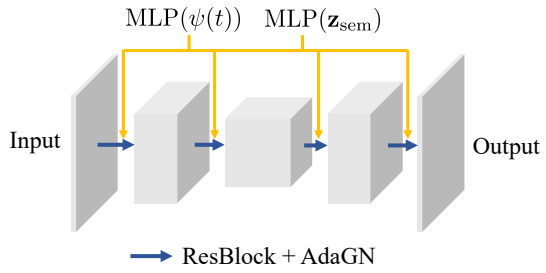
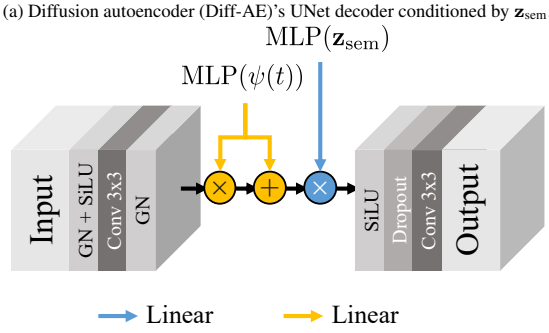
Appendix

A. Diffusion autoencoder architectures

The baseline diffusion models and our diffusion autoencoders are based on the same DDIM model [11] (publicly available at https://github.com/openai/guided-diffusion). The architecture is specified in Table 5. We selected the hyperparameters differently due to the limited computational resources. Note that we used the linear β scheduler as in Ho et al. [22], but we do observe improvements using the cosine β scheduler [36] in our preliminary results.



(a) Diffusion autoencoder (Diff-AE)'s UNet decoder conditioned by $\mathbf{z}_{\text{sem}}$.



(b) ResBlock + AdaGN. The residual path is not depicted.

Figure 7. Architecture overview of our diffusion autoencoder.

A.1. Latent DDIM architectures

For latent DDIMs, we experimented with multiple architectures including MLP, MLP + skip connections, and projecting $\mathbf{z}_{\text{sem}}$ into a spatial vector before using a CNN or UNet. We have found that MLP + skip connection performed reasonably well while being very fast (See unconditional samples in Figure 20). The architecture is specified in Table 6. Each layer of the MLP has a skip connection from the input, which simply concatenates the input with the output from the previous layer. The network is conditioned on t by scaling the hidden representations to help denoising. The architecture is shown in Figure 8 and the hyperparameters are shown in Table 6.

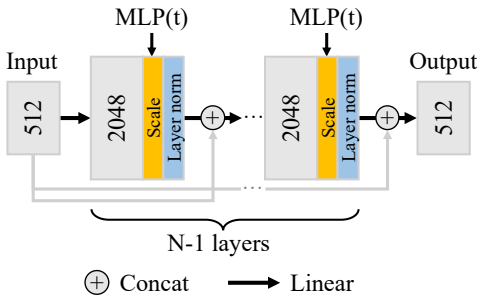


Figure 8. Architecture overview of our latent DDIM.

We have compared different β schedulers including Linear [22], and a constant of 0.008 schedulers. (We found that Cosine [36] scheduler underperformed during preliminary experiments for our latent DDIM.) We compared the two schedulers on the $\mathbf{z}_{\text{sem}}$ of LSUN's Horse 128 diffusion autoencoder model. The latent DDIM is MLP + Skip with 10 layers and 2048 hidden nodes. The validation FID score for using linear beta schedule is 13.36, whereas for constant 0.008 scheduler is 10.50. We found that an L1 loss performed better for the latent DDIM with FID of 11.65 vs 13.36 of MSE (Though, the main autoencoder uses MSE loss). We provide the hyperparameter tuning results of the MLP + Skip network:

Latent model	**FID**
Linear β, 10 layers, size 2048	13.36
Constant 0.008 & L1	
- 10 layers	10.16
- size 3072	9.57
- size 4096	9.43
- 15 layers	9.58
- 20 layers	**9.30**

Even though these results come from LSUN's Horse dataset, we found that similar settings worked well across datasets. We only tuned the network depth and the total training iterations for each dataset separately, a common practice in StyleGAN's training on these datasets.

A.2. Classifiers

We always use linear classifiers (logistic regression) trained on $\mathbf{z}_{\text{sem}}$ space in all relevant experiments, which are attribute manipulation and class-conditional sampling. For training, $\mathbf{z}_{\text{sem}}$ is first normalized so that its entire distribution has zero mean and unit variance before putting to the classifier. For the PU classifier, we oversampled the positive data points to match the negative ones to maintain the balance. For conditional generation, we follow D2C and apply rejection sampling after an additional thresholding.

Table 5. Network architecture of our diffusion autoencoder based on the improved DPM architecture of Dhariwal et al. [11].

