Disentangling Factors of Variation by Mixing Them

Qiyang Hu[1]* Attila Szabó[1]* Tiziano Portenier[1] Paolo Favaro[1]
[1]University of Bern, Switzerland
[1]{hu, szabo, portenier, favaro}@inf.unibe.ch

Matthias Zwicker[2]
[2]University of Maryland, USA
[2]zwicker@cs.umd.edu

Abstract

We propose an approach to learn image representations that consist of disentangled factors of variation without exploiting any manual labeling or data domain knowledge. A factor of variation corresponds to an image attribute that can be discerned consistently across a set of images, such as the pose or color of objects. Our disentangled representation consists of a concatenation of feature chunks, each chunk representing a factor of variation. It supports applications such as transferring attributes from one image to another, by simply mixing and unmixing feature chunks, and classification or retrieval based on one or several attributes, by considering a user-specified subset of feature chunks. We learn our representation without any labeling or knowledge of the data domain, using an autoencoder architecture with two novel training objectives: first, we propose an invariance objective to encourage that encoding of each attribute, and decoding of each chunk, are invariant to changes in other attributes and chunks, respectively; second, we include a classification objective, which ensures that each chunk corresponds to a consistently discernible attribute in the represented image, hence avoiding degenerate feature mappings where some chunks are completely ignored. We demonstrate the effectiveness of our approach on the MNIST, Sprites, and CelebA datasets.

1. Introduction

Deep learning techniques have led to highly successful natural image representations, some focusing on synthesis of detailed, high resolution images of photographic quality [5, 14], and others on disentangling image features into semantically meaningful properties [6, 22, 25].

In this paper, we learn a disentangled image representation that separates the feature vector into multiple chunks, each chunk representing intuitively interpretable properties, or factors of variation, of the image. We propose a completely unsupervised approach that does not require any labeled data, such as pairs of images where only one factor of variation changes (different viewpoints, for example) [22, 28]. The basic assumption of our technique is that images can be represented by a set of factors of variation, each one corresponding to a semantically meaningful image attribute. In addition, each factor of variation can be encoded using its own feature vector, which we call a feature chunk. That is, images are simply represented as concatenations of feature chunks, in a given order. We obtain disentanglement of feature chunks by leveraging autoencoders, and as a key contribution of this paper, by developing a novel invariance objective. The goal of the invariance objective is that each attribute is encoded into a chunk invariant to changes in other attributes, and that each chunk is decoded into an attribute invariant to changes in other chunks. We implement this objective using a sequence of two feature mixing and unmixing autoencoders.

The invariance objective using feature mixing on its own, however, does not guarantee that each feature chunk represents a meaningful factor of variation. Instead, the autoencoder could represent the image with a single chunk, and ignore all the others. This is called the shortcut problem [28]. We address the shortcut problem with a classification constraint, which forces each chunk to have a consistent, discernible effect on the generated image.

We demonstrate successful results of our approach on several datasets, where we obtain representations consisting of feature chunks that determine semantically meaningful image properties. In summary, we make the following contributions: *1)* A novel architecture to learn image representations of disentangled factors of variation without using any annotation or data domain knowledge, and where the representation consists of a concatenation of a fixed number of feature chunks. Our approach can learn several factors of variation simultaneously; *2)* A novel invariance objective to obtain disentanglement by encouraging invariant encoding and decoding of image attributes and feature chunks, respectively; *3)* A novel classification constraint to ensure that each feature chunk represents a consistent, discernible factor of variation of the represented image; *4)* An evaluation on the MNIST, Sprites, and CelebA datasets to demonstrate

*The authors contributed equally.

the effectiveness of our approach.

2. Related work

Autoencoders. Our architecture is built on autoencoders [4, 12, 1], which are neural networks with two main components: an encoder and a decoder. The encoder is designed to extract a feature representation of the input (image), and the decoder translates the features back to the input. Different flavors of autoencoders have been trained to perform image restoration [30, 21, 3] or image transformation [11]. While basic autoencoders do not impose any constraints on the representation itself, variational autoencoders [16] add a generative probabilistic formulation, which forces the representation to follow a Gaussian distribution and allows sampling images by applying the decoder to a Gaussian vector sample. Thanks to their flexibility, autoencoders have become ubiquitous tools in large systems for domain adaptation [33, 15], or unsupervised feature learning [23]. Autoencoders are also used to learn feature disentangling [25, 22, 28]. In our work we also use them as feature extractors. Our contribution is a novel unsupervised training method that ensures the separation of factors of variation into several feature chunks.

GANs. Generative Adversarial Networks (GANs) [9] are designed to provide samples from a data distribution specified as a finite set of real data samples. They use two competing neural networks: a generator translates input noise vectors into fake data samples, while a discriminator tries to distinguish fake samples from real ones. In the ideal case, the trained generator produces convincing data samples of the real data distribution, and the trained discriminator cannot tell them apart from real ones. GANs have been successful at image to image translation [13], learning representation [24], sampling images from a specific domain [33], or ensuring that image-feature pairs have the same distribution when computing one from another [8]. As the adversarial loss constrains the distribution of the generated data but not the individual data samples, it allows to reduce the need for data labeling. In particular, Shrivastava *et al*. [26] use GANs to transfer known attributes of synthetic, rendered examples to the domain of real images, thus creating virtually unlimited datasets for supervised training. In our work we use GANs to enforce that images look realistic when their attributes are transferred.

