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

Daiqing Li, Junlin Yang, Karasten Kreis, Antonio Torralba, and Sanja Fidler

NVIDIA, University of Toronto, Yale University, MIT, and Vector Institute

CVPR 2021

Presented by Minho Park

Semi-supervised learning (SSL)

- Labeling dense **pixel-level tasks** dataset such as semantic segmentation is very time consuming and expensive.
    - This is particularly true in the medical domain (which require highly-skilled experts) and where imaging sensors vary across sites.
- **Self-supervised learning (SSL) facilitates learning with small labeled datasets by augmenting the training set with large amounts of unlabeled data.**
    - Pseudo-labeling, self-supervision, entropy-minimization, consistency-regularization, adversarial training, data augmentation, etc.
    - Current state-of-the-art SSL are based on self-supervised learning with contrastive objectives.
- However, most SSL methods are developed **for simple classification tasks.**

Out-of-domain Generalization

- One of the most important work of semi-supervised learning is out-of-domain generalization.
- In this paper, semanticGAN achieved great performance in out-of-domain semantic segmentation.
  - Face part segmentation, chest X-ray segmentation, and skin lesion segmentation.

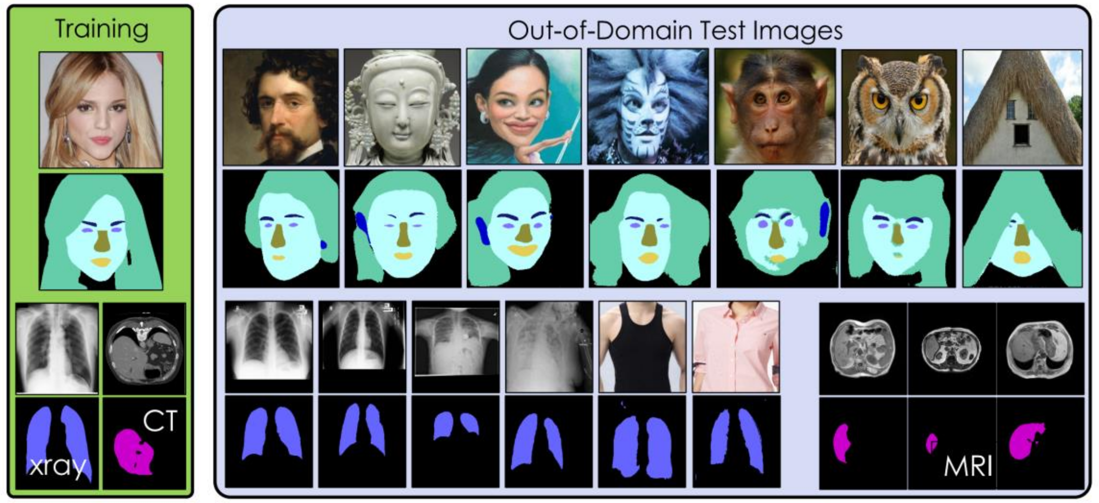


Figure 1: **Out-of-domain Generalization.** Our model trained on real faces generalizes to paintings, sculptures, cartoons and even outputs plausible segmentations for animal faces. When trained on chest x-rays, it generalizes to multiple hospitals, and even hallucinates lungs under clothed people. Our model also generalizes well from CT to MRI medical scans.

Generative vs. Discriminative Models

- To find the conditional probability $p(y|x)$,
- Discriminative model directly assume some functional form for $p(y|x)$.
- Generative model estimate the prior probability $p(y)$ and likelihood probability $p(x|y)$.
  - Bayes' theorem: $p(y|x) = \frac{p(x|y)p(y)}{p(x)}$.

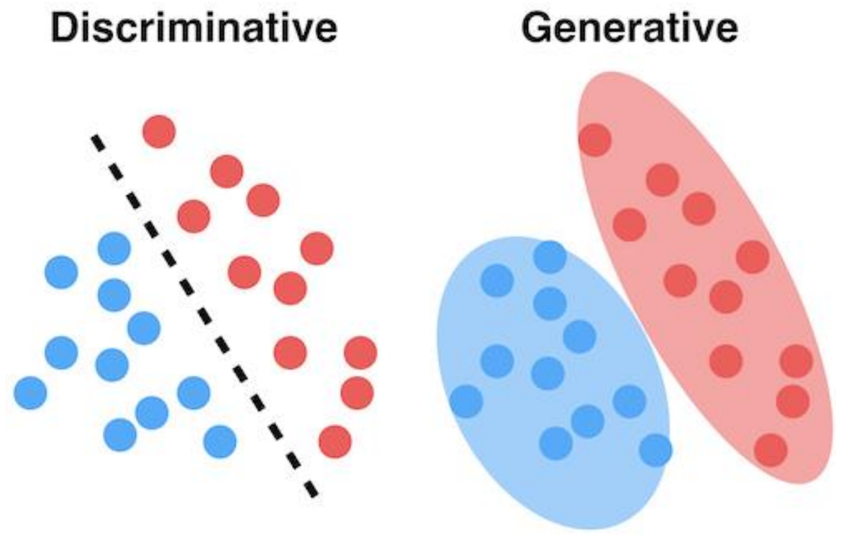


Image source: https://betterprogramming.pub/generative-vs-discriminative-models-d26def8fd64a

Model Overview

- SemanticGAN models the joint distribution of images and labels $p(x, y)$ with a GAN-based generative model.
- Training in two stages: train generator and discriminators first and encoder second.

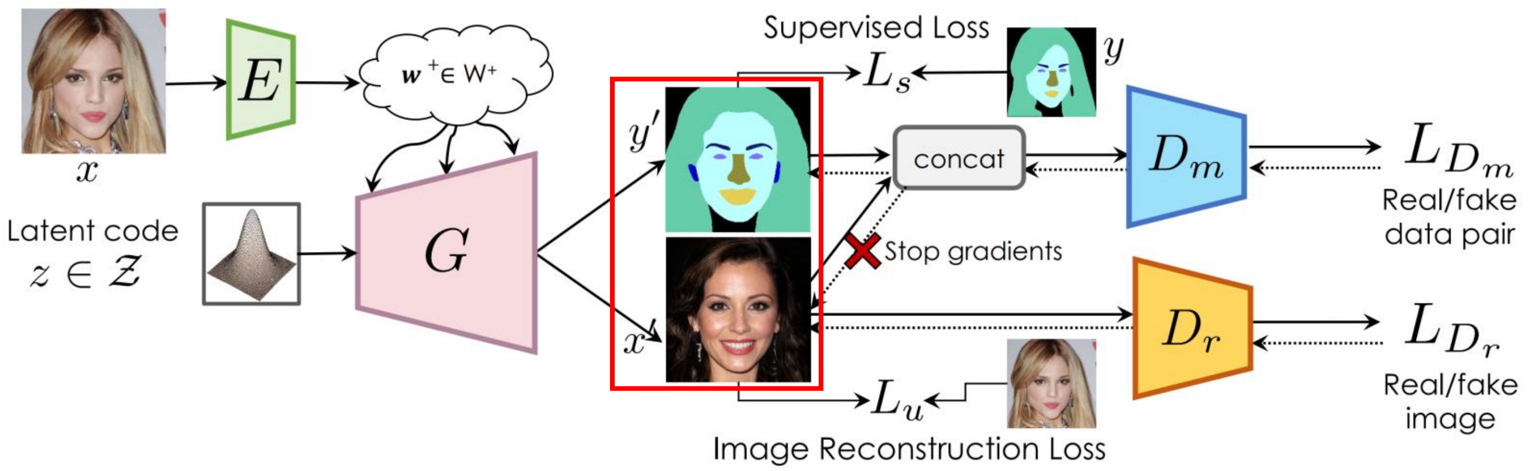


Figure 2: **Model Overview.** Generator G and discriminators D_m and D_r are trained with adversarial objectives $\mathcal{L}_G$ (not indicated here), $\mathcal{L}_{D_m}$ and $\mathcal{L}_{D_r}$. We do not backpropagate gradients from D_m into the generator's image synthesis branch. We train an additional encoder E in a supervised fashion using image and mask reconstruction losses $\mathcal{L}_u$ and $\mathcal{L}_s$.

