Lecture 5

CME 296: Diffusion & Large Vision Models

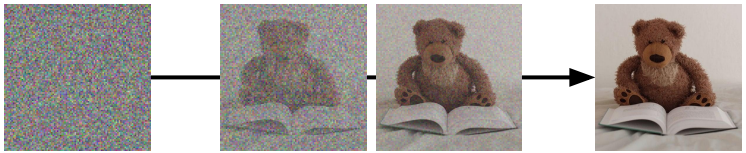
Afshine Amidi & Shervine Amidi

Stanford University

ICME

Recap of last episodes...

Lectures 1, 2, 3
Generation paradigms



Diffusion	$\epsilon_\theta(x_t, t)$	$\|\epsilon_\theta - \epsilon\|^2$
Score matching	$s_\theta(x_t, t)$	$\|s_\theta - s\|^2$
Flow matching	$u_\theta(x_t, t)$	$\|u_\theta - u\|^2$

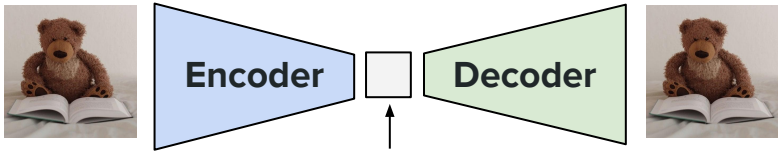
Recap of last episodes...

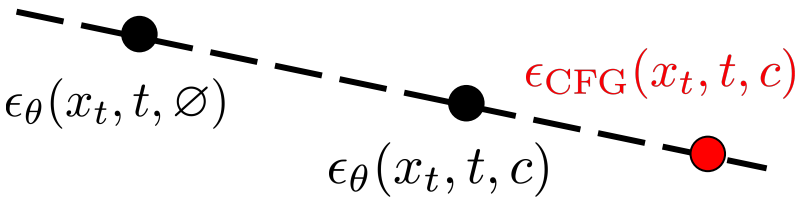
Lectures 1, 2, 3
Generation paradigms

Diffusion	$\epsilon_\theta(x_t, t)$	$\|\epsilon_\theta - \epsilon\|^2$
Score matching	$s_\theta(x_t, t)$	$\|s_\theta - s\|^2$
Flow matching	$u_\theta(x_t, t)$	$\|u_\theta - u\|^2$

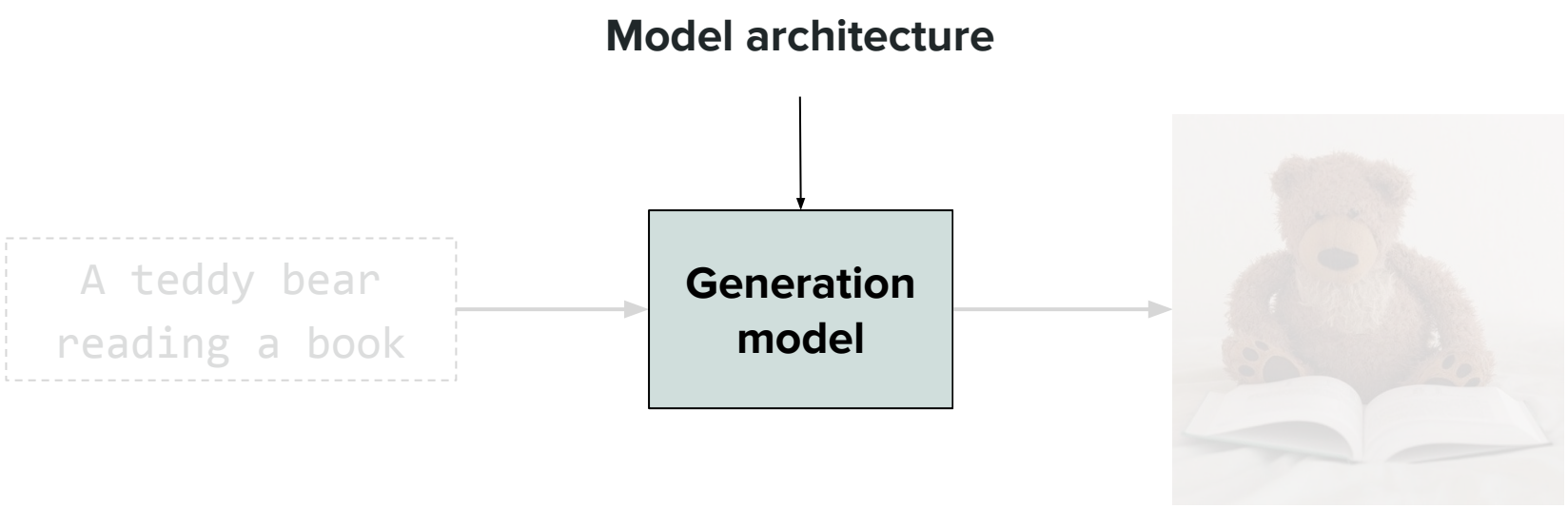
Lecture 4
Latent space, guidance

Variational **A**uto**E**ncoder (VAE)





Today's lecture



Today's lecture: **Image generation architectures**

Diffusion & Large Vision Models

Motivation

U-Net

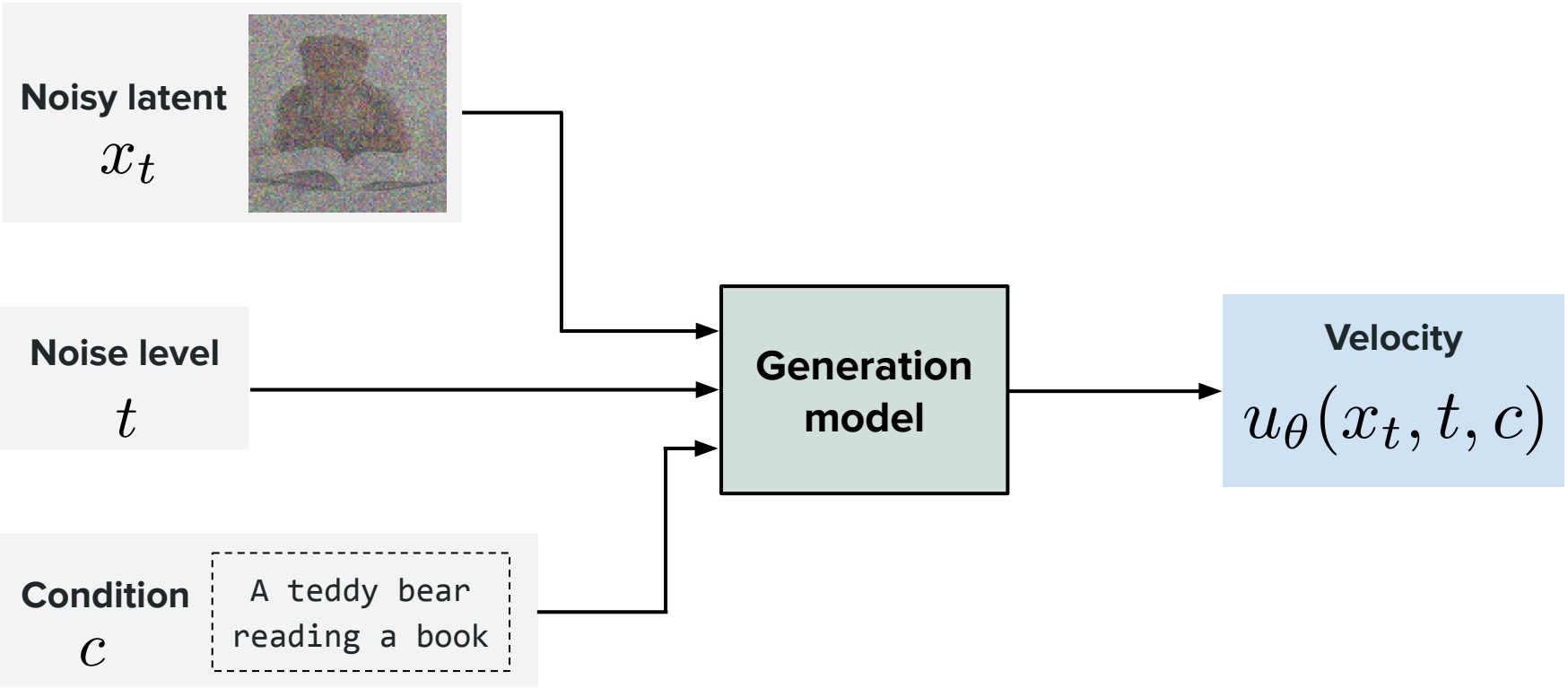
Diffusion Transformer

End-to-end example

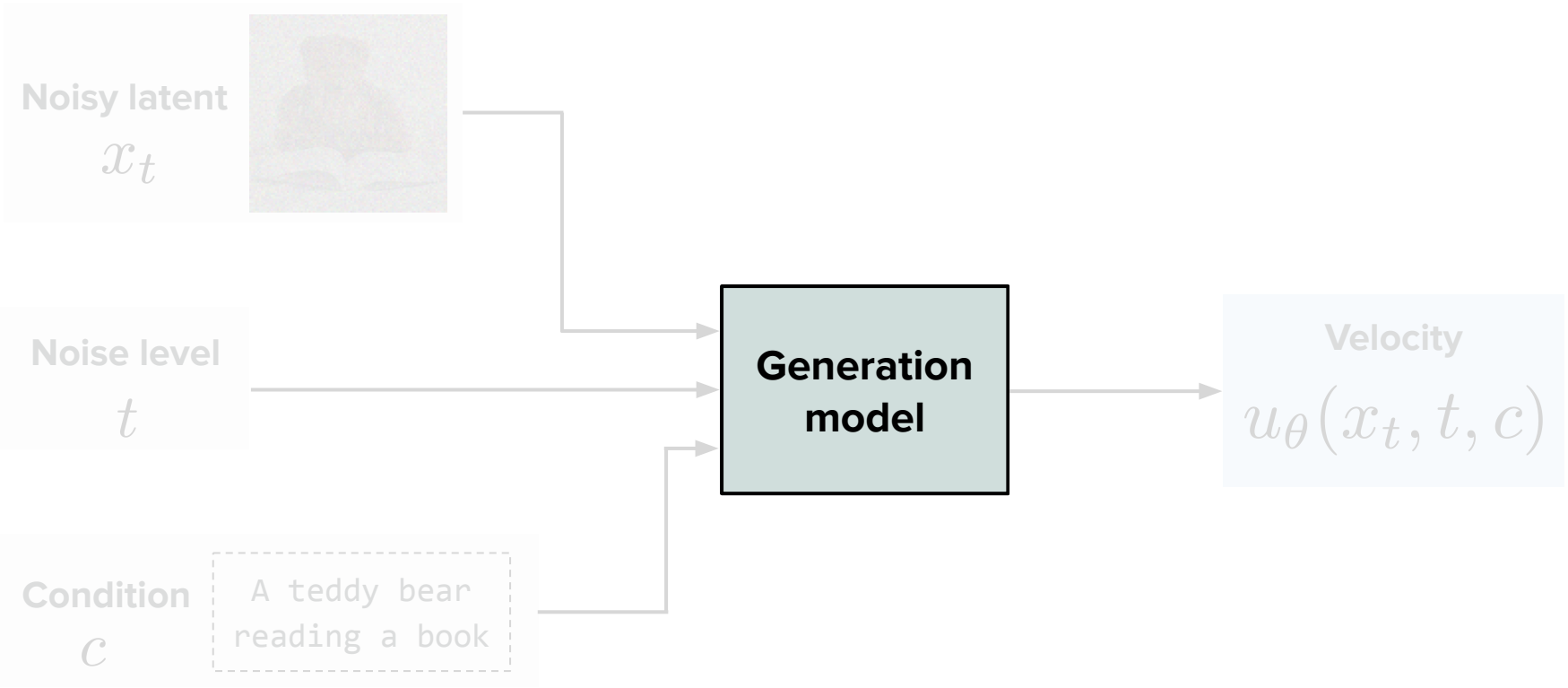
Multimodal DiT

Optimizations

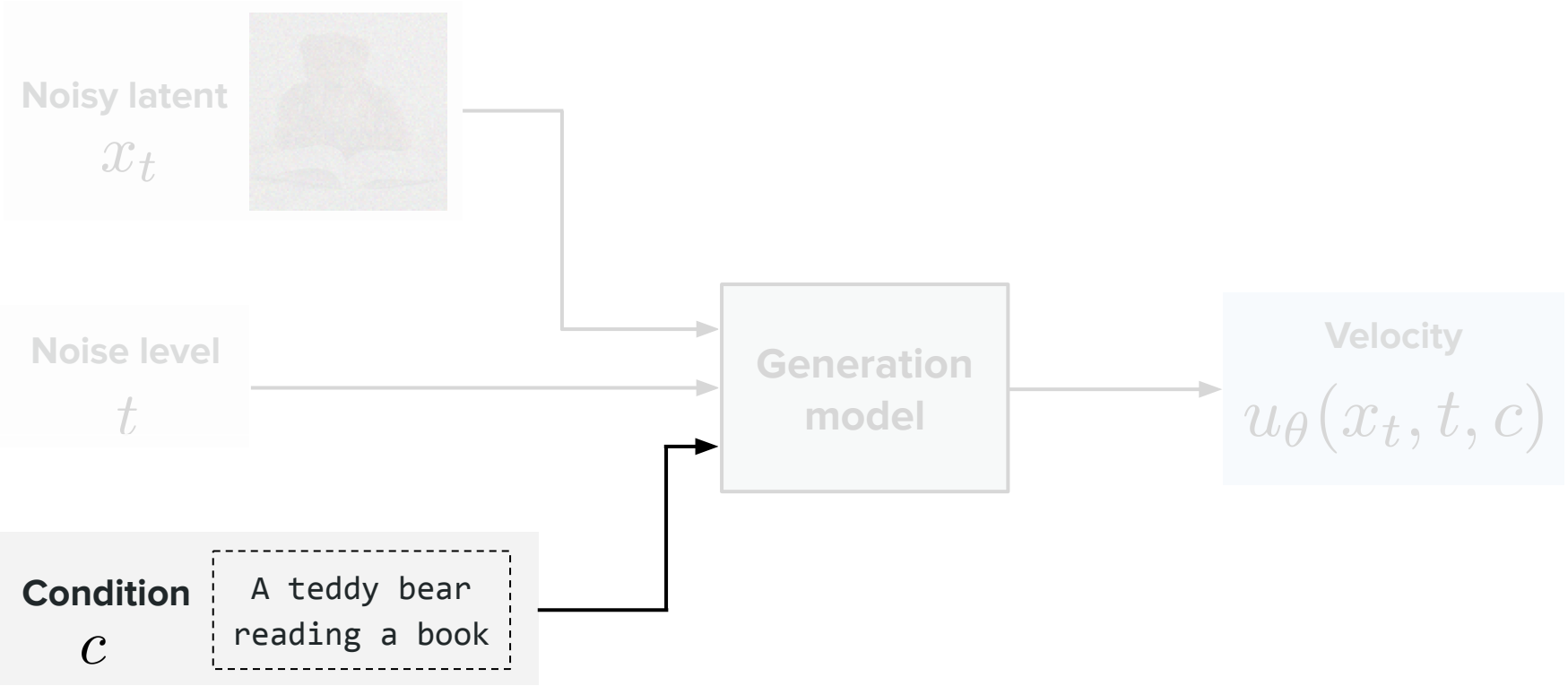
Problem statement



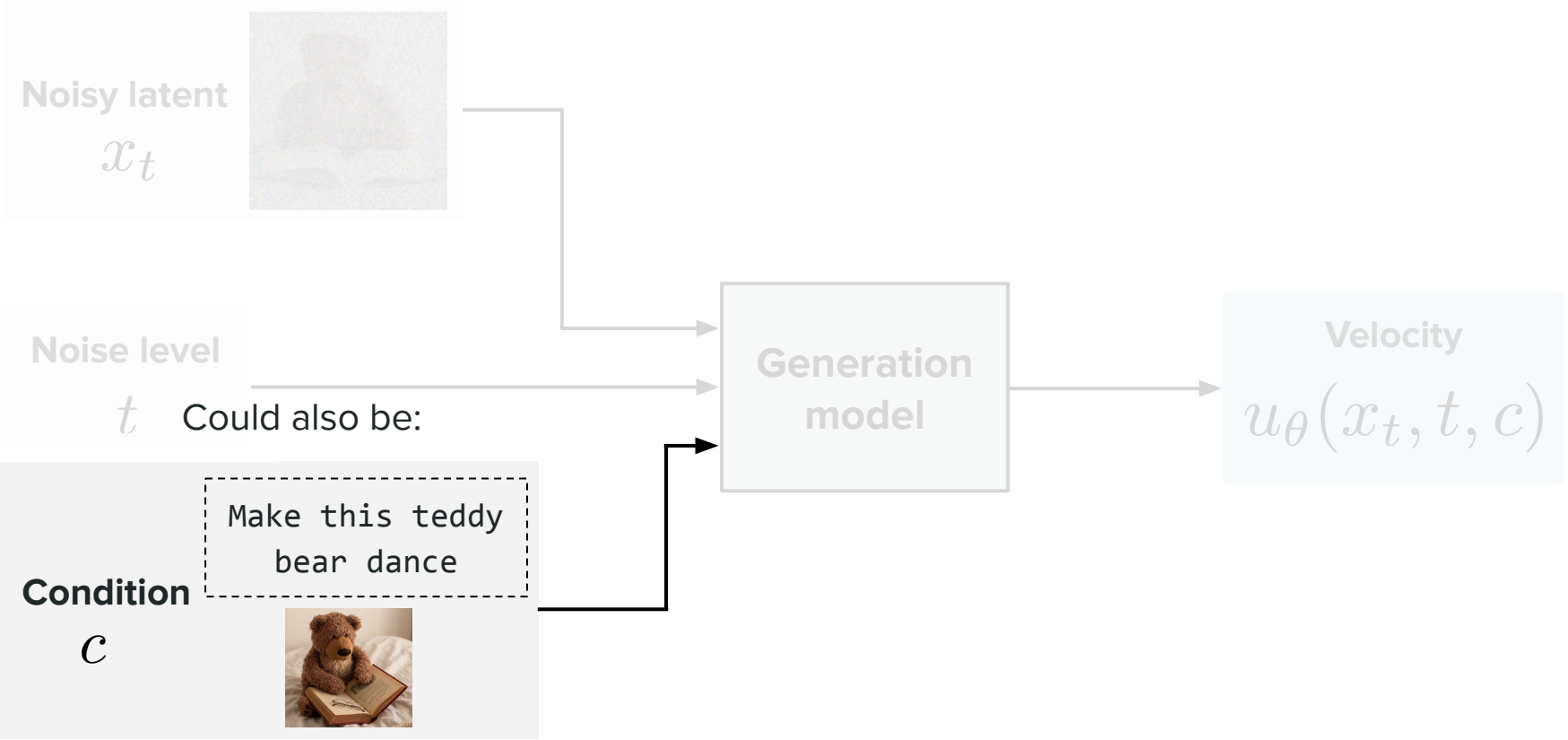
Objective of this lecture



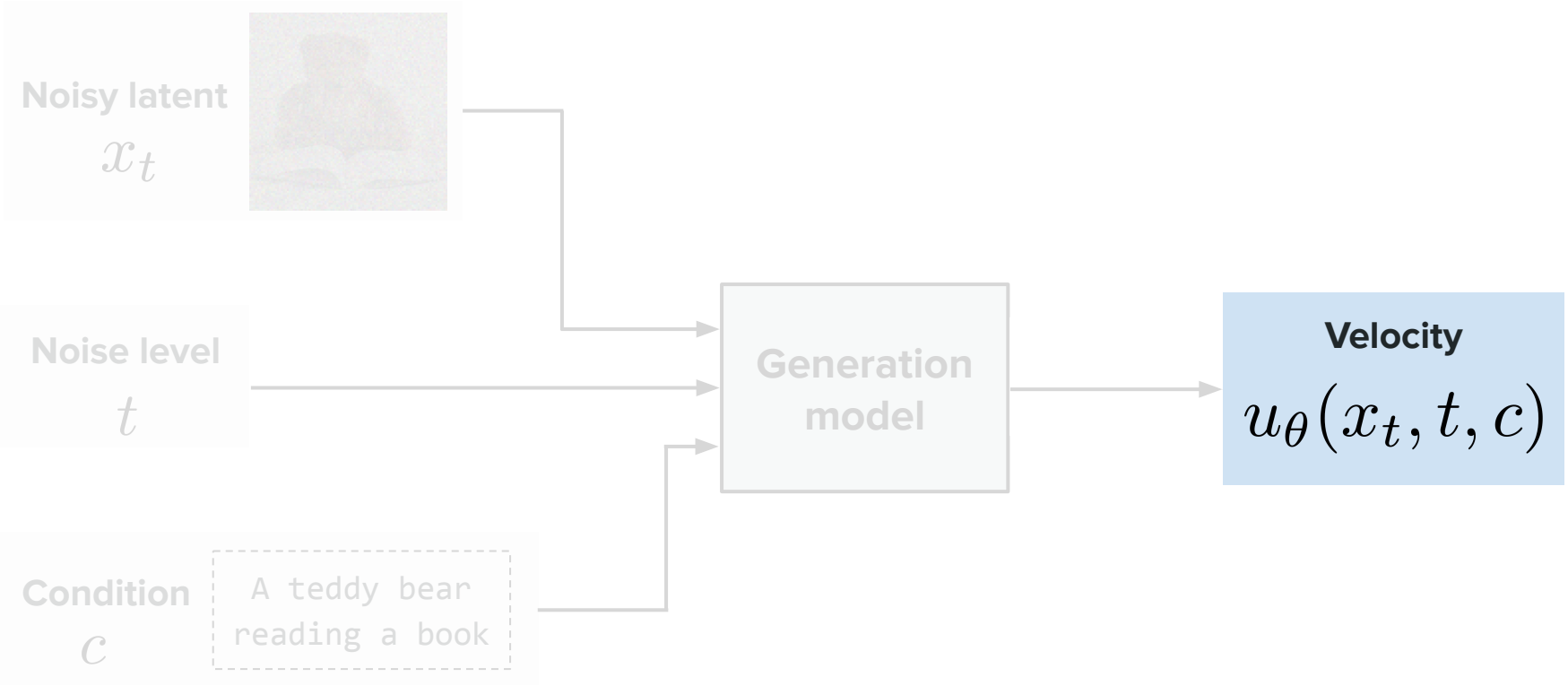
Side notes


Noisy latent
x_t
Noise level
t
Condition
c
A teddy bear reading a book
Generation model
Velocity
$u_\theta(x_t, t, c)$

Side notes


Noisy latent
x_t
Noise level
t
Could also be:
Condition
c
Make this teddy bear dance
Generation model
Velocity
$u_\theta(x_t, t, c)$

Side notes


Noisy latent
x_t
Noise level
t
Condition
c
A teddy bear reading a book
Generation model
Velocity
$u_\theta(x_t, t, c)$

Side notes

Noisy latent x_t

Noise level t

Condition c

A teddy bear reading a book

Generation model

Velocity

$$u_\theta(x_t, t, c)$$

Could also be:

Noise $\epsilon_\theta(x_t, t, c)$

Score $s_\theta(x_t, t, c)$

Goal

Objective. Choose architecture for image generation with criteria below:

- Understand **global structure**
- Preserve **local details**
- Responsive to **external signals** (timestep, condition)
- **Scalable** computationally

Diffusion & Large Vision Models

Motivation

U-Net

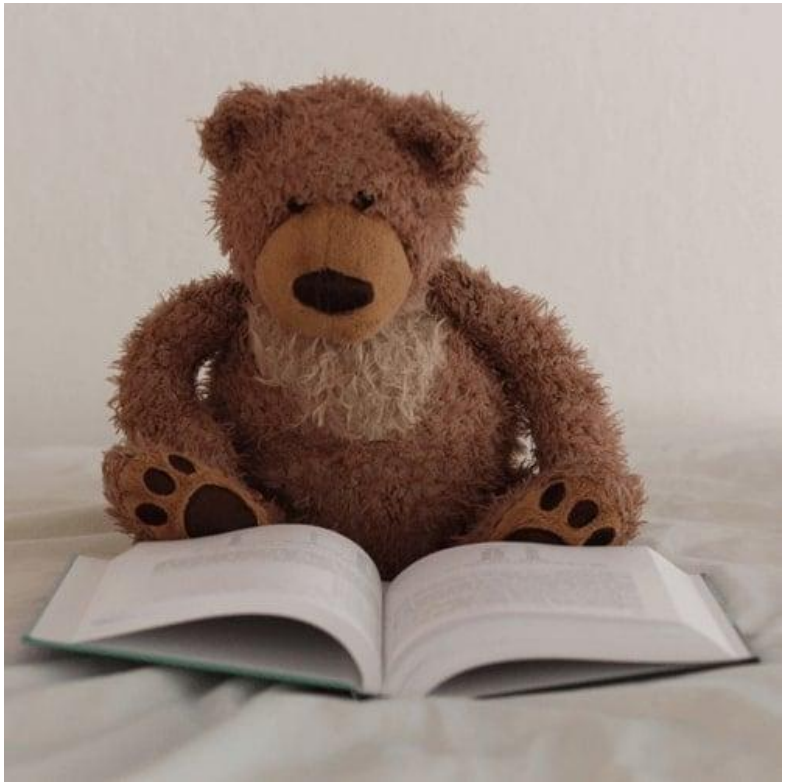
Diffusion Transformer

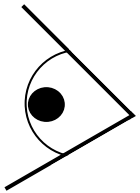
End-to-end example

Multimodal DiT

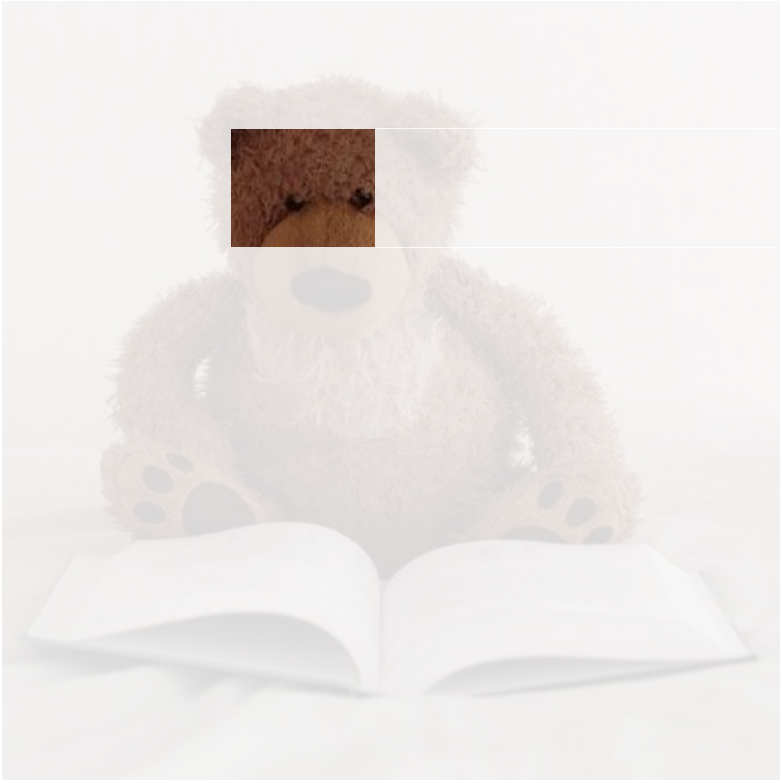
Optimizations

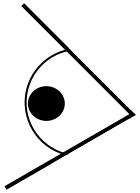
Intuition



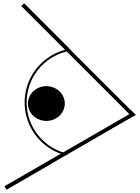


Intuition



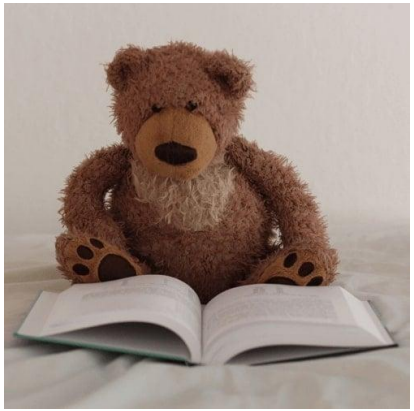


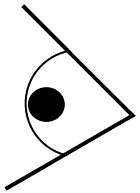
Intuition



Intuition behind convolution-based models

Idea. Mimic human vision via "inductive" bias baked into the model

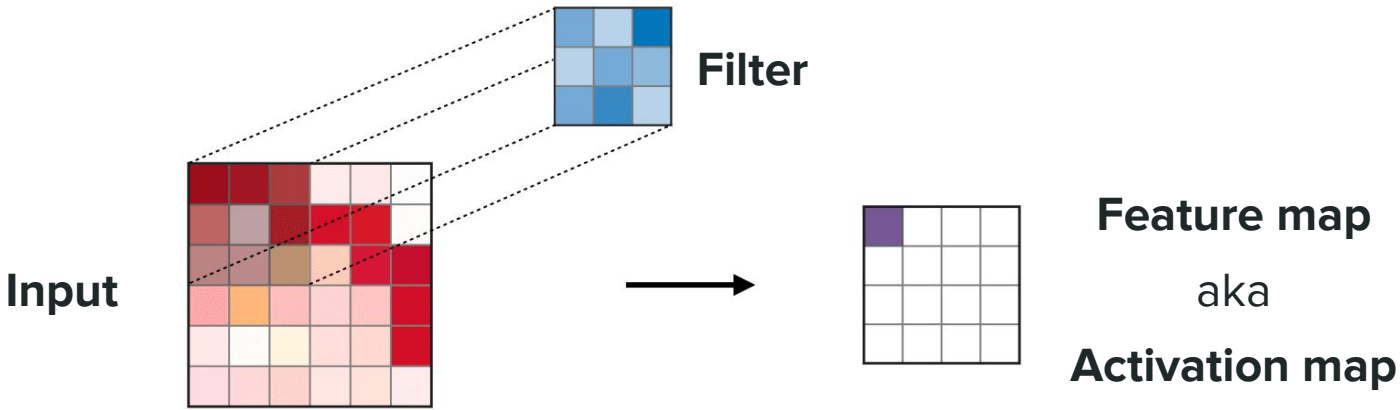




Convolution operation with filters

Convolution filter. Feature detector

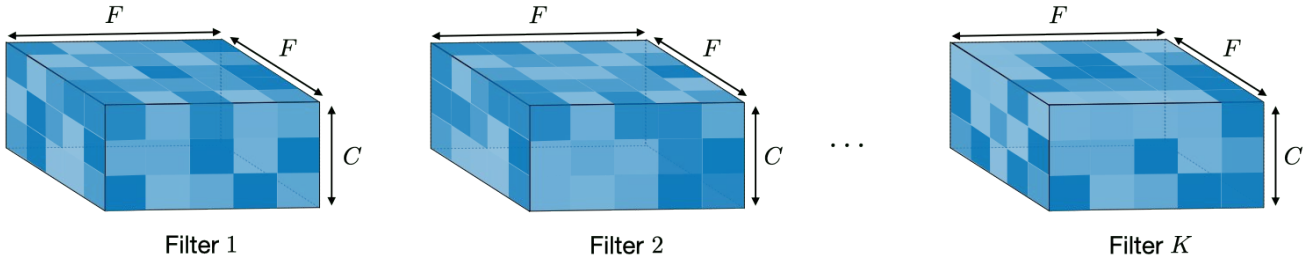
e.g. edges, corners, textures, patterns, shapes



VIP cheatsheets on Convolutional Neural Networks, Amidi, 2018.

Convolution operation with filters

Filter. Performs convolution operations on the input.



Convolution operation with filters

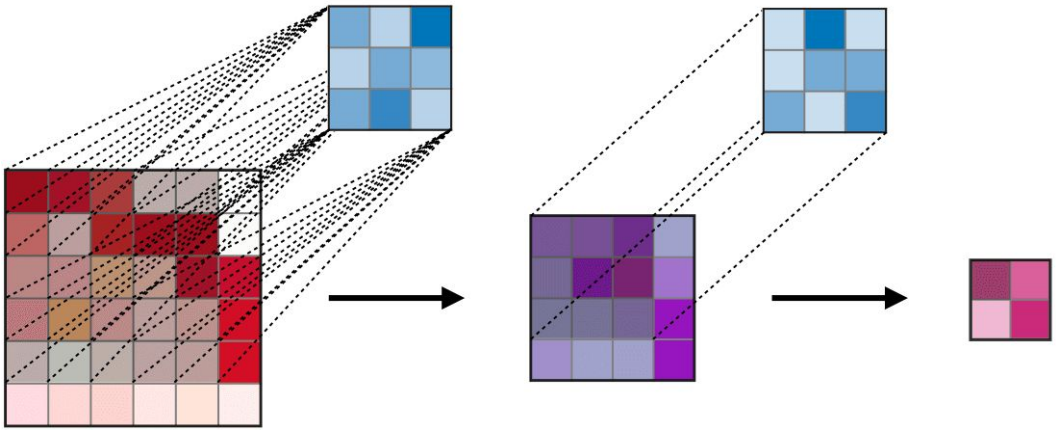
Filter. Performs convolution operations on the input.

Stride. Amount by which the filter moves.

VIP cheatsheets on Convolutional Neural Networks, Amidi, 2018.

