APPENDIX

A LOSS LANDSCAPE VISUALIZATION

To understand the underlying cause of the significant improvement in optimization brought by our mapping network θ, we visualize in Fig. 8 the loss landscape for performing optimization in the original input space X, and our projected space Z. To generate this visualization, we first perform 20 steps of optimization on the validation dataset to collect a set of recovered latents. We then perform principle component analysis (PCA) on these recovered latents to obtain two principle directions. Finally, for individual examples, we evaluate the loss for vertices on a meshgrid spanned by the two principle directions where the center is the last step of optimization.

From the visualization, we can see that the loss landscapes of the baseline are highly non-convex and contain points whose loss are significantly higher than its neighboring regions, while our loss landscapes are significantly smoother, with the "spikes" removed. Besides, our loss landscapes also tend to be steeper than the baseline ones. These two phenomena directly cause our method to perform gradient descent faster and stabler.

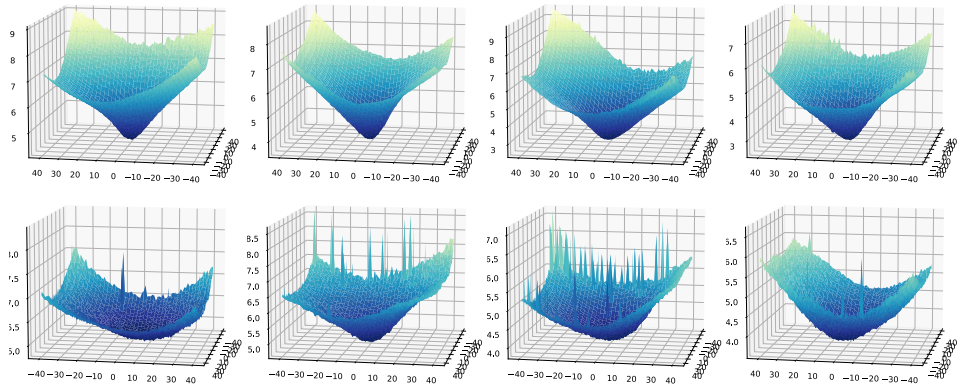

Figure 8: **Visualizing Loss Landscape (Uncurated).** Visualizing the loss landscape of StyleGAN inversion spanned by two principle directions. **Top** row shows 4 examples of the loss landscapes corresponding to our space Z. **Bottom** row shows the loss landscapes corresponding to the original input space X for the same 4 examples. Note that the top row and the bottom row have different minimum loss values on the landscape because the minimas are obtained by performing independent optimization runs in space Z and X respectively. Given a fixed number of optimization step, optimization in Z reaches lower loss values than optimization in X.

B EQUIVALENCE TO EXPECTATION-MAXIMIZATION

We show that our algorithm in 1 is equivalent to hard Expectation-Maximization (McAllester). First we rewrite the objective Eq. 2 as:

$$\hat{\theta} = \underset{\theta}{\operatorname{argmax}} \max_{z_i} \prod_i P_\theta(y_i | F(\Theta(\mathbf{z}_i))) \tag{8}$$

To optimize objective Eq. 8, we perform the following algorithm:

1. Initialize θ with Gaussian distribution
2. Repeat the following until $\prod_i \ln P_\theta(y_i | F(\Theta(\mathbf{z}_i)))$ converges:
   (a) *Expectation*: $\rho_\theta(z_i) = \delta(z_i = \tilde{z}_i)$, where $\tilde{z}_i = \operatorname{argmax}_{z_i} P_\theta(y_i | F(\Theta(\mathbf{z}_i))))$
   (b) *Maximization*: $\hat{\theta} = \operatorname{argmax}_\theta \mathbb{E}_{z\sim\rho}[\ln(P_\theta(y_i | F(\Theta(\mathbf{z}_i))))]$

where θ denotes the parameters of the mapping network Θ, and $\delta(\cdot)$ denotes a Dirac delta distribution. Since both expectation and maximization steps strictly increase or maintain the value of $P_\theta(y_i | F(\Theta(\mathbf{z}_i)))$ under gradient ascent, and the function is bounded, the algorithm is guaranteed to converge.

