Group Project

Deep Learning KU, WS 2024/25

Team Members		
Last name	First name	Matriculation Number
Barić	Dorian	12342213
Pfleger	Marco	00873295
Seidlhofer	Tobias	11813712
Tschulik	Maximilian	01431103

1 Introduction

In this project we trained a diffusion model on an Image dataset. Given a set of observed pixels x_{obs}, we wish to draw a sample for the set of unobserved pixels $x_{unobs} \sim p_\Theta(x_{unobs}|x_{obs})$. Unfortunately we misinterpreted the task description and did not train a generative prior but directly trained our model to condition on observed pixels. In our case, we examined generating a logical continuation of an image given its upper half. Furthermore the goal was to generate a large variety of faces which look plausible. It was a requirement for this task to choose a colored image dataset. As a result, the chosen Image dataset was the CelebA datset. It consists of 202,599 images with a resolution of 178 x 218. This data set was chosen because it has a large number of images on which we could train. Furthermore, faces have a consistent structure (symmetry, nose position, etc.) and alignment, which makes it easier for models such as this to complete missing regions.

2 Methods

2.1 Model architecture

Given a data point $x_0 \sim$ q(x). A diffusion model consists of a forward process that gradually adds noise in T steps according to a known variance scheduler $\beta_1 < \beta_2 < \beta_T$ to an image until the signal is destroyed. We chose T = 1000 and a linear schedule from $\beta_1 = 10^{-4}$ to $\beta_T = 0.02$.

$$q(\mathbf{x}_t \mid \mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\,\mathbf{x}_{t-1}, \beta_t \mathbf{I})$$

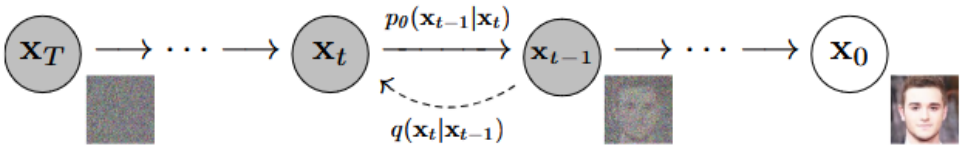


Figure 1: Forward process illustration [?]

The second step is to reverse the forward process step by step, by removing the added noise. This is done via an U-Net. We chose to modify the U-Net that was presented at the KU of the Deep learning course. In general an U-Net has the form as seen in Figure 2. The U-Net consists of a contracting and an expansive path. The contracting path consists of encoder layers. The contracting path consists of encoder layers which find relevant features in the image, at the initial steps the focus is on simple features like detection of edges and lighting while further layers look to extract more complex features like, in our example, facial details (the noses, eyes...). In the encoder layers the spatial resolution of the images is reduced and feature maps are produced that encode the different features noticed in images. The bottom of the U-Net is called the bottleneck layer. The expansive path consists of decoding layers that decode the encoded data and locate features while preserving the input's spatial resolution. Skip connections, direct connections between the encoder and the decoder layer, help the decode layers to preserve spatial information of the contracting path.

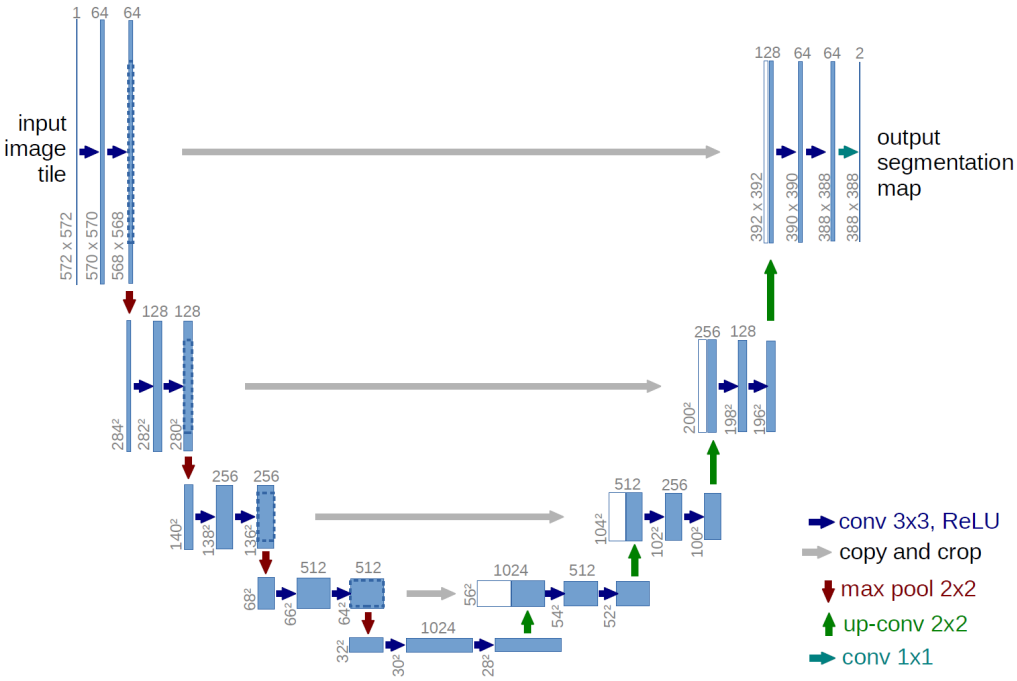


Figure 2: Example of a U-Net structure [4]

Our U-Net consists of 4 en-/decoder steps, each of those en-/decoder has 2 ResNet blocks. To downsample a 2D convolution is used with stride = 2. In the bottleneck, we used 3 RAA blocks [5] which consist of a ResNet block and a self-attention block each. The time embedding is added in the bottleneck layer. Our ResNet block consists as seen in Figure 3 of a first 2D convolution, then a 2-D batch normalization process, then a Leaky ReLu function, after that a 2-D convolution, and last a 2-D batch normalization. It also has a skip connection directly added at the output, this bypasses the previous mentioned blocks. The ResNet block was used to enhance network capacity but keeping computational cost efficient. The 3 RAA blocks were only added in the bottleneck of the U-Net to keep the model complexity in reasonable level. The self-attention [1] at this low resolution helps to capture relationships between distant elements in the image. The input to the U-Net consists of a concatenation the clean top half of an image and the noisy bottom half image at time t, both of which have the same size as the image half. The convolution channels of the convolution unit are set to 64, 128, 256, 512 respectively. The final model consisted of 37 million parameters.

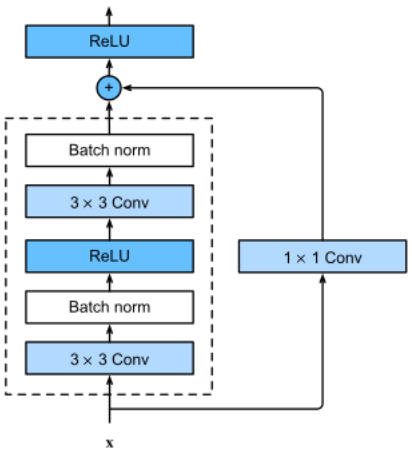


Figure 3: Residual block used in our U-Net [1]

[1] Code for self-attention layer from:`https://github.com/YigitEkin/Attention-Unet-Pytorch/blob/main/modules/modules.py`

2.2 Loss function

The loss function that is optimized is given as:

$$L_{\text{simple}}(\theta) := \mathbb{E}_{t,x_0,\epsilon}\left[\left\|\epsilon - \epsilon_\theta\left(\sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, t\right)\right\|^2\right] \tag{1}$$

ϵ is the true random noise sampled from a Gaussian distribution added to the image, and ϵ_θ is the model's (U-Net's) output, prediction of noise that was iteratively added to the image in the forward diffusion process. $\sqrt{\bar{\alpha}_t}x_0$ represents the signal from the original image x_0 at the start of the forward diffusion process scaled down by a factor of $\sqrt{\bar{\alpha}_t}$, which is the product of all α values acting as noise schedulers, controlling how much of the original signal is still present at time step t.

In the early stages of the forward process ($t \approx 1$), $\bar{\alpha}_t$ is close to 1 so the structure of the original image is going to dominate the signal. At later time steps ($t \approx T$), the pure Gaussian noise will cover a significant portion of the signal leading to the images being nearly non-discernible. The $\sqrt{1-\bar{\alpha}_t}\epsilon$ term describes the noise added to the image at time step t.