Receptive field

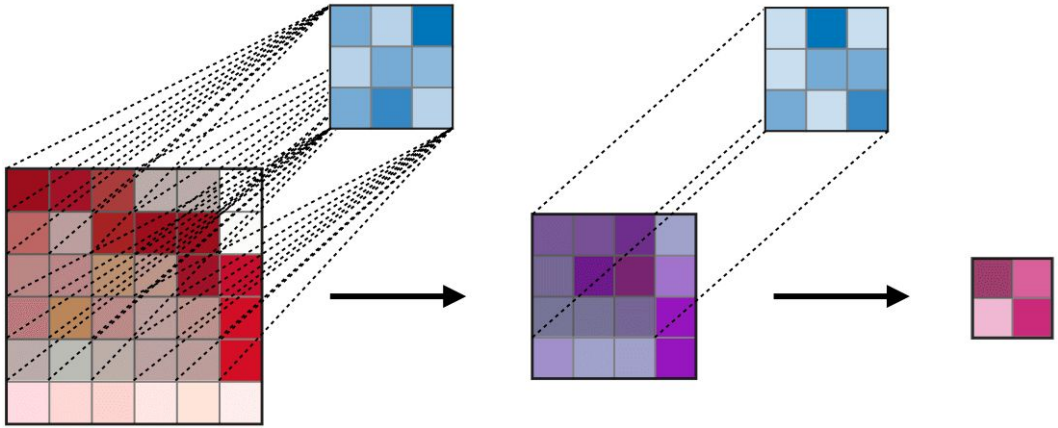
Receptive field. Area that the activation map can "see"



VIP cheatsheets on Convolutional Neural Networks, Amidi, 2018.

Receptive field

Receptive field. Area that the activation map can "see"



A pixel can see an area of $R_k \times R_k$ where $R_k = 1 + \sum_{j=1}^{k}(F_j - 1)\prod_{i=0}^{j-1} S_i$

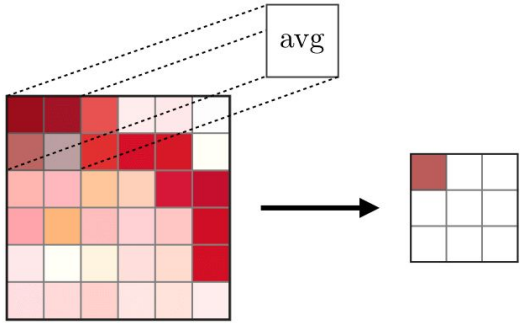
Filter size

Stride

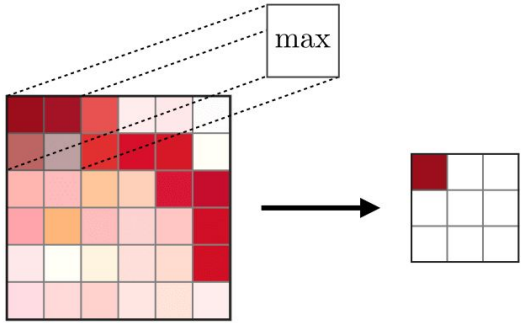
VIP cheatsheets on Convolutional Neural Networks, Amidi, 2018.

Downsampling with pooling operation

Pooling. Downsampling operation, per channel. Typically after convolution.



Average pooling

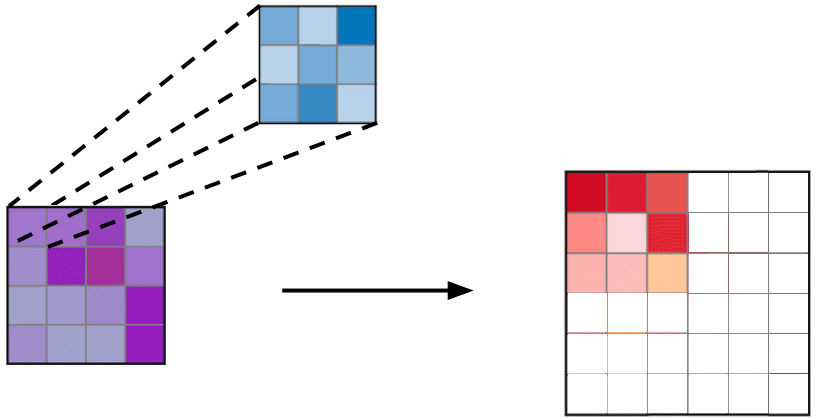


Max pooling

Upsampling with transpose convolution

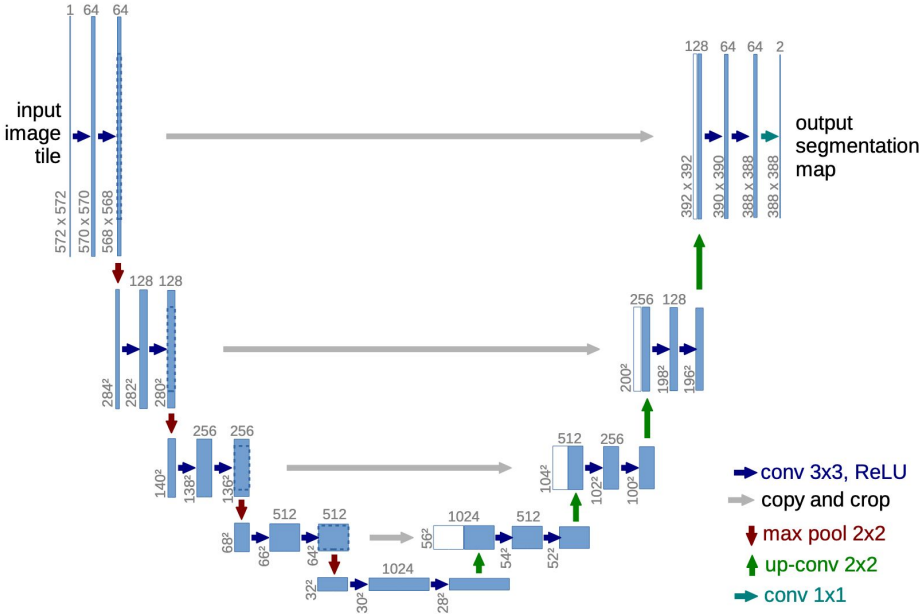
Transpose convolution. Upsampling operation, equivalent of "broadcasting"

Input feature map



Upsampled feature map

Proposal of a convolution-based architecture

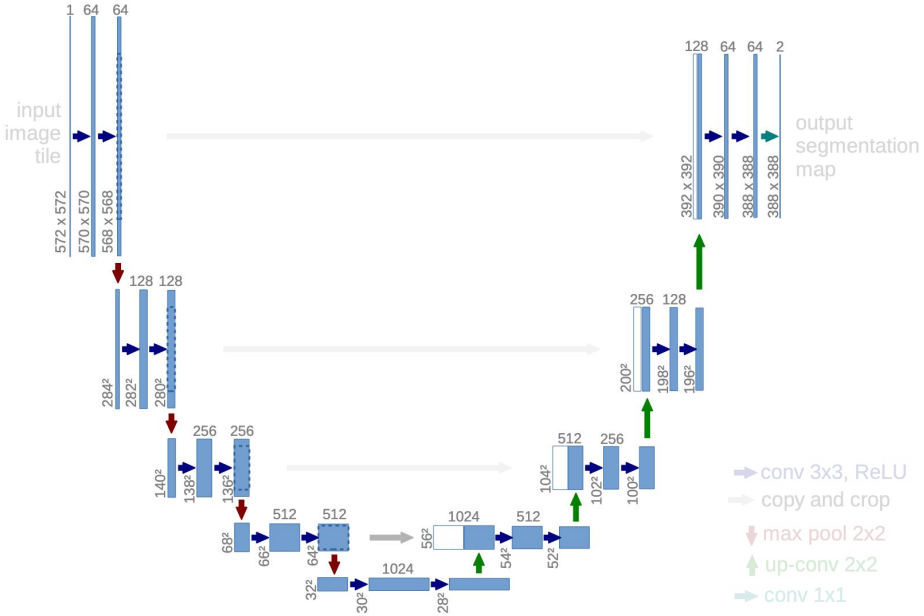


Overview

- Captures both **local** and **global** features
- **Inductive** bias that is relevant to reverse diffusion process
- Same dimensions for input / output

U-Net: Convolutional Networks for Biomedical Image Segmentation, Ronneberger et al., 2015.

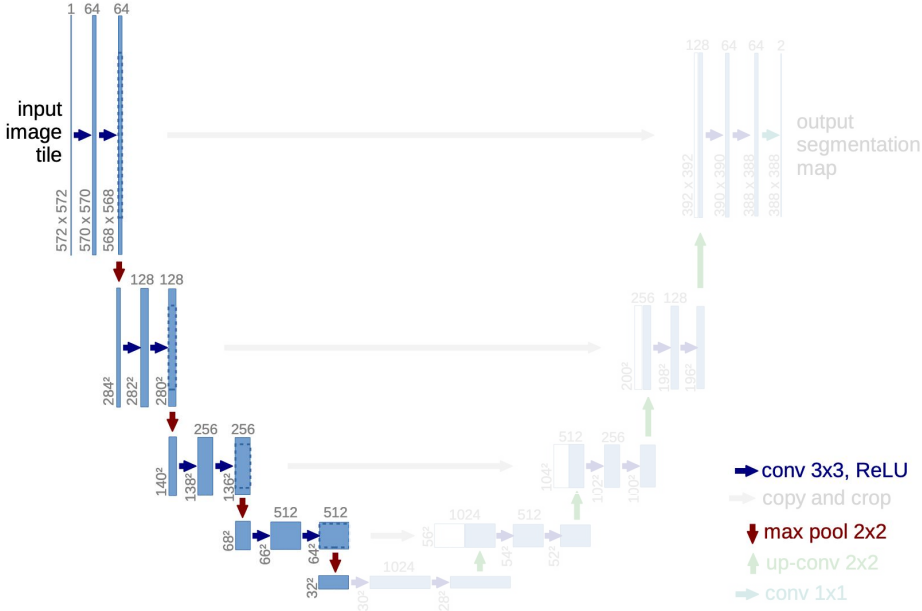
U-Net: convolution-based architecture



-Net

Figure adapted from U-Net: Convolutional Networks for Biomedical Image Segmentation, Ronneberger et al., 2015.

U-Net: downsampling phase

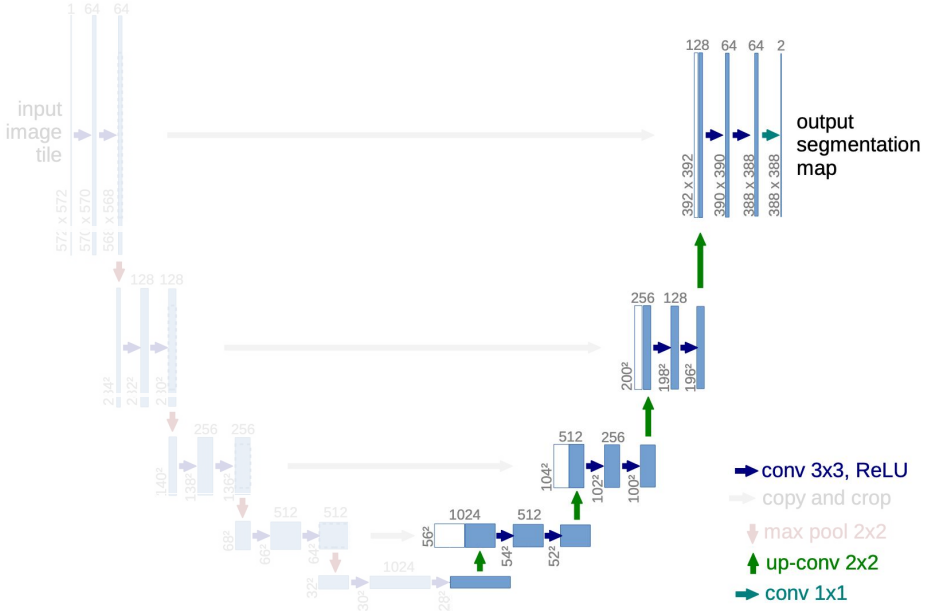


Downsampling

- Mix of **convolution** and max **pooling**
- Number of features is **halved** at each step
- **Extraction** of relevant features
- Similar to an "**Encoder**"

Figure adapted from U-Net: Convolutional Networks for Biomedical Image Segmentation, Ronneberger et al., 2015.

U-Net: upsampling phase

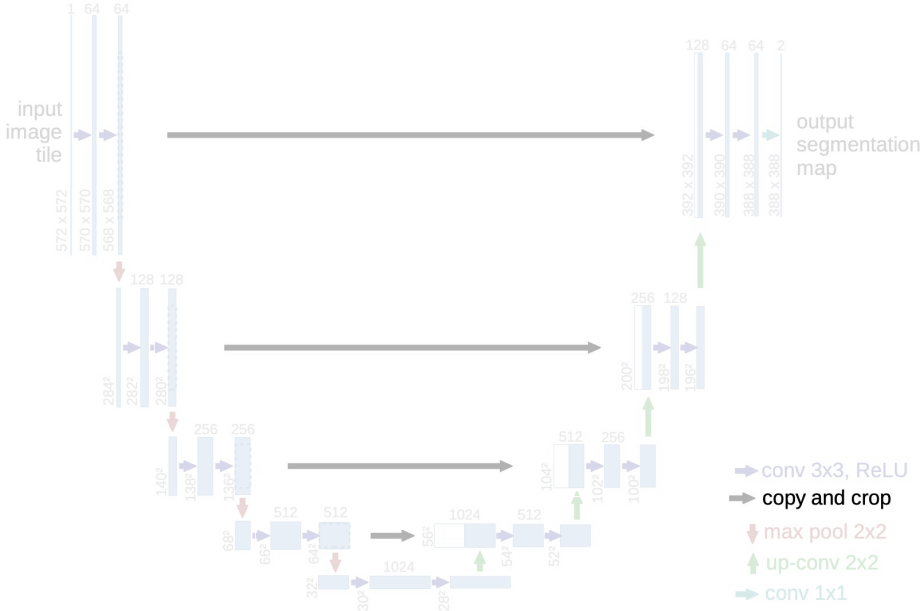


Upsampling

- Mix of **up-convolution** and **convolutions**
- **Number of features** is **doubled** at each step
- **1x1 convolution** at the last step before obtaining output
- Similar to a "**Decoder**"

Figure adapted from U-Net: Convolutional Networks for Biomedical Image Segmentation, Ronneberger et al., 2015.

U-Net: residual connections



"Copy and crop" connections

- Avoids losing **local** information
- "**Highway**" of information

Figure adapted from U-Net: Convolutional Networks for Biomedical Image Segmentation, Ronneberger et al., 2015.

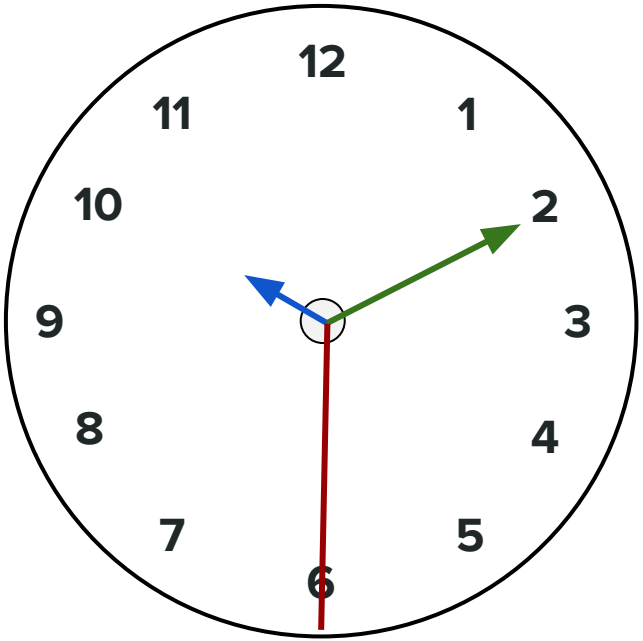
U-Net: adding condition information

t, c

How can we add time and class label information?

Figure adapted from U-Net: Convolutional Networks for Biomedical Image Segmentation, Ronneberger et al., 2015.

Representation of timestep



Hour (slow)

Minute (fast)

Second (very fast)

Representation of timestep

Hour (slow)

Minute (fast)

Second (very fast)

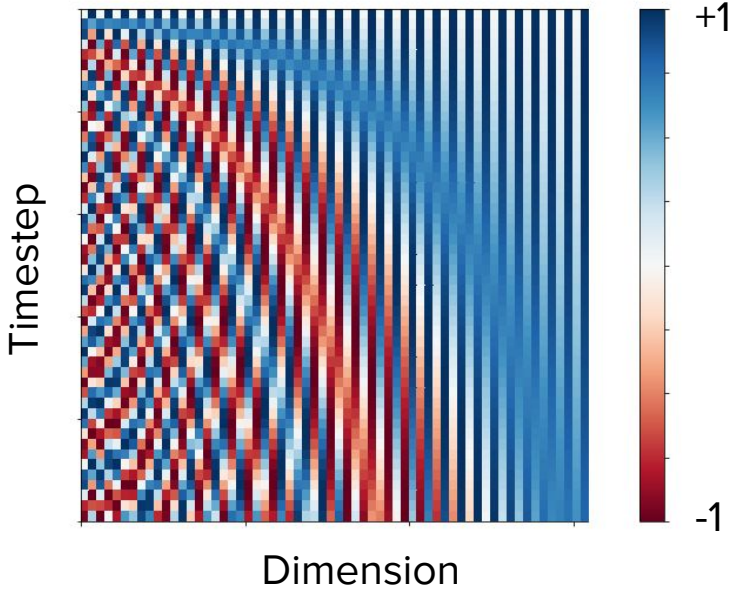
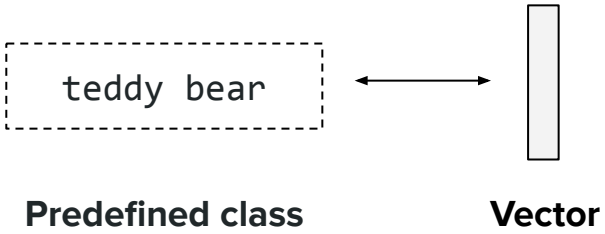


Figure adapted from Transformer Architecture: The Positional Encoding, Kazemnejad, 2019.

Representation of class label

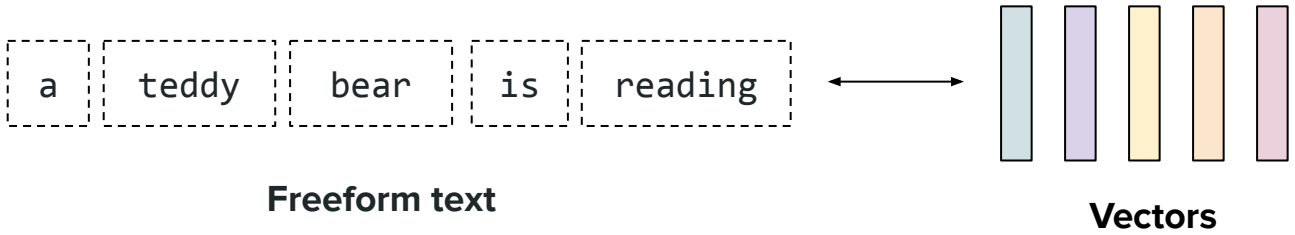
- Embedding corresponding to predefined class



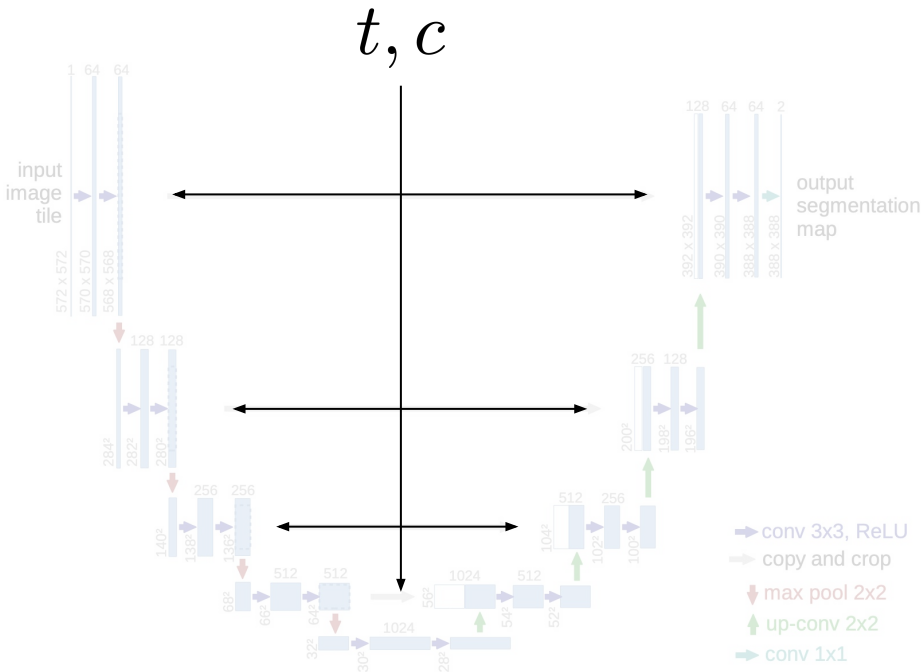
Representation of class label

- Embedding corresponding to predefined class

- Rich embeddings from a pretrained LLM



U-Net: adding condition information

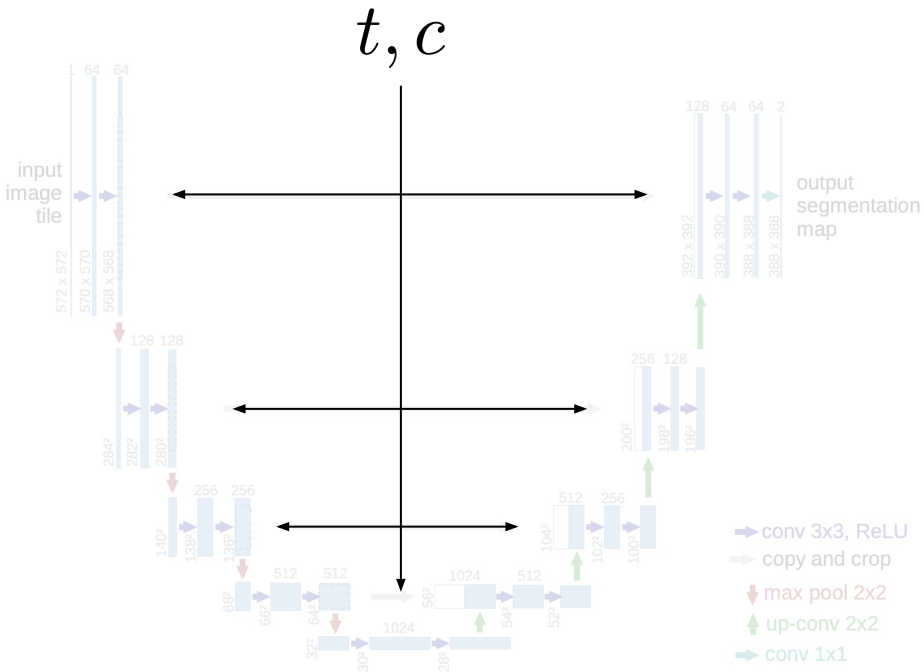


Method 1

Added to the feature map

Figure adapted from U-Net: Convolutional Networks for Biomedical Image Segmentation, Ronneberger et al., 2015.

U-Net: adding condition information



Method 1

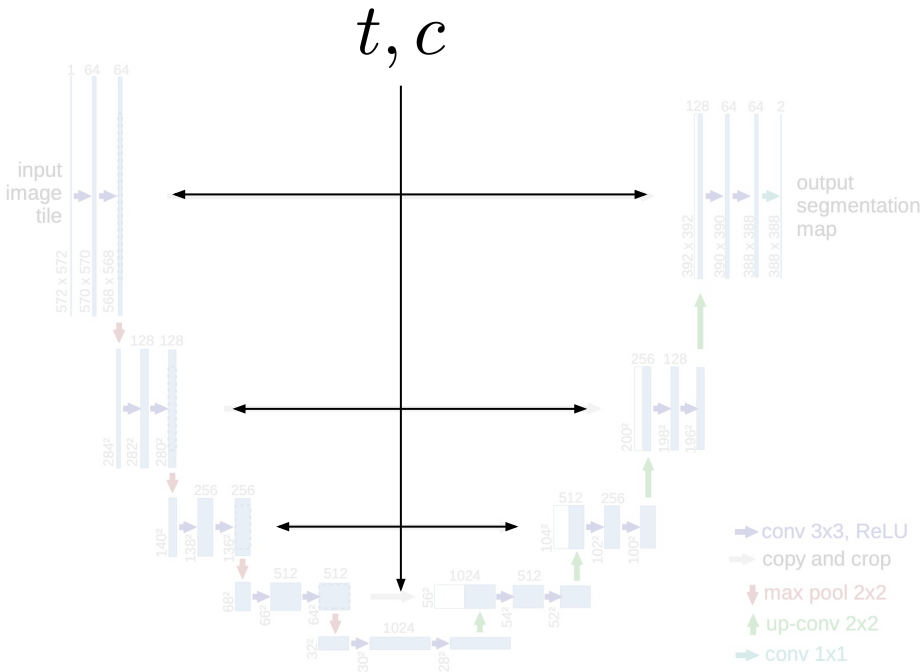
Added to the feature map

Method 2

Modulate feature map via scaling / shifting

Figure adapted from U-Net: Convolutional Networks for Biomedical Image Segmentation, Ronneberger et al., 2015.

U-Net: adding condition information



Method 1

Added to the feature map

Method 2

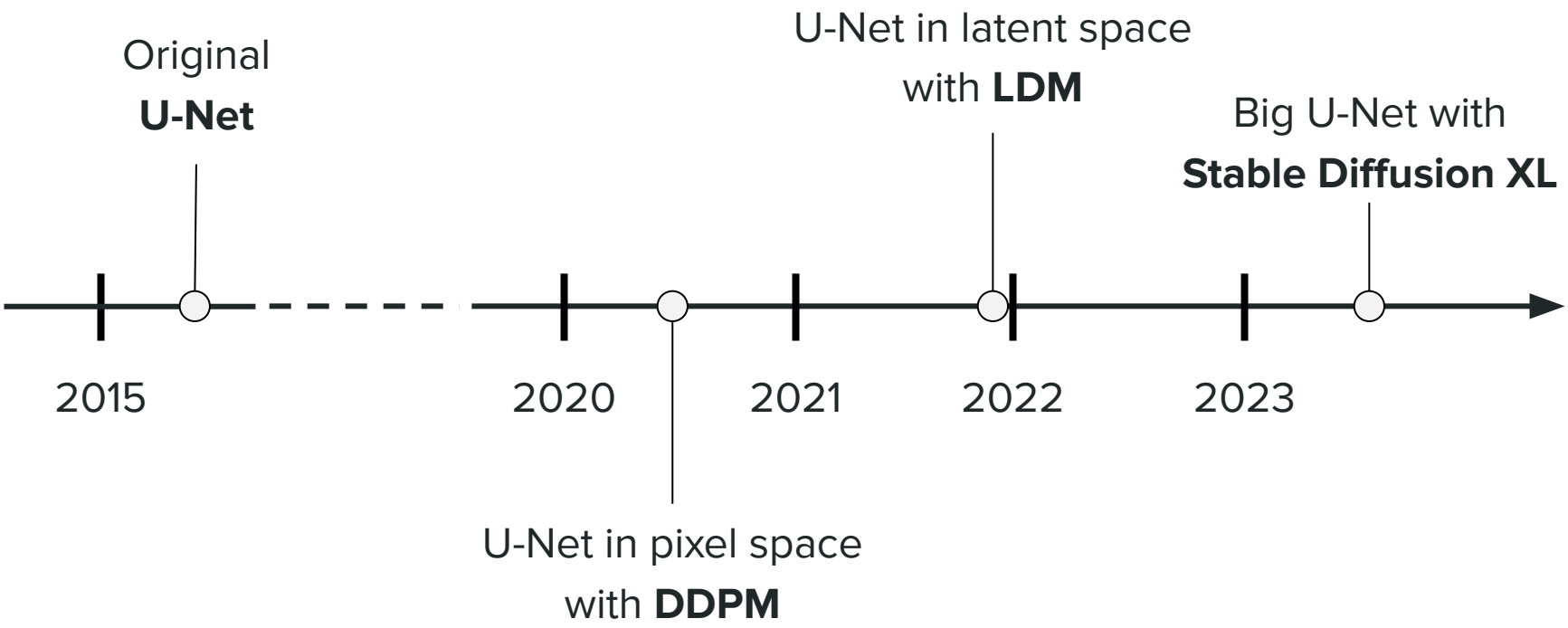
Modulate feature map via scaling / shifting

Method 3

Cross-attention of condition with feature map

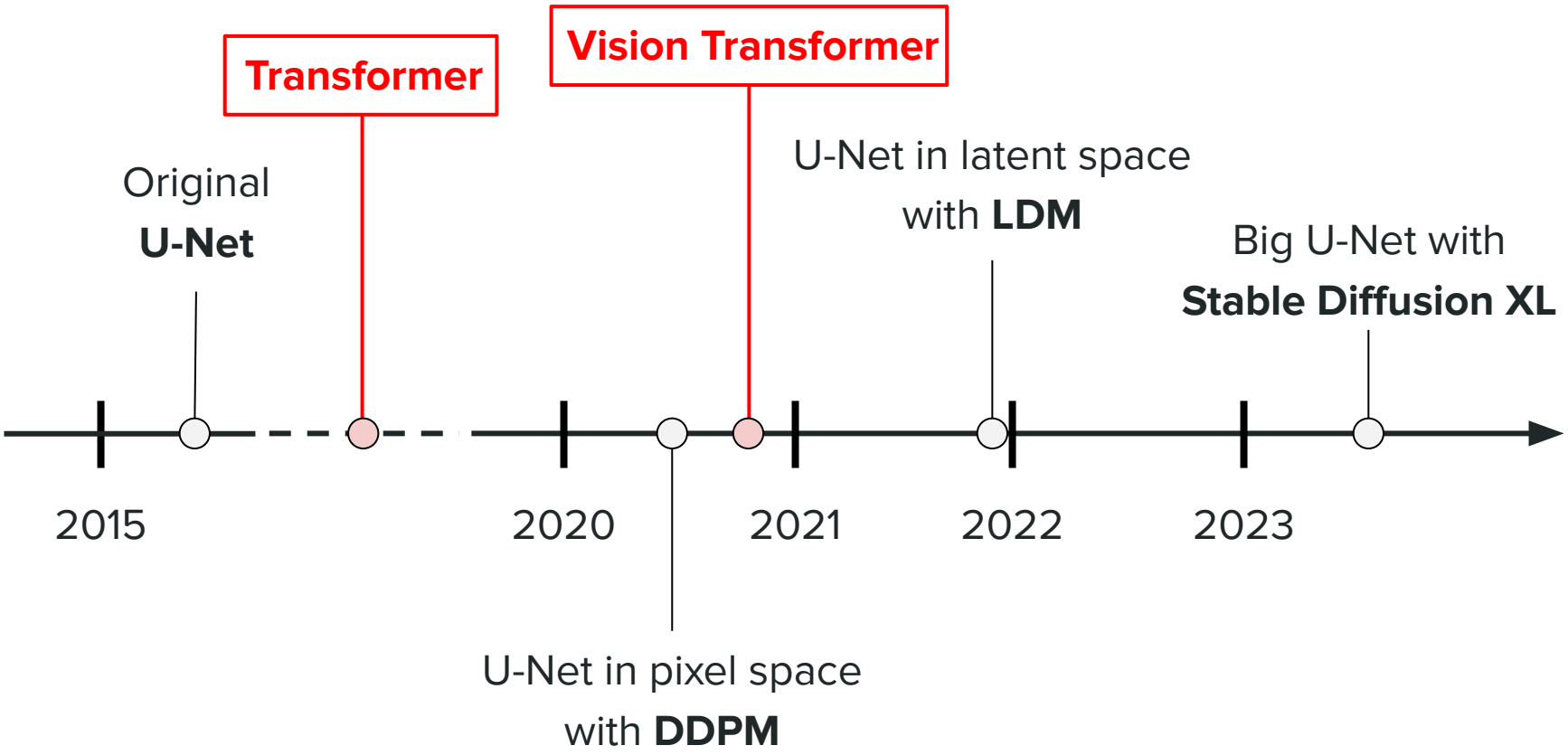
Figure adapted from U-Net: Convolutional Networks for Biomedical Image Segmentation, Ronneberger et al., 2015.

Timeline of U-Net-based models



Note: This timeline is not meant to be exhaustive in any way.

Timeline of U-Net-based models



Note: This timeline is not meant to be exhaustive in any way.

Diffusion & Large Vision Models

Motivation

U-Net

Diffusion Transformer

End-to-end example

Multimodal DiT

Optimizations

Motivating example

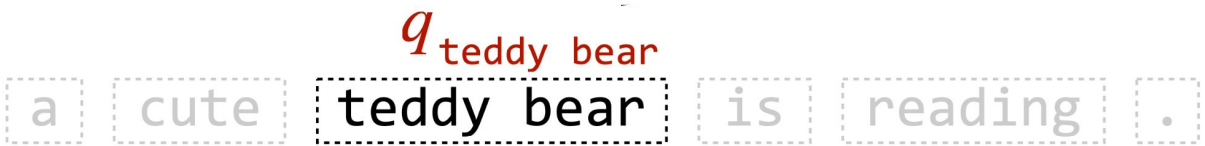
Need to preserve **local details**

across **long distances**

Image generated with ChatGPT, May 5th, 2026.

Attention mechanism

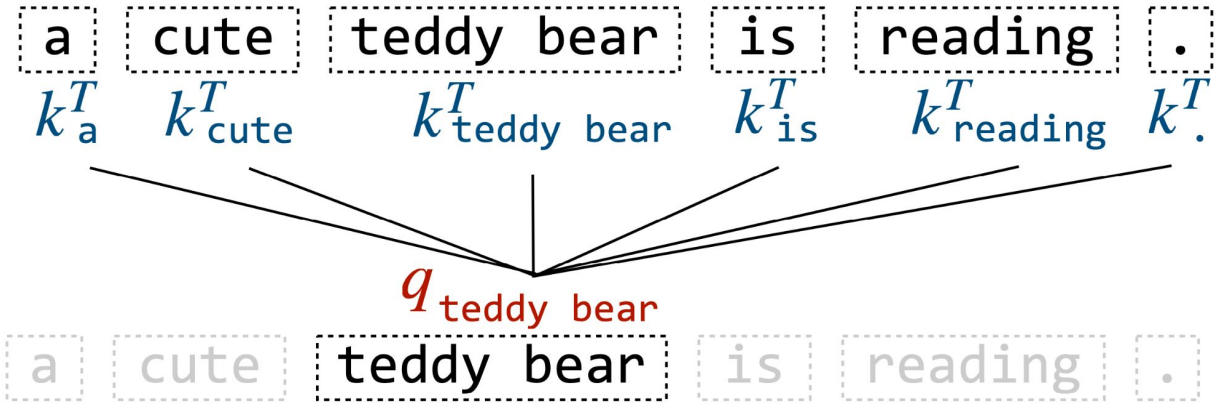
Idea. Remove inductive bias



Attention Is All You Need, Vaswani et al., 2017. Figure from *Super Study Guide: Transformers & LLMs*, Amidi, 2024.

