Generative adversarial networks

BigGAN (2018)
Outline

• Generative tasks
• Original GAN formulations
  • NSGAN
  • DCGAN
• Other popular formulations
  • WGAN, WGAN-GP
  • LSGAN
• State-of-the-art architectures
  • Progressive GAN, StyleGAN
• Evaluating GANs
• Visualizing and controlling GANs
Generative tasks

- Generation (from scratch): learn to sample from the distribution represented by the training set
  - *Unsupervised learning* task
Generative tasks

- Generation conditioned on class label

Figure source
Generative tasks

- Generation conditioned on image (image-to-image translation)

Designing a network for generative tasks

1. We need an architecture that can generate an image
   - Recall upsampling architectures for dense prediction
Designing a network for generative tasks

1. We need an architecture that can generate an image
   • Recall upsampling architectures for dense prediction

Image-to-image translation
Designing a network for generative tasks

1. We need an architecture that can generate an image
   • Recall upsampling architectures for dense prediction
2. We need to design the right loss function
Learning to sample

Training data \(x \sim p_{\text{data}}\)

Generated samples \(x \sim p_{\text{model}}\)

We want to learn \(p_{\text{model}}\) that matches \(p_{\text{data}}\)

Adapted from Stanford CS231n
Generative adversarial networks

- Train two networks with opposing objectives:
  - **Generator**: learns to generate samples
  - **Discriminator**: learns to distinguish between generated and real samples

Figure adapted from F. Fleuret

GAN objective

- The discriminator $D(x)$ should output the probability that the sample $x$ is real
  - That is, we want $D(x)$ to be close to 1 for real data and close to 0 for fake
- Expected conditional log likelihood for real and generated data:
  \[
  \mathbb{E}_{x \sim p_{data}} \log D(x) + \mathbb{E}_{x \sim p_{gen}} \log(1 - D(x))
  \]
  \[
  = \mathbb{E}_{x \sim p_{data}} \log D(x) + \mathbb{E}_{z \sim p} \log(1 - D(G(z)))
  \]
  We seed the generator with noise $z$ drawn from a simple distribution $p$ (Gaussian or uniform)
GAN objective

\[ V(G, D) = \mathbb{E}_{x \sim p_{data}} \log D(x) + \mathbb{E}_{z \sim p} \log(1 - D(G(z))) \]

- The discriminator wants to correctly distinguish real and fake samples:
  \[ D^* = \arg \max_D V(G, D) \]
- The generator wants to fool the discriminator:
  \[ G^* = \arg \min_G V(G, D) \]
- Train the generator and discriminator jointly in a minimax game
GAN objective: Theoretical properties

\[ V(G, D) = \mathbb{E}_{x \sim p_{data}} \log D(x) + \mathbb{E}_{z \sim p} \log(1 - D(G(z))) \]

- Assuming unlimited capacity for generator and discriminator and unlimited training data:
  - The objective \( \min_G \max_D V(G, D) \) is equivalent to *Jensen-Shannon divergence* between \( p_{data} \) and \( p_{gen} \) and global optimum (*Nash equilibrium*) is given by \( p_{data} = p_{gen} \)
  - If at each step, \( D \) is allowed to reach its optimum given \( G \), and \( G \) is updated to decrease \( V(G, D) \), then \( p_{gen} \) with eventually converge to \( p_{data} \)
GAN training

\[ V(G, D) = \mathbb{E}_{x \sim p_{data}} \log D(x) + \mathbb{E}_{z \sim \rho} \log(1 - D(G(z))) \]

- Alternate between
  - **Gradient ascent** on discriminator:
    \[ D^* = \arg \max_D V(G, D) \]
  - **Gradient descent** on generator (minimize log-probability of discriminator being right):
    \[ G^* = \arg \min_G V(G, D) = \arg \min_G \mathbb{E}_{z \sim \rho} \log(1 - D(G(z))) \]
  - In practice, do **gradient ascent** on generator (maximize log-probability of discriminator being wrong):
    \[ G^* = \arg \max_G \mathbb{E}_{z \sim \rho} \log(D(G(z))) \]
Non-saturating GAN loss (NSGAN)

\[
\min_{w_G} \mathbb{E}_{z \sim p} \log(1 - D(G(z))) \quad \text{vs.} \quad \max_{w_D} \mathbb{E}_{z \sim p} \log(D(G(z)))
\]

