HLML
TORONTO

Generative Adversarial Networks

Jeremy Rothschild

Generative Models

Models in which new instances of the data are generated, in contrast to some of the discriminative models we have seen

For example a generative model could be a network trained on images that generates new pictures of dogs or pineapples

Formally: Generative models capture the joint probability of the data and labels p(X, Y), or simply p(X) if there are no labels

Examples of generative models we have seen: Boltzmann Machine, RBM, Autoencoders, RNNs

Problems with generative models studied

Training is not easy, backpropagation is not possible in RNNs, RBMs

Maximizing the log-likelihood is complicated and often requires us to make approximations (for example, approximating the partition function in RBM training)

Maximizing the likelihood is problematic if our model generates many points in the regions with very few data points

To see this: note that maximizing the log likelihood of the data under the model is equivalent to minimizing the KL divergence between the data distribution and model distribution.

$$D_{KL}(p|q) = \int dx q(x) \log\left(\frac{p(x)}{q(x)}\right)$$

Equivalence to maximizing log-likelihood

$$D_{KL}(p_{data}|p_{model}) = \int dx p_{data}(x) \log\left(\frac{p_{data}(x)}{p_{model}(x)}\right)$$
$$D_{KL}(p_{data}|p_{model}) = -S[p_{data}] - \langle \log(p_{model}) \rangle_{data}$$

Evaluate expression $D_{KL}(p_{data}|p_{model})$. Where can our model go wrong if we maximize the log-likelihood?

Note that $D_{KL}(p_{data}|p_{model})$ is not a proper metric, but that the Jensen-Shannon divergence is

$$JS(p,q) = \frac{1}{2}\left[D_{KL}\left(p\left|\frac{p+q}{2}\right.\right) + D_{KL}\left(q\left|\frac{p+q}{2}\right.\right)\right]$$

Penalizing models

Goal: have a network that generates samples close to p_{data} and little sampling ‘far’ from the data in it’s sample space

To circumvent the use of a log-likelihood, the training needs to have some penalty for models that generate samples away from samples from our data distribution

GANs and VAEs (Variational Autoencoders) do this by having 2 networks with different functionalities: generally a generator and a discriminator

