Weakly Supervised Content and Style Disentanglement with Gaussian Mixture VAEs

Jan Nikolas Morshuis [1], Colin Samplawski [1, 2], Moin Nabi [1]
[1]SAP Machine Learning Research
[2]University of Massachusetts Amherst
{jan.nikolas.morshuis, m.nabi}@sap.com, csamplawski@cs.umass.edu

Abstract

Disentanglement of the hidden factors of variation within a set of observed images has received a lot of attention in recent years. A key objective is to disentangle the content information of an image from its style information. We introduce a weakly supervised method for this disentanglement objective using Variational Autoencoders (VAEs) that make use of a mixture of Gaussian prior. In contrast to a fully-supervised method, we only use group-level information, that give information about which elements of a batch form a group without giving information about which class the group belongs to. We show that our method is mathematically well-motivated, produces better disentanglement compared to previous VAE-based methods, and can be used in a semi-supervised setting.

1 Introduction

In representation learning, one research direction which has received significant attention is disentanglement [1]. Some approaches focus on learning a disentangled representation in a fully unsupervised manner [1, 2, 3, 4, 5, 6], while others make use of semi-supervised methods [7]. If supervised methods are used, it is often necessary to use toy data sets, in which all factors of variation are known a priori and can be used for training the model. In real-world data, however, it is usually not the case that all factors of variation are known. Therefore, these supervised approaches are not applicable to most real-world datasets.

In this paper, we formalize a weaker supervised approach that only relies on group-level information instead of class labels. With the group-level information, that was first used by [8] for the purpose of disentanglement, it is known which elements of a batch form groups (e.g. which images show the same digit or the same person), but it is unknown to which classes these groups belong to (e.g. which digit or which person is shown in an image). In this work we focus on the disentanglement of the content information of an image from its style information, as in [8, 9, 10]. We define the content information to be the information of an image that strongly depends on its group, and define the style information to be the information of an image that is largely independent of its group. Our method is based on Variational Autoencoders (VAEs) [11] and we separate the latent space into a part that represents the content of the images and a part that represents the style of the images.

Our contributions. i) We introduce and motivate a novel method for content and style disentanglement that makes use of a mixture of Gaussians as a prior for the content part of the latent space and uses only group-level information. ii) We are able to disentangle the content information from the style information better than previous VAE-based methods, which we show qualitatively as well as quantitatively.

Workshop on Information Theory and Machine Learning, 33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

2 Optimization Algorithm of the Gaussian Mixture Model

Our method is based on VAEs. Similar to [8], we separate the latent space into content $z_c \in \mathbb{R}^{2*D_c}$ and style $z_s \in \mathbb{R}^{2*D_s}$, such that the concatenation of z_c and z_s forms the complete latent space $z \in \mathbb{R}^{2*D}$. For an observation $x^{(i)}$, the encoder $f(x^{(i)})$ can predict the represenation $z^{(i)}$ such that $f(x^{(i)}) = z^{(i)} = (z_c^{(i)}, z_s^{(i)})$. Each $z^{(i)}$ consists of the predicted mean $\mu^{(i)} \in \mathbb{R}^D$ as well as the predicted standard deviation $\sigma^{(i)} \in \mathbb{R}^D$, that can be separated into $\mu^{(i)} = (\mu_c^{(i)}, \mu_s^{(i)}), \sigma^{(i)} = (\sigma_c^{(i)}, \sigma_s^{(i)})$ as well.

The derivation of the loss function is similar to [12]. Given the observations x, group-level information g and the parameters ϕ for the encoder and the parameters θ the decoder, the ELBO of the content part of the latent space can be formulated as follows:

$$\begin{aligned} ELBO(x, g, \theta, \phi) &= \mathbb{E}_{z\sim Q_\phi(z|x))}[\log P_\theta(x|z) + \log P_\theta(g|z) + \log P_\theta(z) - \log Q_\phi(z|x)] \\ &= \mathbb{E}_{z\sim Q_\phi(z|x))}[\log P_\theta(x|z) + \log P_\theta(z|g) + \log P_\theta(g) - \log Q_\phi(z|x)] \end{aligned} \tag{1}$$