Parameter	CelebA 64	FFHQ 64	FFHQ 128	Horse 128	Bedroom 128	FFHQ256
Batch size	128	128	128	128	128	64
Base channels	64	64	128	128	128	128
Channel multipliers	[1,2,4,8]	[1,2,4,8]	[1,1,2,3,4]	[1,1,2,3,4]	[1,1,2,3,4]	[1,1,2,2,4,4,]
Attention resolution	[16]	[16]	[16]	[16]	[16]	[16]
Images trained	72M	48M	130M	130M	120M	90M
Encoder base ch	64	64	128	128	128	128
Enc. attn. resolution	[16]	[16]	[16]	[16]	[16]	[16]
Encoder ch. mult.	[1,2,4,8,8]	[1,2,4,8,8]	[1,1,2,3,4,4]	[1,1,2,3,4,4]	[1,1,2,3,4,4]	[1,1,2,2,4,4,4]
$\mathbf{z}_{\text{sem}}$ size	512	512	512	512	512	512
β scheduler	Linear	Linear	Linear	Linear	Linear	Linear
Learning rate	1e-4					
Optimizer	Adam (no weight decay)					
Training T	1000					
Diffusion loss	MSE with noise prediction ϵ					
Diffusion var.	Not important for DDIM					

Table 6. Network architecture of our latent DDIM.

Parameter	CelebA	FFHQ	Horse	Bedroom
Batch size	512	256	2048	2048
$\mathbf{z}_{\text{sem}}$ trained	300M	100M	2000M	2000M
MLP layers (N)	10	10	20	20
MLP hidden size	2048			
$\mathbf{z}_{\text{sem}}$ size	512			
β scheduler	Constant 0.008			
Learning rate	1e-4			
Optimizer	AdamW (weight decay = 0.01)		Adam (no weight decay)	
Train Diff T	1000			
Diffusion loss	L1 loss with noise prediction ϵ			
Diffusion var.	Not important for DDIM			

That is, we reject samples with the target class probabilities less than 0.5 before performing rejection sampling.

B. Computation resources

We used four Nvidia V100s for both diffusion autoencoders and DDIM and a single Nvidia RTX 2080 Ti for the latent DDIMs. Training the latent DDIMs takes only a fraction of the computational resources compared to the diffusion autencoders. Table 7 shows the throughputs of DDIM and diffusion autoencoders. Diffusion autoencoders were around 20% slower to train than DDIM counterparts due to the additional semantic encoder. The total GPU-hours can be computed by multiplying the throughput with the number of training images for each model provided in Table 6.

Table 7. Throughputs of DDIM and diffusion autoencoders.

Model	DDIMs Throughput (imgs/sec./V100)	Diffusion autoencoders Throughput (imgs/sec./V100)
FFHQ-64	160	128
FFHQ-128	51	41.65
FFHQ-256	-	10.08
Horse-128	51	41.65
Bedroom-128	51	41.65

C. Does the latent DDIM memorize its input?

To verify if our diffusion autoencoder and latent DDIM can generate novel samples and do not simply memorize the input, we generate image samples and compare them to their nearest neighbors in the training set (Figure 9). (They

Figure 9. **Does latent DDIM memorize its input?** For each sampled image at the top, we find its closest images from the training set in terms of LPIPS, MSE in the image space, and MSE in the semantic subcode $\mathbf{z}_{\text{sem}}$ space. The sampled images do not closely resemble any of the training images, suggesting that our latent DDIM does not memorize the input samples.

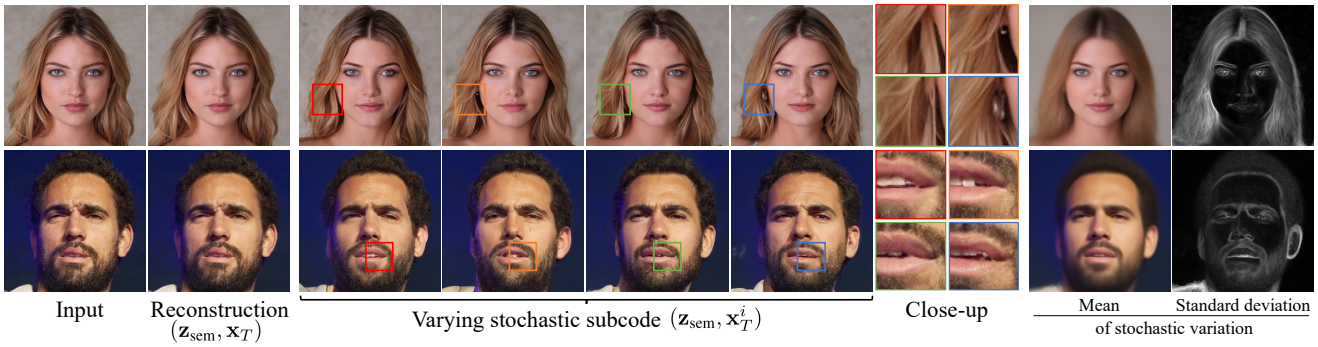


Figure 10. Reconstruction results and the variations induced by changing the stochastic subcode $\mathbf{x}_T$.

should look different). To find nearest neighbors, we used three different metrics: 1) lowest LPIPS [61] in the image space, 2) lowest MSE in the image space, 3) lowest MSE in the semantic subspace ($\mathbf{z}_{\text{sem}}$). We have found that our autoencoder can generate substantially different images from the training set, suggesting no memorization problem.

