DEEP AUTOMODULATORS

Anonymous authors
Paper under double-blind review

ABSTRACT

We introduce a novel autoencoder model that deviates from traditional autoencoders by using the full latent vector to independently modulate each layer in the decoder. We demonstrate how such an 'automodulator' allows for a principled approach to enforce latent space disentanglement, mixing of latent codes, and a straightforward way to utilize prior information that can be construed as a scale-specific invariance. Unlike GANs, autoencoder models can directly operate on new real input samples. This makes our model directly suitable for applications involving real-world inputs. As the architectural backbone, we extend recent generative autoencoder models that retain input identity and image sharpness at high resolutions better than VAEs. We show that our model achieves state-of-the-art latent space disentanglement and achieves high quality and diversity of output samples, as well as faithfulness of reconstructions.

1 INTRODUCTION

This paper introduces a new generative autoencoder for learning representations of image data sets with an architecture that allows arbitrary combinations of latent representations to generate images (see Fig. 1). We achieve this with an architecture that uses adaptive instance normalization (AdaIn, Huang & Belongie, 2017), and a training method that lets the model learn a highly disentangled latent space by utilizing progressively growing autoencoders (Heljakka et al., 2019) and novel training principles that we introduce in this paper. In a typical autoencoder, input images are encoded into latent space, and the information of the latent variables is then passed through successive layers of decoding until a reconstruction of the input image has been formed. In our model, the latent vector independently *modulates* the statistics of each layer of the decoder, to the same effect as style transfer (Gatys et al., 2016). This allows the layers to work independently of each other.

In image generation, the probability mass representing sensible images (such as human faces) lies concentrated on a low-dimensional manifold. Even if impressive results have been shown for image generation (*e.g.*, by GANs, Goodfellow et al., 2014; Karras et al., 2019), efficient reconstruction and manipulation remain open problems. Deep generative autoencoders provide a principled approach for feature extraction and editing. Thus, an encoder–decoder structure would be a natural starting point for image synthesis and manipulation, and we show that modulation of decoder layers with AdaIn allows more powerful and disentangled representations and image manipulation. Previous works on AdaIn are mostly based on GAN models (Karras et al., 2019; Chen et al., 2019). To work on new input images, GANs either need to be extended with a separate encoder, or inverted with optimization-based methods which are prohibitively slow for many applications.

Autoencoders are typically single-pass encoder–decoder structures, where a sample enters from one end and is reconstructed at the other. The reconstructed samples could be re-introduced to the encoder, repeating the process, and requiring consistency between the passes. Unlike stochastic models (like VAEs, Kingma & Welling, 2014; Rezende et al., 2014), our deterministic model better allows for measuring consistency between the 1st and 2nd pass at any network layer. Our model allows us to mix the latent codes of separate samples and measure the conservation of layer-specific information for each. This enforces disentanglement of layer-specific properties, because we can ensure that the latent code introduced on certain layers will affect only those layers on the 2nd pass. Image autoencoders with convolutional architectures are often used for finding attributes that are disentangled from each other in the latent space. Often, they have relatively poor output image quality, and prior information must be fed in either via class-conditioning or by more complex loss functions for the latent variables. Implicit methods such as GANs show good image quality, but have

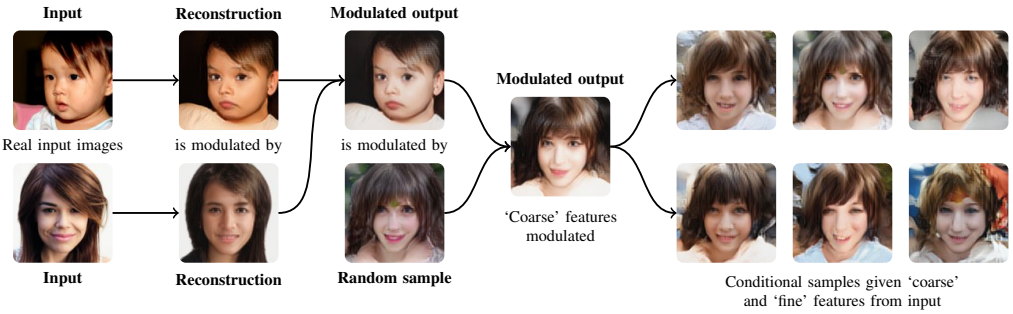


Figure 1: Sketch of the automodulator capabilities. The model can directly encode real (unseen) input images or generate random draws (left). Additionally, inputs can be mixed by modulating one another with different feature levels (center). In the same way, mixing random draws with modulations produces random samples conditioned on features from an input (right).

no built-in encoding mechanism and often cover only a relatively small fraction of the variation in the training data. For the first time, we show a full autoencoder architecture that has a state-of-the-art disentanglement performance even for unsupervised training setup in several data sets, sharp image output quality, good coverage of the variation of the training data, arbitrary mixing of latent representations and a principled approach for incorporating scale-specific prior information.

Our contributions are as follows. *(i)* We introduce the *automodulator*, a new autoencoder-like model with powerful properties not found in regular autoencoders, including style transfer. In contrast to architecturally similar 'style'-based GANs, we can now directly encode and manipulate new inputs. *(ii)* We present methodology that allows training such a model to learn rich latent representation on high resolutions, and specifically explain a novel *multi-pass* training principle that enables the model to learn a more disentangled representation of training data than regular autoencoders. *(iii)* We demonstrate qualitative and quantitative performance and applications on CELEBA-HQ and LSUN Bedrooms and Cars data sets, taking advantage of the highly disentangled model structure.

2 RELATED WORK

Our work builds upon several lines of previous work in unsupervised representation learning. The key concepts related to our approach are variational autoencoders (VAEs, Kingma & Welling, 2014) and generative adversarial networks (GANs, Goodfellow et al., 2014) as described below. VAEs learn an encoder–decoder model, where the encoder maps the data points to a lower dimensional latent space and the decoder maps the latent representations back to the data space. The model is learnt by minimizing the reconstruction error together with a regularization term that encourages the distribution of latents to match a predefined prior. Latent representations often provide useful features for applications (*e.g.*, image analysis and manipulation) and decoding random samples from the prior allows data generation. However, in case of images the generated samples are often blurry and not fully photorealistic and the reconstructions are not perfect either.

Current state-of-the-art in generative image modeling is represented by models based on GANs (Brock et al., 2019; Karras et al., 2019). These recent models generate images with better quality than VAE based models. Nevertheless, the problem with GANs is that that they lack the encoder and hence do not provide a direct way of obtaining the latent representation for a given image. This limits the usefulness of the models. In some cases, a given image can be mapped to the latent space in a semantically meaningful manner via generator inversion but this is a costly iterative process and the result may depend on initialization (Abdal et al., 2019; Creswell & Bharath, 2019).

Thus, to get the best of both worlds, there have been many recent efforts to build hybrid models that combine the properties of VAEs and GANs. Adversarial autoencoders by Makhzani et al. (2016) are among the first efforts and other examples are by Donahue et al. (2017); Dumoulin et al. (2017); Mescheder et al. (2017). All the aforementioned methods learn the mappings between the data space and latent space using three deep networks (*i.e.*, generator, encoder, and discriminator). IntroVAE (Huang et al., 2018) and AGE (Ulyanov et al., 2018) are more compact autoencoders, which both

contain only two deep networks, encoder and decoder, learnt using a combination of reconstruction loss and adversarial loss, which encourages the distributions of real and generated images to match with the prior in latent space. Later, a progressively grown version of AGE, called PIONEER, was proposed by Heljakka et al. (2018; 2019). There is also recent work on vector quantized variational autoencoders (VQ-VAE, Razavi et al., 2019) with a discrete latent space. However, while suitable for image compression, the discrete latent representation cannot *e.g.* be interpolated and hence may not be optimal for semantic image manipulation and learning a disentangled latent space.

