Supplementary Material: Example-Guided Image Synthesis using Masked Spatial-Channel Attention and Self-Supervision

In the supplementary material, we provide details on how to perform interpolation in Sec. 1, more analysis of the learned attention in Sec. 2, the impact of semantic similarity to performance in Sec. 3, results on an additional dataset in Sec. 4, more results of scenes swapping in Sec. 5, more details of the synthesis module in Sec 6, patch sampling strategy and more training details in Sec. 7, pretraining of MSCA modules in Sec. 8. The code containing our model and demos will be made avaiable after the review process.

1 Details of Interpolation

In Fig.5 and Fig. 7 of the main paper, we show how to interpolate between two inputs styles by manipulating the attention of the trained model. Here, we describe the details of style interpolation.

First, we show how to achieve **global style interpolation** (Fig.5 of the main paper). In this setting, we generate interpolated images using condition c_1, two example images x_2 and x_3, and an interpolating factor $\gamma \in [0, 1]$ that controls how much the style of x_2 is used. To perform interpolation, we first compute features $F^{(i)}_{x,2}, F^{(i)}_{x,3}$ and unnormalized spatial-attention maps $\widetilde{\alpha}^{(i)}_2, \widetilde{\alpha}^{(i)}_3$ of the exemplar images x_2, x_3. Next, the unnormalized spatial attention map of x_2 is updated by $\widetilde{\alpha}^{(i)}_2 \coloneqq \widetilde{\alpha}^{(i)}_2 + \log \frac{\gamma}{1-\gamma}$. Afterwards, both feature maps $F^{(i)}_{x,2}, F^{(i)}_{x,3}$ and unnormalized spatial attention $\widetilde{\alpha}^{(i)}_2, \widetilde{\alpha}^{(i)}_3$ are concatenated along the horizontal axis. The concatenated unnormalized spatial attention are used to generated an joint spatial attention map. Then, the masking score (output scores of the 2-layer MLP) is also interpolated. Finally, with the remaining procedures unchanged, i.e., spatial aggregation, feature masking, channel aggregation and synthesis, interpolation results are readily generated.

Next, we show how to achieve **spatial styles interpolation** (Fig.7 of the main paper). To generate spatial style interpolation results, we keep the procedures of our method unchanged before the before channel aggregation step. Then, we perform style interpolate at the channel aggregation stage. Specifically, with condition c_1 and two exemplar images x_2 and x_3, we first compute two sets of masked features $\widetilde{\boldsymbol{V}}^{(i)}_2, \widetilde{\boldsymbol{V}}^{(i)}_3$ and un-normalized channel attention $\widetilde{\beta}^{(i)}_2, \widetilde{\beta}^{(i)}_3$ from inputs c_1, x_2 and c_1, x_3, respectively [1]. Then, using the interpolating factor γ each spatial location, the unnormalized channel attention map of x_2 is updated by $\widetilde{\beta}^{(i)}_2 \coloneqq \widetilde{\beta}^{(i)}_2 + \log \frac{\gamma}{1-\gamma}$. Finally, both masked features and the unnormalized

[1] Note that the un-nromalized channel attention $\widetilde{\beta}^{(i)}_2, \widetilde{\beta}^{(i)}_3$ are the same for the two inputs.

channel attention map are concatenated along the channel axis for channel aggregation.

2 Analysis of the Learned Attention

To understand how the learned attention transfer features across different semantics from exemplar images to target label maps, the following visualization is performed on test set. Specifically, we use the learned MSCA at the finest scale to transfer one-hot encoded semantic label maps of the exemplar images, then compare the transferred results against the target semantic label map. The resulting transfer matrix is visualized in Fig. 1. From the figure, the learned attention tends to transfer feature across semantically similar labels, such as {`plant-other,bush,hill,leaves`} → `tree`, {`clouds,fog`} → `sky-other`, {`grass`} → `playingfield` and {`bush`} → `plant-other`. It shows that our attention can automatically discover semantics labels that shares similar visual appearance.

