
UNIVERSITÀ DI PISA

# Personalized Diffusion Models

Donald Shenaj

https://donaldssh.github.io

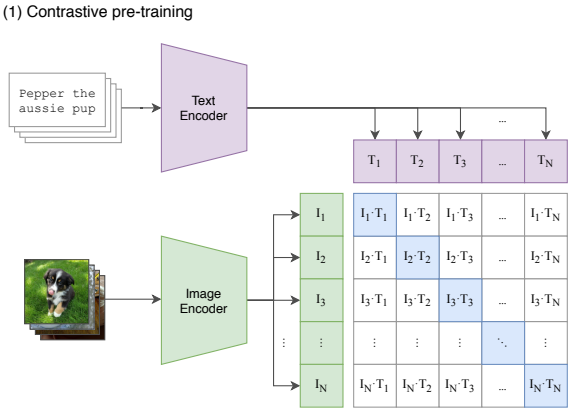
May 21, 2026

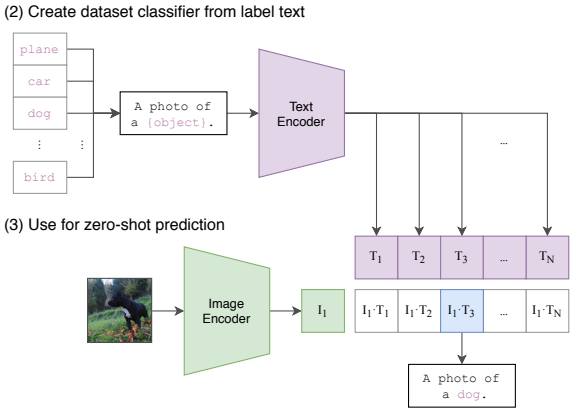
# Lecture Outline

# Section 1

## Diffusion Models Fundamentals

# CLIP: Contrastive Language-Image Pretraining



# Why CLIP Matters for Image Generation

**CLIP as a shared semantic space**

- Image and text embeddings live in the *same* vector space
- Diffusion models can use CLIP text embeddings as conditioning signals
- Enables **text-to-image**: the model learns to denoise conditioned on a text embedding

**CLIP in downstream systems**

- **DALL-E 2**: CLIP image prior + decoder
- **Stable Diffusion**: CLIP text encoder as the conditioning backbone
- **SDXL**: dual CLIP encoders (ViT-L + ViT-bigG)
- **CLIP score**: standard metric for text–image alignment

# UNet: Backbone for Semantic Segmentaiton

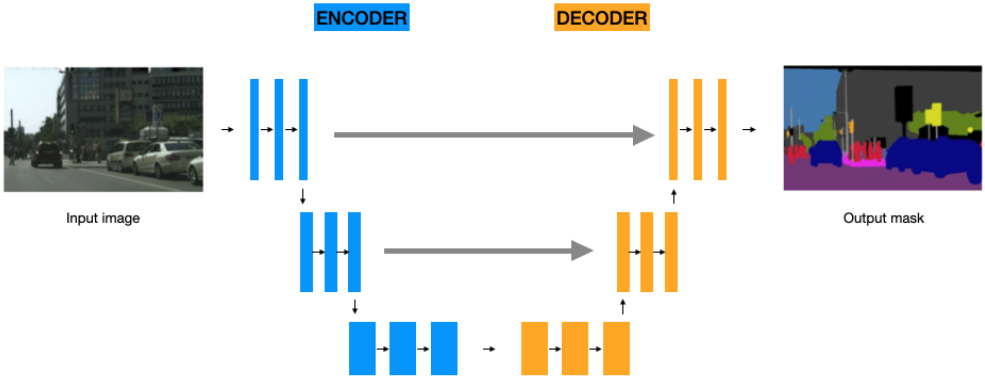
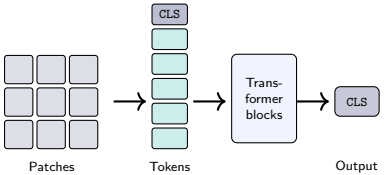



Figure: UNet is a classic encoder-decoder with skip connections

# Vision Transformer (ViT)

**Core idea**

- Split image into fixed-size **patches** (e.g. $16 \times 16$ pixels); flatten each patch into a token
- Prepend a learnable `[CLS]` token; add positional embeddings
- Feed the token sequence through standard **Transformer encoder** blocks (multi-head self-attention + MLP)
- `[CLS]` output used as the global image representation for classification / retrieval




**Why it matters for diffusion models**

- CLIP uses ViT as its image encoder
- DINO is trained with a ViT backbone and exploits its attention maps
- DiT (Flux, SD3) replaces the U-Net with a Transformer over image patch tokens

[*Dosovitskiy et al. 2020 - ViT*]

| Variant | Params | Patch |
|---|---|---|
| ViT-S | 22M | 16 |
| ViT-L | 307M | 16 |
| ViT-bigG (OpenCLIP) | 1.8B | 14 |

# DINO: Emerging Properties in Self-Supervised Vision Transformers

- **Self-distillation with no labels**: a student–teacher framework where both networks share the same architecture; the teacher's weights are an exponential moving average of the student's.
- The student is trained to match the teacher's output distribution via a cross-entropy loss, with a centering and sharpening mechanism to avoid collapse.
- Naturally learns class-specific features, enabling **unsupervised object segmentation** via self-attention maps of the `[CLS]` token.
- Works with Vision Transformers (ViT); attention heads spontaneously focus on semantically meaningful regions without any supervision.

