Generative Adversarial Networks (GAN)

Michael(Mike) Erlihson, Ph.D.

Agenda

- GANs Applications
- So how do GANs work
- GAN Loss Function
- GANs types and architectures
- GANs Shortcomings

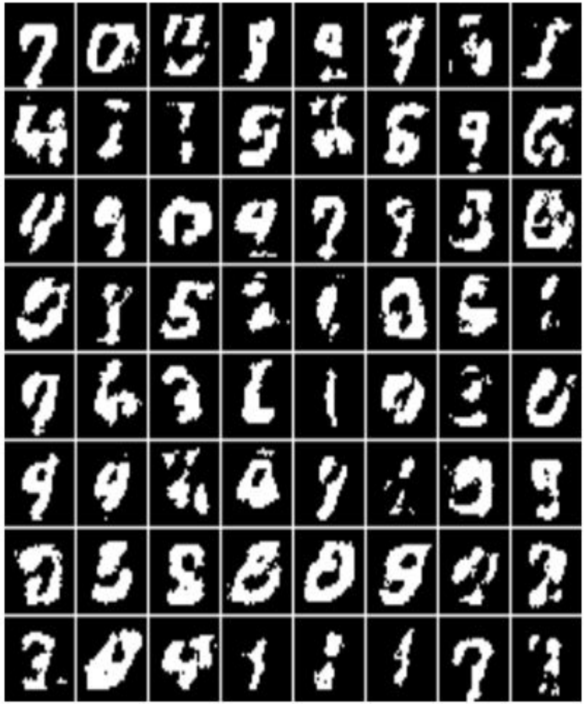
What is GAN

- Type of neural network architecture that allow neural networks to generate data.

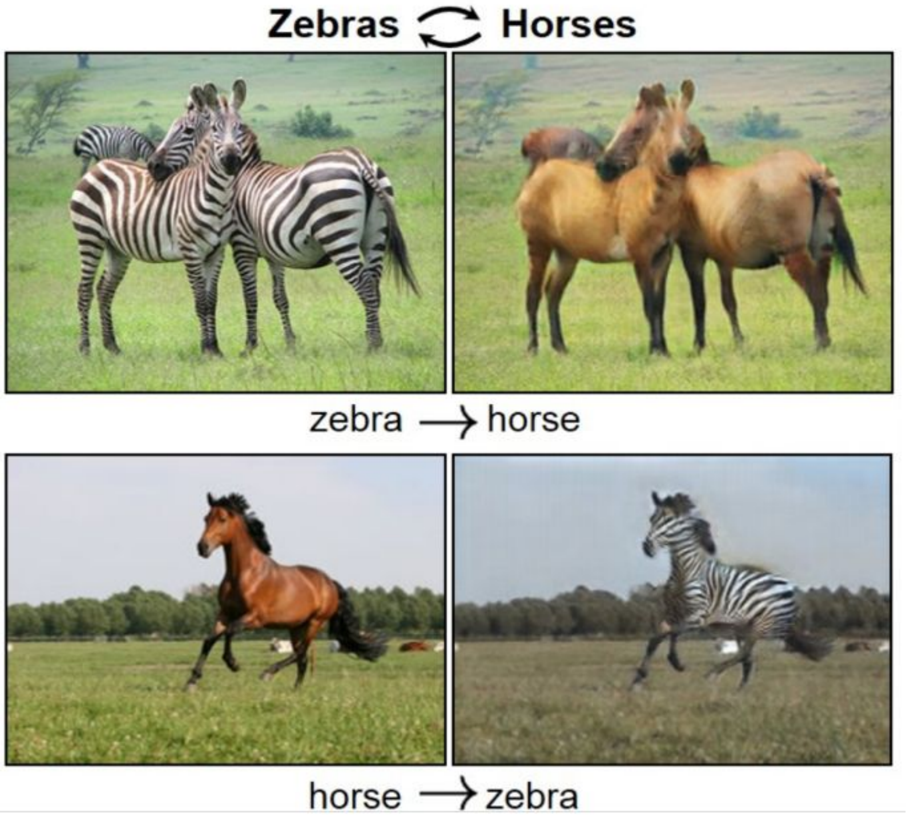
- Recently GANs became one of the hottest subfields in deep learning

- Generating fuzzy images of digits to photorealistic images of faces

What GANs are able to generate?



