# Recitation: HW4

10-423/10-623 Generative AI

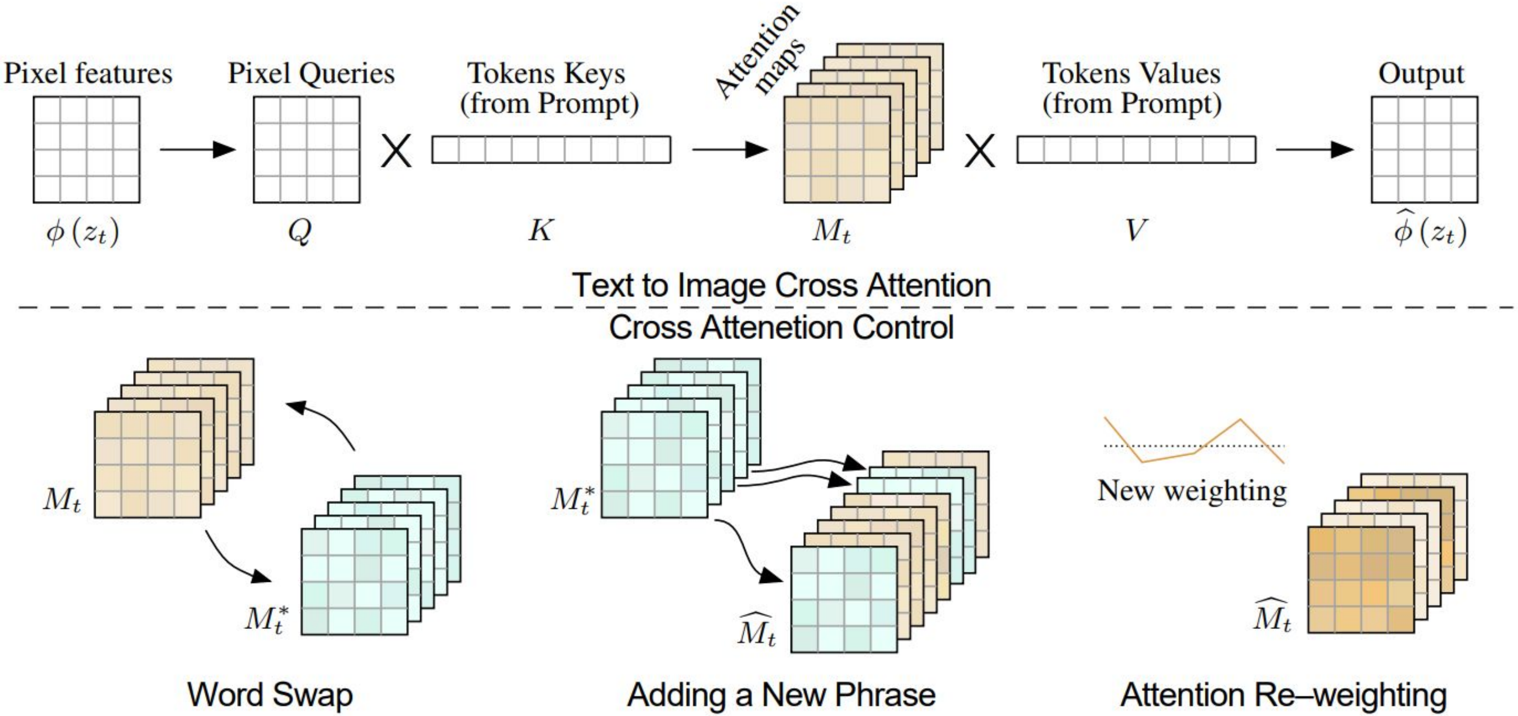
# Outline

- Prompt to prompt
- Diffuser API
- Code run-through
  - prompt2prompt.py
  - run_in_colab.ipynb
- Expected outputs

