Fast 2-Step Regularization on Style Optimization for Real Face Morphing

SUBMISSION DATE / POSTED DATE

02-02-2022 / 19-07-2022

CITATION

Yu, Cheng; Wang, Wenmin; Li, Honglei; Bugiolacchi, Roberto (2022): Fast 2-Step Regularization on Style Optimization for Real Face Morphing. TechRxiv. Preprint. https://doi.org/10.36227/techrxiv.19105493.v2

DOI

10.36227/techrxiv.19105493.v2

Fast 2-Step Regularization on Style Optimization for Real Face Morphing

Cheng Yu[a], Wenmin Wang[a,b,*], Honglei Li[a], Roberto Bugiolacchi[b,c]

[a]School of Computer Science and Engineering, Macau University of Science and Technology, Macau 999078, China
[b]State Key Laboratory of Lunar and Planetary Sciences, Macau University of Science and Technology, Macau 999078, China
[c]Earth Sciences, University College London, London WC1E 6BT, United Kingdom

Abstract

StyleGAN is now capable of achieving excellent results, especially high-quality face synthesis. Recently, some studies have tried to invert real face images into style latent space via StyleGAN. However, morphing real faces via latent representation is still in its infancy. Training costs are high and/or require huge samples with labels. By adding regularization to style optimization, we propose a novel method to morph real faces based on StyleGAN. To do the supervised task, we label latent vectors via synthesized faces and release the label set; then we utilise logistic regression to fast discover interpretable directions in latent space. Appropriate regularization helps us to optimize both latent vectors (faces and directions). Moreover, we use learned directions under different layer representations to handle real face morphing. Compared to the existing methods, our method faster yields a larger diverse and realistic output. Code and cases are available at `https://github.com/disanda/RFM`.

Keywords: Real Face Morphing (RFM), StyleGAN, GAN Inversion, Interpretable Direction

1. Introduction

Generative adversarial networks (GANs) [1] have made a big breakthrough in high-quality (HQ) image synthesis, especially on human faces [2, 3]. Compared to auto-encoders, GAN trains a generator with a partner, the discriminator, which makes GAN reach a better performance. When a convolutional neural network (CNN) is firstly utilised in deep GANs (DCGAN) [4], adopting deeper neural networks with stable training skills become the main trend in state-of-the-art (SOTA) GANs, such as PGGAN [5], BigGAN [6], StyleGAN [3].

Although SOTA GANs, especially StyleGAN, could yield high quality (HQ) synthesized images, training deep GANs requires a substantial investment in both hardware and time: consequently, utilising pre-trained deep GANs for learning image representation is valuable. How do we use deep GANs to present real faces? To answer this question, we focus on real face morphing (RFM) via StyleGAN.

Based on StyleGAN, there are 2 phases to RFM. Phase 1: inverting real faces to latent space [7, 8, 9, 10, 11]. Phase 2: learning interpretable directions in latent space.

In the supervised manner, current methods utilise classifier to learning interpretable directions, e.g, support vector machine (SVM) [12], neural networks.[13, 14]. In the unsupervised manner, [15, 16] utilises principal component analysis (PCA) to find potential attribute vectors that could perform as semantic directions.

However, the above methods have several limitations. The supervised method [12] needs large-scale attribute labels, while the unsupervised method [15, 16] requires observing most principal components to find usable directions. Both methods incur huge costs (both in time and hardware) to learn usable directions. These shortages motivate us to design a fast method based on limited labels; we combine the two phases together and focus on fast RFM.

We revise linear models with different regularizations [17, 18] and use logistic regression (LR) with $\ell_1 - norm$ to save both time and labels. Finally, we offer a new method based on 2-step regularization that can optimize style latent vector fast for RFM. To summarize, there are our contributions:

- For morphing face on the supervised manner, we label 30,000 latent vectors with 40 attributes via Nvidia face attribute classifiers [19], and release the label set to help supervise-based works. Based on the label set, we summarize 32 usable attributes to learn interpretable directions in latent space.
- We offer a novel method to morph real faces. Compare with the existing methods, our approach is a fast and simple way. Cases are shown in Figs. 1, 11 and 14.
- Different from previous works those focus only on learning interpretable directions. We first joint GAN inversion to better operate learned directions for RFM. Here, we summarize 3 novel tricks: normalized zones for face inverted vectors, layer-wise guidelines, and distribution trim on learned directions. Based on these, we can better morph real faces via StyleGAN.

*Corresponding author.
Email addresses: `disanda@outlook.com` (Cheng Yu), `wmwang@must.edu.mo` (Wenmin Wang), `Honglei_li@qq.com` (Honglei Li), `rbugiolacchi@must.edu.mo` (Roberto Bugiolacchi)

Figure 1: Based on our method, the morphing cases of real human faces. More cases could be found in Figs. 11 and 14.

2. Related Work

2.1. GAN Inversion

GAN inversion [20] aims to invert images back into latent space so that latent vectors could reconstruct images faithfully. StyleGANs have achieved high performance on face synthesis and also better representation via style latent space. So we use StyleGAN inversion to reconstruct real faces.

Image2StyleGAN [7] attempts to invert real images into style latent space via directly optimizing specific style latent vectors (i.e., $\mathbf{w}$), but directly optimizing $\mathbf{w}$ is an inefficient approach marred with more cost when optimizing images.

To reduce the cost, deep GAN-Encoders [10, 11, 9] try to encode images into style latent space; that only needs a few times for optimizing $\mathbf{w}$. Currently, the above works could invert real faces well. However, how to represent interpretable directions via $\mathbf{w}$ remains to be explored.

2.2. Latent Representation

Latent representation in GANs could perform layer-wise manner. Crucially, HierarchyGAN [21] found that GANs can represent different semantic laws in different layers. E.g., a latent vector input GANs and then output an image. Each layer in the model orderly represents the image's 'layout', 'whole objects', 'object attributes', and 'colour scheme'.

Similarly, GAN-Dissection [22] demonstrates through PGGAN [5] that different units in the same layer represent different object types and corresponding regions. However, these studies only worked on LSUN dataset [23]. We utilise the rule to operate interpretable directions.

2.3. Interpretable Direction

We aim to morph real images via moving interpretable directions on latent space. To discover usable directions, there are two manners to deal with this.

One is the unsupervised manner, no labels are required. GANSpace[15] and EigenGAN [16] train on huge samples to find principal vectors on latent space. Those vectors may represent interpretable directions. LatentCLR [24] utilise a contrastive learning approach [25] to find usable directions. However, most vectors are invalid or redundant, so the methods should check each vector carefully. GANSpace optimizes $\mathbf{w}$

and can perform on pre-trained models. EigenGAN trains the GAN with a set of orthogonal input vectors.

The other one is the supervised manner [12, 13, 14, 26]. InterfaceGAN [12] uses SVM to classify latent vectors via labels. It finds a boundary on each label and regards the orthogonal boundary as the interpretable direction. However, most boundaries are coupled on the semantic level, so InterfaceGAN uses another boundary to alleviate the coupling. But cooperating two boundaries on latent space is unstable. GuidedStyle [13], StyleFlow [14] and StyleCLIP [26] train neural networks for classifying latent vectors. Especially, StyleCLIP uses a text-vision model (CLIP) [27] to associate latent vectors with text prompts and make awesome performance via the CLIP-based indicators. However, training CLIP needs around 400 million text-image pairs, and a deep vision transformer [28] as text encoders. Basically, these methods need huge samples, labels, and training costs to achieve the desired performance. Due to the limitation, we aim to offer a fast method to learn directions via limited labels.