Minimize log-probability of discriminator being right
Maximize log-probability of discriminator being wrong
Non-saturating GAN loss (NSGAN)

$$\min_{w_G} \mathbb{E}_{z \sim p} \log(1 - D(G(z))) \quad \text{vs.} \quad \max_{w_G} \mathbb{E}_{z \sim p} \log(D(G(z)))$$

- Large gradients for low-quality samples
- Small gradients for high-quality samples
- Want to learn from confidently rejected sample but gradients here are small
- These samples already fool the discriminator so we don’t need large gradients here

Figure source
NSGAN training algorithm

• Update discriminator:
  • Repeat for $k$ steps:
    • Sample mini-batch of noise samples $z_1, ..., z_m$ and mini-batch of real samples $x_1, ..., x_m$
    • Update parameters of $D$ by stochastic gradient ascent on
      $$\frac{1}{m} \sum_m [\log D(x_m) + \log(1 - D(G(z_m)))]$$

• Update generator:
  • Sample mini-batch of noise samples $z_1, ..., z_m$
  • Update parameters of $G$ by stochastic gradient ascent on
    $$\frac{1}{m} \sum_m \log D(G(z_m))$$

• Repeat until happy with results
GAN: Conceptual picture

- Update discriminator: push $D(x_{data})$ close to 1 and $D(G(z))$ close to 0
  - The generator is a “black box” to the discriminator
GAN: Conceptual picture

- Update generator: increase $D(G(z))$
  - Requires back-propagating through the composed generator-discriminator network (i.e., the discriminator cannot be a black box)
  - The generator is exposed to real data only via the output of the discriminator (and its gradients)
GAN: Conceptual picture

- Test time – the discriminator is discarded
Original GAN results

MNIST digits

Toronto Face Dataset

Nearest real image for sample to the left

Original GAN results

DCGAN

• Early, influential convolutional architecture for generator

DCGAN

• Early, influential convolutional architecture for generator
• Discriminator architecture:
  • Don’t use pooling, only strided convolutions
  • Use Leaky ReLU activations (sparse gradients cause problems for training)
  • Use only one FC layer before the softmax output
  • Use batch normalization after most layers (in the generator also)

DCGAN results

Generated bedrooms after one epoch
DCGAN results

Generated bedrooms after five epochs
DCGAN results

More bedrooms

Notice repetition artifacts (analysis)

Source: F. Fleuret
DCGAN results

Interpolation between different points in the z space
DCGAN results

• Vector arithmetic in the z space
DCGAN results

- Vector arithmetic in the z space
DCGAN results

- Pose transformation by adding a “turn” vector
Outline

• Generative tasks
• Original GAN formulations
  • NSGAN
  • DCGAN
• Other popular formulations
  • WGAN, WGAN-GP
  • LSGAN
Problems with GAN training

• Stability
  • Parameters can oscillate or diverge, generator loss does not correlate with sample quality
  • Behavior very sensitive to hyperparameter selection
Problems with GAN training

- Mode collapse
  - Generator ends up modeling only a small subset of the training data
Some popular GAN flavors

• WGAN and improved WGAN (WGANGP)
• LSGAN
Wasserstein GAN (WGAN)

• Motivated by Wasserstein or Earth mover’s distance, which is an alternative to JS divergence for comparing distributions
  • In practice, use linear activation instead of sigmoid in the discriminator and drop the logs from the objective:
    \[
    \min_G \max_D \left[ \mathbb{E}_{x \sim p_{\text{data}}} D(x) - \mathbb{E}_{z \sim p} D(G(z)) \right]
    \]
  • Due to theoretical considerations, important to ensure smoothness of discriminator
  • This paper’s suggested method is clipping weights to fixed range \([-c, c]\)

