



# Machine Learning

## Lecture 16: Dimensionality Reduction

Fall 2023

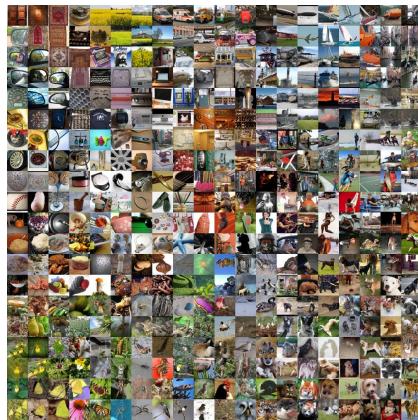
Instructor: Xiaodong Gu





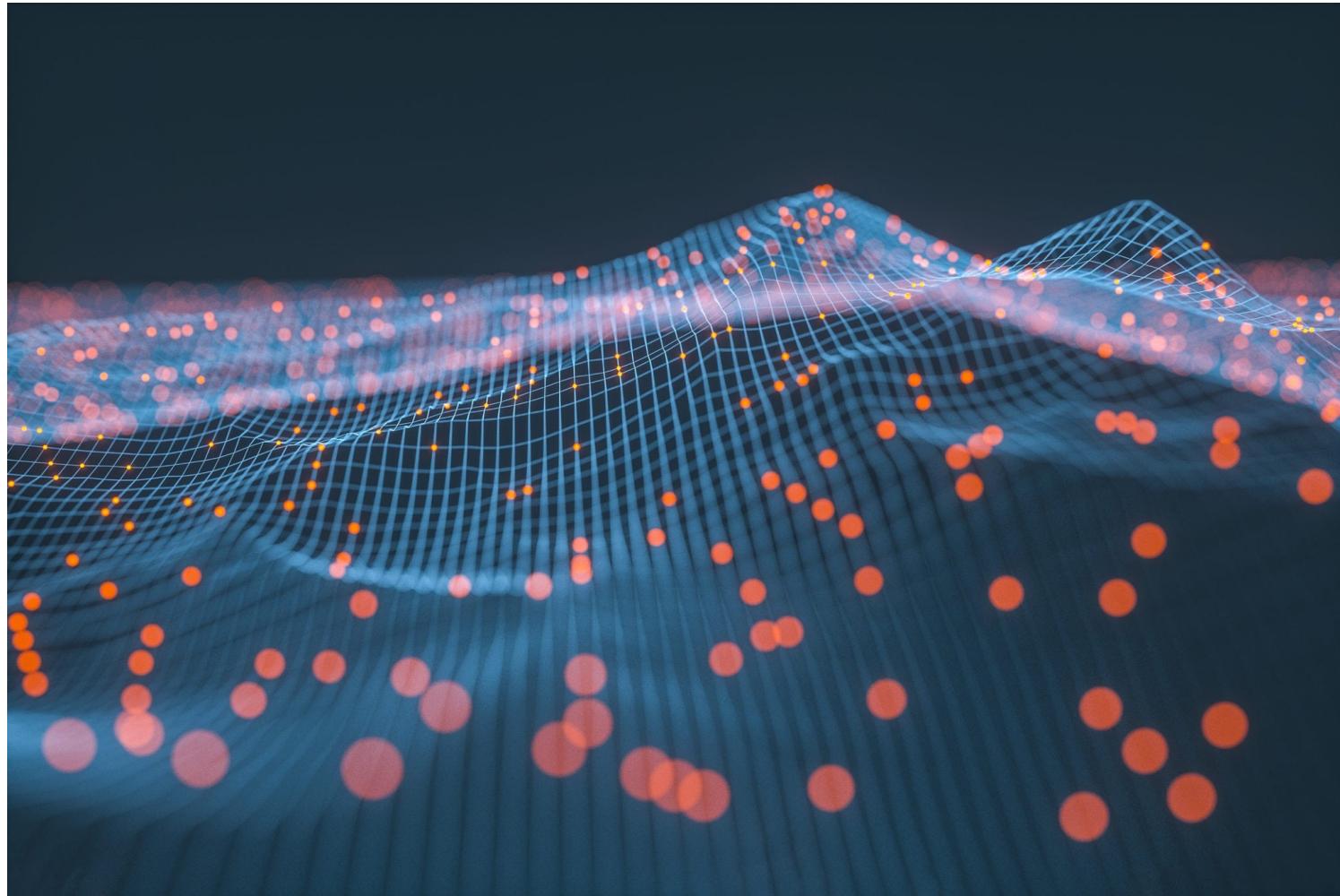
# Unsupervised Learning Again

Any other patterns beyond data clusters?



# Big Data, are they all useful?

---



# Today



物以类聚

化繁為簡

## Clustering (Learning data similarities)

- K-means
- Gaussian Mixture

無中生有

## Generative (density estimation)

- GAN
- VAE
- Diffusion

## Dimensional Reduction (data compression)

- Principal component analysis
- Auto-encoder



# The Curse of Dimensionality

"As the number of dimensions grows, the amount of data we need to generalize accurately grows exponentially."



Charles Isbell,  
Georgia Tech

## Example:

Given  $(x_1, \dots, x_N)$  with binary attributes, estimate  $p(x_1, \dots, x_N)$

$X_1$ : gender	$X_2$ : major	$X_3$ : age	...	$P(x_1, x_2, x_3)$
Male	CS	<25	...	0.9
Female	Finance	>25	...	0.1
...	...	...	...	...
Male	CS	<25	...	0.1

