



10-423/10-623/10-723 Generative AI

Machine Learning Department  
School of Computer Science  
Carnegie Mellon University

# **Latent Diffusion Model + Diffusion Transformer + Prompt to Prompt**

Matt Gormley & Aran Nayebi

Lecture 14

Oct. 20, 2025

# Reminders

- **Homework 3: Applying and Adapting LLMs**
  - Out: Sat, Oct 4
  - Due: Thu, Oct 23 at 11:59pm
- **Quiz 4**
  - In-class, Mon, Oct 27
  - lectures 12 (only the text-to-image topics) – 15
- **Homework 4: Multimodal Foundation Models**
  - Out: Thu, Oct 23
  - Due: Mon, Nov 3 at 11:59pm

# **CONDITIONAL IMAGE GENERATION**

# Image Generation

- Class-conditional generation
- Super resolution
- Image Editing
- Style transfer
- Text-to-image (TTI) generation



Figure from Razavi et al. (2019)

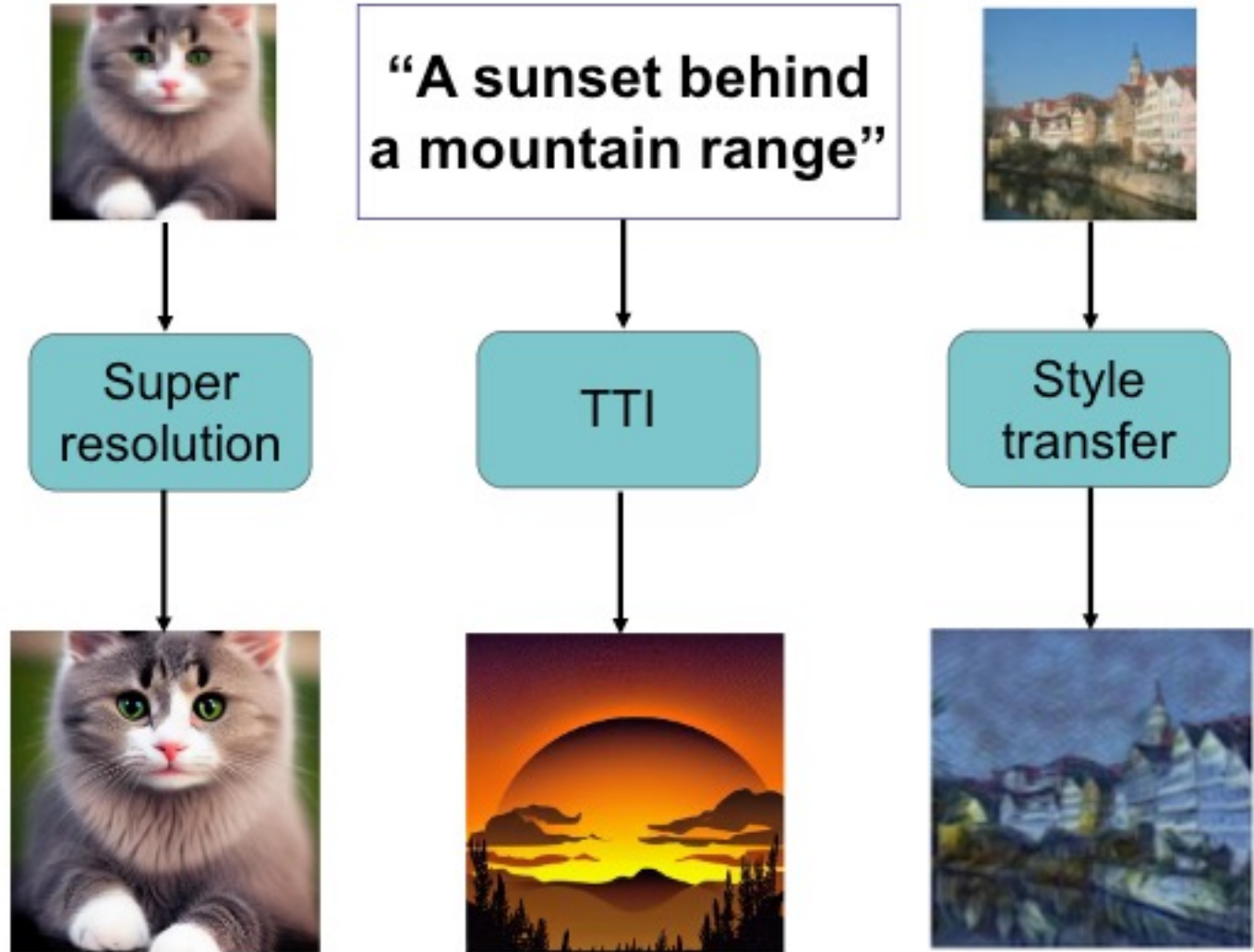
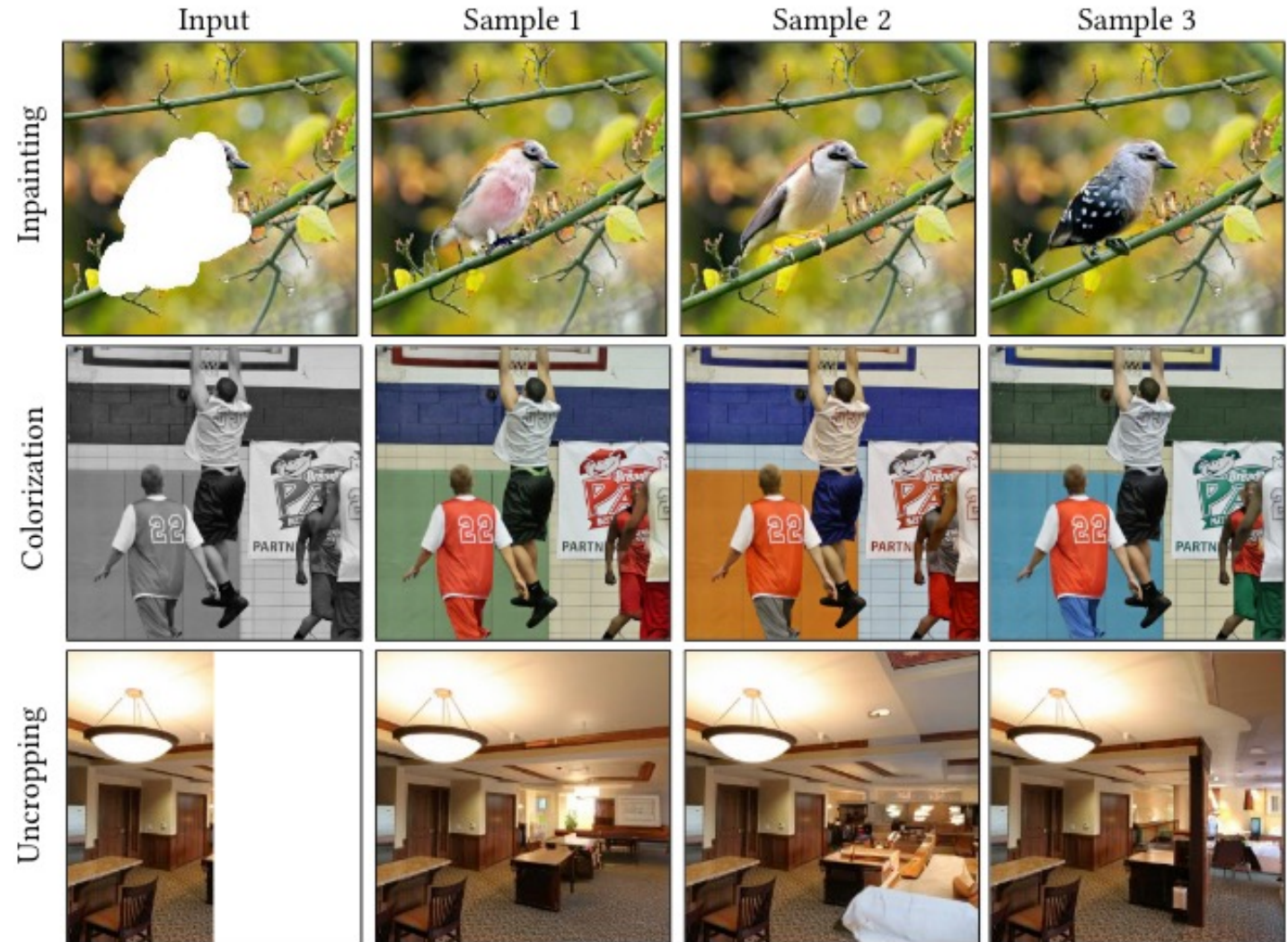


Figure from Bie et al. (2023)

# Image Editing

A variety of tasks involve automatic editing of an image:

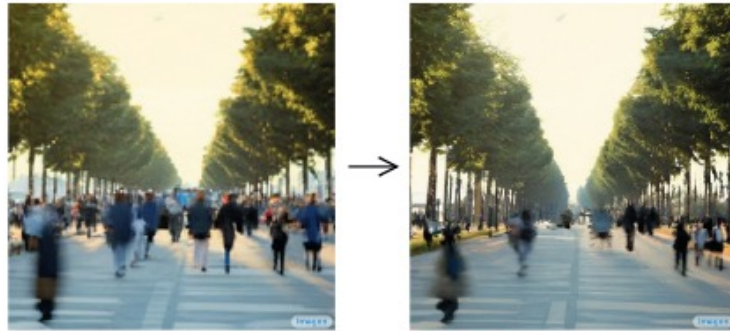
- **Inpainting** fills in the (pre-specified) missing pixels
- **Colorization** restores color to a greyscale image
- **Uncropping** creates a photo-realistic reconstruction of a missing side of an image



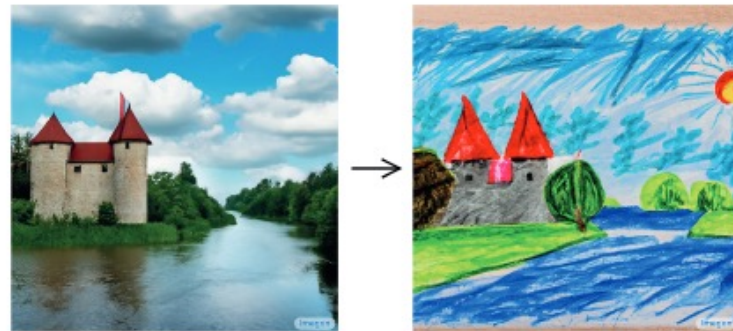
# Editing Images with Text

prompt-to-prompt can edit one generated image simply by adjusting the prompt

down-weight existing descriptor in the prompt



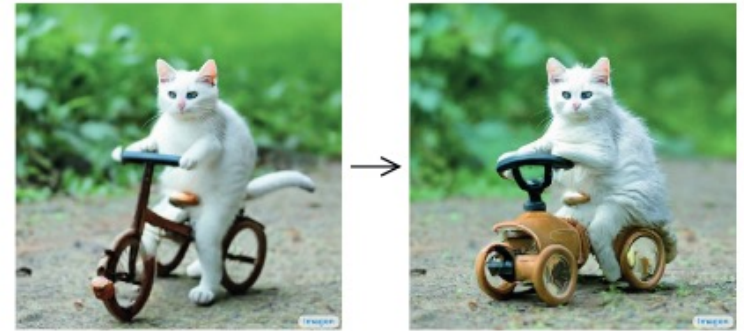
"The boulevards are crowded today."



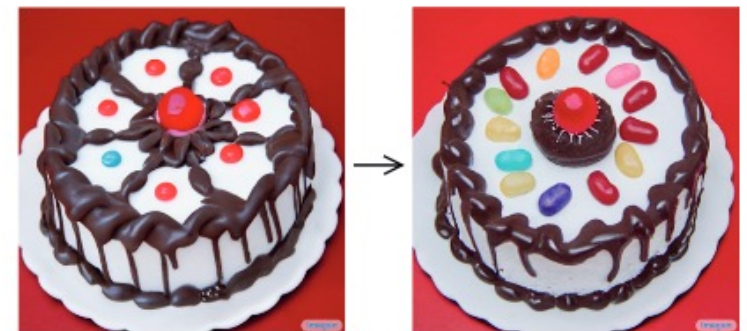
"Children drawing of a castle next to a river."

phrase insertion for style change

swap one word for another



"Photo of a cat riding on a bicycle."



"a cake with decorations."

jelly beans

phrase insertion for content change

# **LATENT DIFFUSION MODEL (LDM)**

# Latent Diffusion Model

## Motivation:

- diffusion models typically operate in pixel space
- yet, training typically takes hundreds of GPU days
  - 150 – 1000 V100 days [Guided Diffusion] (Dhariwal & Nichol, 2021)
  - 256 TPU-v4s for 4 days = 1000 TPU days [Imagen] (Sharia et al., 2022)
- inference is also slow
  - 50k samples in 5 days on A100 GPU [Guided Diffusion] (Dhariwal & Nichol, 2021)
  - 15 seconds per image

## Key Idea:

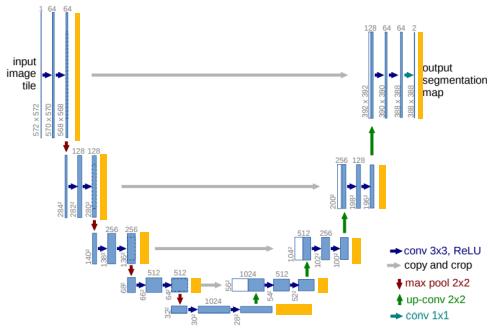
- train an autoencoder (i.e. encoder-decoder model) that learns an efficient latent space that is perceptually equivalent to the data space
- keeping the autoencoder fixed, train a diffusion model on the latent representations of real images  $z_0 = \text{encoder}(x)$ 
  - forward model: latent representation  $z_0 \rightarrow$  noise  $z_T$
  - reverse model: noise  $z_T \rightarrow$  latent representation  $z_0$
- to generate an image:
  - sample noise  $z_T$
  - apply reverse diffusion model to obtain a latent representation  $z_0$
  - decode the latent representation to an image  $x$
- condition on prompt via cross attention in latent space



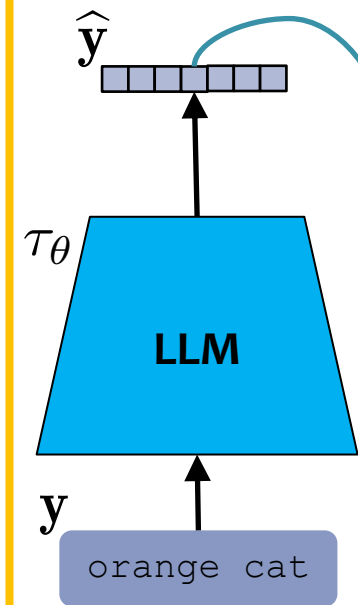
**Latent Diffusion Model**

# Latent Diffusion Model (LDM)

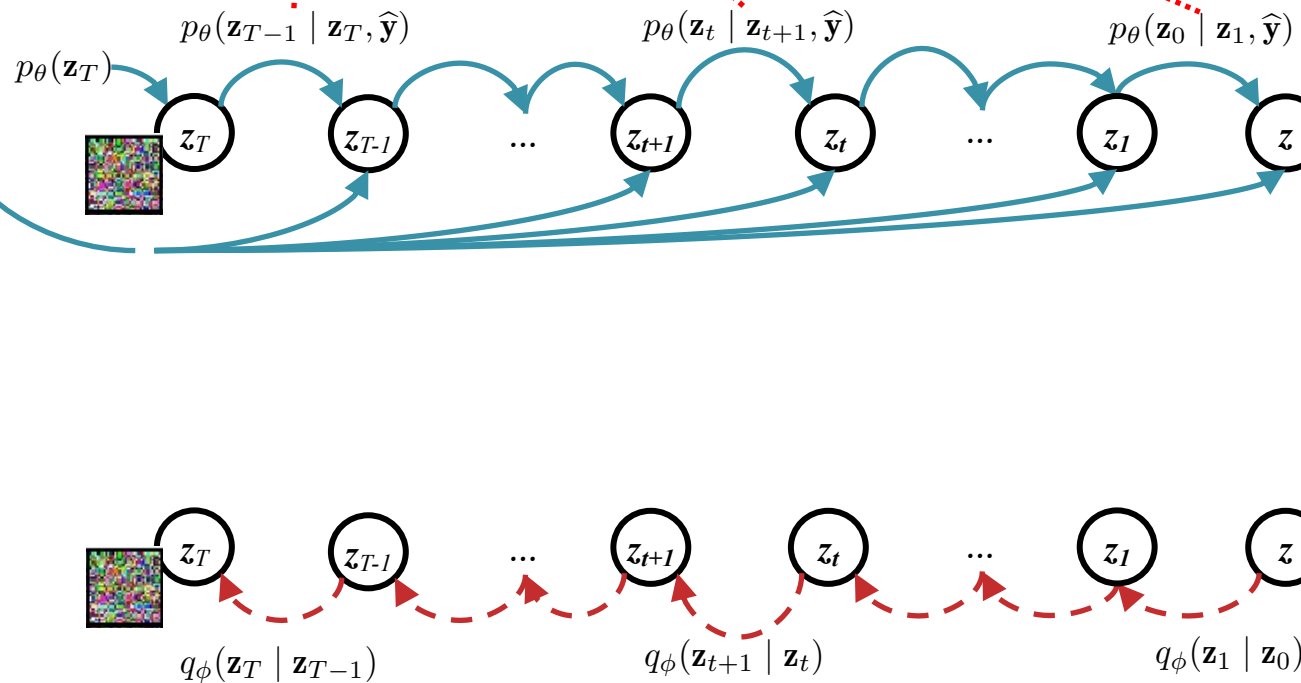
UNet w/cross  
attention



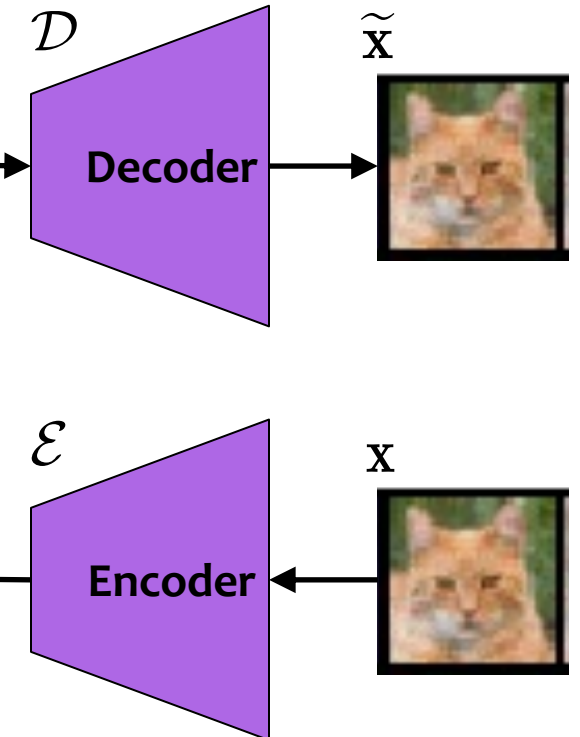
prompt space



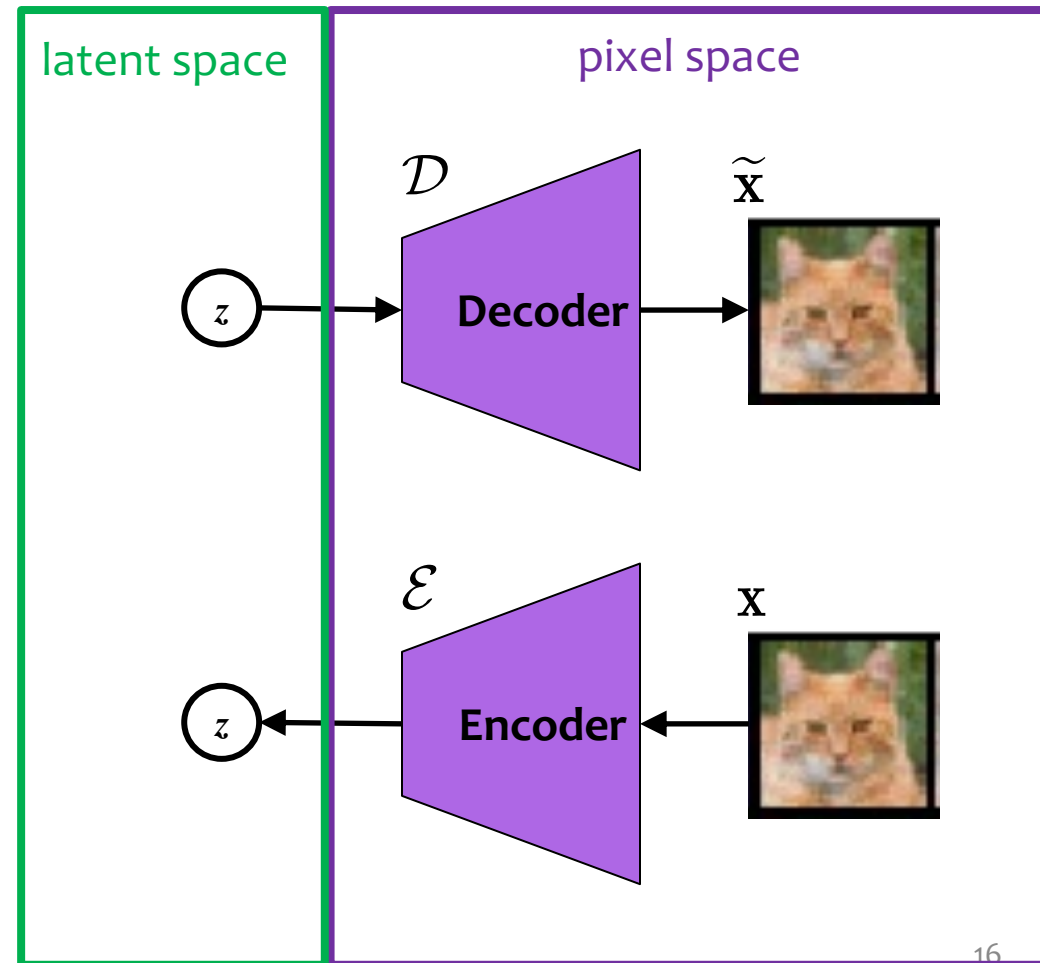
latent space



pixel space

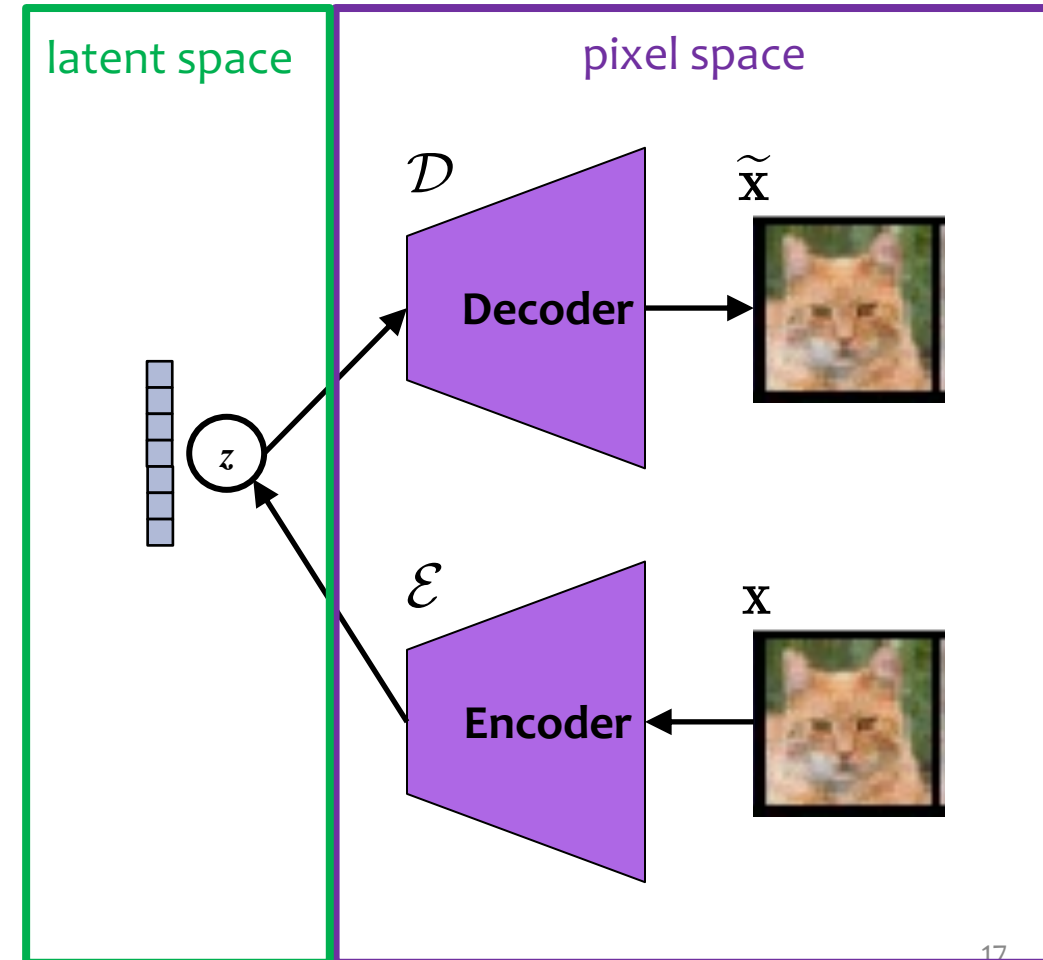


# LDM: Autoencoder



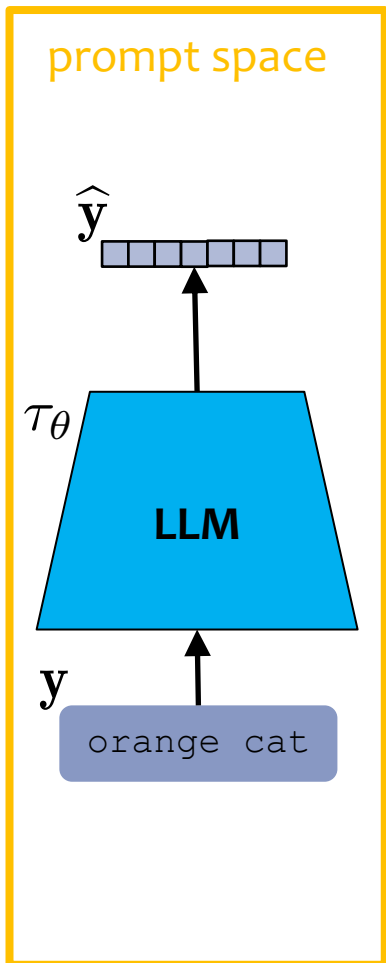
# LDM: Autoencoder

- The autoencoder is chosen so that it can project high dimensional images (e.g. 1024x1024) down to low dimensional **latent space** and **faithfully** project back up to **pixel space**
- The original LDM paper considers two options:
  1. a VAE-like model (regularizes the noise towards a Gaussian)
  2. a VQGAN (performs vector quantization in the decoder; i.e., it uses a discrete codebook)
- This model is trained ahead of time just on raw images (no text prompts) and then frozen
- The frozen encoder-decoder can be reused for all subsequent LDM training



