TOWARDS OBJECT SHAPE TRANSLATION THROUGH UNSUPERVISED GENERATIVE DEEP MODELS

Lies Bollens, Tinne Tuytelaars, José Oramas M.

KU Leuven, ESAT-PSI

ABSTRACT

This paper focuses on the problem of unsupervised image-to-image translation. More specifically, we aim at finding a translation network such that objects and shapes that only appear in the source domain are translated to objects and shapes only appearing in the target domain, while style color features present in the source domain remain the same. To achieve this, we use a domain-specific variational autoencoder and represent each image in its latent space representation. In a second step, we learn a translation between latent spaces of different domains using generative adversarial networks. We evaluate this framework on multiple datasets and verify the effect of multiple perceptual losses. Experiments on the MNIST and SVHN datasets show the effectiveness of the proposed translation method.

1. INTRODUCTION

Image-to-image translation transforms an image from one domain to a corresponding image in another domain. Using supervised learning, significant research has lead to impressive results, e.g. [1, 2, 3, 4]. However, a supervised setup requires corresponding pairs of images from both domains. Obtaining these pairs is not trivial. It may be labor intensive, e.g. for a segmentation task. Sometimes the translation is simply not available or not uniquely defined, e.g. when transforming cats to dogs or when translating an image from virtual reality to the real world. As an alternative, unsupervised methods are interesting to investigate. However, this makes the translation task inherently harder. As there are many possible alignments between the two domains, it requires a mechanism to ensure that the generated images are indeed a "good" translation.

Recent efforts [5, 2, 6] have made significant progress in this direction - specifically for the case when translating the style (appearance) of a given image while preserving structures or shapes depicted therein. A common practice is to enforce a cycle consistency constraint [6], translating an image from a source domain to a target domain, and back. Using this constraint, it has been shown one can change the color/style distribution across domains. However, changing shapes turns out to be more complex.

First, it's important to note that, while necessary, cycle-consistency is not a sufficient requirement to ensure proper image translation. There are in fact a multitude of solutions satisfying cycle-consistency. For style transfer, the spatial structure of the network architecture adequately channels the information flow between the input and output images, resulting in mostly local adjustments. This regularizes the problem and imposes shape preservation. For the case of shape translation, the extent of the depicted content is expected to change. This requires non-local deformations that can no longer benefit from regularization via the spatial structure of the network.

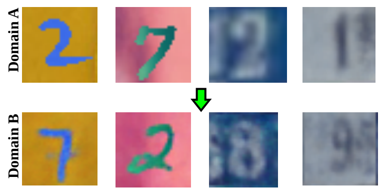

Fig. 1: Shape translation while preserving the style of the image.

In this paper, we learn specialized domain-specific latent representations. Next, we model an explicit mapping between these representations. We also add further constraints evaluating the quality of the translation. More specifically, we do not assume that there exists a shared latent space, as in [7, 8], but rather map images to their respective latent space first. To this end, we train two variational autoencoders [9] (VAE), one for each domain, separately. In a second step, we learn a mapping between the two domain-specific latent spaces using an extension of the GAN architecture. We propose a new architecture by combining VAEs and GANs and obtain successful mappings using perceptual losses, e.g. difference of Structural Similarity loss [10]. As test setting we consider the structure of numerical digits as the shape features to be translated across different digit classes (Please see Fig. 1).

The rest of the paper is organized as follows. Sec. 2 positions our work w.r.t. the literature. In Sec. 3, we describe our methodology. This is followed by our evaluation in Sec. 4. Finally, we draw conclusions in Sec. 5.

2. RELATED WORK

Image-to-Image Translation. GANs [11] are widely used when it comes to image generation [3, 12, 13, 14, 15, 16, 17]. Since GANs are able to create sharp and realistic outputs, they are useful for image-to-image translation. In [1] a network architecture is proposed to transform images based on conditional GANs [18]. This principle is further developed in [19] to generate high-resolution images. The drawback of these models however, is the use of supervised learning, which makes them not applicable in many situations. In contrast, our translation network is unsupervised in the sense that it does not require paired examples for training.

