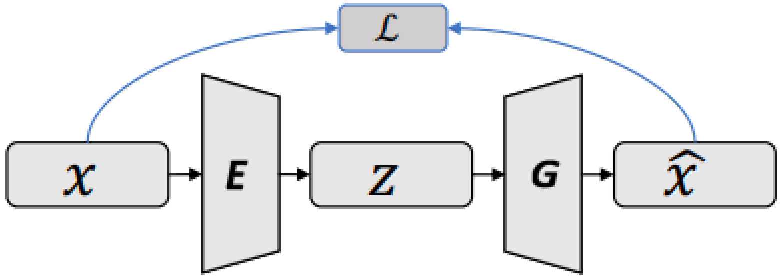
# AE and VAE

# Vanilla Autoencoder

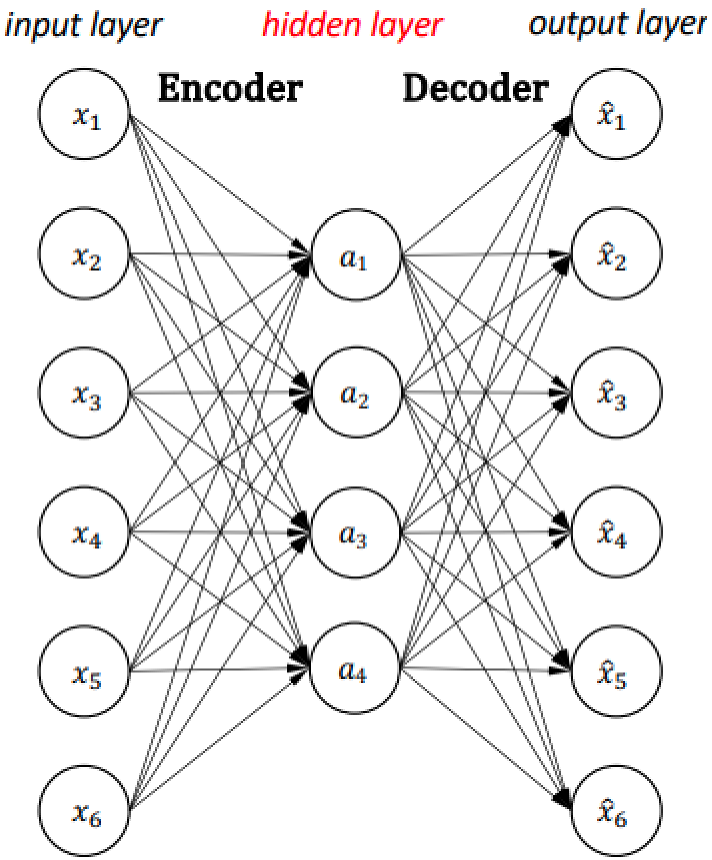



Reconstruct high-dimensional data using a neural network model with a narrow bottleneck layer

The bottleneck layer captures the compressed latent coding, so the nice by-product is dimension reduction

The low-dimensional representation can be used as the representation of the data in various applications, e.g., image retrieval, data compression …

