GANs and their species

IE643 - Lectures 19, 20

October 15 & 19, 2024.

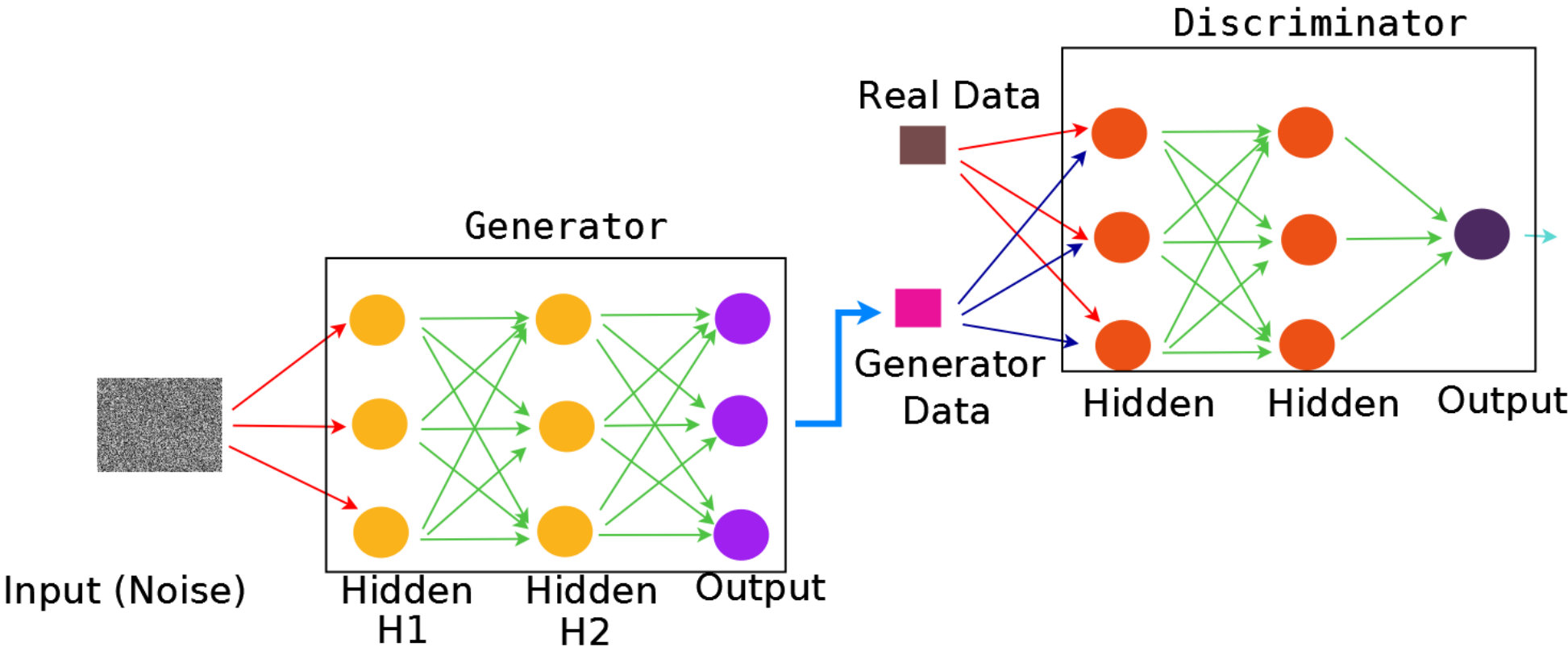
1. GAN
   - Training Strategies
   - Conditional GAN
2. Disco GAN
3. Style GAN

GAN

Generative Adversarial Nets

Ian J. Goodfellow, Jean Pouget-Abadie*, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair†, Aaron Courville, Yoshua Bengio‡

GAN


Discriminator
Real Data
Generator
Data
Hidden
Hidden
Output
Generator
Input (Noise)
Hidden
H1
Hidden
H2
Output

GAN

- Discriminator component D
- Generator component G

GAN

- Aim of Discriminator component D:
  - Maximize likelihood of real data from distribution p_{data}
  - Minimize likelihood of fake data $G(z)$ obtained through generator G, where z comes from noise distribution p_{noise}.

GAN

- Aim of Discriminator component D:
  - Maximize likelihood of real data from distribution p_{data}
  - Minimize likelihood of fake data $G(z)$ obtained through generator G, where z comes from noise distribution p_{noise}.
- This is equivalent to
  - Maximize **log** likelihood of real data from distribution p_{data}
  - Minimize **log** likelihood of fake data $G(z)$ obtained through generator G, where z comes from noise distribution p_{noise}.

GAN

- Aim of Discriminator component D:
  - Maximize log likelihood of real data from distribution p_{data}:
    - ★ This is represented as

$$\max \ \mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x).$$

    - ★ Here x denotes real data samples from distribution p_{data}.
    - ★ θ_d represents the parameters of the discriminator D.
    - ★ $D_{\theta_d}(x)$ denotes the output from discriminator parametrized by θ_d, when input to discriminator is x.
    - ★ **Assumption:** $D_{\theta_d}(x)$ denotes a probability or likelihood.

GAN

- Aim of Discriminator component D:
  - Maximize log likelihood of real data from distribution p_{data}:
    - ★ This is represented as

$$\max \ \mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x).$$

  - Minimize likelihood of fake data $G(z)$ obtained through generator G, where z comes from noise distribution p_{noise}.
    - ★ This is represented as:

$$\min \ \mathbb{E}_{z \sim p_{noise}} \log D_{\theta_d}(G(z)).$$

    - ★ Here z denotes noise from distribution p_{noise}.
    - ★ $G(z)$ denotes the generator's output when the noise z is passed through the generator.
    - ★ $D_{\theta_d}(G(z))$ denotes the discriminator's output when the generator's output $G(z)$ considered as a fake sample is passed through the discriminator.
    - ★ **Recall:** $D_{\theta_d}(\cdot)$ denotes a probability or likelihood.

GAN

- Aim of Discriminator component D:
  - Maximize log likelihood of real data from distribution p_{data}:
    - ★ This is represented as

$$\max \ \mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x).$$

  - Minimize log likelihood of fake data $G(z)$ obtained through generator G, where z comes from noise distribution p_{noise}.
    - ★ This is represented as:

$$\min \ \mathbb{E}_{z \sim p_{noise}} \log D_{\theta_d}(G(z)).$$

GAN

- Objective of Discriminator component D: (straightforward)

$$\max_{\theta_d} \; \mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p_{noise}} \log(1 - D_{\theta_d}(G(z))).$$

Assumption: $D(x)$ is a probability (or likelihood)

GAN

- Aim of Generator component G:
  - Fool the discriminator by providing $G(z)$ as if it comes from p_{data}.
  - How to do that?

GAN

- Aim of Generator component G:
  - Fool the discriminator by providing $G(z)$ as if it comes from p_{data}.
  - How to do that?
- Idea:

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

GAN

- Overall Objective of GAN:

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

GAN

- Overall Objective of GAN (seems to be):

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

- Looks like a max-min problem

GAN

- Overall Objective of GAN (used in paper):

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

- This is a min-max problem

GAN

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 $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ 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 $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ 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.

