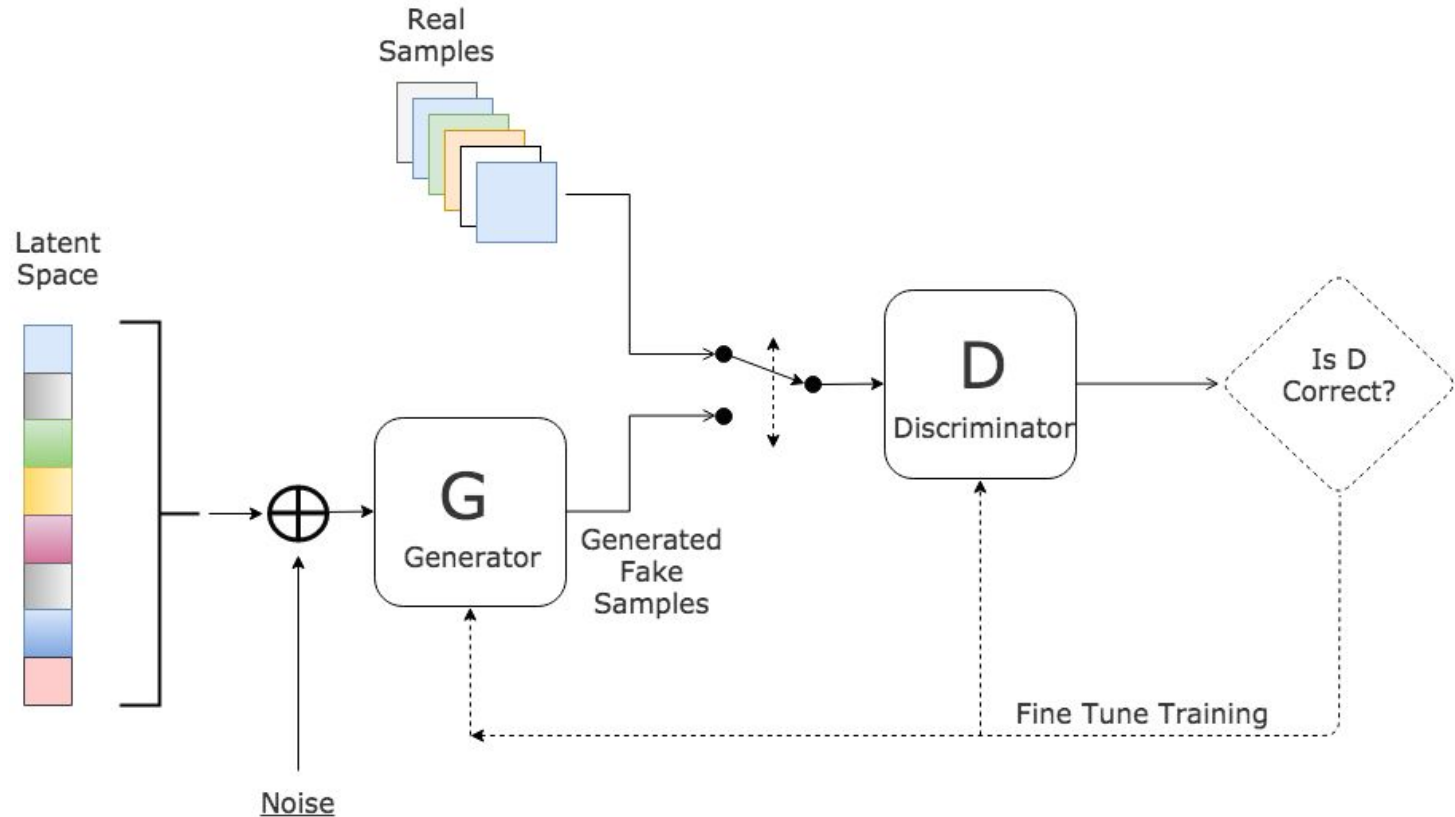


Summary of Recent Generative Adversarial Networks (Classified)

Brandon Liu

Department of Computer Science, University of Virginia
<https://qdata.github.io/deep2Read/>

Generative Adversarial Network



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

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

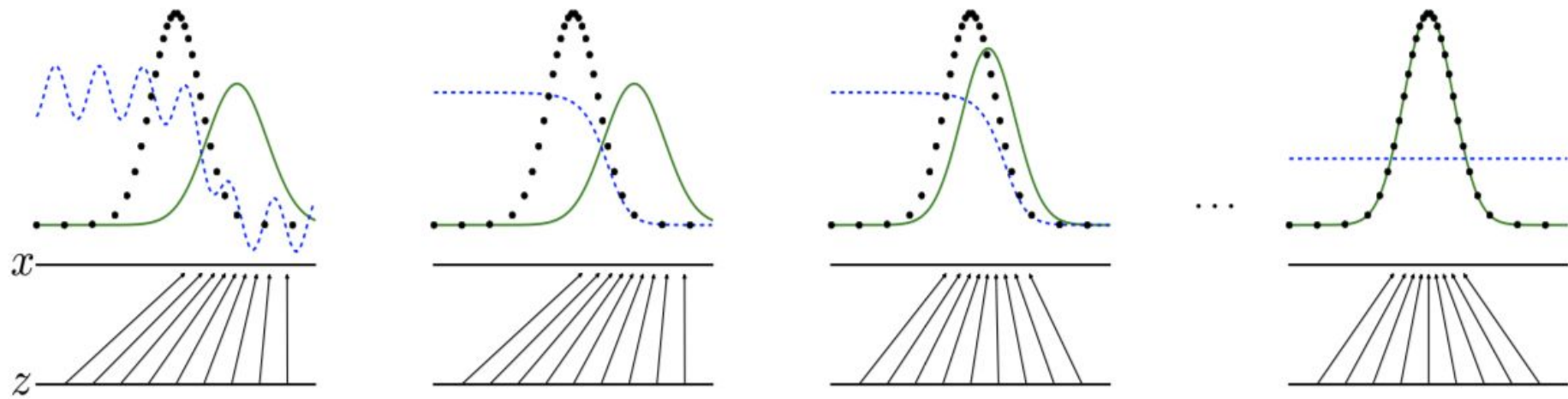
end for

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

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

end for

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



Theorem 1: Global Optimality of $p_g = p_{\text{data}}$

Proposition 1. For G fixed, the optimal discriminator D is

$$D_G^*(\mathbf{x}) = \frac{p_{\text{data}}(\mathbf{x})}{p_{\text{data}}(\mathbf{x}) + p_g(\mathbf{x})}$$

Proof. The training criterion for the discriminator D , given any generator G , is to maximize the quantity $V(G, D)$

$$\begin{aligned} V(G, D) &= \int_{\mathbf{x}} p_{\text{data}}(\mathbf{x}) \log(D(\mathbf{x})) dx + \int_{\mathbf{z}} p_{\mathbf{z}}(\mathbf{z}) \log(1 - D(g(\mathbf{z}))) dz \\ &= \int_{\mathbf{x}} p_{\text{data}}(\mathbf{x}) \log(D(\mathbf{x})) + p_g(\mathbf{x}) \log(1 - D(\mathbf{x})) dx \end{aligned} \quad (3)$$

For any $(a, b) \in \mathbb{R}^2 \setminus \{0, 0\}$, the function $y \rightarrow a \log(y) + b \log(1 - y)$ achieves its maximum in $[0, 1]$ at $\frac{a}{a+b}$. The discriminator does not need to be defined outside of $\text{Supp}(p_{\text{data}}) \cup \text{Supp}(p_g)$, concluding the proof. \square

Contents

1. **Image Generation**
2. Conditional GAN/Domain Transfer
3. Sequence Generation
4. 3D Scene Reconstruction/Lower-Higher Dimension
5. Better Training

Image Generation

Image Generation

- Input image and noise results in output image $(\mathbb{X}, z) \rightarrow \mathbb{X}$
- D: gradient **ascent**

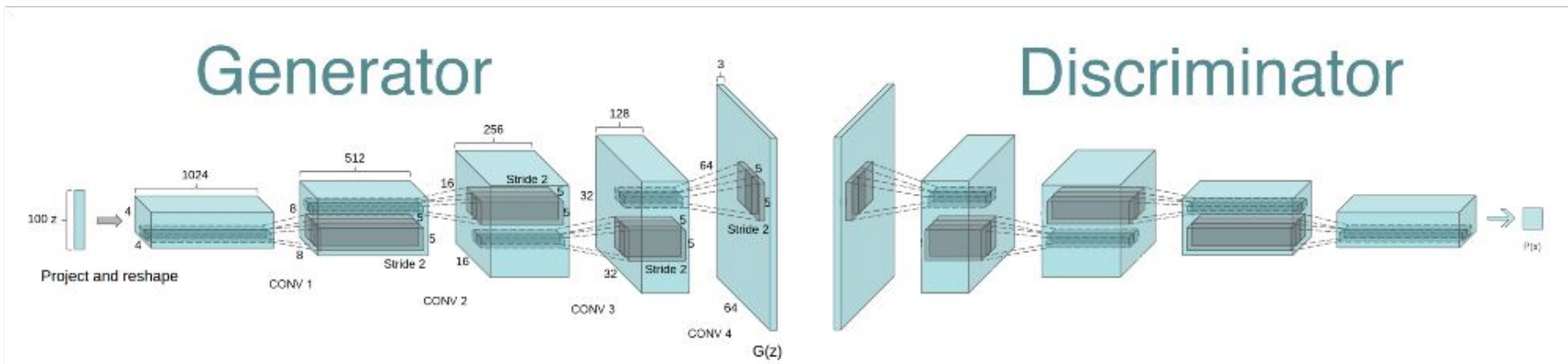
$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^m \left[\log D(\mathbf{x}^{(i)}) + \log \left(1 - D(G(\mathbf{z}^{(i)})) \right) \right]$$

- G: gradient **descent**

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

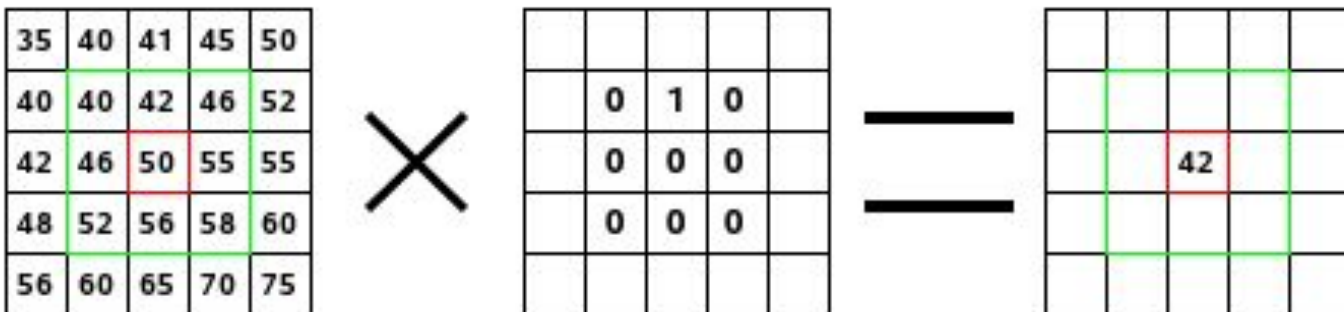
Deep Convolutional GANs (DCGAN)

- Combine CNN and GAN for unsupervised learning
- Learns a hierarchy of feature representations
- Previous attempts to scale up GANs using CNNs unsuccessful

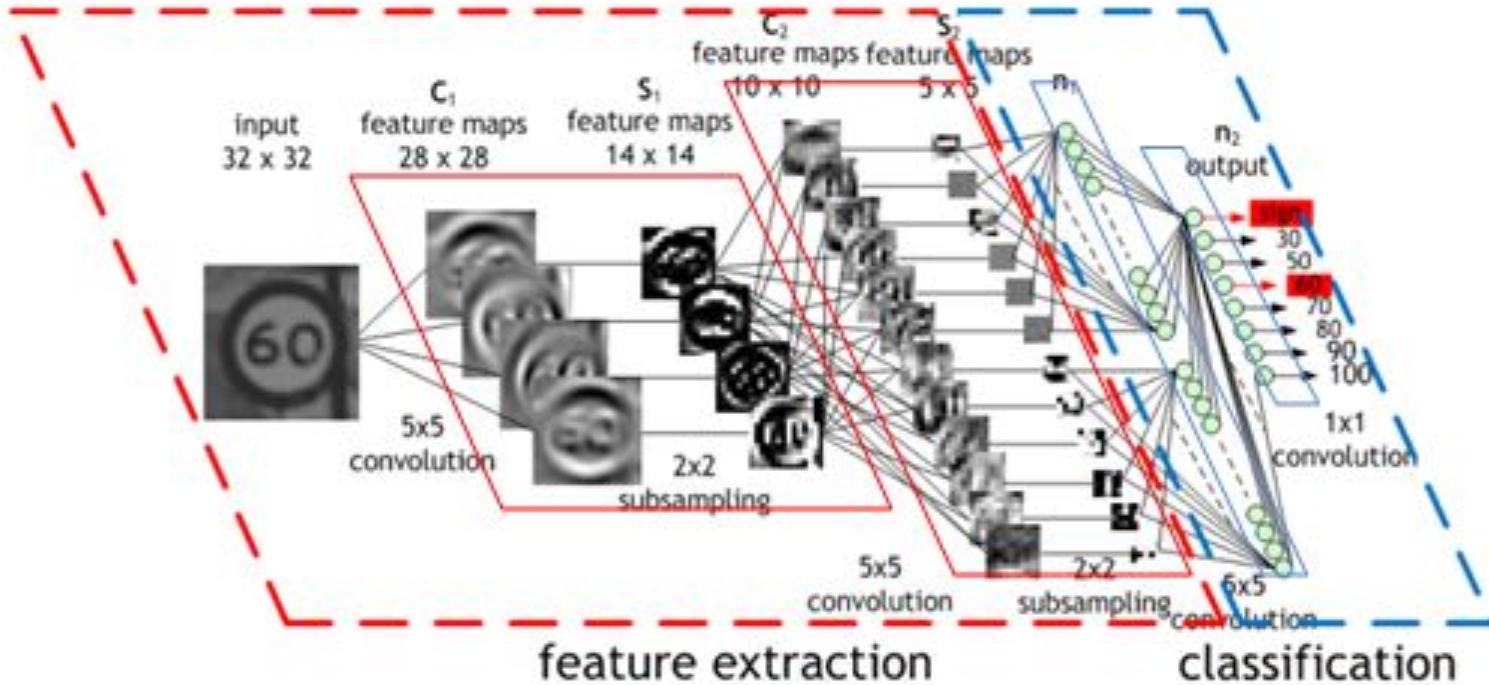


DCGAN - The Convolution Operation

$$(f * g)(u) = \int_{-\infty}^{\infty} f(x)g(u - x)dx$$



DCGAN - Convolutional Neural Networks



DCGAN - Contributions

- Set of constraints on architectural topology of Convolutional GANs that make them stable to train
- Trained discriminators used for classification tasks, competitive performance
- Learned filters learned to draw specific objects
- Generators allow for easy manipulation of semantic qualities of generated samples

DCGAN - Related Work

- **Strided Convolutions:** Tobias, Jost, et al. “Striving for Simplicity: The All Convolutional Net.” [1412.6806] *Striving for Simplicity: The All Convolutional Net*, 13 Apr. 2015, arxiv.org/abs/1412.6806.
- **Representation learning:** Rasmus, Antti, Valpola, Harri, Honkala, Mikko, Berglund, Mathias, and Raiko, Tapani. Semi-supervised learning with ladder network. arXiv preprint arXiv:1507.02672, 2015. <https://arxiv.org/abs/1507.02672>
- **Laplacian Pyramid:** Denton, et al. “Deep Generative Image Models Using a Laplacian Pyramid of Adversarial Networks.” [1506.05751] *Deep Generative Image Models Using a Laplacian Pyramid of Adversarial Networks*, 18 June 2015, arxiv.org/abs/1506.05751. <https://arxiv.org/abs/1506.05751>
- **Deconvolutions:** Dosovitskiy, Alexey, Springenberg, Jost Tobias, and Brox, Thomas. Learning to generate chairs with convolutional neural networks. arXiv preprint arXiv:1411.5928, 2014. <https://arxiv.org/abs/1411.5928>
- **Batch Normalization:** Sergey, et al. “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift.” [1502.03167], 2 Mar. 2015, arxiv.org/abs/1502.03167.
- **Global Average Pooling:** Mordvintsev, Alexander, Olah, Christopher, and Tyka, Mike. Inceptionism : Going deeper into neural networks. <http://googleresearch.blogspot.com/2015/06/inceptionism-going-deeper-into-neural.html>. Accessed: 2015-06-17.

DCGAN - Approach

- Replace spatial pooling layers with strided convolutions
 - Allows network to learn its own downsampling for discriminator
 - Allows network to learn its own upsampling for generator (fractionally strided)
- Use batchnorm in generator and discriminator
 - Input to each unit normalized to have zero mean and unit variance
 - Prevent mode collapse, stabilize training
- Remove fully connected hidden layers for deeper architectures
- ReLU activation in generator for all layers except output, which uses Tanh
- LeakyReLU activation in discriminator for all layers
 - Compare to maxout activation in Goodfellow

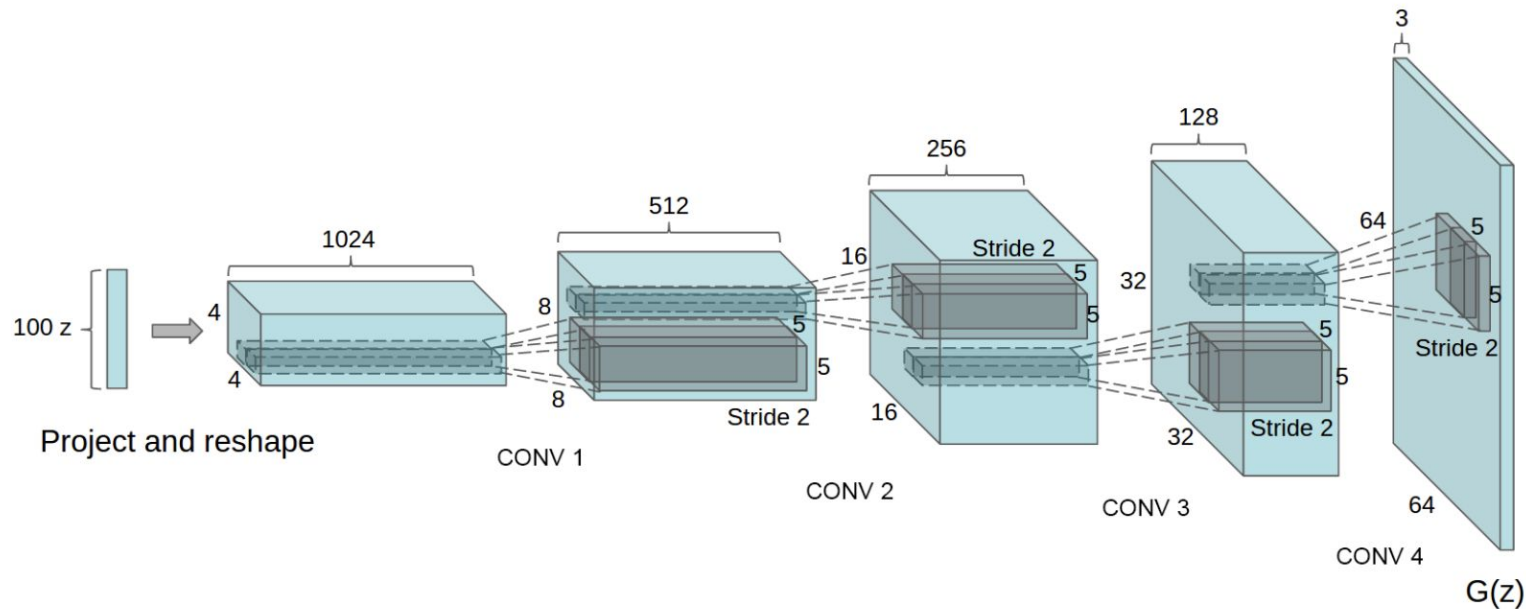
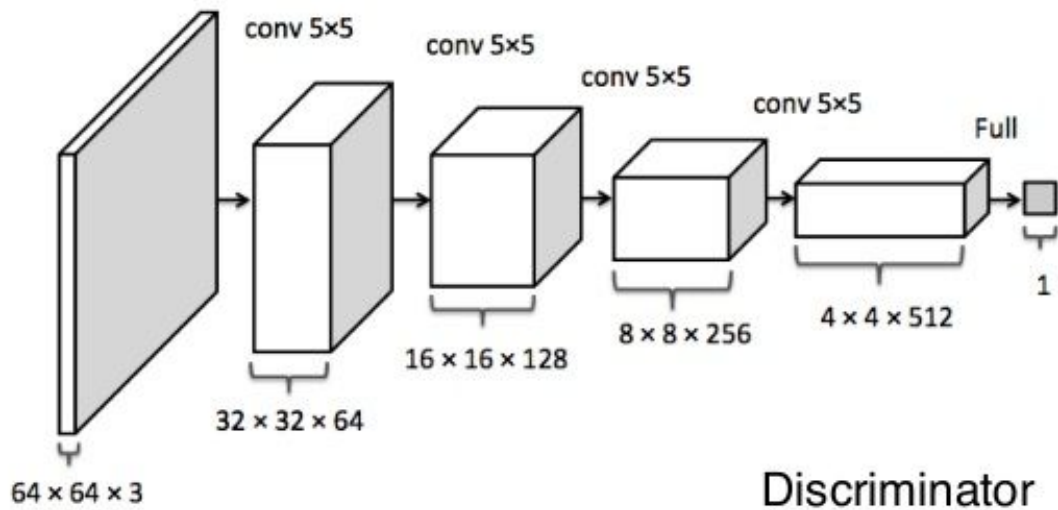


Figure 1: DCGAN generator used for LSUN scene modeling. A 100 dimensional uniform distribution Z is projected to a small spatial extent convolutional representation with many feature maps. A series of four fractionally-strided convolutions (in some recent papers, these are wrongly called deconvolutions)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