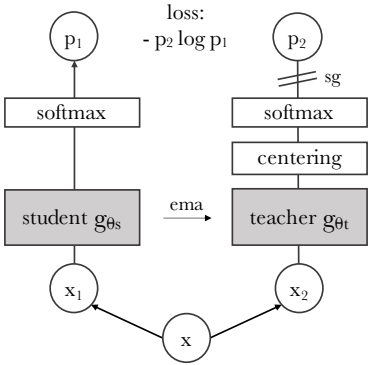


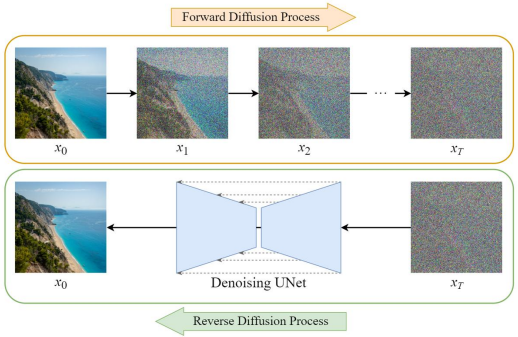

Figure: [*Caron et al., 2021*]

# DINOv2: Learning Robust Visual Features without Supervision


Figure: PCA between the patches of the images from the same column (a, b, c and d) and show their first 3 components. Each component is matched to a different color channel. Same parts are matched between related images despite changes of pose, style or even objects. Background is removed by thresholding the first PCA component

# Diffusion Models

- **Forward process**: gradually corrupt data $\mathbf{x}_0$ by adding Gaussian noise over $T$ steps, yielding a tractable posterior $q(\mathbf{x}_t \mid \mathbf{x}_{t-1})$.
- **Reverse process**: a neural network $\epsilon_\theta$ learns to predict and remove the noise at each step, recovering the original data distribution $p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t)$.
- Training objective: minimise a **simplified MSE** between the true noise and the predicted noise (DDPM, Ho et al., 2020).
- Conditioning (class label, text) is injected via cross-attention or classifier-free guidance, enabling high-quality **conditional generation**.
- Excel at image synthesis (Stable Diffusion, DALL·E 2/3) and have been extended to audio, video, and 3D.

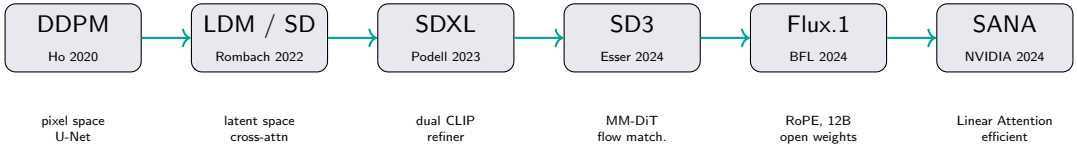



[*Ho et al., 2020*]

# Section 2

Main Diffusion Model Architectures

# Architecture Evolution: A Timeline




**What stayed the same**

- Denoising objective (predict noise / velocity)
- VAE compression of images
- Text conditioning

**What changed**

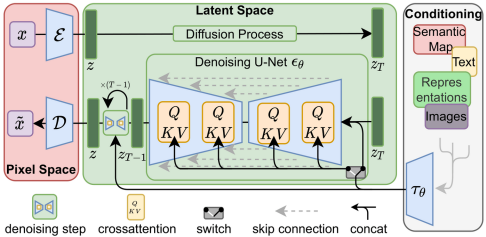
- Backbone: U-Net $\rightarrow$ Transformer (DiT)
- Text integration: cross-attention $\rightarrow$ joint attention
- Sampling: DDPM $\rightarrow$ flow matching

# Latent Diffusion: Stable Diffusion XL (SDXL)

**Key upgrades over SD 1.x / 2.x**

- **Dual text encoders**: OpenCLIP ViT-bigG + CLIP ViT-L; concatenated embeddings give richer conditioning
- **Larger U-Net**: 3× more parameters, transformer blocks in the lowest-resolution layers
- **Micro-conditioning**: original image size and crop coordinates injected as extra tokens (reduces training/inference distribution mismatch)
- **Refiner model**: a separate LDM trained to denoise at high noise levels for detail enhancement

[*Podell et al. 2023 - SDXL*]




**Practical note**

SDXL is the dominant open-source backbone for personalization methods (DreamBooth, ControlNet, LoRA fine-tuning).

