Few-Shot Incremental Learning for Label-to-Image Translation Supplementary Material

Pei Chen[1], Yangkang Zhang[1], Zejian Li[1]*, Lingyun Sun[1,2]
[1]Alibaba-Zhejiang University Joint Institute of Frontier Technologies, Zhejiang University
[2]Zhejiang-Singapore Innovation and AI Joint Research Lab
{chenpei, yangkz, zejianlee, sunly}@zju.edu.cn

We provide a supplementary introduction to the proposed method FILIT and additional experimental results to examine the effectiveness of the FILIT. Firstly, we introduce the limitations of FILIT (Sec. S1). Secondly, we provide a detailed introduction of the architectures of the generator and the discriminator (Sec. S2) and the training setup (Sec. S3). Thirdly, we provide more qualitative and quantitative results (Secs. S4.3 and S4.4). In particular, we additionally compare FILIT with state-of-the-art label-to-image translation models (SPADE [15], CC-FPSE [11], CLADE [22] and OASIS [19]), which are trained with full datasets without incremental learning (Tab. S3, Figs. S6 and S7). Fourthly, we provide more ablation study results and cross-domain generative results (Secs. S4.6 and S4.8). Finally, we analyze failure examples (Sec. S4.9).

S1. Limitations

Our method is based on several assumptions. 1) We assume that base convolution filters and normalization layers trained on a large-scale dataset can be modulated to achieve novel tasks. 2) Convolution filters corresponding to different semantic classes belong to the same affine space. 3) Semantic classes with perceptually similar visual appearances have modulation parameters close to each other.

Limitations of our proposed method are stated as follows. 1) Our method relies heavily on the accuracy of data annotation; when the annotation is unclear or inaccurate, our method cannot capture the differences between semantic classes precisely. 2) Because we only learn a few parameters for a new task, when the task contains complex details, FILIT may fail to capture the details accurately (Sec. S4.9). 3) When the novel semantic classes and the learned classes are perceptually dissimilar, the proposed modulation transfer strategy cannot help the convergence in the learning of new classes (Sec. S4.8). 4) Our method has a limitation in fairness. Our method can be used in human-related tasks, such as fashion or face generation, as shown in Fig. S9. Human-related datasets may have data bias related to fairness issues. Such bias can be captured by our model in the training. Our model may reflect the bias by generating popular modes but ignoring underrepresented modes.

To the best of our knowledge, our method has no other ethical/privacy/transparency impact. Specifically, the proposed method helps to secure privacy and transparency. In our design, user-related data is only used in the incremental training stage. In this state, new components (modulation parameters) are created to learn the data. These components are separated from the base network, so they can be accessed, rectified, erased, transported or frozen easily to protect privacy, without affecting other components.

S2. Details of Architecture

S2.1. Details of Generator

Our semantically-adaptive generator is built on top of CLADE [22]. The generator stacks ResNet blocks [3] with normalization, convolution, and LeakyReLU. Fig. S1(a) illustrates the global architecture of the generator, and Fig. S1(b) shows the details of a ResNet block. The generator takes a random noise and a semantic label map as inputs, and the label map is fed in each ResNet block to produce class-specific modulation parameters for normalization and convolution layers. We add an extra noise to the input of each ResNet block. The scale of such noise is adjusted by a per-channel scaling factor that is learned during training, similar to StyleGAN [6].

We further introduce the motivation of the modulation design of FILIT and discuss the connections to related methods. Main modulation techniques of filters include the modulation along input channels [7], output channels [17] and both [2, 28]. Existing techniques are based on typical convolution filters of size $\mathbb{R}^{C_{out} \times C_{in} \times s_w \times s_h}$. Here C_{out} and C_{in} are the number of output and input channels, while s_w and s_h are the kernel width and height. Label-to-image translation requires to modulate convolution filter according to pixel-wise classes, so the modulation gives an intermedi-

*Corresponding author

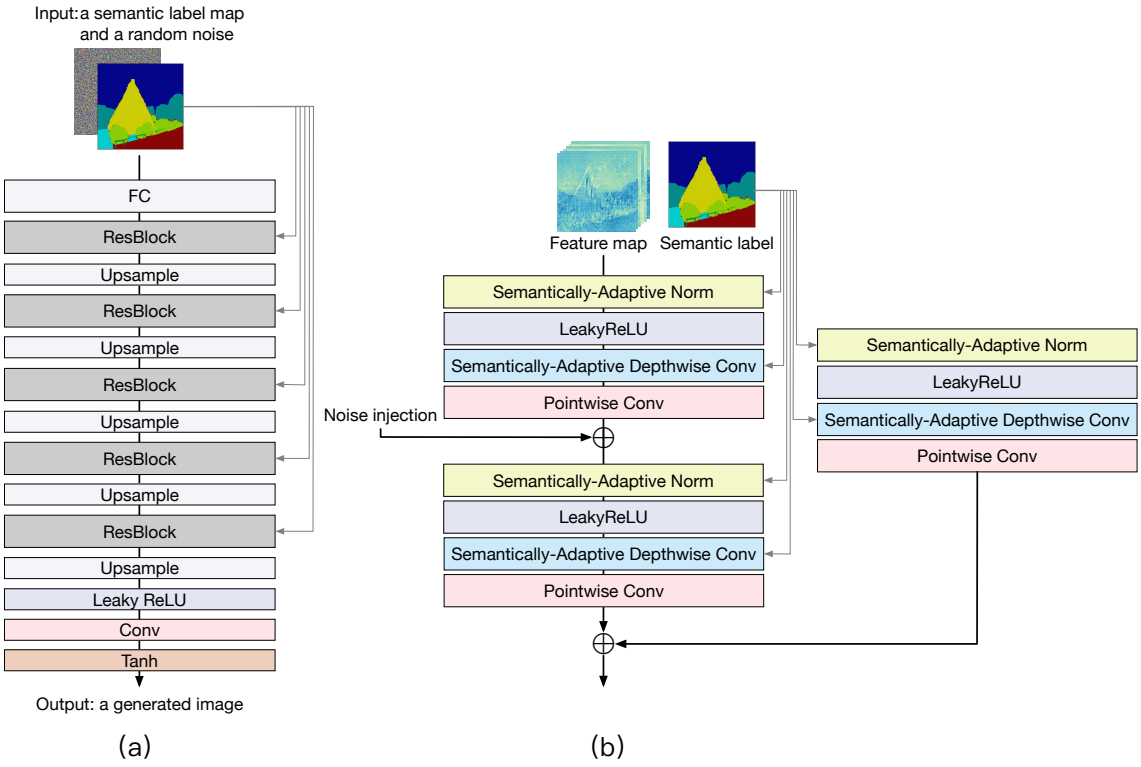

Figure S1. The architecture of generator. (a) The global architecture of the generator. (b) A detailed view of a ResNet block.