D. What is encoded in the stochastic subcode?

Figure 10 shows the stochastic variations induced by varying $\mathbf{x}_T$ given the same $\mathbf{z}_{\text{sem}}$. We also compute the mean and standard deviation of these variations. All generated images look realistic and $\mathbf{x}_T$ changes only minor details, such as the hair pattern, while keeping the overall structure the same.

E. Predictive power of the semantic subcode

We assess the quality of our proposed $\mathbf{z}_{\text{sem}}$ via linear classification performance, which has been extensively used to evaluate the quality of learned representations [4, 7, 16, 19]. In Table 8, we measure the performance of linear classifiers trained on $\mathbf{z}_{\text{sem}}$ and StyleGAN's latent code in $\mathcal{W}$ space (obtained from an inversion process [28]) using Area Under the Receiver Operating Characteristic (AUROC) on the CelebA-HQ's 40 attributes with 30% test data out of 30,000 total data points. The classifiers were trained on z-normalized latent vectors until convergence with Adam optimizer (learning rate 1e-3). For most classes, the linear classifiers using $\mathbf{z}_{\text{sem}}$ outperform those using StyleGAN's $\mathcal{W}$ with weighted averages of 0.92 vs 0.89. This suggests that $\mathbf{z}_{\text{sem}}$ contains attribute-specific information that is more readily predictive than that of StyleGAN's $\mathcal{W}$.

Table 8. Classification AUROC ↑ on CelebA-HQ's 40 attributes of linear classifiers trained on our $\mathbf{z}_{\text{sem}}$ vs. StyleGAN's latent code in $\mathcal{W}$ space (obtained via inversion).

Class	#Positives	$\mathbf{z}_{\text{sem}}$	$\mathcal{W}$
5_o_Clock_Shadow	1318	**0.96**	0.94
Arched_Eyebrows	3262	**0.88**	0.86
Attractive	5183	**0.90**	0.86
Bags_Under_Eyes	2564	**0.89**	0.85
Bald	229	0.99	0.99
Bangs	1601	**0.98**	0.95
Big_Lips	3247	**0.73**	0.68
Big_Nose	2813	**0.88**	0.85
Black_Hair	1989	**0.96**	0.93
Blond_Hair	1546	**0.99**	0.97
Blurry	34	**0.90**	0.82
Brown_Hair	2087	**0.89**	0.81
Bushy_Eyebrows	1682	**0.93**	0.85
Chubby	622	**0.95**	0.93
Double_Chin	530	**0.95**	0.94
Eyeglasses	416	1.00	0.98
Goatee	688	**0.98**	0.96
Gray_Hair	395	**0.98**	0.97
Heavy_Makeup	4143	**0.97**	0.95
High_Cheekbones	4160	**0.95**	0.91
Male	3273	1.00	1.00
Mouth_Slightly_Open	4195	**0.98**	0.94
Mustache	502	**0.97**	0.94
Narrow_Eyes	998	**0.86**	0.77
No_Beard	7335	**0.99**	0.97
Oval_Face	1872	**0.77**	0.71
Pale_Skin	434	**0.96**	0.94
Pointy_Nose	2855	**0.74**	0.70
Receding_Hairline	777	**0.94**	0.89
Rosy_Cheeks	1003	**0.96**	0.92
Sideburns	747	**0.99**	0.97
Smiling	4175	**0.99**	0.96
Straight_Hair	1975	**0.84**	0.77
Wavy_Hair	3197	**0.90**	0.87
Wearing_Earrings	2310	**0.92**	0.86
Wearing_Hat	325	**0.99**	0.96
Wearing_Lipstick	5064	**0.98**	0.97
Wearing_Necklace	1501	**0.79**	0.75
Wearing_Necktie	636	**0.96**	0.95
Young	6978	**0.94**	0.91
Weighted average		**0.92**	0.89
Macro average		**0.93**	0.89

F. Real-image interpolation results

We show interpolation results on real images from FFHQ [27] (Figure 14), LSUN-Bedroom [60] (Figure 15) and LSUN-Horse [60] (Figure 16). Our method can handle challenging morphing between people with and without glasses, bedrooms from different styles and angles, or horses with different body poses.

