Generative Adversial Networks

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

Vision for Robotics - FEL CTU
November 28, 2022

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
  via back propagation of a loss function error
  (internally computes a gradient of error w.r.t net parameters)

- shallow networks - one hidden layer of neurons

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

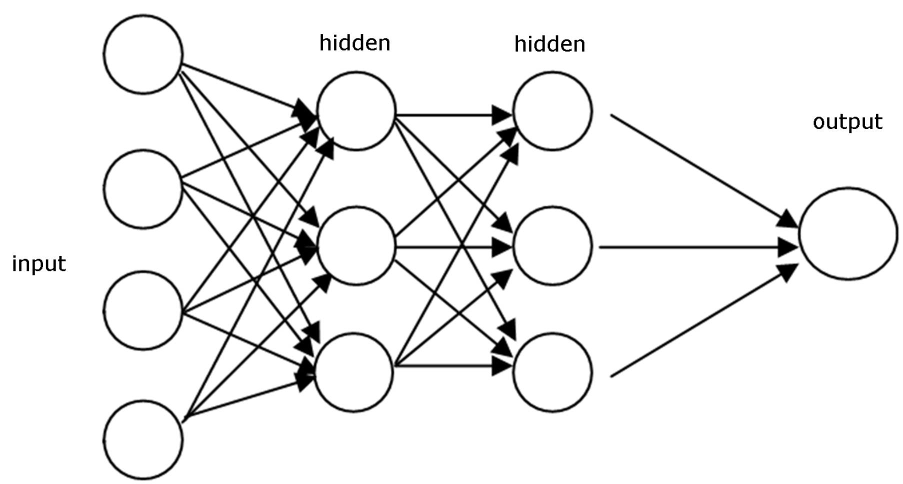
Perceptron neural networks

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

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

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

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



Neural networks for classification

- K classes - $c_1, \ldots, c_K$, K neurons in the output layer

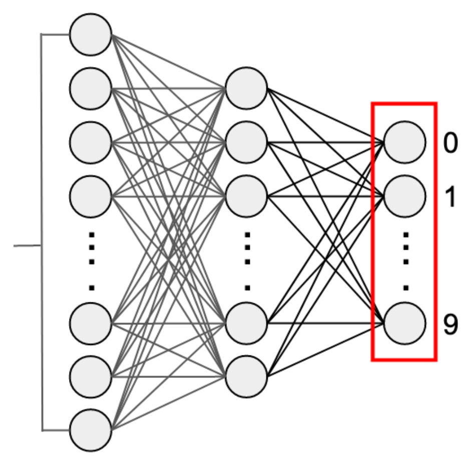
$$q = (o_1(\boldsymbol{x}), \ldots, o_K(\boldsymbol{x}))$$

- normalization using softmax function

$$o_k(\boldsymbol{x}) = \frac{e^{x_k}}{\sum_{k=1}^{K} e^{x_k}}, \quad o_k \in (0,1), \quad \sum_k o_k(\boldsymbol{x}) = 1$$

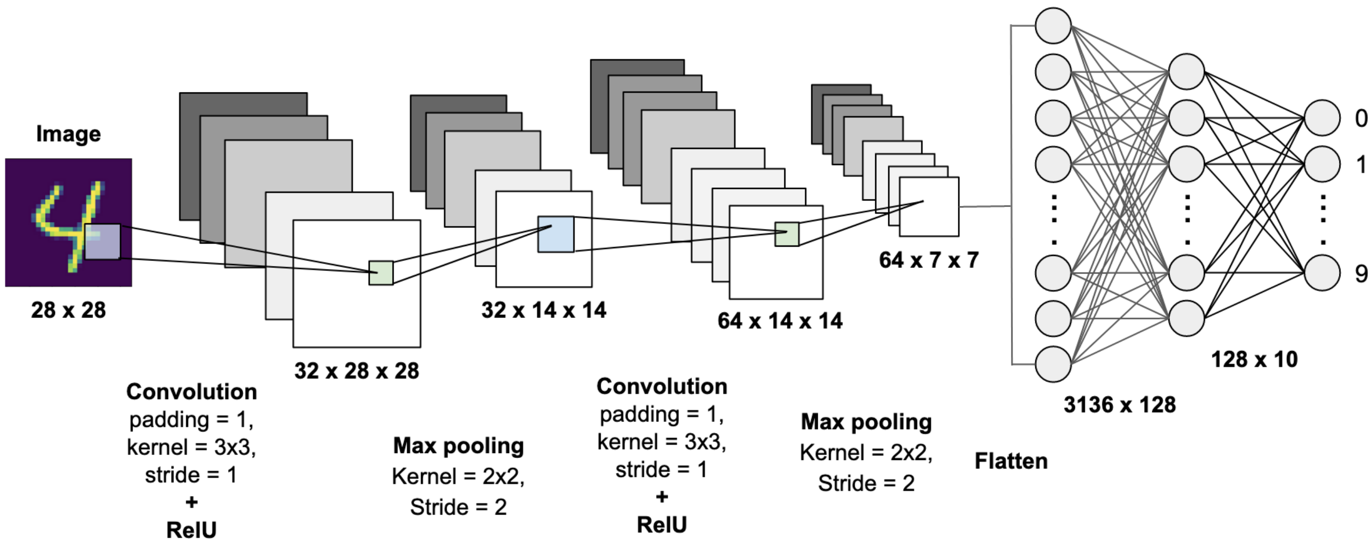
- $q \in \mathcal{P}(\mathbb{N})$ – probability distribution on $\mathbb{N}$
  $q_k = o_k$ for $k \leq K, q_k = 0,$ otherwise

- $K = 10$



Convolutional neural networks

- convolution filters moving over the input

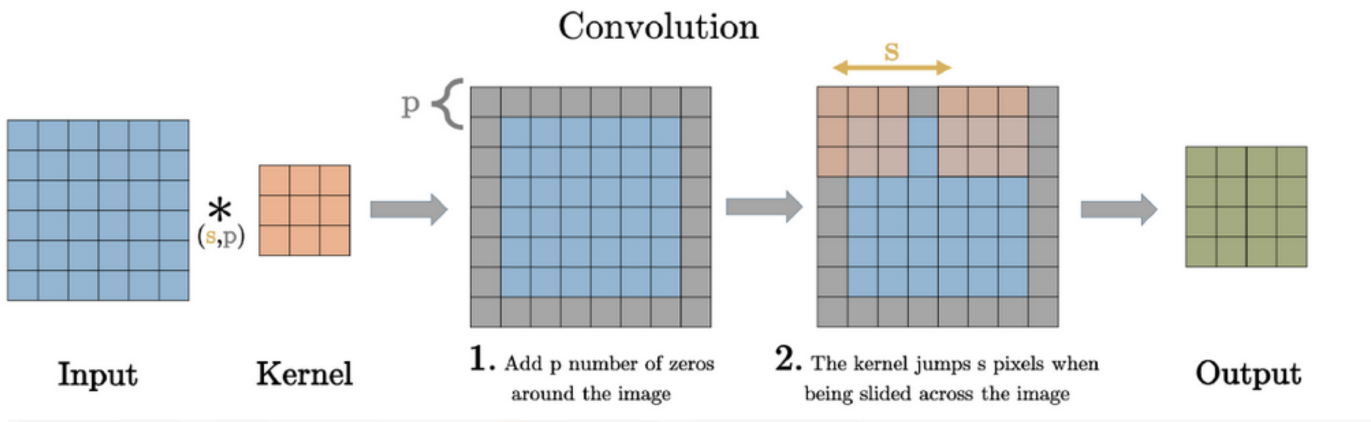


source: https://towardsdatascience.com/mnist-handwritten-digits-classification-using-a-convolutional-neural-network-cnn-af5fafbc35e9

- down-sampling and up-sampling operations

Standard convolutions

- convolution filters moving over the input ($i \times i$)



source: https://towardsdatascience.com/what-is-transposed-convolutional-layer-40e5e6e31c11

- parameters - filter size ($k \times k$), padding (=0), strides (=1)

- convolution animations

Convolutions as matrix operations

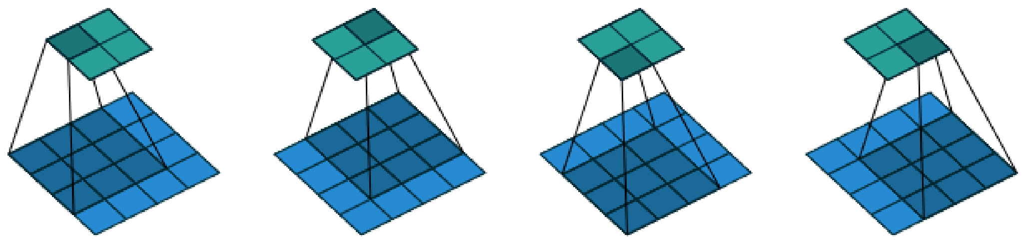


Figure 2.1: (No padding, unit strides) Convolving a 3×3 kernel over a 4×4 input using unit strides (i.e., $i = 4$, $k = 3$, $s = 1$ and $p = 0$).

Take for example the convolution represented in Figure 2.1. If the input and output were to be unrolled into vectors from left to right, top to bottom, the convolution could be represented as a sparse matrix $\mathbf{C}$ where the non-zero elements are the elements $w_{i,j}$ of the kernel (with i and j being the row and column of the kernel respectively):

