Semantic Segmentation with Generative Models: Semi-Supervised Learning and Strong Out-of-Domain Generalization
-
Supplemental Material

Daiqing Li[1*] Junlin Yang[1,3] Karsten Kreis[1] Antonio Torralba[4] Sanja Fidler[1,2,5]

[1] NVIDIA [2] University of Toronto [3] University of Yale [4] MIT [5] Vector Institute

1. Qualitative Analysis of Inference and Test-Time Optimization

In Figure 1, we present qualitative examples of intermediate image reconstructions and label predictions during test-time optimization for embedding inference (see Sec. 3.5 of main paper). At step 0, we show the reconstructed image and pixel-wise label map using the latent code ($w^+ \in \mathcal{W}^+$) predicted by the encoder. Looking at the in-domain test image (first row), we can see that the reconstructed image at step 0 recovers the overall structure of the test image including the overall hair style, the positions of the eyes, nose and mouth. However, some smaller parts, for example the eye brow and ears, do not match the ground truth very well. Such discrepancies at step 0 are especially pronounced for out-of-domain test images (third and fifth row), as the encoder is trained only with in-domain images, *i.e.* the real human faces from CelebA. It has never seen images like paintings and sculptures during training and therefore tries to map those out-of-domain images into the latent space of the in-domain images with similar semantic structure. For example, the encoder-based reconstruction at step 0 of the sculpture still resembles a real human face, but the positions of the parts almost match the ground truth except for the ear, which would be unusual in a real human face. Since the encoder is trained with in-domain images, it cannot precisely recover individual out-of-domain test images. Instead, it predicts common and smooth features learnt from the training dataset with only the overall structure being consistent with the out-of-domain test images.

This is where test-time optimization comes in. As just discussed, the encoder prediction provides a strong latent code initialization which already captures the overall semantic structure of the original image. The goal of the optimization is to take the initial latent code predicted by the encoder and further finetune it to also match the fine details missed by the encoder prediction. For the in-domain image, we can see that from step 0 to step 200 the reconstructed image develops a higher similarity to the original test image, especially with respect to the fine details of the hair, eyes, eye brows and ears. For the painting and the sculpture, the texture changes dramatically and also the semantic structure—for example, the mouth in the painting and the ears in the sculpture—is modified to match the test image as well as the ground truth labels. Note that we only optimize for the reconstruction quality of the image, as we do not have access to the ground truth label at test time. Nevertheless, we still observe strong consistency between the reconstructed image and the generated label. We attribute this to the general consistency between images and labels that is enforced during generator training by the discriminator D_m.

1.1. Ablation Study on Test-Time Optimization Parameters

We conduct an ablation study on two important hyperparameters for the test-time optimization, the number of steps performed as well as the λ_2 parameter, which determines the trade-off between image reconstruction quality and the "encoder-generator" regularization that encourages the optimization trajectory to stay within the region where the encoder approximately inverts the generator (ablation results in Figure 2, left). We see that compared to step 0 (no optimization), a small number of optimization steps helps boosting the performance for all λ_2 configurations (by around 5 percent after 50 steps, for example). We also observe that the performance peaks at around 400 optimization steps. Further optimization decreases the segmentation performance. Usually, a higher number of optimization steps means higher image reconstruction quality. But

[*]Correspondence to {daiqingl,sfidler}@nvidia.com

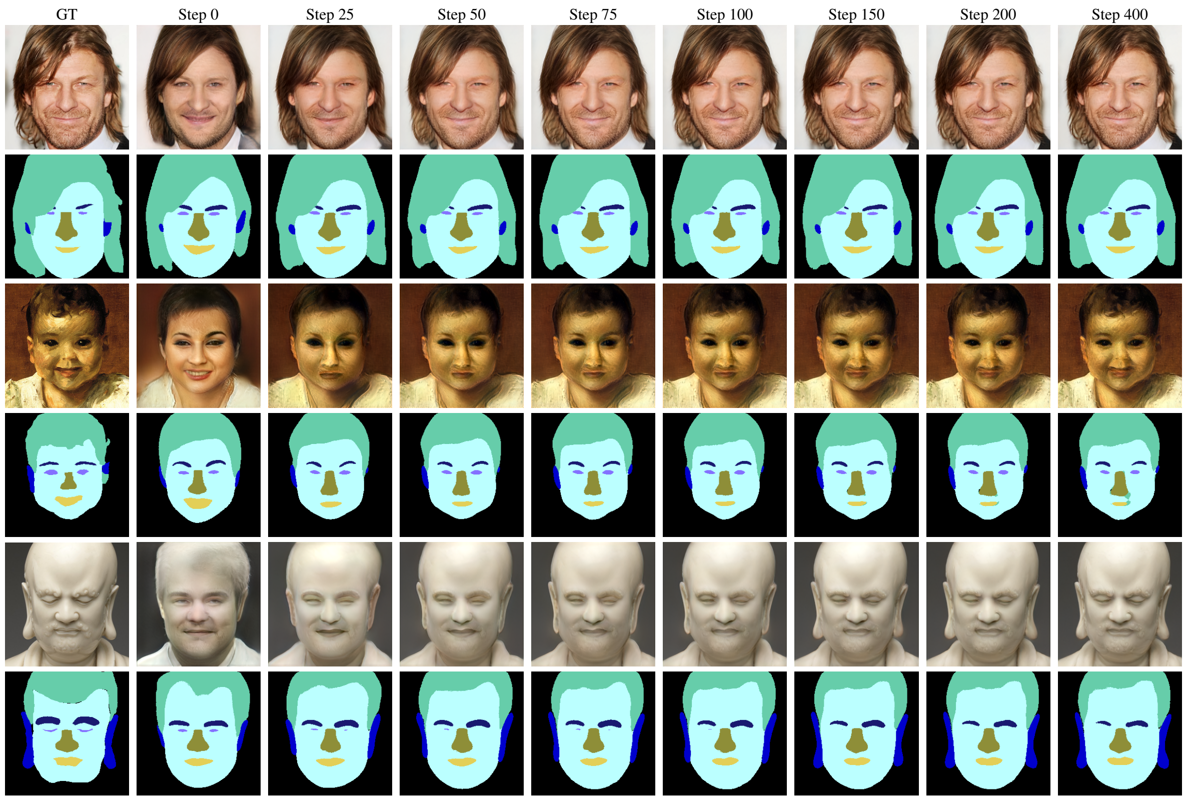

Figure 1: **Face Parts Segmentation Optimization Results.** Image reconstructions and segmentation label predictions at different steps during the optimization process. Step 0 corresponds to using the latent code predicted by the encoder without any further optimization. The model was trained on CelebA-Mask data. Hence, the first example corresponds to in-domain data, while the other two examples, from the MetFace dataset, are out-of-domain cases.

after a certain number of steps, around 400 in our case, the optimization tries to recover details such as tiny wrinkles in the face and the optimization trajectory may propagate too far outside the meaningful region of latent space where the training images reside. The generator was not trained to deal with such far-from-training latent codes and hence the semantic label prediction becomes inaccurate.

We also plot the reconstruction quality measured in LPIPS vs. segmentation performance (Figure 2, right). We find that the reconstruction quality is generally correlated with segmentation quality. However, when image reconstruction loss (LPIPS) approaches 0—the y-axis in the plot—there seems to be a small dip in segmentation performance, most likely corresponding to the “overfitting” effect discussed in the previous paragraph.

1.2. Probabilistic Perspective on the Inference Protocol

It is interesting to note that we can also look at our inference protocol from a fully probabilistic perspective: Given a target image x^*, we aim to find the maximum of a log posterior distribution $\arg\max_{w^+\in\mathcal{W}^+} \log p(w^+|x^*)$ and via Bayes Theorem $\log p(w^+|x^*) \sim \log p(x^*|w^+) + \log p(w^+)$. Since the LPIPS loss is based on an L2 term in the feature space of a neural network, we can interpret the reconstruction loss in Eq. (8) of the main paper as the negative log probability of an image x under a product of Normal distributions that are defined in pixel as well as feature space and centered at the target image x^*. Hence, by using LPIPS and per-pixel L2 reconstruction losses, we are effectively assuming $p(x^*|w^+)$ to be a product of such Normal distributions. Furthermore, as discussed in Sec. 3.5 in the main paper, the “prior” $p(w^+)$ is not a tractable distribution. Therefore, in our protocol we are essentially ignoring this term and instead regularize the objective with the encoder-generator regularization term in Eq. (7) of the main paper.