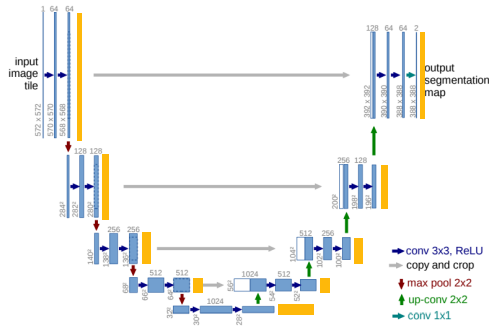
# LDM: the Prompt Model

- The prompt model is just a Transformer LM
- We learn its parameters alongside the diffusion model
- The goal is to build up good representations of the text prompts such that they inform the latent diffusion process

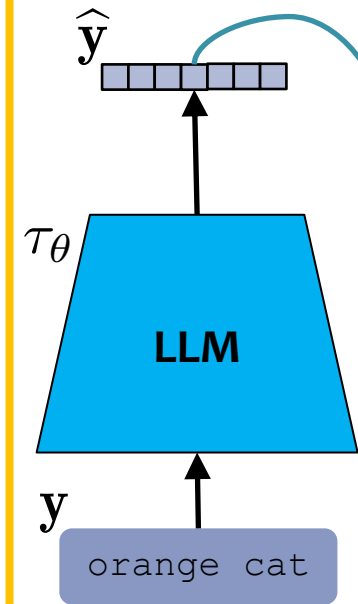


# LDM: with DDPM

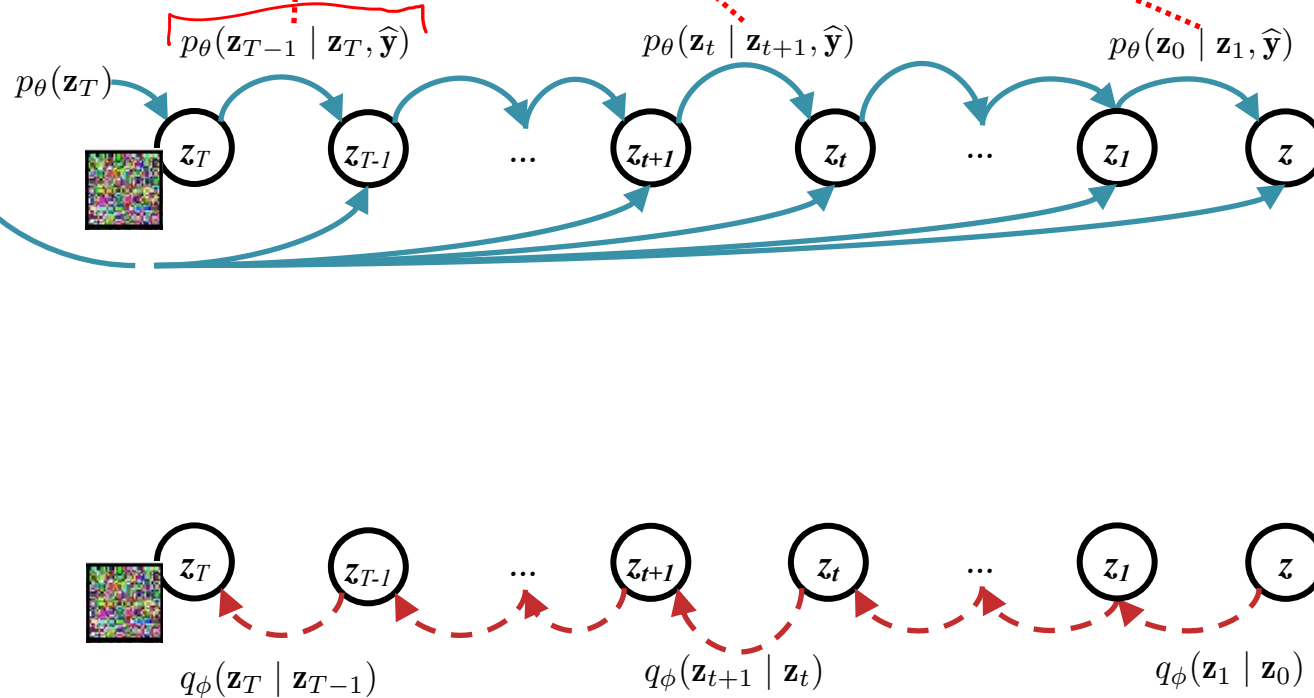
UNet w/cross  
attention



prompt space



latent space



# LDM: with DDPM

## Noise schedule:

We choose  $\alpha_t$  to follow a fixed schedule s.t.  $q_\phi(\mathbf{x}_T) \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ , just like  $p_\theta(\mathbf{x}_T)$ .

Here we let  $\mathbf{z}_0 = \mathbf{z}$ , the output of the encoder from our autoencoder

## Forward Process:

$$q_\phi(\mathbf{z}_{1:T}) = q(\mathbf{z}_0) \prod_{t=1}^T q_\phi(\mathbf{z}_t \mid \mathbf{z}_{t-1})$$

$$\begin{aligned} q(\mathbf{z}_0) &= \text{data distribution} \\ q_\phi(\mathbf{z}_t \mid \mathbf{z}_{t-1}) &\sim \mathcal{N}(\sqrt{\alpha_t} \mathbf{z}_{t-1}, (1 - \alpha_t) \mathbf{I}) \end{aligned}$$

## (Learned) Reverse Process:

$$p_\theta(\mathbf{z}_{1:T}) = p_\theta(\mathbf{z}_T) \prod_{t=1}^T p_\theta(\mathbf{z}_{t-1} \mid \mathbf{z}_t, \tau_\theta(y))$$

$$\begin{aligned} p_\theta(\mathbf{z}_T) &\sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \\ p_\theta(\mathbf{z}_{t-1} \mid \mathbf{z}_t, \tau_\theta(y)) &\sim \mathcal{N}(\mu_\theta(\mathbf{z}_t, t, \tau_\theta(y)), \Sigma_\theta(\mathbf{z}_t, t)) \end{aligned}$$

# LDM: with DDPM

## Noise schedule:

We choose  $\alpha_t$  to follow a fixed schedule s.t.  $q_\phi(\mathbf{x}_T) \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ , just like  $p_\theta(\mathbf{x}_T)$ .

Here we let  $\mathbf{z}_0 = \mathbf{z}$ , the output of the encoder from our autoencoder

## Forward Process:

$$q_\phi(\mathbf{z}_{1:T}) = q(\mathbf{z}_0) \prod_{t=1}^T q_\phi(\mathbf{z}_t \mid \mathbf{z}_{t-1})$$

$$q(\mathbf{z}_0) = \text{data}$$
$$q_\phi(\mathbf{z}_t \mid \mathbf{z}_{t-1}) \sim \mathcal{N}(\sqrt{\alpha_t} \epsilon_t, \Sigma_\phi(\mathbf{z}_{t-1}, t))$$

**Question:** How do we define the mean to condition on the prompt representation?

## (Learned) Reverse Process:

$$p_\theta(\mathbf{z}_{1:T}) = p_\theta(\mathbf{z}_T) \prod_{t=1}^T p_\theta(\mathbf{z}_{t-1} \mid \mathbf{z}_t, \tau_\theta(y))$$

$$p_\theta(\mathbf{z}_T) \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$
$$p_\theta(\mathbf{z}_{t-1} \mid \mathbf{z}_t, \tau_\theta(y)) \sim \mathcal{N}(\mu_\theta(\mathbf{z}_t, t, \tau_\theta(y)), \Sigma_\theta(\mathbf{z}_t, t))$$



# Properties of forward and *exact* reverse process

Recall...

## Property #1:

$$q(\mathbf{x}_t \mid \mathbf{x}_0) \sim \mathcal{N}(\sqrt{\bar{\alpha}_t} \mathbf{x}_0, (1 - \bar{\alpha}_t) \mathbf{I})$$

$$\text{where } \bar{\alpha}_t = \prod_{s=1}^t \alpha_s$$

$\Rightarrow$  we can sample  $\mathbf{x}_t$  from  $\mathbf{x}_0$  at any timestep  $t$  efficiently in closed form

$\Rightarrow \mathbf{x}_t = \sqrt{\bar{\alpha}_t} \mathbf{x}_0 + (1 - \bar{\alpha}_t) \epsilon$  where  $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

**Property #2:** Estimating  $q(\mathbf{x}_{t-1} \mid \mathbf{x}_t)$  is intractable because of its dependence on  $q(\mathbf{x}_0)$ . However, conditioning on  $\mathbf{x}_0$  we can efficiently work with:

$$q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}(\tilde{\mu}_q(\mathbf{x}_t, \mathbf{x}_0), \sigma_t^2 \mathbf{I})$$

$$\text{where } \tilde{\mu}_q(\mathbf{x}_t, \mathbf{x}_0) = \frac{\sqrt{\bar{\alpha}_t}(1 - \alpha_t)}{1 - \bar{\alpha}_t} \mathbf{x}_0 + \frac{\sqrt{\alpha_t}(1 - \bar{\alpha}_t)}{1 - \bar{\alpha}_t} \mathbf{x}_t$$

$$= \alpha_t^{(0)} \mathbf{x}_0 + \alpha_t^{(t)} \mathbf{x}_t$$

$$\sigma_t^2 = \frac{(1 - \bar{\alpha}_{t-1})(1 - \alpha_t)}{1 - \bar{\alpha}_t}$$

**Property #3:** Combining the two previous properties, we can obtain a different parameterization of  $\tilde{\mu}_q$  which has been shown empirically to help in learning  $p_\theta$ .

Rearranging  $\mathbf{x}_t = \sqrt{\bar{\alpha}_t} \mathbf{x}_0 + (1 - \bar{\alpha}_t) \epsilon$  we have that:

$$\mathbf{x}_0 = (\mathbf{x}_t + (1 - \bar{\alpha}_t) \epsilon) / \sqrt{\bar{\alpha}_t}$$

Substituting this definition of  $\mathbf{x}_0$  into property #2's definition of  $\tilde{\mu}_q$  gives:

$$\begin{aligned} \tilde{\mu}_q(\mathbf{x}_t, \mathbf{x}_0) &= \alpha_t^{(0)} \mathbf{x}_0 + \alpha_t^{(t)} \mathbf{x}_t \\ &= \alpha_t^{(0)} ((\mathbf{x}_t + (1 - \bar{\alpha}_t) \epsilon) / \sqrt{\bar{\alpha}_t}) + \alpha_t^{(t)} \mathbf{x}_t \\ &= \frac{1}{\sqrt{\alpha_t}} \left( \mathbf{x}_t - \frac{(1 - \alpha_t)}{\sqrt{1 - \bar{\alpha}_t}} \epsilon \right) \end{aligned}$$

# Parameterizing the *learned* reverse process

Recall...

Recall:  $p_{\theta}(\mathbf{x}_{t-1} \mid \mathbf{x}_t) \sim \mathcal{N}(\mu_{\theta}(\mathbf{x}_t, t), \Sigma_{\theta}(\mathbf{x}_t, t))$

Later we will show that given a training sample  $\mathbf{x}_0$ , we want

$$p_{\theta}(\mathbf{x}_{t-1} \mid \mathbf{x}_t)$$

to be as close as possible to

$$q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0)$$

Intuitively, this makes sense: if the *learned* reverse process is supposed to subtract away the noise, then whenever we're working with a specific  $\mathbf{x}_0$  it should subtract it away exactly as *exact* reverse process would have.

**Idea #1:** Rather than learn  $\Sigma_{\theta}(\mathbf{x}_t, t)$  just use what we know about  $q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0) \sim \mathcal{N}(\tilde{\mu}_q(\mathbf{x}_t, \mathbf{x}_0), \sigma_t^2 \mathbf{I})$ :

$$\Sigma_{\theta}(\mathbf{x}_t, t) = \sigma_t^2 \mathbf{I}$$

**Idea #2:** Choose  $\mu_{\theta}$  based on  $q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0)$ , i.e. we want  $\mu_{\theta}(\mathbf{x}_t, t)$  to be close to  $\tilde{\mu}_q(\mathbf{x}_t, \mathbf{x}_0)$ . Here are three ways we could parameterize this:

**Option C:** Learn a network that approximates the  $\epsilon$  that gave rise to  $\mathbf{x}_t$  from  $\mathbf{x}_0$  in the forward process from  $\mathbf{x}_t$  and  $t$ :

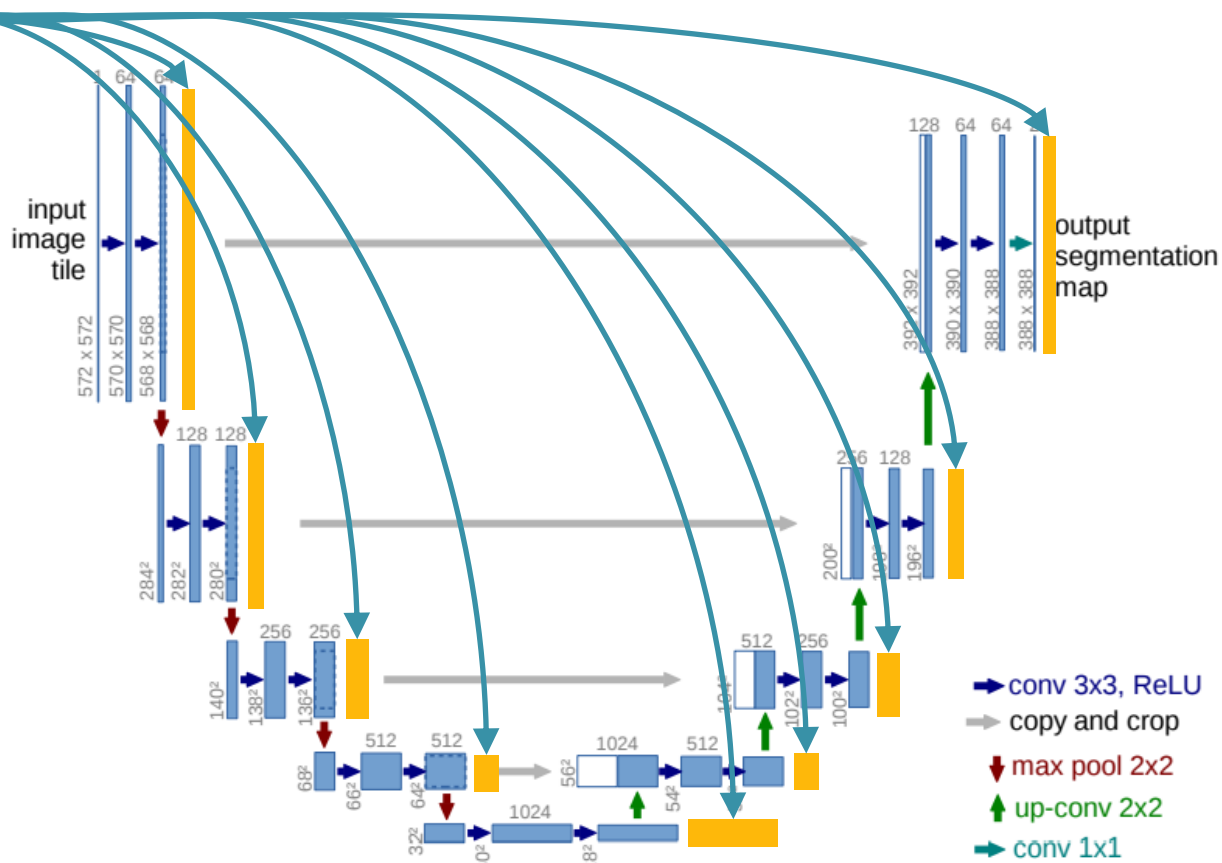
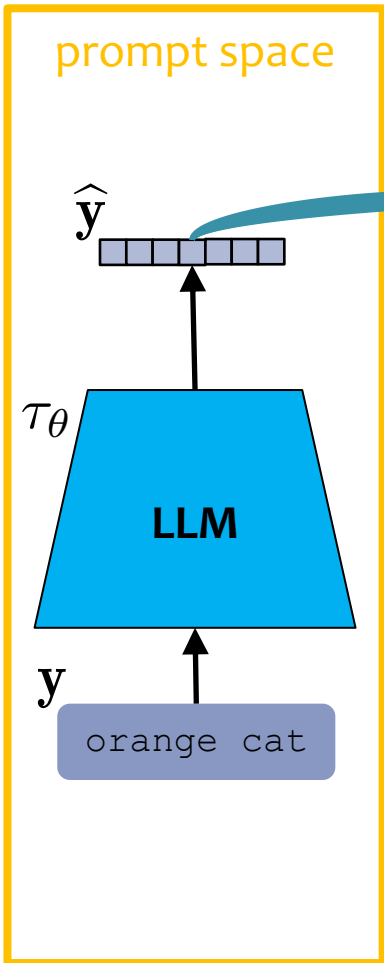
$$\mu_{\theta}(\mathbf{x}_t, t) = \alpha_t^{(0)} \mathbf{x}_{\theta}^{(0)}(\mathbf{x}_t, t) + \alpha_t^{(t)} \mathbf{x}_t$$

where  $\mathbf{x}_{\theta}^{(0)}(\mathbf{x}_t, t) = (\mathbf{x}_t + (1 - \bar{\alpha}_t) \epsilon_{\theta}(\mathbf{x}_t, t)) / \sqrt{\bar{\alpha}_t}$

where  $\epsilon_{\theta}(\mathbf{x}_t, t) = \text{UNet}_{\theta}(\mathbf{x}_t, t)$

# LDM: Noise Model

$$\mu_{\theta}(\mathbf{z}_t, t, \tau_{\theta}(y)) = f(\text{UNet}(\mathbf{z}_t, t, \tau_{\theta}(y)))$$



- The noise model includes **cross attention** (yellow boxes) to the representation of the prompt text
- During training we optimize both the parameters of the UNet noise model and the parameters of the LLM simultaneously

# LDM: Learning the Diffusion Model + LLM

Given a training sample  $\mathbf{z}_0$ , we want

$$p_{\theta}(\mathbf{z}_{t-1} \mid \mathbf{z}_t, \tau_{\theta}(y))$$

to be as close as possible to

$$q(\mathbf{z}_{t-1} \mid \mathbf{z}_t, \mathbf{z}_0)$$

Intuitively, this makes sense: if the *learned* reverse process is supposed to subtract away the noise, then whenever we're working with a specific  $\mathbf{z}_0$  it should subtract it away exactly as *exact* reverse process would have.

Objective Function:

$$L_{LDM} := \mathbb{E}_{\mathcal{E}(x), y, \epsilon \sim \mathcal{N}(0, \mathbf{I}), t} \left[ \|\epsilon - \epsilon_{\theta}(z_t, t, \tau_{\theta}(y))\|_2^2 \right]$$

---

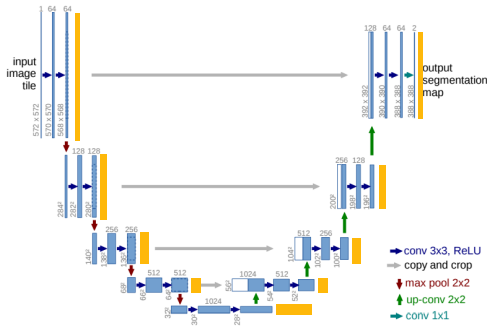
## Algorithm 1 Training

---

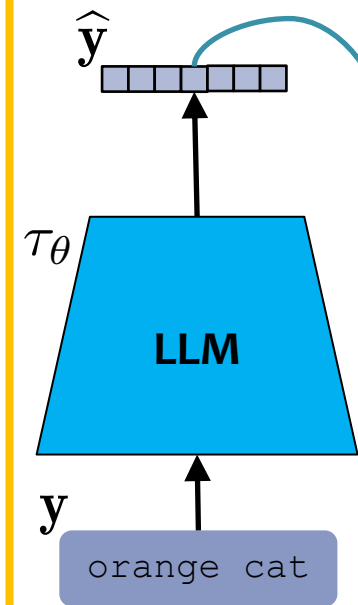
- 1: initialize  $\theta$
  - 2: **for**  $e \in \{1, \dots, E\}$  **do**
  - 3:     **for**  $x_0, y \in \mathcal{D}$  **do**
  - 4:          $t \sim \text{Uniform}(1, \dots, T)$
  - 5:          $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
  - 6:          $\mathbf{x}_t \leftarrow \sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t} \epsilon$
  - 7:          $\ell_t(\theta) \leftarrow \|\epsilon - \epsilon_{\theta}(\mathbf{x}_t, t, \tau_{\theta}(\mathbf{y}))\|^2$
  - 8:          $\theta \leftarrow \theta - \nabla_{\theta} \ell_t(\theta)$
-