GAN

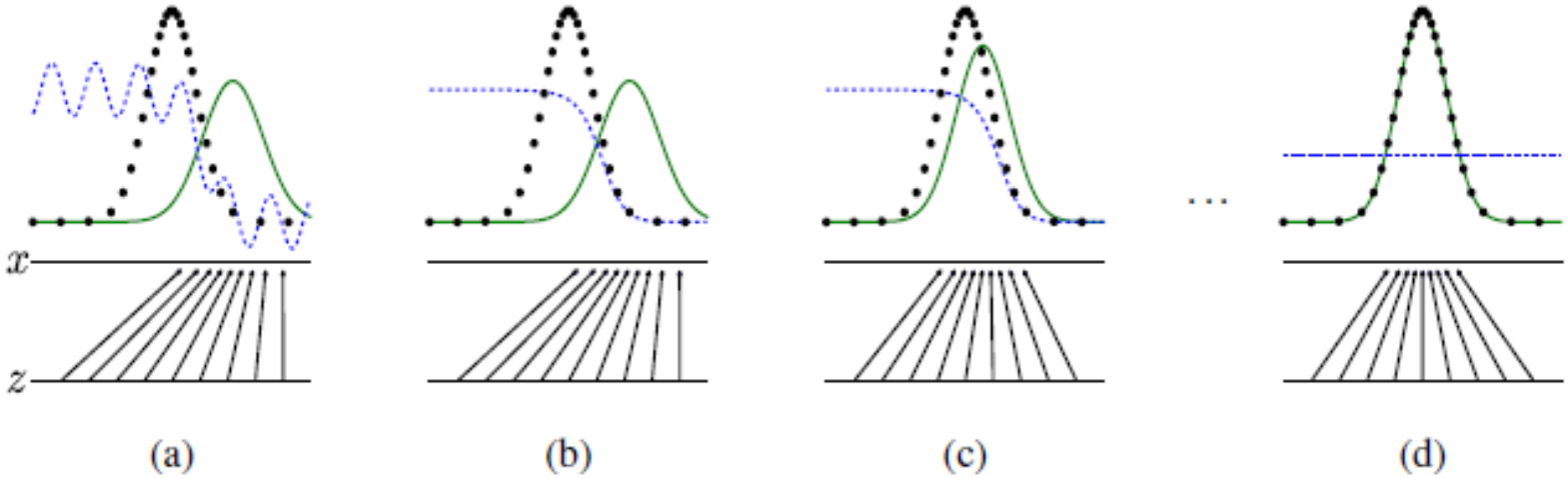


Figure 1: Generative adversarial nets are trained by simultaneously updating the discriminative distribution (D, blue, dashed line) so that it discriminates between samples from the data generating distribution (black, dotted line) p_x from those of the generative distribution p_g (G) (green, solid line). The lower horizontal line is the domain from which z is sampled, in this case uniformly. The horizontal line above is part of the domain of x. The upward arrows show how the mapping $x = G(z)$ imposes the non-uniform distribution p_g on transformed samples. G contracts in regions of high density and expands in regions of low density of p_g. (a) Consider an adversarial pair near convergence: p_g is similar to p_{data} and D is a partially accurate classifier. (b) In the inner loop of the algorithm D is trained to discriminate samples from data, converging to $D^*(x) = \frac{p_{\text{data}}(x)}{p_{\text{data}}(x)+p_g(x)}$. (c) After an update to G, gradient of D has guided $G(z)$ to flow to regions that are more likely to be classified as data. (d) After several steps of training, if G and D have enough capacity, they will reach a point at which both cannot improve because $p_g = p_{\text{data}}$. The discriminator is unable to differentiate between the two distributions, i.e. $D(x) = \frac{1}{2}$.

GAN

Figure 2: Visualization of samples from the model. Rightmost column shows the nearest training example of the neighboring sample, in order to demonstrate that the model has not memorized the training set. Samples are fair random draws, not cherry-picked. Unlike most other visualizations of deep generative models, these images show actual samples from the model distributions, not conditional means given samples of hidden units. Moreover, these samples are uncorrelated because the sampling process does not depend on Markov chain mixing. a) MNIST b) TFD c) CIFAR-10 (fully connected model) d) CIFAR-10 (convolutional discriminator and "deconvolutional" generator)

GAN

Figure 3: Digits obtained by linearly interpolating between coordinates in z space of the full model.

GAN - Mode Collapse Problem

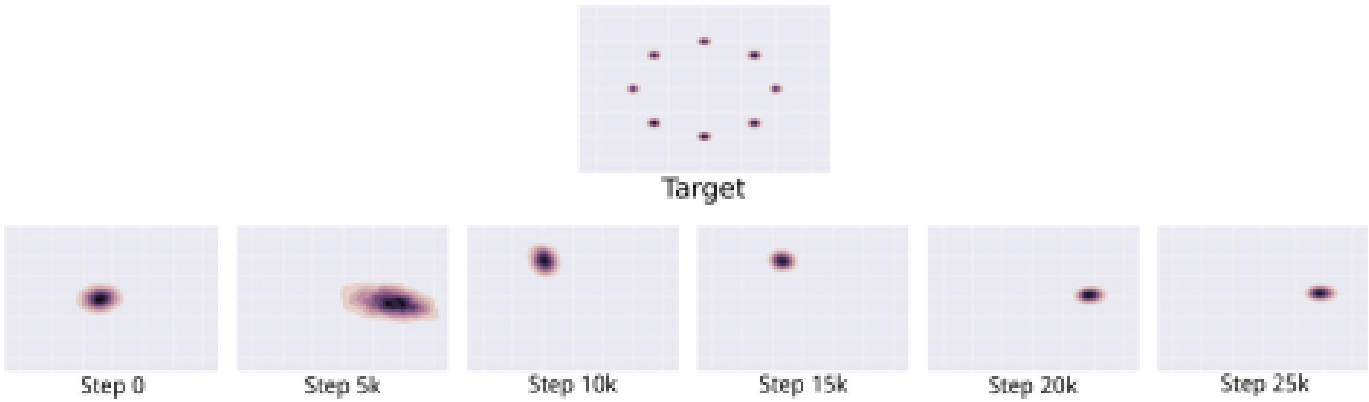


Figure 22: An illustration of the mode collapse problem on a two-dimensional toy dataset. In the top row, we see the target distribution p_{data} that the model should learn. It is a mixture of Gaussians in a two-dimensional space. In the lower row, we see a series of different distributions learned over time as the GAN is trained. Rather than converging to a distribution containing all of the modes in the training set, the generator only ever produces a single mode at a time, cycling between different modes as the discriminator learns to reject each one. Images from Metz *et al.* (2016).

GAN Training

UNSUPERVISED REPRESENTATION LEARNING WITH DEEP CONVOLUTIONAL GENERATIVE ADVERSARIAL NETWORKS

Alec Radford & Luke Metz
Soumith Chintala

DCGAN

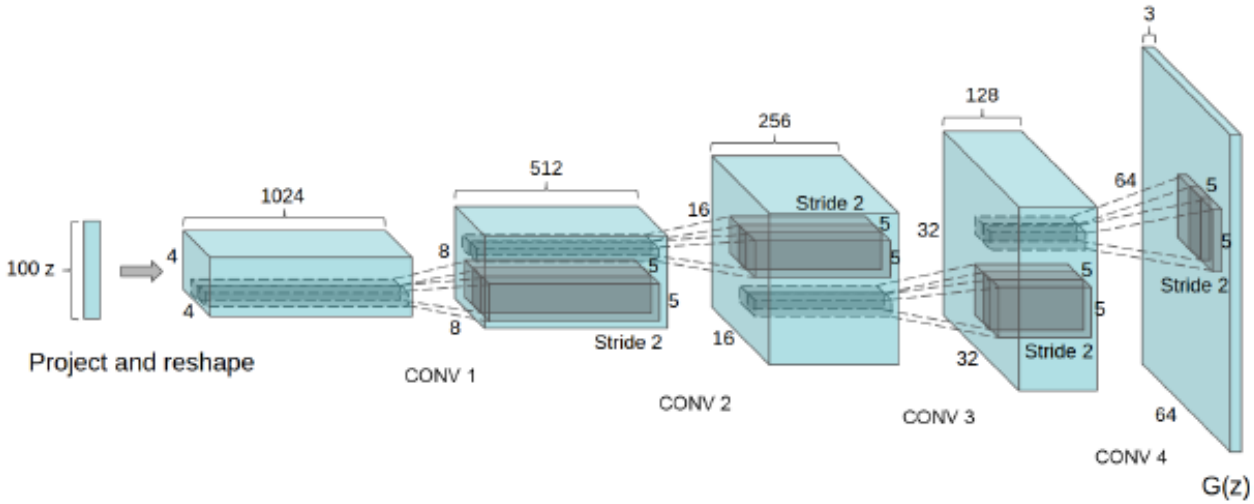


Figure 1: DCGAN generator used for LSUN scene modeling. A 100 dimensional uniform distribution Z is projected to a small spatial extent convolutional representation with many feature maps. A series of four fractionally-strided convolutions (in some recent papers, these are wrongly called deconvolutions) then convert this high level representation into a 64×64 pixel image. Notably, no fully connected or pooling layers are used.