This probabilistic perspective is interesting, because it connects our work to other generative models and also suggests directions for future research. For example, one could employ Markov chain Monte Carlo (MCMC) techniques to sample instead of maximize the posterior distribution $p(w^+|x^*)$ and generate diverse plausible pixel-wise label masks. Such ideas have been explored in the context of GAN-based inpainting [4] and this also highlights our connection to the literature on

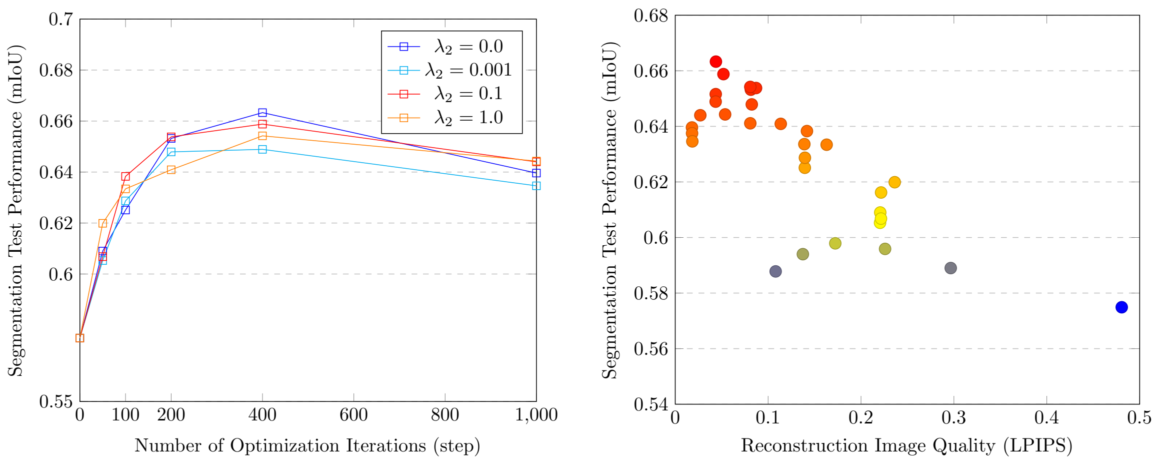

Figure 2: **Optimization Parameter Ablation Study.** *Left:* Ablation study on the number of optimization steps performed during test-time optimization for embedding inference, as well as on the λ_2 hyperparameter. *Right:* Image reconstruction quality measured in LPIPS against segmentation performance. Segmentation performance is calculated as mIoU on the out-of-domain MetFace dataset. The model for these experiments is trained with 1500 CelebA-Mask labels.

energy-based modeling for perception, recognition and generation, where one often performs MCMC-based sampling during training and synthesis [23, 6, 29].

2. Training Details

As described in Sec. 3.4 of the main paper, we divide training of our model into two stages. In the first stage, we train the generator that models the joint distribution of both images and labels using a standard Generative Adversarial Network-based (GAN) approach. In the second stage, we freeze the generator and only train the encoder. For all tasks, we resize the original images and labels into a resolution of 256×256, both during training and inference. In the following, we describe the training parameter and details.

GAN Training. Our implementation is based on a pytorch implementation of StyleGAN2.[1] We follow the same training setup as in [11] and use a latent space $\mathcal{W}$ with dimension 512, leaky ReLU activation functions with $\alpha = 0.2$, bilinear filtering in all up/downsampling layers, exponential moving average of generator weights, and style mixing regularization. For D_r we use a non-saturating logistic loss with R_1 regularization [21]. For D_m, we use a hinge loss [17], feature matching loss [34], and multi-scale discriminators with 3 scales [34]. We use the Adam optimizer with hyperparameters $\beta_1 = 0, \beta_2 = 0.99, \epsilon = 10^{-8}$, learning rate $= 0.002$, batchsize $= 32$. For unlabeled images, we only apply random horizontal flip to the input image and for both labeled, we apply additionally apply random affine transformations with scale factor of 0.1, transition factor of 0.15 and rotation of up to 15 degree. We train the generator until FID converges, which happens usually between 60k and 100k iterations depending on the dataset.

Encoder Training. Our encoder is based on the Feature Pyramid Network (FPN) [18]. We use a feature dimension of 512 for lateral layers. Following the original implementation of the FPN, we extract features $\{P_2, P_3, P_4\}$ corresponding to fine to coarse feature levels. For each style code w_i^+, where i indicates the number of the style layer, we use a small fully-convolutional network to map FPN's intermediate features to 512-dimensional vectors. With an input resolution of 256×256, we have 14 style layers in total. We define $w_i^+, i \in [0, ..., 2]$ as coarse-level style codes, obtained using features from FPN's P_4 layer. We define $w_i^+, i \in [3, ..., 6]$ as medium-level style codes, obtained using features from the P_3 layer. Finally, we define $w_i^+, i \in [7, ..., 13]$ as fine-level style codes, obtained using features from FPN's P_2 layer. For training this encoder, we follow [25] and use the Ranger optimizer, a combination of Rectified Adam [19] with the Lookahead technique [36]. We use a constant learning rate of 0.001 for the first 30k iterations and decrease the learning rate with a cosine decay schedule. Since the gradient needs to backpropagate through the generator, which has many layers, we find that adding gradient clipping with a max. norm of 2.0 helps stabilizing training. We use the same $\lambda_1 = 0.1$ for all tasks. We use a batch size of 16 and train for 60k iterations.

[1]https://github.com/rosinality/stylegan2-pytorch

Dataset	Train	Test
JSRT/SCR [28, 33]	175	72
CXR14 (unlabeled) [35]	108K	-
NLM(MC) [9]	-	138
NLM(Shenzhen) [9, 30]	-	566
NIH [31]	-	100

Table 1: **Chest X-ray Lung Segmentation Datasets.** Number of training and test images taken from the different datasets.

3. Inference Details

For test-time optimization, we use the Adam optimizer with $\beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 1e^{-8}$ and learning rate $= 0.1$. For all tasks, we use $\lambda_2 = 0.1$. For chest x-ray segmentation, we use $\lambda_3 = 0.001$ with 200 steps. For skin lesion segmentation, we use $\lambda_3 = 0.1$ with 200 steps. For the CT-MRI liver segmentation task, we use $\lambda_3 = 0.001$ for CT and $\lambda_3 = 1e^{-5}$ for MRI with 400 steps. For face parts segmentation, we use $\lambda_3 = 0.001$ and 400 steps. For all tasks, we use a threshold of 0.5 to obtain the segmentation mask from the segmentation branch's continuous output in $[0, 1]$.

Dataset	Train	Test
ISIC2018 [3]	2000	594
ISIC20204 (unlabeled) [27]	33126	-
PH2 [20]	-	200
DermIS [5]	-	69
DermQuest [5]	-	98

Table 2: **Skin Lesion Segmentation Datasets.** Number of training and test images taken from the different datasets.

4. Dataset Details

For the chest x-ray segmentation task, we use CXR14 [35] as unlabeled dataset and JSRT [28, 33] as labeled dataset. We combine these two datasets and split into a part for training, consisting of both labeled and unlabeled x-rays, as well as a fully labeled part for in-domain testing. Furthermore, we use three additional datasets, NLM(MC) [9], NLM(Shenzhen) [9, 30] and NIH [31], for out-of-domain testing only (see Table 1 for splits). JSRT is a small collection of high quality chest x-ray datasets with balanced gender and marker-removal preprocessing. The patients are in standard pose. The other three datasets are collected from different medical centers with varying patient poses and scales, among which NIH varies the most in terms of sensor quality and patient poses. We follow [22] using histogram equalization and gamma correction as preprocessing steps. The same preprocessing is applied for all baseline experiments.

For skin lesion segmentation, we use ISIC2020 [27] as unlabeled dataset and ISIC2018 [3] as labeled dataset. We similarly split the combined data into a part for training and a separate labeled part for in-domain testing. We use three further datasets, PH2 [20], DermIS [5] and DermQuest [5], as out-of-domain test datasets (see Table 2 for splits). The lesion segmentation ground truth of ISIC2018 is collected using a combination of fully-automated, semi-automated, and manual annotation methods. Hence, it contains coarse annotations as also reported by other papers [24]. We did not filter or preprocess the original datasets.