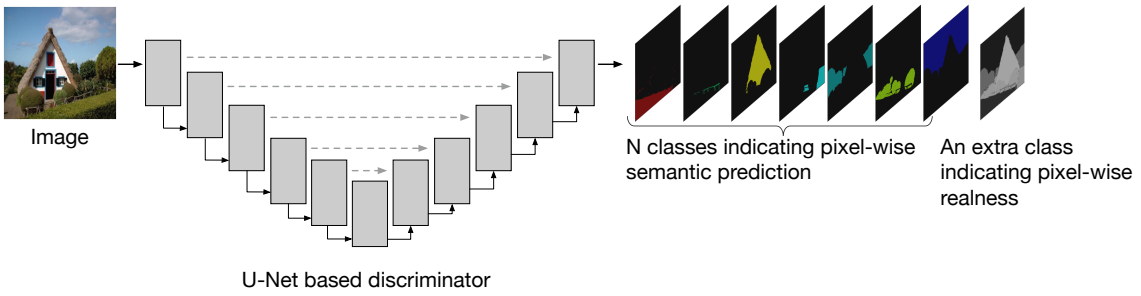

Figure S2. The architecture of discriminator. The discriminator is a U-Net giving $(N+1)$-class prediction. The first N predictions are the classification of semantic classes. The extra one prediction is the realness of the input image.

ate spatial filter of size $\mathbb{R}^{C_{out}\times C_{in}\times s_w\times s_h\times W\times H}$. W and H denote the width and height of the input feature map. Such intermediate filter is costly. To reduce the cost, we adopt depthwise separable convolution [4], and the modulation is applied to the filters of depthwise convolution only. Therefore, the intermediate filter size is only $\mathbb{R}^{C_{in}\times s_w\times s_h\times W\times H}$, while the modulation is applied along input and output channels simultaneously.

Recall that our modulation on a filter $\mathcal{F}$ is defined as

$$\hat{\mathcal{F}} = \gamma'_{conv} \otimes \frac{\mathcal{F}-\mu(\mathcal{F})}{\sigma(\mathcal{F})} + \beta'_{conv}$$
$$\hat{b} = b + b'_{conv} \quad (1)$$

Here $\mu(\mathcal{F})$ and $\sigma(\mathcal{F})$ are the channel-wise mean and standard derivation of $\mathcal{F}$, respectively. The filter is normalized along the spatial footprint [2] to remove base style information to facilitate learning unseen semantic classes. To equip the filter with the target styles of pixels' classes, our modulation applies a scale γ'_{conv} and a shift β'_{conv} to the base filter [2, 28]. The scale and shift introduce more flexibilities to the adapted filter compared with scaling the filter only [28]. Modulation with scales only has limited flexibilities for adaption [28]. Finally, the bias in the convolution is also modulated spatially in our design.

To strengthen semantic information, we use normalization layers modulated according to spatial semantic la-

bels. The normalization operation may wash away style information, so modulation is necessary [15]. Moreover, it is shown modulation in normalization benefits few-shot learning [14] and class information transfer [20]. Contrarily, StyleGAN2 [7] designs a normalized convolution filter without accompanied normalization. However, the normalized filter aims to restore the outputs back to the input's standard deviation. Using normalization and convolution separately introduces more flexibilities in learning unseen semantic classes. Notice that we use a pre-activation [3] structure in which convolution is performed after normalization, so the convolution bias will not be normalized. Ablation study in the main manuscript shows modulated normalization contributes to incremental learning (Tab. S5).

In our design, a block has $5 \times C_{in}$ parameters to learn for a new class. This is smaller than $2 \times C_{in} \times C_{out}$ in AdaFM [28] and GANmemory [2] to reduce computational budget, while it is larger than $2 \times C_{in}$ in FiLM [17] (modulated filters) and cGANTransfer [20] (modulated normalization) to introduce more flexibilities.

To explain proposed semantically-adaptive normalization and convolution more clearly, we present a pseudo code in Fig. S3.

S2.2. Details of Discriminator

We adopt the discriminator in OASIS [19]. The discriminator can be seen as a semantic segmentation network, which casts the discriminator task as a $N+1$ classes segmentation problem. The architecture of discriminator is built upon classic U-Net [18], which is an encoder-decoder network, and has been proven to be effective for semantic segmentation (Fig. S2).

S3. Details of Training

S3.1. Loss function

The loss function to train FILIT contains three components introduced as follows.

Adversarial Loss. Specifically, the adversarial losses for generator and discriminator are given in Eq. 2. Here x denotes the real image, H and W denote the height and width of the real image; (z, y) is the pair of noise and label map given to the generator G to synthesize an image. Given a real or fake image, the discriminator D gives a per-pixel $(N+1)$-class prediction [19].

$$\begin{aligned}\mathcal{L}_D = &- \mathbb{E}_{(x,y)} \left[\sum_{n=1}^{N} \sum_{h,w}^{H \times W} y_{h,w,n} \log D(x)_{h,w,n} \right] \\ &- \mathbb{E}_{(z,y)} \left[\sum_{h,w}^{H \times W} y_{h,w,n} \log D(G(z,y))_{h,w,n=N+1} \right] \\ \mathcal{L}_G = &- \mathbb{E}_{(x,y)} \left[\sum_{n=1}^{N} \sum_{h,w}^{H \times W} y_{h,w,n} \log D(G(z,y))_{h,w,n} \right]\end{aligned} \tag{2}$$

Perceptual Loss. We involve perceptual loss $\mathcal{L}_{vgg}$ [5] to pursue perceptual similarity between a generated image and the ground-truth image. Given a target image, we use VGG-19 [21] to extract feature maps to calculate the perceptual loss.

$$\mathcal{L}_{vgg} = \mathbb{E}_{(z,y)} \sum_{v=1}^{V} \frac{1}{M_v} \left[\| \phi_v(x) - \phi_v(G(z,y)) \|_1 \right] \tag{3}$$

where V denotes the number of used ReLU layers in VGG-19 [21], and ϕ_v denotes the v-th ReLU layer and M_v denotes the number of elements in ϕ_v.

Consistency Loss. This loss is based on the LabelMix augmentation operation [19] (denoting as LM). Specifically, given a semantic label map, it produces a binary label bl to mix a pair $(x, \hat{x})$ of real and fake images conditioned on the same label map: $LM(x, \hat{x}, bl) = bl \odot x + (1 - bl) \odot \hat{x}$. After obtaining the mixed image, a consistency loss term $\mathcal{L}_{con}$ is added to further train the discriminator to be equivariant under the LabelMix operation.

$$\mathcal{L}_{con} = \| D_{lg}(LM(x, \hat{x}, bl)) - LM(D_{lg}(x), D_{lg}(\hat{x}), bl) \|^2 \tag{4}$$

where $\|\cdot\|$ is the L_2 norm and D_{lg} are the logits attained before the last softmax activation layer.