# Vanilla Autoencoder

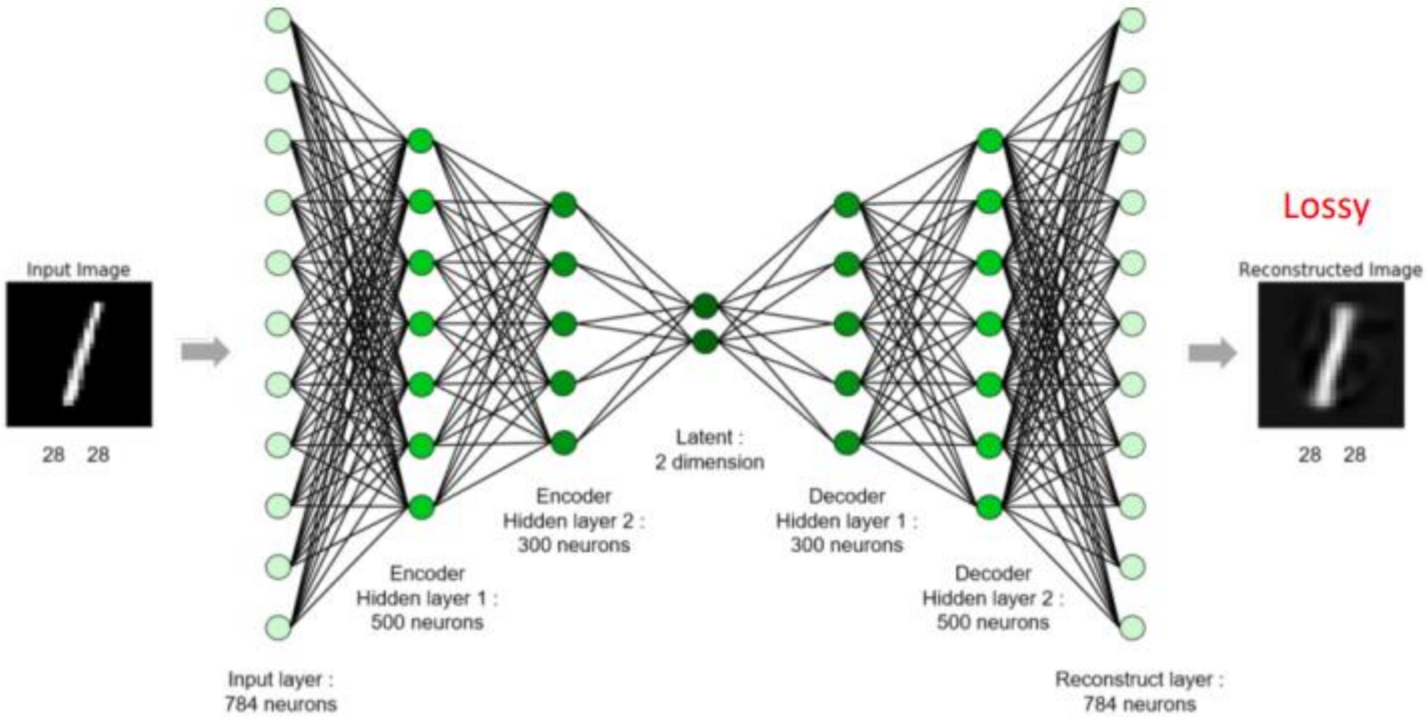



- The hidden units are usually less than the number of inputs
- Dimension reduction --- Representation learning

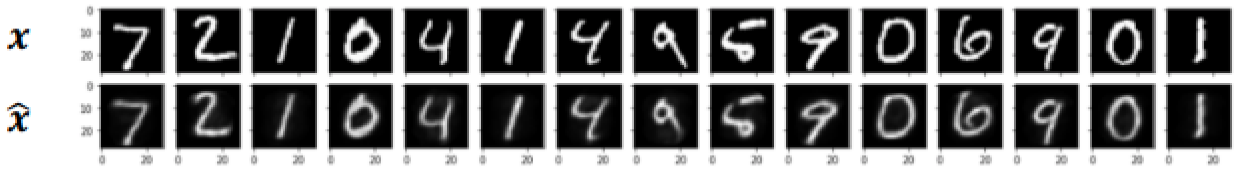
The distance between two data can be measure by Mean Squared Error (MSE):

