ReStyle: A Residual-Based StyleGAN Encoder via Iterative Refinement

Yuval Alaluf
Tel-Aviv University

Or Patashnik
Tel-Aviv University

Daniel Cohen-Or
Tel-Aviv University

Abstract

Recently, the power of unconditional image synthesis has significantly advanced through the use of Generative Adversarial Networks (GANs). The task of inverting an image into its corresponding latent code of the trained GAN is of utmost importance as it allows for the manipulation of real images, leveraging the rich semantics learned by the network. Recognizing the limitations of current inversion approaches, in this work we present a novel inversion scheme that extends current encoder-based inversion methods by introducing an iterative refinement mechanism. Instead of directly predicting the latent code of a given real image using a single pass, the encoder is tasked with predicting a residual with respect to the current estimate of the inverted latent code in a self-correcting manner. Our residual-based encoder, named ReStyle, attains improved accuracy compared to current state-of-the-art encoder-based methods with a negligible increase in inference time. We analyze the behavior of ReStyle to gain valuable insights into its iterative nature. We then evaluate the performance of our residual encoder and analyze its robustness compared to optimization-based inversion and state-of-the-art encoders. Code is available on our project page: https://yuval-alaluf.github.io/restyle-encoder/

1. Introduction

Recently, Generative Adversarial Networks (GANs) have grown in popularity thanks to their ability to synthesize images of high visual quality and diversity. Beyond their phenomenal realism and fidelity on numerous domains, recent works have shown that GANs, e.g., StyleGAN [26, 27, 25], effectively encode semantic information in their latent spaces [18, 37, 23]. Notably, it has been shown that StyleGAN's learnt latent space $\mathcal{W}$ has disentanglement properties [11, 37, 47] which allow one to perform extensive image manipulations by leveraging a well-trained StyleGAN generator. Such manipulations, however, have often been applied to synthetic images generated by the GAN itself. To apply such edits on *real* images, one must first *invert* the given image into StyleGAN's latent space. That is, retrieve the latent code w such that passing w to the pre-trained StyleGAN generator returns the original image. To do so, it has become common practice to invert real images into an extension of $\mathcal{W}$, denoted $\mathcal{W}+$ [3].

Input Iterative Outputs ⟶

Figure 1. Different from conventional encoder-based inversion techniques, our residual-based ReStyle scheme incorporates an iterative refinement mechanism to progressively converge to an accurate inversion of real images. For each domain, we show the input image on the left followed by intermediate inversion outputs.

Previous works have explored learning-based inversion approaches and train encoders to map a given real image into its corresponding latent code [12, 33, 51, 17, 36, 41]. Compared to per-image latent vector optimization [30, 12, 3, 4, 27], encoders are significantly faster, as they invert using a single forward pass, and converge to areas of the latent space which are more suitable for editing [51, 41]. However, in terms of reconstruction accuracy, there remains a significant gap between learning-based and optimization-based inversion methods. Hence, while significant progress has been made in learning-based inversions, designing a proper encoder and training scheme remains a challenge with many works still resorting to using a per-image optimization.

Recognizing that obtaining an accurate inversion in a single shot is difficult, in this work, we introduce a novel encoder-based inversion scheme tasked with encoding real images into the extended $\mathcal{W}+$ StyleGAN latent space. Unlike typical encoder-based inversion methods that infer the input's inverted latent code using a single forward pass, our scheme introduces an iterative feedback mechanism.

Specifically, the inversion is performed using several forward passes by feeding the encoder with the output of the previous iteration along with the original input image. This allows the encoder to leverage knowledge learned in previous iterations to focus on the relevant regions needed for achieving an accurate reconstruction of the input image. Viewing this formulation in terms of the latent space, our *residual encoder* is trained to predict the residual, or an offset, between the current latent code and the new latent code at each step. Doing so allows the encoder to progressively converge its inversion toward the target code and reconstruction, see Figure 1. Note also that the inversion is predicted solely using the encoder with *no* per-image optimization performed thereafter.

In a sense, our inversion scheme, named *ReStyle*, can be viewed as *learning* to perform a *small* number of steps (e.g., 10) in a residual-based manner within the latent space of a pre-trained unconditional generator. ReStyle is generic in the sense that it can be applied to various encoder architectures and loss objectives for the StyleGAN inversion task.

We perform extensive experiments to show that ReStyle achieves a significant improvement in reconstruction quality compared to standard feed-forward encoder-based approaches. This is achieved with a negligible increase in inference time, which is still an order of magnitude faster than the time-costly optimization-based inversion.

We also analyze the iterative nature of our approach. Specifically, we first demonstrate which image regions are refined at each iterative feedback step demonstrating that our scheme operates in a coarse-to-fine manner. Second, we show that the absolute magnitude of change occurring at each step decreases, with the predicted residuals converging after only a small number of steps.

To demonstrate the generalization of ReStyle beyond the StyleGAN inversion task and its appealing properties compared to current inversion techniques, we continue our analysis by exploring the robustness of our scheme on downstream tasks and special use-cases. To this end, we perform latent space manipulations [18, 37, 38] on the inverted latent codes to see if the embeddings are semantically meaningful. We then explore an *encoder bootstrapping* technique allowing one to leverage two well-trained encoders to obtain a more faithful translation of a given real image.

2. Background and Related Works

The idea of employing an iterative refinement scheme is not new. Carreira *et al*. [8] introduced an iterative feedback mechanism for human pose estimation. Other works have proposed using iterative refinement for optical flow [22], object pose estimation [44, 20], object detection [35], and semantic segmentation [49] among other tasks. To the best of our knowledge, we are the first to adopt an iterative refinement approach for a learned inversion of real images.

2.1. GAN Inversion

The task of *GAN Inversion* was first introduced by Zhu *et al*. [52] for projecting real images into their latent representations. In their pioneering work, the authors demonstrate how performing such an inversion enables one to leverage the semantics of the GAN's latent space for performing various image manipulation tasks. Some works [52, 30, 12, 3, 4, 27, 40] approach this task by directly optimizing the latent vector to minimize the reconstruction error for a given image. These works typically achieve high reconstruction quality but require several minutes per image. Other approaches design an encoder [52, 33, 51, 17, 36, 41] to learn a direct mapping from a given image to its corresponding latent vector. While these methods are substantially more efficient than pure optimization, they typically achieve inferior reconstruction quality. Attempting to balance this trade-off, some works have additionally proposed a hybrid approach and combine the two by using an encoder for initializing the optimization [52, 7, 17, 51]. We refer the reader to Xia *et al*. [45] for a comprehensive survey on GAN inversion.

2.2. Latent Space Embedding via Learned Encoders

To perform image manipulations on real images, methods typically follow an "invert first, edit later" approach. There, an image is first embedded into its corresponding latent code, which is then edited in a semantically meaningful manner. Diverging from the above approach, recent works [32, 36, 6, 9] have proposed end-to-end methods for leveraging the high-quality images generated by GANs for various image-to-image translation and image editing tasks. In these works, a real input image is directly encoded into the transformed latent code which is then fed into the generator to obtain the desired transformed image. By training an encoder with some additional constraint, these works are able to directly solve various tasks without the need for inverting the images beforehand. Other works [46] have explored utilizing features produced by a learned StyleGAN encoder for solving various down-stream tasks such as face verification and layout prediction. These works further emphasize the advantage of training a powerful encoder into the latent space of a pre-trained unconditional generator.

2.3. Latent Space Manipulation

With the recent advancements in unconditional image synthesis through GANs [16], many works have proposed diverse methods for understanding and controlling their latent representations with the goal of performing extensive image manipulations. Various works [14, 15, 37] use fully-supervised approaches to find latent directions corresponding to various attributes such as age, gender, and expression. On the other end of the supervision spectrum, several methods [18, 42, 43] find directions in a completely unsupervised manner. Others have explored techniques that go be-

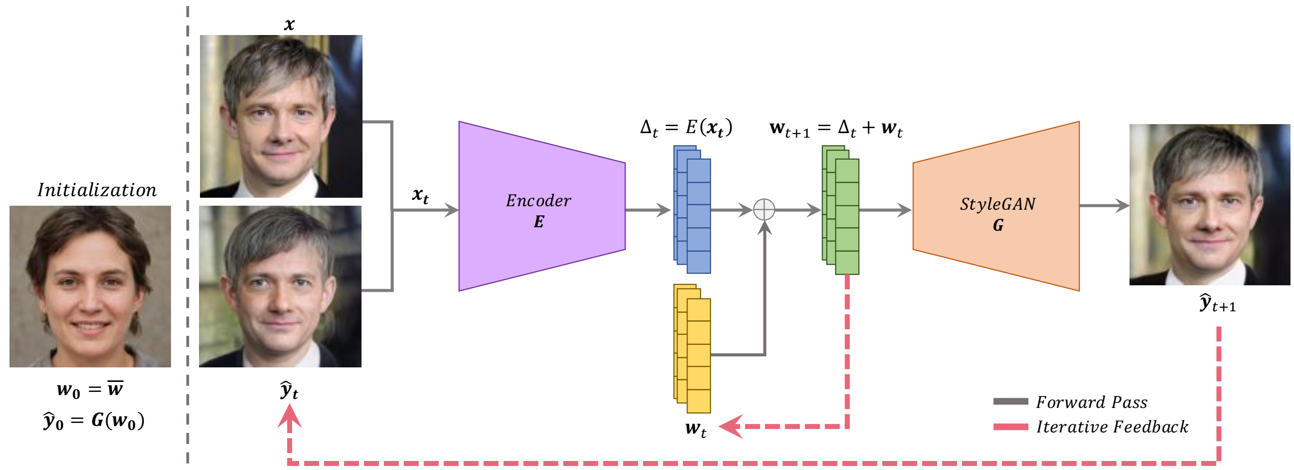


Figure 2. ***Our ReStyle iterative inversion scheme***. Given an input image $\mathbf{x}$, the scheme is initialized with the average latent code $\mathbf{w}_0$ and its corresponding image $\hat{\mathbf{y}}_0$. Consider step t. ReStyle operates on an extended input obtained by concatenating $\mathbf{x}$ with the image $\hat{\mathbf{y}}_t$ corresponding to the current inversion prediction $\mathbf{w}_t \in \mathcal{W}+$ (shown in yellow). The encoder E is then tasked with predicting a *residual* latent code, $\Delta_t \in \mathcal{W}+$ (shown in blue). The predicted residual is then added to the previous latent code $\mathbf{w}_t$ to obtain the updated latent code prediction $\mathbf{w}_{t+1}$ (shown in green). Finally, passing the newly computed latent code to the generator G results in an updated reconstruction $\hat{\mathbf{y}}_{t+1}$, which is then passed as input in the following step. During training, the loss objectives are computed at each forward pass with back-propagation performed accordingly. A similar multi-step process is performed during inference.

yond a linear traversal of the latent space. Tewari *et al*. [39] employ a pre-trained 3DMM to learn semantic face edits. Shen *et al*. [38] learn versatile edit directions through the eigenvector decomposition of the generator weights. Abdal *et al*. [5] learn non-linear paths via normalizing flows conditioned on a target attribute. By designing an efficient and accurate inversion method, one is able to leverage the above works to perform diverse manipulations on real images.

3. Method

We now turn to describe our ReStyle scheme in detail. Recall that our goal is to train an encoder tasked with inverting real images into the StyleGAN latent space. We note that the ReStyle scheme is generic in the sense that it can be applied to different encoder architectures and sets of loss objectives. Therefore, we do not name a specific encoder-based method trained using ReStyle and instead describe our inversion methodology in a general fashion. Let $E(\cdot)$ and $G(\cdot)$ denote our StyleGAN encoder and generator, respectively. Given a source image $\mathbf{x}$, our goal is to generate an image $\hat{\mathbf{y}} = ReStyle(\mathbf{x})$ such that $\hat{\mathbf{y}} \approx \mathbf{x}$.

Using typical encoder-based inversion methods, $\hat{\mathbf{y}}$ is simply computed using a single forward pass through E and G: $\hat{\mathbf{y}} = G(E(\mathbf{x}))$. Diverging from these conventional encoders, given an input image $\mathbf{x}$, ReStyle performs N steps to predict the image inversion and corresponding reconstruction. For clarity, we define a *step* to be a single forward pass through E and G. A full training *iteration* is then defined as a set of N steps performed on a batch of images. Note that the number of steps in each iteration is small (e.g., $N < 10$).

A high-level illustration of the ReStyle methodology is shown in Figure 2. At each step t, ReStyle operates on an expanded input by concatenating $\mathbf{x}$ with the current prediction of the reconstructed image $\hat{\mathbf{y}}_t$:

$$\mathbf{x}_t := \mathbf{x} \parallel \hat{\mathbf{y}}_t. \tag{1}$$

Given the extended 6-channel input $\mathbf{x}_t$, the encoder E is tasked with computing a residual code Δ_t, with respect to the latent code predicted in the previous step. That is,

$$\Delta_t := E(\mathbf{x}_t). \tag{2}$$

The new prediction for the latent code corresponding to the inversion of the input image $\mathbf{x}$ is then updated as:

$$\mathbf{w}_{t+1} \leftarrow \Delta_t + \mathbf{w}_t. \tag{3}$$

This new latent $\mathbf{w}_{t+1}$ is passed through the generator G to obtain the updated prediction of the reconstructed image:

$$\hat{\mathbf{y}}_{t+1} := G(\mathbf{w}_{t+1}). \tag{4}$$

Finally, the updated prediction $\hat{\mathbf{y}}_{t+1}$ is set as the additional input channels in the next step, as defined by Equation 1.

This procedure is initialized with an initial guess $\mathbf{w}_0$ and corresponding image $\hat{\mathbf{y}}_0$. In our experiments, these are set to be the generator's average style vector and its corresponding synthesized image, respectively.

Recall that ReStyle is applied over some encoder-based inversion method. During training, we employ the set of losses defined by this method for training the encoder network E (while the pre-trained generator G remains fixed).

These losses are computed at each forward pass with the encoder weights updated accordingly via back-propagation (i.e., back-propagation occurs N times per batch). During inference, the same multi-step process (without the loss computation) is performed to compute the image inversion.

Observe that constraining the encoder to invert the given image in a single forward pass, as is typically done, imposes a hard constraint on the training process. Conversely, our training scheme can, in a sense, be viewed as relaxing this constraint. In the above formulation, the encoder learns how to best take *several* steps in the latent space with respect to an initial guess $\mathbf{w}_0$ guided by the output obtained in the previous step. This relaxed constraint allows the encoder to iteratively narrow down its inversion to the desired target latent code in a self-correcting manner. One may also view the ReStyle steps in a similar manner to the steps of optimization, with the key difference that here the steps are *learned* by the encoder.

3.1. Encoder Architecture

To show that the presented training scheme can be applied to different encoder architectures and loss objectives, we apply the ReStyle scheme on the state-of-the-art encoders from Richardson *et al*. [36] (pSp) and Tov *et al*. [41] (e4e). These two encoders employ a Feature Pyramid Network [29] over a ResNet [19] backbone and extract the style features from three intermediate levels. Such a hierarchical encoder is well-motivated for well-structured domains such as the human facial domain in which the style inputs can be roughly divided into three levels of detail. With that, we find such a design to have a negligible impact on less-structured, multi-modal domains while introducing an increased overhead during inference. Moreover, we find that the multi-step nature of ReStyle alleviates the need for such a complex encoder architecture.

We therefore choose to design a simpler variant of the pSp and e4e encoder. Rather than extracting the style features from three intermediate levels along the encoder, all style vectors are extracted from the final 16×16 feature map. Given a StyleGAN generator with k style inputs, k different *map2style* blocks [36] are then used to gradually down-sample the feature map to obtain the corresponding 512-dimensional input style vector. A high-level overview of the architectural design choice is provided in Figure 3 with more details provided in Appendix A.1. For completeness, we provide an ablation study in Appendix C to validate this design choice.

