AI VIETNAM
All-in-One Course

Introduction to OpenAI's Sora

Video generated by Sora

Sora: A Review on Background, Technology, Limitations, and Opportunities of Large Vision Models

Yixin Liu[1*] Kai Zhang[1*] Yuan Li[1*] Zhiling Yan[1*] Chujie Gao[1*]
Ruoxi Chen[1*] Zhengqing Yuan[1*] Yue Huang[1*] Hanchi Sun[1*]
Jianfeng Gao[2] Lifang He[1] Lichao Sun[1†]

[1]Lehigh University [2]Microsoft Research

Abstract

Warning: This is not an official technical report from OpenAI.

Sora is a text-to-video generative AI model, released by OpenAI in February 2024. The model is trained to generate videos of realistic or imaginative scenes from text instructions and show potential in simulating the physical world. Based on public technical reports and reverse engineering, this paper presents a comprehensive review of the model's background, related technologies, applications, remaining challenges, and future directions of text-to-video AI models. We first trace Sora's development and investigate the underlying technologies used to build this "world simulator". Then, we describe in detail the applications and potential impact of Sora in multiple industries ranging from film-making and education to marketing. We discuss the main challenges and limitations that need to be addressed to widely deploy Sora, such as ensuring safe and unbiased video generation. Lastly, we discuss the future development of Sora and video generation models in general, and how advancements in the field could enable new ways of human-AI interaction, boosting productivity and creativity of video generation.

Vinh Dinh Nguyen
PhD in Computer Science

Outline

- Objective
- Diffusion Probability Model
- Transformer and ChatGPT Architecture
- Vision Transformer Architecture
- Diffusion and Transformer
- OpenAI's Sora Architecture
- Summary

Outline

- Objective
- Diffusion Probability Model
- Transformer and ChatGPT Architecture
- Vision Transformer Architecture
- Diffusion and Transformer
- OpenAI's Sora Architecture
- Summary

Objective

Sora: A Review on Background, Technology, Limitations, and Opportunities of Large Vision Models

Yixin Liu[1*] Kai Zhang[1*] Yuan Li[1*] Zhiling Yan[1*] Chujie Gao[1*]
Ruoxi Chen[1*] Zhengqing Yuan[1*] Yue Huang[1*] Hanchi Sun[1*]
Jianfeng Gao[2] Lifang He[1] Lichao Sun[1†]

[1]Lehigh University [2]Microsoft Research

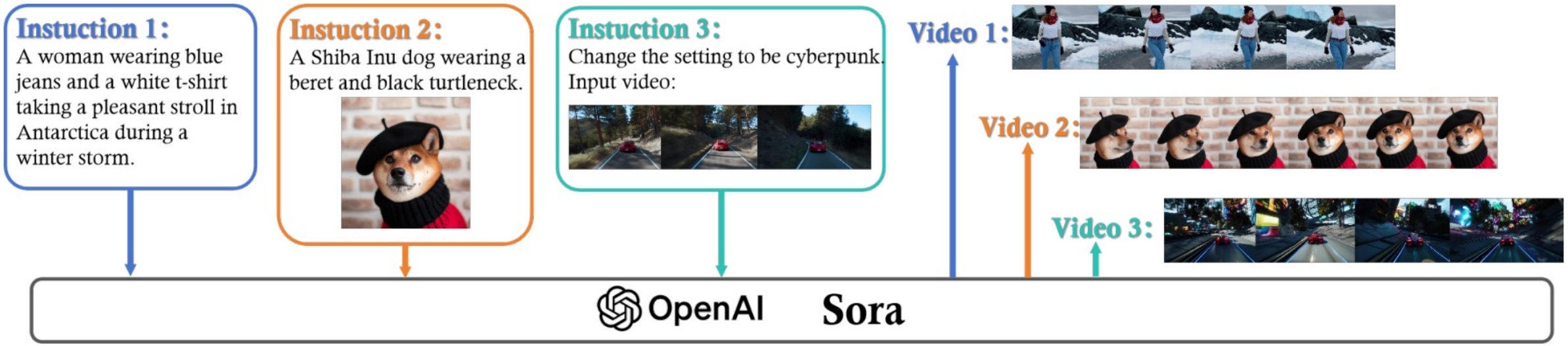
Abstract

Warning: This is not an official technical report from OpenAI.

Sora is a text-to-video generative AI model, released by OpenAI in February 2024. The model is trained to generate videos of realistic or imaginative scenes from text instructions and show potential in simulating the physical world. Based on public technical reports and reverse engineering, this paper presents a comprehensive review of the model's background, related technologies, applications, remaining challenges, and future directions of text-to-video AI models. We first trace Sora's development and investigate the underlying technologies used to build this "world simulator". Then, we describe in detail the applications and potential impact of Sora in multiple industries ranging from film-making and education to marketing. We discuss the main challenges and limitations that need to be addressed to widely deploy Sora, such as ensuring safe and unbiased video generation. Lastly, we discuss the future development of Sora and video generation models in general, and how advancements in the field could enable new ways of human-AI interaction, boosting productivity and creativity of video generation.

1. Understand a Diffusion Model
2. Understand a Transformer Architecture
3. Understand a Vision Transformer
4. Understand a Diffusion Transformer
5. Understand a Sora Architecture

What Can OpenAI' Sora Does?

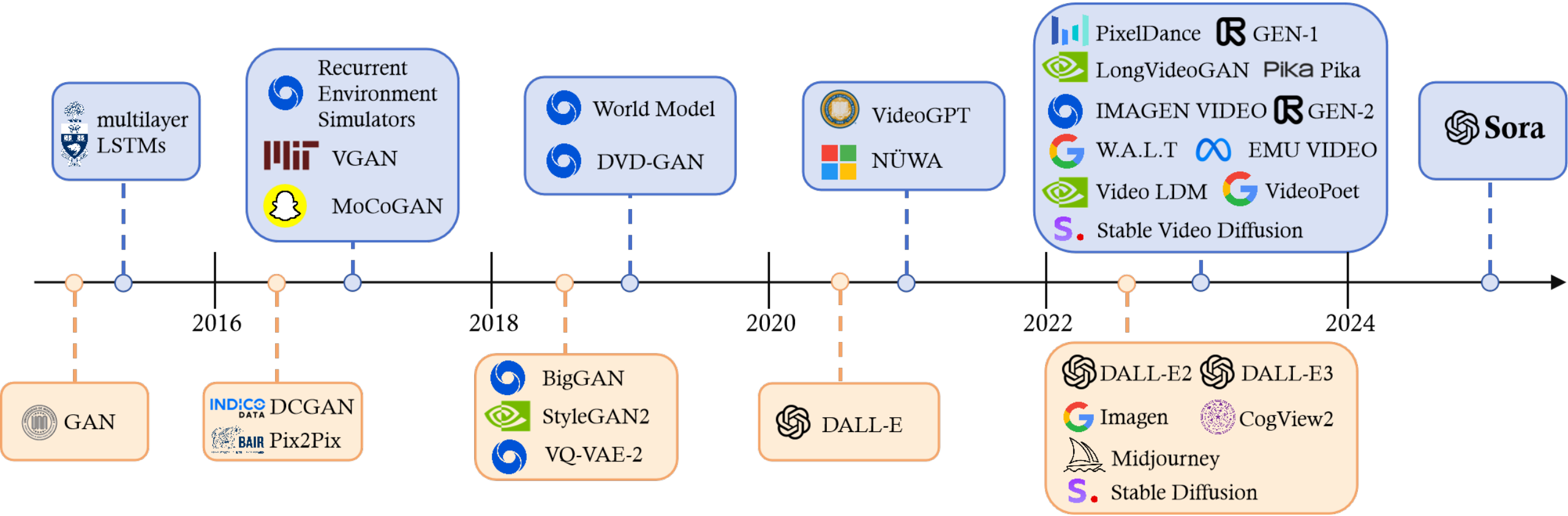


Demo: https://seo.ai/blog/openai-sora-examples

Some related works about the video generation tasks

Model name	Year	Backbone	Task	Group
Imagen Video[29]	2022	Diffusion	Generation	Google
Pix2Seq-D[160]	2022	Diffusion	Segmentation	Google Deepmind
FDM[161]	2022	Diffusion	Prediction	UBC
MaskViT[162]	2022	Masked Vision Models	Prediction	Stanford, Salesforce
CogVideo[163]	2022	Auto-regressive	Generation	THU
Make-a-video[164]	2022	Diffusion	Generation	Meta
MagicVideo[165]	2022	Diffusion	Generation	ByteDance
TATS[166]	2022	Auto-regressive	Generation	University of Maryland, Meta
Phenaki[167]	2022	Masked Vision Models	Generation	Google Brain
Gen-1[168]	2023	Diffusion	Generation, Editing	RunwayML
LFDM[140]	2023	Diffusion	Generation	PSU, UCSD
Text2video-Zero[169]	2023	Diffusion	Generation	Picsart
Video Fusion[170]	2023	Diffusion	Generation	USAC, Alibaba
PYoCo[34]	2023	Diffusion	Generation	Nvidia
Video LDM[36]	2023	Diffusion	Generation	University of Maryland, Nvidia
RIN[171]	2023	Diffusion	Generation	Google Brain
LVD[172]	2023	Diffusion	Generation	UCB
Dreamix[173]	2023	Diffusion	Editing	Google
MagicEdit[174]	2023	Diffusion	Editing	ByteDance
Control-A-Video[175]	2023	Diffusion	Editing	Sun Yat-Sen University
StableVideo[176]	2023	Diffusion	Editing	ZJU, MSRA
Tune-A-Video[78]	2023	Diffusion	Editing	NUS
Rerender-A-Video[177]	2023	Diffusion	Editing	NTU
Pix2Video[178]	2023	Diffusion	Editing	Adobe, UCL
InstructVid2Vid[179]	2023	Diffusion	Editing	ZJU
DiffAct[180]	2023	Diffusion	Action Detection	University of Sydney
DiffPose[181]	2023	Diffusion	Pose Estimation	Jilin University
MAGVIT[182]	2023	Masked Vision Models	Generation	Google
AnimateDiff[138]	2023	Diffusion	Generation	CUHK
MAGVIT V2[47]	2023	Masked Vision Models	Generation	Google
Generative Dynamics[183]	2023	Diffusion	Generation	Google
VideoCrafter[81]	2023	Diffusion	Generation	Tencent
Zeroscope[184]	2023	-	Generation	EasyWithAI
ModelScope	2023	-	Generation	Damo
Gen-2[23]	2023	-	Generation	RunwayML
Pika[22]	2023	-	Generation	Pika Labs
Emu Video[185]	2023	Diffusion	Generation	Meta
PixelDance[186]	2023	Diffusion	Generation	ByteDance
Stable Video Diffusion[27]	2023	Diffusion	Generation	Stability AI
W.A.L.T[187]	2023	Diffusion	Generation	Stanford, Google
Fairy[188]	2023	Diffusion	Generation, Editing	Meta
VideoPoet[189]	2023	Auto-regressive	Generation, Editing	Google
LGVI[190]	2024	Diffusion	Editing	PKU, NTU
Lumiere[191]	2024	Diffusion	Generation	Google
Sora[3]	2024	Diffusion	Generation, Editing	OpenAI

History of Generative AI in Vision Domain



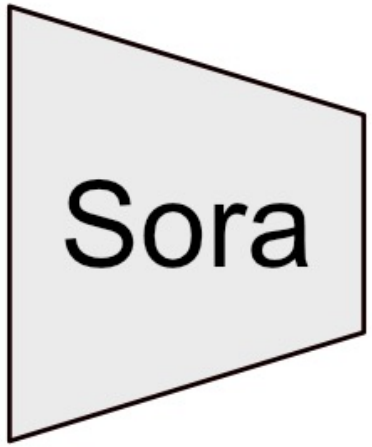
Introduction to Open AI's Sora

An example of the video quality Sora is capable of producing

Sora's Definition

Sora is a generative text to video model. Basically, it's a machine learning model that takes in text and spits out video.

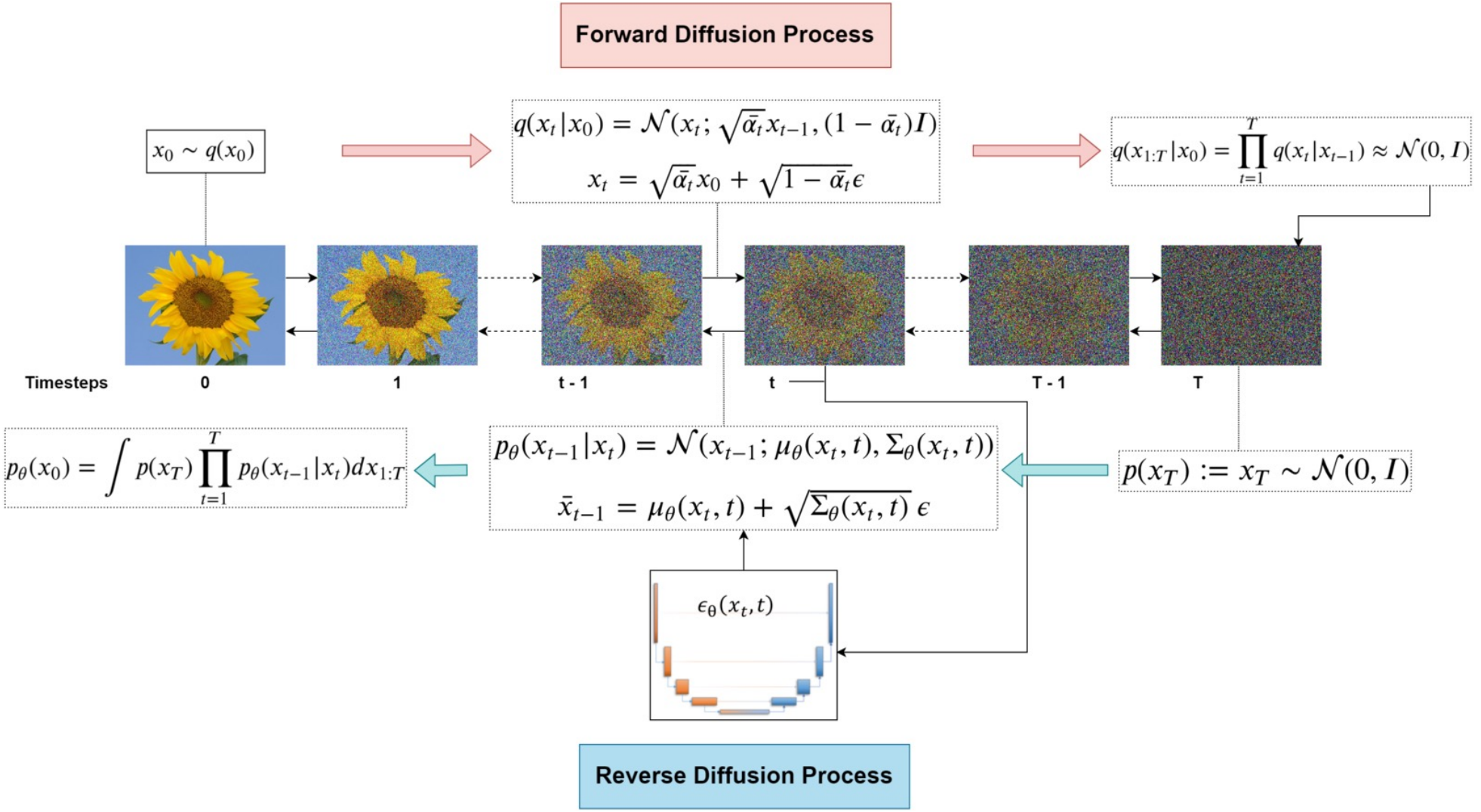
The camera follows behind a white vintage SUV with a black roof rack as it speeds up a steep dirt road surrounded by pine trees on a steep mountain slope, dust kicks up from it's tires, the sunlight shines on the SUV as it speeds along the dirt road, casting a warm glow over the scene. The dirt road curves gently into the distance, with no other cars or vehicles in sight. The trees on either...



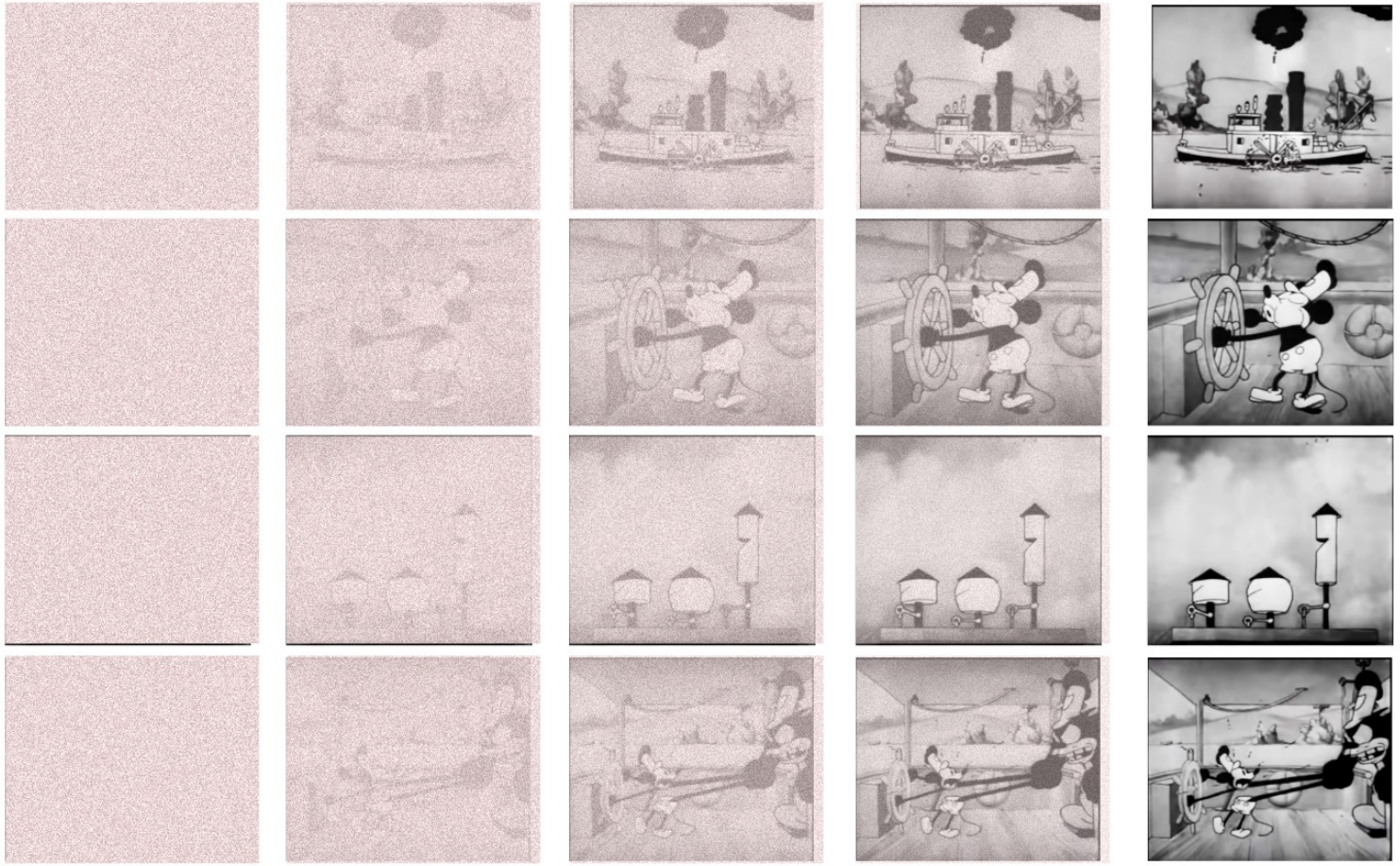
Outline

- Objective
- Diffusion Probability Model
- Transformer and ChatGPT Architecture
- Vision Transformer Architecture
- Diffusion and Transformer
- OpenAI's Sora Architecture
- Summary

Diffusion Model: Review



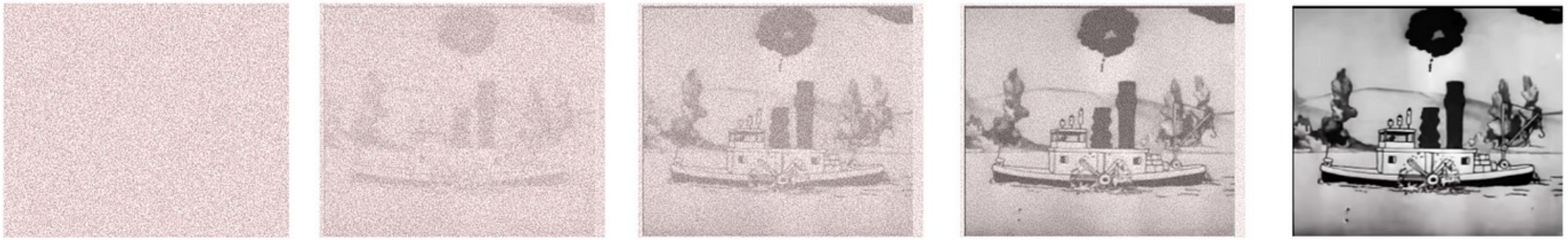
Diffusion Model: Review



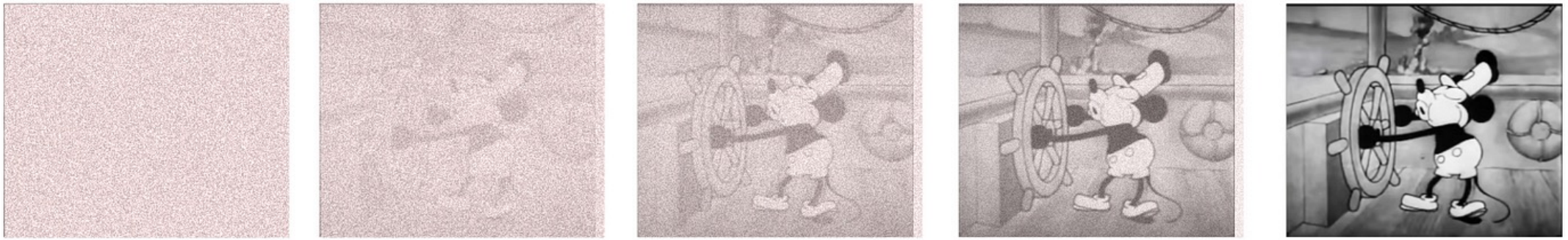
Once the model learns to get really good at turning noise into images, it can be used to generate new images based on random noise.

Diffusion Model: Review

A steam boat blowing a puff of smoke



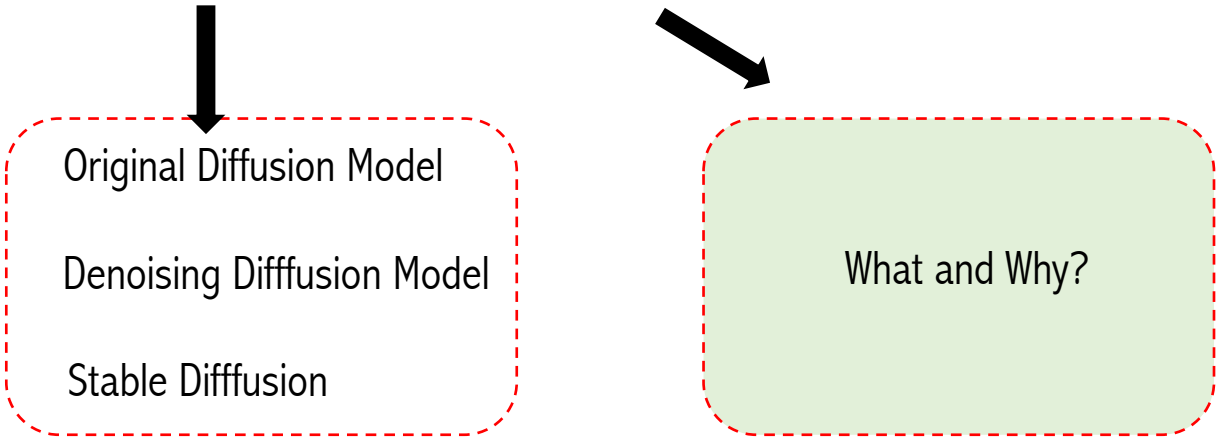
A mouse whistling while piloting a ship



A conceptual diagram of a diffusion model being trained to generate images based on a caption.

Sora: Motivation

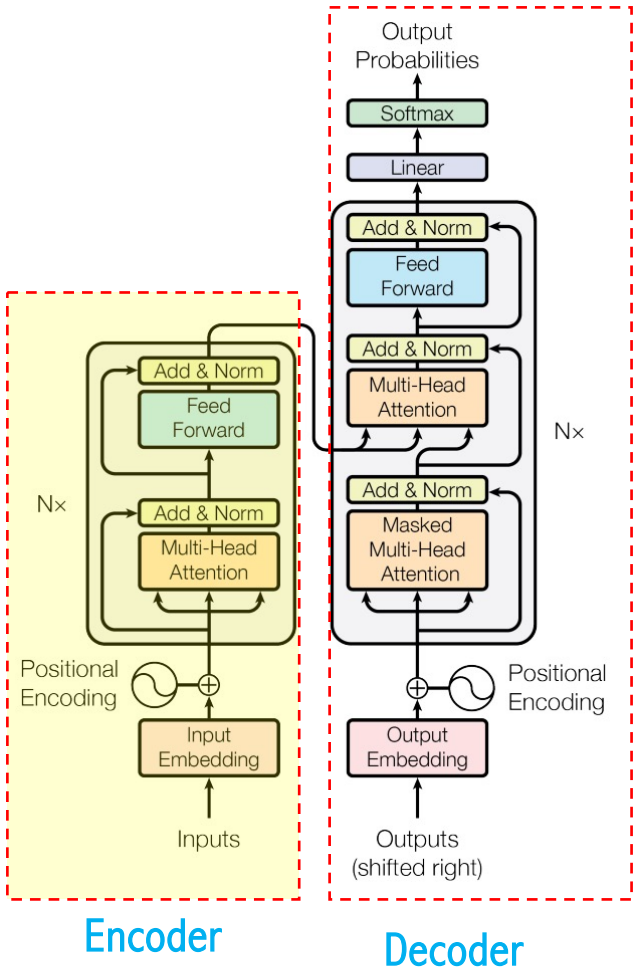
The idea of a diffusion model in a nutshell; they take in text, and use that text to turn noise into images. Sora uses a variation of the diffusion model called a "Diffusion Transformer"



Outline

- Objective
- Diffusion Probability Model
- Transformer and ChatGPT Architecture
- Vision Transformer Architecture
- Diffusion and Transformer
- OpenAI's Sora Architecture
- Summary

Transformer Motivation for OD



Input

Output

English → Vietnamese

I watch movie → Tôi xem phim

Word embeddeding

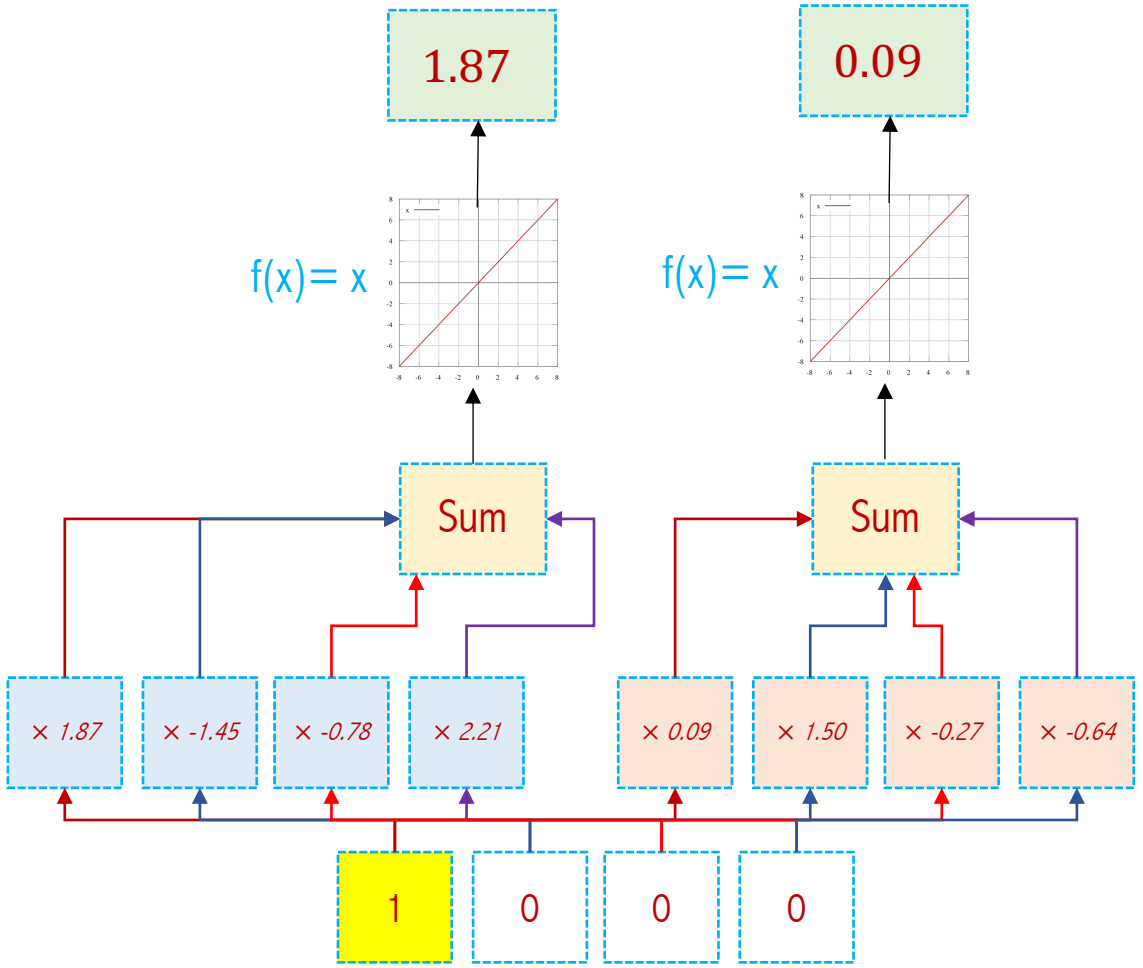
1.9 0.1
-1.5 1.5
1.6 -1.6

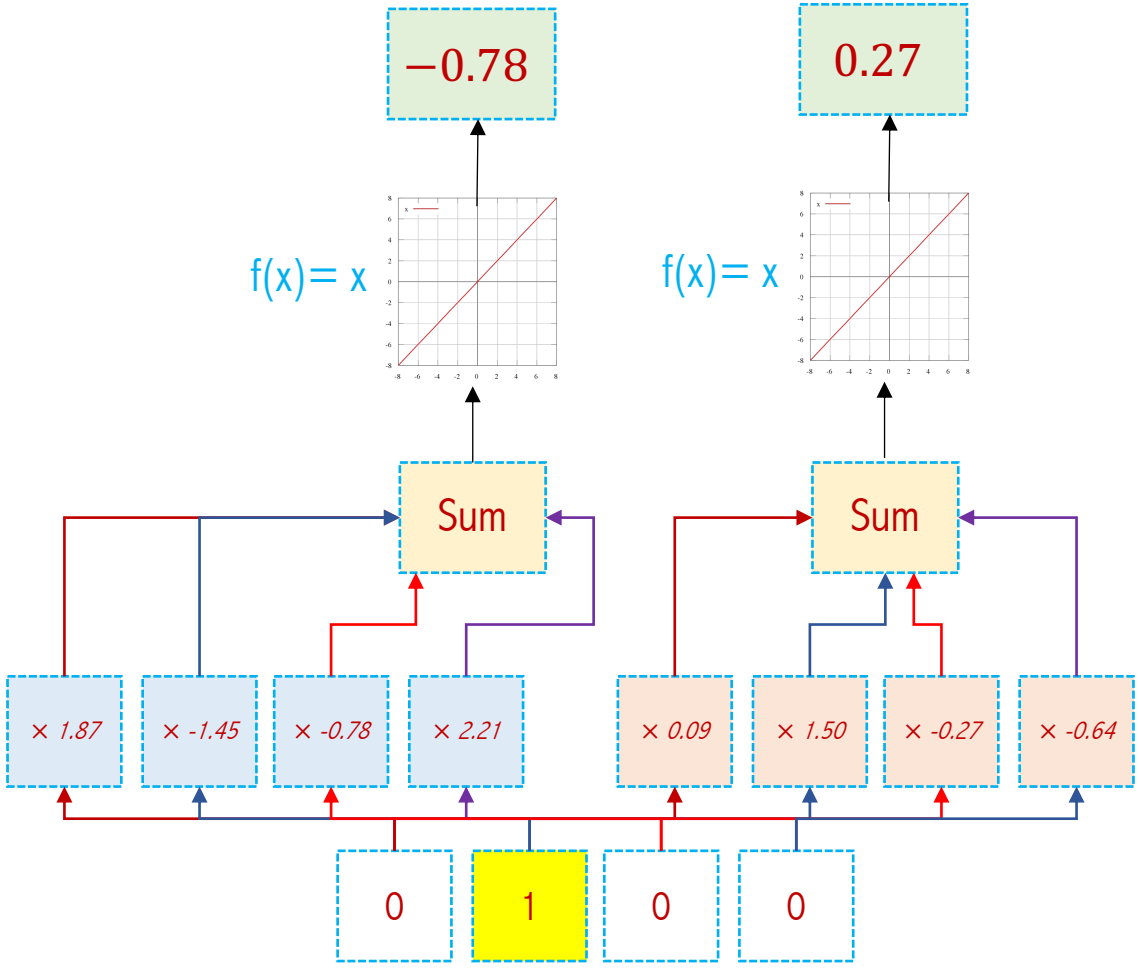
→

0.0 2.0
1.2 1.2
1.7 1.8

Transformer Encoder

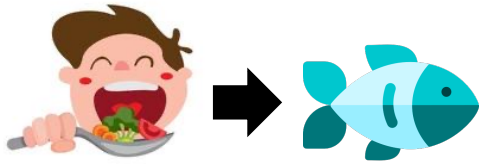
Word Embedding Using Neural Network



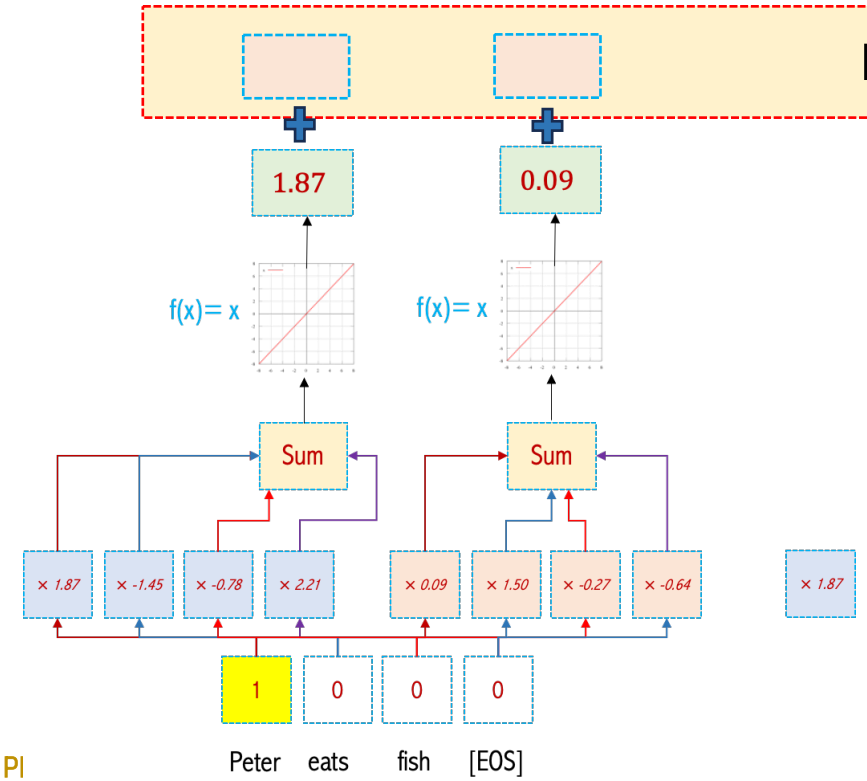


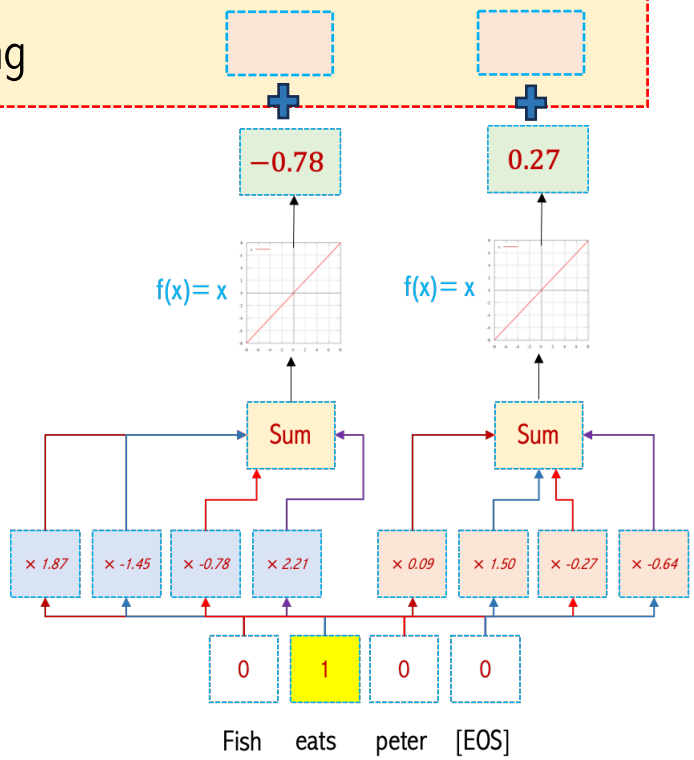
Word Ordering

Vinh eats fish



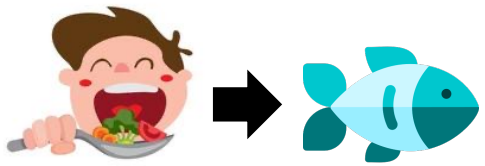
Fish eats Vinh





Word Ordering

Peter eats fish



$$P(k, 2i) = \sin\left(\frac{k}{n^{2i/d}}\right)$$

$$P(k, 2i+1) = \cos\left(\frac{k}{n^{2i/d}}\right)$$

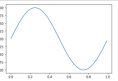
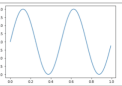
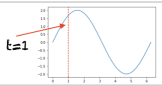
```python
import numpy as np
import matplotlib.pyplot as plt

def getPositionEncoding(seq_len, d, n=10000):
    P = np.zeros((seq_len, d))
    for k in range(seq_len):
        for i in np.arange(int(d/2)):
            denominator = np.power(n, 2*i/d)
            P[k, 2*i] = np.sin(k/denominator)
            P[k, 2*i+1] = np.cos(k/denominator)
    return P

P = getPositionEncoding(seq_len=4, d=4, n=100)
print(P)
```