$$\mathcal{L} = \frac{1}{n}\sum_{i=1}^{n}(x^i - G(E(x^i))^2$$

# Vanilla Autoencoder



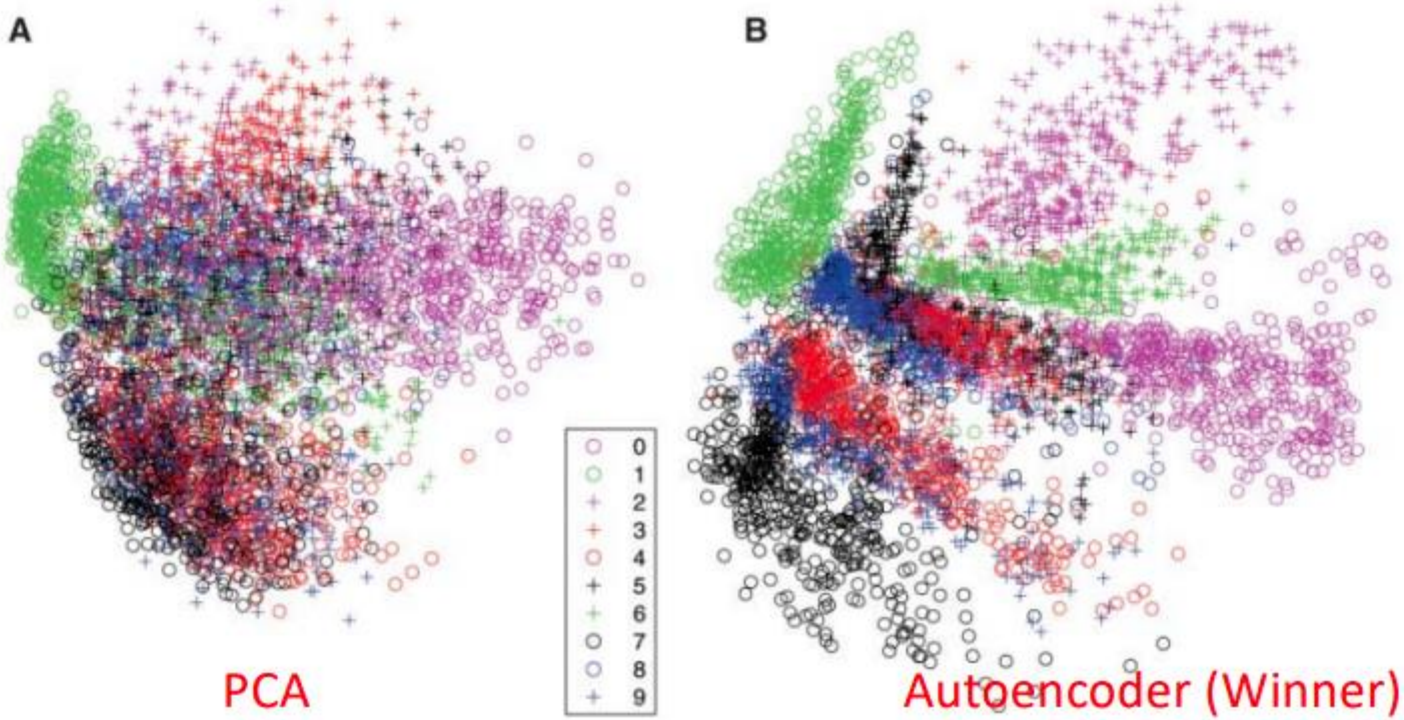

**Autoencoder for MNIST dataset (28×28×1, 784 pixels)**



# Vanilla Autoencoder VS PCA

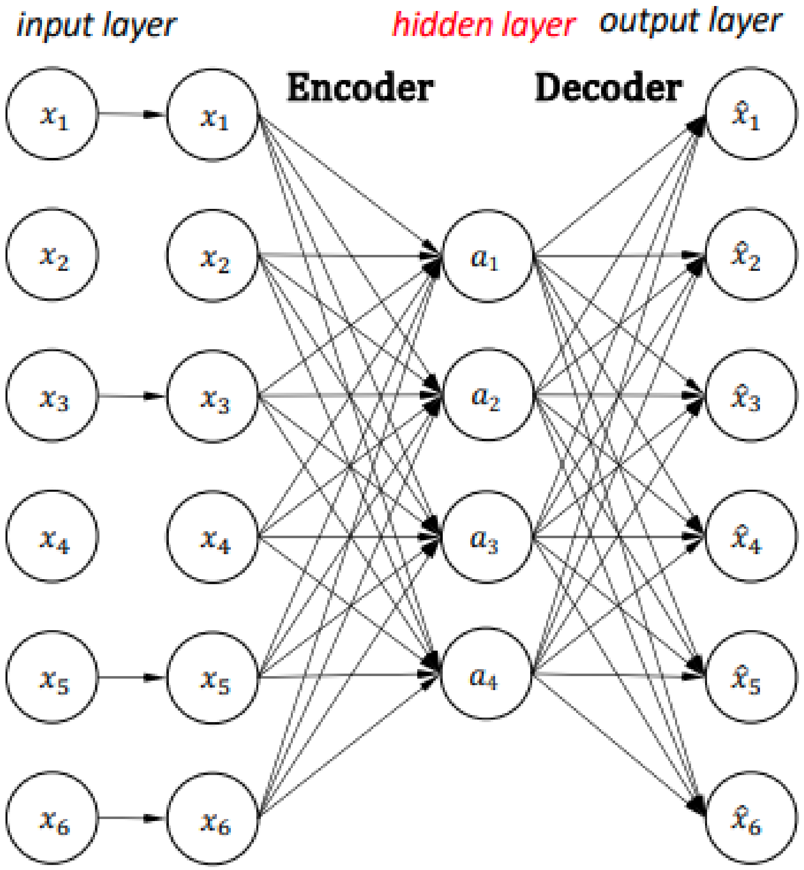
t-SNE visualization on MNIST: PCA vs. Autoencoder

**Fig. 3.** (**A**) The two-dimensional codes for 500 digits of each class produced by taking the first two principal components of all 60,000 training images. (**B**) The two-dimensional codes found by a 784-1000-500-250-2 autoencoder. For an alternative visualization, see (8).




Hinton and Salakhutdinov, science, 2016.

