# PMLDL Project Report: Neural Face Editor (NFE)


Andrey Palaev
*B19-DS*
*Innopolis University*
Innopolis, Russia
a.palaev@innopolis.university

Mikhail Rudakov
*B19-DS*
*Innopolis University*
Innopolis, Russia
m.rudakov@innopolis.university

Anna Startseva
*B19-DS*
*Innopolis University*
Innopolis, Russia
a.startseva@innopolis.university


## I. Introduction

With Generative Adversarial Networks (GANs) [1], data synthesis has advanced significantly. The idea of GANs is to find correspondence between a latent distribution and an image distribution via adversarial training. GANs consist of two different networks: discriminator and generator. The generator tries to synthesize images similar to real ones. The discriminator distinguishes real images from ones produced by the generator, i.e., the discriminator and generator aim to maximize the loss of each other. Then, GANs can synthesize realistic images using random latent codes. However, the origins of semantics in the latent space are still unexplored. Even when the data distribution is fair, GANs may favor some classes over others (e.g., it may generate more dogs than cats). The inability to control the images produced by GANs limits their usage. Therefore, it may be beneficial to investigate the semantics that GANs' latent space encodes and how to use them for controlled data production.

In this project, we present Neural Face Editor (NFE), a tool to change anime pictures' facial attributes. Given an anime face picture and desired feature to be changed (i.e. eye color, hair length), NFE will use GANs to edit the image accordingly and return the new image to the user. NFE is implemented in a form of a telegram bot interface to let anyone try this software. For now, the set of attributes is limited to hair length and hair color, but can be extended in the future.

NFE may be a helpful tool for controlled data generation and editing without using expensive and complex software like Photoshop. Moreover, NFE could be used for personalized profile picture generation.

## II. Related work

Although there is prior directly aimed to manipulate anime face images, there are several works on which our project is based on. Further in this section, we describe them.

### A. Generative Adversarial Networks (GANs)

The ability to create realistic images is why Generative Adversarial Networks (GANs) [1] are researched closely nowadays. They generally map a random latent vector from $\mathbb{R}^n$, where $n$ is the dimensionality, to the generated image.
Many GAN architectures have been created. In one of them, called StyleGAN [2], $\mathcal{Z}$ latent space is first mapped into another latent space, $\mathcal{W}$. This latent space is then passed through affine transformation into $\mathcal{S}$ space at each generator's layer. In some cases. $\mathcal{W}$ space is substituted with $\mathcal{W}$ space where $\mathcal{Z}$ vector is mapped into separate W latent vector for every generator's layer. The architecture of StyleGAN is presented in Fig.1.

For GAN evaluation, Fréchet Inception Distance (FID) [3] is usually used. Its objective is to compare the training dataset distribution and the distribution of the generated images.

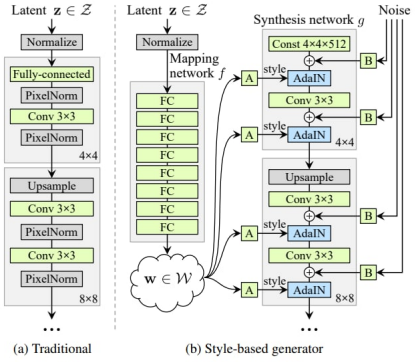


Fig. 1. StyleGAN [2] architecture

### B. InterFaceGAN

InterFaceGAN [4] aims at changing the desired image attribute such as gender and age using latent space of GANs. The main assumption is that a separating hyperplane can be found for any binary feature in the latent space. Hence, they propose to train a Support Vector Machine (SVM) for each attribute and move the latent code toward the separating hyperplane. For the conditional manipulation (e.g. hair colour is changed without affecting hair length), the projection of hyperplane of the changed attribute on the hyperplane of the preserved attribute is subtracted. During evaluation, SVMs achieve minimum accuracy of 75%, verifying the assumption.

