HairFastGAN: Realistic and Robust Hair Transfer with a Fast Encoder-Based Approach

Maxim Nikolaev[1,3], **Mikhail Kuznetsov**[1,2,3], **Dmitry Vetrov**[4], **Aibek Alanov**[1,3]

[1]HSE University, [2]Skolkovo Institute of Science and Technology, [3]AIRI

{m.nikolaev, m.k.kuznetsov, alanov}@airi.net

[4]Constructor University, Bremen

dvetrov@constructor.university

Figure 1: **HairFastGAN: Realistic and Robust Hair Transfer with a Fast Encoder-Based Approach.** *Our method takes as input a photo of the face, desired shape and hair color and then performs the transfer of the selected attributes. You can also see a comparison of our model with the others in the right plot. We were able to achieve excellent image realism while working in near real time.*

Abstract

Our paper addresses the complex task of transferring a hairstyle from a reference image to an input photo for virtual hair try-on. This task is challenging due to the need to adapt to various photo poses, the sensitivity of hairstyles, and the lack of objective metrics. The current state of the art hairstyle transfer methods use an optimization process for different parts of the approach, making them inexcusably slow. At the same time, faster encoder-based models are of very low quality because they either operate in StyleGAN's W+ space or use other low-dimensional image generators. Additionally, both approaches have a problem with hairstyle transfer when the source pose is very different from the target pose, because they either don't consider the pose at all or deal with it inefficiently. In our paper, we present the HairFast model, which uniquely solves these problems and achieves high resolution, near real-time performance, and superior reconstruction compared to optimization problem-based methods. Our solution includes a new architecture operating in the FS latent space of StyleGAN, an enhanced inpainting approach, and improved encoders for better alignment, color transfer, and a new encoder for post-processing. The effectiveness of our approach is demonstrated on realism metrics after random hairstyle transfer and reconstruction when the original hairstyle is transferred. In the most difficult scenario of transferring both shape and color of a hairstyle from different images, our method performs in less than a second on the Nvidia V100.

1 Introduction

Advances in the generation of face images using GANs [5, 12–14, 9, 31, 19, 11, 20, 26, 28] have made it possible to apply them to semantic face editing [10, 21, 34, 35]. One of the most challenging

and interesting topic in this area is hairstyle transfer [27]. The essence of this task is to transfer hair attributes such as color, shape, and structure from the reference photo to the input image while preserving identity and background. The understanding of mutual interaction of these attributes is the key to a quality solution of the problem. This task has many applications among both professionals and amateurs during work with face editing programs, virtual reality and computer games.

Existing approaches that solve this problem can be divided into two types: optimization-based [25, 41, 16, 15, 42, 33, 3], by obtaining image representations in some latent space of the image generator and directly optimizing it for the corresponding loss functions to transfer the hairstyle, and encoder-based [32, 13, 27, 7], where the whole process is done with a single direct pass through the neural network. The optimization-based methods have good quality but take too long, while the encoder-based methods are fast but still suffer from poor quality and low resolution. Moreover, both approaches still have a problem if the photos have a large pose difference.

We present a new HairFast method that works in high resolution, is outperforms in quality to state-of-the-art optimization methods, and is suitable for interactive applications in terms of speed, since we use only encoders in the inference process. We propose our decomposition of the problem and solve each of the subtasks efficiently. In particular, we developed a new approach for pose adaptation, a new approach for FS space regularization, a more efficient approach for hair coloring, and developed a new module for detail recovery. Our framework consists of four modules: pose alignment, shape alignment, color alignment and refinement alignment. Each module solves its own subtask by training specialized encoders.

We have conducted an extensive series of experiments, including attribute changes both individually (color, shape) and in combination (color and shape), on the CelebA-HQ dataset [12] in various scenarios. Based on standard realism metrics such as FID [8], FID_{CLIP} [18] and runtime, the proposed method shows comparable or even better results than state-of-the-art optimization-based methods while having inference time comparable to the fastest HairCLIP [32] method.

2 Related Works

GANs. Generative Adversarial Networks (GANs) have significantly advanced research in image generation, and recent models such as ProgressiveGAN [12], StyleGAN [13], and StyleGAN2 [14] produce highly detailed and realistic images, especially in the area of human faces. Despite the progress made in face generation, high-quality, fully controlled hair editing remains a challenge due to the many side effects.

Latent Space Embedding. Inversion techniques [1, 29, 40, 43, 23, 30] for StyleGAN generate latent representations that balance editability and reconstruction fidelity. Methods that prioritize editability map real images into a more flexible latent subspace, such as W or $W+$ [1], which can reduce the accuracy of the reconstruction - a popular example is E4E [30]. Reconstruction-focused methods, on the other hand, aim for an exact restoration of the original image. For example Barbershop merges the structural feature space (F) with the global style space (S) to form a composite space (FS). Such decomposition enhances the representational capacity of the space. Utilizing both the $W+$ and FS latent spaces, we have created a comprehensive hair editing framework that allows for a wide range of potential realistic adjustments.

Optimization-based methods. Among the classical optimization methods, we can highlight Barbershop [41], which uses multi-stage optimization in the StyleGAN FS space. But Barbershop doesn't work well with large pose differences, a problem that StyleYourHair [16] tries to solve by using local style matching and pose alignment loss, which allows efficient face rotation before hair transfer. Other approaches to hair editing include: StyleGANSalon [15], which solves the rotation problem with EG3D [2], HairNet [42], which has learned to handle complex poses and uses PTI [24] to improve quality but has lost the ability to independently transfer hair color, HairCLIPv2 [33] which can interact with images, masks, sketches and texts, HairNeRF [3] which uses StyleNeRF [6] instead of StyleGAN to provide distortion-free hair transfer in case of complex poses.

Encoder based methods. Encoder-based methods replace optimization processes with training a neural network, speeding up runtime a lot. Among the best models, we can highlight CtrlHair [7] that uses SEAN [44] as a feature encoder and generator. The method still suffers from complex cases
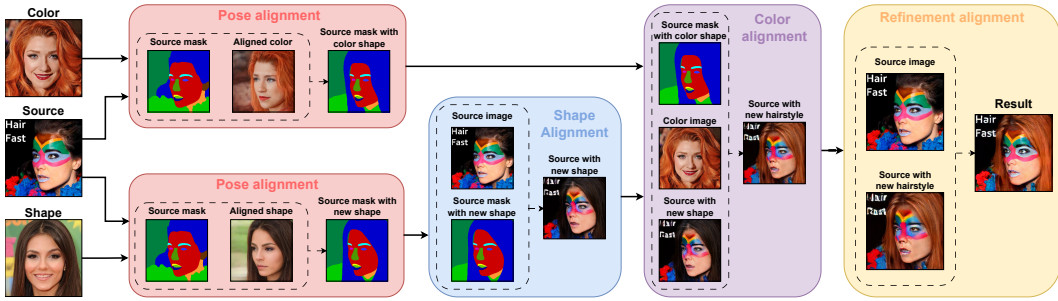
Figure 2: **Overview of HairFast:** the images first pass through the Pose alignment module, which generates a pose-aligned face mask with the desired hair shape. Then we transfer the desired hairstyle shape using Shape alignment and the desired hair color using Color alignment. In the last step, Refinement alignment returns the lost details of the original image where they are needed.

with different facial poses and the authors solve this by inefficient postprocessing of the mask due to which the method is slow. HairCLIP [32], which is an order of magnitude faster than CtrlHair, uses CLIP [22] feature extractor. The method allows to edit hair with text, but it works in $W+$ space, which causes poor preservation of face identity and hair texture. Other approaches to hair editing include: MichiGAN [27], HairFIT [4] Encoder-based methods show significantly worse performance than optimization-based approaches especially in difficult cases with different head poses and lightning conditions. We propose the first encoder-based framework that achieves comparable quality with methods that use optimizations.

3 Method

3.1 Overview

Formally we will solve the following problem, we have a source image I_{source} to which we want to transfer the style and shape of I_{shape}, and an image with the desired hair color I_{color}.

In this problem setting, we will solve the shape and color transfer problems independently using Shape alignment and Color alignment respectively to make the method flexible. Also, before these modules, we propose a new stage Pose alignment which purpose is to remove the pose mismatch between image I_{source} and images I_{shape} and I_{color}. And after all these modules we introduce a new Refinement alignment stage, which should restore the details of the face and background that we may have lost after the hair and color transfer stage. You can see the general pipeline of our approach on the picture Fig. 2.

In this work, we propose a unique solution for each of these modules that shows high quality and high performance compared to existing approaches. In particular, we do not use optimization to solve these problems, which allows us to speed up the overall pipeline greatly. In our HairFast method we propose an efficient Pose alignment approach by adding a new Rotate Encoder whose purpose is to change the face latent to rotate it. In Shape Alignment we propose a new FS mixing approach that allows to transfer the hair while keeping the possibility to edit it for color changes, besides we propose an efficient use of SEAN for inpainting at this stage. Also in Color Alignment we offer a new architecture using CLIP embeddings and new losses, which has significantly improved the results of this stage. Finally, we propose a completely new Refinement alignment stage.

First of all, our method starts with Pose alignment block, its purpose is to generate a segmentation mask with a target hair shape. This block takes as input images of the original face, the desired hair and their $W+$ representations in StyleGAN space. Then, a Rotate Encoder is run inside the block, which rotates the image with the desired hair to the same pose as the original face, followed by a Shape Encoder to adapt the resulting hairstyle at the level of the segmentation mask.

In the next Shape alignment module, we transfer the hairstyle shape from I_{shape} to I_{source} by changing only the tensor F from the FS space of StyleGAN. To do this, we generate the tensor F for the inpaint after changing the shape using SEAN and target mask from Pose alignment. Once we have all the necessary F tensors, we aggregate them taking into account the segmentation masks, selecting the desired parts, thus obtaining a new F tensor that corresponds to the image with the desired hair shape.

The next Color module is designed to transfer the hair color from the I_{color}. To do this, we edit the S space of the source image using our trained encoder, which also takes as input the S tensor of the reference and additional CLIP [22] embeddings of the source images.

(a) Visualization of the mixing block.　　　　(b) Modules for transferring the desired hair shape.
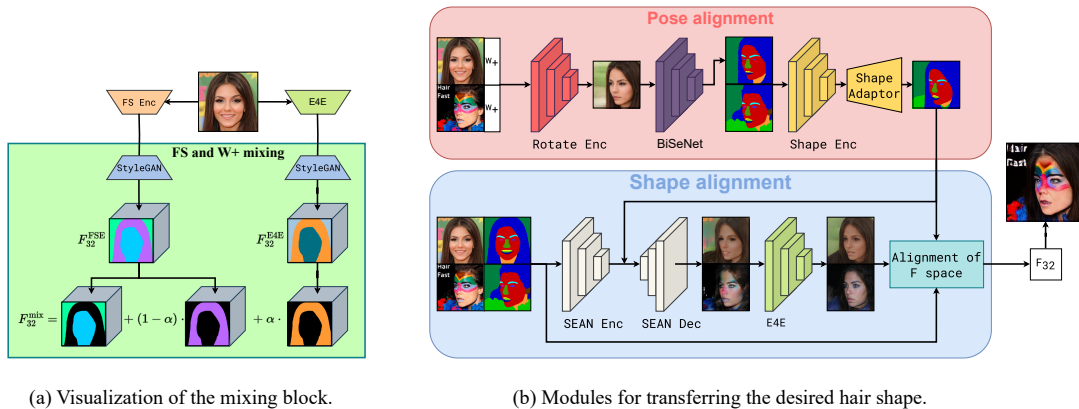
Figure 3: Detailed diagram of the units. (a) Mixing block mixes FS and W+ space representations to allow color editing (b) The Pose alignment module diagram generates a pose-aligned mask with the desired hair shape, and the Shape alignment module diagram that takes the images themselves, their segmentation masks, $W+$ and F representations to transfer the desired hairstyle shape.

The image generated after the Color module can already be considered as the final image, but in our work we also introduce a new Refinement alignment module. The purpose of this module is to restore the necessary details of the original image that were lost after inversion and editing. This allows us to preserve the identity of the face and increase the realism of the method.