The architecture of our generator and AdaIn utilization was inspired by the recent StyleGAN (Karras et al., 2019) but our model does not contain a discriminator. Instead, the model simply contains the generator (*i.e.*, decoder) and an encoder, which are jointly trained applying a similar progressive growing strategy as Heljakka et al. (2018), but modifying the loss formulation. Besides the reconstruction loss and adversarial loss borrowed from Ulyanov et al. (2018) and Heljakka et al. (2018), we can also recirculate style-mixed reconstructions as 'second-pass' training samples in order to better encourage the independence and disentanglement of emerging styles and conservation of layer-specific information. The recirculation idea is biologically motivated and conceptually related to many works, such as that by Hinton & McClelland (1988). Utilizing the outputs of the model as inputs for the next iteration is related to, *e.g.*, Zamir et al. (2017), where feedback is shown to benefit image classification, and in RNN-based methods (Rezende et al., 2016; Gregor et al., 2015; 2016).

3 METHODS

We cover how our approach allows for efficient training via latent space reconstruction errors, learning disentanglement via mixed latent codes, cyclic reconstruction errors at any location of the network, and how to utilize known invariances by the architecture.

3.1 AUTOMODULATOR ARCHITECTURE

Our interest is in unsupervised training of an autoencoder wherein the inputs $\boldsymbol{x}$ are images fed through an encoder ϕ to form a low-dimensional latent space representation $\boldsymbol{z}$ (we use $\boldsymbol{z} \in \mathbb{R}^{512}$ throughout this paper). This representation can then be decoded back into an image $\hat{\boldsymbol{x}}$ through a decoder θ. As the encoding and decoding are deterministic in the AGE-based architecture, we retain, in principle, the full information about the image, at every stage of the processing. This means that when an image has been encoded to a latent vector $\boldsymbol{z}$, decoded back to image space as $\hat{\boldsymbol{x}}$, and re-encoded as latent vector $\boldsymbol{z}'$, it is possible and desirable to require that $\boldsymbol{z}$ is as close to $\boldsymbol{z}'$. This allows for using the reconstruction error in latent space as part of the loss function in any AGE-based architecture. The overall architecture follows the progressively growing Balanced PIONEER form (Heljakka et al., 2018; 2019). As visualized in Fig. 2, the convolutional layers of the encoder and the decoder are faded in gradually during the training, in tandem with the resolutions of training images and generated images. This makes for a stable training scheme.

Adaptive Instance Normalization (AdaIn) We take advantage of AdaIn (Huang & Belongie, 2017) layers inside the model. AdaIn is a simple method to modulate the channel-wise sample mean μ and variance σ^2 of layers separately with 'content' $\boldsymbol{\xi}$ and 'style' $\boldsymbol{y}$ (comprising the marginal mean and variance $\boldsymbol{y} \triangleq (\boldsymbol{\mu}_y, \boldsymbol{\sigma}_y^2)$), as

$$\mathrm{AdaIn}(\boldsymbol{\xi}, \boldsymbol{y}) = \boldsymbol{\sigma}_y \left(\frac{\boldsymbol{\xi} - \mu(\boldsymbol{\xi})}{\sigma(\boldsymbol{\xi})} \right) + \boldsymbol{\mu}_y. \tag{1}$$

A traditional decoder architecture starts from a small-resolution image and expand it layer by layer until the full image is formed, feeding the full information of the latent code through the decoder layer by layer. In contrast, our decoder function $\boldsymbol{\theta}(\boldsymbol{\xi}, \boldsymbol{z})$ separately takes a 'canvas' variable $\boldsymbol{\xi}$ that denotes the content input passed through from the previous layers (see Figs. 2 and 3a). Layer #1 starts from a constant input $\boldsymbol{\xi}^{(0)} \in \mathbb{R}^{4\times4}$. Within the decoder, each residual block also comes with its own fully connected affine mapping layer (incorporated in $\boldsymbol{\theta}(\boldsymbol{\xi}, \boldsymbol{z})$) for Eq. (1), where $\boldsymbol{y}$ is linearly mapped from the input latent vector $\boldsymbol{z}$. We inject input information to the decoder separately for each deconvolution layer, mediated by the affine mapping layers, in order to *modulate* the statistics of the deconvolution layer, hence the name *automodulator*.

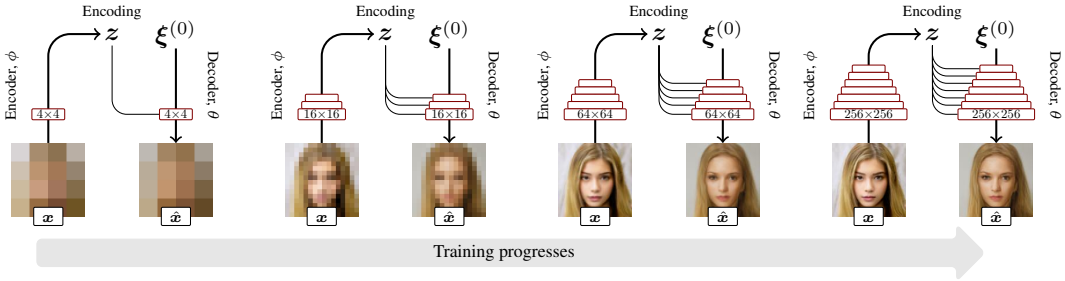


Figure 2: The model grows step-wise during training; the resolution doubles on every step. Input $\boldsymbol{x}$ is encoded into a latent enocoding $\boldsymbol{z}$ (a dimensionality of 512 used throughout this paper). The decoder acts by modulating an empty canvas $\boldsymbol{\xi}^{(0)}$ by the latent encoding and produces the output $\hat{\boldsymbol{x}}$.

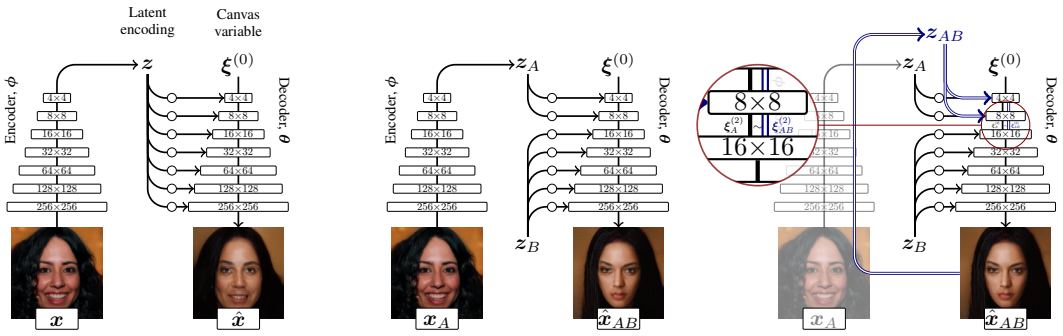
This setup follows the same logic as that of Karras et al. (2019), but we do not require an *ad hoc* disentanglement stack. Here, we focus on pyramidal decoders with monotonically increasing resolution and decreasing number of channels, but any deep decoder would be applicable. The AdaIn-based architecture allows the decoder layers to work independently of each other, enabling finer control over the image information—as covered in the next section—as well as various image mixing schemes and utilization of scale-specific invariances in training data.

