# A Overview

Our contributions can be summarized as follows:

- **Content and Style Latent Space (FDiff):**
  We propose a Diffusion training framework FDiff for obtaining the disentangled content and style latent spaces that correspond to different semantic factors of an image. This lets us control these factors separately to perform reference-based image translation as well as controllable generation and image manipulation.
- **Generalized Composable Diffusion Model (GCDM):**
  We extend Composable Diffusion Models (CDM) by breaking the conditional independence assumption to allow for dependency between conditioning inputs. This results in better generations in terms of realism and extended controllability.
- **Timestep Scheduling:**
  We leverage the inductive bias of diffusion models and propose timestep-dependent weight schedules to compose information from content and style latent codes for better translation.

Here is a list of contents in Supplementary material.

1. We provide a full derivation of GCDM formulation in Section B.
2. We next show a derivation that the GCDM PDF $\tilde{p}$ is proportional to a nested geometric average of different conditional distributions in Section C.
3. Preliminaries on Diffusion Models are provided in Section D.
4. We explain the training details in Section E.
5. We visualize the learned style and content space in Section F; PCA results are provided in the main paper. Here, we additionally provide latent interpolation and KNN results.
6. We also show additional experiment results on timestep scheduling strategy in Section G.
7. We finally provide some additional results such as text2image synthesis and reference-based image translation in Section H.

# B Derivation for Definition 1

## B.1 Classifier-free Guidance [7]

Assuming we have a single condition $c$, CFG formulation can be derived by:

$$\nabla_{x_t} \log p(x_t|c) = \nabla_{x_t} \log p(x_t, c) \tag{1}$$
$$\nabla_{x_t} \log p(x_t, c) = \nabla_{x_t} \log p(x_t) p(c|x_t) \tag{2}$$
$$= \nabla_{x_t} \log p(x_t) \frac{p(x_t|c)}{p(x_t)} \tag{3}$$
$$= \nabla_{x_t} \log p(x_t) + (\nabla_{x_t} \log p(x_t|c) - \nabla_{x_t} \log p(x_t)) \tag{4}$$
$$= \epsilon(x_t, t) + (\epsilon(x_t, t, c) - \epsilon(x_t, t)) . \tag{5}$$

In practice, $\epsilon(x_t, t) + \alpha (\epsilon(x_t, t, c) - \epsilon(x_t, t))$ is used where $\alpha$ is a temperature controlling the condition effect. Please note that we use $c$ to represent a given single condition.

## B.2 Composable Diffusion Models [11]

$$\nabla_{x_t} \log p(x_t|z_c, z_s) = \nabla_{x_t} \log p(x_t, z_c, z_s) \tag{6}$$
$$\nabla_{x_t} \log p(x_t, z_c, z_s) = \nabla_{x_t} \log p(x_t) p(z_c, z_s|x_t), \quad \text{assuming } z_c \perp\!\!\!\perp z_s | x \tag{7}$$
$$= \nabla_{x_t} \log p(x_t) p(z_c|x_t) p(z_s|x_t) \tag{8}$$
$$= \nabla_{x_t} \log p(x_t) \frac{p(x_t|z_c)}{p(x_t)} \frac{p(x_t|z_s)}{p(x_t)} \tag{9}$$
$$= \nabla_{x_t} \log p(x_t) + \sum_{i=\{c,s\}} (\nabla_{x_t} \log p(x_t|z_i) - \nabla_{x_t} \log p(x_t)) \tag{10}$$
$$= \epsilon(x_t, t) + \sum_{i=\{c,s\}} (\epsilon(x_t, t, z_i) - \epsilon(x_t, t)) \tag{11}$$

Similar to the Classifier-free Guidance, hyperparameters for controlling the weight of each condition are used, i.e., $\epsilon(x_t, t) + \sum_{i=\{c,s\}} \alpha_i (\epsilon(x_t, t, z_i) - \epsilon(x_t, t))$.

Now we introduce how to derive the components of GCDM formulation.

### B.3 Generalized Composable Diffusion Models

For brevity purposes, we omit the term that is canceled out because it is constant w.r.t. $x_t$, e.g., $\nabla_{x_t} \log p(z_c, z_s) = 0$ and $\nabla_{x_t} \log p(z_c) = 0$.

$$\nabla_{x_t} \log p(x_t|z_c, z_s) = \nabla_{x_t} \log p(x_t, z_c, z_s) \tag{12}$$
$$\nabla_{x_t} \log p(x_t, z_c, z_s) = \nabla_{x_t} \log p(x_t) p(z_c, z_s|x_t), \text{NOT assuming } z_c \perp\!\!\!\perp z_s|x \tag{13}$$
$$= \nabla_{x_t} \log p(x_t) p(z_c|z_s, x_t) p(z_s|x_t) \tag{14}$$
$$= \nabla_{x_t} \log p(x_t) p(z_s|x_t) \left( \frac{p(z_s|z_c, x_t) p(z_c|x_t)}{p(z_s|x_t)} \right) \tag{15}$$
$$= \nabla_{x_t} \log p(x_t) p(z_s|x_t) p(z_c|x_t) \left( \frac{p(z_s|z_c, x_t)}{p(z_s|x_t)} \right) \tag{16}$$
$$= \nabla_{x_t} \log p(x_t) p(z_s|x_t) p(z_c|x_t) \left( \frac{p(z_c, z_s|x_t)}{p(z_c|x_t) p(z_s|x_t)} \right) \tag{17}$$
$$= \nabla_{x_t} \log \frac{p(x_t|z_s) p(x_t|z_c)}{p(x_t)} \left( \frac{\frac{p(x_t|z_c, z_s)}{p(x_t)}}{\frac{p(x_t|z_c) p(x_t|z_s)}{p(x_t)^2}} \right) \tag{18}$$
$$= \nabla_{x_t} \log \frac{p(x_t|z_s) p(x_t|z_c)}{p(x_t)} \left( \frac{p(x_t|z_c, z_s) p(x_t)}{p(x_t|z_c) p(x_t|z_s)} \right) \tag{19}$$

$$= -\nabla_{x_t} \log p(x_t) + \nabla_{x_t} \log p(x_t|z_s) + \nabla_{x_t} \log p(x_t|z_c) \tag{20}$$
$$+ \nabla_{x_t} \log p(x_t|z_c, z_s) + \nabla_{x_t} \log p(x_t) - (\nabla_{x_t} \log p(x_t|z_c) + \nabla_{x_t} \log p(x_t|z_s))$$
$$= -\epsilon(x_t, t) + \epsilon(x_t, t, z_s) + \epsilon(x_t, t, z_c)$$
$$+ \epsilon(x_t, t, z_c, z_s) + \epsilon(x_t, t) - (\epsilon(x_t, t, z_c) + \epsilon(x_t, t, z_s)) \tag{21}$$