Attention mechanism

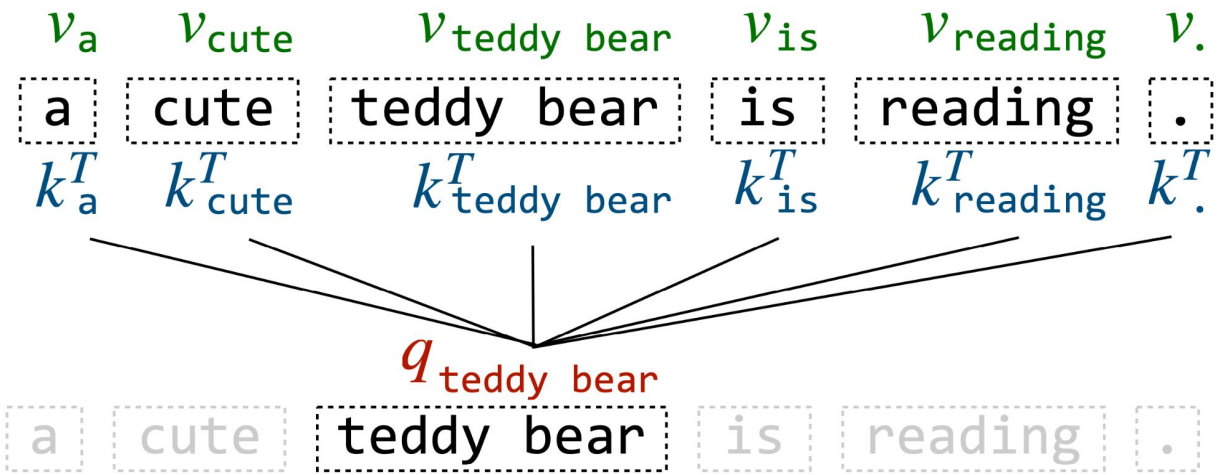
Idea. Remove inductive bias



Attention Is All You Need, Vaswani et al., 2017. Figure from *Super Study Guide: Transformers & LLMs*, Amidi, 2024.

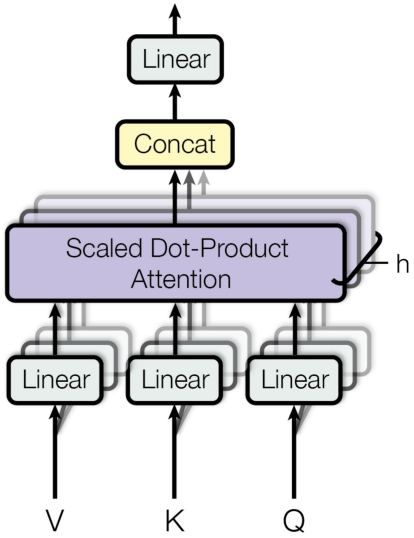
Attention mechanism

Idea. Remove inductive bias



Attention Is All You Need, Vaswani et al., 2017. Figure from *Super Study Guide: Transformers & LLMs*, Amidi, 2024.

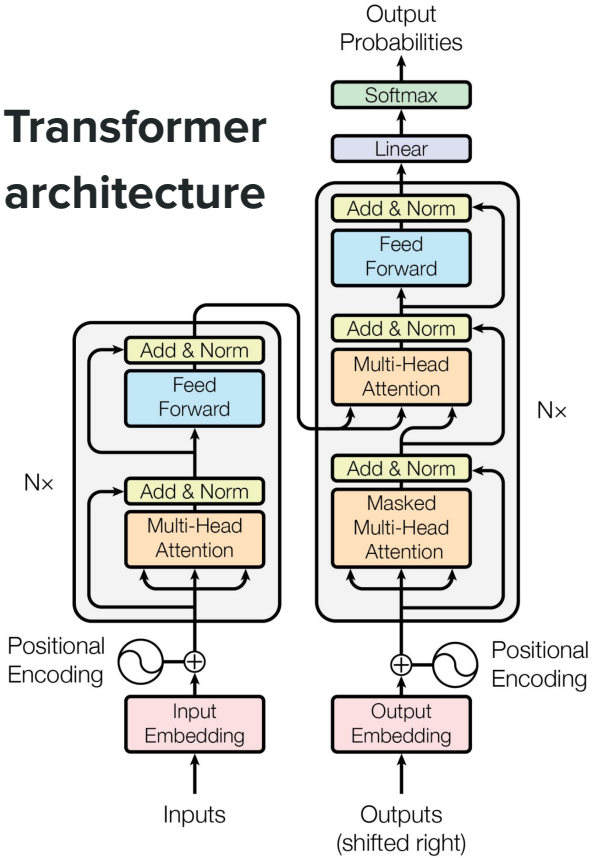
Transformer



Multi-head attention layer

$$\text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

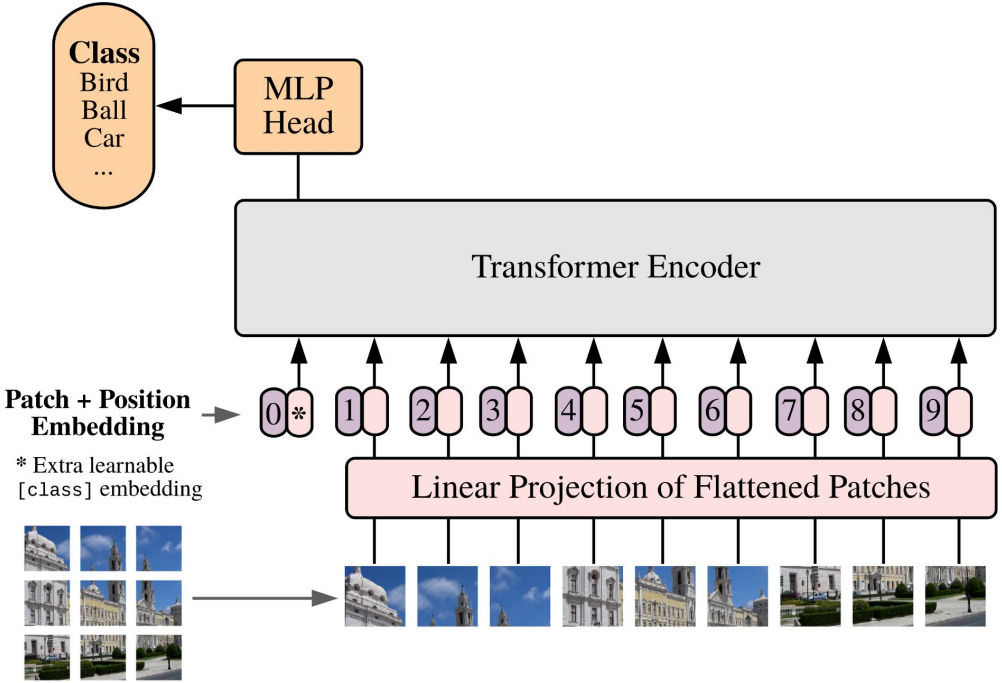
Transformer architecture

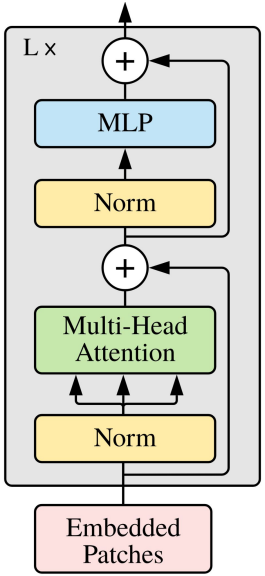


Figures from Attention Is All You Need, Vaswani et al., 2017. Detailed explanations in Stanford's CME 295: Transformers and LLMs.

Transformer for vision understanding...

ViT = **Vi**sion **T**ransformer

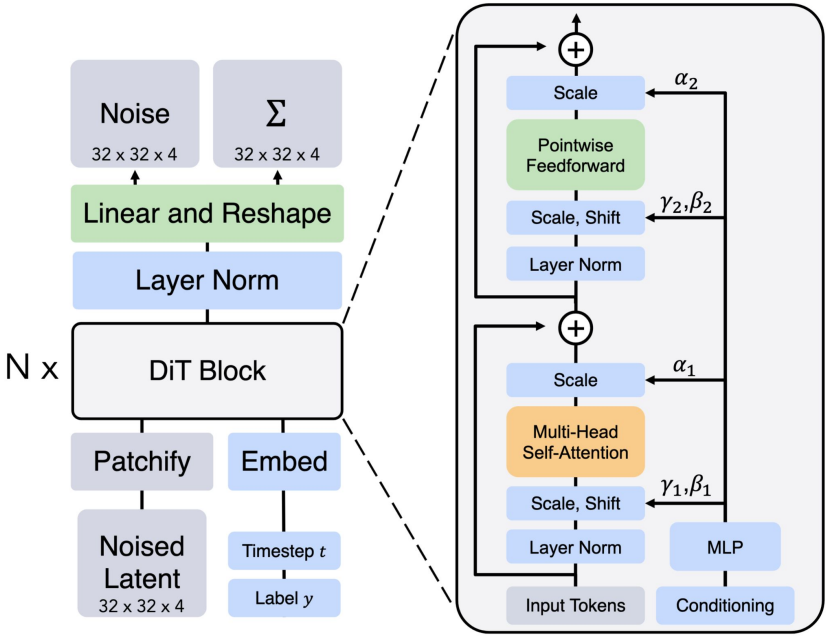




An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale, Dosovitskiy et al., 2020.

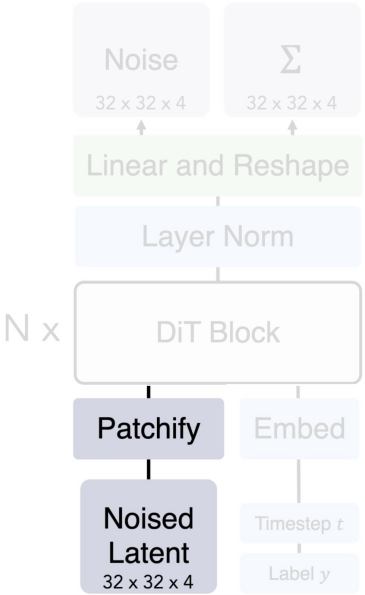
...and Transformer for vision generation!

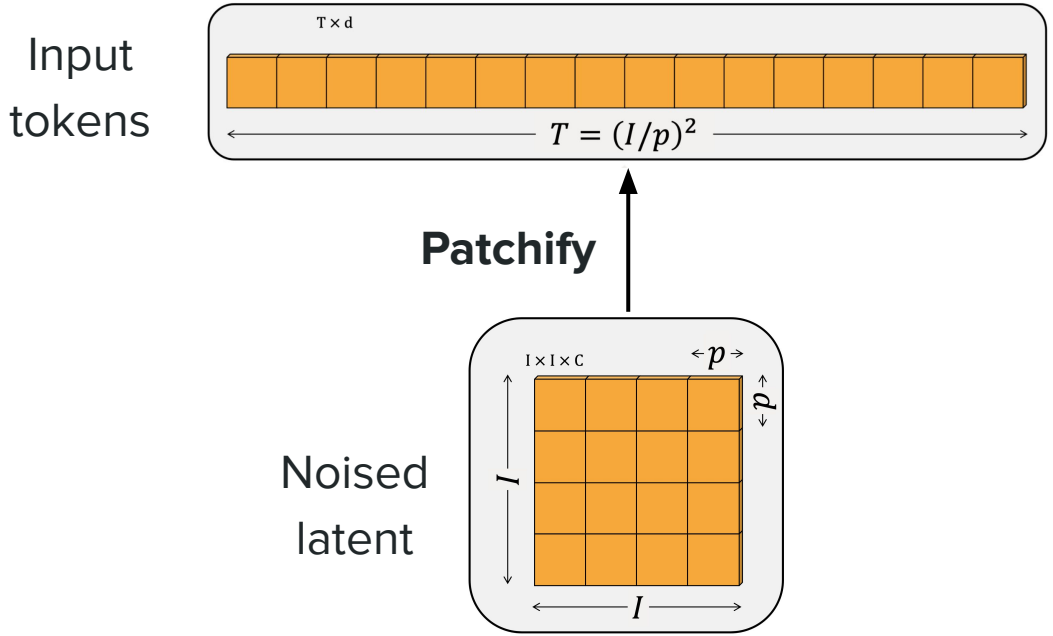
DiT = **Di**ffusion **T**ransformer



Scalable Diffusion Models with Transformers, Peebles et al., 2022.

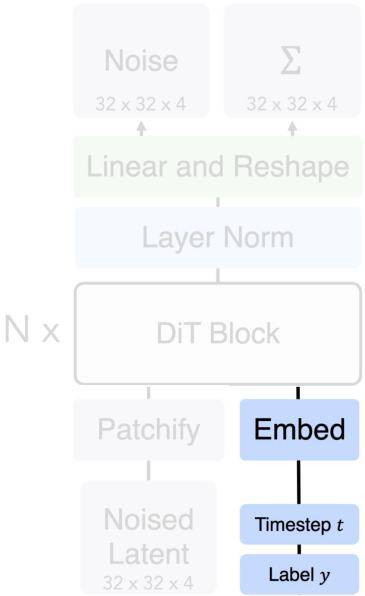
DiT: tokenization of the input via patches



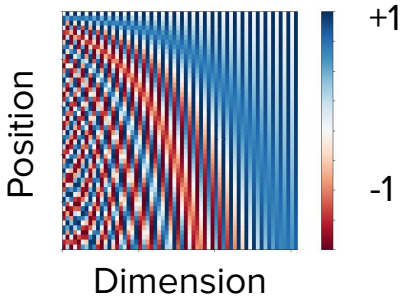


Figures adapted from Scalable Diffusion Models with Transformers, Peebles et al., 2022.

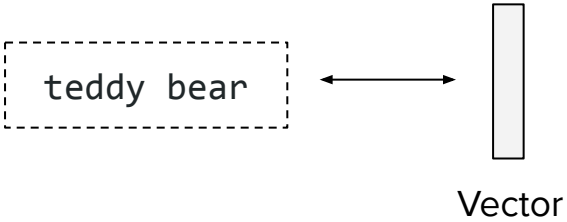
DiT: embedding of conditions



Timestep embedding

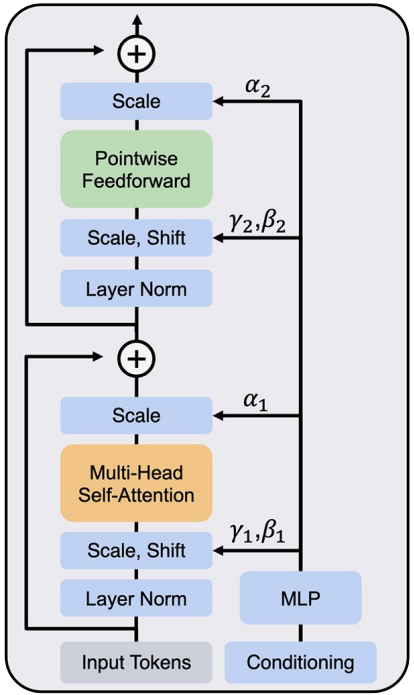


Label embedding



Figures adapted from Scalable Diffusion Models with Transformers, Peebles et al., 2022 and Transformer Architecture: The Positional Encoding, Kazemnejad, 2019.

DiT: injecting conditions with adaptive layer norm

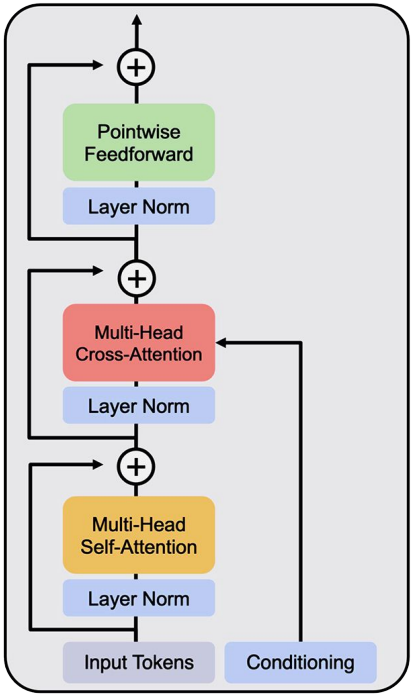


Adaptive Layer Norm

Figures adapted from Scalable Diffusion Models with Transformers, Peebles et al., 2022.

DiT: injecting conditions with cross-attention

Adaptive Layer Norm



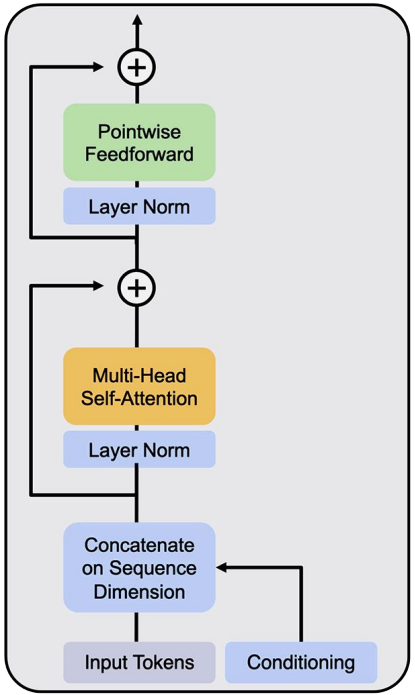
Cross-attention

Figures adapted from Scalable Diffusion Models with Transformers, Peebles et al., 2022.

DiT: injecting conditions with in-context conditioning

Adaptive Layer Norm

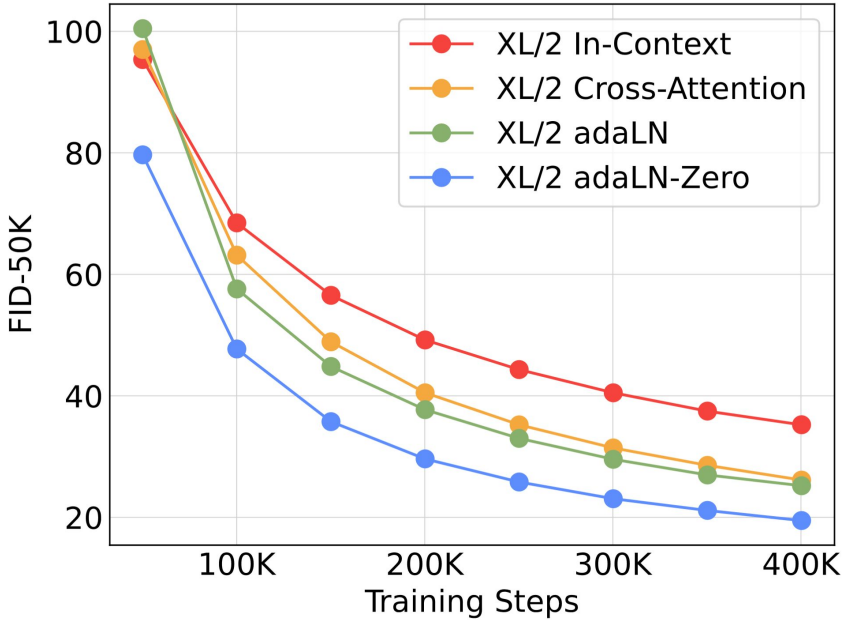
Cross-attention



In-context conditioning

Figures adapted from Scalable Diffusion Models with Transformers, Peebles et al., 2022.

DiT: comparison of different condition injections



1. Adaptive Layer Norm
2. Cross-attention
3. In-context conditioning

Figure from Scalable Diffusion Models with Transformers, Peebles et al., 2022.

DiT: comparison of different condition injections

1. **Adaptive Layer Norm**
2. Cross-attention
3. In-context conditioning

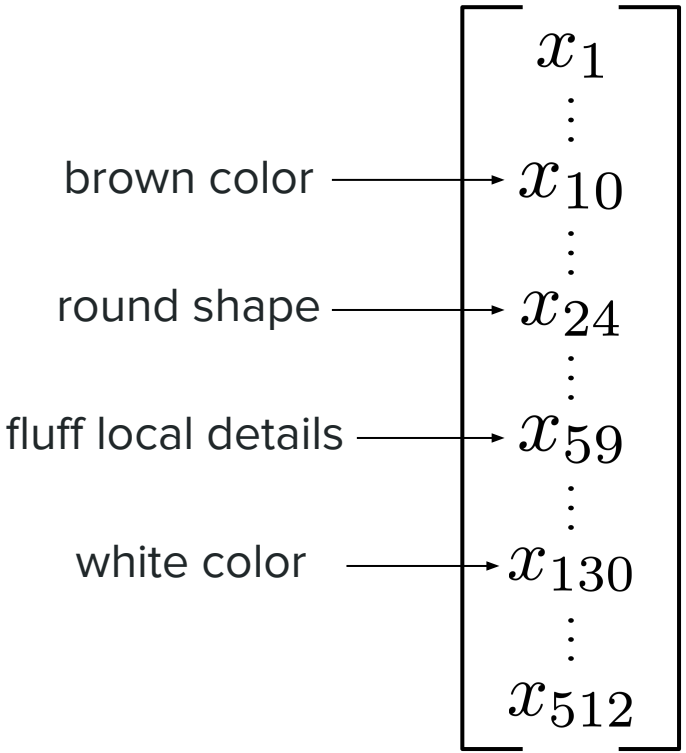
Figure from Scalable Diffusion Models with Transformers, Peebles et al., 2022.

Intuition behind adaptive layer norm

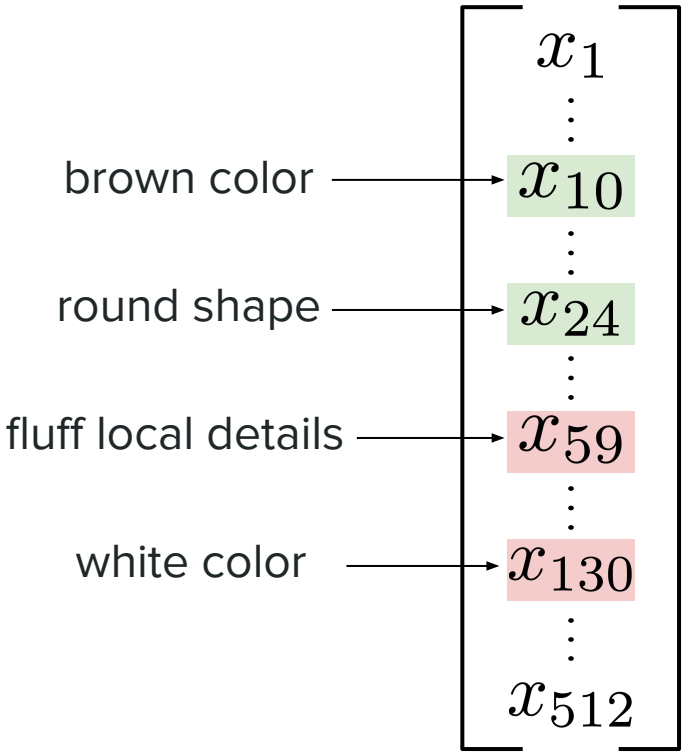
$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix}$$

Representation of a patch embedding

Intuition behind adaptive layer norm



Intuition behind adaptive layer norm



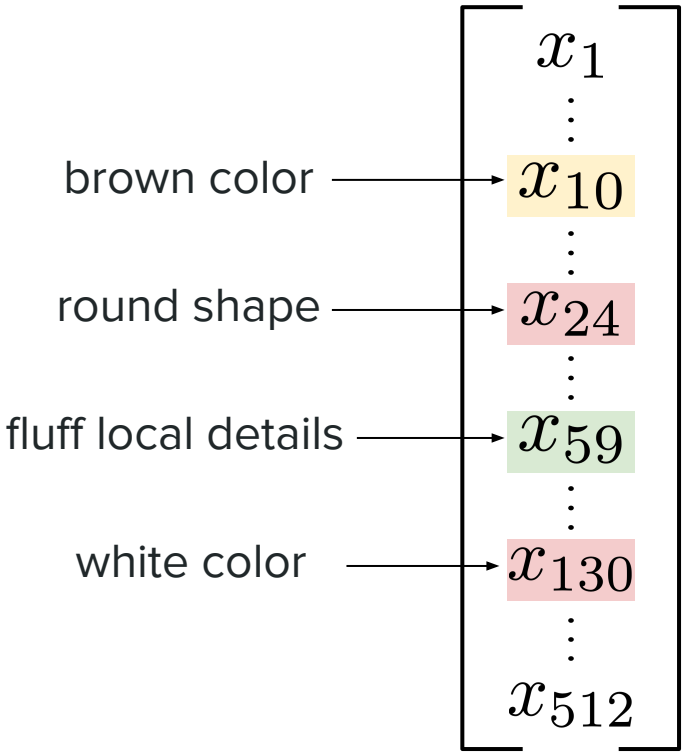
Conditions

c `brown fluffy teddy bear`

$t = 0.1$ **a lot of** noise

Need to focus on **global structure** since at the early stage of generation.

Intuition behind adaptive layer norm



Conditions

c `brown fluffy teddy bear`

$t = 0.9$ almost **no** noise

Need to focus on **local details** since almost arriving at the final image.

Adaptive layer normalization

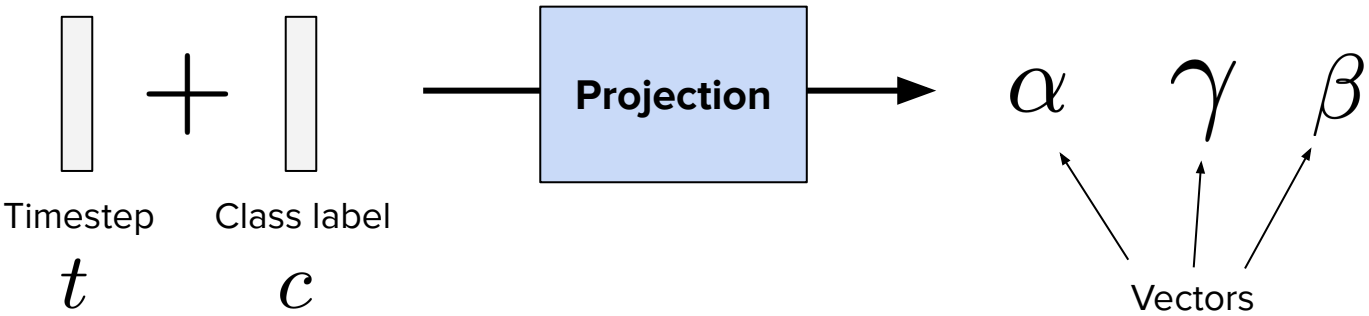
Idea. "Modulate" vector with respect to conditions (timestep and class label)

Adaptive layer normalization

Idea. "Modulate" vector with respect to conditions (timestep and class label)

Steps.

1. Determine intensity of modulation as a function of conditions



Adaptive layer normalization

Idea. "Modulate" vector with respect to conditions (timestep and class label)

Steps.

1. Determine intensity of modulation as a function of conditions

2. Modulate token with quantities

$$x$$

Adaptive layer normalization

Idea. "Modulate" vector with respect to conditions (timestep and class label)

Steps.

1. Determine intensity of modulation as a function of conditions

2. Modulate token with quantities

$$\mathrm{LN}(x)$$

Adaptive layer normalization

Idea. "Modulate" vector with respect to conditions (timestep and class label)

Steps.

1. Determine intensity of modulation as a function of conditions

2. Modulate token with quantities

$$\mathrm{LN}(x) \star (1 + \gamma)$$

Adaptive layer normalization

Idea. "Modulate" vector with respect to conditions (timestep and class label)

Steps.

1. Determine intensity of modulation as a function of conditions

2. Modulate token with quantities

$$\mathrm{LN}(x) \star (1 + \gamma) + \beta$$

Adaptive layer normalization

Idea. "Modulate" vector with respect to conditions (timestep and class label)

Steps.

1. Determine intensity of modulation as a function of conditions

2. Modulate token with quantities

$$\text{Operation}(\text{LN}(x) \star (1 + \gamma) + \beta)$$

Adaptive layer normalization

Idea. "Modulate" vector with respect to conditions (timestep and class label)

Steps.

1. Determine intensity of modulation as a function of conditions

2. Modulate token with quantities

$$\alpha \star \text{Operation}(\text{LN}(x) \star (1 + \gamma) + \beta)$$

Adaptive layer normalization

Idea. "Modulate" vector with respect to conditions (timestep and class label)

Steps.

1. Determine intensity of modulation as a function of conditions

2. Modulate token with quantities

$$x + \alpha \star \text{Operation}(\text{LN}(x) \star (1 + \gamma) + \beta)$$

Adaptive layer normalization

Idea. "Modulate" vector with respect to conditions (timestep and class label)

Steps.

1. Determine intensity of modulation as a function of conditions

2. Modulate token with quantities

$$x \longleftarrow x + \alpha \star \text{Operation}(\text{LN}(x) \star (1 + \gamma) + \beta)$$

Adaptive layer normalization

Idea. "Modulate" vector with respect to conditions (timestep and class label)

Steps.

1. Determine intensity of modulation as a function of conditions

2. Modulate token with quantities

$$x \longleftarrow x + \alpha \star \text{Operation}(\text{LN}(x) \star (1 + \gamma) + \beta)$$

element-wise multiplication

Adaptive layer normalization

Idea. "Modulate" vector with respect to conditions (timestep and class label)

Steps.

1. Determine intensity of modulation as a function of conditions

2. Modulate token with quantities

$$x \longleftarrow x + \alpha \star \text{Operation}(\text{LN}(x) \star (1 + \gamma) + \beta)$$

Gate **Scale** **Shift**

Adaptive layer normalization

Idea. "Modulate" vector with respect to conditions (timestep and class label)

Steps.

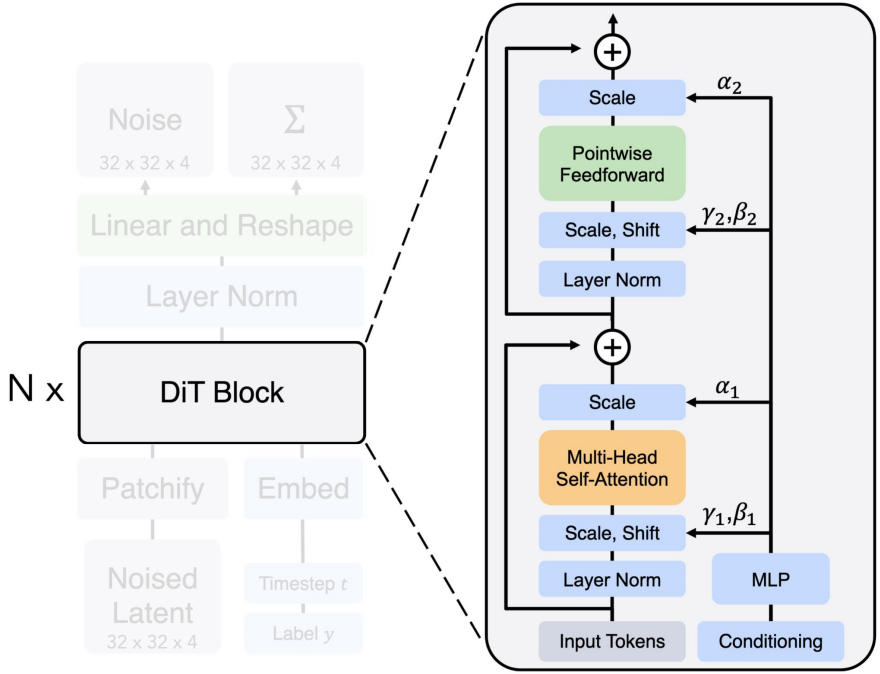
1. Determine intensity of modulation as a function of conditions

2. Modulate token with quantities

$$x \longleftarrow x + \underbrace{\alpha \star \text{Operation}(\text{LN}(x) \star (1+\gamma) + \beta)}$$

At the beginning: $= 0$ adaLN-**Zero**

DiT: injection of conditions with adaLN-Zero



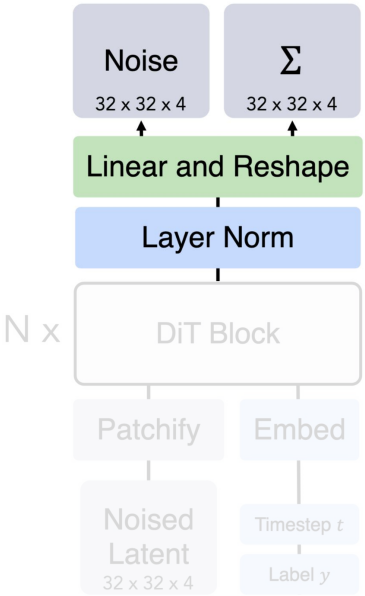
Modulation of representation with:

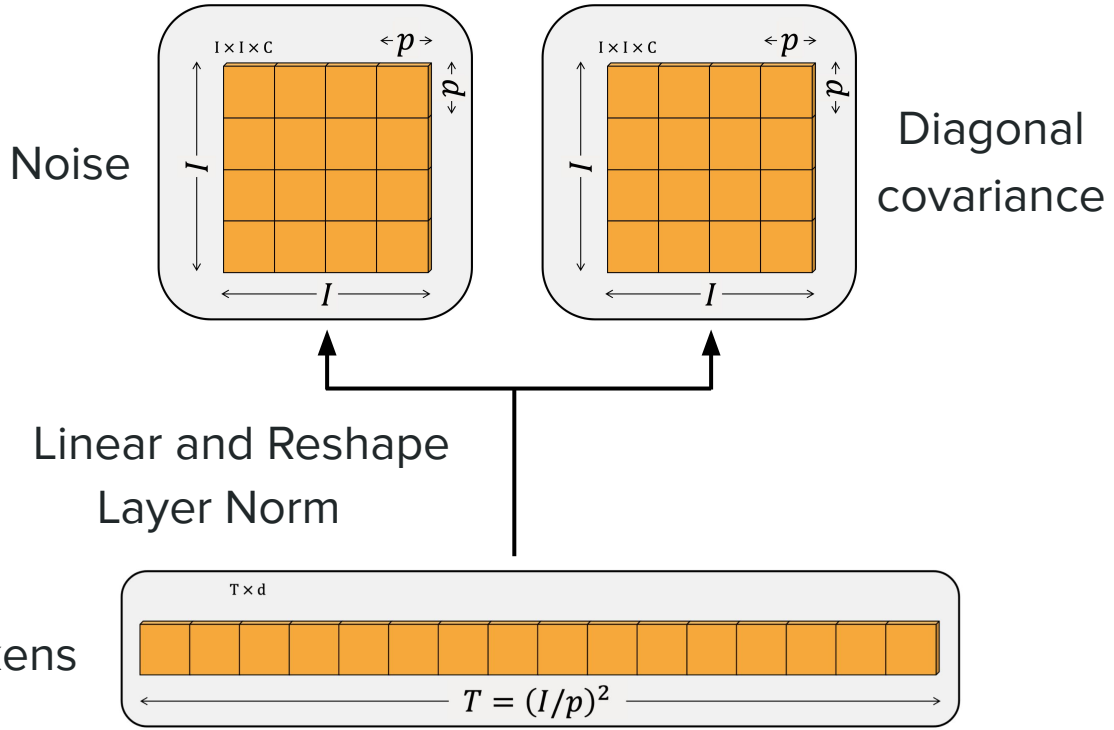
- **Gate** α
- **Scale** γ
- **Shift** β

as a function of t, c

Figures adapted from Scalable Diffusion Models with Transformers, Peebles et al., 2022.

