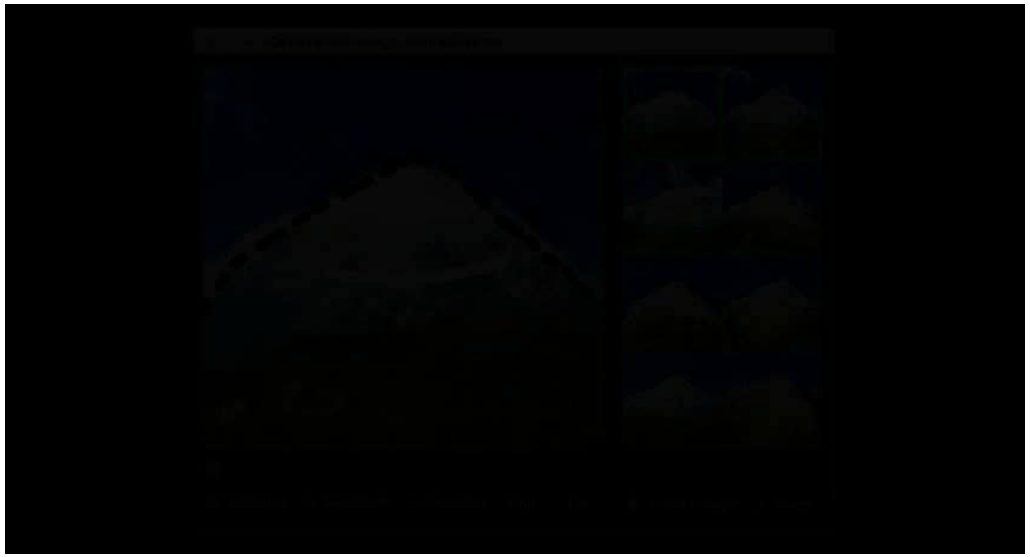
Manipulating the latent vector

- After the previous step, a real image ${x_0}^R$ is projected onto $\widetilde{\mathbb{M}}$ as $x_0 = \mathrm{G}(z_0)$
- Update x_0 by matching the user intention and stay on the manifold close to x_0
- An edit operation is formulated as a constraint $f_g(x) = v_g$
  - If g=color, we constrain pixel p to the selected color v_g



Manipulating the latent vector

- After the previous step, a real image ${x_0}^R$ is projected onto $\widetilde{\mathbb{M}}$ as $x_0 = \mathrm{G}(z_0)$
- Update x_0 by matching the user intention and stay on the manifold close to x_0
- An edit operation is formulated as a constraint $f_g(x) = v_g$
  - If g=color, we constrain pixel p to the selected color v_g
  - If g=sketch, we constrain pixel p close to $v_g = HOG(stroke)_p$



Manipulating the latent vector

- After the previous step, a real image ${x_0}^R$ is projected onto $\tilde{\mathbb{M}}$ as $x_0 = \mathrm{G}(z_0)$
- Update x_0 by matching the user intention and stay on the manifold close to x_0
- An edit operation is formulated as a constraint $f_g(x) = v_g$
  - If g=color, we constrain pixel p to the selected color v_g
  - If g=sketch, we constrain pixel p close to $v_g = HOG(stroke)_p$

- Goal $x^* = arg\min_{x \in \mathbb{M}} \left\{ \sum_g \left\| f_g(x) - v_g \right\|^2 + \lambda_s \mathcal{S}(x, x_0) \right\}$
- On $\tilde{\mathbb{M}}$, the goal becomes $z^* = arg\min_{z \in \mathbb{Z}} \left\{ \sum_g \left\| f_g\left(G(z)\right) - v_g \right\|^2 + \lambda_s \|z - z_0\|^2 \right\}$

Gradient descent update

- Solve $z^* = arg\min_{z \in \mathbb{Z}} \left\{ \sum_g \left\| f_g\left(G(z)\right) - v_g \right\|^2 + \lambda_s \|z - z_0\|^2 \right\}$ using gradient descent

