Generative Adversial Networks

David Coufal
Institute of Computer Science
The Czech Academy of Sciences
david.coufal@cs.cas.cz

Vision for Robotics - FEL CTU
December 9, 2019

Neural networks

- a neural network is a complex composite function built from individual layers of neurons, neurons represent simple computation units

- neurons are parametrized, so the whole network is a highly parametrized function

- adjustment of parameters is called network learning back propagation of an error represented by some loss funtion

- shallow networks - only one hidden layer of neurons

- deep networks - multiple layers (up to 200 layers, millions of parameters)

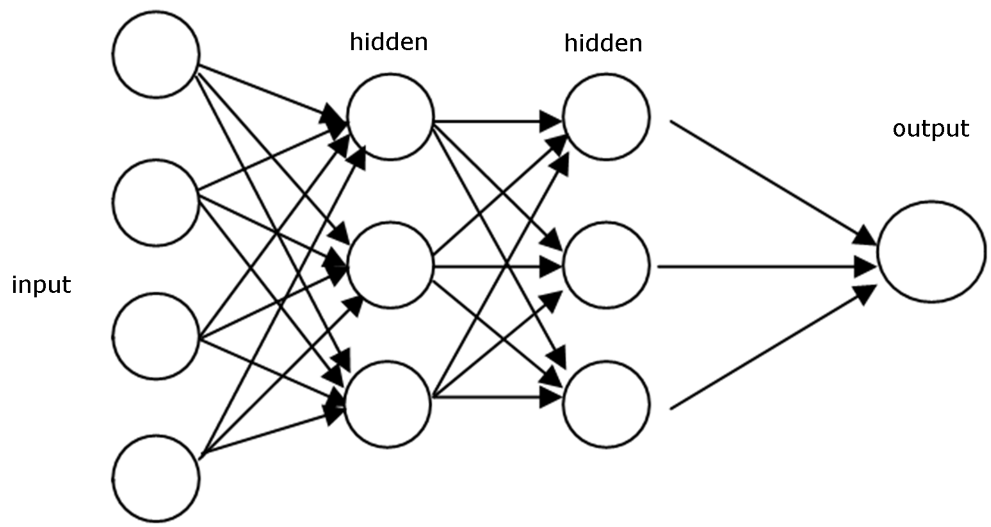
Standard neural networks

- standard neuron $h : \mathbb{R}^d \to \mathbb{R}$ has form

$$h(\boldsymbol{x}) = act(\boldsymbol{w}\boldsymbol{x} + \boldsymbol{b})$$

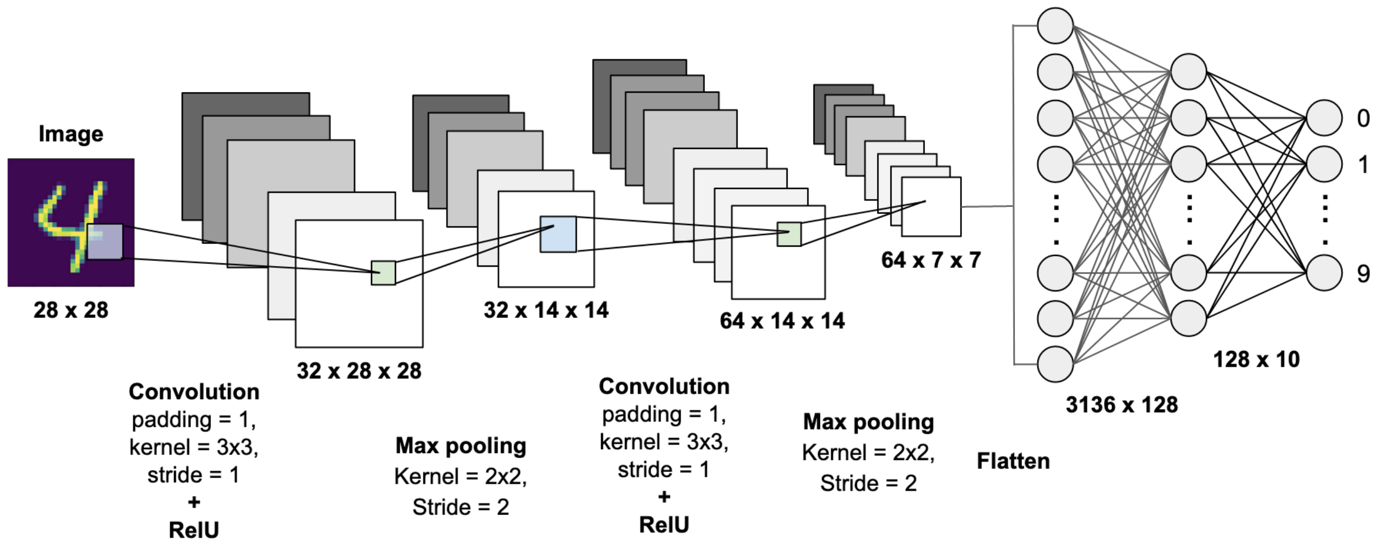
  - $act(z) = \max(0, z)$ (relu), $act(z) = \frac{1}{1+e^{-\beta z}}$ (sigmoid)

- $\boldsymbol{w}, \boldsymbol{b} \in \mathbb{R}^d$ - parameters



Convolutional neural networks

- convolution filters moving over the input



source: https://towardsdatascience.com/mnist-handwritten-digits-classification-using-a-convolutional-neural-network-cnn-af5fafbc35e9

- down-sampling and up-samplig operations, pooling

Well recognized DL tasks

- classification
  ImageNet Large Scale Visual Recognition Challenge AlexNet CNN network won the contest using convolutional implementation (2012)

- reccurent neural networks (RNNs)
  LSTM, GRU - units, NLP tasks, Google Translator

- reinforcement learning DeepMind (UK, Google 2014)
  AlhaGo vs. Lee Sedol (4:1, 2016), AlphaGoZero vs. AlphaGo (100:0, 2017) AlphaZero vs. Stockfish (28:72:0, 2018), Dota 2 tournaments ...

