# Weights2Weights++: Constructing the Weight Space for Customized Diffusion Models with VAE

Felipe Zanforlin Mautner, Yining She, Yuchen Lin
{fmautner, yiningsh, yl7} @ andrew.cmu.edu

December 13 2024*

## 1 Introduction

Recent developments in generative models, such as Generative Adversarial Networks (GANs) [4] and Diffusion Models [6, 12, 9], have provided a series of tools that can be used to synthesize realistic images. These models are trained on large datasets of images and can generate new images that are similar to the training data. Previous work DreamBooth [10] has investigated fine-tuning these models to generate images of a specific subject, such as a person's face. Instead of utilizing a latent code to represent the given subject, DreamBooth simply encodes the subject's information in the model's weights. Therefore, it is natural to hypothesize that the weights of a generative model can be compressed into a latent code which encodes subject-specific knowledge. To this end, Weights2Weights [3] initially proposed to use Principal Component Analysis (PCA) to build a latent space from a set of weights of fine-tuned diffusion models. However, PCA is a linear method and may not capture the complex relationships between the weights, resulting in a suboptimal latent space. In light of this, we propose **Weights2Weights++**, which leverages Variational Autoencoders (VAEs) [8] to construct a more expressive latent space for the weights of diffusion models. Our key insight is that VAEs are more expressive in regressing the weights, thus providing a more accurate latent space and enabling more effective manipulation of the weights. Our code is available at here.

## 2 Dataset and Task

**Task Statement** Given a set of weights $\mathcal{D} = \{\theta_1, \theta_2, \ldots, \theta_n\}$ from different diffusion models fine-tuned on several subject images using DreamBooth [10], we aim to learn a latent space representation of the weights. Hence, we can manipulate, interpolate or sample weights from the latent space to generate new diffusion models which encode the desired properties of subject information. Our implementation will be based on the codebase[1] from the original paper [3] and may use the open-source implementation of VAEs[2].

**Dataset** The original paper [3] fine-tuned over 60000 diffusion models on different subject images from [13], resulting in thousands of GPU hours. We will directly use the weights provided in the original article to construct a dataset $\mathcal{D}$ to train our VAE. We will also collect several more subject images, such as the team members' profile pictures, to personalize new diffusion models for testing and evaluation.

**Evaluation Metrics** As this project aims to improve on the basis of the original w2w paper, we will evaluate our method in the task "Inverting Subjects" (Section 4.3) using the same metric employed in [3], **ID Score**, which is designed to measure identity preservation. First, we detect faces in the original and reconstructed images using MTCNN [14]. Then we calculate the similarity of the FaceNet [11] embeddings as the final ID score.

## 3 Related Work

**Image-based Generative Models** have become a cornerstone of modern machine learning, enabling the creation of realistic and diverse visual content. Early approaches like Variational Autoencoders (VAEs) [8] and Generative Adversarial Networks (GANs) [4] laid the foundation for modeling complex image distributions. More recently, diffusion models [6, 12] have gained prominence for their ability to generate high-fidelity images through iterative denoising processes, outperforming GANs in terms of diversity and realism. These models have been extended to text-guided generation with frameworks like Latent Diffusion Models (LDM) [9], bridging vision and language tasks. By operating in a compressed latent space rather than directly in pixel space, LDMs significantly reduce computational complexity while maintaining high-quality image synthesis. LDMS also enables prompt-to-prompt editing, allowing for fine-grained control over the generated images based on textual descriptions.

*Final project for CMU 10423/10623 Generative AI, 2024 Fall
[1]https://github.com/snap-research/weights2weights
[2]https://github.com/AntixK/PyTorch-VAE

**weights2weights Space** The main motivation for our work is [3], which identifies a subspace within the weight parameters of diffusion models, functioning as a latent space for customized models. In the study, Amil et al. first fine-tuned over 60,000 latent diffusion models with the personalization technique Dreambooth [10] using low-rank approximation (LoRA) [7] to create a weights2weights (w2w) space using the 60k+ personalized model weights. By applying PCA on this large space, they reduce dimensionality by projecting onto the top 1000 principal components. Each point in the w2w space represents a unique combination of these principal components, corresponding to a model specific to a particular identity and capable of consistently generating content unique to that subject.

**Personalization Fine-tuning.** In the original paper [3], DreamBooth [10] is used for personalization fine-tuning. This is a personalized image generation method designed to fine-tune text-to-image diffusion models, allowing them to generate customized outputs that capture the unique characteristics of a given subject. By training on a few reference images, DreamBooth adapts the model to create representations that retain both the distinct features of the subject and the broader stylistic coherence of the generated content.

**Efficient Fine-tuning.** To lower the computational costs of DreamBooth, the original paper [3] uses Low Rank Approximation (LoRA) [7]. Instead of updating all parameters, only low-rank matrices are fine-tuned. As such, the w2w space in [3] is composed by 60k+ distinct LoRA parameters, not all parameters in diffusion models.

**Variational Autoencoders** (VAEs) [8] are a class of generative models that aim to learn a compressed, continuous, and probabilistic latent space representation of data. This is achieved by using an encoder network to approximate the posterior distribution of the latent variables and a decoder network to reconstruct data samples from points in this distribution. Our intuition is that, instead of applying PCA to reduce the dimensionality of w2w space, a VAE may be able to learn a distribution as a better representation of this diffusion model weight space.

**Learning from model parameters**. The original paper [3] treats the weights of learned diffusion models as data for an unsupervised learning algorithm, specifically PCA. This approach of using model parameters as data is at the core of hypernetworks [5], which use a network to generate weights for another network. Other works, such as [1], use model parameters and their gradients to learn optimizers, effectively leveraging parameter data to enhance learning across different models. In general, these works have revealed that the structure of learned model parameters can themselves be exploited in different ways.

## 4 Approach

In this section, we first briefly outline the original approach for constructing the w2w space using PCA, which from now on we denote as PCA-w2w, in [3]. Then, we introduce our novel approach, which is planned to employ a VAE to represent this space and denote it as VAE-w2w.

### 4.1 Construct the weights manifold

**Creating a dataset of model weights.** To create the PCA-w2w space, the original paper first creates a dataset of model weights $\theta_i$, using DreamBooth with LoRA fine-tuning on pre-trained Latent Diffusion models to obtain identity-specific models with the ability to control image instances using text prompts. Each model is fine-tuned on a set of images corresponding to a single human subject from the dataset. After training, all LoRA matrices are flattened and concatenated to form a vector $\theta_i \in \mathbb{R}^d$, representing one identity. By repeating this process for $N$ subjects, they produce a final dataset of model weights, $\mathcal{D} = \{\theta_1, \theta_2, \ldots, \theta_N\}$, capturing a wide range of individual identities.