# Denoising Autoencoder



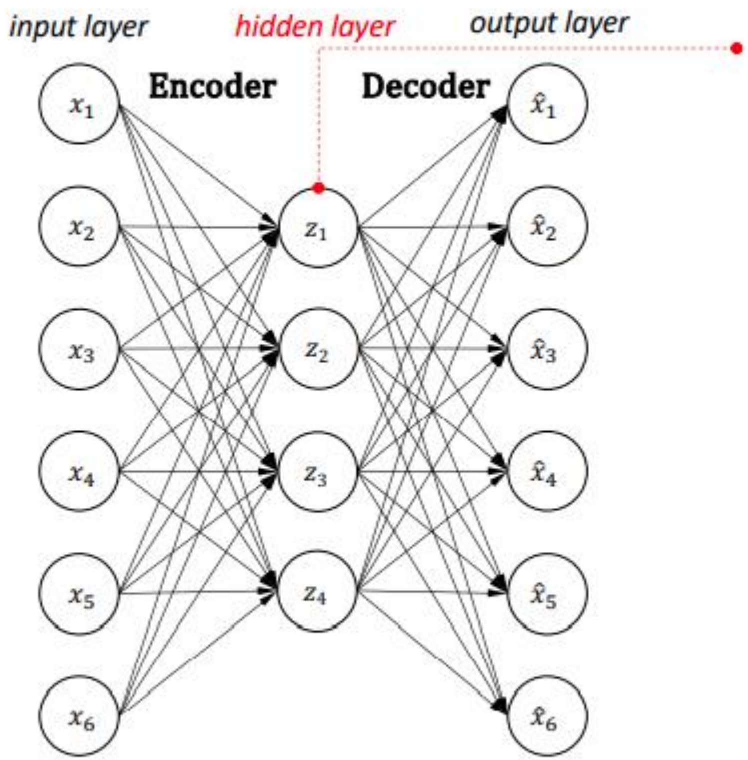

Denoising autoencoder was proposed in 2008, 4 years before the dropout paper (Hinton, et al. 2012).

Denoising autoencoder can be seem as applying dropout between the input and the first layer.

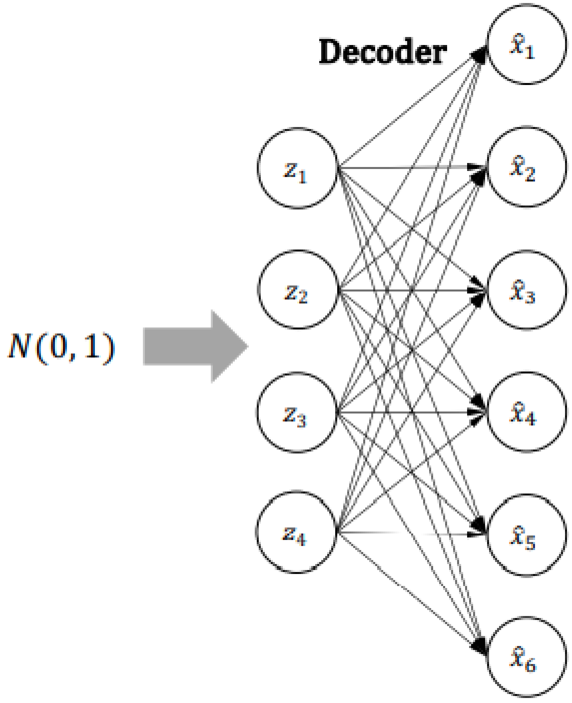
Denoising autoencoder can be seem as one type of data augmentation on the input.

# Variational Autoencoder

AE： the compressed latent codes of autoencoders are not prior distributions, autoencoder cannot learn to represent the data distribution.

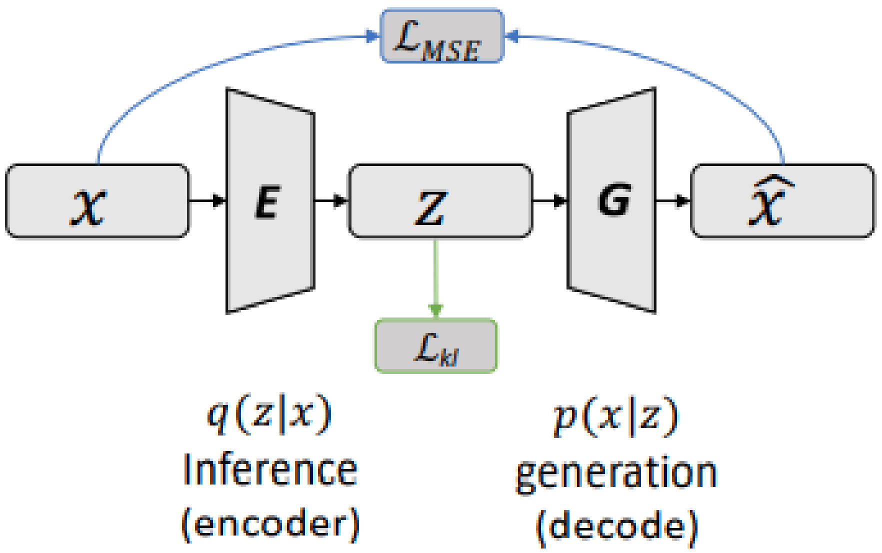



Can the hidden output be a prior distribution, e.g., Normal distribution?