3.2 Strong conservation of Cyclic Information: Layer-specific Loss Metrics

Suppose we feed the information to certain decoder layers from a latent encoding $\boldsymbol{z}_A$, and to the rest of the layers from another encoding $\boldsymbol{z}_B \neq \boldsymbol{z}_A$. Then, when we re-encode the result and decode again, we enforce intermediate decoder layer outputs to be consistent between the passes. Thus encouraging the latent space to become hierarchically disentangled with respect to the levels of image detail. One of the many benefits is that we can now carry out conditional sampling by fixing the latent code for specific layers of the decoder, or mix the scale-specific features of two or more input images—impossible feats for a traditional autoencoder where the layers of the decoder are too mutually entangled to allow for this.

This model allows presenting a decoder (Fig. 3b) with N layers split after the j^{th} one as a composition of $\boldsymbol{\theta}_{j+1:N}(\boldsymbol{\theta}_{1:j}(\boldsymbol{\xi}^{(0)}, \boldsymbol{z}_A), \boldsymbol{z}_B)$, in which we can choose $\boldsymbol{z}_A \neq \boldsymbol{z}_B$ (similarly to Karras et al., 2019). Specifically, consider $\boldsymbol{z}_A = \boldsymbol{\phi}(\boldsymbol{x}_A)$ and $\boldsymbol{z}_B = \boldsymbol{\phi}(\boldsymbol{x}_B)$ for (image) inputs $\boldsymbol{x}_A \neq \boldsymbol{x}_B$. Because the earlier layers operate on image content at lower resolutions, we can produce images that mix the 'coarse' features of $\boldsymbol{z}_A$ with 'fine' features of $\boldsymbol{z}_B$. Here we show a 2-way split, but without loss of generality, the split can be made for up to N parts. On the other hand, in AGE-based models, we can decode any latent vector $\boldsymbol{z}$ back to image space as $\boldsymbol{x}$, re-encode it as $\hat{\boldsymbol{z}}$, and minimize the latent reconstruction error between $\boldsymbol{z}$ and $\hat{\boldsymbol{z}}$. Whether the generated images originate from single-source or multi-source (mixed) latents, we can feed them to the encoder for 'discriminative' purposes (via KL divergences) in the same way.

Now, $\boldsymbol{z}$ holds feature information at different levels of detail, some of which are mutually independent. Hence, when re-encoding an image, we should keep the representation of those levels disentangled in $\boldsymbol{z}$, even if they come from separate source images. Assume that the described network reconstructs input samples perfectly, *i.e.* $\boldsymbol{x} = \boldsymbol{\theta}(\boldsymbol{\phi}(\boldsymbol{x}))$. Consider two latents $\boldsymbol{z}_A$ and $\boldsymbol{z}_B$ (either $\boldsymbol{z}_i \sim \mathrm{N}(\mathbf{0}, \mathbf{I})$ or $\boldsymbol{z}_i \sim q_\phi(\boldsymbol{z} \,|\, \boldsymbol{x}_i)$), $\boldsymbol{x} \overset{\text{def}}{=} \boldsymbol{\xi}^{(N)}$ and $\hat{\boldsymbol{z}}_{AB} \sim q_\phi(\boldsymbol{z} \,|\, \boldsymbol{x})\, q_{\theta_{j+1:N}}(\boldsymbol{x} \,|\, \boldsymbol{\xi}^{(j)}, \boldsymbol{z}_B)\, q_{\theta_{1:j}}(\boldsymbol{\xi}^{(j)} \,|\, \boldsymbol{\xi}^{(0)}, \boldsymbol{z}_A)$. Now, between $\boldsymbol{\xi}^{(j)}$ of the first and $\boldsymbol{\xi}^{(j)}$ of the second pass (see Fig. 3c), the mutual information is $I[q_{\theta_{1:j}}(\boldsymbol{\xi}^{(j)} \,|\, \boldsymbol{\xi}^{(0)}, \hat{\boldsymbol{z}}_{AB}); q_{\theta_{1:j}}(\boldsymbol{\xi}^{(j)} \,|\, \boldsymbol{\xi}^{(0)}, \boldsymbol{z}_A)]$. Then, sampling from $\boldsymbol{z}_A$ and $\hat{\boldsymbol{z}}_{AB}$, we have $\mathcal{L}_j = d(\boldsymbol{\theta}_{1:j}(\boldsymbol{\xi}^{(0)}, \hat{\boldsymbol{z}}_{AB}), \boldsymbol{\theta}_{1:j}(\boldsymbol{\xi}^{(0)}, \boldsymbol{z}_A))$ for some distance function d when splitting after layer $\#j$ (here, d is the L2 norm). We can minimize it by gradient descent, consequently maximizing mutual information separately for each j. In other words, the fusion image can be encoded into a new latent vector in such a way that, at each layer, the decoder will treat the new code similarly to the original two separate latent codes (see Fig. 3b). For a perfect network, $\mathcal{L}_j$ can be viewed as layer entanglement error. Randomizing j during the training, we can measure $\mathcal{L}_j$ for any layers of the decoder.



(a) Architecture (b) 1-pass flow (c) 2-pass flow

Figure 3: (a) The autoencoder-like usage of the model. (b) Modulations in the decoder can come from different latent vectors. This can be leveraged in feature/style mixing, conditional sampling, and during the model training (first pass). (c) The second pass during training.

Fully Unsupervised Automodulator Training and Loss For image reconstruction loss, we utilize a progressively-growing variation of the loss function d_ρ of Barron (2019) that generalizes various norms and exposes robustness as an explicit continuous parameter vector $\boldsymbol{\alpha}$. The complete loss functions are (see Heljakka et al., 2019, for discussion)

$$\begin{aligned}\mathcal{L}_\phi &= \max(-M_{\text{gap}}, \mathrm{D}_{\mathrm{KL}}[q_\phi(\boldsymbol{z} \,|\, \boldsymbol{x}) \,\|\, \mathrm{N}(\mathbf{0}, \mathbf{I})] - \mathrm{D}_{\mathrm{KL}}[q_\phi(\boldsymbol{z} \,|\, \hat{\boldsymbol{x}}) \,\|\, \mathrm{N}(\mathbf{0}, \mathbf{I})]) + \lambda_{\mathcal{X}}\, d_\rho(\boldsymbol{x}, \hat{\boldsymbol{x}}),\\ \mathcal{L}_\theta &= \mathrm{D}_{\mathrm{KL}}[q_\phi(\boldsymbol{z} \,|\, \hat{\boldsymbol{x}}) \,\|\, \mathrm{N}(\mathbf{0}, \mathbf{I})] + \lambda_{\mathcal{Z}}\, d_{\cos}(\boldsymbol{z}, \boldsymbol{\phi}(\boldsymbol{\theta}(\mathbf{z}))) + \mathcal{L}_j,\end{aligned} \tag{2}$$

where $\hat{\boldsymbol{x}}_{1:\frac{3}{4}M} \sim q_\theta(\boldsymbol{x} \,|\, \mathbf{z})$ with $\boldsymbol{z} \sim \mathrm{N}(\mathbf{0}, \mathbf{I})$, and $\hat{\boldsymbol{x}}_{\frac{3}{4}M:M} \sim q_\theta(\boldsymbol{x} \,|\, \hat{\boldsymbol{z}}_{AB})$, with a set 3:4 ratio of regular and mixed samples of batch size M, margin $M_{\text{gap}} = 0.5$ and $j \sim \mathrm{U}\{1, N\}$. In our variation of d_ρ, we first learn the $\boldsymbol{\alpha}$ in the lower resolutions (*e.g.*, 4×4). Each α_i corresponds to one pixel. Then, when switching to the higher resolution stage, we take take each parameter α_i that corresponds to pixels $p_{x,y}$ in the lower resolution, to initialize the $\alpha_j^{1\times4}$ that, in the higher resolution, corresponds to $p_{x,y}$, $p_{x+1,y}$, $p_{x,y+1}$ and $p_{x+1,y+1}$, respectively.