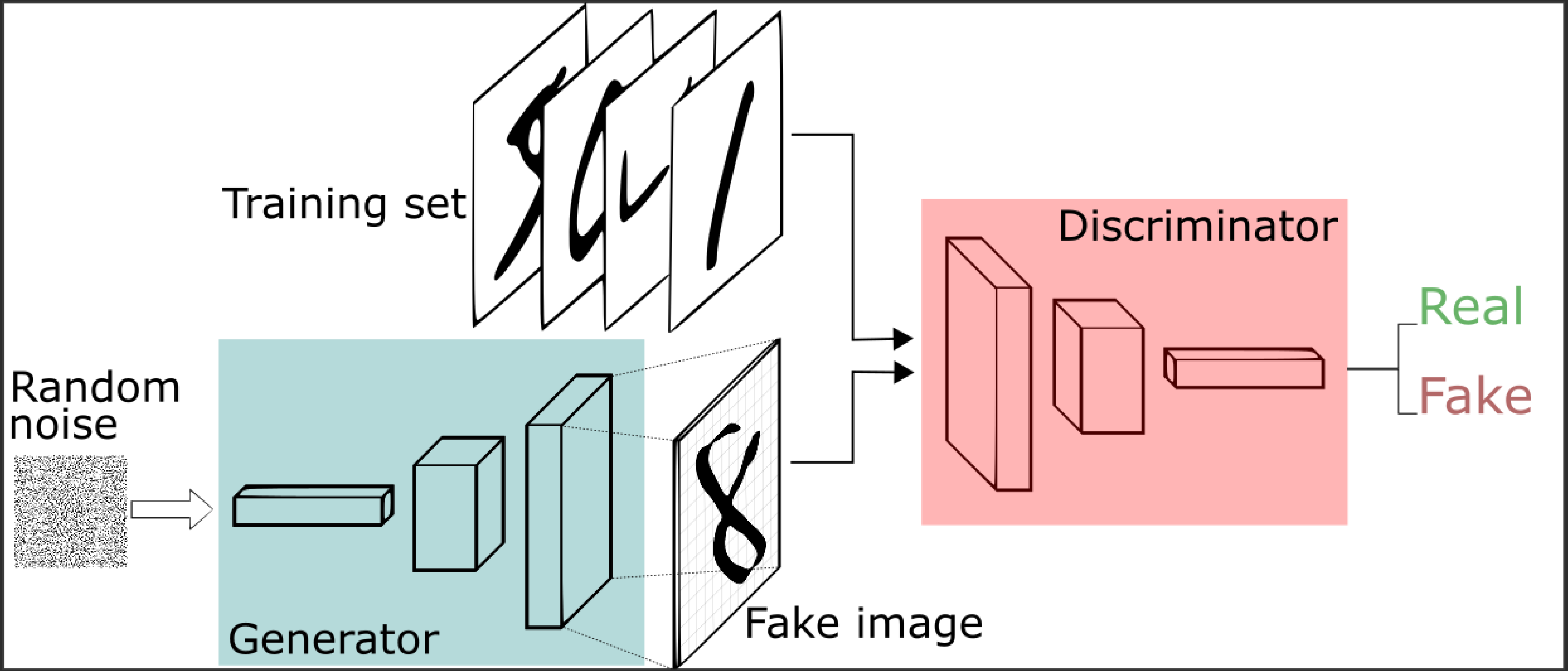
Penalizing models

Goal: have a network that generates samples close to p_{data} and little sampling ‘far’ from the data in it’s sample space

To circumvent the use of a log-likelihood, the training needs to have some penalty for models that generate samples away from samples from our data distribution

GANs and VAEs (Variational Autoencoders) do this by having 2 networks with different functionalities: generally a generator and a discriminator

