Medical Imaging & Intelligent Reality Lab.
Convergence Medicine/Radiology

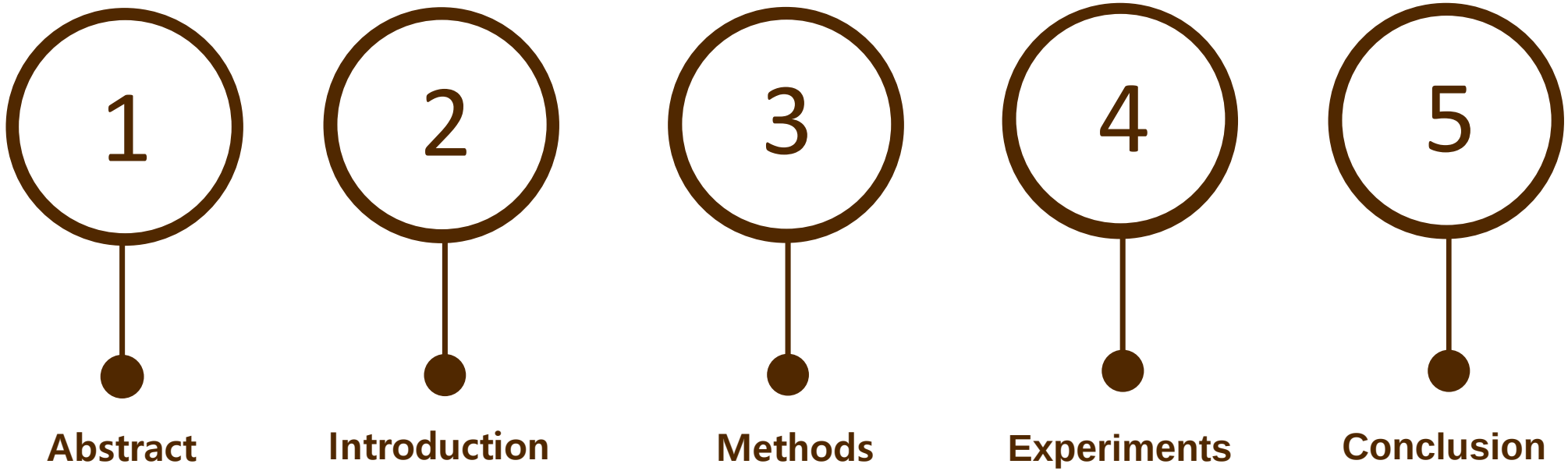
Multi-Concept Customization of Text-to-Image Diffusion

Git: https://github.com/adobe-research/custom-diffusion

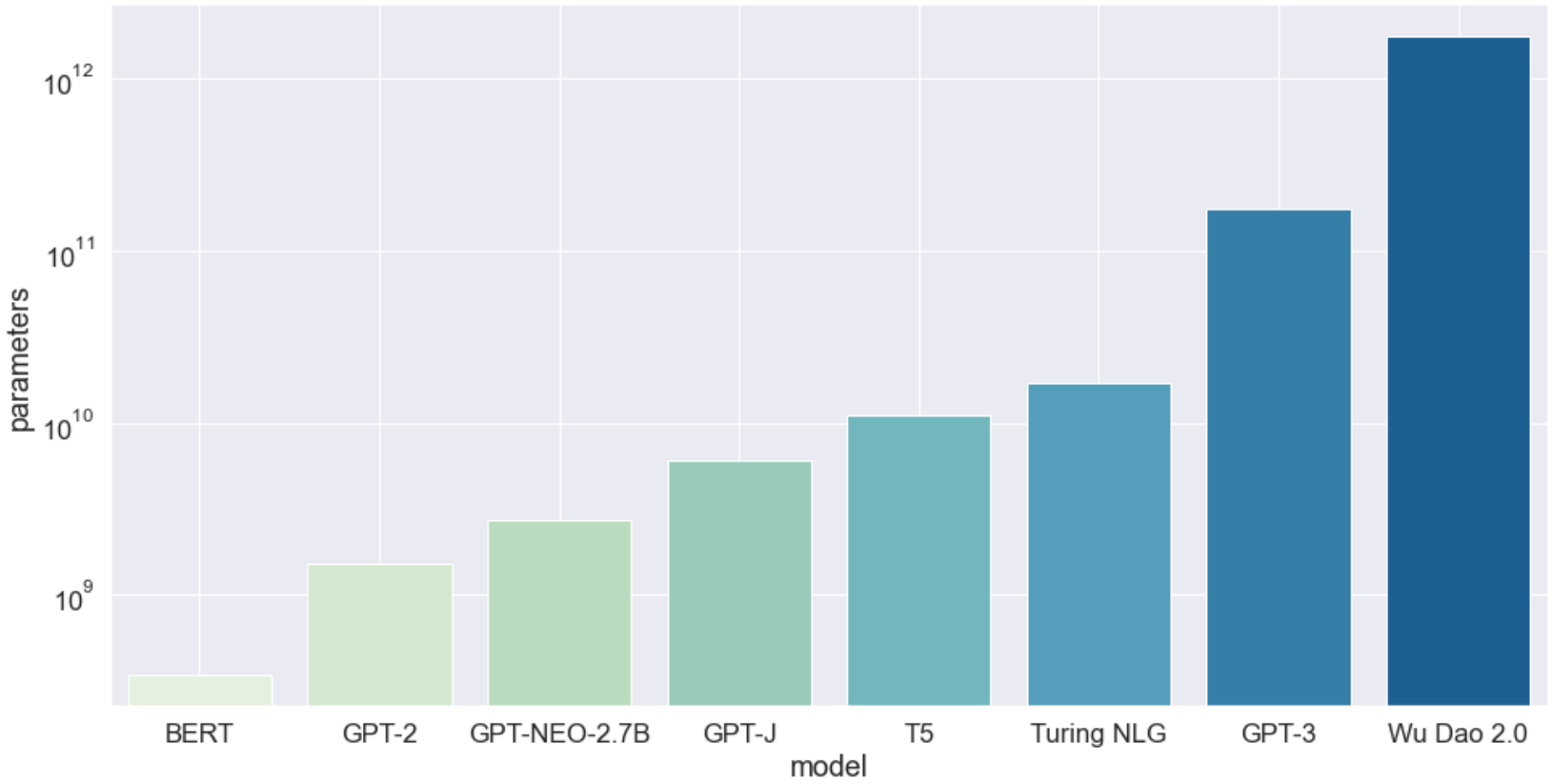
Presenter: Sunggu Kyung

Email: babbu3682@gmail.com

Paper Contents



Background


parameters
10^{12}
10^{11}
10^{10}
10^{9}
BERT
GPT-2
GPT-NEO-2.7B
GPT-J
T5
Turing NLG
GPT-3
Wu Dao 2.0
model

Background

[2021]LoRA

LoRA: Low-Rank Adaptation of Large Language Models

Edward Hu* **Yelong Shen*** **Phillip Wallis** **Zeyuan Allen-Zhu**
Yuanzhi Li **Shean Wang** **Lu Wang** **Weizhu Chen**
Microsoft Corporation
{edwardhu, yeshe, phwallis, zeyuana,
yuanzhil, swang, luw, wzchen}@microsoft.com
yuanzhil@andrew.cmu.edu
(Version 2)

[2021]LoRA

During full fine-tuning, the model is initialized to pre-trained weights Φ_0 and updated to $\Phi_0 + \Delta\Phi$ by repeatedly following the gradient to maximize the conditional language modeling objective:

Fully fine-tuning:

$$\max_{\Phi} \sum_{(x,y)\in\mathcal{Z}} \sum_{t=1}^{|y|} \log\left(P_{\Phi}(y_t|x, y_{<t})\right) \tag{1}$$

One of the main drawbacks for full fine-tuning is that for *each* downstream task, we learn a *different* set of parameters $\Delta\Phi$ whose dimension $|\Delta\Phi|$ equals $|\Phi_0|$. Thus, if the pre-trained model is large (such as GPT-3 with $|\Phi_0| \approx 175$ Billion), storing and deploying many independent instances of fine-tuned models can be challenging, if at all feasible.

In this paper, we adopt a more parameter-efficient approach, where the task-specific parameter increment $\Delta\Phi = \Delta\Phi(\Theta)$ is further encoded by a much smaller-sized set of parameters Θ with $|\Theta| \ll |\Phi_0|$. The task of finding $\Delta\Phi$ thus becomes optimizing over Θ:

Parameter-efficient approach:

$$\max_{\Theta} \sum_{(x,y)\in\mathcal{Z}} \sum_{t=1}^{|y|} \log\left(p_{\Phi_0+\Delta\Phi(\Theta)}(y_t|x, y_{<t})\right) \tag{2}$$

In the subsequent sections, we propose to use a low-rank representation to encode $\Delta\Phi$ that is both compute- and memory-efficient. When the pre-trained model is GPT-3 175B, the number of trainable parameters $|\Theta|$ can be as small as 0.01% of $|\Phi_0|$.

[2021]LoRA

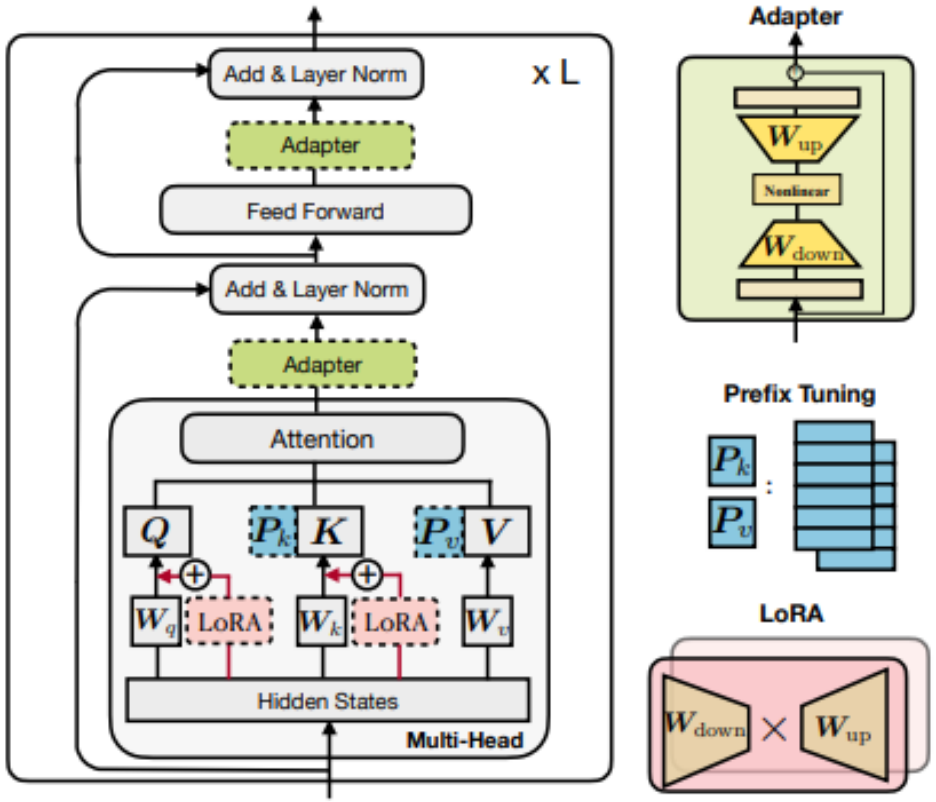
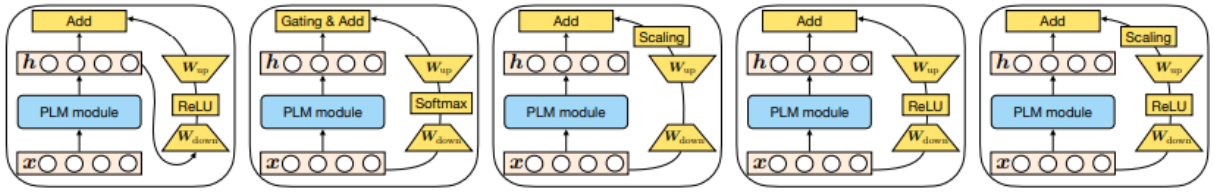


Figure 1: Illustration of the transformer architecture and several state-of-the-art parameter-efficient tuning methods. We use blocks with dashed borderlines to represent the added modules by those methods.



