Progress Report 1 – Disentanglement by Cross Training Naive Approach (7. May 2020)

Name	**Student ID**	**E-Mail**
Li Nguyen	934644485	li.nguyen@tum.de
Alexander Koenig	918254061	awc.koenig@tum.de

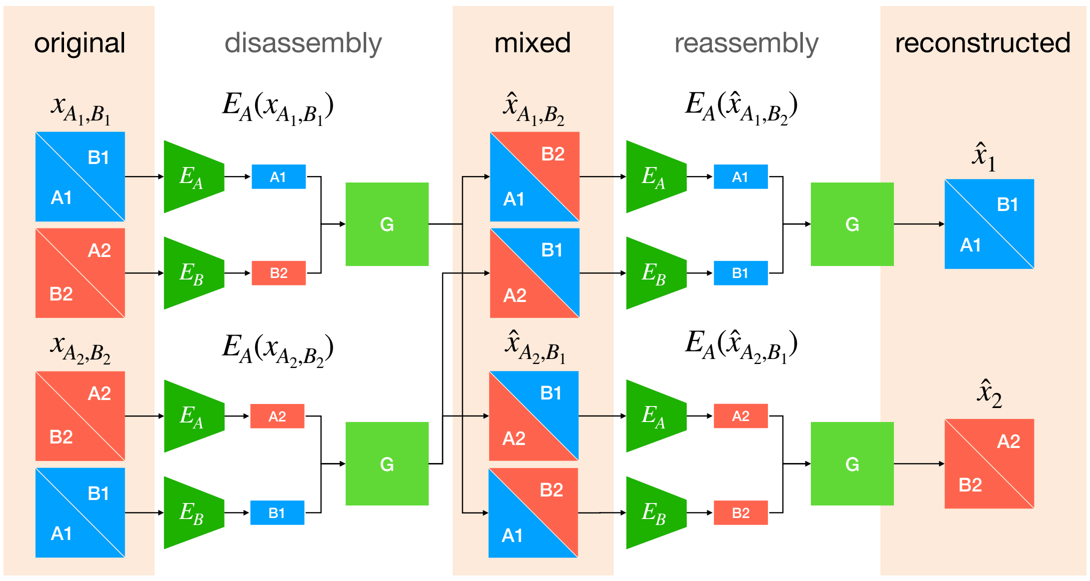


Figure 1: Architecture overview

1 Notation

- Original images
  $x_1 = x_{A_1,B_1}$ and $x_2 = x_{A_2,B_2}$

- Mixed images
  $\hat{x}_{A_1,B_2} = G\big(E_A(x_1), E_B(x_2)\big)$ and
  $\hat{x}_{A_2,B_1} = G\big(E_A(x_2), E_B(x_1)\big)$

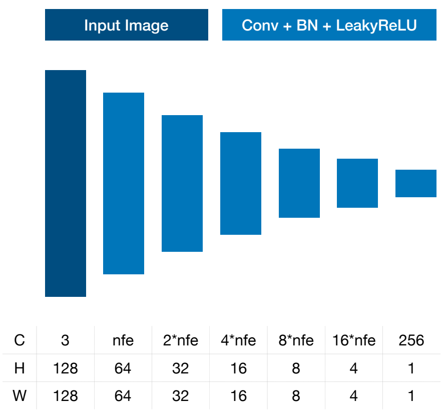
- Reconstructed images
  $\hat{x}_1 = G\big(E_A(\hat{x}_{A_1,B_2}), E_B(\hat{x}_{A_2,B_1})\big)$ and
  $\hat{x}_2 = G\big(E_A(\hat{x}_{A_2,B_1}), E_B(\hat{x}_{A_1,B_2})\big)$

2 Architectures

Encoder – We use the same architecture for both encoder types (figure 3). The architecture of the encoders is largely based on the paper Mask Based Unsupervised Content Transfer [4]. The encoders consist of 6 [Conv, BatchNorm, LeakyReLU] blocks that convert the input image of size 3*128*128 to a 1D latent code of size 256.