- generative programming
  Ian Godfellow et al. (2014) - *Generative Adversial Networks*
  https://arxiv.org/abs/1406.2661

Elementary concepts

- random variable $X \sim P_X$, $(\Omega, \mathcal{A}, P_X)$
  - Ω - space of elementary events $X \in \Omega$
  - $\mathcal{A}$ - sigma algebra of measurable events
  - P_X - distribution of X

- distribution of X
  - set function on $\mathcal{A}$, $P_X : \mathcal{A} \to [0, 1]$
  - obeys Kolmogorov's laws of probability
  - typically $\Omega \in \mathbb{R}^d$ and $\mathcal{A} = \mathcal{B}(\mathbb{R}^d)$

- data $D = \{\boldsymbol{x}_i \in \mathbb{R}^d\}_{i=1}^n$ comes from distribution P_D
  i.e., we assume that there exists a random variable D such that $D \sim P_D$ (sometimes we use P_{data} instead of P_D)

- How to specify P_D on the basis of D?

Elementary concepts

- if Ω is countable, P_D can be given by enumeration, i.e., $P_D(\omega_i) = p_i$, for $i = 1, \ldots, n$ (finite) or $i \in \mathbb{N}$ (countable)

- if $\Omega = \mathbb{R}^d$, specification of cdf is possible, but inconvenient in higher dimensions, so the most common approach is to specify a density $p_D : \mathbb{R}^d \to [0, \infty)$ of P_D and one has

$$P_D(A) = \int_A p_D(\boldsymbol{x})\, d\boldsymbol{x} \quad \text{for } A \in \mathcal{B}(\mathbb{R}^d)$$

- cannot handle distributions which do not have densities, complex formulas in high dimensions for dependent data

- How to get the density from empirical data?

Elementary concepts

- if $p_D \in \{p_\theta, \theta \in \Theta\}$ (a parametric set of densities)
  task reduces to estimate θ^* from data D and $p_D = p_{\theta^*}$
  maximum likelihood estimation

- in a non-parametric context, kernel density estimation
  is the standard choice

$$p_D^*(x) = \frac{1}{nh^d} \sum_{k=1}^{n} K\left(\frac{\boldsymbol{x} - \boldsymbol{x}_i}{h}\right)$$

- $K : \mathbb{R}^d \to \mathbb{R}$, a kernel (bump) function, $h > 0$ is the bandwidth
  practically applicable for d up to 5

- How to sample from a given distribution/density?

Distance of probability distributions

- space of probability distributions on $\mathbb{R}^d, \mathcal{B}(\mathbb{R}^d)$:
  $\mathcal{P} = \{P$: probability distribution on $(\mathbb{R}^d, \mathcal{B}(\mathbb{R}^d))\}$
  $\mathcal{P}$ is metrizable, e.g., using Lévy-Prokhorov metric
  $\pi : \mathcal{P}^2 \to [0, \infty)$, complicated formulas

- another "metric" is the Kullback-Leibler divergence
  let $P, Q \in \mathcal{P}$, $P \ll Q$ (if $Q(x) = 0$, then $P(x) = 0$)

$$
\begin{aligned}
KL(P||Q) &= \int \frac{\mathrm{d}P}{\mathrm{d}Q}\, dP \\
&= \int \log\left(\frac{p(\boldsymbol{x})}{q(\boldsymbol{x})}\right) p(\boldsymbol{x})\, d\boldsymbol{x}
\end{aligned}
$$

- properties:
  $KL(P||Q) \neq KL(Q||P)$, $KL(P||Q) \geq 0$, $KL(P||P) = 0$,

- tight relation to theory of information (relative entropy), theory of large deviations

Kullback-Leibler divergence

- (Wikipedia entry ...) In applications, P typically represents the "true" distribution of data, observations, or a precisely calculated theoretical distribution, while Q typically represents a theory, model, description, or approximation of P. In order to find a distribution Q that is closest to P, we can minimize KL divergence and compute (reverse) information projection

- Kullback-Leibler divergence is a special case of a broader class of statistical divergences called f-divergences

- Jensen-Shannon divergence - symmetrized KL divergence

$$JS(P||Q) = \frac{1}{2}KL(P||M) + \frac{1}{2}KL(Q||M)$$

where $M = \frac{1}{2}(P+Q)$

Reverse information projection (M-projection)

- let $P \in \mathcal{P}$ and $\mathcal{Q} \subset \mathcal{P}$ (subset of prob. distributions)

$$Q^*_{KL} = \arg\min_{Q \in \mathcal{Q}} KL(P||Q)$$

  or for JS

$$Q^*_{JSD} = \arg\min_{Q \in \mathcal{Q}} JSD(P||Q)$$

  Q^* is the closest distribution from subset of $\mathcal{Q}$ to P

- easy to state, generally hard to solve (i.e., to find Q^*)

Specification of $\mathcal{Q} \subset \mathcal{P}$

- via parametrized densities $\mathcal{Q} = \{p_\theta, \theta \in \Theta\}$

- via parametrized transformations
  e.g., let $X \sim N(0,1)$ then $X^2 \sim \chi^2(1)$
  X has some simple distribution which is easy to sample from and is transformed to a complex one using a deterministic function G
  (above $G(z) = z^2$)

- Q is given by set of parametrized functions G_θ, $\theta \in \Theta$
  (neural networks parametrized via their weights)

- easy sampling from $G_\theta(X)$, sample $\boldsymbol{x} \sim X$ (easy)
  and then pass $\boldsymbol{x}$ through $G_\theta(X)$, i.e., compute $G_\theta(\boldsymbol{x})$

- How to solve the information projection problem?

