CHALLENGES IN DISENTANGLING INDEPENDENT FACTORS OF VARIATION

Anonymous authors
Paper under double-blind review

ABSTRACT

We study the problem of building models that disentangle independent factors of variation. Such models could be used to encode features that can efficiently be used for classification and to transfer attributes between different images in image synthesis. As data we use a weakly labeled training set. Our weak labels indicate what single factor has changed between two data samples, although the relative value of the change is unknown. This labeling is of particular interest as it may be readily available without annotation costs. To make use of weak labels we introduce an autoencoder model and train it through constraints on image pairs and triplets. We formally prove that without additional knowledge there is no guarantee that two images with the same factor of variation will be mapped to the same feature. We call this issue the reference ambiguity. Moreover, we show the role of the feature dimensionality and adversarial training. We demonstrate experimentally that the proposed model can successfully transfer attributes on several datasets, but show also cases when the reference ambiguity occurs.

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 identify attributes with the factors. In addition to facilitating classification and regression, this task is beneficial to image synthesis. One could build a model to render images, where each input varies only one attribute of the output, and to transfer attributes between images.

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, image pairs from a stereo system are automatically labeled with a viewpoint change, albeit unknown. A practical model that can learn from these labels is 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.

In this paper we study the ambiguities in mapping images to factors of variation and the effect of the feature dimensionality on the learned representation. Moreover, we introduce a novel architecture and training procedure to disentangle factors of variation. We find that the simple reconstruction constraint may fail at disentangling the factors when the dimensionality of the features is too large. We thus introduce an adversarial training to address this problem. More importantly, in general there is no guarantee that a model will learn to disentangle all factors. We call this challenge the *reference ambiguity* and formally show the conditions under which it appears. In practice, however, we observe experimentally that often the reference ambiguity does not occur on several synthetic datasets.

2 RELATED WORK

Autoencoders. Autoencoders in Bourlard & Kamp (1988), Hinton & Salakhutdinov (2006), Bengio et al. (2013) 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 in Kingma & Welling (2014) use 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, $\text{Dec}(\mathbf{z}) = p(\mathbf{x}|\mathbf{z})$. In Hinton et al. (2011) autoencoders 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 is that they encourage latent representations to keep as much information about the input as possible.

GAN. Generative Adversarial Nets Goodfellow et al. (2014) 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 the real from the generated samples. Adversarial training is often used to enforce constraints on random variables. BIGAN, Donahue et al. (2016) 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, Liu & Tuzel (2016) 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, Chen et al. (2016) learns a subset of factors of variation by reproducing parts of the input vector with the discriminator.

Disentangling and independence. Many recent methods use neural networks for disentangling features, with various degrees of supervision. In Xi Peng (2017) multi-task learning is used with full supervision for pose invariant face recognition. Using both identity and pose labels Tran et al. (2017) can learn pose invariant features and synthesize frontalized faces from any pose. In Yang et al. (2015) autoencoders are used to generate novel viewpoints of objects. They disentangle the object category factor from the viewpoint factor by using as explicit supervision signals: the relative viewpoint transformations between image pairs. In Cheung et al. (2014) the output of the encoder is split in two parts: one represents the class label and the other represents the nuisance factors. Their objective function has a penalty term for misclassification and a cross-covariance cost to disentangle class from nuisance factors. Hierarchical Boltzmann Machines are used in Reed et al. (2014) for disentangling. A subset of hidden units are trained to be sensitive to a specific factor of variation, while being invariant to others. Variational Fair Autoencoders Louizos et al. (2016) learn a representation that is invariant to specific nuisance factors, while retaining as much information as possible. Autoencoders can also be used for visual analogy Reed et al. (2015). GAN is used for disentangling intrinsic image factors (albedo and normal map) in Shu et al. (2017) without using ground truth labelling. They achieve this by explicitly modeling the physics of the image formation in their network.