Sequence	Index of token	Positional Encoding Matrix			
I	0	P_{00}	P_{01}	…	P_{0d}
am	1	P_{10}	P_{11}	…	P_{1d}
a	2	P_{20}	P_{21}	…	P_{2d}
Robot	3	P_{30}	P_{31}	…	P_{3d}

Positional Encoding Matrix for the sequence 'I am a robot'

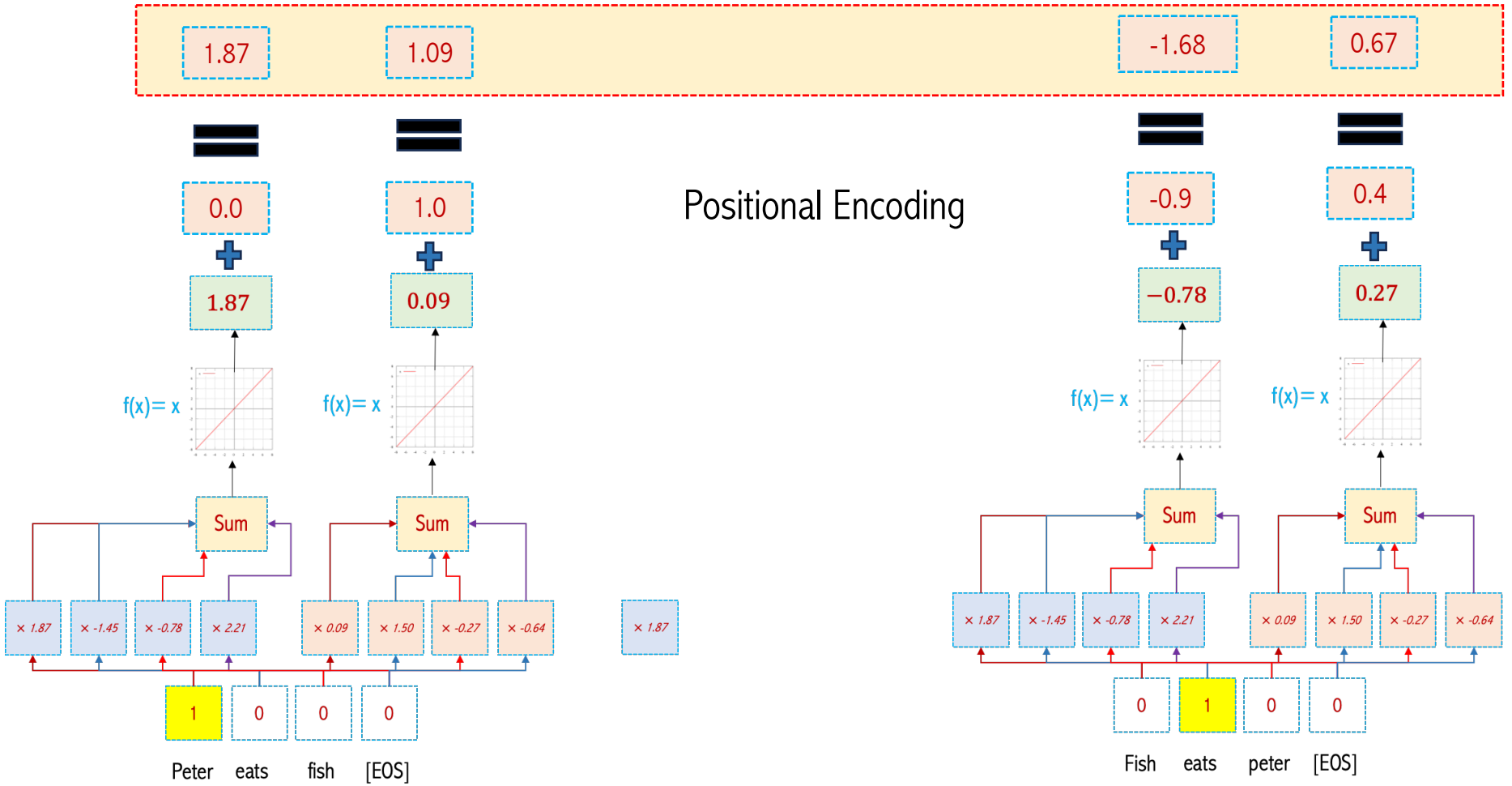
Equation	Graph	Frequency	Wavelength
$\sin(2\pi t)$		1	1
$\sin(2 * 2\pi t)$		2	1/2
$\sin(t)$		$1/2\pi$	2π
$\sin(ct)$	Depends on c	$c/2\pi$	$2\pi/c$

Sequence	Index of token, k	Positional Encoding Matrix with d=4, n=100			
		i=0	i=0	i=1	i=1
I	0	P_{00}=sin(0) = 0	P_{01}=cos(0) = 1	P_{02}=sin(0) = 0	P_{03}=cos(0) = 1
am	1	P_{10}=sin(1/1) = 0.84	P_{11}=cos(1/1) = 0.54	P_{12}=sin(1/10) = 0.10	P_{13}=cos(1/10) = 1.0
a	2	P_{20}=sin(2/1) = 0.91	P_{21}=cos(2/1) = -0.42	P_{22}=sin(2/10) = 0.20	P_{23}=cos(2/10) = 0.98
Robot	3	P_{30}=sin(3/1) = 0.14	P_{31}=cos(3/1) = -0.99	P_{32}=sin(3/10) = 0.30	P_{33}=cos(3/10) = 0.96

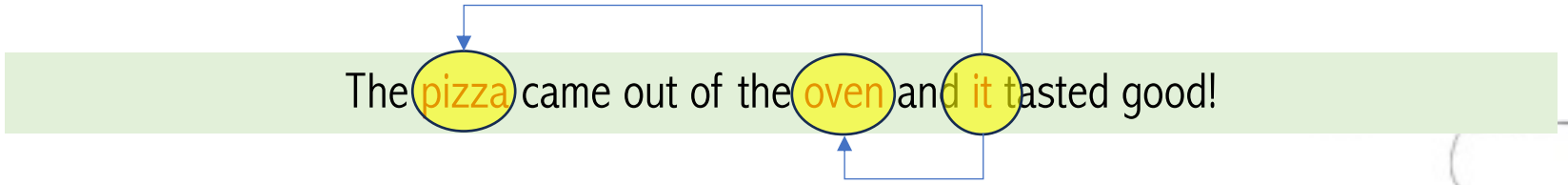
Positional Encoding Matrix for the sequence 'I am a robot'

https://machinelearningmastery.com/a-gentle-introduction-to-positional-encoding-in-transformer-models-part-1/

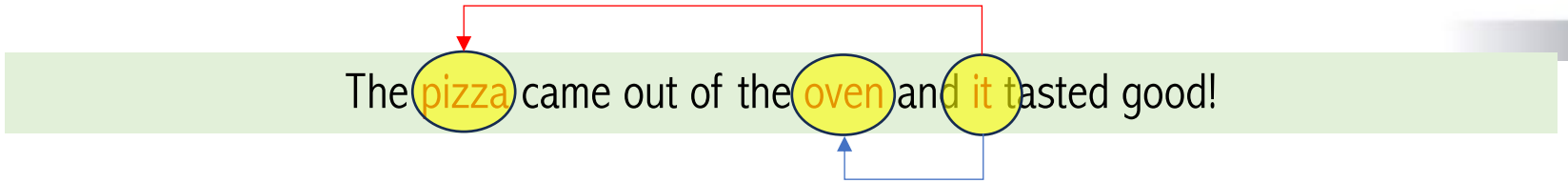
Word Ordering



Relationship Among Words: Idea



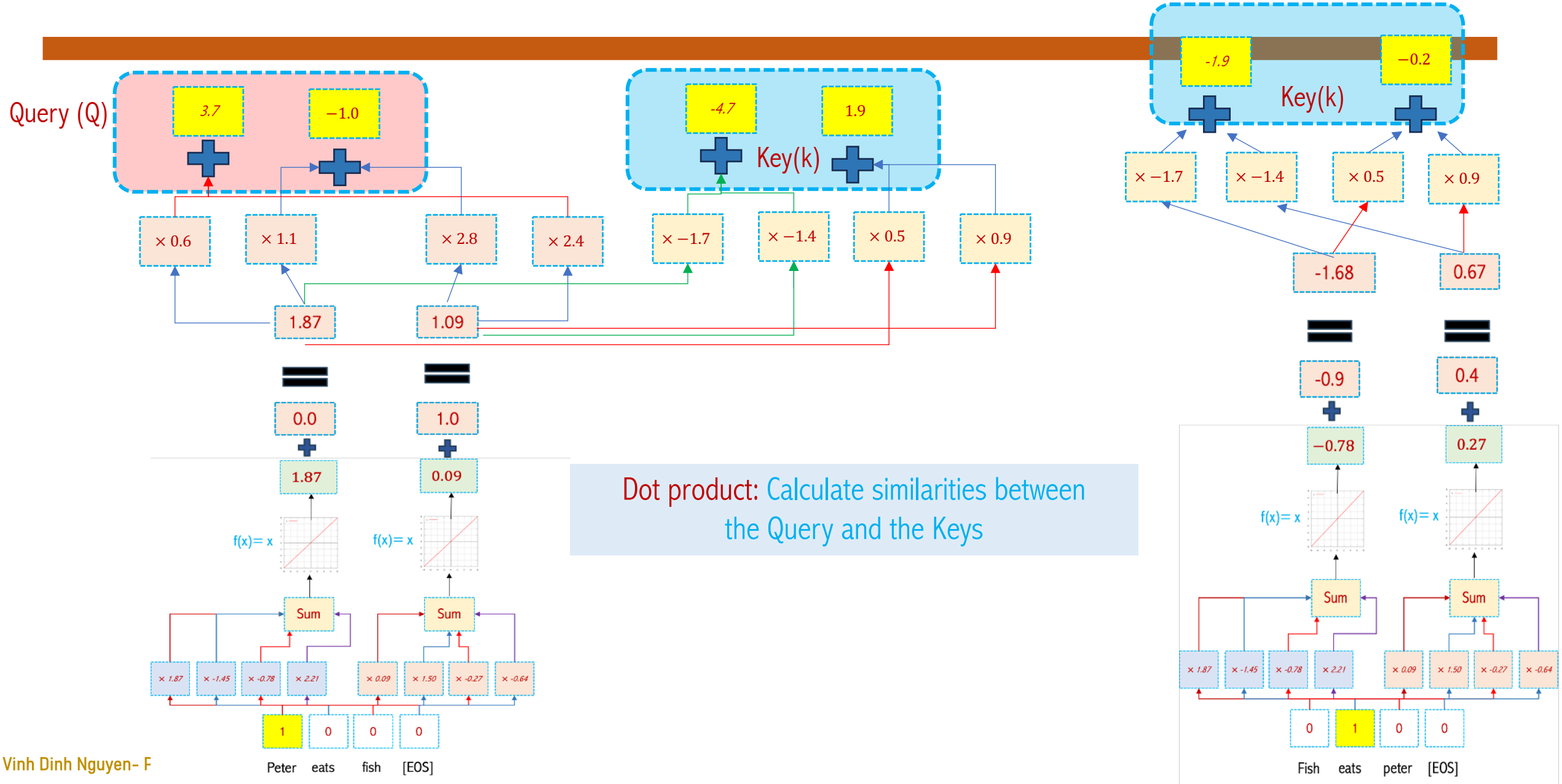
Seft-attention works by seeing how similar each word is to all of the words in the sentense, including itself



If we looked at a lot of sentenses abut pizza and the word it was more commonly associated with pizza than oven

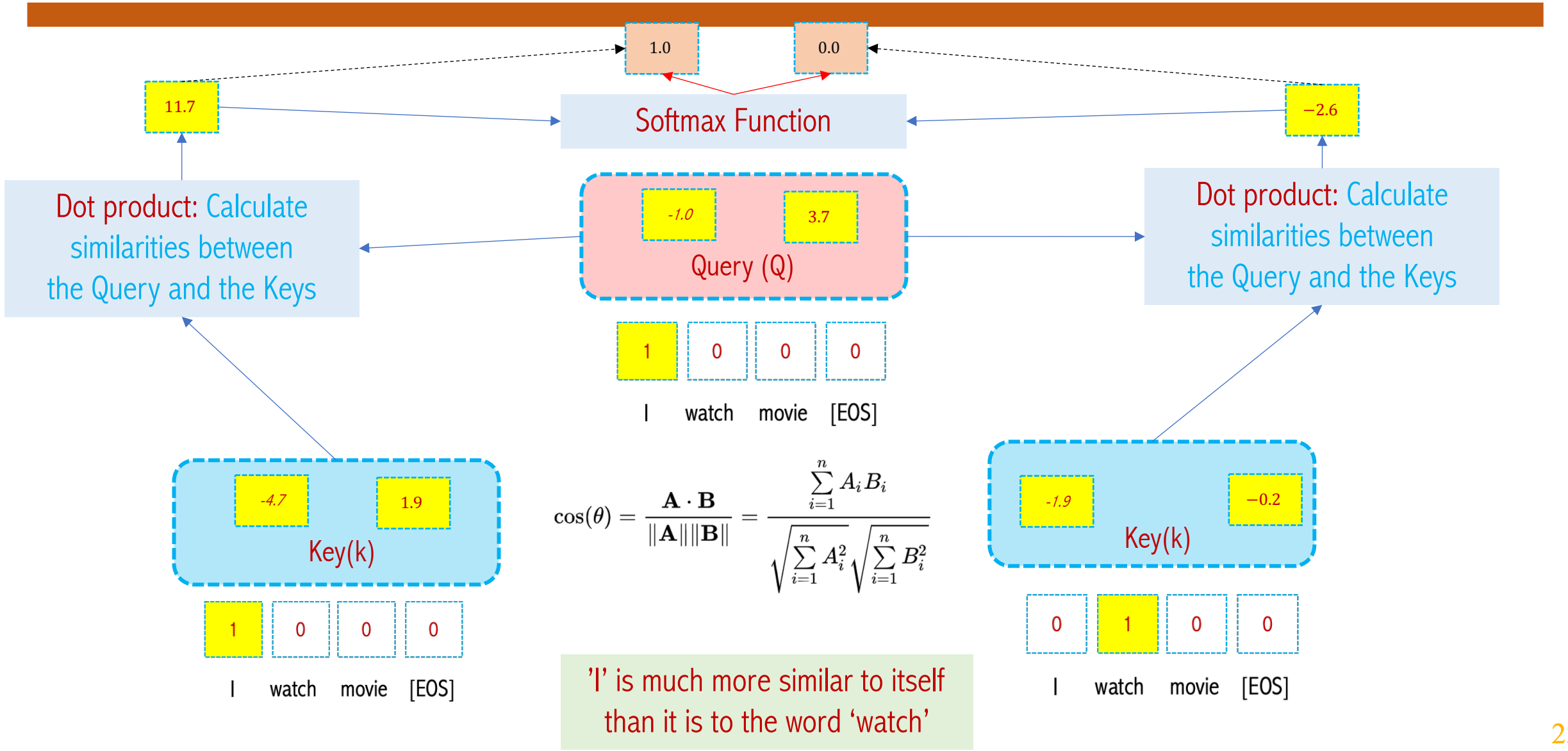
AI VIETNAM
All-in-One Course

Relationship Among Words: Detail



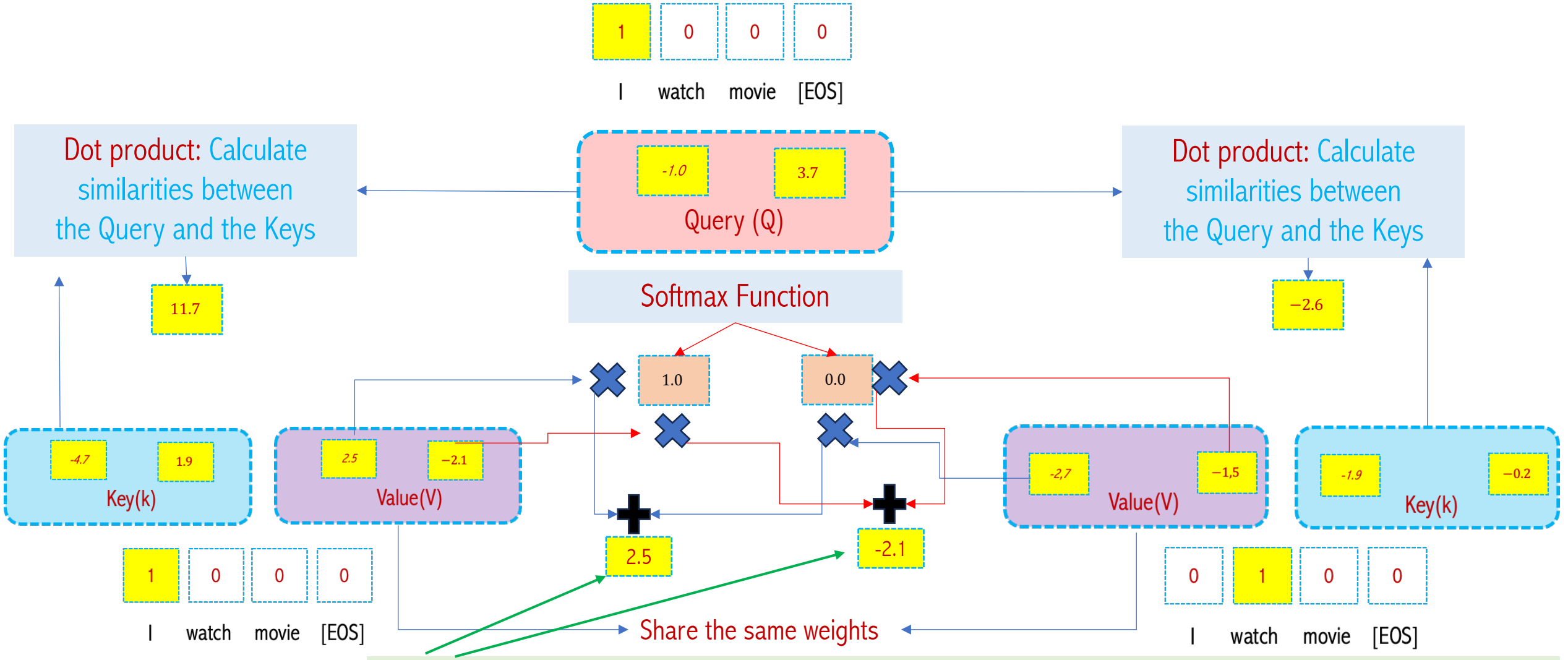
Dot product: Calculate similarities between the Query and the Keys

Relationship Among Words: Detail



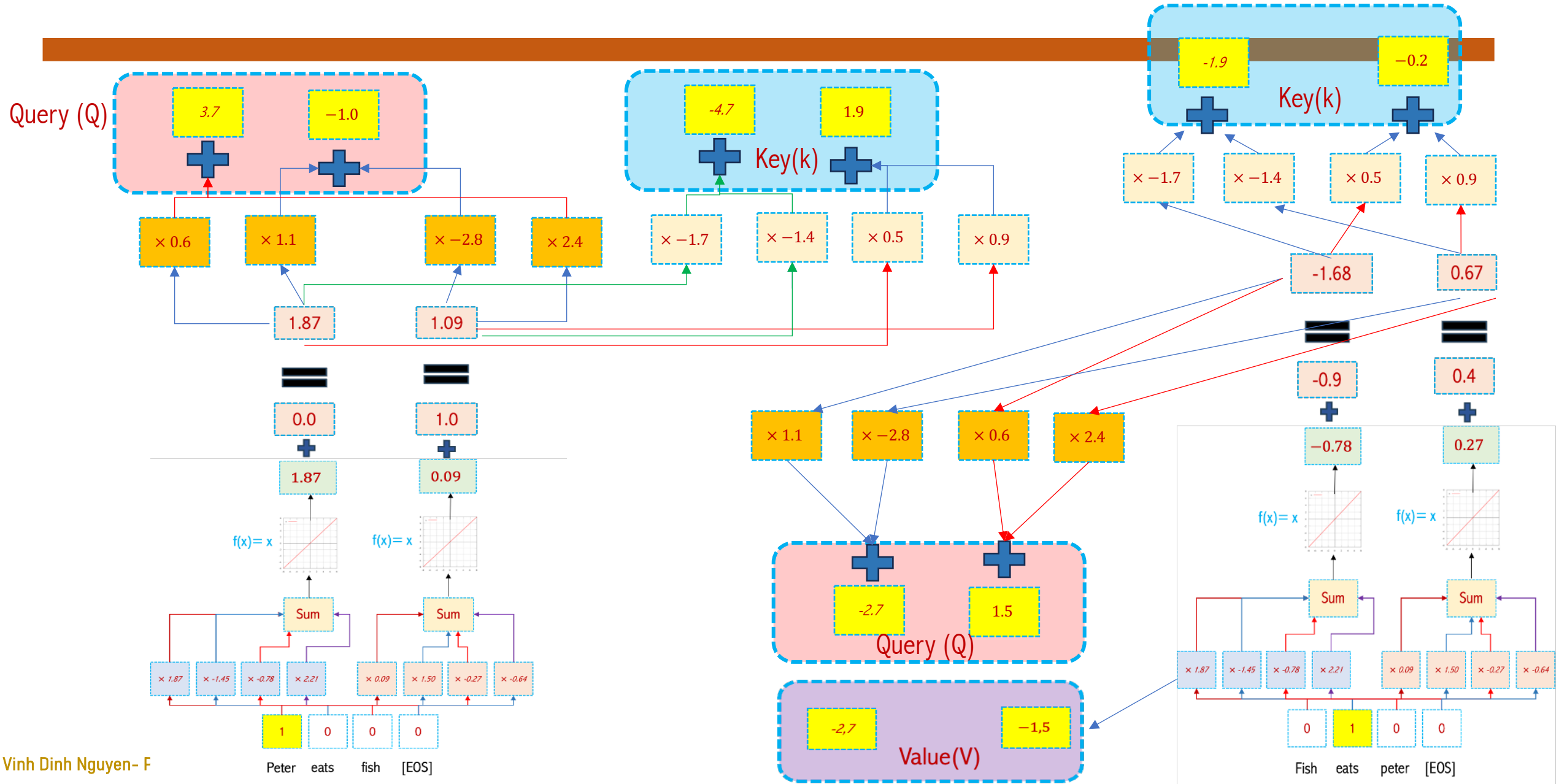
AI VIETNAM
All-in-One Course

Relationship Among Words: Detail

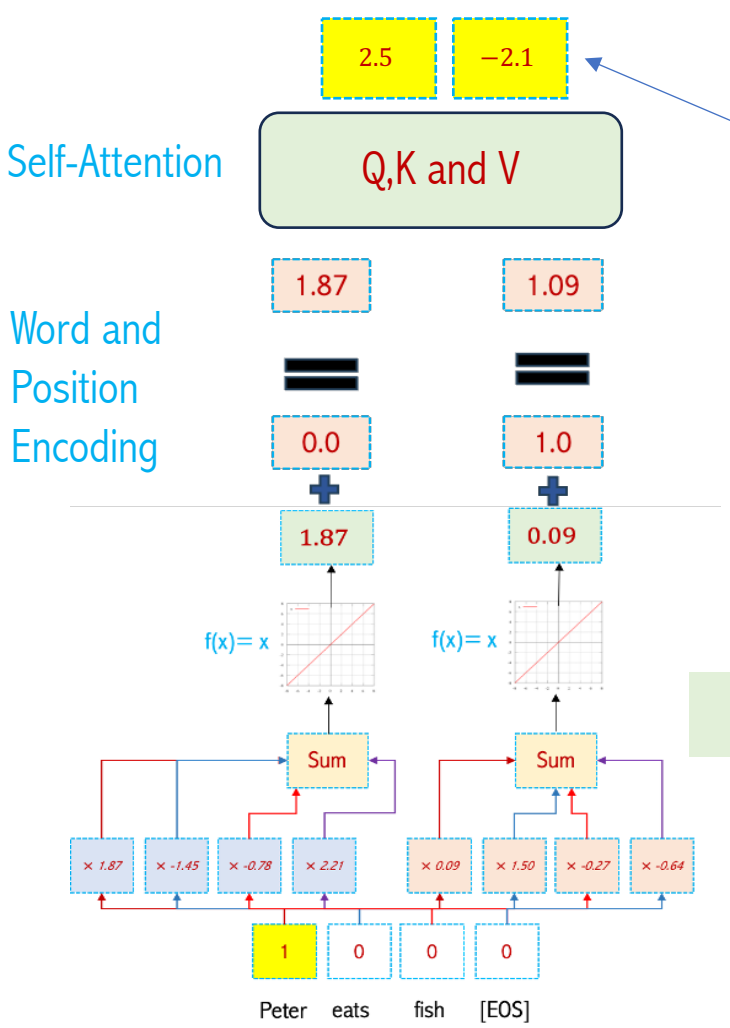


Summation = combine encoding 'I' and 'watch' & Compute similarity to 'I' = selft-attention value for 'I'

Relationship Among Words: Detail

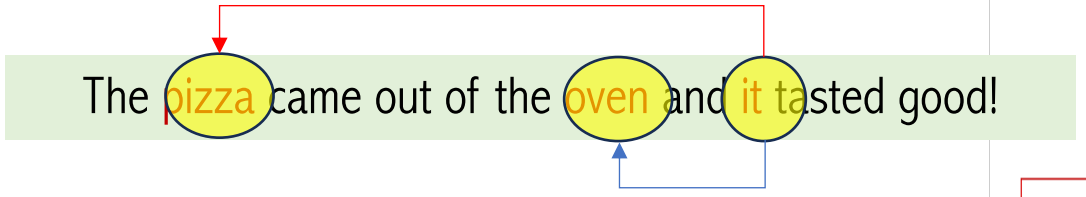


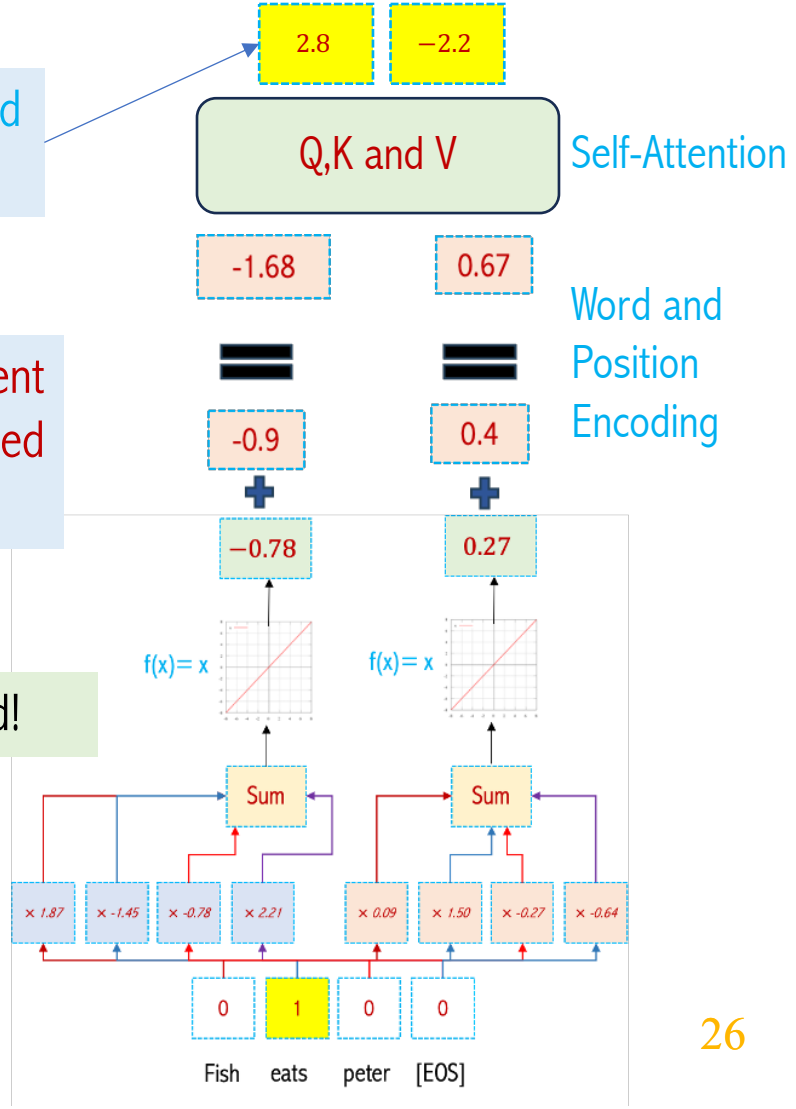
Relationship Among Words: Detail



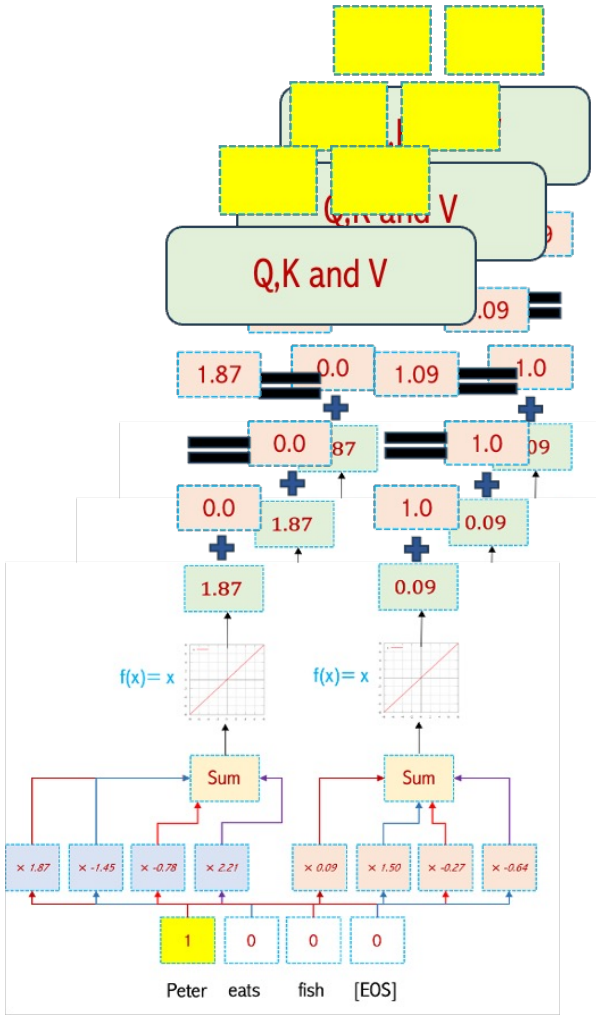
Pay attention: Self-attention values for each word contain input from all of the other words

Stack of Self-Attentions to capture different relationship among the words im complicated sentenses and pagraphs.





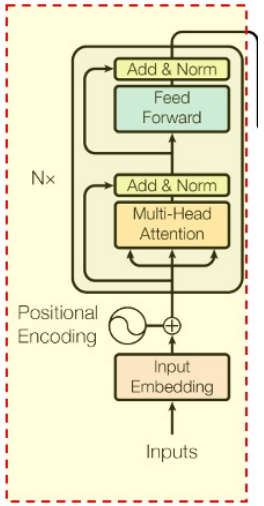
Relationship Among Words: Detail



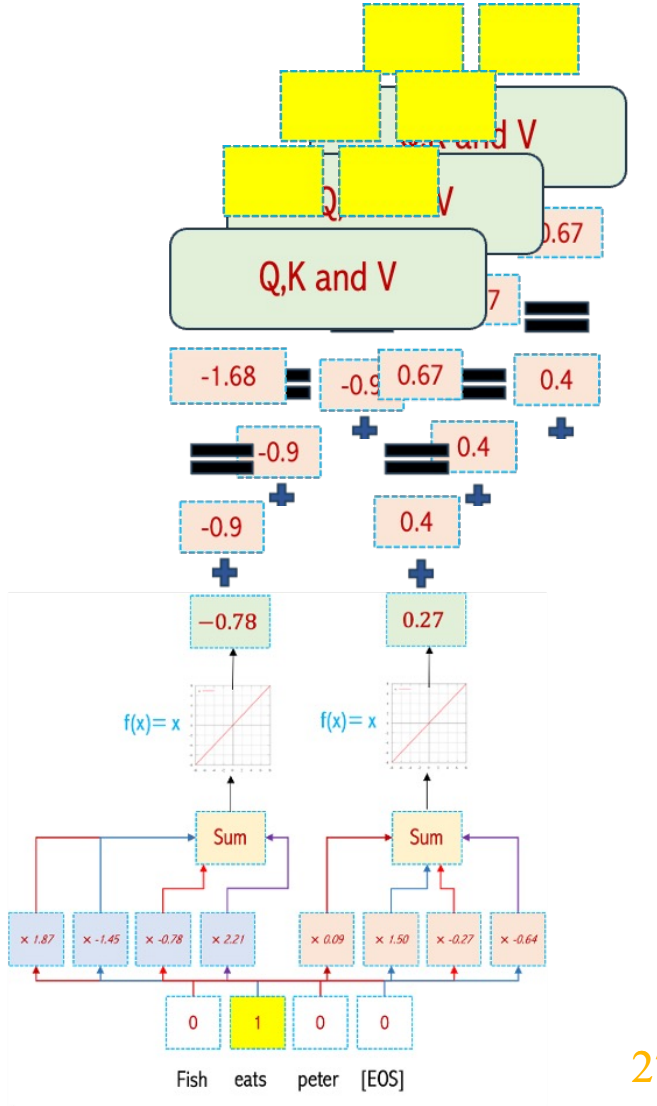
Stack of Self-Attentions to capture different relationship among the words im complicated sentenses and pagraphs.