User constraint v_g at different update step

Update images according to user's edit

z_0 z_1

Gradient descent update

- Once the system obtain the final result $G(z_1)$, the user can see the interpolation sequence between z_0 and z_1
- $G\left(\left(1-\frac{t}{N}\right)z_0+\frac{t}{N}z_1\right)$ is the interpolation sequence

$G(z_0)$ $G(z_1)$

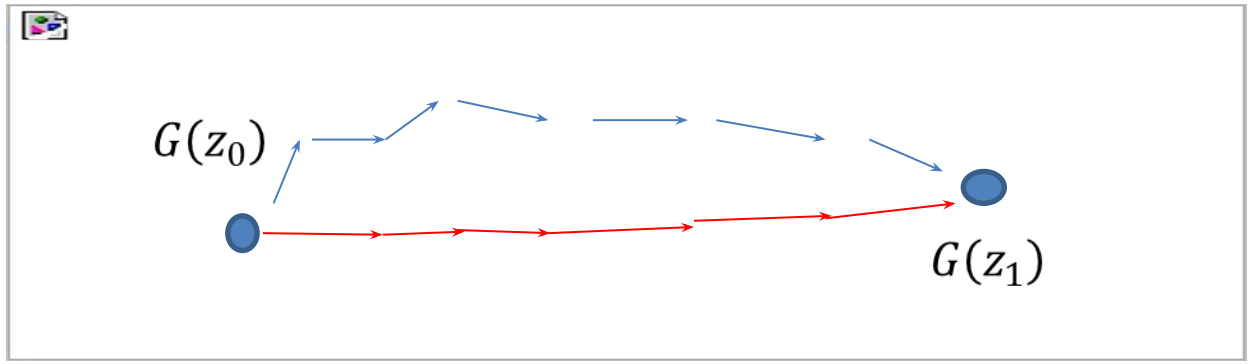
t=0 t=1 t=2 t=3 t=4 t=5 t=6 t=7

Result of N = 7

Gradient descent update

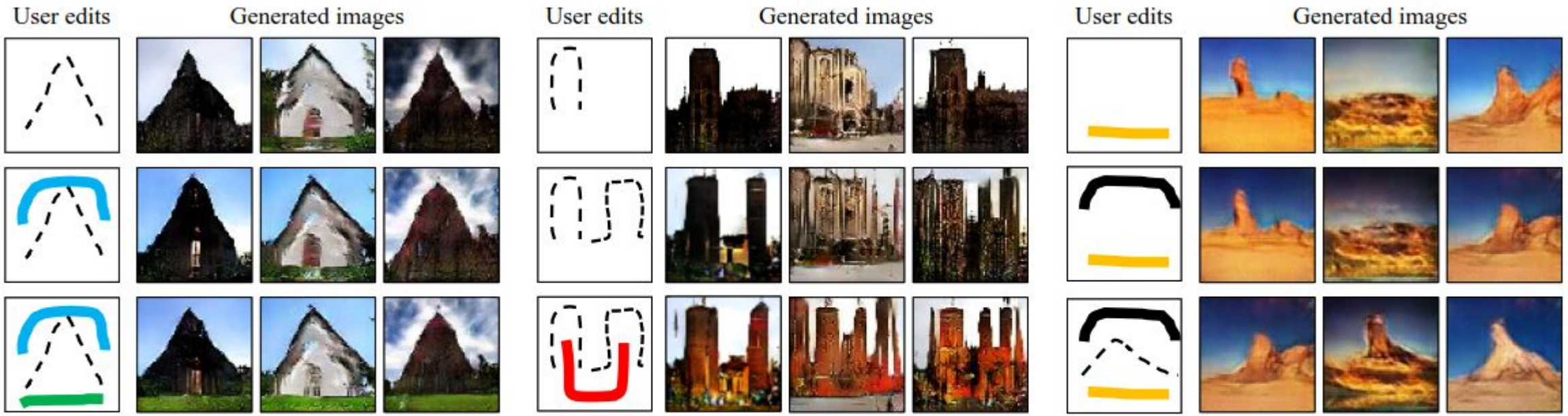
User constraint v_g at different update step