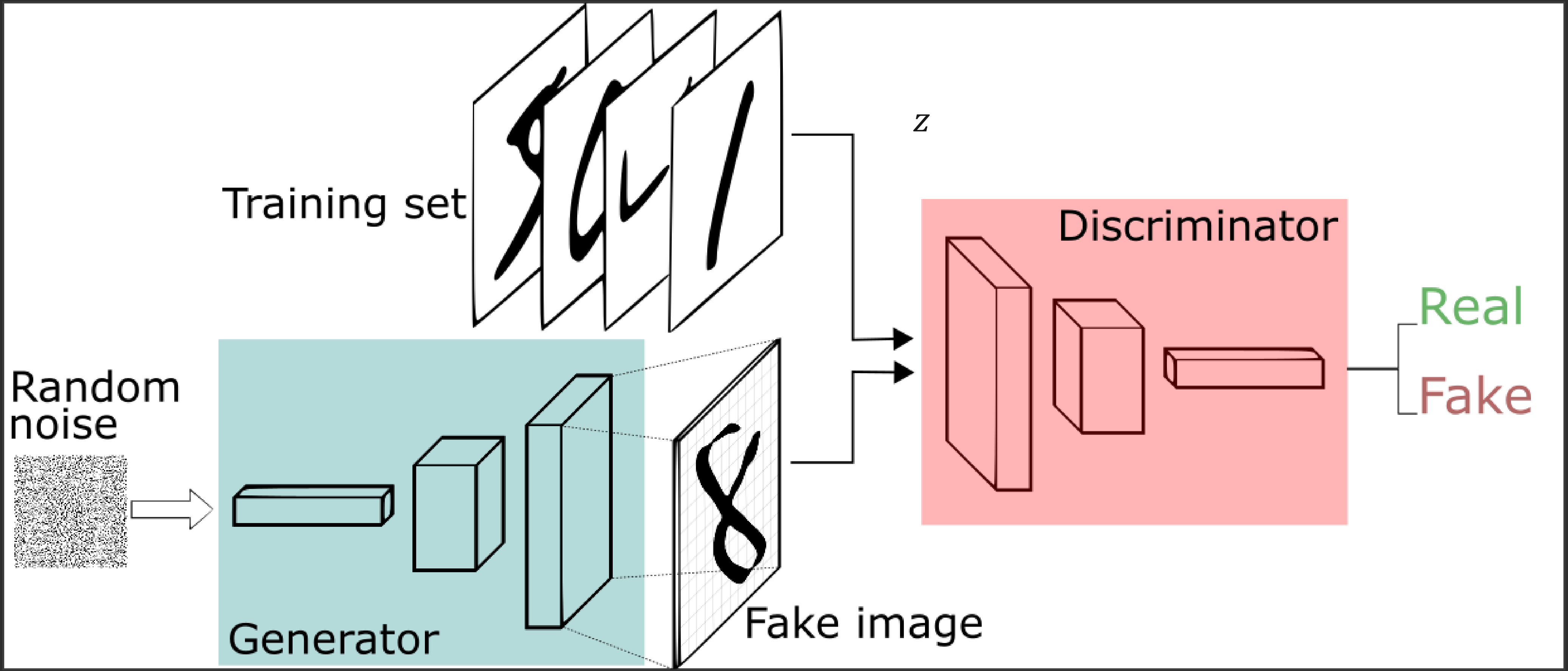
GANs framework



GANs framework

Training set
x

Discriminator network
$D(x;\theta_d)$



$D(x;\theta_d) = 1$

$D(G(z);\theta_d)$
$= 0$

Prior on latent space
$p_z(z)$

Mapping onto data space
$G(z;\theta_g)$

$$V(D,G) = E_{x\sim p_{data}} \log\big(D(x)\big) + E_z \log\Big(1 - D\big(G(z)\big)\Big)$$

This is the original cost function from *Goodfellow, 2014*. Networks (D, G) play a two-player minimax game, i.e.

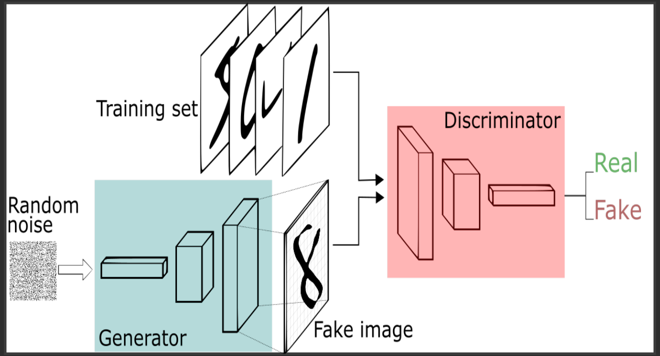
$$\min_G \max_D V(D,G)$$

It can be shown that this is equivalent to min-maxing the Jensen-Shannon divergence

$$JS(p,q) = \frac{1}{2}\left[D_{KL}\left(p\,\middle|\,\frac{p+q}{2}\right) + D_{KL}\left(q\,\middle|\,\frac{p+q}{2}\right)\right]$$

Proposition 1: For G fixed, the optimal discriminator D is $D_G^*(x) = \frac{p_{data}(x)}{p_{data}(x)+p_g(x)}$.

Theorem 1: The global minimum of the virtual training criterion C_G is achievable if and only if $p_g = p_{data}$. At that point $C_G = -\log 4$.