The values of scaling factors of the input image $\sqrt{\bar{\alpha}_t}$ and $\sqrt{1-\bar{\alpha}_t}$ through time steps in the diffusion process can be seen in Figure 4. As mentioned, ϵ_θ is the network's estimate of the noise added to the image x_0 given a noisy image and a time step t and like discussed by Ho *et al.* [3] the parametrization of μ_θ is a model that predicts $\widetilde{\mu}_t$ which is the forward process mean. This is expressed by the formula:

$$\mu_\theta(\mathbf{x}_t, t) = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(\mathbf{x}_t, t)\right) \tag{2}$$

The α values allow the computation of the image in any time step of the forward process (which is fixed and known) through:

$$x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon. \tag{3}$$

This loss function's goal is to train the model ϵ_θ to predict the noise ϵ that was added to an image at time step t in the forward diffusion process, so it can effectively remove the noise in the reverse diffusion process. It does so by calculating (and minimizing) the mean squared error between the random Gaussian noise added in the forward process ϵ and the model's prediction of the added noise ϵ_θ.

In this project's context, the ϵ_θ model takes as input the concatenation of the noisy bottom half of the original image and the "clean" top half, as well as the time step of the given state as seen in the code fragment below. It represents the Python implementation of the loss function defined in 1.

```
loss = F.mse_loss(eps, self.eps_model(x=torch.cat([bottom_half_noisy,
    top_half_clean], dim=1), t=t/self.T))
```

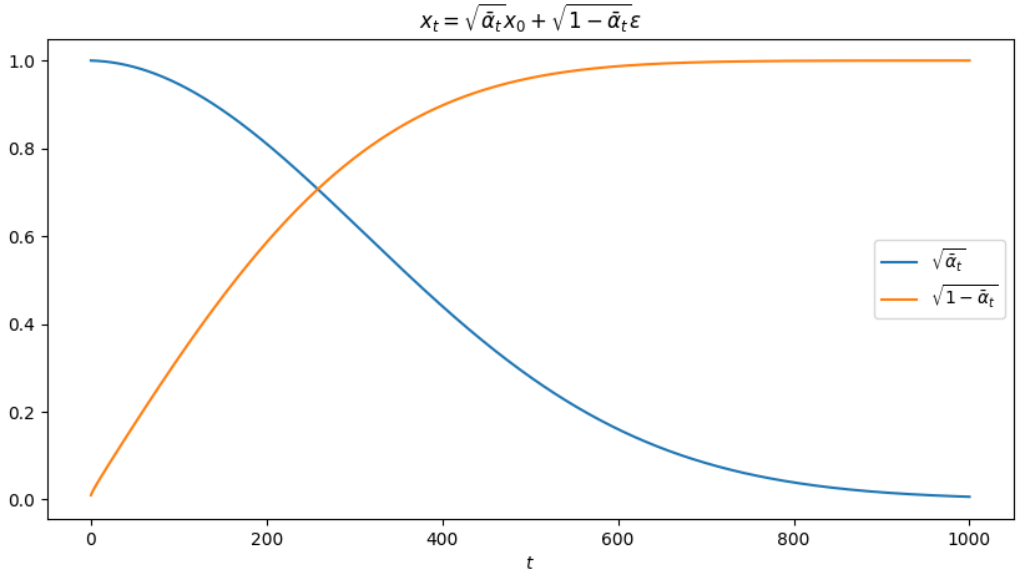
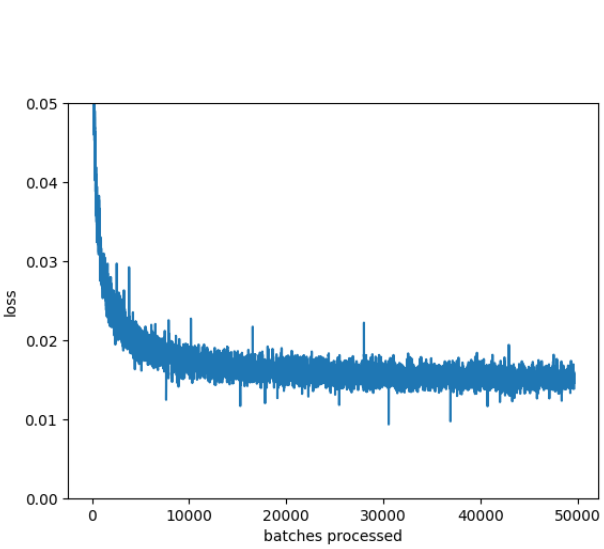


Figure 4: Scaling factors of the original signal and the noise through time steps t

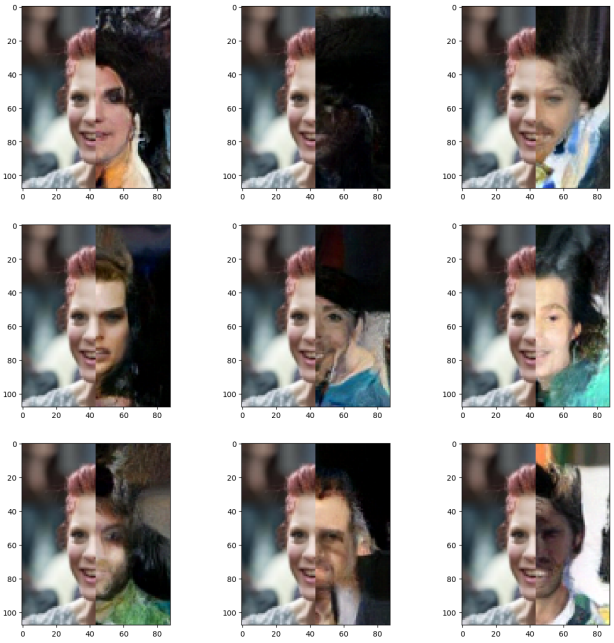
3 Results

3.1 Model Selection and Evaluation

Before selecting the final model described in Section 2.1, we conducted multiple experiments with different architectures and training hyperparameters. The minimal DDPM implementation presented in the KU lecture on U-Nets and diffusion models served as our starting point. This unconditional diffusion model was designed to generate 28×28 gray-scale images of digits and was trained on the *Modified National Institute of Standards and Technology* (MNIST) dataset. As a first baseline, we conditioned the model by concatenating the left side of an RGB image from the CelebA dataset to the input tensor. Using the right side of the image from the dataset as the label, we trained the model with 4.50M parameters for 40 epochs with a constant learning rate of 2×10^{-4} on a *NVIDIA RTX 3090 Ti* GPU. All trainings were executed with a batch size of 128 and using the AdamW optimizer. An image size of 108×88 was chosen to balance performance and image quality. Figure 5 visualizes the loss over iterations and shows some examples of right-half-images sampled from this initial model.



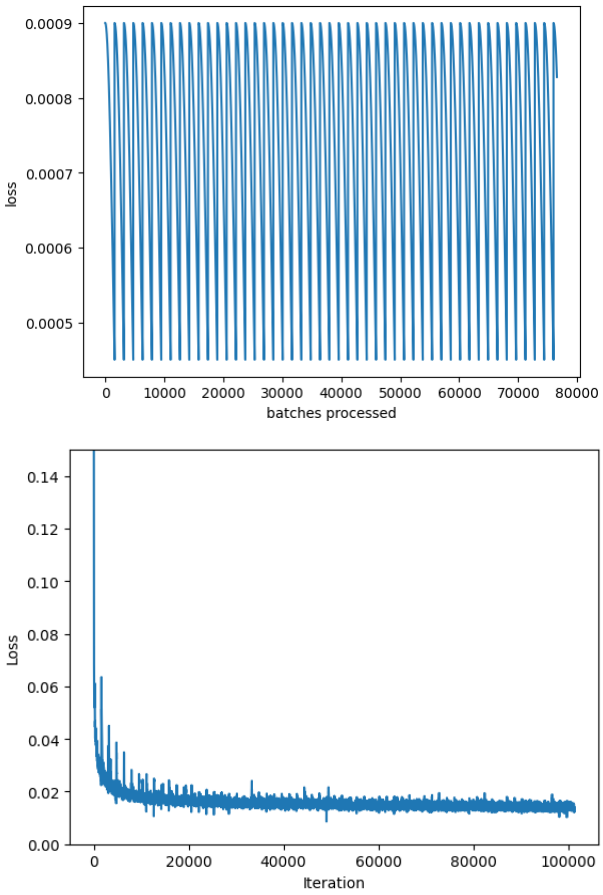
(a) Loss over iterations



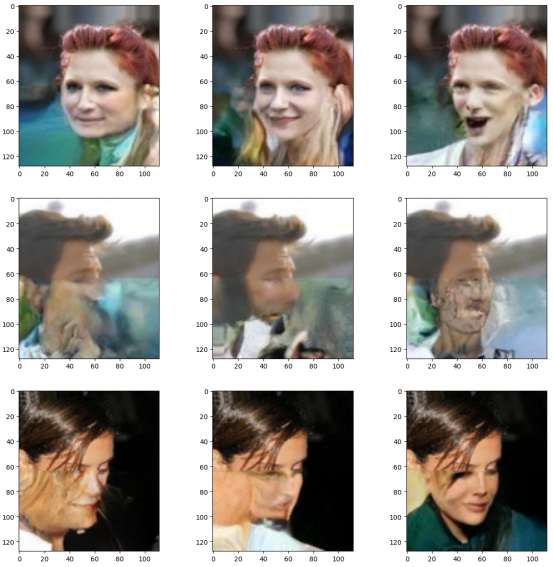
(b) Samples

Figure 5: Training results of the first baseline model with 4.50M parameters.