**Modeling the weights manifold.** They hypothesize that the dataset $\mathcal{D} \subseteq \mathbb{R}^d$ lies on a lower-dimensional subspace of the weight space that encodes identities. They model it as a linear subspace $\mathbb{R}^m$ where $m < d$, calling it weights2weights (PCA-w2w) space. Points in this subspace are represented as a linear combination of basis vectors $W = \{w_1, ..., w_m\}$, $w_i \in \mathbb{R}^d$, attained by applying PCA on the $N$ models.

**Sampling from the weights manifold.** With the weights manifold defined, new models can be sampled from it to generate novel identities. A sampled model is represented by coefficients $\{\beta_1, \ldots, \beta_m\}$ each $\beta_i$ is drawn from a normal distribution with mean $\mu_i$ and standard deviation $\sigma_i$ for each $\theta_i$ in $\mathcal{D}$. Then the exact model weight is the linear combination of PCA bases $W = \{w_1, ..., w_m\}$ with coefficients $\{\beta_1, \ldots, \beta_m\}$.

### 4.2 Leverage VAE to represent the space

The principal components identified through PCA may not always be useful, particularly in cases where the data structure is complex and nonlinear. Additionally, selecting the number of principal components, $m$, is nontrivial and can significantly impact the space expressiveness. In the original work, $m = 1000$ is chosen without extensive justification, raising concerns about whether a fixed 1000-dimensional space is sufficient to capture the diversity of identity-specific models.

Following the limitations of PCA discussed above, we want to explore an alternative approach: applying Variational Autoencoders (VAEs) over the dataset of model weights. VAEs are particularly well-suited for this task, given their powerful ability to capture complex probability distributions in high-dimensional data. Unlike PCA, VAEs can represent nonlinear relationships within the data, making them more effective in cases where the underlying structure is complex and nonlinear. By learning a smooth and continuous latent space, VAEs can capture the nuances of identity-specific features in model weights which are easily sampled.

**Training the VAE on the Dataset of Model Weights.** We will train a VAE model on the dataset of model weights, $\mathcal{D} = \{\theta_1, \theta_2, \ldots, \theta_N\}$. The model will contain an encoder that maps each $\theta_i$ to a latent distribution, and a decoder that reconstructs $\theta_i$ from samples drawn from this distribution. We apply KL Weight Annealing [2] for better training results.

**Sampling New Models from the VAE.** To generate a novel model, we just need to sample a point from the latent Gaussian space, which represents a potential identity configuration, and pass it through the decoder, yielding a new model's weights.

### 4.3 Inversion into Weight Space

The original paper designs a gradient-based method of inverting a single identity from an image into the PCA-w2w space (Sec 3.4 [3]). We take this method as the baseline for evaluation.

Specifically, the inversion with PCA bases constrains the model weights to lie in PCA-w2w space by optimizing a set of basis coefficients $\{\beta_1, ..., \beta_m\}$ rather than the original LoRA parameters. Given a single image $\mathbf{x}$, they follow a constrained denoising objective:

$$\min_{\theta} \; \mathbb{E}_{\mathbf{x},\mathbf{c},\epsilon,t}\left[w_t \|\epsilon - \epsilon_\theta(\mathbf{x}_t, \mathbf{c}, t)\|_2^2\right] \quad \text{s.t. } \theta \in \text{PCA-w2w}$$

where $\epsilon_\theta$ is the denoising UNet, $\mathbf{x}_t$ is the noised version of the latent for an image, $\mathbf{c}$ is the conditioning signal, $t$ is the diffusion timestep, and $w_t$ is a time-dependent weight on the loss.

In practice (code), they defined a wrapper pytorch module whose parameters are $\{\beta_1, ..., \beta_m\}$. The forward function defined in the wrapper module would first compose the exact model weight using PCA bases and $\beta$s, then call the forward function of this composed model. The inversion program initializes the $\beta$s as zero and then fine-tunes the wrapper model on the single image as a normal training process except for restricting the optimizer to only optimize $\beta$s.

**Our approach: Inversion with VAE.** Inspired by the baseline method, we can invert an identity image into the VAE-w2w space with our VAE in a similar way. We constrain the model weights to lie in VAE-w2w space by optimizing the latent variable $z$ that is passed into the VAE decoder to generate the inverted model. Given a single image $\mathbf{x}$, we follow a constrained denoising objective:

$$\min_{z} \; \mathbb{E}_{\mathbf{x},\mathbf{c},\epsilon,t}\left[w_t \|\epsilon - \epsilon_z(\mathbf{x}_t, \mathbf{c}, t)\|_2^2\right] \quad \text{where } \epsilon_z = \text{decoder}(z)$$

where $\epsilon_z$ is the denoising UNet decoded from the latent vector $z$. The exact implementation of this optimization will be investigated empirically. For a more thorough overview of the inversion process and our implementation of the objective above, please see Appendix A.

## 5 Experiments

The original paper evaluates their methods based on a few experiments: (1) Qualitative observation of sampling from PCA-w2w Space (2) Quantitative Comparison with baseline in editing subjects (3) Quantitative Comparison with baseline in inverting subjects (4) Qualitative observation of Out-of-Distribution Projection.

In this project, we will focus on task (1) and (3). (1) Sampling is the main usage of this w2w space. Our motivation is that VAE is a better way to learn a low-dimensional w2w space than PCA. So we will assess the sample models qualitatively. (2) While facilitated by the low-dimensional w2w space, editing can also be performed in the original weight space given a set of model parameters. As such, we focus on exploring the functionalities that can only be achieved with a low-dimensional space. (3) Inverting an identity to the w2w space greatly depends on the expressiveness of this low-dimensional space, which we believe is the most critical metric for assessing how well the w2w space is formed. Therefore, we will compare our new method with a baseline in this task. (4) Inverting out-of-distribution identity is similar to normal inversion, so we skip it. We provide visualization results of our method and the baseline method in Appendix B.

## 5.1 Train and Validate VAE

We trained a vanilla implementation of a VAE with a straightforward architecture. The encoder and decoder are composed of alternating linear layers and LeakyReLU activations. Since the training dataset (the weight dataset described in Section 2) has a range of values between $-0.09$ and $0.09$, we normalized the dataset by scaling it by a factor of 10. To ensure the output of the decoder remains within the original range, we applied a Tanh activation function in the final output layer.

After extensive experimentation, we determined an optimal architecture where the encoder progressively reduces the input weight vector of 99,648 dimensions to [4096, 2048, 1024], and finally encodes it into a latent space of dimension 512. The decoder mirrors this structure, reversing the dimensionality reduction process.

Initially, we trained the VAE using the default combination of reconstruction loss and KL-divergence loss. However, we observed that the model tended to overfit to the mean of the data, likely due to the strong influence of the Gaussian prior. To address this, we adopted a training strategy inspired by [2], which includes a warm-up phase and a KL-scaling phase. During the warm-up phase, we temporarily disabled the KL-divergence loss for the first several epochs to allow the model to focus on accurate reconstruction. We then gradually increased the KL-divergence weight factor to improve the latent space's density. This approach stabilized the training process and prevented overfitting to the dataset's average weights.