DCGAN Training

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 Training

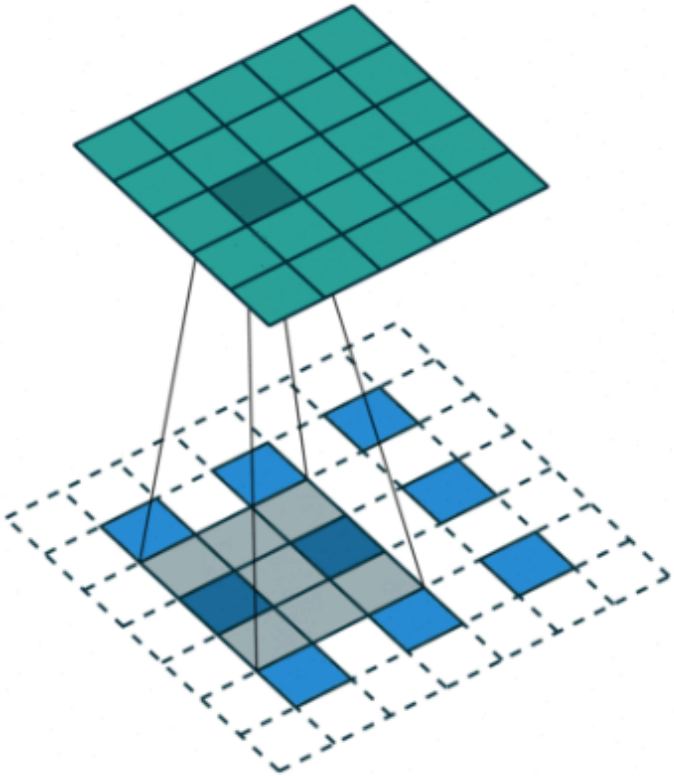
Architecture guidelines for stable Deep Convolutional GANs

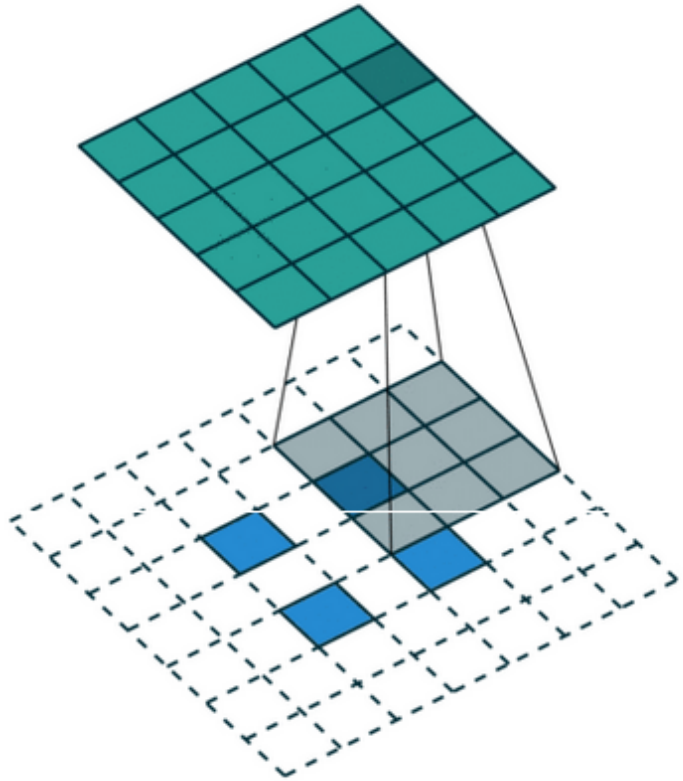
- Replace any pooling layers with strided convolutions (discriminator) and fractional-strided convolutions (generator).
- Use batchnorm in both the generator and the discriminator.
- Remove fully connected hidden layers for deeper architectures.
- Use ReLU activation in generator for all layers except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

???

DCGAN Training

Fractional Convolutions:





https://github.com/vdumoulin/conv_arithmetic

DCGAN Results

Figure 2: Generated bedrooms after one training pass through the dataset. Theoretically, the model could learn to memorize training examples, but this is experimentally unlikely as we train with a small learning rate and minibatch SGD. We are aware of no prior empirical evidence demonstrating memorization with SGD and a small learning rate.

DCGAN Results

Figure 3: Generated bedrooms after five epochs of training. There appears to be evidence of visual under-fitting via repeated noise textures across multiple samples such as the base boards of some of the beds.

DCGAN Results

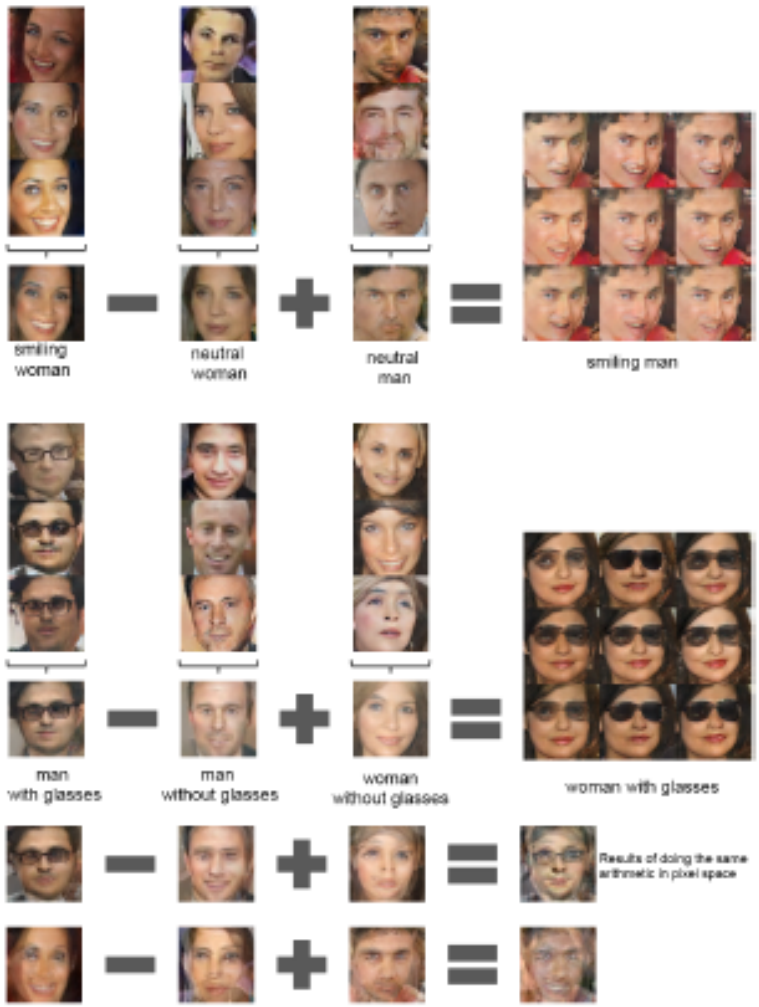


Figure 7: Vector arithmetic for visual concepts. For each column, the Z vectors of samples are averaged. Arithmetic was then performed on the mean vectors creating a new vector Y. The center sample on the right hand side is produce by feeding Y as input to the generator. To demonstrate the interpolation capabilities of the generator, uniform noise sampled with scale +-0.25 was added to Y to produce the 8 other samples. Applying arithmetic in the input space (bottom two examples) results in noisy overlap due to misalignment.

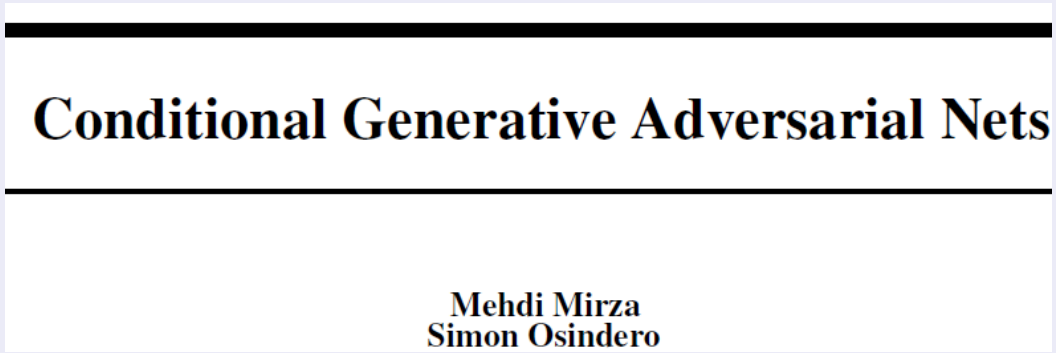
DCGAN Results

Figure 8: A "turn" vector was created from four averaged samples of faces looking left vs looking right. By adding interpolations along this axis to random samples we were able to reliably transform their pose.