Disentangling. There are many methods [29, 31] that disentangle factors of variation by using manual annotation. Kulkarni *et al*. [17] sample the data during the training, such that only one factor changes within a minibatch. They associate a feature chunk to the variation of the images in the minibatch. One of the most immediate methods for disentangling is to mix the feature encodings of two input images with common known attributes in an autoencoder [25] and then train a decoder to map the mixed features to the ground truth image with mixed attributes. In other methods, GANs and adversarial training have been leveraged to reduce the need for complete labeling of all factors of variation. For example, Mathieu *et al*. [22] apply adversarial training on the image domain, while Denton *et al*. [7] propose adversarial training on the feature domain. Szabó *et al*. [28] studied the ambiguities in weakly supervised disentanglement. They can provably avoid a degenerate solution called the shortcut problem, where the complete image representation is condensed in only one feature chunk.

In some approaches, the physics of the image formation model is integrated into the network training, with factors like the depth and camera pose [32] or the albedo, surface normals and shading [27]. Shu *et al*. [27] do no use any label from the training data. However, an externally trained 3D morphable model guides the training, which is also a form of annotation.

By maximizing the mutual information between synthesized images and latent features, InfoGAN [6] makes the latent features interpretable as semantically meaningful attributes. InfoGAN is completely unsupervised, but it does not include an encoding stage. In contrast, we build on an autoencoder, which allows us to recover the disentangled representation from input images, and swap attributes between them. In addition, we use a novel classification constraint instead of the feature consistency in InfoGAN.

Two recent techniques, β-VAE [10] and DIP-VAE [18], build on variational autoencoders (VAEs) to disentangle interpretable factors in an unsupervised way, similarly to our approach. They encourage the latent features to be independent by generalizing the KL-divergence term in the VAE objective, which measures the similarity between the prior and posterior distribution of the latent factors. Instead, we build on mixing autoencoders [25] and adversarial training [9]. We encourage disentanglement using an invariance objective, rather than trying to match an isotropic Gaussian prior. Notice that our feature space is only designed for attribute transfer and not for sampling. Finally, we can use high-dimensional feature chunks, while in [10] and [18] the chunks are one-dimensional.

3. Unsupervised Disentanglement of Factors of Variation

A representation of images where the factors of variations are disentangled can be exploited for various computer vision tasks. At the image level, it allows to transfer attributes from one image to another. At the feature level, this representation can be used for image retrieval and classification. To achieve this representation and to enable the applications at both the image and feature level, we leverage autoencoders. Here, an encoder transforms the input image $\mathbf{x}$ to its feature representation $\mathbf{f} = \text{Enc}(\mathbf{x})$, where $\mathbf{f} = [f^1, f^2, \ldots, f^n]$ consists of multiple chunks $f^i \in \mathbf{R}^d$.

The dimension of the full feature is therefore $n \times d$. In addition, a decoder transforms the feature representation back to the image via $\text{Dec}(\mathbf{f}) = \mathbf{x}$.

Our main objective is to learn a disentangled representation, where each feature chunk corresponds to an image attribute. For example, when the data $\mathbf{x}$ are face images, chunk f^1 could represent the hair color, f^2 the gender and so on. With a disentangled representation, we can transfer attributes from one image to another simply by swapping the feature chunks. An image $\mathbf{x}_3 = \text{Dec}([f_1^1, f_2^2, f_2^3, \ldots, f_2^n])$ could take the hair color from image $\mathbf{x}_1$ and all the other attributes from $\mathbf{x}_2$.

In our approach, we interpret disentanglement as invariance. In a disentangled representation, the encoding of each image attribute into its feature chunk should be invariant to transformations of any other image property. Vice versa, the decoding of each chunk into its corresponding attribute should be invariant to changes of other chunks. In our example, if $\mathbf{x}_1$ and $\mathbf{x}_2$ have the same gender, we must have $f_1^2 = f_2^2$ irrespective of any other attribute. Hence, a disentangled representation is also useful for image retrieval, where we can search for nearest neighbors of a specified attribute. Invariance is also beneficial for classification, where a simple linear classifier is sufficient to classify each attribute based on its corresponding feature chunk. This observation inspired previous work [18] to quantify disentanglement performance using linear classifiers on the full features $\mathbf{f}$.

In the following, we describe how we learn a disentangled representation from data without any additional knowledge (*e.g.*, labels, data domain) by using mixing autoencoders. One of the main challenges in the design of the autoencoder and its training is that the encoder and the decoder could just make use of a single feature chunk (provided that this is sufficient to represent the whole input image) and ignore the other chunks. We call this failure mode a *shortcut* taken by the autoencoder during training. We propose a novel invariance objective to obtain disentanglement, and a classification objective to avoid the shortcut problem.

3.1. Network Architecture

Our network architecture is shown in Figure 1. There are three main components: We enforce invariance using a *sequence of two mixing autoencoders*, and a *discriminator*; we avoid the shortcut problem using a *classifier*. They are all implemented as neural networks.

