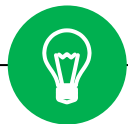


# *Applied Deep Learning*



## Unsupervised Learning



May 25th, 2020 <http://adl.miulab.tw>



國立臺灣大學  
National Taiwan University

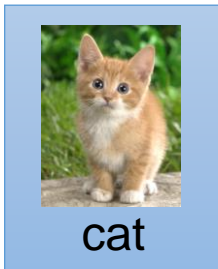
# Introduction

- Big data  $\neq$  Big annotated data
- Machine learning techniques include:
  - Supervised learning (if we have labelled data)
  - Reinforcement learning (if we have an environment for reward)
  - Unsupervised learning (if we do not have labelled data)

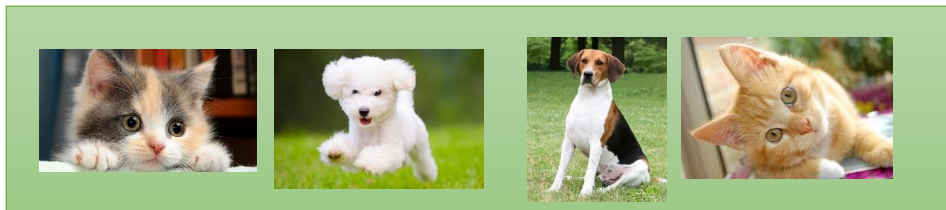
What can we do if there is no sufficient training data?

# 3 Semi-Supervised Learning

Labelled Data



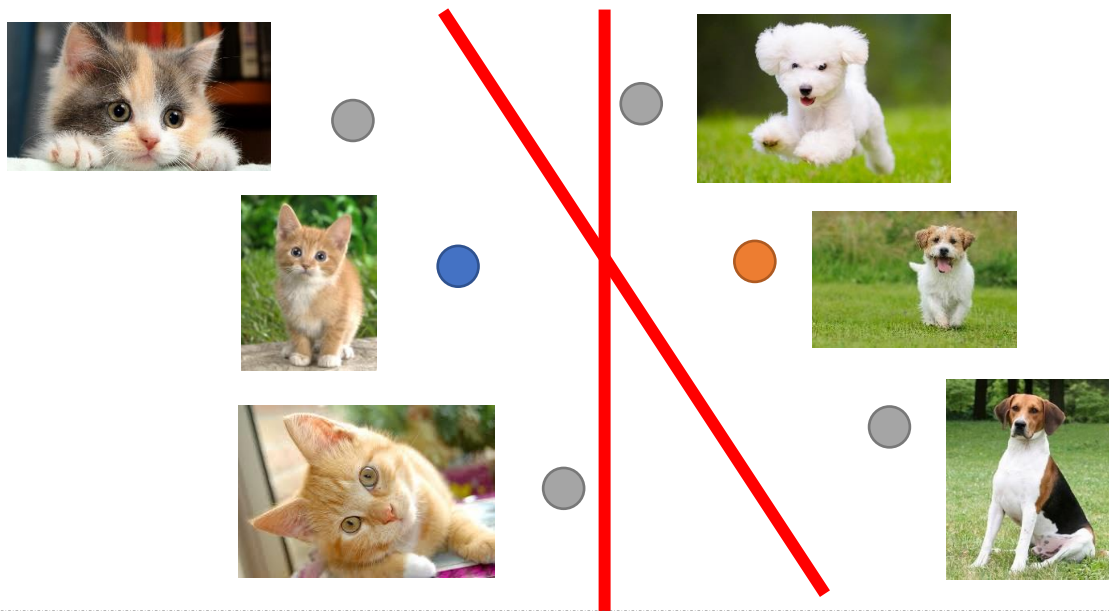
Unlabeled Data



(Image of cats and dogs without labeling)

## 4 Semi-Supervised Learning

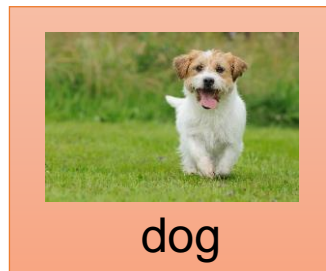
### Why semi-supervised learning helps?



The distribution of the unlabeled data provides some cues

# Transfer Learning

Source Data



Target Data



elephant



elephant



tiger

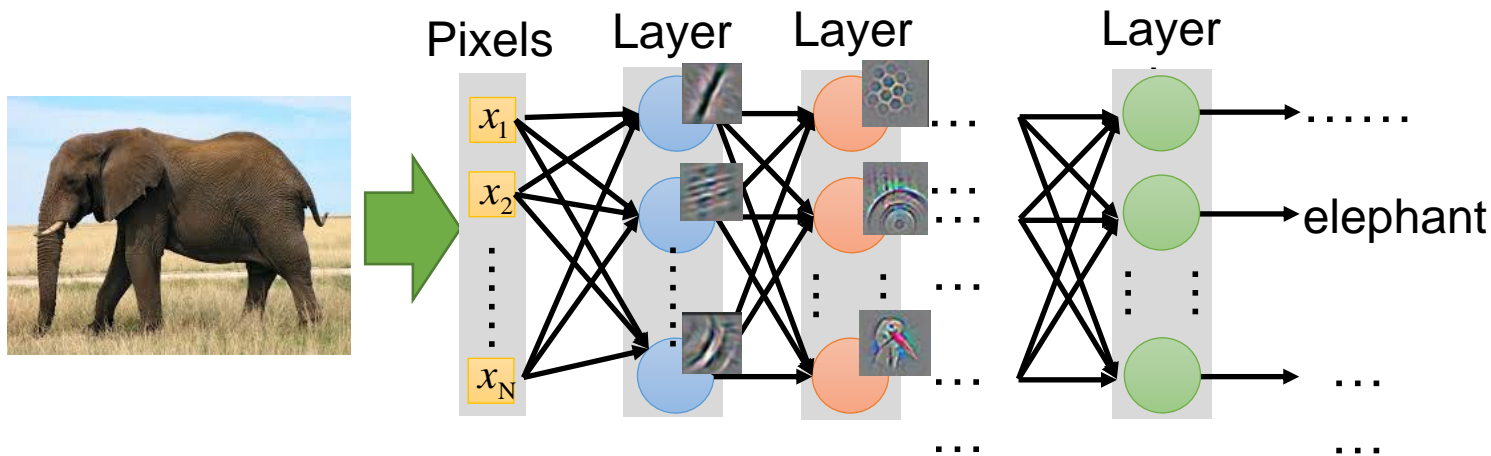


tiger

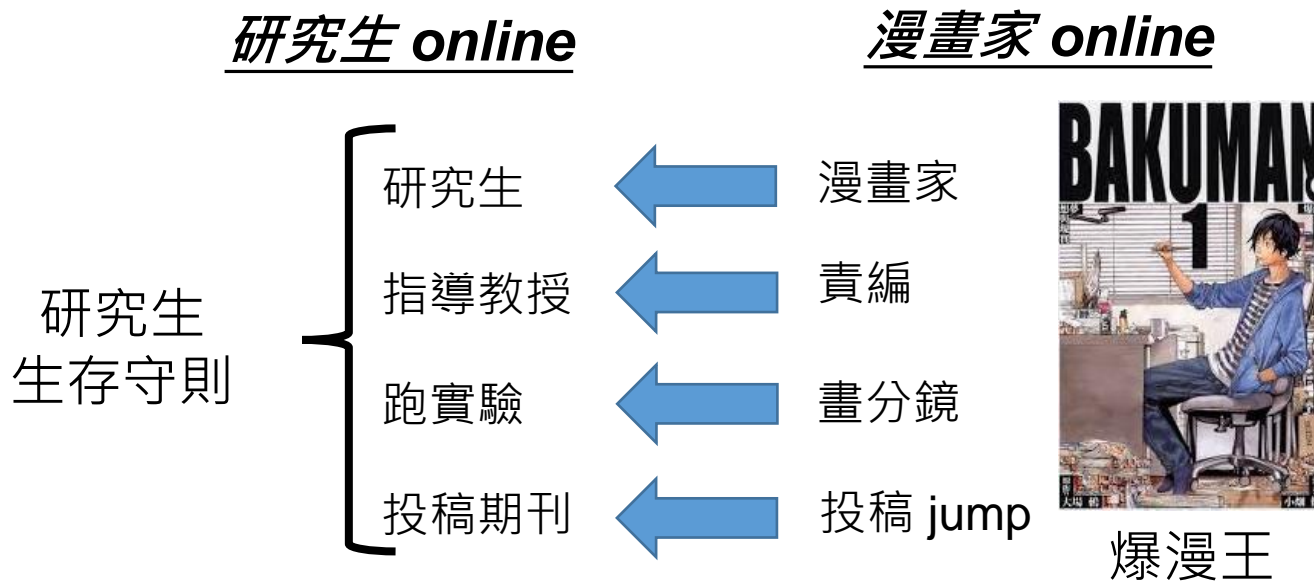
Not related to the task considered

# Transfer Learning

- Widely used on image processing
  - Using sufficient labeled data to learn a CNN
  - Using this CNN as feature extractor