3.3 Enforcing Known Invariances at Specific Layers

The architecture and the cyclic training method also allows for a novel principled approach for leveraging known scale-specific invariances in training data. For a known invariance F, assume that we know that for two inputs $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, and outputs $F(\boldsymbol{x}_1) = F(\boldsymbol{x}_2)$, so that we have a perfect bijective encoder $\boldsymbol{\phi}$ such that we can, with slight abuse of notation, represent the relationship as $F(\boldsymbol{z}_1) = F(\boldsymbol{z}_2)$. Furthermore, assume that we know all information about F to be contained on the level of image detail that is represented in decoder layers $\#j{:}k$. Therefore, by extending the notation, $F(\boldsymbol{\xi}_1^{(j:k)}) = F(\boldsymbol{\xi}_2^{(j:k)})$. Finally, assume that the rest of the information in the images can be represented on layers $\#1{:}(j-1)$ and $\#(k+1){:}N$. This situation occurs, *e.g.*, when two images are known to differ only in terms of their high-frequency properties, which can be represented fully in the 'fine' layers. Utilizing the independence of the layers, we can now require that

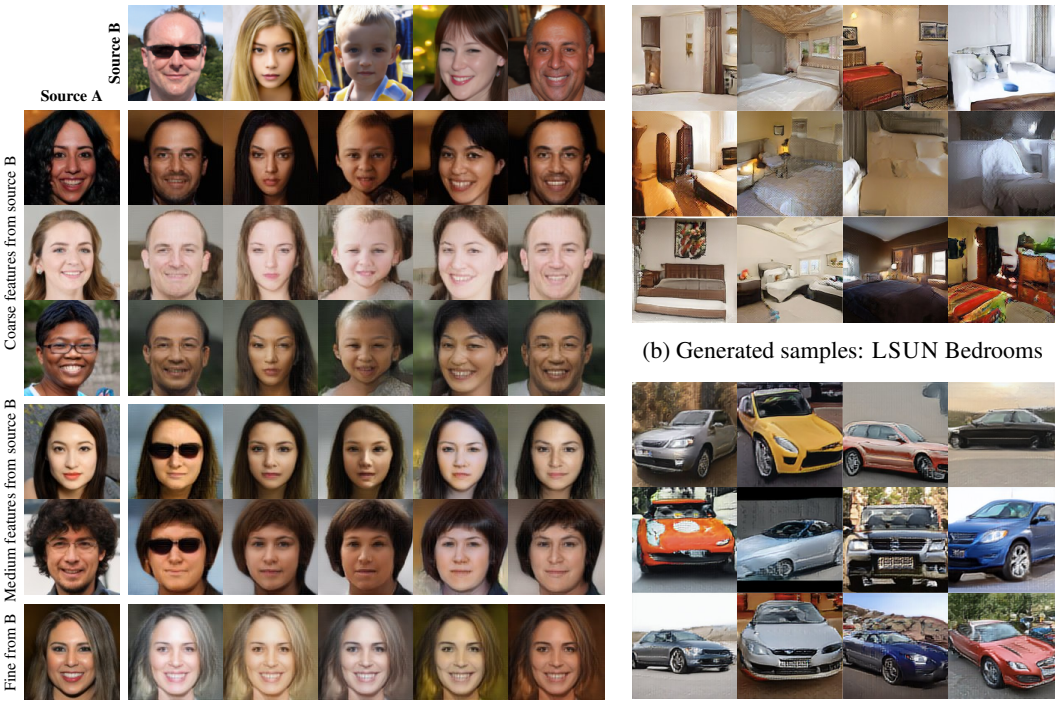
$$\begin{aligned}\boldsymbol{\theta}_{k+1:N}(\boldsymbol{\theta}_{j:k}(\boldsymbol{\theta}_{1:j-1}(\boldsymbol{\xi}^{(0)}, \boldsymbol{z}_2), \boldsymbol{z}_1), \boldsymbol{z}_2) &= \boldsymbol{\theta}_{k+1:N}(\boldsymbol{\theta}_{j:k}(\boldsymbol{\theta}_{1:j-1}(\boldsymbol{\xi}^{(0)}, \boldsymbol{z}_2), \boldsymbol{z}_2), \boldsymbol{z}_2)\\ &= \boldsymbol{\theta}_{1:N}(\boldsymbol{\xi}^{(0)}, \boldsymbol{z}_2)\end{aligned} \tag{3}$$

and vice versa by swapping z_1 and z_2. We turn these into optimization targets

$$\mathcal{L}_{\text{inv}} = d(\boldsymbol{\theta}_{1:N}(\boldsymbol{\xi}^{(0)}, \boldsymbol{z}_1), \boldsymbol{\theta}_{k+1:N}(\boldsymbol{\theta}_{j:k}(\boldsymbol{\theta}_{1:j-1}(\boldsymbol{\xi}^{(0)}, \boldsymbol{z}_2), \boldsymbol{z}_2), \boldsymbol{z}_1)) \quad \text{and} \tag{4}$$

$$\mathcal{L}'_{\text{inv}} = d(\boldsymbol{\theta}_{1:N}(\boldsymbol{\xi}^{(0)}, \boldsymbol{z}_2), \boldsymbol{\theta}_{k+1:N}(\boldsymbol{\theta}_{j:k}(\boldsymbol{\theta}_{1:j-1}(\boldsymbol{\xi}^{(0)}, \boldsymbol{z}_2), \boldsymbol{z}_1), \boldsymbol{z}_2)). \tag{5}$$

In other words, consider a set of images such that their differences occur only on certain scales, and are identical at other scales. Now, we can reserve certain layers to only capture the identical parts. Hence, on those layers, we can use their latent codes interchangeably, without affecting the result of the decoding operation. Then, the result of the decoding is only driven by the latent code that modulates the rest of the layers. Consequently, we push the invariant information to layers $\#j : k$,



(a) Facial feature mixing

(b) Generated samples: LSUN Bedrooms

(c) Generated samples: LSUN Cars

Figure 4: (a) Style-mixing example using the same source images as Karras et al. (2019) (underlining that our model can directly work with real input images). (b–c) Random samples at 256×256.

and the other information *away* from those layers. Of course, this reduces the number of layers available for the rest of the image information, so that we may need to add extra layers to retain the overall decoder capacity. Note that in a pyramidal deconvolutional stack where the resolution increases monotonically, the layers of F can only span one or two consecutive levels of detail, since otherwise there would be 'middle resolutions' that the rest of the architecture could not capture. Finally, note that setting $\boldsymbol{z}_1 = \boldsymbol{z}_2$ with $\boldsymbol{z}_1 \sim q_\theta(\boldsymbol{z} \,|\, \boldsymbol{x} \sim \mathcal{X}_{\text{train}})$ reduces Eqs. (4–5) to the regular sample construction loss, revealing our formulation as a generalization thereof.

4 Experiments

We included both quantitative and qualitative experiments for real-world data sets. First, we assess generation and encoding, whereafter we turn to modulation/style-mixing capabilities.

Metrics and Data Sets We run our experiments on images using CelebA-HQ (Karras et al., 2018), FFHQ (Karras et al., 2019), and LSUN Bedrooms and Cars (Yu et al., 2015). To quantify the image quality and diversity of random draws from the model at 256×256 resolution, we use the Fréchet inception distance (FID, Heusel et al., 2017), which is comparable across models when sample size is fixed (Binkowski et al., 2018). However, FID is known to remain constant under changing precision-recall characteristics, and hence should be treated with caution (Kynkäänniemi et al., 2019). We use LPIPS (Zhang et al., 2018) as the similarity metric, which has better correspondence to human evaluation than traditional L2 metrics. The degree of disentanglement of the latent space is often considered the most important property of a latent variable model. We measure this in terms of Perceptual Path Length (PPL, Karras et al., 2019).