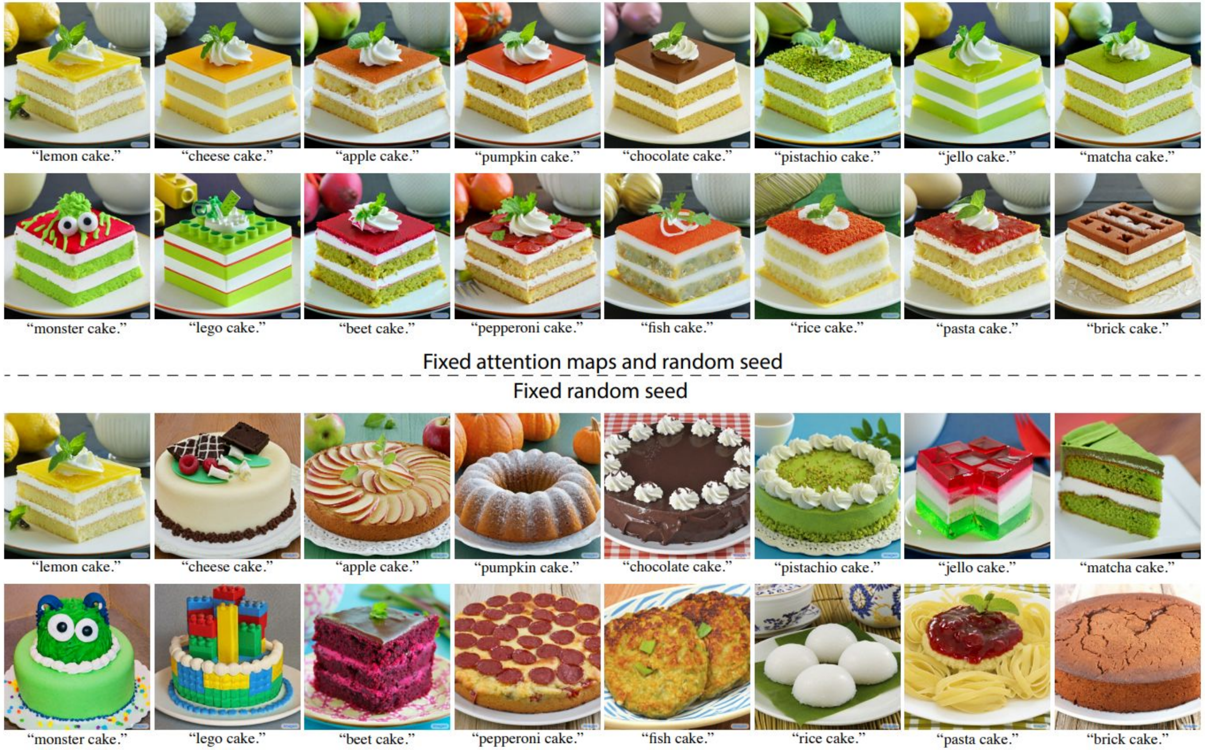
# Prompt to prompt

- The spatial layout and geometry of the generated image depend on the cross-attention maps.
- The composition is determined in the early steps of the diffusion process.

# Method


Pixel features
$\phi(z_t)$
Pixel Queries
$Q$
X
Tokens Keys
(from Prompt)
$K$
Attention maps
$M_t$
X
Tokens Values
(from Prompt)
$V$
Output
$\widehat{\phi}(z_t)$
Text to Image Cross Attention
Cross Atteneti on Control
$M_t$
$M_t^*$
Word Swap
$M_t^*$
$\widehat{M}_t$
Adding a New Phrase
New weighting
$\widehat{M}_t$
Attention Re–weighting

# Replacement



"lemon cake." "cheese cake." "apple cake." "pumpkin cake." "chocolate cake." "pistachio cake." "jello cake." "matcha cake."

"monster cake." "lego cake." "beet cake." "pepperoni cake." "fish cake." "rice cake." "pasta cake." "brick cake."