DCGAN Results

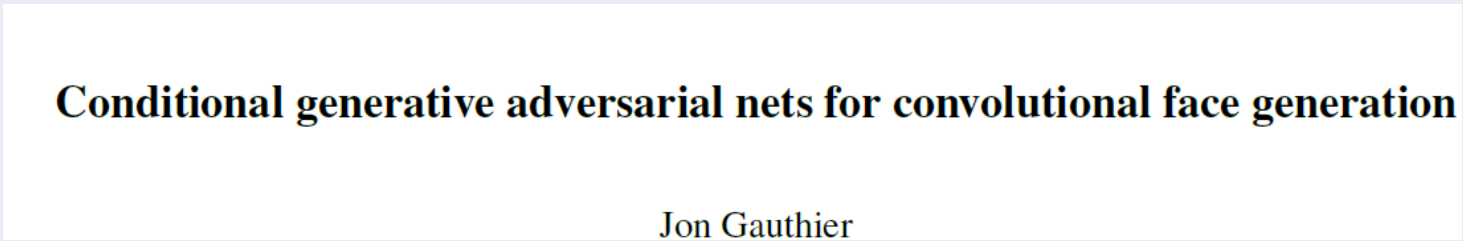
Figure 10: More face generations from our Face DCGAN.

Conditional GAN



Conditional Generative Adversarial Nets

Mehdi Mirza
Simon Osindero



Conditional generative adversarial nets for convolutional face generation

Jon Gauthier

Conditional GAN

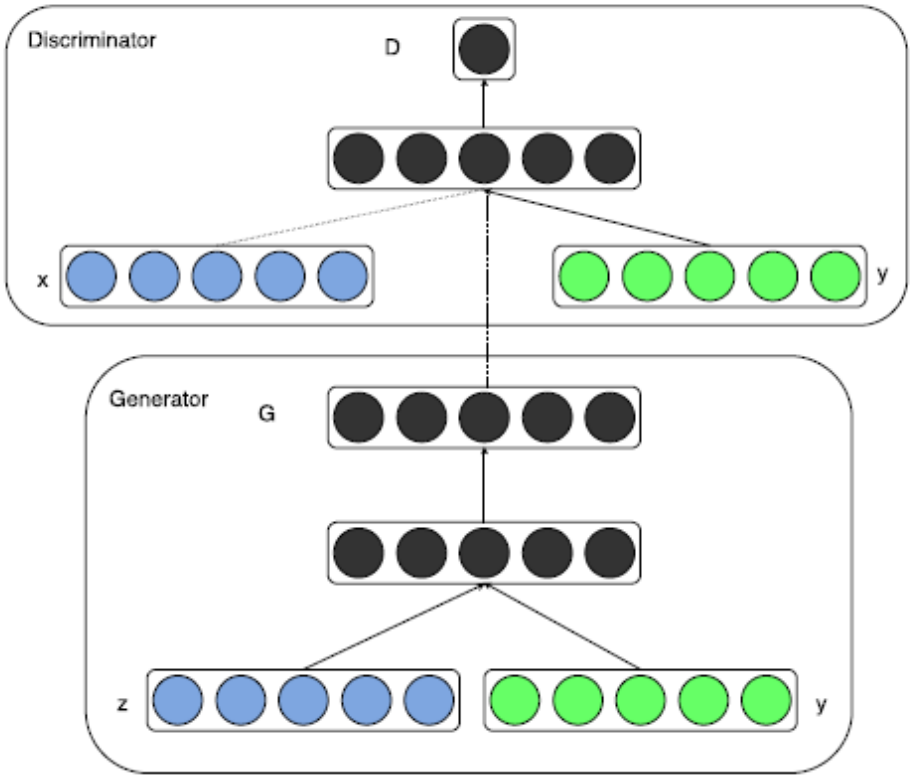
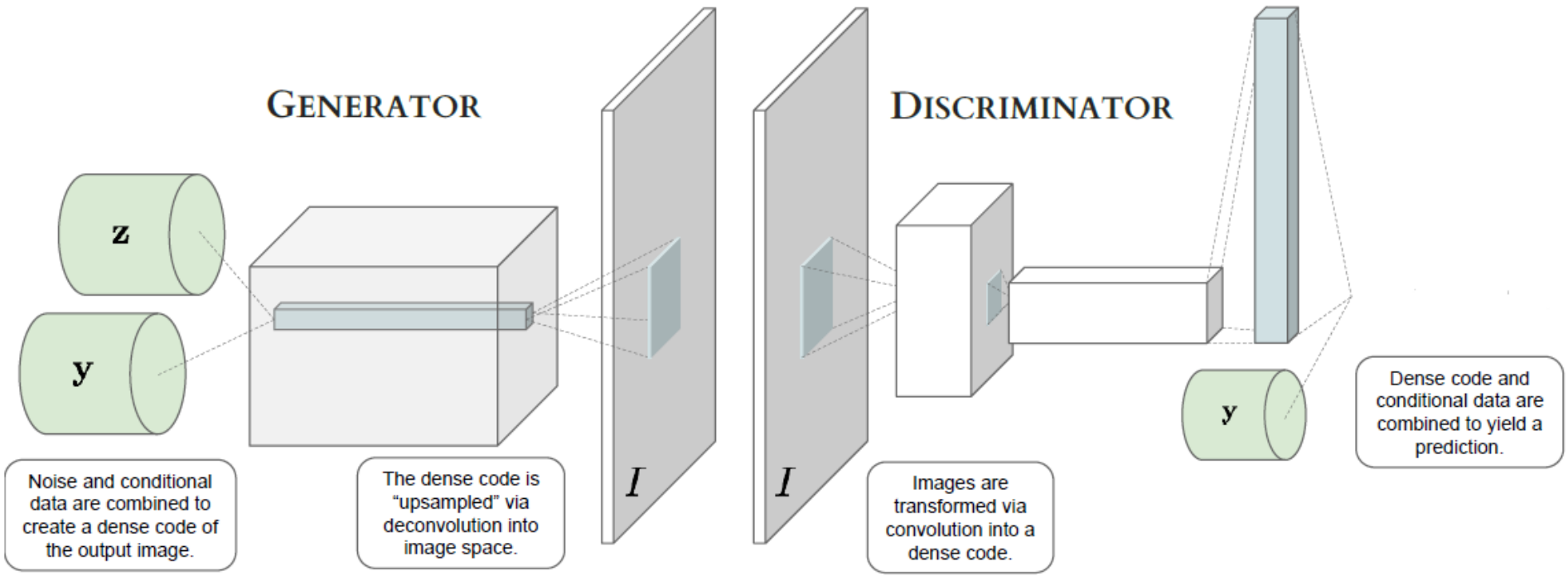


Figure 1: Conditional adversarial net

Conditional GAN



Conditional GAN

- Overall Objective of conditional GAN:

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

Conditional GAN

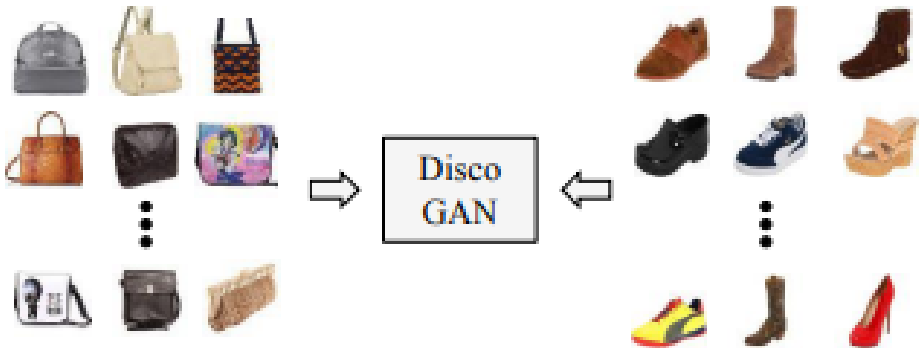
Figure 7: Results of constant additive shifts along particular axes of the conditional data $\mathbf{y}$. (Full details of this shift are given in Section 4.3.1.) Row 1: randomly sampled images from the generator. Row 2: constant additive shift along the SENIOR axis; faces visibly age in the resulting samples. Row 3: constant additive shift along the MOUTHOPEN axis; faces open mouths / smile.