Maximum likelihood estimation

- task
  given set of data $\{\boldsymbol{x}_i \sim P_D\}_{i=1}^n$ describe distribution P_D

- MLE estimate $P_D \in P_\theta = \{P_\theta, \theta \in \Theta\}$
  assume that P_θ has density, i.e., $dP_\theta = p_\theta(\boldsymbol{x})\, d\boldsymbol{x}$
  assume that $\boldsymbol{x}_i$ i.i.d.
  search for optimal $\theta_{\text{mle}} \in \Theta$ and set $P_D = P_{\theta_{\text{mle}}}$

$$
\begin{aligned}
\theta_{\text{mle}} &= \text{argmax}_\theta\, \mathbb{E}_{\boldsymbol{x}\sim P_D} \log p_\theta(\boldsymbol{x}) \\
\text{estimate } \theta^*_{\text{mle}} &= \text{argmax}_\theta\, \frac{1}{n} \sum_{i=1}^{n} \log p_\theta(\boldsymbol{x}_i)
\end{aligned}
$$

- optimization in terms of KL-divergence

$$
\begin{aligned}
\theta_{\text{mle}} &= \text{argmin}_\theta\, KL(P_D(\boldsymbol{x})||P_\theta(\boldsymbol{x})) \\
&= \text{argmin}_\theta \int p_D(\boldsymbol{x}) \frac{p_D(\boldsymbol{x})}{p_\theta(\boldsymbol{x})}\, d\boldsymbol{x}
\end{aligned}
$$

MLE in terms of KL-divergence

- best approximation of P_D using P_θ
  - $\widehat{P}_D$ proxy for P_D, $\widehat{P}_D(d\boldsymbol{x}) = \frac{1}{n}\delta_{\boldsymbol{x}_i}(d\boldsymbol{x})$ (Dirac m.)
  - P_θ - model distribution with density $p_{\text{model}}(\boldsymbol{x}|\theta)$
- maximization MLE = minimization of $KL(P_D||P_\theta)$

$$
\begin{aligned}
KL(P_D||P_\theta) &= \int \log \frac{dP_D}{dP_\theta}\, dP_D = \int \log \frac{p_D(\boldsymbol{x})}{p_\theta(\boldsymbol{x})}\, dP_D \\
&= \int \log p_D(\boldsymbol{x})\, dP_D - \int \log p_\theta(\boldsymbol{x})\, dP_D \\
&\approx -H[P_D] - \int p_\theta(\boldsymbol{x})\, d\widehat{P}_D \quad (P_D \approx \widehat{P}_D) \\
&\propto -\int \log p_\theta(\boldsymbol{x})\, d\widehat{P}_D \quad \text{(integration over Dirac)} \\
&\propto \underbrace{-\frac{1}{n}\sum_{i=1}^{n} \log p_\theta(\boldsymbol{x}_i)}_{=\text{MLE}}
\end{aligned}
$$

Generative modeling

- purpose

  given data from an uknown distribution $\boldsymbol{x} \sim p(\boldsymbol{x})$
  model $p(\boldsymbol{x})$ using a differentaible mapping G so that

$$p(\boldsymbol{x}) \sim G_{\theta_g}(p(\boldsymbol{z})) = G(p(\boldsymbol{z}); \theta_g))$$

  where $p(\boldsymbol{z})$ is a selected, simple prior, e.g. mv Gaussian

- maximum likelihood estimation direct setting of density under i.i.d. assumption, KL divergence minimization

Generative modeling

- solution to the information projection problem
  KL-divergence minimalization
  via playing discriminator, generator adversial game

source: https://towardsdatascience.com/generative-adversarial-networks-learning-to-create-8b15709587c9

Partial criterions

- an ideal discriminator
  $D : \boldsymbol{x} \in \mathbb{R}^d \to (0, 1)$, i.e., $\log D : \boldsymbol{x} \to (-\infty, 0)$
  we would like $D_{\theta_d}(\boldsymbol{x}^{real}) \to 1$, $D_{\theta_d}(\boldsymbol{x}^{fake}) \to 0$
  i.e., maximize w.r.t. θ_d

$$\log(D_{\theta_d}(\boldsymbol{x}^{real})) + \log((1 - D_{\theta_d}(\boldsymbol{x}^{fake})))$$

- an ideal generator
  generator wants to fool discriminator,
  i.e., it generates $\boldsymbol{x}^{fake}$ so that $D_{\theta_d}(\boldsymbol{x}^{fake}) \to 1$
  tune weights of the generator to minimize

$$\log((1 - D_{\theta_d}(\boldsymbol{x}^{fake}))) = \log((1 - D_{\theta_d}(D(G_{\theta_g}(\boldsymbol{z})))$$

w.r.t θ_g for θ_d fixed

Compound criterion

- compound criterion

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

- minimax optimization - set θ_d, θ_g using

$$\min_{\theta_g} \max_{\theta_d} V(D_{\theta_d}, G_{\theta_g})$$

- alternate optimization
  - for fixed generator G_{θ_g} maximize $V(D_{\theta_d}, \cdot)$
  - for fixed discriminator D_{θ_d} minimize $V(\cdot, G_{\theta_g})$

Theoretical analysis

- **Proposition 1.** For any G fixed,
  the optimal discriminator D^*_G computes the function

