Prompt to Prompt

Image Editing with Cross-Attention Control

Yuanzhi Zhu

16/Apr/2023

Content

Prompt-to-Prompt Image Editing with Cross Attention Control

Motivation

Text-2-image editing is sensitive to text prompts

- text influence the high level semantic only

photo of a cat riding a bike

photo of a cat on a bike

A spatial mask to localize the edit

- hard to draw and
- ignoring the original structure & content

Origin Mask Inpainting Example

Prompt: girl with red hair

Toward Mask-free Text-2-Image Editing ☺

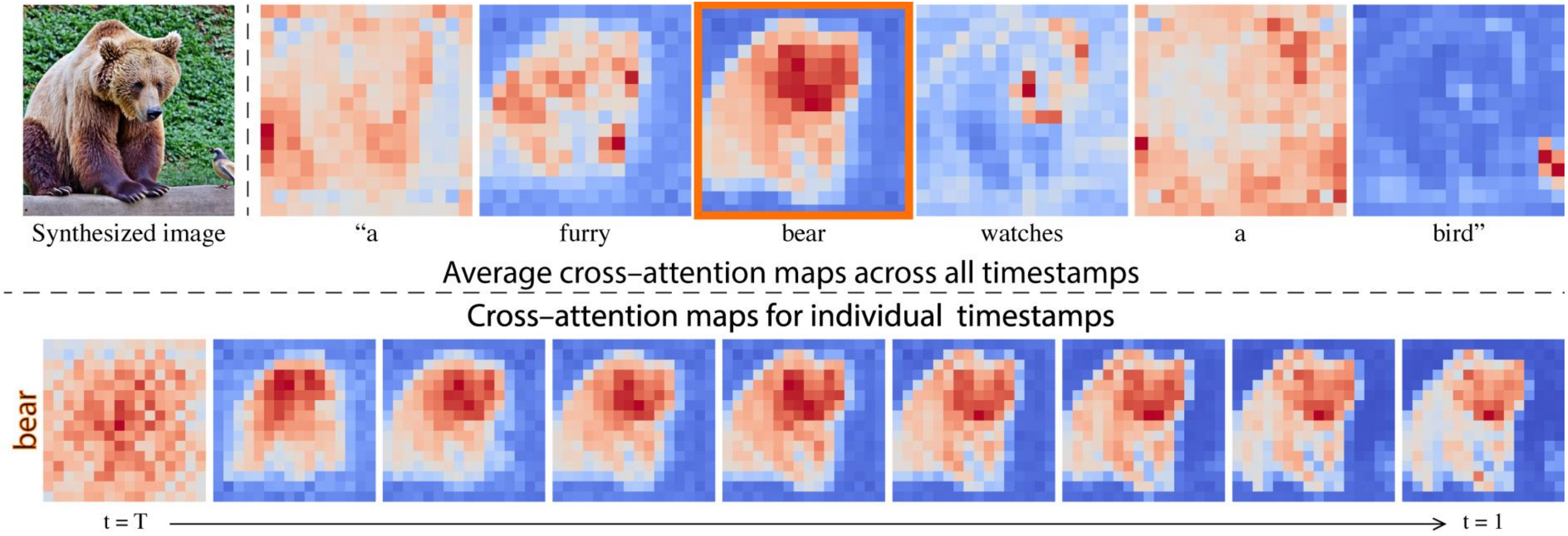
[Prompt-to-Prompt]

[Prompt-to-Prompt Image Editing with Cross-Attention Control | OpenReview]

[Attend-and-Excite/explain.ipynb at main · AttendAndExcite/Attend-and-Excite (github.com)]

Prompt-to-Prompt Image Editing with Cross Attention Control

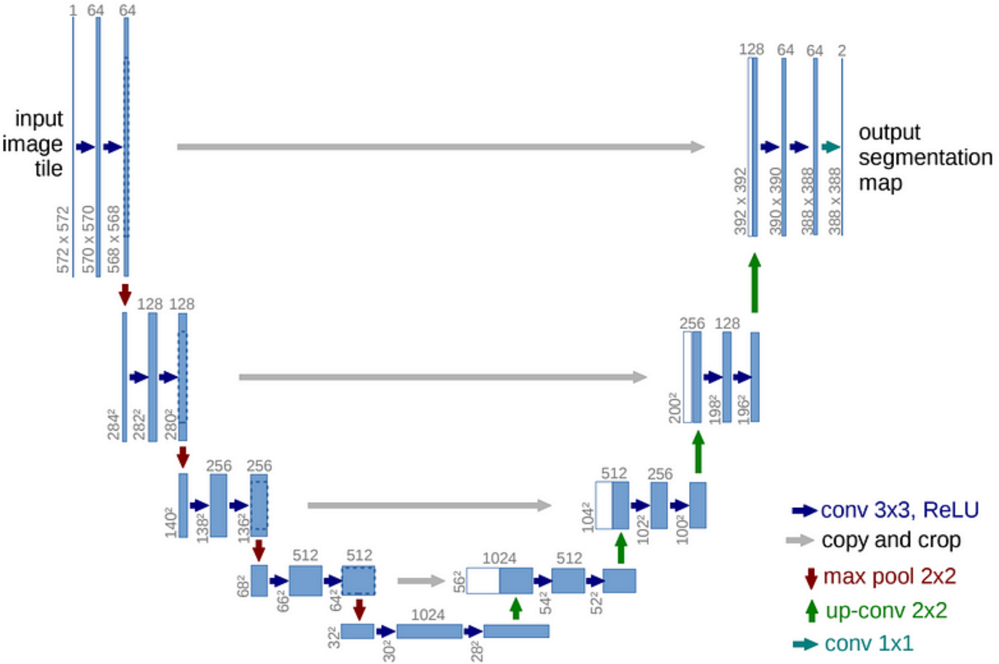
Key Observation: spatial information in the cross-attention maps