C DIMENSION OF Z

As our method learns a mapping network Θ that maps a vector from a new latent space Z to a vector in the original input space X. The dimension of Z becomes a hyperparameter. In table 3, we perform ablation studies on the effect of the dimension of Z space on the performance of the learned mapping network. From table 3, we can see a consistent improvement on performance as we increase the number of dimension of Z. However, such improvement hits a diminishing return when the dimension is in the same order of magnitudes as the dimension of X.

Dimension of Z	16	32	64	128	256	512	2048	Baseline
In-distribution	3.157	2.797	2.489	2.489	2.212	1.964	1.920	2.569
OOD	4.872	4.592	4.277	3.916	3.618	3.498	3.392	4.617

Table 3: Ablation studies on the dimension of Z. We trained variations of our full model with different number of dimension of Z space, varying from 16 to 2048 given the dimension of input space X is 512. Numbers correspond to loss values defined in Eq. 5.

D MORE EVALUATION ON OUT-OF-DISTRIBUTION GENERALIZATION

Since our proposed mapping network Θ is parameterized by a neural network, there's no guarantee that the learned mapping function is surjective. Therefore, we empirically study the generalization performance by testing a mapping network trained on CelebA-HQ against a spectrum of datasets from very similar ones (in-distribution) to completely different ones (OOD).

D.1 SYNTHETIC SPECTRUM

We first created a synthetic version of this spectrum of datasets by varying the level of Gaussian noise injected into the images of CelebA-HQ. With a higher level of Gaussian noise injected into the original images, more of the original image content is corrupted, creating a distribution of images further away from the original test images.

level of Corruption	$\mathcal{N}(0, 0.0^2)$	$\mathcal{N}(0, 0.1^2)$	$\mathcal{N}(0, 0.2^2)$	$\mathcal{N}(0, 0.3^2)$	$\mathcal{N}(0, 0.4^2)$
Improvement	29.53%	18.95%	12.57%	6.30%	6.72%

Table 4: Evaluation of a mapping network trained with Celeba-HQ trainset and tested on the Celeba-HQ testset corrupted with different levels of Gaussian noise. The improvement is calculated by the percentage improvement of loss defined by Eq. 5 from baseline (no mapping network) to ours (with mapping network) after 200 steps of optimization.

D.2 NATURAL IMAGE SPECTRUM

We then created a natural image version of this spectrum of datasets containing: CelebA-HQ (original in-distribution testset), AFHQ-Cat (center-aligned cat faces), LSUN-Cat (unaligned cat images), Container-Ship ("Container Ship" class from ImageNet), and Gaussian noise with individual pixel sampled from $\mathcal{N}(0.5, 0.5^2)$. From left to right, images in the datasets vary from very similar to human faces to very different.

From results evaluated on both synthetic spectrum and real spectrum, we observed consistent improvements of our method over the baseline, which shows the generalization performance of our method when applied to OOD data. From table 4 and 5, we see the improvements drop as the evaluation data is less and less in-distribution with the training data, which indicates that the mapping network does learn a prior from the training data.

dataset	CelebA-HQ	AFHQ-Cat	LSUN-Cat	Container-Ship	$\mathcal{N}(0.5, 0.5^2)$
Improvement	29.53%	35.99%	24.36%	16.87%	6.60%

Table 5: Evaluation of a mapping network trained with Celeba-HQ trainset and tested on a spectrum of datasets from the original CelebA-HQ (in-distribution) to Gaussian noise (OOD). AFHQ-Cat Choi et al. (2020) is a dataset with aligned cat faces. LSUN-Cat Yu et al. (2015) is a dataset with unaligned cat images. Container-Ship is 200 images sampled from "Container Ship" class of ImageNet Deng et al. (2009). The improvement is calculated the same way as table 4

E Dependence on Training Dataset

From previous section, we know that the learned mapping network contains priors learned from the training data. To exclude such influence, we train our mapping network using randomly generated images made of Gaussian noise and evaluate on testset of CelebA-HQ. From table 6, we see that even trained with the task of reconstructing images of Gaussian noise, our mapping network still bring some improvement over the baseline model, though much less significant. Interestingly, this improvement number is consistent with the last columnes of table 4 and 5, where a mapping network is trained with CelebA-HQ and tested on images with Gaussian noise.