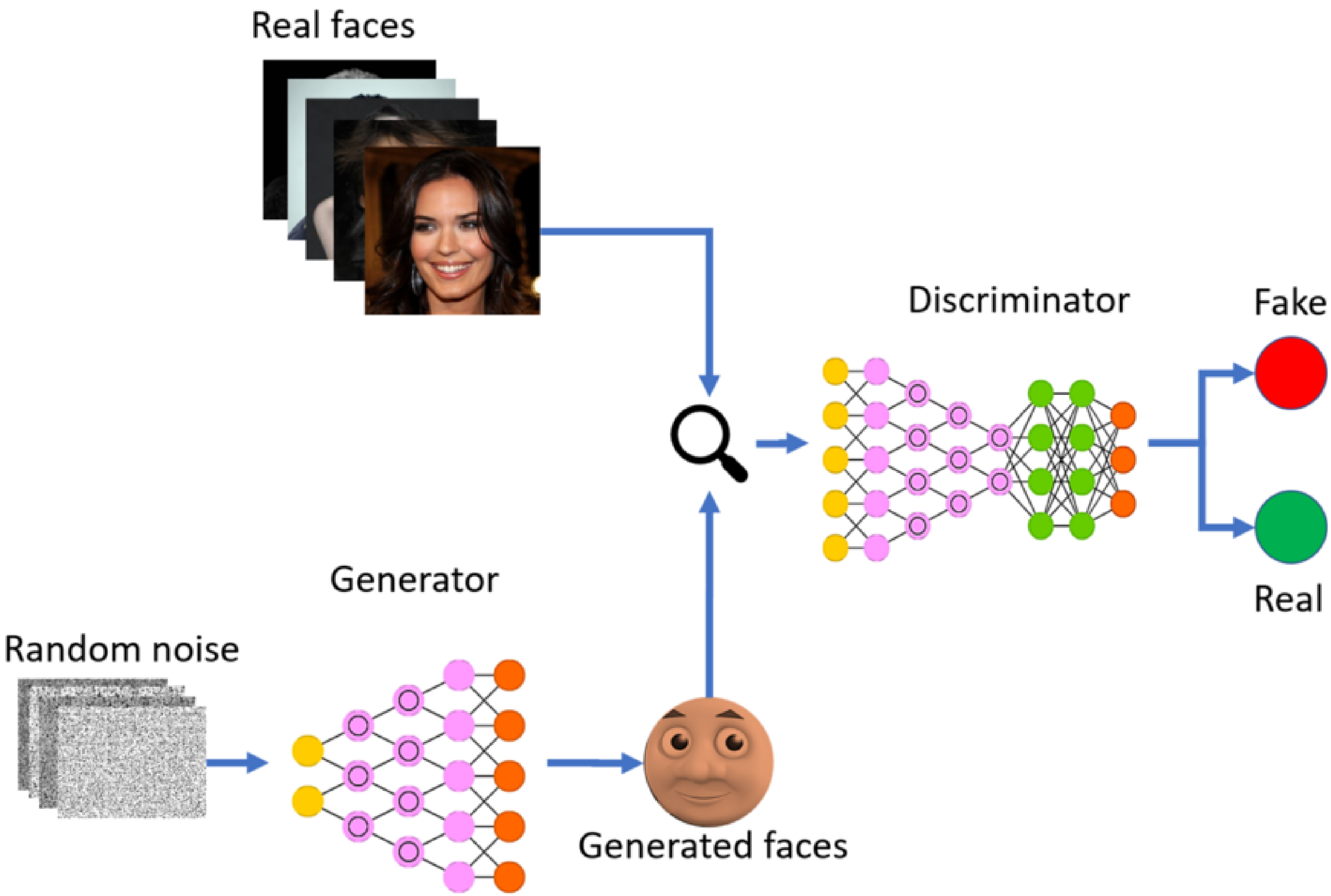
$$D^*_G = \frac{p_{\text{data}}(\boldsymbol{x})}{p_{\text{data}}(\boldsymbol{x}) + p_g(\boldsymbol{x})}$$

- **Proposition 2.** Let $C(G) = V(D^*_G, G)$, then global minimum of $\min_G C(G)$ is achieved if and only if $p_g = p_{\text{data}}$.
  At that point $C(G)$ achieves value $-\log 4$

- **Proposition 3.** Optimizing $\min_G \max_D V(D, G)$ corresponds to minimizing $JS(p_{\text{data}}||p_g)$, which is minimal (=0) if and only if $p_{\text{data}} = p_g$

source: https://arxiv.org/abs/1406.2661

A GAN concept



source: https://medium.com/sigmoid/a-brief-introduction-to-gans

Learning algorithm

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

for number of training iterations **do**

  for k steps **do**

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

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

  end for

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

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

end for

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

source: https://arxiv.org/abs/1406.2661

MNIST dataset

- 60000/10000 - 28x28 greyscale images of handwritten digits
  http://yann.lecun.com/exdb/mnist/

MNIST dataset

- 60000/10000 - 28x28 greyscale images of handwritten digits
  GAN architecture: D,G - perceptron networks