Mixing/Unmixing Autoencoders. We leverage a sequence of two mixing autoencoders to enforce invariance, ensuring that we encode each attribute into a feature chunk invariant to changes in other attributes, and that we decode each chunk similarly in an invariant manner into its attribute. More precisely, the sequence of two mixing autoencoders

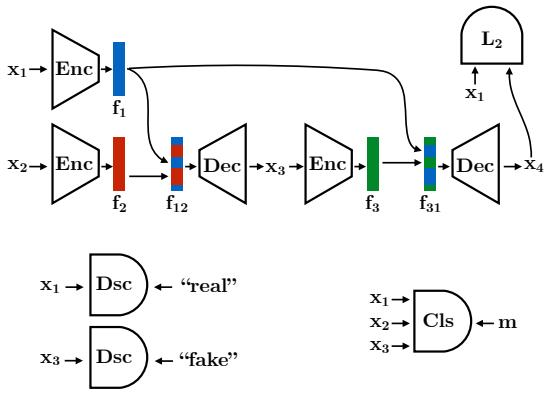


Figure 1: Overview of our architecture. The core component is a sequence of two mixing autoencoders (top). This implements our invariance objective, which encourages that the decoding of each feature chunk into an image attribute is invariant to a perturbation (mixing) in other chunks, and similarly, the encoding of each attribute into a chunk is invariant to a perturbation of other attributes. We include an adversarial loss to ensure the intermediate images obtained by perturbing some chunks is from our data distribution (bottom left). Finally, a classification objective avoids the shortcut problem, where chunks would be ignored completely. Components with the same name share weights.

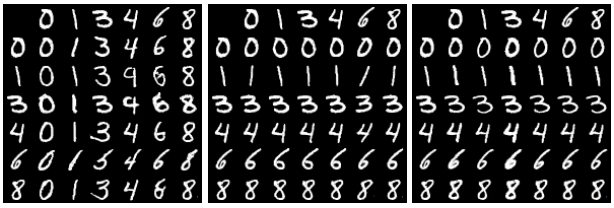
performs the following operations (Figure 1):

1. Sample two images $\mathbf{x}_1$ and $\mathbf{x}_2$ independently, and encode them into $\mathbf{f}_1 = \text{Enc}(\mathbf{x}_1)$ and $\mathbf{f}_2 = \text{Enc}(\mathbf{x}_2)$.
2. **Mix**: Define a mask $\mathbf{m} = [m^1\mathbf{1}, m^2\mathbf{1}, \ldots, m^n\mathbf{1}]$, where m^i are uniformly sampled in $\{0, 1\}$, and $\mathbf{1} = [1, 1, \ldots, 1] \in \mathbf{R}^d$. Select the i-th feature chunk from $\mathbf{f}_1$ if $m^i = 1$ and from $\mathbf{f}_2$ if $m^i = 0$; collect them into a new feature $\mathbf{f}_{1\oplus 2} = \mathbf{m} \odot \mathbf{f}_1 + (\mathbb{1} - \mathbf{m}) \odot f_2$, where $\odot$ is the element-wise multiplication and $\mathbb{1} = [\mathbf{1}, \mathbf{1}, \ldots, \mathbf{1}]$.
3. Decode a new image $\mathbf{x}_3 = \text{Dec}(\mathbf{f}_{1\oplus 2})$.
4. Encode again, $\mathbf{f}_3 = \text{Enc}(\mathbf{x}_3)$.
5. **Unmix** $\mathbf{f}_3$ by replacing feature chunks from $\mathbf{f}_2$, given by the mask $\mathbb{1} - \mathbf{m}$, with the corresponding ones from $\mathbf{f}_1$, that is, $\mathbf{f}_{3\oplus 1} = \mathbf{m} \odot \mathbf{f}_3 + (\mathbb{1} - \mathbf{m}) \odot \mathbf{f}_1$.
6. Decode the final image $\mathbf{x}_4 = \text{Dec}(\mathbf{f}_{3\oplus 1})$, from the mixed features of $\mathbf{f}_3$ and $\mathbf{f}_1$.

Finally, we minimize the squared L^2 distance between $\mathbf{x}_1$ and $\mathbf{x}_4$, thus the loss function can be written as

$$\mathcal{L}_M(\theta_{\text{Enc}}, \theta_{\text{Dec}}) = E_{\mathbf{x}_1,\mathbf{x}_2}\Big[\textstyle\sum_{\mathbf{m}} |\mathbf{x}_4 - \mathbf{x}_1|^2\Big], \qquad (1)$$

where we sum over all possible mask settings, and θ_{Enc} and θ_{Dec} are the encoder and decoder parameters respectively.



(a) Digit class (b) Rotation angle (c) Stroke width

Figure 2: Attribute transfer on the MNIST dataset by mixing individual chunks between pairs of source images, shown in the topmost row and leftmost column. To generate an image in column i and row j, we take one chunk from the i-th image in the top row, and the other chunks from the j-th image in the leftmost column. In each subfigure, the mixed chunk corresponds to the attribute indicated in the caption of the subfigure.

Intuitively, the key idea is that the cycle of decoding and re-encoding of the mixed feature vector $\mathbf{f}_{1\oplus 2}$ should preserve the chunks from $\mathbf{f}_1$ that were copied in $\mathbf{f}_{1\oplus 2}$. In other words, these chunks from $\mathbf{f}_1$ should be decoded into corresponding attributes of $\mathbf{x}_3$. In addition, re-encoding into $\mathbf{f}_3$ the intermediate image $\mathbf{x}_3$ consisting of a mix of attributes from $\mathbf{x}_1$ and attributes from $\mathbf{x}_2$, should return the same feature chunks originally from $\mathbf{x}_1$.

Discriminator. To ensure that the generated perturbed images $\mathbf{x}_3$ are valid images according to the input data distribution, we impose an additional adversarial term, which is defined as

$$\mathcal{L}_G(\theta_{\text{Enc}}, \theta_{\text{Dec}}, \theta_{\text{Dsc}}) = \sum_{\mathbf{m}} E_{\mathbf{x}_1,\mathbf{x}_2}\Big[\log(\text{Dsc}(\mathbf{x}_1)) + \log(1 - \text{Dsc}(\mathbf{x}_3))\Big], \tag{2}$$

where θ_{Dsc} are the discriminator parameters. In the ideal case when the GAN objective reaches the global optimum, the distribution of fake images should match the real image distribution. With the invariance and adversarial loss, however, is still possible to encode all image attributes into one feature chunk and keep the rest constant. This solution optimizes both the invariance loss and the adversarial loss perfectly. As mentioned before, this is called the shortcut problem and we address it using an additional loss based on a classification task.

