
Supplementary Material: Low-Rank Head Avatar Personalization with Registers

Anonymous Author(s)

Affiliation

Address

email

1 The appendix is organized as follows:

2 1. Additional Ablation Study in Sec. A

3 2. Implementation Details in Sec. B

4 3. Dataset Collection Details in Sec. C

5 4. Additional Results in Sec. D

6 5. User Study Details in Sec. E

7 6. Discussion: Broader Impact, Limitations, and Ethical Considerations in Sec. F

8 We strongly encourage the readers to watch our supplementary video.

9 A Additional Ablation Study

Table 1: Ablation study on losses to adapt with our method. In (a), we remove L_{feat} during adaptation. In (b), we remove L_{reg} during adaptation.

Method	LPIPS↓	ACD↓
(a) Ours w/o L_{feat}	0.2574	0.3604
(b) Ours w/o L_{reg}	0.2508	0.3574
Ours	0.2470	0.3559

Table 2: Ablation study on length of videos to adapt the head avatar. We set the adaptation video length to 4, 2, and 1 seconds.

Method	LPIPS↓	ACD↓
(a) Ours w/ 4 sec	0.2471	0.3639
(b) Ours w/ 2 sec	0.2471	0.3656
(c) Ours w/ 1 sec	0.2477	0.3687

10 We conduct additional ablation studies on our proposed method. Specifically, we ablate the various
11 losses we propose. In Table 1(a), we remove the loss L_{feat} to supervise the output of our register
12 module during adaptation, and in (b) we remove L_{reg} to make the learned embeddings in our Register
13 Module different from each other. We see that removal of these losses causes a drop in performance
14 as compared to when both losses are present. Furthermore, we ablate on the length of the videos used
15 to adapt our head avatars in Table 2. We see that reducing the length of the adaptation video causes a
16 drop in the performance of the method. Note that we trim the original videos to the first 4, 2, and 1
17 second for these experiments.

18 B Implementation Details

19 B.1 Addition of LoRA to layers

20 We use minLoRA (Chang and Kelly) library to add LoRA (Hu et al., 2022) parameters to all
21 layers of a pretrained pytorch model. During training LoRA is instantiated as separate parameters
22 $B \in \mathbb{R}^{m \times r}$, $A \in \mathbb{R}^{r \times n}$ and $r \ll \min(m, n)$ from the pretrained parameters $W_{pre} \in \mathbb{R}^{m \times n}$ of the
23 module, so that these parameters can be trained. During inference the LoRA parameters are *merged*
24 with the pretrained parameters and assigned as the new pretrained parameters according to:

$$W_{adapt} := W_{pre} + \Delta W = W_{pre} + BA. \quad (1)$$

25 This ensures that the LoRA layers add no overhead to the pipeline during the inference. Given a
26 pretrained GAGAvatar model, we add LoRA weights to all layers except in the DINOv2 feature
27 extractor.

28 B.2 Dataset Preprocessing

29 Given the expression code, shape code, and camera pose predicted during 3DMM fitting, we predict
30 a 3DMM mesh. We compute visible vertices of 3DMM mesh from the computed camera pose of a
31 particular frame using trimesh’s RayMeshIntersector implementation. Specifically, we cast rays from
32 the camera origin to each point in the mesh and compute whether any line intersects the mesh (if an
33 intersection exists, the point is not visible). Given these visible points in 3D space, we find a screen
34 space camera projection on a screen of the same size as DINOv2 feature space ($H = W = 296$)
35 using Perspective Projection. Then we find an alpha-shape of these projected points with $\alpha = 0.065$.
36 Next, we compute all the points in the alpha-shape polygon using a parallelized point-in-polygon test.
37 For all points in the polygon that are not projected points, we also compute the k nearest projected
38 points and distances from those projected points, where $k = 11$.

39 B.3 Register Module Details

40 The embeddings rigged to vertices on the 3DMM mesh (Li et al., 2017) are modeled in a 3D space in
41 our register module. However, these points are projected onto a 2D space using a camera projection
42 following which, we interpolate the features using a weighted sum of the k nearest neighbors to fill
43 up the face region in densely constructed feature f_S . Using the entire set of the vertices would cause
44 all points to be projected onto the densely constructed feature f_S , thus impacting the interpolation
45 process (projection of points from the back of the head might be in the k nearest neighbors of a
46 point $p \in \text{interior}(P_{U_S})$). Thus, we only project visible points from any given view in our register
47 module.

48 We model E_{proc-1} as a convolutional module, with 4 convolutional layers of channel sizes
49 $[512, 512, 256, 256]$ and kernel sizes set to 3 for each layer. E_{proc-2} is also a convolutional module
50 with 4 layers of channel sizes set to 256. The first 3 layers have a kernel size of 3, while the last layer
51 has a kernel size of 1.