# Latent Diffusion Model (LDM)

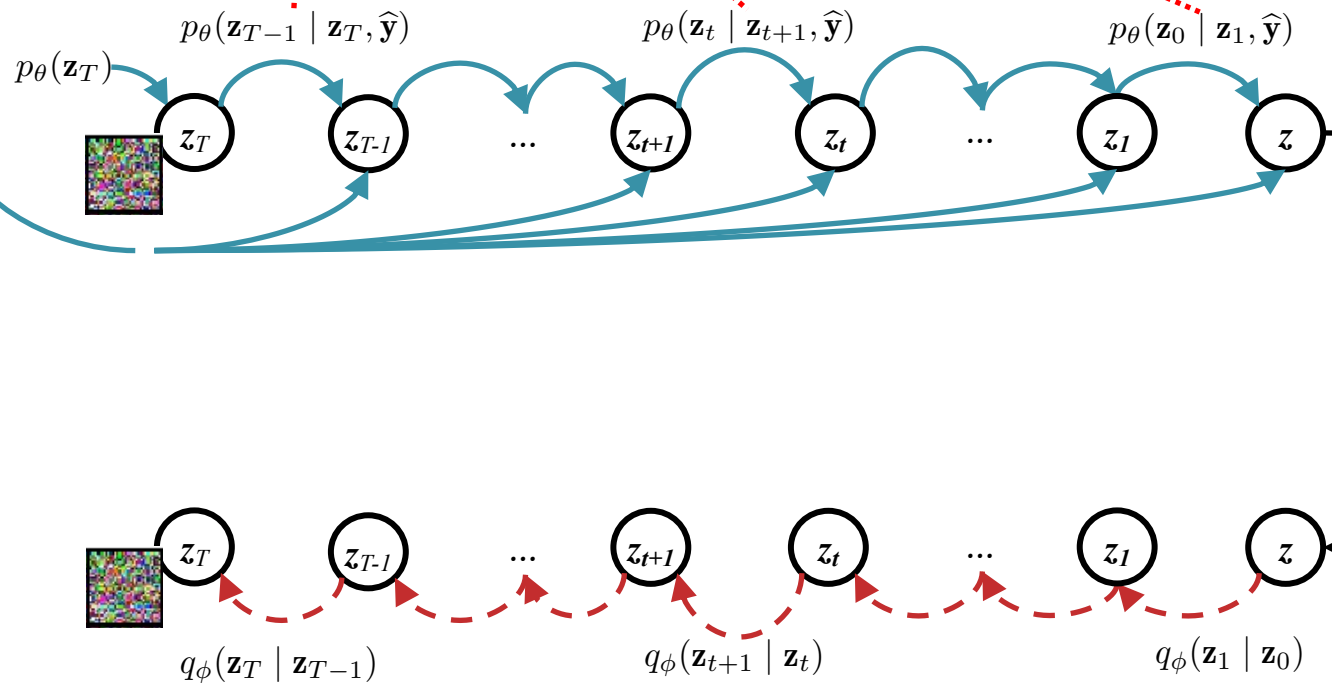
UNet w/cross  
attention



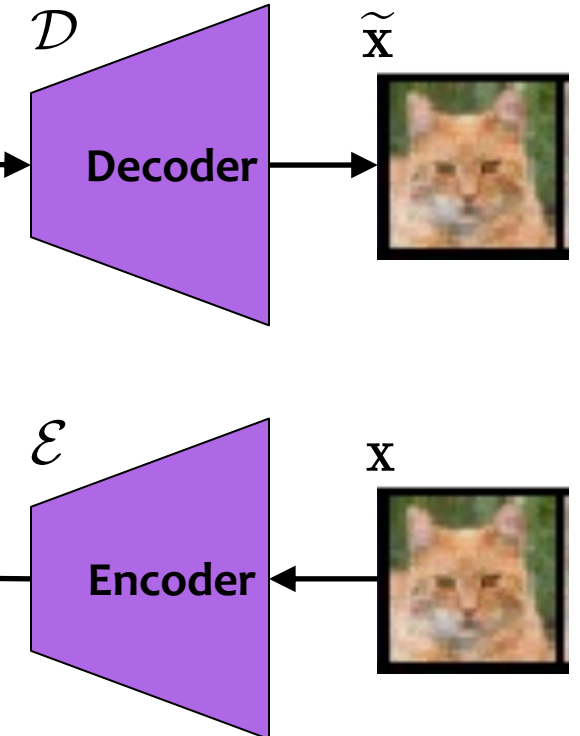
prompt space



latent space

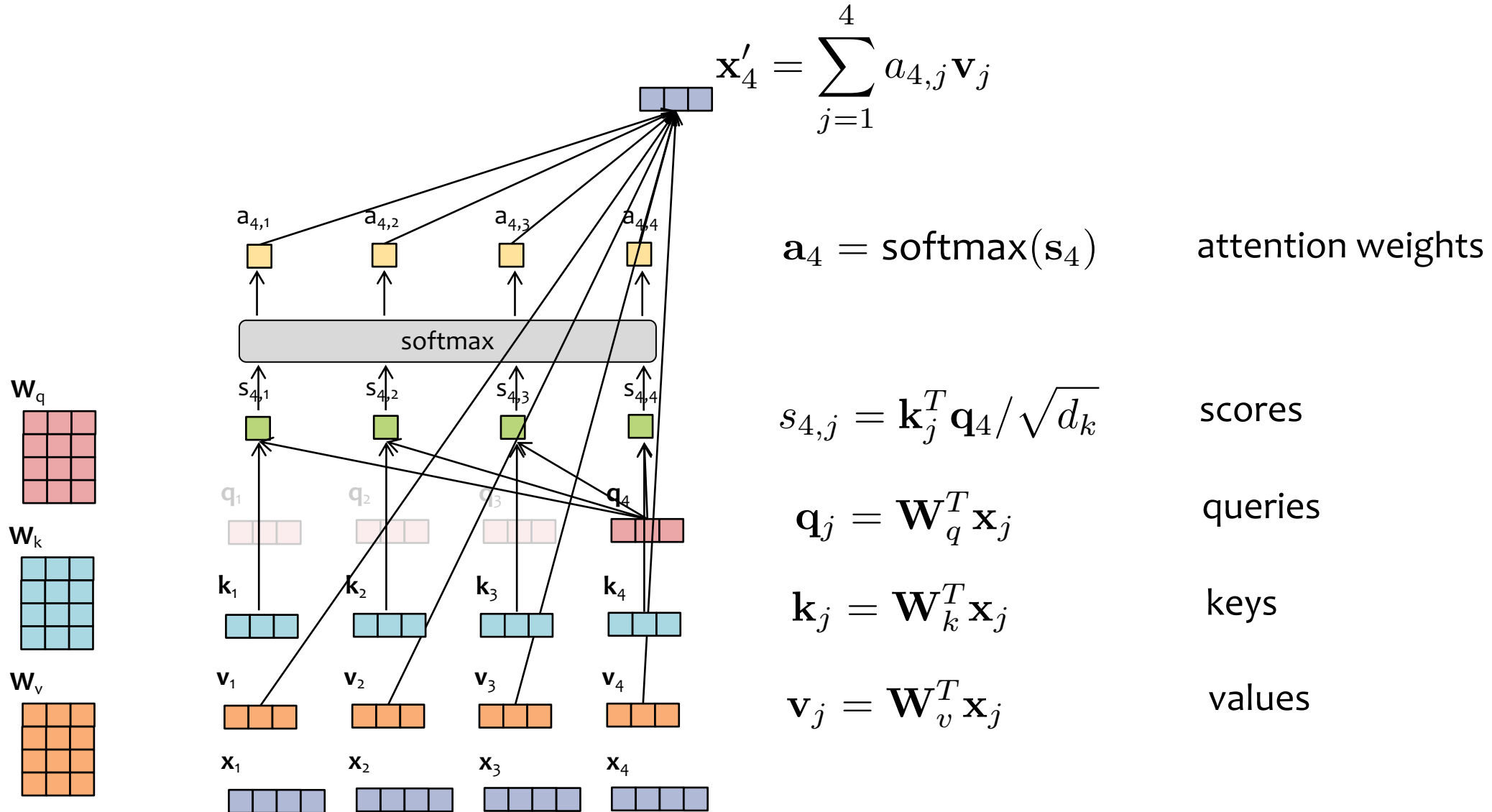


pixel space

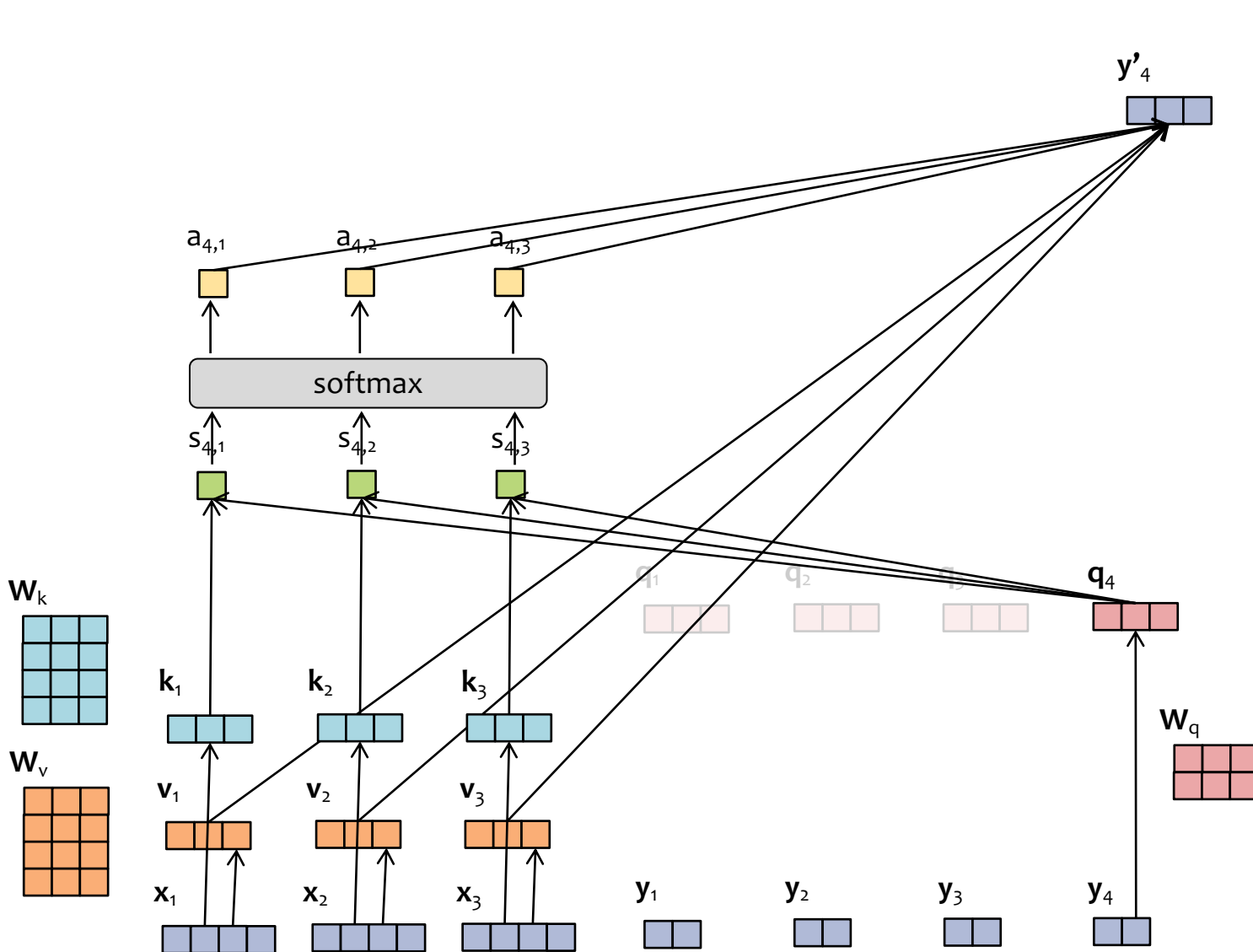


# **CROSS-ATTENTION**

# Scaled Dot-Product Attention



# Cross Attention



$$y'_t = \sum_{j=1}^m a_{t,j} \mathbf{v}_j, \forall t$$

$$\mathbf{a}_t = \text{softmax}(\mathbf{s}_t), \forall t \quad \text{attention weights}$$

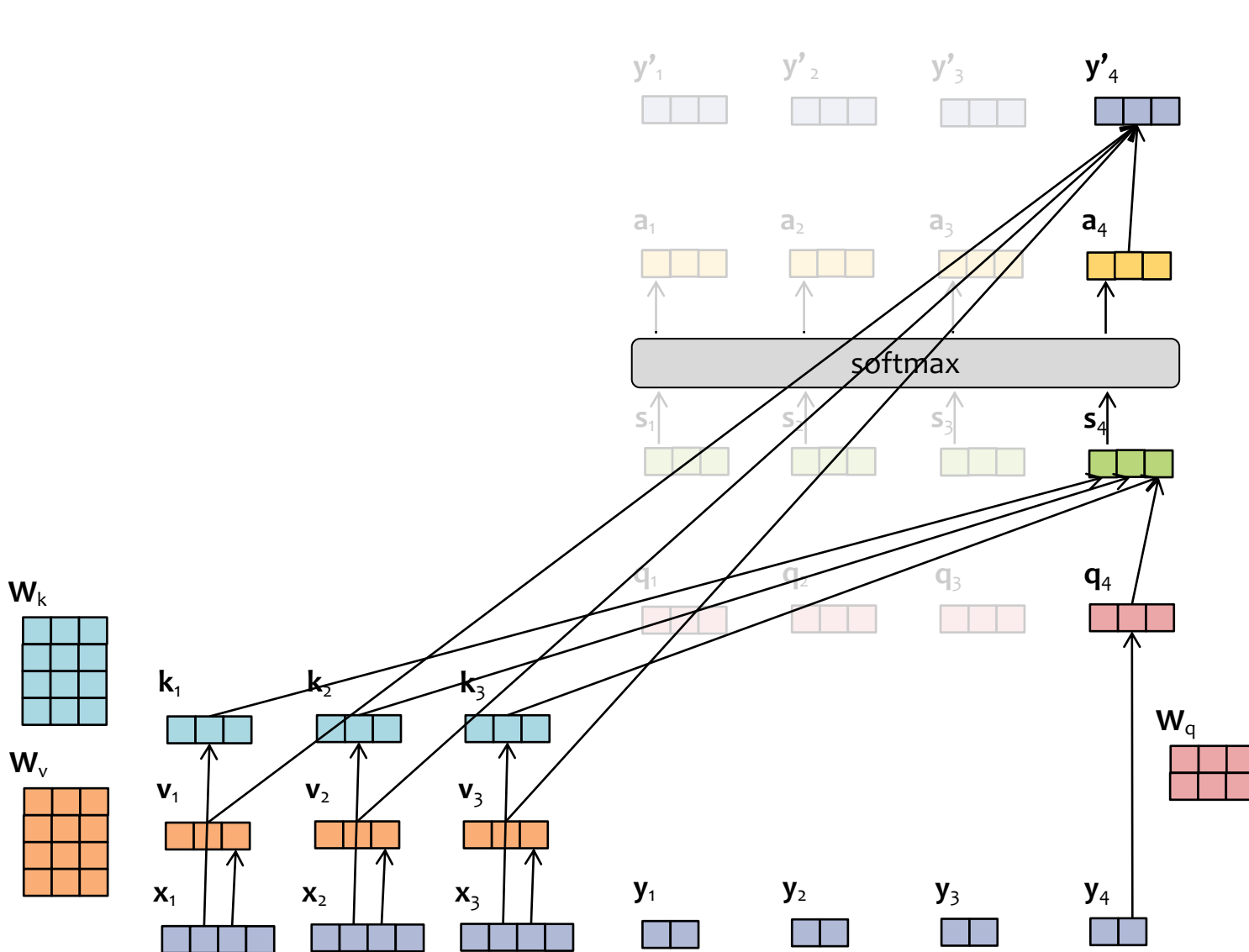
$$s_{t,j} = \mathbf{k}_j^T \mathbf{q}_t / \sqrt{d}, \forall j, t \quad \text{scores}$$

$$\mathbf{q}_t = \mathbf{W}_q^T \mathbf{y}_t, \forall t \in \{1, \dots, n\} \quad \text{queries}$$

$$\mathbf{k}_j = \mathbf{W}_k^T \mathbf{x}_j, \forall j \in \{1, \dots, m\} \quad \text{keys}$$

$$\mathbf{v}_j = \mathbf{W}_v^T \mathbf{x}_j, \forall j \in \{1, \dots, m\} \quad \text{values}$$

# Cross Attention



$$y'_t = \sum_{j=1}^m a_{t,j} \mathbf{v}_j, \forall t$$

$$\mathbf{a}_t = \text{softmax}(\mathbf{s}_t), \forall t \quad \text{attention weights}$$

$$s_{t,j} = \mathbf{k}_j^T \mathbf{q}_t / \sqrt{d}, \forall j, t \quad \text{scores}$$

$$\mathbf{q}_t = \mathbf{W}_q^T \mathbf{y}_t, \forall t \in \{1, \dots, n\} \quad \text{queries}$$

$$\mathbf{k}_j = \mathbf{W}_k^T \mathbf{x}_j, \forall j \in \{1, \dots, m\} \quad \text{keys}$$

$$\mathbf{v}_j = \mathbf{W}_v^T \mathbf{x}_j, \forall j \in \{1, \dots, m\} \quad \text{values}$$

# Cross Attention

$$\mathbf{Y} = \mathbf{A}\mathbf{V} = \text{softmax}(\mathbf{Q}\mathbf{K}^T / \sqrt{d})\mathbf{V}$$

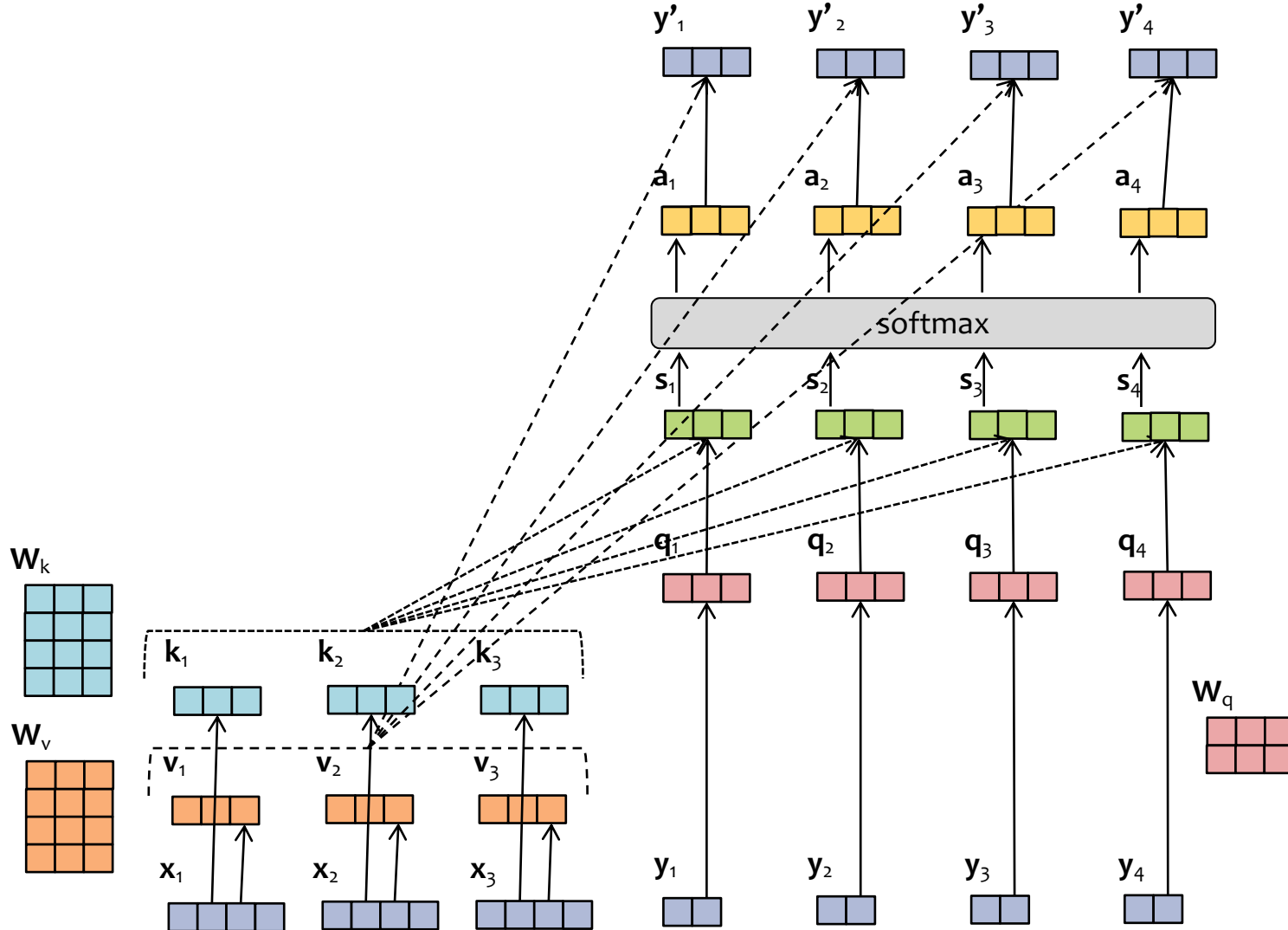
$$\mathbf{A} = \text{softmax}(\mathbf{S}) \quad \text{attention weights}$$

$$\mathbf{S} = \mathbf{Q}\mathbf{K}^T / \sqrt{d} \in \mathbb{R}^{n \times m} \quad \text{scores}$$

$$\mathbf{Q} = \mathbf{Y}\mathbf{W}_q \in \mathbb{R}^{n \times d} \quad \text{queries}$$

$$\mathbf{K} = \mathbf{X}\mathbf{W}_k \in \mathbb{R}^{m \times d} \quad \text{keys}$$

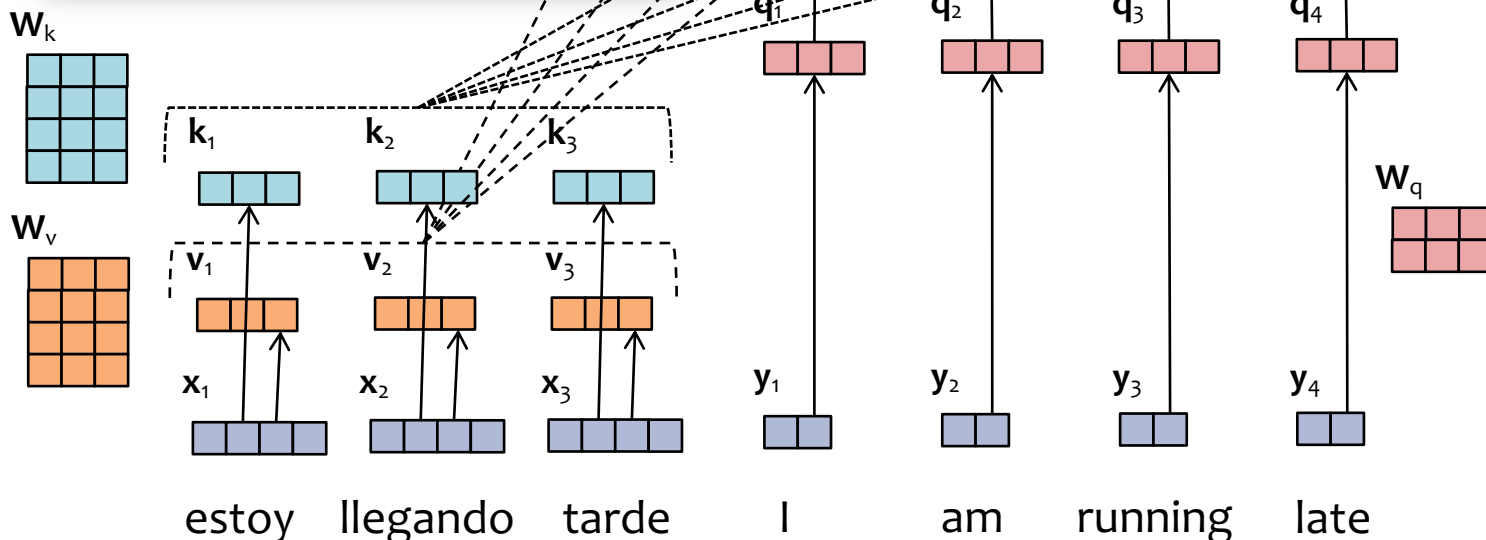
$$\mathbf{V} = \mathbf{X}\mathbf{W}_v \in \mathbb{R}^{m \times d} \quad \text{values}$$



# Example: Cross Attention for Translation

for translation:

- $m$  is the number of tokens in the source language
- $n$  is the number of tokens in the target language
- the attention weights for a target word define a probability distribution over the source words



$$Y = AV = \text{softmax}(QK^T / \sqrt{d})V$$

$$A = \text{softmax}(S) \quad \text{attention weights}$$

$$S = QK^T / \sqrt{d} \in \mathbb{R}^{n \times m} \quad \text{scores}$$

$$Q = YW_q \in \mathbb{R}^{n \times d} \quad \text{queries}$$

$$K = XW_k \in \mathbb{R}^{m \times d} \quad \text{keys}$$

$$V = XW_v \in \mathbb{R}^{m \times d} \quad \text{values}$$

## Cross-Attention in LDM:

- the query matrix is built from a layer of UNet
- the key/value matrices are built from the text-encoder representation of the prompt

# LDM: Cross-Attention

$$\mathbf{Y} = \mathbf{A}\mathbf{V} = \text{softmax}(\mathbf{Q}\mathbf{K}^T / \sqrt{d})\mathbf{V}$$

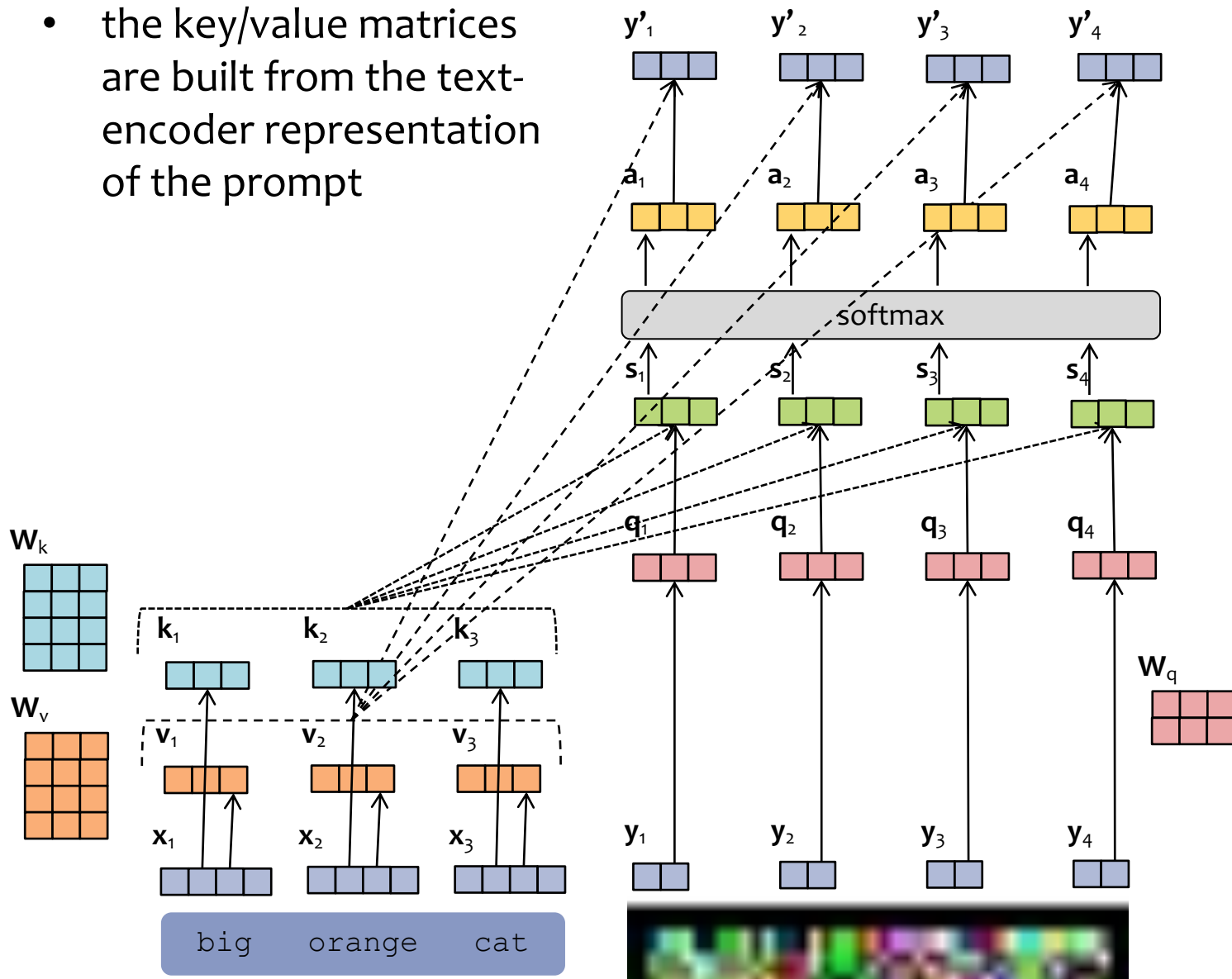
$$\mathbf{A} = \text{softmax}(\mathbf{S})$$

$$\mathbf{S} = \mathbf{Q}\mathbf{K}^T / \sqrt{d} \in \mathbb{R}^{n \times m}$$

$$\mathbf{Q} = \mathbf{Y}\mathbf{W}_q \in \mathbb{R}^{n \times d}$$

$$\mathbf{K} = \mathbf{X}\mathbf{W}_k \in \mathbb{R}^{m \times d}$$

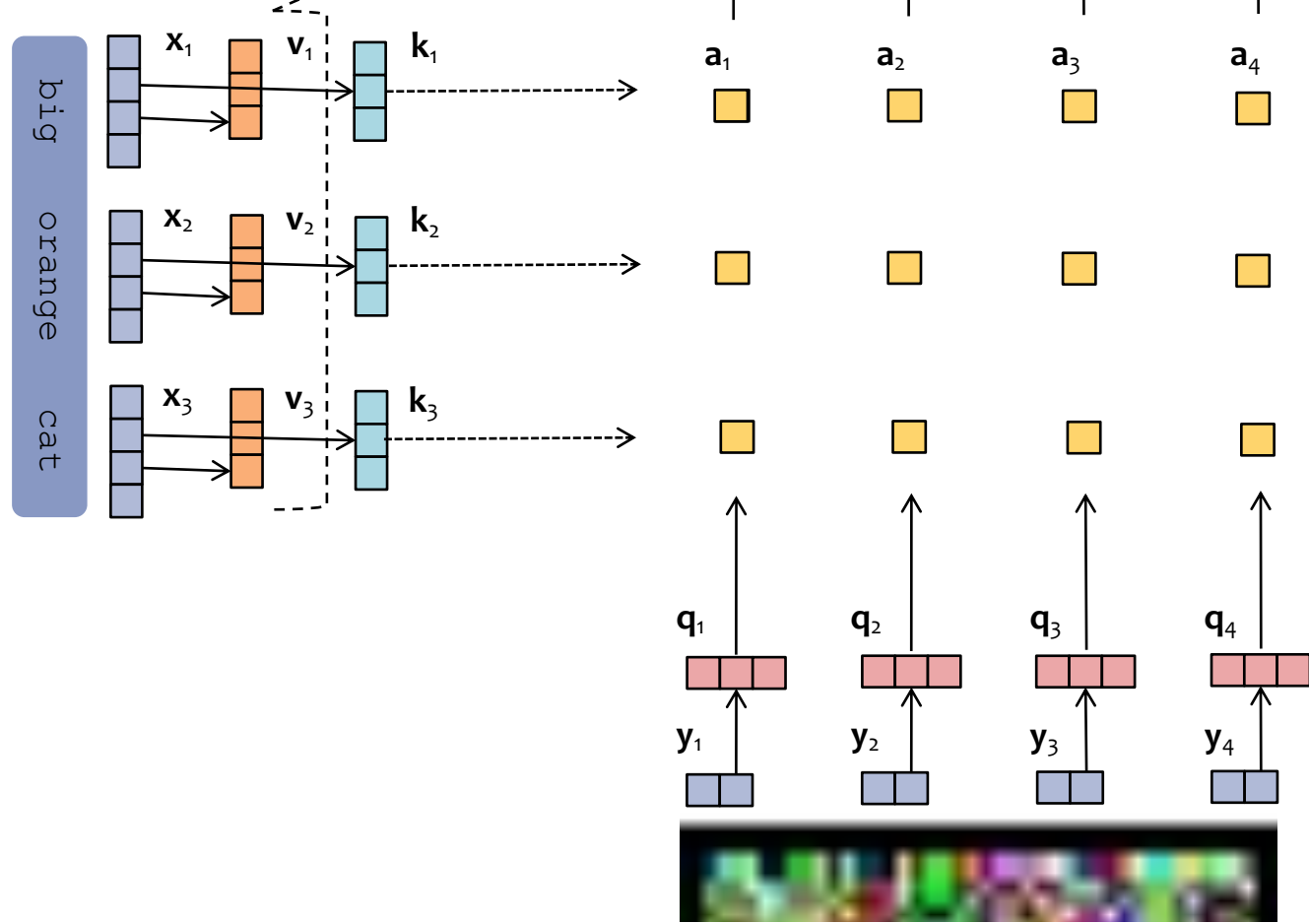
$$\mathbf{V} = \mathbf{X}\mathbf{W}_v \in \mathbb{R}^{m \times d}$$



