One Model to Reconstruct Them All: A Novel Way to Use the Stochastic Noise in StyleGAN

Joseph Bethge[*], Christian Bartz[*], Haojin Yang, Christoph Meinel
Hasso Plattner Insititute
University of Potsdam
{firstname.lastname}@hpi.de

Abstract

Generative Adversarial Networks (GANs) have achieved state-of-the-art performance for several image generation and manipulation tasks. Different works have improved the limited understanding of the latent space of GANs by embedding images into specific GAN architectures to reconstruct the original images. We present a novel StyleGAN-based autoencoder architecture, which can reconstruct images with very high quality across several data domains. We demonstrate a previously unknown grade of generalizablility by training the encoder and decoder independently and on different datasets. Furthermore, we provide new insights about the significance and capabilities of noise inputs of the well-known StyleGAN architecture. Our proposed architecture can handle up to 40 images per second on a single GPU, which is approximately 28× faster than previous approaches. Finally, our model also shows promising results, when compared to the state-of-the-art on the image denoising task, although it was not explicitly designed for this task.

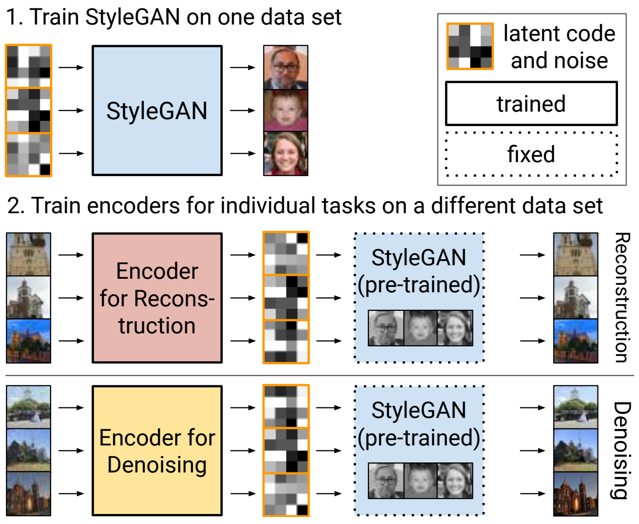


Figure 1: Our basic approach. A StyleGAN generator is trained on a dataset, *e.g*. FFHQ [18]. Afterwards, we train the encoder part of an autoencoder for reconstruction and denoising tasks on a *different* dataset, without updating the pre-trained StyleGAN which is used as a decoder.

1. Introduction

Generative Adversarial Networks (GANs) are the current state-of-the-art models in the area of unconditional and conditional image generation. They are applied in various computer vision areas, *e.g*., image-to-image-translation [15, 45, 14, 34], image superresolution [22, 38, 33], or unconditional generation of various image types [9, 26, 4, 17]. Over time, image quality, resolution, and realism of synthesized images was improved by a large margin [9, 26, 17, 18]. The StyleGAN architecture [18, 19] is one of the current state-of-the-art models for unconditional image generation. The architecture of StyleGAN with its projection into a semantically meaningful latent space $\mathcal{W}$ and the usage of noise inputs for stochastic variation not only allows to generate a diverse range of images but also enables meaningful image editing operations. Using recent GAN inversion techniques, it is also possible to perform similarly meaningful edit operations on embeddings of real images in the latent space [2, 1, 29]. We briefly analyze the related work in Section 2. However, to the best of our knowledge, previous work has not provided complete answers for the following open research questions in the domain of GAN inversion: (1) To what degree does the latent code influence the reconstruction result? (2) Can the stochastic noise provide more than stochastic variations of the generated image? (3) Can a generator model (*i.e*. StyleGAN) trained in one domain be used to effectively reconstruct images for a different domain? (4) Can an encoder model trained in one domain be effectively used reconstruct images for a different domain?

[*]Equal Contribution

In this work, we provide answers to all of these questions. We introduce a novel conditional GAN that uses the StyleGAN generator architecture and a ResNet-based encoder model for image reconstruction and present strategies to maximize the semantic meaning of the latent code in Section 3. During the training of our model, we use an off-the-shelf, pre-trained StyleGAN generator as a decoder and train an encoder, while leaving the pre-trained weights of the generator untouched (see Figure 1). We show in Section 4 that our approach is not only able to faithfully reconstruct input images from the domain the generator was trained on, but also from other domains. For example, we show that a StyleGAN generator pre-trained on the FFHQ dataset [18] and an encoder trained to reconstruct images from the FFHQ dataset is able to faithfully reconstruct images from several different LSUN datasets [37] (churches, cats, or bedrooms). Furthermore, we provide in-depth insights into how a pre-trained StyleGAN generator is able to reconstruct images from other data distributions. We conclude our work in Section 5. Overall, our contributions can be summarized as follows: (1) The first approach for faithful cross-domain image reconstruction, based on a fixed generator model, tested on a large variety of images from several domains. (2) Novel insights about the capabilities of noise inputs in StyleGAN. (3) A training scheme that increases the semantic meaning of the latent codes. (4) A fast method that allows embedding of up to 40 images per second on a single GPU (NVIDIA RTX 2080 TI), which is much faster than other recent GAN inversion models, *e.g.* by [11], which can process approximately 1.4 images per second. (5) A practical application of our model in the area of image denoising, where we match other state-of-the-art models. Our code and trained models are available online[1].

2. Related Work

GANs have first been proposed by Goodfellow *et al.* [9] in 2014. Since then, they have been improved through different measures, such as training at different scales [17], adding novel weight normalization techniques [26] or generating high-resolution images over a diverse set of classes [4].

A recent work by Karras *et al.* [18] proposes a novel architecture inspired by a work on neural style transfer [13]. They train their StyleGAN architecture on their collected FFHQ dataset containing images of human faces to generate high-quality, realistic images of human faces. Moreover, Karras *et al.* propose several improvements regarding architecture and normalization methods for StyleGAN in a more recent work [19]. In the following, we describe the previous art related to the two tasks covered in our work, image reconstruction and image denoising.

[1]`https://github.com/Bartzi/one-model-to-reconstruct-them-all`

2.1. Image Reconstruction (Embedding, Inversion)

Generative models usually operate on a latent code and/or random noise as an input to generate new images [9, 18, 19]. Previous work on GAN inversion attempts to understand and interpret the underlying mechanisms of GANs by embedding existing images into a GAN architecture. These works can be roughly divided into two categories.

On the one hand, a given image can be embedded into the latent space of a trained GAN on a per-image basis [2, 1, 44, 6]. These methods achieve very faithful reconstructions, but require optimization or training a model for each image making image embeddings of large-scale datasets infeasible.

On the other hand, there are works similar to the idea of autoencoder architectures (*e.g.* [21]) which learn an encoder network for embedding an image. This approach is used *e.g.* in [28, 3, 11]. It is computationally efficient, since the learned encoder can retrieve an encoding for a given image. However, learning a code with semantic meaning proves to be a challenge as stated by Zhu *et al.* [43].

2.2. Image Denoising

A typical application area for image reconstruction is image denoising, where the task is to remove noise from a given noisy image, to restore the original image. Here, we focus on image denoising techniques based on deep neural networks, for more detailed information about image denoising research, please refer to the following survey papers [8, 10].

Several neural-network-based image denoising systems have been proposed in the past [16, 5, 36, 40, 38, 41, 22, 39]. Some works have been trained for image denoising at fixed noise level [5], while others are able to denoise noisy images with various noise levels. Over time, the most common approach shifted from directly predicting the denoised image to predicting residual/noise images which are then subtracted from the input image, returning the denoised image [22, 39]. Based on this residual prediction strategy, further enhancements have been proposed. Zhang *et al.* propose multiple extensions, *i.e.* using multiple Convolutional Neural Networks (CNNs) based denoising networks and model based optimization [40], providing an extra noise level map as auxiliary input to the neural network [38, 41], or creating specific network architectures for iamge denoising [38].

3. Method

In this section, we describe the method that we are using for the reconstruction of arbitrary input images (not limited to the training domain) by using a generative model that has only been trained for unconditional generation in a certain domain, *e.g.*, face images (see Figure 1). In the following, we first introduce our overall architecture of the autoen-

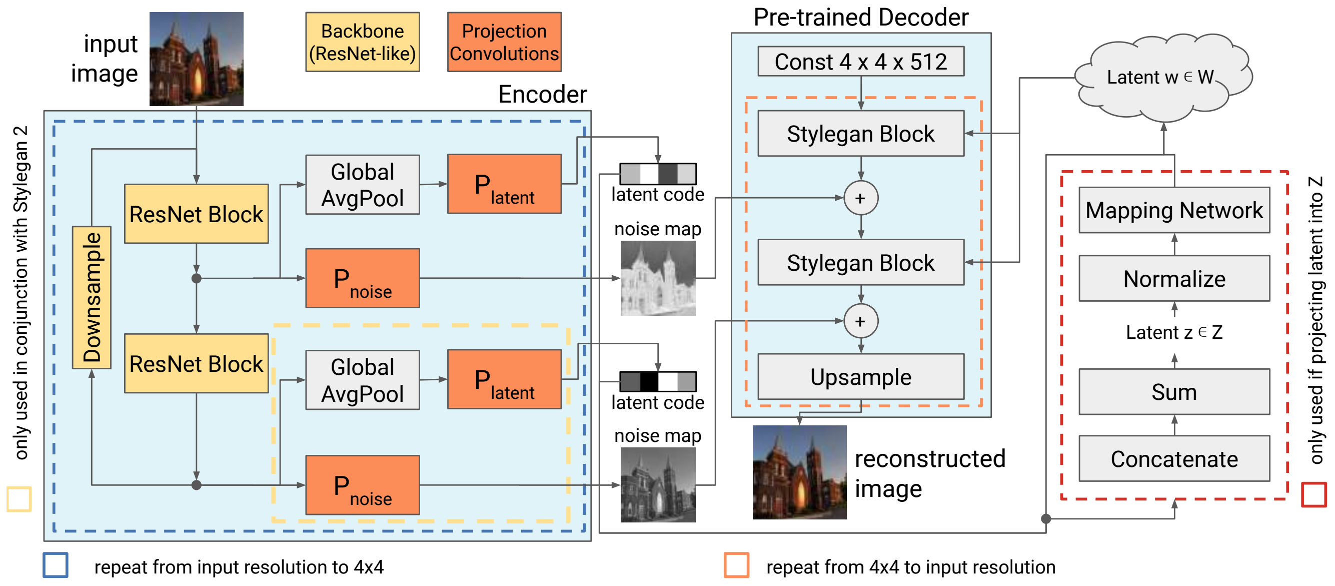


Figure 2: An overview of the structure of our proposed autoencoder. The autoencoder consists of an encoder and a pre-trained StyleGAN 1 or 2 as decoder. Our encoder consists of multiple ResNet blocks, each followed by convolutional layers that predict part of the latent code or a noise map. Each output of the encoder is then used by the pre-trained StyleGAN to reconstruct the input image. As indicated in the figure, some parts of the model can be switched on or off, depending on the StyleGAN version or latent projection strategy we use.

coder that we use for image reconstruction (see Figure 2), split into the decoder (Section 3.1) and the encoder (Section 3.2). During our experiments, we discovered how the encoder uses the noise maps instead of the latent code for image reconstruction (more details in Section 4.2). Therefore, we further developed two methods to reduce this behavior and instead maximize the semantic meaning of the latent code (see Section 3.3). Finally, we describe the training details used in our experiments (see Section 3.4).