# Flux: Diffusion Transformers (DiT)

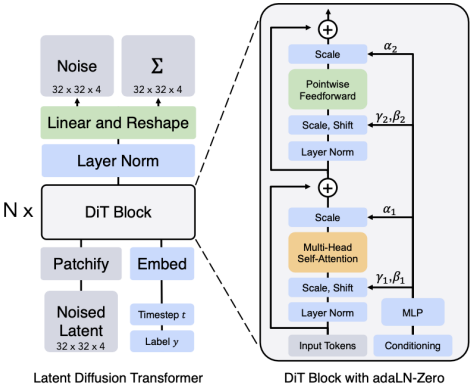
**Architectural shift**: replace the U-Net with a **Diffusion Transformer**

## MM-DiT blocks (Multimodal DiT)

- Image patches *and* text tokens are processed in the **same attention layers**
- Separate weight matrices per modality, but they attend to each other via joint self-attention
- No explicit cross-attention - text and image information flow bidirectionally

**Flux specifics** (Black Forest Labs, 2024)

- Flow matching (instead of DDPM) for faster sampling
- Rotary positional embeddings (RoPE)
- 12B parameters (Flux.1); open weights available



# SANA: Efficient High-Resolution Image Synthesis

**Key ideas**

- **Linear Attention Transformer**: replaces quadratic self-attention with a linear-complexity variant, enabling efficient processing at high resolution ($4096 \times 4096$)
- **Deep compression autoencoder**: $32\times$ spatial compression (vs. $8\times$ in SD), dramatically reducing the latent token count
- **Auto-regression-free**: pure diffusion in the compressed latent space; no AR decoding
- **Efficient scale**: 0.6B and 1.6B parameter variants competitive with much larger models

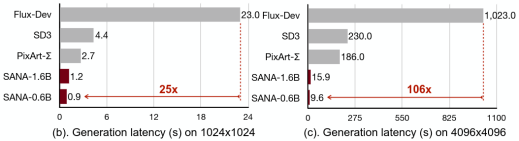



Figure: [*Chen et al. 2024*]

**vs. Flux**

Flux.1 (12B) vs. SANA-1.6B: similar quality, $\sim 8\times$ fewer parameters. Speed and memory matter for personalization fine-tuning.

# Section 3

Personalized Image Generation

# Personalized Image Generation

## Problem statement

Given **3-10 reference images** of a specific subject (person, object, style, ...), generate new images of *that exact subject* in novel contexts, poses, and styles, guided by text prompts.

**Main challenges**

- A pre-trained model has never seen *your* dog
- Generic text tokens ("dog") refer to a class, not an instance
- Need to **inject** new concept identity without destroying general knowledge


Figure: [*Ruiz et al., 2022*]

# Textual Inversion

## Core idea

Find a new pseudo-word embedding $v^*$ in the text encoder's vocabulary space such that the frozen diffusion model generates the target concept when conditioned on the prompt containing $v^*$.

**Training procedure**

1. Freeze *all* model weights (U-Net + text encoder)
2. Initialize $v^*$ randomly (or from a related word)
3. Optimize only $v^*$ via the standard LDM loss:

$$v^* = \arg\min_v \mathbb{E}_{t,\epsilon}\left[\|\epsilon - \epsilon_\theta(\mathbf{z}_t,\, t,\, \tau(y_v))\|^2\right]$$

4. Use: prompt = ‘‘a photo of $S^*$’’ where $S^*$ maps to $v^*$

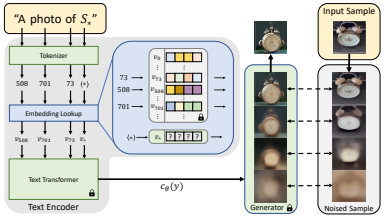



Figure: [*Gal et al. 2022*]

**Properties**

- ✓ Very lightweight (∼768 floats)
- ✓ Easily shareable
- ✗ Limited expressiveness (one token)
- ✗ Slow optimization (2 000–5 000 steps)

# DreamBooth

## Core idea

Fine-tune the *entire* diffusion model (or the U-Net) on 3–5 subject images, binding the subject to a rare identifier token `[V]`.

**Two-loss training**

1. **Reconstruction loss** on subject images: train to reconstruct the specific subject
2. **Prior preservation loss**: provide class images ("a dog") (or generate them) and include them in training to prevent language drift and catastrophic forgetting

$$\mathcal{L} = \mathcal{L}_{\text{subject}} + \lambda \mathcal{L}_{\text{prior}}$$

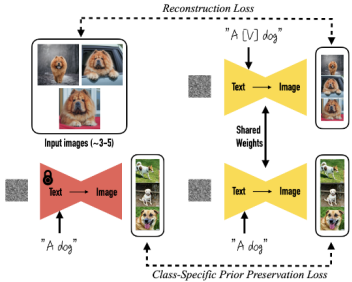



Figure: [*Ruiz et al. 2022* ]

**Properties**

- ✓ High fidelity to subject
- ✓ Flexible prompt composition
- ✗ Full model fine-tune = large file

# Textual Inversion vs. DreamBooth

| | **Textual Inversion** | **DreamBooth** | **DreamBooth + LoRA** |
|---|---|---|---|
| What is trained | Embedding $v^*$ only | Full U-Net | LoRA $\Delta W$ |
| Storage per concept | $\sim$4 KB | $\sim$3 GB | $\sim$50–200 MB |
| Training images | 3–10 | 3–5 | 3–5 |
| Training steps | 2 000–5 000 | 800–1 200 | 500–1 000 |
| Fidelity | Moderate | High | High |
| Editability | High | Moderate | Moderate–High |

## Practical recommendation

DreamBooth + LoRA is today's go-to method: near-DreamBooth fidelity at a fraction of the storage cost. Hundreds of thousands of subject LoRAs are shared on Hugging Face and Civitai.

