



# Reconstruction of encrypted faces for presentation attacks on a face recognition scheme.

Armin Niedermüller, Ahmet Bozkurt



## Goals

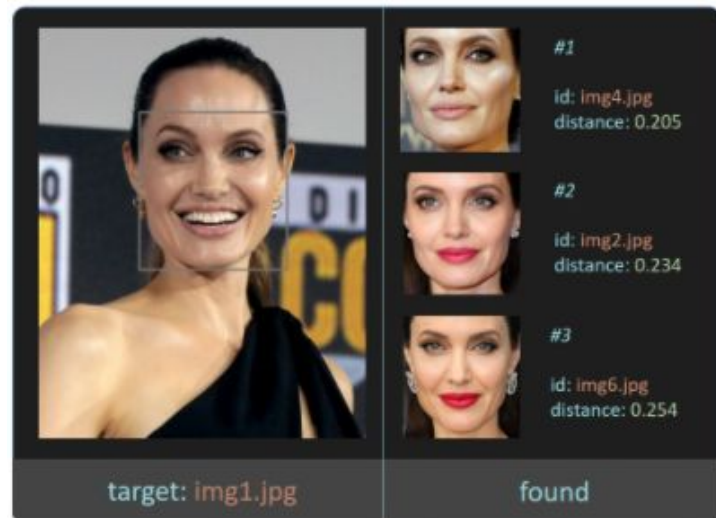
1. Set up a face recognition scheme - in our case: DeepFace using Google FaceNet
2. Test different models to enhance and / or generate plain faces from encrypted faces.
  - a. Use pretrained model weights to generated results
  - b. Train the model weights with our own data (encrypted and plain faces)
3. The goal is to trick a face recognition scheme into recognizing our data as valid results.

# Face Recognition Benchmark - DeepFace FaceNet

Link: <https://github.com/serengil/deepface>

*"A face recognition and facial attribute analysis (age, gender, emotion and race) framework for python. It is a hybrid face recognition framework wrapping state-of-the-art models. Google FaceNet is used for recognition."*

This framework outputs a face distance between two images. A distance  $\leq 0.4$  means that both faces are from the same person.



## Attack Method 1: Pixel2Style2Pixel (pSp) - Super Resolution

Link: <https://github.com/eladrich/pixel2style2pixel>

*“A generic image-to-image translation framework, consisting of different submodules for different tasks.”*

We tried the submodule “Super-Resolution”. Our hypothesis is, that the visually good-looking results of blurred-image-reconstruction (image on the right) can be transferred to encrypted-image-reconstruction.

The network will be trained with our own data, consisting of plain faces and their encrypted counterparts.