Full Objective. We train FILIT by solving the overall loss function

$$\arg \min_G \max_D \mathcal{L}_{full} \equiv \mathcal{L}_{GAN} + \lambda_{vgg} \mathcal{L}_{vgg} + \lambda_{con} \mathcal{L}_{con} \tag{5}$$

S3.2. Training details

The optimizers, learning rates and loss functions in the pre-training phase and the incremental learning phase are the same. We use the ADAM optimizer [9] with $\beta_1 = 0, \beta_2 = 0.999$. The learning rates are set to 0.0001 for the generator and 0.0004 for the discriminator, respectively. We set the loss weight $\lambda_{vgg} = 10$ and $\lambda_{con} = 10$. In the ResNet blocks of generator, we use a synchronized version of batch normalization [27] to get better performance. In addition, we also add the positional encoding maps of semantic label maps as extra inputs [22]. It is defined as the relative

```
from torch.nn import functional as F
def basic_block(f, y, gamma_norm, beta_norm, gamma_conv, beta_conv, bias_conv,
        base_depthwise_filter, base_pointwise_filter, base_bias):
    # f : the input feature map, B x Cin x W x H
    # y : the semantic label map, B x N x W x H
    # gamma_norm : N x Cin
    # beta_norm : N x Cin
    # gamma_conv : N x Cin
    # beta_conv : N x Cin
    # bias_conv : N x Cin
    # base_depthwise_filter : Cin x sw x sh
    # base_pointwise_filter : Cout x Cin x 1 x 1
    # base_bias : Cin
    # N is the number of semantic classes.

    """ Guided Sampling for Normalization """
    dense_gamma_norm = F.embedding(y, gamma_norm) # B x Cin x W x H
    dense_beta_norm = F.embedding(y, beta_norm) # B x Cin x W x H

    """ Modulation in Normalization """
    hat_f = dense_gamma_norm * F.batch_norm(f) + dense_beta_norm

    """ Nonlinear """
    hat_f = F.leaky_relu(hat_f)

    """ Guided Sampling for Convolution """
    dense_gamma_conv = F.embedding(y, gamma_conv) # B x Cin x W x H
    dense_beta_conv = F.embedding(y, beta_conv) # B x Cin x W x H
    dense_bias_conv = F.embedding(y, bias_conv) # B x Cin x W x H

    """ Modulation in Convolution """
    # Cin x sw x sh
    n_depthwise_filter = (base_depthwise_filter-base_depthwise_filter.mean([1,2]))/(base_depthwise_filter.std([1,2]))
    # B x Cin x W x H x sw x sh
    m_depthwise_filter = einsum("BCWH,Cwh->BCWHwh",dense_gamma_conv,n_depthwise_filter)+dense_beta_conv
    # B x Cin x W x H x sw x sh
    f_unfolded = F.unfold(hat_f, [sw, sh], padding=1).view(B, Cin, sw, sh, W, H)
    # B x Cin x W x H
    hat_f2 = einsum("BCWHwh,BCwhWH->BCWH",m_depthwise_filter,f_unfolded)+dense_bias_conv

    return F.conv(hat_f2, base_pointwise_filter)
```

Figure S3. Pseudo code of FILIT's basic block in a PyTorch [16] style.

distance from each pixel to its corresponding object center, and it improves intra-class spatial adaptiveness. We resize images to 256×256 for training. In the pre-training phase, the training batch size is 30, and we train 100 epochs for ADE20K dataset, and 50 epochs for COCO-Stuff dataset. In the incremental learning phase, the batch size is 10 and the model is trained for 100 epochs.

S4. Experiment

S4.1. Dataset

We conduct experiments on two semantically-annotated datasets: ADE20K [29] and COCO-Stuff dataset [1]. We divide them into 21 sub-datasets $\{\mathcal{D}_0, \mathcal{D}_1 \cdots, \mathcal{D}_{20}\}$, respectively. $\mathcal{D}_0$ is used for pre-training and $\{\mathcal{D}_1, \mathcal{D}_2, \cdots, \mathcal{D}_{20}\}$ for incremental learning. Tab. S1 demonstrates the semantic class and the number of training samples of each task in ADE20K and COCO-Stuff dataset. Semantic classes learned in incremental learning are those have the least number of training samples in the original dataset.

S4.2. Baselines

We first compare FILIT with image generation models in incremental learning setting as follows.

- *LifelongGAN* [26] introduces knowledge distillation losses into BicycleGAN [30] to perform conditional image generation in the incremental learning setting. We reproduce LifelongGAN for our experiments.
- *PiggybackGAN* [24] factorizes filters trained on previous tasks to learn new tasks, and maintains a task-specific filter bank to memorize learned tasks. We use the code from [8] for our experiments.
- *fCLADE* is a weight-freezing baseline to investigate whether simply freezing most weights of a pre-trained

label-to-image model is able to achieve incremental learning. We use the CLADE [22] pre-trained on $\mathcal{D}_0$ as the starting model, and freeze its parameters except for the input channels for new classes when incrementally learning on $\{\mathcal{D}_1, \cdots, \mathcal{D}_T\}$.

- *FILIT-Oracle* is a variant of FILIT trained with $\{\mathcal{D}_0, \mathcal{D}_1, \cdots, \mathcal{D}_T\}$ jointly without incremental learning. This model marks the performance upper bound that FILIT can achieve in the incremental learning.

- *FILIT-SFT (Sequential Fine-Tuning)* is another variant of FILIT. It is also pre-trained on the Task 0, and then it is fine-tuned in a sequential manner. In the fine-tuning process, the whole generator is used to learn new tasks, and the decoder of discriminator is fixed [13].

Then we compare FILIT with state-of-the-art label-to-image translation models as follows.

- *SPADE* [15] uses spatially-adaptive normalization to avoid semantic information of label maps being washed away in normalization layers.

- *CC-FPSE* [11] predicts convolutional kernels conditioned on semantic label maps to effectively exploit the semantic layout for the generator.

- *CLADE* [22] uses class-adaptive normalization to propagate semantic information, which is a lightweight variant of SPADE.

- *OASIS* [19] first designs the discriminator for label-to-image translation as a semantic segmentation network, which provides stronger supervision to the discriminator as well as to the generator. It also injects a 3D noise tensor into the generator, which enables high-quality multi-modal image synthesis.

These label-to-image translation models are not designed for the incremental learning setting. Therefore, we trained them with $\{\mathcal{D}_0, \mathcal{D}_1, \cdots, \mathcal{D}_T\}$ jointly without the incremental learning process. We denote them as SPADE-Oracle, CC-FPSE-Oracle, CLADE-Oracle, and OASIS-Oracle in following sections. They are trained on ADE20K and COCO-Stuff datasets for 100 epochs, separately. They also present the performance upper bound that FILIT can achieve in the incremental learning.

S4.3. Qualitative Results

Figs. S4 and S5 present additional generated images of FILIT, FILIT's variants and compared conditional generative models based on incremental learning. As can be seen, LifelongGAN, PiggybackGAN and FILIT-SFT fail to generate clear images. fCLADE effectively avoids forgetting, but it hardly learns the features of new classes in incremental learning. FILIT generates images as visually-pleasing as FILIT-Oracle.

Figs. S6 and S7 present generated images of FILIT and compared label-to-image translation models on ADE20K and COCO-Stuff. Because compared models have no incremental training phase, we do not conduct recalling experiments to examine the catastrophic forgetting. Results show that FILIT generate images as visually-appearing as compared full-trained models.

