OVERCOMING THE DISENTANGLEMENT VS RECONSTRUCTION TRADE-OFF VIA JACOBIAN SUPERVISION

Anonymous authors
Paper under double-blind review

ABSTRACT

Learning image representations where the factors of variation are disentangled is typically achieved with an encoder-decoder architecture where a subset of the latent variables is constrained to correspond to specific factors, and the rest of them are considered nuisance variables. This widely used approach has an important drawback: as the dimension of the nuisance variables is increased, better image reconstruction is achieved, but the decoder has the flexibility to ignore the specified factors, thus losing the ability to condition the output on those factors. In this work, we propose to overcome this trade-off by progressively growing the dimension of the latent code, while constraining the Jacobian of the output image with respect to the disentangled variables to remain the same. As a result, the obtained models are effective at both disentangling and reconstruction. We demonstrate the aplicability of this method in both unsupervised and supervised scenarios for learning disentangled representations. In a facial attribute manipulation task, we obtain high quality image generation while smoothly controlling dozens of attributes with a single model. This is an order of magnitude more disentangled factors than state-of-the-art methods, while obtaining visually similar or superior results, and avoiding adversarial training[1].

1 INTRODUCTION

A desired characteristic of generative models is the ability to output realistic images while controlling one or more factors of variation underlying the image formation. This task, known as learning disentangled image representations, has been approached in the past by training an encoder-decoder network where a subset of the latent variables are constrained to correspond to given factors of variation, which can be specified (supervised) or learned from the data (unsupervised) (Bengio et al., 2013; Mathieu et al., 2016; Szabo et al., 2018; Hu et al., 2017; Kim & Mnih, 2018). The remaining latent variables are typically considered nuisance variables and are used by the autoencoder to complete the reconstruction of the image.

There exists one fundamental problem when learning disentangled representations, sometimes referred to as the "shortcut problem" (Hu et al., 2017; Szabo et al., 2018). If the dimension of the latent code is large enough, the decoder ignores the latent variables associated to the specified factors of variation, and achieves the reconstruction by using the bandwidth available in the nuisance variables. On the other hand, if the dimension of the latent code is very small, the decoder is encouraged to use the specified variables, but is also limited in the amount of information it can use for reconstruction, so the generated image is more distorted with respect to the input. Szabo et al. (2018) showed that this tradeoff between reconstruction and disentangling can indeed be traversed by varying the dimension of the latent code. However, no principled method exists to choose the optimal latent code dimension.

This problem was approached in the past by using additional methods to make sure the decoder output is a function of the specified factors in the latent code. One common approach, for example, consists in swapping the specified part of the latent code between different samples, and using adversarial training to ensure the output distribution is indeed conditioned to the specified factors (Mathieu et al., 2016; Lample et al., 2017; Szabó et al., 2017; Szabo et al., 2018).

[1] A supplementary material video demonstrating our method can be viewed at `https://www.dropbox.com/s/om4iqoycvf4vibl/iclr2019_v2_720.mp4?dl=0` (double-blind safe).

Based on these observations, we present a method for avoiding the shortcut problem that requires no adversarial training and achieves strong disentanglement and good reconstruction at the same time.

Our method consists on first training an autoencoder model, the teacher, where the dimension of the latent code is small, so that the autoencoder is able to effectively disentangle the factors of variation and condition its output on them. These factors can be specified in a supervised manner or learned from the data in an unsupervised way, as we shall demonstrate. After this teacher model is trained, we construct a student model that has a larger latent code dimension for the nuisance variables. For the student, we optimize the reconstruction loss as well as an additional loss function that constraints the variation of the output with respect to the specified latent variables to be the same as the teacher's.

In what follows, we consider autoencoder models (E, D), that receive an image $\boldsymbol{x}$ and produce a reconstruction $\hat{\boldsymbol{x}}$, (i.e., $D(E(\boldsymbol{x})) = \hat{\boldsymbol{x}}$). We consider that the latent code is always split into a specified factors part $\boldsymbol{y} \in \mathbb{R}^k$ and a nuisance variables part $\boldsymbol{z} \in \mathbb{R}^d$: $E(\boldsymbol{x}) = (\boldsymbol{y}, \boldsymbol{z}), D(\boldsymbol{y}, \boldsymbol{z}) = \hat{\boldsymbol{x}}$.

Our method uses a teacher autoencoder (E^T, D^T), with nuisance variables dimension d^T, and a student autoencoder (E^S, D^S) with nuisance variables dimension $d^S > d^T$. Because the dimension of the nuisance variables of the student is larger than in the first model, we expect a better reconstruction from it: $||\boldsymbol{x} - \hat{\boldsymbol{x}}^S|| < ||\boldsymbol{x} - \hat{\boldsymbol{x}}^T||$, for some norm.

At the same time, we want the student model to maintain the same disentangling ability as the teacher as well as the conditionning of the output on the specified factors. A first order approximation of this desired goal can be expressed as

$$\frac{\partial \hat{x}_j^S}{\partial y_i} \approx \frac{\partial \hat{x}_j^T}{\partial y_i}, \tag{1}$$

where $j \in \{1...H \cdot W \cdot C\}, i \in \{1...k\}$, and H, W and C are the dimensions of the output image.