52 B.4 Training Details

53 B.4.1 Our Method

54 We initialize embeddings e using Xavier Normal initialization (Glorot and Bengio, 2010). We adapt
55 head avatars with our method for a total of 1000 iterations. The batch size is set to 2. We use
56 Adam (Kingma and Ba, 2017) optimizer with learning rate set to $1e-4$ for the LoRA layers whereas
57 the learning rate is set to $1e-3$ for parameters in the Register Module. We use a linear learning rate
58 scheduler with a start factor of 1.0 and an end factor of 0.1 at the 1000th iteration. Along with our
59 proposed losses, we also keep the losses proposed by GAGAvatar Chu and Harada (2024), namely,
60 RGB losses between predicted image and driving image, a perceptual loss between the predicted
61 images and driver image, and $L_{lifting}$, loss between predicted points from reconstruction branch and
62 vertices of the 3DMM mesh fitted on the driving image. Our adaptation takes ≈ 35 minutes on an
63 RTX A5000 GPU, consuming ≈ 23 GB of VRAM. During inference, we load and merge the LoRA
64 weights into their corresponding layer parameters. Thus, there is no overhead during inference, i.e., it
65 consumes the same amount of resources as GAGAvatar during inference.

66 B.4.2 Baselines and Ablations

67 For all comparisons with the vanilla LoRA, we use the same hyperparameters as our method. That is,
68 we adapt with vanilla LoRA for a total of 1000 iterations, set the batch size to 2, use Adam optimizer
69 with learning rate set to $1e - 4$ for the LoRA layers, and use a linear learning rate scheduler with a
70 start factor of 1.0 and an end factor of 0.1 at the 1000th iteration. This adaptation takes ≈ 25 minutes
71 on an RTX A5000 GPU, consuming ≈ 14.9 GB of VRAM.

72 Following MetaPortrait (Zhang et al., 2023), we implement Reptile Nichol and Schulman (2018), a
73 MAML based strategy for meta-learning in our low-rank adaptation task. We perform pre-training
74 on our RareFace-50 dataset. Following the formulation of MetaPortrait, we formulate the task of
75 adapting to a particular identity as an inner loop task. Thus, we adapt to a randomly sampled identity
76 at each outer step. For all comparisons with Meta-Learning on LoRA, we set rank $r = 32$, the inner
77 loop learning rate to $2e - 4$, and the outer update step size to $2e - 5$. The number of inner loop steps
78 are set to 120, and number of elements in batch in inner loop is set to 4. We set the number of outer
79 iterations to 4800. Given resource constraints, we implement a single GPU version of Reptile (Nichol
80 and Schulman, 2018), thus taking 12 days to complete the pretraining task on a Quadro RTX 8000
81 GPU, consuming ≈ 45 GB of VRAM. After the pre-training task, we adapt the model on an identity
82 for 120 steps with the same learning rate as the inner loop, which takes ≈ 4 minutes on an RTX
83 A5000 GPU consuming ≈ 14.9 GB of VRAM. We then use these adapted weights for inference by
84 merging these LoRA weights to the corresponding layers.

85 For all experiments with learnable parameters e_{learn} as a replacement to our Register Module, we set
86 the learning rate for e_{learn} parameters to be $1e - 3$.

87 B.5 Metrics

88 We compute the visual quality metrics namely, LPIPS Zhang et al. (2018), on specific challenging
89 crops with high frequency details from predicted frames and compare them against source image
90 patches. The identity preservation metric (ACD) is measured using ArcFace Deng et al. (2019), a
91 ResNet50-based network trained on WebFace Zhu et al. (2021). Specifically, we used “buffalo_l”
92 model from the insightface repository Guo et al..

93 C Data Collection

94 We collect data from Youtube of people knowingly appearing in interviews in public broadcasts with
95 distinctive facial details, such as wrinkles or tattoos. These characteristics are under-represented in
96 existing datasets. We will present the dataset as a set of links, along with trim times and crop position
97 coordinates. Additionally, this dataset will be maintained using an automatic script that checks and
98 removes links from the list that no longer exist in Youtube.

99 D Additional Results

100 D.1 Details of Adaptation

101 **Adaptation Duration.** In this section, we discuss the adaptation durations of our baselines and our
102 method. Vanilla LoRA and our method take ≈ 25 minutes and ≈ 35 minutes to adapt respectively on
103 an RTX A5000 GPU. Whereas, meta-learning on LoRA requires a much longer 12 day period on
104 an RTX Quadro 8000 GPU for the pretraining objective, after which it requires ≈ 4 minutes on a
105 RTX A5000 GPU. However, during inference, all of these methods have the same inference times as
106 GAGAvatar Chu and Harada (2024).