To quantify the smoothness of the interpolation, we use Perceptual Path Length (PPL) introduced in StyleGAN [27], to measure the perceptual difference in the image as we move along the interpolation path by a small $\epsilon = 10^{-4}$ in the latent space. Specifically, we compute the following expectation over multiple sampled pairs of latent codes (z_1, z_2) and $t \in [0, 1]$:

$$\text{PPL} = \mathbb{E}\left[\frac{1}{\epsilon^2} d(G(\text{slerp}(z_1, z_2; t)), G(\text{slerp}(z_1, z_2; t+\epsilon))\right] \tag{10}$$

where G is the decoder, and d computes the perceptual distance based on the VGG16 network. slerp$(\cdot)$ denotes spherical interpolation. We compute this expected value using 200 samples (400 images) from FFHQ. Our method significantly outperforms DDIM in terms of interpolation smoothness as shown below.

Model	DDIM	**Ours**
PPL	2,634.14	**613.73**

G. Real-image attribute manipulation results

We show real-image attribute manipulation results on FFHQ [27] and CelebA-HQ [26] in Figure 5 for smiling, wavy hair, aging, and gender change. For more results, please visit https://Diff-AE.github.io/. Our generated results look highly realistic and plausible.

FID between the input and its manipulated version. To assess the quality of our manipulated results, we compare their distribution with that of real images with the target positive attribute, such as smiling. Our manipulation is done by moving $\mathbf{z}_{\text{sem}}$ linearly along the target direction $\mathbf{w}$, found by training a linear classifier (logistic regression) $\mathbf{y} = \mathbf{w}^\top \mathbf{z} + b$ to predict the target attribute using a labeled dataset. The stochastic subcode $\mathbf{x}_T$ is kept intact. Given $\mathbf{z}$, its manipulated version is produced by decoding $\mathbf{z}' = \mathbf{z} + s\mathbf{w}$, where $s \in \mathbb{R}$ controls the degree of manipulation. For this experiment, each input image will be manipulated by a different s_i so that the manipulated result reaches the same degree of the target attribute (e.g., similarly big smile) Specifically, we pick s_i so that the logit confidence of its $\mathbf{z}'_i$ equals the median confidence of all real positive images:

$$s_i = \frac{\text{median} - b - \mathbf{z}_i^\top \mathbf{w}}{\mathbf{w}^\top \mathbf{w}} \tag{11}$$

In our implementation, we use normalized $\mathbf{z}$ instead of $\mathbf{z}$ for this operation and unnormalize it before decoding.

In Table 9, we measure FID scores ↓ between the manipulated (to be positive) and real positive images, as well as FID scores between real negative and real positive images as baselines for five different attributes from CelebA-HQ [26]. While we expect the manipulated images to get closer to the positive images, we also expect them to not deviate too far from the negative as some original content, such as the background, the identity, should be retained. Hence, we also provide FID scores ↓ between the manipulated images and the real negative images. Our $\mathbf{z}_{\text{sem}}$ manipulated images are closer to the real positive images for 4 out of 5 attributes than those of StyleGAN-$\mathcal{W}$ while better preserving the original contents in all 5 attributes.

Identity preservation. We quantitatively evaluate how well the input's identity is preserved under the manipulation by computing the cosine similarity ↑ between the ArcFace embeddings [10] of the input and its manipulated version, following [39]. Table 10 shows our scores on CelebA-HQ images of 4 classes used in Figure 5: Male, Smiling, Wavy Hair, Young. For this experiment, we use the original $\mathcal{W}$ space inversion of StyleGAN that produces the same 512D latent code as our $\mathbf{z}_{\text{sem}}$. Their lower scores can be attributed partly to the poor inversion in this space.

Table 10. Average cosine similarity ↑ of the ArcFace embeddings [10] of the input and its manipulated version.

Model	Male	Smiling	Wavy Hair	Young
StyleGAN-$\mathcal{W}$	0.4174	0.7850	0.8544	0.6955
Ours	**0.6247**	**0.8160**	**0.9821**	**0.8922**

H. Attribute manipulation comparison to D2C

We show a qualitative comparison to D2C [45] on real-image attribute manipulation in Figure 17. These official D2C's results are from https://d2c-model.github.io/. The results of the other baselines are also borrowed from the same website.

I. Class-conditional samples

We show our conditional samples of *Blond*, *Non-blond*, *Famale*, and *Male* classes in Figures 18, 19. This is done by training a linear classifier for each attribute using only 100 labeled examples and 10k unlabeled examples, similar to the few-shot experiment done in D2C [45]. The details are in Section 5.6 in the main paper.

J. Unconditional samples

We show uncurated unconditional samples from our diffusion autoencoder on FFHQ [27], LSUN-Bed [60], and LSUN-horse [60] in Figure 20, 21, 22.

K. Encoding out-of-distribution images

As discussed in the main paper, when encoding images that are out of the training distribution, our diffusion autoencoders can still reconstruct the images well but the inferred semantic and stochastic subcodes may fall outside the learned distributions. We simulate simple out-of-distribution samples by translating an FFHQ face image in Figure 11 and by encoding a horse image using our diffusion autoencoder trained on *face* images in Figure 12. The reconstruction results still look very close to the input images, but the noise maps $\mathbf{x}_T$ show some residual details and do not look normally distributed.