Unsupervised learning methods are an alternative whenever paired examples are not available. Recent unsupervised methods enforce a series of constraints in order to steer examples from different domains to converge on a shared latent representation. Examples of these constraints include cycle-consistency [6], semantic features [8], pixel values [20], pixel gradients [21], class labels [21] or the pairwise distance between samples [8, 7, 22]. These methods convert the images to a shared latent space before translating them to the target domain. This assumes that there exists a shared latent space between both domains. In contrast, we learn domain-specific latent spaces and learn a translation between them.

Neural Style Transfer. Another way to translate images is by making use of a neural style loss. [5] uses a single image as an example for the style and then divides the source image into a content and a style representation. An optimization algorithm is then run to transfer the style of the image, while keeping the content. The division between style and content is made using a perceptual loss, calculated using output features of a pre-trained neural network. In [23] this idea is used to generate photo-realistic output images. [2] extends this idea by incorporating a feed-forward neural network to translate the images. A common aspect that characterizes these methods is that the shape of the content in their inputs is aligned with that of their outputs while depicting different styles. Here we follow a complementary direction where we aim at translating, i.e. modifying, the shape while preserving the style between input and output.

Shape Translation. One of the first methods aiming at shape translation was proposed in [24], where clothing-items as worn by a person are translated to a catalog-like image depicting only that item. Recently, [25] proposed a warping-based method aimed at virtual try-on of clothing items. Similar to our work, these methods aim at translating the shape of the objects (clothing-items) while preserving their style. In contrast to our method, these methods need paired data during training. Closer to our work very recently [26] proposed to perform unpaired shape translation. However, different from them, we relax the assumption that a shared latent space exists between the domains of interest.

3. METHODOLOGY

We define a mapping between two domains A and B. We have samples $\{x_i\}_{i=1}^n \in dom(A)$ and $\{y_i\}_{i=1}^n \in dom(B)$, but there exists no correspondence between the samples of different domains. We create a network architecture based

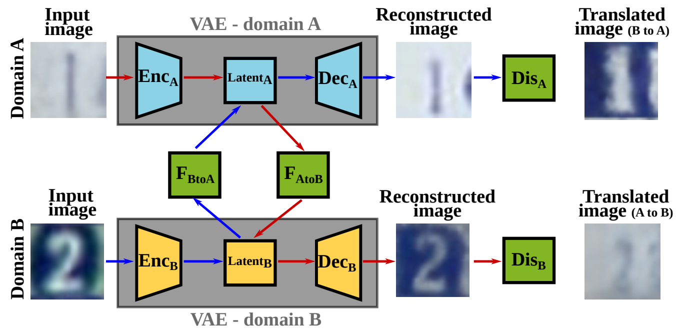

Fig. 2: Overview of the proposed network architecture. Each domain is modelled by a domain-specific VAE while shape translation across domains is handled by a specific module (in green) which maps one latent space into the other. End-to-end translation from domains A to B, and v.v., are achieved by following the red and blue lines, respectively.

on VAEs [9, 27] and GANs [11], more specifically the cycleGAN [6] architecture. Our architecture is illustrated in Figure 2. We train it in two steps. In the first step, we train two VAEs separately: one with images from domain A and one with images from domain B (blue and yellow blocks in Fig. 2). In a second step, we train the mapping between the two resulting latent spaces. This is achieved through a GAN where the generator is directly focuses on improving the mapping components (see below) and additional discriminators assess the quality of images produced considering the output of the mapping components (green blocks in Fig. 2). The network learns to translate images from domain A to B and from domain B to A at the same time as illustrated in Fig. 2 by following the blue and red lines, respectively. Formally speaking, this is equivalent to the following equations: $Gen_A(x){=}Dec_A(F_{BtoA}(Enc_B(x)))$ and $Gen_B(x){=}Dec_B(F_{AtoB}(Enc_A(x)))$. In the next sections we describe the different modules.

