Supplementary Document

Linear Semantics in Generative Adversarial Networks

A. Definition of IoU.

Intersection-over-Union (IoU) is a widely used metric in semantic segmentation literature. A segmentation of a category is represented as a set of pixels among all pixels that belong to this category. Suppose we have a segmentation A and B, their IoU is $\text{IoU}(A,B) = \frac{|A \cap B|}{|A \cup B|}$. Taking the average across a set of segmentations $\mathcal{A} = \{A_i\}$ and $\mathcal{B} = \{B_i\}$, we get the average IoU on this dataset:

$$IoU(\mathcal{A},\mathcal{B}) = \frac{1}{N} \sum_{A_i \cup B_i \neq \emptyset} IoU(A_i, B_i) \tag{4}$$

IoU evaluates how well the segmentation is for a particular category. Mean IoU (mIoU) evaluates the overall performance of multi-class segmentation. It is calculated as the mean of IoUs over all categories.

B. Proof of commutative property

Now we are going to prove that $\mathsf{u}^{\uparrow}(\mathbf{T}_i \cdot \mathbf{x}_i) = \mathbf{T}_i \cdot \mathsf{u}^{\uparrow}(\mathbf{x}_i)$.

Suppose that a pixel p that we want to interpolate lies in the rectangle of four pixels $(x_{11}, x_{12}, x_{21}, x_{22})$ and its relative position is described by (α, β) as distance ratio to the edges of the rectangle. The interpolated value is

$$\begin{aligned} &\mathsf{u}_p^{\uparrow}(x_{11}, x_{12}, x_{21}, x_{22}, \alpha, \beta) \\ =&(1-\beta)[(1-\alpha)x_{11} + \alpha x_{12}] \\ &+ \beta[(1-\alpha)x_{21} + \alpha x_{22}] \end{aligned} \tag{5}$$

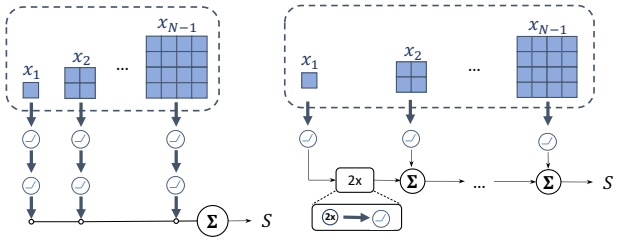
When we do convolution then upsample, we get the following result

$$\begin{aligned} &\mathsf{u}_p^{\uparrow}(\mathbf{T}_i x_{11}, \mathbf{T}_i x_{12}, \mathbf{T}_i x_{21}, \mathbf{T}_i x_{22}, \alpha, \beta) \\ =&(1-\beta)[(1-\alpha)\mathbf{T}_i x_{11} + \alpha \mathbf{T}_i x_{12}] \\ &+ \beta[(1-\alpha)\mathbf{T}_i x_{21} + \alpha \mathbf{T}_i x_{22}] \\ =&\mathbf{T}_i(1-\beta)[(1-\alpha)x_{11} + \alpha x_{12}] \\ &+ \mathbf{T}_i \beta[(1-\alpha)x_{21} + \alpha x_{22}] \\ =&\mathbf{T}_i \mathsf{u}_p^{\uparrow}(x_{11}, x_{12}, x_{21}, x_{22}, \alpha, \beta) \end{aligned} \tag{6}$$

which is exactly upsampling then convoluting.

C. Nonlinear semantic extractors

The architectures of NSEs are shown in Fig. 7. NSE-1 is a direction generalization from LSE. Instead of extracting semantics from each layer linearly, NSE-1 extract semantics with 3 nonlinear convolution layers from each layer. The results from each layer are upsampled and summed up the same as LSE. The architecture of NSE-2 is inspired by DCGAN, where the resolution gradually increases. The output from the last layer of NSE-2 is upsampled and summed with embedding extracted from GAN's feature maps.



(a) NSE-1 (b) NSE-2

Figure 7. The architecture of NSEs. The thick blue arrow refers to 3×3 convolution with stride 1. "2x" refers to nearest upsampling with factor 2.

D. Experiment details

Pretrained networks For segmentation on facial images, we train a UNet [29] on CelebAMask-HQ [24] dataset to perform semantic segmentation. The training script is adapted from the project repo [3] of MaskGAN [24]. Our UNet follows standard UNet architecture and takes 512×512 images as input. It is trained using Adam optimizer [23] for 40 epochs (about 76k iterations), with learning rate 3×10^{-4}, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and batch size 16.

The CelebAMask-HQ dataset contains 30k human-labeled face-segmentation pairs. The face images are aligned to the center and have 1024×1024 resolution. The semantic labeling's resolution is 512×512, consisting of 19 semantic categories. However, there are duplicate semantic concepts like "right ear" and "left ear", "right eye" and "left eye", "right brow" and "left brow". In those pairs, as both categories differ only in spatial location, we unify them into "ear", "eye", and "brow". Besides, only 50 instances are labeled with "necklace", thus we remove it by merging "necklace" into "neck". As a result, we get 15 semantic categories listed in category results Table 6.

For segmentation on GANs trained on LSUN's bedroom and church datasets, we use the DeepLabV3 [8] with

