Progress Report 2: Disentanglement by Cross Training
Using FUNIT Implementation

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

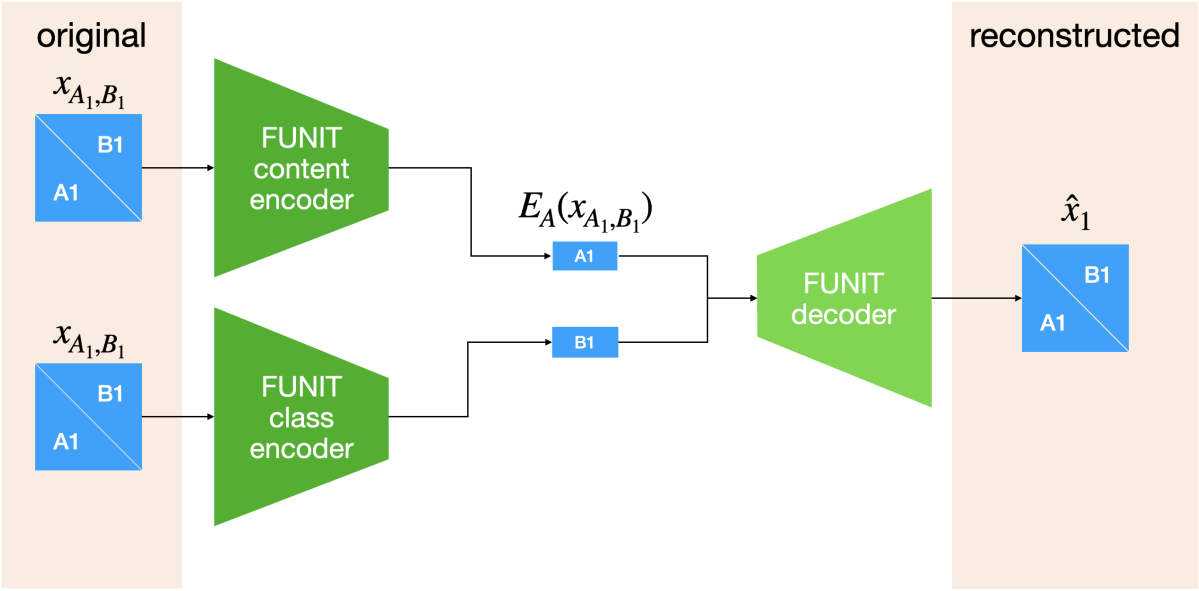


Figure 1: Step 1: Simple Reconstruction using FUNIT encoder and decoder architecture and VGG perceptual loss

Next Steps We agreed on the following next steps via E-Mail:

Step 1: Reconstruction using FUNIT Encoders + Decoder
We take the encoder and decoder architecture from FUNIT [2] and train it only with a reconstruction loss to try to do a simple reconstruction. This means that we feed the same image into the network's class and content encoder and try to reconstruct it without mixing. As a loss function we use the VGG loss as Ron suggested. An overview can be seen in Figure 1, while the VGG loss is shown in Figure 4 and is by Johnson et al. [1].

Step 2: Integrate FUNIT into G2G Architecture
If good visual results are obtained from the simple reconstruction we move on and integrate the FUNIT encoders and their decoder into our own architecture as shown in Figure 2. Here we add the cycle consistency losses on top of the VGG reconstruction loss of Step 1. Note that the loss is calculated over the original and reconstructed image, and not the original and mixed.

Step 3: Adjustment of GAN Discriminator and Loss
To increase photorealism and visual quality of the mixed images, we want to integrate adversarial training by comparing FUNIT's GAN loss to CycleGAN's loss.

1 Notation

To recall our 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
  $\hat{x}_2 = G\big(E_A(\hat{x}_{A_2,B_1}), E_B(\hat{x}_{A_1,B_2})\big)$