The NFE project is mainly inspired by works on interpreting the latent space of GANs for people face generation [4] and other methods of manipulating GAN output [5], [6]. All of these methods try to edit the latent space of GANs to edit the real image in the desired way. We aim to focus on the approach by Shen et al. [4] but might improve it with other approaches mentioned.

### C. GAN Inversion

GANs are used not only in image synthesis but also for real-world image processing. For this GAN Inversion is necessary - method of mapping the images to the latent space. There are two main approaches to GAN Inversion:

1) Optimization-based, where for the given image the latent vector is optimized to minimize the loss between the original image and reconstructed one. Although the training with such a method is simple and accurate, it highly depends on the intialization and has a lot of local minima. Moreover, for every image a separate training is required.
2) Encoder-based, where the encoder is trained to map the image to the corresponding latent vector. Such method is stable, however, choice of an appropriate encoder architecture is difficult and training takes significantly more time than in optimization-based approach.

In contrast, [7] proposed to combine these two approaches: use encoder result as the initialization to the latent vector optimization. In such a way, GAN inversion results are both accurate and fast enough.

## III. Methodology

In this section, we describe the main components of the implemented model. All the neural networks were implemented using PyTorch library [8].

### A. GAN

For the Generative Adversarial Network, the original version of StyleGAN [2] was implemented. The original $\mathcal{Z}$ latent space with the size of 512 features is mapped into $\mathcal{W}$ with the same size using 5 fully connected layers instead of 8 to simplify the network training. The rest of the architecture is the same as Fig.1. For the discriminator, the traditional architecture is utilized. For training, standard loss function was used:

$D^* = \arg\min_D -\mathbb{E}_{x\sim p_{real}} \log D(x) - \mathbb{E}_{z\sim p_z(z)} \log(1 - D(G(z)))$

$G^* = \arg\min_G -\mathbb{E}_{z\sim p_z(z)} \log D(G(z))$

The GAN was trained for 150 epochs with the use of Adam optimizer [9] with the learning rate of 3e-4 for both discriminator and generator.

### B. Image labelling and SVM training

InterFaceGAN [4] requires a dataset with the structure $(z_1, a_1, \ldots, a_m), \ldots (z_n, a_1, \ldots, a_m)$, where $z_i$ is the latent vector, $a_i$ is the value of ith attribute. Therefore, after training a GAN, we need to obtain the labelled dataset. For this purpose, we generated 10k images using trained generator. Then, we labelled them using Illustration2vec [10], a tool for annotating anime images. 3 attributes were obtained: hair colour, hair length and eye colour. Then, 3 SVMs with linear kernel were trained on each attribute.

The NFE project is mainly inspired by works on interpreting the latent space of GANs for people face generation [4] and other methods of manipulating GAN output [5], [6]. All of these methods try to edit the latent space of GANs to edit the real image in the desired way. We aim to focus on the approach by Shen et al. [4] but might improve it with other approaches mentioned.

### C. Image manipulation

Next, we implemented InterFaceGAN [4] for image manipulation in exactly the same way as proposed in the paper. Specifically, we move the latent vector in the direction of separating hyperplane corresponding to the attribute to be changed.

### D. GAN Inversion

Hybrid approach for GAN inversion is implemented which was described in Sec. II.

For training the Encoder, we used the following loss function:

$E^* = \arg\min_E(||G(E(imgs)) - imgs||^2 + 0.01 * D(G(E(imgs))) + 0.002||VGG(G(E(imgs))) - VGG(imgs)||^2)$

The first term aims at making the images produced by inversion similar to the original ones, the second one makes the produced images realistic to the discriminator, the third one makes them similar in the feature space of VGG network [11].

The Encoder was trained on the same dataset as GAN.

Then, for the specific input image we initialize z as $z_{init} = E(img)$

Finally, the latent vector $z_{init}$ is optimized as follows:

$z^* = \arg\min_z ||\sigma(out, 3, 11.0) - \sigma(img, 3, 11.0)||^2 + 20||z||^2$,

where $\sigma$ is the Gaussian kernel.

