Understanding Degeneracies and Ambiguities in Attribute Transfer

⋆Attila Szabó[1][0000−0003−3808−5874], ⋆Qiyang Hu[1][0000−0002−9392−9112],
Tiziano Portenier[1][0000−0003−1766−1705], Matthias Zwicker[2][0000−0001−8630−5515],
and Paolo Favaro[1][0000−0003−3546−8247]

[1] University of Bern, Switzerland
{szabo, hu, portenier, favaro}@inf.unibe.ch
[2] University of Maryland, USA
zwicker@cs.umd.edu

Abstract. We study the problem of building models that can transfer selected attributes from one image to another without affecting the other attributes. Towards this goal, we develop analysis and a training methodology for autoencoding models, whose encoded features aim to disentangle attributes. These features are explicitly split into two components: one that should represent attributes in common between pairs of images, and another that should represent attributes that change between pairs of images. We show that achieving this objective faces two main challenges: One is that the model may learn degenerate mappings, which we call shortcut problem, and the other is that the attribute representation for an image is not guaranteed to follow the same interpretation on another image, which we call reference ambiguity. To address the shortcut problem, we introduce novel constraints on image pairs and triplets and show their effectiveness both analytically and experimentally. In the case of the reference ambiguity, we formally prove that a model that guarantees an ideal feature separation cannot be built. We validate our findings on several datasets and show that, surprisingly, trained neural networks often do not exhibit the reference ambiguity.

1 Introduction

One way to simplify the problem of classifying or regressing attributes of interest from data is to build an intermediate representation, a feature, where the information about the attributes is better separated than in the input data. Better separation means that some entries of the feature vary only with respect to one and only one attribute. In this way, classifiers and regressors would not need to build invariance to many nuisance attributes. Instead, they could devote more capacity to discriminating the attributes of interest, and possibly achieve better performance. We call this task *disentangling factors of variation*, and we speak interchangeably of attributes and factors. In addition to facilitating classification and regression, this task is beneficial to image synthesis. One could build a model

---

⋆ equal contribution

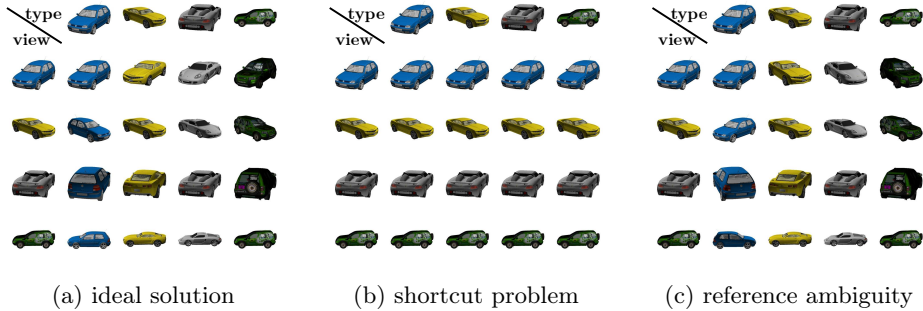|		
(a) ideal solution	(b) shortcut problem	(c) reference ambiguity

Fig. 1: Illustration of the challenges of attribute transfer. Consider a feature split into two parts, one representing the viewpoint, the other the car type. For all subfigures, the viewpoint feature is taken from the leftmost column and the car type feature is taken from the topmost row. (a) Ideal solution: the viewpoint and the car type are transferred correctly. (b) Shortcut problem: the car type is not transferred. The car type information from the image on the top row is ignored. (c) Reference ambiguity: the blue car has a different viewpoint orientation interpretation compared to the other car types.

to transfer attributes between images by rendering images where some elements of the input vary only one attribute of the output at a time.

When labeling is possible and available, supervised learning can be used to solve this task. In general, however, some attributes may not be easily quantifiable (*e.g.* style). Therefore, we consider using *weak labeling*, where we only know what attribute has changed between two images, although we do not know by how much. This type of labeling may be readily available in many cases without manual annotation. For example, objects in image pairs from a stereo system are automatically labeled with an unknown viewpoint change. A practical model that can learn from these labels is an autoencoder (*i.e.*, an encoder-decoder pair) subject to a reconstruction constraint. In this model the weak labels can be used to define similarities between subsets of the feature obtained from two input images. However, training such a model faces two fundamental challenges: one is that it may learn degenerate encodings, which we call the *shortcut problem*, and the other is that attributes extracted from one image must be interpreted in the same way on another image (*e.g.*, the attribute about the viewpoint of a car may be mapped to different angles in different car models), which we call the *reference problem*. These challenges are illustrated in Fig. 1.

Our contributions can be summarized as follows: *1)* We introduce a novel adversarial training of autoencoders to solve the disentangling task when only weak labels are available. The discriminator network in the adversarial training takes image pairs as input. In contrast to [15], our discriminator is not conditioned on class labels, so the number of parameters in our model can be kept constant; *2)* We show analytically and experimentally that our training method fully addresses the *shortcut problem*, where all the information is encoded only in one part of the

feature (see Fig. 1b); *3)* We show analysis on the *reference ambiguity*, and prove that it is unavoidable in the disentangling task when only weak labels are used. In Fig. 1c images are characterized by two car attributes: the viewpoint and the type. In this case, the reference ambiguity means that the viewpoint extracted from one image can have a different meaning than that of a different car type. Surprisingly, this ambiguity seems to occur rarely, typically only when the data dependence on the attribute of interest is complex.

2   Related work

In this paper we use autoencoders as the main model to build features and to synthesize new data. Therefore, we briefly review methods related to autoencoders. Since we train our model with an adversarial scheme, we also give a brief overview of some of the recent developments in this area. Finally, we discuss prior work on disentangling factors of variation that closely relates to our aims.

Autoencoders. Autoencoders [2,9,1] learn to reconstruct the input data as $\mathbf{x} = \text{Dec}(\text{Enc}(\mathbf{x}))$, where $\text{Enc}(\mathbf{x})$ is the internal image representation (the encoder) and Dec (the decoder) reconstructs the input of the encoder. Variational autoencoders [10] use instead a generative model $p(\mathbf{x}, \mathbf{z}) = p(\mathbf{x}|\mathbf{z})p(\mathbf{z})$, where $\mathbf{x}$ is the observed data (images), and $\mathbf{z}$ are latent variables. The encoder estimates the parameters of the posterior, $\text{Enc}(\mathbf{x}) = p(\mathbf{z}|\mathbf{x})$, and the decoder estimates the conditional likelihood, $\text{Dec}(\mathbf{z}) = p(\mathbf{x}|\mathbf{z})$. Transforming autoencoders [8] are trained with transformed image input pairs. The relative transformation parameters are also fed to the network. Because the internal representation explicitly represents the objects presence and location, the network can learn their absolute position. One important aspect of the autoencoders, which we exploit, is that they encourage latent representations to keep as much information about the input as possible.