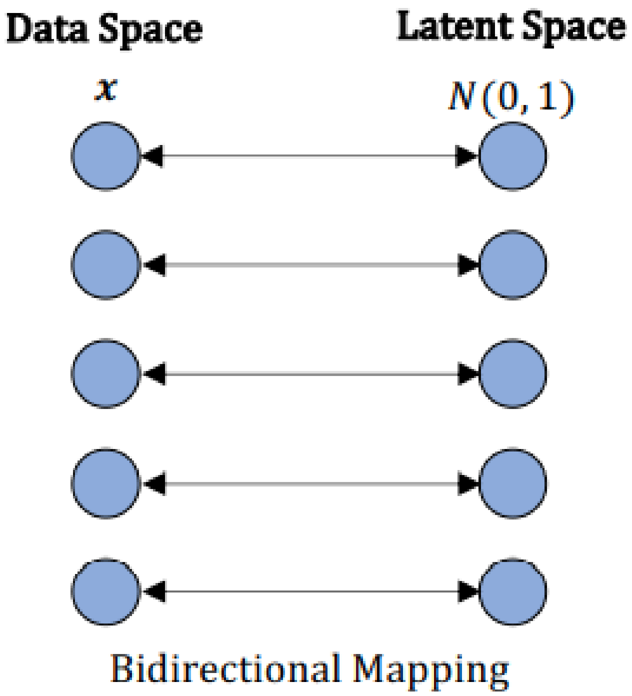



Decoder maps N(0, 1) to data space

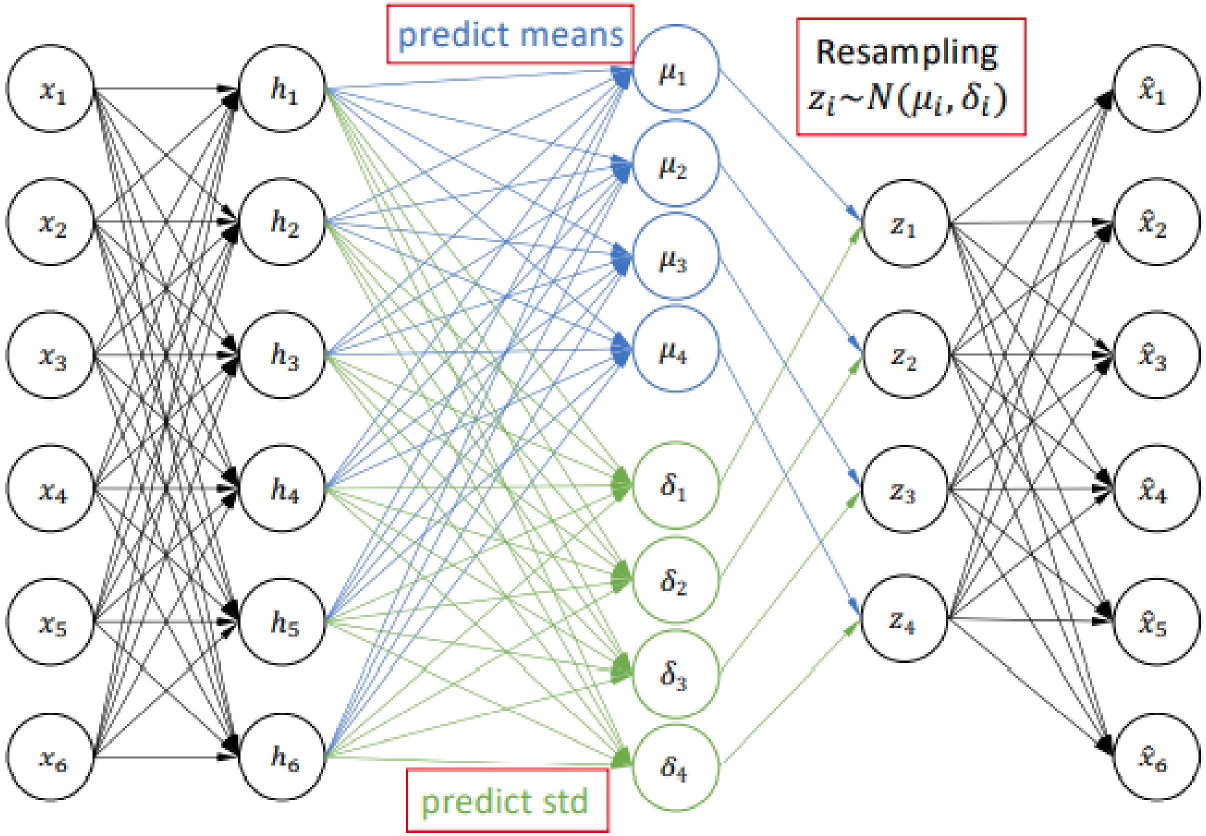
# Variational Autoencoder




$$\mathcal{L}_{total} = \mathcal{L}_{MSE} + \mathcal{L}_{kl}$$



# Variational Autoencoder




1. Encode the input
2. Predict means
3. Predict standard derivations
4. Use the predicted means and standard derivations to sample new latent variables individually
5. Reconstruct the input