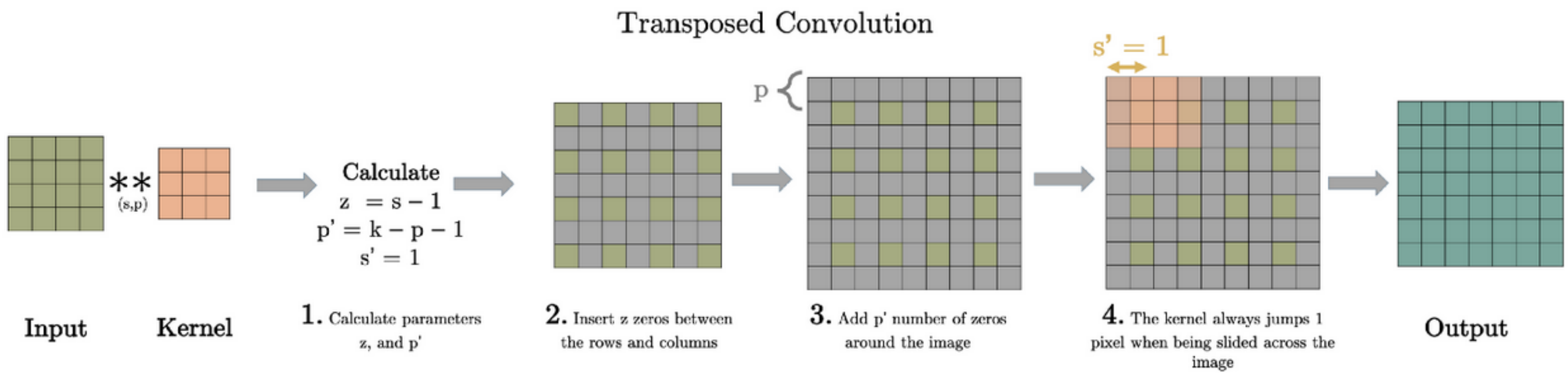
$$\begin{pmatrix}
w_{0,0} & w_{0,1} & w_{0,2} & 0 & w_{1,0} & w_{1,1} & w_{1,2} & 0 & w_{2,0} & w_{2,1} & w_{2,2} & 0 & 0 & 0 & 0 & 0 \\
0 & w_{0,0} & w_{0,1} & w_{0,2} & 0 & w_{1,0} & w_{1,1} & w_{1,2} & 0 & w_{2,0} & w_{2,1} & w_{2,2} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & w_{0,0} & w_{0,1} & w_{0,2} & 0 & w_{1,0} & w_{1,1} & w_{1,2} & 0 & w_{2,0} & w_{2,1} & w_{2,2} & 0 \\
0 & 0 & 0 & 0 & 0 & w_{0,0} & w_{0,1} & w_{0,2} & 0 & w_{1,0} & w_{1,1} & w_{1,2} & 0 & w_{2,0} & w_{2,1} & w_{2,2}
\end{pmatrix}$$

This linear operation takes the input matrix flattened as a 16-dimensional vector and produces a 4-dimensional vector that is later reshaped as the 2×2 output matrix.

source: https://github.com/vdumoulin/conv_arithmetic

Transposed convolutions

- convolution filters moving over the input (ixi)



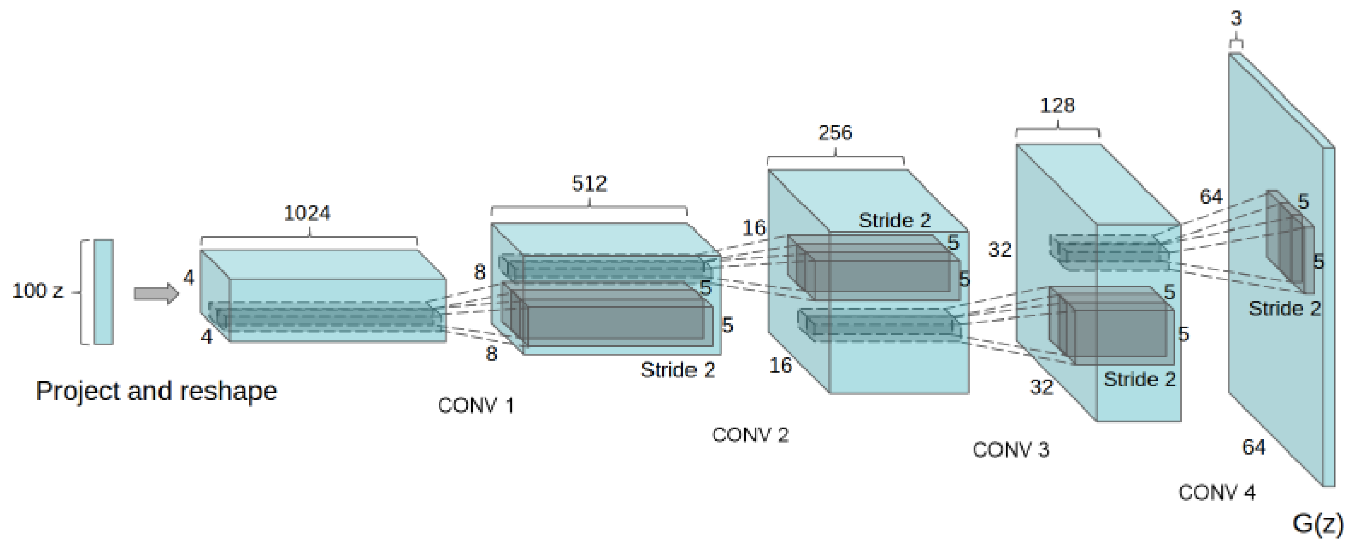
source: https://towardsdatascience.com/what-is-transposed-convolutional-layer-40e5e6e31c11

- parameters - filter size (kxk), padding (=0), strides (=1)

- transposed convolution animations

Convolutional neural networks - upsampling

- transposed convolutions - increase in spatial dimensions



- standard convolutions with manipulated inputs

Neural networks - learning

- loss function in supervised learning context

  - regression $\mathcal{D} = \{\boldsymbol{x}_i \in \mathbb{R}^d, y_i \in \mathbb{R}\}_{i=1}^N$, $NN_{\boldsymbol{\theta}} : \mathbb{R}^d \to \mathbb{R}$

$$loss_{\mathcal{D}}(\boldsymbol{\theta}) = \frac{1}{N} \sum_i (NN_{\boldsymbol{\theta}}(\boldsymbol{x}_i) - y_i)^2$$

  - classification $\{\boldsymbol{x}_i \in \mathbb{R}^d, c_i \in \mathbb{N}\}_{i=1}^N$, $NN_{\boldsymbol{\theta}} : \mathbb{R}^d \to \mathcal{P}(\mathbb{N})$

$$\begin{aligned} loss_{\mathcal{D}}(\boldsymbol{\theta}) &= \frac{1}{N} \sum_i H(p^i = \delta_{c_i}(\mathbb{N}), q^i = NN_{\boldsymbol{\theta}}(\boldsymbol{x}_i)) \\ H(p^i, q^i) &= - \sum_{k \in \mathbb{N}} p_k^i \log(q_k^i) \;\; - \;\; \text{cross-entropy} \end{aligned}$$

- parameters update - sequential stochastic gradient descent

$$\boldsymbol{\theta}_{\text{new}} = \boldsymbol{\theta}_{\text{cur}} - \eta \cdot \nabla_\theta \, loss_{\mathcal{D}}(\boldsymbol{\theta}), \quad \text{where } \eta > 0 - \text{learning rate}$$

  practically - sophisticated optimizers (Adam, RMSProp ...)

Well recognized DL tasks

- classification
  ImageNet Large Scale Visual Recognition Challenge
  AlexNet CNN network won the contest in 2012

- reinforcement learning DeepMind (UK, Google 2014)
  AlhaGo vs. Lee Sedol (4:1, 2016), AlphaGo Zero vs. AlphaGo (100:0, 2017) AlphaZero vs. Stockfish (28:72:0, 2018), Dota 2 tournaments, AlphaFold (2021)

- recurrent neural networks / transformers (2017)
  LSTM, GRU - neurons, NLP tasks, Google Translator, DeepL
  GPT-2, GPT-3, CLIP, DALL-E ... (Open-AI, 2018-2022)

- 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 $(\mathbb{R}^d, \mathcal{B}(\mathbb{R}^d))\}$
  it is metrizable

- standard metric/distance on $\mathcal{P}$, $d : \mathcal{P} \times \mathcal{P} \to [0, \infty]$ such that
  1. $d(P, Q) \geq 0$
  2. $d(P, Q) = 0$ iff P=Q
  3. $d(P, Q) = d(Q, P)$ (symmetry)
  4. $d(P, Q) \geq d(P, Q) + d(P, Q)$ triangle inquality

- divergence on $\mathcal{P}$. $P \times P \to [0, \infty]$ such that
  1. $d(P, Q) \geq 0$
  2. $d(P, Q) = 0$ iff P=Q

Examples of metrics/distances

- total variation

$$\delta(P,Q) = \sup_{A \in \mathcal{A}} |P(A) - Q(A)| = \frac{1}{2} \int |p(\boldsymbol{x}) - q(\boldsymbol{x})| \, d\boldsymbol{x}$$

- Hellinger distance

$$H(P,Q) = \left(\frac{1}{2} \int \left(\sqrt{p(\boldsymbol{x})} - \sqrt{q(\boldsymbol{x})} \right)^2 d\boldsymbol{x} \right)^{1/2}$$

- Wasserstein distance

$$W_p(P,Q) = \left(\inf_{\gamma \in \Gamma(P,Q)} \int d(\boldsymbol{x}, \boldsymbol{y})^p \, d\gamma(\boldsymbol{x} \times \boldsymbol{y}) \right)^{1/p}$$

Frechet distance

- in GAN context, W_2 is also called the Frechet distance - FD
- FD for two multivariate normal distributions
  let $P = N(\boldsymbol{\mu}_1, \Sigma_1)$, $Q = N(\boldsymbol{\mu}_2, \Sigma_1)$, then

$$\begin{aligned}\text{FD}(P,Q) &= W_2(\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \Sigma_1, \Sigma_2) \\ &= ||\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2||_2^2 + \text{Tr}(\Sigma_1 + \Sigma_2 - 2(\Sigma_1\Sigma_2)^{1/2})\end{aligned}$$

- empirical estimate, given two sets of empirical data

$$\mathcal{D}_P = \{\boldsymbol{x}_i\}_{i=1}^N, \mathcal{D}_Q = \{\boldsymbol{y}_i\}_{i=1}^N$$

  compute sample mean and covariance $\boldsymbol{m}_1$, $\boldsymbol{S}_1$ from $\mathcal{D}_P$
  and similarly $\boldsymbol{m}_2$, $\boldsymbol{S}_2$ from $\mathcal{D}_Q$

- finally we compute

$$\text{FD} = W_2(\boldsymbol{m}_1, \boldsymbol{m}_2, \boldsymbol{S}_1, \boldsymbol{S}_2)$$

Kullback-Leibler divergence

- Kullback-Leibler divergence
  let $P, Q \in \mathcal{P}$, $P \ll Q$ (if $Q(d\boldsymbol{x}) = 0$, then $P(d\boldsymbol{x}) = 0$)

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

- properties:
  $D_{\mathsf{KL}}(P||Q) \geq 0$
  $D_{\mathsf{KL}}(P||Q) = 0$ iff $P = Q$, i.e., $D_{\mathsf{KL}}(P||P) = 0$
  $D_{\mathsf{KL}}(P||Q) \neq D_{\mathsf{KL}}(Q||P)$

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

Jensen-Shannon divergence

- Jensen-Shannon divergence - symmetrized KL divergence

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

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

- properties:
  $D_{\mathsf{JSD}}(P||P) = 0$ iff
  $0 \leq D_{\mathsf{JSD}}(P||Q) \leq 1$
  $D_{\mathsf{JSD}}(P||Q) = D_{\mathsf{JSD}}(Q||P)$