By rearranging the terms in Eq. (21) and adding hyperparameters $\alpha$, $\lambda$ and $\{\beta_c, \beta_s\}$, the proposed GCDM method in Definition 1 in the main paper can be obtained.

***Clarification of Eq. (17) and Eq. (18).*** By Bayes Theorem, Eq. (17) becomes

$$\nabla_{x_t} \log \left[ \underbrace{p(x_t) \frac{p(x_t|z_s) p(z_s)}{p(x_t)} \frac{p(x_t|z_c) p(z_c)}{p(x_t)}}_{①} \underbrace{\left( \frac{\frac{p(x_t|z_c, z_s) p(z_c, z_s)}{p(x_t)}}{\frac{p(x_t|z_c) p(z_c) p(x_t|z_s) p(z_s)}{p(x_t)^2}} \right)}_{②} \right].$$

By rearranging ① and ② separately, the above equation becomes

$$= \nabla_{x_t} \log \left[ \underbrace{p(z_s)p(z_c)\frac{p(x_t|z_s)p(x_t|z_c)}{p(x_t)}}_{\text{rearranged from } ①} \underbrace{\left( \frac{\frac{p(x_t|z_c,z_s)}{p(x_t)}}{\frac{p(x_t|z_c)p(x_t|z_s)}{p(x_t)^2}} \right) \left( \frac{p(z_c,z_s)}{p(z_c)p(z_s)} \right)}_{\text{rearranged from } ②} \right].$$

By canceling out $p(z_c)p(z_s)$ in the first and the last term and by rearranging the equation, we can obtain Eq. (18), i.e.,

$$\begin{aligned}
&= \nabla_{x_t} \log \left[ \cancel{p(z_s)p(z_c)} \frac{p(x_t|z_s)p(x_t|z_c)}{p(x_t)} \left( \frac{\frac{p(x_t|z_c,z_s)}{p(x_t)}}{\frac{p(x_t|z_c)p(x_t|z_s)}{p(x_t)^2}} \right) \left( \frac{p(z_c,z_s)}{\cancel{p(z_c)p(z_s)}} \right) \right] \\
&= \nabla_{x_t} \log \left[ \frac{p(x_t|z_s)p(x_t|z_c)}{p(x_t)} \left( \frac{\frac{p(x_t|z_c,z_s)}{p(x_t)}}{\frac{p(x_t|z_c)p(x_t|z_s)}{p(x_t)^2}} \right) p(z_c,z_s) \right] \\
&= \underbrace{\nabla_{x_t} \log \left[ \frac{p(x_t|z_s)p(x_t|z_c)}{p(x_t)} \left( \frac{\frac{p(x_t|z_c,z_s)}{p(x_t)}}{\frac{p(x_t|z_c)p(x_t|z_s)}{p(x_t)^2}} \right) \right]}_{\text{Eq. (18)}} + \cancel{\nabla_{x_t} \log p(z_c,z_s)}^{\,0},
\end{aligned}$$

where $\nabla_{x_t} \log p(z_c, z_s) = 0$ because it is constant w.r.t. $x_t$.

## C Derivation for Corollary 1

The derivation starts from GCDM formulation proposed in Definition 1 in the main paper.

$$\nabla_{x_t} \log \tilde{p}_{\alpha,\lambda,\beta_c,\beta_s}(x_t|z_c,z_s) \triangleq \epsilon(x_t,t) + \alpha\Big[\lambda(\underbrace{\epsilon(x_t,t,z_c,z_s) - \epsilon(x_t,t)}_{\nabla_{x_t}\log p(z_c,z_s|x_t)}) + (1-\lambda)\sum_{i=\{c,s\}} \beta_i\Big(\underbrace{\epsilon(x_t,t,z_i) - \epsilon(x_t,t)}_{\nabla_{x_t}\log p(z_i|x_t)}\Big)\Big]. \tag{22}$$

Given the fact that $\epsilon(x_t,t) = \nabla_{x_t}\log p(x_t)$, taking integral w.r.t. $x_t$ to the equation yields:

$$\log \tilde{p}_{\alpha,\lambda,\beta_c,\beta_s}(x_t|z_c,z_s) = \log p(x_t) + \alpha\Big[\lambda(\log p(x_t|z_c,z_s) - \log p(x_t)) + (1-\lambda)\sum_{i=\{c,s\}} \beta_i\Big(\log p(x_t|z_i) - \log p(x_t)\Big)\Big] + C\,, \tag{23}$$

where $C$ is a constant. Merging all the terms with log:

$$\log \tilde{p}_{\alpha,\lambda,\beta_c,\beta_s}(x_t|z_c,z_s) = \log\exp(C) + \log\Big(p(x_t)\left(\frac{p(x_t|z_c,z_s)}{p(x_t)}\right)^{\alpha\lambda}\left(\frac{p(x_t|z_c)^{\beta_c}p(x_t|z_s)^{\beta_s}}{p(x_t)^{\beta_c+\beta_s}}\right)^{\alpha(1-\lambda)}\Big) \tag{24}$$