More advanced GAN capabilities:



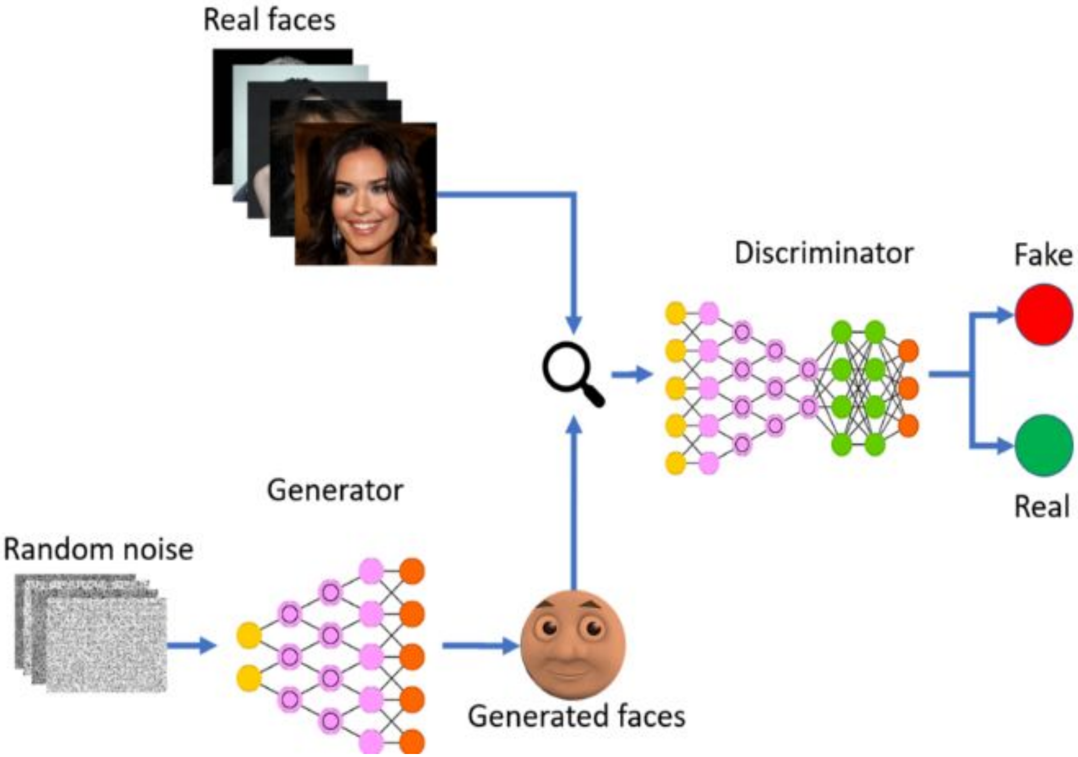
GANs Basics

- Learn dataset probability distribution via training two neural networks against each other.

- Solve two problems simultaneously:
    - **Discrimination** - distinguish between real and fake images)
    - **“Realistic” fake data generation** - construct samples similar to real

- These tasks are complete opposites - what would occur if we split them into different models: **Generator vs. Discriminator?**

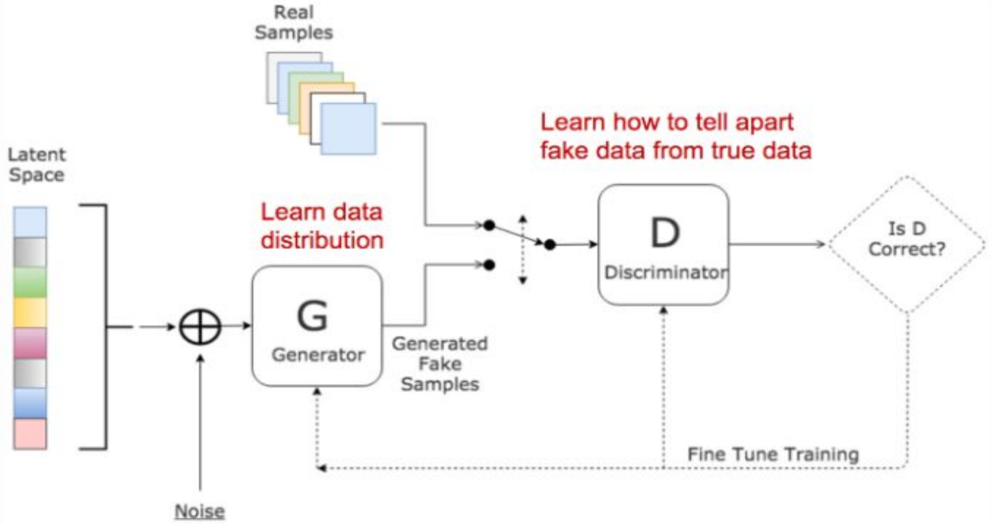
GAN Architecture: Basic Scheme



GAN Architecture: An essence

- **Generator**(painting forger): <u>tries to create images</u> looking very similar to dataset

- **Discriminator**(police)- <u>tries to detect</u> whether generated images were fake or not.