Using this architecture and training strategy, the VAE requires approximately 20 minutes to train 12 epochs on a single Nvidia A6000 GPU. The training loss over time is shown in Fig. 1. In the final training step, the training loss and validation loss were [recon: 0.00221, KL: 0.03539] and [recon: 0.00221, KL: 0.03533], respectively.

To further demonstrate the quality of our trained VAE, we include additional visualizations of reconstruction and interpolation in the latent space in Appendix B. These results highlight the model's ability to effectively encode and decode identity-specific features while maintaining smooth transitions in the latent space.

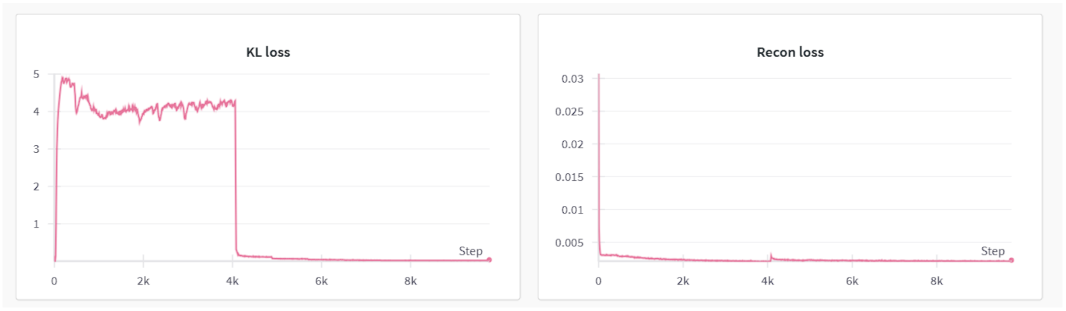

Figure 1: Training loss

## 5.2 Sampling

In the original PCA-w2w space, this is done by sampling the basis coefficients of the principal components according to pre-computed distributions for each basis component's coefficient (See 4.1). With our VAE-w2w space, we simply sample a latent from unit Gaussian distribution and decode it to obtain the weight. Sampling can be done in very few seconds since it just needs a forward inference of the decoder. The images generated with two sampled weights are shown in fig 2.

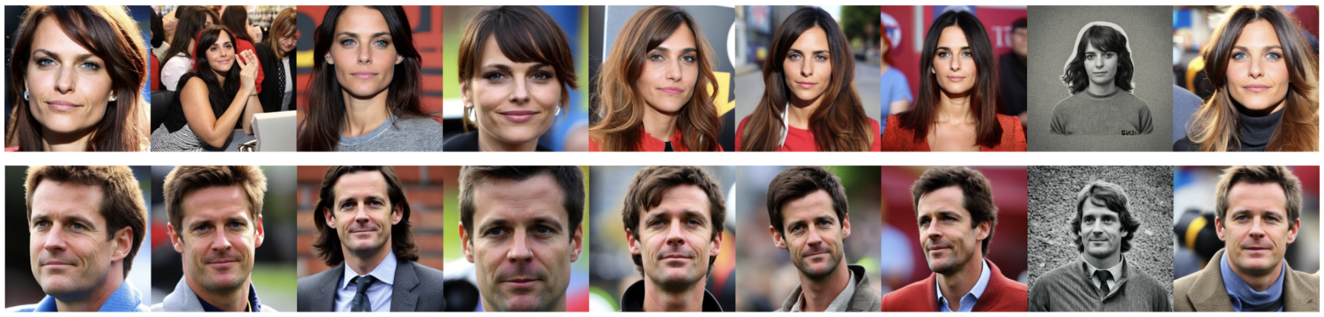

Figure 2: Images generated by two sampled weights (row 1 & 2) using our VAE. The generated subjects are consistent for a certain sampled weight.

### 5.3 Inversion

We compare our VAE-based inversion method with the inversion presented in [3] and discussed in 4.3. A key contribution of the original work was addressing the gap between multiple-image DB-LoRA finetuning and single-image finetuning by constraining the optimized parameters of inverted models to the PCA-w2w space. It usually takes 1.1 minutes to inversion a subject. For more implementation details, please see Appendix A. The following table displays the average ID score of each approach:

| Method | Single-Image | # Param | Opt. Time (s) | ID Score ↑ |
|---|---|---|---|---|
| PCA-w2w Inversion | ✓ | 10,000 | 67 | $0.56 \pm 0.03$ |
| VAE-w2w Inversion | ✓ | 512 | 71 | $-0.024 \pm 0.03$ |

Table 1: Quantative comparison of PCA-w2w and our VAE-w2w in inversion task

## 6 Code Overview

We've open-sourced our code in this repository[3]. Our code is mainly based on the public implementation of weights2weights [3]. Our training and generating code is included in /vae_w2w/. Specifically, we implement our VAE models including the vanilla version and the one with adversarial loss (see vanilla_vae.py and yuchen_vae.py). We provide a training script in train_vanilla_vae.py. The sampling code is in the evaluate_trained_model.ipynb. The inversion is implemented in inversion_vae.py, inversion_cmaes.py, and lora_VAEw2w.py under the main folder, and the inversion is run in /vae_w2w/inversion_scrap.ipynb. We also provide some other useful scripts for visualization and analysis in the same folder. The evaluation protocol, ID-score, is implemented in /idscore/.

## 7 Timeline

Table 2 shows the stages of our projects and percentage of time we spent at each stage. It is summarized from our emails and Whatsapp messages.

## 8 Research Log

The key challenge of this project was training a robust VAE model to effectively capture the structure of the weight space without overfitting to the data or collapsing the latent space.

Initially, we trained a vanilla VAE with the default loss function (reconstruction loss weighted equally with KL-divergence loss). The training process appeared promising: the loss curve was smooth, consistently decreasing and converging after just a few epochs (2–4). Sampling weights from the trained model yielded images of faces that, while not identical, bore strikingly similar features. This was an encouraging signal, suggesting that our approach had potential.

[3]https://github.com/fzmautner/weights2weightspp

| | | | |
|---|---|---|---|
| Stages | Literature review & topic selection | Make plan & Write proposal | Read codes and documents |
| Time | 8% | 10% | 7% |
| Stages | Implement ID score | Run baseline | Midway report |
| Time | 5% | 5% | 5% |
| Stages | Implement & train VAE | Implement & run VAE-based inversion | Final report |
| Time | 37% | 13% | 10% |

Table 2: Timeline of the project.