4. Experiments

Datasets. We conduct extensive evaluations on a diverse set of domains to illustrate the generalization of our approach. For the human facial domain we use the FFHQ [26] dataset for training and the CelebA-HQ [31, 24] test set for evaluation. For the cars domains, we use the Stanford Cars [28] dataset for training and evaluation. Additional evaluations are performed on the LSUN [48] Horse and Church datasets as well as the AFHQ Wild [10] dataset.

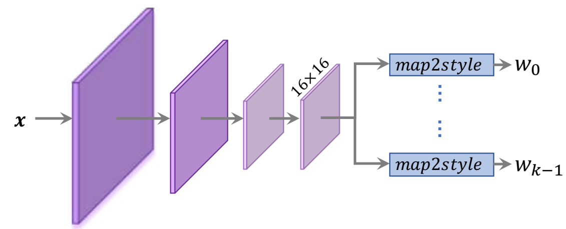


Figure 3. ***Our simplified encoder architecture.*** All k input style vectors of the generator are extracted from the encoder's final 16×16 feature map which is passed through k *map2style* blocks [36].

Baselines. Throughout this section, we explore and analyze encoder-based, optimization-based, and hybrid inversion techniques. For encoder-based methods, we compare our iterative ReStyle approach with the IDInvert encoder from Zhu *et al*. [51], pSp from Richardson *et al*. [36], and e4e from Tov *et al*. [41]. For optimization-based methods, we compare our results with the inversion technique from Karras *et al*. [27]. For each of the above encoder-based inversion methods we also perform optimization on the resulting latents for a comparison with hybrid approaches. Additional details can be found in Appendix A.2.

ReStyle Architectures and Training Details. For the human facial domain, we employ the ResNet-IRSE50 architecture from Deng *et al*. [13] pre-trained for facial recognition. For all other domains, we use a simple ResNet34 architecture pre-trained on ImageNet. These architectures have a modified input layer to accommodate the 6-channel input used by ReStyle. All results in this work were obtained using StyleGAN2 [27] generators.

Throughout this section, we apply ReStyle on pSp [36] and e4e [41] using the loss objectives and training details (e.g., batch size, loss weights) as originally defined in their respective works. Note that when applying ReStyle, we utilize the simplified encoder architecture presented in Section 3.1 for extracting the image inversion. All ReStyle encoders are trained using $N = 5$ steps per batch.

4.1. Comparison with Inversion Methods

We first compare ReStyle with current state-of-the-art StyleGAN inversion techniques. While per-image optimization techniques have achieved superior image reconstruction compared to learning-based approaches, they come with a significantly higher computational cost. Therefore, when analyzing the inversion approaches, it is essential to measure reconstruction quality with respect to inference time, resulting in a so-called *quality-time trade-off*.

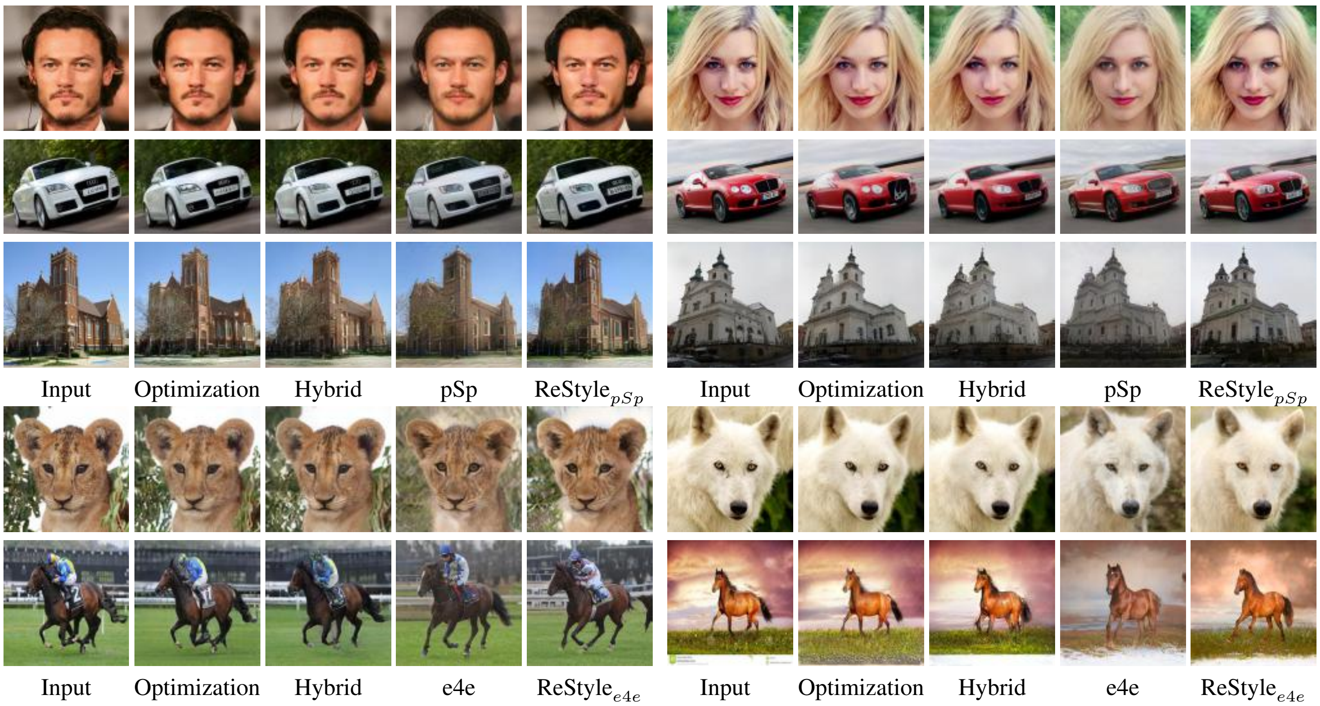


Figure 4. ***Qualitative Comparison.*** We compare various encoder-based and optimization-based inversion methods with our ReStyle scheme applied over pSp [36] and e4e [41] (denoted by ReStyle$_{pSp}$ and ReStyle$_{e4e}$). Hybrid results are obtained by performing optimization on the latent codes obtained by the adjacent encoder. Additional comparisons in the supplementary materials. Best viewed zoomed-in.

Qualitative Evaluation. We begin by showing a qualitative comparison of ReStyle and the alternative inversion approaches across various domains in Figure 4. We refer the reader to Appendix F for a large-scale gallery of additional comparisons and results. It is important to emphasize that we do not claim to achieve superior reconstruction quality over optimization. The comparison instead serves to show that ReStyle is visually comparable to the latter. Attaining comparable reconstruction quality with a significantly lower inference time places ReStyle at an appealing point on the quality-time trade-off curve.

With that, we do note the improved reconstruction obtained by ReStyle in comparison with the pSp and e4e encoders, especially in the preservation of fine details. For example in the comparison with pSp (the top three rows), observe the collar of the man in the top left comparison and the hair of the woman in the top right comparison. Similarly, observe the Audi symbol and the license plate in the car comparison on the left-hand side. In the comparison with e4e (the bottom two rows), observe how ReStyle better captures the background of the wild animals and the pose of the horse.

Quantitative Evaluation. We now perform a quantitative comparison of the different inversion approaches across various data domains. To measure both pixel-wise and perceptual similarities we apply the commonly-used L_2 and LPIPS [50] metrics. In addition, for the human facial domain, to measure each method's ability to faithfully preserve identity, we measure the identity similarity between the reconstructed images and their source using the state-of-the-art CurricularFace [21] facial recognition method. To illustrate the quality-time trade-off between the different methods we additionally measure each method's inference time per image. As mentioned, both optimization-based inversion and ReStyle can be viewed as a continuous curve on this quality-time graph — with each additional step, we attain improved reconstruction quality at the cost of additional inference time.

To provide a complete comparison of all inversion methods, we construct a quality-time graph for each domain. Such graphs can be visualized in Figure 5. There, we illustrate the reconstruction trade-off for the human facial, cars, and churches domains with each evaluated via a different reconstruction metric. To form each graph, we performed the following evaluations for each inversion technique. For each encoder-based inversion, we ran a single forward pass to obtain the reconstruction image, resulting in a single point on the graph. For measuring the optimization technique from Karras *et al.* [27], we invert the input image using a varying number of steps from 1 optimization step up to $1,500$ steps. For the hybrid approaches, given the computed latent codes obtained from the corresponding en-

Figure 5. ***Quantitative comparison.*** We compare ReStyle with current state-of-the-art optimization-based and encoder-based methods by analyzing reconstruction via three evaluation metrics — ID similarity for human faces, L2 loss for cars, and LPIPS loss for churches — while measuring each method's inference time. Each encoder-based method is represented using a $\star$ symbol. The corresponding hybrid method is marked using a dashed line of the same color with ReStyle applied over the base method shown using a solid line of the same color. Optimization [27] results are shown using a dashed green line. Methods based on pSp [36] are shown in red with methods based on e4e [41] shown in blue. Finally, results obtained using IDInvert [51] are shown in orange. Note that both axes are shown in log-scale.

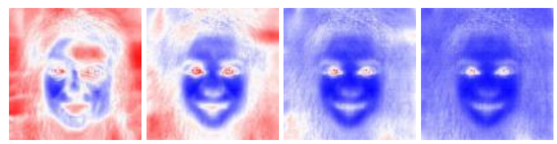
coder, we performed optimization with an increasing number of steps between 1 and 500 steps. Finally, for our two ReStyle encoders, we performed up to 10 feedback loops.

We begin by analyzing the human facial domain. Compared to the pSp and e4e encoders, our ReStyle variants match or surpass their counterparts in identity preservation. More notably, while optimization techniques achieve improved identity similarity compared to ReStyle, they require $\approx 20\times$ more inference time to match the reconstruction attained by ReStyle. A similar trade-off can be observed in the cars domain where now the advantage of ReStyle over typical encoders is more pronounced when evaluating the L_2 loss of the reconstructions. In the highly unstructured churches domain, ReStyle applied over pSp nearly matches both optimization and hybrid techniques in reconstruction quality with a significantly lower inference time. Observe that the first output of ReStyle may be *worse* than that of a conventional encoder due to the more relaxed training formulation of ReStyle as it is trained to perform multiple steps at inference. With that, ReStyle quickly matches or surpasses the quality of single-shot encoders.

These comparisons point to the appealing nature of ReStyle: although optimization typically achieves superior reconstruction, ReStyle offers an excellent balance between reconstruction quality and inference time. Comparisons for all domains and metrics can be found in Appendix E.

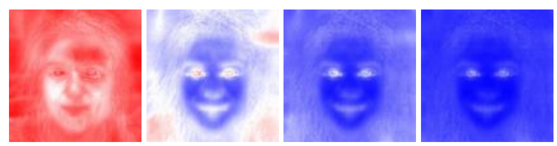
4.2. ReStyle Analysis

In this section, we explore various aspects of ReStyle to gain a stronger understanding of its behavior and attain key insights into its efficiency. Specifically, we analyze the main details focused on by the encoder at each step and analyze the number of steps needed for convergence during inference. Additional analyses in both the image space and latent space can be found in Appendix B.



$1 \rightarrow 2$ $2 \rightarrow 3$ $3 \rightarrow 4$ $4 \rightarrow 5$

Figure 6. ***Which regions are refined at each step?*** In each sub-image, we display a heatmap showing which image regions change the most (in red) and which regions change the least (in blue) between the specified iterations.



$1 \rightarrow 2$ $2 \rightarrow 3$ $3 \rightarrow 4$ $4 \rightarrow 5$

Figure 7. ***How many steps are needed for convergence?*** Similar to Figure 6, with the difference that here all images are normalized with respect to each other. As shown, the magnitude of change decreases with each step.

What's the focus? We begin by exploring which regions of the image are focused on by the encoder at each step during inference. To do so, we consider the human facial domain. For each step t and each input image $\mathbf{x}$, we compute the squared difference in the image space between the generated images at steps t and $t-1$. That is, we compute $d = \left\|\mathbf{y}_t - \mathbf{y}_{t-1}\right\|_2$ where $\mathbf{y}_t$ is defined in Equation 4.

Averaging over all $2,800$ test samples we obtain the average image difference between the two steps. Finally, we normalize the average image to the range $[0, 1]$ and visualize the regions of the image that incur the most change at the current step t. We visualize this process in Figure 6 showing ReStyle's incremental refinements. As can be seen, in the early steps the encoder focuses on refining the background and pose while in subsequent steps the encoder moves its focus to adjusting finer details along the eyes and hair.

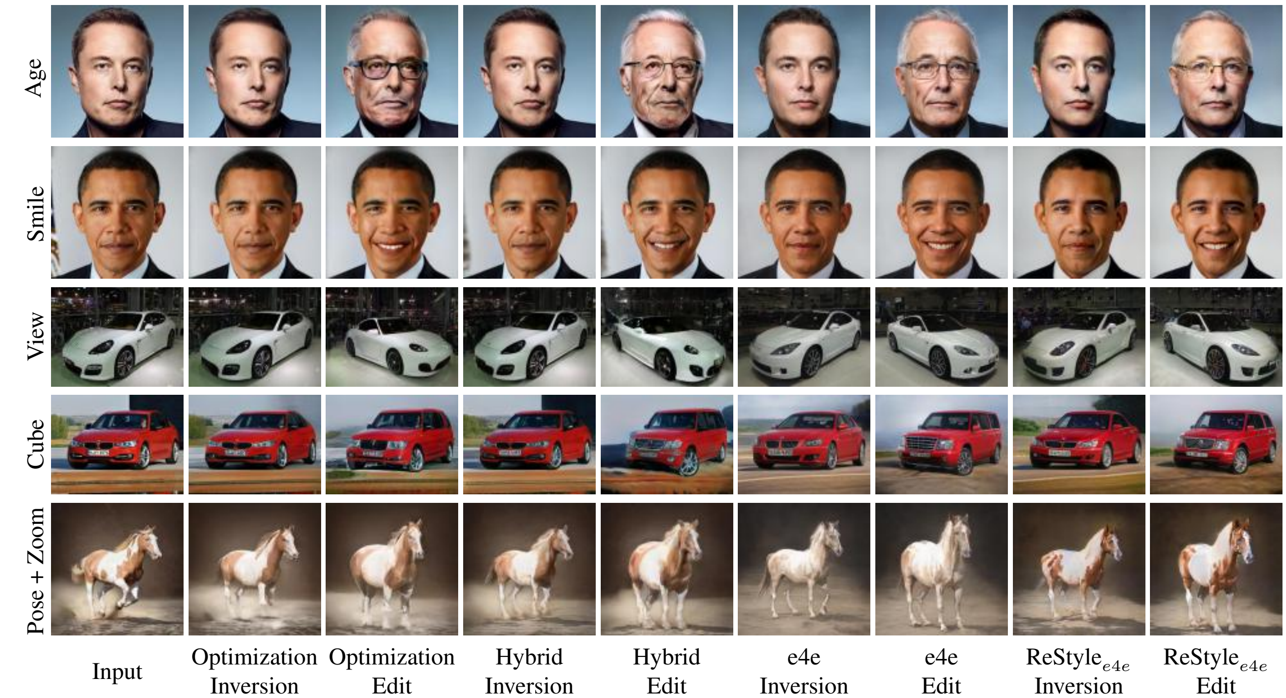


Figure 8. ***Editing comparison.*** Applying edits on inversions of several methods. For performing the edits in the human facial domain we use InterFaceGAN [37], for the cars domain we use GANSpace [18], and for the horse domain we use SeFa [38]. Image credits: [1],[2].

In Figure 6 we show only the magnitude of change *within* each step. That is, the absolute magnitude of change may vary between the different steps. To show that the overall amount of change decreases with each step, we refer the reader to Figure 7. There, all images are normalized with respect to each other allowing one to see how the largest changes occur in the first step and decrease thereafter.

In a sense, the encoder operates in a coarse-to-fine manner, beginning by concentrating on low-frequency details which are then gradually complemented by adjusting high frequency, fine-level details.

ReStyle's iterative progress. We now turn to Figure 9 and show how the reconstruction quality incrementally improves with each additional step of ReStyle. Specifically, observe how ReStyle$_{pSp}$ is able to gradually improve the reconstruction of the highly non-frontal input image in the top row. Similarly, notice how ReStyle$_{e4e}$ is able to iteratively refine the posture of the horse rider and capture the skewed structure of the church building.

