Deep Generative Models

Shenlong Wang

Overview

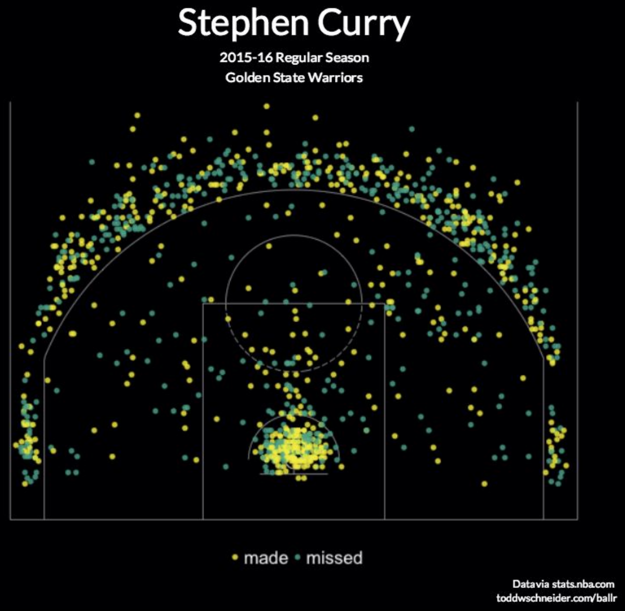
- Why unsupervised learning?
- Old-school unsupervised learning
  - PCA, Auto-encoder, KDE, GMM
- Deep generative models
  - VAEs, GANs

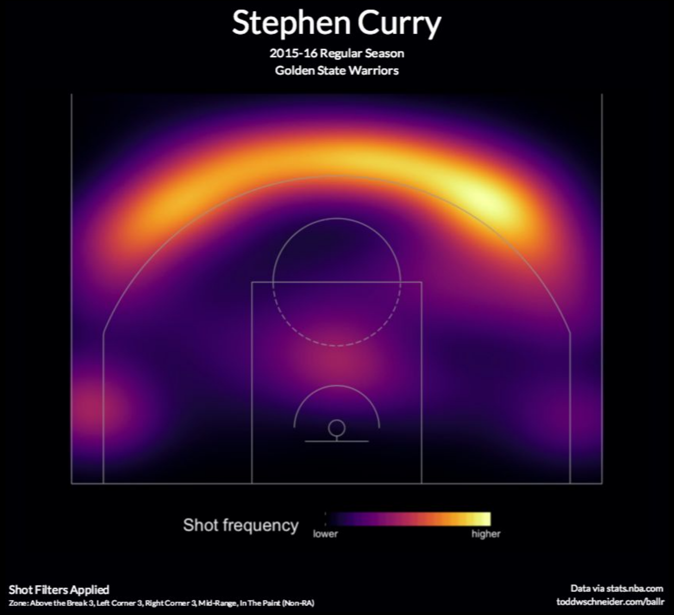
Unsupervised Learning

- No labels are provided during training
- General objective: inferring a function to describe hidden structure from unlabeled data
  - Density estimation (continuous probability)
  - Clustering (discrete labels)
  - Feature learning / representation learning (continuous vectors)
  - Dimension reduction (lower-dimensional representation)
  - etc.

Why Unsupervised Learning?

- Density estimation: estimate the probability density function p(x) of a random variable x, given a bunch of observations {X1, X2, ...}



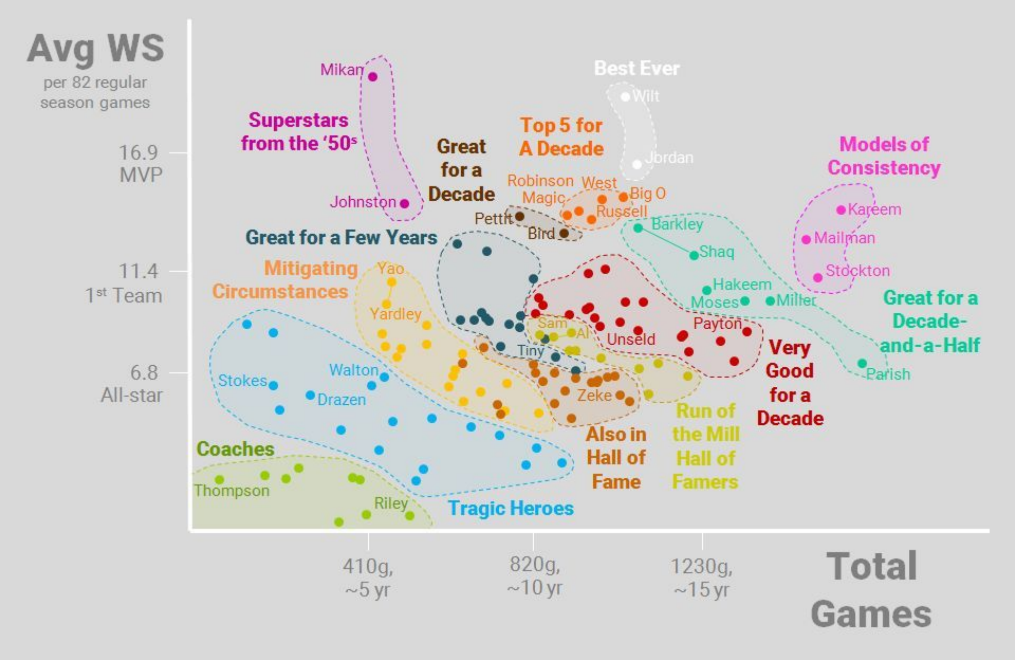


2D density estimation of Stephen Curry's shooting position

Credit: BallR

Why Unsupervised Learning?

- Clustering: grouping a set of input {X1, X2, ...} in such a way that objects in the same group (called a cluster) are more similar

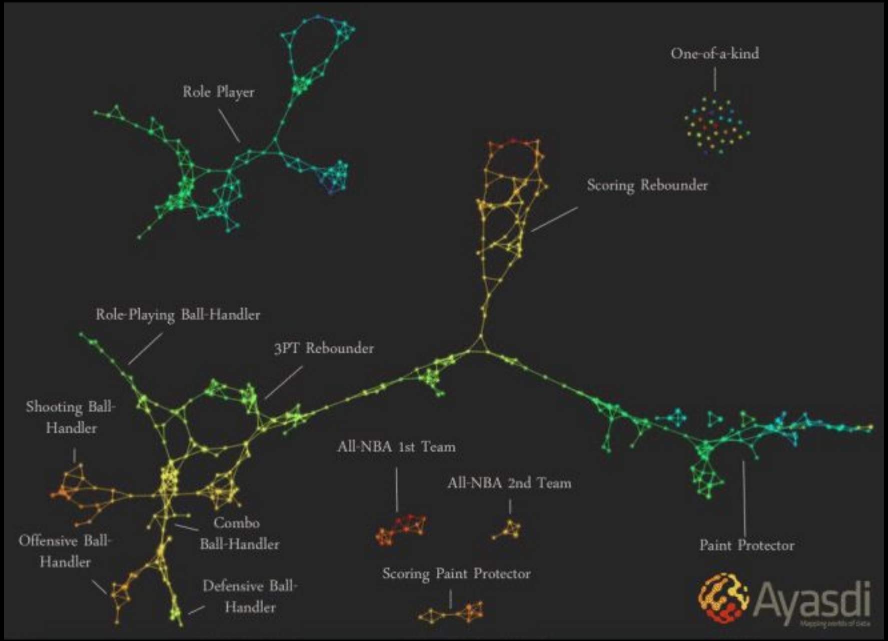


Clustering analysis of Hall-of-fame players in NBA

Credit: BallR

Why Unsupervised Learning?

- Feature learning: a transformation of raw data input to a representation that can be effectively exploited in machine learning tasks

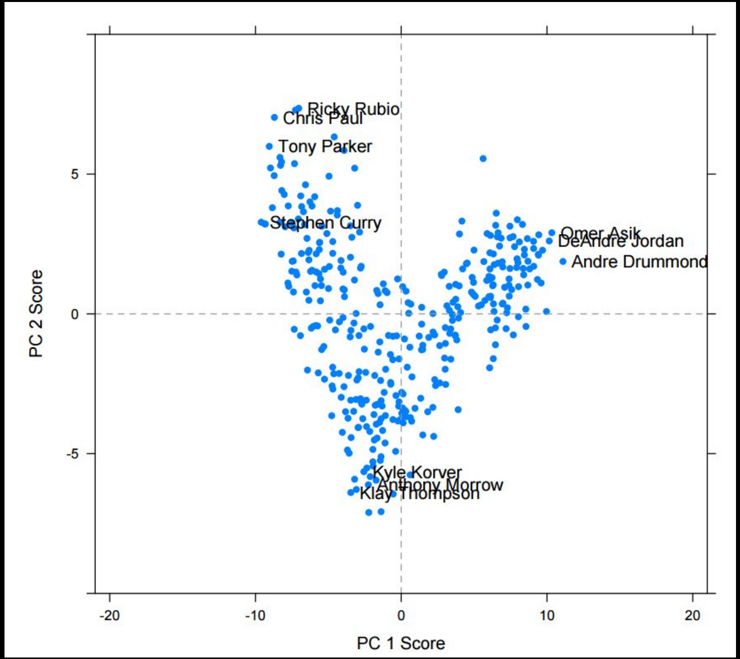


2D topological visualization given the input how similar players are with regard to points, rebounds, assists, steals, rebounds, blocks, turnovers and fouls

Credit: Ayasdi

Why Unsupervised Learning?