The work most related to ours is Mathieu et al. (2016), where an autoencoder restores an image from another by swapping parts of the internal image representation. Their main improvement over Reed et al. (2015) is the use of adversarial training, which allows for learning with image pairs instead of image triplets. Therefore, expensive labels like viewpoint alignment between different car types are no longer needed. One of the differences between this method and ours is that it trains a discriminator for each of the given labels. A benefit of this approach is the higher selectivity of the discriminator, but a drawback is that the number of model parameters grows linearly with the number of labels. In contrast, we work with image pairs and use a single discriminator so that our method is uninfluenced by the number of labels. Moreover, we show formally and experimentally the difficulties of disentangling factors of variation.

3 DISENTANGLING FACTORS OF VARIATION

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 to a specific factor of variation. The other model should map this feature back to an image. We call the first model the

encoder and the second model the *decoder*. For example, given the image of a car we would like the encoder to yield a feature with two subvectors: one related to the car viewpoint, and the other related to every other car attribute. The subvectors of the feature obtained from the encoder should be useful for classification or regression of the corresponding factor that they depend on (the car viewpoint in the example). This subvector separation would also be very useful to the decoder. In fact, given a valid feature, one could vary only one of its subvectors (for example, the one corresponding to the viewpoint) and observe a variation of the output of the decoder just about its associated factor (the viewpoint). Such decoder would enable advanced editing of images. For example, it would allow the transfer of the viewpoint or other car attributes from an image to another.

The main challenge in the design of these models, when trained on weakly labeled data, is that the factors of variation are latent and introduce ambiguities in the representation. We explain later that avoiding these ambiguities is not possible without using further prior knowledge about the data. We prove this fundamental issue formally, provide an example where it manifests itself and demonstrate it experimentally. Interestingly, as the experiments will show, whether the ambiguity emerges or not depends on the complexity of the data. Next, we introduce our model of the data and formal definitions of our encoder and decoder.

Data model. We assume that our observed data $\mathbf{x}$ is generated through some deterministic invertible and smooth process f that depends on the factors $\mathbf{v} \sim p_{\mathbf{v}}$ and $\mathbf{c} \sim p_{\mathbf{c}}$, so that $\mathbf{x} = f(\mathbf{v}, \mathbf{c})$. In our earlier example, $\mathbf{x}$ is an image, $\mathbf{v}$ is a viewpoint (the varying component), $\mathbf{c}$ is a car type (the common component), and f is the rendering engine. We assume that f is unknown, and $\mathbf{v}$ and $\mathbf{c}$ are independent.

The encoder. Let Enc be the encoder mapping 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 $\text{Enc}(\mathbf{x}) = [N_{\mathbf{v}}^\top(\mathbf{x})\ N_{\mathbf{c}}^\top(\mathbf{x})]^\top$. 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, this is not possible because any bijective transformation of $\mathbf{v}$ and $\mathbf{c}$ could be undone by f and produce the same output $\mathbf{x}$. Therefore, we aim for $N_{\mathbf{v}}$ and $N_{\mathbf{c}}$ that satisfy the following *feature disentangling* properties

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

for all $\mathbf{v}$, $\mathbf{c}$, and for 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 mapping features to images. A first constraint is that the sequence encoder-decoder forms an *autoencoder*, that is,

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

To use the decoder for image synthesis, so that each input subvector affects only one factor in the rendered image, the ideal decoder should satisfy the *data disentangling* property

$$\text{Dec}(N_{\mathbf{v}}(f(\mathbf{v}_1, \mathbf{c}_1)), N_{\mathbf{c}}(f(\mathbf{v}_2, \mathbf{c}_2))) = f(\mathbf{v}_1, \mathbf{c}_2) \tag{4}$$

for any $\mathbf{v}_1$, $\mathbf{v}_2$, $\mathbf{c}_1$, and $\mathbf{c}_2$. The equation above describes the transfer of the varying factor $\mathbf{v}_1$ of $\mathbf{x}_1$ and the common factor $\mathbf{c}_2$ of $\mathbf{x}_2$ to a new image $\mathbf{x}_3 = f(\mathbf{v}_1, \mathbf{c}_2)$.

In the next section, we show that there is an inherent ambiguity in recovering $\mathbf{v}$ from images and in transferring it from one image to another. We show that, given our weakly labeled data, it may not be possible to satisfy all the feature and data disentangling properties. We call this challenge the *reference ambiguity*.

