



# 人工智能实践

# Artificial Intelligence Practice

*DCS3015 Autumn 2022*

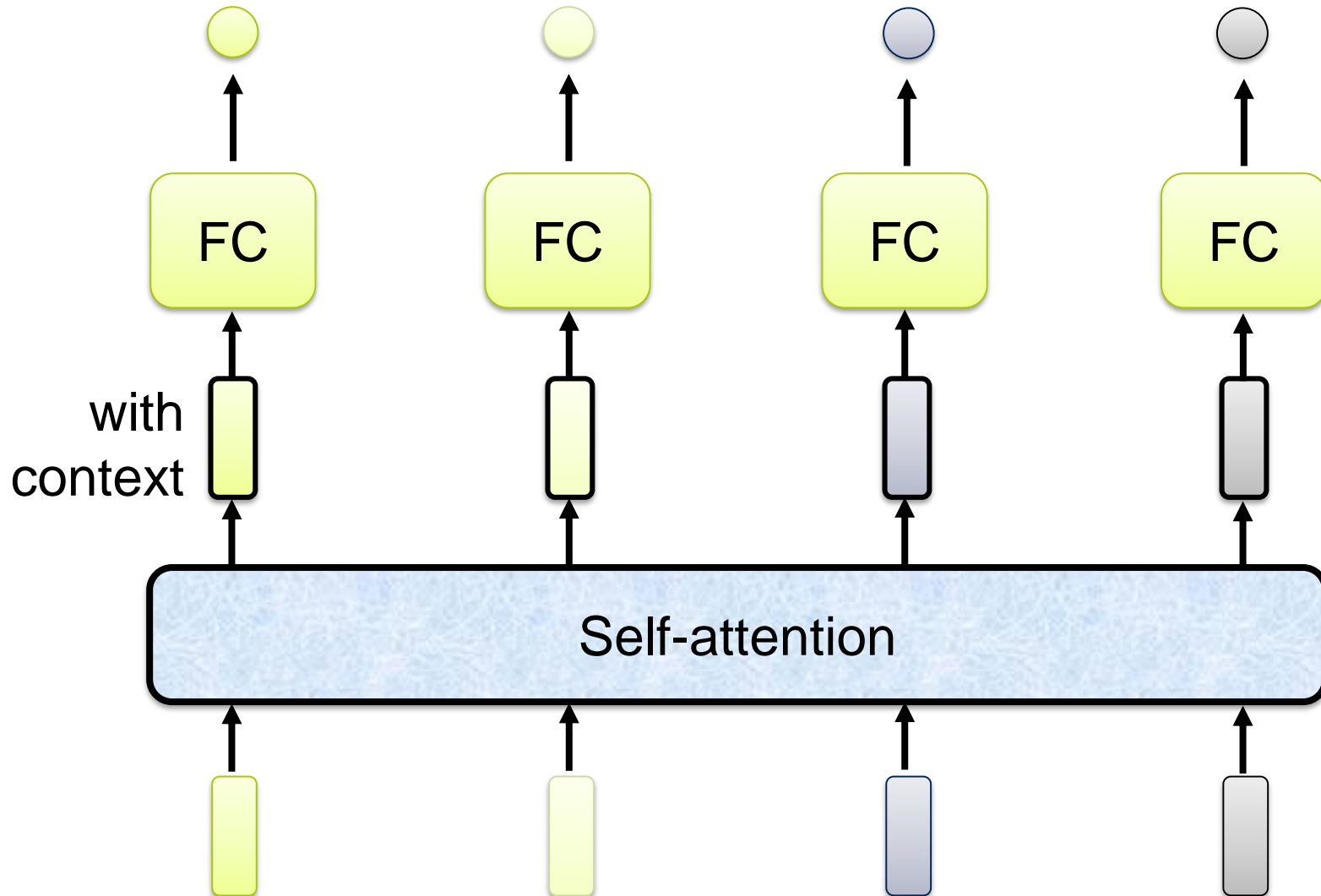
Chao Yu (余超)

School of Computer Science and Engineering  
Sun Yat-Sen University

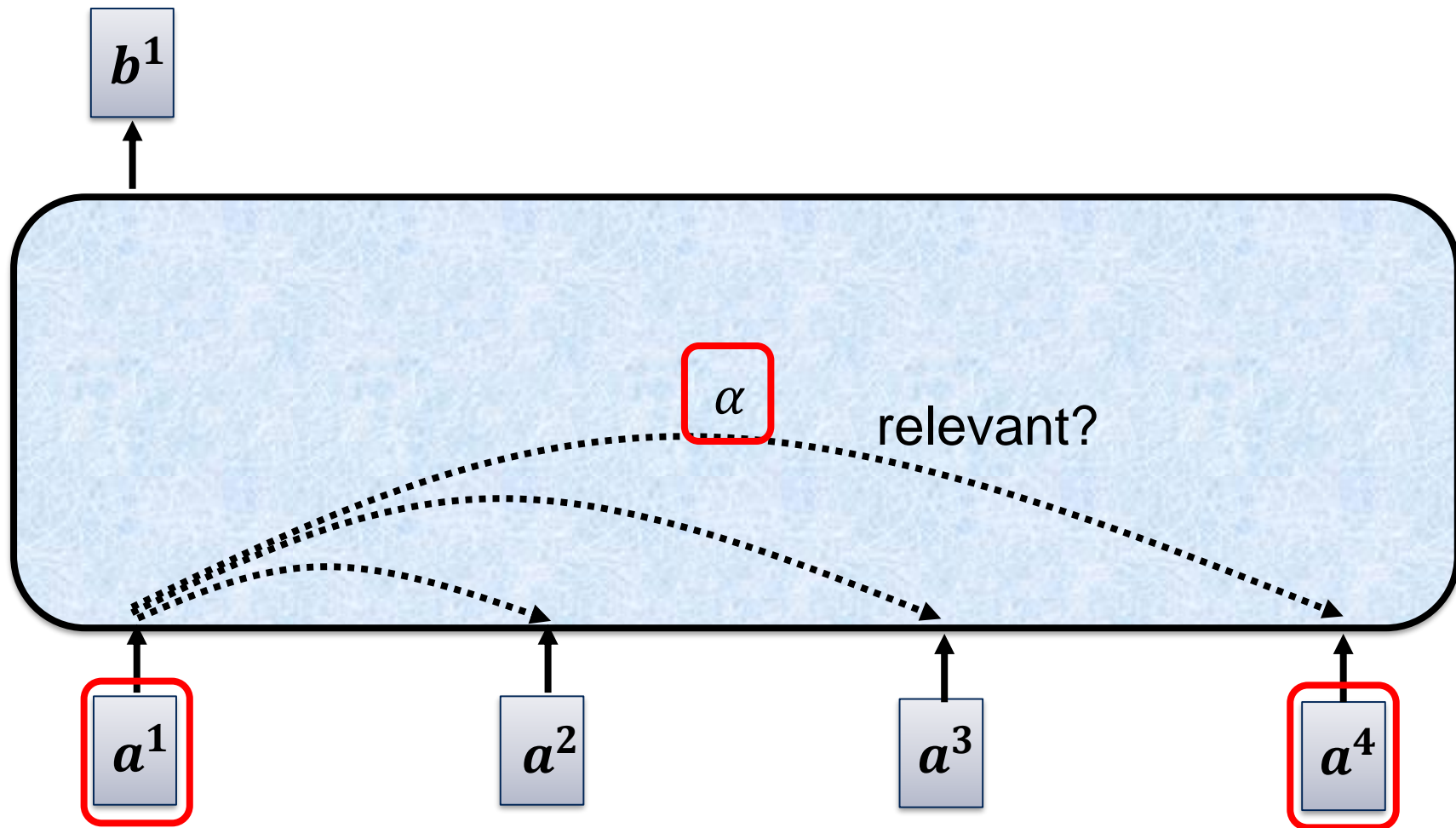


# Lecture 4: 机器学习-3

# Attention

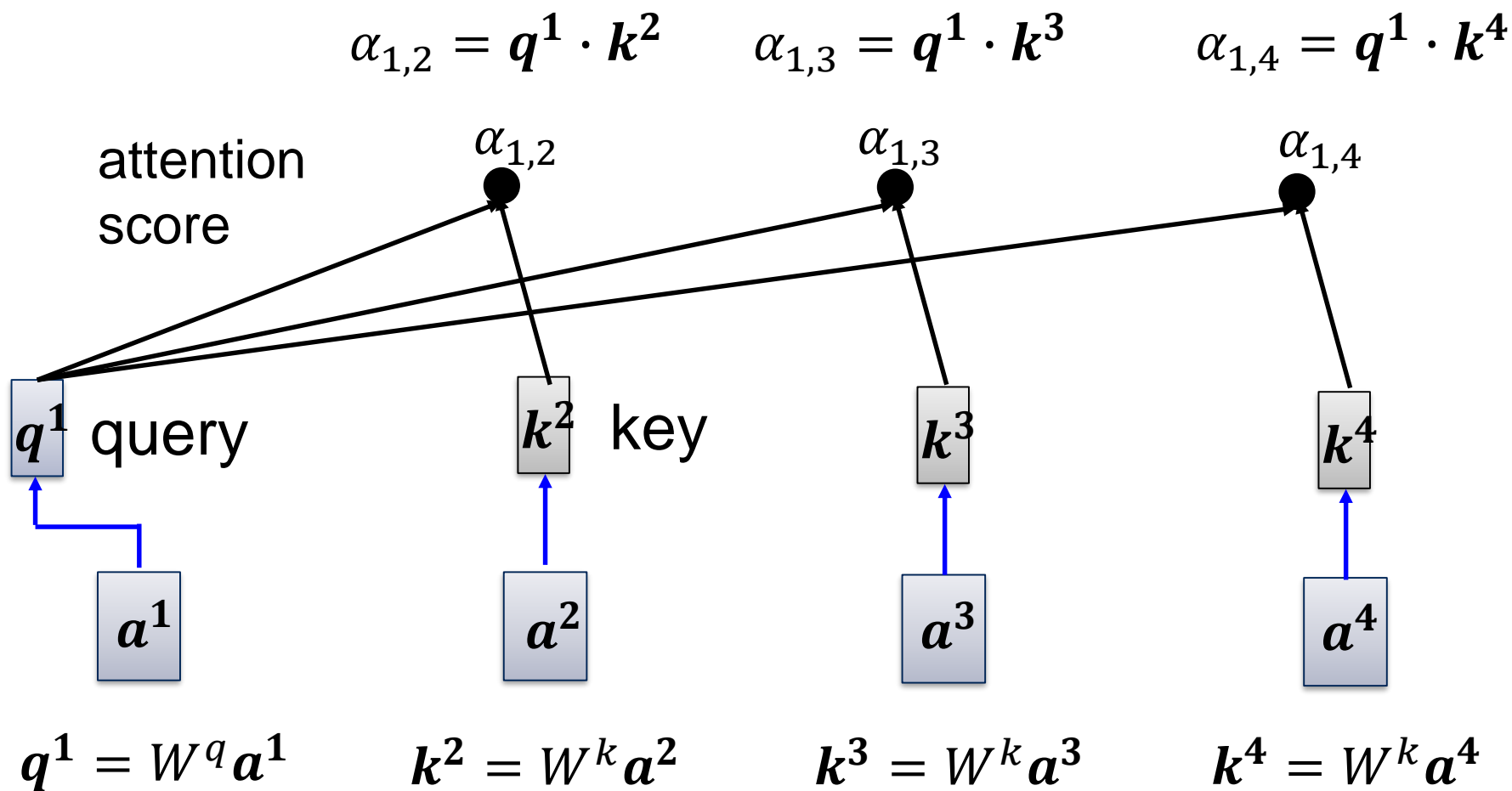


# Attention



Find the relevant vectors in a sequence

# Attention

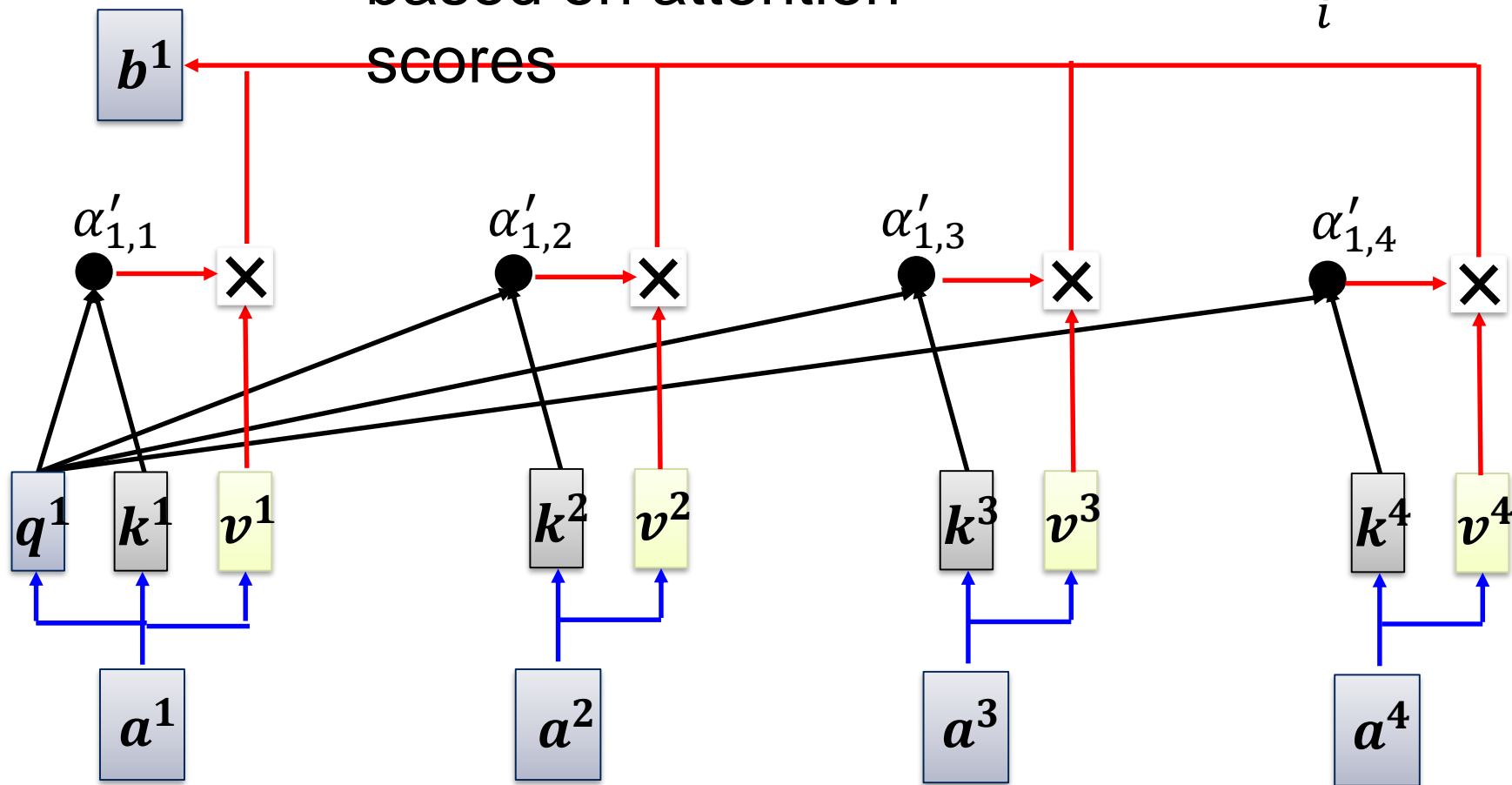


# Attention



Extract information  
based on attention  
scores

$$b^1 = \sum_i \alpha'_{1,i} v^i$$



$$v^1 = W^v a^1$$

$$v^2 = W^v a^2$$

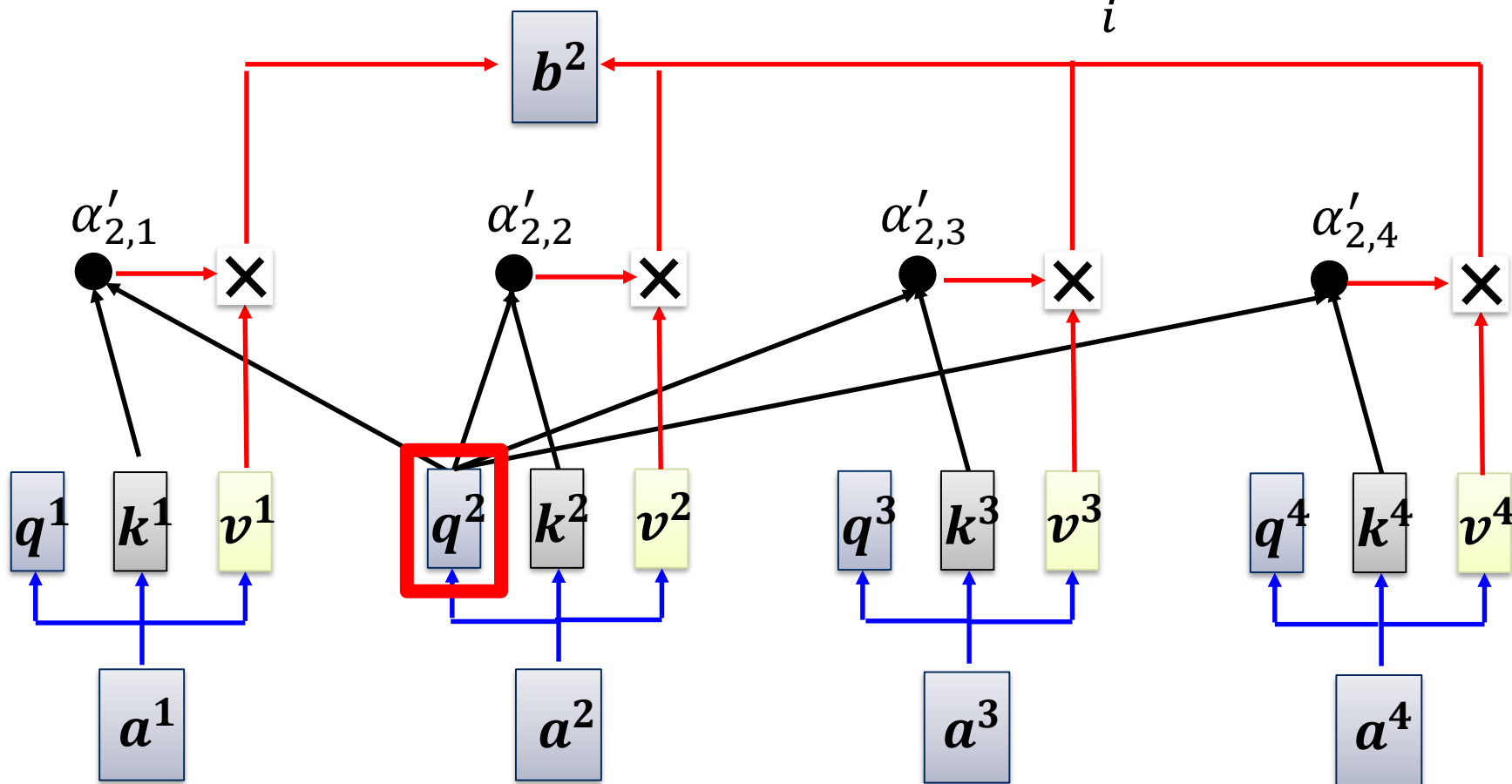
$$v^3 = W^v a^3$$

$$v^4 = W^v a^4$$

# Attention



$$b^2 = \sum_i \alpha'_{2,i} v^i$$



# Attention

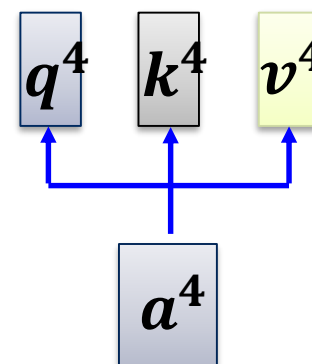
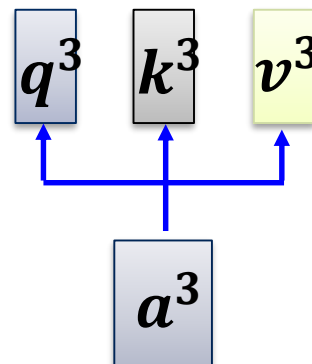
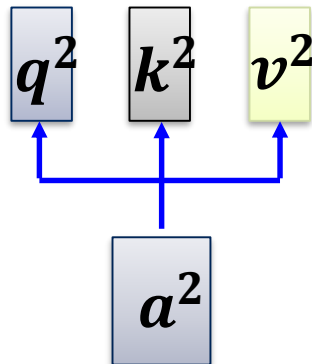
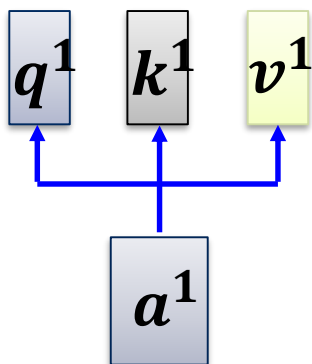