Generator – The generator was taken from a PyTorch implementation [5] of the LORD paper [1] (figure 2). The only modification we did was to use codes of equal size (256). In the original implementation Code A's size was 256 and Code B's size was 128. In the implementation [5] Code A is referred to as class code and Code B is called content code. We desire in our implementation that Code A encodes the class (i.e. identity) and Code B encodes the rest of the image.

The generator first passes Code B through several dense layers with a LeakyReLU activation and then reshapes the output to 256*8*8. Then four [Upsampling, Conv, LeakyReLU, AdaIn] blocks scale the height and width up to the desired 128*128. Each AdaIn normalization layer [2] gets two modulation parameters from the dense layers that process the information in the class code A (2 parameters for 256 channels = 512 neurons in FC layers). Finally, two layers and activations aggregate the information into 3 feature maps.



C	3	nfe	2*nfe	4*nfe	8*nfe	16*nfe	256
H	128	64	32	16	8	4	1
W	128	64	32	16	8	4	1

Figure 3: Encoder Architecture

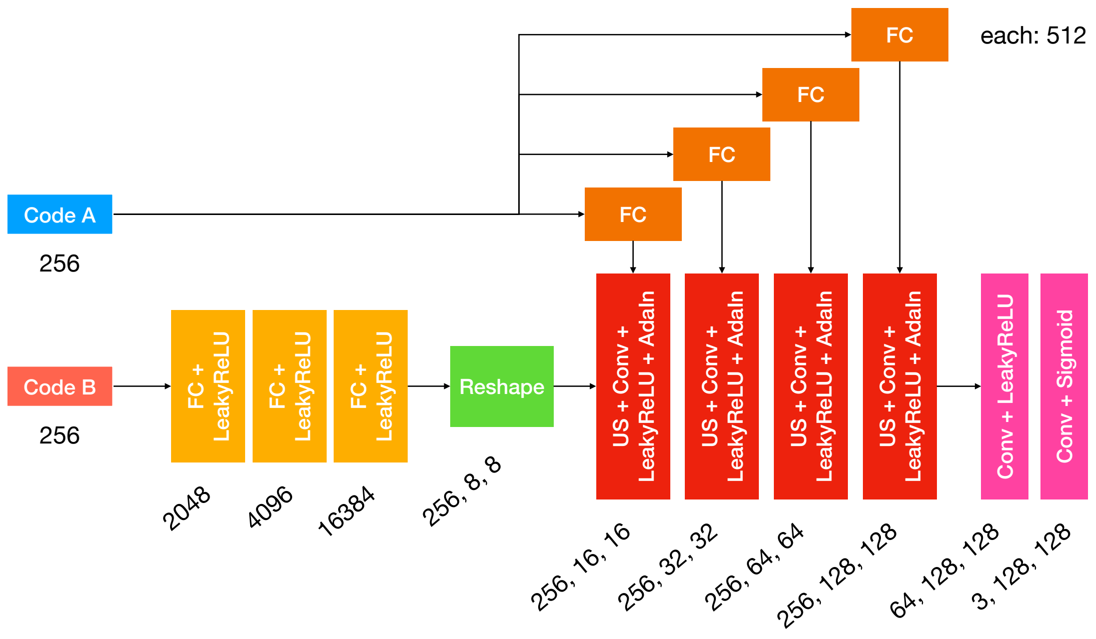


Figure 2: Generator architecture ([1] and [5])

3 Losses

We use a reconstruction loss for both images in combination with a cycle consistency loss for both encoders. The **reconstruction loss** is motivated by the fact that the original images should be as similar as possible to the reconstructed images. The **cycle consistency losses** should guide the network to produce the same low-dimensional embeddings in the disassembly stage as in the reassembly stage (e.g. the feature code A_1 should be the same from original image x_{A_1,B_1} as from the mixed image $\hat{x}_{A_1,B_2}$). We define the reconstruction loss in equation 1 and one cycle consistency loss for each encoder (equations 2 and 3).