Model

- Generator: StyleGAN2-based
  - Based on StyleGAN.
  - Redesign of instance normalization to remove generation artifacts
  - Latent space path-length regularization to encourage generator smoothness.
  - Replace progressive growing strategy to residual skip-connection design.

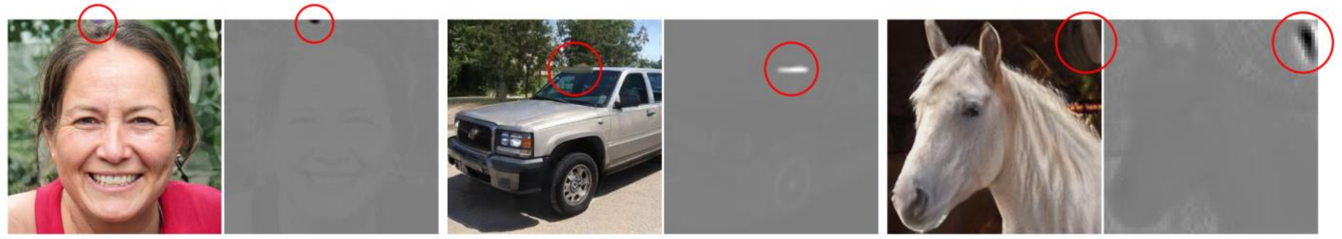
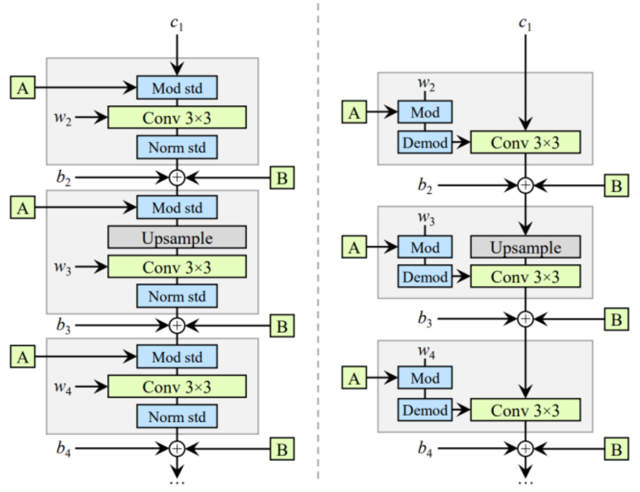


Figure 1. Instance normalization causes water droplet -like artifacts in StyleGAN images. These are not always obvious in the generated images, but if we look at the activations inside the generator network, the problem is always there, in all feature maps starting from the 64x64 resolution. It is a systemic problem that plagues all StyleGAN images.

Droplet-like artifact in StyleGAN images (StyleGAN2)



(c) Revised architecture (d) Weight demodulation

Weaker instance normalization (StyleGAN2)

Karras, Tero, et al. "Analyzing and improving the image quality of stylegan." *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020. 6

Model

- Generator: StyleGAN2-based
- They add an additional branch at each style layer to output a segmentation mask y along with the image output x.
- Define $G: \mathcal{Z} \to \mathcal{W} \to (\mathcal{X}, \mathcal{Y})$.

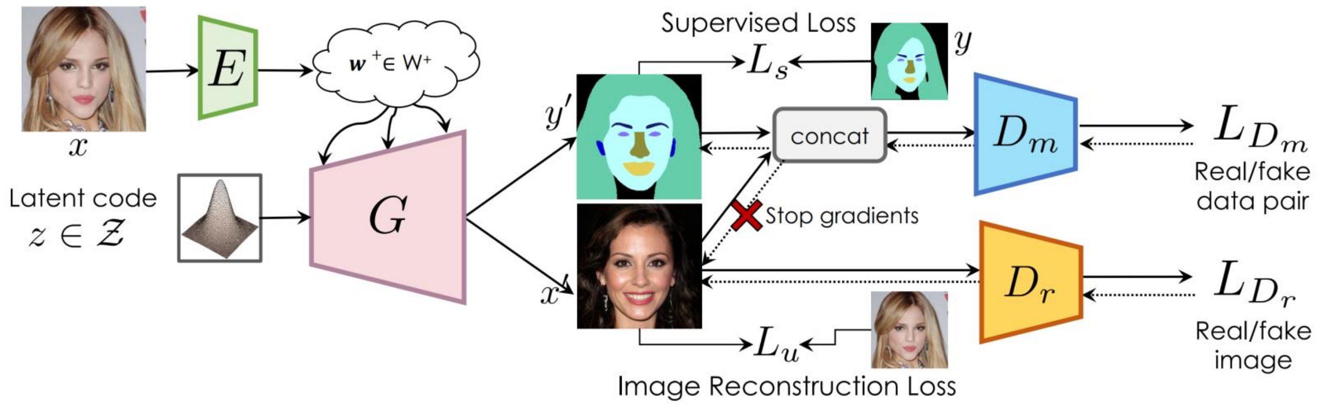


Figure 2: **Model Overview.** Generator G and discriminators D_m and D_r are trained with adversarial objectives $\mathcal{L}_G$ (not indicated here), $\mathcal{L}_{D_m}$ and $\mathcal{L}_{D_r}$. We do not backpropagate gradients from D_m into the generator's image synthesis branch. We train an additional encoder E in a supervised fashion using image and mask reconstruction losses $\mathcal{L}_u$ and $\mathcal{L}_s$.

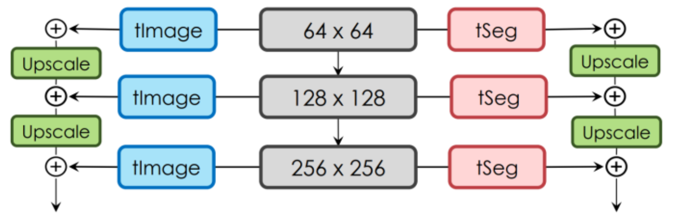


Figure 3: **Generator Architecture.** We modify StyleGAN2's image synthesis network to also output masks. The *tImage* and *tSeg* blocks output intermediate images and segmentation masks at different resolutions, respectively. Both share the same style layers.

Model

- Discriminators: Two discriminator D_r and D_m.
- $D_r: \mathcal{X} \rightarrow \mathbb{R}$ is applied on real and generated images.
- $D_m: (\mathcal{X}, \mathcal{Y}) \rightarrow \mathbb{R}$ consumes both images and pixel-wise label masks via concatenation.
  - Non-aligned image-label pairs could be easily detected as “fake”.
  - To enforce strong consistency between images and labels, multi-scale patch-based discriminator is used.