Prompt-to-Prompt Image Editing with Cross Attention Control

Model Architecture & Location of Cross Attention

```
unet_config:
    target: ldm.modules.diffusionmodules.openaimodel.UNetModel
```




```
for level, mult in enumerate(channel_mult):
    for _ in range(num_res_blocks):
        layers = [ ResBlock(...)]
        ch = mult * model_channels
        if ds in attention_resolutions:
            layers.append(SpatialTransformer(...))
```

```
assert use_spatial_transformer, 'Fool!! \
You forgot to use the spatial transformer \
for your cross-attention conditioning...'
```

https://github.com/CompVis/stable-diffusion/blob/main/ldm/modules/diffusionmodules/openaimodel.py#L541

Prompt-to-Prompt Image Editing with Cross Attention Control

Attentions in stable diffusion

```
from ldm.modules.attention import SpatialTransformer
```

```
SpatialTransformer → [BasicTransformerBlock(…) for d in range(depth)]
```

```
def _forward(self, x, context=None):
    x = self.attn1(self.norm1(x), context=context if self.disable_self_attn else None) + x
    x = self.attn2(self.norm2(x), context=context) + x
    x = self.ff(self.norm3(x)) + x
    return x
```

→self
→cross

```
# x.shape = b (h w) c
# context.shape = b n_token dim
```

https://github.com/CompVis/stable-diffusion/blob/main/ldm/modules/attention.py#L211

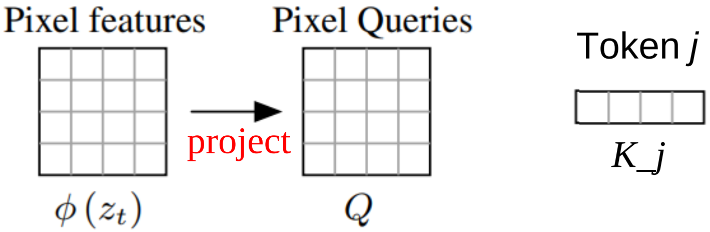
Prompt-to-Prompt Image Editing with Cross Attention Control

$$M = \text{Softmax}\left(\frac{QK^T}{\sqrt{d}}\right)$$

```
# x.shape = b (h w) c --> b h*w inner_dim
# context.shape = b n_token dim --> b n_token inner_dim
```

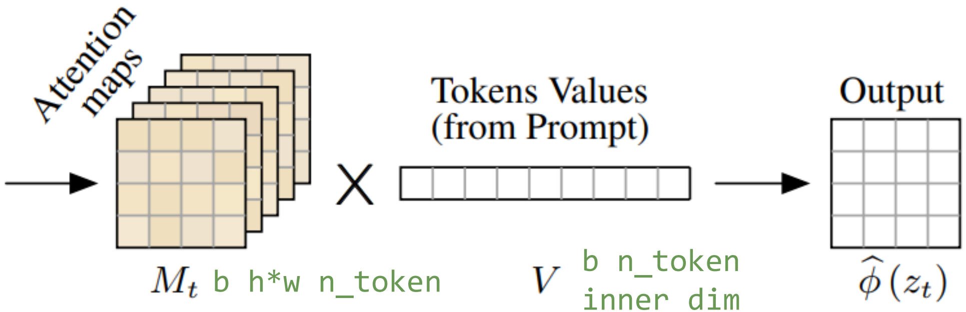
cross-attention

```
# attention, what we cannot get enough of
sim = einsum('b i d, b j d -> b i j', q, k) * self.scale
attn = sim.softmax(dim=-1)     # normalizes values along axis -1 (j)
out = einsum('b i j, b j d -> b i d', attn, v)
```