Tranformer: 8 self-attention was used

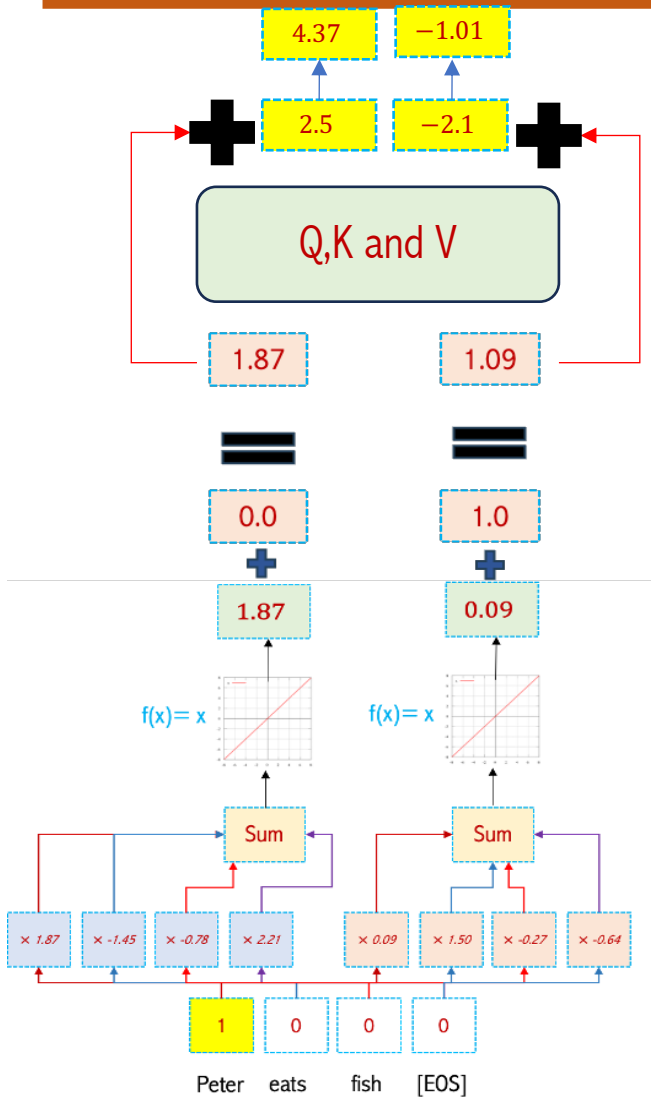
GPT-v1: 12 sefl-attention was used



Encoder



Residual Connections

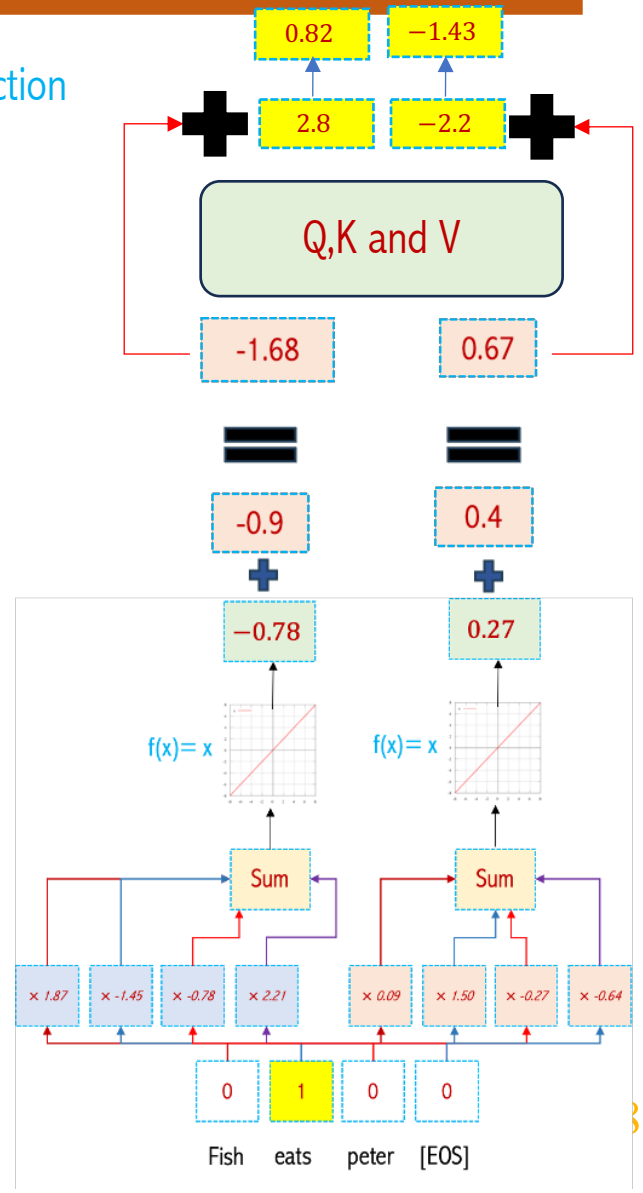


Residual Connection ↔ Easy to train ↔ Residual Connection

Self-Attention ↔ Encode the relationship of words ↔ Self-Attention

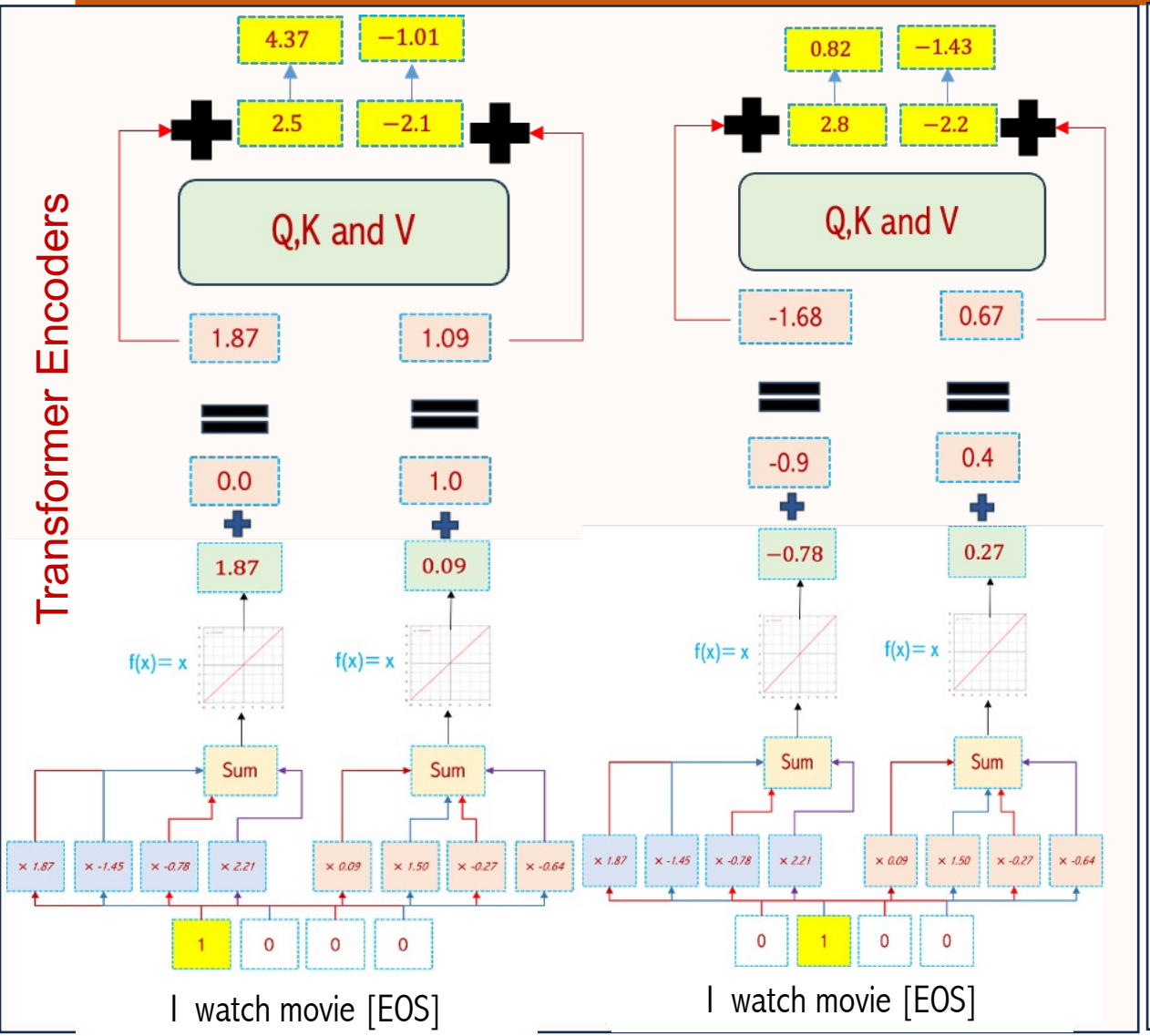
Position Encoding ↔ Encode the position of the words ↔ Position Encoding

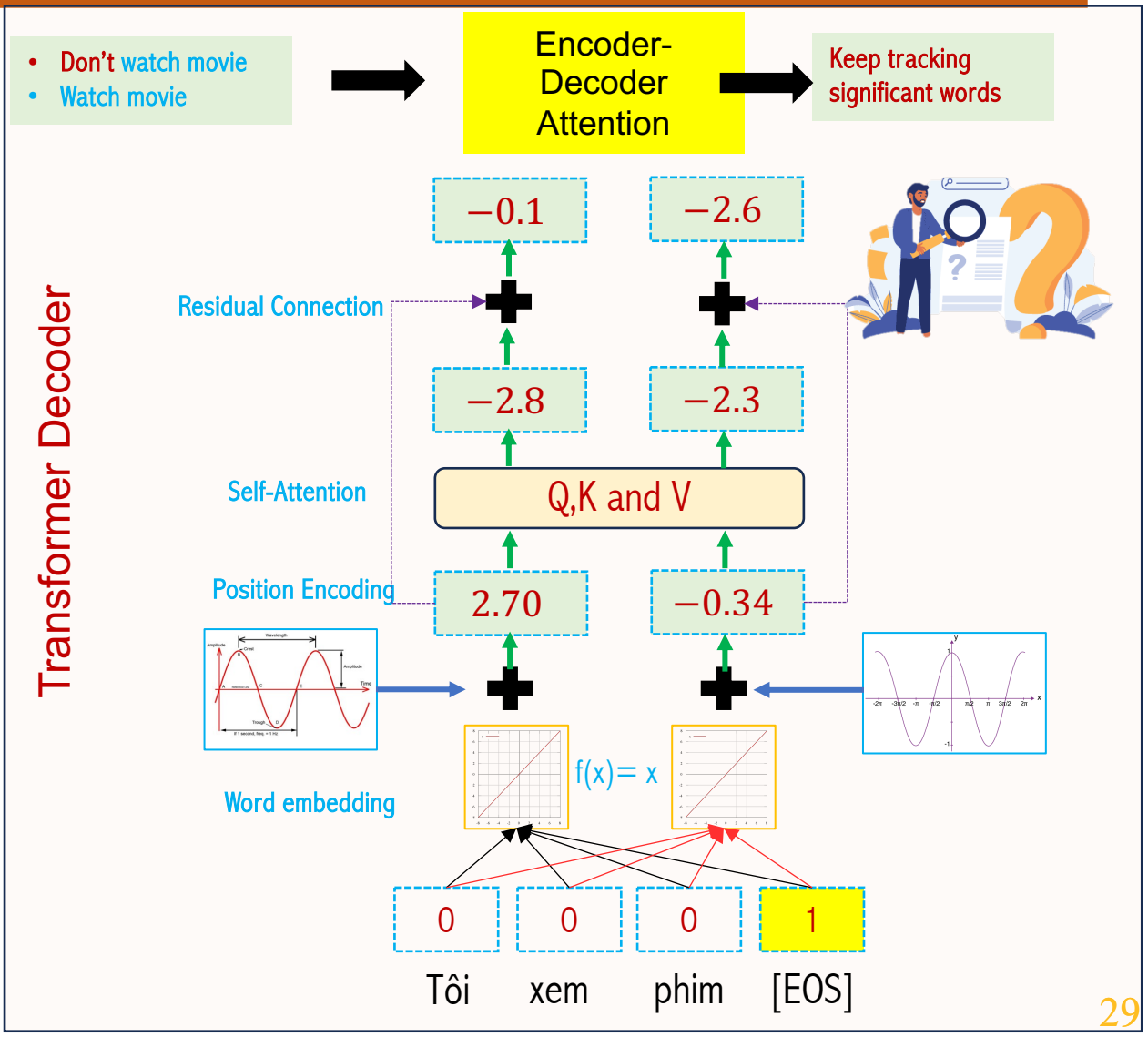
Word embedding ↔ Encode words into numbers ↔ Word embedding



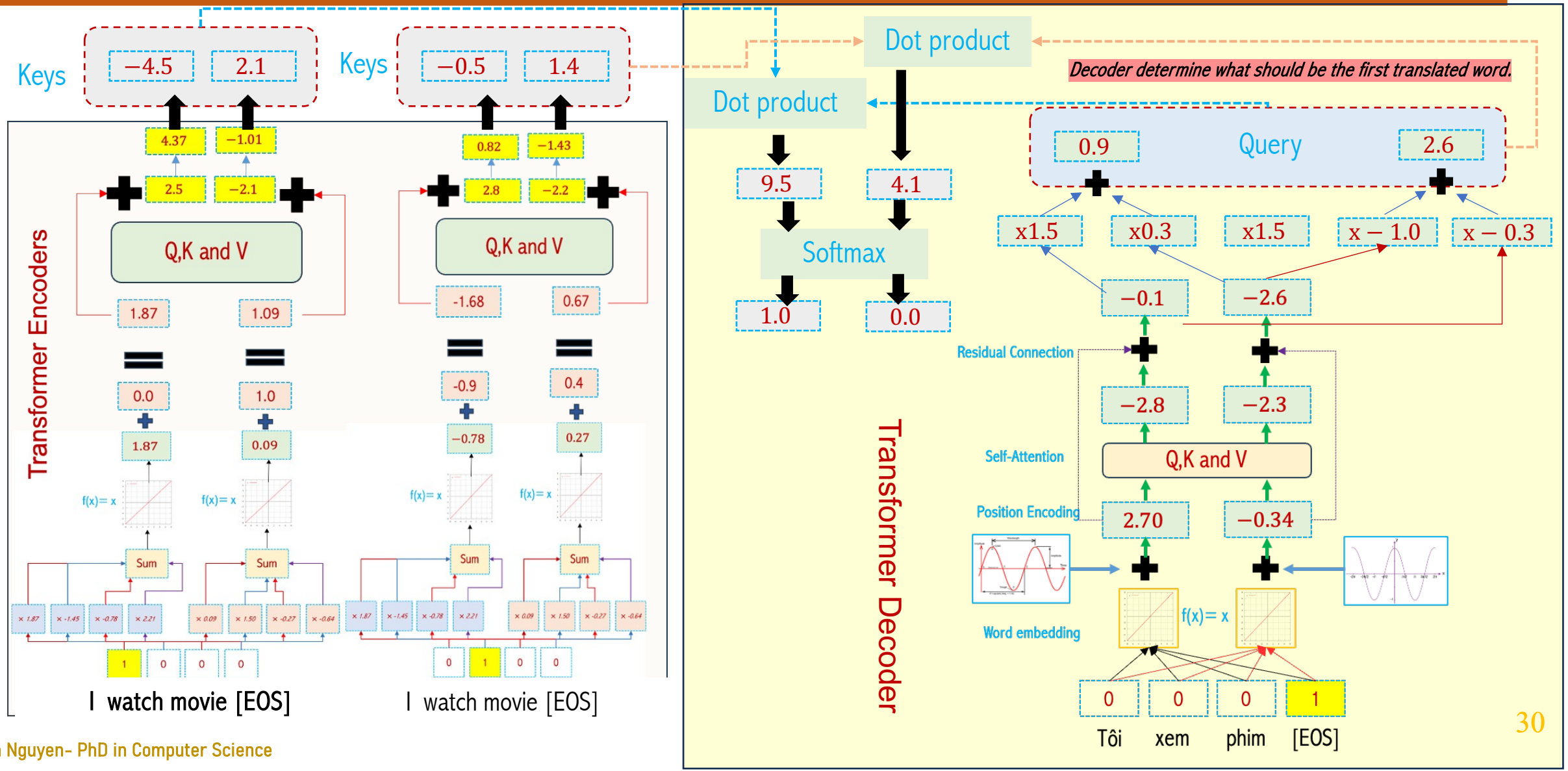
Transformer Encoders:
Encode the input

Transformer Encoder-Decoder

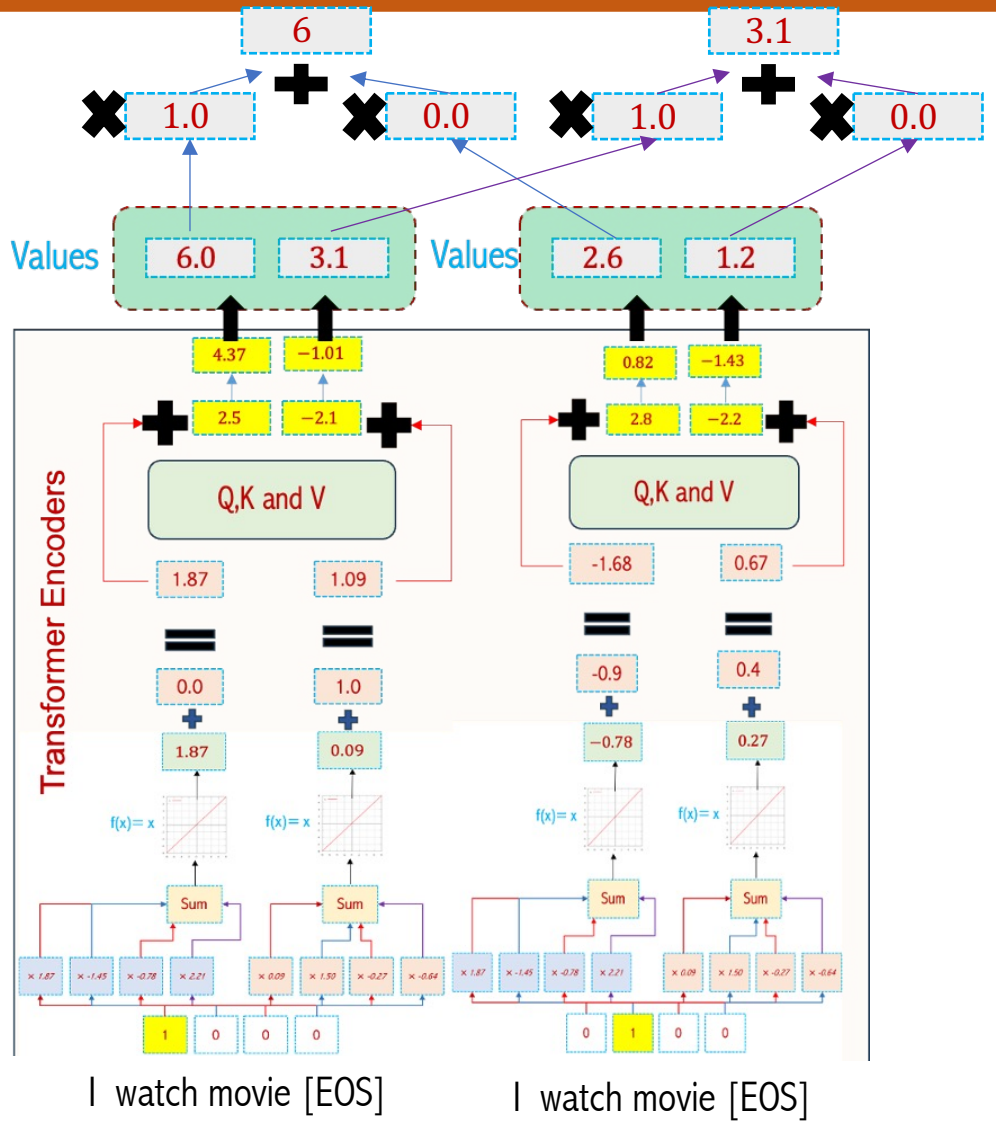


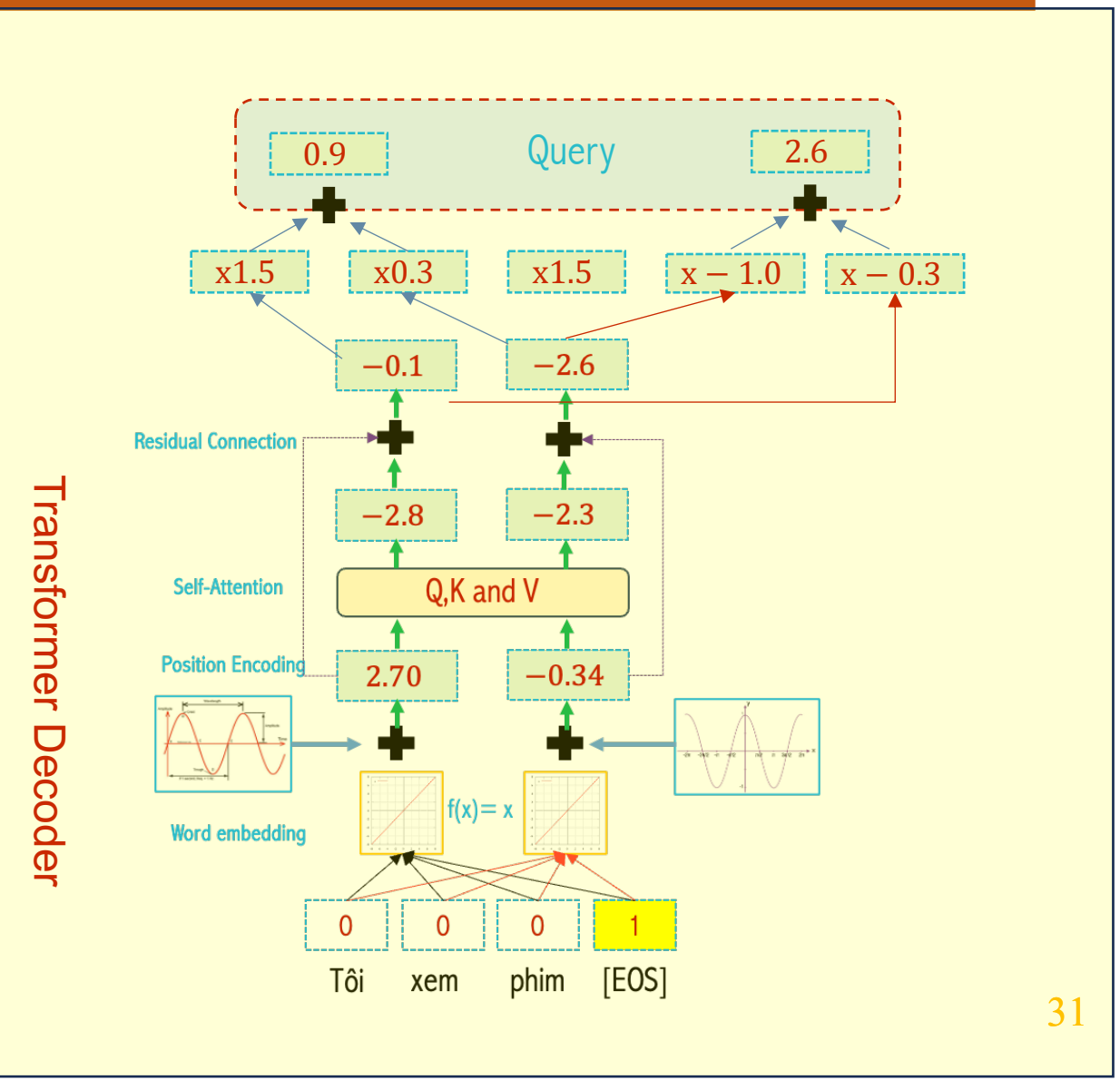


Encoder-Decoder Attention

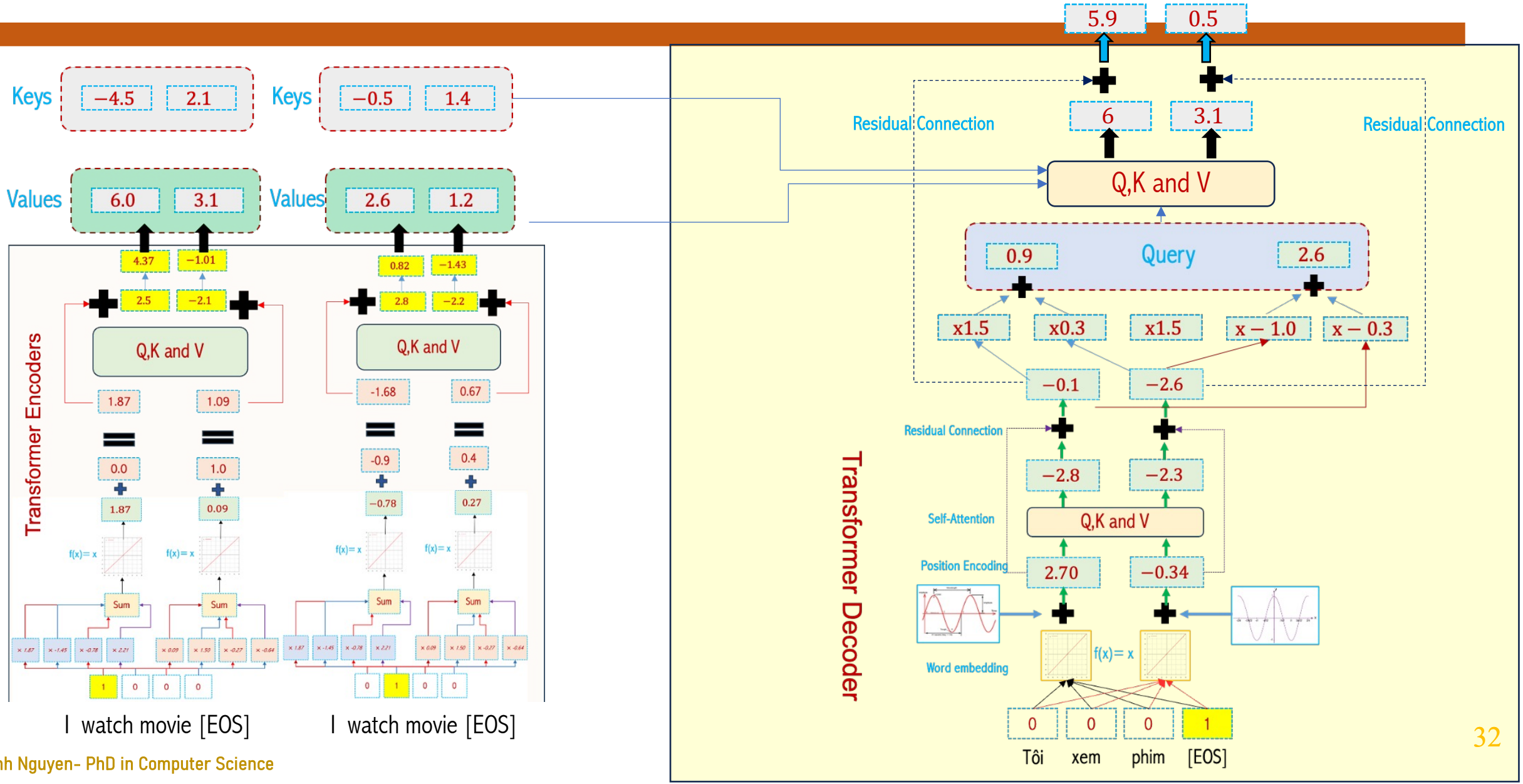


Encoder-Decoder Attention

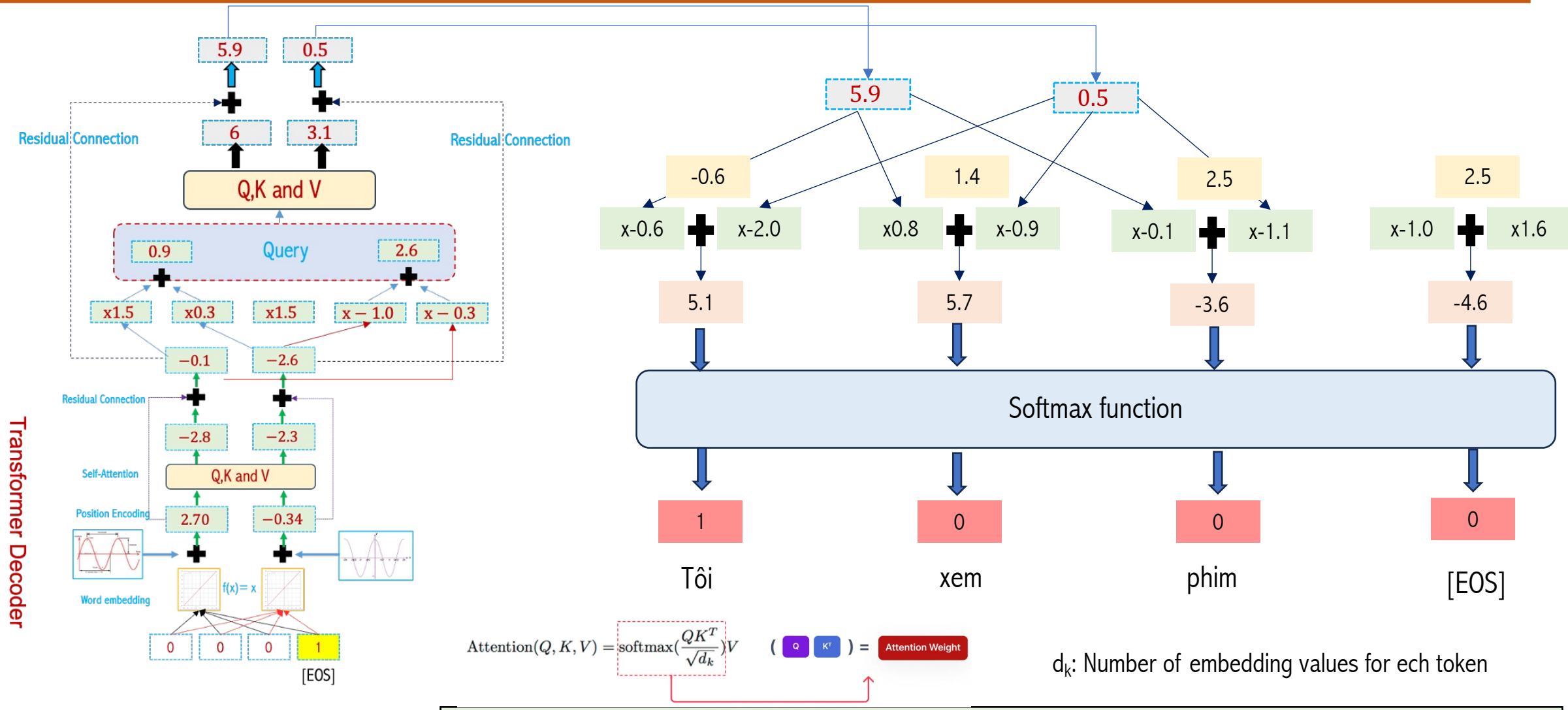




Encoder-Decoder Attention



Decoder Output



Translation is correct, but the Decoder does not stop until it outputs an [EOS] token.

What is Self-Attention?

Self-Attention allows the model to relate words to each other.

In this simple case we consider the sequence length **seq** = 6 and $\mathbf{d_{model}} = \mathbf{d_k} = 512$.

The matrices **Q**, **K** and **V** are just the input sentence.

$$Attention(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

$$\text{softmax}\left(\frac{\mathbf{Q}_{(6,512)} \times \mathbf{K}^T_{(512,6)}}{\sqrt{512}}\right) =$$

	YOUR	CAT	IS	A	LOVELY	CAT	Σ
YOUR	0.268	0.119	0.134	0.148	0.179	0.152	1
CAT	0.124	0.278	0.201	0.128	0.154	0.115	1
IS	0.147	0.132	0.262	0.097	0.218	0.145	1
A	0.210	0.128	0.206	0.212	0.119	0.125	1
LOVELY	0.146	0.158	0.152	0.143	0.227	0.174	1
CAT	0.195	0.114	0.203	0.103	0.157	0.229	1

(6, 6)

* all values are random.

* for simplicity I considered only one head, which makes $d_{model} = d_k$.

What is Self-Attention?

$$Attention(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

	YOUR	CAT	IS	A	LOVELY	CAT
YOUR	0.268	0.119	0.134	0.148	0.179	0.152
CAT	0.124	0.278	0.201	0.128	0.154	0.115
IS	0.147	0.132	0.262	0.097	0.218	0.145
A	0.210	0.128	0.206	0.212	0.119	0.125
LOVELY	0.146	0.158	0.152	0.143	0.227	0.174
CAT	0.195	0.114	0.203	0.103	0.157	0.229

(6, 6)

X

V

(6, 512)

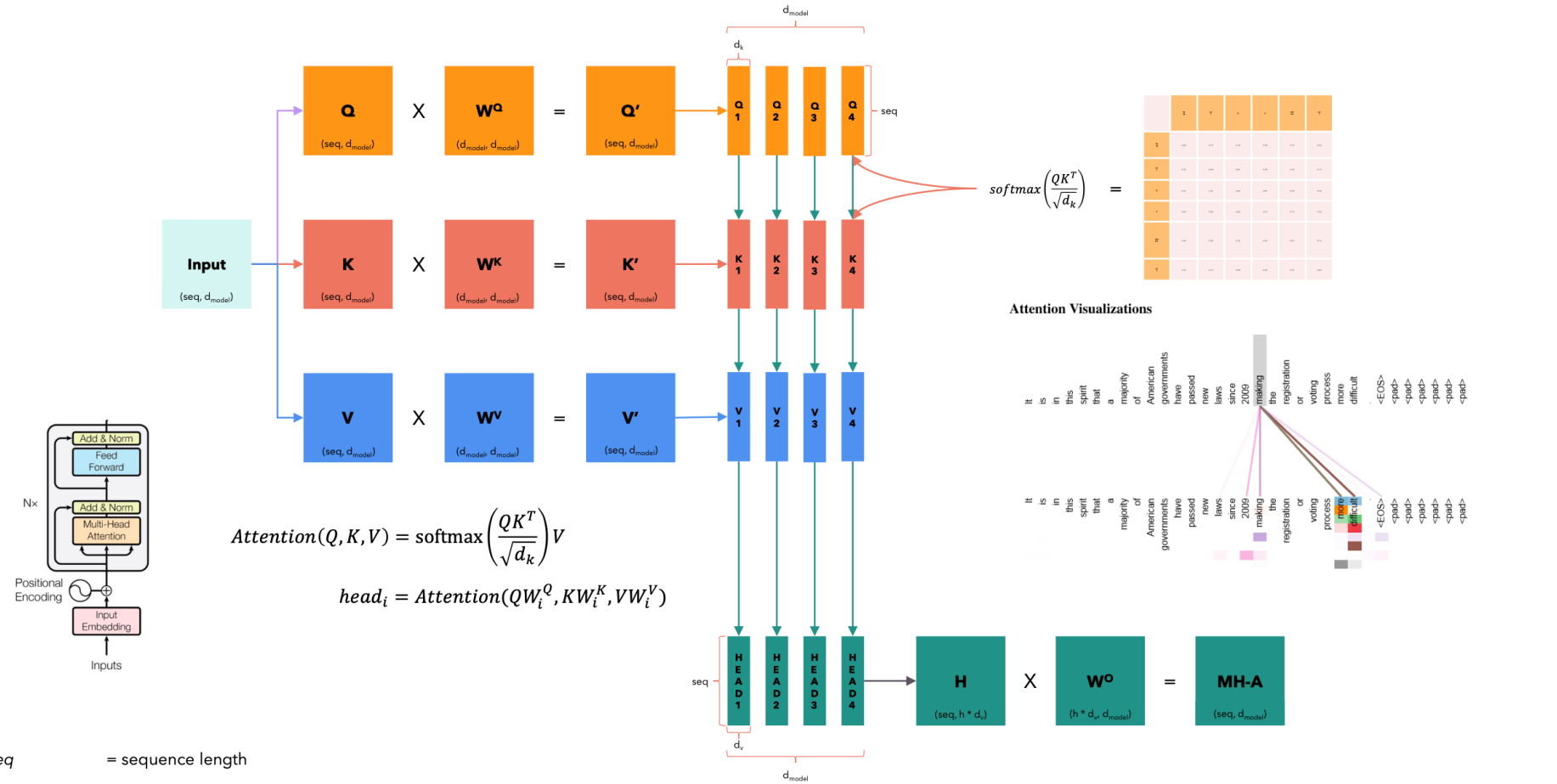
=

Attention

(6, 512)

Each row in this matrix captures not only the meaning (given by the embedding) or the position in the sentence (represented by the positional encodings) but also each word's interaction with other words.