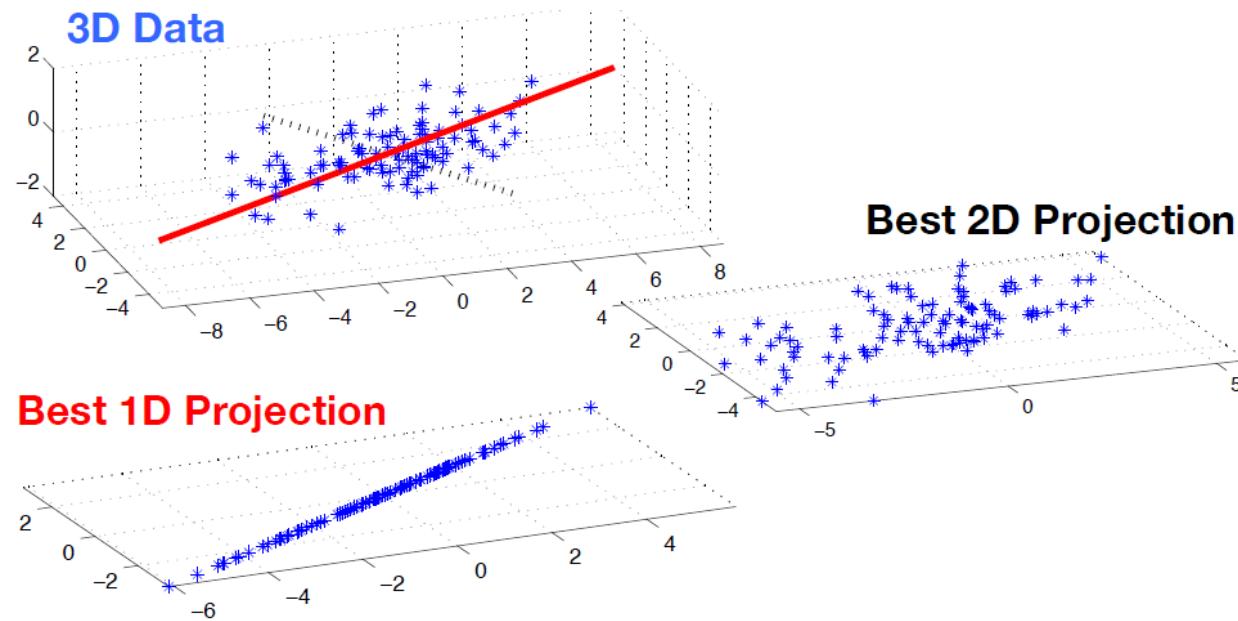
Q: How many items are needed in this table?



# Dimensionality Reduction

Reduce the number of dimensions (attributes) of the data while preserving the intrinsic characteristics of the data

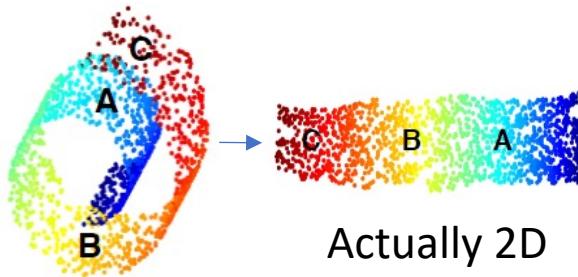
*Try to identify some “hidden” aspects that determines the characteristics of the data*





# Why Reduce Dimensionality?

Data may intrinsically live in a lower-dim space



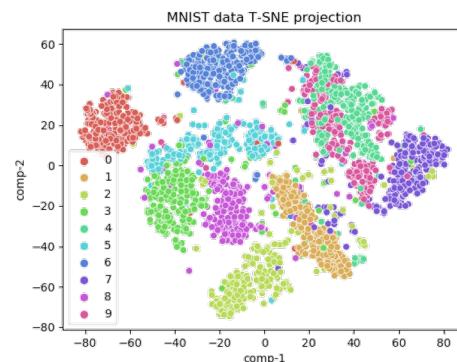
Too many features and too few data

$X_1:$	$X_2:$	$X_3:$	...	$P(x_1, x_2, x_3)$
M	CS	<25	...	0.9
F	Fin	>25	...	0.1
...	...	...	...	...
M	CS	<25	...	0.1

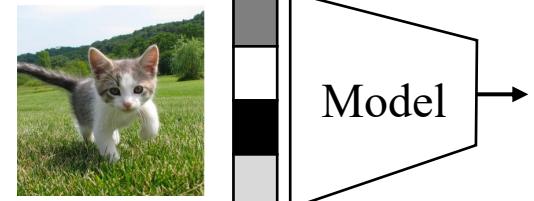
Lower computational expense (memory, time)