For the cross-domain CT-MRI liver segmentation task, we use LITS2017-test [1] with 70 CT volumes as unlabeled dataset and LITS2017-train [1] with 131 volumes in total as labeled dataset. We further split LITS2017-train into train and test sets with 118 and 13 volumes respectively as labeled datasets for training and in-domain testing. We use the other two MRI datasets CHAOS-T1-in [12], and CHAOS-T1-out [12] as out-of-domain test datasets (see Table 3 for details). T1-in denotes T1-weighted MRI for "in"-phase mode and T1-out denotes T1-weighted MRI for "out"-phase mode. LITS2017 is a collection of liver CTs with liver and tumor segmentations. We only use the liver class and merge the tumor class into the background class for the whole dataset. As preprocessing of the raw CT data, we clip the attenuation coefficient in a range between -200 and 400 and normalize it by subtracting the minimum and dividing by the signal range in a slice-wise manner. The CHAOS

Dataset	Train	Test
LITS2017-train [1]	118	13
LITS2017-test4 (unlabeled) [1]	70	-
CHAOS-T1-in [12, 13, 14]	-	20
CHAOS-T1-out [12, 13, 14]	-	20

Table 3: **CT-MRI Liver Segmentation Datasets.** Number of training and test patient volumes taken from the different datasets.

dataset contains multi-organ MRIs with different sequences. We only use the liver segmentation class and merge other classes into the background class. As preprocessing, we clip the raw MR signal in a range between the 0.1 and 99.9 percentiles and normalize it by subtracting the minimum and dividing by the signal range in a slice-wise manner. Note that the CT and MRI volumes have different sensor poses and might have different regions of interest. We did not do any cropping, resampling, or alignment, but only the slice-wise preprocessing described above.

For face parts segmentation, we split the CelebA-Mask dataset [16] into CelebA-mask-u with 28k images as unlabeled dataset and CelebA-mask-l with 2k images and segmentation labels. We further devide the labeled portion of the data into train and test sets. For out-of-domain testing, we manually annotate 40 images randomly selected from the MetFace dataset [10], which is a collection of human face paintings and sculptures (see Table 4). We merge the 19 label classes originally defined in the CelebA-Mask dataset into 8 classes (background, ear, eye, eyebrow, skin, hair, mouth, nose) and we follow the same annotation protocol when labelling the selected images from the MetFace dataset.

Dataset	Train	Test
CelebA-mask-l [16]	1500	500
CelebA-mask-u4 (unlabeled) [16]	28000	-
MetFace-40 [10]	-	40

Table 4: **Face Parts Segmentation Datasets.** Number of training and test images taken from the different datasets.

5. Baseline Implementations and Metrics Details

As described in the main paper, our baseline methods include the fully supervised U-Net [26] and DeepLabV2 [2] as well as the semi-supervised segmentation methods MT [32], AdvSSL [7] and GCT [15]. The implementations of the semi-supervised segmentation approaches are based on the PixelSSL[2] repository [15] and we also use the DeepLabV2 implementation from this repository. The U-Net implementation is based on a public pytorch implementation.[3] The hyperparameters for MT, advSSL and GCT are kept at their defaults.

For all baseline methods, we use the Adam optimizer with a learning rate of 0.00025 and a batch size of 16. For the semi-supervised methods, the batch size for unlabeled data is set to 8. We train all baseline models for 10k iterations and choose the best model based on validation set performance.

For the evaluation metrics, we follow the common practice in the literature. For chest x-ray lung segmentation and CT-MRI transfer liver segmentation, we report the Dice score per patient, with each patient's Dice calculated as $DSC = \frac{2|A \cap B|}{|A|+|B|}$, where A is the prediction and B is the ground truth. For skin lesion segmentation, we report the Jaccard index per patient. The Jaccard index is calculated as $JC = \frac{|A \cap B|}{|A \cup B|}$ for each patient. For face parts segmentation, we report mean Intersection over Union (mIoU) over all classes, excluding the background class. IoU is generally calculated as $IoU = \frac{|A \cap B|}{|A \cup B|}$. Hence, IoU and JC are essentially the same. mIoU is popular for computer vision tasks, where usually the pixel-wise mean IoU is computed as $mIoU = \frac{\Sigma_i |A_i \cap B_i|}{\Sigma_i |A_i \cup B_i|}$ for images i. The term JC is popular in the medical literature, where usually the patient-wise/per-image mean JC is calculated as $JC = \Sigma_i \frac{|A_i \cap B_i|}{|A_i \cup B_i|}$. For validation, we follow the original PixelSSL repository and use mIoU per image for all experiments, except for CT-MRI transfer experiments, where we modify it to mean IoU per

[2]https://github.com/ZHKKKe/PixelSSL/tree/master/pixelssl
[3]https://github.com/milesial/Pytorch-UNet

	Trained with **8** labeled examples			Trained with **20** labeled examples			Trained with **118** labeled examples		
Method	CT	MRI T1-in	MRI T1-out	CT	MRI T1-in	MRI T1-out	CT	MRI T1-in	MRI T1-out
U-Net	0.7610	0.2568	0.3293	0.8229	0.3428	0.2310	0.8680	0.4453	0.4177
DeepLab	0.7591	0.2848	0.3286	0.8001	0.4019	0.3798	0.9056	0.4390	0.3767
MT	0.7815	0.3589	0.0955	0.8197	0.5319	0.1908	0.8834	0.4873	0.2596
AdvSSL	0.8005	0.5381	**0.5855**	0.8105	0.5409	0.4087	0.8551	0.4003	0.3548
GCT	0.6997	0.4733	0.4091	0.7951	0.3469	**0.4620**	0.9085	0.4341	0.3816
Ours-NO	0.8036	0.4811	0.5135	0.8462	**0.5538**	0.4511	0.8603	0.5055	**0.5633**
Ours	**0.8747**	**0.5565**	0.5678	**0.8961**	0.4989	0.4575	**0.9169**	**0.5097**	0.5243

Table 5: **CT-MRI Transfer Liver Segmentation.** Numbers are Dice per patient. Here, CT is the in-domain data set on which we both train and evaluate. We also evaluate on additional unseen MRI data [12] for liver segmentation. Ours as well as the semi-supervised methods use additional 70 CT volumes from the LITS2017 [1] testing set as unlabeled data samples for training.

pixel. The original implementation in the repository does not include empty slices, where the liver is not visible, in the mIoU calculation. We modify this to calculate pixel-wise mIoU using all slices, including the empty ones.

For final evaluation, we follow the popular metrics in the literature. As described above, we report Dice per patient for both x-ray and CT-MRI experiments. Note that each x-ray image corresponds to one patient, while each CT/MR volume from one patient consists of many slices. mIoU per pixel is reported for face parts segmentation and mean JC per patient for skin segmentation.

6. Additional Results

6.1. Baselines on CT-MRI Liver Segmentation Task

In the main paper, we only report the supervised baseline performance of U-Net for comparison to our method. Here, we provide additional results from our other semi-supervised and fully-supervised baselines, both quantitatively (see Table 5) and also qualitatively (see Figure 3). Note that we did not update our method's results; only semi-supervised baseline results of MT, AdvSSL and GCT as well as the fully-supervised baseline results of DeepLabV2 are added to the table. Our method outperforms these baselines on in-domain CT test data by a large margin for all training setups with different numbers of labeled images and it is still robust during cross-domain application on MRI data. In two cases, AdvSSL and GCT slightly outperform our method on MRI T1-out data, but their in-domain performance is significantly lower than our method's by up to 10 percent. We also observe that some methods are not able to predict anything for out-of-domain MR test images; for example, see qualitative results of MT and AdvSSL in Figure 3.

6.2. Test-time Optimization Visualizations

We also visualize the test-time optimization trajectories for embedding inference of test images. Please see Figure 4 for chest x-ray optimization results, Figure 5 for skin lesion optimization results, and Figure 6 for CT-MRI liver optimization results.

6.3. Latent Code Interpolation

We show interpolations between two random latent codes including both images and their segmentation labels generated by our generator on the CelebA-Mask dataset (see Figure 7). Note that the interpolation results are smooth between two random latent codes and the images and labels are consistent. These interpolations are performed in $\mathcal{Z}$-space. We further show interpolation results between random latent codes and latent codes obtained by test-time optimization from other out-of-domain datasets. These interpolations are performed in $\mathcal{W}^+$-space, because embedding inference takes place in $\mathcal{W}^+$. Please see Figure 8 for inputs from MetFace dataset, Figure 9 for inputs from Webcaricature [8], which contains caricatures of human faces, and Figure 10 for extreme out-of-domain inputs, including animal faces and a house that resembles a human face.

6.4. Class-Wise Face Parts Segmentation Results

We provide additional class-wise face parts segmentation results for different numbers of labeled data used during training. Please see Tables 6, 7, and 8. Note that our method outperforms other baseline methods in most classes, both in- and out-of-domain by a large margin. We also observe that test-time optimization improves class-wise performance, especially for rare and “small” classes, such as ear and eyebrow.