[3] https://github.com/switchablenorms/CelebAMask-HQ

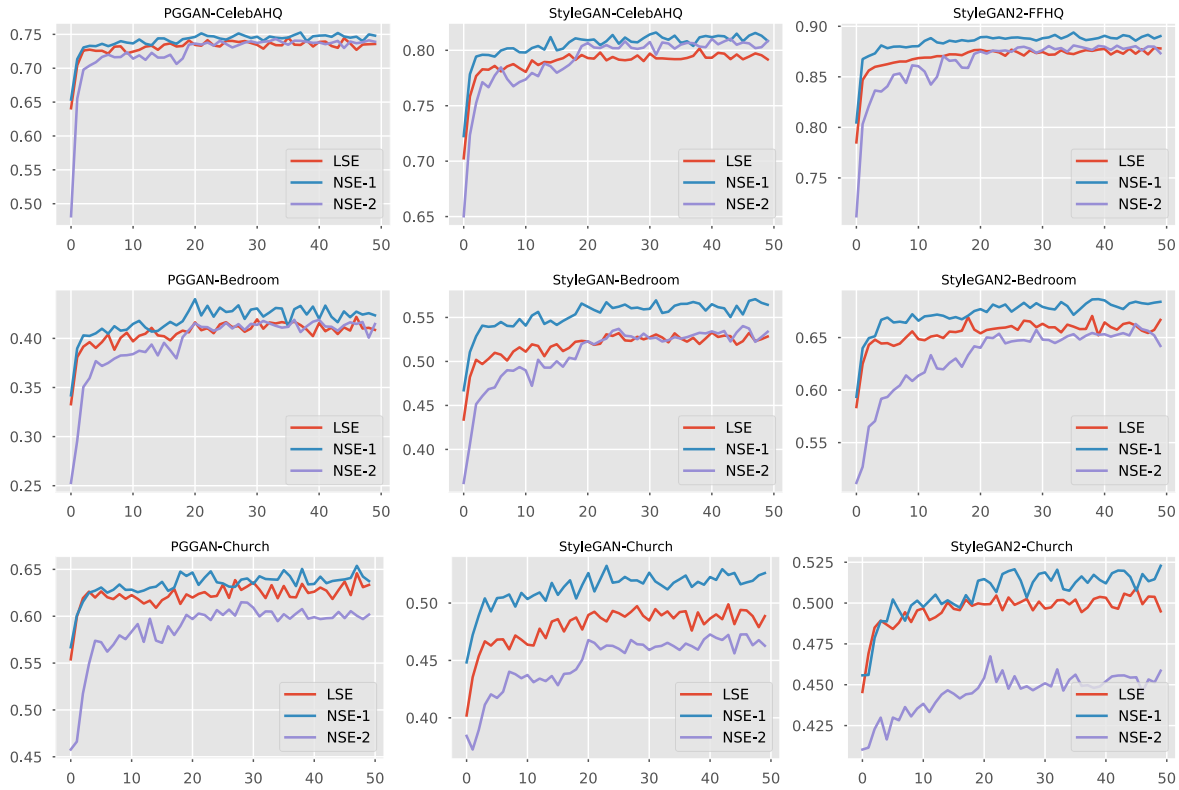


Figure 8. Training evolution of mIoU of all the semantic extractors on GANs.

ResNeSt [39] backbone trained on ADE20k dataset [42]. Model parameters are obtained from here[4]. However, the DeepLabV3 predicts in total 150 categories, where most are not present in generated images, because GANs are train on LSUN datasets rather than the ADE20k dataset. We apply a category selection process (detailed in Appx. E) to remove irrelevant semantic categories.

All the pretrained GAN models are adapted from GenForce[5]. The image resolution of GANs trained on face datasets are 1024×1024, and the rest are 256×256.

Training For fully supervised training, we sample 51,200 images from the GAN and record their feature maps. These images are then semantically segmented with an off-the-shelf segmenter in the corresponding data domain. The semantic masks and feature maps are then used to train the transformation matrix $\mathbf{T}_i$ for every GAN layer. To be specific, the total matrix $\mathbf{T}$ (defined in (2)) for StyleGAN2-FFHQ are of size 15×5568. For StyleGAN2-Bedroom, $\mathbf{T}$ is shaped as 16×5376.

$\mathbf{T}_i$ are optimized with Adam [23] with $\beta_1 = 0.9$, $\beta_2 = 0.999$ and initial learning rate 10^{-3}. The training takes 50 epochs in total, where each epoch consists of 1,024 samples. The learning rate is reduced with a factor of 10 at epoch 20. For the first two epochs, the batch size is 1. For the next 16 epochs (3 to 19), the batch size is set to 4. For epoch 20 to 50, the batch size is 64. The total optimization iterations are $1024 \times 2 + \frac{1024}{4} \times 16 + \frac{1024}{64} \times 32 = 6,656$. LSE, NSE-1, and NSE-2 are trained in the same settings.

We record the mIoU of training samples, and show the evolution of training mIoU in Fig. 8. All the semantic extractors converge during the training.

For the few-shot training of LSEs, we also sample the latent space and segment the images. The difference is that only a few annotations are made available to the LSE. We experimented with 1, 4, 8, 16 samples. The resultant models are named as the one-shot, 4-shot, 8-shot and 16-shot LSEs, respectively. For the one-shot LSE, the training takes 2000 iterations with batch size 1. For 4, 8, and 16 samples, the training uses batch sizes 4, 8, and 16 and iteration numbers 2000, 1000, and 500, respectively. For PGGAN, each batch is exactly the same. For StyleGAN and StyleGAN2, the layer noises are re-sampled in each batch. The optimizer setting is the same as in full supervision.

Evaluation Conventionally, semantic segmentation methods are evaluated on real image-segmentation datasets. However, our semantic extractors cannot take real images as input. One may invert real images in GAN's representation, but the inversion is another challenging problem, thus we

[4]https://github.com/zhanghang1989/PyTorch-Encoding
[5]https://github.com/genforce/genforce

Generator	PGGAN		StyleGAN		StyleGAN2	
Dataset	Bedroom	Church	Bedroom	Church	Bedroom	Church
LSE	32.4 (-4.8)	49.4 (-2.4)	39.8 (-8.0)	34.8 (-7.0)	54.3 (-4.2)	36.8 (-3.7)
NSE-1	**34.1**	**50.6**	**43.2**	**37.4**	**56.6**	**38.2**
NSE-2	28.9 (-15.1)	45.1 (-10.8)	39.5 (-8.7)	33.3 (-11.0)	51.9 (-8.4)	34.5 (-9.8)
Results adaped from Table 1.						
LSE	33.2 (-3.2)	51.3 (-3.2)	39.9 (-7.8)	35.4 (-6.3)	53.9 (-3.4)	37.7 (-2.6)
NSE-1	**34.3**	**53.0**	**43.3**	**37.8**	**55.8**	**38.7**
NSE-2	30.7 (-10.5)	49.5 (-6.6)	38.9 (-10.2)	34.0 (-10.1)	52.1 (-6.8)	35.3 (-8.8)

Table 4. The evaluation of LSE, NSE-1, and NSE-2 trained on the full list of ADE20K 150 classes. The mIoU(%) is calculated on the final selected categories, which is the same as the the categories used in the paper. The results of models re-trained on the selected categories are also shown in the last three rows for reference.

do not consider this approach. As a result, the evaluation cannot be conducted on the common annotated dataset. Ideally, we should annotate synthetic images manually, but the cost would then be prohibitive. Therefore, we choose to use the prediction from the off-the-shelf segmenter as the ground-truth for evaluation.