$\underbrace{\texttt{\#(x*context[j].T)}}_{(M_t)_j}\texttt{.shape = b h*w}$

reshaped and visualized as images



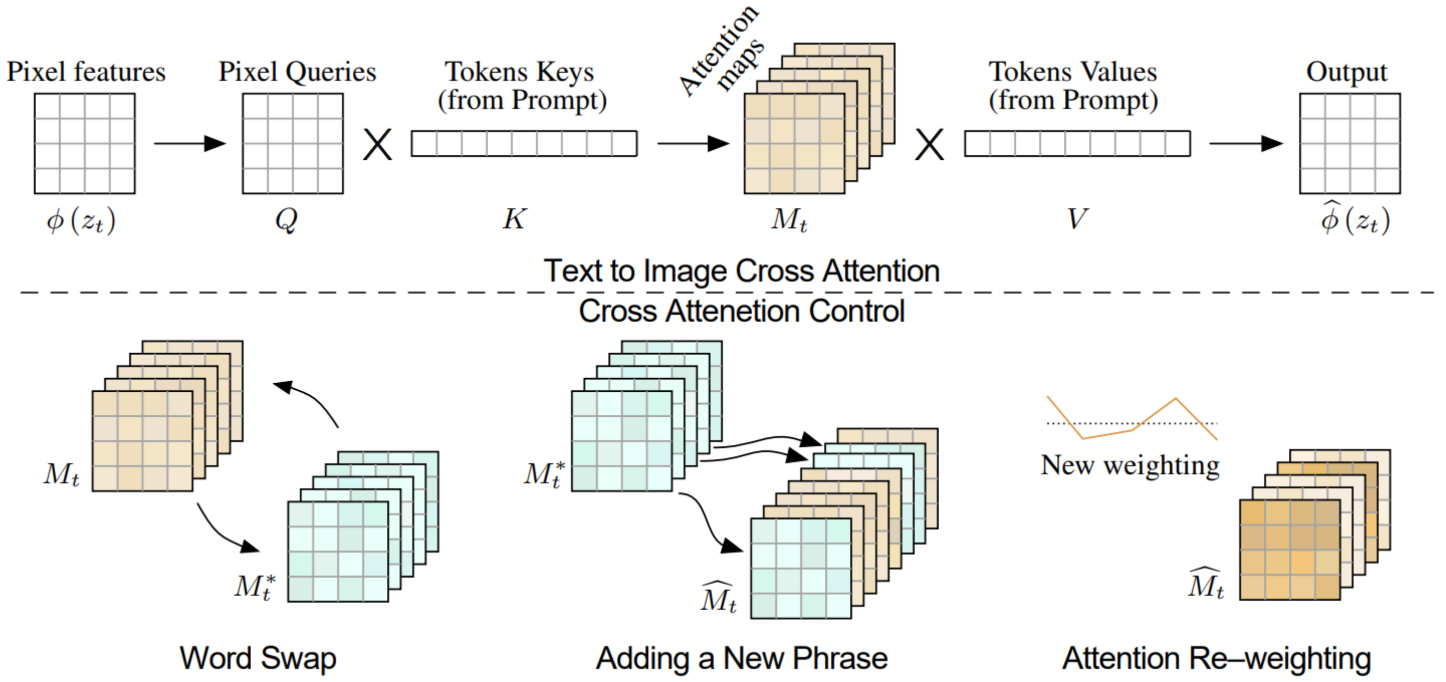
#outer product --> b h*w inner_dim

Content

Prompt-to-Prompt Image Editing with Cross Attention Control

$$M = \text{Softmax}\left(\frac{QK^T}{\sqrt{d}}\right)$$

we can inject the attention maps M that were obtained from the generation with the original prompt P, into a second generation with the modified prompt $P*$



Prompt-to-Prompt Image Editing with Cross Attention Control

Algorithm 1: Prompt-to-Prompt image editing

Input: A source prompt $\mathcal{P}$, a target prompt $\mathcal{P}^*$, and a random seed s.
Optional for local editing: w and w^*, words in $\mathcal{P}$ and $\mathcal{P}^*$, specifying the editing region.
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**
  $z_{t-1}, M_t \leftarrow DM(z_t, \mathcal{P}, t, s)$;
  $M_t^* \leftarrow DM(z_t^*, \mathcal{P}^*, t, s)$;
  $\widehat{M}_t \leftarrow Edit(M_t, M_t^*, t)$;
  $z_{t-1}^* \leftarrow DM(z_t^*, \mathcal{P}^*, t, s)\{M \leftarrow \widehat{M}_t\}$;
  if *local* **then**
    $\alpha \leftarrow B(\overline{M}_{t,w}) \cup B(\overline{M}^*_{t,w^*})$;
    $z_{t-1}^* \leftarrow (1-\alpha) \odot z_{t-1} + \alpha \odot z_{t-1}^*$;
  end
end
Return (z_0, z_0^*)