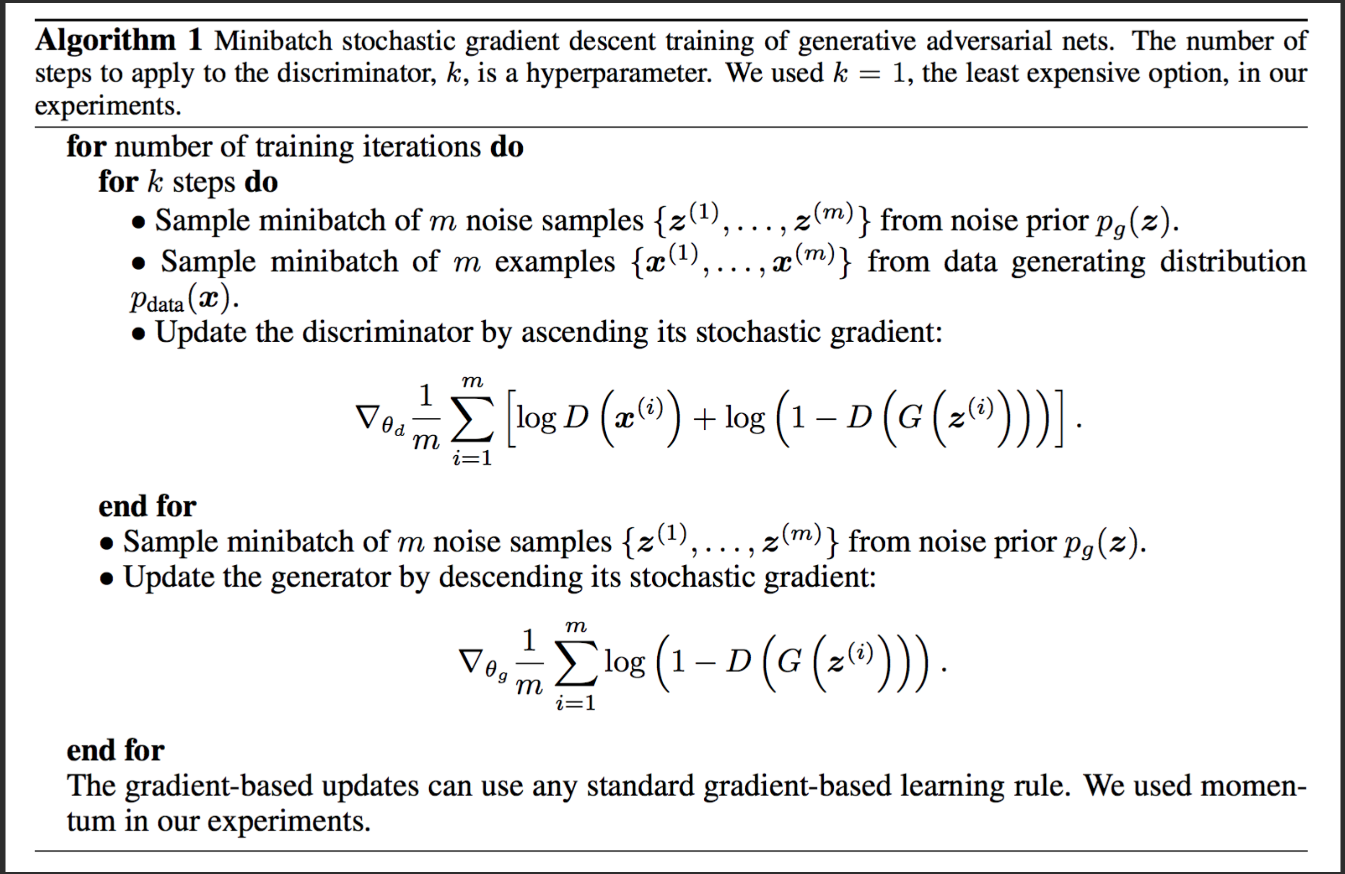


Proposition 1: If G and D have enough capacity, and at each step of the training algorithm the discriminator is allowed to reach its optimum given G, and p_g is updated so as to improve the criterion

$$E_{x\sim p_{data}} \log\big(D_G^*(x)\big) + E_z \log\Big(1 - D_G^*\big(G(z)\big)\Big)$$

then p_g converges to p_{data}.

Training



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.