We sample and segment another 10,000 images different from those used in training. Every time GAN generates an image, we apply the semantic extractor to the generator's feature maps to predict a semantic mask. The segmentation is compared with the pretrained segmenter's prediction to compute the IoU.

As some datasets (e.g., LSUN's bedroom dataset) may be more difficult to segment than some others (e.g., the CelebAHQ dataset), we compute relative performance differences between semantic extractors. Concretely, for each GAN model, there are three semantic extractors to be evaluated, which are LSE, NSE-1, and NSE-2. Denoting their mIoUs with the pretrained segmenter as y_i, and the highest mIoU among the three as y^*, the relative performance difference of each semantic extractor is defined as $\frac{y_i - y^*}{y^*}$.

E. Category selection

For GANs trained on bedroom and church images, we rely on DeepLabV3 trained on ADE20K to provide the training supervision. In this section, we aim to remove categories that are not generated by GANs.

First of all, we train and evaluate semantic extractors on the full set of 150 classes. Then, we remove all the categories that are predicted with mIoU $< 10\%$ by all the semantic extractors. In other words, a category will be selected as long as any of the LSE, NSE-1, and NSE-2 predicts it with mIoU $> 10\%$. In this way, the list of selected categories for each GAN model are decided. The formal results are obtained by training and evaluating on the selected categories under the same settings.

The evaluation of LSE, NSE-1, and NSE-2 trained on full categories is shown in Table 4. Generally, the re-trained semantic extractors obtain slightly better performance, which is expected. We also show the IoU for each category in Table 5, where the table headers also list the final selected categories for each GAN model.

F. Cosine similarity of categories

As mentioned in Sec. 3.1, the linear formulation indicates that the features of a particular category can be bounded by a hyper-cone. To verify this geometric intuition, we propose to test a stronger hypothesis: the features of different categories are well-separated. In other words, the distances of features within a category are closer than those between different cateogories. Our approach is to sample features for each category fairly, and compute the cosine distances between features.

First of all, we need to ensure the fairness of comparison for each category. For this purpose, we propose a fair sampling algorithm (Algorithm 1) which repeatedly samples images and record features fairly until enough features are collected. In every image, if the feature number of a category is larger than a threshold T_1, then T_1 feature vectors from that category are chosen randomly without replacement (denoted by $choice(a, N)$). The chosen vectors would be accumulated to a category feature pool until the feature

Algorithm 1: Fair feature sampling algorithm.

Input: G; P; T_1; T_2
Output: $\{f_k\}$
for $k = 1, 2, \ldots, M$ **do**
  $f_k = \emptyset$
while $\exists k, |f_k| < T_2$ **do**
  $z \sim \mathcal{N}(0, I)$
  $I, F = G(z)$ // F denotes features
  $S = P(I)$
  for $k = 1, 2, \ldots, M$ **do**
    if $|f_k| < T_2$ *and* $|\{p|S_p = k\}| \geq T_1$ **then**
      $R = choice(\{p|S_p = k\}, T_1)$
      $f_k = f_k \cup \{F_p|p \in R\}$

	wall	floor	ceiling	bed	win.	table	curtain	painting	lamp	cushion	pillow	flower	light	chdr.	fan	clock
LSE	91.29	85.56	88.47	90.53	75.58	67.19	43.13	68.79	59.43	32.29	46.05	12.98	36.73	18.06	37.79	14.77
NSE-1	92.13	87.06	89.79	91.99	76.40	69.98	46.14	73.71	62.70	34.21	48.10	15.76	40.04	20.34	45.62	12.39
NSE-2	91.56	86.22	88.78	91.15	72.94	67.52	42.77	69.43	58.61	30.87	46.20	4.76	26.75	12.51	35.14	5.39
LSE	91.29	85.95	88.27	90.94	76.23	67.31	42.08	69.29	59.28	31.17	45.57	11.77	36.17	18.07	35.72	13.62
NSE-1	92.21	87.12	89.05	91.80	76.37	69.62	45.74	71.69	62.34	33.22	47.98	12.40	39.10	18.57	43.28	12.74
NSE-2	91.66	86.31	88.65	91.21	74.79	68.85	43.66	70.70	60.51	26.36	45.34	4.01	31.83	11.22	30.61	7.13

(a) StyleGAN2-Bedroom

	wall	floor	ceiling	bed	win.	table	curtain	chair	painting	rug	wdrb.	lamp	cushion	chest	pillow	flower	light	chdr.	fan
LSE	82.30	74.12	73.79	88.28	55.34	47.58	37.13	9.10	64.01	8.93	11.07	42.91	33.42	19.09	46.51	9.02	11.45	22.70	18.55
NSE-1	83.78	75.50	76.55	89.40	57.24	50.63	40.96	10.52	67.41	11.54	12.31	48.90	36.62	18.66	49.07	10.67	25.08	26.49	29.66
NSE-2	83.05	74.61	75.44	89.34	53.99	47.98	39.63	6.03	64.04	5.11	8.78	45.89	35.95	17.45	48.88	3.02	12.32	16.52	21.49
LSE	83.12	74.70	73.90	88.82	56.80	45.78	37.19	10.24	62.87	9.35	10.81	42.43	33.26	20.91	46.19	7.17	11.01	25.37	18.18
NSE-1	84.37	75.01	76.30	89.90	58.27	49.78	39.98	11.95	66.87	12.77	12.04	49.36	35.88	22.91	48.95	10.32	24.10	25.93	27.83
NSE-2	83.81	74.19	75.38	89.64	55.27	47.43	40.03	11.23	63.60	6.76	9.27	46.61	34.22	15.45	47.84	0.66	8.92	14.58	13.85