Progress Report 2: Disentanglement by Cross Training
Using FUNIT Implementation

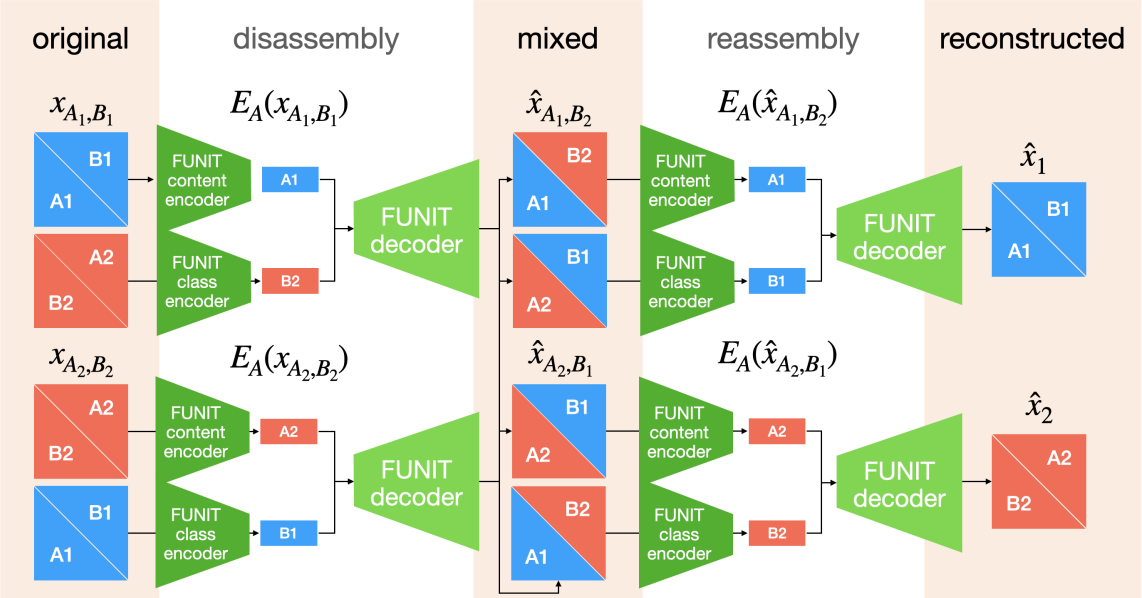


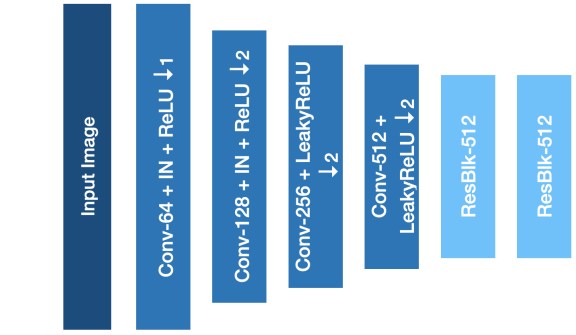
Figure 2: FUNIT encoders and decoder integrated in our G2G architecture

2 Simple Reconstruction using FUNIT

2.1 Architecture

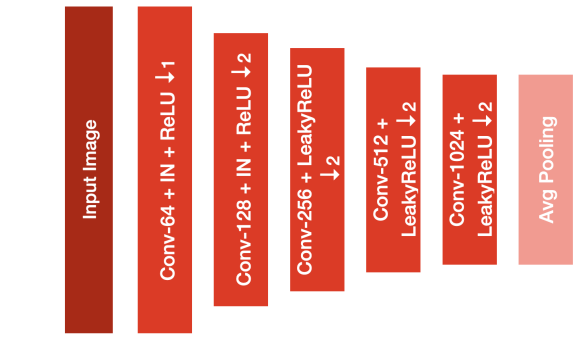
Content Encoder - We adopt the class and content encoder architecture of FUNIT. The content encoder encodes the separate feature while the class encoder encodes the "rest of the image". The content encoder consists of 4 [Conv, InstanceNorm, ReLU] layers followed by two ResBlocks while the first layer does not downsample the height and width, and the following three do downsample by half of their size. Note that nfe stands for the number of feature maps and is 64. It takes a $3 \times 128 \times 128$ image as an input and produces a latent content code of size $512 \times 16 \times 16$, compare Figure 5.

Class Encoder - The class encoder is made from 5 [Conv, InstanceNorm, ReLU] Layers and is followed by AvgPooling. The first layer does not downsample the height and width similarly to the content encoder, while the following four layers do. The AvgPooling layer also downsamples the height and width again by half of their size resulting in a latent class code of dimension $1024 \times 1 \times 1$, compare Figure 6. Again, $nfe = 64$. The class code is more lightweight as it is fed into the decoder via AdaIN layers later.