3.2　Pose alignment

In this step, we want to get a segmentation mask of the original face with the desired hair shape. Such a task of generating a target mask without optimization problems was very successfully solved by the authors of the CtrlHair [7] method using Shape Encoder, which encodes the segmentation masks of two images as separate embeddings of hair and face, and Shape Adaptor reconstructs the segmentation mask of the desired face with the desired hair shape, additionally performing inpaint. Nevertheless, this approach still has problems.

First of all, the Shape Adaptor and Shape Encoder itself has been trained to transfer the hair shape as it is in the current pose and so for the case where the source and shape photos have too different poses, the method performs very poorly, causing the final photo to show severe hair shifts. The authors of CtrlHair have partially solved this problem with a slow and ineffective post-processing of the mask.

In our approach, we introduce a new Rotate Encoder that is trained to rotate the shape image to the same pose as the source image. This is accomplished by changing the latency of the image w^{E4E} received from the E4E encoder [30]. The new image is then segmented and given to the input of the encoder and shape adapter as the desired hair shape. Since we don't need detailed hair to generate the mask, we use the E4E representation of the image. The Rotate Encoder has been trained with very good interpolation and can rotate the image to the most complex pose while maintaining the original shape of the hair. At the same time, Encoder does not mess up the hairstyles if the image poses already match. After rotation, we obtain the image mask using BiSeNet [38]:

$$w_{\text{rotate}} = \text{Rotate}_{\text{Enc}}(w_{\text{source}}^{\text{E4E}},\ w_{\text{shape}}^{\text{E4E}}),\qquad M_{\text{rotate}} = \text{BiSeNet}(G(w_{\text{rotate}})),\qquad(1)$$

where G is StyleGAN and $w^{\text{E4E}} = \text{E4E}(I)$ – the latent E4E representation for I_{source} and I_{shape}.

For training Rotate Encoder, we used keypoint optimization with a pre-trained STAR [39] model as well as cycle-consistency reconstruction loss. See the Appendix 9.1 for more information about the Rotate Encoder.

This approach allows for a high quality transfer of most hairstyles even with the most complex pose differences, correcting artifacts that occur even in the StyleYourHair method, as will be shown in the experiments section. A diagram of the Pose alignment module architecture is shown in Fig. 3. More formally, the generation of the final $M_{\text{align}} = \text{Shape}_{\text{Adaptor}}(\text{hair}_{\text{emb}},\ \text{face}_{\text{emb}})$, where

$$\text{hair}_{\text{emb}} = \text{Shape}_{\text{Enc}}^{\text{hair}}(M_{\text{rotate}}),\quad \text{face}_{\text{emb}} = \text{Shape}_{\text{Enc}}^{\text{face}}(M_{\text{source}}),\quad M_{\text{source}} = \text{BiSeNet}(I_{\text{source}}).\quad(2)$$

The last step we extract the hair area mask H_{align} from M_{align}.

These masks H_{align} and M_{align} from the Pose alignment module will be needed for the next alignment tasks – inpaint generation and shape transfer in the Shape alignment module. A diagram of the Pose alignment module is shown in Fig. 3.

4

3.3 FS and W+ mixing

Before transferring the new hairstyle and doing coloring, there's one more problem to address. The native integration of the FS encoder [36] cannot edit S space in a way that transfers hair color. To solve this problem, we additionally use E4E – it is a very simple encoder with relatively poor image reconstruction quality, but has high editability. For this, we also reconstruct all images with E4E and mix the F tensor corresponding to the hair with the F tensor obtained with the FS encoder.

Formally, if I is input image, then F_{16}^{FSE}, $S = \text{FS}_{\text{Enc}}(I)$, $w^{\text{E4E}} = \text{E4E}(I)$, where $F_{16}^{\text{FSE}} \in \mathbb{R}^{16 \times 16 \times 512}$, $S \in \mathbb{R}^{12 \times 512}$ – the FS representation obtained from FS encoder and $w^{\text{E4E}} \in \mathbb{R}^{18 \times 512}$ – the encoding from E4E.

Since we want to edit images in F space of 32x32 resolution while FS encoder produces only 16x16, we need to run a few more StyleGAN blocks, while for E4E we run all 6 first blocks:

$$F_{32}^{\text{FSE}} = G_{4:6}(F_{16}^{\text{FSE}}, S), \qquad F_{32}^{\text{E4E}} = G_6(w^{\text{E4E}}), \tag{3}$$

where G_6 – the output of the first 6 StyleGAN blocks and $G_{4:6}$ – the generator starts at block 4.

To find the hair region in the F tensor, we use BiSeNet to segment a face and downsize the selected hair mask. Then final reconstruction for images F_{32}^{mix}:

$$M = \text{BiSeNet}(I), \quad H = \text{Downsample}_{32}(M = \text{hair}), \tag{4}$$

$$F_{32}^{\text{mix}} = \overline{H} \cdot F_{32}^{\text{FSE}} + (1 - \alpha) \cdot H \cdot F_{32}^{\text{FSE}} + \alpha \cdot H \cdot F_{32}^{\text{E4E}}. \tag{5}$$

Here $\overline{H}$ is an inversion of the mask H and α is the hyperparameter for mixing. In our work it is equal to 0.95. This means that we only take 5% of the hair from FS encoder, but according to our experiments, even this small mixing with the hair F tensor of FS encoder hair greatly increases the quality. The visualization of the mixing procedure shown in Fig. 3.

This F_{32}^{mix} tensor allows us to get an excellent quality of face and background reconstruction and still edit the hairstyle.

3.4 Shape alignment

In this step, our goal is to transfer the desired hair shape from I_{shape} to I_{source}. For this purpose, we edit only the F space. To achieve this, we solve 2 subtasks: generation of a target mask with the desired hair shape and generation of F tensor with the inpainted parts of the image.

For the inpaint task, we use the pre-trained SEAN model, which produces style vectors for each segmentation class using the input image and its segmentation mask, and its decoder reconstructs the image using the style vectors and any new segmentation mask. Thus, using this model, we can obtain a 256x256 resolution image with the desired hair shape for both source and shape photos.

$$\text{style}_{\text{codes}} = \text{SEAN}_{\text{Enc}}(I, M), \qquad I^{\text{inpaint}} = \text{SEAN}_{\text{Dec}}(\text{style}_{\text{codes}}, M_{\text{align}}). \tag{6}$$

To get the F tensor representation of these images we use E4E. According to our experiments, the SEAN model in some cases produces strong artifacts in weakly represented segmentation classes such as ears, and due to artifacts on the target segmentation mask, it can also produce images with similar artifacts, such as when the hair is not connected to the head. E4E due to its good generalization is a good regularizer that automatically handles all such kind of artifact.

$$F_{32}^{\text{inpaint}} = G_6(\text{E4E}(I^{\text{inpaint}})). \tag{7}$$

In the current step we have two initial F_{32}^{mix} tensors of images and two F_{32}^{inpaint} tensors after the inpaint part. The last step Alignment of F space combines all four F tensors into one new F_{32}^{align}, which can be used to generate an image with a given hairstyle. To do this, we assemble the tensor by selecting its corresponding parts using segmentation masks. A diagram of the Shape alignment module is shown in Fig. 3.

$$F_{32}^{\text{align}} = H_{\text{align}} \cdot H_{\text{shape}} \cdot F_{\text{shape}}^{\text{mix}} + H_{\text{align}} \cdot \overline{H_{\text{shape}}} \cdot F_{\text{shape}}^{\text{inpaint}} + \\ + \overline{H_{\text{align}}} \cdot \overline{H_{\text{source}}} \cdot F_{\text{source}}^{\text{mix}} + \overline{H_{\text{align}}} \cdot H_{\text{source}} \cdot F_{\text{source}}^{\text{inpaint}}. \tag{8}$$

Here, H_{shape} and H_{source} are obtained from I_{shape} and I_{source} from according to Eq. (4).

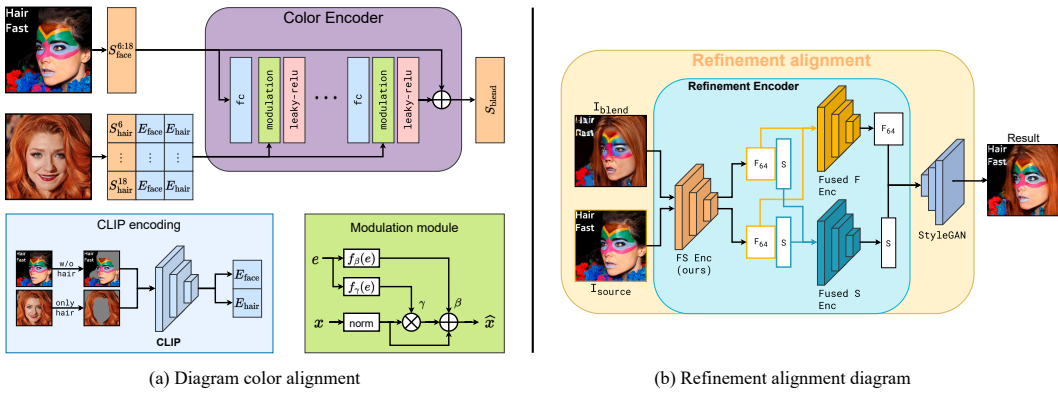(a) Diagram color alignment (b) Refinement alignment diagram

Figure 4: Detailed diagram of the units. (a) A color alignment module diagram that takes as input S image representations as well as segmentation masks. The purpose of this block is to encode the details of the original image and change the S space to transfer the desired hair color and preserve the identity. (b) A refinement alignment diagram that takes as input the source image and post Color alignment module image. At this module, the goal is to get a new representation in StyleGAN space to get a realistic image, with the original details of the source image that were lost after inverting images into latents.

3.5 Color alignment

In the next step, we solve the problem of changing the S space so as to change the hair color to the desired color. According to our experiments, we find that learning the convex combination of S_{source} and S_{color} as previous methods did is not enough for good quality, so we develop a new encoder to predict the change of S_{source}. Moreover, we find that using only S space is not enough to change hair color and preserve the rest of the image, so we include CLIP image embeddings to bring more information into the features. Finally, we experiment with losses and find that the cosine distance between CLIP embeddings works better than LPIPS [36].

As Color Encoder architecture we use 1D modulation layers similar to those used in StyleGAN. Such layers are excellent for style changing and have good stability. The purpose of this encoder is to predict the change of S_{source} in such a way as to change the hair color to the desired color while preserving the rest of the image. We input S_{color} as a style to the modulation layers with CLIP embeddings of the source image without hair: $\text{emb}_{\text{face}} = \text{CLIP}_{\text{enc}}(I_{\text{source}} \cdot \overline{H_{\text{align}}} \cdot \overline{H_{\text{source}}})$ and CLIP embeddings of the hair only from the color image: $\text{emb}_{\text{hair}} = \text{CLIP}_{\text{enc}}(I_{\text{color}} \cdot H_{\text{color}})$, where H_{align} hair mask obtained from a Pose alignment module run to transfer a hair shape from I_{color} to I_{source}. This helps to convey additional information about the original images that might have been lost after inverting images into latent spaces. And I_{blend} is the final image before Refinement alignment stage:

$$S_{\text{blend}} = \text{Blend}_{\text{Enc}}(S_{\text{source}}, \ S_{\text{color}}, \ \text{emb}_{\text{face}}, \ \text{emb}_{\text{hair}}), \tag{9}$$

$$I_{\text{blend}} = \text{StyleGAN}(F_{32}^{\text{align}}, \ S_{\text{blend}}), \tag{10}$$

A diagram of the method is shown in Fig. 4.

$$\mathcal{L}_{\text{clip}}(I_1, I_2, M_1, M_2) = 1 - \text{CosSim}_{\text{CLIP}}(I_1 \cdot M_1, I_2 \cdot M_2), \tag{11}$$

To train the model, we use the $\mathcal{L}_{\text{clip}}$ one of which optimizes the cosine distance between the CLIP embeddings of the final and source images on both the reconstruction and transfer of the desired color. See the Appendix 9.2 for training and encoder details.