DiT: reformatting the output





Figures adapted from Scalable Diffusion Models with Transformers, Peebles et al., 2022.

Scaling up DiT...

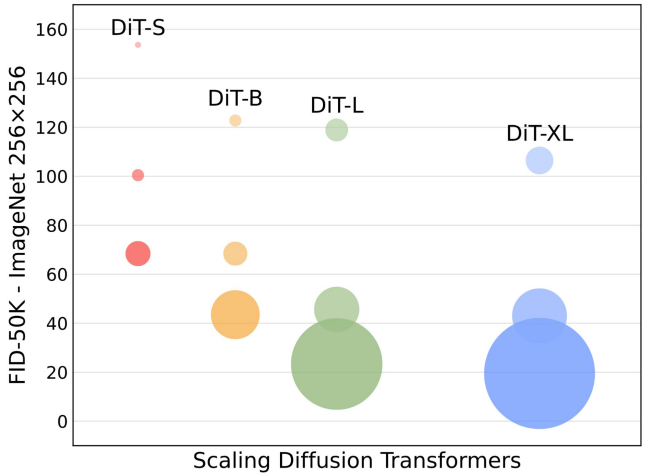
Model	Layers N	Hidden size d	Heads	Gflops (I=32, p=4)
DiT-S	12	384	6	1.4
DiT-B	12	768	12	5.6
DiT-L	24	1024	16	19.7
DiT-XL	28	1152	16	29.1

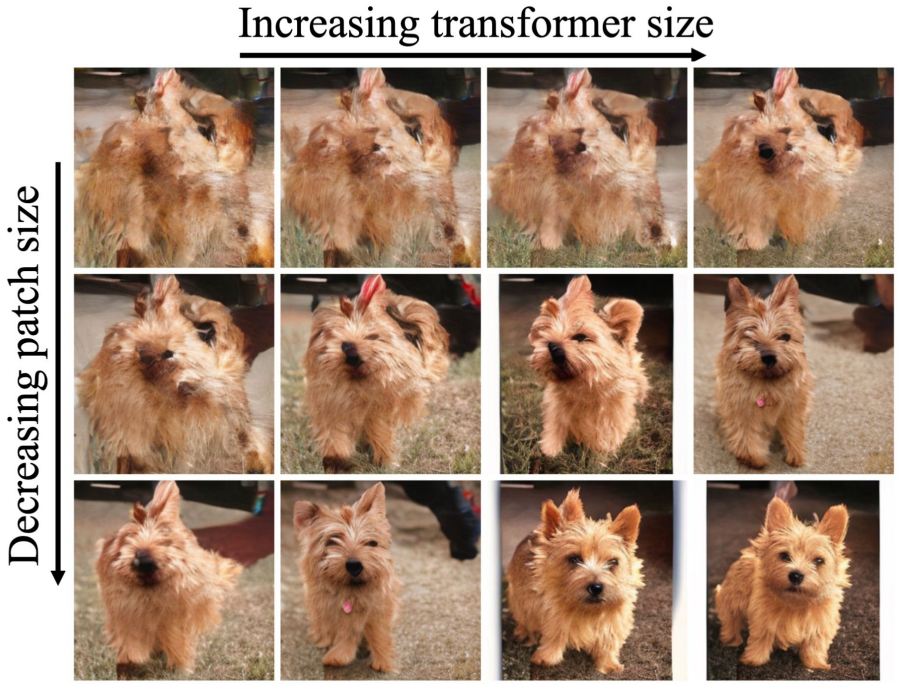
- Number of parameters is **not enough** to quantify model complexity
- **FL**oating-point **OP**eration**s** (FLOPs) = # operations in forward pass
- Smaller patch size corresponds to **higher FLOPs**

Figures from Scalable Diffusion Models with Transformers, Peebles et al., 2022.

Scaling up DiT... improves results!

Model	Layers N	Hidden size d	Heads	Gflops (I=32, p=4)
DiT-S	12	384	6	1.4
DiT-B	12	768	12	5.6
DiT-L	24	1024	16	19.7
DiT-XL	28	1152	16	29.1





Figures from Scalable Diffusion Models with Transformers, Peebles et al., 2022.

Diffusion & Large Vision Models

Motivation

U-Net

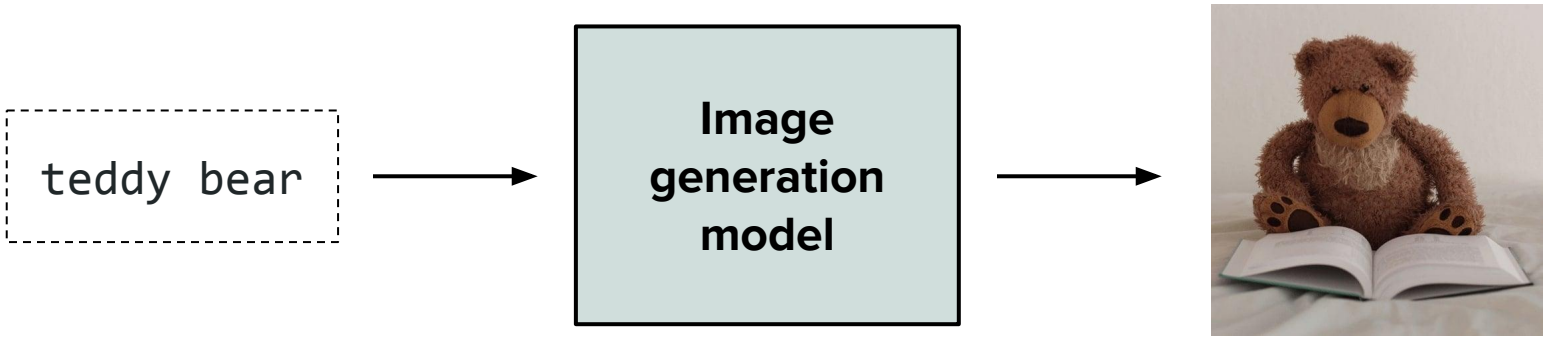
Diffusion Transformer

End-to-end example

Multimodal DiT

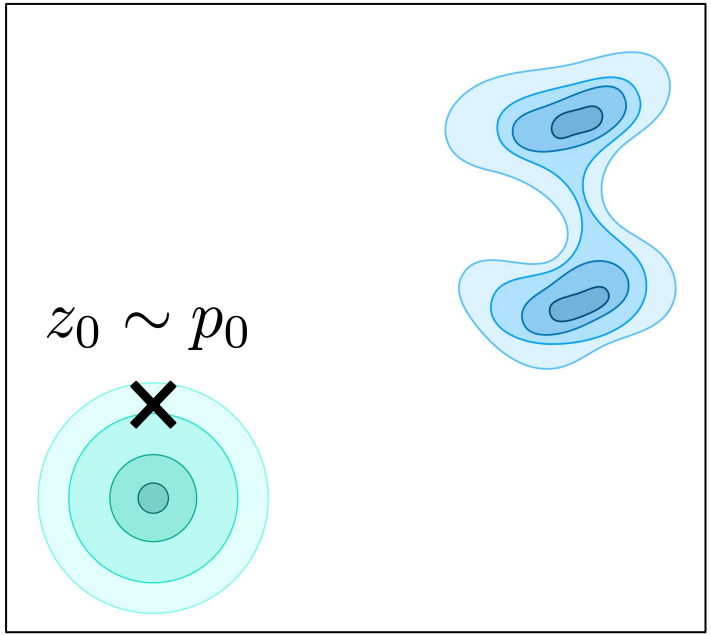
Optimizations

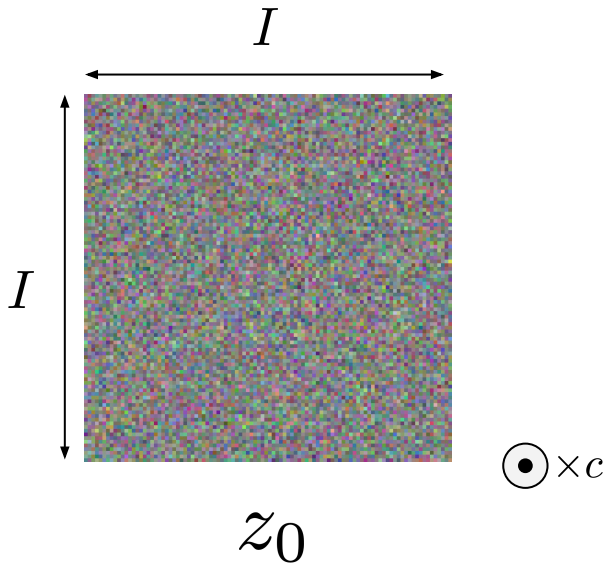
Let's go through an end-to-end example together!