Want to visualize data in a lower-dim space



Want to use data of different dimension



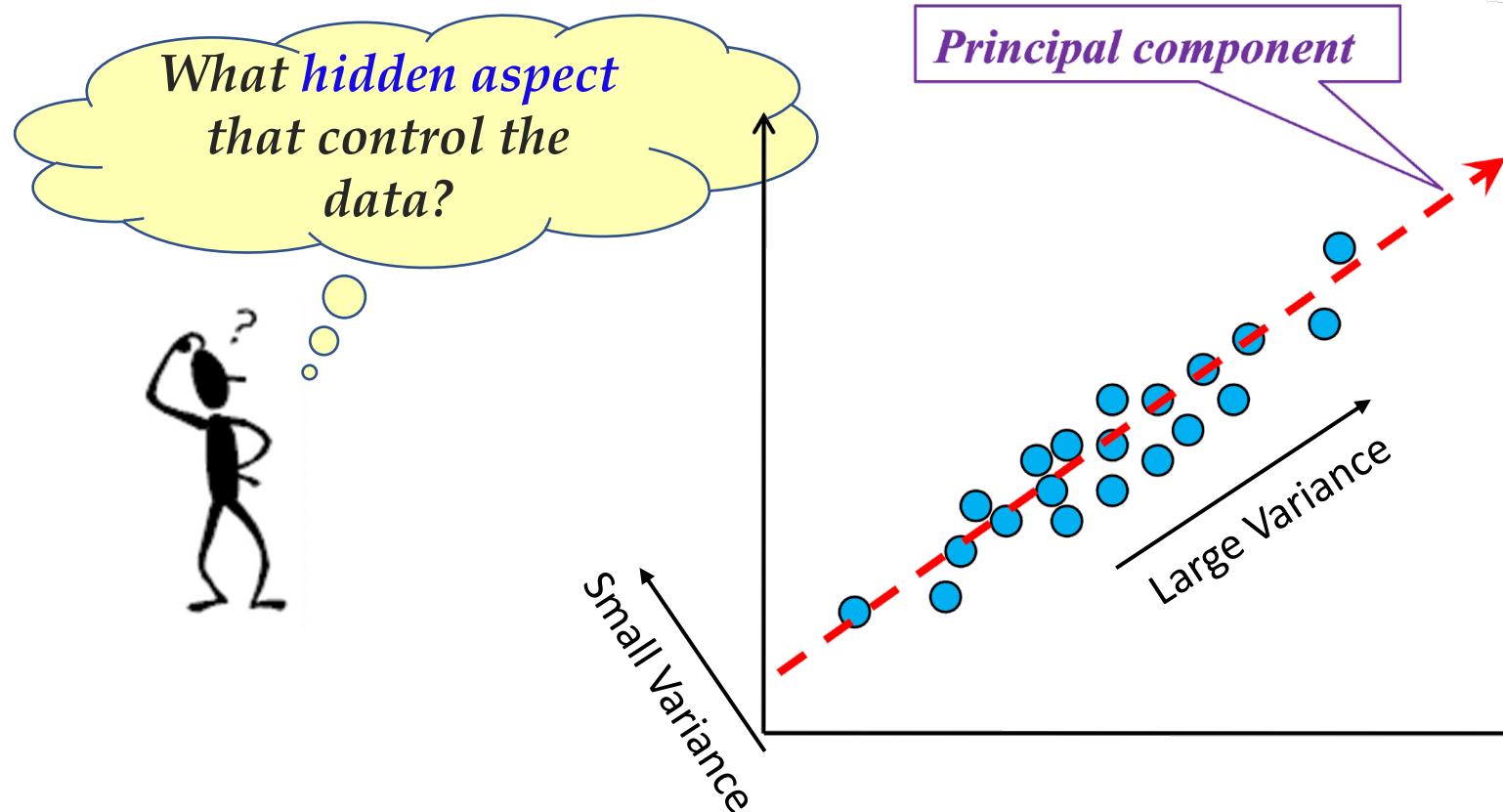
# Overview of Methods

---



- Linear Method
  - Principal component analysis (PCA)
  - Linear discriminate analysis (LDA)
  - Factor analysis
  - ...
- Non-linear Method
  - KPCA, KLDA
  - Multidimensional scaling (MDS)
  - t-SNE
  - Autoencoder
  - ...

# Principal Component Analysis (PCA)

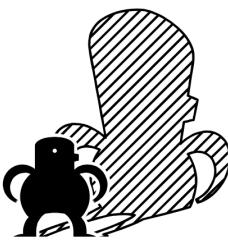


**Principal component**

= the spreading tendency

= dimensionality with the **maximum variance**

# Problem Formulation



- Data representations

$$X = [x_1, \dots, x_N], \quad x_i \in \mathbb{R}^d, \text{ where } \sum_{i=1}^N x_i = 0$$

- Projection of a data point

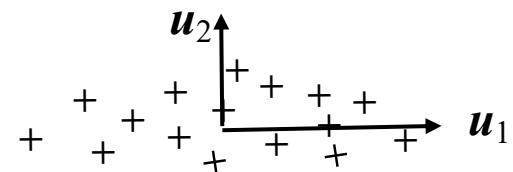
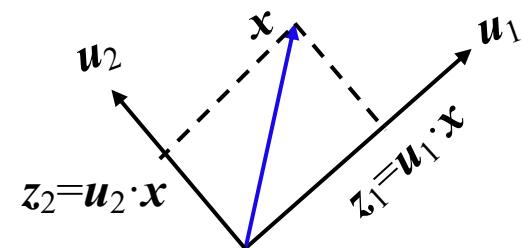
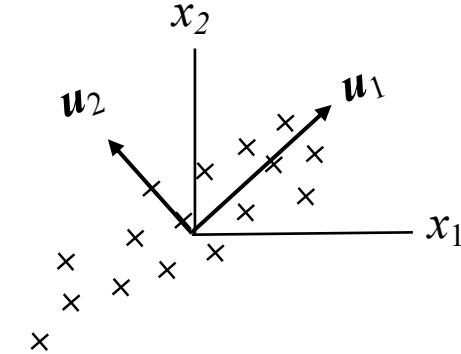
$$z_j = \mathbf{u}_j^T x \quad \mathbf{Z} = \mathbf{U}^T X$$

- Variance in any projected dimension

$$\begin{aligned} \text{var}(\mathbf{u}^T X) &= \frac{1}{N} (\mathbf{u}^T X)(\mathbf{u}^T X)^T \\ &= \frac{1}{N} \mathbf{u}^T \mathbf{X} \mathbf{X}^T \mathbf{u} \\ &= \mathbf{u}^T S \mathbf{u} \end{aligned}$$

$$S = \text{cov}(X) = \frac{1}{N} \mathbf{X} \mathbf{X}^T$$

Can simply subtract each  $x_i$  by the sample mean.





# PCA = Solving an Optimization Problem

- In summary, PCA aims to solve a constrained optimization problem:

$$\max_u \ u^T S u$$

$$\text{s.t. } u^T u = 1$$

- This is a **quadratic programming** (QP) problem, which is one type of convex optimization problem.
- $S$  is positive semi-definite.

$$\begin{aligned} x^T S x &= x^T E[(X - \mu)^T (X - \mu)] x \\ &= E[x^T (X - \mu)^T (X - \mu) x] \\ &= E[((X - \mu)x)^T ((X - \mu)x)] \\ &= E[\|(X - \mu)x\|^2] \\ &= \|(X - \mu)x\|^2 \\ &\geq 0 \end{aligned}$$



# Solving PCA

- Lagrangian:  $\mathcal{L} = \mathbf{u}^T \mathbf{S} \mathbf{u} - \lambda(\mathbf{u}^T \mathbf{u} - 1)$

$$\nabla_{\mathbf{u}} \mathcal{L} = 0 \Rightarrow \mathbf{S} \mathbf{u} = \lambda \mathbf{u}$$

- Solving PCA subjects to **eigen decomposition**:

$\mathbf{u}$ : eigen vector,  $\lambda$ : eigen value

- Since  $\mathbf{S}$  is positive semi-definite, we have

$$\mathbf{S} = \mathbf{U} \Sigma \mathbf{U}^T = [\mathbf{u}_1, \dots, \mathbf{u}_d] \text{diag}(\lambda_1, \lambda_2, \dots, \lambda_d) [\mathbf{u}_1, \dots, \mathbf{u}_d]^T$$