- Dimension reduction: reducing the number of random variables under consideration, via obtaining a set of principal variables



Principle component analysis over players trajectory data

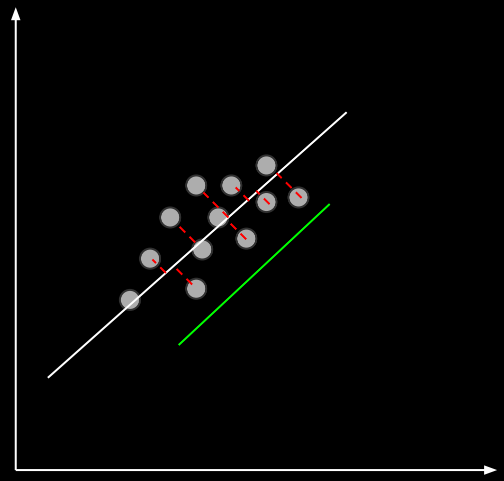
Credit: Bruce, Arxiv 2016

Principle Component Analysis (PCA)

An algorithm that conducts dimension reduction

Intuition:

- Finds the lower-dimension projection that minimizes reconstruction error
- Keep the most information (maximize variance)



See more details in Raquel’s CSC411 slides:
http://www.cs.toronto.edu/~urtasun/courses/CSC411_Fall16/14_pca.pdf

Principle Component Analysis (PCA)

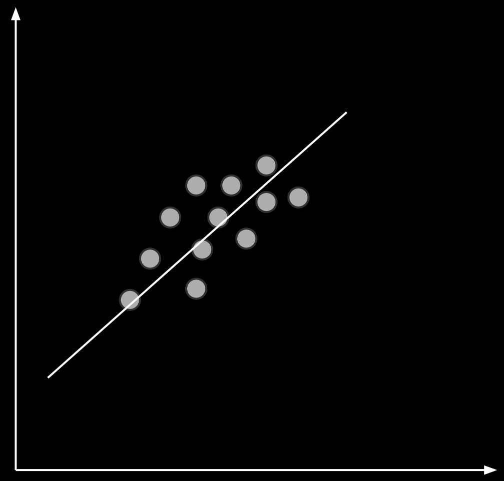
An algorithm that conducts dimension reduction

Intuition:

- Finds the lower-dimension projection that minimizes reconstruction error
- Keep the most information (maximize variance)

Algorithm:

- Conduct eigen decomposition
- Find K-largest eigenvectors
- Linear projection with the matrix composed of K eigenvectors



See more details in Raquel's CSC411 slides:
http://www.cs.toronto.edu/~urtasun/courses/CSC411_Fall16/14_pca.pdf

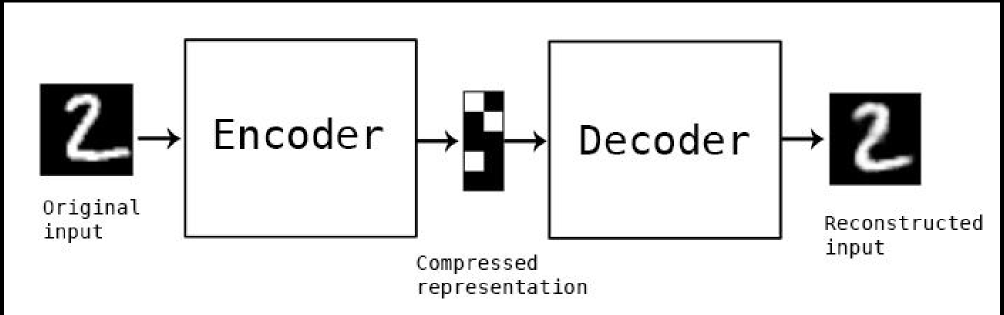
Auto-encoder

A neural network that the output is the input itself.

Intuition:

- A good representation should keep the information well (reconstruction error)
- Deep + nonlinearity might help enhance the representation power

$$\min_{\mathbf{w}_1, \mathbf{w}_2} \|\mathbf{x}_i - g(f(\mathbf{x}_i; \mathbf{w}_1); \mathbf{w}_2)\|_2^2$$



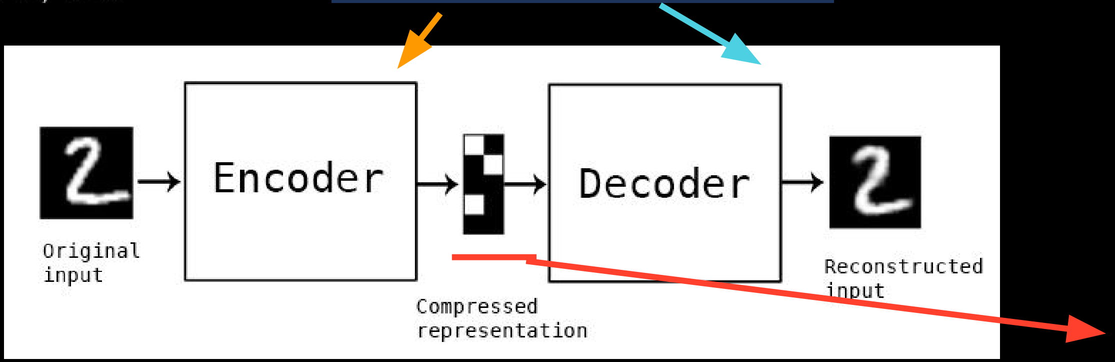
Auto-encoder

A neural network that the output is the input itself.

Intuition:

- A good representation should keep the information well (reconstruction error)
- Deep + nonlinearity might help enhance the representation power

$$\min_{\mathbf{w}_1,\mathbf{w}_2} \|\mathbf{x}_i - g(f(\mathbf{x}_i;\mathbf{w}_1);\mathbf{w}_2)\|_2^2$$

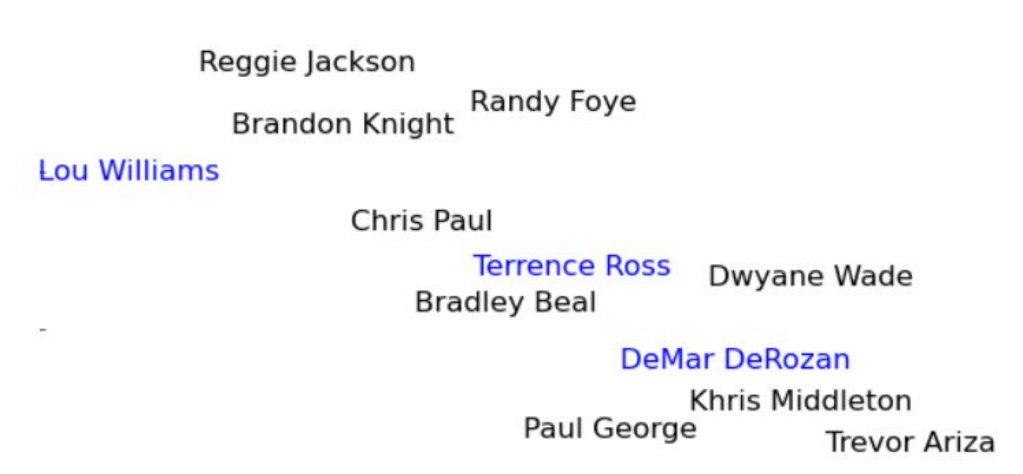


Learnt representation

Credit: LeCun

Auto-encoder

A neural network that the output is the input itself.