Classifier. The last component of our network takes three images as inputs: the input images $\mathbf{x}_1$ and $\mathbf{x}_2$, and the generated image $\mathbf{x}_3$. It decides for every chunk whether the composite image was generated using the feature from the first or the second input image. The formal loss function is

$$\mathcal{L}_C(\theta_{\text{Enc}}, \theta_{\text{Dec}}, \theta_{\text{Cls}}) = E_{\mathbf{x}_1,\mathbf{x}_2}\Big[-\sum_{\mathbf{m}}\sum_i m^i \log(y^i) + (1 - m^i)\log(y^i))\Big], \tag{3}$$

Table 1: Network architectures of encoder (**Enc**), decoder (**Dec**), discriminator (**Dsc**) and classifier (**Cls**) on different datasets. We denote the convolutional layer with "c", the deconvolutional layer with "d" and the fully connected layer with "f". The numbers denote the number of channels. The kernel size and stride are denoted with "k" and "s", and they are omitted when they are equal to 1. The pooling layers "p" have kernel size 3 and stride 2. After each convolutional and deconvolutional layer we added a normalization and a leaky ReLU layer with a leak coefficient of 0.2. For BEGAN, the discriminator architecture is the same as that of the autoencoder. We used ReLU after the convolutional layers, and "r" stands for reshape and "u" for upsampling by a factor of 2. We choose $\gamma = 0.5$ for training.

	CelebA (DCGAN)
Enc	c64k3s2-c128k3s2-c256k3s2-c512k3s2-c512k2-f
Dec	d512k4-d512k4s2-d256k4s2-d128k4s2-d3k2
Dsc	c64k3s2-c128k3s2-c256k3s2-c512-f
Cls	c96k8s2-p-c256k5-p-c384k3-c384k3-c256k3-p-f4096-f4096-f
	CelebA (BEGAN)
Enc	c32k3-c32k3-c32k3-c64-p-c64k3-c64k3-c96-p-c96k3-c96k3-c128-p-c128k3-c128k3-c160-c160-p-c160k3-c160k3-f
Dec	f4096-r(8,8,64)-c64k3-c64k3-u-c64k3-c64k3-u-c64k3-c64k3-u-c64k3-c64k3-u-c64k3-c64k3-c3
Cls	c96k8s2-p-c256k5-p-c384k3-c384k3-c256k3-p-f4096-f4096-f
	MNIST
Enc	c64k3s2-c128k3s2-c256k3s2-f
Dec	d512k4-d256k4s2-d128k4s2-d3k2
Dsc	c64k3s2-c128k3s2-c256k3s2-c512-f
Cls	c96k8s2-p-c256k5-p-c384k3-c384k3-c256k3-p-f4096-f4096-f
	Sprites
Enc	c64k3s2-c128k3s2-c256k3s2-c512k2s2-c512k2-f
Dec	d512k4-d512k4s2-d256k4s2-d128k4s2-d3k2
Dsc	c64k3s2-c128k3s2-c256k3s2-c512-f
Cls	c96k8s2-p-c256k5-p-c384k3-c384k3-c256k3-p-f4096-f4096-f

where θ_{Cls} are the classifier parameters, and its outputs are $\text{Cls}(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3) = \mathbf{y} = [y^1, y^2, \ldots, y^n]$. The classifier consists of n binary classifiers, one for each chunk, that decide whether the composite image $\mathbf{x}_3$ was generated using the corresponding chunk from the first image or the second. We use the cross entropy loss for classification, so the last layer of the classifier is a sigmoid. The classifier loss can only be minimized if there is a meaningful attribute encoded in every chunk. Hence, the shortcut problem cannot occur as it would be impossible to decide which chunks were used to create the composite image.

Finally, our overall objective consists of the weighted sum of the three components described above,

$$\min_{\theta_{\text{Enc}},\theta_{\text{Dec}},\theta_{\text{Cls}}} \max_{\theta_{\text{Dsc}}} \quad \lambda_M \mathcal{L}_M + \lambda_G \mathcal{L}_G + \lambda_C \mathcal{L}_C. \tag{4}$$

Note that during training, we randomly sample the masks $\mathbf{m}$ instead of computing a sum over all possibilities for all image sample pairs (Eqns. (1), (2) and (3)).

(a) MIX (b) MIX+G (c) MIX+C (d) MIX+G+C

Figure 3: Comparison of different methods on Sprites. In all subfigures, images are generated by taking one of the 8 feature chunks from the topmost row, and the others from the leftmost column. Red frames indicate whether a feature chunk encodes an attribute. MIX denotes the mixing loss, G the adversarial loss and C the classifier loss in the objective. MIX+G+C disentangles pose, torso color, hair color, and leg color (columns marked with red boxes, left to right).

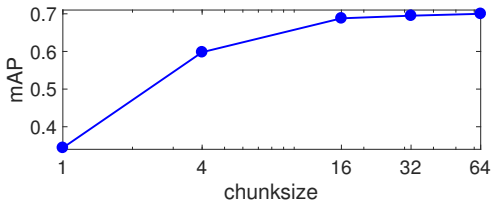


Figure 4: Mean average precision of the nearest neighbor search averaged across labeled attributes, as a function of the chunk size on the Sprites dataset with the complete model: the mixing autoencoder + the classifier + GAN.