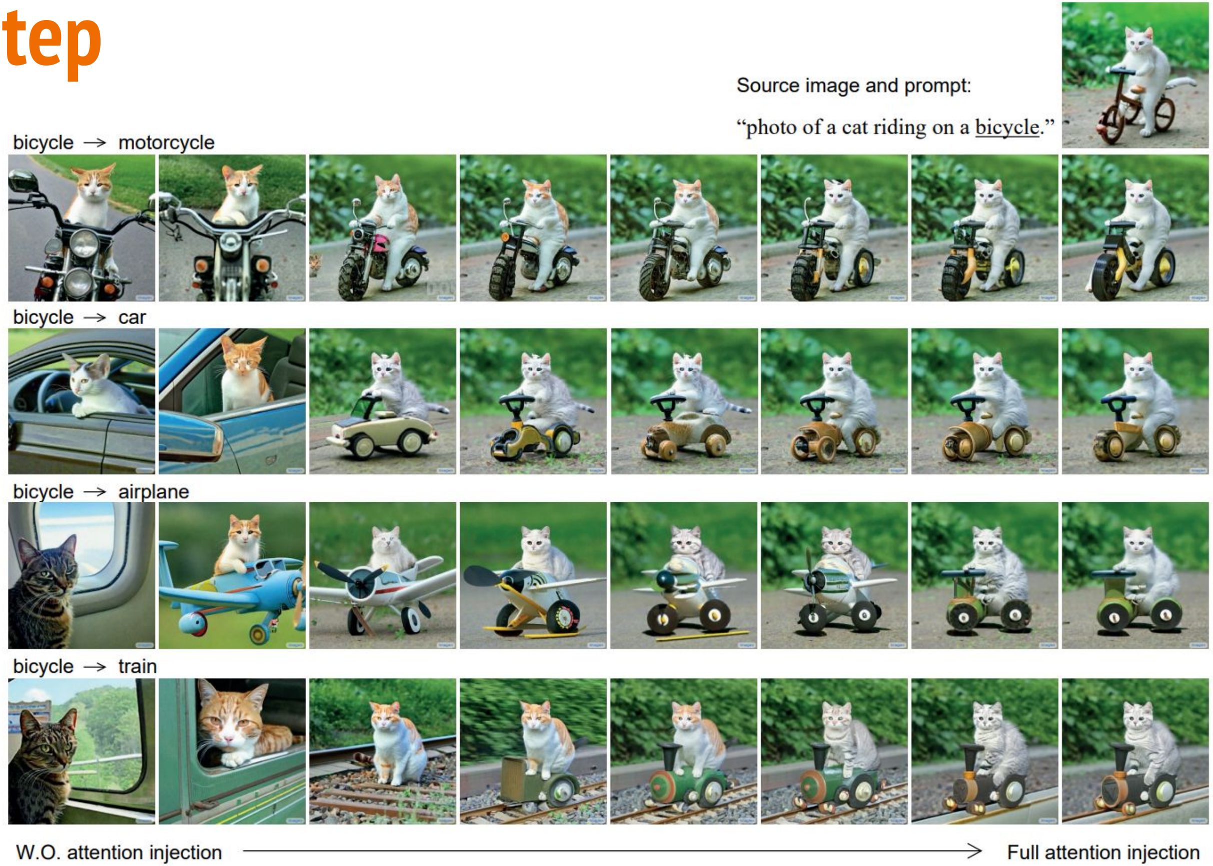
Fixed attention maps and random seed

Fixed random seed

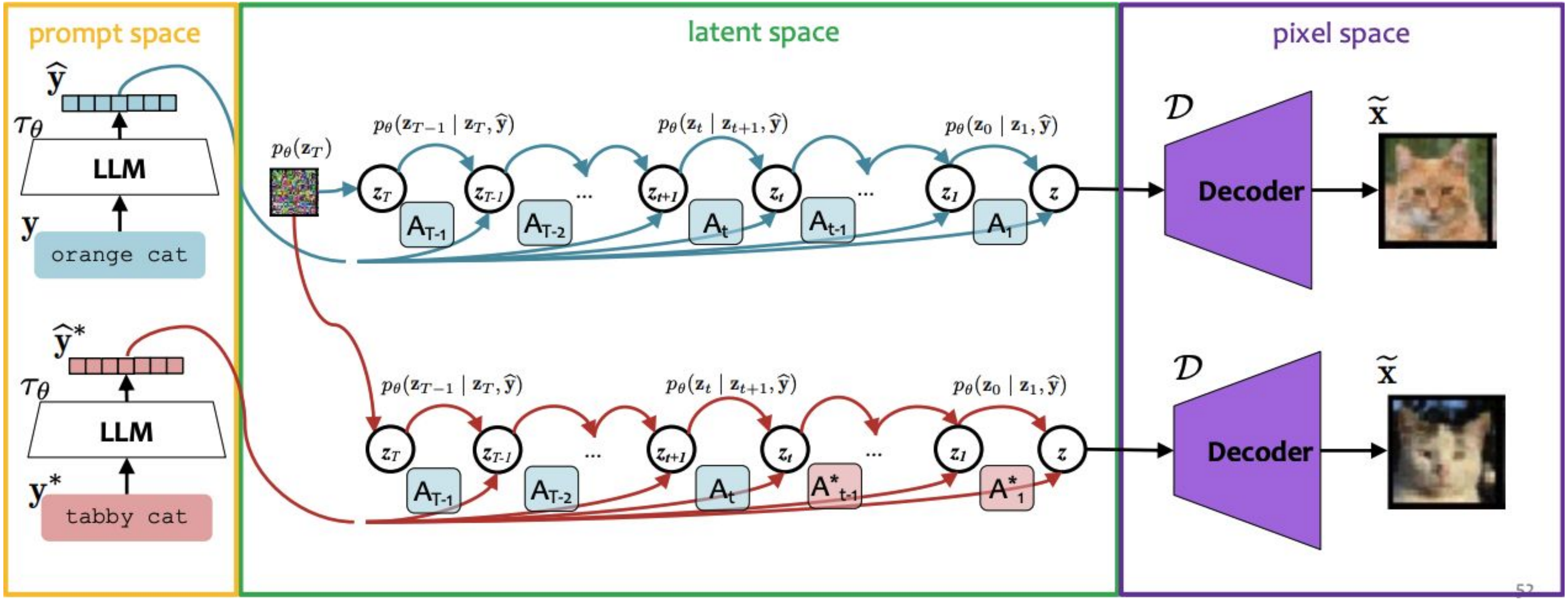
"lemon cake." "cheese cake." "apple cake." "pumpkin cake." "chocolate cake." "pistachio cake." "jello cake." "matcha cake."