(b) StyleGAN-Bedroom

	wall	floor	ceiling	bed	windowpane	table	curtain	painting	lamp	pillow	light	chandelier	fan
LSE	69.46	45.07	54.05	68.39	36.38	12.58	25.77	32.67	17.18	16.10	13.65	12.14	18.06
NSE-1	70.75	46.78	57.01	70.35	38.84	14.78	29.07	35.66	18.35	18.86	15.42	9.49	17.36
NSE-2	68.60	45.19	54.56	68.12	33.03	12.06	27.67	34.59	15.31	16.79	0.11	0.00	0.00
LSE	71.91	47.88	54.53	70.29	37.06	11.71	25.39	33.74	16.82	17.90	15.57	10.83	17.94
NSE-1	72.91	49.01	56.42	71.69	38.90	14.35	28.34	35.39	18.21	19.31	13.58	10.45	17.08
NSE-2	72.11	47.90	54.72	71.41	38.46	13.14	27.77	35.41	16.83	18.65	0.00	1.22	1.30

(c) PGGAN-Bedroom

	building	sky	tree	road	grass	sidewalk	person	earth	plant	car	stairs
LSE	85.94	97.52	76.51	24.19	40.16	16.71	15.78	13.72	8.92	12.22	13.43
NSE-1	86.93	97.87	78.59	25.76	44.45	17.73	17.03	14.04	10.62	13.30	14.39
NSE-2	86.65	97.71	77.96	22.16	37.87	10.78	12.92	8.87	7.52	8.21	8.91
LSE	87.96	97.46	76.31	27.32	41.61	17.08	17.62	14.21	8.28	12.84	14.23
NSE-1	88.75	97.70	78.16	27.14	44.96	18.82	16.76	15.92	9.99	12.68	15.26
NSE-2	88.77	97.69	77.66	22.33	39.78	13.38	12.52	10.80	6.31	8.58	10.81

(d) StyleGAN2-Church

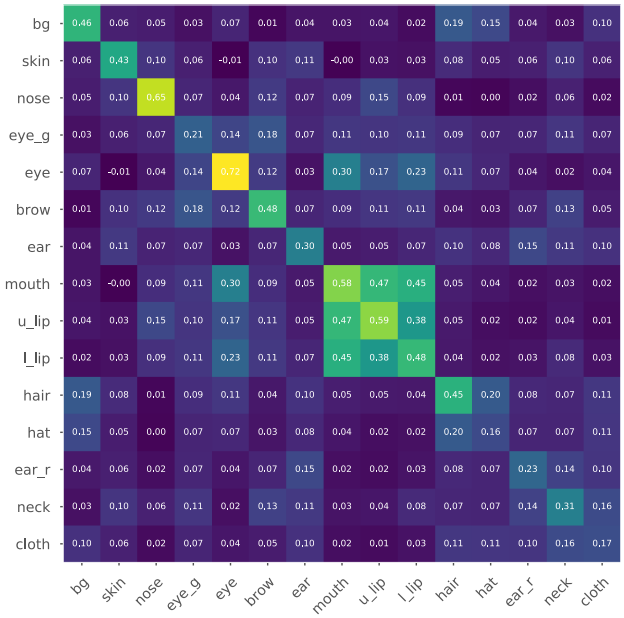
	building	sky	tree	road	grass	sidewalk	person	plant	signboard	path
LSE	88.18	95.53	49.14	23.29	39.34	11.07	9.42	9.32	14.11	8.52
NSE-1	88.55	95.69	54.25	25.06	42.24	11.05	11.61	12.55	22.53	10.40
NSE-2	87.74	95.34	48.18	19.77	34.36	7.79	10.11	8.51	14.37	6.63
LSE	91.30	95.53	47.46	25.30	41.63	13.08	8.39	9.17	13.80	8.84
NSE-1	92.01	96.01	53.01	28.06	44.84	13.33	11.56	12.25	16.62	10.63
NSE-2	91.58	95.66	49.60	22.96	35.72	7.95	9.47	9.12	12.69	5.40

(e) StyleGAN-Church

	building	sky	tree	road	grass	signboard
LSE	83.98	91.21	45.55	17.01	30.14	28.51
NSE-1	84.60	91.46	47.92	18.79	31.17	29.71
NSE-2	83.79	90.91	42.99	15.19	23.64	14.35
LSE	88.17	91.33	44.57	29.10	34.05	20.78
NSE-1	88.98	92.02	47.18	31.43	36.08	22.37
NSE-2	88.35	91.75	44.85	25.91	30.31	16.00

(f) PGGAN-Church

Table 5. The IoU (%) of each category for LSE, NSE-1, and NSE-2. In every subtable, the first three rows show the results of models trained with full classes during the category selection process. The last three rows of each subtable show the results of the models used in Table 1, which are obtained by re-training on the selected classes. The abbreviation "win.", "wdrb.", "chdr." stands for "windowpane", "wardrobe", and "chandelier", respectively.



(a) StyleGAN-CelebAHQ

(b) StyleGAN2-FFHQ

Figure 9. The cosine similarity between categories. The features for each category are collected using Algorithm 1.

Algorithm 2: Image editing algorithm.

Input: G; L; L_reg; N
Output: latent code z
for $i = 1, \ldots, N$ **do**
  $z_i \leftarrow z_{i-1} + \text{optimizer}(L(z_{i-1}) + L_{\text{reg}}(z_{i-1}))$
$z \leftarrow z_N$

number reach T_2. The algorithm would terminate when all the category feature pools have collected T_2 features. The fair sampling algorithm gauruantees that each category feature pool consists of T_1 randomly chosen vectors from $\frac{T_2}{T_1}$ randomly sampled images. As the sampling procedure is identical for each category, the sampled features are fair for each category. In practice, we choose $T_1 = 200$ and $T_2 = 4000$.

Second, we calculate the cosine similarity between categories using the fairly sampled features. Specifically, we first calculate the pairwise cosine similarity between feature vectors of two pools, resulting in a $T_2 \times T_2$ confusion matrix. The two pools can belong to different categories (inter-class) or the same category (intra-class). The cosine similarity $cos(A, B)$ of category A and B is defined as the mean of the entire matrix.