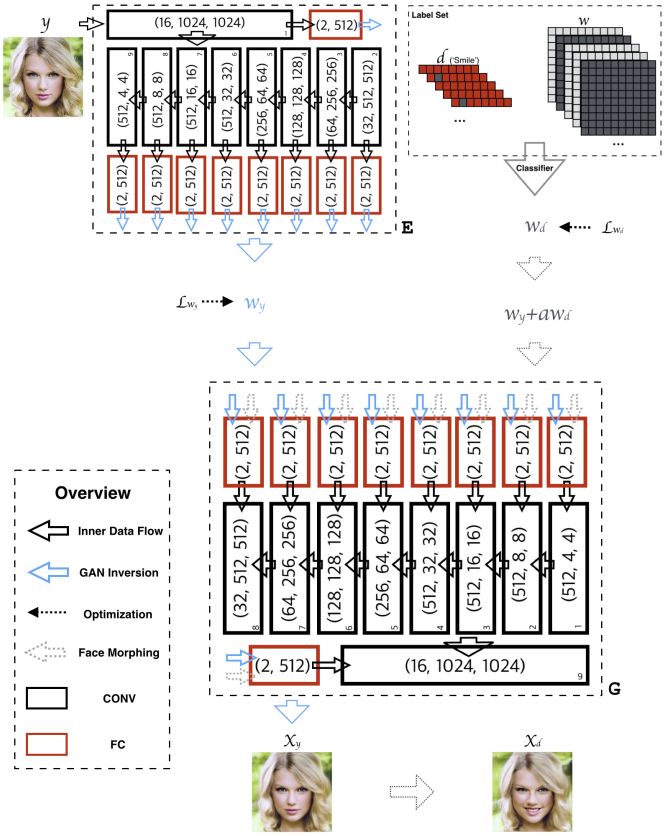


Figure 2: RFM flowchart

3. Approach

3.1. Problem Description

3.1.1. Basic Explication

Latent Space. We use pre-trained StyleGAN [3] for RFM. $\mathbf{z}$ denotes the normal latent vector whose size is 512. $\mathbf{w}$ denotes the latent style vector, and its size is (18, 512). Here, $\mathbf{w}$ is layer-size latent vector. $\mathbf{w}$ is divided into 18 layers throughout whole generator, and each layer have 512 dimensions. M denotes the mapping network and it maps $\mathbf{z}$ to $\mathbf{w}$, i.e., $w = M(\mathbf{z})$.

GAN Inversion. $\mathbf{x}$ denotes synthesized faces and $\mathbf{y}$ denotes real faces. Both sizes ($\mathbf{x}$ and $\mathbf{y}$) are pixel-wise, and the sizes are (3, 1024, 1024). G denotes the StyleGAN generator and it can synthesize 1024×1024 faces, i.e., $\mathbf{x} = G(\mathbf{w})$.

$\mathbf{w}_y$ denotes a learnable $\mathbf{w}$ which represents a real face ($\mathbf{y}$). Following [9], we use a pre-trained encoder (E) to invert $\mathbf{y}$ into $\mathbf{w}_y$, then we get a synthesized image $\mathbf{x}_y$ which is the reconstruction of $\mathbf{y}$. $\mathbf{w}_y$ comes from $E(\mathbf{y})$ and a further optimization as follows:

$$\mathcal{L}_{\mathbf{w}_y} = \arg\min_{\mathbf{w}_y} \sum_{i=1}^{n} (\mathcal{L}_1(G(\mathbf{w}_{y_i}), \mathbf{y}) + \mathcal{L}_2(\mathbf{w}_{y_i}, E(\mathbf{y}))) \quad (1)$$

Where $\mathcal{L}_1$ is used to evaluate the similarity of $\mathbf{y}$ and $\mathbf{x}_y$, and $\mathcal{L}_2$ is used to evaluate the similarity of $\mathbf{w}_y$ and $E(\mathbf{y})$.

Interpretable Direction. $\mathbf{d}$ denotes a semantic direction related to a label attribute. $\mathbf{d}$ performs as a one-hot vector: $\mathbf{d}$ indicates the positive correlation for the attribute, and $-\mathbf{d}$ indicates the negative correlation. $\mathbf{w}_d$ denotes a specific $\mathbf{w}$, which represents $\mathbf{d}$ in style latent space.

To learn fast $\mathbf{w}_d$ with limited labels, we aim to design a loss function (i.e., $\mathcal{L}_{w_d}$) to optimize a simple classifier.

Finally, $\mathbf{x}_d$ denotes the morphed real image on $\mathbf{d}$. α denotes the coefficient of $\mathbf{w}_d$, as follows:

$$\mathbf{x}_d = G(\mathbf{w}_y + \alpha \mathbf{w}_d) \quad (2)$$

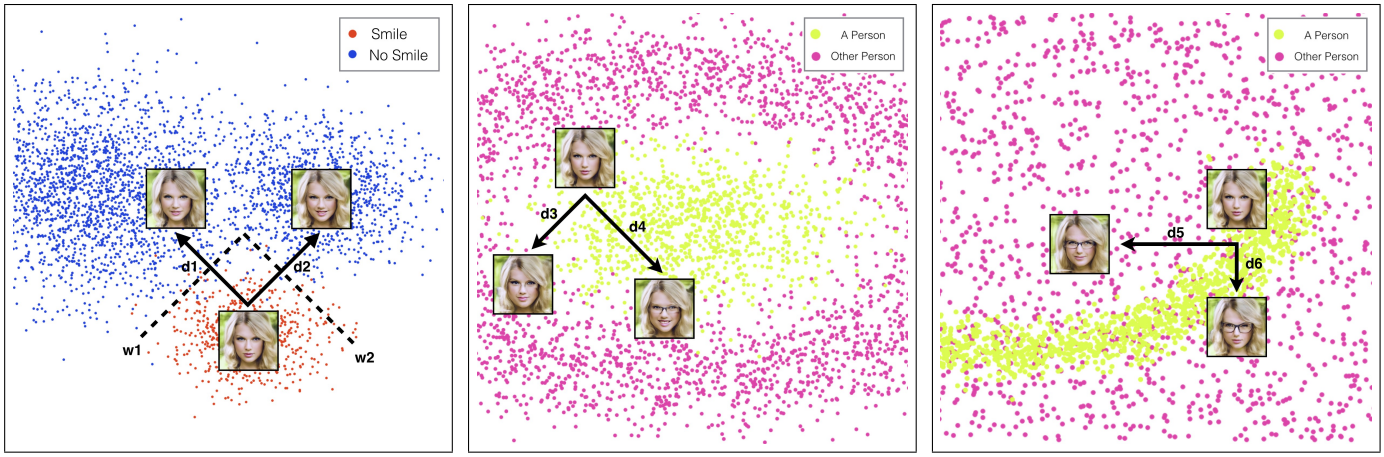
We display the above processes in Fig. 2. Where CONV denotes the convolutional layer with feature size: (channels, heights, widths). FC denotes the fully connected layer with input size: (layers, nodes). Here, FC operates $\mathbf{w}$, and each block contains 2 FC layers.

3.1.2. Typical Occasions

Depending on different samples, as well as different methods, RFM results are likely to be different. Testing through a large number of cases, it emerged that the different $\mathbf{w}_d$ are orthogonal to each other. Therefore, we consider that there exist multiple $\mathbf{d}$ for the same attribute, but some $\mathbf{d}$ may be unusable for morphing real faces.

As shown in Fig. 3, $\mathbf{w}_1$ and $\mathbf{w}_2$ are the boundary walls. when looking for a simple attribute; different $\mathbf{d}$ may be all valid, for instance, the smile directions ($\mathbf{d}_1$ and $\mathbf{d}_2$, Fig. 3(a)). However, searching for usable $\mathbf{d}$ is complicated on most attributes, such as the attribute of eyeglasses, these are likely to make a difference of coupling, e.g, $\mathbf{d}_4$ is coupling with smile. $\mathbf{d}_5$ is coupling with old. $\mathbf{d}_6$ is coupling with young, and even $\mathbf{d}_3$ is invalid (see Fig. 3(b) and Fig. 3(c)).

When we move $\mathbf{w}_d$ to a short distance, most of the morphing cases are valid ($\alpha < 100$ in Eq. 2). As the distance increases, most attributes stay coupled on morphing faces, thus our goal



(a) (b) (c)

Figure 3: 3 occasions for the attribute change. Both smile directions ($\mathbf{d}_1$ and $\mathbf{d}_2$) can make real face smile (a). However, different glasses directions ($\mathbf{d}_3$ - $\mathbf{d}_6$) yield attribute coupling on different cases. $\mathbf{d}_3$ changes to another person. $\mathbf{d}_4$ is coupling with smile. $\mathbf{d}_5$ and $\mathbf{d}_6$ are coupling with age (younger or older).

shifts to finding the decoupling way to move $\mathbf{w}_d$ to a longer distance.

According to the training strategy of StyleGAN, $\mathbf{w}$ obeys a standard Gaussian distribution, which means that their value of mean μ is 0. We demotes an original point in latent space: all element values of original point are 0, i.e., ($\|w\|_{\rho=2} = 0$). The other faces and morphed results are located at further points in the latent space.

