# Supplementary material for paper submission #6103
# *Detecting Overfitting of Deep Generative Networks* via *Latent Recovery*

Ryan Webster, Julien Rabin, Loic Simon and Frederic Jurie
Normandie Univ., UNICAEN, ENSICAEN, CNRS Caen, France
GREYC - Groupe de Recherche en Informatique, Image, Automatique et Instrumentation de Caen
ryan.webster@unicaen.fr

## Abstract

*This document gives additional details and experiments regarding:*

- *study of local overfitting (Section 1),*
- *visual results of recovery with various objective loss functions (Section 2),*
- *failure cases and success rate of recovery (Section 3),*
- *convergence study of the latent recovery optimization (Section 4).*

## 1. Local vs Global overfitting

While GANs geneartors appear to not overfit the training set on the entire image, one may wonder if they do however overfit training image patches. To investigate this, we take $\phi$ of Eq. $\text{NN}_{\mathcal{G}}$ to be a masking operator on eye and mouth regions of the image. To first verify this optimization is stable (see Section 4, for more information of stability of optimization), we recover eyes PGGAN for a number of random initializations in Fig. 1. Finally, we observe the recovery histograms and KS p-values for patches in Fig. 2.

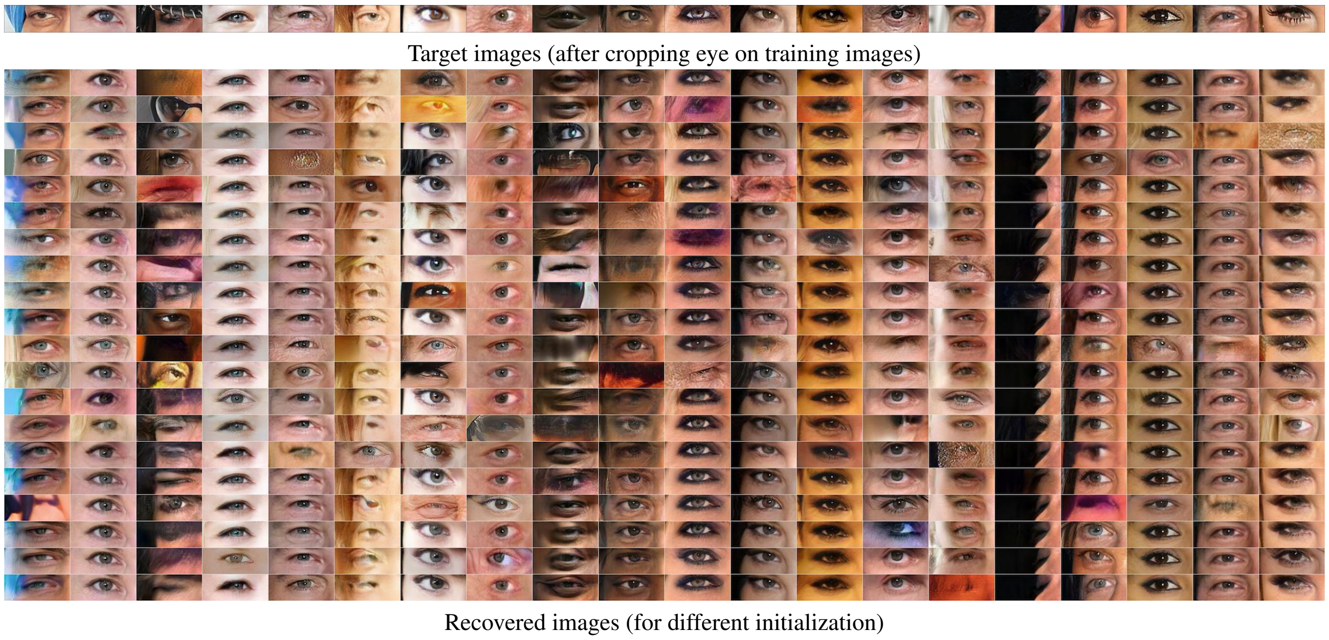
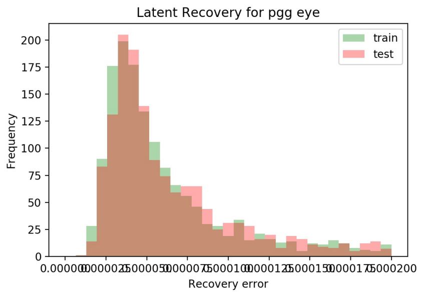
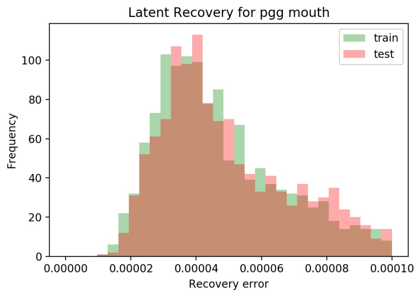

