**Artist-Critic Co-Evolution**

Critic is rewarded for distinguishing real images from those generated by the artist.

Artist is rewarded for fooling the critic into thinking that generated images are real.
Blind Watchmaker (Dawkins, 1986)

- the Human is presented with 15 images
- the chosen image(s) are used to breed the next generation

Interactive Evolution (Sims, 1991)

- Artist = Genetic Program (GP)
  - used as function to compute R,G,B values for each pixel x,y
- Critic = Human

Blind Watchmaker Biomorphs

PicBreeder Examples
PicBreeder (Secretan, 2011)

- Artist = Convolutional Pattern Producing Neural Network (CPPN)
- Interactive Web site (picbreeder.org) where you can choose existing individual and use it for further breeding
- Interactive Evolution is cool, but it may require a lot of work from the Human – Can the Human be replaced by an automated Critic?

Computational Aesthetics (Machado, 2008)

- Generator = Genetic Program
- Critic = 2-layer NN, using statistical features of image

Recall: Variational Autoencoder

![variational autoencoder faces](image)

Variational Autoencoder Faces

![variational autoencoder faces](image)
Generative Adversarial Networks

\[ \tilde{z} \xrightarrow{G} \tilde{x} \xrightarrow{D} x \]

Discriminator tries to assign a high number to real images \((x)\) and a low number to “fake” images \((\tilde{x})\) produced by the Generator.

Generative Adversarial Networks

Generator (Artist) \(G_\theta\) and Discriminator (Critic) \(D_\psi\) are both Deep Convolutional Neural Networks.

Generator \(G_\theta: z \mapsto x\), with parameters \(\theta\), generates an image \(x\) from latent variables \(z\) (sampled from a standard Normal distribution).

Discriminator \(D_\psi: x \mapsto D_\psi(x) \in (0, 1)\), with parameters \(\psi\), takes an image \(x\) and estimates the probability of the image being real.

Generator and Discriminator play a 2-player zero-sum game to compute:

\[
\min_{\theta} \max_{\psi} \left( E_{x \sim p_{\text{data}}} \left[ \log D_\psi(x) \right] + E_{z \sim p_{\text{model}}} \left[ \log (1 - D_\psi(G_\theta(z))) \right] \right)
\]

Discriminator tries to maximize the bracketed expression, Generator tries to minimize it.

GAN Generated Images

Alternate between:

Gradient ascent on Discriminator:

\[
\max_{\psi} \left( E_{x \sim p_{\text{data}}} \left[ \log D_\psi(x) \right] + E_{z \sim p_{\text{model}}} \left[ \log (1 - D_\psi(G_\theta(z))) \right] \right)
\]

Gradient descent on Generator, using:

\[
\min_{\theta} E_{z \sim p_{\text{model}}} \left[ \log (1 - D_\psi(G_\theta(z))) \right]
\]
Generative Adversarial Networks

Alternate between:

Gradient ascent on Discriminator:

\[
\max_{\psi} \left( \mathbb{E}_{x \sim p_{data}} \left[ \log D_{\psi}(x) \right] + \mathbb{E}_{z \sim p_{model}} \left[ \log (1 - D_{\psi}(G_{\theta}(z))) \right] \right)
\]

Gradient descent on Generator, using:

\[
\min_{\theta} \mathbb{E}_{z \sim p_{model}} \left[ \log (1 - D_{\psi}(G_{\theta}(z))) \right]
\]

This formula puts too much emphasis on images that are correctly classified. Better to do gradient ascent on Generator, using:

\[
\max_{\theta} \mathbb{E}_{z \sim p_{model}} \left[ \log (D_{\psi}(G_{\theta}(z))) \right]
\]

This puts more emphasis on the images that are wrongly classified.

GAN Convolutional Architectures

■ normalize images to between $-1$ and $+1$
■ replace pooling layers with:
  ▶ strided convolutions (Discriminator)
  ▶ fractional-strided convolutions (Generator)
■ use BatchNorm in both Generator and Discriminator
■ remove fully connected hidden layers for deeper architectures
■ use tanh at output layer of Generator, ReLU activation in all other layers
■ use LeakyReLU activation for all layers of Discriminator

GAN properties:

■ one network aims to produces the full range of images $x$, with different values for the latent variables $z$
■ differentials are backpropagated through the Discriminator network and into the Generator network
■ compared to previous approaches, the images produced are much more realistic!
Generator Architecture

Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks (Radford et al., 2016)

GAN Image Vector Arithmetic

Oscillation and Mode Collapse

- Like any coevolution, GANs can sometimes oscillate or get stuck in a mediocre stable state.
- **oscillation**: GAN trains for a long time, generating a variety of images, but quality fails to improve (compare IPD)
- **mode collapse**: Generator produces only a small subset of the desired range of images, or converges to a single image (with minor variations)

Methods for avoiding mode collapse:
- Conditioning Augmentation
- Minibatch Features (Fitness Sharing)
- Unrolled GANs
The GAN Zoo

- Contex-Encoder for Image Inpainting
- Texture Synthesis with Patch-based GAN
- Conditional GAN
- Text-to-Image Synthesis
- StackGAN
- Patch-based Discriminator
- S²-GAN
- Style-GAN
- Plug-and-Play Generative Networks

Attribute Swapping with Autoencoders

<table>
<thead>
<tr>
<th>Class</th>
<th>Reconstruction</th>
<th>Combination</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>H(ident)</td>
<td>H(ident)</td>
</tr>
<tr>
<td></td>
<td>H(attrib)</td>
<td>H(attrib)</td>
</tr>
<tr>
<td>Input</td>
<td></td>
<td>I(ident)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>I(attrib)</td>
</tr>
</tbody>
</table>

Attribute Swapping

Deployment

Combination Reconstruction

H(ident) H(attrib)

Input

Training

Computer-Generated Art (pickartso.com)
Bi-GAN

\[ \tilde{z} \xrightarrow{G} \tilde{x} \xrightarrow{D} \]
\[ z \xleftarrow{E} x \]

Discriminator tries to assign a high number to the combination \((z, x)\) and a low number to the combination \((\tilde{z}, \tilde{x})\)

References


http://www.iangoodfellow.com/slides/2016-12-04-NIPS.pdf

https://arxiv.org/abs/1612.00005

https://pickartso.com