107 **Adaptation Parameters.** In this section, we discuss the number of parameters during adaptation
108 and inference of our baselines and our method. During adaptation, we introduce 4.7M parameters as
109 LoRA weights to the pretrained layers in all baselines. Our Register Module adds another 18.5M
110 parameters. Thus, our method has 23.2M parameters during adaptation, which is $\approx 11\%$ of the total
111 number of parameters (199M parameters) in GAGAvatar. During adaptation, we discard the trained
112 Register Module, which lends to the same efficiency as GAGAvatar and other baselines.



Figure 1: We show the facial details are well captured by our method with Register Module, such as wrinkles and skin folds are realistic and have higher quality than vanilla LoRA. Please note the enlarged insets of specific details.

113 D.2 User Study

114 We conduct a user study to qualitatively and subjectively compare our method against LoRA (see
 115 Sec. E for details). We find that 78.2% of the users prefer our method as compared to LoRA in terms
 116 of identity preservation. Furthermore, we find that 89.9% of the users prefer our results as compared
 117 to LoRA in terms of visual quality.

118 D.3 Additional Qualitative Results

119 In Fig. 1, we show that, compared to LoRA, our method effectively captures high-frequency details
 120 like wrinkles. We show additional results of our method against the baselines on VFHQ Test in Fig. 2
 121 and on RareFace-50 in Fig. 3. Fig 4 and 5 show visualizations of feature norms along the channel
 122 dimensions for two identities from RareFaces-50. The values are visualized using colormaps between
 123 $[-3\sigma, 3\sigma]$ for f_{dense}^{src} , and f_{reg} and $[-\sigma, \sigma]$ for f_{proc-1} . We visualize the first four channel-wise
 124 PCA components for two identities from RareFaces-50 in Fig. 6 to 11. We observe that the Register
 125 Module improves the learning signals by highlighting face regions and dampening the background
 126 regions.



Figure 2: Additional results of personalized head avatar generation on VFHQ Test. Please zoom in for better details.

E User Study Details

As mentioned in Sec. D.2, we qualitatively compare our method with vanilla LoRA as an adaptation method with a user study. We describe the details of the user study here. Fig. 12 shows the interface that we use for this user study. A total of 18 users responded to our user study. We generate head avatars given source videos from VFHQ Test and RareFace-50 using our method and LoRA. The outputs are placed side by side and the left-right orders are assigned randomly to make sure that the users are unaware of which method is ours. Each generated video is ≈ 5 to 10 seconds long, concatenated with the source identity image and the driving video. Users are asked two questions: “Which method’s avatar best looks like the source image identity?” and “Which method avatar has better visual quality?” The users can choose as answer “Method A” or “Method B” or both. Label “Method A” is placed to the left of label “Method B” and the generated videos are randomly placed in



Figure 3: Additional results of personalized head avatar generation on our RareFace-50 dataset. Please zoom in for better details.

terms of a left-right order. The answers are collected through a google form. The videos are attached to the google form using a link to google drive, and the users are encouraged to download the videos to view them on their system. This is done to make sure that differences in high resolution are evident to the users.

F Discussions

F.1 Limitations

An important factor of our method is the 3DMM fitting that is used to extract the head pose, camera parameters, and 3DMM mesh parameters (see Sec. 3.3 of the main paper). This fitting can be noisy and the error can be propagated to the final generated videos. Improving the face tracking further

147 would be an interesting future work. Further, 3DMM fitting does not model asymmetric/extreme
148 expressions (such as winking) and the movement of the tongue, which is another interesting line of
149 work to pursue.

150 **F.2 Ethical Considerations and Broader Impacts**

151 While our method has significant promise across diverse applications, it also carries the risk of
152 abuse — for example, in creating “deep fakes”. These can be used by users with malicious intent
153 to spread misinformation. To prevent this, it is imperative to develop forensic tools to detect fake
154 videos Cai et al. (2023); Reiss et al. (2023). We intend to share our code, dataset and models to
155 improve this research, in which we will release them with strict licenses that only allow usage for
156 academic research. When used ethically and responsibly, our method can offer profound benefits
157 across industries — from video conferencing to the entertainment sector. In addition, we have also
158 put appropriate procedures (see Sec. C) to ensure fair and safe use of videos from the dataset we
159 collect.