If we assume all groups to be equally likely, $P_\theta(g)$ will be constant and can therefore be ignored during the optimization process. We assume that the style part of the latent space z_s is independent of g, therefore we can factorize $P_\theta(z|g)$, such that $\log P_\theta(z|g) = \log P_\theta(z_c|g) + \log P_\theta(z_s)$. The ELBO can then be formulated as follows:

$$ELBO(x, \theta, \phi) = \mathbb{E}_{z\sim Q_\phi(z|x))}[\log P_\theta(x|z) + \log P_\theta(z_c|g) + \log P_\theta(z_s) - \log Q_\phi(z|x)] \tag{2}$$

In our method, we use a mixture of Gaussians for the prior $P_\theta(z_c|g)$. Because we have access only to group-level information, rather than class label information, we have to estimate the mean of the mixture coefficients. Similar to [8], for each group G in the set of all groups $\mathcal{G}$, we calculate the product of the inferred Gaussian distributions with mean $\mu_c^{(i)}$ and standard deviation $\sigma_c^{(i)}$ in the content part of the latent space, such that the product is again a Gaussian with mean μ_G and standard deviation σ_G. This can be calculated with the following equation, where $\Sigma = \sigma^2 I$:

$$\mu_G^T \Sigma_G^{-1} = \sum_{i\in G} (\mu_c^{(i)})^T (\Sigma_c^{(i)})^{-1}, \quad \Sigma_G^{-1} = \sum_{i\in G} (\Sigma_c^{(i)})^{-1} \tag{3}$$

For the reparameterization, $\epsilon_1^{(i)} \in \mathbb{R}^{D_c}, \epsilon_2^{(i)} \in \mathbb{R}^{D_s}$ are sampled from the Normal distribution $\mathcal{N}(0, I)$. z can then be reparameterized to z_{rep} as follows:

$$z_{rep}^{(i)} = (z_{c,rep}^{(i)}, \quad z_{s,rep}^{(i)}) = (\mu_c^{(i)} + \epsilon_1^{(i)} \cdot \sigma_c^{(i)}, \quad \mu_s^{(i)} + \epsilon_2^{(i)} \cdot \sigma_s^{(i)}) \tag{4}$$

Note that in contrast to [8], the class latent space is reparameterized using μ_c instead of μ_G. As not all elements of a group share the same μ_c, more content depending information of the image can be represented in the content part of the latent space, which reduces the content information available in the style part of the latent space and therefore improves disentanglement.

The optimization objective for the network is to minimize the negative ELBO, that can be formulated as follows:

$$\mathcal{L}(x^{(i)}, g^{(i)}, \epsilon) = ||x_i - h(z_{rep}^{(i)})||^2 + 1/2 \cdot \Big(\mathrm{tr}(\Sigma^{(i)}) + k \cdot ||\mu_c^{(i)} - \mu_G||^2 + ||\mu_s^{(i)}||^2 - D - \log|\Sigma^{(i)}| \Big) \tag{5}$$

where $h(z_{rep}^{(i)})$ is the decoder, D is the dimension of the latent space, and k is a constant added to adjust how close the representations of the same content should be in the content latent space.

In order to assure that not all $\mu_G = \mu_{G'}$ for distinct groups $G, G' \in \mathcal{G}$, we added a repulsion term $F_R \propto 1/(s^2 + \delta)$ to the loss function, where s is the euclidean distance between μ_G and $\mu_{G'}$ and δ is a small constant added for numerical stability. To assure that μ_G will not be arbitrary large, we also added a regularization term $U \propto ||\mu_G||^2$ to the minimization objective.