10-dimensional Auto-encoder feature embedding based on players shooting tendency

Credit: Wang et al. 2016 Sloan Sports Conference

Kernel Density Estimation (KDE)

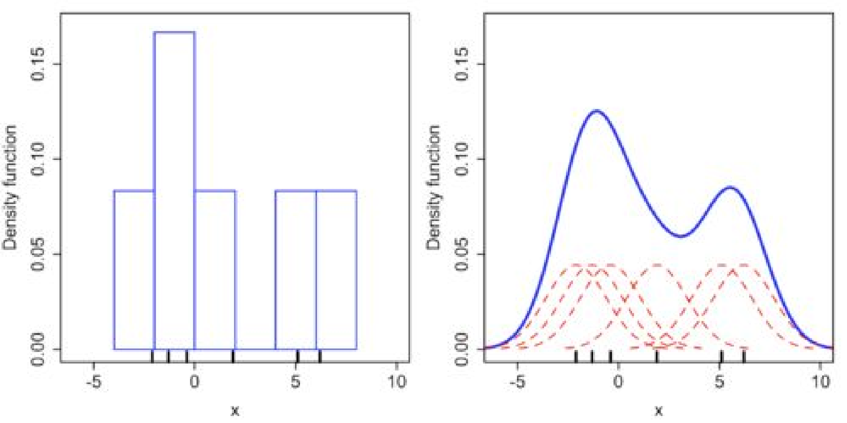
A nonparametric way to estimate the probability density function of a random variable

Intuition:

- Point with more neighbouring samples have higher density
- Smoothed histogram, centered at data point

$$f(\mathbf{x}) = \frac{1}{N}\sum_i \frac{1}{h_i} \underline{K(\frac{\mathbf{x}-\mathbf{x}_i}{h_i})}$$

Kernel function, measures the similarity



Credit: Wikipedia

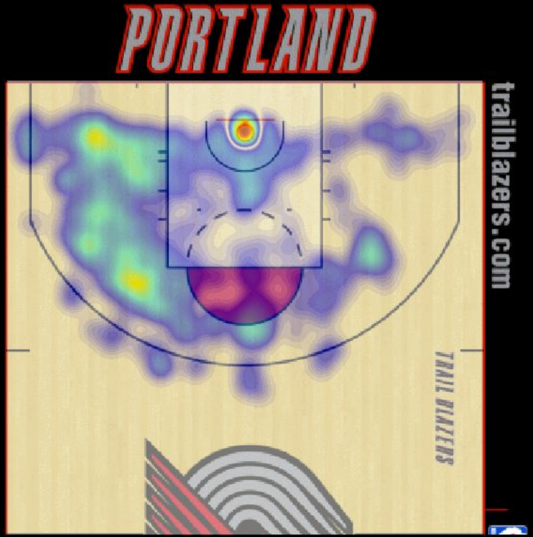
Kernel Density Estimation (KDE)

A nonparametric way to estimate the probability density function of a random variable

Applications:

- Visualization
- Sampling

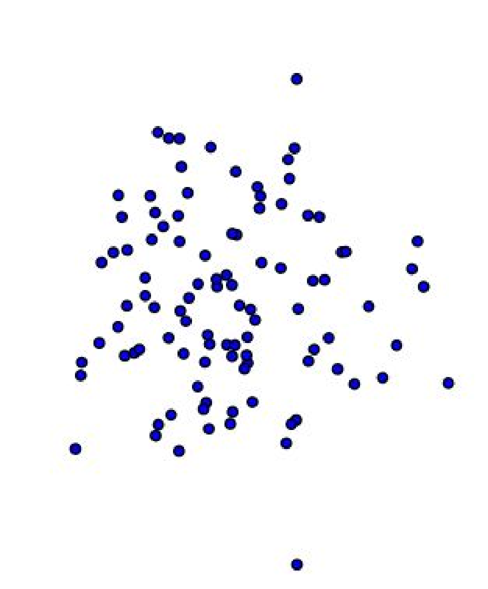
$$f(\mathbf{x}) = \frac{1}{N}\sum_i \frac{1}{h_i} K(\frac{\mathbf{x}-\mathbf{x}_i}{h_i})$$



Shooting heat map of Lamarcus Aldridge 2015-2016. Credit: Squared Statistics

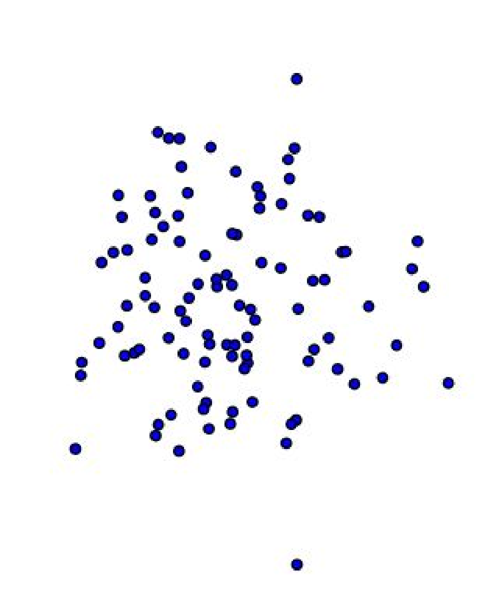
Generative models

Task: generate new samples follows the same probabilistic distribution of a given a training dataset

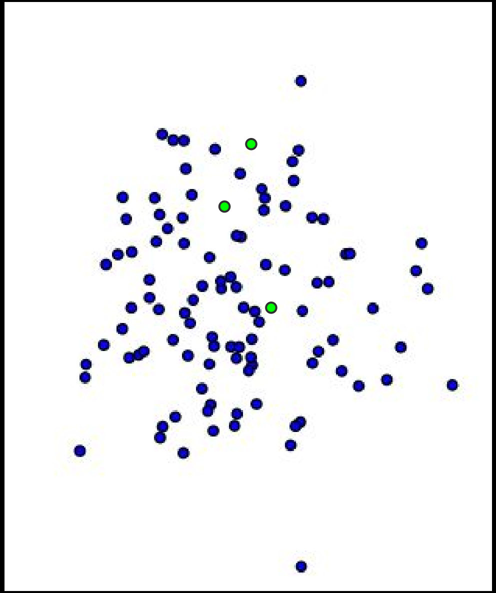


Generative models

Task: generate new samples follows the same probabilistic distribution of a given a training dataset

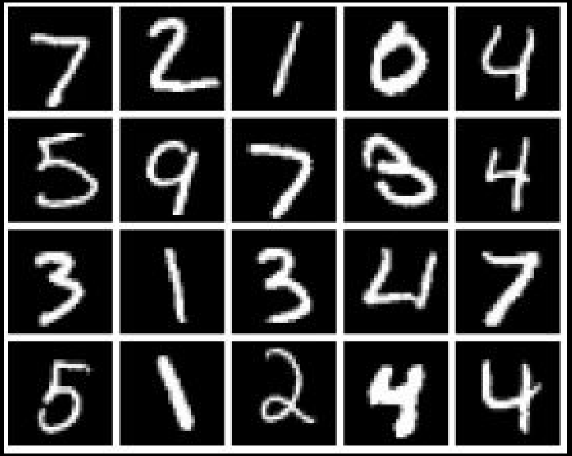


→ $Z \sim \mathcal{N}(0, 1)$ →



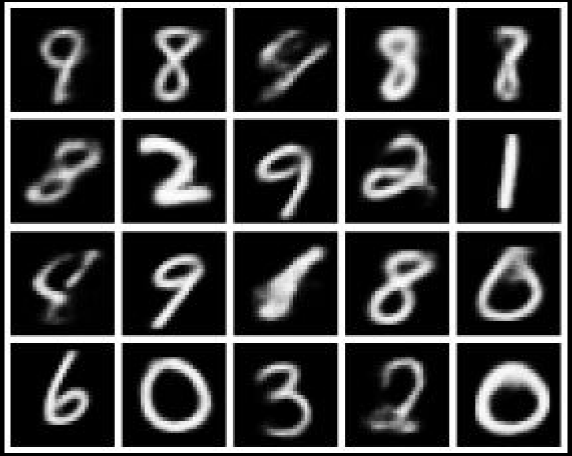
Generative models

Task: generate new samples follows the same probabilistic distribution of a given a training dataset



Training samples

→ $p(x) = ?$ →



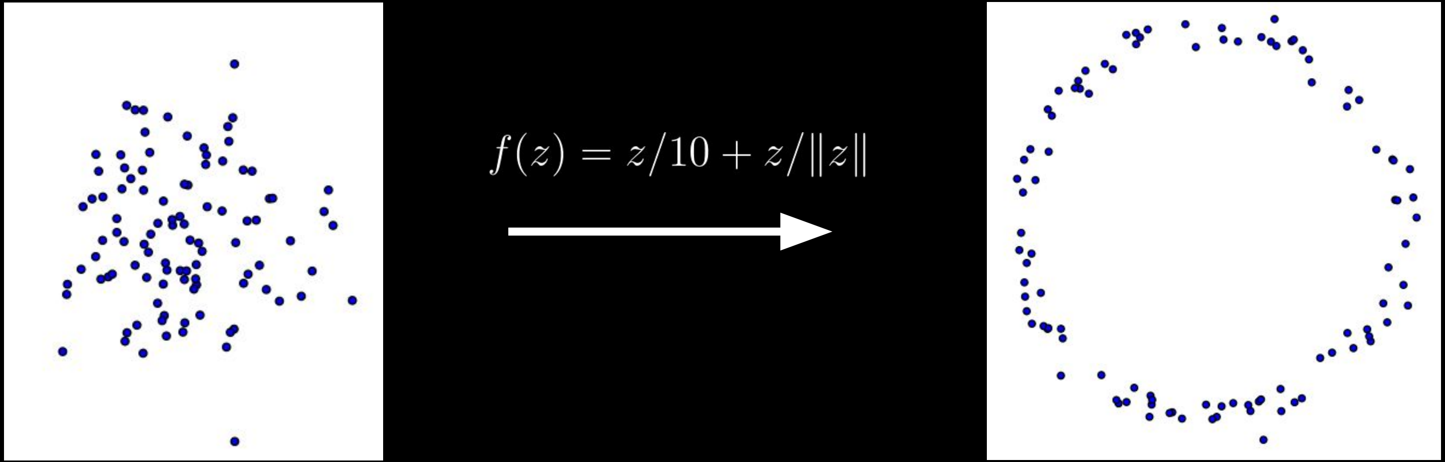
Generated samples

Credit: Kingma

Note: sometimes it’s fine if we cannot estimate the explicit form of p(x), since it might be over complicated