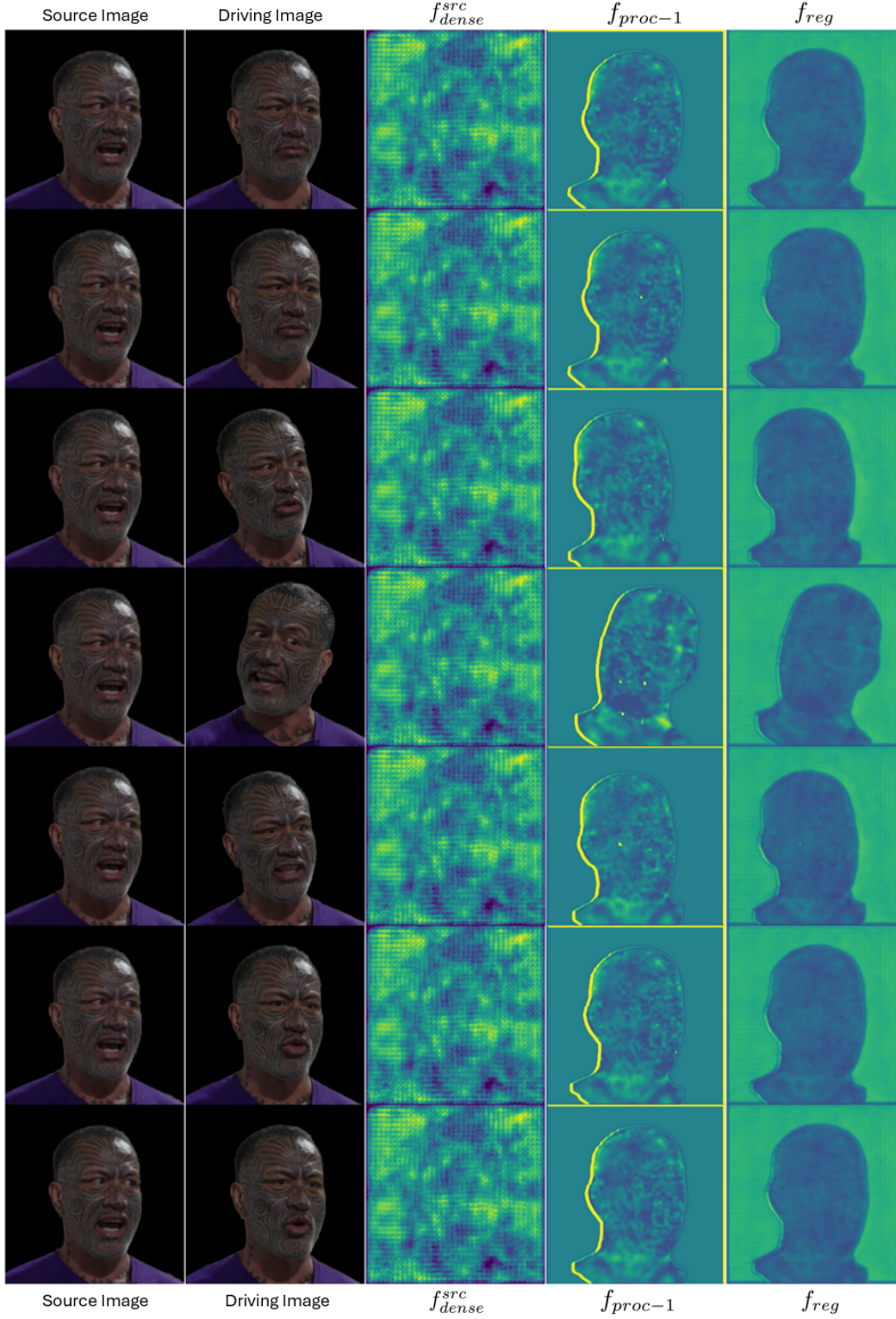


Figure 4: Visualization of learned features by the Register Module on our RareFace-50 dataset. We visualize the 1) source image’s DINOv2 feature f_{dense}^{src} , 2) f_{proc-1} , output from E_{proc-1} , and 3) f_{reg} , output of the Register Module. We compute norms of the features along the embedding dimensions and standardize values.