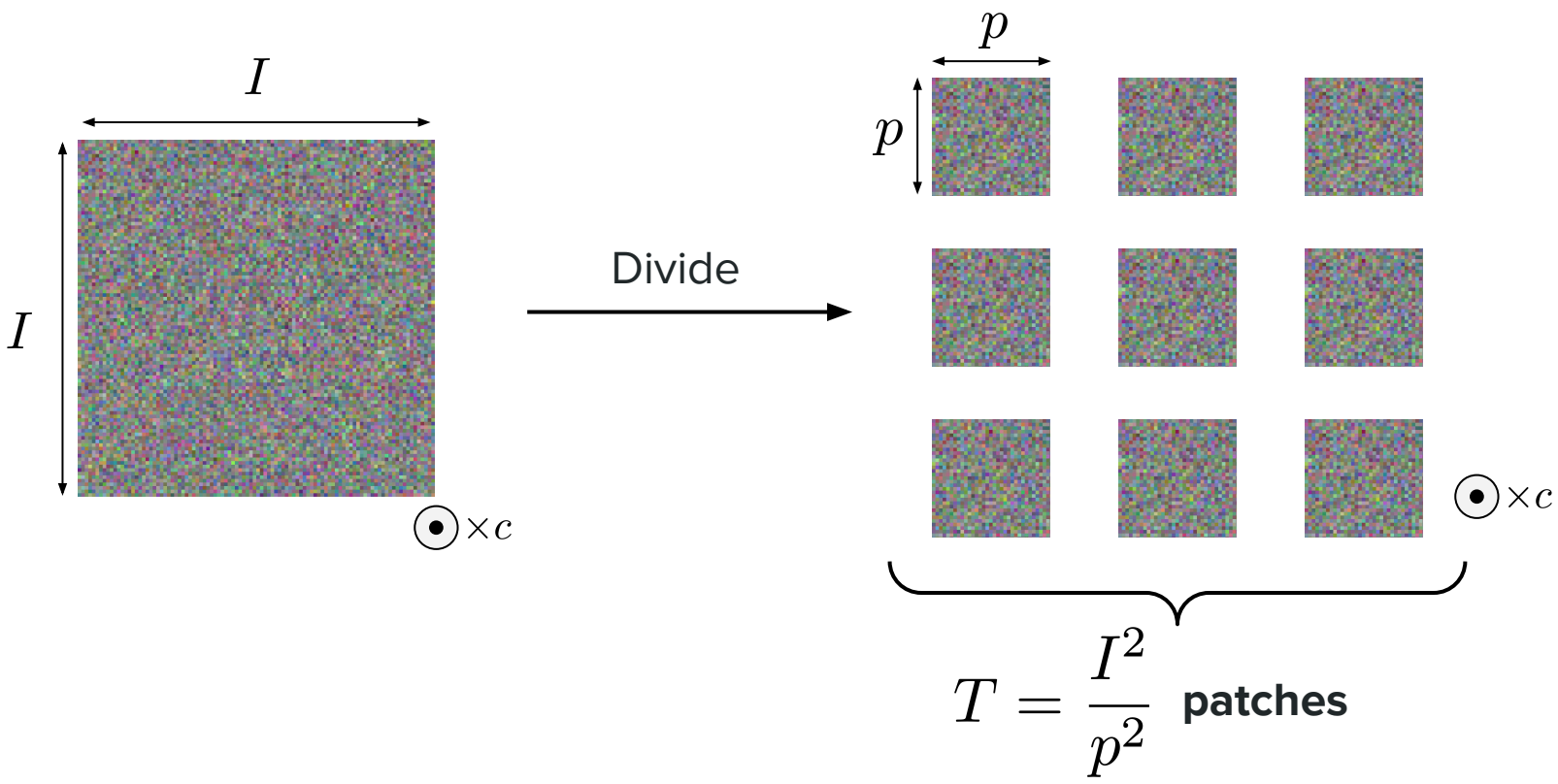
Sample a noisy latent

Latent space
learned by the VAE



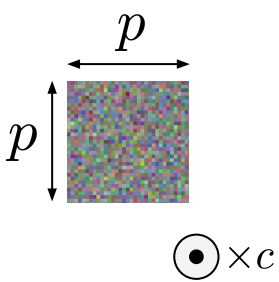


Patchify noisy latent



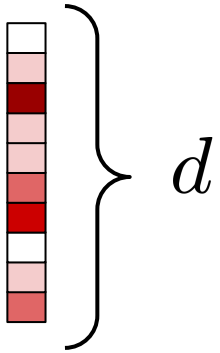
Patchify noisy latent

Patch

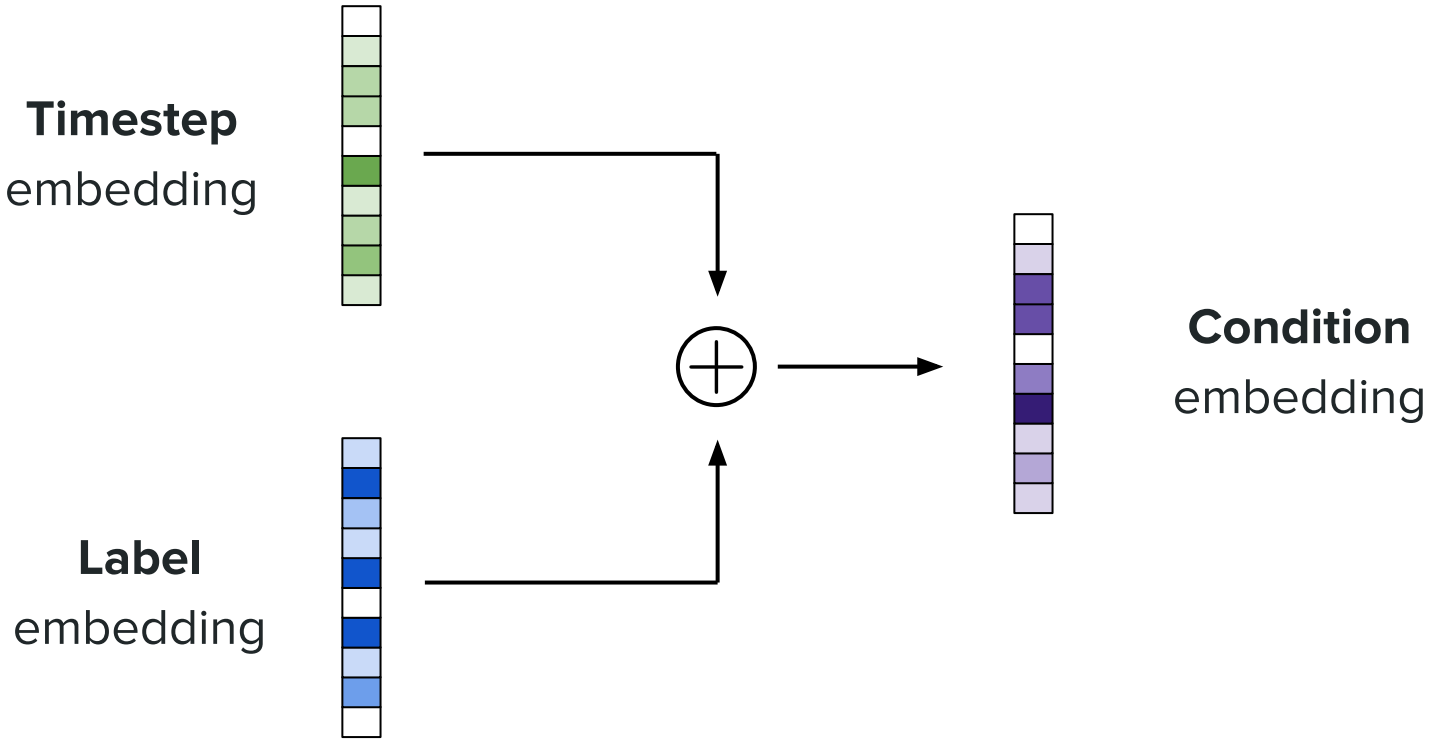


Perform projection

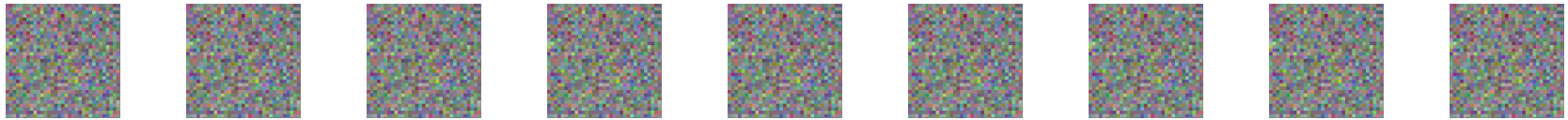
Patch embedding



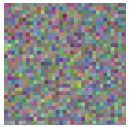
Embed conditions



Attention layers with noisy latent

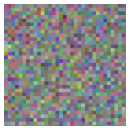


Attention layers with noisy latent



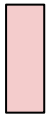
Attention layers with noisy latent

Patch embedding



Attention layers with noisy latent

Position embedding

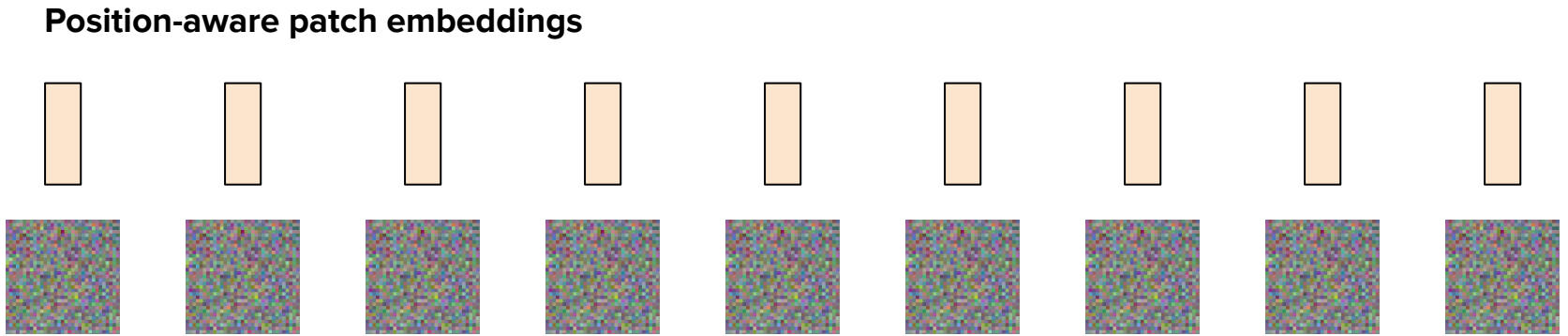


Patch embedding

Attention layers with noisy latent

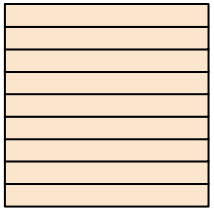
Position-aware patch embedding

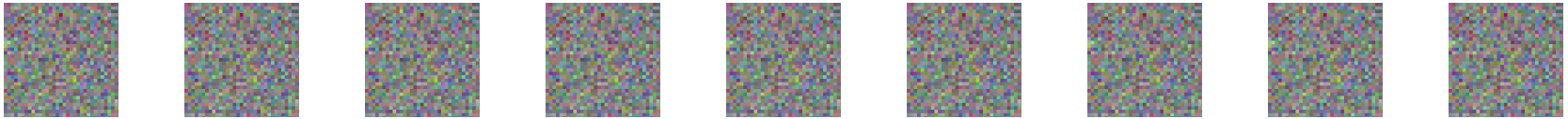
Attention layers with noisy latent



Attention layers with noisy latent

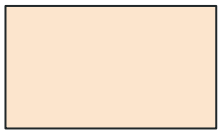
Position-aware patch embeddings matrix

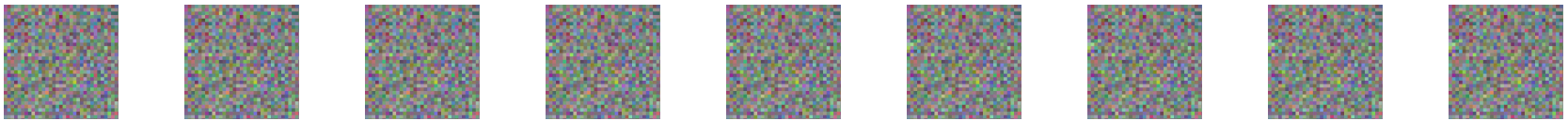




Attention layers with noisy latent

Position-aware patch embeddings matrix

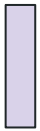


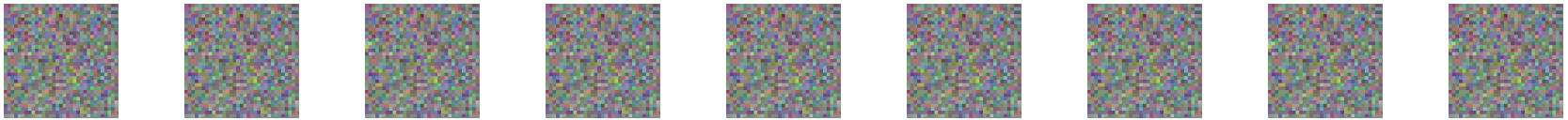


Attention layers with noisy latent

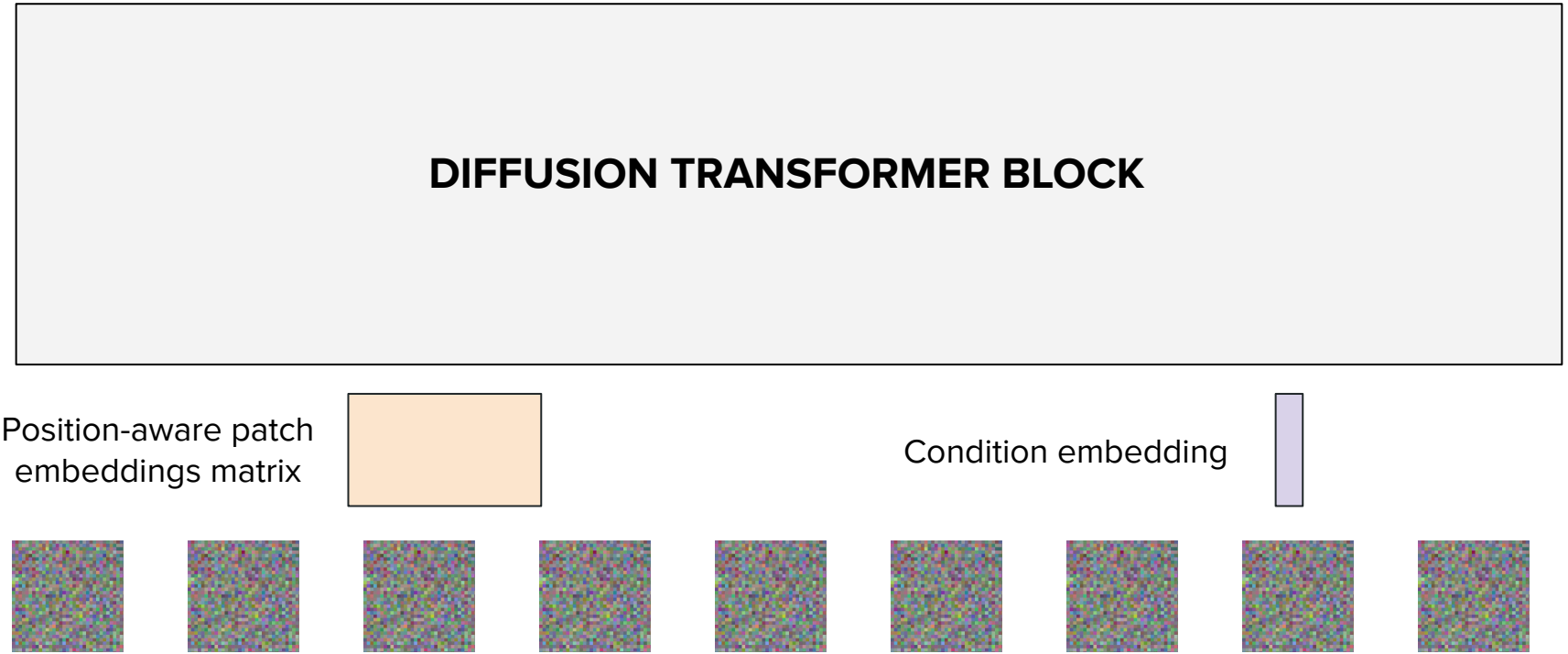
Position-aware patch embeddings matrix

Condition embedding

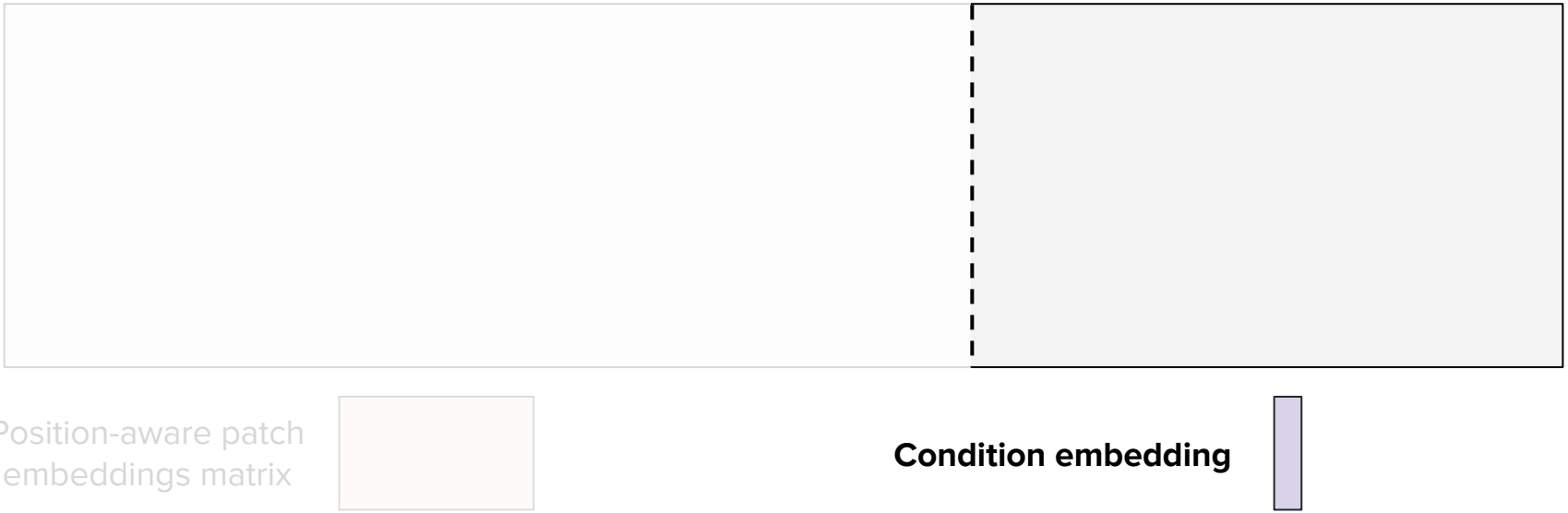




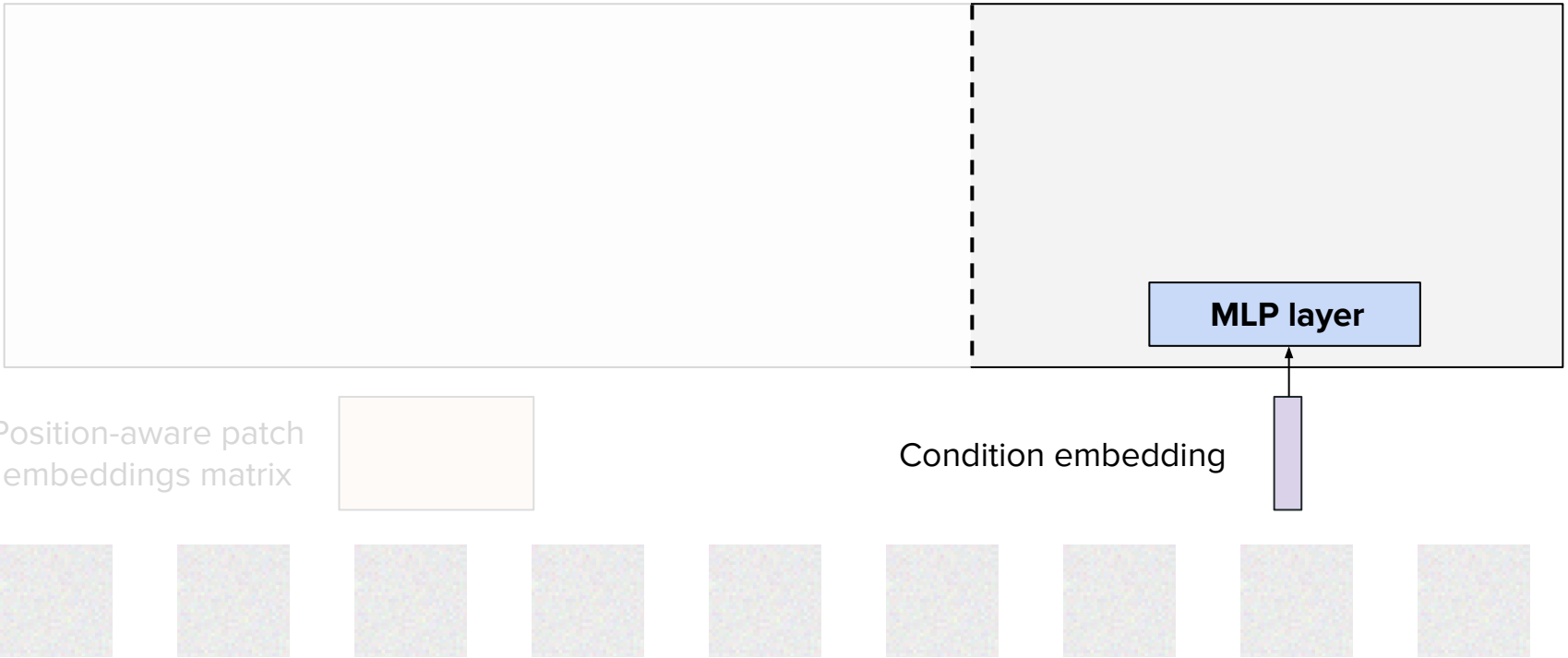
Attention layers with noisy latent



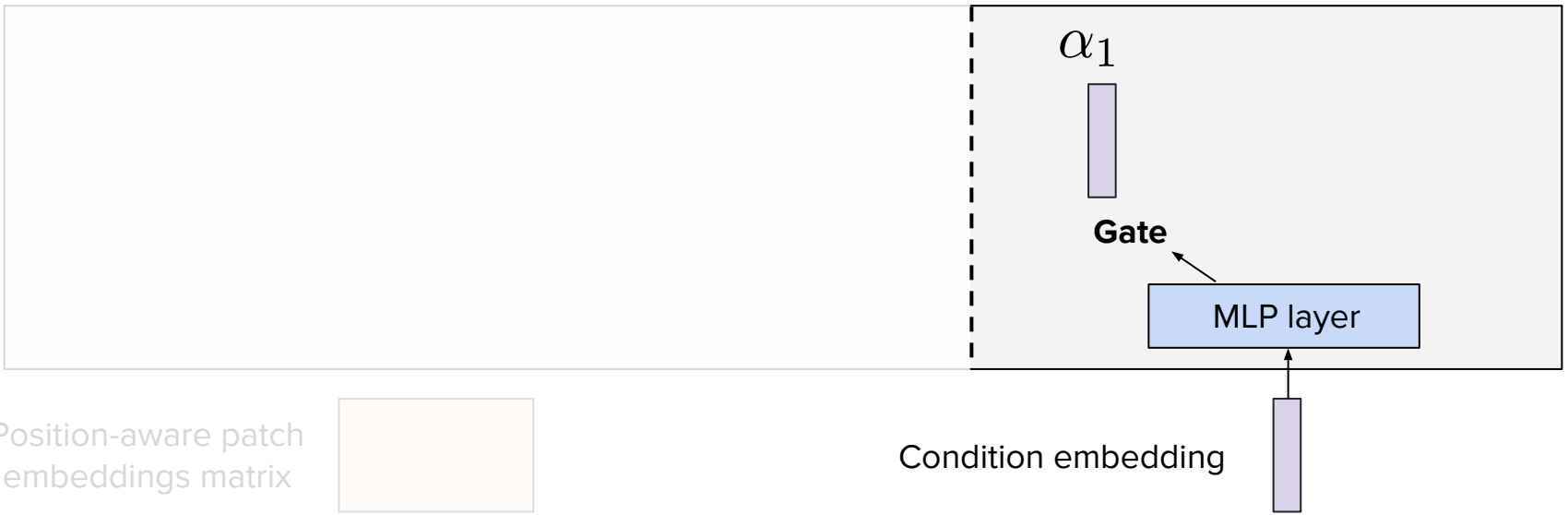
Process condition embeddings



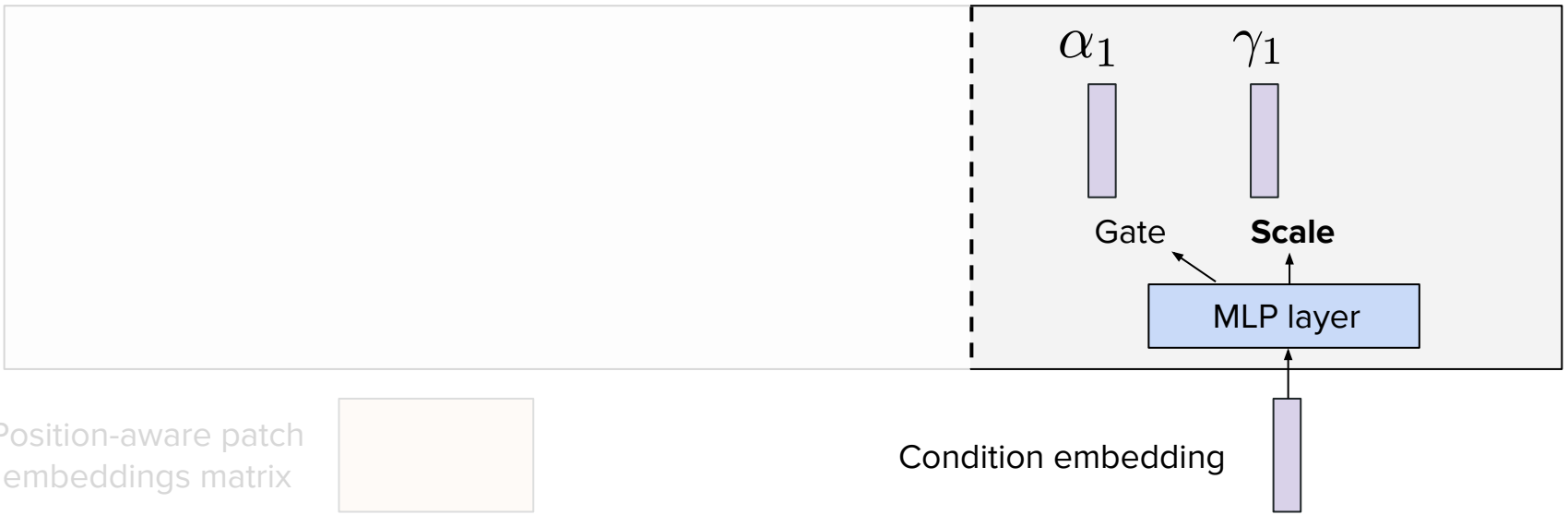
Process condition embeddings



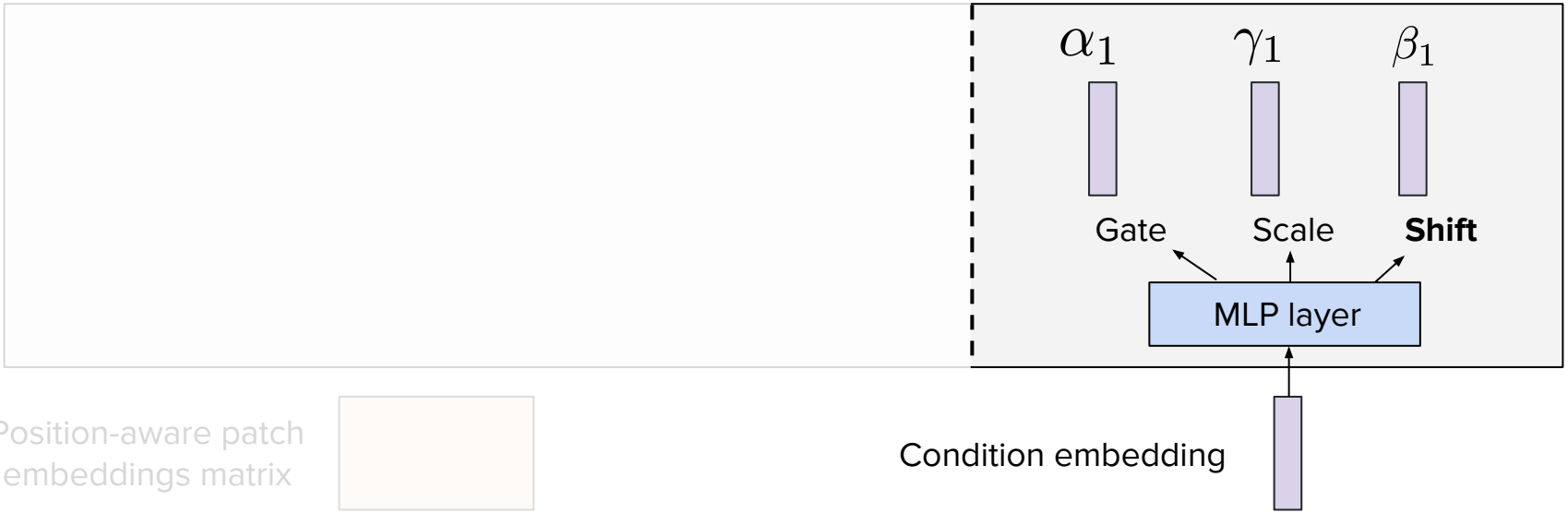
Obtain gate from conditions



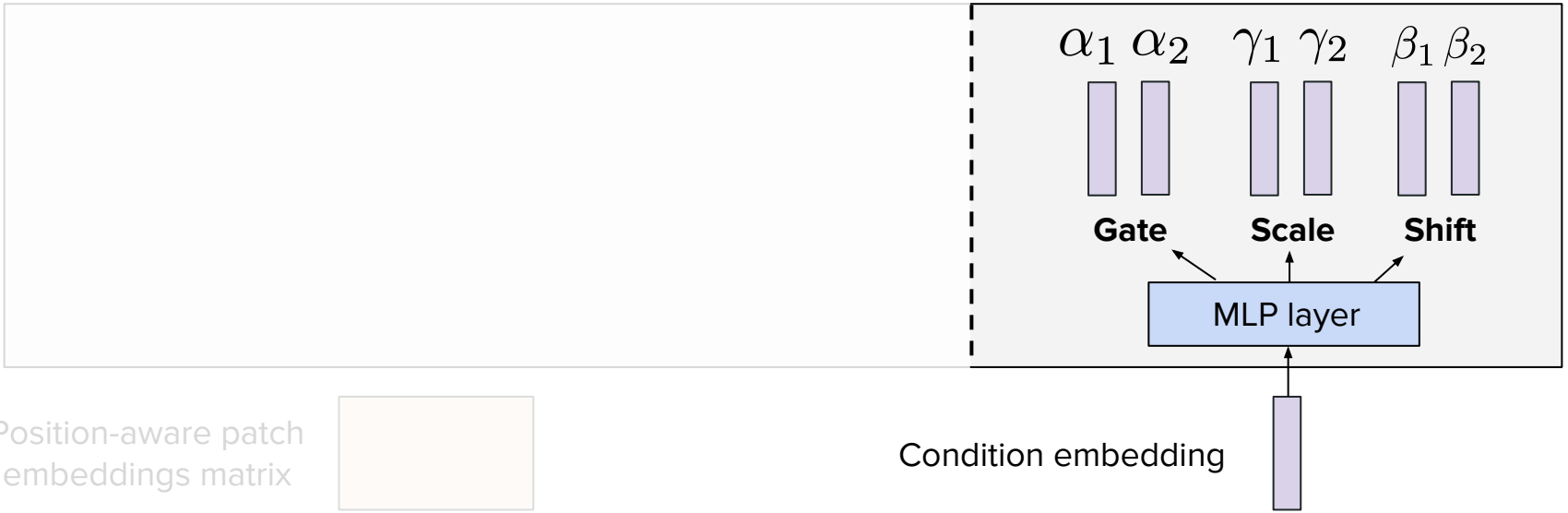
Obtain scale from conditions



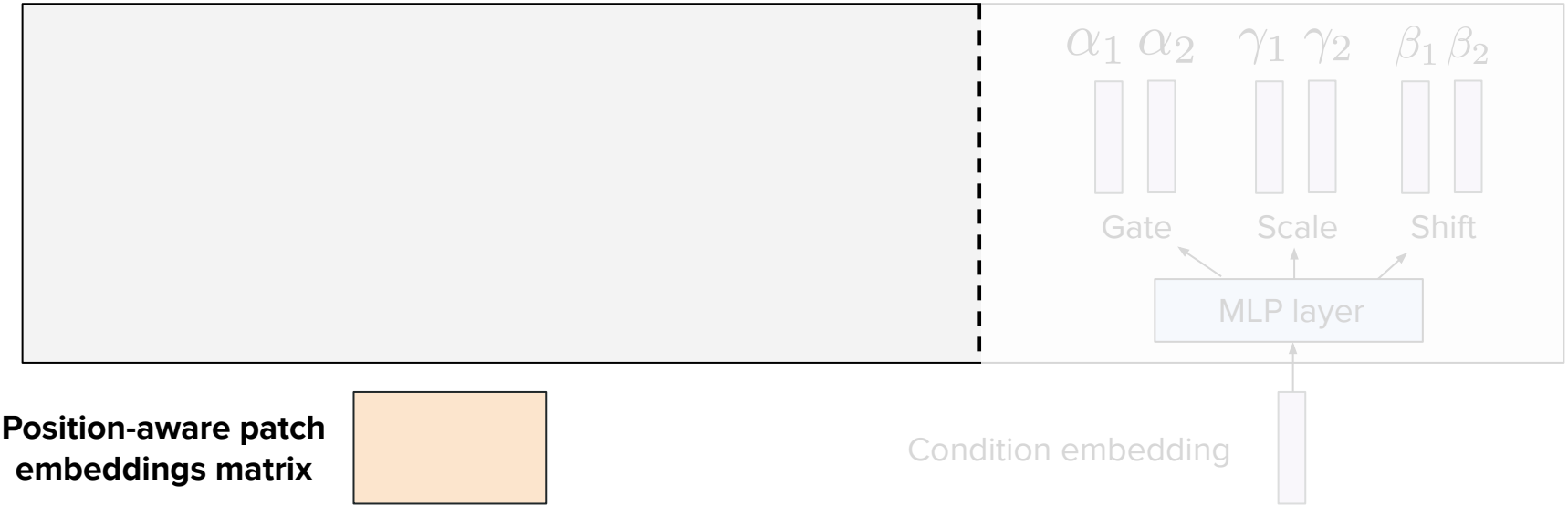
Obtain shift from conditions



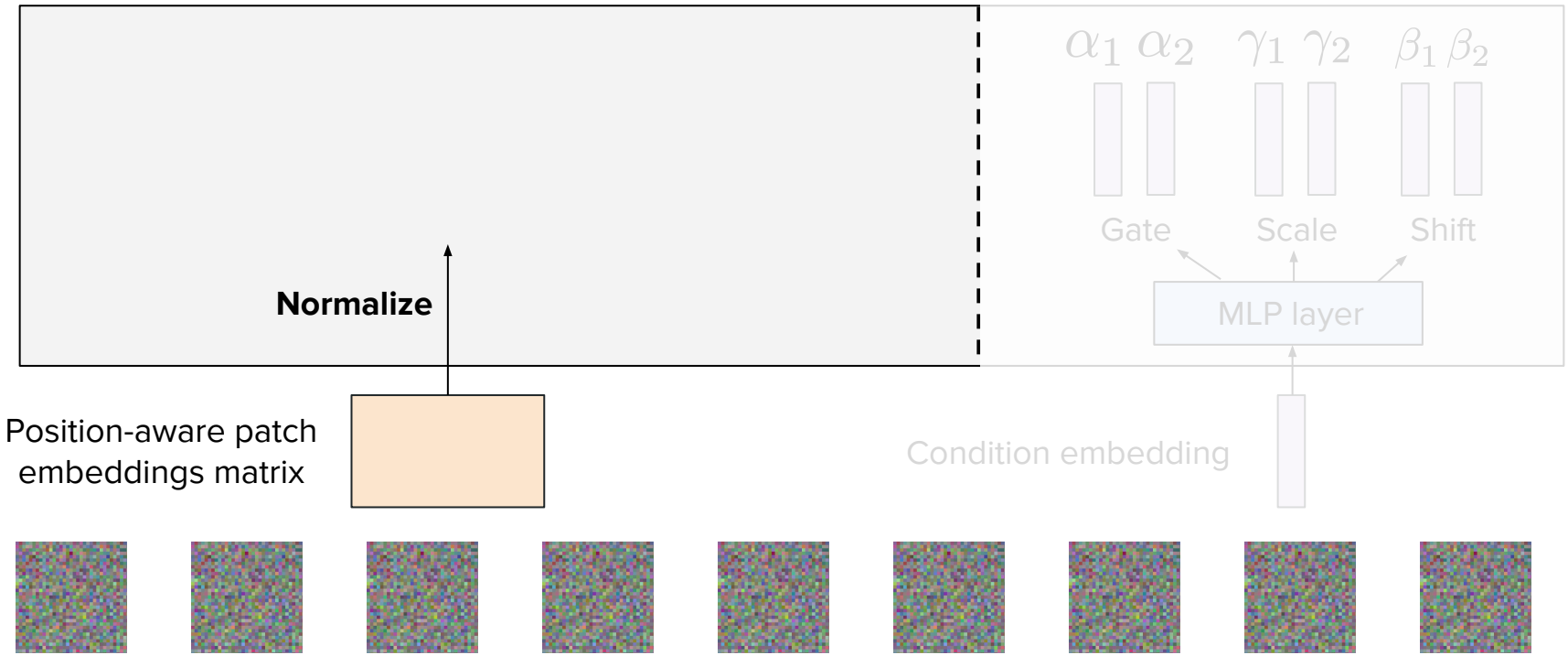
Obtain quantities from condition



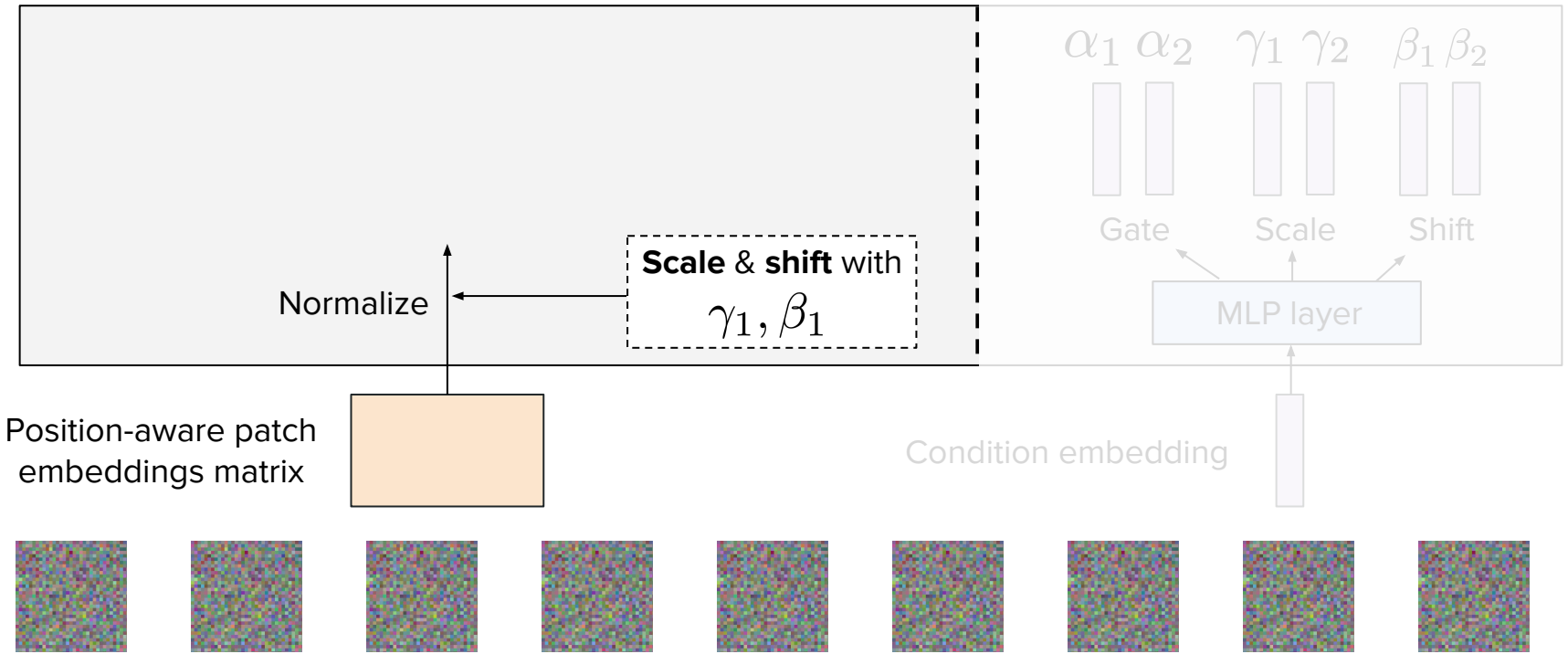
Process patch embeddings



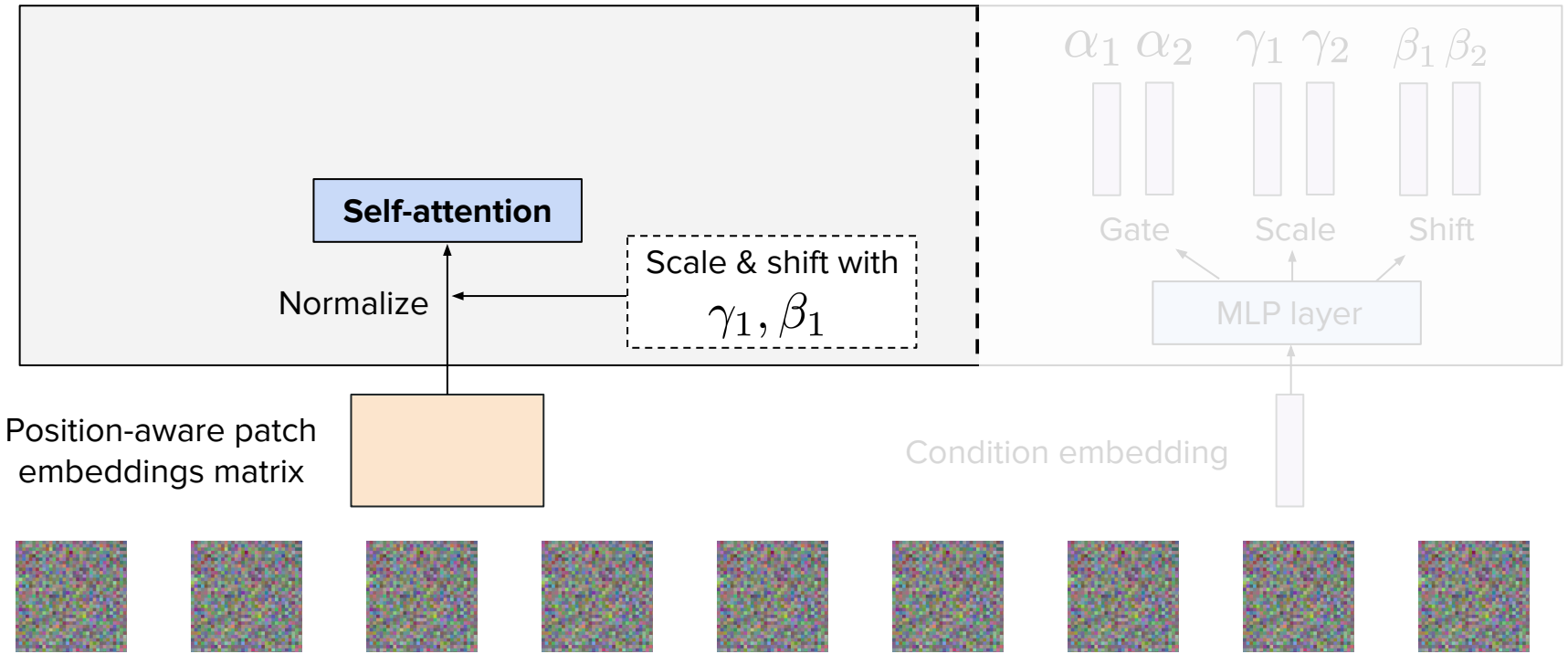
Process patch embeddings through attention



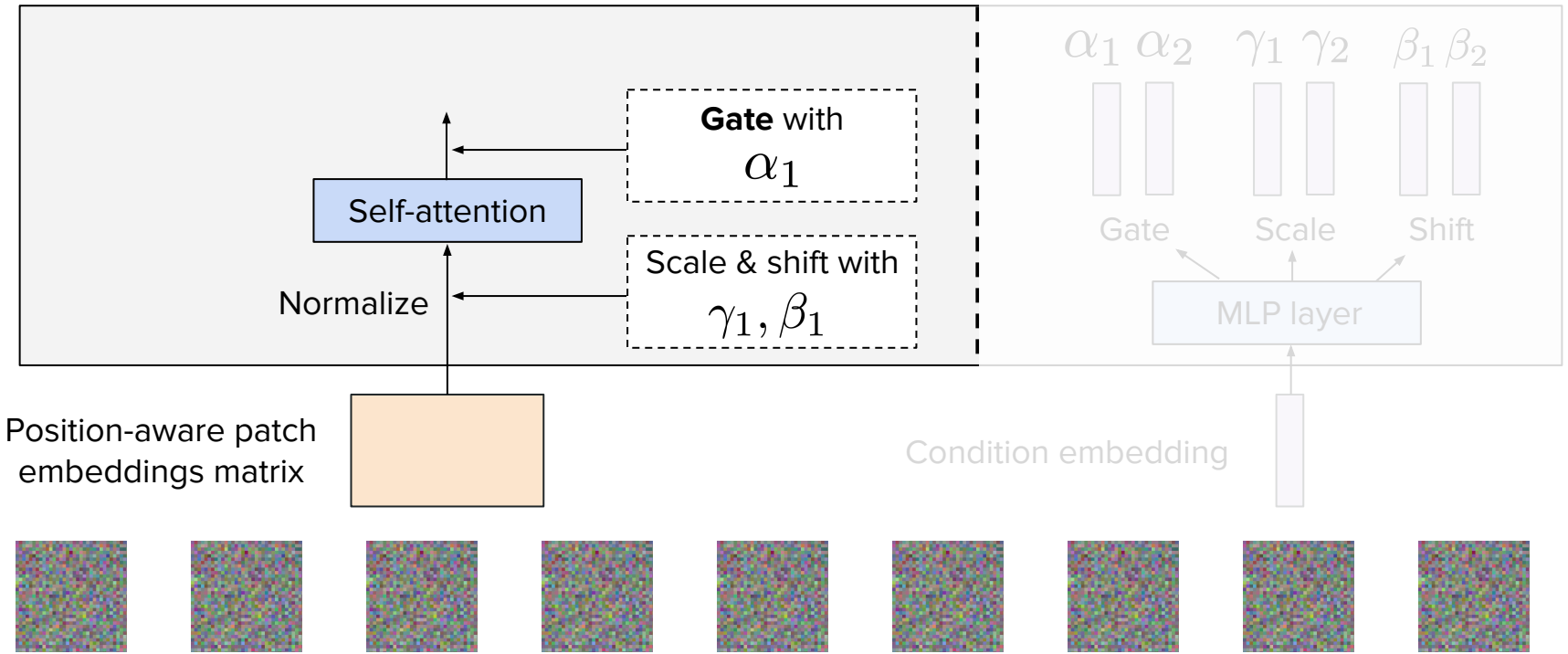
Process patch embeddings through attention



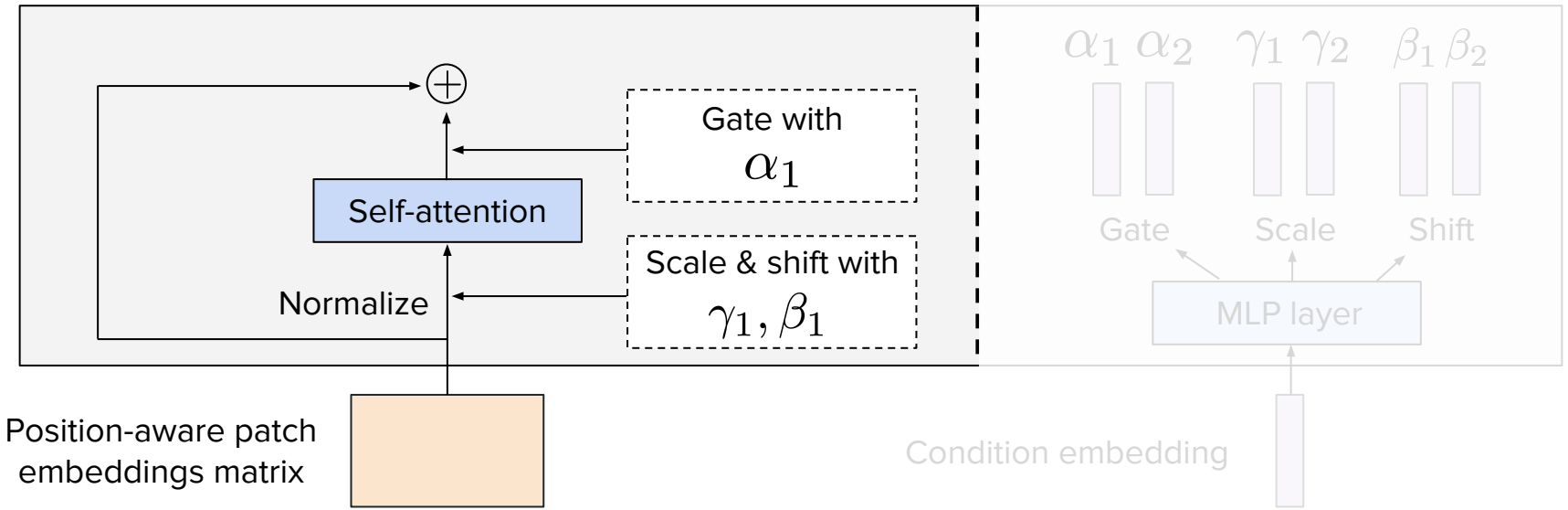
Process patch embeddings through attention



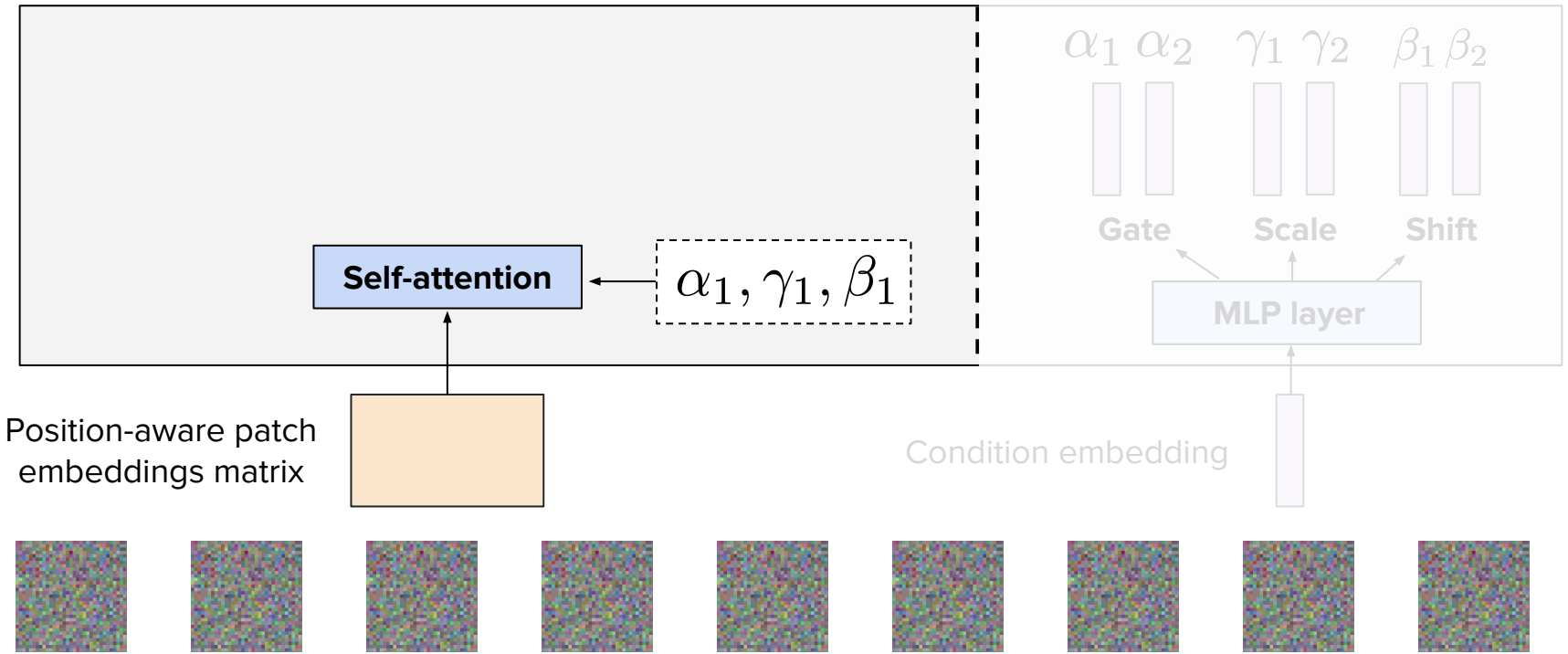
Process patch embeddings through attention



Process patch embeddings through attention



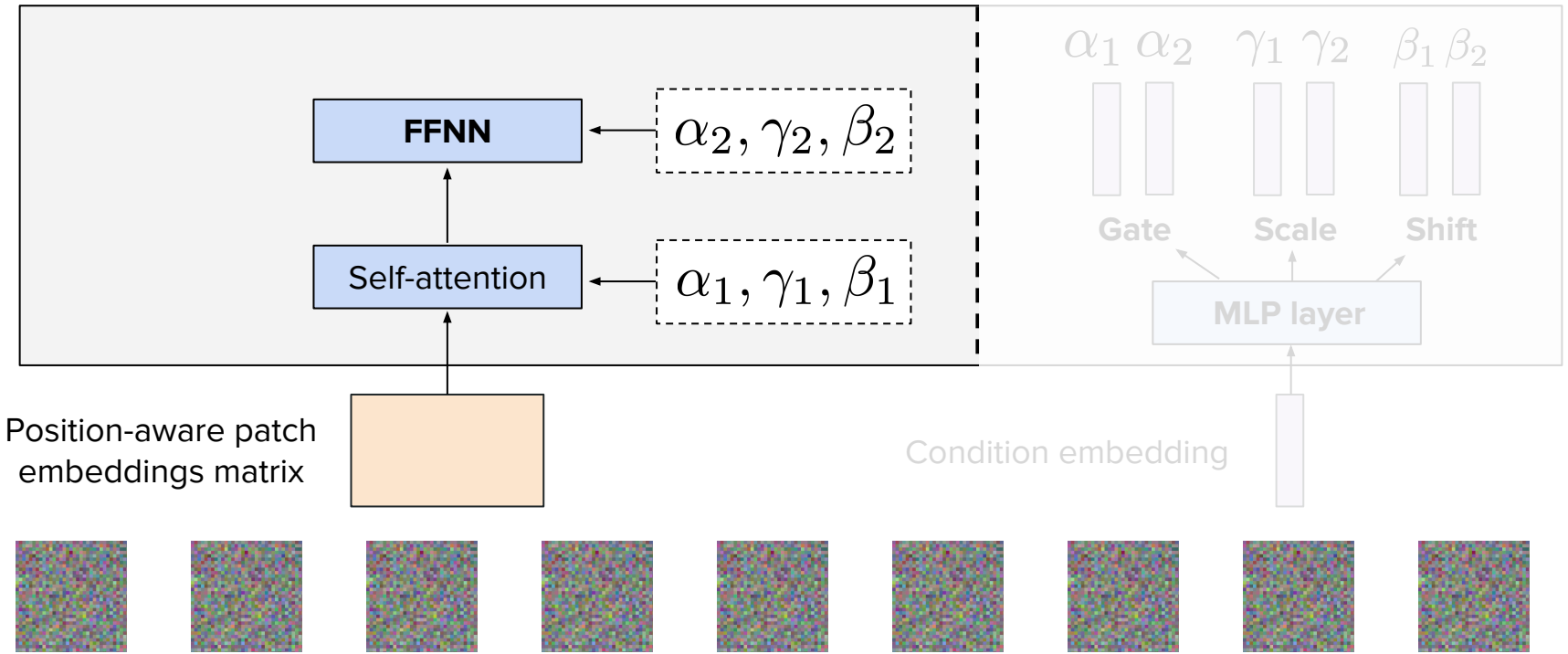
Process patch embeddings through attention