3.2. Implementation

We use a network architecture similar to DCGAN [24] for the encoder, decoder, and discriminator. For the classifier, we use AlexNet with batch normalization after each convolutional layer, but we do not use any dropout. The image inputs of the classifier are concatenated along the RGB channels. We use equal weights $\lambda_M = \lambda_G = \lambda_C = 1$ for the mixing autoencoder, GAN, and classifier for our experiments on the MNIST and Sprites datasets. For CelebA, we increase the weight of the mixing autoencoder to $\lambda_M = 30$. In all experiments, the feature vector is the output of the last layer of the encoder. We separate it into 8 chunks, where each chunk is expected to represent one attribute, with equal size for each of the eight chunks. Our results are obtained with chunk size 8 for MNIST, 64 for Sprites and 64 for CelebA. We observed that reducing the chunk size in CelebA leads to lower rendering quality. For CelebA, we also show experiments using BEGAN [2] for the adversarial training. The detailed architectures are shown in Table 1.

4. Experiments

We experimented on three public datasets, the MNIST handwritten digits [19], Sprites animated figures [25], and CelebA faces [20]. We show qualitative results on all datasets and quantitative evaluations and ablation studies on Sprites and CelebA.

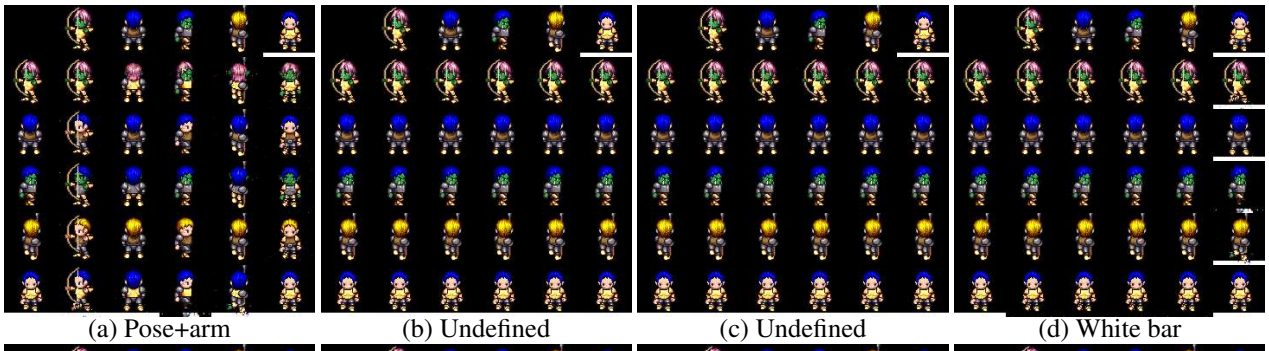
MNIST. The MNIST dataset consists of 60K handwritten digits for the training and 10K for the test set, given as grayscale images with a size of 28×28 pixels. There are 10 different classes referring to the different digits. Other attributes like rotation angle or stroke width are not labeled. Our method can disentangle the labeled attribute as well as some non-labeled ones. Figure 2 shows visual attribute transfers for three factors: digit class, rotation angle, and stroke width. The three chunks were chosen by visually inspecting which chunk corresponded to which attribute. All discernible variations seem to be encoded in the three chunks, and transferring the other chunks seem to have little visual effect.

Sprites. The sprites dataset has 672 synthetically rendered animated characters (sprites). The dataset is split into a training set with 500, a validation set with 72, and a test set with 100 sprites. Each sprite is rendered at 178 positions, thus the number of images is 120K in total. The dataset has many labeled attributes: body shape, skin color, vest color, hairstyle, arm and leg color, and finally weapon type. The pose labels can be extracted from the frame number of the animations. This rich attribute labeling is ideal for testing the disentanglement of our algorithms.

We perform ablation studies on the components of our method. The qualitative results are shown in Figure 3. We can see that mixing autoencoder already learned to disentangle 2 chunks. Adding only GAN does not improve the disentangling, as its job is to make the images look more realistic. However, the rendering quality without GAN is already good. Adding only the classifier does not improve disentangling either, it rather creates artifacts in the rendering. The intuitive explanation is that the classifier solves the shortcut problem in the sense that it forces all chunks to carry information about the inputs. However, the informa-

Table 2: Mean average precision performance of nearest neighbor classification on the Sprites dataset, which comes with labeled attributes. Each row contains different methods, while the columns show the classification performance of different attributes. MIX denotes the mixing loss, G the adversarial loss, C the classifier loss and AE is the vanilla autoencoder in the objective.

Method	body	skin	vest	hair	arm	leg	pose	average
Random	0.5	0.25	0.33	0.17	0.5	0.5	0.006	0.32
C+G	0.53	0.31	0.41	0.24	0.51	0.52	0.06	0.37
AE	0.56	0.37	0.40	0.31	0.54	0.56	0.46	0.46
AE+C+G	0.59	0.50	0.53	0.46	0.56	0.54	0.44	0.52
MIX	0.57	0.61	0.51	0.62	0.54	0.94	**0.53**	0.62
MIX + C	0.57	0.65	0.43	**0.63**	0.55	0.58	0.51	0.56
MIX + G	**0.59**	0.31	0.44	0.24	0.54	0.96	0.47	0.51
MIX + C + G	0.58	**0.80**	**0.94**	0.49	**0.58**	**0.96**	0.52	**0.70**



(a) Pose+arm (b) Undefined (c) Undefined (d) White bar

(e) Vest (f) Skin+hair (g) Leg (h) Undefined

Figure 5: Attribute transfer on the Sprites dataset. For every subfigure (a) to (h), one of the eight chunks is taken from the topmost row and the rest from the leftmost column. Each subfigure visualizes the role of one of the eight chunks, and the subfigure captions indicate the attribute (if semantically meaningful) associated with the chunk.

tion seem to be stored as artifacts, while the interpretable attributes are ignored. The full objective with all three components on the other hand improves the performance, as the artifacts are eliminated by GAN, and the shortcut problem can only be avoided by disentangling the factors. The method recovers 4 independent factors.

