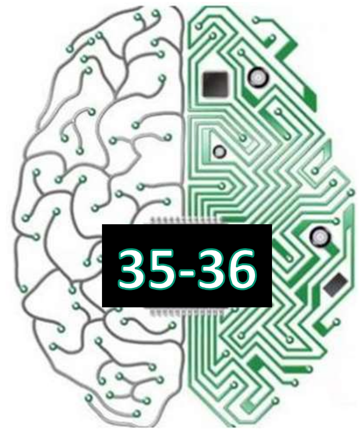
Open Elective Course [OE]
Course Code: CSO507
Winter 2023-24

Lecture#

Deep Learning

Unit-8: Generative Models (Part-III)



Course Instructor:

Dr. Monidipa Das
Assistant Professor
Department of Computer Science and Engineering
Indian Institute of Technology (Indian School of Mines) Dhanbad, Jharkhand 826004, India

Variational Autoencoders

2

1. We modelled the autoencoding process using a probabilistic (Bayesian) framework, with an observed and a hidden variable
2. We wanted to calculate the posterior distribution $p_\theta(z|x)$, but this is complicated
3. We used an approximation $q_\phi(z|x)$ to $p_\theta(z|x)$
4. We used the ELBO as a loss function to minimise $KL(q_\phi(z|x)||p_\theta(z))$
   - Ensures a good reconstruction
   - Encourages the latent space to follow our chosen distribution (the prior $p_\theta(z)$)

Variational Autoencoders

3

- Probabilistic spin to traditional autoencoders => allows generating data
- Defines an intractable density => derive and optimize a (variational) lower bound

- Advantages
  - Theoretically-motivated, loss function meaningful
  - Learn to and from mapping (encoder and decoder)

- Drawbacks
  - In practice, samples blurrier and lower quality
  - Have to re-write loss function for each different model, not always easy

Variational Autoencoders

4

VAEs define intractable density function with latent variables z (that we need to marginalize):

$$p_\theta(x) = \int p_\theta(z) p_\theta(x|z) dz$$

cannot optimize directly, derive and optimize lower bound of likelihood instead

What if we give up on explicitly modeling density, and just want to sample?

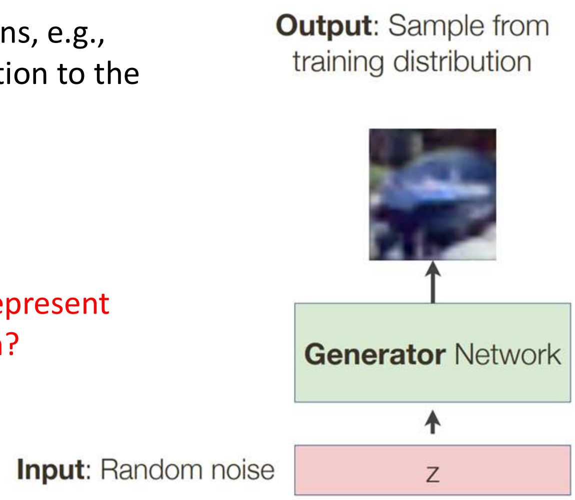
GANs: don't work with any explicit density function

Generative Adversarial Networks (GANs)

5

Sample from a simple distributions, e.g., random noise. Learn transformation to the training distribution

Question: What can we use to represent complex transformation function?



Generative Adversarial Networks (GANs)

6

- Generative
  - Learn a generative model
- Adversarial
  - Trained in an adversarial setting
- Networks
  - Use Deep Neural Networks

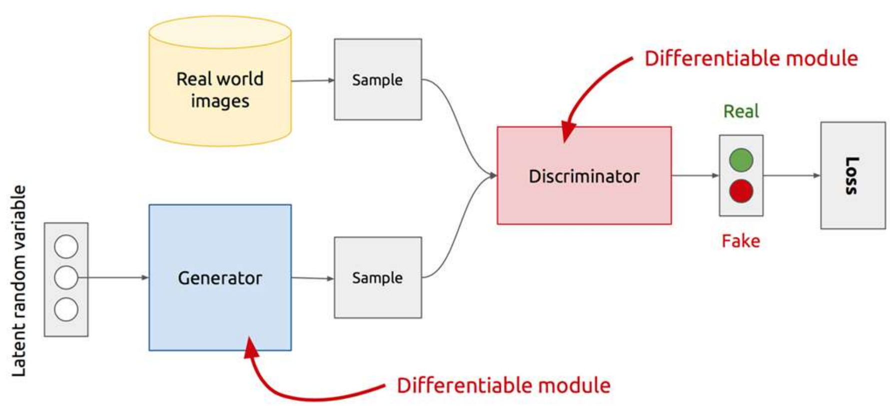
Generative Adversarial Networks (GANs)