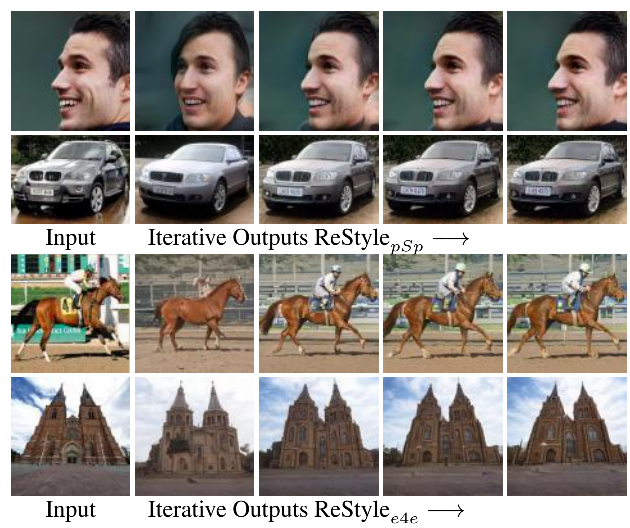


Figure 9. ***The iterative refinement of ReStyle.*** Given the input image on the left, we visualize the intermediate outputs of ReStyle applied over pSp [36] and e4e [41].

4.3. Editability via Latent Space Manipulations

Previous works [51, 53, 41] have discussed the importance of evaluating the editability of inversion methods. Here, we show that the editability achieved by ReStyle is comparable to the editability of the single-pass encoder which ReStyle is applied over. Since e4e is designed specifically for image manipulations, we choose to show that inversions obtained by combining e4e with ReStyle are still editable. We show visual examples in Figure 8. Compared to e4e, ReStyle is able to better reconstruct the input while still allowing for realistic edits. Notably, observe the more plausible edits obtained on ReStyle's inversions compared to those obtained via optimization. For example, notice the artifacts in the age edit and in the front bumpers of the car edits when applied on the optimization inversions.

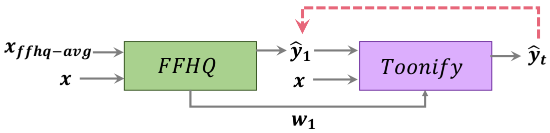


Figure 10. ***Encoder bootstrapping overview.***

4.4. Encoder Bootstrapping

Finally, we explore a new concept, which we call *encoder bootstrapping*. To help motivate this idea, let us consider the image toonification task in which we would like to translate real face images into their toonified, or animated, version. Pinkney *et al*. [34] propose solving this image-to-image task by projecting each real input image to its closest toon image residing in the latent space of a toon StyleGAN, which is obtained via fine-tuning the FFHQ StyleGAN generator. In a similar sense, ReStyle can be applied over pSp to solve this task. Here, the ReStyle process is initialized with the average toon latent code and its corresponding synthesized image. Then, N steps are performed to translate the image to its toonified equivalent.

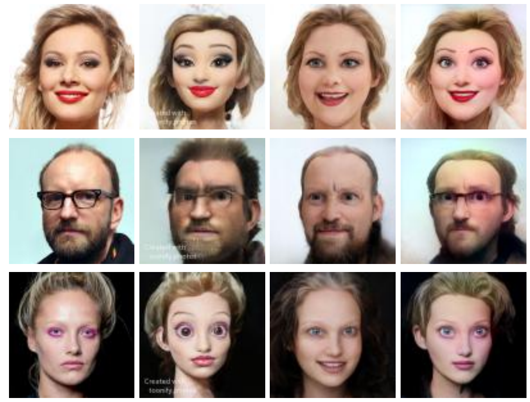
With encoder bootstrapping, we take a slightly different approach. Rather than initializing the iterative process using the average toon image, we first pass the given real image to an encoder tasked with embedding real images into the latent space of a StyleGAN trained on FFHQ. Doing so will result in an inverted code $\mathbf{w}_1$ and reconstructed image $\hat{\mathbf{y}}_1$. This inverted code and reconstructed image are then taken to initialize the toonification translation using ReStyle. This idea is illustrated in Figure 10. Notice that this technique is possible thanks to the residual nature of ReStyle. By using the FFHQ encoder to obtain a better initialization, the encoder is able to more easily learn a proper residual for translating the input image while more faithfully preserving identity.

We compare several real-to-toon variants in Figure 11. Observe how bootstrapping the toonification process with the FFHQ inverted latent code results in translations that are better able to capture the input characteristics and toonify style. Specifically, observe the ability of the bootstrapped variant to better preserve make-up, eyeglasses, hairstyle, and expression. In Figure 12, we visualize the inverted real image used to initialize the toonify encoder followed by the toonified outputs generated by ReStyle.

The success of the bootstrapping technique is intriguing as it is not immediately clear why the inverted latent code in the FFHQ latent space results in a meaningful code in the toonify latent space. We refer the reader to Appendix D for an analysis of this occurrence.

5. Conclusions

In our work, we have focused on improving the inversion accuracy of encoders and presented a new scheme for training and deploying GAN inversion encoders. Instead of predicting the inversion in one shot, we perform multiple forward passes, which more accurately and quickly converge to the target inversion. In a sense, this scheme allows the encoder to *learn* how to efficiently guide its convergence to the desired inversion. Moreover, the encoder is trained on a larger and richer set of images, which consist not only of the original dataset itself, but also utilizes the intermediate reconstructions of the images. We showed that this scheme improves the reconstruction accuracy of current state-of-the-art encoders. This improvement is especially noticeable for familiar human faces like those of celebrities, but (as we have shown) our scheme attains improved results across a diverse set of domains. While this work focused on StyleGAN, we believe our scheme to be generalizable to additional generators. We leave this as an interesting direction for future work. We also explored pairing ReStyle with an encoder bootstrapping technique for the image toonification task. We view this bootstrapping idea and the resulting transformations to be intriguing and may further open the door for additional tasks, leveraging the nature of our proposed residual-based, iterative scheme.



Input Toonify [34] ReStyle$_{pSp}$ ReStyle$^{BS}_{pSp}$

Figure 11. ***Toonify comparison***. Applying ReStyle with bootstrapping, denoted ReStyle$^{BS}_{pSp}$, is able to better preserve the identity characteristics of the input real image.

Input Inverted Iterative Outputs $\longrightarrow$

Figure 12. For each input, we show the inverted image obtained after a single step of our ReStyle$_{pSp}$ FFHQ encoder followed by the iterative outputs of our ReStyle$_{pSp}$ toonify encoder.

References

[1] Barak Obama image credits: Pete Souza https://www.loc.gov/resource/ppbd.00358/. 7

[2] Elon Musk image credits: The CEO Magazine website https://www.theceomagazine.com/business/coverstory/a-touch-of-madness-elon-musk/. 7

[3] Rameen Abdal, Yipeng Qin, and Peter Wonka. Image2stylegan: How to embed images into the stylegan latent space? In *Proceedings of the IEEE international conference on computer vision*, pages 4432–4441, 2019. 1, 2

[4] Rameen Abdal, Yipeng Qin, and Peter Wonka. Image2stylegan++: How to edit the embedded images? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8296–8305, 2020. 1, 2

[5] Rameen Abdal, Peihao Zhu, Niloy Mitra, and Peter Wonka. Styleflow: Attribute-conditioned exploration of stylegan-generated images using conditional continuous normalizing flows, 2020. 3

[6] Yuval Alaluf, Or Patashnik, and Daniel Cohen-Or. Only a matter of style: Age transformation using a style-based regression model, 2021. 2

[7] Baylies. stylegan-encoder, 2019. Accessed: February 2021. 2

[8] Joao Carreira, Pulkit Agrawal, Katerina Fragkiadaki, and Jitendra Malik. Human pose estimation with iterative error feedback, 2016. 2

[9] Lucy Chai, Jonas Wulff, and Phillip Isola. Using latent space regression to analyze and leverage compositionality in gans, 2021. 2

[10] Yunjey Choi, Youngjung Uh, Jaejun Yoo, and Jung-Woo Ha. Stargan v2: Diverse image synthesis for multiple domains, 2020. 4, 11, 24

[11] Edo Collins, Raja Bala, Bob Price, and Sabine Susstrunk. Editing in style: Uncovering the local semantics of gans. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5771–5780, 2020. 1

[12] Antonia Creswell and Anil Anthony Bharath. Inverting the generator of a generative adversarial network. *IEEE transactions on neural networks and learning systems*, 30(7):1967–1974, 2018. 1, 2

[13] Jiankang Deng, Jia Guo, Niannan Xue, and Stefanos Zafeiriou. Arcface: Additive angular margin loss for deep face recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4690–4699, 2019. 4

[14] Emily Denton, Ben Hutchinson, Margaret Mitchell, and Timnit Gebru. Detecting bias with generative counterfactual face attribute augmentation. *arXiv preprint arXiv:1906.06439*, 2019. 2

[15] Lore Goetschalckx, Alex Andonian, Aude Oliva, and Phillip Isola. Ganalyze: Toward visual definitions of cognitive image properties, 2019. 2

[16] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, page 2672–2680, Cambridge, MA, USA, 2014. MIT Press. 2

[17] Shanyan Guan, Ying Tai, Bingbing Ni, Feida Zhu, Feiyue Huang, and Xiaokang Yang. Collaborative learning for faster stylegan embedding. *arXiv preprint arXiv:2007.01758*, 2020. 1, 2

[18] Erik Härkönen, Aaron Hertzmann, Jaakko Lehtinen, and Sylvain Paris. Ganspace: Discovering interpretable gan controls. *arXiv preprint arXiv:2004.02546*, 2020. 1, 2, 7, 14, 32

[19] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015. 4

[20] Shao-Kang Huang, Chen-Chien Hsu, Wei-Yen Wang, and Cheng-Hung Lin. Iterative pose refinement for object pose estimation based on rgbd data. *Sensors*, 20(15):4114, 2020. 2

[21] Yuge Huang, Yuhan Wang, Ying Tai, Xiaoming Liu, Pengcheng Shen, Shaoxin Li, Jilin Li, and Feiyue Huang. Curricularface: adaptive curriculum learning loss for deep face recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5901–5910, 2020. 5, 14

[22] Junhwa Hur and Stefan Roth. Iterative residual refinement for joint optical flow and occlusion estimation, 2019. 2

[23] Ali Jahanian, Lucy Chai, and Phillip Isola. On the "steerability" of generative adversarial networks, 2020. 1

[24] 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. 4, 11, 17, 18, 19, 26

[25] Tero Karras, Miika Aittala, Janne Hellsten, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Training generative adversarial networks with limited data, 2020. 1

[26] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4401–4410, 2019. 1, 4, 11

[27] 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. 1, 2, 4, 5, 6, 14, 31, 32, 33

[28] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13)*, Sydney, Australia, 2013. 4, 11, 20, 21, 26

[29] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection, 2017. 4

[30] Zachary C Lipton and Subarna Tripathi. Precise recovery of latent vectors from generative adversarial networks. *arXiv preprint arXiv:1702.04782*, 2017. 1, 2

[31] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild, 2015. 4, 11, 17, 18, 19, 26

[32] Yotam Nitzan, Amit Bermano, Yangyan Li, and Daniel Cohen-Or. Disentangling in latent space by harnessing a pretrained generator. *arXiv preprint arXiv:2005.07728*, 2020. 2

[33] Stanislav Pidhorskyi, Donald A Adjeroh, and Gianfranco Doretto. Adversarial latent autoencoders. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14104–14113, 2020. 1, 2

[34] Justin N. M. Pinkney and Doron Adler. Resolution dependent gan interpolation for controllable image synthesis between domains, 2020. 8, 13

[35] Rakesh N Rajaram, Eshed Ohn-Bar, and Mohan M Trivedi. Refinenet: Iterative refinement for accurate object localization. In *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, pages 1528–1533. IEEE, 2016. 2

[36] 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*, 2021. 1, 2, 4, 5, 6, 7, 11, 13, 14, 17, 19, 20, 22, 24

[37] Yujun Shen, Jinjin Gu, Xiaoou Tang, and Bolei Zhou. Interpreting the latent space of gans for semantic face editing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9243–9252, 2020. 1, 2, 7, 14, 31, 33

[38] Yujun Shen and Bolei Zhou. Closed-form factorization of latent semantics in gans. *arXiv preprint arXiv:2007.06600*, 2020. 2, 3, 7, 14

[39] Ayush Tewari, Mohamed Elgharib, Gaurav Bharaj, Florian Bernard, Hans-Peter Seidel, Patrick Pérez, Michael Zollhöfer, and Christian Theobalt. Stylerig: Rigging stylegan for 3d control over portrait images. *arXiv preprint arXiv:2004.00121*, 2020. 3

[40] Ayush Tewari, Mohamed Elgharib, Mallikarjun B R., Florian Bernard, Hans-Peter Seidel, Patrick Pérez, Michael Zollhöfer, and Christian Theobalt. Pie: Portrait image embedding for semantic control, 2020. 2

[41] Omer Tov, Yuval Alaluf, Yotam Nitzan, Or Patashnik, and Daniel Cohen-Or. Designing an encoder for stylegan image manipulation, 2021. 1, 2, 4, 5, 6, 7, 11, 14, 18, 21, 23, 25

[42] Andrey Voynov and Artem Babenko. Unsupervised discovery of interpretable directions in the gan latent space. *arXiv preprint arXiv:2002.03754*, 2020. 2

[43] Binxu Wang and Carlos R Ponce. A geometric analysis of deep generative image models and its applications. In *International Conference on Learning Representations*, 2021. 2

[44] Chen Wang, Danfei Xu, Yuke Zhu, Roberto Martín-Martín, Cewu Lu, Li Fei-Fei, and Silvio Savarese. Densefusion: 6d object pose estimation by iterative dense fusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3343–3352, 2019. 2

[45] Weihao Xia, Yulun Zhang, Yujiu Yang, Jing-Hao Xue, Bolei Zhou, and Ming-Hsuan Yang. Gan inversion: A survey, 2021. 2

[46] Yinghao Xu, Yujun Shen, Jiapeng Zhu, Ceyuan Yang, and Bolei Zhou. Generative hierarchical features from synthesizing images, 2020. 2

[47] Ceyuan Yang, Yujun Shen, and Bolei Zhou. Semantic hierarchy emerges in deep generative representations for scene synthesis, 2020. 1

[48] Fisher Yu, Ari Seff, Yinda Zhang, Shuran Song, Thomas Funkhouser, and Jianxiong Xiao. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop, 2016. 4, 11, 16, 22, 23, 25

[49] Chi Zhang, Guosheng Lin, Fayao Liu, Rui Yao, and Chunhua Shen. Canet: Class-agnostic segmentation networks with iterative refinement and attentive few-shot learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5217–5226, 2019. 2

[50] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric, 2018. 5

[51] Jiapeng Zhu, Yujun Shen, Deli Zhao, and Bolei Zhou. In-domain gan inversion for real image editing. *arXiv preprint arXiv:2004.00049*, 2020. 1, 2, 4, 6, 7, 11, 14, 15, 26

[52] Jun-Yan Zhu, Philipp Krähenbühl, Eli Shechtman, and Alexei A Efros. Generative visual manipulation on the natural image manifold. In *European conference on computer vision*, pages 597–613. Springer, 2016. 2

[53] Peihao Zhu, Rameen Abdal, Yipeng Qin, and Peter Wonka. Improved stylegan embedding: Where are the good latents?, 2020. 7

Appendix

In this appendix, we provide additional details and analysis to complement those provided in the main manuscript. Along with the additional details, we also perform an ablation study to validate our design choices and provide a large gallery of comparisons and results at full resolution obtained using the proposed ReStyle scheme.