## Self-attention

$$q^i = W^q a^i \quad \begin{matrix} q^1 & q^2 & q^3 & q^4 \\ \hline Q \end{matrix} = \begin{matrix} W^q & \begin{matrix} a^1 & a^2 & a^3 & a^4 \\ \hline I \end{matrix} \end{matrix}$$

$$k^i = W^k a^i \quad \begin{matrix} k^1 & k^2 & k^3 & k^4 \\ \hline K \end{matrix} = \begin{matrix} W^k & \begin{matrix} a^1 & a^2 & a^3 & a^4 \\ \hline I \end{matrix} \end{matrix}$$

$$v^i = W^v a^i \quad \begin{matrix} v^1 & v^2 & v^3 & v^4 \\ \hline V \end{matrix} = \begin{matrix} W^v & \begin{matrix} a^1 & a^2 & a^3 & a^4 \\ \hline I \end{matrix} \end{matrix}$$





# Attention



## Self-attention

$$\begin{aligned} Q &= W^q I \\ K &= W^k I \\ V &= W^v I \end{aligned}$$

Parameters  
to be  
learned

$$A' \leftarrow A = K^T Q$$

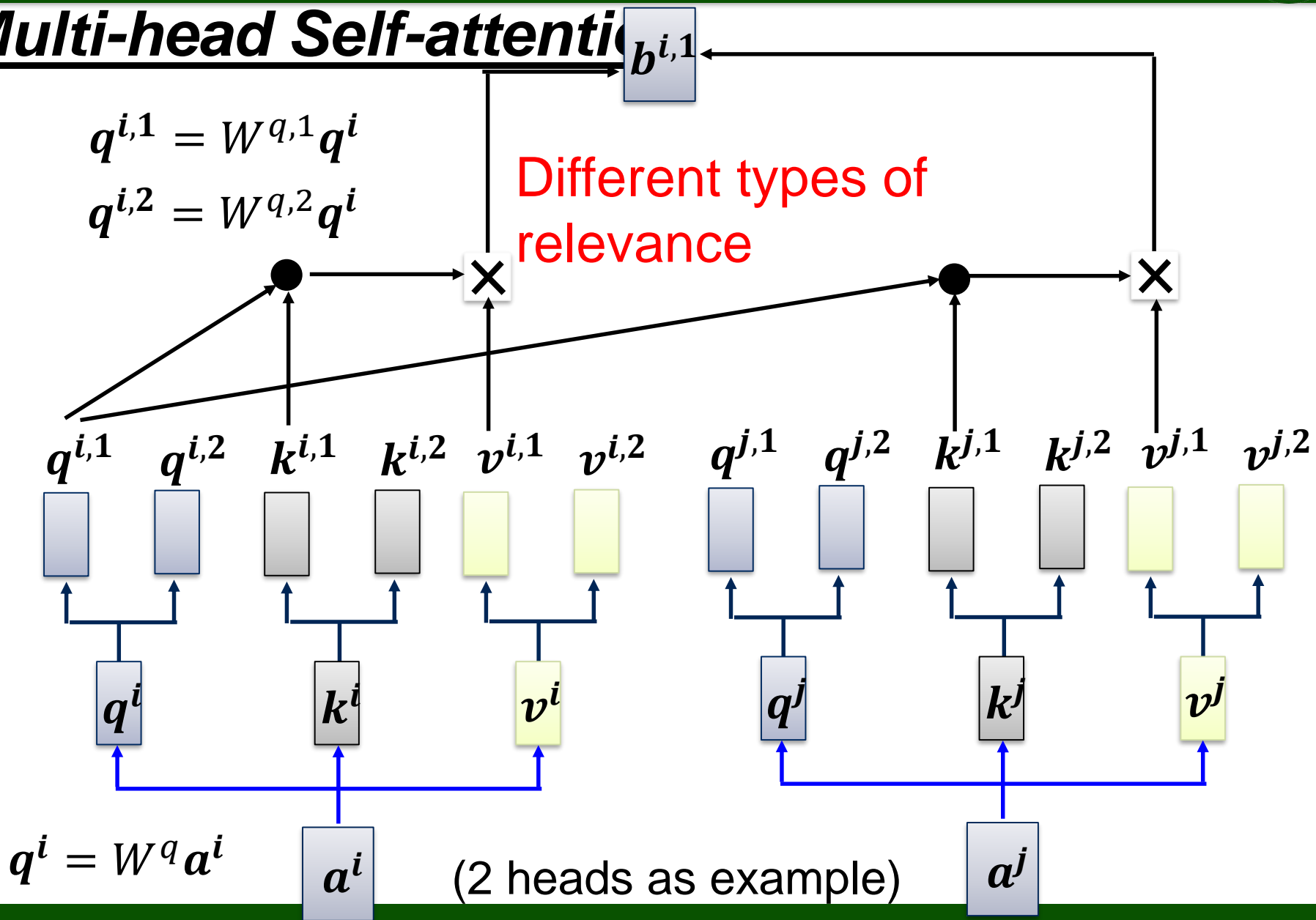
Attention Matrix

$$O = V A'$$

# Attention

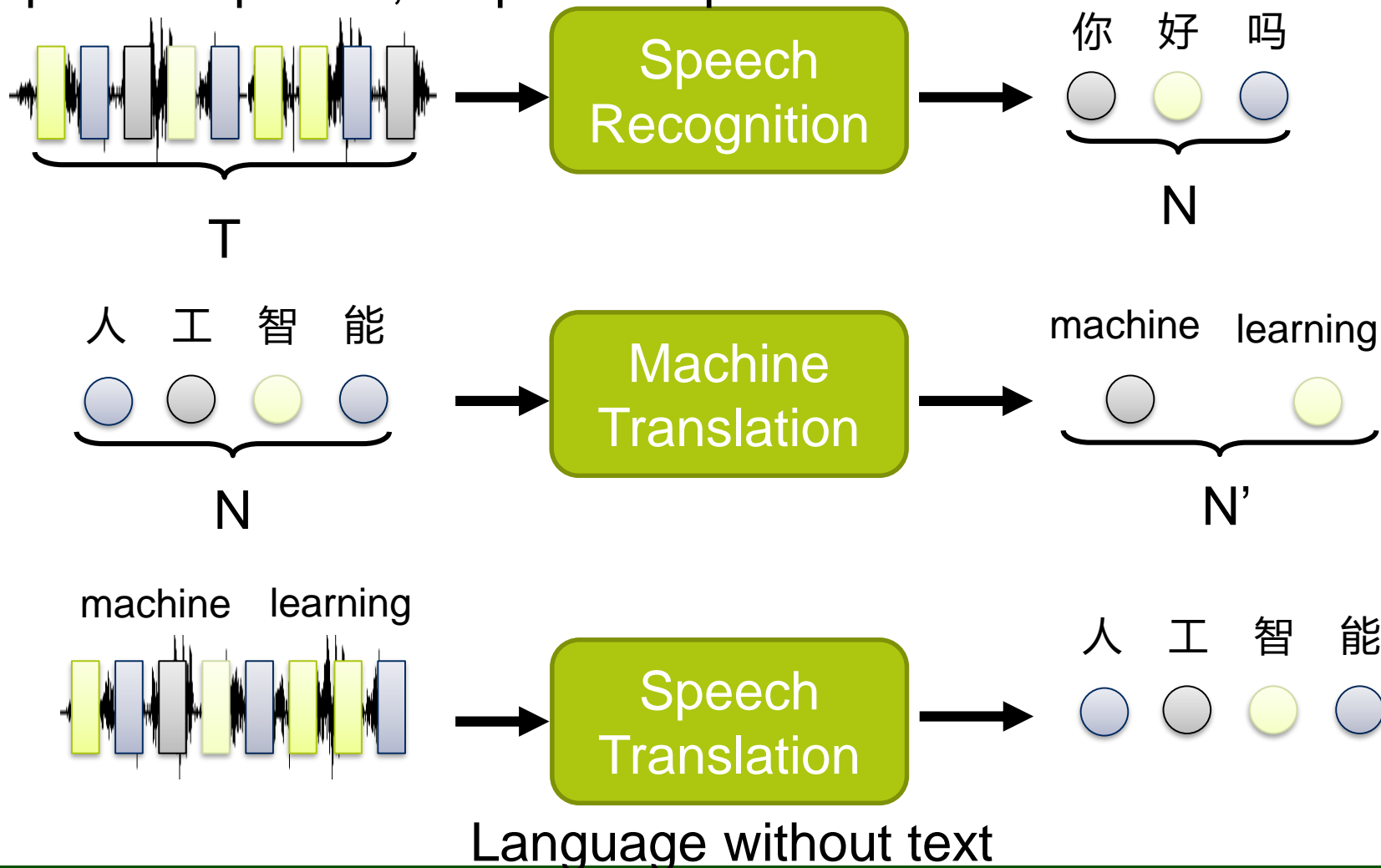


## Multi-head Self-attention



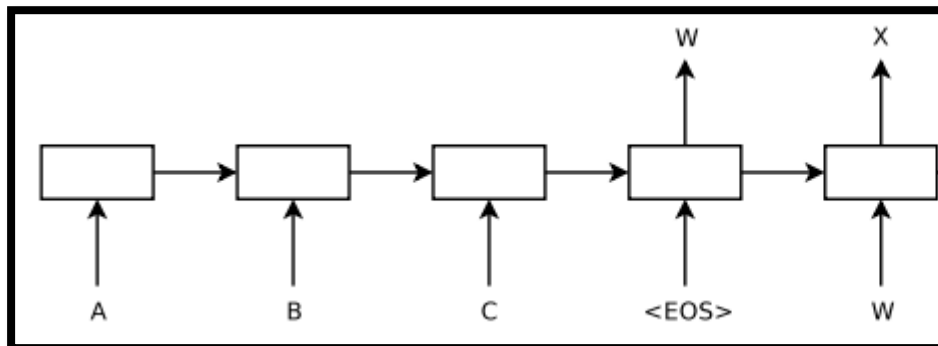
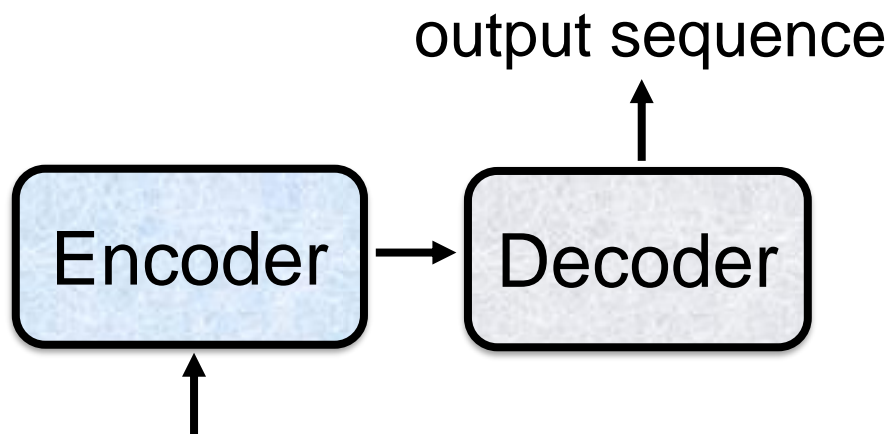
## Sequence-to-sequence (Seq2seq)

Input a sequence, output a sequence



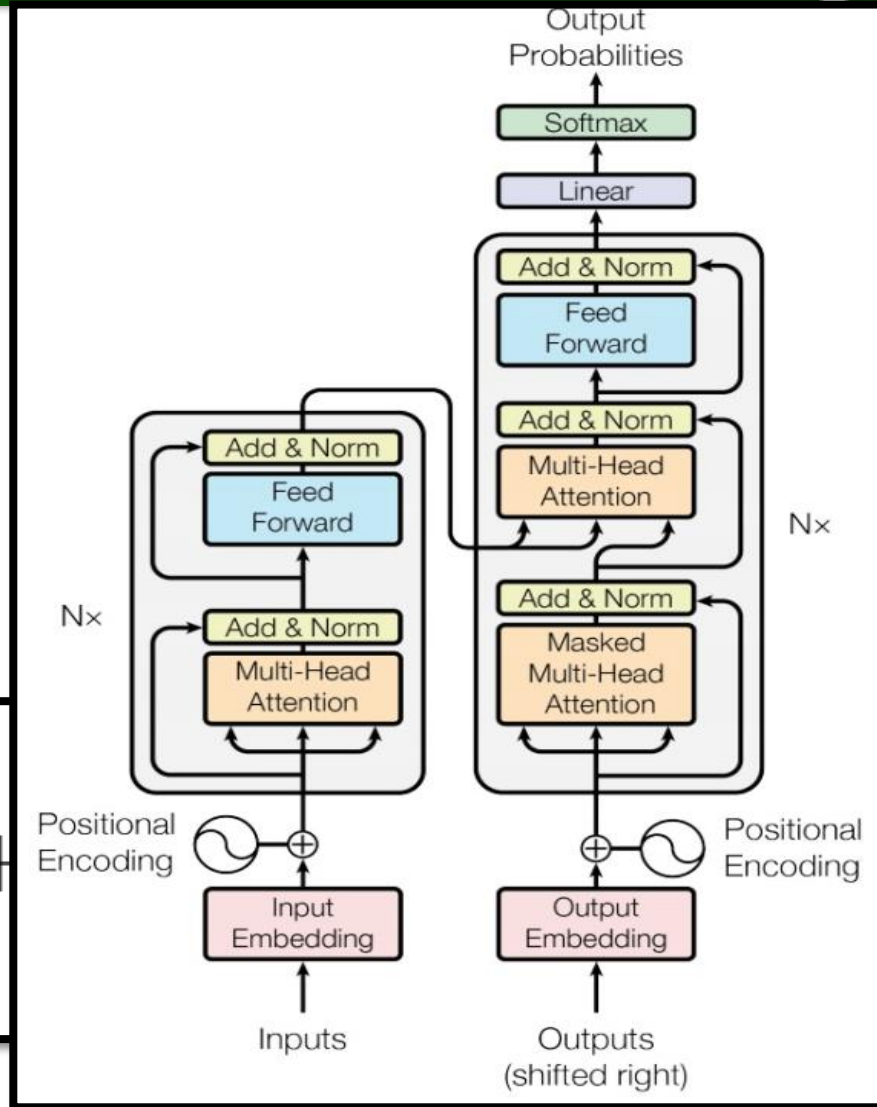
# Transformer

## Seq2seq



Sequence to Sequence Learning with  
Neural Networks

<https://arxiv.org/abs/1409.3215>

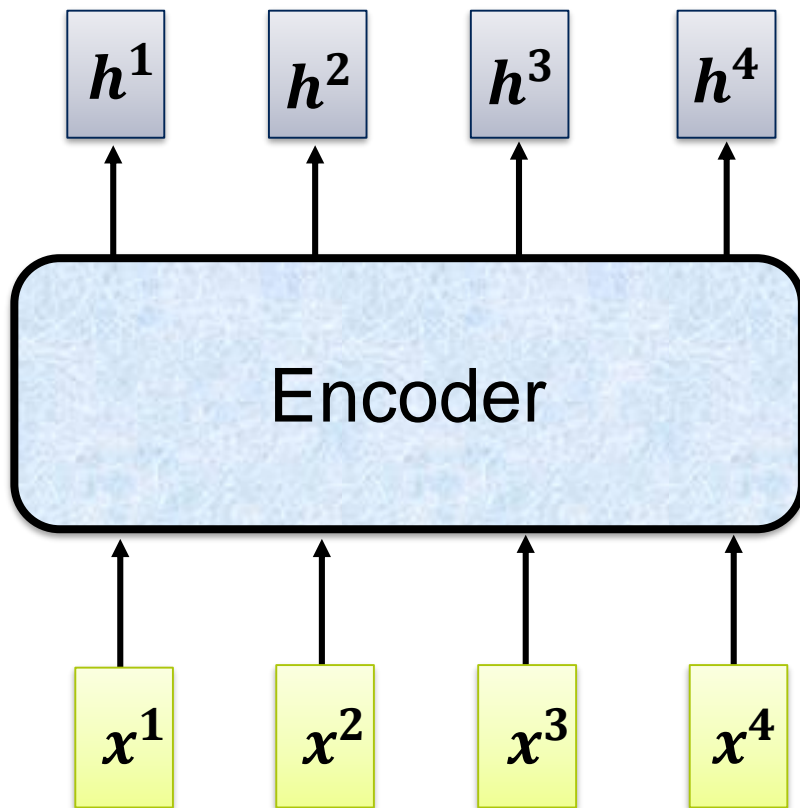


## Transformer

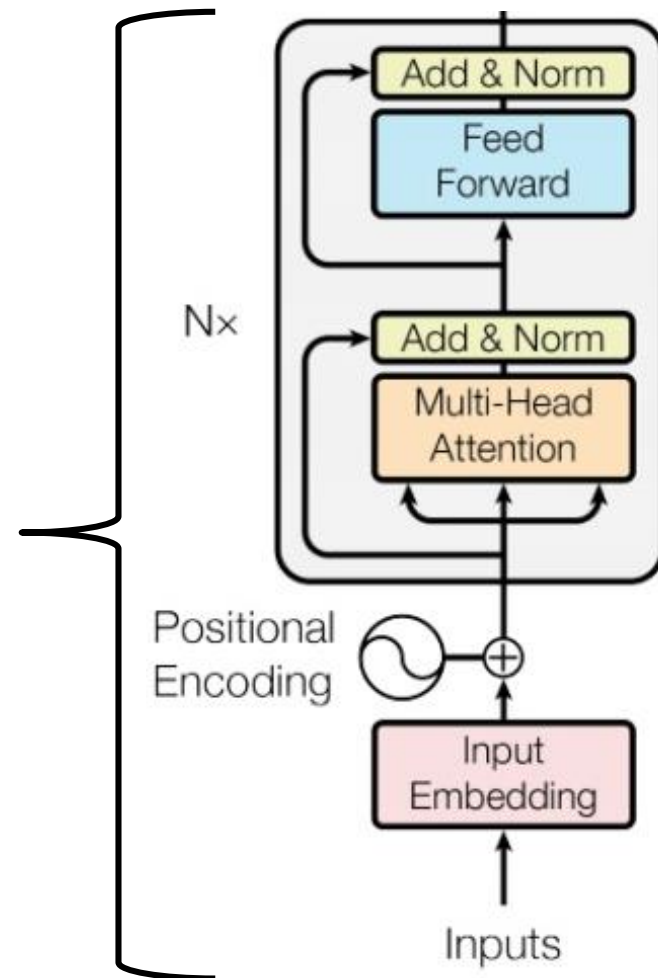
<https://arxiv.org/abs/1706.03762>

## Encoder

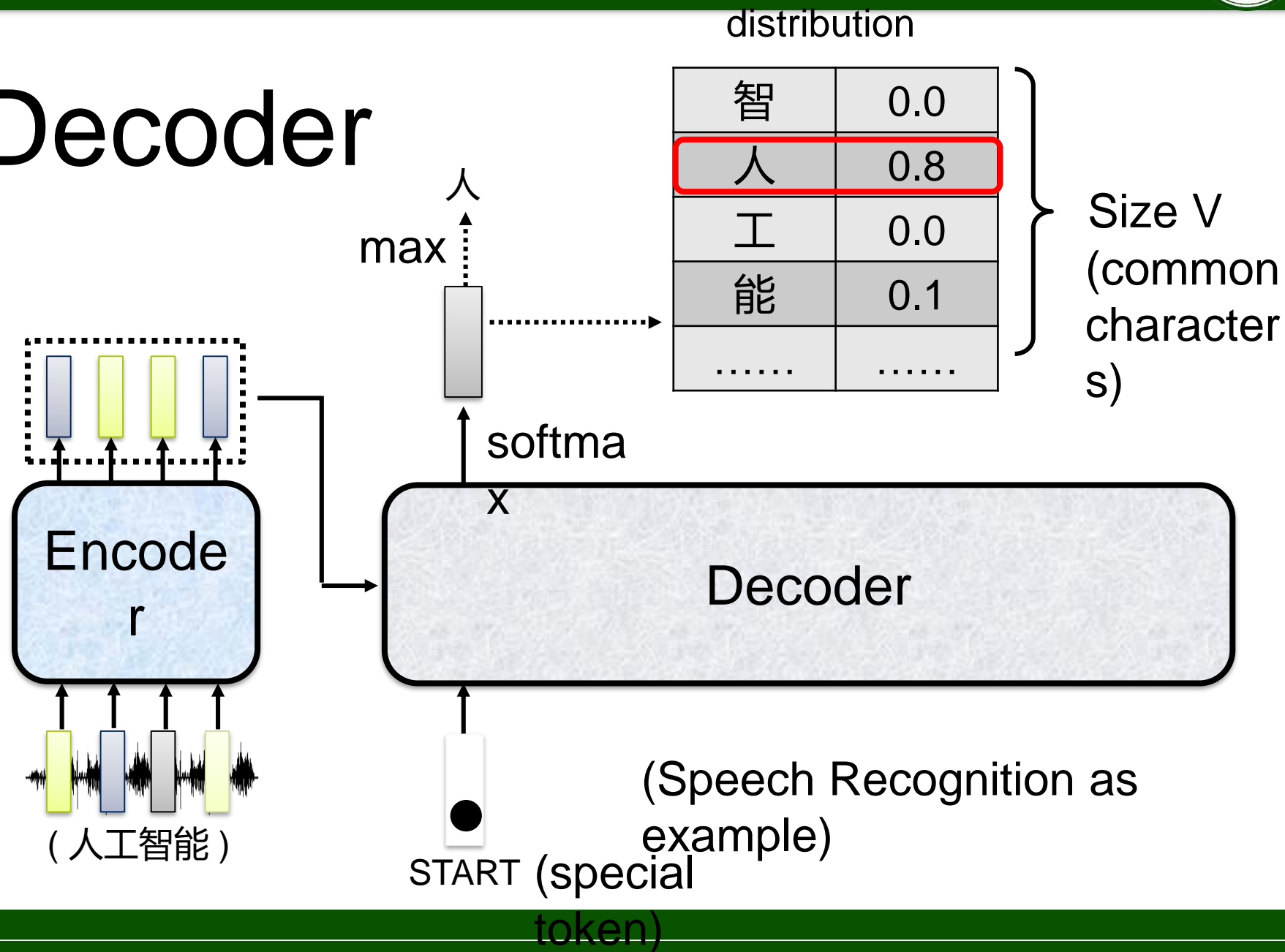
You can use **RNN** or **CNN**.