3.1 The reference ambiguity

Let us consider the ideal case where we observe the space of all images. When weak labels are made available to us, we also know what images $\mathbf{x}_1$ and $\mathbf{x}_2$ share the same $\mathbf{c}$ factor (for example, which

images have the same car). This labeling is equivalent to defining the probability density function $p_{\mathbf{c}}$ and the joint conditional $p_{\mathbf{x}_1,\mathbf{x}_2|\mathbf{c}}$, where

$$p_{\mathbf{x}_1,\mathbf{x}_2|\mathbf{c}}(\mathbf{x}_1,\mathbf{x}_2|\mathbf{c}) = \int \delta(\mathbf{x}_1 - f(\mathbf{v}_1,\mathbf{c}))\delta(\mathbf{x}_2 - f(\mathbf{v}_2,\mathbf{c}))p(\mathbf{v}_1)p(\mathbf{v}_2)d\mathbf{v}_1 d\mathbf{v}_2. \tag{5}$$

Firstly, we show that the labeling allows us to satisfy the feature disentangling property for $\mathbf{c}$ (2). For any $[\mathbf{x}_1,\mathbf{x}_2] \sim p_{\mathbf{x}_1,\mathbf{x}_2|\mathbf{c}}$ we impose $N_{\mathbf{c}}(\mathbf{x}_1) = N_{\mathbf{c}}(\mathbf{x}_2)$. In particular, this equation is true for pairs when one of the two images is held fixed. Thus, $N_{\mathbf{c}}(\mathbf{x}_1) = C(\mathbf{c})$, where the function C only depends on $\mathbf{c}$, as $N_{\mathbf{c}}$ is invariant to $\mathbf{v}$. Lastly, images with the same varying factor, but different common factor must also result in different features, $C(\mathbf{c}_1) = N_{\mathbf{v}}(f(\mathbf{v},\mathbf{c}_1)) \neq N_{\mathbf{v}}(\mathbf{v},\mathbf{c}_2) = C(\mathbf{c}_2)$, otherwise the autoencoder constraint cannot be satisfied. Then, there exists a bijective function $R_{\mathbf{c}} = C^{-1}$ such that property (2) is satisfied for $\mathbf{c}$. Unfortunately, this is not true in general for the other disentangling properties.

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

The reference ambiguity occurs, when a decoder reproduces the data without satisfying the disentangling properties.

Proposition 1. *Let $p_{\mathbf{v}}$ assign the same probability value to at least two different instances of $\mathbf{v}$. Then, we can find encoders that reproduce the data distribution, but do not satisfy the disentangling properties for $\mathbf{v}$ in* (2) *and* (4)*.*

Proof. We already saw that $N_{\mathbf{c}}$ satisfies (2), so we can choose $N_{\mathbf{c}} = f_{\mathbf{c}}^{-1}$, the ideal encoding. Now we look at defining $N_{\mathbf{v}}$ and the decoder. The iso-probability property of $p_{\mathbf{v}}$ implies that there exists a mapping $T(\mathbf{v},\mathbf{c})$, such that $T(\mathbf{v},\mathbf{c}) \sim p_{\mathbf{v}}$ and $T(\mathbf{v},\mathbf{c}_1) \neq T(\mathbf{v},\mathbf{c}_2)$ for some $\mathbf{v}$ and $\mathbf{c}_1 \neq \mathbf{c}_2$. For example, let us denote with $\mathbf{v}_1 \neq \mathbf{v}_2$ two varying components such that $p_{\mathbf{v}}(\mathbf{v}_1) = p_{\mathbf{v}}(\mathbf{v}_2)$. Then, let

$$T(\mathbf{v},\mathbf{c}) \doteq \begin{cases} \mathbf{v} & \text{if } \mathbf{v} \neq \mathbf{v}_1, \mathbf{v}_2 \\ \mathbf{v}_1 & \text{if } \mathbf{v} = \mathbf{v}_1 \vee \mathbf{v}_2 \text{ and } \mathbf{c} \in \mathcal{C} \\ \mathbf{v}_2 & \text{if } \mathbf{v} = \mathbf{v}_1 \vee \mathbf{v}_2 \text{ and } \mathbf{c} \notin \mathcal{C} \end{cases} \tag{6}$$