"monster cake." "lego cake." "beet cake." "pepperoni cake." "fish cake." "rice cake." "pasta cake." "brick cake."

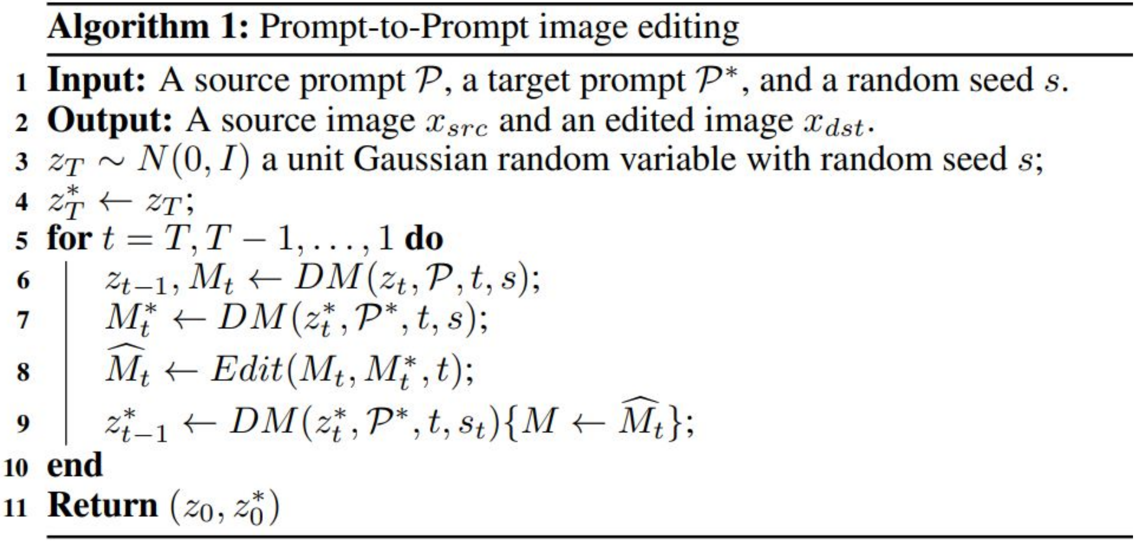
# Injection Step



# Method


prompt space
latent space
pixel space
$\hat{\mathbf{y}}$
$\tau_\theta$
LLM
$\mathbf{y}$
orange cat
$\hat{\mathbf{y}}^*$
$\tau_\theta$
LLM
$\mathbf{y}^*$
tabby cat
$p_\theta(\mathbf{z}_T)$
$p_\theta(\mathbf{z}_{T-1} \mid \mathbf{z}_T, \hat{\mathbf{y}})$
$p_\theta(\mathbf{z}_t \mid \mathbf{z}_{t+1}, \hat{\mathbf{y}})$
$p_\theta(\mathbf{z}_0 \mid \mathbf{z}_1, \hat{\mathbf{y}})$
$z_T$
$z_{T\text{-}1}$
...
$z_{t+1}$
$z_t$
...
$z_1$
$z$
$A_{T\text{-}1}$
$A_{T\text{-}2}$
$A_t$
$A_{t\text{-}1}$
$A_1$
$p_\theta(\mathbf{z}_{T-1} \mid \mathbf{z}_T, \hat{\mathbf{y}})$
$p_\theta(\mathbf{z}_t \mid \mathbf{z}_{t+1}, \hat{\mathbf{y}})$
$p_\theta(\mathbf{z}_0 \mid \mathbf{z}_1, \hat{\mathbf{y}})$
$z_T$
$z_{T\text{-}1}$
...
$z_{t+1}$
$z_t$
...
$z_1$
$z$
$A_{T\text{-}1}$
$A_{T\text{-}2}$
$A_t$
$A^*_{t\text{-}1}$
$A^*_1$
$\mathcal{D}$
Decoder
$\tilde{\mathbf{x}}$
$\mathcal{D}$
Decoder
$\tilde{\mathbf{x}}$