## Cross-Attention in LDM:

- the query matrix is built from a layer of UNet
- the key/value matrices are built from the text-encoder representation of the prompt

# LDM: Cross-Attention



$$\mathbf{Y} = \mathbf{A}\mathbf{V} = \text{softmax}(\mathbf{Q}\mathbf{K}^T / \sqrt{d})\mathbf{V}$$

$$\mathbf{A} = \text{softmax}(\mathbf{S}) \quad (\text{attention weights})$$

$$\mathbf{S} = \mathbf{Q}\mathbf{K}^T / \sqrt{d} \in \mathbb{R}^{n \times m}$$

$$\mathbf{Q} = \mathbf{Y}\mathbf{W}_q \in \mathbb{R}^{n \times d}$$

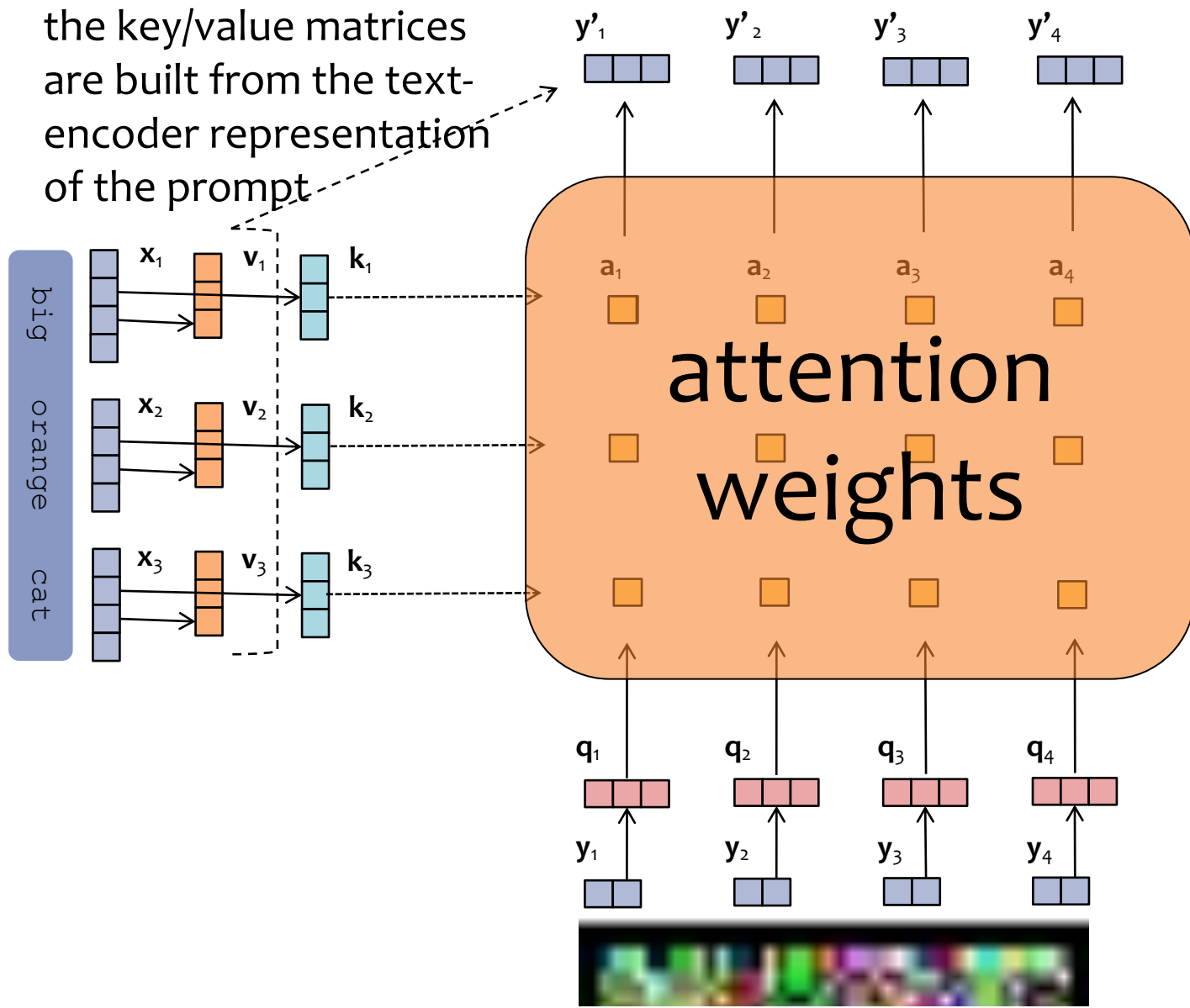
$$\mathbf{K} = \mathbf{X}\mathbf{W}_k \in \mathbb{R}^{m \times d}$$

$$\mathbf{V} = \mathbf{X}\mathbf{W}_v \in \mathbb{R}^{m \times d}$$

## Cross-Attention in LDM:

- the query matrix is built from a layer of UNet
- the key/value matrices are built from the text-encoder representation of the prompt

# LDM: Cross-Attention



$$Y = AV$$

$$A = \text{softmax}$$

$$S = QK^T$$

$$Q = YW$$

$$K = XW$$

$$V = XW$$

for LDM:

- $m$  is the number of tokens in the text prompt
- $n$  is the number of dimensions in the latent space (if we have compression)
- $n$  would be the number of pixels in the image (if we had no compression)
- the attention weights for a (latent) pixel define a probability distribution over the prompt tokens

The actual attention and cross-attention blocks are **multi-head**

# LDM: Cross-Attention in Noise Model

- The cross-attention is placed within a larger Transformer layer

Transformer Layer inside UNet

| input                                                                                                               | $\mathbb{R}^{h \times w \times c}$                                                                                                        |
|---------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------|
| LayerNorm                                                                                                           | $\mathbb{R}^{h \times w \times c}$                                                                                                        |
| Conv1x1                                                                                                             | $\mathbb{R}^{h \times w \times d \cdot n_h}$                                                                                              |
| Reshape                                                                                                             | $\mathbb{R}^{h \cdot w \times d \cdot n_h}$                                                                                               |
| $\times T \left\{ \begin{array}{l} \text{SelfAttention} \\ \text{MLP} \\ \text{CrossAttention} \end{array} \right.$ | $\mathbb{R}^{h \cdot w \times d \cdot n_h}$<br>$\mathbb{R}^{h \cdot w \times d \cdot n_h}$<br>$\mathbb{R}^{h \cdot w \times d \cdot n_h}$ |
| Reshape                                                                                                             | $\mathbb{R}^{h \times w \times d \cdot n_h}$                                                                                              |
| Conv1x1                                                                                                             | $\mathbb{R}^{h \times w \times c}$                                                                                                        |

- The cross-attention modifies the keys and values to be the prompt representation
- The queries are the current layer of UNet

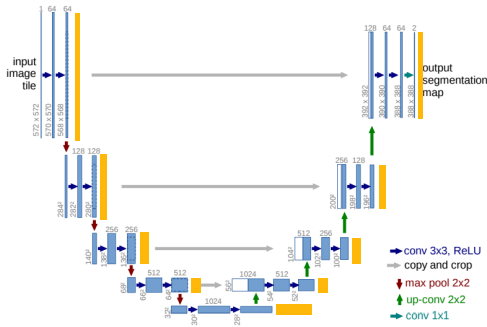
$$\text{Attention}(Q, K, V) = \text{softmax} \left( \frac{QK^T}{\sqrt{d}} \right) \cdot V, \text{ with}$$

$$Q = W_Q^{(i)} \cdot \varphi_i(z_t), K = W_K^{(i)} \cdot \tau_\theta(y), V = W_V^{(i)} \cdot \tau_\theta(y).$$

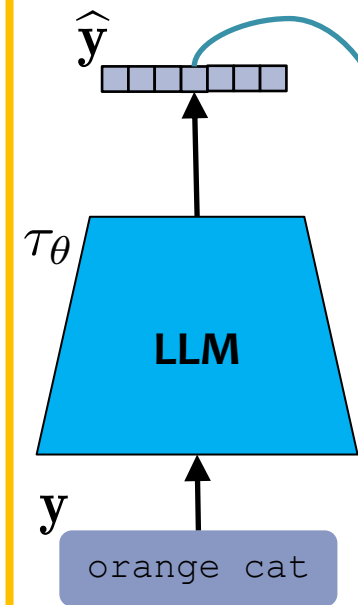
Here,  $\varphi_i(z_t) \in \mathbb{R}^{N \times d_\epsilon^i}$  denotes a (flattened) intermediate representation of the UNet implementing  $\epsilon_\theta$  and  $W_V^{(i)} \in \mathbb{R}^{d \times d_\epsilon^i}$ ,  $W_Q^{(i)} \in \mathbb{R}^{d \times d_\tau}$  &  $W_K^{(i)} \in \mathbb{R}^{d \times d_\tau}$  are learnable projection matrices [36, 97]. See Fig. 3 for a visual depiction.

# Latent Diffusion Model (LDM)

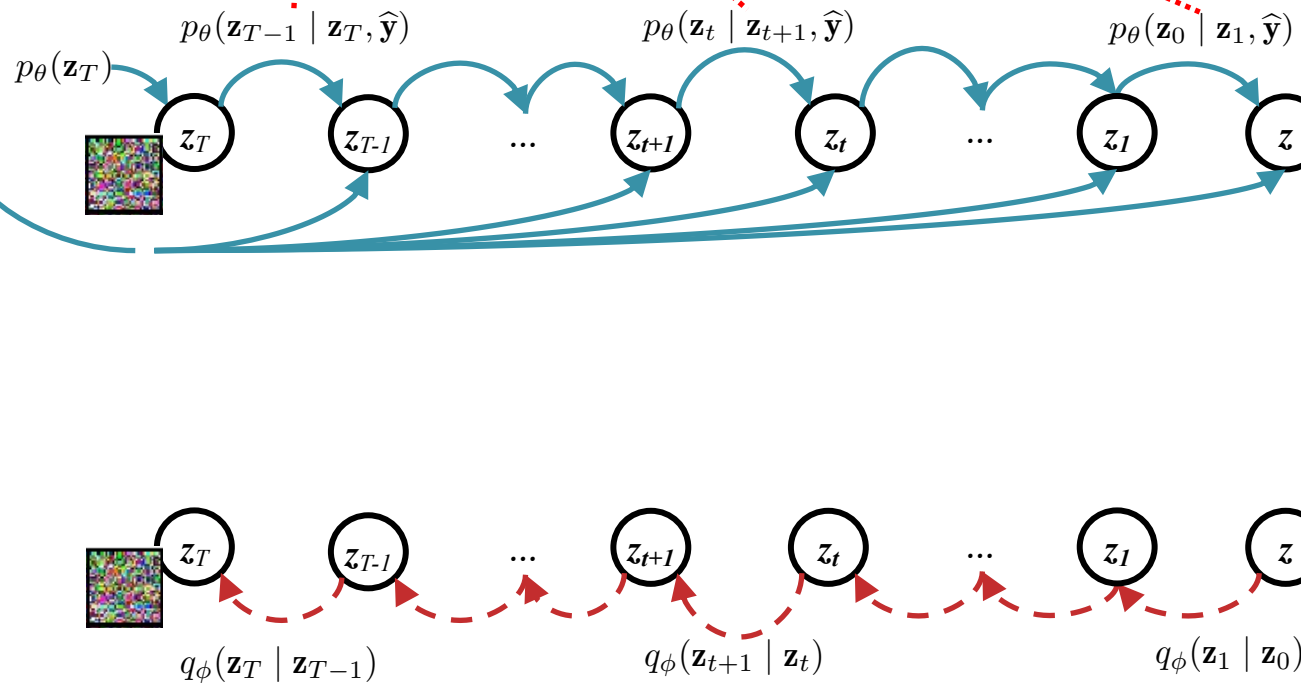
UNet w/cross  
attention



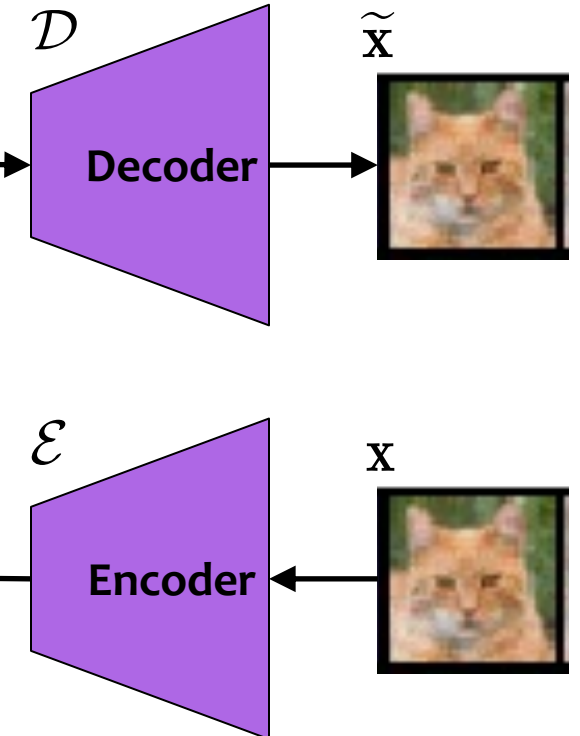
prompt space



latent space



pixel space



# LDM Results

## Text-to-Image Synthesis on LAION. 1.45B Model.

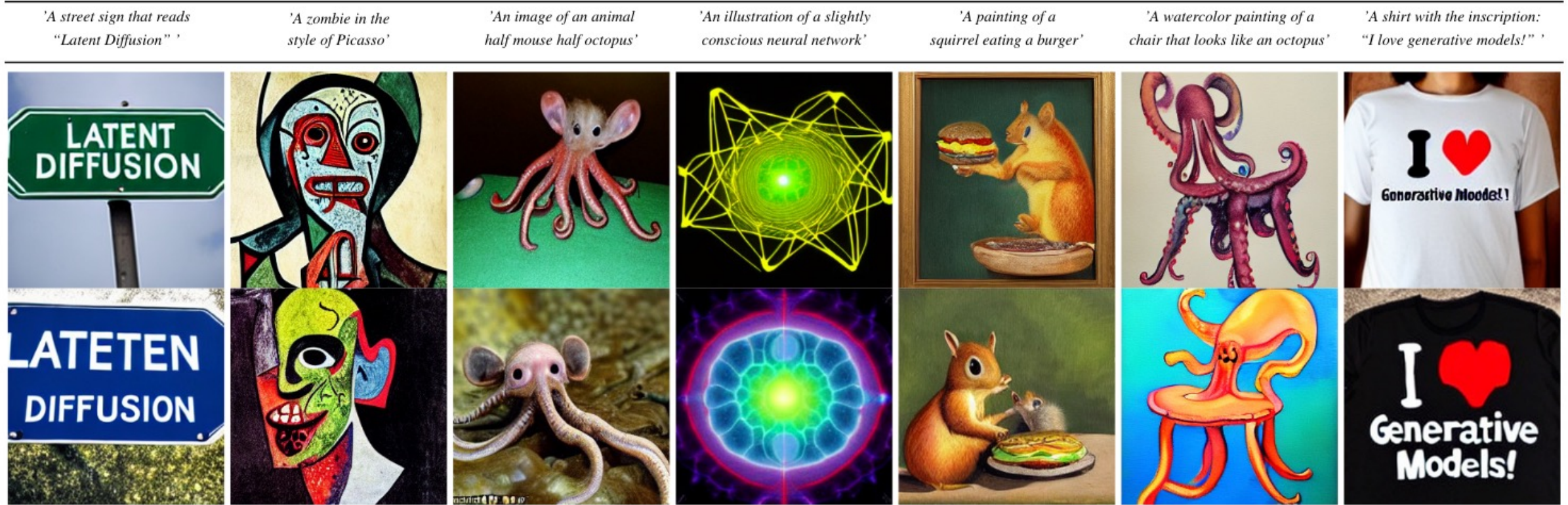


Figure 5. Samples for user-defined text prompts from our model for text-to-image synthesis, *LDM-8 (KL)*, which was trained on the LAION [78] database. Samples generated with 200 DDIM steps and  $\eta = 1.0$ . We use unconditional guidance [32] with  $s = 10.0$ .

# LDM Results

- The result models obtain very high quality FID / IS scores with many fewer parameters than competing models
- The models are much more efficient than vanilla diffusion models because the most computationally intensive step happens in low dimensional latent space, instead of high dimensional pixel space

| Text-Conditional Image Synthesis |              |                                    |                     |                                       |
|----------------------------------|--------------|------------------------------------|---------------------|---------------------------------------|
| Method                           | FID ↓        | IS↑                                | $N_{\text{params}}$ |                                       |
| CogView <sup>†</sup> [17]        | 27.10        | 18.20                              | 4B                  | self-ranking, rejection rate 0.017    |
| LAFITE <sup>†</sup> [109]        | 26.94        | <u>26.02</u>                       | 75M                 |                                       |
| GLIDE* [59]                      | <u>12.24</u> | -                                  | 6B                  | 277 DDIM steps, c.f.g. [32] $s = 3$   |
| Make-A-Scene* [26]               | <b>11.84</b> | -                                  | 4B                  | c.f.g for AR models [98] $s = 5$      |
| <i>LDM-KL-8</i>                  | 23.31        | $20.03 \pm 0.33$                   | 1.45B               | 250 DDIM steps                        |
| <i>LDM-KL-8-G*</i>               | 12.63        | <b><math>30.29 \pm 0.42</math></b> | 1.45B               | 250 DDIM steps, c.f.g. [32] $s = 1.5$ |

Table 2. Evaluation of text-conditional image synthesis on the  $256 \times 256$ -sized MS-COCO [51] dataset: with 250 DDIM [84] steps our model is on par with the most recent diffusion [59] and autoregressive [26] methods despite using significantly less parameters. <sup>†</sup>/\*:Numbers from [109]/[26]

# **CLASSIFIER-FREE GUIDANCE**

# Classifier-free Guidance

$$p(\mathbf{c} | \mathbf{z}_0) = \frac{p(\mathbf{z}_0 | \mathbf{c}) p(\mathbf{c})}{p(\mathbf{z}_0)}$$

- Diffusion models (unlike GANs) are great at generating diverse samples
- But when diffusion is conditioned on some input (text, label, etc.) that diversity may cause it to stray away from the prompt
- **Classifier-free guidance** helps diffusion to adhere to the prompt, yielding higher quality images

## Motivation:

- CFG steers generation to increase:

$$+ \log p(\mathbf{c})$$

$$\log p(\mathbf{c} | \mathbf{z}_0) \propto \log p(\mathbf{z}_0 | \mathbf{c}) - \log p(\mathbf{z}_0)$$

$$\text{where } \nabla_{\mathbf{z}_0} \log p(\mathbf{c} | \mathbf{z}_0) \propto \nabla_{\mathbf{z}_0} \log p(\mathbf{z}_0 | \mathbf{c}) - \nabla_{\mathbf{z}_0} \log p(\mathbf{z}_0)$$

- Define  $\log p(\mathbf{z}_0)$  through  $\epsilon_\theta(\mathbf{z}_t, t) = \epsilon_\theta(\mathbf{z}_t, t, \emptyset)$  where  $\emptyset$  is given a null embedding and otherwise shares parameters with  $\epsilon_\theta(\mathbf{z}_t, t, \mathbf{c})$
- So we guide the DDPM sampling to high  $p(\mathbf{c} | \mathbf{z}_0)$  by:

$$\epsilon_\theta \leftarrow \epsilon_\theta(\mathbf{z}_t, t, \emptyset) + w \cdot \nabla_{\mathbf{z}_0} \log p(\mathbf{z}_0 | \mathbf{c})$$

$$\propto \epsilon_\theta(\mathbf{z}_t, t, \emptyset) + w (\epsilon_\theta(\mathbf{z}_t, t, \mathbf{c}) - \epsilon_\theta(\mathbf{z}_t, t, \emptyset))$$

where  $w > 1$  is the scale of the guidance and  $w = 1$  gives standard sampling

- Larger  $w \Rightarrow$  higher  $p(\mathbf{c} | \mathbf{z}_0)$ , lower diversity.

# Classifier-free Guidance

- Diffusion models (unlike GANs) are great at generating diverse samples
- But when diffusion is conditioned on some input (text, label, etc.) that diversity may cause it to stray away from the prompt
- **Classifier-free guidance** helps diffusion to adhere to the prompt, yielding higher quality images

$$w' = w - 1$$

---

## Algorithm 1 Sampling from DDPM with Classifier-free Guidance

---

```

1:  $w = 7.5$ 
2:  $\mathbf{c} = \text{tokenize}(\text{"a cat with green eyes"})$ 
3:  $\mathbf{c}' = \text{tokenize}(\text{""})$ 
4:  $\mathbf{z}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ 
5: for  $t \in \{T, \dots, 1\}$  do
6:    $\epsilon_\theta \leftarrow (1 + w')\epsilon_\theta(\mathbf{z}_t, t, \mathbf{c}) - w'\epsilon_\theta(\mathbf{z}_t, t, \mathbf{c}')$ 
7:    $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ 
8:    $\hat{\mathbf{z}}_0 \leftarrow (\mathbf{z}_t - \sqrt{1 - \bar{\alpha}_t}\epsilon_\theta) / \sqrt{\bar{\alpha}_t}$ 
9:    $\hat{\mu}_t \leftarrow \alpha_t^{(0)}\hat{\mathbf{z}}_0 + \alpha_t^{(t)}\mathbf{z}_t$ 
10:   $\mathbf{z}_{t-1} \leftarrow \hat{\mu}_t + \sigma_t^2\epsilon$ 
11: return  $\mathbf{x}_0$ 

```

---

$$\epsilon_\theta \leftarrow \frac{w}{w} \epsilon_\theta(\mathbf{z}_t, t, \mathbf{c}) - \frac{(w-1)}{w} \epsilon_\theta(\mathbf{z}_t, t, \mathbf{c}') = \epsilon_\theta(\mathbf{z}_t, t, \mathbf{c}) + (1 - \frac{1}{w}) \epsilon_\theta(\mathbf{z}_t, t, \mathbf{c}')$$

# Classifier-free Guidance

- Increasing guidance scale yields samples that more closely adhere to the class label
- Guidance scale  $w$  increases from the left block of samples to the right block of samples

$w=0.0$



$w=3.0$



# Classifier-free Guidance

- Increasing guidance scale yields samples that more closely adhere to the class label
- Guidance scale  $w$  increases from the left block of samples to the right block of samples

$w=1.0$

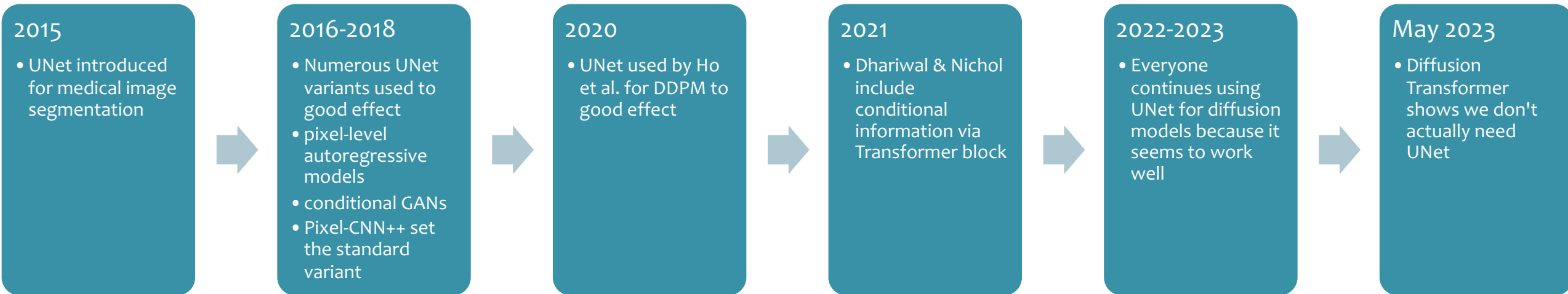


$w=3.0$

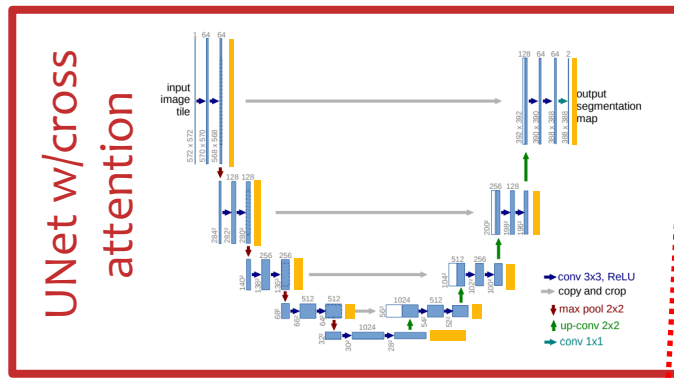


# **DIFFUSION WITH TRANSFORMERS**

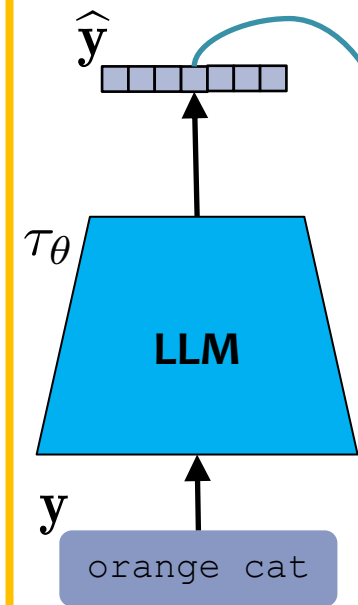
# Diffusion Transformer (DiT)