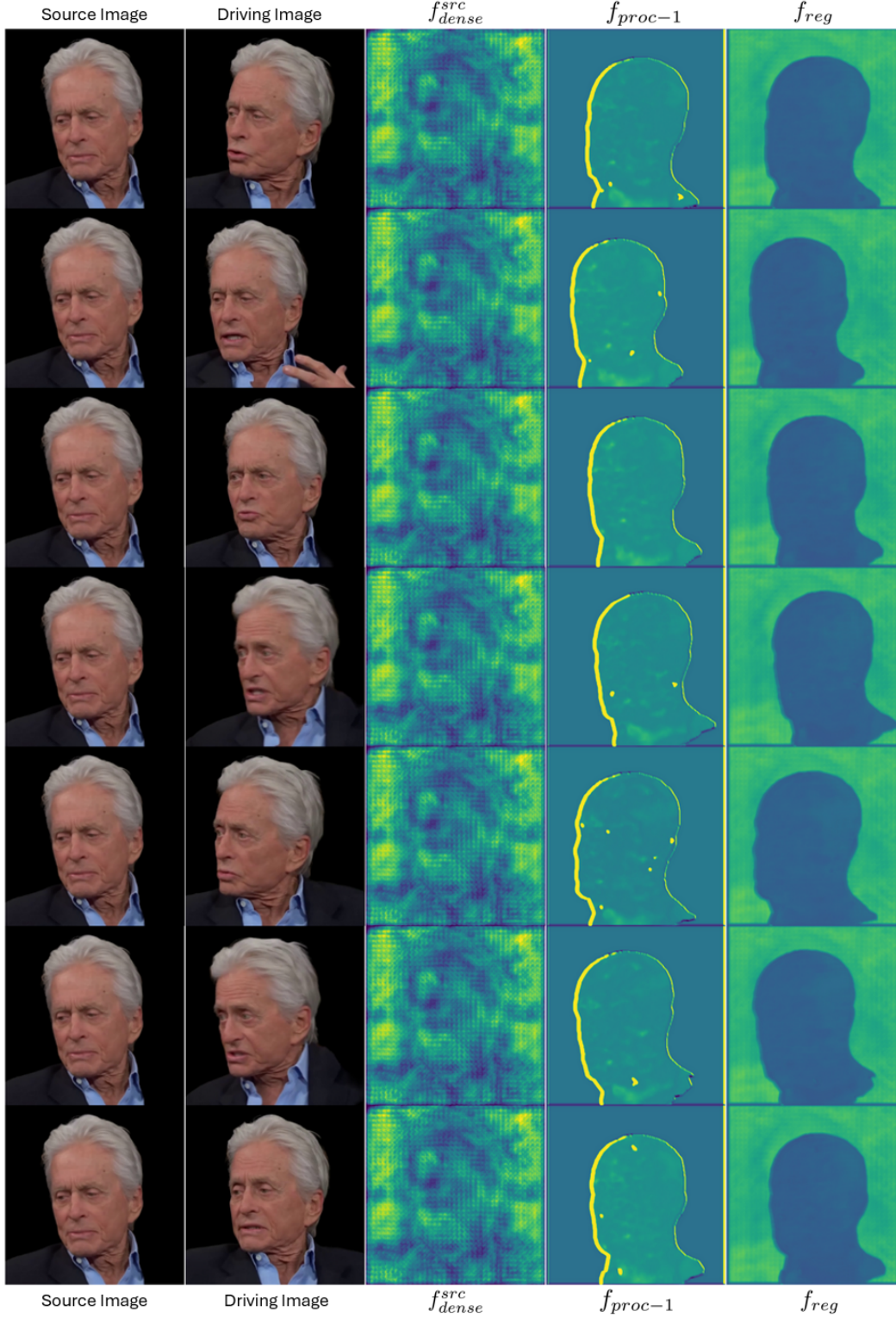


Figure 5: Visualization of learned features by the Register Module on our RareFace-50 dataset. We visualize the 1) source image’s DINOv2 feature f_{dense}^{src} , 2) f_{proc-1} , output from E_{proc-1} , and 3) f_{reg} , output of the Register Module. We compute norms of the features along the embedding dimensions and standardize values.

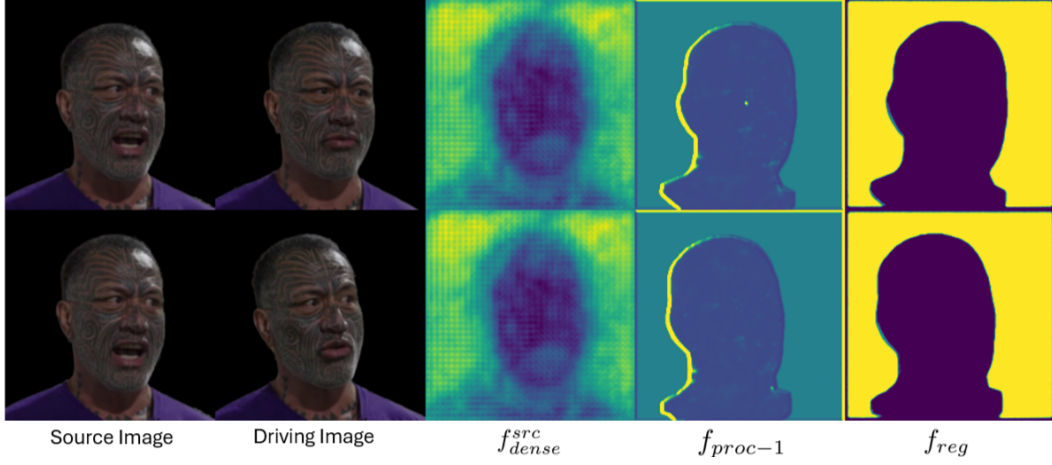


Figure 6: Visualization of learned features by the Register Module on our RareFace-50 dataset. We compute the 1st channel-wise PCA component and standardize the values.