- square root of JSD, i.e. $\sqrt{D_{\mathsf{JSD}}(P||Q)}$ is a metric on $\mathcal{P}$

f-divergences

- for a convex function $f : \mathbb{R}_+ \to \mathbb{R}$, lower-semicontinuous such that $f(1) = 0$

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

- KL-divergence, $f(u) = u \log(u)$

- JSD-divergence
  $f(u) = -(u+1)\log\left(\frac{1+u}{2}\right) + u\log(u)$

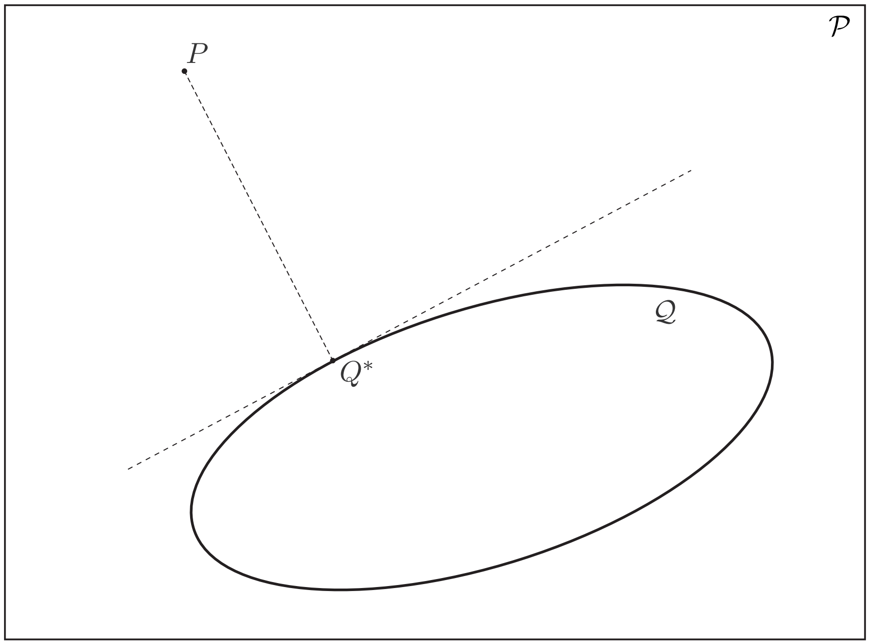
- squared Hellinger distance, $f(u) = (\sqrt{u} - 1)^2$

Reverse information projection (M-projection)

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

$$Q^* = \text{argmin}_{Q \in \mathcal{Q}} D_f(P||Q),$$

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



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

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

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

- $\mathcal{Q}$ 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 the set of data $\{\boldsymbol{x}_i \sim P_D\}_{i=1}^n$, describe distribution P_D

- MLE estimate $P_D \in \mathcal{P}_\theta = \{P_\theta, \theta \in \Theta\}$
  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 then set $P_D = P_{\theta_{\text{mle}}}$

$$\theta_{\text{mle}} = \text{argmax}_\theta\, \mathbb{E}_{\boldsymbol{x}\sim P_D} \log p_\theta(\boldsymbol{x})$$

$$\text{estimate } \theta^*_{\text{mle}} = \text{argmax}_\theta\, \frac{1}{n}\sum_{i=1}^n \log p_\theta(\boldsymbol{x}_i)$$

- optimization in terms of KL-divergence

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

MLE in terms of KL-divergence

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

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

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
  JS-divergence minimalization
  via playing an adversial game between
  generator and discriminator

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 for generator fixed

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

- 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 θ_g of the generator to minimize

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

  w.r.t θ_g for discriminator 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 $V(D_{\theta_d}, \cdot)$

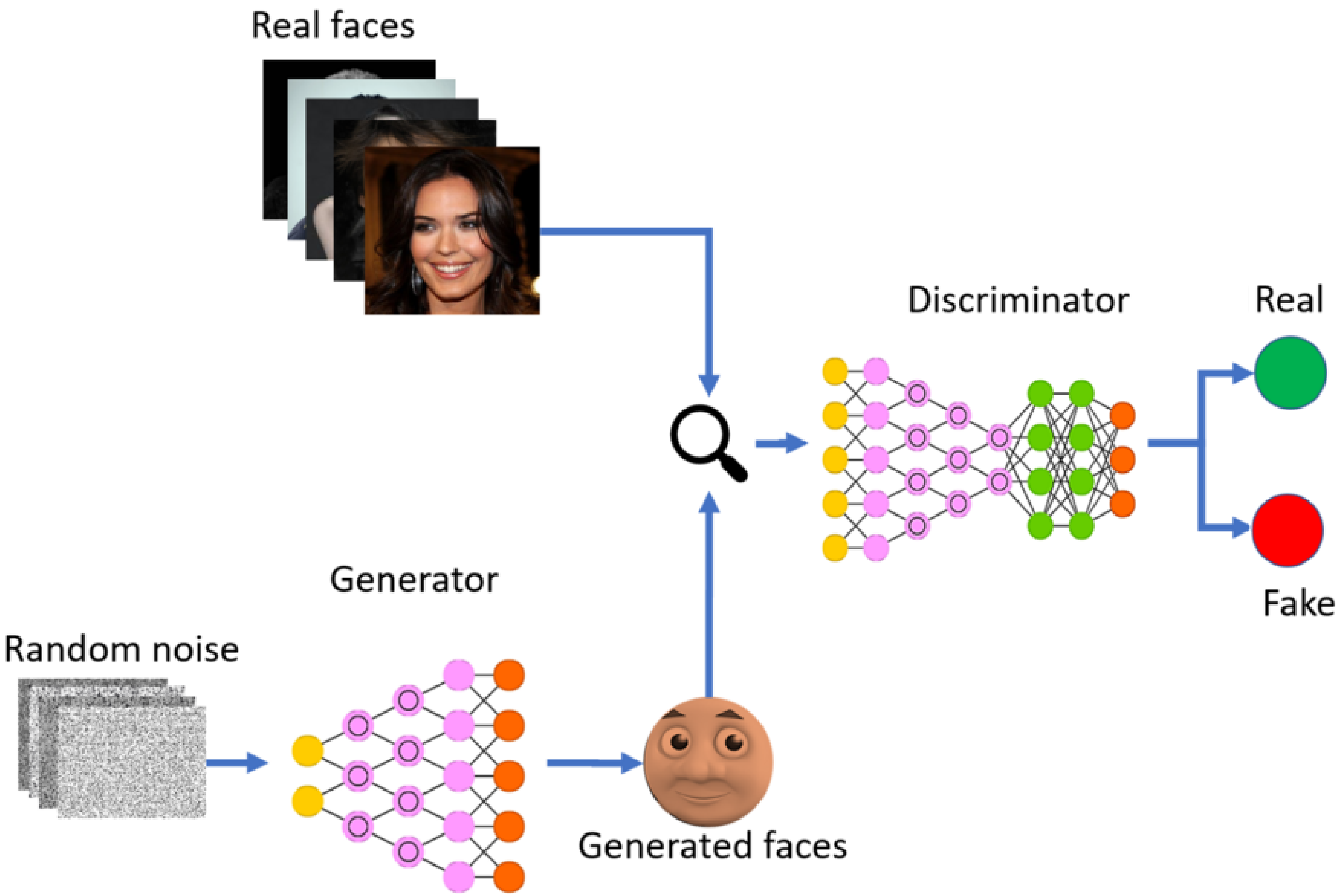
  - for fixed discriminator D_{θ_d} minimize $V(\cdot, G_{\theta_g})$

Theoretical analysis

- **Proposition.** Optimizing $\min_G \max_D V(D, G)$ corresponds to minimizing $D_{\mathsf{JSD}}(P_{\mathsf{data}}||P_G)$. It attains its global minimum (= -log(4)) if and only if $P_{\mathsf{data}} = P_G$.

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 $\{z^{(1)}, \dots, z^{(m)}\}$ from noise prior $p_g(z)$.
  - Sample minibatch of m examples $\{x^{(1)}, \dots, x^{(m)}\}$ 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 $\{z^{(1)}, \dots, z^{(m)}\}$ 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 - 28x28 greyscale images of handwritten digits

  http://yann.lecun.com/exdb/mnist/

MNIST dataset

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

MNIST dataset

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

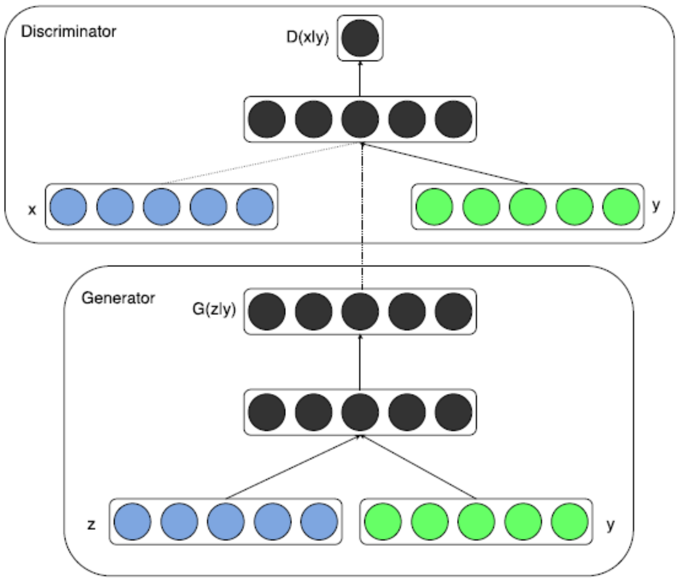
cGAN - 2014

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

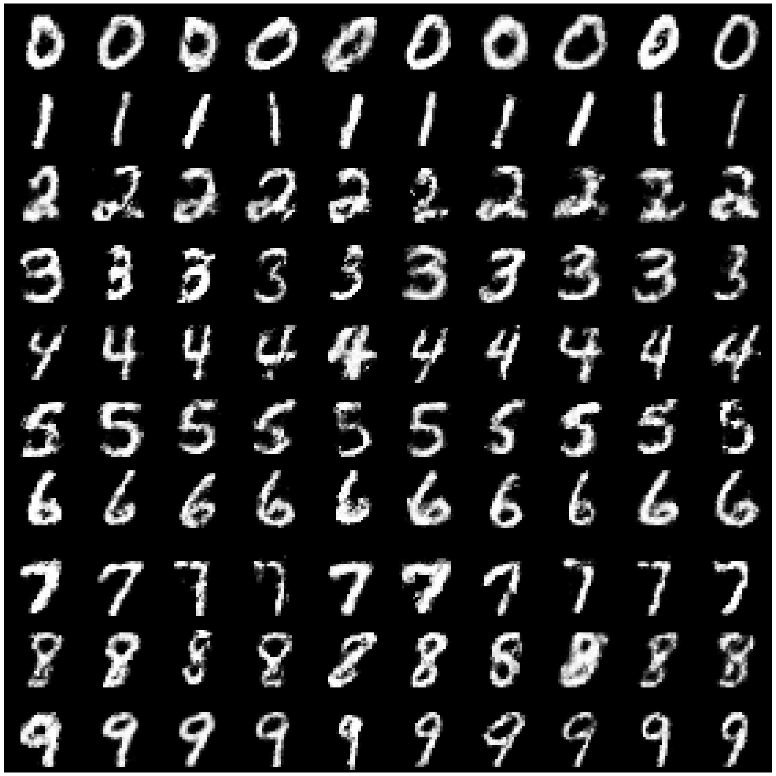
- unconditional vs. conditional GAN, $\boldsymbol{y} - condition$