Figure 11. Noise maps $\mathbf{x}_T$ when the input face image is shifted to the right to simulate out-of-distribution input image.

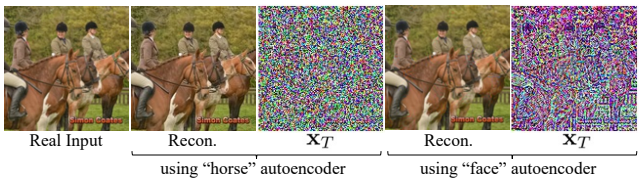


Figure 12. We test how the noise map $\mathbf{x}_T$ of a horse image would look if it is encoded by a diffusion autoencoder trained on face images. Both reconstructions look reasonably close to the input image, but $\mathbf{x}_T$ from the face autoencoder does not look normally distributed and contains details from the input image.

L. Potential negative impact

The ability to generate image samples and manipulate attributes of a real image can be used to generate synthetic media, such as deepfakes. We realize the potential negative impact and further conducted a study to determine the difficulty in distinguishing real and synthesized images from our method, as well as discussing some possible directions.

To detect fake images, we train a CNN architecture based on ResNet-50 [20], which is pretrained on ImageNet [9], followed by a linear layer used for classification. Our training dataset consists of "real" images from FFHQ256 [27]

Table 9. Image manipulation FID scores ↓.

Mode	Model	Male	Smiling	Wavy Hair	Young	Blond Hair
Positive vs negative		95.82	11.15	25.04	36.75	39.65
Manipulated vs. positive	**Ours**	52.85	**9.19**	**20.80**	**20.68**	**33.51**
	StyleGAN-$\mathcal{W}$	**42.90**	18.52	27.10	31.15	33.89
Manipulated vs. negative	**Ours**	**23.15**	**7.25**	**4.89**	**11.81**	**6.79**
	StyleGAN-$\mathcal{W}$	66.92	22.15	20.70	31.15	27.54

and "fake" images from either the unconditional sampling experiment (Section 5.7) or the attribute manipulation experiment (Section 5.3). This dataset contains 20k images: 10K images for each real and fake. The dataset is randomly split into train, test, and validation class-balanced subsets with the ratios of 0.7, 0.2, and 0.1, respectively. The classifier is trained using a binary cross-entropy loss function with an SGD optimizer (learning rate 0.001, momentum 0.9, batchsize 64). Fake detection accuracy is reported here:

Method	T=100	T=200	T=500
Unconditional sampling	0.9551	0.9483	0.9313
Attribute manipulation	0.9950	0.9643	0.9213

The results suggest that even though the generated samples look highly realistic, there could be some certain artifacts that can be easily detected by another neural network. Diffusion-based models also do not have a mechanism to purposely fool a classifier or discriminator like GANs do, and a neural network-based technique is currently found to be $> 90\%$ effective at detecting fake images from diffusion models. Note that sampling with higher T leads to samples that are harder to detect. A further study on how easy it is to circumvent detection through adversarial training and an analysis on those giveaway artifacts will be useful for future technical safeguards.