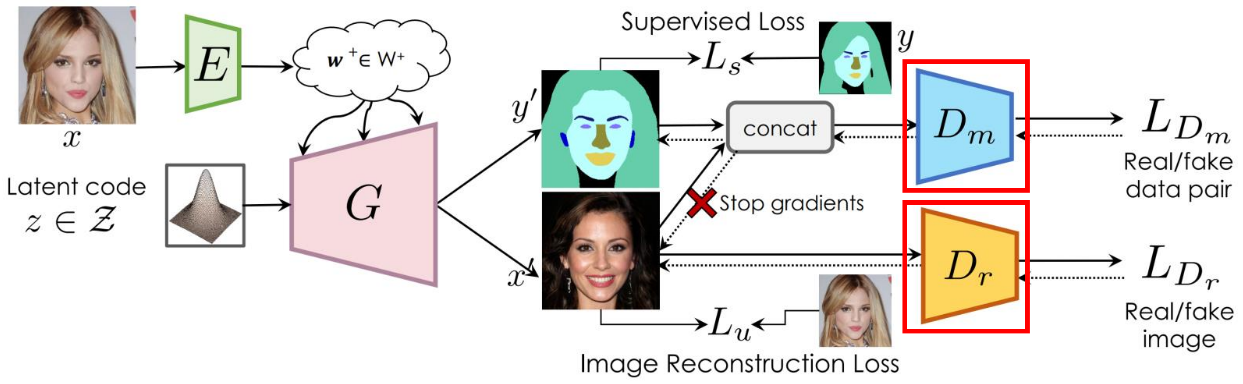


Figure 2: **Model Overview.** Generator G and discriminators D_m and D_r are trained with adversarial objectives $\mathcal{L}_G$ (not indicated here), $\mathcal{L}_{D_m}$ and $\mathcal{L}_{D_r}$. We do not backpropagate gradients from D_m into the generator's image synthesis branch. We train an additional encoder E in a supervised fashion using image and mask reconstruction losses $\mathcal{L}_u$ and $\mathcal{L}_s$.

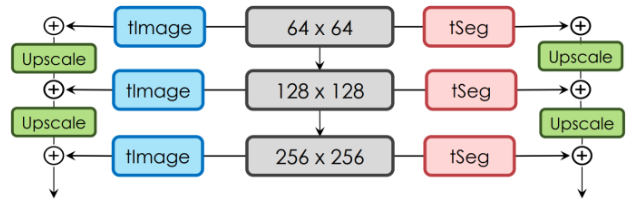
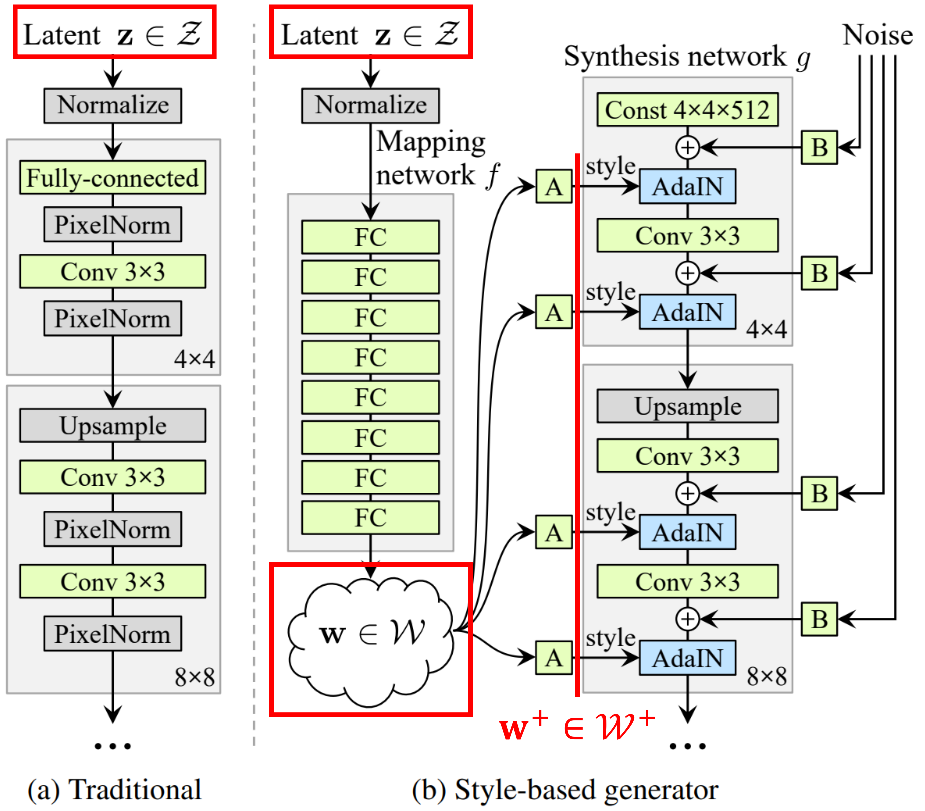


Figure 3: **Generator Architecture.** We modify StyleGAN2's image synthesis network to also output masks. The *tImage* and *tSeg* blocks output intermediate images and segmentation masks at different resolutions, respectively. Both share the same style layers.

Model

- Various type of style spaces.
- $\mathcal{Z}$: Normal distribution.
- $\mathcal{W}$: Generated from $\mathcal{Z}$ and embedded by f.
- $\mathcal{W}^+$: Layer-wise affine projected style.



Latent space $\mathcal{Z}$, intermediate latent space $\mathcal{W}$,
and extended intermediate latent space $\mathcal{W}^+$ (StyleGAN).

Karras, Tero, Samuli Laine, and Timo Aila. "A style-based generator architecture for generative adversarial networks." *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019.

Model

- Encoder $E: \mathcal{X} \rightarrow \mathcal{W}^+$.
- Architecture is based on pSp which uses a feature pyramid network as backbone.

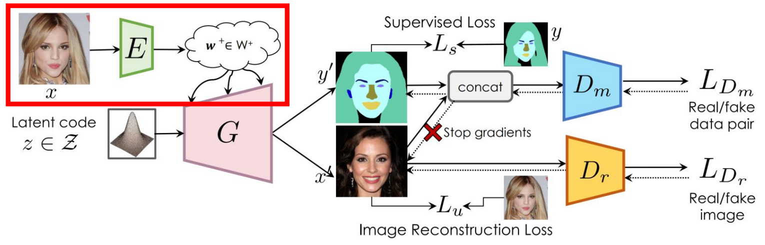


Figure 2: **Model Overview.** Generator G and discriminators D_m and D_r are trained with adversarial objectives $\mathcal{L}_G$ (not indicated here), $\mathcal{L}_{D_m}$ and $\mathcal{L}_{D_r}$. We do not backpropagate gradients from D_m into the generator's image synthesis branch. We train an additional encoder E in a supervised fashion using image and mask reconstruction losses $\mathcal{L}_u$ and $\mathcal{L}_s$.

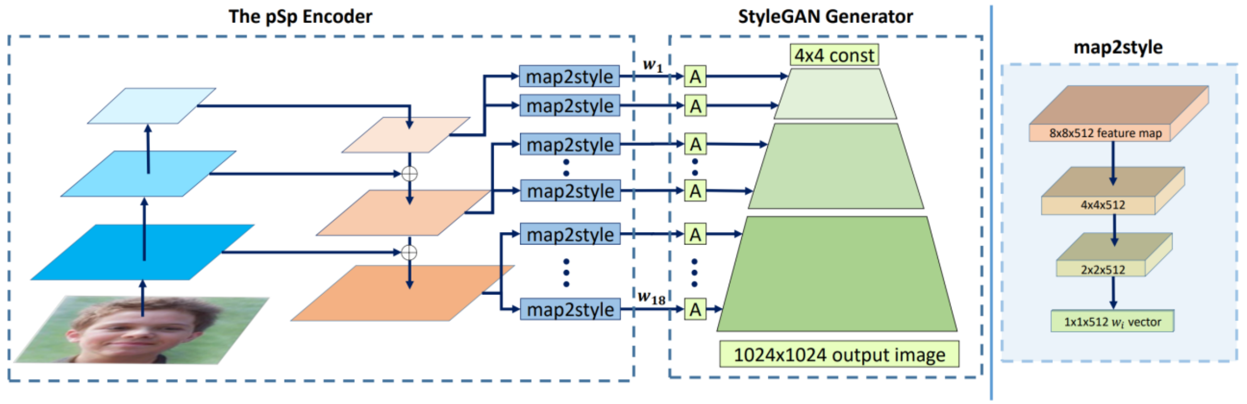


Figure 2. Our pSp architecture. Feature maps are first extracted using a standard feature pyramid over a ResNet backbone. For each of the 18 target styles, a small mapping network is trained to extract the learned styles from the corresponding feature map, where styles (0-2) are generated from the small feature map, (3-6) from the medium feature map, and (7-18) from the largest feature map. The mapping network, *map2style*, is a small fully convolutional network, which gradually reduces spatial size using a set of 2-strided convolutions followed by LeakyReLU activations. Each generated 512 vector, is fed into StyleGAN, starting from its matching affine transformation, A.