In this paper we propose a method to impose the first-order constraint in equation 1, which we term Jacobian supervision. We show two applications of this method. First, we propose an algorithm that progressively disentangles the principal factors of variation in a dataset of images, in a completely unsupervised manner. Secondly, we use the Jacobian supervision to train an autoencoder model for face image generation, in which the factors of variation that can be controlled are facial attributes. Our resulting model outperforms the state-of-the-art in terms of both reconstruction quality and attribute manipulation ability.

2 Related Work

Autoencoders (Bengio et al., 2013; Hinton & Salakhutdinov, 2006; Kingma & Welling, 2014) are trained to reconstruct an input image while learning an internal low-dimensional representation of the input. Ideally, this representation should be 'disentangled', in the sense that each hidden unit in the latent code should encode one factor of variation in the formation of the image. There exist extensive literature addressing the problem of learning disentangled representations (Chen et al., 2016; Rifai et al., 2012; Mathieu et al., 2016; Szabó et al., 2017; Kingma et al., 2014; Perarnau et al., 2016; Cheung et al., 2014; Cogswell et al., 2015; Hu et al., 2017; Kim & Mnih, 2018).

Disentangled representations have two main applications. One is their use as rich features for classification (Rifai et al., 2012; Tran et al., 2017) or semi-supervised learning (Kingma et al., 2014). In the face recognition community, for example, disentanglement is often used to learn viewpoint- or pose-invariant features (Peng et al., 2017; Tran et al., 2017; Yang et al., 2015). The second main application is in a generative setting, where a disentangled representation can be used to control the factors of variation in the generated image (Yan et al., 2016; Lample et al., 2017; Mathieu et al., 2016; Szabó et al., 2017; Perarnau et al., 2016).

In recent years, with the advent of Generative Adversarial Networks (GANs) (Goodfellow et al., 2014), a broad family of methods uses adversarial training to learn disentangled representations (Mathieu et al., 2016; Szabó et al., 2017; Lample et al., 2017; Perarnau et al., 2016; Chen et al., 2016). In a generative setting, the adversarial discriminator can be used to assess the quality of a reconstructed image for which the conditioning factors do not exist in the training set (Mathieu et al., 2016; Szabó et al., 2017; Chen et al., 2016).

Another alternative, proposed in Fader Networks (Lample et al., 2017), is to apply the adversarial discriminator on the latent code itself, to prevent it from containing any information pertaining to the specified factors of variation. Then, the known factors of variation or attributes are appended to the latent code. This allows to specify directly the amount of variation for each factor, generating visually pleasing attribute manipulations. Despite being trained on one-hot binary labels, Fader Networks generalize remarkably well to real-valued attribute conditionning.

Despite recent advances (Arjovsky et al., 2017; Gulrajani et al., 2017), adversarial training remains a non-trivial min-max optimization problem, that in this work we wish to avoid. Other remarkable disentangling methods that require no adversarial training are: Cheung et al. (2014), where the cross-covariance between parts of the latent representation is minimized, so that the hidden factors of variation can be learned unsupervised and Higgins et al. (2016); Kim & Mnih (2018) where a factorized latent representation is learned using the VAE framework.

Related to the task of varying the factors of image generation is that of domain-transfer (Reed et al., 2015; Zhu et al., 2017; Isola et al., 2017; Choi et al., 2017; Donahue et al., 2016; Liu & Tuzel, 2016). Here the challenge is to "translate" an image into a domain for which examples of the original image are unknown and not available during training. For example, in the face generation task, the target domain can represent a change of facial attribute such as wearing eyeglasses or not, gender, age, etc. (Liu & Tuzel, 2016; Perarnau et al., 2016; Yan et al., 2016; Choi et al., 2017).

3 Unsupervised Progressive Learning of Disentangled Representations

In this section we detail how the Jacobian supervision motivated in Section 1 can be achieved, by ways of a practical example. We will use the Jacobian supervision to learn a disentangled image representation, where the main factors of variation are progressively discovered and learned unsupervised.

We start with a simple autoencoder model, the teacher T, identified by its encoder and decoder parts (E^T, D^T). The output of the encoder (the latent code) is split into two parts. One part corresponds to the factors of variation $\boldsymbol{y} \in \mathbb{R}^k$ and the other part corresponds to the nuisance variables, $\boldsymbol{z} \in \mathbb{R}^d$.

We begin by using $k = 2$ and $d = 0$, meaning that the latent code is only 2-dimensional. We consider the information encoded in these two variables as the two principal factors of variation in the dataset. This choice was done merely for visualization purposes.

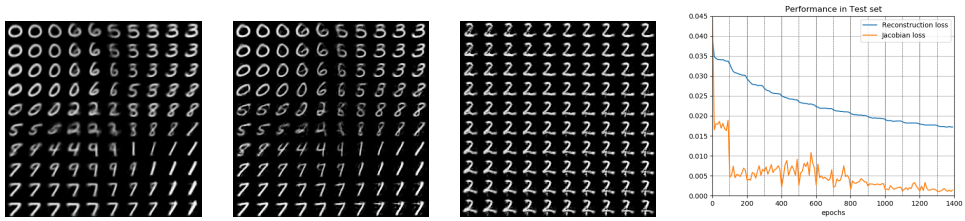
For this example, we trained a 3-layer multi-layer perceptron (MLP) on MNIST digits, using only the L2 reconstruction loss. We used BatchNorm at the end of the encoder, so that $\boldsymbol{y}$ is normally distributed inside a mini-batch. In Figure 1 (a) we show the result of sampling from this two-dimensional Gaussian and feeding the samples to the decoder D^T. The resulting digits are blurry, but the hidden variables learned to encode the digit class.