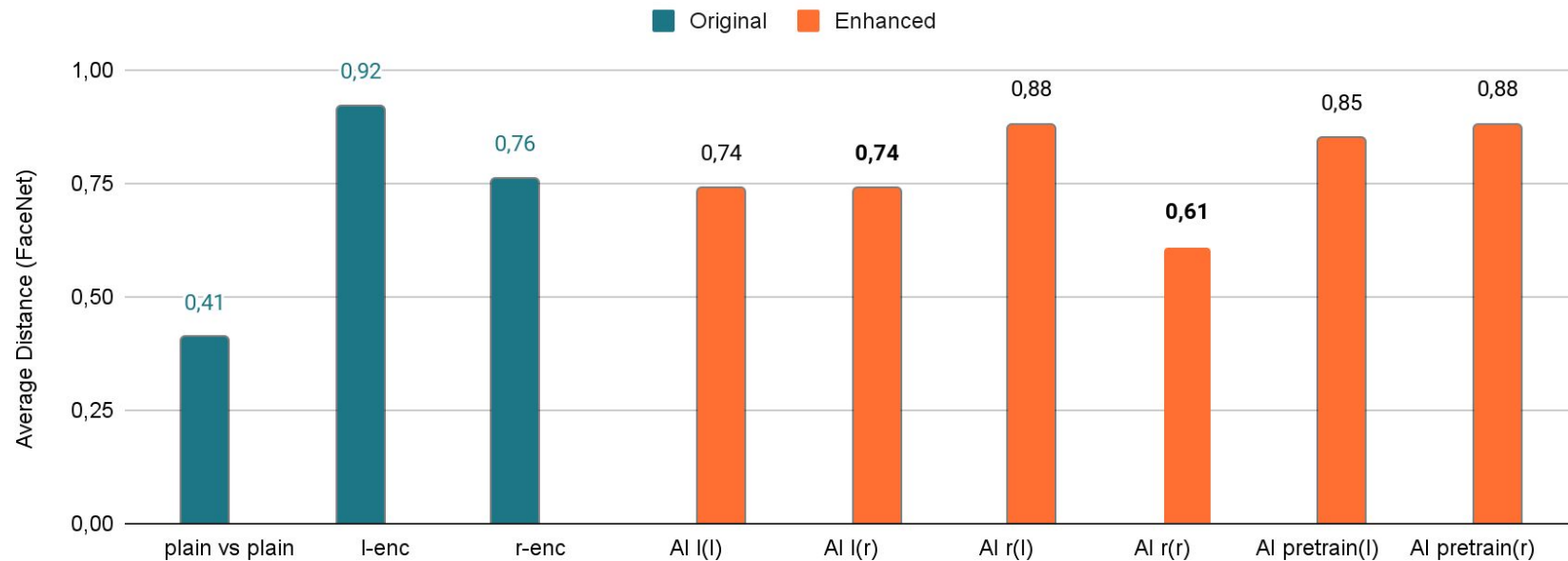
## pSp - Methodology

- **Training:**
  - 175k iterations per encryption method (approx. 24 hours) on:
    - RTX 3090
    - 32 GB DDR3 3600 Mhz
    - 5950 16 Cores / 32 Threads
- **Data:**
  - Test: 50 faces / 1.392 images
  - Train: 10522 faces / 489.231 images
- **Comparison on FaceNet:**
  - Intra class with the last 15 faces of each test set with 5 images each = 60 faces











# Results pSp



## Intra Class Face Comparison - PSPGAN



# Results pSp - Images

<p>P: 0,41</p>  <p>plain</p>	<p>L: 0,72 / R: 0,76</p>  <p><math>l_{enc}</math></p>	<p>L: 0,85 / R: 0,88</p>  <p><math>psp\_pre\{l_{enc}\}</math></p>	<p>L: 0,74 / R: 0,74</p>  <p><math>psp\_l(l_{enc})</math></p>	<p>L: 0,88 / R: 0,61</p>  <p><math>psp\_r(l_{enc})</math></p>
 <p>plain</p>	 <p><math>r_{enc}</math></p>	 <p><math>psp\_pre\{r_{enc}\}</math></p>	 <p><math>psp\_l(r_{enc})</math></p>	 <p><math>psp\_r(r_{enc})</math></p>

## Attack Method 2: : GFPGAN

Link: <https://github.com/TencentARC/GFPGAN>

*"GFPGAN aims at real-world face restoration."*

As an alternative to face generation / inpainting (pSp), we wanted to try a network which is specialized on denoising and restoration. Looking at the examples, the network not only seems capable of deblurring but also denoising.

The network will be trained with our own data, consisting of plain faces and their encrypted counterparts.





