DEEP LEARNING

Lecture 9: Generative Adversarial Networks

Dr. Yang Lu
Department of Computer Science
luyang@xmu.edu.cn
GAN Applications

- Image Translation

Input | Generated outputs (Female) | Generated outputs (Male)
---|---|---
Celeb-HQ
AFHQ

GAN Applications

Image Translation

GAN Applications

Scene Generation

GAN Applications

Facial Attribute Manipulation

GAN Applications

- Facial Attribute Manipulation

GAN Applications

- Gaze Correction

GAN Applications

- Image Animation

GAN Applications

- Image Inpainting

GAN Applications

Image Inpainting

GAN Applications

- Image blending

GAN Applications

- Image Super-Resolution

bicubic
(21.59dB/0.6423)

SRResNet
(23.53dB/0.7832)

SRGAN
(21.15dB/0.6868)

original

GAN Applications

- Shadow Detection and Removal

GAN Applications

- Makeup

GAN Applications

- Facial Landmark Detection

GAN Applications

- Person Re-identification

GAN Applications

Music Generation

![Musical notation images]

Generated music samples: [https://soundcloud.com/vgtsv6jf5fwq/sets/midinet-samples](https://soundcloud.com/vgtsv6jf5fwq/sets/midinet-samples)
In machine learning, two main approaches are called the **generative approaches** and the **discriminative approaches**.

Given an observable variable $X$ and a target variable $Y$:

- A generative model is a statistical model of the data distribution $P(X)$ or the joint probability distribution on $X \times Y$: $P(X, Y)$.
- A discriminative model is a model of the conditional distribution of $Y$ given $X$: $P(Y|X = x)$.
Most supervised learning methods fall into discriminative approaches.

- Given data: \((x, y)\), \(x\) is data, \(y\) is label.
- Goal: Learn a function to map \(x \rightarrow y\), namely posterior probability \(P(Y|X = x)\).
- Examples: Classification, regression, object detection, face recognition, sentiment classification, etc.
Generative Approaches

- Given training data, generate new samples from same distribution.
- Objectives:
  1. Learn $p_{model}(x)$ that approximates $p_{data}(x)$.
  2. Sample a new $x$ from $p_{model}(x)$.

Outlines

- GAN
- DCGAN
- CGAN
- WGAN
- SAGAN
- pix2pix and CycleGAN
- SRGAN
GAN
GAN was proposed by Ian Goodfellow in 2014.

Yann LeCun described GANs as “the most interesting idea in the last 10 years in machine learning”.

Ian presented and explained his paper in NIPS 2016 with a 2-hour presentation.
GAN: How to Do

- GAN is composed by a **generator** and a **discriminator**. They are both neural networks.
  - Generator network: try to fool the discriminator by generating real-looking images.
  - Discriminator network: try to distinguish between real and fake images.

Image source: https://sthalles.github.io/intro-to-gans/
GAN: How to Do

- Generator and discriminator tells each other where it was wrong.
  - Generator tells discriminator how I detect you.
  - Discriminator tells generator how I fool you.

Image source: https://sthalles.github.io/intro-to-gans/
GAN: How to Learn

- Given a prior on input noise variables $p_z(z)$, generator $G_{\theta_g}(z)$ take $z$ as input and map it into data space.
- Discriminator $D_{\theta_d}(x)$ takes the data $x$ as input and output probability that $x$ came from the real data rather than generated data $G_{\theta_g}(z)$.
- $D_{\theta_d}$ and $G_{\theta_g}$ have different goals (1 for real, 0 for fake):
  - Generator wants: $D_{\theta_d}\left(G_{\theta_g}(z)\right) \rightarrow 1$.
  - Discriminator wants: $D_{\theta_d}(x) \rightarrow 1$, $D_{\theta_d}\left(G_{\theta_g}(z)\right) \rightarrow 0$. 
GAN: How to Learn

- By maximizing the log-likelihood, the overall objective is to simultaneously train over all $x$ with random generated $z$:
  - train $G_{\theta_g}$ to minimize $\log \left( 1 - D_{\theta_d} \left( G_{\theta_g}(z) \right) \right)$;
  - train $D_{\theta_d}$ to maximize $\log D_{\theta_d}(x)$ and $\log \left( 1 - D_{\theta_d} \left( G_{\theta_g}(z) \right) \right)$.