Taking exponential to the above equation:

$$\tilde{p}_{\alpha,\lambda,\beta_c,\beta_s}(x_t|z_c,z_s) = \exp(C)p(x_t)\left(\frac{p(x_t|z_c,z_s)}{p(x_t)}\right)^{\alpha\lambda}\left(\frac{p(x_t|z_c)^{\beta_c}p(x_t|z_s)^{\beta_s}}{p(x_t)^{\beta_c+\beta_s}}\right)^{\alpha(1-\lambda)} \tag{25}$$

$$= \exp(C)p(x_t)^{(1-\alpha\lambda-\alpha(1-\lambda)(\beta_c+\beta_s))}p(x_t|z_c,z_s)^{\alpha\lambda}\left(p(x_t|z_c)^{\beta_c}p(x_t|z_s)^{\beta_s}\right)^{\alpha(1-\lambda)}. \tag{26}$$

Given the fact that $\beta_c + \beta_s = 1$,

$$\tilde{p}_{\alpha,\lambda,\beta_c,\beta_s}(x_t|z_c,z_s) = \exp(C)p(x_t)^{(1-\alpha)}\left[p(x_t|z_c,z_s)^{\lambda}\left(p(x_t|z_c)^{\beta_c}p(x_t|z_s)^{(1-\beta_c)}\right)^{(1-\lambda)}\right]^{\alpha}. \tag{27}$$

Since the exponential function is always positive,

$$\tilde{p}_{\alpha,\lambda,\beta_c,\beta_s}(x_t|z_c,z_s) \propto p(x_t)^{(1-\alpha)} \left[ p(x_t|z_c,z_s)^{\lambda} \left( p(x_t|z_c)^{\beta_c} p(x_t|z_s)^{(1-\beta_c)} \right)^{(1-\lambda)} \right]^{\alpha}, \tag{28}$$

which is the same as Corollary 1 in the main paper.

## D Preliminaries on Diffusion Models

Diffusion Models [6, 17] are one class of generative models that map the complex real distribution to the simple known distribution. In high level, DMs aim to train the networks that learn to denoise a given noised image and a timestep $t$. The noised image is obtained by a fixed noising schedule. Diffusion Models [6, 17] are formulated as $p_\theta(x_0)$. The marginal $p_\theta(x_0)$ can be formulated as a marginalization of the joint $p_\theta(x_{0:T})$ over the variables $x_{1:T}$, where $x_1, ...x_T$ are latent variables, and $p(x_T)$ is defined as standard gaussian. Variational bound of negative log likelihood of $p_\theta(x_0)$ can be computed by introducing the posterior distribution $q(x_{1:T}|x_0)$ with the joint $p_\theta(x_{0:T})$. In Diffusion Models [6, 17], the forward process $q(x_{1:T}|x_0)$ is a predefined Markov Chain involving gradual addition of noise sampled from standard Gaussian to an image. Hence, the forward process can be thought of as a fixed noise scheduler with the $t$-th factorized component $q(x_t|x_{t-1})$ represented as: $q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t I)$, where $\beta_t$ is defined manually. On the other hand, the reverse or the generative process $p_\theta(x_{0:T})$ is modelled as a denoising neural network trained to remove noise gradually at each step. The $t$-th factorized component $p_\theta(x_{t-1}|x_t)$ of the reverse process is then defined as, $\mathcal{N}(x_{t-1}|\mu_\theta(x_t,t), \Sigma(x_t,t))$. Assuming that variance is fixed, the objective of Diffusion Models (estimating $\mu$ and $\epsilon$) can be derived using the variational bound [6, 17] (Refer the original papers for further details).

Following Denoising Diffusion Probabilistic Models [6] (DDPM), Denoising Diffusion Implicit Model [18] (DDIM) was proposed that significantly reduced the sampling time by deriving a non-Markovian diffusion process that generalizes DDPM. The latent space of DDPM and DDIM has the same capacity as the original image making it computationally expensive and memory intensive. Latent Diffusion Models [16] (LDM) used a pretrained autoencoder [3] to reduce the dimension of images to a lower capacity space and trained a diffusion model on the latent space of the autoencoder, reducing time and memory complexity significantly without loss in quality.

All our experiments are based on LDM as the base diffusion model with DDIM for sampling. However the techniques are equivalently applicable to any diffusion model and sampling strategy.

# E Implementation Details

We build our models on top of LDM codebase[1]. For FFHQ and LSUN-church, we train our model for two days with eight V–100 GPUs. The model for AFHQ dataset is trained for one and a half days with the same device. All models are trained for approximately 200000 iterations with a batch size of 32, 4 samples per GPU without gradient accumulation. All models are trained with 256×256 images with a latent $z$ size of 3×64×64. The dimensions of content code $z_c$ is 1×8×8 while that of style code $z_s$ is 512×1×1. $t_1, t_2$ and $t_3$ from Eq. 1 in the main paper are timestep embeddings learned to specialize according to the latent code they are applied for to support learning different behavior for content and style features at different timesteps. We also experimented with different sizes for content and style code and chose these for best empirical performance. The content encoder takes as input $z$ and outputs $z_c$ following a sequence of ResNet blocks. The style encoder has a similar sequence of ResNet blocks followed by a final global average pooling layer to squish the spatial dimensions similar to the semantic encoder in [14].

To support GCDM during sampling, we require the model to be able to generate meaningful scores and model the style, content and joint distributions. Hence, during training we provide only style code, only content code and both style and content code all with probability 0.3 (adding up to 0.9) and no conditioning with probability 0.1 following classifier–free guidance literature. This helps learn the conditional and unconditional models that are required to use the proposed GCDM formulation. The code will be released upon acceptance of the paper.

During sampling, without *reverse DDIM*, if all the joint, conditionals, and unconditional guidance are used, sampling time for a single image is 10 seconds. With *reverse DDIM* to get $x_T$ where T is the final timestep, it takes 22 seconds. This might be lesser if *reverse DDIM* is stopped early and generation happens from the stopped point. Specific hyperparameters used to generate results in the main paper and appendix are provided in Table. 1.

## E.1 Experimental Setup

**Datasets**
We train different models on the commonly used datasets such as AFHQ [2], FFHQ [8] and LSUN-church [19].

**Baselines**

**DiffuseIT:** The most similar work to ours based on diffusion models is DiffuseIT [9] that tackles the same problem formulation. We compare our results with DiffuseIT using their pretrained model and default parameters.