Target images (after cropping eye on training images)
Recovered images (for different initialization)

Figure 1: *Visual results on training images recovery with a* ***masking operator*** *on right eyes (using LBFGS and Euclidean objective loss, and PGGAN generator). First row: target (real) images $y_i$. First column: initialization. Second column: optimization after 100 iterations. Recovery is more consistent than global optimization.*



(a) *Eye patch recovery error.*



(b) *Mouth patch recovery error.*

Figure 2: *Recover of eye patches (left) and mouth patches (right). The KS p-values for these two graphs are $p = 0.0545$ and $p = 0.6918$ respectively from left to right (see paper).*

## 2. Comparison with Other Loss Functions

We visually compare in Fig. 3 the simple Euclidean loss used in this paper for analyzing overfitting (*i.e.* $\phi = \mathrm{Id}$ in Eq. $\mathrm{NN}_G$) with other operators:

- $\phi =$ pooling by a factor of 32 (as used in applications for super-resolution);
- $\phi =$ various convolutional layers of the VGG-19 (*i.e.* the *perceptual loss* previously mentioned in the paper).

While the perceptual loss has been shown to be effective for many synthesis tasks, it appears to hinder optimization in the case when interacting with a high quality generator $G$.

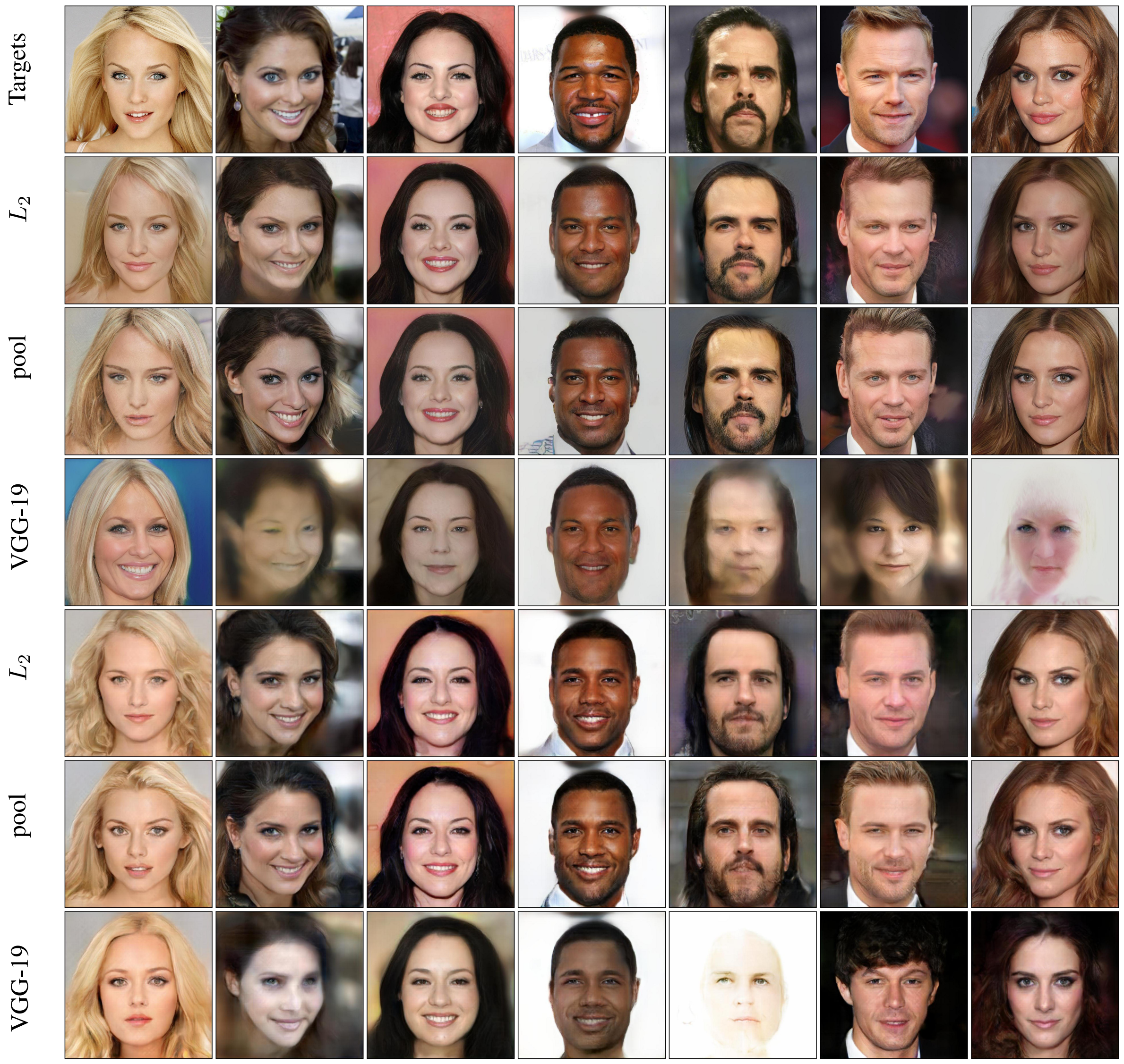
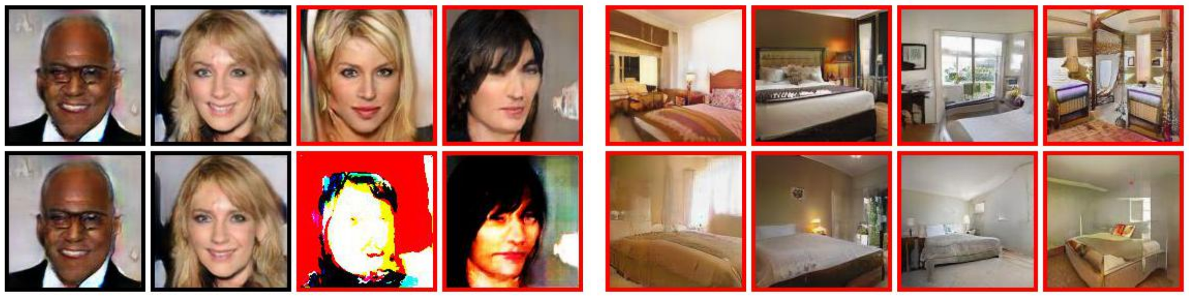