Process patch embeddings through attention

$$x \longleftarrow x + \alpha_1 \star \text{Attention}(\text{LN}(x) \star (1 + \gamma_1) + \beta_1)$$

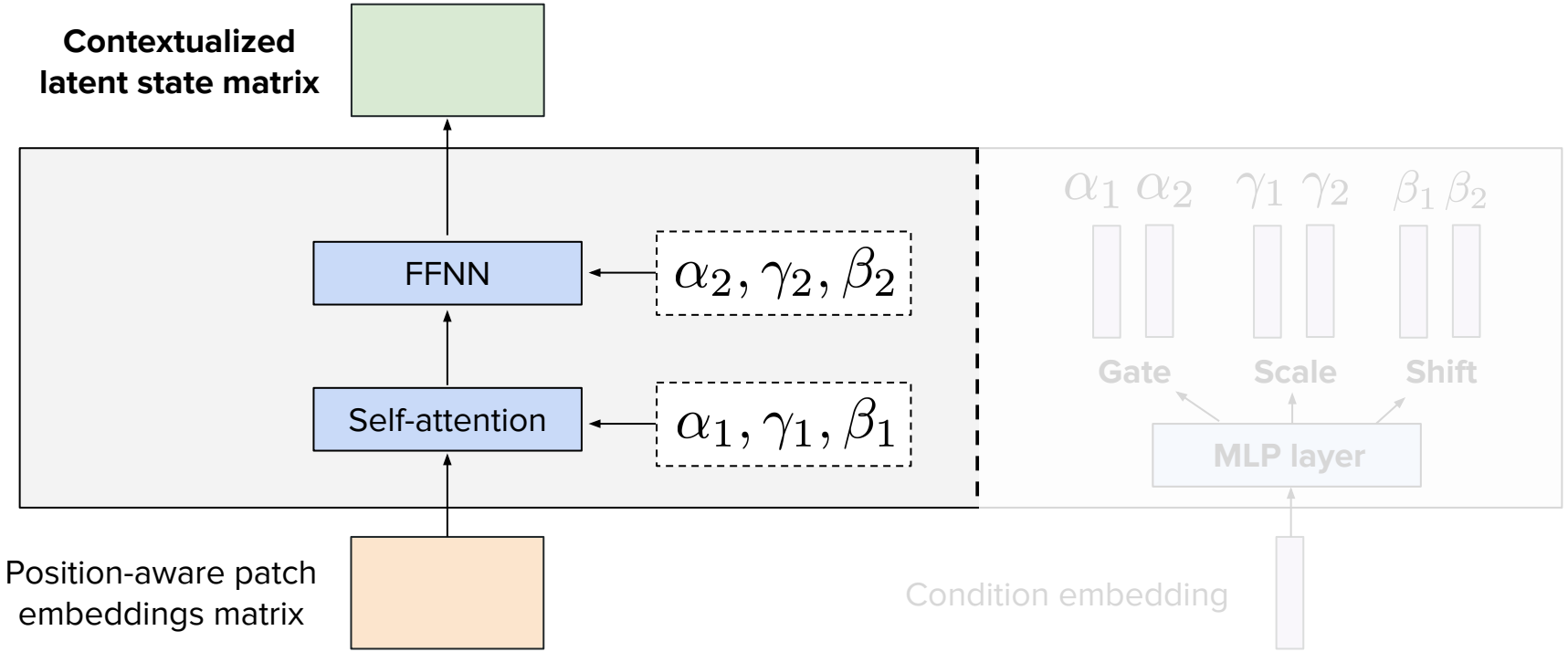
Process patch embeddings through FFNN



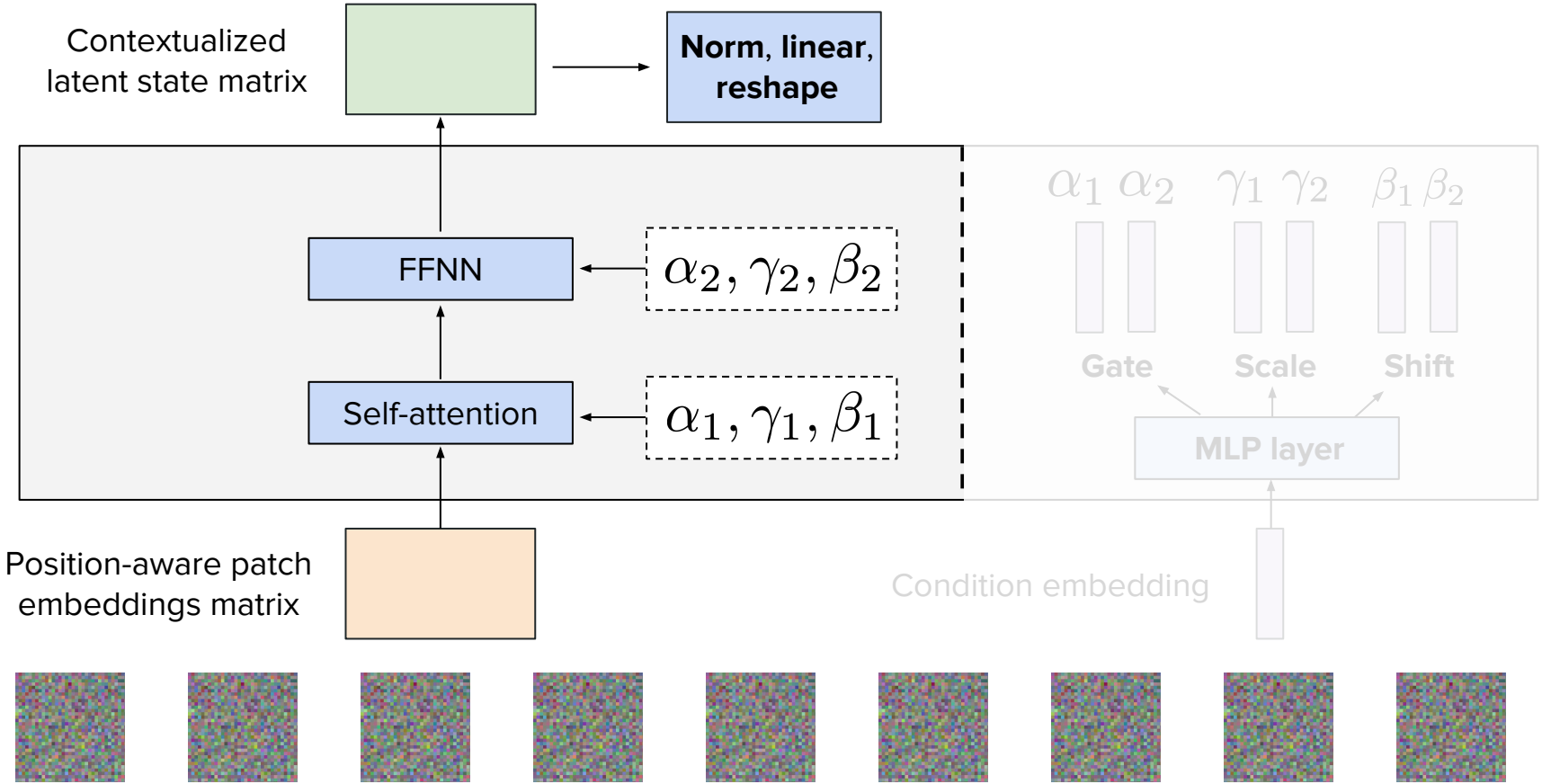
Process patch embeddings through FFNN

$$x \longleftarrow x + \alpha_2 \star \mathrm{FFNN}(\mathrm{LN}(x) \star (1 + \gamma_2) + \beta_2)$$

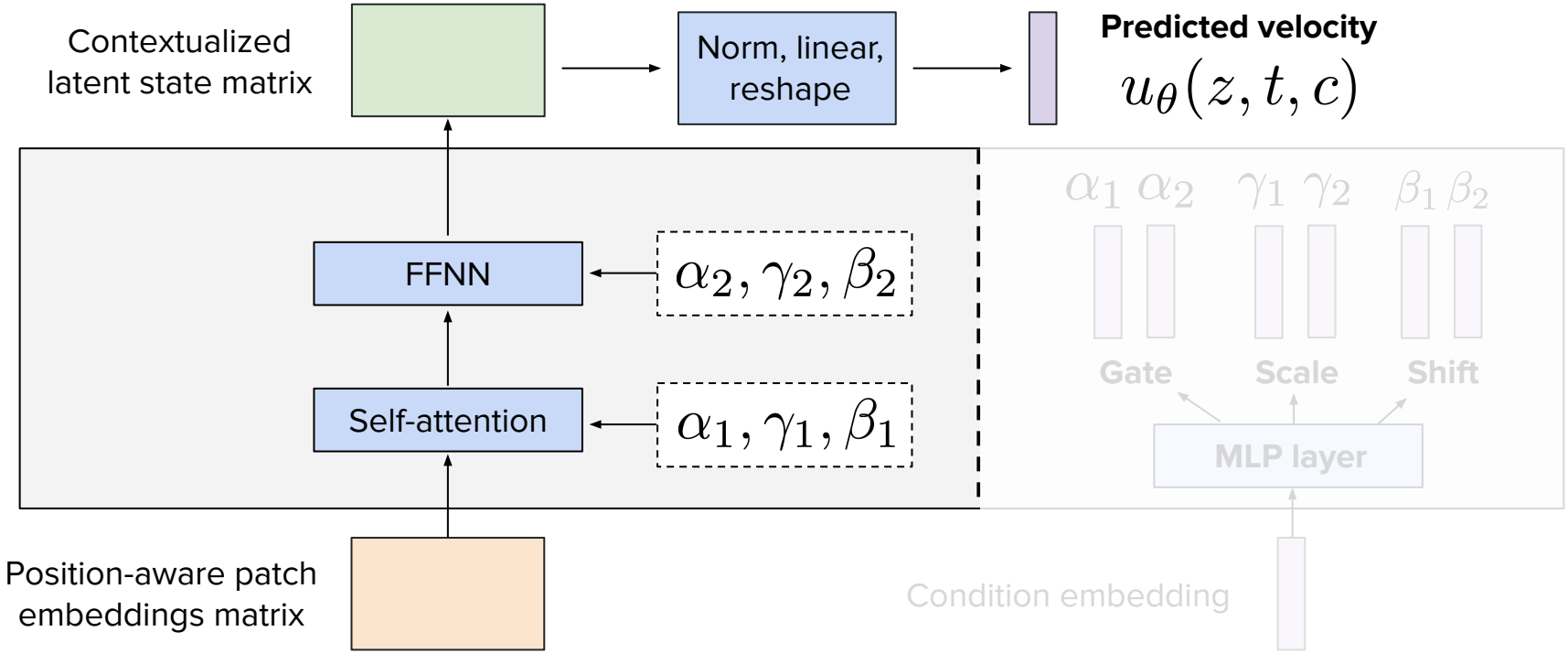
Obtain output from DiT block



Project and reshape to desired quantity



Project and reshape to desired quantity



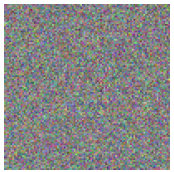
Recap of prediction

Predicted velocity

$$u_\theta(z_0, 0, c)$$

DIFFUSION TRANSFORMER

Latent

z_0 

Timestep

$t = 0$

Class label

c `teddy bear`

Deduce resulting latent

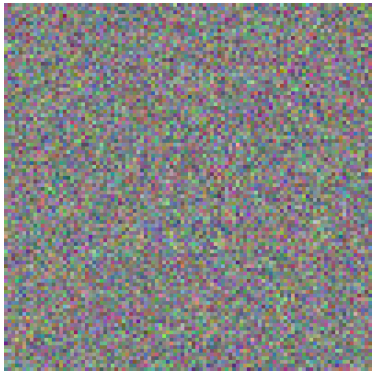
Latent space
learned by the VAE

$$\underbrace{u_\theta(z_0, 0, c)}$$

Prediction from the DiT

Deduce resulting latent

Latent space
learned by the VAE

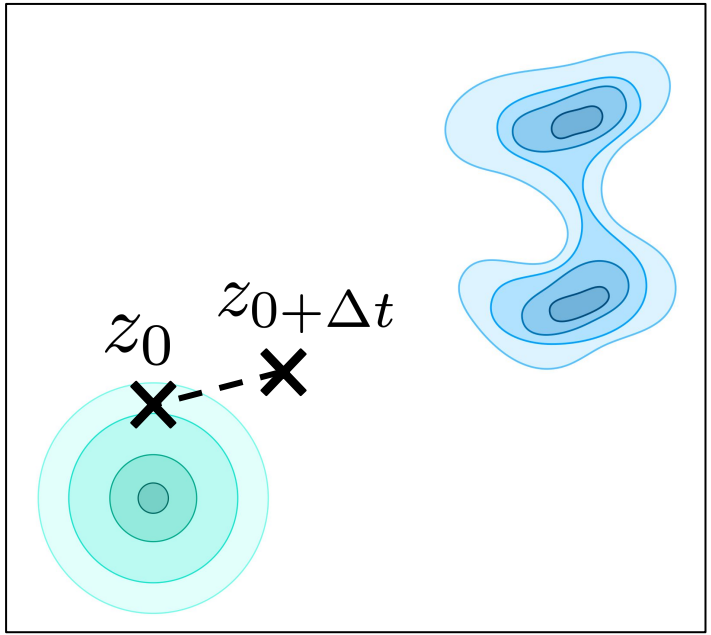


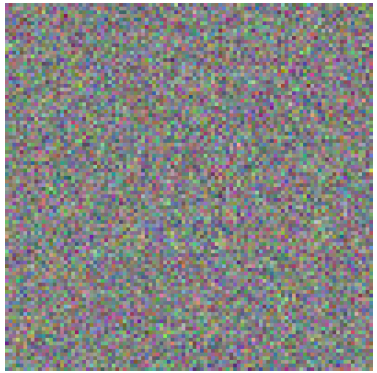
$$z_{0+\Delta t} = z_0 + \underbrace{u_\theta(z_0, 0, c)}\Delta t$$

Prediction from the DiT

Deduce resulting latent

Latent space
learned by the VAE





$$z_{0+\Delta t} = z_0 + \underbrace{u_\theta(z_0, 0, c)}_{\text{Prediction from the DiT}} \Delta t$$

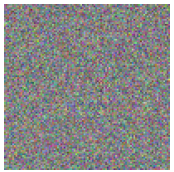
Perform iteration over latents

Predicted velocity

$$u_\theta(z_{t_i}, t_i, c)$$

DIFFUSION TRANSFORMER

Latent

z_{t_i} 

Timestep

t_i

Class label

c `teddy bear`

Perform iteration over latents

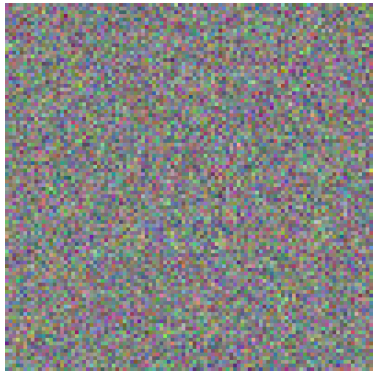
Latent space
learned by the VAE

$$\underbrace{u_\theta(z_{t_i}, t_i, c)}$$

Prediction from the DiT

Perform iteration over latents

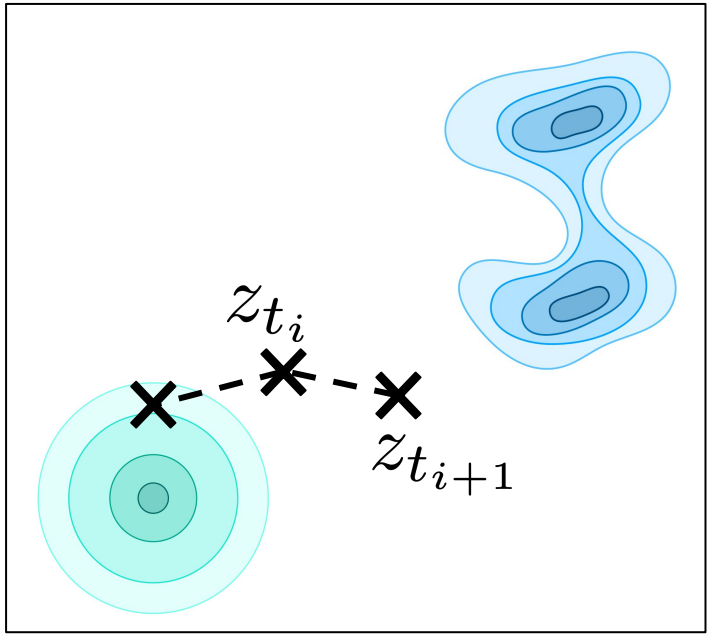
Latent space
learned by the VAE

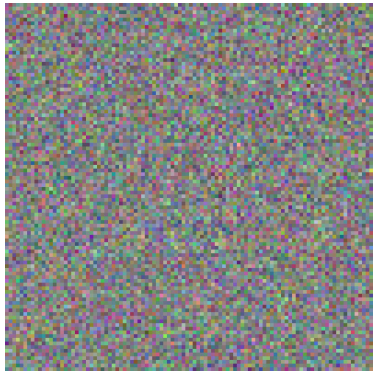


$$z_{t_{i+1}} = z_{t_i} + \underbrace{u_\theta(z_{t_i}, t_i, c)}_{\text{Prediction from the DiT}}(t_{i+1} - t_i)$$

Perform iteration over latents

Latent space
learned by the VAE



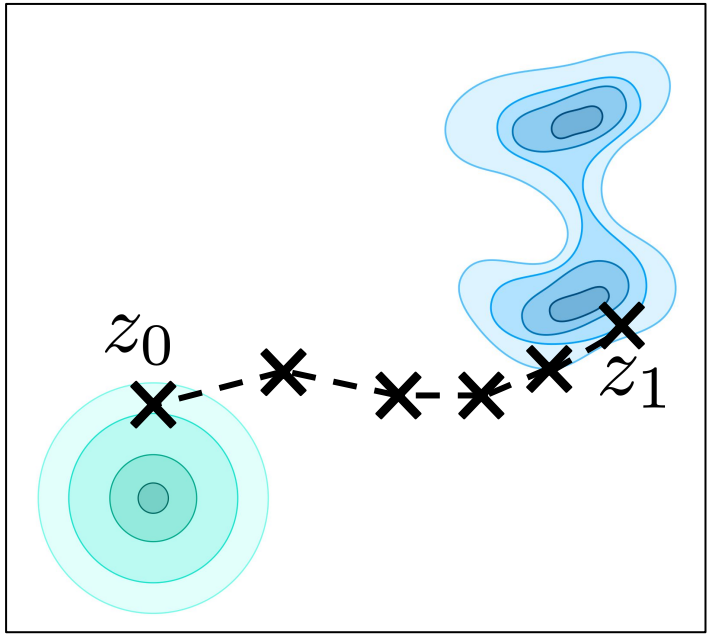


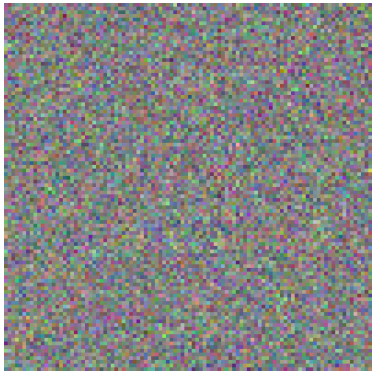
$$z_{t_{i+1}} = z_{t_i} + \underbrace{u_\theta(z_{t_i}, t_i, c)}(t_{i+1} - t_i)$$

Prediction from the DiT

Obtain final latent

Latent space
learned by the VAE

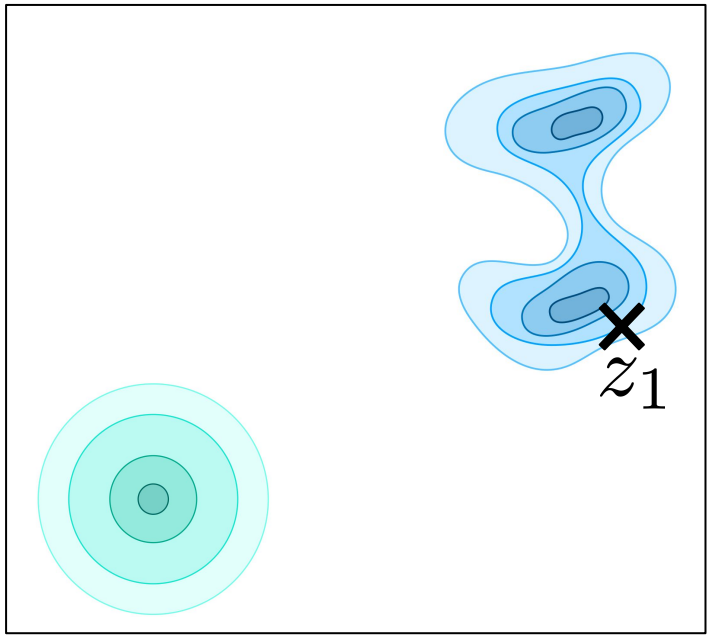




z_1

Decode latent back into pixel space

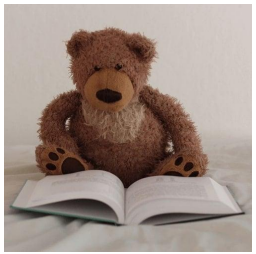
Latent space
learned by the VAE



z_1 → **Decoder** → $x_1 \sim p_\theta(x_1|z_1)$

p_θ

Pixel space



Diffusion & Large Vision Models

Motivation

U-Net

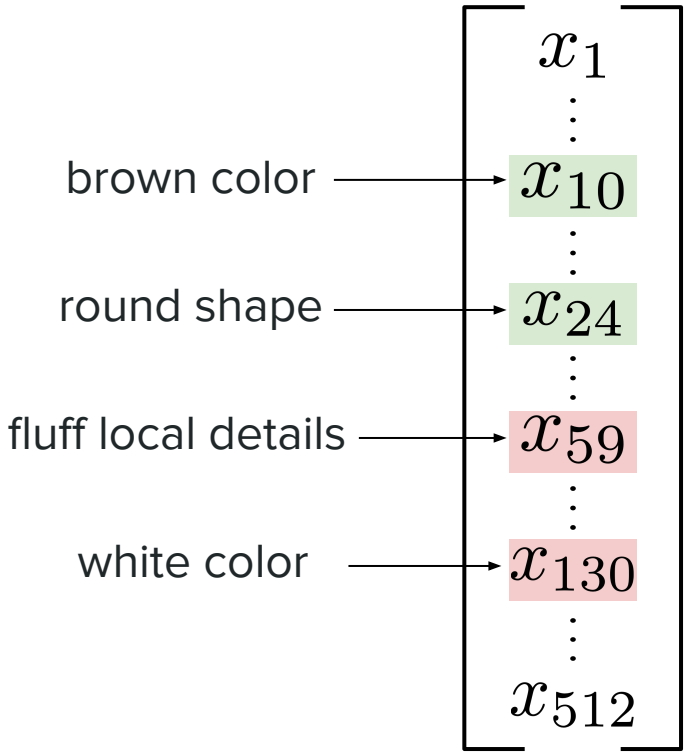
Diffusion Transformer

End-to-end example

Multimodal DiT

Optimizations

Intuition: back to the example we had



Conditions

c `brown fluffy teddy bear`

$t = 0.1$ **a lot of** noise

Need to focus on **global structure** since at the early stage of generation.

Intuition: back to the example we had

Conditions

c `brown fluffy teddy bear`

$t = 0.1$ **a lot of** noise

Need to focus on **global structure** since at the early stage of generation.

Intuition: more subtle text prompt

Conditions

c `brown fluffy teddy bear` **`surrounded by white walls`**

$t = 0.1$ **a lot of** noise

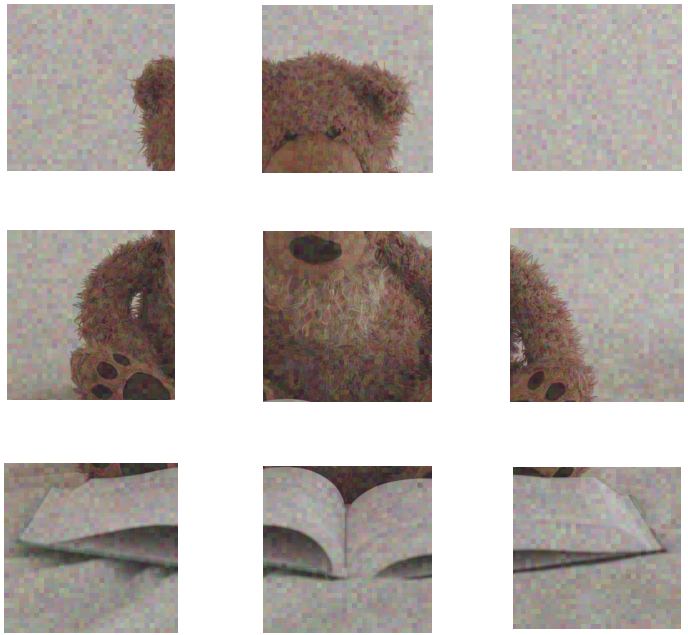
Need to focus on **global structure** since at the early stage of generation.

Intuition: more subtle text prompt

Limitation. All patch latents are subject to same "modulation"

Intuition: more subtle text prompt

Limitation. All patch latents are subject to same "modulation"



Mitigate lack of nuance of previous approach

Strategy.

1. **Keep "modulating"** using the **timestep** embedding

Mitigate lack of nuance of previous approach

Strategy.

1. **Keep "modulating"** using the **timestep** embedding
2. **Do something else** to allow for more complex interaction for the **condition**

Cross-attention	Joint attention

Mitigate lack of nuance of previous approach

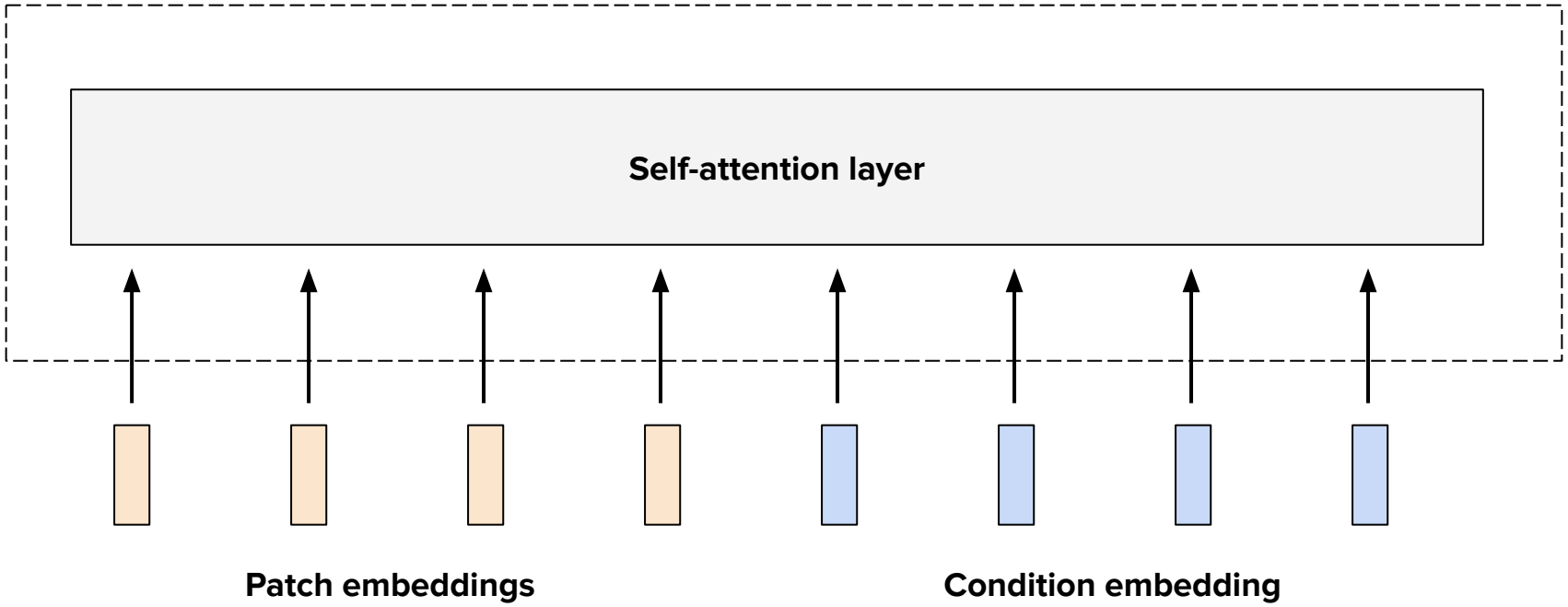
Strategy.

1. **Keep "modulating"** using the **timestep** embedding
2. **Do something else** to allow for more complex interaction for the **condition**

Cross-attention

Joint attention

Intuition behind joint attention



MultiModal-Diffusion Transformer

MM-DiT = **M**ulti**M**odal-**Di**ffusion **T**ransformer

- Term coined in the **Stable Diffusion 3** paper published in 2024
- Relies on **joint attention** of different modalities

Single-stream
Everyone treated equally

Scaling Rectified Flow Transformers for High-Resolution Image Synthesis, Esser et al., 2024.

MultiModal-Diffusion Transformer

MM-DiT = **M**ulti**M**odal-**Di**ffusion **T**ransformer

- Term coined in the **Stable Diffusion 3** paper published in 2024
- Relies on **joint attention** of different modalities

Single-stream
Everyone treated equally

Double-stream
Each modality is in its own stream

Scaling Rectified Flow Transformers for High-Resolution Image Synthesis, Esser et al., 2024.

MultiModal-Diffusion Transformer

MM-DiT = **M**ulti**M**odal-**Di**ffusion **T**ransformer

- Term coined in the **Stable Diffusion 3** paper published in 2024
- Relies on **joint attention** of different modalities

Single-stream	**Hybrid**	Double-stream
Everyone treated equally	Contains both "single-stream" and "double-stream" layer	Each modality is in its own stream

Scaling Rectified Flow Transformers for High-Resolution Image Synthesis, Esser et al., 2024.

"Double-stream" DiT variant: SD3, Qwen-Image

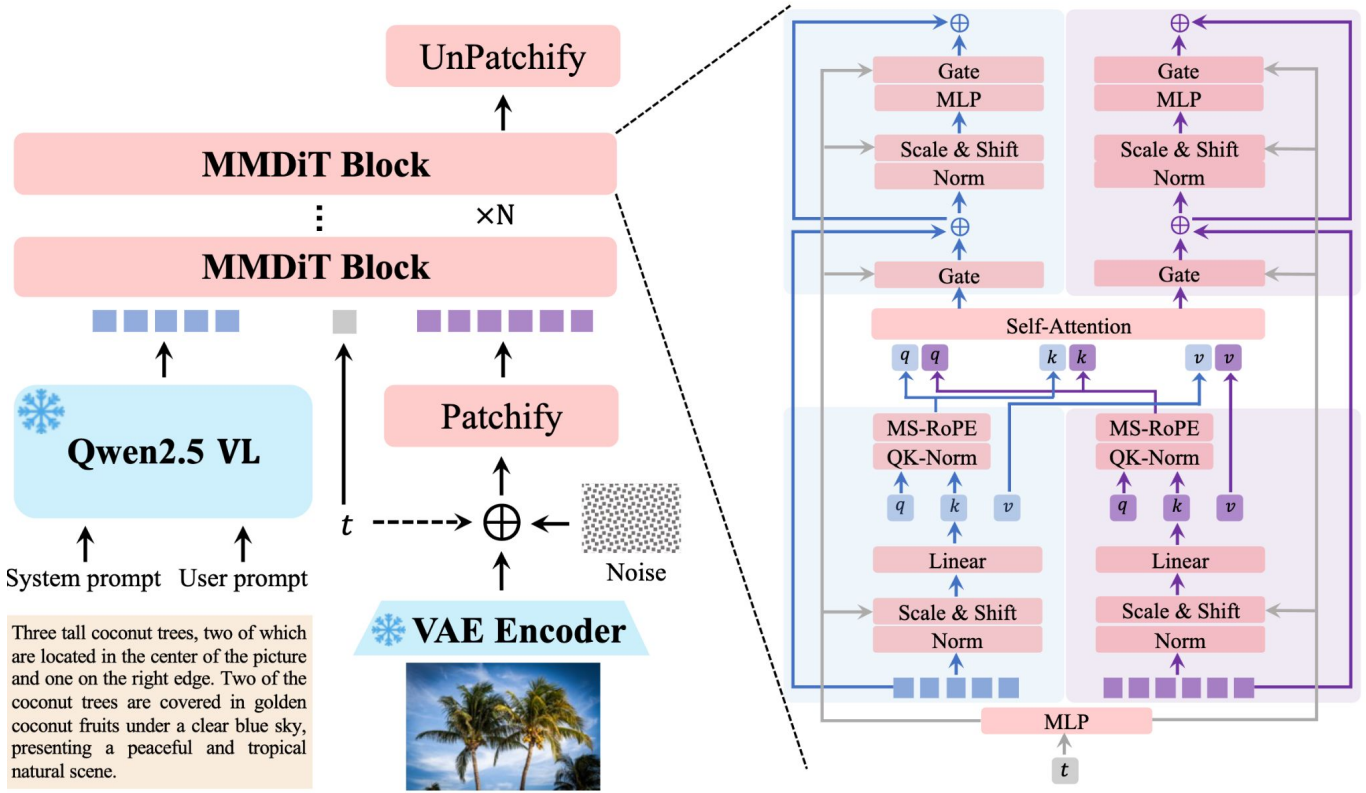


Figure from Qwen-Image Technical Report, Qwen Team, 2025.