S4.4. Quantitative Results

Tab. S2 reports quantitative results of FILIT, compared conditional generative models based on incremental learning, and FILIT's variants. The results show that FILIT achieves comparable performance with FILIT-Oracle and outperforms other compared models.

Tab. S3 reports quantitative results of FILIT and compared label-to-image translation models on ADE20K and COCO-Stuff. The results on COCO-Stuff show that FILIT achieves comparable performance with compared state-of-the-art label-to-image models. The results of FID and SceneFID on ADE20K show large differences across compared models. This may attribute to the instability of FID when the amount of testing data is small (1,098 testing images of 20 tasks in total).

S4.5. Human Evaluation Results

Results are in Tab. S4. For each comparison, we randomly picked 500 questions from the ADE20K dataset, and each question is answered by 10 anonymous workers. For each question, we give the workers an input semantic label and two shuffled images generated by FILIT and a compared method. They pick the image with better visual quality and layout alignment. Our method achieves a comparable performance with FILIT-Oracle and outperforms the others. We have approvals from workers.

S4.6. Ablation Study

We conduct ablation studies on variants of our proposed model on ADE20K dataset. Experimental results (Tab. S5) show that semantically-adaptive convolution filters, semantically-adaptive normalization and modulation transfer strategy all have positive influences on FID and SceneFID. In addition, we investigate the influence of the number of training samples of each task in the few-shot incremental training. Results show that the number of training samples of each task between 1-20 has little effect on the model performance. Fig. S8 presents the qualitative results of ablation study, and it shows that without adaptive normalization or modulation transfer strategy, the generated objects with novel classes become blurring. Without adaptive

convolution filters, the generated novel classes have fewer texture details.

In Tab. S5, the segmentation performance of compared ablation models have minor differences. In our opinion, the insensitivity of segmentation performance is due to that those new semantic classes learned in the incremental learning only occupy a small part of area in the generated images. The segmentation models we use, like UpperNet-101 [23] for ADE20K, are trained on large-scale datasets. They are able to perceive images globally. They can infer the regions corresponding to the new classes from other regions of images.

S4.7. Model Expansion

FILIT adds class-specific modulation parameters to the original model when learning a new task, and it belongs to the expansion-based method. We compare the model expansion of FILIT with PiggybackGAN [24], which is also an expansion-based method. As shown in Tab. S6, the additional parameters for each subsequent class of FILIT are only 0.06M. In addition, to exclude the influence of initial model size on performance, we design FILIT-mini with 6.70M parameters from start, less than PiggybackGAN's (7.84M). FILIT-mini still outperforms PiggybackGAN (FID: 81.3 vs 253.9, SceneFID: 94.8 vs 153.3) in the incremental learning on Task 1 to 20, so the improvement is from model design instead of size. In summary, FILIT requires little amount of expansion while achieving compelling performance.

S4.8. Experiments on Cross-Domain Tasks

We use the model trained on all tasks of ADE20K to continually learn cross-domain samples from DeepFashion [12] and CelebAMask-HQ [10] dataset. Ten samples from DeepFashion are picked to construct $\mathcal{D}_{21}$, which contains nine novel semantic classes: "background", "dress", "neckwear", "headwear", "belt" , "footwear", "hair", "skin", and "face". Ten samples from CelebAMask-HQ are picked to construct $\mathcal{D}_{22}$, which contains sixteen classes: "skin", "left brow", "right brow", "left eye", "right eye", "left ear", "right ear", "ear ring", "nose", "mouth", "up lips", "down lip", "neck", "hair", "cloth", and "background". Fig. S9 shows additional generated results of FILIT in the domain of DeepFashion and CelebAMask-HQ. After only learning a few samples, FILIT can generate persons in different poses and generate faces with vivid facial features.

S4.9. Failure Examples

Fig. S10 presents several failure cases of the proposed method. These generated objects of classes learned in the incremental learning lack fine-grained texture. There are two reasons. Firstly, semantic classes learned in incremental learning in our experiments are those having the least number of samples in the original dataset. These classes are usually atypical, and their training samples are diverse. We provide examples of training samples for four classes in Fig. S10, and their appearances are diverse with complex textures. Therefore, it is difficult to capture the statistical characteristics of these classes with only a few samples. Secondly, the model only learns a few modulation parameters for a novel class, and thus it fails to capture the target distribution with a few iterations.

Dataset	Task ID	Semantic class	NT
ADE20K	0	wall, building, sky, floor, tree, ceiling, road, bed, windowpane, grass, abinet, sidewalk person, earth, door, table, mountain, plant, curtain, chair, car, water, painting, sofa, shelf, house, sea, mirror, rug, field, seat, armchair, fence, desk, rock, wardrobe, lamp, bathtub, railing, cushion, base, box, column, signboard, sand, chest of drawers, counter, sink, skyscraper, fireplace, refrigerator, grandstand, covered, path, stairs, case, pool table, pillow, screen door, stairway, adiator, glass, clock, flag, screen, bulletin board, hood, sconce, river, bridge, bookcase, blind, coffee table, book, hill, oilet, bench, countertop, palm, stove, vase, tray, fan, kitchen island, swivel chair, boat, bar, towel, light, compute, bus, tower, handelier, awning, crt screen, television receiver, truck, apparel, pole, land, bannister, ottoman, bottle, buffet, poster, van, traffic light, washer, plaything, stool, basket, bag, minibike, oven, ball, food, plate,monitor, shower, ashcan, step, trade name, microwave, pot, animal, bicycle, airplane, dishwasher, blanket sculpture, streetlight,	-
	1	runway	72
	2	arcade machine	54
	3	hovel	50
	4	booth(or cubicle or stall or kiosk)	56
	5	dirt track	67
	6	escalator (or moving staircase or moving stairway)	28
	7	stage	96
	8	ship	34
	9	fountain	79
	10	canopy	55
	11	conveyer belt (or conveyor belt or conveyer or conveyor or transporter)	41
	12	washer(or automatic washer or washing machine)	66
	13	swimming pool (or swimming bath or natatorium)	46
	14	barrel (or cask)	33
	15	waterfall (or falls)	67
	16	tent (or collapsible shelter)	41
	17	cradle	59
	18	tank (storage tank)	46
	19	lake	37
	20	pier	71
COCO-Stuff	0	unlabeled, person, bicycle, car, motorcycle, airplane, bus, train, truck, boat, traffic light, fire, furniture-other, street sign, stop sign, bench, bird, cat, dog, horse, sheep, cow, elephant, zebra, hydrant, zebra, wood, door-stuff, giraffe, hat ,backpack, umbrella, shoe, eye glasses, handbag, tie, suitcase, frisbee, skis, wall-tile, fence, snowboard, sports ball, kite, baseball bat, baseball glove, skateboard, surfboard, tennis racket, wall-wood, bottle, plate, wine glass, cup, fork, knife, spoon, bowl, banana, apple, sandwich, orange, broccoli, floor-other, carrot, pizza, cake, chair, couch, potted plant, bed, mirror, dining table, window, desk, toilet, door, floor-tile, tv, laptop, mouse, remote, keyboard, cell phone, microwave, oven, sink, refrigerator, blender, book, floor-wood, clock, vase, teddy bear, hair brush, banner, blanket, branch, bridge, building-other, bush, cabinet, flower, fog, cage, cardboard, carpet, ceiling-other, cloth, clothes, clouds, counter, curtain, desk-stuff, dirt, water-other, grass, gravel, ground-other, hill, house, leaves, light, metal, mirror-stuff, mountain, napkin, paper, fruit, pavement, pillow, plant-other, plastic, platform, playingfield, railing, railroad, river, road, rock, roof, rug, sand, sea, shelf, sky-other, skyscraper, snow, stairs, stone, structural-other, table, tent, window-blind, textile-other, towel, tree, vegetable, wall-brick, wall-concrete, wall-other, wall-panel, wall-stone,window-other	-
	1	parking meter	672
	2	bear	868
	3	donut	1350
	4	toaster	134
	5	hot dog	1040
	6	scissors	833
	7	hair drier	120
	8	cupboard	882
	9	floor-stone	1213
	10	moss	197
	11	floor-marble	964
	12	ceiling-tile	276
	13	mat	518
	14	toothbrush	749
	15	mud	566
	16	salad	447
	17	solid-other	708
	18	net	1356
	19	waterdrops	76
	20	straw	1390