A. Additional Details

A.1. The ReStyle Encoder Architecture

We begin by providing additional details regarding the ReStyle encoder architecture presented in Section 3.1. Recall that our simplified architecture is derived from the architecture used in Richardson *et al*. [36]. There, the authors employ an FPN-based architecture for encoding real images into the $\mathcal{W}+$ StyleGAN latent space. Specifically, the encoder extracts the style input vectors using three intermediate feature maps of spatial resolutions 64×64 (for inputs $0-2$), 32×32 (for inputs $3-6$), and 16×16 (for inputs $7-17$). Each style vector is extracted from their corresponding feature map using a map2style [36] block, which is a small convolutional network containing a series of 2-strided convolutions with LeakyReLU activations. This FPN-based architecture is illustrated in Figure 13.

With ReStyle, we take a simpler approach. Instead of extracting the style vectors from three intermediate feature maps along the encoder, each style input is extracted from the final 16×16 feature map and a map2style block, as illustrated in Figure 3 in the main paper. An ablation study comparing these two architectures is provided below in Appendix C.

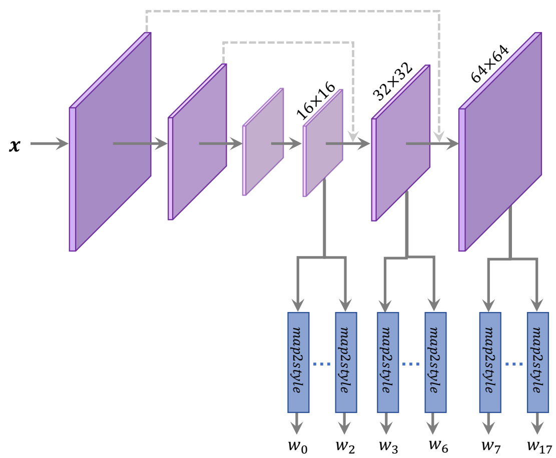


Figure 13. ***The FPN-based encoder.*** Visualization of the original FPN architecture used in Richardson *et al*. [36] and Tov *et al*. [41]. A visualization of our simplified encoder architecture is provided in the main paper.

A.2. Datasets

Here, we provide additional details regarding the datasets used in each of the evaluated domains.

Human Faces. For the human facial domain, we use all $70,000$ images from the FFHQ [26] dataset for training the ReStyle encoders. For evaluations, we use all $2,824$ test images from the CelebA-HQ [31, 24] dataset using the official train-test split.

Cars. For the cars domain, we use the Stanford Cars [28] dataset with training performed using the $8,144$ training images. For our evaluations, due to the large test set ($8,041$ images), we randomly select $1,000$ images from the test set with all metrics computed using the selected subset.

AFHQ Wild. Here, training and evaluations are performed on $4,738$ and 500 images, respectively, taken from the official AFHQ [10] Wild dataset.

Horses. From the LSUN [48] Horse dataset, we randomly select $10,000$ images to be used for training and $2,215$ images used for evaluations.

Churches. For the churches domain, we use all $126,227$ training images and 300 testing images from the official LSUN [48] Church dataset.

A.3. Baselines

In our evaluations in Sections 4.1 and 4.3, we compare ReStyle with various inversion methods. Below, we provide additional details on the encoder-based inversion baselines evaluated in the main paper.

IDInvert. We compared our ReStyle approach to the IDInvert encoder from Zhu *et al*. [51] on the human facial and cars domains. For the human facial domain, we use the official pre-trained model, which employs a StyleGAN1 [26] generator. For the cars domain, we trained IDInvert using an input resolution of 512×384 and the official StyleGAN2 generator. Note, due to the long training time required by IDInvert (over three weeks on two NVIDIA P40 GPUs), we chose to train IDInvert only on the cars domain.

pSp. For the human facial domain, we use the official pre-trained model from Richardson *et al*. [36]. For all other domains, we trained pSp using StyleGAN2 generators and default pSp hyper-parameters. During training, we also incorporated the MOCO-based similarity loss from [41] which was shown to improve reconstruction quality.

e4e. For our comparison with Tov *et al*. [41], we trained e4e on the AFHQ [10] Wild dataset using the official implementation and default e4e hyper-parameters. All other domains were evaluated using official pre-trained models.

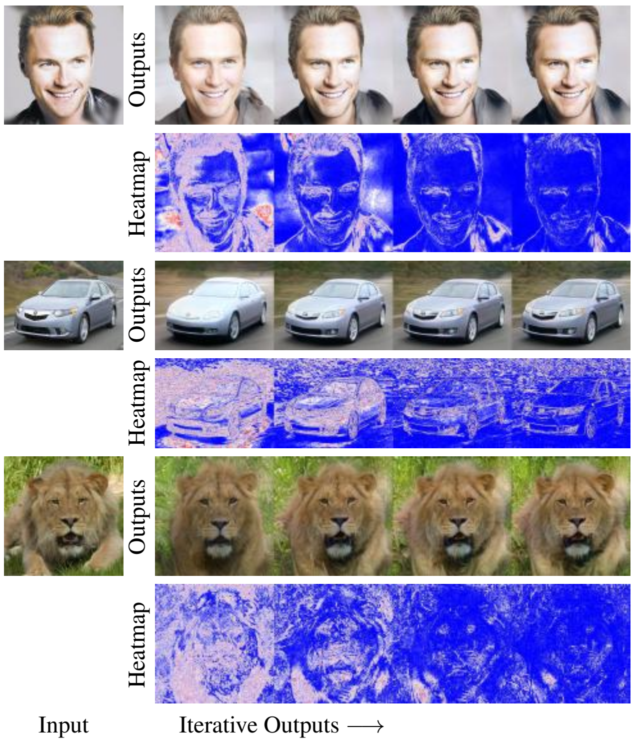


Figure 14. ***Visualizing the iterative refinement.*** For each image we visualize the iterative outputs produced by ReStyle. Below each output, we show a heatmap illustrating which image regions were altered the most at the corresponding step. Note, all heatmaps are normalized globally with respect to each other. Red regions indicate a large change with blue representing small changes in the pixel-space.

B. ReStyle Analysis

In this section, we provide additional analyses to complement those performed in the main paper (Section 4.2).

Where's the focus? (Part II) In Section 4.2 (Figures 6 and 7), we showed which image regions change the most at each inference step and showed how the magnitude of change decreases over time. There, the resulting Figures were obtained by averaging over all $2,000+$ test images. To complement these figures, we can examine this behavior while observing each image *independently* rather than averaging over all images. Consider Figure 14. There, we illustrate the intermediate outputs of ReStyle$_{pSp}$ alongside the normalized heat-maps where red denotes a large pixel change and blue denotes a small pixel change. As shown, in the early iterations, ReStyle focuses on adjusting global features such as the head pose or the car shape while in subsequent iterations finer details are adjusted.

How many steps are needed? The ReStyle analysis performed in the main paper and above explore ReStyle's behavior in the image space. Here, we explore the behavior of ReStyle in the latent space. Specifically, we analyze (i) which latent entries change the most across the inference steps and (ii) how many steps are needed for convergence. To do so, we focus on the cars domain and perform the following process. Consider some iteration t and some latent entry $\mathbf{w}_l$ where $l \in [1, k]$ and k is the number of style inputs of the generator. To compute how much the values of $\mathbf{w}_l$ change between iterations $t-1$ and t we compute the squared difference between the latent entries averaged across all test samples. That is,

$$\mathbf{d}_{l,t} = \frac{1}{N}\sum_{i=1}^{N}\left(\mathbf{w}_{l,t}^{(i)} - \mathbf{w}_{l,t-1}^{(i)}\right)^2, \tag{5}$$

$$v_{l,t} = ||\mathbf{d}_{l,t}||_2 \tag{6}$$

where $\mathbf{w}_{l,t}^{(i)}$ is the l-th latent entry of the i-th sample obtained in iteration t and $v_{l,t}$ is the L_2 norm of the average difference $\mathbf{d}_{l,t}$. Having computed the average change of each of the k latent entries across all test samples, we group the entries into the coarse, medium, and fine inputs as defined by StyleGAN and compute the average change of each group.

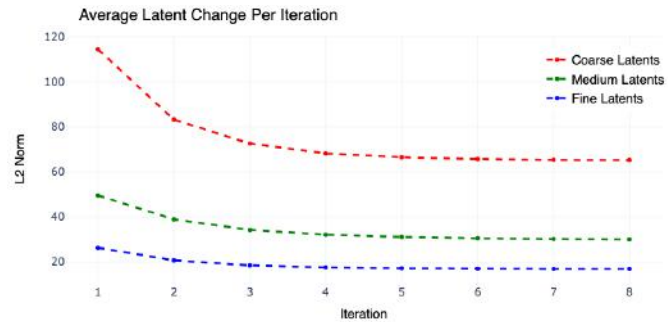


Figure 15. ***Which latents change the most?*** We plot the average magnitude of change in each group of latent inputs per step during inference. As shown, ReStyle focuses on adjusting the coarse and medium-level inputs and converges after a few steps.

In Figure 15, we plot the change along each step for each of the three groups. As can be seen, the coarse input group incurs the most change in the early iterations as the encoder focuses on refining the background and pose of the inversions. Conversely, the fine styles undergo the least change and converge after only a few steps, indicating that refining these aspects (e.g., the color) of the output image may be easier for the encoder. Overall, it can be seen that a small number of steps are needed for the encoder to converge to its final inversion prediction. Another interpretation of the above phenomenon is that the learned residuals decrease at each inference step, as is desired.

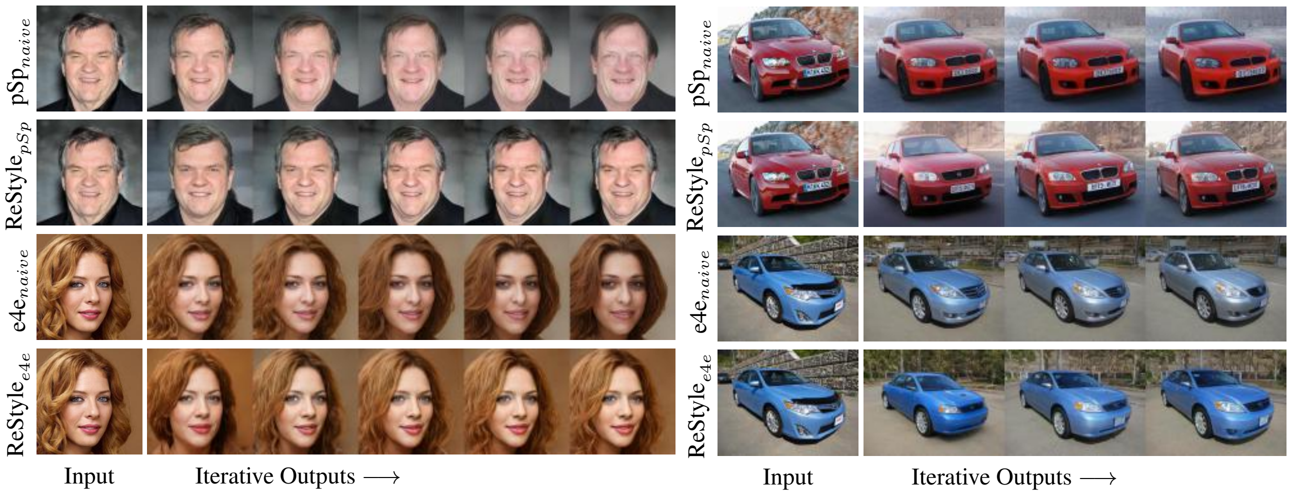


Figure 16. ***Ablation study.*** Using a pre-trained pSp or e4e encoder and simply feeding the output image multiple times leads to deteriorating reconstruction. Conversely, our dedicated ReStyle scheme incrementally improves its reconstruction outputs with each additional step.

C. Ablation Study

In this section, we validate our design choices for our ReStyle training scheme and encoder architecture.

The Iterative Training Scheme. We begin by showing that a dedicated iterative training scheme is truly needed. A natural first attempt for creating an iterative inversion scheme is simply using a pre-trained conventional encoder and passing the output image back as input multiple times. Notice that there are key differences between the above formulation and the ReStyle formulation. First, in the ReStyle scheme, we pass both the input and current output to the encoder. Second, a ReStyle encoder is trained explicitly to output a *residual* with respect to the previous latent at each step.

In Figure 16 we provide a comparison on the human facial and cars domains using both a pSp encoder and e4e encoder trained using the ReStyle formulation and naive formulation presented above. As can be seen, at each additional step, the naive formulation moves away from the reconstruction of the input image. This shows that our dedicated iterative, residual-based training scheme is indeed needed over readily-available encoders.

The ReStyle Encoder Architecture. Here, we show that our simplified encoder architectural design choice leads to a negligible difference in reconstruction quality while reducing the inference time compared to the FPN-based architecture from Richardson *et al*. [36]. Table 1 summarizes the reconstruction results across 5 domains. For each domain and encoder, we perform 5 steps during inference and display quantitative results computed on the final output. As shown, the two variants attain nearly identical reconstruction quality in terms of both L_2 and LPIPS similarity. However, our simplified architecture is more than 10% faster.

Domain	Method	↓ LPIPS	↓ MSE	↓ Runtime
Faces (1024)	ReStyle$_{fpn}$	0.03	0.14	0.538
	ReStyle$_{simple}$	0.03	**0.13**	**0.451**
Cars (512)	ReStyle$_{fpn}$	0.08	0.26	0.411
	ReStyle$_{simple}$	**0.07**	**0.25**	**0.361**
Wild (512)	ReStyle$_{fpn}$	0.06	0.23	0.413
	ReStyle$_{simple}$	0.06	**0.21**	**0.363**
Churches (256)	ReStyle$_{fpn}$	0.09	0.28	0.355
	ReStyle$_{simple}$	0.09	**0.26**	**0.298**
Horses (256)	ReStyle$_{fpn}$	0.09	0.32	0.355
	ReStyle$_{simple}$	0.09	**0.31**	**0.298**

Table 1. ***Encoder architecture ablation study.*** Ablation study on applying ReStyle to pSp using our simpler encoder architecture variant compared to the original FPN-based architecture. All results shown are computed on the final reconstruction outputs after 5 inference steps. Our simplified architecture is comparable to the FPN-variant with a reduced inference time.

D. Analyzing the Toonify Latent Space

In Section 4.4, we explored a new *encoder bootstrapping* technique for performing image toonification [34]. Rather than initializing ReStyle using the average toon latent code and corresponding toon image, the iterative process is initialized by first inverting the real image into the FFHQ StyleGAN latent space. The resulting inverted code and reconstructed image are then used to initialize the ReStyle toonify encoder. Thanks to the improved initialization from the inverted latent code, the toonify encoder is able to learn a more faithful translation of the real image into its corresponding toon image. However, as mentioned in the main text, it is not trivial to assume that the real in-

Figure 17. ***Exploring the toonify latent space.*** Synthesized images generated by passing the same randomly sampled latent code through the FFHQ StyleGAN generator and toonify StyleGAN generator. As shown, the two latent spaces are well-aligned.

verted code in the FFHQ latent space corresponds to the same identity characteristics in the toonify latent space.

To show that the above is true we randomly sample $\mathbf{w}$ vectors from the FFHQ latent space. We then pass the same $\mathbf{w}$ vector to both the FFHQ StyleGAN generator and toonify StyleGAN generator to see if the synthesized images are semantically similar. We provide several examples in Figure 17. As shown, the same latent code produces semantically similar images in both latent spaces, indicating that the two latent spaces are indeed well-aligned. As a result of the above, we gain valuable insights as to the effectiveness of the encoder bootstrapping technique in the task of image toonification.

E. Quantitative Results

In Section 4.1, we quantitatively compared various inversion methods by constructing *quality-time graphs* which allowed us to visualize how reconstruction quality changes with respect to inference time. In Figures 18 and 19 we provide the quality-time graphs for both the L_2 and LPIPS loss metrics across all five domains. For the human facial domain we additionally measure the identity similarity of the reconstructed images by using the CurricularFace [21] method for facial recognition.