and $\mathcal{C}$ is a subset of the domain of $\mathbf{c}$, where $\int_{\mathcal{C}} p_{\mathbf{c}}(\mathbf{c})d\mathbf{c} = 1/2$. Now, let us define the encoder as $N_{\mathbf{v}}(f(\mathbf{v},\mathbf{c})) = T(\mathbf{v},\mathbf{c})$. By using the autoencoder constraint, the decoder satisfies

$$\text{Dec}(N_{\mathbf{v}}(f(\mathbf{v},\mathbf{c})), N_{\mathbf{c}}(f(\mathbf{v},\mathbf{c}))) = \text{Dec}(T(\mathbf{v},\mathbf{c}),\mathbf{c}) = f(\mathbf{v},\mathbf{c}). \tag{7}$$

Because $T(\mathbf{v},\mathbf{c}) \sim p_{\mathbf{v}}$ and $\mathbf{c} \sim p_{\mathbf{c}}$ by construction, and $T(\mathbf{v},\mathbf{c})$ and $\mathbf{c}$ are independent, our encoder-decoder pair defines a data distribution identical to that given as training set

$$[\text{Dec}(T(\mathbf{v}_1,\mathbf{c}),\mathbf{c}), \text{Dec}(T(\mathbf{v}_2,\mathbf{c}),\mathbf{c})] \sim p_{\mathbf{x}_1,\mathbf{x}_2}. \tag{8}$$

The feature disentanglement property is not satisfied because $N_{\mathbf{v}}(f(\mathbf{v}_1,\mathbf{c}_1)) = T(\mathbf{v}_1,\mathbf{c}_1) \neq T(\mathbf{v}_1,\mathbf{c}_2) = N_{\mathbf{v}}(f(\mathbf{v}_1,\mathbf{c}_2))$, when $\mathbf{c}_1 \in \mathcal{C}$ and $\mathbf{c}_2 \notin \mathcal{C}$. Similarly, the data disentanglement property does not hold, because $\text{Dec}(T(\mathbf{v}_1,\mathbf{c}_1),\mathbf{c}_1) \neq \text{Dec}(T(\mathbf{v}_1,\mathbf{c}_2),\mathbf{c}_2)$. □

The above proposition implies that we cannot learn to 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 $p_{\mathbf{c}}$.

To better understand it, let us consider a practical example. Let $\mathbf{v} \sim \mathcal{U}[-\pi,\pi]$ be the (continuous) viewpoint (the azimuth angle) and $\mathbf{c} \sim \mathcal{B}(0.5)$ the car type, where $\mathcal{U}$ denotes the uniform distribution and $\mathcal{B}(0.5)$ the Bernoulli distribution with probability $p_{\mathbf{c}}(\mathbf{c}=0) = p_{\mathbf{c}}(\mathbf{c}=1) = 0.5$ (*i.e.*, there are only 2 car types). In this case, every instance of $\mathbf{v}$ is iso-probable in $p_{\mathbf{v}}$ so we have the worst scenario for the reference ambiguity. We can define the 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 definition $T(\mathbf{v},\mathbf{c}) \sim \mathcal{U}[-\pi,\pi]$ for any $\mathbf{c}$ and $T(\mathbf{v},\mathbf{c}_1) \neq T(\mathbf{v},\mathbf{c}_2)$ for $\mathbf{v} \neq 0$ and $\mathbf{c}_1 \neq \mathbf{c}_2$. So we cannot tell the difference between T and the ideal correct mapping to the viewpoint factor. This is equivalent to an encoder $N_{\mathbf{v}}(f(\mathbf{v},\mathbf{c})) = T(\mathbf{v},\mathbf{c})$ that 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.

We now introduce a training procedure to build the encoder and decoder from weakly labeled data. We will use these models to illustrate several challenges: 1) the reference ambiguity, 2) the choice of the feature dimensionality and 3) the normalization layers (see the Implementation section).

3.2 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

$$\mathbf{c}_1, \mathbf{c}_3 \sim p_{\mathbf{c}}, \quad \mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3 \sim p_{\mathbf{v}}. \tag{9}$$

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. 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.