M. Arjovsky, S. Chintala, L. Bottou, Wasserstein generative adversarial networks, ICML 2017
Wasserstein GAN (WGAN)

- Benefits (claimed)
  - Better gradients, more stable training

M. Arjovsky, S. Chintala, L. Bottou, *Wasserstein generative adversarial networks*, ICML 2017
Wasserstein GAN (WGAN)

- Benefits (claimed)
  - Better gradients, more stable training
  - Objective function value is more meaningfully related to quality of generator output

M. Arjovsky, S. Chintala, L. Bottou, *Wasserstein generative adversarial networks*, ICML 2017
Improved Wasserstein GAN (WGAN-GP)

• Weight clipping leads to problems with discriminator training
• Improved Wasserstein discriminator loss:

\[ \mathbb{E}_{\tilde{x} \sim p_{\text{gen}}} D(\tilde{x}) - \mathbb{E}_{x \sim p_{\text{real}}} D(x) \]

\[ + \lambda \mathbb{E}_{\tilde{x} \sim p_{\tilde{x}}} \left[ (\| \nabla_{\tilde{x}} D(\tilde{x}) \|_2 - 1)^2 \right] \]

Unit norm gradient penalty on points \( \tilde{x} \) obtained by interpolating real and generated samples

# Improved Wasserstein GAN: Results

<table>
<thead>
<tr>
<th>DCGAN</th>
<th>LSGAN</th>
<th>WGAN (clipping)</th>
<th>WGAN-GP (ours)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Baseline</strong> (G: DCGAN, D: DCGAN)</td>
<td>![Image]</td>
<td>![Image]</td>
<td>![Image]</td>
</tr>
<tr>
<td>G: No BN and a constant number of filters, D: DCGAN</td>
<td>![Image]</td>
<td>![Image]</td>
<td>![Image]</td>
</tr>
<tr>
<td>G: 4-layer 512-dim ReLU MLP, D: DCGAN</td>
<td>![Image]</td>
<td>![Image]</td>
<td>![Image]</td>
</tr>
<tr>
<td>No normalization in either G or D</td>
<td>![Image]</td>
<td>![Image]</td>
<td>![Image]</td>
</tr>
<tr>
<td>Gated multiplicative nonlinearities everywhere in G and D</td>
<td>![Image]</td>
<td>![Image]</td>
<td>![Image]</td>
</tr>
<tr>
<td>tanh nonlinearities everywhere in G and D</td>
<td>![Image]</td>
<td>![Image]</td>
<td>![Image]</td>
</tr>
<tr>
<td>101-layer ResNet G and D</td>
<td>![Image]</td>
<td>![Image]</td>
<td>![Image]</td>
</tr>
</tbody>
</table>

Least Squares GAN (LSGAN)

• Use least squares cost for generator and discriminator
  • Equivalent to minimizing Pearson $\chi^2$ divergence

\[
D^* = \arg \min_D \left[ \mathbb{E}_{x \sim p_{\text{data}}} (D(x) - 1)^2 + \mathbb{E}_{z \sim p} (D(G(z)))^2 \right]
\]

Push discrim. response on real data close to 1

Push response on generated data close to 0

\[
G^* = \arg \min_G \mathbb{E}_{z \sim p} (D(G(z)) - 1)^2
\]

Push response on generated data close to 1

Least Squares GAN (LSGAN)

• Benefits (claimed)
  • Higher-quality images

Least Squares GAN (LSGAN)

• Benefits (claimed)
  • Higher-quality images
  • More stable and resistant to mode collapse

Are GANs created equal?

- From the abstract:

  “We find that most models can reach similar scores with enough hyperparameter optimization and random restarts. This suggests that improvements can arise from a higher computational budget and tuning more than fundamental algorithmic changes … We did not find evidence that any of the tested algorithms consistently outperforms the non-saturating GAN introduced in Goodfellow et al. (2014)”

M. Lucic, K. Kurach, M. Michalski, O. Bousquet, S. Gelly, Are GANs created equal? A large-scale study, NIPS 2018
Outline

• Generative tasks
• Original GAN formulations
  • NSGAN
  • DCGAN
• Other popular formulations
  • WGAN, WGAN-GP
  • LSGAN
• State-of-the-art architectures
  • Progressive GAN, StyleGAN
Recent progress in GANs

4.5 years of GAN progress on face generation.