7

- Adversarial Training
  - Can generate adversarial samples to fool a discriminative model
  - Can use those adversarial samples to make models robust
  - Require more effort to generate adversarial samples
  - Repeat this and we get better discriminative model

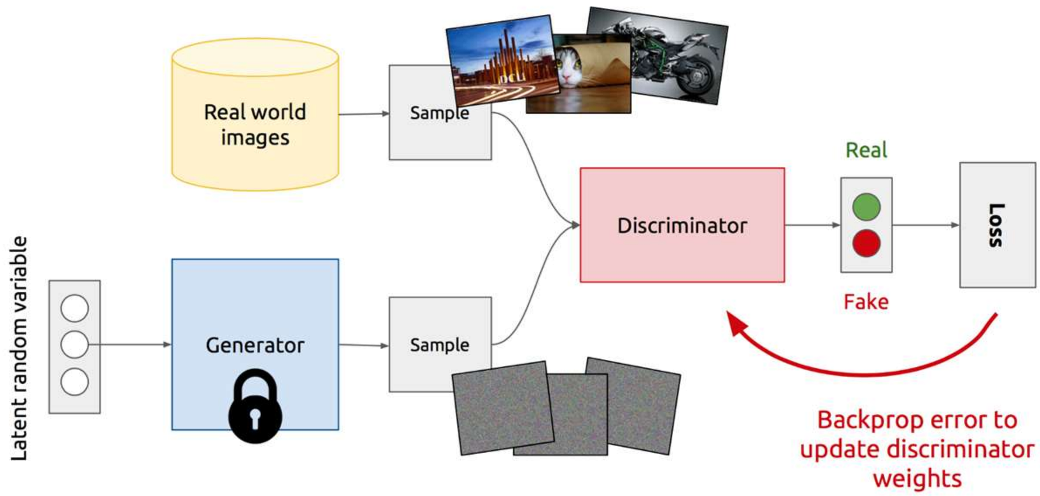
GAN's Architecture

8

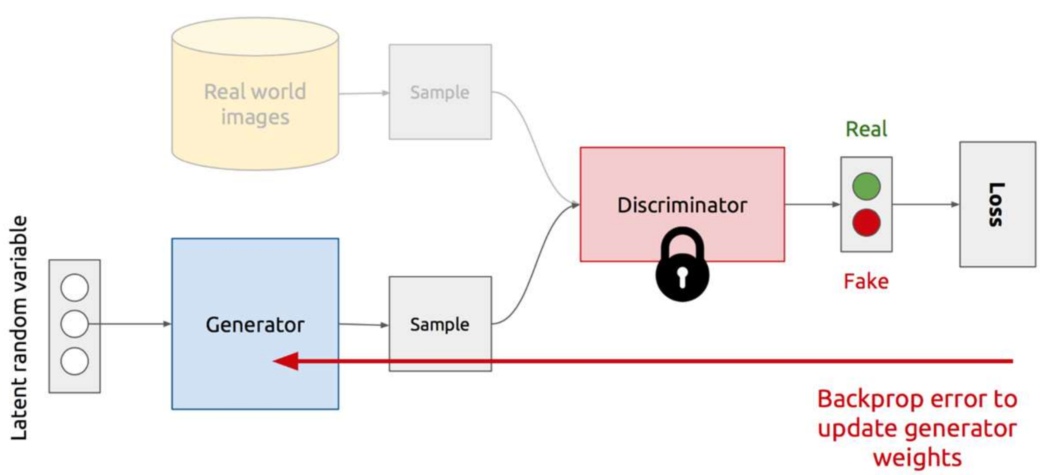


- **Z** is some random noise (Gaussian/Uniform).
- **Z** can be thought as the latent representation of the image.