Disco GAN

Learning to Discover Cross-Domain Relations with Generative Adversarial Networks

Taeksoo Kim [1] Moonsu Cha [1] Hyunsoo Kim [1] Jung Kwon Lee [1] Jiwon Kim [1]

Disco GAN



(a) Learning cross-domain relations **without any extra label**

(b) Handbag images (input) & **Generated** shoe images (output)

(c) Shoe images (input) & **Generated** handbag images (output)

Disco GAN

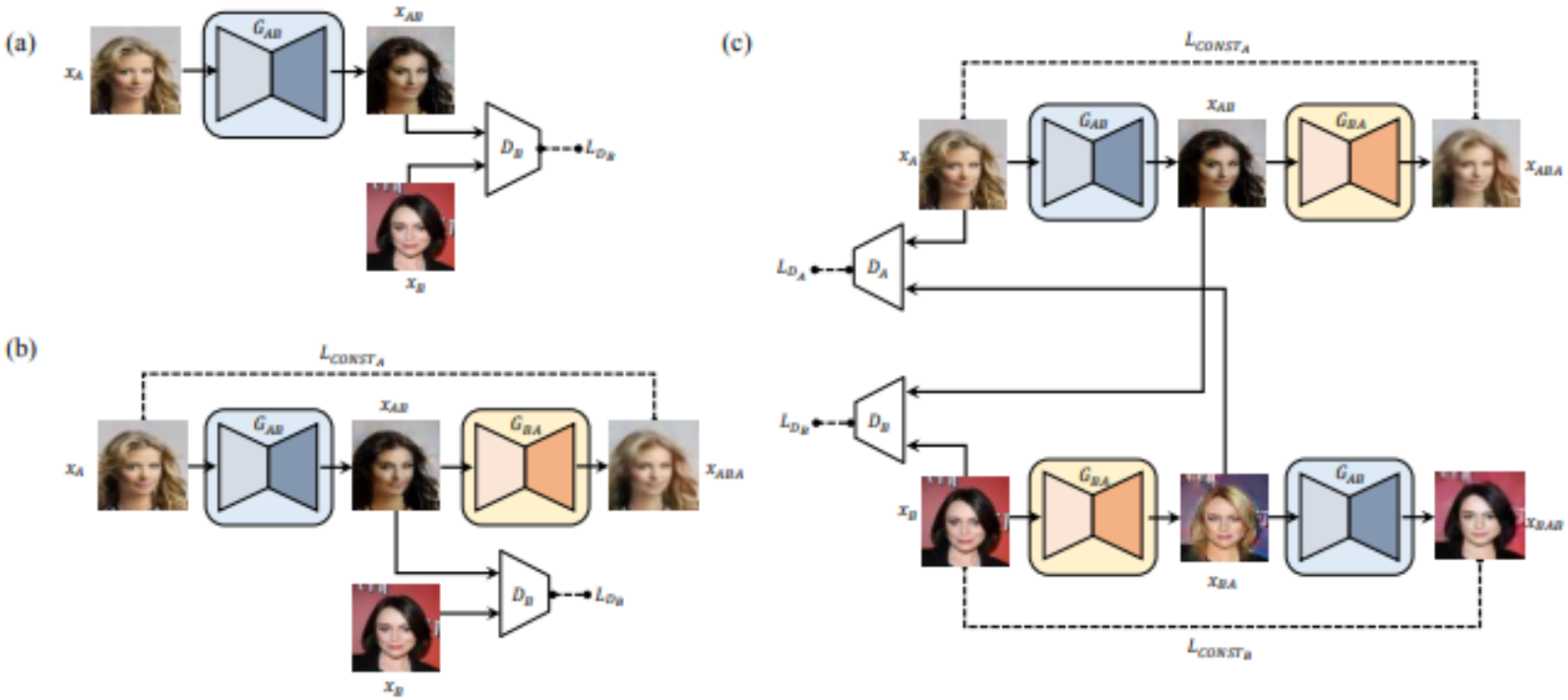
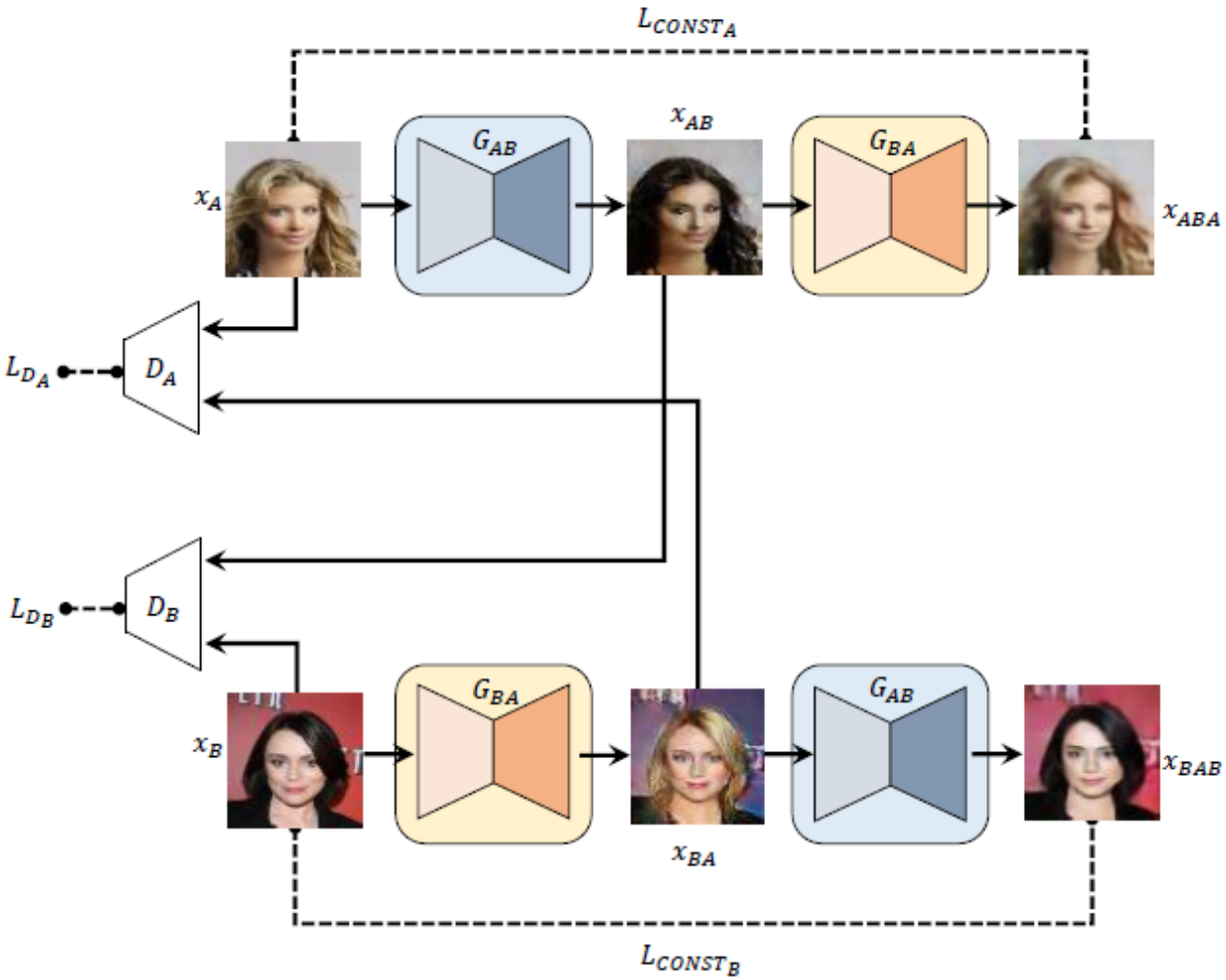


Figure 2. Three investigated models. (a) standard GAN (Goodfellow et al., 2014), (b) GAN with a reconstruction loss, (c) our **proposed model (DiscoGAN)** designed to discover relations between two unpaired, unlabeled datasets. Details are described in Section 3.

Disco GAN


L_{CONST_A}
x_A
G_{AB}
x_{AB}
G_{BA}
x_{ABA}
L_{D_A}
D_A
L_{D_B}
D_B
x_B
G_{BA}
x_{BA}
G_{AB}
x_{BAB}
L_{CONST_B}

Disco GAN - Components

- Two domains A and B
- Examples:
  - A = Vehicles B = Mesh Diagrams
  - A = Face Emotions B = Music
  - A = Adult animals B = Cubs
  - A = Actors B = Videos
- Two generators G_{AB} and G_{BA}
- Two discriminators D_A and D_B

Disco GAN - Components

- Generator G_{AB}
  - Randomly sample x_A from A
  - Form $x_{AB} = G_{AB}(x_A)$
  - Get back $x_{ABA} = G_{BA}(x_{AB}) = G_{BA}(G_{AB}(x_A))$
  - Reconstruction loss: $R_{G_{AB}} = \mathbb{E}_{x_A \sim A} d(x_A, x_{ABA})$
  - Discriminator loss: $GAN_{G_{AB}} = \mathbb{E}_{x_A \sim A} \log[1 - D_B(G_{AB}(x_A))]$
  - Total loss: $\min L_{G_{AB}} = R_{G_{AB}} + GAN_{G_{AB}}$