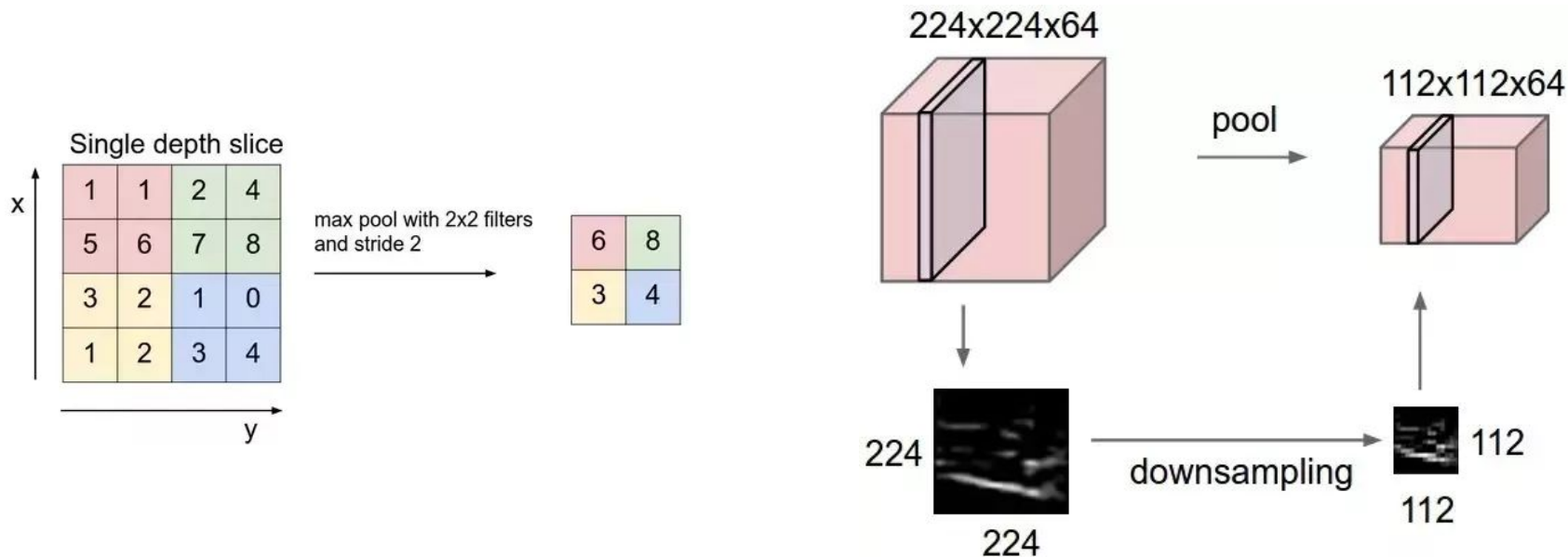
DCGAN Architecture



(Fig. from Yeh et al 2016)

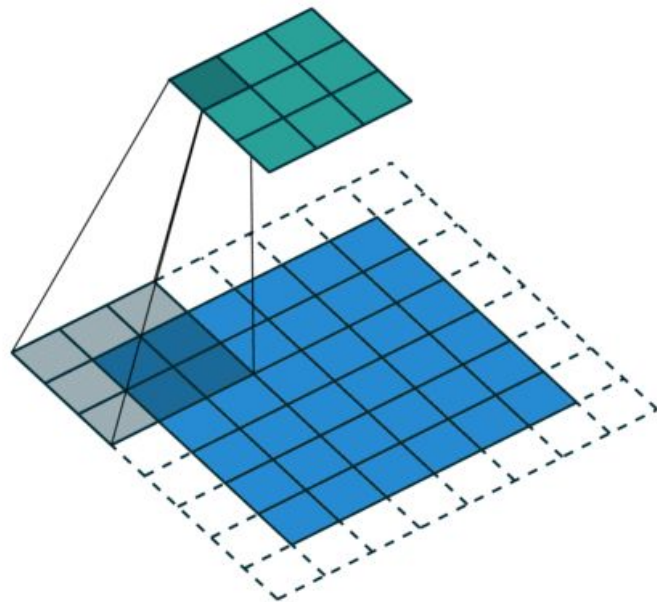
Max Pooling

- Reduces dimensionality of input
- Prevents overfitting, reduces computational cost



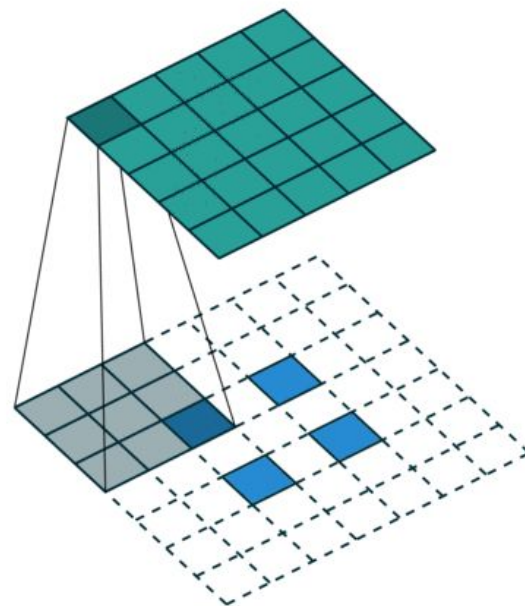
Strided Convolutions (Discriminator)

- Normal convolution operation
- Stride determines step size across the input
- Reduces dimensionality of output
- Compared to pooling
 - Pros: more general, better summarizer
 - Cons: higher training time



Fractional Convolutions (Generator)

- Also known as transposed convolutions and wrongly as deconvolutions
- Upsampling method (lower to higher resolution)
- Stride s is less than 1
- Filter weights are learned by backprop



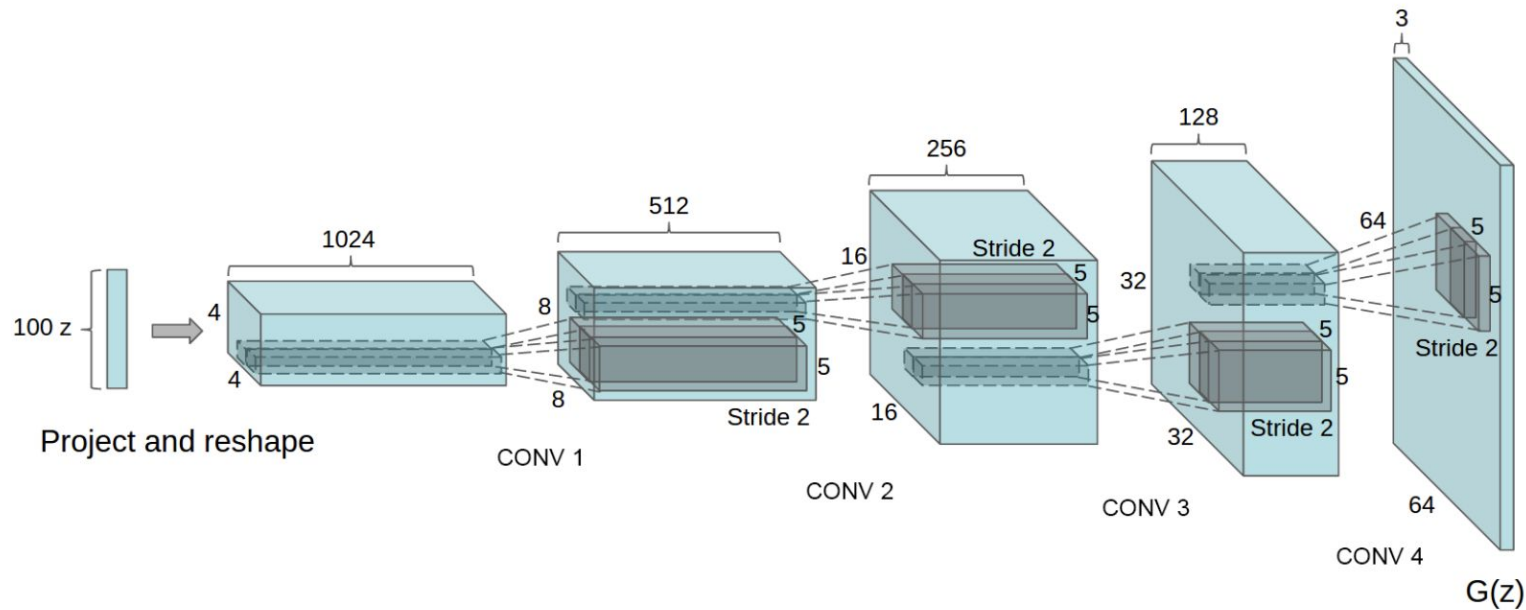
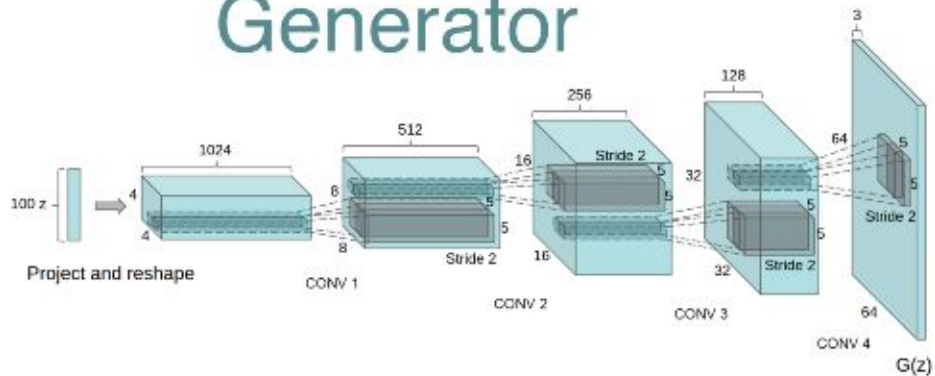
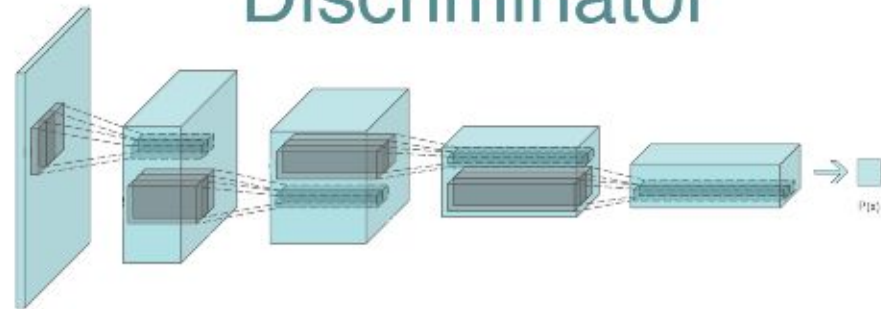


Figure 1: DCGAN generator used for LSUN scene modeling. A 100 dimensional uniform distribution Z is projected to a small spatial extent convolutional representation with many feature maps. A series of four fractionally-strided convolutions (in some recent papers, these are wrongly called deconvolutions)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

Generator



Discriminator



Fractional Convolutions (Generator)

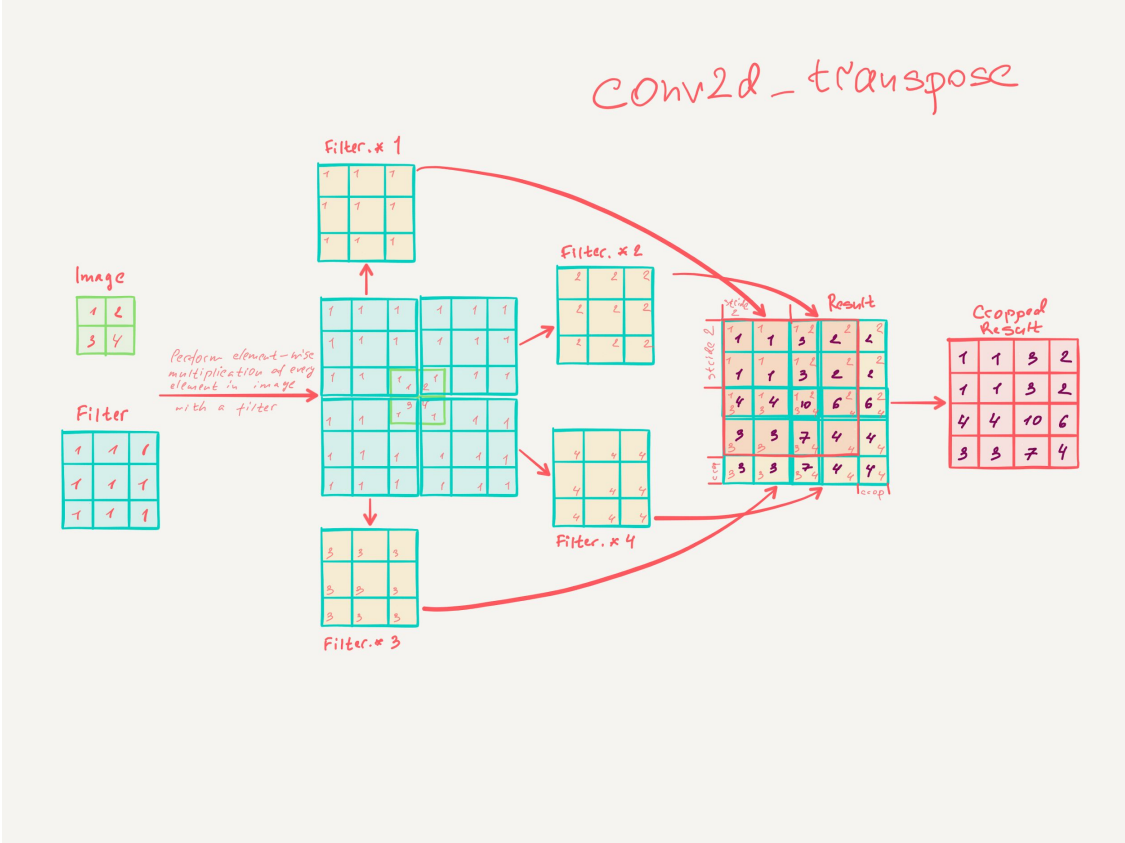




Figure 2: Generated bedrooms after one training pass through the dataset. Theoretically, the model could learn to memorize training examples, but this is experimentally unlikely as we train with a small learning rate and minibatch SGD. We are aware of no prior empirical evidence demonstrating memorization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

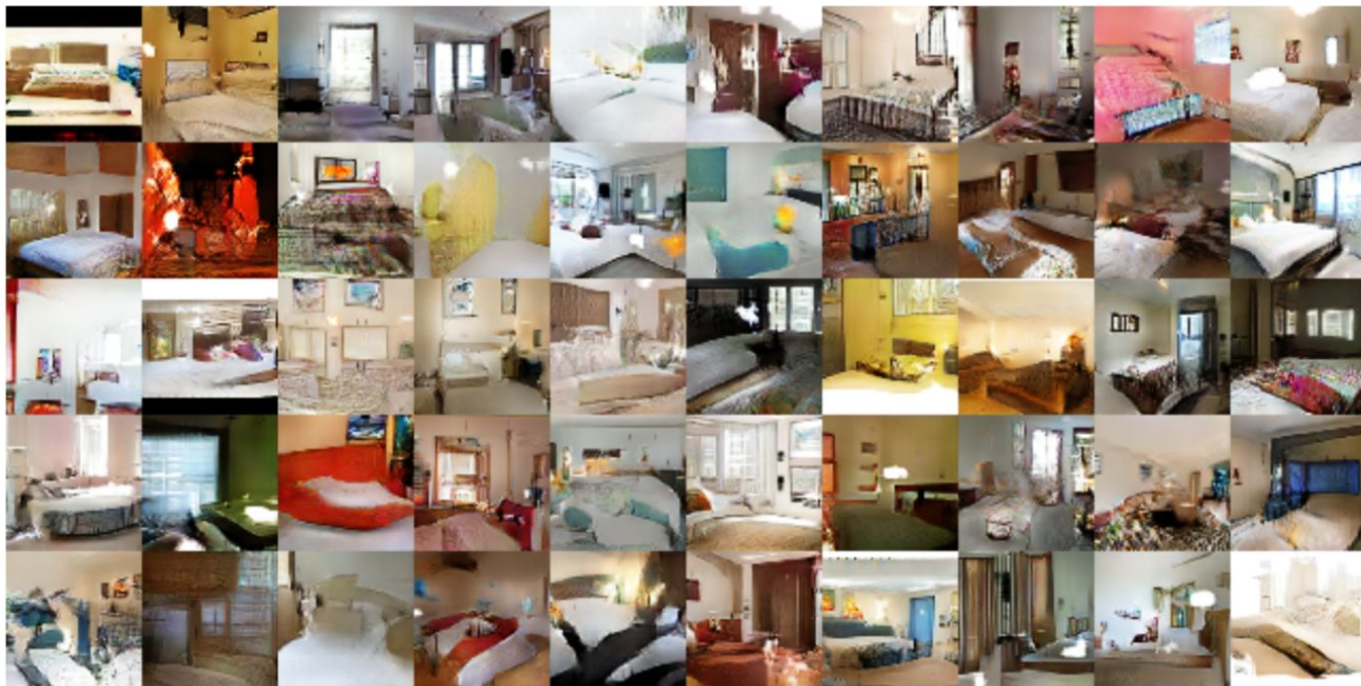


Figure 3: Generated bedrooms after five epochs of training. There appears to be evidence of visual under-fitting via repeated noise textures across multiple samples such as the base boards of some of the beds.

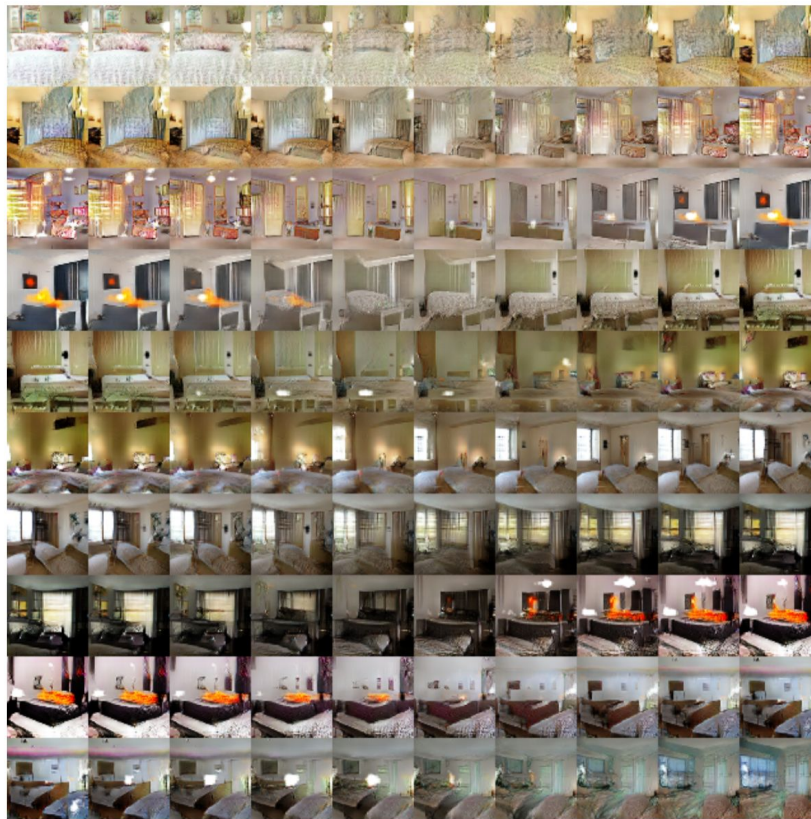


Figure 4: Top rows: Interpolation between a series of 9 random points in Z show that the space learned has smooth transitions, with every image in the space plausibly looking like a bedroom. In the 6th row, you see a room without a window slowly transforming into a room with a giant window. In the 10th row, you see what appears to be a TV slowly being transformed into a window.



Random filters

Trained filters

Figure 5: On the right, guided backpropagation visualizations of maximal axis-aligned responses for the first 6 learned convolutional features from the last convolution layer in the discriminator. Notice a significant minority of features respond to beds - the central object in the LSUN bedrooms dataset. On the left is a random filter baseline. Comparing to the previous responses there is little to no discrimination and random structure.

Contents

1. Image Generation
2. **Conditional GAN/Domain Transfer**
3. Sequence Generation
4. 3D Scene Reconstruction/Lower-Higher Dimension
5. Better Training

Conditional GAN

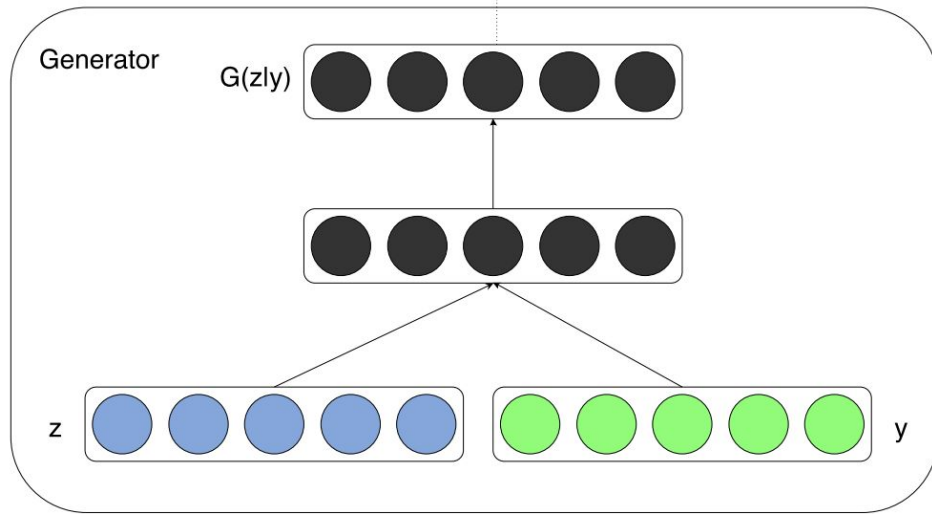
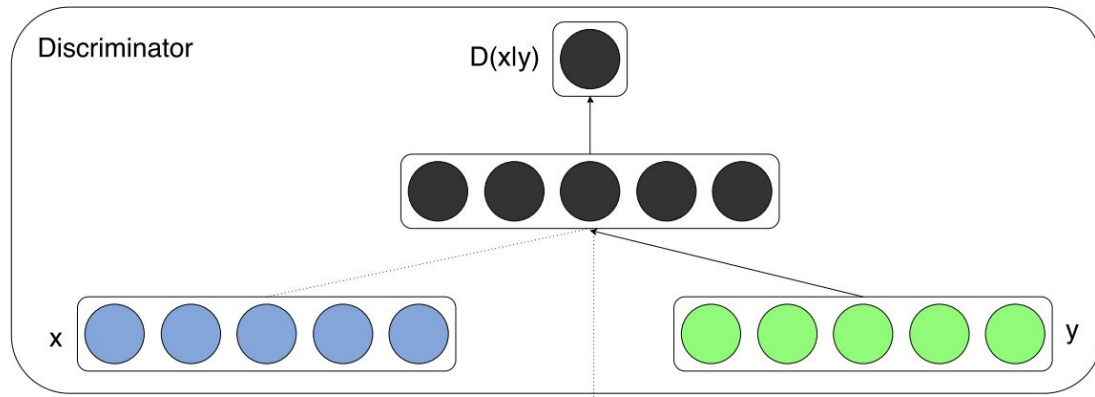
Conditional GANs

- Multimodal learning with class conditional $(\mathbb{X}, z, y) \rightarrow \mathbb{Y}$
- Modified GAN Loss