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 subvector $N_{\mathbf{v}}(\mathbf{x}_1)$ and the encoder subvector $N_{\mathbf{c}}(\mathbf{x}_2)$, and similarly for the reconstruction of $\mathbf{x}_2$. The autoencoder objective is thus defined as

$$\mathcal{L}_{AE} \doteq E_{\mathbf{x}_1,\mathbf{x}_2}\Big[\big|\mathbf{x}_1 - \text{Dec}(N_{\mathbf{v}}(\mathbf{x}_1), N_{\mathbf{c}}(\mathbf{x}_2))\big|^2 + \big|\mathbf{x}_2 - \text{Dec}(N_{\mathbf{v}}(\mathbf{x}_2), N_{\mathbf{c}}(\mathbf{x}_1))\big|^2\Big]. \tag{10}$$

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 $\mathbf{x}_{3\oplus 1} \doteq \text{Dec}(N_{\mathbf{v}}(\mathbf{x}_3), N_{\mathbf{c}}(\mathbf{x}_1))]$. If the encoder were ideal, the image $\mathbf{x}_{3\oplus 1}$ would be the result of taking the common component from $\mathbf{x}_1$ and the viewpoint component from $\mathbf{x}_3$. The generator learns to fool the discriminator, so that $\mathbf{x}_{3\oplus 1}$ looks like the random variable $\mathbf{x}_2$ (the common component is $\mathbf{c}_1$ and the varying component is independent of $\mathbf{v}_1$). To this purpose, the decoder must make use of $N_{\mathbf{c}}(\mathbf{x}_1)$, since $\mathbf{x}_3$ does not carry any information about $\mathbf{c}_1$. The objective function is thus defined as

$$\mathcal{L}_{GAN} \doteq E_{\mathbf{x}_1,\mathbf{x}_2}\Big[\log(\text{Dsc}(\mathbf{x}_1, \mathbf{x}_2))\Big] + E_{\mathbf{x}_1,\mathbf{x}_3}\Big[\log(1 - \text{Dsc}(\mathbf{x}_1, \mathbf{x}_{3\oplus 1}))\Big]. \tag{11}$$

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

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

where λ regulates the relative importance of the two losses.

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 the shortcut problem occurs, the decoder is invariant to its second output, so it does not transfer the $\mathbf{c}$ factor correctly,

$$\text{Dec}(N_{\mathbf{v}}(\mathbf{x}_3), N_{\mathbf{c}}(\mathbf{x}_1)) = \mathbf{x}_3. \tag{12}$$

The shortcut problem can be addressed by reducing the dimensionality of $N_{\mathbf{v}}$, so that it cannot build a complete representation of all input images. This also forces the encoder 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. A better way to address the shortcut problem is to use adversarial training. For our analysis we assume that the discriminator is perfect and the global optimum of the adversarial training has been reached. Thus, the real and fake image pair distributions are identical, and any statistics of the two distributions should also match. We compute statistics of the inverse of the common component $f_{\mathbf{c}}^{-1}$. For the images $\mathbf{x}_1$ and $\mathbf{x}_2$ we obtain

$$E_{\mathbf{x}_1,\mathbf{x}_2}\Big[|f_{\mathbf{c}}^{-1}(\mathbf{x}_1) - f_{\mathbf{c}}^{-1}(\mathbf{x}_2)|^2\Big] = E_{\mathbf{c}_1}\Big[|\mathbf{c}_1 - \mathbf{c}_1|^2\Big] = 0 \tag{13}$$

by construction (of $\mathbf{x}_1$ and $\mathbf{x}_2$). For the images $\mathbf{x}_1$ and $\mathbf{x}_{3\oplus 1}$ we obtain

$$E_{\mathbf{x}_1,\mathbf{x}_3}\Big[|f_{\mathbf{c}}^{-1}(\mathbf{x}_1) - f_{\mathbf{c}}^{-1}(\mathbf{x}_{3\oplus 1})|^2\Big] = E_{\mathbf{v}_1,\mathbf{c}_1,\mathbf{v}_3,\mathbf{c}_3}\Big[|\mathbf{c}_1 - \mathbf{c}_{3\oplus 1}|^2\Big] \geq 0, \tag{14}$$

where $\mathbf{c}_{3\oplus 1} = f_{\mathbf{c}}^{-1}(\mathbf{x}_{3\oplus 1})$. We achieve equality if and only if $\mathbf{c}_1 = \mathbf{c}_{3\oplus 1}$ everywhere. This means that the decoder must use $N_{\mathbf{c}}$ to recover the common component $\mathbf{c}$ of its input and $N_{\mathbf{c}}$ is sufficient to recover it.

