Supplementary Material: Compensation Sampling for Improved Convergence in Diffusion Models

Hui Lu[1], Albert Ali Salah[1], and Ronald Poppe[1]

Utrecht University, Princetonplein 5, 3584 CC Utrecht, The Netherlands
{h.lu1, a.a.salah, r.w.poppe}@uu.nl

We first give an overview of our implementation in Section 1 to allow replication of our network and training process. Upon acceptance of the paper, our Github repository will be made available publicly. Next, we provide an analysis of the effect of the number of training epochs of the compensation module (Section 2). We then present additional quantitative and qualitative results for the three tasks in the main paper: unconditional face generation (Section 3), face inpainting (Section 4), and face de-occlusion (Section 5). Finally, we demonstrate the merits of our compensation sampling approach beyond tasks related to face reconstruction, and present additional experiments on unconditional general image generation using CIFAR-10 (Section 6).

1 Implementation details

1.1 Model architecture

Our diffusion model is based on the DDIM model, the Ablated Diffusion Model (ADM) (publicly available at `https://github.com/openai/guided-diffusion`). We summarize the architecture in Table 1.

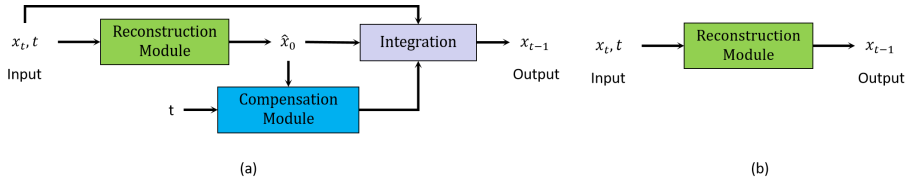

Fig. 1: Training and inference of our diffusion model during the denoising process. (a) Accelerated training process using the compensation module (b) Inference process for sampling from noise x_t

.

1.2 Training and inference pipeline

We show the pipeline of training and inference of our diffusion model in Figure 1. For training in Figure 1(a), during the denoising process, the input noise data x_t

Compensation Module	UNet Block	nn.Conv2d(in_channels, out_channels, 3, 1, 1, bias=False) nn.BatchNorm2d(out_channels) nn.ReLU() nn.MaxPool2d(kernel_size=2, stride=2)
	UNet Up Block	nn.ConvTranspose2d(in_channels_up, out_channels_up, kernel_size=2, stride=2)) UNetBlock(out_channels_up, out_channels_up//2) torch.cat(dim=1)
	Initial Layer	UNet Block(in_channels=3, out_channels=initial channel)
	Down Sampling	UNet Block(in_channels=channel multiplier[0], out_channels=channel_multiplier[1]) UNet Block(in_channels=channel_multiplier[0], out_channels=channel_multiplier[1]) UNet Block(in_channels=channel_multiplier[0], out_channels=channel_multiplier[1]) UNet Block(in_channels=channel_multiplier[0], out_channels=channel_multiplier[1])
	Up Sampling	UNet Up Block(channel_multiplier_up[0], channel_multiplier_up[1]) UNet Up Block(channel_multiplier_up[0], channel_multiplier_up[1]) UNet Up Block(channel_multiplier_up[0], channel_multiplier_up[1]) UNet Up Block(channel_multiplier_up[0], channel_multiplier_up[1])
	Output Layer	nn.Conv2d(channel_multiplier_up[0], channel_multiplier_up[1], kernel_size=1)
Reconstruction Module	Residulal Block	Conv1 = nn.Conv2d(in_channels, in_channels, kernel_size=3, stride=1, padding=1) bn = nn.BatchNorm2d(in_channels) Conv2 = nn.Conv2d(in_channels, in_channels, kernel_size=3, stride=1, padding=1) identity = x out = self.bn1(self.Conv1(x)).relu() out = self.bn(Conv2(out)) out += identity out = self.relu(out)
	Residual Up Block	nn.ConvTranspose2d(in_channels_up, out_channels_up, kernel_size=2, stride=2)) Residual Block(out_channels_up, out_channels_up//2) torch.cat(dim=1)
	Initial layer	Residulal Block(in_channels=3, out_channels=initial channel)
	Down Sampling	Residulal Block(in_channels=channel multiplier[0], out_channels=channel_multiplier[1]) Residulal Block(in_channels=channel_multiplier[0], out_channels=channel_multiplier[1]) Residulal Block(in_channels=channel_multiplier[0], out_channels=channel_multiplier[1]) Residulal Block(in_channels=channel_multiplier[0], out_channels=channel_multiplier[1])
	Up Sampling	Residual Up Block(channel_multiplier_up[0], channel_multiplier_up[1]) Residual Up Block(channel_multiplier_up[0], channel_multiplier_up[1]) Residual Up Block(channel_multiplier_up[0], channel_multiplier_up[1]) Residual Up Block(channel_multiplier_up[0], channel_multiplier_up[1])
	Output Layer	nn.Conv2d(channel_multiplier_up[0], channel_multiplier_up[1], kernel_size=1)