3.2. Step 1: $\mathbf{w}_y$ Regularization

We denote *Norm* as the value of $\|\mathbf{w}_y\|_{\rho=2}$. As shown in Fig. 4, original point is red point (*Norm* = 0) and represents a woman face. We divide the value range of $\mathbf{w}_d$ into four regions ($Zone_1$ - $Zone_4$) according to different *Norm*. While *Norm* ≈ 0, the face identity is essentially similar to the original point.

We use pose direction as the test. If $\mathbf{w}_y$ falls into $Zone_1$, $\mathbf{w}_y$ cannot well represent the real face $\mathbf{y}$ (*Norm* = 98) and adding $\mathbf{w}_d$ will make it sensitive to real face morphing.

With the increasing of *Norm*, $\mathbf{w}_y$ dropping to $Zone_4$ from $Zone_1$; the face identity can be represented well, i.e, $x_y \simeq y$, but the sensitivity of moving directions will be decreased. When *Norm* is too large (dropping into $Zone_4$), although $\mathbf{w}_y$ can fully represent the identity of $\mathbf{y}$, $\mathbf{w}_d$ becomes hard to change faces. This phenomenon is similar to model collapse.

To find a compromise representations of $\mathbf{w}_d$ and $\mathbf{w}_y$, we normalize $\mathbf{w}_d$ into $Zone_1$, and then we normalize $\mathbf{w}_y$ into $Zone_3$. So that $\mathbf{w}_y$ fully represent $\mathbf{y}$ and $\mathbf{w}_d$ could morphing faces smoothly. Consequently, we upgrade Eq. 1 by adding the first regularization:

$$\mathcal{L}_{\mathbf{w}_y} = \arg\min_{\mathbf{w}_y} \sum_{i=1}^{n} (\mathcal{L}_1(G(\mathbf{w}_{y_i}), \mathbf{y}) + \mathcal{L}_2(\mathbf{w}_{y_i}, E(\mathbf{y}))) + \beta \|\mathbf{w}_y\|_{\rho=2} \quad (3)$$

For ensuring $\mathbf{w}_y$ drops into $Zone_3$, we test different parameter sets: coefficient β with ρ, and we report more details in Section 4.2.1.

3.3. Step 2: $\mathbf{w}_d$ Regularization

For learning $\mathbf{w}_d$, we use linear models to optimize $\mathbf{w}_d$ and a bias $\mathbf{b}$ via label classification on latent space. In InterfaceGAN [12], SVM is used to classify $\mathbf{w}_d$ with $\mathbf{b}$ as Eq. 5:

$$\mathcal{L}_{\mathbf{w}_d} = \arg\min_{\mathbf{w}_d, \mathbf{b}} \sum_{i=1}^{n} \max(0, 1 - \mathbf{d}_i({\mathbf{w}_{\mathbf{x}_i}}^T \mathbf{w}_d + \mathbf{b})) + \gamma \|\mathbf{w}_d\|_{\rho=2} \quad (4)$$

Where $\mathbf{b}$ is frozen to 0. According to the implement of SVM [29], the default coefficients are $\gamma = 1$ and $\rho = 2$. InterfaceGAN costs huge labels and time, so we aim to reduce costs; we consider a logistics regression (LR) classification as follows:

$$\mathcal{L}_{\mathbf{w}_d} = \arg\min_{\mathbf{w}_d, \mathbf{b}} \sum_{i=1}^{n} \log(\exp(-\mathbf{d}_i({\mathbf{w}_{\mathbf{x}_i}}^T \mathbf{w}_d + \mathbf{b})) + 1) + \gamma \|\mathbf{w}_d\|_{\rho=1} \quad (5)$$

We found that the higher accuracy of the classifier, the higher performance of $\mathbf{w}_d$. In addition, when we use LR with default $\gamma = 1$, the performance of $\rho = 1$ is slightly better than $\rho = 2$, due to $\rho = 1$ produces more sparse $\mathbf{w}_d$ that is more suitable for $\mathbf{w}_d$ representation. We denote $\rho = 1$ as $\ell_1 - norm$, and $\rho = 2$ as $\ell_2 - norm$ and report the ablation study in Section 4.2.2.

3.4. Layer-Wise Representation

As shown in Fig. 5, we get $\mathbf{w}$ from $\mathbf{z}$ which passes through a mapping network M. By truncating the first 8 layers towards original point (*Norm* = 0), we get w_{trunc}, i.e., $w_{trunc} = 0.7w$ for the first 8 layers. Compared to $\mathbf{w}_d$, w_{trunc} is closer to original point, and better represent the face identity. Next, we get $\mathbf{w}_d$ and the bias ($\mathbf{b}$) via linear models (Eq. 5).

HierarchyGAN [21] found that: in the bedroom dataset of LSUN [23], layers of different depths in StyleGAN have different representations. We tested 32 face label attributes and found that StyleGAN has similar patterns in the face dataset [2, 3].

Following the layer-wise guideline (Fig. 5), we divide StyleGAN layers into two categories (texture and colour) and divide

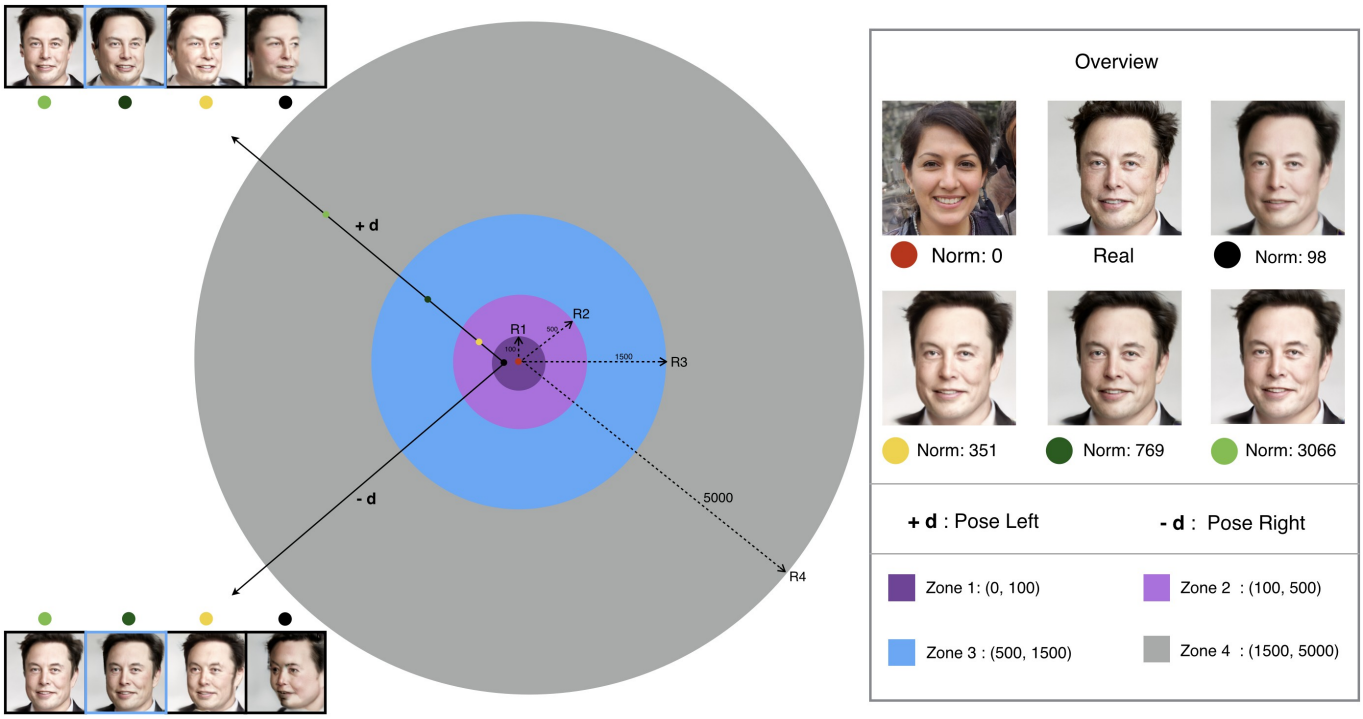


Figure 4: Overview of different $\mathbf{w}_y$ *Norm*. Moving same d (pose) will make difference results within 4 zones of $\mathbf{w}_y$.