F. Additional Results

The remainder of this document contains additional comparisons and results, as follows:

1. Figures 20 and 21 contain additional comparisons on the human facial domain. Additionally, Figure 22 provides a comparison on more challenging input poses and expressions.
2. Figures 23 and 24 contain additional comparisons on the cars domain.
3. Figures 25 and 26 contain additional comparisons on the churches domain.
4. Figure 27 contains additional comparisons on the wild animals domain.
5. Figure 28 contains additional comparisons on the horses domain.
6. Figure 29 contains comparisons to the IDInvert encoder from Zhu *et al*. [51] on the human facial and cars domains.
7. Figure 30 shows iterative outputs generated by ReStyle applied over pSp [36] on the human facial domain.
8. Figure 31 shows iterative outputs generated by ReStyle applied over e4e [41] on the cars domain.
9. Figure 32 shows iterative outputs generated by ReStyle applied over pSp [41] on the churches domain.
10. Figure 33 shows iterative outputs generated by ReStyle applied over e4e [41] on the horses domain.
11. Figure 34 contains editing comparisons with the optimization technique from Karras *et al*. [27] on the human facial domain obtained using InterFaceGAN [37].
12. Figure 35 contains editing comparisons with the optimization technique from Karras *et al*. [27] on the cars domain obtained using GANSpace [18].
13. Figure 36 contains editing comparisons with the optimization technique from Karras *et al*. [27] on the horses domain obtained using SeFa [38].
14. Figures 37 and 38 contain additional results on the image toonification task using the encoder bootstrapping method presented in Section 4.4.

Note, all results are shown at full-resolution: 1024×1024 for human faces, 512×512 for cars and wild animal faces, and 256×256 for churches and horses. Results are best viewed zoomed-in.

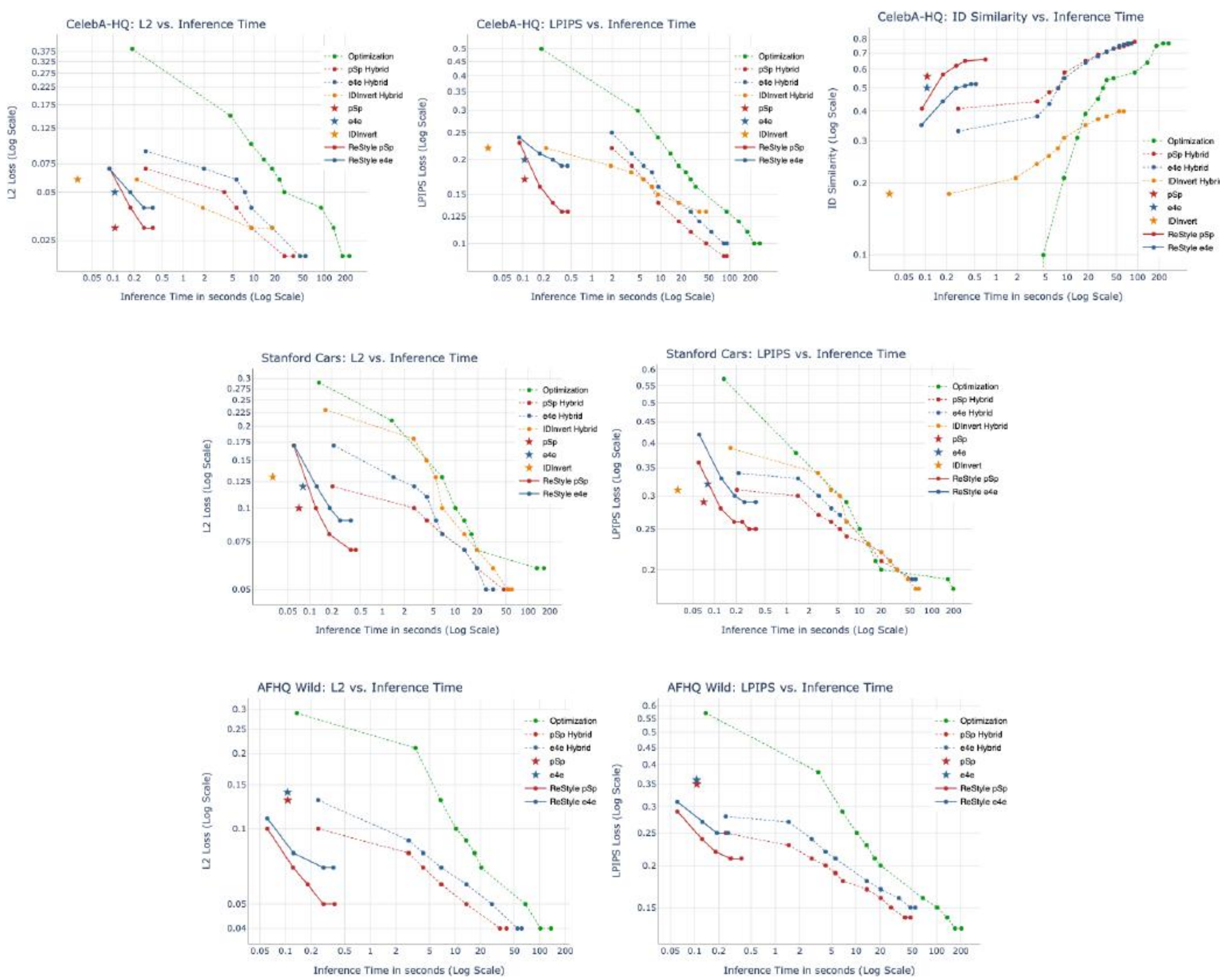


Figure 18. ***Quantitative comparison.*** We compare ReStyle with current state-of-the-art optimization-based and encoder-based methods by analyzing reconstruction via multiple evaluation metrics while measuring each method's run-time during inference. Each encoder-based method is represented using a $\star$ symbol. The corresponding hybrid method is marked using a dashed line of the same color with the ReStyle applied over the base method shown using a solid line of the same color. Optimization results are shown using a dashed green line. Methods based on pSp are shown in red with methods based on e4e shown in blue. Finally, results obtained using IDInvert [51] are shown in orange. Note that both axes are shown in log-scale.

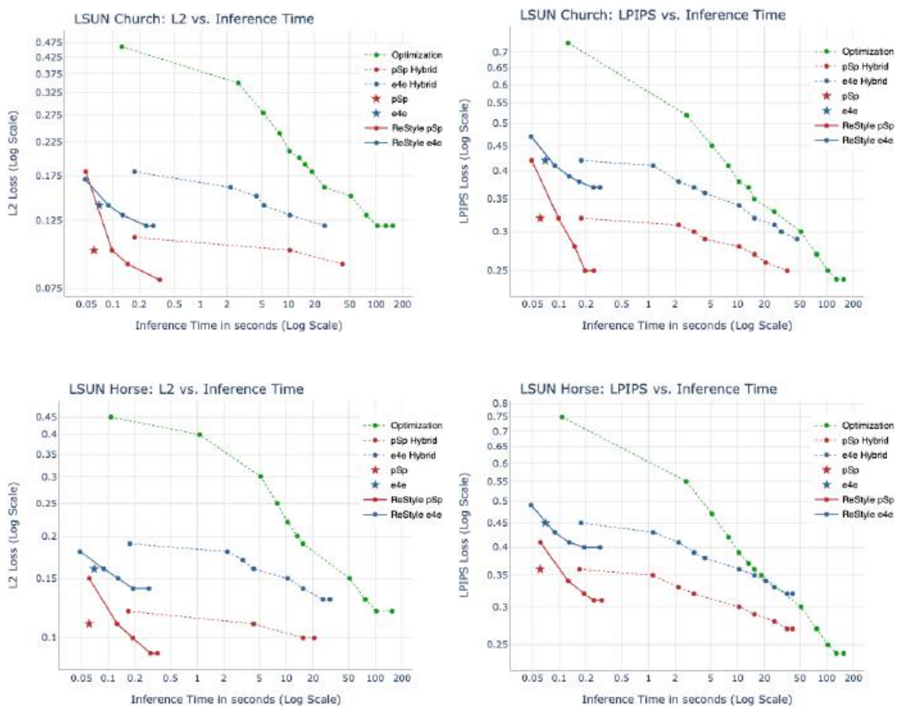


Figure 19. ***Quantitative comparison.*** Same setting as Figure 18 for the LSUN [48] Church and LSUN Horse domains.

Figure 20. Additional qualitative comparisons on the CelebA-HQ [31, 24] test set. Here, we apply ReStyle over the pSp encoder [36].

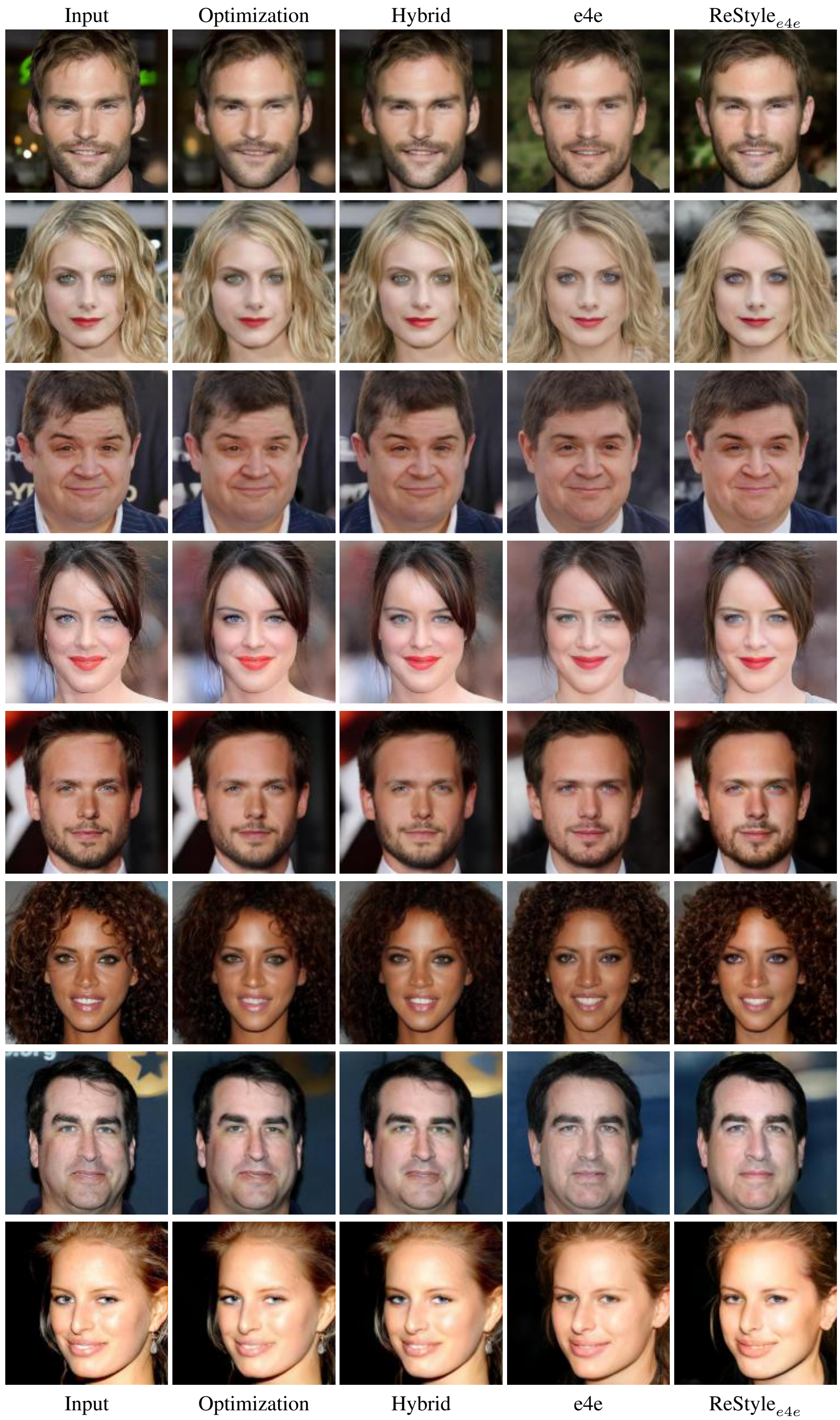


Figure 21. Additional qualitative comparisons on the CelebA-HQ [31, 24] test set. Here, we apply ReStyle over the e4e encoder [41].

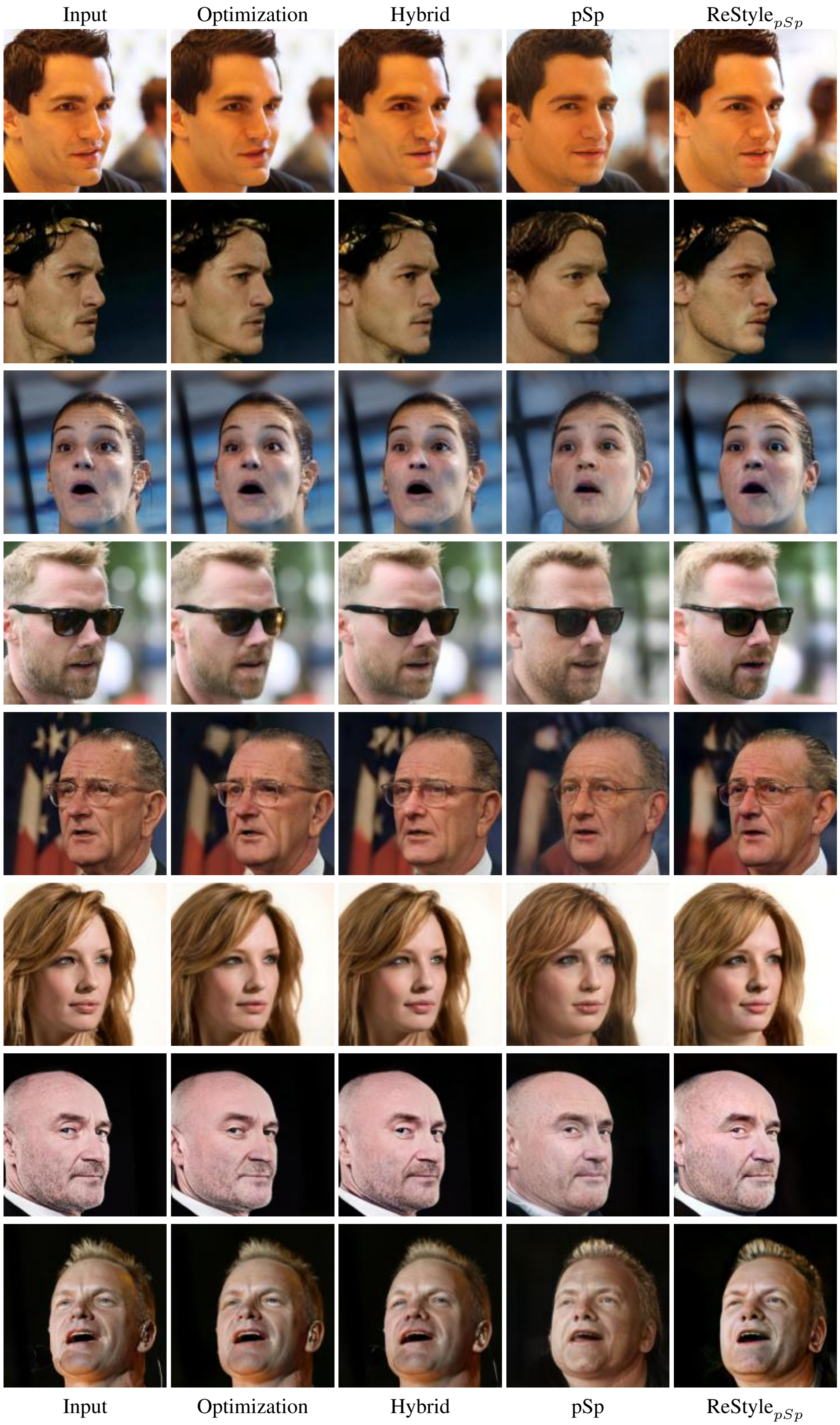
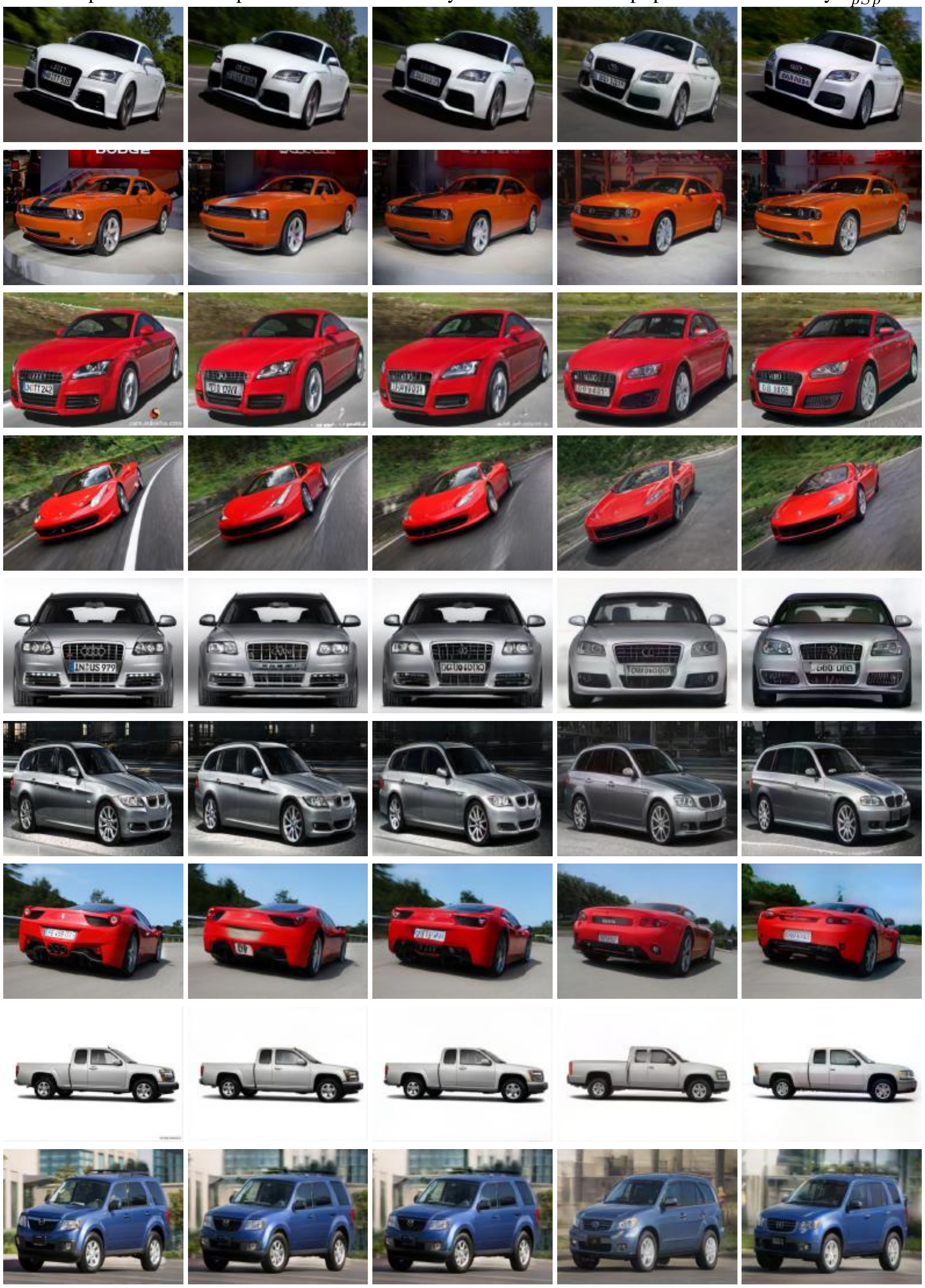