**DiffAE+SDEdit:** Since Diffusion Autoencoder [14] does not directly support image-to-image translation, we combine that with SDEdit [12]. The input image for the reverse process is $x_{600}$ (chosen empirically) obtained as $q(x_{600}|x_c)$

[1] `https://github.com/CompVis/latent-diffusion`

**Table 1:** Hyperparameters used to generate the figures in the main paper and appendix. Timestep scheduling is only used in the sampling process. The parenthesis in the second column indicates the number of steps we used for sampling. Note that $\beta_c = 1 - \beta_s$.

| Dataset | sampler | $x_T$ | $\alpha$ | $\lambda$ | $\beta_s$ | $a$ | $b$ | scheduler |
|---|---|---|---|---|---|---|---|---|
| Main paper | | | | | | | | |
| FFHQ | DDIM+SDEdit (60) | *reverse DDIM* | 1.5 | 0.9 | 1.0 | 0.025 | 550 | sigmoid |
| LSUN-church | DDIM (100) | $q(x_{991}\|x_0)$ | 2.0 | 0.5 | 0.0 | - | - | - |
| AFHQ | DDIM+SDEdit (60) | $q(x_{591}\|x_0)$ | 3.0 | 0.75 | 1.0 | - | - | - |
| Appendix | | | | | | | | |
| FFHQ | DDIM (100) | *reverse DDIM* | 1.5 | 0.9 | 1.0 | 0.025 | 550 | sigmoid |
| LSUN-church | DDIM (100) | *reverse DDIM* | 5.0 | 0.5 | 0.0 | 0.025 | 600 | sigmoid |

by running the forward process on the content image. The semantic feature $z_{sem}$ from the semantic encoder of DiffAE is used given the style image $x_s$.

**DiffAE+MagicMix:** We also combine MagixMix [10] with DiffAE. Similar to DiffAE+SDEdit, this model takes $x_{600}$ from $x_c$ as input and $z_{sem}$ from $x_s$ as conditioning. Additionally, at each timestep, the approximated previous timestep $\hat{x}_{t-1}$ is combined with $x_{t-1}$ from the content image $x_c$, i.e., $\hat{x}_{t-1} = v\hat{x}_{t-1} + (1-v)q(x_{t-1}|x_c)$. For this experiment, $v = 0.5$ is used and the noise mixing technique is applied between $t = [600, 300]$.

**SAE:** Swapping Autoencoder [13] based on GAN [4] is also evaluated. Since the available pretrained model is on a resolution of 512, we resize the generated results to 256 for a fair comparison.

**DEADiff:** The sampling method of DEADiff [15] only uses the style representation (Fig. 2 in their paper). To see the performance of both style and content, we give both representations during the inference stage (Note that Fig. 7 in their paper is not from the content representation but from depth ControlNet [20]). We did not finetune the model specifically for facial data since adjusting the facial dataset to their training data settings is infeasible (e.g., two images with the same style but with a different subject, and two images with the same subject but with a different style).

**StarGAN v2:** The training and sampling process of StarGAN v2 [2] requires gender attributes of the image. Since FFHQ does not have attribute labels, we use their pretrained models on CelebA-HQ. When sampling, we preprocess and label the test images by using a pretrained gender classifier (c.f. EasyFace).

The training and sampling process of StarGAN v2 requires gender attributes of the image. Since FFHQ does not have attribute labels, we use their pretrained models on CelebA-HQ. When sampling, we preprocess and label the test images by using a pretrained gender classifier (c.f. EasyFace).

**Evaluation Metrics**

**FID:** We use the commonly used Fréchet inception distance (FID) [5] to ensure the generated samples are realistic. We follow the protocol proposed in [2] for reference-based image translation. To obtain statistics from generated images,

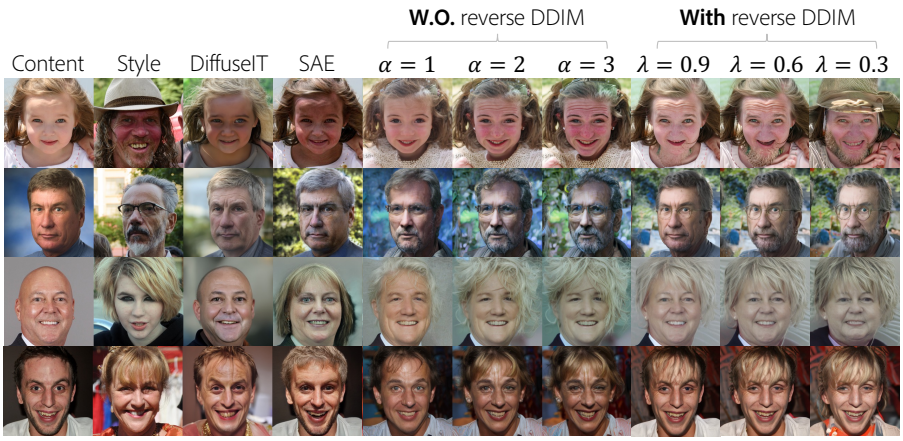


**Fig. 1:** Comparisons between with and without *reverse DDIM* sampling for FFHQ model. We use the proposed GCDM and timestep scheduling during sampling. We can clearly see better identity preservation than not using reverse DDIM, on par with SAE [13] while still providing better controllability.

2000 test samples are used as the content images, and five randomly chosen images from the rest of the test set are used as style images for each content image to generate 10,000 synthetic images.

**LPIPS:** Even though FID evaluates the realism of the generations, the model could use just content and ignore style (or vice versa) and still get good FID. Following [2], we use LPIPS score obtained by measuring the feature distances between pairs of synthetic images generated from the same content image but with different style images. **Higher LPIPS indicates more diverse results**. Ideally, the models incorporate enough style information from different style images for the same content image (increasing LPIPS) but without going out of the real distribution (decreasing FID).