The small baseline model, while capable of generating face-like structures, did not meet our expectations regarding the quality of the output images. Considering the small number of parameters, we concluded, that the model size was insufficient to generate high-quality results. This conclusion was supported by the stagnating loss, however, further experiments showed that the loss is not a sufficient indicator of the quality of the samples, as different models with the same loss sometimes create very different output qualities. To increase the model capacity, another en-/decoder step was added to the U-Net, and the `DoubleConv` blocks were extended to `TripleConv` blocks, by adding another convolution and batch normalization layer as well as a Leaky ReLu activation function. Additionally, the time step embedding was added in the latent space instead of concatenating it to the input, by generating a vector with the same number of channels as the tensor in the bottleneck and extending each of its elements to the resolution of the latent space. This decision was made to match the DDPM paper more closely and to make the *Multi-Layer-Perceptron* (MLP) generating the time embedding more efficient: instead of having to compute a 108×44 tensor to add to the input, this approach only requires a single 512-dimensional vector. To make the output more interesting, we adapted the model such that it conditions on the top half of the image, generating the bottom half during sampling. The extended model has 24M parameters and was trained using a cosine annealing learning rate scheduler with warm restarts. The initial learning rate was set to 9×10^{-4} and supposed to fall to zero throughout a period of two epochs before restarting. At each restart, the number of epochs between restarts should double, to increase the interval over time. Due to an error in our implementation, the learning rate scheduler did not work as intended and instead restarted after each epoch. Because the training was stable, we decided to continue with the faulty scheduler. Figure 6 shows the loss and learning rate over iterations and some exemplary samples from the trained model.



(a) Learning rate and loss over iterations



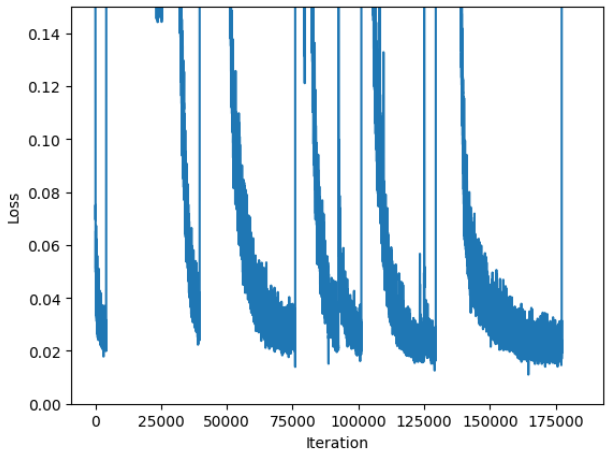
(b) Samples

Figure 6: Training results of the extended model with 24M parameters.

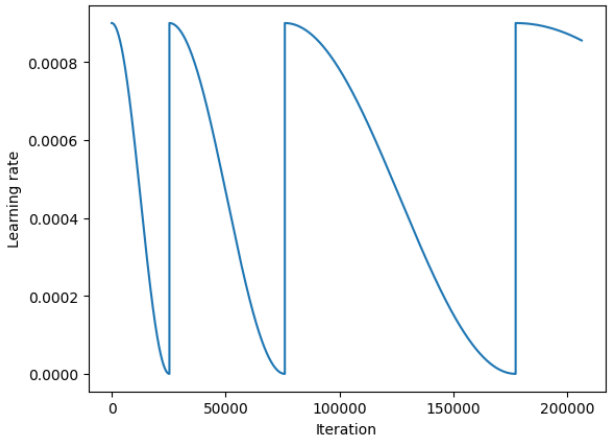
The extended model outperforms the baseline models concerning the quality of the inpainted image half. We observe, that the top and bottom half match better and some details like hair strands from the top half are continued in the bottom half.

To further enhance the inpainting quality, we decided to introduce more parameters. Since adding too many convolution layers to the individual en/-decoder steps can make the propagation of the gradient through the network unstable, we decided to instead replace the `TripleConv` module from before with two *ResNet* blocks, originally introduced by He *et al.* [2]. The ResNet blocks are very similar to the `DoubleConv` blocks but add a residual connection between the input and the output, before applying the Leaky ReLu activation function. Because the number of channels going into the block does not match the number of output channels, a 1×1 convolution was added to the residual connection to resize the input to the output's dimension. We additionally added self-attention modules to each en-/decoder step in the bottleneck, to increase the model's capability to process global information.

After fixing the learning rate scheduler, the model was trained with the same training hyperparameters as the previous model. A few epochs into the training, it became evident, that the learning process was highly unstable with frequent spikes in the loss as can be seen in Figure 7. It should be mentioned that we only used a batch size of 16 for this training which could have also contributed to this behavior. In a first attempt to fix this issue, the self-attention mechanisms were removed from all layers but the bottleneck, to more closely resemble the model structure of Song *et al.* [5] who added RAA blocks (residual block + attention block) only at the bottleneck resolution. This change improved stability but did not completely mitigate the problem. Further investigation showed that the spikes primarily occur during learning rate restarts as depicted in Figure 8, indicating an excessively high learning rate.

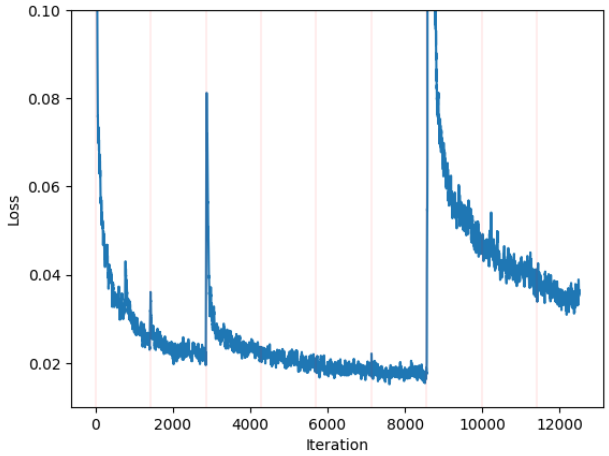


(a) Loss over iterations

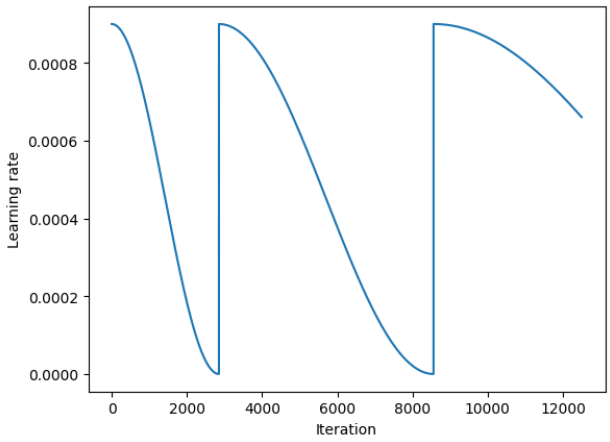


(b) Learning rate over iterations

Figure 7: Training results of the further extended model with ResNet blocks and self-attention blocks at every resolution (31M parameters).