# Evaluation Metrics for Personalized Generation

**Subject fidelity**

- **DINO similarity**: cosine similarity of ViT-DINO features between generated and reference images (identity preservation)
- **CLIP-I**: image–image CLIP similarity

**Text alignment**

- **CLIP-T**: cosine similarity between generated image and text prompt embedding

**Image quality**

- **FID**: Fréchet Inception Distance (distribution-level quality)
- **IS**: Inception Score
- **Human evaluation**: subject consistency, prompt faithfulness, visual quality (3-axis)

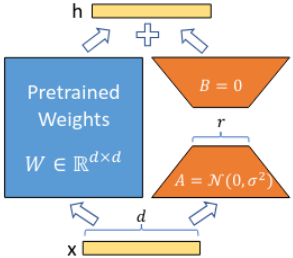
**Trade-off**

Subject fidelity and text editability are **inversely correlated**: more fine-tuning $\Rightarrow$ higher fidelity, lower editability. Good personalization methods optimize the Pareto front.

# LoRA for Personalized Image Generation

**LoRA applied to the U-Net / DiT**

- Freeze $W_0$; add trainable low-rank branch $\Delta W = BA, \quad r \ll d$
- Applied to attention projection matrices $W_Q, W_K, W_V, W_O$
- At inference: $W = W_0 + \alpha\, BA$




**Why LoRA is ideal for personalization**

- Much smaller than full fine-tune
- Composable: multiple LoRAs can be added or merged
- Widely supported (diffusers, AUTOMATIC1111, ComfyUI)

**Standard recipe**

`DreamBooth + LoRA`: run DreamBooth training but only optimize the LoRA parameters. Result: a small, shareable subject adapter.

# How to Select the Optimal Rank?

- A crucial question is often overlooked: which rank should be used?
- Too low, and the model lacks the capacity to capture the subject's unique features. Too high, and you waste memory and degrade prompt alignment.
- This choice is largely driven by community consensus, with little regard for the actual complexity of the subject being personalized.

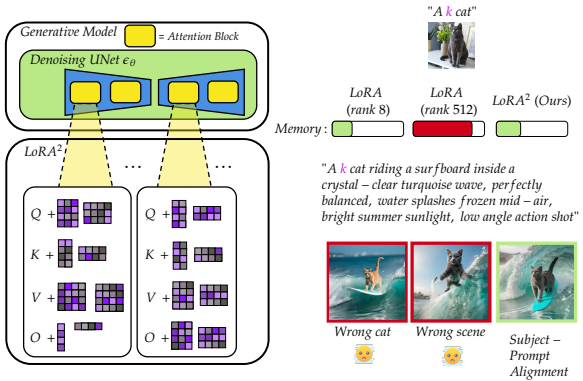



Figure: [*Shenaj et al., 2026*]

# Dynamic and Adaptive Rank Selection

- For each model's component $\ell$, given a frozen weight $W_\ell^* \in \mathbb{R}^{m\times n}$, LoRA updates $\Delta W_\ell \in \mathbb{R}^{m\times n}$, which is computed as two low learnable matrices $B_\ell \in \mathbb{R}^{m\times r}$ and $A_\ell \in \mathbb{R}^{r\times n}$, with rank $r \ll \min(m, n)$:

$$W'_\ell = W_\ell^* + \Delta W_\ell = W_\ell + B_\ell A_\ell.$$

- Inspired by variational methods that learn an adaptive width of neural networks [*Errica et al., 2026*], each LoRA component can be constructed as follows:

$$\Delta W_\ell = B_\ell \Lambda_\ell A_\ell,$$
$$\Lambda_\ell = diag\left(f(1;\nu_\ell), \ldots, f(D_\ell;\nu_\ell)\right)$$

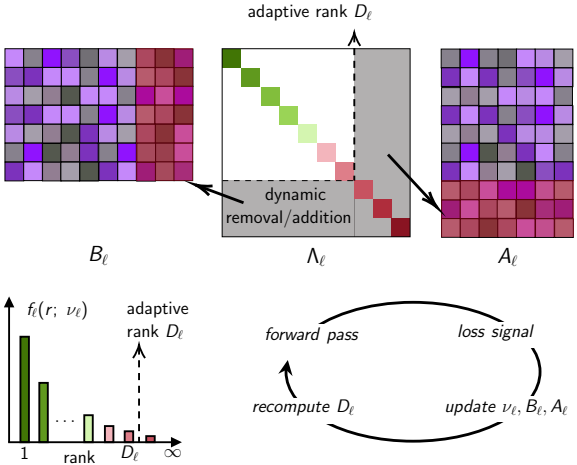
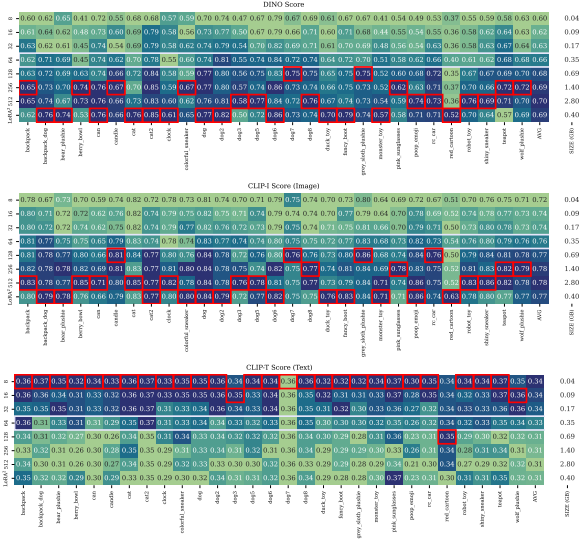