We show results of StyleGAN-CelebAHQ and StyleGAN2-FFHQ in Fig. 9. Most diagonal elements of the confusion matrix have higher cosine similarity than other elements in a row. It is indicated that the features in a category is more similar to one another than features between different categories.

G. Details of Semantic Image Editing

Algorithm. A general image editing algorithm is shown in Algorithm 2, whose inputs are the generator G, the edit loss L, the optional regularization loss L_{reg}, and total iteration number N.

For color space editing, the editing loss will be the color editing loss L_c, defined as $L_c = \frac{1}{||M||_2^2}||M \odot (G(z_i) - C)||_2^2$, where C is the color stroke, M is the mask of the modified region. For semantic image editing, the editing loss will be the semantic editing loss L_s as defined in Sec. 4.1. The regularization loss is composed by items including the color preservation loss $L_p = \frac{||(1-M)\odot(G(z_i)-G(z_0))||_2^2}{||1-M||_2^2}$, the neighbor regularization loss $L_n = ||z_i - z_0||_2^2$, and the prior regularization loss $L_z = ||z_i||_2^2$. z_i denotes the latent vector for iteration i and z_0 denotes the initial latent vector.

For color space editing, its total loss is $L = L_s + 10^{-3}L_n + 10^{-3}L_z$. For SIE, the total loss is $L = L_c + L_p + 10^{-3}L_n + 10^{-3}L_z$.

Usage. In practice, our image editing application works in two steps: The first step is to annotate 1 to 8 images sampled

from GAN. The backend of the application will then train a few-shot LSE using the annotations. The second step is to edit any sampled images. The editing interface will provide the semantic mask extracted by the few-shot LSE along with the image. When the user wants to edit an image, he draws some strokes on the semantic mask to form a target mask. Then, the backend would run the SIE algorithm and return an image that is closer to the target.

H. Semantic-Conditional Sampling.

Algorithm. To sample an image matching the given mask, we first try to find a good initialization. We randomly sample n_{init} latent codes and select the initialization to be the one closest to the target mask. Next, we iteratively optimize the latent code to match the target mask using the cross-entropy loss defined in (3.1). The semantic masks can be predicted using either a pretrained segmenter or a few-shot LSE.

The SCS algorithm is defined formally in Algorithm 3. Its inputs are the current latent code z, the target semantic segmentation Y, the generator G, the semantic predictor P, the initialization number n_{init}, and the iteration number N. Its output will be image samples that respect the given mask Y.

In practice, we use $n_{\text{init}} = 10$ for SCS on face images. $n_{\text{init}} = 100$ is used for bedroom and church images, as they are much more diverse than faces. The optimization is repeated for 50 iterations. The optimizer is Adam with default hyperparameters (lr=10^{-3}, $\beta_1 = 0.9$, $\beta_2 = 0.999$). These settings are manually selected without tuning.

For SCS on facial images, the target masks are selected randomly from the annotations in the CelebAMask-HQ [24] dataset. For bedroom and church, the masks are predicted from images sampled from truncated latent space, which has better image quality than the full latent space [21]. The truncated latent space $\mathcal{W}^-$ is obtained by truncating the latent vectors of $\mathcal{W}$ within a distance of the statistical center.

Evalution. Our proposed method plugs in a few-shot LSE for P, while the baseline uses a pretrained segmentation network as P. To evaluate the performance of SCS models, we again rely on a pretrained segmentation network, P^*. In this work, the pretrained network used by the baseline is exactly the same as the one used in evaluation. This is slightly biased toward the baseline, yet our method is still able to match or surpass the baseline.

Formally, let the set of targets be $\mathcal{Y}$. The images sampled by a SCS model given a target Y_i are denoted as a set $\mathcal{I}_i$. The semantic agreement A of sampled images can be measured by the mean IoU between the predicted segmentation masks and the target mask:

$$A(\mathcal{I}, \mathcal{Y}; P^*) = \sum_{\substack{1 \le i \le |\mathcal{Y}| \\ 1 \le j \le |\mathcal{I}_i|}} \frac{1}{|\mathcal{I}_i||\mathcal{Y}|} \text{mIoU}(Y_i, P^*(I_{i,j})) \quad (7)$$

In practice, we select 100 target masks and conditionally sample 10 images for each target, i.e., $|\mathcal{Y}| = 100$ and $|\mathcal{I}_i| = 10$. As a result, we obtain 1,000 images for the evaluation of each setting of SCS. To account for the variance of few-shot LSEs, we repeat the training for each model 5 times, as mentioned in Sec. 3.3.

Algorithm 3: Semantic-conditional sampling algorithm.

Input: G; P; Y; n_{init}; N
Output: latent code z
$\bar{z}_i \sim N(0, I)$, $i = 1, \ldots, n_{\text{init}}$
$S_i = P(G(\bar{z}_i))$
$P_i = |\{p | S_{i,p} = Y_p\}|$
$z_0 = \bar{z}_\alpha$, $\alpha = argmin_i P_i$
for $i = 1, \ldots, N$ **do**
  $L = \mathcal{L}(P(G(z_{i-1})), Y)$
  $z_i = \text{optimizer}(L, z_{i-1})$
$z \leftarrow z_N$

I. Layerwise analysis

During this work, we examined layer-wise semantics, which refers to the semantics extracted from each layers alone. As the original training objective (3.1) only optimizes the summation from all layers, the semantics from each layer may not be a good segmentation individually. To extract the layer-wise semantics better, we add a cross-entropy loss term on each layer. Theoretically, the best possible layer-wise semantics should be obtained by training only on that layer. However, the computational cost would then be prohibitive. Thus, we choose to optimize all layer losses together.

To put it more formally, we denote the output of LSE on layer i to be $\mathbf{S}_i = \mathbf{T}_i \cdot \mathbf{x}_i$ (the final segmentation is $\mathbf{S} = \sum_{i=1}^{N-1} \text{u}_i^{\uparrow}(\mathbf{S}_i)$). The training objective becomes

$$\mathcal{L}_l = \mathcal{L}(\mathbf{S}, Y) + \sum_{i=1}^{N-1} \alpha_i \mathcal{L}(\text{u}_i^{\uparrow}(\mathbf{S}_i), Y), \quad (8)$$

where Y is the segmentation label, $\mathcal{L}$ is the standard cross-entropy loss, and α_i is the coefficient for each layer. In practice, we set $\alpha_i = 0.1$. The training procedure is exactly the same.