MNIST dataset

- 60000/10000 - 28x28 greyscale images of handwritten digits

  GAN architecture: D,G - convolution networks

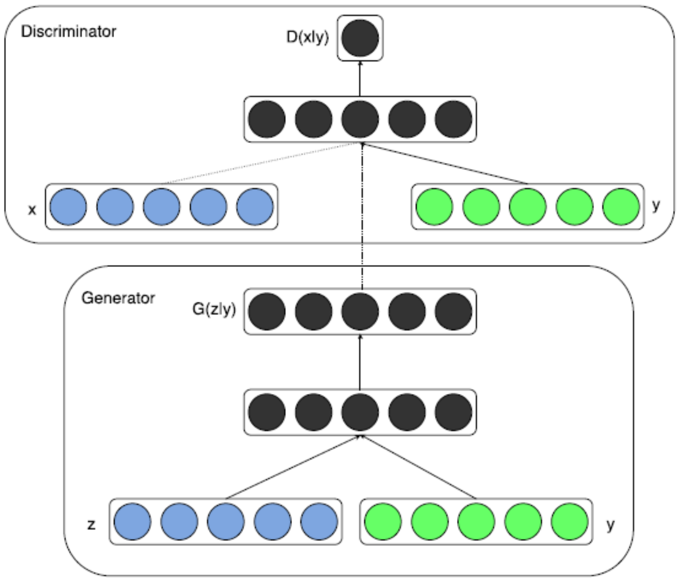
cGAN - 2014

- *Conditional Generative Adversarial Nets* https://arxiv.org/abs/1411.1784

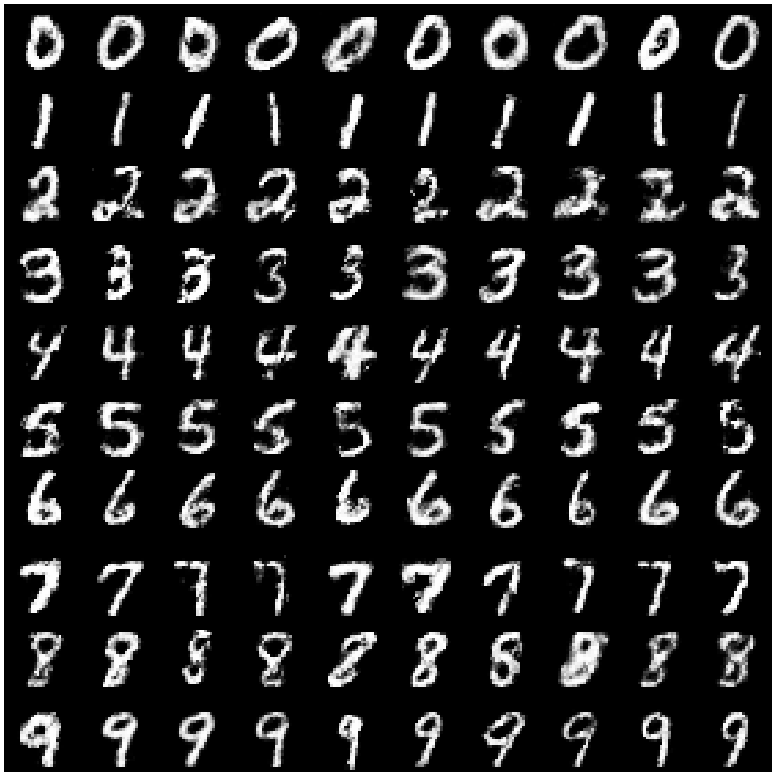
- unconditional vs. conditional GAN, $\boldsymbol{y} - condition$

$$\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log\ D(\boldsymbol{x})] \ + \ \mathbb{E}_{\boldsymbol{x}\sim p_{\boldsymbol{z}}(\boldsymbol{x})}[\log(1 - D(G(z))]$$
$$\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log\ D(\boldsymbol{x}|\boldsymbol{y})] \ + \ \mathbb{E}_{\boldsymbol{x}\sim p_{\boldsymbol{z}}(\boldsymbol{x})}[\log(1 - D(G(z|\boldsymbol{y}))]$$

- conditioning by extending latent variable of generator



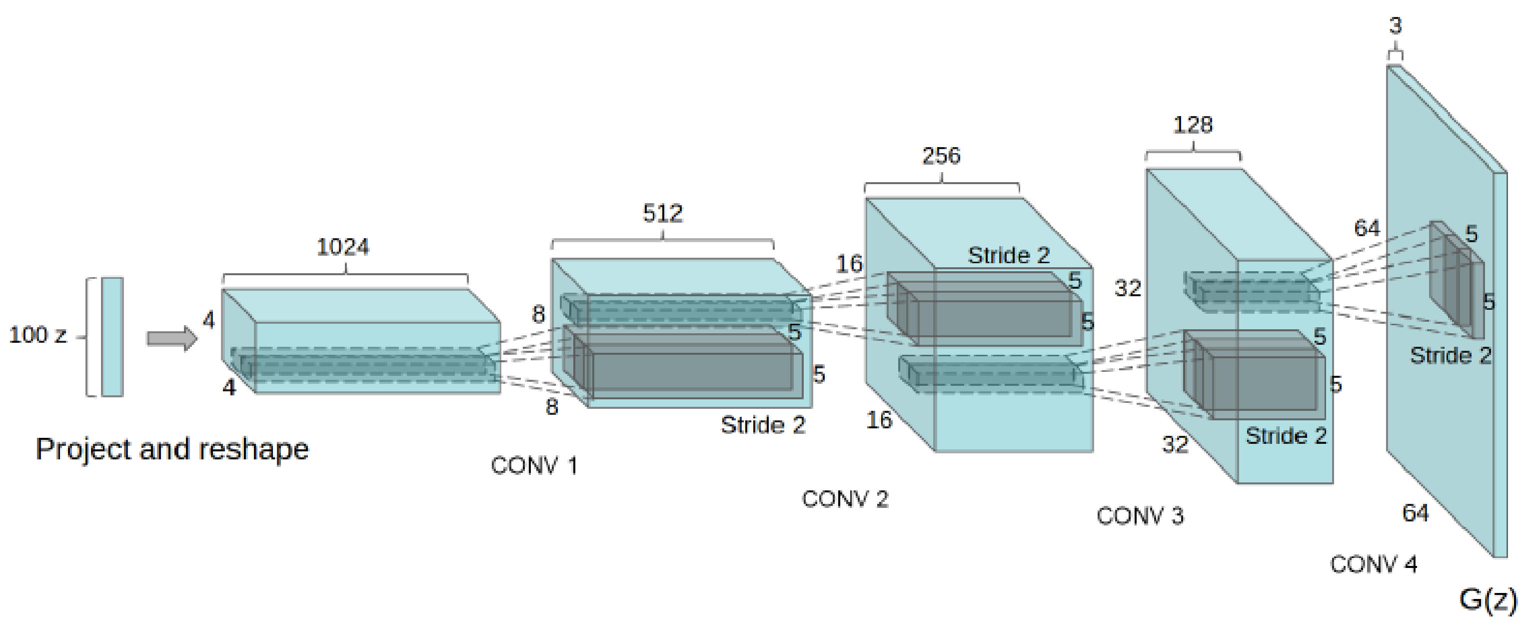
MNIST dataset



DCGAN - 2015

- *Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks* https://arxiv.org/abs/1511.06434