# The Idea

- Find  $k$  orthogonal principal components  
correspond to the  $k$  eigen vectors with  $k$  largest eigen values

$$\mathbf{S} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{U}^T \approx \mathbf{U}_{1:k}\boldsymbol{\Sigma}_{1:k}\mathbf{U}^T_{1:k}$$

- Project each input vector  $\mathbf{x}$  into this subspace

$$z_j = \mathbf{u}_j^T \mathbf{x}$$

$$\mathbf{z} = \mathbf{U}_{1:k}^T \mathbf{x}$$

## Intuitions:

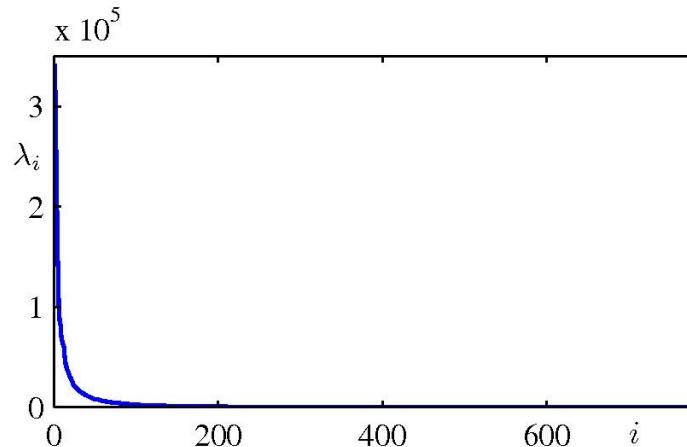
- Eigen value is the variance of the data projected to the corresponding eigen vector
- Eigen vectors are essentially orthogonal



# Determine “k”

- **What to discard:** the eigen vectors with small eigen values
- Limit the loss of information within an acceptable interval (usually 5%)

$$\text{info loss} = \frac{\lambda_{k+1} + \dots + \lambda_d}{\lambda_1 + \lambda_2 + \dots + \lambda_d}$$



Variance preserved at  $i$ -th eigenvalue



# The Algorithm

---

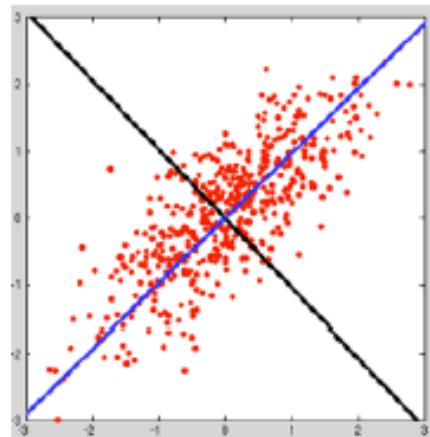
1. Shifting data set to have zeros mean
  2. Compute the covariance matrix  $S$
  3. Conduct eigen decomposition on  $S$ , and rank the eigen vectors according to their eigen values
  4. Determine the number of dimensions  $k$  of the new space
  5. Select the first  $k$  eigen vectors with  $d$  largest eigen values as the basis of the new space
-



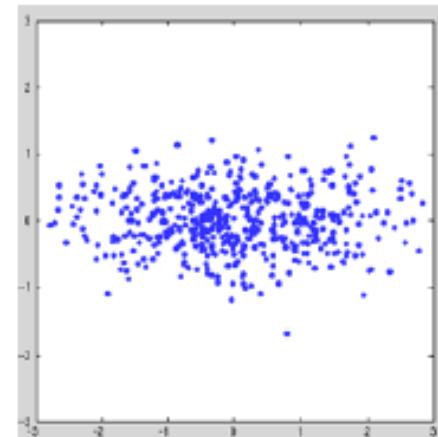
# The Algorithm

## Matlab Code

```
function [E, U, lambda] = PCA(X, K)
mu = mean(X); % Data mean
N = size(X,1); % Number of data
X = X - ones(N,1)*mu; % Center the data
covX = X'*X; % Data covaraince
[U,lambda] = eigs(covX, K); % Compute top K eigenvalues
E = X*U; % Compute embeddings
```



PCA projections





# Applications of PCA

---

- Data visualization
- Preprocessing
- Modeling - prior for new data
- Compression



# Application: Face Recognition

- The space of face images are extremely high-dimensional
  - $24 \times 24$  image = 576 dimensions
  - slow and lots of storage



But very few 576-dimensional vectors are valid face images

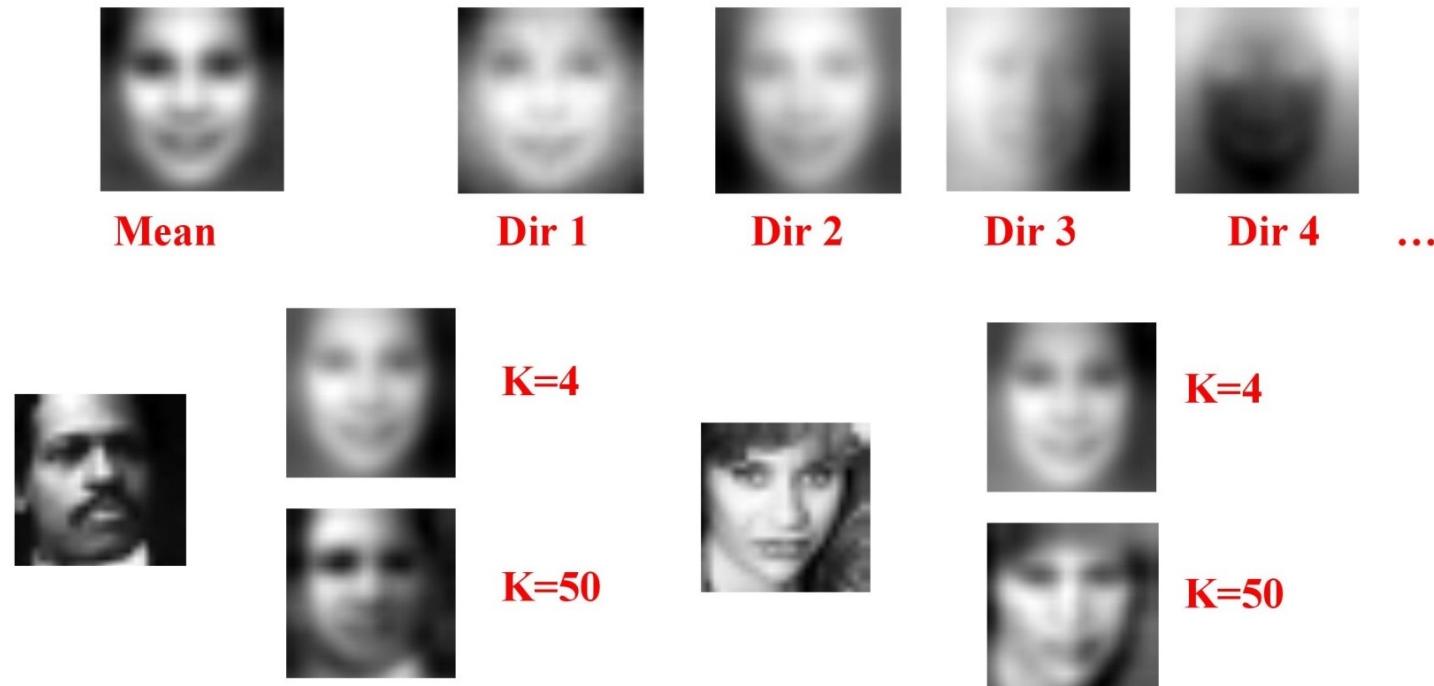
How to effectively model the subspace of face images?