# Pseudo Code



**Algorithm 1:** Prompt-to-Prompt image editing

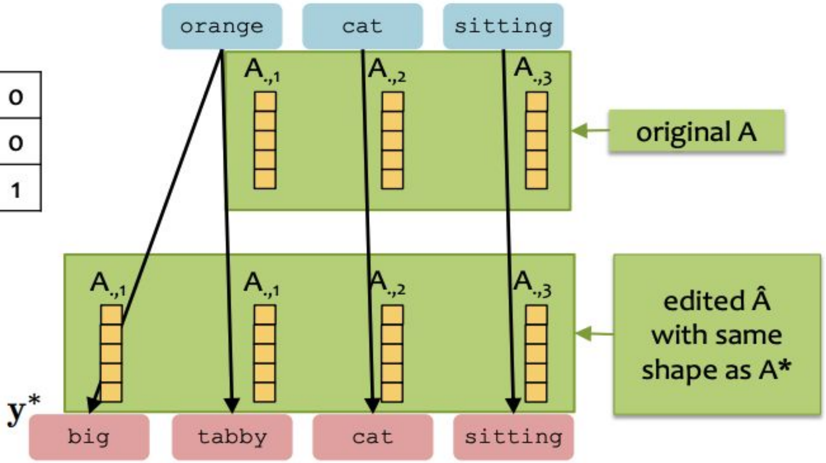
```
Input: A source prompt $\mathcal{P}$, a target prompt $\mathcal{P}^*$, and a random seed $s$.
Output: A source image $x_{src}$ and an edited image $x_{dst}$.
$z_T \sim N(0, I)$ a unit Gaussian random variable with random seed $s$;
$z_T^* \leftarrow z_T$;
for $t = T, T-1, \ldots, 1$ do
    $z_{t-1}, M_t \leftarrow DM(z_t, \mathcal{P}, t, s)$;
    $M_t^* \leftarrow DM(z_t^*, \mathcal{P}^*, t, s)$;
    $\widehat{M}_t \leftarrow Edit(M_t, M_t^*, t)$;
    $z_{t-1}^* \leftarrow DM(z_t^*, \mathcal{P}^*, t, s_t)\{M \leftarrow \widehat{M}_t\}$;
end
Return $(z_0, z_0^*)$
```

# Attention Mapping

Swapping attention scores for the tokens of one word to another can be intuitively represented with a matrix

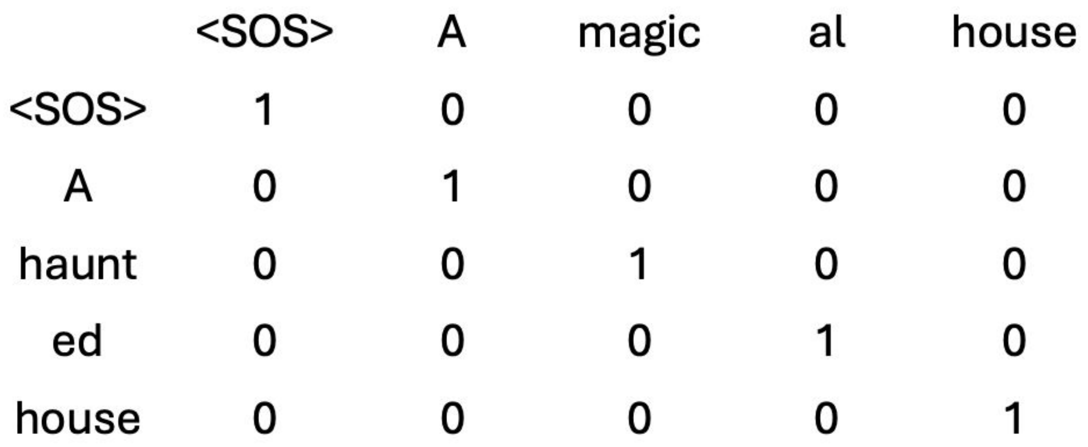
- CLIP tokenizes sentences to a length of 77 with padding, so each matrix will by 77 by 77
- Suppose original word has n token and the replacement word has m
  - If n=m, the mapper matrix is simply the identity matrix
  - Otherwise, replacement occupies an area n*m of the matrix with values of 1/max(n, m)

$$\overset{\hat{A}}{\begin{bmatrix} 1/2 & 1/2 & 6 & 11 \\ 2/2 & 2/2 & 7 & 12 \\ 3/2 & 3/2 & 8 & 13 \\ 4/2 & 4/2 & 9 & 14 \\ 5/2 & 5/2 & 10 & 15 \end{bmatrix}} = \overset{A}{\begin{bmatrix} 1 & 6 & 11 \\ 2 & 7 & 12 \\ 3 & 8 & 13 \\ 4 & 9 & 14 \\ 5 & 10 & 15 \end{bmatrix}} \overset{M}{\begin{bmatrix} .5 & .5 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}$$