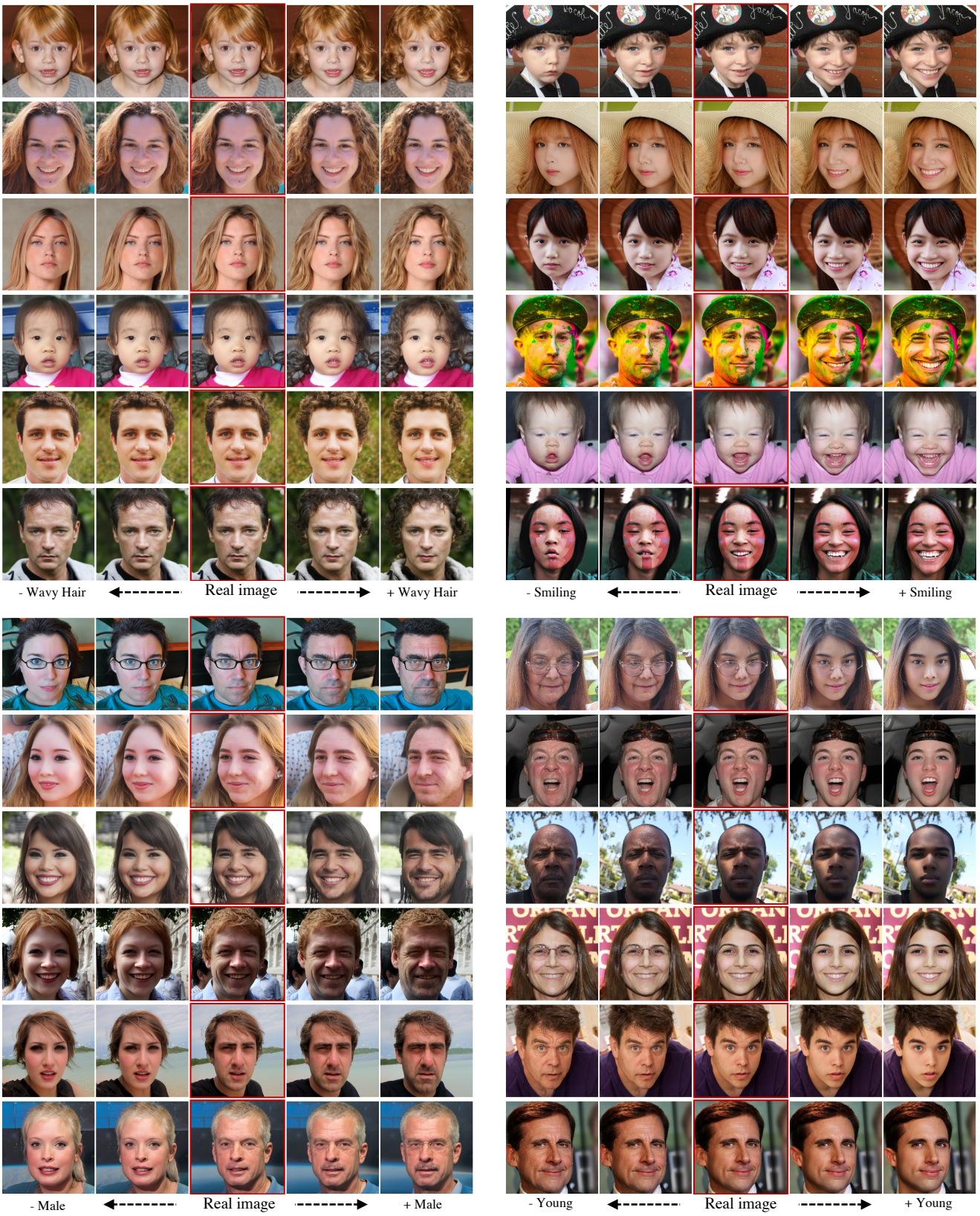


Figure 13. Real-image attribute manipulation for attributes: *Wavy Hair, Smiling, Male, Young*.

Figure 14. Real-image interpolation on FFHQ dataset [27]

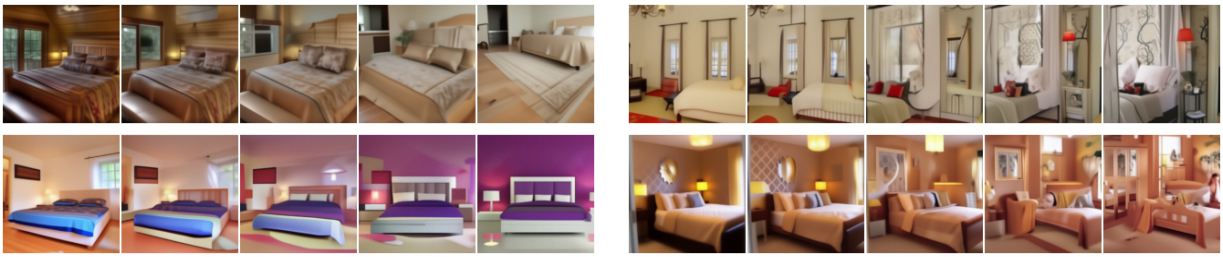


Figure 15. Real-image interpolation on LSUN bedroom-128 [60]

Figure 16. Real-image interpolation on LSUN horse-128 [60]