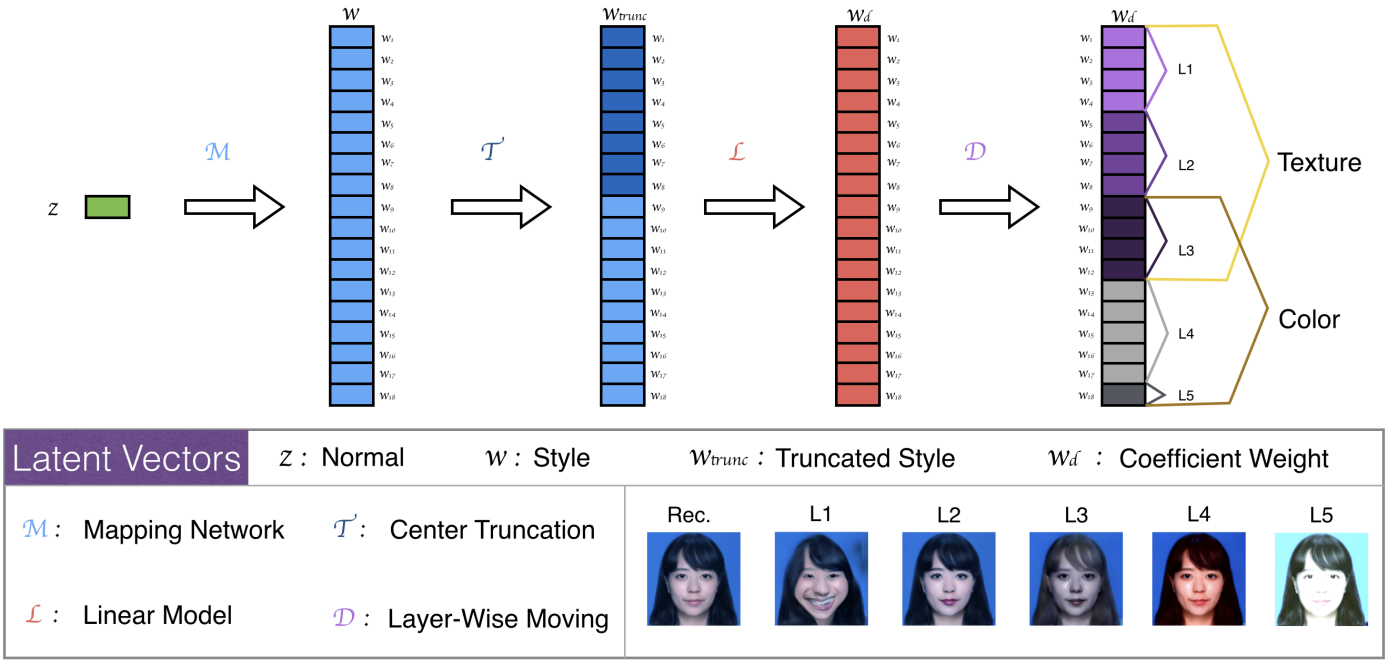


Figure 5: Layer-wise guideline for $\mathbf{w}_d$, M contains 8 fully connected layers. T and D are vector operations. L classifies $\mathbf{d}$ into $\mathbf{w}_d$.

them into five levels (L1-L5) according to the network layer from deep to shallow, where texture represents the high-level changes. Here, L1 is mainly the attribute statements, representing the whole facial changes, such as smile, glasses, expression, etc. L2 represents local changes on the face, such as the beard, hair, eyes, and lips. L3 is more related to whole face colours, such as hair colours. In shallow layers, L4-L5, represent whole image colours, with the difference that L4 mainly highlights the colours of the face, while L5 highlights the colours of the background.

We added a trim operation. Let $\mathbf{w}_d$ keep the features of $\mathbf{y}$, while highlight the $\mathbf{d}$ representation as much as possible. The specific method is to set $\delta = [0.01, 0.05]$ and $\mathbf{w}_d = 0$ when $\mathbf{w}_d < \delta$, so that the each dimension of $\mathbf{w}_d$ is larger than δ.

Algorithm 1: Fast 2-Step Regularization

 # Input:
Pre-trained Networks: G , M, E, C.
 # Initialize:
Normal Latent Vector: $\mathbf{z} \sim \mathcal{N}(0, 1)$,
Parm.: $i = 0$, $j = 0$
$\alpha \in [0, 1500]$, $\beta \in$ [1e-4, 3e-4] , $\gamma = 1$.
 # Frist Regularization - Optimizing $\mathbf{w}_y$:
while *($i < 2,000$)* **do**
  $\mathbf{w}_y \leftarrow \nabla\mathcal{L}_1(G(\mathbf{w}_{\mathbf{y}_i}), \mathbf{y}) + \nabla\mathcal{L}_2(\mathbf{w}_{\mathbf{y}_i}, E(\mathbf{y}))$,
  $\mathbf{w}_y \leftarrow \nabla\beta\|\mathbf{w}_{\mathbf{y}_i}\|_{\rho=2}$.
  $i = i + 1$.
end
Output: $\mathbf{w}_y$
# Labeling $\mathbf{d}$ on $\mathbf{w}$
while *($j < 30,000$)* **do**
  $\mathbf{w}_j \leftarrow M(\mathbf{z}_i)$, $\mathbf{x}_j \leftarrow G(\mathbf{w}_j)$, $\mathbf{d}_j = C(\mathbf{x}_j)$,
  $j = j + 1$.
end
 # Second Regularization - Optimizing $\mathbf{w}_d$:
while *(**if** $acc(\mathbf{w}_d, \mathbf{d}) < 0.85$ **and** $\nabla > 0.001$)* **do**
  $\mathbf{w}_d \leftarrow \nabla \log(\exp(-\mathbf{d}_i(\mathbf{w}_{\mathbf{x}_i}{}^T\mathbf{w}_{\mathbf{d}_i} + \mathbf{b}_i)) + 1)$,
  $\mathbf{w}_d \leftarrow \nabla\gamma\|\mathbf{w}_d\|_{\rho=1}$.
end
Output: $\mathbf{w}_d$
Moving: $\mathbf{x}_d = G(\mathbf{w}_y + \alpha\mathbf{w}_d)$.

4. Experiments

Implement Details. Our frameworks are Python (3.7.3), PyTorch (1.5.1, with CUDA 10.2) and SciKit Learning (SKLearn, 0.24.2). We used a GPU Card (GeForce RTX 3090) to invert image ($\mathbf{y}$ to $\mathbf{w}_y$), and used a CPU card (Inter Core I5 10600KF) to optimize interpretable directions ($\mathbf{d}$ to $\mathbf{w}_d$). For implemented $\mathbf{w}_d$ optimization, we used LibSVM [29] to perform SVM ($\ell_2 - norm$) and Sega to perform LR ($\ell_1 - norm$ and $\ell_2 - norm$) [17, 18]. More details refer to our released code.

Algorithm. We describe the whole method as the following algorithm 1 (Pseudocode). Pre-trained networks are StyleGAN G (FFHQ-F) [3], corresponding E [9]. we set 2,000 iterations for GAN inversion (i.e., $i < 2,000$). We used Nvidia Face Attribute Classifier [19] to label 30,000 w (i.e., $j = 30,000$), but we only uses 8,000 w to save time costs. Here, C denotes the face attribute classifier which contains 40 networks to detect 40 face attributes.

α is coefficient of $\mathbf{w}_d$ and we empirically give a guide range. β and γ are coefficients for the 2-step optimizations respectively. $\mathcal{L}_1$ and $\mathcal{L}_2$ come from Eq. 1. $acc(\mathbf{w}_d, \mathbf{d})$ denotes the accuracy of attribute classifier and we set the board value as 0.85.

Evaluation Metrics. To evaluate the similarity between morphed faces with real faces. We used peak signal-to-noise ratio (PSNR) to measure pixel similarity, and we used SSIM [30] and LPIPS [31] to measure feature similarity. To compare different methods, we also used FID [32] to measure statistic similarity.

Table 1:
Label set (valid attributes and layer levels).