- In other words, $D_{\theta_d}$ and $G_{\theta_g}$ play the following two-player minimax game:

$$\min_{\theta_g} \max_{\theta_d} \left[ \mathbb{E}_{x \sim p_{\text{data}}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p_z(z)} \log \left( 1 - D_{\theta_d} \left( G_{\theta_g}(z) \right) \right) \right]$$

Discriminator output for real data $x$  
Discriminator output for generated fake data $G_{\theta_g}(z)$. 
GAN: How to Learn

Objective:
\[
\min_{\theta_g} \max_{\theta_d} \left[ \mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p_z(z)} \log \left( 1 - D_{\theta_d}(G_{\theta_g}(z)) \right) \right]
\]

Alternate between:
- Gradient ascent on discriminator:
  \[
  \max_{\theta_d} \left[ \mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p_z(z)} \log \left( 1 - D_{\theta_d}(G_{\theta_g}(z)) \right) \right]
  \]
- Gradient descent on generator:
  \[
  \min_{\theta_g} \left[ \mathbb{E}_{z \sim p_z(z)} \log \left( 1 - D_{\theta_d}(G_{\theta_g}(z)) \right) \right]
  \]
GAN: How to Learn

- After several steps of training, if $G$ and $D$ have enough capacity, they will reach a point at which both cannot improve because $p_g = p_{data}$.
  - Generator can generate real image.
  - Discriminator is unable to differentiate between the two distributions, i.e. $D_{\theta_d}(x) = 1/2$. 

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, $k$, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

for number of training iterations do
  for $k$ steps do
    • Sample minibatch of $m$ noise samples $\{z^{(1)}, \ldots, z^{(m)}\}$ from noise prior $p_g(z)$.
    • Sample minibatch of $m$ examples $\{x^{(1)}, \ldots, x^{(m)}\}$ from data generating distribution $p_{\text{data}}(x)$.
    • Update the discriminator by ascending its stochastic gradient:
      \[
      \nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[ \log D\left(x^{(i)}\right) + \log \left(1 - D\left(G\left(z^{(i)}\right)\right)\right) \right].
      \]
  end for
  • Sample minibatch of $m$ noise samples $\{z^{(1)}, \ldots, z^{(m)}\}$ from noise prior $p_g(z)$.
  • Update the generator by descending its stochastic gradient:
    \[
    \nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log \left(1 - D\left(G\left(z^{(i)}\right)\right)\right).
    \]
end for

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.
GAN: Result

Rightmost column shows the nearest training example of the neighboring sample, in order to demonstrate that the model has not memorized the training set.

GAN: Result

Image source: https://sthalles.github.io/intro-to-gans/
GAN: Frontiers

Image source: Ian Goodfellow. Samples from Goodfellow et al., 2014, Radford et al., 2015, Liu et al., 2016, Karras et al., 2017, Karras et al., 2018
The GAN Zoo: Explosion of GANs