Training Discriminator



Training Generator



Training GAN: Formulation

11

- Training GNN: Two-player Game

$$\min_{G} \max_{D} V(D, G)$$

- Formulated as a **minimax game**, where:
  - The Discriminator is trying to maximize its reward $\boldsymbol{V(D, G)}$
  - The Generator is trying to minimize Discriminator's reward (or maximize its loss)

$$\boldsymbol{V(D, G)} = \left[\mathbb{E}_{x \sim p_{data}} \log \underbrace{D_{\theta_d}(x)}_{\text{Discriminator output for real data x}} + \mathbb{E}_{z \sim p(z)} \log(1 - \underbrace{D_{\theta_d}(G_{\theta_g}(z))}_{\text{Discriminator output for generated fake data G(z)}})\right]$$

Training GAN

12

Minimax objective function:

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

Alternate between:

1. Gradient **ascent** on discriminator

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

2. Gradient **descent** on generator

$$\min_{\theta_g} \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z)))$$

Training GAN

13

Discriminator updates

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

Generator updates

- 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.

Training GAN

14

Minimax objective function:

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

Alternate between:

1. Gradient **ascent** on discriminator

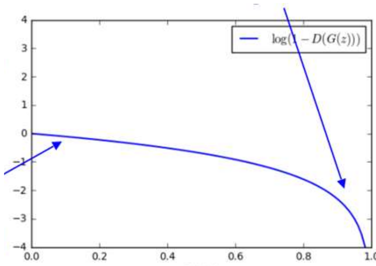
$$\max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

2. Gradient **descent** on generator

$$\min_{\theta_g} \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z)))$$

Gradient signal dominated by region where sample is already good

When sample is likely fake, want to learn from it to improve generator.



Training GAN

15

Minimax objective function:

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

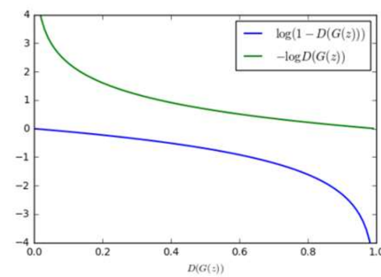
Alternate between:

1. Gradient **ascent** on discriminator

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

2. Instead, gradient **ascent** on generator, different objective

$$\max_{\theta_g} \mathbb{E}_{z \sim p(z)} \log(D_{\theta_d}(G_{\theta_g}(z)))$$



Advantages of GAN

16

- Plenty of existing work on Deep Generative Models but GANs
  - Don’t take explicit density function
  - Game theoretic Approach
  - Better Sample generation
- Why GANs?
  - Sampling (or generation) is straightforward
  - Training doesn't involve Maximum Likelihood estimation
  - Robust to Overfitting

Disadvantages of GAN

17

- Probability Distribution is Implicit
  - Not straightforward to compute P(X).
  - Thus Vanilla GANs are only good for Sampling/Generation
- Training is Hard
  - **Non-Convergence**
    - SGD is not designed to find equilibrium
  - **Mode-Collapse**
    - Can focus on a few realistic images from the training dataset

Non-Convergence

18

- **Deep Learning models (in general) involve a single player**
  - The player tries to maximize its reward (minimize its loss).
  - Use SGD (with Backpropagation) to find the optimal parameters.
  - SGD has convergence guarantees (under certain conditions).
  - **Problem:** With non-convexity, we might converge to local optima.

$$\min_G L(G)$$

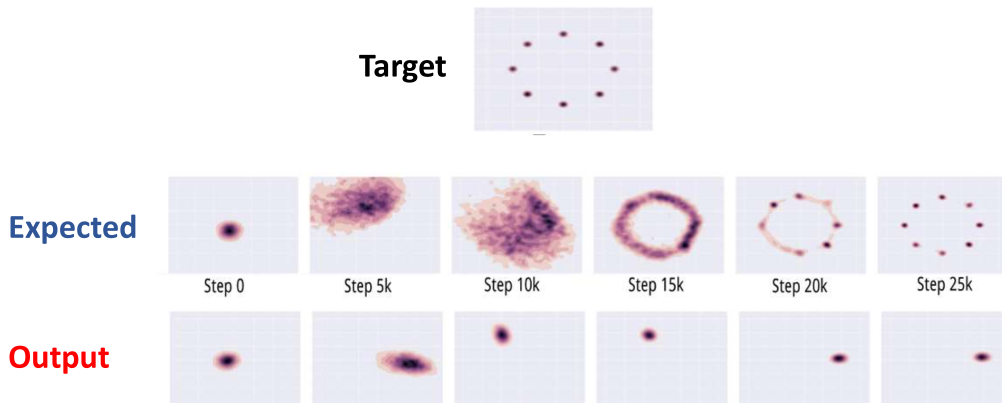
- **GANs instead involve two (or more) players**
  - Discriminator is trying to maximize its reward.
  - Generator is trying to minimize Discriminator's reward.

