Image Synthesis with ConvNets: knowing what to ask the genie

Alexei (Alyosha) Efros
UC Berkeley
the brains behind the talk

Dr. Phillip Isola

Jun-Yan Zhu

Tinghui Zhou
Life is good in Deepland!

• “No need to design algorithms anymore”
• For any given problem, just:
  1. get some training data
  2. define objective function
  3. train network

  4. sell your startup for millions!
\[
\arg \max_{\mathcal{F}} L(\mathcal{F}(X), Y)
\]

- **Input**
- **Output**
- **Loss function**
- **Neural Network**
arg max \_\_F \_\_ L(F(X), Y) \_\_F \_\_

“What should I do?”

“How should I do it?”
Defining Objective Function

- Objective (Loss) functions tell the CNN what we want it to do
- Important to avoid King Midas’ mistake
- Usually designed by hand:
  - Not too hard for low-dimensional labels, e.g. classification
  - but much harder for high-dimensional, structured losses, e.g. computer graphics
Designing loss functions

“Minimize Euclidean distance between each predicted pixel color and its ground truth value.”
Designing loss functions

Color distribution cross-entropy loss with colorfulness enhancing term.

[Zhang, Isola, Efros, ECCV 2016]
Let’s look at another way to hack the loss function

This time, by *limiting* the space of outputs
View Synthesis by Appearance Flow
Tinghui Zhou, Shubham Tulsiani, Weilun Sun, Jitendra Malik, Alexei A. Efros
ECCV’16

(a) input
A Simple Baseline (pixel generation)

Brox et al
Issues

• Generating pixels to minimize L2 loss tends to lose details (as the CNN would collapse all the modes to produce a ‘mean car’)

• **Our solution**: formulate the problem as a *pixel selection* task such that the CNN is no longer allowed to predict the ‘mean’
Comparison on Real PASCAL Images

Input

Baseline prediction

Our prediction
Comparison on Real PASCAL Images

Input

Baseline prediction

Our prediction
Error Modes

Pixel Generation

Pixel Selection

Input  Prediction  Ground-truth

Our single-view failure modes
Learning to Leverage Multiple Views

Input Tuples

\[(\text{Car Image}, T_1) \rightarrow \text{Single View CNN} \rightarrow \text{Single-view prediction} \rightarrow \text{Selection Mask} \rightarrow \text{Final Prediction}\]

\[\vdots\]

\[(\text{Car Image}, T_N) \rightarrow \text{Single View CNN} \rightarrow \text{Single-view prediction} \rightarrow \text{Selection Mask} \rightarrow \text{Final Prediction}\]

Tied Weights
So far, all our losses were per-pixel…

But visual data has lots of spatial structure
Super-resolution: designing loss functions

“Minimize Euclidean distance between each predicted pixel color and its ground truth value.”

[Johnson, Alahi, Li, ECCV 2016]
Match the covariance statistics of deep feature activations (a la Gatys et al)
Structured Losses

Each pixel treated as independent

Looks at *joint* configuration of pixels
Up to now, for every application, we had to hack its own loss function

- How did we do it?
- “Hack until results look good” algorithm
  - a.k.a. Graduate Student Descent (GSD)
- Can we operationalize this “looks good”?
Generative adversarial nets (GANs) [Goodfellow et al, 2014]
Generative adversarial nets (GANs) [Goodfellow et al, 2014]

Generator net

Discriminator net

$x \sim G(z)$

$D(x) = \text{prob. } x \text{ is real}$
Generative Adversarial Networks

Adversarial process:
• simultaneously train two models
  – a generative model $G$ captures the data distribution.
  – discriminative model $D$ - tells whether a sample comes from the training data or not.

Optimal solution:
• $G$ recovers the data distribution.
• $D$ is 1/2 everywhere.