- 3D-ED-GAN - Shape Inpainting using 3D Generative Adversarial Network and Recurrent Convolutional Networks
- 3D-GAN - Learning a Probabilistic Latent Space of Object Shapes via 3D Generative-Adversarial Modeling (github)
- 3D-IWGAN - Improved Adversarial Systems for 3D Object Generation and Reconstruction (github)
- 3D-PhysNet - 3D-PhysNet: Learning the Intuitive Physics of Non-Rigid Object Deformations
- 3D-ReccGAN - 3D Object Reconstruction from a Single Depth View with Adversarial Learning (github)
- ABC-GAN - ABC-GAN: Adaptive Blur and Control for improved training stability of Generative Adversarial Networks (github)
- ABC-GAN - GANs for LIFE: Generative Adversarial Networks for Likelihood-Free Inference
- AC-GAN - Conditional Image Synthesis With Auxiliary Classifier GANs
- AC-GAN - Face Aging With Conditional Generative Adversarial Networks
- ACGAN - Coverless Information Hiding Based on Generative adversarial networks
- ACGAN - On-line Adaptive Curriculum learning for GANs
- ACTUAL - ACTUAL: Actor-Critic Under Adversarial Learning
- AdaGAN - AdaGAN: Boosting Generative Models
- Adaptive GAN - Customizing an Adversarial Example Generator with Class-Conditional GANs
- AdvEntRe - AdvEntRe: Adversarial Training for Textual Entailment with Knowledge-Guided Examples
- AdvGAN - Generating adversarial examples with adversarial networks
- AE-GAN - AE-GAN: adversarial eliminating with GAN
- AE-OT - Latent Space Optimal Transport for Generative Models
- AEGAN - Learning Inverse Mapping by Autoencoder based Generative Adversarial Nets
- AF-DCGAN - AF-DCGAN: Amplitude Feature Deep Convolutional GAN for Fingerprint Construction in Indoor Localization System
- AIRGAN - Amortised MAP inference for Image Super-resolution
- AIM - Generating Informative and Diverse Conversational Responses via Adversarial Information Maximization
- AL-CGAN - Learning to Generate Images of Outdoor Scenes from Attributes and Semantic Layouts
- ALI - Adversarially Learned Inference (github)
- AlignGAN - AlignGAN: Learning to Align Cross-Domain Images with Conditional Generative Adversarial Networks
- AlphaGAN - AlphaGAN: Generative adversarial networks for natural image matting
- AM-GAN - Activation Maximization Generative Adversarial Nets
- AmbientGAN - AmbientGAN: Generative models from lossy measurements (github)
- AMC-GAN - Video Prediction with Appearance and Motion Conditions
- AnoGAN - Unsupervised Anomaly Detection with Generative Adversarial Networks to Guide Marker Discovery
- APD - Adversarial Distillation of Bayesian Neural Network Posteriors
- APE-GAN - APE-GAN: Adversarial Perturbation Elimination with GAN
- ARAE - Adversarially Regularized Autoencoders for Generating Discrete Structures (github)
- ARDA - Adversarial Representation Learning for Domain Adaptation
- ARIGAN - ARIGAN: Synthetic Arabidopsis Plants using Generative Adversarial Network
- ArtGAN - ArtGAN: Artwork Synthesis with Conditional Categorical GANs
- ASDL-GAN - Automatic Steganographic Distortion Learning Using a Generative Adversarial Network
- ATA-GAN - Attention-Aware Generative Adversarial Networks (ATA-GANs)
- Attention-GAN - Attention-GAN for Object Transfiguration in Wild Images
- AttnGAN - AttnGAN: Fine-Grained Text to Image Generation with Attentional Generative Adversarial Networks (github)
- AVID - AVID: Adversarial Visual Irregularity Detection
- B-DCGAN - B-DCGAN:Evaluation of binarized DCGAN for FPGA
- b-GAN - Generative Adversarial Nets from a Density Ratio Estimation Perspective
- BAGAN - BAGAN: Data Augmentation with Balancing GAN
- Bayesian GAN - Deep and Hierarchical Implicit Models
- Bayesian GAN - Bayesian GAN (github)
- BCGAN - Bayesian Conditional Generative Adversarial Networks
- BCGAN - Bidirectional Conditional Generative Adversarial networks
- BEAM - Boltzmann Encoded Adversarial Machines
- BÉG - BÉG: Boundary Equilibrium Generative Adversarial Networks
- BÉG-SC - Escaping from Collapsing Modes in a Constrained Space
- Bellman GAN - Distributional Multivariate Policy Evaluation and Exploration with the Bellman GAN
- BGAN - Binary Generative Adversarial Networks for Image Retrieval (github)
- BI-GAN - Autonomously and Simultaneously Refining Deep Neural Network Parameters by a Bi-Generative Adversarial Network Aided Genetic Algorithm
- BicyclegAN - Toward Multimodal image-to-image Translation (github)
- BIGAN - Adversarial Feature Learning
- BinGAN - BinGAN: Learning Compact Binary Descriptors with a Regularized GAN
- BourGAN - BourGAN: Generative Networks with Metric Embeddings
- BranchGAN - BranchGAN: Branched Generative Adversarial Networks for Multi-Scale Image Manifold Learning
- BRE - Improving GAN training via Binarized Representation Entropy (BRE) Regularization (github)

Source: https://github.com/hindupuravinash/the-gan-zoo
DCGAN
Vanilla GAN simply uses MLP, rather than CNN in both generator and discriminator.