CelebA-Mask (In-Domain)								
Method	Ear	Eye	Eyebrow	Skin	Hair	Mouth	Nose	Mean
U-Net	0.3381	0.4429	0.5138	0.7575	0.7022	0.5988	0.5259	0.5764
DeepLab	0.2706	0.4828	0.3921	0.7746	0.6654	0.6308	0.6718	0.5554
MT	0.0000	0.0000	0.0000	0.5699	0.1860	0.0004	0.0010	0.1082
AdvSSL	0.2908	0.3275	0.3945	0.7572	**0.5932**	0.6094	0.6271	0.5142
GCT	0.1317	0.1926	0.0295	0.6866	0.5565	0.4520	0.5366	0.3694
Ours-NO	0.4855	0.6004	0.5403	0.8383	0.5659	0.7139	0.7872	0.6473
Ours	**0.6400**	**0.6069**	**0.6071**	**0.8531**	0.5878	**0.7437**	**0.7925**	**0.6902**
MetFace-40 (Out-Of-Domain)								
Method	Ear	Eye	Eyebrow	Skin	Hair	Mouth	Nose	Mean
U-Net	0.1036	0.1568	0.2371	0.4840	0.3718	0.2272	0.3817	0.2803
DeepLab	0.1016	0.3957	0.2479	0.6569	0.4784	0.5052	0.5975	0.4262
MT	0.0000	0.0000	0.0000	0.6157	0.3737	0.0000	0.0009	0.1415
AdvSSL	0.1247	0.1624	0.2526	0.7014	0.4795	0.4956	0.6018	0.4026
GCT	0.0421	0.1247	0.0192	0.5929	0.3976	0.4095	0.5403	0.3038
Ours-NO	0.2982	**0.4986**	0.3972	0.8114	**0.5502**	0.6034	0.6953	0.5506
Ours	**0.4395**	0.4674	**0.4704**	**0.8293**	0.5452	**0.6581**	**0.7082**	**0.5883**

Table 6: **CelebA-Mask Segmentation Results using 30 Labels during Training.** CelebA-Mask segmentation results for different face parts for in-domain data (real faces, top part) and for out-of-domain data (MetFaces, bottom part).

CelebA-Mask (In-Domain)								
Method	Ear	Eye	Eyebrow	Skin	Hair	Mouth	Nose	Mean
U-Net	0.5095	0.5686	0.6040	0.8414	0.7790	0.7627	0.7513	0.6880
DeepLab	0.4313	0.6308	0.4905	0.8317	0.7392	0.7174	0.7729	0.6591
MT	0.4125	0.4946	0.4035	0.7969	0.7370	0.6398	0.6153	0.5857
AdvSSL	0.4902	0.6465	0.5103	0.8517	0.7567	0.7453	0.7912	0.6846
GCT	0.4411	0.5714	0.3623	0.8502	0.7977	0.6965	0.7631	0.6403
Ours-NO	0.5064	0.6474	0.5437	0.8516	0.7834	0.7601	0.8183	0.7016
Ours	**0.6722**	**0.6900**	**0.6101**	**0.8798**	**0.8164**	**0.8173**	**0.8343**	**0.7600**
MetFace-40 (Out-Of-Domain)								
Method	Ear	Eye	Eyebrow	Skin	Hair	Mouth	Nose	Mean
U-Net	0.1036	0.1568	0.2371	0.4840	0.3718	0.2272	0.3817	0.2803
DeepLab	0.1804	0.4882	0.3494	0.7437	0.5372	0.5491	0.6434	0.4988
MT	0.2217	0.2709	0.2603	0.6986	0.5048	0.5031	0.5537	0.4305
AdvSSL	0.1913	0.4663	0.3184	0.7374	0.5620	0.5745	0.6708	0.5029
GCT	0.1014	0.3530	0.1819	0.7898	0.6430	0.5838	0.6715	0.4749
Ours-NO	0.2845	0.5260	0.4011	0.8000	0.6141	0.6108	0.7133	0.5643
Ours	**0.5030**	**0.5517**	**0.4226**	**0.8459**	**0.6810**	**0.6702**	**0.7612**	**0.6336**

Table 7: **CelebA-Mask Segmentation Results using 150 Labels during Training.** CelebA-Mask segmentation results for different face parts for in-domain data (real faces, top part) and for out-of-domain data (MetFaces, bottom part).

CelebA-Mask (In-Domain)								
Method	Ear	Eye	Eyebrow	Skin	Hair	Mouth	Nose	Mean
U-Net	0.5290	0.6608	0.6219	0.8708	0.8129	0.7955	0.7711	0.7231
DeepLab	0.6135	0.6836	0.5911	0.8774	0.8252	0.7957	0.8242	0.7444
MT	0.5396	0.6586	0.5317	0.8619	0.8104	0.7688	0.7951	0.7094
AdvSSL	0.6708	0.7254	**0.6309**	**0.8955**	0.8464	0.8376	**0.8445**	0.7787
GCT	0.6467	0.6991	0.6108	0.8932	**0.8538**	0.8174	0.8408	0.7660
Ours-NO	0.5143	0.6770	0.5479	0.8574	0.7937	0.7758	0.8201	0.7123
Ours	**0.6968**	**0.7470**	0.6150	0.8893	0.8238	**0.8539**	0.8416	**0.7810**

MetFace-40 (Out-Of-Domain)								
Method	Ear	Eye	Eyebrow	Skin	Hair	Mouth	Nose	Mean
U-Net	0.1026	0.3393	0.3117	0.6922	0.5347	0.4039	0.4762	0.4086
DeepLab	0.2549	0.5120	0.4026	0.8158	0.6269	0.6155	0.7350	0.5661
MT	0.1890	0.5135	0.3925	0.7206	0.5266	0.5663	0.6837	0.5132
AdvSSL	0.3241	0.5340	0.4236	0.8345	0.6556	0.6642	0.7604	0.5995
GCT	0.2467	0.5486	0.4393	0.8434	0.6753	0.6744	0.7563	0.5977
Ours-NO	0.2795	0.5422	0.3880	0.8114	0.6370	0.6173	0.7492	0.5749
Ours	**0.5224**	**0.6025**	**0.4810**	**0.8534**	**0.6995**	**0.6957**	**0.7915**	**0.6633**

Table 8: **CelebA-Mask Segmentation Results using 1500 Labels during Training.** CelebA-Mask segmentation results for different face parts for in-domain data (real faces, top part) and for out-of-domain data (MetFaces, bottom part).

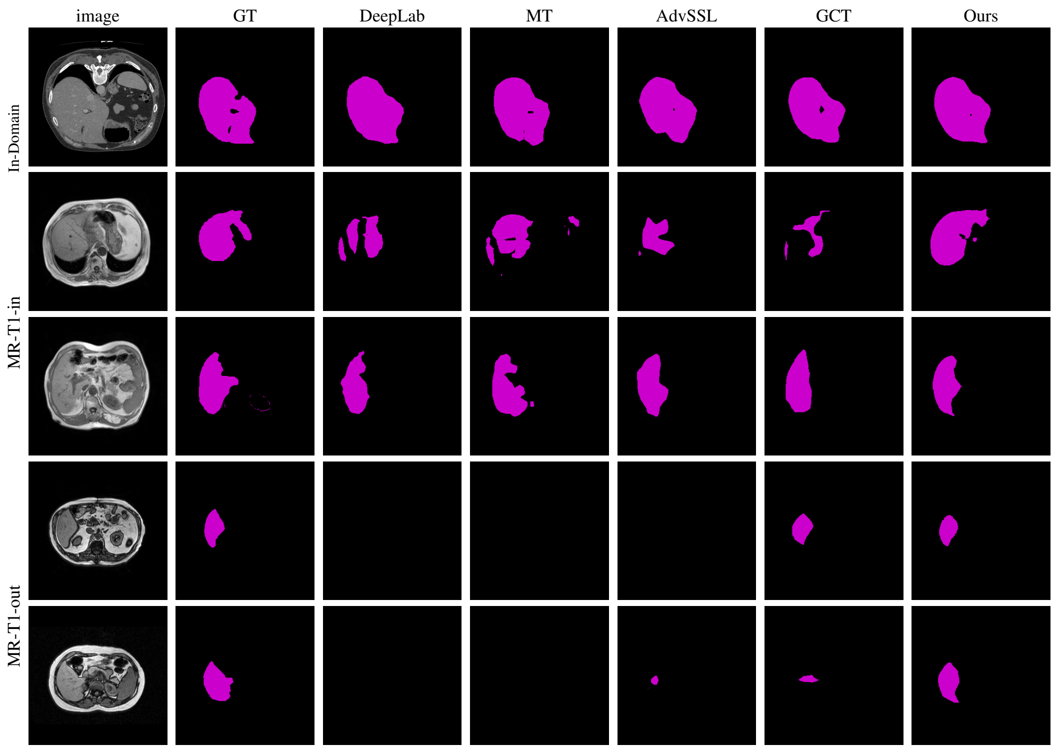

Figure 3: **CT-MRI Liver Segmentation.** Qualitative examples for both in-domain (CT, first row) and out-of-domain (MRI, row 2-5) data. Compared to the baseline methods, *Ours* is more robust to out-of-domain input, which the other methods do not predict well or not at all.