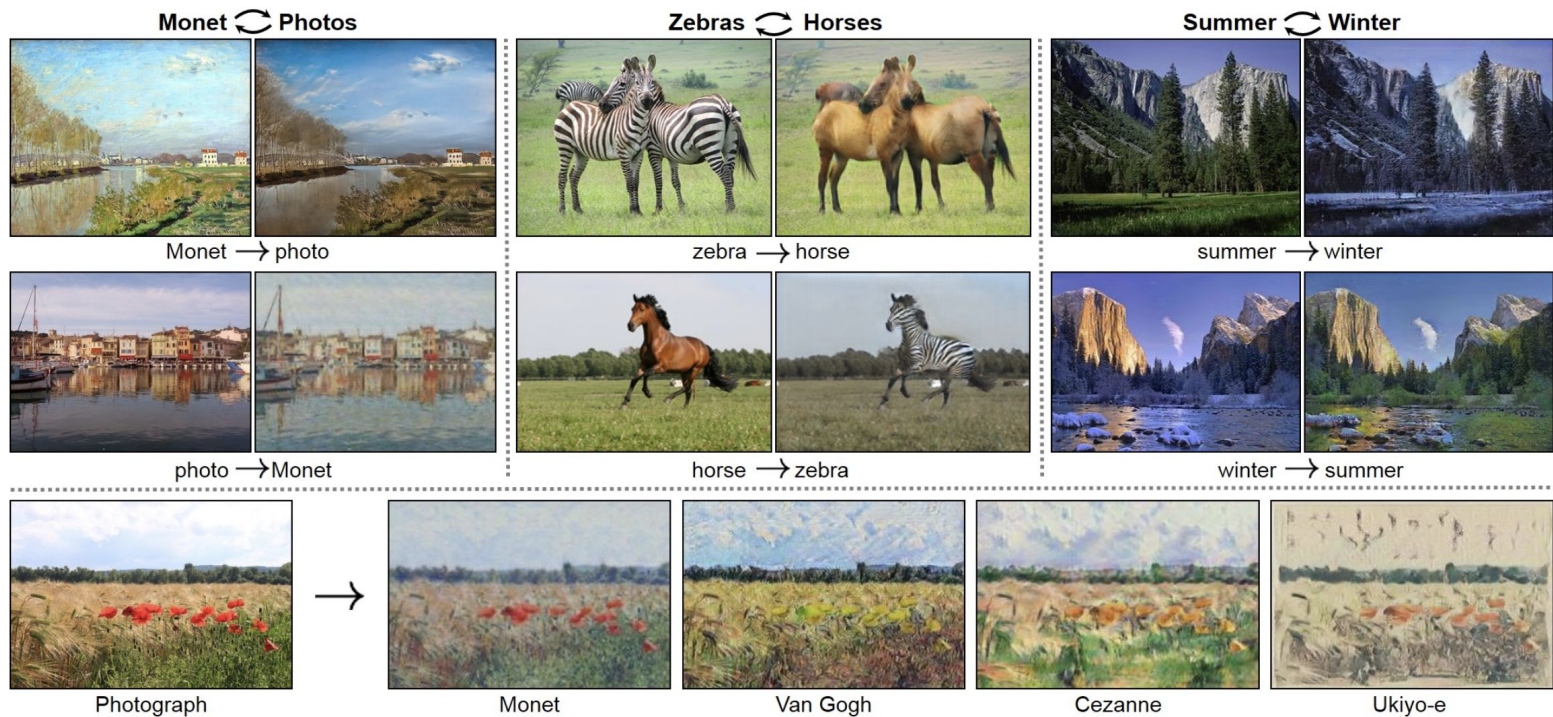
$$\min_G \max_D V(D, G) = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})} [\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p_z(\mathbf{z})} [\log(1 - D(G(\mathbf{z})))] \quad (1)$$

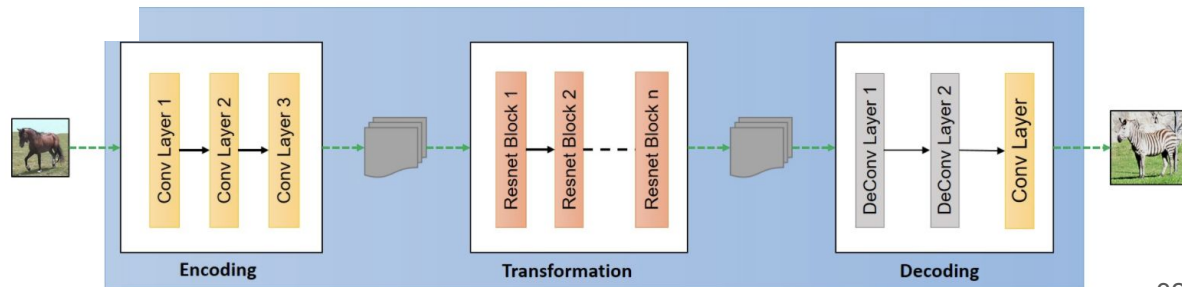
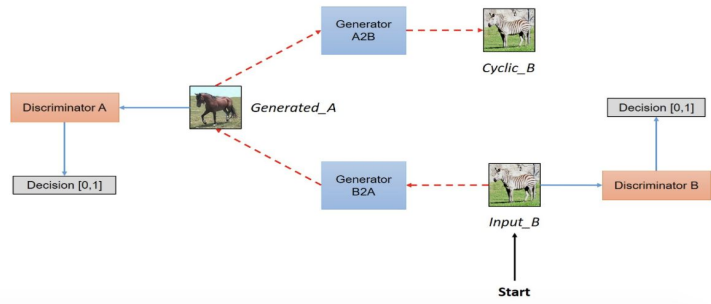
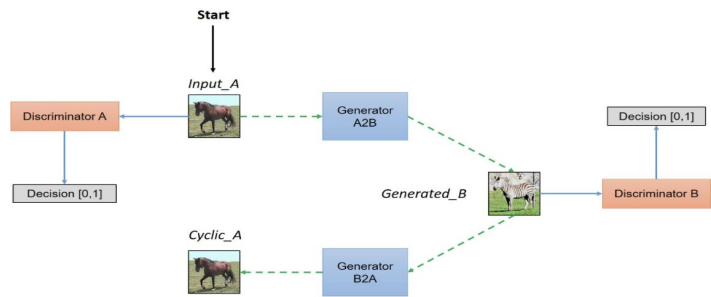
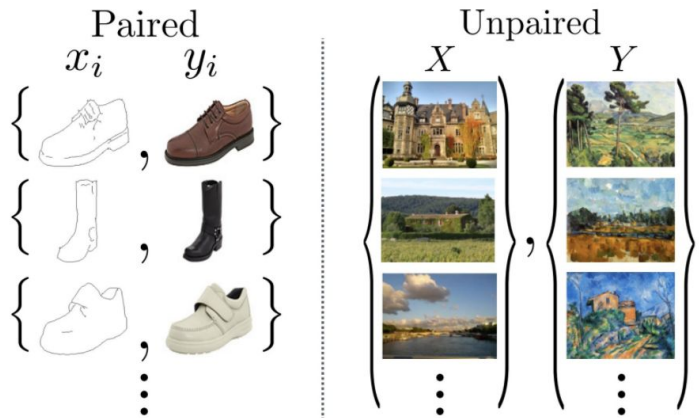


$$\min_G \max_D V(D, G) = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})} [\log D(\mathbf{x}|\mathbf{y})] + \mathbb{E}_{\mathbf{z} \sim p_z(\mathbf{z})} [\log(1 - D(G(\mathbf{z}|\mathbf{y})))] \quad (2)$$



Conditional GANs - Domain Transfer





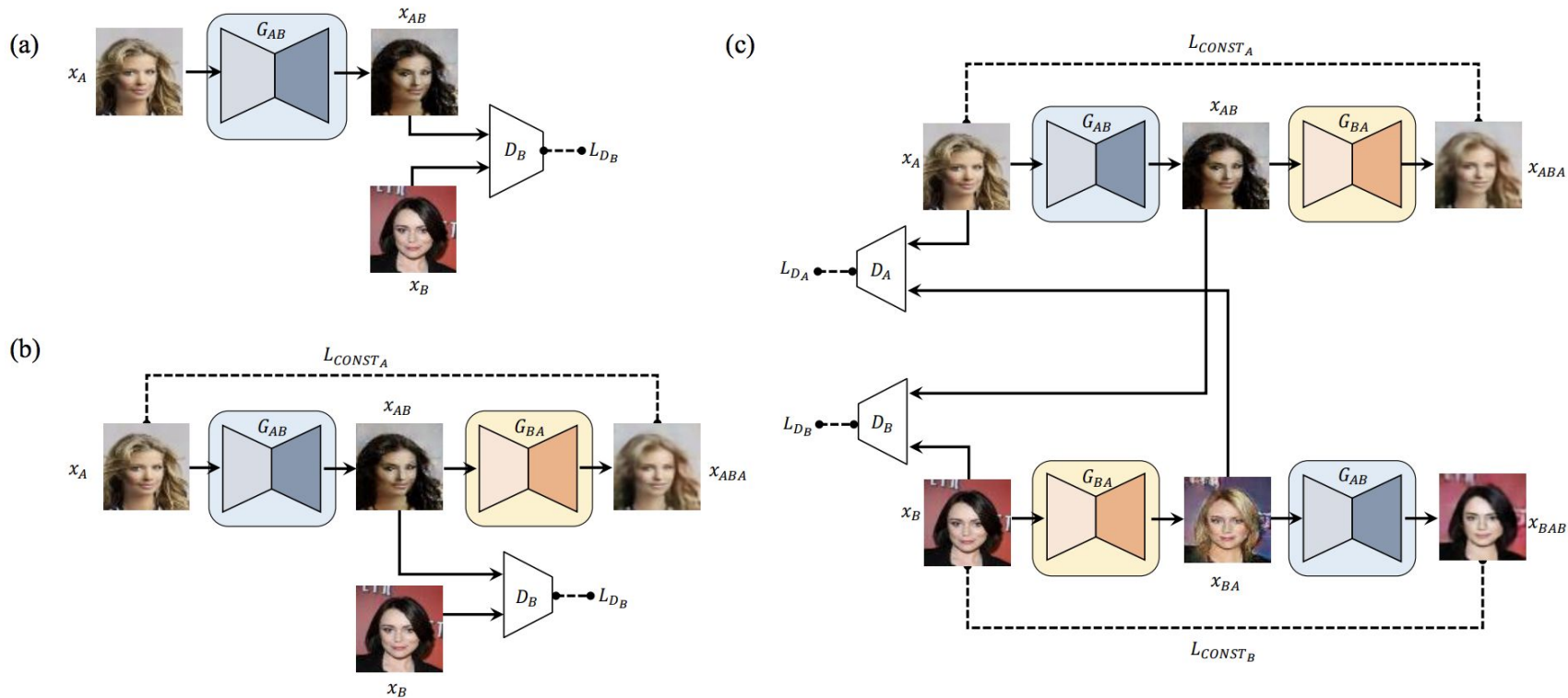


Figure 2. Three investigated models. (a) standard GAN (Goodfellow et al., 2014), (b) GAN with a reconstruction loss, (c) our **proposed model (DiscoGAN)** designed to discover relations between two unpaired, unlabeled datasets. Details are described in Section 3.

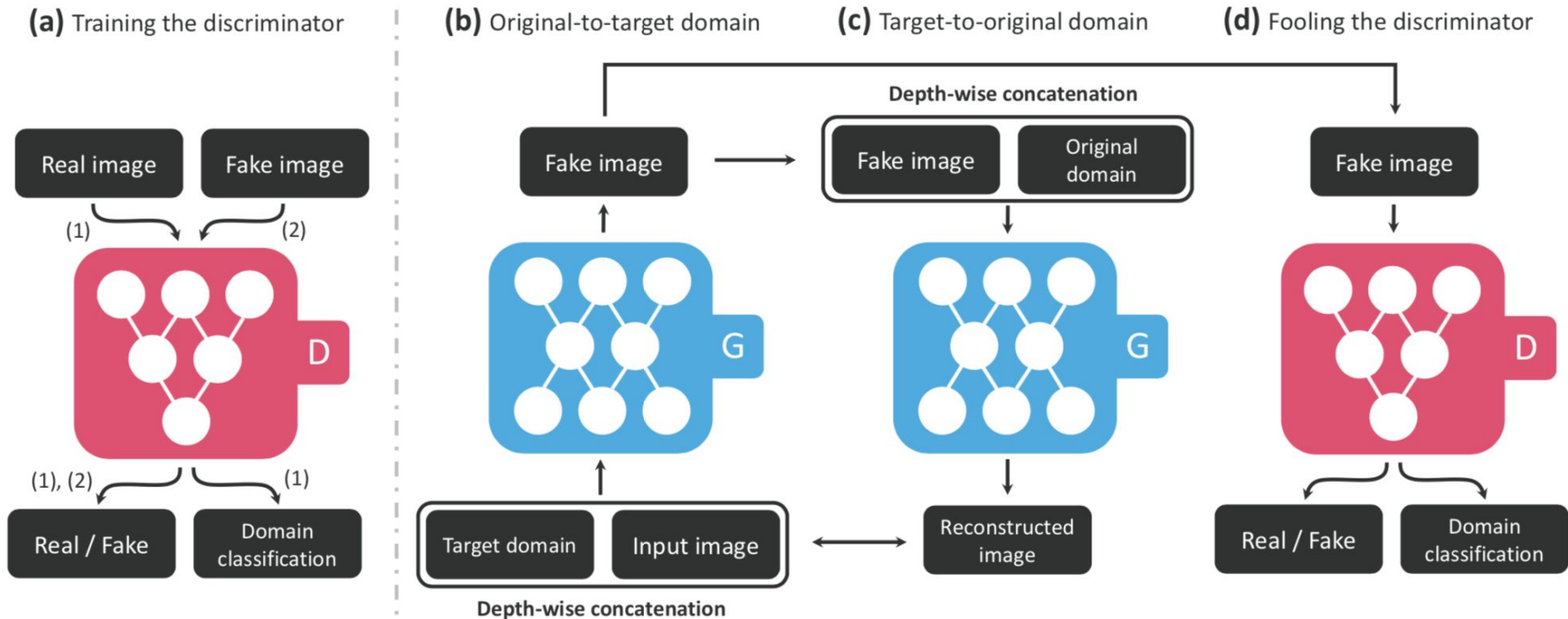


Figure 3. Overview of StarGAN, consisting of two modules, a discriminator D and a generator G . **(a)** D learns to distinguish between real and fake images and classify the real images to its corresponding domain. **(b)** G takes in as input both the image and target domain label and generates an fake image. The target domain label is spatially replicated and concatenated with the input image. **(c)** G tries to reconstruct the original image from the fake image given the original domain label. **(d)** G tries to generate images indistinguishable from real images and classifiable as target domain by D .

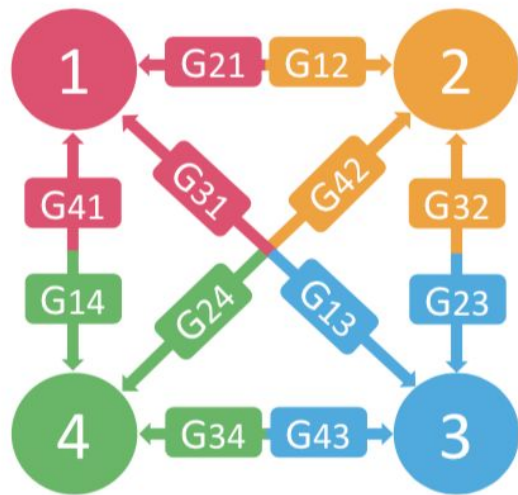
StarGAN - Background

- Image-to-image translation: change a particular aspect of a given image to another (e.g. facial expression smiling to frowning)
- Attribute: meaningful feature inherent in image (e.g. hair color/gender/age)
- Attribute value: value of attribute (e.g. brown hair)
- Domain: a set of images sharing the same attribute value

Previous Work

- Conditional GANs - steer image translation to various target domains by providing conditional domain information
- Image-to-image translation
 - Can preserve key attributes of domains being transferred
 - HOWEVER every current framework can only transfer between two domains at a time, and is not scalable.

(a) Cross-domain models



(b) StarGAN

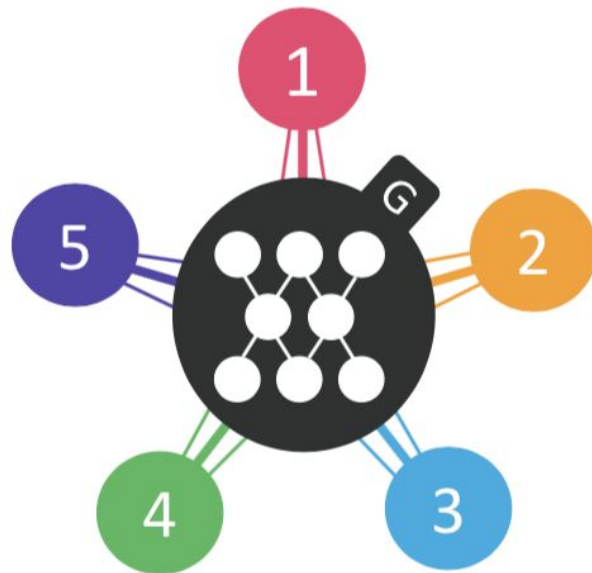


Figure 2. Comparison between cross-domain models and our proposed model, StarGAN. (a) To handle multiple domains, cross-domain models should be built for every pair of image domains. (b) StarGAN is capable of learning mappings among multiple domains using a single generator. The figure represents a star topology connecting multi-domains.

Objective: Train a single generator to learn mappings between domains.

Model

- Generator $G(\bar{x}, c) \rightarrow \bar{y}$
 - Where x is the input image to translate
 - c is a given target domain label
 - y is the output image
- Discriminator $D : x \rightarrow \{D_{src}(x), D_{cls}(x)\}$
 - Where src is the probability distribution over sources
 - cls is the probability over domain labels

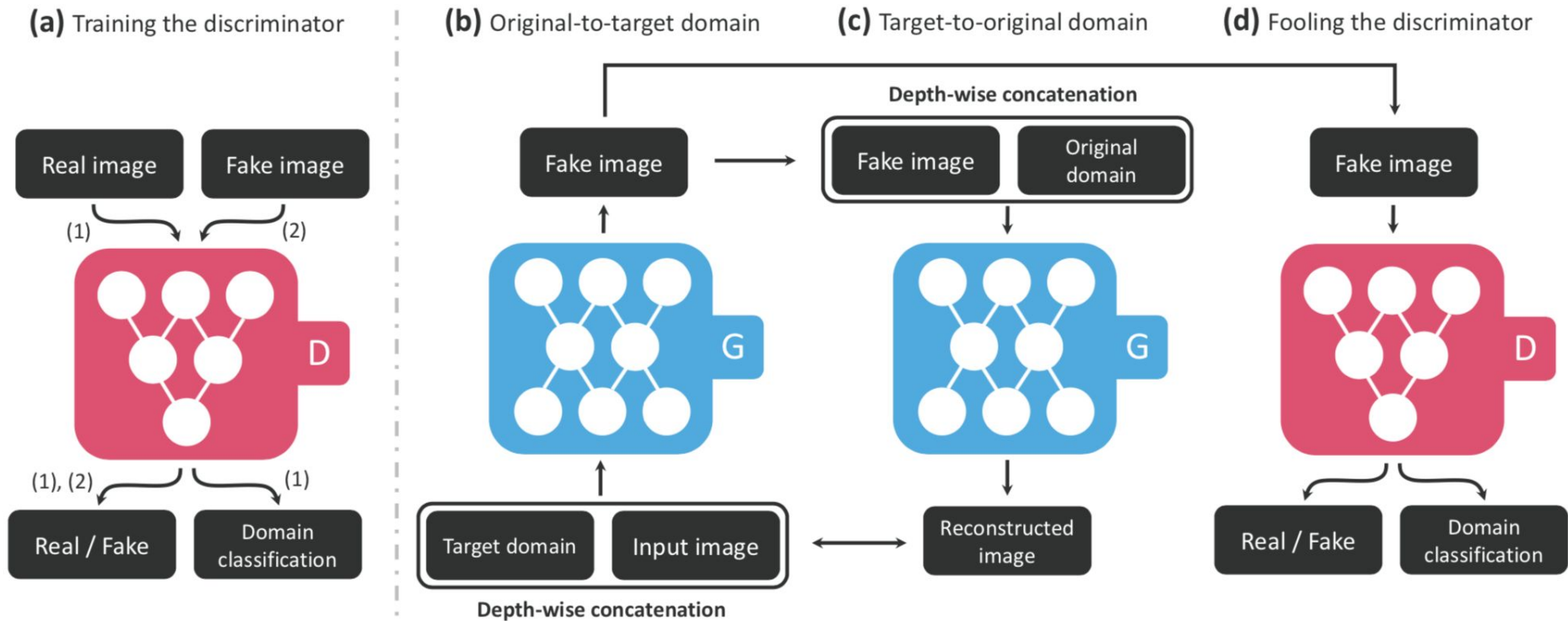


Figure 3. Overview of StarGAN, consisting of two modules, a discriminator D and a generator G . **(a)** D learns to distinguish between real and fake images and classify the real images to its corresponding domain. **(b)** G takes in as input both the image and target domain label and generates an fake image. The target domain label is spatially replicated and concatenated with the input image. **(c)** G tries to reconstruct the original image from the fake image given the original domain label. **(d)** G tries to generate images indistinguishable from real images and classifiable as target domain by D .

