Unsupervised Cross-Domain Image Generation

Ondřej Pleticha
Faculty of Information Technology
Czech Technical University in Prague

June 25, 2019

1 Introduction

When choosing the topic of my seminar project, I wanted me to learn something new and to make the topic interesting, both for me and for others. So I focused on an article titled Unsupervised Cross-Domain Image Generation[1], which was presented in 2017 at the ICLR conference. The article focuses on unsupervised image transfer between domains. The example can be the source domain of the face portraits and the target domain of the animated faces, as shown 1. In my semester project I was dealing with an experiment on this model.

Figure 1: Example of domain transfer

2 Theoretical background

Given a set s of unlabeled samples in a source domain S sampled i.i.d according to some distribution D_S, a set of samples in the target domain $t \subset T$ sampled i.i.d from distribution D_T. We wish to learn a function $G : S \to T$, $G = g \circ f$. We have to minimise two losses L_D and $L_G = L_{GANG} + \alpha L_{CONST} + \beta L_{TID} + \gamma L_{TV}$, for some weights α, β, γ, where

$$L_{\mathrm{D}} = -\mathbb{E}_{x\in\mathbf{s}} \log D_1(g(f(x))) - \mathbb{E}_{x\in\mathbf{t}} \log D_2(g(f(x))) - \mathbb{E}_{x\in\mathbf{t}} \log D_3(x)$$

$$L_{\mathrm{GANG}} = -\mathbb{E}_{x\in\mathbf{s}} \log D_3(g(f(x))) - \mathbb{E}_{x\in\mathbf{t}} \log D_3(g(f(x)))$$

$$L_{\mathrm{CONST}} = \sum_{x\in\mathbf{s}} d(f(x), f(g(f(x))))$$

$$L_{\mathrm{TID}} = \sum_{x\in\mathbf{t}} d_2(x, G(x))$$

and where D is a ternary classification function from the domain T to 1, 2, 3, and $D_i(x)$ is the probability it assigns to class i = 1, 2, 3 for an input sample x, and d_2 is a distance function in T. During optimization, L_G is minimized over g and L_D is minimized over D. The last loss, L_{TV} is an anisotropic total variation loss, which is added in order to slightly smooth the resulting image. The loss is defined on the generated image z = $[z_{ij}]$ = G(x) as

$$L_{TV}(z) = \sum_{i,j} \left((z_{i,j+1} - z_{ij})^2 + (z_{i+1,j} - z_{ij})^2 \right)^{\frac{B}{2}},$$

where we employ B = 1. Mean squared error was used for both d and d_2.[1]

So as you can see on image 2 L_{GAND} and L_{GANG} are loss functions for the GAN discriminator and generator. The L_{ID} is used to keep the identity of the target domain in function G, and L_{CONST} guarantees similarity of images after G function.

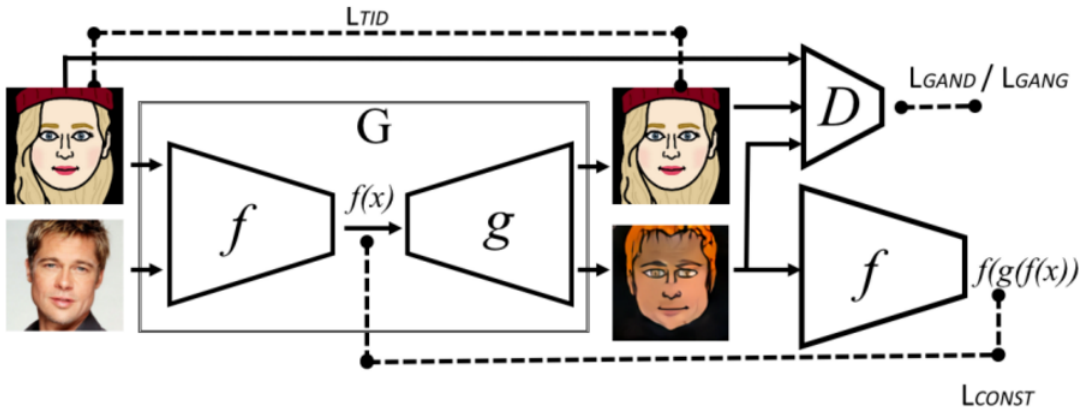

Figure 2: The Domain Transfer Network. Losses are drawn with dashed lines, input/output with solid lines. After training, the forward model G is used for the sample transfer.

3 Experiments

In my seminar work I focused on two tasks. First I wanted to repeat the experiment, which was mentioned in the article, namely the conversion from the source domain of the house numbers (SVHN dataset) to the target domain of handwritten numbers (MNIST handwritten digit dataset).
And as a second task, I tried the model on a new, more difficult dataset, between clothes photos (DeepFashion dataset) and Fashion MNIST dataset.
Unfortunately the source code of the article is not published, so I used a foreign implementation [2]. Code is written in Python 3 and I used Google Colab for training the model.