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

Figure 5: Content Encoder of FUNIT



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

Figure 6: Class Encoder of FUNIT

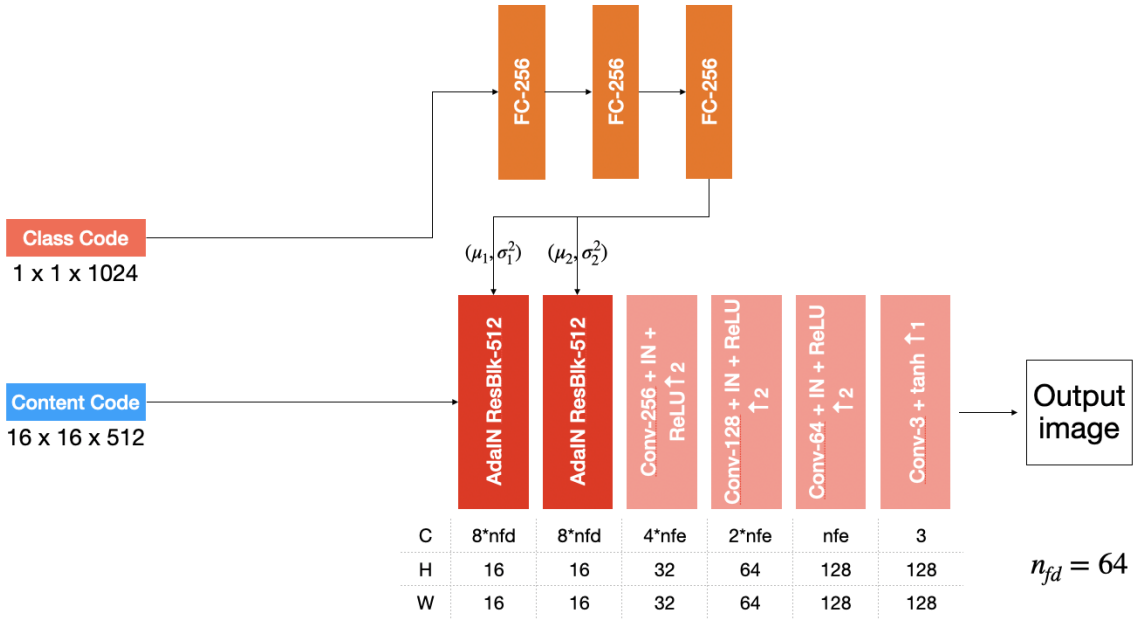


Figure 3: FUNIT decoder

Decoder - The decoder takes to latent codes. The content code is taken directly as an input into the network, while the class code is fed through three fully connected layers, and then via AdaIn layers into the first two ResBlocks of the decoder. These ResBlocks are then followed by 3 [Conv, InstanceNorm, ReLU] layers which upsample the feature map dimensions by two. The final [Conv, tanh] layer then constructs the output image of the original dimension $3 \times 128 \times 128$, see Figure 3.

2.2 Losses

In this very first step we use only the VGG perceptual loss by using a pre-trained VGG16 model. The input image and the reconstructed image are both fed into the VGG and the activations at the same layer are compared using mean squared error. We chose the activations of relu layer 2_2 to compare the features, see Figure 4 and Equation 1.

$$L_{rec} = \sum_{i=1}^{n} \left\| vgg(\hat{x}_i)^{relu2\_2} - vgg(x_i)^{relu2\_2} \right\| \quad (1)$$

2.3 Results

Figure 8 shows the visual result after training for only 2 epochs. The reconstructed image looks for human observers nearly the same aside from lighting differences.

3 Integrate FUNIT into G2G Architecture

As the first results have high visual quality we moved onto the next step, which is integrating the FUNIT encoders and their decoder into our G2G architecture as seen in Figure 2.

3.1 Architecture and Implementation Details

We take replace our original encoder E_A and E_B by FUNIT's content and class encoder, our generator has the architecture of FUNIT's decoder. Then, two different images are fed into the encoders respectively and mixed, the mixed images are again fed into the encoders and reconstructed as seen in Figure 2.

3.2 Losses

We adapt the VGG perceptual loss to fit our G2G architecture, this means that we feed the vgg with the original image x_1 and the reconstructed one $\hat{x_1}$. Also, we now add the cycle consistency losses 2 and 3, shown below. These are the cycle consistency losses that we used before in our naive approach, but now with FUNIT architecture.