3.6 Refinement alignment

Our method, even though it has a higher quality Color alignment step, still has a problem on complex cases where the face hue may change. Particularly because of this we cannot simply use Poisson blending like a CtrlHair, as the difference in shades emphasizes the overlay more and visually it doesn't look realistic.

For this reason, we are developing our own Refinement alignment module, which is essentially a larger and more powerful reconstruction encoder, but for a more complex task – reconstruction of the original face and background, reconstruction of hair after Color alignment module and inpaint of non-matching parts. This Refinement encoder generates an F tensor 4 times higher resolution than the FS encoder we used in the Shape alignment module. This allows for unrivaled reconstruction quality. Unlike traditional encoders that sacrifice reconstruction quality for good editing, we are able to use such a large resolution F tensor due to the fact that we do not have to edit the image after this module.

6

Table 1: **Realism Metrics**. These metrics were measured on the same pre-selected triples of images (face, shape and color) from the CelebaHQ [12] dataset. Then, applying the method, FID was measured on the original dataset and the modified dataset. FID_{CLIP} [18] was counted similarly to FID, but a CLIP encoder was used instead of Inception V3. Running time was measured as the median time among a bunch of method runs, without taking into account loading images from disk. **Pose Metrics**. For this metrics, we consider color and shape transfer from the target image to the source image. We divided all pairs into 3 equal buckets: easy, medium and hard according to the difference of face key points. **Reconstruction**. For this each method is started on the task of transferring the color and shape of the hairstyle from itself to itself, thus at the end we measure the metrics with the original image.

Model	Realism metrics				Pose metrics				Reconstruction		Time (s)↓	
	FID↓		FID_{CLIP}↓		FID↓		FID_{CLIP}↓		LPIPS↓	PSNR↑		
	full	both	full	both	medium	hard	medium	hard			A100	V100
HairCLIP [32]	34.95	40.68	12.20	13.32	55.77	54.35	15.53	15.73	0.36	14.08	**0.28**	**0.36**
HairCLIPv2 [33]	<u>14.28</u>	<u>23.37</u>	10.98	12.14	44.62	<u>45.28</u>	14.56	18.66	0.16	19.71	112	221
CtrlHair [7]	15.10	24.81	9.52	10.42	46.45	50.12	12.96	16.42	0.15	19.96	6.57	7.87
StyleYourHair [16]	-	25.90	-	10.91	46.32	47.19	13.70	15.93	0.14	<u>21.74</u>	84	239
Barbershop [41]	15.94	24.52	<u>7.07</u>	<u>8.12</u>	<u>44.08</u>	46.13	<u>11.27</u>	<u>13.30</u>	<u>0.11</u>	21.18	213	645
HairFast (ours)	**13.12**	**22.71**	**5.12**	**6.06**	**43.25**	**44.85**	**8.90**	**10.33**	**0.08**	**23.45**	<u>0.41</u>	<u>0.78</u>

Refinement itself consists of a trained FS encoder at resolution 64 for the usual reconstruction task, we use it to encode the original image I_{source} and the I_{blend} image after our method:

$$F_{64}^{\text{blend}}, \ S^{\text{blend}} = FS_{\text{Enc (ours)}}(I_{\text{blend}}), \tag{12}$$

$$F_{64}^{\text{source}}, \ S^{\text{source}} = FS_{\text{Enc (ours)}}(I_{\text{source}}). \tag{13}$$

The resulting tensors F are fused using IResNet blocks. In turn, S space is fused using two similar Color Encoder models, but without additional CLIP features. The output of this composite encoder is $F_{64}^{\text{final}} = \text{Fused}_{\text{F Enc}}(F_{64}^{\text{blend}}, F_{64}^{\text{source}})$ tensor and an $S_{\text{final}} = \text{latent}_{\text{avg}} + \text{Fused}_{\text{S Enc}}(S^{\text{blend}}, S^{\text{source}})$ vector, which are input to StyleGAN to generate the final image I_{final}:

$$I_{\text{final}} = \text{StyleGAN}(F_{64}^{\text{final}}, \ S_{\text{final}}). \tag{14}$$

For model training, we use loss functions for hair reconstruction and original parts of the image, and for inpaint we use guidelines from the more robust StyleGAN space and adversarial loss. For reconstruction, these include multi-scale perceptual loss [36], DSC++ [37], ArcFace and regularizations. See the Appendix 9.3 for training and encoders details.

A diagram of the Refinement procedure is shown in the Fig. 4. This produces the final HairFast model, which is shown in the Fig. 2.

4 Experiments

Realism after editing. The task of hairstyle transfer is very challenging, largely due to the lack of objective metrics. One possible metric to reflect the quality of hairstyle transfer is to measure the realism of the image using FID. To measure this metric, we consider 4 main cases: transferring hairstyle and color from different images (full), transferring only a new hairstyle shape (shape), transferring only a new color (color), and transferring both color and shape from the same image (both). To measure the metrics, we use the CelebA-HQ [12] dataset, from which we capture 1000 to 3000 experiments for each case, on which we run all methods. We used methods such as HairCLIP [32], HairCLIPv2 [33], CtrlHair [7], StyleYourHair [16] and Barbershop [41] for comparison, and for their inference we used the official code implementation. Additionally, we measure the median running time among all runs of these experiments, excluding the time to save the results to disk and initialize the neural networks. In the Table 1 you can observe the results of this experiment for the "full" and "both" cases, the complete table can be seen in the Appendix 12.

In these experiments, we do not compare with HairNet [42], HairNeRF [3] and StyleGANSalon [15] due to the lack of their code and the inability to run the methods on our images. Instead, we compare with StyleGANSalon on images they published from their inference along with LOHO [25] in Appendix 13, and we also make a visual comparison with HairNet and HairNeRF on images from their article in Appendix 14.

As we can see in the Table 1, a method like CtrlHair outperforms optimization-based methods like Barbershop and StyleYourHair by FID metrics. However, visual analysis reveals that the method performs much worse and artifacts are visible, which appear as a consequence of strong Poisson Blending of the final image with the original image. The authors in [18] studied the problem that makes images with strong artifacts appear more realistic by FID metric. They were able to solve this

Table 2: A comparison of the characteristics of the main hair transfer methods.

	Barbershop [41]	StyleYourHair [16]	HairNet [42]	HairNeRF [3]	StyleSalon [15]	CtrlHair [7]	HairCLIP [32]	HairCLIPv2 [33]	Ours
Quality									
Hair realism	High	High	High	High	High	Medium	Low	Medium	High
Face-background preservation	Medium	Medium	Medium	High	High	High	Low	Medium	High
Functionality									
Pose alignment	✗	✓	✓	✓	✓	✗	✗	✓	✓
Separate shape/color transfer	✓	✗	✗	✗	✗	✓	✓	✓	✓
Efficiency									
W/o optimization	✗	✗	✗	✗	✗	✓	✓	✗	✓
Runtime	>10m	>3m	>3m	>3m	>10m	>5s	<1s	>3m	<1s
Reproducibility									
Code access	✓	✓	✗	✗	✗	✓	✓	✓	✓

Figure 5: Visual comparison of methods on different cases for transferring hair and color together, or separately. StyleYourHair transfers color only from the Shape image. According to the results of visual comparison, our model better preserves the identity of the source image. At the same time, our method on most cases better transfers the desired hair color and texture, and works better with complex pose differences. For a more detailed comparison, see Appendix 17.

problem by using the FID_{CLIP} metric, which simply uses higher quality embeddings from the CLIP model. We also compute this metric in our experiments. Note that the metric uses the CLIP-ViT-B-32 checkpoint while we use CLIP-VIT-B-16 for color encoder training, so there is no leakage in our measurements.

Analyzing the results, our method performs better on all metrics. Looking at runtime, we outperform Barbershop on V100 by a factor of 800 and even CtrlHair by more than 10 times. This is because CtrlHair has an expensive post-processing implementation for alignment and Poisson blending. The only method that is faster is HairCLIP, but its performance in our problem setup is quite poor.

Pose difference. Table 1 shows the results of the metrics on a subsample of our main experiment "both", but split into different cases of pose difference. For this purpose, we counted the RMSE of key points of the source image with the shape image and split all cases of hairstyle transfer into 3 equal folds: easy, medium and hard. The last two cases are presented in the table. The full table can be seen in the Appendix 12.

Reconstruction. Another quality metric can be the reconstruction metric, where each method tries to transfer the shape and color of the hairstyle from itself, in this case we have a ground truth image with which we can measure the metrics. For this metric, we measure the LPIPS and PSNR between the original images and the resulting images. 3000 random images from CelebA-HQ were taken for reconstruction. Additional metrics for reconstruction such as FID and FID_{CLIP} can be viewed in the Appendix 12.

Analyzing the results Table 1, we also outperform other methods. This confirms the effectiveness of our Refinement alignment stage, which recovers lost image details during encoding, outperforming even optimization-based methods.

	Configuration	FID↓	FID$_{CLIP}$↓	Time (s)↓
A	Baseline	16.23	6.92	0.67
B	w/o Color alignment	26.88	11.45	39
C	w/o F mix.	16.57	6.72	0.67
D	config B w/o F mix.	27.74	11.51	39
E	w/o Rotate Encoder	16.87	7.52	**0.62**
F	w/o Shape Adaptor	18.72	6.37	37
G	w/o SEEN inpaint	**12.79**	**4.58**	92
H	+ Refinement (ours)	13.12	5.12	0.78

Table 3: Ablation results. Baseline is HairFast, but without Refinement alignment. In each configuration, we replace the specified part with optimizations from Barbershop as needed.

Figure 6: Ablation Study for different configurations of our model. Our model is used as the Baseline, but without Refinement alignment. Each column represents a change in the Baseline of the model.

Overall comparison. The Table 2 shows a comparison of the characteristics of the methods. Hair realism was determined according to realism metrics on reconstruction tasks and visual comparison. HairCLIPv2 has medium realism because of poor reconstruction, which due to the peculiarities of the architecture does not allow to transfer the desired texture accurately enough, in turn, CtrlHair despite the excellent metrics in visual comparison shows not similar to the desired results due to the limitations of the generator. The other methods, except HairCLIP, transfer the hairstyle realistically.

When it comes to preserving face and background details, the latent space of methods in which image inversions take place is mainly responsible for this. Methods such as Barbershop, StyleYourHair, HairNet and HairCLIPv2 use FS resolution space 32, which does not allow them to preserve much details. In turn, the HairNeRF and StyleGANSalon methods use PTI, which allows them to preserve more details of the original image, and the CtrlHair method uses Poisson Blending, which also allows direct transfer of all original details. Our method uses FS resolution space 64, which when compared visually and reconstruction metrics shows even better quality than methods with PTI. HairCLIP, in contrast, uses the weakest W+ space.

The runtime of each method that has a code we tested on the Nvidia V100. StyleGANSalon's time estimate came from their article, where they claim to run longer than Barbershop, while methods like HairNet and HairNeRF use PTI which makes them take at least a few minutes per image.

Looking at the rest of the features, unlike some other methods we are able to transfer hair color and shape independently, we are also able to handle large pose differences and our entire architecture consists of encoders, which allows us to work very fast. Moreover our method has code for inference, all pre-trained weights and scripts for training for full reproducibility.

Ablation study. As ablation, we remove some parts of our method and replace them with Barbershop optimization processes if necessary. On these configurations we measure the realism metrics after the hairstyle transfer, the results of which can be seen in the table Tab. 3 and images Fig. 6.

By ablation, we proved the high quality of Color Encoder, the necessity of mixing FS and W+ spaces, the effectiveness of Rotate Encoder, Shape Adaptor, SEEN, and the effectiveness of Refinement alignment module. For more detailed ablation conclusions, see Appendix 7.

Failure cases. The main problems of our method arise in the inpaint part, it may not work well when long hair is replaced by short hair. Also, our method suffers from transferring hair with complex textures such as ponytails, ribbons and braids. While these problems are important, they are inherent in all baseline models and we will address them in our future work. A detailed analysis of failure cases can be seen in the Appendix 8.