Figure 3: *Using other loss functions for image recovery. The first row is target images from CelebA-HQ, the next three rows are recovery from PGGAN network and the final three are from MESCH generator.*

## 3. Optimization Failures

We noted that most networks had the ability to exactly recover generated images. This is shown in Fig. 3, with failure cases highlighted in red. Interestingly, some networks were not able to recover their generated images at all, for example Fig. 4b was a PGGAN trained on LSUN Bedroom, which did not verbatim recover any image. We think this may suggest a more complex latent space for some networks trained on LSUN, with many local minima to equation $\text{NN}_{\mathcal{G}}$. Because we assert that we are finding the nearest neighbors in the space of generated images, we did not analyze networks which could not recover generated images. It should be noted that some LSUN networks did recover generated images however.



(a) *Generated recovery for MESCH on CelebA-HQ.* (b) *Generated recovery for PGGAN on LSUN Bedroom.*

Figure 4: Recovery failure detection with thresholding. First row generated images and second row is recoveries.

### 3.1. Recovery Success Rate

Disregarding networks which could not recover generated images, some networks had higher failure rates than others. To determine failure cases numerically, we chose a recovery error threshold of $MSE < .1$ to signify a plausible recovery for real images (for generated images a much smaller threshold of $MSE < .025$ can be used). Table 1 summarizes recovery rates for a few networks. The MESCH resnets were notably less consistent than other architectures. To study if these failures were due to bad initialization, we tried simply restarting optimization 10 times per image, and saw the success rate go from 68% to 98% shown in Table 1 as MESCH-10-RESTART. This shows that likely all training and generated images can be recovered decently well with enough restarts.

Table 1: *Success rate for real and generated images using a threshold of MSE¡.1, which corresponds to a plausible recovery. Failures seem to be due to bad initialization as MESCH-10-RESTART simply restarts optimization 10 times per image and has a much higher success rate.*

| | train | test | generated |
|---|---|---|---|
| MESCH | 68% | 67% | 67% |
| MESCH-10-RESTART | 98% | 99% | 96% |
| DC-CONV | 82% | 82% | 100% |
| PGGAN | 97% | 96% | 95% |

## 4. Convergence analysis of latent recovery

In general, optimization was successful and converges nicely for most random initializations. We provide numerical and visual evidence in this section supporting fast and consistent convergence of LBFGS compared to other optimization techniques like SGD or Adam.