For quantitative analysis we perform nearest neighbor search using a chunk of the features and compute the mean average precision using an attribute as ground truth. We repeat the search for all chunk and attribute pairs, and for each attribute we choose the best performing chunk to represent it. We ignore the weapon type attribute in our evaluation, as it is only visible in a small subset of poses. We also compare our method to the vanilla autoencoder. It has only one chunk, but its dimensionality is the same as the full feature of the other methods. In Table 2 we compare the results of our methods. We can see a consistent improvement of our proposed mixing autoencoder over the vanilla autoencoder, whether we use the classifier and the GAN or not. The classifier and the GAN together also consistently help, no matter which autoencoder was used (MIX, vanilla or none). The classifier or the GAN alone do not help the performance, which is in line with the qualitative experiments as well. Figure 4 shows the effect of the chunk size on the classification performance. Increasing the number of dimensions helps, but we reach a plateau at 16 dimensions. We chose a large chunk size 64 for our experiments to better highlight that we can avoid the shortcut problem, the degenerate solu-

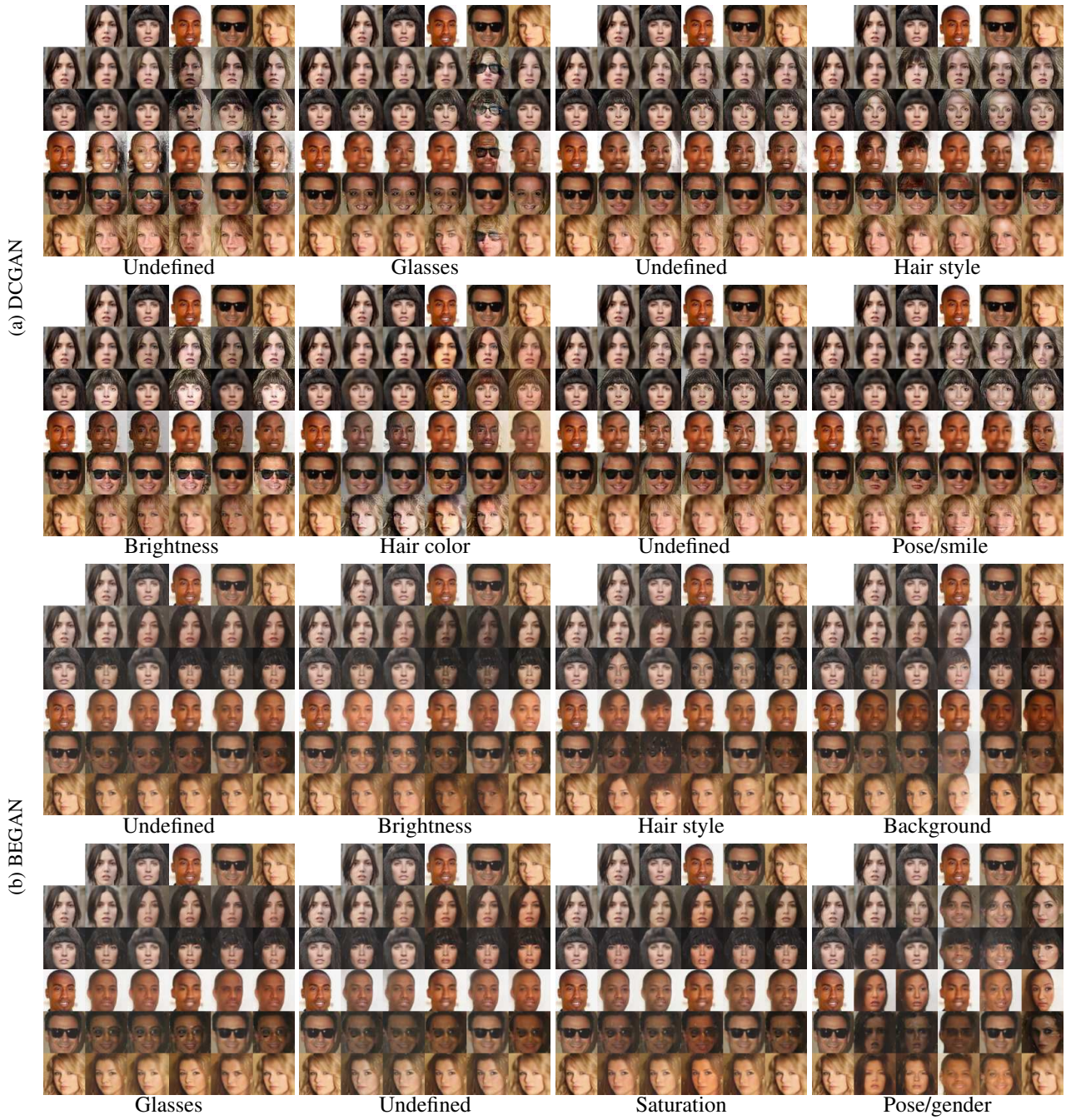


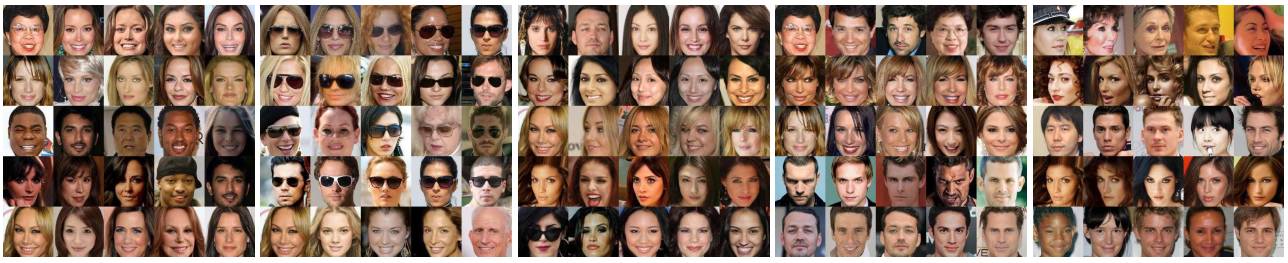
Figure 6: Attribute transfer on the CelebA dataset with our method using (a) DCGAN and (b) BEGAN. For every subfigure, one chunk is taken from the topmost row and the rest from the leftmost column. Different subfigures show the role of different chunks. The captions indicate the attribute associated with the chunk.

tion where all information is stored in one chunk. Figure 5 visualizes attribute transfer for all chunks, similarly to Figure 2, using our complete method. We can recover the leg and vest colors into single chunks, while the pose and arm color attribute pair is represented by one chunk. The skin color and hairstyle attributes are also entangled and represented by another chunk. There are 6 positions, where the sprites stand on a white bar. Even though this attribute is fully determined by the position, our method separates it to its own chunk.

CelebA. CelebA contains 200K color images of celebrity faces. The training, validation, and test sizes are 160K, 20K and 20K respectively. There are 40 labeled binary attributes indicating gender, hair color, facial hair and so

Table 3: The classification performance on CelebA. Each row contains different methods, while the columns show the different attributes ("eyebr." is arched eyebrows and "attr." is attractive).

Method	Eyebr.	Attr.	Bangs	Black	Blond	Makeup	Male	Mouth	Beard	Wavy	Hat	Lips	Avg.
VAE	71.8	73.0	89.8	78.0	88.9	79.6	83.9	76.3	87.3	70.2	95.8	83.0	81.5
β=2	71.6	72.6	90.6	79.3	89.1	79.3	83.5	76.1	86.9	67.8	95.9	82.4	81.3
β=4	71.6	72.6	90.0	76.6	88.9	77.8	82.3	75.7	85.3	66.8	95.8	80.6	80.3