Disco GAN - Components

- Generator G_{AB}
  - Randomly sample x_A from A
  - Form $x_{AB} = G_{AB}(x_A)$
  - Get back $x_{ABA} = G_{BA}(x_{AB}) = G_{BA}(G_{AB}(x_A))$
  - Reconstruction loss: $R_{G_{AB}} = \mathbb{E}_{x_A \sim A} d(x_A, x_{ABA})$
  - Discriminator loss: $GAN_{G_{AB}} = \mathbb{E}_{x_A \sim A} \log[1 - D_B(G_{AB}(x_A))]$
  - Total loss: $\min L_{G_{AB}} = R_{G_{AB}} + GAN_{G_{AB}}$
- Generator G_{BA}
  - Randomly sample x_B from B
  - Form $x_{BA} = G_{BA}(x_B)$
  - Get back $x_{BAB} = G_{AB}(x_{BA}) = G_{AB}(G_{BA}(x_B))$
  - Reconstruction loss: $R_{G_{BA}} = \mathbb{E}_{x_B \sim B} d(x_B, x_{BAB})$
  - Discriminator loss: $GAN_{G_{BA}} = \mathbb{E}_{x_B \sim B} \log[1 - D_A(G_{BA}(x_B))]$
  - Total loss: $\min L_{G_{BA}} = R_{G_{BA}} + GAN_{G_{BA}}$

Disco GAN - Components

- Discriminator D_A
  - Randomly sample x_A from A
  - Compute $L_{D_A}^{real} = \mathbb{E}_{x_A \sim A} \log D_A(x_A)$
  - Obtain input from generator G_{BA}
  - Compute $L_{D_A}^{gen} = \mathbb{E}_{x_A \sim A} \log[1 - D_A(G_{BA}(x_B))]$
  - Total loss: $\max L_{D_A} = L_{D_A}^{real} + L_{D_A}^{gen}$.
- Discriminator D_B
  - Randomly sample x_B from B
  - Compute $L_{D_B}^{real} = \mathbb{E}_{x_B \sim B} \log D_B(x_B)$
  - Obtain input from generator G_{AB}
  - Compute $L_{D_B}^{gen} = \mathbb{E}_{x_B \sim B} \log[1 - D_B(G_{AB}(x_A))]$
  - Total loss: $\max L_{D_B} = L_{D_B}^{real} + L_{D_B}^{gen}$.

Disco GAN - Components

- Total Generator loss: $L_{Gen} = \min_{\theta_{gen}} [L_{G_{AB}} + L_{G_{BA}}]$
- Total Discriminator loss: $L_{Disc} = \max_{\theta_{disc}} [L_{D_A} + L_{D_B}]$

Disco GAN

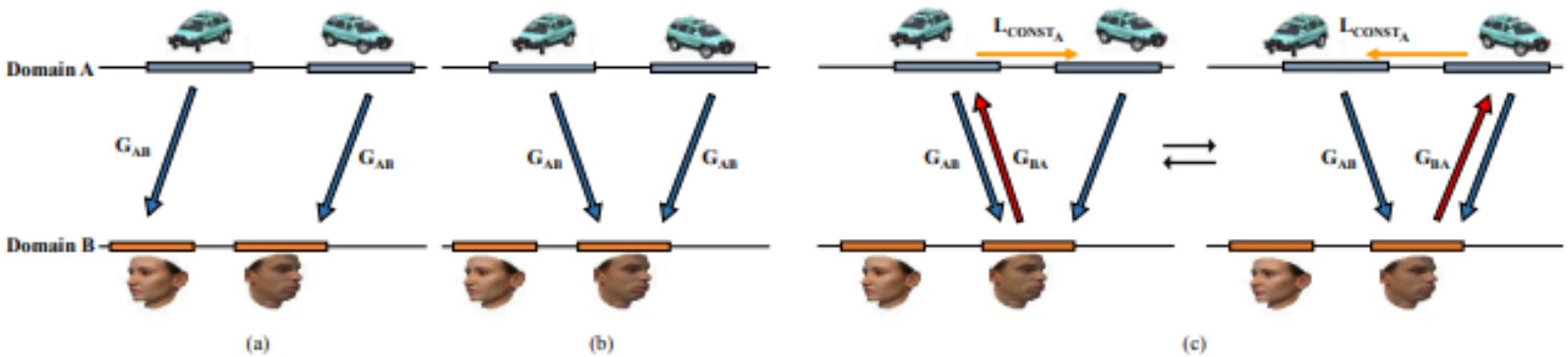


Figure 3. Illustration of our models on simplified one dimensional domains. (a) ideal mapping from domain A to domain B in which the two domain A modes map to two different domain B modes, (b) GAN model failure case, (c) GAN with reconstruction model failure case.

Disco GAN

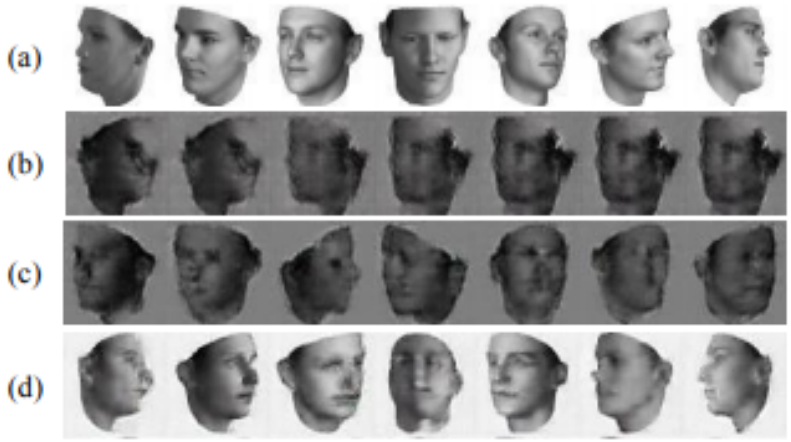
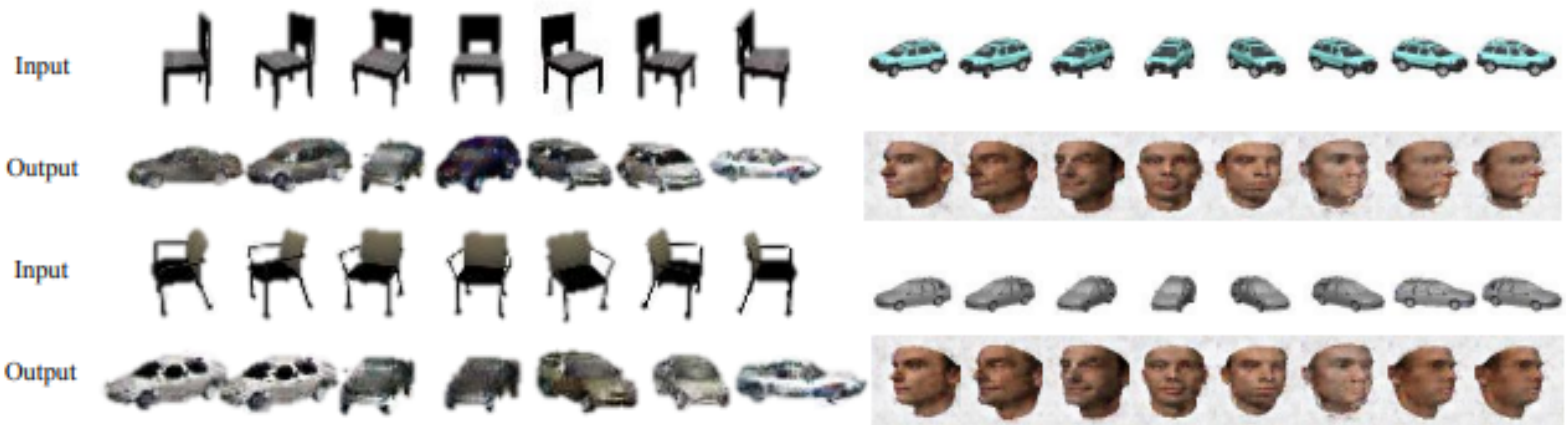


Figure 6. Face to Face translation experiment. (a) input face images from -90° to +90° (b) results from a standard GAN (c) results from GAN with a reconstruction loss (d) results from our DiscoGAN. Here our model generated images in the opposite range, from +90° to -90°.

Disco GAN

Figure 7. (a,b) Translation of gender in Facescrub dataset and CelebA dataset. (c) Blond to black and black to blond hair color conversion in CelebA dataset. (d) Wearing eyeglasses conversion in CelebA dataset (e) Results of applying a sequence of conversion of gender and hair color (left to right) (f) Results of repeatedly applying the same conversions (upper: hair color, lower: gender)

Disco GAN



(a) Chair to Car

(b) Car to Face

Figure 8. Discovering relations of images from visually very different object classes. (a) chair to car translation. DiscoGAN is trained on chair and car images (b) car to face translation. DiscoGAN is trained on car and face images. Our model successfully pairs images with similar orientation.