### 4.1. Protocol

To demonstrate that the proposed optimization of the latent recovery is stable enough to detect overfitting, the same protocol is repeated in the following experiments. We used the same 20 random latent codes $z_i^*$ to generate images as target for recovery: $y_i = G(z_i*)$ . We also used 20 real images as targets the same as in Section 2 for local recovery. We also initialized the various optimization algorithms with the same 20 random latent codes $z_i$. We plot the median recovery error (MRE) for 100 iterations. This curve (in red) is the median of all MSE curves (whatever the objective function is) and is compared to the 25th and 75th percentile (in blue) of those 400 curves.

### 4.2. Comparison of optimization algorithm

We first show the average behavior in Fig. 6 the chosen optimization algorithm (LBFGS) to demonstrate that it convergences much faster than SGD and Adam. A green dashed line shows the threshold used to detect if the actual nearest neighbor is well enough recovered (MRE $= 0.024$). One can see that only 50 iterations are required in half the case to recover the target image.

### 4.3. Comparison of objective loss functions

In Figure 7 are plotted the MRE (median recovery error) when optimizing various objective functions:

- Euclidean distance ($L_2$) as used throughout the paper,
- Manhattan distance ($L_1$), which is often used as an alternative to the Euclidean distance that is more robust to outliers,
- VGG-based *perceptual loss*.

### 4.4. Convergence with operator $\phi$

Figure 8 demonstrates convergence under various operators $\phi$.

### 4.5. Recovery with other generators

Figure 9 displays median recovery error (MRE) when optimizing with LBFGS and SGD for DCGAN and MESCH generators. Visual results are given for LBFGS in Figures 11 and 12. The MESCH network is more inconsistent, but using 10 random initialization is enough to ensure the recovery of a generated (or real) image with 96% chance.

### 4.6. Convergence on real images

Figure 10 shows highly consistent recover on real images for the PGGAN network.