Next, we create a student autoencoder model (E^S, D^S), similar to the teacher, but with a larger latent code. Namely, $d = 1$ instead of $d = 0$, so that the latent code has now an extra dimension and reconstruction can be improved. In order to maintain the conditioning of the digit class by the two hidden variables $\boldsymbol{y}$, we will impose that the Jacobian of the student with respect to $\boldsymbol{y}$ be the same as that of the teacher (equation 1). How to achieve this is described next.

We take two random samples from the training set $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, and feed them to the student autoencoder, producing two sets of latent codes: $(\boldsymbol{y}_1^S, \boldsymbol{z}_1^S)$ and $(\boldsymbol{y}_2^S, \boldsymbol{z}_2^S)$, and two reconstructions $\hat{\boldsymbol{x}}_1^S$ and $\hat{\boldsymbol{x}}_2^S$, respectively. We then swap the parts of the latent code to form $(\boldsymbol{y}_2^S, \boldsymbol{z}_1^S)$ and $(\boldsymbol{y}_1^S, \boldsymbol{z}_2^S)$ and feed them to the student decoder to tobtain their respective reconstructions $\hat{\boldsymbol{x}}_{21}^S$ and $\hat{\boldsymbol{x}}_{12}^S$. We also feed the same pair of images to the teacher model to obtain $\boldsymbol{y}_1^T, \boldsymbol{y}_2^T, \hat{\boldsymbol{x}}_1^T, \hat{\boldsymbol{x}}_2^T$ (note that the teacher encoder does not produce a $\boldsymbol{z}$).

We observe, by a first-order Taylor expansion, that:

$$D^T(\boldsymbol{y}_2^T) = D^T(\boldsymbol{y}_1^T) + \boldsymbol{J}_T(\boldsymbol{y}_1^T)(\boldsymbol{y}_2^T - \boldsymbol{y}_1^T) + o^T(||\boldsymbol{y}_2^T - \boldsymbol{y}_1^T||), \tag{2}$$



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

Figure 1: Unsupervised learning of disentangled representations on MNIST digits, using Jacobian supervision. **(a)** Output of teacher model ($k = 2, d = 0$) when varying its two hidden units. **(b)** Output of the final student model ($k = 2, d = 14$), while varying the same hidden units. The Jacobian supervision makes the model maintain control of the factors of variation of the teacher, while obtaining much better reconstruction. **(c)** A student model ($k = 2, d = 14$) trained without Jacobian supervision loses the control of the factor of variation discovered by the teacher. **(d)** Performance curves for the test set, as training of the student progresses. The gray vertical bars indicate the moments where the latent code was progressively grown by one hidden unit.

and

$$D^S(\boldsymbol{y}_2^S, \boldsymbol{z}_1^S) = D^S(\boldsymbol{y}_1^S, \boldsymbol{z}_1^S) + \boldsymbol{J}_S(\boldsymbol{y}_1^S, \boldsymbol{z}_1^S)\left[(\boldsymbol{y}_2^S, \boldsymbol{z}_1^S) - (\boldsymbol{y}_1^S, \boldsymbol{z}_1^S)\right] + o^S(||(\boldsymbol{y}_2^S, \boldsymbol{z}_1^S) - (\boldsymbol{y}_1^S, \boldsymbol{z}_1^S)||) \tag{3}$$

$$= D^S(\boldsymbol{y}_1^S, \boldsymbol{z}_1^S) + \boldsymbol{J}_S(\boldsymbol{y}_1^S, \boldsymbol{z}_1^S)\left[(\boldsymbol{y}_2^S - \boldsymbol{y}_1^S, \boldsymbol{0})\right] + o^S(||\boldsymbol{y}_2^S - \boldsymbol{y}_1^S||), \tag{4}$$

where $\boldsymbol{J}_T$ and $\boldsymbol{J}_S$ are the Jacobian of the teacher and student decoders respectively. Note we do not indicate transpositions for clarity.

Suppose

$$\boldsymbol{y}_1^S = \boldsymbol{y}_1^T \text{ and } \boldsymbol{y}_2^S = \boldsymbol{y}_2^T, \tag{5}$$

and

$$D^T(\boldsymbol{y}_2^T) - D^T(\boldsymbol{y}_1^T) = D^S(\boldsymbol{y}_2^S, \boldsymbol{z}_1^S) - D^S(\boldsymbol{y}_1^S, \boldsymbol{z}_1^S) \tag{6}$$

then, by simple arithmetics,

$$\boldsymbol{J}_S(\boldsymbol{y}_1, \boldsymbol{z}_1)\left[(\boldsymbol{y}_2 - \boldsymbol{y}_1, \boldsymbol{0})\right] \approx \boldsymbol{J}_T(\boldsymbol{y}_1)(\boldsymbol{y}_2 - \boldsymbol{y}_1), \tag{7}$$

where, since we assume equation 5 holds, we dropped the superscripts for clarity.