Table S1. The semantic class and the number of testing samples (denoting as NT in table) of each task in ADE20K and COCO-Stuff dataset.

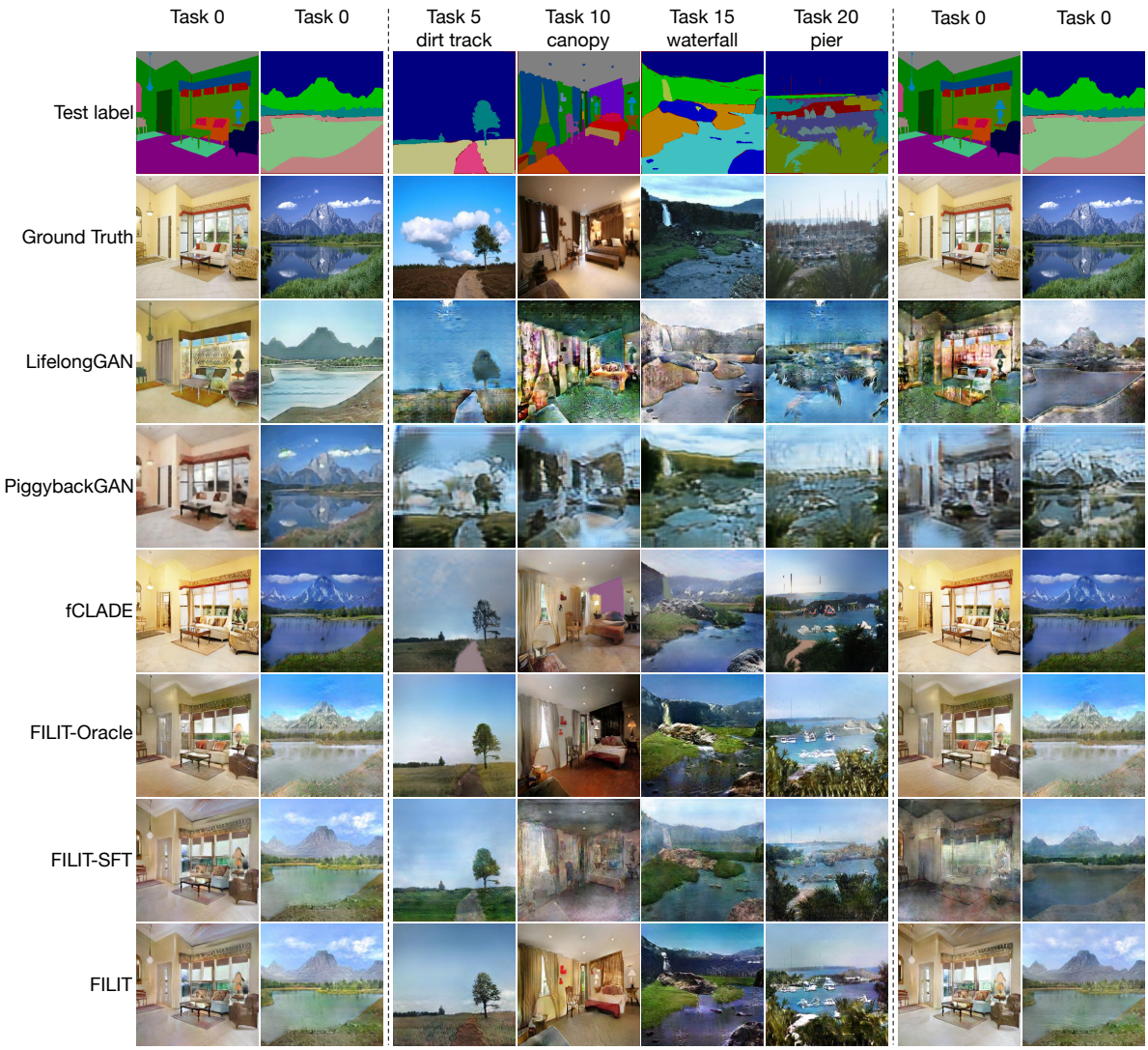

Figure S4. Qualitative results on ADE20K dataset. The first two columns show the generative results on Task 0 after pre-training. The central four columns show the incremental learning results on Task 5, 10, 15 and 20. The last two columns on the right display results of recalling Task 0 after the incremental learning, and they are to examine catastrophic forgetting.

Figure S5. Qualitative results on COCO-Stuff dataset. The first two columns show the generative results on Task 0 after pre-training. The central four columns show the incremental learning results on Task 5, 10, 15 and 20. The last two columns on the right display results of recalling Task 0 after the incremental learning, and they are to examine catastrophic forgetting.

Method	ADE20K							COCO-Stuff						
	mIoU↑	accu↑	FID↓	FmIoU↓	Faccu↓	FFID↓	SceneFID↓	mIoU↑	accu↑	FID ↓	FmIoU↓	Faccu↓	FFID↓	SceneFID↓
LifelongGAN	10.0	25.2	170.5	37.7%	32.6%	>50%	112.3	2.2	4.8	160.0	>50%	>50%	>50%	57.3
PiggybackGAN	5.2	15.4	253.9	>50%	>50%	>50%	153.3	2.6	7.7	208.4	>50%	>50%	>50%	116.2
fCLADE	21.8	49.1	78.4	**0%**	**0%**	**0%**	87.0	14.9	34.4	48.2	**0%**	**0%**	**0%**	45.8
FILIT-Oracle	**23.5**	**50.4**	**74.4**	-	-	-	**77.8**	**18.0**	**38.0**	**32.0**	-	-	-	28.8
FILIT-SFT	16.8	40.7	170.2	35.6%	24.4%	>50%	147.8	15.0	32.4	64.4	16.3%	13.5%	>50%	37.6
FILIT	23.2	49.6	77.1	**0%**	**0%**	**0%**	80.7	**18.0**	37.9	32.7	**0%**	**0%**	**0%**	**22.3**

Table S2. Quantitative results on ADE20K and COCO-Stuff. ↑ means a higher value is better, and vice versa. The best and second best performances are highlighted by using bold and underline, separately.