Adversarial Loss

- Make generated images indistinguishable from real images (1)

$$\mathcal{L}_{adv} = \mathbb{E}_x [\log D_{src}(x)] + \mathbb{E}_{x,c} [\log (1 - D_{src}(G(x, c)))]$$

Domain Classification Loss (DCL)

- Goal is to translate x into an output image y , which is properly classified to the target domain c .
- Objective is decomposed into two terms: DCL of real images used to optimize D , and domain classification loss of fake images used to optimize G .

$$\mathcal{L}_{cls}^r = \mathbb{E}_{x, c'} [-\log D_{cls}(c' | x)]$$

$$\mathcal{L}_{cls}^f = \mathbb{E}_{x, c} [-\log D_{cls}(c | G(x, c))]$$

Reconstruction Loss

- Minimizing adversarial loss and classification loss means G is able to generate realistic images to the right domain, but it does not guarantee that translated images only change the domain-related part of image.
- AKA cycle-consistency loss, where G takes in the translated image and the original domain label c'

$$\mathcal{L}_{rec} = \mathbb{E}_{x,c,c'} [\|x - G(G(x, c), c')\|_1]$$

Summary

- Adversarial (1)
- Domain Classification (2)(3)
- Reconstruction (4)
- Generator Objective (5)
- Discriminator Objective (6)
 - With importance hyperparams

$$\mathcal{L}_{adv} = \mathbb{E}_x [\log D_{src}(x)] + \mathbb{E}_{x,c} [\log (1 - D_{src}(G(x, c)))] \quad (1)$$

$$\mathcal{L}_{cls}^r = \mathbb{E}_{x,c'} [-\log D_{cls}(c'|x)], \quad (2)$$

$$\mathcal{L}_{cls}^f = \mathbb{E}_{x,c} [-\log D_{cls}(c|G(x, c))]. \quad (3)$$

$$\mathcal{L}_{rec} = \mathbb{E}_{x,c,c'} [\|x - G(G(x, c), c')\|_1], \quad (4)$$

$$\mathcal{L}_D = -\mathcal{L}_{adv} + \lambda_{cls} \mathcal{L}_{cls}^r, \quad (5)$$

$$\mathcal{L}_G = \mathcal{L}_{adv} + \lambda_{cls} \mathcal{L}_{cls}^f + \lambda_{rec} \mathcal{L}_{rec}, \quad (6)$$

Training with Multiple Datasets

- Label information is only partially known to each dataset
- Ex: Face Datasets
 - CelebA (hair color, eye color, skin color)
 - RaFB (facial expressions)
- Complete information on class label vector c' is required for reconstruction of input image x from translated image $G(x, c)$
- Mask Vector: a n -dimensional one-hot vector m that allows StarGAN to ignore unspecified labels and focus on explicitly known labels
 - $n = \#$ of datasets
 - $c.i =$ vector of labels for i th dataset

$$\tilde{c} = [c_1, \dots, c_n, m]$$

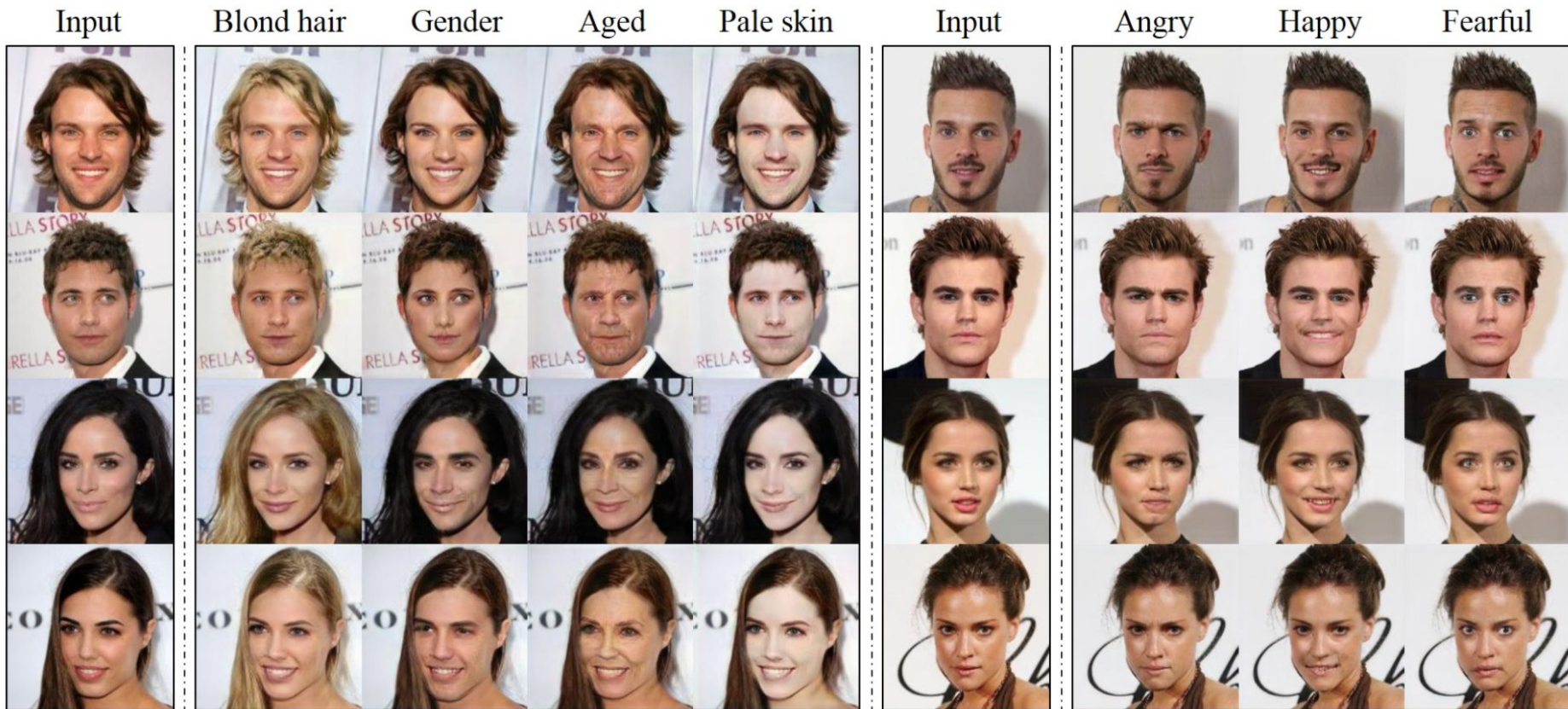


Figure 1. Multi-domain image-to-image translation results on the CelebA dataset via transferring knowledge learned from the RaFD dataset. The first and sixth columns show input images while the remaining columns are images generated by StarGAN. Note that the images are generated by a single generator network, and facial expression labels such as angry, happy, and fearful are from RaFD, not CelebA.

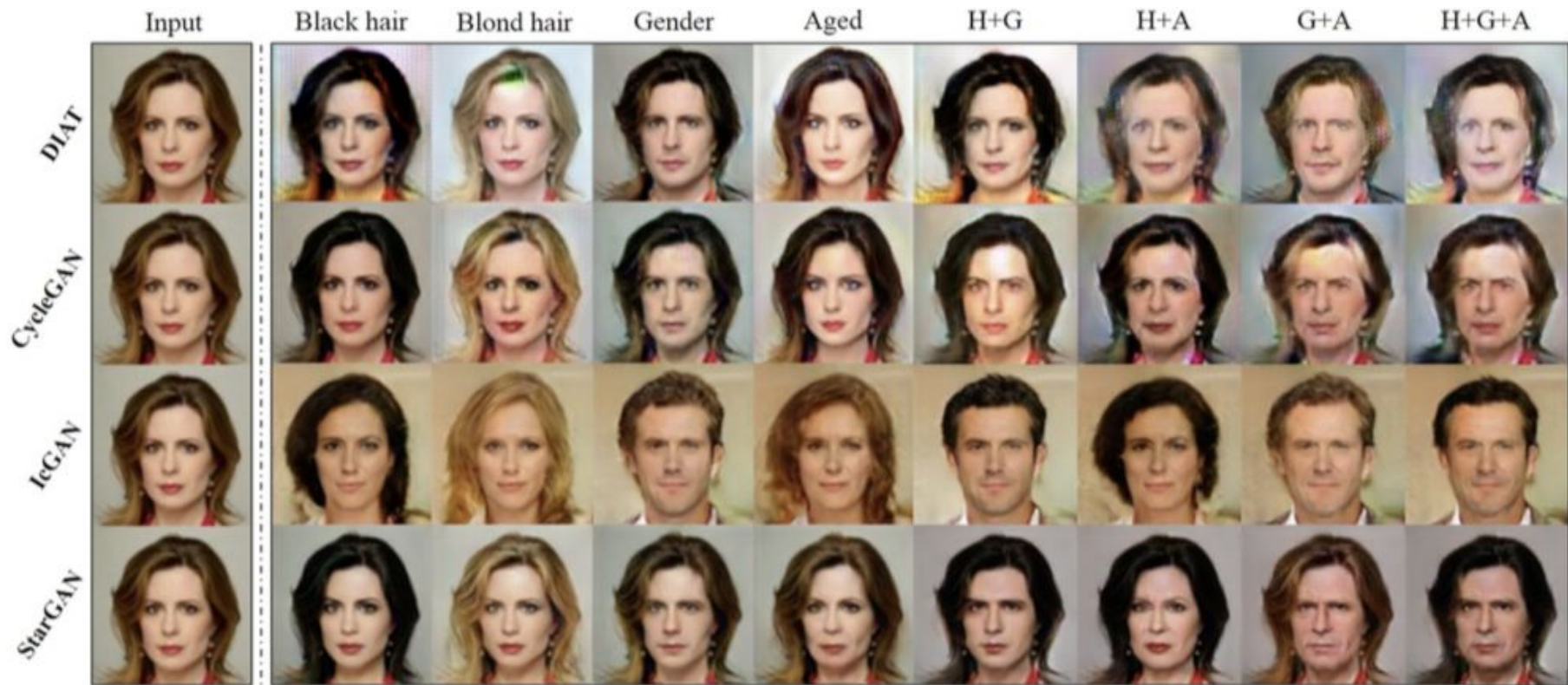


Figure 4. Facial attribute transfer results on the CelebA dataset. The first column shows the input image, next four columns show the single attribute transfer results, and rightmost columns show the multi-attribute transfer results. H: Hair color, G: Gender, A: Aged.



Figure 6. Facial expression synthesis results of StarGAN-SNG and StarGAN-JNT on CelebA dataset.

Method	Hair color	Gender	Aged
DIAT	9.3%	31.4%	6.9%
CycleGAN	20.0%	16.6%	13.3%
IcGAN	4.5%	12.9%	9.2%
StarGAN	66.2%	39.1%	70.6%

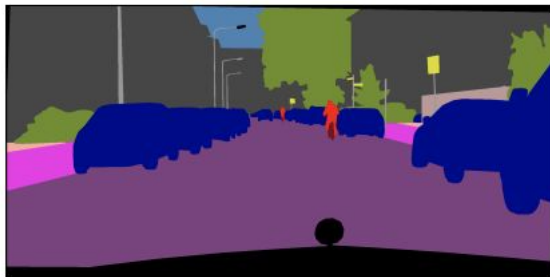
Table 1. AMT perceptual evaluation for ranking different models on a single attribute transfer task. Each column sums to 100%.

Method	H+G	H+A	G+A	H+G+A
DIAT	20.4%	15.6%	18.7%	15.6%
CycleGAN	14.0%	12.0%	11.2%	11.9%
IcGAN	18.2%	10.9%	20.3%	20.3%
StarGAN	47.4%	61.5%	49.8%	52.2%

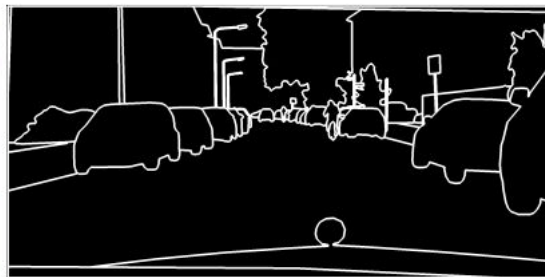
Table 2. AMT perceptual evaluation for ranking different models on a multi-attribute transfer task. H: Hair color; G: Gender; A: Aged.



Figure 7. Learned role of the mask vector. All images are generated by StarGAN-JNT. The last row shows the result of applying the mask vector incorrectly.



(a) Semantic labels



(b) Boundary map

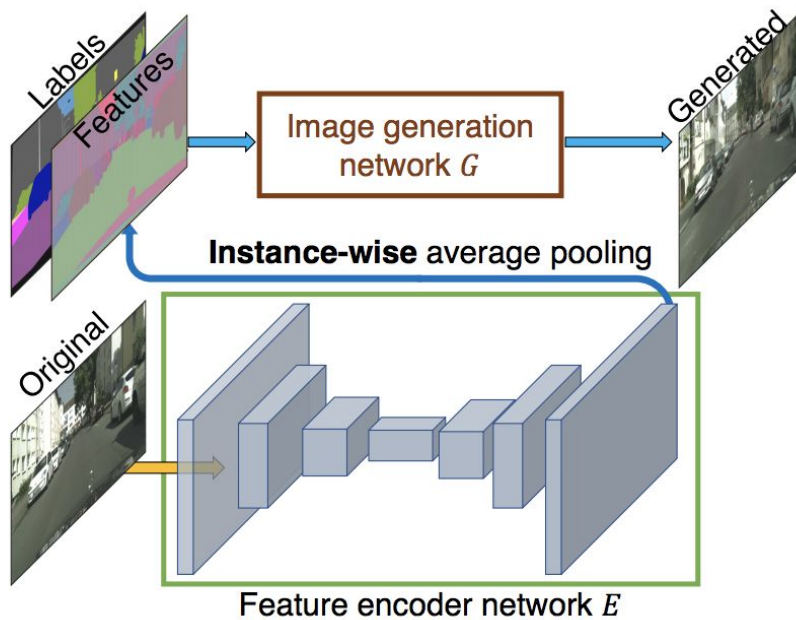


Image Generation via Text

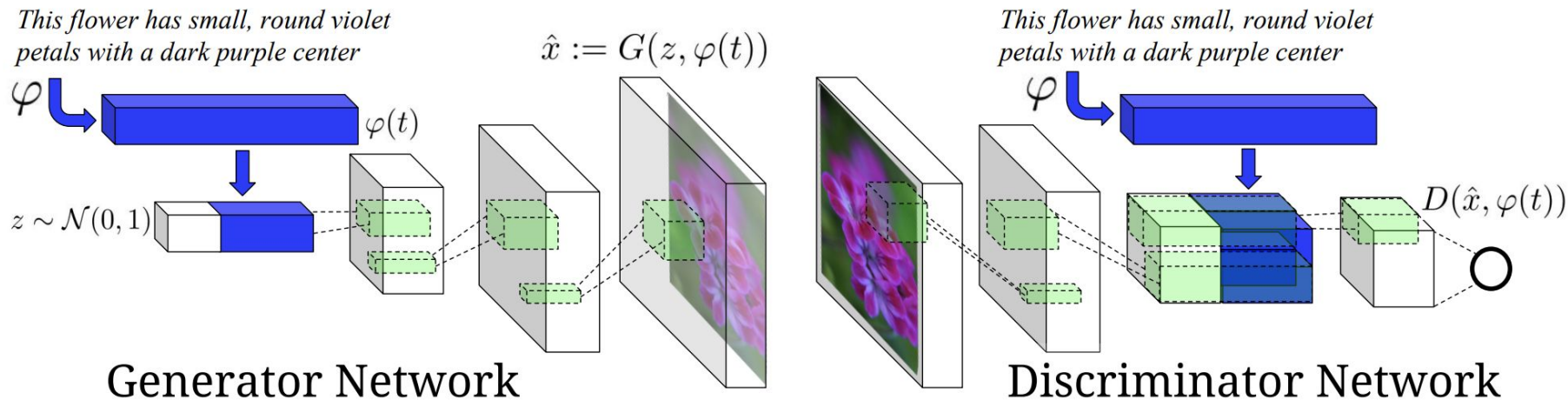


Figure 2. Our text-conditional convolutional GAN architecture. Text encoding $\varphi(t)$ is used by both generator and discriminator. It is projected to a lower-dimensions and depth concatenated with image feature maps for further stages of convolutional processing.

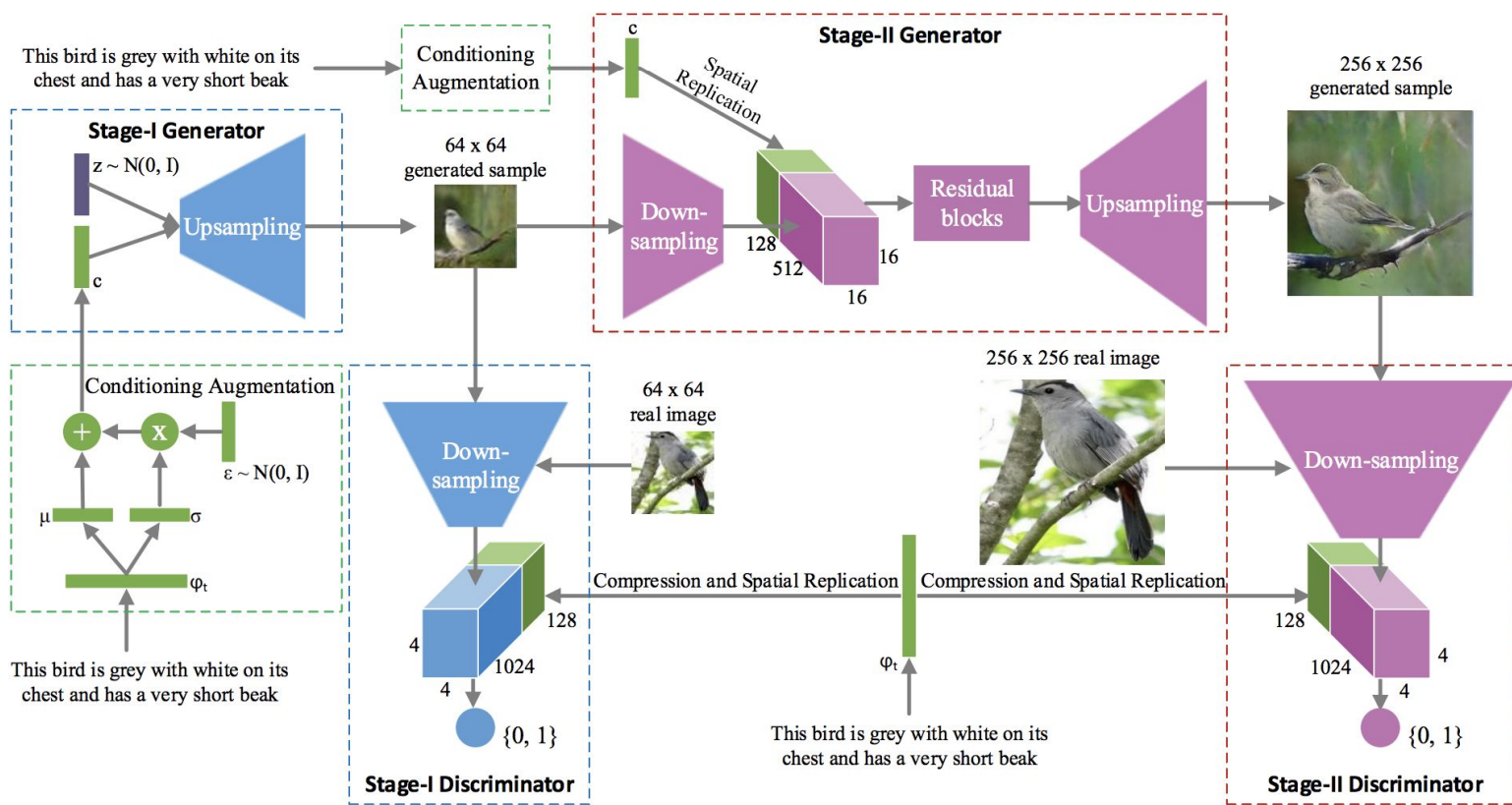


Figure 2. The architecture of the proposed StackGAN. The Stage-I generator draws a low resolution image by sketching rough shape and basic colors of the object from the given text and painting the background from a random noise vector. The Stage-II generator generates a high resolution image with photo-realistic details by conditioning on both the Stage-I result and the text again.

StackGAN - Motivation

- Many practical applications of generating images from text.
 - Photo editing
 - Computer-aided design
- Current state of the art fails to generate necessary details and vivid object parts.

StackGAN - Introduction

- Generating photo-realistic images is difficult.
- Training instability from higher resolutions - can't simply add upsampling.
- Natural image distribution and implied model distribution might not overlap in high dimensional pixel space.
- Contributions:
 - StackGAN - generates 256x256 photorealistic images conditioned on text descriptions
 - Novel conditional augmentation technique that increases diversity of produced images

StackGAN - Related Work

- Stable Training:
 - L. Metz, B. Poole, D. Pfau, and J. Sohl-Dickstein. Unrolled generative adversarial networks. In *ICLR*, 2017.
 - T. Salimans, I. J. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training gans. In *NIPS*, 2016.
 - A. Odena, C. Olah, and J. Shlens. Conditional image synthesis with auxiliary classifier gans. In *ICML*, 2017.
 - M. Arjovsky and L. Bottou. Towards principled methods for training generative adversarial networks. In *ICLR*, 2017.
 - J. Zhao, M. Mathieu, and Y. LeCun. Energy-based generative adversarial network. In *ICLR*, 2017.
- Conditional Image Generation
 - X. Yan, J. Yang, K.Sohn, and H. Lee. Attribute2image:Conditional image generation from visual attributes. In *ECCV*, 2016.
 - S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee. Generative adversarial text-to-image synthesis. In *ICML*, 2016.
- Image Super-Resolution
 - C. K. Snderby, J. Caballero, L.Theis, W. Shi, and F. Huszar. Amortised map inference for image super-resolution. In *ICLR*, 2017.
 - E. L. Denton, S. Chintala, A. Szlam, and R. Fergus. Deep generative image models using a laplacian pyramid of adversarial networks. In *NIPS*, 2015.
- Text Embedding
 - S. Reed, Z. Akata, B. Schiele, and H. Lee. Learning deep representations of fine-grained visual descriptions. In *CVPR*, 2016.

StackGAN - Method

- 2-stage training process
- Stage-I GAN
 - sketches the primitive shape and basic colors of the object conditioned on the given text description
 - draws the background layout from a random noise vector, yielding a low-resolution image
- Stage-II GAN
 - corrects defects in the low-resolution image from Stage-I
 - completes details of the object by reading the text description again, producing a high-resolution photo-realistic image.