(a) Adapter (b) Prefix Tuning (c) LoRA (d) Parallel Adapter (e) Scaled PA

Figure 3: Graphical illustration of existing methods and the proposed variants. "PLM module" represents a certain sublayer of the PLM (e.g. attention or FFN) that is frozen. "Scaled PA" denotes scaled parallel adapter. We do not include multi-head parallel adapter here to save space.

Table 1: Parameter-efficient tuning methods decomposed along the defined design dimensions. Here, for clarity, we directly write the adapter nonlinear function as ReLU which is commonly used. The bottom part of the table exemplifies new variants by transferring design choices of existing approaches.

Method	Δh functional form	insertion form	modified representation	composition function
		Existing Methods		
Prefix Tuning	$\text{softmax}(xW_q P_k^\top)P_v$	parallel	head attn	$h \leftarrow (1-\lambda)h + \lambda\Delta h$
Adapter	$\text{ReLU}(hW_{\text{down}})W_{\text{up}}$	sequential	ffn/attn	$h \leftarrow h + \Delta h$
LoRA	$xW_{\text{down}}W_{\text{up}}$	parallel	attn key/val	$h \leftarrow h + s \cdot \Delta h$
		Proposed Variants		
Parallel adapter	$\text{ReLU}(hW_{\text{down}})W_{\text{up}}$	parallel	ffn/attn	$h \leftarrow h + \Delta h$
Muti-head parallel adapter	$\text{ReLU}(hW_{\text{down}})W_{\text{up}}$	parallel	head attn	$h \leftarrow h + \Delta h$
Scaled parallel adapter	$\text{ReLU}(hW_{\text{down}})W_{\text{up}}$	parallel	ffn/attn	$h \leftarrow h + s \cdot \Delta h$

Reference: TOWARDS A UNIFIED VIEW OF PARAMETER-EFFICIENT TRANSFER LEARNING, ICLR 2022

[2021]LoRA

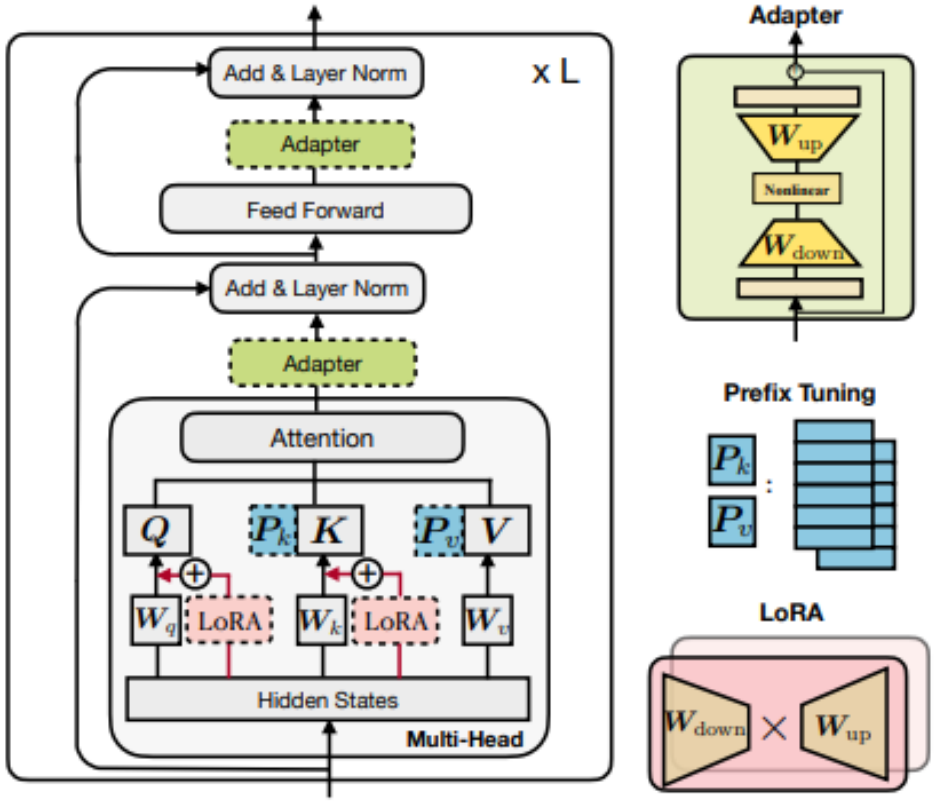


Figure 1: Illustration of the transformer architecture and several state-of-the-art parameter-efficient tuning methods. We use blocks with dashed borderlines to represent the added modules by those methods.

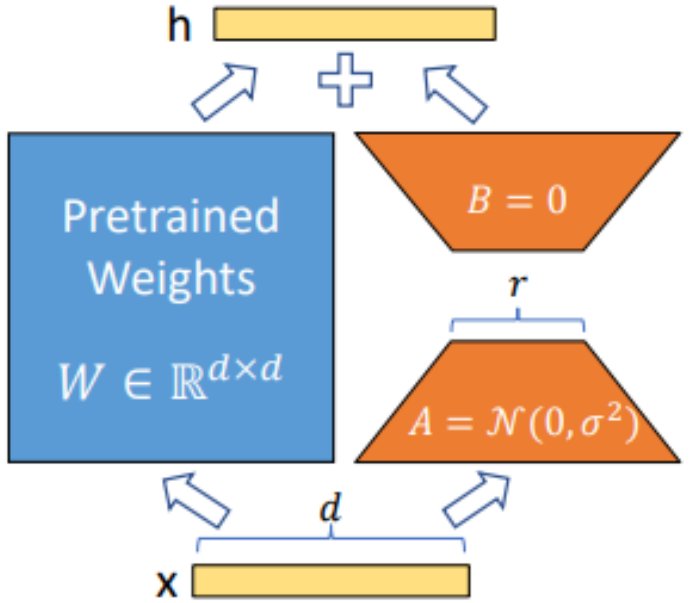


Figure 1: Our reparametrization. We only train A and B.

[2021]LoRA

4.1 Low-Rank-Parametrized Update Matrices

A neural network contains many dense layers which perform matrix multiplication. The weight matrices in these layers typically have full-rank. When adapting to a specific task, Aghajanyan et al. (2020) shows that the pre-trained language models have a low "instrisic dimension" and can still learn efficiently despite a random projection to a smaller subspace. Inspired by this, we hypothesize the updates to the weights also have a low "intrinsic rank" during adaptation. For a pre-trained weight matrix $W_0 \in \mathbb{R}^{d\times k}$, we constrain its update by representing the latter with a low-rank decomposition $W_0 + \Delta W = W_0 + BA$, where $B \in \mathbb{R}^{d\times r}$, $A \in \mathbb{R}^{r\times k}$, and the rank $r \ll \min(d, k)$. During training, W_0 is frozen and does not receive gradient updates, while A and B contain trainable parameters. Note both W_0 and $\Delta W = BA$ are multiplied with the same input, and their respective output vectors are summed coordinate-wise. For $h = W_0 x$, our modified forward pass yields:

$$h = W_0 x + \Delta W x = W_0 x + BAx \tag{3}$$

We illustrate our reparametrization in Figure 1. We use a random Gaussian initialization for A and zero for B, so $\Delta W = BA$ is zero at the beginning of training. We then scale $\Delta W x$ by $\frac{\alpha}{r}$, where α is a constant in r. When optimizing with Adam, tuning α is roughly the same as tuning the learning rate if we scale the initialization appropriately. As a result, we simply set α to the first r we try and do not tune it. This scaling helps to reduce the need to retune hyperparameters when we vary r (Yang & Hu, 2021).

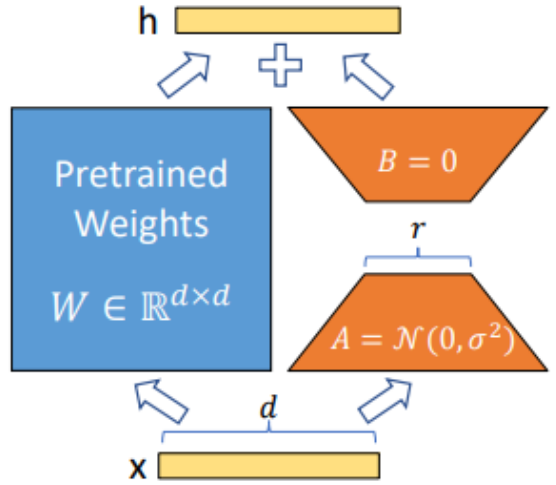


Figure 1: Our reparametrization. We only train A and B.

[2021]LoRA

```python
class Linear(nn.Linear, LoRALayer):
    # LoRA implemented in a dense layer
    def __init__(
        self,
        in_features: int,
        out_features: int,
        r: int = 0,
        lora_alpha: int = 1,
        lora_dropout: float = 0.,
        fan_in_fan_out: bool = False, # Set this to True if the layer to replace stores weight like (fan_in, fan_out)
        merge_weights: bool = True,
        **kwargs
    ):
        nn.Linear.__init__(self, in_features, out_features, **kwargs)
        LoRALayer.__init__(self, r=r, lora_alpha=lora_alpha, lora_dropout=lora_dropout,
                           merge_weights=merge_weights)

        self.fan_in_fan_out = fan_in_fan_out
        # Actual trainable parameters
        if r > 0:
            self.lora_A = nn.Parameter(self.weight.new_zeros((r, in_features)))
            self.lora_B = nn.Parameter(self.weight.new_zeros((out_features, r)))
            self.scaling = self.lora_alpha / self.r
            # Freezing the pre-trained weight matrix
            self.weight.requires_grad = False
        self.reset_parameters()
        if fan_in_fan_out:
            self.weight.data = self.weight.data.T
```

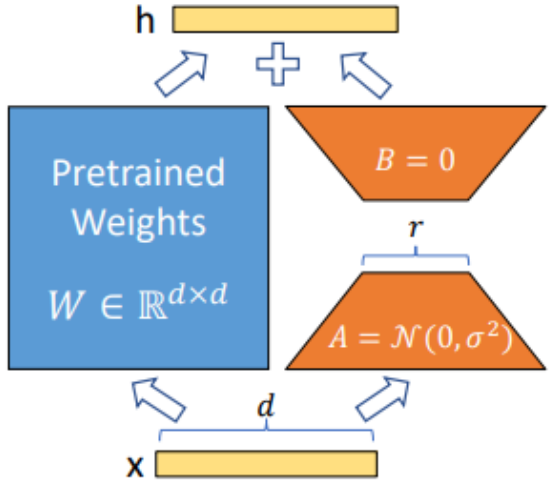


Figure 1: Our reparametrization. We only train A and B.

Background

[2022]Textual Inversion

An Image is Worth One Word: Personalizing Text-to-Image Generation using Textual Inversion

Rinon Gal[1,2*] Yuval Alaluf[1] Yuval Atzmon[2] Or Patashnik[1]

Amit H. Bermano[1] Gal Chechik[2] Daniel Cohen-Or[1]

[1]Tel-Aviv University [2]NVIDIA

[2022]Textual Inversion

Figure 1: (left) We find new pseudo-words in the embedding space of a pre-trained text-to-image model which describe specific concepts. (right) These pseudo-words can be composed into new sentences, placing our targets in new scenes, changing their style or composition, or ingraining them into new products.