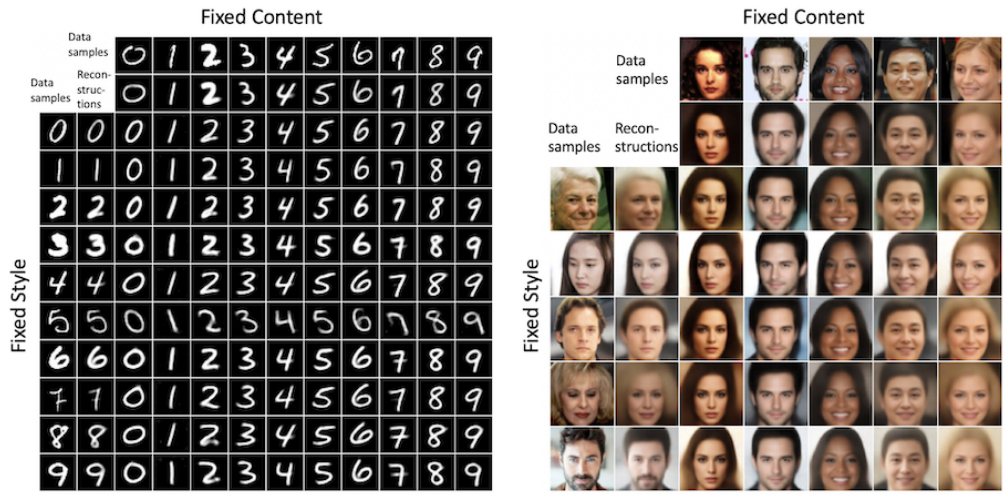

Figure 1: Qualitative results. The first row and the first column of each figure are test samples, the second row and the second column are reconstructions, in the remaining columns the content is kept fixed while in the remaining rows the style is kept fixed.

3 Experiments

In order to evaluate our method, we performed a qualitative and a quantitative analysis. Details on the training procedure and the network architecture can be found in the appendix. Experiments in a semi-supervised setting, further comparisons to the Multi-Level-VAE (ML-VAE) [8], and additional experiments can be found in the appendix.

Qualitative Evaluation As a qualitative evaluation of the disentanglement, we reconstructed several images, interchanged their content parts of the latent space, and reconstructed the images given the altered latent space. The newly obtained images should have the content according to the content latent space information and the style according to the style latent space information. The results on the MNIST dataset [13] and the CelebA dataset [14] are presented in Figure 1. The results suggest that most content related information is preserved when the style part of the latent space is changed. The style part of the latent space does still encode most information that does not depend on the content, like the background, pose and lightning conditions for the CelebA dataset or line-thickness for the MNIST dataset.

Quantitative Evaluation In order to confirm quantitatively that the content latent space represents information about the content and that the style latent space does not, we followed the idea of [8] and trained a classifier on both latent spaces. If the disentanglement of content and style latent information was successful, we expect the classifier to achieve high accuracy when trained on the content latent space z_c and low accuracy when trained on the style latent space z_s, as the latter should not encode significant information about an image's content. The results are shown in Table 1.

Table 1: Classification accuracy for MNIST

Method	Accuracy z_c	Accuracy z_s
Our-method	99.2%	13.9%
ML-VAE [8]	89.3%	16.6%

4 Conclusion

We showed qualitatively and quantitatively that our proposed method of using group-level information to train Gaussian Mixture VAEs is able to disentangle content information from style information. Unlike in previous work [8], with our method more information that depends on the content (like e.g. hair-color) is encoded in the content part of the latent space.

References

[1] F. Locatello, S. Bauer, M. Lucic, G. Raetsch, S. Gelly, B. Schölkopf, and O. Bachem. Challenging common assumptions in the unsupervised learning of disentangled representations. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, volume 97 of *Proceedings of Machine Learning Research*, pages 4114–4124. PMLR, June 2019.

[2] Hyunjik Kim and Andriy Mnih. Disentangling by factorising. *Proceedings of the 35th International Conference on Machine Learning*, 2018.

[3] Abhishek Kumar, Prasanna Sattigeri, and Avinash Balakrishnan. Variational inference of disentangled latent concepts from unlabeled observations. *International Conference on Learning Representations*, 2017.

[4] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. β-vae: Learning basic visual concepts with a variational framework. *International Conference on Learning Representations*, 2017.

[5] Tian Qi Chen, Xuechen Li, Roger B Grosse, and David K Duvenaud. Isolating sources of disentanglement in variational autoencoders. In *Advances in Neural Information Processing Systems 31*, pages 2610–2620. 2018.

[6] Babak Esmaeili, Hao Wu, Sarthak Jain, Alican Bozkurt, N Siddharth, Brooks Paige, Dana H. Brooks, Jennifer Dy, and Jan-Willem van de Meent. Structured disentangled representations. In *Proceedings of Machine Learning Research*, volume 89 of *Proceedings of Machine Learning Research*, pages 2525–2534. PMLR, 16–18 Apr 2019.

[7] Francesco Locatello, Michael Tschannen, Stefan Bauer, Gunnar Rätsch, Bernhard Schölkopf, and Olivier Bachem. Disentangling factors of variation using few labels. *arXiv preprint arXiv:1905.01258*, 2019.