Figure 22. Additional qualitative comparisons on the CelebA-HQ [31, 24] test set on more challenging input poses and expressions. Here, we apply ReStyle over the pSp encoder [36].

Input | Optimization | Hybrid | pSp | ReStyle$_{pSp}$



Input | Optimization | Hybrid | pSp | ReStyle$_{pSp}$

Figure 23. Additional qualitative comparisons on the Stanford Cars [28] test set. Here, we apply ReStyle over the pSp encoder [36].

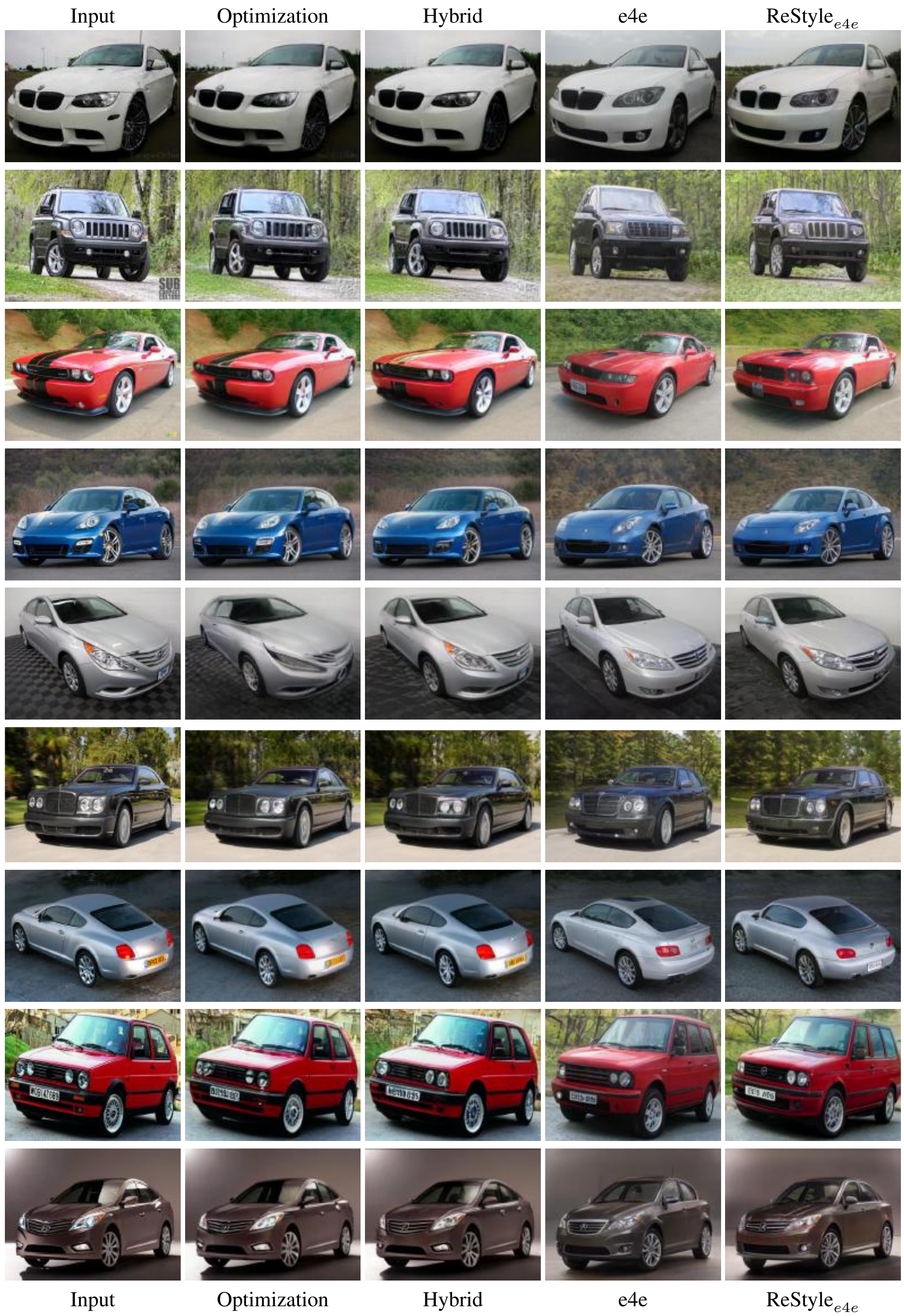


Figure 24. Additional qualitative comparisons on the Stanford Cars [28] test set. Here, we apply ReStyle over the e4e encoder [41].

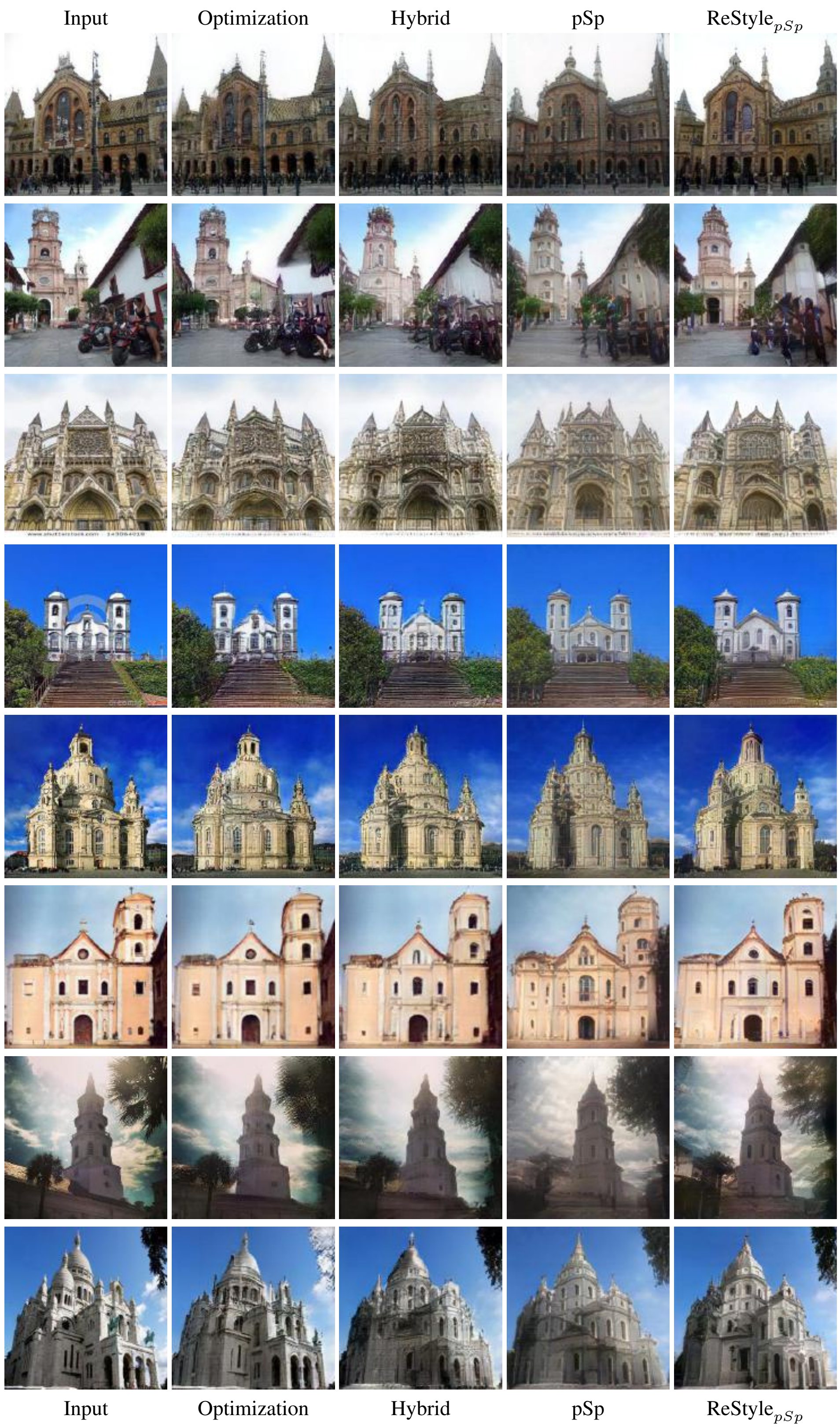


Figure 25. Additional qualitative comparisons on the LSUN [48] Church test set. Here, we apply ReStyle over the pSp encoder [36].

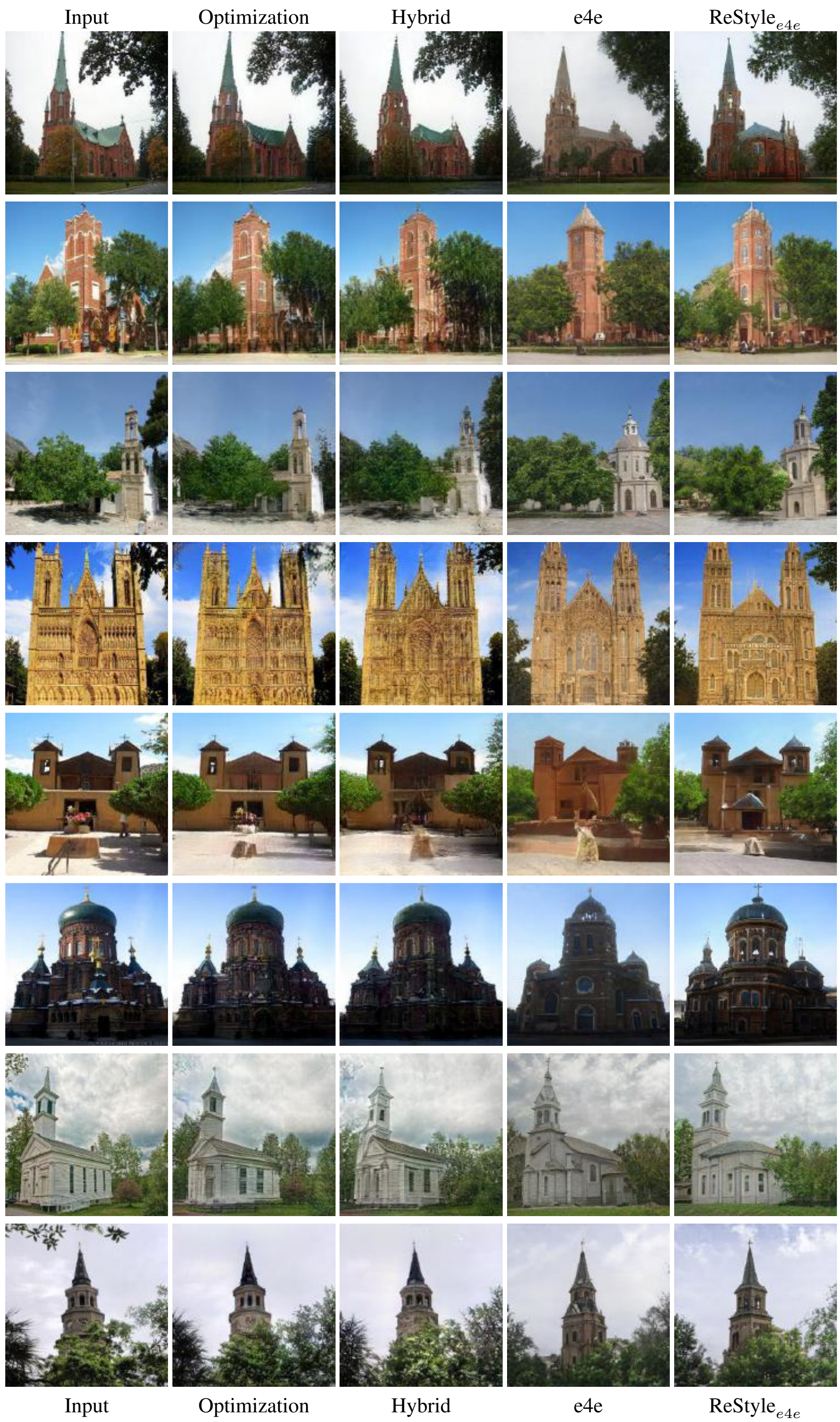


Figure 26. Additional qualitative comparisons on the LSUN [48] Church test set. Here, we apply ReStyle over the e4e encoder [41].

Figure 27. Additional qualitative comparisons on the AFHQ [10] Wild test set. Here, we apply ReStyle over the pSp encoder [36].