### E.2 Identity Preservation

We notice that when the proposed sampling technique is used with randomly sampled noise $x_T$ for reference based image translation or manipulation, particularly on FFHQ dataset, that the identity of the content image is not preserved. This is an important aspect of image manipulation for faces. One of the ways to preserve better identity is to use the deteministic *reverse DDIM* process described in [14] to obtain $x_T$ that corresponds to a given content image. To do this, we pass the content image to both the content and style encoders as well as the diffusion model to get $x_T$ that reconstructs the content image. This $x_T$ is then used along with content code from content image and style code from style image to generate identity preserving translation.

**Table 2:** Quantitative comparison between the variants of ours with and without *reverse ddim*.

| | w/o *reverse ddim* | | | w/ *reverse ddim* | | |
|---|---|---|---|---|---|---|
| | Ours($\alpha = 1.0$) | Ours($\alpha = 2.0$) | Ours($\alpha = 3.0$) | Ours($\lambda = 0.9$) | Ours($\lambda = 0.6$) | Ours($\lambda = 0.3$) |
| FID | 20.38 | 23.68 | 26.45 | **11.99** | 13.40 | 15.45 |
| LPIPS | 0.53 | 0.57 | **0.6** | 0.34 | 0.42 | 0.49 |

The comparisons between with and without *reverse DDIM* during sampling are provided in Table 2. We can see that with *reverse DDIM*, we can expect to get more realistic results (lower FID) while the less diverse results (lower LPIPS). Fig. 1 also shows comparisons between with and without *reverse DDIM*. In contrast to SAE [13] that preserves better identity by trading of magnitude of style applied, our approach provides the ability to control the magnitude of identity preservation and style transfer independently.

Additional comparisons between with and without *reverse DDIM* are provided in Fig. 2 and 3. We can see that the results with *reverse DDIM* better preserve the content identity while applying the style reasonably. On the other hand, the results without *reverse DDIM* have a stronger impact on style with lesser identity preservation, which could be preferable in non-face domains such as abstract or artistic images or for semantic mixing.

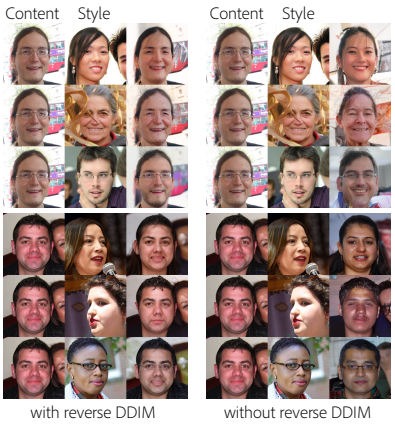


**Fig. 2:** Comparisons between with and without DDIM reverse sampling method in FFHQ dataset.


**Fig. 3:** Comparisons between with and without DDIM reverse sampling method in LSUN-church dataset.

# F Extent of Controllability

In this Section, we present the rich controllability of our proposed framework. The latent space exploration by PCA is presented in the main paper. Interpolation results are shown in Section F.1. Further analysis of the latent space by KNN is reported in Section F.3.

## F.1 Interpolation

We conducted experiments on latent space interpolation in order to analyze the effects of the content, the style, and $x_T$ during the sampling process. All the results use *reverse DDIM* with content image to get $x_T$.

Fig. 4 shows the content-only interpolation results where style feature $z_s$ and noise $x_T$ are fixed to the image in the first column. The gray box on the top indicates the fixed input while $z_c$ is interpolated between the two images in the first two columns. From the figure, we can see that the style information and the person's identity are maintained while pose and facial shape are changed.

Fig. 5 shows the case $x_T$ obtained from *reverse DDIM* of the images in the first two columns is interpolated while the style and content features are fixed to the image in the first column. The content (e.g., pose, facial shape) and the style (e.g., beard, eyeglasses, and facial color) are maintained while stochastic properties change. We can see that identity is not entirely tied to $x_T$ but the stochasticity causes changes in the identity. This could be an evidence of why using *reverse DDIM* to fix $x_T$ preserves better identity.

Fig. 6 visualizes the style interpolation while the content and $x_t$ are fixed to the first image. The person's identity, pose, and facial shape are preserved while the facial expression, gender, and age are smoothly changed validating our results.

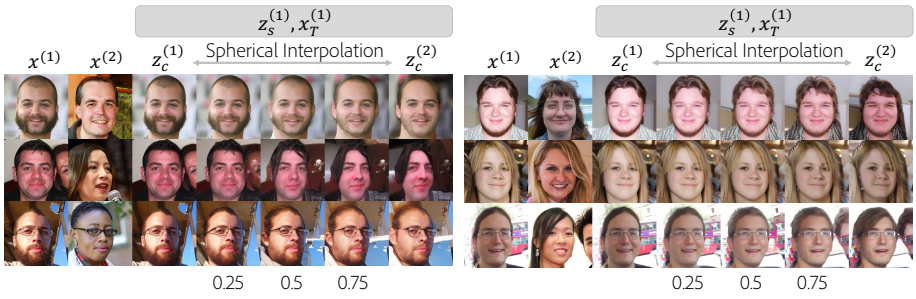


**Fig. 4:** Content interpolation results. Style and $x_T$ are obtained from images in the first column while content code is interpolated between images in column 1 and column 2. We can see how content-specific factors vary smoothly.

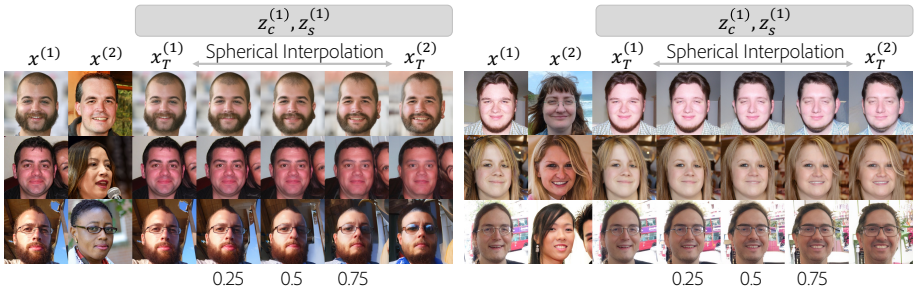


**Fig. 5:** $x_T$ interpolation results where style and content are obtained from images in the first column while $x_T$ is interpolated between reverse DDIM of both images. We can see stochastic changes causing mild identity changes. Fixing $x_T$ to the content image hence provides better identity preservation for image translation and manipulation.