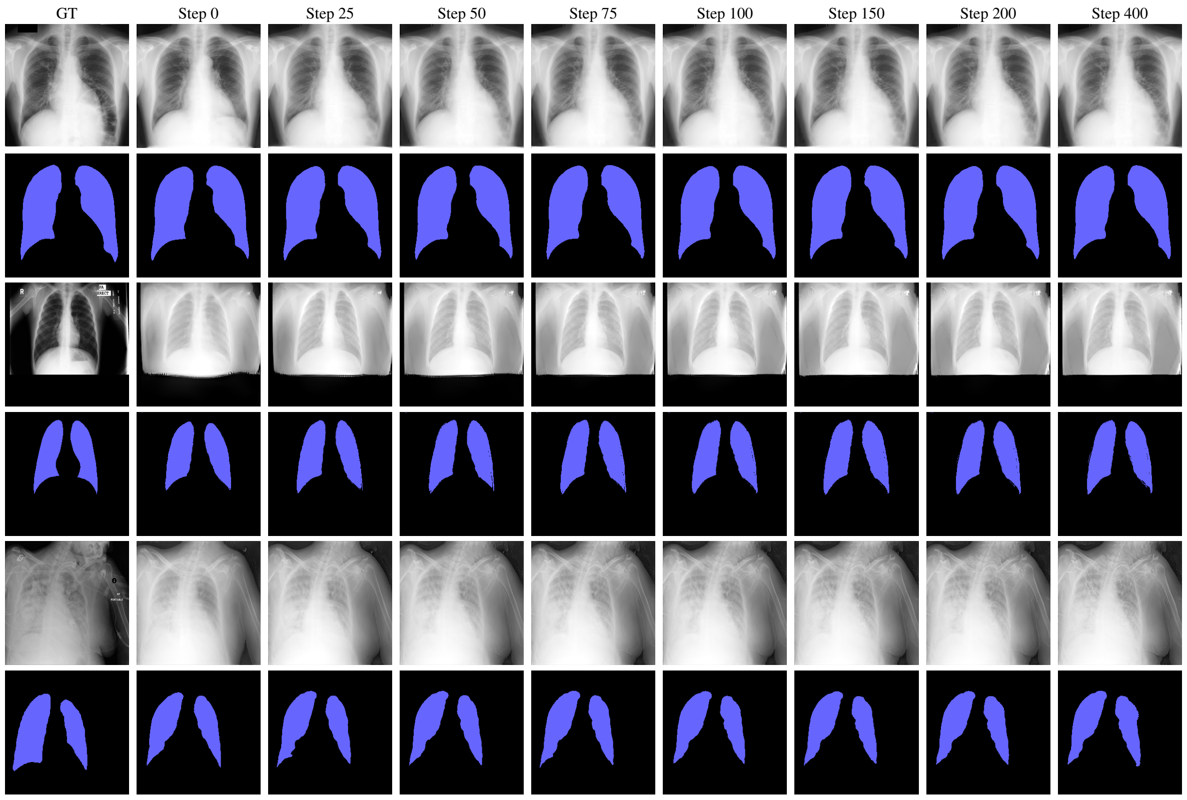

Figure 4: **Chest X-ray Segmentation Optimization.** Qualitative examples for reconstructions and segmentation label predictions at different optimization steps. Step 0 means using the latent code predicted by the encoder without any iterative optimization.

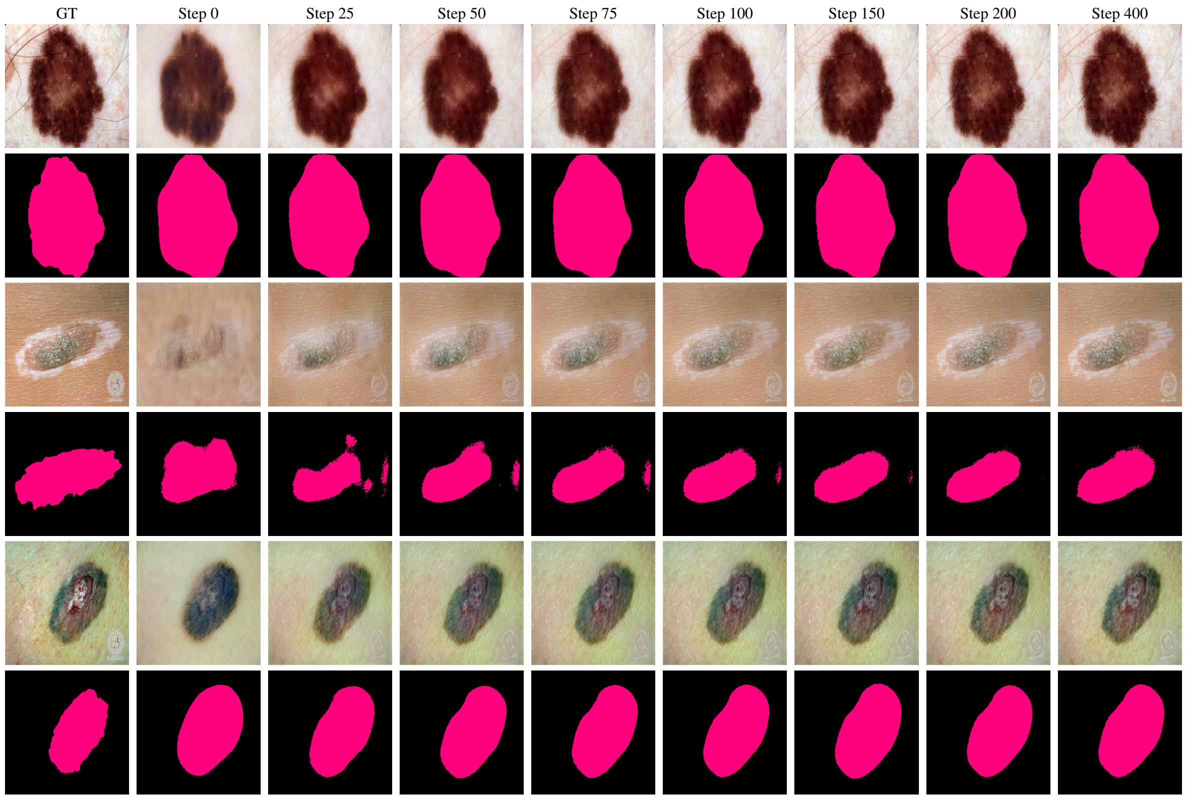

Figure 5: **Skin Lesion Segmentation Optimization.** Qualitative examples for reconstructions and segmentation label predictions at different optimization steps. Step 0 means using the latent code predicted by the encoder without any iterative optimization.

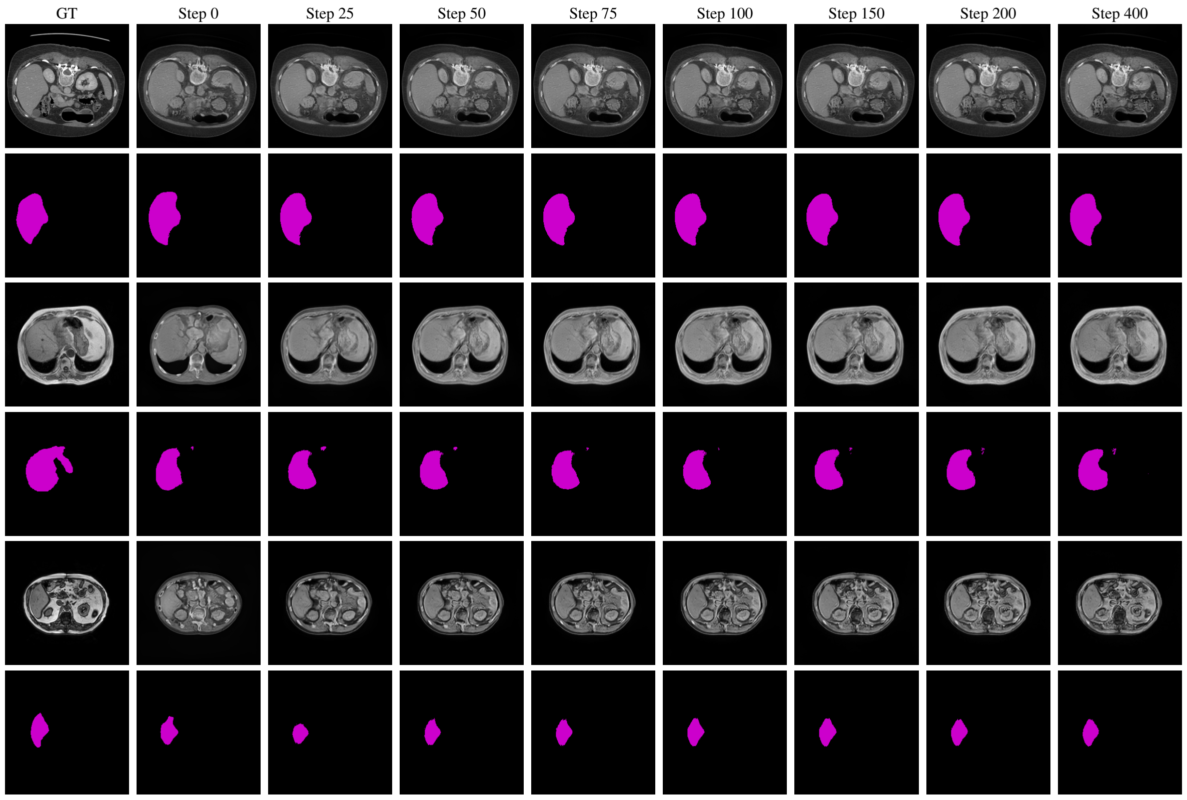

Figure 6: **CT-MRI Liver Segmentation Optimization.** Qualitative examples for reconstructions and segmentation label predictions at different optimization steps. Step 0 means using the latent code predicted by the encoder without any iterative optimization.