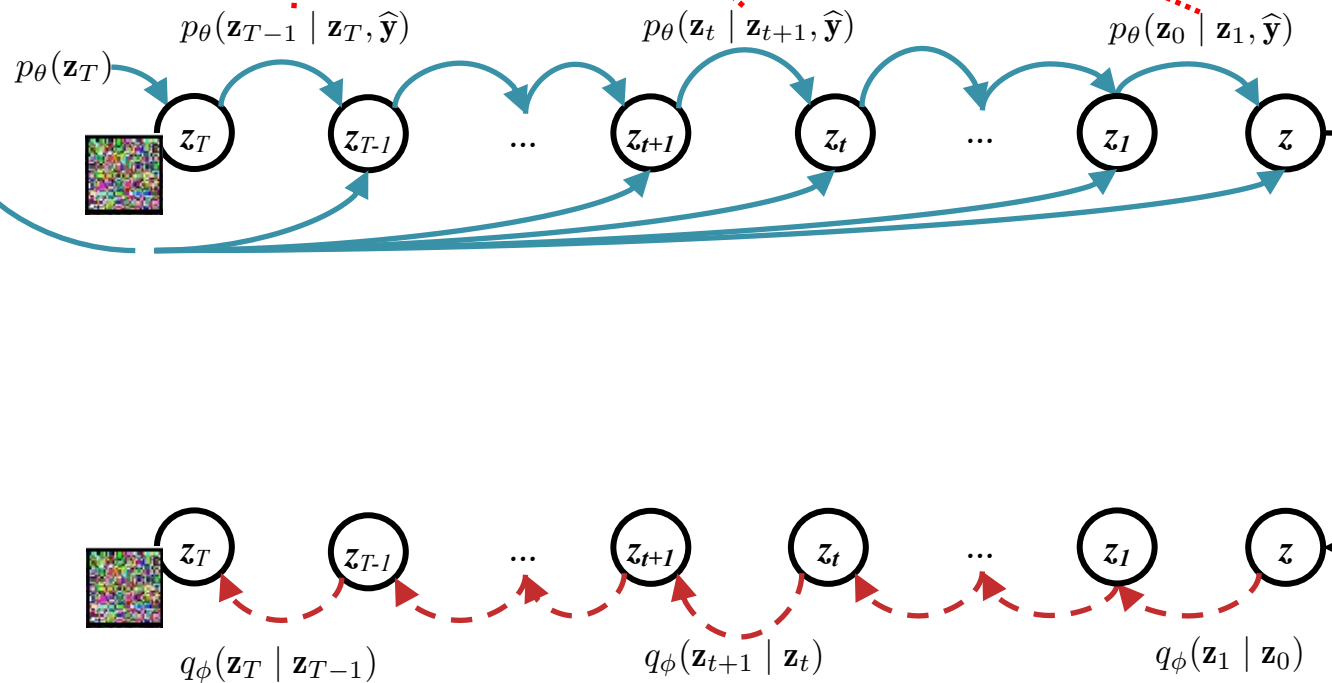
# Latent Diffusion Model (LDM) with UNet backbone



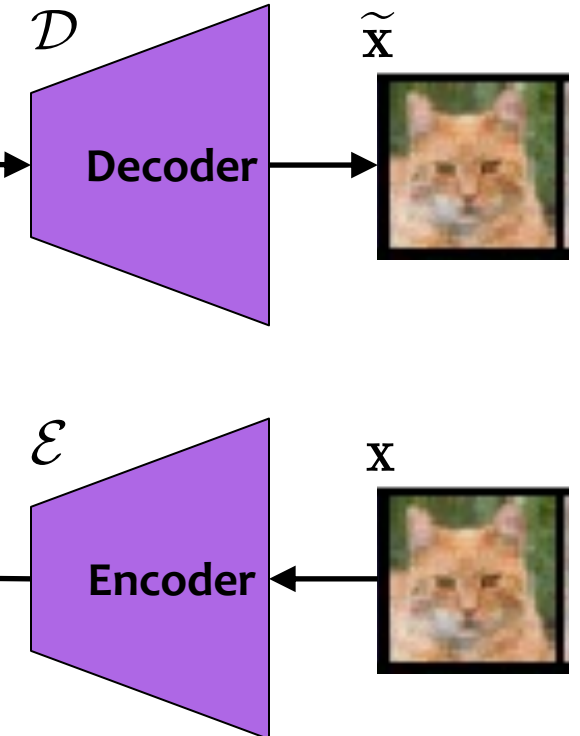
prompt space



latent space

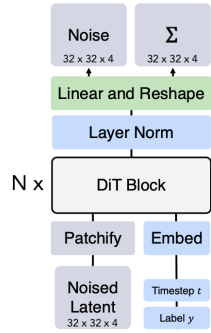


pixel space

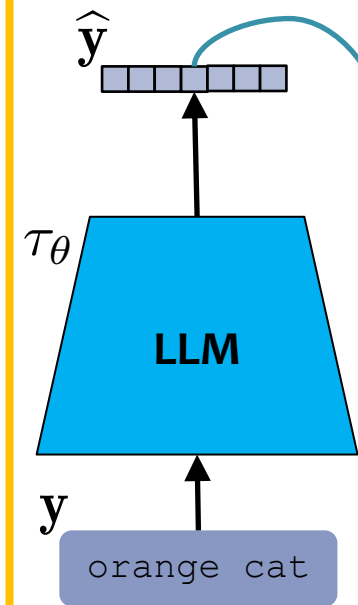


# Latent Diffusion Model (LDM) with Diffusion Transformer backbone

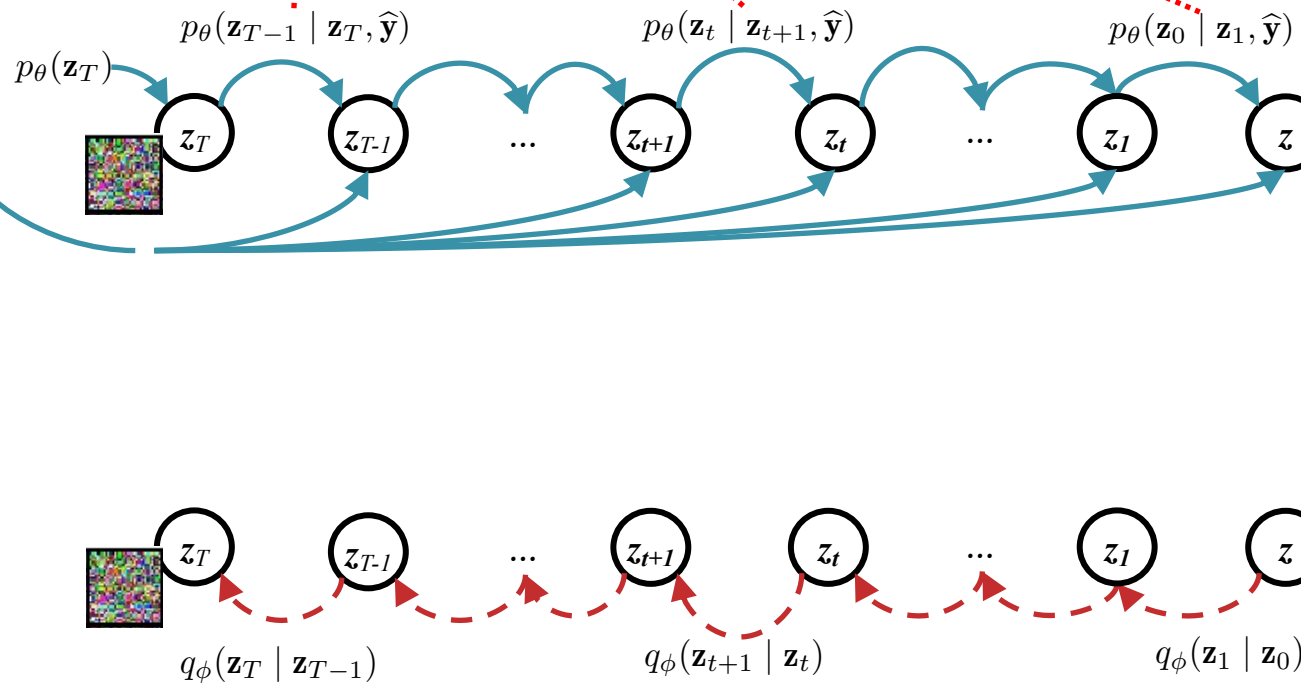
Diffusion  
Transformer



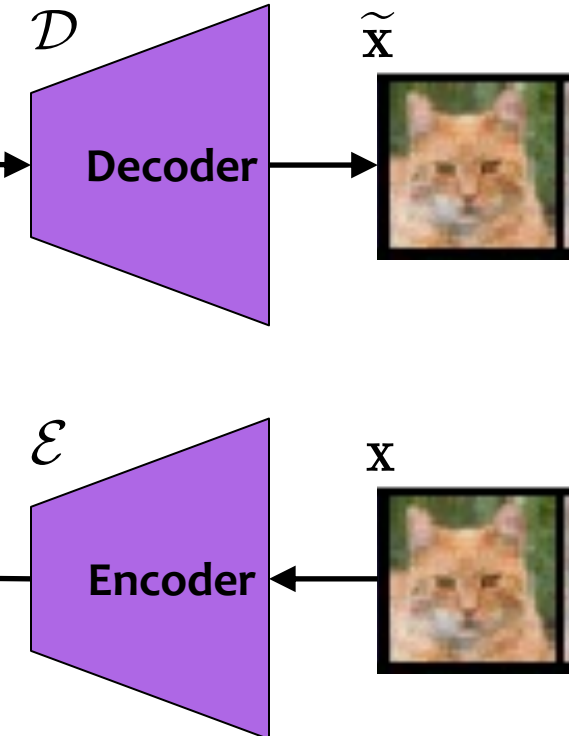
prompt space



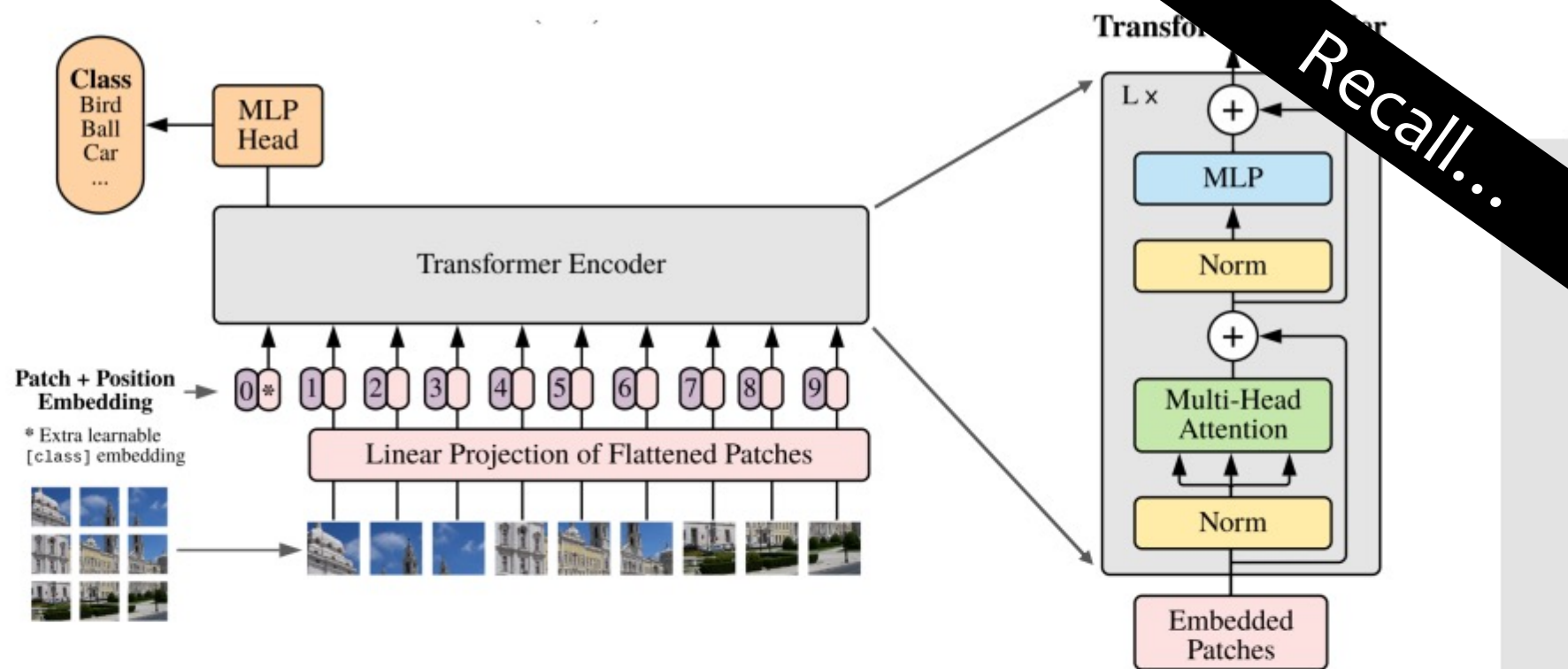
latent space



pixel space



# Vision Transformer (ViT)

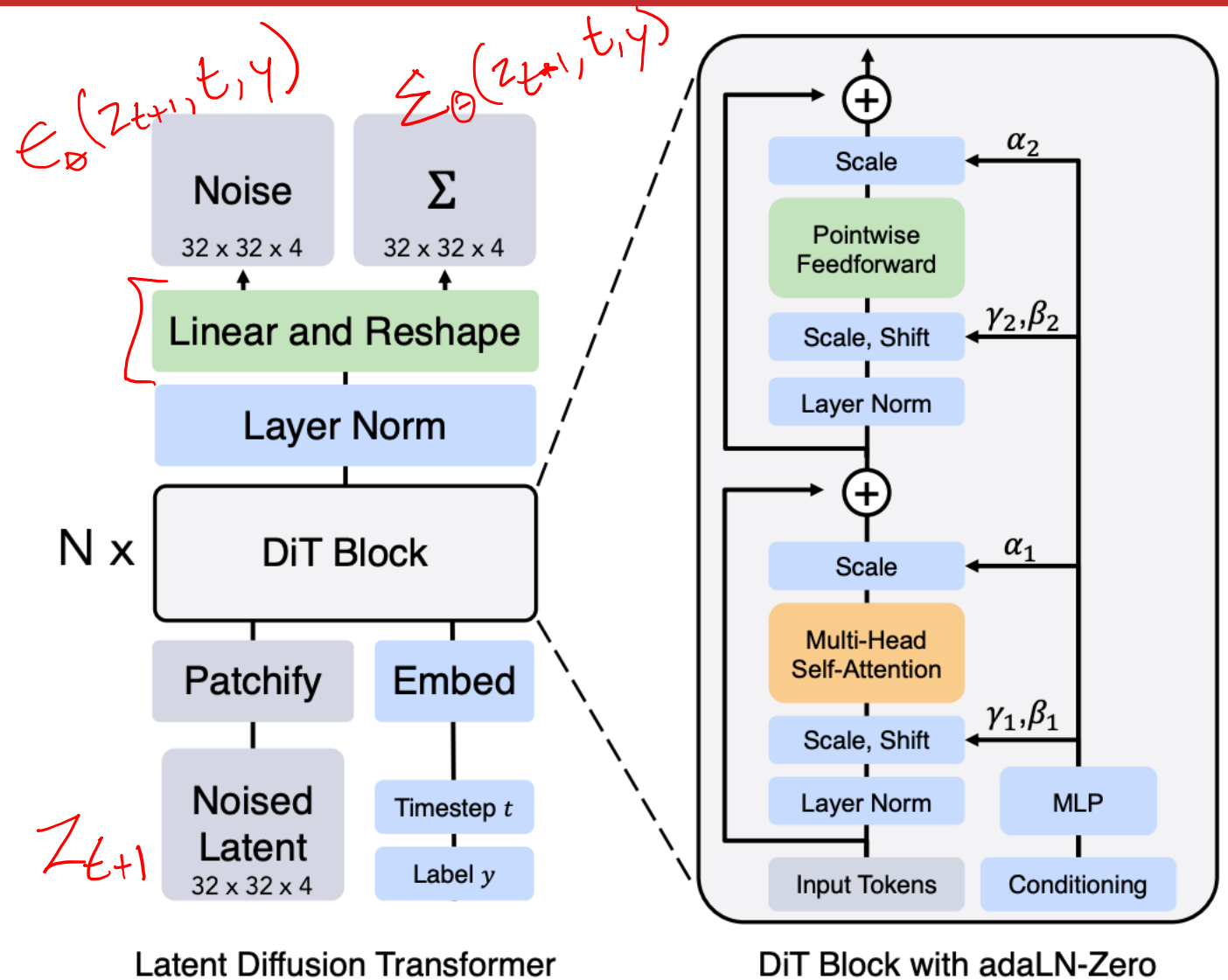


- Instead of words as input, the inputs are  $P \times P$  pixel *patches*
- Each patch is embedded linearly into a vector of size 1024
- Uses 1D positional embeddings
- Pre-trained on a large, supervised dataset (e.g., ImageNet 21K, JFT-300M)

# Diffusion Transformer (DiT)

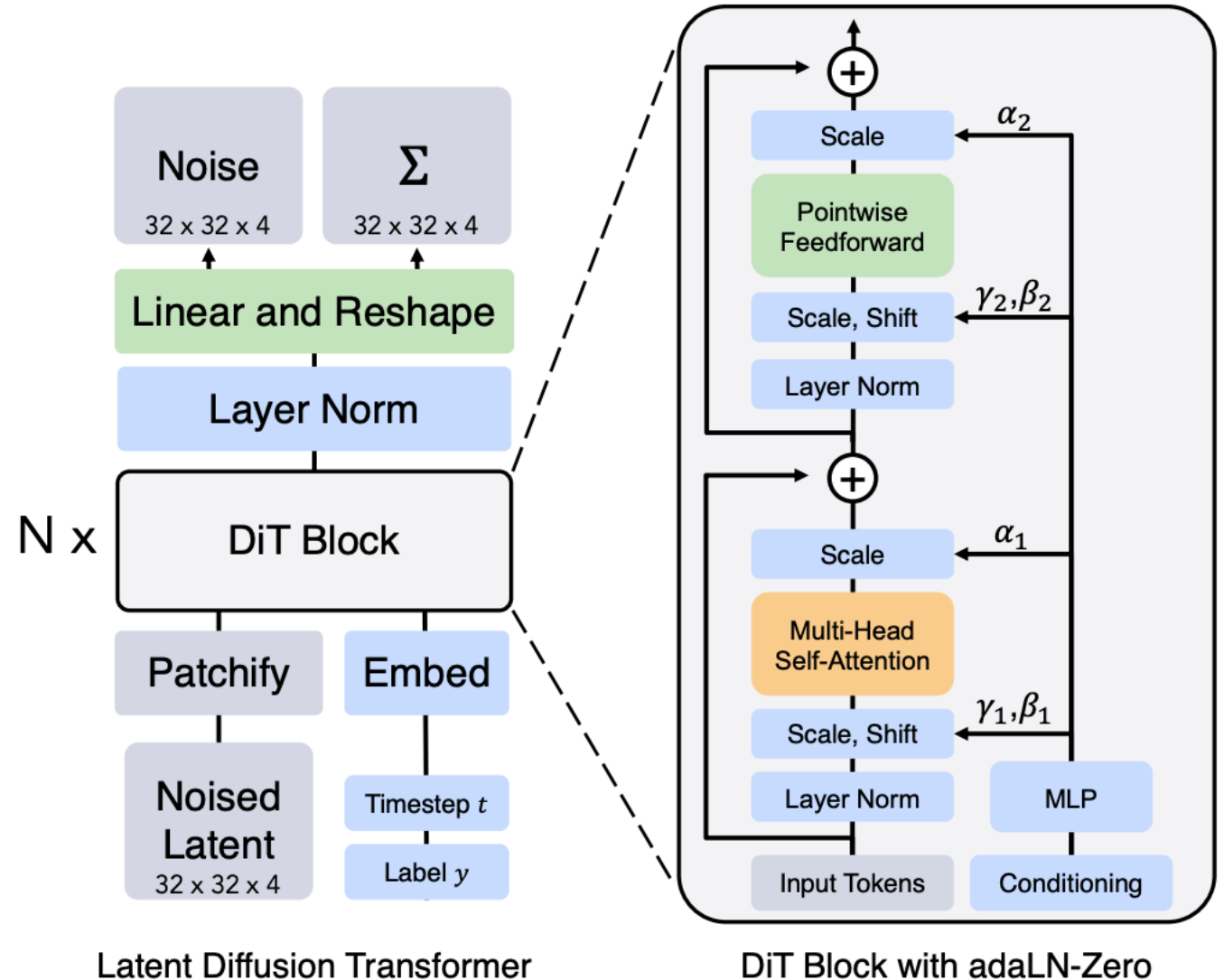
Diffusion Transformer

- DiT backbone is essentially a Vision Transformer (ViT) with some tweaks
- Input is a noisy latent, a timestep, and a label (or other conditional information)
- Output is a mean and covariance output, each of fixed size
- After a final layer norm, a linear layer is used to convert from a sequence of T token embeddings to fixed size output



# Adaptive LayerNorm

- Within the **DiT Block**, the interesting part is how we condition on the label
- Original DiT paper tries out various approaches:
  - In-context conditioning
  - Cross-attention block
  - Adaptive layer norm (adaLN) block
  - Adaptive layer norm with zero initialization strategy (adaLN-Zero)
- adaLN-Zero is the best approach empirically
  - key insight: learn an MLP that outputs the scale and shift parameters for LayerNorm and residual connections



# Adaptive LayerNorm

Diffusion Transformer

- Within the **DiT Block**, the interesting part is how we condition on the label
- Original DiT paper tries out various approaches:
  - In-context conditioning 1
  - Cross-attention block 2
  - Adaptive layer norm (adaLN) block 3
  - Adaptive layer norm with zero initialization strategy (adaLN-Zero) 4
- adaLN-Zero is the best approach empirically
  - key insight: learn an MLP that outputs the scale and shift parameters for LayerNorm and residual connections

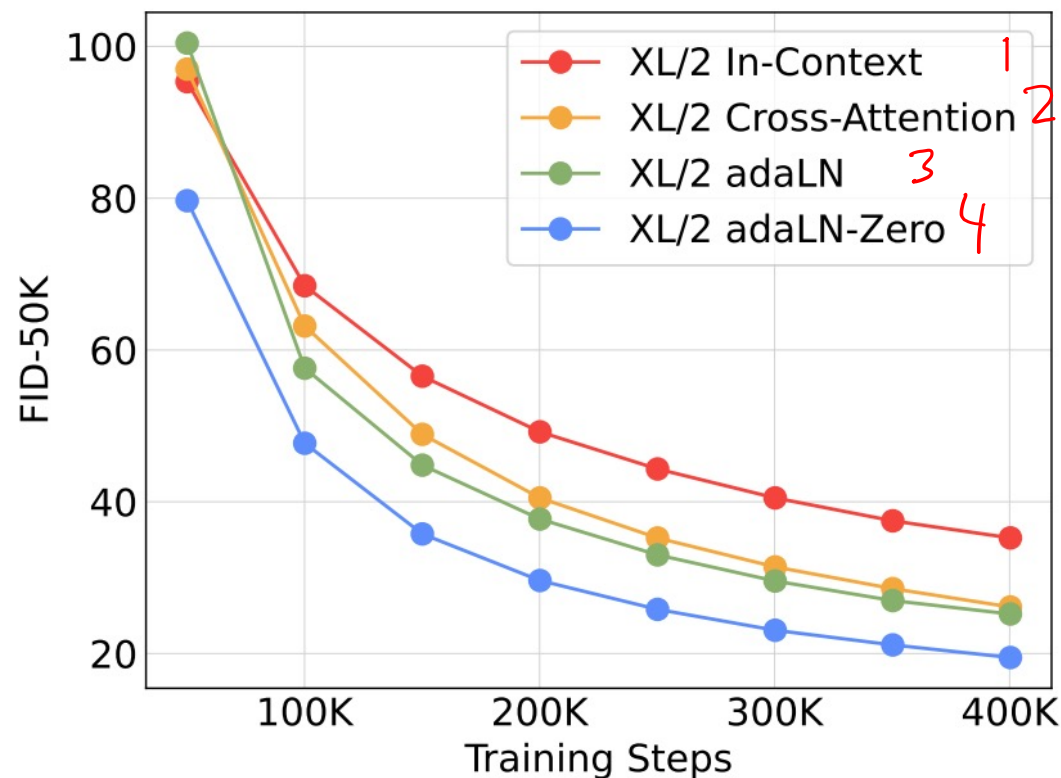


Figure 5. **Comparing different conditioning strategies.** adaLN-Zero outperforms cross-attention and in-context conditioning at all stages of training.

# GFLOPS and FID

## GFLOPS

*Definition:* GFLOPS are a hardware-independent measure of the computation cost of a model

- A FLOP is one floating-point operation (like a single addition, multiplication, etc. on real numbers).
- 1 GFLOP =  $10^9$  FLOPs.
- GFLOPs measure how much computation a model requires for one forward pass (or sometimes per image/sample)

## FID

*Definition: Fréchet Inception Distance:* a standard metric used to evaluate the **quality of generated images**

- **To compute:**

1. Pass both real and generated images through a pretrained Inception-v3 network.
2. Collect the activations (features) from a mid-level layer (often the pool3 layer)
3. Model these feature sets as multivariate Gaussians with means  $\mu_r, \mu_g$  and covariances  $\Sigma_r, \Sigma_g$
4. Compute the Fréchet distance between the two Gaussians:

$$\text{FID} = \|\mu_r - \mu_g\|_2^2 + \text{Tr}(\Sigma_r + \Sigma_g - 2(\Sigma_r \Sigma_g)^{1/2})$$

- **Interpretation:**

- Lower FID → generated images are more similar to real images (better quality and diversity).
- Higher FID → larger gap between generated and real distributions (worse quality).

# Diffusion Transformer

- Decreasing the patch size of a DiT increases the GFLOPs even though it does not increase the number of parameters
- So DiT studies the relationship between FID performance and GFLOPs (rather than FID and # of param.s)
- Question:** How much does the total computation increase if we drop the patch size from  $p=4$  to  $p=2$ ?