[8] Diane Bouchacourt, Ryota Tomioka, and Sebastian Nowozin. Multi-level variational autoencoder: Learning disentangled representations from grouped observations. *AAAI*, 2018.

[9] Michael F Mathieu, Junbo Jake Zhao, Junbo Zhao, Aditya Ramesh, Pablo Sprechmann, and Yann LeCun. Disentangling factors of variation in deep representation using adversarial training. In *Advances in Neural Information Processing Systems 29*, pages 5040–5048. 2016.

[10] Siddharth Narayanaswamy, T. Brooks Paige, Jan-Willem van de Meent, Alban Desmaison, Noah Goodman, Pushmeet Kohli, Frank Wood, and Philip Torr. Learning disentangled representations with semi-supervised deep generative models. In *Advances in Neural Information Processing Systems 30*, pages 5925–5935. 2017.

[11] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. *International Conference on Learning Representations*, 2014.

[12] Partha Ghosh, Arpan Losalka, and Michael J. Black. Resisting adversarial attacks using gaussian mixture variational autoencoders. *AAAI*, 2019.

[13] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[14] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

[15] Scott E Reed, Yi Zhang, Yuting Zhang, and Honglak Lee. Deep visual analogy-making. In *Advances in Neural Information Processing Systems 28*, pages 1252–1260. 2015.

Appendices

A Qualitative Comparison of the ML-VAE with our method.

A qualitative comparison of our method with the ML-VAE is shown for the CelebA dataset in Figure 2. As previously, the content information is taken to be the identity, while the style information is the information in the image that does not depend on the identity (like pose, lightning conditions, background, etc.).

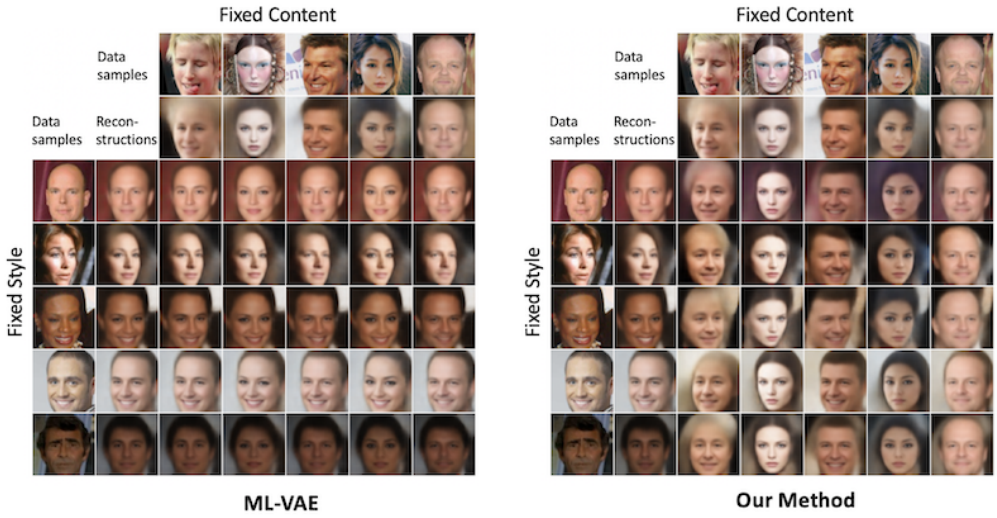

Figure 2: Qualitative results of the ML-VAE compared to our method for the CelebA dataset.

Information about the identity is better preserved using our method. Elements that are strongly correlated with the identity like hair-color, hair-style and skin-tone depend on the content information in our method, while they seem to depend on the style information in the ML-VAE. With our method, however, one can adjust how much information should be stored in the content part of the latent space by adjusting the constant k of Equation 5. This gives more flexibility on what is considered to belong to the content and to the group, such that our method might be a better choice for different datasets.