[2022]Textual Inversion

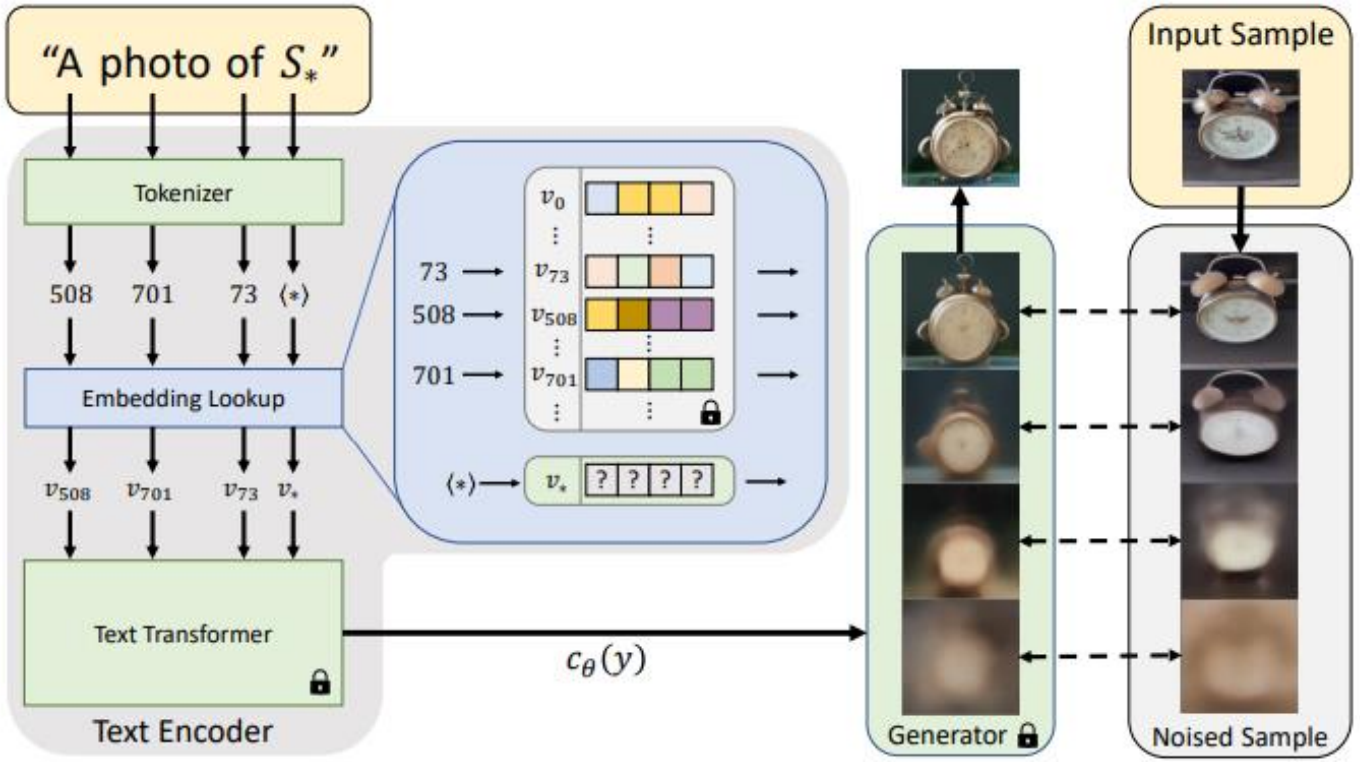


Figure 2: Outline of the text-embedding and inversion process. A string containing our placeholder word is first converted into tokens (*i.e.* word or sub-word indices in a dictionary). These tokens are converted to continous vector representations (the "embeddings", v). Finally, the embedding vectors are transformed into a single conditioning code $c_\theta(y)$ which guides the generative model. We optimize the embedding vector v_* associated with our pseudo-word S_*, using a reconstruction objective.


Background

[2022]DreamBooth

DreamBooth: Fine Tuning Text-to-Image Diffusion Models for Subject-Driven Generation

Nataniel Ruiz[*,1,2] Yuanzhen Li[1] Varun Jampani[1]
Yael Pritch[1] Michael Rubinstein[1] Kfir Aberman[1]
[1] Google Research [2] Boston University

[2022]DreamBooth

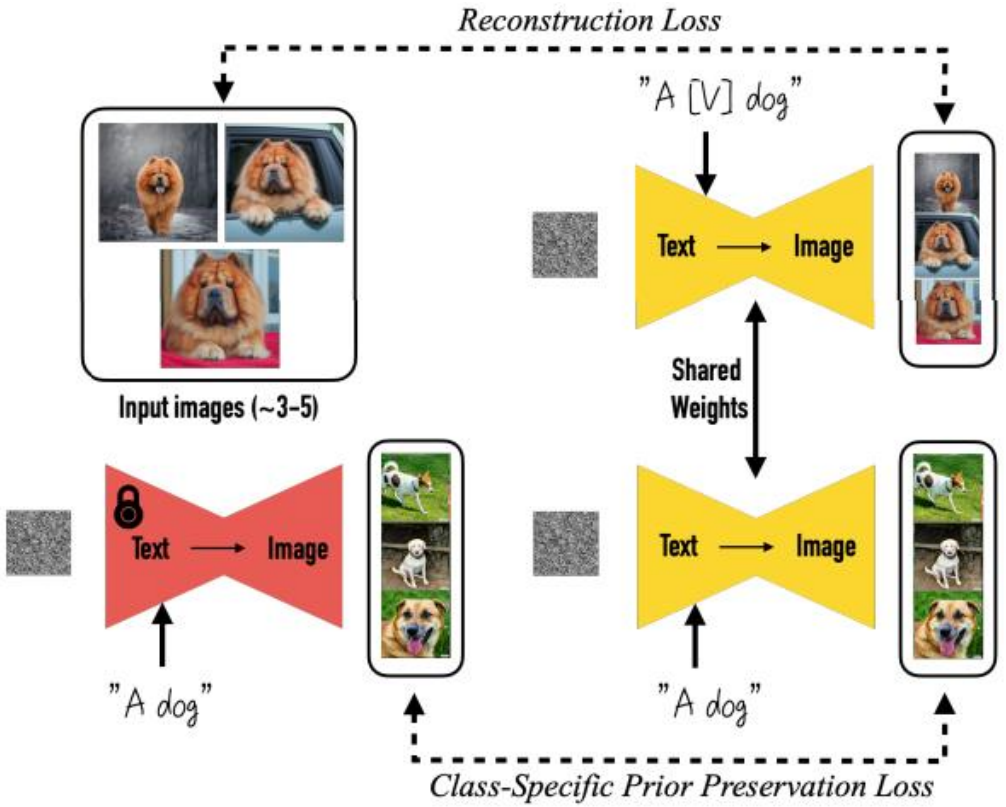
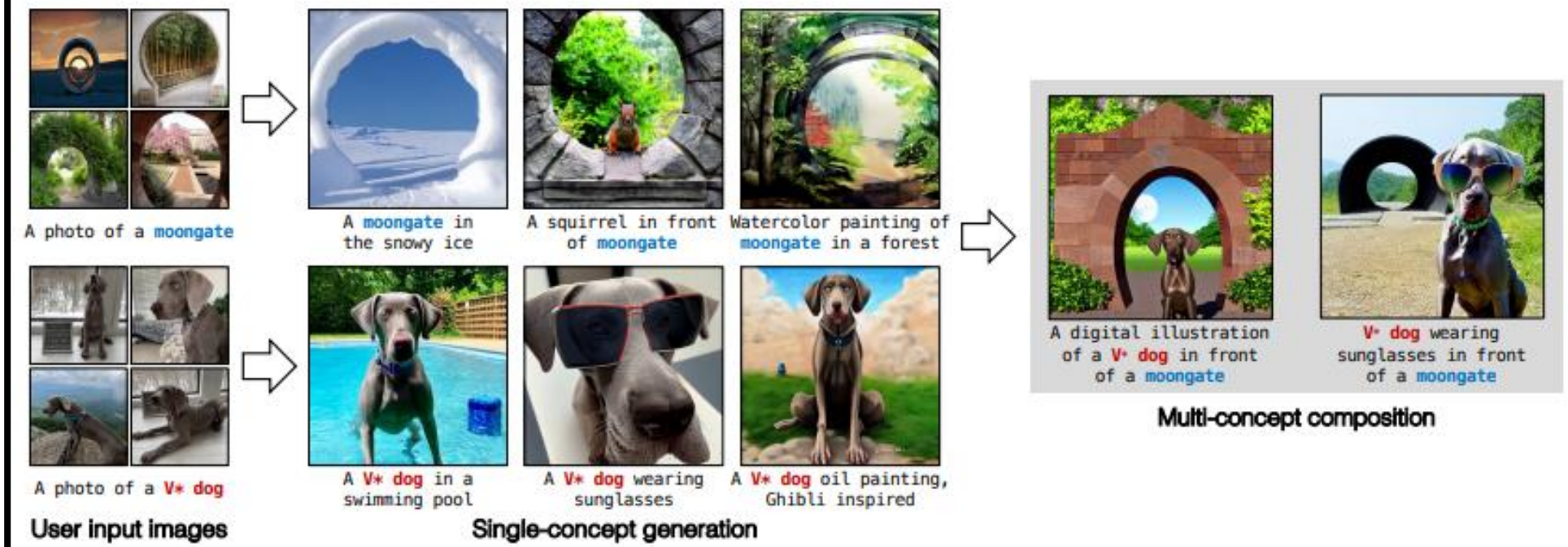


Figure 3. **Fine-tuning.** Given $\sim 3-5$ images of a subject we fine-tune a text-to-image diffusion model with the input images paired with a text prompt containing a unique identifier and the name of the class the subject belongs to (e.g., "A [V] dog"), in parallel, we apply a class-specific prior preservation loss, which leverages the semantic prior that the model has on the class and encourages it to generate diverse instances belong to the subject's class using the class name in a text prompt (e.g., "A dog").

Multi-Concept Customization of Text-to-Image Diffusion

Nupur Kumari[1] Bingliang Zhang[2] Richard Zhang[3] Eli Shechtman[3] Jun-Yan Zhu[1]
[1]Carnegie Mellon University [2]Tsinghua University [3]Adobe Research



Abstract

- While generative models produce high-quality images of concepts learned from a large-scale database, a user often wishes to synthesize **instantiations** of their own concepts (for example, their family, pets, or items).

- Can we teach a model to quickly acquire a new concept, given a few examples? Furthermore, can we **compose multiple new concepts together**?

- We propose **Custom Diffusion**, an efficient method for augmenting existing text-to-image models. We find that only optimizing a few parameters in the text-to-image conditioning mechanism is sufficiently powerful to represent new concepts while enabling fast tuning **(~ 6 minutes)**.

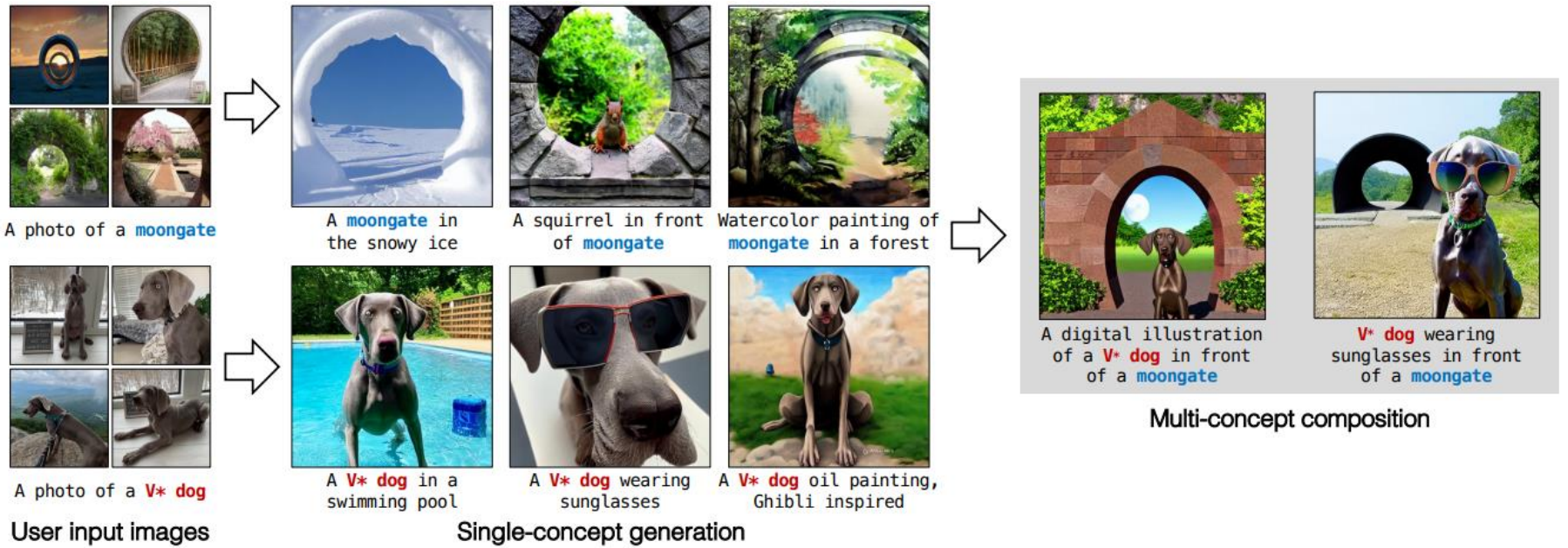
- Additionally, **we can jointly train for multiple concepts** or **combine multiple fine-tuned models into one** via closed-form constrained optimization.

- Our fine-tuned model generates variations of multiple, new concepts and seamlessly composes them with existing concepts in novel settings.

- Our method outperforms several baselines and concurrent works, regarding both qualitative and quantitative evaluations, while being memory and computationally efficient.