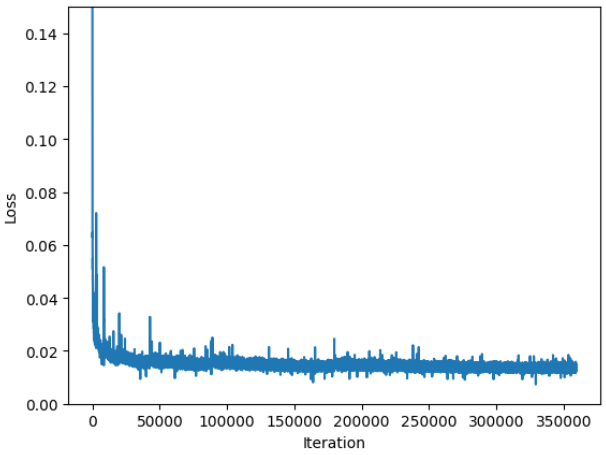
(a) Loss over iterations



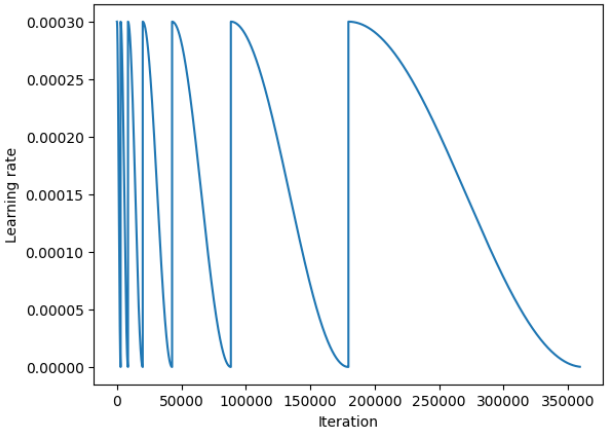
(b) Learning rate over iterations

Figure 8: Training results of the final model with an excessively high learning rate (37M parameters).

After reducing the max learning rate to 3×10^{-4}, the training process became substantially more stable. There were still some small spikes during learning rate restarts, but they no longer negatively affected the training. Additionally, the magnitude of the loss spikes was reduced after the first few restarts, suggesting that the model became more stable as depicted in Figure 9.



(a) Loss over iterations



(b) Learning rate over iterations

Figure 9: Training results of the final model with a smaller learning rate (37M parameters).

After training for 252 epochs, which with the available computing power lasted around 25 hours, the final model is capable of performing high-quality image inpainting on images from the CelebA dataset. This is discussed in the next section containing samples from our final model.

3.2 Sampling Experiments

We have used the following formula for sampling introduced by Ho *et al.* [3]:

$$x_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \epsilon_\theta(x_t, t) \right) + \sigma_t z$$

and chose $\sigma_t = \sqrt{\beta_t}$ as stated in their publication.

Figure 10 shows samples from our final model. The first column of this grid shows the original images. Our model generates a wide variety of realistic looking continuations of the given top half of a face image. Also, even though the depicted images where contained in the training dataset, our model generates different faces than the real one. Compared to the previous model the final model qualitatively generates better samples with less artifacts and stronger conditioning on the top half. Especially the continuation of hair, ears and noses looks more plausible and realistic.

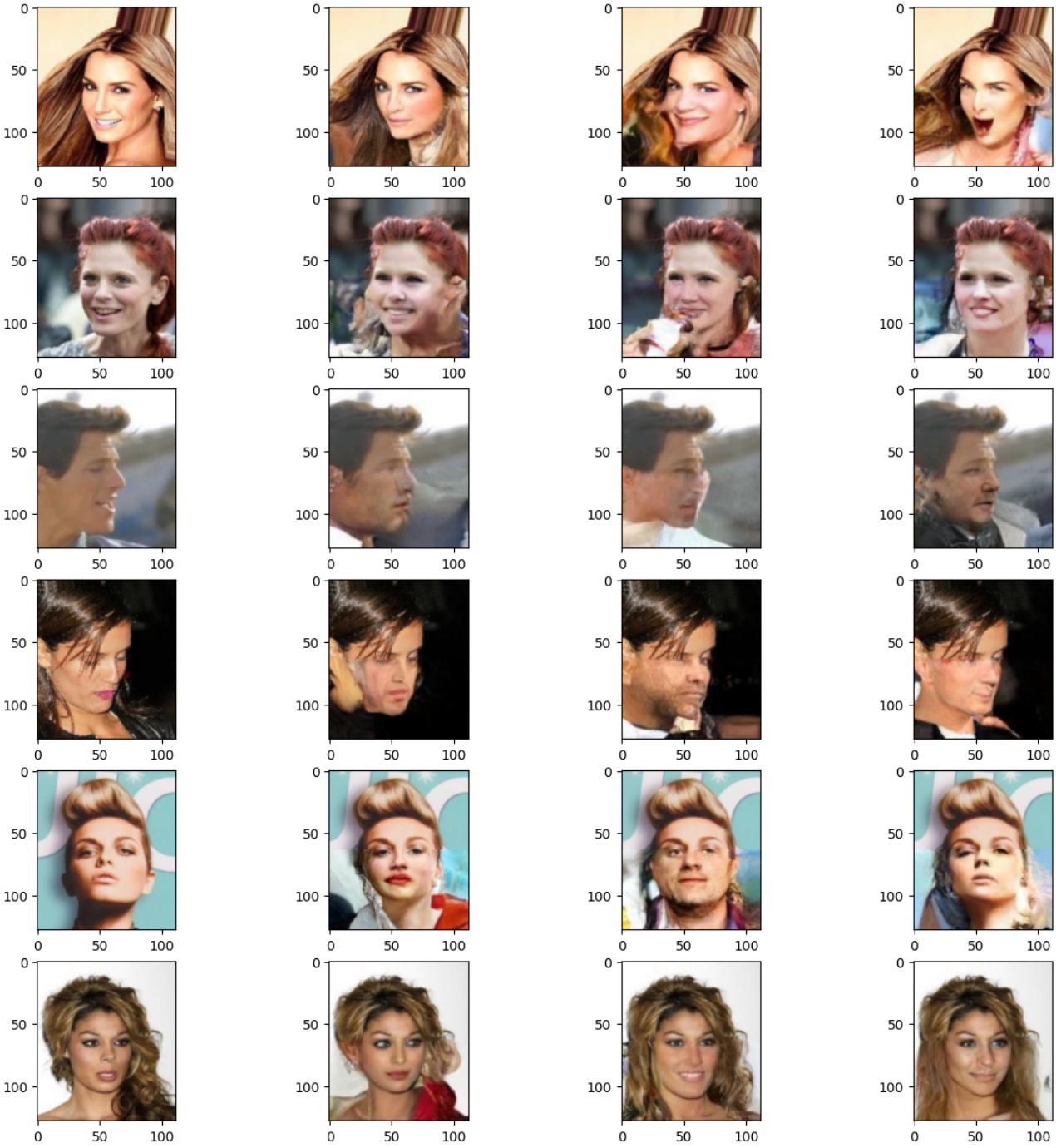
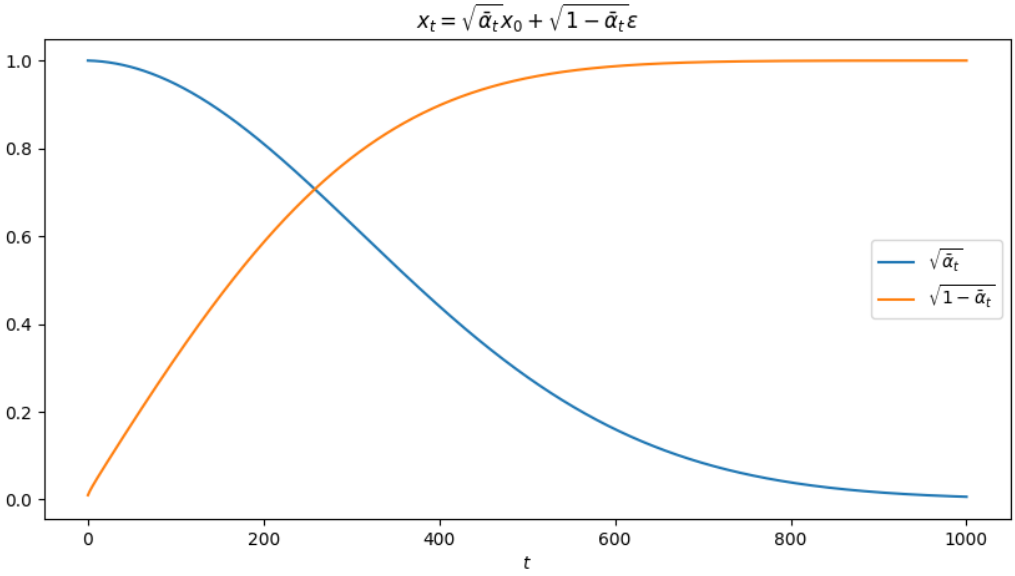