GAN Architecture: Training

Beginning of training

Later training stage

GAN Architecture: Training Process

- **Generator**(forger) is improving in creating fakes, while **discriminator**(police) keep getting better at detecting fakes.

- These two models keep competing with each other during the training process

- **Goal**: produce two networks that are <u>each as good as they can be</u>: we don't end up with a "winner"

- Finally the generator "learns" to create images "indistinguishable" from the real dataset

Training Procedure: Details

- Generator network takes as input a N-dimensional random noise and produces a fake image

- Discriminator network input is both fake samples as well as samples from the dataset

- **Generator Training Goal**: maximise D’s final classification error.
  - The generated images are perceived as real

- **Discriminator Training Goal**: minimise the final classification error.
  - Real data is correctly distinguished from fake data
  - Makes binary decision about image G(z): a real image *D(z)=1*, or a fake image, *D(z)=0*.

Loss Function

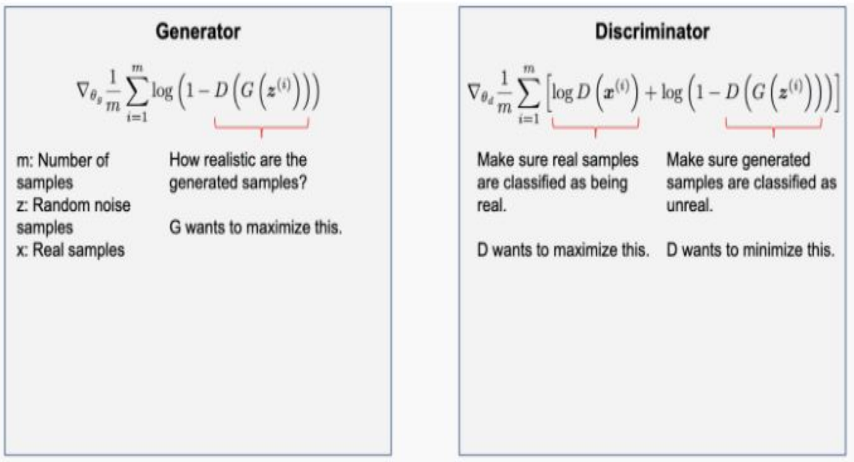
- Loss function should estimate the **cumulative performance** of the two networks?

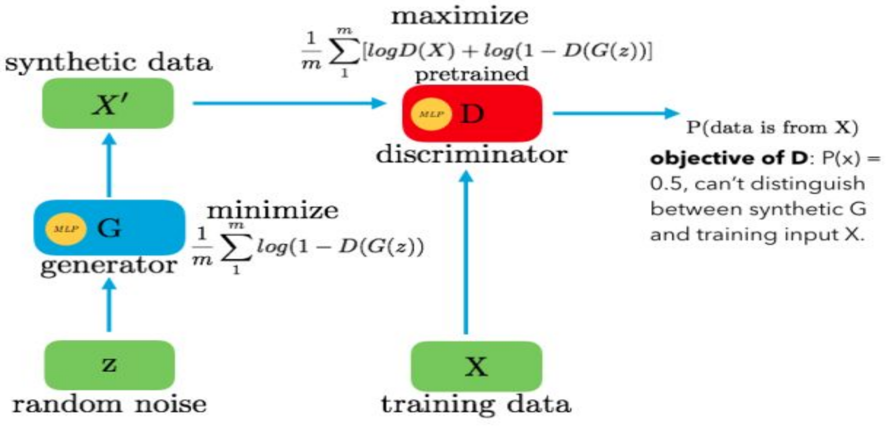
- Predictions on the dataset by the discriminator should be as close to 1 as possible, and on the generator to be as close to 0 as possible.

- To achieve this, use the log-likelihood of D(x) and 1-D(z) in the objective function.