Introduction

- Recent text-to-image models revolutionize image generation. Thus, users can create diverse, high-quality images with text prompts.
- However, despite the diverse, general capability of such models, users often wish to synthesize specific concepts from their own personal lives (e.g., loved ones, personal items). → **Need for model customization arises**
- **Challenges: model forgetting/changing existing concepts, overfitting, compositional fine-tuning.**
- **Compositional fine-tuning:** learning and composing multiple new concepts together.



Introduction

In this work,

- Propose **Custom Diffusion** fine-tuning technique for text-to-image models.
- Computationally and memory efficient method.
- Targets small subset of model weights in cross-attention layers (key and value mapping from text to latent features).
- Prevents model forgetting using real images, similar captions.
- Augmentation during fine-tuning for faster convergence, better results.
- Supports injecting multiple concepts simultaneously or separately.
- Built on Stable Diffusion, tested on various datasets, as few as four images.
- Outperforms concurrent works for single concept addition and composing multiple concepts.
- Requires only 3% of model weights and faster fine-tuning time (2-4x).

Methods

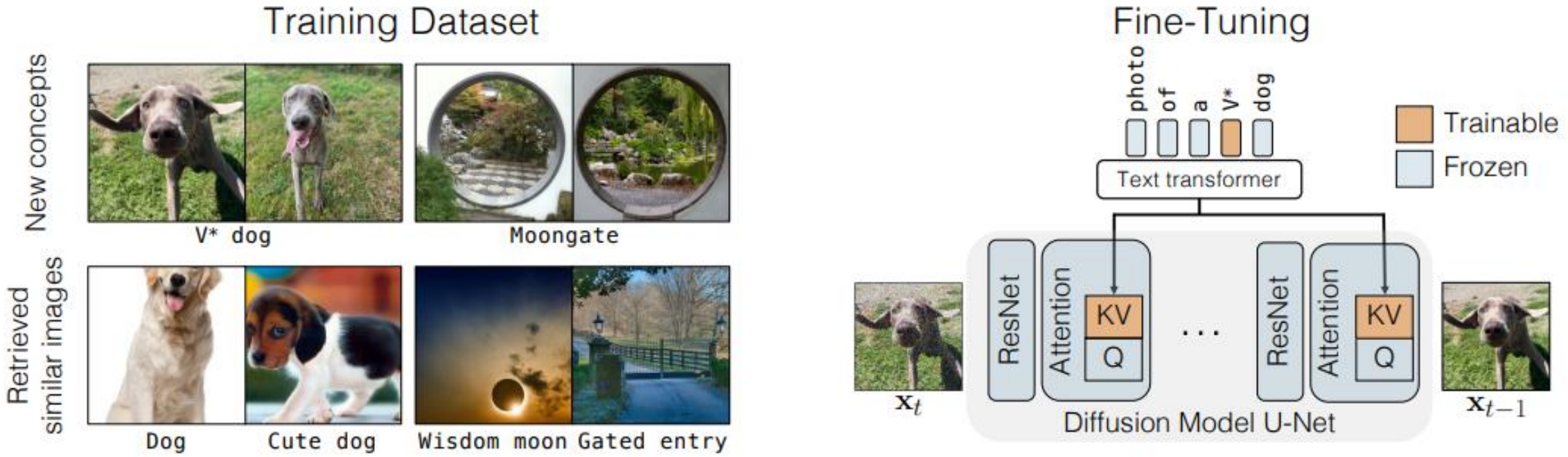


Figure 2. **Custom Diffusion.** Given images of new concepts, we retrieve real images with similar captions as the given concepts and create the training dataset for fine-tuning, as shown on the left. To represent personal concepts of a general category, we introduce a new modifier token $\mathtt{V}^*$, used in front of the category name. During training, we optimize key and value projection matrices in the diffusion model cross-attention layers along with the modifier token. The retrieved real images are used as a regularization dataset during fine-tuning.

Methods

- **Learning objective of diffusion models.**

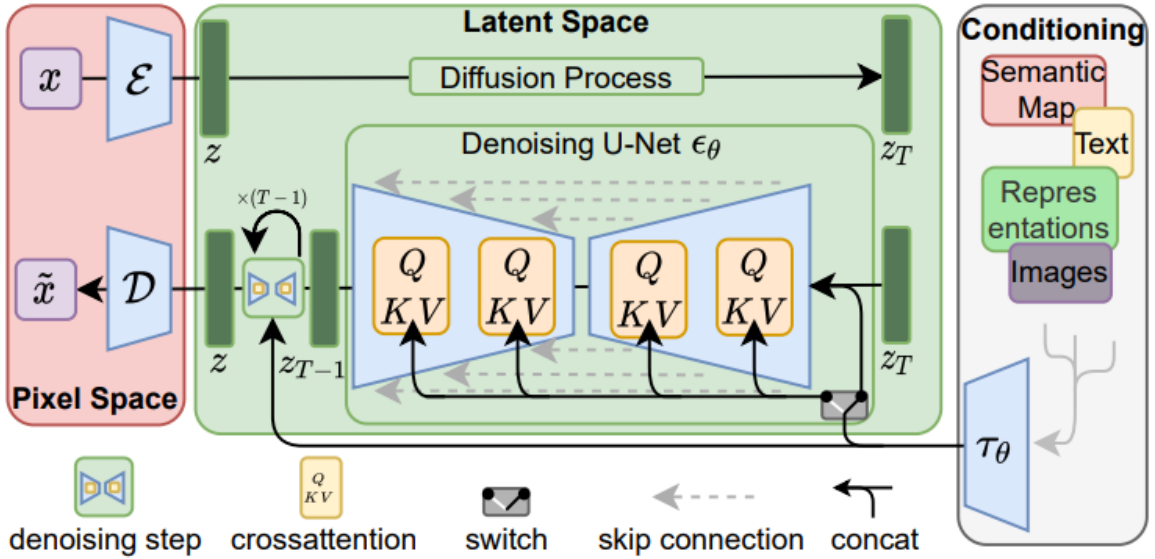
Learning objective of diffusion models. Diffusion models [28, 68] are a class of generative models that aim to approximate the original data distribution $q(\mathbf{x}_0)$ with $p_\theta(\mathbf{x}_0)$:

$$p_\theta(\mathbf{x}_0) = \int \Big[p_\theta(\mathbf{x}_T) \prod p^t_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) \Big] d\mathbf{x}_{1:T}, \quad (1)$$

where $\mathbf{x}_1$ to $\mathbf{x}_T$ are latent variables of a forward Markov chain s.t. $\mathbf{x}_t = \sqrt{\alpha_t}\mathbf{x}_0 + \sqrt{1-\alpha_t}\epsilon$. The model is trained to learn the reverse process of a fixed-length (usually 1000) Markov chain. Given noisy image $\mathbf{x}_t$ at timestep t, the model learns to denoise the input image to obtain $\mathbf{x}_{t-1}$. The training objective of the diffusion model can be simplified to:

$$\mathbb{E}_{\epsilon,\mathbf{x},\mathbf{c},t}[w_t||\epsilon - \epsilon_\theta(\mathbf{x}_t, \mathbf{c}, t)||], \quad (2)$$

where ϵ_θ is the model prediction and w_t is a time-dependent weight on the loss. The model is conditioned on timestep t and can be further conditioned on any other modality $\mathbf{c}$, e.g., text. During inference, a random Gaussian image (or latent) $\mathbf{x}_T$ is denoised for fixed timesteps using the model [69].



- Naive fine-tuning: update all layers, risks inefficiency and overfitting.
- Goal: identify minimal set of weights for efficient fine-tuning.

Methods

- **Rate of change of weights.**

- Analyze change in parameters for each layer in finetuned model.
- Parameters from three layer types: cross-attention, self-attention, and others (e.g., convolutional blocks, normalization).
- **Cross-attention layer parameters have higher Δ, crucial for fine-tuning.**
- **Cross-attention layers only 5% of total parameter count, suggesting significant role.**

i.e., $\Delta_l = ||\theta'_l - \theta_l||/||\theta_l||$, where θ'_l and θ_l are the updated and pretrained model parameters of layer l.

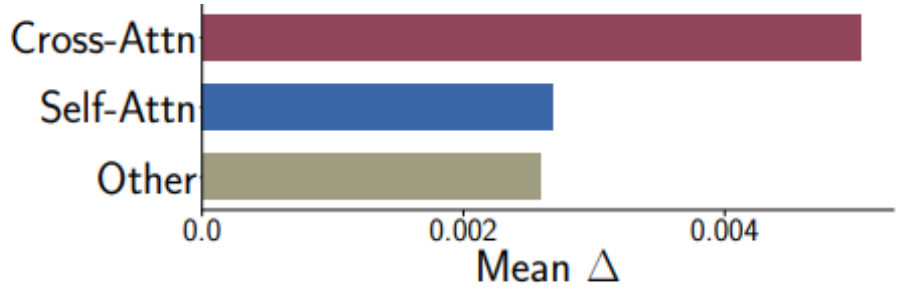


Figure 3. **Analysis of change in weights** on updating all network weights during fine-tuning. The mean change in the cross-attention layers is significantly higher than other layers even though they only make up 5% of the total parameter count.

Methods

- **Model fine-tuning.**
  - Cross-attention block modifies latent features based on condition features (e.g., text).
  - The task of fine-tuning aims at updating the mapping from given text to image distribution, and the text features are only input to Wk and Wv projection matrix in the cross-attention block (i.e. **text-to-image distribution mapping**).
  - **Therefore, we propose to only update Wk and Wv parameters of the diffusion model during the fine-tuning process.**
  - As shown in our experiments, this is sufficient to update the model with a new text-image paired concept.
  - Figure 4 shows an instance of the cross-attention layer and the trainable parameters.

Given text features $\mathbf{c} \in \mathbb{R}^{s \times a}$ and latent image features $\mathbf{f} \in \mathbb{R}^{(h \times w) \times l}$, a single-head cross-attention [73] operation consists of $Q = W^q\mathbf{f}$, $K = W^k\mathbf{c}$, $V = W^v\mathbf{c}$, and a weighted sum over value features as:

$$\text{Attention}(Q, K, V) = \text{Softmax}\left(\frac{QK^T}{\sqrt{d'}}\right)V, \quad (3)$$

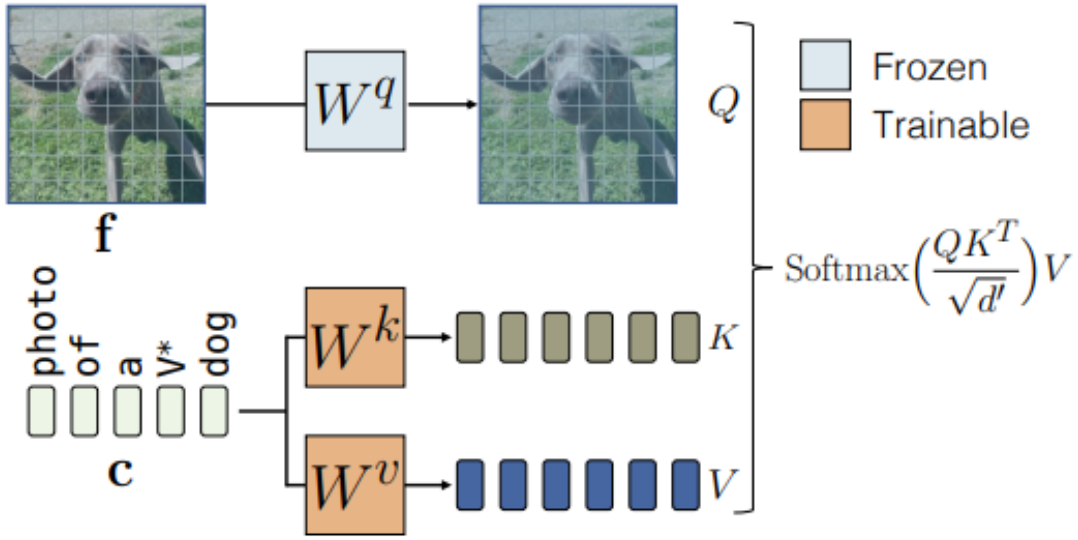


Figure 4. **Single-head Cross-Attention.** Latent image feature $\mathbf{f}$ and text feature $\mathbf{c}$ are projected into query Q, key K, and value V. Output is a weighted sum of values, weighted by the similarity between the query and key features. We highlight the updated parameters W^k and W^v in our method.