Word Swap

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

Adding a New Phrase

$$(Edit\,(M_t, M_t^*, t))_{i,j} := \begin{cases} (M_t^*)_{i,j} & \text{if } A(j) = None \\ (M_t)_{i,A(j)} & \text{otherwise.} \end{cases}$$

i :pixel value; j :text token

Attention Re–weighting

$$(Edit\,(M_t, M_t^*, t))_{i,j} := \begin{cases} c \cdot (M_t)_{i,j} & \text{if } j = j^* \\ (M_t)_{i,j} & \text{otherwise} \end{cases}$$

Prompt-to-Prompt Image Editing with Cross Attention Control

How to implement?

https://github.com/google/prompt-to-prompt/blob/main/ptp_utils.py#L173

```
### prompt to prompt setup
controller = _setup_attention_controller(attention_control_type, prompts,
                 cross_replace_steps=cross_replace_steps, self_replace_steps=self_replace_steps,
                 LocalBlend_pair=LocalBlend_pair,
                 Reweightwords=Reweightwords, Reweightscales=Reweightscales,
                 sampling_steps=steps, tokenizer=model.cond_stage_model.tokenizer)
ptp_utils.register_attention_control(model, controller)


def register_recr(net_, count, place_in_unet):
    ## modify the forward function of the cross attention module
    if net_.__class__.__name__ == 'CrossAttention’:
        net_.forward = ca_forward(net_, place_in_unet)
        return count + 1
```

Prompt-to-Prompt Image Editing with Cross Attention Control

ca_forward:

```
sim = einsum('2 i d, 2 j d -> 2 i j', q, k) * self.scale
attn = sim.softmax(dim=-1)    # normalizes values along axis -1 (j)
attn = controller(attn, is_cross, place_in_unet)
out = einsum('2 i j, 2 j d -> 2 i d', attn, v)
```

https://github.com/google/prompt-to-prompt/blob/main/ptp_utils.py#L203

```
### forward method of controller
def forward(self, attn, is_cross: bool, place_in_unet: str):
    super(AttentionControlEdit, self).forward(attn, is_cross, place_in_unet)
    if is_cross or (self.num_self_replace[0] <= self.cur_step < self.num_self_replace[1]):
        h = attn.shape[0] // (self.batch_size)
        attn = attn.reshape(self.batch_size, h, *attn.shape[1:])
        attn_base, attn_repalce = attn[0], attn[1:]
        if is_cross:
            alpha_words = self.cross_replace_alpha[self.cur_step]
            attn_repalce_new = self.replace_cross_attention(attn_base, attn_repalce) * alpha_words \
                                + (1 - alpha_words) * attn_repalce
            attn[1:] = attn_repalce_new
        else:
            attn[1:] = self.replace_self_attention(attn_base, attn_repalce)
        attn = attn.reshape(self.batch_size * h, *attn.shape[2:])
    return attn
```

Prompt-to-Prompt Image Editing with Cross Attention Control

How to implement?

https://github.com/google/prompt-to-prompt/blob/main/ptp_utils.py#L74

```
img, pred_x0 = self.p_sample_ddim(…)
### apply local_blend to img
if controller:
    img = controller.step_callback(img)
```

```
class LocalBlend:
    def __call__(self, x_t, attention_store):
        k = 1
        maps = attention_store["down_cross"][2:4] + attention_store["up_cross"][:3]
        maps = [item.reshape(self.alpha_layers.shape[0], -1, 1, 16, 16, MAX_NUM_WORDS) for item in maps]
        maps = torch.cat(maps, dim=1)
        maps = (maps * self.alpha_layers).sum(-1).mean(1)
        mask = nnf.max_pool2d(maps, (k * 2 + 1, k * 2 +1), (1, 1), padding=(k, k))
        mask = nnf.interpolate(mask, size=(x_t.shape[2:]))
        mask = mask / mask.max(2, keepdims=True)[0].max(3, keepdims=True)[0]
        mask = mask.gt(self.threshold)
        mask = (mask[:1] + mask[1:]).float()
        x_t = x_t[:1] + mask * (x_t - x_t[:1]) # x_t[1:] = mask * x_t[1:] + (1-mask) * x_t[:1]
        return x_t
```