- architecture - uses convolutional layers



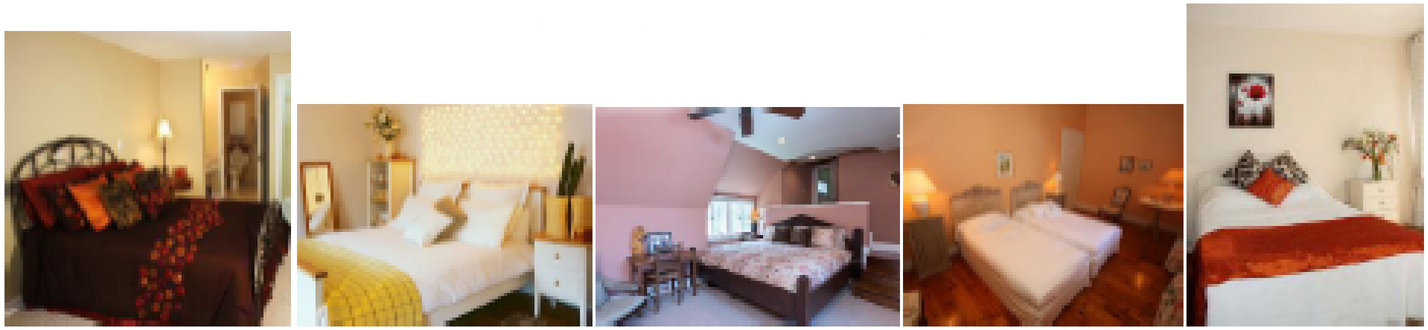
LSUN dataset

- 10 - categories, (church_outdoor, bedroom, bridge ...) https://www.yf.io/p/lsun

DCGAN - 2015

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 - 2015



LSUN/bedroom

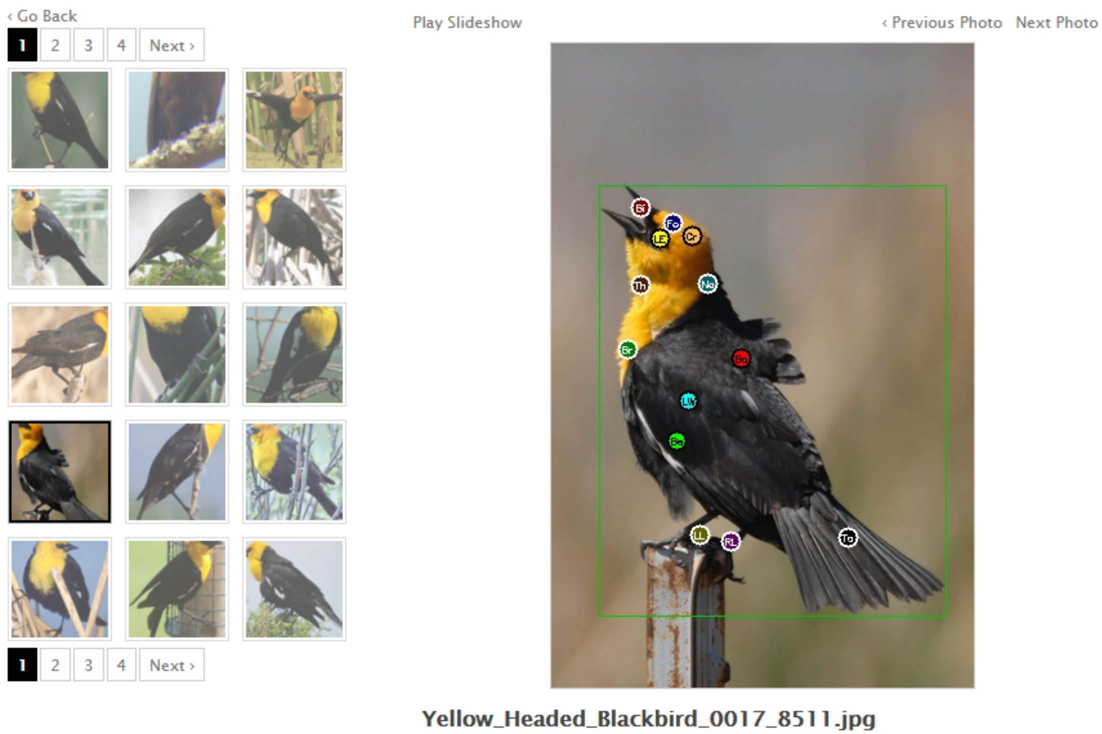
StackGAN - 2016

- *StackGAN: Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks* https://arxiv.org/abs/1612.03242

- Caltech-UCSD Birds 200 Dataset
  http://www.vision.caltech.edu/visipedia/CUB-200-2011.html

- 102 Category Flower Dataset
  https://www.robots.ox.ac.uk/ vgg/data/flowers/102/

StackGAN - 2016



- *a bird has a bright golden crown and throat, it's breast is yellow, and back is black*
- *upper body yellow and lower black with black color around beak*
- *this bird has a bright yellow crown, a long straight bill, and white wingbars*
- *this is a black bird with a yellow head and breast ...*