Table 1: Network architecture of our diffusion model with compensation module.

first passes through the reconstruction module (with ADM backbone in Table 1) to produce the initial clean data reconstruction $\hat{x_0}$. $\hat{x_0}$ and t is further processed by the compensation module. The output of compensation module and $\hat{x_0}$ is integrated following our compensation sampling algorithm, and outputs x_{t-1}. The pipeline iteratively works until the diffusion model outputs the final reconstruction x_0. For inference in Figure 1(b), our approach aligns entirely with that of DDIM, but utilizes the reconstruction module for sampling.

2 Effect of compensation module training epochs

In the main paper, we use a single training epoch for the compensation module in the unconditional generation experiment, so that we can introduce additional noise to increase the diversity of the generation. In the face inpainting and face de-occlusion tasks, we use more training epochs for the compensation module, so the compensation term will generate pixels close to human faces, thus we have a higher chance to get realistic human faces.

Considering the compensation term will encourage more faithful reconstruction of the original image x_0, it has the possibility to memorize images from the training set. Although the generated diverse samples can qualitatively prove this is not the case, to further quantify the distribution coverage and show the

	CelebA-64			FFHQ-256		
Iterations	FID	Precision	Recall	FID	Precision	Recall
1	2.12	0.79	0.71	11.89	0.76	0.65
2	2.13	0.79	0.69	11.89	0.76	0.64
5	2.15	0.82	0.52	11.93	0.78	0.55
10	2.15	0.83	0.48	11.87	0.78	0.50
20	2.13	0.84	0.44	11.93	0.80	0.45
40	2.12	0.85	0.41	11.92	0.80	0.43
80	2.12	0.85	0.40	11.89	0.79	0.43

Table 2: Effect of training epochs of the compensation module on unconditional and conditional generation.

diversity of our approach, we use the improved precision and recall proposed in [3], and experiment with the unconditional generation task with CelebA-64 and FFHQ-256 dataset with increasing number of training epochs. The results are summarized in Table 2.

From Table 2, on CelebA-64, it follows that the initial precision is 0.79, and the recall is 0.71. When increasing the number of iterations, the FID remains stable, but the recall starts decreasing from 0.71 to 0.44 at 20 iterations, and it remains stable when further increasing the number of iterations. Similarly, on FFHQ-256 dataset, the initial precision is 0.76, and recall is 0.65. When increasing the iterations, the recall decreases from 0.65 to 0.45 at 20 iterations, and stays more or less the same for increasing numbers of iterations.

Based on the analysis we can see that the diversity is connected to the training epochs of compensation term since it will learn the data distribution to some extent, so we can choose a lower number of epochs during the unconditional generation to increase the diversity of results, and choose a higher number of epochs to generate pixels similar to human faces for face inpainting tasks.

3 Unconditional face generation

3.1 Additional qualitative results

We show more generation results on CelebA-64 and FFHQ-256 in Figures 2 and 3 with the same setting in the main paper. Again, the diversity of the faces becomes apparent. With a close-up in Figure 4, we can see the high quality details of the generated image. In particular, we observe realistic facial hair and wrinkles around the eyes and mouth.