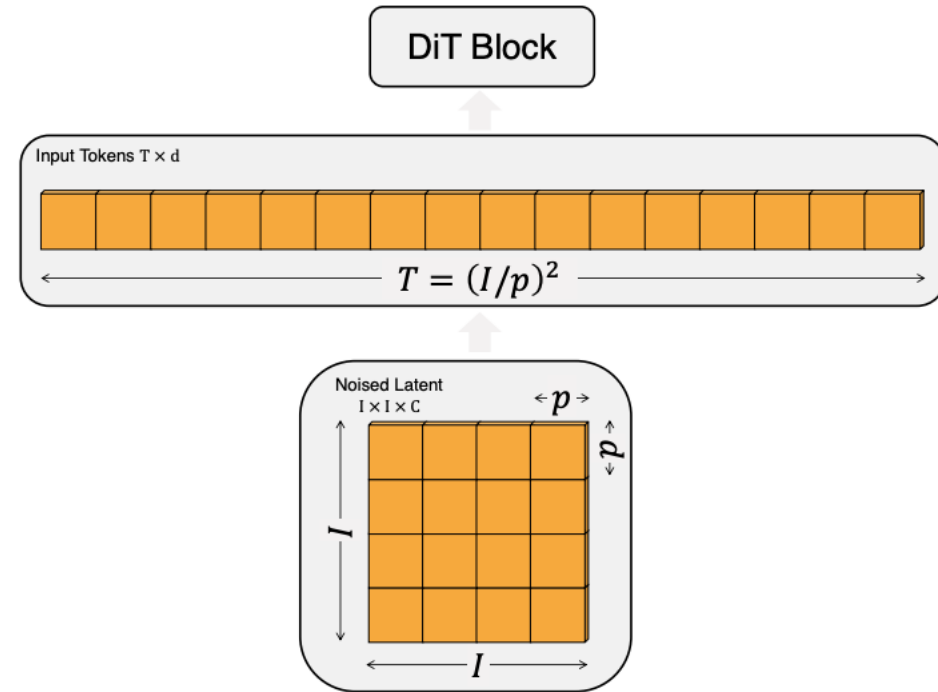
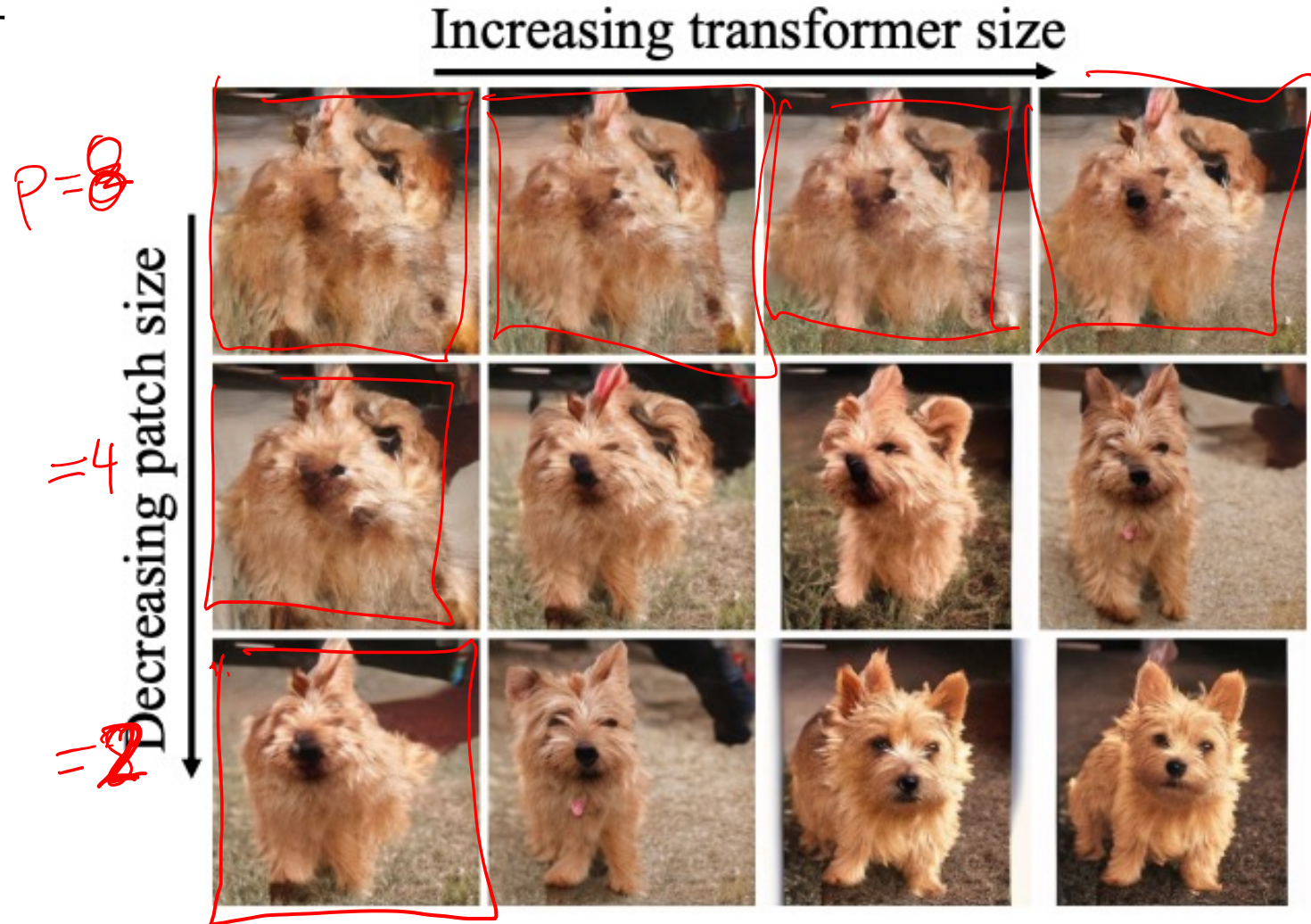


Figure 4. **Input specifications for DiT.** Given patch size  $p \times p$ , a spatial representation (the noised latent from the VAE) of shape  $I \times I \times C$  is “patchified” into a sequence of length  $T = (I/p)^2$  with hidden dimension  $d$ . A smaller patch size  $p$  results in a longer sequence length and thus more Gflops.

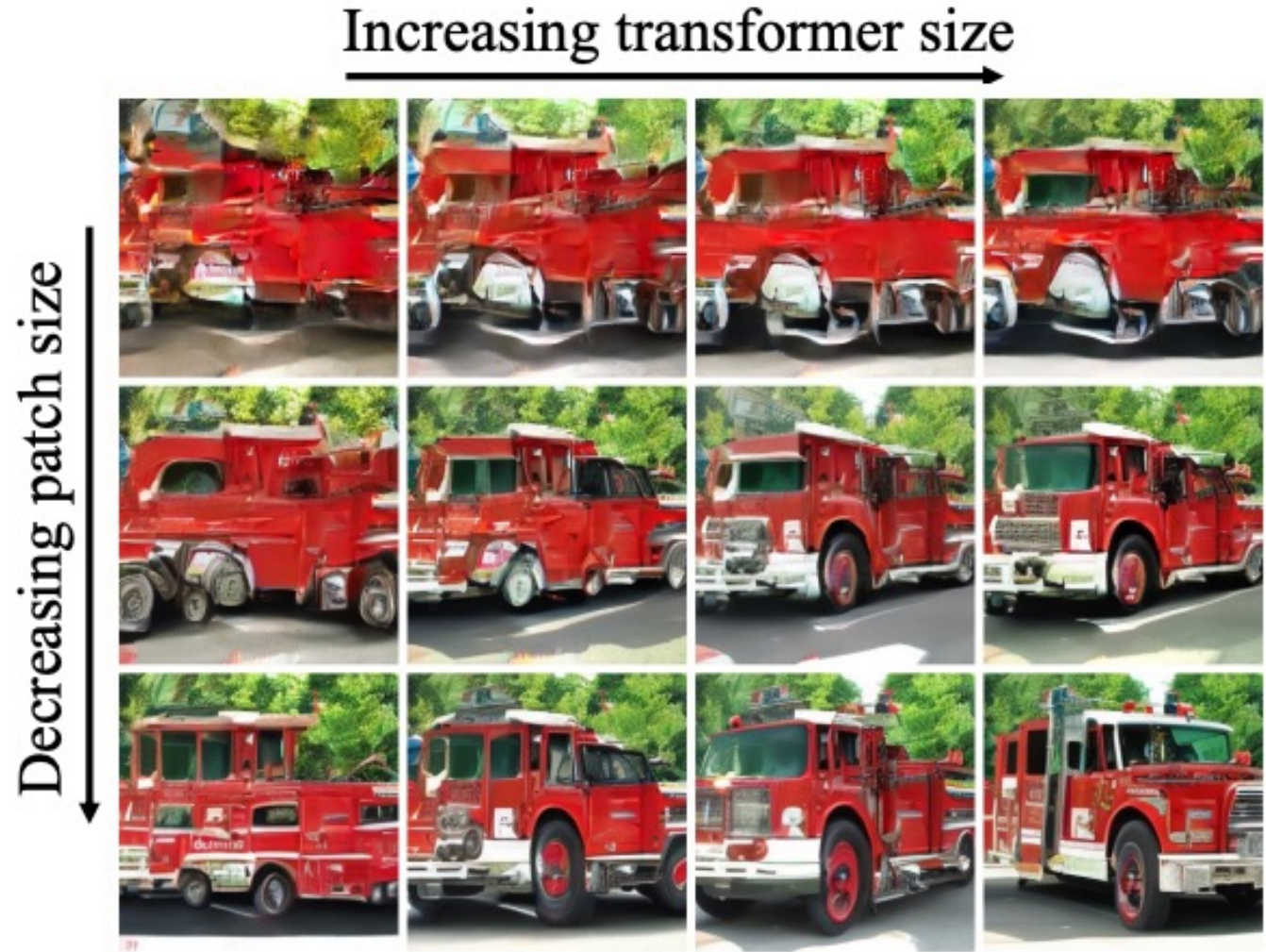
# Results: Diffusion Transformer

- Decreasing the patch size of a DiT increases the GFLOPs even though it does not increase the number of parameters
- So DiT studies the relationship between FID performance and GFLOPs (rather than FID and # of param.s)
- Question:** How much does the total computation increase if we drop the patch size from  $p=4$  to  $p=2$ ?
- Of course, we can also increase the number of model parameters just by increasing the size of the Transformer



# Results: Diffusion Transformer

- Decreasing the patch size of a DiT increases the GFLOPs even though it does not increase the number of parameters
- So DiT studies the relationship between FID performance and GFLOPs (rather than FID and # of param.s)
- **Question:** How much does the total computation increase if we drop the patch size from  $p=4$  to  $p=2$ ?
- Of course, we can also increase the number of model parameters just by increasing the size of the Transformer



# Results: Diffusion Transformer

Scaling results:

1. **GFLOPs and FID are strongly correlated (i.e. if we increase GFLOPs then we get better FID score)**
2. Larger DiT models are more compute efficient than smaller DiT models during training (i.e., if we use the same amount of training computation for both, the larger DiT does better at test time than the smaller DiT)
3. The DiT beats a latent diffusion model (LDM) with a UNet backbone even at the same amount of compute:
  - LDM-4 (103.6 Gflops)
  - DiT-XL/2 (118.6 Gflops)

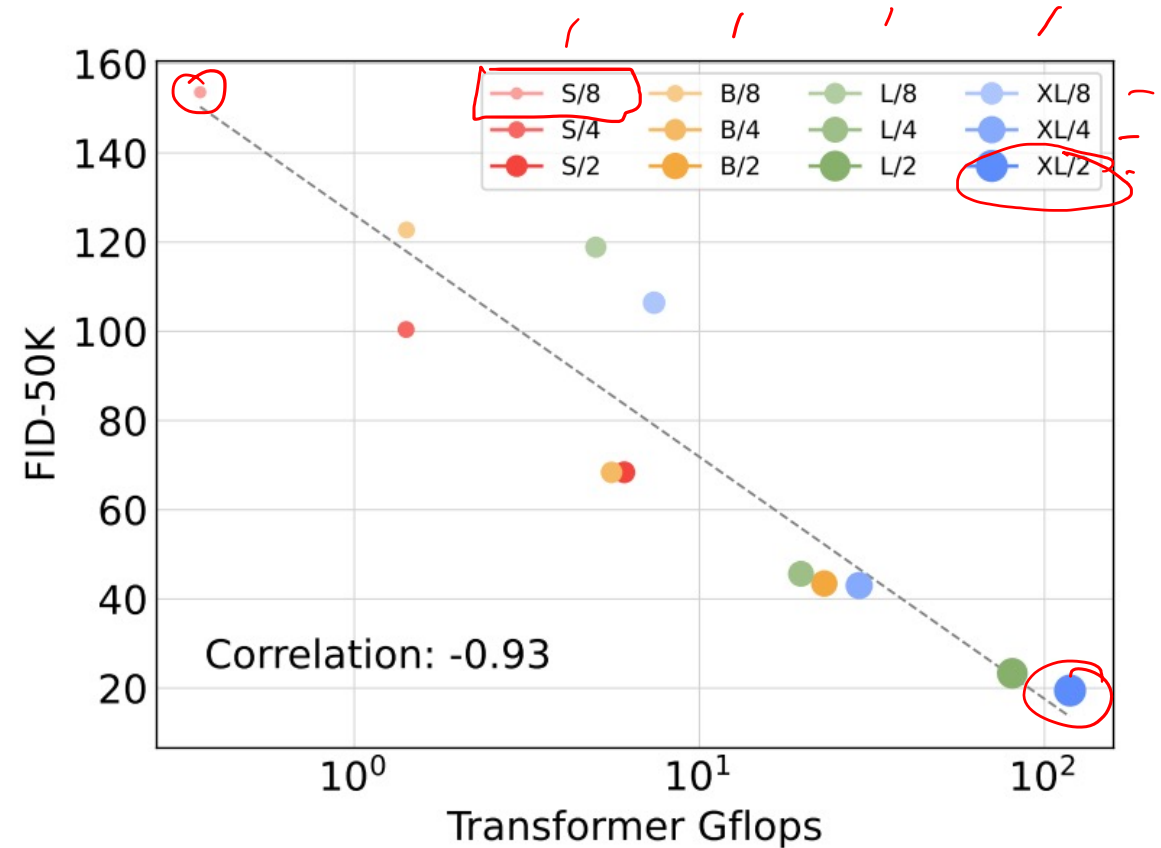


Figure 8. **Transformer Gflops are strongly correlated with FID.** We plot the Gflops of each of our DiT models and each model's FID-50K after 400K training steps.

# Results: Diffusion Transformer

Scaling results:

1. GFLOPs and FID are strongly correlated (i.e. if we increase GFLOPs then we get better FID score)
2. **Larger DiT models are more compute efficient than smaller DiT models during training (i.e., if we use the same amount of training computation for both, the larger DiT does better at test time than the smaller DiT)**
3. The DiT beats a latent diffusion model (LDM) with a UNet backbone even at the same amount of compute:
  - LDM-4 (103.6 Gflops)
  - DiT-XL/2 (118.6 Gflops)

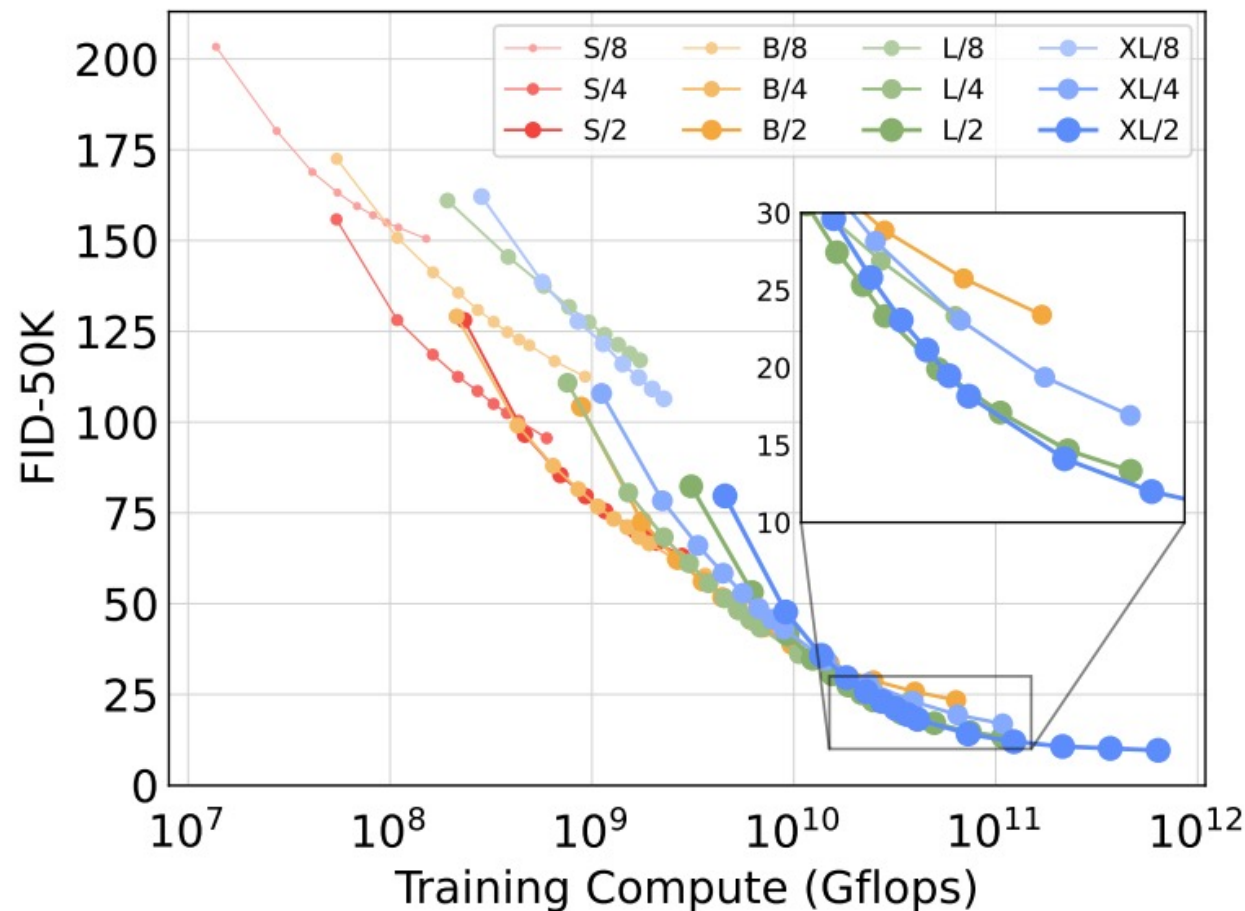


Figure 9. **Larger DiT models use large compute more efficiently.** We plot FID as a function of total training compute.

# Results: Diffusion Transformer

Scaling results:

1. GFLOPs and FID are strongly correlated (i.e. if we increase GFLOPs then we get better FID score)
2. Larger DiT models are more compute efficient than smaller DiT models during training (i.e., if we use the same amount of training computation for both, the larger DiT does better at test time than the smaller DiT)
3. **The DiT beats a latent diffusion model (LDM) with a UNet backbone even at the same amount of compute:**
  - LDM-4 (103.6 Gflops)
  - DiT-XL/2 (118.6 Gflops)

| Class-Conditional ImageNet $256 \times 256$ |             |             |               |             |             |
|---------------------------------------------|-------------|-------------|---------------|-------------|-------------|
| Model                                       | FID↓        | sFID↓       | IS↑           | Precision↑  | Recall↑     |
| BigGAN-deep [2]                             | 6.95        | 7.36        | 171.4         | 0.87        | 0.28        |
| StyleGAN-XL [53]                            | 2.30        | 4.02        | 265.12        | 0.78        | 0.53        |
| ADM [9]                                     | 10.94       | 6.02        | 100.98        | 0.69        | 0.63        |
| ADM-U                                       | 7.49        | 5.13        | 127.49        | 0.72        | 0.63        |
| ADM-G                                       | 4.59        | 5.25        | 186.70        | 0.82        | 0.52        |
| ADM-G, ADM-U                                | 3.94        | 6.14        | 215.84        | 0.83        | 0.53        |
| CDM [20]                                    | 4.88        | -           | 158.71        | -           | -           |
| LDM-8 [48]                                  | 15.51       | -           | 79.03         | 0.65        | 0.63        |
| LDM-8-G                                     | 7.76        | -           | 209.52        | 0.84        | 0.35        |
| LDM-4                                       | 10.56       | -           | 103.49        | 0.71        | 0.62        |
| LDM-4-G (cfg=1.25)                          | 3.95        | -           | 178.22        | 0.81        | 0.55        |
| LDM-4-G (cfg=1.50)                          | 3.60        | -           | 247.67        | <b>0.87</b> | 0.48        |
| <b>DiT-XL/2</b>                             | 9.62        | 6.85        | 121.50        | 0.67        | <b>0.67</b> |
| <b>DiT-XL/2-G (cfg=1.25)</b>                | 3.22        | 5.28        | 201.77        | 0.76        | 0.62        |
| <b>DiT-XL/2-G (cfg=1.50)</b>                | <b>2.27</b> | <b>4.60</b> | <b>278.24</b> | 0.83        | 0.57        |

# PROMPT-TO-PROMPT

# Background: Image Editing

- Fixing the Random Seed:
  - A simple baseline for image editing with text: change part of the prompt, **keep the random seed fixed** (e.g. the noise at the start of diffusion), and then run diffusion sampler
  - Problem: the entire **structure** of the image may **change dramatically**
  - Doesn't feel like “editing” at all, more like generation of **unrelated images**
- Mask-based Image Editing:
  - standard approaches to text-based image editing typically **require an image mask** as well
  - the **mask** specifies which part of the image should remain **unchanged**
  - then the **text prompt** informs how the **unmasked part** should be adapted (e.g. by a diffusion model)
  - (Example: Blended Diffusion)



# Background: Image Editing

- Fixing the Random Seed:

- A simple baseline for image editing with text: change part of the prompt, **keep the random seed fixed** (e.g. the noise at the start of diffusion), and then run
- Problem: the composition is inconsistent in various ways: the background, image mask
- Doesn't fix whole cake vs. single slice, or like

the composition is inconsistent in various ways: the background, whole cake vs. single slice, or how much cake is in view



- Mask-based Image Editing:

- standard approaches to text-based image editing typically **require an image mask** as well
- the **mask** specifies which part of the image should be **unchanged** (the prompt informs how the rest should be changed by the diffusion model)

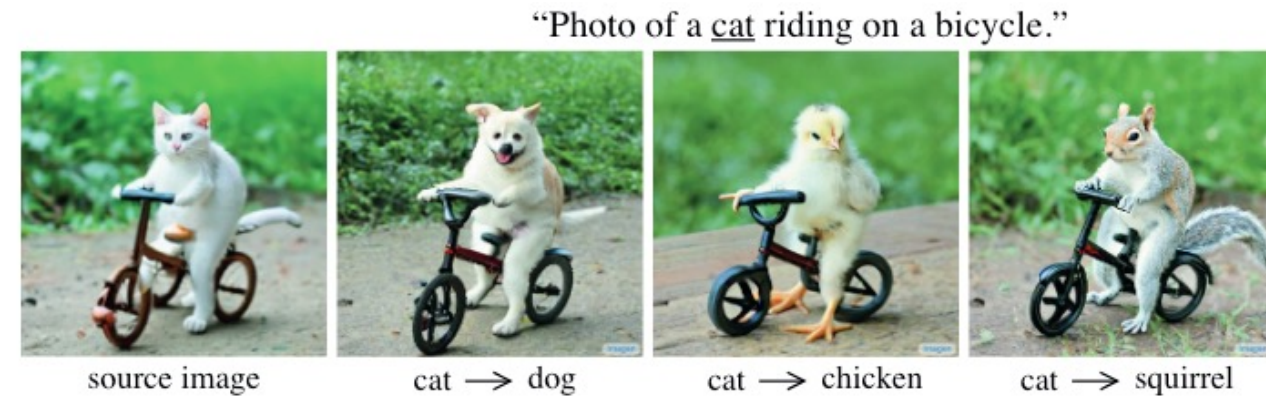
here the composition remains consistent across images



# Prompt-to-Prompt

## Prompt-to-Prompt:

- **Goal:** edit images with text only and do not require the user to provide a mask
- **Key Idea:**
  - **given** pre-trained latent diffusion model
  - run diffusion model with **original prompt** and **store** the attention weights and cross-attention weights (from the pixels back to the text)
  - re-run diffusion with **edited prompt**, but (carefully) **copy in** the cross-attention weights from the previous run
  - exactly *how* to copy in the attention weights depends on the type of edit
- **Inference only:** no training is involved! we only modify how the samples are drawn from the pre-trained latent diffusion model

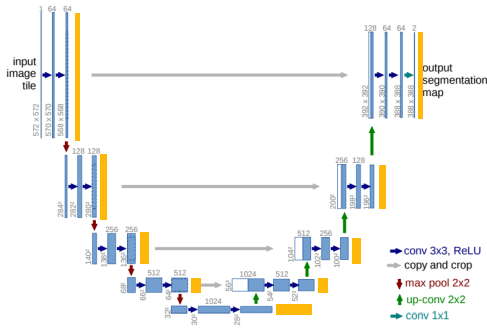


the composition remains consistent across images,  
but with only the text for guidance (no mask)

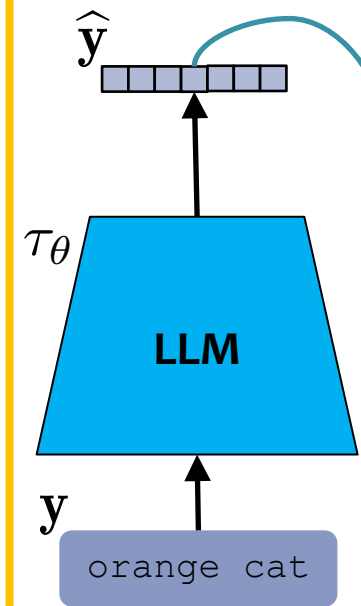
# Latent Diffusion Model (LDM)

Prompt-to-prompt: (1) assumes we have a pretrained latent diffusion model and (2) does no parameter estimation