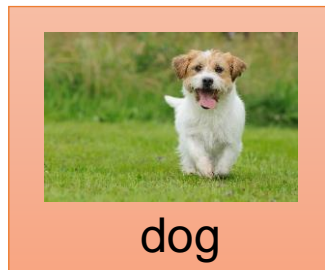
# Transfer Learning Example



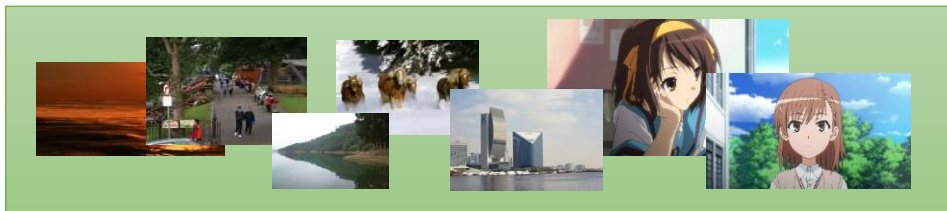
# Self-Taught Learning

- The unlabeled data sometimes is not related to the task

Labelled Data



Unlabeled Data




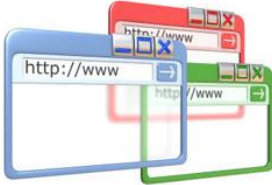
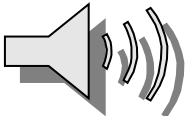



(Just crawl millions of images from the Internet)



# Self-Taught Learning

- The unlabeled data sometimes is not related to the task

	Labelled Data	Unlabeled Data
Digit Recognition	 Digits	 character
Document Classification	 News	 Webpages
Speech Recognition	 Taiwanese	 English Chinese .....

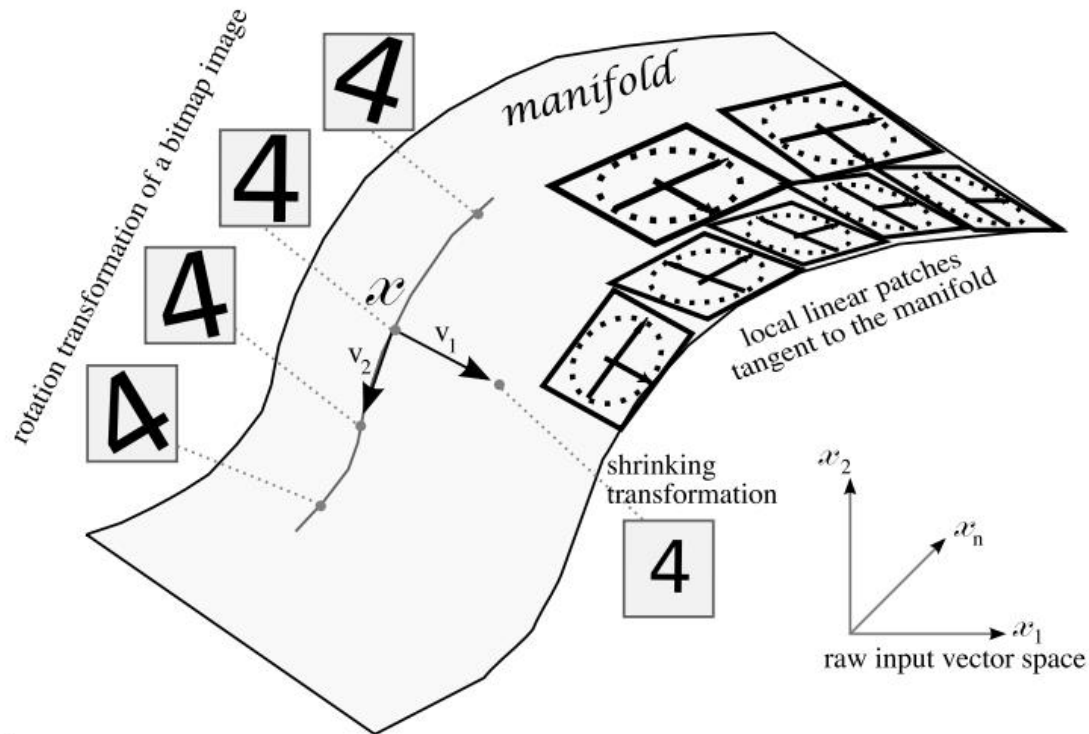
Why can we use unlabeled and unrelated data to help our tasks?

# Self-Taught Learning

- How does self-taught learning work?
- Why does unlabeled and unrelated data help the tasks?

Finding latent factors that control the observations

# Latent Factors for Handwritten Digits



# 12 Latent Factors for Documents

## Topics

gene	0.04
dna	0.02
genetic	0.01
...	

life	0.02
evolve	0.01
organism	0.01
...	

brain	0.04
neuron	0.02
nerve	0.01
...	

data	0.02
number	0.02
computer	0.01
...	

## Documents

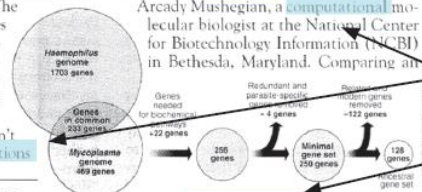
### Seeking Life's Bare (Genetic) Necessities

COLD SPRING HARBOR, NEW YORK—How many **genes** does an **organism** need to **survive**? Last week at the genome meeting here,\* two genome researchers with radically different approaches presented complementary views of the basic genes needed for **life**. One research team, using **computer** analyses to compare known **genomes**, concluded that today's **organisms** can be sustained with just 250 genes, and that the earliest life forms required a mere 128 **genes**. The other researcher mapped genes in a simple parasite and estimated that for this organism, 800 genes are plenty to do the job—but that anything short of 100 wouldn't be enough.

Although the numbers don't match precisely, those **predictions**

"are not all that far apart," especially in comparison to the 75,000 **genes** in the human genome, notes Siv Andersson at Uppsala University in Sweden, who arrived at the 800 number. But coming up with a consensus answer may be more than just a **genetic numbers game**, particularly if more and more **genomes** are completely mapped and sequenced. "It may be a way of organizing any newly **sequenced genome**," explains

Arcady Mushegian, a **computational** molecular biologist at the National Center for Biotechnology Information (NCBI) in Bethesda, Maryland. Comparing an

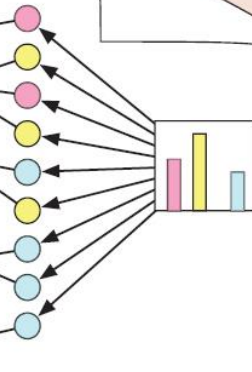


\* Genome Mapping and Sequencing. Cold Spring Harbor, New York, May 8 to 12.

Stripping down. **Computer analysis** yields an estimate of the minimum modern and ancient genomes.

SCIENCE • VOL. 272 • 24 MAY 1996











## Topic proportions and assignments



# Latent Factors for Recommendation System

單純呆

傲嬌

A	 	 	 	 
B				
C		 		

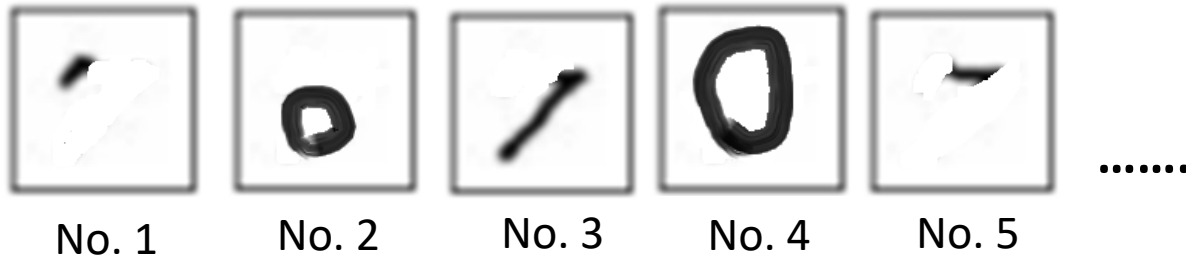
# Latent Factor Exploitation

## Handwritten digits



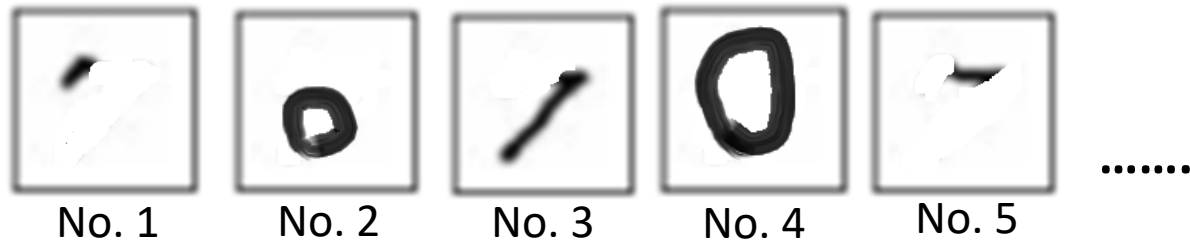
The handwritten images are composed of **strokes**