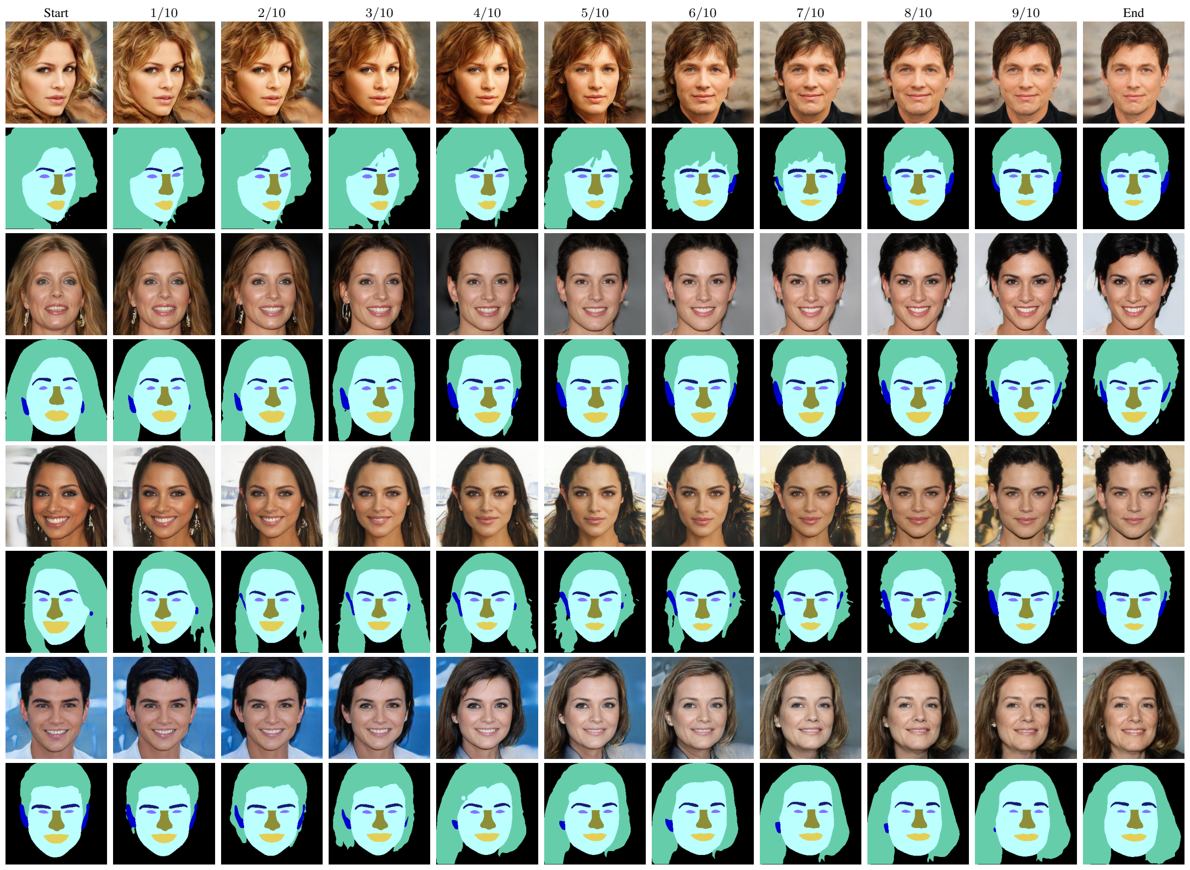

Figure 7: **Interpolations between Random Latent Codes on CelebA-Mask.** Linear interpolations between two random latent codes $z_1 \in \mathcal{Z}$ and $z_2 \in \mathcal{Z}$. We show both the interpolated images and their semantic segmentation labels. The results show that the generative model learnt a smooth latent space with meaningful images along the interpolation path. Furthermore, we observe consistency between images and predicted labels along the interpolation path.

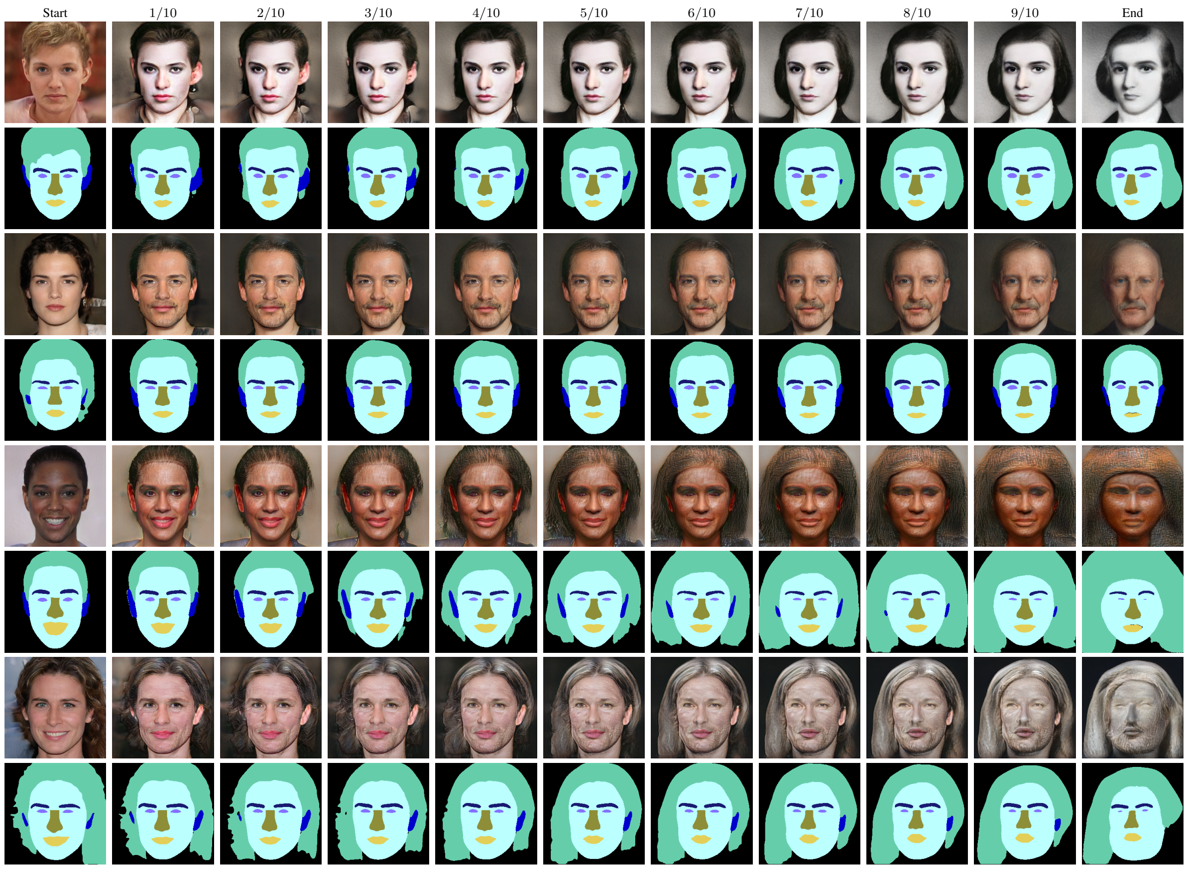

Figure 8: **Interpolations between Random Latent Codes and MetFaces.** Linear interpolations between random latent codes and latent codes of MetFace images. We obtain the MetFace latent codes by performing inverse optimization. The interpolation is done in $\mathcal{W}^+$-space. We show both the interpolated images and their semantic segmentation labels. The results show that the generative model learnt a smooth latent space with meaningful images along the interpolation path. Furthermore, we observe consistency between images and predicted labels along the interpolation path. This is noteworthy, since we are interpolating beyond the training domain.