3.3 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

$$\hat{\theta}_{\text{Dec}}, \hat{\theta}_{\text{Enc}}, \hat{\theta}_{\text{Dsc}} = \arg \min_{\theta_{\text{Dec}}, \theta_{\text{Enc}}} \max_{\theta_{\text{Dsc}}} \mathcal{L}(\theta_{\text{Dec}}, \theta_{\text{Enc}}, \theta_{\text{Dsc}}). \quad (15)$$

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. 1. The architecture of the encoder and the decoder were taken from DCGAN Radford et al. (2015), with slight modifications. We added fully connected layers at the output of the encoder and to the input of the decoder. For the discriminator we used a simplified version of the VGG Simonyan & Zisserman (2014) network. As the input to the discriminator is an image pair, we concatenate them along the color channels.

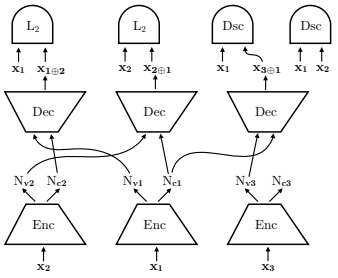


Figure 1: Learning to disentangle factors of variation. The scheme above shows how the encoder (Enc), the decoder (Dec) and the discriminator (Dsc) are trained with input triplets. The components with the same name share weights.

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 the recently introduced *instance normalization* (IN) Ulyanov et al. (2017). The main difference between BN and IN is that the latter just computes the mean 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. In practice, we find that IN improves the performance.

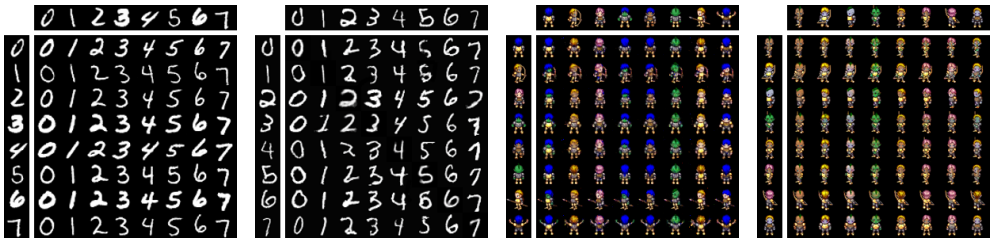
4 EXPERIMENTS

We tested our method on the MNIST, Sprites and ShapeNet datasets. We performed qualitative experiments on attribute transfer, and quantitative tests on the nearest neighbor classification task. We show results with models using only the autoencoder loss (**AE**) and the composite loss (**AE+GAN**).

MNIST. The MNIST dataset LeCun et al. (1998) 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 **AE+GAN** with dimensions 64 for $N_{\mathbf{v}}$ and 64 for $N_{\mathbf{c}}$. In Fig. 2 (a) and (b) we show the transfer of varying factors. Qualitatively, both our method and Mathieu et al. (2016) perform well. We observe neither the reference ambiguity nor the shortcut problem in this case.

Sprites. The Sprites dataset Reed et al. (2015) 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, armor, 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 **AE+GAN** model with dimensions 64 for $N_{\mathbf{v}}$ and 448 for $N_{\mathbf{c}}$. Fig. 2 (c) and (d) show results on the attribute transfer task. Both our method and Mathieu et al. (2016)'s transfer the identity of the sprites correctly.

