GAN-based Image Editing

Kepler Warrington-Arroyo, Tomás Ebensperger Buschmann, Julie Favre

EPFL **

Supervisor: Ehsan Pajouheshgar

Professor: Sabine Süsstrunk

Date: 02.06.2023

Abstract. Generative Adversarial Networks (GANs) allow generating natural looking images of a specific domain, such as faces or bedrooms. In this paper, we aim to build upon the LELSD framework as defined in Optimizing Latent Space Directions For GAN-based Local Image Editing (Pajouheshgar, Zhang, and Süsstrunk, 2018) to find directions of the latent space that correspond to different semantics, to then find ways to manipulate the input latent code or weights of a pretrained GAN to achieve the desired change in the generated image from a text prompt. We will use three different models: a pretrained StyleGAN2 model, CLIP and Face-Bisenet, a segmentation model, to make sure that the generated images are on the natural image manifold, and have the desired properties from the text prompt.

Keywords: GAN, StyleGAN, StyleGan2, CLIP, Semantic Segmentation

** project supervised by Ehsan Pajouheshgar in CS413 at EPFL.

1 Introduction

1.1 Problem definition

The aim of this project is to train a model being able to modify sequentially an image given a text prompt. We want it to be specific, i.e. change only the region of interest and leave the rest of it as intact as possible. For this, the model has to recognize which part of the text prompt is most important, and modify the picture in consequence. Models being able to modify the image on only certain regions already exist [8], as well as models computing the similarity between an image and a text prompt [9]. The interesting part of this project is incorporating both in one single model, in a balanced way, while preserving the interesting properties of both approaches.

1.2 Existing solutions

A proposed solution exists in Paint by Word [1]. In their approach, a first model solely uses CLIP to try to modify the image. However, this does not guarantee locality: if the model is asked to 'make the bed purple', CLIP will give the best score to the image with everything being tainted purple, without specific focus on the 'bed' element of the prompt. If a hand-drawn mask is added to the image, on the bed, and the model is asked to only modify this part, the results are much better: the bed is purple but not the rest of the room. However, even though this produces good results, it is not user friendly. Having to paint and select the mask every time is long, and we believe it could be done more efficiently as some models using an appropriate segmentation model.

LELSD, upon which this project is based, is such solution for generating images and being able to sequentially edit them. The semantic regions are predetermined with the help of masks, provided by a segmentation model. The model then finds the latent directions corresponding for each region, or semantic feature. With this framework, we can then modify only the eye, as the model know which latent direction such a localized edit corresponds to. One of the downside is that there is no control over the semantic quality of the edit: though it only changes the eye, the user has a lack of control on the specific eye attribute being changed (color, size, orientation, etc).

InterFaceGAN [11] is a technique mixing GANs and an interface letting the user modify the input image on different latent spaces. However, it does only work on faces, and on the static latent spaces decided by the developer. We want to develop something more flexible, as applicable to churches as to faces. We also want the user to be able to choose whatever direction they want, corresponding to a text prompt, and not limit to a few chosen directions at the beginning.

1.3 Our proposition

We propose eCLIPsed, an extension of LELSD which incorporates CLIP, in order to have a locally effective edit, based on a text prompt. From the text prompt, we will select the masking region we need, (eye, bed, etc). CLIP loss will be incorporated in the training, being add to the localization loss. This results in models trained to perform edits for one specific feature of a specific region (i.e. blue eye).

1.4 Motivation

GANs are a very interesting tool, and a quickly-growing field of research. They have developped outstanding capacity and power in these last years. However, for a lambda user, using them to edit a picture in a specific way is not always easy. This project aims to ease the access to GANs with accessible software and models, allowing users with everyday computation power to leverage the power of GANs to generate images of their choosing, without having to re-train an entire model.

2 Literature review

This project required a somewhat extensive literature review before getting started.
DCGAN [10] uses GANs and a CNN, and builds a model whose latent space's structure allows for arithmetic of semantically relevant vectors. For example, a possible operation is Vec('King')$-Vector$('Man')$+Vec$('Woman') $= Vec$('Queen'). DCGAN is a striking example of the power of leveraging the structure of the latent space, using relevant directions to easily edit elements according to the user's wishes.