3.1. Modelling domain information

The two VAEs are trained separately. The encoder and decoder pair $\{Enc_A, Dec_A\}$ constitute a VAE for domain A. An image from domain A is mapped via Enc_A to its latent space representation and then decoded via Dec_A to the reconstructed version of the input image (upper part of Fig. 2). The VAEs are implemented using the standard practice, as described in [9]. The encoder and decoder $\{Enc_B, Dec_B\}$ are designed in a similar way (lower part of Fig. 2). We apply the following loss function at this step:

$$\mathcal{L}_{VAE}{=}-\mathbb{E}_{q(z|x)}\left[\log p(x|z)\right] + \mathcal{D}_{KL}\big(q(z|x)||p(z)\big) \quad (1)$$

where $z{\sim}Enc(x){=}q(z|x)$ is the latent space representation of an input image x, $\tilde{x}{\sim}Dec(z){=}p(x|z)$ is the reconstructed version of input image x and $\mathcal{D}_{KL}$ is the Kullback-Leibler divergence.

3.2. Shape translation

Once the VAEs have been trained, we train the networks that learn a mapping between the two latent spaces. We take the standard cycleGAN [6] network and extend it to be able to transform shapes. We conducted some experiments with the standard cycleGAN network and while we were able to change the color distribution, we were not able to change shapes and structures – hence the need to extend this framework. The cycleGAN architecture contains two GANs and imposes an extra cycle-consistency constraint. There are three losses in total: two for training the GANs and a third one to make sure the cycle-consistency constraint is enforced. The GAN loss function is:

$$\begin{aligned}\mathcal{L}_{GAN}(Gen_A, Dis_A) = &\ \mathbb{E}_{x \in dom(A)} \left[log(Dis_A(x) \right] \\ &+ \mathbb{E}_{y \in dom(B)} \left[log(1 - Dis_A(Gen_A(y)) \right]\end{aligned} \tag{2}$$

(and similar for domain B). The cycle-consistency loss can be formulated as follows:

$$\begin{aligned}\mathcal{L}_{cyc} = &\ \mathbb{E}_{x \in dom(A)} \left[||Gen_B(Gen_A(x)) - x||_1 \right] \\ &+ \mathbb{E}_{y \in dom(B)} \left[||Gen_A(Gen_B(y)) - y||_1 \right]\end{aligned} \tag{3}$$

We add an extra loss term, to favor "good" translations, i.e. those that preserve certain aspects between the input and output image:

$$\begin{aligned}\mathcal{L}_{sim}(x, y, Gen_A, Gen_B) = &\ \mathbb{E}_{x \in dom(B)}\, d(x, Gen_A(x)) \\ &+ \mathbb{E}_{y \in dom(A)}\, d(y, Gen_B(y))\end{aligned} \tag{4}$$

where d is a distance function. We compare the two following alternatives for the distance functions:

Color Histogram Distance. First we highlight the most frequent color within the image by quantizing each channel of the RGB space into 20 intensity values. The distance between two images is then calculated by taking the mean-squared distance between their quantized color representations.

Difference of Structural Similarity (DSSIM) [10]. This function calculates the difference between the SSIM values for two images. This is a perceptual loss function designed to create visually pleasing results. The SSIM index is defined as

$$SSIM(x, y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \tag{5}$$

where x and y are vectors of image intensities, with their corresponding mean and standard deviation values μ_x, μ_y and σ_x, σ_y. σ_{xy} the covariance and $c_1 = (k_1 L)^2$, $c_2 = (k_2 L)^2$ two variables to stabilize the division. L is the dynamic range of the pixel values, typically taken to be $2^n - 1$, with n equal to the number of bits per pixel, $k_1 = 0.01$ and $k_2 = 0.03$. The two images to be compared are split into N smaller patches x_p and y_p, typically taken to be 3×3. The SSIM loss is calculated by sliding a window over the images and calculating the SSIM value for all these small regions. The final value is then taken as the mean. The SSIM loss function can be written as

$$\mathcal{L}^{SSIM}(x, y) = \frac{1}{N} \sum_{p=1}^{N} (1 - SSIM(x_p, y_p)) \tag{6}$$