$$\min_G \max_D V(D, G)$$

  - SGD was not designed to find the Nash equilibrium of a game.
  - **Problem:** We might not converge to the Nash equilibrium at all.

Mode-Collapse

- **Generator fails to output diverse samples**



Generative Adversarial Network

Optimal GAN discriminator D^*

For a fixed G, the optimal D is $D^*(x) = \frac{p_{data}(x)}{p_{data}(x)+p_G(x)}$

$$\begin{aligned}\mathcal{L}(G,D) &= \mathbb{E}_{x\ \sim p_{data}}[\log D(x)] + \mathbb{E}_{z\ \sim p_z}[\log(1-D(G(z)))] \\ &= \int_{\mathcal{X}} p_{data}(x)\log(D(x))dx + \int_{\mathcal{Z}} p_z(z)\log(1-D(G(z)))dz \\ &= \int_{\mathcal{X}} p_{data}(x)\log(D(x)) + p_G(x)\log(1-D(x))dx.\end{aligned}$$

- For every x, the maximum of the previous equation w.r.t $D(x)$ is

$$D^*(x) = \frac{p_{data}(x)}{p_{data}(x)+p_G(x)}$$

Generative Adversarial Network

21

Global optimum of the GAN loss function

The global optimum of the GAN loss function is achieved if and only if $p_G = p_{data}$. At this point $\mathcal{L}(G, D) = -\log 4$.

- First, our previous lemma allows us to rewrite the loss function

$$\max_D \mathcal{L}(G, D) = \mathbb{E}_{x \sim p_{data}} [\log D^*(x)] + \mathbb{E}_{z \sim p_z} [\log(1 - D^*(G(z)))] \quad (3)$$

$$= \mathbb{E}_{x \sim p_{data}} [\log D^*(x)] + \mathbb{E}_{x \sim p_G} [\log(1 - D^*(x))]$$

$$= \mathbb{E}_{x \sim p_{data}} \left[\log \frac{p_{data}(x)}{p_{data}(x) + p_G(x)}\right] + \mathbb{E}_{x \sim p_G} \left[\log \left(\frac{p_G(x)}{p_{data}(x) + p_G(x)}\right)\right]. \quad (4)$$

- Therefore, if $p_G = p_{data}$, then $\mathcal{L}(G, D) = \log \frac{1}{2} + \log \frac{1}{2} = -\log(4)$

Generative Adversarial Network

22

- Now, we are going to show that $-\log(4)$ is the optimal value of the loss function
- First, we remark that :

$$-\log(4) = \mathbb{E}_{x \sim p_{data}} [-\log(2)] + \mathbb{E}_{x \sim p_G} [-\log(2)]. \quad (5)$$

- Therefore, by subtracting Equation 5 from Equation 4, we have

$$\mathcal{L}(G, D^*) = -\log(4) + \int p_{data}(x) \log \frac{p_{data}(x)}{\frac{1}{2}(p_{data}(x) + p_G(x))} dx + \int p_G(x) \log \frac{p_G(x)}{\frac{1}{2}(p_{data}(x) + p_G(x))} dx$$

$$= -\log(4) + KL\left(p_{data} \,||\, \frac{p_{data} + p_G}{2}\right) + KL\left(p_G \,||\, \frac{p_{data} + p_G}{2}\right).$$

Generative Adversarial Network

23

- This can also be rewritten as
$$\mathcal{L}(G, D^*) = -\log(4) + 2\,\mathbf{JSD}\,(\mathbf{p_{data}} \,||\, \mathbf{p_G}).$$
- The **JSD** is the **Jensen-Shannon divergence**
- This is another distance between distributions. For p and q, we have :
$$JSD(p,q) = \frac{1}{2}KL\left(p \,||\, \frac{1}{2}(p+q)\right) + \frac{1}{2}KL\left(q \,||\, \frac{1}{2}(p+q)\right)$$
- The JSD is **non-negative and equal to zero if and only if** $p_{data} = p_G$
- Therefore $-\log(4)$ is the optimal value, and only reached when $p_{data} = p_G$