What equation 7 expresses is that the partial derivative of the output with respect to the latent variables $\boldsymbol{y}$ in the direction of $(\boldsymbol{y}_2 - \boldsymbol{y}_1)$ of the student model approximates that of the teacher model.

To achieve this, our method consists essentially of enforcing the assumptions in equation 5 and equation 6 by simple reconstruction losses used during training of the student. Note one could exhaustively explore partial derivatives in all the canonical directions of the space. In our case however, by visiting many random pairs during training, we impose the constraint in equation 7 for random directions sampled from the data itself. This allows for more efficient training than exhaustive exploration.

Putting everything together, the loss function for training the student autoencoder with Jacobian supervision is composed of a reconstruction part $\mathcal{L}_{rec}$ and a Jacobian part $\mathcal{L}_{jac}$:

$$\mathcal{L}_{rec}(\boldsymbol{x}, E^S, D^S) := ||\boldsymbol{x} - D^S(E^S(\boldsymbol{x}))||_2 = ||\boldsymbol{x} - D^S(\boldsymbol{y}^S, \boldsymbol{z}^S)||_2 = ||\boldsymbol{x} - \hat{\boldsymbol{x}}^S||_2 \tag{8}$$

$$\mathcal{L}_{jac}(\boldsymbol{x}, E^S, D^S) := \lambda_y||\boldsymbol{y}^S - \boldsymbol{y}^T||_2 + \lambda_{diff}||\left(D^T(\boldsymbol{y}_j^T) - D^T(\boldsymbol{y}^T)\right) - \left(D^S(\boldsymbol{y}_j^S, \boldsymbol{z}^S) - D^S(\boldsymbol{y}^S, \boldsymbol{z}^S)\right)||_2 \tag{9}$$

where the subscript j indicates a paired random sample. For the experiments in Figure 1 we used $\lambda_y = 0.1, \lambda_{diff} = 1.0$.

Figure 2: 3rd to 6th principal factors of variation discovered by our unsupervised algorithm. The first two factors of variation are learned by the first teacher model, see Figure 1 (a). Each time a hidden unit is added to the autoencoder, a new factor of variation is discovered and learned. Each row shows the variation of the newly discovered factor for three different validation samples, while fixing all the other variables. The automatically discovered factors are related to stroke and handwriting style.

In practice, we found it also helps to add a term computing the cross-covariance between $\boldsymbol{y}$ and $\boldsymbol{z}$, to obtain further decorrelation between disentangled features (Cheung et al., 2014):

$$\mathcal{L}_{xcov}(\boldsymbol{y}, \boldsymbol{z}) := \sum_{ij} \left[\frac{1}{M} \sum_{m=1}^{M} (z_i^m - \bar{z}_i) \left(y_j^m - \bar{y}_j \right) \right]^2, \tag{10}$$

where M is the number of samples in the data batch, m is an index over samples and i, j index feature dimensions, and $\bar{z}_i$ and $\bar{y}_j$ denote means over samples. In our experiments we weigh this loss with $\lambda_{xcov} = 1e^{-}3$.

Once the student model is trained, it generates a better reconstructed image than the teacher model, thanks to the expanded latent code, while maintaining the conditionning of the output that the teacher had. The extra variable in the student latent code will be exploited to learn the next important factor of variation in the dataset. Examples of factors of variations progressively learned in this way are shown in Figure 2.

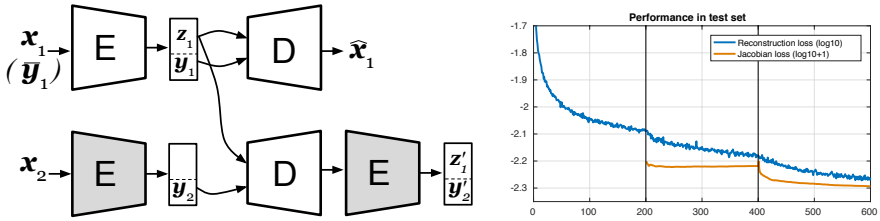
To progressively obtain an unsupervised disentangled representation we do the following procedure. After training of the student with $d = 1$ is finished, we consider it as the new teacher and we create a new student model with one more hidden unit, and then repeat the same procedure. Results of repeating this procedure 14 times, using 100 epochs for each stage are shown in Figure 1. The resulting final model maintains the conditionning of the digit class, while obtaining much better reconstruction. A model trained without Jacobian supervision, shown in Figure 1(c), also obtains good reconstruction but loses this conditionning.

4 Application to Facial Attribute Modification

In photographs of human faces, many factors of variation affect the image formation, such as subject identity, pose, illumination, viewpoint, etc., or even more subtle ones such as gender, age, expression. Modern facial manipulation algorithms allow one to control these factors in the generative process. Our goal here is to obtain a model that has both good control of these factors and produces faithful image reconstruction at the same time, using the Jacobian supervision introduced in Section 3. In this more challenging case, the disentangling will be first learned by a teacher autoencoder using available annotations and an original training procedure. After the teacher is trained to correctly disentangle and control said attributes, a student model will be trained to improve the visual quality of the reconstruction, while maintaing the attribute manipulation ability.