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

- conditioning by extending latent variable of generator



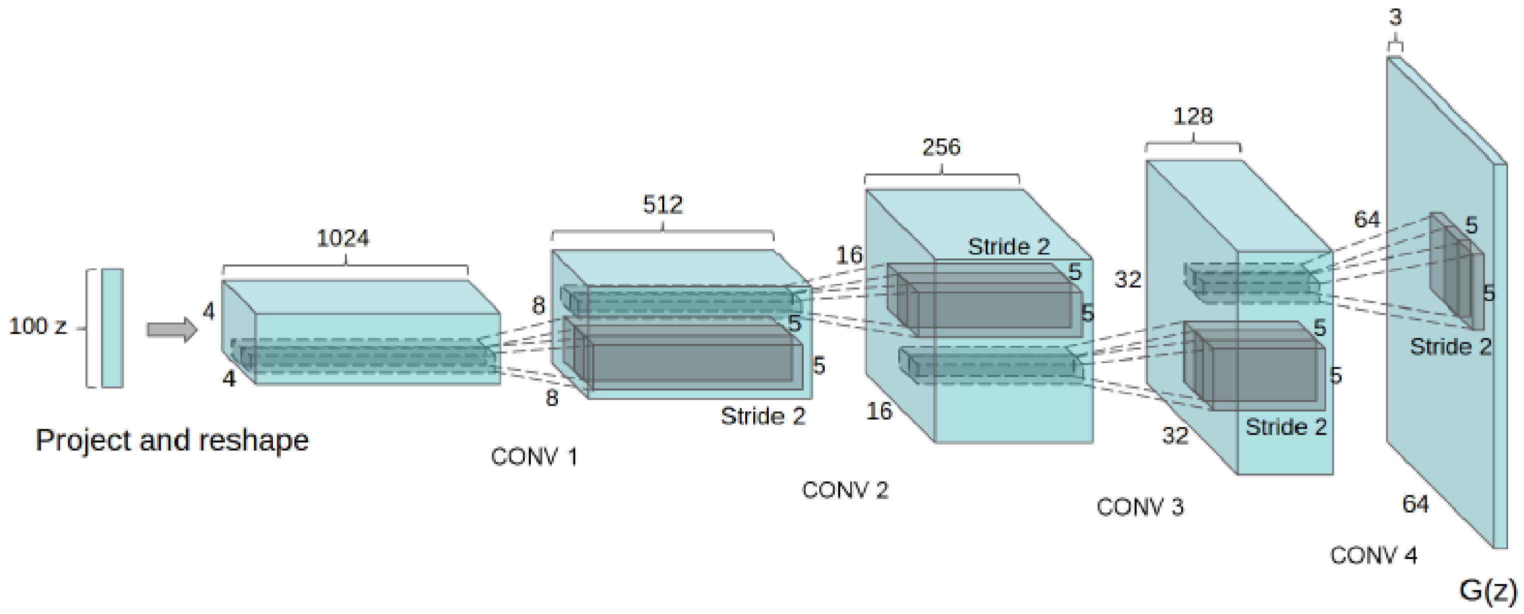
MNIST dataset



DCGAN - 2015

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

- architecture - uses convolutional layers



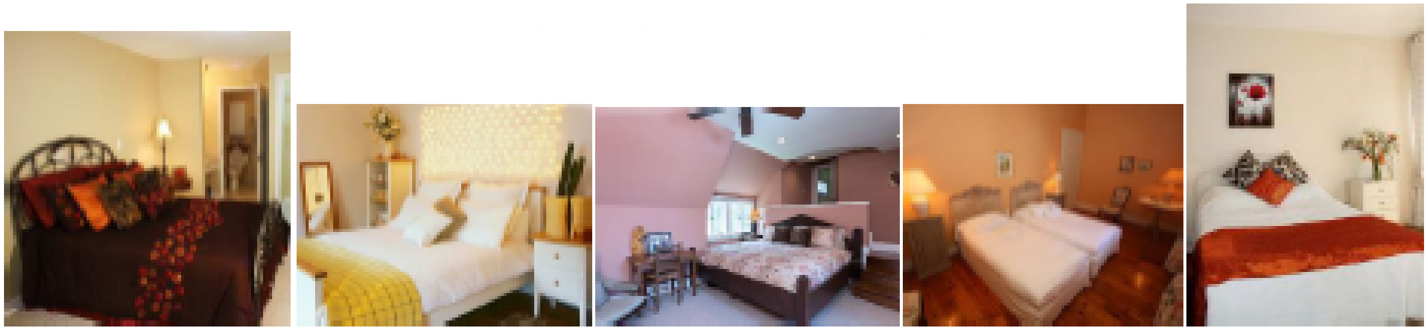
LSUN dataset

- 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

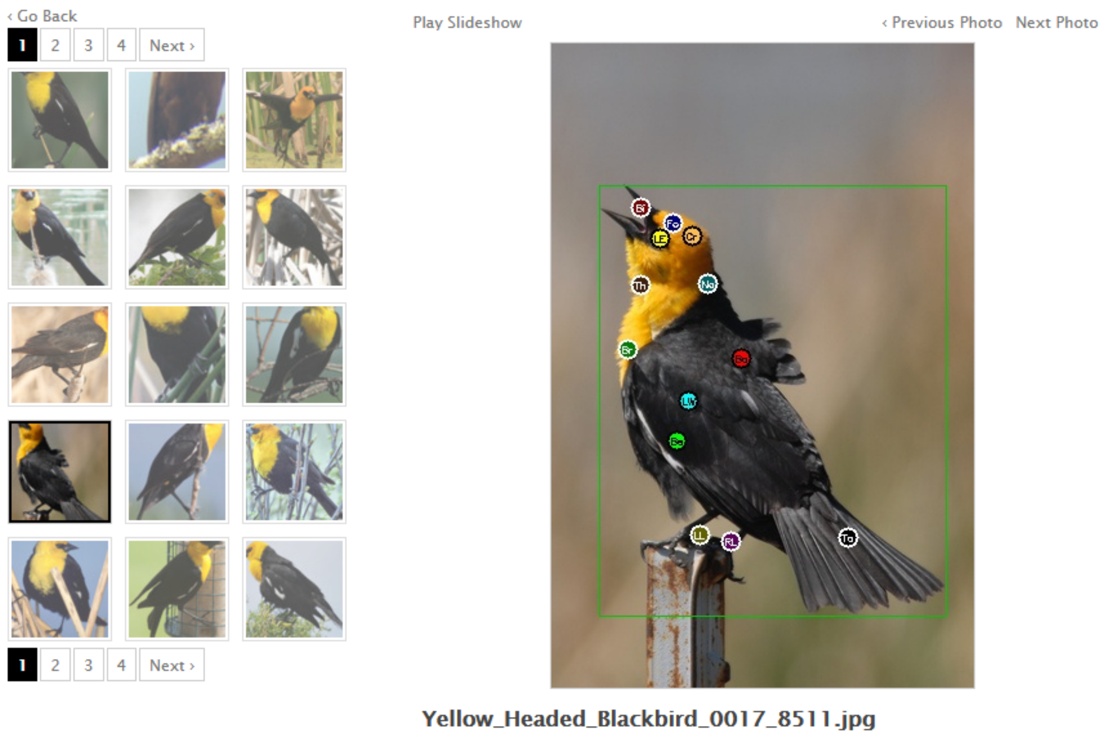
StackGAN - 2016

- *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

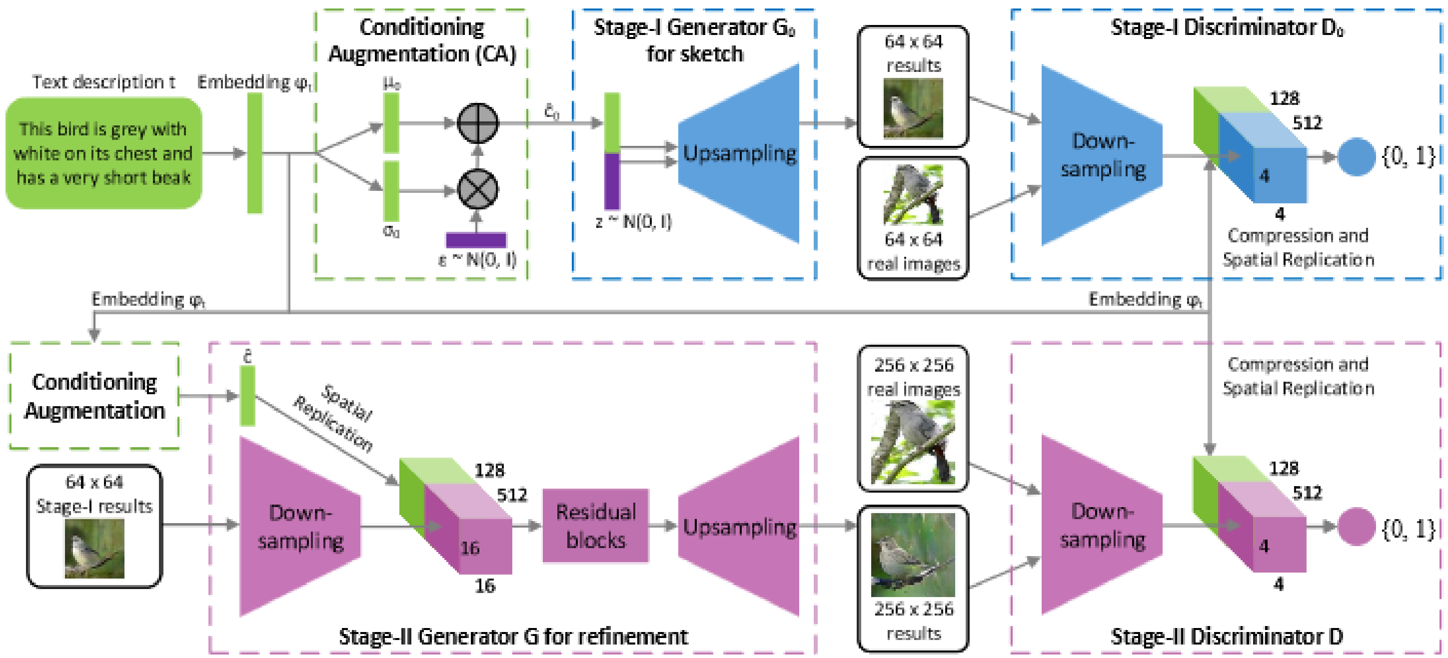


Figure 2. The architecture of the proposed StackGAN. The Stage-I generator draws a low-resolution image by sketching rough shape and basic colors of the object from the given text and painting the background from a random noise vector. Conditioned on Stage-I results, the Stage-II generator corrects defects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