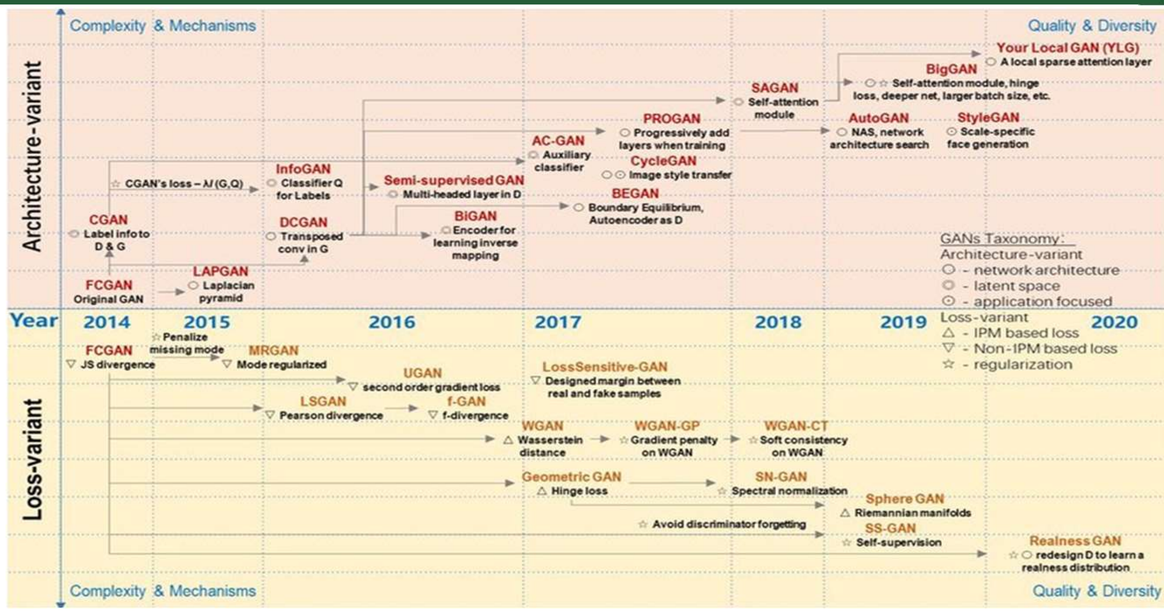
GAN Variants

24

- GAN - Generative Adversarial Networks
- 3D-GAN - Learning a Probabilistic Latent Space of Object Shapes via 3D Generative-Adversarial Modeling
- acGAN - Face Aging With Conditional Generative Adversarial Networks
- AC-GAN - Conditional Image Synthesis With Auxiliary Classifier GANs
- AdaGAN - AdaGAN: Boosting Generative Models
- AEGAN - Learning Inverse Mapping by Autoencoder based Generative Adversarial Nets
- AffGAN - Amortised MAP Inference for Image Super-resolution
- AL-CGAN - Learning to Generate Images of Outdoor Scenes from Attributes and Semantic Layouts
- ALI - Adversarially Learned Inference
- AM-GAN - Generative Adversarial Nets with Labeled Data by Activation Maximization
- AnoGAN - Unsupervised Anomaly Detection with Generative Adversarial Networks to Guide Marker Discovery
- ArtGAN - ArtGAN: Artwork Synthesis with Conditional Categorial GANs
- b-GAN - b-GAN: Unified Framework of Generative Adversarial Networks
- Bayesian GAN - Deep and Hierarchical Implicit Models
- BEGAN - BEGAN: Boundary Equilibrium Generative Adversarial Networks
- BiGAN - Adversarial Feature Learning
- BS-GAN - Boundary-Seeking Generative Adversarial Networks
- CGAN - Conditional Generative Adversarial Nets
- CaloGAN - CaloGAN: Simulating 3D High Energy Particle Showers in Multi-Layer Electromagnetic Calorimeters with Generative Adversarial Networks
- CCGAN - Semi-Supervised Learning with Context-Conditional Generative Adversarial Networks
- CatGAN - Unsupervised and Semi-supervised Learning with Categorical Generative Adversarial Networks
- CoGAN - Coupled Generative Adversarial Networks
- Context-RNN-GAN - Contextual RNN-GANs for Abstract Reasoning Diagram Generation
- C-RNN-GAN - C-RNN-GAN: Continuous recurrent neural networks with adversarial training
- CS-GAN - Improving Neural Machine Translation with Conditional Sequence Generative Adversarial Nets
- CVAE-GAN - CVAE-GAN: Fine-Grained Image Generation through Asymmetric Training
- CycleGAN - Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks
- DTN - Unsupervised Cross-Domain Image Generation