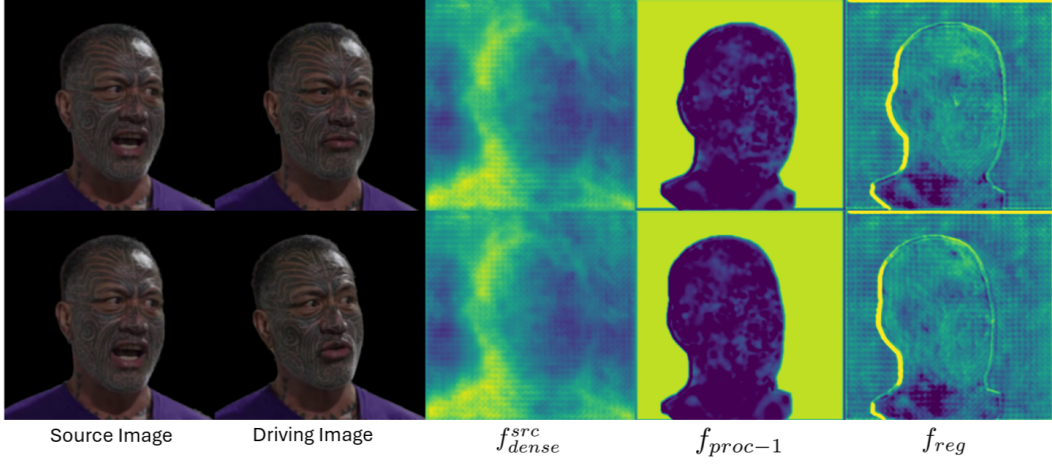


Figure 7: Visualization of learned features by the Register Module on our RareFace-50 dataset. We compute the 2nd channel-wise PCA component and standardize the values.

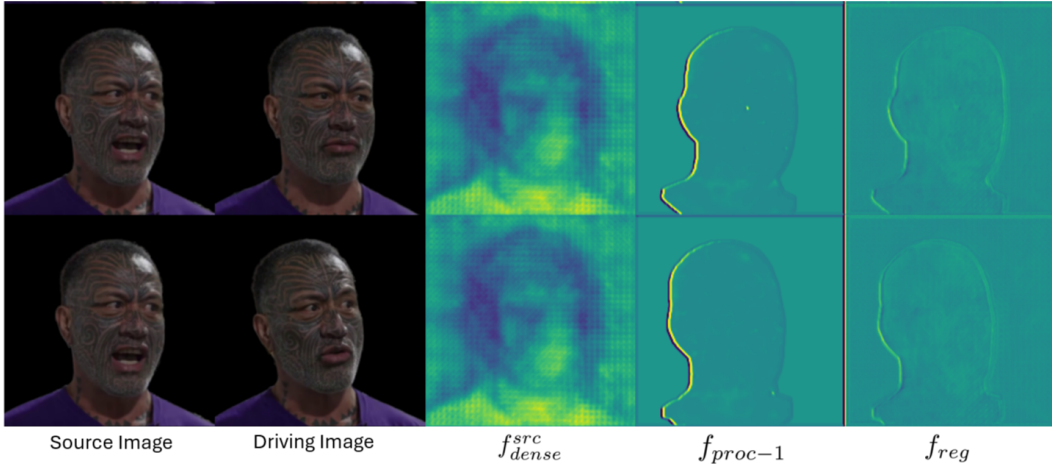


Figure 8: Visualization of learned features by the Register Module on our RareFace-50 dataset. We compute the 3rd channel-wise PCA component and standardize the values.