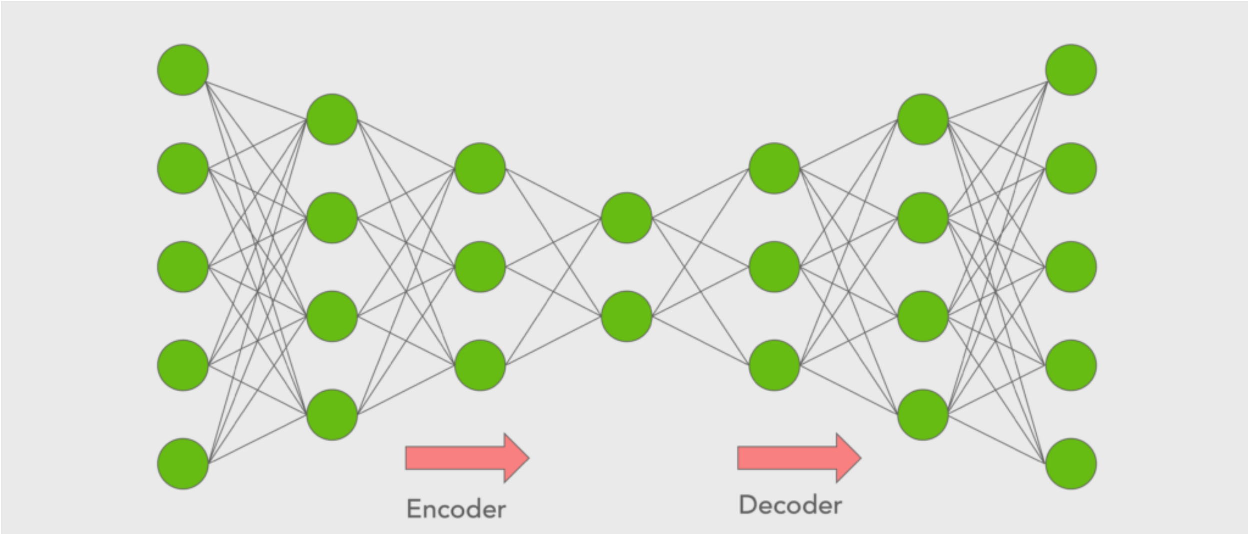
- energy based GAN, discriminator assigns low energy values to real data and high to fake ones - generalized view of loss functions, training - loss minimization

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

where m is a positive margin, $(\cdot)_+ = \max(0,\cdot)$ and $0 \leq D_{\theta_d}$

BEGAN - 2017

- discriminator as autonecoder



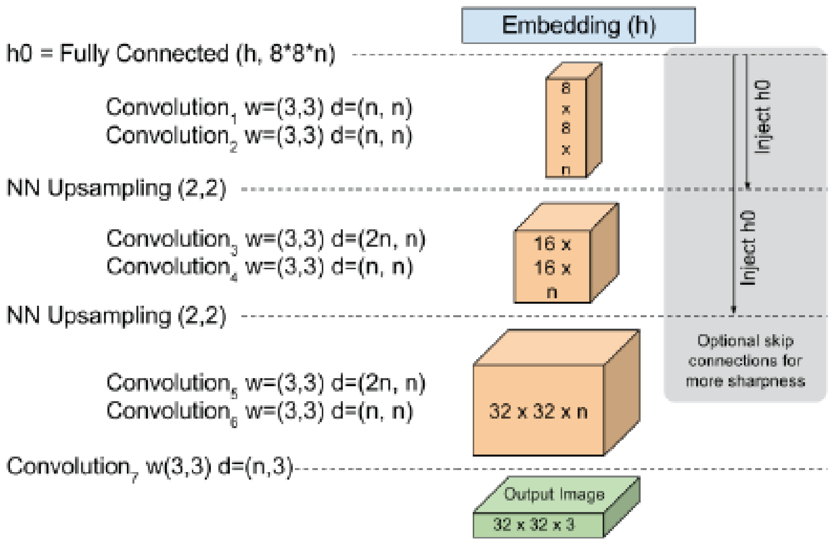
source: https://www.mygreatlearning.com/blog/autoencoder/

- loss - reconstruction errors for real and fake images

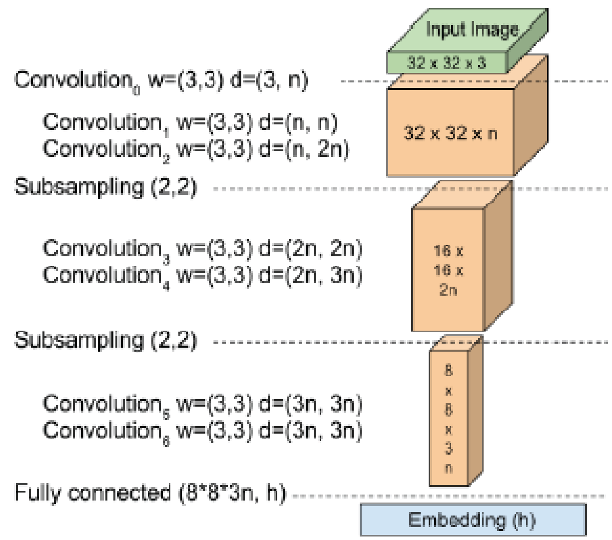
$$
\begin{aligned}
D_\theta(\boldsymbol{x}_{real}) &= ||Dec(Enc(\boldsymbol{x}_{real})) - \boldsymbol{x}_{real}|| \to 0 \\
D_\theta(\boldsymbol{x}_{fake}) &= ||Dec(Enc(\boldsymbol{x}_{fake})) - \boldsymbol{x}_{fake}|| \to \infty
\end{aligned}
$$

BEGAN - 2017

- architecture of generator/decoder and encoder



(a) Generator/Decoder



(b) Encoder

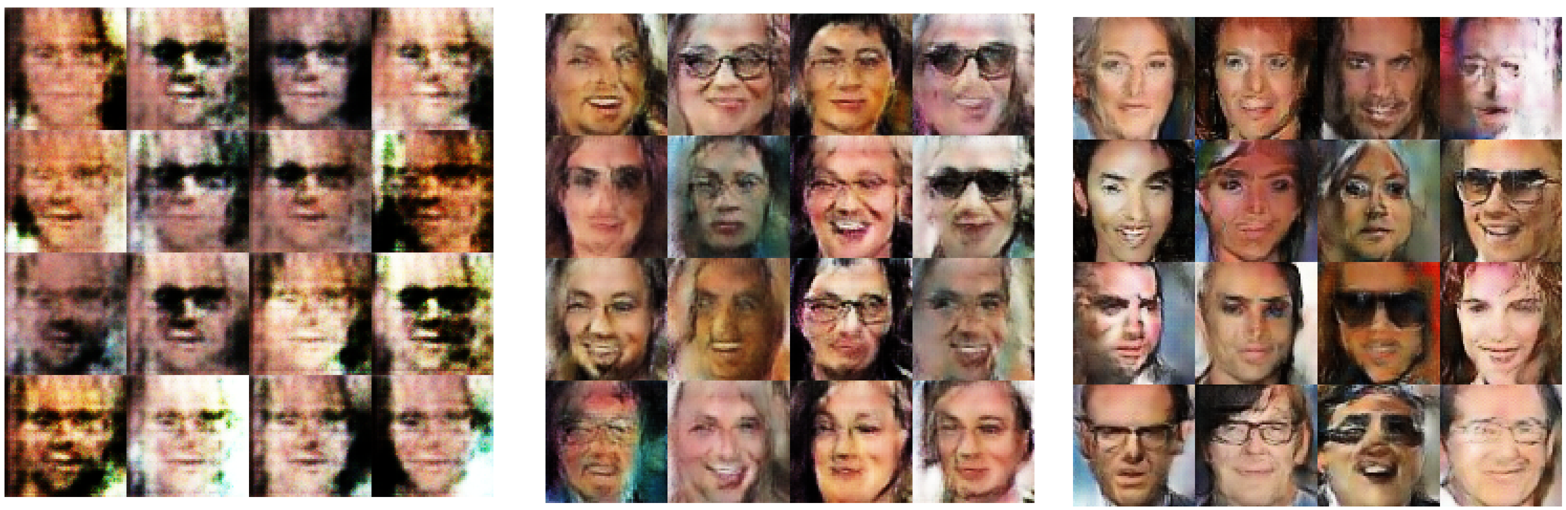
CelebA dataset

- CelebA dataset - 202599 annotated (40 attributes) celebrity portraits

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

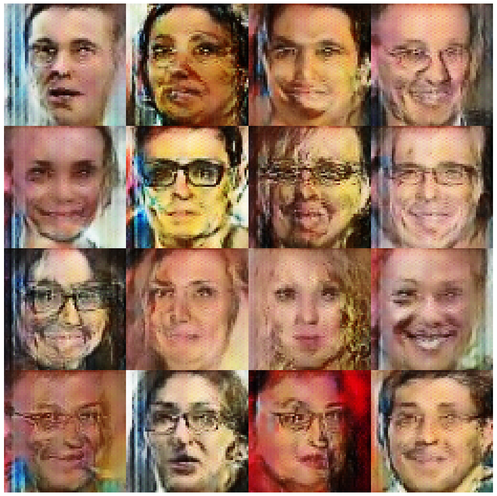
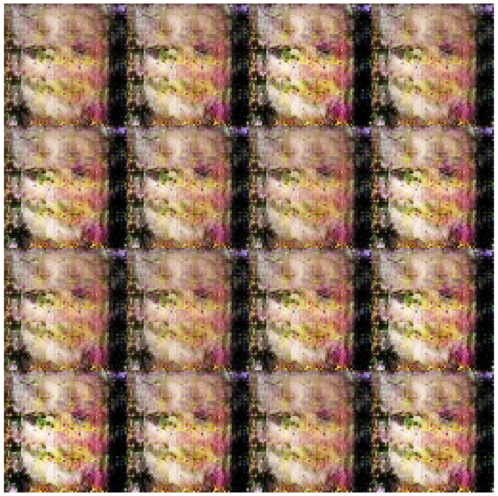
CelebA - DCGAN