4.1 Model Architecture and Loss Function

Figure 3 (a) shows a diagram depicting the training architecture for the teacher model. As before, the latent code is split into a specified subvector $\boldsymbol{y} \in \mathbb{R}^k$ and an unspecified subvector $\boldsymbol{z} \in \mathbb{R}^d$. The training procedure is divided into two stages. First, the autoencoder reconstructs the input $\boldsymbol{x}$, while at the same time predicting inside the latent code the ground truth labels for the attributes $\boldsymbol{y}$. Second, the attributes part of the latent code is swapped with that of another training sample. The randomly fabricated latent code is fed into the decoder to produce a new image. Typically, this combined latent code is not represented in the training set, so evaluating the reconstruction is not possible. Instead, we use the same encoder to assess the new image: If the disentangling is achieved, the part of the



(a) (b)

Figure 3: **(a)** Diagram of the proposed training procedure. E and D always denote the same encoder and decoder module, respectively. Images $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ are randomly sampled and do not need to share any attribute or class. Their ground truth attribute labels are $\bar{\boldsymbol{y}}_1$ and $\bar{\boldsymbol{y}}_2$ respectively. The latent code is split into a vector predicting the attributes $\boldsymbol{y}$ and an unspecified part $\boldsymbol{z}$. Shaded E indicates its weights are frozen, i.e., any loss over the indicated output does not affect its weights. **(b)** Mean squared reconstruction error for validation samples. Black bars indicate the stages of dimensionality expansion (1st: $d = 2048 \rightarrow 4096, 2^{nd}$: $d = 4096 \rightarrow 8192$). Reconstruction quality improves while maintaining the better disentanglement of a short code.

latent code that is not related to the attributes ($\boldsymbol{z}$) should be the same for the two images. Also, the predicted labels should match those of the sample from which they were swapped.

Let $\boldsymbol{x} \in \mathbb{R}^{H \times W \times 3}$ be an image with annotated ground truth binary attributes $\bar{\boldsymbol{y}} \in \{-1, 1\}^k$, where k is the number of attributes for which annotations are available. Our goal is to learn the parameters of the encoder $E^T : \mathbb{R}^{H \times W \times 3} \rightarrow \mathbb{R}^{k+d}$ and the decoder $D^T : \mathbb{R}^{k+d} \rightarrow \mathbb{R}^{H \times W \times 3}$ such that $E^T(\boldsymbol{x}) = (\boldsymbol{y}, \boldsymbol{z})$ and $D^T(\boldsymbol{y}, \boldsymbol{z}) \approx \boldsymbol{x}$ (Figure 3 (a)). Ideally, $\boldsymbol{y}$ should encode the specified attributes of $\boldsymbol{x}$, while $\boldsymbol{z}$ should encode the remaining information necessary for reconstruction. In what follows, we describe step by step the loss function used for training, which consists of the sum of multiple loss terms. Note that, contrary to relevant recent methods (Mathieu et al., 2016; Lample et al., 2017; Szabó et al., 2017), our proposed method does not require adversarial training.

Reconstruction loss. The first task of the autoencoder is to reconstruct the input image. The first term of the loss is given by the L2 reconstruction loss, as in equation 8.

Prediction loss. In order to encourage $\boldsymbol{y}$ to encode the original attributes of $\boldsymbol{x}$ indicated in the ground truth label $\bar{\boldsymbol{y}}$, we add the following penalty based on the hinge loss with margin 1:

$$\mathcal{L}_{pred}(\boldsymbol{y}, \bar{\boldsymbol{y}}) = \frac{1}{k} \sum_{i=1}^{k} \max(1 - y_i \bar{y}_i, 0), \tag{11}$$

where the subscript $[i]$ indicates the ith attribute. Compared to recent related methods (Perarnau et al., 2016; Lample et al., 2017), the decoder sees the real-valued predicted attributes instead of an inserted vector of binary attribute labels. This allows the decoder to naturally learn from continuous attribute variables, leaving a degree of freedom to encode subtle variations of the attributes.

Cycle-consistency loss. Recall our goal is to control variations of the attributes in the generated image, with the ability to generalize to combinations of content and attributes that are not present in the training set. Suppose we have two randomly sampled images $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ as in Figure 3(a). After obtaining $(\boldsymbol{y}_1, \boldsymbol{z}_1) = E(\boldsymbol{x}_1)$ and $(\boldsymbol{y}_2, \boldsymbol{z}_2) = E(\boldsymbol{x}_2)$, we form the new artificial latent code $(\boldsymbol{y}_2, \boldsymbol{z}_1)$. Ideally, using this code, the decoder should produce an image with the attributes of $\boldsymbol{x}_2$ and the content of $\boldsymbol{x}_1$. Such an image generally does not exist in the training set, so using a reconstruction loss is not possible. Instead, we resort to a cycle-consistency loss (Zhu et al., 2017). We input this image to the same encoder, which will produce a new code that we denote as $(\boldsymbol{y}_2', \boldsymbol{z}_1') = E^T(D^T(\boldsymbol{y}_2, \boldsymbol{z}_1))$. If the decoder correctly generates an image with attributes $\boldsymbol{y}_2$, and the encoder is good at predicting the input image attributes, then $\boldsymbol{y}_2'$ should predict $\boldsymbol{y}_2$. We use again the hinge loss to enforce this:

$$\mathcal{L}_{cyc_1} = \sum_{i=1}^{k} \max(1 - y_{2i} y_{2i}', 0). \tag{12}$$

Here we could have used any random values in lieu of $\boldsymbol{y}_2$. However, we found that sampling predictions from the data eases the task of the decoder, as it is given combinations of attributes that it has

already seen. Despite this simplification, the decoder shows remarkable generalization to unseen values of the specified attributes $\boldsymbol{y}$ during evaluation.