Combining all this, our total loss function becomes:

$$\begin{aligned}\mathcal{L} = &\ \mathcal{L}_{GAN}(Gen_A, Dis_A) + \mathcal{L}_{GAN}(Gen_B, Dis_B) \\ &+ \lambda_{cycle} * \mathcal{L}_{cyc}(Gen_A, Gen_B) \\ &+ \lambda_{sim} * \mathcal{L}_{sim}(x, y, Gen_A, Gen_B)\end{aligned} \tag{7}$$

The training at this step is equal to solving a mini-max problem, where the encoders, decoders and generators are playing against the discriminators. The first player has to defeat the second player and minimize the cycle loss and the similarity loss at the same time. We train by iteratively updating the different components. Firstly, we update the encoders, decoders and mapping networks, while keeping the discriminators fixed. Secondly, we update the two discriminators, while the other components are fixed.

4. EVALUATION

Implementation details. We use Adam [28] for the training. For training the VAEs, we train with a learning rate of 0.0001, with $\beta_1 = 0.9, \beta_2 = 0.999$ and with a batch size of 25. For the translation step, we train with a learning rate of 0.0001 for both the generator and the discriminator components, with values $\beta_1 = 0.5, \beta_2 = 0.9$ and with a batch size of 1. The discriminator is updated 5 times for each update of the generator. We use standard values ($\lambda_{cycle} = 2, \lambda_{sim} = 5$) for the hyper-parameters. We initialize the weights from a Gaussian distribution, with mean 0 and a standard deviation of 0.02.

Our framework contains several modules. The encoders consist of four convolutional layers with a stride of 2. The decoders consist of four transposed convolutional layers. These networks have the ReLU activation function. The mapping from domain A to domain B (and v.v.) is done with 6 Residual Blocks [29]. The discriminator has 4 convolutional layers and uses patchGANs [1]. We use instance normalization [30] and LeakyReLUs for both of these modules.

4.1. Translating numbers in synthetic images

In this first experiment, we test our network to determine which loss function and which hyper-parameters work best. To this end, we use a simple synthetic dataset derived from MNIST. Since we assume, the shape/structure is the descriptive characteristic of the domains, in this experiment we consider classes 1 and 3 as the domains to be translated. In total there are 6742 images of class 1 and 6131 images of class 3 in the training partition of the original dataset. We add color styles to all these images using the color MNIST procedure[1].

[1]https://www.wouterbulten.nl/blog/tech/getting-started-with-gans-2-colorful-mnist/

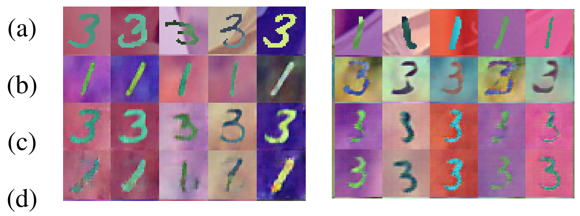

Fig. 3: Translation between classes 3 and 1 on the color MNIST dataset considering different loss functions. (a) Input images, (b) Color Histogram distance (c), DSSIM loss, λ_{DSSIM}=20, (d) DSSIM loss λ_{DSSIM}=5.

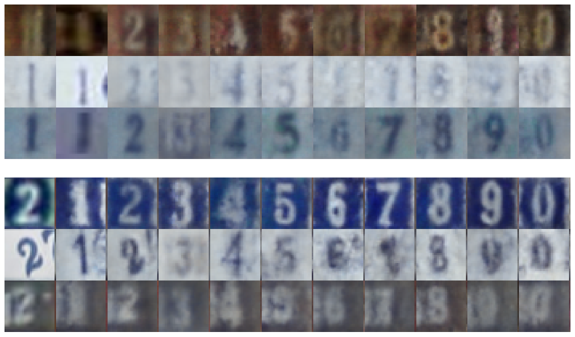

Fig. 4: Translation of classes 1 and 2 (first column) to target classes 0-9 (other columns) from the SVHN dataset .

