ID2223 Project
Generating manga characters from celebrity faces using Generative Adversarial Networks (GANs).

Group 4
Alexandra Josefsson, Leïla Kany
ajose@kth.se, lkany@kth.se

January 12, 2020

1 Introduction

The task which we decided to solve was generating manga characters from pictures of real people using Generative Adversarial Networks (GANs). The project was inspired by the work done by Royer et al. [1]. Royer et al. used GANs to generate cartoon faces from celebrity faces. The general task done was translating semantics from one image domain to another. Similarly, we aimed to translate faces from a real image domain to a manga domain. The desired outcome can be seen in Figure 1.

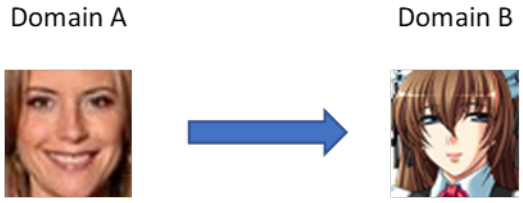

Figure 1: Idea of translating face from real to manga domain

2 Datasets

Images of faces from the two domains were needed. The faces needed to be of a roughly similar nature in terms of semantics. In this case, we opted for having faces at roughly the same closeness, see the example in Figure 1.

Faces from the real domain were taken from the kin dataset consisting of 3066 images [2] [3]. The dataset could be found at `http://www.kinfacew.com/datasets.html`.

Faces from the manga domain were found at `https://www.kaggle.com/soumikrakshit/anime-faces`.

3 Method

An XGAN network [1] was used to solve the task.

The general architecture of the XGAN network can be seen in Figure 2, where e1 and d1 are the encoder and decoder for the first domain (here real domain), and e2 and d2 are the encoder and decoder for the second domain (here manga domain). The XGAN consists of two autoencoder networks making up the generator part. There is one autoencoder generating fake "real" faces, and there is one autoencoder generating fake manga faces. The two autoencoders share some layers, called the shared embedding. This shared embedding in combination with custom losses help in preserving semantic feature-level information, i.e. the type of information that may be shared in the both domains, e.g. eye color, hair color, smiling/not smiling, etc.

The two autoencoders have the same structure with some shared layers. The structure of one of the autoencoders can be seen in Figure 4. The discriminator structure can be seen in Figure 5.

The XGAN generator part is trained according to the following loss function:

$$L_{XGAN_{Gen}} = L_{rec} + w_d L_{dann} + w_s L_{sem} + w_g L_{gan} \tag{1}$$

Some of these losses are common for GANs, such as the reconstruction loss L_{rec}, which is a sum of how well the two autoencoders reconstruct the inputs. The domain-adversarial loss between the two domains, L_{dann}, encourages the embeddings learned by the two encoders to lie in the same subspace. Semantic consistency loss, L_{sem}, is a feature-level semantic consistency loss. This loss is focused on preserving the embedding when translating between the two domains in both directions. Finally, GAN objective, L_{GAN}, is a loss where the generator is meant to produce as realistic samples as possible in the manga domain (in our case) from real faces. The discriminator is also involved in this loss, where it tries to distinguish this generated fake manga image from an original manga image.

The discriminator is trained in parallel to the generator using the GAN objective, L_{GAN}.

A discriminator is trained in parallel to generator.

Additional details about project:

- Use adam optimizer to optimize generator loss (where beta=0.9)

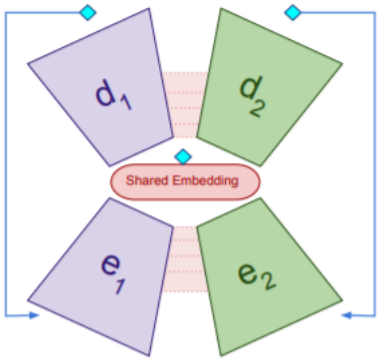

Figure 2: High-level view of XGAN

4 Results

After training for 5 epochs, an example output can be seen in Figure 3.

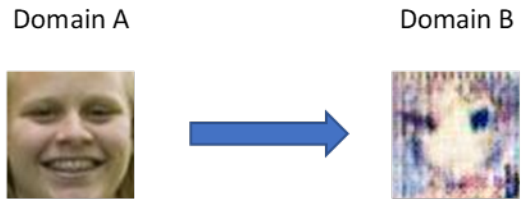

Figure 3: Translation of face from real to manga domain

5 How to run code

Training was done on Google Cloud Platform using a Compute Engine. The model must first be built by running model.py. Then training and test data for the two domains should be placed in train1, train2, test1, test2 folders (the 1 and 2 refers to real face domain or manga domain).

Changes are still being made to the code. The code is uploaded at its current state.