ID	Name	Sim.	Semantic Level				
			L1	L2	L3	L4	L5
1	Smile		✓				
2	Happiness	•	✓				
3	Pose		✓				
4	Gender		✓				
5	Age		✓				
6	Young	•	✓				
7	FaceShape (Oval)		✓				
8	DoubleChin	•	✓				
9	Cheekbones	•	✓				
10	Glasses		✓	✓			
11	SunGlasses	•	✓	✓			
12	ReadingGlasses	•	✓	✓			
13	HairShape (Wavy)		✓	✓	✓		
14	Bald			✓			
15	Hairline	•		✓			
16	Bangs	•		✓			
17	LipsSize			✓			
18	NoseSize			✓			
19	Makeup (Face)			✓	✓		
20	Makeup (Eye)	•		✓	✓		
21	Makeup (Lip)	•		✓	✓		
22	Sadness			✓	✓		
23	Eyebrow (Loc.)			✓	✓		
24	Eyebrow (Dense)			✓	✓		
25	Moustache			✓	✓		
26	Beard	•		✓	✓		
27	Sideburns	•		✓	✓		
28	Goatee	•		✓	✓		
29	BlackHair				✓	✓	
30	BlondHair				✓	✓	
31	RedHair				✓	✓	
32	Exposure						✓

• denotes that the attribute is similar to the above one.
✓ indicates valid layer levels

4.1. Label Set

An existing label set comes from Microsoft face attribute classifiers [33], which contains 20,307 samples with 40 attributes i.e., w and d. But in the label set, some attributes have no samples or the attributes are not valid for image morphing. Therefore, we labelled 30,000 samples with 40 similar attributes via Nvidia face attribute classifiers [19]. Finally, we integrated two label sets and sorted out 32 valid attributes (see Table 1).

These attributes are divided into different layers according to the different semantic levels (5 layer levels). From L1 to L5, the facial features represent texture to colour. Some attributes control the same semantic (Sim.), only a few details are different representations, such as Smile and Happiness, Age and Young, etc. In addition, there is a certain correlation between adjacent layer levels, and it is necessary to increase or decrease 1-2 layers to discover a better representation.

Figure 6: Moving $\mathbf{w}_d$ from the original point $\mathbf{w}_y$ ($Norm$ = 0). Right top values are coefficient α in Eq. 2.

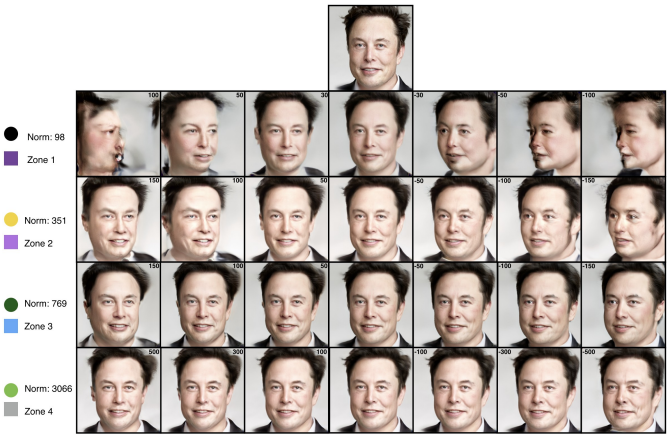


Figure 7: Moving the pose direction ($\mathbf{w}_d$) from different $\mathbf{w}_y$ for the same person. $\mathbf{w}_y$ filled into different $Norm$ zones. Right top values are α in Eq. 2.

4.2. Ablation Study

4.2.1. Norm Zone of $\mathbf{w}_y$

$\mathbf{w}_y$ near the origin point are most sensitive (see Fig. 6) to face changes. That means when $\mathbf{w}_y$ located in $Norm \in (100)$, $Zone_1$, the attribute changes are more coupled than $\mathbf{w}_y$ in the other zones (large $Norm$).

Table 2:
Ablation study among different ρ and β for $\mathbf{w}_y$ optimization (Eq. 3).

$\mathbf{w}_y$		$Norm$	Zone	SSIM ↑	LPIPS ↓
ρ = 1	β = 10e-4	(**85**, 62, 104)	1, 2	0.384	0.381
	β = 5e-4	(**117**, 98, 139)	1, 2	0.441	0.332
	β = 1e-4	(**212**, 176, 243)	2	0.579	0.225
ρ = 2	β = 10e-4	(**367**, 313, 429)	2	0.632	0.185
	β = 5e-4	(**506**, 416, 573)	2,3	0.643	0.181
	β = 3e-4	(**601**, 497, 688)	2,3	0.680	0.158
	β = 1e-4	(**849**, 640, 1032)	3	**0.687**	**0.156**

* $Norm$ values are (Avg., Min., Max.)

As shown in Fig. 7, by moving the same $\mathbf{w}_d$ (pose) with different $\mathbf{w}_y$, where the black point located in $Zone_1$ is the most sensitive, and there is an easy coupling for image morphing, or even invalid. We tested 32 real face images and observed that when $\mathbf{w}_y$ falls into $Zone_2$ and $Zone_3$, the attribute changes tend to be smooth, and we chose $Zone_3$ for the better performance that makes face change more smooth. If $\mathbf{w}_y$ falls into $Zone_4$, moving $\mathbf{w}_d$ will make little or even no change to face morphing.

To evaluate the regularized effect of different parameter values (β and ρ) on $\mathbf{w}_y$, we inverted 32 real faces via 7 test groups and report the results on Table 2. Among the same range of β, ρ = 1 only regularized $\mathbf{w}_y$ to $Zone_1$ and $Zone_2$, but ρ = 2 regularized $\mathbf{w}_y$ to $Zone_3$ and $Zone_4$ while β is in the range of [1e-4, 3e-4]. So we added $\ell_2 - norm$ (i.e., ρ = 2) with β = [1e-4, 3e-4] to regularize $\mathbf{w}_y$ into $Zone_3$ (Eq. 3).

4.2.2. Linear Classifier for $\mathbf{w}_d$

Based on our label set, we used linear models (LMs) to learn interpretable directions. To evaluate LMs, we compared three linear classifiers within 8k label samples. The 3 classifiers are SVM Classifier (SVC) [12], LM ($\ell_2 - norm$) [33], and our method (LM, $\ell_1 - norm$). Depending on the increasing number of training pairs ($\mathbf{w}_d$ and $\mathbf{d}$), we measured 33 attributes upon the time cost (Sec.) and classification accuracy (Acc.). We report 3 typical results in Fig. 8. The accuracy turns stable after the training samples exceed 8K. However, the time cost has increased significantly with samples increasing, especially on SVC. For trading off the cost and performance, we chose 8k samples for optimizing $\mathbf{w}_d$. Compared with other methods, our method has improved accuracy and saved time costs.

4.3. $\mathbf{w}_d$ Trim

As shown in Fig. 9, the first two rows show different $Norm$ distributions of Musk's $\mathbf{w}_y$. The 3rd row shows different $\mathbf{w}_d$ distributions which were optimized from 3 classifiers on the same attribute (age). The 4th row shows the trimmed $\mathbf{w}_d$ distributions which were optimized from LR ($\ell_1 - norm$).