Conditional Augmentation

- Latent space for text embedding is extremely high dimensional (>100)
- Limited data causes discontinuity in latent data manifold
- Randomly sample conditioning variables from independent Gaussian distribution where:
 - Mean and diagonal covariance matrix are functions of the text embedding
- Enforce smoothness through regularization during training

$$D_{KL}(\mathcal{N}(\mu(\varphi_t), \Sigma(\varphi_t)) || \mathcal{N}(0, I)), \quad (2)$$

Stage-1 GAN

- Real image I
- Text embedding ϕ
- Gaussian conditional variable c to capture the meaning of embedding with variations

$$\mathcal{L}_{D_0} = \mathbb{E}_{(I_0, t) \sim p_{data}} [\log D_0(I_0, \varphi_t)] + \mathbb{E}_{z \sim p_z, t \sim p_{data}} [\log(1 - D_0(G_0(z, \hat{c}_0), \varphi_t))], \quad (3)$$

$$\mathcal{L}_{G_0} = \mathbb{E}_{z \sim p_z, t \sim p_{data}} [\log(1 - D_0(G_0(z, \hat{c}_0), \varphi_t))] + \lambda D_{KL}(\mathcal{N}(\mu_0(\varphi_t), \Sigma_0(\varphi_t)) \parallel \mathcal{N}(0, I)), \quad (4)$$

Stage-2 GAN

- Conditioning on output of first stage and text embedding
- Real 256x256 images input to the discriminator
- Randomness z not used, assumed to be preserved from previous output s_0

$$\mathcal{L}_D = \mathbb{E}_{(I,t) \sim p_{data}} [\log D(I, \varphi_t)] + \mathbb{E}_{s_0 \sim p_{G_0}, t \sim p_{data}} [\log(1 - D(G(s_0, \hat{c}), \varphi_t))], \quad (5)$$

$$\mathcal{L}_G = \mathbb{E}_{s_0 \sim p_{G_0}, t \sim p_{data}} [\log(1 - D(G(s_0, \hat{c}), \varphi_t))] + \lambda D_{KL}(\mathcal{N}(\mu(\varphi_t), \Sigma(\varphi_t)) \parallel \mathcal{N}(0, I)), \quad (6)$$

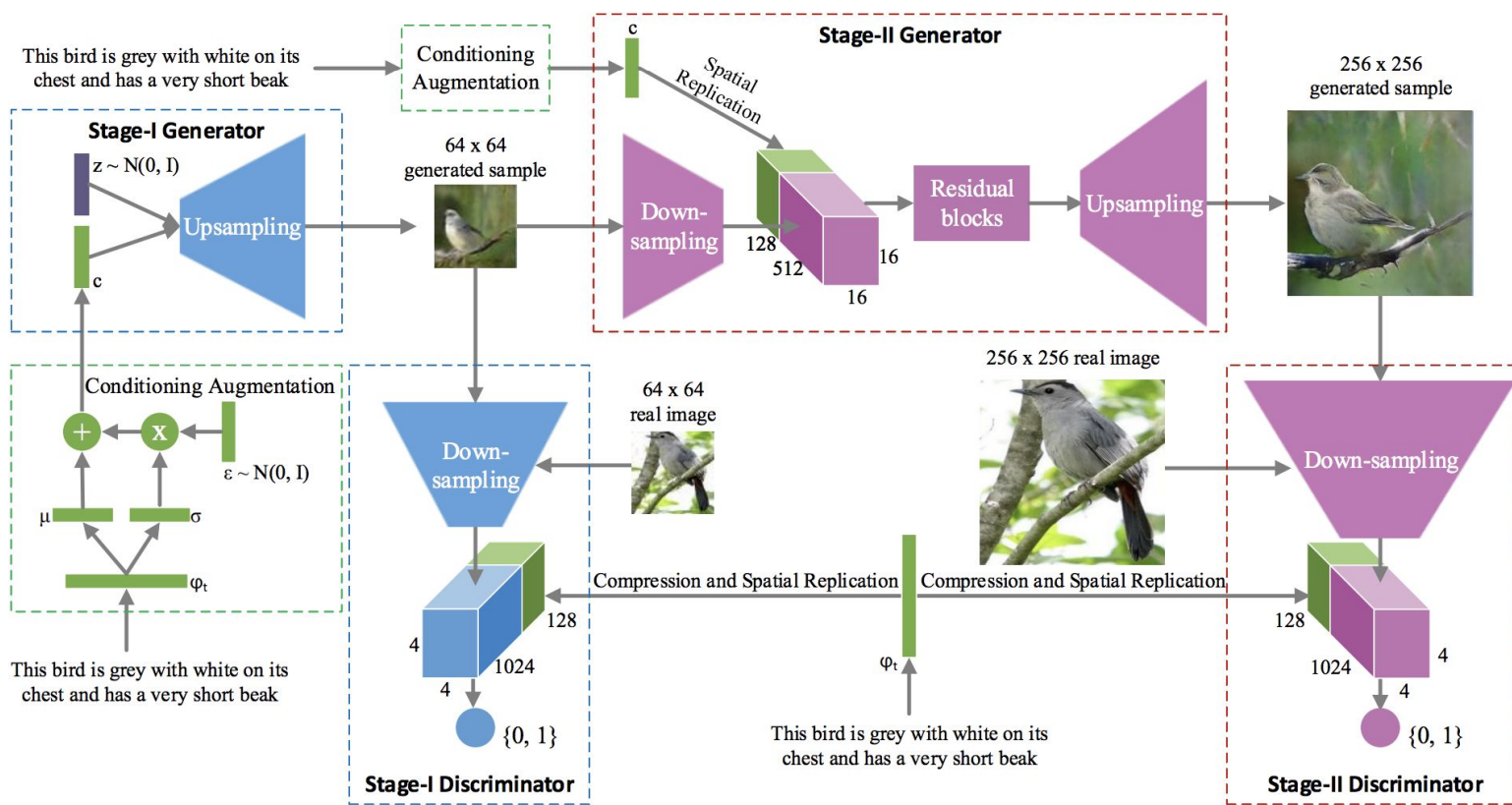


Figure 2. The architecture of the proposed StackGAN. The Stage-I generator draws a low resolution image by sketching rough shape and basic colors of the object from the given text and painting the background from a random noise vector. The Stage-II generator generates a high resolution image with photo-realistic details by conditioning on both the Stage-I result and the text again.

Text description	This bird is red and brown in color, with a stubby beak	The bird is short and stubby with yellow on its body	A bird with a medium orange bill white body gray wings and webbed feet	This small black bird has a short, slightly curved bill and long legs	A small bird with varying shades of brown with white under the eyes	A small yellow bird with a black crown and a short black pointed beak	This small bird has a white breast, light grey head, and black wings and tail
64x64 GAN-INT-CLS							
128x128 GAWWN							
256x256 StackGAN							

Figure 3. Example results by our StackGAN, GAWWN [24], and GAN-INT-CLS [26] conditioned on text descriptions from CUB test set.

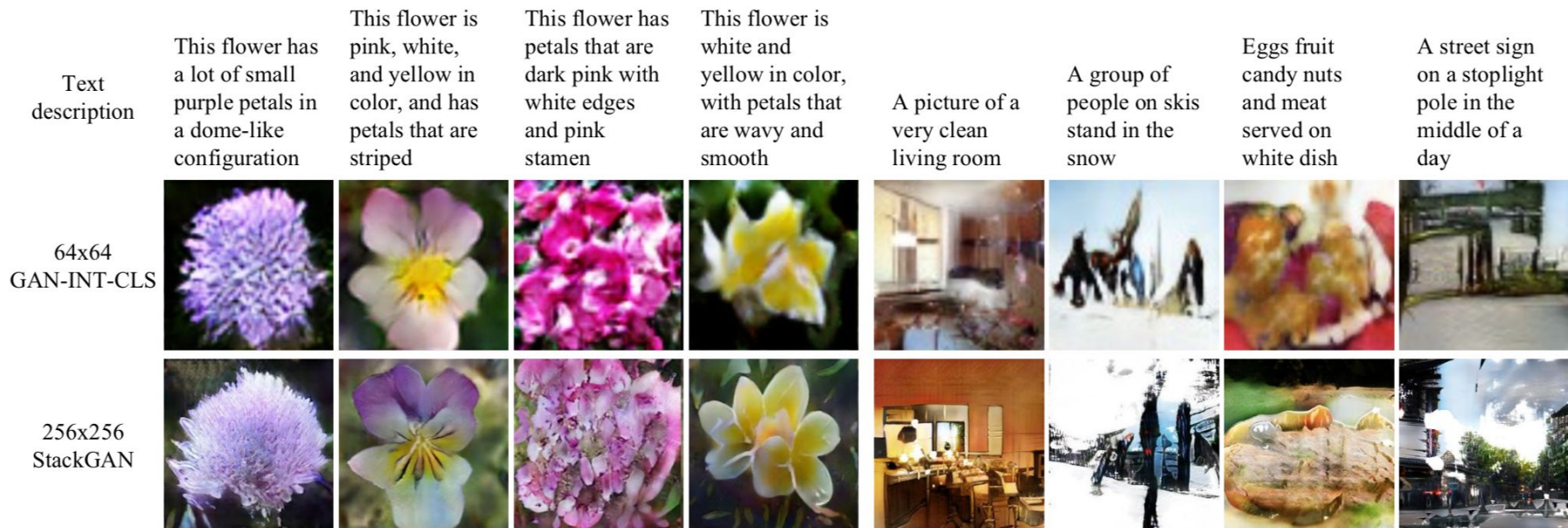


Figure 4. Example results by our StackGAN and GAN-INT-CLS [26] conditioned on text descriptions from Oxford-102 test set (leftmost four columns) and COCO validation set (rightmost four columns).

Metric	Dataset	GAN-INT-CLS	GAWWN	Our StackGAN
Inception score	CUB	$2.88 \pm .04$	$3.62 \pm .07$	$3.70 \pm .04$
	Oxford	$2.66 \pm .03$	/	$3.20 \pm .01$
	COCO	$7.88 \pm .07$	/	$8.45 \pm .03$
Human rank	CUB	$2.81 \pm .03$	$1.99 \pm .04$	$1.37 \pm .02$
	Oxford	$1.87 \pm .03$	/	$1.13 \pm .03$
	COCO	$1.89 \pm .04$	/	$1.11 \pm .03$

Table 1. Inception scores and average human ranks of our StackGAN, GAWWN [24], and GAN-INT-CLS [26] on CUB, Oxford-102, and MS-COCO datasets.

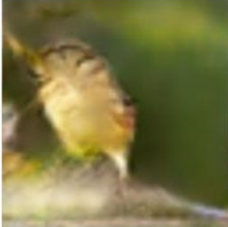
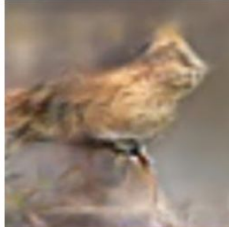






Text description	This bird is blue with white and has a very short beak	This bird has wings that are brown and has a yellow belly	A white bird with a black crown and yellow beak	This bird is white, black, and brown in color, with a brown beak	The bird has small beak, with reddish brown crown and gray belly	This is a small, black bird with a white breast and white on the wingbars.	This bird is white black and yellow in color, with a short black beak
Stage-I images							
Stage-II images							

Figure 5. Samples generated by our StackGAN from unseen texts in CUB test set. Each column lists the text description, images generated from the text by Stage-I and Stage-II of StackGAN.

Contents

1. Image Generation
2. Conditional GAN/Domain Transfer
3. **Sequence Generation**
4. 3D Scene Reconstruction/Lower-Higher Dimension
5. Better Training

Sequence Generation

Generating Sequences

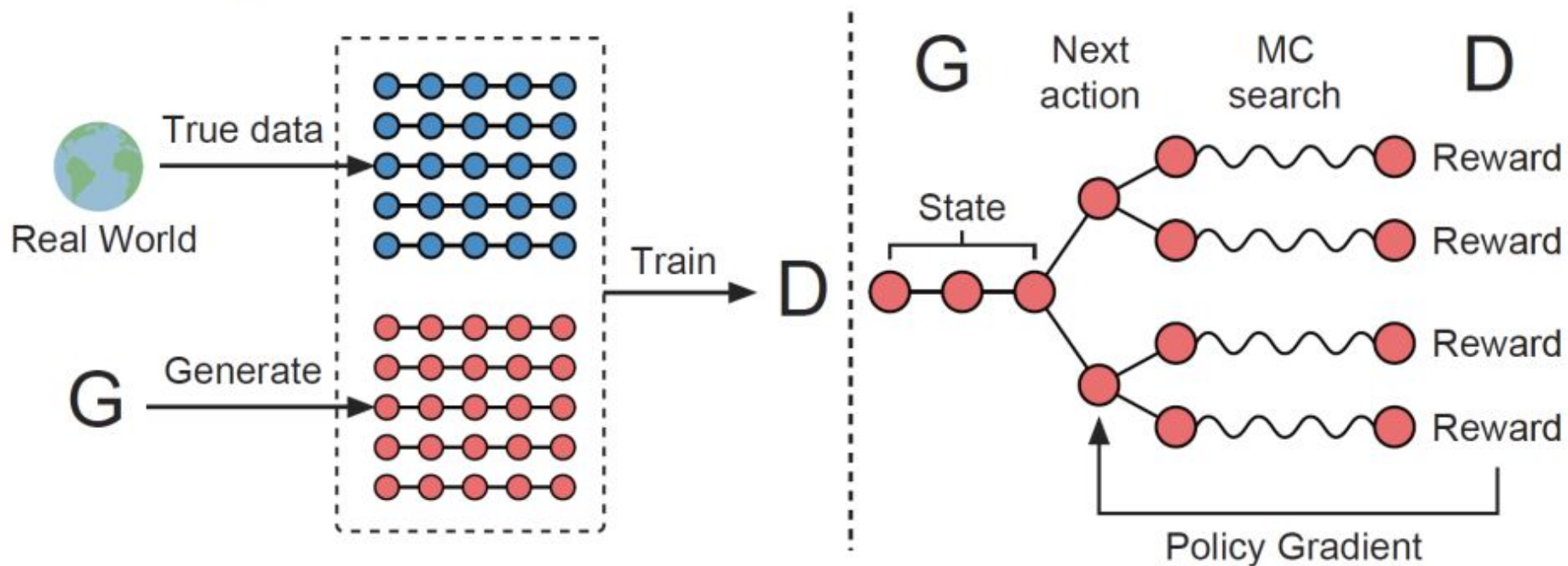
- Inputs/Outputs $(\mathbb{X}, z, s) \rightarrow \mathbb{X}$
- Generator

$$\min_{\phi} -\mathbb{E}_{Y \sim p_{\text{data}}} [\log D_{\phi}(Y)] - \mathbb{E}_{Y \sim G_{\theta}} [\log(1 - D_{\phi}(Y))]$$

- Discriminator - RL objective

$$\nabla_{\theta} J(\theta) = \mathbb{E}_{Y_{1:t-1} \sim G_{\theta}} \left[\sum_{y_t \in \mathcal{Y}} \nabla_{\theta} G_{\theta}(y_t | Y_{1:t-1}) \cdot Q_{D_{\phi}}^{G_{\theta}}(Y_{1:t-1}, y_t) \right]$$

SeqGAN



GAN Motivation - good for real data

- Two problems
- Struggles with generating sequences of discrete tokens
 - Continuous data has a direct gradient

$$\nabla_{\theta(G)} \frac{1}{m} \sum_{i=1}^m \log(1 - D(G(z^{(i)})))$$

- Slight changes from gradient do not have corresponding discrete tokens (e.g. "dog+.001")
- Score only evaluated by D after fully generated sequence

SeqGAN Problem Statement

Given a dataset of real-world structured sequences, train a theta-parameterized generative model G to produce a sequence

$$Y_{1:T} = (y_1, \dots, y_t, \dots, y_T), y_t \in \mathcal{Y},$$

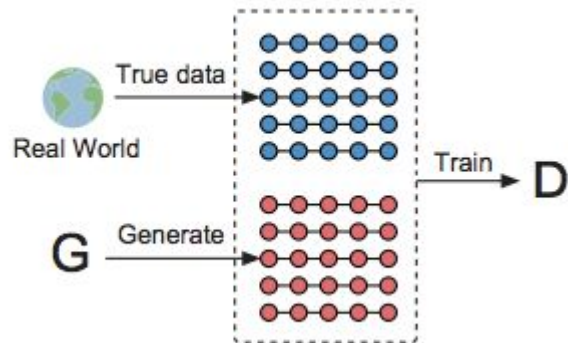
Where \mathcal{Y} is the vocabulary of candidate tokens.

Traditional Approach: Maximum Likelihood Estimation - exposure bias (trained on data distribution but tested on model distribution - quickly accumulate error)

$$\max_{\theta} \frac{1}{|D|} \sum_{Y_{1:T} \in D} \sum_t \log[G_{\theta}(y_t | Y_{1:t-1})]$$

SeqGAN Approach - GAN (Goodfellow)

- Generator G
 - Noise vector for entropy
 - Generate real-looking data to fool discriminator
- Discriminator D
 - Output real value for $P(\text{real data})$ versus $P(\text{fake data})$
 - Essentially a classifier between real and fake
- Minimax game:



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

SeqGAN Approach - Reinforcement Learning

- Generator G as an agent of RL - sequential decision making
- S : set of states $s \in \mathbf{S}$ - tokens generated so far
- Policy $G_\theta(y_t|Y_{1:t-1})$ - determine the next token to generate given previous state
- Reward model is the discriminator $D_\phi(Y_{1:T}^n) = P(\text{real})$

Approach - RL Objective

- Generator loss: Maximize expected reward

$$J(\theta) = \mathbb{E}[R_T | s_0, \theta] = \sum_{y_1 \in \mathcal{Y}} G_\theta(y_1 | s_0) \cdot Q_{D_\phi}^{G_\theta}(s_0, y_1)$$

- Q-function is state-action value, which is how good it is for agent to take action y (next possible token) at state s .

$$Q_{D_\phi}^{G_\theta}(a = y_T, s = Y_{1:T-1}) = D_\phi(Y_{1:T}).$$

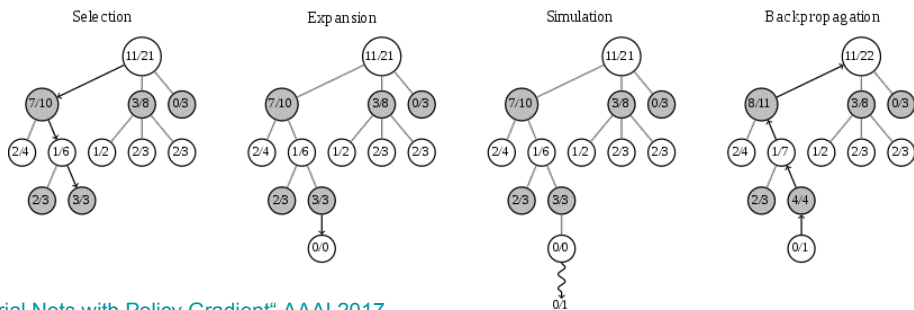
Monte Carlo Tree Search (AlphaGo)

- Q only calculated for full sequence, use Monte Carlo tree search for intermediates (4)

$$\{Y_{1:T}^1, \dots, Y_{1:T}^N\} = \text{MC}^{G_\beta}(Y_{1:t}; N)$$

$$Q_{D_\phi}^{G_\theta}(s = Y_{1:t-1}, a = y_t) = \begin{cases} \frac{1}{N} \sum_{n=1}^N D_\phi(Y_{1:T}^n), & Y_{1:T}^n \in \text{MC}^{G_\beta}(Y_{1:t}; N) & \text{for } t < T \\ D_\phi(Y_{1:t}) & & \text{for } t = T, \end{cases}$$

- Algorithm:
 - **Selection:** start from root R and select successive child nodes down to a leaf node L .
 - **Expansion:** unless L ends the sequence, create one (or more) child nodes and choose node C from one of them.
 - **Simulation:** play a random playout from node C . This step is sometimes also called playout or rollout.
 - **Backpropagation:** use the result of the playout to update information in the nodes on the path from C to R .



Training

- Generator - policy gradient to maximize long-term reward (8)

$$J(\theta) = \mathbb{E}[R_T | s_0, \theta] = \sum_{y_1 \in \mathcal{Y}} G_\theta(y_1 | s_0) \cdot Q_{D_\phi}^{G_\theta}(s_0, y_1)$$

$$\nabla_\theta J(\theta) = \sum_{t=1}^T \mathbb{E}_{Y_{1:t-1} \sim G_\theta} \left[\sum_{y_t \in \mathcal{Y}} \nabla_\theta G_\theta(y_t | Y_{1:t-1}) \cdot Q_{D_\phi}^{G_\theta}(Y_{1:t-1}, y_t) \right]$$

$$Q_{D_\phi}^{G_\theta}(a = y_T, s = Y_{1:T-1}) = D_\phi(Y_{1:T}).$$

$$\theta \leftarrow \theta + \alpha_h \nabla_\theta J(\theta)$$

- Discriminator - classification into two classes: real/fake, same as Goodfellow (5)

$$\min_{\phi} -\mathbb{E}_{Y \sim p_{\text{data}}} [\log D_\phi(Y)] - \mathbb{E}_{Y \sim G_\theta} [\log(1 - D_\phi(Y))].$$

- Start with MLE to pretrain
- Train generator in g steps, then discriminator, alternating

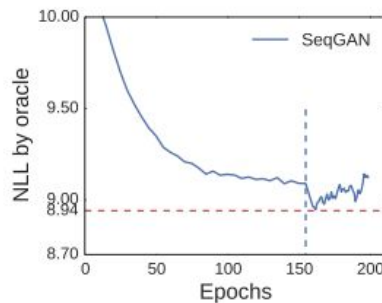
Algorithm 1 Sequence Generative Adversarial Nets

Require: generator policy G_θ ; roll-out policy G_β ; discriminator D_ϕ ; a sequence dataset $\mathcal{S} = \{X_{1:T}\}$

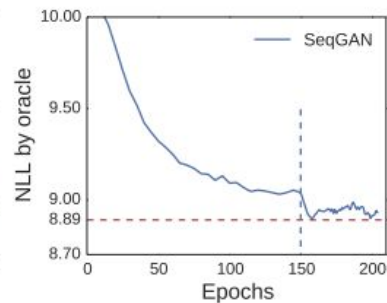
- 1: Initialize G_θ, D_ϕ with random weights θ, ϕ .
- 2: Pre-train G_θ using MLE on \mathcal{S}
- 3: $\beta \leftarrow \theta$
- 4: Generate negative samples using G_θ for training D_ϕ
- 5: Pre-train D_ϕ via minimizing the cross entropy
- 6: **repeat**
- 7: **for** g-steps **do**
- 8: Generate a sequence $Y_{1:T} = (y_1, \dots, y_T) \sim G_\theta$
- 9: **for** t in $1 : T$ **do**
- 10: Compute $Q(a = y_t; s = Y_{1:t-1})$ by Eq. (4)
- 11: **end for**
- 12: Update generator parameters via policy gradient Eq. (8)
- 13: **end for**
- 14: **for** d-steps **do**
- 15: Use current G_θ to generate negative examples and combine with given positive examples \mathcal{S}
- 16: Train discriminator D_ϕ for k epochs by Eq. (5)
- 17: **end for**
- 18: $\beta \leftarrow \theta$
- 19: **until** SeqGAN converges

Models

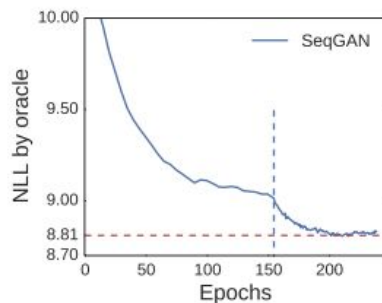
- G - RNN with LSTM
- D - CNN



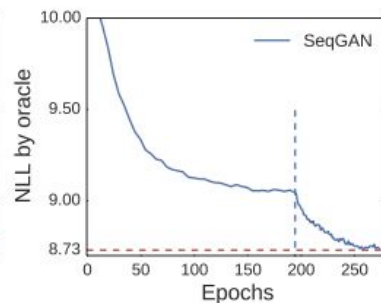
(a) $g\text{-steps}=100$, $d\text{-steps}=1$, $k=10$



(b) $g\text{-steps}=30$, $d\text{-steps}=1$, $k=30$



(c) $g\text{-steps}=1$, $d\text{-steps}=1$, $k=10$



(d) $g\text{-steps}=1$, $d\text{-steps}=5$, $k=3$

Experiments

Obama Speech Text Generation

- when he was told of this extraordinary honor that he was the most trusted man in america
 - but we also remember and celebrate the journalism that walter practiced a standard of honesty and integrity and responsibility to which so many of you have committed your careers. it's a standard that's a little bit harder to find today
 - i am honored to be here to pay tribute to the life and times of the man who chronicled our time.
- i stood here today i have one and most important thing that not on violence throughout the horizon is OTHERS american fire and OTHERS but we need you are a strong source
 - for this business leadership will remember now i can't afford to start with just the way our european support for the right thing to protect those american story from the world and
 - i want to acknowledge you were going to be an outstanding job times for student medical education and warm the republicans who like my times if he said is that brought the

Human

Machine

- Chinese poem generation

南陌春风早，东邻去日斜。

紫陌追随日，青门相见时。

胡风不开花，四气多作雪。

Human

山夜有雪寒，桂里逢客时。

此时人且饮，酒愁一节梦。

四面客归路，桂花开青竹。

Machine

Table 2: Chinese poem generation performance comparison.

Algorithm	Human score	p -value	BLEU-2	p -value
MLE	0.4165	0.0034	0.6670	$< 10^{-6}$
SeqGAN	0.5356		0.7389	
Real data	0.6011		0.746	

Table 3: Obama political speech generation performance.

Algorithm	BLEU-3	p -value	BLEU-4	p -value
MLE	0.519	$< 10^{-6}$	0.416	0.00014
SeqGAN	0.556		0.427	

Table 4: Music generation performance comparison.

Algorithm	BLEU-4	p -value	MSE	p -value
MLE	0.9210	$< 10^{-6}$	22.38	0.00034
SeqGAN	0.9406		20.62	

Oracle LSTM with $N(0,1)$ as real distribution

Table 1: Sequence generation performance comparison. The p -value is between SeqGAN and the baseline from T-test.