Variational Auto-encoder (VAE)

Intuition: given a bunch of random variables that can be sampled easily, we can generate random samples following other distributions, through a complicated non-linear mapping x = f(z)

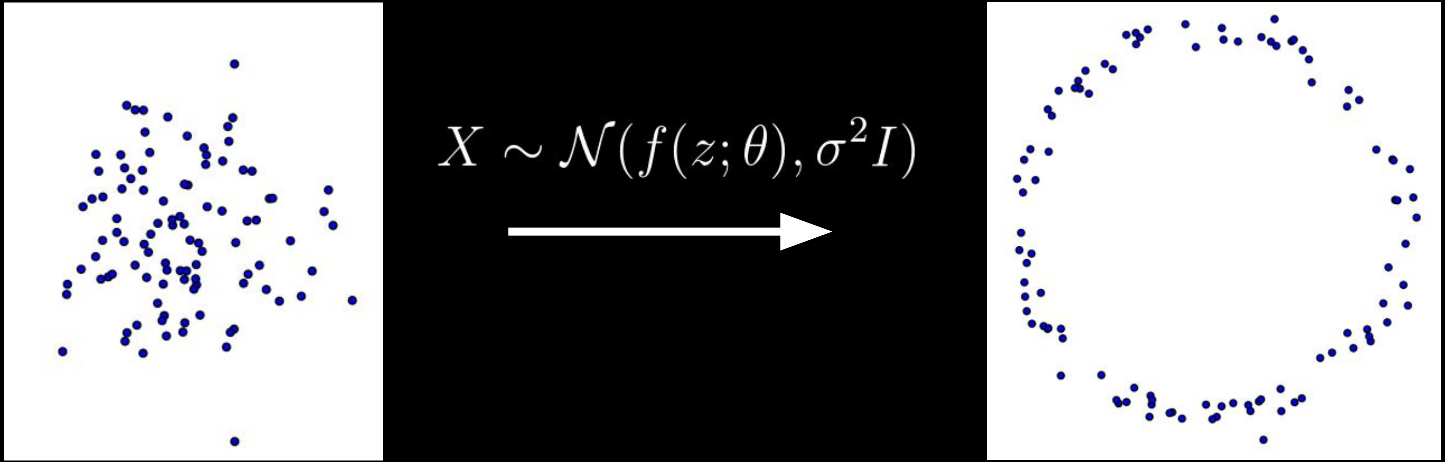


$$f(z) = z/10 + z/\|z\|$$

Image Credit: Doersch 2016

Variational Auto-encoder (VAE)

Intuition: given a bunch of random variables that can be sampled easily, we can generate some new random samples through a complicated non-linear mapping x = f(z)



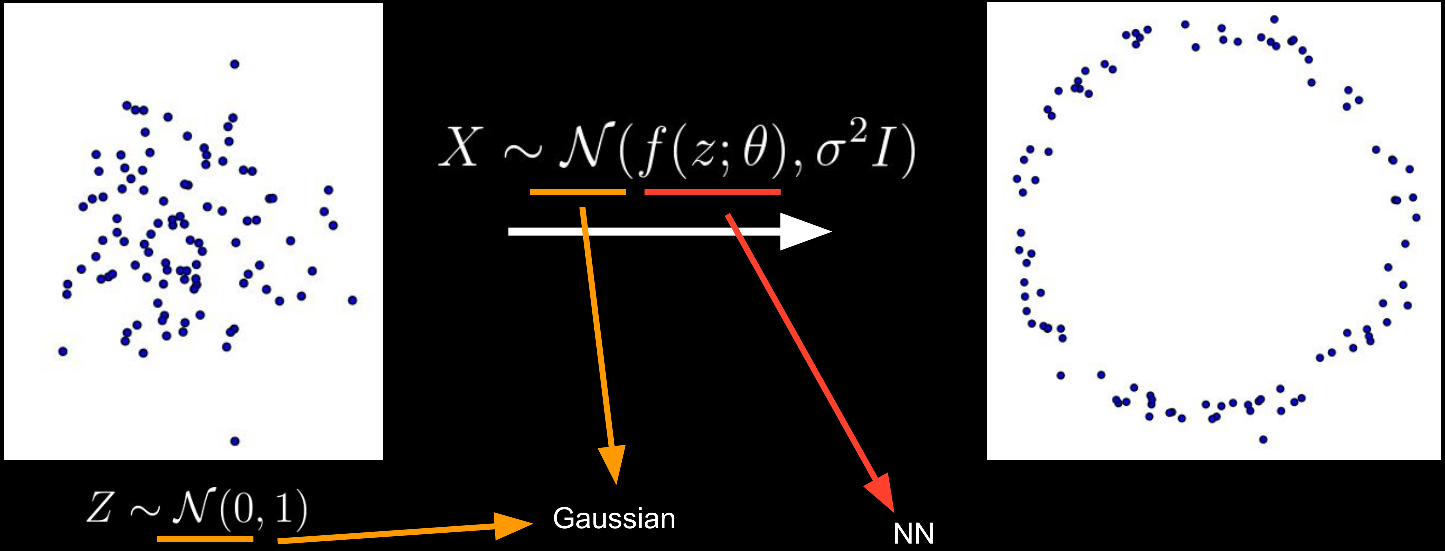
$$X \sim \mathcal{N}(f(z;\theta), \sigma^2 I)$$

$$Z \sim \mathcal{N}(0, 1)$$

Image Credit: Doersch 2016

Variational Auto-encoder (VAE)

Intuition: given a bunch of random variables, we can generate some new random samples through a complicated non-linear mapping x = f(z)



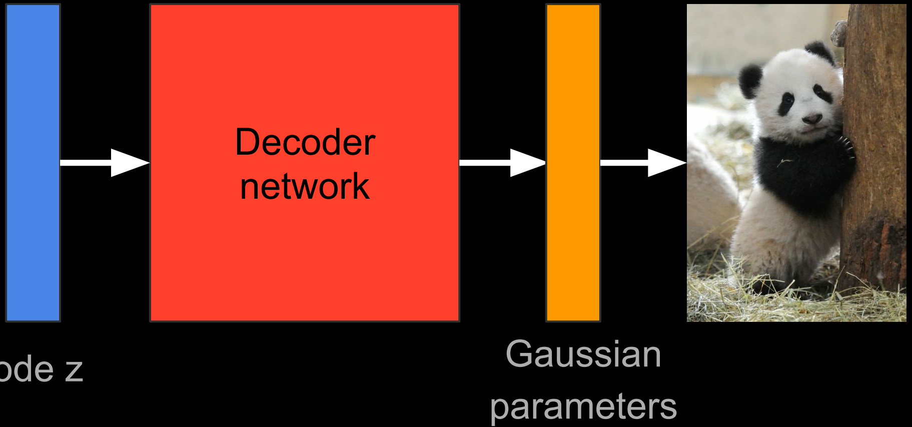
$$X \sim \underline{\mathcal{N}}(\underline{f(z;\theta)}, \sigma^2 I)$$

$$Z \sim \underline{\mathcal{N}}(0, 1)$$

Image Credit: Doersch 2016

Variational Auto-encoder (VAE)

You can consider it as a decoder!



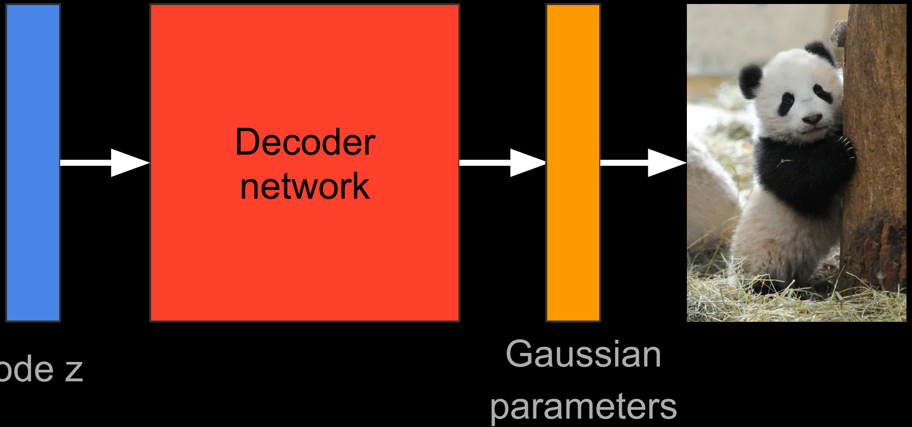
Decoder network

Code z

Gaussian parameters

Variational Auto-encoder (VAE)

How do we learn the parameters?