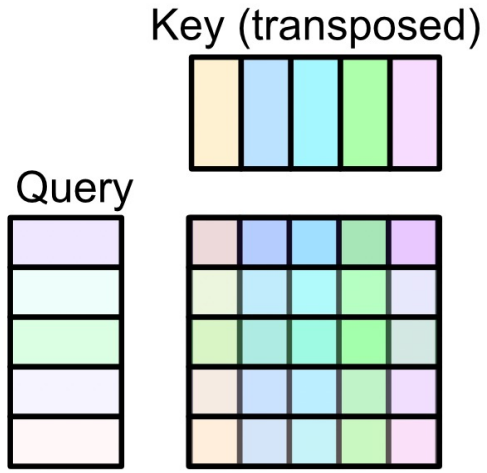
What is Self-Attention?

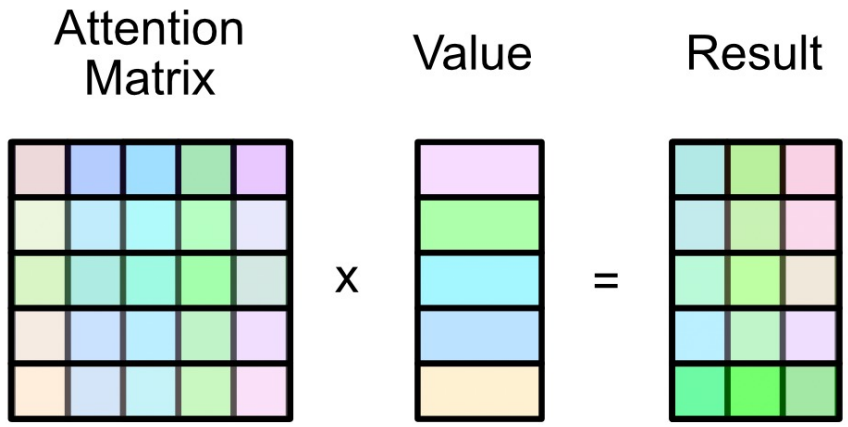


seq	= sequence length
d_{model}	= size of the embedding vector
h	= number of heads
$d_k = d_v$	= d_{model} / h

Umar Jamil - https://github.com/hkproj/transformer-from-scratch-notes

Attention: Summary





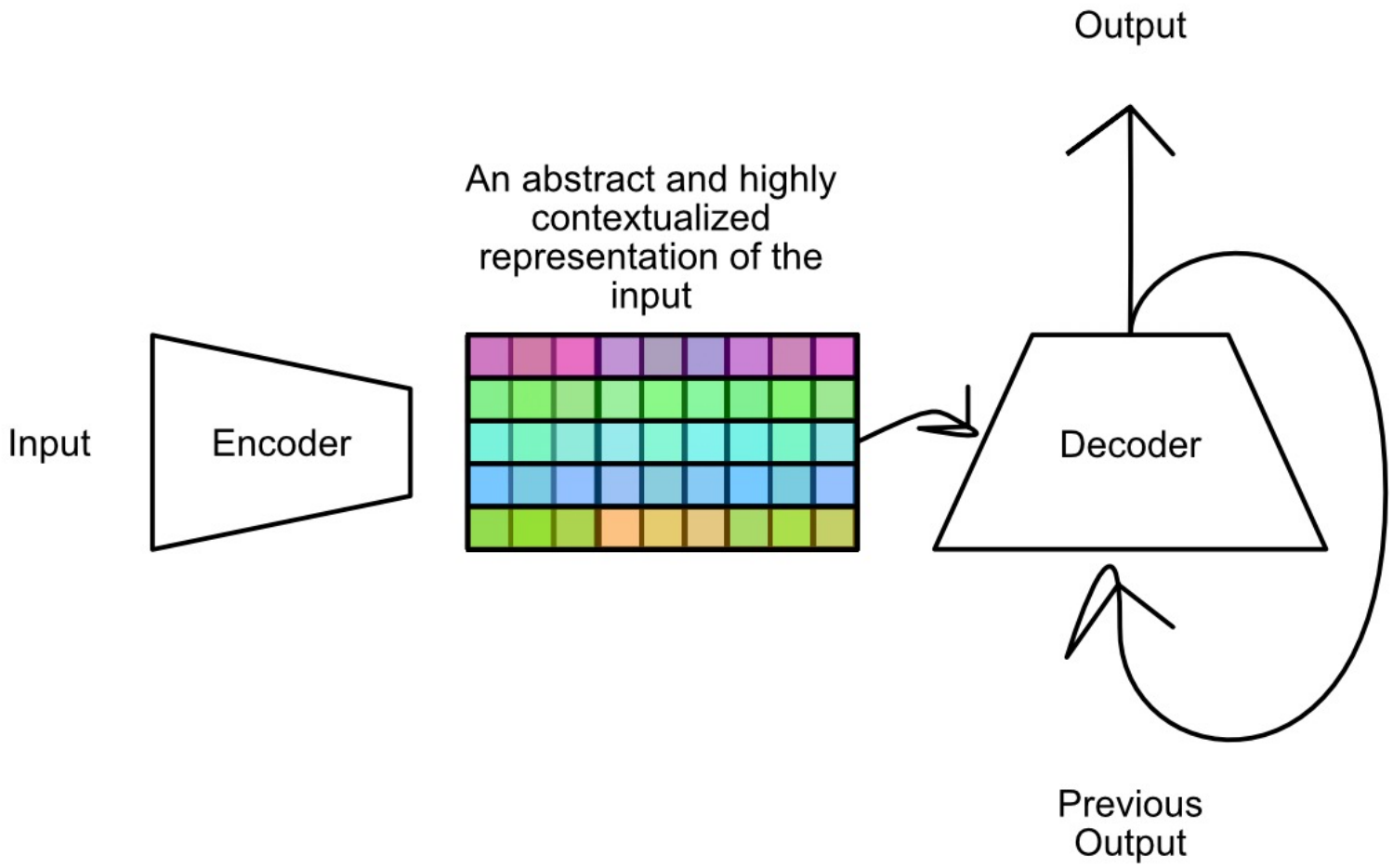
Two inputs, called the "key" and "query" get multiplied together to create the attention matrix. The key is usually transposed (rotated) to make the matrix multiplication work out right

Then, the attention matrix is used as a filter to transform the value matrix into the final output.
The attention matrix acts like a filter, which transforms an input called the "value" into the final result.

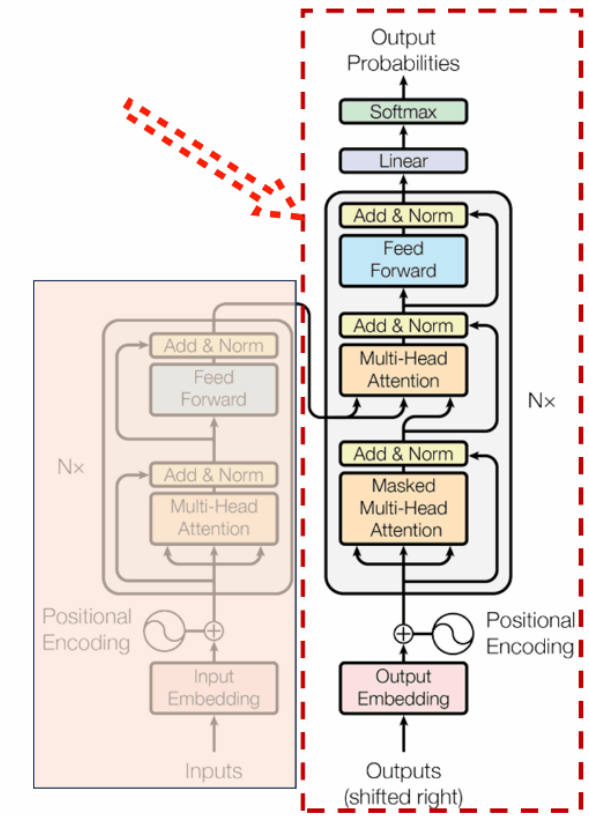
the attention mechanism uses some inputs to filter other inputs

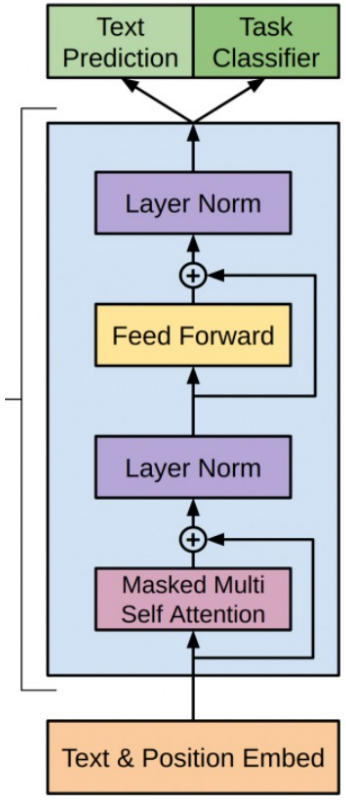
Transformer: Summary

The encoder converts an input into an abstract representation which the decoder uses to iteratively generate output.



Decoder Only Transformers (GPT)



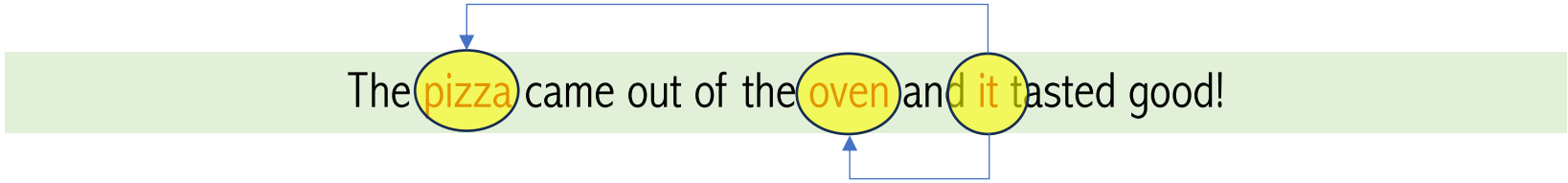


Encoder-only style models are good at extracting information from text for tasks like classification and regression, while decoder-only style models focus on generating text. GPT, being a model focused on text generation, is a decoder only style model.

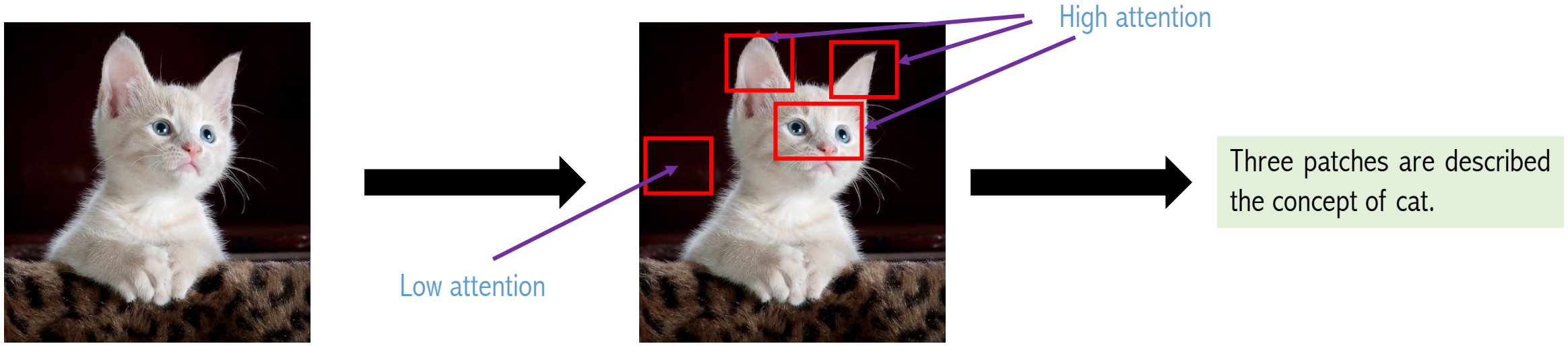
Outline

- Objective
- Diffusion Probability Model
- Transformer and ChatGPT Architecture
- Vision Transformer Architecture
- Diffusion and Transformer
- OpenAI's Sora Architecture
- Summary

Attention in Text and Image

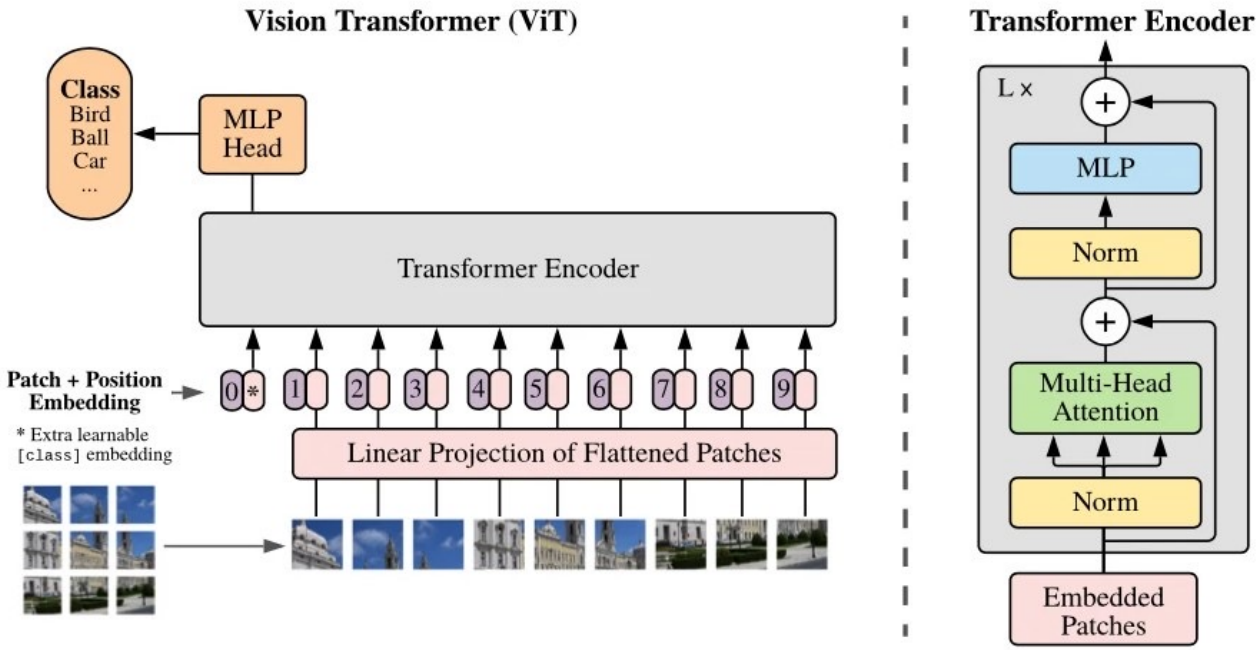


Seft-attention works by seeing how similar each word is to all of the words in the sentense, including itself



The self-attention mechanism can be applied to the feature maps of a convolutional neural network (CNN) in order to allow the network to selectively focus on important image regions while suppressing noise and irrelevant information.

Vision Transformer

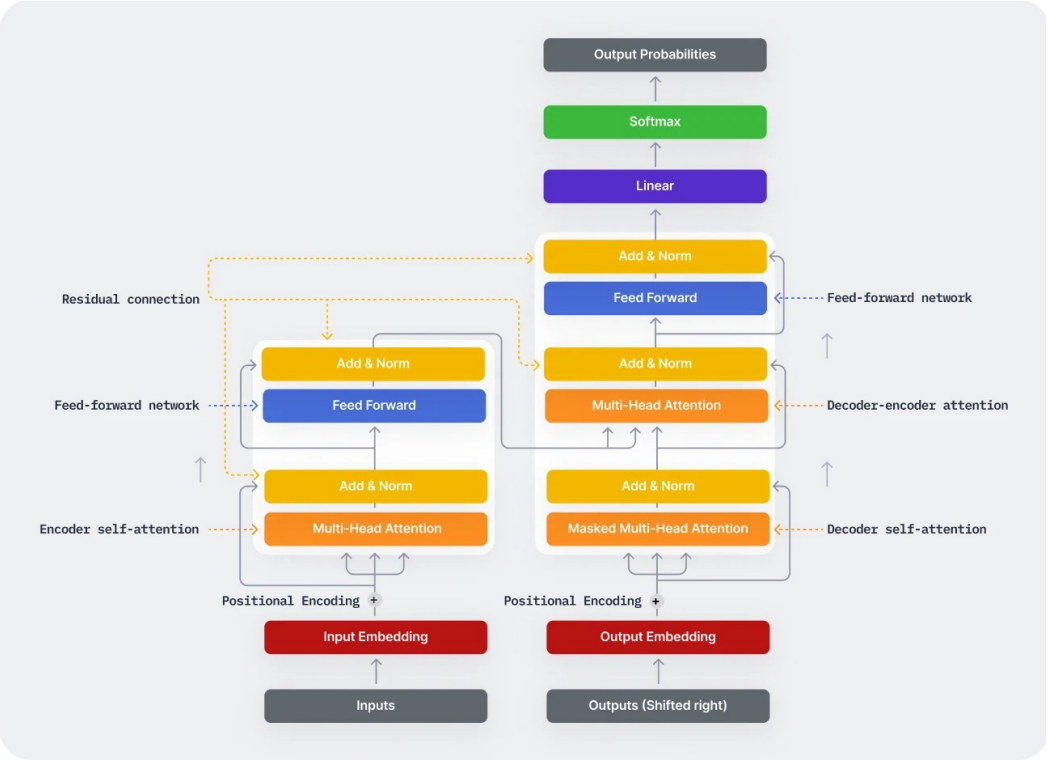


1.Split an image into patches
2.Flatten the patches
3.Produce lower-dimensional linear embeddings from the flattened patches
4.Add positional embeddings
5.Feed the sequence as an input to a standard transformer encoder
6.Pretrain the model with image labels (fully supervised on a huge dataset)
7.Finetune on the downstream dataset for image classification

An Image is Worth 16*16 Words: Transformers for Image Recognition at Scale," published at ICLR 2021.

The ViT model represents an input image as a series of image patches, like the series of word embeddings used when using transformers to text, and directly predicts class labels for the image.

Vision Transformer for Classification



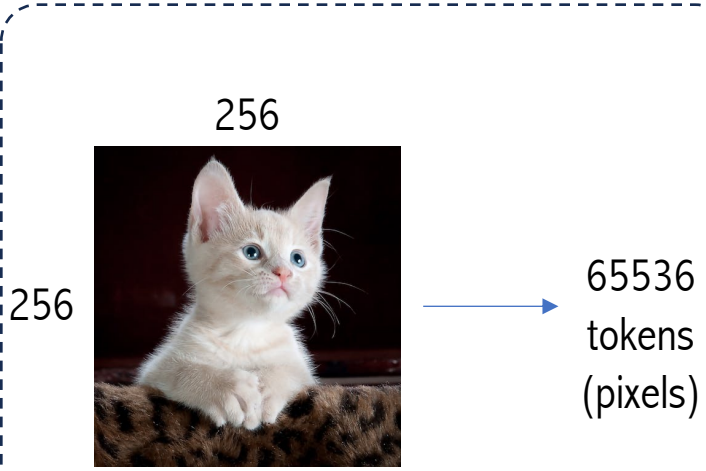
1.Split an image into patches
2.Flatten the patches
3.Produce lower-dimensional linear embeddings from the flattened patches
4.Add positional embeddings
5.Feed the sequence as an input to a standard transformer encoder
6.Pretrain the model with image labels (fully supervised on a huge dataset)
7.Finetune on the downstream dataset for image classification

An Image is Worth 16*16 Words: Transformers for Image Recognition at Scale," published at ICLR 2021.

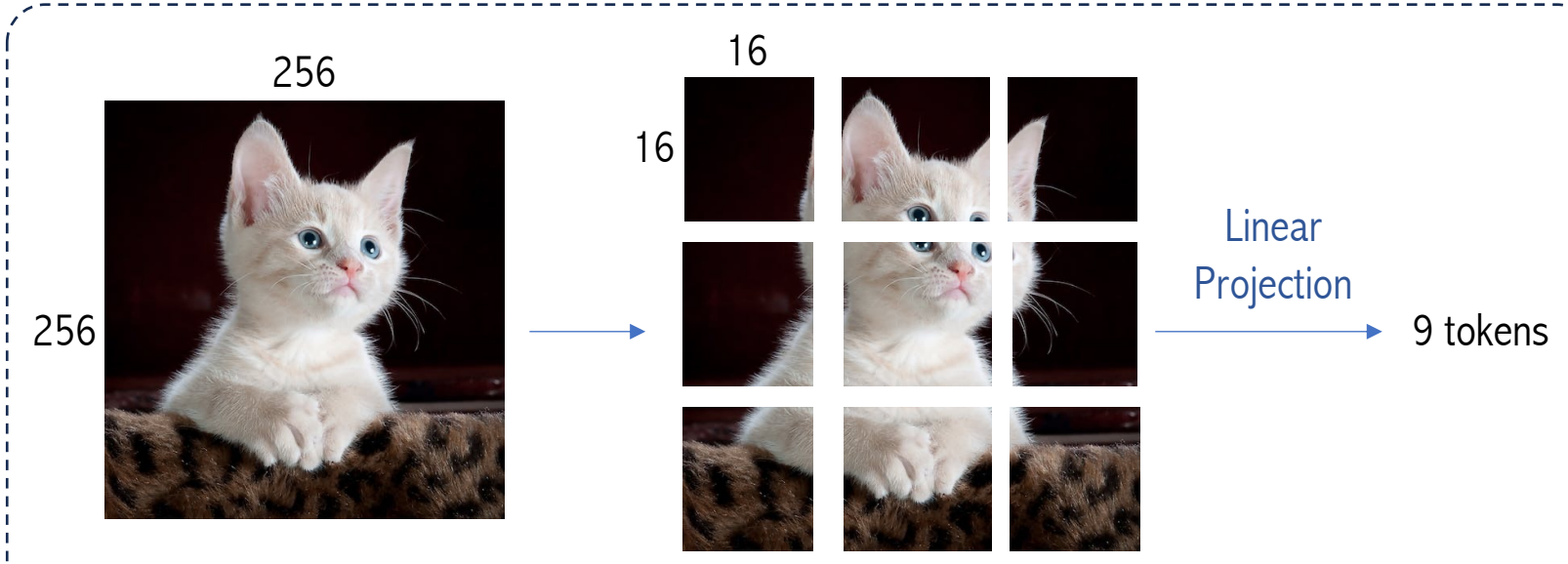
The ViT model represents an input image as a series of image patches, like the series of word embeddings used when using transformers to text, and directly predicts class labels for the image.

Vision Transformer for Classification

What are tokens in images?



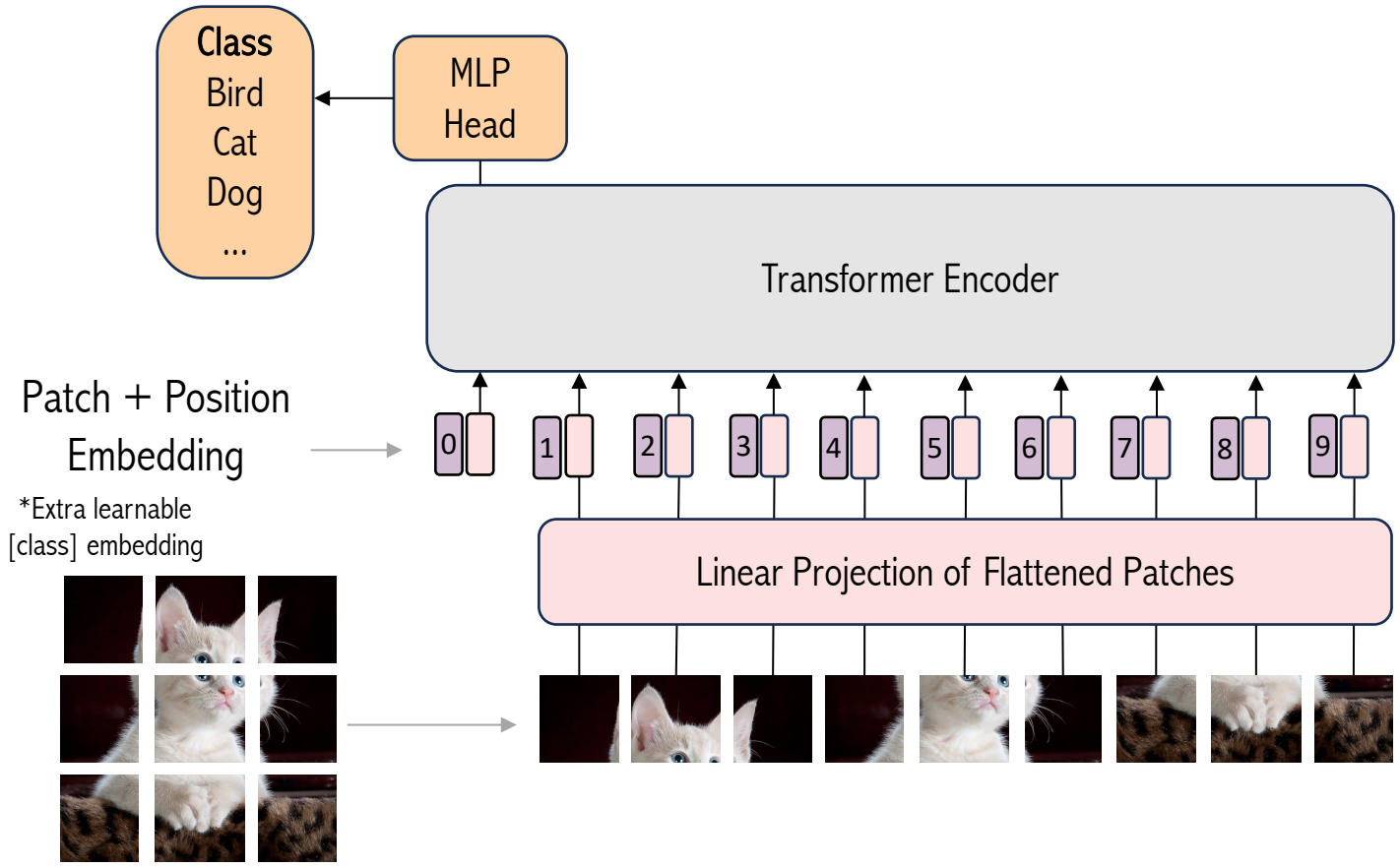
To much expensive!



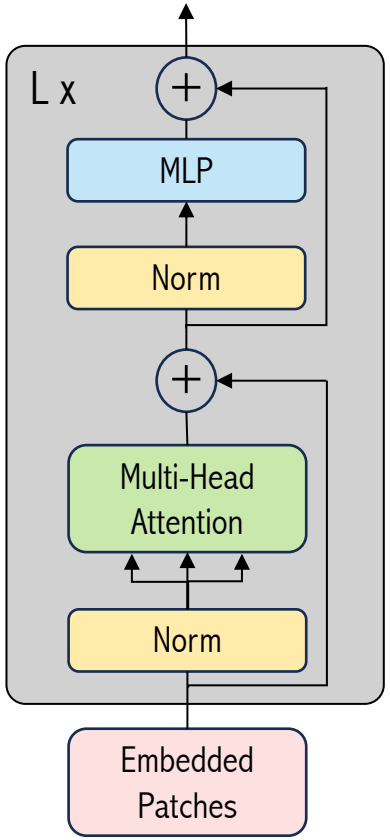
Much cheaper!

Vision Transformer for Classification

Vision Transformer (ViT)

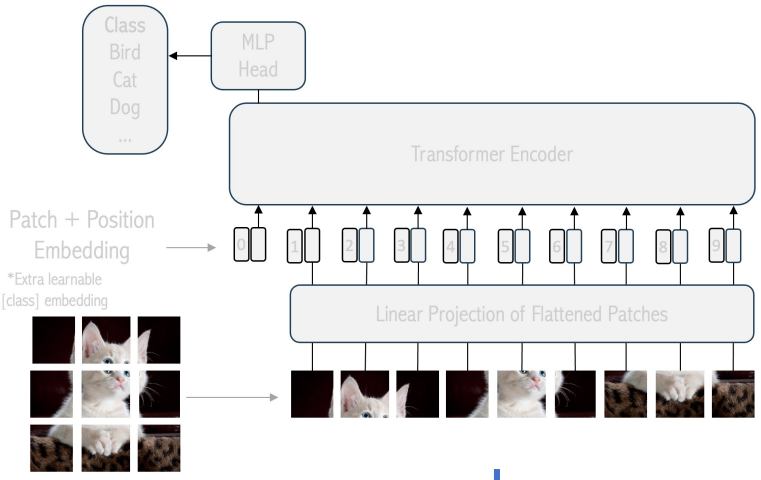


Transformer Encoder

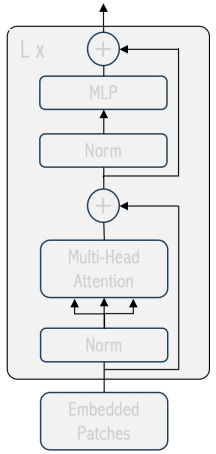


Vision Transformer

Vision Transformer (ViT)



Transformer Encoder



In the original paper they use total 196 tokens.

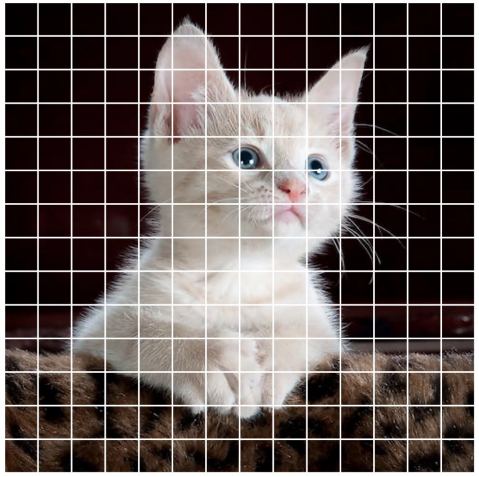
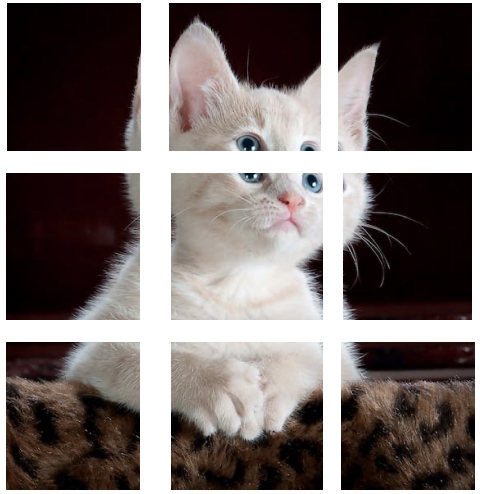


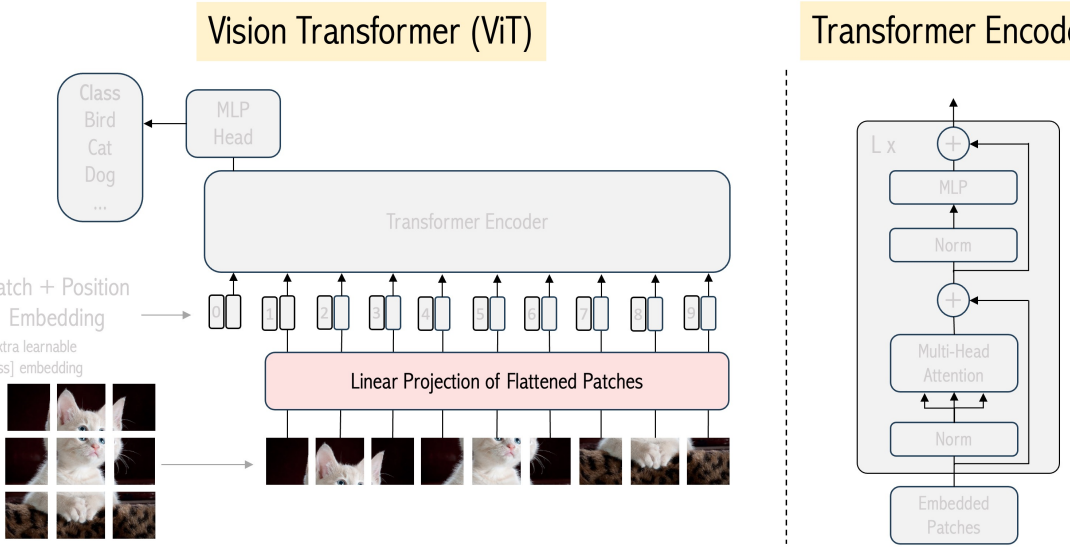
Image to Patches

$\boldsymbol{x} \in \mathbb{R}^{W \times H \times C}$

$\boldsymbol{x_p^i} \in \mathbb{R}^{P \times P \times C}, i = 1 \ldots 9$



Vision Transformer



Normal Linear Projection Patches

For each patch:

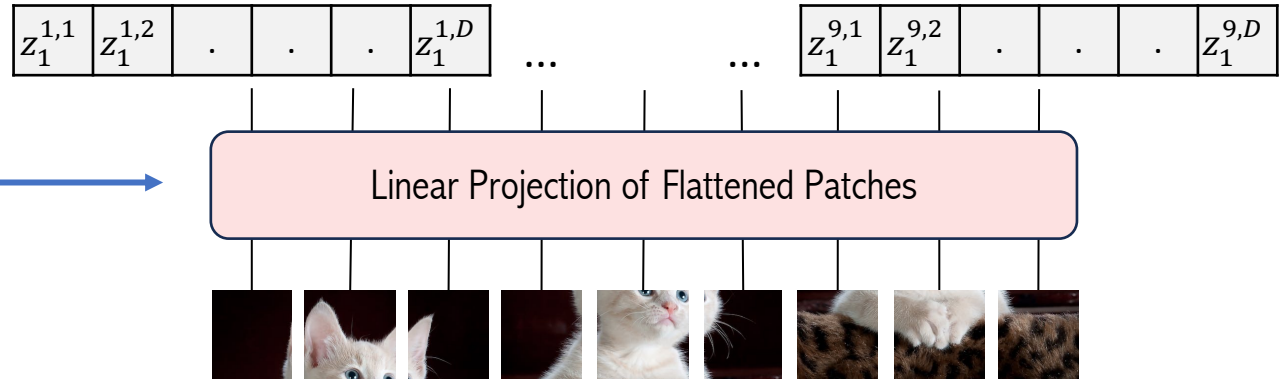
- $\boldsymbol{x_p^i} \in \mathbb{R}^{P\times P\times C} \rightarrow \boldsymbol{x_p^i} \in \mathbb{R}^{1\times P^2C} \rightarrow Reshape$
- $\boldsymbol{x_p^i} \in \mathbb{R}^{1\times P^2C} \cdot \boldsymbol{W} \in \mathbb{R}^{P^2C\times D} = \boldsymbol{x_p^i W} = \boldsymbol{z_1^i} \in \mathbb{R}^{1\times D}$