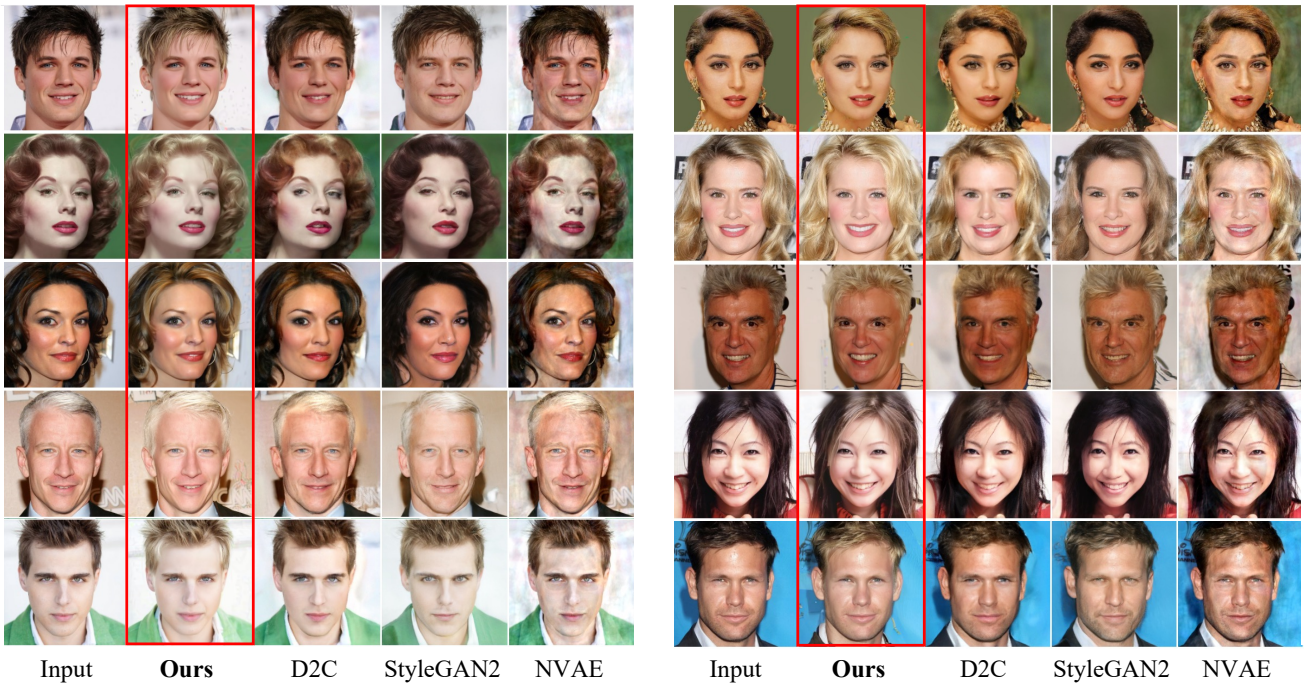


Figure 17. Comparison on attribute manipulation (blond hair) between our method, D2C [45], StyleGAN2 [28], and NVAE [50].

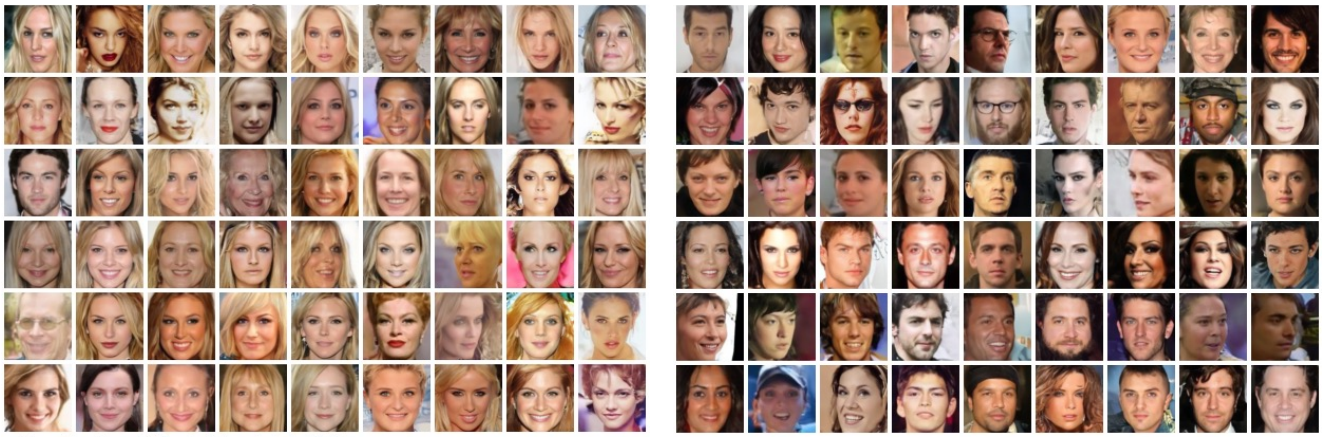


Figure 18. Class-conditional generation using 100 positive labeled examples and 10k unlabeled examples on *Blond* and *Non-blond* from CelebA [26]. These results are uncurated. Please see Section 5.6 in the main paper for details.