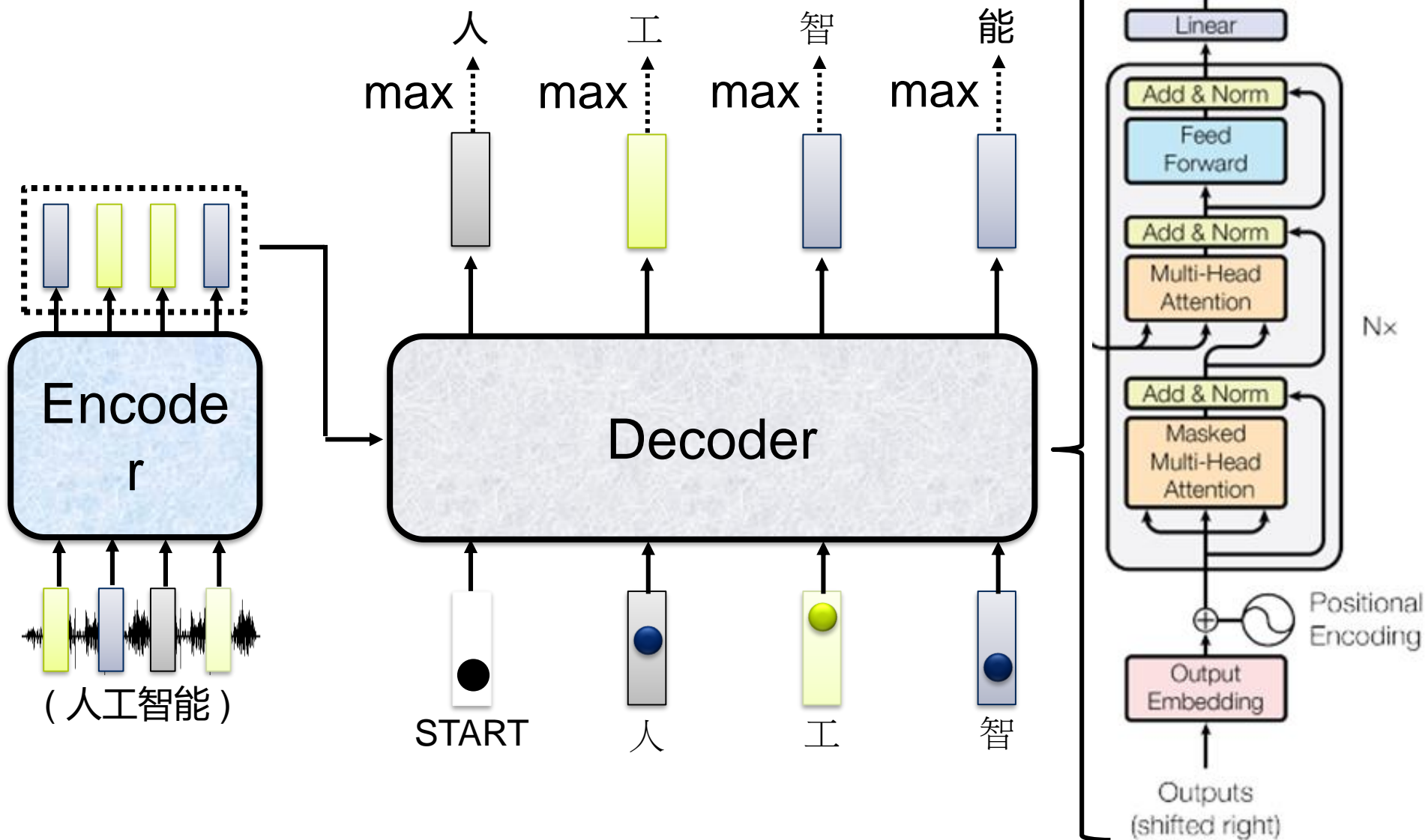
Transformer's Encoder



## Decoder



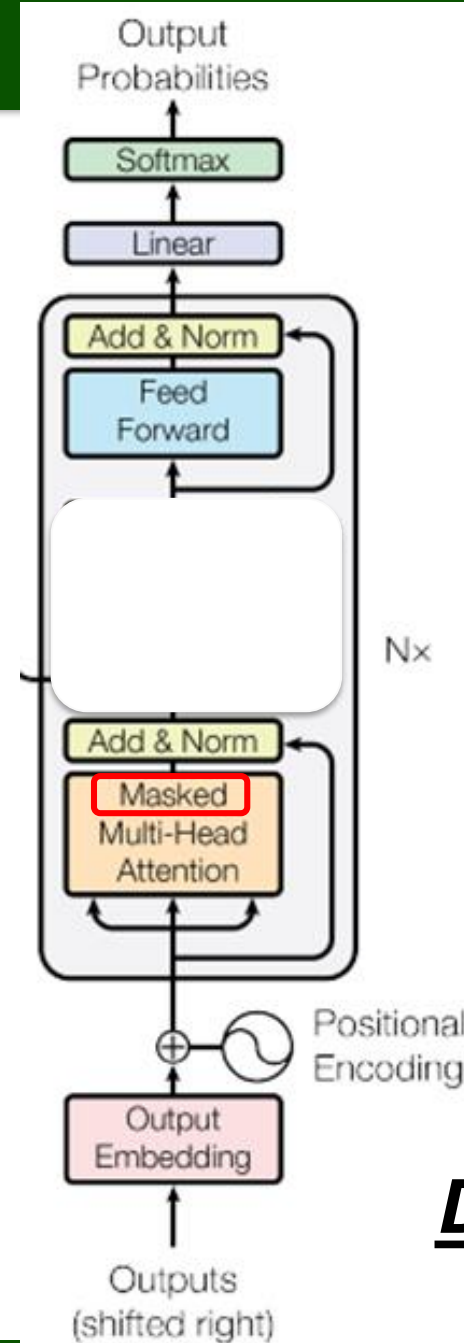
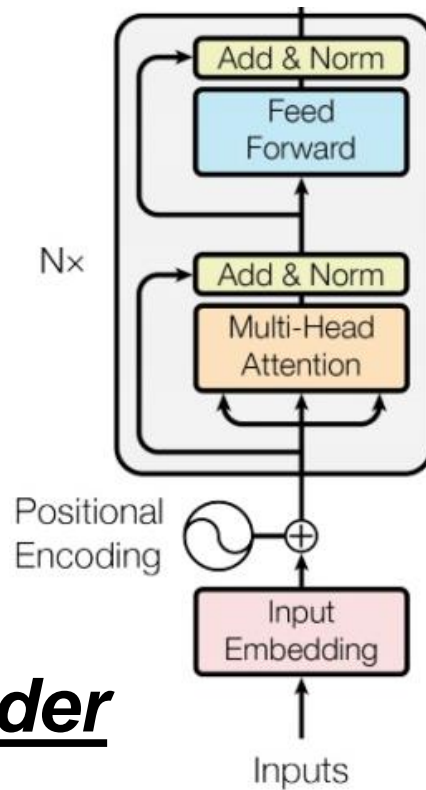
# Transformer