5 Conclusion and Limitations

In this article, we introduced the new HairFast method for hair transfer. Unlike other approaches, we were able to achieve high quality and high resolution outperforms to other optimization-based methods, but still working in near real time. We developed a new approach for pose adaptation, a new approach for FS space regularization, a more efficient approach for hair coloring, and developed a new module for detail recovery.

But our method, like many others, is limited by the small number of ways to transfer hairstyles, but our architecture allows to fix this in future work. For example, our Color alignment module architecture allows similarly to HairCLIP to do hair color editing with text, and using Shape Adaptor allows similarly to CtrlHair to edit hair shape with sliders.

We prove the effectiveness of our approach by comparing it with other methods in the Section 4, and refer to additional experiments in the Appendix 11, 12, 13, 14, 15, 16 and 17 for further details.

6 Acknowledgments

References

[1] Rameen Abdal, Yipeng Qin, and Peter Wonka. Image2stylegan: How to embed images into the stylegan latent space? In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4432–4441, 2019.

[2] Eric R Chan, Connor Z Lin, Matthew A Chan, Koki Nagano, Boxiao Pan, Shalini De Mello, Orazio Gallo, Leonidas J Guibas, Jonathan Tremblay, Sameh Khamis, et al. Efficient geometry-aware 3d generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16123–16133, 2022.

[3] Seunggyu Chang, Gihoon Kim, and Hayeon Kim. Hairnerf: Geometry-aware image synthesis for hairstyle transfer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2448–2458, 2023.

[4] Chaeyeon Chung, Taewoo Kim, Hyelin Nam, Seunghwan Choi, Gyojung Gu, Sunghyun Park, and Jaegul Choo. Hairfit: pose-invariant hairstyle transfer via flow-based hair alignment and semantic-region-aware inpainting. *arXiv preprint arXiv:2206.08585*, 2022.

[5] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[6] Jiatao Gu, Lingjie Liu, Peng Wang, and Christian Theobalt. Stylenerf: A style-based 3d-aware generator for high-resolution image synthesis, 2021.

[7] Xuyang Guo, Meina Kan, Tianle Chen, and Shiguang Shan. Gan with multivariate disentangling for controllable hair editing. In *European Conference on Computer Vision*, pages 655–670. Springer, 2022.

[8] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017.

[9] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017.

[10] Wentao Jiang, Si Liu, Chen Gao, Jie Cao, Ran He, Jiashi Feng, and Shuicheng Yan. Psgan: Pose and expression robust spatial-aware gan for customizable makeup transfer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5194–5202, 2020.

[11] Youngjoo Jo and Jongyoul Park. Sc-fegan: Face editing generative adversarial network with user's sketch and color. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 1745–1753, 2019.

[12] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017.

[13] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4401–4410, 2019.

[14] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8110–8119, 2020.

[15] Sasikarn Khwanmuang, Pakkapon Phongthawee, Patsorn Sangkloy, and Supasorn Suwajanakorn. Stylegan salon: Multi-view latent optimization for pose-invariant hairstyle transfer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8609–8618, 2023.

[16] Taewoo Kim, Chaeyeon Chung, Yoonseo Kim, Sunghyun Park, Kangyeol Kim, and Jaegul Choo. Style your hair: Latent optimization for pose-invariant hairstyle transfer via local-style-aware hair alignment. In *European Conference on Computer Vision*, pages 188–203. Springer, 2022.

[17] PS Kostenetskiy, RA Chulkevich, and VI Kozyrev. HPC resources of the higher school of economics. In *Journal of Physics: Conference Series*, volume 1740, page 012050, 2021.

[18] Tuomas Kynkäänniemi, Tero Karras, Miika Aittala, Timo Aila, and Jaakko Lehtinen. The role of imagenet classes in fréchet inception distance. In *Proc. ICLR*, 2023.

[19] Cheng-Han Lee, Ziwei Liu, Lingyun Wu, and Ping Luo. Maskgan: Towards diverse and interactive facial image manipulation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5549–5558, 2020.

[20] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic image synthesis with spatially-adaptive normalization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2337–2346, 2019.

[21] Tiziano Portenier, Qiyang Hu, Attila Szabo, Siavash Arjomand Bigdeli, Paolo Favaro, and Matthias Zwicker. Faceshop: Deep sketch-based face image editing. *arXiv preprint arXiv:1804.08972*, 2018.

[22] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.

[23] Elad Richardson, Yuval Alaluf, Or Patashnik, Yotam Nitzan, Yaniv Azar, Stav Shapiro, and Daniel Cohen-Or. Encoding in style: a stylegan encoder for image-to-image translation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2287–2296, 2021.

[24] Daniel Roich, Ron Mokady, Amit H Bermano, and Daniel Cohen-Or. Pivotal tuning for latent-based editing of real images. *ACM Transactions on graphics (TOG)*, 42(1):1–13, 2022.

[25] Rohit Saha, Brendan Duke, Florian Shkurti, Graham W Taylor, and Parham Aarabi. Loho: Latent optimization of hairstyles via orthogonalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1984–1993, 2021.

[26] Zhentao Tan, Menglei Chai, Dongdong Chen, Jing Liao, Qi Chu, Bin Liu, Gang Hua, and Nenghai Yu. Diverse semantic image synthesis via probability distribution modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7962–7971, 2021.

[27] Zhentao Tan, Menglei Chai, Dongdong Chen, Jing Liao, Qi Chu, Lu Yuan, Sergey Tulyakov, and Nenghai Yu. Michigan: multi-input-conditioned hair image generation for portrait editing. *arXiv preprint arXiv:2010.16417*, 2020.

[28] Zhentao Tan, Dongdong Chen, Qi Chu, Menglei Chai, Jing Liao, Mingming He, Lu Yuan, Gang Hua, and Nenghai Yu. Efficient semantic image synthesis via class-adaptive normalization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(9):4852–4866, 2021.

[29] Ayush Tewari, Mohamed Elgharib, Florian Bernard, Hans-Peter Seidel, Patrick Pérez, Michael Zollhöfer, and Christian Theobalt. Pie: Portrait image embedding for semantic control. *ACM Transactions on Graphics (TOG)*, 39(6):1–14, 2020.

[30] Omer Tov, Yuval Alaluf, Yotam Nitzan, Or Patashnik, and Daniel Cohen-Or. Designing an encoder for stylegan image manipulation. *ACM Transactions on Graphics (TOG)*, 40(4):1–14, 2021.

[31] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8798–8807, 2018.

[32] Tianyi Wei, Dongdong Chen, Wenbo Zhou, Jing Liao, Zhentao Tan, Lu Yuan, Weiming Zhang, and Nenghai Yu. Hairclip: Design your hair by text and reference image. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18072–18081, 2022.

[33] Tianyi Wei, Dongdong Chen, Wenbo Zhou, Jing Liao, Weiming Zhang, Gang Hua, and Nenghai Yu. Hairclipv2: Unifying hair editing via proxy feature blending, 2023.

[34] Chufeng Xiao, Deng Yu, Xiaoguang Han, Youyi Zheng, and Hongbo Fu. Sketchhairsalon: Deep sketch-based hair image synthesis. *arXiv preprint arXiv:2109.07874*, 2021.

[35] Shuai Yang, Zhangyang Wang, Jiaying Liu, and Zongming Guo. Deep plastic surgery: Robust and controllable image editing with human-drawn sketches. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XV 16*, pages 601–617. Springer, 2020.

[36] Xu Yao, Alasdair Newson, Yann Gousseau, and Pierre Hellier. A style-based gan encoder for high fidelity reconstruction of images and videos. *European conference on computer vision*, 2022.

[37] Michael Yeung, Leonardo Rundo, Yang Nan, Evis Sala, Carola-Bibiane Schönlieb, and Guang Yang. Calibrating the dice loss to handle neural network overconfidence for biomedical image segmentation. *Journal of Digital Imaging*, 36(2):739–752, 2023.

[38] Changqian Yu, Jingbo Wang, Chao Peng, Changxin Gao, Gang Yu, and Nong Sang. Bisenet: Bilateral segmentation network for real-time semantic segmentation. In *European Conference on Computer Vision*, pages 334–349. Springer, 2018.

[39] Zhenglin Zhou, Huaxia Li, Hong Liu, Nanyang Wang, Gang Yu, and Rongrong Ji. Star loss: Reducing semantic ambiguity in facial landmark detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15475–15484, June 2023.

[40] Jiapeng Zhu, Yujun Shen, Deli Zhao, and Bolei Zhou. In-domain gan inversion for real image editing. In *European conference on computer vision*, pages 592–608. Springer, 2020.

[41] Peihao Zhu, Rameen Abdal, John Femiani, and Peter Wonka. Barbershop: Gan-based image compositing using segmentation masks. *arXiv preprint arXiv:2106.01505*, 2021.

[42] Peihao Zhu, Rameen Abdal, John Femiani, and Peter Wonka. Hairnet: Hairstyle transfer with pose changes. In *European Conference on Computer Vision*, pages 651–667. Springer, 2022.

[43] Peihao Zhu, Rameen Abdal, Yipeng Qin, John Femiani, and Peter Wonka. Improved stylegan embedding: Where are the good latents? *arXiv preprint arXiv:2012.09036*, 2020.

[44] Peihao Zhu, Rameen Abdal, Yipeng Qin, and Peter Wonka. Sean: Image synthesis with semantic region-adaptive normalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5104–5113, 2020.

Appendix

In this appendix, we provide additional explanations, experiments, and results:

7 Ablation detail

7.1 Color Encoder and FS space mixing

To prove the effectiveness of our Color Encoder and the necessity of mixing F spaces, we perform 4 experiments: configurations A – D in Table 3, in which we measure metrics on 1000 random triples from the CelebA-HQ dataset.

We first compare Baseline (config A) with configurations without mixing F spaces (config C), examining the results of the metrics we see that using full F space does not give a statistically significant gain in realism. But more importantly, if we examine the visual comparison Fig. 6, we see that F space without mixing does not allow us to edit the hair color with our Color Encoder.

Similarly, we experiment with config B, in which we replace the Color Encoder using the Barbershop optimization process, and config D, in which we additionally remove the mixing of F spaces. From the metrics results, we can see that using such an optimization process significantly degrades the realism metrics, which proves the effectiveness of our approach. Moreover, on the visual comparison we see that config B can still edit the hair and get the desired hair color, but for config D we see the same problem where even the optimization process cannot get the desired hair color, which confirms the need to use F space mixing.

7.2 Rotate Encoder

Next experiment config E, we remove the Rotate Encoder from our model, essentially leaving the generation of the targeting mask as in the CtrlHair method. In terms of metrics, removing the Rotate Encoder results in reduced realism in both FID and FID_{CLIP}. Moreover, when visually comparing ablations Fig. 6 we see 2 problems in this approach, which were described in detail in the Barbershop article: in case of a strong pose difference either vertically or horizontally, the hair mask adapts directly, which leads to severe distortions in the final image, but Rotate Encoder allows us to eliminate them effectively.

To verify that Rotate Encoder does not mess up the desired hair shape we additionally perform a 1000 image reconstruction experiment where we transfer the hair shape and color from the image itself to itself. For this, we compute the IOU metric on the hair mask of the original and the resulting Baseline configuration image and the config E image without Rotate Encoder. As a result, the use of Rotate Encoder led to a relative decrease in IOU by 2.7%, compared to the model without the encoder. This is a very small difference compared to the improvements seen even with small pose differences, including in Fig. 6.
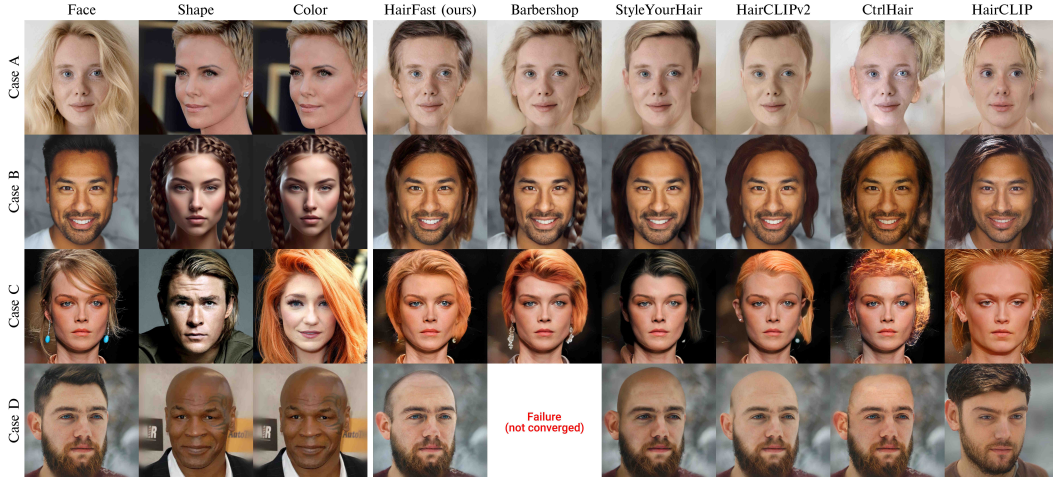
Figure 7: Failed cases. Case A poor color transfer and unrealistic inpaint. Case B unsuccessful transfer of complex texture. Case C earrings did not transfer. Case D unwanted hair in bald hairstyle transfer.