β=8	71.6	71.7	90.0	76.0	87.2	76.2	80.5	73.1	85.3	63.7	95.8	79.6	79.2
DIP-VAE	**73.7**	**73.2**	**90.9**	**80.6**	**91.9**	**81.5**	**85.9**	75.9	85.3	**71.5**	96.2	**84.7**	**82.6**
Ours (DCGAN)	72.2	68.5	88.8	75.7	89.9	76.9	80.1	73.6	83.8	70.5	95.8	78.6	79.5
Ours (BEGAN)	73	69.7	90.2	79.6	89.3	78.9	85.4	**77.1**	**88.1**	70.8	**96.4**	81.7	81.7



(a) Brightness (b) Glasses (c) Hair color (d) Hair style (e) Pose/smile

Figure 7: Image retrieval on CelebA of our method with DCGAN. Subfigures show the nearest neighbor matches for different feature chunks. For all subfigures, the first column contains the query images and subsequent columns contain the top matches using the L^2 distance. The caption indicates the discovered semantic meaning.

on. We applied our method with both BEGAN and DCGAN architectures. Figure 6 shows the attribute transfer for each chunk. We can see that DCGAN exhibits more pronounced attribute transfer, while BEGAN tends to blur out the changes. Figure 7 shows the nearest neighbors of some query images in the dataset using DCGAN. We used the L^2 distance on the specified feature chunks to search for top matches. For each chunk the top matches preserve a semantic attribute of the query image. Our method could recover five semantically meaningful attributes: brightness, glasses, hair color, hair style, and pose and smile. Notice that the attributes discovered with attribute transfer match the attributes in image retrieval. For brevity we only show those five chunks. We performed quantitative tests on our learned features. We followed the evaluation technique based on the equivariant disentanglement property described in [18]. A feature representation is considered disentangled when the attributes can be classified using a simple linear classifier. In our special case when an attribute depends only on one chunk (a subspace), a linear classifier would perform well by setting the classifier weights with respect to the other chunks to zero. We train binary classifiers on the whole feature vector, each with a different labeled attribute as ground truth. The classifier prediction is $\text{sign}(\mathbf{w}^T\mathbf{f} + b)$, where the classifier weights are computed as

$$\mathbf{w} = \frac{1}{|i:c_i=+1|}\sum_{i:c_i=+1}\mathbf{f}_i - \frac{1}{|i:c_i=-1|}\sum_{i:c_i=-1}\mathbf{f}_i, \quad (5)$$

where $c_i \in \{-1, +1\}$ are the attribute labels. The bias term b is set by minimizing the hinge loss. For a fair comparison we normalize the features by setting the variance for each coordinate to one, as in [18] the features are already normalized by the variational autoencoder. The results are shown in Table 3. We can see that our network is competitive with the state of the art methods, β-VAE [10] and concurrent work DIP-VAE [18]. The BEGAN architecture performs slightly better than DCGAN, despite the superior rendering quality of the latter.

