[SKT AI Course: Deep Learning Basics]

Gentle Introduction to Deep Generative Models

Heung-Il Suk

hisuk@korea.ac.kr

http://www.ku-milab.org

Department of Brain and Cognitive Engineering,
Korea University

October 26, 2017

Contents

Generative Models In a Nutshell

Why Generative Model?

"What I cannot create, I do not understand."

—Richard Feynman

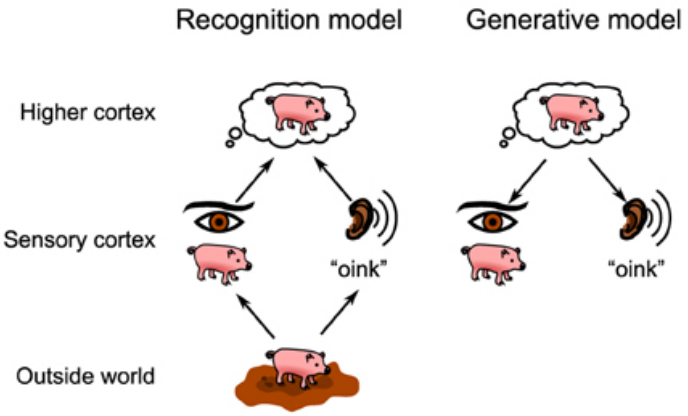
- Excellent test of our ability to use high-dimensional, complicated probability distributions

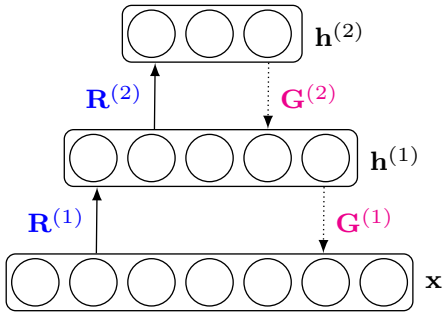
- Missing data imputation or data completion

Deep Generative Models

Helmholtz Machine [Dayan et al., 1995]

- Brain as a constructive or predictive organ that actively generates predictions of its sensory inputs using an internal or generative model
- Perception that can be traced back to Helmholtz's original writings on unconscious inference (Helmholtz, 1866/1962)





$\mathbf{R}$: recognition model, $\mathbf{G}$: generative model

[Sountsov and Miller, *Frontiers in Computational Neuroscience*, 2015]

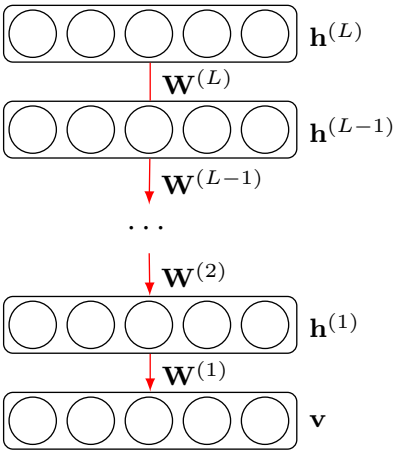
Deep Belief Networks (DBN)

Graphical models that learn to extract a deep hierarchical representation of the training data

- Model the joint distribution between an observed vector $\mathbf{x}$ and the L hidden layers $\mathbf{h}^{(l)}$

$$P\left(\mathbf{x}, \mathbf{h}^{(1)}, \cdots, \mathbf{h}^{(L)}\right) = P\left(\mathbf{h}^{(L-1)}, \mathbf{h}^{(L)}\right) \times \left(\prod_{l=0}^{l-2} P(\mathbf{h}^{(l)}|\mathbf{h}^{(l+1)})\right)$$