GAN. Generative Adversarial Nets [7] learn to sample realistic images with two competing neural networks. The generator Dec creates images $\mathbf{x} = \text{Dec}(\mathbf{z})$ from a random noise sample $\mathbf{z}$ and tries to fool a discriminator Dsc, which has to decide whether the image is sampled from the generator p_g or from real images p_{real}. After a successful training the discriminator cannot distinguish real from generated samples. Adversarial training is often used to enforce (implicit) constraints on random variables as we do. For instance, BIGAN [6] learns a feature representation with adversarial nets by training an encoder Enc, such that $\text{Enc}(\mathbf{x})$ is Gaussian, when $\mathbf{x} \sim p_{real}$. CoGAN [13] learns the joint distribution of multi-domain images by having generators and discriminators in each domain, and sharing their weights. They can transform images between domains without being given correspondences. InfoGan [4] learns a subset of factors of variation by reproducing parts of the input vector with the discriminator.

Disentangling factors of variation. Many recent methods use neural networks for disentangling factors of variation. A lot of them are fully supervised [16,22,11,18,19], *i.e.*, they use labels for all factors they aim to disentangle. For example, Peng *et al.* [16] disentangle the face identities and poses using multiple source of labels including identity, pose and landmarks. With identity and pose

labels Tran *et al.* [22] can learn pose invariant features and synthesize frontalized faces from any pose. In deep visual analogy making [19] the supervisory signal is an image. The feature representation is split into two parts to represent different factors. The combination of these parts from different inputs are fed to the decoder, which has to reconstruct the target image. We also use the same feature swapping technique as in [19], but we do not need the ground truth target image for our training. Semi-supervised methods use labels of only part of the data samples. Siddharth *et al.* [21] propose a hybrid generative model to combine structured graphical models and unstructured random variables, thus enabling semi-supervised disentanglement. Our main focus is weakly supervised learning, where not all attributes come with labels. Shu *et al.* [20] disentangle intrinsic image factors (albedo and normal map) by modeling the physics of the image formation in their network. They use a 3D morphable model prior to guide the training. DrNet [5] disentangles the pose and content from videos. Assuming that the subsequent frames contain the same object, they can eliminate the content information from the pose, using an adversarial term on the features. Mathieu *et al.* [15] also use the feature swapping as in [19]. They use a GAN to avoid using the ground truth target images. In our work we do not use any prior information like in [20]. Compared to [5], our adversarial term allows for higher dimensional features, and unlike [15], we do not condition our GAN on class labels, thus we can keep the number of parameters constant. Moreover, with our adversarial term we can provably avoid the shortcut problem.

3   Disentangling attributes

We are interested in the design and training of two models. One should map a data sample (*e.g.*, an image) to a feature that is explicitly partitioned into subvectors, each associated with a specific attribute. The other model should map this feature back to an image. We call the first model the *encoder* and the second one the *decoder*. For example, given the image of a car as input we would like the encoder to output a feature with two subvectors: one related to the car viewpoint, and the other to the car type. This separation should simplify classification or regression of the attributes (the car viewpoint and type in the example). It should also be very useful for the advanced editing of images through the decoder. For example, the transfer of the viewpoint or car types from an image to another can be achieved by swapping the corresponding subvectors. Next, we introduce our model of the data and the definitions of our encoder and decoder (see Fig. 2).
Data model. We assume that observed data $\mathbf{x}$ is generated through an unknown deterministic invertible and smooth process f that depends on the factors $\mathbf{v}$ and $\mathbf{c}$, so that $\mathbf{x} = f(\mathbf{v}, \mathbf{c})$. In our earlier example, $\mathbf{x}$ is an image, $\mathbf{v}$ is a viewpoint, $\mathbf{c}$ is a car type, and f is the rendering engine. It is reasonable to assume that f is invertible, as for most cases the factors are readily apparent from the image. f is smooth, because we assume that a small change in the factors results in a small change in the image and vice versa. We denote the inverse of the rendering engine as $f^{-1} = [f_{\mathbf{v}}^{-1}, f_{\mathbf{c}}^{-1}]$, where the subscript refers to the recovered factor.
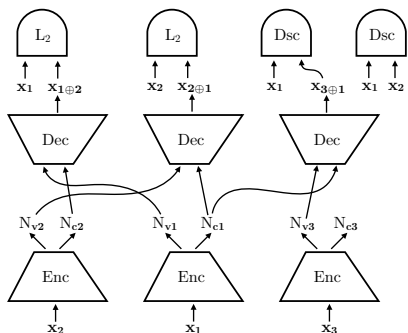
Fig. 2: Learning to disentangle factors of variation. The scheme above shows how the encoder (Enc), the decoder (Dec) and the discriminator (Dsc) are trained with input triplets. The components with the same name share weights.

Weak labeling. In the training we are given pairs of images $\mathbf{x}_1$ and $\mathbf{x}_2$, which differ in $\mathbf{v}$ (varying factor), but have the same $\mathbf{c}$ (common factor). We also assume that the two varying factors and the common factor are sampled independently, $\mathbf{v}_1 \sim p_\mathbf{v}$, $\mathbf{v}_2 \sim p_\mathbf{v}$ and $\mathbf{c} \sim p_\mathbf{c}$. The images are generated as $\mathbf{x}_1 = f(\mathbf{v}_1, \mathbf{c})$ and $\mathbf{x}_2 = f(\mathbf{v}_2, \mathbf{c})$. We call this labeling weak, because we do not know the absolute values of either the $\mathbf{v}$ or $\mathbf{c}$ factors or even relative changes between $\mathbf{v}_1$ and $\mathbf{v}_2$. All we know is that the image pairs share the same common factor $\mathbf{c}$.

The encoder. Let Enc be the encoder that maps images to features. For simplicity, we consider features split into only two column subvectors, $N_\mathbf{v}$ and $N_\mathbf{c}$, one associated to the varying factor $\mathbf{v}$ and the other associated to the common factor $\mathbf{c}$. Then, we have that $\text{Enc}(\mathbf{x}) = [N_\mathbf{v}(\mathbf{x}), N_\mathbf{c}(\mathbf{x})]$. Ideally, we would like to find the inverse of the image formation function, $[N_\mathbf{v}, N_\mathbf{c}] = f^{-1}$, which separates and recovers the factors $\mathbf{v}$ and $\mathbf{c}$ from data samples $\mathbf{x}$, *i.e.*,

$$N_\mathbf{v}(f(\mathbf{v}, \mathbf{c})) = \mathbf{v} \qquad N_\mathbf{c}(f(\mathbf{v}, \mathbf{c})) = \mathbf{c}. \tag{1}$$

In practice, these equations are not usable because all our constraints include the decoder, which can undo any bijective transformation of $\mathbf{v}$ and $\mathbf{c}$ and produce the same output $\mathbf{x}$. Therefore, we aim to find $N_\mathbf{v}$ and $N_\mathbf{c}$ that satisfy

$$R_\mathbf{v}(N_\mathbf{v}(f(\mathbf{v}, \mathbf{c}))) = \mathbf{v} \qquad R_\mathbf{c}(N_\mathbf{c}(f(\mathbf{v}, \mathbf{c}))) = \mathbf{c}, \tag{2}$$

which we call the *feature disentangling* properties, for all $\mathbf{v}$, $\mathbf{c}$, and some bijective functions $R_\mathbf{v}$ and $R_\mathbf{c}$, so that $N_\mathbf{v}$ is invariant to $\mathbf{c}$ and $N_\mathbf{c}$ is invariant to $\mathbf{v}$.