The images are scaled to 32x32 to fit in our network. The VAE learns a 128-dimensional latent space representation of the numbers. We train the network with both the *color histogram distance* and the *DSSIM* as loss functions. The first metric does not produce desirable results. The color style is not transferred adequately. Therefore we continue using only the DSSIM loss metric. We vary the λ_{sim}, which indicates the importance of the similarity loss. We consider values equal to 5 and 20. The best results are obtained using the value λ_{sim}=5. For a value λ_{sim}=20, the network is not able to produce a realistic shape. See Fig. 3 for a qualitative comparison of different loss functions.

4.2. Translating numbers in real images

After obtaining successful results with a synthetic dataset, we now use a dataset containing real images. This experiment aims at showing the effectiveness of our method in a realistic setting. This second experiment is conducted on the SVHN dataset [31]. The images have input size 32x32x3. There are around 73K training images, split up into ten classes of numbers. We use the same network structure as used with the MNIST dataset. We translate the classes 1 and 2 (source domains) to all other classes from 0 to 9 (target domains). Qualitative examples of this translation can be seen in Fig. 4.

In order to quantify the performance of the translation process, we run a classifier on the translated images from the SVHN test set. This classifier is trained on the SVHN dataset and determines for each image to which digit class it belongs. We call the translation successful if the classifier assigns the translated image to the number of the target domain.

Table 1: Accuracy of SVHN classifier on images from: (a) the original test set, (b) test set images reconstructed by the VAE, (c) class 1 mapped to other classes, and (d) class 2 mapped to other classes.

target class	(a)	(b)	(c)	(d)
0	96.39	85.09	65.95	75.68
1	97.94	98.47	97.71	92.50
2	95.61	95.64	92.74	95.81
3	92.68	95.91	93.29	95.40
4	96.23	96.04	93.80	91.97
5	94.80	89.77	79.58	84.62
6	95.70	88.67	74.25	81.49
7	94.06	94.95	84.37	89.47
8	93.07	86.57	73.41	82.28
9	94.98	89.34	75.09	85.06
Avg.	**95.46**	**93.47**	**86.51**	**89.27**

The performance of the classifier can be found in Table 1. The original classes, after encoding by the VAE, are correctly classified in 93% of the cases. The images from class 1 and class 2 translated to all other classes, can be correctly classified in respectively 86.51% and 89.27% of the cases. Our translation is thus successful, but not yet optimal. We have not fully trained all the translation networks until convergence. All the mappings have been trained between 30K and 100K iterations. An interesting observation is that the class 0 scores remarkably worse than all other classes after the transformation: only 66% and 76% of the images are correctly classified. This low accuracy can partly be explained by the sub-par performance of the VAE for class 0, only achieving a classification accuracy of 85%. This is a stark contrast with the VAEs for classes 1, 2 and 3, which score as good as the original test set on the classification. The images translated to these classes also score almost as good as the original test set. By training some of the VAEs and most of the translation networks further, the classification accuracy gap between the original test set and the translated set can most probably be decreased. Following this, we ran experiments performing translations across images of dogs and cats. Early results were blurry, which might be caused by the higher variability on these images due to changes in pose and scale. Morever, this might be the reason why [26] uses a mask to guide the translation process.

5. CONCLUSION

We use an architecture combining VAEs and GANs to perform unsupervised shape translation across images. We learn to change shapes in images, while preserving style information by learning separate latent space representations and learning a mapping between these two latent spaces. The translation model is currently applicable in simple datasets. Further experimentation needs to be conducted before it is useful in a broader context.

Acknowledgments: This work is supported by a NVIDIA GPU grant, the KUL C1 projet Macchina and the FWO project SfS.

6. REFERENCES

[1] P. Isola, J. Zhu, T. Zhou, and A. Efros, "Image-to-image translation with conditional adversarial networks," *CVPR*, 2017.