The visualizations of layer-wise semantics are shown in Fig. 10. Our main discoveries are twofold: **(1)** the semantic layout in each layer becomes refined as the network layer progresses from input to output; **(2)** the most semantically rich layers are often near the middle layer of the network. However, it remains unclear how to make use of the layer-wise semantics and we choose to leave this question for future research.

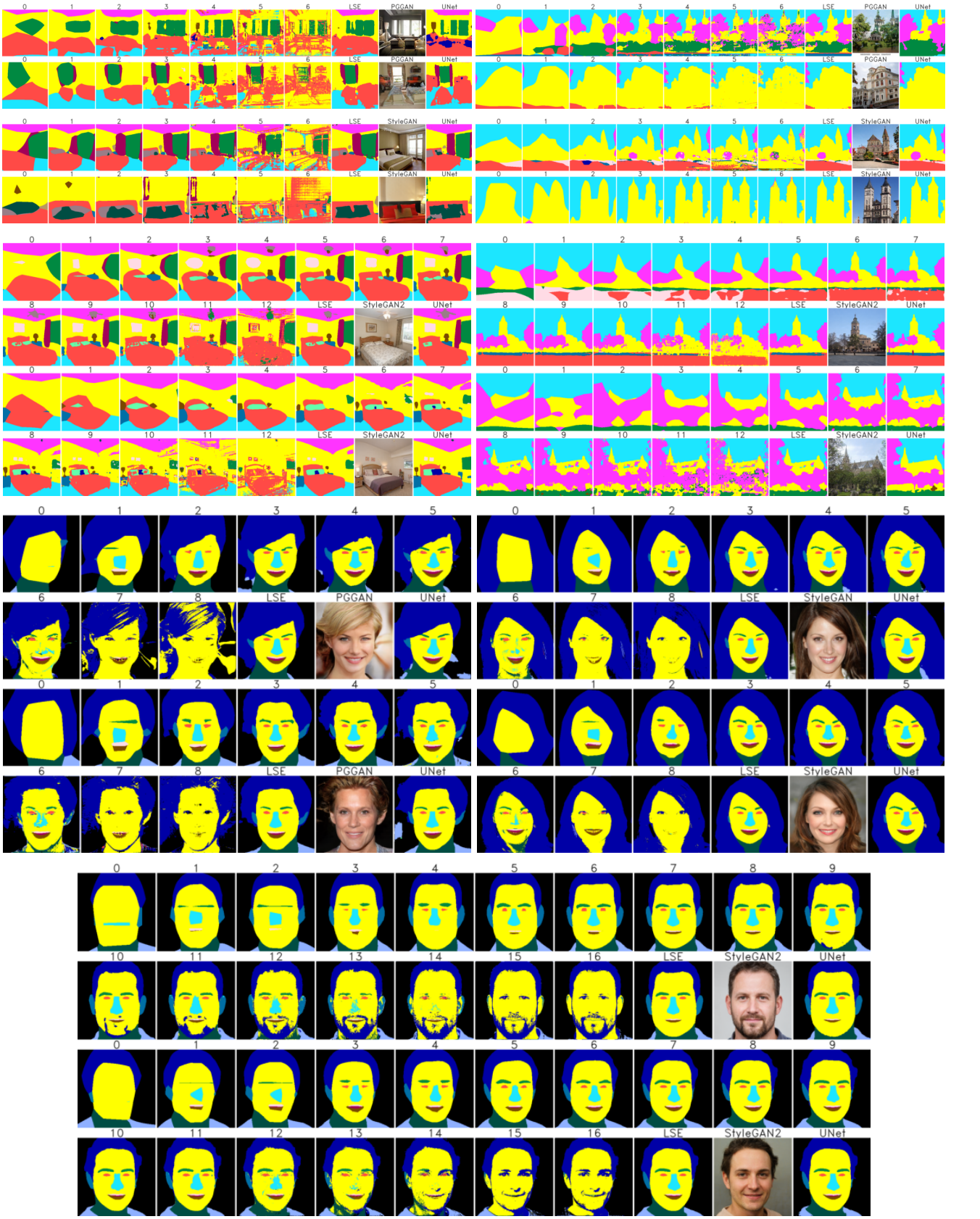


Figure 10. The layer-wise semantics extracted from PGGAN, StyleGAN, and StyleGAN2. Layer indices are shown in the headers.

	skin	nose	eye-g	eye	brow	ear	mouth	u-lip	l-lip	hair	hat	ear-r	neck	cloth
	StyleGAN2-FFHQ													
LSE	95.9%	94.7%	69.9%	91.0%	83.5%	80.5%	84.5%	87.8%	91.2%	92.9%	11.1%	22.8%	91.0%	72.5%
NSE-1	97.0%	95.4%	72.4%	92.1%	88.2%	83.0%	87.4%	91.6%	92.8%	94.2%	12.9%	34.0%	92.9%	75.9%
NSE-2	96.9%	95.3%	73.4%	92.0%	87.7%	82.8%	87.7%	90.9%	92.9%	94.1%	12.9%	28.6%	92.4%	72.8%
	StyleGAN-CelebAHQ													
LSE	93.9%	91.3%	25.7%	86.2%	75.9%	63.5%	75.6%	81.1%	85.4%	87.5%	0.0%	13.1%	84.5%	35.9%
NSE-1	95.8%	93.6%	22.8%	89.3%	83.2%	69.4%	78.8%	87.4%	88.7%	90.8%	0.3%	21.3%	88.0%	41.2%
NSE-2	96.0%	94.1%	22.1%	89.4%	84.7%	69.7%	79.0%	88.0%	89.5%	90.9%	0.0%	19.0%	87.8%	39.2%
	PGGAN-CelebAHQ													
LSE	92.7%	89.4%	19.7%	84.9%	71.7%	61.9%	72.4%	81.4%	84.7%	85.2%	5.0%	16.1%	79.8%	34.1%
NSE-1	93.8%	90.9%	22.0%	86.3%	78.4%	63.0%	71.6%	83.0%	85.6%	86.4%	6.3%	20.3%	81.5%	37.0%
NSE-2	94.1%	92.0%	20.8%	86.2%	78.9%	64.4%	73.0%	83.9%	86.4%	86.9%	6.2%	21.3%	82.2%	37.4%

Table 6. The IoU for each category (excluding background category) of LSE, NSE-1 and NSE-2. The groundtruth used in the IoU computation is obtained from UNet.