3.1. Decoder Architecture

For our experiments, we use generators based on StyleGAN [18] and the improved version of StyleGAN [19] as our decoder network. In the following we refer to the models based on the first version of StyleGAN [18] as "StyleGAN 1" and models based on the improved version of StyleGAN [19] as "StyleGAN 2".

The StyleGAN architecture currently sets the state-of-the-art in unconditional high-resolution image generation for a multitude of different natural image categories such as faces, buildings, and animals. In order to generate high quality and high resolution images, StyleGAN 1 and StyleGAN 2 make use of a specialized generator architecture that is based on the idea of progressive growing for GANs [17].

All in all, the generator consists of three main components. The first component is the mapping network that converts a latent vector $z \in \mathcal{Z}$ with $\mathcal{Z} \in \mathbb{R}^n$ into an intermediate latent space $w \in \mathcal{W}$ with $\mathcal{W} \in \mathbb{R}^n$. The mapping network is implemented using a multilayer perceptron that typically consists of 8 layers with Leaky ReLU [24] as activation function. The resulting vector w in that intermediate latent space is then transformed using learned affine transformations and used as an input to a synthesis network.

The second important component is stochastic noise, which is another input to the generator network. Stochastic noise is added in the form of single channel images, where the value of each pixel is individually drawn from a normal distribution. We denote these noise images as noise maps in the remainder of this paper. In the original works [18, 19], the authors add these noise inputs in order to generate stochastic detail. However, in our experiments (see Section 4) we show that these noise maps can be used for even more than the generation of stochastic detail.

The third component is the synthesis network. It consists of multiple blocks that each take three inputs. First, they take a feature map that contains the current content information of the image that is to be generated. Second, each block takes a transformed representation of the vector w as an input to its style parts. These style parts are implemented differently in the two versions of StyleGAN. In StyleGAN 1 the Adaptive Instance Normalization (AdaIN) [13] is used to guide the normalization of the feature map. In StyleGAN 2 AdaIN is not used anymore, because AdaIN leads to characteristic artifacts in images generated by StyleGAN 1. The artifacts are mitigated by the introduction of novel modulation and demodulation operations The noise input is then added to the network after a normalization of the feature map. The resulting feature map is then fed to the next block, which performs the same steps again and includes an upsample operation in every second block.

We note, that we use the proposed decoder architectures, StyleGAN 1 and StyleGAN 2, as is and without any changes.

3.2. Encoder Architecture

The second part of the autoencoder architecture is our encoder (see Figure 2). The encoder is a fully convolutional network, that predicts latent vectors either in $\mathcal{Z}$ or in $\mathcal{W}$ and noise maps for each resolution of the generated images. If we predict the latent vectors in $\mathcal{W}$, we follow the notion of [2, 1] and name this space $\mathcal{W}^+$. Our fully convolutional network follows a residual structure [12] and predicts latent vectors and all noise vectors following the U-Net architecture [30] (see the supplementary material for further details). The predicted latent vectors and noise maps are then used as an input to a pre-trained StyleGAN 1 or StyleGAN 2 based generator, which is *not* updated during the training process of the encoder.

3.3. Maximizing the Semantic Meaning of the Latent Code

We use a two-stage training scheme to increase the semantic meaning of the latent code. The first stage disables learning (or usage) of the noise maps, forcing the model to only rely on the latent code for reconstruction. The second stage then fine-tunes (or extends) the model to further improve the reconstruction results by training the layers responsible for predicting the noise maps. We employ two different ways to achieve this goal.

Two Networks. On the one hand, we use two independent encoder networks L and N, where L predicts the latent code and N predicts the noise maps. This allows us to train only L during the first stage, while disabling N, thus forcing the model to use the latent code for the best result. In the second stage, we then use the pre-trained L and only train N.

Learning Rate. On the other hand, we use only one encoder model. We adapt the learning rate or disable learning of the projection layers, P_{latent} and P_{noise}, predicting the latent code and the noise maps, respectively (see Figure 2). During the first stage, we only train P_{latent} (projecting into $\mathcal{Z}$ or $\mathcal{W}^+$) and do not train the P_{noise} layers. In the second stage, we train all layers of our model, but we use different learning rates for different parts of the network. P_{noise} layers are trained with the regular learning rate, whereas the learning rate for the backbone and P_{latent} is reduced, *e.g.*, multiplied by $\frac{1}{100}$.

The second method is more efficient, since it only requires one encoder network. However, it is also susceptible to unlearn the usage of the latent code during stage two of the training if the learning rate is not tuned carefully. We discuss the effect and results of both training strategies in Section 4.3 and our supplementary material.

3.4. Training Details and Loss Function

We use two different loss functions for the training of our models. These loss functions are only used to update the weights of the *encoder*, the weights of the decoder (a pre-trained StyleGAN) are *fixed*. On the one hand, we use Mean Squared Error (MSE) between the pixels of the generated image and the reconstructed image. On the other hand, we utilize the Learned Perceptual Image Patch Similarity (LPIPS) [42] metric for judging the reconstruction quality. The resulting loss function is the following: $\mathcal{L}(x, y) = \mathcal{L}_{\text{mse}}(x, y) + \mathcal{L}_{\text{lpips}}(x, y)$. With $x, y \in \mathbb{R}^{[3,H,W]}$ being the input image and desired output image with three channels, height H and width W, respectively. $\mathcal{L}_{\text{mse}}$ and $\mathcal{L}_{\text{lpips}}$ denote MSE and LPIPS loss, respectively.

4. Results and Discussion

In this section, we show experimental results of our approach on different datasets and two different tasks, image reconstruction and image denoising. First, we show that we are able to faithfully reconstruct images with our presented autoencoder architecture with a StyleGAN decoder pretrained on the FFHQ dataset [18] and an encoder trained on the same dataset. Second, we show the results for cross-domain reconstruction, where the encoder is trained on a different dataset, but the same pre-trained StyleGAN decoder (trained on FFHQ) is used. Third, we investigate how high-quality reconstruction is possible in this setting by examining the role of the noise maps in StyleGAN. Afterwards, we present the results for our two-stage training method to increase the semantic meaning of the latent code. Finally, we show the capabilities of our model when applied for the task of image denoising.

4.1. Experimental Setup

We implement our model using PyTorch [27]. We use the FFHQ dataset [18] for high-quality images of human faces, and three LSUN datasets [37], each of which contains only images of one type: churches, bedrooms, and cats. We use the following three metrics on the given validation sets to evaluate our models: (1) The FID [32] of reconstructed images with the original images (using a sample size of 50 000 images). Furthermore, we calculate (2) PSNR and (3) SSIM [35] between each input and its corresponding reconstructed image to measure the reconstruction quality. Further details, such as, system details, number of itera-

Training Dataset, StyleGAN Version, Projection Target	Dataset and Metric for Evaluation											
	FFHQ			Church			Bedroom			Cat		
	FID↓	PSNR↑	SSIM↑	FID↓	PSNR↑	SSIM↑	FID↓	PSNR↑	SSIM↑	FID↓	PSNR↑	SSIM↑
FFHQ, 1, $\mathcal{Z}$	9.85	25.03	0.91	7.17	20.37	0.88	3.74	23.32	0.91	4.17	22.14	0.88
FFHQ, 1, $\mathcal{W}^+$	0.64	25.54	0.94	1.37	22.26	0.93	0.55	24.21	0.94	0.90	23.02	0.91
FFHQ, 2, $\mathcal{Z}$	3.92	24.39	0.88	4.66	19.87	0.82	3.24	22.71	0.86	4.47	21.50	0.84
FFHQ, 2, $\mathcal{W}^+$	0.75	29.11	0.95	1.23	24.48	0.94	0.57	28.13	0.96	0.96	26.01	0.93
Church, 1, $\mathcal{Z}$	17.28	19.83	0.86	3.17	23.32	0.91	4.22	21.90	0.90	4.98	20.50	0.86
Church, 1, $\mathcal{W}^+$	3.30	20.99	0.90	0.26	26.18	0.95	1.25	23.65	0.94	1.37	22.15	0.90
Church, 2, $\mathcal{Z}$	12.24	20.92	0.83	3.17	23.08	0.86	9.76	22.04	0.85	7.33	20.76	0.82
Church, 2, $\mathcal{W}^+$	2.33	23.10	0.91	0.21	28.99	0.95	0.60	26.43	0.96	1.06	24.35	0.91
Bedroom, 1, $\mathcal{Z}$	5.82	23.14	0.91	2.23	23.41	0.92	1.13	26.94	0.95	2.11	23.71	0.90
Bedroom, 1, $\mathcal{W}^+$	1.60	24.10	0.91	1.16	23.00	0.93	0.30	26.22	0.95	0.79	23.86	0.91
Bedroom, 2, $\mathcal{Z}$	7.17	22.70	0.87	6.48	20.49	0.84	2.96	24.29	0.88	5.10	21.97	0.85
Bedroom, 2, $\mathcal{W}^+$	1.59	26.44	0.93	1.06	26.97	0.94	0.36	31.01	0.97	0.84	26.78	0.93
Cat, 1, $\mathcal{Z}$	39.65	24.60	0.92	6.68	24.25	0.93	4.01	26.71	0.94	2.96	25.03	0.92
Cat, 1, $\mathcal{W}^+$	1.12	24.94	0.93	0.88	24.29	0.94	0.28	26.40	0.96	0.57	24.86	0.93
Cat, 2, $\mathcal{Z}$	5.31	23.61	0.90	2.83	22.73	0.90	2.18	24.49	0.92	2.71	23.31	0.89
Cat, 2, $\mathcal{W}^+$	1.34	28.38	0.95	0.55	27.99	0.95	0.33	30.35	0.97	0.68	28.41	0.94

Table 1: The results of our reconstruction experiments. We use a StyleGAN model for decoding, which was pre-trained on the FFHQ dataset [18] and is not updated during the training of the encoder. Only the first highlighted row uses an encoder which is also trained on FFHQ, the other encoders are trained on different LSUN datasets [37]. The first column shows the dataset, the version of StyleGAN (1, 2), and the projection target ($\mathcal{Z}, \mathcal{W}^+$). Each model is evaluated on different datasets and we report Frechet Inception Distance (FID), Peak Signal-to-Noise Ratio (PSNR), and Structural Similarity Index (SSIM).

tions, optimizer, learning rate, and data preprocessing can be found in the supplementary material.

4.2. FFHQ-based Image Reconstruction

In our first set of experiments, we determined how well our autoencoder architecture (introduced in Section 3) is able to reconstruct images of the FFHQ dataset [18], when using a StyleGAN model pre-trained on the FFHQ dataset. In this line, we trained a range of different encoders, using both StyleGAN 1 and StyleGAN 2 decoders. Furthermore, we examined the influence of different latent code projection strategies. On the one hand, we project into $\mathcal{Z}$ and use the mapping network for creating a single vector in $\mathcal{W}$. On the other hand, we project into $\mathcal{W}^+$, using multiple latent vectors.