The decoder. Let Dec be the decoder that maps features to images. The sequence encoder-decoder is constrained to form an *autoencoder*, so

$$\text{Dec}(N_\mathbf{v}(\mathbf{x}), N_\mathbf{c}(\mathbf{x})) = \mathbf{x}, \qquad \forall \mathbf{x}. \tag{3}$$

To use the decoder to synthesize images, where different factors are transferred from different images, we can define the combined image as

$$\mathbf{x}_{1\oplus 2} \triangleq \text{Dec}(N_\mathbf{v}(\mathbf{x}_1), N_\mathbf{c}(\mathbf{x}_2)). \tag{4}$$

The ideal decoder should satisfy for all $\mathbf{x}_1$ and $\mathbf{x}_2$ the *data disentangling* properties

$$f_{\mathbf{v}}^{-1}(\mathbf{x}_{1\oplus 2}) = f_{\mathbf{v}}^{-1}(\mathbf{x}_1) \qquad f_{\mathbf{c}}^{-1}(\mathbf{x}_{1\oplus 2}) = f_{\mathbf{c}}^{-1}(\mathbf{x}_2) \tag{5}$$

In the next section we describe our training method for disentangling. We introduce a novel adversarial term that does not need to be conditioned on the common factor, but rather uses only image pairs, so the number of model parameters are constant. Then, we address the two main challenges of disentangling, the *shortcut problem* and the *reference ambiguity*. We discuss which disentanglement properties can be provably achieved by our, or any other, method.

3.1   Model training

In our training procedure we use two terms in the objective function: an *autoencoder loss* and an *adversarial loss*. We describe these losses in functional form, however the components are implemented using neural networks. In all our terms we use the following sampling of independent factors

$$\mathbf{c}_1, \mathbf{c}_3 \sim p_{\mathbf{c}}, \quad \mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3 \sim p_{\mathbf{v}}. \tag{6}$$

The images are formed as $\mathbf{x}_1 = f(\mathbf{v}_1, \mathbf{c}_1)$, $\mathbf{x}_2 = f(\mathbf{v}_2, \mathbf{c}_1)$ and $\mathbf{x}_3 = f(\mathbf{v}_3, \mathbf{c}_3)$. The images $\mathbf{x}_1$ and $\mathbf{x}_2$ share the same common factor, and $\mathbf{x}_1$ and $\mathbf{x}_3$ are independent. In our objective functions, we use either pairs or triplets of the above images.

Autoencoder loss. In this term, we use images $\mathbf{x}_1$ and $\mathbf{x}_2$ with the same common factor $\mathbf{c}_1$. We feed both images to the encoder. Since both images share the same $\mathbf{c}_1$, we impose that the decoder should reconstruct $\mathbf{x}_1$ from the encoder subvectors $N_{\mathbf{v}}(\mathbf{x}_1)$ and $N_{\mathbf{c}}(\mathbf{x}_2)$. Similarly $\mathbf{x}_2$ is reconstructed from $N_{\mathbf{v}}(\mathbf{x}_2)$ and $N_{\mathbf{c}}(\mathbf{x}_1)$. The autoencoder loss is thus defined as

$$\mathcal{L}_{AE} \triangleq \mathrm{E}_{\mathbf{x}_1,\mathbf{x}_2}\Big[\big|\mathbf{x}_1 - \mathrm{Dec}(N_{\mathbf{v}}(\mathbf{x}_1), N_{\mathbf{c}}(\mathbf{x}_2))\big|^2 + \big|\mathbf{x}_2 - \mathrm{Dec}(N_{\mathbf{v}}(\mathbf{x}_2), N_{\mathbf{c}}(\mathbf{x}_1))\big|^2\Big]. \tag{7}$$

Adversarial loss. We introduce an adversarial training where the *generator* is our encoder-decoder pair and the *discriminator* Dsc is a neural network, which takes image pairs as input. The discriminator learns to distinguish between real image pairs $[\mathbf{x}_1, \mathbf{x}_2]$ and fake ones $[\mathbf{x}_1, \mathbf{x}_{3\oplus 1}]$, where $\mathbf{x}_{3\oplus 1} \triangleq \mathrm{Dec}(N_{\mathbf{v}}(\mathbf{x}_3), N_{\mathbf{c}}(\mathbf{x}_1))$. The generator learns to fool the discriminator, so that $\mathbf{x}_{3\oplus 1}$ looks like the random variable $\mathbf{x}_2$ (the common factor is $\mathbf{c}_1$ and the varying factor is independent of $\mathbf{v}_1$). The adversarial loss function is then defined as

$$\mathcal{L}_{GAN} \triangleq \mathrm{E}_{\mathbf{x}_1,\mathbf{x}_2}\Big[\log(\mathrm{Dsc}(\mathbf{x}_1,\mathbf{x}_2))\Big] + \mathrm{E}_{\mathbf{x}_1,\mathbf{x}_3}\Big[\log(1 - \mathrm{Dsc}(\mathbf{x}_1,\mathbf{x}_{3\oplus 1}))\Big]. \tag{8}$$

Composite loss. Finally, we optimize the weighted sum of the two losses $\mathcal{L} = \mathcal{L}_{AE} + \lambda\mathcal{L}_{GAN}$,

$$\min_{\mathrm{Dec},\mathrm{Enc}} \max_{\mathrm{Dsc}} \mathcal{L}_{AE}(\mathrm{Dec}, \mathrm{Enc}) + \lambda\mathcal{L}_{GAN}(\mathrm{Dec}, \mathrm{Enc}, \mathrm{Dsc}) \tag{9}$$

where λ regulates the relative importance of the two losses.

3.2 The shortcut problem

Ideally, at the global minimum of $\mathcal{L}_{AE}$, $N_{\mathbf{v}}$ relates only to the factor $\mathbf{v}$ and $N_{\mathbf{c}}$ only to $\mathbf{c}$. However, the encoder may map a complete description of its input into $N_{\mathbf{v}}$ and the decoder may completely ignore $N_{\mathbf{c}}$. We call this challenge the *shortcut problem*. When this occurs, the decoder is invariant to its second input, thus the data disentanglement property for $\mathbf{c}$ eq. (5) does not hold, and we have

$$\text{Dec}(N_{\mathbf{v}}(\mathbf{x}_3), N_{\mathbf{c}}(\mathbf{x}_1)) = \mathbf{x}_3. \tag{10}$$

The shortcut problem can be addressed by reducing the dimensionality of $N_{\mathbf{v}}$, so that the encoder cannot build a complete representation of all input images. This also forces the encoder and decoder to make use of $N_{\mathbf{c}}$ for the common factor. However, this strategy may not be convenient as it leads to a time consuming trial-and-error procedure to find the correct dimensionality, which is unknown. In the next proposition we show that a better way to address the shortcut problem is instead to use adversarial training through the losses (8) and (9).