Baseline Methods As baselines, we compare separately against autoencoder and non-autoencoder models. For the former, we compare to Balanced Pioneer by Heljakka et al. (2019), a vanilla VAE, and a more recent Wasserstein Autoencoder (WAE, Tolstikhin et al., 2017). For VAE and WAE, we trained both the automodular and traditional architecture, using 128×128 to make the task easier.

Table 1: Performance in CELEBA-HQ (CAHQ), FFHQ, and LSUN Bedrooms and Cars. We measure LPIPS, Fréchet Inception Distance (FID), and perceptual path length (PPL). Resolution is 256×256, except $^*128\times128$. For all numbers, **smaller is better**.

(a) Encoder–decoder comparison

(Using CAHQ*)	LPIPS (cropped)	FID	PPL
B-PIONEER	**0.092**	**21.51**	92.84
WAE-AdaIn	0.165	100.02	62.17
WAE-classic	0.162	108.93	236.82
VAE-AdaIn	0.267	114.52	83.52
VAE-classic	0.291	173.37	71.75
Automodulator	0.102	36.19	**41.45**

(b) Generative models comparison

	FID (CAHQ)	FID (FFHQ)	FID (Bedrooms)	FID (Cars)	PPL (CAHQ*)	PPL (FFHQ)
StyleGAN	**5.17**	**4.40**	**2.65**	**3.27**	50.08	195.9
PGGAN	7.79	8.04	8.34	8.36	81.33	412.0
GLOW	68.93	—	—	—	138.21	—
B-PIONEER	25.25	—	21.52	42.81	92.84	—
Automodulator	51.96	77.49	35.74	35.61	**41.45**	237.8

For the non-autoencoders, we compare to GLOW (Kingma & Dhariwal, 2018) and two recent GAN models: StyleGAN and Progressively Growing GAN (PGGAN, Karras et al., 2018). Models that only generate samples will produce higher-quality samples than encoder–decoder models, but lack direct reconstruction capabilities.

4.1 ENCODING, DECODING, AND RANDOM SAMPLING

In Table 1a, we compare encoder–decoder performance for the autoencoder models in CELEBA-HQ on 128×128, with our proposed architecture ('AdaIn') and the corresponding regular architecture ('classic'). We measure LPIPS, Fréchet Inception Distance (FID), and perceptual path length (PPL). Our method has the best PPL result, while Balanced PIONEER has the best FID. FID is based on a 50k batch of generated samples compared to training samples. PPL (with $\varepsilon = 10^{-4}$) was calculated with 100k samples, cropped to 64×64 or 128×128 at faces.

Table 1b shows comparison of random sampling (examples in Figs. 4b–4c) performance via FID and latent space structure via PPL for various data sets of 256×256 images on CELEBA-HQ, FFHQ and LSUN Cars and Bedrooms. The performance of the automodulator is comparable to the Balanced PIONEER on most data sets, but the FID is worse in general. The GANs have clearly best FID results on all data sets (NB: a hyper-parameter search with various schemes was used in Karras et al., 2019, to achieve their PPL for FFHQ). We train on the actual 60k training set of FFHQ only (unlike StyleGAN that trained on all 70k images). We also evaluate the 4-way image interpolation capabilities in unseen FFHQ test images (Fig. 12 in the appendix) and observe smooth transitions. We emphasize that in GANs, such interpolations are often made between the codes of generated samples. As such, they cannot tell much about the recall characteristics of those models. The qualitative style-mixing capability and low PPL may indicate a high degree of latent space disentanglement, although the more advanced network architecture comes at the cost of weaker FID.

4.2 STYLE MIXING

We demonstrate the 'style-mixing' capabilities of our model. For comparison with prior work, we use the source images from the StyleGAN paper (Karras et al., 2019). In Fig. 4a, we mix specific input faces so that the 'coarse' (latent resolutions 4×4 – 8×8), 'intermediate' (16×16 – 32×32) or 'fine' (64×64 – 256×256) layers of decoder use one input, and the rest of the layers use the other. Importantly, StyleGAN cannot take real inputs, so it can only mix between random images created by the model itself. For new input images, one must run a separate costly optimization process to determine the most fitting latent code. For our model, those images appear as completely new test images. Additional style mixing results are included in Figs. 13–14 in the appendix.

4.3 ENFORCING INVARIANCES

To demonstrate scale-specific invariances, we utilize the simplest image transformation possible: horizontal flipping. For the CELEBA-HQ face data, this provides us with pairs of images that share every other property except the azimuth rotation angle of the face. Since the original rotation of faces in the set varies, the flip-augmented data set contains faces rotated across a wide continuum of angles. For further simplicity, we make an artificially strong hypothesis that the 2D projected face shape is the only relevant feature at 4×4 scale, and does not need to affect scales finer than 8×8. This lets us enforce the $\mathcal{L}_{\text{inv}}$ loss for layers 1–2. Since we do not want to restrict the scale

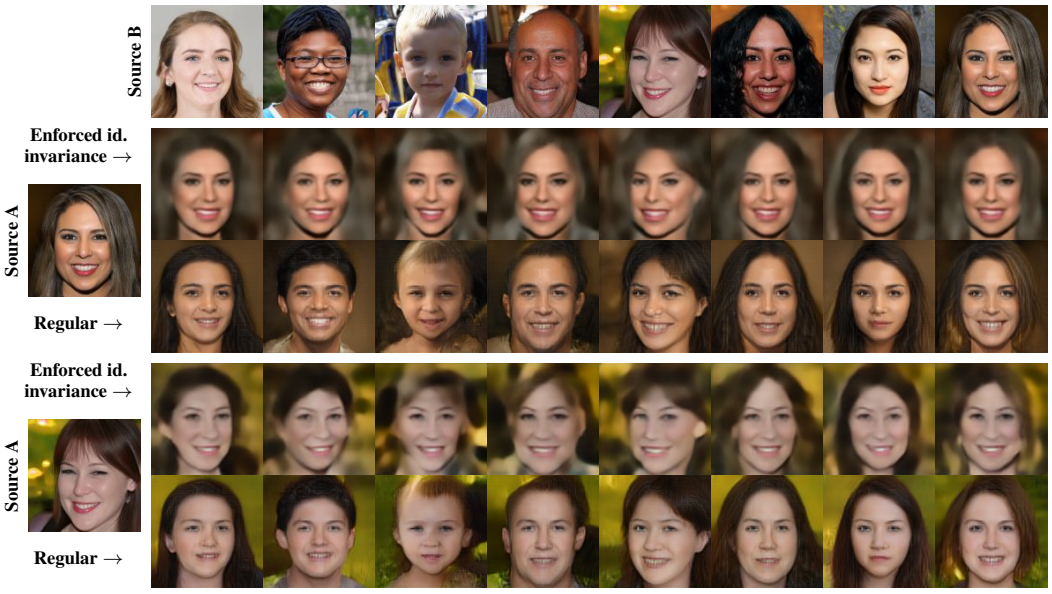


Figure 5: Comparison between driving the face pose with all the coarse-level features of a regular automodulator vs. the automodulator trained with enforced identity invariance under azimuth rotation. When trained for identity invariance, the driving images only change the pose of the source image, whereas the regular training will also affect other coarse characteristics, including identity.