However, upon further sampling, we observed that the model consistently generated images of nearly identical faces, regardless of the latent vector $z$. The decoded weights seemed to converge toward the mean of all training weights. This raised a critical issue: the latent space was not being effectively utilized, and the model was overfitting to the average of the data. To address this, we hypothesized several potential causes, including the inadequacy of linear layers compared to convolutional layers commonly used in VAEs, the latent space being too small to capture meaningful variance, and insufficient dataset size. We ran extensive experiments to validate these hypotheses. Ultimately, we identified the root cause: the KL-divergence loss was dominating the optimization process. Analyzing the loss dynamics, we found that the reconstruction loss plateaued early in training, while the KL-divergence loss continued decreasing to a minuscule value—more than 100 times smaller than the reconstruction loss. This imbalance forced the decoder to output weights close to the mean of the training set, regardless of the latent vector.

Our first solution was to adjust the weighting of the KL-divergence loss. While tuning this hyperparameter, we encountered a difficult trade-off. A high KL weight led to the decoder averaging all weights, as described earlier. Conversely, a low KL weight allowed the model to achieve lower reconstruction losses on both training and testing data, but sampling from the latent space resulted in unstructured, meaningless outputs, indicating an inadequately regularized latent space. Moderate KL weights improved results a bit, generating weights that produced more consistent faces, but these faces still closely resembled the mean. At this stage, we hit a bottleneck.

To overcome this, we revisited the literature for alternative solutions and explored various strategies, including incorporating ResNet passthrough blocks and adversarial loss inspired by GANs. After extensive experimentation, we foudn the KL-loss weight scheduling strategy proposed by Bowman et al. [2], which offered a structured approach to balance reconstruction and latent space regularization. The adopted strategy includes two phases: a warm-up phase and a KL-scaling phase. During the warm-up phase, the model is trained exclusively with reconstruction loss for the initial epochs, allowing it to focus on accurately reconstructing the input without considering latent space regularization. In the KL-scaling phase, the weight of the KL-divergence loss is gradually increased over subsequent epochs, encouraging the model to learn a dense and meaningful latent space. This approach effectively addressed our earlier challenges.

However, a new challenge emerged when we transitioned to subject inversion. The task is in fact to fit the diffusion model to a target image under restriction of only optimizing the latent vector $z$. In our latest experiments, the loss curve during inversion did not decrease smoothly. Furthermore, the optimized diffusion model failed to generate faces resembling the input subject. One hypothesis is that the learned latent space might not fully capture the necessary identity-specific features, resulting in suboptimal representations. Another possibility is that the inversion process requires additional regularization to ensure $z$ remains within the learned latent distribution. As we approach the final stages of this project, resolving this inversion issue remains an open challenge, and we are actively investigating solutions to improve both the latent space and the inversion strategy.

## 9 Conclusion

In this project, we successfully investigate the possibility of utilizing generative models to generate generative models themselves, i.e. meta generative models. Specifically, we trained a VAE on a dataset of weights from thousands of fine-tuned diffusion models. Although our VAE model is not perfectly trained, our visualization results show that the VAE model can learn a latent space that captures the structure of the weight data, thus generating new models with the desired subject properties. Future works could consider extending the weight datasets to scale the model up or consider adopting more powerful generative models like diffusion models to generate weights.

## References

[1] Marcin Andrychowicz et al. *Learning to learn by gradient descent by gradient descent*. 2016. arXiv: 1606.04474 [cs.NE]. URL: https://arxiv.org/abs/1606.04474.

[2] Samuel R Bowman et al. "Generating sentences from a continuous space". In: *arXiv preprint arXiv:1511.06349* (2015).

[3] Amil Dravid et al. "Interpreting the Weight Space of Customized Diffusion Models". In: *arXiv preprint arXiv:2406.09413* (2024).

[4] Ian Goodfellow et al. "Generative adversarial nets". In: *Advances in neural information processing systems* 27 (2014).

[5] David Ha, Andrew Dai, and Quoc V. Le. *HyperNetworks*. 2016. arXiv: 1609.09106 [cs.LG]. URL: https://arxiv.org/abs/1609.09106.

[6] Jonathan Ho, Ajay Jain, and Pieter Abbeel. "Denoising diffusion probabilistic models". In: *Advances in neural information processing systems* 33 (2020), pp. 6840–6851.

[7] Edward J Hu et al. "LoRA: Low-Rank Adaptation of Large Language Models". In: *International Conference on Learning Representations*. 2022. URL: https://openreview.net/forum?id=nZeVKeeFYf9.

[8] Diederik P. Kingma and Max Welling. "Auto-Encoding Variational Bayes". In: *International Conference on Learning Representations*. 2014.

[9] Robin Rombach et al. "High-resolution image synthesis with latent diffusion models". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2022, pp. 10684–10695.

[10] Nataniel Ruiz et al. "Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2023, pp. 22500–22510.

[11] Florian Schroff, Dmitry Kalenichenko, and James Philbin. "Facenet: A unified embedding for face recognition and clustering". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 815–823.

[12] Jiaming Song, Chenlin Meng, and Stefano Ermon. "Denoising diffusion implicit models". In: *arXiv preprint arXiv:2010.02502* (2020).

[13] Kuan-Chieh Wang et al. "Moa: Mixture-of-attention for subject-context disentanglement in personalized image generation". In: *arXiv preprint arXiv:2404.11565* (2024).

[14] Kaipeng Zhang et al. "Joint face detection and alignment using multitask cascaded convolutional networks". In: *IEEE signal processing letters* 23.10 (2016), pp. 1499–1503.

## A Inversion

**Baseline: Inversion with PCA bases.** The goal of single-image inversion is to produce model parameters $\theta$ for a denoising UNet $\epsilon_\theta$ that minimize the denoising objective given an image $\mathbf{x}$ [6]:

$$\mathbb{E}_{\mathbf{x},\mathbf{c},\epsilon,t}\left[w_t\|\epsilon-\epsilon_\theta(\mathbf{x}_t,\mathbf{c},t)\|_2^2\right] \tag{1}$$

Therefore, a naïve way to implement model inversion is to simply minize this objective directly:

$$\theta=\underset{\theta}{\operatorname{argmin}}\ \mathbb{E}_{\mathbf{x},\mathbf{c},\epsilon,t}\left[w_t\|\epsilon-\epsilon_\theta(\mathbf{x}_t,\mathbf{c},t)\|_2^2\right] \tag{2}$$

However, doing this can result in model parameters $\theta$ that are out of distribution for the more general class of images we want to generate (in this case, faces), as the objective focuses solely on the given image $\mathbf{x}$. This motivates the DreamBooth [10] objective, which includes a prior term encouraging the network to be capable of generating images of a given class of images:

$$\mathbb{E}_{\mathbf{x},\mathbf{c},\epsilon,t}\left[w_t\|\epsilon-\epsilon_\theta(\mathbf{x}_t,\mathbf{c},t)\|_2^2+\lambda w_t'\|\epsilon'-\epsilon_\theta(\mathbf{x}_t',\mathbf{c}',t')\|_2^2\right] \tag{3}$$

Where $\mathbf{x}'$ represents images sampled from the broader class (e.g., generic faces) used for regularization, and the image class $\mathbf{c}'$ is used with the model to generate these class-regularization images ([3] Sec. 3.1).