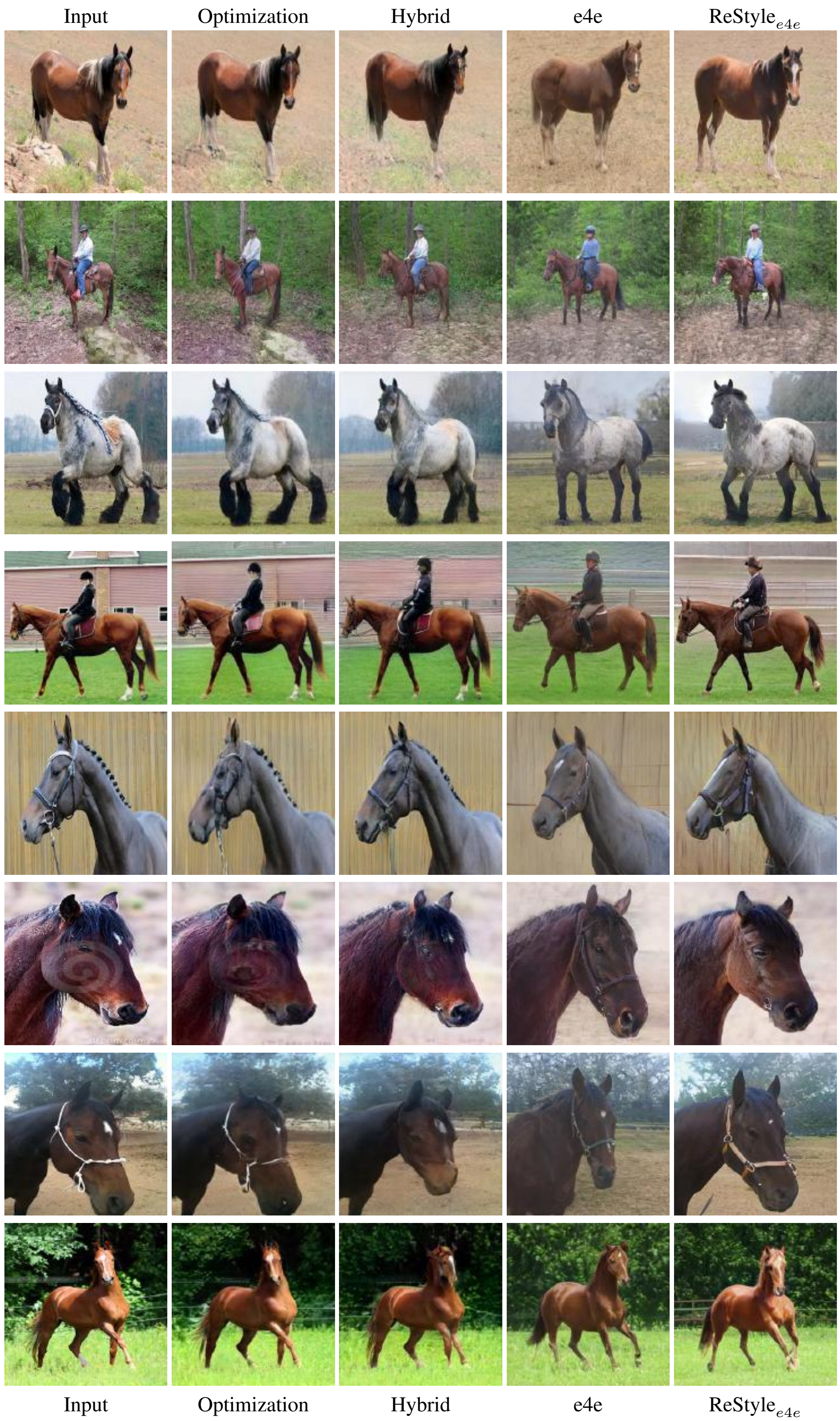


Figure 28. Additional qualitative comparisons on the LSUN [48] Horse test set. Here, we apply ReStyle over the e4e encoder [41].

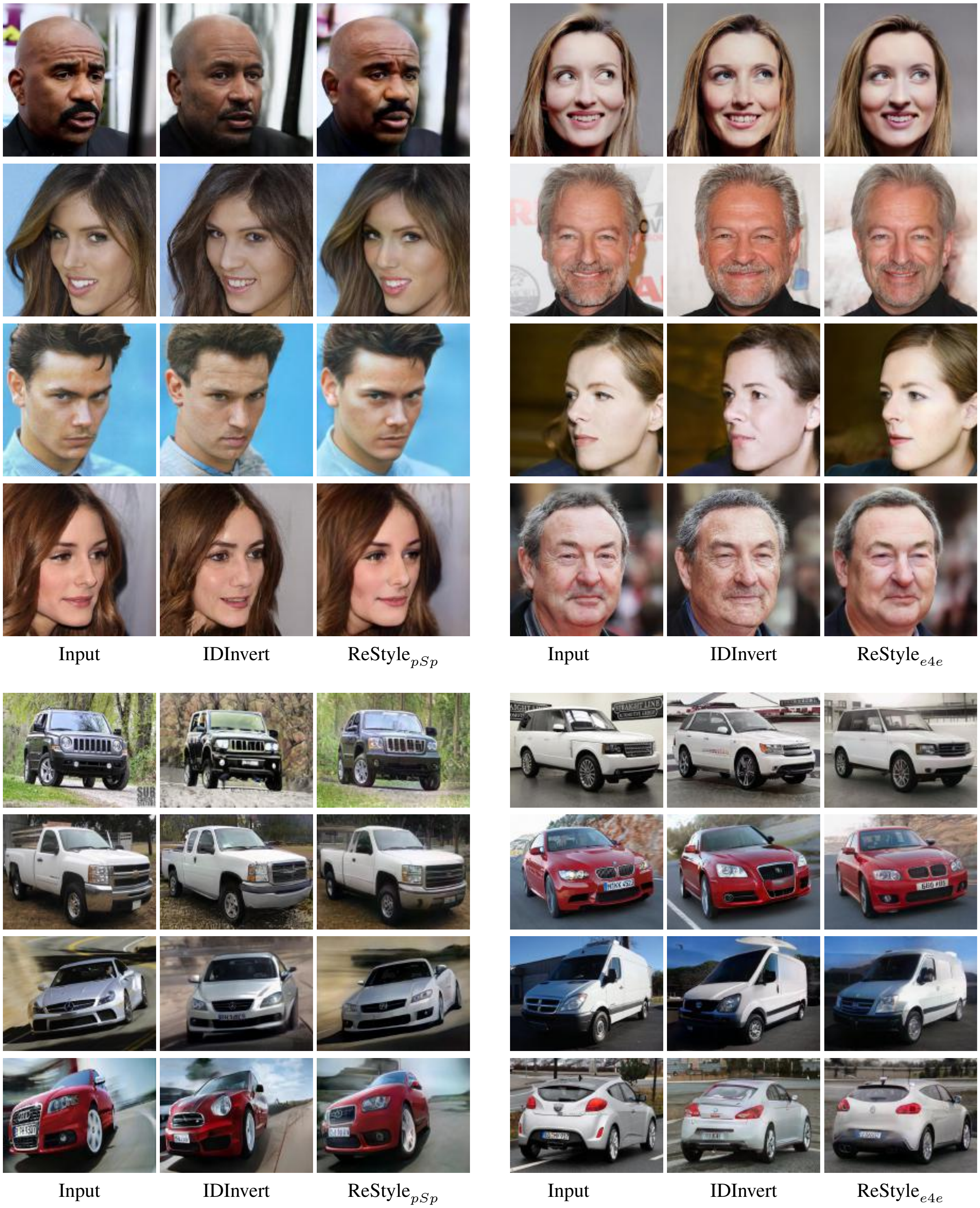
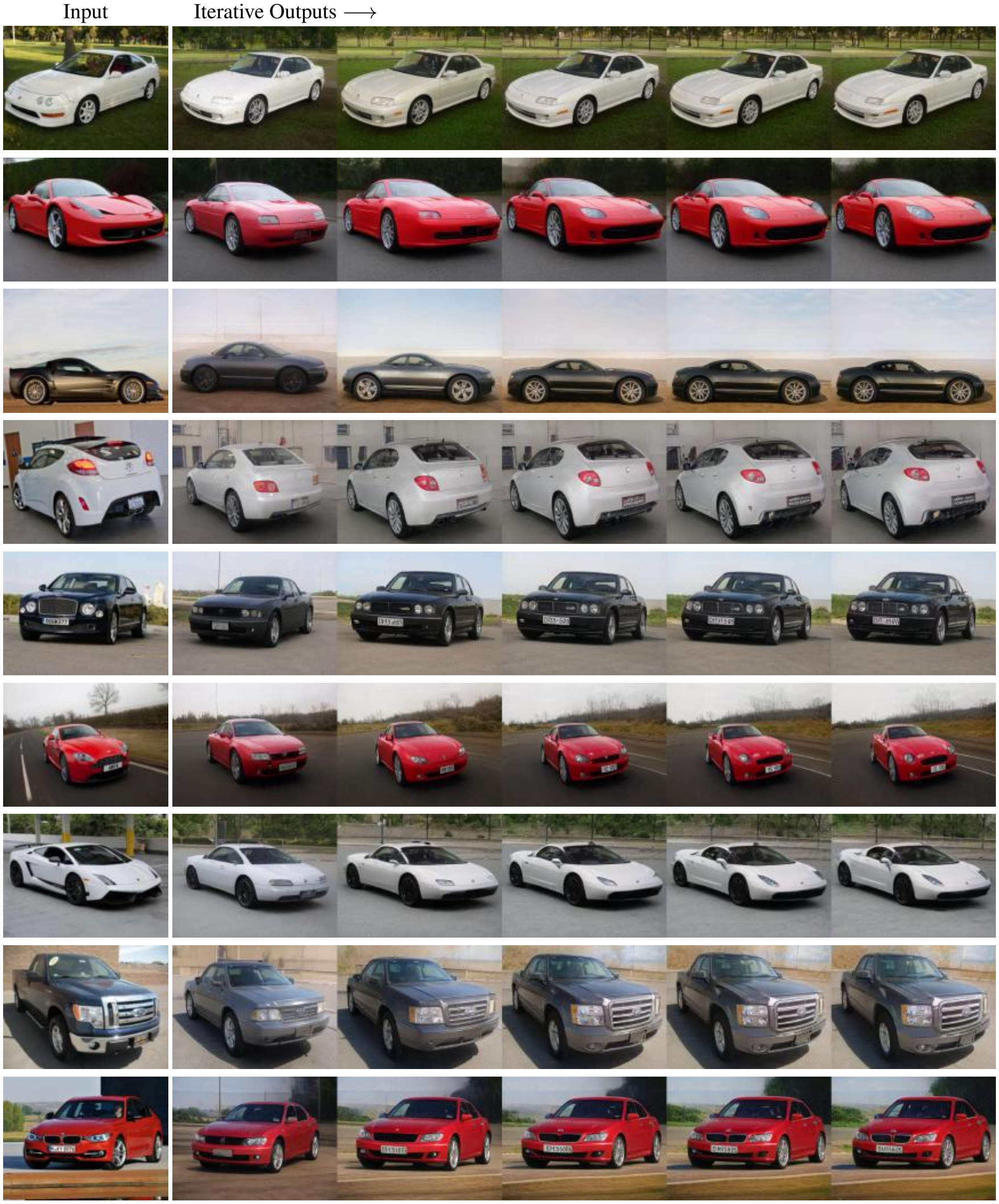


Figure 29. Qualitative comparisons with the IDInvert encoder from Zhu *et al*. [51] on the CelebA-HQ [31, 24] and Stanford Cars [28] test sets. Here, we show results with ReStyle applied over both the pSp and e4e encoders.

Figure 30. Iterative outputs generated by ReStyle applied over pSp on the human facial domain.

Input Iterative Outputs ⟶



Input Iterative Outputs ⟶

Figure 31. Iterative outputs generated by ReStyle applied over e4e on the cars domain.

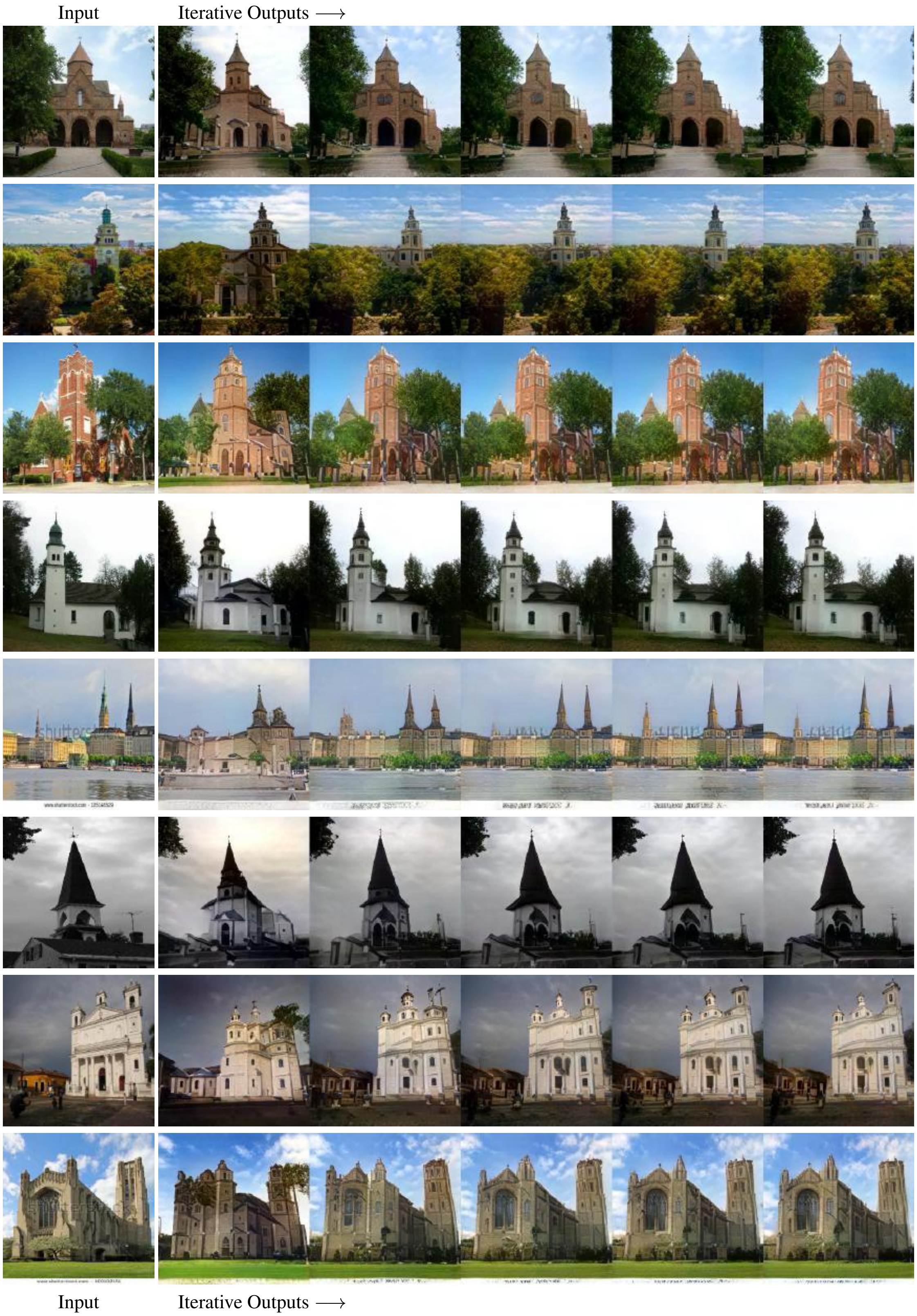
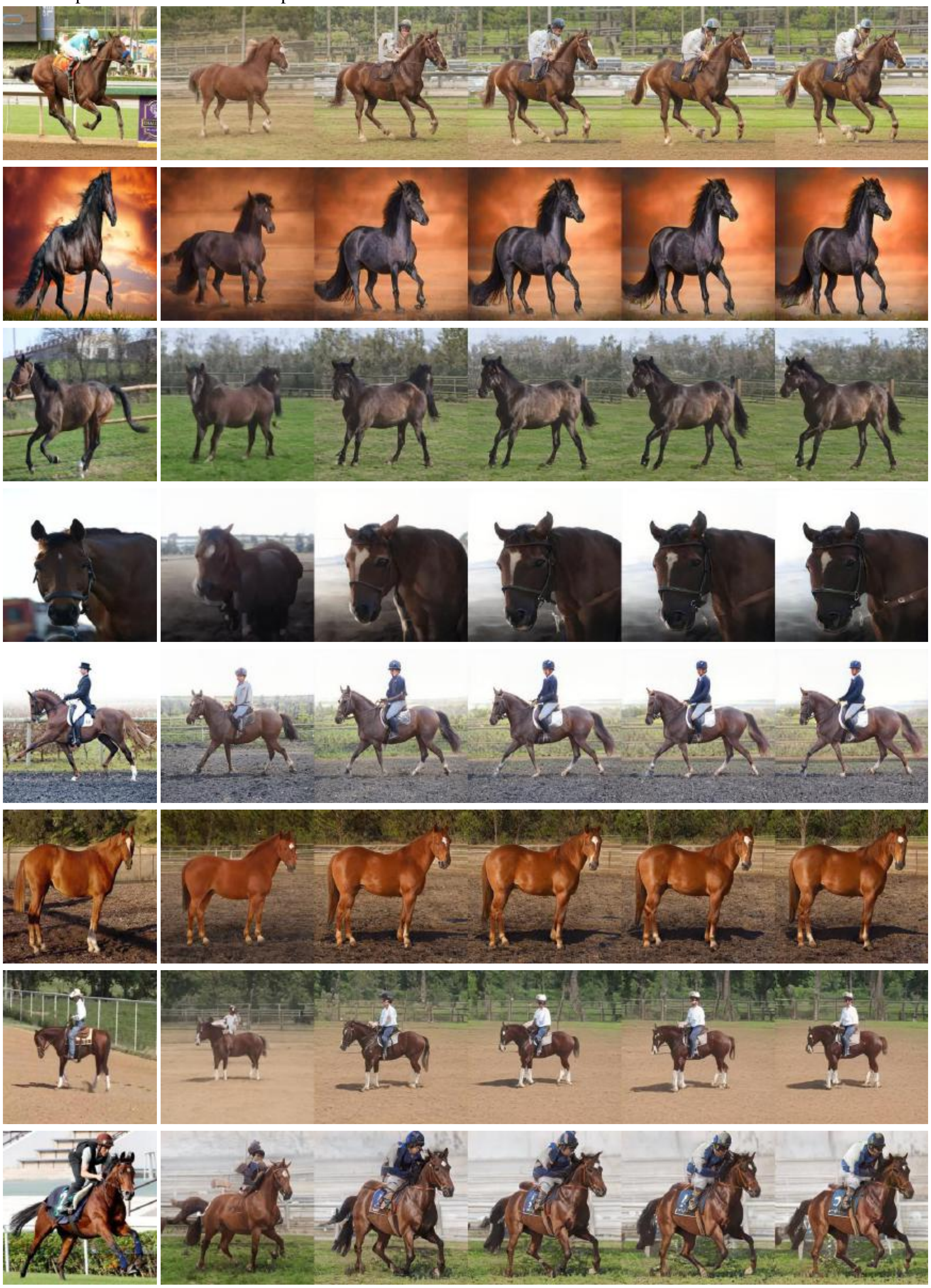