Proposition 1. *Let $\mathbf{x}_1$, $\mathbf{x}_2$ and $\mathbf{x}_3$ be data samples satisfying (6), where the factors $\mathbf{c}_1, \mathbf{c}_3, \mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3$ are jointly independent, and let $\mathbf{x}_{3\oplus1} \triangleq Dec(N_{\mathbf{v}}(\mathbf{x}_3), N_{\mathbf{c}}(\mathbf{x}_1))$. When the global optimum of the composite loss (9) is reached, the $\mathbf{c}$ factor has been disentangled, i.e., $f_{\mathbf{c}}^{-1}(\mathbf{x}_{3\oplus1}) = \mathbf{c}_1$.*

Proof. At the global optimum of (9), the distributions of $[\mathbf{x}_1, \mathbf{x}_2]$ and $[\mathbf{x}_1, \mathbf{x}_{3\oplus1}]$ image pairs are identical. We compute statistics of the inverse of the common factor $f_{\mathbf{c}}^{-1}$ on the data. For the images $\mathbf{x}_1$ and $\mathbf{x}_2$ we obtain

$$E_{\mathbf{x}_1,\mathbf{x}_2}\left[|f_{\mathbf{c}}^{-1}(\mathbf{x}_1) - f_{\mathbf{c}}^{-1}(\mathbf{x}_2)|^2\right] = E_{\mathbf{c}_1}\left[|\mathbf{c}_1 - \mathbf{c}_1|^2\right] = 0 \tag{11}$$

by construction (of $\mathbf{x}_1$ and $\mathbf{x}_2$). For the images $\mathbf{x}_1$ and $\mathbf{x}_{3\oplus1}$ we obtain

$$E_{\mathbf{x}_1,\mathbf{x}_3}\left[|f_{\mathbf{c}}^{-1}(\mathbf{x}_1) - f_{\mathbf{c}}^{-1}(\mathbf{x}_{3\oplus1})|^2\right] = E_{\mathbf{v}_1,\mathbf{c}_1,\mathbf{v}_3,\mathbf{c}_3}\left[|\mathbf{c}_1 - \mathbf{c}_{3\oplus1}|^2\right] \geq 0, \tag{12}$$

where $\mathbf{c}_{3\oplus1} = f_{\mathbf{c}}^{-1}(\mathbf{x}_{3\oplus1})$. We achieve equality if and only if $\mathbf{c}_1 = \mathbf{c}_{3\oplus1}$ for all samples (in the support of $p_{\mathbf{c}}$). □

3.3 The reference ambiguity

When the varying attribute (*e.g.*, the viewpoint) is transferred from an image to another, the numerical value of the varying attribute is interpreted in a reference frame, and the reference frame can depend on the common attribute (car type). Let us consider a practical example, where $\mathbf{v} \sim \mathcal{U}[-\pi, \pi]$ is the (continuous) viewpoint (the azimuth angle) and $\mathbf{c} \sim \mathcal{B}(1/2)$ is the car type, where $\mathcal{U}$ denotes the uniform distribution and $\mathcal{B}(1/2)$ the Bernoulli distribution with probability $p_{\mathbf{c}}(\mathbf{c} = 0) = p_{\mathbf{c}}(\mathbf{c} = 1) = 1/2$ (*i.e.*, there are only 2 car types). We can define a function $T(\mathbf{v}, \mathbf{c}) = \mathbf{v}(2\mathbf{c}-1)$ so that the mapping of $\mathbf{v}$ is mirrored as we change the car type. By construction $T(\mathbf{v}, \mathbf{c}) \sim \mathcal{U}[-\pi, \pi]$ for any $\mathbf{c}$ and $T(\mathbf{v}, \mathbf{c}_1) \neq T(\mathbf{v}, \mathbf{c}_2)$

for $\mathbf{v} \neq 0$ and $\mathbf{c}_1 \neq \mathbf{c}_2$. The encoder $N_{\mathbf{v}}(f(\mathbf{v}, \mathbf{c})) = T(\mathbf{v}, \mathbf{c})$ is feasible and reverses the ordering of the azimuth of car 1 with respect to car 0. Each car has its own reference system, and thus it is not possible to transfer the viewpoint from one system to the other, as illustrated in Fig. 1c. Below we prove that it is possible to disentangle $\mathbf{c}$, but not $\mathbf{v}$, as the task itself gives rise to this ambiguity.

Let us consider the ideal case where we observe the space of all images. Given the weak labels, we also know what images $\mathbf{x}_1$ and $\mathbf{x}_2$ share the same $\mathbf{c}$ factor (*e.g.*, which images have the same car). This labeling is equivalent to defining the probability density function $p_{\mathbf{c}}$ and the joint $p_{\mathbf{x}_1, \mathbf{x}_2}$. In the following proposition, we show that the labeling allows a learning algorithm to satisfy the feature disentangling property (2) for $\mathbf{c}$, but in Proposition 3 we show that this is not true for $\mathbf{v}$ (the reference ambiguity holds). The key step is that weak labels allow one to impose stricter constraints on $N_{\mathbf{c}}$, than on $N_{\mathbf{v}}$.

Proposition 2. *Given weak labels, the data is sampled according to $[\mathbf{x}_1, \mathbf{x}_2] \sim p_{\mathbf{x}_1, \mathbf{x}_2}$. Then, the feature disentangling property (2), for $\mathbf{c}$, can be satisfied.*

Proof. For any $[\mathbf{x}_1, \mathbf{x}_2] \sim p_{\mathbf{x}_1, \mathbf{x}_2}$, one can impose $N_{\mathbf{c}}(\mathbf{x}_1) = N_{\mathbf{c}}(\mathbf{x}_2)$, which implies that $N_{\mathbf{c}}$ is invariant to $\mathbf{v}$. Thus, $\forall \mathbf{c}$ let us define $C(\mathbf{c}) \triangleq N_{\mathbf{c}}(\mathbf{x}_1)$ as a function that depends only on $\mathbf{c}$. One can impose $f_{\mathbf{c}}^{-1}(\mathbf{x}_{a \oplus b}) = f_{\mathbf{c}}^{-1}(\mathbf{x}_b)$ (see Proposition 1), then images with the same $\mathbf{v}$, but different $\mathbf{c}$ must also result in different features, $C(\mathbf{c}_a) = N_{\mathbf{c}}(f(\mathbf{v}, \mathbf{c}_a)) \neq N_{\mathbf{c}}(f(\mathbf{v}, \mathbf{c}_b)) = C(\mathbf{c}_b)$. Then, there exists a bijective function $R_{\mathbf{c}} = C^{-1}$ such that property (2) is satisfied for $\mathbf{c}$. $\square$

We now introduce a definition that we need to formalize the reference ambiguity.

Definition 1. *We say that a function g reproduces the data distribution, when it generates samples $[\mathbf{y}_1, \mathbf{y}_2]$, where $\mathbf{y}_1 = g(\mathbf{v}_1, \mathbf{c})$ and $\mathbf{y}_2 = g(\mathbf{v}_2, \mathbf{c})$, that have the same distribution as the data $[\mathbf{x}_1, \mathbf{x}_2]$. Formally, $[\mathbf{y}_1, \mathbf{y}_2] \sim p_{\mathbf{x}_1, \mathbf{x}_2}$, where the latent factors are independent, i.e., $\mathbf{v}_1 \sim p_{\mathbf{v}}$, $\mathbf{v}_2 \sim p_{\mathbf{v}}$ and $\mathbf{c} \sim p_{\mathbf{c}}$.*

The next proposition illustrates the second main result in this paper: The reference ambiguity of the varying factor $\mathbf{v}$ occurs when a decoder reproduces the data without satisfying the disentangling properties. This implies that we cannot provably disentangle all the factors of variation from weakly labeled data, even if we had access to all the data and knew the distributions $p_{\mathbf{v}}$ and $p_{\mathbf{c}}$.