7.3 Pose and Shape alignment

We also run a series of experiments in which we remove one of our encoders each and replace it with a similar optimization problem from Barbershop, configuration F-G.

Analyzing the metrics of experiments E and F, we see a serious gap in all metrics, not to mention the running time due to the optimization processes.

If we consider the config G, the metrics get much better, but by visually evaluating the results in Fig. 6, we see that this optimization process strongly regularizes the desired hair shape and transfers quite differently than expected. In addition, this optimization process is very long.

This also confirms the effectiveness of our alignment step for both the task of target mask generation and inpaint not only in terms of performance but also quality.

7.4 Refinement alignment

The last experiment config H we consider is our own HairFast model, which is Baseline (config A) with added Refinement alignment. In addition to significant gains in metrics, we can also observe its effectiveness on visual comparison. Such Post-Processing effectively fixes Color Encoder problems, for example, in cases where it cannot preserve the original face hue due to the need to change the hair color as happened in the last example Fig. 6. It also effectively reverts identity and fine details such as piercings, makeup and others.

8 Failure cases

The Fig. 7 shows examples of cases where our method does not work correctly. In particular, inpaint may not work very well for cases where long hair is replaced by short hair, in part by creating an unrealistic skin texture, or by creating shadows from past hair as in Figure 7A. In some cases, color reproduction with color encoder does not work perfectly, in particular, the problem may occur when there is a large difference in illumination, an example of failed color reproduction is shown in Figure 7A. Also, our approach does not allow to transfer hairstyles with complex textures, such as ponytails, ribbons, braids as in the figure 7B. In addition, the model may have a problem retaining the original attributes that are exposed in the Alignment step, in which case the Refinement alignment may not be able to restore them (Figure 7C). Furthermore, our method may fail to remove some of the hair when transferring a bald hairstyle (Figure 7D). While these problems are important, they are inherent in all baseline models and we will address them in our future work.

9 Model Training

9.1 Rotate Encoder

Figure 8: Demonstration of how Rotate Encoder works to rotate an image to generate the correct segmentation mask for complex pose difference images.

To train the Rotate Encoder, we collected 10'000 random images from FFHQ to learn how to rotate in W+ space from the E4E encoder to obtain an image with the desired pose. During the rotation, we must be able to preserve the shape of the hair.

The Rotate Encoder itself takes as input the first 6 vectors corresponding to the first StyleGAN blocks in W+ space of the source image and 6 vectors of the target image with the desired pose. And on the output it returns new 6 vectors, which should generate the image with the desired pose. We do not modify the remaining 18 - 6 vectors, which also helps us to preserve the identity in some details. An example of how Rotate Encoder works is shown in the image Fig. 8.

Formally, this can be described as follows:

$$w_{\text{rotate}}^{1:6} = \text{Rotate}_{\text{Enc}}(w_{\text{source}}^{1:6}, w_{\text{target}}^{1:6}), \tag{15}$$

$$w_{\text{restore}}^{1:6} = \text{Rotate}_{\text{Enc}}(w_{\text{rotate}}^{1:6}, w_{\text{source}}^{1:6}), \tag{16}$$

here we additionally have w_{restore}, which tries to turn back the modified representation. And since we do not change the last vector representations, the following is true: $w_{\text{rotate}}^{7:18} = w_{\text{restore}}^{7:18} = w_{\text{source}}^{7:18}$. And also $w_{\text{rotate}}, w_{\text{restore}} \in \mathbb{R}^{18 \times 512}$

During training, we randomly generate source image and target image pairs and train according to the following loss functions:

$$H_{\text{target}} = E(I_{\text{target}}), \tag{17}$$

$$\mathcal{L}_{\text{pose}} = ||H_{\text{target}} - E(G(w_{\text{rotate}}))||_2^2. \tag{18}$$

In this case, E – a pre-trained model for extracting 2D face key-points, we used the STAR [39] model for this purpose. For optimization, we used the first 76 key-points which corresponded to the face contour, eyebrows, eyes and nose. Thus $H_{\text{target}} \in \mathbb{R}^{76 \times 2}$. In this way, due to $\mathcal{L}_{\text{pose}}$ loss, we train the model to rotate the image to the required pose for hair alignment.

The following loss functions are used to keep the original shape of the hair:

$$\mathcal{L}_{\text{recon}} = ||w_{\text{source}}^{\text{E4E}} - w_{\text{restore}}||_2^2, \tag{19}$$

$$\mathcal{L}_{\text{id}} = \text{ArcFace}(I_{\text{source}}, G(w_{\text{rotate}})). \tag{20}$$

Here the main loss function for attribute conservation is $\mathcal{L}_{\text{recon}}$, it is what motivates the model to learn transformations that do not change the attributes of the image other than its pose. In this case, when we have rotated the image and requires to rotate it to the original pose, we know the ground truth and in this loss function we just take L2 between it. The loss function $\mathcal{L}_{\text{id}}$ is only a guide and also helps to preserve attributes a bit when rotating.

3

Total final loss function:

$$\mathcal{L} = \frac{\lambda_{\text{pose}} \cdot \mathcal{L}_{\text{pose}}}{\text{EMA}_t(\mathcal{L}_{\text{pose}})} + \frac{\lambda_{\text{recon}} \cdot \mathcal{L}_{\text{recon}}}{\text{EMA}_t(\mathcal{L}_{\text{recon}})} + \frac{\lambda_{\text{id}} \cdot \mathcal{L}_{\text{id}}}{\text{EMA}_t(\mathcal{L}_{\text{id}})} \tag{21}$$

The main challenge in training this model is to find the correct coefficients for the given loss functions. The model learns to adopt the correct pose very quickly and because of this, ArcFace soon begins to dominate, and with heavy overtraining, artifacts begin to form because of it. To avoid this, we first normalize the loss functions by their exponential moving average (EMA), which we compute with a factor of $t = 0.02$. This allows us to maintain the correct prioritization of the loss functions throughout training.

The final model was trained with $\lambda_{\text{pose}} = 6, \lambda_{\text{recon}} = 2, \lambda_{\text{id}} = 1$. The optimizer was Adam with learning rate 1×10^{-4} and weight decay 1×10^{-6}. Batch size 16.

The architecture of the model is similar to the one used in Color Encoder Fig. 4. We predict the change of $w_{\text{source}}^{1:6} \in \mathbb{R}^{6 \times 512}$, and input $w_{\text{target}}^{1:6} \in \mathbb{R}^{6 \times 512}$ to the modulation layer. In total, there are 5 blocks in the model form $\text{Linear}(512, 512) \rightarrow \text{Modulation} \rightarrow \text{LeakyReLU}(0.01)$. The block diagram of the Modulation block can be seen on the Fig. 4. The f_β and g_γ are $\text{Linear}(512, 512) \rightarrow \text{LayerNorm}(512) \rightarrow \text{LeakyReLU}(0.01) \rightarrow \text{Linear}(512, 512)$.

9.2 Color Encoder

To train the Color Encoder, we collect about 5800 image pairs from the FFHQ dataset. For this purpose, we run our model on 3000 triples from FFHQ and save FS spaces of source, shape and color images after FS and W+ mixing, and then run Shape Module to transfer hair shape from shape image to source image and additionally run one more time to transfer hair shape from color image to source. The resulting F spaces are also saved. This allows us to create 6000 pairs for color transfer, which we additionally filter and discard images without hair, from which we can not take the desired color.

Thus each object in the training sample consists of I_{source}, S_{source} and $F_{\text{source}}^{\text{align}}$ of the original image with the hair shape from the color image, and I_{color}, S_{color} of the color image. Our goal is to modify S_{source} to get the same hair color as I_{color} with the given $F_{\text{source}}^{\text{align}}$ tensor.

The first thing we do for model training is to prepare the masks:

$$H_{\text{color}} = (\text{BiSeNet}(I_{\text{color}}) = \text{hair}), \tag{22}$$
$$H_{\text{source}} = (\text{BiSeNet}(I_{\text{source}}) = \text{hair}), \tag{23}$$
$$H_{\text{align}} = (\text{BiSeNet}(\text{G}(F_{\text{source}}^{\text{align}}, S_{\text{source}})) = \text{hair}), \tag{24}$$
$$M_{\text{target}} = \overline{H_{\text{source}}} \cdot \overline{H_{\text{align}}}. \tag{25}$$

Now we can apply our model:

$$\text{emb}_{\text{face}} = \text{CLIP}_{\text{enc}}(I_{\text{source}} \cdot M_{\text{target}}), \tag{26}$$
$$\text{emb}_{\text{hair}} = \text{CLIP}_{\text{enc}}(I_{\text{color}} \cdot H_{\text{color}}), \tag{27}$$
$$S_{\text{blend}} = \text{Blend}_{\text{Enc}}(S_{\text{source}}, S_{\text{color}}, \text{emb}_{\text{face}}, \text{emb}_{\text{hair}}), \tag{28}$$
$$I_{\text{blend}} = \text{StyleGAN}(F_{\text{source}}^{\text{align}}, S_{\text{blend}}), \tag{29}$$

And now we can apply the following loss function, which worked better than LPIPS or other combinations:

$$\mathcal{L}_{\text{clip}}(I_1, I_2, M_1, M_2) = 1 - \text{CosSim}_{\text{CLIP}}(I_1 \cdot M_1, I_2 \cdot M_2), \tag{30}$$

here $\text{CosSim}_{\text{CLIP}}$ is the cosine distance between the embedding images from the CLIP model. The images are before that multiplied by the corresponding masks M.

We get the following loss function from here, which we optimize during training for our model.

$$\mathcal{L}_{\text{color}} = \mathcal{L}_{\text{clip}}(I_{\text{blend}}, I_{\text{color}}, H_{\text{align}}, H_{\text{color}}), \tag{31}$$

$$\mathcal{L}_{\text{face}} = \mathcal{L}_{\text{clip}}(I_{\text{blend}}, I_{\text{source}}, M_{\text{target}}, M_{\text{target}}), \tag{32}$$

$$\mathcal{L} = \lambda_{\text{color}} \cdot \mathcal{L}_{\text{color}} + \lambda_{\text{face}} \cdot \mathcal{L}_{\text{face}}. \tag{33}$$

The final model was trained with $\lambda_{\text{color}} = 1$ and $\lambda_{\text{face}} = 1$. The optimizer was Adam with learning rate 1×10^{-4} and weight decay 1×10^{-6}. Batch size 16.

The architecture of the Color Encoder is shown in the Fig. 4. The f_β and g_γ are $\text{Linear}(1536, 1024) \rightarrow \text{LayerNorm}(1024) \rightarrow \text{LeakyReLU}(0.01) \rightarrow \text{Linear}(1024, 512)$, for a total of 5 blocks in the model.

9.3 Refinement alignment

To train Refinement encoder, we collect 10'000 triples from FFHQ on which we run our method without this encoder. The image I_{blend} after Color Encoder and the original face image I_{source} are the object of the training sample.

The training of this stage consisted of three parts: FS encoder training, fusers training, and the whole model finetuning.