Figure: [*Shenaj et al., 2026*]

# Qualitative Results

| “a k can on a mossy rock in a forest” | “a k can with mountains and mist in the background” | “a k can placed on pink silk fabric” | “a k can on the snow under warm sunlight’ | “a k on a sandy beach at sunset” |
|---|---|---|---|---|
| Rank 8 |  |  |  |  |  |
| Rank 64 |  |  |  |  |  |
| Rank 512 |  |  |  |  |  |
| LoRA[2] |  |  |  |  |  |

# DINO, CLIP-I, CLIP-T per subject scores



# Aggregated Results

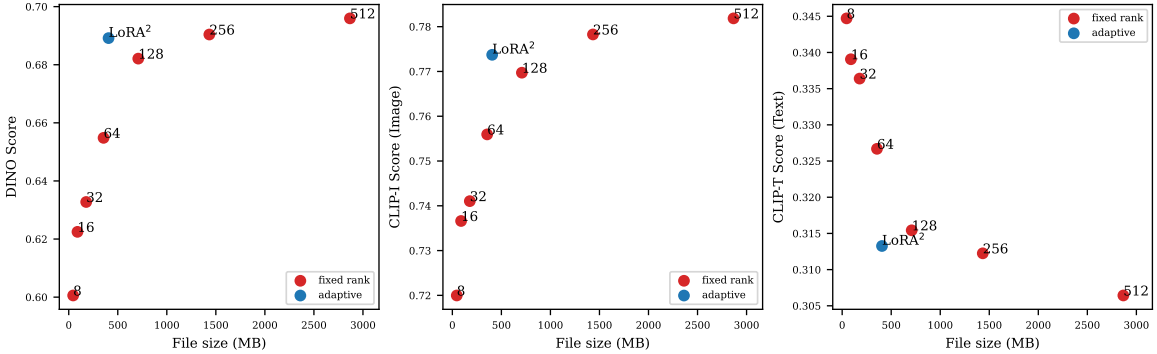
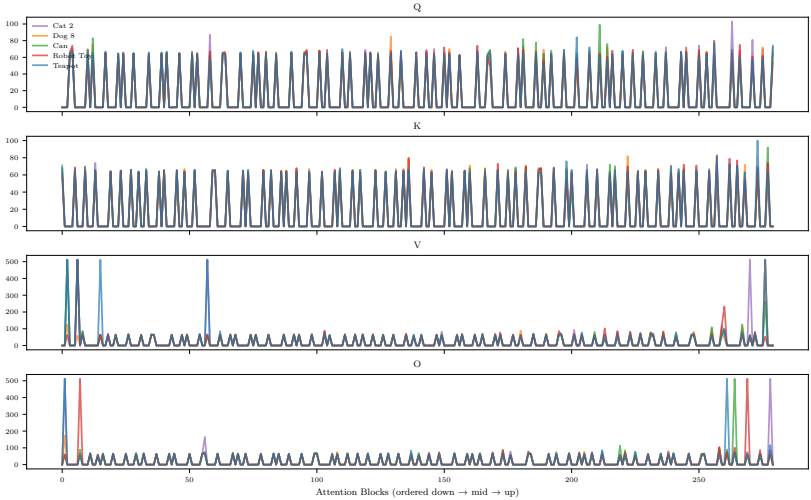
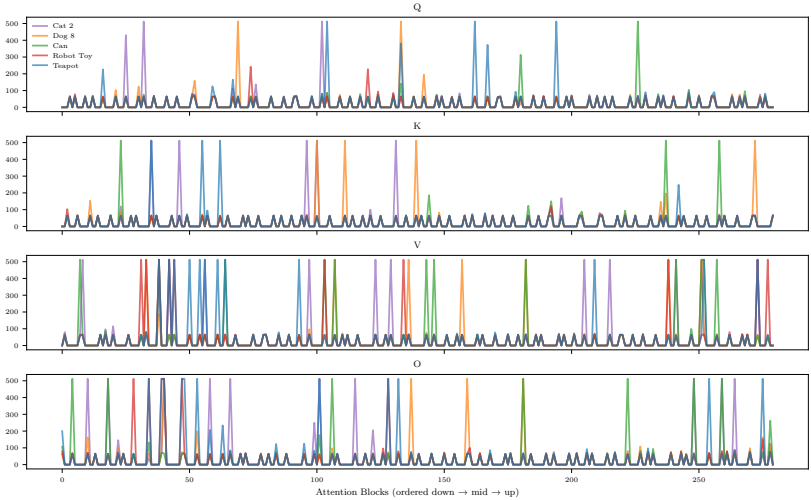



Figure: SDXL backbone. Aggregated results (average of all subjects).