The authors of the original w2w paper [3] use a different approach. In it, the objective function is exactly that of 1, but the parameters are enforced to be representative of the broader class of facial images by constraining them to the linear subspace of the set of diffusion models, the PCA-w2w space:

$$\min_{\theta}\ \mathbb{E}_{\mathbf{x},\mathbf{c},\epsilon,t}\left[w_t\|\epsilon-\epsilon_\theta(\mathbf{x}_t,\mathbf{c},t)\|_2^2\right]\quad \text{s.t. } \theta\in\text{PCA-w2w} \tag{4}$$

Doing so also allows for more efficient optimization, as the model is optimized not directly over $\theta$, but instead over it's latent representation of principal component coefficients $\beta_1,\dots,\beta_m$.

**Our approach: Inversion with VAE.** In a high level, we want to find parameters $\theta$ that minimize Equation. 1 while remaining in the distribution of models of that class. Furthermore, we want to do so in a low-dimensional latent space. A simple optimizing objective with respect to $z$ is

$$\min_{z}\ \mathbb{E}_{\mathbf{x},\mathbf{c},\epsilon,t}\left[w_t\|\epsilon-\epsilon_z(\mathbf{x}_t,\mathbf{c},t)\|_2^2\right]\quad \text{where } \epsilon_z=\text{decoder}(z) \tag{5}$$

In order to ensure that the inverted model remains in-distribution, we simply need to maximize it's latent probability, $p(z)$, while minimizing the base objective.

$$\min_{z}\ \mathbb{E}_{\mathbf{x},\mathbf{c},\epsilon,t}\left[w_t\|\epsilon-\epsilon_z(\mathbf{x}_t,\mathbf{c},t)\|_2^2\right]-p(z)\quad \text{where } \epsilon_z=\text{decoder}(z) \tag{6}$$

During training, we encourage the VAE to learn a dense and well-structured latent space by constraining the encoded distributions $q(z|x)$ to be close to our prior $p(z)=\mathcal{N}(0,\mathbf{I})$ using KL divergence. This same principle carries over naturally to the inversion task: instead of directly optimizing a single latent vector $z$, one could optimize the parameters $\mu$ and $\sigma$ of a distribution from which we sample $z$, maintaining consistency with the training objective by including the KL divergence term as a regularizer, ensuring the optimized distribution remains close to our chosen prior:

$$\min_{\mu,\sigma}\ \mathbb{E}_{\mathbf{x},\mathbf{c},\epsilon,t,z}\left[w_t\|\epsilon-\epsilon_z(\mathbf{x}_t,\mathbf{c},t)\|_2^2\right]+KL(\mathcal{N}(\mu,\sigma)\ ||\ \mathcal{N}(0,\mathbf{I}))\quad \text{where } \epsilon_z=\text{decoder}(z) \text{ and } z\sim\mathcal{N}(\mu,\sigma) \tag{7}$$

In other words, we optimize the standard denoising objective with a latent representation of the model which is encouraged to lie in the distribution learned by our VAE.

In practice, we opt for directly optimizing for $z$ in both gradient-based and evolutionary methods, as illustrated below. Unfortunately, we were unable to be successful in our inversion attempts, and we believe that this is due in part to the quality of our latent space. In the gradient-based setting, this is also related to the quality of our VAE decoder, mainly its ability to backpropagate sensible gradients from the UNet back to the latent vector $z$. In the evolutionary setting, we believe that the DDPM loss function over our latent space was not smooth and regular enough to allow for successful optimization, and local minima were often overexplored. Both implementations can be found in our code submission as `inversion_vae.py` and `inversion_cmaes.py`.

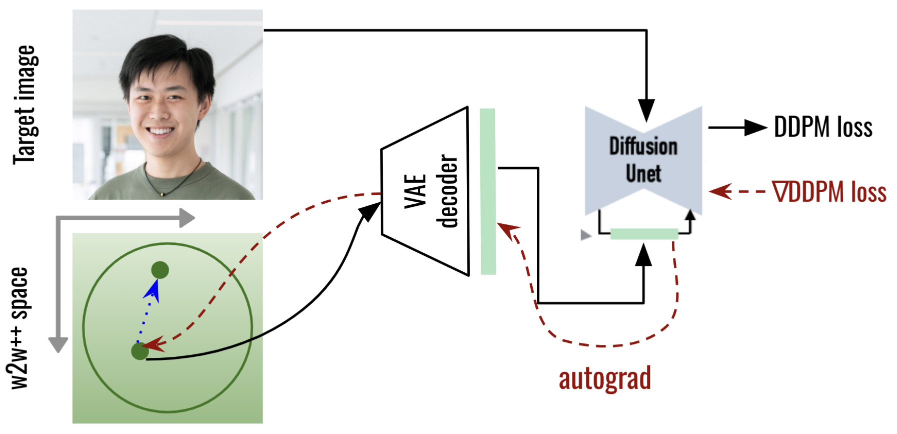

Figure 3: Gradient-based inversion pipeline

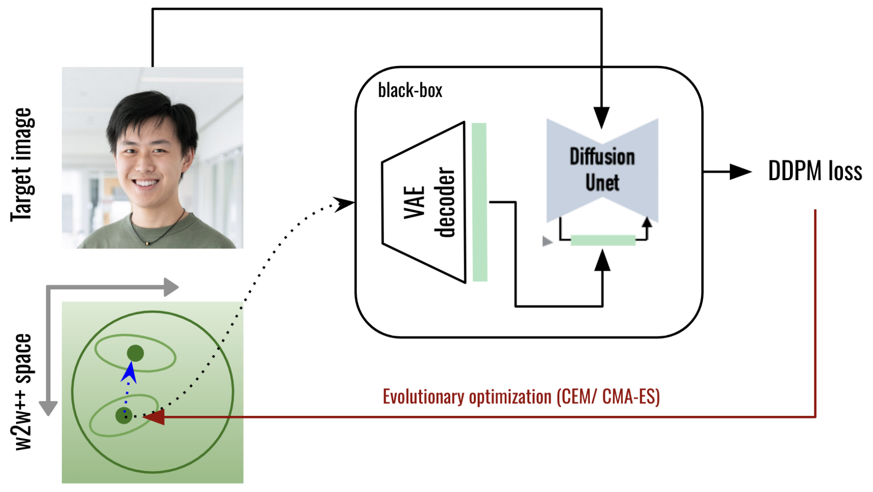

Figure 4: Evolutionary-based inversion pipeline

## B Visualization Results

### B.1 Ours

#### B.1.1 Reconstruction

Figure 5 shows the images generated from a ground truth weight and the VAE reconstructed weights. Although not perfect, our VAE can output a similar subject to that encoded in the ground truth weight.