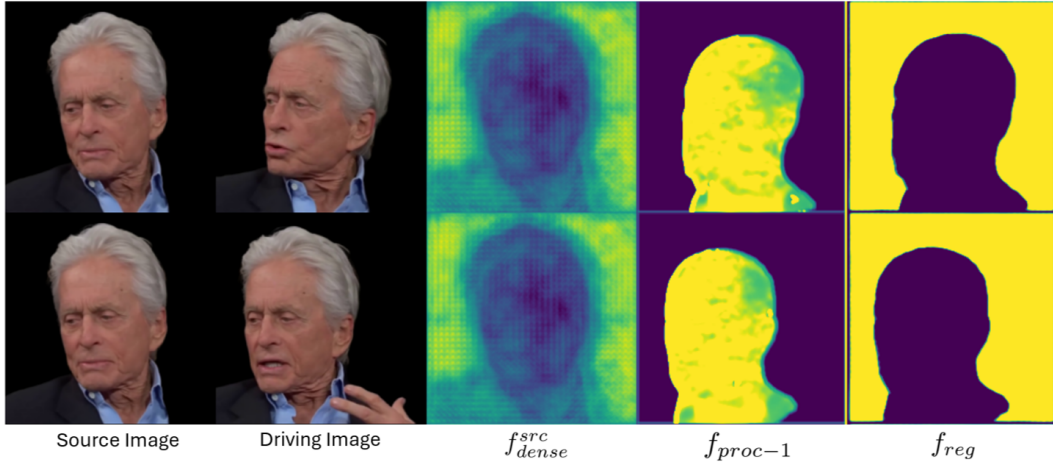


Figure 9: Visualization of learned features by the Register Module on our RareFace-50 dataset. We compute the 1st channel-wise PCA component and standardize the values.

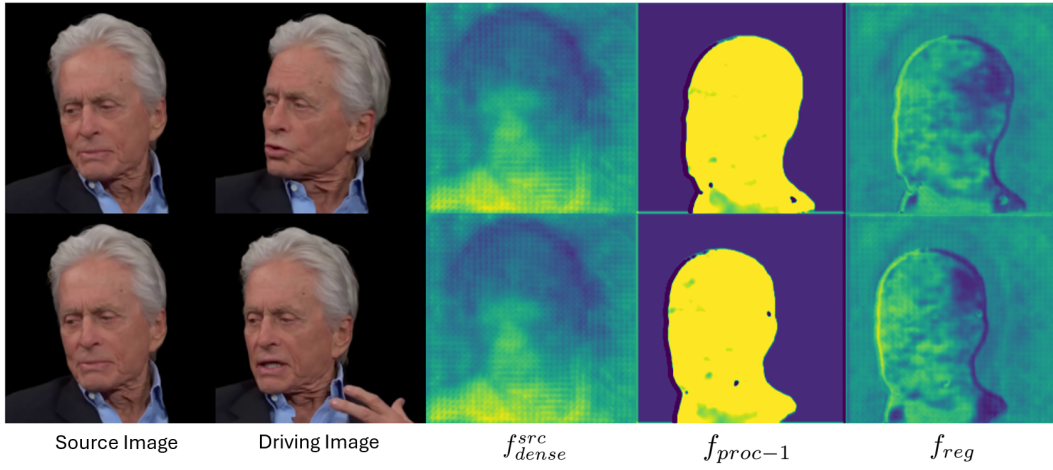


Figure 10: Visualization of learned features by the Register Module on our RareFace-50 dataset. We compute the 2nd channel-wise PCA component and standardize the values.

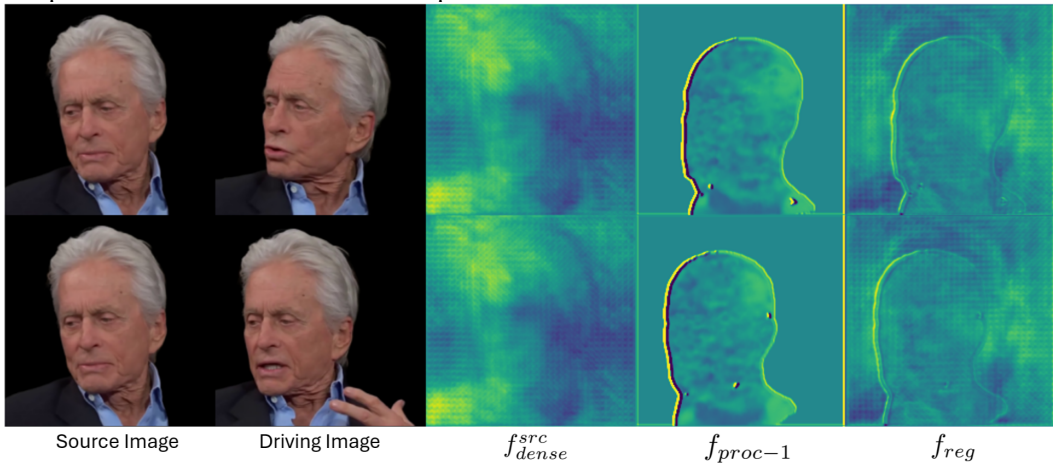


Figure 11: Visualization of learned features by the Register Module on our RareFace-50 dataset. We compute the 3rd channel-wise PCA component and standardize the values.