For all patches at a time, N is total tokens (N=P*P):

- $\boldsymbol{x} \in \mathbb{R}^{N\times P^2C} \cdot \boldsymbol{W} \in \mathbb{R}^{P^2C\times D} = \boldsymbol{xW} = \boldsymbol{z_1} \in \mathbb{R}^{N\times D}$

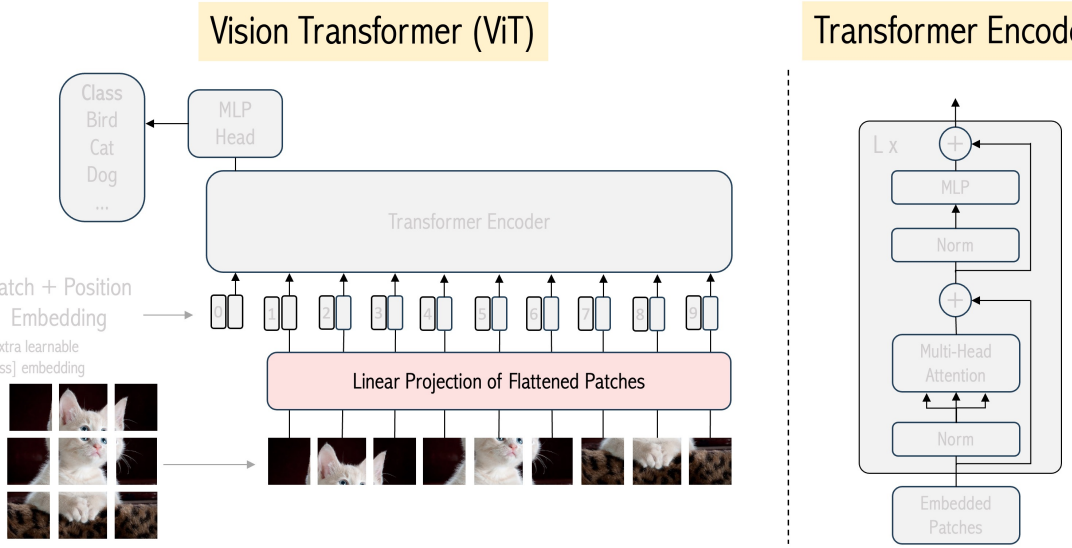
Linear Projection Patches
Convert patches to feature vectors
Flatten the patches from 2D to 1D vector

$$\boldsymbol{z_1^i} \in \mathbb{R}^{1\times D}, i = 1 \dots 9$$



$$\boldsymbol{x_p^i} \in \mathbb{R}^{P\times P\times C}, i = 1 \dots 9$$

Vision Transformer



Normal Linear Projection Patches

For each patch:

- $\boldsymbol{x_p^i} \in \mathbb{R}^{P \times P \times C} \rightarrow \boldsymbol{x_p^i} \in \mathbb{R}^{1 \times P^2 C} \rightarrow Reshape$
- $\boldsymbol{x_p^i} \in \mathbb{R}^{1 \times P^2 C} \cdot \boldsymbol{W} \in \mathbb{R}^{P^2 C \times D} = \boldsymbol{x_p^i W} = \boldsymbol{z_1^i} \in \mathbb{R}^{1 \times D}$

Patch Linear Projection

For all patches at a time:

- $\boldsymbol{x} \in \mathbb{R}^{N \times P^2 C} \cdot \boldsymbol{W} \in \mathbb{R}^{P^2 C \times D} = \boldsymbol{xW} = \boldsymbol{z_1} \in \mathbb{R}^{N \times D}$

Image Linear Projection

$$\boldsymbol{x} \in \mathbb{R}^{N \times P^2 C} \rightarrow \boldsymbol{z_1} \in \mathbb{R}^{N \times D}$$

$\boldsymbol{x}$

W

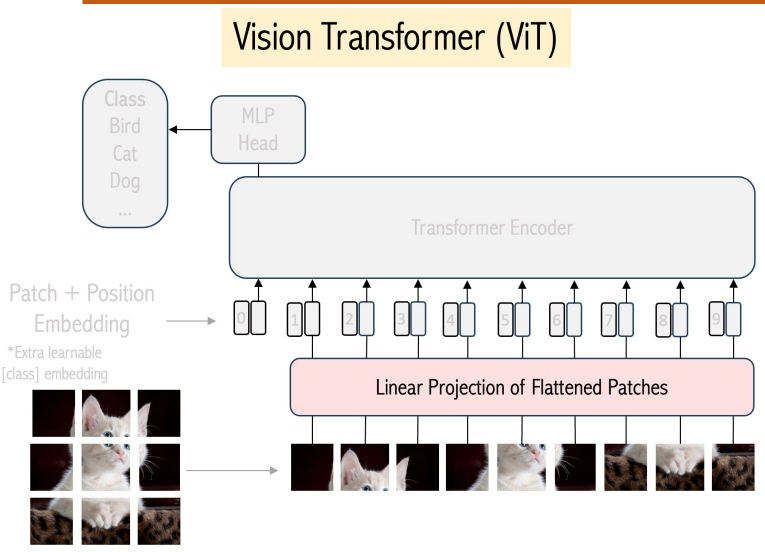
MLP

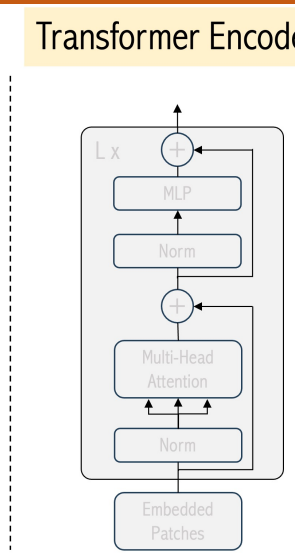
*no bias

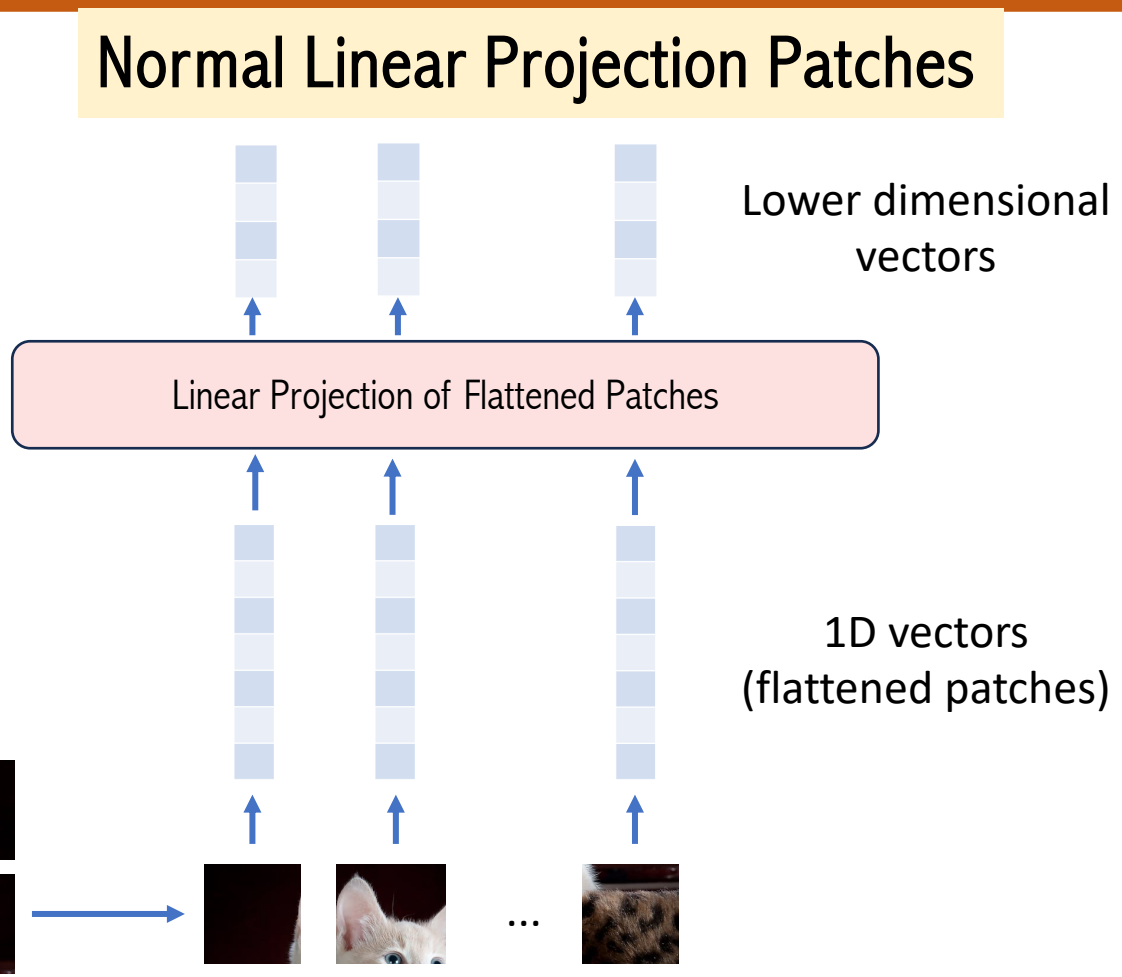
Features z_1

$N \times D$

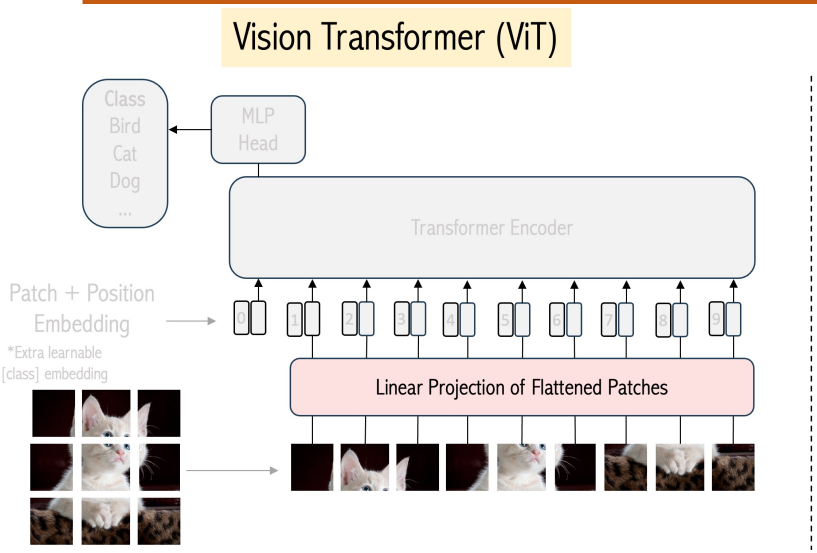
Vision Transformer

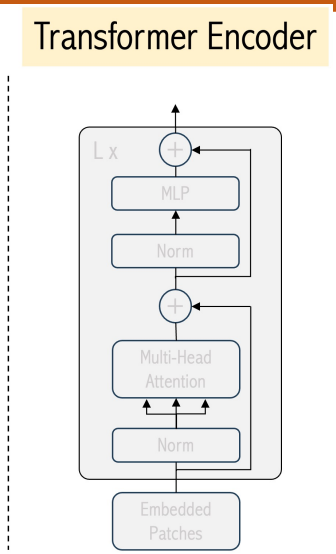






Vision Transformer





Linear Projection Patches w/ Convolution

For each patch:

- $\boldsymbol{x_p^i} \in \mathbb{R}^{1\times P^2C} \cdot \boldsymbol{W_1} \in \mathbb{R}^{P\times P\times C} = \boldsymbol{x_p^i W_1} = \boldsymbol{z_1^{i,1}} \in \mathbb{R}$
- ...
- $\boldsymbol{x_p^i} \in \mathbb{R}^{1\times P^2C} \cdot \boldsymbol{W_D} \in \mathbb{R}^{P\times P\times C} = \boldsymbol{x_p^i W_D} = \boldsymbol{z_1^{i,D}} \in \mathbb{R}$

} Number of filters

- We can use Conv2D to project patches into features instead.
- Each kernel (filter) has the same size as a patch.

$\boldsymbol{x}$

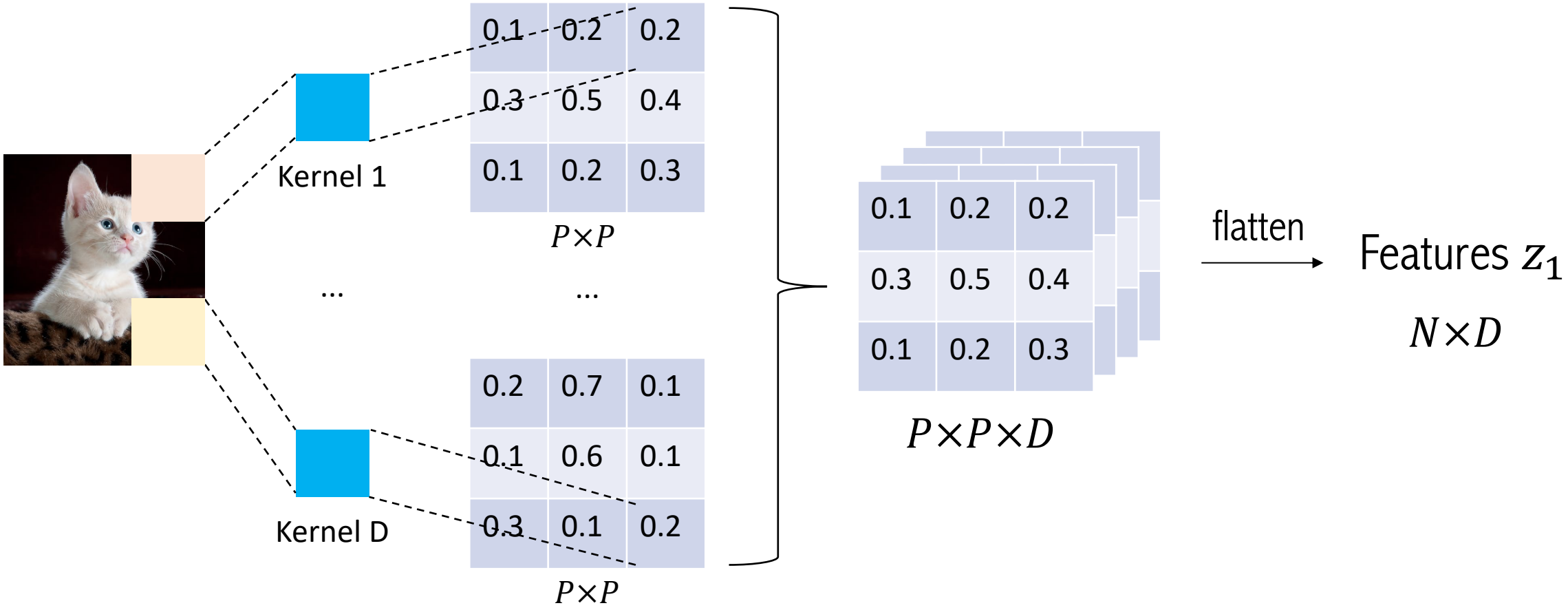
→ Conv2D →

Kernel = W
Stride = P
Out_channels = D

Features $\boldsymbol{z_1}$
$N \times D$

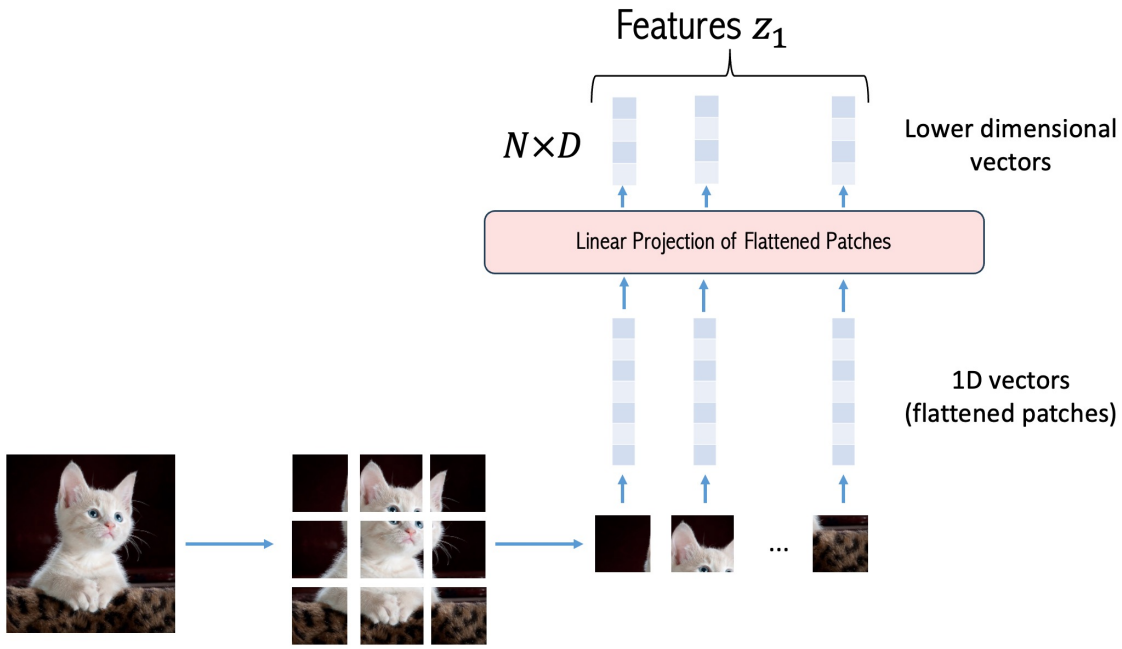
Vision Transformer

Linear Projection Patches w/ Convolution

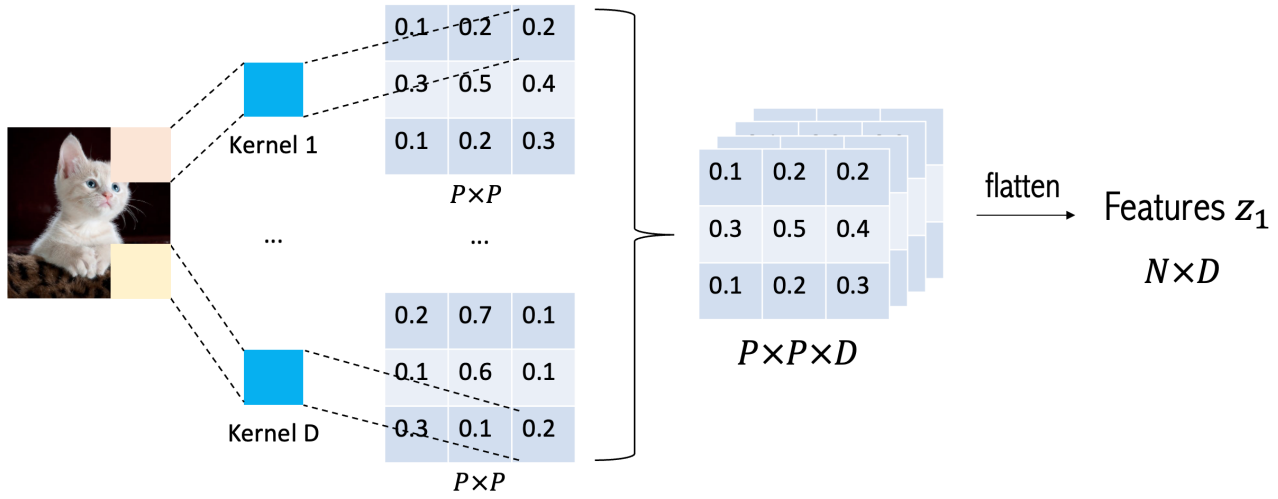


Vision Transformer

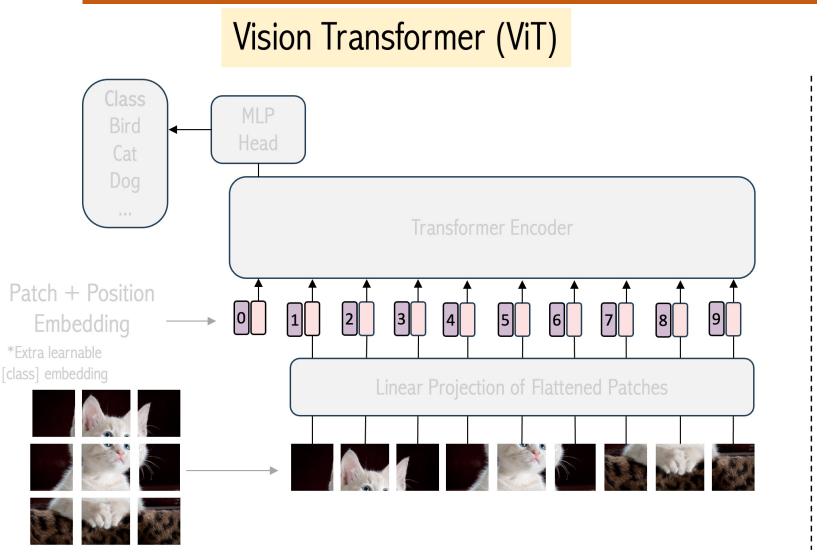
Normal Linear Projection Patches

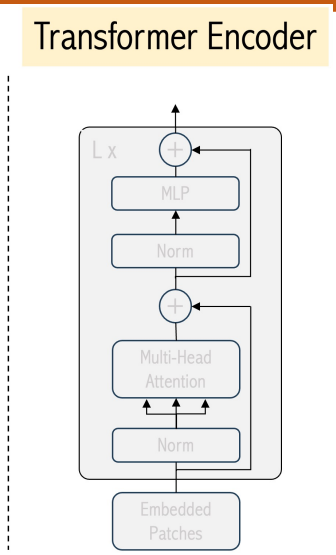


Linear Projection Patches w/ Convolution



Vision Transformer

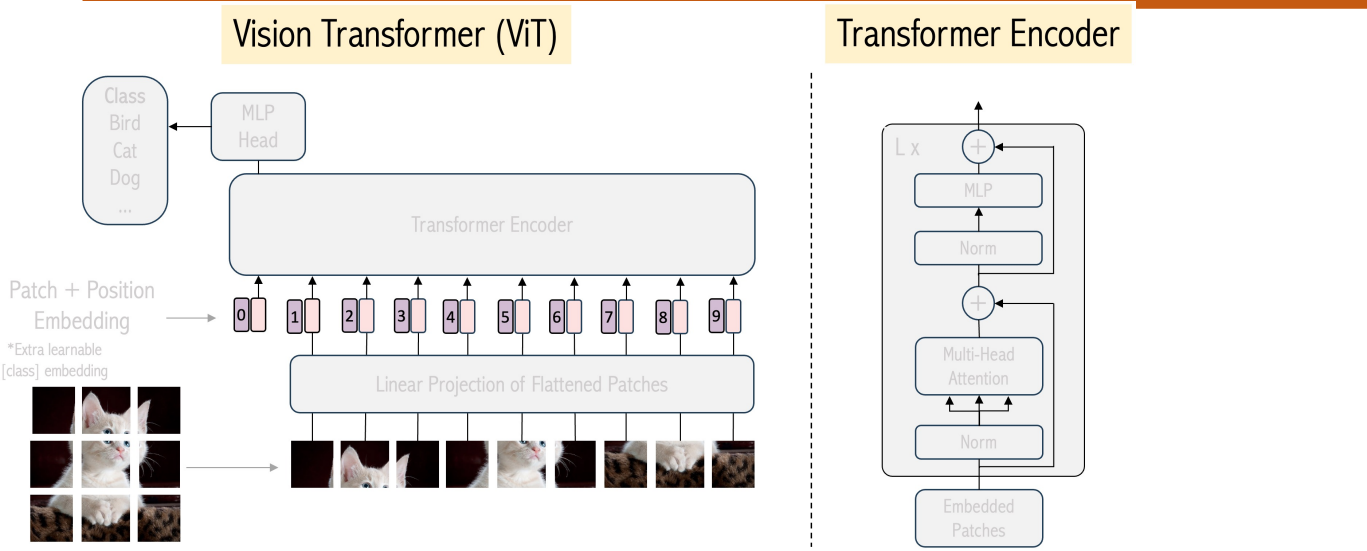




Without Position Embedding

The model still predict the correct class of an input image, but it losses the meaning because the input had shuffled.

Vision Transformer

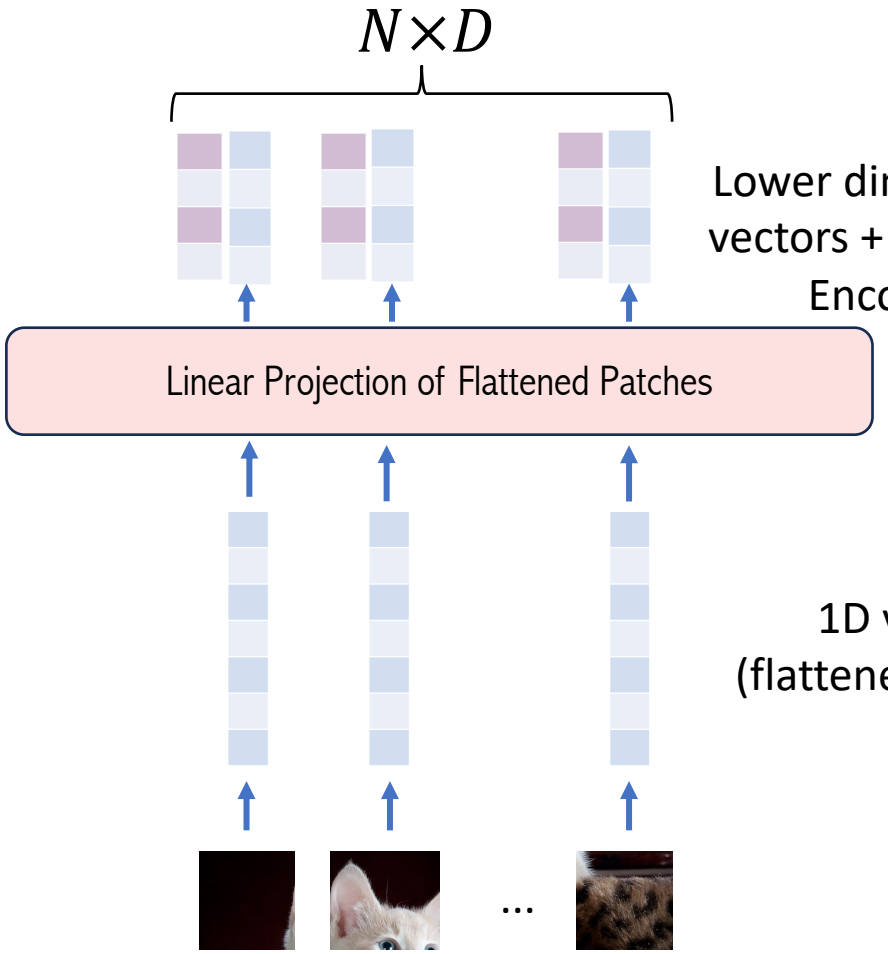


Possition	1	2	3	4	5	6	7	8	9

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$

$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$

With Position Embedding

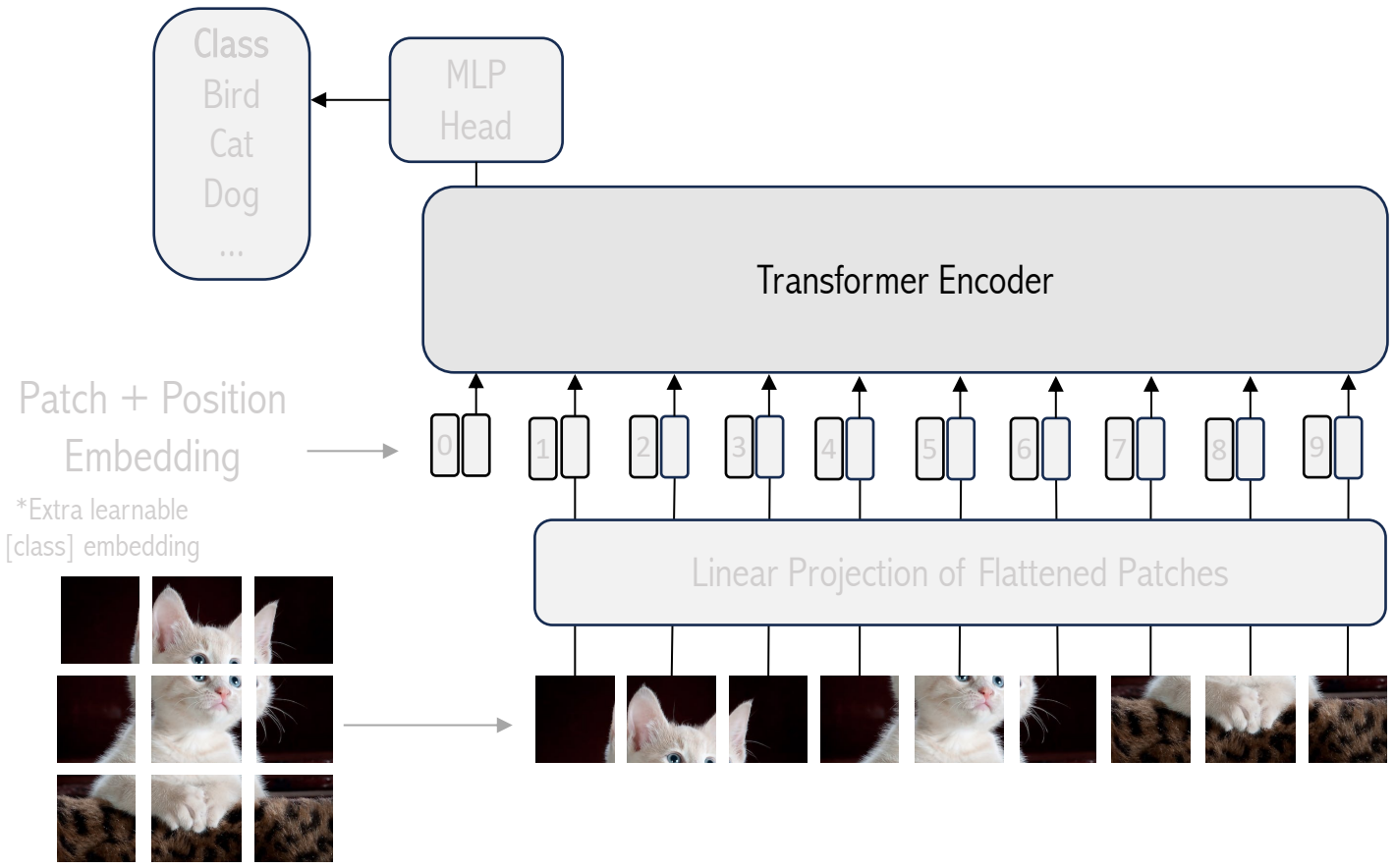


Lower dimensional vectors + Positional Encoding

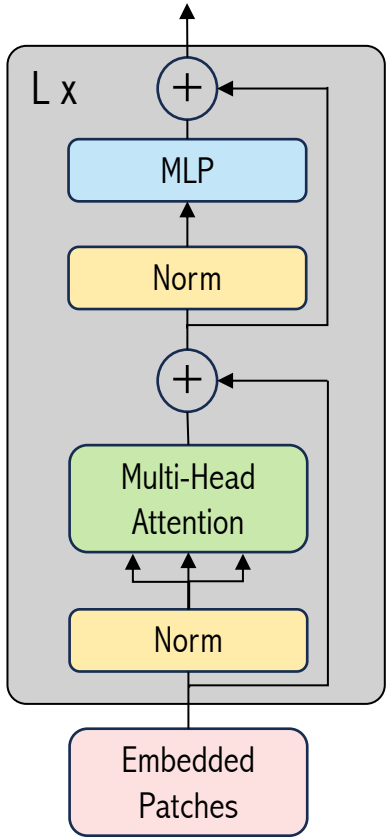
1D vectors (flattened patches)

Vision Transformer

Vision Transformer (ViT)

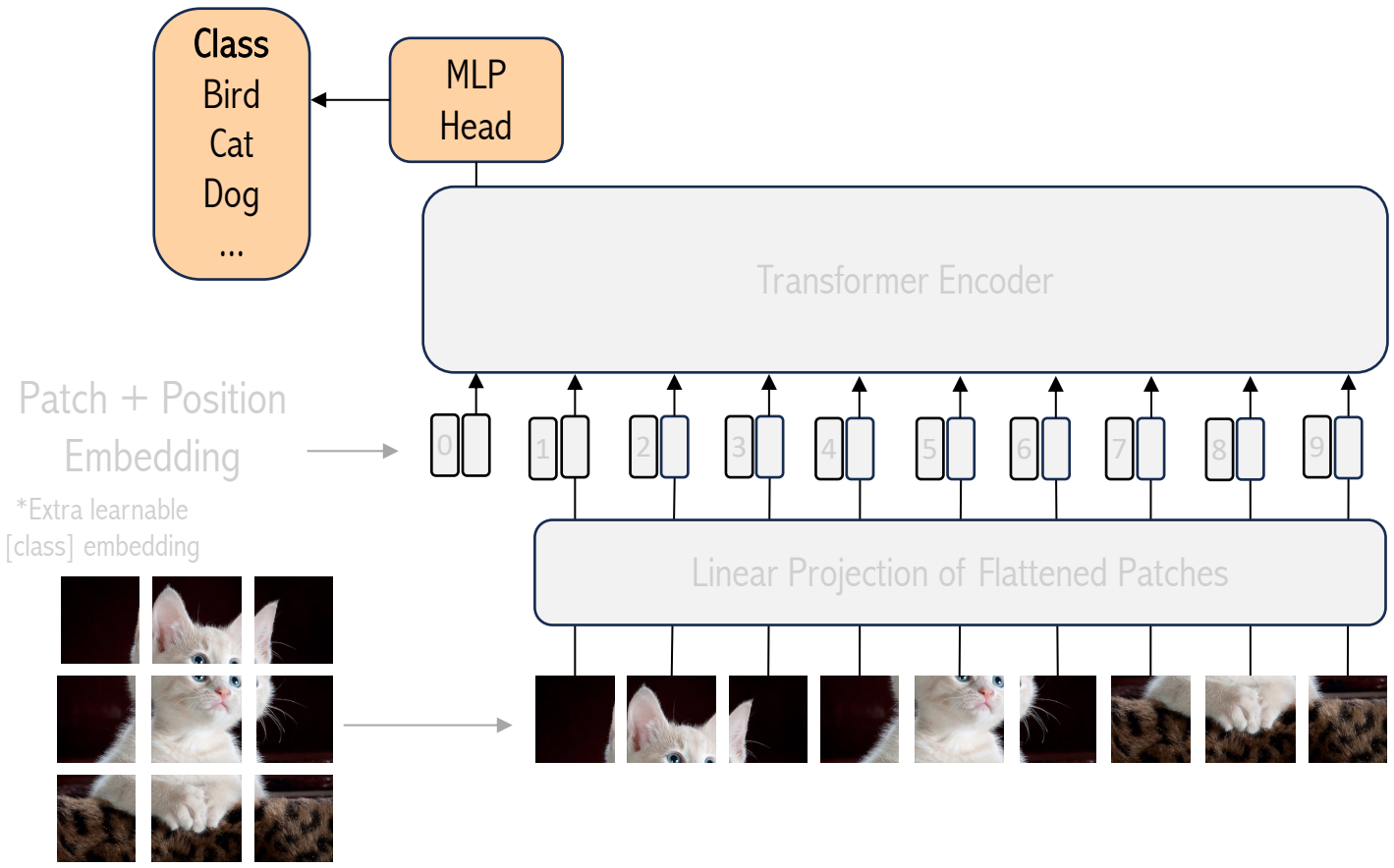


Transformer Encoder

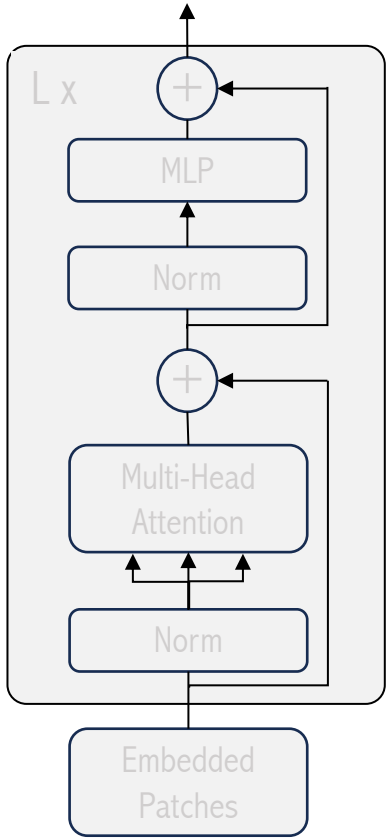


Vision Transformer

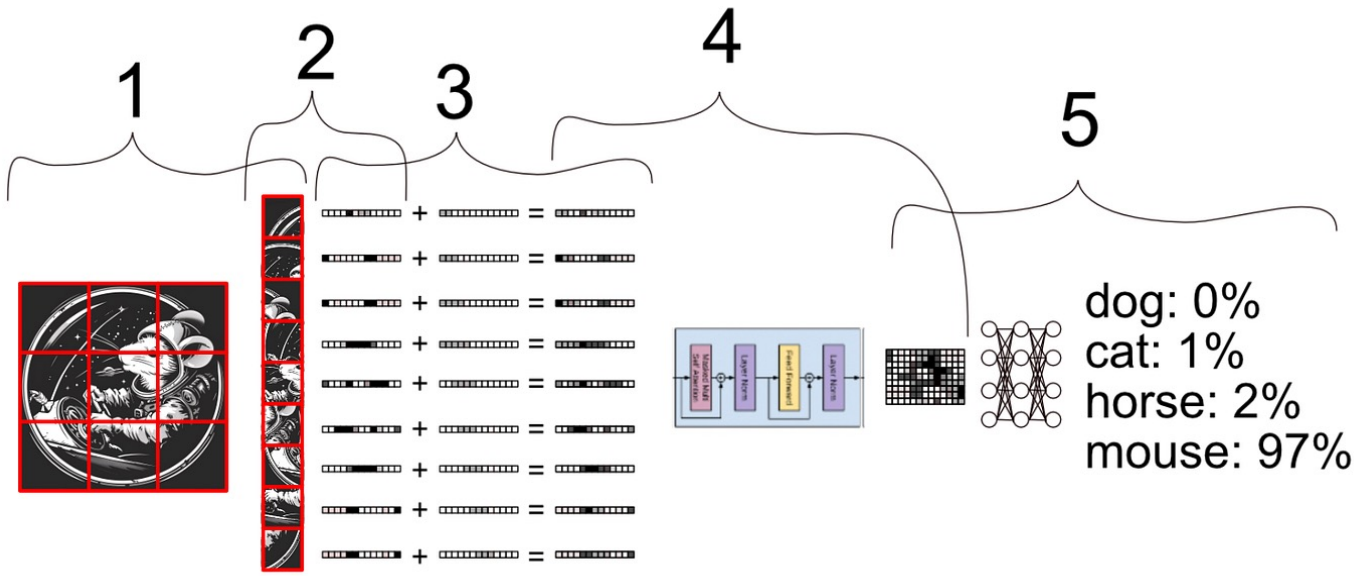
Vision Transformer (ViT)