Finally, we add a cycle-consistency check on the unspecified part of the latent code, $\boldsymbol{z}_1$ and $\boldsymbol{z}_1'$:

$$\mathcal{L}_{cyc_2} = ||\boldsymbol{z}_1 - \boldsymbol{z}_1'||_2 \tag{13}$$

Encoder freezing. The training approach we just described presents a major pitfall. It could happen that the reversed autoencoder learns to replicate the input code $(\boldsymbol{y}_2, \boldsymbol{z}_1)$ by encoding this information inside a latent image in whatever way it finds easier, that does not induce a natural attribute variation. To avoid this issue, a key ingredient of the procedure is to freeze the weights of the encoder when back-propagating $\mathcal{L}_{cyc_1}$ and $\mathcal{L}_{cyc_2}$. This forces the decoder to produce a naturally looking image so that the encoder correctly classifies its attributes.

Global teacher loss. Overall, the global loss used to train the teacher is the sum of the 5 terms:

$$\mathcal{L}(\theta_E, \theta_D) = \lambda_1 \mathcal{L}_{rec} + \lambda_2 \mathcal{L}_{pred} + \lambda_3 \mathcal{L}_{xcov} + \lambda_4 \mathcal{L}_{cyc_1} + \lambda_5 \mathcal{L}_{cyc_2}, \tag{14}$$

where $\lambda_i \in \mathbb{R}, i = 1:5$ represent weights for each term in the sum. Details on how their values are found and how we optimize equation 14 in practice are described in the next section.

Student training. After the teacher is trained, we create a student autoencoder model with a larger dimension for the nuisance variables $\boldsymbol{z}$ and train it using Jacobian supervision (equation 8 and equation 9), as detailed in the next section.

4.2 Implementation

We implement both autoencoders as Convolutional Neural Networks (CNN). Implementation details are detailed in the Appendix. We train and evaluate our method on the standard CelebA dataset (Liu et al., 2015), which contains 200,000 aligned faces of celebrities with 40 annotated attributes.

The unspecified part of latent code ($\boldsymbol{z}$) of the autoencoder is implemented as a feature map of 512 channels of size 2×2. To encode the attributes part $\boldsymbol{y}$, we concatenate an additional $k = 40$ channels. At the output of the encoder the values of these 40 channels are averaged, so the actual latent vector has $k = 40$ and $d = 2048$, dimensions for $\boldsymbol{y}$ and $\boldsymbol{z}$ respectively.

The decoder uses a symmetrical architecture and, following Lample et al. (2017), the attribute prediction $\boldsymbol{y}$ is concatenated as constant channels to every input feature map of the decoder.

We perform grid search to find the values of the weights in equation 14 by training for 10 epochs and evaluating on a hold-out validation set. The values we used in the experiments in this paper are $\lambda_1 = 10^2, \lambda_2 = 10^{-1}, \lambda_3 = 10^{-1}, \lambda_4 = 10^{-3}, \lambda_5 = 10^{-4}$

4.2.1 Teacher Training

At the beginning of the training of the teacher, the weights of the cycle-consistency losses λ_4 and λ_5 are set to 0, so the autoencoder is only trained for reconstruction ($\mathcal{L}_{rec}$), attribute prediction ($\mathcal{L}_{pred}$) and linear decorrelation ($\mathcal{L}_{cov}$). After 100training epochs, we restart the training turning on $\mathcal{L}_{cyc_1}$ and $\mathcal{L}_{cyc_2}$ and resume for another 100 epochs. At each iteration, we do the parameter updates in two separate steps. We first update for $\mathcal{L}_1 = \lambda_1 \mathcal{L}_{rec} + \lambda_2 \mathcal{L}_{pred} + \lambda_3 \mathcal{L}_{cov}$. Then, freezing the encoder, we do the update (only for the decoder), for $\mathcal{L}_2 = \lambda_4 \mathcal{L}_{cyc_1} + \lambda_5 \mathcal{L}_{cyc_2}$.

4.2.2 Student Training

After the teacher autoencoder training is completed, we create the student model by appending new convolutional filters to the output of the encoder and the input of the decoder, so that the effective dimension of the latent code is increased.

In this experiment, we first doubled the size of the latent code from $d = 2048$ to $d = 4096$ at the 200th epoch and then from $d = 4096$ to $d = 8192$ at the 400th epoch. Note that this is different to the illustrative experiment of Section 3, where we grew d by one unit at at time.