### Strokes (Latent Factors)



# Latent Factor Exploitation

## Strokes (Latent Factors)



$$\begin{array}{c}
 28 \\
 \begin{array}{|c|} \hline \text{7} \\ \hline \end{array}
 \end{array}
 =
 \begin{array}{c}
 \text{No. 1} \\
 \begin{array}{|c|} \hline \text{stroke} \\ \hline \end{array}
 \end{array}
 +
 \begin{array}{c}
 \text{No. 3} \\
 \begin{array}{|c|} \hline \text{stroke} \\ \hline \end{array}
 \end{array}
 +
 \begin{array}{c}
 \text{No. 5} \\
 \begin{array}{|c|} \hline \text{stroke} \\ \hline \end{array}
 \end{array}$$

Represented by  
 28 X 28 = 784 pixels

[1 0 1 0 1 0 .....]  
 (simpler representation)

16

# Autoencoder

Representation Learning

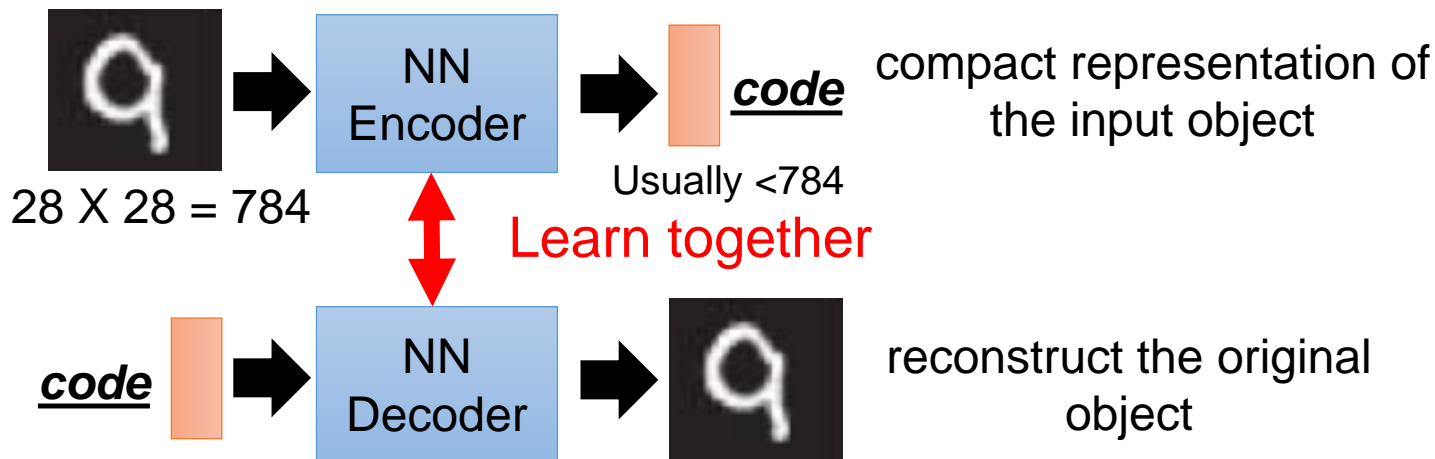


# Autoencoder

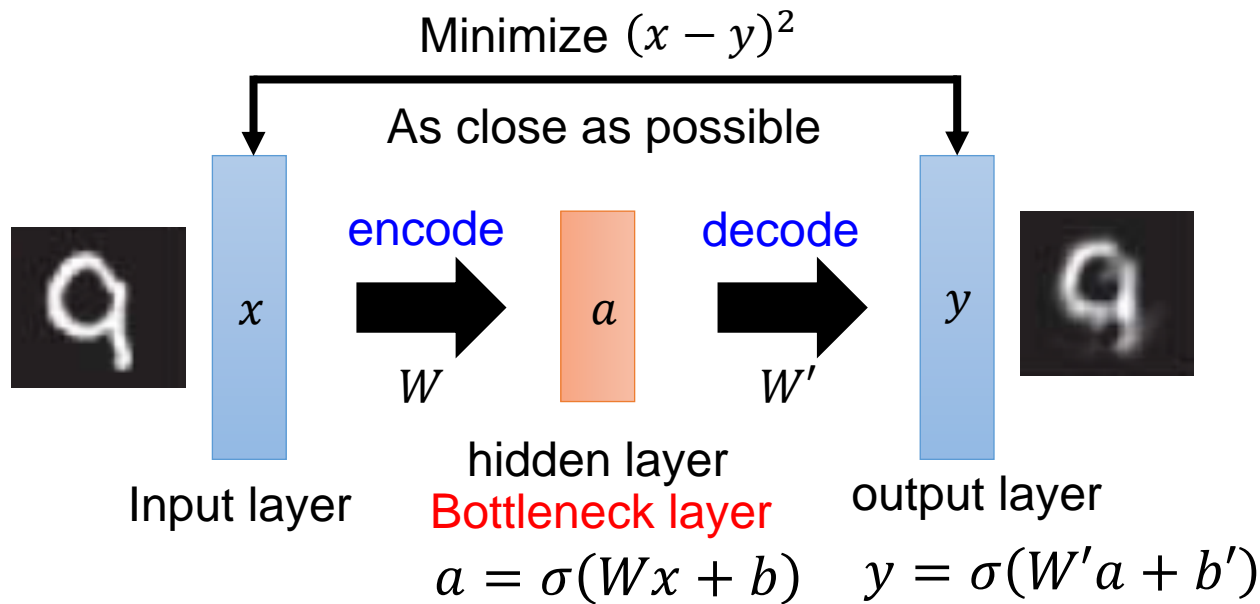


- Represent a digit using 28 X 28 dimensions
- Not all 28 X 28 images are digits

Idea: represent the images of digits in a more compact way



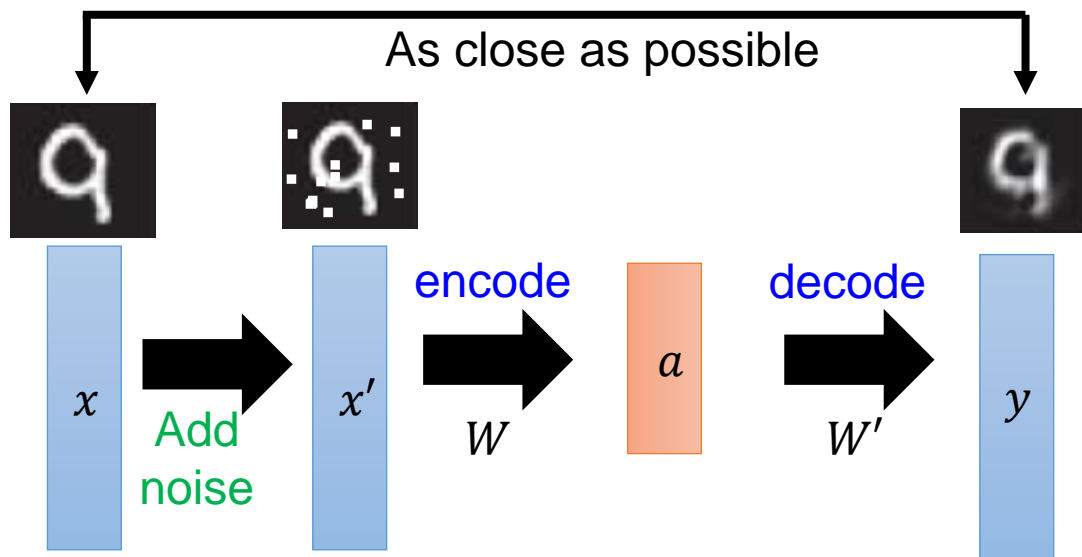
# Autoencoder



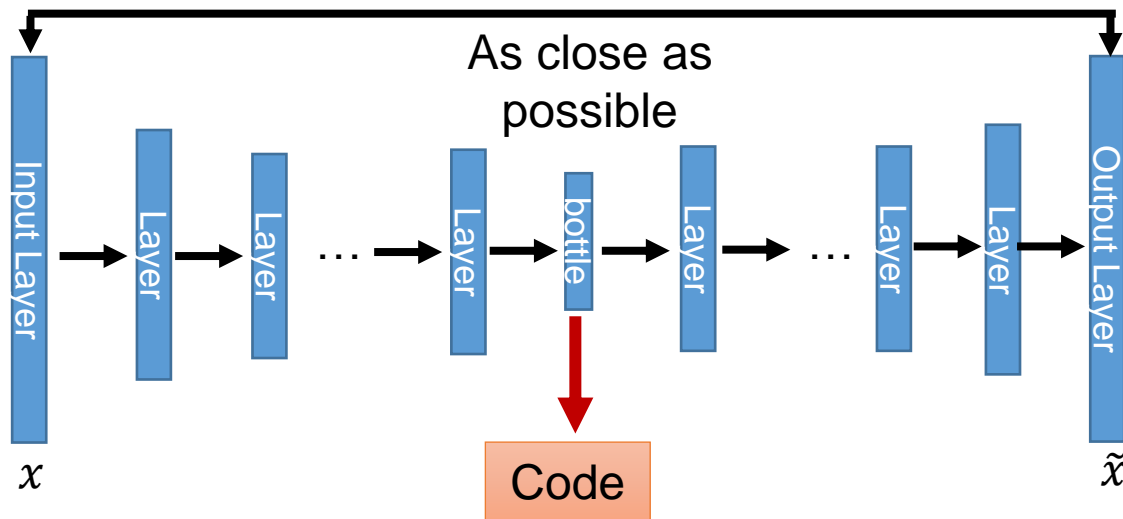
Output of the hidden layer is the code