[2] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual losses for real-time style transfer and super-resolution," in *ECCV*, 2016.

[3] C. Ledig, L. Theis, F. Huszar, J. Caballero, A. P. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi, "Photo-realistic single image super-resolution using a generative adversarial network," in *CVPR*, 2017.

[4] X. Wang and A. Gupta, "Generative image modeling using style and structure adversarial networks," in *ECCV*, 2016.

[5] L. A. Gatys, A. S. Ecker, and M. Bethge, "A neural algorithm of artistic style," *arXiv:1508.06576*, 2015.

[6] J. Zhu, T. Park, P. Isola, and A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networkss," in *ICCV*, 2017.

[7] M. Liu, T. Breuel, and J. Kautz, "Unsupervised image-to-image translation networks," in *NIPS*, 2017.

[8] Y. Taigman, A. Polyak, and L. Wolf, "Unsupervised cross-domain image generation," in *ICLR*, 2017.

[9] D. P Kingma and M. Welling, "Auto-Encoding Variational Bayes," in *arXiv:1312.6114*, 2013.

[10] Z. Wang, E. P. Simoncelli, and A. C. Bovik, "Multiscale structural similarity for image quality assessment," in *Asilomar Conf. on Signals, Systems Computers*, 2003.

[11] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *NIPS*, 2014.

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

[13] E. L. Denton, S. Chintala, A. Szlam, and R. Fergus, "Deep generative image models using a laplacian pyramid of adversarial networks," in *NIPS*, 2015.

[14] M. Mathieu, C. Couprie, and Y. LeCun, "Deep multi-scale video prediction beyond mean square error," *arXiv:1511.05440*, 2015.

[15] J. Wu, C. Zhang, T. Xue, W. T. Freeman, and J. B. Tenenbaum, "Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling," in *NIPS*, 2016.

[16] S. E. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee, "Generative adversarial text to image synthesis," *arXiv:1605.05396*, 2016.

[17] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel, "Infogan: Interpretable representation learning by information maximizing generative adversarial nets," in *NIPS*, 2016.

[18] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *arXiv:1411.1784*, 2014.

[19] T. Wang, M. Liu, J. Zhu, A. Tao, J. Kautz, and B. Catanzaro, "High-resolution image synthesis and semantic manipulation with conditional gans," in *CVPR*, 2018.

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

[21] K. Bousmalis, N. Silberman, D. Dohan, D. Erhan, and D. Krishnan, "Unsupervised pixel-level domain adaptation with generative adversarial networks,," in *CVPR*, 2017.

[22] S. Benaim and L. Wolf, "One-sided unsupervised domain mapping," in *arXiv:1706.00826*, 2017.

[23] F. Luan, S. Paris, E. Shechtman, and K. Bala, "Deep photo style transfer," in *arXiv:1703.07511*, 2017.

[24] D. Yoo, N. Kim, S. Park, A. S. Paek, and I. Kweon, "Pixel-level domain transfer," in *ECCV*, 2016.

[25] B. Wang, H. Zheng, X. Liang, Y. Chen, L. Lin, and M. Yang, "Toward characteristic-preserving image-based virtual try-on network," in *ECCV*, 2018.

[26] K. Wang, L. Ma, J. Oramas, L. Van Gool, and T. Tuytelaars, "Integrated unpaired appearance-preserving shape translation across domains," *arXiv:1812.02134*, 2018.

[27] A. Boesen Lindbo Larsen, S. Kaae Sønderby, and O. Winther, "Autoencoding beyond pixels using a learned similarity metric," in *ICML*, 2016.

[28] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *ICLR*, 2015.

[29] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016.

[30] D. Ulyanov, A. Vedaldi, and V. S. Lempitsky, "Instance normalization: The missing ingredient for fast stylization," *arXiv:1607.08022*, 2016.

[31] A. Coates A. Bissacco B. Wu A. Y. Ng Y. Netzer, T. Wang, "Reading digits in natural images with unsupervised feature learning," .