arxiv.org/abs/1406.2661 arxiv.org/abs/1511.06434
arxiv.org/abs/1606.07536 arxiv.org/abs/1710.10196
arxiv.org/abs/1812.04948
Recent progress in GANs

**EBGAN** (2017)

**BigGAN** (2018)
Progressive GANs

Realistic face images up to 1024 x 1024 resolution

Progressive GANs

- Key idea: train lower-resolution models, gradually add layers corresponding to higher-resolution outputs

Progressive GANs

- Key idea: train lower-resolution models, gradually add layers corresponding to higher-resolution outputs

Transition from 16x16 to 32x32 images

Progressive GANs: Implementation details

- Loss: WGAN-GP loss (preferred) or LSGAN
- Architectures:
  - Nearest neighbor upsampling (2x2 replication) followed by regular convolutions instead of transposed conv layers
  - Average pooling instead of striding for downsampling in discriminator
  - Leaky ReLUs used in discriminator and generator
  - Per-pixel response normalization in generator: rescale feature vector in each pixel to unit length after each conv layer
- Use of minibatch standard deviation in discriminator (append to feature map)
- Exponential moving average of generator weights for display

Progressive GANs: Results

256 x 256 results for LSUN categories
StyleGAN

- Built on top of Progressive GAN
- Start with learned constant (instead of noise vector)
- Use a mapping network to produce a *style code* $w$ using learned affine transformations $A$
- Use *adaptive instance normalization* (AdaIN): scale and bias each feature map using learned style values
- Add noise after each convolution and before nonlinearity (enables stochastic detail)

StyleGAN: Results

“Two sets of images were generated from their respective latent codes (sources A and B); the rest of the images were generated by copying a specified subset of styles from source B and taking the rest from source A.”
Mixing styles
Mixing styles
StyleGAN: Bedrooms

StyleGAN: Cars

StyleGAN2

- Change normalization, remove progressive growing to address StyleGAN artifacts

Figure 1. Instance normalization causes water droplet-like artifacts in StyleGAN images. These are not always obvious in the generated images, but if we look at the activations inside the generator network, the problem is always there, in all feature maps starting from the 64x64 resolution. It is a systemic problem that plagues all StyleGAN images.

Figure 6. Progressive growing leads to “phase” artifacts. In this example the teeth do not follow the pose but stay aligned to the camera, as indicated by the blue line.

T. Karras et al. Analyzing and Improving the Image Quality of StyleGAN. CVPR 2020
Reminder: Assignment 3 due next Tuesday!
Outline

• Generative tasks
• Original GAN formulations
  • NSGAN
  • DCGAN
• Other popular formulations
  • WGAN, WGAN-GP
  • LSGAN
• State-of-the-art architectures
  • Progressive GAN, StyleGAN
• Evaluating GANs
How to evaluate GANs?

• Showing pictures of samples is not enough, especially for simpler datasets like MNIST, CIFAR, faces, bedrooms, etc.
• We cannot directly compute the likelihoods of high-dimensional samples (real or generated), or compare their distributions
• Many GAN approaches claim mainly to improve stability, which is hard to evaluate
GAN evaluation: Human studies

• Example: Turing test

GAN evaluation: Inception score (IS)

- **Key idea:** generators should produce images with a variety of recognizable object classes

- **Defined as**
  \[
  IS(G) = \exp \left[ \mathbb{E}_{x \sim G} KL(P(y|x) \parallel P(y)) \right]
  \]

  where \( P(y|x) \) is the posterior label distribution returned by an image classifier (e.g., InceptionNet) for sample \( x \)
- If \( x \) contains a recognizable object, entropy of \( P(y|x) \) should be low
- If generator generates images of diverse objects, the marginal distribution \( P(y) \) should have high entropy

---

GAN evaluation: Inception score (IS)

- Disadvantages
  - A GAN that simply memorizes the training data (overfitting) or outputs a single image per class (mode dropping) could still score well
  - Is sensitive to network weights, not necessarily valid for generative models not trained on ImageNet, can be gamed (Barratt & Sharma 2018)