GANSpace [3] puts forward a way to introduce edits and specifications to a pre-trained GAN's generator network, acting as somewhat of a "black box", to control attributes of the output image. This allows for more general use of the generator without having to re-train it or use supervised methods. In the case of both StyleGAN [7] and BigGAN [2], using PCA in the latent space enables "browsing" through the concepts that the GAN has learned. The authors also explain the method for applying that knowledge to introduce specifications to the network and control its output.

StyleGAN1 [6] proposes an alternative generator architecture for GANs, borrowing from style transfer literature. It leads to an automatically learned, unsupervised separation of high-level attributes (e.g., pose and identity when trained on human faces) and stochastic variation in the generated images (e.g., freckles, hair), and it enables intuitive, scale-specific control of the synthesis. The new generator improves the state-of-the-art in terms of traditional distribution

quality metrics, leads to demonstrably better interpolation properties, and also better disentangles the latent factors of variation. They also create a new, highly varied and high-quality dataset of human faces.

The core architecture described in the implementation of StyleGAN plays a fundamental role in generating images with specific characteristics. This architecture is composed by multiple layers responsible for generating images with distinct components. Moreover, StyleGAN introduces the concept of "styles" that vectorize the inputs, influencing different parts of the generated images and contributing to semantic variations.

StyleGAN2 [7], a study upon which our work heavily relies, represents a substantial advancement over StyleGAN, with the aim of achieving superior results in image generation. This work presents several notable advancements in the methodology, significantly refining the image generation process. Notably, it features a revised pipeline for image generation. StyleGAN2 addresses the issue of "droplets" that appeared in images generated by StyleGAN1. These droplets, caused by AdaIN [5] (Adaptive Instance Normalization) normalization, inadvertently revealed that the images were artificially generated. StyleGAN2 effectively resolves this problem, resulting in more visually realistic outputs.

Additionally, the discriminator network in StyleGAN2 exhibits improved discernment, successfully identifying and rectifying the aforementioned artifacts that were not effectively captured by the discriminator in StyleGAN1.

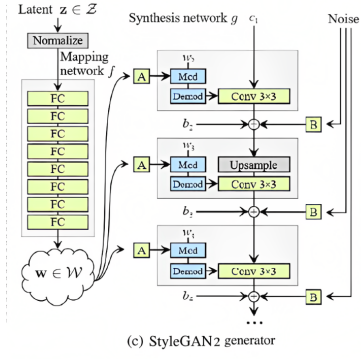

Fig. 1: Comparison of the Pipelines of StyleGAN and StyleGAN2 [4]

Figure 1 shows a visual representation of the pipeline used in StyleGAN2. This diagram illustrates the various stages and operations involved in the image generation process. The pipeline begins with data preprocessing, before feeding the

input into a specific network architecture, which then synthesizes the output image, all steps working together to produce high-quality synthetic images.

Another main resource for our project is the aforementioned LELSD [8]. It presents a new approach to evaluate the locality of an image edit. It uses a pretrained segmentation network and a localization score that they maximise in order to isolate the region of interest penalize edits to the image that cause modifications outside of the boundaries of the mask. This scoring function encourages the latent space directions to precisely edit the desired part. It uses the layer-wise approach of style-based GANs discussed above, which allow coarse and fine-grained semantic changes. The method used is shown in Figure 2

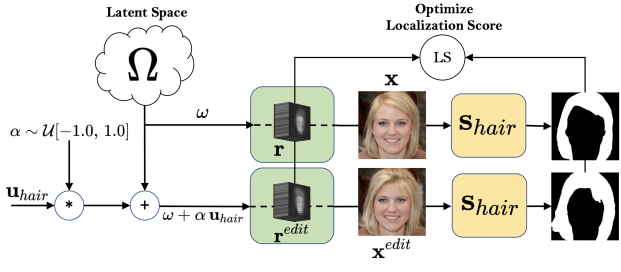

Fig. 2: The method used in LELSD

It is compatible with all existing GANs, and can be used with any dataset. Our approach leverages LELSD for a dataset of human faces, but the approach is generalizable to other models capable of generating a variety of objects, such as churches, horses, cars, and more.