Variational Auto-encoder (VAE)

Graphical model

$$p(\mathbf{x}) = \int p_\theta(\mathbf{x}|\mathbf{z})p_\theta(\mathbf{z})d\mathbf{z}$$

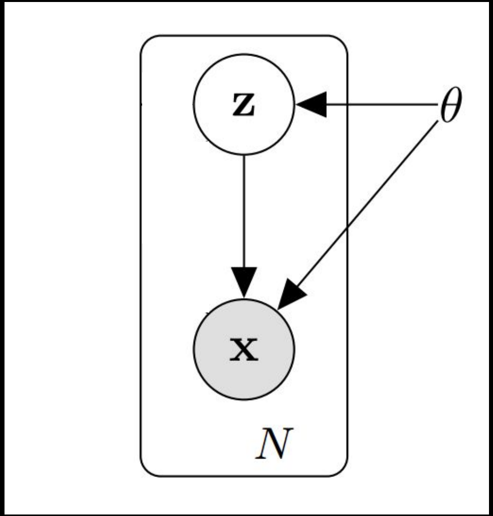


Image Credit: Doersch 2016

Variational Auto-encoder (VAE)

Learning objective: maximize the log-probability

$$\max_{\theta} \sum_{i} \log p_{\theta}(\mathbf{x}_i)$$

$$p_{\theta}(\mathbf{x}) = \underline{\int p_{\theta}(\mathbf{x}|\mathbf{z})p_{\theta}(\mathbf{z})d\mathbf{z}}$$

many sampled z will have a close-to-zero p(x|z)

Quiz: Why not doing this? $\log p_{\theta}(\mathbf{x}) \approx \log \frac{1}{N} \sum_{j} p_{\theta}(\mathbf{x}|\mathbf{z}_j)$

Image Credit: Doersch 2016

Variational Auto-encoder (VAE)

Learning objective: maximize variational lower-bound

$$\log p_\theta(\mathbf{x}_i) \geq \underline{\mathbb{E}_{q(\mathbf{z})}[\log p_\theta(\mathbf{x}_i|\mathbf{z})] - KL[q(\mathbf{z})||p_\theta(\mathbf{z})]}$$

Variational lower-bound

Quiz: How to choose a good proposal distribution?

Proposal distribution

Variational Auto-encoder (VAE)

Learning objective: maximize variational lower-bound

$$\log p_\theta(\mathbf{x}_i) \geq \underbrace{\mathbb{E}_{q(\mathbf{z})}[\log p_\theta(\mathbf{x}_i|\mathbf{z})] - KL[q(\mathbf{z})||p_\theta(\mathbf{z})]}_{\text{Variational lower-bound}}$$

Proposal distribution

Quiz: How to choose a good proposal distribution?

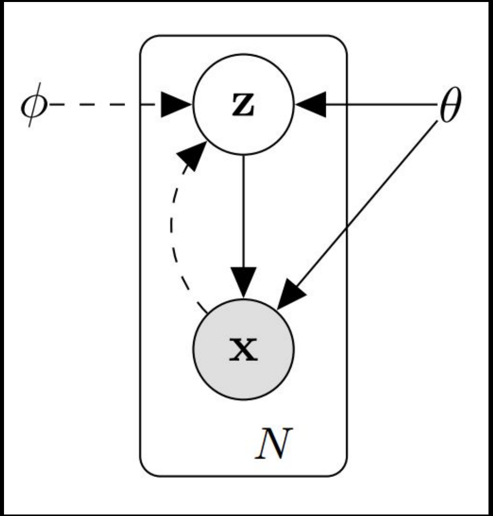
- Easy to sample
- Differentiable
- Given a training sample X, the sampled z is likely to have a non-zero p(x|z)

Variational Auto-encoder (VAE)

Learning objective: maximize variational lower-bound

$$\log p_\theta(\mathbf{x}_i) \geq \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x}_i)}[\log p_\theta(\mathbf{x}_i|\mathbf{z})] - KL[q_\phi(\mathbf{z}|\mathbf{x}_i)||p_\theta(\mathbf{z})]$$

Answer: Another neural network + Gaussian to approximate the posterior!



Variational Auto-encoder (VAE)

Learning objective: maximize variational lower-bound

$$\log p_\theta(\mathbf{x}_i) \geq \underline{\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x}_i)}[\log p_\theta(\mathbf{x}_i|\mathbf{z})]} \underline{- KL[q_\phi(\mathbf{z}|\mathbf{x}_i)||p_\theta(\mathbf{z})]}$$

Reconstruction error:

- Training samples have higher probability

Prior:

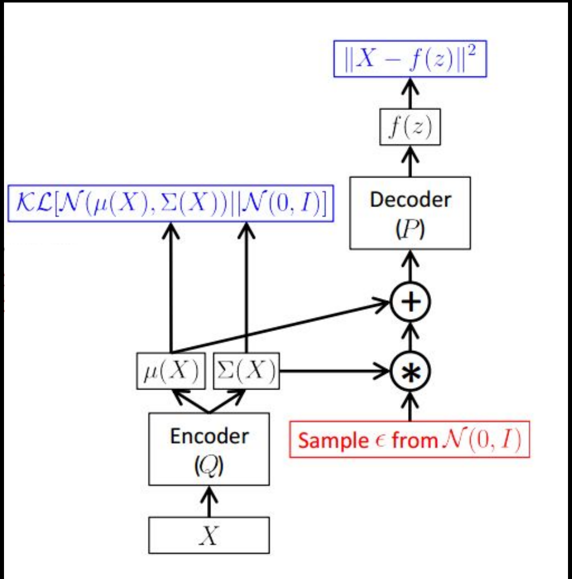
- Proposal distribution should be like Gaussian

Variational Auto-encoder (VAE)

Learning objective: maximize variational lower-bound

$$\log p_\theta(\mathbf{x}_i) \geq \underline{\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x}_i)}[\log p_\theta(\mathbf{x}_i|\mathbf{z})]} \underline{- KL[q_\phi(\mathbf{z}|\mathbf{x}_i)||p_\theta(\mathbf{z})]}$$

- KL-Divergence: closed-form and differentiable if both are Gaussians
- Reconstruction error: approximate by just sampling one z



$||X - f(z)||^2$
$f(z)$
Decoder (P)
$\mathcal{KL}[\mathcal{N}(\mu(X), \Sigma(X))||\mathcal{N}(0, I)]$
+
$*$
$\mu(X)$
$\Sigma(X)$
Encoder (Q)
Sample ϵ from $\mathcal{N}(0, I)$
X

Computation graph
Credit: Doersch

Variational Auto-encoder (VAE)

Why it is the variational lower-bound?

$$\log p_\theta(\mathbf{x}) = \log \int p_\theta(\mathbf{x}|\mathbf{z}) p_\theta(\mathbf{z}) d\mathbf{z}$$

$$\log p_\theta(\mathbf{x}) = \log \int p_\theta(\mathbf{x}|\mathbf{z}) \frac{p_\theta(\mathbf{z})}{q(\mathbf{z})} q(\mathbf{z}) d\mathbf{z}$$

Jenson inequality

$$\log \int p(\mathbf{x}) g(\mathbf{x}) d\mathbf{x} \geq \int p(\mathbf{x}) \log g(\mathbf{x}) d\mathbf{x}$$

$$\log p_\theta(\mathbf{x}) \geq \int q(\mathbf{z}) \log \left(p_\theta(\mathbf{x}|\mathbf{z}) \frac{p_\theta(\mathbf{z})}{q(\mathbf{z})} \right) d\mathbf{z}$$