Figure 1. Sample of generated images achieving an Inception Score of 900.15. The maximum achievable Inception Score is 1000, and the highest achieved in the literature is on the order of 10.
GAN evaluation: Fréchet Inception Distance (FID)

- Key idea: fit simple distributions (Gaussians) to statistics of feature activations for real and generated data; estimate divergence parametrically
  - Pass generated samples through a network (InceptionNet), compute activations for a chosen layer
  - Estimate multivariate mean and covariance of activations, compute Fréchet distance to those of real data
- Advantages: correlated with visual quality of samples and human judgment, can detect mode dropping (unlike IS)
- Disadvantage: cannot detect overfitting (like IS)

M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, S. Hochreiter, *GANs trained by a two time-scale update rule converge to a local Nash equilibrium*, NIPS 2017
Outline

- Generative tasks
- Original GAN formulations
  - NSGAN
  - DCGAN
- Other popular formulations
  - WGAN, WGAN-GP
  - LSGAN
- State-of-the-art architectures
  - Progressive GAN, StyleGAN
- Evaluating GANs
- Visualizing and controlling GANs
GAN Dissection

- Recall: network dissection

GAN Dissection

D. Bau et al. GAN Dissection: Visualizing and understanding generative adversarial networks. ICLR 2019
GAN Dissection

- Dissection:

D. Bau et al. GAN Dissection: Visualizing and understanding generative adversarial networks. ICLR 2019
GAN Dissection

- Interpreting units at different levels of a progressive GAN (trained on “bedroom”):

D. Bau et al. GAN Dissection: Visualizing and understanding generative adversarial networks. ICLR 2019
GAN Dissection

- Intervention:
GANPaint demo

https://ganpaint.io/demo/?project=church

D. Bau et al. Semantic Photo Manipulation with a Generative Image Prior. SIGGRAPH 2019
Seeing what a GAN cannot generate

- Compare semantic segmentations of real and generated images, see which classes get dropped by the GAN

D. Bau et al. Seeing what a GAN cannot generate. ICCV 2019
Seeing what a GAN cannot generate

- Given real images, try to find “closest” generated images and see what gets omitted

D. Bau et al. Seeing what a GAN cannot generate. ICCV 2019
Seeing what a GAN cannot generate

• Layer inversion method:

D. Bau et al. Seeing what a GAN cannot generate. ICCV 2019
Seeing what a GAN cannot generate

- Reconstruction results for “bedroom” GAN

<table>
<thead>
<tr>
<th></th>
<th>photograph</th>
<th>generated</th>
<th>photograph</th>
<th>generated</th>
<th>photograph</th>
<th>generated</th>
<th>photograph</th>
<th>generated</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>LSUN bedrooms data</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

D. Bau et al. [Seeing what a GAN cannot generate](https://arxiv.org/abs/1906.02976), ICCV 2019
Traversing the GAN latent space

- Try to find simple “walks” in the latent space of GANs to achieve various meaningful transformations to explore structure of that space and test GANs’ ability to interpolate between training samples

A. Jahanian, L. Chai, and P. Isola. On the "steerability" of generative adversarial networks. ICLR 2020
Traversing the GAN latent space

- Goal: learn a set of directions inducing “disentangled” image transformations that are easy to distinguish from each other

A.Voynov and A. Babenko. *Unsupervised Discovery of Interpretable Directions in the GAN Latent Space*. ICML 2020
GAN editing

- Supervised training to find latent space directions corresponding to pose, smile, age, gender, eyeglasses

GANs for representation learning

- Bidirectional GAN (BiGAN): simultaneously train generator and encoder (mapping from images to $z$ vectors or approximate inverse of the generator), show that the encoder creates a latent representation useful for other tasks

GANs for representation learning: BigBiGAN

• Train BiGAN with BigGAN architecture on ImageNet
• Show that bidirectional framework improves image generation
• Encoder representation gives results comparable to state-of-the-art self-supervised models on ImageNet classification

J. Donahue, K. Simonyan, Large Scale Adversarial Representation Learning, NeurIPS 2019
GANs for representation learning: BigBiGAN

Real images $x$

Reconstructions $G(E(x))$

J. Donahue, K. Simonyan, Large Scale Adversarial Representation Learning, NeurIPS 2019