CLIP [9] plays a fundamental role in our project, since our greatest contribution comes from using it to implement a semantic editing functionality linked to natural language. CLIP is a neural network which learns visual concepts from natural language supervision for efficiently classifying given groups. As stated by its creators, the network "can be instructed in natural language to perform a great variety of classification benchmarks, without directly optimizing for the benchmark's performance".

The last paper we used in our research is Paint by Word [1]. They use CLIP in order to edit the selected region with a text prompt describing the wanted output. Instead of creating a new image based on a prompt, they wish to edit an already existing one. This paper shows how well two very different models, StyleGAN2 (or BigGAN) and CLIP, can be easily combined and work hand in hand to create interesting results. Paint By Word uses a semantic similarity network $C(x, t)$ which scores the semantic consistency between an image x and a text t in order to ensure that the output images match the text prompt.

They also use a convolutional generative network $G(z)$ to enforce realism of the image, and be sure that the modification blends in with the rest of the image, accounting for the lightning, placement, etc. The image z^* is then generated such that $z^* = \arg\max C(G(z), t)$.

3 Implementation

The goal of eCLIPsed is to present an extension of LELSD, producing small models which are capable of performing single specific edits (of variable strength) to images within the StyleGAN2's latent space(s). As such, the crux of our implementation relies on adding a CLIP score as an optimizable loss metric during LELSD training.

CLIP similarity is computed from the embeddings of both the text and image in CLIP-space, which can be obtained by using the functions below:

```
model.encode_text(text: Tensor)
model.encode_image(image: Tensor)
```

The similarity is given by the cosine similarity of the embeddings, meaning it is constrained to the [-1, 1] continuous interval. From this, we derived a straightforward loss metric:

$$\mathcal{L}_{\text{CLIP}}(\texttt{img}, \texttt{text}) = 1.0 - \mathit{encode}(\texttt{text}) \times \mathit{encode}(\texttt{img})^T$$

which we aim to minimize during training, such that generated images are as CLIP-similar to the prompt as possible.

Furthermore, to make the project more user-friendly, and automate some technical aspects away from the end-user, we chose to add some NLP functionality, using the spaCy Python package. Firstly, given that the use of segmentation models within LELSD requires that their segmentation parts be internally specified, we leveraged this to automatically determine which part of the image should be masked. We compute sentence similarity between the prompt and the recognizable parts; the part with the biggest similarity to the prompt is then chosen as the segmentation part. The implementation of the NLP features also allows for prompt engineering, which we used to run small experiments, to little overall success. However, we still believe prompt engineering could bring out interesting possibilities for the project as a whole.

To evaluate and implement our project, we made use of many different tools, presented in the subsections below.

3.1 Streamlit

Incorporating numerous Streamlit apps into our workflow enhanced our project's data visualization capabilities and facilitated efficient data manipulation. With Streamlit, we visualized our data and were able to perform data manipulation operations with ease. Streamlit apps allow users to quickly generate a website interface to edit parameters in the code and visualize results painlessly. During the development of the project, we mostly used `sequential_editing.py`, which we adapted from the LELSD implementation for our project. This app allowed us to change the value of the α parameter quickly, and visualize the effects and strength of our edits. Several other apps were created for development purposes and then abandoned, and did not make it into the final project, but all served the general purpose of visualizing our results and making sure the trained models behaved as expected.

By running the `train_eclipsed.ipynb` notebook, one can train a model according to a specified prompt using our final chosen parameters, and run one of our apps to visualize the results.

3.2 Using CLIP

We did a few tests with CLIP to better understand its workings and performance, in order to optimise the computation of the CLIPLoss when integrating it to our model.

We first wanted to see how good CLIP was at recognizing features on a face, for example beard, glasses, etc. We can see in Fig. 3 that it sometimes gives very good results, but also struggles with some features. It is pretty good at differentiating high-level features, such as the difference between a man, a woman and a child. There is no issue with recognizing which people have a hat or not. However, in the case of glasses, it asserts with 78% probability that the 6th woman is wearing a pair, when it is not the case. This may also be due to the image being a better match for the prompt overall — however, in this case, the score is still too high for our purposes, as we are mostly interested in editing finer-grained elements of an image, and this could be confusing for the segmentation inference.