Algorithm	Random	MLE	SS	PG-BLEU	SeqGAN
NLL	10.310	9.038	8.985	8.946	8.736
p -value	$< 10^{-6}$	$< 10^{-6}$	$< 10^{-6}$	$< 10^{-6}$	

Discrete GAN - Rationale

- Similar to SeqGAN
- Back-propagation difficulties with discrete random variables
- Inherent instability of GAN training objective

Approach

- Discrete problem - RL by using $\log(D)$ as reward
- Maximum likelihood along with discriminator as training signals
- Novel generator training objective that reduces variance
- Importance sampling to make objective trainable

MLE Augmented Discrete GAN (MaliGAN)

- Delayed copy of generator $p'(\mathbf{x})$
- Optimal discriminator property $D(\mathbf{x}) = \frac{p_d}{p_d + p'}$ $p_d = \frac{D}{1-D}p'$
- q target distribution for MLE = $\frac{D}{1-D}p'$
- Let $r_D(\mathbf{x}) = \frac{D(\mathbf{x})}{1-D(\mathbf{x})}$
- Augmented target distribution: $q(\mathbf{x}) = \frac{1}{Z(\theta')} \frac{D(\mathbf{x})}{1-D(\mathbf{x})} p'(\mathbf{x}) = \frac{r_D(\mathbf{x})}{Z(\theta')} p'(\mathbf{x})$
- Regarding q as fixed, target to optimize is

$$L_G(\theta) = \text{KL}(q(\mathbf{x}) || p_\theta(\mathbf{x}))$$

$$\nabla L_G = \mathbb{E}_q[\nabla_\theta \log p_\theta(\mathbf{x})]$$

- From importance sampling: $\nabla L_G = \mathbb{E}_{p'}\left[\frac{q(\mathbf{x})}{p'(\mathbf{x})} \nabla_\theta \log p_\theta(\mathbf{x})\right] = \frac{1}{Z} \mathbb{E}_{p_\theta}[r_D(\mathbf{x}) \nabla_\theta \log p_\theta(\mathbf{x})]$

$$\nabla L_G(\theta) \approx \sum_{i=1}^m \left(\frac{r_D(\mathbf{x}_i)}{\sum_i r_D(\mathbf{x}_i)} - b \right) \nabla \log p_\theta(\mathbf{x}_i) = E(\{\mathbf{x}_i\}_1^m)$$

Algorithm 1 MaliGAN

Require: A generator p with parameters θ .

A discriminator $D(x)$ with parameters θ_d .

A baseline b .

1: **for** number of training iterations **do**

2: **for** k steps **do**

3: Sample a minibatch of samples $\{\mathbf{x}_i\}_{i=1}^m$ from p_θ .

4: Sample a minibatch of samples $\{\mathbf{y}_i\}_{i=1}^m$ from p_d .

5: Update the parameter of discriminator by taking gradient ascend of discriminator loss

$$\sum_i [\nabla_{\theta_d} \log D(\mathbf{y}_i)] + \sum_i [\nabla_{\theta_d} \log(1 - D(\mathbf{x}_i))]$$

6: **end for**

7: Sample a minibatch of samples $\{\mathbf{x}_i\}_{i=1}^m$ from p_θ .

8: Update the generator by applying gradient update

$$\sum_{i=1}^m \left(\frac{r_D(\mathbf{x}_i)}{\sum_i r_D(\mathbf{x}_i)} - b \right) \nabla \log p_\theta(\mathbf{x}_i)$$

9: **end for**

Variance Reduction in MaliGAN

- MC Tree Search (similar to SeqGAN)
- Mixed MLE-Mali Training

$$\begin{aligned}\nabla L_G^N &\approx \sum_{i=1, j=1}^{m, n} \left(\frac{r_D(\mathbf{x}_{i,j})}{\sum_j r_D(\mathbf{x}_{i,j})} - b \right) \nabla \log p_\theta(\mathbf{x}_{i,j}^{>N} | \mathbf{x}_i^{\leq N}) \\ &+ \frac{1}{m} \sum_{i=1}^m \sum_{t=0}^N p_\theta(a_t^i | \mathbf{s}_t^i) = E_N(\mathbf{x}_{i,j})\end{aligned}$$

Algorithm 2 Sequential MaliGAN with Mixed MLE Training

Require: A generator p with parameters θ .
 A discriminator $D(x)$ with parameters θ_d .
 Maximum sequence length T , step size K .
 A baseline b , sampling multiplicity m .

- 1: $N = T$
- 2: Optional: Pretrain model using pure MLE with some epochs.
- 3: **for** number of training iterations **do**
- 4: $N = N - K$
- 5: **for** k steps **do**
- 6: Sample a minibatch of sequences $\{\mathbf{y}_i\}_{i=1}^m$ from p_d .
- 7: While keeping the first N steps the same as $\{\mathbf{y}_i\}_{i=1}^m$, sample a minibatch of sequences $\{\mathbf{x}_i\}_{i=1}^m$ from p_θ from time step N .
- 8: Update the discriminator by taking gradient ascend of discriminator loss.

$$\sum_i [\nabla_{\theta_d} \log D(\mathbf{y}_i)] + \sum_i [\nabla_{\theta_d} \log(1 - D(\mathbf{x}_i))]$$

- 9: **end for**
- 10: Sample a minibatch of sequences $\{\mathbf{x}_i\}_{i=1}^m$ from p_d .
- 11: For each sample \mathbf{x}_i with length larger than N in the minibatch, clamp the generator to the first N words of s , and freely run the model to generate m samples $\mathbf{x}_{i,j}$, $j = 1, \dots, m$ till the end of the sequence.
- 12: Update the generator by applying the mixed MLE-Mali gradient update

$$\begin{aligned} \nabla L_G^N \approx & \sum_{i=1, j=1}^{m, n} \left(\frac{r_D(\mathbf{x}_{i,j})}{\sum_j r_D(\mathbf{x}_{i,j})} - b \right) \nabla \log p_\theta(\mathbf{x}_{i,j}^{>N} | \mathbf{x}_i^{\leq N}) \\ & + \frac{1}{m} \sum_{i=1}^m \sum_{t=0}^N p_\theta(a_t^i | \mathbf{s}_t^i) \end{aligned}$$

- 13: **end for**
-

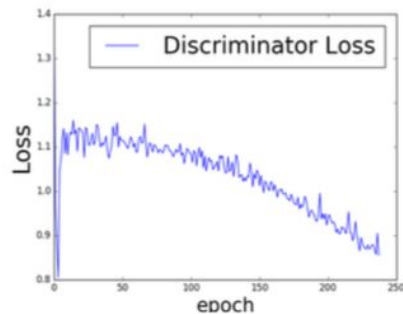
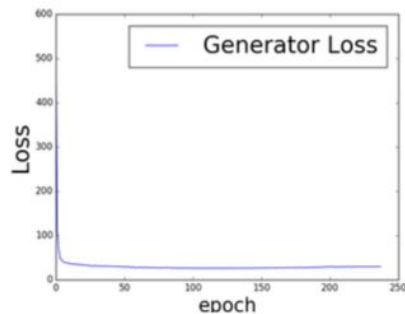


Figure 1. The training loss of the generator (left) and the discriminator (right) of MaliGAN on Discrete MNIST task.



Figure 2. Samples generated by REINFORCE-like model (left) and by MaliGAN (right).

Table 1. Experimental results on Poetry Generation task. The result of SeqGAN is directly taken from (Yu et al., 2017).

Model	Poem-5		Poem-7	
	BLEU-2	PPL	BLEU-2	PPL
MLE	0.6934	564.1	0.3186	192.7
SeqGAN	0.7389	-	-	-
MaliGAN-basic	0.7406	548.6	0.4892	182.2
MaliGAN-full	0.7628	542.7	0.5526	180.2

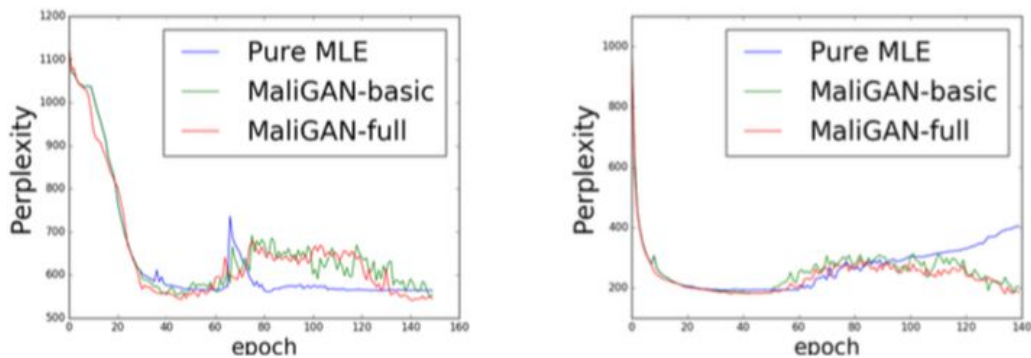
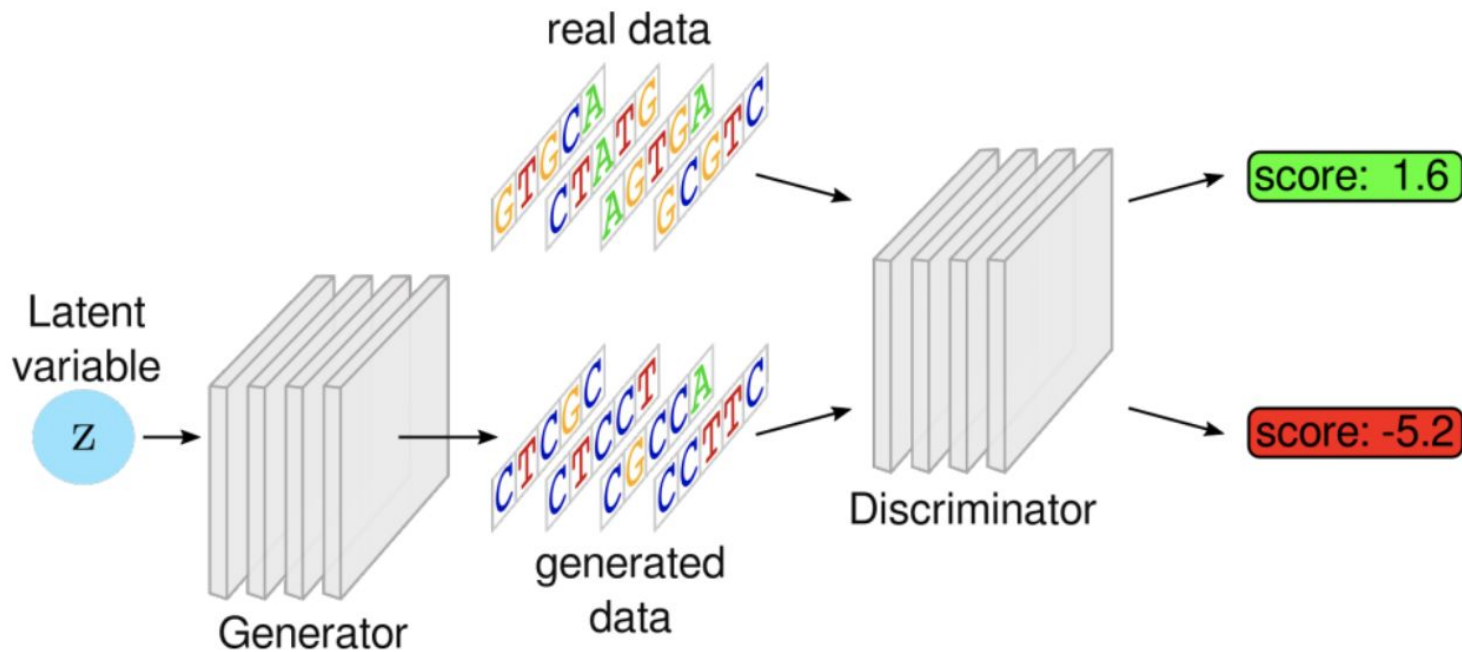


Figure 3. Perplexity curves on Poem-5 (left) and Poem-7 (right).

Table 2. Experimental results on PTB. Note that we evaluate the models in sentence-level.

	MLE	MaliGAN-basic	MaliGAN-full
Valid-Perplexity	141.9	131.6	128.0
Test-Perplexity	138.2	125.3	123.8

DNA Sequence Generation



Contents

1. Image Generation
2. Conditional GAN/Domain Transfer
3. Sequence Generation
- 4. 3D Scene Reconstruction/Lower-Higher Dimension**
5. Better Training

3D GAN

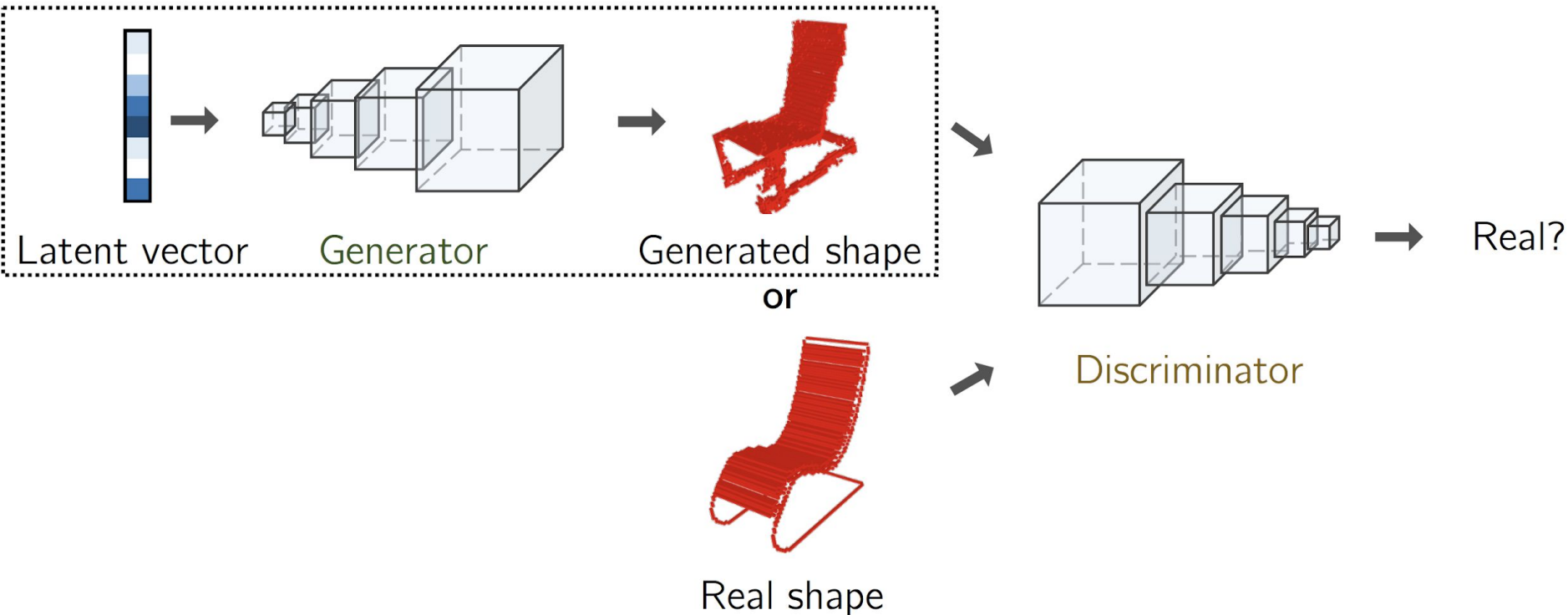
Reconstruction from lower dimensions

- Transformation into higher dimension using data from lower dimension

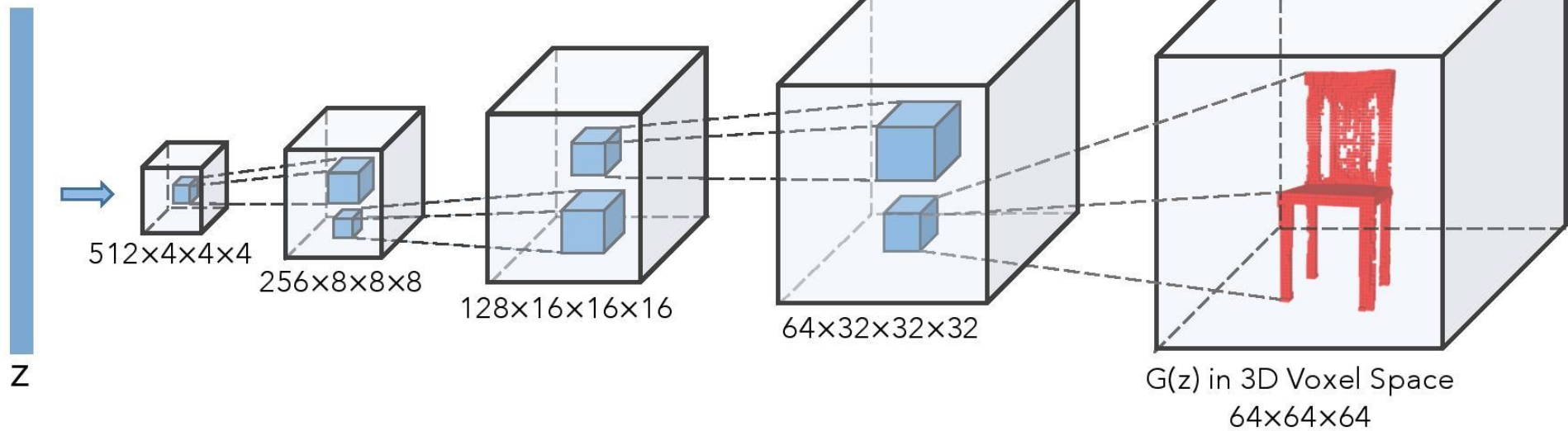
$$(X : \mathbb{R}, z) \rightarrow X : \mathbb{R}^n$$

- Protein synthesis
- 3D Scene reconstruction from 2D projection
- Image super-resolution

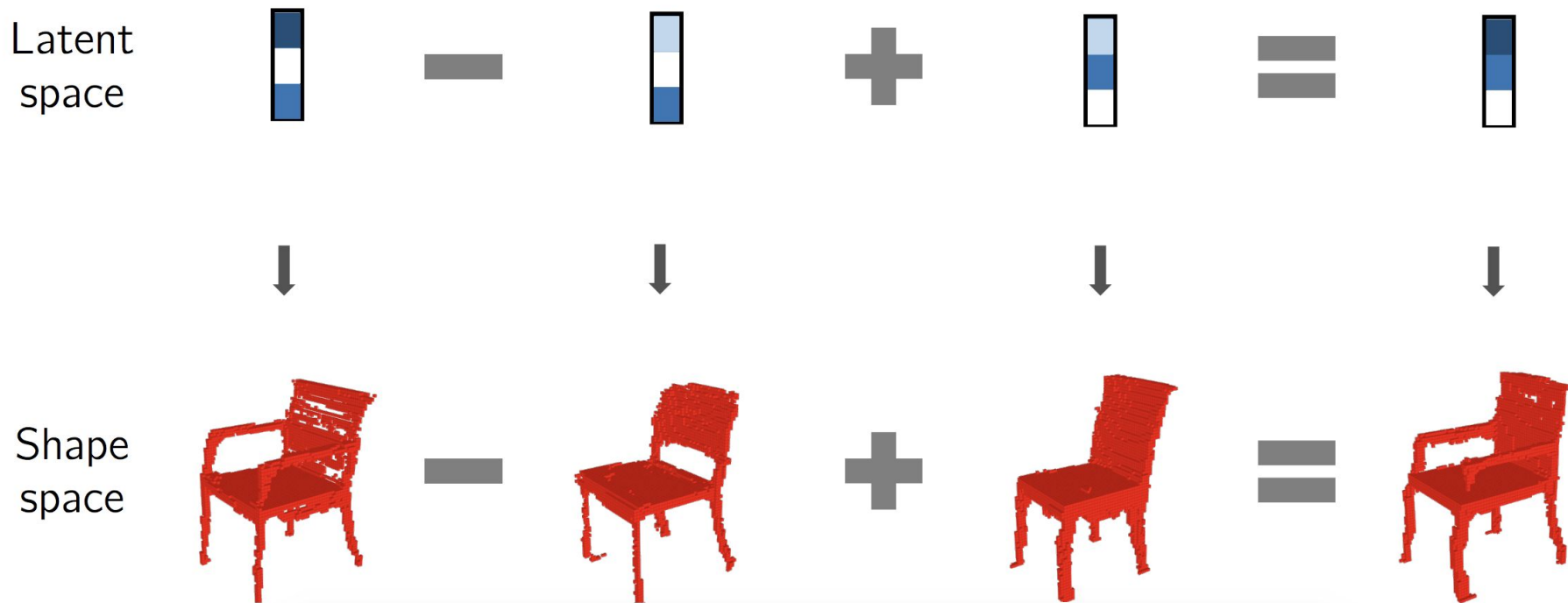
3D Generative Adversarial Network

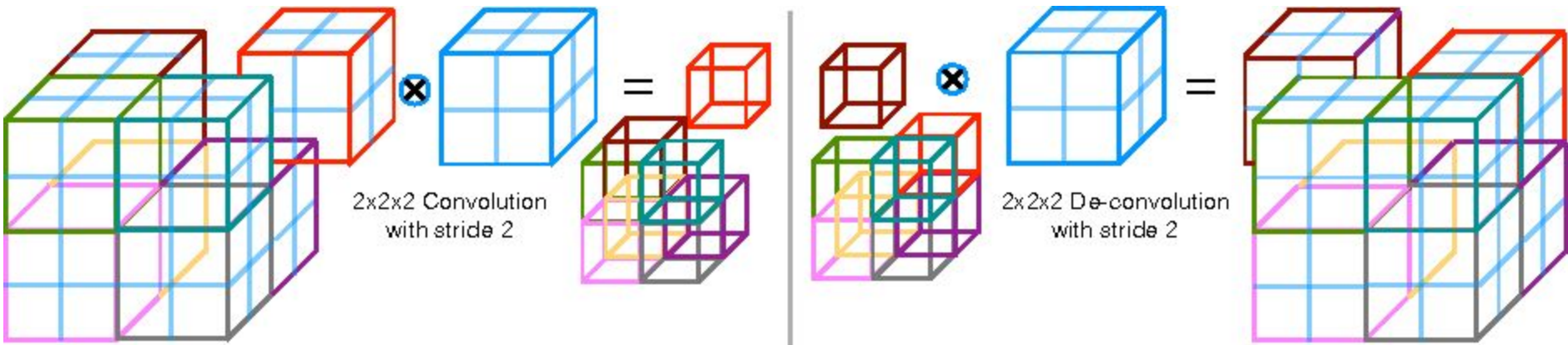


Training on ShapeNet [Chang et al., 2015]



Arithmetic in Latent Space





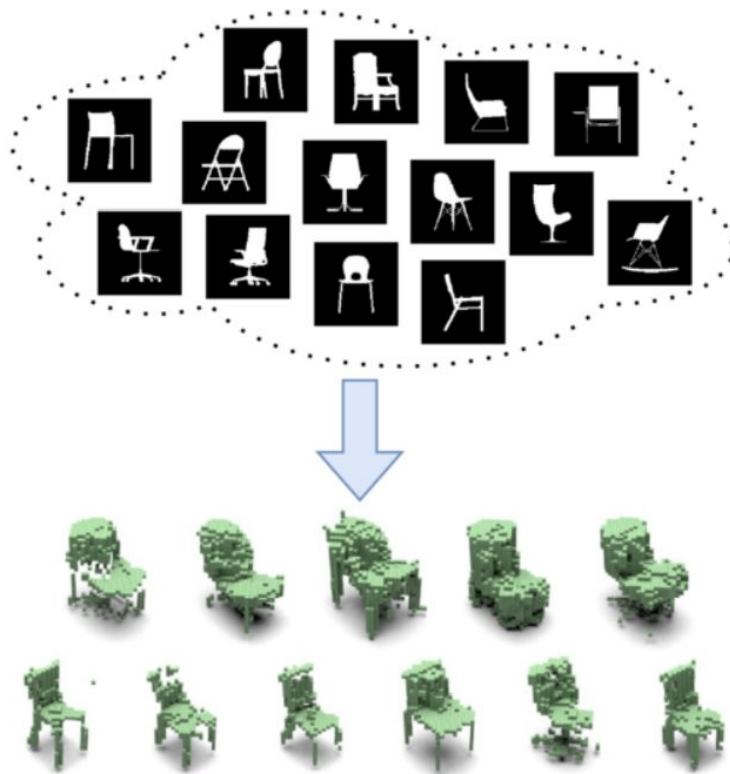


Figure 1. Given a collection of 2D views of multiple objects, our algorithm infers a generative model of the underlying 3D shapes.