### E. Overall pipeline

Fig. 2, 3 show the general architecture of the project. Firstly, a GAN is trained to generate anime face images. After that, images are labeled (either by hand or automatically) with facial attributes (eye color, hair length, etc.). Finally, each attribute subset is passed as an input to the SVM classifier model, which learns to separate latent vectors based on each attribute separately. Image editing is possible by using SVM's hyperplanes in latent space to change specific attributes.

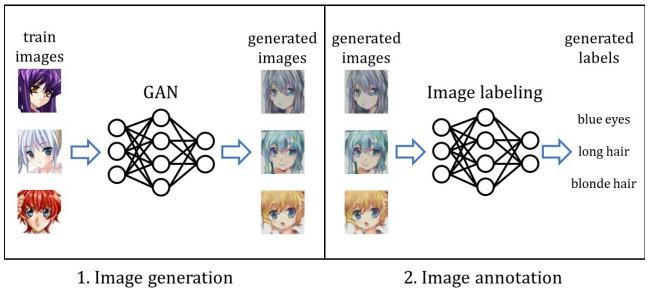


Fig. 2. NFE architecture: Image generation and annotation.

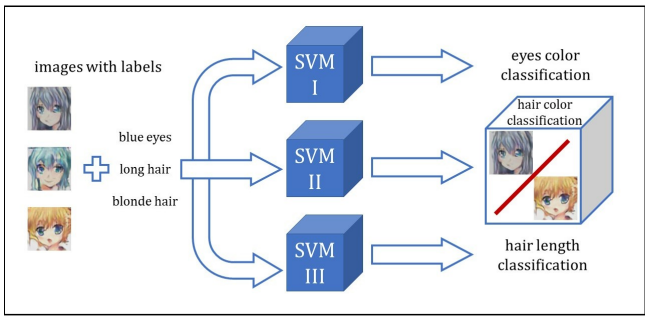


Fig. 3. NFE architecture: Generated images classification.

### F. Dataset

For training GAN, we need an anime faces dataset. We found several datasets that could be used for the project. The first one is the anime face dataset from Kaggle [12], which contains 63k images. It can be useful for training GAN as no image preprocessing is needed and it can be simply downloaded from Kaggle. Another dataset we found is Danbooru [13]. This is a large public dataset (5m images) of anime images with 162 million tags present. This dataset could be used for large, high-quality GAN creation, but requires a lot of preprocessing work to be done (extract images, and select proper tags). We will firstly focus on exploring these two datasets, however, we will also continue to search for more appropriate datasets.

## IV. GitHub Link

The project code is available at the following link: https://github.com/Palandr1234/NFE.

In the project Readme file, demo gif showcasing bot interaction is available.

## V. Experiments and Evaluation

### A. Empirical study

First, we evaluated the quality of the images produced by GAN. Fig. 4 shows the examples of generated images. Although their quality may seem acceptable, images from GAN still lack diversity and visual quality from the original dataset.

Next, we empirically evaluated the quality image manipulations from InterFaceGAN [4]. Fig. 5, 6 show the examples of produced manipulations. Although the overall quality of the manipulations is acceptable, with large manipulation strength $\alpha$, the image becomes unrealistic. Also, Fig. 5 show that sometimes the hair length is still changed while we explicitly set it as preserved attribute.

Finally, we empirically evaluated GAN Inversion. Fig. 7 shows some examples of image produced by GAN Inversion. Despite the fact that overall image patterns are preserved well, the image details are not well preserved.

### B. Quantitative evaluation

Firstly, we calculated Fréchet Inception Distance (FID) [3] between the original dataset and 10k images generated by GAN. For that we used the following repository: https://github.com/mseitzer/pytorch-fid. The resulting FID score is 55.32 which is not a state-of-the-art, but still satisfactory.

Next, we evaluated the performance of SVMs trained for InterFaceGAN [4]. Table I shows train and test accuracy of SVMs. As can be seen, all SVMs are overfitted. Moreover, SVM trained on eye colour did not manage to achieve accuracy of 1.0.