$$L_{rec} = \sum_{i=1}^{n} \|\hat{x}_i - x_i\| \tag{1}$$

$$L_{cyc,E_A} = \sum_{i=1}^{n-1} \left\|E_A(x_{A_i,B_i}) - E_A(\hat{x}_{A_i,B_{i+1}})\right\|^2 + \left\|E_A(x_{A_{i+1},B_{i+1}}) - E_A(\hat{x}_{A_{i+1},B_i})\right\|^2 \tag{2}$$

$$L_{cyc,E_B} = \sum_{i=1}^{n-1} \left\|E_B(x_{A_i,B_i}) - E_B(\hat{x}_{A_{i+1},B_i})\right\|^2 + \left\|E_B(x_{A_{i+1},B_{i+1}}) - E_B(\hat{x}_{A_1,B_{i+1}})\right\|^2 \tag{3}$$

$$L = L_{rec} + \gamma \cdot (L_{cyc,E_A} + L_{cyc,E_B}) \tag{4}$$

Finally, we define the overall loss in equation 4. It is a linear combination of the above losses. We introduce the weighting factor γ. Both cycle consistency losses are scaled by this parameter.

As a first milestone we wanted to determine the best performing weight γ. We trained the model on the CelebA dataset [3] with different parameters $\gamma \in [0.01, 0.4, 1, 1.5, 2]$ for 20 epochs with a batch size of 32 (batch size of 64 did not fit on the memory of two GPUs).

4 Results

Figures 5 and 6 show the training and validation losses for all runs. To compare the visual results of the different approaches we introduce a plot with the following layout.

x_{A_1,B_1}	x_{A_2,B_2}
$\hat{x}_{A_1,B_2}$	$\hat{x}_{A_2,B_1}$
$\hat{x}_1$	$\hat{x}_2$

Figure 4: Layout for plots

From the qualitative results in figures 7, 8, 9, 10 and 11 it becomes clear that a value of $\gamma = 0.4$ performs best. The probably **best result** can be seen on the far right of figure 8 (in red box). The smile from image x_2 seems to be successfully transferred to the mixed image $\hat{x}_{A_1,B_2}$. Similarly, the closed mouth of the person in x_1 is transferred to the image $\hat{x}_{A_2,B_1}$. Finally, the reconstructed images $\hat{x}_1$ and $\hat{x}_2$ get their original facial expression back. A very small γ (0.01) seems to result in a noisy mixed image. For strong γ parameters (1.5 and 2.0) the mixed image looks very similar to the reconstructed one.

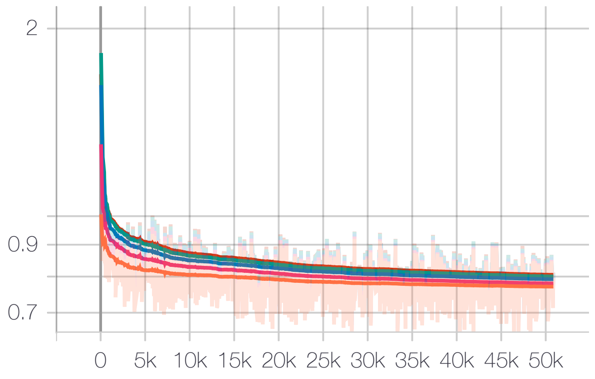


Figure 5: Training loss (orange: $\gamma = 0.01$, pink: $\gamma = 0.4$, blue: $\gamma = 1.0$, green: $\gamma = 1.5$, red $= \gamma = 2.0$)