ShapeNet with a white background. The ShapeNet dataset Chang et al. (2015) contains 3D objects than we can render from different viewpoints. We consider only one category (cars) for a set of fixed viewpoints. Cars have high intraclass variability and they do 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. We trained both **AE** and **AE+GAN** on ShapeNet, and tried different settings for the feature dimensions $N_{\mathbf{v}}$. The size of the common feature $N_{\mathbf{c}}$ was fixed to 1024 dimensions. Fig. 3 shows the attribute transfer on



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

Figure 2: Renderings of transferred features. In all figures the variable factor is transferred from the left column and the common factor from the top row. (a) MNIST Mathieu et al. (2016); (b) MNIST (ours); (c) Sprites Mathieu et al. (2016); (d) Sprites (ours).

(a) (b)

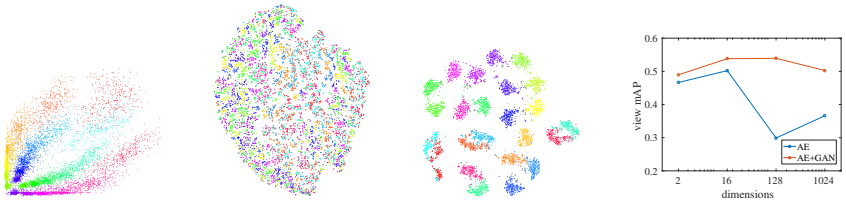
Figure 3: Feature transfer on Shapenet. (a) synthesized images with **AE**, where the top row shows images from which the car type is taken. The second, third and fourth row show the decoder renderings using 2, 16 and 128 dimensions for the feature $N_\mathbf{v}$. (b) images synthesized with **AE+GAN**.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

the Shapenet dataset with a white background. We compare the methods **AE** and **AE+GAN** with different feature dimension of $N_\mathbf{v}$. We can observe that the transferring performance degrades for **AE**, when we increase the feature size of $N_\mathbf{v}$. As expected, the autoencoder takes the shortcut and tries to store all information into $N_\mathbf{v}$. The model **AE+GAN** instead renders images without loss of quality, independently of the feature dimension. Furthermore, none of the models exhibits the reference ambiguity: In all cases the viewpoint could be transferred correctly.

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 modify the data. We can see that both **AE** with 2 dimensions and **AE+GAN** with 128 separate the viewpoints well, but **AE** with 128 dimensions does not 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 **AE** and **AE+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 **AE** is sensitive to the size of $N_\mathbf{v}$, while **AE+GAN** is not. **AE+GAN** also achieves a better performance. We used the ShapeNet with a white background dataset also to compare the different normalization choices in Table 1. We evaluate the case when batch, instance and no normalization are used and compute the performance on the nearest neighbor classification task. We fixed the feature dimensions at 1024 for both $N_\mathbf{v}$ and $N_\mathbf{c}$ features 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.

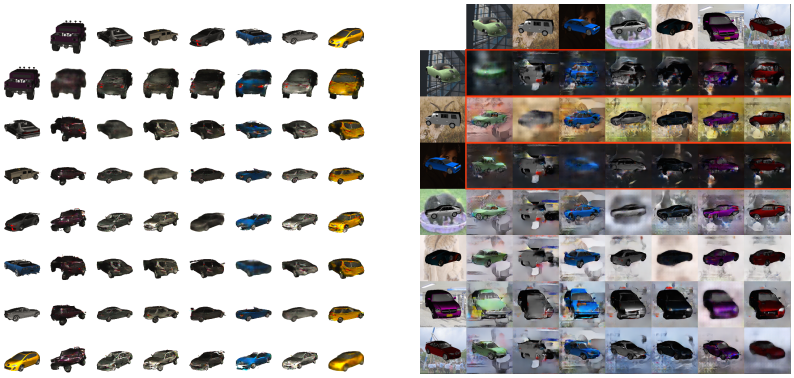
Table 1: Nearest neighbor classification on $N_\mathbf{v}$ and $N_\mathbf{c}$ features using different normalization techniques on ShapeNet with a white background.

Normalization	$N_\mathbf{v}$ mAP	$N_\mathbf{c}$ mAP
None	0.47	0.13
Batch	0.50	0.08
Instance	0.50	0.20



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

Figure 4: The effect of dimensions and objective function on N_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) **AE** 2 dim, (b) **AE** 128 dim, (c) **AE+GAN** 128 dim. (d) Mean average precision curves for the viewpoint prediction from the viewpoint feature using different models and dimensions for $N_\mathbf{v}$.