Disco GAN

Figure 9. Edges to photos experiment. Our model is trained on a set of object sketches and colored images and learns to generate new sketches or photos. (a) colored images of handbags are generated from sketches of handbags, (b) colored images of shoes are generated from sketches of shoes, (c) sketches of handbags are generated from colored images of handbags

Style GAN

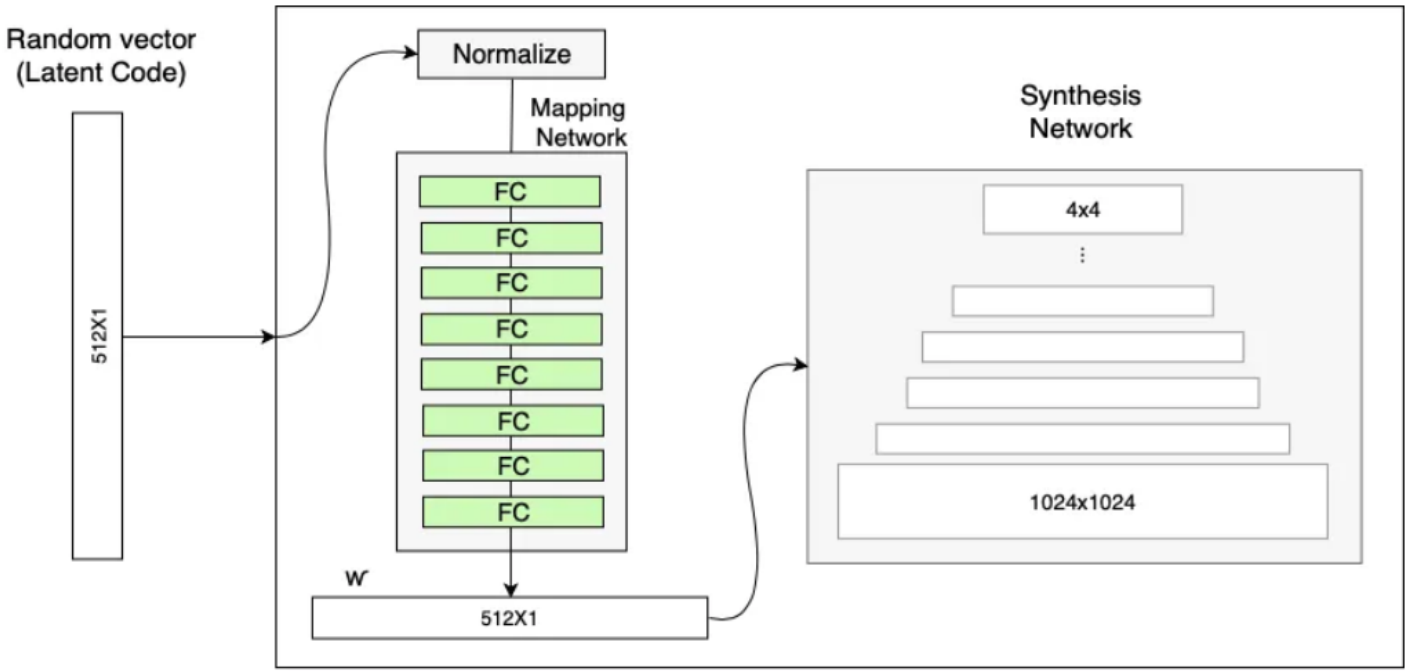
A Style-Based Generator Architecture for Generative Adversarial Networks

Tero Karras
NVIDIA
tkarras@nvidia.com

Samuli Laine
NVIDIA
slaine@nvidia.com

Timo Aila
NVIDIA
taila@nvidia.com

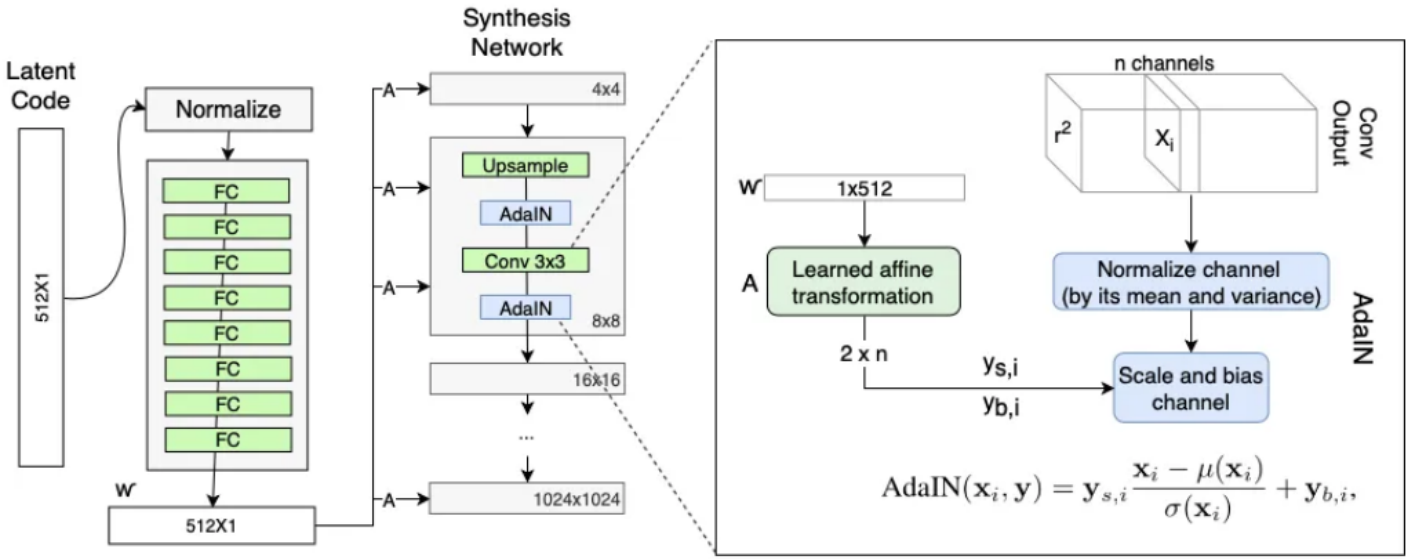
Style GAN



The generator with the Mapping Network (in addition to the ProGAN synthesis network)