# Attention Mapping: 1 to 1 or N to N

Original: A haunt-ed house Modified: A magic-al house

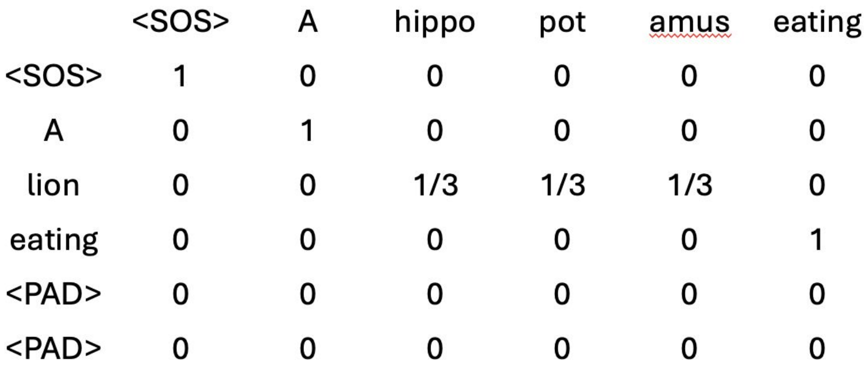


| | <SOS> | A | magic | al | house |
|---|---|---|---|---|---|
| <SOS> | 1 | 0 | 0 | 0 | 0 |
| A | 0 | 1 | 0 | 0 | 0 |
| haunt | 0 | 0 | 1 | 0 | 0 |
| ed | 0 | 0 | 0 | 1 | 0 |
| house | 0 | 0 | 0 | 0 | 1 |

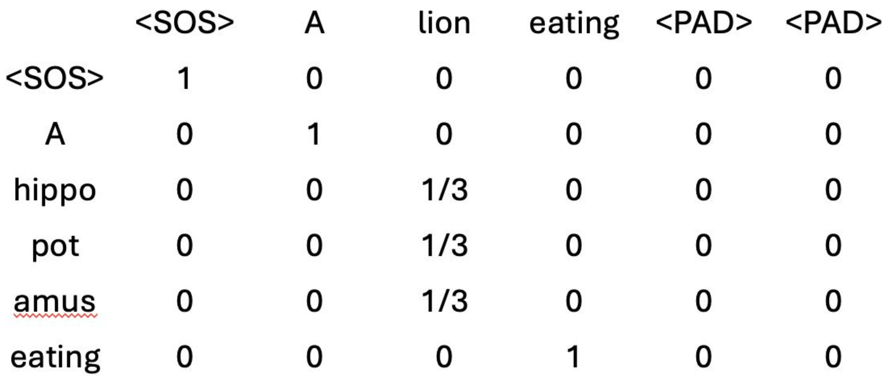
# Attention Mapping: 1 to N, N to 1

Original: A lion eating
Modified: A hippo-pot-amus eating

Original: A hippo-pot-amus eating
Modified: A lion eating



| | <SOS> | A | hippo | pot | amus | eating |
|---|---|---|---|---|---|---|
| <SOS> | 1 | 0 | 0 | 0 | 0 | 0 |
| A | 0 | 1 | 0 | 0 | 0 | 0 |
| lion | 0 | 0 | 1/3 | 1/3 | 1/3 | 0 |
| eating | 0 | 0 | 0 | 0 | 0 | 1 |
| <PAD> | 0 | 0 | 0 | 0 | 0 | 0 |
| <PAD> | 0 | 0 | 0 | 0 | 0 | 0 |