# Transformer



**Encoder**



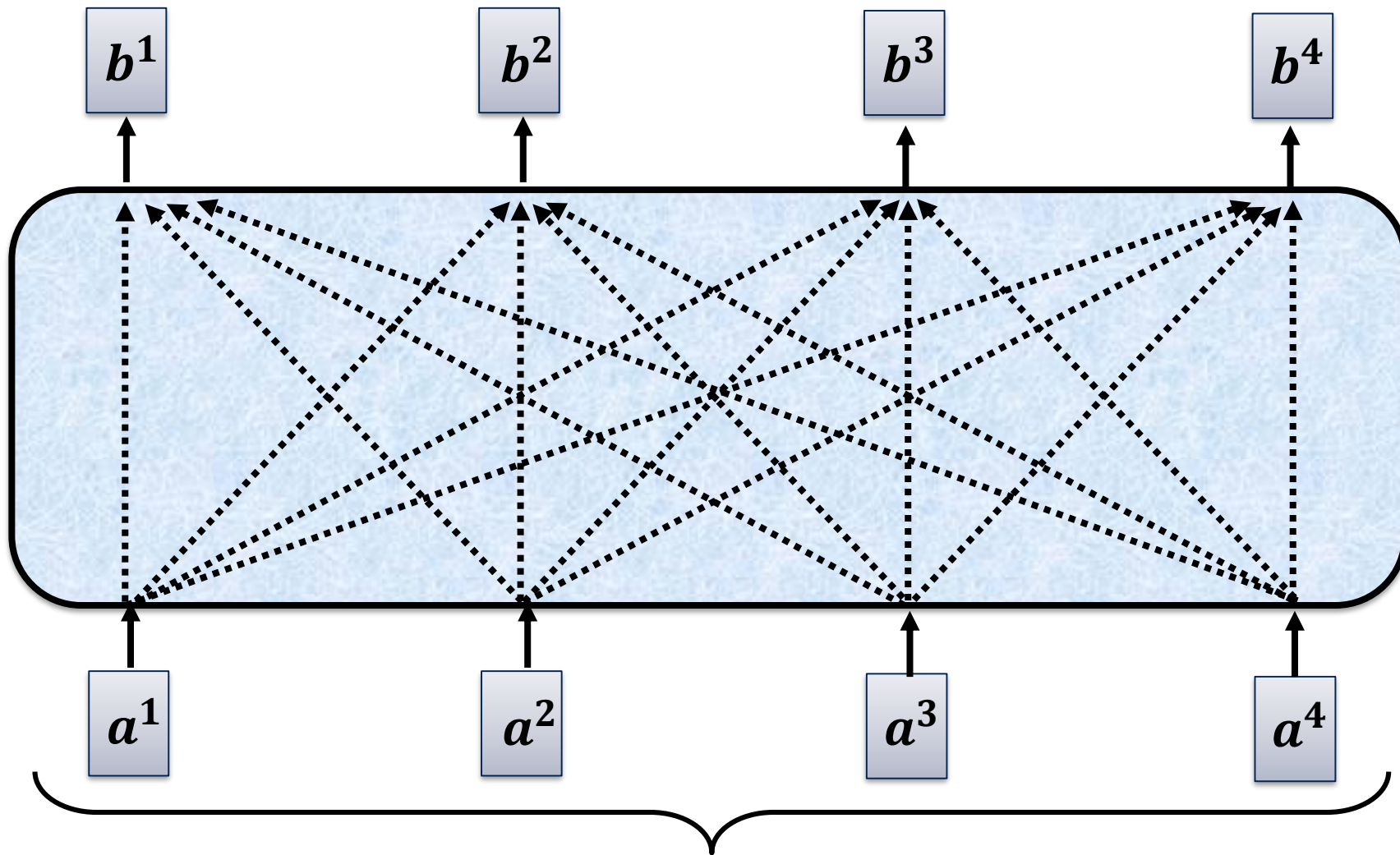
**Decoder**



# Attention (Transformer)



**Self-attention** → **Masked Self-attention**

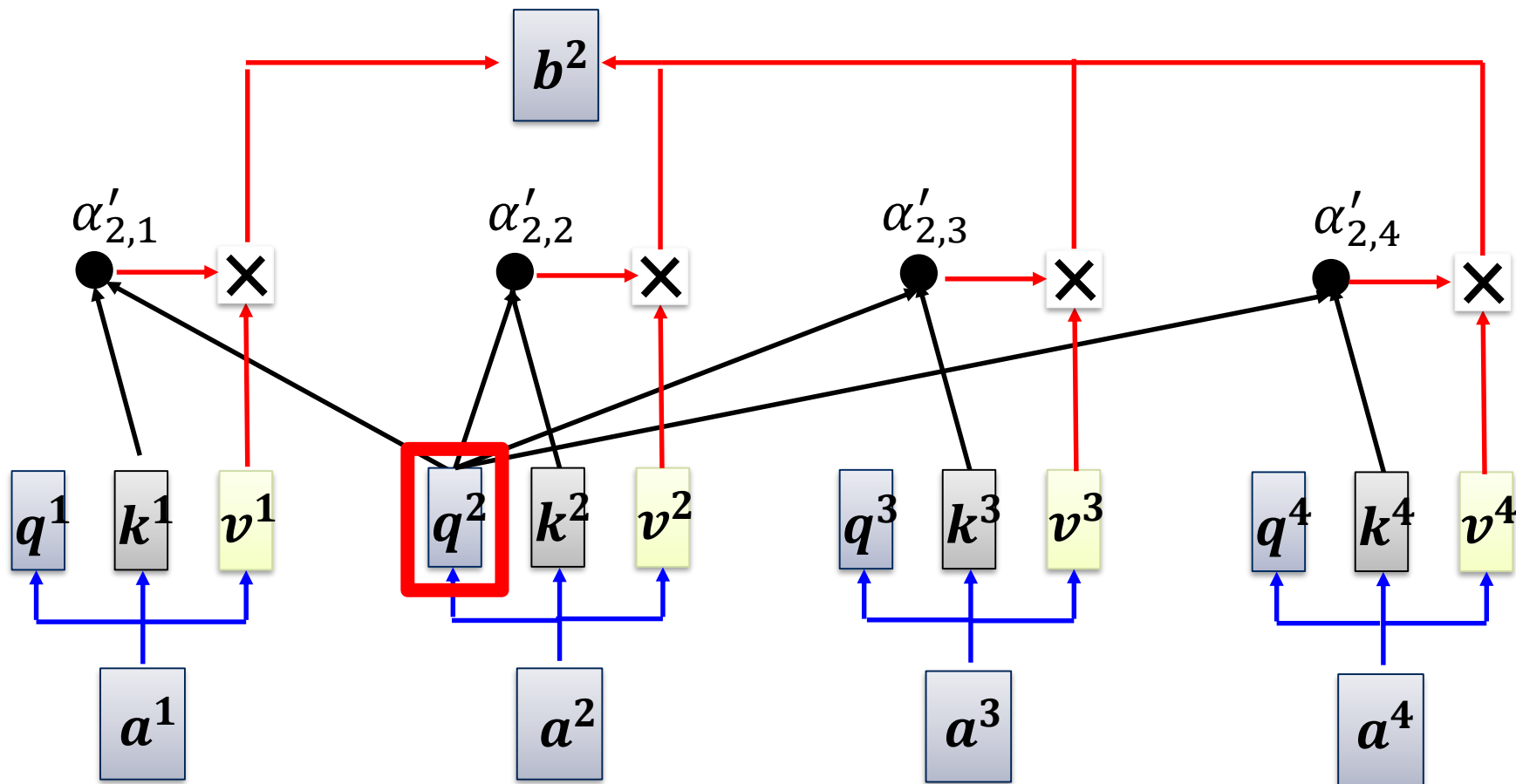


Can be either **input** or a **hidden layer**

# Attention (Transformer)

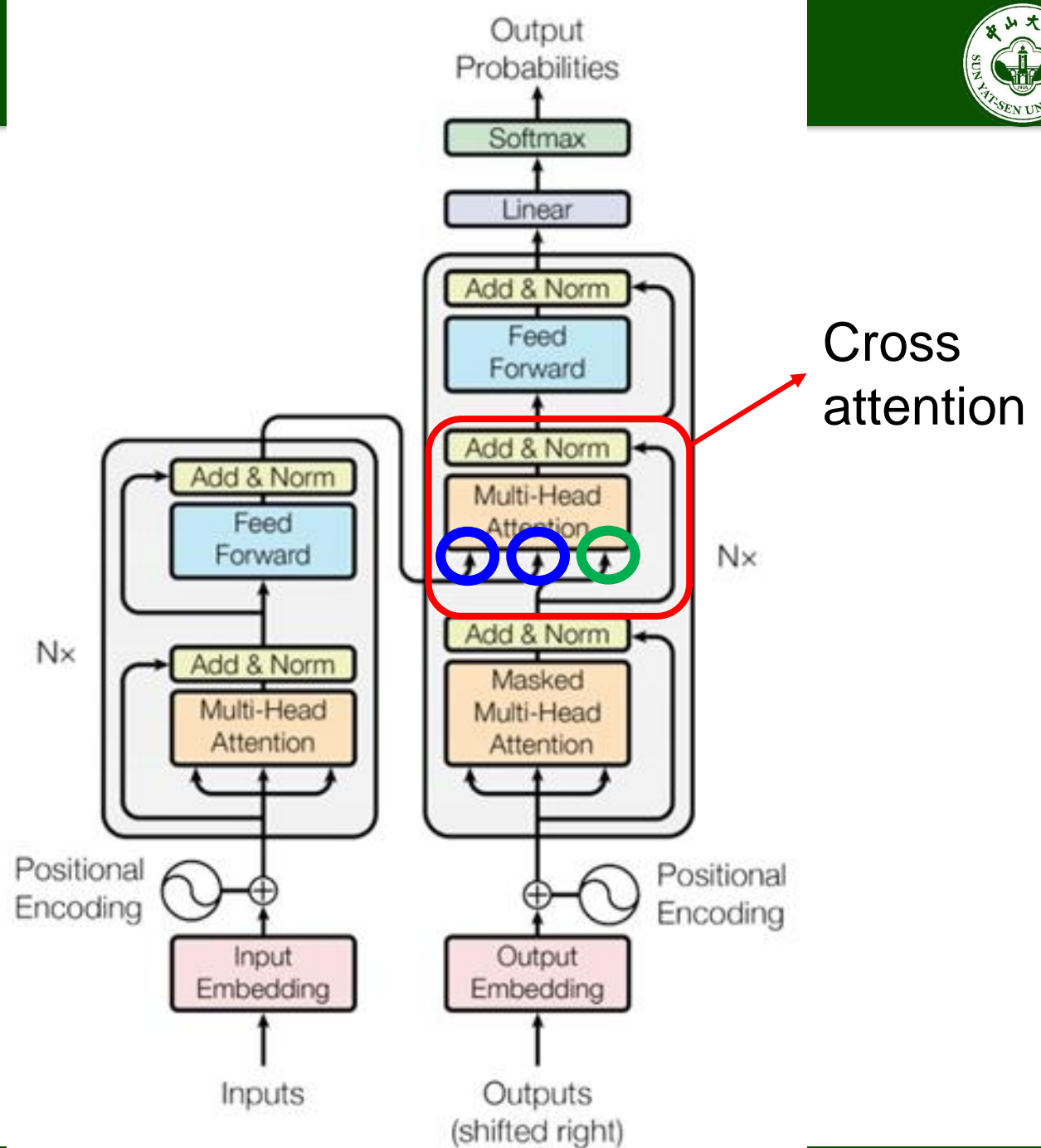


Self-attention  $\rightarrow$  Masked Self-attention



Why masked? Consider how does decoder work

# Transformer

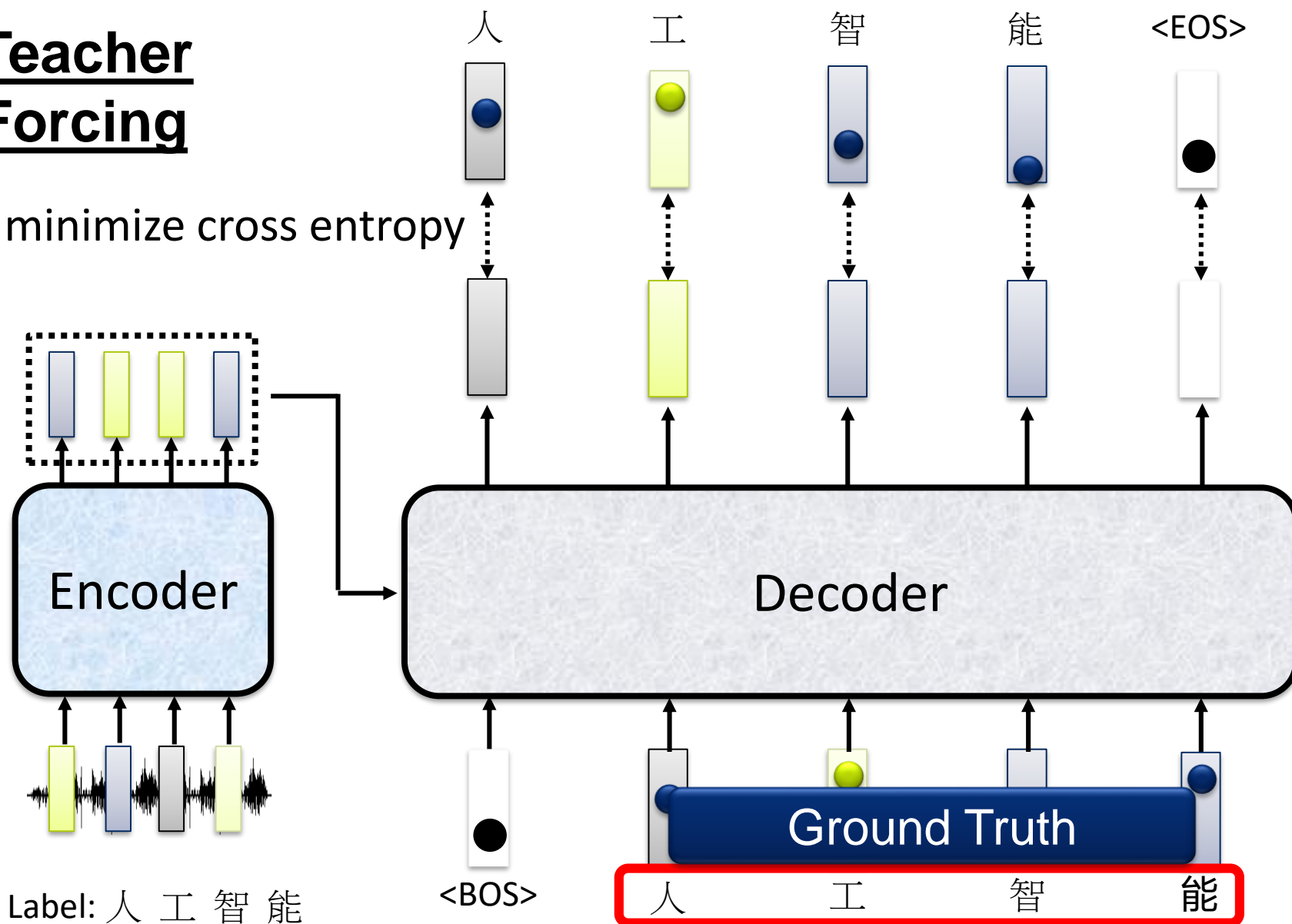


# Transformer



## Teacher Forcing

minimize cross entropy



## Transformer改进及最新工作

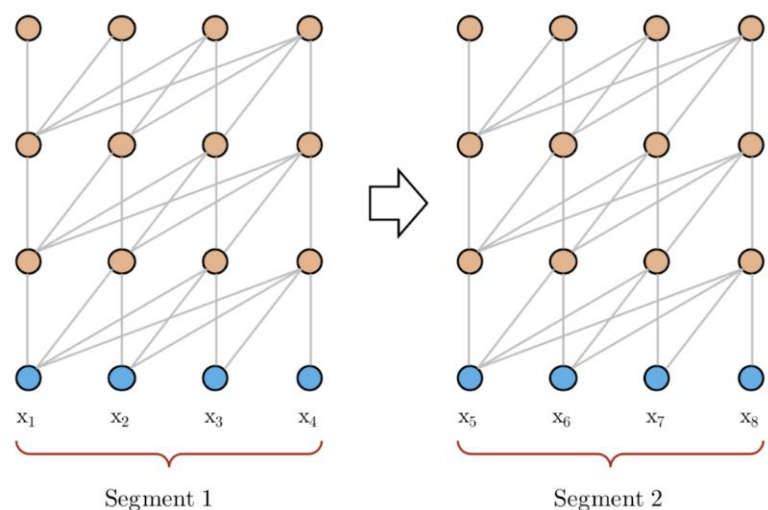
- 针对self-attention改进
    - 目标：1.降低attention的计算复杂度，从而能学习更长的序列。2.针对具体的应用场景，修改attention结构，提升性能。
  - Transformer-XL:
    - vanilla Transformer处理大段文本信息时，将文本分成若干个段(大小为一个batch-size)，每次处理一个段的文本信息。然而，文本语句之间存在上下文联系，因此直接使用划分段的方式导致每个段之间没有任何联系，以至于不能学习更长的句子之间的依赖关系。
- 改进方法：
- 循环机制（Recurrence Mechanism）：仍使用分段的方式建模，但在段与段之间引入循环机制，使得当前段在建模的时候能够利用之前段的信息来实现句子之间长期依赖关系。
  - 相对位置编码（Relative Positional Encoding）：对当前段建模的时候需要用到前面段，但前面段对当前段的重要性是不同的。针对这个，提出一种新的位置编码方式，即会根据词之间的相对距离进行编码。

# Transformer

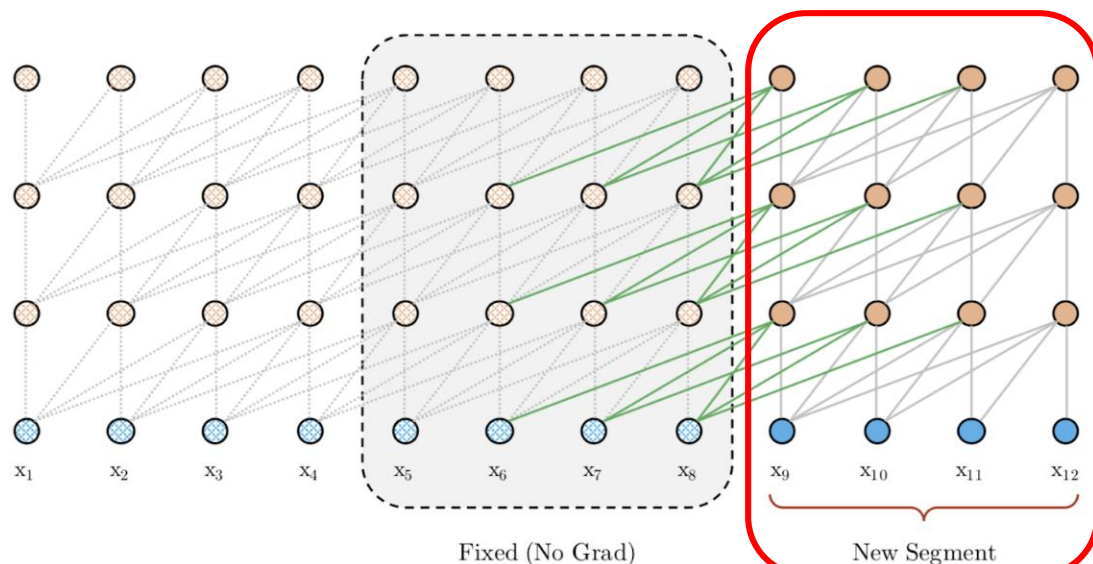
## Transformer改进及最新工作

- 针对self-attention改进
  - Transformer-XL
    - 在训练阶段，处理一个new段（红框）时，每个隐藏层接受两个输入：
      - 该段前面的隐藏层输出（encode层信息，灰线）
      - 前面段的隐藏层的输出（绿线）

Transformer (Training)

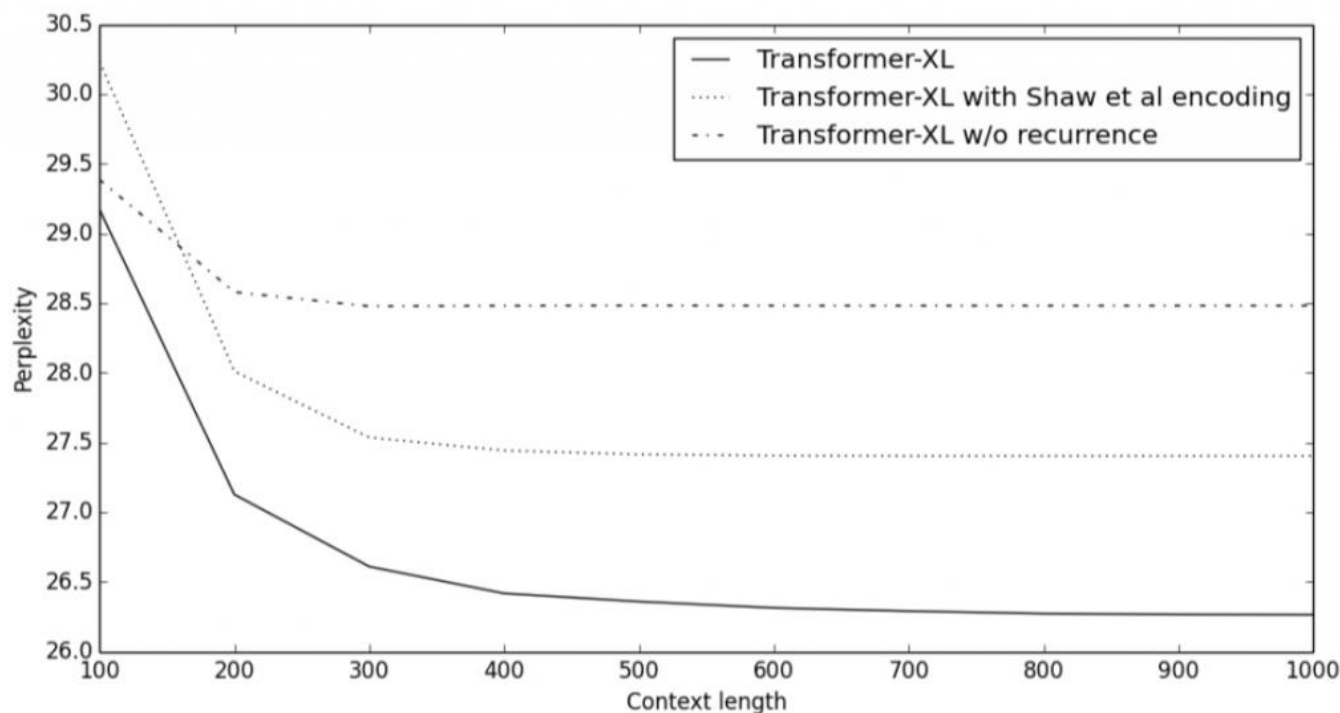


Transformer-XL (Training)



## Transformer改进及最新工作

- 针对self-attention改进
  - Transformer-XL
    - Ablation:比较了不同上下文长度中包不包含循环机制，以及是不是用相对位置编码的得分。可见，使用循环机制和相对位置编码的Transformer-XL明显优于其他的模型，并且能够有效利用长期依赖性，而且它能捕获超出RNN 80%的依赖性，和超出Transformer 450%的依





# Transformer

## Transformer改进及最新工作

- 针对self-attention改进

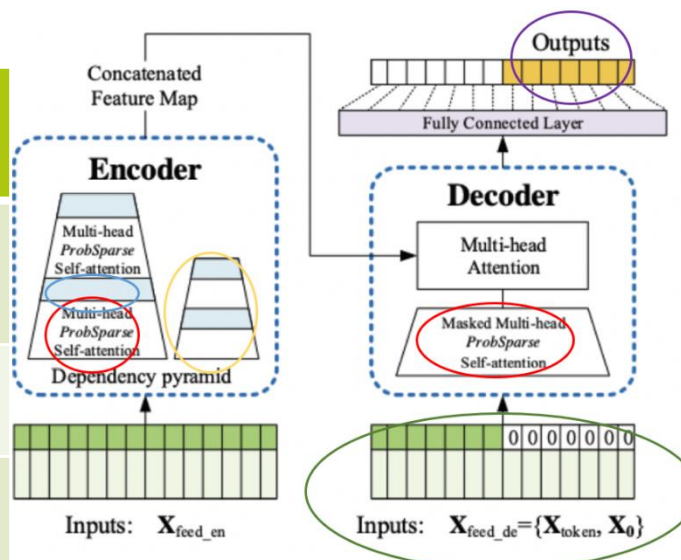
其他改进:

- Adaptive Attention Span: 根据Transformer的设计, 多头attention中每个head会学到不同的模块分析。为了减少时间复杂度和空间复杂度, 利用一个自适应机制使得多头attention的每个head只学习关注的部分, 而不关注的部分被mask为0。

Sukhbaatar, Sainbayar, et al. "Adaptive Attention Span in Transformers." *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 2019.

- Informer (AAAI21 Best Paper)

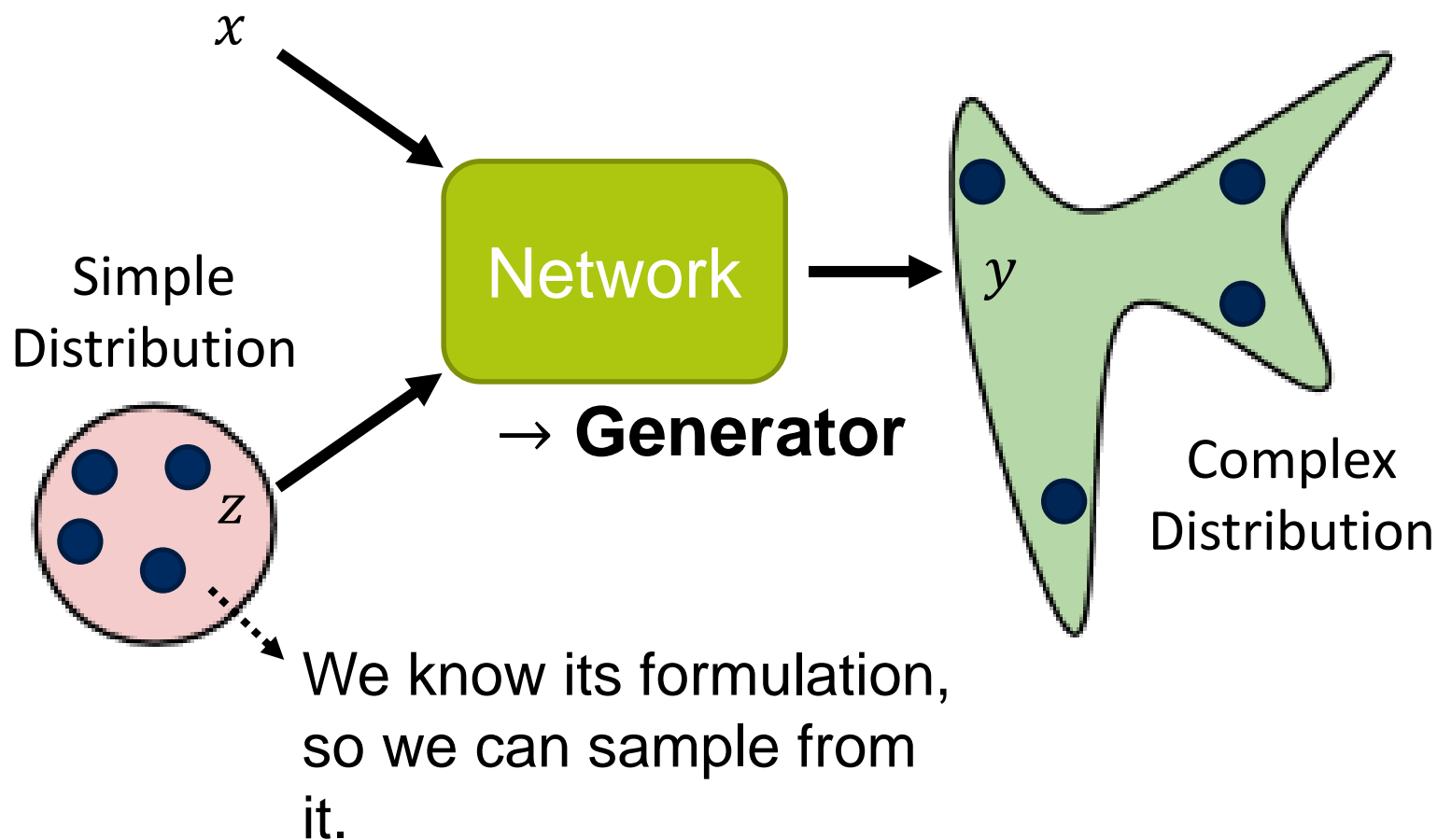
Motivation (Transformer 不足)	Contribution (Informer 改进)
Self-attention平方级的计算复杂度	提出ProbSparse Self-attention, 筛选出最重要的query, 降低计算复杂度
堆叠多层网络, 内存占用瓶颈	提出Self-attention Distilling, 减少维度和网络参数量
Step-by-step解码预测, 速度较慢	提出Generative Style Decoder, 一步得到所有预测结果



Zhou, Haoyi, et al. "Informer: Beyond efficient transformer for long sequence time-series forecasting." *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 35. No. 12. 2021.