The quantitative results (see the highlighted first row in Table 1) show, that our autoencoder is able to perform reconstruction for different datasets with overall high quality. Note that even though both encoder and decoder were trained on FFHQ only, they show a high reconstruction quality when evaluated on other datasets, *e.g.*, containing churches, bedrooms, or cats. The similarity metrics PSNR and SSIM also clearly show the advantages of using the projection strategy $\mathcal{W}^+$ over $\mathcal{Z}$.

The qualitative results also show nearly no perceptual differences (see first two columns of Figure 3). However, we can see that the models based on StyleGAN 1 exhibit the "bubble" artifacts typical for images produced by StyleGAN 1 [19]. The absence of these artifacts in the reconstructed images of the StyleGAN 2 based models is most likely the reason for the better quantitative results.

Cross-Domain Image Reconstruction Intrigued by our results on the FFHQ dataset, we trained a different set of autoencoder models that use the same pre-trained and fixed StyleGAN generator, but use an LSUN dataset for training the encoder part of our autoencoder. The quantitative results of these experiments are shown in rows 2-4 of Table 1. Further, we show the qualitative results of the encoders trained on churches and bedrooms in column 3 and 4 of Figure 3, respectively. These results show that such cross-domain models are able to reconstruct images with comparable perceptual quality and scores. We think that this is a very interesting result, since we are using a model trained for generating faces, but were able to use this model to generate virtually any image. However, these finding are in line with the findings of Abdal *et al.* [2, 1], who show that a latent code for many different images can be found, not only for images the model has been trained on.

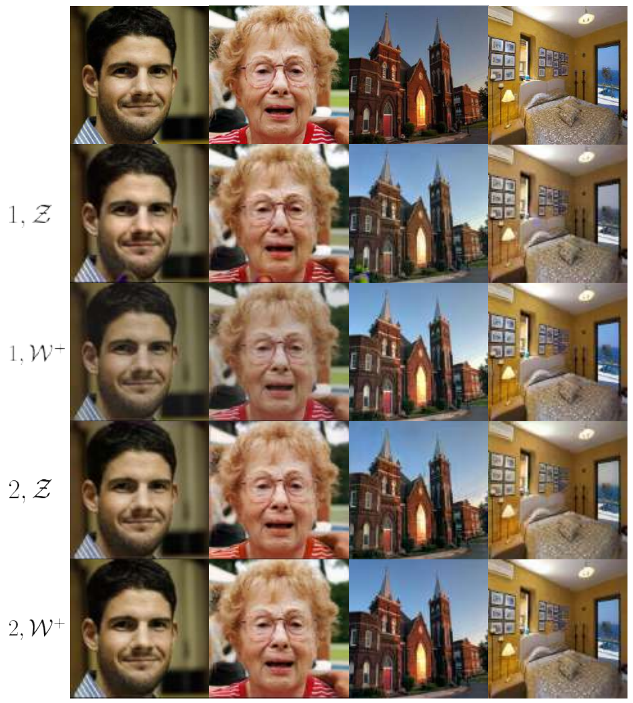


Figure 3: Reconstruction results of our models trained with a StyleGAN model pre-trained on the FFHQ dataset. Images in the first row are real images, images in the following rows are reconstructions where the naming is as follows: StyleGAN variant, latent projecting strategy. Best viewed in color, more details visible when zoomed in.

The Significance of Noise for Image Reconstruction To understand how our reconstruction model can produce such high quality results, we examined the latent code and the noise maps predicted by our model. First, we directly visualized the (normalized) noise maps predicted by our encoder (see Figure 4). It is clearly visible that the encoder learns to use the noise maps for retaining the content of the input image, especially in the noise maps of higher resolution. This is an interesting observation, since the work of Karras et al. shows that the latent code provides semantic meaning [18, 19] and the noise only provides semantically irrelevant details when used for a regular image generation task. During reconstruction the roles seem effectively reversed, the latent code is barely used to store information, instead the noise maps capture the information up to a pixel level.

To further analyze the meaning of the noise maps, we multiplied the value of each pixel in a noise map with a factor from the interval $[-2, 3]$ and examined the reconstructed image. The results (see Figure 5, an extended version is included in the supplementary material) show that the encoder uses the noise maps not only to capture the content of the image, but can also (at least to some degree) encode the colors of each pixel in these noise maps.

4.3. Semantic Image Reconstruction

Similarly, we examined the semantic meaningfulness of the latent code with sample interpolations between two images (see Figure 6a) based on a StyleGAN 2 decoder trained on FFHQ. This visualization shows, that a model simply trained for reconstruction is not able to perform semantic interpolation. It is visually more similar to an alpha blending between two images. Thus, it seems the influence of the latent code is degraded in such a way that it is only used to provide some basic colors for the resulting image and the resulting reconstruction completely depends on the predicted noise maps.

However, we also tested our improved two-stage training strategy (introduced in Section 3.3) to find a more meaningful (semantic) latent code and perform more meaningful semantic interpolations. The results of the *two network* strategy (see Figure 6b) show that the semantic latent code captures the coarse structure of the content, but fine details are still added by the predicted noise maps. We also observe that the interpolations seem to be more reasonable, but the visual quality of images reconstructed by using only the latent code is still visibly lower than the original images. When incorporating the predicted noise maps on top of the semantic latent codes, the quality increases, but can not achieve the same level compared to our other experiments (the metrics and visualized interpolations for these models are included in the supplementary material). The results of the *learning rate* strategy can achieve similar results (see the supplementary material for details), however we found it can be tedious to find the optimal learning rates. Without careful tuning of the actual learning rate in stage 2, the model unlearns the previously learned semantic meaning of the latent code.

4.4. Reconstruction Speed

We reproduced the approach of Abdal *et al.* [2, 1] to measure its speed and found it needs approximately 7 minutes for a *single* image on a RTX 2080 TI GPU. The approach of Guan *et al.* [11] is much faster, but still needs about 0.71 seconds per image on a Tesla V100 GPU, meaning they can process about 1.4 images per second. For comparison, our model can process approximately 40 images per second on a RTX 2080 TI GPU, while still producing reconstructions with very high perceptual quality.

4.5. Image Denoising

As a second task, we tested the capabilities of our approach on the exemplary application of image denoising. As already introduced in Section 2, several deep learning based image denoising techniques have been proposed in the past. We note that our model was not specifically designed as an

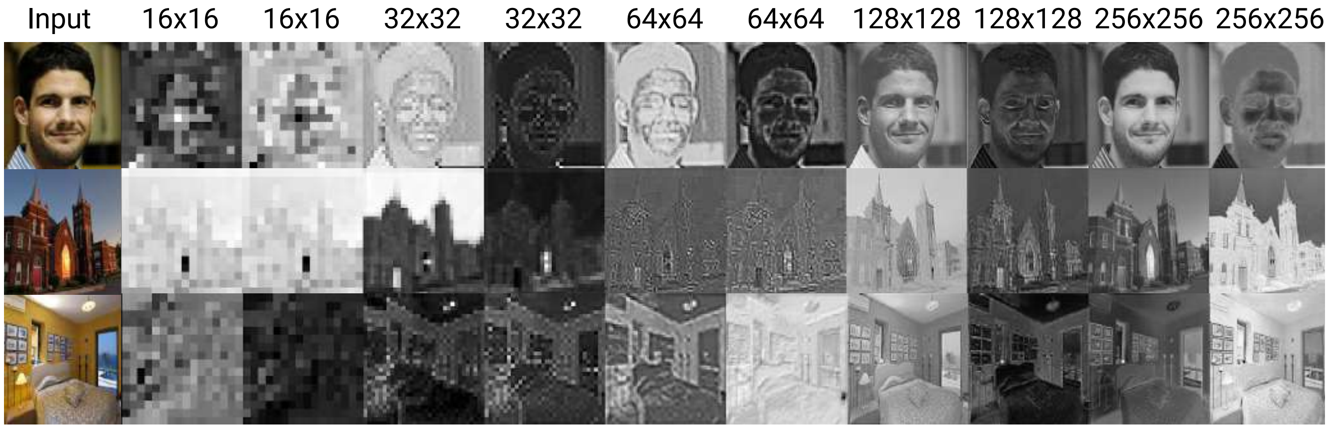


Figure 4: Noise maps predicted by our model for the given input images on the left. The noise maps are made visible by normalizing the values of each noise map independently. It is clearly visible that all content information is saved in the noise maps. The level of detail follows the progressive growing architecture of StyleGAN and guides the generator by adding more and more content information to the already generated image.

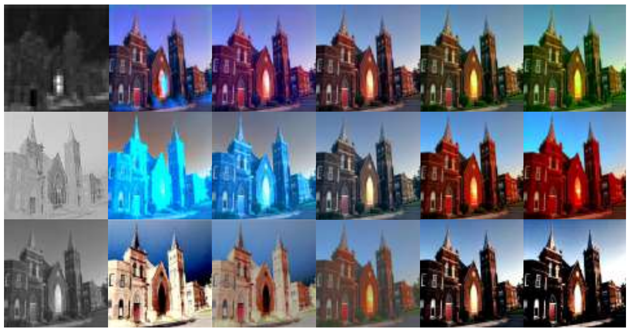


Figure 5: Results of our experiments where we shifted the values of the predicted noise maps. Each row shows the results when "shifting" each pixel of the noise map shown in the first column by multiplying the noise map with -2, -0.75, 0.5, 1.75, and 3 (from left to right). It is clearly visible that the noise maps are not only used to encode the content of an image, but that noise can also be used to encode color and contrast of images. An extended version is in the supplementary material.

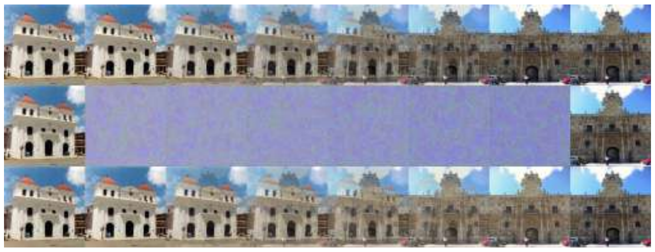
image denoising network, unlike approaches from other related work. Compared to other works, our model harnesses the generative power of StyleGAN to directly predict the denoised image without the need for a residual image that is subtracted from the noisy input image.

For image denoising, we trained multiple models and compare them to the state-of-the-art in image denoising on the BSD68 [31] and the SET 12 [25] benchmark datasets. We trained models based on StyleGAN 1 and StyleGAN 2 (both pre-trained on the FFHQ dataset), with latent code embedding into $\mathcal{Z}$ and $\mathcal{W}^+$, and use the ImageNet dataset [7] for training the encoder. We report the average PSNR and SSIM of our model on different noise levels $\sigma = 15, 25,$ and 50 on the benchmark datasets in Table 2. The qualitative results can be seen in Figure 7. Our results show that our model is able to perform image denoising at a level close to the state-of-the-art, even though our network has not been designed with the application of image denoising in mind. Some ideas mentioned in related word (*e.g.* network design decisions, similar to [22] or [38]) could be used to boost the performance of our model, but such improvements are out of the scope of this work, since here we only want to show that such an application is possible.