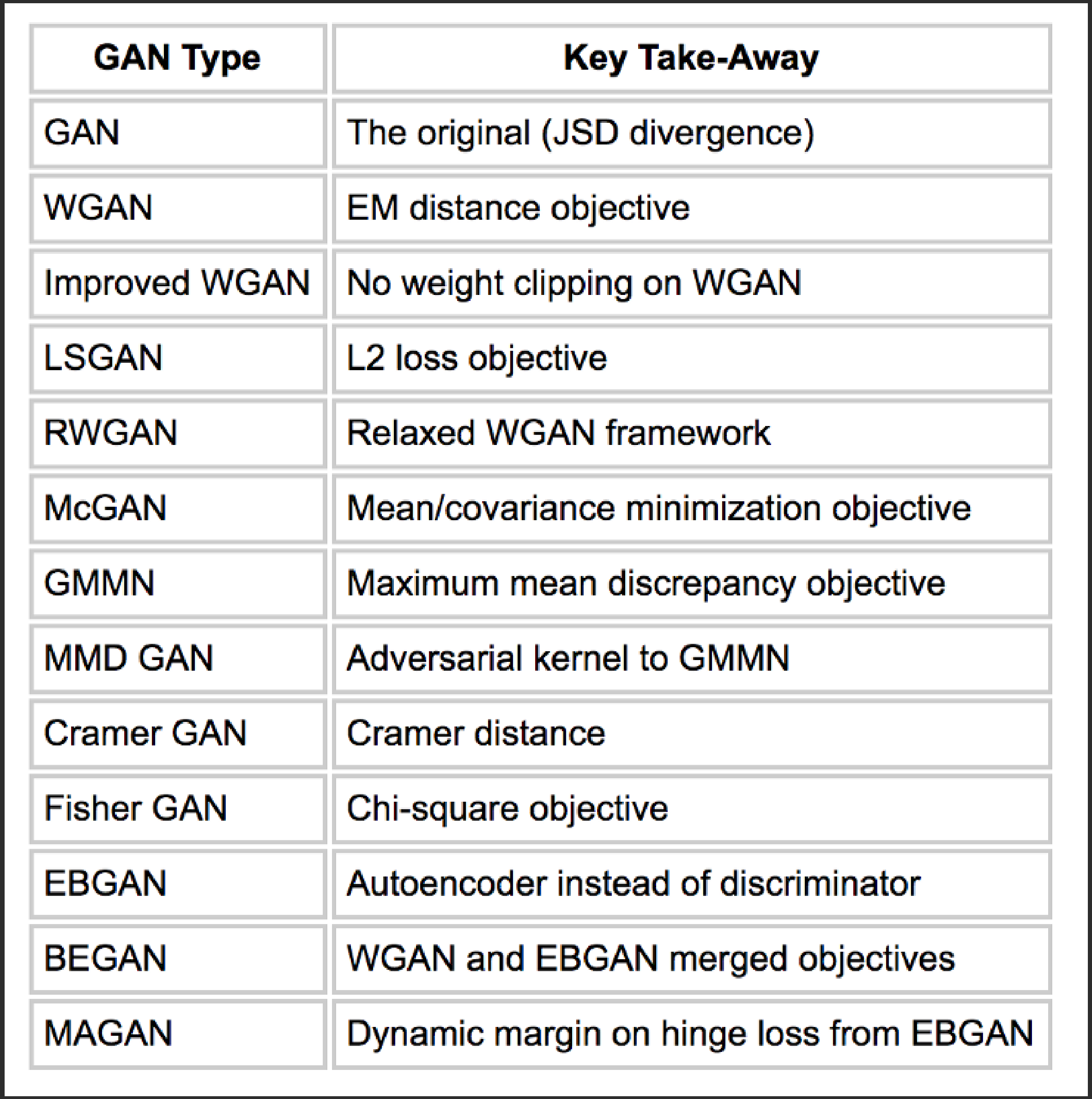
This is the general form of training for most GANs. Things to note:

- Multiple discriminator update steps for each
- Leaves a lot of room to select for sampling strategy, choice of gradient-based learning rule, cost functions
- Where does the algorithm fail? Where are local minimums?
- Is it possible to use already learned networks?