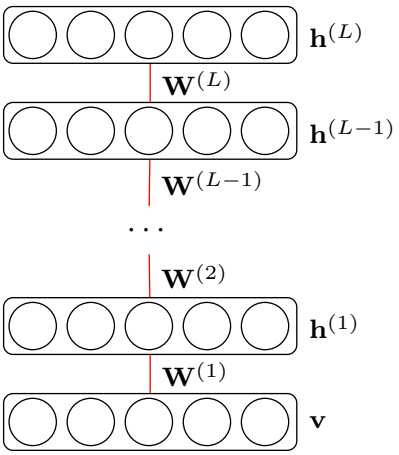
  - $P(\mathbf{h}^{(l-1)}|\mathbf{h}^{(l)})$: a conditional distribution for the visible units conditioned on the hidden units of the RBM at level l
  - $P\left(\mathbf{h}^{(L-1)}, \mathbf{h}^{(L)}\right)$: visible-hidden joint distribution in the top-level RBM



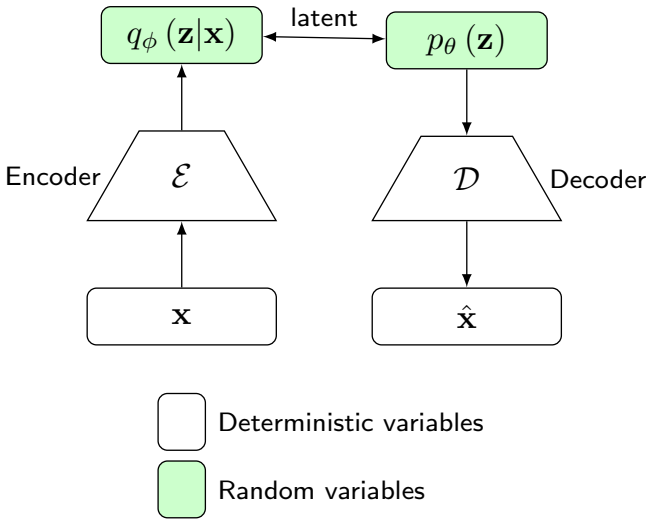
Deep Boltzmann Machine (DBM)

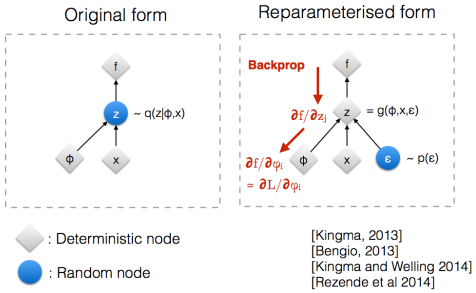
- Markov Random Field (MRF)
  - Undirected connections between layers

⋆ In approximate inference, incorporating top-down feedback
  - Using higher-level knowledge to resolve uncertainty about intermediate-level features
  - Better data-dependent representations; better data-dependent statistics for learning

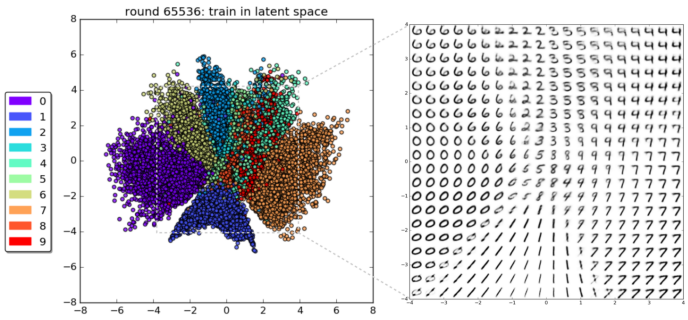


Variational Auto-Encoder (VAE) [Kingma and Welling, 2014]