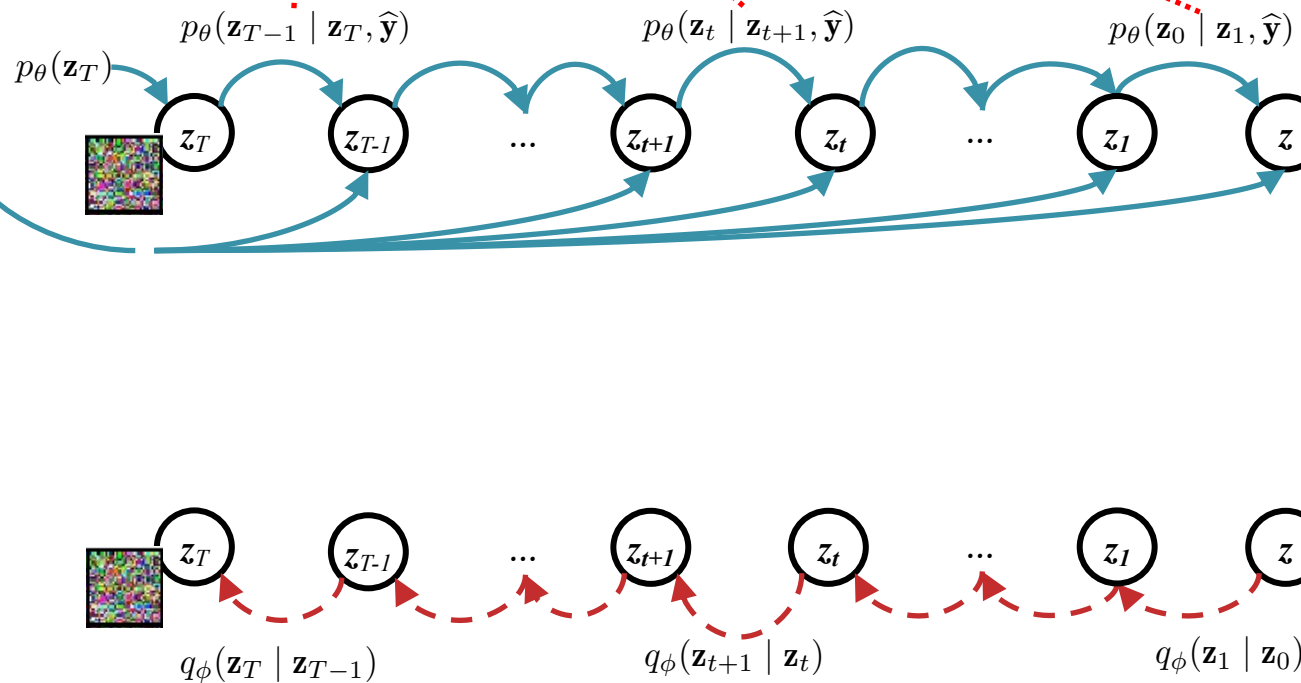
UNet w/cross  
attention



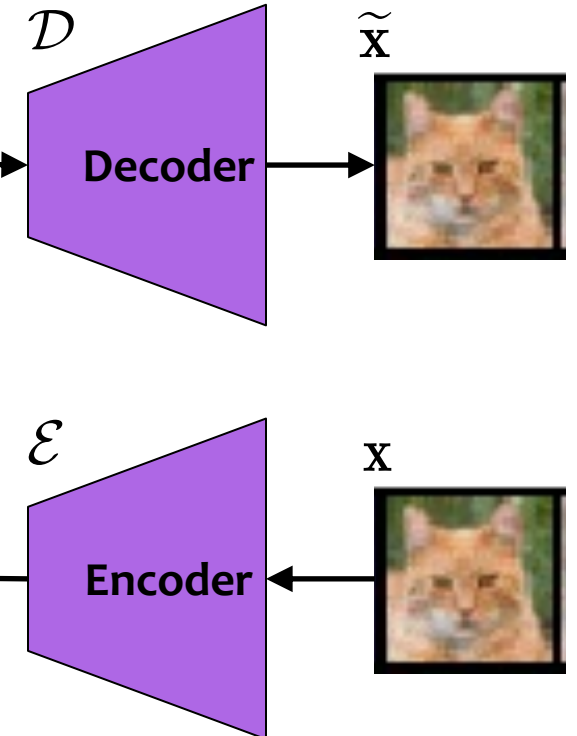
prompt space



latent space

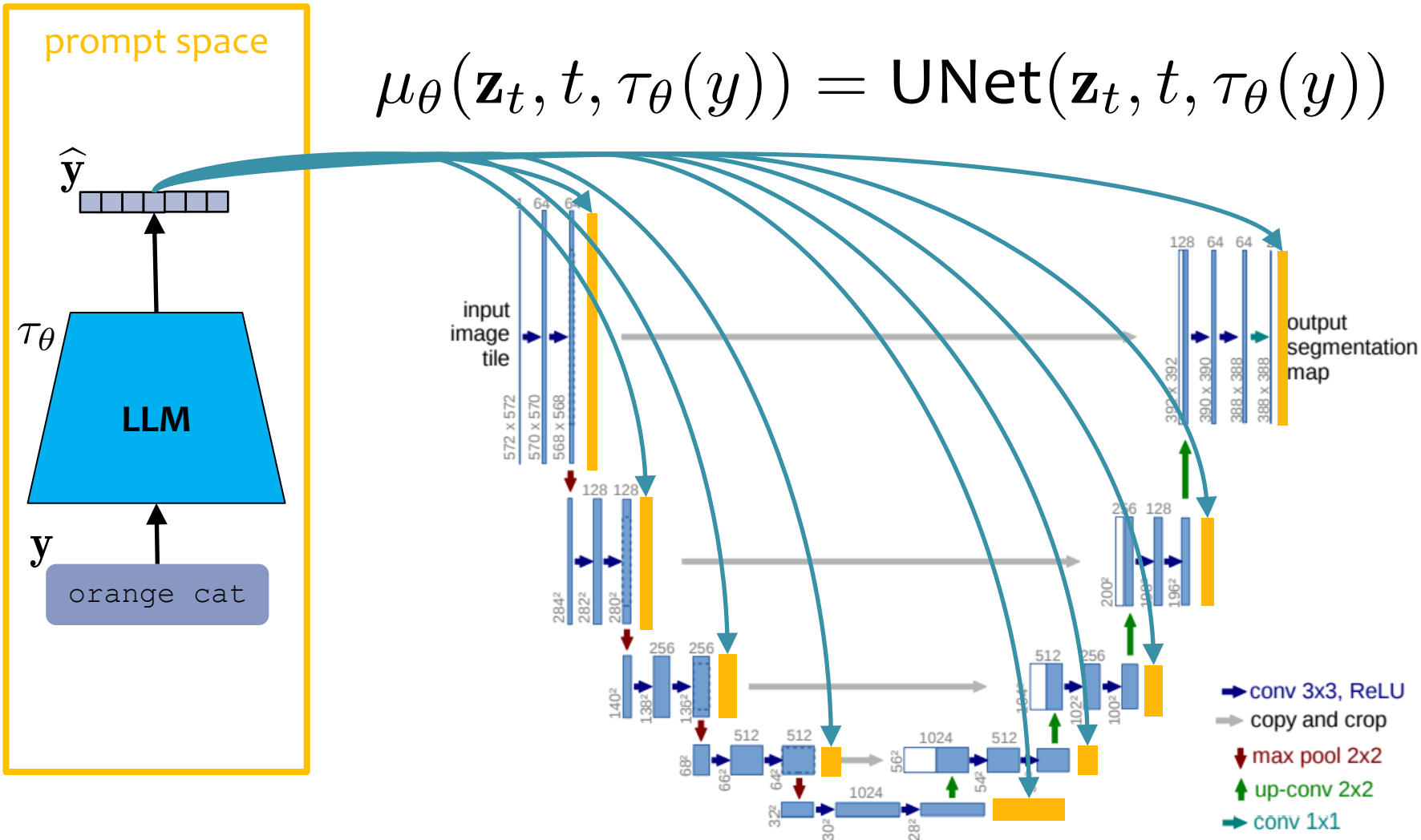


pixel space



# LDM: Noise Model

$$\mu_{\theta}(\mathbf{z}_t, t, \tau_{\theta}(y)) = \text{UNet}(\mathbf{z}_t, t, \tau_{\theta}(y))$$



- The noise model includes **cross attention** (yellow boxes) to the representation of the prompt text
- During training we optimize both the parameters of the UNet noise model and the parameters of the LLM simultaneously

Prompt-to-prompt modifies the cross attention in LDM, which looks at both the text encoding and the (latent) representation of the image

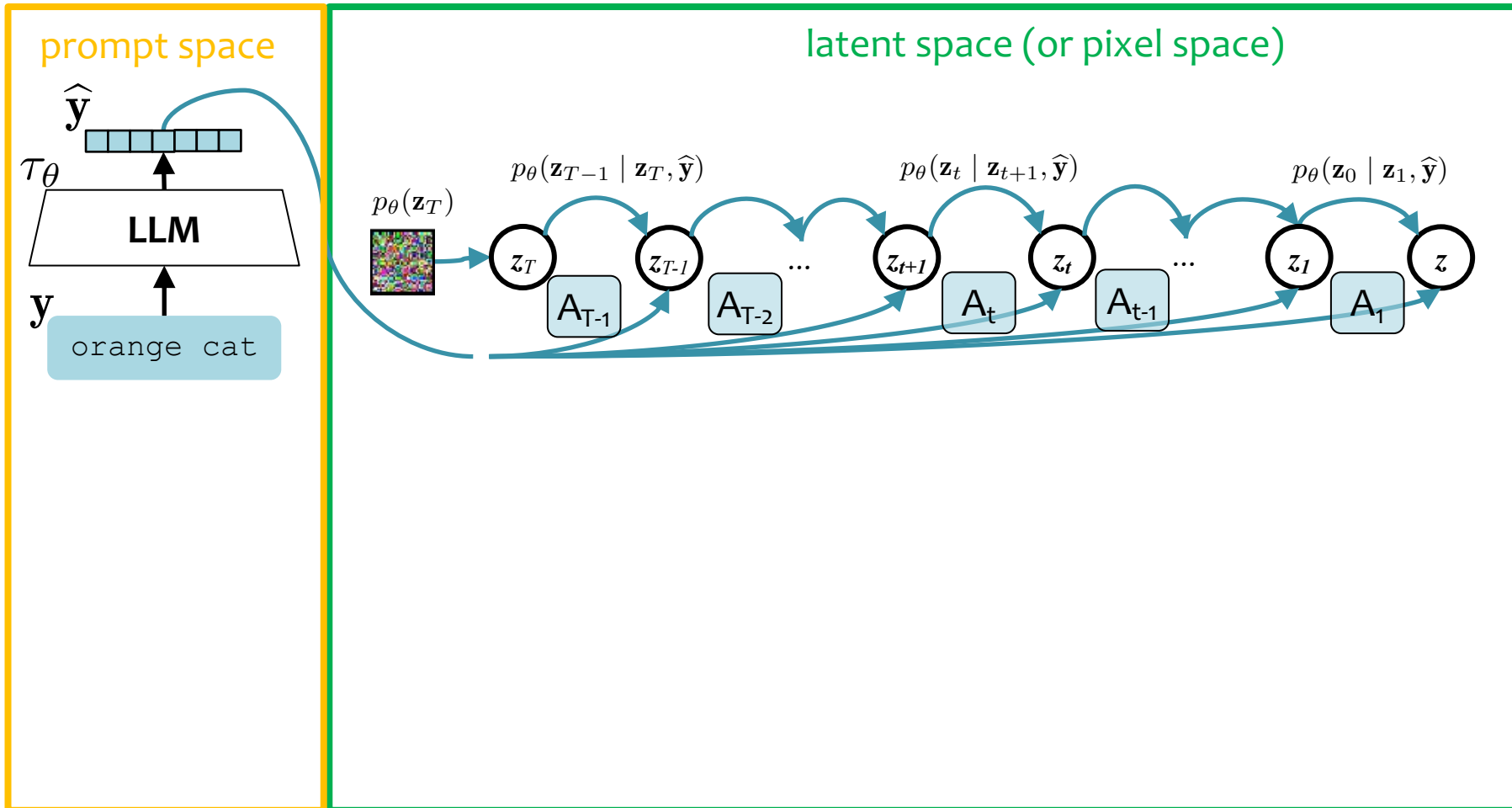
# Prompt-to-Prompt: Editing Cross Attention

## Prompt-to-Prompt:

- **Goal:** edit images with text only and do not require the user to provide a mask
- **Key Idea:**
  - **given** pre-trained latent diffusion model
  - run diffusion model with **original prompt** and **store** the attention weights and cross-attention weights (from the pixels back to the text)
  - re-run diffusion with **edited prompt**, but (carefully) **copy in** the cross-attention weights from the previous run
  - exactly *how* to copy in the attention weights depends on the type of edit
- **Inference only:** no training is involved! we only modify how the samples are drawn from the pre-trained latent diffusion model

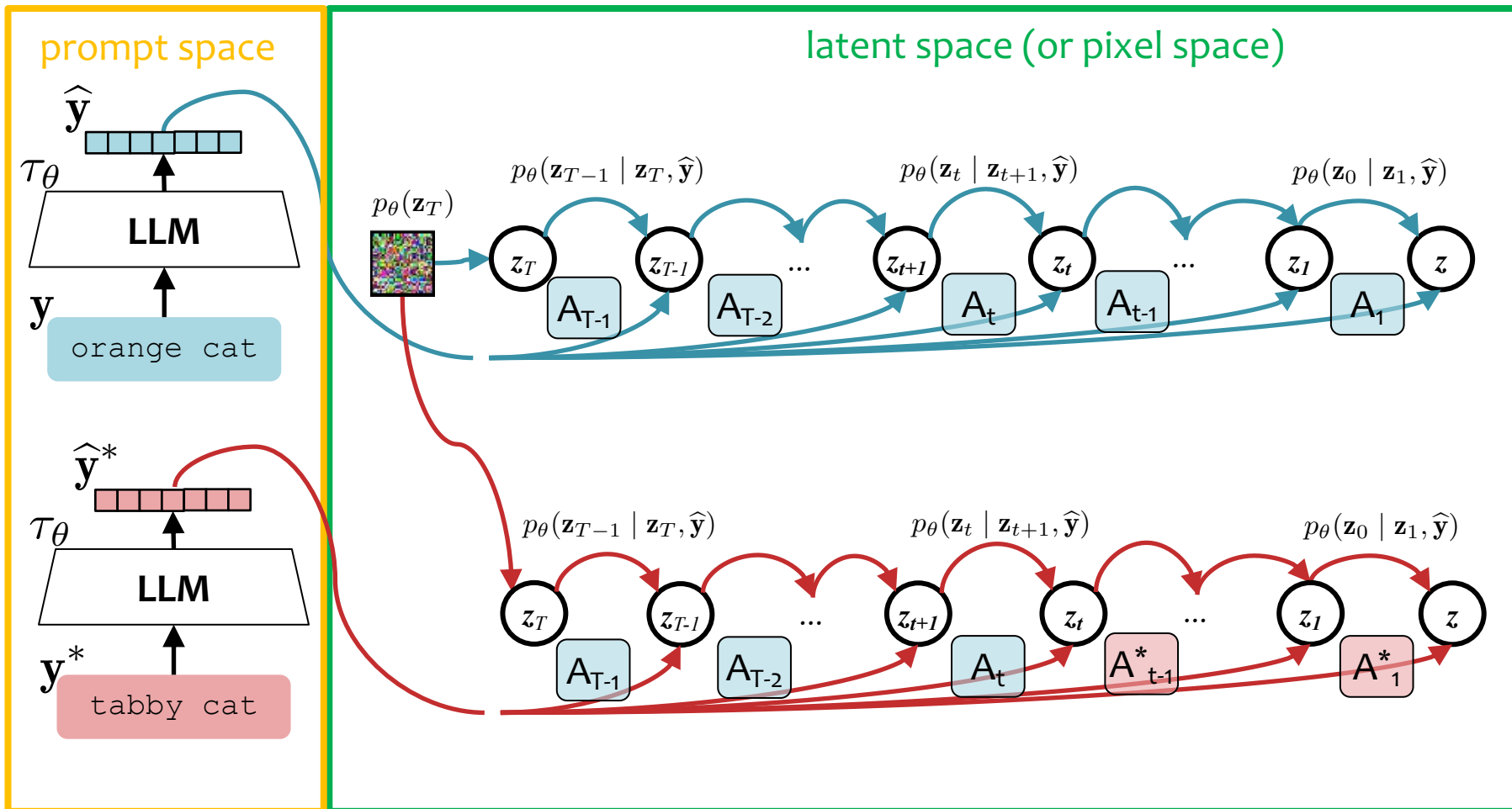
# Prompt-to-Prompt: Editing Cross Attention

1. encode the original prompt  $y$
2. run diffusion on  $y$  and obtain attention weights  $A_{T-1}, \dots, A_1$



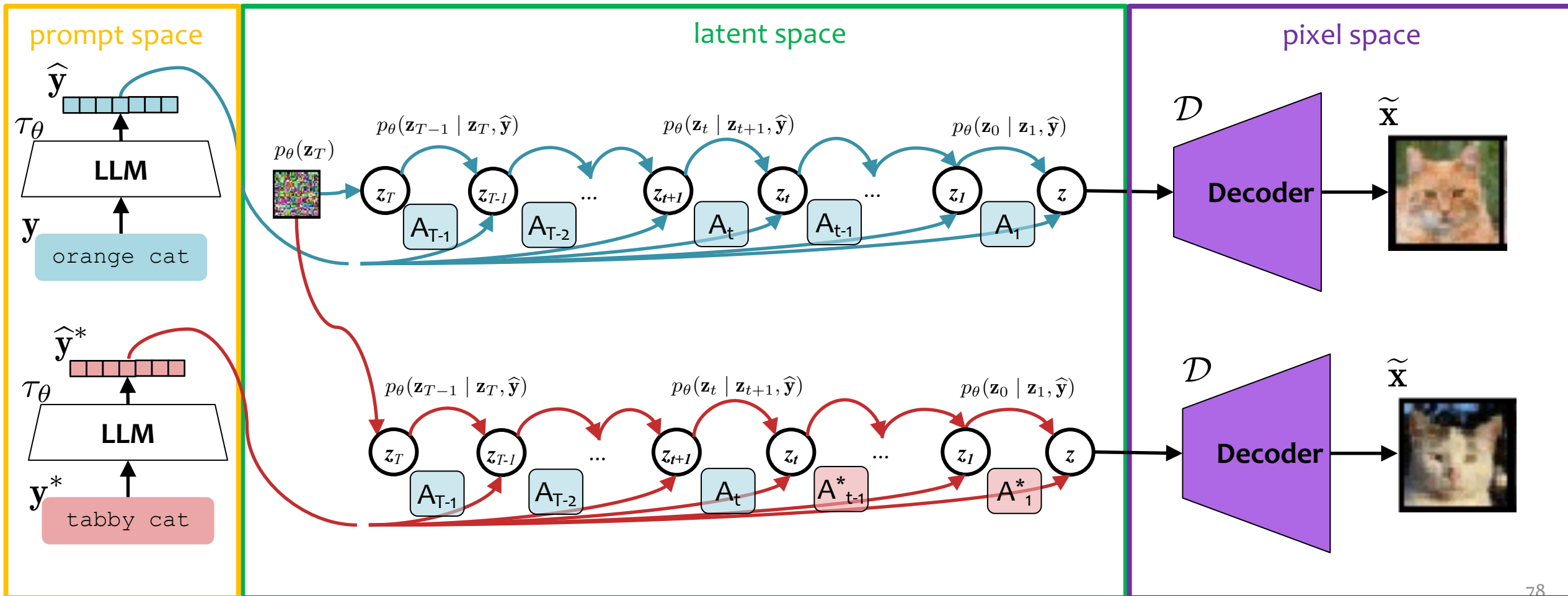
# Prompt-to-Prompt: Editing Cross Attention

1. encode the original prompt  $y$
2. run diffusion on  $y$  and obtain attention weights  $A_{T-1}, \dots, A_1$
3. encode the modified prompt  $y^*$
4. run diffusion again
  - a) reuse the noise  $z_T$  from the original run
  - b) use the attention weights from the original run until timestep  $\tau$   $A_{T-1}, \dots, A_t$
  - c) then switch to using attention weights from this current run  $A_{t-1}^*, \dots, A_1^*$
  - d) regardless of which attention weights, you still attend to  $y^*$



# Prompt-to-Prompt: Editing Cross Attention

5. if running in latent space, then use decoder to recover pixel space representation

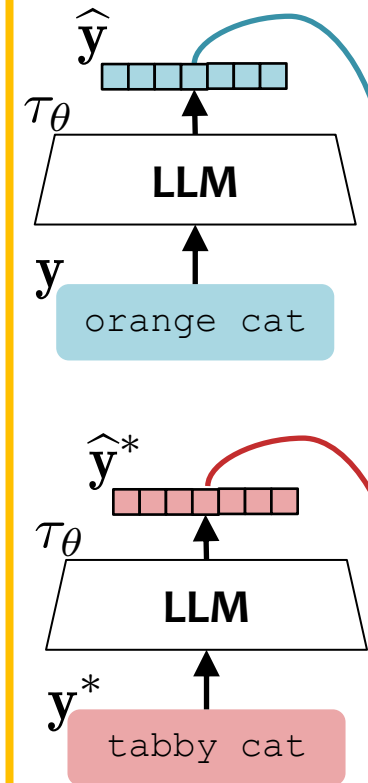


## Question:

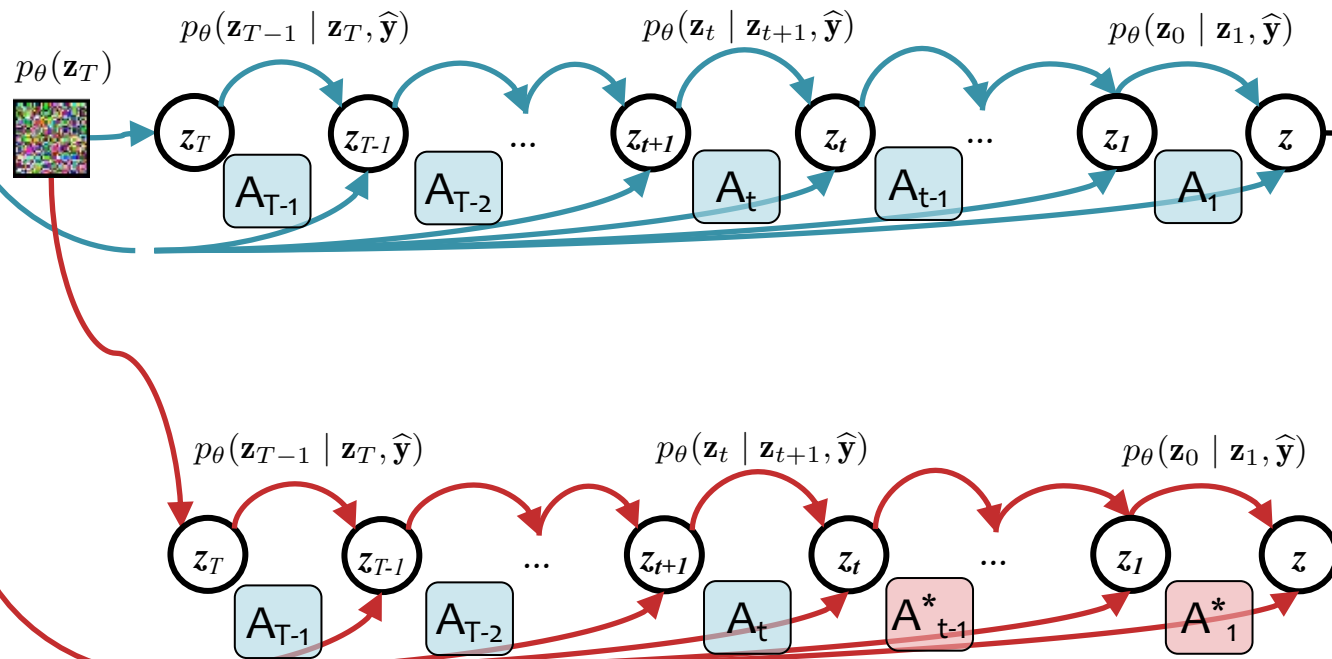
Why do we use from the original attention weights for a while before swapping to the new attention weights?

## Answer:

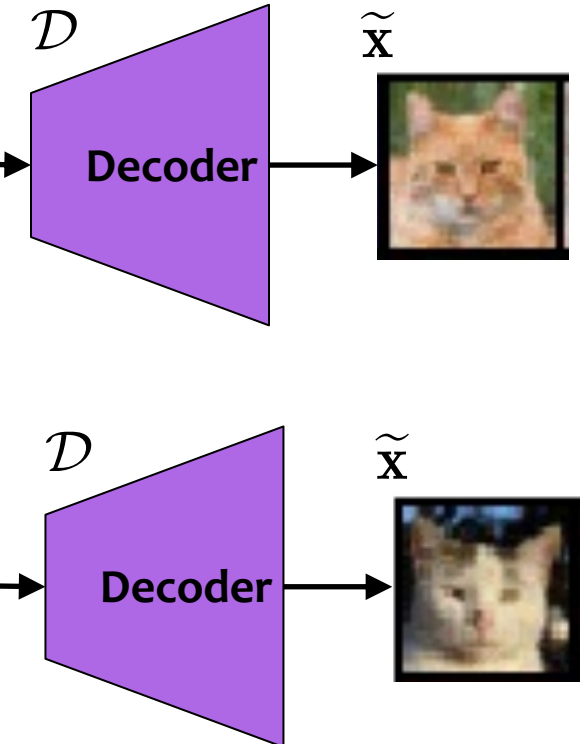
prompt space



latent space



pixel space



# Prompt-to-Prompt: Editing Cross Attention

## Prompt-to-Prompt:

- **Goal:** edit images with text only and do not require the user to provide a mask
- **Key Idea:**
  - **given** pre-trained latent diffusion model
  - run diffusion model with **original prompt** and **store** the attention weights and cross-attention weights (from the pixels back to the text)
  - re-run diffusion with **edited prompt**, but (carefully) **copy in** the cross-attention weights from the previous run
  - exactly how to copy in the attention weights depends on the type of edit
- **Inference only:** no training is involved! we only modify how the samples are drawn from the pre-trained latent diffusion model

---

## Algorithm 1 Prompt-to-Prompt image editing

---

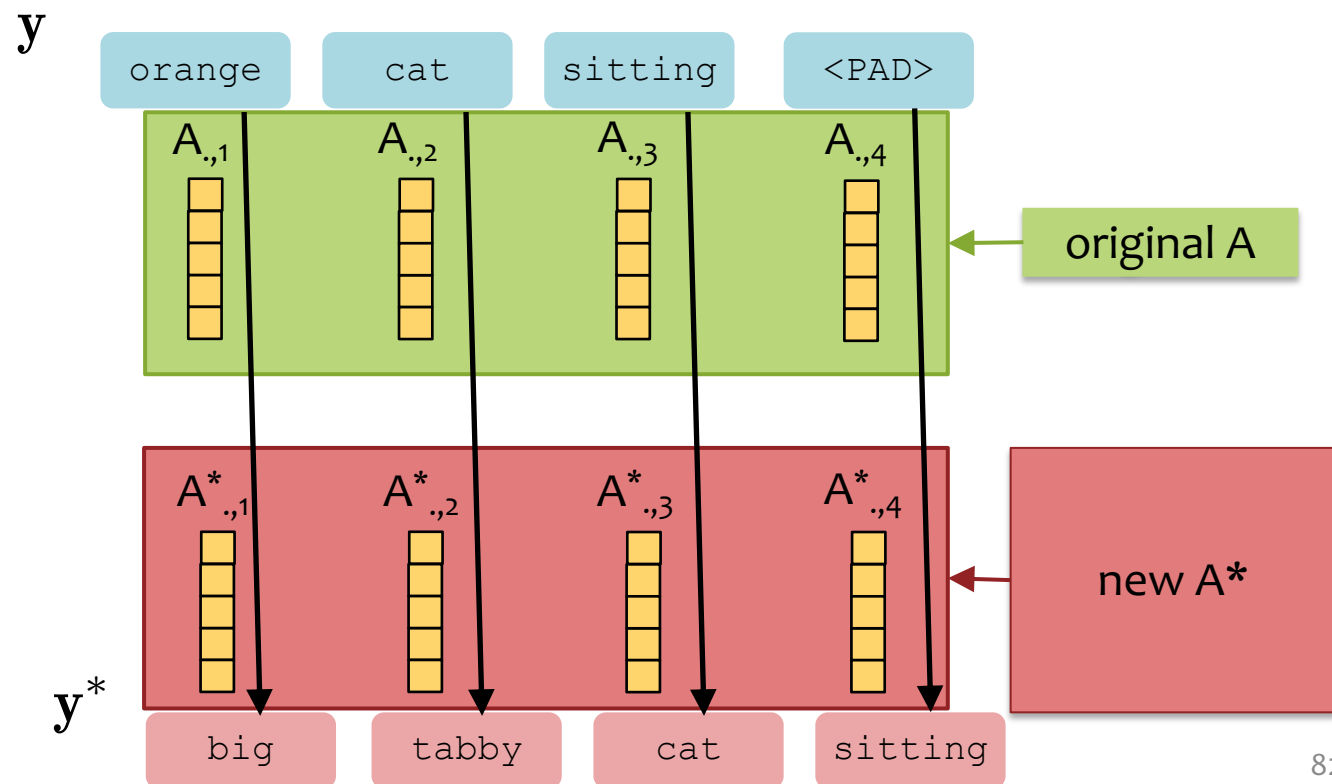
```
1: Input: A source prompt  $y$ , a target prompt  $y^*$ , and a random seed  $s$ .
2: Output: A source image  $x_{src}$  and an edited image  $x_{dst}$ .
3:  $z_T \sim \mathcal{N}(0, I)$  a unit Gaussian random variable with random seed  $s$ ;
4:  $z_T^* \leftarrow z_T'$ ;
5: for  $t = T, T - 1, \dots, 1$  do
6:    $z_{t-1}, \underline{A}_t \leftarrow DM(z_t, \underline{y}, t, s)$ ;  $z_t$ 
7:    $A_t^* \leftarrow DM(z_t^*, \underline{y}^*, t, s)$ ;
8:    $\hat{A}_t \leftarrow \text{Edit}(A_t, A_t^*, t)$ ;
9:    $z_{t-1}^* \leftarrow DM(z_t^*, y^*, t, s, t) \{A \leftarrow \hat{A}_t\}$ ;
10: return  $(z_0, z_0^*)$ 
```