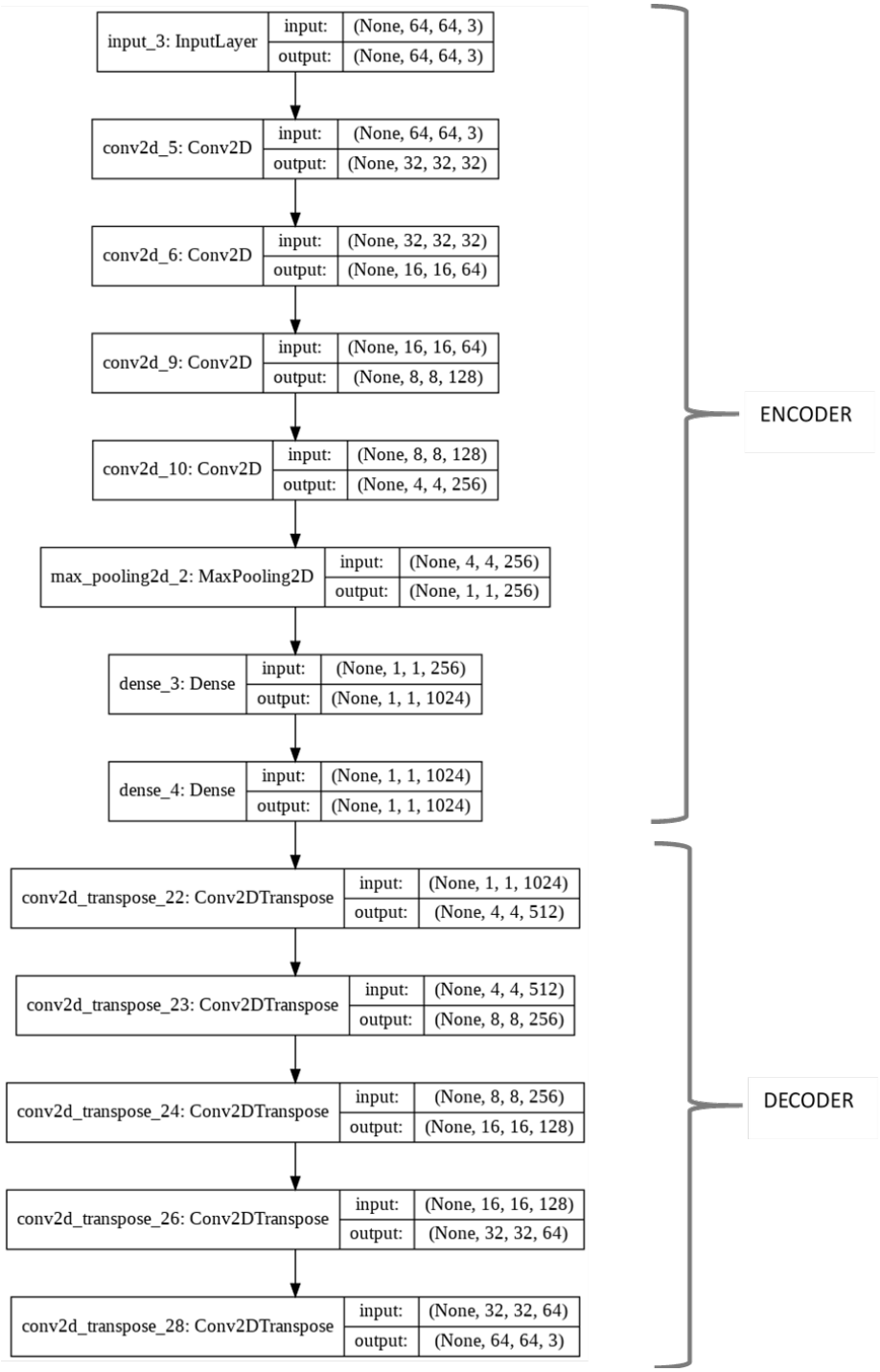

Figure 4: Architecture of one autoencoder

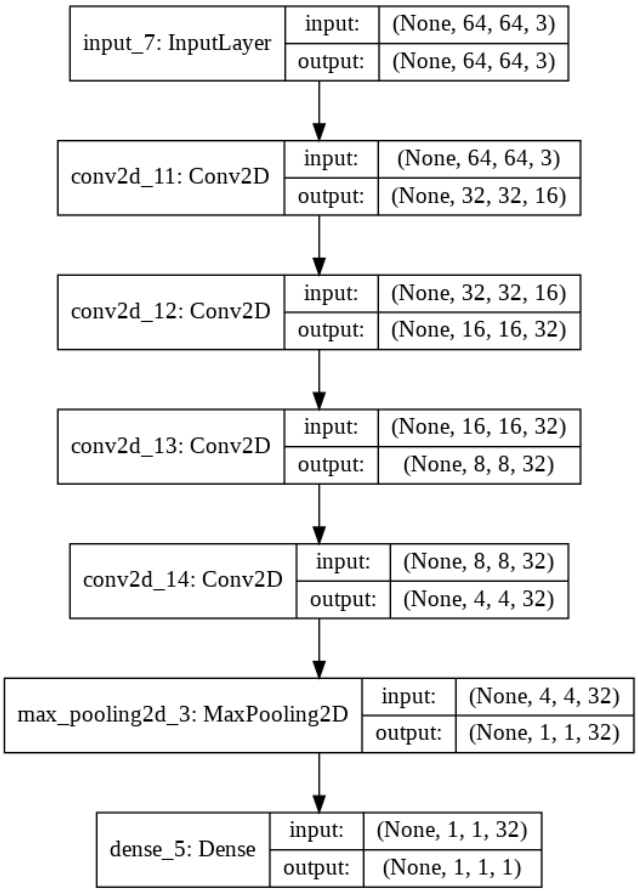

Figure 5: Architecture of the discriminator

A Appendix

References

[1] Amélie Royer et al. "Xgan: Unsupervised image-to-image translation for many-to-many mappings". In: *arXiv preprint arXiv:1711.05139* (2017).

[2] Jiwen Lu et al. "Neighborhood repulsed metric learning for kinship verification". In: *IEEE transactions on pattern analysis and machine intelligence* 36.2 (2013), pp. 331–345.

[3] Haibin Yan, Xiuzhuang Zhou, and Yongxin Ge. "Neighborhood repulsed correlation metric learning for kinship verification". In: *2015 Visual Communications and Image Processing (VCIP)*. IEEE. 2015, pp. 1–4.