Update images according to user's edit



t=0 t=1 t=2 t=3 t=4 t=5 t=6 t=7

Gradient descent update



Edit transfer

- Given $G(z_0)$, $G(z_1)$ and interpolation results, the system will project them back to $\mathbb{M}$
- A straightforward way ${x_1}^R = {x_0}^R + (G(z_1) - G(z_0))$
- Due to the misalignment, the straightforward way does not produce good result
- Use the motion and color flow method

Edit transfer

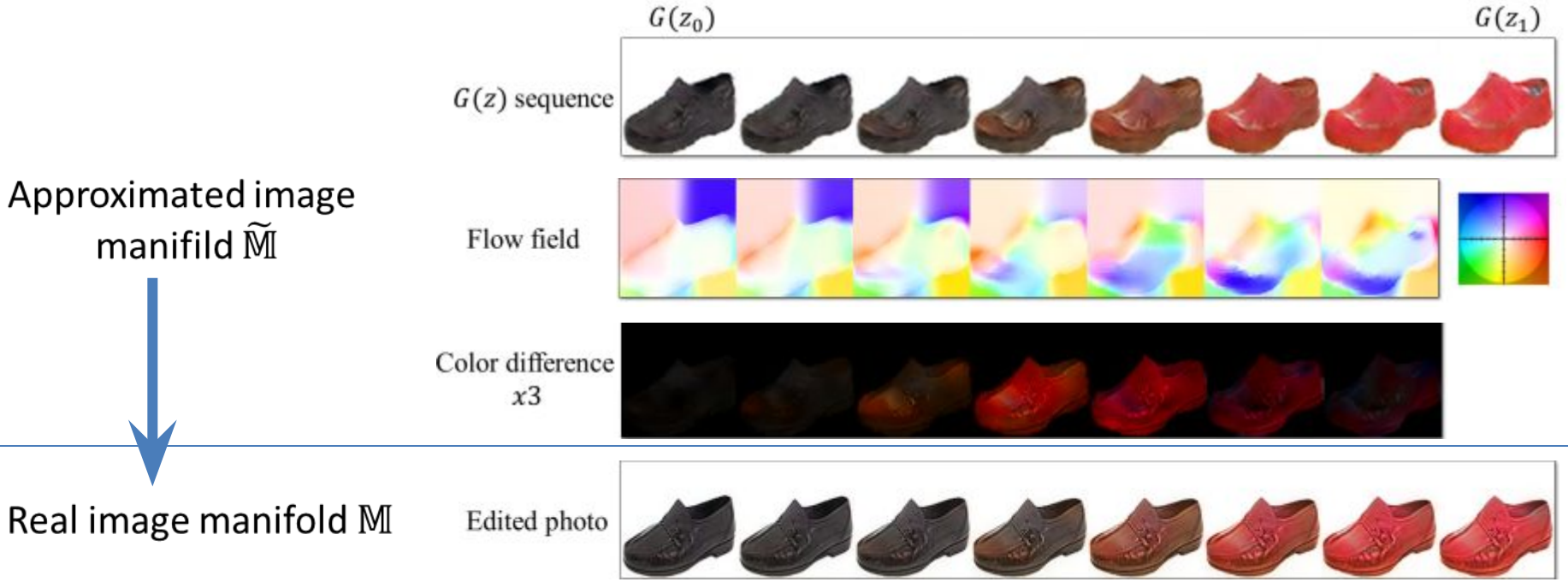
- Use the motion and color flow method

$$\iint \underbrace{\|I(x,y,t)-A\cdot I(x+u,y+v,t+1)\|^2}_{\text{data term}}+\underbrace{\sigma_s(\|\nabla u\|^2+\|\nabla v\|^2)}_{\text{spatial reg}}+\underbrace{\sigma_c\|\nabla A\|^2}_{\text{color reg}}dxdy$$