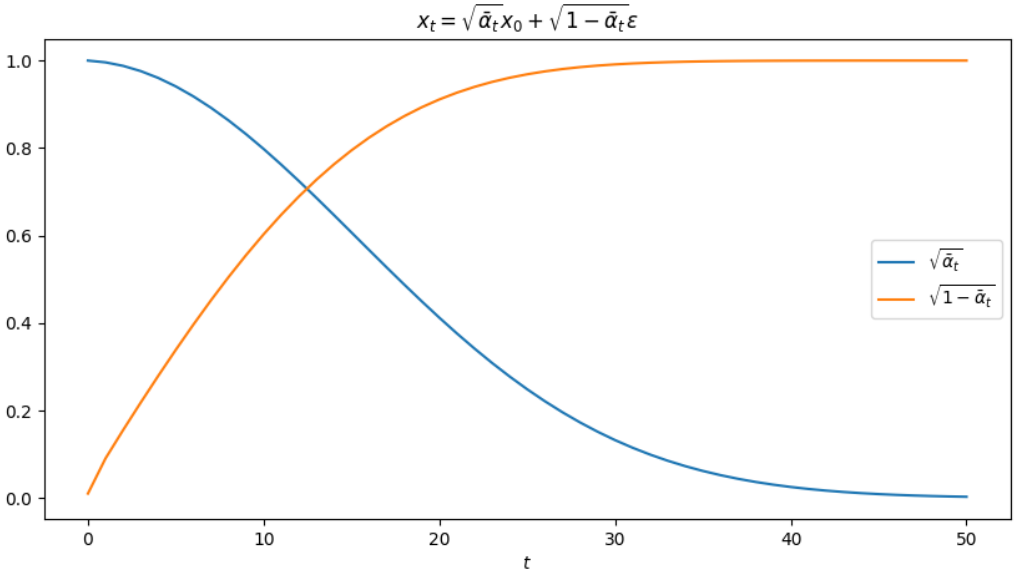
Figure 10: Samples from our final model

3.2.1 Influence of reverse diffusion steps

We also experimented with the step size during the reverse diffusion process, basically skipping steps in order to speed up the sampling. Our approach was to create a new DDPM schedule (new $\beta_t, \alpha_t, \bar{\alpha}_t$) with a new number of diffusion steps $T' = \frac{T}{step\_size}$. In order to approximate the added cumulative noise correctly with less diffusion steps while still linearly interpolating between β_1 and β_T, we had to adapt β_T. We kept β_1 the same and changed $\beta'_T = \beta_1 + (\beta_T - \beta_1) * step\_size$. With this adaptation we could approximate the original DDPM schedule with less steps which can be seen in Figure 11.



(a) T=1000



(b) T=50 (step_size=20)

Figure 11: DDPM schedule for T=1000 and T=50.

Figures 12, 13, and 14 show samples for different step sizes or number of diffusion steps. The corresponding original image can be seen in the first column. We observed that for larger steps more noise is introduced to the sampled lower half of the image. This might be due to an inaccurate approximation of the DDPM schedule. Furthermore, we got the impression that for larger steps the samples look more similar to each other. Especially when comparing Figure 10 with Figure 14 it can be seen that for T=1000 there is more variety in the sampled faces (open/closed mouth, different hair length, lipstick, smiling etc.). This makes intuitive sense since for every reverse diffusion step the image gets the chance to evolve in another direction leading to more variety when sampling for more steps.

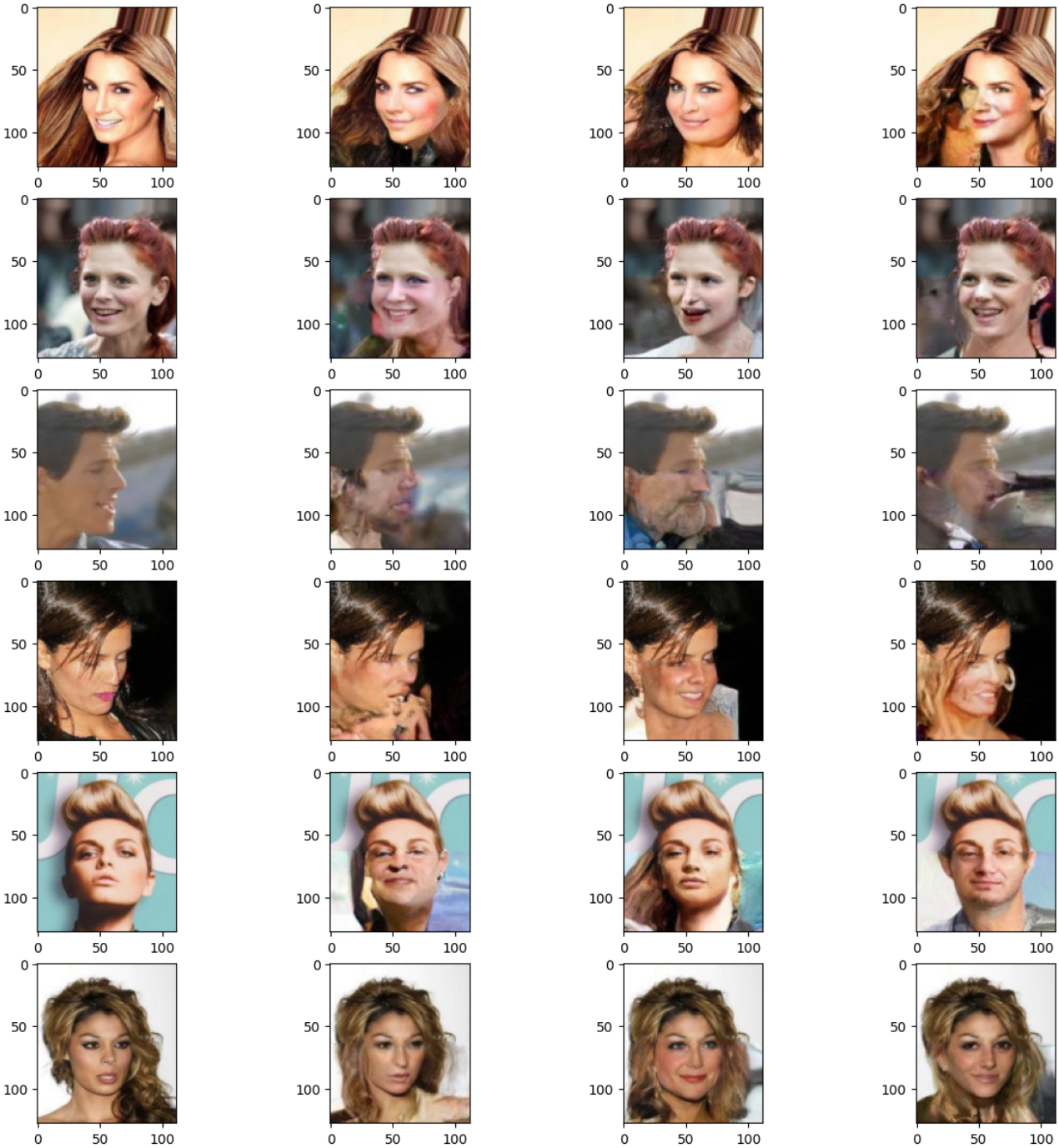


Figure 12: step_size=2 (500 steps)