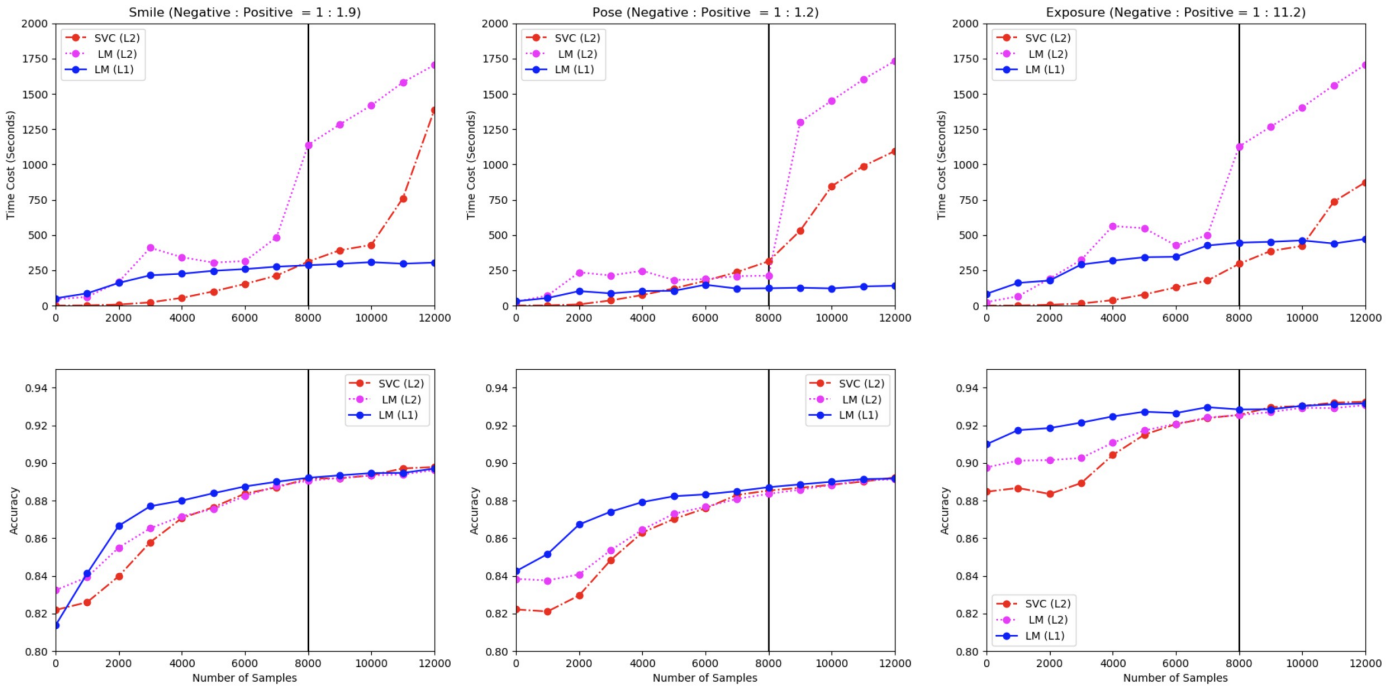


Figure 8: Comparison of 3 supervised methods for learning $\mathbf{w}_d$, the first row shows the cost times. The second row shows the accuracies.

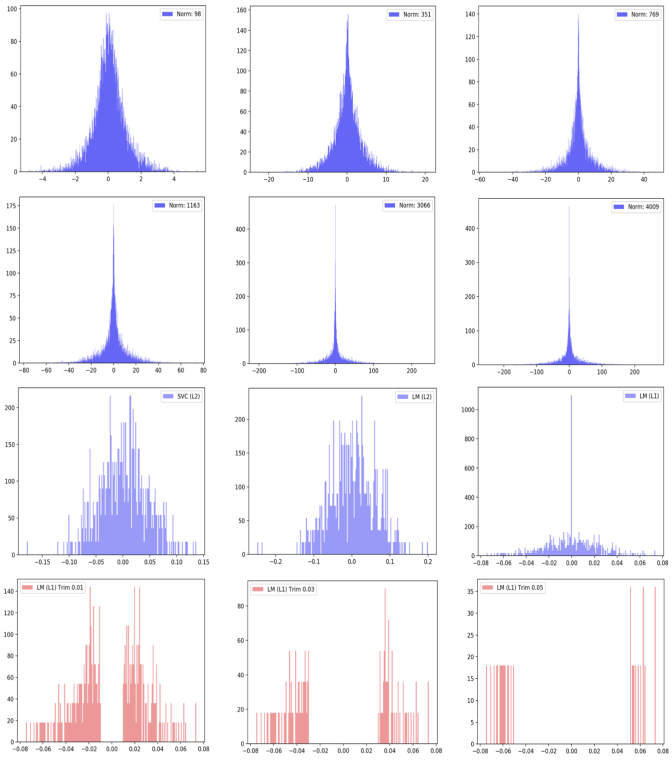


Figure 9: Different distributions of $\mathbf{w}_y$ (row 1,2), $\mathbf{w}_d$ (row 3), and trimmed $\mathbf{w}_d$ (row 4).

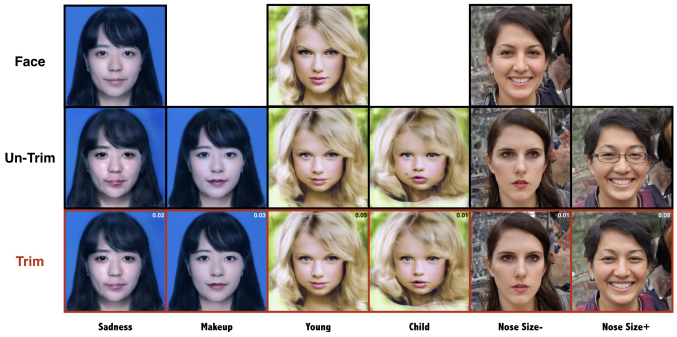


Figure 10: Comparison of trimmed and un-trimmed $\mathbf{w}_d$. The 2nd row shows un-trim results: sadness is coupled with hair colour (col. 1); makeup is coupled with exposure (col. 2); age is coupled with nevus (col. 3-4); nose size (+) is coupled with lips (col. 5) and nose size (-) is coupled with eyeglasses (col. 6).

Figure 11: RFM via fast 2-Step regularization.

To match $\mathbf{w}_y$ and $\mathbf{w}_d$, we aim to optimize both to a similar distribution shape. $\mathbf{w}_d$ optimized from LR ($\ell_1 - norm$) is same shape with $\mathbf{w}_y$ while $\mathbf{w}_y$ falls into $Zone_3$.

Based on same distribution shape, we added a trim to $\mathbf{w}_d$. Let $\mathbf{w}_d$ keep the features of $\mathbf{y}$, while highlight the $\mathbf{d}$ representation as much as possible. Performing trim is simple: we set a threshold value $\delta = [0.01, 0.05]$ and $\mathbf{w}_d = 0$ when $\mathbf{w}_d < \delta$, so that the each dimension value of $\mathbf{w}_d$ is larger than δ.

As shown in Fig. 10, we compared trimmed with untrimmed $\mathbf{w}_d$. When we moved the same distance (α), trimmed $\mathbf{w}_d$ was more decoupled than untrimmed $\mathbf{w}_d$. Based on the well regularized $\mathbf{w}_y$ and $\mathbf{w}_d$, we moved the trimmed $\mathbf{w}_d$ could perform frame by frame RFM (see Fig. 11).

4.4. Comparisons

Table 3:
Comparing baselines via 4 different metrics.

Attribute	Methods	PSNR ↑	SSIM ↑	LPIPS ↓	FID ↓
Inversion	LatentCLR [24]	24.36	0.741	0.359	93.69
	GANSpace [15]				
	InterfaceGAN [12]	25.14	0.750	0.348	106.58
	Ours	**26.63**	**0.761**	**0.329**	**89.26**
Gender	LatentCLR [24]	17.49	0.656	0.487	177.94
	GANSpace [15]	17.62	0.649	0.453	163.62
	InterfaceGAN [12]	17.97	0.672	0.436	173.27
	Ours	**19.04**	**0.673**	**0.433**	**145.01**
Age	LatentCLR [24]	15.77	0.640	0.505	242.41
	GANSpace [15]	18.44	0.668	0.460	171.01
	InterfaceGAN [12]	20.71	0.685	0.414	161.87
	Ours	**21.17**	**0.706**	**0.389**	**127.65**
Hair	LatentCLR [24]	19.52	0.698	0.438	152.95
	GANSpace [15]	17.78	0.661	0.454	150.10
	InterfaceGAN [12]	18.75	0.692	0.407	130.02
	Ours	**20.85**	**0.708**	**0.387**	**119.94**

We compared our method with 3 open source methods: InterfaceGAN [12], GANSpace [15], and LatentCLR [24]. We evaluated 4 attributes: GAN inversion (image reconstruction) and three directions (gender, age, hair). GANSpace and LatentCLR are unsupervised methods and have not implemented GAN inversion for real images. So we used [19] to perform their inversions and only evaluated inversion results for LatentCLR. On each method implementation, we used the same sample size (8,000) to learn each direction, and we randomly synthesized 1k morphed images respectively to measure FID.

As shown in Table 3, our method outperforms other methods in the quantitative evaluations. We illustrate the morphed cases in Fig. 13. We also conducted a subjective test to evaluate results on human perception. We asked one question: "which is better for visual quality while preserving human identity ?". Let 40 audiences judge the morphed results among the 4 methods. The test results are shown in Fig. 12.