#### B.1.2 Sampling

Figure 6 shows the results of randomly sampling a latent from a Gaussian distribution and decoding it to generate a new weight. The generated subjects from the decoded weight is consistent.

#### B.1.3 Interpolation

Figure 7 shows the results of interpolating two different latent vectors, decoding the interpolated vectors to weights, and using them to generate images with the same prompt and random seed. With VAE's expressive latent space, the generated subject can shift gradually from an individual to another, resulting in fancy visualization results.

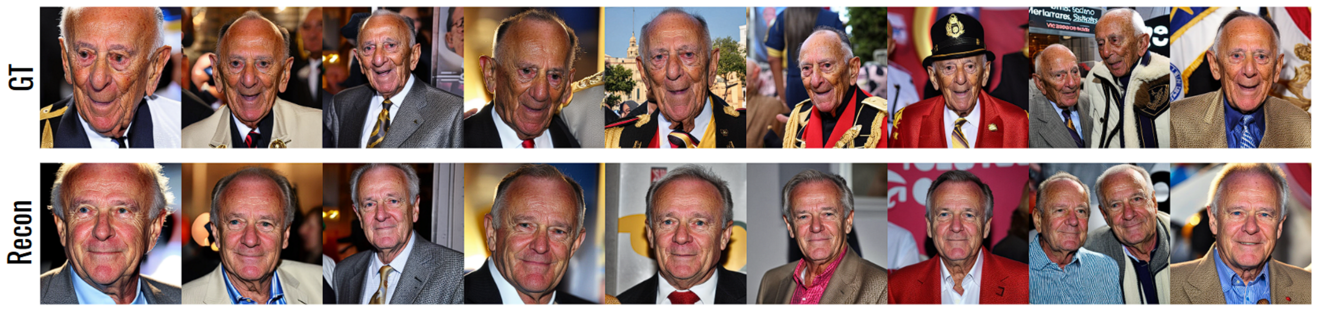

Figure 5: Images generated by the VAE-reconstructed weights and the original weights. Row 1: the ground truth weight. Row 2: the reconstructed weight.

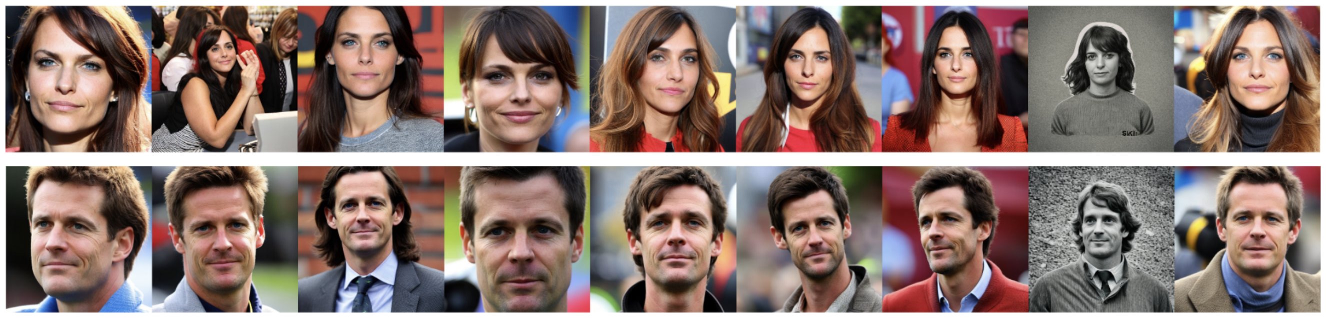

Figure 6: Images generated by two sampled weights (row 1 & 2) using our VAE. The generated subjects are consistent for a certain sampled weight.

#### B.1.4 Inversion

Figure 8 shows results of our inversion. While the identities are consistent, they do not reflect the target image.

### B.2 Baseline

#### B.2.1 Sampling

Figure B.2.1 shows the images generated by the diffusion models whose weights are sampled from PCA-w2w space.

#### B.2.2 Inversion

Figure B.2.2 and B.2.2 showcase images generated by the diffusion model attained through the inversion of each member of our team using the baseline PCA-w2w inversion approach.

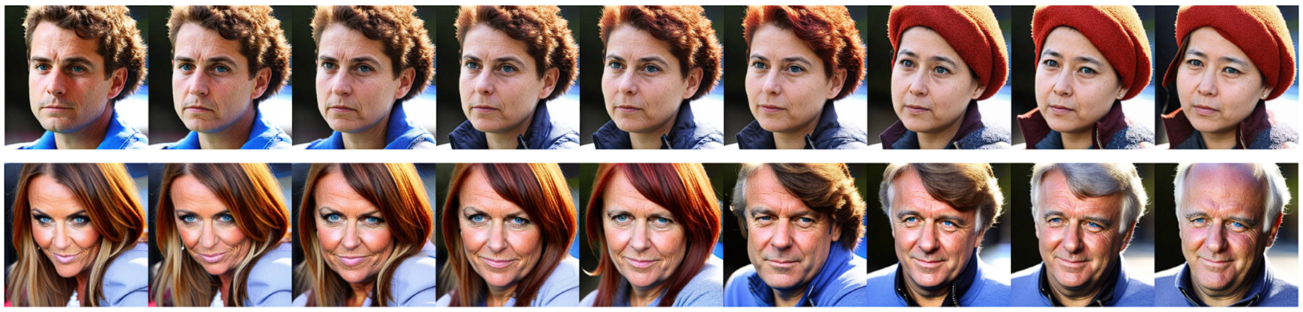

Figure 7: Images generated by different weights which are generated by interpolating two latent linearly in our VAE latent space. With random seed fixed, the generated subject can gradually shift from left to right.