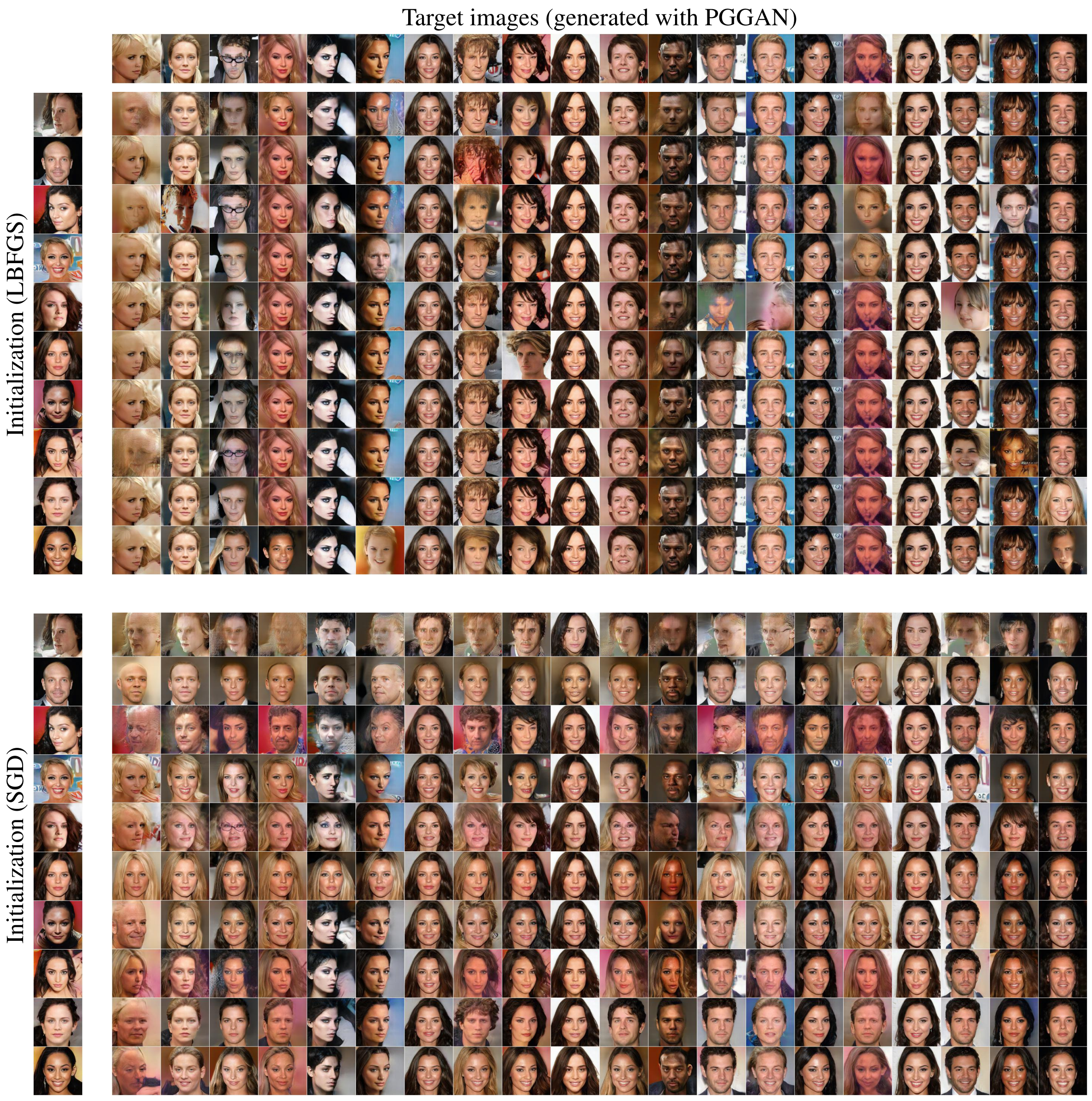
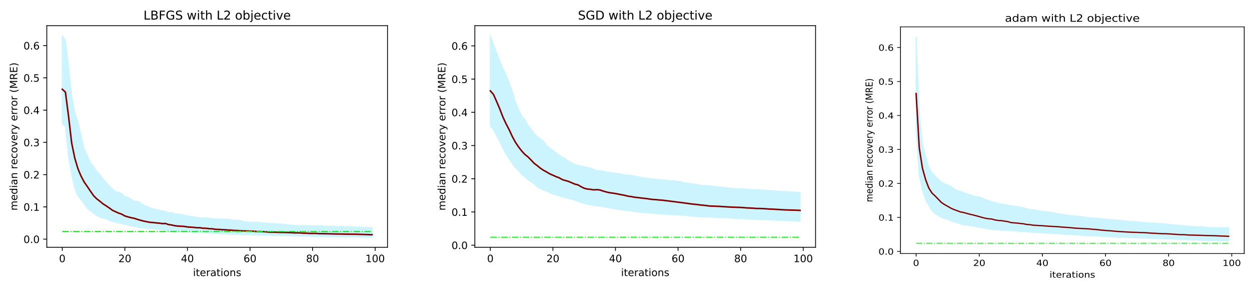
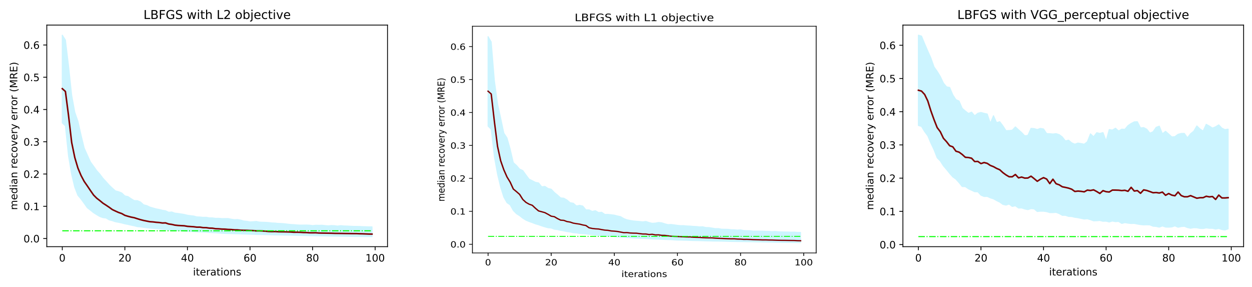


Figure 5: *Visual comparison of recovery with **LBFGS** (top) and **SGD** for the Euclidean loss (see (NNG) optimization problem in the main paper.). First row: target (generated) images $y_i = G(z^{i*})$. First column: initialization ($G(z_i^{(0)})$). Second column: optimization after 100 iterations ($G(z_i^{(100)})$). LBFGS gives much better results than SGD that is much slower to converge, but still needs sometimes some restart (here shown without restarting).*