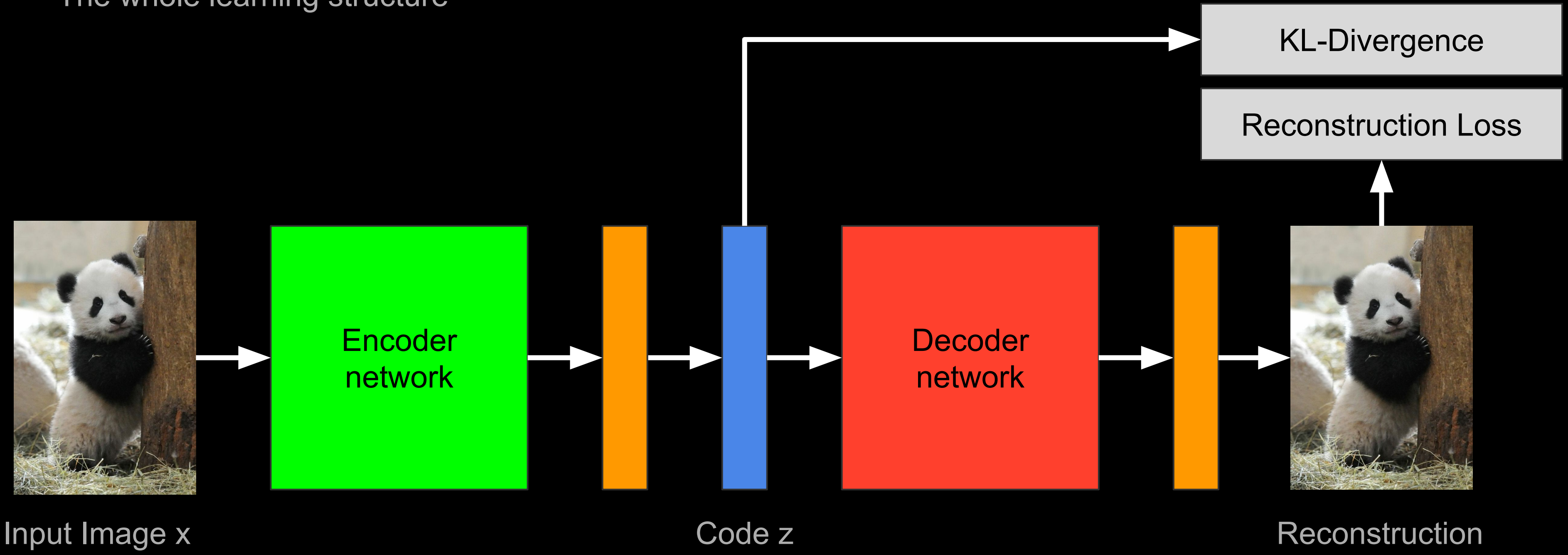
$$\log p_\theta(\mathbf{x}) \geq \int q(\mathbf{z}) \log p_\theta(\mathbf{x}|\mathbf{z}) d\mathbf{z} - \int q(\mathbf{z}) \log \frac{p_\theta(\mathbf{z})}{q(\mathbf{z})} d\mathbf{z}$$

$$\log p_\theta(\mathbf{x}) \geq \mathbb{E}_{q(\mathbf{z})}[\log p_\theta(\mathbf{x}|\mathbf{z})] - KL[q(\mathbf{z})||p_\theta(\mathbf{z})]$$

Kingma et al. 2014

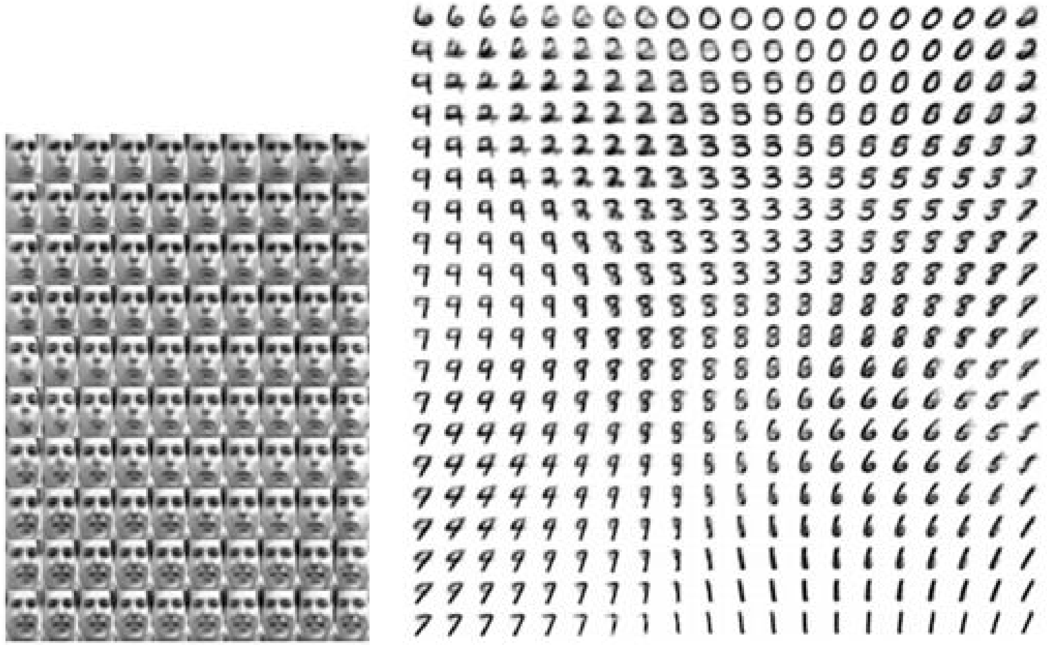
Variational Auto-encoder (VAE)

The whole learning structure



Variational Auto-encoder (VAE)

Results

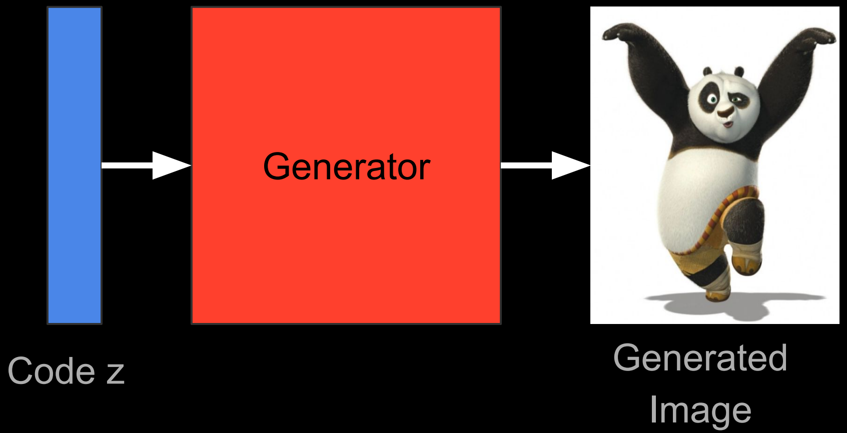


(a) Learned Frey Face manifold

(b) Learned MNIST manifold

Kingma et al. 2014

Generative Adversarial Network (GAN)