---

$$\text{Edit}(A_t, A_t^*, t) := \begin{cases} A_t^* & \text{if } t < \tau \\ A_t & \text{otherwise.} \end{cases}$$

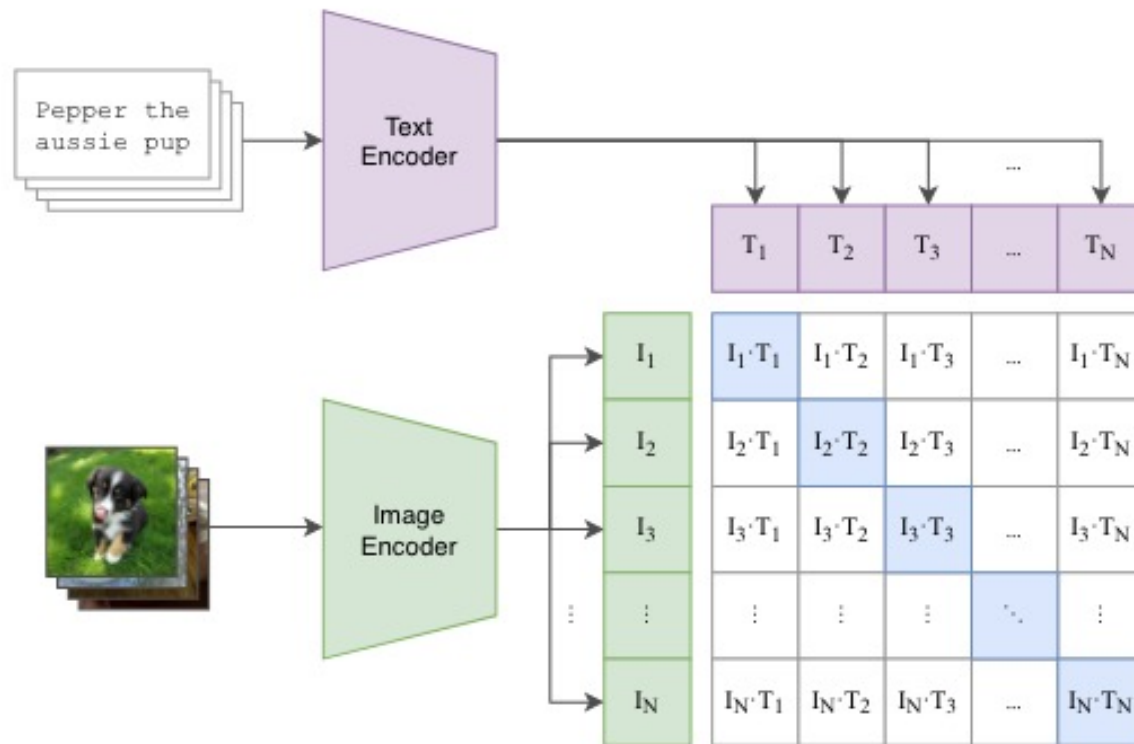
# Attention Swapping

- Problem: What if  $A_t$  and  $A_t^*$  are not the same shape?
- Solution: Swap in just the appropriate parts!
  - The dimension in latent space will always remain constant (e.g. 1024)
  - The dimension in text prompt space also remain constant if we use a fixed length encoder
    - e.g. length = 77, if we use CLIP encoder
    - orange cat sitting <PAD> <PAD>  
... <PAD>
  - However, the words might not align properly!
- Example:
  - we replace “orange” with “big tabby”
  - then copy the attention weights for “orange” to both “big” and “tabby” in the new attention weights

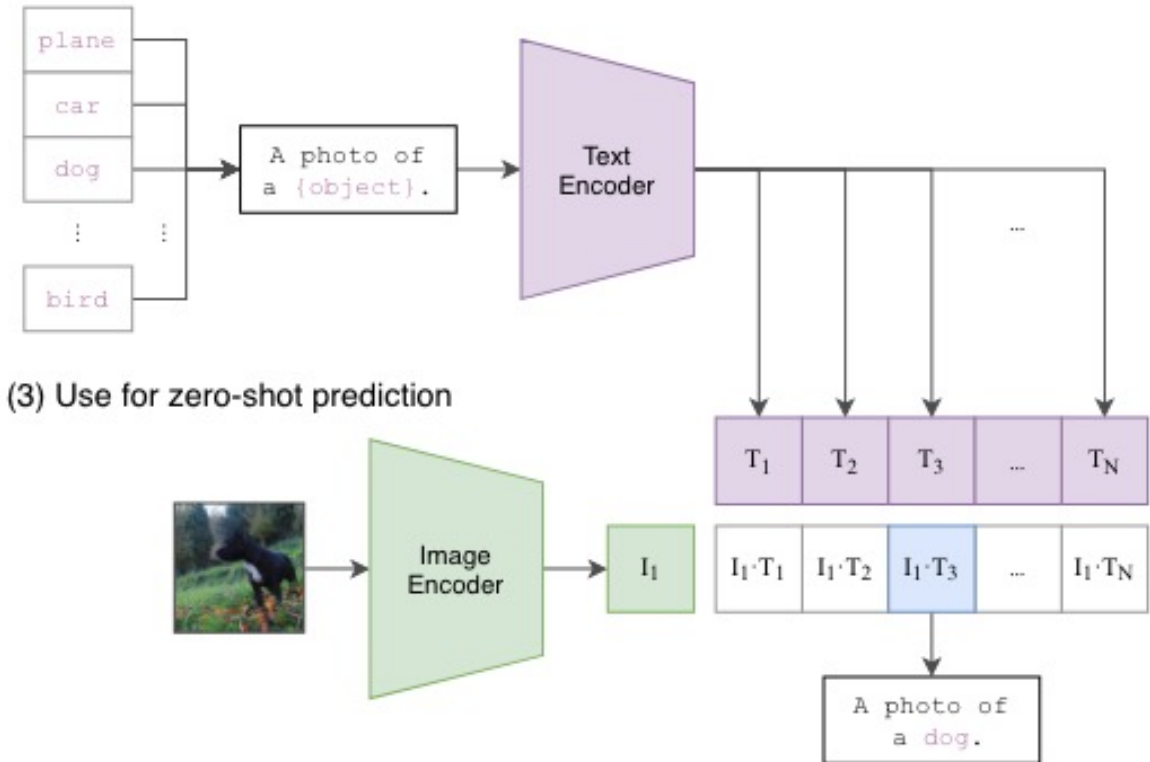


# CLIP (the text encoder for Prompt to Prompt)

(1) Contrastive pre-training



(2) Create dataset classifier from label text

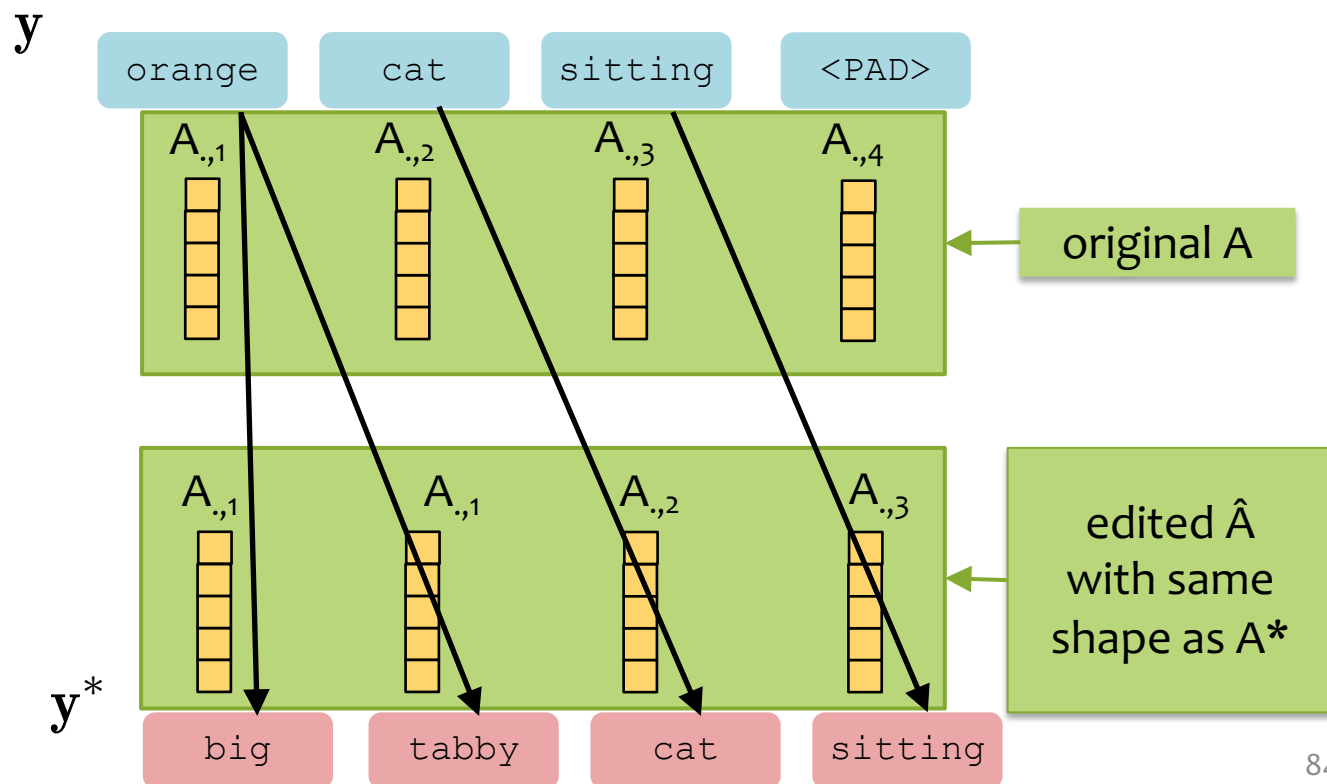


(3) Use for zero-shot prediction

*Figure 1. Summary of our approach. While standard image models jointly train an image feature extractor and a linear classifier to predict some label, CLIP jointly trains an image encoder and a text encoder to predict the correct pairings of a batch of (image, text) training examples. At test time the learned text encoder synthesizes a zero-shot linear classifier by embedding the names or descriptions of the target dataset's classes.*

# Attention Swapping

- Problem: What if  $A_t$  and  $A_t^*$  are not the same shape?
- Solution: Swap in just the appropriate parts!
  - The dimension in latent space will always remain constant (e.g. 1024)
  - The dimension in text prompt space also remain constant if we use a fixed length encoder
    - e.g. length = 77, if we use CLIP encoder
    - orange cat sitting <PAD> <PAD>  
... <PAD>
  - However, the words might not align properly!
- Example:
  - we replace “orange” with “big tabby”
  - then copy the attention weights for “orange” to both “big” and “tabby” in the new attention weights



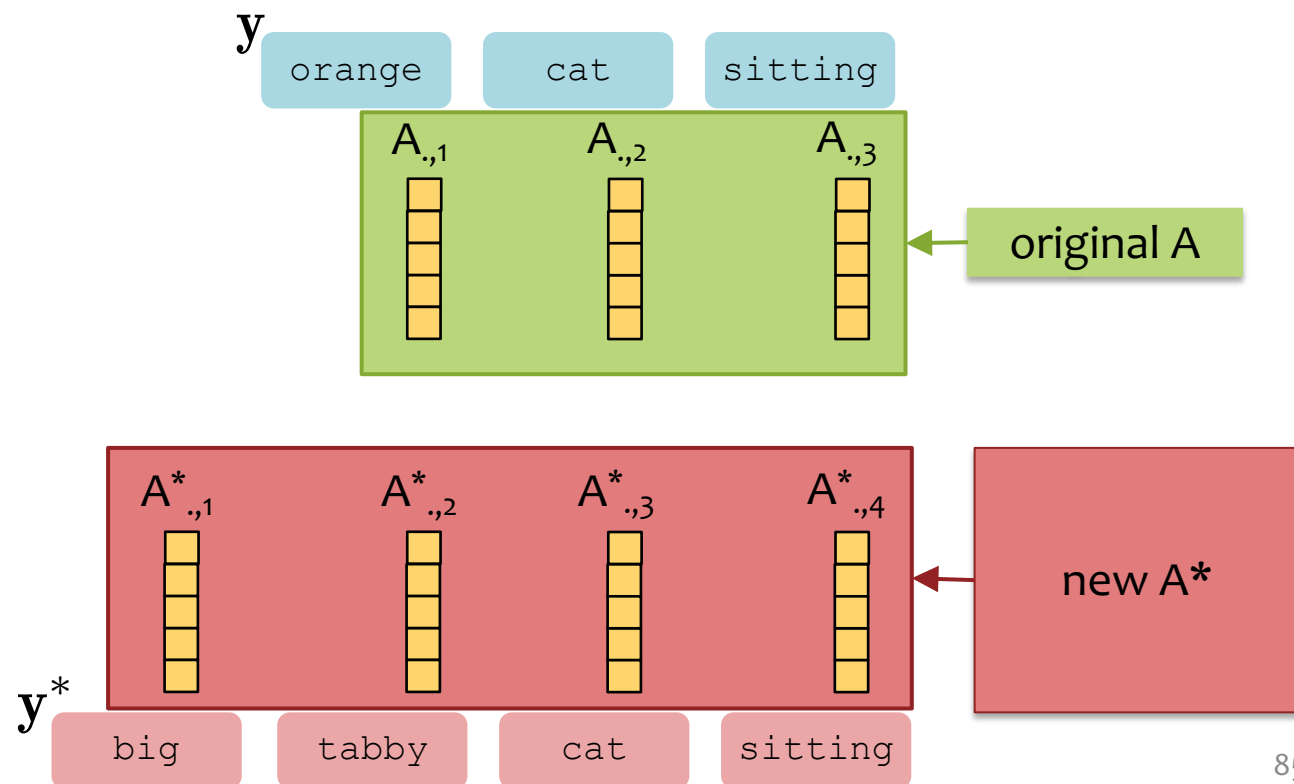
# Attention Swapping

- Problem: What if  $A_t$  and  $A^*_t$  are not the same shape?
- Solution: Swap in just the appropriate parts!
  - The dimension in latent space will always remain constant (e.g. 1024)
  - The dimension in text prompt space also remain constant if we use a fixed length encoder

- e.g. length = 77, if we use CLIP encoder
- orange cat sitting <PAD> <PAD>  
... <PAD>

- However, the words might not align properly!

- Example:
  - we replace “orange” with “big tabby”
  - then copy the attention weights for “orange” to both “big” and “tabby” in the new attention weights



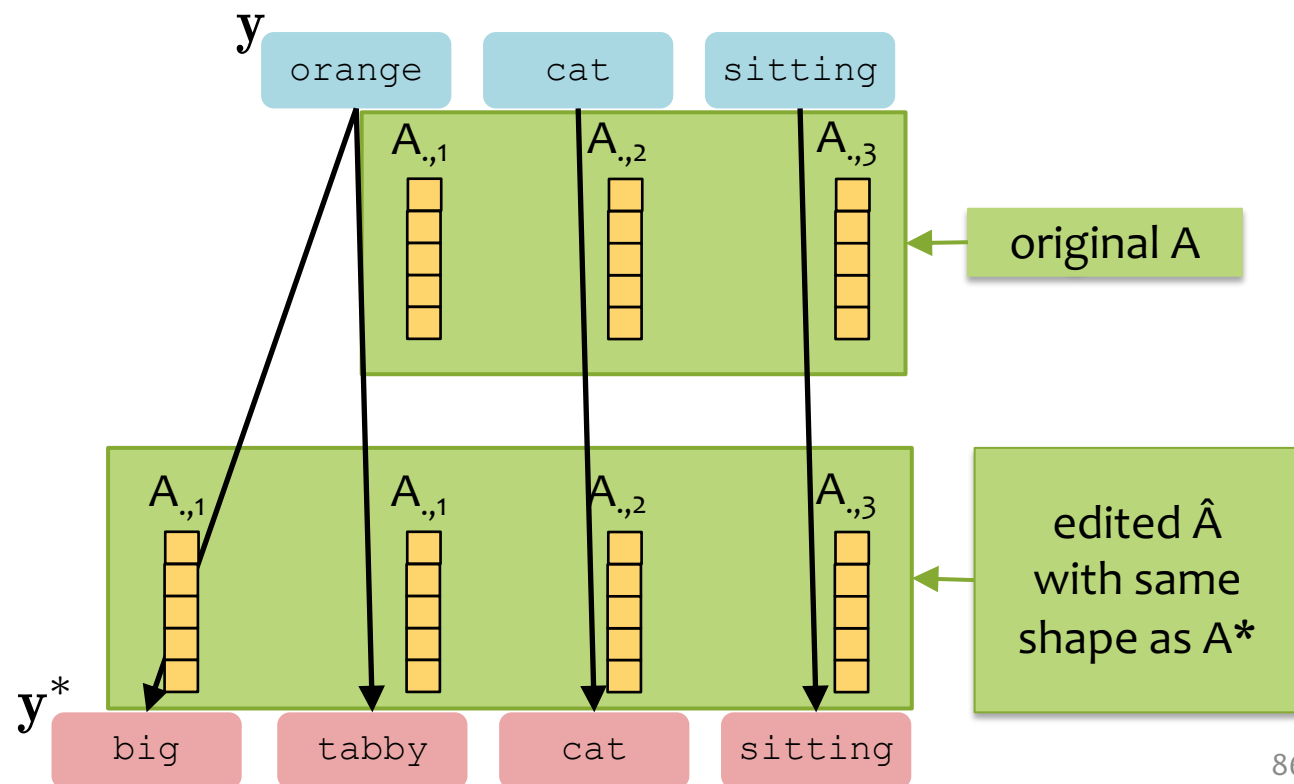
# Attention Swapping

- Problem: What if  $A_t$  and  $A_t^*$  are not the same shape?
- Solution: Swap in just the appropriate parts!
  - The dimension in latent space will always remain constant (e.g. 1024)
  - The dimension in text prompt space also remain constant if we use a fixed length encoder

- e.g. length = 77, if we use CLIP encoder
- orange cat sitting <PAD> <PAD>  
... <PAD>

- However, the words might not align properly!

- Example:
  - we replace “orange” with “big tabby”
  - then copy the attention weights for “orange” to both “big” and “tabby” in the new attention weights



# Prompt-to-Prompt Results

- word/phrase cross-attention swapping automatically identifies the regions of the image that need to remain constant and those that should be adapted



Figure 5: Object preservation. By injecting only the attention weights of the word “butterfly”, taken from the top-left image, we can preserve the structure and appearance of a single item while replacing its context. Note how the butterfly sits on top of all objects in a very plausible manner.

# Prompt-to-Prompt Results

- varying the moment of the attention swap to  $A^*$  allows us to see the effect of our cross-attention manipulation

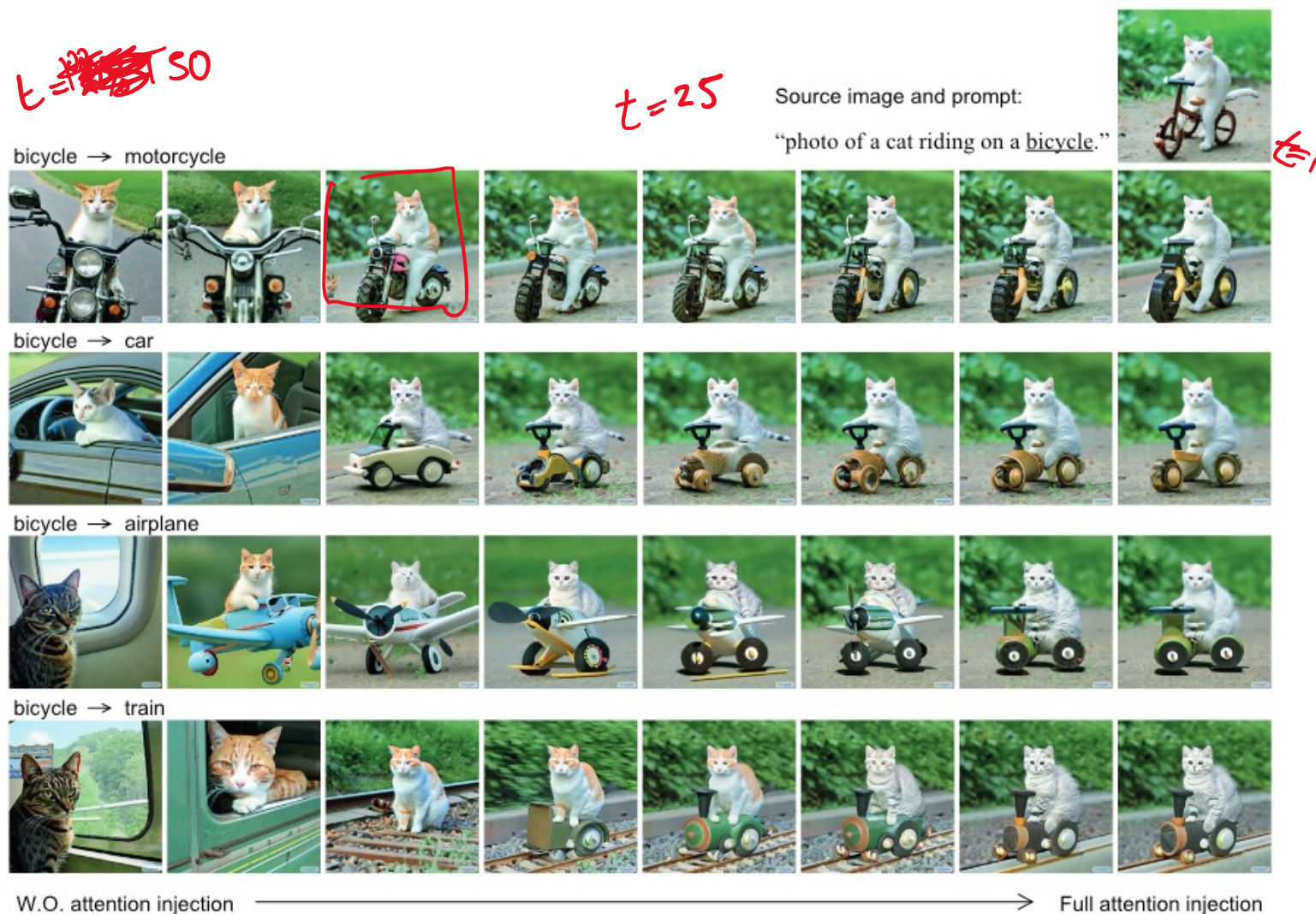
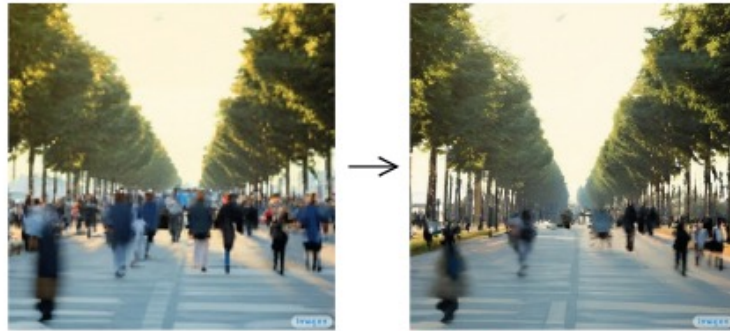


Figure 6: Attention injection through a varied number of diffusion steps. On the top, we show the source image and prompt. In each row, we modify the content of the image by replacing a single word in the text and injecting the cross-attention maps of the source image ranging from 0% (on the left) to 100% (on the right) of the diffusion steps. Notice that on one hand, without our method, none of the source image content is guaranteed to be preserved. On the other hand, injecting the cross-attention throughout all the diffusion steps may over-constrain the geometry, resulting in low fidelity to the text prompt, e.g., the car (3rd row) becomes a bicycle with full cross-attention injection.

# Prompt-to-Prompt Results

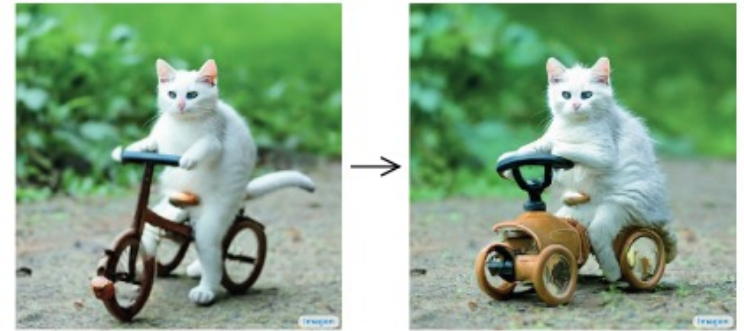
- So far we've focused on swapping one word/phrase for another
- Prompt-to-prompt supports different types of edits
- Different types of edits are achieved through different manipulations of cross-attention weights

down-weight existing descriptor in the prompt

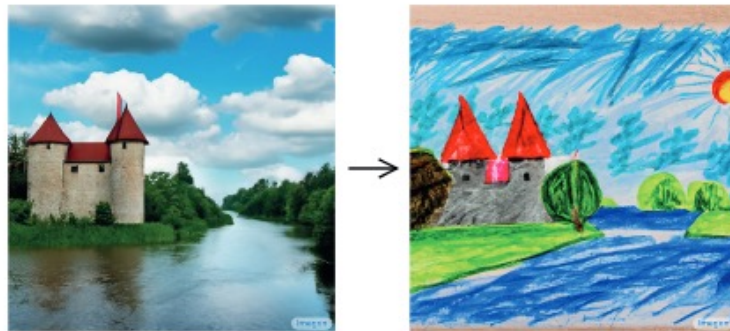


"The boulevards are crowded today."

swap one word for another



"Photo of a cat riding on a bicycle."



"Children drawing of a castle next to a river."

phrase insertion for style change



"a cake with decorations."

phrase insertion for content change