We then wanted to see if CLIP was good at recognizing emotions: as we can see in Fig. 4, this is unfortunately not the case. The model decides that everyone has a neutral expression, even the images of people who are clearly smiling. This indicates that CLIP struggles with higher-level abstract concepts, which influences its performance on certain prompts (i.e. 'a picture of a happy person').

Lastly, we noted that even with accurate results, the difference in scores is sometimes very small: between two images which we would classify as being very

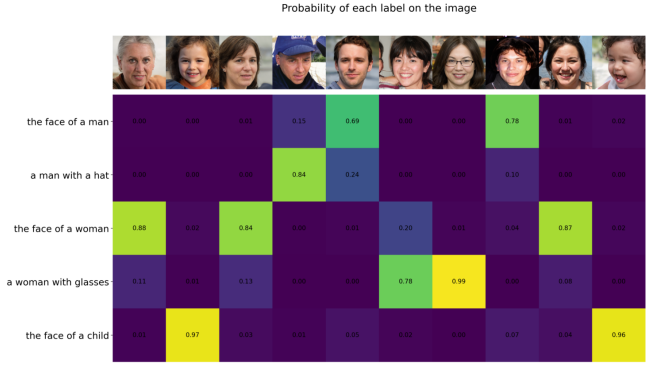

Fig. 3: Results for CLIP when given features on a face

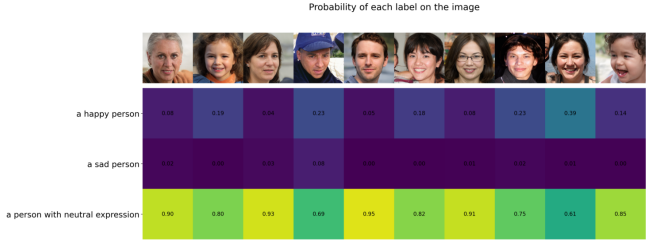

Fig. 4: Results for CLIP when given emotions

dissimilar, the CLIP score can nonetheless be very similar. Given two pictures of men, CLIP will hang onto the first part of the prompt, 'the face of a man with ...'. We aimed to find solutions to highlight this difference and increase the delta between scores. We were given the idea of emphasizing the most relevant part of the text prompt. Here, we once again used spaCy to retrieve the noun chunk most relevant to the segmentation part (i.e. from 'A person with an open mouth', we retrieve 'open mouth'). The prompt is then reconstructed, removing stop words (which have little influence in CLIP's scoring), copying the noun chunk many times where it previously appeared in the prompt. We hoped that CLIP would recognize this part as more important and increase the difference in scores. However, as seen in Fig. 5, this did not produce convincing results: CLIP's evaluation worsened, even where it used to give good results, such as short/long hair differentiation. We had to give up this technique, and simply added more weight to the CLIPLoss when computing it.

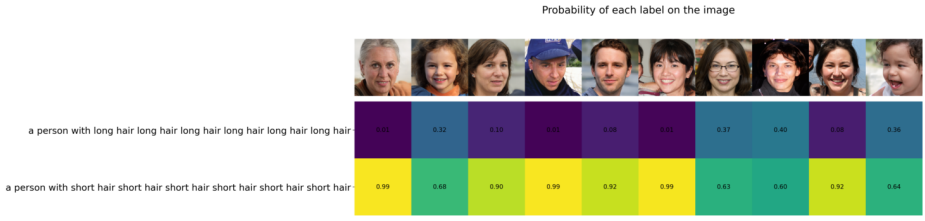

Fig. 5: CLIP results when emphasing the most important part of the sentence

3.3 Choice of parameters

To arrive at our final implementation, we tried a lot of different parameters. We had external (prompt engineering, CLIP loss weight) and internal parameters (LELSD training hyperparameters) to play with. Table 1 shows comparisons of the results between our final choice of parameters, and an alternate choice. For each comparison, we train one model per both parameter choice, then choose one random seed, and compute CLIP's similarity score between the prompt and both images individually. For the results reported in the table, we trained with the following constant parameters:

- **Prompt:** *"a photo of a person with curly hair"*
- **LELSD layers where the edit is applied:** {3, 4}
- **Alpha value:** 6 or 7 (same for both images generated from a same seed), depending on the aesthetics of the image.