9.3.1 Feature Style Encoder

For FS encoder training, we first obtain a reconstruction of the real image:

$$F_{64}^{\text{source}}, \, S^{\text{source}} = FS_{\text{Enc (ours)}}(I_{\text{source}}), \tag{34}$$

$$F_{\text{style}}^{\text{source}} = \text{G}_8(S^{\text{source}}), \tag{35}$$

$$F_{64}^{\text{recon}} = \alpha \cdot F_{64}^{\text{source}} + (1 - \alpha) \cdot F_{\text{style}}^{\text{source}}, \tag{36}$$

$$I_{\text{style}} = \text{StyleGAN}(F_{\text{style}}^{\text{source}}, \, S^{\text{source}}), \tag{37}$$

$$I_{\text{recon}} = \text{StyleGAN}(F_{64}^{\text{recon}}, \, S^{\text{source}}). \tag{38}$$

Besides the reconstruction image from FS space here we also get the F tensor from S space and the image generated by S. Also here α is a parameter which is 0 at the beginning and gradually increases to 1 during the learning process. And after that we use the following loss functions:

$$\mathcal{L}_{\text{id}}(I_1, I_2) = \text{ArcFace}(I_1, I_2), \tag{39}$$

$$\mathcal{L}_{\text{m\_LPIPS}}(I_1, I_2) = \text{m\_LPIPS}(I_1, I_2), \tag{40}$$

$$\mathcal{L}_{\text{recon. feat}}(F_1, F_2) = ||F_1 - F_2||_2^2, \tag{41}$$

here $\mathcal{L}_{\text{m\_LPIPS}}$ is a multi-scale perceptual loss [36] presented by the authors of FS encoder, which in addition with $\mathcal{L}_{\text{id}}$ reconstructs the original image. In turn, it will $\mathcal{L}_{\text{recon. feat}}$ help in the initial stages to obtain an F tensor similar to the F tensor created from S space, so here we throw gradients only through one F tensor, not pull them together.

Thus the final loss function for training our FS encoder:

$$\begin{aligned} \mathcal{L} = \lambda_{\text{id}} \cdot (&\mathcal{L}_{\text{id}}(I_{\text{source}}, I_{\text{style}}) + \mathcal{L}_{\text{id}}(I_{\text{source}}, I_{\text{recon}})) + \\ + \lambda_{\text{m\_LPIPS}} \cdot (&\mathcal{L}_{\text{m\_LPIPS}}(I_{\text{source}}, I_{\text{style}}) + \\ + &\mathcal{L}_{\text{m\_LPIPS}}(I_{\text{source}}, I_{\text{recon}})) + \\ + \lambda_{\text{recon. feat}} \cdot &\mathcal{L}_{\text{recon. feat}}(F_{\text{style}}^{\text{source}}, F_{64}^{\text{source}}) + \\ + \lambda_{\text{ADV}} \cdot &\mathcal{L}_{\text{ADV}}(I_{\text{recon}}), \end{aligned} \tag{42}$$

5

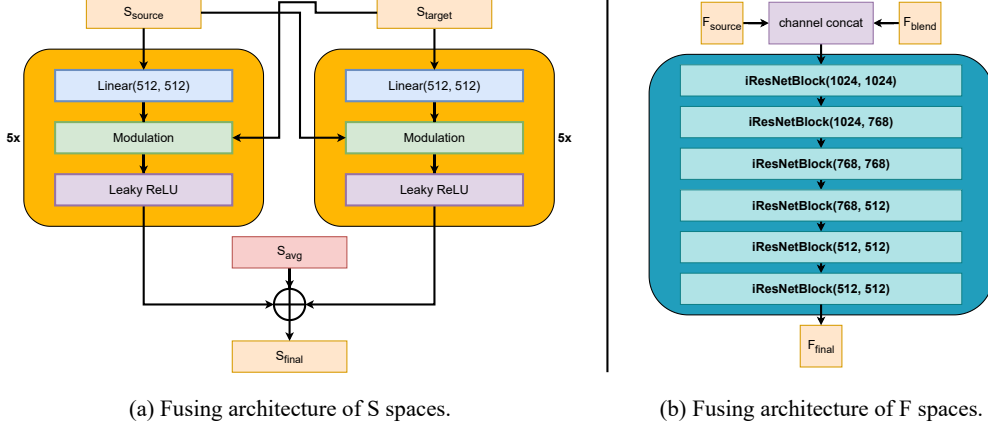(a) Fusing architecture of S spaces.

(b) Fusing architecture of F spaces.

Figure 9: Diagram of fusing encoders in post processing. (a) The modulation architecture is shown in the Fig. 4. f_β and g_γ in modulation are $\text{Linear}(512, 512) \rightarrow \text{LayerNorm}(512) \rightarrow \text{LeakyReLU}(0.01) \rightarrow \text{Linear}(512, 512)$. S_{avg} is the average latent vector. (b) The architecture consists of the usual IResNet blocks.

here we also use adversarial loss function, as a discriminator we take the pre-trained one from StyleGAN and train it in the process using the classical loss function and R1 regularization.

The following parameters were used for training: $\lambda_{\text{id}} = 0.1$, $\lambda_{\text{m\_LPIPS}} = 0.8$, $\lambda_{\text{recon. feat}} = 0.01$ and $\lambda_{\text{ADV}} = 0.2$. Adam was used for encoder optimization with learning rate 2×10^{-4} and weight decay 0, and for discriminator learning rate 1×10^{-4}, betas=(0.9, 0.999) and weight decay 0. The model was trained with a batch size 16.

9.3.2   Fusing Encoders

Once we have trained the FS encoder we can proceed to train models for fusing spaces. First of all we need to get FS representations from images:

$$F_{64}^{\text{blend}}, \ S^{\text{blend}} = FS_{\text{Enc (ours)}}(I_{\text{blend}}), \tag{43}$$

$$F_{64}^{\text{source}}, \ S^{\text{source}} = FS_{\text{Enc (ours)}}(I_{\text{source}}). \tag{44}$$

Now we can apply our models to fusing:

$$F_{\text{final}} = \text{Fused}_{\text{F Enc}}(F_{64}^{\text{blend}}, \ F_{64}^{\text{source}}), \tag{45}$$

$$S_{\text{final}} = \text{latent}_{\text{avg}} + \text{Fused}_{\text{S Enc}}(S^{\text{blend}}, \ S^{\text{source}}), \tag{46}$$

$$F_{\text{style}} = G_8(S^{\text{final}}), \tag{47}$$

$$I_{\text{style}} = \text{StyleGAN}(F_{\text{style}}, \ S_{\text{final}}), \tag{48}$$

$$I_{\text{final}} = \text{StyleGAN}(F_{64}^{\text{final}}, \ S_{\text{final}}). \tag{49}$$

Similar loss functions are used to train the encoders, but in addition we use DSC++ [37] to make the segmentation mask match the original one. We also add a loss function for inpaint. First of all, we will need new masks:

$$H_{\text{source}} = (\text{BiSeNet}(I_{\text{source}}) = \text{hair}), \tag{50}$$

$$H_{\text{blend}} = (\text{BiSeNet}(I_{\text{blend}}) = \text{hair}), \tag{51}$$

$$M_{\text{target}} = \overline{H_{\text{source}}} \cdot \overline{H_{\text{blend}}}, \tag{52}$$

$$M_{\text{inpaint}} = \overline{M_{\text{target}}} \cdot \overline{H_{\text{blend}}}. \tag{53}$$

6

The inpaint is quite sensitive to the boundaries of the mask on which it is applied, so we use a special soft dilation that runs this mask through a convolution with a kernel consisting of ones in the shape of a circle of radius 25. After that we raise the result to degree 1/4, which allows us to get more concentrated values. We also create a hard mask for the I_{style} guide as we don't want to look at the hair of this image:

$$M_{\text{smooth}} = \text{Dilation}(M_{\text{inpaint}}), \tag{54}$$

$$M_{\text{hard}} = M_{\text{smooth}} \cdot \overline{H_{\text{blend}}}. \tag{55}$$

The discriminator is the main contributor to inpaint, but to help ease the load on it we guide it with $\mathcal{L}_{\text{m\_LPIPS}}$ over the inpaint area using two images: I_{blend} and I_{style}. The first one is used to guide and preserve better details, while the second image has a more correct shading.

The final loss function is as follows:

$$
\begin{aligned}
\mathcal{L} = \lambda_{\text{id}} \cdot (&\mathcal{L}_{\text{id}}(I_{\text{source}} \cdot M_{\text{target}}, I_{\text{style}} \cdot M_{\text{target}}) + \\
&+ \mathcal{L}_{\text{id}}(I_{\text{source}} \cdot M_{\text{target}}, I_{\text{final}} \cdot M_{\text{target}})) + \\
&+ \lambda_{\text{m\_LPIPS}} \cdot (\mathcal{L}_{\text{m\_LPIPS}}(I_{\text{source}} \cdot M_{\text{target}}, I_{\text{style}} \cdot M_{\text{target}}) + \\
&+ \mathcal{L}_{\text{m\_LPIPS}}(I_{\text{source}} \cdot M_{\text{target}}, I_{\text{final}} \cdot M_{\text{target}})) + \\
&+ \lambda_{\text{m\_LPIPS}} \cdot (\mathcal{L}_{\text{m\_LPIPS}}(I_{\text{target}} \cdot H_{\text{blend}}, I_{\text{style}} \cdot H_{\text{blend}}) + \\
&+ \mathcal{L}_{\text{m\_LPIPS}}(I_{\text{target}} \cdot H_{\text{blend}}, I_{\text{final}} \cdot H_{\text{blend}})) + \\
&+ \lambda_{\text{recon. feat}} \cdot \mathcal{L}_{\text{recon. feat}}(F_{\text{style}}, F_{\text{final}}) + \\
&+ \lambda_{\text{DSC++}} \cdot \mathcal{L}_{\text{DSC++}}(\text{BiSeNet}(I_{\text{blend}}), \text{BiSeNet}(I_{\text{final}})) + \\
&+ \lambda_{\text{inpaint}} \cdot (\mathcal{L}_{\text{m\_LPIPS}}(I_{\text{style}} \cdot M_{\text{hard}}, I_{\text{final}} \cdot M_{\text{hard}}) + \\
&+ \mathcal{L}_{\text{m\_LPIPS}}(I_{\text{blend}} \cdot M_{\text{smooth}}, I_{\text{final}} \cdot M_{\text{smooth}})) + \\
&+ \lambda_{\text{ADV}} \cdot \mathcal{L}_{\text{ADV}}(I_{\text{final}}).
\end{aligned} \tag{56}
$$

The following parameters were used for training: $\lambda_{\text{id}} = 0.1, \lambda_{\text{m\_LPIPS}} = 0.4, \lambda_{\text{recon. feat}} = 0.01, \lambda_{\text{DSC++}} = 0.1, \lambda_{\text{inpaint}} = 0.2$ and $\lambda_{\text{ADV}} = 0.2$. Adam was used for encoders optimization with learning rate 2×10^{-4} and weight decay 0, and for discriminator learning rate 3×10^{-4}, betas=(0.9, 0.999) and weight decay 0. The model was trained with a batch size 16.

After we have trained the fusers, we unfreeze the FS encoder and retrain the whole model with the same parameters, but with half the learning rate.

The fusing architecture of S and F spaces is presented in Fig. 9.

10 Operating modes

Each module of the method has its own strict function, but not all modes of operation require all modules.

10.1 Transferring the desired color

Thus, for the task of changing only the hair color, we do not need to run the Alignment module, since we do not need to change the hair shape.

In this mode, our method performs in 0.52 and 0.27 seconds on V100 and A100, respectively, while HairCLIP performs in 0.31 and 0.25 seconds. This enables us to achieve nearly identical speed as the fastest hair transfer methods, but still have twice the quality according to the realism metrics.