The results show that supervised methods are better than unsupervised methods. We believe that some specific directions (e.g., age and hair) are difficult to find in unsupervised directions while training samples are limited. In supervised methods, our method is better than InterfaceGAN due to better GAN inversion, 2-step regularization and direction trim.

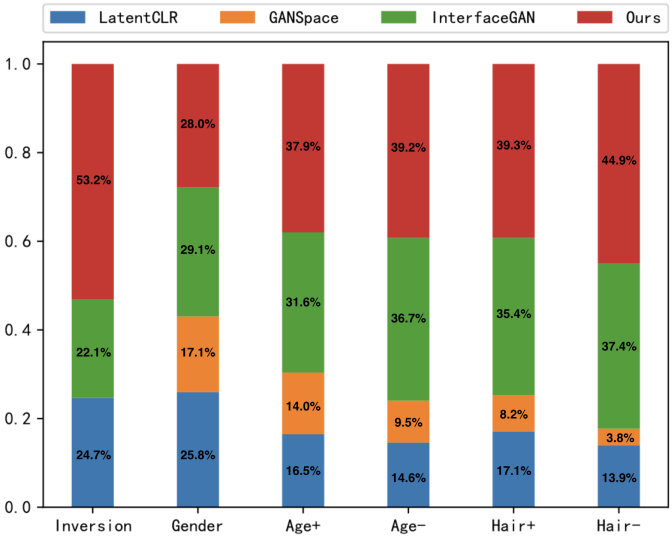


Figure 12: Subjective test for the comparison among 4 methods.

5. Conclusion and Future Work

We proposed a simple method for fast real face morphing. Firstly, we labelled face attributes on latent vectors and build the label set. Next, we merged GAN inversion into interpretable direction via 2-step regularization. Finally, we proposed the layer-wise guideline for moving directions. In addition, we trimmed the directions for better morphing performance. Compared to previous works, our method outperformed other methods while limited labels. Next, we intend to explore interpretable directions on more datasets and GANs via unsupervised manner and neural networks.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This work was supported by the Science and Technology Development Fund (FDCT) of Macau (0016/2019/A1).

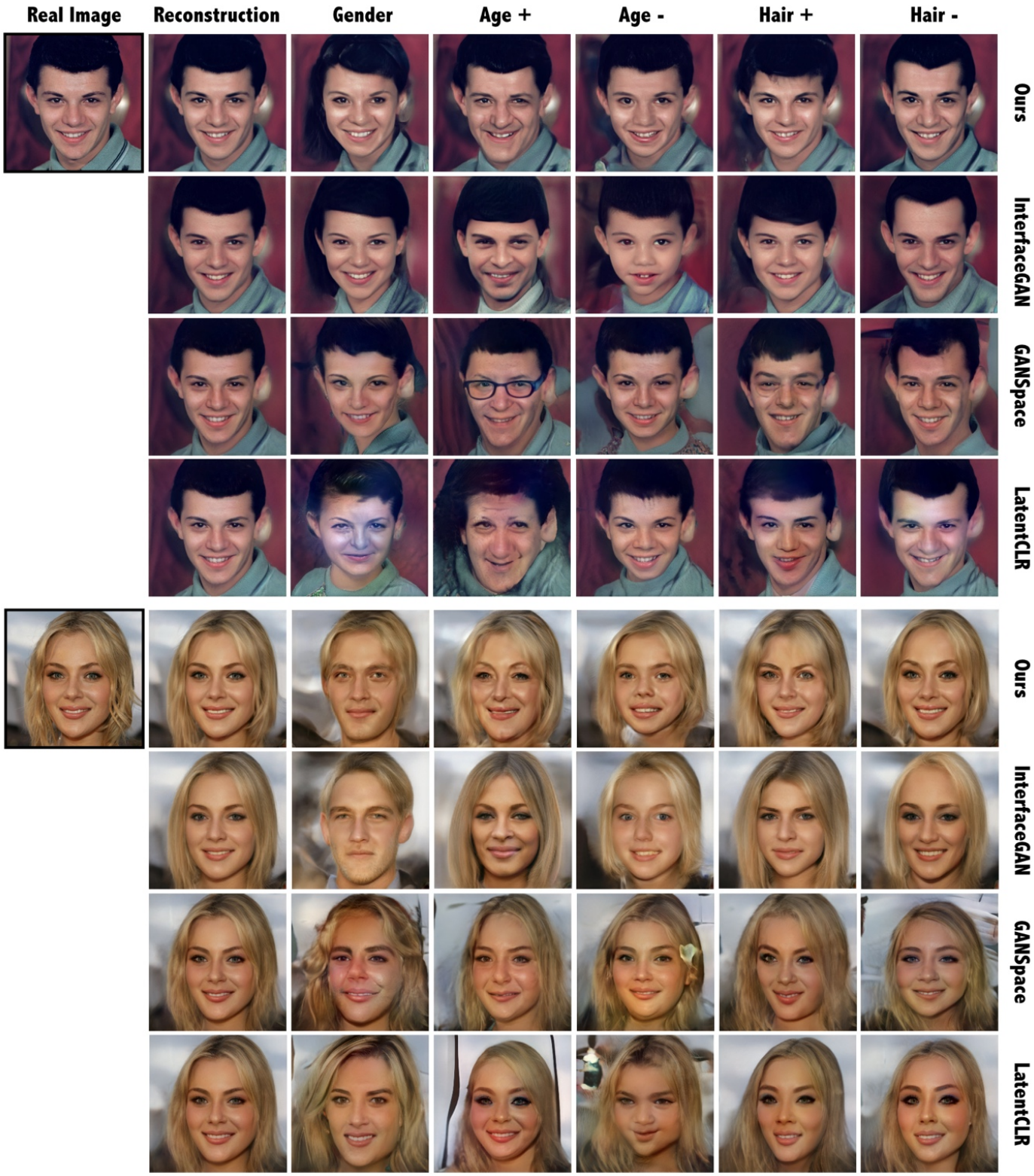


Figure 13: Visual comparison among our method, InterfaceGAN [12], GANSpace [15], and LatentCLR [24].