Goodfellow et al, “Generative Adversarial Networks”, 2014
Discriminative vs. generative models

- Generative model
  * (The artist)

- Discriminative model
  * (The lousy painter)

Slide from Luke Zettlemoyer, Carlos Guestrin, and Ben Taskar
Generative Adversarial Networks

[Goodfellow et al., 2014]

[Radford, Metz, Chintala, 2015]
Generative Adversarial Networks

Positive examples
Real or fake?

\[ D \]

\[ G \text{ tries to synthesize fake images that fool } D \]

\[ D \text{ tries to identify the fakes} \]

Negative examples
Real or fake pair?

\[ D \]

\[ G \]

\[ \text{noise } z \]
Limitations

• Very hard to train
  – Discriminator tends to win too easily
  – Co-adaptation, i.e. “orbiting”

• Not quite clear why we want to sample images at random

• What about starting with an real image?
Image Manipulation is hard

no “safety wheels”
Adding the “safety wheels”

A desired output:
- stay close to the input.
- satisfy user’s constraint.
- Lie on the natural image manifold
Generative Visual Manipulation on the Natural Image Manifold

Jun-Yan Zhu¹  Philipp Krähenbühl¹
Eli Shechtman²  Alexei A. Efros¹

UC Berkeley¹  Adobe²
ECCV 2016
Overview

original photo

projection on manifold

Project

Editing UI

different degree of image manipulation

transition between the original and edited projection

Edit Transfer
Overview

original photo

Project

projection on manifold

Editing UI

different degree of image manipulation

Edit Transfer

transition between the original and edited projection
Projecting an Image onto the Manifold

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

Optimization

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

Reconstruction loss $L$
Generative model $G(z)$
Projecting an Image onto the Manifold

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

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

Inverting Network $z = P(x)$
$$\theta^*_P = \arg\min_{\theta_P} \sum_{x^R_n} \mathcal{L}(G(P(x^R; \theta_P)), x^R)$$

Auto-encoder with a fixed decoder $G$
Projecting an Image onto the Manifold

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

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

Inverting Network $z = P(x)$
\[ \theta^*_P = \arg \min_{\theta_P} \sum_{x^R_n} \mathcal{L}(G(P(x^R; \theta_P)), x^R) \]

Hybrid Method
Use the network as initialization for the optimization problem
Overview

- original photo
- projection on manifold
- different degree of image manipulation
- transition between the original and edited projection
- editing UI
- edit transfer
Manipulating the Latent Vector

Objective:

\[ z^* = \arg \min_{z \in \mathbb{Z}} \left\{ \sum_{g} (L_g(G(z)) v_g) + \lambda_s \cdot \| z - z_0 \|^2_2 \right\}. \]

- **Guidance** \( v_g \)
- **Data Term** \( \sum_{g} (L_g(G(z)) v_g) \)
- **User Guidance Image**
- **Manifold Smoothness** \( \lambda_s \cdot \| z - z_0 \|^2_2 \)
Overview

- **Original Photo**: The initial image.
- **Projection on Manifold**: The initial projection of the image.
- **Editing UI**: The interface for editing the image.
- **Transition Between the Original and Edited Projection**: The process of transitioning between the original and edited images.
- **Different Degree of Image Manipulation**: The variety of manipulated images.
Edit Transfer

\[ G(z_0) \quad \text{Linear Interpolation in } z \text{ space} \quad G(z_1) \]

Input
Edit Transfer

\[ G(z_0) \]

Linear Interpolation in \( z \) space

\[ G(z_1) \]
Edit Transfer

**Motion** \((u, v)\)+ **Color** \((A_{3 \times 4})\): estimate per-pixel geometric and color variation

\[
\int \int \left[ \| I(x, y, t) - A \cdot I(x+u, y+v, t+1) \|^2 + \sigma_s (\| \nabla u \|^2 + \| \nabla v \|^2) + \sigma_c \| \nabla A \|^2 \right] dxdy
\]

- **Data term**
- **Spatial reg**
- **Color reg**

\(G(z_0)\)

**Input**

Linear Interpolation in \(z\) space

\(G(z_1)\)

**Result**
Image Manipulation Demo
Image Manipulation Demo
Interactive Image Generation
Interactive Image Generation
iGAN (aka. interactive GAN)