# Autoencoder

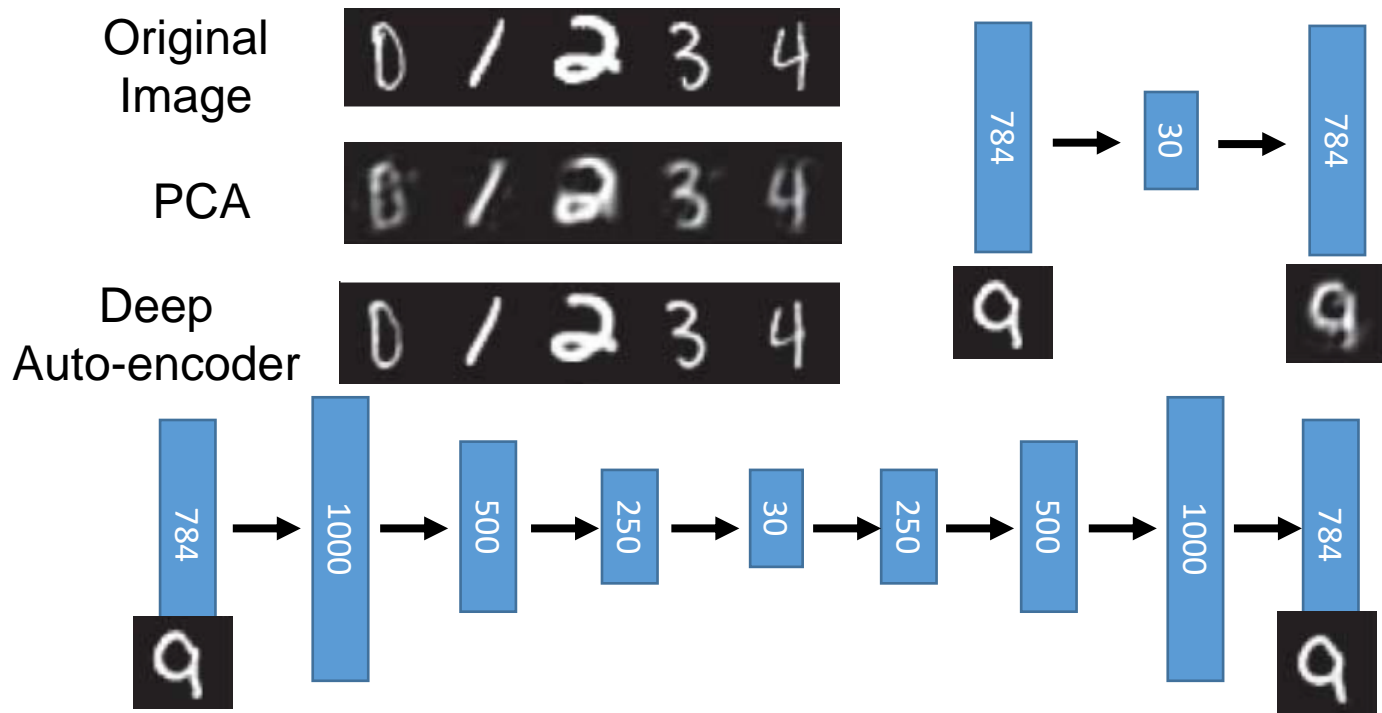
## De-noising auto-encoder



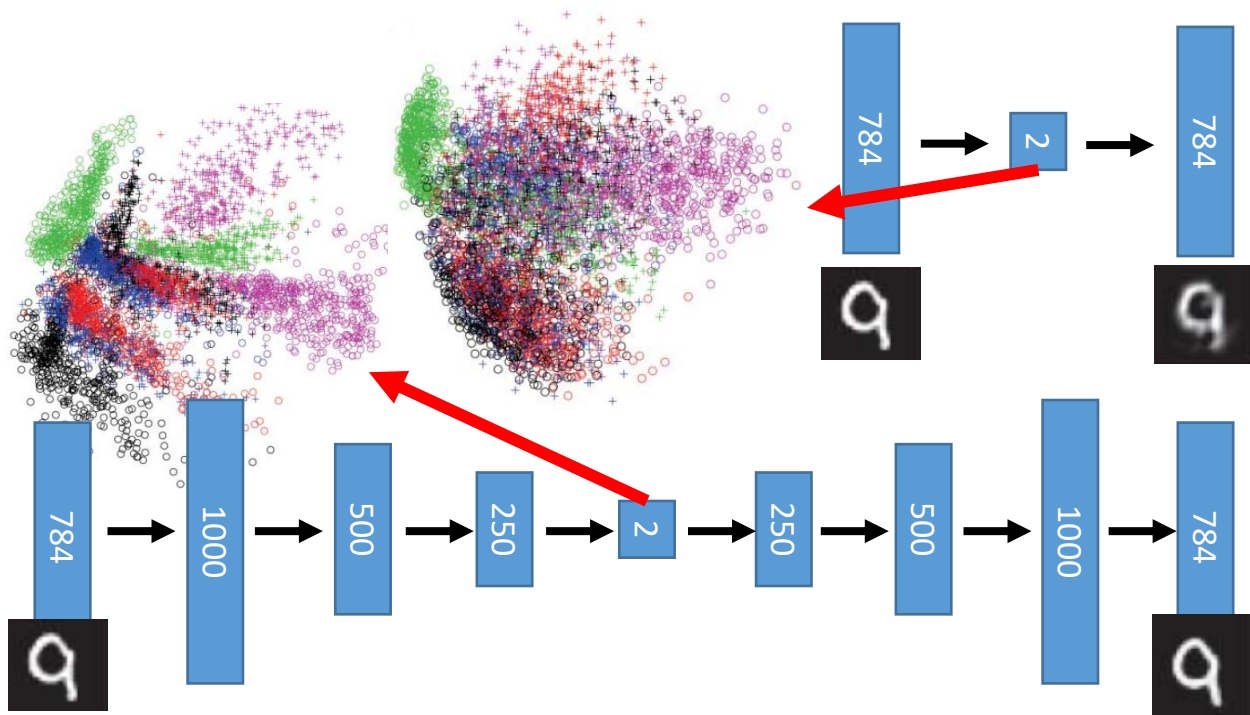
# Deep Autoencoder



# Deep Autoencoder

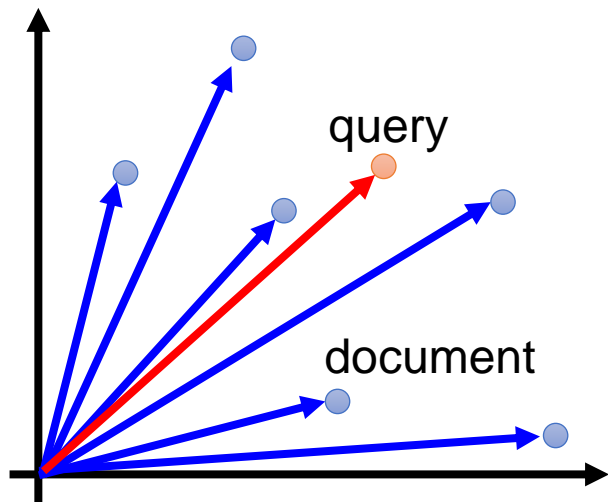


# Feature Representation



## 23 Auto-encoder – Text Retrieval

### Vector Space Model

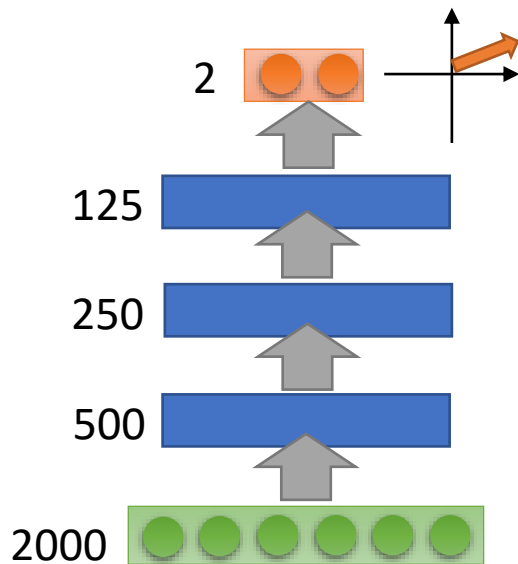


### Bag-of-words word string: "This is an apple"

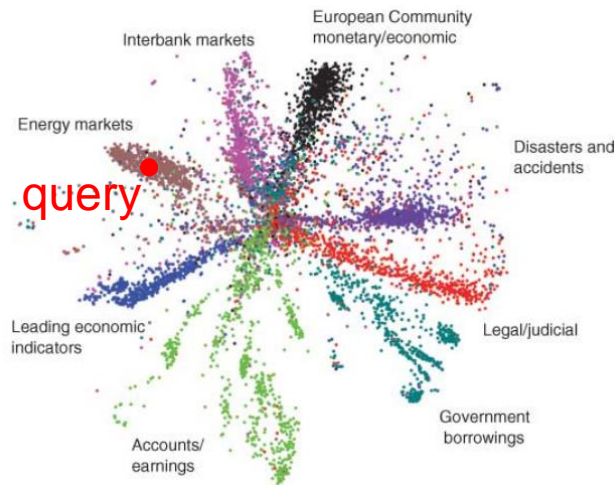
this	●	1
is	●	1
a	●	0
an	●	1
apple	●	1
pen	●	0
	⋮	

Semantics are not considered

# Autoencoder – Text Retrieval



Bag-of-words (document or query)



The documents talking about the same thing will have close code



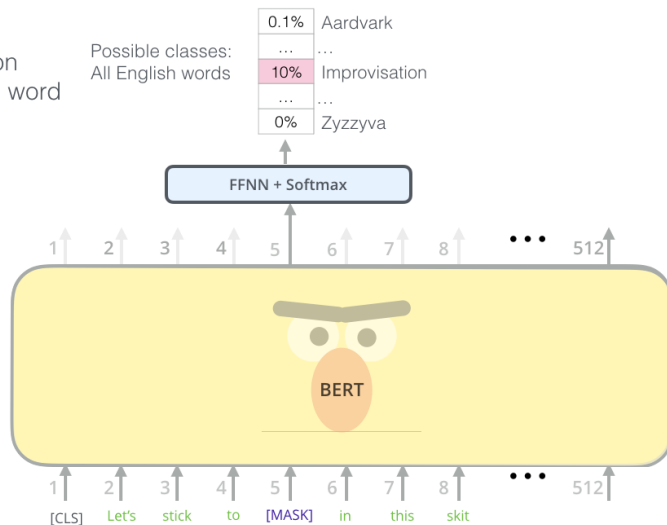
# Auto-Encoding (AE)

## Objective: reconstructing $\bar{x}$ from $\hat{x}$

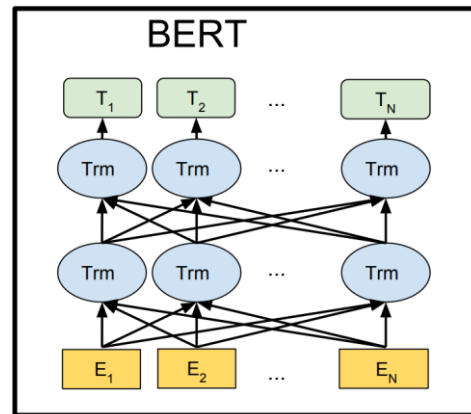
$$\max_{\theta} \log p_{\theta}(\bar{x} | \hat{x}) \approx \sum_{t=1}^T m_t \log p_{\theta}(x_t | \hat{x}) = \sum_{t=1}^T m_t \log \frac{\exp(H_{\theta}(\hat{x})_t^{\top} e(x_t))}{\sum_{x'} \exp(H_{\theta}(\hat{x})_t^{\top} e(x'))}$$