Transformer Encoder



Vision Transformer: Summary



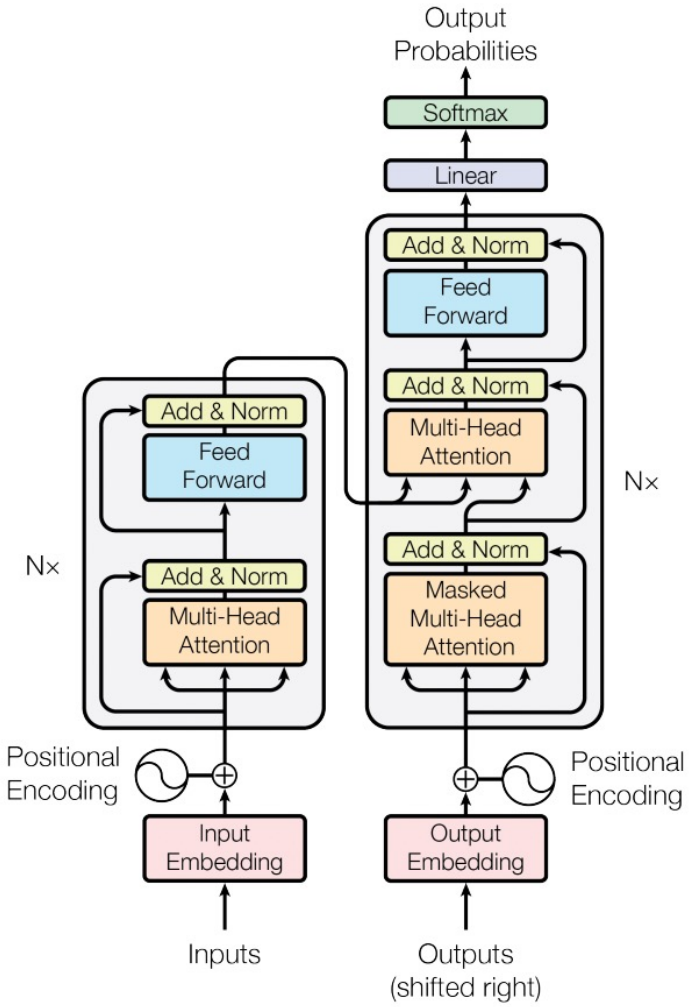
1.Broke images into patches
2.Flattened those patches into vectors
3.Added some information about where in the image the chunks came from (positional encoding)
4.Passed those vectors through a transformer
5.Took the output, put it into a dense neural network, and predicted what was in the image.


QUIZ TIME

Outline

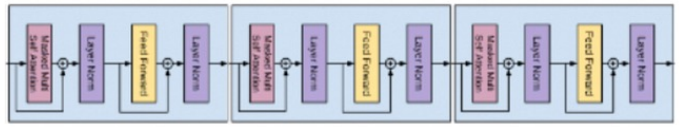
- Objective
- Diffusion Probability Model
- Transformer and ChatGPT Architecture
- Vision Transformer Architecture
- Diffusion and Transformer
- OpenAI's Sora Architecture
- Summary

Diffusion Transformer: Motivation



GPT (as in ChatGPT) pioneered the "decoder only transformer", which is a modification of the original transformer that kind of acts like a filter

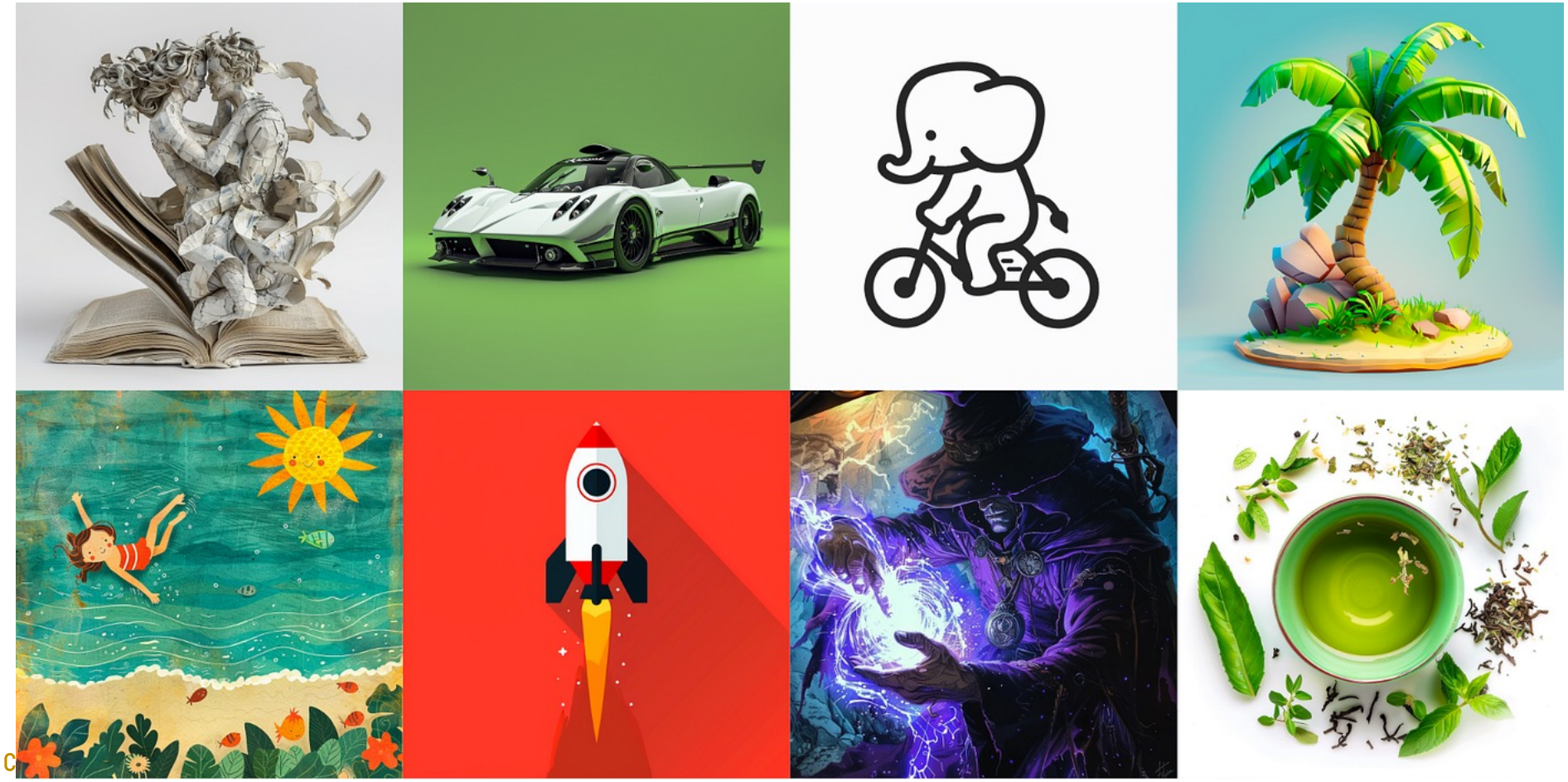
Naturally, with the success of transformers, people wanted to see what happened if you tried to use them for diffusion.



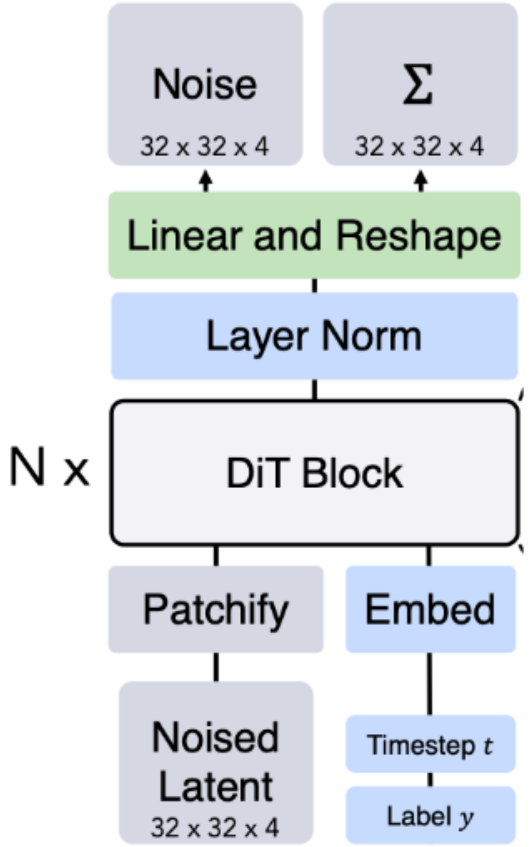
Using a transformer, which can be thought of as a really complicated filter, to turn noise into image

Diffusion Transformer: Motivation

The results were shockingly good, entering us into the era of large scale and performant image generation tools like Stable Diffusion and MidJourney.



The Architecture Behind Diffusion Transformers



Scalable Diffusion Models with Transformers

William Peebles*
UC Berkeley

Saining Xie
New York University

Figure 1. **Diffusion models with transformer backbones achieve state-of-the-art image quality.** We show selected samples from two of our class-conditional DiT-XL/2 models trained on ImageNet at 512×512 and 256×256 resolution, respectively.

Abstract

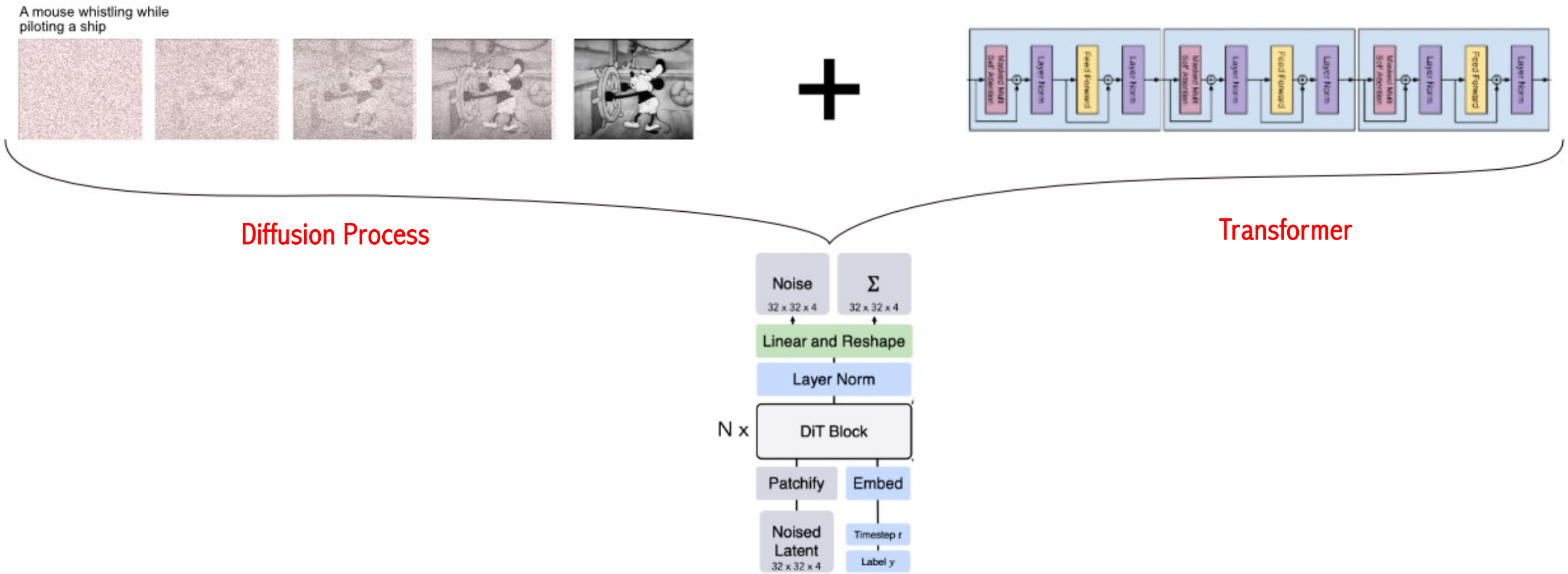
We explore a new class of diffusion models based on the transformer architecture. We train latent diffusion models of images, replacing the commonly-used U-Net backbone with a transformer that operates on latent patches. We analyze the scalability of our Diffusion Transformers (DiTs)

1. Introduction

Machine learning is experiencing a renaissance powered by transformers. Over the past five years, neural architectures for natural language processing [8, 42], vision [10] and several other domains have largely been subsumed by transformers [60]. Many classes of image-level genera-

The paper Scalable Diffusion Models with Transformers was the landmark paper which popularized transformers in diffusion models. From it's highest level, the paper takes the pre-existing vision transformer, and the pre-existing idea of diffusion models, and combines them together to make a diffusion model which leverages transformers.

The Architecture Behind Diffusion Transformers



Combining diffusion (on the left) and transformers (on the right) to make a diffusion transformer.

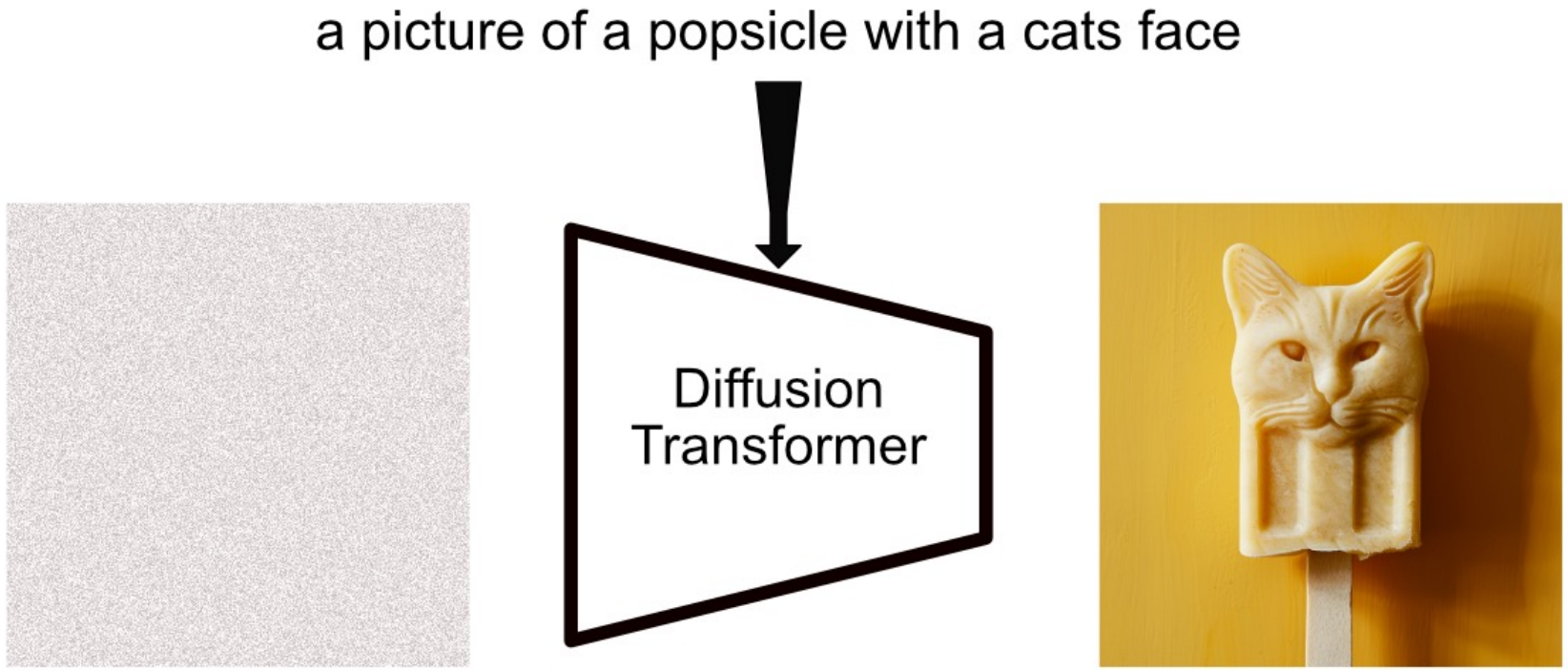
The Architecture Behind Diffusion Transformers

Question 1: How do we get a prompt into the transformer, so we can tell the transformer what image we want it to generate?

Question 2: How do we get the model to output a less noisy image?

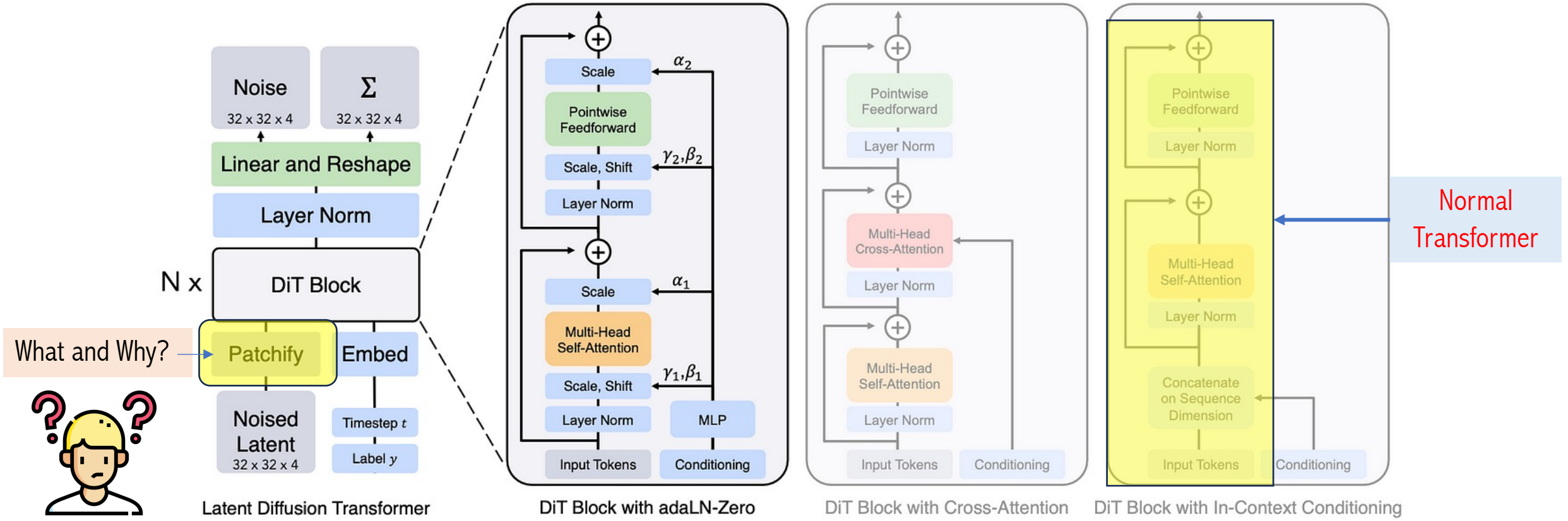
The Architecture Behind Diffusion Transformers

Question 1: How do we get a prompt into the transformer, so we can tell the transformer what image we want it to generate?

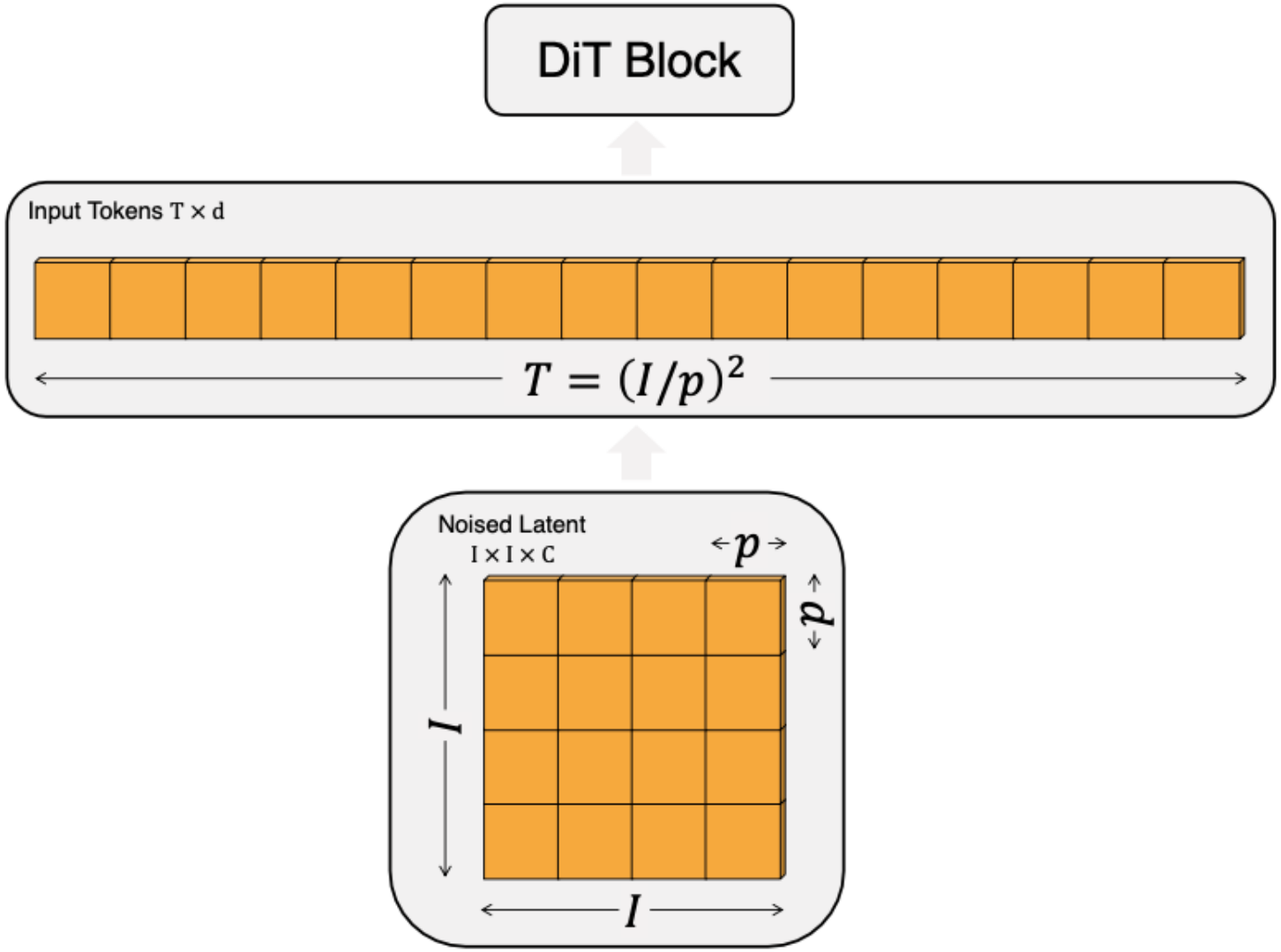


The Architecture Behind Diffusion Transformers

The Scalable Diffusion Models with Transformers paper covers three approaches to injecting text into the diffusion process:



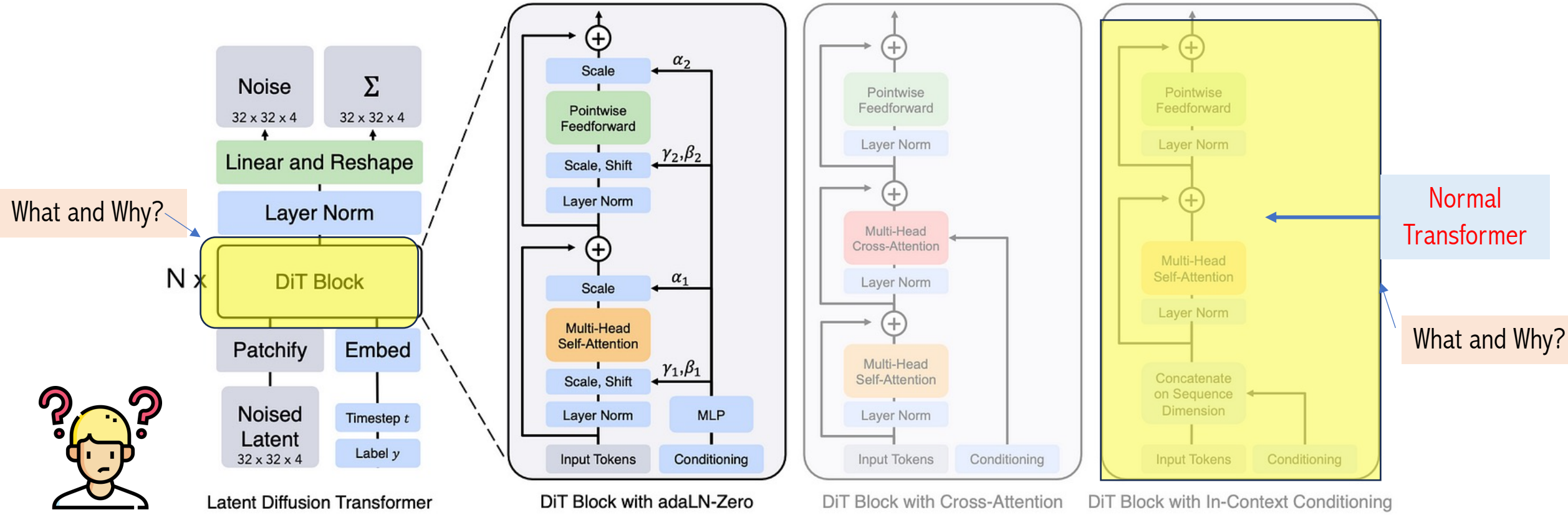
Patchify Architecture



Patchify is very simple; it divides z into a grid (32 / p) x (32 / p) x 4, where each grid element p x p x 4 is then linearly projected to become 1 x d, where d is a hyperparameter

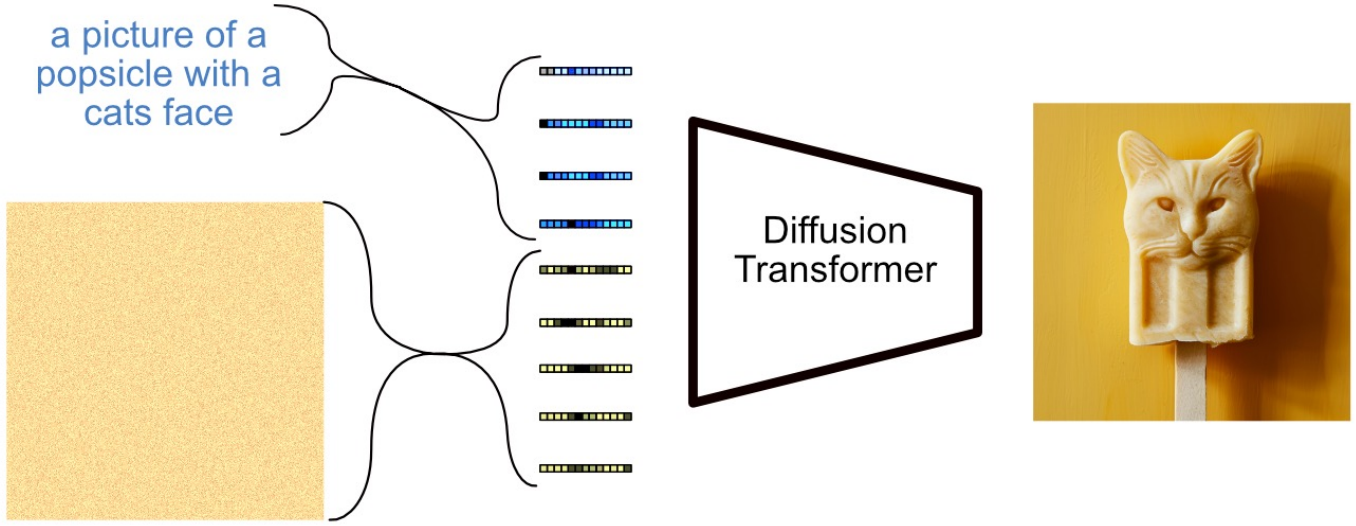
The Architecture Behind Diffusion Transformers

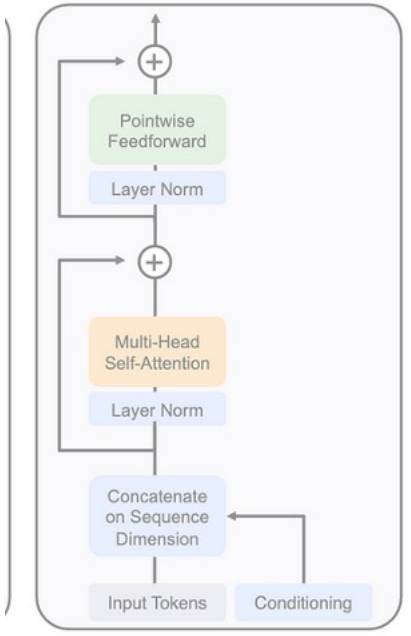
The Scalable Diffusion Models with Transformers paper covers three approaches to injecting text into the diffusion process:



The Architecture Behind Diffusion Transformers

A conceptual diagram of a DiT Block with In-Context Conditioning. i.e. just combining the text and image information together at the input.





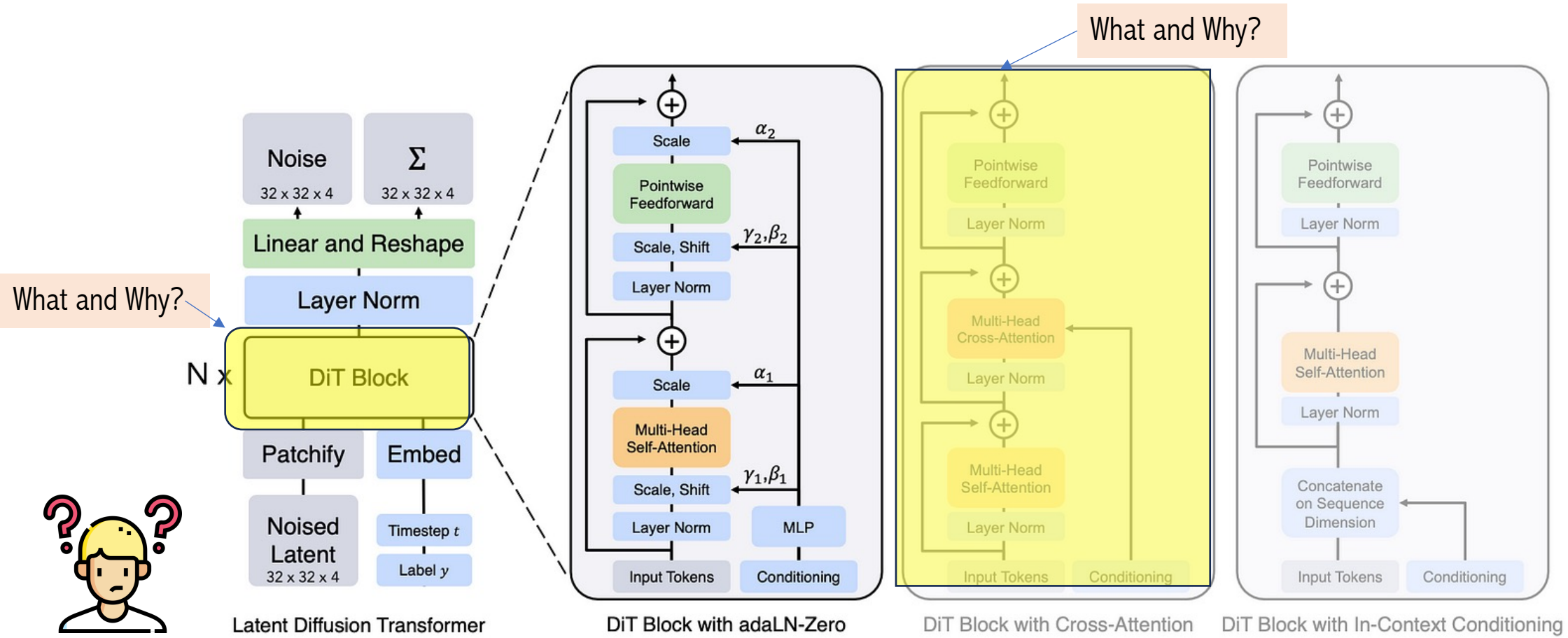
DiT Block with In-Context Conditioning

Solution 1:

This is the simplest approach, but also the least performant. In various domains it's been shown that mixing important information at various levels of a model is better than just sticking everything into the beginning and hoping the model sorts it out.