CNN can be easily applied to discriminator.

Now the problem is: how can CNN be used as a generator?

- Pooling leads to downsampling, how to upsampling?
Fractionally-Strided Convolutions

Input: 2×2

Filter: 3×3

Output: 4×4
Fractionally-Strided Convolutions

Input: 2×2

Filter: 3×3

Output: 4×4
Fractionally-Strided Convolutions

Input: 2×2

Filter: 3×3

Output: 4×4
Fractionally-Strided Convolutions

Input: 2×2

Filter: 3×3

Output: 4×4

```
Input:

1  2
3  4

Filter:

10  20  30
40  50  60
30  20  10

Output:

10  40  70  60
70  190  250  120+
150  230+ 230+ 20+
90  60+ 30+ 40
```
Fractionally-Strided Convolutions

- Fractionally-strided convolutions are also called **transposed convolutions**.
  - **PyTorch**: `torch.nn.ConvTranspose2d`.
  - **TensorFlow**: `tf.keras.layers.Conv2DTranspose`.

- Some researchers are used to call **deconvolutions**. However, true deconvolutions are the inverse operation of convolution, which is not the same as fractionally-strided convolutions.

Transposed convolution with stride is equivalent to convolving with zero-padding and inserting zeros.

Image source: https://github.com/vdumoulin/conv_arithmetic
Architecture guidelines for stable Deep Convolutional GANs

- Replace any pooling layers with strided convolutions (discriminator) and fractional-strided convolutions (generator).
- Use batchnorm in both the generator and the discriminator.
- Remove fully connected hidden layers for deeper architectures.
- Use ReLU activation in generator for all layers except for the output, which uses Tanh.
- Use LeakyReLU activation in the discriminator for all layers.
DCGAN: Visual Results

The generated bedrooms look very nice!

DCGAN: Walking in the Latent Space

- If walking in this latent space results in semantic changes to the image generations (such as objects being added and removed), we can reason that the model has learned relevant and interesting representations.

Interpolation between a series of 9 random points in $Z$ show that the space learned has smooth transitions.

DCGAN: Vector Arithmetic

For each column, the $Z$ vectors of samples are averaged. Arithmetic was then performed on the mean vectors creating a new vector $Y$. 

DCGAN: Use as Feature Extractor

- Train on Imagenet-1k and then use the discriminator’s convolutional features from all layers.
- Maxpooling each layers representation to produce a $4 \times 4$ spatial grid.
- These features are then flattened and concatenated to form a 28672 dimensional vector.
- A regularized linear L2-SVM classifier is trained on top of them.

<table>
<thead>
<tr>
<th>Model</th>
<th>Accuracy</th>
<th>Accuracy (400 per class)</th>
<th>max # of features units</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Layer K-means</td>
<td>80.6%</td>
<td>63.7% (±0.7%)</td>
<td>4800</td>
</tr>
<tr>
<td>3 Layer K-means Learned RF</td>
<td>82.0%</td>
<td>70.7% (±0.7%)</td>
<td>3200</td>
</tr>
<tr>
<td>View Invariant K-means</td>
<td>81.9%</td>
<td>72.6% (±0.7%)</td>
<td>6400</td>
</tr>
<tr>
<td>Exemplar CNN</td>
<td>84.3%</td>
<td>77.4% (±0.2%)</td>
<td>1024</td>
</tr>
<tr>
<td>DCGAN (ours) + L2-SVM</td>
<td>82.8%</td>
<td>73.8% (±0.4%)</td>
<td>512</td>
</tr>
</tbody>
</table>
We can’t control what we generate from the vanilla GAN.
- Noise is the only input and it is totally random.

How can we tell GAN what we want it to generate?