Parameter	Final parameters	Alternate parameter
Prompt (Normal vs extended)	0.192	0.177
Single seed training (same seed)	0.188	0.205
Single seed training (on random seed)	0.207	0.193
$\mathcal{W}+$ vs $\mathcal{W}$	0.213	0.201
localization score (0.05 vs 0)	0.201	0.204
localization score (0.05 vs 0.2)	0.201	0.174

Table 1: Comparison of the CLIP similarity score with the change of one parameter.

where "single seed training" refers to training the model on a single seed: our first experiment consisted of training the model on a single seed, and evaluating the CLIP similarity between the edit being performed by that model on that same seed and the same edit being applied to that seed by a model trained on

random seeds. The second experiment consisted of comparing the CLIP similarity between an edit being applied to a different seed than the one the model was trained on, and the same edit being applied to that seed by a model trained on random seeds. These results mostly highlight that CLIP is an unreliable metric to report results by: though it is useful during training, and our CLIP loss does indeed decrease, the change in score between images generated from the same model with the same parameters but different seeds often brings out CLIP's difficulty to evaluate differences in similarities in a non-constrastive experiment. However, the small differences in CLIP score are sufficient to effectively guide the finding of latent directions during training.

For the external parameters, we had the prompt, and whether to train on one seed or a collection of random seeds. We tried, as explained in 3.2, to modify the text prompt itself. We made it as descriptive sentence (a photo of a woman with blond hair), imperative sentence (make the hair blond), or with the main part of the sentence enhanced (a photo of a woman with blond hair blond hair ...). The imperative sentence had very poor result, as CLIP works best with a description of a picture. Enhancing the main part had little to no result, so we ended up not exploring that path further. We can also see in the table that the similarity score drops when using the extended prompt.

For the training dataset, we both tried to train on random seeds or only on a single image. Training on a single image makes the similarity score higher when evaluating on the sample that we trained on. (Tab. 1, row 2). However, it is less generalizable, as when looking at the score on a random sample, the score is higher when trained on random seeds.

We observe various behaviors associated with different internal parameters. First of all, we compared the results when using $\mathcal{W}$ or $\mathcal{W}+$. The score is slightly better when training on $\mathcal{W}+$. However, when we analyzed them qualitatively, the $\mathcal{W}+$ results seemed to produce much more expressive changes. Therefore, we do not need to change the alpha range much to obtain clearer results.

The last change of parameters we tried was the localization score. This score seemed to really easily swallow up the CLIP loss during training. Indeed, whenever it is higher than 0.1, the edits are barely noticeable, even with extreme alphas. We then tested for our benchmark with 0, 0.05 and 0.2. You can see the differences in Fig. 6. Having no localization score allows the photo be updated in other regions than the desired semantic part: here, we can see that the mouth gets opened when the prompt was 'A person with curly hair'. Adding too much localization (0.2) really dampens the changes from CLIP, and the results are less expressive. We then chose 0.05 as a localization factor, as we can still clearly see the expression of curly hair, but the rest of the image is unchanged. For the score, as seen in Table 1, we can see that CLIP gives a better score to localization 0, as to be expected, but the difference with the small localization factor is

(a) LS 0 (b) LS 0.05 (c) LS 0.2

Fig. 6: Comparison of the results with the change of parameter Localization score (LS)

very small. With a localization factor of 0.2, we see that CLIP computes a much lower score, with a difference of 0.03, which is one of the biggest differences in comparison with the other deltas.

The final parameters we chose are as follows:

- **Prompt:** Not expanded, used as-is
- **Training:** Random seeds
- **Latent space:** $\mathcal{W}+$
- **Localization score:** 0.05

We had to evaluate many parameters qualitatively, following 'visual' criteria. When adjusting the alpha range, we noticed that this parameter determines the intensity of the changes obtained, but scales the edits differently with different training parameters, namely the choice of latent space. We came to the conclusion that it is critical to keep a small range with our final parameters, as expanding this parameter in all cases leads to distorted and extreme results, which affects the realistic appearance of the photos.

Finally, we performed a crucial test for our app by experimenting with the different layers. Based on the literature, we knew that image changes are associated with specific layers. Therefore, by applying modifications to given layers, we can get even better and more accurate results, depending on the desired modifications. The layers also have a strong influence on localization: indeed, using our current parameters and models, and 18 style layers, we found that layers 7-8-9 seem to uniquely affect hair color, leaving other semantic aspects untouched. Similarly, layers 4-5 affect the mouth and some of the hair.