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

Loss Function: Intuition





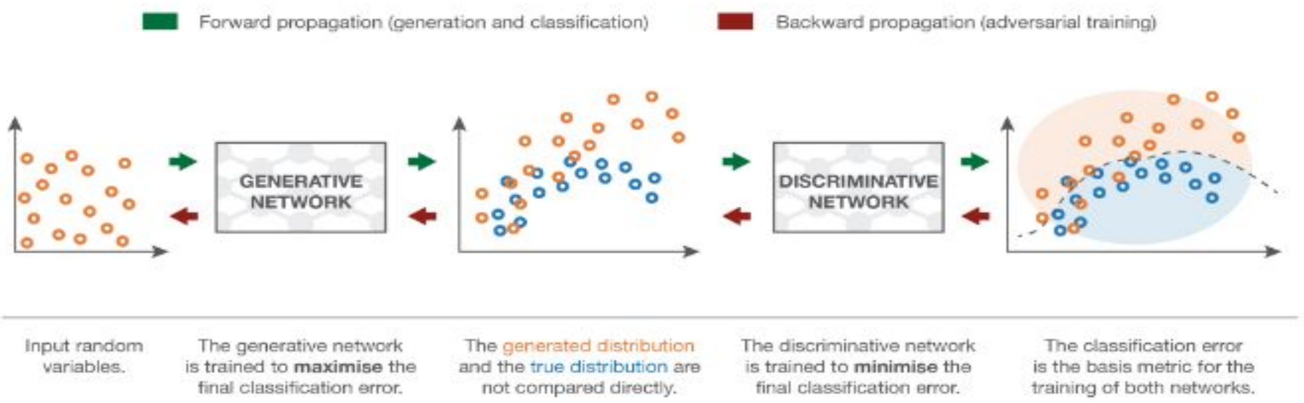
Architecture of GANs

For the Generator, we want to minimize log(1-D(G(z)) i.e. when the value of D(G(z)) is high then D will assume that G(z) is nothing but X and this makes 1-D(G(z)) very low and we want to minimize it which this even lower. For the Discriminator, we want to maximize D(X) and (1-D(G(z))). So the optimal state of D will be P(x)=0.5. However, we want to train the generator G such that it will produce the results for the discriminator D so that D won't be able to distinguish between z and X.

Training Procedure: Backpropagation:

- To train the networks we perform backpropagation whilst freezing the other network's neuron weights.

- Generator's weights are updated using <u>Gradient Ascent</u> to maximise the error, while Discriminator uses <u>Gradient Descent</u> to <u>minimise</u> it.



Training Procedure: Details

- Perform several iterations of gradient descent on D using real and generated images by fixing G

- Then fix D and train G for several iterations (usually more iterations required to train G) to fool a fixed D.

- We want to optimize our minimax function by iterating both G and D in alternating steps until discriminator won't be able to differentiate between real and fake images ($D(x) = 0.5$)

Training Procedure: Algorithm

< itc >

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 $\{\boldsymbol{z}^{(1)}, \dots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  - Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
  - Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right)\right].$$

  end for

  - Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \dots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  - Update the generator by descending its stochastic gradient:

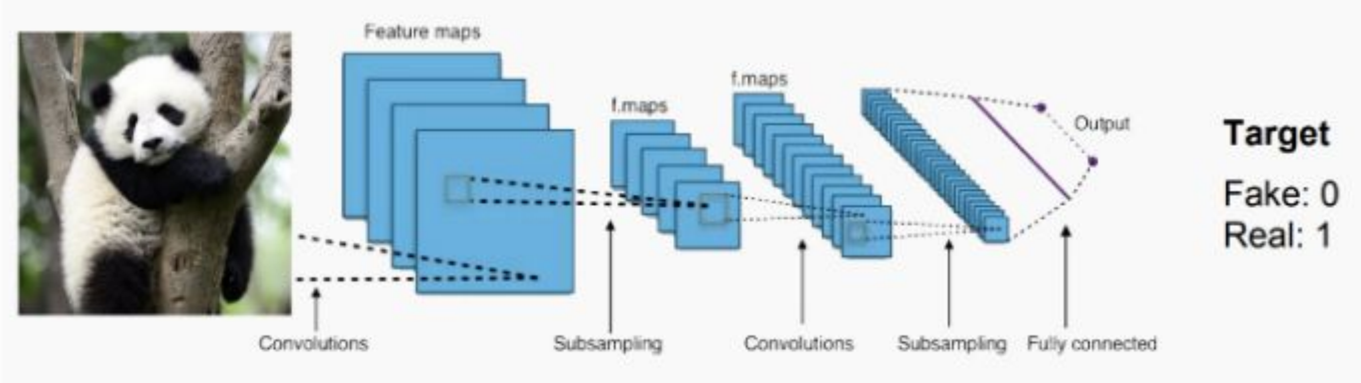
$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