The results are nonetheless quite interesting considering the generator of the network has never been trained for image reconstruction and also never for the creation of images apart from faces of the FFHQ dataset. Another observation based on the qualitative results of the denoised images is that our model not only removes noise from an image, but also performs a slight shift in colors similar to a color correction operation. Since changing the colors might be an unwanted result when removing noise from input images, the model might perform even better in the quantitative analysis, without this kind of color correction. This behavior is an interesting property of our overall model and opens up further application possibilities of our proposed network in future work.

5. Conclusion

In this work we showed a novel approach, which can achieve high-quality image reconstruction results. In contrast to previous work, our model is highly efficient and can be applied for cross-domain image reconstruction. We showed that a pre-trained StyleGAN generator can be used to reconstruct images from virtually any dataset. We pro-


(a) Regular training approach.

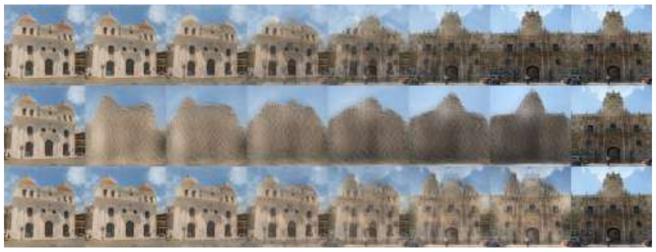

(b) Two-stage training with two networks (see Section 3.3).

Figure 6: Two images showcasing the behaviour of our models when interpolating latent code and noise maps between two reconstruction images based on two of our training strategies. Both encoders embed into $\mathcal{W}^+$ of a StyleGAN 2 decoder pre-trained on the FFHQ dataset. The rows of each image show the following: The first row shows the interpolation between predicted latent code and noise at the same time. The second row shows the interpolation of only the latent code, using a fixed random noise. The third row shows the interpolation of only the noise maps, using the fixed latent code of the left image.

Dataset	Set12			BSD68		
σ	15	25	50	15	25	50
Liu *et al.* [22]	33.15/**0.90**	30.79/**0.87**	27.74/**0.81**	31.86/0.90	29.41/0.84	26.53/0.74
Zhang *et al.* [41]	32.75/ -	30.43/ -	27.32/ -	31.63/ -	29.19/ -	26.29/ -
Zhang *et al.* [38]	**33.25/ -**	**30.94/ -**	**27.90/ -**	**31.91/ -**	**29.48/ -**	**26.59/ -**
StyleGAN 1, $\mathcal{Z}$	24.67/0.82	24.13/0.76	22.27/0.60	24.59/0.86	24.48/0.83	23.71/0.73
StyleGAN 1, $\mathcal{W}^+$	26.42/0.87	25.93/0.81	24.18/0.64	25.17/0.91	25.02/0.87	24.78/0.77
StyleGAN 2, $\mathcal{Z}$	26.02/0.84	25.58/0.80	23.88/0.67	24.70/0.85	24.76/0.83	24.30/0.75
StyleGAN 2, $\mathcal{W}^+$	27.46/0.88	27.08/0.85	24.94/0.71	27.57/**0.92**	27.46/**0.89**	26.22/**0.81**

Table 2: Average PSNR/SSIM results of our model compared to state-of-the-art image denoising models. **Bold font** indicates the best performing result.

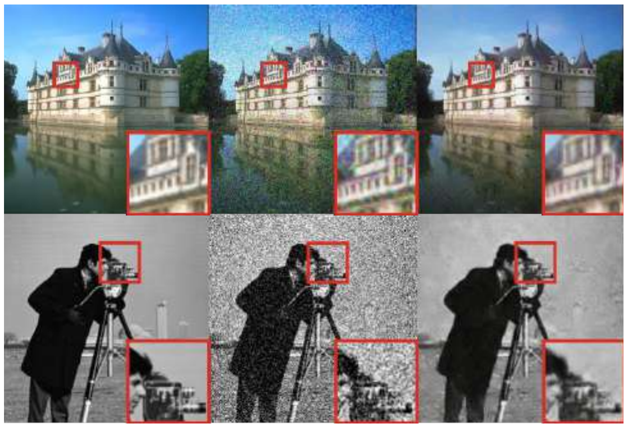


Figure 7: Qualitative results of our model on the image denoising task. We show two examples. The example on the top is from the BSD68 dataset and the example on the bottom from the Set12 dataset. For denoising, we used an autoencoder trained for denoising with an encoder embedding into the $\mathcal{W}^+$ space of a StyleGAN 2 pre-trained on FFHQ. For creating the noisy image, we used additive gaussian white noise with a standard deviation of 50.

vided an in-depth analysis of the causes for this behavior and found that the stochastic noise inputs, which are only meant to produce stochastic variations, can be used to capture small details, and manipulate colors of images generated by StyleGAN without using the latent code. Since we can circumvent using the latent code, the latent code loses its semantic expressiveness. However, we proposed a two-stage training strategy for two slightly different architectures, which increases the semantic expressiveness of the computed latent code without decreasing the reconstruction quality by a large margin.

In future work, we would like to investigate further application possibilities of our proposed autoencoder architecture, *e.g.* applications in the area of image-to-image translation.

Acknowledgement We wish to thank the Wildenstein Plattner Institute for supplying us with data and a research objective that ultimately led to the results published in this paper.

References

[1] Rameen Abdal, Yipeng Qin, and Peter Wonka. Image2StyleGAN++: How to Edit the Embedded Images? *arXiv:1911.11544 [cs]*, Nov. 2019. arXiv: 1911.11544.

[2] Rameen Abdal, Yipeng Qin, and Peter Wonka. Image2StyleGAN: How to Embed Images Into the StyleGAN Latent Space? pages 4432–4441, 2019.

[3] David Bau, Jun-Yan Zhu, Jonas Wulff, William Peebles, Hendrik Strobelt, Bolei Zhou, and Antonio Torralba. Seeing What a Gan Cannot Generate. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4502–4511, 2019.

[4] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large Scale GAN Training for High Fidelity Natural Image Synthesis. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[5] Yunjin Chen and Thomas Pock. Trainable Nonlinear Reaction Diffusion: A Flexible Framework for Fast and Effective Image Restoration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1256–1272, June 2017. Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence.

[6] Antonia Creswell and Anil Anthony Bharath. Inverting the Generator of a Generative Adversarial Network. *IEEE Transactions on Neural Networks and Learning Systems*, 30(7):1967–1974, 2018.

[7] J. Deng, W. Dong, R. Socher, L. Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, June 2009.

[8] Linwei Fan, Fan Zhang, Hui Fan, and Caiming Zhang. Brief review of image denoising techniques. *Visual Computing for Industry, Biomedicine, and Art*, 2(1):7, July 2019.

[9] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc., 2014.

[10] Bhawna Goyal, Ayush Dogra, Sunil Agrawal, B. S. Sohi, and Apoorav Sharma. Image denoising review: From classical to state-of-the-art approaches. *Information Fusion*, 55:220–244, Mar. 2020.

[11] Shanyan Guan, Ying Tai, Bingbing Ni, Feida Zhu, Feiyue Huang, and Xiaokang Yang. Collaborative Learning for Faster StyleGAN Embedding. *arXiv preprint arXiv:2007.01758*, 2020.

[12] 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*, pages 770–778, 2016.

[13] Xun Huang and Serge Belongie. Arbitrary Style Transfer in Real-Time With Adaptive Instance Normalization. pages 1501–1510, 2017.

[14] Xun Huang, Ming-Yu Liu, Serge Belongie, and Jan Kautz. Multimodal Unsupervised Image-to-image Translation. pages 172–189, 2018.

[15] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-Image Translation with Conditional Adversarial Networks. *arXiv:1611.07004 [cs]*, Nov. 2016. arXiv: 1611.07004.

[16] Viren Jain and Sebastian Seung. Natural Image Denoising with Convolutional Networks. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems 21*, pages 769–776. Curran Associates, Inc., 2009.

[17] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive Growing of GANs for Improved Quality, Stability, and Variation. In *International Conference on Learning Representations*, 2018.

[18] Tero Karras, Samuli Laine, and Timo Aila. A Style-Based Generator Architecture for Generative Adversarial Networks. *arXiv:1812.04948 [cs, stat]*, Mar. 2019. arXiv: 1812.04948.

[19] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[20] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *Proceedings of the 3rd International Conference on Learning Represenations*, San Diego, 2015. arXiv: 1412.6980.

[21] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *2nd International Conference on Learning Representations, ICLR 2014 - Conference Track Proceedings*, 2014.

[22] Pengju Liu, Hongzhi Zhang, Kai Zhang, Liang Lin, and Wangmeng Zuo. Multi-Level Wavelet-CNN for Image Restoration. pages 773–782, 2018.

[23] Ilya Loshchilov and Frank Hutter. SGDR: stochastic gradient descent with warm restarts. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[24] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier nonlinearities improve neural network acoustic models. In *in ICML Workshop on Deep Learning for Audio, Speech and Language Processing*. Citeseer, 2013.

[25] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, volume 2, pages 416–423 vol.2, July 2001.

[26] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral Normalization for Generative Adversarial Networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

[27] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d\textquotesingle Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8026–8037. Curran Associates, Inc., 2019.

[28] Guim Perarnau, Joost Van De Weijer, Bogdan Raducanu, and Jose M Álvarez. Invertible Conditional Gans for Image Editing. *arXiv preprint arXiv:1611.06355*, 2016.

[29] Stanislav Pidhorskyi, Donald Adjeroh, and Gianfranco Doretto. Adversarial Latent Autoencoders. *arXiv:2004.04467 [cs]*, Apr. 2020. arXiv: 2004.04467.

[30] 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.

[31] Stefan Roth and Michael J. Black. Fields of Experts. *International Journal of Computer Vision*, 82(2):205–229, Apr. 2009.

[32] Tim Salimans, Ian J. Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved Techniques for Training GANs. In Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 2226–2234, 2016.

[33] Tamar Rott Shaham, Tali Dekel, and Tomer Michaeli. SinGAN: Learning a Generative Model From a Single Natural Image. pages 4570–4580, 2019.

[34] Zhiqiang Shen, Mingyang Huang, Jianping Shi, Xiangyang Xue, and Thomas S. Huang. Towards Instance-Level Image-To-Image Translation. pages 3683–3692, 2019.

[35] Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, Apr. 2004. Conference Name: IEEE Transactions on Image Processing.

[36] Junyuan Xie, Linli Xu, and Enhong Chen. Image Denoising and Inpainting with Deep Neural Networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 341–349. Curran Associates, Inc., 2012.

[37] Fisher Yu, Yinda Zhang, Shuran Song, Ari Seff, and Jianxiong Xiao. LSUN: Construction of a Large-scale Image Dataset using Deep Learning with Humans in the Loop. *arXiv preprint arXiv:1506.03365*, 2015.