What cost function do we select for training the discriminator and the generator? This choice has diversified the ecology landscape of GANs as they are today

The choice appears to be very problem dependent and it is said that training GANs is very difficult… (why?)

Implementations can be found on github.



GAN Type	Key Take-Away
GAN	The original (JSD divergence)
WGAN	EM distance objective
Improved WGAN	No weight clipping on WGAN
LSGAN	L2 loss objective
RWGAN	Relaxed WGAN framework
McGAN	Mean/covariance minimization objective
GMMN	Maximum mean discrepancy objective
MMD GAN	Adversarial kernel to GMMN
Cramer GAN	Cramer distance
Fisher GAN	Chi-square objective
EBGAN	Autoencoder instead of discriminator
BEGAN	WGAN and EBGAN merged objectives
MAGAN	Dynamic margin on hinge loss from EBGAN

Pros and cons

Pros:

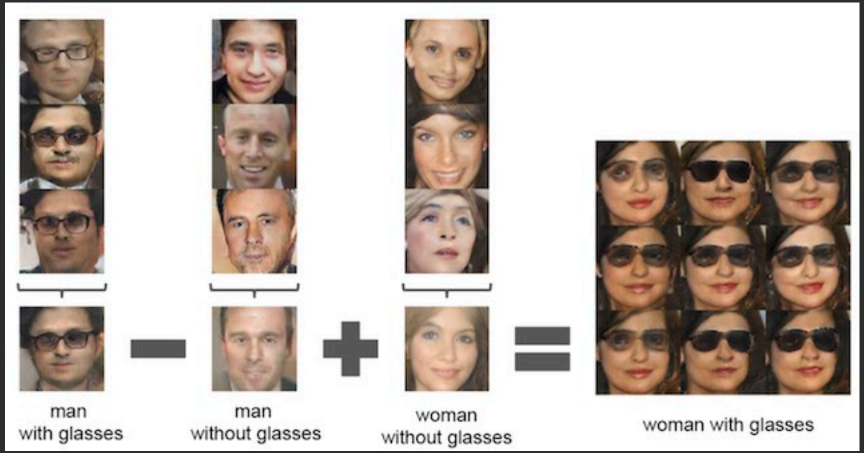
- Directed graph, hence backpropagation is used (easily implementable)
- Generator not updated from data examples (in a sense) since there is limited information about the data in the gradient descent scheme for it
- Short runtimes to generate samples compared to other models (video generation)

Cons:

- No explicit representation of p_g, we only get sample generation
- Difficult to train: Cost function choice not obvious, G and D must be well balanced during training, you must be careful that the generator not map too many points of latent space to data points

