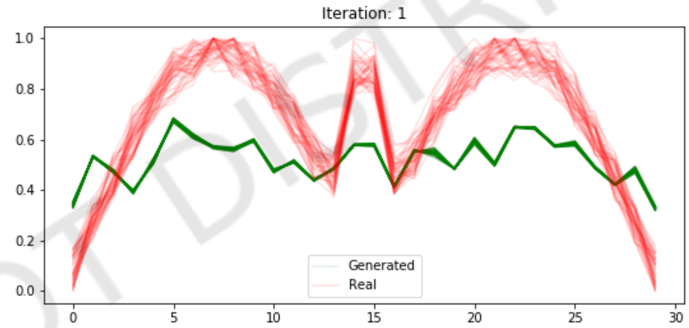
kptan86@gmail.com

## Generative adversarial network

**Generative adversarial networks (GAN)** are models used to **generate sample data of interest** (rather than discriminative purposes such as classification / scoring).




## Generative adversarial network

It is based on the idea* that to **training generative models using discriminatory function as supervisor**.

The system has two actors:

- A "*generator*" function
  - with trainable parameters to generate sample data mimicry to training data
- A "*discriminator*" function
  - capable of measuring the difference between generated data and training data
  - usually also with trainable parameters

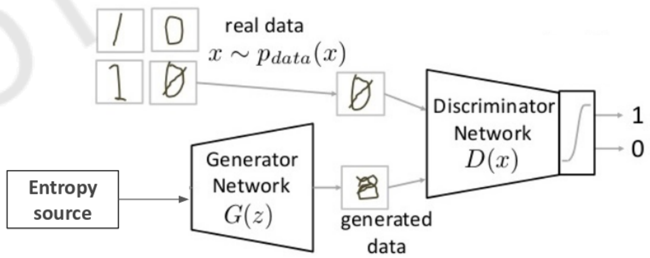
(* special note: The idea is neither novel nor restricted to neural networks.)

kptan86@gmail.com

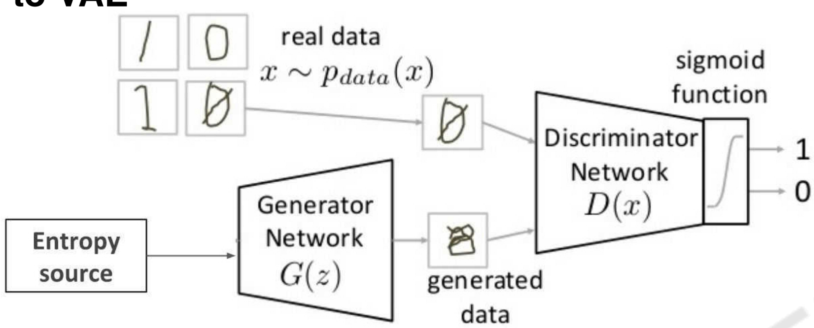
# Generative adversarial network

## Steps

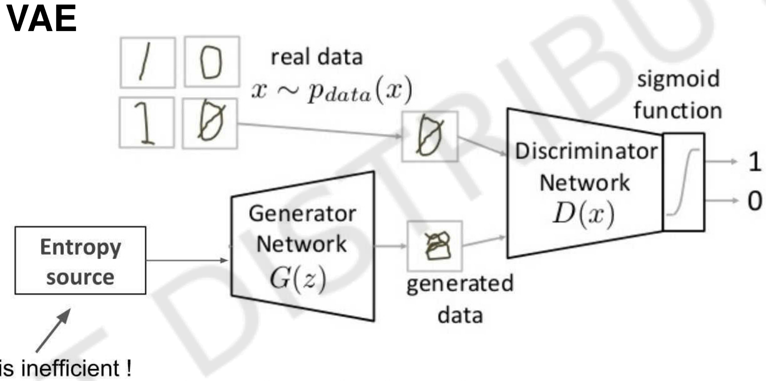
- *Generator* draws (random) inputs from an "entropy source" (e.g. a random number generator) and generates output samples from it
- *Discriminator* computes an error measure (e.g. binary cross entropy or softmax loss) for the each generated output sample against real data
- The errors (dissimilarity between generated and real data) are back-propagated through the *generator* to fit the parameters.
- (The generated output samples are used, together with the real data, to improve the *discriminator*)
- Repeat until the error is minimized.