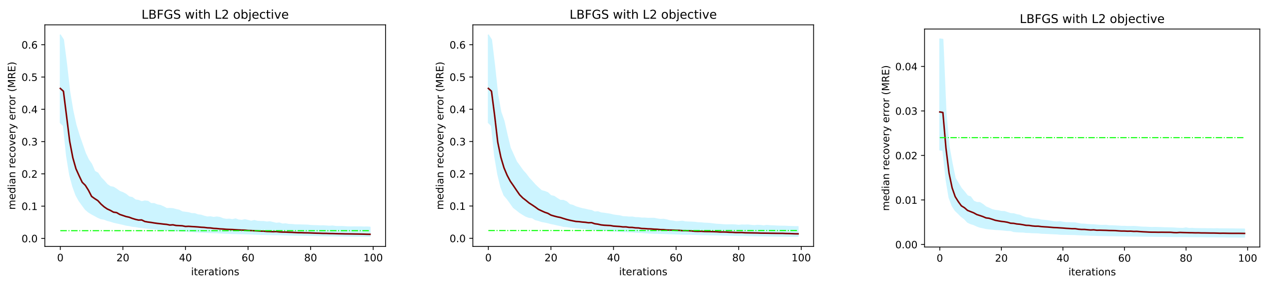
LBFGS SGD Adam

Figure 6: *First row: median recovery error (MRE) curve. Second row: 400 superimposed recovery error curves for 20 images with 20 random initialization. LBFGS (first column) converges faster than SGD or Adam (second and third column respectively).*



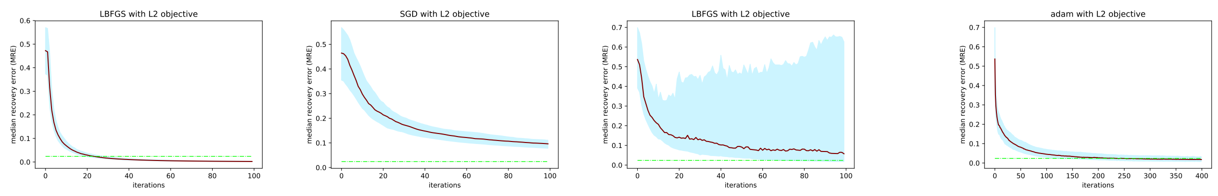
Euclidean distance ($L_2$) Manhattan distance ($L_1$) Perceptual loss (VGG-19)

Figure 7: LBFGS with various objectives for latent recovery for PGGAN.



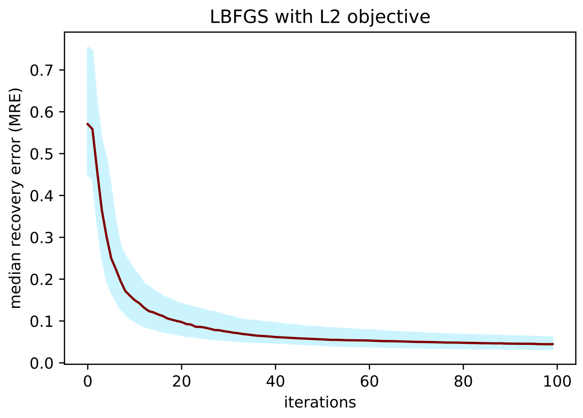
$\phi =$ identity (resolution of 1024) $\phi =$ pooling (resolution of 128) $\phi =$ mask (cropping around mouth area)

Figure 8: Using various $\phi$ operator for applications (super-resolution, inpainting) has no effect on convergence.



DCGAN with LBFGS DCGAN with SGD MESCH with LBFGS MESCH with Adam (400 iterations)

Figure 9: Comparing LBFGS with SGD and Adam algorithms for generated image recovery with a DCGAN and a MESCH network