4 Results

After performing a wide variety of experiments, we were able to obtain the desired results, finally managing to modify images by moving along specific directions within the latent space.

First, we used a model trained on the prompt "a person with the mouth open" for two different images, as seen in figure 7. This provided expected results, with the mouth seeming to naturally "open" as the alpha increases, and supports that models trained with eCLIPsed are generalizable. We then generated images using another prompt, "a person with blonde hair", the results of which can be found in figure 8. We observed not only that following that direction makes the person blonder, but also going to the opposite direction makes their hair darker. This is also the case for the "open mouth" pictures, where the opposite direction shows results with a mouth that is more closed than the original image. From this, we can state that the directions found with our methods do indeed correspond to semantic elements of the image which allow manipulation beyond the specific given prompt.

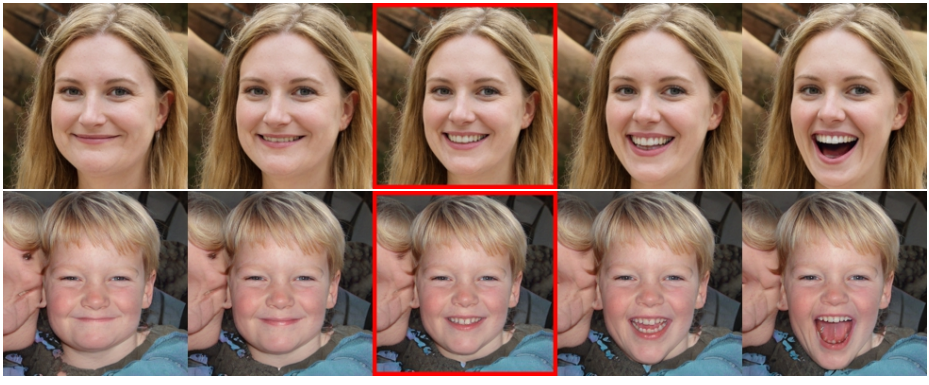

Fig. 7: Result for prompt: a person with the mouth open

Fig. 8: Result for prompt: a person with blond hair

It is important to note that the only semantic that is changing in this process is the one explicit in the prompt, so the final result shows the same person that appears in the original image; it is possible and easy to recognize them. Edits performed with no regards for locality often change key elements of the face as "collateral" edits, leading to the faces morphing into entirely different people as the strength of the edit is changed. This effect is enhanced when CLIP is brought into play, as it gives better scores to images with many correlated attributes which are highly related to the prompt. Indeed, during our experiments, we found training on prompts such as "a person with red hair" quickly produced edits where a person appeared paler, some with freckles, which we can attribute to both the distribution of data within the dataset and CLIP's effect.

Despite the good results above, we also found that eCLIPsed's performance suffered when faced with some specific prompts. For instance, as seen in 9, we observed that eCLIPsed had difficulty generating images for people with glasses without modifying other attributes. We can in part attribute this kind of behavior to the automatic segmentation detection part of our project, which uses a small language model which does not contain appropriate embeddings for precise computation of sentence similarity, and sometimes selects the wrong segmentation part. However, even by manually specifying which semantic element to mask, we did not obtain good results.

Fig. 9: Result for prompt: A person with glasses

Finally, using prompts which do not correspond to features within the dataset leads to poor results. Prompts such as 'a picture of an alien' or 'a picture of a truck', used with localization turned off, result in latent space directions which do not seem to correspond to any particular edit - the 'alien' edit in particular made people paler, but quickly started generating images which did not look human or alien. This behavior is to be expected for the more outlandish prompts, as the GAN has no means of generating such images. However, it seems as though the GAN is not capable of purely heavily relying on CLIP guidance to generate humanoid figures, and generally stays within a recognizable distribution of human faces before quickly diverging to abstract images.