# From GAN to VAE



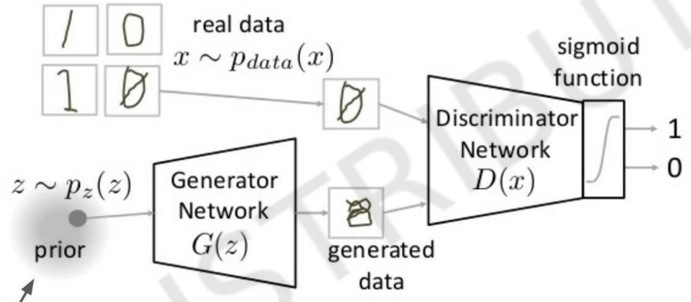

# From GAN to VAE




This is inefficient !
Randomly drawn sample usually could not generate sufficiently 'difficult' data for Discriminator network to differential

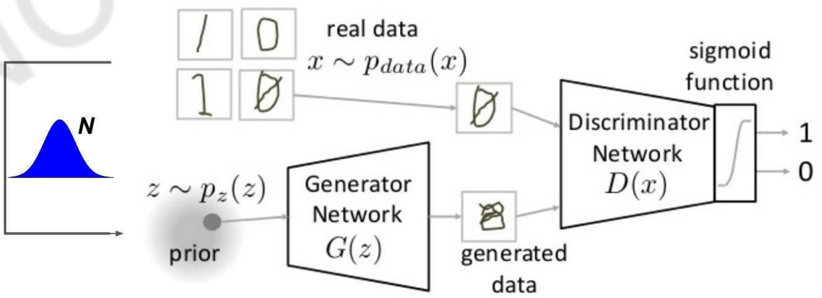
kptan86@gmail.com

## From GAN to VAE




Draw samples from distribution with *some* prior knowledge is much more efficient

## From GAN to VAE




Prior distribution can be learned from training set data

- Prior(s) should be easily computable, e.g. Gaussian form

## Variational (Bayesian) Inference formulation

Sampling $X$ over latent variable $Z$

$$P(X) = \int P(X|z)P(z)dz$$

Use distribution $Q$ to approximate $P$

$$Q(z|X) \sim P(z|X)$$

"Difference" between $P$ and $Q$ can be measured with K.L. divergence:

$$\mathrm{KL}\,(Q(z|X)||P(z|X)) = E_{z\sim Q}\left[\log Q(z|X) - \log P(z|X)\right]$$

(applying Bayes' rule)

$$= E_{z\sim Q}\left[\log(Q|X) - \log P(X|z) - \log P(z)\right] + \log P(X)$$

Rearranging gives

$$\log P(X) - \mathrm{KL}\,(Q(z|X)||P(z|X)) = E_{z\sim Q}\left[\log(Q|X) - \log P(X|z) - \log P(z)\right]$$

$$= E_{z\sim Q}\left[\log P(X|z)\right] - \mathrm{KL}\,(Q(z|X)||P(z))$$

kptan86@gmail.com

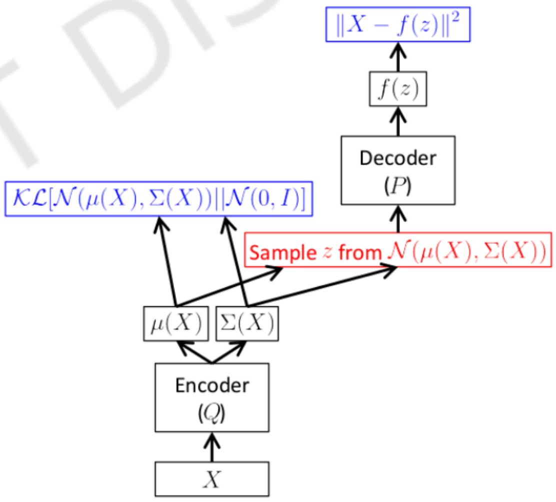
### Analytic solution for Gaussian case

if $P(z) \sim N(0, I)$ and $Q(z|X) \sim N(z; \mu, \sigma^2)$

KL has an analytic solution:

$$\mathrm{KL}\left(Q(z|X)||P(z)\right) = -\frac{1}{2}\sum_{j=1}^{J}\left(1 + \log\sigma_j^2 - \mu_j^2 - \sigma_j^2\right)$$