3.2 Failure cases

We also present some failure cases of DDIM and our model in Figures 5 and 6 to better analyze the patterns. Similar to DDIM, our method's failures mainly

Fig. 2: Unconditional face generation on CelebA-64. More samples with compensation sampling.

focus on the details of face or hands. Symmetry is not always warranted and the generation of realistic hair remains challenging. In addition, DDIM tends to have structural errors, where the face shapes are corrupted. For example, the first and third faces of DDIM in Figure 5 have an improbable neck, and the second face of DDIM in Figure 6 is missing an eye.

4 Face inpainting on CelebA-HQ

We show more face inpainting results on CelebA-HQ-256 in Figures 7 and 8, where Figure 7 shows the input and the ground truth, and Figure 8 shows the generation results. Similar to the previous face inpainting task, we use more epochs to train the compensation module to generate pixels close to human faces, which can reduce the diversity of results to some extent. The failures mainly happen when large areas are occluded. In some cases, the eyes are not realistically recovered.

5 Face de-occlusion on FSG

Similar to the face inpainting task, and with the same settings, we show more face de-occlusion results on the FSG dataset in Figures 9, 10, and 11. Figure 9

Fig. 3: Unconditional face generation on FFHQ-256. More samples with compensation sampling.

shows the input, Figure 10 the ground truth, and Figure 11 shows the results generated with our model. Our results seem blurred, and failure cases occur especially when the top face area is occluded.

6 Unconditional generation on CIFAR-10

6.1 Experimental setting

To evaluate on a non-face generation task, we use the more general CIFAR-10 (60k images) [2] dataset. For fair comparison, our training hyper-parameters including batch size and decay rate are the same as in DDPM [1]. We again use Gaussian noise as our corruption mechanism, and adopt the fixed linear variance

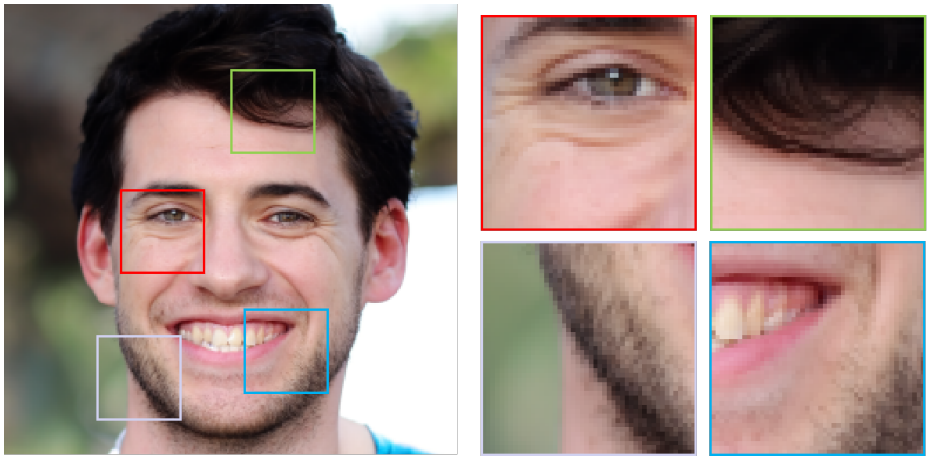

Fig. 4: Close-up of sample face generated after training on FFHQ-256. Note the realistic details in all parts of the face.

Fig. 5: Failure cases of DDIM and our model on CelebA-64.

schedule $\beta_1, ..., \beta_T$ as in DDPM for the prior noising process. In line with the experiments on unconditional face generation, we reduce the training time steps for our model to $T = 100$. Both the compensation module and diffusion model use the Adam optimizer. Experiments are conducted on 4 NVIDIA Tesla A100 GPUs.

6.2 Qualitative results

We show generated images of our method for CIFAR-10 in Figure 12. We observe a wide diversity of our method, and the images generally have good realism and quality.

6.3 Failure cases

We show and compare failure cases of DDIM and ours on CIFAR-10 in Figure 13 to better understand the limitations of either model.

Fig. 6: Failure cases of DDIM and our model on FFHQ-256.