## VI. Analysis and Observations

First observation is that our GAN suffer from mode collapse. It happens when generator tries to produce the same (or several similar) output(s) for different latent vector. One of the possible reasons is that StyleGAN is hard to train and requires a lot of computational resources which we do not have. Therefore, in the future we may try to increase the amount of parameters in our GAN.

The next point is the overfitting of all 3 SVMs. The reason is the low amount of generated latent vectors comparing to the amount of parameters (maximum 4k latent vectors comparing to 512 parameters of the latent vector). In addition, illustration2vec network is not precise, especially with eye colour. That is also the reason why the manipulations on the eye colour often fail. Hence, in the future we can create more robust annotator that will generate more examples (at least 10k) for training SVMs.

Finally, we noted that manipulations on the images produced by our GAN Inversion are unrealistic even comparing to manipulations on GAN generated images. The reason is that GAN Inversion produces the vector that lies in the low likelihood of the latent space. In the future, we need to elaborate on the GAN Inversion approach more to make manipulations on such images more realistic.

## VII. Application Technical Details

We developed telegram bot for convenient use of our NFE application. Bot tag is @neural_face_editor_bot. Telegram bot uses TelegramBotAPI to connect to Telegram and PyTorch to use our model scripts to construct our app into one pipeline.

Telegram bot is deployed on Yandex Cloud service. However, current limitation is that server does not have GPU, as it is too expensive. Therefore, it is not possible to compute GAN Inversion in suitable time on that server, and this option is disabled there. If one hosts bot on local machine with GPU, GAN inversion would work and computes rather fast (around 5 seconds).

Fig. 8-12 show possible interactions with our bot, including image uploading and GAN inversion, random image generation and image manipulation. Bot contains help command to tell possible interaction options to the user.

| | Dataset size | Train accuracy | Test accuracy |
|---|---|---|---|
| Eye colour | 4258 | 0.9 | 0.67 |
| Hair length | 1711 | 1.0 | 0.73 |
| Hair colour | 3182 | 1.0 | 0.67 |

TABLE I
ACCURACY OF TRAINED SVMS

Fig. 4. Examples of generated images

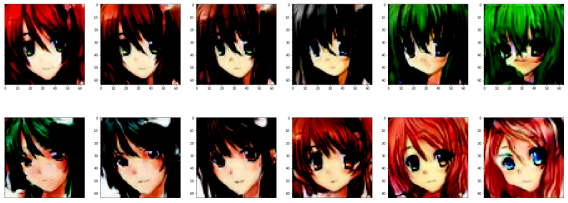

Fig. 5. Changing hair colour preserving the hair length

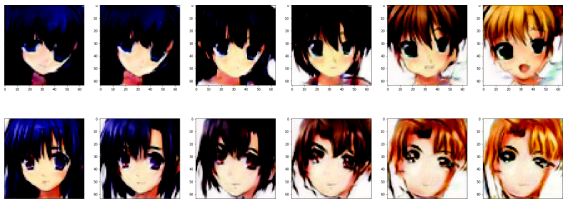

Fig. 6. Changing hair length

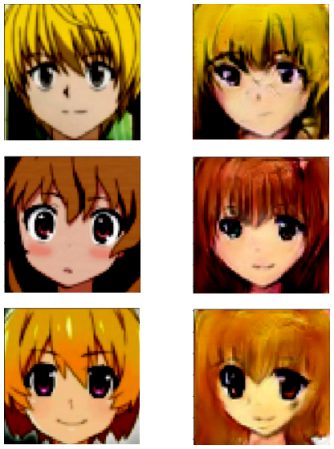

Fig. 7. Examples of GAN Inversion: Input images are on the left, results - on the right