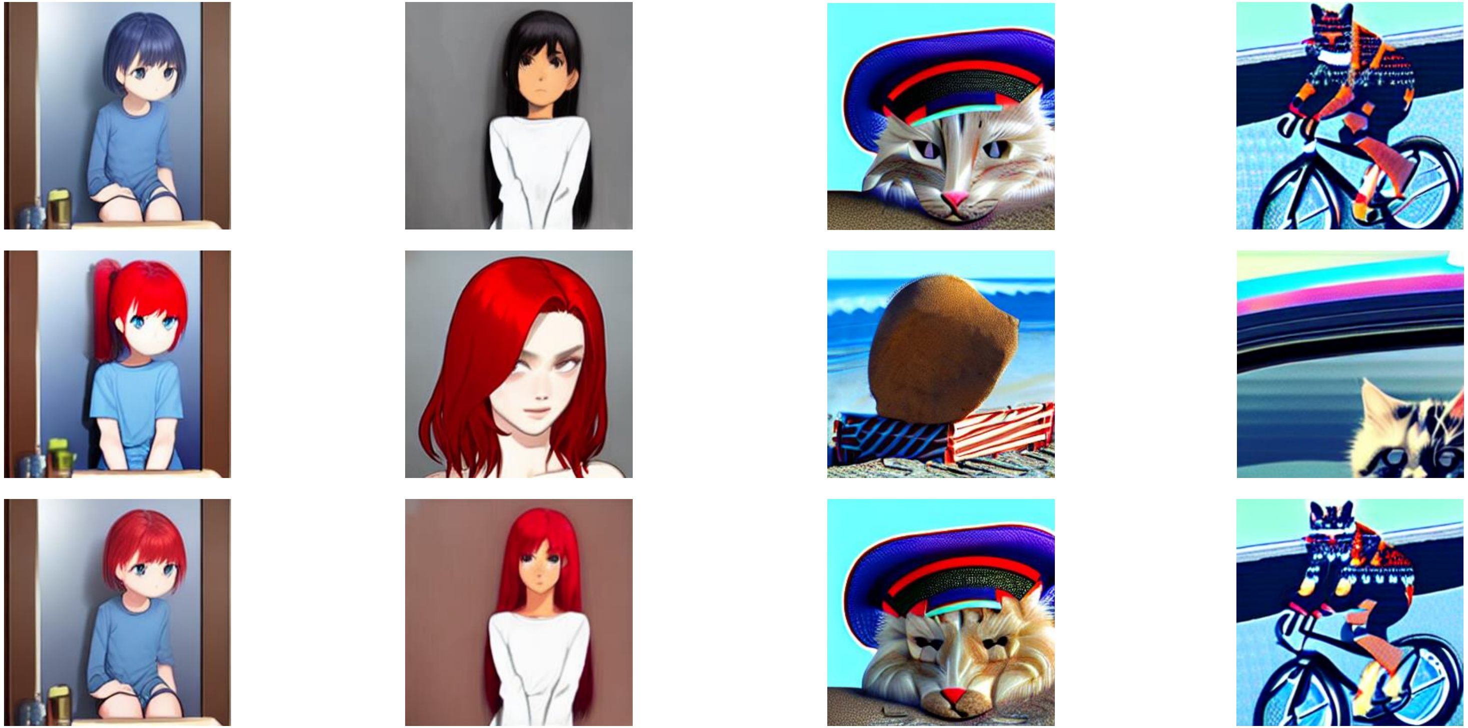
Prompt-to-Prompt Image Editing with Cross Attention Control

a small girl with blue shirt sitting in front of a mirror, red hair

girl, red hair

a cat(lion) with a hat is lying on a beach chair.

photo of a cat riding a bike(car)



original

w/o ptp

w/ ptp

Prompt-to-Prompt Image Editing with Cross Attention Control

photo of a cat on a bike(car)

cross_replace_steps
self_replace_steps

0.5/0.5

0.8/0.2

0.8/0.3

0.8/0.4

0.8/0.5

0.0/0.0

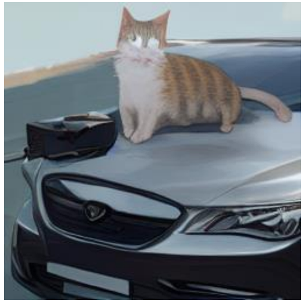
0.8/0.6

0.5/0.4

0.6/0.4

0.7/0.4

0.8/0.4



w/o LocalBlend

Prompt-to-Prompt Image Editing with Cross Attention Control

Conclusions

Pros:

- √ Training free
- √ Mask free
- √ Flexible

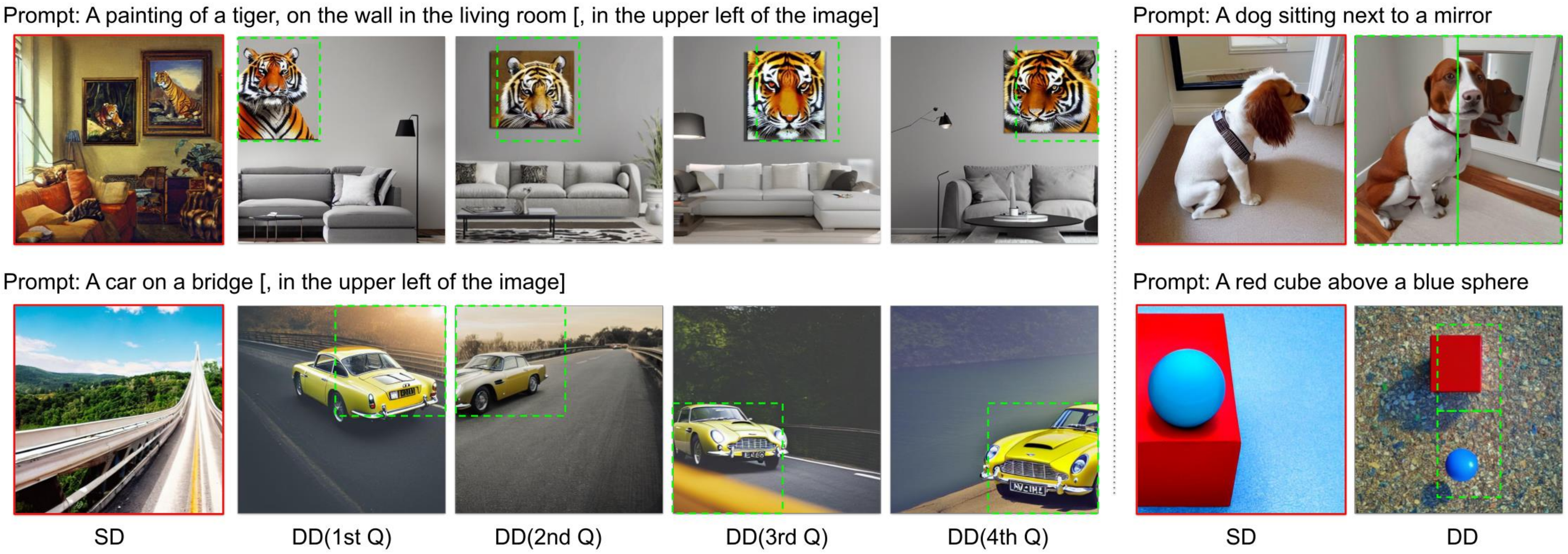
Cons:

- × Precise control
- × Stable hyper-parameter
- × Natural language instruction!

Content

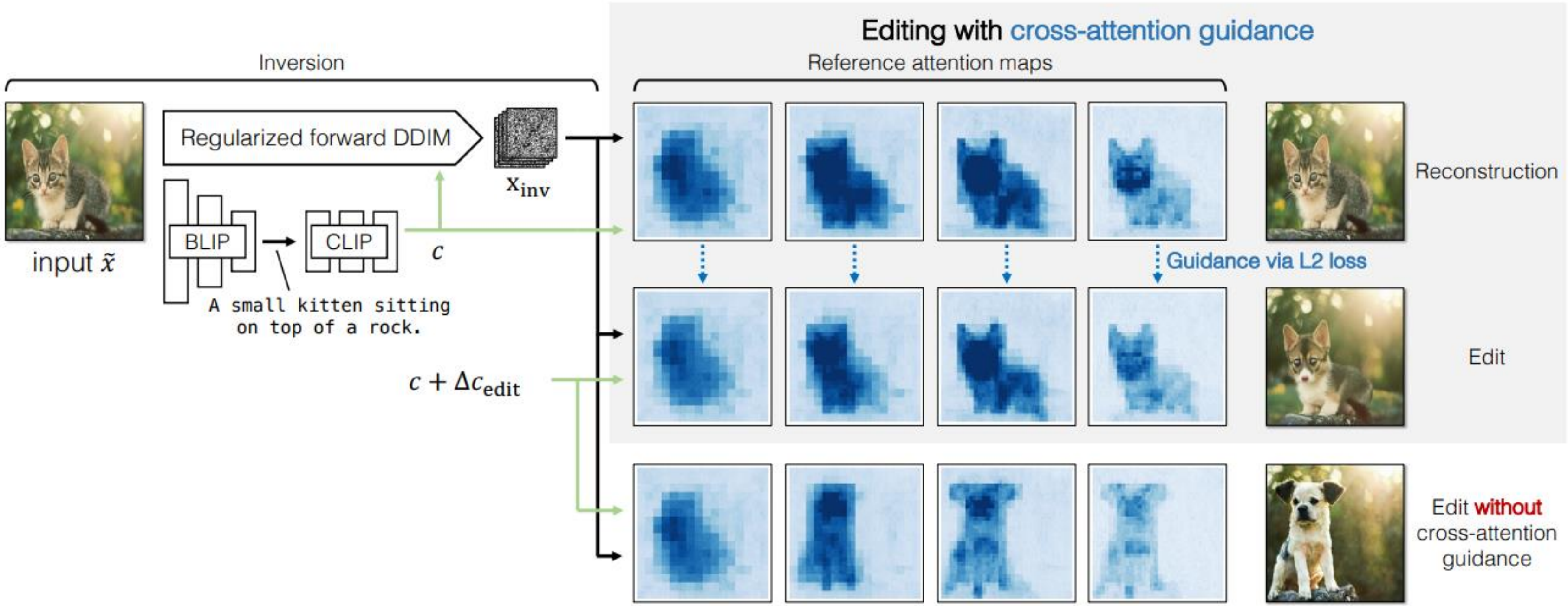
- Motivation & Recap
- Prompt-to-Prompt
- Extensions