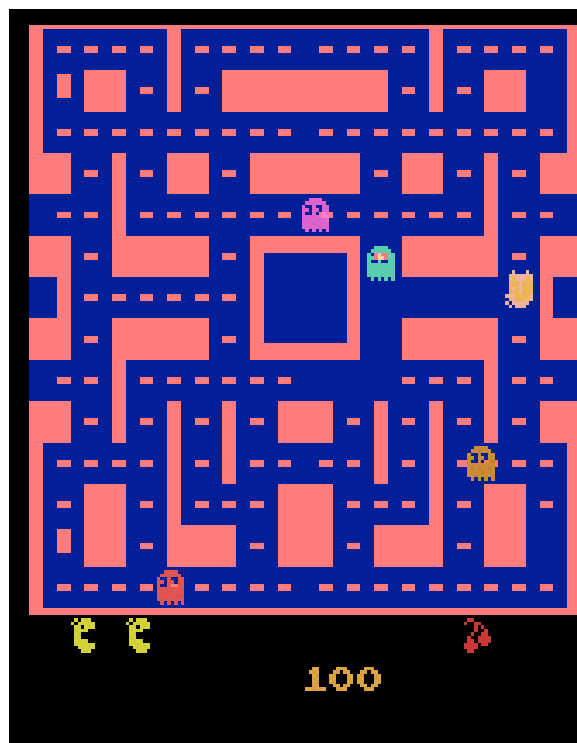
## Network as Generator



# GAN (Generative Adversarial Networks)

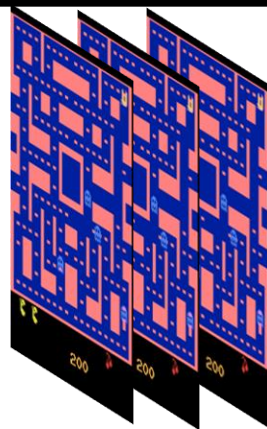


## Why distribution?

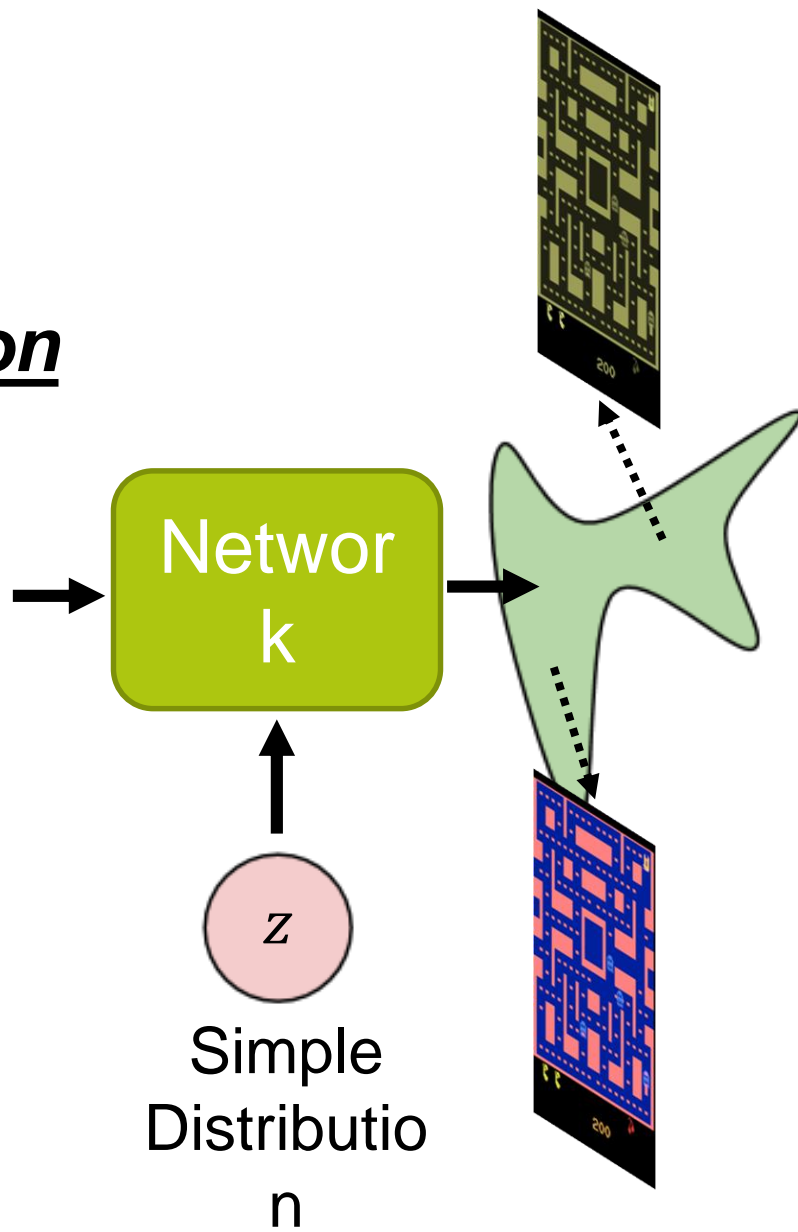


Prediction

### Video Prediction



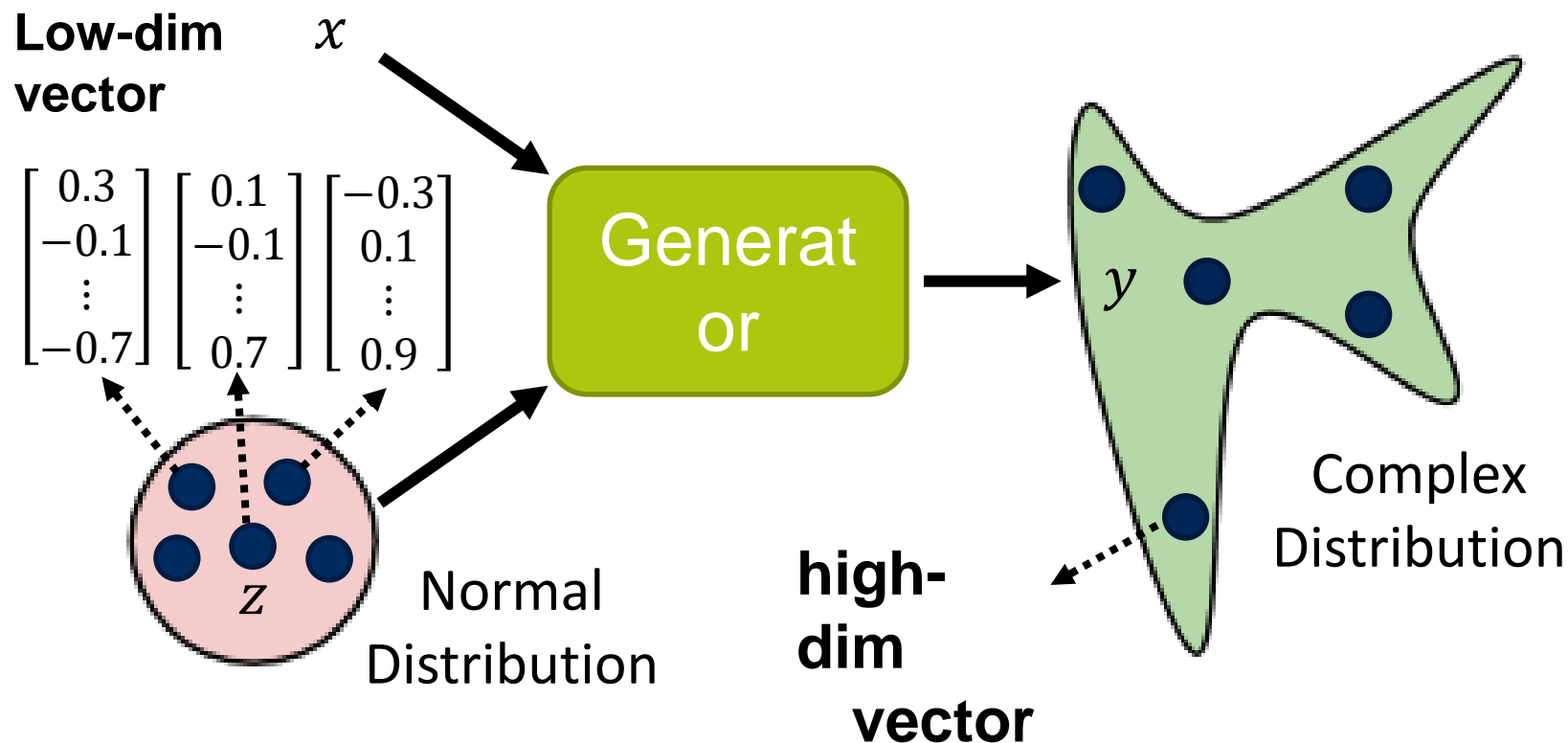
Previous  
frames



# GAN (Generative Adversarial Networks)



Unconditional generation

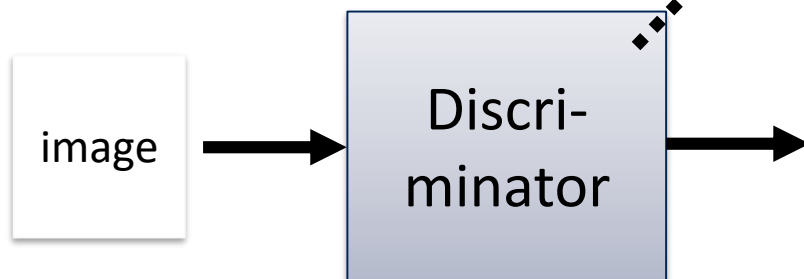


# GAN (Generative Adversarial Networks)



## Discriminator

It is a neural network  
(that is, a function).



**Scalar:** Larger means real,  
smaller value fake.



# GAN (Generative Adversarial Networks)



Initialize generator and discriminator

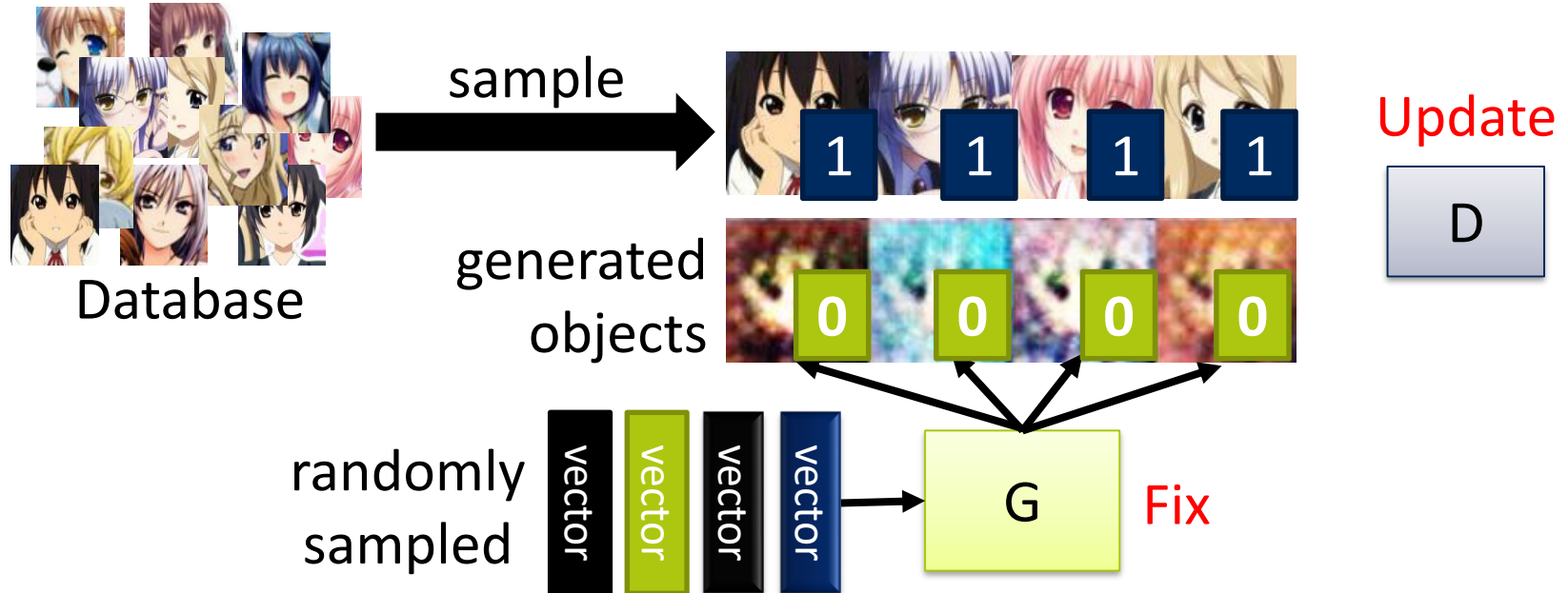
## Algorithm

In each training iteration:

G

D

**Step 1:** Fix generator G, and update discriminator D



Discriminator learns to assign high scores to real objects and low scores to generated objects.

# GAN (Generative Adversarial Networks)



Initialize generator and discriminator

## Algorithm

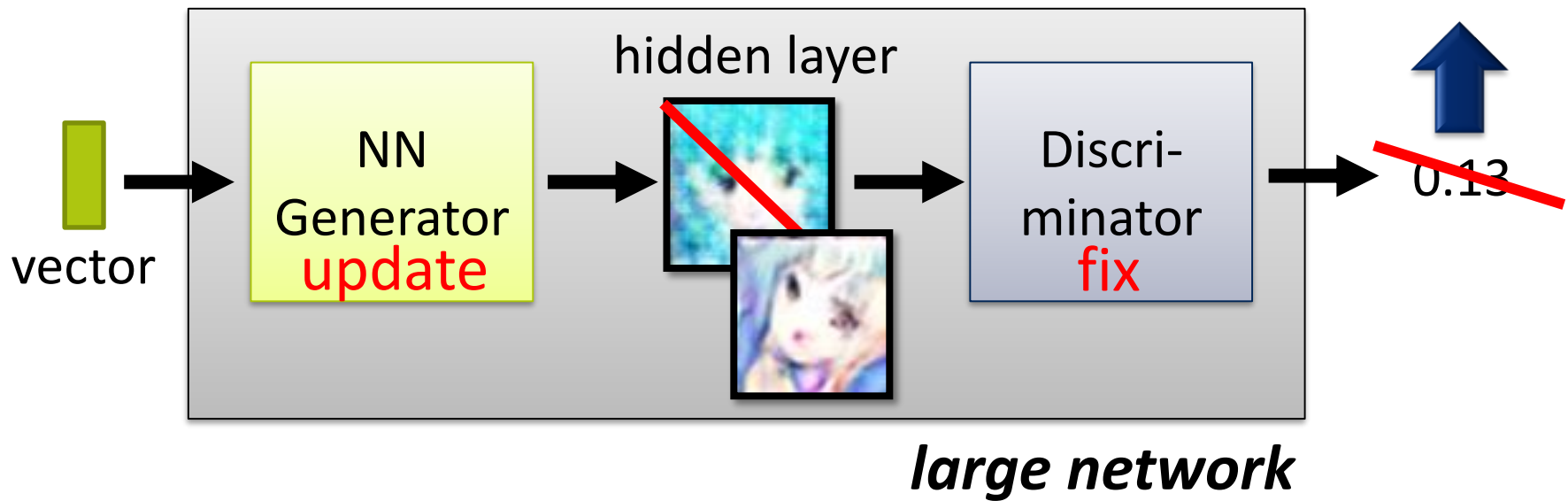
In each training iteration:

G

D

Step 2: Fix discriminator D, and update generator G

Generator learns to “fool” the discriminator



# GAN (Generative Adversarial Networks)



## Algorithm

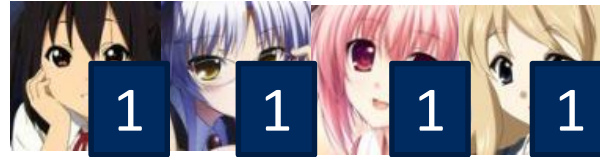
Initialize generator and discriminator

In each training iteration:

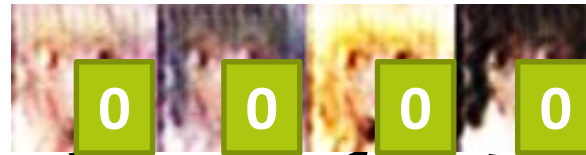


Learning  
D

Sample some  
real objects:



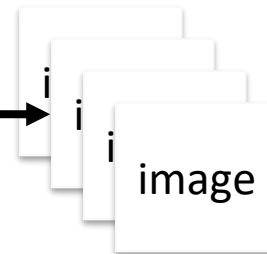
Generate some  
fake objects:



Update



Learning  
G



1



# GAN (Generative Adversarial Networks)



## Progressive GAN



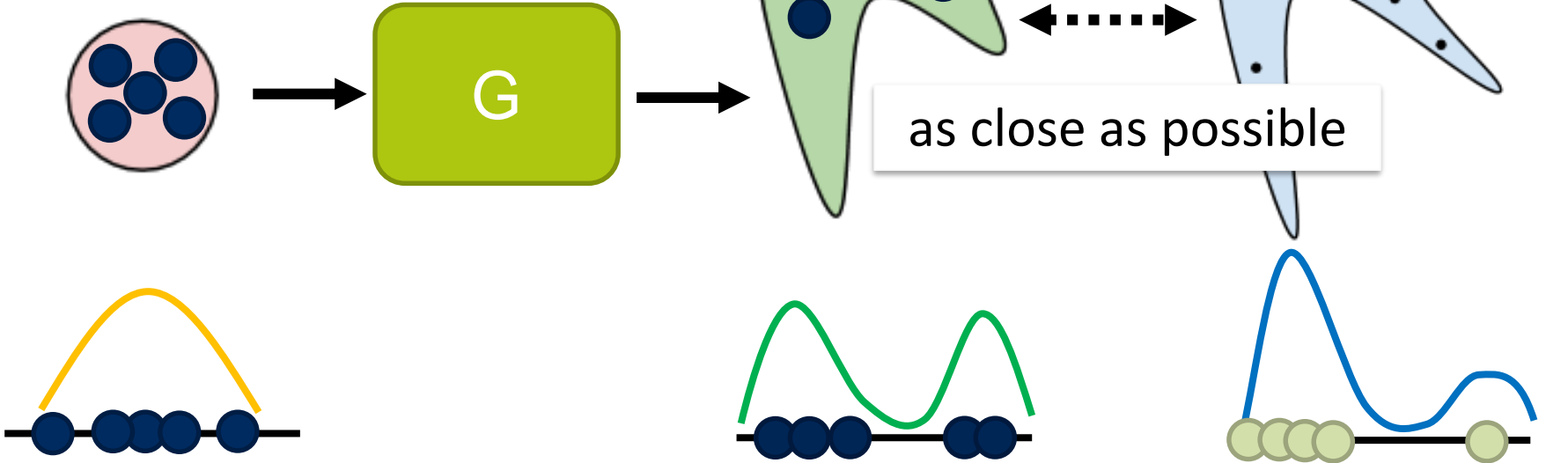
# GAN (Generative Adversarial Networks)



## Our Objective

$$\text{c.f. } w^*, b^* = \arg \min_{w, b} L$$

Normal  
Distribution



$$G^* = \arg \min_G \underline{\text{Div}(P_G, P_{data})}$$

Divergence between distributions  $P_G$  and  $P_{data}$

How to compute the divergence?

# GAN (Generative Adversarial Networks)

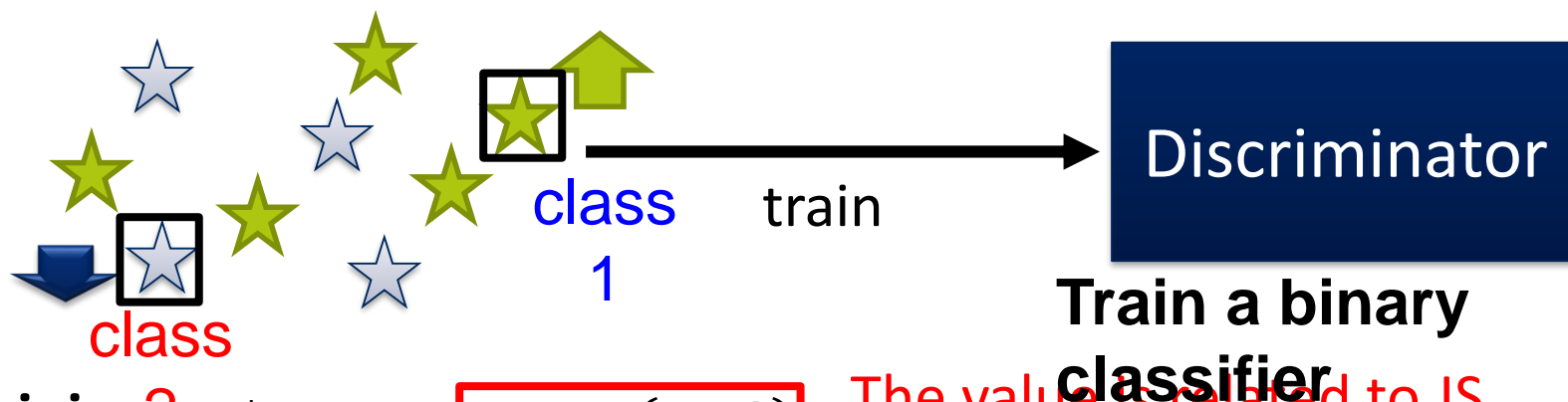


## Discriminator

$$G^* = \arg \min_G \text{Div}(P_G, P_{data})$$

★ : data sampled from  $P_{data}$

★ : data sampled from  $P_G$



**Training:**  $D^* = \arg \max_D V(D, G)$  The value is related to JS divergence.

**Objective Function for D**

$$V(G, D) = E_{y \sim P_{data}} [\log D(y)] + E_{y \sim P_G} [\log(1 - D(y))]$$

$$D^* = \arg \max_D V(D, G)$$

negative cross

entropy

=

Training classifier:

minimize cross

entropy

# GAN (Generative Adversarial Networks)



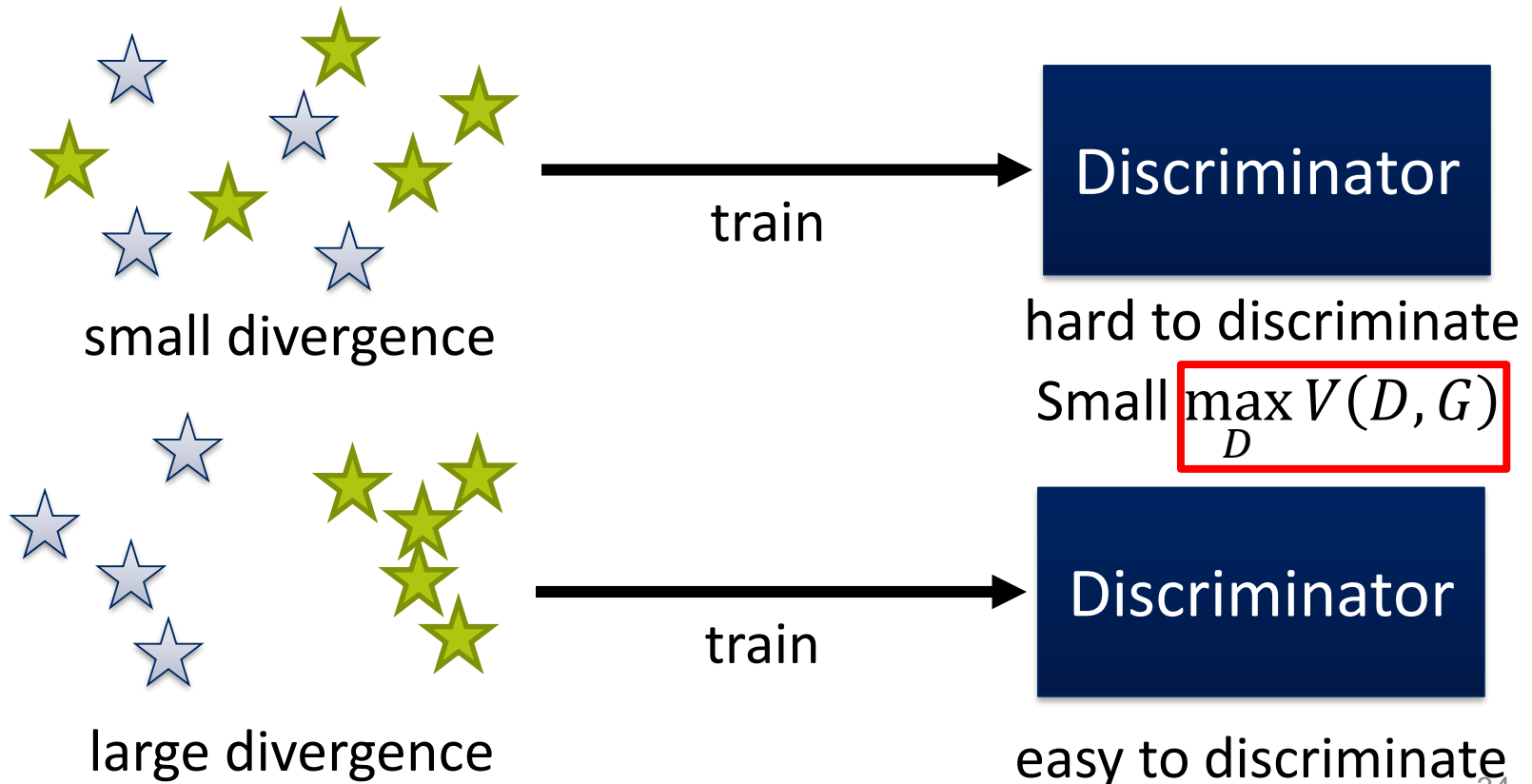
**Discriminator**  $G^* = \arg \min_G \text{Div}(P_G, P_{data})$

★ : data sampled from  $P_{data}$

★ : data sampled from  $P_G$

**Training:**

$$D^* = \arg \max_D V(D, G)$$



# GAN (Generative Adversarial Networks)



$$G^* = \arg \min_G \max_D V(G, D)$$

$$D^* = \arg \max_D V(D, G)$$

The maximum objective value is related to JS divergence.