Both DDIM and our model can generate images that are hard to classify. Specifically, DDIM tends to generate unknown animals, while our method tends to generate images of animals that combine features from different animals. For example, the first picture in the top row seems a combination of a horse and a deer. Considering that the network has not seen additional classification information, conditioning the generation on the class may largely prevent this issue. For both models, we observe that images of classes without articulation, such as cars and planes, are typically more realistically generated.

References

1. Ho, J., Jain, A., Abbeel, P.: Denoising diffusion probabilistic models. Advances in Neural Information Processing Systems **33**, 6840–6851 (2020)
2. Krizhevsky, A.: Learning multiple layers of features from tiny images. Tech. rep., University of Toronto (2009)
3. Kynkäänniemi, T., Karras, T., Laine, S., Lehtinen, J., Aila, T.: Improved precision and recall metric for assessing generative models. Advances in Neural Information Processing Systems **32** (2019)

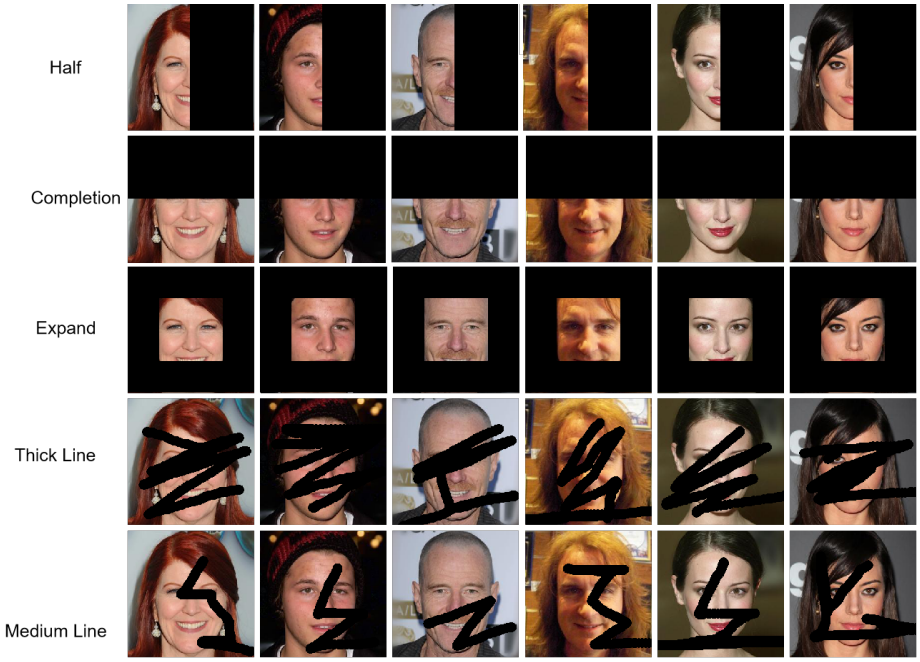

Fig. 7: Inputs for face inpainting experiment. Each column represents a different person.