# deepSEM

nature computational science

ARTICLES

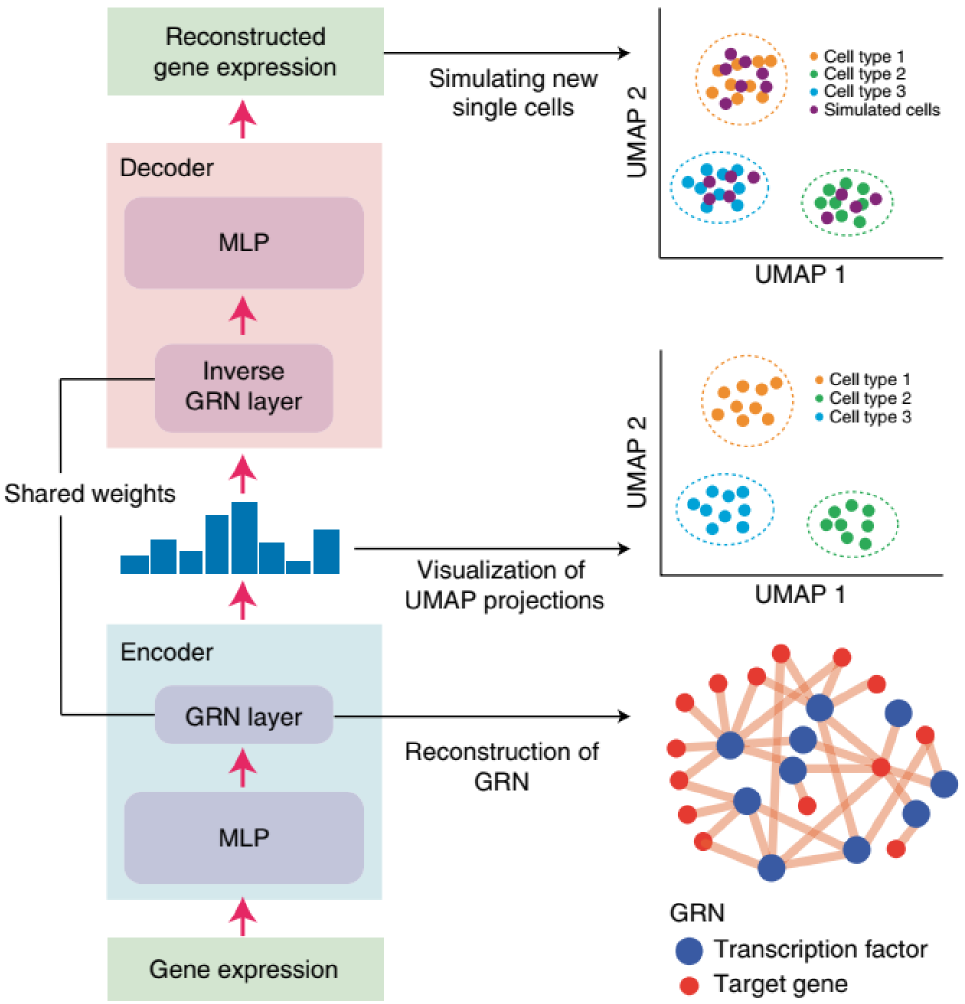
https://doi.org/10.1038/s43588-021-00099-8



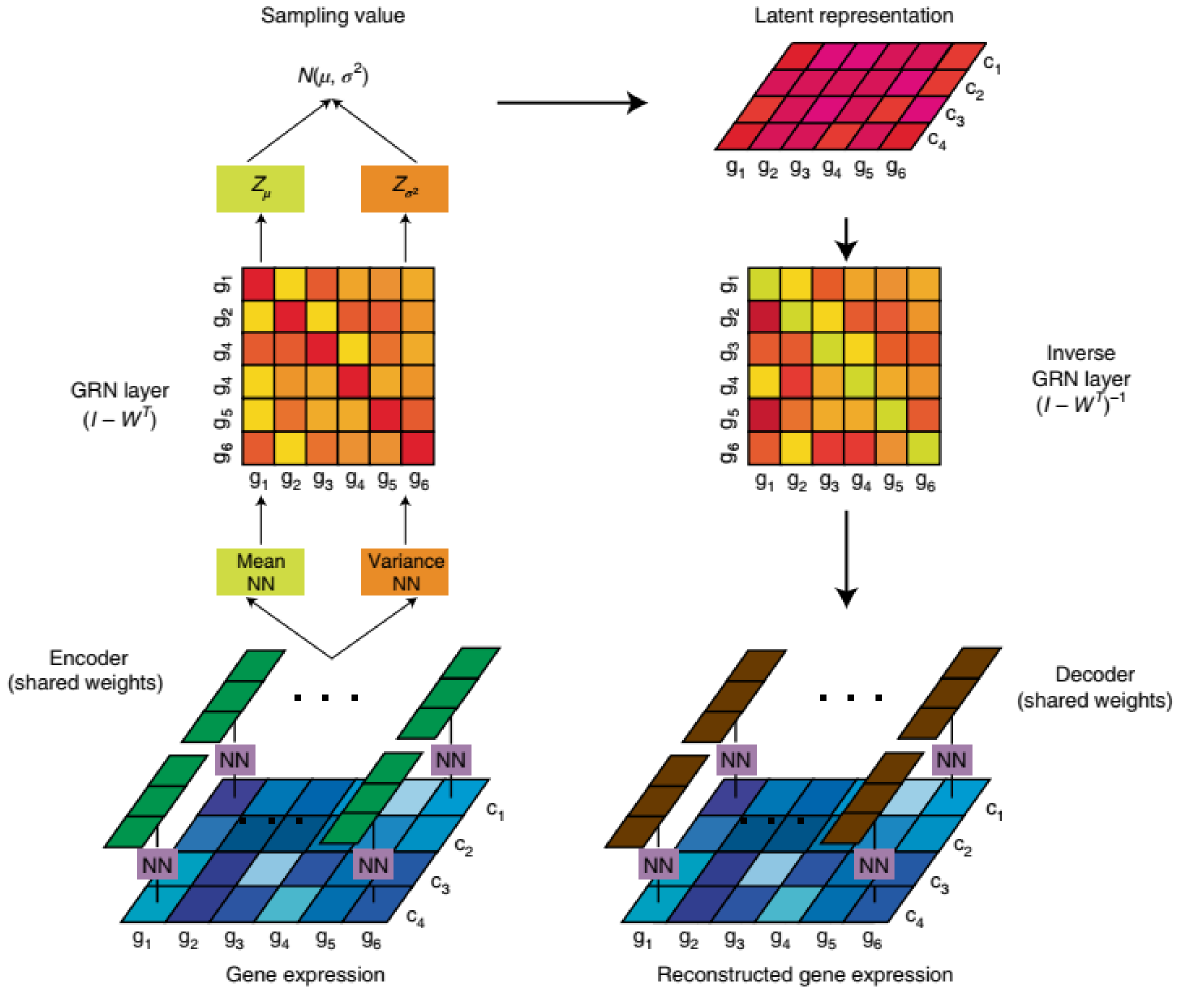
## Modeling gene regulatory networks using neural network architectures