5 Future improvements

5.1 Extend the model to different image contents

Our current project has been tested exclusively with the FFHQ dataset, which has allowed us to experiment only with semantic modifications on images of people. However, due to the structure of the model, it is easy to extend its use to work with different data sets, such as churches, horses, or cars. By adding this functionality, we extend the usefulness of our project, not only to generate people, but also to create different types of scenarios that may be required by different groups of users. We believe our project to be highly adaptable to these new image contents, provided the generator model and segmentation model display good synergy, and these variants could be easily deployed after some due process validation.

5.2 E4E Integration

To further improve our app, we plan to add an additional functionality based on Encoder4Editing (E4E) [12]. With this addition, we seek to create a more engaging app for users, giving them the ability to take their own photos and use our project to edit them to their preferences. This new functionality has the potential to generate greater public interest in the results we offer. By using E4E to invert images onto the latent space of StyleGAN2, we could import a user-provided image into the latent space, where we are capable of performing the aforementioned edits. We would have liked to include this functionality into our project before submitting it, but unfortunately ran out of time to implement and test it properly.

6 Conclusion

Despite its naive implementation, we have shown that our project can produce some very visually convincing results, and opens up new possibilities for using GANs and CLIP in everyday settings. We have developed a way to reliably find directions in the latent space which correspond to precise semantic concepts, and the structure of the space allows us to edit the images in an intuitive and straightforward way. Furthermore, our project quickly trains models which are capable of performing this task very well, on any image found within the latent space of the generator model, without requiring specialized hardware or software. The accessibility of this project and its potential highlight the power of GANs and show why they have captured the attention of researchers far and wide. Despite our project struggling with some prompts, as outlined above, and displaying some instability in its NLP components, we believe that further iterations on both the CLIP integration (for prompt engineering, or replacing CLIP with some other similar model) and the generalizability to other StyleGAN2-Segmentation model pairs could provide an easy, accessible way to leverage the power of GANs to perform a variety of image editing tasks for the everyday user, by standing on the shoulders of giants with far larger resources and computational power.

References

1. Andonian, A., Osmany, S., Cui, A., Park, Y., Jahanian, A., Torralba, A., Bau, D.: Paint by word (2021)
2. Brock, A., Donahue, J., Simonyan, K.: Large scale GAN training for high fidelity natural image synthesis. CoRR **abs/1809.11096** (2018), http://arxiv.org/abs/1809.11096
3. Härkönen, E., Hertzmann, A., Lehtinen, J., Paris, S.: Ganspace: Discovering interpretable GAN controls. CoRR **abs/2004.02546** (2020), https://arxiv.org/abs/2004.02546
4. Hermosilla, G., Henriquez, D.I., Allende-Cid, H., Farias, G., Vera, E.: Thermal face generation using stylegan. IEEE Access **PP**, 1–1 (06 2021). https://doi.org/10.1109/ACCESS.2021.3085423
5. Huang, X., Belongie, S.J.: Arbitrary style transfer in real-time with adaptive instance normalization. CoRR **abs/1703.06868** (2017), http://arxiv.org/abs/1703.06868
6. Karras, T., Laine, S., Aila, T.: A style-based generator architecture for generative adversarial networks. CoRR **abs/1812.04948** (2018), http://arxiv.org/abs/1812.04948
7. Karras, T., Laine, S., Aittala, M., Hellsten, J., Lehtinen, J., Aila, T.: Analyzing and improving the image quality of stylegan. CoRR **abs/1912.04958** (2019), http://arxiv.org/abs/1912.04958
8. Pajouheshgar, E., Zhang, T., Süsstrunk, S.: Optimizing latent space directions for gan-based local image editing. CoRR **abs/2111.12583** (2021), https://arxiv.org/abs/2111.12583
9. Radford, A., Kim, J.W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., Sutskever, I.: Learning transferable visual models from natural language supervision. CoRR **abs/2103.00020** (2021), https://arxiv.org/abs/2103.00020
10. Radford, A., Metz, L., Chintala, S.: Unsupervised representation learning with deep convolutional generative adversarial networks. In: Bengio, Y., LeCun, Y. (eds.) 4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings (2016), http://arxiv.org/abs/1511.06434
11. Shen, Y., Gu, J., Tang, X., Zhou, B.: Interpreting the latent space of gans for semantic face editing (2020)
12. Tov, O., Alaluf, Y., Nitzan, Y., Patashnik, O., Cohen-Or, D.: Designing an encoder for stylegan image manipulation (2021)