5. Conclusions

We have introduced a novel method to disentangle factors of variation of a single set of images where no annotation is available. Our representation is computed through an autoencoder, which is trained by imposing constraints between the encoded features and the rendered images. We train the decoder to render realistic images by feeding features obtained by randomly mixing features from two images and by using adversarial training. Moreover, we force the autoencoder to make full use of the features by training it jointly with a classifier that determines how features have been mixed from an input image. We show that this technique successfully disentangles factors of variation in the MNIST, Sprites and CelebA datasets.

Acknowledgements. QH, TP and AS have been supported by the Swiss National Science Foundation (SNSF) grants 200021_149227 and 200021_156253.

References

[1] Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013. 2

[2] D. Berthelot, T. Schumm, and L. Metz. BEGAN: Boundary equilibrium generative adversarial networks. *arXiv:1703.10717*, 2017. 5

[3] S. A. Bigdeli, M. Jin, P. Favaro, and M. Zwicker. Deep mean-shift priors for image restoration. In *Advances in Neural Information Processing Systems*, pages 763–772, 2017. 2

[4] H. Bourlard and Y. Kamp. Auto-association by multilayer perceptrons and singular value decomposition. *Biological cybernetics*, 59(4):291–294, 1988. 2

[5] Q. Chen and V. Koltun. Photographic image synthesis with cascaded refinement networks. In *ICCV*, 2017. 1

[6] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2172–2180, 2016. 1, 2

[7] E. L. Denton et al. Unsupervised learning of disentangled representations from video. In *Advances in Neural Information Processing Systems*, pages 4417–4426, 2017. 2

[8] J. Donahue, P. Krähenbühl, and T. Darrell. Adversarial feature learning. *International Conference on Learning Representations*, 2017. 2

[9] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014. 2

[10] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. In *ICLR*, 2016. 2, 8

[11] G. E. Hinton, A. Krizhevsky, and S. D. Wang. Transforming auto-encoders. In *International Conference on Artificial Neural Networks*, pages 44–51. Springer, 2011. 2

[12] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006. 2

[13] P. Isola, J. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. In *CVPR*, 2016. 2

[14] T. Karras, T. Aila, S. Laine, and J. Lehtinen. Progressive growing of GANs for improved quality, stability, and variation. In *International Conference on Learning Representations*, 2018. 1

[15] T. Kim, M. Cha, H. Kim, J. K. Lee, and J. Kim. Learning to discover cross-domain relations with generative adversarial networks. In *International Conference on Machine Learning*, pages 1857–1865, 2017. 2

[16] D. P. Kingma and M. Welling. Auto-encoding variational bayes. In *ICLR*, 2014. 2

[17] T. D. Kulkarni, W. F. Whitney, P. Kohli, and J. Tenenbaum. Deep convolutional inverse graphics network. In *Advances in Neural Information Processing Systems*, pages 2539–2547, 2015. 2

[18] A. Kumar, P. Sattigeri, and A. Balakrishnan. Variational Inference of Disentangled Latent Concepts from Unlabeled Observations. In *International Conference on Learning Representations*, 2018. 2, 3, 8

[19] 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. 5

[20] Z. Liu, P. Luo, X. Wang, and X. Tang. Deep learning face attributes in the wild. In *ICCV*, 2015. 5

[21] X. Mao, C. Shen, and Y.-B. Yang. Image restoration using very deep convolutional encoder-decoder networks with symmetric skip connections. In *Advances in neural information processing systems*, pages 2802–2810, 2016. 2

[22] M. F. Mathieu, J. J. Zhao, J. Zhao, A. Ramesh, P. Sprechmann, and Y. LeCun. Disentangling factors of variation in deep representation using adversarial training. In *Advances in Neural Information Processing Systems*, pages 5041–5049, 2016. 1, 2

[23] D. Pathak, P. Krähenbühl, J. Donahue, T. Darrell, and A. Efros. Context encoders: Feature learning by inpainting. In *CVPR*, 2016. 2

[24] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv:1511.06434*, 2015. 2, 5

[25] S. E. Reed, Y. Zhang, Y. Zhang, and H. Lee. Deep visual analogy-making. In *Advances in Neural Information Processing Systems*, pages 1252–1260, 2015. 1, 2, 5

[26] A. Shrivastava, T. Pfister, O. Tuzel, J. Susskind, W. Wang, and R. Webb. Learning from simulated and unsupervised images through adversarial training. In *CVPR*, 2017. 2

[27] Z. Shu, E. Yumer, S. Hadap, K. Sunkavalli, E. Shechtman, and D. Samaras. Neural face editing with intrinsic image disentangling. In *CVPR*, 2017. 2

[28] A. Szabó, Q. Hu, T. Portenier, M. Zwicker, and P. Favaro. Challenges in disentangling independent factors of variation. *arXiv:1711.02245*, 2017. 1, 2

[29] L. Tran, X. Yin, and X. Liu. Disentangled representation learning gan for pose-invariant face recognition. In *CVPR*, 2017. 2

[30] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol. Extracting and composing robust features with denoising autoencoders. In *International Conference on Machine Learning*. ACM, 2008. 2

[31] P. Xi, Y. Xiang, S. Kihyuk, M. Dimitris, and C. Manmohan. Reconstruction for feature disentanglement in pose-invariant face recognition. In *ICCV*, 2017. 2

[32] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe. Unsupervised learning of depth and ego-motion from video. In *CVPR*, 2017. 2

[33] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *ICCV*, 2017. 2