Directed Diffusion: Direct Control of Object Placement through Attention Guidance



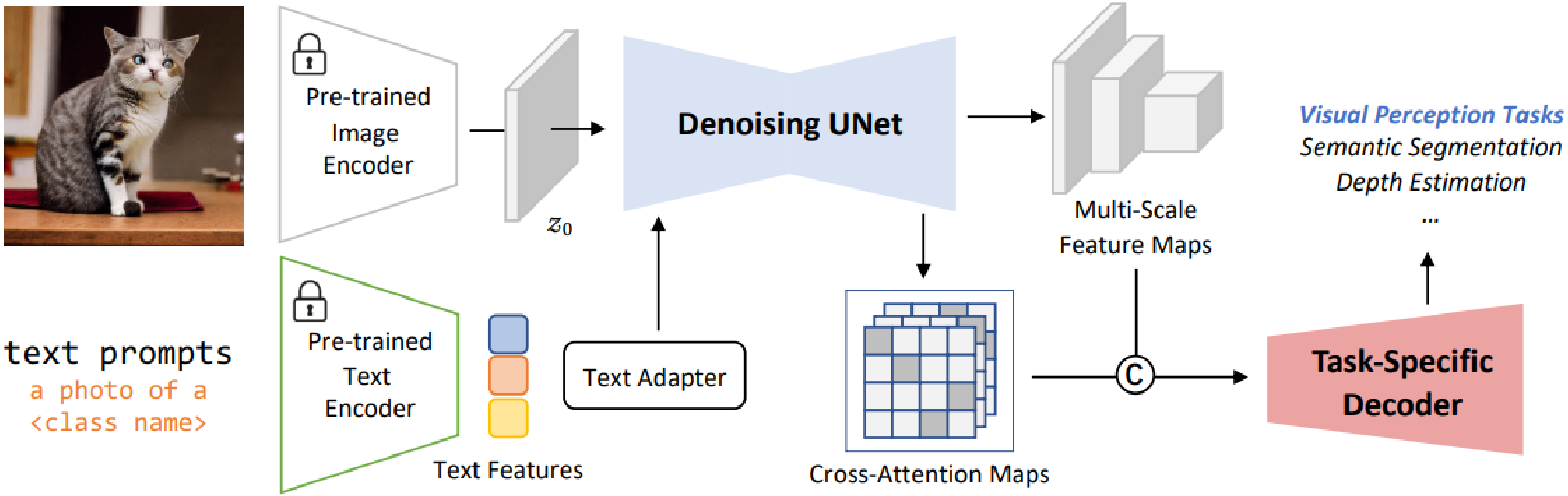
[Directed Diffusion (hohonu-vicml.github.io)](hohonu-vicml.github.io)

Zero-shot Image-to-Image Translation



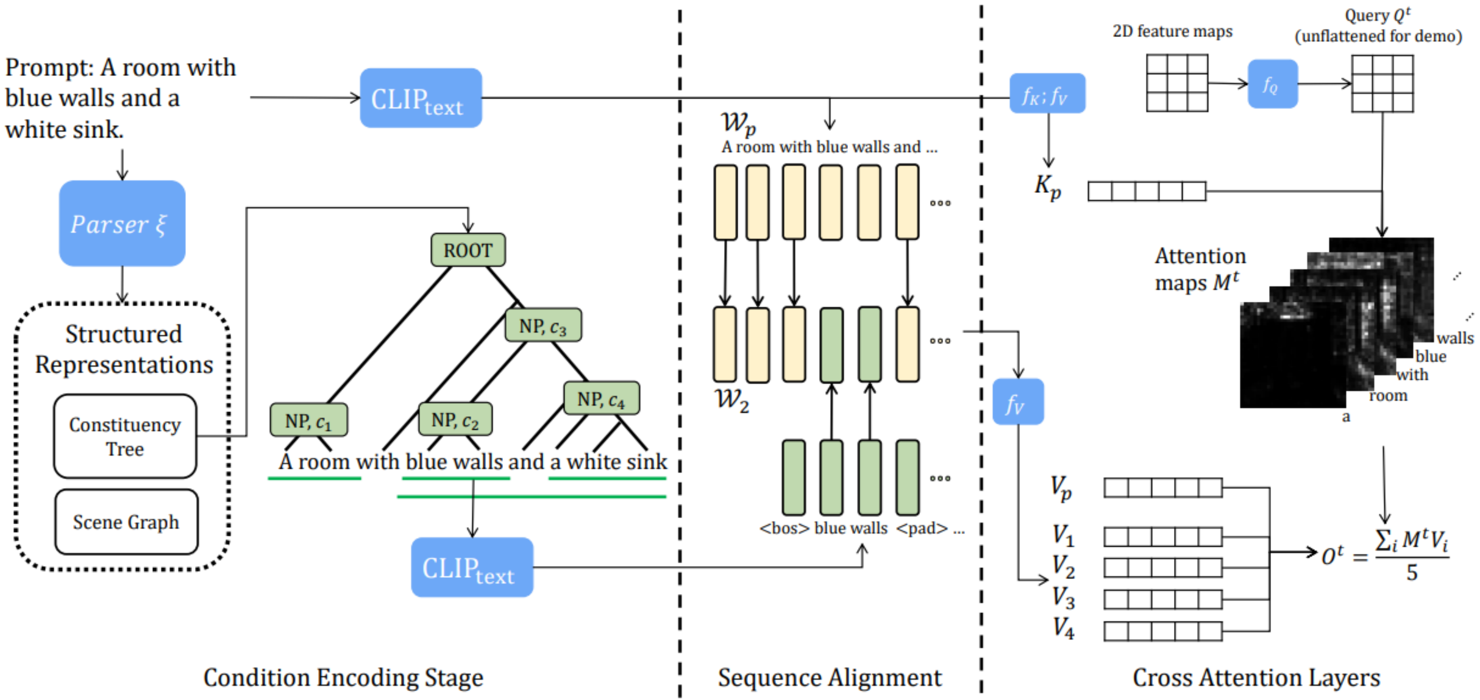
[2302.03027] Zero-shot Image-to-Image Translation (arxiv.org)