	Baseline	Ours (trained on Gaussian noise)	Improvement
CelebA-HQ	2.569	2.396	6.72%
LSUN-Cat	4.617	3.717	19.49%

Table 6: Evaluation of a mapping network trained with images of Gaussian noise sampled from $\mathcal{N}(0.5, 0.5^2)$ clipped to [0, 1]. Numbers show the average loss defined in Eq. 5 evaluated on testset of CelebA-HQ and LSUN-Cat.

F Limitations

Optimization-based inference has intrinsic advantages to robustness, accuracy, and flexibility, which comes at the cost of additional computation time during inference. Encoder-based methods will usually perform faster because they only require a single forward pass of a neural network, while our approach requires several computational passes in both the forward and backward (gradient) direction. We believe that for many applications this trade-off will be desirable, especially in cases where accuracy is more important than speed. Our approach aims to minimize this additional computational overhead brought by optimization-based inference, and our experiments on multiple datasets show the significant computational savings compared to other optimization-based inference methods.

Unlike many other optimization-based inference algorithms, our approach also requires a training step in order to fit a suitable landscape, which requires both training time and training data. However, we believe this overhead is insignificant for most applications and we have designed our neural networks to be efficient. For example, Θ is relatively lightweight, making its training time fairly marginal compared to the training of the forward model F. In all our experiments, we found that the training time of Θ is orders of magnitudes faster than the training time for F.

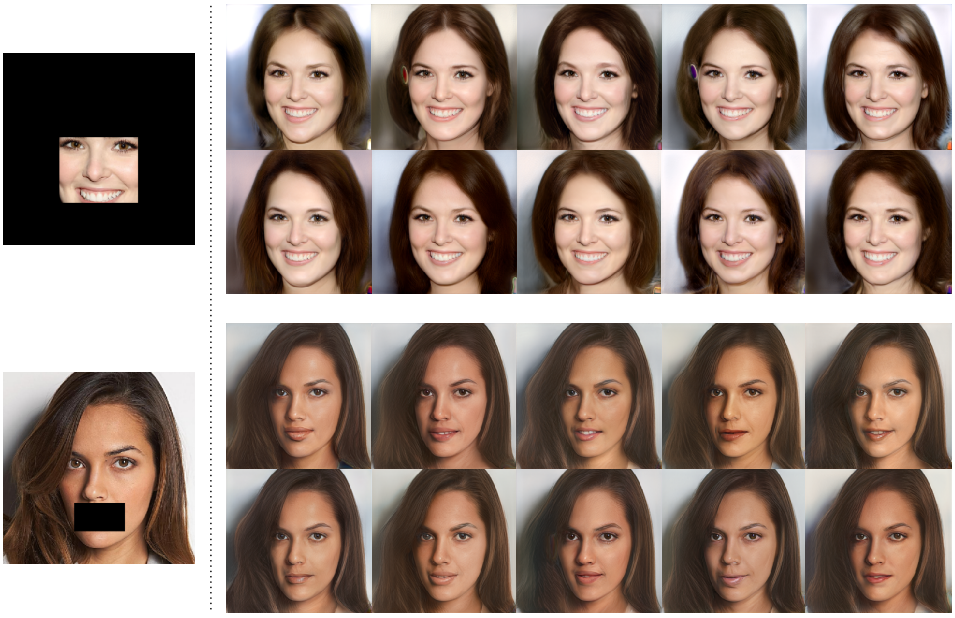

Figure 9: **Diversity of Masked Reconstructions.(Uncurated)** We visualize reconstructions for partially observable inputs from random initialization. The masked regions are not considered for loss computation, i.e., the gradient is set to be zero. By optimizing only on the partial observation, we obtain diverse, feasible solutions for the hidden regions. All reconstruction results presented are randomly sampled.