(a) (b)

Figure 5: ShapeNet transfers with (a) a white and (b) ImageNet background.

ShapeNet with ImageNet background. We render the ShapeNet dataset (same set of cars as in the previous section) with ImageNet images as background. The settings for the rendering (image size, viewpoints) are the same as in the case with a white background. We choose the backgrounds randomly for each car image, so that the overall dataset size of the data is the same, 80K. Since the image pairs use a different background during the training, the background is also part of the varying component. In Fig. 5 we show results on attribute transfer in the case of ShapeNet with a white and with ImageNet background. We found that the reference ambiguity does not emerge in the first dataset, but it does emerge in the second dataset, possibly due to the higher complexity. We highlight these incorrect viewpoint transfers with a red border (see top three rows in Fig. 5 (b). Nonetheless, we find that the proposed model more often than not correctly transfers the viewpoint. The background seems to transfer less well than the viewpoint, but we speculate that the background transfer might improve with better tuning and longer training.

5 Conclusions

In this paper we studied the challenges of disentangling factors of variation. We described the reference ambiguity and showed that it is inherently present in the task, when weak labels are used. Most importantly this ambiguity can be stated independently of the learning algorithm. We also introduced a novel method to train models to disentangle factors of variation. The model must be part of an autoencoder since our method requires that the representation is sufficient to reconstruct the input data. We have shown how the shortcut problem due to feature dimensionality can be kept under control through adversarial training. We demonstrated that training and transfer of factors of variation may not be guaranteed. However, in practice we observe that our trained model works well on most datasets and exhibits good generalization capabilities.

REFERENCES

Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

Hervé Bourlard and Yves Kamp. Auto-association by multilayer perceptrons and singular value decomposition. *Biological cybernetics*, 59(4):291–294, 1988.

Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. ShapeNet: An Information-Rich 3D Model Repository. Technical Report arXiv:1512.03012 [cs.GR], 2015.

Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *NIPS*, 2016.

Brian Cheung, Jesse A Livezey, Arjun K Bansal, and Bruno A Olshausen. Discovering hidden factors of variation in deep networks. *arXiv:1412.6583*, 2014.

Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. Adversarial feature learning. *arXiv:1605.09782*, 2016.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NIPS*, 2014.

Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

Geoffrey E Hinton, Alex Krizhevsky, and Sida D Wang. Transforming auto-encoders. In *International Conference on Artificial Neural Networks*, pp. 44–51. Springer, 2011.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *ICLR*, 2014.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Ming-Yu Liu and Oncel Tuzel. Coupled generative adversarial networks. In *Advances in Neural Information Processing Systems*, pp. 469–477, 2016.

Christos Louizos, Kevin Swersky, Yujia Li, Max Welling, and Richard Zemel. The variational fair autoencoder. In *ICLR*, 2016.

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*, pp. 5041–5049, 2016.

Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv:1511.06434*, 2015.

Scott Reed, Kihyuk Sohn, Yuting Zhang, and Honglak Lee. Learning to disentangle factors of variation with manifold interaction. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pp. 1431–1439, 2014.

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

Zhixin Shu, Ersin Yumer, Sunil Hadap, Kalyan Sunkavalli, Eli Shechtman, and Dimitris Samaras. Neural face editing with intrinsic image disentangling. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Luan Tran, Xi Yin, and Xiaoming Liu. Disentangled representation learning gan for pose-invariant face recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

Dmitry Ulyanov, Andrea Vedaldi, and Victor S. Lempitsky. Improved texture networks: Maximizing quality and diversity in feed-forward stylization and texture synthesis. In *CVPR*, 2017.

Kihyuk Sohn Dimitris Metaxas Manmohan Chandraker Xi Peng, Xiang Yu. Reconstruction for feature disentanglement in pose-invariant face recognition. *arXiv:1702.03041*, 2017.

Jimei Yang, Scott E Reed, Ming-Hsuan Yang, and Honglak Lee. Weakly-supervised disentangling with recurrent transformations for 3d view synthesis. In *NIPS*, 2015.