"Pros and cons of GAN Evaluation Measures", A.Borji 2018

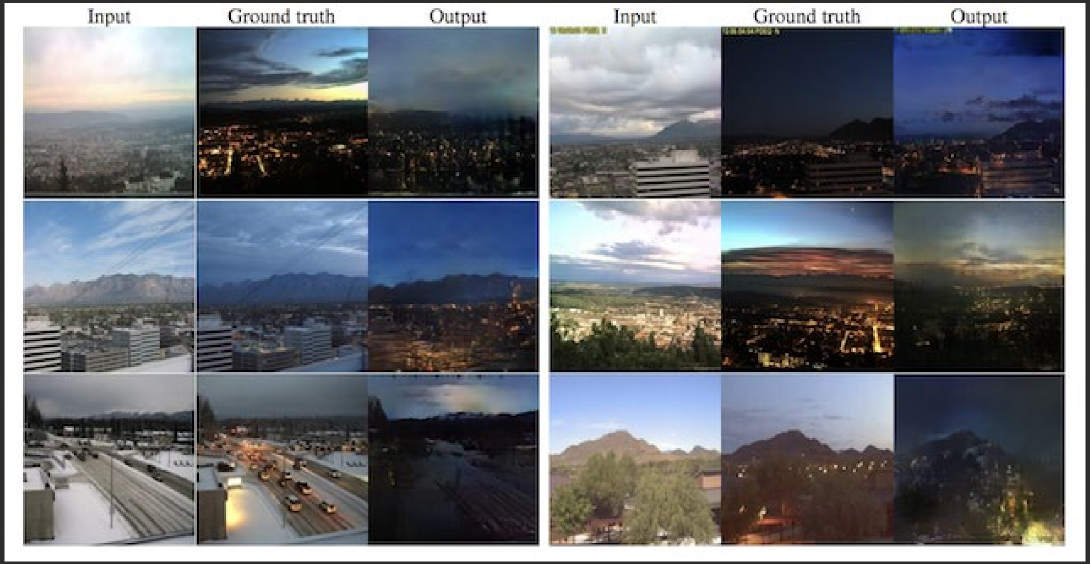
Generate Examples for Datasets



"Generative Adverserial Networks", Goodfellow, et al. 2014

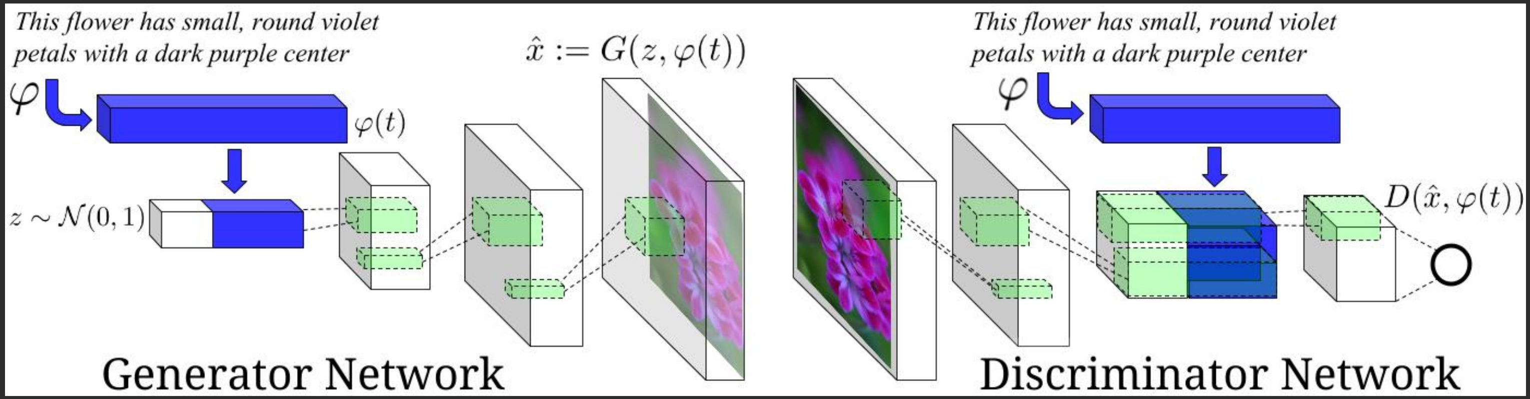
"Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks" by Alec Radford, et al. 2015

Image-to-image translation



"Image-to-Image Translation with Conditional Adversarial Networks", Phillip Isola, et al. 2016
"Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks" Jun-Yan Zhu 2017

Text-to-image generation


This flower has small, round violet petals with a dark purple center
$\hat{x} := G(z, \varphi(t))$
This flower has small, round violet petals with a dark purple center
φ
$\varphi(t)$
φ
$z \sim \mathcal{N}(0,1)$
$D(\hat{x}, \varphi(t))$
Generator Network
Discriminator Network

"Generative Adversarial Text to Image Synthesis", Reed, et al. 2016

Curated list of applications: https://github.com/nashory/gans-awesome-applications
Interesting literature: Attack vs Defence
Medical application: https://arxiv.org/pdf/1706.02633.pdf