Methods

Figure 5. **Role of regularization data in mitigating overfitting behavior during fine-tuning.**

- **Regularization dataset**
  - Fine-tuning can lead to **language drift**, model forgets associations.
  - Regularization needed to prevent leaks and maintain existing concepts.
  - **Select 200 regularization images with similar captions to target text prompt.**
  - Use threshold of 0.85 similarity in **CLIP text encoder** feature space.

Methods

- **Joint training on multiple concepts.**

  - Joint training for multiple concepts by combining individual concept datasets.
  - Use different modifier tokens V_i^* initialized with different rarely-occurring tokens for target concepts.
  - Initialize with rarely-occurring tokens, optimize with cross-attention key and value matrices.
  - Better results for composing two concepts compared to methods that fine-tune all weights (e.g., DreamBooth). → Figure. 7

Methods

- **Figure 7. Multi-concept fine-tuning results.**

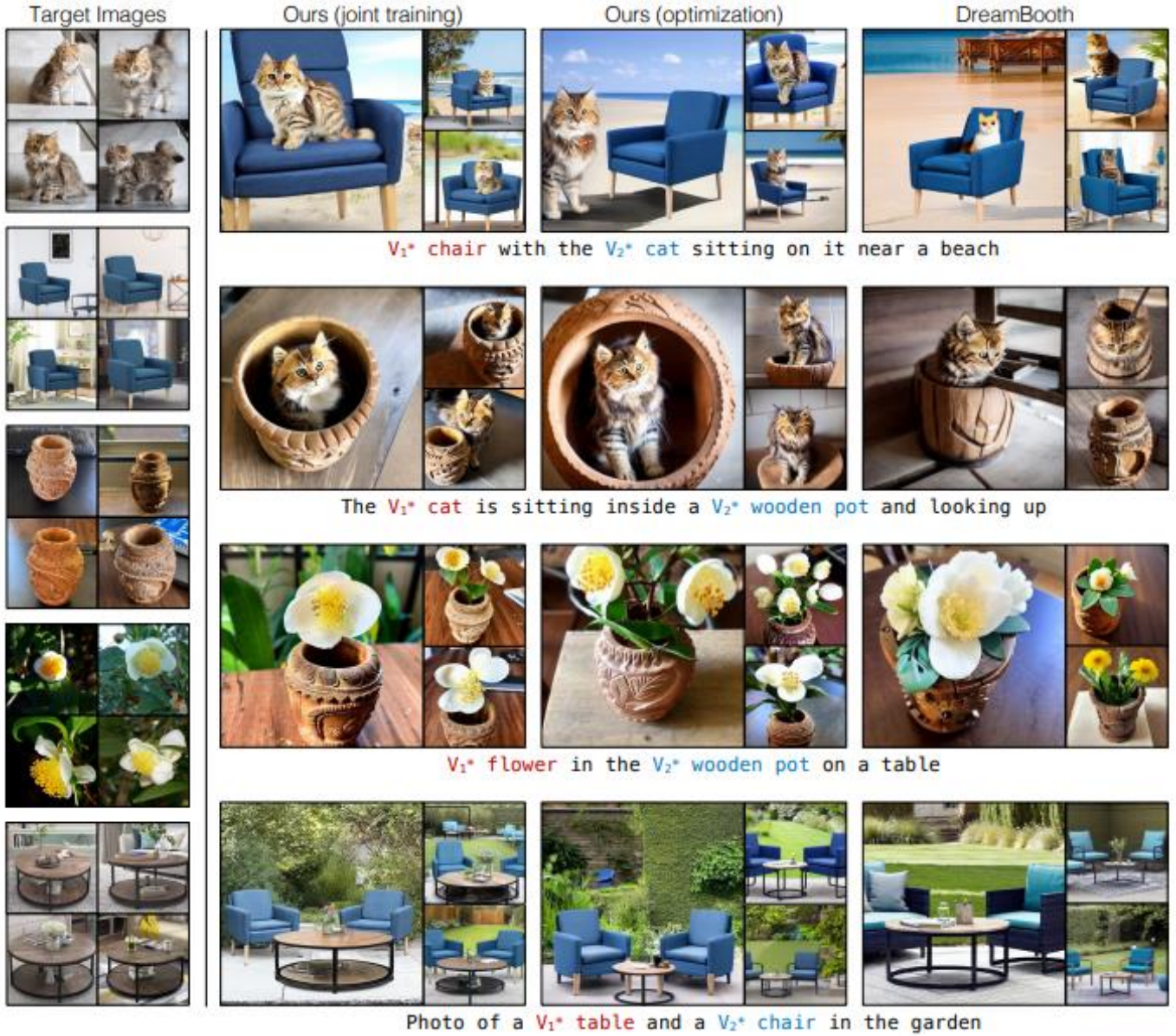
First row: our method has higher visual similarity with the personal cat and chair images shown in the first column while following the text condition.

Second row: DreamBooth omits the cat in 3 out of 4 images, whereas our method generates both cats and wooden pots.

Third row: our method generates the target flower in the wooden pot while maintaining the visual similarity to the target images.

Fourth row: generating the target table and chair together in a garden.

For all settings, our optimization-based approach and joint training perform better than DreamBooth, and joint training performs better than the optimization-based method.



Methods

- **Constrained optimization to merge concepts.**
  - Method updates key and value projection matrices for text features.
  - Merges matrices to allow generation with multiple fine-tuned concepts.
  - Composition objective formulated as constrained least squares problem.
  - Closed-form solution using method of Lagrange multipliers.
  - Faster than joint training, coherent generation of two new concepts in single scene.

Methods

- **Constrained optimization to merge concepts.**

Constrained optimization to merge concepts. As our method only updates the key and value projection matrices corresponding to the text features, we can subsequently merge them to allow generation with multiple fine-tuned concepts. Let set $\{W^k_{0,l}, W^v_{0,l}\}_{l=1}^L$ represent the key and value matrices for all L cross-attention layers in the pretrained model and $\{W^k_{n,l}, W^v_{n,l}\}_{l=1}^L$ represent the corresponding updated matrices for added concept $n \in \{1 \cdots N\}$. As our subsequent optimization applies to all layers and key-value matrices, we will omit superscripts $\{k, v\}$ and layer l for notational clarity. We formulate the composition objective as the following constrained least squares problem:

$$\begin{aligned} \hat{W} = \arg\min_W ||WC_{\text{reg}}^\top - W_0 C_{\text{reg}}^\top||_F \\ \text{s.t. } WC^\top = V, \text{ where } C = [\mathbf{c}_1 \cdots \mathbf{c}_N]^\top \\ \text{and } V = [W_1\mathbf{c}_1^\top \cdots W_N\mathbf{c}_N^\top]^\top. \end{aligned} \tag{4}$$

Here, $C \in \mathbb{R}^{s \times d}$ is the text features of dimension d. These are compiled of s target words across all N concepts, with all captions for each concept flattened out and concatenated. Similarly, $C_{\text{reg}} \in \mathbb{R}^{s_{\text{reg}} \times d}$ consists of text features of $\sim$1000 randomly sampled captions for regularization. Intuitively, the above formulation aims to update the matrices in the original model, such that the words in target captions in C are mapped consistently to the values obtained from fine-tuned concept matrices. The above objective can be solved in closed form, assuming C_{reg} is non-degenerate and the solution exists, by using the method of Lagrange multipliers [8]:

$$\begin{aligned} \hat{W} = W_0 + \mathbf{v}^\top \mathbf{d}, \text{ where } \mathbf{d} = C(C_{\text{reg}}^\top C_{\text{reg}})^{-1} \\ \text{and } \mathbf{v}^\top = (V - W_0 C^\top)(\mathbf{d}C^\top)^{-1}. \end{aligned} \tag{5}$$

We show full derivation in the Appendix A. Compared to joint training, our optimization-based method is faster ($\sim$2 seconds) if each individual fine-tuned model exists. Our proposed methods lead to the coherent generation of two new concepts in a single scene, as shown in Section 4.2.

Methods

- **Constrained optimization to merge concepts.**

A. Multi-Concept Optimization Based Method

To compose multiple concepts together, we solved a constrained least squares problem (introduced in Section 3.2, Eqn. 4). We solved the problem in closed form as shown in Eqn. 5 in the main paper. Here, we show the full derivation. We first restate the objective below.

$$\hat{W} = \arg\min_{W} ||WC_{\text{reg}}^{\top} - W_0 C_{\text{reg}}^{\top}|| \quad \text{s.t. } WC^{\top} = V, \text{ where } C = [\mathbf{c}_1 \cdots \mathbf{c}_N]^{\top} \text{ and } V = [W_1\mathbf{c}_1^{\top} \cdots W_N\mathbf{c}_N^{\top}]^{\top}. \tag{6}$$

Here, the matrix norms are the Frobenius norm, $W \in \mathbb{R}^{o\times d}$ is the matrix from the pretrained model, $C \in \mathbb{R}^{s\times d}$ is the text features of dimension d. These are compiled of s target words across all N concepts, with all captions for each concept flattened out and concatenated. Similarly, $C_{\text{reg}} \in \mathbb{R}^{s_{\text{reg}}\times d}$ consists of text features of ~1000 randomly sampled captions for regularization.

By using the method of Lagrange multipliers [8], we need to minimize the following objective:

$$L = \frac{1}{2}||WC_{\text{reg}}^{\top} - W_0 C_{\text{reg}}^{\top}|| - \text{trace}(\mathbf{v}(WC^{\top} - V)), \tag{7}$$

here $\mathbf{v} \in \mathbb{R}^{s\times o}$ is the Lagrangian multiplier corresponding to the constraints. Differentiating the above objective and equating it to 0, we obtain:

$$\begin{aligned} &WC_{\text{reg}}^{\top}C_{\text{reg}} - W_0 C_{\text{reg}}^{\top}C_{\text{reg}} - \mathbf{v}^{\top}C = 0 \\ &\implies W = W_0 + \mathbf{v}^{\top}C(C_{\text{reg}}^{\top}C_{\text{reg}})^{-1}. \end{aligned} \tag{8}$$

We assume C_{reg} is non-degenerate. Using the above solution in Eqn. 6, $WC^{\top} = V$, we obtain:

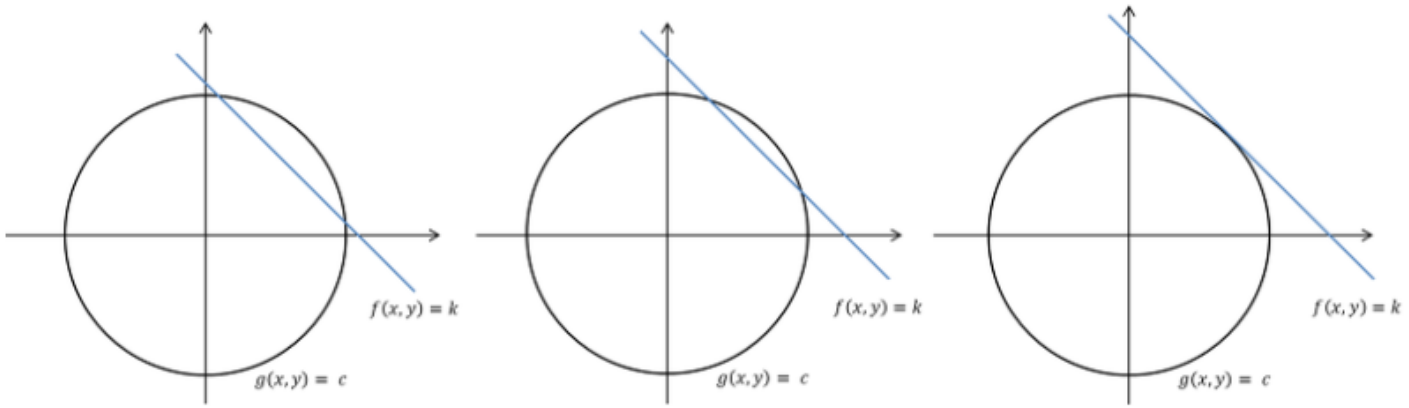
$$\begin{aligned} &(W_0 + \mathbf{v}^{\top}C(C_{\text{reg}}^{\top}C_{\text{reg}})^{-1})C^{\top} = V \\ &\text{Let } \mathbf{d} = C(C_{\text{reg}}^{\top}C_{\text{reg}})^{-1} \\ &\mathbf{v}^{\top} = (V - W_0C^{\top})(\mathbf{d}C^{\top})^{-1}. \end{aligned} \tag{9}$$

Lagrange multiplier, KKT (Karush-Kuhn-Tucker)

- 라그랑주 승수법 (Lagrange multiplier method)은 프랑스의 수학자 조세프루이 라그랑주 (Joseph-Louis Lagrange)가 제약 조건이 있는 최적화 문제를 풀기 위해 고안한 방법이다.
- 라그랑주 승수법은 어떠한 문제의 최적점을 찾는 것이 아니라, 최적점이 되기 위한 조건을 찾는 방법이다. 즉, 최적해의 필요조건을 찾는 방법이다.