**Step 1**: Fix generator  $G$ , and update discriminator  $D$

**Step 2**: Fix discriminator  $D$ , and update generator  $G$

# GAN (Generative Adversarial Networks)



## GAN的改进

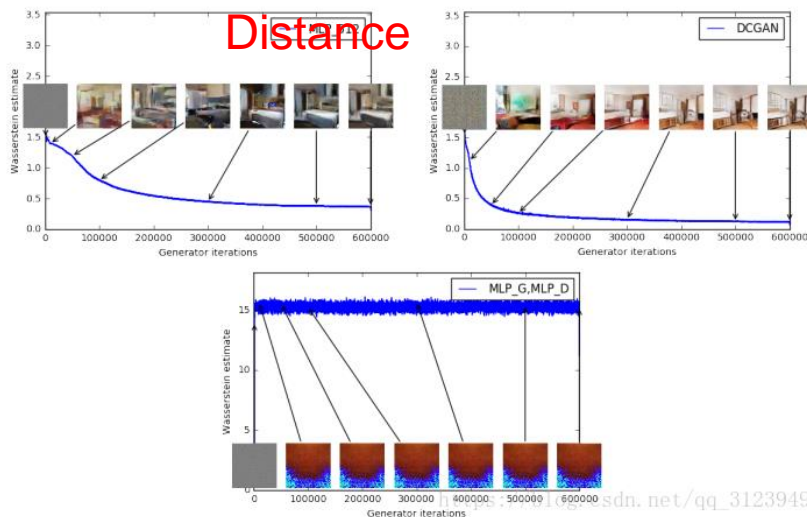
### ● WGAN

- 改进: Discriminator最有一层去掉sigmoid; Discriminator和Generator的loss不取log; 每次更新Discriminator的参数之后, 把它们的绝对值截断到不超过一个固定常数c; 不用基于动量的优化器 (Momentum和Adam), 推荐RMSProp。
- 提升: 由于交叉熵 (JS散度) 不适合衡量不相交部分的分布之间的距离, 转而利用 Wasserstein Distance度量生成数据分布和真实数据之间的距离, 彻底解决GAN训练不稳定的问题, 不再需要小心平衡生成器和判别器的训练程度; 基本解决了模式崩溃的问题, 确保了生成样本的多样性; 提供了具有意义的价值函数, 可以分别判断 Discriminator和Generator 是否收敛。

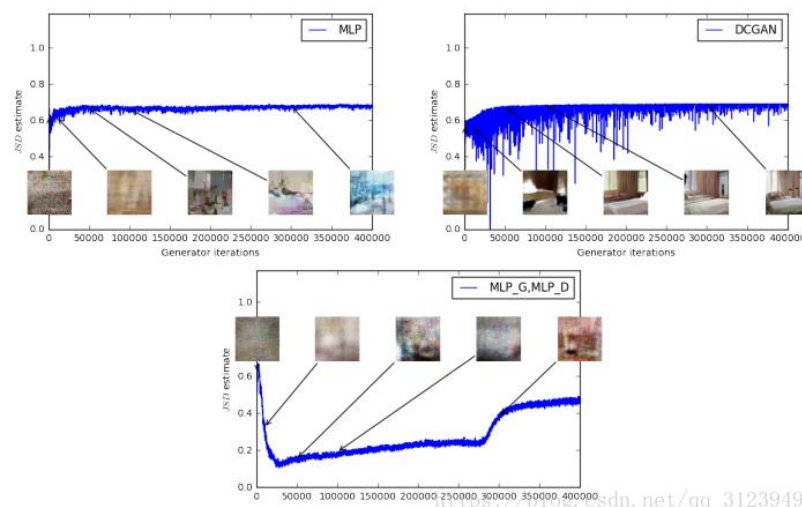
### Wasserstein

### Distance

LOS  
S



### JS Divergence



## GAN最新工作

- **Dual Contrastive Loss and Attention for GANs**: 使用大规模图像数据集时, GAN在无条件图像生成方面生成的图像仍然很容易被甄别出来, 尤其是在具有高方差的数据集 (例如卧室、教堂) 上。因此, 提出一种新的双重对比损失, 并表明通过这种损失, 判别器可以学习更通用和可区分的表示来激励生成质量。同时研究了判别器中不同的注意力架构, 并提出了一个参考注意力机制。

Yu, Ning, et al. "Dual contrastive loss and attention for gans." *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021.

- **Gradient Normalization for Generative Adversarial Networks**: 提出一种新的归一化方法: 梯度归一化(GN), 以解决生成式对抗网络(GANs)梯度不稳定问题。与现有的梯度惩罚和谱归一化等不同, 提出的GN算法只对判别器函数施加梯度范数约束。

Wu, Yi-Lun, et al. "Gradient normalization for generative adversarial networks." *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021.

- **EigenGAN (Layer-Wise Eigen-Learning for GANs)**: 生成对抗网络 (GAN) 的不同层有不同的图像语义。很少有 GAN 模型具有明确的维度来控制特定层中表示的语义属性。提出EigenGAN, 能够无监督地从不同的生成器层挖掘可解释和可控的维度。

He, Zhenliang, Meina Kan, and Shiguang Shan. "Eigengan: Layer-wise eigen-learning for gans." *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021.

The models become larger  
and larger ...

ELMO  
(94M)



BERT  
(340M)

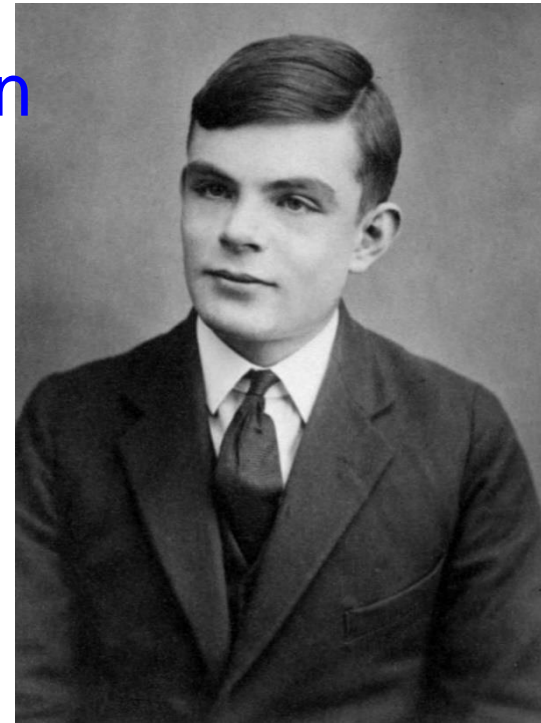


GPT-2  
(1542M)



The models become larger  
and larger ...

GPT-3 is **10** times larger than  
Turing NLG.



GPT-2

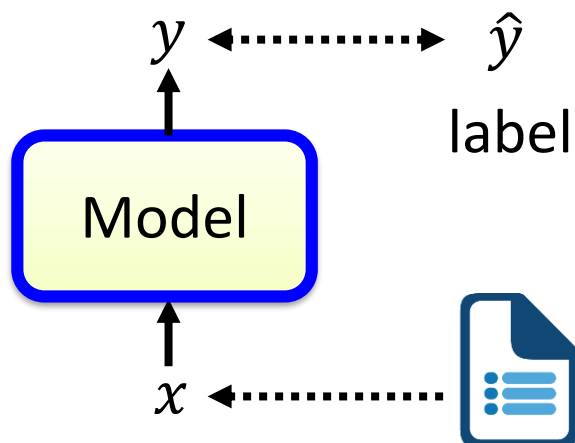
Megatron (8B)

Turing NLG (17B)

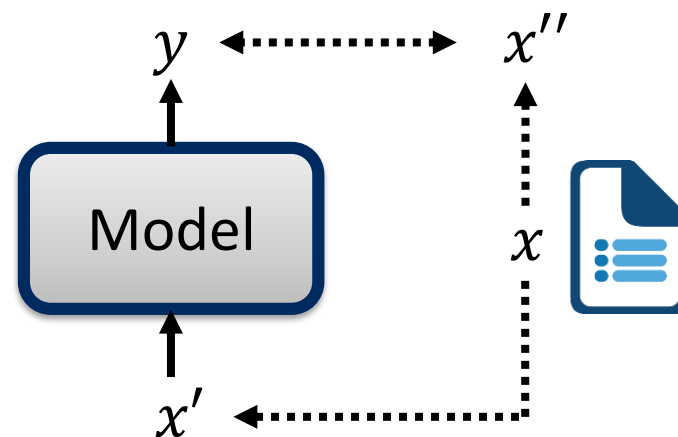


## Self-supervised Learning

Supervised



Self-supervised



Yann LeCun

2019年4月30日 · 🌐

I now call it "self-supervised learning". because "unsupervised" is both a loaded and confusing term.

In self-supervised learning, the system learns to predict part of its input from other parts of its input. In other words a portion of the input is used as a supervisory signal to a predictor fed with the remaining portion of the input.

## Masking Input

<https://arxiv.org/abs/1810.04805>

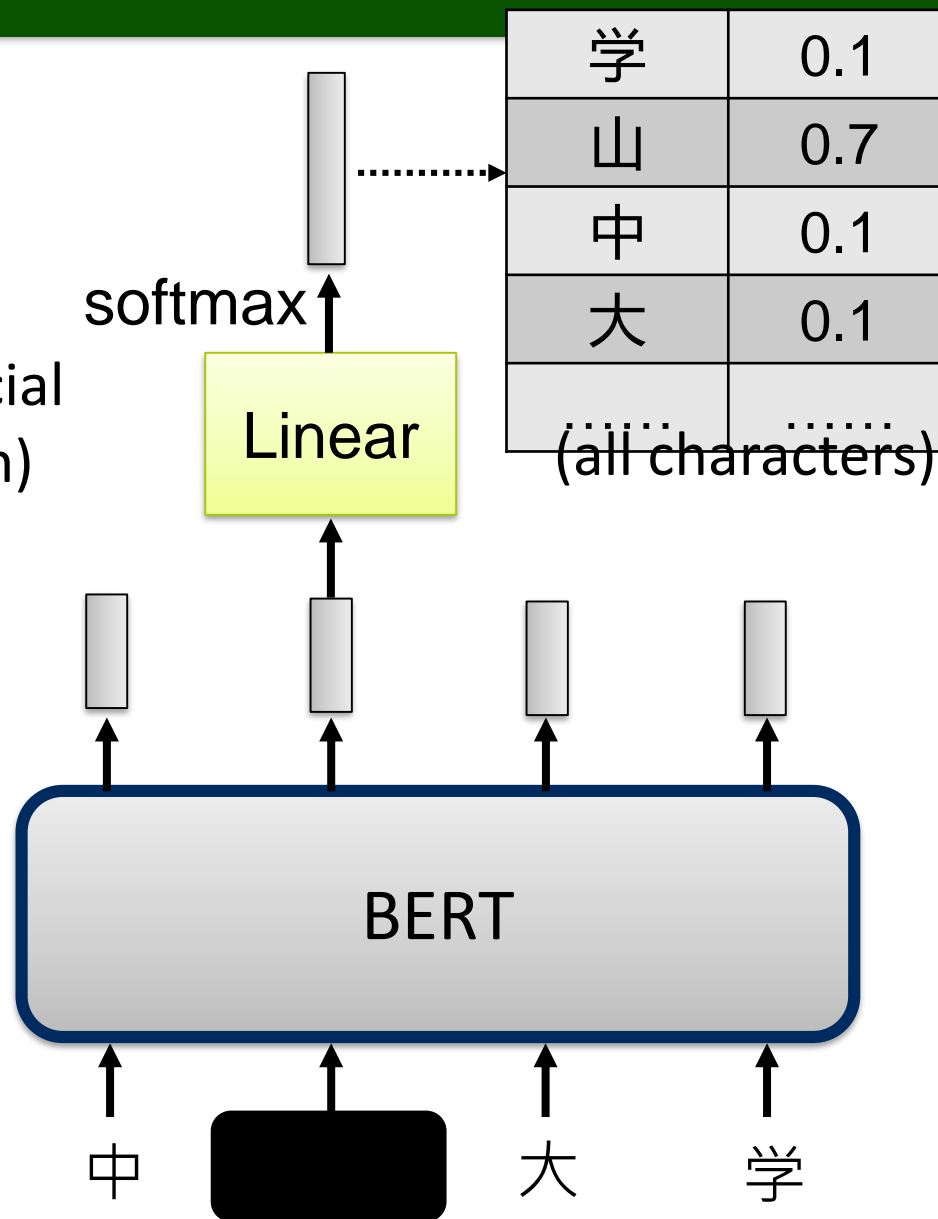
5

 =  (special token)  
or

 =   
一、天、大、小 ...



Transformer  
Encoder

Randomly  
masking some  
tokens



# Masking Input

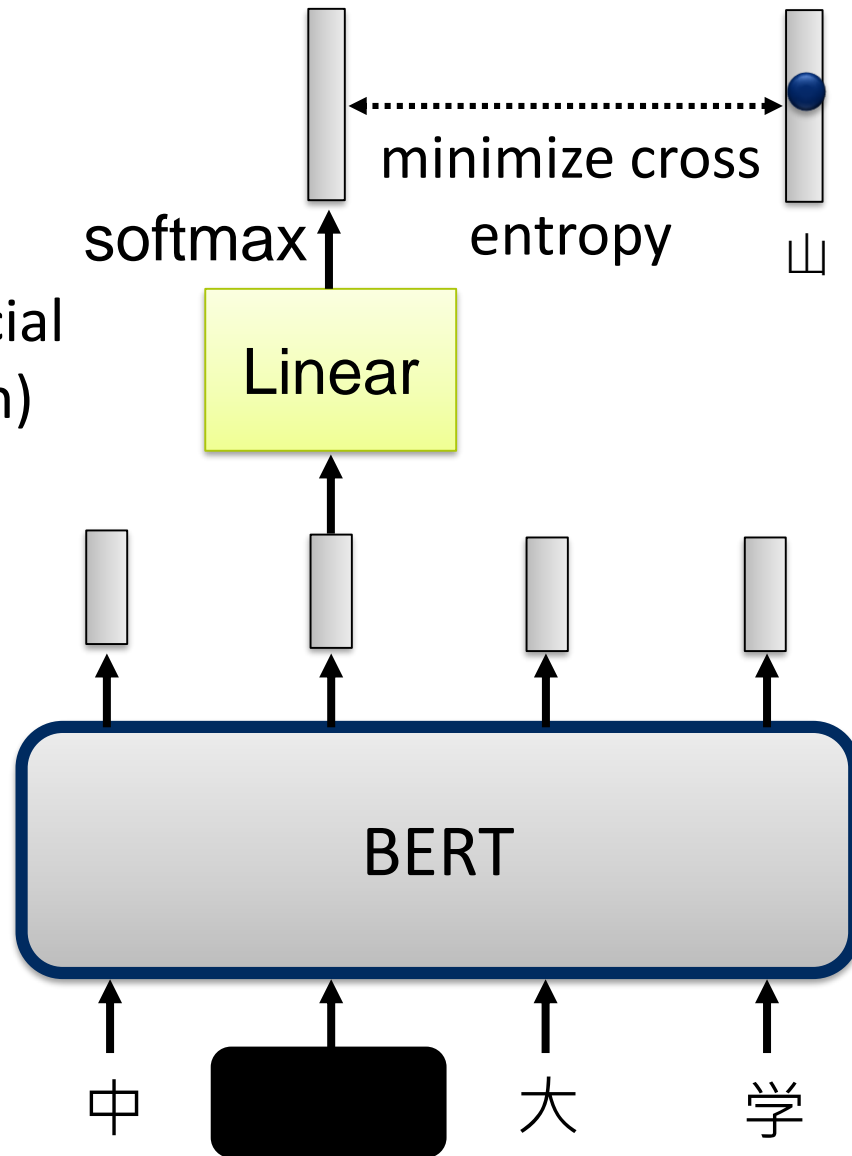
<https://arxiv.org/abs/1810.04805>

 =  (special token)  
or

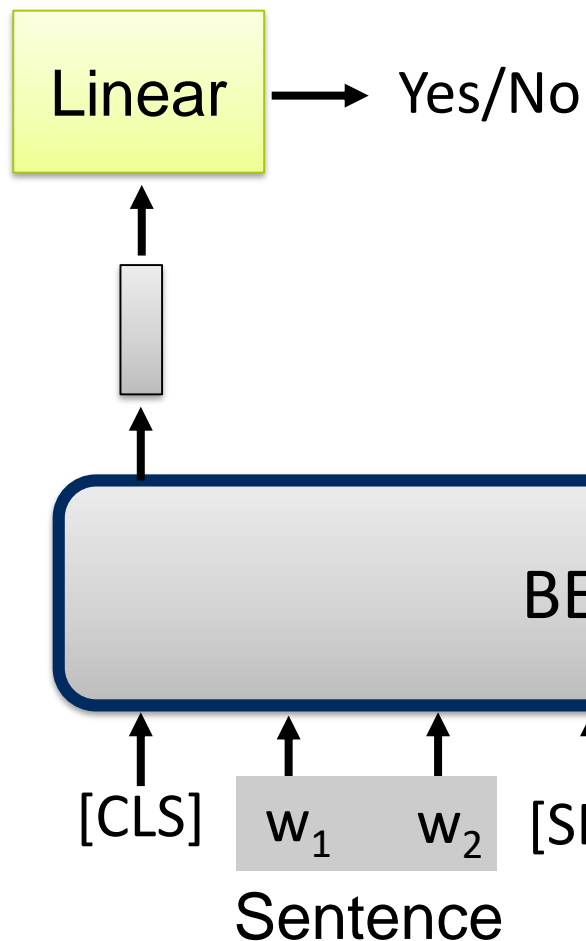
 =   
一、天、大、小 ...

Transformer  
Encoder

Randomly  
masking some  
tokens



# Next Sentence Prediction

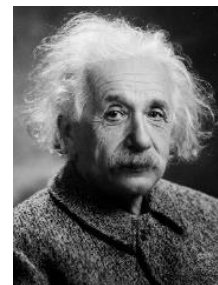


- This approach is not helpful.  
Robustly optimized BERT approach (RoBERTa)

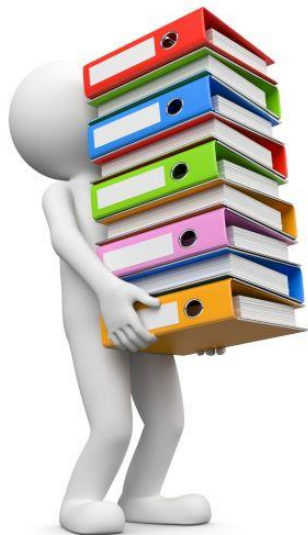
- **SOP**: Sentence order prediction

Used in ALBERT

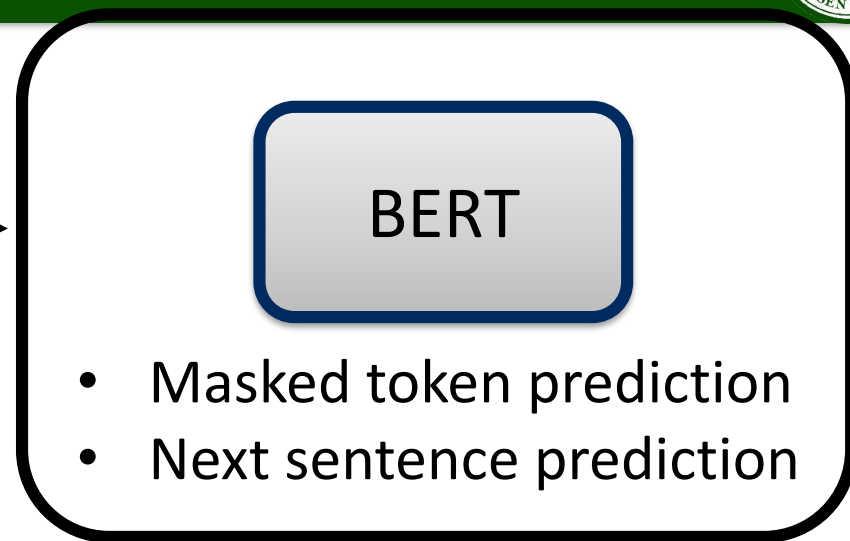
<https://arxiv.org/abs/1909.11942>



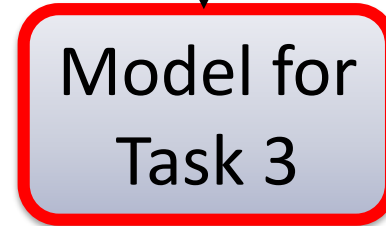
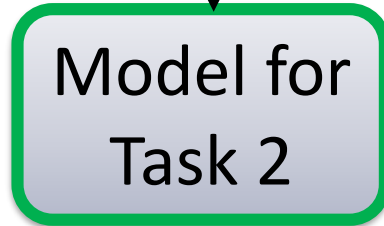
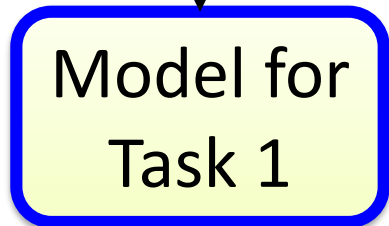
# BERT



Self-supervised  
Learning  
**Pre-train**



**Fine-tune**

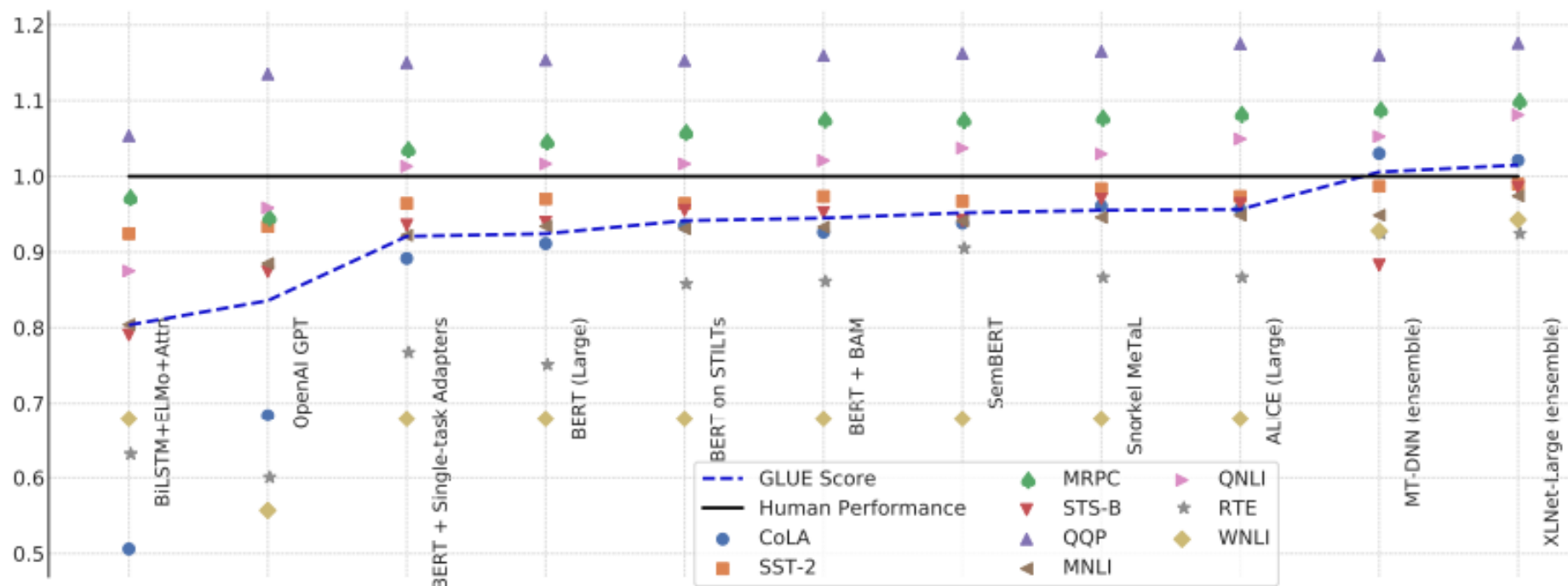


*Downstream Tasks*

- The tasks we care
- We have a little bit labeled data.