8×8 only for the shape features, we add an extra 8×8 resolution layer to the regular exponentially widening layer hierarchy, so that layer #3 also operates at 8×8, layer #4 at 16×16, *etc.* Now, take $\boldsymbol{z}_2$ that corresponds to the horizontally flipped counterpart of $\boldsymbol{z}_1$. This means that $\boldsymbol{\theta}_{3:N}(\boldsymbol{\xi}^{(2)}, \boldsymbol{z}_1) = \boldsymbol{\theta}_{3:N}(\boldsymbol{\xi}^{(2)}, \boldsymbol{z}_2)$. With our previous architecture choice, we have $j = 1$ and can drop the innermost part of Eq. (4). Hence, our additional encoder loss terms are

$$\mathcal{L}_{\text{inv}} = d(\boldsymbol{\theta}_{1:N}(\boldsymbol{\xi}^{(0)}, \boldsymbol{z}_1), \boldsymbol{\theta}_{3:N}(\boldsymbol{\theta}_{1:2}(\boldsymbol{\xi}^{(0)}, \boldsymbol{z}_1), \boldsymbol{z}_2)) \quad \text{and} \tag{6}$$

$$\mathcal{L}'_{\text{inv}} = d(\boldsymbol{\theta}_{1:N}(\boldsymbol{\xi}^{(0)}, \boldsymbol{z}_2), \boldsymbol{\theta}_{3:N}(\boldsymbol{\theta}_{1:2}(\boldsymbol{\xi}^{(0)}, \boldsymbol{z}_2), \boldsymbol{z}_1)). \tag{7}$$

Fig. 5 shows the results after training with the new loss (50% of the training samples flipped in each minibatch). The model forces the two first decoder layers to affect the face pose only. When modulating the rest of the layers with the original latent, we receive variations of the original face such that they only differ in terms of the pose, while the face identity is conserved.

5 Discussion and Conclusion

In this paper, we proposed the first generative autoencoder model with a hierarchical latent representation that supports controllable image generation and editing, including conditional image sampling by fixing styles of specific layers, and style-mixing of real images. In our model, the latent vector independently modulates each decoder layer. The model outperforms other generative autoencoders in terms of latent space disentanglement and matches them in faithfulness of reconstructions, with slight reduction of output sample quality. We use the term *automodulator* to denote any autoencoder that uses the latent code only to modulate the statistical properties of the information that flows through the layers of the decoder. Such decoders could also include, *e.g.*, 3D or graph convolutions.

Potential future applications include introducing completely independent 'plugin' layers or modules in the decoder, trained afterwards on top of the pretrained base automodulator, leveraging the mutual independence of the layers. The affine maps themselves could also be re-used across domains, potentially offering mixing of different domains. Such examples highlight that the range of applications of our model is far wider than the initial ones shown here, making the family of automodulators a viable alternative to state-of-the-art autoencoders and GANs. Upon acceptance for publication, our source code will be released at http://github.com/anonymized.

REFERENCES

Rameen Abdal, Yipeng Qin, and Peter Wonka. Image2StyleGAN: How to embed images into the StyleGAN latent space? In *International Conference on Computer Vision (ICCV)*, 2019.

Jonathan T. Barron. A general and adaptive robust loss function. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4331–4339, 2019.

Mikolaj Binkowski, Dougal J. Sutherland, Michael Arbel, and Arthur Gretton. Demystifying MMD GANs. In *International Conference on Learning Representations (ICLR)*, 2018.

Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale GAN training for high fidelity natural image synthesis. In *International Conference on Learning Representations (ICLR)*, 2019.

Ting Chen, Mario Lucic, Neil Houlsby, and Sylvain Gelly. On self modulation for generative adversarial networks. In *International Conference on Learning Representations (ICLR)*, 2019.

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, 2019.

Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. Adversarial feature learning. In *International Conference on Learning Representations (ICLR)*, 2017.

Vincent Dumoulin, Ishmael Belghazi, Ben Poole, Olivier Mastropietro, Alex Lamb, Martin Arjovsky, and Aaron Courville. Adversarially learned inference. In *International Conference on Learning Representations (ICLR)*, 2017.

Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2414–2423, 2016.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems (NIPS)*, volume 27, pp. 2672–2680. Curran Associates, Inc., 2014.

Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Rezende, and Daan Wierstra. DRAW: A recurrent neural network for image generation. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, volume 37 of *PMLR*, pp. 1462–1471, 2015.

Karol Gregor, Frederic Besse, Danilo Jimenez Rezende, Ivo Danihelka, and Daan Wierstra. Towards conceptual compression. In *Advances in Neural Information Processing Systems (NIPS)*, volume 29, pp. 3549–3557. Curran Associates, Inc., 2016.

Ari Heljakka, Arno Solin, and Juho Kannala. Pioneer networks: Progressively growing generative autoencoder. In *Asian Conference on Computer Vision (ACCV)*, pp. 22–38, 2018.

Ari Heljakka, Arno Solin, and Juho Kannala. Towards photographic image manipulation with balanced growing of generative autoencoders. *arXiv preprint arXiv:1904.06145*, 2019.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs trained by a two time-scale update rule converge to a local Nash equilibrium. In *Advances in Neural Information Processing Systems (NIPS)*, volume 30, pp. 6626–6637. Curran Associates, Inc., 2017.

Geoffrey E. Hinton and James L. McClelland. Learning representations by recirculation. In *Neural Information Processing Systems (NIPS)*, pp. 358–366. American Institute of Physics, 1988.

Huaibo Huang, Zhihang Li, Ran He, Zhenan Sun, and Tieniu Tan. IntroVAE: Introspective variational autoencoders for photographic image synthesis. In *Neural Information Processing Systems (NeurIPS)*, volume 31, pp. 52–63. Curran Associates, Inc., 2018.

Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pp. 1501–1510, 2017.

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 (ICLR)*, 2018.

Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4401–4410, 2019.

Diederik P. Kingma and Max Welling. Auto-encoding variational Bayes. In *International Conference on Learning Representations (ICLR)*, 2014.

Durk P. Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 31, pp. 10236–10245. Curran Associates, Inc., 2018.

Tuomas Kynkäänniemi, Tero Karras, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Improved precision and recall metric for assessing generative models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, and Ian Goodfellow. Adversarial autoencoders. In *International Conference on Learning Representations (ICLR)*, 2016.

Lars Mescheder, Sebastian Nowozin, and Andreas Geiger. Adversarial variational Bayes: Unifying variational autoencoders and generative adversarial networks. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, volume 70 of *PMLR*, pp. 2391–2400, 2017.

Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In *International Conference on Learning Representations (ICLR)*, 2018.

Ali Razavi, Aäron van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with VQ-VAE-2. *arXiv preprint arXiv:1906.00446*, 2019.

Danilo J. Rezende, Shakir Mohamed, Ivo Danihelka, Karol Gregor, and Daan Wierstra. One-shot generalization in deep generative models. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning (ICML)*, volume 48 of *PMLR*, pp. 1521–1529, 2016.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of the 31st International Conference on Machine Learning (ICML)*, volume 32 of *PMLR*, pp. 1278–1286, 2014.

Ilya Tolstikhin, Olivier Bousquet, Sylvain Gelly, and Bernhard Schölkopf. Wasserstein auto-encoders. *arXiv preprint arXiv:1711.01558*, 2017.

Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. It takes (only) two: Adversarial generator-encoder networks. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI)*, pp. 1250–1257, 2018.

Fisher Yu, Ari Seff, Yinda Zhang, Shuran Song, Thomas Funkhouser, 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.

Amir Roshan Zamir, Te-Lin Wu, Lin Sun, William B. Shen, Bertram E. Shi, Jitendra Malik, and Silvio Savarese. Feedback networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1808–1817, 2017.

Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 586–595, 2018.

A APPENDIX

In the appendix, we include further details on training and complement the results in the main paper with examples of random samples, reconstruction, latent space interpolations, and style mixing.