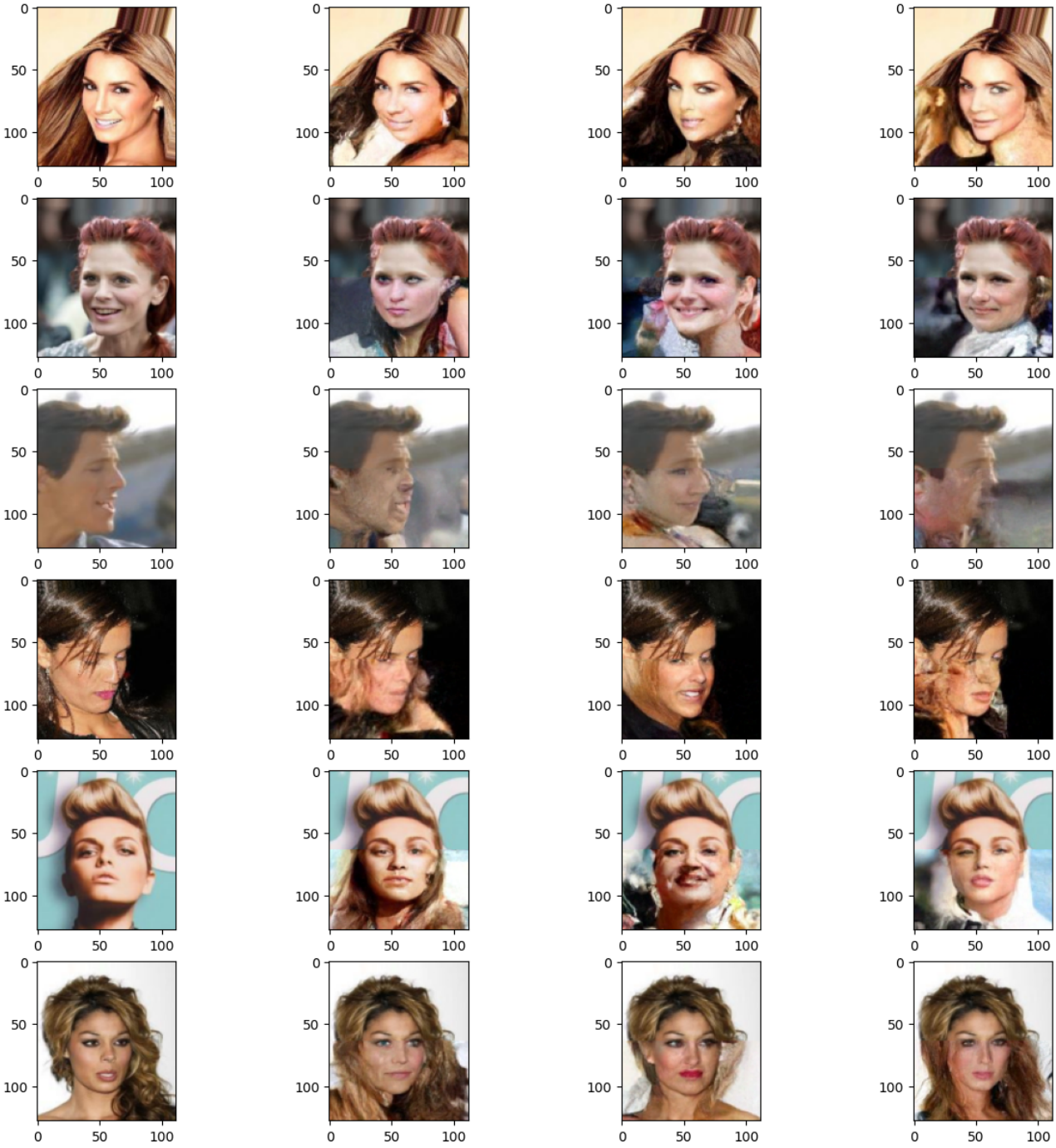


Figure 13: step_size=10 (100 steps)

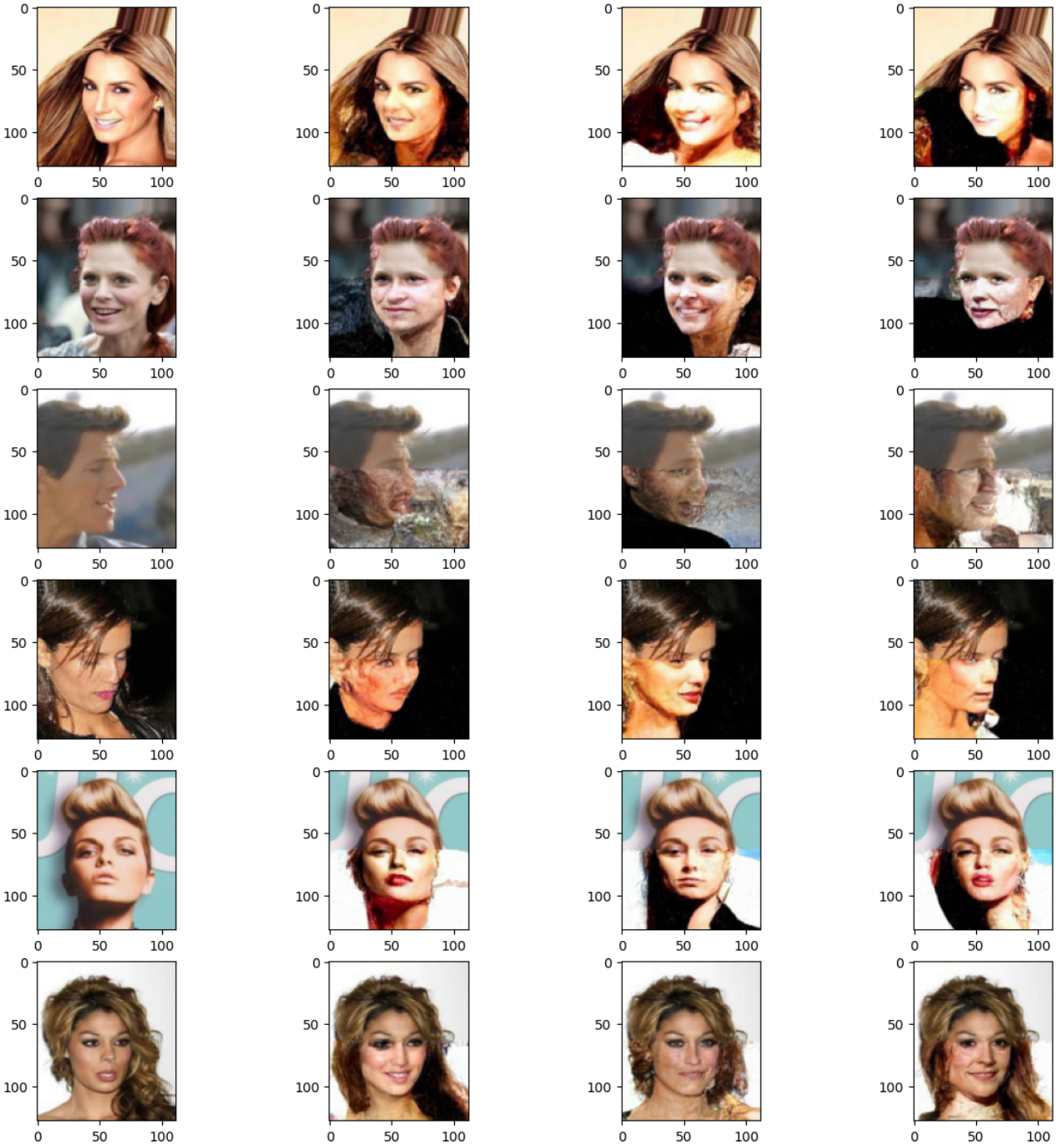


Figure 14: step_size=40 (25 steps)

3.2.2 Influence of noise scaling during sampling

We did some further experiments by changing the noise schedule in the sampling process and setting the scaling factor $\sigma_t^2 = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$.

Although for T=1000 there is no distinguishable difference between $\sigma_t^2 = \beta_t$ and $\sigma_t^2 = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$, it is apparent that the second method introduces less noise when taking larger steps during sampling. The faces depicted in Figure 16 and Figure 17 show a smoother continuation of the faces and less noise. The underlying facial structure seems to be the same for both methods.