[38] Kai Zhang, Yawei Li, Wangmeng Zuo, Lei Zhang, Luc Van Gool, and Radu Timofte. Plug-and-Play Image Restoration with Deep Denoiser Prior. *arXiv:2008.13751 [cs, eess]*, Aug. 2020. arXiv: 2008.13751.

[39] Kai Zhang, Wangmeng Zuo, Yunjin Chen, Deyu Meng, and Lei Zhang. Beyond a Gaussian Denoiser: Residual Learning of Deep CNN for Image Denoising. *IEEE Transactions on Image Processing*, 26(7):3142–3155, July 2017. Conference Name: IEEE Transactions on Image Processing.

[40] Kai Zhang, Wangmeng Zuo, Shuhang Gu, and Lei Zhang. Learning Deep CNN Denoiser Prior for Image Restoration. pages 3929–3938, 2017.

[41] Kai Zhang, Wangmeng Zuo, and Lei Zhang. FFDNet: Toward a Fast and Flexible Solution for CNN-Based Image Denoising. *IEEE Transactions on Image Processing*, 27(9):4608–4622, Sept. 2018. Conference Name: IEEE Transactions on Image Processing.

[42] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The Unreasonable Effectiveness of Deep Features as a Perceptual Metric. pages 586–595, 2018.

[43] Jiapeng Zhu, Yujun Shen, Deli Zhao, and Bolei Zhou. Indomain Gan Inversion for Real Image Editing. *The European Conference on Computer Vision (ECCV)*, 2020.

[44] Jun-Yan Zhu, Philipp Krähenbühl, Eli Shechtman, and Alexei A Efros. Generative Visual Manipulation on the Natural Image Manifold. In *European Conference on Computer Vision*, pages 597–613. Springer, 2016.

[45] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, 2017.

A. Further Implementation Details

We implemented our model in PyTorch, as already stated in the main paper. We based our implementation of the StyleGAN 1 and StyleGAN 2 models on freely on Github available re-implementations of StyleGAN 1[2] and StyleGAN 2[3]. Our code is also available on Github[4] and our training logs can be viewed online[5].

We perform our experiments on a range of different GPUs with at least 11 GB of GPU memory. For all of our experiments, we train a model for 100 000 iterations, using two GPUs with a batch size of 4 per GPU. We use the Adam [20] optimizer with a cosine annealing learning rate schedule [23] and an initial learning rate of 0.0001. During the preprocessing, all input images are resized to 256×256 pixels, disregarding aspect ratios.

Network Details Our encoder is based on the ResNet architecture, but does not follow the layout of other well-known ResNet feature extractors, *e.g*. ResNet-18, or ResNet-152. Instead, the number of convolutional layers in our feature extractors depends on the resolution of the input image. The number of necessary ResNet blocks can be calculated using the following formula:

$$\text{number of blocks} = 2 + 2 \cdot (\log_2(\text{insize}) - \log_2(\text{outsize})). \tag{1}$$

We first start with two ”start blocks”, then we use two ResNet blocks for each resolution from input size to output size of the encoder. The output size of the encoder is typically set to 4, wheras we use 256 as our input size. When using 256 and 4 as an input and output size, respectively, we get the following number of ResNet blocks:

$$\begin{aligned} \text{number of blocks} &= 2 + 2 \cdot (\log_2(256) - \log_2(4)) && (2)\\ &= 2 + 2 \cdot (8 - 2) && (3)\\ &= 2 + 2 \cdot 6 && (4)\\ &= 14. && (5) \end{aligned}$$

Following each ResNet block, the network splits into three branches. If we use StyleGAN 1 as decoder, we only split after the first ResNet block of each resolution. The first split is followed by a global average pooling and a 1×1 convolution that predicts parts of the latent code. The second split is followed by a 1×1 convolution that is used to predict the noise map for the current feature map resolution. The third branch goes to the next ResNet block. Please see Figure 2 of the main paper for a structural overview.

[2]`https://github.com/rosinality/style-based-gan-pytorch`

[3]`https://github.com/rosinality/stylegan2-pytorch`

[4]`https://github.com/Bartzi/one-model-to-reconstruct-them-all`

[5]`https://wandb.ai/hpi/OneModeltoGeneratethemAll`

B. Experimental Results on Further Pre-Trained StyleGAN Models

In order to show that reconstruction with StyleGAN does not only work with a generator pre-trained on the FFHQ dataset, we also performed experiments with generators pre-trained on the datasets LSUN church and LSUN cat. The results of these experiments show that our approach is also applicable to other pre-trained generators and reaches a very similar performance on a range of different datasets. Please note, that here we only experimented with models based on StyleGAN 2, since we were not able to get pre-trained models for the Pytorch implementation of StyleGAN 1 that we base our work on, but we are absolutely certain that we can reach the same results with models based on StyleGAN 1. We show the quantitative results of our experiments in Table 3 and Table 4. Furthermore, we also show more image reconstruction results in Figure 8a and Figure 8b.

Further Visualizations Explaining the Role of Noise

In Figures 9, 10, and 11 we show more detailed results of our noise shifting experiments. These experiments show that noise is used in several ways: (1) Noise can be used to control the content of the generated image. (2) Noise can be used to control the colors of individual pixels of the generated image. (3) A StyleGAN model trained on one dataset can make use of noise in a different way than a StyleGAN model trained on a different dataset, *i.e*. the color coding depends on the StyleGAN model and used dataset for training the StyleGAN model.

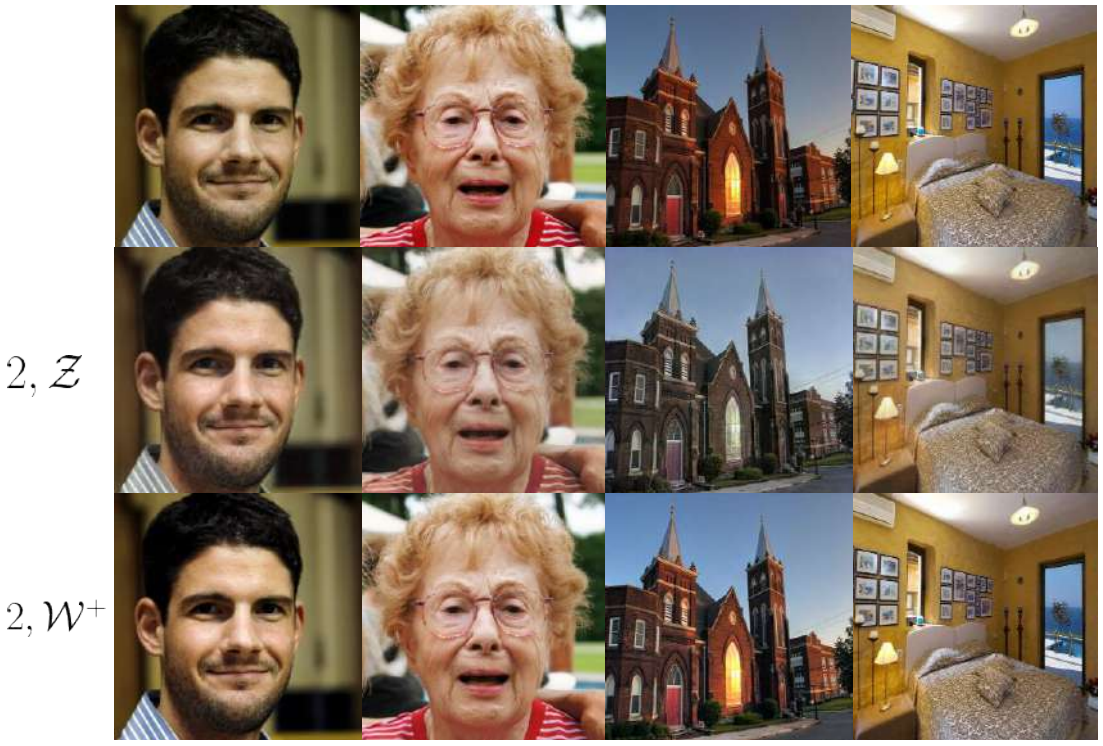
Further Visualizations Showing Interpolation Results

In Figure 12 we show further interpolation results. Again, we can observe that models trained with the two network approach provide more meaningful interpolations. We can also make the following observations: (1) If using a model trained on another domain than the input image, *e.g*. a model trained to reconstruct FFHQ using the two network strategy and using images from the LSUN church dataset (see Figure 12b), we can see that the encoder explicitly learns to embed the latent code into the regions where faces can be generated, regardless of the input image. We can hence conclude that the encoder does not learn anything about the content of the image when embedding into the latent code. (2) in Figure 12b and Figure 12d, we can observe that the predicted noise maps are “rendered” on top of the results of the latent code. This might be because the two networks in our two network architecture are independent of each other. In the future it might be worthwile to achieve a closer coupling of noise and latent code in the two network approach. (3) Overall, we can observe that, when not training with the two network strategy, the latent code is only used to add color variations into the resulting images. This can, for instance, be observed in Figure 12a in the second-to-right image in the bottom-most row. Here, we can see that the latent code also influences the colors of the resulting image, since the image on the right is the reconstruction of the second image without any interpolation between the two input images.

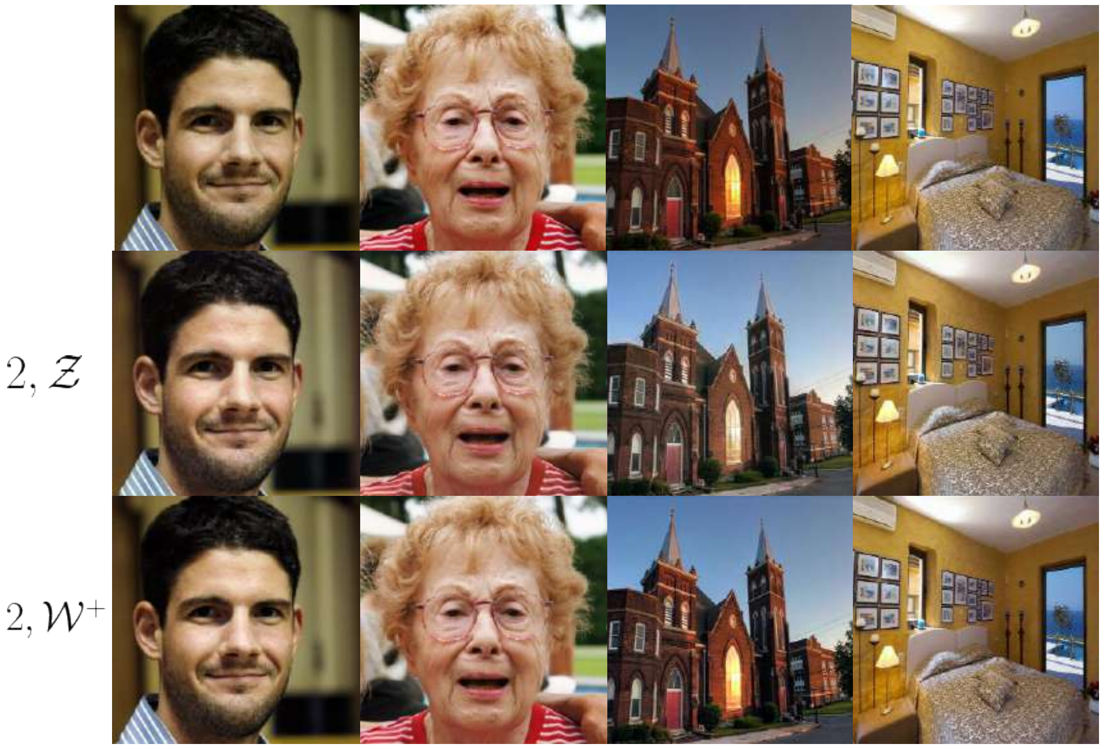
Since, we also want to compare the two approaches for maximising the semantic meaning of the latent code, we provide a comparison of interpolations for the approaches based on the two network and the learning rate strategy in Figure 13. We can observe that the learning rate split strategy produces better qualitative results. However, it is very difficult to achieve these results for each StyleGAN version, since the learning rate has to be tuned specifically.