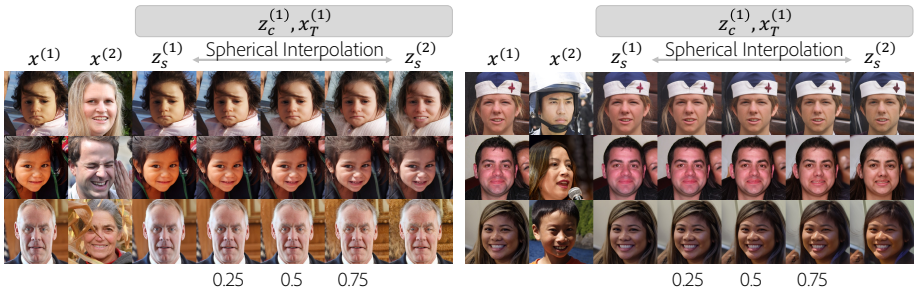


**Fig. 6:** Style interpolation results when content and $x_T$ are obtained from images in the first column. We can see smooth changes in the semantic attributes such as age, gender, smile, eyeglasses, etc. allowing for effective style manipulations.

### F.2 Information Encoded in Each Latent Space

We analyze the role of the denoising network $\epsilon$ and the encoders $E_c$ and $E_s$ by analyzing what information is encoded in the latent spaces. Fig. 7 and Fig. 8 show the results of fixing the content while varying the style images (and vice versa). $x_T$ is fixed as well to reduce the stochasticity. The remaining stochasticity comes from the white noise at each timestep during the reverse process. From the results, we can see that the structure information is maintained while style information changes according to the style image (and vice versa) as we intended.

### F.3 Interpreting the Latent Spaces based on KNN

We additionally perform K Nearest Neighbor (KNN) experiments to understand what features are encoded in the content and style latent representations. We pass 10000 unseen images through the style and the content encoders to get $z_c$ and $z_s$. We then compute the distance of an arbitrary sample with the entire validation set and sort the 10000 distances.

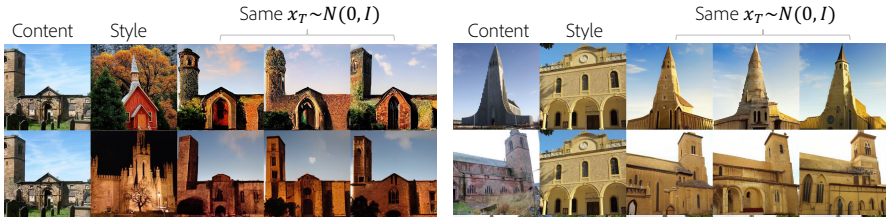


**Fig. 7:** Reference-based image translation results on LSUN-church.

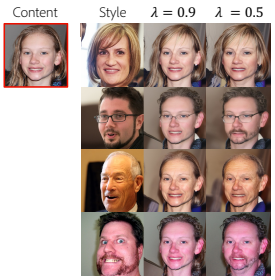


**Fig. 8:** Visualizations of a wide spectrum of our generation given a fixed content with different styles.

The results are shown in Fig. 9 and Fig. 10. The first column denotes the input image while the rest of the columns show the top 10 images that have the closest content or style features indicated by $z_c$ (first row within each macro row) and $z_s$ (second row within each macro row) respectively. The second column is the same image. We can see that the content feature mainly contains the pose and the facial shape information while the style has high-level semantics, such as wearing eyeglasses, gender, age, accessories, and hair color.

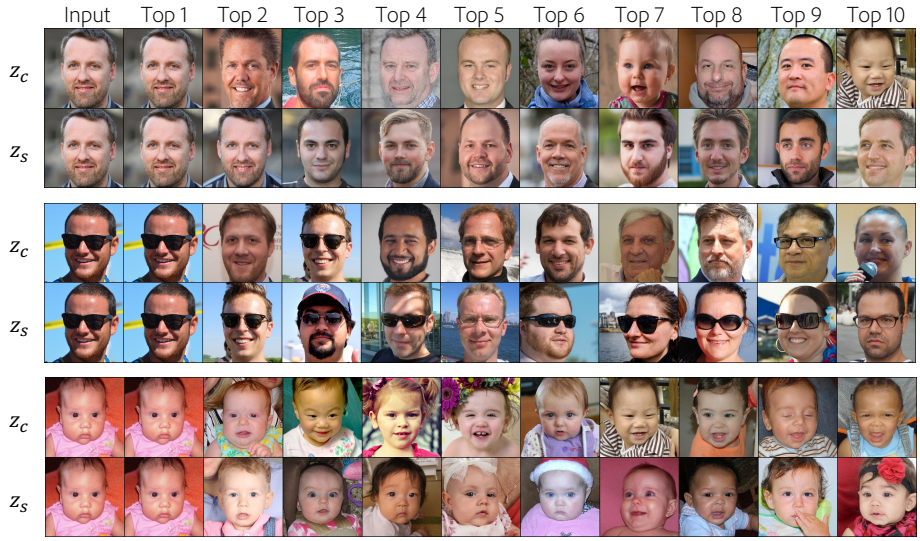


**Fig. 9:** KNN results of the content and the style features showing what semantic attributes content and style codes encode.


**Fig. 10:** KNN results of the content and the style features showing what semantic attributes content and style codes encode.