StackGAN - 2016

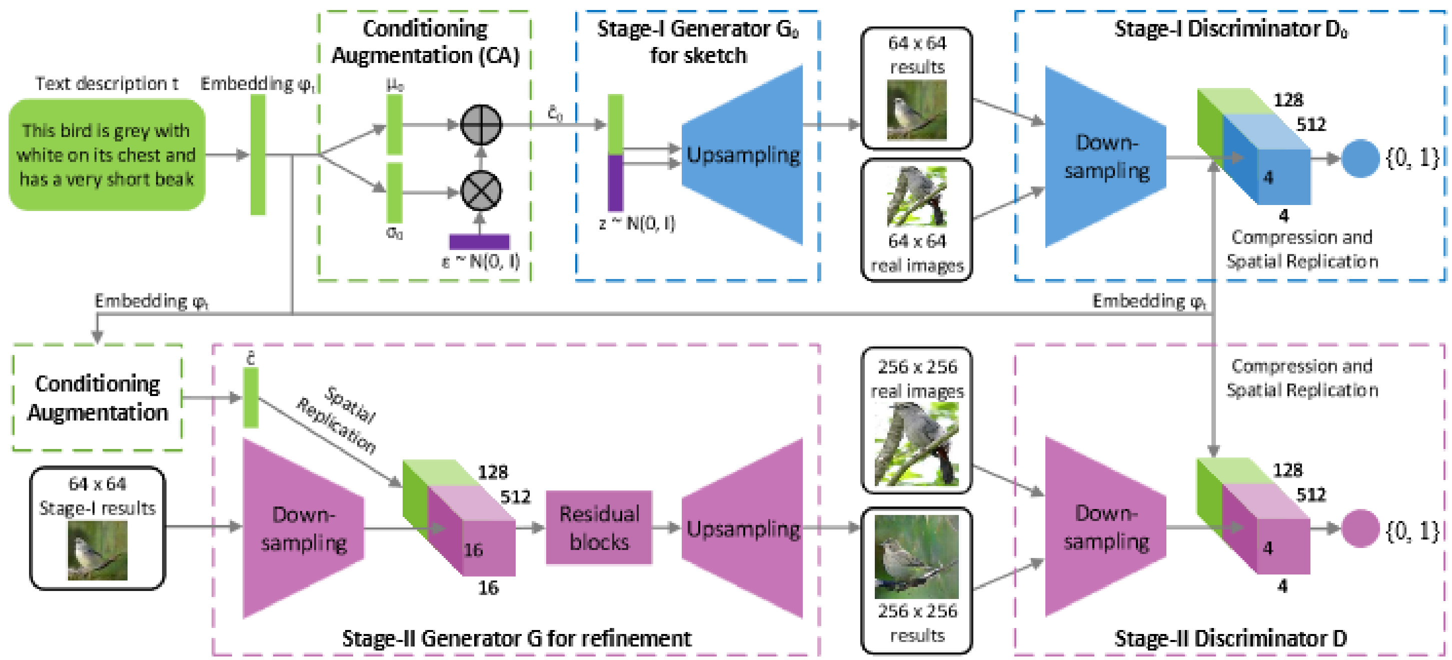


Figure 2. The architecture of the proposed StackGAN. The Stage-I generator draws a low-resolution image by sketching rough shape and basic colors of the object from the given text and painting the background from a random noise vector. Conditioned on Stage-I results, the Stage-II generator corrects defects and adds compelling details into Stage-I results, yielding a more realistic high-resolution image.

StackGAN - 2016

Figure 3. Example results by our StackGAN conditioned on text descriptions from CUB test set.

Figure 4. Example results by our StackGAN conditioned on text descriptions from Oxford-102 test set and COCO validation set

StackGAN - 2016

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.

- https://github.com/hanzhanggit/StackGAN

BEGAN - 2017

- *BEGAN: Boundary Equilibrium Generative Adversarial Networks*
  https://arxiv.org/abs/1703.10717

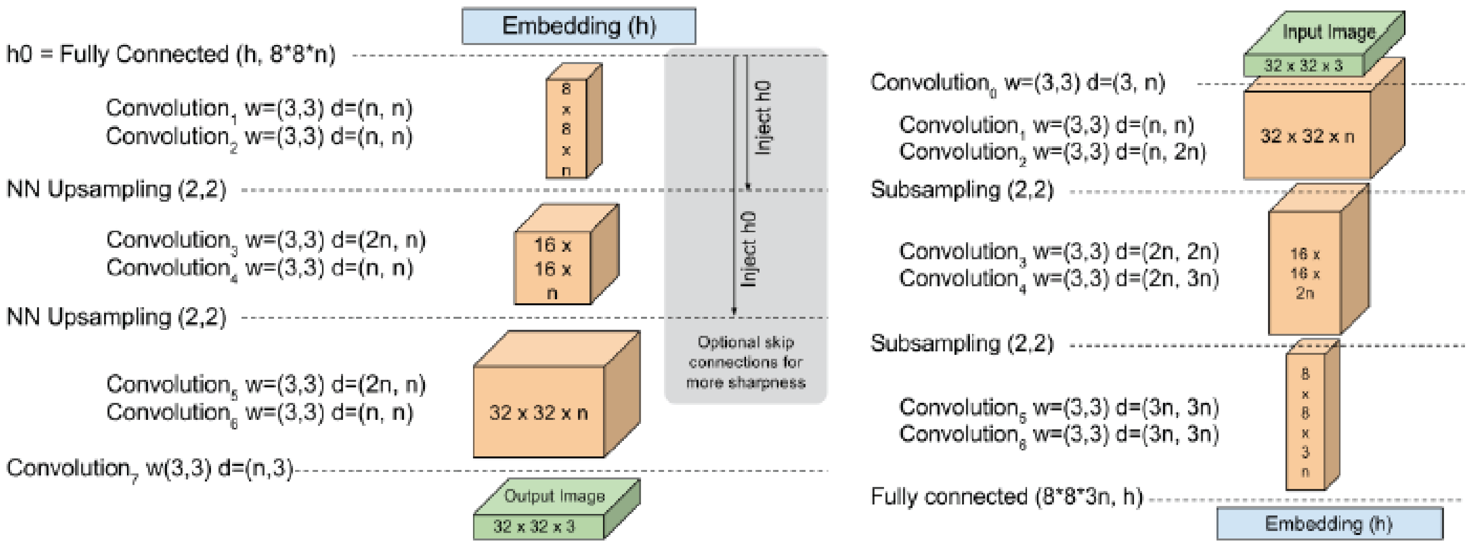
- energy based GAN, discriminator assigns low energy values to real data and high otherwise, generator produces samples assigned with low energy by discriminator - generalized view of loss functions
  training minimization of loss

$$V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[D_{\theta_d}(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{x}\sim p_{\boldsymbol{z}}(\boldsymbol{x})}[(m - D_{\theta_d}(G_{\theta_g}(z)))_+]$$

where m is a positive margin and $0 \le D_{\theta_d} \le m$

BEGAN - 2017

- architecture - uses convolutional layers



(a) Generator/Decoder (b) Encoder

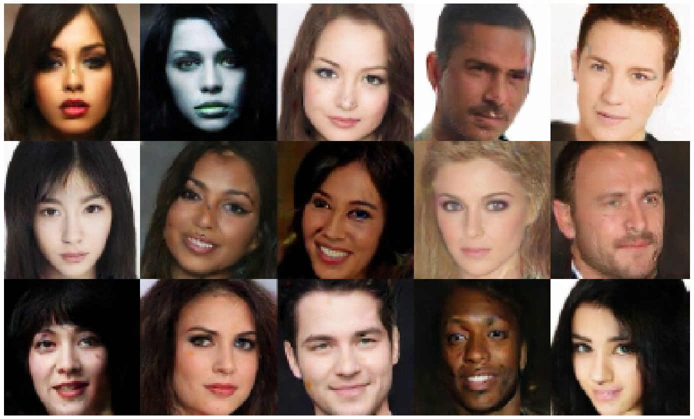
Figure 1: Network architecture for the generator and discriminator.