end for

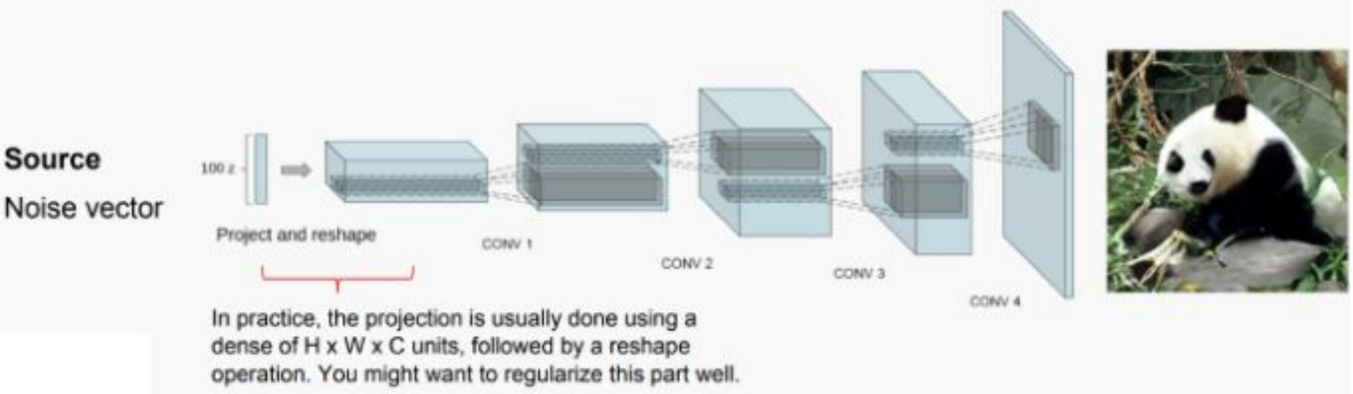
The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

GANs Architectures: Basics

Discriminator



Generator



GANs Architectures: Advanced Considerations (DCGANs)

< itc >

- **Replace Pooling with Strided convolutions** (D learns its own spatial downsampling and G its respective upsampling without adding any bias)
- **Use BatchNorm** (stabilizes learning by normalizing the input, creates more robust deep models without having the gradients diverging)
- **Avoid using Fully-Connected hidden layers** (hurts convergence speed)
- **G: use ReLU and Tanh for the output.** (tanh is preferred activation for images as an output as it has a range of [-1, 1])
- **D: use LeakyReLU** (tested empirically, works well for modelling to a higher resolution)

GANs Evolution and Flavours

< itc >

GANs keeps improving!!

GAN's Jungle is thriving!!

Reference: The GAN Zoo on GitHub - Hindu Puravinash ~ 300 GAN Papers!!!! Mostly GAN variations

- 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-RecGAN - 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
- acGAN - Face Aging With Conditional Generative Adversarial Networks
- ACtuAL - ACtuAL: Actor-Critic Under Adversarial Learning
- AdaGAN - AdaGAN: Boosting Generative Models
- AdvGAN - Generating adversarial examples with adversarial networks
- AE-GAN - AE-GAN: adversarial eliminating with GAN
- 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 (github)
- AlignGAN - AlignGAN: Learning to Align Cross-Domain Images with Conditional Generative Adversarial Networks
- AM-GAN - Activation Maximization Generative Adversarial Nets
- AnoGAN - Unsupervised Anomaly Detection with Generative Adversarial Networks to Guide Marker Discovery
- 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 Categorial GANs
- AttGAN - Arbitrary Facial Attribute Editing: Only Change What You Want
- AttnGAN - AttnGAN: Fine-Grained Text to Image Generation with Attentional Generative Adversarial Networks

- TV-GAN - TV-GAN: Generative Adversarial Network Based Thermal to Visible Face Recognition
- UGACH - Unsupervised Generative Adversarial Cross-modal Hashing
- UGAN - Enhancing Underwater Imagery using Generative Adversarial Networks
- Unim2im - Unsupervised Image-to-Image Translation with Generative Adversarial Networks (github)
- Unrolled GAN - Unrolled Generative Adversarial Networks (github)
- VAE-GAN - Autoencoding beyond pixels using a learned similarity metric
- VariGAN - Multi-View Image Generation from a Single-View
- VAW-GAN - Voice Conversion from Unaligned Corpora using Variational Autoencoding Wasserstein Generative Adversarial Networks
- VEEGAN - VEEGAN: Reducing Mode Collapse in GANs using Implicit Variational Learning (github)
- VGAN - Generating Videos with Scene Dynamics (github)
- VGAN - Generative Adversarial Networks as Variational Training of Energy Based Models (github)
- VGAN - Text Generation Based on Generative Adversarial Nets with Latent Variable
- ViGAN - Image Generation and Editing with Variational Info Generative Adversarial Networks
- VIGAN - VIGAN: Missing View Imputation with Generative Adversarial Networks
- VoiceGAN - Voice Impersonation using Generative Adversarial Networks
- VRAL - Variance Regularizing Adversarial Learning
- WaterGAN - WaterGAN: Unsupervised Generative Network to Enable Real-time Color Correction of Monocular Underwater Images
- WaveGAN - Synthesizing Audio with Generative Adversarial Networks
- weGAN - Generative Adversarial Nets for Multiple Text Corpora
- WGAN - Wasserstein GAN (github)
- WGAN-GP - Improved Training of Wasserstein GANs (github)
- WS-GAN - Weakly Supervised Generative Adversarial Networks for 3D Reconstruction
- XGAN - XGAN: Unsupervised Image-to-Image Translation for many-to-many Mappings
- ZipNet-GAN - ZipNet-GAN: Inferring Fine-grained Mobile Traffic Patterns via a Generative Adversarial Neural Network
- α-GAN - Variational Approaches for Auto-Encoding Generative Adversarial Networks (github)
- Δ-GAN - Triangle Generative Adversarial Networks