A.1 TRAINING DETAILS

The training method follows largely those in Heljakka et al. (2019), with progressively growing symmetric encoder and decoder, and decreasing the batch size when moving to higher resolutions. The encoder and decoder consist of 7 blocks each containting two residual blocks with a 3×3 filter. In the encoder, these are followed by a spectral normalization operation (Miyato et al., 2018) and binomial filtering. In the decoder, by AdaIn normalization and binomial filtering. A leaky ReLU ($p = 0.2$) is used as activation. In the encoder, each block halves the resolution of the convolution map, while in the decoder, each block doubles it. The output of the final encoder layers is flattened into a 512-dimensional latent block. As in Karras et al. (2019), the block is mapped by affine mapping layers so that each convolutional layer C in the decoder block is preceded by its own fully connected layer that maps the latent to two vectors each of length N, when N equals the number of channels in C.

Each resolution phase until 32×32 for all data sets use a learning rate $\alpha = 0.0005$ and thereafter 0.001. Optimization is done with ADAM ($\beta_1 = 0, \beta_2 = 0.99, \epsilon = 10^{-8}$). After the first two resolution steps, the KL margin is turned on and fixed to 0.5. The length of training phases amounts to 2.4M training samples until 64×64 resolution phase, which lasts for 10.4M samples. For FFHQ and CelebAHQ, the 128×128 phase uses 13.0M samples while LSUN Bedrooms and Cars use 10.0M samples. The final 256×256 phase uses 7–10M samples for each data set. The training of the final stage was generally cut off when reasonable FID results had been obtained. More training and learning rate optimization would likely improve results. With two Titan V100 GPUs for pre-training stages and four GPUs for the 256×256 stage, the training time for CELEBA-HQ and FFHQ were 18 days each, and for LSUN Bedfrooms 17 days and Cars 19 days.

For evaluating the model after training, a moving expontential running average of generator weights was used, as in both Karras et al. (2018) and Heljakka et al. (2019). For all data sets, training/test set splits were used as given or defined by data set authors, except for LSUN Cars, where we used 4,968,695 samples for training and 552,061 for testing. Note that in regular GAN training, complete data sets are often used without train/test split, leading them to use larger training sets.

For baselines, we used pre-trained models for StyleGAN, PGGAN, PIONEER, and GLOW with default settings provided by the authors. We trained the VAE and WAE models manually. For all VAE baselines the weight for KLD loss was 0.005. For all WAE baseline, we used the WAE-MMD algorithm. The weights for the MMD loss with automodular architecture (WAE-AdaIn) was 4 and with Balanced PIONEER (WAE-classic) architecure it was 2. For VAEs, the learning rate for the encoder was 0.0001, and for the generator 0.0005. For WAEs, the learning rate for both was 0.0002.

For evaluating the encoding and decoding performance, we used 10k unseen test images from the FFHQ data set, cropped the input and reconstruction to 128×128 as in Karras et al. (2019) and evaluated the LPIPS distance between the inputs and reconstructions. We evaluated 50k random samples in all data sets and compare against the provided training set. The GLOW model has not been shown to work with 256×256 resolution on LSUN (the authors show qualitative result only for 128×128). Training of PIONEER did not converge on FFHQ, however we believe this is an issue with the default hyper-parameters not suitable for FFHQ.

The Perceptual Path Length (PPL) was calculated with 100k samples, cropped to 128×128 ($\varepsilon = 10^{-4}$). Pre-trained models for PGGAN and GLOW were used with default settings provided by the authors. Note that we train on the actual 60k training images of FFHQ only (unlike StyleGAN that trained on all 70k images).

A.2 RANDOM SAMPLES

Our model is capable of fully random sampling by specifying $\boldsymbol{z} \sim \mathrm{N}(\mathbf{0}, \mathbf{I})$ to be draws from a unit Gaussian. Fig. 6–8 show samples from an automodulator trained with the FFHQ/CELEBA-

HQ/LSUN data sets up to resolution 128×128. The samples here indicate the full range of samples and face features the model can support.

Figure 6: Random samples from the automodulator trained on FFHQ at a resolution 256×256.

Figure 7: Random samples for an automodulator trained on CELEBA-HQ at resolution 256×256.

(a) LSUN Bedrooms (b) LSUN Cars

Figure 8: Additional samples from an automodulator trained on LSUN Bedrooms and Cars a resolution of at 256×256.

A.3 RECONSTRUCTIONS

We include examples of the reconstruction capabilities of the automodulator at 256×256 in for uncurated test set samples from the FFHQ and CELEBA-HQ data sets. These examples are provided in Figs. 9–10.

A.4 CONDITIONAL SAMPLING

The automodulator directly allows for conditional sampling in the sense of fixing a latent encoding $\boldsymbol{z}_A$, but allowing some of the modulations come from a random encoding $\boldsymbol{z}_B \sim \mathrm{N}(\mathbf{0}, \mathbf{I})$. In Fig. 11, we show conditional sampling of 128×128 random face images based on 'coarse' (latent resolutions

Figure 9: Uncurated examples of reconstruction quality in 256×256 resolution with images from the FFHQ test set (top row).

Figure 10: Uncurated examples of reconstruction quality in 256×256 resolution with images from the CELEBA-HQ test set (top row).

4×4 – 8×8) and 'intermediate' (16×16 – 32×32) latent features of the fixed input. The input image controls the coarse features (such as head shape, pose, gender) on the top and more fine features (expressions, accessories, eyebrows) on the bottom.

A.5 STYLE MIXING AND INTERPOLATION

The well disentangled latent space allows for interpolations between encoded images. We show regular latent space interpolations between the reconstructions of new input images (Fig. 12).

As two more systematic style mixing examples, we include style mixing results based on both FFHQ and LSUN Cars. The source images are unseen real test images, not self-generated images. In Figs. 13 and 14 we show a matrix of cross-mixing either 'coarse' (latent resolutions 4×4 – 8×8) or 'intermediate' (16×16 – 32×32) latent features. Mixing coarse features results in large-scale changes, such as pose, while the intermediate features drive finer details, such as color.

Figure 11: Conditional sampling of 128×128 random face images based on 'coarse' (latent resolutions 4×4 – 8×8) and 'intermediate' (16×16 – 32×32) latent features of the fixed input. The input image controls the coarse features (such as head shape, pose, gender) on the top and more fine features (expressions, accessories, eyebrows) on the bottom.