Proposition 3. *Let $p_{\mathbf{v}}$ assign the same probability value to at least two different instances of $\mathbf{v}$. Then, there exists a decoder that reproduces the data distribution, but does not satisfy the disentangling properties for $\mathbf{v}$ in eq. (2) and eq. (5).*

Proof. Let us choose $N_{\mathbf{c}} \triangleq f_{\mathbf{c}}^{-1}$, the inverse of the rendering engine. Now we look at defining $N_{\mathbf{v}}$ and the decoder. Let us denote with $\mathbf{v}_a \neq \mathbf{v}_b$ two varying factors such that $p_{\mathbf{v}}(\mathbf{v}_a) = p_{\mathbf{v}}(\mathbf{v}_b)$. Then, let the encoder for $\mathbf{v}$ be defined as

$$N_{\mathbf{v}}(f(\mathbf{v}, \mathbf{c})) \triangleq \begin{cases} \mathbf{v} & \text{if } \mathbf{v} \neq \mathbf{v}_a, \mathbf{v}_b \text{ or } \mathbf{c} \in \mathcal{C} \\ \mathbf{v}_a & \text{if } \mathbf{v} = \mathbf{v}_b \text{ and } \mathbf{c} \notin \mathcal{C} \\ \mathbf{v}_b & \text{if } \mathbf{v} = \mathbf{v}_a \text{ and } \mathbf{c} \notin \mathcal{C} \end{cases} \tag{13}$$

and $\mathcal{C}$ is a subset of the domain of $\mathbf{c}$, where $\int_{\mathcal{C}} p_{\mathbf{c}}(\mathbf{c})d\mathbf{c} \notin \{0,1\}$. Therefore, $N_{\mathbf{v}}(f(\mathbf{v},\mathbf{c})) \sim p_{\mathbf{v}}$ and $N_{\mathbf{v}}(f(\mathbf{v},\mathbf{c}_1)) \neq N_{\mathbf{v}}(f(\mathbf{v},\mathbf{c}_2))$ for $\mathbf{v} \in \{\mathbf{v}_a, \mathbf{v}_b\}$, $\mathbf{c}_1 \in \mathcal{C}$, and $\mathbf{c}_2 \notin \mathcal{C}$. Finally, we define the decoder as

$$\text{Dec}(\mathbf{v},\mathbf{c}) \triangleq \begin{cases} f(\mathbf{v},\mathbf{c}) & \text{if } \mathbf{v} \neq \mathbf{v}_a, \mathbf{v}_b \text{ or } \mathbf{c} \in \mathcal{C} \\ f(\mathbf{v}_a,\mathbf{c}) & \text{if } \mathbf{v} = \mathbf{v}_b \text{ and } \mathbf{c} \notin \mathcal{C} \\ f(\mathbf{v}_b,\mathbf{c}) & \text{if } \mathbf{v} = \mathbf{v}_a \text{ and } \mathbf{c} \notin \mathcal{C}. \end{cases} \tag{14}$$

Notice that $N_{\mathbf{v}}(f(\mathbf{v},\mathbf{c}))$ depends on $\mathbf{c}$ functionally, but is statistically independent from it. In fact, because $p_{\mathbf{v}}(\mathbf{v}_a) = p_{\mathbf{v}}(\mathbf{v}_b)$ we have

$$\begin{aligned} p_{N_{\mathbf{v}},\mathbf{c}}(\mathbf{v},\mathbf{c}) &= p_{N_{\mathbf{v}}|\mathbf{c}}(\mathbf{v}|\mathbf{c})p_{\mathbf{c}}(\mathbf{c}) \tag{15} \\ &= [\mathbf{1}_{\mathcal{C}}(\mathbf{c})p_{\mathbf{v}}(\mathbf{v}) + \mathbf{1}_{\overline{\mathcal{C}}}(\mathbf{c})\left[\delta(\mathbf{v}-\mathbf{v}_a)p_{\mathbf{v}}(\mathbf{v}_b) + \delta(\mathbf{v}-\mathbf{v}_b)p_{\mathbf{v}}(\mathbf{v}_a)\right]] p_{\mathbf{c}}(\mathbf{c}) \\ &= [\mathbf{1}_{\mathcal{C}}(\mathbf{c})p_{\mathbf{v}}(\mathbf{v}) + \mathbf{1}_{\overline{\mathcal{C}}}(\mathbf{c})\left[\delta(\mathbf{v}-\mathbf{v}_a)p_{\mathbf{v}}(\mathbf{v}_a) + \delta(\mathbf{v}-\mathbf{v}_b)p_{\mathbf{v}}(\mathbf{v}_b)\right]] p_{\mathbf{c}}(\mathbf{c}) \\ &= p_{\mathbf{v}}(\mathbf{v})p_{\mathbf{c}}(\mathbf{c}). \end{aligned}$$

Thus, no statistical constraint on the encoded factors $N_{\mathbf{v}}, N_{\mathbf{c}}$ will allow distinguishing them from the original factors $\mathbf{v},\mathbf{c}$. Finally, we can substitute in $[\text{Dec}(N_{\mathbf{v}}(\mathbf{x}_1), N_{\mathbf{c}}(\mathbf{x}_1)), \text{Dec}(N_{\mathbf{v}}(\mathbf{x}_2), N_{\mathbf{c}}(\mathbf{x}_2))]$ and reproduce the data distribution, *i.e.*, $[\text{Dec}(\mathbf{v}_1,\mathbf{c}), \text{Dec}(\mathbf{v}_2,\mathbf{c})] \sim p_{\mathbf{x}_1,\mathbf{x}_2}$. The feature disentanglement property is not satisfied because $N_{\mathbf{v}}(f(\mathbf{v}_a,\mathbf{c}_1)) = \mathbf{v}_a \neq \mathbf{v}_b = N_{\mathbf{v}}(f(\mathbf{v}_a,\mathbf{c}_2))$, when $\mathbf{c}_1 \in \mathcal{C}$ and $\mathbf{c}_2 \notin \mathcal{C}$. Similarly, the data disentanglement property does not hold, because $f_{\mathbf{v}}^{-1}(\text{Dec}(N_{\mathbf{v}}(f(\mathbf{v}_a,\mathbf{c}_1)),\mathbf{c}_1)) \neq f_{\mathbf{v}}^{-1}(\text{Dec}(N_{\mathbf{v}}(f(\mathbf{v}_a,\mathbf{c}_1)),\mathbf{c}_2))$. $\square$

3.4   Implementation

In our implementation we use convolutional neural networks for all the models. We denote with θ the parameters associated to each network. Then, the optimization of the composite loss can be written as

$$\hat{\theta}_{\text{Dec}}, \hat{\theta}_{\text{Enc}}, \hat{\theta}_{\text{Dsc}} = \arg\min_{\theta_{\text{Dec}},\theta_{\text{Enc}}} \max_{\theta_{\text{Dsc}}} \mathcal{L}(\theta_{\text{Dec}}, \theta_{\text{Enc}}, \theta_{\text{Dsc}}). \tag{16}$$

We choose $\lambda = 1$ and also add regularization to the adversarial loss so that each logarithm has a minimum value. We define $\log_{\epsilon} \text{Dsc}(\mathbf{x}_1,\mathbf{x}_2) = \log(\epsilon + \text{Dsc}(\mathbf{x}_1,\mathbf{x}_2))$ (and similarly for the other logarithmic term) and use $\epsilon = 10^{-12}$. The main components of our neural network are shown in Fig. 2. The architecture of the encoder, decoder and the discriminator were taken from DCGAN [17], with slight modifications. We added fully connected layers at the output of the encoder and to the input of the decoder. As the input to the discriminator is an image pair, we concatenate them along the color channels. The details of the architecture is described in Table 1 for all datasets we experimented on.