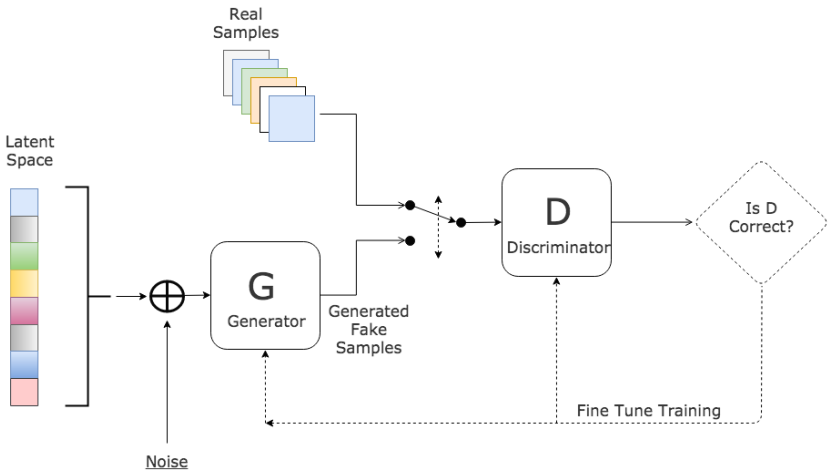




[Kingma, 2013]
[Bengio, 2013]
[Kingma and Welling 2014]
[Rezende et al 2014]



Generative Adversarial Network (GAN) [Goodfellow et al., 2014]

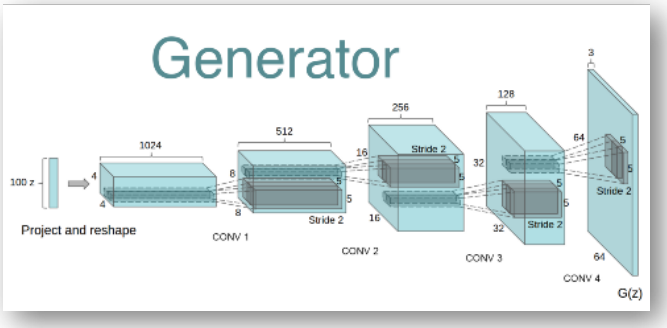


$$\underset{\boldsymbol{\theta}^{(G)}}{\operatorname{argmin}} \max_{\boldsymbol{\theta}^{(D)}} V\left(\boldsymbol{\theta}^{(D)}, \boldsymbol{\theta}^{(G)}\right) = \frac{1}{2}\mathbb{E}_{\mathbf{x}\sim p_{data}}\left[\log D\left(\mathbf{x}\right)\right] + \frac{1}{2}\mathbb{E}_{\mathbf{z}}\left[\log\left(1 - D\left(G\left(\mathbf{z}\right)\right)\right)\right]$$