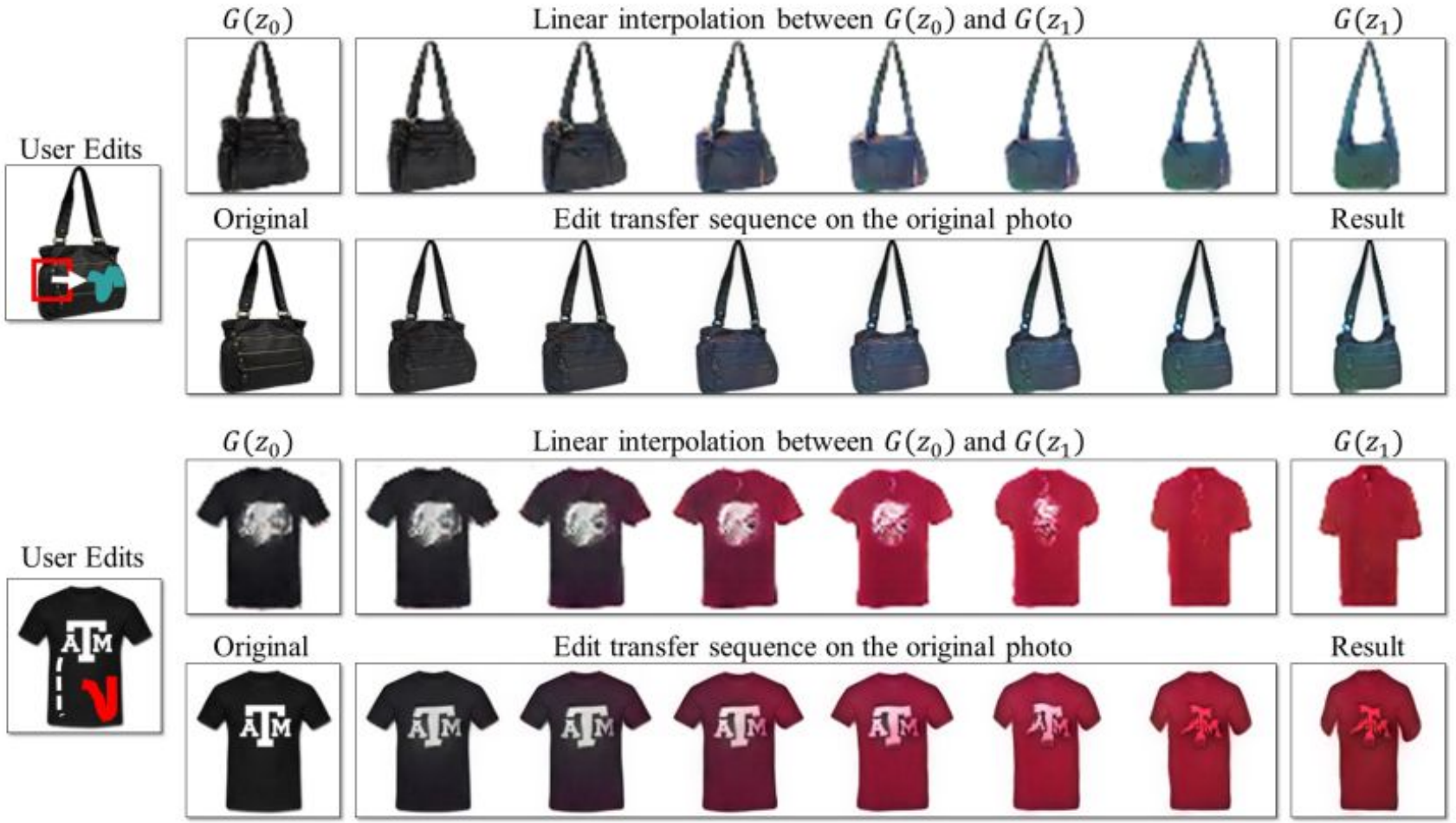
- $I(x,y,t)$ is the RGB value of pixel (x,y) in the generated image $G\left(\left(1-\frac{t}{N}\right)z_0+\frac{t}{N}z_1\right)$
- (u,v) is the flow vector w.r.t the change of t and A is a 3x4 color affine transformation matrix

Edit transfer

- The system estimates the color and shape changes between nearby frames.
- Apply those changes to the original photo and produce a transition sequence of images.



Edit transfer



Thank you