1. 기하학적 해석

라그랑주 승수법의 기본 가정은 "제약 조건 g를 만족하는 f의 최솟값 또는 최댓값은 f와 g가 접하는 지점에 존재할 수도 있다."는 것이다. 아래의 [그림 1]은 제약 조건 $g(x, y) = c$를 만족하는 $f(x, y)$의 최댓값을 구하는 문제를 나타낸다. 만약, $f(x, y)$의 최댓값을 k라고 하면, k는 기하학적으로 xy평면에서 직선의 y축 절편 (y-intercept)을 나타낸다. 따라서, 제약 조건 $g(x, y) = c$와 $f(x, y)$가 접할 때 $f(x, y)$는 최대가 된다 (제 3사분면에서 접하는 경우는 최소가 된다).



[그림 1] 제약 조건 $g(x, y) = c$를 만족하는 $f(x, y)$의 최댓값 문제에 대한 기하학적 표현

Lagrange multiplier, KKT (Karush-Kuhn-Tucker)

공식

$h(x, \lambda) = f(x) + \sum_{j=1}^{M} \lambda_j g_j(x)$

예시

$f(x_1, x_2) = x_1^2 + x_2^2$를 최소화하며,
$g(x_1, x_2) = x_1 + x_2 - 1 = 0$ 제약조건을 만족해야 할 때
$h(x_1, x_2, \lambda) = f(x_1, x_2) + \lambda g(x_1, x_2) = x_1^2 + x_2^2 + \lambda(x_1 + x_2 - 1)$로 식을 전개할 수 있다.

최저점을 구하기 위해선 위의 식에서 각 변수에 대해 편미분을 진행하면 된다.

$$\frac{\partial h}{\partial x_1} = 2x_1 + \lambda = 0$$

$$\frac{\partial h}{\partial x_2} = 2x_2 + \lambda = 0$$

$$\frac{\partial h}{\partial \lambda} = x_1 + x_2 - 1 = 0$$

위의 해를 풀면

$$x_1 = x_2 = \frac{1}{2}, \quad \lambda = -1$$

최소점을 찾을 수 있다.

최소점을 찾는 것이 목표기 때문에 라그랑주 승수값은 필요 없다.

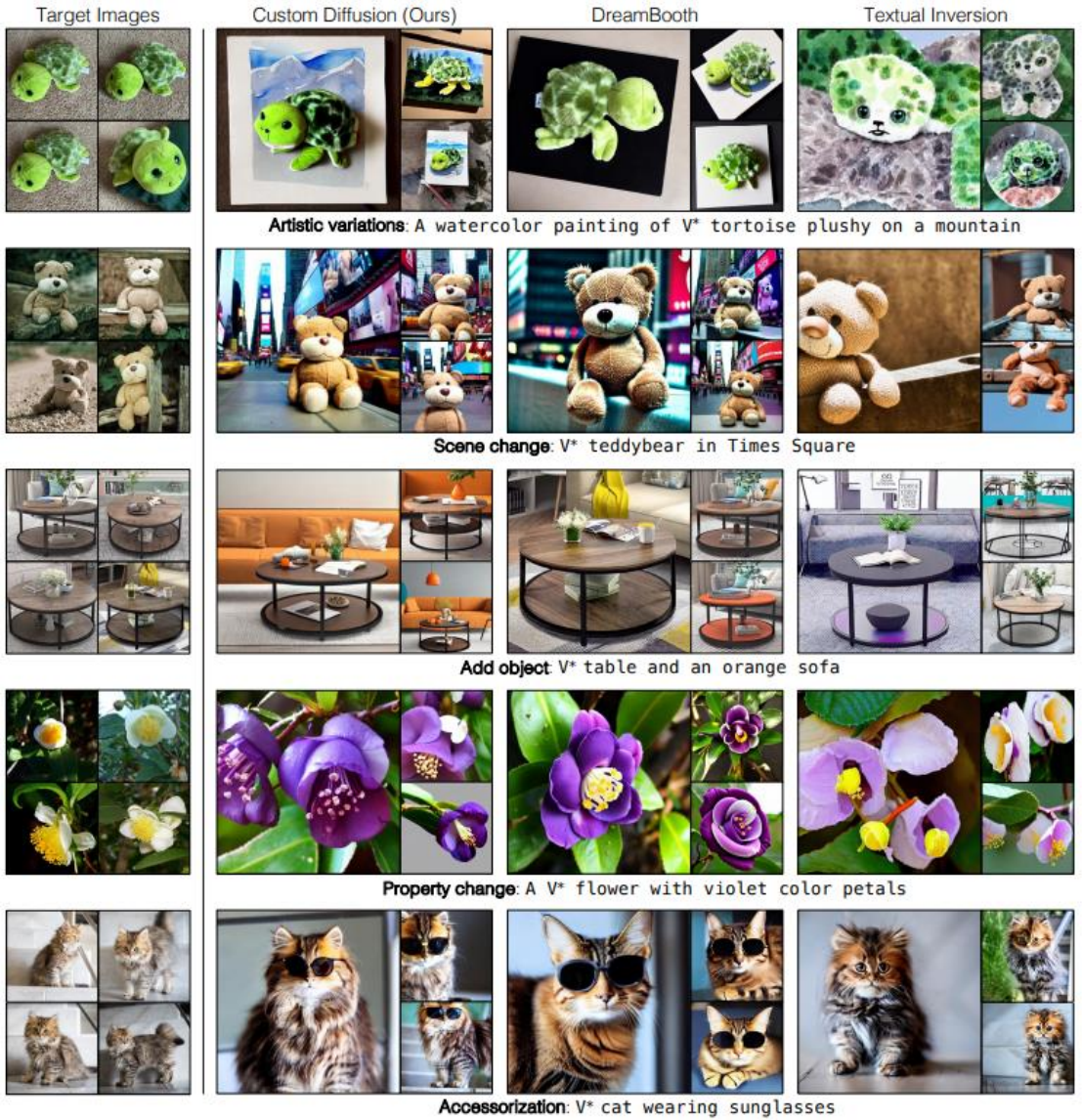
Abstract & Conclusion

- **Training details.**

  - Training for 250 steps (single-concept) or 500 steps (two-concept joint training).
  - Batch size: 8; Learning rate: 8×10^{-5}.
  - **During training, we also randomly resize only the target images from 0.4−1.4× and append the prompt “very small”, “far away” or “zoomed in”, “close up” accordingly to the text prompt based on resize ratio.**
  - Backpropagate loss on valid regions → cross-attention layer.
  - This leads to faster convergence and improved results.

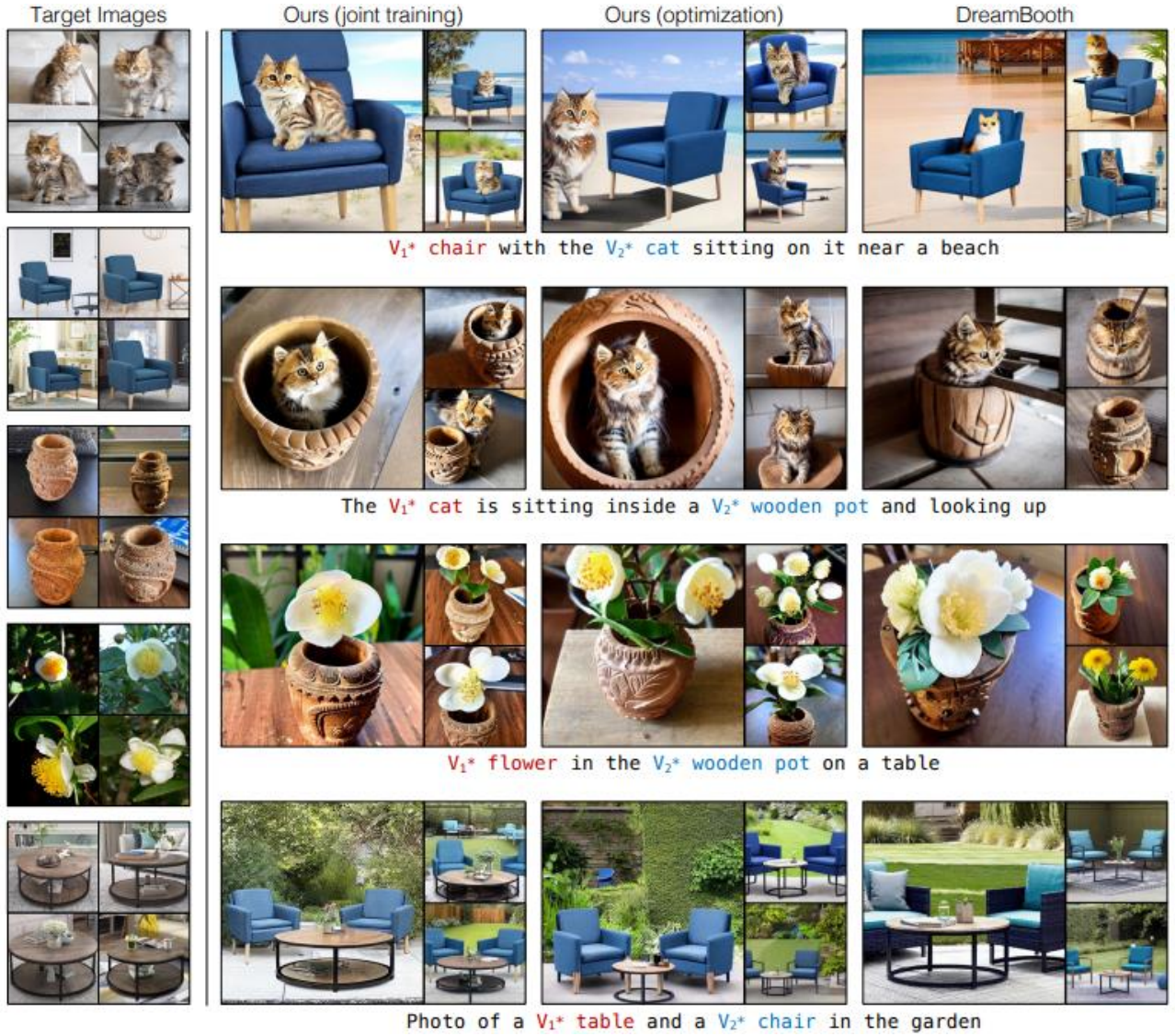
Experiments

- **Single-Concept Fine-tuning Results**



Experiments

- **Multiple-Concept Fine-tuning Results**



Experiments

- **Single- and Multiple-Concept Fine-tuning Results**

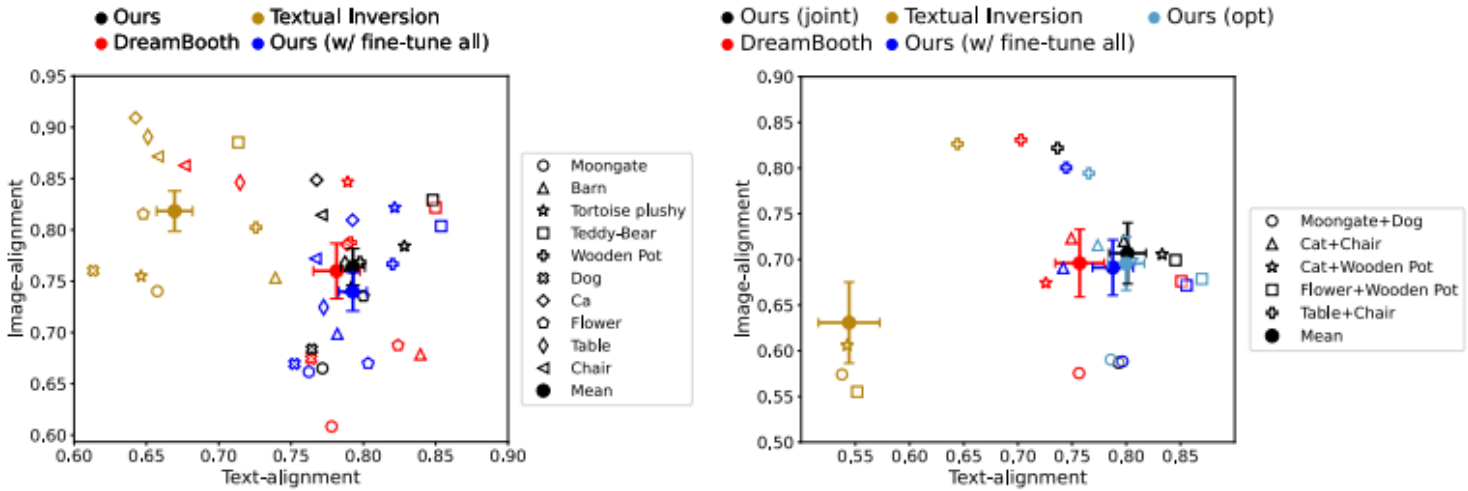


Figure 8. **Text- and image-alignment for single-concept (left) and multi-concept (right) fine-tuning**. Compared to DreamBooth, Textual Inversion, and our w/ fine-tune all baseline, our method lies further along the upper right corner (with less variance). There exists a trade-off between high similarity to target images vs. text-alignment on new prompts. Keeping in consideration this trade-off, our method is on-par or better than the baselines. For multi-concept both joint training and optimization based method are better than other baselines.