Figure 32. Iterative outputs generated by ReStyle applied over pSp on the churches domain.

Input Iterative Outputs ⟶



Input Iterative Outputs ⟶

Figure 33. Iterative outputs generated by ReStyle applied over e4e on the horses domain.

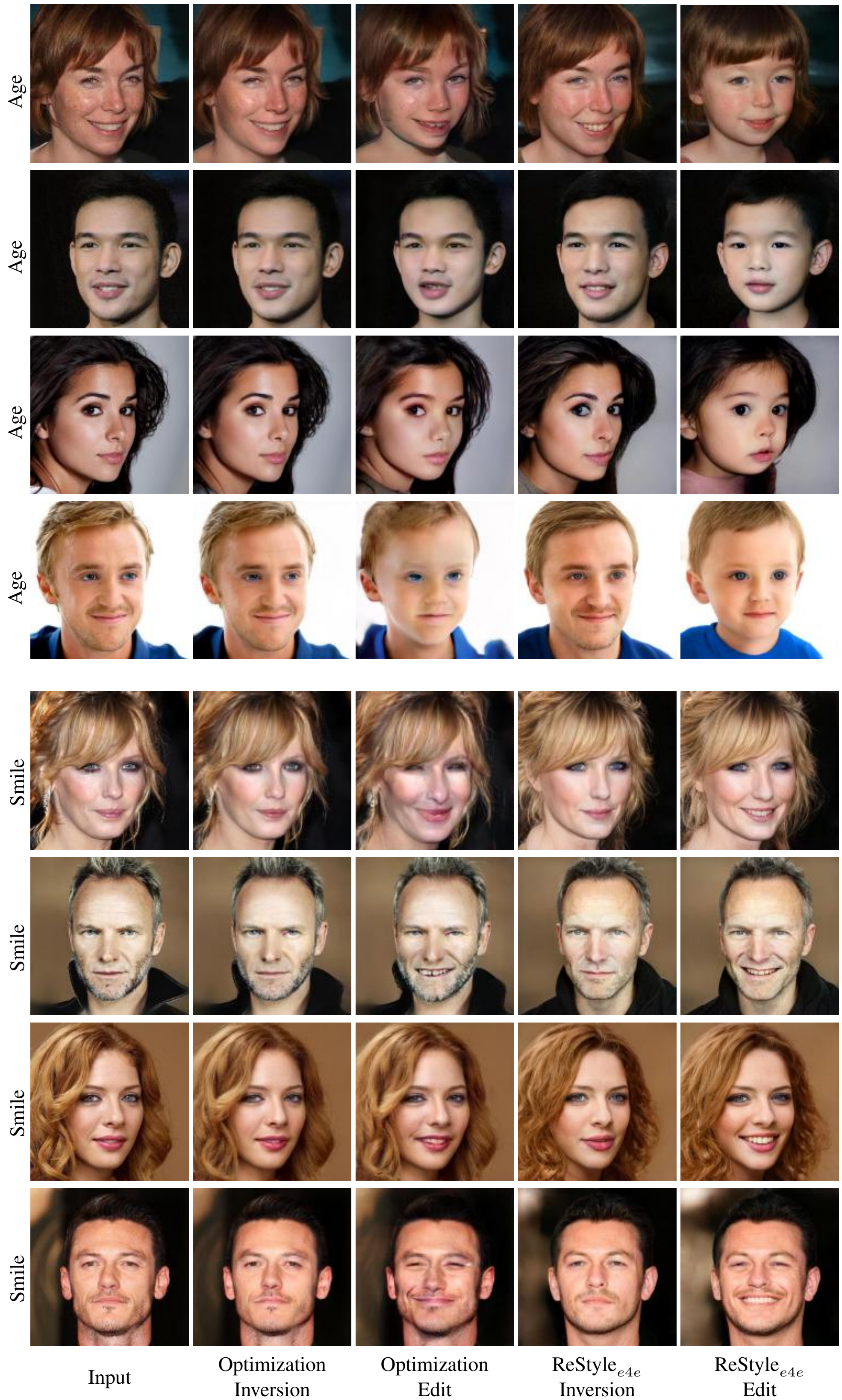


Figure 34. Editing comparisons with the optimization technique from Karras *et al*. [27] on the human facial domain obtained using InterFaceGAN [37]. Here ReStyle is applied over the e4e encoder. We perform two edits: an age edit and smile edit.

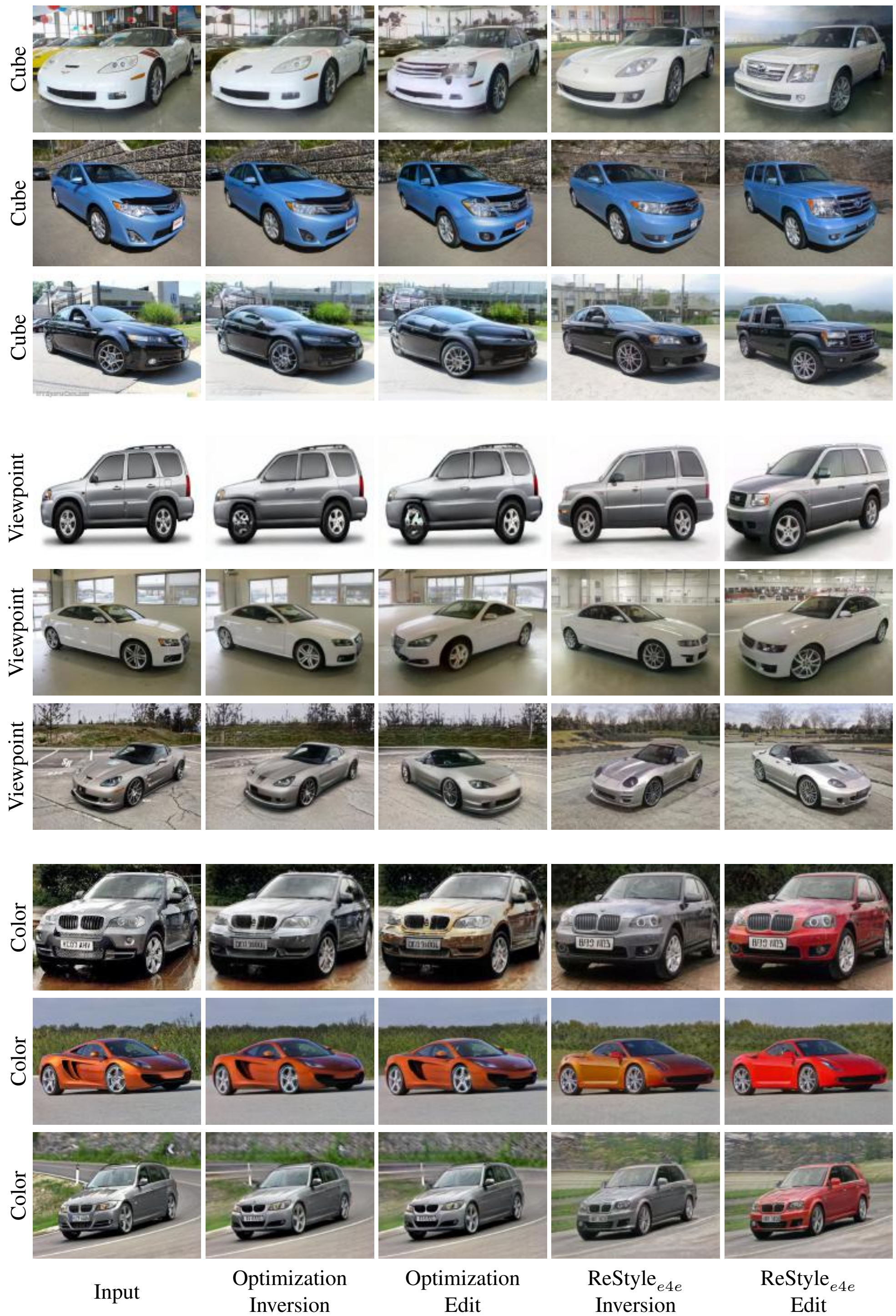
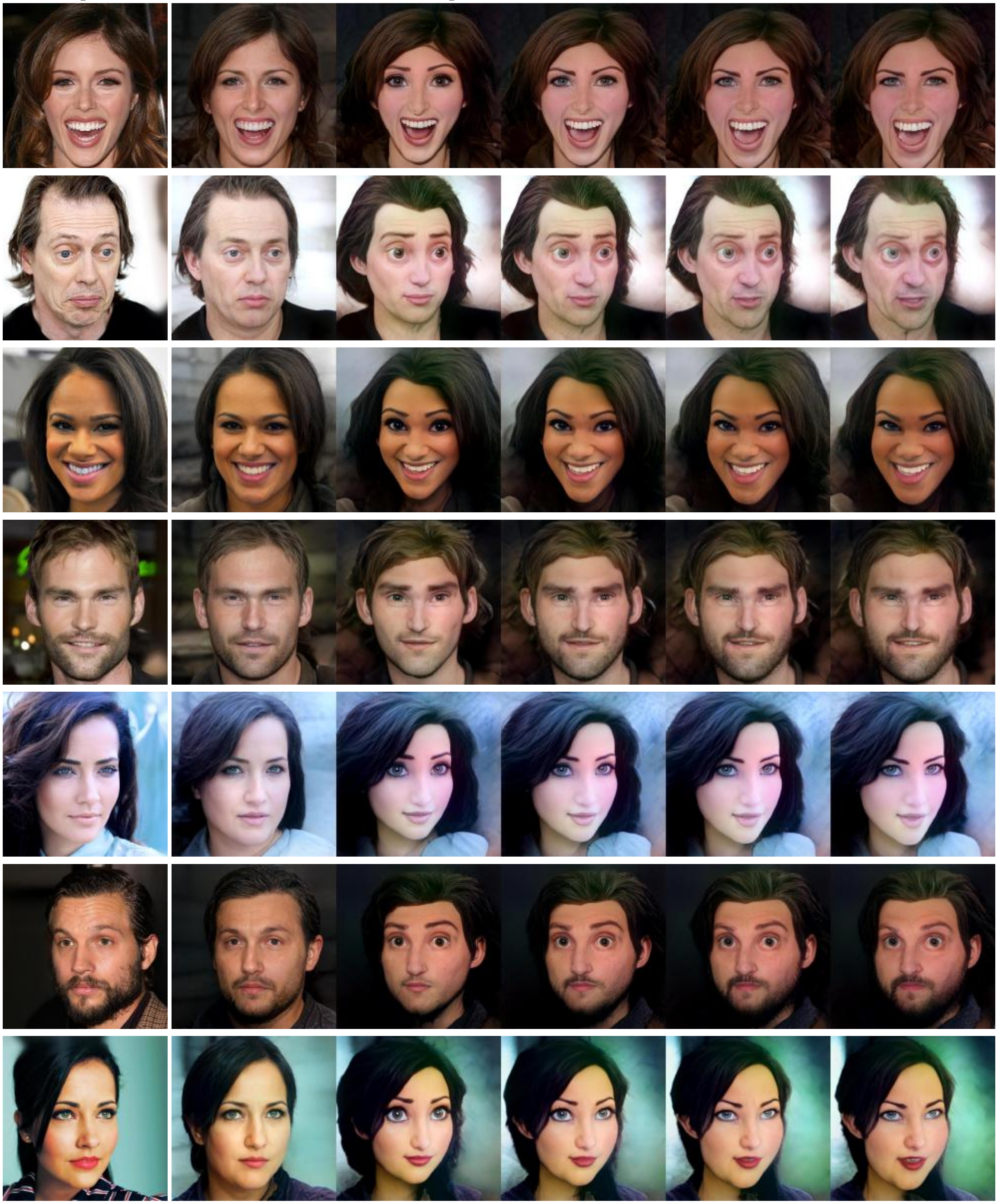


Figure 35. Editing comparisons with the optimization technique from Karras *et al*. [27] on the cars domain obtained using GANSpace [18]. Here ReStyle is applied over the e4e encoder. We perform three edits: a cube-shape edit, viewpoint edit, and color edit.

Figure 36. Editing comparisons with the optimization technique from Karras *et al*. [27] on the horses domain obtained using SeFa [37]. Here ReStyle is applied over the e4e encoder. We perform two edits: a pose edit and an edit to add/remove a horse rider.

Figure 37. Additional results on the image toonification task using ReStyle with our encoder bootstrapping technique. For each input, we show the inverted image obtained after a single step of our ReStyle$_{pSp}$ FFHQ encoder followed by the iterative outputs of our ReStyle$_{pSp}$ toonify encoder.

Input Inverted Iterative Outputs ⟶



Input Inverted Iterative Outputs ⟶

Figure 38. Additional image toonification results obtained using the same setting as Figure 37.