10.2 Transfer of the desired shape

Similarly, for the task of transferring only the hair shape, we do not need to run the Pose alignment module for the Color Encoder, since the hair shape of the color matches the original hair shape that can be obtained from BiSeNet. However, the Color Module must still be run for this mode, even

though we want to keep the original hair color. This is necessary due to the fact that when we transfer the hair shape, the hair color from the shape image may leak into the F space, similar to what is seen in the experiments without mixing F spaces in the ablation Fig. 6. In addition, Color Encoder recovers lost details during the inversion stage, which also improves the quality.

In this mode, our method performs in 0.71 and 0.40 seconds on V100 and A100, respectively. While in both shape and color transfer mode, the run times were 0.78 and 0.52 seconds.

10.3 Transferring hairstyle and color from one image.

This mode as well as the previous one formally works with only 2 images, which reduces the load on the encoders and also we do not need to re-run the Pose alignment module for Color Encoder.

Like the previous one, it performs in 0.71 and 0.40 seconds on V100 and A100, respectively.

11 Analysis of computational complexity

Table 4: This table presents a comprehensive analysis of the method's performance metrics. The evaluation criteria include execution time, computational efficiency measured in TFLOPS, the total number of model parameters, and CUDA memory requirements.

Model	Time (s)↓	TFLOPS↓	Params.↓	CUDA (GB)↓
HairCLIP [32]	**0.36**	**0.74**	**437M**	**2.1**
HairCLIPv2 [33]	221	1213	Opt.	13.56
CtrlHair [7]	7.87	<u>1.04</u>	<u>512M</u>	4.8
StyleYourhair [16]	239	2637	Opt.	11.21
Barbershop [41]	645	3745	Opt.	<u>2.3</u>
HairFast (ours)	<u>0.78</u>	6.5	1.4B	7.5

In the Table 4, we can see additional comparisons of the computational efficiency of the methods. For testing, we used cases with independent shape and color of hairstyles of the methods where possible. To calculate TFLOPS, we used the torchprofile library, which uses torch.jit.profile. We approximated the backward complexity of optimization-based methods by the forward complexity.

Analyzing the results, our method needs significantly more computation than other encoder-based methods, but we outperform CtrlHair in speed at the expense of their inefficient postprocessing. At the same time, we significantly outperform state-of-the-art optimization-based methods.

12 Complete tables

Table 5: All metrics were measured on the same pre-selected triples of images (face, shape and color) from the CelebaHQ [12] dataset. Then, applying the method, FID was measured on the original dataset and the modified dataset. FID_{CLIP} [18] was counted similarly to FID using the torchmetrics library, but a CLIP encoder was used instead of Inception V3. Running time was measured as the median time among a bunch of method runs, without taking into account models initialization and loading/saving images to disk.

Model	FID↓				FID_{CLIP}↓				Time (s)↓	
	full	both	color	shape	full	both	color	shape	A100	V100
HairCLIP [32]	34.95	40.68	40.08	42.92	12.20	13.32	10.94	13.44	**0.28**	**0.36**
HairCLIPv2 [33]	<u>14.28</u>	<u>23.37</u>	20.21	<u>23.90</u>	10.98	12.14	6.55	10.06	112	221
CtrlHair [7]	15.10	24.81	**19.65**	25.60	9.52	10.42	<u>3.62</u>	9.59	6.57	7.87
StyleYourHair [16]	-	25.90	-	-	-	10.91	-	-	84	239
Barbershop [41]	15.94	24.52	20.54	24.08	<u>7.07</u>	<u>8.12</u>	3.89	<u>6.76</u>	213	645
HairFast (ours)	**13.12**	**22.71**	<u>20.17</u>	**23.36**	**5.12**	**6.06**	**3.00**	**5.34**	<u>0.41</u>	<u>0.78</u>

Analyzing the complete Table 5, our method performs better on all cases according to the FID_{CLIP} metric. But we lose to CtrlHair when transferring only hair color by FID metric because in this case CtrlHair blends almost the whole image except hair, which is strongly encouraged by the metric.

Table 6: **Pose Metrics**. For this metrics, we consider color and shape transfer from the target image to the source image. For all pairs of images we calculate MAE of key facial points and split the pairs into three equal folds, which correspond to weak, medium and high pose-difference on which the corresponding metrics were measured. There were 1000 image pairs taken from CelebA-HQ. **Reconstruction Metrics**. For this each method is started on the task of transferring the color and shape of the hairstyle from itself to itself, thus at the end we measure the metrics with the original image. There were 3000 images taken from CelebA-HQ.

Model	Pose metrics				Reconstruction metrics			
	$FID\downarrow$		$FID_{CLIP}\downarrow$					
	medium	hard	medium	hard	LPIPS$\downarrow$	PSNR$\uparrow$	FID$\downarrow$	FID$_{CLIP}\downarrow$
HairCLIP [32]	55.77	54.35	15.53	15.73	0.36	14.08	35.49	10.48
HairCLIPv2 [33]	44.62	45.28	14.56	18.66	0.16	19.71	10.09	4.08
CtrlHair [7]	46.45	50.12	12.96	16.42	0.15	19.96	**8.03**	1.25
StyleYourHair [16]	46.32	47.19	13.70	15.93	0.14	21.74	10.69	2.73
Barbershop [41]	44.08	46.13	11.27	13.30	0.11	21.18	13.73	2.61
HairFast (ours)	**43.25**	**44.85**	**8.90**	**10.33**	**0.08**	**23.45**	9.72	**0.97**

The Table 6 shows the full reconstruction results, we only lose to CtrlHair on the FID metric due to Poisson mixing, which is strongly encouraged by the metric.

Table 7: **Identity preservation metric.** This metric quantifies facial similarity before and after hair transfer across various configurations. Facial similarity was assessed using the SFace model, with cosine distance between image embeddings.

Model	Identity preservation metric$\uparrow$				
	full	both	color	shape	reconstruct
HairCLIP [32]	0.530	0.534	0.563	0.539	0.581
HairCLIPv2 [33]	0.630	**0.642**	0.761	0.648	0.798
CtrlHair [7]	0.622	0.629	0.777	0.633	0.818
StyleYourhair [16]	-	0.623	-	-	0.831
Barbershop [41]	0.590	0.591	0.754	0.637	**0.845**
HairFast (ours)	**0.639**	0.640	**0.778**	**0.659**	0.826

Table 7 shows additional comparisons of the methods on the metric of face identity preservation after hairstyle transfer. The results prove that our method preserves the details of the original image well and preserves identity better than other approaches in the main cases.

13 Realism after editing by StyleGAN-salon

Table 8: The task of transferring hair color and shape from the target image to the original image. Selected 450 pairs from the FFHQ dataset by the authors of StyleGAN-Salon. Images of method results are provided by the authors of StyleGAN-Salon, we compute the images for CtrlHair and our method. The table shows the realism metrics as well as the RMSE of key points of the generated face and the original face, also as this metric is shown by the authors of StyleGAN-Salon.

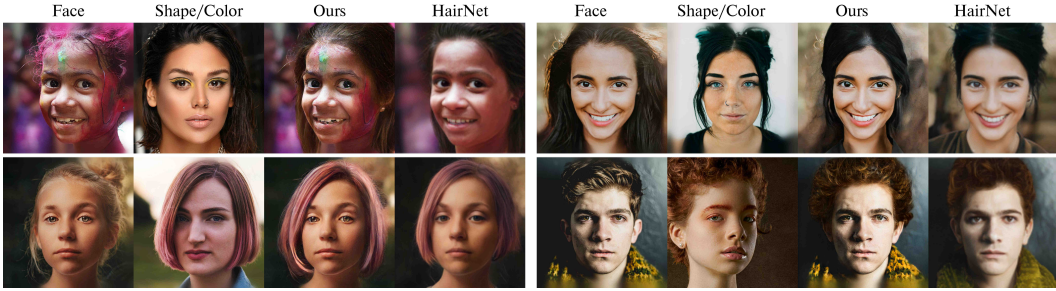| Model	Based	FID$\downarrow$	FID$_{CLIP}\downarrow$	RMSE$\downarrow$
CtrlHair [7]	Enc.	52.14	15.27	5.48
LOHO [25]	Opt.	50.24	15.12	8.16
HairCLIPv2 [33]	Opt.	48.95	14.59	5.66
StyleYourHair [16]	Opt.	52.10	12.68	7.14
Barbershop [41]	Opt.	50.76	10.37	7.31
StyleGAN-Salon [15]	Opt.	50.32	9.91	4.65
HairFast (ours)	Enc.	**47.15**	**8.45**	**4.52**

Figure 10: Visual comparison of our method with HairNet [42].

The authors of the StyleGAN-Salon [15] method have not yet published their code, which makes them hard to compare, but they do provide a small dataset with an inference of many basic models, including LOHO [25], one of the first optimization-based methods, as well as their own method. The StyleGAN-Salon method itself uses optimization-based methods and also PTI, which should make it very long. In the article itself, the authors talk about 21 minutes on a single input pair, but the measurements were done on a different hardware.

Unlike our past metrics, these were measured on the FFHQ dataset on which we trained our models. The authors are measured on 450 image pairs, which together did not occur in any training from our models.

Analyzing the results Table 8 we see that we outperform all methods on all presented metrics, including the new metric RMSE measured between key points of the original face and the generated face. This metric confirms the effectiveness of our Shape alignment approach, while other methods may corrupt the shape of the original face. Also, these results confirm the effectiveness of our Refinement alignment stage, which outperforms even PTI in terms of realism.

Also see a visual comparison of our model with StyleGAN-Salon in Fig. 15.

14 Comparison with HairNet and HairNeRF

The authors of HairNet [42] and HairNeRF [3] have not published their code at this time, nor do they provide any datasets on which to compare with them. So instead we compare with them based on features that can be seen partially in the Table 2 and also visually in the images from their paper.

14.1 Comparison with HairNet

By analyzing HairNet we can highlight some key points compared to our work: (1) HairNet uses optimization based inversion and PTI for pre-processing, which implies that it must run hundreds of times slower than our method. (2) HairNet has worse FID than Barbershop and visually worse than StyleGAN Salon. Consequently, we should expect better performance. (3) HairNet works only in low FS space, so the method retains much less details than our approach. (4) Unlike our approach, HairNet is incapable of transferring hair color independent of hair shape.

Visual analysis with images reported in HairNet paper is shown in Figure 10. HairNet compared to our method tend to be poor at preserving the details of the original image and may change the identity, also the method may be worse at transferring hair texture and color.

Summarizing the above, we can say that HairNet is inferior to ours in many characteristics.

14.2 Comparison with HairNeRF

Unlike most other approaches HairNeRF [3] works on StyleNeRF [6] rather than StyleGAN, which makes it more difficult to compare features without their code. But in their work they use optimizations including image inversion with PTI, alignment and blending, which means they have to work hundreds of times longer than our approach. In addition, HairNeRF cannot independently transfer hair color or hair shape.
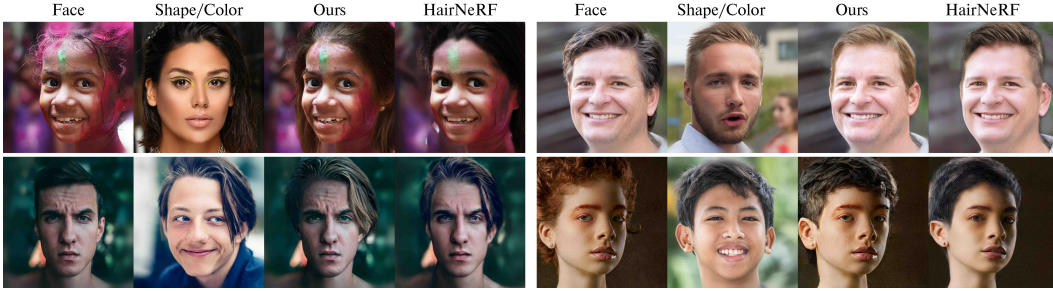
Figure 11: Visual comparison of our method with HairNeRF [3].

A visual comparison Fig. 11 shows the effectiveness of StyleNeRF for image alignment compared to other baseline models, but still comparable to our approach. Although HairNeRF uses PTI, our method still preserves more details of the original image and preserves the identity better.