Straightforward solution: replace data distribution by conditional distribution.

\[ p(x) \rightarrow p(x|y). \]

Now, the problem becomes:
- Generator: generate a sample for class \( y \).
- Discriminator: distinguish the real sample in class \( y \) and the generated sample in class \( y \).
CGAN

- Both generator and discriminator are conditioned on some extra information $y$:
  $$\min_{\theta_g} \max_{\theta_d} \left[ \mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x|y) \right. \left. + \mathbb{E}_{z \sim p(z)} \log \left(1 - D_{\theta_d}\left(G_{\theta_g}(z|y)\right)\right) \right].$$

- $y$ could be any kind of auxiliary information, such as class labels or data from other modalities.
  - E.g. the speech of saying that class.
Generate MNIST digits by directly feeding one-hot class label.
WGAN
The first paper lists several problems of GANs based on theoretical analysis:

- Unstable behavior of training.
- The cost function doesn’t represent the quality of generated images.

Based on the analysis, the second paper proposed Wasserstein GAN (WGAN).
Problems of GANs

Problem 1: Good discriminator makes the gradient of generator vanish.

- If discriminator is well trained, the gradient of generator vanishes.
- If discriminator is poorly trained, the gradient of generator has great variance.
- Generator gets well trained only if discriminator is neither too good nor too poor.
  - That’s why GANs are extremely difficult to train.
Problems of GANs

- Using the annotations in the paper, we have the original loss function to minimize:

\[ L(D, g_\theta) = \mathbb{E}_{x \sim P_r}[\log D(x)] + \mathbb{E}_{x \sim P_g}[\log(1 - D(x))] \]

- By making the derivative of \( D(x) \) to 0, the optimal discriminator has the shape:

\[ D^*(x) = \frac{P_r(x)}{P_r(x) + P_g(x)} \]

- Replace \( D^*(x) \) into the loss function we get:

\[ L(D^*, g_\theta) = 2JS(P_r \| P_g) - 2 \log 2, \]

where \( JS \) is the Jensen-Shannon divergence. This is the loss of generator with optimal discriminator.

- In most of cases, \( JS(P_r \| P_g) = \log 2 \), which is a constant. That makes the gradient of generator 0.
Problems of GANs

Problem 2: the objective of the $- \log D$ trick leads to collapse mode, where the generated samples lack of diversity.

- Ian Goodfellow also proposed to replace $\log(1 - D(x))$ by $\log(-D(x))$ to solve the generator gradient vanishing problem. It is usually referred to the $- \log D$ trick.

- Then, the loss of generator with optimal discriminator was derived:

$$L(D^*, g_\theta) = KL(P_g || P_r) - 2JS(P_r || P_g)$$

where $KL$ is the KL divergence.
Problems of GANs

- By optimizing this generator loss, we may encounter collapse mode.
- The loss penalizes
  - slightly for the case that generator is not able to generate real sample;
  - heavily for the case that generator generates unreal samples.
- Thus, generator is more likely to generate safe samples (low diversity), rather than trying to explore new sample space.
Replace JS and KL divergence by Wasserstein distance, aka the Earth-Mover (EM) distance:

\[
W(P_r, P_g) = \inf_{\gamma \in \Pi(P_r, P_g)} \mathbb{E}_{(x,y) \sim \gamma} [||x - y||].
\]

where \(\Pi(P_r, P_g)\) denotes the set of all joint distributions \(\gamma(x, y)\) whose marginals are respectively \(P_r\) and \(P_g\).

Intuitively, \(\gamma(x, y)\) indicates how much “mass” must be transported from \(x\) to \(y\) in order to transform the distributions \(P_r\) into the distribution \(P_g\).

The EM distance then is the “cost” of the optimal transport plan.

The new loss function can represent image quality.
How to make the optimization to minimize Wasserstein distance?

Only three modifications to the original GAN:

1. Remove sigmoid activation in discriminator.
2. Remove log in loss of both discriminator and generator.
3. Weight clipping in a certain range $[-c, c]$. 
Algorithm 1 WGAN, our proposed algorithm. All experiments in the paper used the default values $\alpha = 0.00005$, $c = 0.01$, $m = 64$, $n_{\text{critic}} = 5$.

Require: $\alpha$, the learning rate. $c$, the clipping parameter. $m$, the batch size. $n_{\text{critic}}$, the number of iterations of the critic per generator iteration.