We initialize the weight of the student with the weight of the teacher wherever possible. Then, we train the student using the reconstruction loss (equation 8) and the Jacobian loss (equation 9)

Table 1: Summary comparison of the characteristics of recent related methods. Our method has advantages over each of them, and together with Fader Networks are the only ones to generate 256×256 images while continuously varying the generated facial attributes.

Method	end-to-end training	requires aligned pairs	requires adversarial training	face image resolution	number of attributes per model	generates continuous attributes
CoGAN	yes ✓	no ✓	yes ✗	128x128 ✗	1 ✗	no ✗
IcGAN	no ✗	no ✓	yes ✗	64x64 ✗	18 ✓	no ✗
Attribute2Image	no ✓	no ✓	no ✓	64x64 ✗	1 ✗	yes ✓
StarGAN	yes ✓	no ✓	yes ✗	128x128 ✗	7 ✗	no ✗
Fader Networks	yes ✓	no ✓	yes ✗	256x256 ✓	3 ✗	yes ✓
This work	yes ✓	no ✓	no ✓	256x256 ✓	32 ✓	yes ✓

as defined in Section 3. After training, we verified visually that the modification of attributes is still effective, unsing an interactive interface, as demonstrated in the supplementary matrial video[2]. Figure 3 (b) shows the reconstruction loss for validation samples during training. Overall, training took 3.5 days on a GTX1080 GPU (8.5 minutes per epoch).

4.3 Experimental Results

From CelebA, we use 162,770 images of size 256x256 for training and the rest for validation. All the result figures in this paper show images from the validation set and were obtained using the same single model. Source code and pre-trained models to reporduce all experiments in this paper will be made publicly available upon publication.

Figure 4 shows the result of manipulating 32 attributes for eight different subjects, while Figure 3(b) shows the evolution of the reconstruction loss on the validation set. A video showing interactive control of the attributes is included as supplementary material. In comparison, a student model with enlarged latent code but that continues with the training procedure as the teacher, without Jacobian supervision, achieves good reconstruction but loses the effective conditionning on the attributes. We kindly refer the reader to the supplementary video to observe this comparison. Note that our model is designed to learn the 40 attributes, however in practice there are 8 of them which the model does not learn to manipulate, possibly because they are poorly represented in the dataset (e.g. sideburns, wearing necktie) or too difficult to generate (e.g. wearing hat, wearing earrings).

Table 1 presents a summarized comparison of the most recent methods related to the task of manipulating facial attributes and the proposed one. To the best of our knowledge, Fader Networks (Lample et al., 2017) constitutes the state-of-the-art in face image generation with continuous control of the facial attributes. Because of the harmful effect of using an adversarial discriminator on the latent code, Fader Networks are limited in that a single model cannot control more than a few attributes (the maximum demonstrated in Lample et al. (2017) is 3) [3]. A direct comparison to Fader Networks, using the author's implementation is shown in Figure 5.

5 Conclusion

A natural trade-off between disentanglement and reconstruction exists when learning image representations using autoencoder architectures. In this paper, we showed it is possible to overcome this trade-off by first learning a teacher model that is good at disentangling and then imposing the Jacobian of this model with respect to specific variables to a student model that is good at reconstruction. The student model then becomes good at both disentangling and reconstruction. We showed two example applications of this idea. One was to progressively learn the principal factors of variation in a dataset, in an unsupervised manner. The second application is a conditional generative model that is able to manipulate facial attributes in human faces. The obtained model is able to manipulate one order of magnitude more facial attributes than state-of-the-art methods, while obtaining similar or superior visual results, and requiring no adversarial training.

[2] https://www.dropbox.com/s/om4iqoycvf4vibl/iclr2019_v2_720.mp4?dl=0

[3] This limitation is also documented in the author's Github repository: https://github.com/facebookresearch/FaderNetworks

Figure 4: Results of our method for attribute manipulation. All the images were produced with the same autoencoder model and were not seen during training. Our method is able to achieve both a faithful reconstruction and effective attribute manipulation.

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

Figure 5: Comparison with Fader Networks (Lample et al., 2017). **(a)** Original image. **(b)** Reconstruction of Fader Networks with the provided 'eyeglasses' model. **(c)** Our teacher model achieves a sharper reconstruction using the same latent code dimension, and is able to effectively maniuplate up to 32 attributes, instead of only one. **(d)** Result of amplifying age with Fader Networks with the provided aging model. **(e)** Our result for the same task exhibits a more natural aging aspect.

REFERENCES

Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.

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.

Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. InfoGAN: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in Neural Information Processing Systems*, pp. 2172–2180, 2016.

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

Yunjey Choi, Minje Choi, Munyoung Kim, Jung-Woo Ha, Sunghun Kim, and Jaegul Choo. StarGAN: Unified generative adversarial networks for multi-domain image-to-image translation. *arXiv preprint arXiv:1711.09020*, 2017.

Michael Cogswell, Faruk Ahmed, Ross Girshick, Larry Zitnick, and Dhruv Batra. Reducing overfitting in deep networks by decorrelating representations. *arXiv preprint arXiv:1511.06068*, 2015.

Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. Adversarial feature learning. *arXiv preprint 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 *Advances in Neural Information Processing Systems*, pp. 2672–2680, 2014.

Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of Wasserstein GANs. In *Advances in Neural Information Processing Systems*, pp. 5769–5779, 2017.

Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. 2016.

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

Qiyang Hu, Attila Szabó, Tiziano Portenier, Paolo Favaro, and Matthias Zwicker. Disentangling factors of variation by mixing them. *arXiv preprint arXiv:1711.07410*, 2017.

Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. *arXiv preprint*, 2017.

Hyunjik Kim and Andriy Mnih. Disentangling by factorising. *arXiv preprint arXiv:1802.05983*, 2018.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

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

Diederik P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. Semi-supervised learning with deep generative models. In *Advances in Neural Information Processing Systems*, pp. 3581–3589, 2014.

Guillaume Lample, Neil Zeghidour, Nicolas Usunier, Antoine Bordes, Ludovic Denoyer, et al. Fader networks: Manipulating images by sliding attributes. In *Advances in Neural Information Processing Systems*, pp. 5969–5978, 2017.

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

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

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. 5040–5048, 2016.

Xi Peng, Xiang Yu, Kihyuk Sohn, Dimitris N Metaxas, and Manmohan Chandraker. Reconstruction-based disentanglement for pose-invariant face recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1623–1632, 2017.

Guim Perarnau, Joost van de Weijer, Bogdan Raducanu, and Jose M. Álvarez. Invertible Conditional GANs for image editing. In *NIPS Workshop on Adversarial Training*, 2016.

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.

Salah Rifai, Yoshua Bengio, Aaron Courville, Pascal Vincent, and Mehdi Mirza. Disentangling factors of variation for facial expression recognition. In *European Conference on Computer Vision*, pp. 808–822. Springer, 2012.

Attila Szabó, Qiyang Hu, Tiziano Portenier, Matthias Zwicker, and Paolo Favaro. Challenges in disentangling independent factors of variation. *arXiv preprint arXiv:1711.02245*, 2017.

Attila Szabo, Qiyang Hu, Tiziano Portenier, Matthias Zwicker, and Paolo Favaro. Understanding degeneracies and ambiguities in attribute transfer. In *The European Conference on Computer Vision (ECCV)*, September 2018.

Luan Tran, Xi Yin, and Xiaoming Liu. Disentangled representation learning GAN for pose-invariant face recognition. In *CVPR*, volume 3, pp. 7, 2017.

Xinchen Yan, Jimei Yang, Kihyuk Sohn, and Honglak Lee. Attribute2image: Conditional image generation from visual attributes. In *European Conference on Computer Vision*, pp. 776–791. Springer, 2016.

Jimei Yang, Scott E Reed, Ming-Hsuan Yang, and Honglak Lee. Weakly-supervised disentangling with recurrent transformations for 3d view synthesis. In *Advances in Neural Information Processing Systems*, pp. 1099–1107, 2015.

Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *IEEE International Conference on Computer Vision*, 2017.

A APPENDIX

A.1 IMPLEMENTATION DETAILS FOR SECTION 3

For the autoencoder utilized for experients in Section 3, we used the following architecture. For the encoder:

$$F(768, 256) \rightarrow ReLU \rightarrow F(256, 128) \rightarrow ReLU \rightarrow F(128, 64) \rightarrow ReLU \rightarrow FC(64, k + d)$$

where $F(I, O)$ indicates a fully connected layer with I inputs and O outputs. For the first teacher model ($k = 2, d = 0$), we also used BatchNorm after the encoder output.

The decoder is the exact symmetric of the encoder, with a Tanh layer appended at the end.

We used Adam Kingma & Ba (2014) with a learning rate of $3e - 4$, a batch size of 128 and weight decay coefficient $1e - 6$.

A.2 IMPLEMENTATION DETAILS FOR SECTION 4

Following Lample et al. (2017), we use convolutional blocks of Convolution-BatchNorm-ReLU layers and a geometric reduction in spatial resolution by using stride 2. The convolutional kernels are all of size 4×4 with padding of 1, and we use Leaky ReLU with slope 0.2. The input to the encoder is a 256×256 image. Denoting by k the number of attributes, the encoder architecture can be summarized as:

$$C(16) \rightarrow C(32) \rightarrow C(64) \rightarrow C(128) \rightarrow C(256) \rightarrow C(512) \rightarrow C(512 + k),$$

where $C(f)$ indicates a convolutional block with f output channels.

The decoder architecture can be summarized as:

$$D(512 + k) \rightarrow D(512 + k) \rightarrow D(256 + k) \rightarrow D(128 + k) \rightarrow D(64 + k) \rightarrow D(32 + k) \rightarrow D(16 + k),$$

where $D(f)$ in this case indicates a deconvolutional block doing $\times 2$ upsampling (using transposed convolutions, BatchNorm and ReLU) with f input channels.

We train all networks using Adam Kingma & Ba (2014), with learning rate of 0.002, $\beta_1 = 0.5$ and $\beta_2 = 0.999$. We use a batch size of 128.

A.2.1 STUDENT MODEL

For the student model, we only need to change the last layer in the encoder from $C(512 + k)$ to $C(1024 + k)$ in the first stage and $C(2048 + k)$ in the second stage. Similarly, the first layer of the decoder needs to be changed from $D(512 + k)$ to $D(1024 + k)$ in the first stage and $D(2048 + k)$ in the second stage.