# G Timestep Scheduling

## G.1 Training an Implicit Mixture-Of-Experts

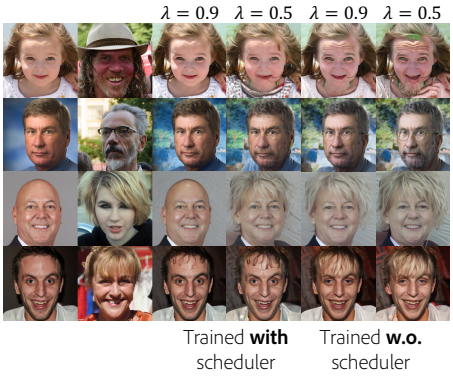


**Fig. 11:** Effects of using the proposed timestep scheduling in training.

Our timestep scheduling approach proposed in Sec 3.2 in the main paper was applied only during sampling for results in the main paper. We trained a model with timestep scheduling applied during training to analyze how it affects the behavior of our framework. Fig. 12 shows the comparisons between the models trained with and without the scheduler. For the results trained with scheduler, we used $a = 0.1$ and $b = 529$ ($\text{SNR}^{-1}(0.1)$) for both training and sampling. As can be seen in the third and fourth columns (i.e., trained with scheduler), the style effects are relatively small although given $\lambda$ is controlled. It is because the style encoder is trained to be injected only in the early timesteps (0-528), which makes the style representations learn limited features (e.g., eyeglasses are not encoded in the style, as shown in the second row). However, we observe better disentanglement of the content and the style spaces compared to using the timestep scheduling only during sampling. We believe this is because, using timestep scheduling to vary the conditioning input at each timestep implicitly trains the model to specialize to the varied conditioning, implicitly learning a mixture–of–experts like model [1]. We believe this could be a promising avenue for future research.

## G.2 Experiment with different timestep schedules

We compare the different timestep schedulers illustrated in Fig. 12 during sampling. Note that these schedules are not used for training. In the exclusive scheduling, the style weight is one if $t \leq 550$ and zero otherwise. The content weight is applied when style weight is not applied. In the linear scheduling, the style weight linearly decreases from 1 at $t = 0$ to 0 at $t = 999$ while the content

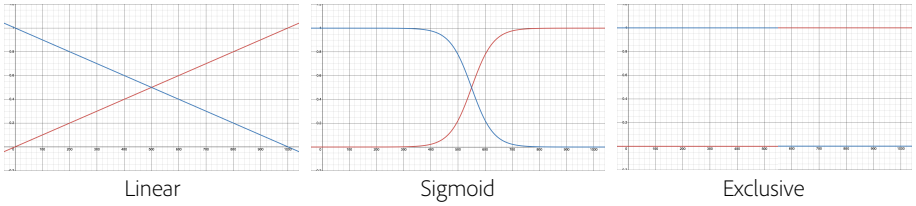


**Fig. 12:** Plots for different timestep scheduling strategies. The illustrated plot of the sigmoid scheduler is from $a = 0.025$ and $b = 550$. Bigger $a$ makes it similar to the exclusive scheduler while smaller $a$ makes it close to the linear scheduler. The blue line indicates the weight scheduler for the style and the red line is for the content.

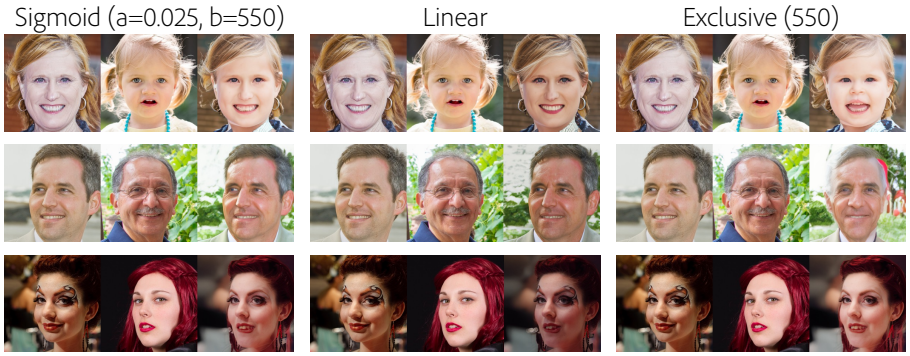


**Fig. 13:** Comparisons between different timestep schedulers during sampling. Sigmoid has a softer schedule with more controllability and thus results are more natural generations compared to the other techniques.

weight increases linearly from 0 to 1. The sigmoid scheduling is the one proposed in Eq. 11 in the main paper.

The comparison results are shown in Fig. 13. We can observe that the exclusive scheduling shows either magnified style or unnatural generations compared to the sigmoid scheduling. Since it is difficult to exactly define the role of each timestep, naively separating the point where to exclusively apply the content and style yields undesirable results. The linear schedule does not work for all images and has limited control. However, the sigmoid scheduling provides a softer weighting scheme leading to better generations, and has additional controls to get desired results.

## H Additional Results

In this Section, we provide additional results of our proposed framework. Fig. 14 shows example generations using CDM and GCDM from the same model. CDM consistently shows worse results than GCDM in reference-based image translation. We can see that by assuming conditional independence, CDM unnaturally overlaps the content and style features.

Fig. 15 shows the effect of the starting point $x_T$ given same content and style codes. Fig. 16 shows the additional results on FFHQ dataset. Fig. 17 shows the results of multiple style and single content, and vice versa on LSUN church dataset. Fig. 18 shows the hyperparameters used for CDM and GCDM on Stable Diffusion V2.


**Fig. 14:** Comparisons between GCDM and CDM demonstrating that CDM can output unnatural images while GCDM can generate realistic images. We use DDIM [18] sampler, and the reverse process is done from $T = 600$ inspired by SDEdit [12]. $z_{600}$ is obtained by $q(z_{600}|E_{LDM}(x_c))$ using the content image. $x_T$ is randomly sampled.


**Fig. 15:** Example showing the role of the denoising network during sampling when content and style codes are unchanged. $x_T$ is randomly sampled. The images show that the denoising network play a role in stochasticity since the outputs have consistent shape, color and texture information while minor details of the buildings or clouds are changed.