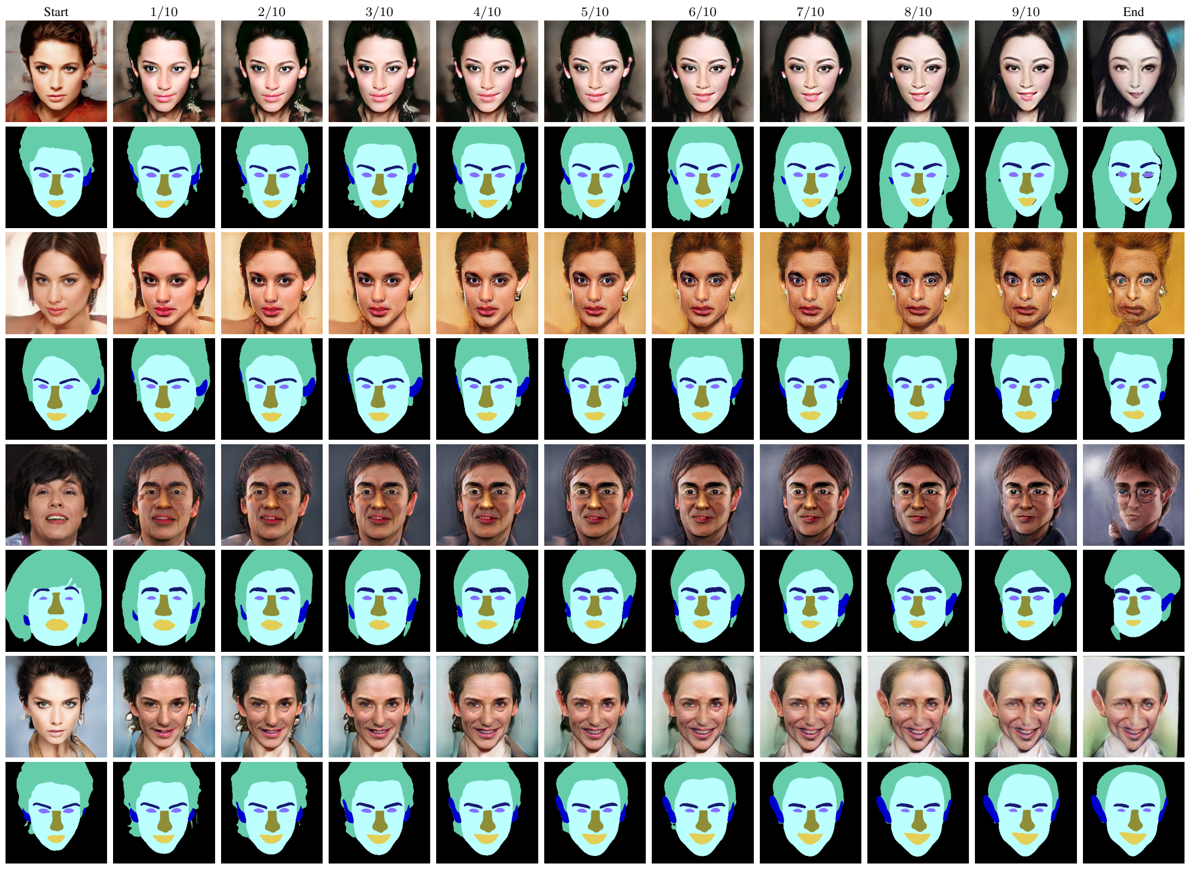

Figure 9: **Interpolations between Random Latent Codes and Cartoon Faces.** Linear interpolations between random latent codes and latent codes of selected Cartoons. We obtain the Cartoon latent codes by performing inverse optimization. The interpolation is done in $\mathcal{W}^+$-space. We show both the interpolated images and their semantic segmentation labels. The results show that the generative model learnt a smooth latent space with meaningful images along the interpolation path. Furthermore, we observe consistency between images and predicted labels along the interpolation path. This is noteworthy, since we are interpolating beyond the training domain.

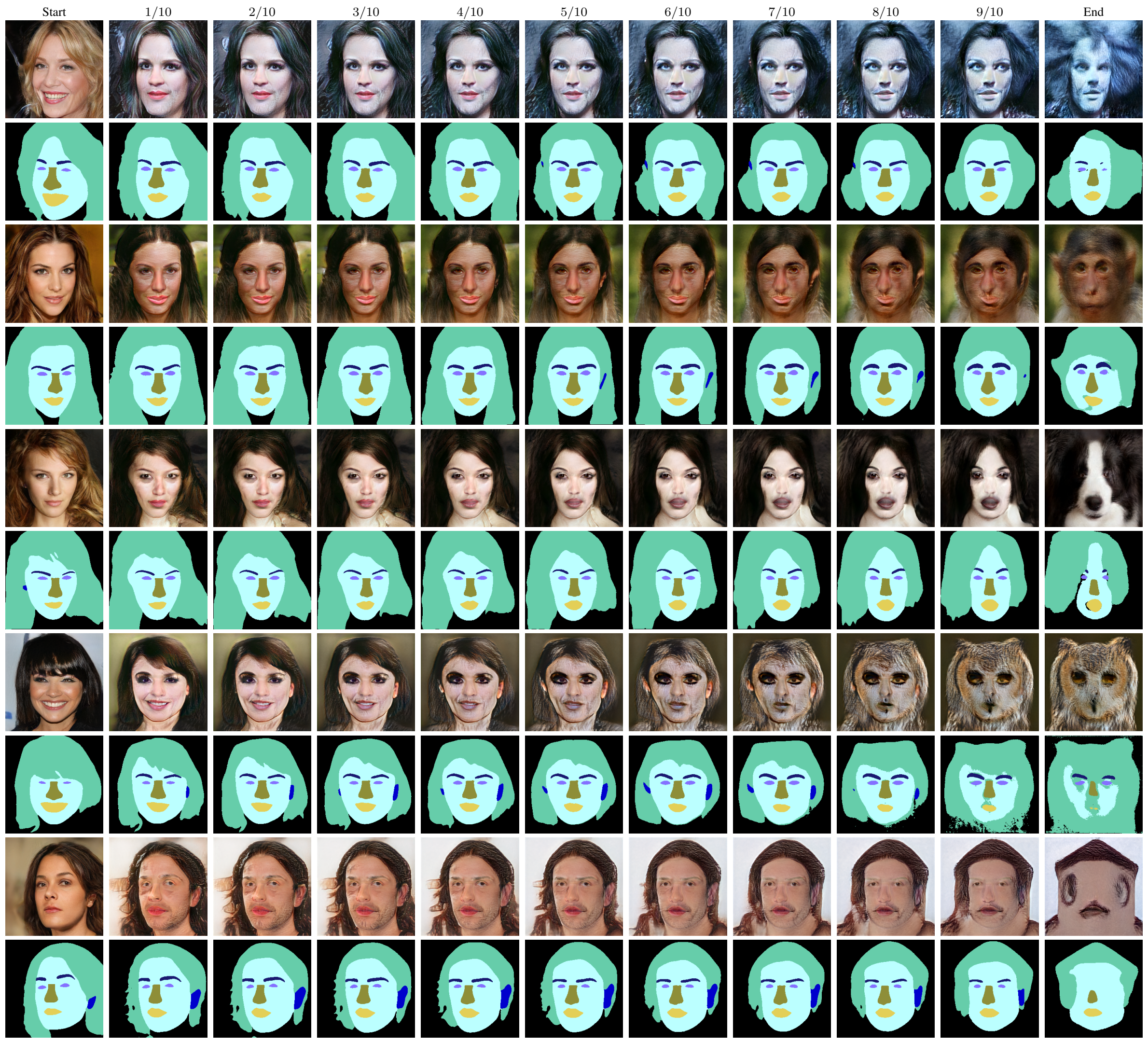

Figure 10: **Interpolations between Random Latent Codes and Extreme Out-Of-Domain Faces.** Linear interpolations between random latent codes and latent codes of selected extreme out-of-domain faces and face-like objects. We obtain the extreme out-of-domain faces' latent codes by inverse optimization. The interpolation is done in $\mathcal{W}^{+}$-space. We show both the interpolated images and their semantic segmentation labels. The results show that the generative model learnt a smooth latent space with meaningful images along the interpolation path. Furthermore, we observe reasonable consistency between images and predicted labels along the interpolation path, which we consider remarkable, since the considered images are extremely different compared to the real faces used for training the model.