3.1 SVHN → MNIST handwritten digits

For model learning, I used 73,000 images (32x32, RGB) from the source domain and 60,000 images (32x32, Grayscale) from the target domain.

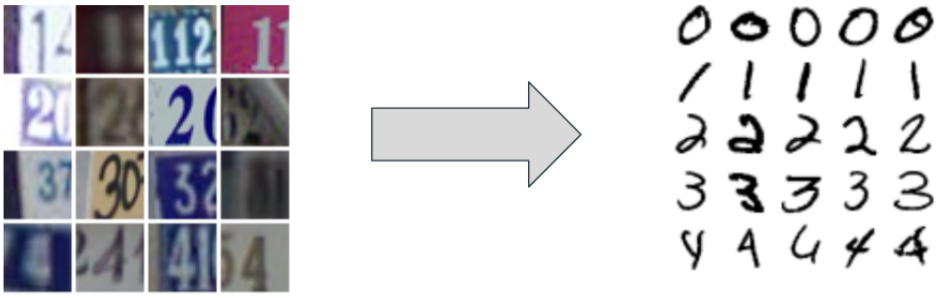

Figure 3: SVHN → MNIST handwritten digits. Sample of source and target domain.

I did the training on a different number of iterations to determine the optimal number when the model is not undertrained or overtrained.
The results are shown in image 4, where the number indicates the number of iterations at batch-size of 100. Generally, the results are good, but it is difficult to determine the optimal number of iterations. For the highest number of iterations, the model is always overtrained, but for a specific case it is not possible to tell exactly how many iterations gave best results.

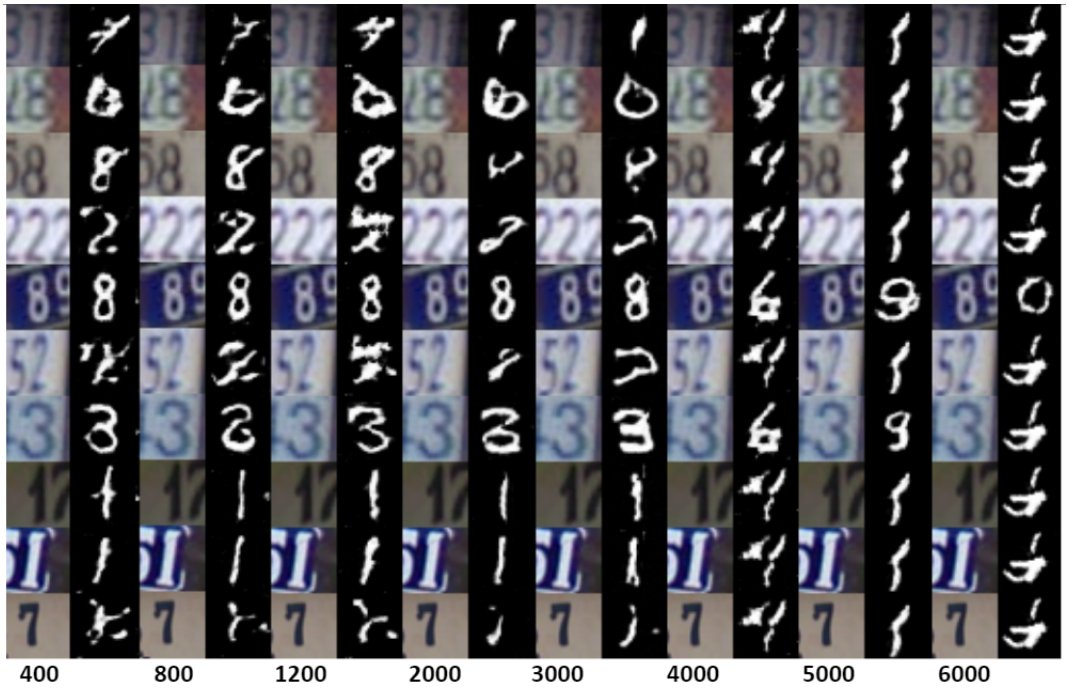

Figure 4: SVHN → MNIST handwritten digits - results

Although it is unsupervised learning, but to improve model performance, it is necessary to pre-train function f (convolution neural network) on the source domain. I did an experiment (image 5) where I didn't use this pre-training and the results were wrong, as I expected.