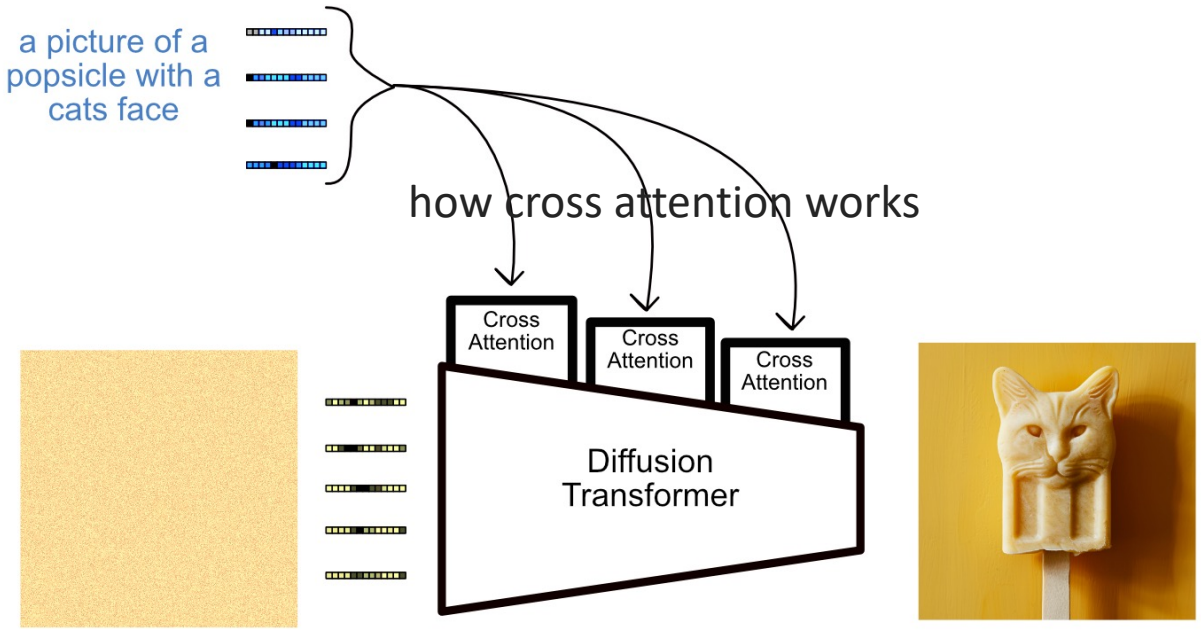
The Architecture Behind Diffusion Transformers

The Scalable Diffusion Models with Transformers paper covers three approaches to injecting text into the diffusion process:



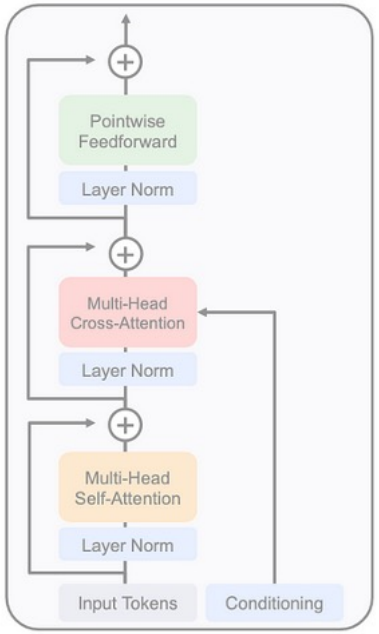
The Architecture Behind Diffusion Transformers

Another approach, which is inspired by other multimodal problems, is to use cross attention to incrementally inject the textual conditioning into the vision transformer throughout various steps.



Cross attention does the same thing but with two different inputs, allowing the model to create a highly contextualized and abstract representation based on both inputs.

Solution 2:



DiT Block with Cross-Attention

Q K,V

attention mechanisms ain't cheap, and while this is a highly performant strategy, it comes at a steep price tag.

Flamingo: a Visual Language Model for Few-Shot Learning

Jean-Baptiste Alayrac*,‡ Jeff Donahue* Pauline Luc* Antoine Miech*
Iain Barr† Yana Hasson† Karel Lenc† Arthur Mensch† Katie Millican†
Malcolm Reynolds† Roman Ring† Eliza Rutherford† Serkan Cabi Tengda Han
Zhitao Gong Sina Samangooei Marianne Monteiro Jacob Menick
Sebastian Borgeaud Andrew Brock Aida Nematzadeh Sahand Sharifzadeh
Mikolaj Binkowski Ricardo Barreira Oriol Vinyals Andrew Zisserman
Karen Simonyan*,‡

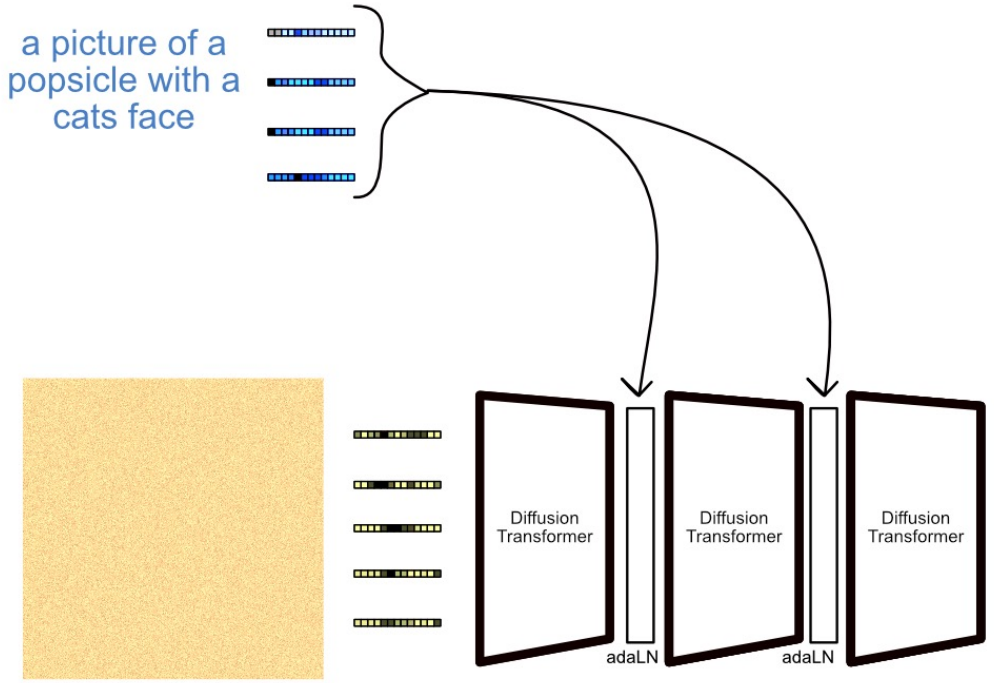
* Equal contributions, ordered alphabetically, † Equal contributions, ordered alphabetically, ‡ Equal senior contributions

DeepMind

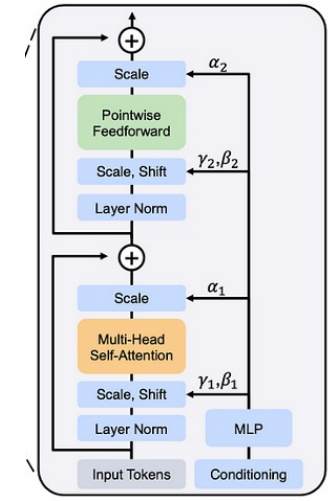
The Architecture Behind Diffusion Transformers

A third option, which the Scalable Diffusion Models with Transformers paper used, was adaLN-Zero.

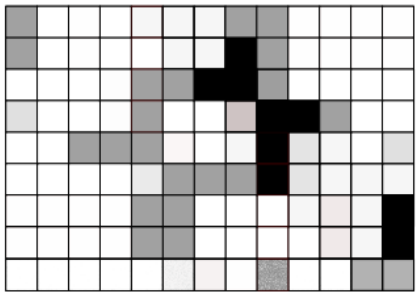
Solution 3:

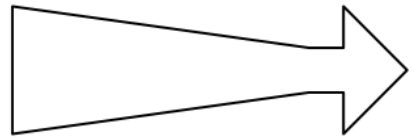


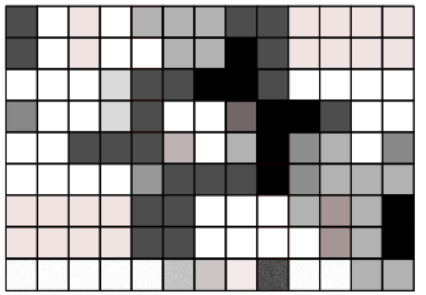
A conceptual diagram of layer norm (or, layer normalization)



DiT Block with adaLN-Zero





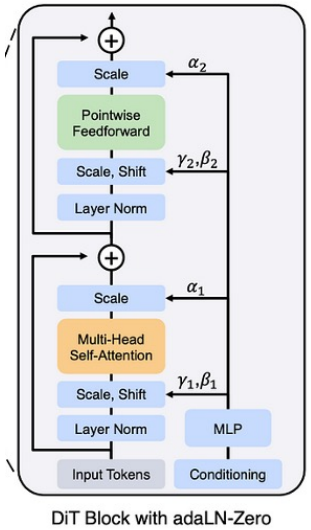


adaLN-Zero (adaptive layer normalization-Zero) allows a text to interact with image information in a very cost-efficient way

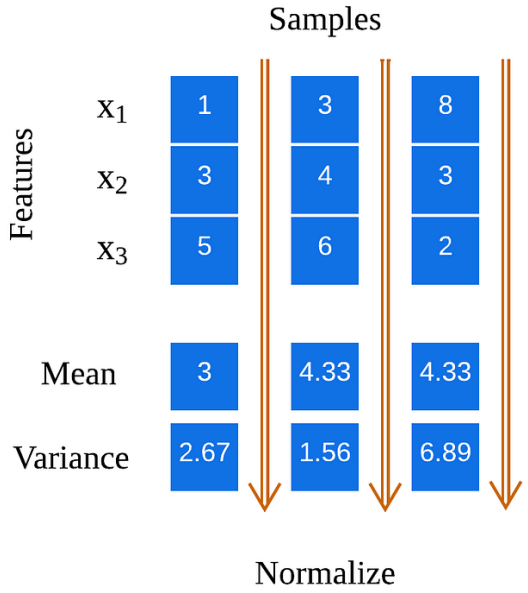
The Architecture Behind Diffusion Transformers

A third option, which the Scalable Diffusion Models with Transformers paper used, was adaLN-Zero.

Solution 3:



DiT Block with adaLN-Zero



(1)
$$\mu_l \leftarrow \frac{1}{d}\sum_{i=1}^{d} x_i$$

(2)
$$\sigma_l^2 \leftarrow \frac{1}{d}\sum_{i=1}^{d}(x_i - \mu_l)^2$$

(3)
$$\hat{x}_i \leftarrow \frac{x_i - \mu_l}{\sqrt{\sigma_l^2 + \epsilon}}$$

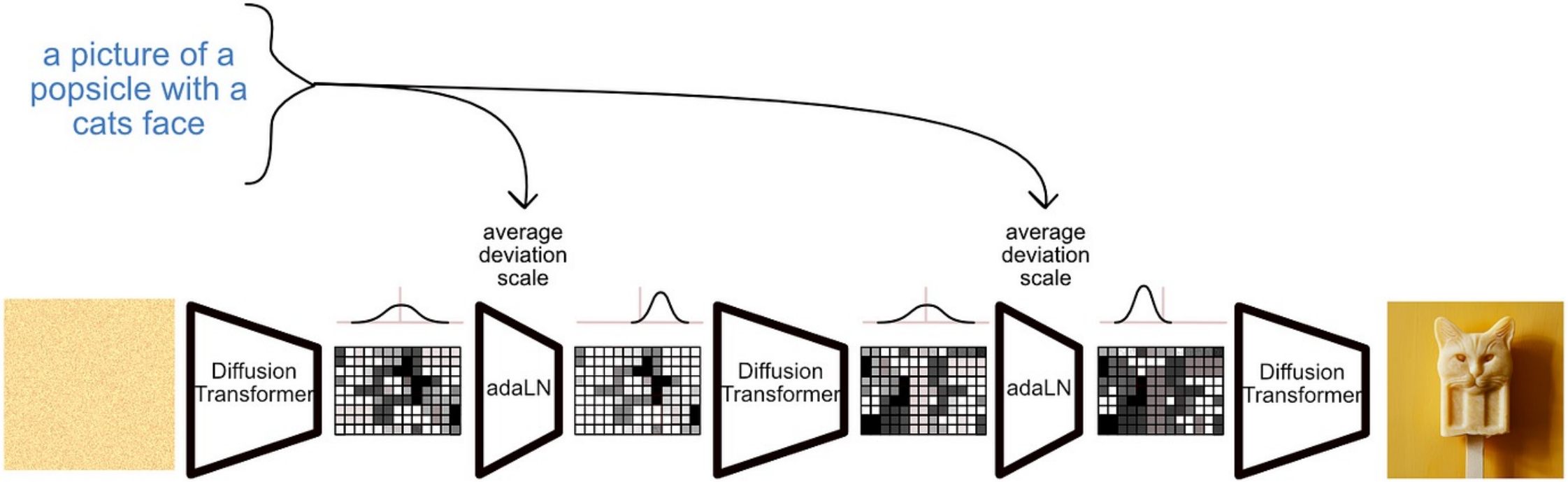
(4)
$$y_i \leftarrow \gamma \hat{x}_i + \beta = LN_{\gamma,\beta}(x_i)$$

How Layer Normalization (LN) works

The Architecture Behind Diffusion Transformers

Solution 3:

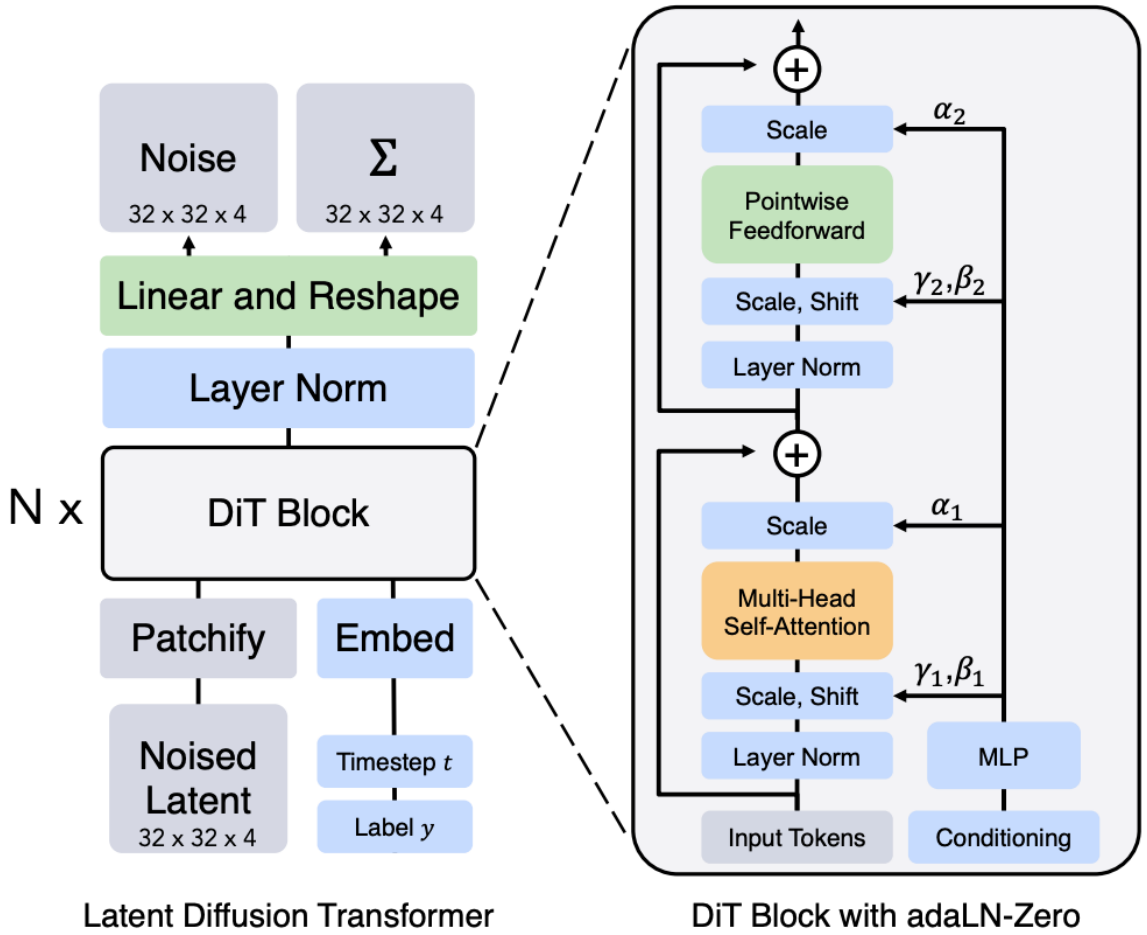
A third option, which the Scalable Diffusion Models with Transformers paper used, was adaLN-Zero.



A conceptual diagram of how adaLN allows text to control distributions from within the transformer

The Architecture Behind Diffusion Transformers

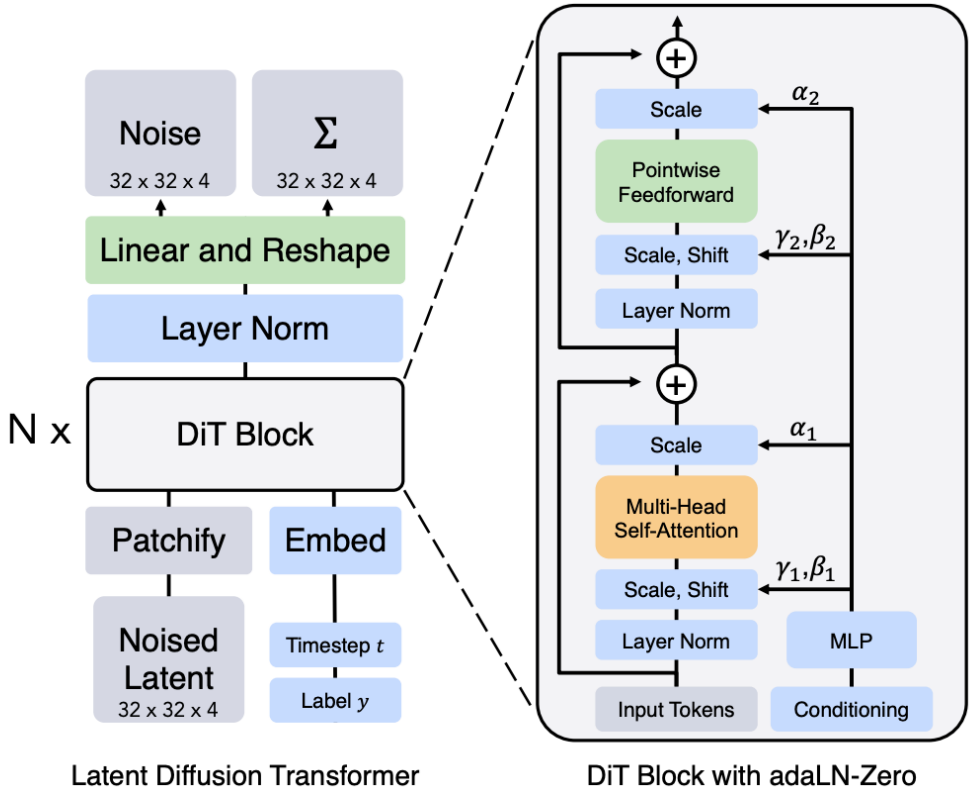
Question 2: How do we get the model to output a less noisy image?



So, instead of the hefty amount of parameters in cross attention, adLN employs three: mean, standard deviation, and scale (as well as the parameters in the text encoder, naturally).

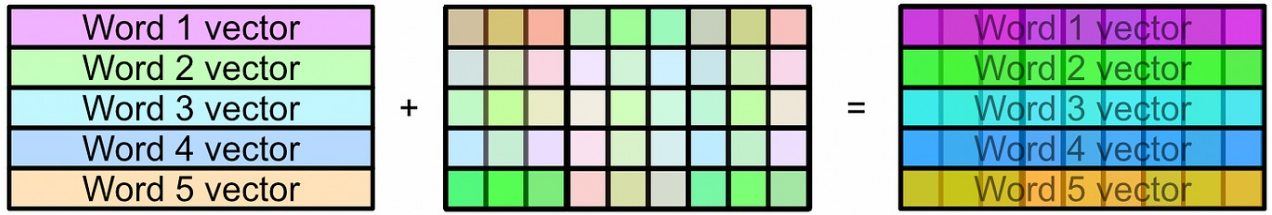
The Architecture Behind Diffusion Transformers

Question 2: How do we get the model to output a less noisy image?



Latent Diffusion Transformer

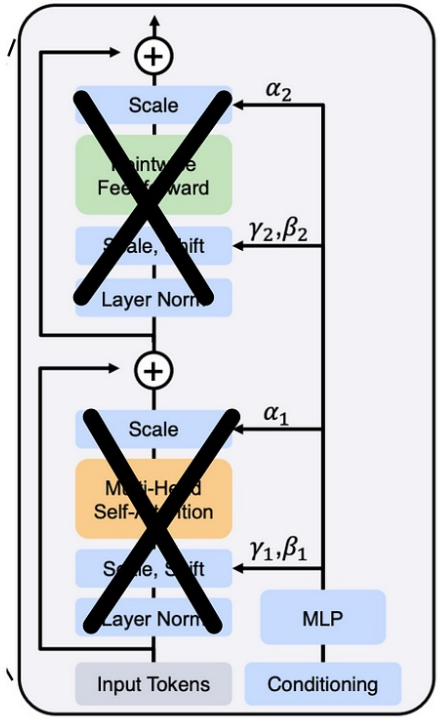
DiT Block with adaLN-Zero



Transformers employ skip connections which allow old data to be combined with new data; a strategy that's been shown to improve the performance of large models. By allowing the text to scale the importance of information before the addition of the skip connection, adaLN effectively lets the text decide how much a certain operation should contribute to the data.

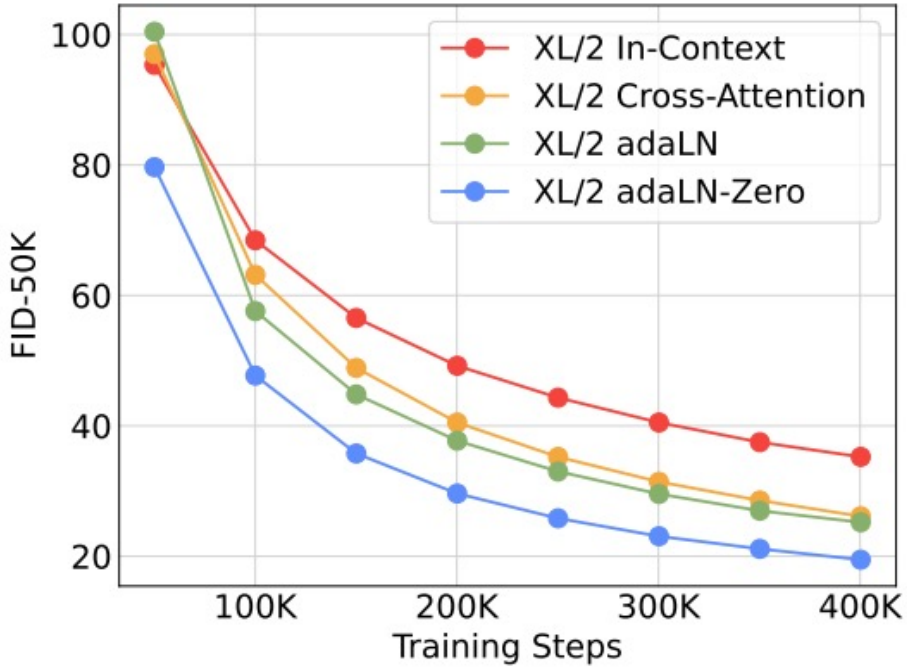
The Architecture Behind Diffusion Transformers

Question 2: How do we get the model to output a less noisy image?



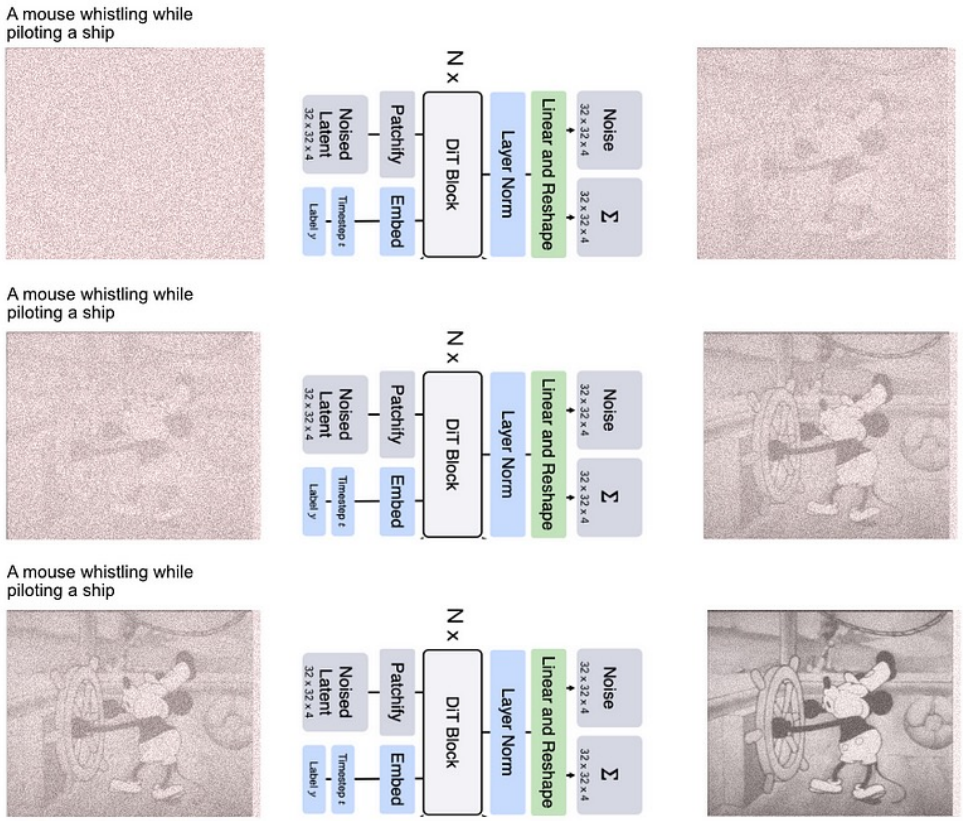
DiT Block with adaLN-Zero

The Scalable Diffusion Models with Transformers paper doesn't recommend adaLN, but "adaLN-Zero". The only real difference here is in initialization. Some research has suggested that setting certain key values to zero at the beginning of training can improve performance.



The Architecture Behind Diffusion Transformers

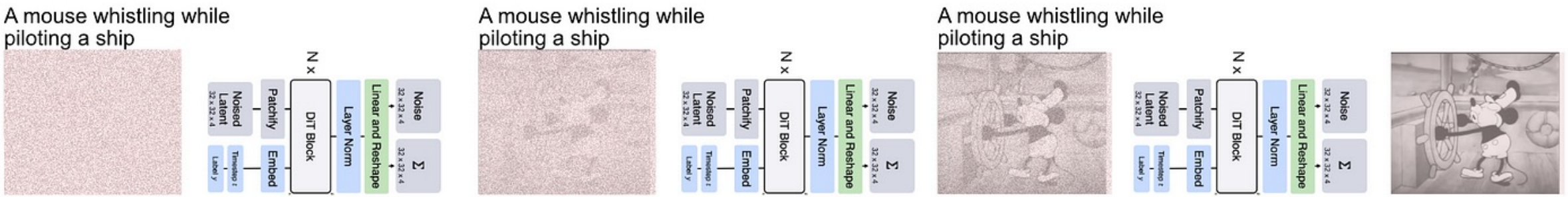
Question 2: How do we get the model to output a less noisy image?



A naive approach to training a diffusion transformer. Feed in a noisy image, and expect a slightly less noisy output.

The Architecture Behind Diffusion Transformers

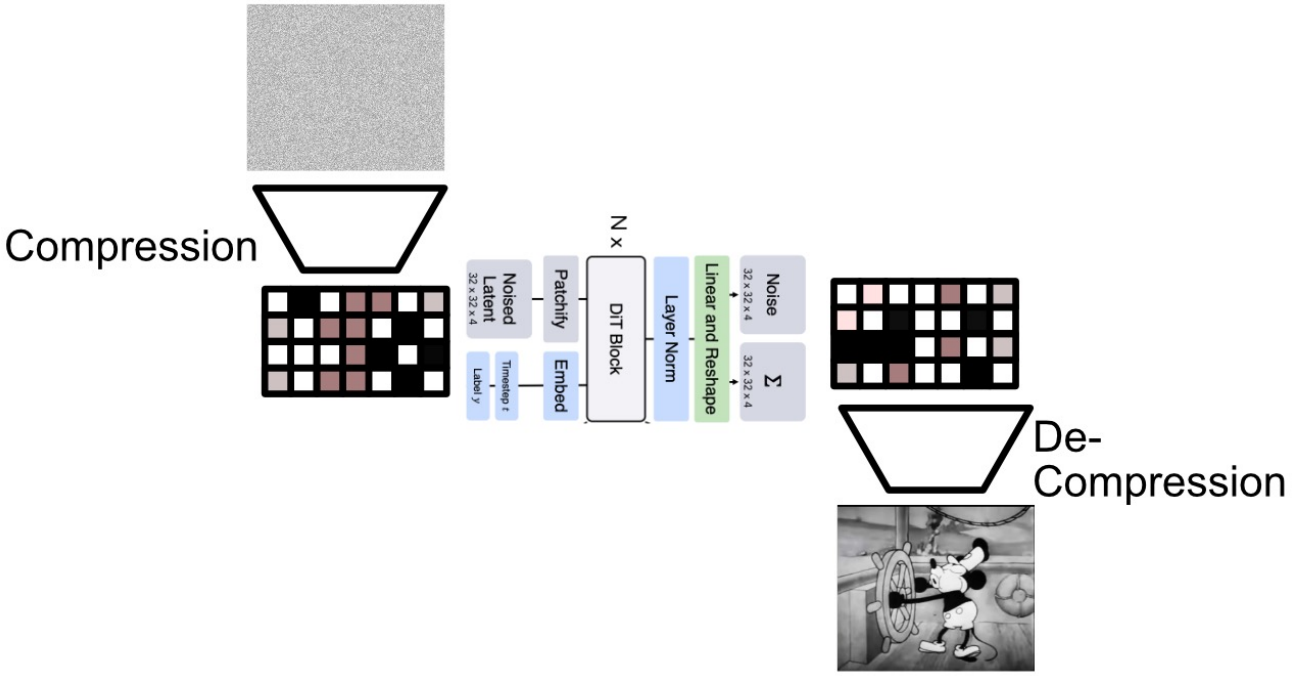
Question 2: How do we get the model to output a less noisy image?

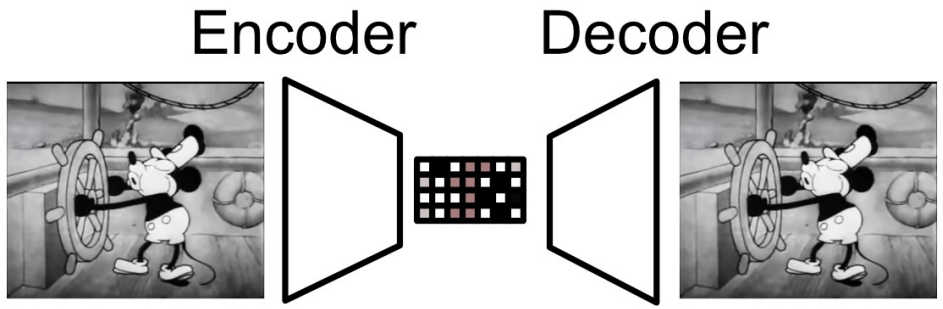


An example of, if a diffusion transformer were trained naively, what it might do to denoise an image.

The Architecture Behind Diffusion Transformers

Question 2: How do we get the model to output a less noisy image?

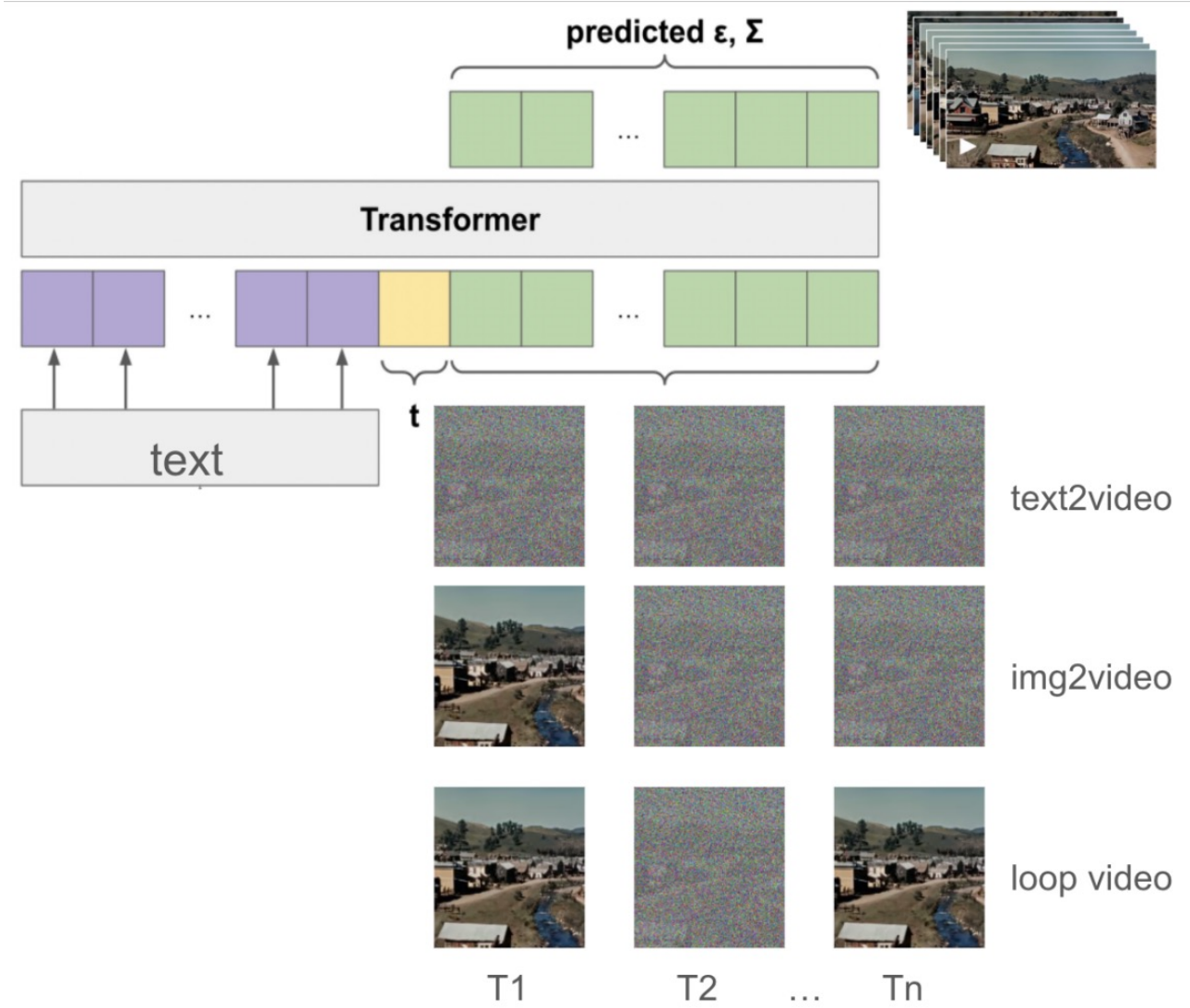




Diffusion transformers don't work in images, but in "latent image embeddings"

Diffusion Transformer

- how to compress the video spatially and temporally to a latent space for efficient denoising;
- how to convert the compressed latent to patches and feed them to the transformer;
- how to handle long-range temporal and spatial dependencies and ensure content consistency.