# “Eigen-Faces”

Example: Viola Jones data set

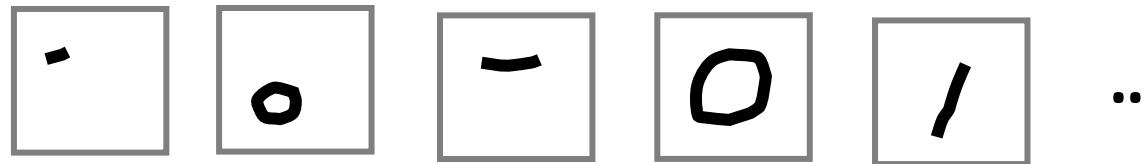
- $24 \times 24$  images of faces = 576 dimensional measurements
- Take the first-K PCA components





# PCA – Another Perspective

Basic components of handwriting digits:



$$u_1 \quad u_2 \quad u_3 \quad u_4 \quad u_5$$

$$\begin{matrix} 7 \\ x \end{matrix} \approx \begin{matrix} \text{---} \\ u_1 \end{matrix} + \begin{matrix} \text{---} \\ u_3 \end{matrix} + \begin{matrix} / \\ u_5 \end{matrix} \rightarrow \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ \vdots \end{bmatrix}$$

$$x \approx z_1 u_1 + z_2 u_2 + \dots + z_k u_k + \bar{x}$$

Image pixels

components

sample mean

$$\begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_k \end{bmatrix}$$

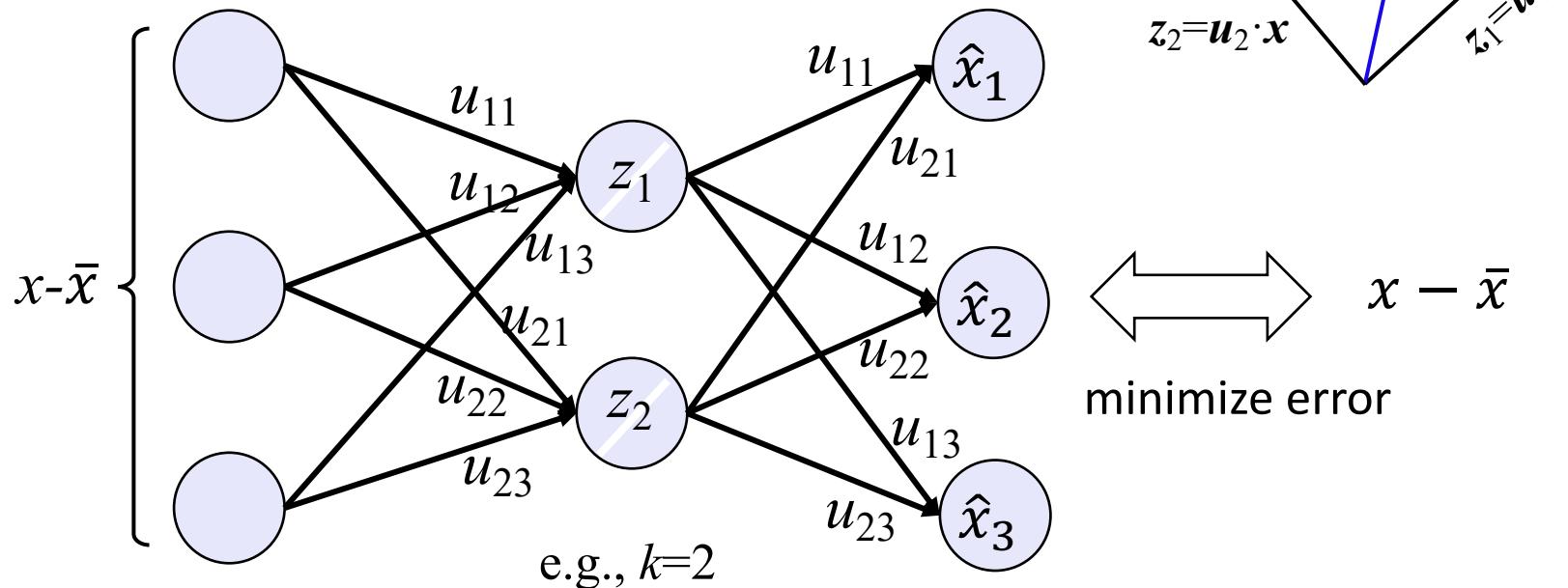
Compressed representation

# PCA = Neural Network

$$\hat{x} = \sum_{j=1}^K z_j \mathbf{u}_j \iff x - \bar{x}$$

To minimize reconstruction error:

$$z_j = (x - \bar{x}) \cdot \mathbf{u}_j$$

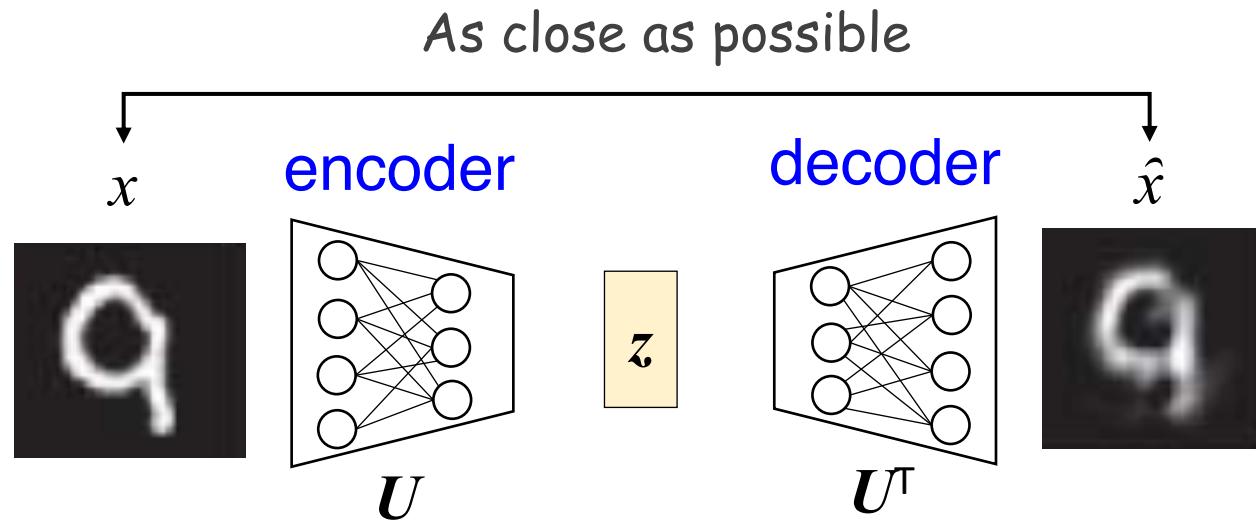


PCA = a neural network with one hidden layer (linear activation)



# PCA = Auto-encoder

- **Auto-encoder:** an encoder-decoder **neural network** whose output tries to reconstruct the input.

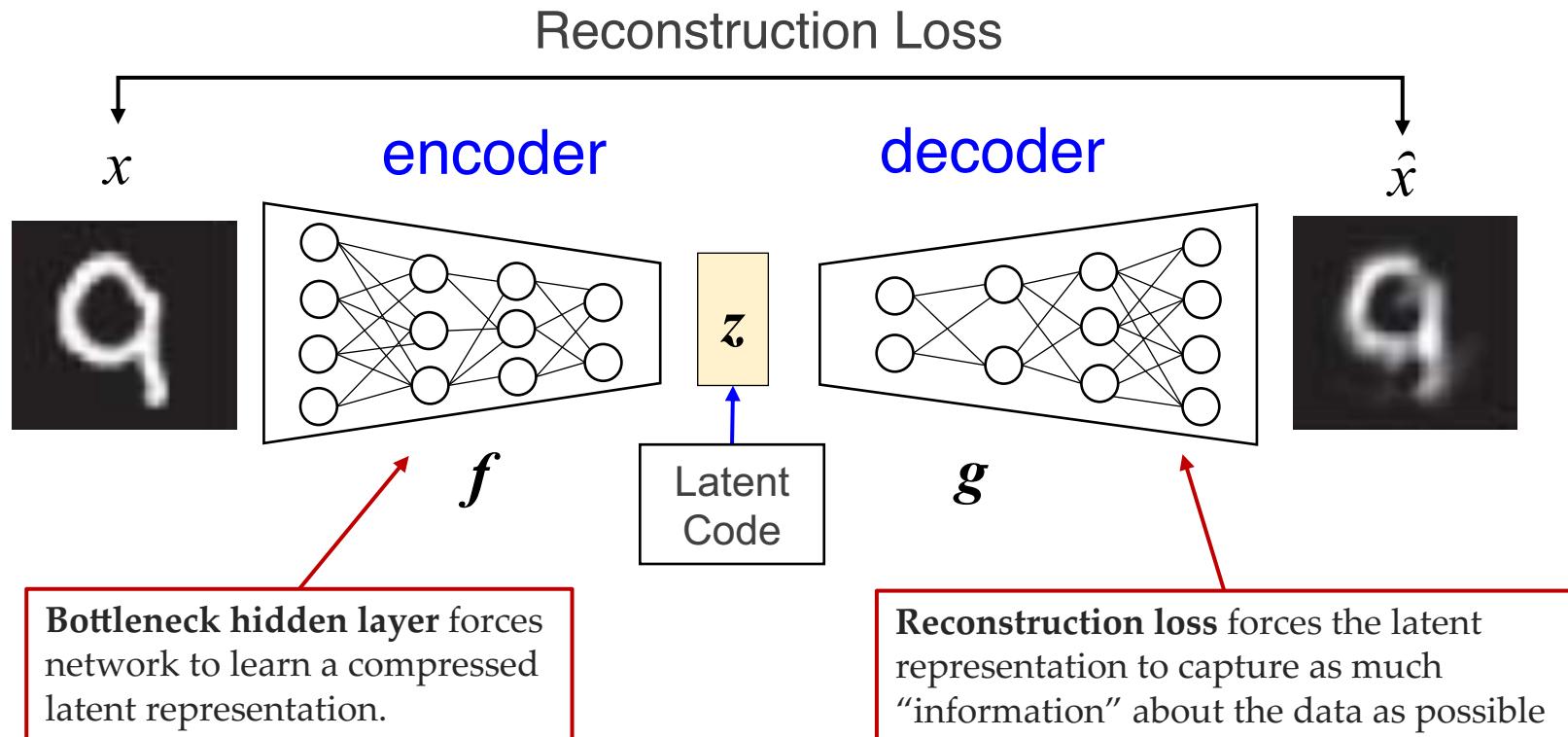


Learns a **lower-dimensional** feature representation ( $z$ ) from unlabeled training data ( $x$ ).