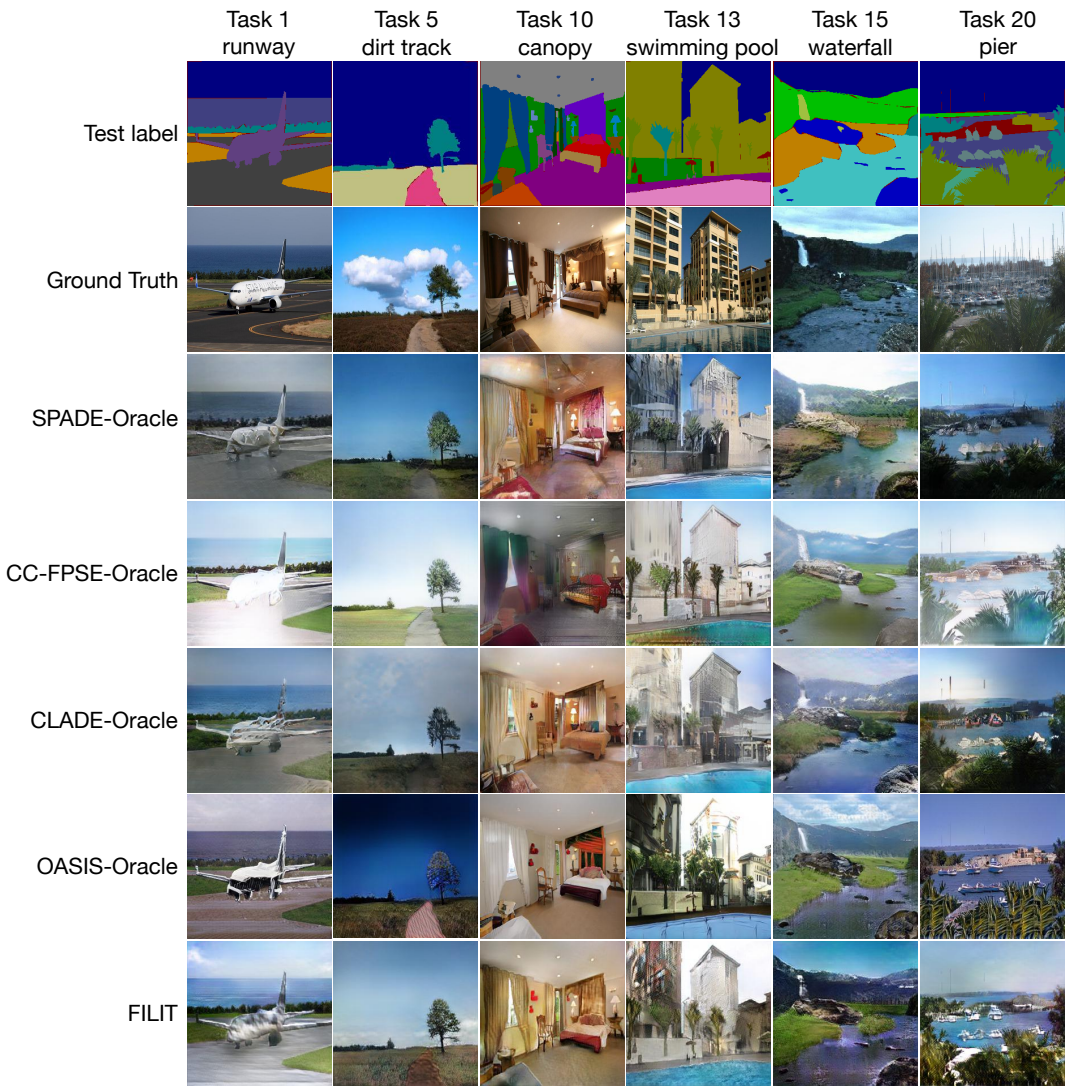

Figure S6. Qualitative results on ADE20K dataset. The compared models SPADE-Oracle, CC-FPSE-Oracle, CLADE-Oracle and OASIS-Oracle are trained with training samples of all tasks jointly without the incremental learning process. FILIT incrementally trains novel classes. Results show that FILIT generates images as visually-appearing as compared models without forgetting already-learned classes.