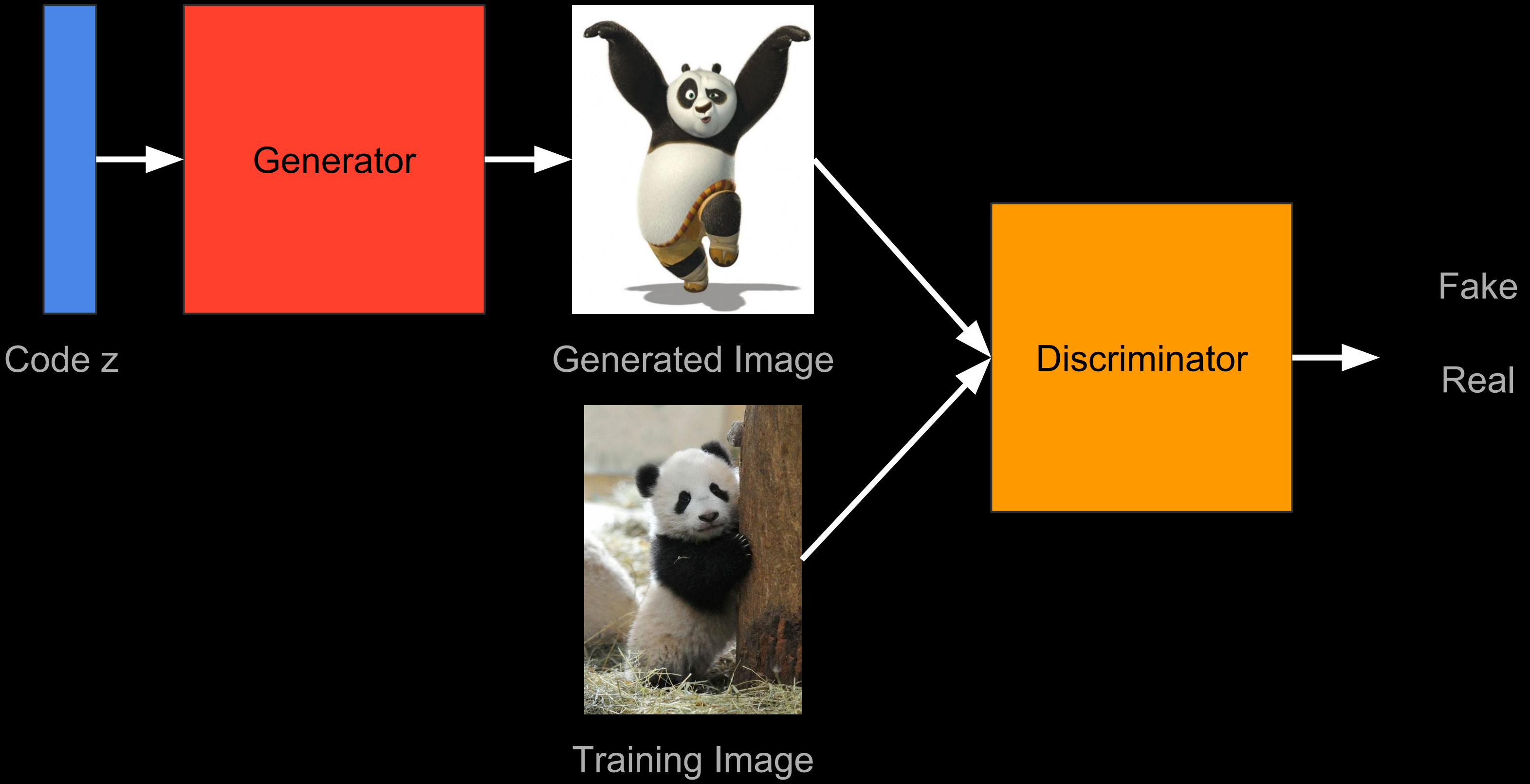


Code z

Generator

Generated Image

Generative Adversarial Network (GAN)



Code z

Generator

Generated Image

Training Image

Discriminator

Fake

Real

Generative Adversarial Network (GAN)

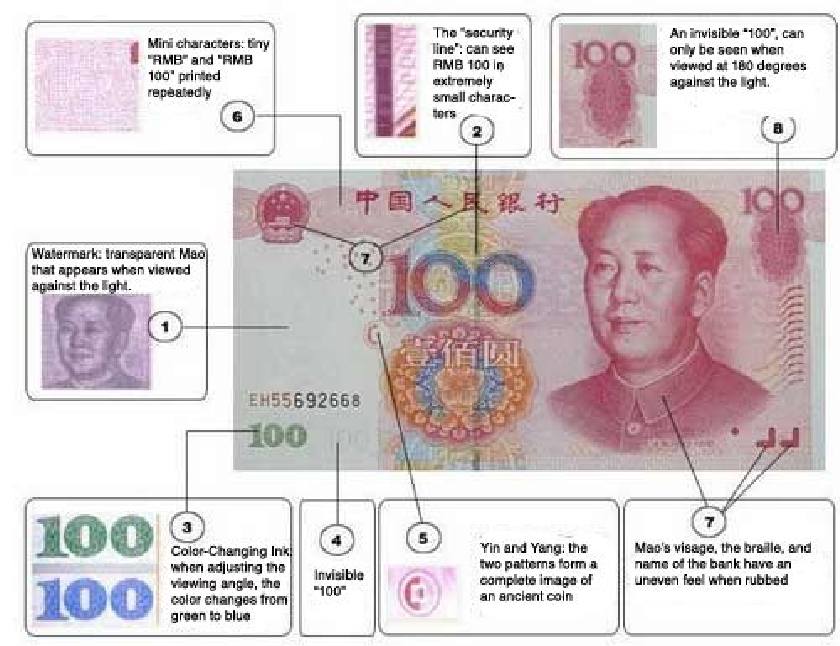
Intuitions

Crook

Generative Adversarial Network (GAN)

Intuitions

Generator



6 Mini characters: tiny "RMB" and "RMB 100" printed repeatedly

2 The "security line": can see RMB 100 in extremely small characters

8 An invisible "100", can only be seen when viewed at 180 degrees against the light.

1 Watermark: transparent Mao that appears when viewed against the light.

3 Color-Changing Ink: when adjusting the viewing angle, the color changes from green to blue

4 Invisible "100"

5 Yin and Yang: the two patterns form a complete image of an ancient coin

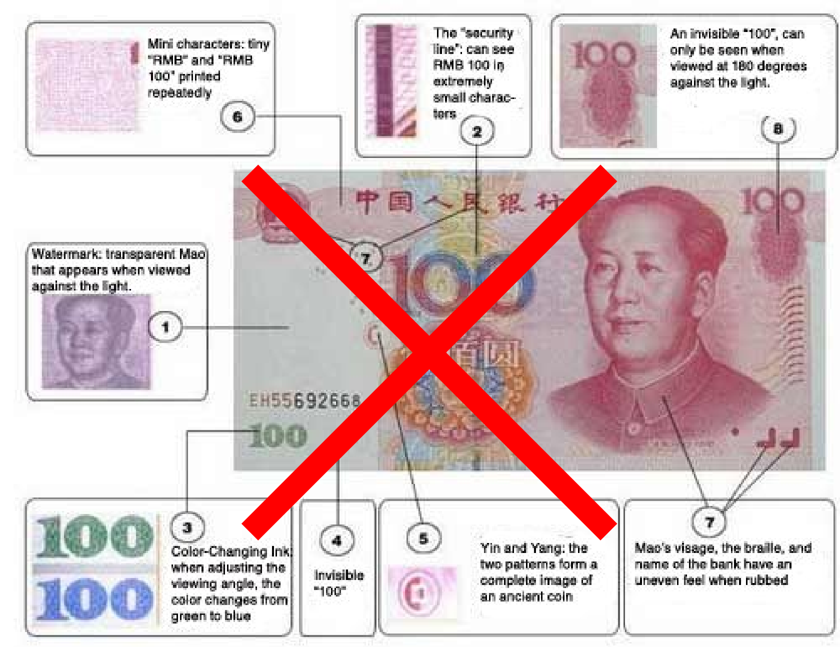
7 Mao's visage, the braille, and name of the bank have an uneven feel when rubbed

Teller

Generative Adversarial Network (GAN)

Intuitions

Crook



Mini characters: tiny "RMB" and "RMB 100" printed repeatedly
6
The "security line": can see RMB 100 in extremely small characters
2
An invisible "100", can only be seen when viewed at 180 degrees against the light.
8
Watermark: transparent Mao that appears when viewed against the light.
1
7
EH55692668
3
Color-Changing Ink: when adjusting the viewing angle, the color changes from green to blue
4
Invisible "100"
5
Yin and Yang: the two patterns form a complete image of an ancient coin
7
Mao's visage, the braille, and name of the bank have an uneven feel when rubbed

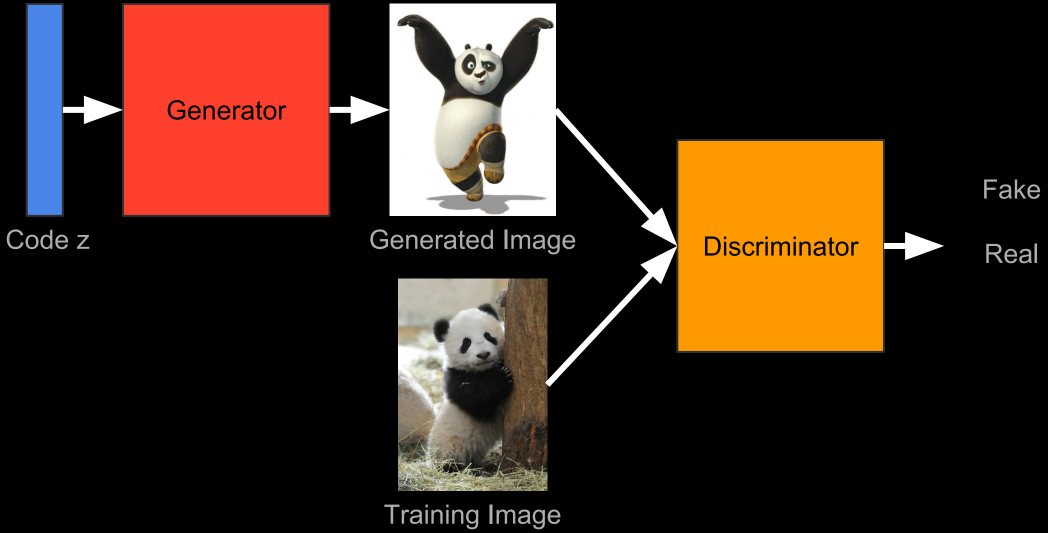
Teller

Generative Adversarial Network (GAN)

Intuitions:

- Generator tries the best to cheat the discriminator by generating more realistic images

- Discriminator tries the best to distinguish whether the image is generated by computers or not



Generative Adversarial Network (GAN)

Objective function:

$$\min_G \max_D V(D, G) = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})}[1 - \log D(G(\mathbf{z}))]$$