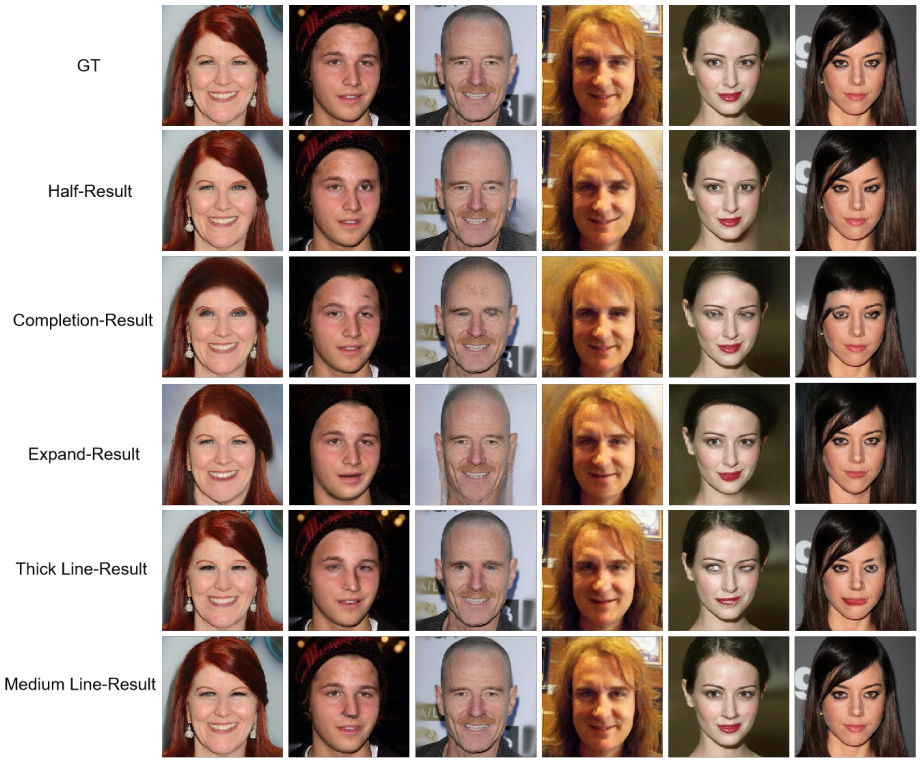

Fig. 8: Results on face inpainting obtained with our model trained on CelebA-HQ-256. Each column represents a different person.

Fig. 9: Inputs for face de-occlusion experiment from FSG.

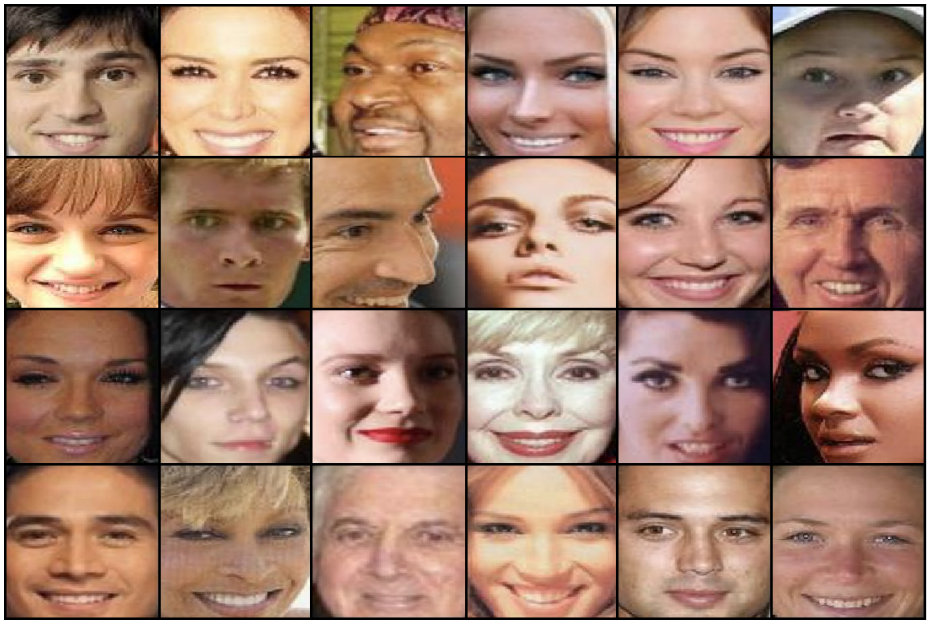

Fig. 10: Ground truth for face inpainting obtained with our model trained on FSG.

Fig. 11: Results on face inpainting obtained with our model trained on FSG.