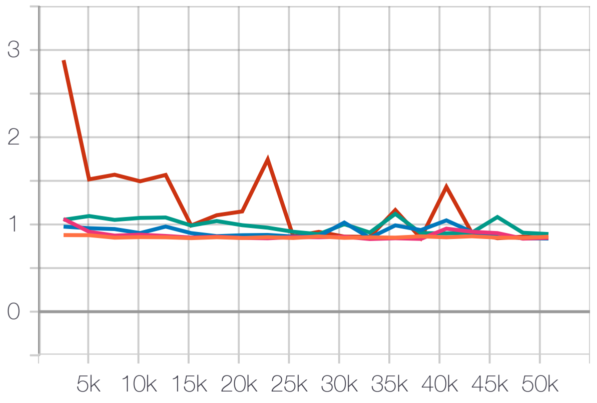


Figure 6: Validation loss (orange: $\gamma = 0.01$, pink: $\gamma = 0.4$, blue: $\gamma = 1.0$, green: $\gamma = 1.5$, red $= \gamma = 2.0$)

5 Future Work

Next Steps

1. Run more tests on γ values close to 0.4
2. Implement GAN Loss

Open Questions

1. Should we use a different architecture for E_A than for E_B?
2. Are the sizes of the latent codes suitable? Should the class code A_i be smaller than the code for the rest of the image B_i?
3. Why are the generated images so pale/grayish?
4. Is it a good idea to give both cycle consistency losses the same weight (i.e. γ)?
5. Does the GAN loss help to generate more detailed images?
6. Does it make sense to introduce a weight for the GAN loss as well?
7. Would it improve performance if we increase this weight successively? Specifically, we thought about giving more weight to the discriminator initially to make the discriminator become good in its predictions.