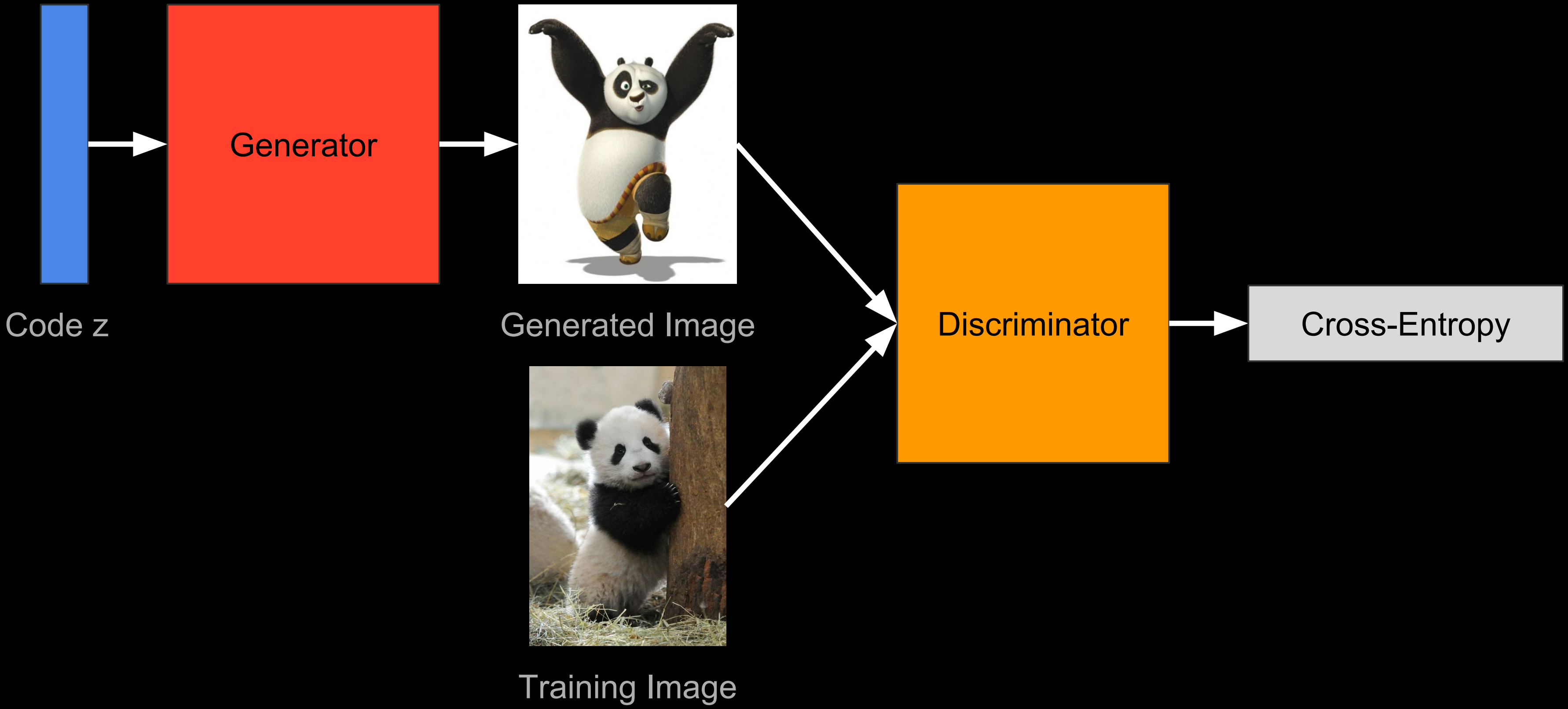
For each iteration:

- Sample a mini-batch of fake images and true images
- Update G using back-prop
- Update D using back-prop

Very difficult to optimize:

- Min-max problem: finding a saddle point instead of a local optimum, unstable

Generative Adversarial Network (GAN)



Code z

Generator

Generated Image

Training Image

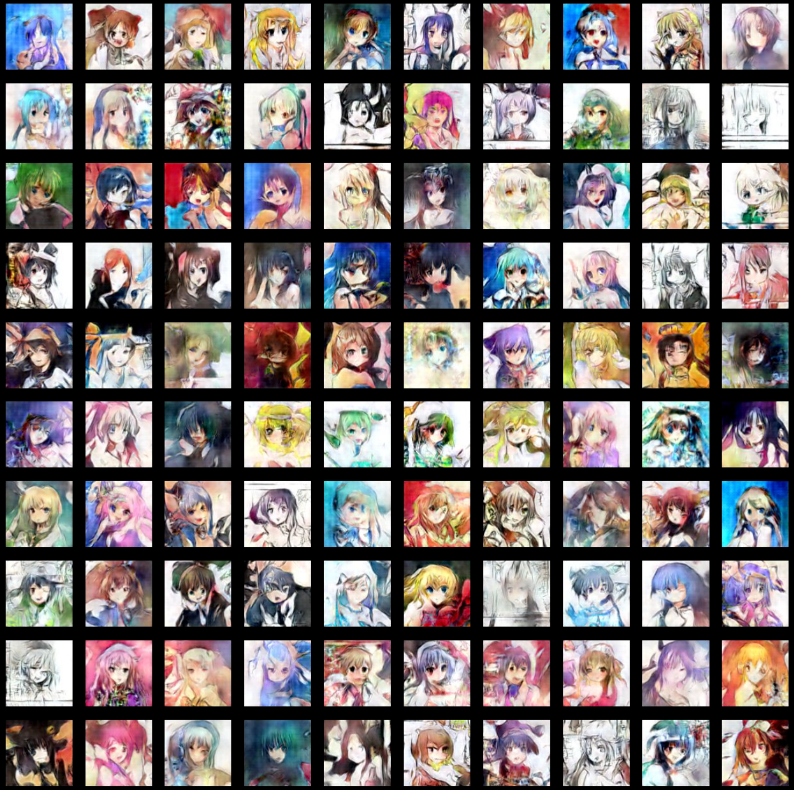
Discriminator

Cross-Entropy

GANs for face and bedroom

Credit: Denton

GANs for Japanese Anime



Credit: Radford

GAN for videos

Credit: Vondrick

Generative Adversarial Network (GAN)

Extensions:

- DCGANs: some hacks that work well
- LAPGANs: coarse-to-fine conditional generation through Laplacian pyramids
- f-GANs: more general GANs with different loss other than cross-entropy
- infoGANs: additional objective that maximize mutual-information between the latent and the sample
- EBGANs: Discriminative as energy functions
- GVMs: using GANs as an energy term for interactive image manipulation
- Conditional GANs: not random z, instead z is some data from other domain
- ...

Generative Adversarial Network (GAN)

Hacks:

- How to train a GAN?
- 17 hacks that make the training work.
- https://github.com/soumith/ganhacks

GANs vs VAEs

GANs:

- High-quality visually appealing result
- Difficult to train
- The idea of adversarial training can be applied in many other domains

VAEs:

- Easy to train
- Blurry result due to minimizing the MSE based reconstruction error
- Nice probabilistic formulation, easy to introduce prior

Demos

VAEs:

- https://github.com/oduerr/dl_tutorial/blob/master/tensorflow/vae/vae_demo.ipynb

GANs:

- https://github.com/ericjang/genadv_tutorial/blob/master/genadv1.ipynb
- https://gist.github.com/wiseodd/b2697c620e39cb5b134bc6173cfe0f56

References

[1] Goodfellow and Bengio, “*Deep learning* ” 2016
[2] Goodfellow, Ian, et al. "Generative adversarial nets." Advances in neural information processing systems. 2014.
[3] Kingma, Diederik P., and Max Welling. "Auto-encoding variational bayes." arXiv preprint arXiv:1312.6114 (2013).
[4] Radford, Alec, Luke Metz, and Soumith Chintala. "Unsupervised representation learning with deep convolutional generative adversarial networks." arXiv preprint arXiv:1511.06434 (2015).
[5] Nowozin, Sebastian, Botond Cseke, and Ryota Tomioka. "f-GAN: Training generative neural samplers using variational divergence minimization." Advances in Neural Information Processing Systems. 2016.
[6] Denton, Emily L., Soumith Chintala, and Rob Fergus. "Deep Generative Image Models using a￼ Laplacian Pyramid of Adversarial Networks." Advances in neural information processing systems. 2015.
[7] Chen, Xi, et al. "Infogan: Interpretable representation learning by information maximizing generative adversarial nets." Advances in Neural Information Processing Systems. 2016.
[8] Zhao, Junbo, Michael Mathieu, and Yann LeCun. "Energy-based generative adversarial network." arXiv preprint arXiv:1609.03126 (2016).
[9] Doersch, Carl. "Tutorial on variational autoencoders." arXiv preprint arXiv:1606.05908 (2016).
[10] Wang, K-C., and Richard Zemel. "classifying NBA offensive plays using neural networks." MIT Sloan Sports Analytics Conference, 2016.
[11] Wikipedia “Kernel density estimation”
[12] Wikipedia “Principal component analysis”
[13] Wikipedia “Autoencoder”

Thanks