Require: $w_0$, initial critic parameters. $\theta_0$, initial generator’s parameters.

1: while $\theta$ has not converged do
2:     for $t = 0, \ldots, n_{\text{critic}}$ do
3:         Sample $\{x^{(i)}\}_{i=1}^m \sim \mathbb{P}_r$ a batch from the real data.
4:         Sample $\{z^{(i)}\}_{i=1}^m \sim p(z)$ a batch of prior samples.
5:         $g_w \leftarrow \nabla_w \left[ \frac{1}{m} \sum_{i=1}^m f_w(x^{(i)}) - \frac{1}{m} \sum_{i=1}^m f_w(g_\theta(z^{(i)})) \right]
6:         $w \leftarrow w + \alpha \cdot \text{RMSProp}(w, g_w)$
7:         $w \leftarrow \text{clip}(w, -c, c)$
8:     end for
9:     Sample $\{z^{(i)}\}_{i=1}^m \sim p(z)$ a batch of prior samples.
10:    $g_\theta \leftarrow -\nabla_\theta \frac{1}{m} \sum_{i=1}^m f_w(g_\theta(z^{(i)}))$
11:    $\theta \leftarrow \theta - \alpha \cdot \text{RMSProp}(\theta, g_\theta)$
12: end while

Weight clipping

No sigmoid

No log

Replace Adam by RMSProp
The Wasserstein estimate directly reflects the quality of generated images.

WGAN: Result

DCGAN with WGAN

DCGAN with GAN

WGAN without BN

GAN without BN

WGAN using MLP

GAN using MLP

**SAGAN: Motivation**

Problem of long-range dependencies.

- Traditional convolutional GANs generate high-resolution details as a function of only spatially local points in lower-resolution feature maps.

- Long range dependencies can only be processed after passing through several convolutional layers.

- It fails to capture geometric or structural patterns that occur consistently in some classes.
  - For example, dogs are often drawn with realistic fur texture but without clearly defined separate feet.

---

**Self-attention generative adversarial networks**

In this paper, we propose the Self-Attention Generative Adversarial Network (SAGAN) which allows attention-driven, long-range dependency modeling for image generation tasks. Traditional convolutional GANs generate high-resolution details as a function of only spatially local points in lower-resolution feature maps. In SAGAN, details can be generated using cues from all feature locations. Moreover, the discriminator can check that highly detailed features in distant portions of the image are consistent with each other.
SAGAN

Three mapping functions $f(x), g(x), h(x)$ and finally a weighted feature is generated.

Is it similar to something we have seen before?
SAGAN

- goldfish (44.4, 58.1)
- indigo bunting (53.0, 66.8)
- redshank (48.9, 60.1)
- saint bernard (35.7, 55.3)

PIX2PIX AND CYCLEGAN
Given a pair of images, transfer the style of one image to another.

Image-to-image translation with conditional adversarial networks
P. Isola, J.Y. Zhu, T. Zhou… - Proceedings of the IEEE ..., 2017 - openaccess.thecvf.com
We investigate conditional adversarial networks as a general-purpose solution to image-to-image translation problems. These networks not only learn the mapping from input image to output image, but also learn a loss function to train this mapping. This makes it possible to apply the same generic approach to problems that traditionally would require very different loss formulations. We demonstrate that this approach is effective at synthesizing photos from label maps, reconstructing objects from edge maps, and colorizing images, among other ...

The model is based on CGAN.

As an improvement, the generator is tasked to not only fool the discriminator but also to be near the ground truth output.

$L_1$ penalization is added to the loss of CGAN to encourage less blurring:

$$L_{L_1}(G) = \mathbb{E}_{x,y,z}[\|y - G(x, z)\|_1].$$
pix2pix: More Applications

#edges2cats by Christopher Hesse

Sketch → Portrait by Mario Klingemann

Depth → Streetview by Jasper van Loenen

Background removal by Kaihu Chen

“Do as I do” by Brannon Dorsey

Palette generation by Jack Qiao

Sketch → Pokemon by Bertrand Gondouin

CycleGAN

- Paired examples can be expensive to obtain.
- Can we translate from $X \leftrightarrow Y$ in an unsupervised manner?