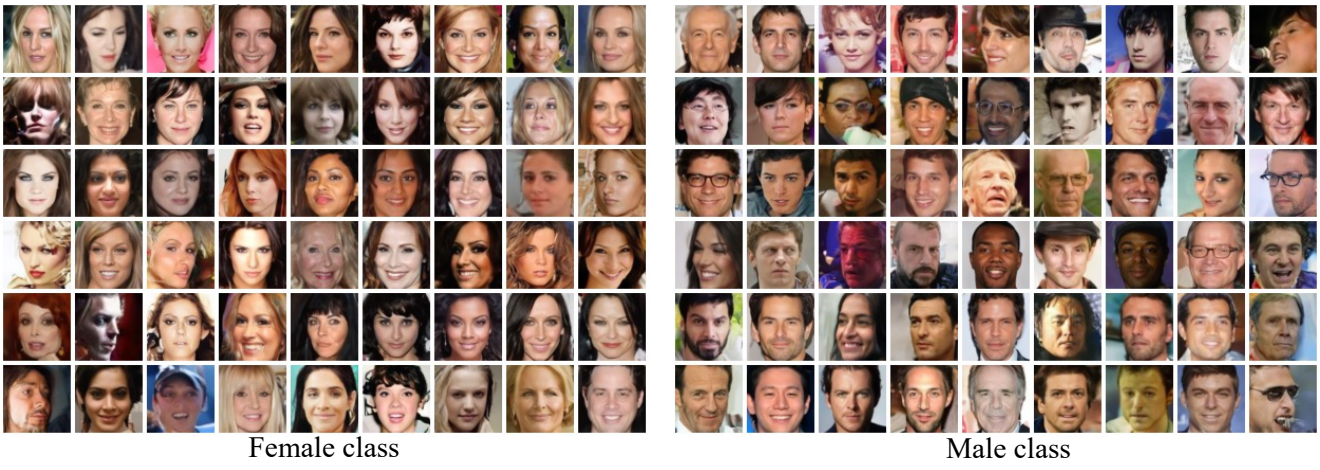


Figure 19. Class-conditional generation using 100 positive labeled examples and 10k unlabeled examples on *Female* and *Male* from CelebA [26]. These results are uncurated. Please see Section 5.6 in the main paper for details.

Figure 20. Unconditional samples (uncurated) from our diffusion autoencoder and latent DDIM trained on FFHQ-256 [27].

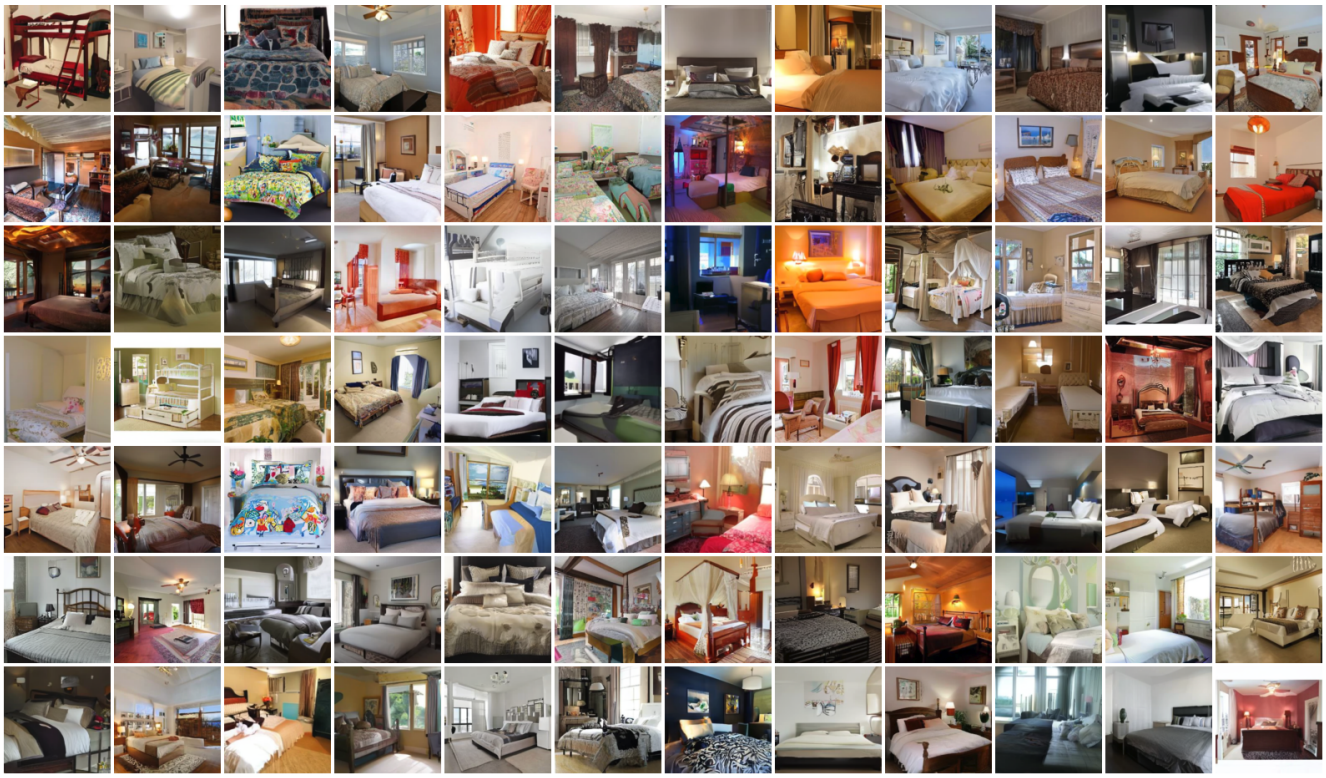


Figure 21. Unconditional samples (uncurated) from our diffusion autoencoder and latent DDIM trained on LSUN bedroom-128 [60].

Figure 22. Unconditional samples (uncurated) from our diffusion autoencoder and latent DDIM trained on LSUN horse-128 [60].