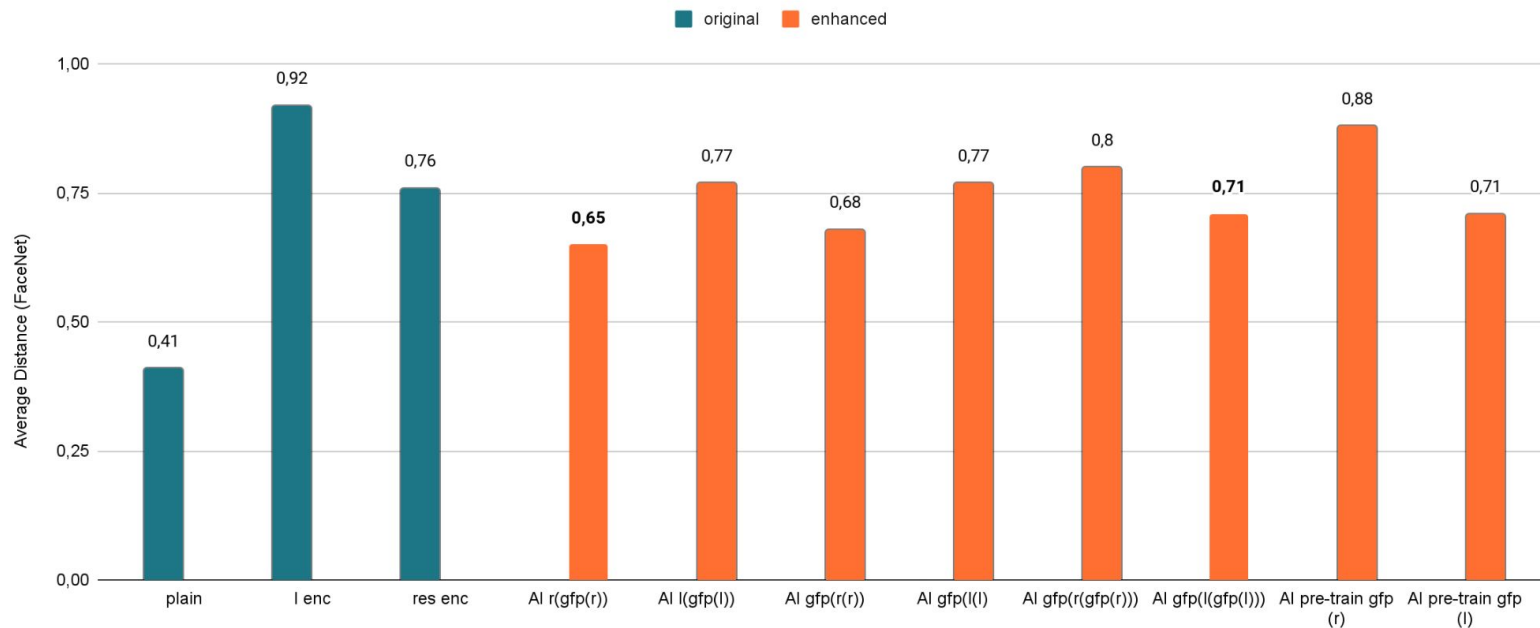


## GFP GAN - Methodology










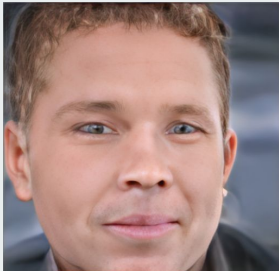


- Training did not work due to not solvable problems in their code
- Even if it would work, the specific distortions from the encrypted data cannot be trained, since GFP GAN creates distorted images from the plain data automatically
- However, GFP Gan showed some very good results in denoising and thus we wanted to test it for post and preprocessing of the PSP Network
- **Comparison on FaceNet:**
  - Intra class with the last 15 faces of each test set with 5 images each = 60 faces

# GFPGAN Results (only pretrained)

Intra Class Face Comparison - GFP GAN / PSP GAN



# GFPGAN Results (only pretrained) - Images

<p>P: 0,41</p>  <p>plain</p>	<p>L: 0,72 / R: 0,76</p>  <p>l_enc</p>	<p>L: 0,71 / R: 0,88</p>  <p>gsp_pre{l_enc}</p>	<p>L: 0,77 / R: 0,68</p>  <p>gsp{psp_l(l_enc)}</p>	<p>L: 0,77 / R: 0,65</p>  <p>psp_l[gsp{l_enc}]</p>	<p>L: 0,71 / R: 0,80</p>  <p>gsp(psp_l[gsp{l_enc}])</p>
 <p>plain</p>	 <p>r_enc</p>	 <p>gsp_pre{r_enc}</p>	 <p>gsp{psp_r(r_enc)}</p>	 <p>psp_r[gsp{r_enc}]</p>	 <p>gsp(psp_r[gsp{r_enc}])</p>