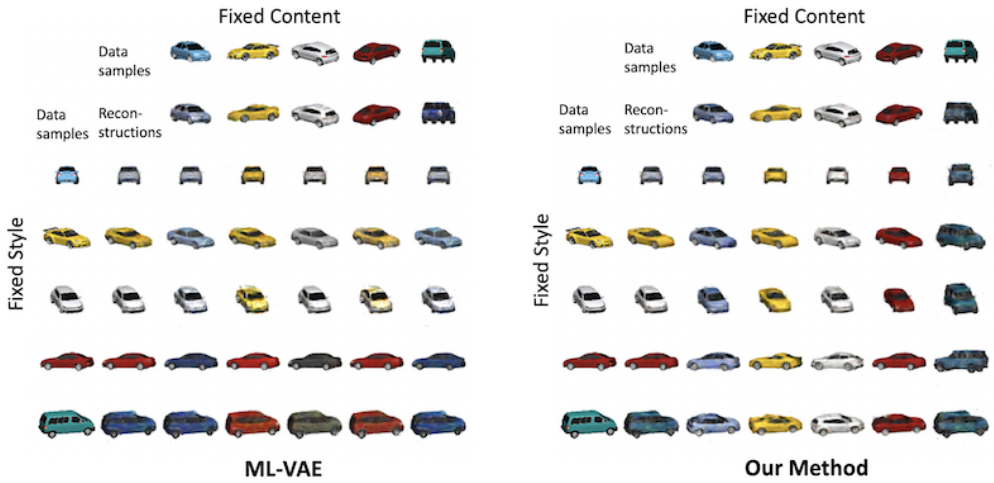

Figure 3: Qualitative results of the ML-VAE compared to our method for the Cars3D dataset.

Further qualitative comparisons between our method and the ML-VAE are shown in Figure 3 for the Cars3D dataset [15]. Our model performs well in disentangling the content information, which is

taken to be the car models with specific colors, from the style information, which is taken to be the pose of the cars. The ML-VAE in contrast was not always able to reconstruct the same car model with the same color for all the different style information of this example. This might improve if group-level information would have been used for the creation of the figure, such that the group mean μ_G could have been calculated for the shown classes. However, as our method does not require the calculation of μ_G at test-time, we decided to compare both approaches without the use of group-level information and use $\mu_G = \mu_c$ for the ML-VAE.

B Semi-Supervised Setting

We tested our method in a semi-supervised setting as well. We made the group-level information only available for 10% of the data. For the data, for which no group-level information was available, we set $\mu_G = 0$ in Eq. 5.

Qualitative results of this experiment are shown in Figure 4. The content information (digit-class) does not always stay the same in all columns. Because the content seems to partly depend on the style as well, the disentanglement is worse than for the fully-supervised example. However, the content part of the latent space still seems to be most important for the representation of the content, while the style part of the latent space seems to be most important for the representation of the style, so the method still works in a semi-supervised setting, albeit less well.

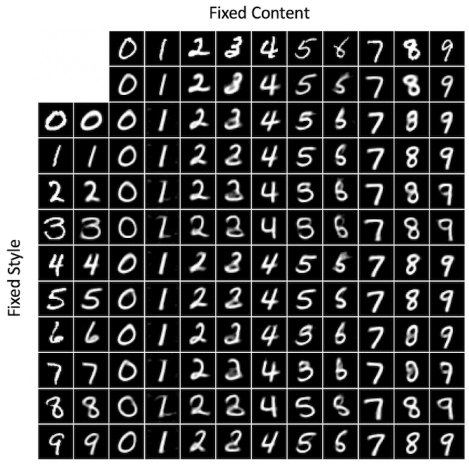

Figure 4: Qualitative results of a semi-supervised setting, where the group-information is only available for 10% of the data.

C Interpolation and Sampling Results

In order to empirically show that our model learns a meaningful latent representation, we show results of interpolation and random sampling in Figure 5. The random sampling was performed by sampling both parts of the latent space z_c and z_s according to the Normal distribution $\mathcal{N}(0, I)$. For the interpolation, the upper-left and the bottom-right images are reconstructions of a test-image of the CelebA-dataset. The remaining images are interpolations between these two images, where z_c is kept fixed for each column and z_s is kept fixed for each row.

It can be seen that the interpolated images do appear as sharp as the original reconstructions and a meaningful reconstruction is achieved for all interpolated latent variables z. The reconstructions of the random sampled latent variable z show a wide variety of modes while still appearing relatively sharp.