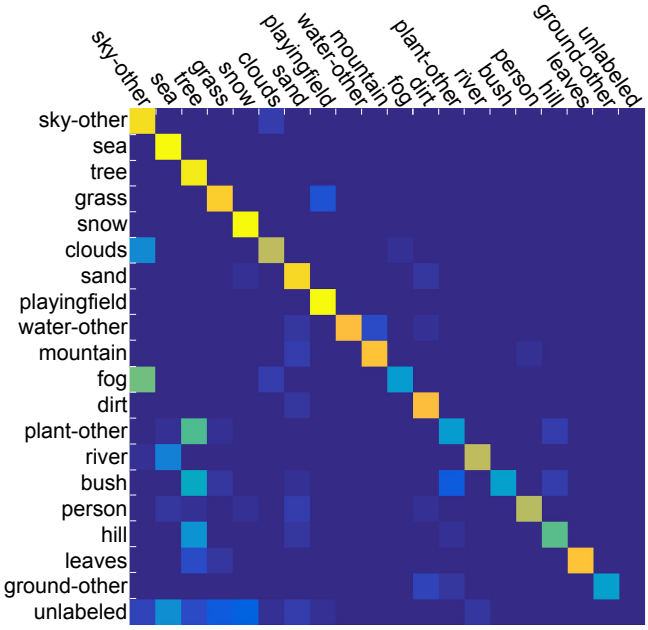

Fig. 1. The transfer matrix visualizes how our learned attention transform features from semantics of exemplar images (vertical axis) to semantics of target label maps (horizontal axis) on the test set. Colors from blue to red represent ratios from 0 to 1. The most appeared 20 semantics are shown in figure. The learned attention can transfer across semantic labels with similar appearance, for instance, from {`plant-other,bush,hill,leaves`} to `tree` on column 3.

In addition, we perform a hierarchical clustering on the top-25 semantic labels on the test set. Specifically, we use each row of the transfer matrix as the feature vector of each semantic labels. The diagonal of the transfer matrix is set to the second largest value of each row. From the dengrogram visualized in Fig. 2, we observe that semantics with similar appearances such as {`cloud`, `sky-other`}, {`river`, `sea`}, {`dirt`, `sand`} tend to group together, suggesting that our attention model can discover semantic labels with similar visual appearances for effective feature transfer.

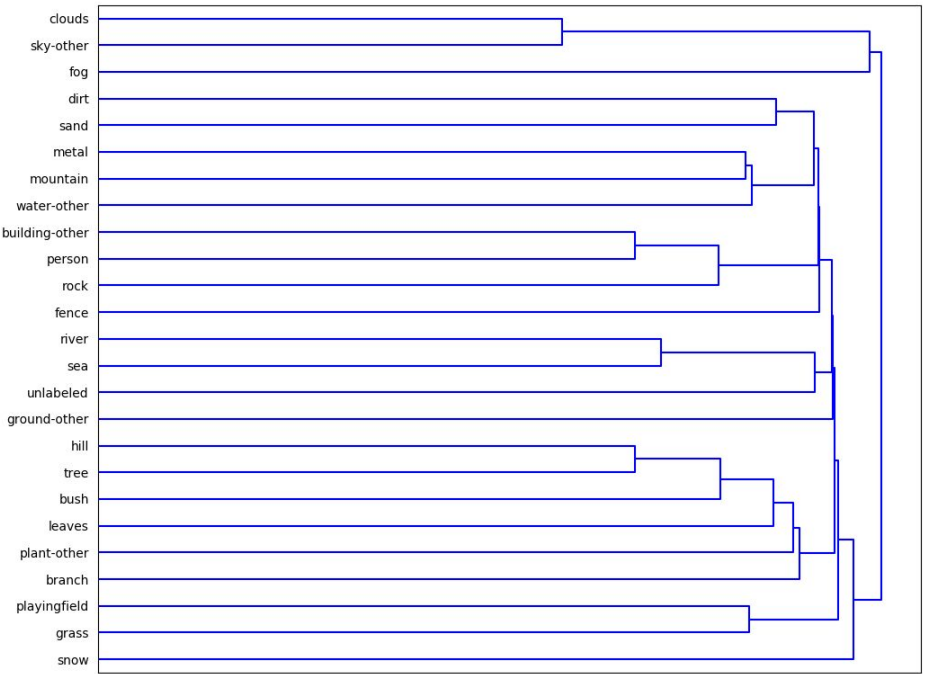

Fig. 2. Hierarchical clustering on semantics by using each row of transfer matrix (Fig. 1) as feature vector (See Sec. 2). Our learned attention can discovery similar semantics such as {`clound`, `sky-other`}, {`river`, `sea`}, {`dirt`, `sand`} without additional semantic-level supervision.