• Get the code: github.com/junyanz/iGAN
• Intelligent drawing tools via GAN.
• Debugging tools for understanding and visualizing deep generative networks.
• Work in progress: supporting more models (GAN, VAE, theano/tensorflow).
Image-to-Image Translation with Conditional Adversarial Nets

Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, Alexei A. Efros

https://phillipi.github.io/pix2pix/
Image-to-image translation

Object labeling
- [Long et al. 2015]

Edge Detection
- [Isola et al. 2014]

Season transfer
- [Laffont et al. 2014]

Colorization
- [Zhang et al. 2016]
Conditional GANs
Conditional GANs

Positive examples

Real or fake pair?

\[ \text{D} \]

\[ \text{G} \]

\text{G} \text{ tries to synthesize fake images that fool } \text{D}

\text{D} \text{ tries to identify the fakes}

Negative examples

Real or fake pair?

\[ \text{D} \]

\[ \text{G} \]

\[ \text{D} \]

\[ \text{G} \]
Conditional GANs

G’s perspective: D is a loss function.

Rather than being hand-designed, it is *learned*. 
Generator architecture

Encoder-decoder

U-Net
Effect of the generator architecture

L1
L1+cGAN

Encoder-decoder
U-Net
### Effect of the objective

![Image showing the effect of different objectives on image quality](image_url)

<table>
<thead>
<tr>
<th>Loss</th>
<th>Per-pixel acc.</th>
<th>Per-class acc.</th>
<th>Class IOU</th>
</tr>
</thead>
<tbody>
<tr>
<td>L1</td>
<td>0.44</td>
<td>0.14</td>
<td>0.10</td>
</tr>
<tr>
<td>GAN</td>
<td>0.22</td>
<td>0.05</td>
<td>0.01</td>
</tr>
<tr>
<td>cGAN</td>
<td>0.61</td>
<td>0.21</td>
<td>0.16</td>
</tr>
<tr>
<td>L1+GAN</td>
<td><strong>0.64</strong></td>
<td>0.19</td>
<td>0.15</td>
</tr>
<tr>
<td>L1+cGAN</td>
<td>0.63</td>
<td><strong>0.21</strong></td>
<td><strong>0.16</strong></td>
</tr>
<tr>
<td>Ground truth</td>
<td>0.80</td>
<td>0.26</td>
<td>0.21</td>
</tr>
</tbody>
</table>
Discriminator architectures

PixelGAN

PatchGAN

ImageGAN

[Slide from Victor Garcia]
Effect of the discriminator patch size

1x1  

16x16  

70x70  

256x256
Effect of the discriminator patch size

<table>
<thead>
<tr>
<th>Discriminator receptive field</th>
<th>Per-pixel acc.</th>
<th>Per-class acc.</th>
<th>Class IOU</th>
</tr>
</thead>
<tbody>
<tr>
<td>1x1</td>
<td>0.44</td>
<td>0.14</td>
<td>0.10</td>
</tr>
<tr>
<td>16x16</td>
<td>0.62</td>
<td>0.20</td>
<td>0.16</td>
</tr>
<tr>
<td>70x70</td>
<td>0.63</td>
<td>0.21</td>
<td>0.16</td>
</tr>
<tr>
<td>256x256</td>
<td>0.47</td>
<td>0.18</td>
<td>0.13</td>
</tr>
</tbody>
</table>
A bunch of results
L2 regression
Zhang et al., 2016
cGAN
Groundtruth
Colorizing sketches
Failure cases
Code available: https://github.com/phillipi/pix2pix

[Arthur Juliani]

[Opendotlab]

[twitter @thorikawa]

[twitter @hiromu1996]
kaihu chen @kaihuchen

[twitter @hiromu1996]

Brannon Dorsey @brannondorsey

Bertrand Gondouin @bgondouin
Twitter, the new way to do research!

https://twitter.com/search?q=pix2pix&src=typd
Conclusions

• ConvNets have moved the difficulty from algorithm design to loss function design

• Everyone is well-advised to learn from King Midas