Image Source: https://towardsdatascience.com/

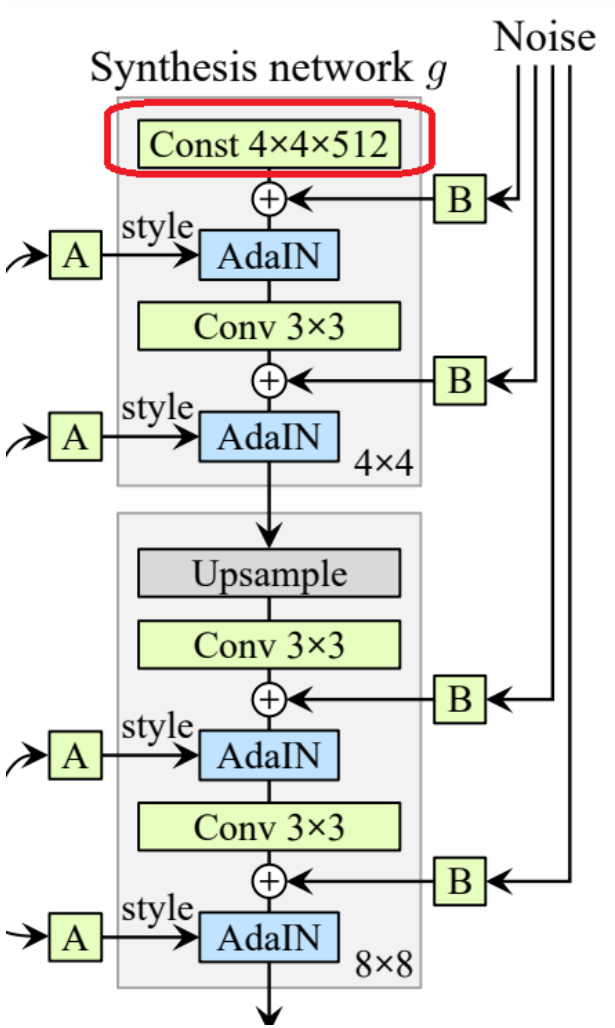
Style GAN



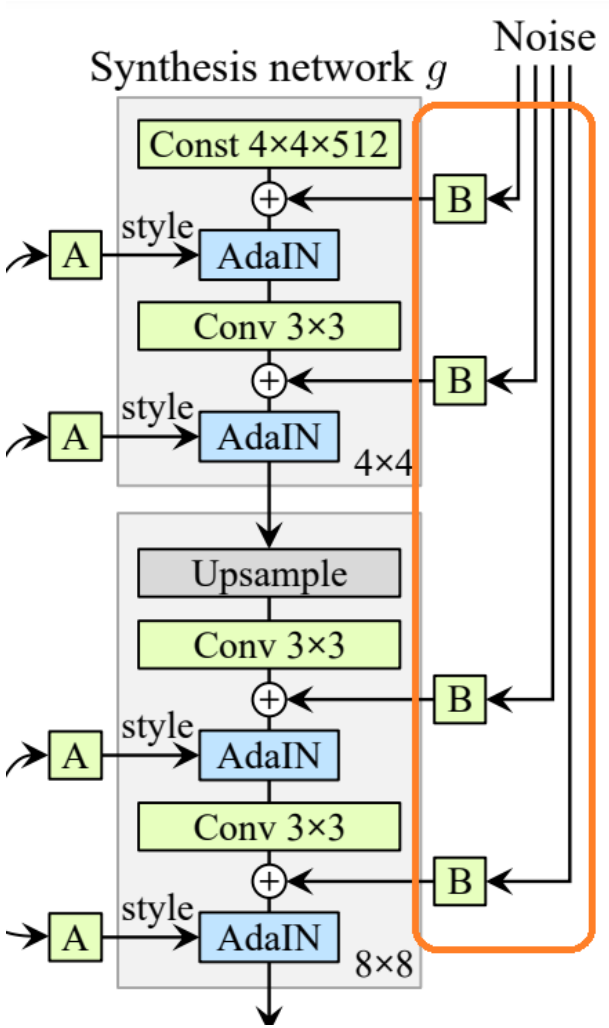
The generator's Adaptive Instance Normalization (AdaIN)

Image Source: https://towardsdatascience.com/

Style GAN


Noise
Synthesis network g
Const 4×4×512
B
style
A
AdaIN
Conv 3×3
B
style
A
AdaIN
4×4
Upsample
Conv 3×3
B
style
A
AdaIN
Conv 3×3
B
style
A
AdaIN
8×8

Style GAN


Noise
Synthesis network g
Const 4×4×512
B
style
A
AdaIN
Conv 3×3
B
style
A
AdaIN
4×4
Upsample
Conv 3×3
B
style
A
AdaIN
Conv 3×3
B
style
A
AdaIN
8×8

Style GAN

(a) Generated image (b) Stochastic variation

Figure 4. Examples of stochastic variation. (a) Two generated images. (b) Zoom-in with different realizations of input noise. While the overall appearance is almost identical, individual hairs are placed very differently.

Style GAN

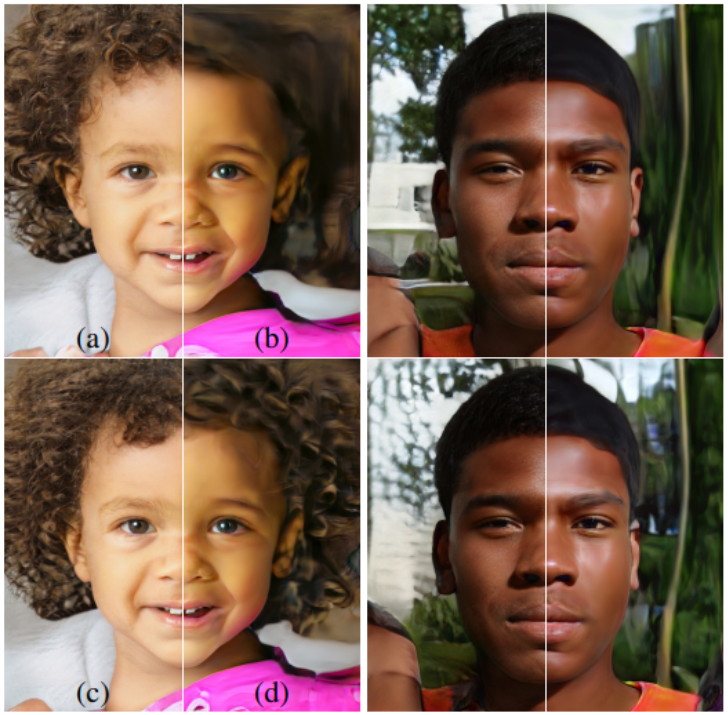
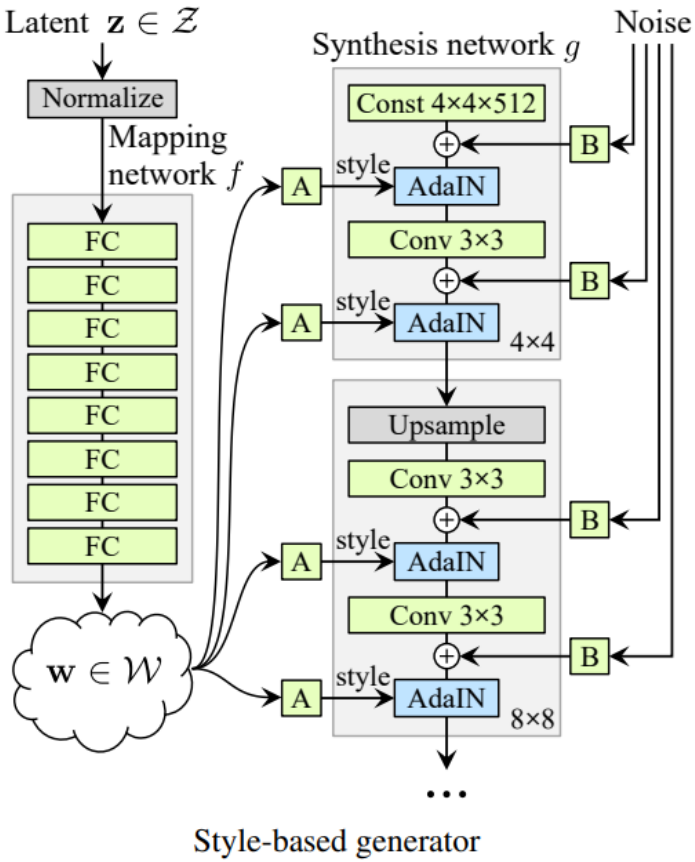


Figure 5. Effect of noise inputs at different layers of our generator. (a) Noise is applied to all layers. (b) No noise. (c) Noise in fine layers only (64^2 – 1024^2). (d) Noise in coarse layers only (4^2 – 32^2). We can see that the artificial omission of noise leads to featureless "painterly" look. Coarse noise causes large-scale curling of hair and appearance of larger background features, while the fine noise brings out the finer curls of hair, finer background detail, and skin pores.

Style GAN



Style-based generator

Style GAN

Other techniques tried in the paper

- Mixing multiple latent vectors during training (style mixing)
  - Sample z_1 and z_2 and obtain corresponding w_1 and w_2 from mapping network $\mathcal{W}$.
  - During the adaIn operations, w_1 is considered as inputs for some initial layers and w_2 is considered as inputs for later layers.
  - This has the effect of reducing correlation impacts among different adjacent styles.

Style GAN

Performance metrics proposed in the paper

- Frechet Inception Distance (FID)
  - Compares the means and covariances of gaussians fit on the true and generated data. (Check the formula for FID!)
- Perceptual path length
  - $\mathbb{E}[\frac{1}{\epsilon^2} d(G(\text{slerp}(z_1, z_2, t)), G(slerp(z_1, z_2, t + \epsilon)))]$
  - slerp denotes spherical interpolation.
  - G denotes the generator.
  - d denotes the quadratic distance.
  - ϵ denotes a perturbation.
- Linear separability for coarse styles: e.g. gender, pose.

Style GAN

Figure 3. 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. Copying the styles corresponding to coarse spatial resolutions ($4^2 - 8^2$) brings high-level aspects such as pose, general hair style, face shape, and eyeglasses from source B, while all colors (eyes, hair, lighting) and finer facial features resemble A. If we instead copy the styles of middle resolutions ($16^2 - 32^2$) from B, we inherit smaller scale facial features, hair style, eyes open/closed from B, while the pose, general face shape, and eyeglasses from A are preserved. Finally, copying the fine styles ($64^2 - 1024^2$) from B brings mainly the color scheme and microstructure.