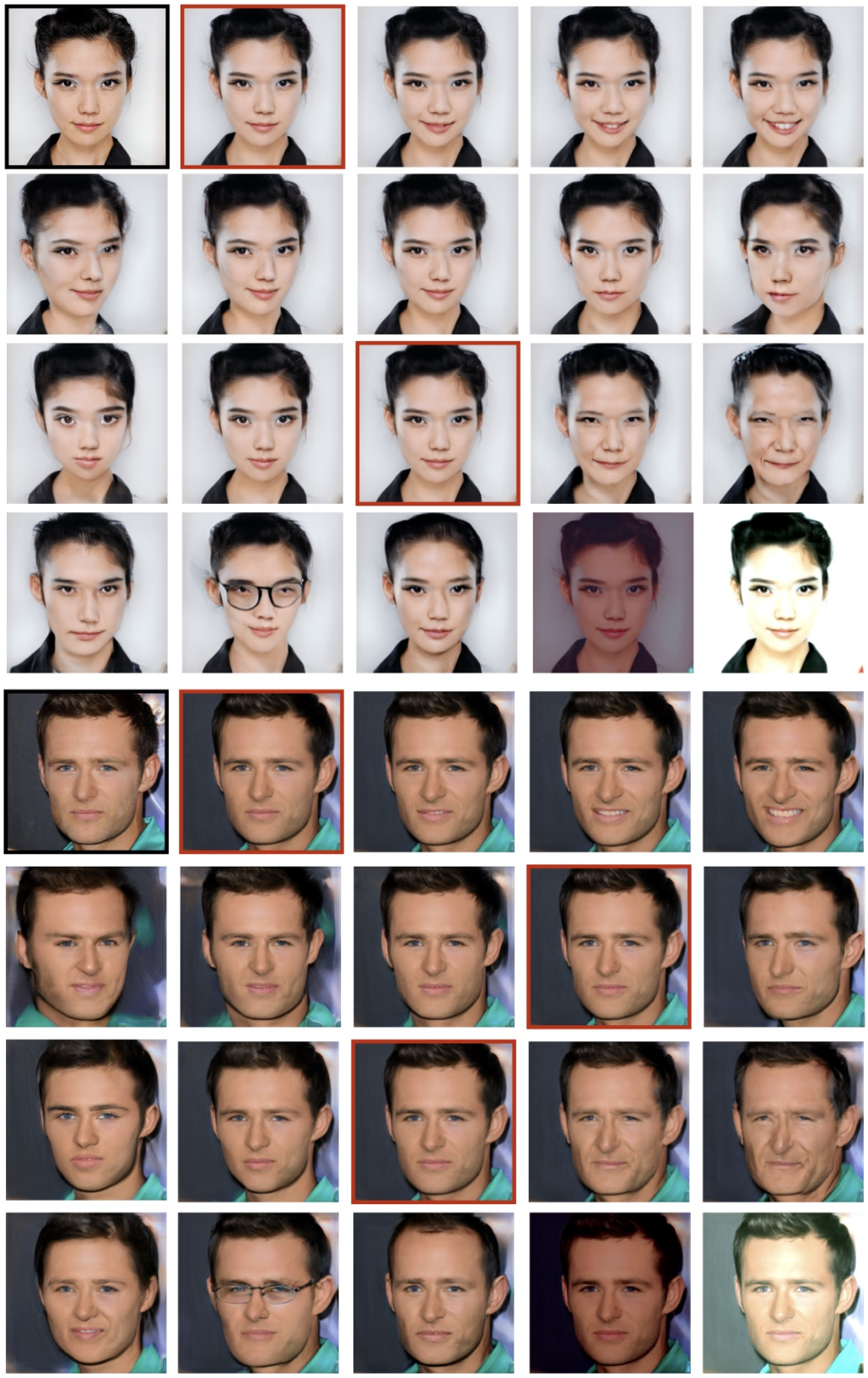


Figure 14: Two real face cases. The black window indicates the real face and the red window indicates its reconstruction. For one face, the first three rows show the results about smile, pose, and age. The last row lists the other results (gender, eyeglasses, hair, less or over-exposure).

References

[1] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial nets, in: *Proc. Int. Conf. Neural Inf. Process. Syst. (NIPS)*, Vol. 27, 2014, pp. 2672–2680.

[2] Z. Liu, P. Luo, X. Wang, X. Tang, Deep learning face attributes in the wild, in: *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, 2015, pp. 3730–3738.

[3] T. Karras, S. Laine, T. Aila, A style-based generator architecture for generative adversarial networks, in: *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2019.

[4] A. Radford, L. Metz, S. Chintala, Unsupervised representation learning with deep convolutional generative adversarial networks, in: *Int. Conf. Learn. Represent. (ICLR)*, 2016.

[5] T. Karras, T. Aila, S. Laine, J. Lehtinen, Progressive growing of gans for improved quality, stability, and variation, in: *Int. Conf. Learn. Represent. (ICLR)*, 2018.

[6] A. Brock, J. Donahue, K. Simonyan, Large scale GAN training for high fidelity natural image synthesis, in: *Int. Conf. Learn. Represent. (ICLR)*, 2019.

[7] R. Abdal, Y. Qin, P. Wonka, Image2stylegan: How to embed images into the stylegan latent space?, in: *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, 2019, pp. 4431–4440.

[8] C. Yu, W. Wang, Fast transformation of discriminators into encoders using pre-trained gans, Pattern Recognition Letters 153 (2022) 92–99.

[9] C. Yu, W. Wang, Adaptable gan encoders for image reconstruction via multi-type latent vectors with two-scale attentions (2021). `arXiv:2108.10201`.

[10] J. Zhu, Y. Shen, D. li Zhao, B. Zhou, In-domain gan inversion for real image editing, in: *Europ. Conf. Comput. Vis. (ECCV)*, 2020.

[11] S. Pidhorskyi, D. A. Adjeroh, G. Doretto, Adversarial latent autoencoders, in: *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2020.

[12] Y. Shen, C. Yang, X. Tang, B. Zhou, Interfacegan: Interpreting the disentangled face representation learned by gans, IEEE Trans. on Pattern Analysis and Machine Intelligence (TPAMI).

[13] X. Hou, X. Zhang, H. Liang, L. Shen, Z. Lai, J. Wan, Guidedstyle: Attribute knowledge guided style manipulation for semantic face editing, Neural Netw. 145 (C) (2022) 209–220.

[14] R. Abdal, P. Zhu, N. J. Mitra, P. Wonka, Styleflow: Attribute-conditioned exploration of stylegan-generated images using conditional continuous normalizing flows, ACM Trans. Graph. 40 (3) (2021) 21:1–21:21.

[15] E. Härkönen, A. Hertzmann, J. Lehtinen, S. Paris, Ganspace: Discovering interpretable GAN controls, in: *Proc. Int. Conf. Neural Inf. Process. Syst. (NIPS)*, 2020.

[16] Z. He, M. Kan, S. Shan, Eigengan: Layer-wise eigen-learning for gans, in: *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, 2021.

[17] Jerome, Friedman, Trevor, Hastie, Rob, Tibshirani, Regularization paths for generalized linear models via coordinate descent., Journal of statistical software 33 (1) (2010) 1–22.

[18] S. Kim, K. Koh, M. Lustig, S. P. Boyd, D. Gorinevsky, An interior-point method for large-scale ℓ_1-regularized least squares, J. Sel. Topics Signal Processing 1 (4) (2007) 606–617.

[19] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, T. Aila, Analyzing and improving the image quality of stylegan, in: *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2020, pp. 8107–8116.

[20] W. Xia, Y. Zhang, Y. Yang, J. Xue, B. Zhou, M. Yang, GAN inversion: A survey, arXiv preprint abs/2101.05278.

[21] C. Yang, Y. Shen, B. Zhou, Semantic hierarchy emerges in deep generative representations for scene synthesis, Int. J. Comput. Vis. 129 (5) (2021) 1451–1466.

[22] D. Bau, J. Zhu, H. Strobelt, B. Zhou, J. B. Tenenbaum, W. T. Freeman, A. Torralba, GAN dissection: Visualizing and understanding generative adversarial networks, in: *Int. Conf. Learn. Represent. (ICLR)*, 2019.

[23] F. Yu, Y. Zhang, S. Song, A. Seff, J. Xiao, LSUN: construction of a large-scale image dataset using deep learning with humans in the loop, arXiv preprint abs/1506.03365.

[24] O. K. Yüksel, E. Simsar, E. G. Er, P. Yanardag, Latentclr: A contrastive learning approach for unsupervised discovery of interpretable directions, in: *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, 2021, pp. 14243–14252.

[25] T. Chen, S. Kornblith, M. Norouzi, G. E. Hinton, A simple framework for contrastive learning of visual representations, in: *Proc. Int. Conf. Mach. Learn. (ICML)*, Vol. 119 of Proceedings of Machine Learning Research, PMLR, 2020, pp. 1597–1607.

[26] O. Patashnik, Z. Wu, E. Shechtman, D. Cohen-Or, D. Lischinski, Styleclip: Text-driven manipulation of stylegan imagery, in: ICCV, IEEE, 2021, pp. 2065–2074.

[27] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, I. Sutskever, Learning transferable visual models from natural language supervision, in: *Proc. Int. Conf. Mach. Learn. (ICML)*, Vol. 139 of Proceedings of Machine Learning Research, PMLR, 2021, pp. 8748–8763.

[28] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, I. Polosukhin, Attention is all you need, in: *Proc. Int. Conf. Neural Inf. Process. Syst. (NIPS)*, 2017, pp. 5998–6008.

[29] C. Chang, C. Lin, LIBSVM: A library for support vector machines, ACM Trans. Intell. Syst. Technol. 2 (3) (2011) 27:1–27:27.

[30] Zhou Wang, A. C. Bovik, H. R. Sheikh, E. P. Simoncelli, Image quality assessment: from error visibility to structural similarity, *IEEE Trans. Image Process.* 13 (4) (2004) 600–612.

[31] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, O. Wang, The unreasonable effectiveness of deep features as a perceptual metric, in: *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2018.

[32] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, S. Hochreiter, Gans trained by a two time-scale update rule converge to a local nash equilibrium, in: *Proc. Int. Conf. Neural Inf. Process. Syst. (NIPS)*, 2017, pp. 6626–6637.

[33] D. Nikitko, Stylegan encoder for official tensorflow implementation, `https://github.com/Puzer/stylegan-encoder` (2019).