Famous GANs Types: An essence

< itc >

- **Conditional GANs:** generate images given their features (labels)
  - Use extra label information and are able to control how generated images will look.
  - Learn to produce better images by exploiting the information fed to the model

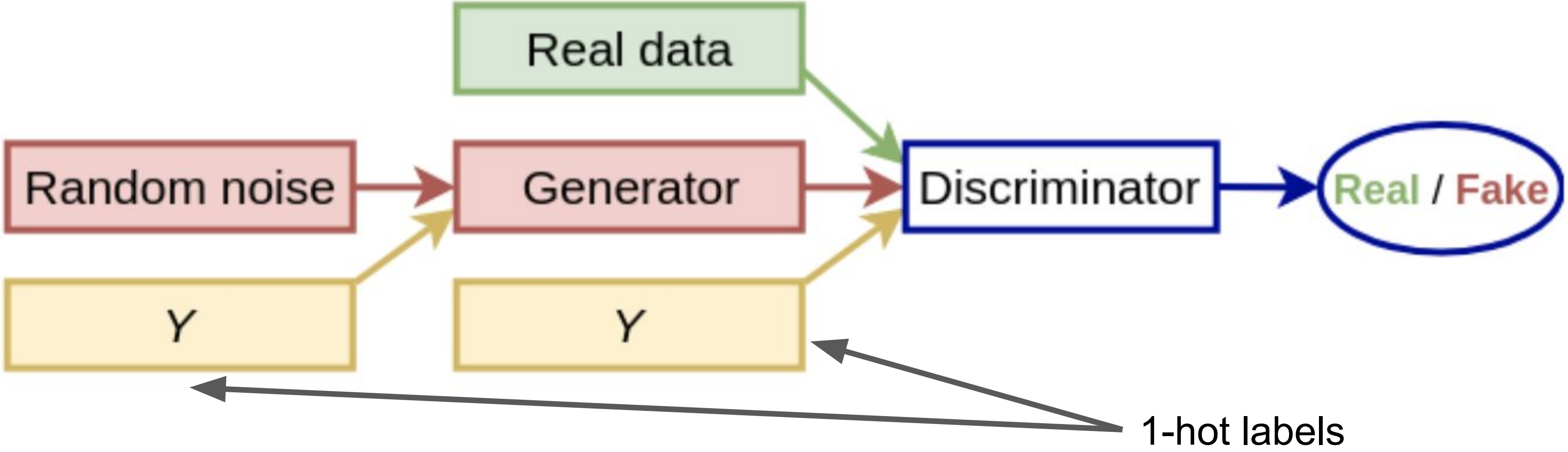
- **Info GAN:** information-theoretic extension.
  - Learn disentangled representations in an unsupervised manner.
  - Used for very complex datasets
  - Train cGAN for unlabeled datasets to extract the most important images feature

- **Cycle GAN:** unpaired image to image translation.
  - Transforms an image from one domain (zebras images), to another (horses images)
  - Other image features(not directly related to either domain as background) stays recognizably the same

Conditional Gans

Discriminator: similar to a standard Discriminator except for the 1-hot vector, which is used to condition Discriminator outputs

Generator: similar to a standard Generator except for the 1-hot vector, which is used to condition Generator outputs



Concept: Disentangled vs Entangled Latent Space < itc >

- The only way to turn" to change the generator output is the noise input
- Since it's noise, there's no intuition about how to modify it to get a desired effect
- What if you wanted an image of a man with glasses — how do you **change the noise**?
- **Impossible** as your representation is *entangled.* InfoGAN tries to solve this problem and provide a *disentangled representation*.