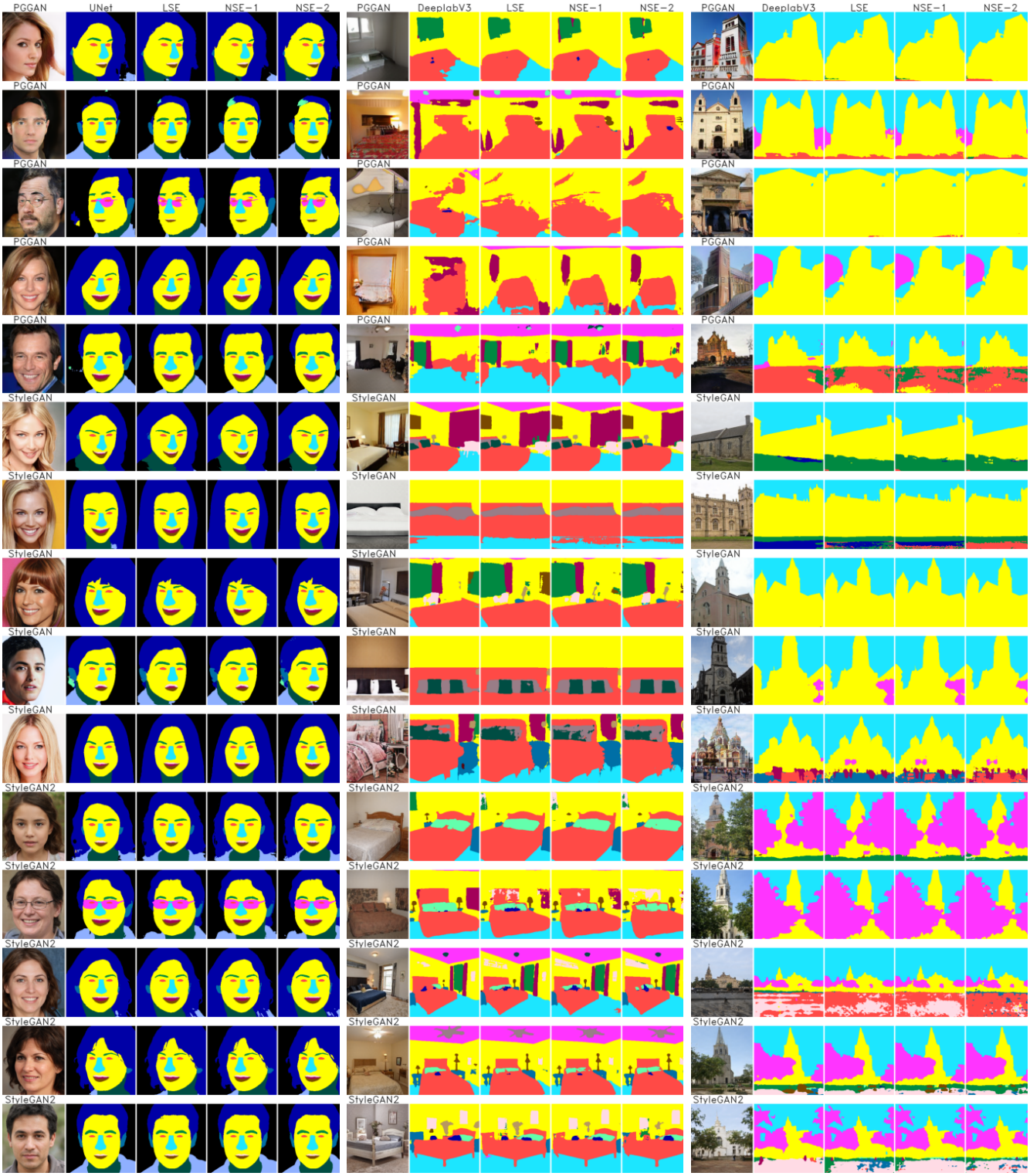
J. Supplementary results

Additional qualitative results comparing LSEs and NSEs are shown in Fig. 11.

Category IoUs for bedroom and church models are summarized in Table 5. They are shown together with category IoUs from models trained with full ADE20K categories. For face GANs, the results are shown in Table 6.

More results for Semantic Image Editing are shown in Fig. 12.

We present supplementary results for Semantic Conditional Sampling on facial images (Fig. 13), bedroom images(Fig. 14) and church images(Fig. 15).



(a) Face datasets. (b) LSUN-bedroom dataset. (c) LSUN-church dataset.

Figure 11. Qualitative comparisions of LSEs and NSEs. For each GAN, 5 samples are shown.