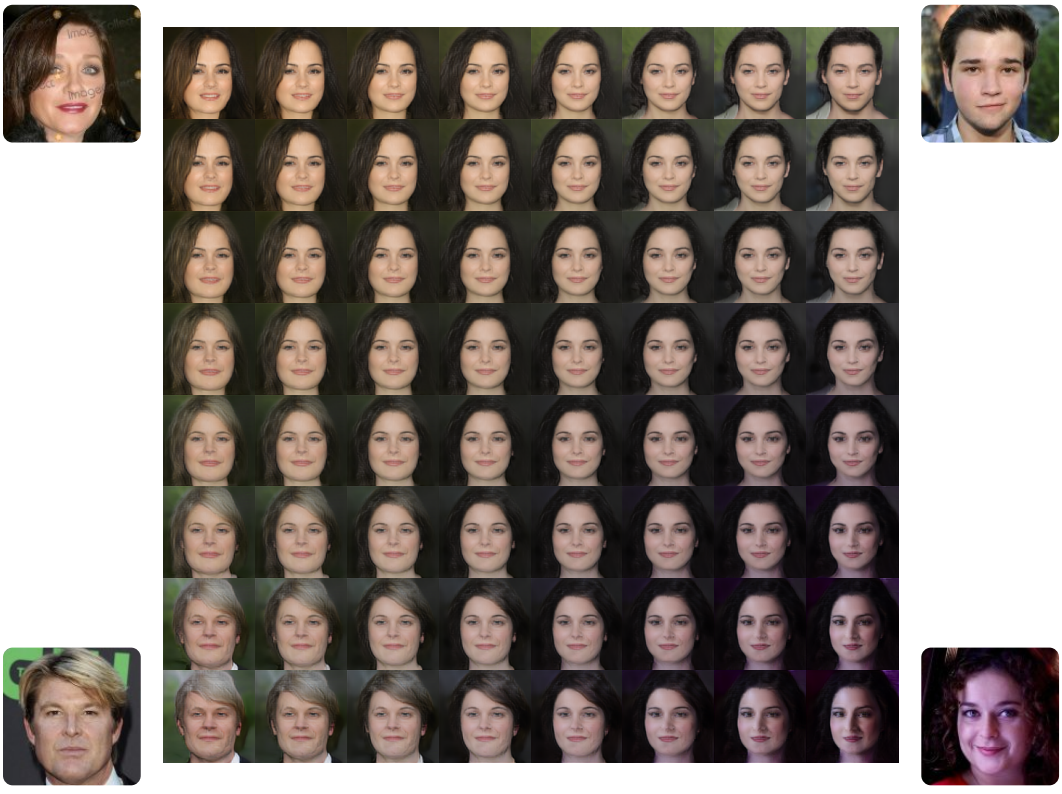
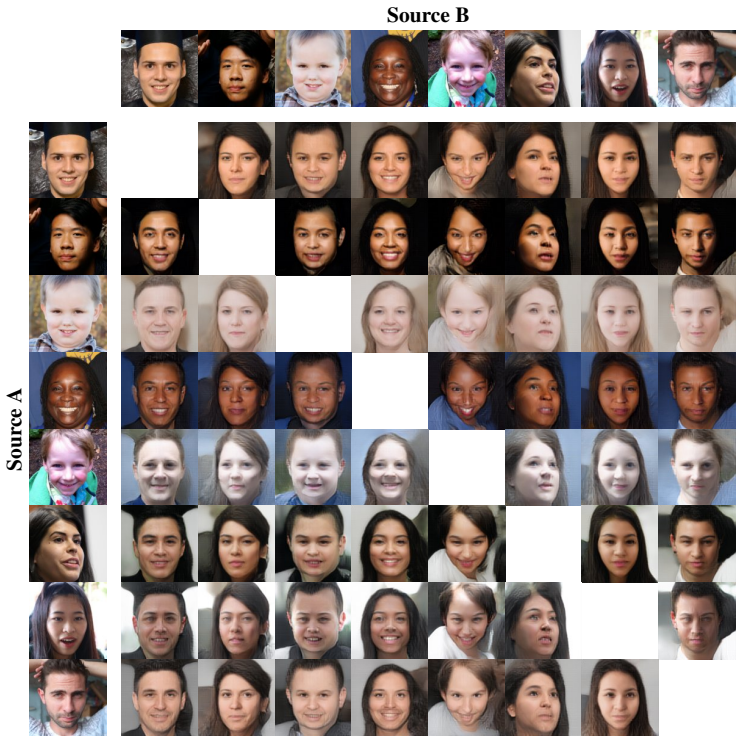
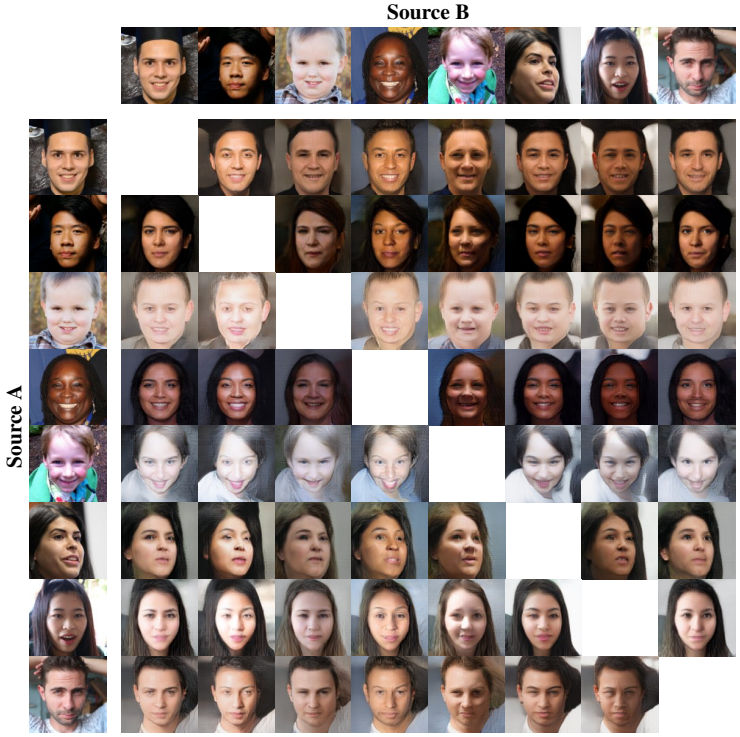


Figure 12: Interpolation between random test set CELEBA-HQ images in 128×128 (in the corners) which the model has not seen during training. The model captures most of the salient features in the reconstructions and produces smooth interpolations at all points in the traversed space.



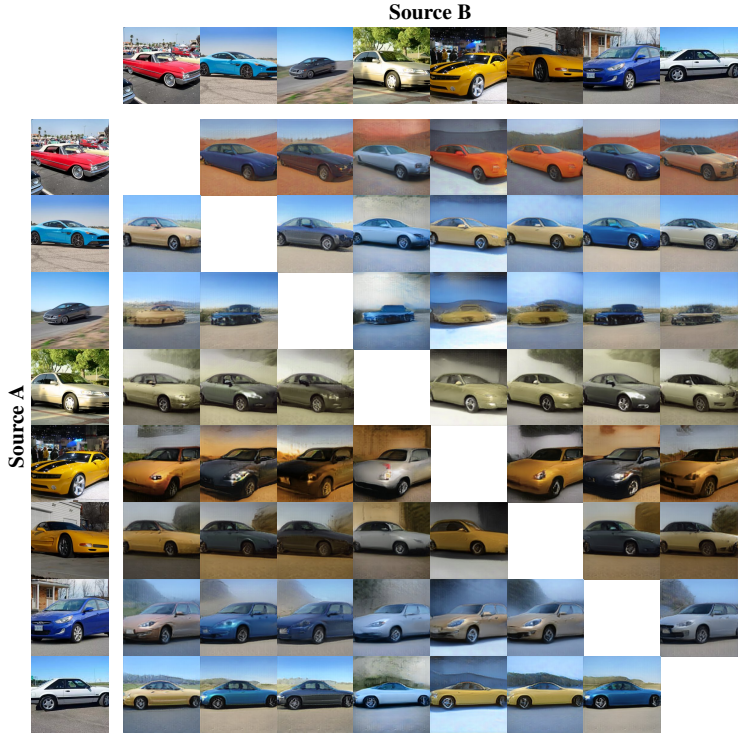
(a) Using ‘coarse’ (latent resolutions 4×4 – 8×8) latent features from B and the rest from A.



(b) Using the ‘intermediate’ (16×16 – 32×32) latent features from B and the rest from A.

Figure 13: Style mixing of FFHQ face images. The source images are unseen real test images, not self-generated images.

(a) Using ‘coarse’ (latent resolutions 4×4 – 8×8) latent features from B and the rest from A. Most notably, the B cars drive the car pose.



(b) Using the ‘intermediate’ (16×16 – 32×32) latent features from B and the rest from A.

Figure 14: Style mixing of LSUN Cars. The source images are unseen real test images, not self-generated images.