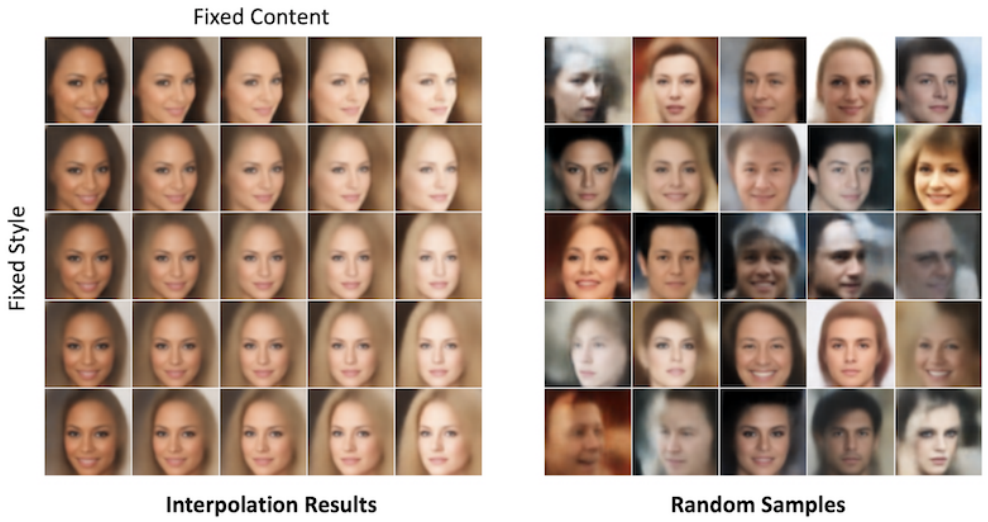

Figure 5: Results of interpolation and random sampling in the latent space. In the left figure, the images on the top-left and on the bottom-right are reconstructions of test-images of the CelebA-dataset, while the remaining images are interpolations between them.

D Training and Network Architecture

Training procedure. We tested our model on the MNIST, Cars3D and the CelebA datasets. For MNIST, we trained on the 50000 training images and used examples from the 10000 test images for the creation of the plot in Figure 1. For the Cars3D dataset, which consists of 183 classes of different kinds of animated cars, we trained on 160 classes and used examples of the remaining 23 classes for the creation of the Figure 1. For the CelebA dataset, we used the images of the 8192 identities in the training set for training and used examples of the 1000 identities of the test set for the creation of the figures. All images of the CelebA dataset were centercropped to 140×140 pixels and then resized to 64×64 pixels.

During training, we sample batches of 256 images for most experiments and batches of 512 images for the semi-supervised experiment. We then apply the encoder to the images, such that we get the inferred values z for the latent space. We then calculate μ_G and Σ_G for every group $G \in \mathcal{G}$ according to Equation 3. We reparameterize z as shown in Equation 4 to get $z_{rep}^{(i)}$ for every sample i. The loss can then be calculated using Equation 5. The adjustable constant k of Equation 5 was defined to be $k = 2$ for all MNIST experiments, $k = 4$ for the Cars3D dataset, and $k = 1.5$ for the CelebA dataset. We train for 200 epochs on MNIST and CelebA and for 600 epochs for Cars3D.

The gradient of the loss is then used to update the value of the parameters of the encoder and the decoder. An Adam optimizer with an initial learning rate of 0.001 is used for this purpose. For the CelebA dataset and for MNIST, the learning rate is divided by 5 after 50 and after 100 epochs. For the Cars3D data, we divide the learning rate by a factor of 5 after 200 and after 400 epochs.

To ensure that the group-level information can be used during training, several elements of one class have to be in one batch. For the CelebA dataset with 8192 classes (identities), it will be rare that several elements of one class will be in one batch if a random data-sampler is used. We therefore make use of a customized data-sampler, that samples the different classes in groups of 10 whenever possible (e.g. there have to be at least 10 samples of the class in the remaining dataset), such that the group-level information can be used for most samples in a batch. For the MNIST dataset and the Cars3D dataset, the common random-data-sampler is used.

Network architecture Table 2 gives an overview of the network architectures that were used for the different datasets. All networks consist of convolutional layers (Conv) that have a kernel-size of 5, a stride of 2, and zero-padding of 2. These convolutional layers are followed by Batch Normalization (BN), which are then followed by the non-linear activation function ReLU. Each encoder has two fully connected linear layers that each use the output of the last ReLU of the encoder and map it to

the content part and the style part of the latent space respectively. For the Cars3D dataset, we chose the content and style latent dimension to be $M = 16$ (32 in total), for the CelebA dataset we chose the latent dimension to be $M = 32$ (64 in total).