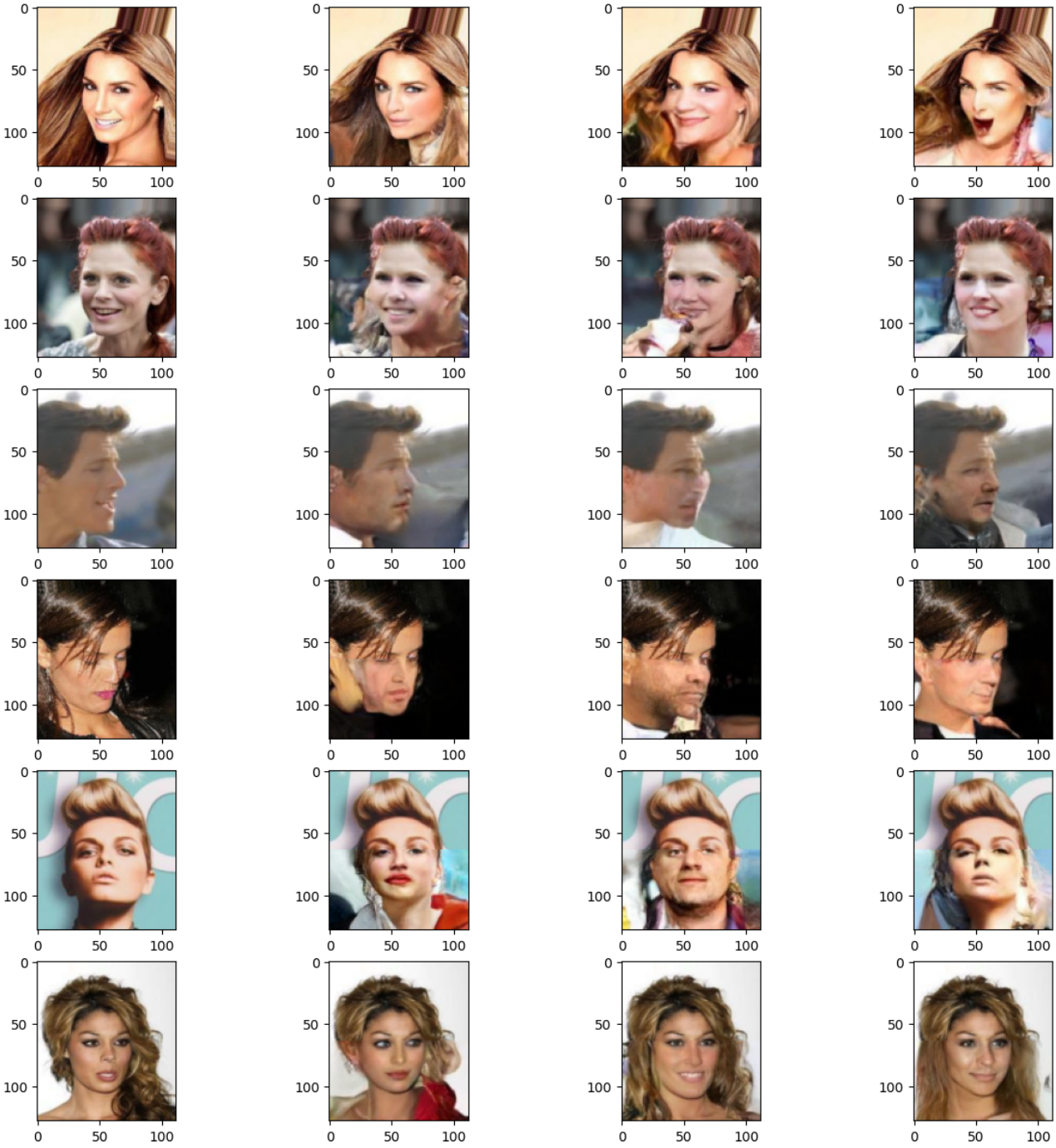


Figure 15: step_size=1

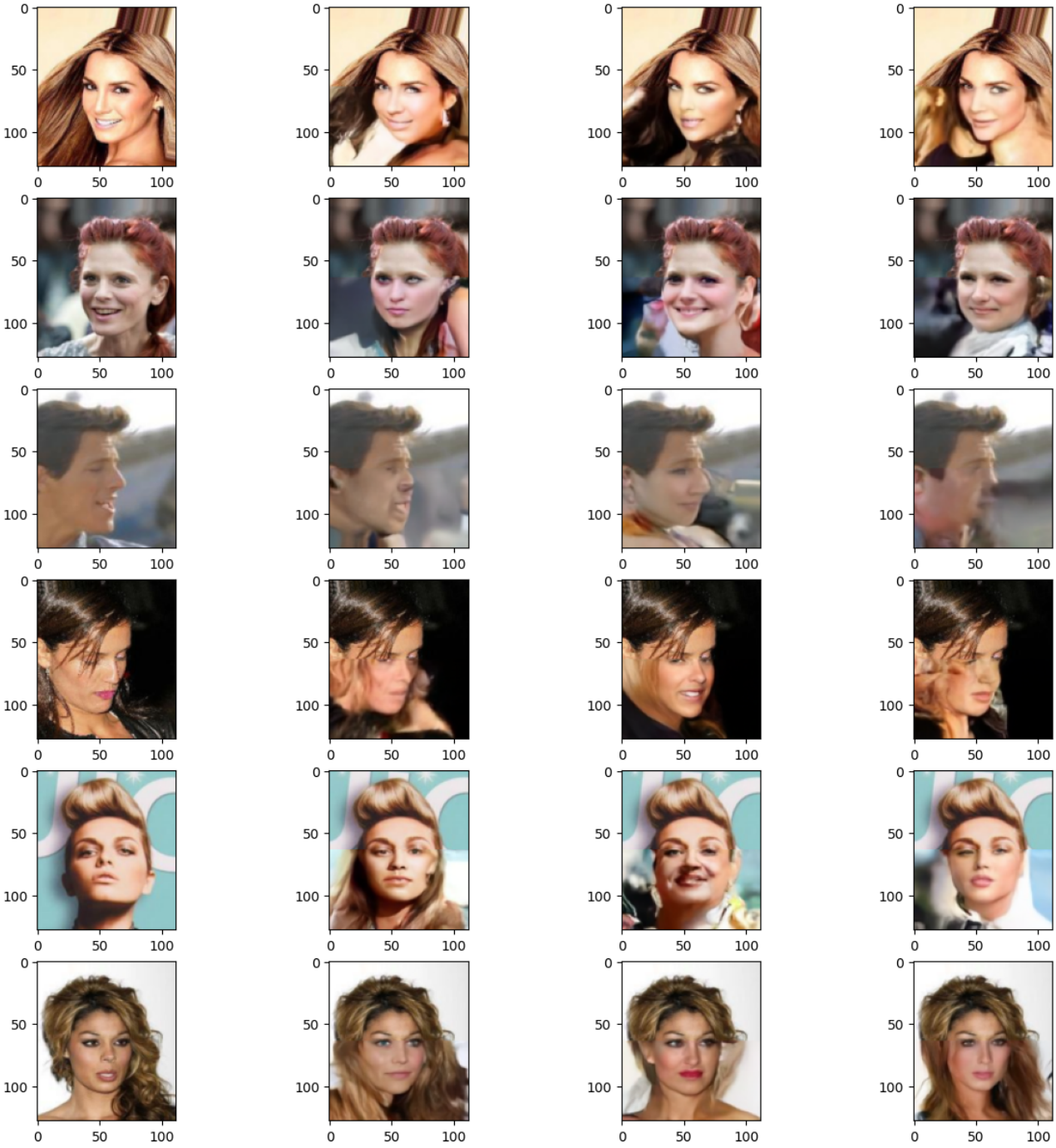


Figure 16: step_size=10 (100 steps)