"Single-stream" DiT variant: Z-Image

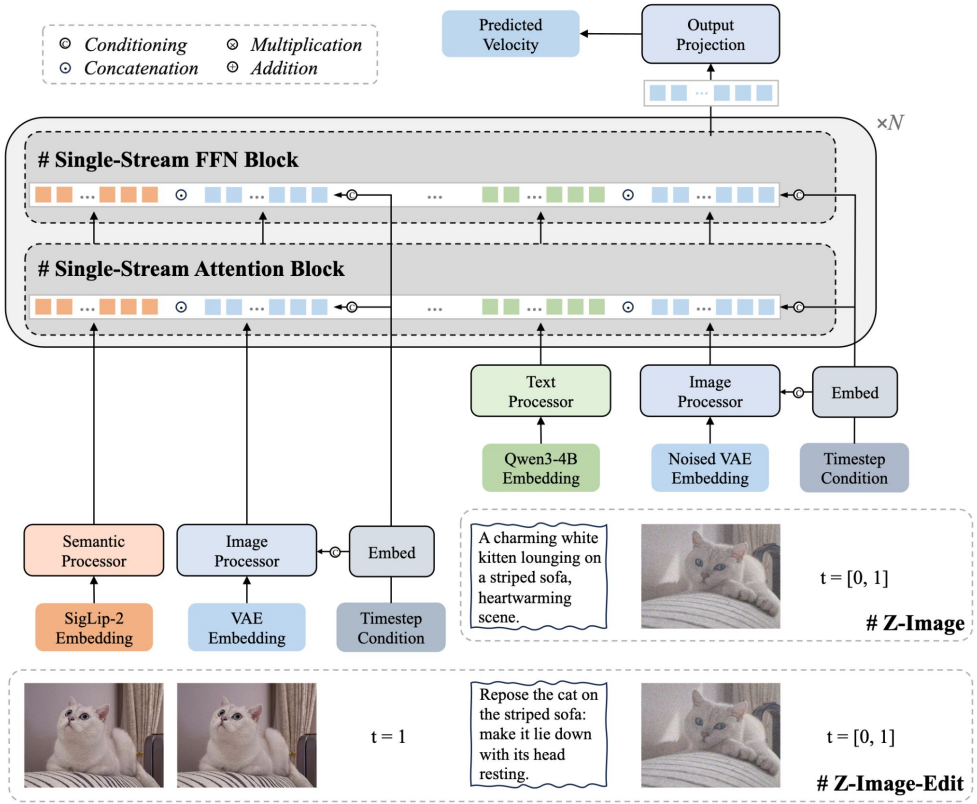


Figure from Z-Image: An Efficient Image Generation Foundation Model with Single-Stream Diffusion Transformer, Z-Image Team, 2025.

Hybrid approach: FLUX.1 Kontext

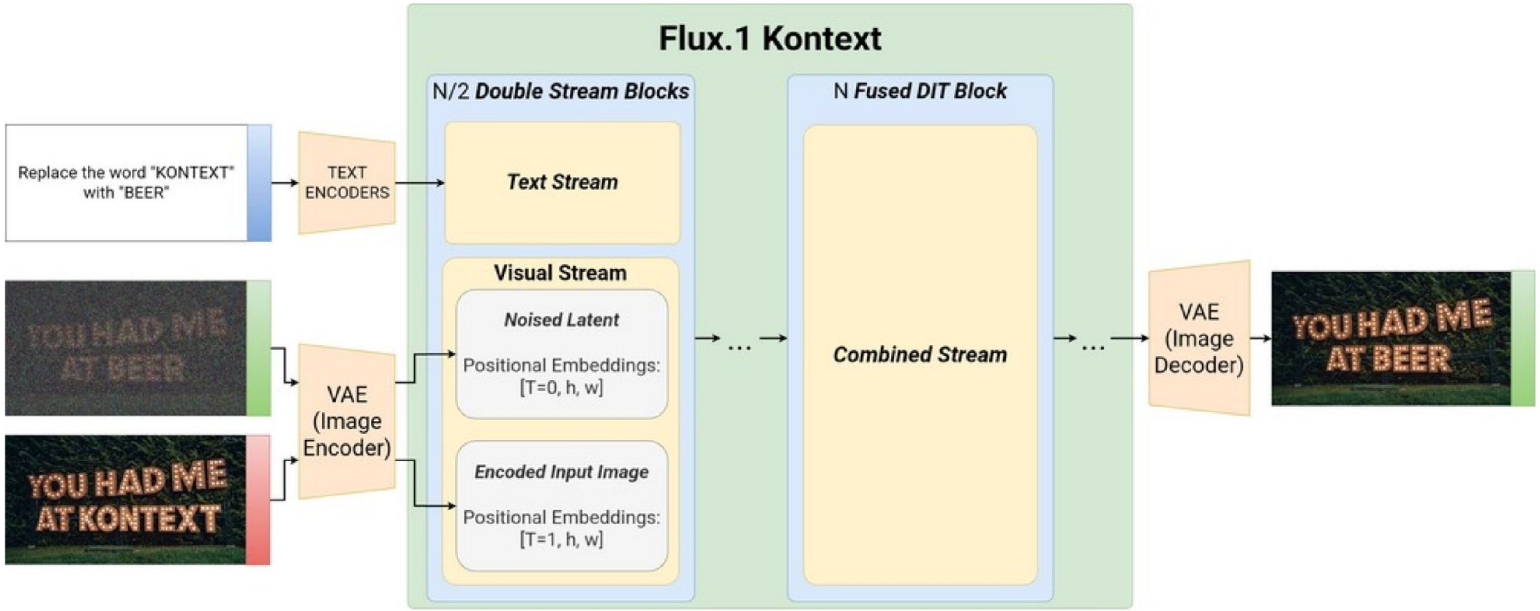
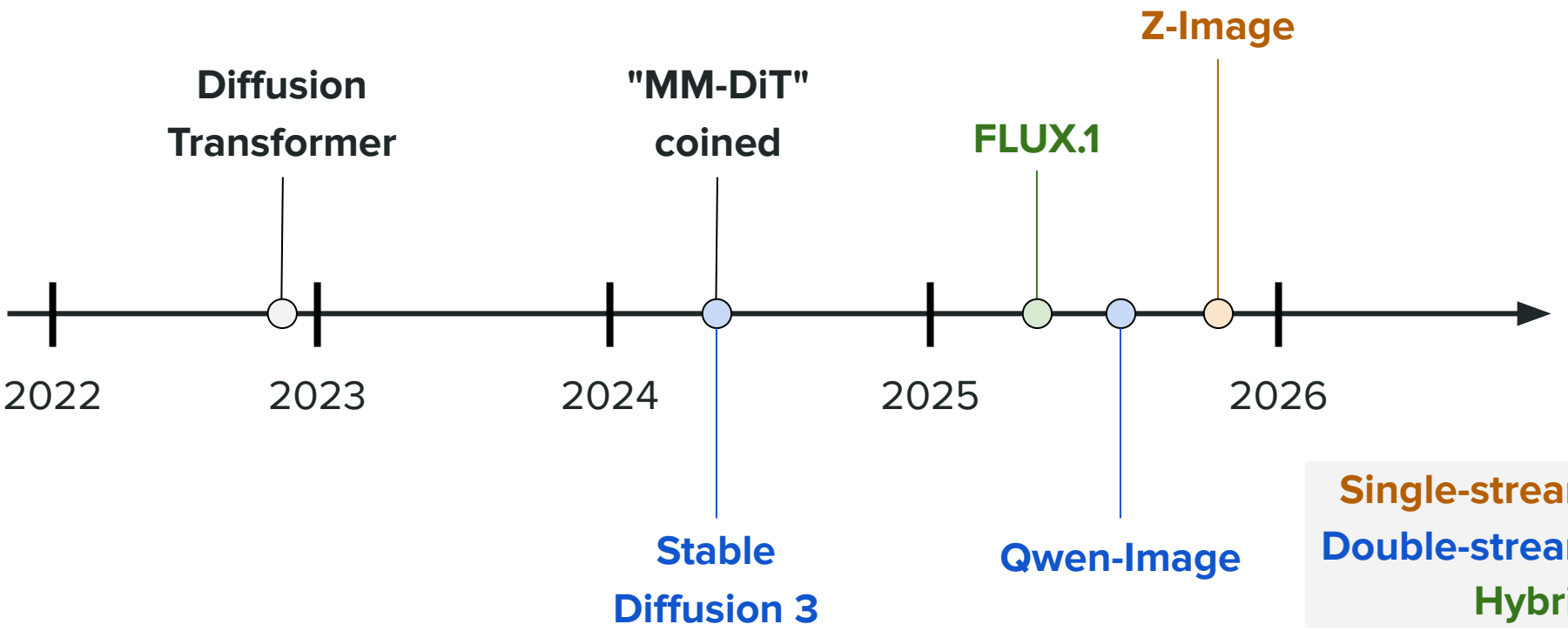


Figure from FLUX.1 Kontext: Flow Matching for In-Context Image Generation and Editing in Latent Space, Black Forest Labs, 2025.

Timeline



Note: This timeline is not meant to be exhaustive in any way.

Diffusion & Large Vision Models

Motivation

U-Net

Diffusion Transformer

End-to-end example

Multimodal DiT

Optimizations

Need for position information

Motivation. Direct links "lose" position info

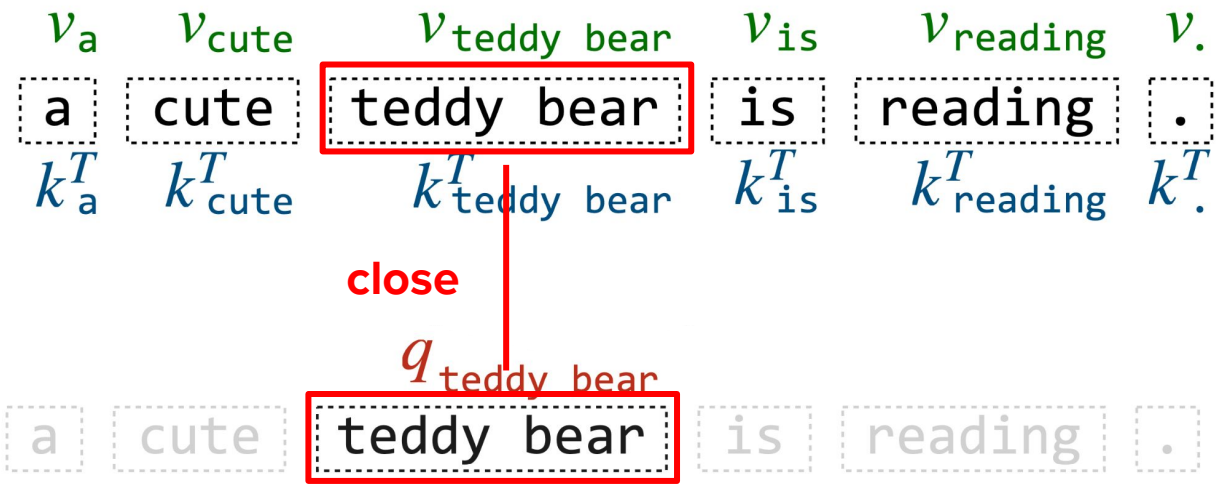


Figure from Super Study Guide: Transformers & Large Language Models, Amidi et al., 2024.

Need for position information

Motivation. Direct links "lose" position info

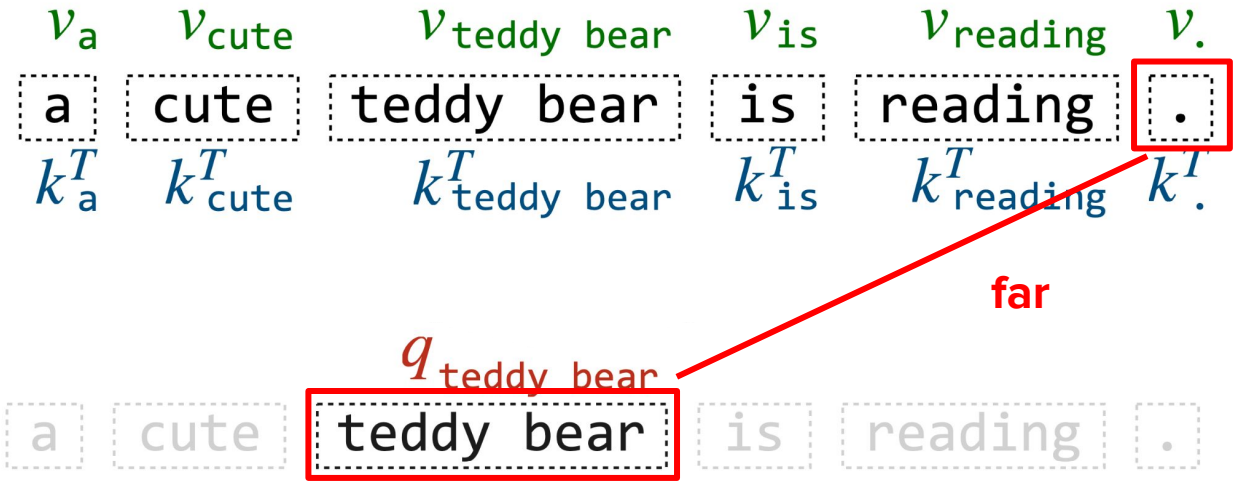


Figure from Super Study Guide: Transformers & Large Language Models, Amidi et al., 2024.

Need for position information

Hope. Dot products convey distance between token representations

$$\text{"}\langle \mathrm{E}_m, \mathrm{E}_n \rangle\text{"} = f(m-n)$$

Absolute positional embeddings

Idea. Add position-specific embedding to token vector

$$\mathrm{E}_m = \mathrm{IE}_m + \mathrm{PE}_m$$

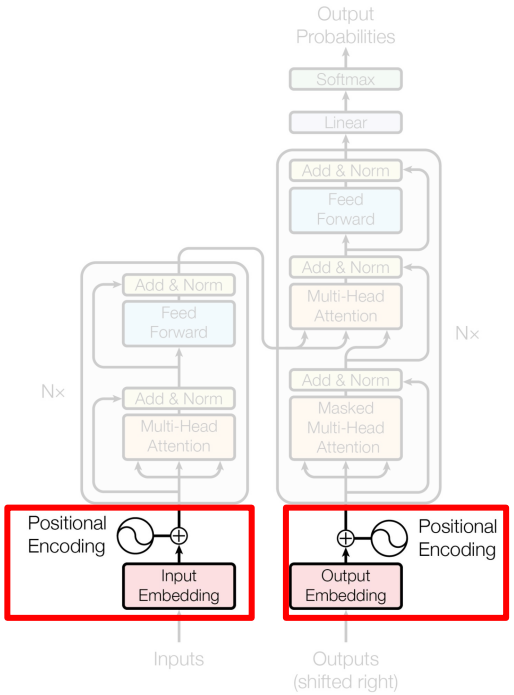


Figure adapted from Attention Is All You Need, Vaswani et al., 2017.

Absolute positional embeddings

Idea. Add position-specific embedding to token vector

$$\mathrm{E}_m = \mathrm{IE}_m + \mathrm{PE}_m$$

$$\langle \mathrm{E}_m, \mathrm{E}_n \rangle = \text{some terms} + \boxed{\langle \mathrm{PE}_m, \mathrm{PE}_n \rangle}$$

What choice for position embeddings?

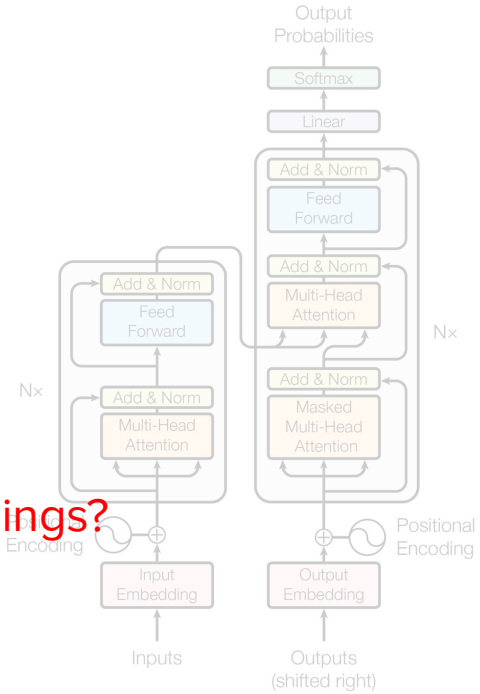
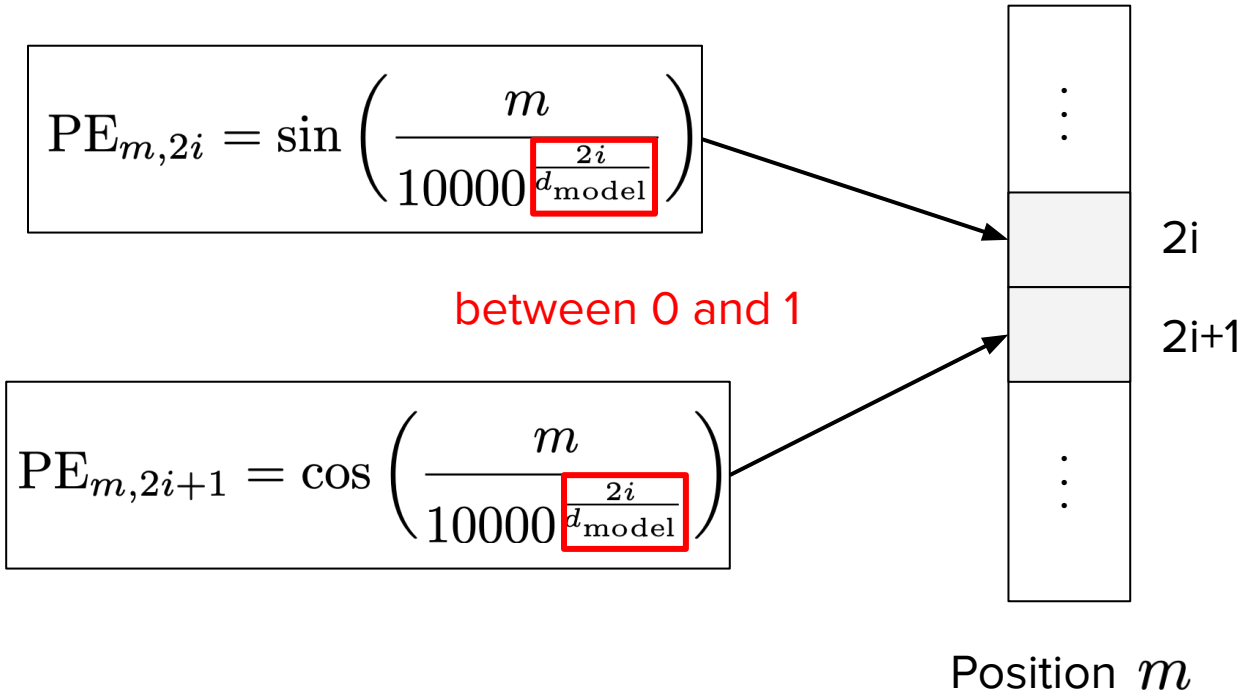


Figure adapted from Attention Is All You Need, Vaswani et al., 2017.

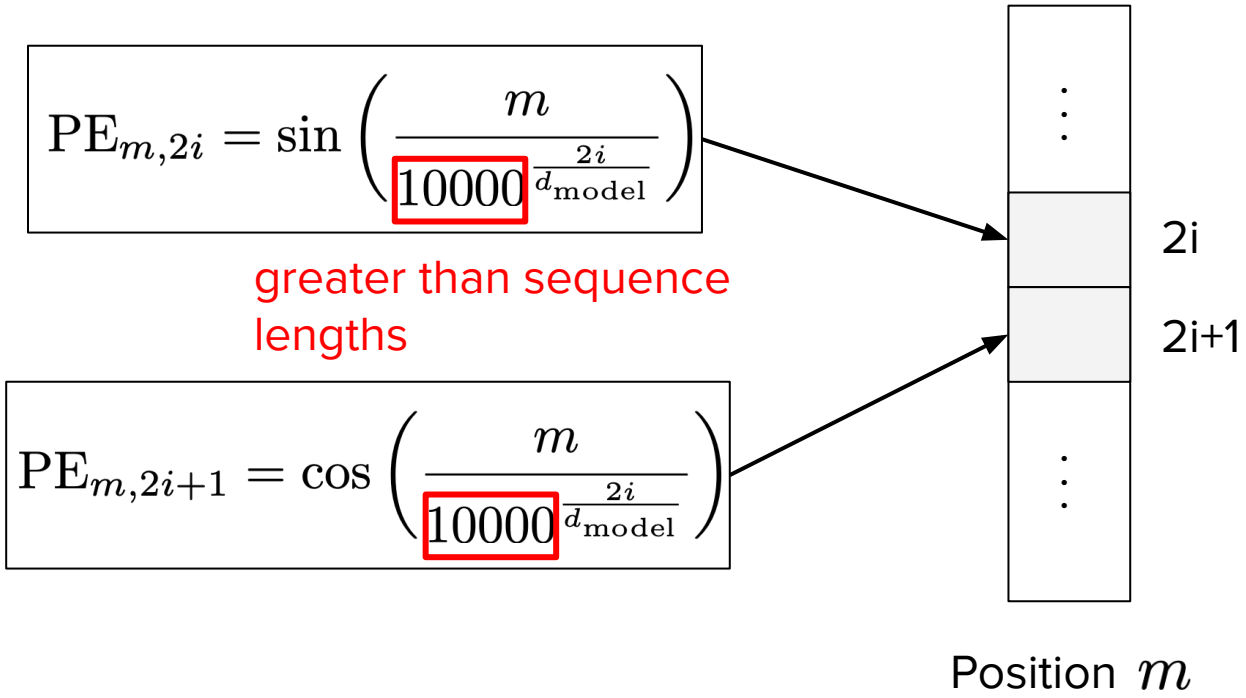
Hardcoded position embeddings

Proposal. A mix of sines and cosines.



Hardcoded position embeddings

Proposal. A mix of sines and cosines.



Hardcoded position embeddings

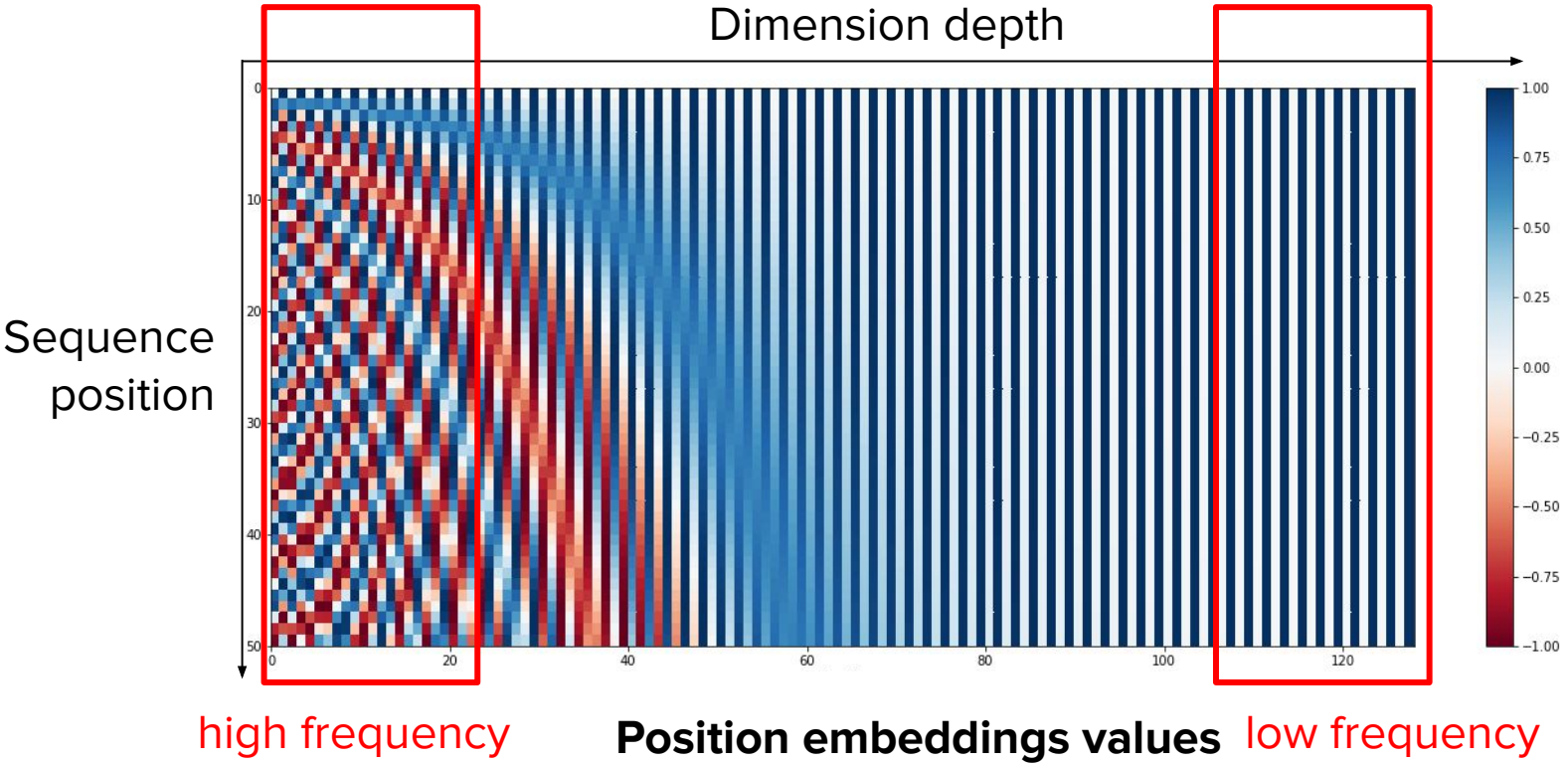
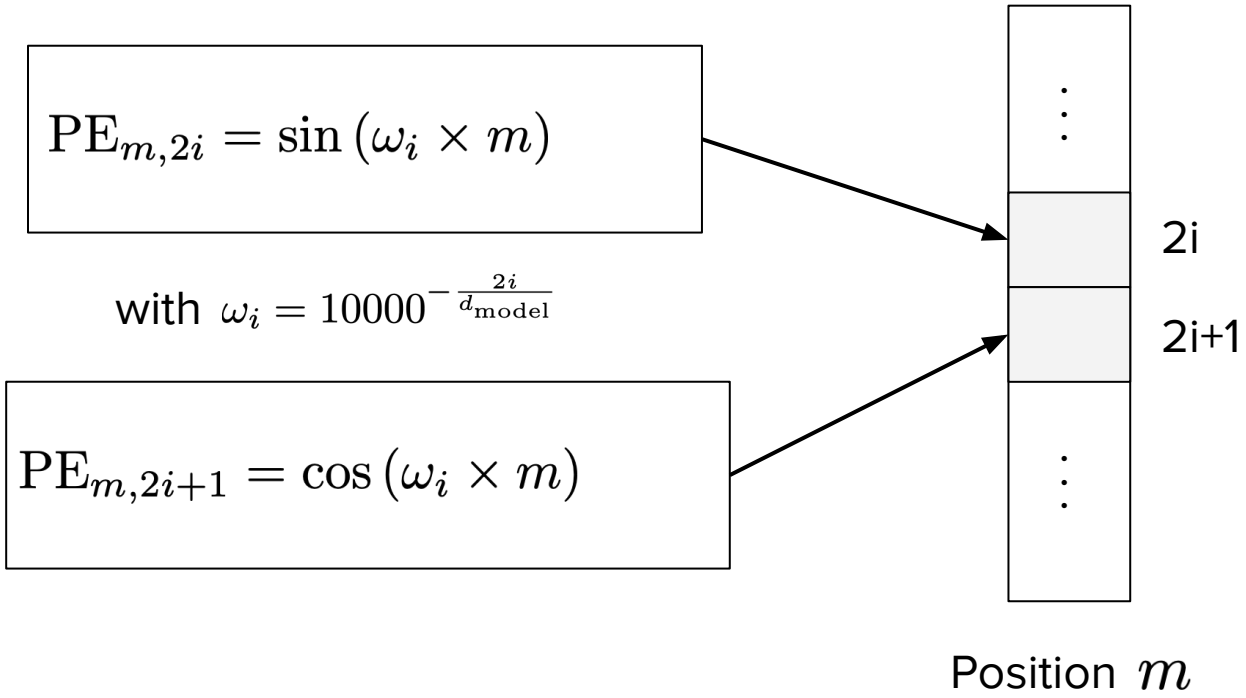


Figure from Transformer Architecture: The Positional Encoding, Kazemnejad, 2019.

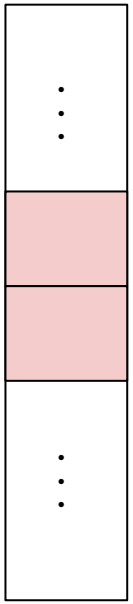
Hardcoded position embeddings



Hardcoded position embeddings

$$\mathrm{PE}_{m,2i} = \sin\left(\omega_i \times m\right)$$

$$\mathrm{PE}_{m,2i+1} = \cos\left(\omega_i \times m\right)$$



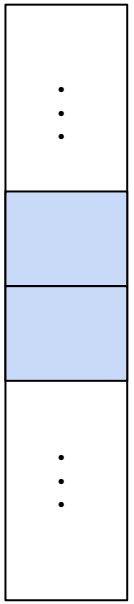
2i

2i+1

Position m

$$\mathrm{PE}_{n,2i} = \sin\left(\omega_i \times n\right)$$

$$\mathrm{PE}_{n,2i+1} = \cos\left(\omega_i \times n\right)$$



2i

2i+1

Position n

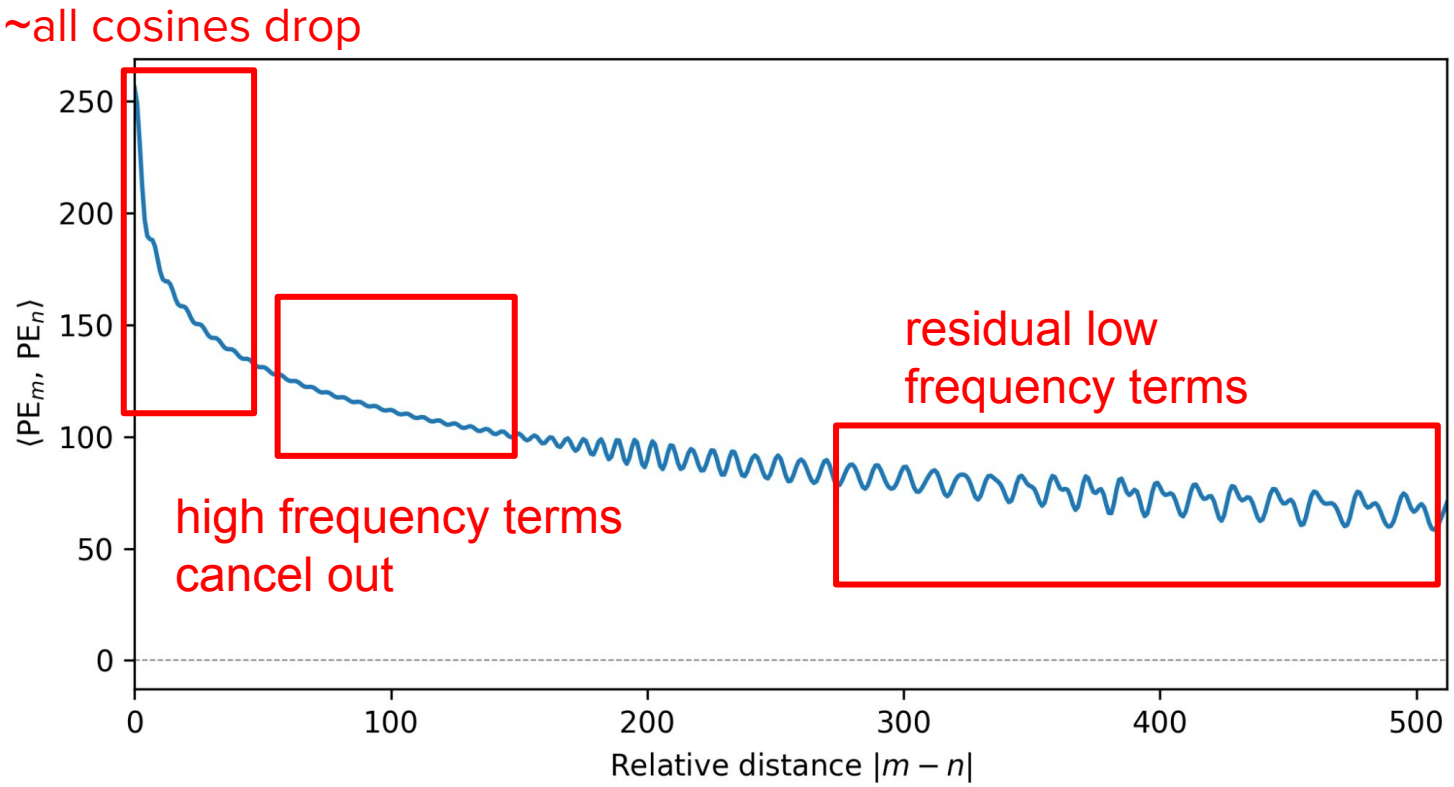
Hardcoded position embeddings

$$\langle \mathrm{PE}_m, \mathrm{PE}_n \rangle = \cdots + \sin(\omega_i m)\sin(\omega_i n) + \cos(\omega_i m)\cos(\omega_i n) + \ldots$$

Property: $\cos(a-b) = \sin(a)\sin(b) + \cos(a)\cos(b)$

$$\langle \mathrm{PE}_m, \mathrm{PE}_n \rangle = \ldots + \cos(\omega_i(m-n)) + \ldots$$

Hardcoded position embeddings



Discussion

Benefits.