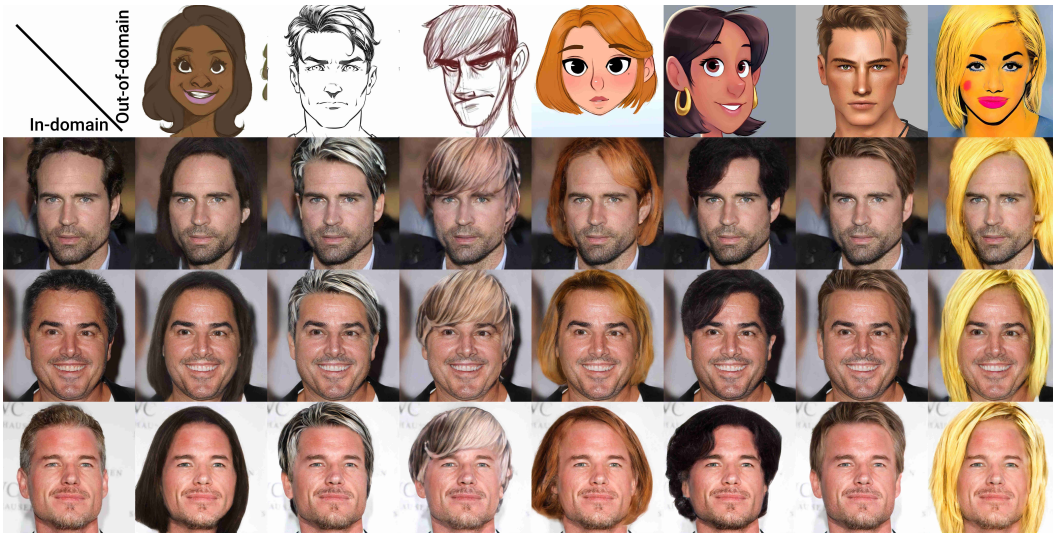## 15   Cross-domain hair transfer

Figure 12: The visual results of cross-domain hair transfer demonstrate the robustness of our method.

The Fig. 12 shows hair transfer experiments in different domains. The results show that our method is able to handle such cases and the main bottleneck may occur in the BiSeNet model, which will not be able to correctly segment the hair mask for further hair transfer.

16   Color transfer metric

In our work, we also try to find a metric for the quality of hair color transfer, other methods have not provided anything similar before. Our attempts to find a metric were aimed at using different hair loss functions or estimation models that were applied on the results when transferring random color and desired color. But no one model found a statistical difference between these results, which did not allow us to make a conclusion about the quality of the color transfer.

But we were able to get one of the estimates that showed statistical significance between random and set experiments. To do this, we convert the color image and the final image into HSV format, from which we take the pixels corresponding to the eroded hair mask for each image. After that, for each pixel coordinate corresponding to hue, saturation and value we construct a discrete distribution of values over 500 bins. Finally, we compute the similarity of the resulting discrete distributions using

Table 9: A color transfer metric that measures the Jensen-Shannon divergence between the histogram of the hue, saturation and value channel distributions of the hair of resulting image and the target hair image in HSV format. For this metric, we consider the color and shape transfer from the target image to the source image. 1000 image pairs from CelebA-HQ are taken.

Model	H mean↓	S mean↓	V mean↓
HairCLIP [32]	**0.346**	0.275	0.265
HairCLIPv2 [33]	0.418	0.291	0.278
CtrlHair [7]	<u>0.349</u>	**0.249**	**0.236**
StyleYourHair [16]	0.429	0.310	0.290
Barbershop [41]	0.405	0.266	<u>0.249</u>
HairFast (ours)	0.357	<u>0.262</u>	0.269

Jensen-Shannon divergence. The average results for 1000 random experiments from CelebA-HQ are summarized in Table Table 9.

Unfortunately, this metric doesn't take into account too many factors, such as things like lighting in the images, but it still allows us to draw some conclusions.

So our method shows very good results for hue and saturation channels, which indicates quite accurate color reproduction. This correlates with the visual Fig. 13 and Fig. 14 comparison.

17 Visual comparison

The Fig. 13 and Fig. 14 shows the results of a visual comparison with the Barbershop [41], StyleYourHair [16], HairCLIPv2 [33], CtrlHair [7] and HairCLIP [32] methods. StyleYourHair cannot transfer the desired color from a different image, so it tried to transfer the color from the shape image. Also, for visual comparison, we disabled StyleYourHair's horizontal flip feature, which was enabled when we calculated our metrics in the main part. This feature allows StyleYourHair to check if the reflected image will have a smaller pose difference than the original image and transfer the hairstyle from it. We disabled it in this compare to see how their approach to image rotation works compared to ours, and because most hairstyles are asymmetrical and often want to be transferred as they are in the example.

Analyzing the Fig. 13 and Fig. 14 results, in addition to the speed gain, we would like to mention the excellent quality, which in most cases outperforms other methods. For example, our method successfully captures the desired hue in all submitted images, while other methods fail. In addition, our method is much better at preserving the identity of the face, its details and its surroundings. And with all this, our method handles well the difficult cases related to pose differences, while even StyleYourHair, which is focused on this task, does worse.

We also do a visual comparison with StyleGAN-Salon in the image Fig. 15. Although their method produces PTI for each example to preserve more details of the original image, our large encoder approach performs much better and preserves much more details. This in particular allows us to better preserve the identity of the face and attributes such as glasses. This is also confirmed by the Table 8 metrics. In addition, StyleGAN-Salon's 3D image rotation approach with hair shape sometimes creates artifacts on the hair, and also sometimes changes long hair to short hair because of this.

You can also see in the Fig. 16 image how our method compares to other methods for large pose differences, in particular 30, 50, 70 and 90 degree differences. Our approach with Rotate Encoder significantly outperforms other methods, including StyleGANSalon which is focused on solving this problem. At the same time, our method can handle even the largest pose differences without causing artifacts and still keep the hairstyle recognizable.

The last visual experiment can be observed in the Fig. 17 image. Here we compare our method with others in various complex cases including complex face, details on the face and complex background. Our Refinement alignment step significantly outperforms other detail preservation approaches by producing significantly more realistic images. While CtrlHair also outperforms other methods using Poisson blending, this approach leaves noticeable artifacts at mask boundaries.

Figure 13: The first part of a visual comparison of the performance of the methods. All methods transfer hair shape and color from given images, except StyleYourHair, which due to its limitations transfers both shape and color only from Shape images.

Figure 14: The second part of a visual comparison of the performance of the methods. All methods transfer hair shape and color from given images, except StyleYourHair, which due to its limitations transfers both shape and color only from Shape images.
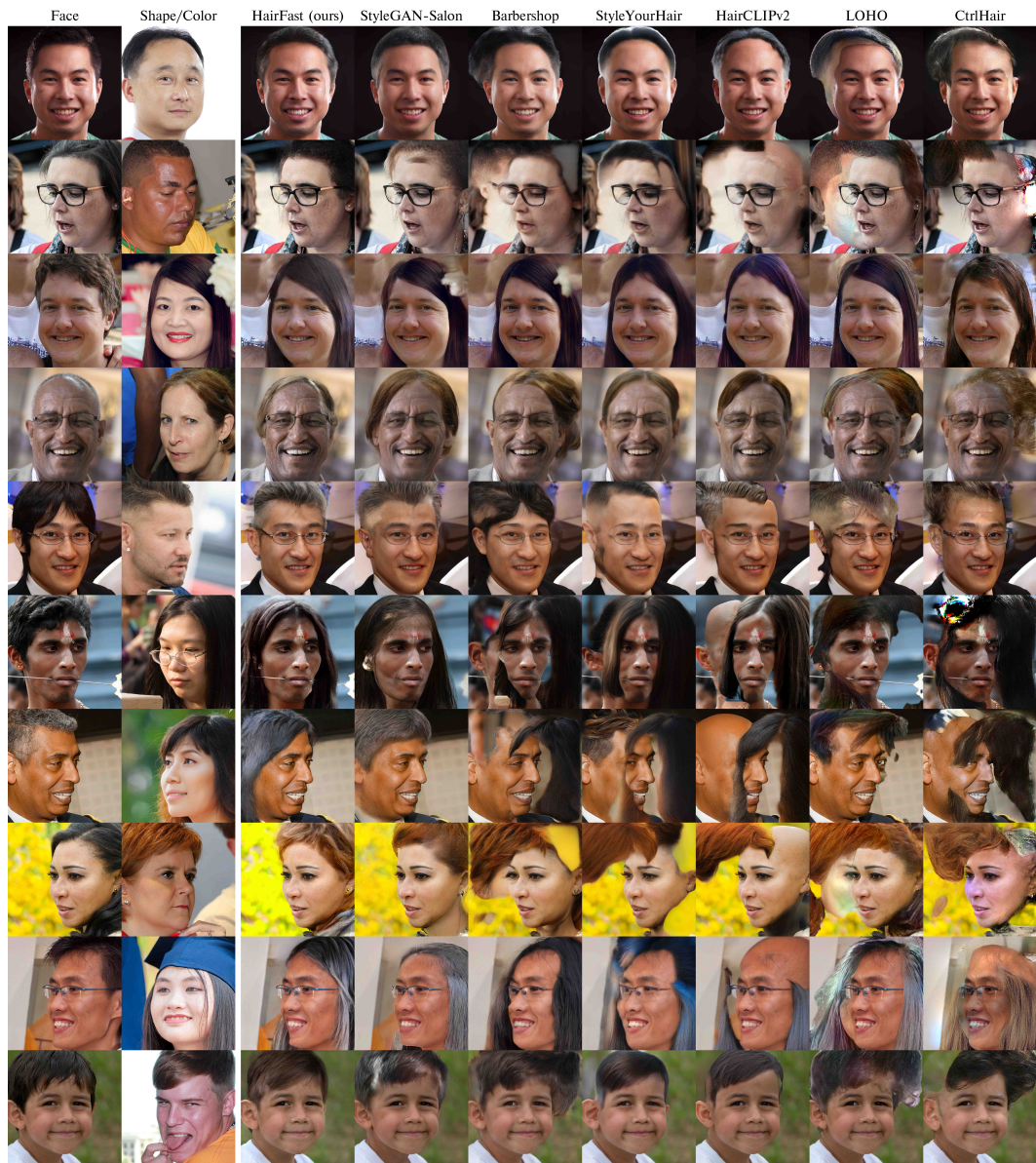
Figure 15: Visual comparison with pictures presented by the authors of StyleGAN Salon, where we are additionally compared to their model as well as LOHO.

Figure 16: Detailed comparison of methods with different face rotation, the image shows cases with 30, 50, 70 and 90 degree rotation. Our method to rotation is significantly superior to other methods of hair transfer.

Figure 17: The image shows the comparison of our method with the others on various difficult cases for the method: complex face, objects on the face and complex background.

Figure 18: Additional visual comparisons on the transfer of wavy and curly attributes compared to the baseline methods.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We presented a new method and clearly described what our paper contributed.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We described the limits and problem cases in the paper. More detailed results are also presented in the supplementary materials.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

Justification: There are no theoretical results in our paper.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: Our work describes the sampling of the data on which the reproduced results can be obtained, as well as the metrics implementations.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: Our work has published code along with pre-trained weights of the models, scripts for training, and scripts to measure all metrics for full reproducibility of the entire work.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: All training details, including hyperparameters, optimizers, descriptions of architectures, and anything needed for self-reproduction are provided in the supplemental materials.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [No]

Justification: Despite the article does not explicitly mention statistical significance, everything necessary to calculate it is given. All results are statistically significant.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: The article indicates which GPUs were used to compute the benchmarks.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

Answer: [Yes]

Justification: The study complies in all respects with the NeurIPS code of ethics. In particular, we made sure that we did not infringe anyone's copyright under the licenses.

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [No]

Justification: This point is not discussed in the article. But the task of changing hairstyles seems quite harmless as it does not generate potentially dangerous content and cannot be used for deepfakes.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: We don't need those for our model.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: Both in the article and in our code, we explicitly specify all the assets used and do not violate anyone's rights under the license.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: All new assets are well documented.

    Guidelines:
    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: We didn't do crowdsourcing.

    Guidelines:
    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: We didn't have any study participants.

    Guidelines:
    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.