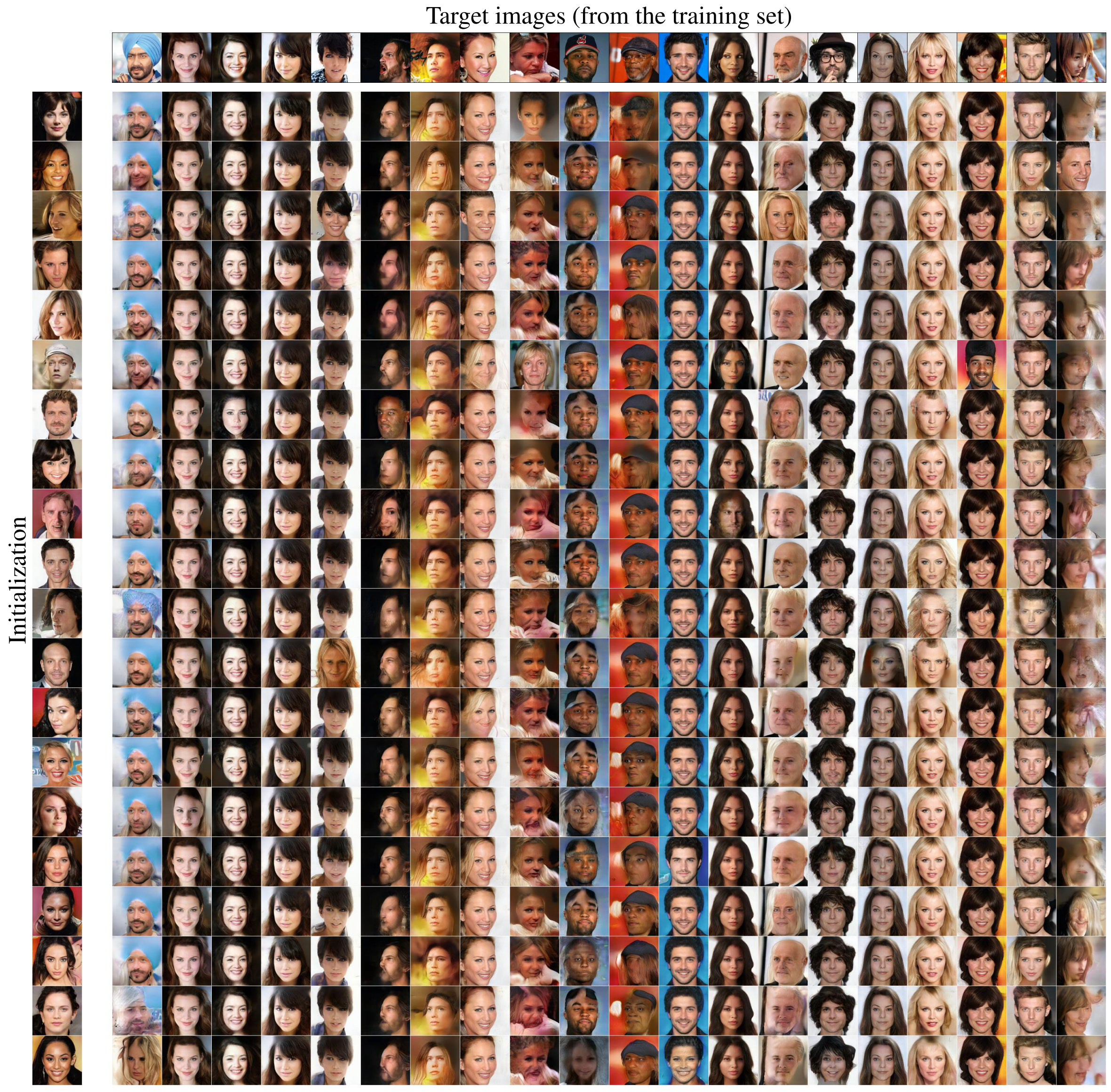


Figure 10: *Visual results on* ***real*** *images recovery (training set from celeba-HQ) with LBFGS and Euclidean objective loss, and PGGAN generator. First row: target (real) images $y_i$. First column: initialization ($G(z_i^{(0)})$). Second column: optimization after 100 iterations ($G(z_i^{(100)})$).*