Figure 5: SVHN → MNIST handwritten digits - without pre-training

3.2 DeepFashion → Fashion MNIST

For model learning, I used 150,000 images (64x64, RGB) from the source domain and 60,000 images (64x64, Grayscale) from the target domain (image 6). Unfortunately, the quality of the source dataset was not very good for this case. Pictures do not have the same background, clothing is not centred, sometimes the whole figure is taken, sometimes only half, etc.

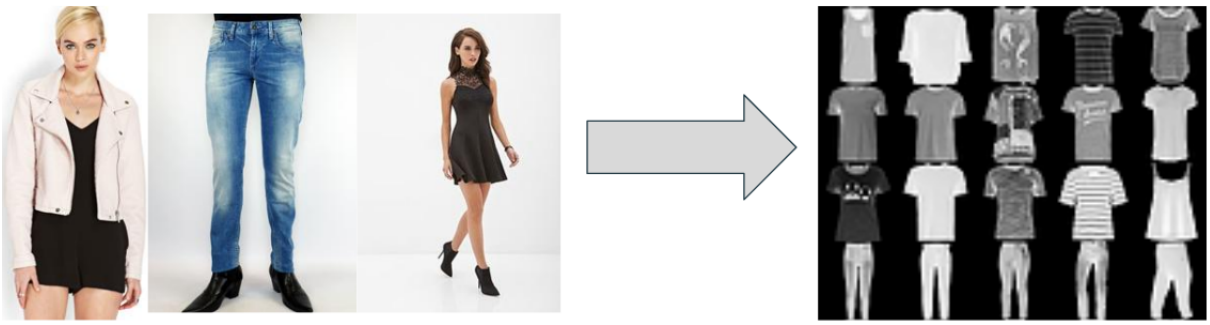

Figure 6: DeepFashion → Fashion MNIST. Sample of source and target domain.

First, I used the entire source dataset for learning that contained the following types of clothes:

- T-shirt – 38,000
- Trousers – 3,000
- Pullover – 4,000
- Dress – 71,000
- Coat/Jacket – 13,000

As you can see in the image 7 with the results, it can be seen that due to class imbalances and poor data quality, the results are insufficient. The model is not so accurate as to recognize individual types of clothing on a human (e.g. jacket and sweater), so the result is always similar. At least image grayscale retains the original color.