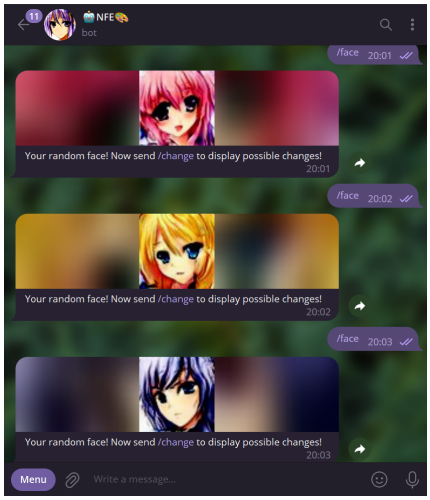


Fig. 8. Bot Face Generation

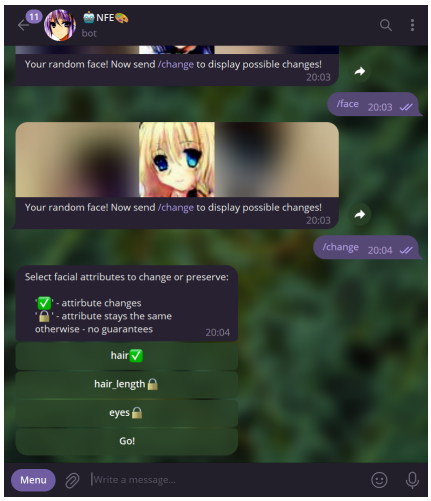


Fig. 9. Bot Face Manipulation Input

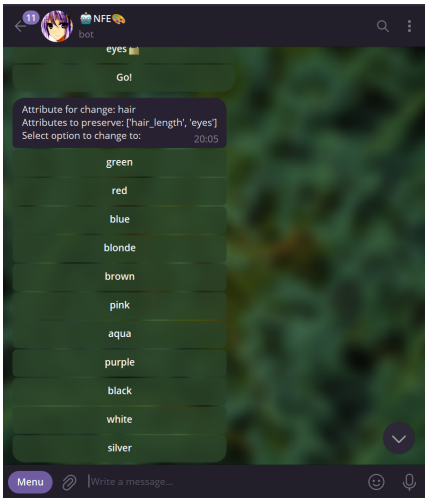


Fig. 10. Bot Changing Options

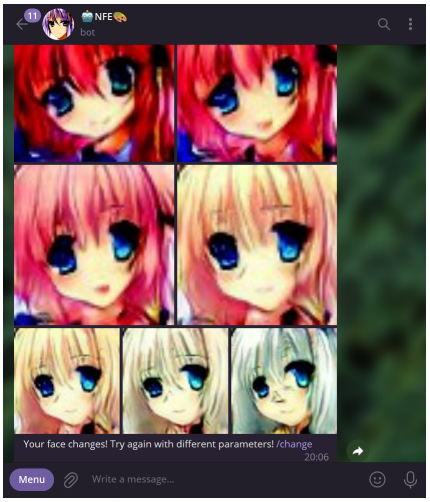


Fig. 11. Bot Face Manipulation Output

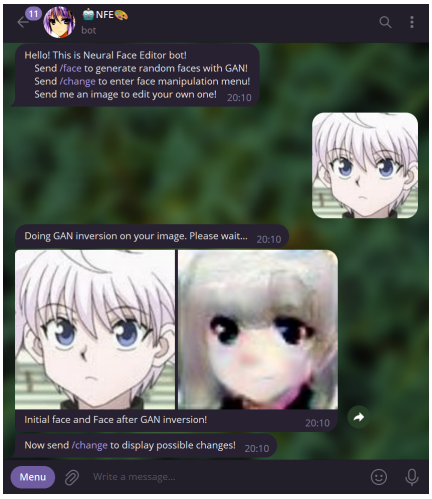


Fig. 12. GAN Inversion

## VIII. Timeline and Individual Contributions

The project timeline and individual contributions can be seen in Table II.