C. Quantitative Reconstruction Results of our Two Network Models

In Table 5 we show quantitative results for reconstruction on multiple datasets when using a model trained with the two network strategy. The results show that a model trained with the two network strategy is still able to provide meaningful reconstructions. However, the results also show that the resulting reconstructions are of worse quality than the reconstructions that nearly completely rely on the noise input. Furthermore, we can see that the resulting models are not well suited for cross domain reconstruction. We argue that this is because the latent code is specialized to a specific dataset/data distribution and is not able to handle inputs that are different, since the encoder has learned to project the input image into the regions of the latent space for the specific type of data it has been trained for. From these observations we conclude that our models, which are not trained to maximise the semantic meaning of the latent code, are so versatile, because they learn to encode the content of the image in the noise maps and to use the values of each pixel to further encode the color of the pixel, making them independent of the latent code in StyleGAN, which is adjusted for one distribution only. However, it would be interesting to investigate the latent projections of different encoders to learn more about the structure of the latent code in StyleGAN. We leave these experiments open for future work.



(a) Results for models based on StyleGAN models pre-trained on the LSUN Cat dataset.



(b) Results for models based on StyleGAN models pre-trained on the LSUN Church dataset.

Figure 8: Results of further Reconstruction experiments with StyleGAN models pre-trained on other datasets than FFHQ. Images in the first row are real images, images in the following rows are reconstructions where the naming is as follows: StyleGAN variant, latent projecting strategy. Best viewed in color.

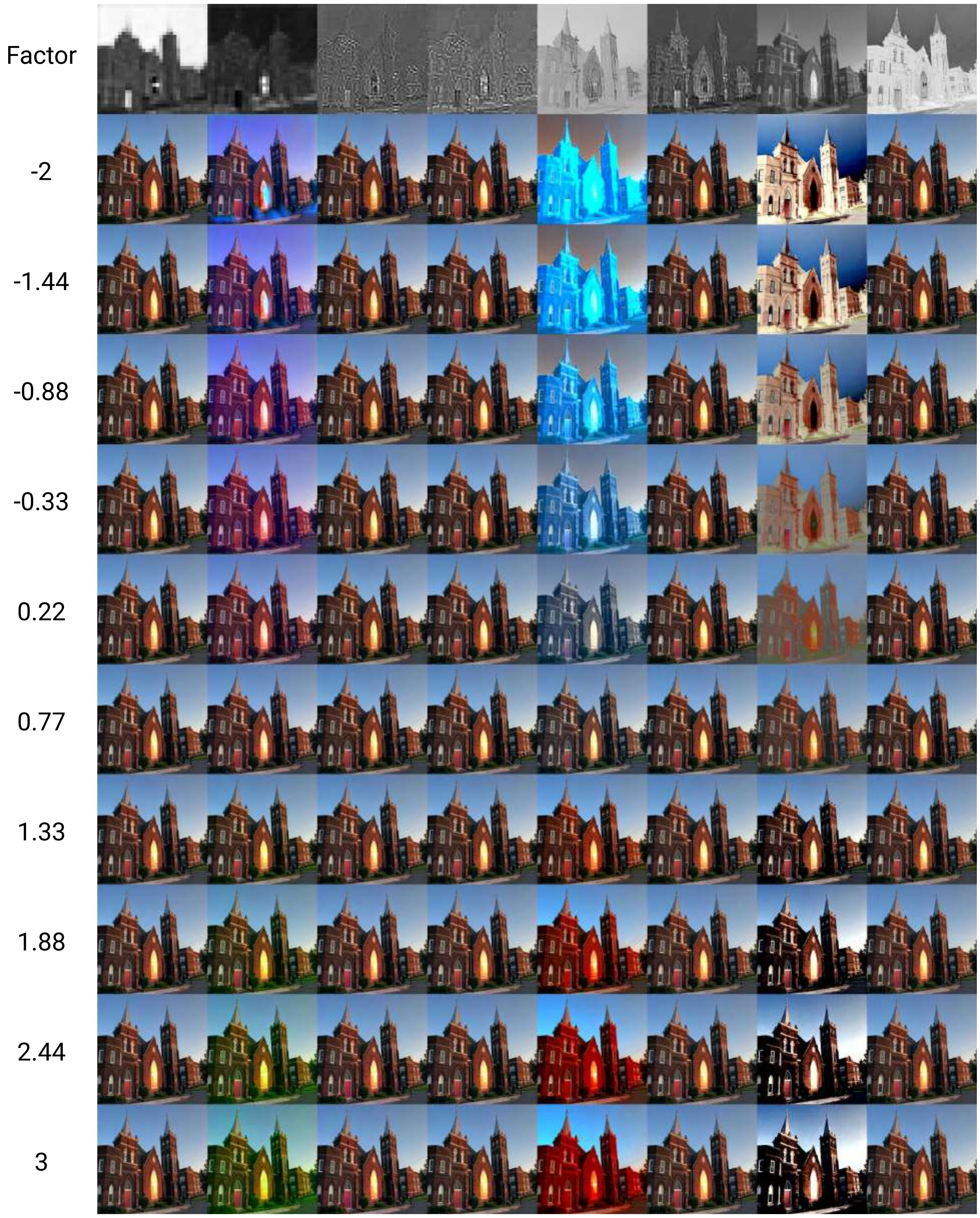


Figure 9: Longer version of noise shifting experiments with a StyleGAN 2 generator pre-trained on the FFHQ dataset. Each column shows the results when “shifting” each pixel of the corresponding noise map shown in the first row of the column by multiplying the noise map with the factors indicated at the left side. It is clearly visible that the noise maps are not only used to encode the content of an image, but that noise can also be used to encode color and contrast of images. However, not all noise maps are necessary for this color coding, as the result does not change for some noise maps, regardless of the factor used.

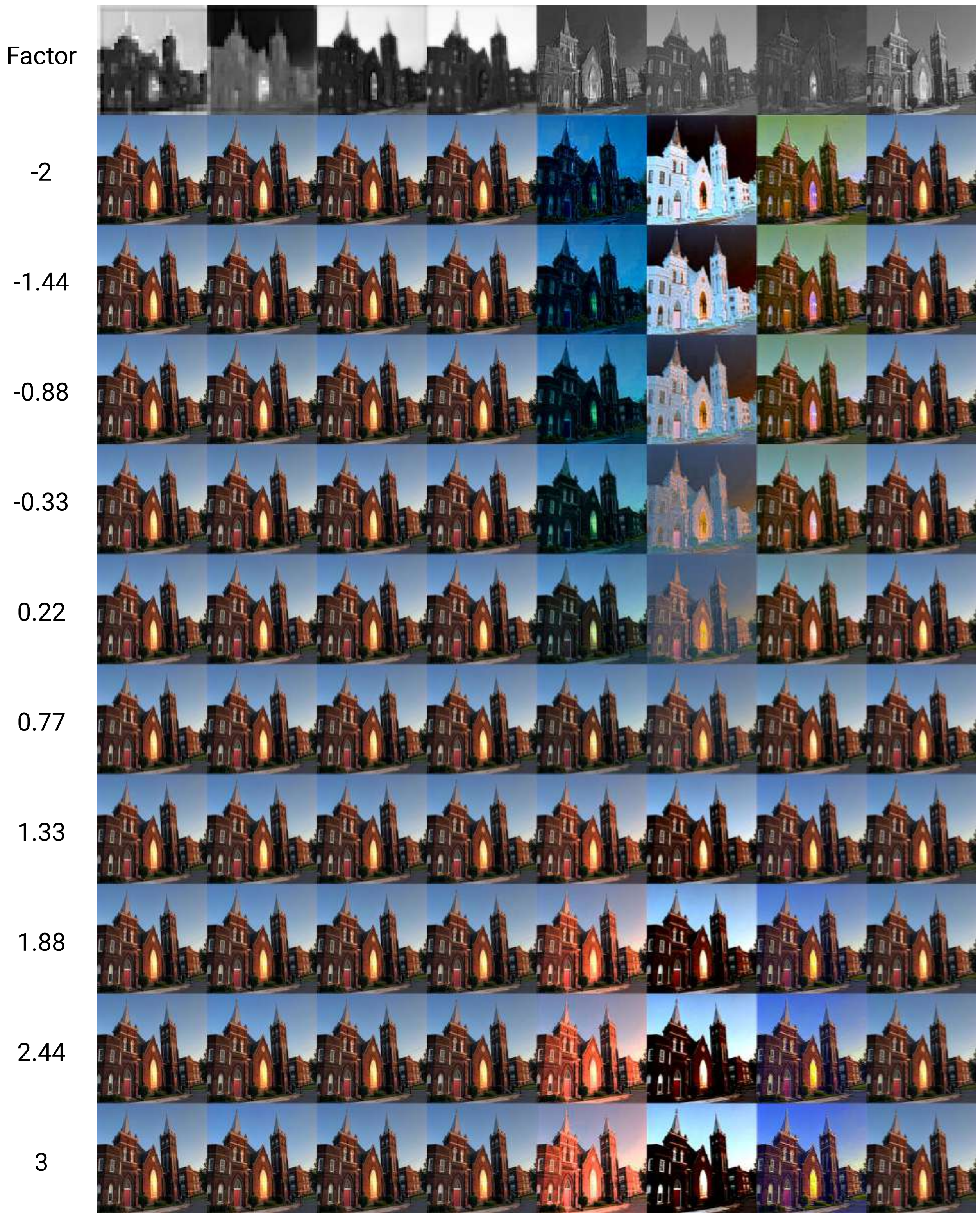


Figure 10: Further results of noise shifting experiments. Here we show the results with a StyleGAN 2 generator pre-trained on the LSUN church dataset. The semantics are the same as in Figure 9. Here it is also clearly visible that the noise maps are not only used to encode the content of an image, but that noise is also used to encode color and contrast of images, but for this model different noise maps are used and the color model encoded is also different.