Normalization.   In our architecture both the encoder and the decoder networks use blocks with a convolutional layer, a nonlinear activation function (ReLU/leaky ReLU) and a normalization layer, typically, batch normalization (BN). As an alternative to BN we consider *instance normalization* (IN) [23]. The main difference between BN and IN is that the latter just computes the mean

Table 1: Network architectures. In the encoder and discriminator we used convolutional layers with a kernel size 4 and stride 2. After each convolutional layer we added normalization and a leaky ReLU layer with a leak coefficient of 0.2. In the decoder we used deconvolutional layers with kernel size 4 and stride 2 followed by a ReLU. c stands for convolutional, d for deconvolutional and f for fully connected layers, and the numbers indicate the number of channels.

	ShapeNet, CelebA, CUB	MNIST	Sprites
Enc	c64-c128-c256-c512-c512-f	c64-c128-c256-f	c64-c128-c256-c512-f
Dec	f-d512-d512-d256-d128-d3	f-d512-d256-d128-d3	f-d512-d256-d128-d3
Dsc	c64-c128-c256-c512-f	c64-c128-c256-f	c64-c128-c256-c512-f

Table 2: Nearest neighbor classification on viewpoint and car type features using different normalization techniques on ShapeNet cars. The performance is measured in mean average precision.

Normalization	viewpoint	car type
None	0.47	0.13
Batch	0.50	0.08
Instance	0.50	0.20

and standard deviation across the spatial domain of the input and not along the batch dimension. Thus, the shift and scaling for the output of each layer is the same at every iteration for the same input image. We compared the different normalization choices for the ShapeNet dataset in Table 2, where we report the performance on the nearest neighbor classification task. The feature dimensions were fixed at 1024 for both $N_{\mathbf{v}}$ and $N_{\mathbf{c}}$ in all normalization cases. We can see that both batch and instance normalization perform equally well on viewpoint classification and no normalization is slightly worse. For the car type classification instance normalization is clearly better.

4   Experiments

We tested our method on the MNIST, Sprites, CelebA, CUB and ShapeNet datasets and performed ablation studies on the shortcut problem using ShapeNet cars. We focused on the effect of the feature dimensionality and having the adversarial term (*i.e.*, $\mathcal{L}_{AE} + \mathcal{L}_{GAN}$) or not (*i.e.*, only $\mathcal{L}_{AE}$). We also show that in most cases the reference ambiguity does not arise in practice (MNIST, Sprites, CelebA, CUB, ShapeNet cars and motorcycles), we can only observe it when the data is more complex (ShapeNet chairs and vessels).

The shortcut problem. The ShapeNet dataset [3] contains 3D objects than we can render from different viewpoints. We consider only one category (cars) and a set of fixed viewpoints. Cars have high intraclass variability and they do
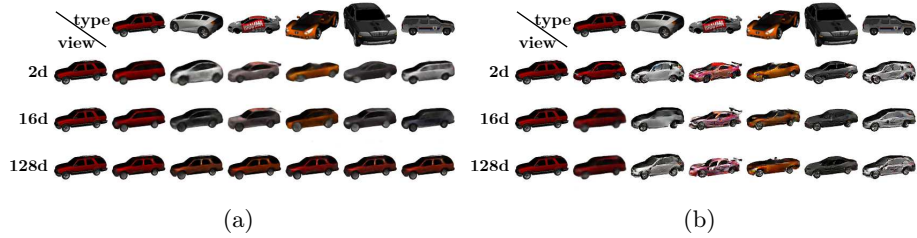
(a)   (b)

Fig. 3: Attribute transfer on ShapeNet. (a) Synthesized images with $\mathcal{L}_{AE}$, where the top row shows images from which the car type is taken. The second, third and fourth row show the decoder renderings using 2, 16 and 128 dimensions for the feature $N_{\mathbf{v}}$. (b) Images synthesized with $\mathcal{L}_{AE} + \mathcal{L}_{GAN}$. The setting for the inputs and feature dimensions are the same as in (a).

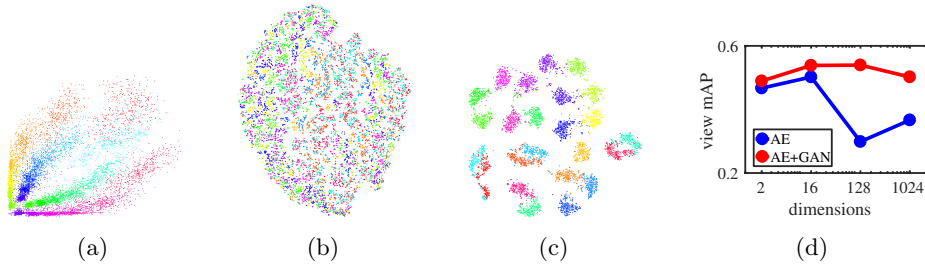(a)   (b)   (c)   (d)

Fig. 4: The effect of dimensions and objective function on $N_{\mathbf{v}}$ features. (a), (b), (c) t-SNE embeddings on $N_{\mathbf{v}}$ features. Colors correspond to the ground truth viewpoint. The objective functions and the $N_{\mathbf{v}}$ dimensions are: (a) $\mathcal{L}_{AE}$ 2 dim, (b) $\mathcal{L}_{AE}$ 128 dim, (c) $\mathcal{L}_{AE} + \mathcal{L}_{GAN}$ 128 dim. (d) Mean average precision curves for the viewpoint prediction from the viewpoint feature using different models and dimensions for $N_{\mathbf{v}}$.

not have rotational symmetries. We used approximately 3K car types for training and 300 for testing. We rendered 24 possible viewpoints around each object in a full circle, resulting in 80K images in total. The elevation was fixed to 15 degrees and azimuth angles were spaced 15 degrees apart. We normalized the size of the objects to fit in a 100×100 pixel bounding box, and placed it in the middle of a 128×128 pixel image. Fig. 3 shows the attribute transfer on the ShapeNet cars. We compare the methods $\mathcal{L}_{AE}$ and $\mathcal{L}_{AE} + \mathcal{L}_{GAN}$ with different feature dimension of $N_{\mathbf{v}}$. The size of the common feature $N_{\mathbf{c}}$ was fixed to 1024 dimensions. We can observe that the transferring performance degrades for $\mathcal{L}_{AE}$, when we increase the feature size of $N_{\mathbf{v}}$. This illustrates the shortcut problem, where the autoencoder tries to store all the information into $N_{\mathbf{v}}$. The model $\mathcal{L}_{AE} + \mathcal{L}_{GAN}$ instead renders images without loss of quality, independently of the feature dimension. In Fig. 4 we visualize the t-SNE embeddings of the $N_{\mathbf{v}}$ features for several models using different feature sizes. For the $2D$ case, we do not