- learning progess



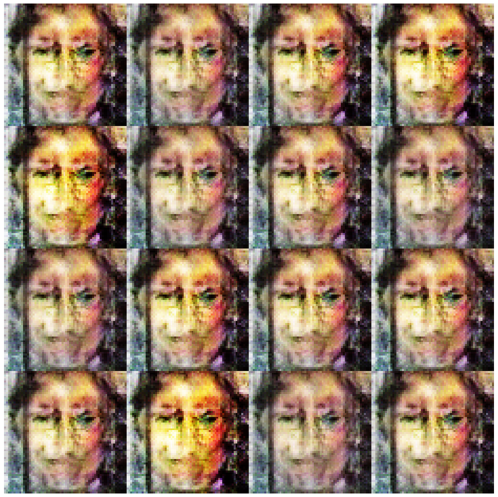
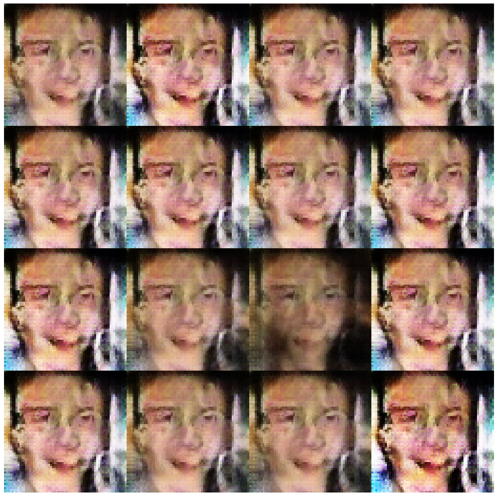
CelebA - DCGAN

- detoriation starts

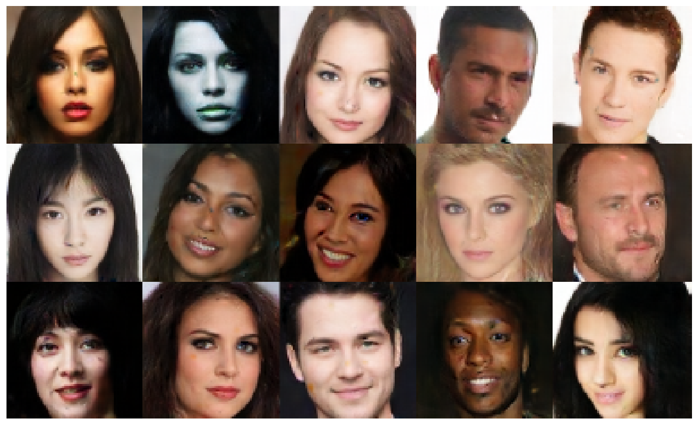
DCGAN

- mode collapse

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 - 2017

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

- CelabA HQ dataset - 30000 imgs at 1024x1024 resolution

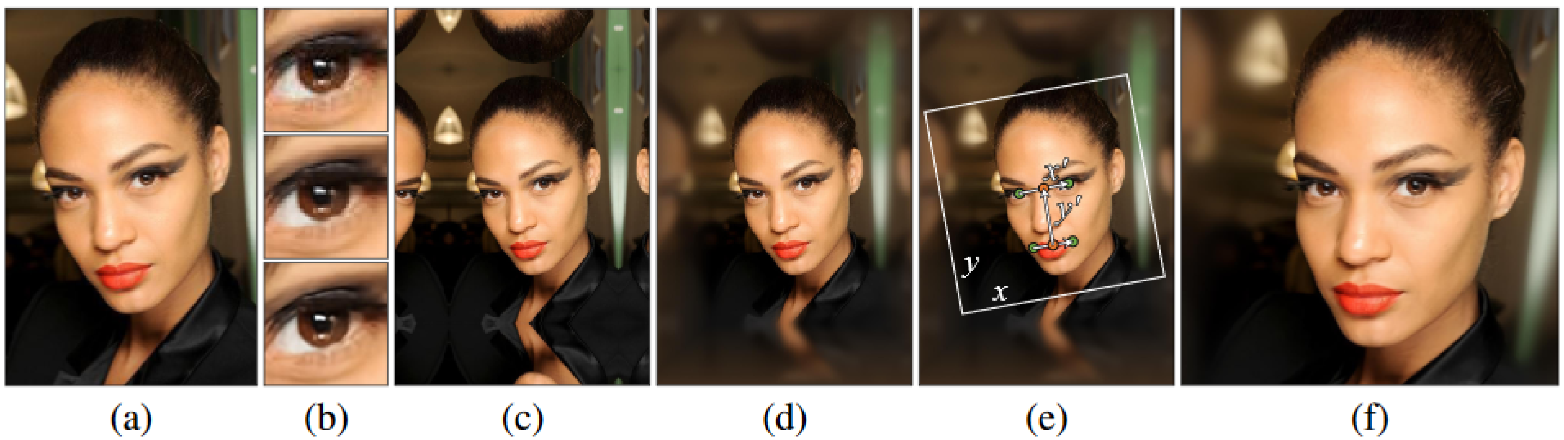


Figure 8: Creating the CELEBA-HQ dataset. We start with a JPEG image (a) from the CelebA in-the-wild dataset. We improve the visual quality (b,top) through JPEG artifact removal (b,middle) and 4x super-resolution (b,bottom). We then extend the image through mirror padding (c) and Gaussian filtering (d) to produce a visually pleasing depth-of-field effect. Finally, we use the facial landmark locations to select an appropriate crop region (e) and perform high-quality resampling to obtain the final image at 1024×1024 resolution (f).

PGGAN - 2017

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

- architecture - progressive growing of convolutional layers

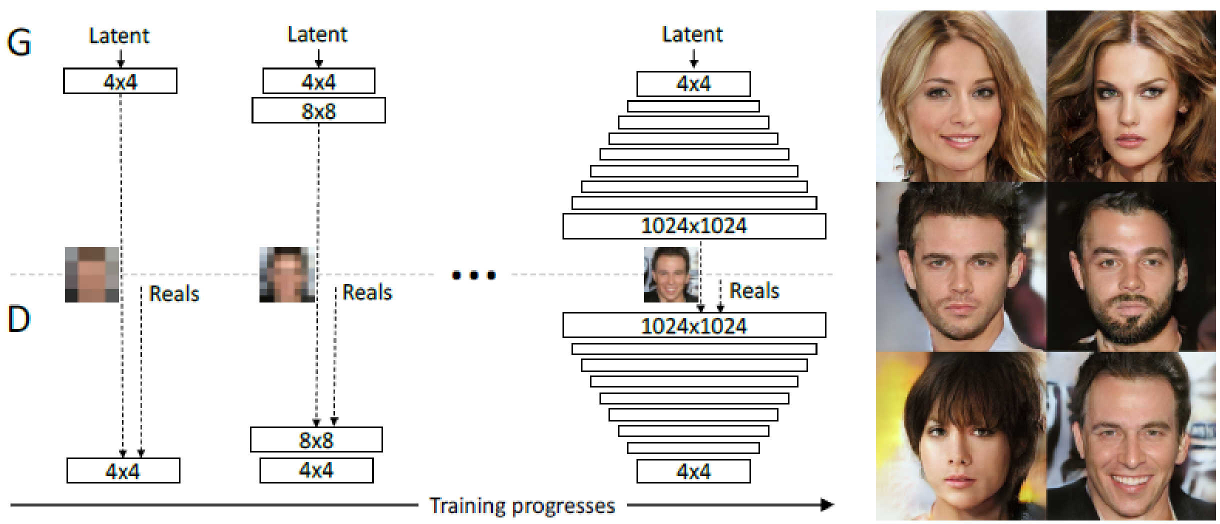


Figure 1: Our training starts with both the generator (G) and discriminator (D) having a low spatial resolution of 4×4 pixels. As the training advances, we incrementally add layers to G and D, thus increasing the spatial resolution of the generated images. All existing layers remain trainable throughout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

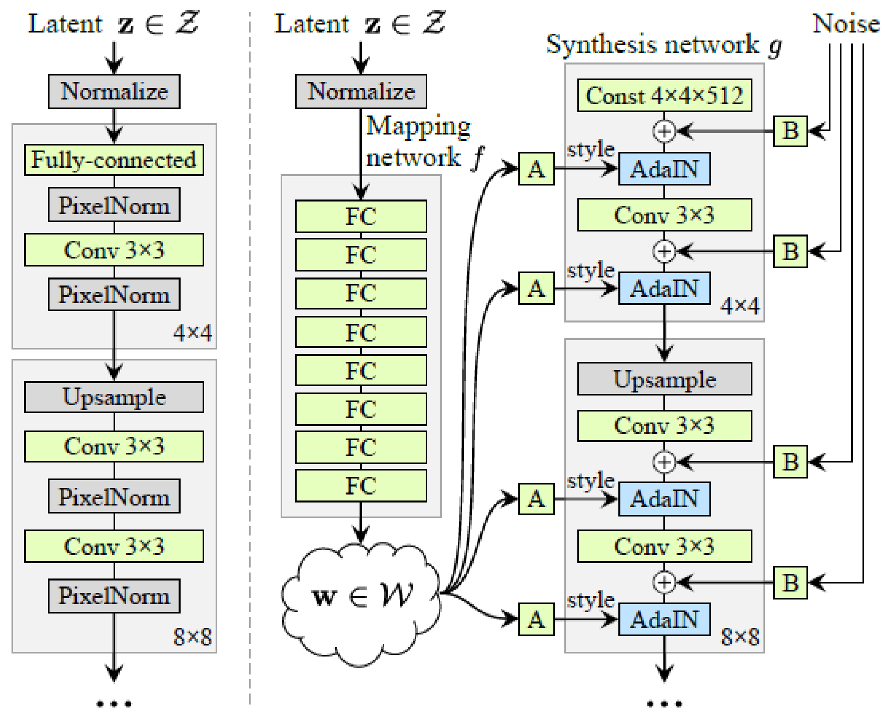
PGGAN - 2017

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.