**Fig. 16:** Additional results on FFHQ. The results are sampled by *reverse DDIM*.

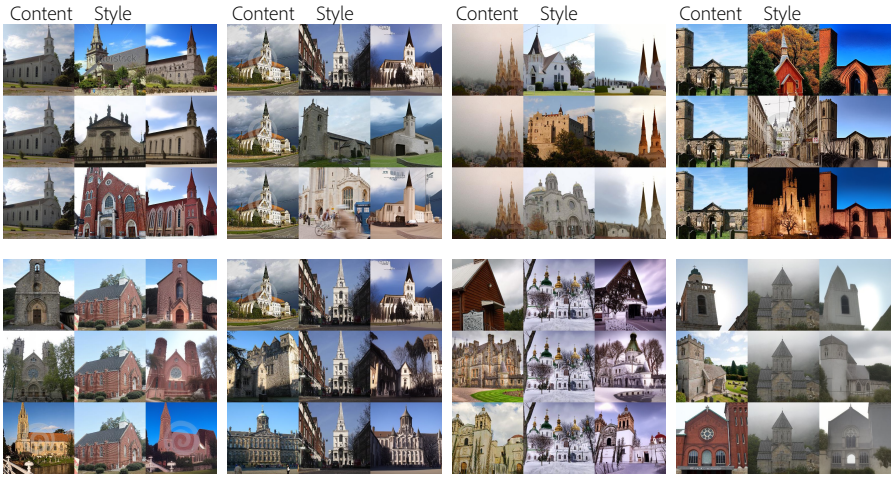


**Fig. 17:** Additional results on LSUN-church. The results are sampled by *reverse DDIM*.

$c_1$ = Photo of a bear
$c_2$ = Photo of a car in the red forest
$c_{1,2}$ = Photo of a bear and a car in the red forest

CDM
$(\alpha = 9.0, \lambda = 0.0, \beta_1 = 1.0, \beta_2 = 1.0)$

GCDM
$(\alpha = 9.0, \lambda = 1.0)$

GCDM
$(\alpha = 9.0, \lambda = 0.85, \beta_1 = 1.0, \beta_2 = 0.0)$

CDM
$(\alpha = 9.0, \lambda = 0.0, \beta_1 = 1.0, \beta_2 = 1.0)$

GCDM
$(\alpha = 9.0, \lambda = 1.0)$

GCDM
$(\alpha = 9.0, \lambda = 0.5, \beta_1 = 0.0, \beta_2 = 1.0)$

$c_1$ = Photo of a couch
$c_2$ = Photo of a dog sitting in the living room
$c_{1,2}$ = Photo of a couch and a dog sitting in the living room

**Fig. 18:** Text2image synthesis results with GCDM hyperparameters.

# Bibliography


[1] Balaji, Y., Nah, S., Huang, X., Vahdat, A., Song, J., Kreis, K., Aittala, M., Aila, T., Laine, S., Catanzaro, B., et al.: ediffi: Text-to-image diffusion models with an ensemble of expert denoisers. arXiv preprint arXiv:2211.01324 (2022)

[2] Choi, Y., Uh, Y., Yoo, J., Ha, J.W.: Stargan v2: Diverse image synthesis for multiple domains. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 8188–8197 (2020)

[3] Esser, P., Rombach, R., Ommer, B.: Taming transformers for high-resolution image synthesis. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 12873–12883 (2021)

[4] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N., Weinberger, K. (eds.) Advances in Neural Information Processing Systems. vol. 27. Curran Associates, Inc. (2014), https://proceedings.neurips.cc/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf

[5] Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., Hochreiter, S.: Gans trained by a two time-scale update rule converge to a local nash equilibrium. Advances in neural information processing systems **30** (2017)

[6] Ho, J., Jain, A., Abbeel, P.: Denoising diffusion probabilistic models. Advances in Neural Information Processing Systems **33**, 6840–6851 (2020)

[7] Ho, J., Salimans, T.: Classifier-free diffusion guidance. arXiv preprint arXiv:2207.12598 (2022)

[8] Karras, T., Laine, S., Aila, T.: A style-based generator architecture for generative adversarial networks. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 4401–4410 (2019)

[9] Kwon, G., Ye, J.C.: Diffusion-based image translation using disentangled style and content representation. arXiv preprint arXiv:2209.15264 (2022)

[10] Liew, J.H., Yan, H., Zhou, D., Feng, J.: Magicmix: Semantic mixing with diffusion models. arXiv preprint arXiv:2210.16056 (2022)

[11] Liu, N., Li, S., Du, Y., Torralba, A., Tenenbaum, J.B.: Compositional visual generation with composable diffusion models. Proceedings of the European conference on computer vision (ECCV) (2022)

[12] Meng, C., He, Y., Song, Y., Song, J., Wu, J., Zhu, J.Y., Ermon, S.: Sdedit: Guided image synthesis and editing with stochastic differential equations. In: International Conference on Learning Representations (2021)

[13] Park, T., Zhu, J.Y., Wang, O., Lu, J., Shechtman, E., Efros, A., Zhang, R.: Swapping autoencoder for deep image manipulation. Advances in Neural Information Processing Systems **33**, 7198–7211 (2020)

[14] Preechakul, K., Chatthee, N., Wizadwongsa, S., Suwajanakorn, S.: Diffusion autoencoders: Toward a meaningful and decodable representation. In:

Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 10619–10629 (2022)
[15] Qi, T., Fang, S., Wu, Y., Xie, H., Liu, J., Chen, L., He, Q., Zhang, Y.: Deadiff: An efficient stylization diffusion model with disentangled representations. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 8693–8702 (2024)
[16] Rombach, R., Blattmann, A., Lorenz, D., Esser, P., Ommer, B.: High-resolution image synthesis with latent diffusion models. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 10684–10695 (2022)
[17] Sohl-Dickstein, J., Weiss, E., Maheswaranathan, N., Ganguli, S.: Deep unsupervised learning using nonequilibrium thermodynamics. In: International Conference on Machine Learning. pp. 2256–2265. PMLR (2015)
[18] Song, J., Meng, C., Ermon, S.: Denoising diffusion implicit models. In: International Conference on Learning Representations (2020)
[19] Yu, F., Seff, A., Zhang, Y., Song, S., Funkhouser, T., Xiao, J.: Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. arXiv preprint arXiv:1506.03365 (2015)
[20] Zhang, L., Agrawala, M.: Adding conditional control to text-to-image diffusion models. arXiv preprint arXiv:2302.05543 (2023)