Unleashing Text-to-Image Diffusion Models for Visual Perception



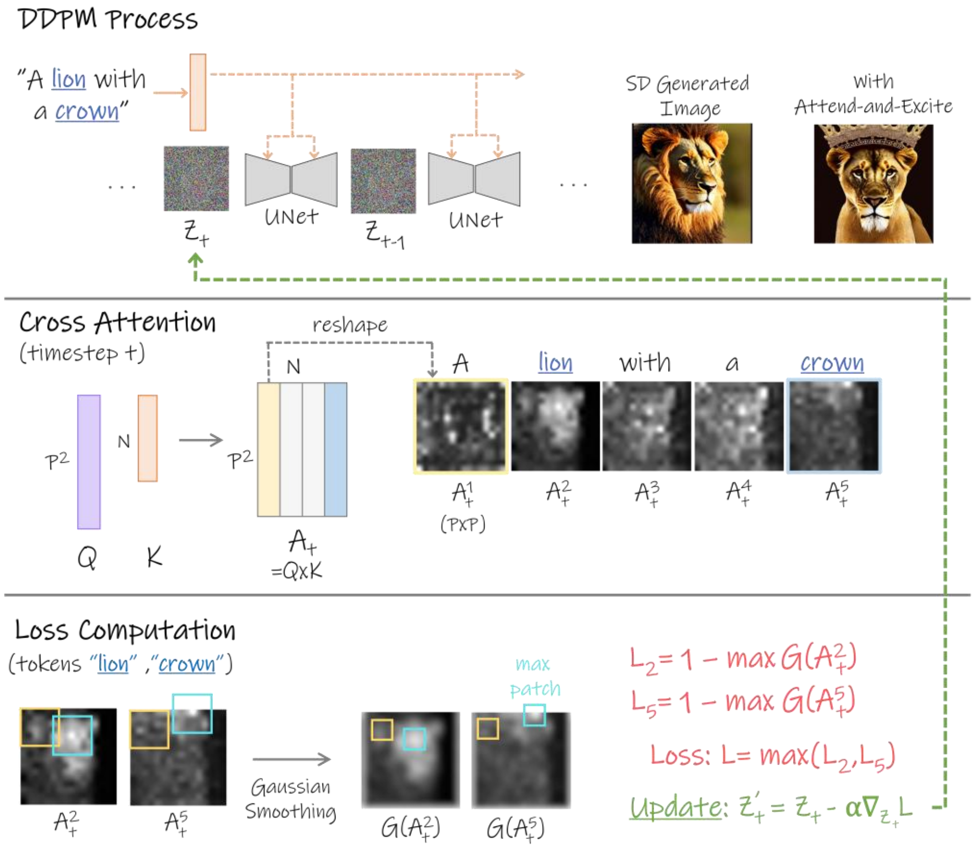
[2303.02153] Unleashing Text-to-Image Diffusion Models for Visual Perception (arxiv.org)

Training-Free Structured Diffusion Guidance for Compositional Text-to-Image Synthesis



[2212.05032] Training-Free Structured Diffusion Guidance for Compositional Text-to-Image Synthesis (arxiv.org)

Attend-and-Excite: Attention-Based Semantic Guidance for Text-to-Image Diffusion Models



[2301.13826] Attend-and-Excite: Attention-Based Semantic Guidance for Text-to-Image Diffusion Models (arxiv.org)