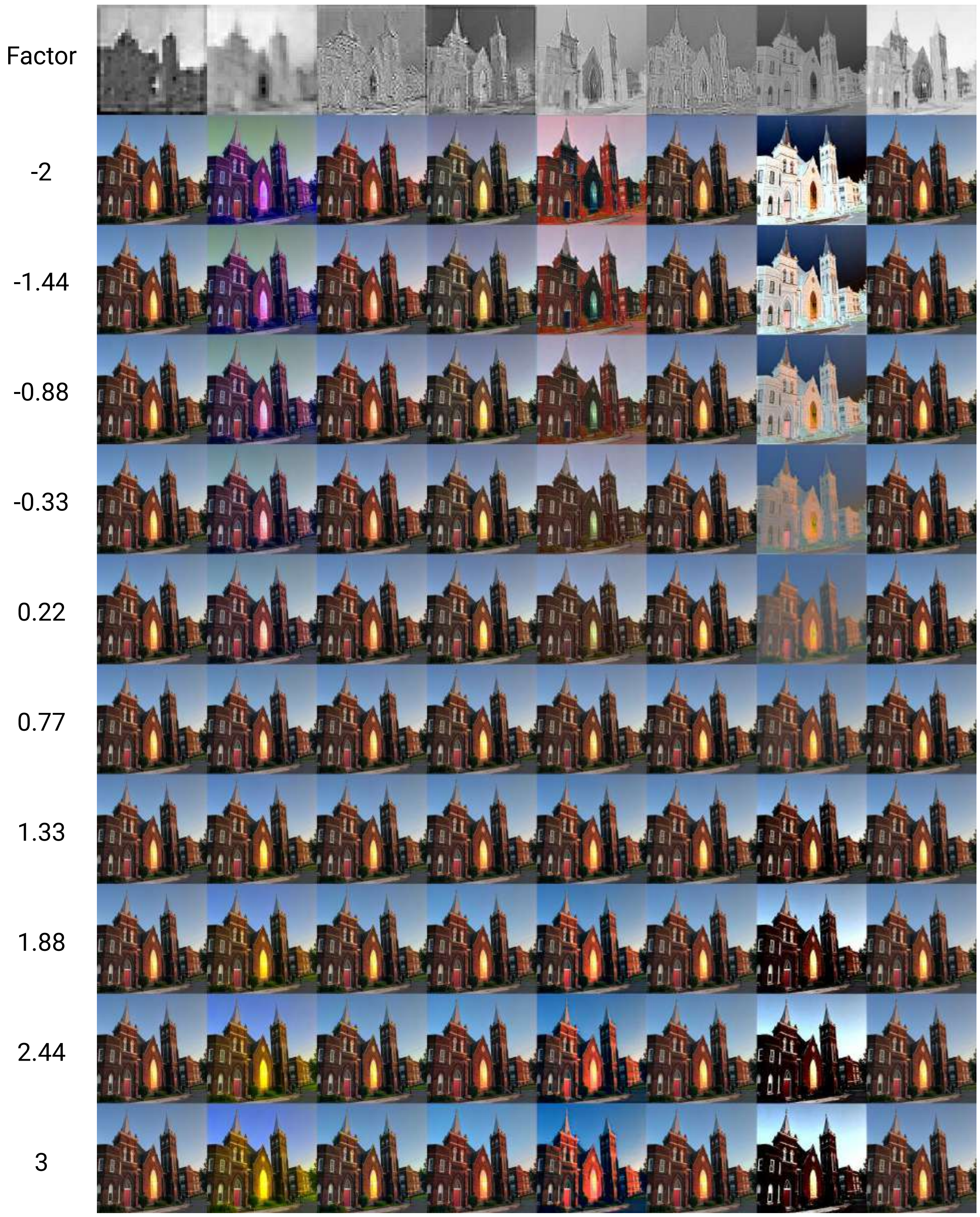
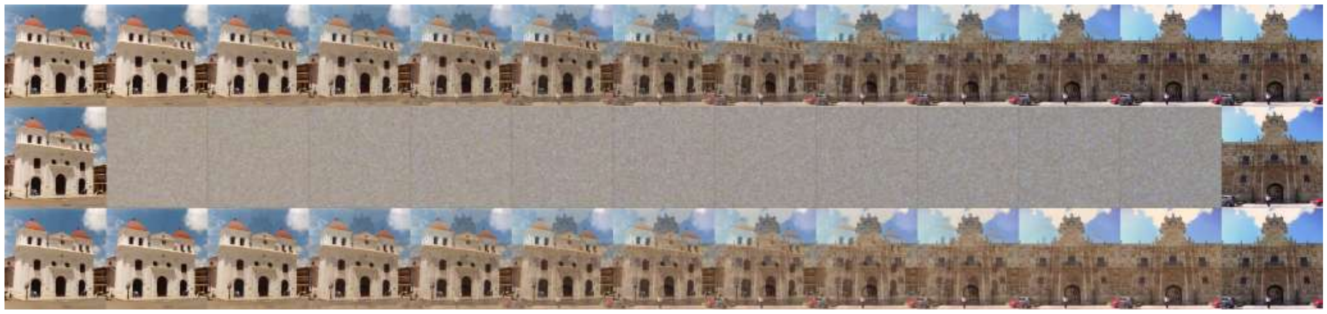
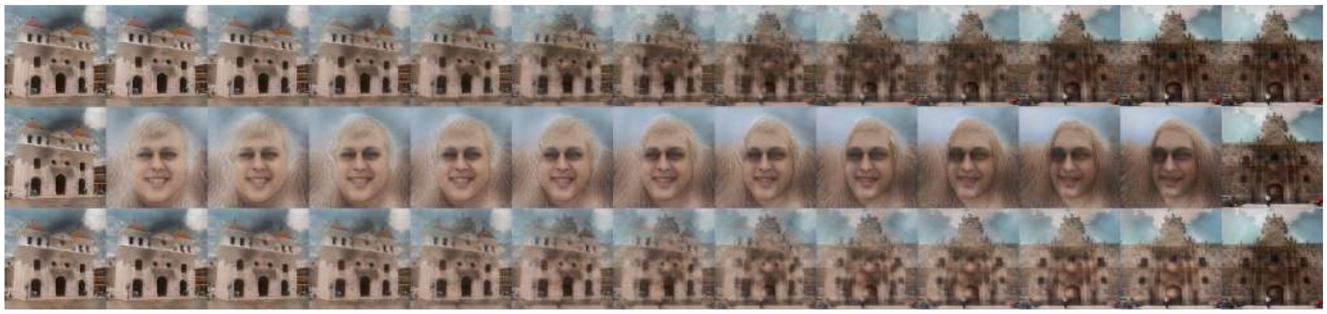
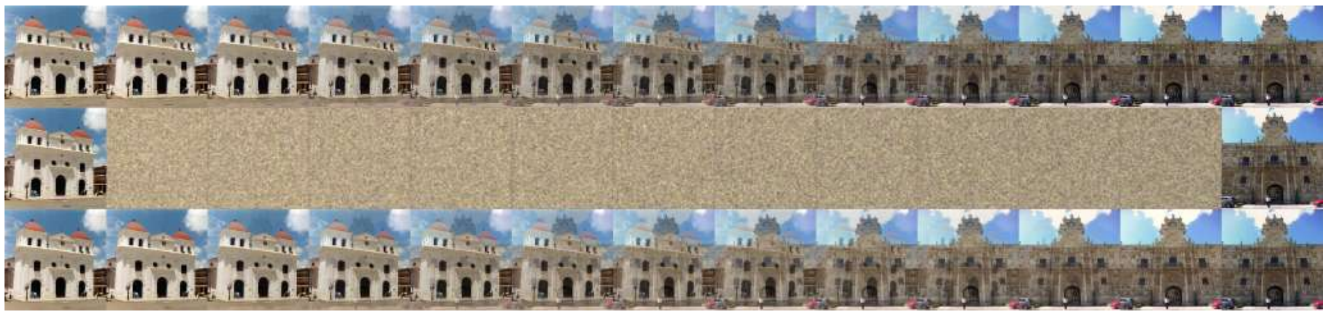


Figure 11: Further results of noise shifting experiments. Here we show the results with a StyleGAN 2 generator pre-trained on the LSUN cat dataset. The semantics are the same as in Figure 9 and Figure 10. Here it is also clearly visible that the noise maps are not only used to encode the content of an image, but that noise is also used to encode color and contrast of images, but for this model different noise maps are used and the color model encoded is again different.


(a) Encoder trained on FFHQ, decoder pre-trained on FFHQ, with StyleGAN 2, and projecting into $\mathcal{W}^+$.


(b) Encoder trained on FFHQ, decoder pre-trained on FFHQ, with StyleGAN 2, projecting into $\mathcal{W}^+$, and our two network strategy.


(c) Encoder trained on LSUN cat, decoder pre-trained on FFHQ, with StyleGAN 2, and projecting into $\mathcal{W}^+$.

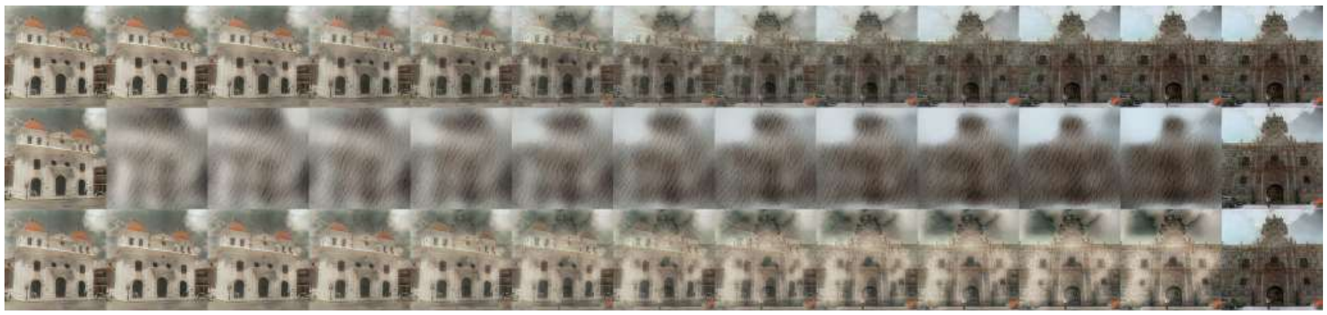

(d) Encoder trained on LSUN cat, decoder pre-trained on FFHQ, with StyleGAN 2, projecting into $\mathcal{W}^+$, and our two network strategy.

Figure 12: Several images showcasing the behaviour of our models when interpolating latent code and noise maps between two input images. We can observe that models trained without the two network strategy do not make any ”intelligent” use of the latent code. The latent code is only used to encode some colors, as can be seen in the second-to-right image in the bottom-most row in a and c, since these images should show the target image, which is the right-most image in each row. For the other models, we can observe that our two network strategy can only be used to faithfully reconstruct images of the same data distribution. A semantically meaningful interpolation can be observed in b, but since the base model was trained on the FFHQ dataset, the latent code can only directly be used to generate faces. The results in d are similar to the results in the main paper, but here we can also see that noise is only rendered on top of the image generated by the latent code.