- https://github.com/tkarras/progressive_growing_of_gans

StyleGAN - 2018

- *A Style-Based Generator Architecture for Generative Adversarial Networks* https://arxiv.org/abs/1812.04948



(a) Traditional (b) Style-based generator

Figure 1. While a traditional generator [30] feeds the latent code though the input layer only, we first map the input to an intermediate latent space $\mathcal{W}$, which then controls the generator through adaptive instance normalization (AdaIN) at each convolution layer. Gaussian noise is added after each convolution

StyleGAN - 2018

- style disentanglement

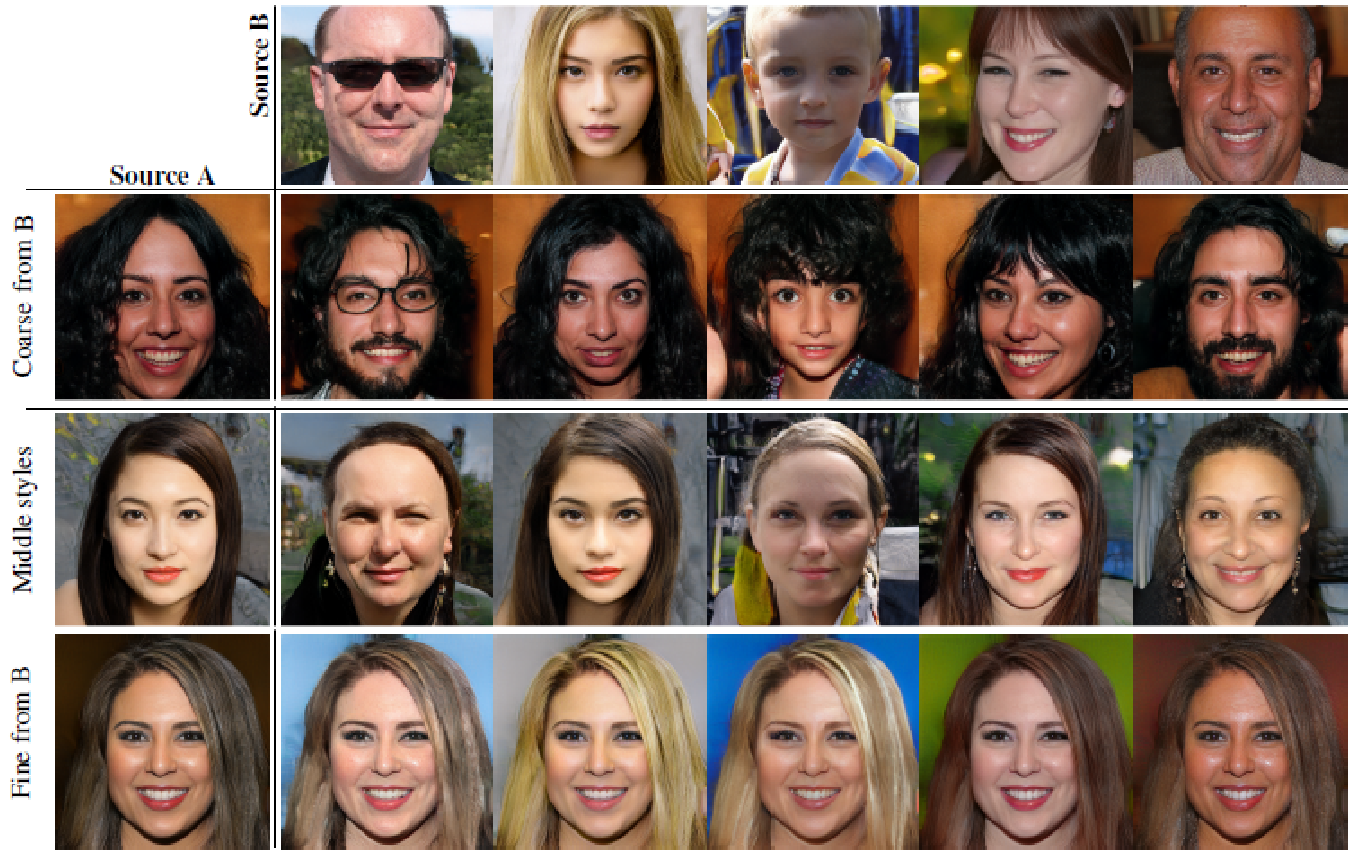


Figure 3. Two sets of images were generated from their respective latent codes (sources A and B); the rest of the images were generated by copying a specified subset of styles from source B and taking the rest from source A. Copying the styles corresponding to coarse spatial resolutions (4^2 – 8^2) brings high-level aspects such as pose, general hair style, face shape, and eyeglasses from source B, while all colors

StyleGAN2 - 2019

- *Analyzing and Improving the Image Quality of StyleGAN*
  https://arxiv.org/abs/1912.04958

Figure 1. Instance normalization causes water droplet -like artifacts in StyleGAN images.

StyleGAN2 - 2019

- *Analyzing and Improving the Image Quality of StyleGAN*

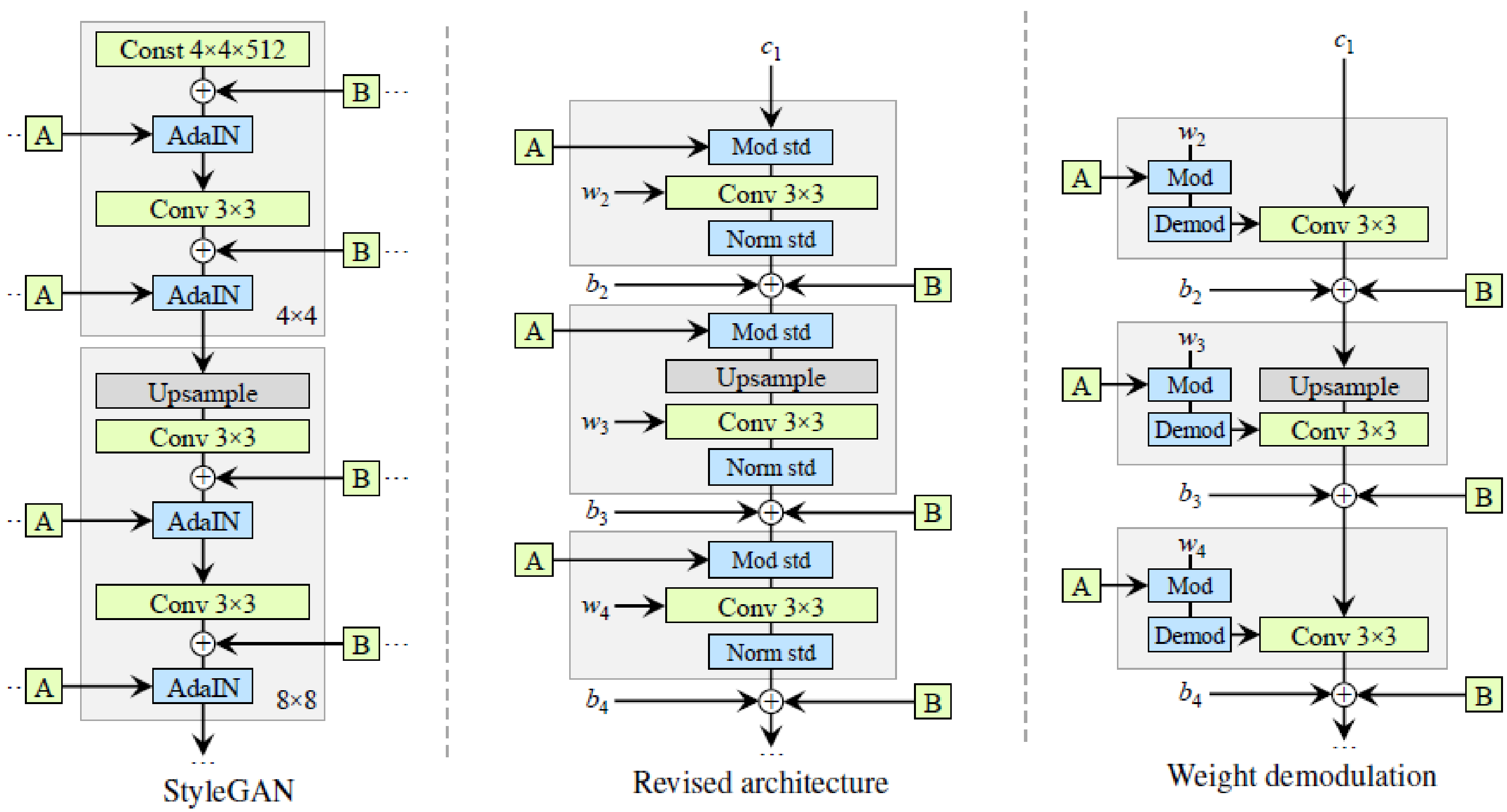


Figure 2. We redesign the architecture of the StyleGAN synthesis network.

StyleGAN2 - 2019

- *abandonment of progressive growing*

Figure 6. Progressive growing leads to "phase" artifacts. In this example the teeth do not follow the pose but stay aligned to the camera, as indicated by the blue line.

- https://github.com/NVlabs/stylegan2

StyleGAN-ADA - 2020

- *Training Generative Adversarial Networks with Limited Data Using Adaptive Discriminator Adaptation - (ADA)* https://arxiv.org/abs/2006.06676