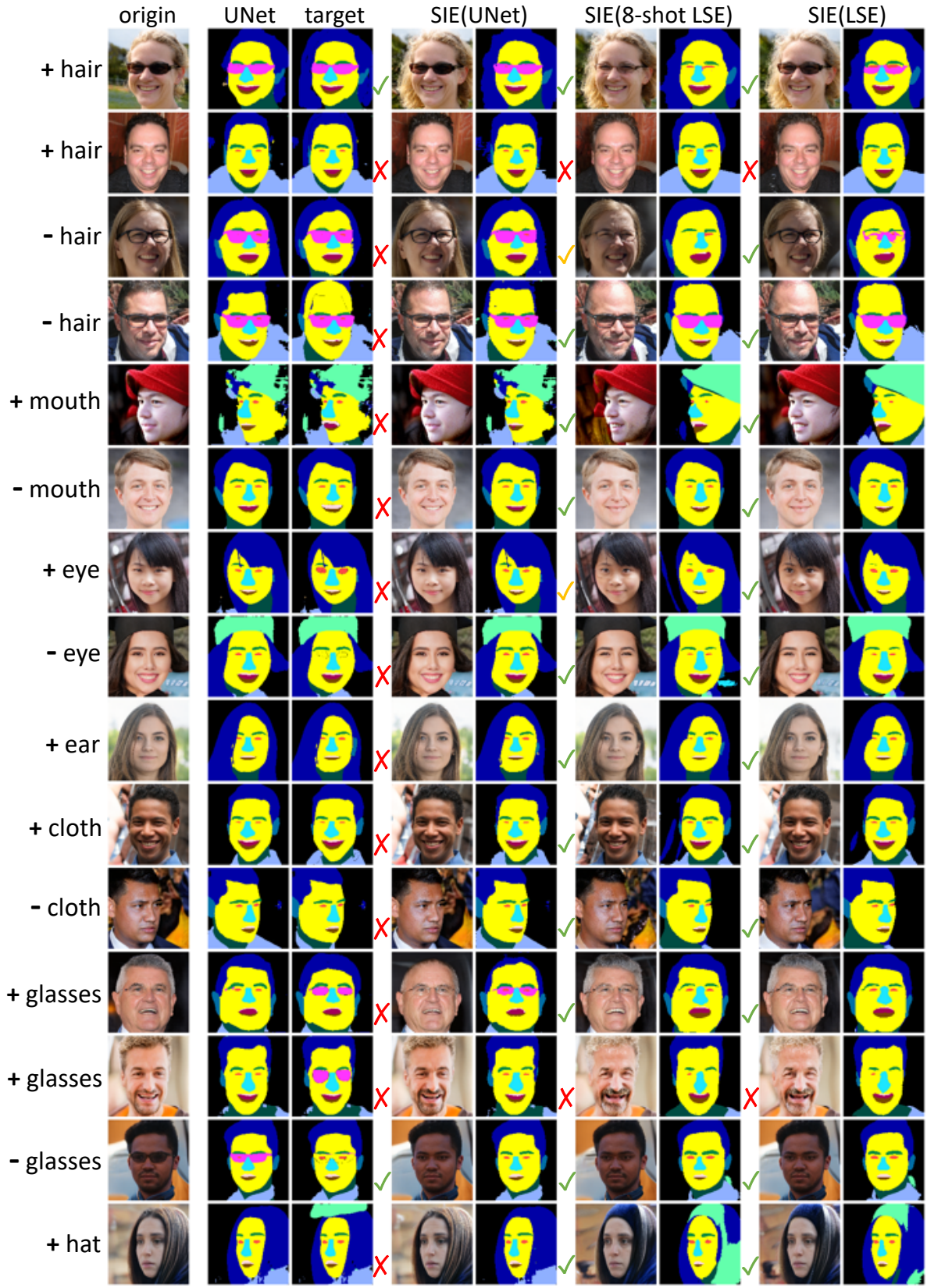
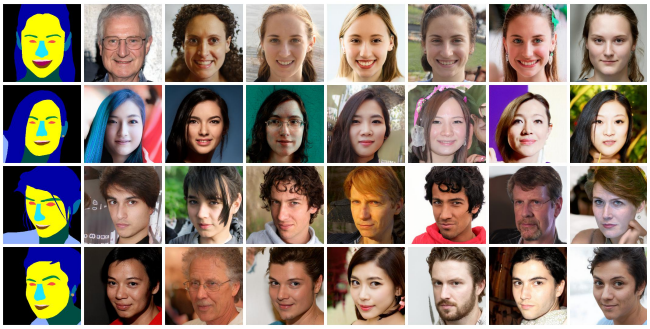


Figure 12. More SIE results on StyleGAN2-FFHQ. Annotations on the left are users' edit intentions. The following columns are original images, the face segmentation from UNet, the modified semantic mask by the user, the results from SIE(UNet), SIE(8-shot LSE), and SIE(LSE), respectively. The green ticks and red crosses represent whether the editing success or not. Other yellow ticks indicate that the image quality degrades.

(a) SCS(1-shot LSE)

(b) SCS(4-shot LSE)

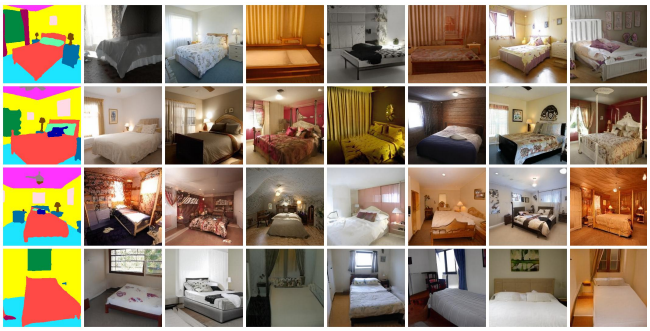


(c) SCS(8-shot LSE)

(d) SCS(16-shot LSE)

(e) SCS(UNet)

Figure 13. The results of SCS on StyleGAN2-FFHQ using LSEs and UNet.



(a) SCS(1-shot LSE)

(b) SCS(4-shot LSE)

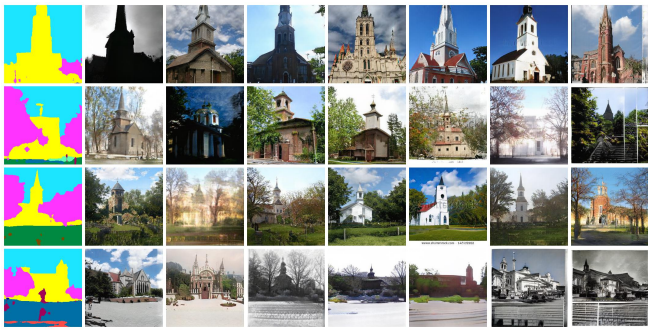
(c) SCS(8-shot LSE)

(d) SCS(16-shot LSE)

(e) SCS(DeepLabV3)

Figure 14. The results of SCS on StyleGAN2-Bedroom using LSEs and DeepLabV3.

(a) SCS(1-shot LSE)



(b) SCS(4-shot LSE)

(c) SCS(8-shot LSE)

(d) SCS(16-shot LSE)

(e) SCS(DeepLabV3)

Figure 15. The results of SCS on StyleGAN2-Church using LSEs and DeepLabV3.