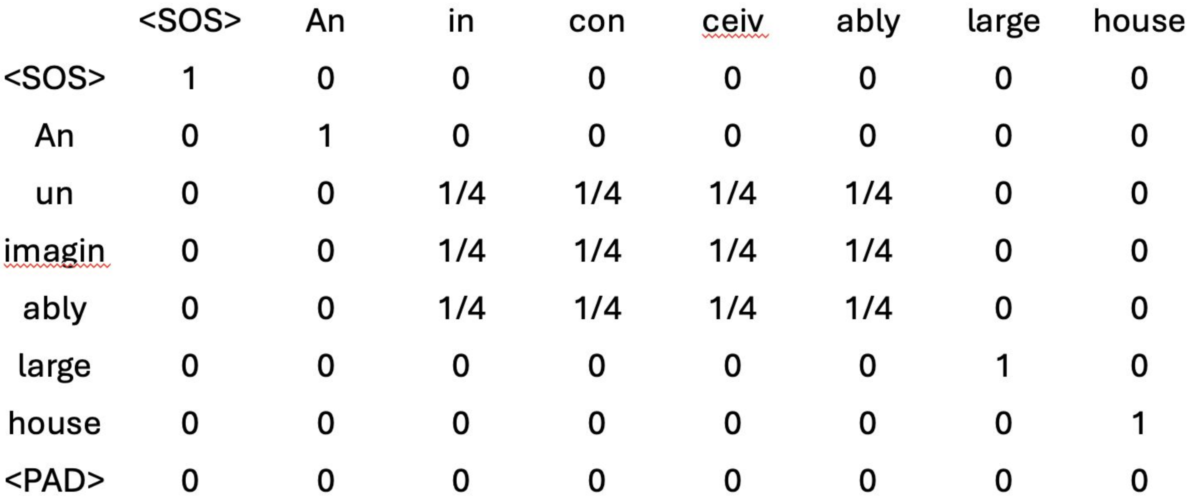


| | <SOS> | A | lion | eating | <PAD> | <PAD> |
|---|---|---|---|---|---|---|
| <SOS> | 1 | 0 | 0 | 0 | 0 | 0 |
| A | 0 | 1 | 0 | 0 | 0 | 0 |
| hippo | 0 | 0 | 1/3 | 0 | 0 | 0 |
| pot | 0 | 0 | 1/3 | 0 | 0 | 0 |
| amus | 0 | 0 | 1/3 | 0 | 0 | 0 |
| eating | 0 | 0 | 0 | 1 | 0 | 0 |

# Attention Mapping: N to M

Original: An un-imagin-ably large house

Modified: An in-con-ceiv-ably large house



| | <SOS> | An | in | con | ceiv | ably | large | house |
|---|---|---|---|---|---|---|---|---|
| <SOS> | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| An | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| un | 0 | 0 | 1/4 | 1/4 | 1/4 | 1/4 | 0 | 0 |
| imagin | 0 | 0 | 1/4 | 1/4 | 1/4 | 1/4 | 0 | 0 |
| ably | 0 | 0 | 1/4 | 1/4 | 1/4 | 1/4 | 0 | 0 |
| large | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| house | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| <PAD> | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

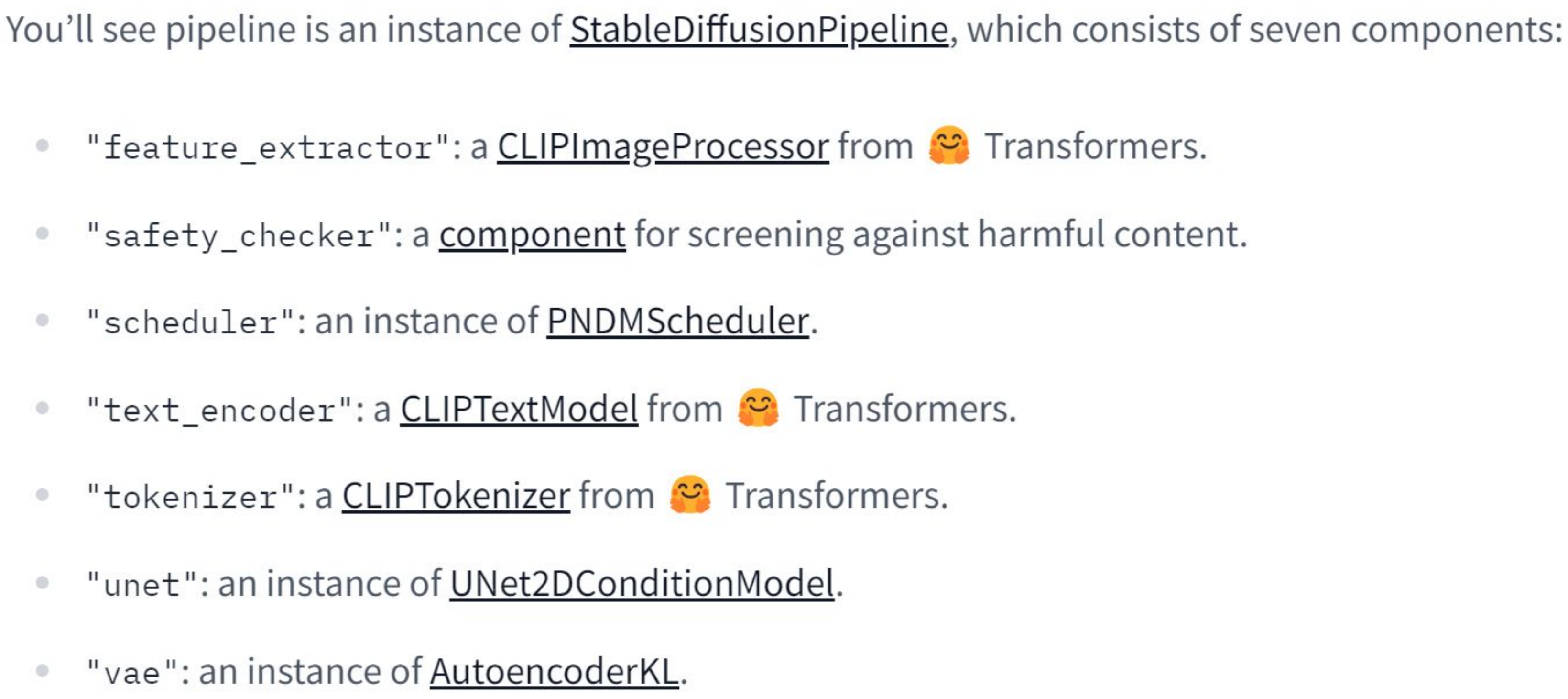
# Diffuser API

Pipeline: Load models according to model id, which is a string

Model class includes attributes like:

- Tokenizer: output is a dictionary
- Text Encoder: output is an object
- VAE
- Scheduler

# Diffuser API

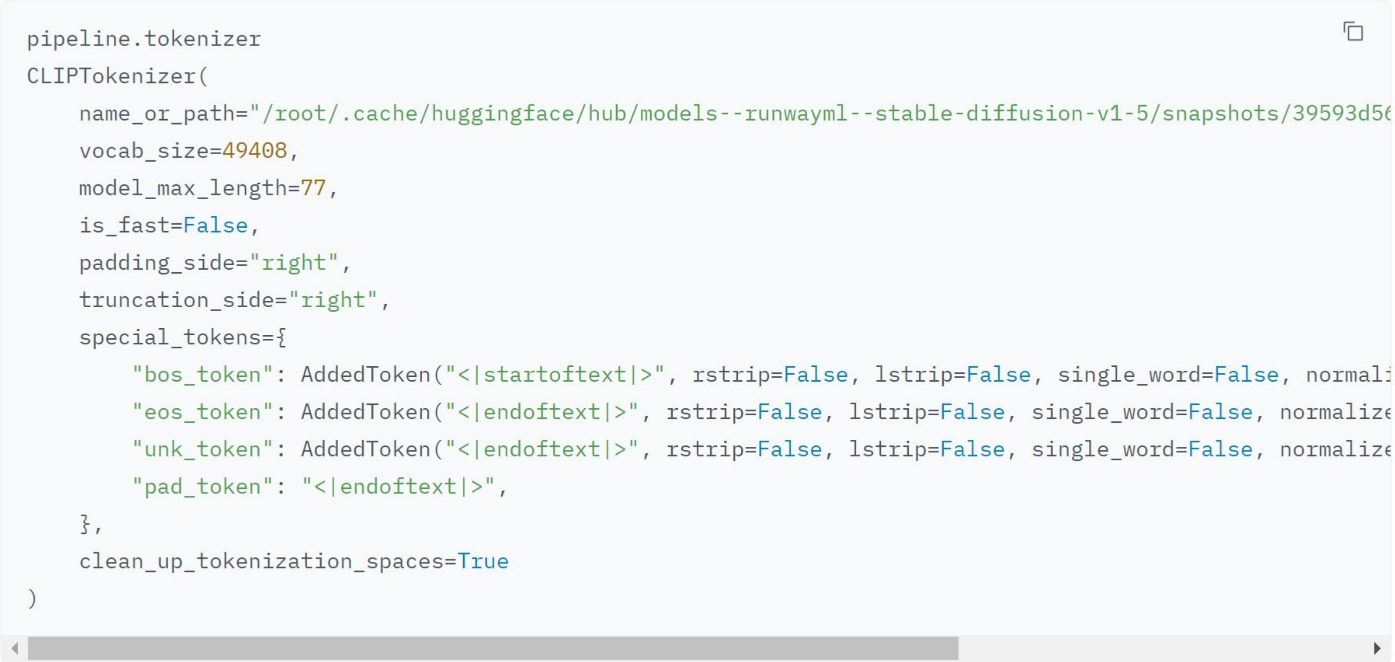


You’ll see pipeline is an instance of StableDiffusionPipeline, which consists of seven components:

- "feature_extractor": a CLIPImageProcessor from 🤗 Transformers.
- "safety_checker": a component for screening against harmful content.
- "scheduler": an instance of PNDMScheduler.
- "text_encoder": a CLIPTextModel from 🤗 Transformers.
- "tokenizer": a CLIPTokenizer from 🤗 Transformers.
- "unet": an instance of UNet2DConditionModel.
- "vae": an instance of AutoencoderKL.

# Diffuser API

You can access each of the components of the pipeline as an attribute to view its configuration:



```
pipeline.tokenizer
CLIPTokenizer(
    name_or_path="/root/.cache/huggingface/hub/models--runwayml--stable-diffusion-v1-5/snapshots/39593d56
    vocab_size=49408,
    model_max_length=77,
    is_fast=False,
    padding_side="right",
    truncation_side="right",
    special_tokens={
        "bos_token": AddedToken("<|startoftext|>", rstrip=False, lstrip=False, single_word=False, normali
        "eos_token": AddedToken("<|endoftext|>", rstrip=False, lstrip=False, single_word=False, normalize
        "unk_token": AddedToken("<|endoftext|>", rstrip=False, lstrip=False, single_word=False, normalize
        "pad_token": "<|endoftext|>",
    },
    clean_up_tokenization_spaces=True
)
```

# Code Run-through