# Self-Attention Analysis



# Cross-Attention Analysis



# Section 4

LoRA Merging for Personalized Image Generation

# Joint Subject-Style Personalized Image Generation

- Style-conditioning such as StyleGAN [*Karras et al., 2019*], StyleDrop [*Sohn et al., 2023*], DreamArtist [*Dong et al., 2022*] can't handle joint subject and style personalization
- Multi-Concept variations of Dreambooth do not work well in practice, and need to train jointly for each combination
- Model Merging:
  - Direct Merging: Arithmetic, DARE, TIES, DARE-TIES
  - Learned Merging: ZipLoRA [*Shah et al., 2024*]

# ZipLoRA: Merging Subject + Style LoRAs

**Motivation**

- Subject LoRA $L_c$ captures identity
- Style LoRA $L_s$ captures artistic appearance
- Naive merging causes interference

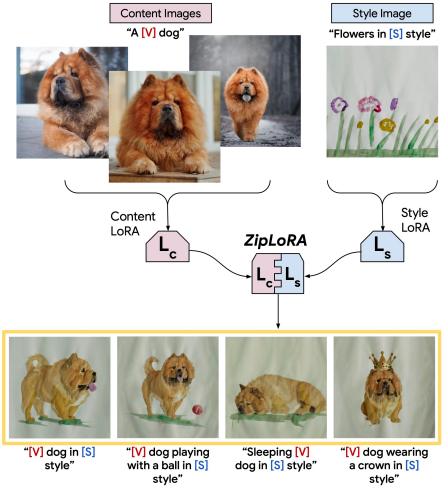
## ZipLoRA

Learn per-parameter merge coefficients $m_c$ and $m_s$:

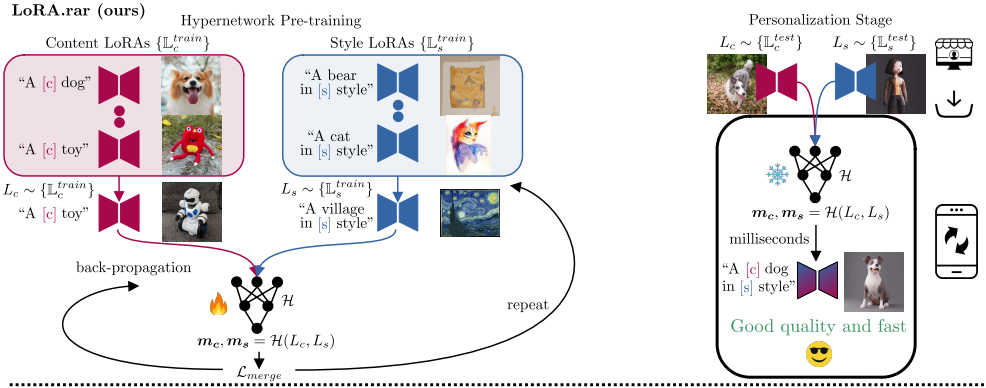
$$
\begin{aligned}
L_m &= \mathrm{Merge}(L_c, L_s, m_c, m_s) \\
\Delta W_m &= m_c \otimes \Delta W_c + m_s \otimes \Delta W_s
\end{aligned} \tag{1}
$$

Optimize:

$$
\begin{aligned}
\mathcal{L}_{\text{merge}} = &\|(D \oplus L_m)(x_c, p_c) - (D \oplus L_c)(x_c, p_c)\|_2 \\
&+ \|(D \oplus L_m)(x_s, p_s) - (D \oplus L_s)(x_s, p_s)\|_2 \\
&+ \lambda \sum_i |m_c^{(i)} \cdot m_s^{(i)}|
\end{aligned} \tag{2}
$$

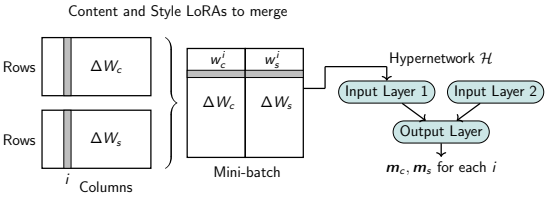


# LoRA.rar: Learning to Merge LoRAs via Hypernetworks



# Hypernetwork Training

- Multiple input layers, each matching the dimensionality of corresponding layers in the generative model
- Content and style LoRAs are concatenated and input into the hypernetwork
- It predicts the columnwise merging coefficients for each specified layer



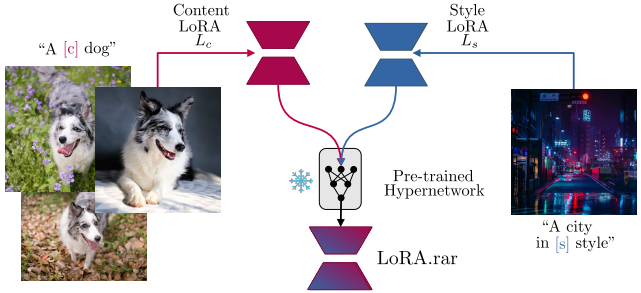

**Require:** # training steps $T$, learning rate $\eta$, base model $\mathcal{D}$, training dataset of content and style LoRAs $\{\mathbb{L}_c^{train}\}$, $\{\mathbb{L}_s^{train}\}$

1: **Initialize** hypernetwork $\mathcal{H}$
2: **for** $t = 1, \ldots, T$ **do**
3: Sample content and style LoRAs from the training set: $L_c \sim \{\mathbb{L}_c^{train}\}$, $L_s \sim \{\mathbb{L}_s^{train}\}$
4: Predict merging coefficients $\boldsymbol{m_c}, \boldsymbol{m_s} = \mathcal{H}(L_c, L_s)$
5: Obtain merged LoRA $L_m$ with weight update matrix $\Delta W_m = \boldsymbol{m_c} \otimes \Delta W_c + \boldsymbol{m_s} \otimes \Delta W_s$.
6: Compute $\mathcal{L}_{merge}$:

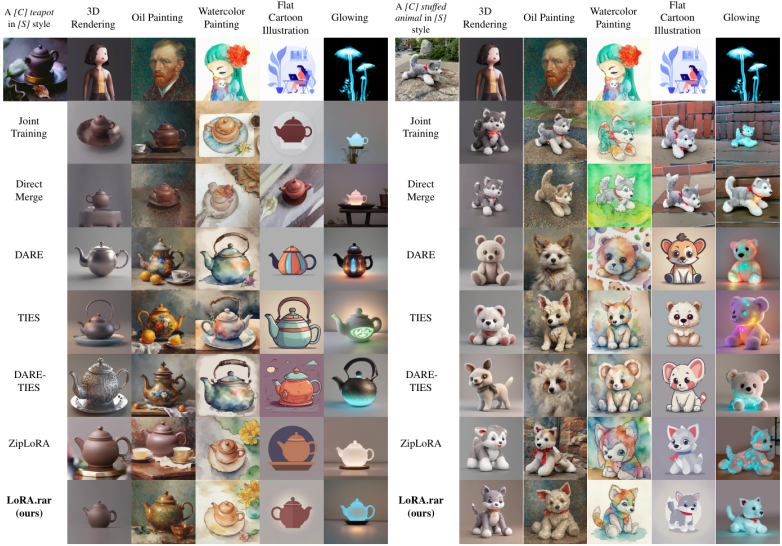
$$\begin{aligned}\mathcal{L}_{merge} =& \|(\mathcal{D} \oplus L_m)(\boldsymbol{x_c}, \boldsymbol{p_c}) - (\mathcal{D} \oplus L_c)(\boldsymbol{x_c}, \boldsymbol{p_c})\|_2 \\ &+ \|(\mathcal{D} \oplus L_m)(\boldsymbol{x_s}, \boldsymbol{p_s}) - (\mathcal{D} \oplus L_s)(\boldsymbol{x_s}, \boldsymbol{p_s})\|_2 \\ &+ \lambda |\boldsymbol{m_c} \cdot \boldsymbol{m_s}|,\end{aligned}$$

7: Update $\mathcal{H} \leftarrow \mathcal{H} - \eta \nabla_{\mathcal{H}} \mathcal{L}_{merge}$
8: **end for**

# Personalization Stage



# Qualitative Comparison



# Overcoming DINO/CLIP-I/CLIP-T Limits with MLLM as a Judge

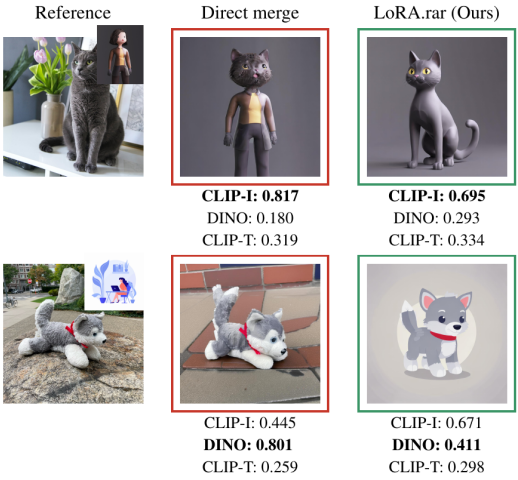



Figure: Limitations of DINO/CLIP-I/CLIP-T

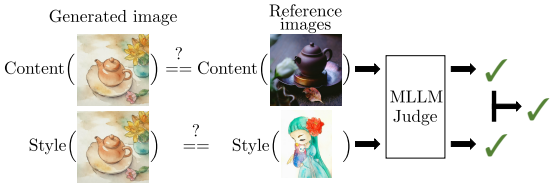



Figure: Evaluation via MLLM Judge

# Quantitative Comparison

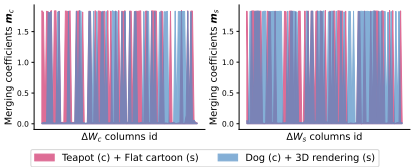



Figure: ZipLoRA Merging Coefficients

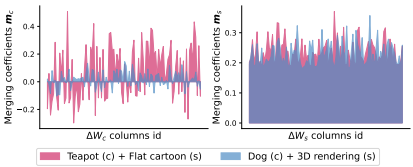



Figure: LoRA.rar Merging Coefficients

| | Average case | Best case |
|---|---|---|
| Joint Training [2] | 0.53 | 0.84 |
| Direct Merge [3] | 0.40 | 0.76 |
| DARE [4] | 0.34 | 0.72 |
| TIES [5] | 0.43 | 0.80 |
| DARE-TIES [6] | 0.30 | 0.60 |
| ZipLoRA [7] | 0.58 | **1.00** |
| **LoRA.rar** | **0.71** | **1.00** |

Table: MLLM Evaluation.

| | ZipLoRA | **LoRA.rar** |
|---|---|---|
| Time to predict merging coeffs | 158s | **0.037s** |
| # Parameters | 1.5M† | **0.49M** |
| # Attempts to *good** image | 2.55 | **2.28** |
| Extra memory at test time | 4GB | **0GB** |

Table: Footprint Analysis.