3 The Impact of Semantic Similarity

To show the impact of semantic similarity to example-guided image synthesis, we conduct a style swapping experiment. We randomly select 2500 exemplars and targets from the COCO-stuff outdoor scene dataset to test style swapping. We use label histogram intersection to measure the semantic similarity between the reference image and target label map. To measure the style transfer performance

of different approaches, we apply several metrics such as: 1) style loss on VGG conv1&conv2 ($\mathcal{L}_{style}$), 2) averaged nearest neighbor distance from output to exemplar at each location on pixel space (NN_pixel), 3) averaged nearest neighbor distance from output to exemplar at each location on VGG conv1 feature space (NN_conv1).

Methods	$\mathcal{L}_{style} \downarrow$	NN_pixel↓	NN_conv1↓
SPADE_VAE	1.13e-03	6.24e-03	1.80e+00
ours_GAP	1.02e-03	1.01e-02	1.89e+00
ours	**8.08e-04**	**3.91e-03**	**1.68e+00**

Table 1. The averaged style transfer performances on 2500 random exemplar-label map pairs. Our approach performs the best among two comparative baselines.

4 Results on an Additional Dataset

We also train our model on an additional indoor scene dataset. Specifically, we collect 26713 diversified indoor scene images from the COCO-stuff dataset for training. As shown in Fig. 3, our model is able to consistently transfer style across different indoor scenes, showing that our approach can effectively generalize to other datasets with complex structures.

5 More Scenes Swapping Results

We show our style swapping results on 12 diversified scenes in Fig. 4 and compare them against the style swapping results of SPADE_VAE and GAP (Fig. 5 and Fig. 6 respectively). As shown in the figures, our model can successfully transfer and preserve style for each scene semantics region. Note that our model can infer the style of river (column 2) from other visual content of the exemplar images that does not contain water. Also, our model can precisely transfer styles for the foreground animals.

6 The Synthesis Module

As shown in Fig. 7, our image synthesis module (the dash block on the right) takes the image features map $F_{x,1}^{(i)}$ and segmentation features map $F_{c,1}^{(i)}$ as inputs to output a new image $\hat{x}_{2\to1}$. Specifically, at each scale, a SPADE residue block [1] with upsampling layer takes the concatenation of $F_{x,1}^{(i)}$ and $F_{c,1}^{(i)}$ as input to generate an upsampled feature map or image.

7 Patch Sampling Strategy and Training Details

We employ the following patch sampling strategy during training. In the first 20 epochs during training, we sample two 256×256 patches from the 512×512 global

Fig. 3. Style-structure swapping on 8 arbitrary indoor scenes at resolution 256×256. Our model can generalize across recognizably different scenes of different semantics, and synthesize images with reasonable and consistent styles. Please zoom in for details.