- DCGAN - Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks
- DiscoGAN - Learning to Discover Cross-Domain Relations with Generative Adversarial Networks
- DR-GAN - Disentangled Representation Learning GAN for Pose-Invariant Face Recognition
- DualGAN - DualGAN: Unsupervised Dual Learning for Image-to-Image Translation
- EBGAN - Energy-based Generative Adversarial Network
- f-GAN - f-GAN: Training Generative Neural Samplers using Variational Divergence Minimization
- FF-GAN - Towards Large-Pose Face Frontalization in the Wild
- GAWWN - Learning What and Where to Draw
- GeneGAN - GeneGAN: Learning Object Transfiguration and Attribute Subspace from Unpaired Data
- Geometric GAN - Geometric GAN
- GoGAN - Gang of GANs: Generative Adversarial Networks with Maximum Margin Ranking
- GP-GAN - GP-GAN: Towards Realistic High-Resolution Image Blending
- IAN - Neural Photo Editing with Introspective Adversarial Networks
- iGAN - Generative Visual Manipulation on the Natural Image Manifold
- IcGAN - Invertible Conditional GANs for image editing
- ID-CGAN - Image De-raining Using a Conditional Generative Adversarial Network
- Improved GAN - Improved Techniques for Training GANs
- InfoGAN - InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets
- LAGAN - Learning Particle Physics by Example: Location-Aware Generative Adversarial Networks for Physics Synthesis
- LAPGAN - Deep Generative Image Models using a Laplacian Pyramid of Adversarial Networks

Evolution of Generative Model

25



Laplacian Pyramid of Adversarial Networks

26

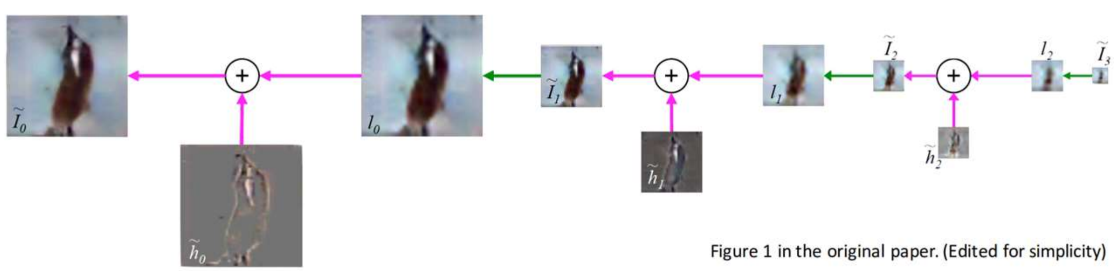


Figure 1 in the original paper. (Edited for simplicity)

- Based on the Laplacian Pyramid representation of images. (1983)
- Generate high resolution (dimension) images by using a hierarchical system of GANs
- Iteratively increase image resolution and quality.

Denton, E.L., Chintala, S. and Fergus, R., 2015. "**Deep Generative Image Models using a Laplacian Pyramid of Adversarial Networks**". NIPS (2015)

Laplacian Pyramid of Adversarial Networks

27

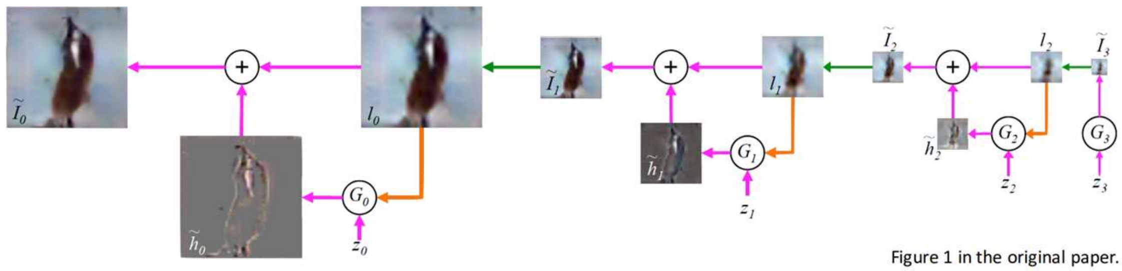


Figure 1 in the original paper.