	Method	Text-alignment	Image-alignment	KID (validation)
Single-Concept	Textual Inversion	0.670	**0.827**	22.27
	DreamBooth	0.781	0.776	32.53
	Ours (w/ fine-tune all)	**0.795**	0.748	**19.27**
	Ours	**0.795**	0.775	20.96
Multi-Concept	Textual Inversion	0.544	0.630	
	DreamBooth	0.783	0.695	
	Ours (w/ fine-tune all)	0.787	0.691	—
	Ours (Sequential)	0.797	0.700	
	Ours (Optimization)	0.800	0.695	
	Ours (Joint)	**0.801**	**0.706**	

Table 1. **Quantitative comparisons.** *Top row*: single-concept fine-tuning evaluation averaged across datasets. The last column shows the KID ($\times 10^3$) between real validation set images and generated images with the same caption. Since our method uses a regularization set of real images, it achieves lower KID and even improves slightly over the pretrained model. Textual Inversion has the same KID as the pretrained model, as it does not update the model. *Bottom row*: evaluation on multi-concept averaged across the five composition pairs. We show individual scores for all in Table 6 and 7 in the Appendix. We also evaluate single-concept fine-tuned models on FID [27] (MS-COCO [41]) in Table 5 and show the trend of image-, text-alignment with training steps in Figure 17.

Experiments

- **Human Preference Study**

  - Human preference study using **Amazon Mechanical Turk**.
  - Paired test comparing: **DreamBooth, Textual Inversion, and Ours (w/ fine-tune all)**.
  - **Text-alignment**: Assess image consistency with text prompt.
  - **Image-alignment**: Assess how well image represents objects in target images.
  - 800 responses collected for each questionnaire.
  - Importance of updating only cross-attention parameters demonstrated.

Ours	Textual Inversion		DreamBooth		Ours (w/ fine-tune all)	
	Text Alignment	Image Alignment	Text Alignment	Image Alignment	Text Alignment	Image Alignment
Single-concept	**72.62** ± 2.38%	**51.62** ± 2.62%	**53.50** ± 2.64%	**56.62** ± 2.44%	**55.17** ± 2.55%	**53.99** ± 2.44%
Multi-concept (Joint)	**86.65** ± 2.25 %	**81.89** ± 2.09 %	**56.39** ± 2.46 %	**61.80** ± 2.59%	**59.00** ± 2.61 %	**59.12** ± 2.72%
Multi-concept (Optimization)	**81.22** ± 2.72%	**83.11** ± 2.18%	**57.00** ± 2.62 %	**61.75** ± 2.68%	**57.60** ± 2.43 %	**53.49** ± 2.71%

Table 2. **Human preference study.** For each paired comparison, our method is preferred (over $\geq 50\%$) over the baseline in both image- and text-alignment. Textual Inversion seems to overfit to target images and thus has a similar image-alignment as ours but performs worse on text-alignment in the single-concept setting.

Experiments

- KID(Kernel Inception Distance)는 MMD(Maximum Mean Discrepancy)를 feature space에서 진행하는 것을 의미합니다.
- 간단하게는 실제 이미지와 가짜 이미지 , 실제 이미지와 가짜 이미지의 세트 간의 similarity를 보는 방법이라고 볼 수 있습니다.

- **Ablation and Applications**

  - Generated images as regularization (Ours w/ Gen): Similar performance on target concept, **signs of overfitting** (Table 3).
  - Without regularization dataset (Ours w/o Reg): Slightly lower image-alignment, tendency to forget existing concepts (Table 3).
  - Without data augmentation (Ours w/o Aug): Lower visual similarity with target images (Table 3).
  - Fine-tuning on a style: Works with specific styles, e.g., "A painting in the style of V* art."
  - Image editing with fine-tuned models: Compatible with existing image-editing methods, e.g., Prompt-to-prompt.

Method	Text-alignment	Image-alignment	KID (validation)
Ours	0.795	**0.775**	**20.96**
Ours w/o Aug	**0.800**	0.736	20.67
Ours w/o Reg	0.799	0.756	32.64
Ours w/ Gen	0.791	0.768	34.70

Table 3. **Ablation Study.** No augmentation during training leads to lower image-alignment. No regularization dataset or using generated images as regularization produces much worse KID.

Experiments

- **Ablation and Applications**

  - Fine-tuning on a style: Works with specific styles, e.g., "A painting in the style of V∗ art."
  - Image editing with fine-tuned models: Compatible with existing image-editing methods, e.g., Prompt-to-prompt.

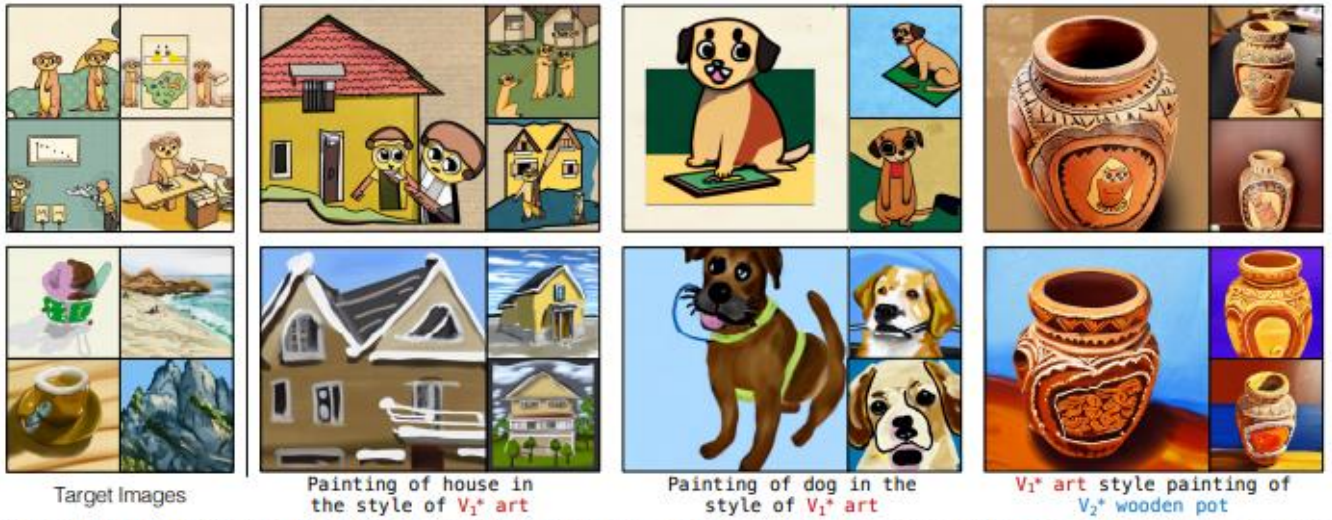


Figure 9. **Custom Diffusion with artistic styles.** Our method can also be used to learn new styles. We finetune our model on 13 and 15 images for the top and bottom rows, respectively. The last column shows the composition of style with the new "wooden pot" concept (target images shown in Figure 7) using the optimization-based method. Style image credits: Mia Tang (top row), Aaron Hertzmann (bottom row).

Figure 10. **Image editing with our models using Prompt-to-Prompt [25].** *Left:* replacement edit to only change the cap. *Right:* image stylization while preserving the image content.

Discussion and Limitations

In conclusion, this study presents:

- A method for fine-tuning large-scale text-to-image diffusion models on new concepts, categories, personal objects, or artistic styles with few image examples.
- A computationally efficient approach that generates novel variations of fine-tuned concepts while preserving visual similarity with target images.
- The need to save only a small subset of model weights.
- The ability to coherently compose multiple new concepts in the same scene.

However, challenges remain:

- Difficult compositions (e.g., a pet dog and a pet cat) are still problematic.
- **Composing three or more concepts together is challenging.**

Figure 11. **Failure cases on multi-concept fine-tuning.** Our method fails at difficult compositions like a cat and dog together in a scene or similar category objects like teddybear and tortoise plushy as shown on the right. Though as shown on the left, the pretrained model also struggles with similar compositions.

Appendix

- **Ablation and Applications**

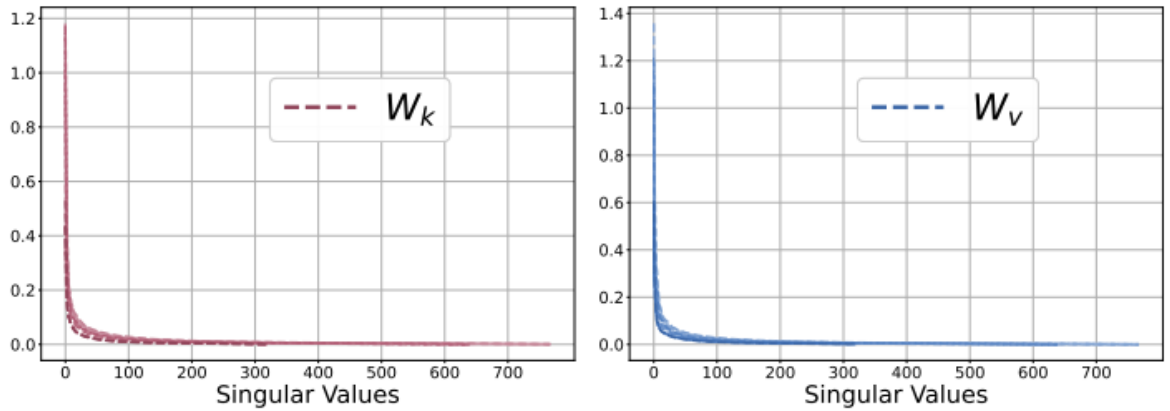


Figure 12. **Analysis of singular values** of the difference of the key and value projection matrices in the cross-attention layers between pretrained and fine-tuned model. As shown in the plot, the singular values drop to 0, suggesting that we can approximate the difference matrix with a low-rank matrix.

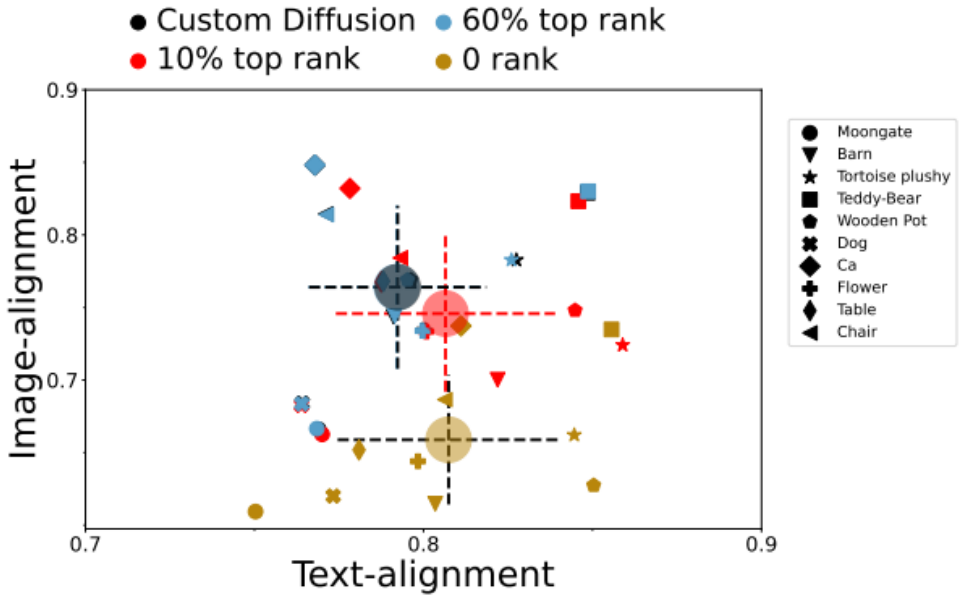


Figure 13. **Quantitative results with compressed models**. We save the low-rank approximation of the difference between the pretrained and fine-tuned model updated weights. Even with the top 60% rank (5× compression), the performance remains similar (overlapping blue and black points). As we increase the compression, the image-alignment score decreases, and the model approaches the pretrained model weights with high text-alignment, as illustrated with qualitative samples in Figure 18.