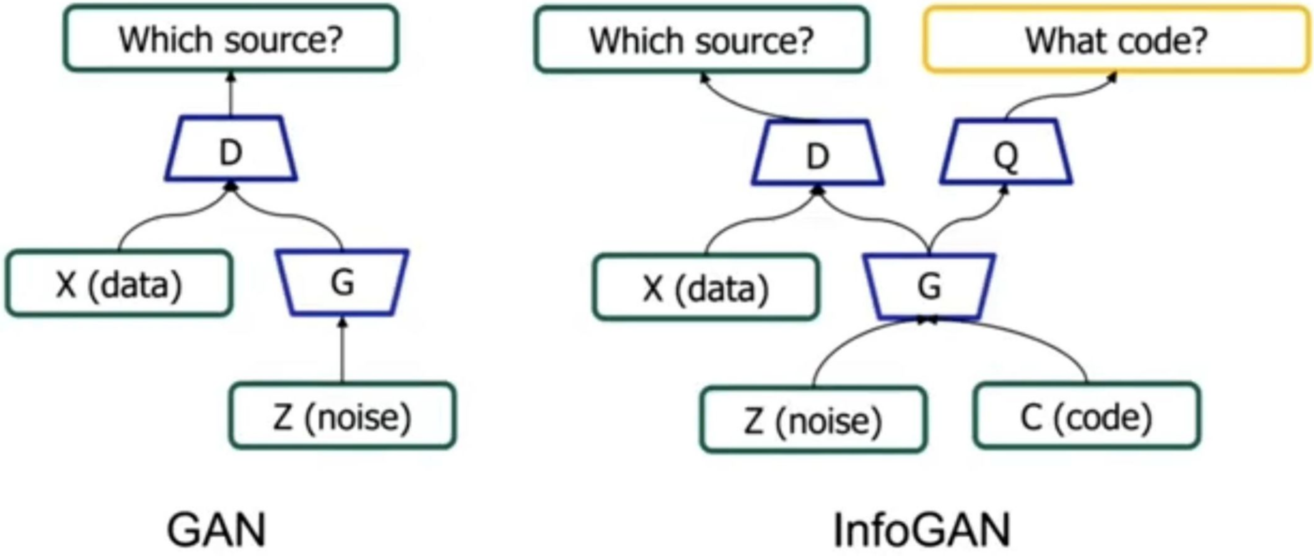
Entangled

Disentangled

InfoGAN: meaningful latent code/consistent effects on the output

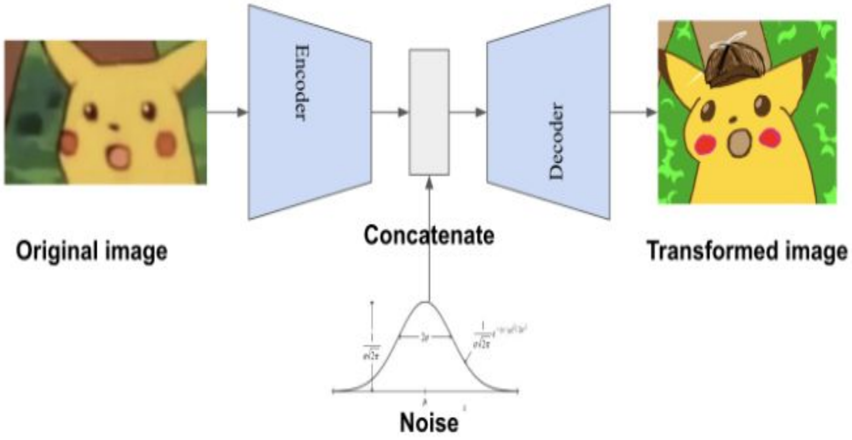
Split Generator input into 2 parts:

- Traditional noise vector
- New “latent code” vector. The codes are then made meaningful by maximizing the Mutual Information between the code and the generator output



Cool GAN types: DaGAN(data augmentation)

- Generates a synthetic image using a lower-dim representation of a real image(autoencoder style)
- Takes an existing image, encodes it, adds noise, and decodes it.
- Decoder learns a large family of transformations for data augmentation.



The DAGAN generator

The DAGAN discriminator

Cool GAN types: ProGAN (progressive)

- Grows generator and discriminator progressively:
  - Starts from a low resolution, adds new layers modeling increasingly finer details while training progresses
  - Capable to generate high-quality images compared to its predecessors