# Deep Auto-encoder

- Of course, the auto-encoder can be deep



Hinton, Geoffrey E., and Ruslan R. Salakhutdinov. "Reducing the dimensionality of data with neural networks." *Science* 313.5786 (2006): 504-507

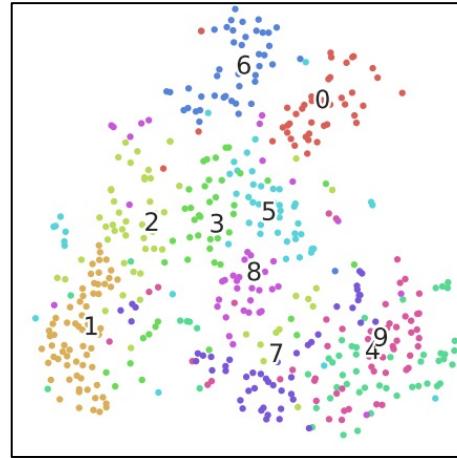


# Auto-encoder vs PCA

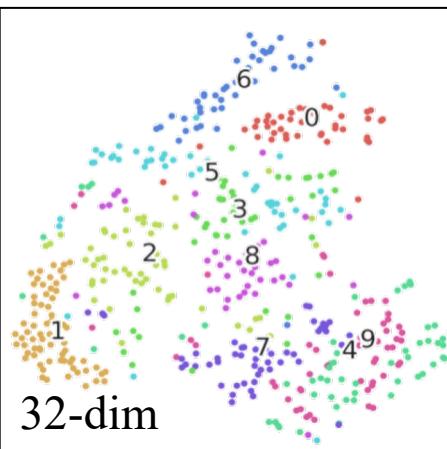
## Example

```
0 1 2 3 4 5 6 7 8 9  
0 1 2 3 4 5 6 7 8 9  
0 1 2 3 4 5 6 7 8 9  
0 1 2 3 4 5 6 7 8 9  
0 1 2 3 4 5 6 7 8 9  
0 1 2 3 4 5 6 7 8 9  
0 1 2 3 4 5 6 7 8 9  
0 1 2 3 4 5 6 7 8 9  
0 1 2 3 4 5 6 7 8 9  
0 1 2 3 4 5 6 7 8 9
```

PCA

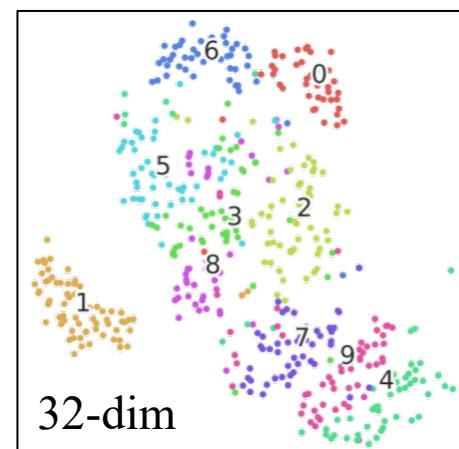


Pixels  
(tSNE)



32-dim

NN Auto-encoder

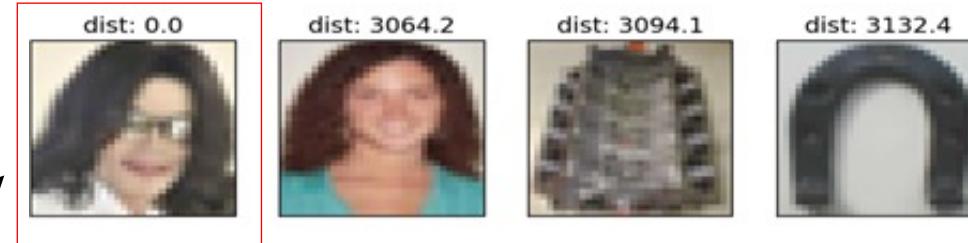


32-dim



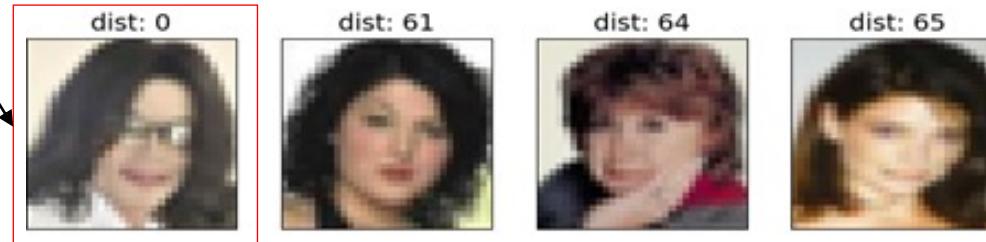
# Applications – Image Search

Retrieved in pixel intensity space



Query

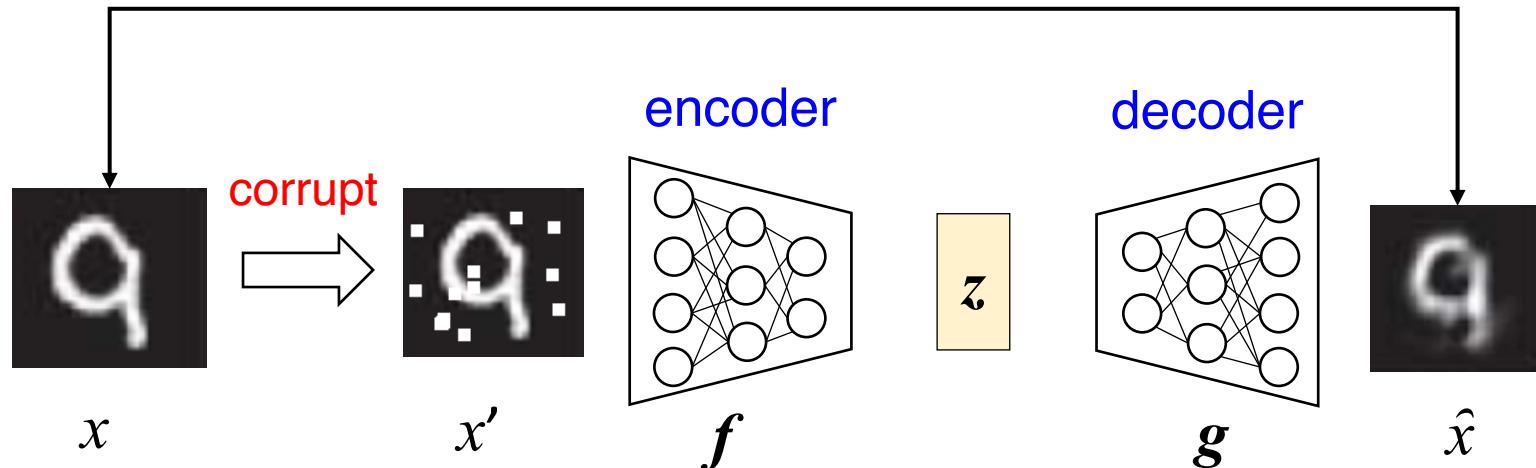
Retrieved using 256 codes



# Denoising Auto-encoder

- An autoencoder that receives a **corrupted** data point as input and is trained to predict the original data point.