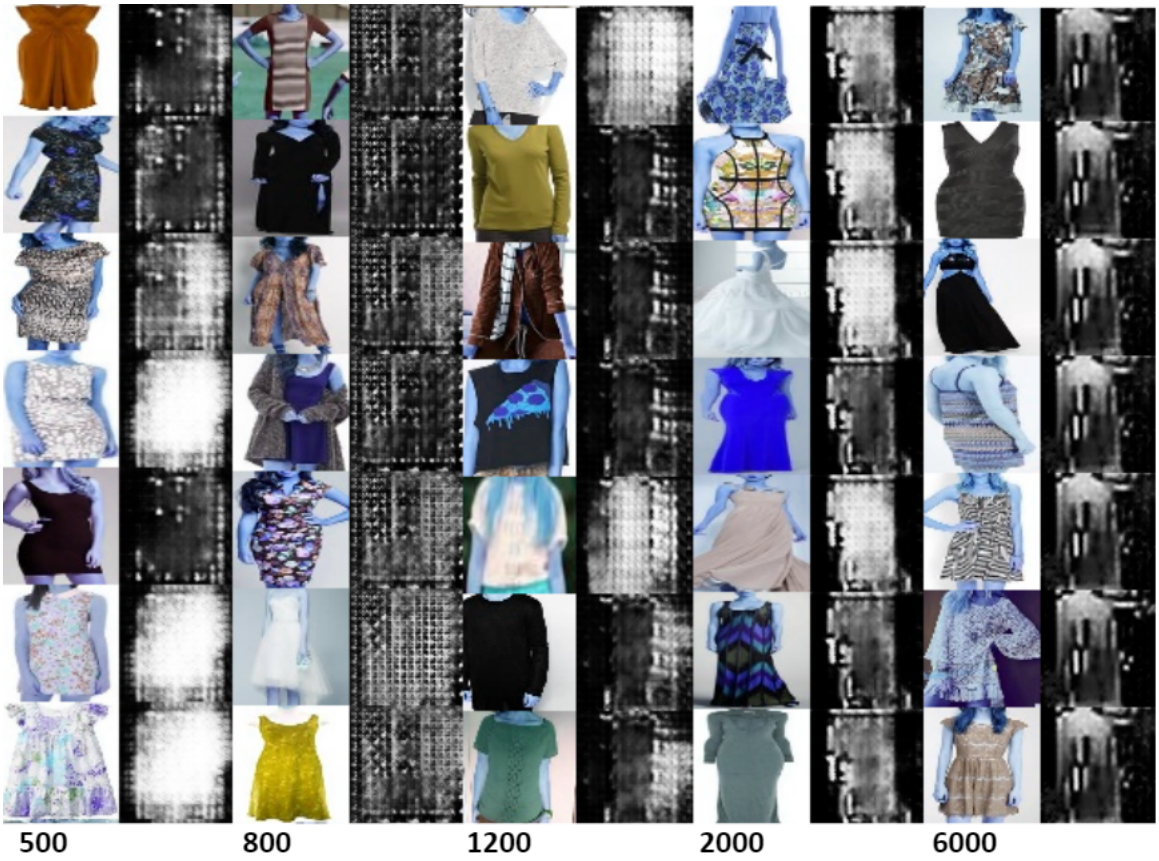

Figure 7: DeepFashion → Fashion MNIST - results

To test the case when the classes are balanced, I have selected only trousers(3,000) and pullovers(4,000) from the dataset and trained the model on this data. Due to the very low number of samples, I did not expect good results, but in image 8 the results show small differences between classes.

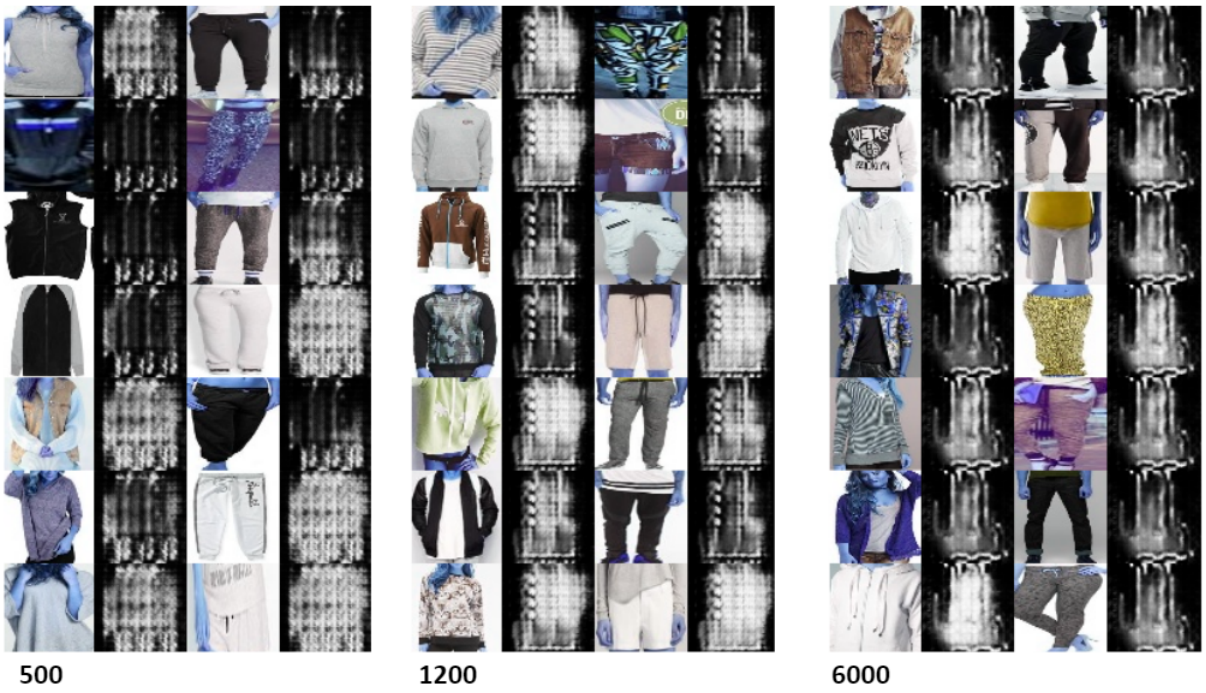

Figure 8: DeepFashion → Fashion MNIST - only two classes

4 Conclusion

In conclusion, I will summarize the main findings of the experiments. Although it is unsupervised learning, it is necessary to have the source dataset labeled or, if possible, use a pre-trained convolutional neural network. It is also a common problem to find the optimal number of iterations for learning so that all samples get the best results. It is a good idea to choose the appropriate datasets so that each class is significantly different (e.g. number 5 and 7). And, of course, it is optimal to have a very large dataset with the best possible data quality. This model comes to me very interesting and in the future I want to test its possibilities on non-image data.

References

1. TAIGMAN, Yaniv; POLYAK, Adam; WOLF, Lior. Unsupervised Cross-Domain Image Generation. *CoRR*. 2016, vol. abs/1611.02200. Available also from: `http://arxiv.org/abs/1611.02200`.
2. CHOI, Yunjey. *Domain Transfer Network* [`https://github.com/yunjey/domain-transfer-network`]. GitHub, 2017.