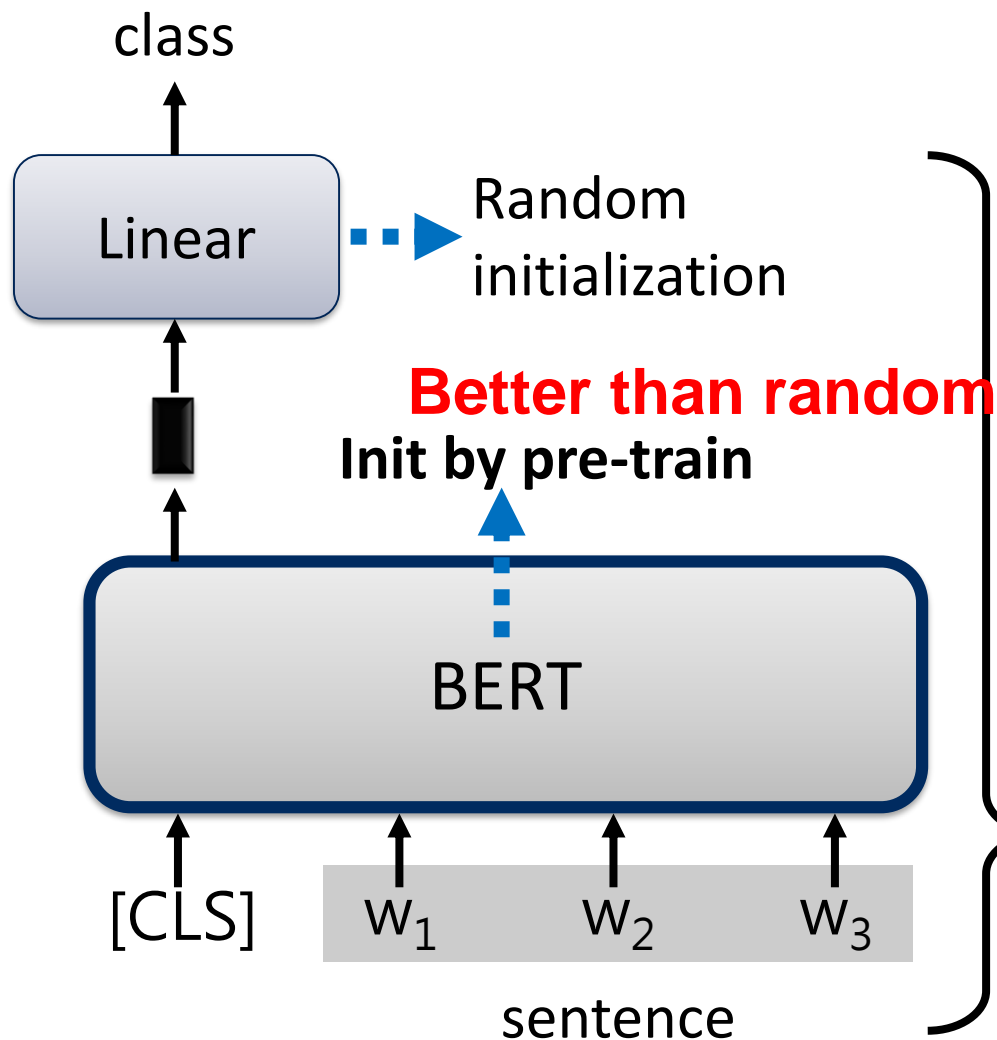
## BERT and its Family

GLUE scores



Source of image: <https://arxiv.org/abs/1905.00537>

# BERT

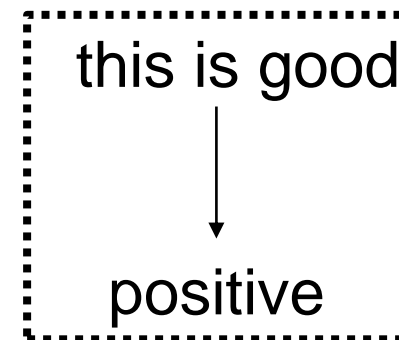


Input: sequence

output: class

Example:

Sentiment analysis

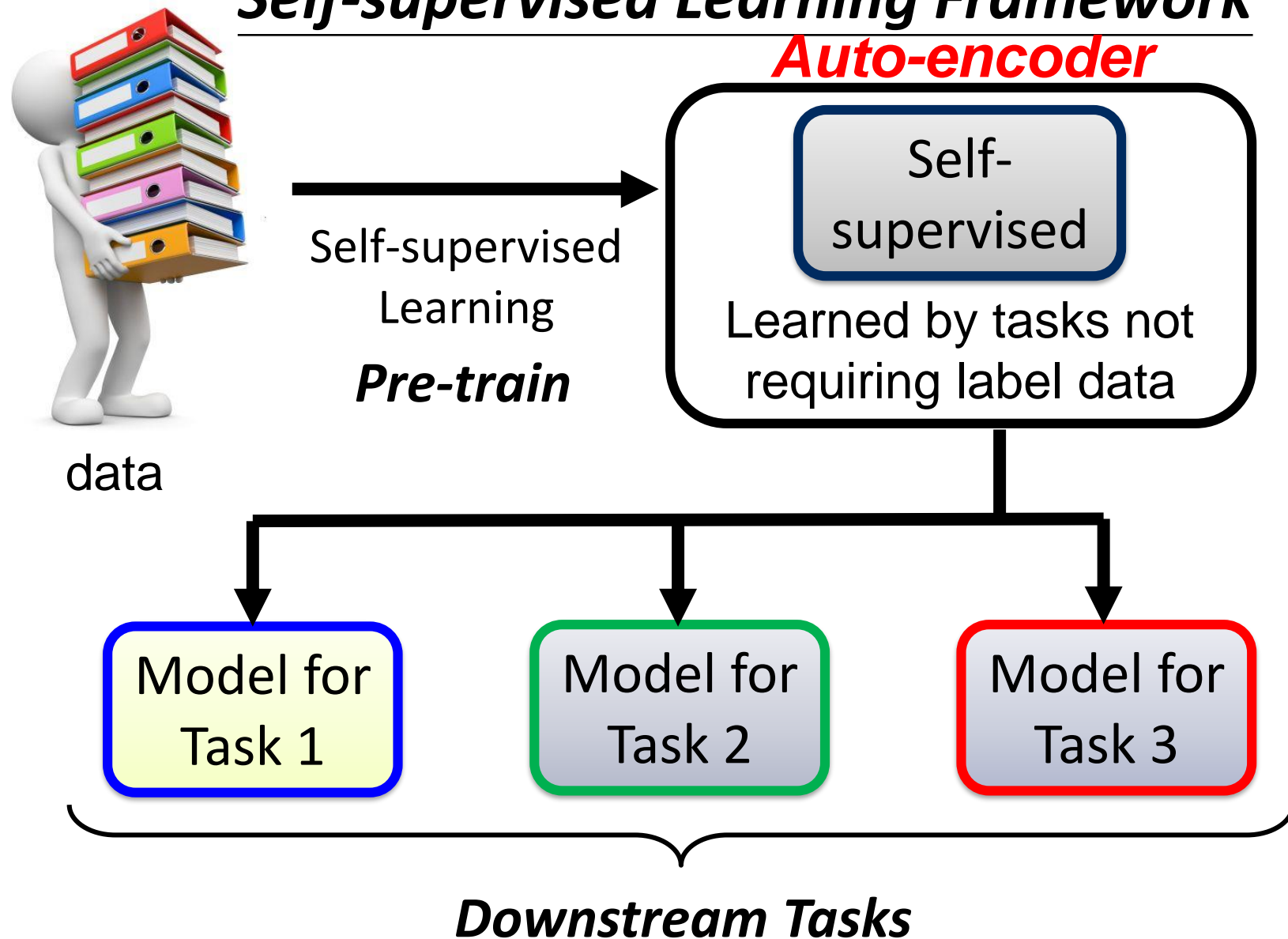


This is the model to be learned.

# VAE (Variational Auto Encoder)



## *Self-supervised Learning Framework*

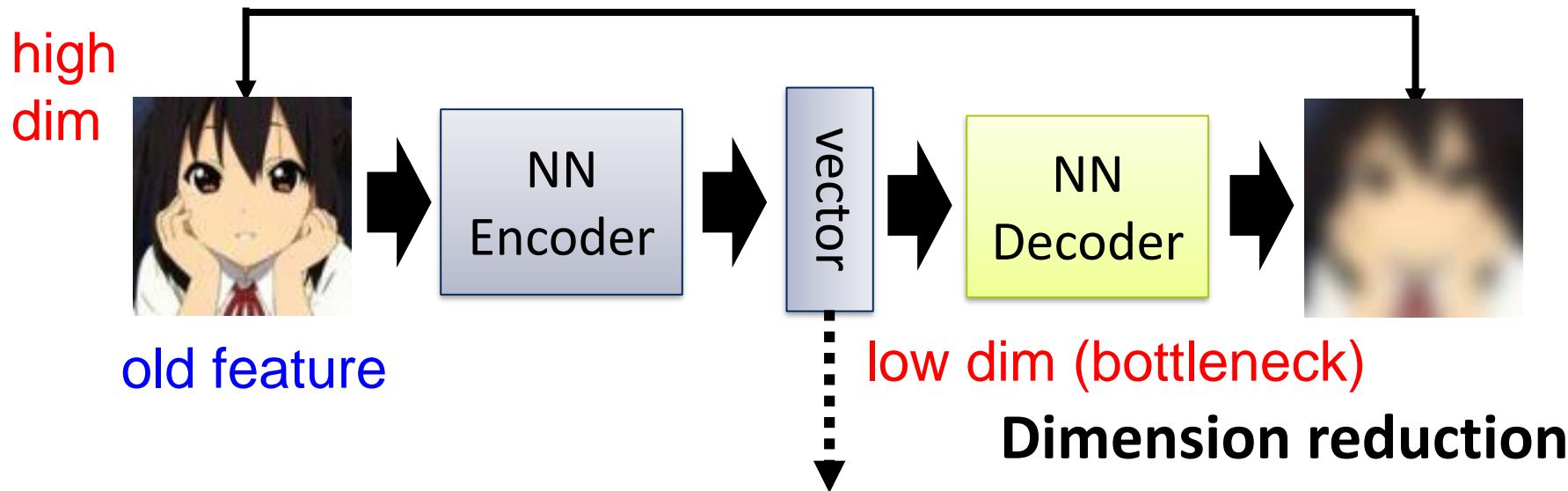




# Auto-encoder

Sounds familiar? We have seen the same idea in Cycle GAN. ☺

As close as possible (reconstruction)

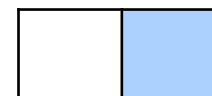
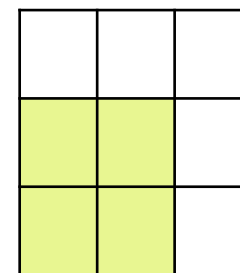
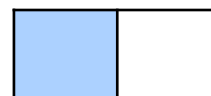
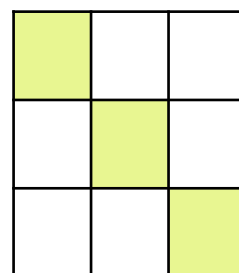
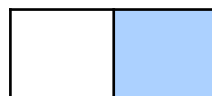
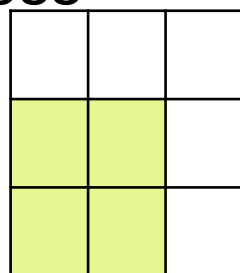
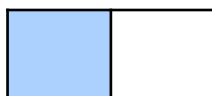
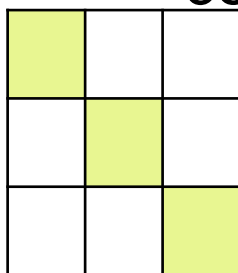
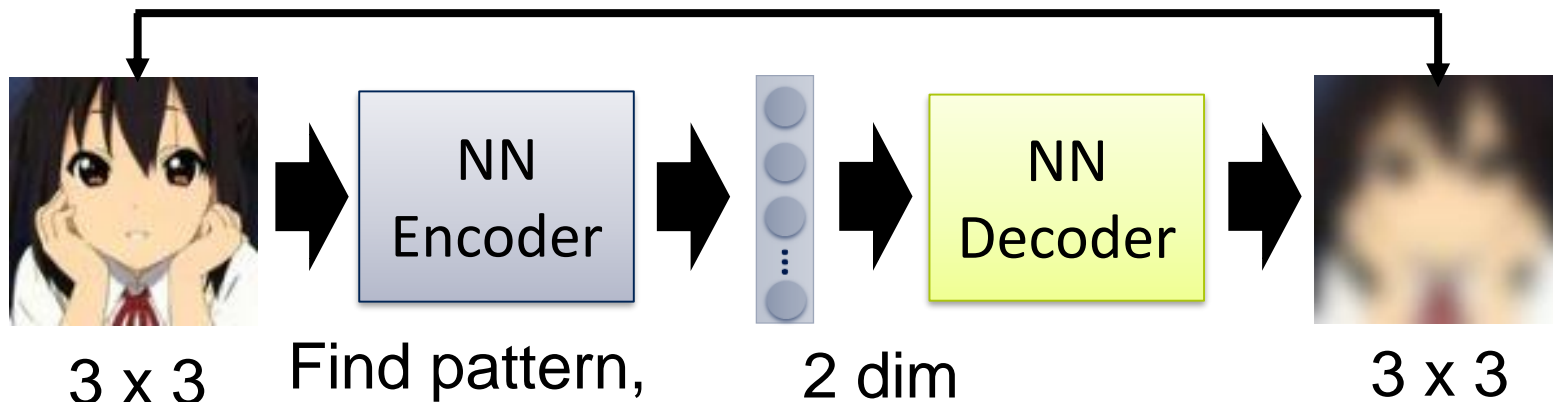


Embedding, Representation, Code

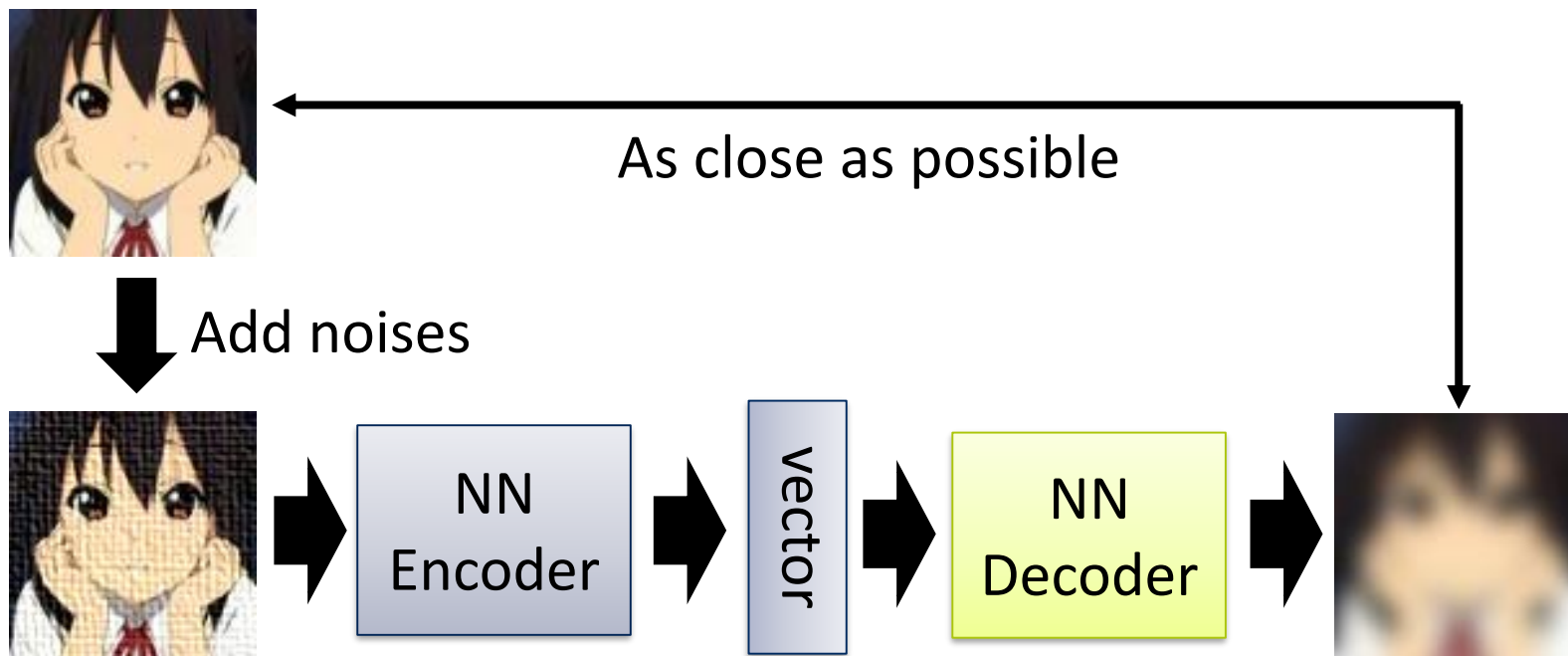
New feature for downstream tasks

# Why Auto-encoder?

As close as possible (reconstruction)



# De-noising Auto-encoder



The idea sounds familiar? 😊

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

## Motivation

You have trained many neural networks.

We seek to deploy neural networks in the real world.

Are networks robust to the inputs that are built to fool them?

Useful for spam classification  
malware detection, network  
intrusion detection, etc.