(a) ShapeNet cars

(b) ShapeNet chairs

(c) ShapeNet vessels
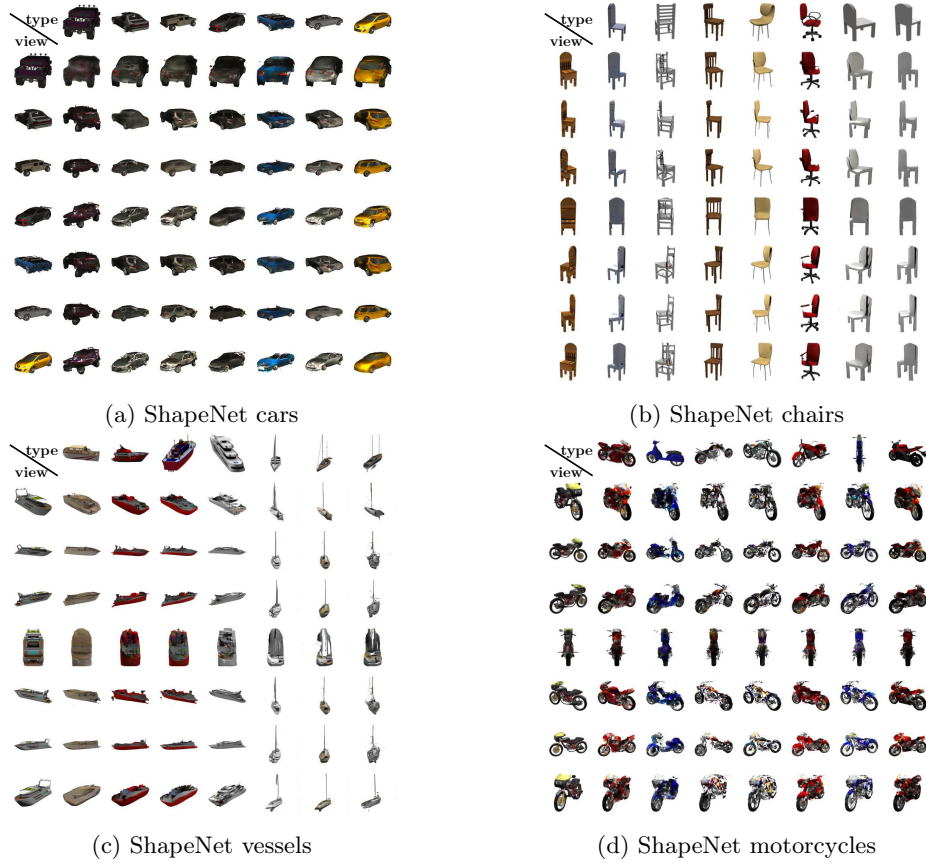

(d) ShapeNet motorcycles

Fig. 5: Attribute transfer on ShapeNet categories. For all subfigures the object type is taken from the topmost row and the viewpoint is taken from the leftmost column.

modify the data. We can see that both $\mathcal{L}_{AE}$ with 2 dimensions and $\mathcal{L}_{AE} + \mathcal{L}_{GAN}$ with 128 dimensions separate the viewpoints well, but the lone $\mathcal{L}_{AE}$ with 128 dimensions does not separate the viewpoints well due to the shortcut problem. We investigate the effect of dimensionality of the $N_\mathbf{v}$ features on the nearest neighbor classification task. The performance is measured by the mean average precision. For $N_\mathbf{v}$ we use the viewpoint as ground truth. Fig. 4 also shows the results on $\mathcal{L}_{AE}$ and $\mathcal{L}_{AE} + \mathcal{L}_{GAN}$ models with different $N_\mathbf{v}$ feature dimensions. The dimension of $N_\mathbf{c}$ was fixed to 1024 for this experiment. One can now see quantitatively that $\mathcal{L}_{AE}$ is sensitive to the size of $N_\mathbf{v}$, while $\mathcal{L}_{AE} + \mathcal{L}_{GAN}$ is not. $\mathcal{L}_{AE} + \mathcal{L}_{GAN}$ also achieves a better performance.

The reference ambiguity. We rendered the ShapeNet chairs, vessels and motorcycles with the same settings (viewpoints, image size) as the cars. There are 3500 chair, 1500 vessel and 300 motorcycle types for training, which provide
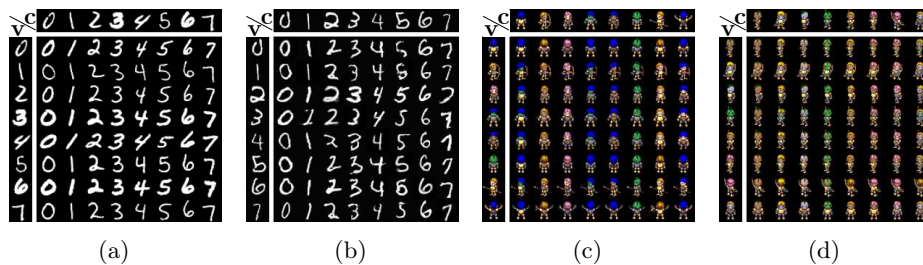
Fig. 6: Renderings of transferred features. In all figures the varying factor is transferred from the left column and the common factor from the top row. (a) MNIST [15]; (b) MNIST (ours); (c) Sprites [15]; (d) Sprites (ours).

between 7K and 84K images for each category. We trained our network with the full objective $\mathcal{L}_{AE} + \mathcal{L}_{GAN}$ and with the same settings as in the case of ShapeNet cars. In Fig. 5 we show the attribute transfer results. We can see that the object type is transferred correctly in all cases, and we can observe the reference ambiguity in two out of four categories. Some of the rendered chairs are flipped. The most interesting case is the vessel category, where the viewpoint angle is interpreted in two different ways depending on the boat type. We may be inclined to conclude that objects with a similar shape tend to share the same reference for the varying attribute (in the case of vessels, large boats seem to transfer the viewpoint between each other, but not with thinner boats).

MNIST evaluation. The MNIST dataset [12] contains handwritten grayscale digits of size 28×28 pixel. There are 60K images of 10 classes for training and 10K for testing. The common factor is the digit class and the varying factor is the intraclass variation. We take image pairs that have the same digit for training, and use our full model $\mathcal{L}_{AE} + \mathcal{L}_{GAN}$ with dimensions 64 for $N_\mathbf{v}$ and 64 for $N_\mathbf{c}$. In Fig. 6 (a) and (b) we show the transfer of varying factors. Qualitatively, both our method and [15] perform well. We observe neither the reference ambiguity nor the shortcut problem in this case, probably due to the high similarity of objects within the same category.