Image Generation using a LAPGAN

- Generator G_3 generates the base image $\widetilde{I_3}$ from random noise input z_3.
- Generators (G_2, G_1, G_0) iteratively generate the *difference image* $(\hat{h})$ conditioned on previous small image (l).
- This *difference image* is added to an up-scaled version of previous smaller image.

Denton, E.L., Chintala, S. and Fergus, R., 2015. "**Deep Generative Image Models using a Laplacian Pyramid of Adversarial Networks**". NIPS (2015)

Laplacian Pyramid of Adversarial Networks

28

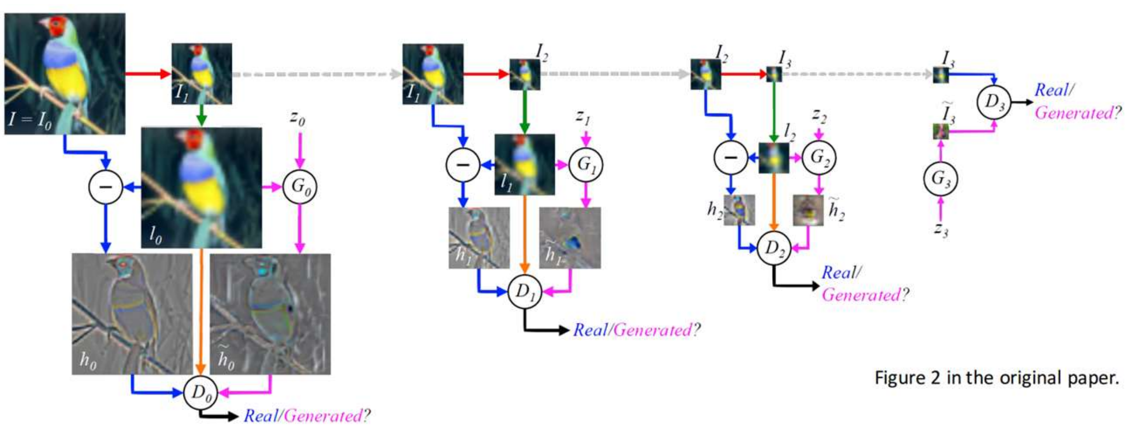


Figure 2 in the original paper.

Training Procedure:
Models at each level are trained independently to learn the required representation.

Denton, E.L., Chintala, S. and Fergus, R., 2015. "**Deep Generative Image Models using a Laplacian Pyramid of Adversarial Networks**". NIPS (2015)

Coupled GAN

29

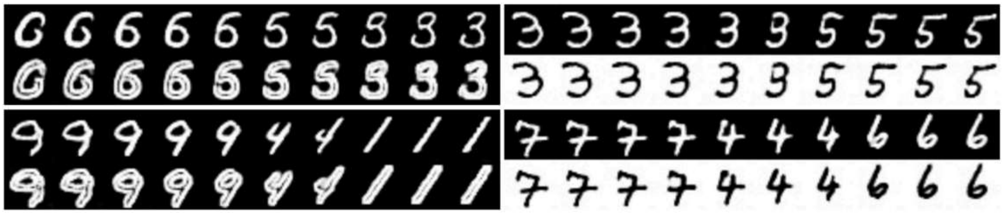


Figure 2 in the original paper.

- Learning a *joint distribution* of *multi-domain* images.
- Using GANs to learn the joint distribution with samples drawn from the marginal distributions.
- Direct applications in domain adaptation and image translation.

Liu, Ming-Yu, and Oncel Tuzel. "**Coupled generative adversarial networks**". NIPS (2016).