Training Dataset, StyleGAN Version, Projection Target	FFHQ			Church			Bedroom			Cat		
	FID↓	PSNR↑	SSIM↑	FID↓	PSNR↑	SSIM↑	FID↓	PSNR↑	SSIM↑	FID↓	PSNR↑	SSIM↑
FFHQ, 2, $\mathcal{Z}$	5.87	25.19	0.91	8.28	19.95	0.84	4.96	23.45	0.89	3.90	22.65	0.87
FFHQ, 2, $\mathcal{W}^+$	0.37	29.48	0.96	1.21	23.99	0.93	0.48	27.96	0.96	0.83	25.95	0.93
Church, 2, $\mathcal{Z}$	13.31	21.04	0.86	2.58	22.92	0.89	4.04	22.85	0.89	5.95	21.56	0.86
Church, 2, $\mathcal{W}^+$	2.06	24.18	0.92	0.19	30.12	0.96	0.40	27.73	0.96	0.96	25.40	0.92
Bedroom, 2, $\mathcal{Z}$	8.05	24.07	0.90	3.05	22.45	0.88	2.28	26.17	0.92	2.85	23.70	0.89
Bedroom, 2, $\mathcal{W}^+$	0.75	26.75	0.94	0.62	26.69	0.95	0.17	31.64	0.97	0.53	27.28	0.93
Cat, 2, $\mathcal{Z}$	10.72	24.31	0.91	4.12	22.23	0.87	4.62	24.87	0.90	3.62	24.42	0.90
Cat, 2, $\mathcal{W}^+$	0.92	28.06	0.95	0.48	28.15	0.96	0.25	30.55	0.97	0.48	28.60	0.94

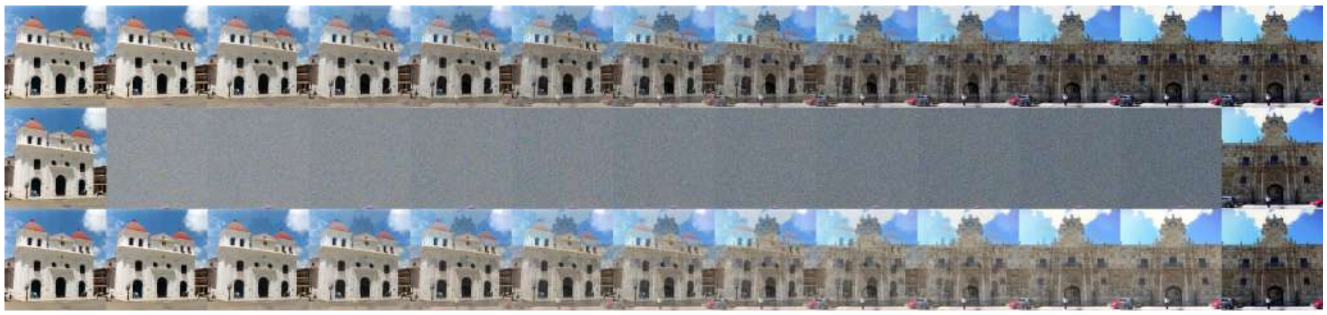
Table 3: The results of further reconstruction experiments. We use a StyleGAN model for decoding, which was pre-trained on the LSUN cat dataset and not updated during the training of the encoder. We train the decoder of our models on different datasets (FFHQ and LSUN datasets), StyleGAN 2, and different projection targets ($\mathcal{Z}, \mathcal{W}^+$) as shown in the first column. We evaluate each model on different datasets and report FID, PSNR, and SSIM.

Training Dataset, StyleGAN Version, Projection Target	FFHQ			Church			Bedroom			Cat		
	FID↓	PSNR↑	SSIM↑	FID↓	PSNR↑	SSIM↑	FID↓	PSNR↑	SSIM↑	FID↓	PSNR↑	SSIM↑
FFHQ, 2, $\mathcal{Z}$	2.89	26.52	0.93	2.59	21.75	0.91	1.33	25.41	0.94	1.66	23.77	0.91
FFHQ, 2, $\mathcal{W}^+$	0.33	29.32	0.96	0.87	24.29	0.94	0.41	27.77	0.96	0.67	25.88	0.93
Church, 2, $\mathcal{Z}$	6.75	21.82	0.89	0.85	25.69	0.92	1.92	23.77	0.92	2.52	22.49	0.89
Church, 2, $\mathcal{W}^+$	2.05	24.34	0.92	0.18	29.62	0.95	0.44	27.74	0.96	0.88	25.36	0.92
Bedroom, 2, $\mathcal{Z}$	5.19	24.55	0.91	1.72	23.86	0.92	0.87	27.63	0.94	1.50	24.22	0.90
Bedroom, 2, $\mathcal{W}^+$	0.83	26.84	0.94	0.72	26.59	0.95	0.16	31.58	0.97	0.53	27.25	0.93
Cat, 2, $\mathcal{Z}$	6.33	25.36	0.92	1.58	24.72	0.92	1.16	26.77	0.94	1.58	25.22	0.91
Cat, 2, $\mathcal{W}^+$	0.56	28.25	0.95	0.36	28.28	0.96	0.17	30.72	0.97	0.38	28.63	0.94

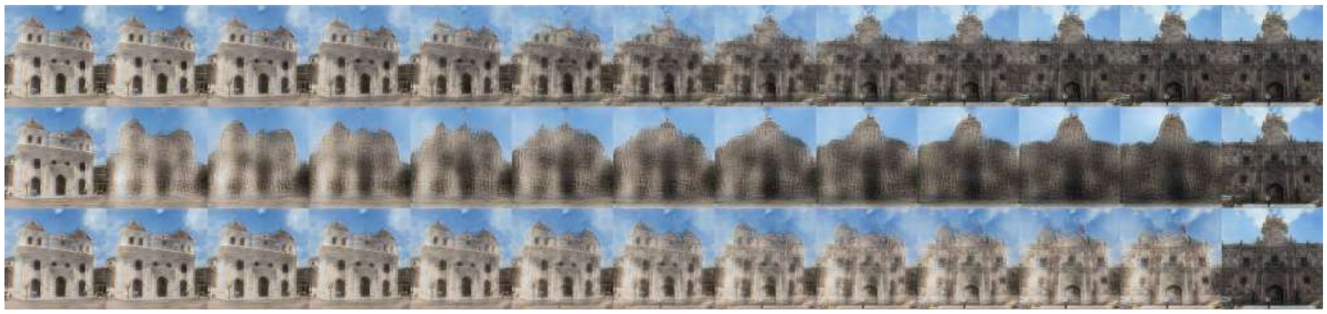
Table 4: The results of further reconstruction experiments. Here, we use a StyleGAN model for decoding, which was pre-trained on the LSUN church dataset and not updated during the training of the encoder. We train the decoder of our models on different datasets (FFHQ and LSUN datasets), StyleGAN 2, and different projection targets ($\mathcal{Z}, \mathcal{W}^+$) as shown in the first column. We evaluate each model on different datasets and report FID, PSNR, and SSIM.

Training Dataset, StyleGAN Version, Projection Target	FFHQ			Church			Bedroom			Cat		
	FID↓	PSNR↑	SSIM↑	FID↓	PSNR↑	SSIM↑	FID↓	PSNR↑	SSIM↑	FID↓	PSNR↑	SSIM↑
FFHQ, 2, $\mathcal{Z}$	24.27	15.35	0.69	38.62	12.76	0.60	41.53	14.15	0.66	29.47	13.77	0.63
FFHQ, 2, $\mathcal{W}^+$	9.73	22.16	0.85	22.53	17.33	0.74	20.35	19.09	0.80	15.20	18.92	0.79
Church, 2, $\mathcal{Z}$	31.02	16.32	0.72	18.50	17.78	0.74	27.47	17.48	0.75	23.33	16.35	0.71
Church, 2, $\mathcal{W}^+$	36.19	18.81	0.79	5.46	21.03	0.84	13.94	19.21	0.81	15.77	18.50	0.78
Bedroom, 2, $\mathcal{Z}$	29.12	15.94	0.68	23.61	15.14	0.66	15.72	17.00	0.70	24.38	15.53	0.66
Bedroom, 2, $\mathcal{W}^+$	16.12	21.23	0.83	15.23	19.46	0.80	6.67	22.17	0.84	10.34	20.76	0.82
Cat, 2, $\mathcal{Z}$	23.55	19.03	0.78	22.14	17.25	0.74	20.52	19.11	0.78	18.81	18.24	0.76
Cat, 2, $\mathcal{W}^+$	20.22	21.75	0.86	15.73	19.64	0.83	10.09	21.30	0.85	9.00	21.19	0.85

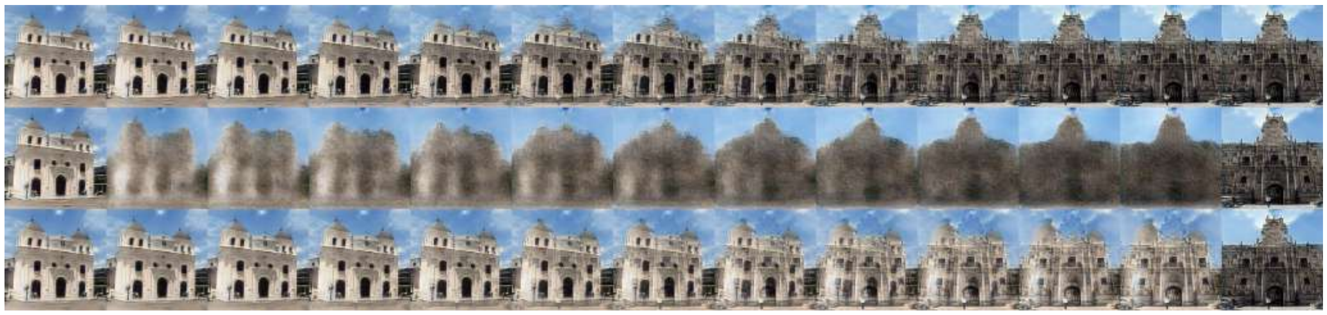
Table 5: Image reconstruction results for models trained with our two network strategy. We can observe that these models are not as versatile as our other image reconstruction models that are not trained to maximise the semantic meaning of the latent code. However, the reconstruction quality is still high on the datasets they have been trained on.



(a) Encoder trained on LSUN church, decoder pre-trained on FFHQ, with StyleGAN 1, and projecting into $\mathcal{W}^+$.



(b) Encoder trained on LSUN church, decoder pre-trained on FFHQ, with StyleGAN 1, projecting into $\mathcal{W}^+$, and our two network strategy.



(c) Encoder trained on LSUN church, decoder pre-trained on FFHQ, with StyleGAN 1, projecting into $\mathcal{W}^+$, and using the learning rate strategy.

Figure 13: Further images showcasing the behavior of our models when interpolating latent code and noise maps between two input images. Here, we used models trained on StyleGAN 1 that project into $\mathcal{W}^+$. We compare plain projection into $\mathcal{W}^+$, our two network strategy and our learning rate strategy to maximise the semantic meaning of the predicted latent code. We can observe that the results of the two network and the learning rate strategy (see b and a) are of similar visual quality. However, the reconstructions in a have a slightly better visual quality and also the interpolations seem to be more reasonable. We conclude that the learning rate strategy is superior to the two network strategy, but it is more difficult to find the correct learning rate ratio, as already discussed in the main paper.