References

[1] Patrick Bilic, Patrick Ferdinand Christ, Eugene Vorontsov, Grzegorz Chlebus, Hao Chen, Qi Dou, Chi-Wing Fu, Xiao Han, Pheng-Ann Heng, Jürgen Hesser, et al. The liver tumor segmentation benchmark (lits). *arXiv preprint arXiv:1901.04056*, 2019. 4, 5, 6

[2] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834—848, April 2018. 5

[3] Noel Codella, Veronica Rotemberg, Philipp Tschandl, M Emre Celebi, Stephen Dusza, David Gutman, Brian Helba, Aadi Kalloo, Konstantinos Liopyris, Michael Marchetti, et al. Skin lesion analysis toward melanoma detection 2018: A challenge hosted by the international skin imaging collaboration (isic). *arXiv preprint arXiv:1902.03368*, 2019. 4

[4] Tiantian Fang and Alexander Schwing. Co-generation with gans using ais based hmc. In *Advances in Neural Information Processing Systems*, pages 5808–5819, 2019. 2

[5] Jeffrey Luc Glaister. Automatic segmentation of skin lesions from dermatological photographs. Master's thesis, University of Waterloo, 2013. 4

[6] Geoffrey E. Hinton. To recognize shapes, first learn to generate images. *Progress in brain research*, 165:535–47, 2007. 3

[7] Wei-Chih Hung, Yi-Hsuan Tsai, Yan-Ting Liou, Yen-Yu Lin, and Ming-Hsuan Yang. Adversarial learning for semi-supervised semantic segmentation. *arXiv preprint arXiv:1802.07934*, 2018. 5

[8] Jing Huo, Wenbin Li, Yinghuan Shi, Yang Gao, and Hujun Yin. Webcaricature: a benchmark for caricature recognition. In *British Machine Vision Conference*, 2018. 6

[9] Stefan Jaeger, Sema Candemir, Sameer Antani, Yì-Xiáng J Wáng, Pu-Xuan Lu, and George Thoma. Two public chest x-ray datasets for computer-aided screening of pulmonary diseases. *Quantitative imaging in medicine and surgery*, 4(6):475, 2014. 4

[10] Tero Karras, Miika Aittala, Janne Hellsten, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Training generative adversarial networks with limited data. In *NeurIPS*, 2020. 5

[11] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4401–4410, 2019. 3

[12] A. Emre Kavur, N. Sinem Gezer, Mustafa Barış, Pierre-Henri Conze, Vladimir Groza, Duc Duy Pham, Soumick Chatterjee, Philipp Ernst, Savaş Özkan, Bora Baydar, Dmitry Lachinov, Shuo Han, Josef Pauli, Fabian Isensee, Matthias Perkonigg, Rachana Sathish, Ronnie Rajan, Sinem Aslan, Debdoot Sheet, Gurbandurdy Dovletov, Oliver Speck, Andreas Nürnberger, Klaus H. Maier-Hein, Gözde Bozdağı Akar, Gözde Ünal, Oğuz Dicle, and M. Alper Selver. CHAOS Challenge - Combined (CT-MR) Healthy Abdominal Organ Segmentation. *arXiv preprint arXiv:2001.06535*, 2020. 4, 5, 6

[13] A. Emre Kavur, Naciye Sinem Gezer, Mustafa Barış, Yusuf Şahin, Savaş Özkan, Bora Baydar, Ulaş Yüksel, Çağlar Kılıkçıer, Şahin Olut, Gözde Bozdağı Akar, Gözde Ünal, Oğuz Dicle, and M. Alper Selver. Comparison of semi-automatic and deep learning based automatic methods for liver segmentation in living liver transplant donors. *Diagnostic and Interventional Radiology*, 26:11–21, Jan. 2020. 5

[14] Ali Emre Kavur, M. Alper Selver, Oğuz Dicle, Mustafa Barış, and N. Sinem Gezer. CHAOS - Combined (CT-MR) Healthy Abdominal Organ Segmentation Challenge Data, Apr. 2019. 5

[15] Zhanghan Ke, Di Qiu, Kaican Li, Qiong Yan, and Rynson W. H. Lau. Guided collaborative training for pixel-wise semi-supervised learning. *arXiv preprint arXiv:2008.05258*, 2020. 5

[16] Cheng-Han Lee, Ziwei Liu, Lingyun Wu, and Ping Luo. Maskgan: Towards diverse and interactive facial image manipulation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 5

[17] Jae Hyun Lim and Jong Chul Ye. Geometric gan. *arXiv preprint arXiv:1705.02894*, 2017. 3

[18] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017. 3

[19] Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. On the variance of the adaptive learning rate and beyond. *arXiv preprint arXiv:1908.03265*, 2019. 3

[20] Teresa Mendonça, Pedro M Ferreira, Jorge S Marques, André RS Marcal, and Jorge Rozeira. Ph 2-a dermoscopic image database for research and benchmarking. In *2013 35th annual international conference of the IEEE engineering in medicine and biology society (EMBC)*, pages 5437–5440. IEEE, 2013. 4

[21] Lars Mescheder, Andreas Geiger, and Sebastian Nowozin. Which training methods for gans do actually converge? *arXiv preprint arXiv:1801.04406*, 2018. 3

[22] Yujin Oh, Sangjoon Park, and Jong Chul Ye. Deep learning covid-19 features on cxr using limited training data sets. *IEEE Transactions on Medical Imaging*, page 1–1, 2020. 4

[23] M. Ranzato, J. Susskind, V. Mnih, and G. Hinton. On deep generative models with applications to recognition. In *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '11, page 2857–2864, USA, 2011. IEEE Computer Society. 3

[24] Vinicius Ribeiro, Sandra Avila, and Eduardo Valle. Handling inter-annotator agreement for automated skin lesion segmentation. *arXiv preprint arXiv:1906.02415*, 2019. 4

[25] Elad Richardson, Yuval Alaluf, Or Patashnik, Yotam Nitzan, Yaniv Azar, Stav Shapiro, and Daniel Cohen-Or. Encoding in style: a stylegan encoder for image-to-image translation. *arXiv preprint arXiv:2008.00951*, 2020. 3

[26] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015. 5

[27] Veronica Rotemberg, Nicholas Kurtansky, Brigid Betz-Stablein, Liam Caffery, Emmanouil Chousakos, Noel Codella, Marc Combalia, Stephen Dusza, Pascale Guitera, David Gutman, Allan Halpern, Harald Kittler, Kivanc Kose, Steve Langer, Konstantinos Lioprys, Josep Malvehy, Shenara Musthaq, Jabpani Nanda, Ofer Reiter, George Shih, Alexander Stratigos, Philipp Tschandl, Jochen Weber, and H. Peter Soyer. A patient-centric dataset of images and metadata for identifying melanomas using clinical context. *arXiv preprint arXiv:2008.07360*, 2020. 4

[28] Junji Shiraishi, Shigehiko Katsuragawa, Junpei Ikezoe, Tsuneo Matsumoto, Takeshi Kobayashi, Ken-ichi Komatsu, Mitate Matsui, Hiroshi Fujita, Yoshie Kodera, and Kunio Doi. Development of a digital image database for chest radiographs with and without a lung nodule: receiver operating characteristic analysis of radiologists' detection of pulmonary nodules. *American Journal of Roentgenology*, 174(1):71–74, 2000. 4

[29] Nitish Srivastava and Russ R Salakhutdinov. Multimodal learning with deep boltzmann machines. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25, pages 2222–2230. Curran Associates, Inc., 2012. 3

[30] Sergii Stirenko, Yuriy Kochura, Oleg Alienin, Oleksandr Rokovyi, Yuri Gordienko, Peng Gang, and Wei Zeng. Chest x-ray analysis of tuberculosis by deep learning with segmentation and augmentation. In *2018 IEEE 38th International Conference on Electronics and Nanotechnology (ELNANO)*, pages 422–428. IEEE, 2018. 4

[31] Youbao Tang, Yuxing Tang, Jing Xiao, and Ronald M Summers. Xlsor: A robust and accurate lung segmentor on chest x-rays using criss-cross attention and customized radiorealistic abnormalities generation. *arXiv preprint arXiv:1904.09229*, 2019. 4

[32] Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *Advances in neural information processing systems*, pages 1195–1204, 2017. 5

[33] Bram Van Ginneken, Mikkel B Stegmann, and Marco Loog. Segmentation of anatomical structures in chest radiographs using supervised methods: a comparative study on a public database. *Medical image analysis*, 10(1):19–40, 2006. 4

[34] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8798–8807, 2018. 3

[35] Xiaosong Wang, Yifan Peng, Le Lu, Zhiyong Lu, Mohammadhadi Bagheri, and Ronald M Summers. Chestx-ray8: Hospital-scale chest x-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2097–2106, 2017. 4

[36] Michael Zhang, James Lucas, Jimmy Ba, and Geoffrey E Hinton. Lookahead optimizer: k steps forward, 1 step back. In *Advances in Neural Information Processing Systems*, pages 9597–9608, 2019. 3