Appendix

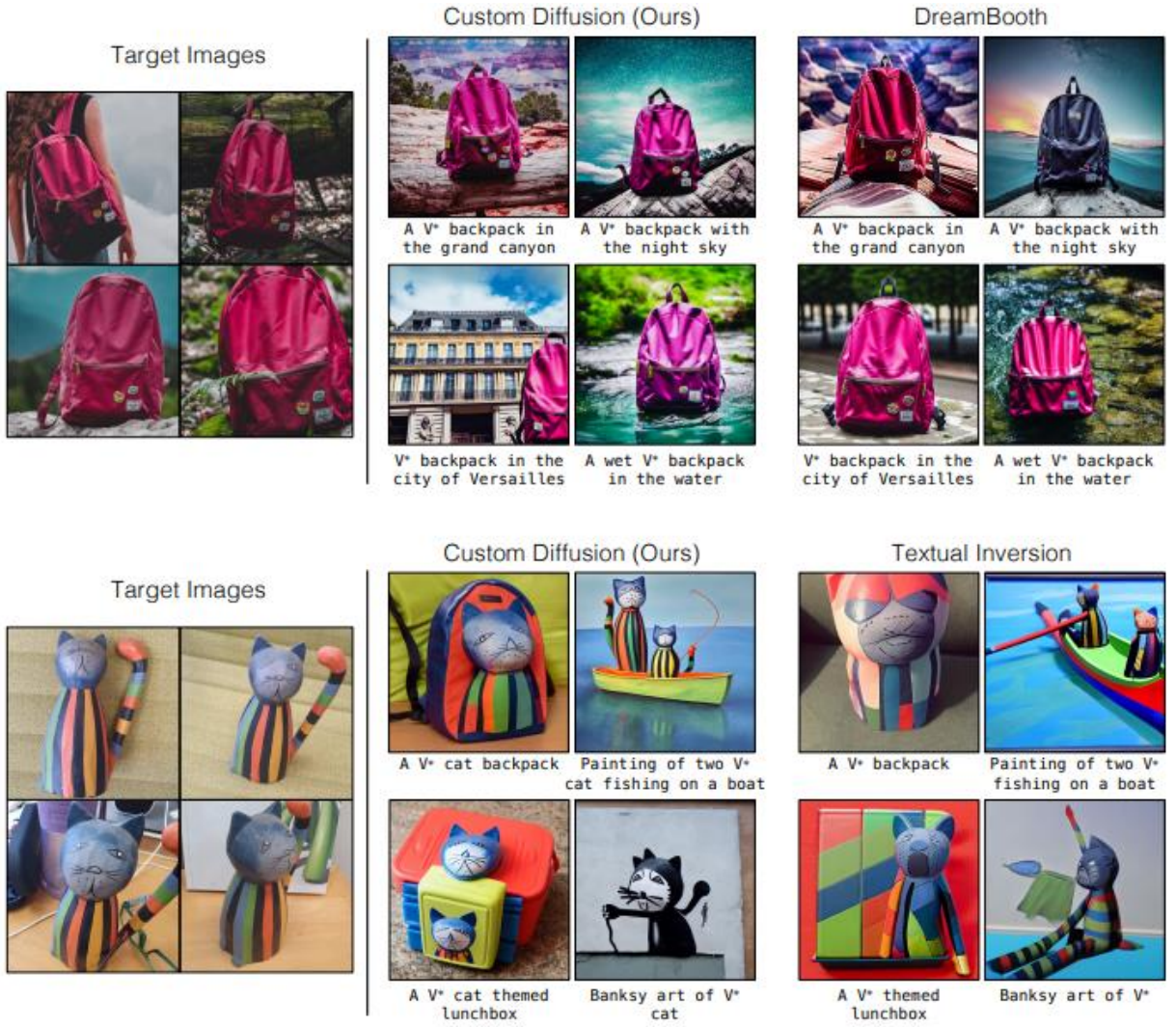


Figure 14. **Results of our method on DreamBooth [62] and Textual Inversion [20] datasets**. Our method works similarly or better in some instances, for example, `V* backpack with the night sky`, and for a `V* cat backpack`. The samples are generated with 200 steps of the DDPM sampler, scale 6.

Appendix

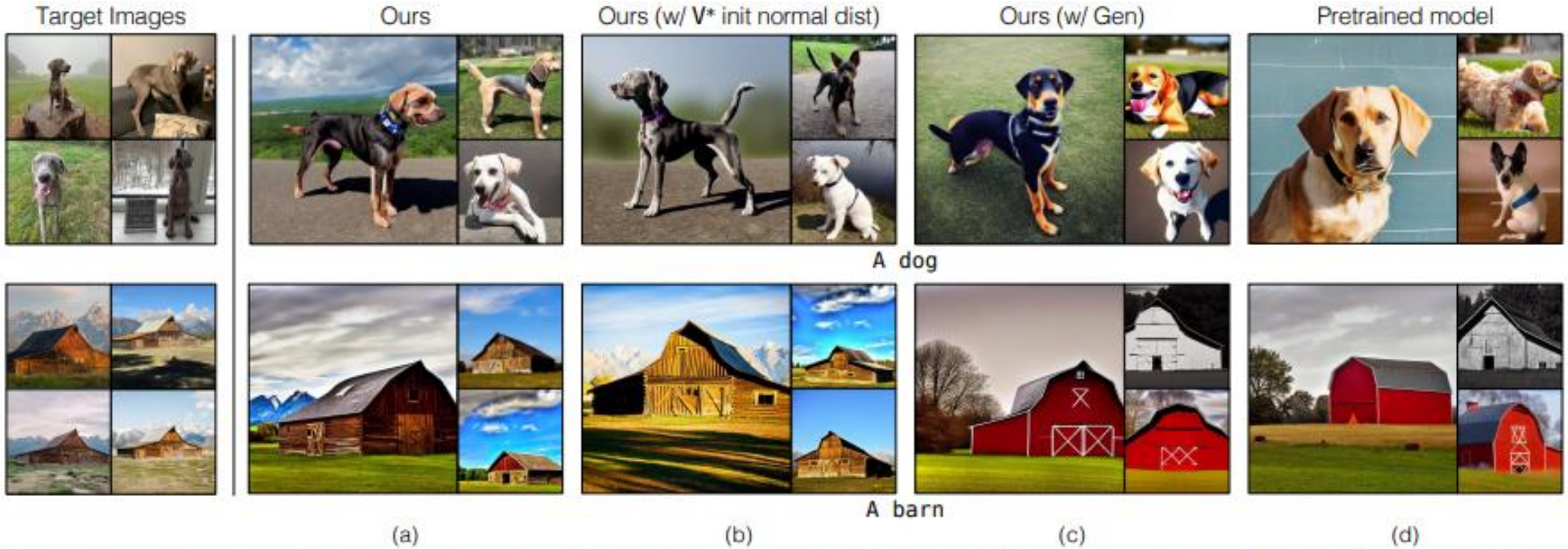


Figure 15. **Qualitative analysis of Choice of V^* (Ours w/ random init V^*) and generated images as regularization (Ours w/ Gen).** We show the generated samples for the original category word, for e.g., images generated on the prompt `a dog` for models fine-tuned on V^* `dog`. We also show the sample generated by pretrained model in column (d) for comparison. Column (b) shows that initializing V^* randomly from the normal distribution of existing token embeddings and then optimizing leads to more shifts in original category words getting mapped to target images compared to our method in column (a). Similarly, with generated images as regularization, the quality of samples gets worse for the original category as shown in column (c).

Appendix

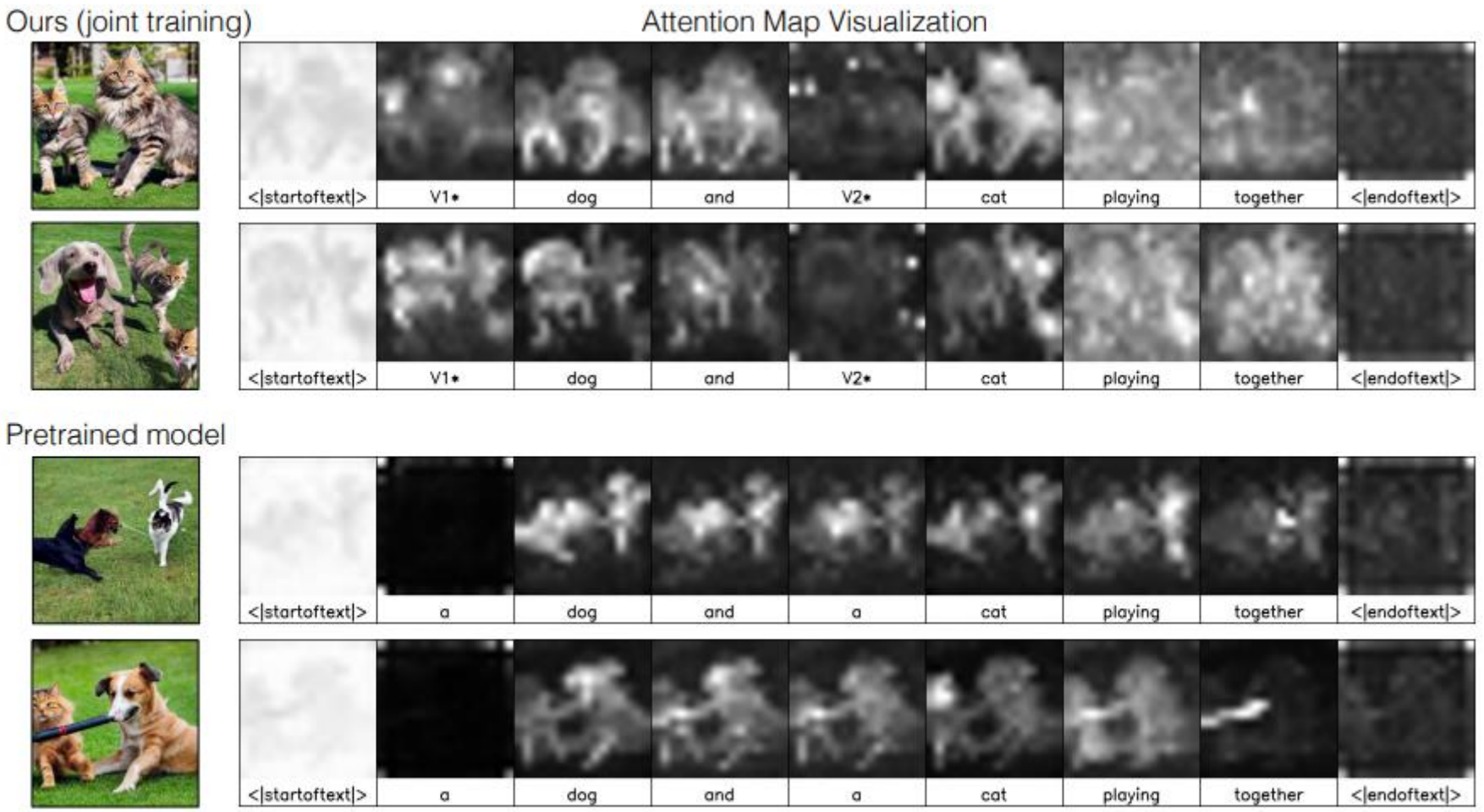


Figure 16. **Attention map visualization of failed compositions.** We show the average attention across timestep and layers for each word (token). For both our and pretrained models, the attention map of "cat" and "dog" overlap more often. This can be one of the reasons for the failed compositions.

Reference

- **LoRA: https://www.youtube.com/watch?v=BJqwmDpa0wM**

Thank you for your Attention….!

Future Works

https://doooob.tistory.com/700