Aim to fool the network



## Example of Attack

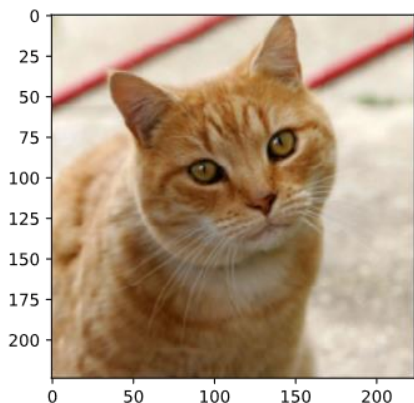
Non-targeted

Anything other than “Cat”

Targeted

Misclassified as a specific class (e.g., “Star Fish”)

Benign Image



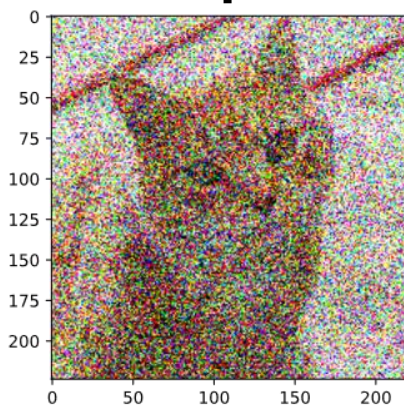
$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \end{bmatrix} + \begin{bmatrix} \Delta x_1 \\ \Delta x_2 \\ \Delta x_3 \\ \vdots \end{bmatrix}$$

small



Something Else

~~Tiger Cat~~

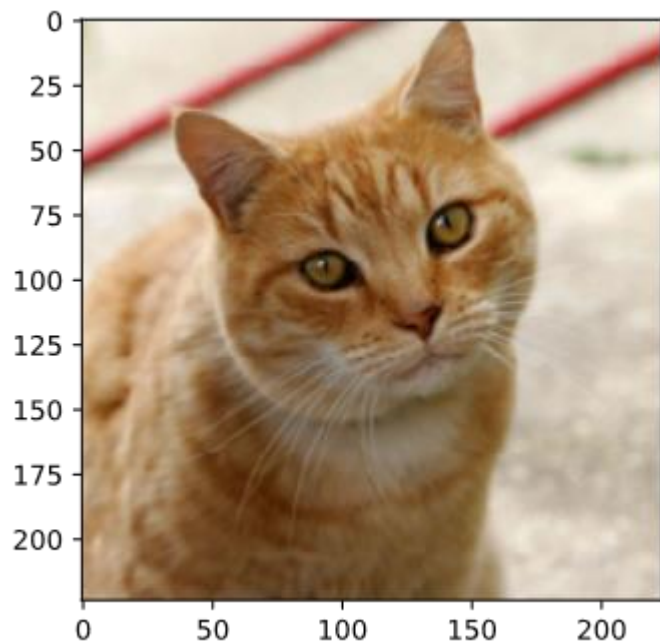


Attacked Image

## Example of Attack

The target is “Star Fish”

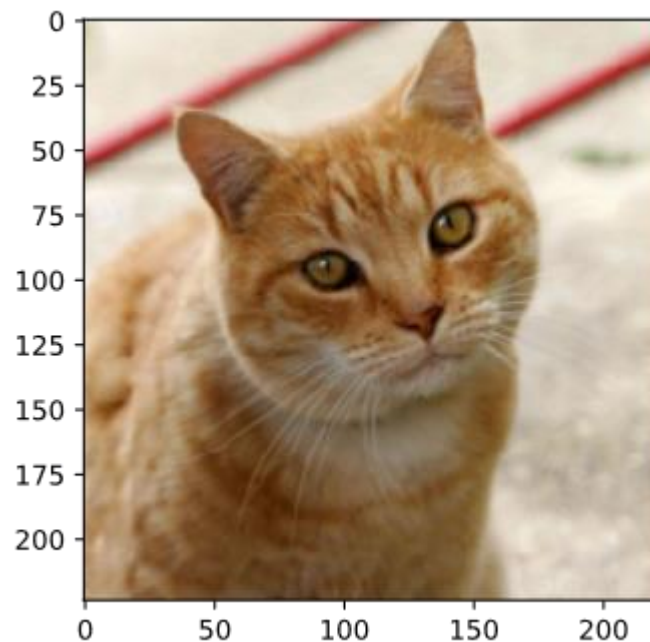
Benign Image



Tiger Cat

0.64

Attacked Image



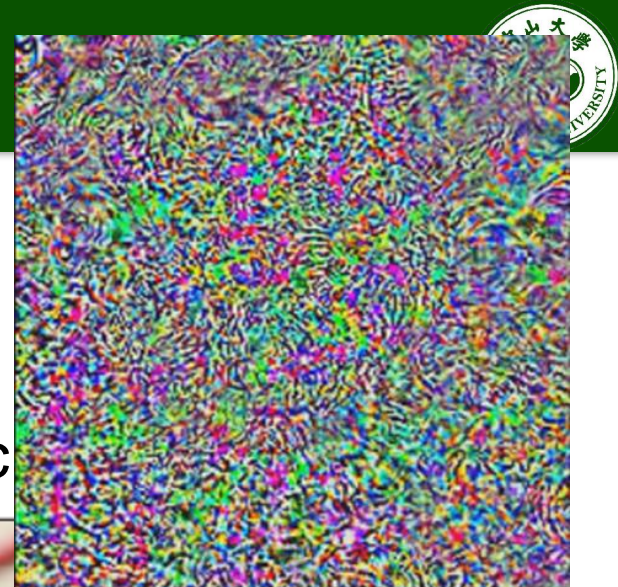
Star Fish

1.00

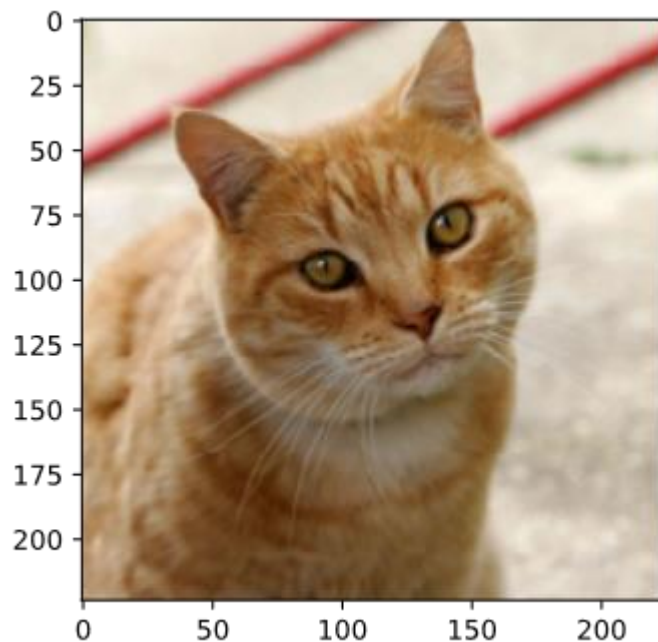


# Attack

## Example of Attack =



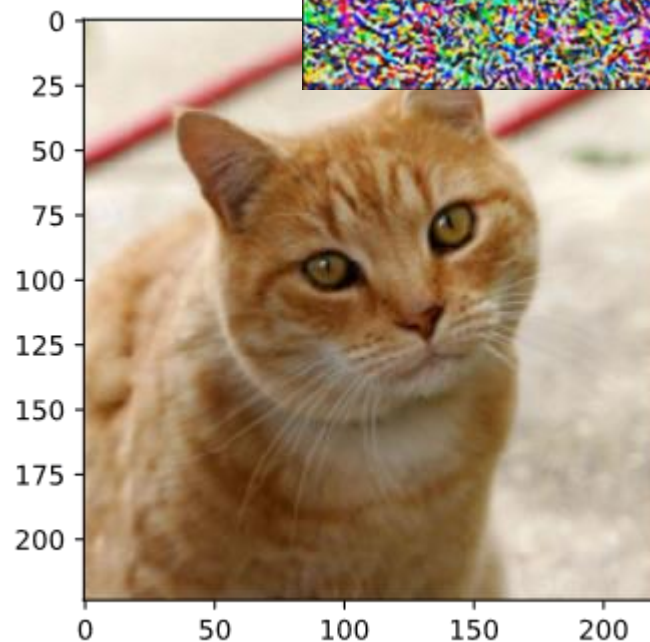
Benign Image



Tiger Cat

0.64

Attack



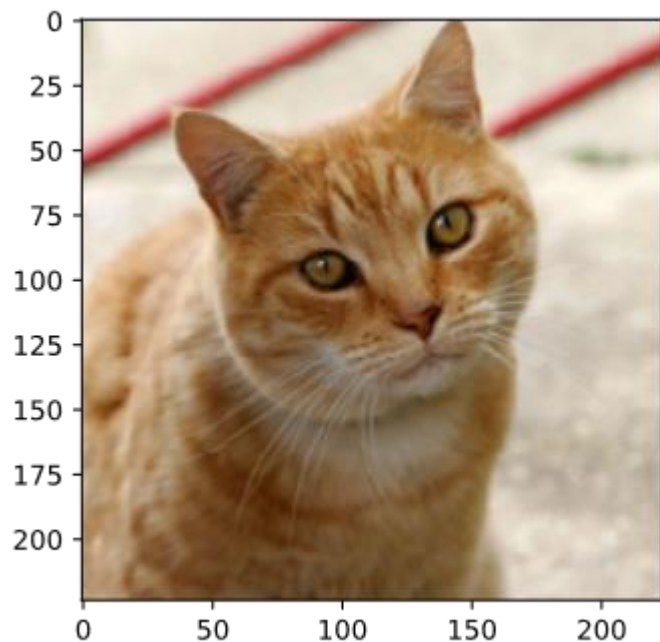
Star Fish

1.00

50  
x

## Example of Attack

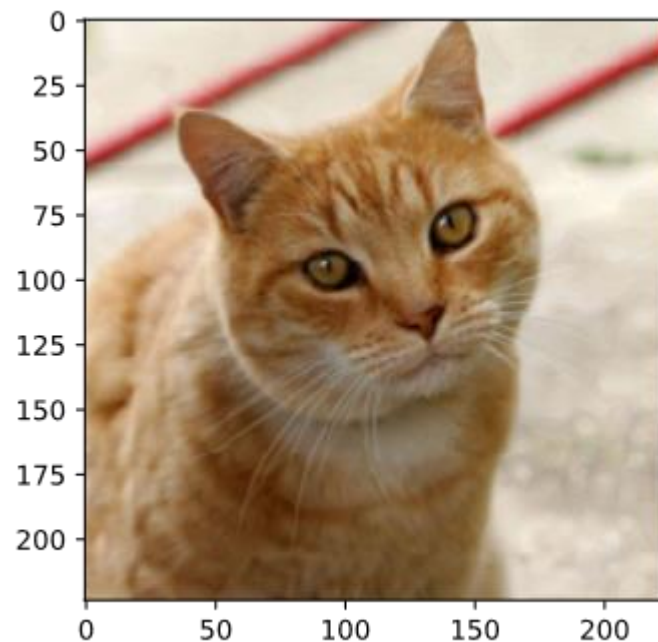
Benign Image



Tiger Cat

0.64

The target is  
“Keyboard”  
Attacked Image

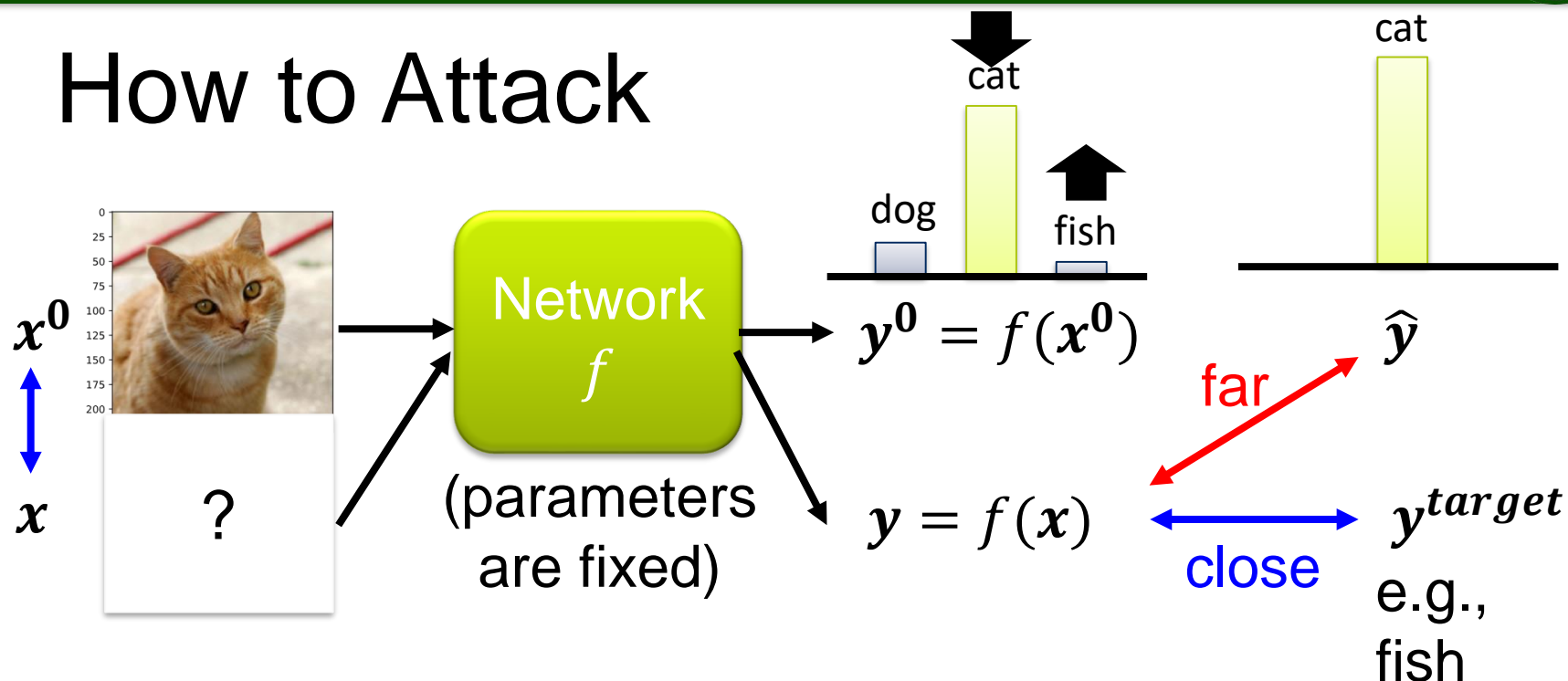


Keyboard

0.98



## How to Attack



**Non-targeted**

$$x^* = \arg \min L(x)$$

$$L(x) = -e(y, \hat{y})$$

not  
perceived by  
humans

**Targeted**

$$L(x) = -e(y, \hat{y}) + e(y, y^{target})$$

## White Box v.s. Black Box

In the previous attack, we know the network parameters  $\theta$

This is called **White Box Attack**.

You cannot obtain model parameters in most online API.

Are we safe if we do not release model?



No, because **Black Box Attack** is possible. ☹️

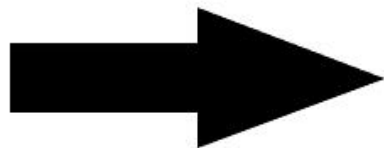
$$g = \begin{bmatrix} \text{sign} \left( \frac{\partial L}{\partial x_1} \Big|_{x=x^{t-1}} \right) \\ \text{sign} \left( \frac{\partial L}{\partial x_2} \Big|_{x=x^{t-1}} \right) \\ \vdots \end{bmatrix}$$

## Introduction

Diffusion Model are **generative models**.

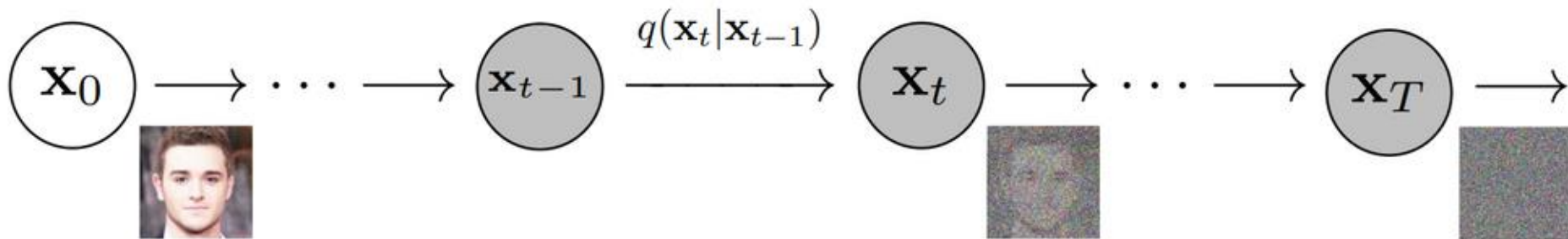
Diffusion Models work by **destroying training data** through the successive addition of Gaussian noise, and then **learning to recover** the data by reversing this noising process.

After training, the Diffusion Model can be used to generate data by simply **passing randomly sampled noise through the learned denoising process**.



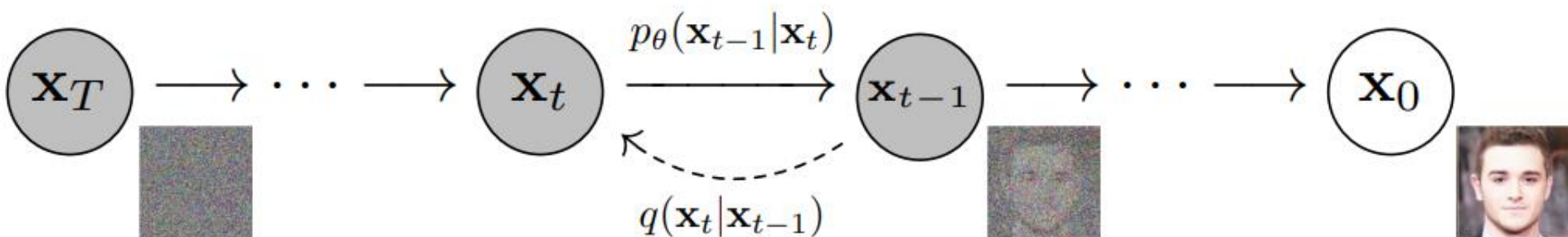
## Introduction

More specifically, a Diffusion Model is **a latent variable model** which maps to the latent space using a **fixed Markov chain**. This chain gradually adds noise to the data in order to obtain the approximate posterior  $q(x_{1:T}|x_0)$ , where  $x_1, \dots, x_T$  are the latent variables with the same dimensionality as  $x_0$ . In the figure below, we see such a Markov chain manifested for image data.



## Introduction

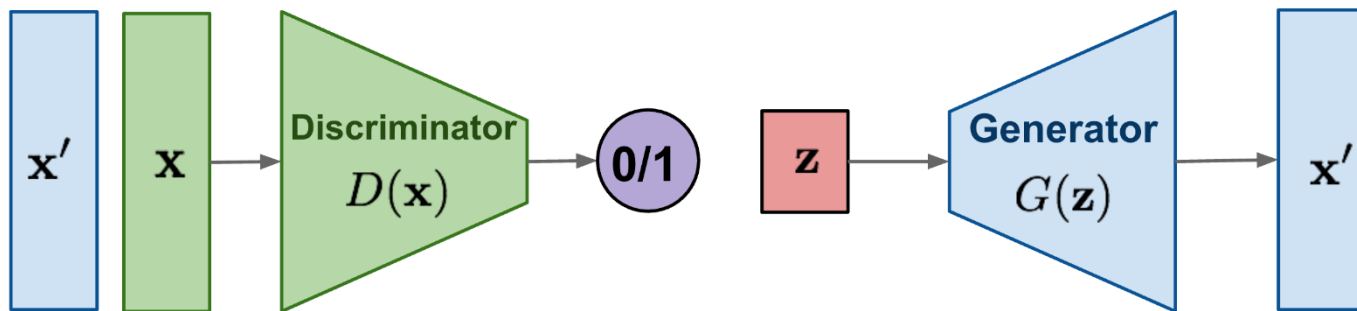
Ultimately, the image is asymptotically transformed to pure Gaussian noise. The **goal** of training a diffusion model is to learn the **reverse** process - i.e. training  $P_{\theta}(x_{t-1}|x_T)$ . By traversing backwards along this chain, we can generate new data.



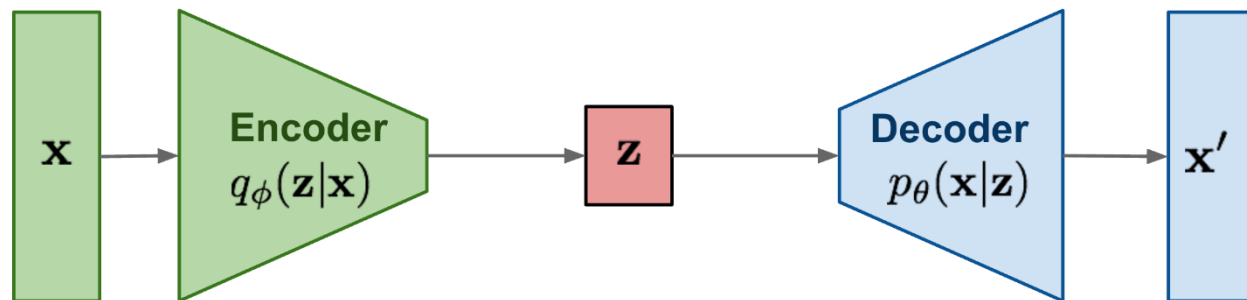
## Introduction

Comparing Diffusion model with GAN and VAE

**GAN:** Adversarial training

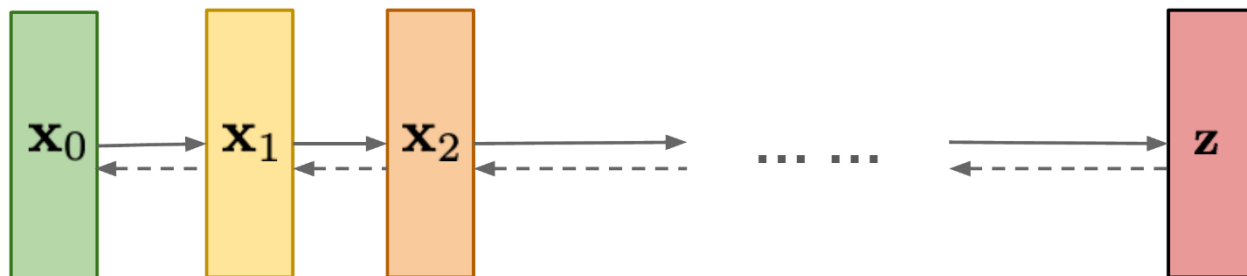


**VAE:** maximize variational lower bound



**Diffusion models:**

Gradually add Gaussian noise and then reverse



## Benefits

**State-of-the-Art image quality**

**not requiring adversarial training**

**comparable performance and training efficiency**

**scalability and parallelizability**



# Thanks