Richardson, Elad, et al. "Encoding in style: a stylegan encoder for image-to-image translation." *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021.

Training

- Train generator and discriminators first.
- Use only adversarial loss.

$$\mathcal{L}_{D_r} = \mathbb{E}_{x_r \sim D_u}[\log D_r(x_r)] + \mathbb{E}_{(x_f,\cdot)=G(z), z\sim p(z)}\left[\log\left(1 - D_r(x_f)\right)\right]$$

$$\mathcal{L}_{D_m} = \mathbb{E}_{(x_r,y_r)\sim D_l}[\log D_m(x_r, y_r)] + \mathbb{E}_{(x_f,y_f)=G(z), z\sim p(z)}\left[\log\left(1 - D_m(x_f, y_f)\right)\right]$$

$$\mathcal{L}_{G} = \mathbb{E}_{(x_f,\cdot)=G(z), z\sim p(z)}\left[\log D_r\left(x_f\right)\right] + \underline{\mathbb{E}_{(x_f,y_f)=G(z), z\sim p(z)}\left[\log\left(D_m\left(x_f, y_f\right)\right)\right]}$$

Stopping gradient into the image synthesis branch.
They want the synthesized label to be adjusted to the synthesized image.

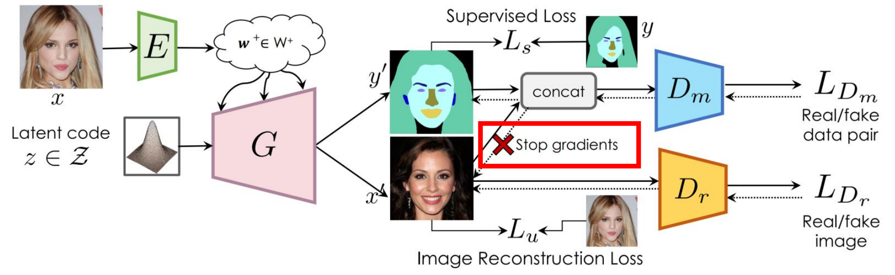


Figure 2: **Model Overview.** Generator G and discriminators D_m and D_r are trained with adversarial objectives $\mathcal{L}_G$ (not indicated here), $\mathcal{L}_{D_m}$ and $\mathcal{L}_{D_r}$. We do not backpropagate gradients from D_m into the generator's image synthesis branch. We train an additional encoder E in a supervised fashion using image and mask reconstruction losses $\mathcal{L}_u$ and $\mathcal{L}_s$.

Training

- Train encoder second.

$$\mathcal{L}_E = \mathcal{L}_s + \mathcal{L}_u$$

$$\mathcal{L}_s = \mathop{\mathbb{E}}_{(x,y)\sim D_l} \mathbf{H}(y, G_y(E(x))) + \mathbf{DC}(y, G_y(E(x)))$$

$$\mathcal{L}_u = \mathop{\mathbb{E}}_{x\sim D_l \cup D_u} \mathcal{L}_{\text{LPIPS}}(x, G_x(E(x))) + \lambda_1 ||x - G_x(E(x))||_2^2$$

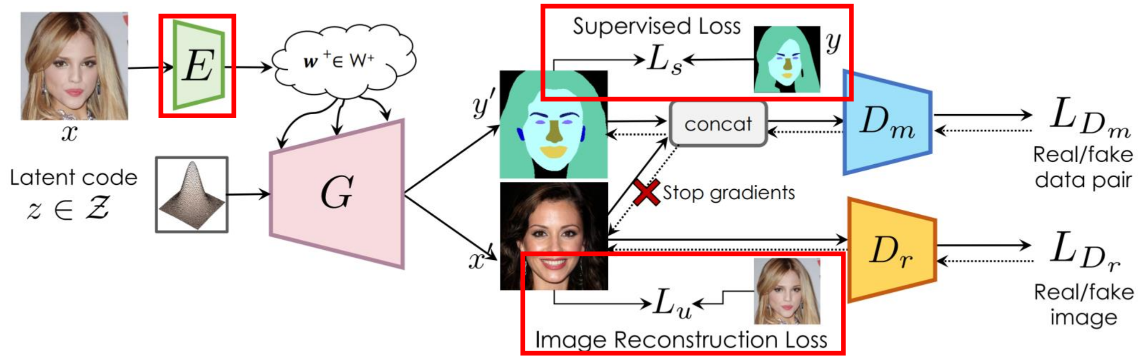


Figure 2: **Model Overview.** Generator G and discriminators D_m and D_r are trained with adversarial objectives $\mathcal{L}_G$ (not indicated here), $\mathcal{L}_{D_m}$ and $\mathcal{L}_{D_r}$. We do not backpropagate gradients from D_m into the generator's image synthesis branch. We train an additional encoder E in a supervised fashion using image and mask reconstruction losses $\mathcal{L}_u$ and $\mathcal{L}_s$.

Synthetic Samples

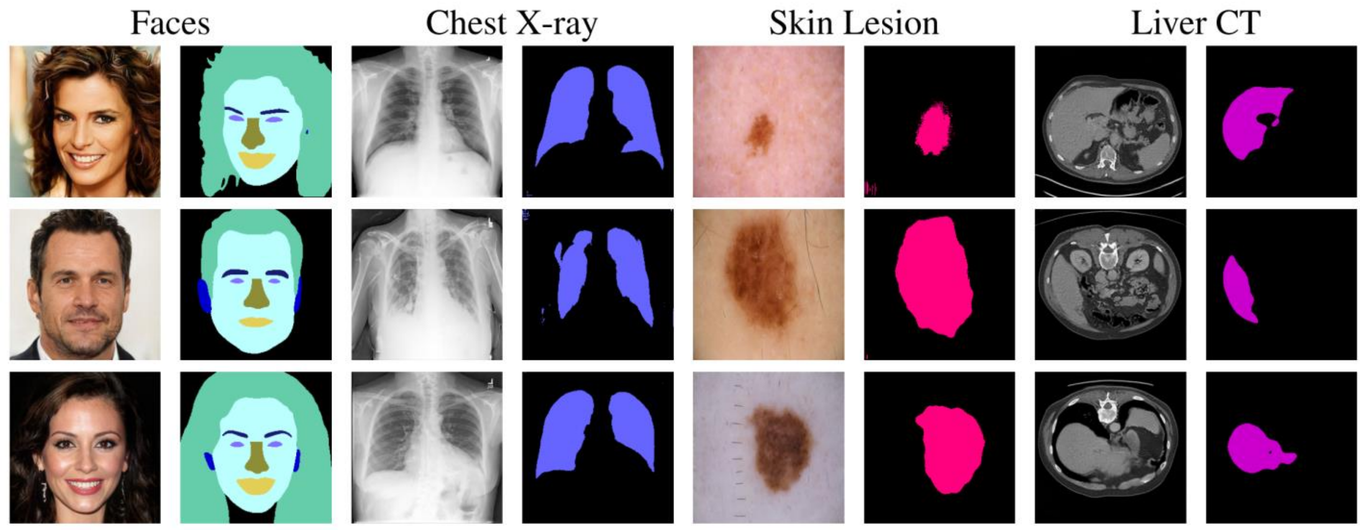


Figure 4: **Synthetic Samples** of image and pixel-wise segmentation label pairs from our generator for multiple datasets.