Hantao Shu[1], Jingtian Zhou[2,3], Qiuyu Lian[4,5], Han Li[1], Dan Zhao[1], Jianyang Zeng[1] ✉ and Jianzhu Ma[6] ✉

# deepSEM



Reconstructed gene expression
Simulating new single cells
Decoder
MLP
Inverse GRN layer
Shared weights
Visualization of UMAP projections
Encoder
GRN layer
Reconstruction of GRN
MLP
Gene expression
UMAP 2
UMAP 1
Cell type 1
Cell type 2
Cell type 3
Simulated cells
GRN
Transcription factor
Target gene

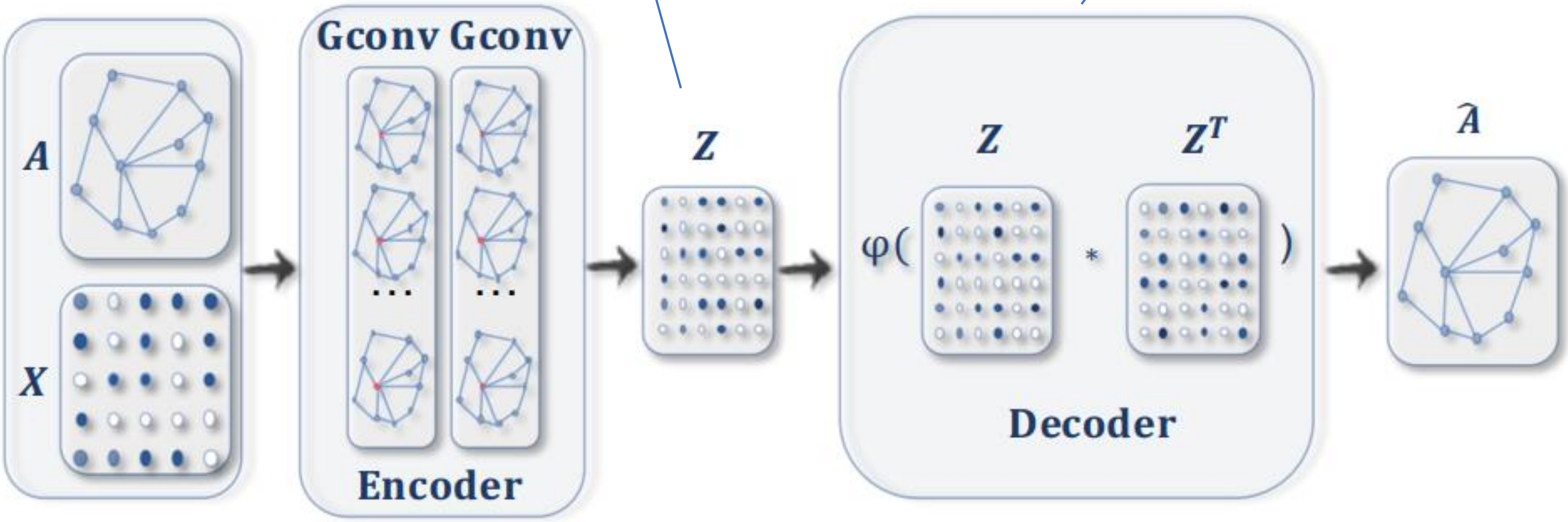
Sampling value
$N(\mu, \sigma^2)$
Latent representation
$Z_\mu$
$Z_{\sigma^2}$
GRN layer
$(I - W^T)$
Inverse GRN layer
$(I - W^T)^{-1}$
Mean NN
Variance NN
Encoder (shared weights)
Decoder (shared weights)
NN
$g_1$ $g_2$ $g_3$ $g_4$ $g_5$ $g_6$
$c_1$ $c_2$ $c_3$ $c_4$
Gene expression
Reconstructed gene expression