## Variational Autoencoder




## Paradigm shift on universal function approximator

Previously

- Any distributions could be approximated with neural network

VAE

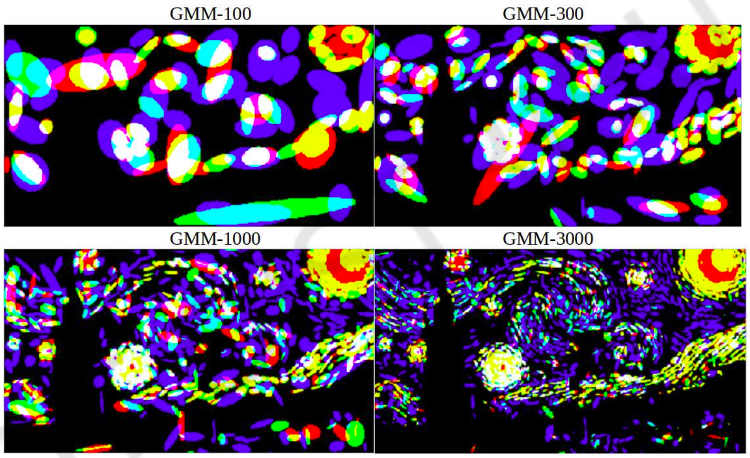
- Any distributions could be approximated with **high-dimensional Gaussian distribution**
- Parameters of the Gaussian distribution can be learned by neural network

Future (?)

- What other distributions can we use ?

kptan86@gmail.com

## High dimensional Gaussian approximator

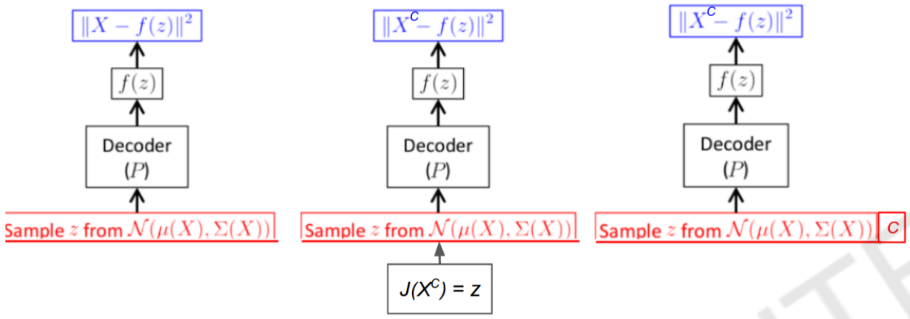



## Conditional Variational Autoencoder (CVAE)

- In the vanilla implementation of VAE, one cannot control the output

**Easy solution**

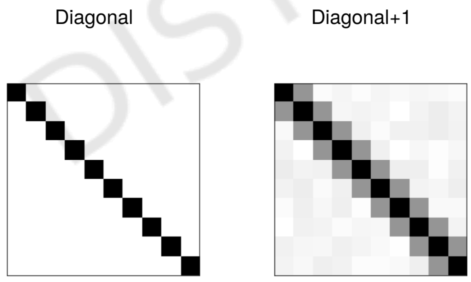
- Recognize that each output corresponds to an input
  - (drawn from an entropy source (e.g. a random number generator))
- Train an encapsulating network to relate between input and output label



kptan86@gmail.com

## (extra) Recurrent latent Gaussian model

- Remember we draw $z$ from $P(z) \sim N(0, I)$
- Assume each latent Gaussian component is independent to one another
- The constraint was introduced for computational purpose
- $N$-dim Gaussian: $N$ means, $N(N+1)/2$ covariances to model
  - at dim=1000, full covariance matrix has 501,500 parameters; diagonal matrix only with 2000 parameters
- Can we model a bit more parameters (in between diagnoal and full) ?




- Make a Gaussian component dependent on its previous component
  - Essentially a *recurrent Gaussian model*



DO NOT DISTRIBUTE