BEGAN - 2017

- celebA dataset – http://mmlab.ie.cuhk.edu.hk/projects/CelebA.html

BEGAN - 2017

- generated fake images



(b) Our results (128x128)

Figure 3: Random 64x64 samples at varying $\gamma \in \{0.3, 0.5, 0.7\}$

PGGAN - 2018

- *Progressive Growing of GANs for Improved Quality, Stability, and Variation* https://arxiv.org/abs/1710.10196

- architecture - uses convolutional layers

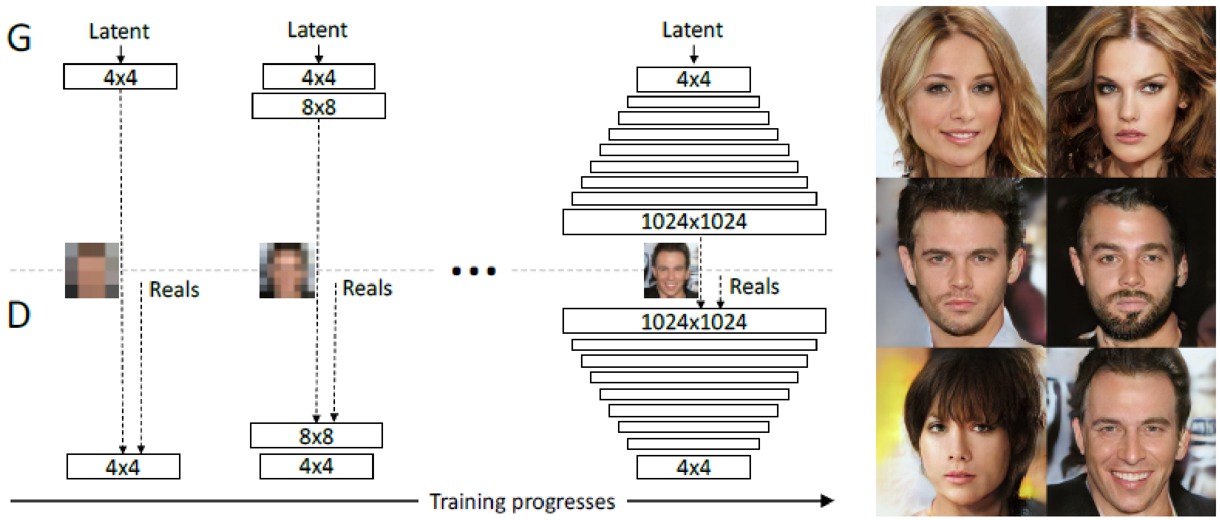


Figure 1: Our training starts with both the generator (G) and discriminator (D) having a low spatial resolution of 4×4 pixels. As the training advances, we incrementally add layers to G and D, thus increasing the spatial resolution of the generated images. All existing layers remain trainable throughout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

PGGAN - 2018

- architecture - uses convolutional layers

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.

PGGAN - 2018

- architecture - uses convolutional layers

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.

ImageNet

- over 14 mil. of images from 20 thousand categories based on the WordNet database (a dictionary)

BigGAN - 2019

- *Large Scale GAN Training for High Fidelity Natural Image Synthesis* https://arxiv.org/abs/1809.11096

- we show that GANs benefit dramatically from scaling, and train models with two to four times as many parameters and eight times the batch size compared to prior art

- training on 128 to 512 cores of a Google TPUv3 Pod

Batch	Ch.	Param (M)	Shared	Skip-z	Ortho.	Itr $\times 10^3$	FID	IS
256	64	81.5	SA-GAN Baseline			1000	18.65	52.52
512	64	81.5	✗	✗	✗	1000	15.30	58.77($\pm$1.18)
1024	64	81.5	✗	✗	✗	1000	14.88	63.03($\pm$1.42)
2048	64	81.5	✗	✗	✗	732	12.39	76.85($\pm$3.83)
2048	96	173.5	✗	✗	✗	295($\pm$18)	9.54($\pm$0.62)	92.98($\pm$4.27)
2048	96	160.6	✓	✗	✗	185($\pm$11)	9.18($\pm$0.13)	94.94($\pm$1.32)
2048	96	158.3	✓	✓	✗	152($\pm$7)	8.73($\pm$0.45)	98.76($\pm$2.84)
2048	96	158.3	✓	✓	✓	165($\pm$13)	8.51($\pm$0.32)	99.31($\pm$2.10)
2048	64	71.3	✓	✓	✓	371($\pm$7)	10.48($\pm$0.10)	86.90($\pm$0.61)

Table 1: Fréchet Inception Distance (FID, lower is better) and Inception Score (IS, higher is better) for ablations of our proposed modifications. *Batch* is batch size, *Param* is total number of parameters, *Ch.* is the channel multiplier representing the number of units in each layer, *Shared* is using shared embeddings, *Skip-z* is using skip connections from the latent to multiple layers, *Ortho.* is Orthogonal Regularization, and *Itr* indicates if the setting is stable to 10^6 iterations, or it collapses at the given iteration. Other than rows 1-4, results are computed across 8 random initializations.

BigGAN - 2019

- architecture - uses convolutional layers

Figure 1: Class-conditional samples generated by our model.

Open questions

- What sorts of distributions can GANs model?

- How can we scale GANs beyond image synthesis?
  (text, audio, computer-aided drug design - https://insilico.com)

- What can we say about the global convergence of the training dynamics?

- How does GAN training scale with batch size?

- What is the relationship between GANs and adversarial examples?

source: https://distill.pub/2019/gan-open-problems