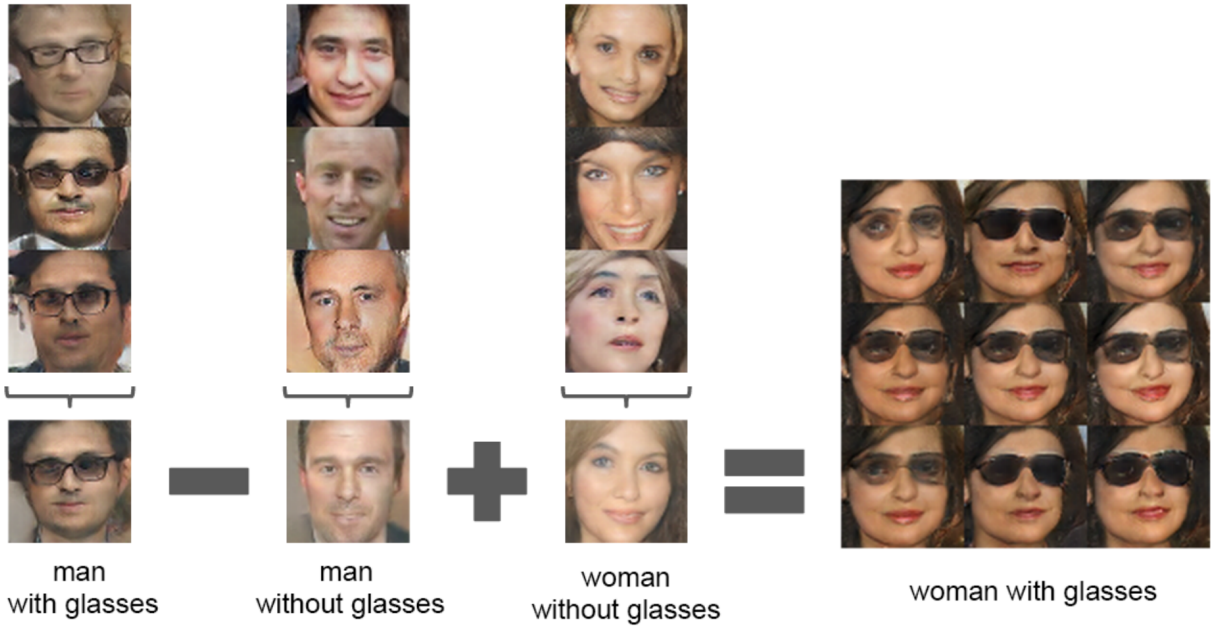
GAN's convergence problem

- Don't know when to stop training (no convergence)
  - *i.e.*, the loss function doesn't correlate with image quality
- Need to be constantly looking at the samples to tell whether your model is training correctly.
- Don't have a numerical value that tells you how well you are tuning the parameters.

Deep Convolutional GAN (DCGAN) [Radford *et al.*, 2015]



- Focused on improving the architecture of the original vanilla GAN
- Empirical findings
  - Batch normalization is a must in both networks.
  - Fully hidden connected layers are not a good idea.
  - Avoid pooling, simply stride your convolutions!
  - ReLU activations are your friend (almost always).

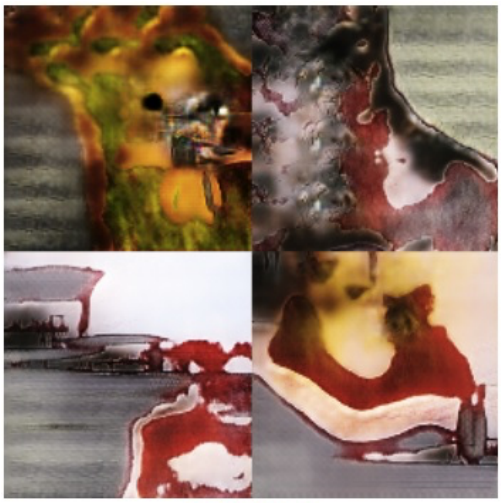


vector arithmetic on the latent space

Improved DCGAN [Salimans *et al.*, 2016]

to be better at generating high resolution images

- **Feature matching**: $\mathbb{G}$ to generate data that matches the statistics of the real data; $\mathbb{D}$ is only used to specify which are the statistics worth matching
- **Historical averaging**: when updating the parameters, also take into account their past values
- **One-sided label smoothing**: simply make your $\mathbb{D}$ target output from [0=fake image, 1=real image] to [0=fake image, 0.9=real image]
- **Virtual batch normalization**: avoid dependency of data on the same batch by using statistics collected on a reference batch. It is computationally expensive, so its only used on $\mathbb{G}$.