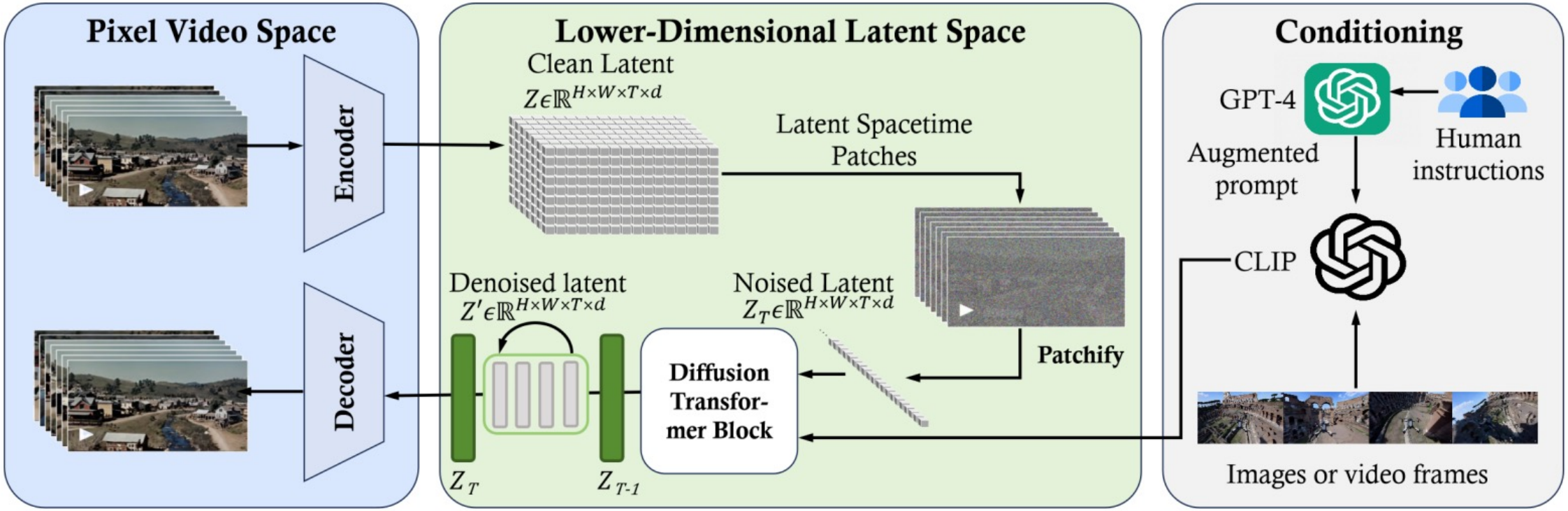


Outline

- Objective
- Diffusion Probability Model
- Transformer and ChatGPT Architecture
- Vision Transformer Architecture
- Diffusion and Transformer
- OpenAI's Sora Architecture
- Summary

Sora Achitecture

Basically, instead of feeding noisy images in and getting images out like in the traditional diffusion transformer, Sora puts in noisy videos in and gets video out.



Sora : Data Pre-processing

Variable Durations, Resolutions, Aspect Ratios. Traditional methods often resize, crop, or adjust the aspect ratios of videos to fit a uniform standard

(a) Vertical

(b) Square

(c) Horizontal

Sora can generate images in flexible sizes or resolutions ranging from 1920x1080p to 1080x1920p and anything in between.

Sora : Data Pre-processing

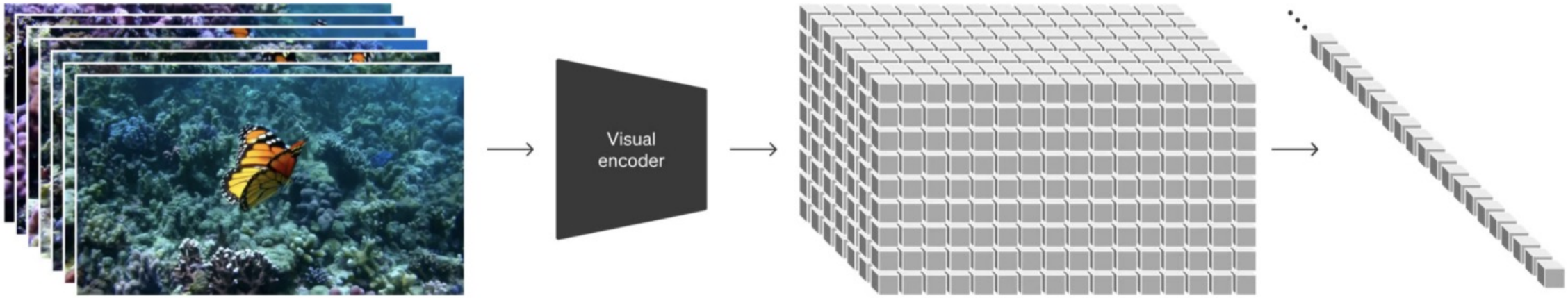
Variable Durations, Resolutions, Aspect Ratios. Traditional methods often resize, crop, or adjust the aspect ratios of videos to fit a uniform standard

(a) Training on videos that are cropped to squares leads to unnatural compositions and framing.

(b) Training in native sizes improves framing.

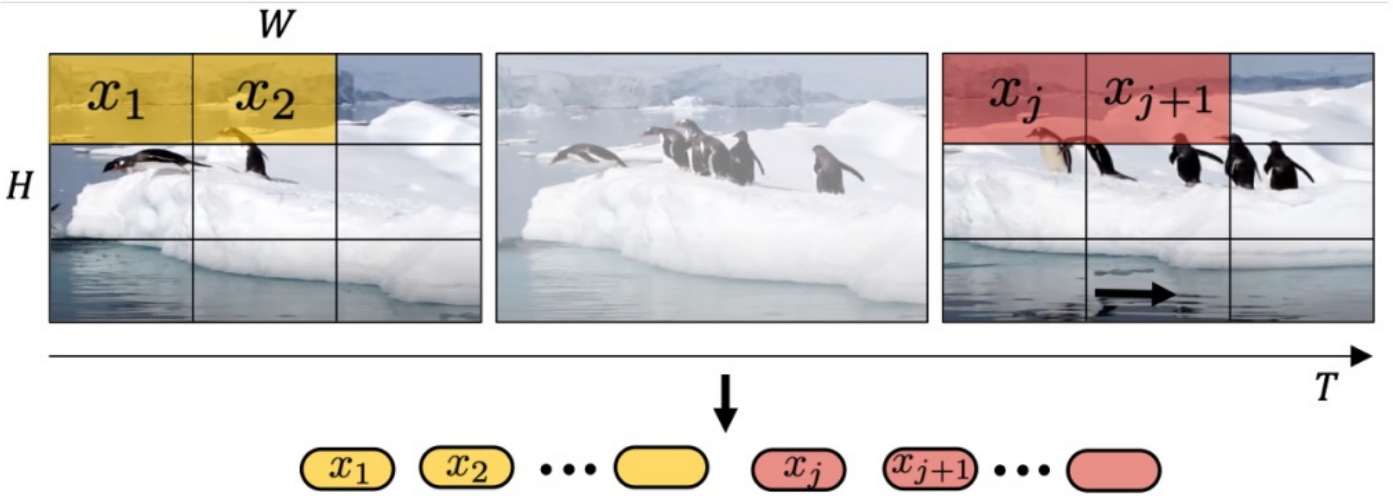
Sora can generate images in flexible sizes or resolutions ranging from A comparison between Sora (right) and a modified version of the model (left), which crops videos to square shapes—a common practice in model training—highlights the advantages to 1080x1920p and anything in between.

Sora: Unified Visual Representation

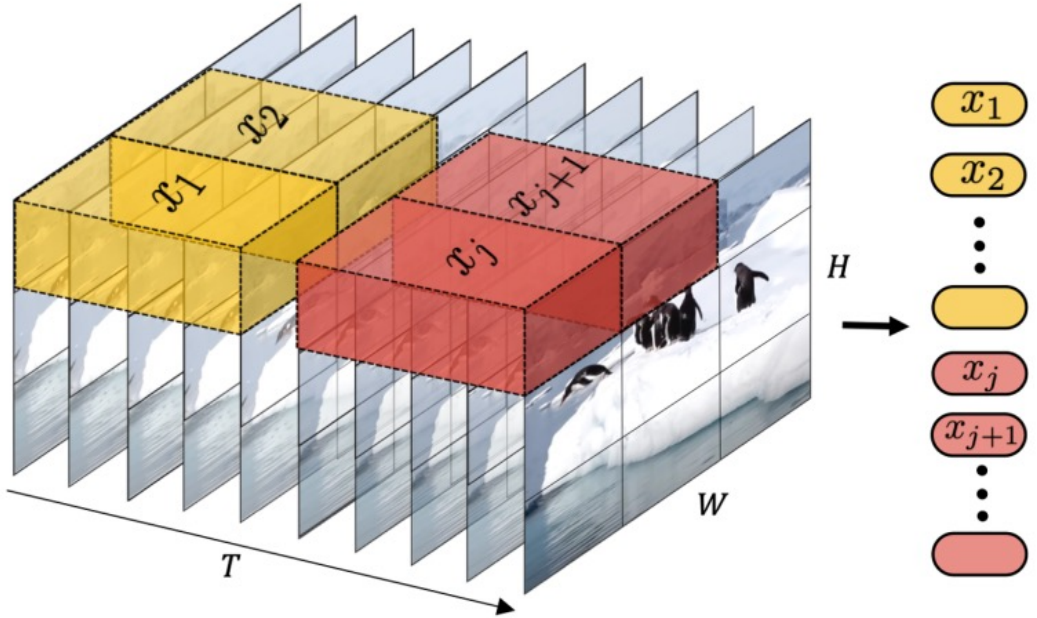


Sora turns videos into patches by first compressing videos into a lowerdimensional latent space, and subsequently decomposing the representation into spacetime patches

Sora: Video Compression Network

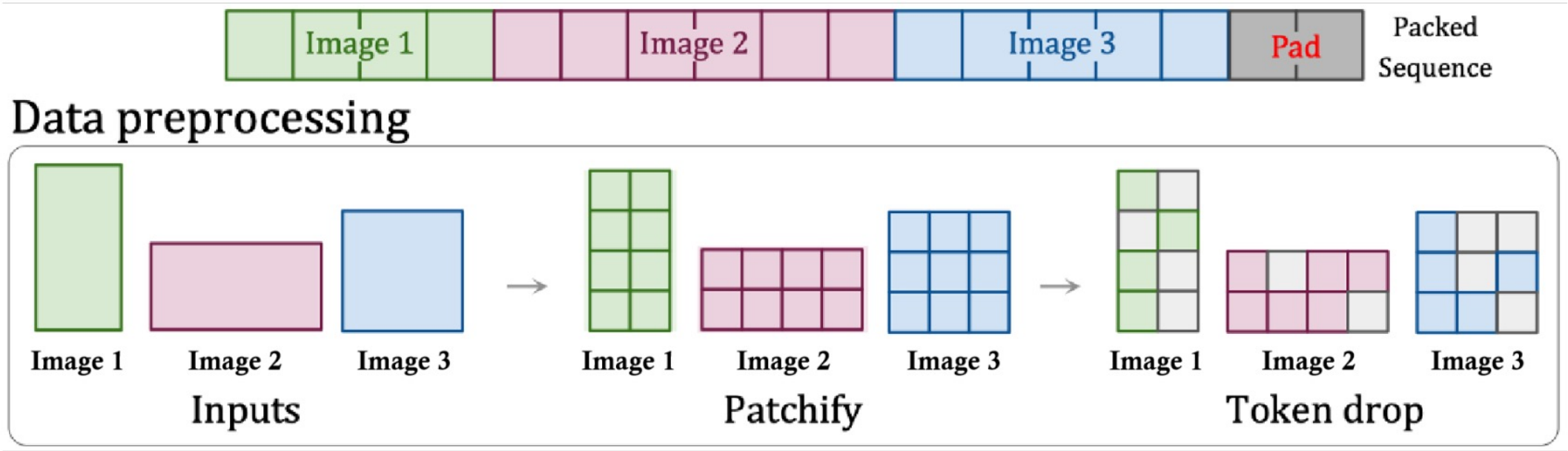


Spatial patchification simply samples nt frames and embeds each 2D frame independently following ViT



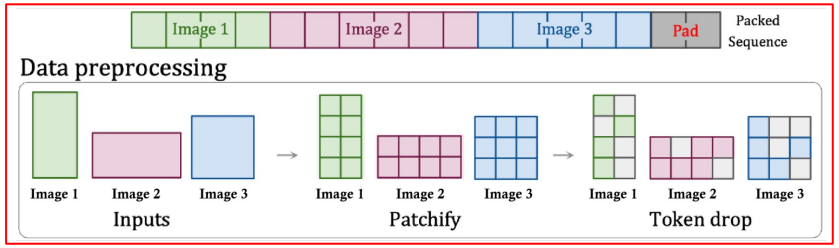
Spatial-temporal patchification extracts and linearly embeds non-overlapping or overlapping tubelets that span the spatiotemporal input volume

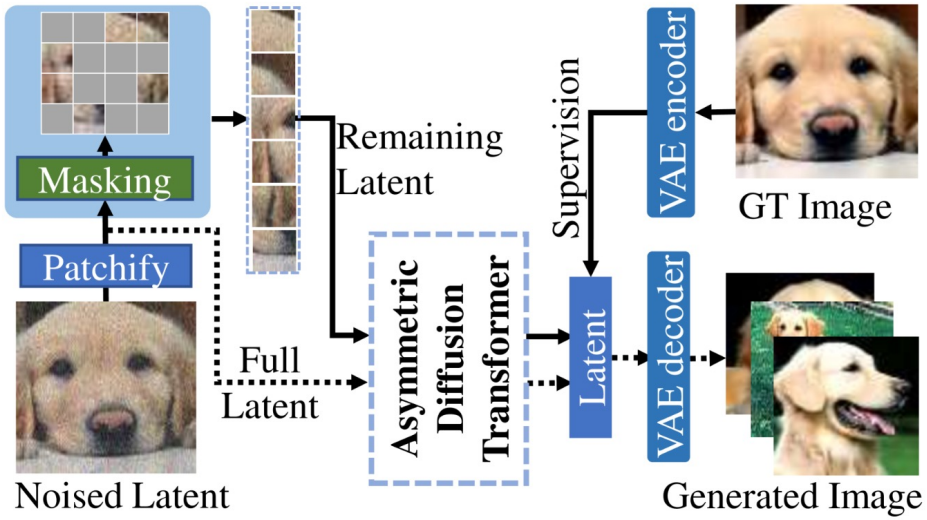
Sora: Video Compression Network



Patch packing enables variable resolution images or videos with preserved aspect ratio.6 Token dropping somehow could be treated as data augmentation

Sora: Video Compression Network





The overall framework of Masked Diffusion Transformer (MDT). Solid/dotted lines indicate each time step's training/inference process. Masking and side-interpolater are only used during training and are removed during inference.

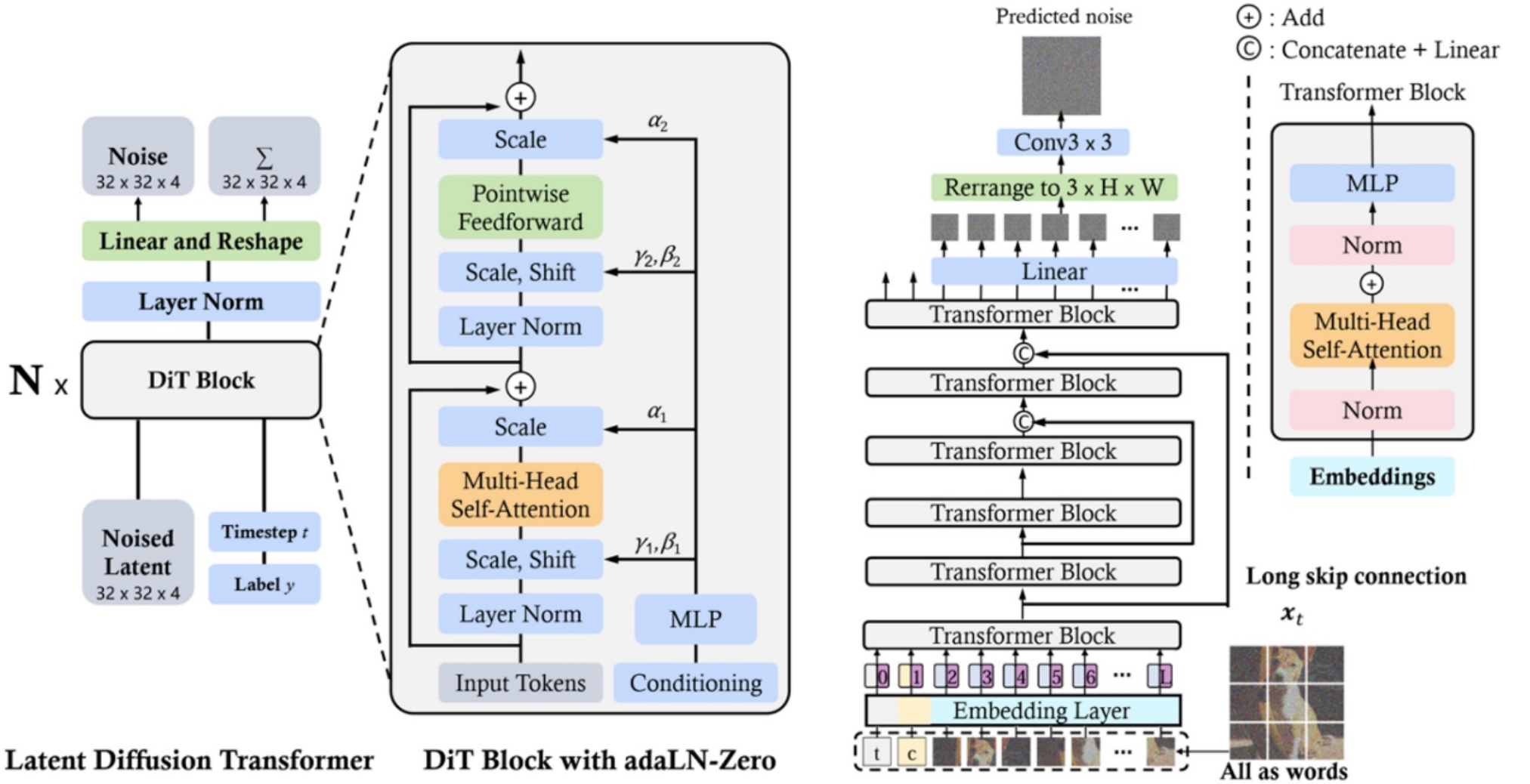
MDTv2: Masked Diffusion Transformer is a Strong Image Synthesizer

Shanghua Gao, Pan Zhou, Ming-Ming Cheng, and Shuicheng Yan

Abstract—Despite its success in image synthesis, we observe that diffusion probabilistic models (DPMs) often lack contextual reasoning ability to learn the relations among object parts in an image, leading to a slow learning process. To solve this issue, we propose a Masked Diffusion Transformer (MDT) that introduces a mask latent modeling scheme to explicitly enhance the DPMs' ability to contextual relation learning among object semantic parts in an image. During training, MDT operates in the latent space to mask certain tokens. Then, an asymmetric diffusion transformer is designed to predict masked tokens from unmasked ones while maintaining the diffusion generation process. Our MDT can reconstruct the full information of an image from its incomplete contextual input, thus enabling it to learn the associated relations among image tokens. We further improve MDT with a more efficient macro network structure and training strategy, named MDTv2. Experimental results show that MDTv2 achieves superior image synthesis performance, *e.g.,* a new SOTA FID score of 1.58 on the ImageNet dataset, and has more than 10× faster learning speed than the previous SOTA DiT. The source code is released at https://github.com/sail-sg/MDT.

Index Terms—Masked diffusion transformer, image generation

Sora Architecture



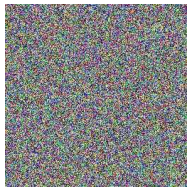
Generating videos provided an image and prompt as input

Image

Video

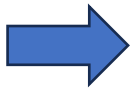
C

X^t



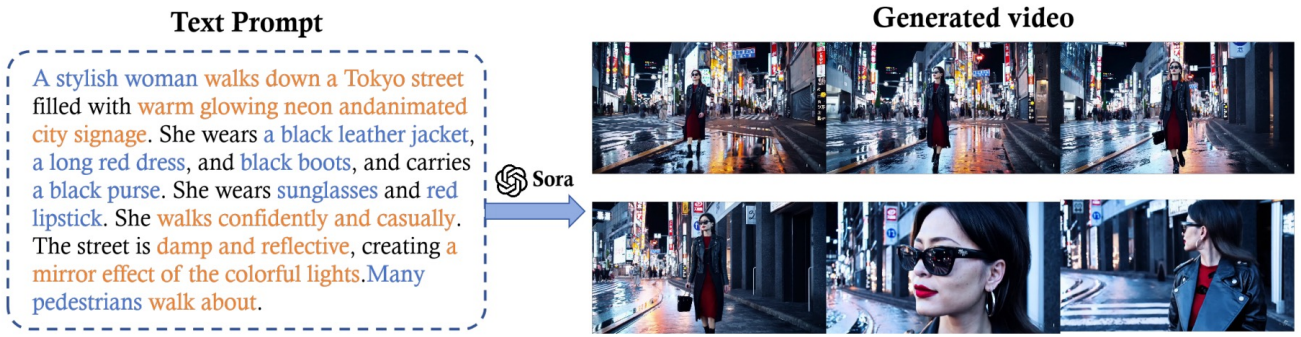
Mask

1	0	0	0	0



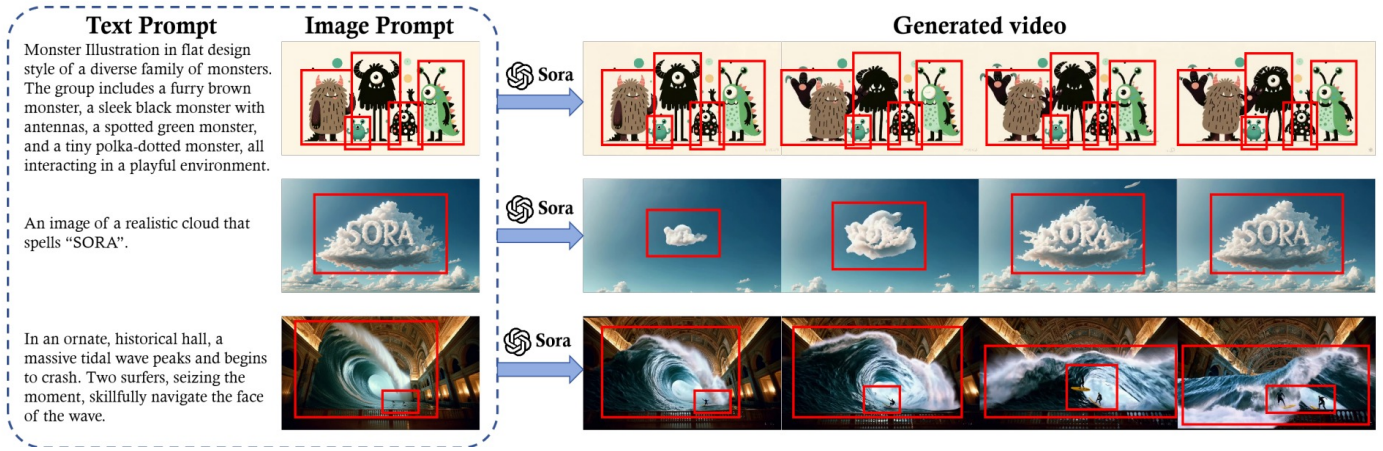
Diffusion Transformer

Video Generation



Give a model a bunch of noise, a description of some video you want, and Sora attempts to build that video.

Animating Images



You can feed Sora an image, followed by a bunch of noise.

Extending Videos

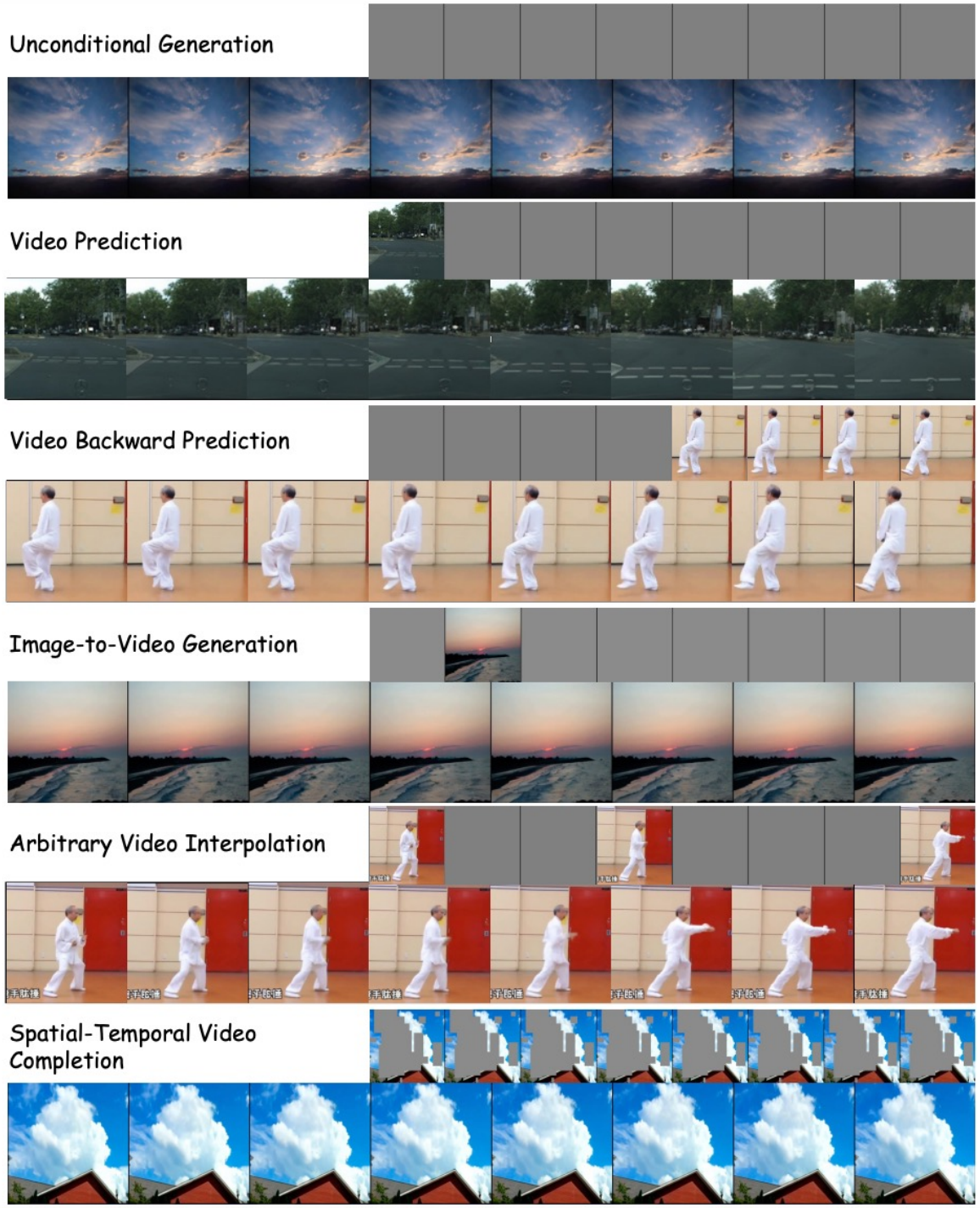
You can put a sequence of images into the input of the model, surrounded by noise

Connecting Videos

we stick noise between two videos, the model will naturally attempt to reconstruct a video which respects all surrounding video

Image Generation

You can just ask the model to build a video consisting of one frame, effectively turning the model into a typical image generation model.



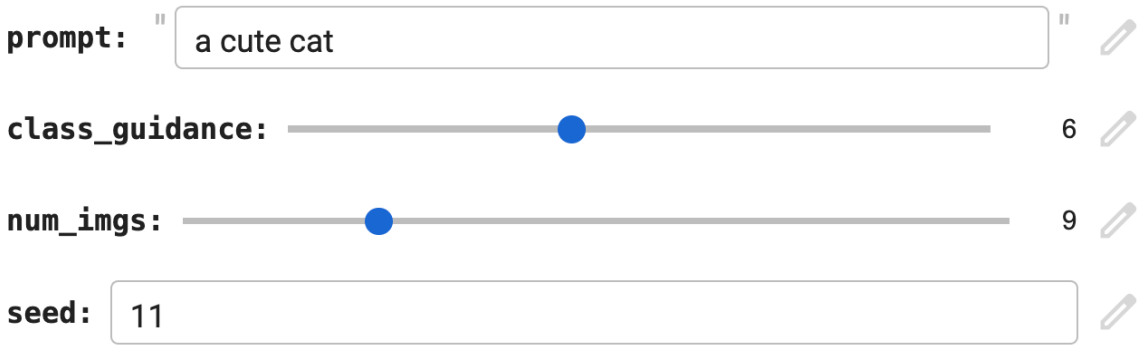
Qualitative results on unified video generation tasks

https://arxiv.org/pdf/2305.13311.pdf

Small Latent Diffusion Transformer

```
prompt = "a cute cat" #@param {type:"string"}
class_guidance = 6 #@param {type:"slider", min:0, max:15, step:0.5
n_iter = 20
###@param {type:"slider", min:0, max:60, step:1}
num_imgs = 9 #@param {type:"slider", min:1, max:36, step:1}
img_size = 32
seed = 11 #@param{}
nrow = int(np.sqrt(num_imgs))


cur_prompts = [prompt]*num_imgs
labels = encode_text(cur_prompts, clip_model)
out, out_latent = diffuser.generate(labels=labels,
                                    num_imgs=num_imgs,
                                    class_guidance=class_guidance,
                                    seed=seed,
                                    n_iter=n_iter,
                                    exponent=1,
                                    scale_factor=8,
                                    sharp_f=0,
```



Small Latent Diffusion Transformer

```
prompt = "a cute cat" #@param {type:"string"}
class_guidance = 6 #@param {type:"slider", min:0, max:15, step:0.5
n_iter = 20
###@param {type:"slider", min:0, max:60, step:1}
num_imgs = 9 #@param {type:"slider", min:1, max:36, step:1}
img_size = 32
seed = 11 #@param{}
nrow = int(np.sqrt(num_imgs))


cur_prompts = [prompt]*num_imgs
labels = encode_text(cur_prompts, clip_model)
out, out_latent = diffuser.generate(labels=labels,
                                    num_imgs=num_imgs,
                                    class_guidance=class_guidance,
                                    seed=seed,
                                    n_iter=n_iter,
                                    exponent=1,
                                    scale_factor=8,
                                    sharp_f=0,
```

prompt: " a cute cat "

class_guidance: 6

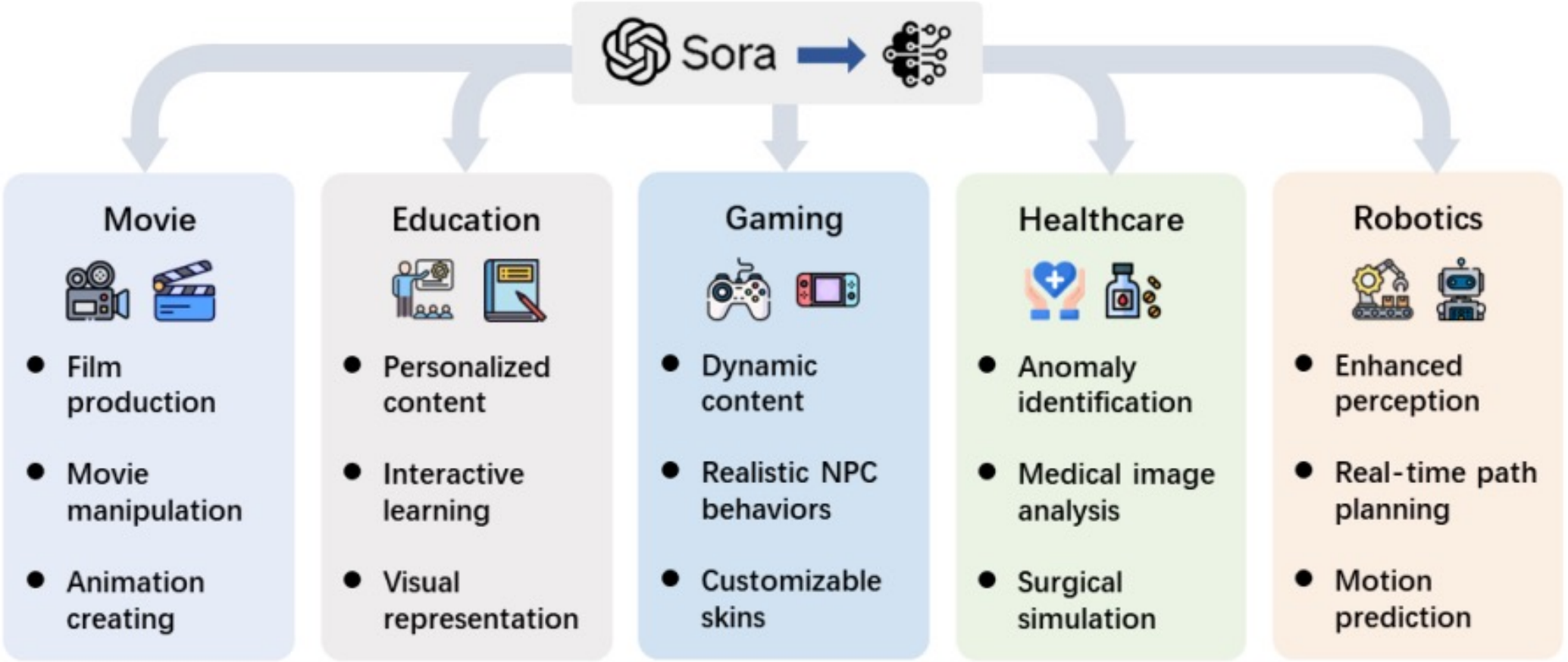
num_imgs: 9

seed: 11

Dicussion: Talk with Sora Developers

https://www.youtube.com/watch?v=eBvvJUYtnEA

Applications of Sora



Outline

- Objective
- Diffusion Probability Model
- Transformer and ChatGPT Architecture
- Vision Transformer Architecture
- Diffusion and Transformer
- OpenAI's Sora Architecture
- Summary

Summary

Sora: A Review on Background, Technology, Limitations, and Opportunities of Large Vision Models

Yixin Liu[1*] Kai Zhang[1*] Yuan Li[1*] Zhiling Yan[1*] Chujie Gao[1*]
Ruoxi Chen[1*] Zhengqing Yuan[1*] Yue Huang[1*] Hanchi Sun[1*]
Jianfeng Gao[2] Lifang He[1] Lichao Sun[1†]

[1]Lehigh University [2]Microsoft Research

Abstract

Sora is a text-to-video generative AI model, released by OpenAI in February 2024. The model is trained to generate videos of realistic or imaginative scenes from text instructions and show potential in simulating the physical world. Based on public technical reports and reverse engineering, this paper presents a comprehensive review of the model's background, related technologies, applications, remaining challenges, and future directions of text-to-video AI models. We first trace Sora's development and investigate the underlying technologies used to build this "world simulator". Then, we describe in detail the applications and potential impact of Sora in multiple industries ranging from film-making and education to marketing. We discuss the main challenges and limitations that need to be addressed to widely deploy Sora, such as ensuring safe and unbiased video generation. Lastly, we discuss the future development of Sora and video generation models in general, and how advancements in the field could enable new ways of human-AI interaction, boosting productivity and creativity of video generation.

Figure 1: Sora: A Breakthrough in AI-Powered Vision Generation.

1. Know how Do Diffusions Model Work
2. Know how Do Transformer and ChatGP work
3. Know how Does a Vision Transformer Work
4. Know how Does a Diffusion Transformer Work
5. Know how Does an OpenAI's Sora Work


Thank
you