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

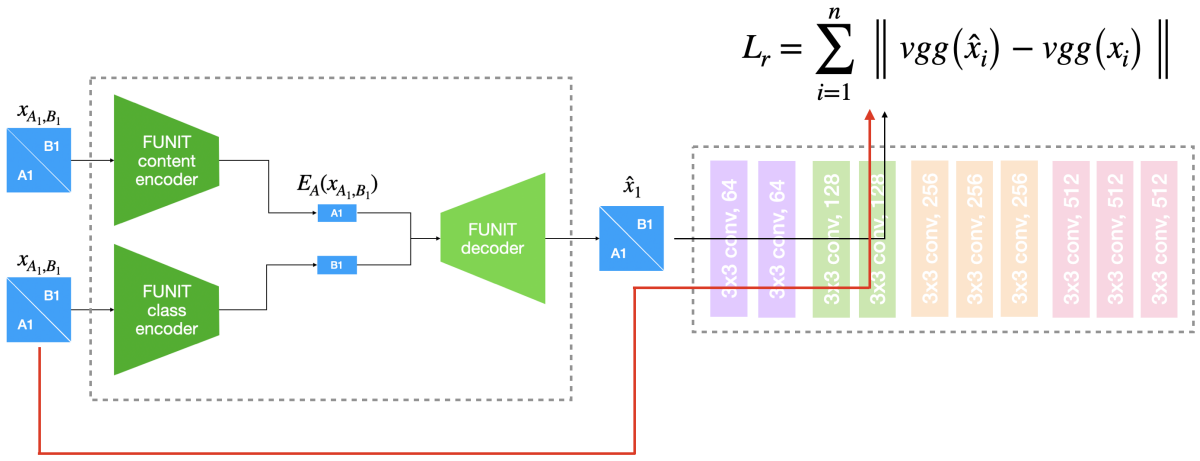


Figure 4: VGG perceptual loss using pre-trained VGG16

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

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

3.3 Results

For the results of this second step we used the plot with the same layout as last time, i.e. the top row shows the images x_1 and x_2, the second row, the mixed images and the last row the reconstructed images $\hat{x_1}$ and $\hat{x_2}$. The visual results can be seen in Figure 9 for $\gamma = 0.01$. The grid search is currently still running, so we are expecting more results for $\gamma = 0.4$, $\gamma = 1.3$, $\gamma = 10$ and $\gamma = 100$.

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 7: Layout for Plots

4 Future Work

Next Steps

1. Evaluate the upcoming results from different γ values. As suggested by Ron, we also include extreme gamma values, such as 0.01, 10 and 100.

2. Implement GAN Loss using FUNIT and CycleGAN architecture and loss

Open Questions

1. The quality of the mixed images degraded during the epochs. We assume that due to the currently small weight of 0.01 on the cycle consistency losses, the reconstruction loss gets much stronger and minimizes its loss at the expense of the cycle consistency loss. We have fixed the VGG reconstruction loss weight at 1.0. We assume that tuning the γ hyperparameter as well as introducing the GAN loss would improve this visual quality.

2. Until the current state of the grid search, i.e. $\gamma = 0.01$ we don't see any disentanglement yet. Is this due to the setting of the hyperparameter or are there any other possible reasons why disentanglement does not yet work?

3. Would it be a good idea to use FUNIT's loss instead of the VGG perceptual loss? That is:

$$L_{recon} = \frac{1}{n} \sum_{i=1}^{n} \|predict - target\| \quad (5)$$

4. Where do the shiny highlights in the mixed images come from? They do not seem to

always minimize the reconstruction (at least visually), since they are not always located where the light hits the face or head but are located where there's background.