Inference

- Embed the target image into the generator's embedding space.

$$w^{+*} = \arg\min_{w^+ \in \mathcal{W}^+} [\mathcal{L}_{\text{reconst}}(x^*, G_x(w^+)) + \lambda_2 ||w^+ - \underline{E}(G(w^+))||_2^2]$$

Regularizer controlled by E and λ_2

$$\mathcal{L}_{\text{reconst}}(x, x^*) = \mathcal{L}_{\text{LPIPS}}(x, x^*) + \lambda_3 ||x - x^*||_2^2$$

Test time optimization

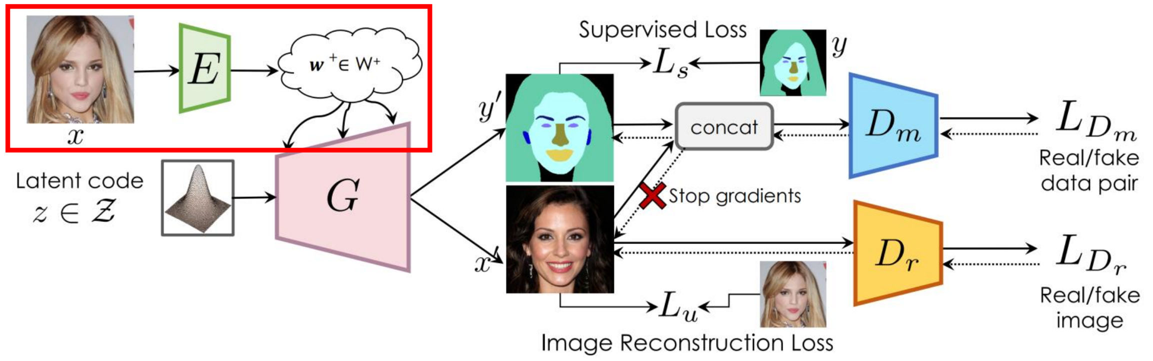


Figure 2: **Model Overview.** Generator G and discriminators D_m and D_r are trained with adversarial objectives $\mathcal{L}_G$ (not indicated here), $\mathcal{L}_{D_m}$ and $\mathcal{L}_{D_r}$. We do not backpropagate gradients from D_m into the generator's image synthesis branch. We train an additional encoder E in a supervised fashion using image and mask reconstruction losses $\mathcal{L}_u$ and $\mathcal{L}_s$.

Inference

- Pass w^{+*} back to the generator to get $G(w^{+*}) = (x^{inv}, y^{inv})$.
- If we have $x^{inv} \approx x^*$, **as the generator was trained to align synthesized labels and images**, we can expect y^{inv} to be correct label of the reconstructed image x^{inv}. Then $y^{inv} \approx y^*$.

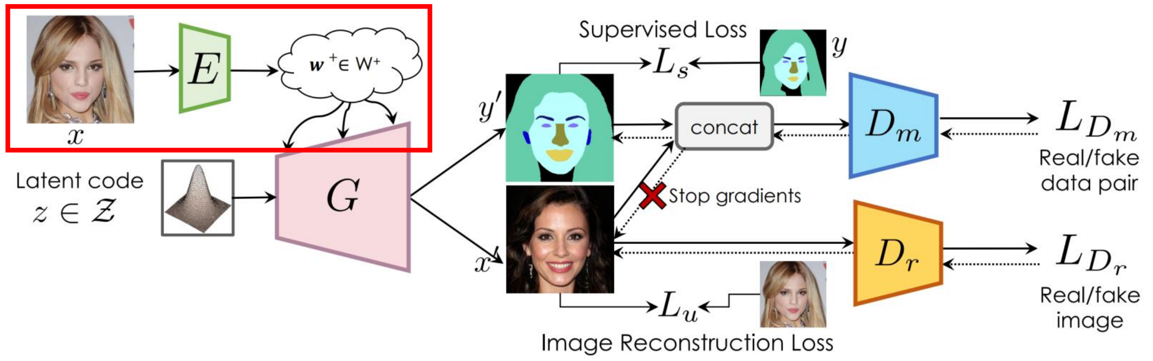


Figure 2: **Model Overview.** Generator G and discriminators D_m and D_r are trained with adversarial objectives $\mathcal{L}_G$ (not indicated here), $\mathcal{L}_{D_m}$ and $\mathcal{L}_{D_r}$. We do not backpropagate gradients from D_m into the generator's image synthesis branch. We train an additional encoder E in a supervised fashion using image and mask reconstruction losses $\mathcal{L}_u$ and $\mathcal{L}_s$.

Qualitative analysis of test time optimization (Appendix 1.).

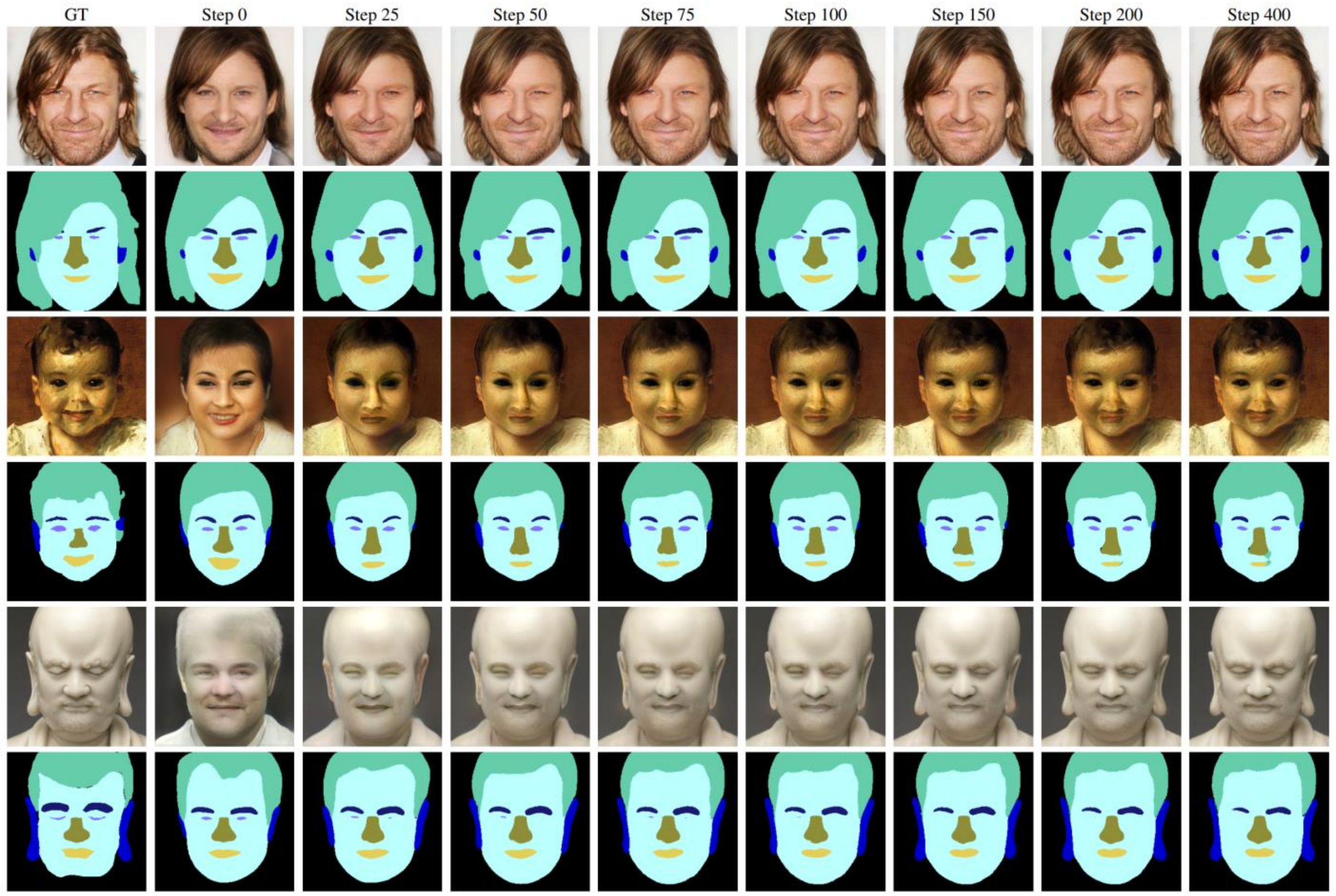


Figure 1: **Face Parts Segmentation Optimization Results.** Image reconstructions and segmentation label predictions at different steps during the optimization process. Step 0 corresponds to using the latent code predicted by the encoder without any further optimization. The model was trained on CelebA-Mask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