### dimension reduction or denoising (masked LM)

Use the output of the masked word's position to predict the masked word



Randomly mask  
15% of tokens

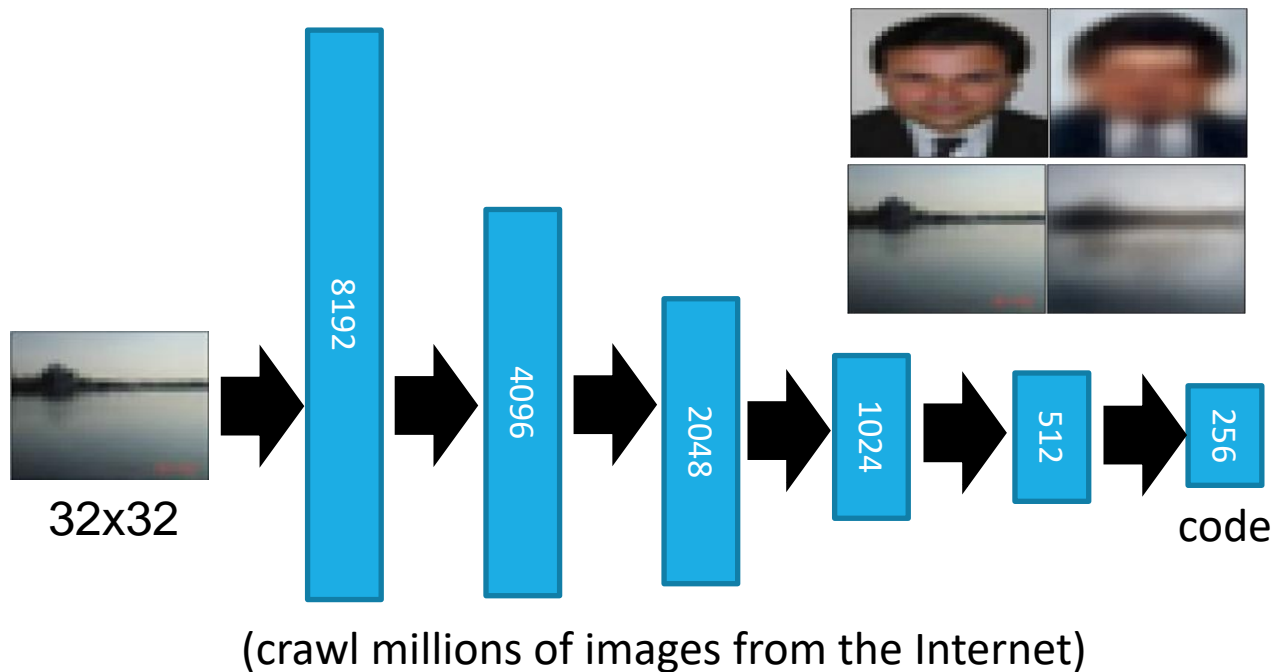


# Autoencoder – Similar Image Retrieval

- Retrieved using Euclidean distance in pixel intensity space

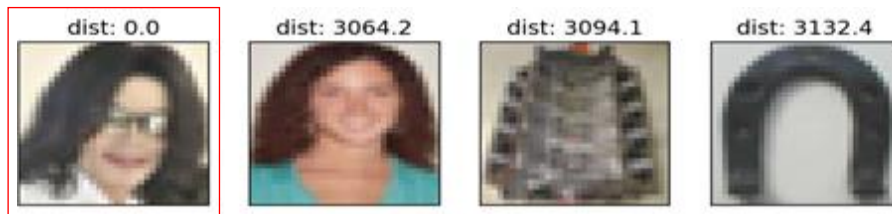


# Autoencoder – Similar Image Retrieval



# Autoencoder – Similar Image Retrieval

- Images retrieved using Euclidean distance in pixel intensity space