Coupled GANs

30

- Architecture

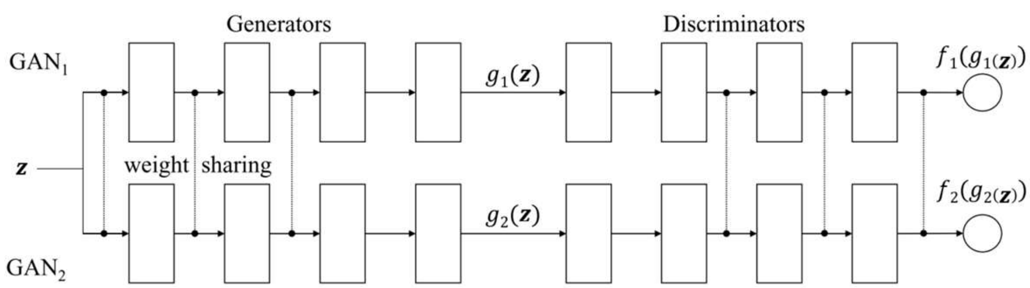


Figure 1 of the original paper.

Liu, Ming-Yu, and Oncel Tuzel. "**Coupled generative adversarial networks**". NIPS (2016).

Coupled GANs

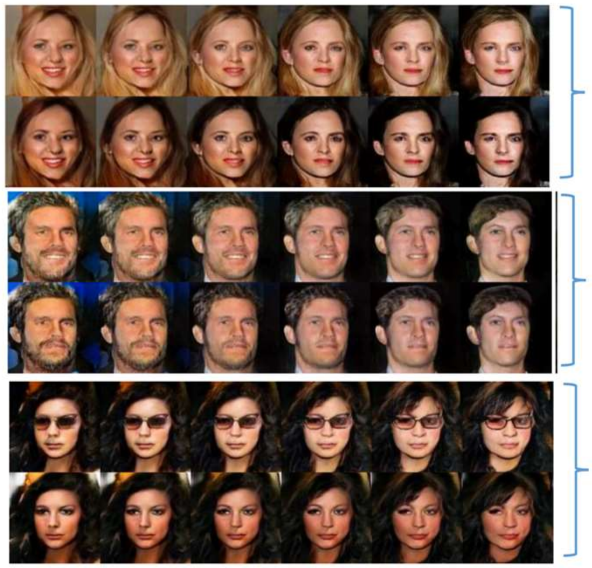


Figure 4 in the original paper.

StyleGAN Explained

StyleGAN attempts to tackle the limitation of GAN by adding progressive training to adjust each detail level separately. In doing so, **user can control visual features expressed in each detail level in an isolated manner without affecting other level**s.

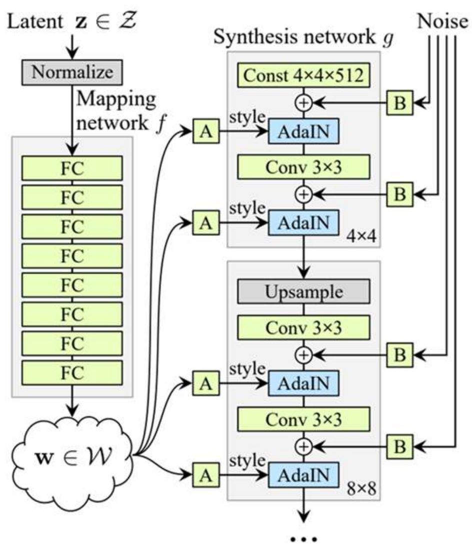
Detail level	Resolution	What is affected?
Coarse	Up to 82	Pose, general hair style, face shape etc.
Middle	162 to 322	Finer facial features, hair style, eyes open/closed etc.
Fine	642 to 10242	Color scheme (eye, hair and skin) and micro features.

Our generator thinks of an image as a collection of “styles”, where each style controls the effects at a particular scale

- Coarse styles ⟶ pose, hair, face shape
- Middle styles ⟶ facial features, eyes
- Fine styles ⟶ color scheme

StyleGAN Model Architecture

33



StyleGAN is described as a **progressive growing** GAN architecture with 5 modifications, each of which was added and evaluated incrementally:

- **Baseline Progressive GAN**
- Addition of tuning and bilinear up-sampling
- Addition of mapping network and AdaIN (styles)
- Removal of latent vector input to generator
- Addition of noise to each block.
- Addition of Mixing regularization

34

Questions?