- Simple to implement
- Does the job well as a baseline
- Gives every position a direct representational identity

Discussion

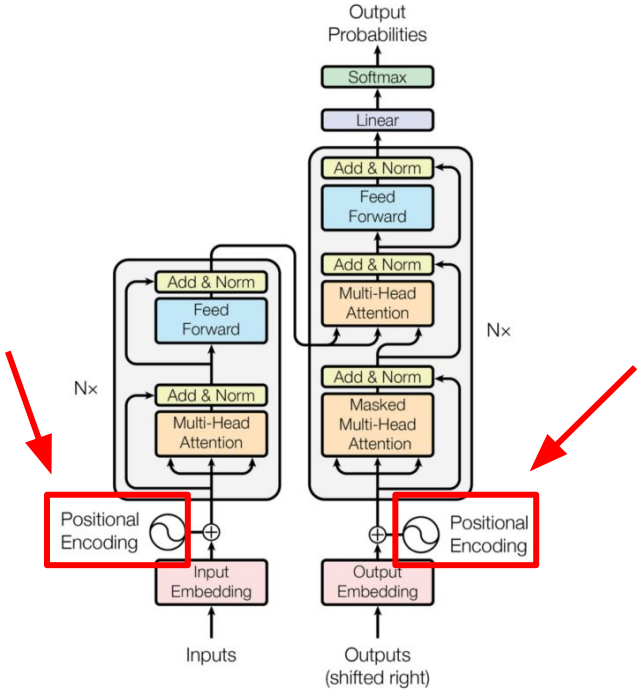
Benefits.

- Simple to implement
- Does the job well as a baseline
- Gives every position a direct representational identity

Limitations.

- Doesn't seem to be injected at the appropriate place
- Side effect: extra interaction terms
- Position embedding also makes it to value embeddings

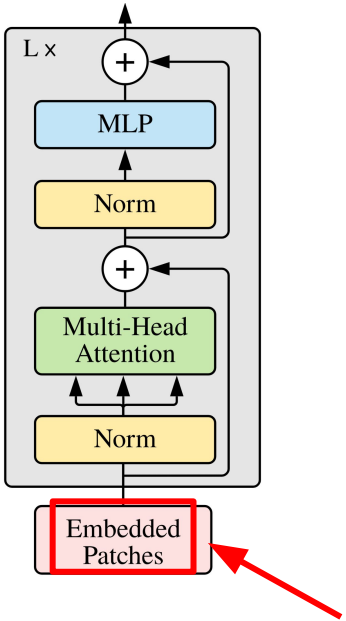
Absolute position embeddings



Original Transformer

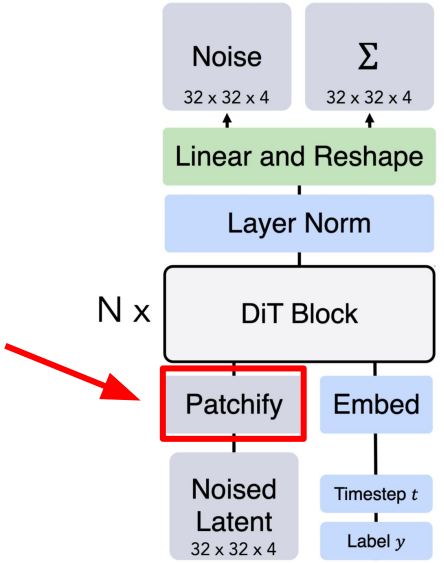
2017

Who used it?



Original ViT

2020



Original DiT

2022

Figures adapted from Attention Is All You Need, Vaswani et al., 2017, An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale, Dosovitskiy et al., 2020, Scalable Diffusion Models with Transformers, Peebles et al., 2022.

From absolute to relative position info

Refinement. Any way to move position information to where it's intuitively needed?

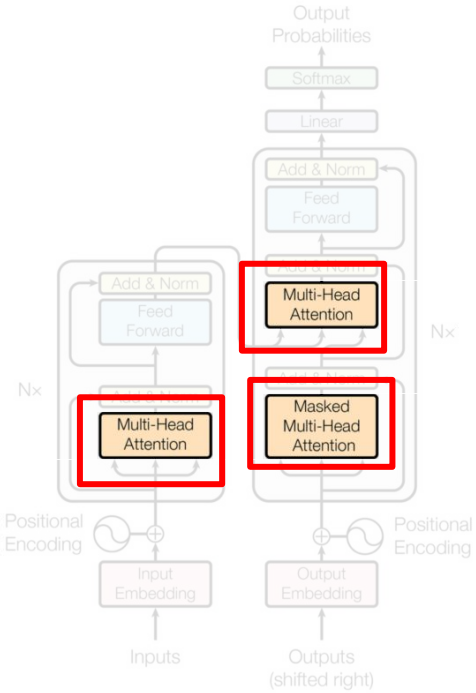
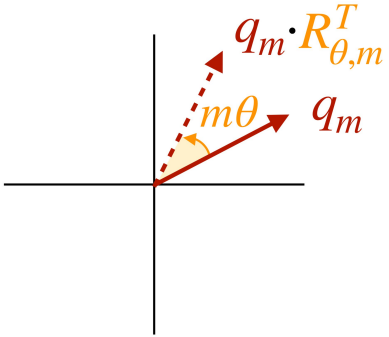


Figure adapted from Attention Is All You Need, Vaswani et al., 2017.

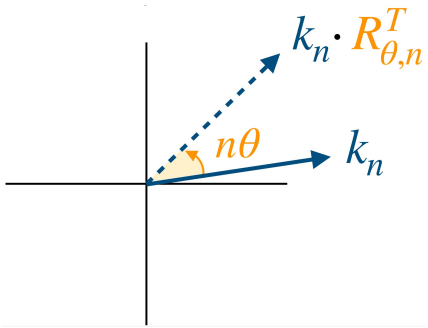
Default choice nowadays: rotations in attention layer

RoPE = **Ro**tary **P**osition **E**mbeddings

Idea. Rotate query and key vectors with rotation matrix $R_{\theta,m} = \begin{pmatrix} \cos(m\theta) & -\sin(m\theta) \\ \sin(m\theta) & \cos(m\theta) \end{pmatrix}$



queries



keys

RoFormer: Enhanced Transformer with Rotary Position Embeddings, Su et al., 2021.
Figure from Super Study Guide: Transformers & Large Language Models, Amidi et al., 2024.

Default choice nowadays: rotations in attention layer

Benefits. Relative distance nicely captured:

$$q_m k_n^T = x_m W_q R_{\theta, n-m} W_k^T x_n^T$$

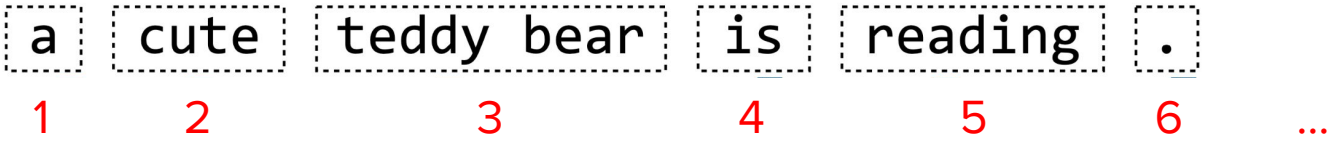
effective rotation by angle

$$\theta(n-m)$$

RoFormer: Enhanced Transformer with Rotary Position Embeddings, Su et al., 2021.

Generalization to 2D?

✅ **Text.**

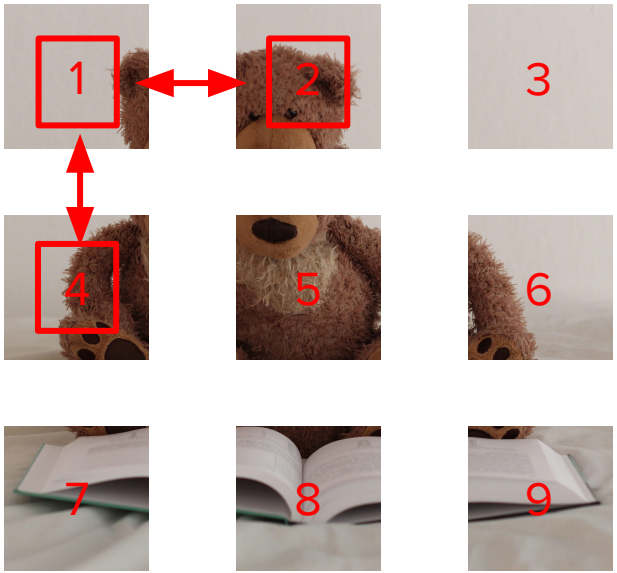


Generalization to 2D?

✅ Text.

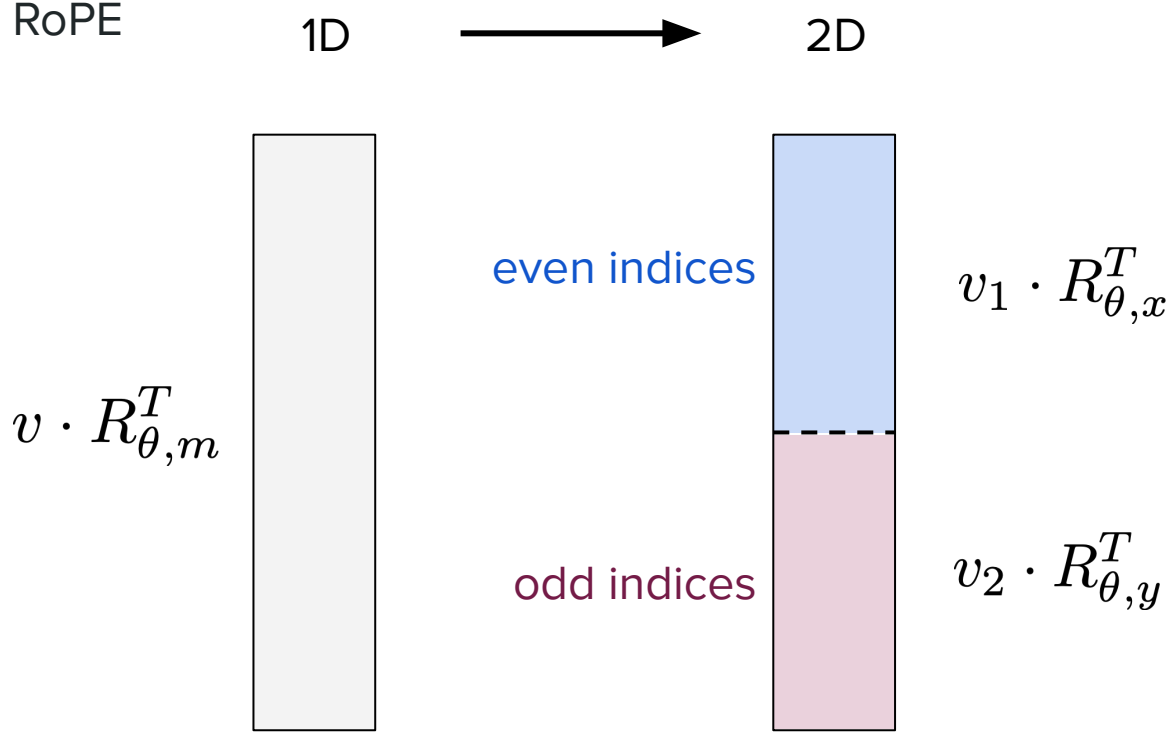
a	cute	teddy bear	is	reading	.	
1	2	3	4	5	6	...

❓ **Images.**



Generalization to 2D?

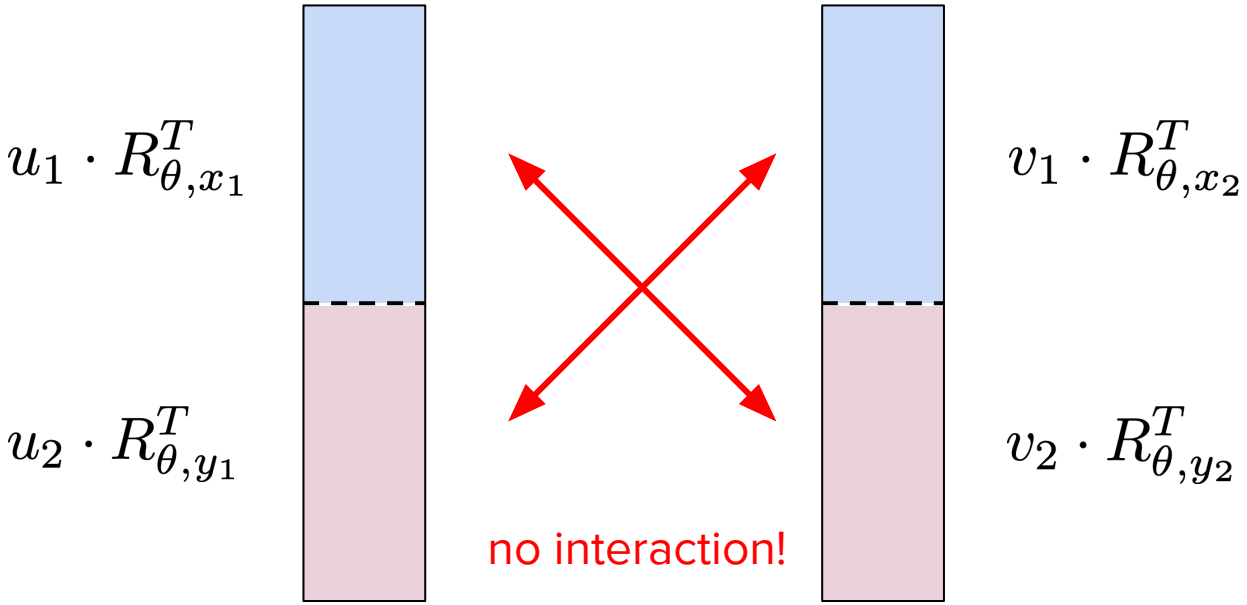
First idea. Axial RoPE



Rotary Position Embedding for Vision Transformer, Heo et al., 2024.

Generalization to 2D?

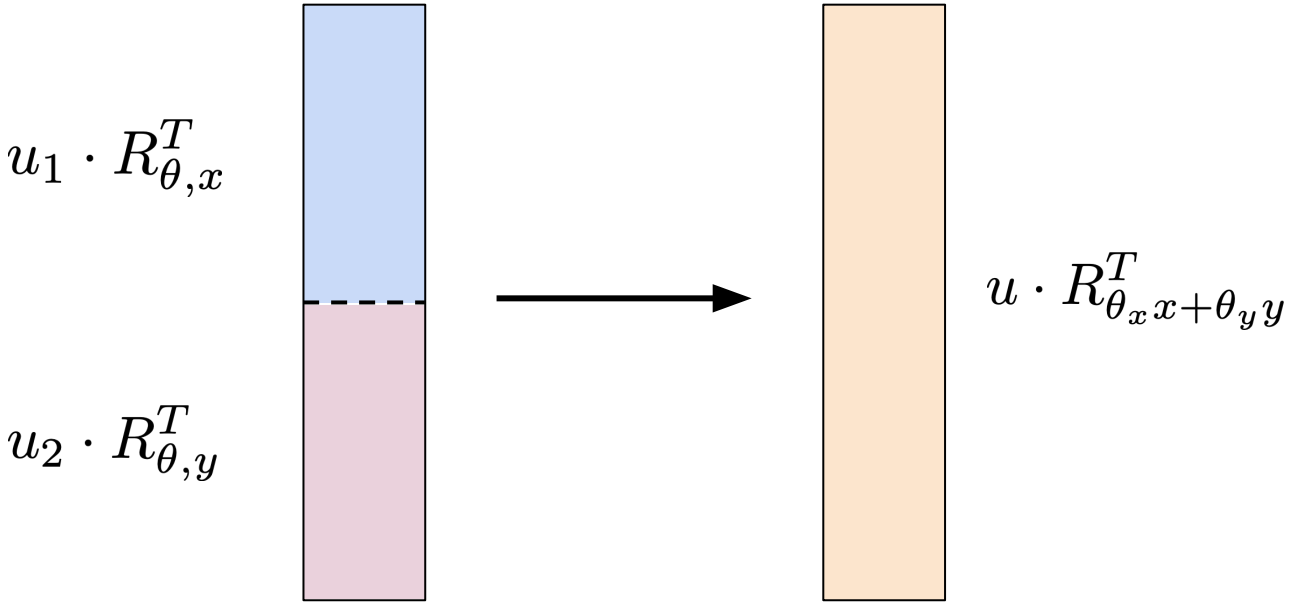
Problems. No cross axes interactions + segregated information is arbitrary



Rotary Position Embedding for Vision Transformer, Heo et al., 2024.

2D RoPE

Proposal. Mix both axes as part of the same rotation matrix



Rotary Position Embedding for Vision Transformer, Heo et al., 2024.

2D RoPE

Consequence. Avoid axes artifacts.

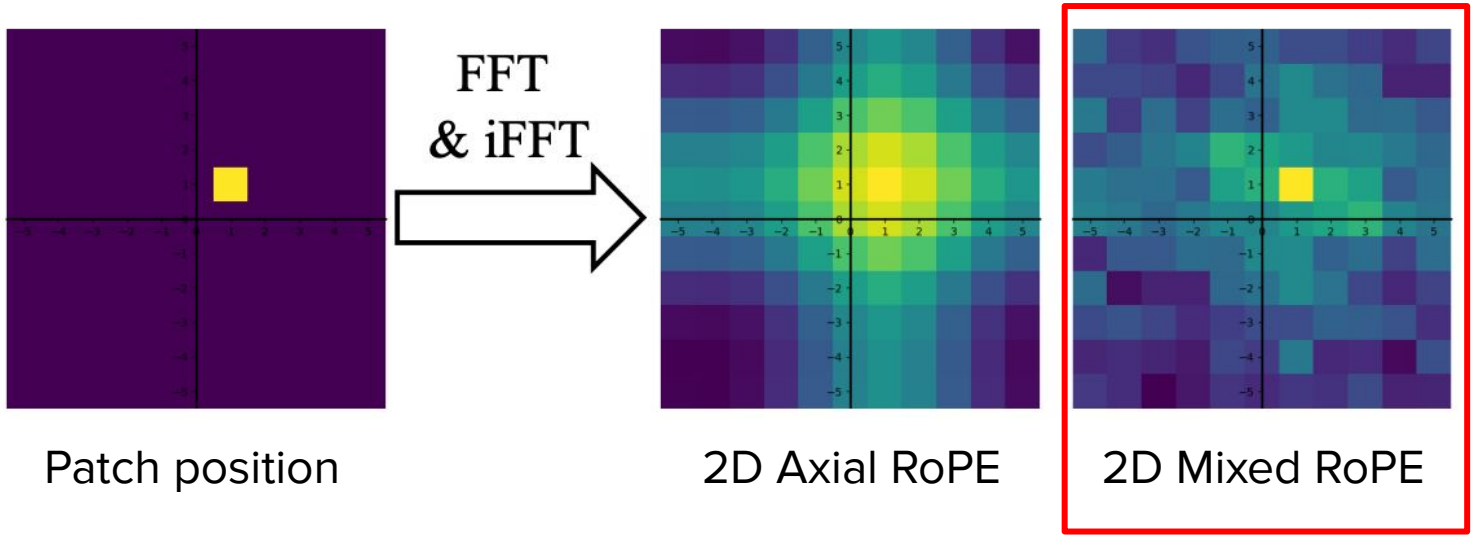
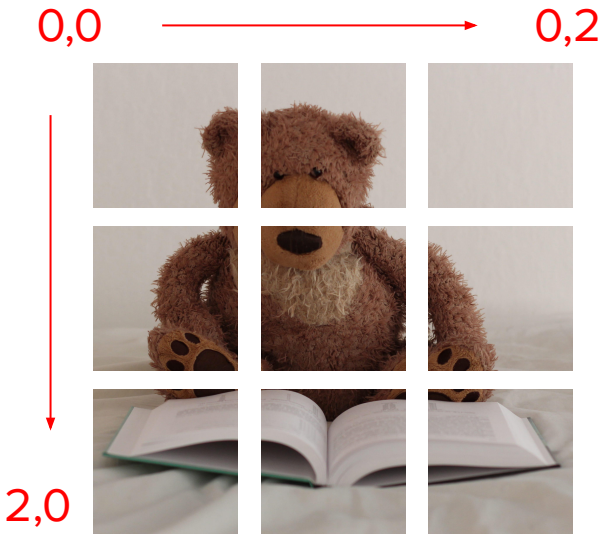


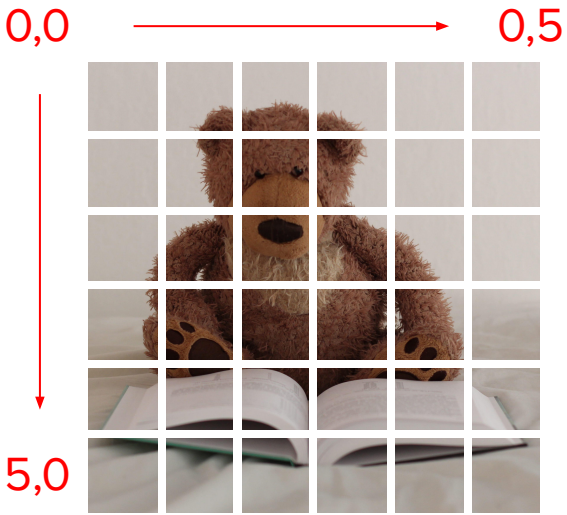
Figure from Rotary Position Embedding for Vision Transformer, Heo et al., 2024.

Scaling RoPE

Common case. Resolution variations?

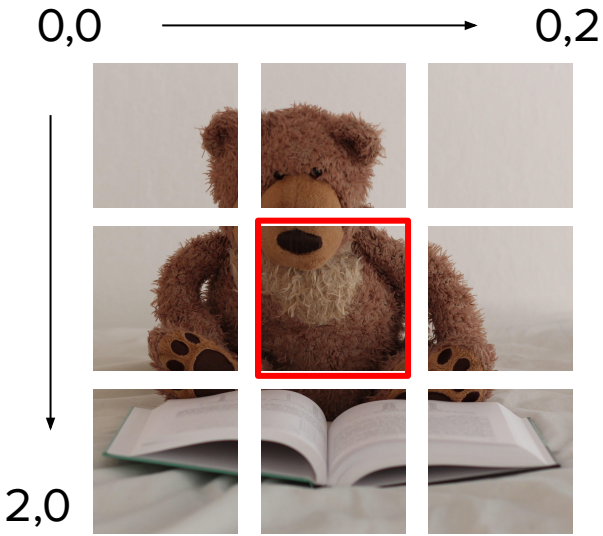


vs



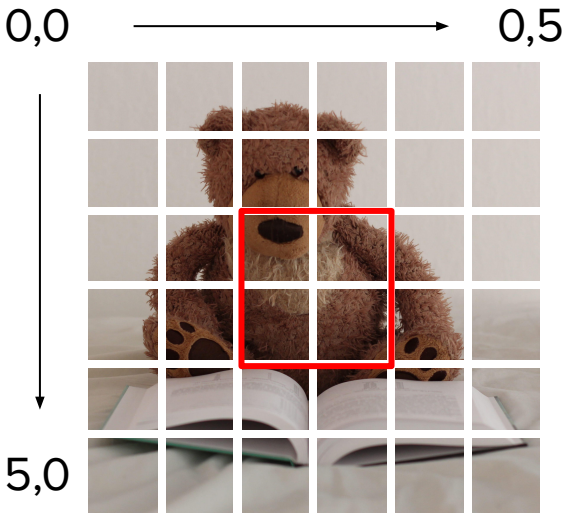
Scaling RoPE

Problem. Spatial meaning also changes.



"center"

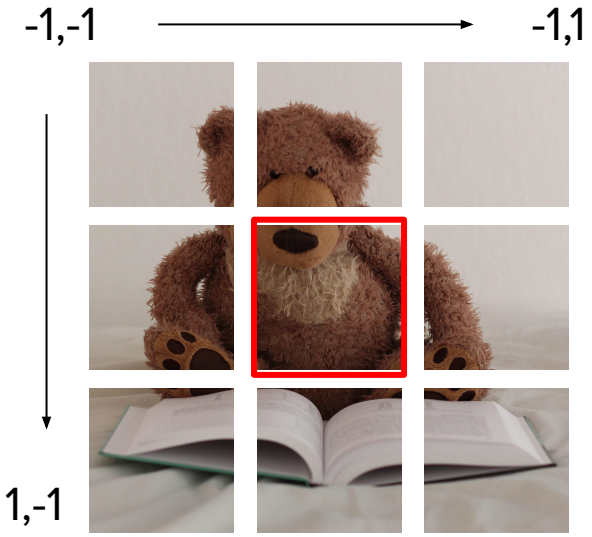
vs



"center"

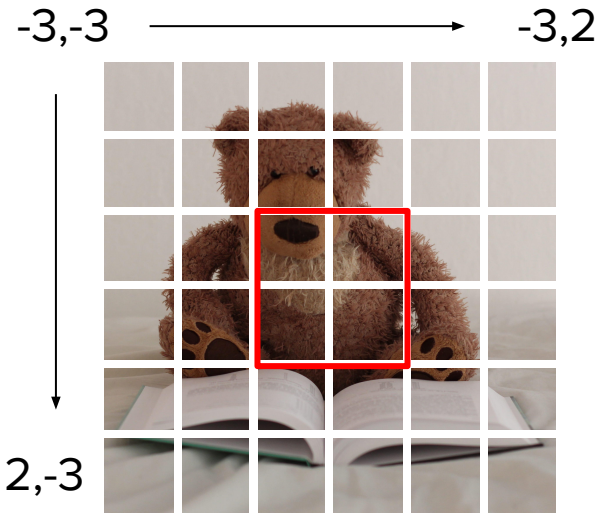
Scaling RoPE

Remedy. Scale to canonical coordinates.



0,0

vs



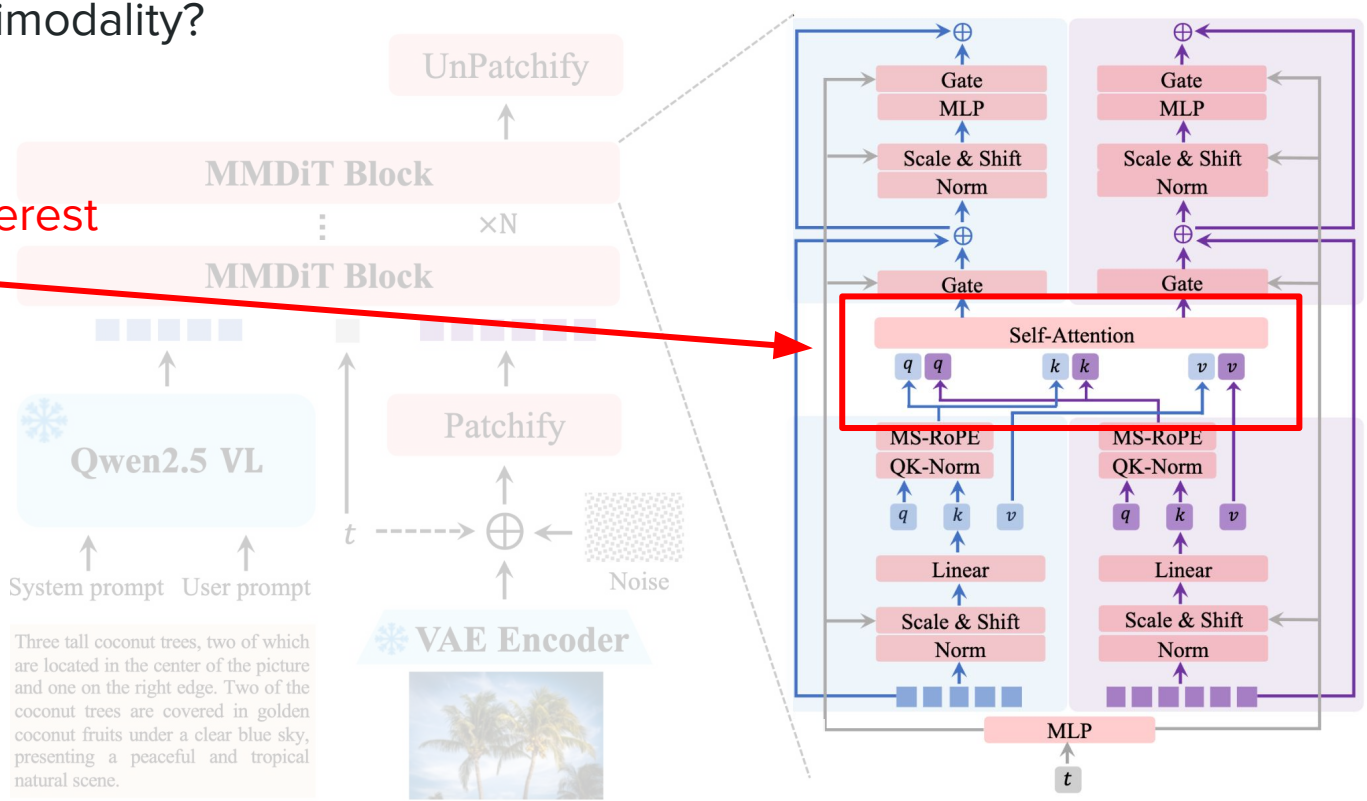
≈ 0,0

Seedream 2.0: A Native Chinese-English Bilingual Image Generation Foundation Model, Gong et al., 2025.

Multimodal RoPE

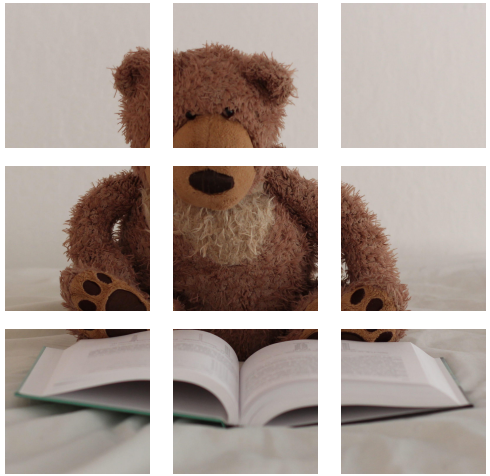
Common case. Multimodality?

Potential area of interest



Qwen-Image Technical Report, Wu et al., 2025.

Multimodal RoPE



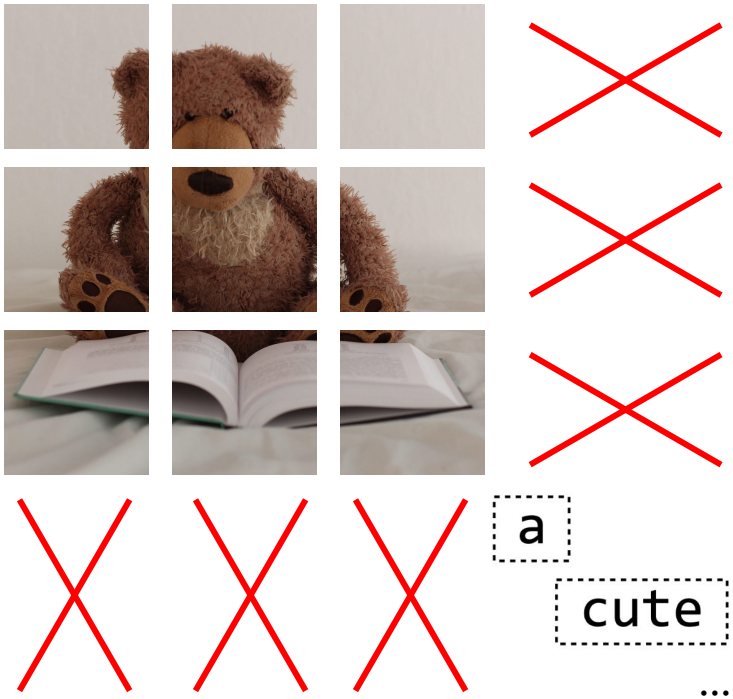
a | cute | teddy bear | is | reading | .

2D ⟷ 1D

How to reconcile the two?

Multimodal RoPE

MSRoPE = **M**ultimodal **S**calable **RoPE**



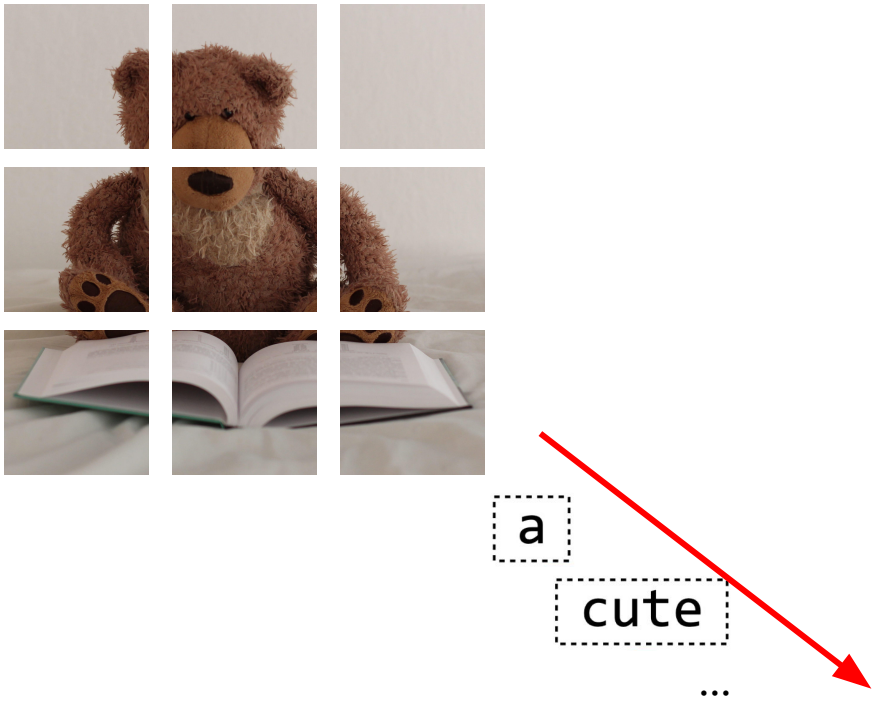
Benefits.

- Image and text can cohabit without interference

Qwen-Image Technical Report, Wu et al., 2025.

Multimodal RoPE

MSRoPE = **M**ultimodal **S**calable **RoPE**



Benefits.

- Image and text can cohabit without interference
- Text is functionally equivalent to 1D RoPE

Qwen-Image Technical Report, Wu et al., 2025.

Conclusion

Embedding positions = open problem.

- Many variations out there
- Dust hasn't settled yet
- Trade-offs

Thank you for your attention!