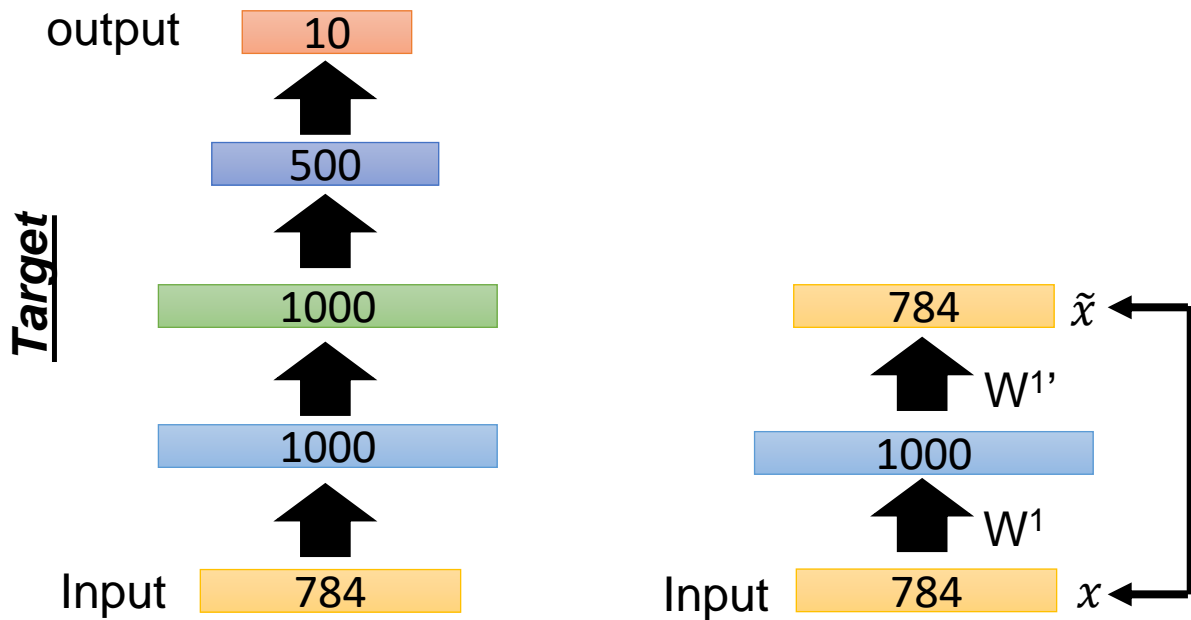
- Images retrieved using 256 codes



Learning the useful latent factors

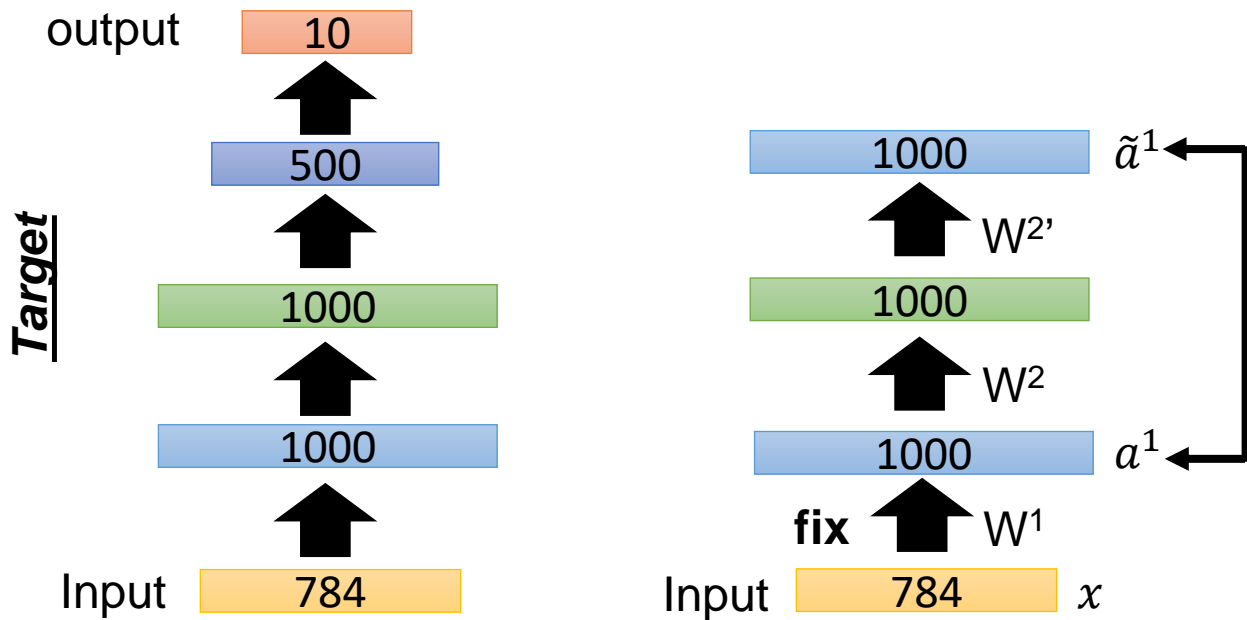
# Autoencoder for DNN Pre-Training

- Greedy layer-wise pre-training *again*



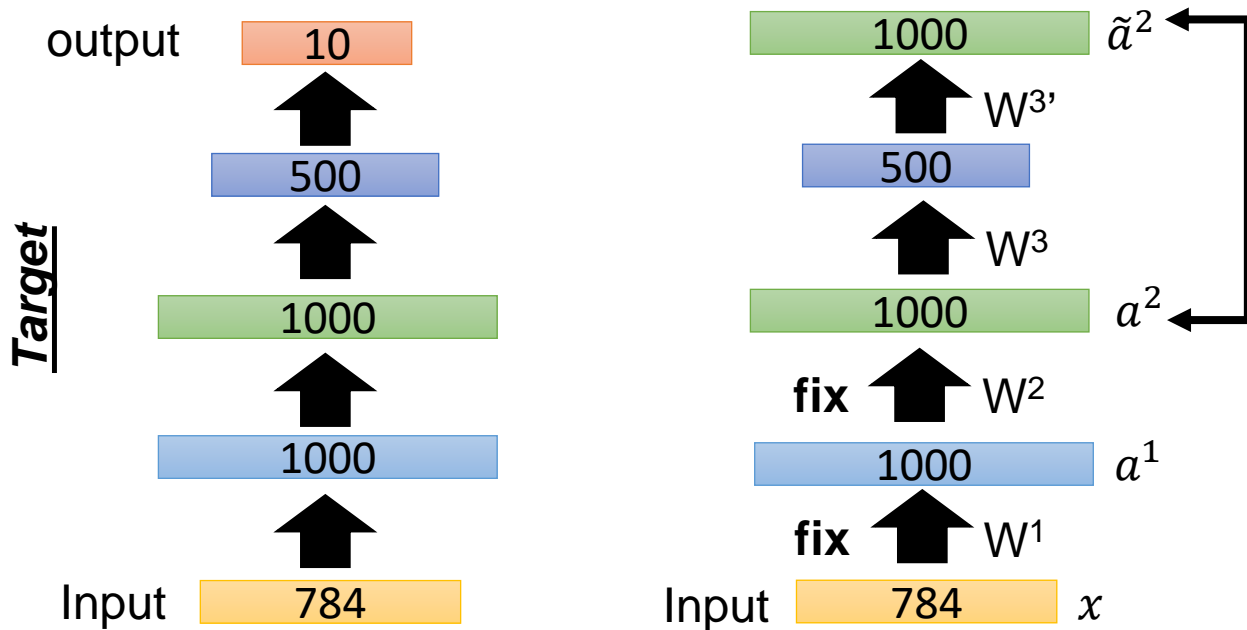
# Autoencoder for DNN Pre-Training

- Greedy layer-wise pre-training *again*



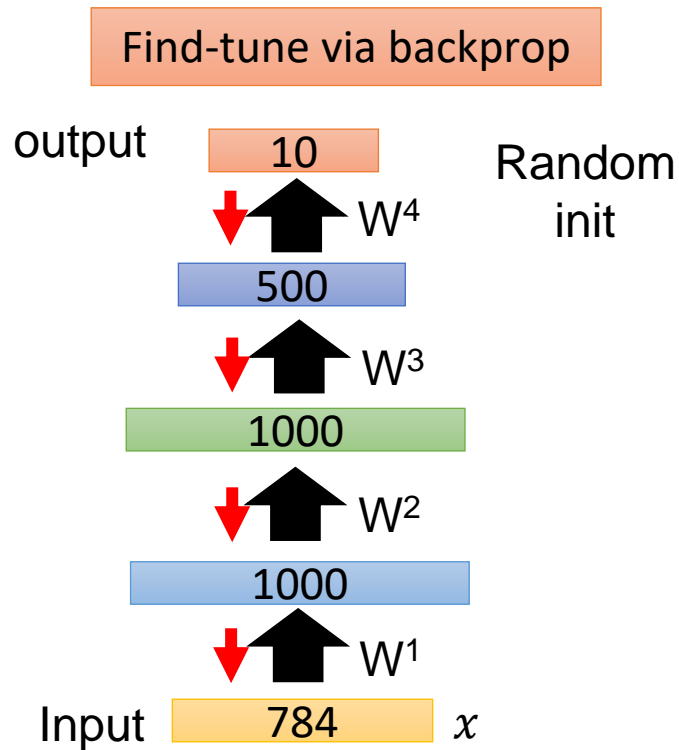
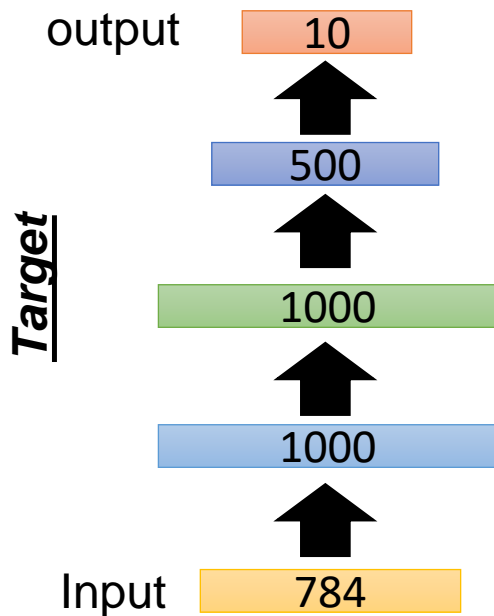
# Autoencoder for DNN Pre-Training