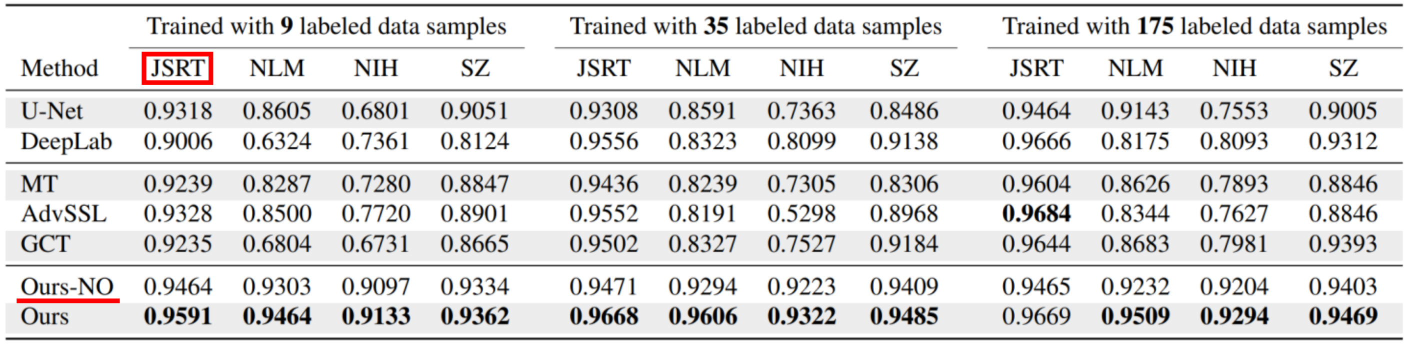
Supplementary: https://nv-tlabs.github.io/semanticGAN/resources/SemanticGAN_supp.pdf

Experiments

- **SemanticGAN is limited by the expressivity of the generative model.**
- GANs have achieved outstanding synthesis quality for “unimodal” data such as images of faces.
- However, current generative models cannot model highly complex data such as images of vivid outdoor scenes (e.g., COCO, ADE20K).
- $\Rightarrow$ Datasets: Chest X-ray segmentation, skin lesion segmentation, and face part segmentation.

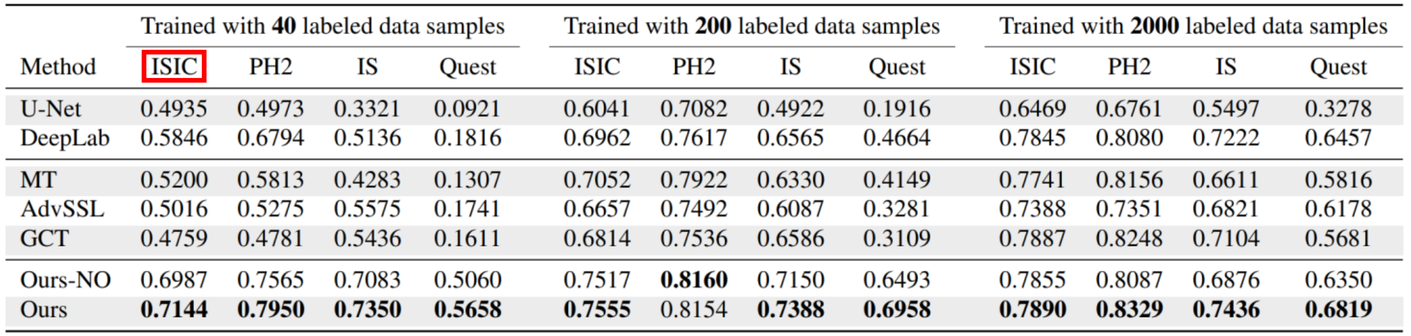
Quantitative Results

Ours w/o optimization



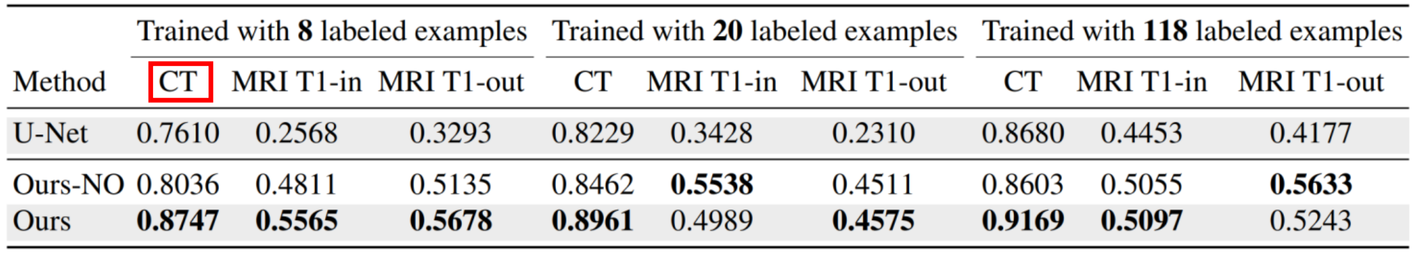
Method	Trained with **9** labeled data samples				Trained with **35** labeled data samples				Trained with **175** labeled data samples			
	JSRT	NLM	NIH	SZ	JSRT	NLM	NIH	SZ	JSRT	NLM	NIH	SZ
U-Net	0.9318	0.8605	0.6801	0.9051	0.9308	0.8591	0.7363	0.8486	0.9464	0.9143	0.7553	0.9005
DeepLab	0.9006	0.6324	0.7361	0.8124	0.9556	0.8323	0.8099	0.9138	0.9666	0.8175	0.8093	0.9312
MT	0.9239	0.8287	0.7280	0.8847	0.9436	0.8239	0.7305	0.8306	0.9604	0.8626	0.7893	0.8846
AdvSSL	0.9328	0.8500	0.7720	0.8901	0.9552	0.8191	0.5298	0.8968	**0.9684**	0.8344	0.7627	0.8846
GCT	0.9235	0.6804	0.6731	0.8665	0.9502	0.8327	0.7527	0.9184	0.9644	0.8683	0.7981	0.9393
Ours-NO	0.9464	0.9303	0.9097	0.9334	0.9471	0.9294	0.9223	0.9409	0.9465	0.9232	0.9204	0.9403
Ours	**0.9591**	**0.9464**	**0.9133**	**0.9362**	**0.9668**	**0.9606**	**0.9322**	**0.9485**	0.9669	**0.9509**	**0.9294**	**0.9469**

Table 1: **Chest X-ray Lung Segmentation.** Numbers are DICE scores. CXR14 [90] JSRT [82]are the in-domain data set, on which we both train and evaluate. We also evaluate on additional out-of-domain datasets (NLM [39], NIH [86], SZ [39], details in supplemental material). Ours as well as the semi-supervised methods use additional 108k unlabeled data samples.