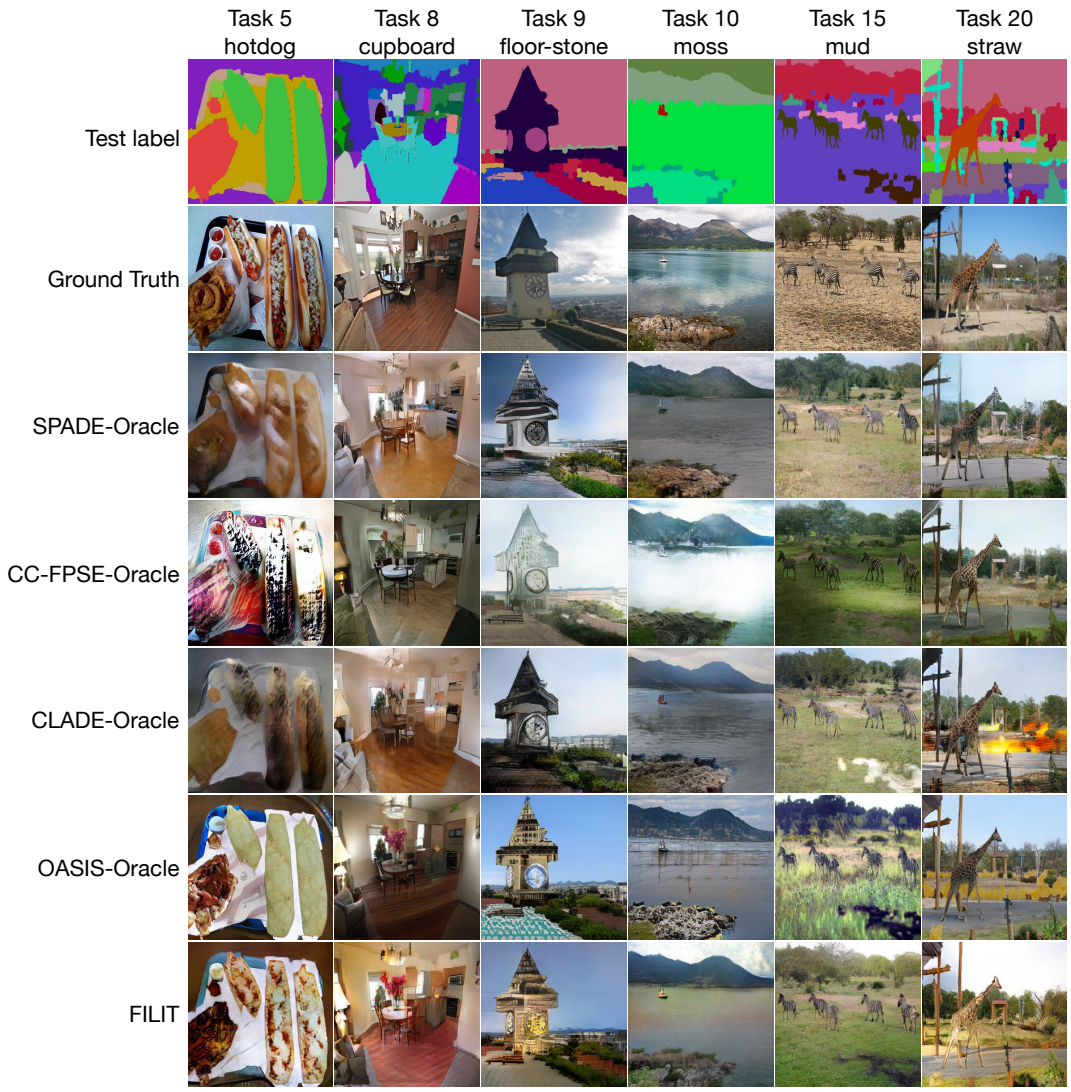

Figure S7. Qualitative results on COCO-Stuff dataset. The compared models SPADE-Oracle, CC-FPSE-Oracle, CLADE-Oracle and OASIS-Oracle are trained with training samples of all tasks jointly without the incremental learning process. FILIT incrementally trains novel classes. Results show that FILIT generates images as visually-appearing as compared models without forgetting already-learned classes.

Method	ADE20K				COCO-Stuff			
	mIoU↑	accu↑	FID↓	SceneFID↓	mIoU↑	accu↑	FID ↓	SceneFID↓
SPADE-Oracle	23.5	52.8	**59.0**	**60.0**	18.4	**41.5**	**23.0**	**17.7**
CC-FPSE-Oracle	21.6	48.1	81.8	72.1	16.6	35.6	45.8	25.7
CLADE-Oracle	23.8	**53.5**	65.0	66.9	17.2	39.6	32.2	29.1
OASIS-Oracle	**25.0**	51.8	68.4	79.2	**20.0**	**41.5**	25.2	30.6
FILIT-Oracle	23.5	50.4	74.4	77.8	18.0	38.0	32.0	28.8
FILIT	23.2	49.6	77.1	80.7	18.0	37.9	32.7	22.3

Table S3. Quantitative results on ADE20K and COCO-Stuff. The compared models are SPADE-Oracle, CC-FPSE-Oracle, CLADE-Oracle, OASIS-Oracle and FILIT-Oracle. They are trained with training samples of all tasks jointly without the incremental learning process. ↑ means a higher value is better, and vice versa. The best performances are highlighted.

Ours vs.	Task 0 after Pre-Training	Task 1-20	Task 0 after Incremental Learning
LifelongGAN	78.8%	81.8%	78.1%
PiggybackGAN	68.2%	81.4%	82.2%
FILIT-Oracle	45.7%	59.1%	47.7%
FILIT-SFT	50.0%	79.4%	80.0%
fCLADE	-	85.8%	-

Table S4. User study Results.

Method	mIoU↑	accu↑	FID↓	SceneFID↓
FILIT-20	**23.2**	**49.6**	77.1	80.7
w/o adaptive convolution filters	23.0	48.8	81.6	84.0
w/o adaptive normalization	22.5	48.5	83.7	88.4
w/o modulation transfer	23.0	49.4	84.1	87.2
FILIT-15	23.2	49.5	77.4	80.1
FILIT-10	23.2	49.4	**77.0**	**79.9**
FILIT-5	23.0	49.1	77.9	82.2
FILIT-1	23.0	49.1	79.6	82.0

Table S5. Results in the ablation study for 20-task incremental learning on ADE20K dataset.

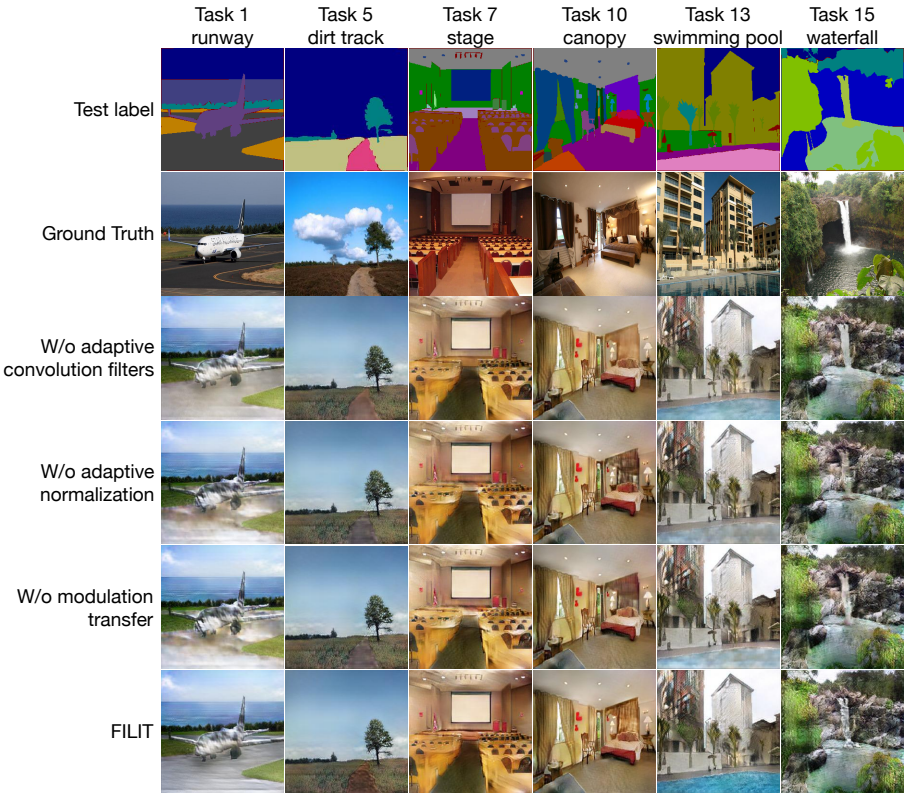

Figure S8. Qualitative results of ablation study.

Method	Task 0	Task 1	Task 20	Additional↓
PiggybackGAN †	7.84M	12.22M	95.44M	4.38M
PiggybackGAN ○	13.16M	13.64M	22.32M	0.48M
FILIT	18.99M	19.05M	20.15M	0.06M
FILIT-mini	6.70M	6.73M	7.28M	**0.03M**

Table S6. The numbers of additional parameters when learning a new semantic class of ADE20K. † means the numbers of Task 0 and Task 1 are from Zhai et al. [25] and the number of Task 20 is calculated accordingly. ○ means the numbers are measured in the implementation of [8].