- Greedy layer-wise pre-training *again*



# Autoencoder for DNN Pre-Training

- Greedy layer-wise pre-training *again*



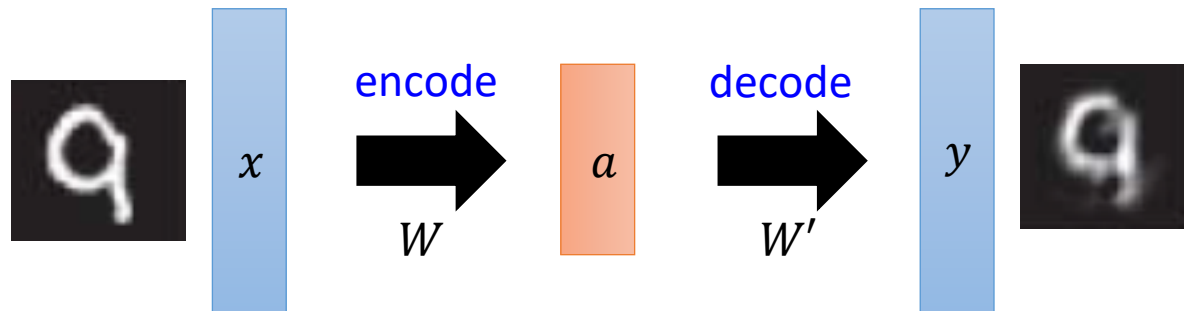


33

# Variational Autoencoder

Representation Learning and Generation

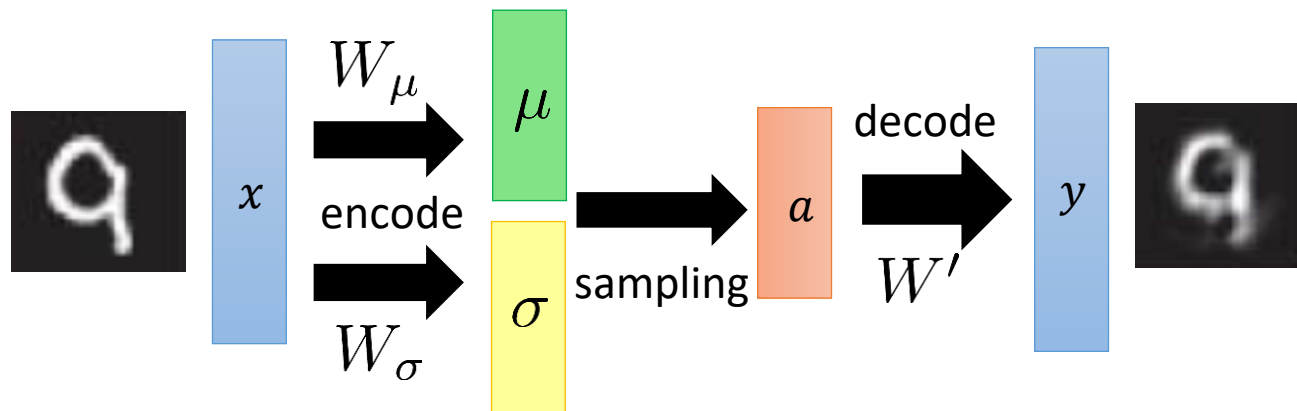
# Generation from Latent Codes



How can we set a latent code for generation?

# Latent Code Distribution Constraints

- Constrain the data distribution for learned latent codes
- Generate the latent code via a prior distribution



# Reconstruction

AE



VAE



37

# Distant Supervision

Representation Learning by Weak Labels

# Convolutional Deep Structured Semantic Models (CDSSM/DSSM)

Semantic Layer:  $y$

Semantic Projection Matrix:  $W_s$

Max Pooling Layer:  $I_m$

Max Pooling Operation

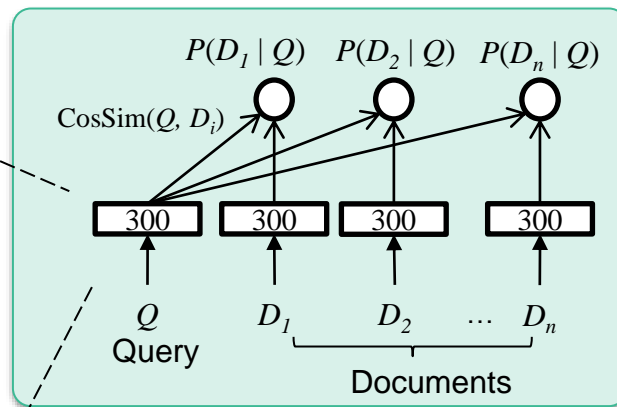
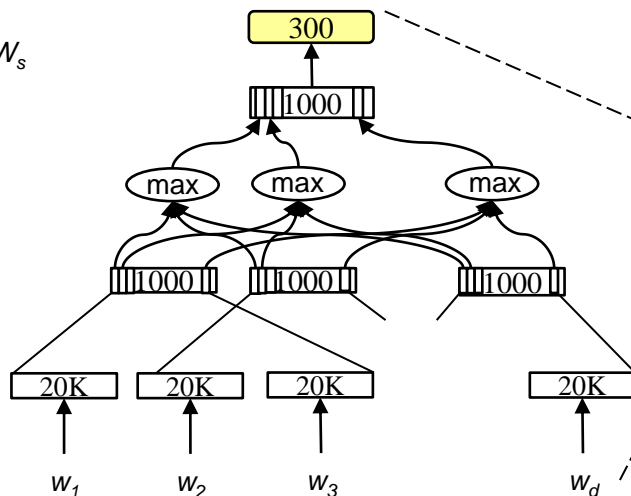
Convolutional Layer:  $I_c$

Convolution Matrix:  $W_c$

Word Hashing Layer:  $I_h$

Word Hashing Matrix:  $W_h$

Word Sequence:  $x$



$$P(D | Q) = \frac{\exp(\text{CosSim}(Q, D))}{\sum_{D'} \exp(\text{CosSim}(Q, D'))}$$