# GAE/ VGAE

# GAE

embedding

Similarity
reconstruct A




$X : N \times D$

$A : N \times N$

$Z : N \times F$

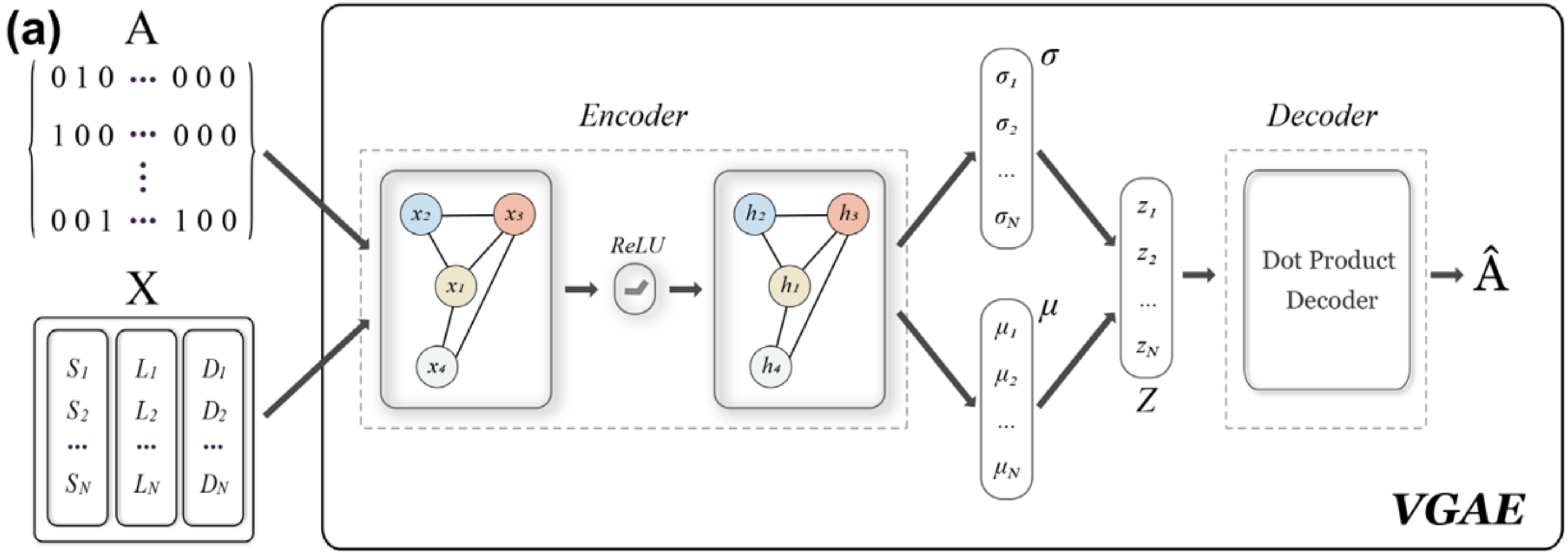
$Z = GCN(X, A) = \hat{A} ReLU(\hat{A} X W_0) W_1$

$\hat{A} = D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$

$A' : N \times N$

$A' = \sigma(ZZ^T) = sigmoid(ZZ^T)$

# VGAE




[DaehanKim/vgae_pytorch: This repository implements variational graph auto encoder by Thomas Kipf. (github.com)](https://github.com/DaehanKim/vgae_pytorch)

TECHNICAL NOTE

# Graph2GO: a multi-modal attributed network embedding method for inferring protein functions

Kunjie Fan[1], Yuanfang Guan,[2,*] and Yan Zhang[1,3,*]

[1]Department of Biomedical Informatics, College of Medicine, The Ohio State University, Columbus, OH 43210, USA; [2]Department of Computational Medicine and Bioinformatics, University of Michigan, Ann Arbor, MI 48109, USA and [3]The Ohio State University Comprehensive Cancer Center (OSUCCC - James), Columbus, OH 43210, USA

*Correspondence address. Yuanfang Guan, Department of Computational Medicine and Bioinformatics, University of Michigan, Ann Arbor, MI 48109, USA. E-mail: gyuanfang@umich.edu; Yan Zhang, Department of Biomedical Informatics, College of Medicine, The Ohio State University, Columbus, OH 43210, USA. E-mail: yanzhang.biomed@gmail.com http://orcid.org/0000-0002-3357-5121