Sprites evaluation. The Sprites dataset [19] contains 60 pixel color images of animated characters (sprites). There are 672 sprites, 500 for training, 100 for testing and 72 for validation. Each sprite has 20 animations and 178 images, so the full dataset has 120K images in total. There are many changes in the appearance of the sprites, they differ in their body shape, gender, hair, armour, arm type, greaves, and weapon. We consider character identity as the common factor and the pose as the varying factor. We train our system using image pairs of the same sprite and do not exploit labels on their pose. We train the $\mathcal{L}_{AE} + \mathcal{L}_{GAN}$ model with dimensions 64 for $N_\mathbf{v}$ and 448 for $N_\mathbf{c}$. Fig. 6 (c) and (d) show results on the attribute transfer task. Both our method and [15] transfer the identity of the sprites correctly, the reference ambiguity does not arise.

CUB evaluation. The CUB birds dataset [24] contains 12K images of 200 bird species. In our model we choose the bird species as the common factor and used
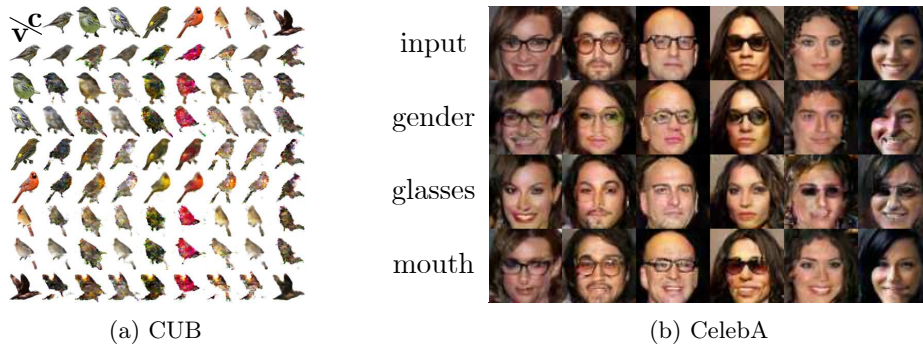
| (a) CUB | (b) CelebA |

Fig. 7: Attribute transfer on CUB and CelebA datasets. (a) CUB birds, where the pose is taken from the leftmost column and the species is taken from the topmost row. (b) CelebA, the first row shows the original image and each subsequent row shows the change of the following attributes: gender, glasses, mouth-open.

the same settings as for ShapeNet. The results on attribute transfer can be seen on Fig. 7a.

CelebA evaluation. The CelebA dataset [14] contains 200K face images. It contains labelled binary attributes such as male/female, old/young and so on. We used the same settings as for ShapeNet, and trained separate models, where the common attribute was one of the labelled ones. The results on attribute transfer can be seen in Fig. 7b.

5   Conclusions

In this paper we studied two fundamental challenges of disentangling factors of variation: the shortcut problem and the reference ambiguity. The shortcut problem occurs when all information is stored in only one feature subvector, while the other is ignored. The reference ambiguity means that the reference in which a factor is interpreted, may depend on other factors. This makes the attribute transfer ambiguous. We introduced a novel training of autoencoders to solve disentangling using image triplets. We showed theoretically and experimentally how to avoid the shortcut problem through adversarial training. Moreover, our method allows using arbitrarily large feature dimensions, which simplifies the design of the autoencoder model. We proved that the reference ambiguity is inherently present in the disentangling task when weak labels are used. Most importantly this can be stated independently of the learning algorithm. We demonstrated that training and transfer of factors of variation may not be guaranteed. However, in practice we observe that some trained models work well on many datasets and exhibit good attribute transfer capabilities.

References

1. Bengio, Y., Courville, A., Vincent, P.: Representation learning: A review and new perspectives. IEEE transactions on pattern analysis and machine intelligence **35**(8), 1798–1828 (2013)
2. Bourlard, H., Kamp, Y.: Auto-association by multilayer perceptrons and singular value decomposition. Biological cybernetics **59**(4), 291–294 (1988)
3. Chang, A.X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., Xiao, J., Yi, L., Yu, F.: ShapeNet: An Information-Rich 3D Model Repository. arXiv:1512.03012 (2015)
4. Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., Abbeel, P.: Infogan: Interpretable representation learning by information maximizing generative adversarial nets. NIPS (2016)
5. Denton, E.L., Birodkar, V.: Unsupervised learning of disentangled representations from video. NIPS (2017)
6. Donahue, J., Krähenbühl, P., Darrell, T.: Adversarial feature learning. ICLR (2014)
7. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. NIPS (2014)
8. Hinton, G.E., Krizhevsky, A., Wang, S.D.: Transforming auto-encoders. International Conference on Artificial Neural Networks pp. 44–51 (2011)
9. Hinton, G.E., Salakhutdinov, R.R.: Reducing the dimensionality of data with neural networks. Science **313**(5786), 504–507 (2006)
10. Kingma, D.P., Welling, M.: Auto-encoding variational bayes. ICLR (2014)
11. Lample, G., Zeghidour, N., Usunier, N., Bordes, A., Denoyer, L., Ranzato, M.A.: Fader networks:manipulating images by sliding attributes. NIPS (2017)
12. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE **86**(11), 2278–2324 (1998)
13. Liu, M.Y., Tuzel, O.: Coupled generative adversarial networks. NIPS (2016)
14. Liu, Z., Luo, P., Wang, X., Tang, X.: Deep learning face attributes in the wild. ICCV (2015)
15. Mathieu, M.F., Zhao, J.J., Zhao, J., Ramesh, A., Sprechmann, P., LeCun, Y.: Disentangling factors of variation in deep representation using adversarial training. NIPS (2016)
16. Peng, X., Yu, X., Sohn, K., Metaxas, D.N., Chandraker, M.: Reconstruction-based disentanglement for pose-invariant face recognition. ICCV (2017)
17. Radford, A., Metz, L., Chintala, S.: Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv:1511.06434 (2015)
18. Reed, S., Sohn, K., Zhang, Y., Lee, H.: Learning to disentangle factors of variation with manifold interaction. ICML (2014)
19. Reed, S.E., Zhang, Y., Zhang, Y., Lee, H.: Deep visual analogy-making. NIPS (2015)
20. Shu, Z., Yumer, E., Hadap, S., Sunkavalli, K., Shechtman, E., Samaras, D.: Neural face editing with intrinsic image disentangling. CVPR (2017)
21. Siddharth, N., Paige, T.B., van de Meent, J.W., Desmaison, A., Goodman, N., Kohli, P., Wood, F., Torr, P.: Learning disentangled representations with semi-supervised deep generative models. NIPS (2017)
22. Tran, L., Yin, X., Liu, X.: Disentangled representation learning gan for pose-invariant face recognition. CVPR (2017)
23. Ulyanov, D., Vedaldi, A., Lempitsky, V.S.: Improved texture networks: Maximizing quality and diversity in feed-forward stylization and texture synthesis. CVPR (2017)
24. Wah, C., Branson, S., Welinder, P., Perona, P., Belongie, S.: The Caltech-UCSD Birds-200-2011 Dataset. Tech. rep. (2011)