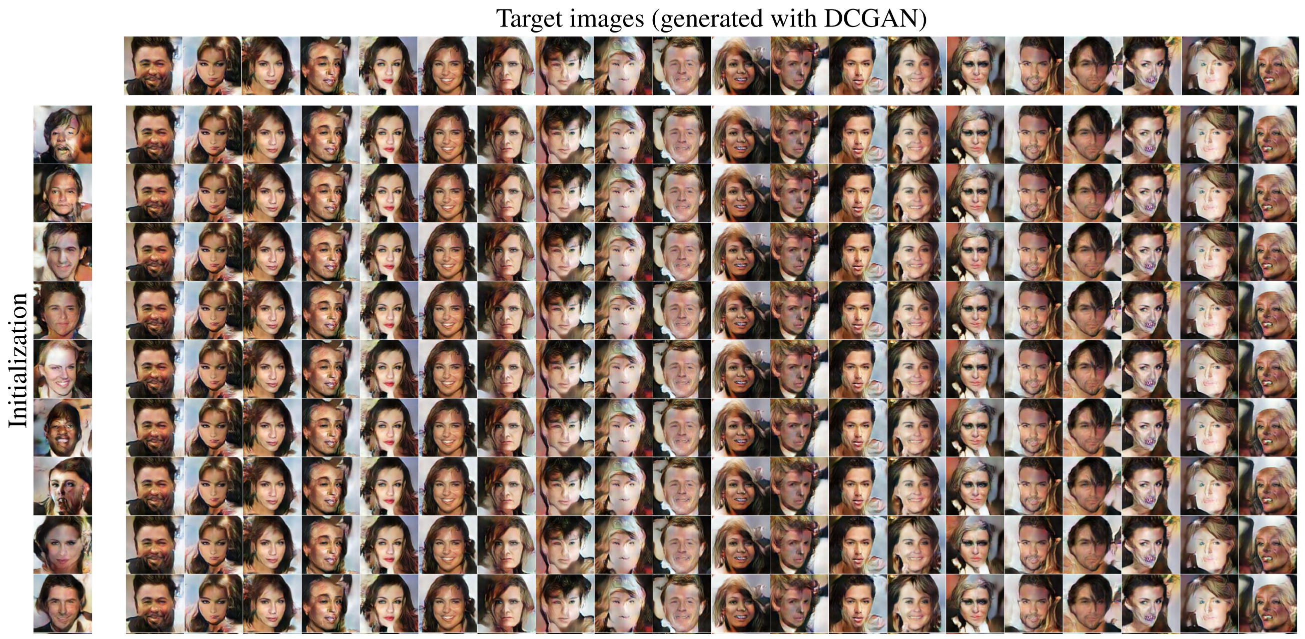


Figure 11: *Visual results on recovery of generated images of a **DCGAN** network. First row: target (generated) images $y_i = G(z^{i*})$. First column: initialization ($G(z_i^{(0)})$). Second column: optimization with LBFGS and Euclidean loss after 100 iterations ($G(z_i^{(100)})$).*

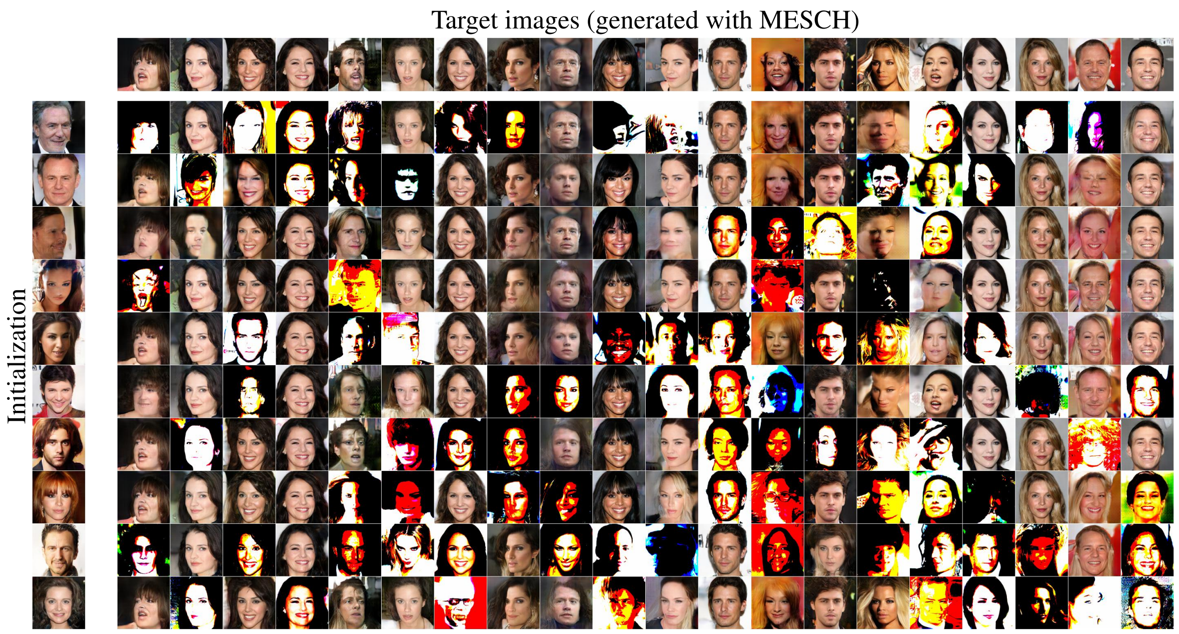


Figure 12: *Visual results on recovery of generated images of a **MESCH** network. First row: target (generated) images $y_i = G(z^{i*})$. First column: initialization ($G(z_i^{(0)})$). Second column: optimization with LBFGS and Euclidean loss after 100 iterations ($G(z_i^{(100)})$). As previously reported in Section 4, success rate of optimization for MESCH generator is only around 67%, but it can be circumvented easily by restarting with new random initialization or by using Adam optimization.*