$$\Lambda(\theta) = \log \prod_{(Q, D^+)} P(D^+ | Q)$$

maximizes the likelihood of clicked documents given queries

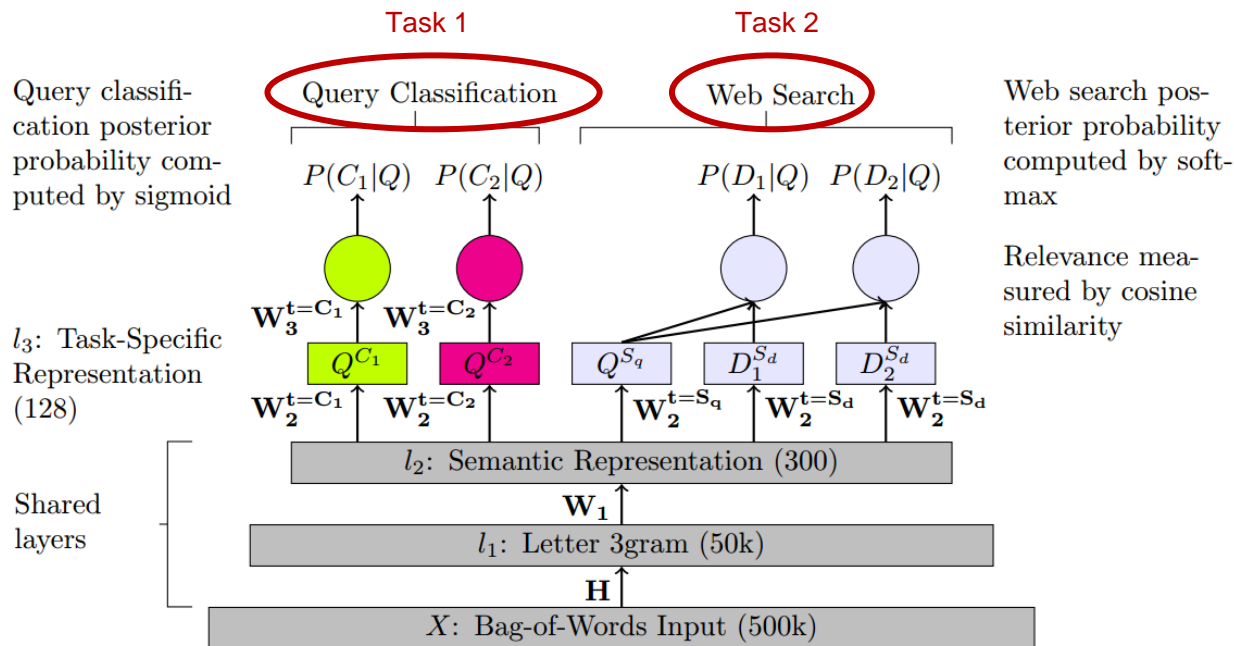
Semantically related documents are close to the query in the encoded space

39

# Multi-Task Learning

Representation Learning by Different Tasks

# Task-Shared Representation

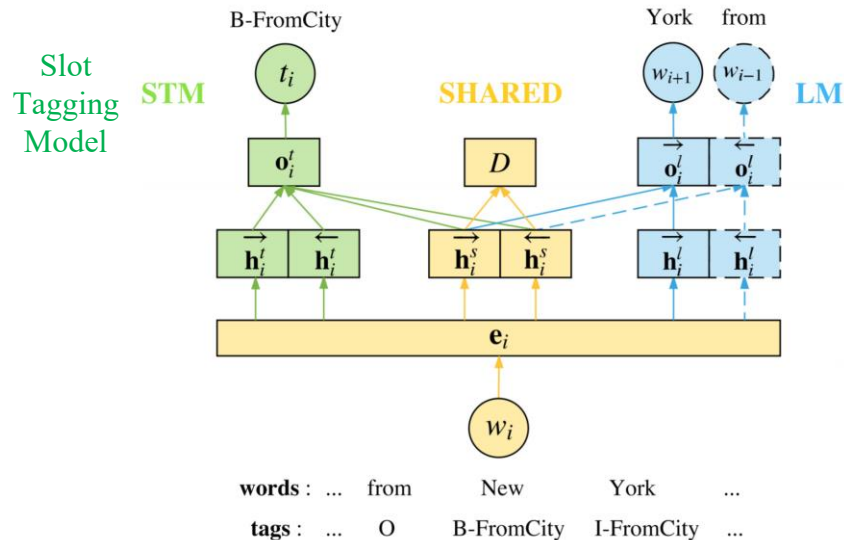


The latent factors can be learned by different tasks



# Semi-Supervised Multi-Task SLU (Lan et al., 2018)

- Idea: language understanding objective can enhance other tasks



## Algorithm 1: Adversarial Multi-task Learning for SLU

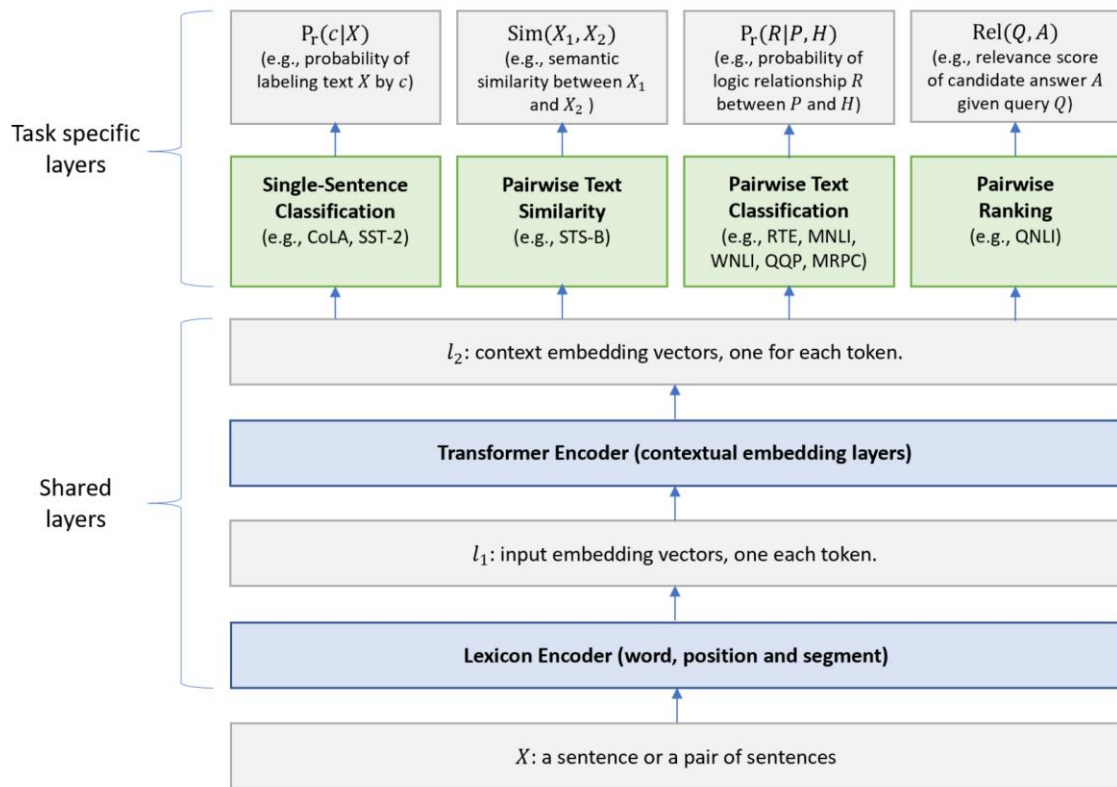
**Input** : Labeled training data  $\{(\mathbf{w}^l, \mathbf{t}^l)\}$   
 Unlabeled data  $\{\mathbf{w}^u\}$

**Output**: Adversarially enhanced slot tagging model

- 1 Initialize parameters  $\{\theta^s, \theta^t, \theta^l, \theta^d\}$  randomly.
- 2 **repeat**
  - /\* Sample from  $\{(\mathbf{w}^l, \mathbf{t}^l)\}$  \*/
  - 3 Train the STM and shared model by Eq.(8).
  - 4 Train the task discriminator and the shared model by Eq.(6) or Eq.(7) as slot tagging task ( $y = 1$ ).
  - /\* Sample from  $\{\mathbf{w}^l\}$  and  $\{\mathbf{w}^u\}$  \*/
  - 5 Train the LM and shared models by Eq.(9) (and Eq.(10) for BLM).
  - 6 Train the task discriminator and the shared model by Eq.(6) or Eq.(7) as LM task ( $y = 0$ ).
- 7 **until** convergence;

BLM exploits the *unsupervised knowledge*, the *shared-private framework* and *adversarial training* make the slot tagging model more generalized

# MT-DNN (Liu et al., 2019)



## Algorithm 1: Training a MT-DNN model.

```

Initialize model parameters  $\Theta$  randomly.
Pre-train the shared layers (i.e., the lexicon
encoder and the transformer encoder).
Set the max number of epoch:  $epoch_{max}$ .
//Prepare the data for  $T$  tasks.
for  $t$  in  $1, 2, \dots, T$  do
    | Pack the dataset  $t$  into mini-batch:  $D_t$ .
end
for  $epoch$  in  $1, 2, \dots, epoch_{max}$  do
    1. Merge all the datasets:
         $D = D_1 \cup D_2 \dots \cup D_T$ 
    2. Shuffle  $D$ 
    for  $b_t$  in  $D$  do
        // $b_t$  is a mini-batch of task  $t$ .
    3. Compute loss :  $L(\Theta)$ 
         $L(\Theta)$  = Eq. 6 for classification
         $L(\Theta)$  = Eq. 7 for regression
         $L(\Theta)$  = Eq. 8 for ranking
    4. Compute gradient:  $\nabla(\Theta)$ 
    5. Update model:  $\Theta = \Theta - \epsilon \nabla(\Theta)$ 
    end
end
  
```

# Concluding Remarks

- Labeling data is expensive, but we have large unlabeled data
- Autoencoder
  - exploits unlabeled data to learn latent factors as representations
  - learned representations can be transfer to other tasks
- Distant Labels / Labels from Other Tasks
  - learn the representations that are useful for other tasks
  - learned representations may be also useful for the target task