Unpaired image-to-image translation using cycle-consistent adversarial networks

JY Zhu, T Park, P Isola, AA Efros - Proceedings of the IEEE ..., 2017 - openaccess.thecvf.com

Image-to-image translation is a class of vision and graphics problems where the goal is to learn the mapping between an input image and an output image using a training set of aligned image pairs. However, for many tasks, paired training data will not be available. We …

Paired vs. unpaired examples
CycleGAN

- **Two generators:**
  - $G: X \rightarrow Y$;
  - $F: Y \rightarrow X$.

- **Two discriminators:**
  - $D_X$ aims to distinguish between images $\{x\}$ and translated images $\{F(y)\}$;
  - $D_Y$ aims to discriminate between $\{y\}$ and $\{G(x)\}$.

CycleGAN

- If we can go from $X$ to $\hat{Y}$ via $G$, then it should also be possible to go from $\hat{Y}$ back to $X$ via $F$.

- Cycle consistency loss is added to the original adversarial loss:

$$L_{cyc}(G, F) = \mathbb{E}_x \left[ \| F(G(x)) - x \|_1 \right] + \mathbb{E}_y \left[ \| G(F(x)) - y \|_1 \right].$$
CycleGAN

Failed case:

SRGAN
SRGAN

- Task: Given a low-resolution image, reconstruct the high-resolution image.
- Previous work: Residual transpose CNN with MSE loss has shown great performance on this task but the reconstructions are blurry.

Photo-realistic single image super-resolution using a generative adversarial network
C. Ledig, L. Theis, F. Huszár… - Proceedings of the …, 2017 - openaccess.thecvf.com

Despite the breakthroughs in accuracy and speed of single image super-resolution using faster and deeper convolutional neural networks, one central problem remains largely unsolved: how do we recover the finer texture details when we super-resolve at large …

SRGAN

- **Generator:** Similar to ResNet, but uses Transposed Convolutions to increase the spatial dimensions.
- **Discriminator:** Fully-convolutional CNN + MLP Classifier + Binary Cross Entropy.

SRGAN: VGG Loss

- Pixel-wise MSE loss often lacks high-frequency content which results in smooth textures:
  \[
  l^{SR}_{MSE} = \frac{1}{r^2 WH} \sum_{x=1}^{rW} \sum_{y=1}^{rH} (I_{x,y}^{HR} - G_{\theta_G}(I_{x,y}^{LR}))^2
  \]

- The authors uses VGG loss based on the ReLU activation layers of the pre-trained 19 layer VGG network
  \[
  l^{SR}_{VGG/i,j} = \frac{1}{W_{i,j} H_{i,j}} \sum_{x=1}^{W_{i,j}} \sum_{y=1}^{H_{i,j}} (\phi_{i,j}(I_{x,y}^{HR}) - \phi_{i,j}(G_{\theta_G}(I_{x,y}^{LR})))^2
  \]

- \(\phi_{i,j}\) is the feature map obtained by the \(j\)-th convolution (after activation) before the \(i\)-th maxpooling layer.

Image source: Lecture 4, MSBA434, UCLA
SRGAN: Dual Loss

- Weighted sum of adversarial loss and VGG activation loss:

\[ l^{SR} = l_x^{SR} + 10^{-3} l_{Gen}^{SR} \]

- Content loss
- Adversarial loss
SRGAN: Results

**General Idea of Using GAN**

- GAN can be applied to lots of applications, rather than just images.
- We should identify several things to make sure that our task can be solved by GAN.
  - What are fake samples?
  - What are real samples?
  - How to transform fake samples to real samples (generator)?
  - How to distinguish fake sample from real samples (discriminator)?
Conclusion

After this lecture, you should know:

- What is the basic idea of GAN?
- How generator and discriminator improve each other?
- How does transposed convolution work?
- How to design application specific loss to train with adversarial loss?
Suggested Reading

- Adversarial Nets Papers
- Tips and tricks to make GANs work
- 惊人拍案叫绝的Wasserstein GAN
Assignment 4 is released. The deadline is 18:00, 11th December.
Any question?

Don’t hesitate to send email to me for asking questions and discussion. 😊