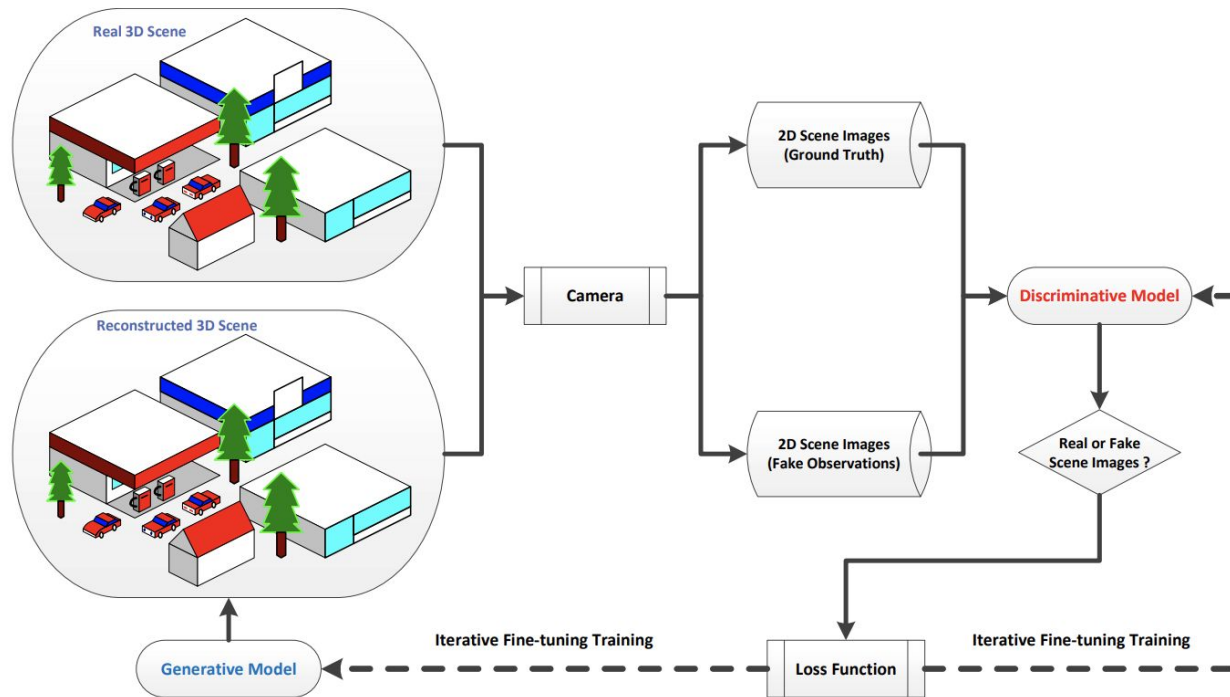


Figure 2: Framework and workflow chart of 3D-Scene-GAN.

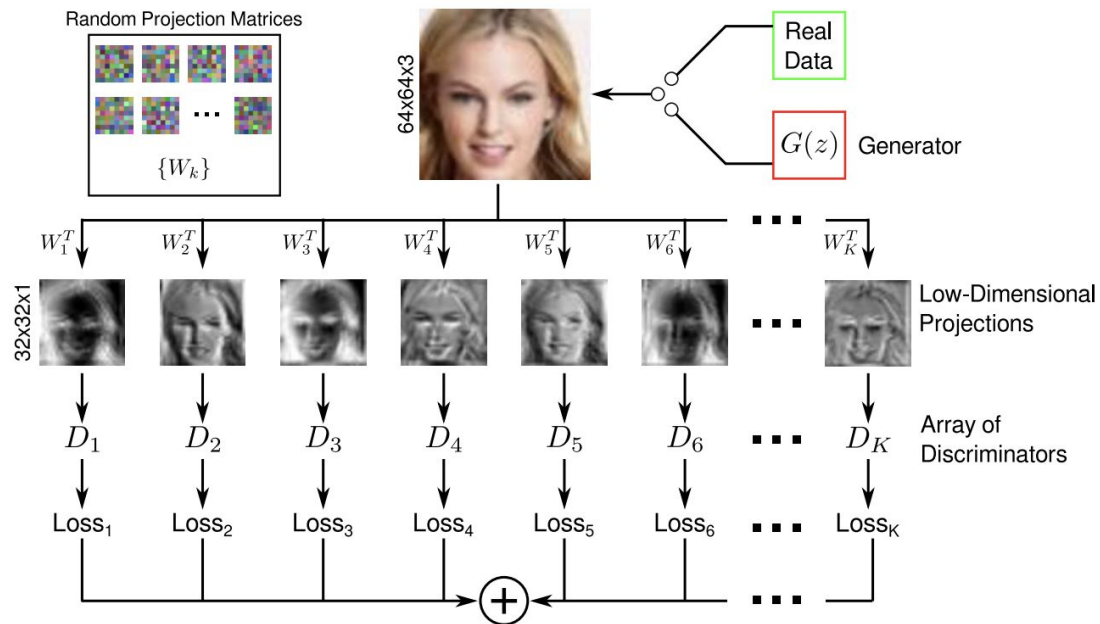


Figure 1: Overview of our approach. We train a single generator against an array of discriminators, each of which receives lower-dimensional projections—chosen randomly prior to training—as input. Individually, these discriminators are unable to perfectly separate real and generated samples, and thus provide stable gradients to the generator throughout training. In turn, by trying to fool all the discriminators simultaneously, the generator learns to match the true full data distribution.

Reconstruction from lower dimensions - AmbientGAN

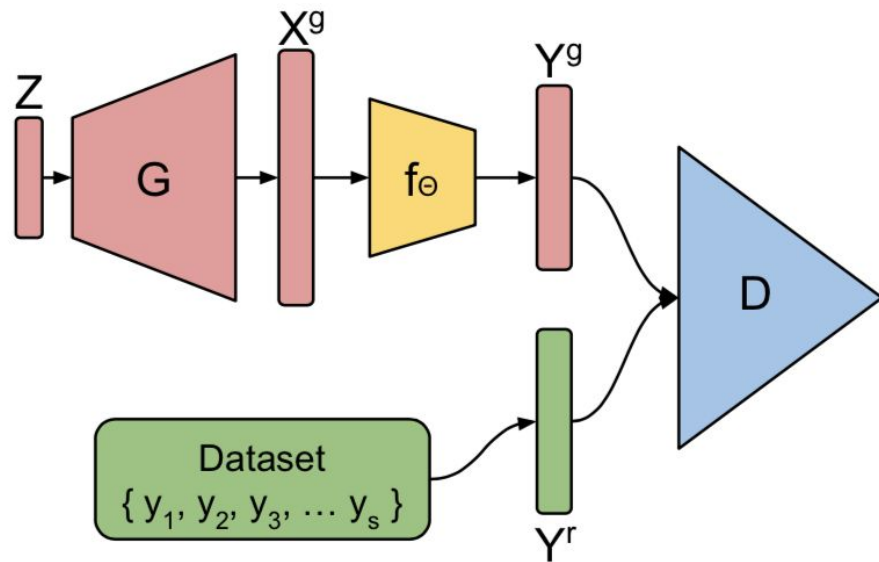


Figure 1: AmbientGAN training. The output of the generator is passed through a simulated random measurement function f_Θ . The discriminator must decide if a measurement is real or generated.

AmbientGAN - Introduction

- Problem: current GAN techniques require access to fully-observed samples.
- Fully observed samples are required for training but are expensive.
- Task: learn implicit generative model given only **lossy measurements** of samples from the distribution of interest.
 - Lossy measurements = noisy/distorted/incomplete samples

Related Work

- David Berthelot, Tom Schumm, and Luke Metz. Began: Boundary equilibrium generative adversarial networks. 2017. [Link](#)
- Low dimensional projections of data:
 - Matheus Gadelha, Subhansu Maji, and Rui Wang. 3d shape induction from 2d views of multiple objects. 2016. [Link](#)
 - Behnam Neyshabur, Srinadh Bhojanapalli, and Ayan Chakrabarti. Stabilizing gan training with multiple random projections. arXiv preprint arXiv:1705.07831, 2017. [Link](#)
- Maya Kabkab, Pouya Samangouei, Rama Chellappa. Task-Aware Compressed Sensing with Generative Adversarial Networks. [Link](#)
- [GAN](#) (Goodfellow) and [WGAN](#) for objective function.

Approach

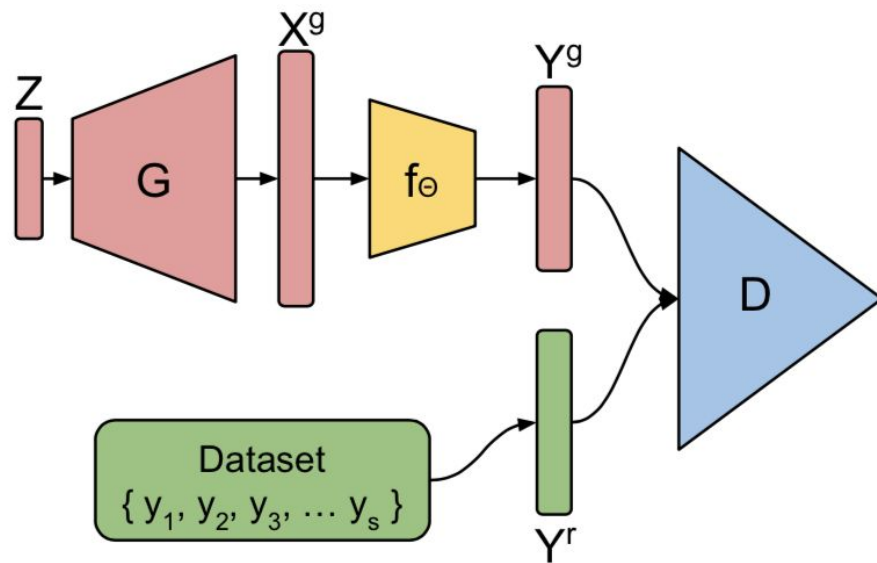


Figure 1: AmbientGAN training. The output of the generator is passed through a simulated random measurement function f_{Θ} . The discriminator must decide if a measurement is real or generated.

Notation

Let:

'r' denote real/true distribution,

'g' denote generated distributions,

'x' denote underlying space,

'y' denote measurements (incomplete samples).

p_x^r be the real underlying distribution over \mathbb{R}^n .

Approach

- Goal: learn a generator G such that p_x^g is close to p_x^r .
- However, do not have access to $X \sim p_x^r$

Observe lossy measurements of size m performed on samples from p_x^r . Assume that we have a measurement function $f_\theta : \mathbb{R}^n \rightarrow \mathbb{R}^m$. Thus measurements are given by $y = f_\theta(x)$ and the distribution over the measurements is given by p_y^r . If $X \sim p_x^r$ and $\theta \sim p_\theta$ then $Y = f_\theta(X) \sim p_y^r$.

Assumptions:

- we have the measurement function $f_\theta(x)$ and p_θ
- easy to sample $\theta \sim p_\theta$
- easy to compute $f_\theta(x)$ for any x and θ

Approach

Steps:

- sample random measurement function f_θ
- $X^g = G(Z) \sim p_x^g$
- $D : \mathbb{R}^m \rightarrow \mathbb{R}$ predicts if y is real from p_y^r or generated from p_y^g

$$\min_G \max_D \mathbb{E}_{Y^r \sim p_y^r} [q(D(Y^r))] + \mathbb{E}_{Z \sim p_z, \Theta \sim p_\theta} [q(1 - D(f_\Theta(G(Z))))]$$

$$q(x) = \log(x) \text{ for GAN}$$

$$q(x) = x \text{ for WGAN}$$

Model Architectures

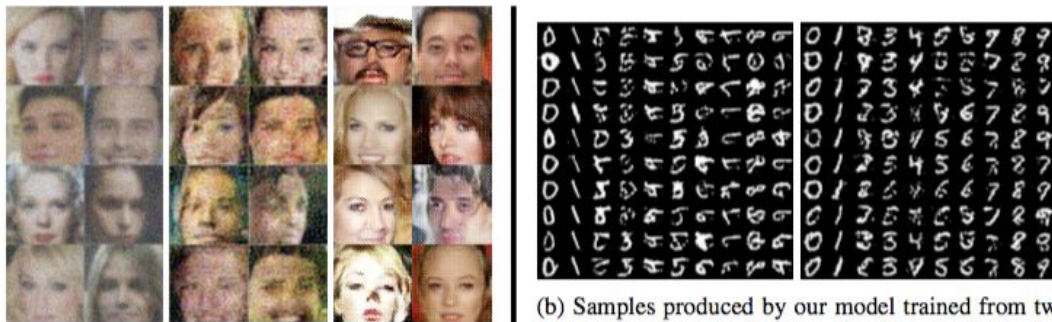
- Conditional DCGAN
- Unconditional WGAN with gradient penalty
- Goodfellow GAN

Theoretical Results

The mapping of distributions of samples p_x^r to distribution of measurements p_y^r is invertible even though the map from individual image x to its measurement $f_\theta(x)$ is not.



Figure 2: (Left) Samples of lossy measurements used for training. Samples produced by (middle) a baseline that trains from inpainted images, and (right) our model.



(a) (left) Samples of lossy measurements. Each image is a blurred noisy version of the original. Samples produced by (middle) a baseline that uses Wiener deconvolution, and (right) our model.

(b) Samples produced by our model trained from two 1D projections of each image. On left, the training data does not include the angle of the projections, so it cannot identify orientation or chirality. On right, the training data includes the angle.

Figure 3: Results with Convolve+Noise on celebA (left) and 1D-projections on MNIST (right).

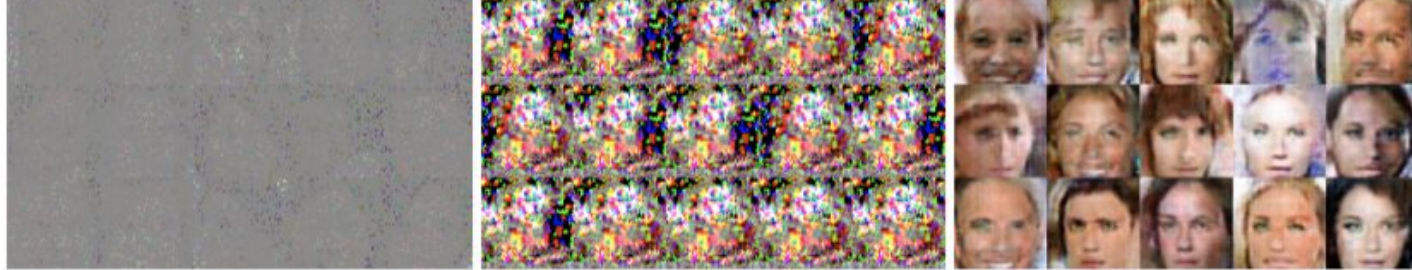
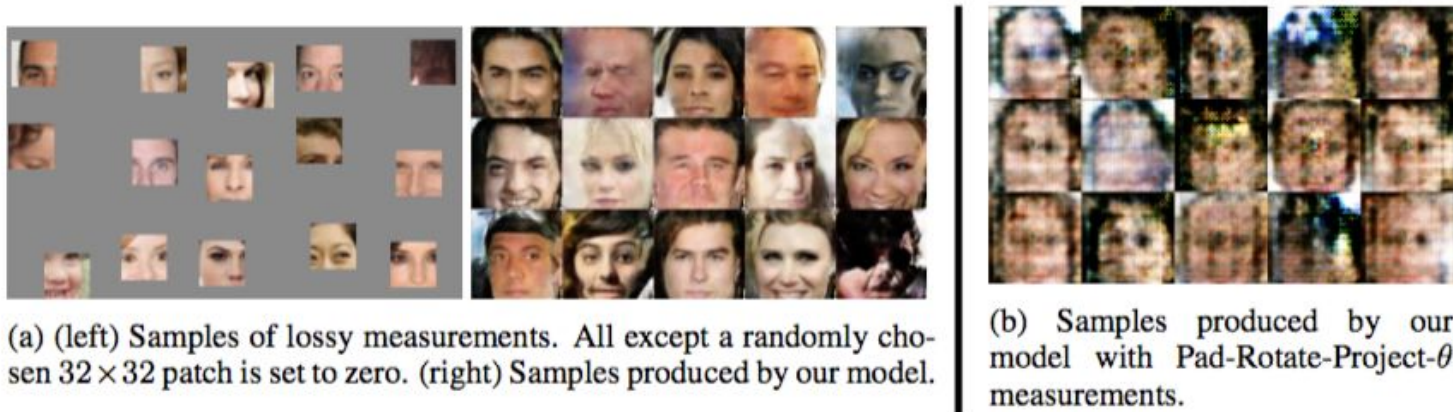


Figure 4: Results with Block-Pixels on celebA. (left) Samples of lossy measurements. Each pixel is blocked independently with probability $p = 0.95$. Samples produced by (middle) unmeasure-blur baseline, and (right) our model.



(a) (left) Samples of lossy measurements. All except a randomly chosen 32×32 patch is set to zero. (right) Samples produced by our model.

(b) Samples produced by our model with Pad-Rotate-Project- θ measurements.

Figure 5: Results on celebA with (a) Keep-Patch, and (b) 1D projections.



Figure 6: Results with Block-Pixels on CIFAR-10. (left) Samples of lossy measurements. Each pixel is blocked independently with probability $p = 0.8$. Samples produced by (middle) unmeasure-blur baseline, and (right) our model.

Contents

1. Image Generation/Noisy Image
 - a. ambient
2. Conditional GAN/Domain Transfer
3. Sequence Generation
4. 3D Scene Reconstruction/Lower-Higher Dimension
 - a. StackGAN - text to image
5. **Better Training**

Progressive Growing Generative Adversarial Nets

- **Faster, more stable training methodology through progressive growing**
- **Image generation with unprecedented quality**
- Propose simple way to increase variation in generated images
- Training tips for discouraging unhealthy competition between generator and discriminator
- Provide a new way of evaluating GAN results



Progressive GAN - Previous Problems

- Gradient instability
- Higher resolution makes generated images easier to tell apart from training images
- Memory constraints on higher resolutions

Progressive GAN - Related Work

- Generating high-resolution images
 - Qifeng Chen and Vladlen Koltun. Photographic image synthesis with cascaded refinement networks. CoRR, abs/1707.09405, 2017. [Link](#)
 - Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional image synthesis with auxiliary classifier GANs. In ICML, 2017. [Link](#)
- Growing GANs progressively
 - Zhou Wang, Eero P. Simoncelli, and Alan C. Bovik. Multi-scale structural similarity for image quality assessment. In Proc. IEEE Asilomar Conf. on Signals, Systems, and Computers, pp. 1398–1402, 2003. [Link](#)
 - Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaolei Huang, Xiaogang Wang, and Dimitris N. Metaxas. StackGAN: text to photo-realistic image synthesis with stacked generative adversarial networks. In ICCV, 2017. [Link](#)
 - Arnab Ghosh, Viveka Kulharia, Vinay P. Namboodiri, Philip H. S. Torr, and Puneet Kumar Dokania. Multi-agent diverse generative adversarial networks. CoRR, abs/1704.02906, 2017. [Link](#)

Progressive GAN - Related Work Continued

- Solving gradient problems
 - Ishaan Gulrajani, Faruk Ahmed, Martín Arjovsky, Vincent Dumoulin, and Aaron C. Courville. Improved training of Wasserstein GANs. *CoRR*, abs/1704.00028, 2017. [Link](#)
 - Martin Arjovsky and Leon Bottou. Towards principled methods for training generative adversarial networks. In ICLR, 2017. [Link](#)
- Measuring GAN performance
 - Tim Salimans, Ian J. Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training GANs. In NIPS, 2016. [Link](#)
 - MS-SSIM: Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional image synthesis with auxiliary classifier GANs. In ICML, 2017. [Link](#)

Progressive GAN - Method

- Start with low-resolution images
- Progressively increase resolution by adding network layers
- Learn high level structure first, then detail
- Benefits:
 - Reduced training time
 - Substantially more stable training, especially early with smaller images

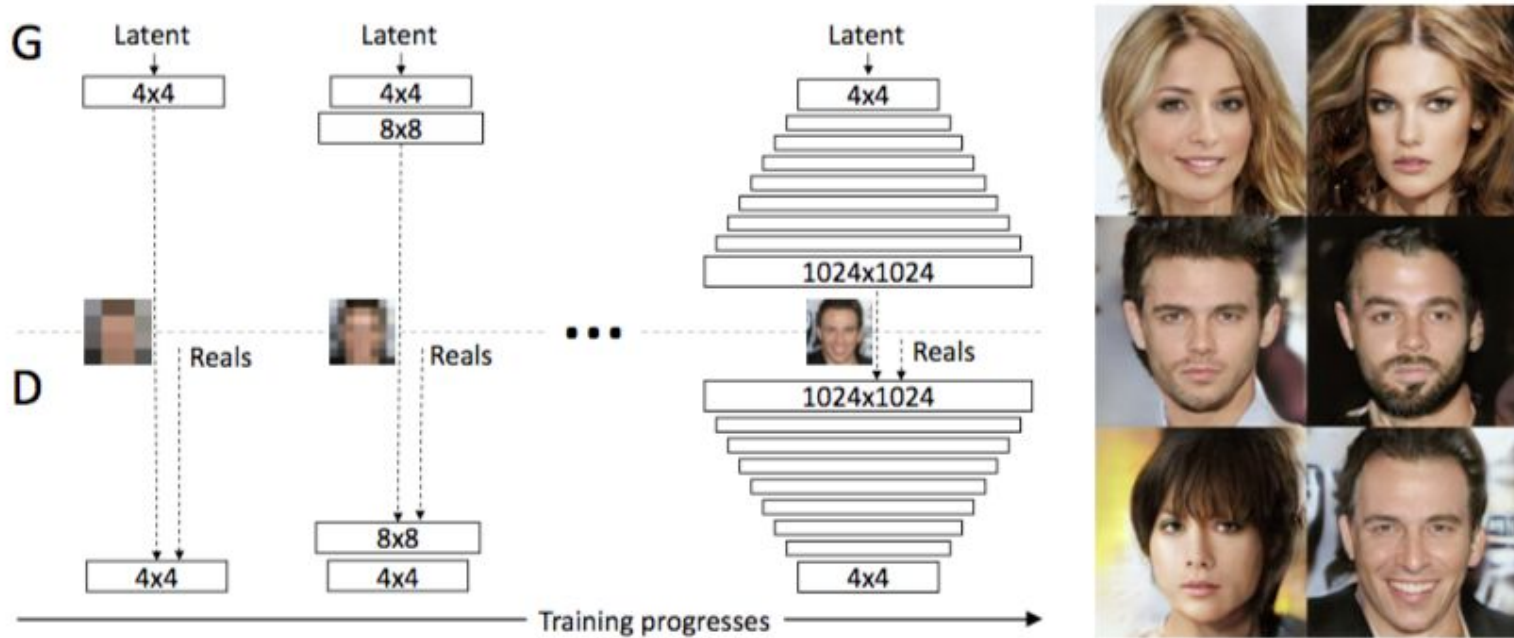


Figure 1: Our training starts with both the generator (G) and discriminator (D) having a low spatial resolution of 4×4 pixels. As the training advances, we incrementally add layers to G and D, thus increasing the spatial resolution of the generated images. All existing layers remain trainable throughout the process. Here $N \times N$ refers to convolutional layers operating on $N \times N$ spatial resolution. This allows stable synthesis in high resolutions and also speeds up training considerably. On the right we show six example images generated using progressive growing at 1024×1024 .