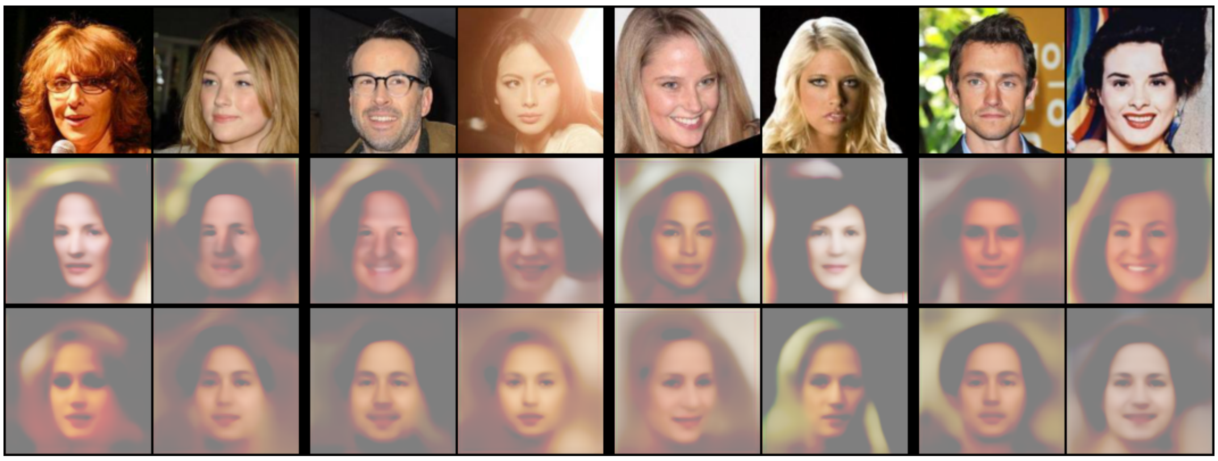


Figure 7: Plots from validation set with $\gamma = 0.01$

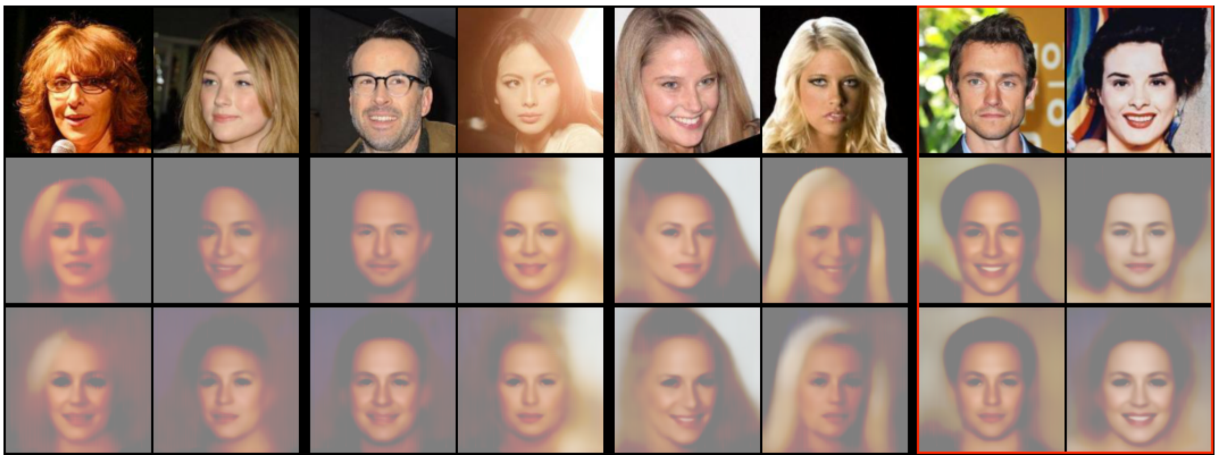


Figure 8: Plots from validation set with $\gamma = 0.4$

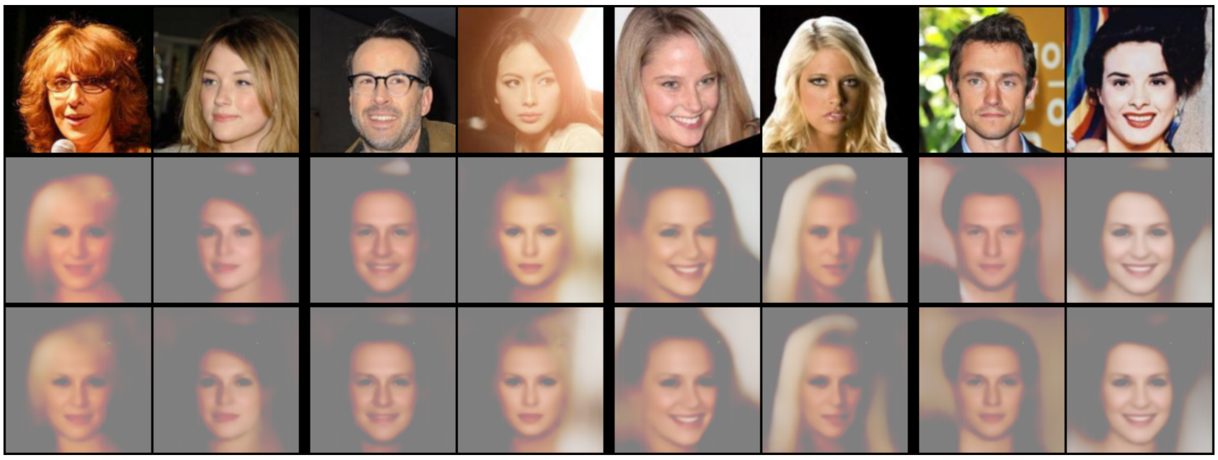


Figure 9: Plots from validation set with $\gamma = 1.0$

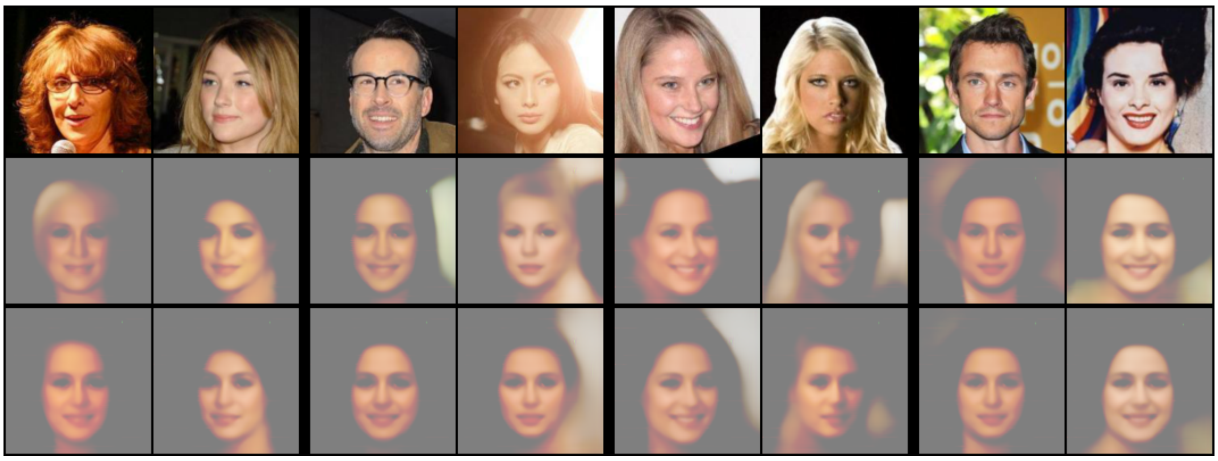


Figure 10: Plots from validation set with $\gamma = 1.5$

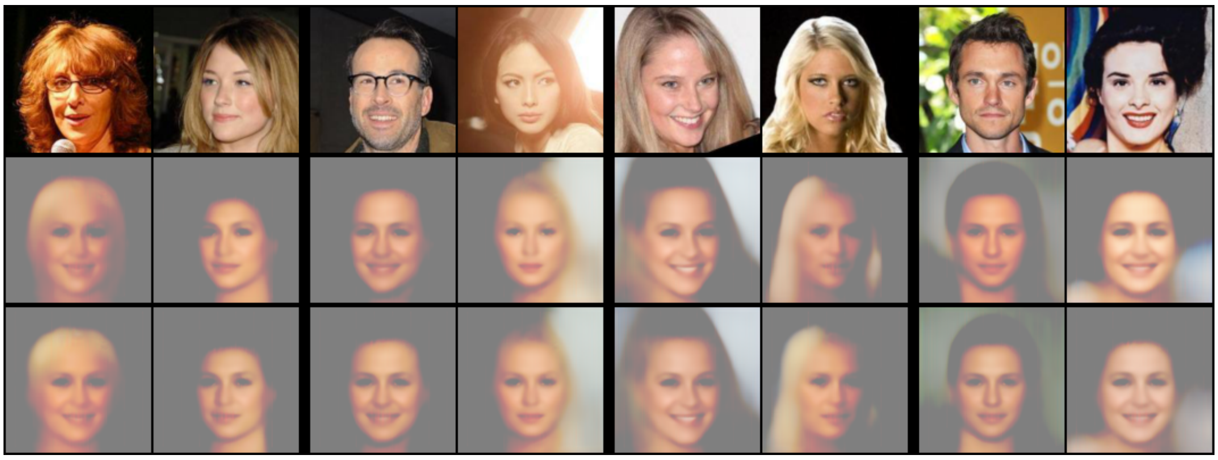


Figure 11: Plots from validation set with $\gamma = 2.0$

References

[1] Aviv Gabbay and Yedid Hoshen. "Demystifying Inter-Class Disentanglement". In: *International Conference on Learning Representations (ICLR)*. 2020.

[2] Xun Huang and Serge Belongie. *Arbitrary Style Transfer in Real-time with Adaptive Instance Normalization.* 2017. arXiv: `1703.06868 [cs.CV]`.

[3] Ziwei Liu et al. "Deep Learning Face Attributes in the Wild". In: *Proceedings of International Conference on Computer Vision (ICCV).* Dec. 2015.

[4] Ron Mokady et al. *Mask Based Unsupervised Content Transfer.* 2019. arXiv: `1906.06558 [cs.CV]`.

[5] *PyTorch implementation of LORD Paper.* `https://github.com/avivga/lord-pytorch`. 2020.