# Section 5

Personalized Diffusion Models in Practice

# Hands-On: DreamBooth + LoRA with `diffusers`

**Goal**: fine-tune SDXL on 5 photos of a subject using DreamBooth + LoRA and generate novel compositions.

**Notebook outline**

1. **Setup**: install `diffusers`, `accelerate`, `peft`
2. **Data**: load 5 subject images; generate class images for prior preservation
3. **Training**: `train_dreambooth_lora_sdxl.py` (rank 16, $\sim$500 steps on a T4)
4. **Inference**: load LoRA with `load_lora_weights()`
5. **LoRA merging**: add a style LoRA via `set_adapters()` + weights

**Key code snippets**

```
# Load fine-tuned LoRA
pipe.load_lora_weights(
  "my_subject_lora",
  adapter_name="subject"
)
# Add style LoRA
pipe.load_lora_weights(
  "watercolor_style",
  adapter_name="style"
)
# Merge with weights
pipe.set_adapters(
  ["subject","style"],
  [0.8, 0.5]
)
# Generate
image = pipe(
  "[V] dog in watercolor",
  ...
).images[0]
```

# Notebook: What You Will Observe

**Experiment 1 - CLIP and DINO metrics debug**

- Feed images of the same class, and check the various metrics
- You can implement your own version or use the implmented methods

**Experiment 2 - Personalized Subject/Style Training**

- Train for 200, 500, 800 steps
- Measure DINO similarity and CLIP-T
- Observe overfitting regime: fidelity $\uparrow$, editability $\downarrow$

**Experiment 3 - Personalized Subject/Style Inference**

- Compare rank different LoRAs
- Compare different $r \in \{16, 32, 64\}$

**Experiment 4 - LoRA merging**

- Merge subject LoRA + style LoRA at various weights $(\lambda_S, \lambda_A)$
- Visual trade-off between subject and style

**Runtime note**

On a free Colab T4 (15 GB): 1000-step DreamBooth+LoRA training $\approx$ 10/15 min. Inference $\approx$ 1 min

# Open Problems & Research Directions

1. **Evaluation**: how to improve current evaluation pipelines for multi-subject and more complex settings
2. **Video personalization**: extend subject consistency across frames; temporal coherence is an open challenge.
3. **Multi-concept composition**: generate multiple distinct subjects in the same scene; concept bleeding remains a key limitation.
4. **Personalization for DiT / Flux**: most methods assume U-Net; adapting to MM-DiT joint attention is still being explored.
5. **Scaling LoRA merging**: can model merging ideas scale to personalization of 100+ concepts simultaneously?
6. **Personalized LoRA Routing**: given a library of LoRAs, which to select and/or merge to match better my prompt?

# Key References I


- Radford et al. (2021). Learning Transferable Visual Models From Natural Language Supervision (CLIP).
- Caron et al. (2021). Emerging Properties in Self-Supervised Vision Transformers.
- Dosovitskiy et al. (2021). An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. ICLR 2021
- Ho et al. (2020). Denoising Diffusion Probabilistic Models (DDPM).
- Rombach et al. (2022). High-Resolution Image Synthesis with Latent Diffusion Models (LDM / Stable Diffusion).
- Podell et al. (2023). Improving Latent Diffusion Models for High-Resolution Image Synthesis. (SDXL)
- Esser et al. (2024). Scaling Rectified Flow Transformers for High-Resolution Image Synthesis (SD3 / MM-DiT).
- Black Forest Labs (2024). Flux.1 Technical Report.
- Gal et al. (2022). An Image is Worth One Word: Personalizing Text-to-Image Generation using Textual Inversion.
- Ruiz et al. (2022). DreamBooth: Fine Tuning Text-to-Image Diffusion Models for Subject-Driven Generation.
- Hu et al. (2022). LoRA: Low-Rank Adaptation of Large Language Models.
- Zhang & Agrawala (2023). Adding Conditional Control to Text-to-Image Diffusion Models (ControlNet).
- Ye et al. (2023). IP-Adapter: Text Compatible Image Prompt Adapter for Text-to-Image Diffusion Models.
- Wang et al. (2024). InstantID: Zero-shot Identity-Preserving Generation in Seconds.
- Yadav et al. (2023). Resolving Interference When Merging Models (TIES).
- Yu et al. (2024). Language Models are Super Mario (DARE).
- Shah et al. (2024). ZipLoRA: Any Subject in Any Style by Effectively Merging LoRAs. ECCV 2024.
- Shenaj et al. (2025). LoRA.rar: Learning to Merge LoRAs via Hypernetworks for Subject-Style Conditioned Image Generation. ICCV 2025.
- Shenaj et al. (2026). Not All Layers Are Created Equal: Adaptive LoRA Ranks for Personalized Image Generation.

# Any questions?


1

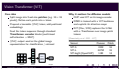

2


3


4


5


6


7


8

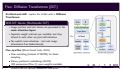

9


10


11


12


13


14

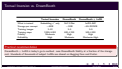

15


16


17


18

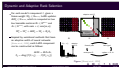

19


20


21

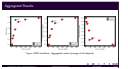

22


23

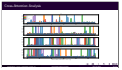

24


25


26


27


28


29


30


31


32


33


34


35


36


37


38


39


40


41


42


43