Method	Trained with **40** labeled data samples				Trained with **200** labeled data samples				Trained with **2000** labeled data samples			
	ISIC	PH2	IS	Quest	ISIC	PH2	IS	Quest	ISIC	PH2	IS	Quest
U-Net	0.4935	0.4973	0.3321	0.0921	0.6041	0.7082	0.4922	0.1916	0.6469	0.6761	0.5497	0.3278
DeepLab	0.5846	0.6794	0.5136	0.1816	0.6962	0.7617	0.6565	0.4664	0.7845	0.8080	0.7222	0.6457
MT	0.5200	0.5813	0.4283	0.1307	0.7052	0.7922	0.6330	0.4149	0.7741	0.8156	0.6611	0.5816
AdvSSL	0.5016	0.5275	0.5575	0.1741	0.6657	0.7492	0.6087	0.3281	0.7388	0.7351	0.6821	0.6178
GCT	0.4759	0.4781	0.5436	0.1611	0.6814	0.7536	0.6586	0.3109	0.7887	0.8248	0.7104	0.5681
Ours-NO	0.6987	0.7565	0.7083	0.5060	0.7517	**0.8160**	0.7150	0.6493	0.7855	0.8087	0.6876	0.6350
Ours	**0.7144**	**0.7950**	**0.7350**	**0.5658**	**0.7555**	0.8154	**0.7388**	**0.6958**	**0.7890**	**0.8329**	**0.7436**	**0.6819**

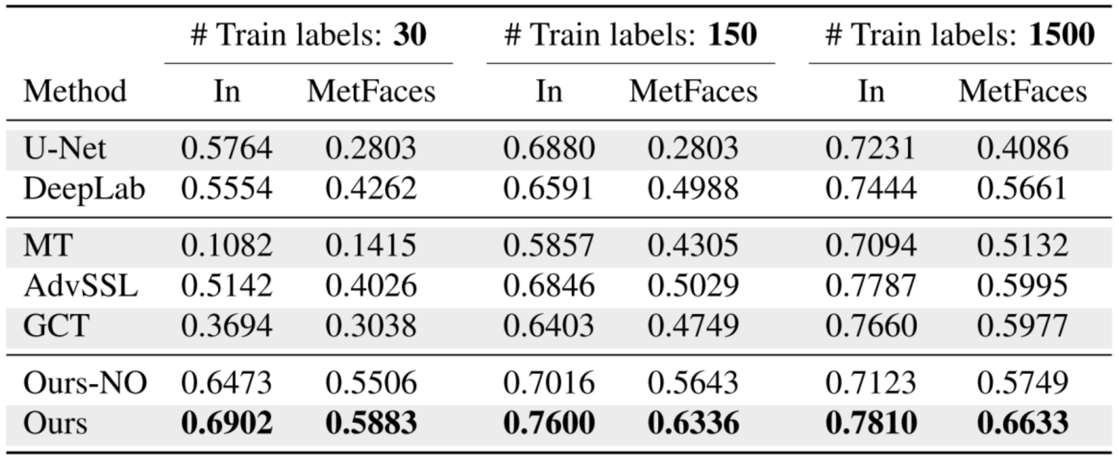
Table 2: **Skin Lesion Segmentation.** Numbers are JC index. Here, ISIC [14] is the in-domain data set, on which we train and also evaluate. Additionally, we perform segmentation on three out-of-domain datasets (PH2 [60], IS [25], Quest [25], details in supplemental material). Ours as well as the semi-supervised methods use additional ≈33k unlabeled data samples.



Method	Trained with **8** labeled examples			Trained with **20** labeled examples			Trained with **118** labeled examples		
	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
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 3: **CT-MRI Transfer Liver Segmentation.** Numbers are DICE per patient. Here, CT is the in-domain data set. We evaluate on unseen MRI data [46] for liver segmentation task. Ours uses additional 70 volumes from LITS2017 [9] testing set as unlabeled data samples.

Quantitative Results



	# Train labels: **30**		# Train labels: **150**		# Train labels: **1500**	
Method	In	MetFaces	In	MetFaces	In	MetFaces
U-Net	0.5764	0.2803	0.6880	0.2803	0.7231	0.4086
DeepLab	0.5554	0.4262	0.6591	0.4988	0.7444	0.5661
MT	0.1082	0.1415	0.5857	0.4305	0.7094	0.5132
AdvSSL	0.5142	0.4026	0.6846	0.5029	0.7787	0.5995
GCT	0.3694	0.3038	0.6403	0.4749	0.7660	0.5977
Ours-NO	0.6473	0.5506	0.7016	0.5643	0.7123	0.5749
Ours	**0.6902**	**0.5883**	**0.7600**	**0.6336**	**0.7810**	**0.6633**

Table 4: **Face Part Segmentation.** Numbers are mIoU. We train on CelebA and evaluate on CelebA as well as the MetFaces dataset. "# Train labels" denotes the number of annotated examples used during training. Our model as well as the semi-supervised baselines additionally use 28k unlabeled CelebA data samples.

Qualitative Results

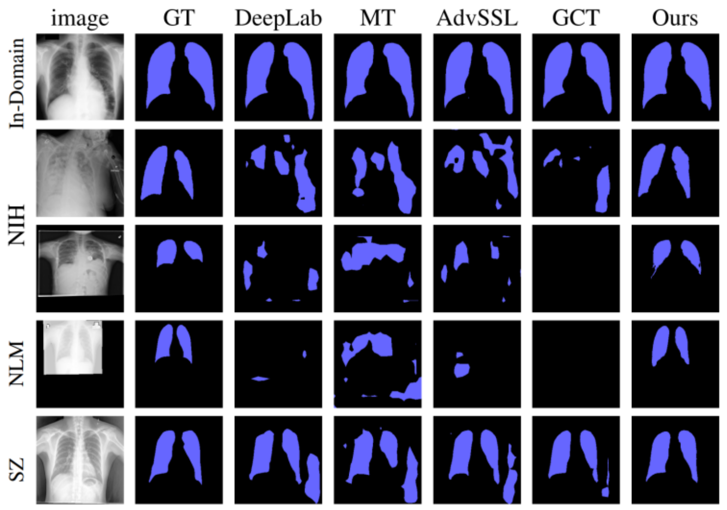


Figure 5: **Chest X-ray Segmentation.** Qualitative examples for both in-domain and out-of-domain datasets.

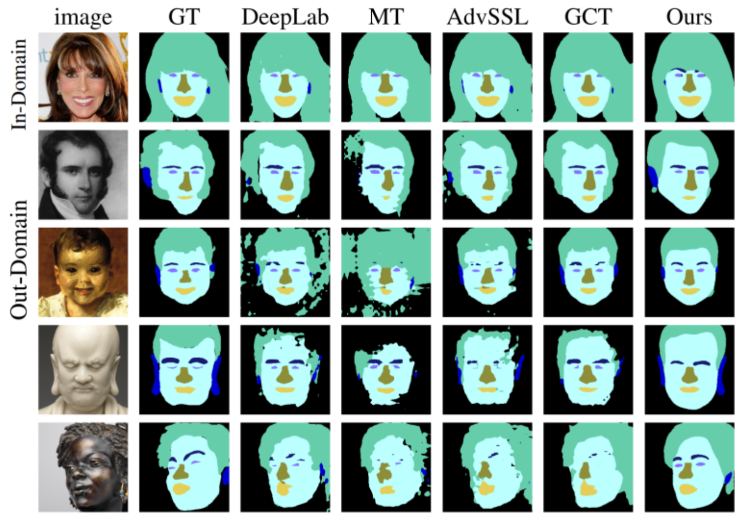


Figure 6: **Face Parts Segmentation.** Qualitative examples for both in-domain and out-of-domain datasets.