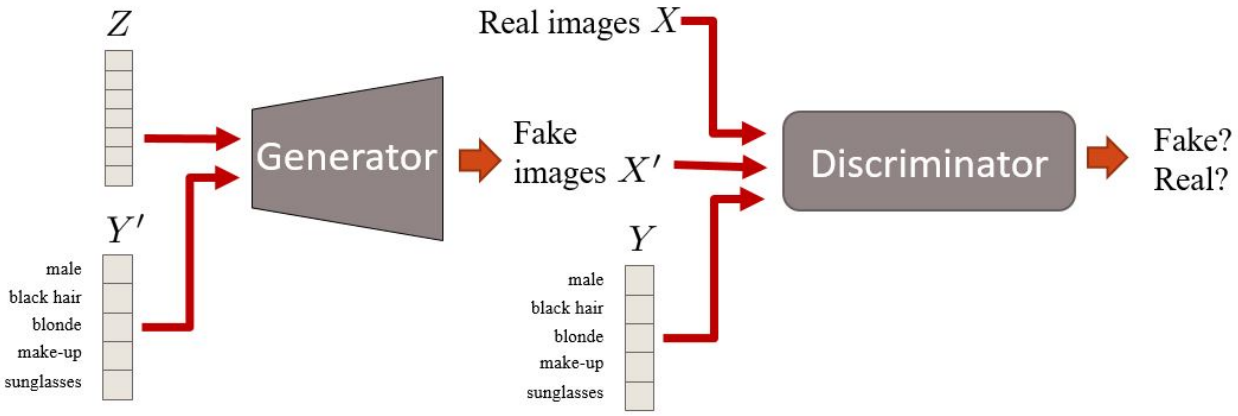


DCGAN

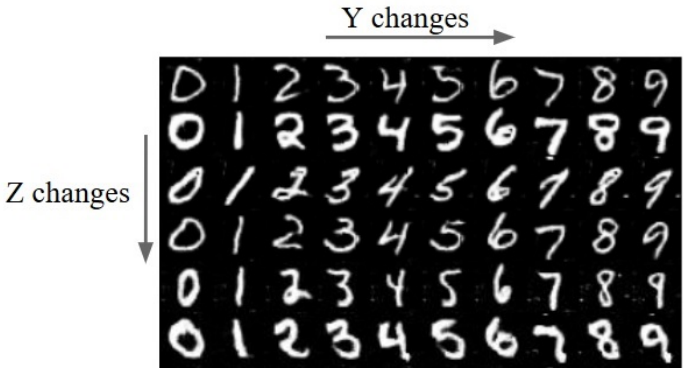
Improved DCGAN

Conditional GAN [Mirza *et al.*, 2014]

- Using extra label information
- Conditional information Y that describes some aspect of the data
  - if we are dealing with faces, Y could describe attributes such as hair color or gender. Then, this attribute information is inserted in both the generator and the discriminator

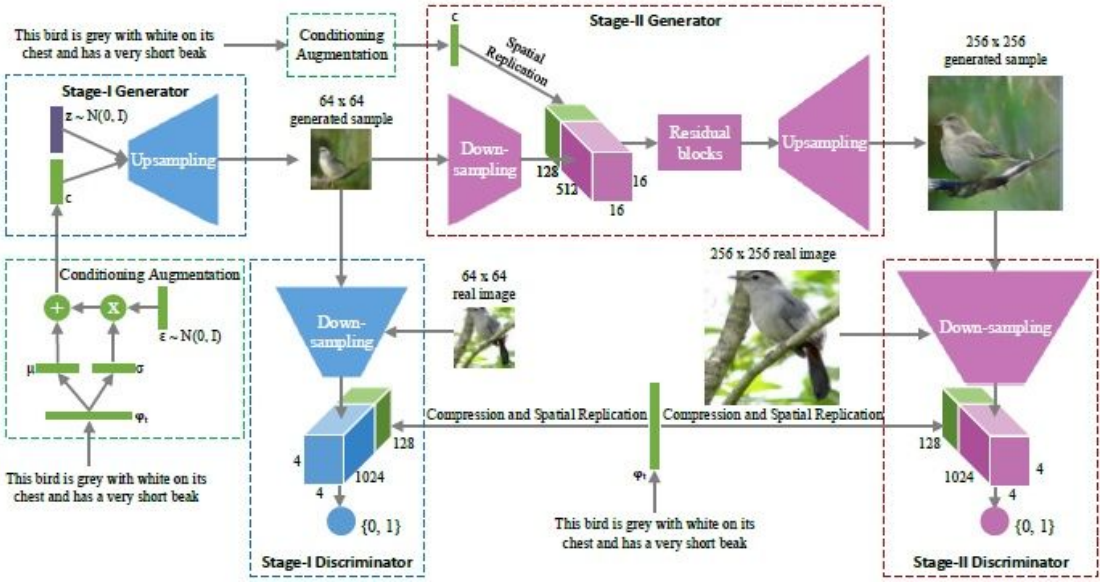


- Z and Y: will encode different information
- *e.g.*, Suppose Y encodes the digit of a hand-written number (from 0 to 9). Then, Z would encode all the other variations that are not encoded in Y. That could be, for example, the style of the number (size, weight, rotation, etc).