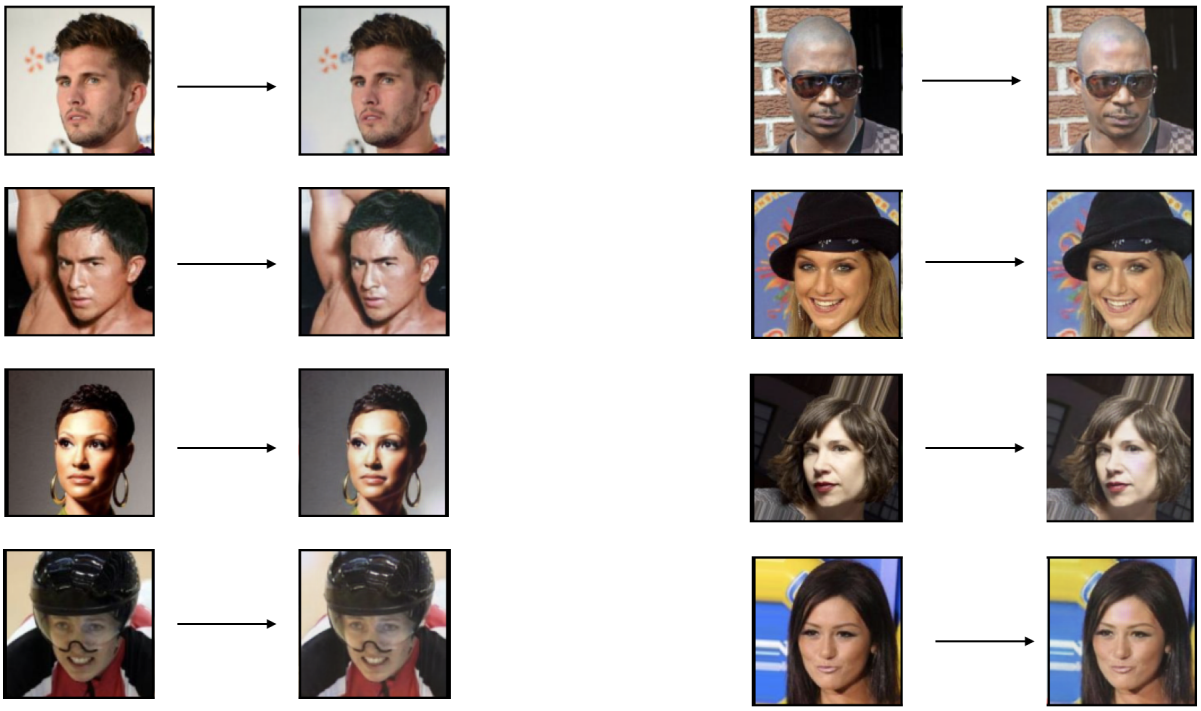


Figure 8: Plots from validation set for Step 1

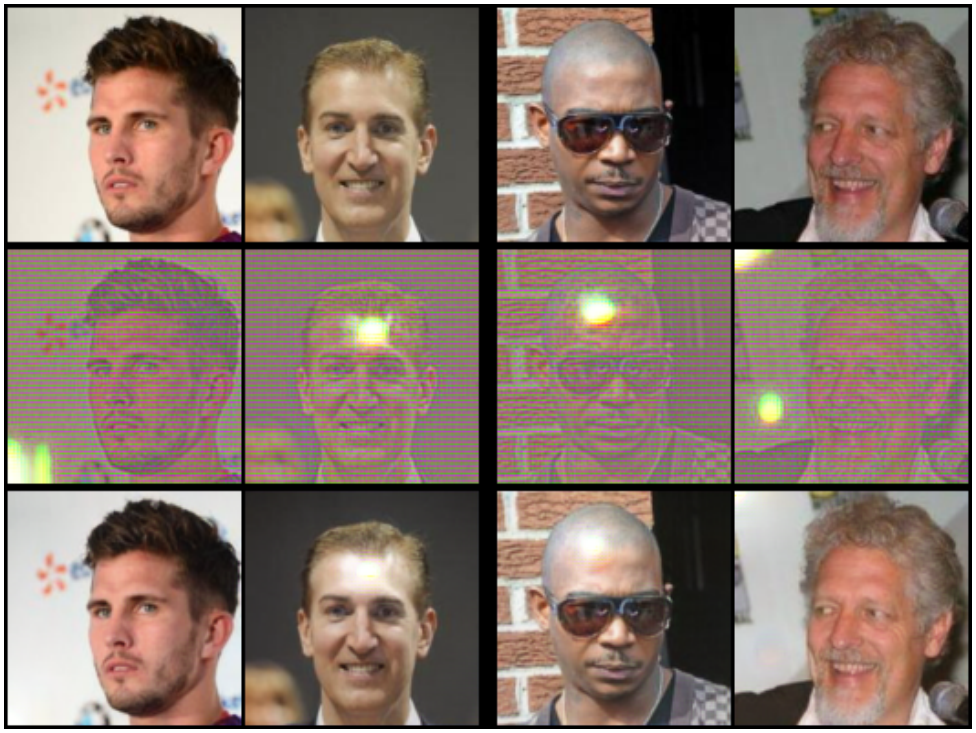


Figure 9: Plots from validation set with $\gamma = 0.01$

Progress Report 2: Disentanglement by Cross Training
Using FUNIT Implementation

References

[1] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. "Perceptual losses for real-time style transfer and super-resolution". In: *European conference on computer vision*. Springer. 2016, pp. 694–711.

[2] Ming-Yu Liu et al. "Few-shot unsupervised image-to-image translation". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2019, pp. 10551–10560.