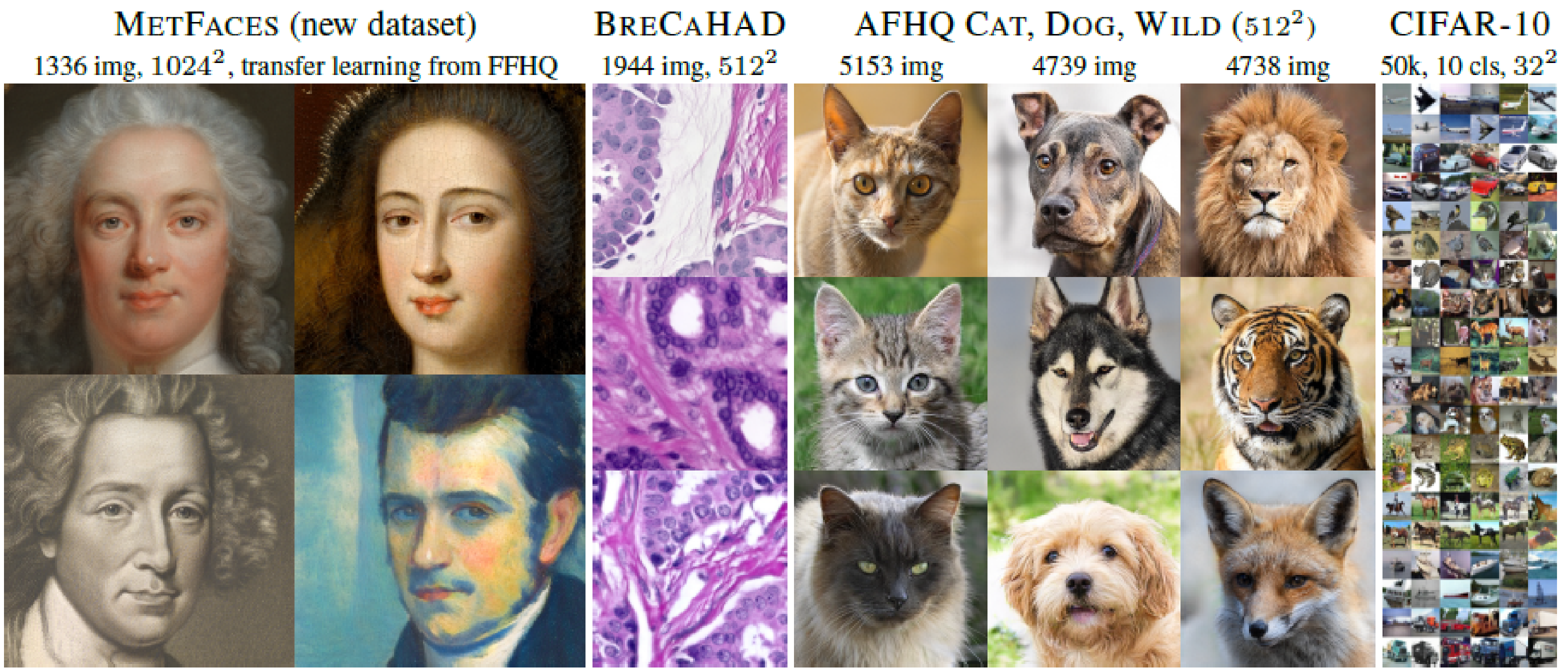


Figure 10: Example generated images for several datasets with limited amount of training data, trained using ADA.

- https://github.com/NVlabs/stylegan

StyleGAN3 - 2021

- *Alias-Free Generative Adversarial Networks (StyleGAN3)*
  https://arxiv.org/abs/2106.12423

- https://nvlabs.github.io/stylegan3

StyleGAN-XL - 2022

- *StyleGAN-XL: Scaling StyleGAN to Large Diverse Datasets*
  https://arxiv.org/abs/2202.00273

Fig. 1. Class-conditional samples generated by StyleGAN3 (left) and StyleGAN-XL (right) trained on ImageNet at resolution 256^2.

- https://github.com/autonomousvision/stylegan_xl

ImageNet

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

- ImageNet-1K (1,281/50/100k images) 1000 categories used in ILSVRC 2012-2017 challenges

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.

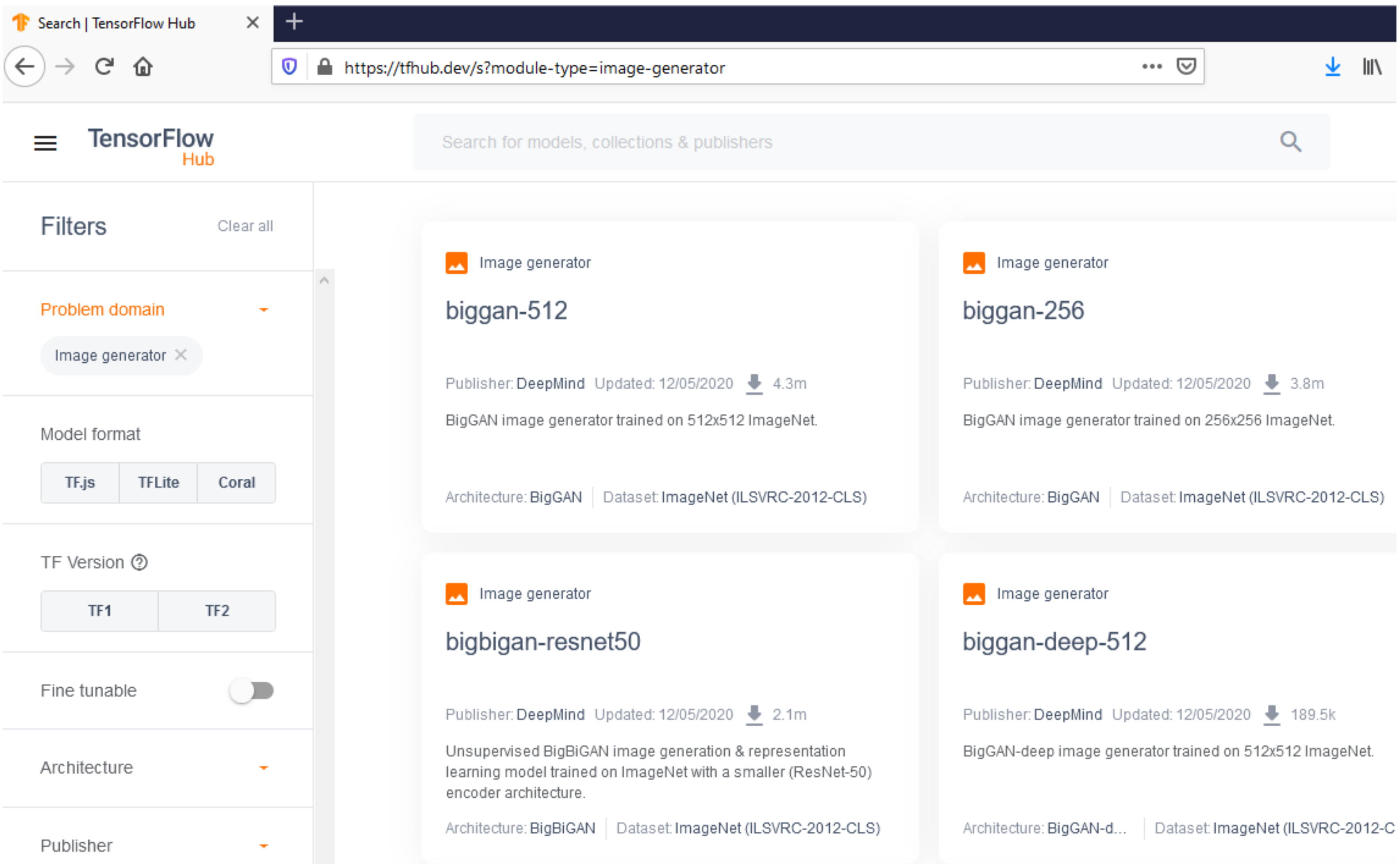
BigGAN - 2019

- architecture - convolutional layers, no pg

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

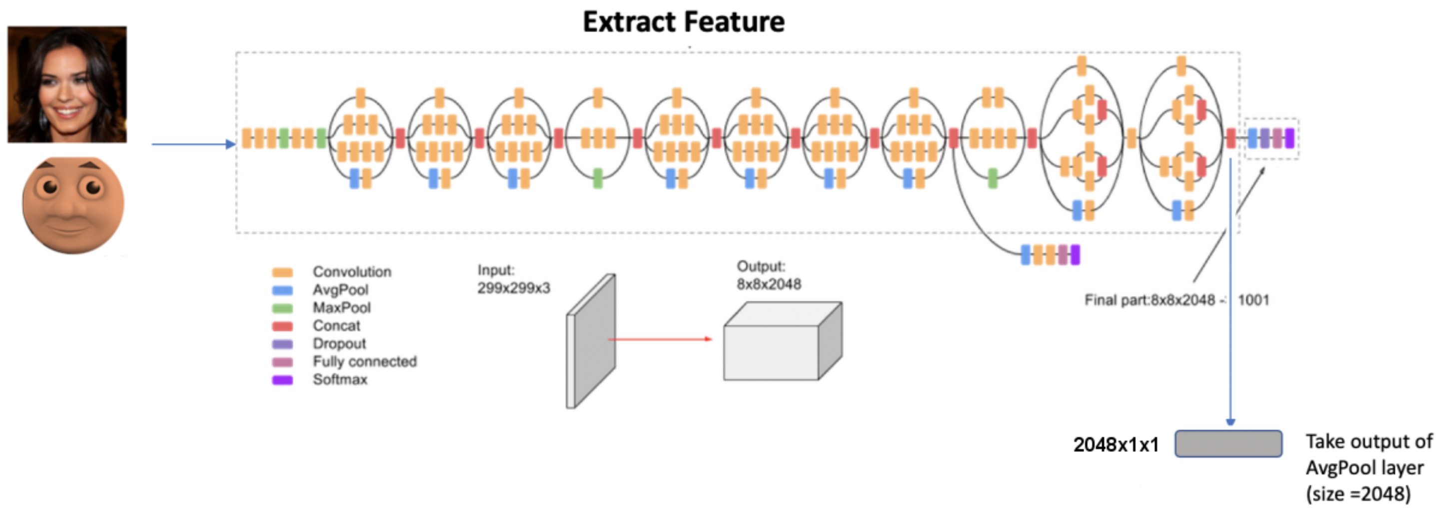
BigGAN - 2019

- TensorFlow Hub - pretrained weights



Frechet Inception Distance

- Inception V3 network - ImageNet classification



source: https://alquarizm.files.wordpress.com/2019/03/image-4.png?w=1280

- $\mathcal{D}_{real} = \{\text{IncV3}(\boldsymbol{x}^i_{real})\}_{i=1}^{\text{N\_FID}}$, $\mathcal{D}_{fake} = \{\text{IncV3}(\boldsymbol{x}^i_{fake})\}_{i=1}^{\text{N\_FID}}$

$$\text{FID}(\mathcal{D}_{real}, \mathcal{D}_{fake}) = W_2(\boldsymbol{m}_1, \boldsymbol{m}_2, \boldsymbol{S}_1, \boldsymbol{S}_2)$$

SOTA FID

OpenAI DALL-E - 2021

- DALL-E is a 12-billion parameter version of GPT-3 trained to generate images from text descriptions, using a dataset of text-image pairs.

- *The supercomputer developed for OpenAI is a single system with more than 285,000 CPU cores, 10,000 GPUs and 400 gigabits per second of network connectivity for each GPU server.*

- https://openai.com/blog/dall-e

Stable diffusion - 2022

- text-to-image model, trained on LAION-5B dataset which consists of 5.85 billion image-text pairs
- training - 256 NVIDIA A100 GPUs on AWS
  150,000 GPU-hours (24 days) at a cost of \$ 600,000
- weights were made public, open-source model
- backed by Stability AI company, based in London
- several APIs available to play with ...

Open questions

- What sorts of distributions can GANs model?
- What can we say about the global convergence of the training dynamics?
- How does GAN training scale with batch size?
- How can we scale GANs beyond image synthesis? (text, audio, computer-aided drug design - https://insilico.com)

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