As close as possible



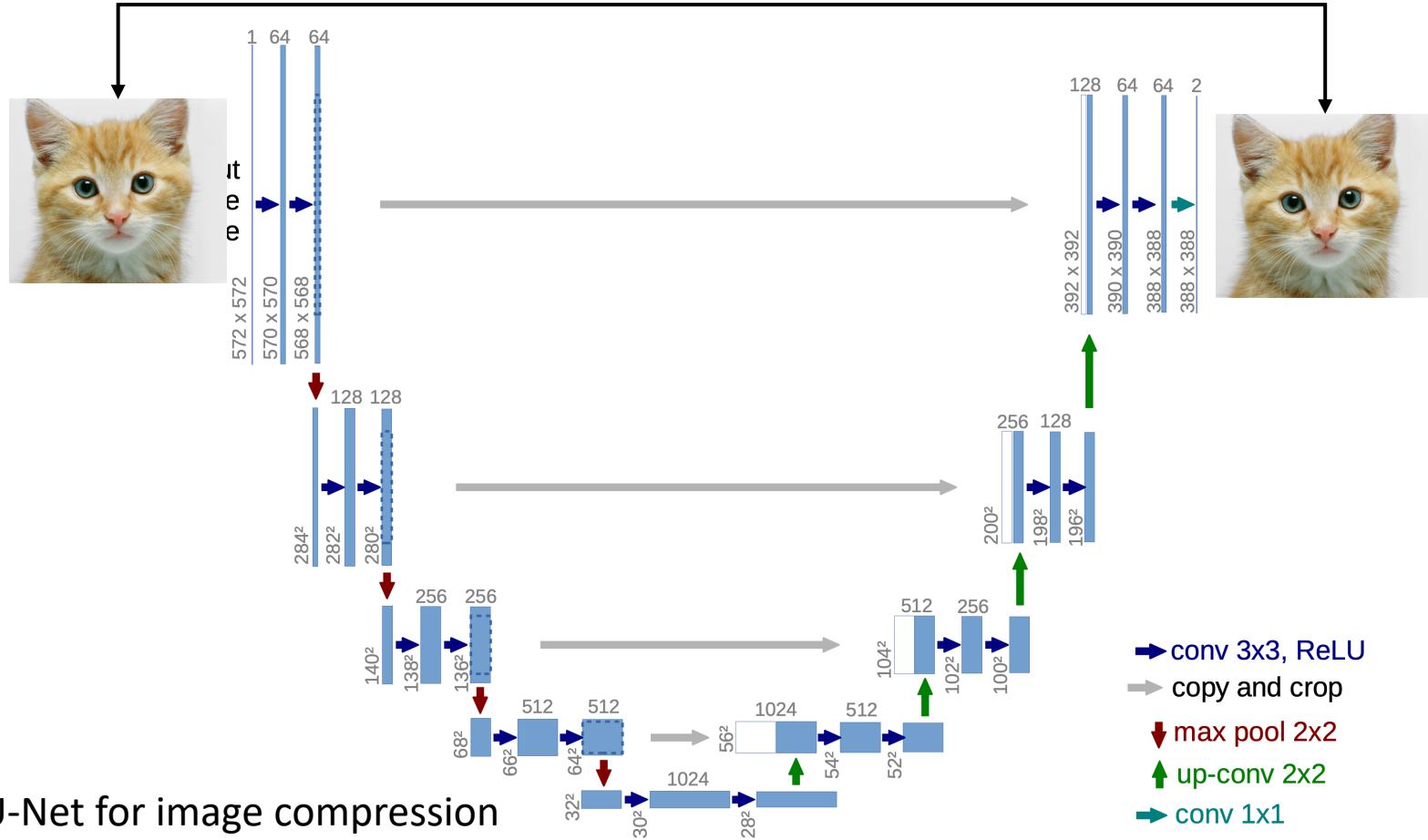
**Applications:** removes noise from data.

Vincent, Pascal, et al. "Extracting and composing robust features with denoising autoencoders." *ICML*, 2008.

# Auto-encoder with CNN?



As close as possible



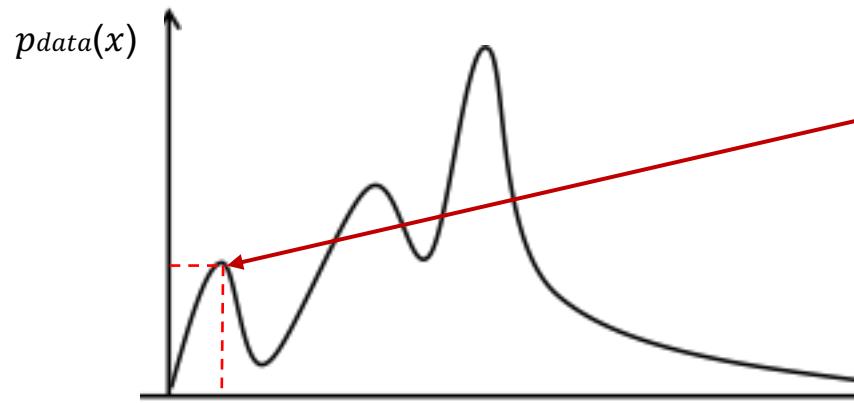
U-Net for image compression



# What's Next?

## Deep Generative Models

Learning to generate data



GAN   VAE   Diffusion