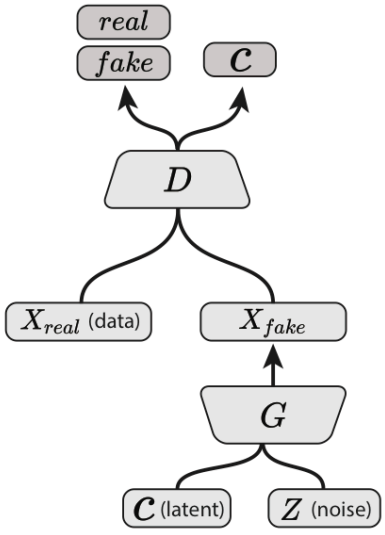
Stacked Generative Adversarial Networks (StackGAN) [Zhang *et al.*, 2016]

- Focusing on improving the quality of the image by using 2 GANs at the same time
  - Stage-I is used to get a low-resolution image containing the "general" idea of the image.
  - Stage-II refines Stage-I's images with more details and higher resolution.



InfoGAN: Interpretable Representation Learning by Information Maximizing GAN [Chen *et al.*, 2016]

- Encode meaningful image features in part of the noise vector Z in an unsupervised manner
  - *e.g.*, rotation of a digit
- Take Z vector and split it into two parts: C and Z
  - C will encode the semantic features of the data distribution.
  - Z will encode all the unstructured noise of this distribution.
- An information-theoretic regularization which ensures a high mutual information between C and the generator distribution



Varying c_1 on InfoGAN (Digit type)

Varying c_2 from -2 to 2 on InfoGAN (Rotation)

c_1 encodes the digit class; c_2 encodes the rotation

- These encodings only work with fairly simple datasets, such as MNIST digits.
- Moreover, you still need to "hand-craft" each position of C.

Wasserstein GANs [Chen *et al.*, 2016]

- GANs can be interpreted to minimize the Jensen-Shannon divergence, which is 0 if the real and fake distribution don't overlap (which is usually the case).
- So, instead of minimizing the Jensen-Shannon divergence, use the **Wasserstein distance**, which describes the distance between the "points" from one distribution to the other.
  - Correlate with image quality and enables convergence
  - Training stability improves and is not as dependent on the architecture

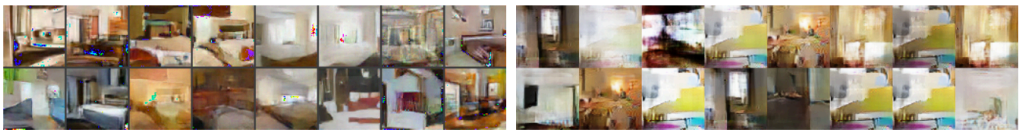
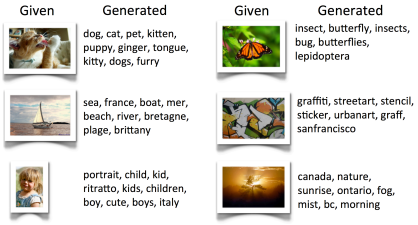
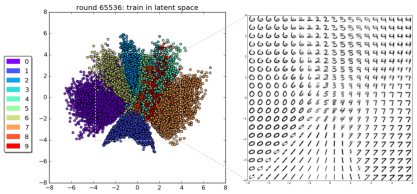


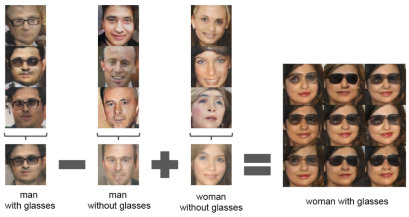
Figure 7: Algorithms trained with an MLP generator with 4 layers and 512 units with ReLU nonlinearities. The number of parameters is similar to that of a DCGAN, but it lacks a strong inductive bias for image generation. Left: WGAN algorithm. Right: standard GAN formulation. The WGAN method still was able to produce samples, lower quality than the DCGAN, and of higher quality than the MLP of the standard GAN. Note the significant degree of mode collapse in the GAN MLP.

Summary

- Why Generative Models?
- Helmholtz Machine
- Deep Belief Networks (DBN)
- Deep Boltzmann Machine (DBM)
- Variational Auto-Encoder (VAE)
- Generative Adversarial Network (GAN)







Thank you for your attention!!!

(Q & A)

hisuk (AT) korea.ac.kr

http://www.ku-milab.org

Machine Intelligence Lab