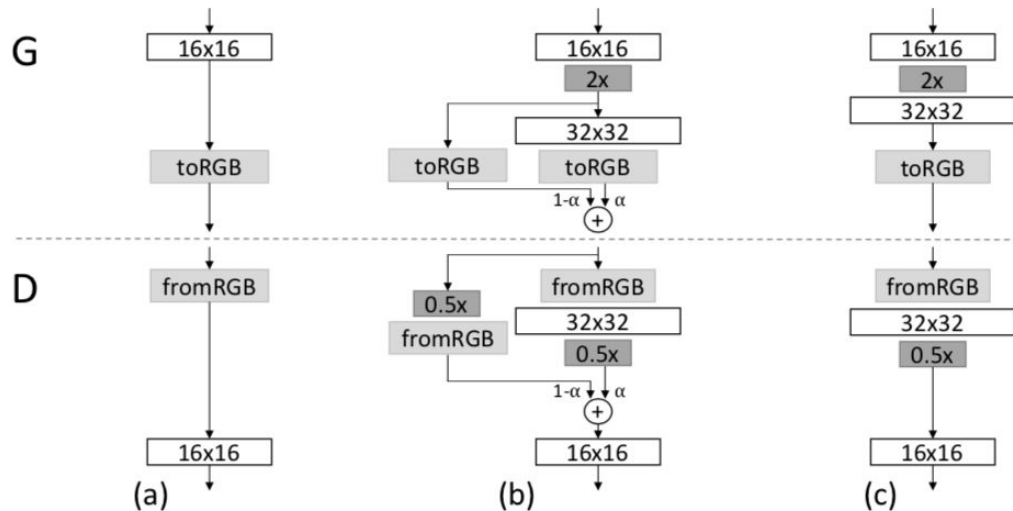


Figure 2: When doubling the resolution of the generator (G) and discriminator (D) we fade in the new layers smoothly. This example illustrates the transition from 16×16 images (a) to 32×32 images (c). During the transition (b) we treat the layers that operate on the higher resolution like a residual block, whose weight α increases linearly from 0 to 1. Here $2 \times$ and $0.5 \times$ refer to doubling and halving the image resolution using nearest neighbor filtering and average pooling, respectively. The `toRGB` represents a layer that projects feature vectors to RGB colors and `fromRGB` does the reverse; both use 1×1 convolutions. When training the discriminator, we feed in real images that are downsampled to match the current resolution of the network. During a resolution transition, we interpolate between two resolutions of the real images, similarly to how the generator output combines two resolutions.

Training configuration	CELEBA						LSUN BEDROOM					
	Sliced Wasserstein distance $\times 10^3$					MS-SSIM	Sliced Wasserstein distance $\times 10^3$					MS-SSIM
	128	64	32	16	Avg		128	64	32	16	Avg	
(a) Gulrajani et al. (2017)	12.99	7.79	7.62	8.73	9.28	0.2854	11.97	10.51	8.03	14.48	11.25	0.0587
(b) + Progressive growing	4.62	2.64	3.78	6.06	4.28	0.2838	7.09	6.27	7.40	9.64	7.60	0.0615
(c) + Small minibatch	75.42	41.33	41.62	26.57	46.23	0.4065	72.73	40.16	42.75	42.46	49.52	0.1061
(d) + Revised training parameters	9.20	6.53	4.71	11.84	8.07	0.3027	7.39	5.51	3.65	9.63	6.54	0.0662
(e*) + Minibatch discrimination	10.76	6.28	6.04	16.29	9.84	0.3057	10.29	6.22	5.32	11.88	8.43	0.0648
(e) Minibatch stddev	13.94	5.67	2.82	5.71	7.04	0.2950	7.77	5.23	3.27	9.64	6.48	0.0671
(f) + Equalized learning rate	4.42	3.28	2.32	7.52	4.39	0.2902	3.61	3.32	2.71	6.44	4.02	0.0668
(g) + Pixelwise normalization	4.06	3.04	2.02	5.13	3.56	0.2845	3.89	3.05	3.24	5.87	4.01	0.0640
(h) Converged	2.42	2.17	2.24	4.99	2.96	0.2828	3.47	2.60	2.30	4.87	3.31	0.0636

Table 1: Sliced Wasserstein distance (SWD) between the generated and training images (Section 5) and multi-scale structural similarity (MS-SSIM) among the generated images for several training setups at 128×128 . For SWD, each column represents one level of the Laplacian pyramid, and the last one gives an average of the four distances.

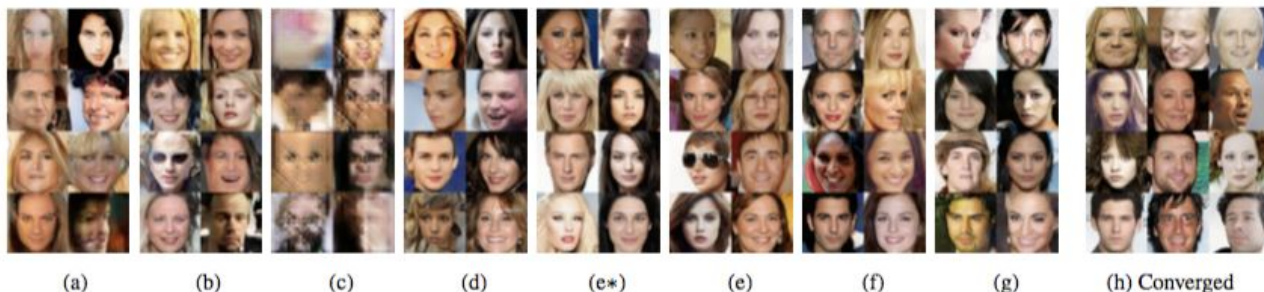


Figure 3: (a) – (g) CELEBA examples corresponding to rows in Table 1. These are intentionally non-converged. (h) Our converged result. Notice that some images show aliasing and some are not sharp – this is a flaw of the dataset, which the model learns to replicate faithfully.

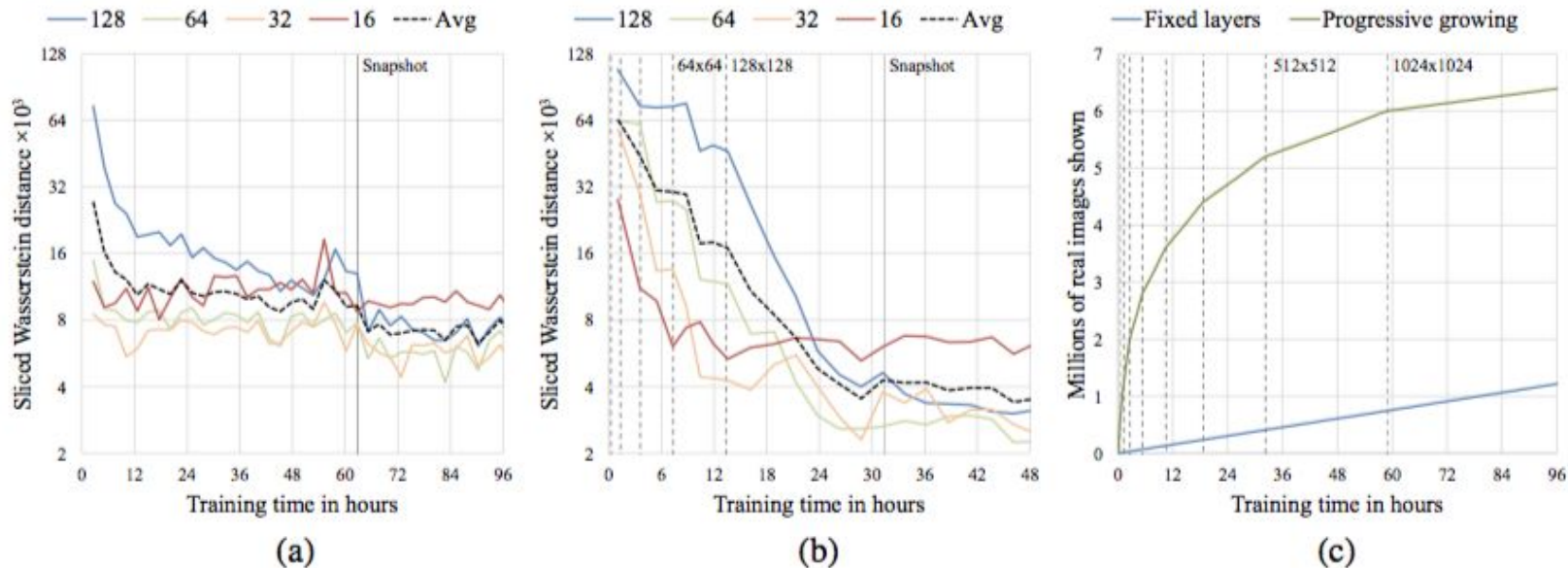


Figure 4: Effect of progressive growing on training speed and convergence. The timings were measured on a single-GPU setup using NVIDIA Tesla P100. (a) Statistical similarity with respect to wall clock time for Gulrajani et al. (2017) using CELEBA at 128×128 resolution. Each graph represents sliced Wasserstein distance on one level of the Laplacian pyramid, and the vertical line indicates the point where we stop the training in Table 1. (b) Same graph with progressive growing enabled. The dashed vertical lines indicate points where we double the resolution of G and D. (c) Effect of progressive growing on the raw training speed in 1024×1024 resolution.

Progressive GAN - Increasing Variation

- Minibatch standard deviation
- Compute the standard deviation for each feature in each spatial location over the minibatch
- Create a constant feature map using the average of these estimates
- Allows the discriminator to use these statistics internally, encourages minibatches to show similar statistics

Generator	Act.	Output shape	Params
Latent vector	–	$512 \times 1 \times 1$	–
Conv 4×4	LReLU	$512 \times 4 \times 4$	4.2M
Conv 3×3	LReLU	$512 \times 4 \times 4$	2.4M
Upsample	–	$512 \times 8 \times 8$	–
Conv 3×3	LReLU	$512 \times 8 \times 8$	2.4M
Conv 3×3	LReLU	$512 \times 8 \times 8$	2.4M
Upsample	–	$512 \times 16 \times 16$	–
Conv 3×3	LReLU	$512 \times 16 \times 16$	2.4M
Conv 3×3	LReLU	$512 \times 16 \times 16$	2.4M
Upsample	–	$512 \times 32 \times 32$	–
Conv 3×3	LReLU	$512 \times 32 \times 32$	2.4M
Conv 3×3	LReLU	$512 \times 32 \times 32$	2.4M
Upsample	–	$512 \times 64 \times 64$	–
Conv 3×3	LReLU	$256 \times 64 \times 64$	1.2M
Conv 3×3	LReLU	$256 \times 64 \times 64$	590k
Upsample	–	$256 \times 128 \times 128$	–
Conv 3×3	LReLU	$128 \times 128 \times 128$	295k
Conv 3×3	LReLU	$128 \times 128 \times 128$	148k
Upsample	–	$128 \times 256 \times 256$	–
Conv 3×3	LReLU	$64 \times 256 \times 256$	74k
Conv 3×3	LReLU	$64 \times 256 \times 256$	37k
Upsample	–	$64 \times 512 \times 512$	–
Conv 3×3	LReLU	$32 \times 512 \times 512$	18k
Conv 3×3	LReLU	$32 \times 512 \times 512$	9.2k
Upsample	–	$32 \times 1024 \times 1024$	–
Conv 3×3	LReLU	$16 \times 1024 \times 1024$	4.6k
Conv 3×3	LReLU	$16 \times 1024 \times 1024$	2.3k
Conv 1×1	linear	$3 \times 1024 \times 1024$	51
Total trainable parameters			23.1M

Discriminator	Act.	Output shape	Params
Input image	–	$3 \times 1024 \times 1024$	–
Conv 1×1	LReLU	$16 \times 1024 \times 1024$	64
Conv 3×3	LReLU	$16 \times 1024 \times 1024$	2.3k
Conv 3×3	LReLU	$32 \times 1024 \times 1024$	4.6k
Downsample	–	$32 \times 512 \times 512$	–
Conv 3×3	LReLU	$32 \times 512 \times 512$	9.2k
Conv 3×3	LReLU	$64 \times 512 \times 512$	18k
Downsample	–	$64 \times 256 \times 256$	–
Conv 3×3	LReLU	$64 \times 256 \times 256$	37k
Conv 3×3	LReLU	$128 \times 256 \times 256$	74k
Downsample	–	$128 \times 128 \times 128$	–
Conv 3×3	LReLU	$128 \times 128 \times 128$	148k
Conv 3×3	LReLU	$256 \times 128 \times 128$	295k
Downsample	–	$256 \times 64 \times 64$	–
Conv 3×3	LReLU	$256 \times 64 \times 64$	590k
Conv 3×3	LReLU	$512 \times 64 \times 64$	1.2M
Downsample	–	$512 \times 32 \times 32$	–
Conv 3×3	LReLU	$512 \times 32 \times 32$	2.4M
Conv 3×3	LReLU	$512 \times 32 \times 32$	2.4M
Downsample	–	$512 \times 16 \times 16$	–
Conv 3×3	LReLU	$512 \times 16 \times 16$	2.4M
Conv 3×3	LReLU	$512 \times 16 \times 16$	2.4M
Downsample	–	$512 \times 8 \times 8$	–
Conv 3×3	LReLU	$512 \times 8 \times 8$	2.4M
Conv 3×3	LReLU	$512 \times 8 \times 8$	2.4M
Downsample	–	$512 \times 4 \times 4$	–
Minibatch stddev	–	$513 \times 4 \times 4$	–
Conv 3×3	LReLU	$512 \times 4 \times 4$	2.4M
Conv 4×4	LReLU	$512 \times 1 \times 1$	4.2M
Fully-connected	linear	$1 \times 1 \times 1$	513
Total trainable parameters			23.1M

Table 2: Generator and discriminator that we use with CELEBA-HQ to generate 1024×1024 images.

Progressive GAN - Example Training Configuration

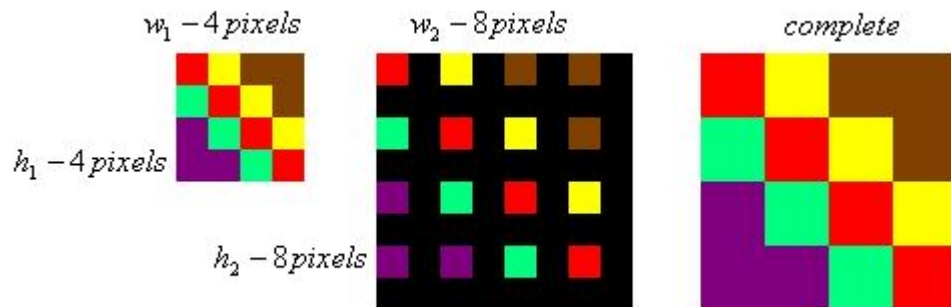
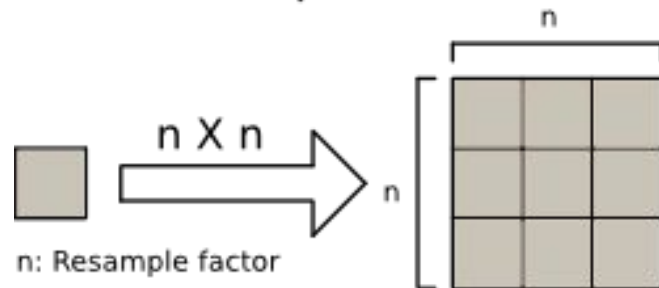
- Start with 4x4 resolution, train with 800k real images
- Then alternate:
 - Fade in the first 3-later block for the next 800k images
 - Stabilize with 800k images
- Upsampling with 2x2 pixel replication
- Downsampling with average pooling

Generator	Act.	Output shape	Params
Latent vector	–	$512 \times 1 \times 1$	–
Conv 4×4	LReLU	$512 \times 4 \times 4$	4.2M
Conv 3×3	LReLU	$512 \times 4 \times 4$	2.4M
Upsample	–	$512 \times 8 \times 8$	–
Conv 3×3	LReLU	$512 \times 8 \times 8$	2.4M
Conv 3×3	LReLU	$512 \times 8 \times 8$	2.4M
Upsample	–	$512 \times 16 \times 16$	–
Conv 3×3	LReLU	$512 \times 16 \times 16$	2.4M
Conv 3×3	LReLU	$512 \times 16 \times 16$	2.4M
Upsample	–	$512 \times 32 \times 32$	–
Conv 3×3	LReLU	$512 \times 32 \times 32$	2.4M
Conv 3×3	LReLU	$512 \times 32 \times 32$	2.4M
Upsample	–	$512 \times 64 \times 64$	–
Conv 3×3	LReLU	$256 \times 64 \times 64$	1.2M
Conv 3×3	LReLU	$256 \times 64 \times 64$	590k
Upsample	–	$256 \times 128 \times 128$	–
Conv 3×3	LReLU	$128 \times 128 \times 128$	295k
Conv 3×3	LReLU	$128 \times 128 \times 128$	148k
Upsample	–	$128 \times 256 \times 256$	–
Conv 3×3	LReLU	$64 \times 256 \times 256$	74k
Conv 3×3	LReLU	$64 \times 256 \times 256$	37k
Upsample	–	$64 \times 512 \times 512$	–
Conv 3×3	LReLU	$32 \times 512 \times 512$	18k
Conv 3×3	LReLU	$32 \times 512 \times 512$	9.2k
Upsample	–	$32 \times 1024 \times 1024$	–
Conv 3×3	LReLU	$16 \times 1024 \times 1024$	4.6k
Conv 3×3	LReLU	$16 \times 1024 \times 1024$	2.3k
Conv 1×1	linear	$3 \times 1024 \times 1024$	51
Total trainable parameters			23.1M

Discriminator	Act.	Output shape	Params
Input image	–	$3 \times 1024 \times 1024$	–
Conv 1×1	LReLU	$16 \times 1024 \times 1024$	64
Conv 3×3	LReLU	$16 \times 1024 \times 1024$	2.3k
Conv 3×3	LReLU	$32 \times 1024 \times 1024$	4.6k
Downsample	–	$32 \times 512 \times 512$	–
Conv 3×3	LReLU	$32 \times 512 \times 512$	9.2k
Conv 3×3	LReLU	$64 \times 512 \times 512$	18k
Downsample	–	$64 \times 256 \times 256$	–
Conv 3×3	LReLU	$64 \times 256 \times 256$	37k
Conv 3×3	LReLU	$128 \times 256 \times 256$	74k
Downsample	–	$128 \times 128 \times 128$	–
Conv 3×3	LReLU	$128 \times 128 \times 128$	148k
Conv 3×3	LReLU	$256 \times 128 \times 128$	295k
Downsample	–	$256 \times 64 \times 64$	–
Conv 3×3	LReLU	$256 \times 64 \times 64$	590k
Conv 3×3	LReLU	$512 \times 64 \times 64$	1.2M
Downsample	–	$512 \times 32 \times 32$	–
Conv 3×3	LReLU	$512 \times 32 \times 32$	2.4M
Conv 3×3	LReLU	$512 \times 32 \times 32$	2.4M
Downsample	–	$512 \times 16 \times 16$	–
Conv 3×3	LReLU	$512 \times 16 \times 16$	2.4M
Conv 3×3	LReLU	$512 \times 16 \times 16$	2.4M
Downsample	–	$512 \times 8 \times 8$	–
Conv 3×3	LReLU	$512 \times 8 \times 8$	2.4M
Conv 3×3	LReLU	$512 \times 8 \times 8$	2.4M
Downsample	–	$512 \times 4 \times 4$	–
Minibatch stddev	–	$513 \times 4 \times 4$	–
Conv 3×3	LReLU	$512 \times 4 \times 4$	2.4M
Conv 4×4	LReLU	$512 \times 1 \times 1$	4.2M
Fully-connected	linear	$1 \times 1 \times 1$	513
Total trainable parameters			23.1M

Table 2: Generator and discriminator that we use with CELEBA-HQ to generate 1024×1024 images.

Pixel Replication



Progressive GAN - Results

- CIFAR10 Inception Score: 8.80
- High quality images from different LSUN categories
- Able to generate 1024x1024 images from CelebA-HQ



Figure 7: Selection of 256×256 images generated from different LSUN categories.



Figure 5: 1024×1024 images generated using the CELEBA-HQ dataset. See Appendix F for a larger set of results, and the accompanying video for latent space interpolations.

References

- [GAN](#) (Goodfellow)
- Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks [Link](#)
- Conditional Generative Adversarial Nets [Link](#)
- Generative Adversarial Text to Image Synthesis [Link](#)
- Learning to Discover Cross-Domain Relations with Generative Adversarial Networks (DiscoGAN) [Link](#)
- StarGAN [Link](#)
- Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks [Link](#)
- Generative Face Completion [Link](#)
- 3D Shape Induction from 2D Views of Multiple Objects [Link](#)
- Low dimensional projections of data:
 - Matheus Gadelha, Subhransu Maji, and Rui Wang. 3d shape induction from 2d views of multiple objects. 2016. [Link](#)
 - Behnam Neyshabur, Srinadh Bhojanapalli, and Ayan Chakrabarti. Stabilizing gan training with multiple random projections. arXiv preprint arXiv:1705.07831, 2017. [Link](#)
- Maya Kabkab, Pouya Samangouei, Rama Chellappa. Task-Aware Compressed Sensing with Generative Adversarial Networks. [Link](#)
- Zhiming Zhou, Han Cai, Shu Rong, Yuxuan Song, Kan Ren, Weinan Zhang, Jun Wang, and Yong Yu. Activation Maximization Generative Adversarial Networks. [Link](#)