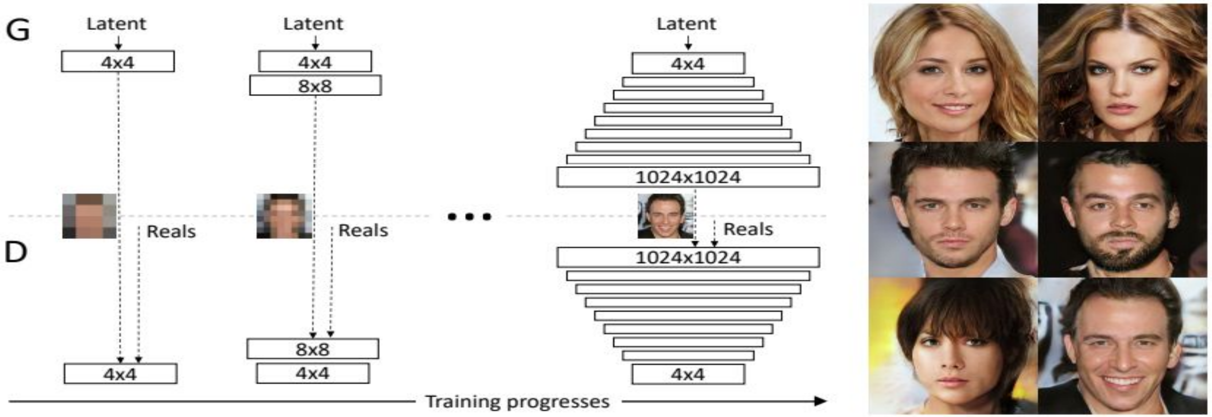
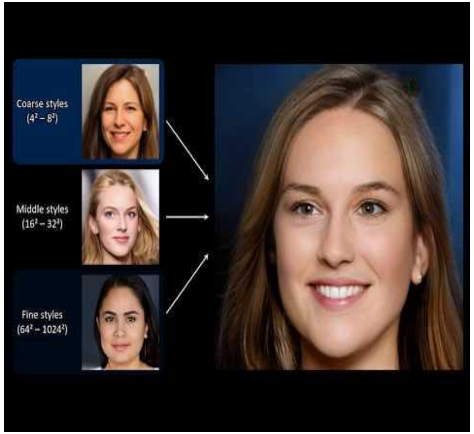
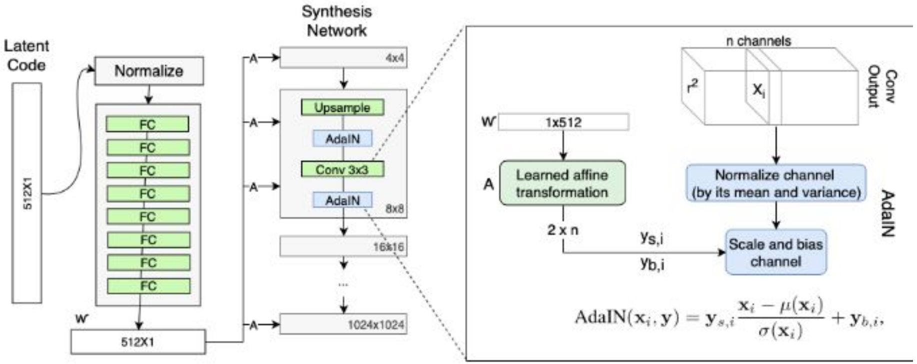


Figure 1: Our training starts with both the generator (G) and discriminator (D) having a low spatial resolution of 4×4 pixels. As the training advances, we incrementally add layers to G and D, thus increasing the spatial resolution of the generated images. All existing layers remain trainable

Cool GAN types: StyleGAN

- Control visual features of the generated images, such as hair color, pose, nose form etc
- Exploits ProGAN progressive layers' ability to control image visual features, if utilized properly
- The lower the layer (the resolution), the coarser the features it affects
  - Coarse - affects pose, general hair style, face shape, etc
  - Middle - affects finer facial features, hair style, eyes open/closed, etc.
  - Fine - affects color scheme (eye, hair and skin) and micro features.





The generator's Adaptive Instance Normalization (AdaIN)

GANs shortcomings(1)

< itc >

- **Sensitivity** to both structure and parameters
  - If either the discriminator or generator gets better than the other too quickly, the other will never be able to catch up
  - Finding the right combination can be very challenging

- **Convergence -** No proof that GANs converge.
  - GANs perform well with the right parameters, but there's no guarantee beyond that.
  - The more complicated network gets, the tougher convergence becomes, the more difficult hyperparameter selection becomes

GANs shortcomings(2)

- **Generation of high-resolution big size images** (partially solved by ProGANs, CycleGANs but still…)
    - It’s easy for “D” to tell the generated fakes from the real images
    - Many pixels can lead to error gradients that cause the Gs output to move in almost random directions

- **Mode(modal) Collapse:** “G” produces the same image every time independently of input noise (InfoGAN solves this issue partially)
    - When training our network, the “G” somehow finds one image that fools the “D”
    - “D” will always say it is real, so the “G” has accomplished its goal and stops learning.
    - However, the problem is that **every sample made by the generator is identical**.

Further readings about GANs

< itc >

- Brief introduction to GANs
- Brief Introduction to GAN 2
- Comprehensive GAN analysis with code
- Generating dogs images with GANs(tensorflow)
- How to Measure GAN Performance
- GANs Flaws
- GANs type (with some a bit advanced math)
- StyleGAN: thorough analysis and examples
- GANs for Data Augmentation

Ask the lecturer for more cool GAN Stuff