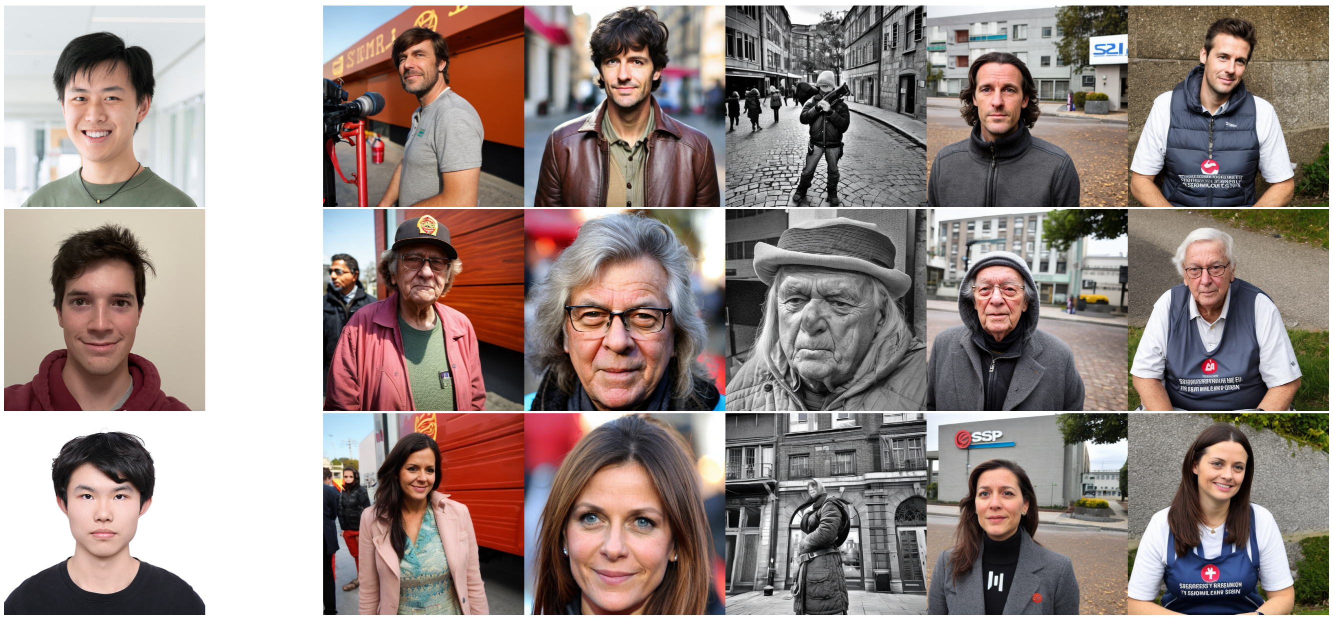

Figure 8: Inversion results for team members

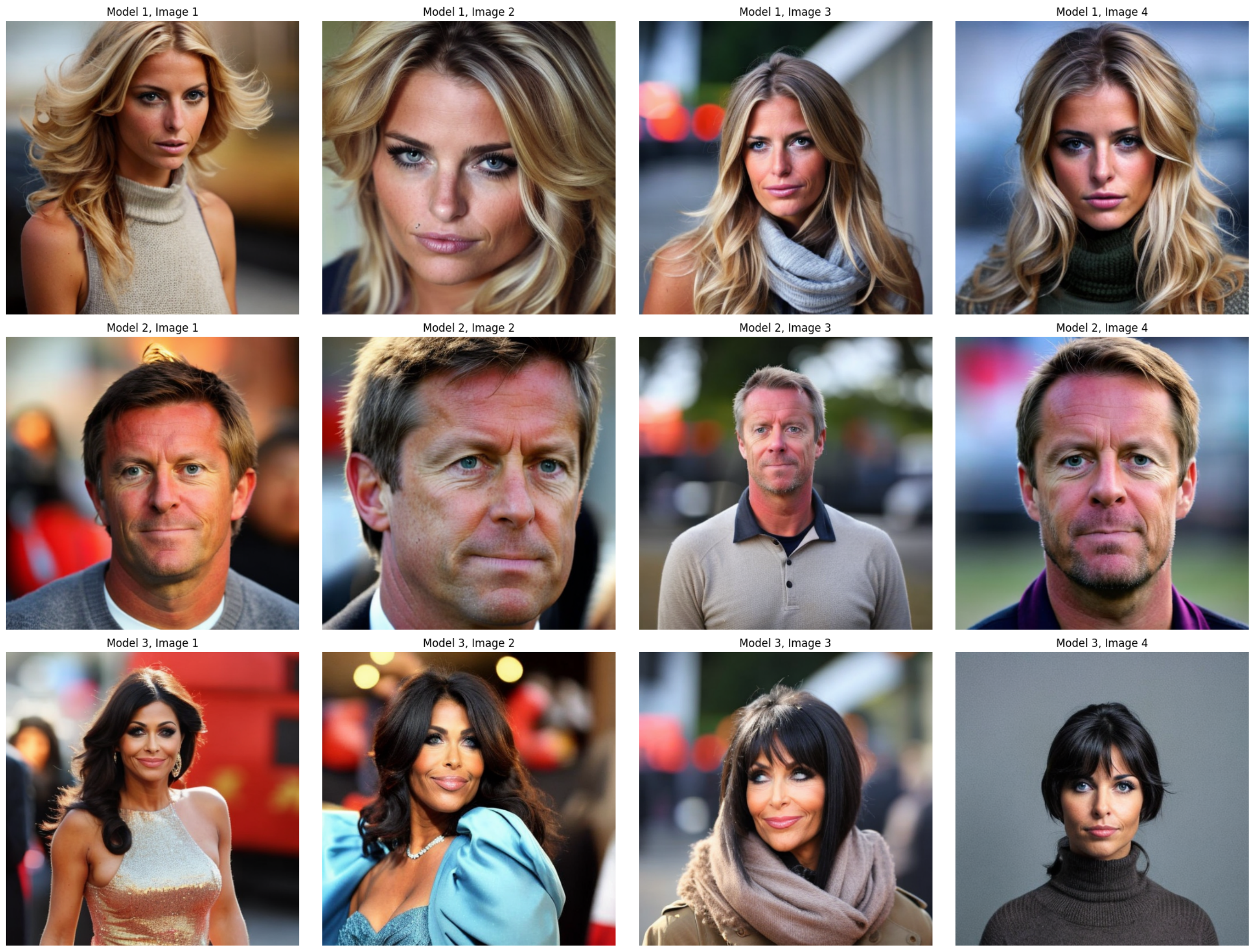

Figure 9: Images from diffusion models sampled from PCA-w2w space.