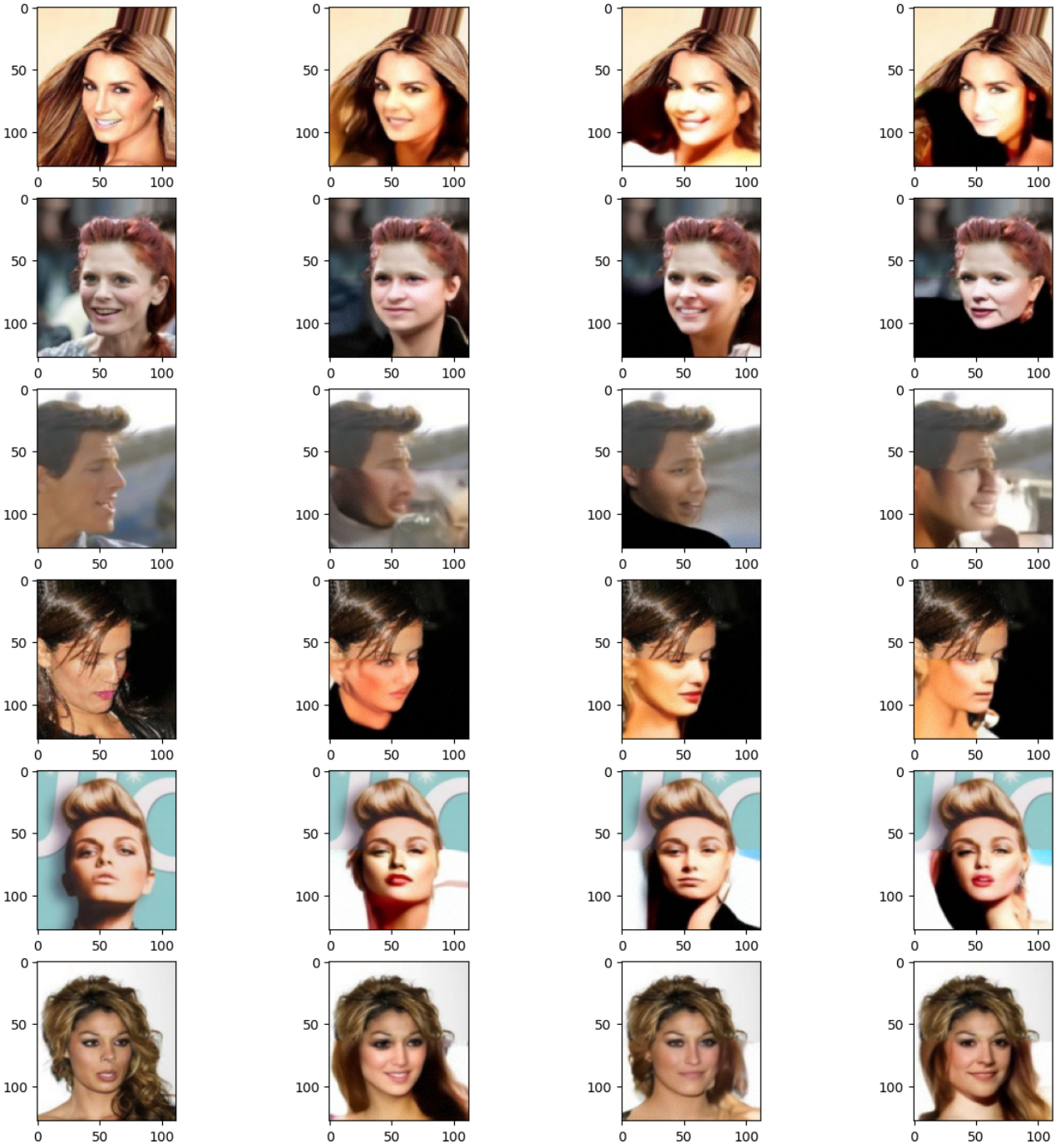


Figure 17: step_size=40 (25 steps)

3.2.3 Conditioning on unseen half-faces

Our model seems to generalize well to unseen data. Upon qualitative inspection of the images depicted in Figure 18, we concluded that our model perform similarly well when sampling conditioned on unseen data.

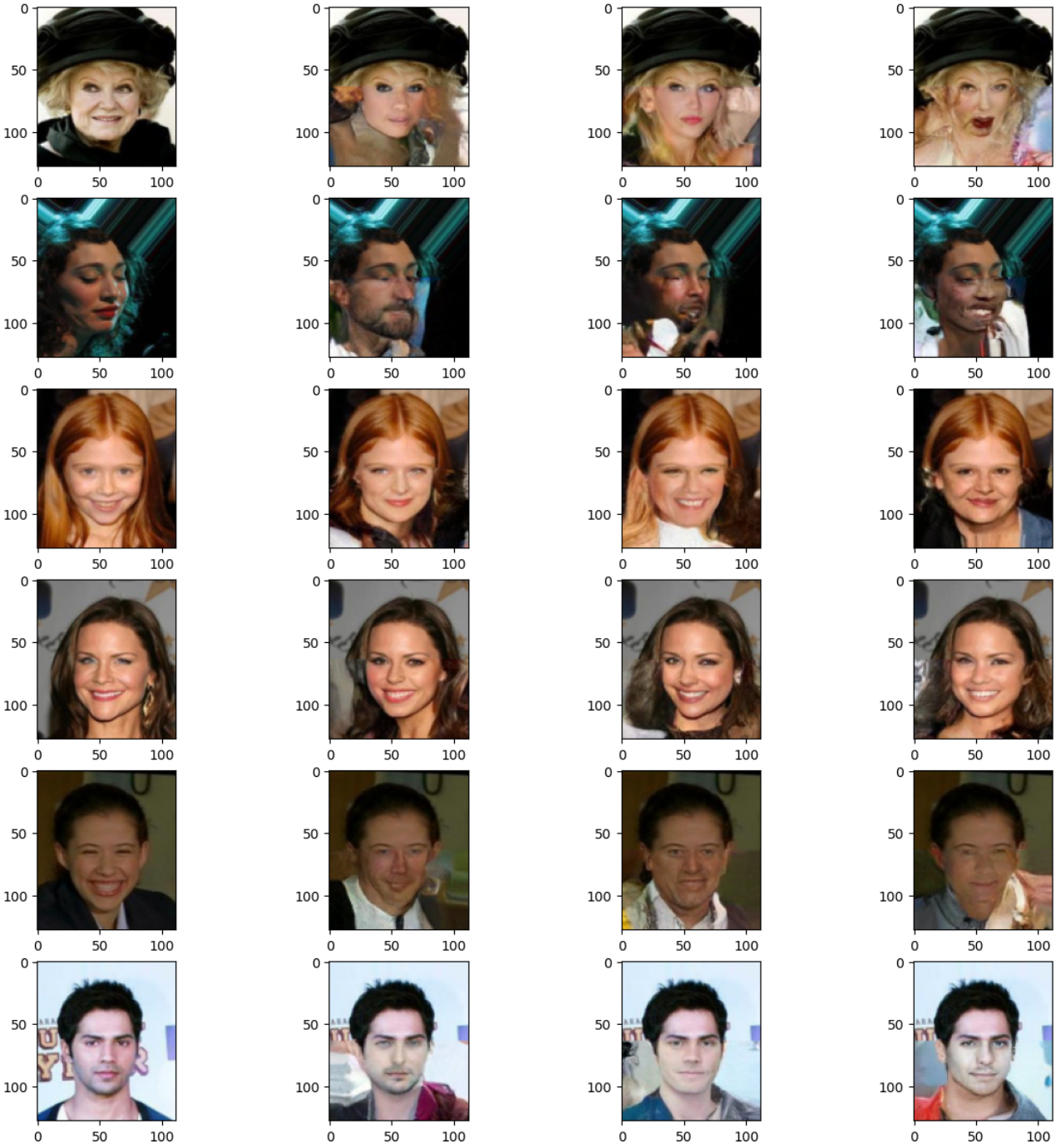


Figure 18: Sampling conditioned on unseen half-faces

4 Discussion and Conclusions

As a result of our investigation, adding more parameters significantly improved the performance of our model. However, this resulted in a longer training time. Looking at the loss of Figure 6a and Figure 11a, one can see that it does not change much between the two different models. As a result we concluded to visually asses the performance of our models. We came to the conclusion that our final model significantly outperforms the previous models. Our model struggles when a face is photographed from the side, as shown in the result figures. This could be due to the limited number of side-profile photos in the dataset. It also does not seem to recognize the facial features of a child as seen in the result figures which leads to the inpainting of mature faces onto what is initially a child in the original picture. This could be because there is only a small number of children photos in the dataset. Also noticed is when the model has to generate a continuation of the hair of a person whose hair is,

say, brown or black, our model tends to produce the continuation of an image of blonde hair. These observations could indicate a bias in the dataset towards these features. Furthermore, we noticed the model also struggles with generating the background of the images since there is more variety and ambiguity towards the possible backgrounds, whereas the faces all share similar features like the facial contours, eyes, noses etc. While diffusion models perform the task of generating quality images very well and provide a wide variance of images they generate, the area of struggle is the time it takes to sample the images.

References

[1] Mu Li Aston Zhang, Zachary C. Lipton and Alexander J. Smola. *Dive into Deep Learning.* d2l.ai, 0.16.1 edition, 2021.

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[3] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 6840–6851. Curran Associates, Inc., 2020.

[4] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In Nassir Navab, Joachim Hornegger, William M. Wells, and Alejandro F. Frangi, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pages 234–241, Cham, 2015. Springer International Publishing.

[5] Molei Song, Hongzhen Shi, Dan Xu, and Kangjian He. Facial image inpainting using condition guided diffusion models. In Zhaohui Wang, Jindong Tian, and Mrinal Mandal, editors, *Sixteenth International Conference on Digital Image Processing (ICDIP 2024)*, volume 13274 of *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, page 132741A, October 2024.