| Due Date | Task | Assignee |
|---|---|---|
| 20.09 | Read papers on latent space manipulation | Mikhail |
| 20.09 | Select approach for image editing | Whole Team |
| 20.09 | Find dataset with anime faces | Anna |
| 27.09 | Implement selected GAN | Andrey |
| 27.09 | Train GAN on dataset with anime faces | Andrey |
| 27.09 | Generate anime faces with GAN | Andrey |
| 04.10 | Find tool for face labeling | Anna |
| 04.10 | Generate labels for GAN images | Andrey |
| 11.10 | Assess generated by GAN images | Anna |
| 11.10 | Implement SVM for each attribute | Andrey |
| 19.10 | Evaluate SVM for each attribute | Anna |
| 26.10 | Read papers about GAN Inversion | Mikhail |
| 01.11 | Implement image manipulation based on SVMs | Andrey |
| 01.11 | Implement GAN Inversion | Anna |
| 08.11 | Implement basic Telegram bot functionalities | Mikhail |
| 15.11 | Implement anime face generation in Telegram bot | Anna |
| 22.11 | Implemet the rest of functionality in Telegram bot | Mikhail |
| 29.11 | Test Telegram bot | Anna |
| 04.12 | Final Report and Presentation | Whole Team |

TABLE II
Project Timeline and Work Distribution

## IX. Conclusion

Neural Face Editor is application that utilizes GAN latent space manipulation to change anime faces facial attributes. Application is available in form of telegram bot via @neural_face_editor_bot.

To achieve such result, StyleGAN [2] is trained on animefacesdataset [12] for anime faces generation. Then, several SVM are trained for separate attributes. Finally, ideas from [4] are used to perform unconditional and conditional attributes manipulation. To let user use their own pictures, hybrid GAN inversion [7] is implemented. To assess GAN image generation, FID [3] distance is calculated on the dataset of real and fake images. Output quality of generated images has been assessed manually.

We want to outline several directions for future work. Firstly, GAN inversion technique seems to produce to produce too poor result on user images, which leads to bad results in attributes manipulation. Possible advancements are to increase number of epochs for GAN inversion or choose better GAN inversion approach. Secondly, FID metrics between distributions of real and fake images is not perfect. This indicates that GAN Generator did not succeed in generating good images. The reason may be the small size of the anime faces dataset we are using. Hence, possible improvement is to use larger dataset for GAN training. Finally, our telegram bot is currently hosted on local machine to let GAN Inversion work, which requires GPU. Version of the bot, hosted on the Yandex Cloud, has no GPU available. Hence, for better bot maintainability and functionality, bot should be hosted on reliable server with GPU.

## References

[1] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," in *NeurIPS*, 2014.

[2] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," in *CVPR*, 2019.

[3] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "Gans trained by a two time-scale update rule converge to a local nash equilibrium," *Advances in neural information processing systems*, vol. 30, 2017.

[4] Y. Shen, J. Gu, X. Tang, and B. Zhou, "Interpreting the latent space of gans for semantic face editing," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 9240–9249.

[5] A. Plumerault, H. Le Borgne, and C. Hudelot, "Controlling generative models with continuous factors of variations," in *International Conference on Machine Learning (ICLR)*, 2020. [Online]. Available: https://openreview.net/forum?id=H1laeJrKDB

[6] Y. Jin, J. Zhang, M. Li, Y. Tian, H. Zhu, and Z. Fang, "Towards the automatic anime characters creation with generative adversarial networks," 08 2017.

[7] J. Zhu, Y. Shen, D. Zhao, and B. Zhou, "In-domain gan inversion for real image editing," in *ECCV*, 2020.

[8] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *NeurIPS*, 2019.

[9] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *ICLR*, 2015.

[10] M. Saito and Y. Matsui, "Illustration2vec: a semantic vector representation of illustrations," in *SIGGRAPH Asia Technical Briefs*, 2015.

[11] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *ICLR*, 2015.

[12] S. Churchill and B. Chao, "Anime face dataset," 2019. [Online]. Available: https://www.kaggle.com/ds/379764

[13] Anonymous, D. community, and G. Branwen, "Danbooru2021: A large-scale crowdsourced and tagged anime illustration dataset," https://www.gwern.net/Danbooru2021, January 2022. [Online]. Available: https://www.gwern.net/Danbooru2021