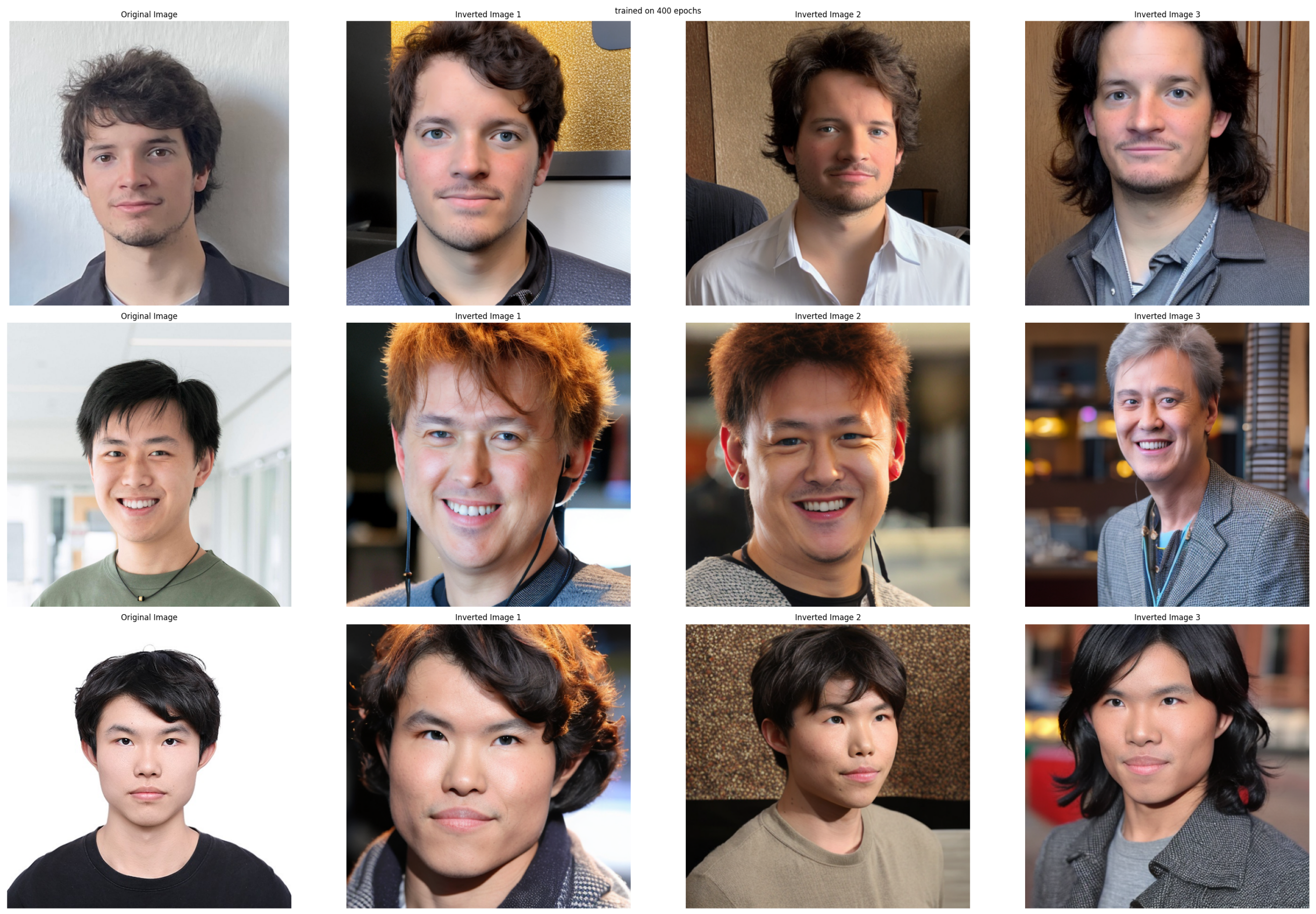

Figure 10: Images from inverted diffusion model constrained to baseline PCA-w2w approach. (400 epochs)

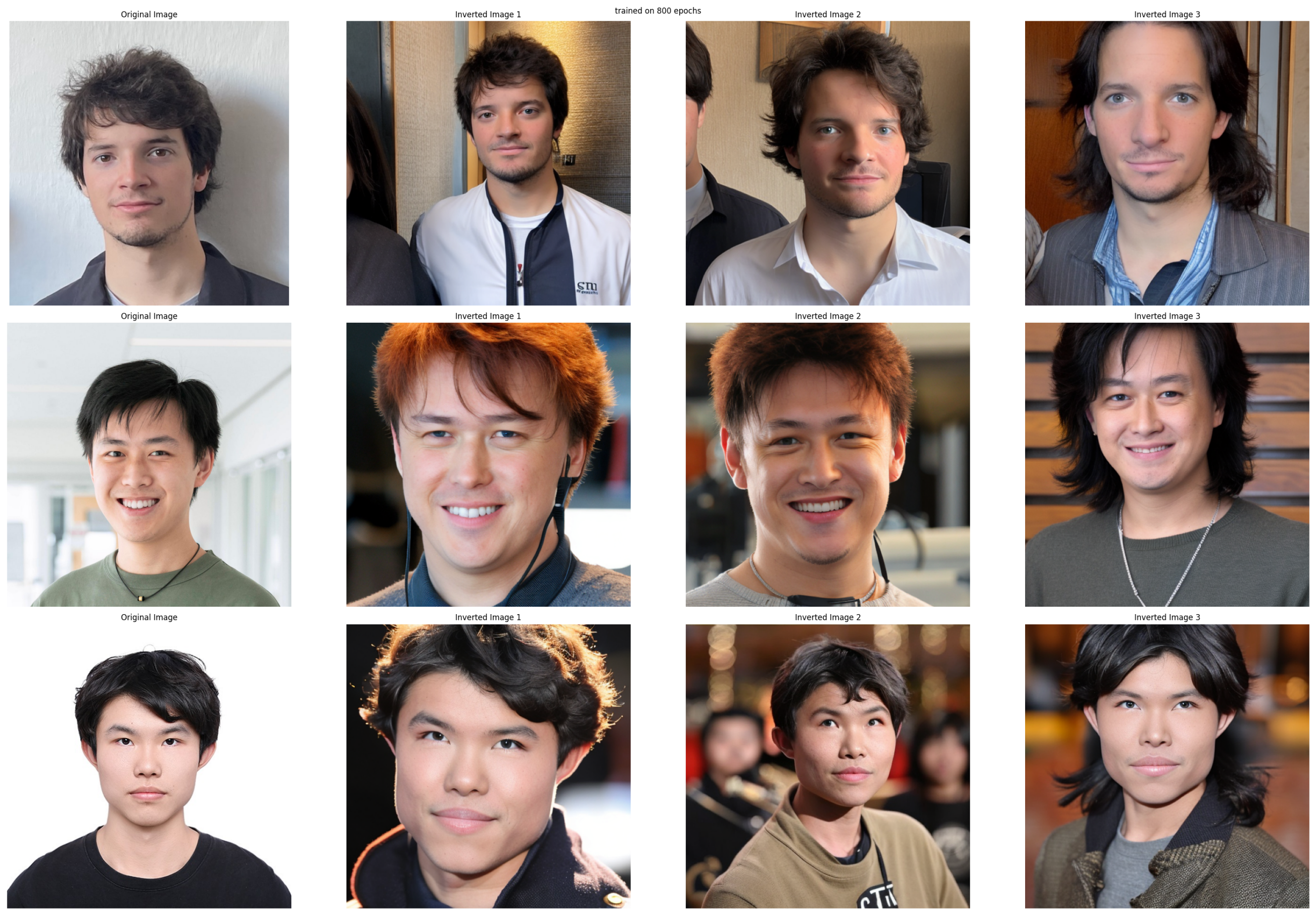

Figure 11: Images from inverted diffusion model constrained to baseline PCA-w2w approach. (800 epochs)