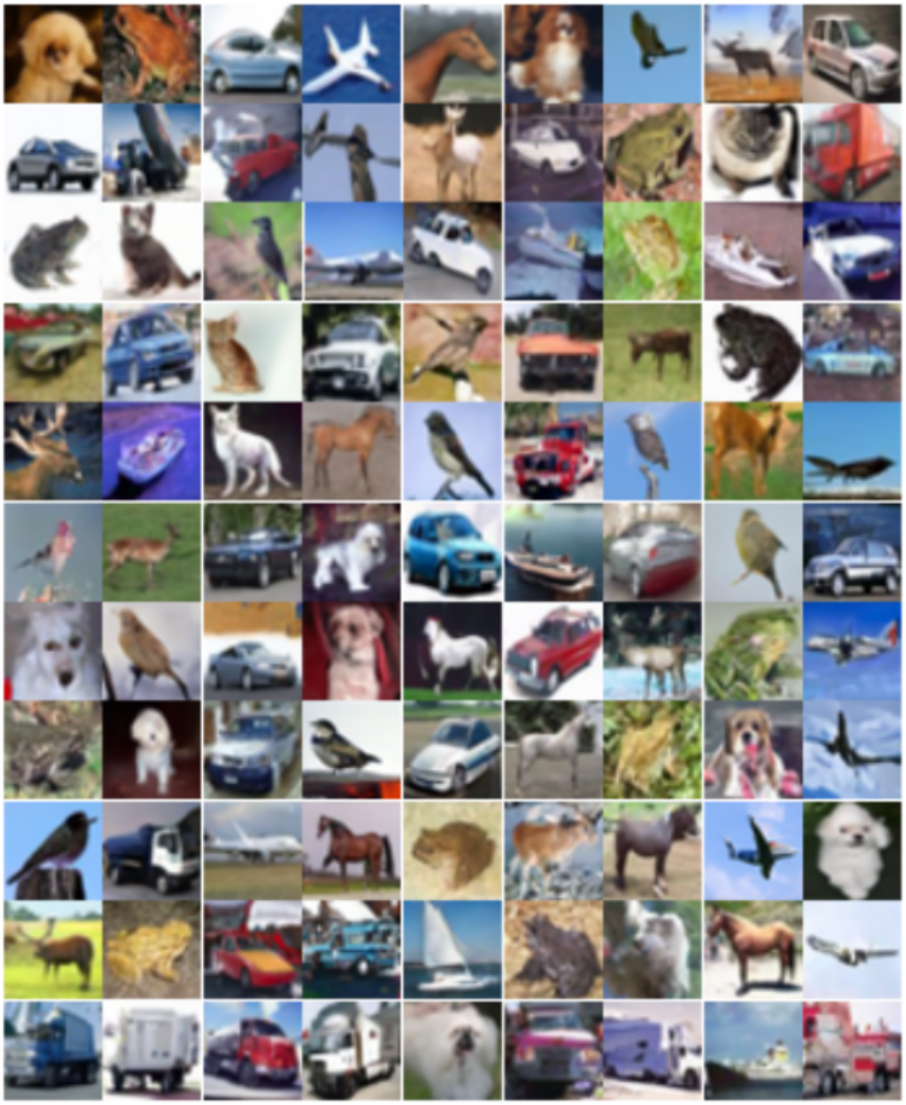

Fig. 12: Generated images from our model trained on CIFAR-10. Note the diversity of the various classes.

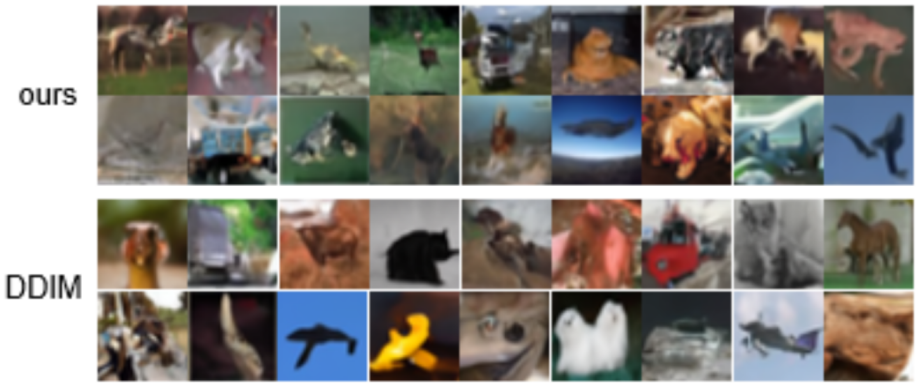

Fig. 13: Failure cases of DDIM and ours on CIFAR-10.