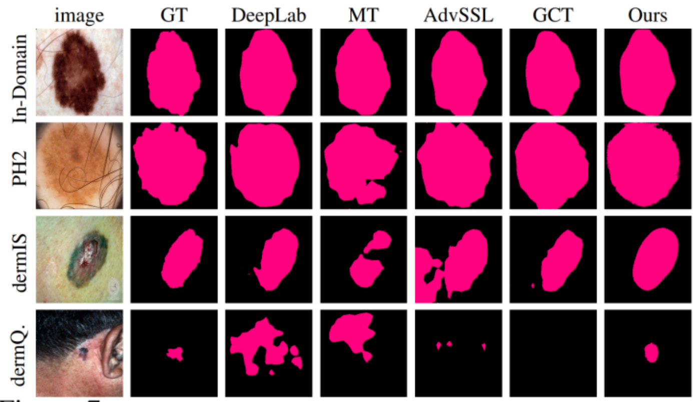


Figure 7: **Skin Lesion Segmentation.** Qualitative examples for both in-domain and out-of-domain datasets.

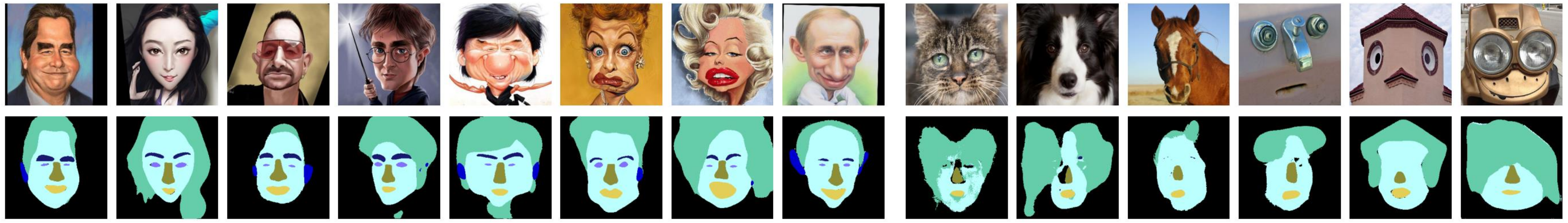
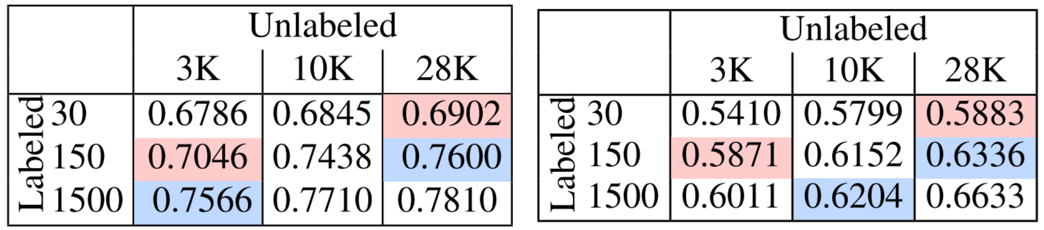


Figure 8: **Extreme Out-Of-Domain Segmentation.** Results on images with a large visual gap to CelebA, on which our model was trained.

Labeled Data vs. Unlabeled Data

- Traditionally, labeled data is considered more valuable than unlabeled data **but how big is the difference?**
- They show that the performance for different amount of labeled and unlabeled data during training.



	Unlabeled		
Labeled	3K	10K	28K
30	0.6786	0.6845	0.6902
150	0.7046	0.7438	0.7600
1500	0.7566	0.7710	0.7810

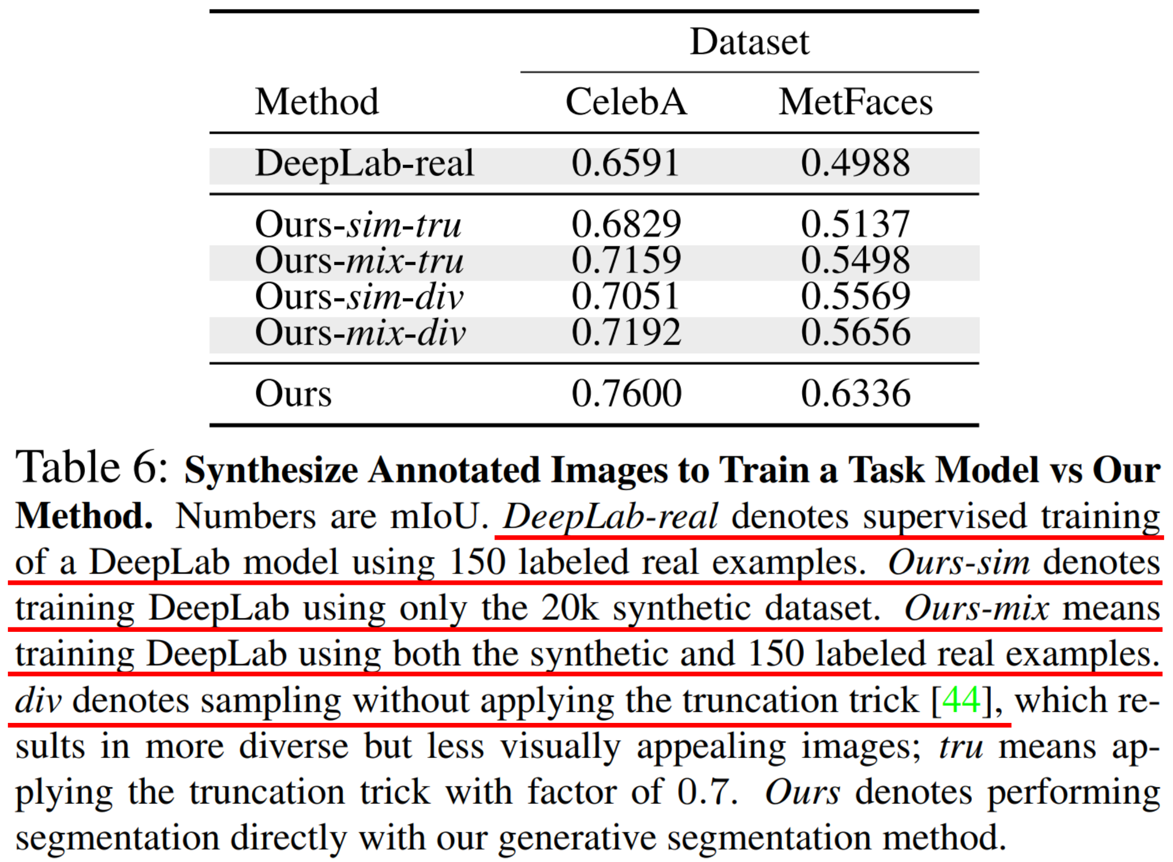
(a) CelebA-Mask (In-Domain)

	Unlabeled		
Labeled	3K	10K	28K
30	0.5410	0.5799	0.5883
150	0.5871	0.6152	0.6336
1500	0.6011	0.6204	0.6633

(b) MetFaces-40 (Out-Domain)

Table 5: **Ablation Study on Number of Labeled vs Unlabeled Examples.** Numbers are mIoU. Entries marked with red or blue color roughly correspond to each other, *i.e.* 30 labeled and 28k unlabeled results in similar performance as 150 labeled and 3k unlabeled examples.

Quality of Synthetic Data



Method	Dataset	
	CelebA	MetFaces
DeepLab-real	0.6591	0.4988
Ours-*sim-tru*	0.6829	0.5137
Ours-*mix-tru*	0.7159	0.5498
Ours-*sim-div*	0.7051	0.5569
Ours-*mix-div*	0.7192	0.5656
Ours	0.7600	0.6336

Table 6: **Synthesize Annotated Images to Train a Task Model vs Our Method.** Numbers are mIoU. *DeepLab-real* denotes supervised training of a DeepLab model using 150 labeled real examples. *Ours-sim* denotes training DeepLab using only the 20k synthetic dataset. *Ours-mix* means training DeepLab using both the synthetic and 150 labeled real examples. *div* denotes sampling without applying the truncation trick [44], which results in more diverse but less visually appealing images; *tru* means applying the truncation trick with factor of 0.7. *Ours* denotes performing segmentation directly with our generative segmentation method.