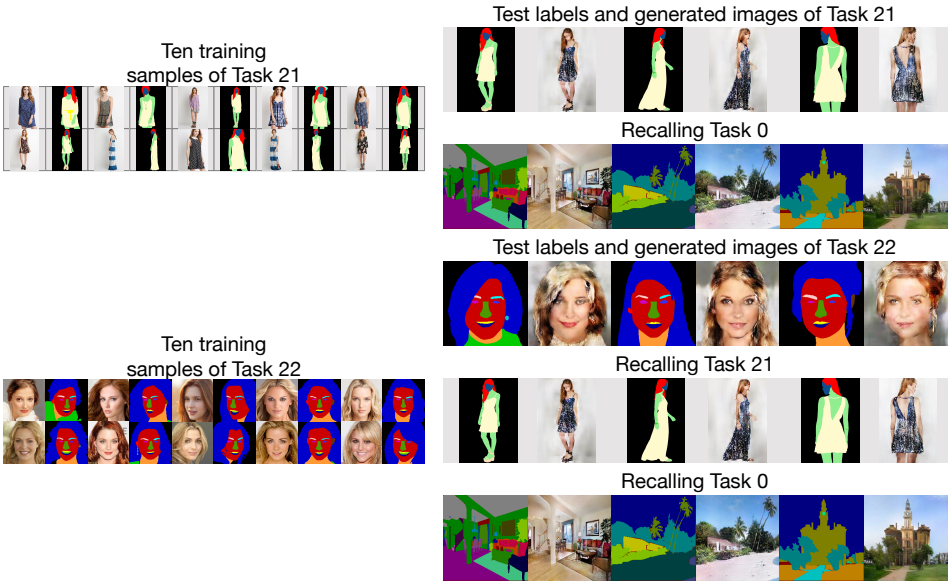

Figure S9. Incremental learning results from DeepFashion and CelebAMask-HQ dataset in a ten-shot setting. After training on ten samples from DeepFashion, we recall Task 0 from ADE20K dataset. After training on another ten samples from CelebAMask-HQ, we recall Task 0 and Task 21 again. Both recalling experiments show FILIT does not forget learned tasks.

Figure S10. Failure examples from ADE20K and COCO-Stuff.

References

[1] Holger Caesar, Jasper Uijlings, and Vittorio Ferrari. Cocostuff: Thing and stuff classes in context. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1209–1218, 2018. 4

[2] Yulai Cong, Miaoyun Zhao, Jianqiao Li, Sijia Wang, and Lawrence Carin. GAN memory with no forgetting. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pages 16481–16494. Curran Associates, Inc., 2020. 1, 2, 3

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 630–646. Springer, 2016. 1, 3

[4] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. MobileNets: efficient convolutional neural networks for mobile vision applications. *ArXiv*, abs/1704.04861, 2017. 2

[5] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 694–711. Springer, 2016. 3

[6] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4401–4410, 2019. 1

[7] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8107–8116, 2020. 1, 3

[8] Kaushilk. PiggybackGAN. https://github.com/kaushik333/Piggyback-GAN-Pytorch. Last accessed on Nov 17, 2021. 4, 13

[9] Diederik P Kingma and Jimmy Ba. Adam: a method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 3

[10] Cheng-Han Lee, Ziwei Liu, Lingyun Wu, and Ping Luo. MaskGAN: towards diverse and interactive facial image manipulation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 6

[11] Xihui Liu, Guojun Yin, Jing Shao, Xiaogang Wang, et al. Learning to predict layout-to-image conditional convolutions for semantic image synthesis. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 570–580. Curran Associates, Inc., 2019. 1, 5

[12] Ziwei Liu, Ping Luo, Shi Qiu, Xiaogang Wang, and Xiaoou Tang. Deepfashion: powering robust clothes recognition and retrieval with rich annotations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1096–1104, 2016. 6

[13] Sangwoo Mo, Minsu Cho, and Jinwoo Shin. Freeze the discriminator: a simple baseline for fine-tuning gans. In *CVPR AI for Content Creation Workshop*, 2020. 5

[14] Atsuhiro Noguchi and Tatsuya Harada. Image generation from small datasets via batch statistics adaptation. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 2750–2758, 2019. 3

[15] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic image synthesis with spatially-adaptive normalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2337–2346, 2019. 1, 3, 5

[16] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 8024–8035. Curran Associates, Inc., 2019. 4

[17] Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron Courville. FiLM: Visual reasoning with a general conditioning layer. *AAAI Conference on Artificial Intelligence*, 32(1):3942–3951, Apr. 2018. 1, 3

[18] 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. 3

[19] Edgar Schönfeld, Vadim Sushko, Dan Zhang, Juergen Gall, Bernt Schiele, and Anna Khoreva. You only need adversarial supervision for semantic image synthesis. In *International Conference on Learning Representations (ICLR)*, 2021. 1, 3, 5

[20] Mohamad Shahbazi, Zhiwu Huang, Danda Pani Paudel, Ajad Chhatkuli, and Luc Van Gool. Efficient conditional GAN transfer with knowledge propagation across classes. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12162–12171, 2021. 3

[21] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 3

[22] Zhentao Tan, Dongdong Chen, Qi Chu, Menglei Chai, Jing Liao, Mingming He, Lu Yuan, Gang Hua, and Nenghai Yu. Efficient semantic image synthesis via class-adaptive normalization. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2021. 1, 3, 5

[23] Tete Xiao, Yingcheng Liu, Bolei Zhou, Yuning Jiang, and Jian Sun. Unified perceptual parsing for scene understanding. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 418–434, 2018. 6

[24] Mengyao Zhai, Lei Chen, Jiawei He, Megha Nawhal, Frederick Tung, and Greg Mori. Piggyback GAN: Efficient lifelong learning for image conditioned generation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 397–413. Springer, 2020. 4, 6

[25] Mengyao Zhai, Lei Chen, and Greg Mori. Hyper-LifelongGAN: scalable lifelong learning for image condi-

tioned generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2246–2255, 2021. 13

[26] Mengyao Zhai, Lei Chen, Frederick Tung, Jiawei He, Megha Nawhal, and Greg Mori. Lifelong GAN: Continual learning for conditional image generation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 2759–2768, 2019. 4

[27] Hang Zhang, Kristin Dana, Jianping Shi, Zhongyue Zhang, Xiaogang Wang, Ambrish Tyagi, and Amit Agrawal. Context encoding for semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7151–7160, 2018. 3

[28] Miaoyun Zhao, Yulai Cong, and Lawrence Carin. On leveraging pretrained GANs for generation with limited data. In *International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 11340–11351. PMLR, 2020. 1, 2, 3

[29] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ADE20K dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 633–641, 2017. 4

[30] Jun-Yan Zhu, Richard Zhang, Deepak Pathak, Trevor Darrell, Alexei A Efros, Oliver Wang, and Eli Shechtman. Multimodal image-to-image translation by enforcing bi-cycle consistency. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 465–476. Curran Associates, Inc., 2017. 4