In the decoder, two fully connected layers map the content and the style part of the latent space to $8 \times 8 \times 128$ dimensions for MNIST or $8 \times 8 \times 256$ dimensions for the Cars3D dataset and CelebA. The outputs of the fully-connected layers are then concatenated. Transposed Convolutional layers ($\text{ConvT}_{\text{output layers}}$) with a stride of 2, a kernel-size of 4, and zero-padding of 1 are then used for the rest of the decoding process. These layers are again followed by Batch-Normalization (BN) and ReLU-activation functions. The last ConvT-layer uses a kernel-size of 5, a stride of 1, and zero-padding of 1. After the last ConvT-layer, we apply a customized non-linearity function, that has a slop of 0.01 for values smaller than -1, a slope of 0.5 for values between -1 and 1, and a slope of 0.01 for values higher than 1. The value of the non-linearity at -1 is 0. At 1 the function has a value of 1.

Table 2: Network architecture for the different datasets. We chose $M = 16$ for the Cars3D dataset and $M = 32$ for the CelebA dataset.

	MNIST	Cars3D/CelebA
Encoder	$x \in \mathbb{R}^{32\times32}$ $\rightarrow \text{Conv}_{32} \rightarrow \text{BN} \rightarrow \text{ReLU}$ $\rightarrow \text{Conv}_{64} \rightarrow \text{BN} \rightarrow \text{ReLU}$ $\rightarrow \text{Conv}_{128} \rightarrow \text{BN} \rightarrow \text{ReLU}$ $\rightarrow \text{Conv}_{256} \rightarrow \text{BN} \rightarrow \text{ReLU}$ $\hookrightarrow \text{FC}_{8\times2} \rightarrow z_c$ $\hookrightarrow \text{FC}_{8\times2} \rightarrow z_s$	$x \in \mathbb{R}^{64\times64\times3}$ $\rightarrow \text{Conv}_{64} \rightarrow \text{BN} \rightarrow \text{ReLU}$ $\rightarrow \text{Conv}_{128} \rightarrow \text{BN} \rightarrow \text{ReLU}$ $\rightarrow \text{Conv}_{256} \rightarrow \text{BN} \rightarrow \text{ReLU}$ $\rightarrow \text{Conv}_{512} \rightarrow \text{BN} \rightarrow \text{ReLU}$ $\hookrightarrow \text{FC}_{M\times2} \rightarrow z_c$ $\hookrightarrow \text{FC}_{M\times2} \rightarrow z_s$
Decoder	$z_{c,rep} \in \mathbb{R}^8 \rightarrow \text{FC}_{8\times8\times128}$ $z_{s,rep} \in \mathbb{R}^8 \rightarrow \text{FC}_{8\times8\times128}$ $\rightarrow \text{Concatenate}_{8\times8\times256}$ $\rightarrow \text{ConvT}_{128} \rightarrow \text{BN} \rightarrow \text{ReLU}$ $\rightarrow \text{ConvT}_{64} \rightarrow \text{BN} \rightarrow \text{ReLU}$ $\rightarrow \text{ConvT}_{1}$	$z_{c,rep} \in \mathbb{R}^M \rightarrow \text{FC}_{8\times8\times256}$ $z_{s,rep} \in \mathbb{R}^M \rightarrow \text{FC}_{8\times8\times256}$ $\rightarrow \text{Concatenate}_{8\times8\times512}$ $\rightarrow \text{ConvT}_{256} \rightarrow \text{BN} \rightarrow \text{ReLU}$ $\rightarrow \text{ConvT}_{128} \rightarrow \text{BN} \rightarrow \text{ReLU}$ $\rightarrow \text{ConvT}_{64} \rightarrow \text{BN} \rightarrow \text{ReLU}$ $\rightarrow \text{ConvT}_{3}$

Classifier. The network used for the classifier takes a part of the latent space as input (either z_c or z_s) and consists of 2 linear layers, each having 256 hidden units. Each linear unit is followed by a batch normalization, which is then followed by a Leaky-ReLU with a slope of 0.2. The last layer is again a linear layer, which maps the 256 hidden units from the second linear layer to the number of classes.

We train two classifiers for each model, one for the content part of the latent space and one for the style part of the latent space. We use Adam optimizers with learning rates of $1e-5$ and train the classifiers for 8 epochs using batches of size 128 on the model's encoding of the training data of the MNIST dataset. We then test the accuracy of the classifiers using the model's encoding of the test data of the MNIST dataset and reported the results for both classifiers in Table 1.