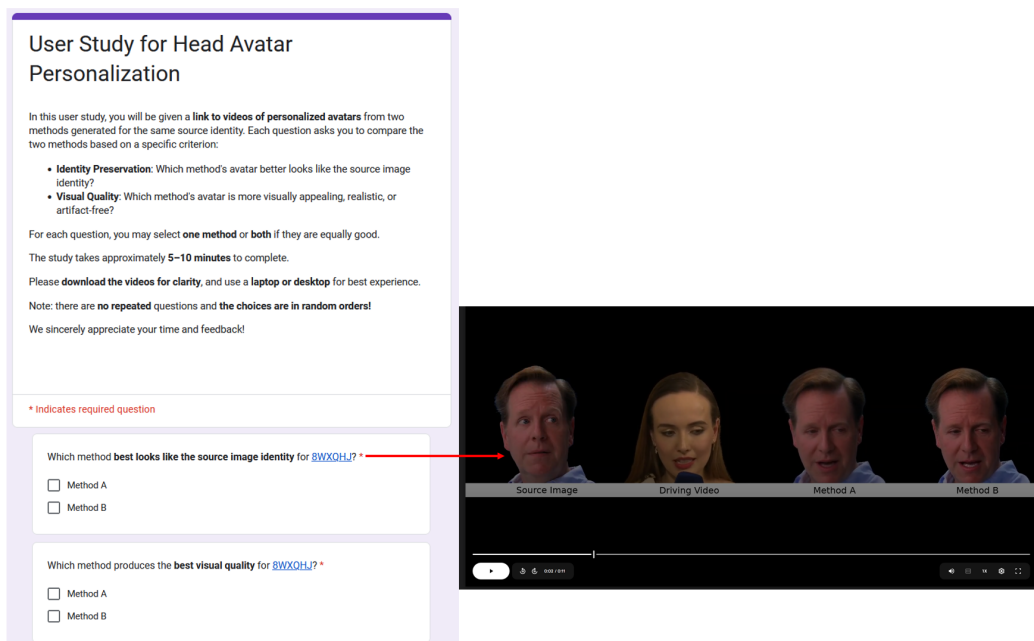


Figure 12: User Study Interface. We ask each user to watch 8 videos and answer which method preserves the source image identity and which method has the best visual quality.

References

- Zhixi Cai, Shreya Ghosh, Kalin Stefanov, Abhinav Dhall, Jianfei Cai, Hamid Rezaatofghi, Reza Haffari, and Munawar Hayat. Marlin: Masked autoencoder for facial video representation learning. In *CVPR*, 2023.
- Jonathan Chang and James Kelly. minlora: a minimal pytorch library that allows you to apply lora to any pytorch model. <https://github.com/cccntu/minLoRA>. Accessed: 2025-05-17.
- Xuangeng Chu and Tatsuya Harada. Generalizable and animatable gaussian head avatar. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.
- Jiankang Deng, Jia Guo, Niannan Xue, and Stefanos Zafeiriou. Arcface: Additive angular margin loss for deep face recognition. In *CVPR*, 2019.
- Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 249–256, Chia Laguna Resort, Sardinia, Italy, 2010. PMLR.
- Jia Guo, Jiankang Deng, Xiang An, Jack Yu, and Baris Gecer. Insightface repository model zoo.
- Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.
- Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.
- Tianye Li, Timo Bolkart, Michael J. Black, Hao Li, and Javier Romero. Learning a model of facial shape and expression from 4D scans. *ACM Transactions on Graphics, (Proc. SIGGRAPH Asia)*, 36(6):194:1–194:17, 2017.
- Alex Nichol and John Schulman. Reptile: a scalable metalearning algorithm. *arXiv preprint arXiv:1803.02999*, 2(3):4, 2018.
- Tal Reiss, Bar Cavia, and Yedid Hoshen. Detecting deepfakes without seeing any. *arXiv preprint arXiv:2311.01458*, 2023.
- Bowen Zhang, Chenyang Qi, Pan Zhang, Bo Zhang, HsiangTao Wu, Dong Chen, Qifeng Chen, Yong Wang, and Fang Wen. Metaportrait: Identity-preserving talking head generation with fast personalized adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22096–22105, 2023.
- Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018.
- Zheng Zhu, Guan Huang, Jiankang Deng, Yun Ye, Junjie Huang, Xinze Chen, Jiagang Zhu, Tian Yang, Jiwen Lu, Dalong Du, and Jie Zhou. Webface260m: A benchmark unveiling the power of million-scale deep face recognition. In *CVPR*, 2021.