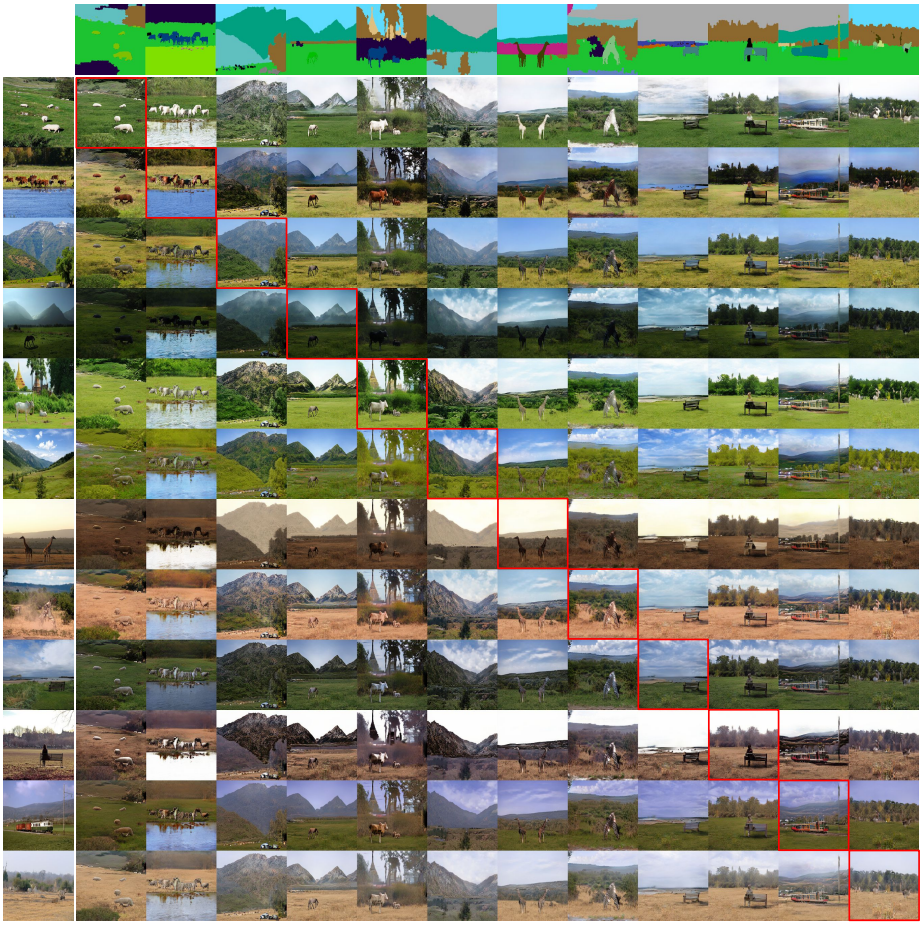

Fig. 4. Style-structure swapping results of our model. Results are generated on 12 arbitrary scenes at resolution 256×256. Our model can generalize across recognizably different scenes of different semantics, and synthesize images with reasonable and consistent styles. Our model can effectively transfer the color of foreground animals. Also note that our model can implicitly infer the style of river (column 2) from exemplar images without relying on water-related visual contents. Please zoom in for details.

Fig. 5. Style-structure swapping results of `SPADE_VAE`. Results are generated on 12 arbitrary scenes at resolution 256×256.

Fig. 6. Style-structure swapping results of `ours_GAP`. Results are generated on 12 arbitrary scenes at resolution 256×256. Note that with the global average pooling, style of individual regions are not well-transferred (e.g. sky in the second last row).

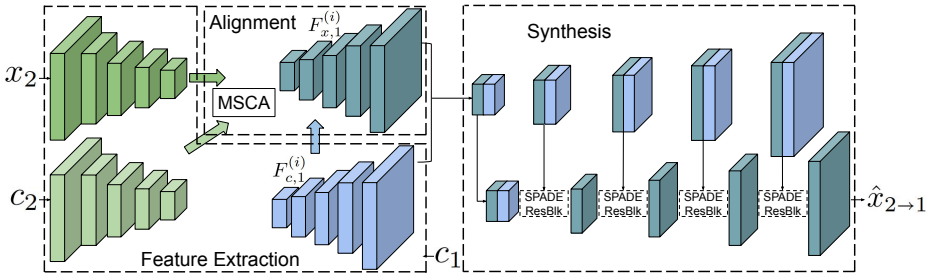

Fig. 7. The details of the image synthesis module (the dash block on the right). The image synthesis module takes image features maps $F^{(i)}_{x,1}$ and segmentation features maps $F^{(i)}_{c,1}$ at all scale i as inputs to output a new image $\hat{x}_{2\to1}$. Multiple SPADE residue blocks [1] with upsampling layers are used to upsample the spatial resolutions.

images. Then we perform random collision avoidance in horizontal direction or vertical direction. To help our model generalize better across scenes that have larger variances, we employ only the patch-scale cross-reconstruction task at the first 20 epochs.

During the next 20 epochs of training, we randomly crop 384×384 patches from the 512×512 global image, and resize patches to 256×256 to generate patches for the patch-scale cross-reconstruction task. The enlarged patch scale can potentially helps our model to generalize at the global scale. In this stage, we employ global image to facilitate better global scale synthesis. Random flipping is employed to augment data during training.

8 MSCA Pretraining

As shown in Fig. 8, an auxiliary feature decoder (the dash block on the right) is used to pretrain the feature extractors and the MSCA modules. Specifically, at each scale, the concatenation of $F^{(i)}_{x,1}$ and $F^{(i)}_{c,1}$ at each scale is fed into a 1×1 convolutional layer to reconstruct the ground-truth VGG feature of x_1 at the corresponding scale. We weighted sum the L1 losses between predictions and ground-truth at each scales, then apply backpropagation to update weights of the whole model. We pretrain the model for 20 epochs. Because of the light-weight design of the feature decoder, the pretraining step only takes around 12 hours, and around 2% of the total training time. During the MSCA pretraining, we crop non-overlapping, 256×256 patches from the 512×512 global images.

References

1. Park, T., Liu, M.Y., Wang, T.C., Zhu, J.Y.: Semantic image synthesis with spatially-adaptive normalization. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2019)

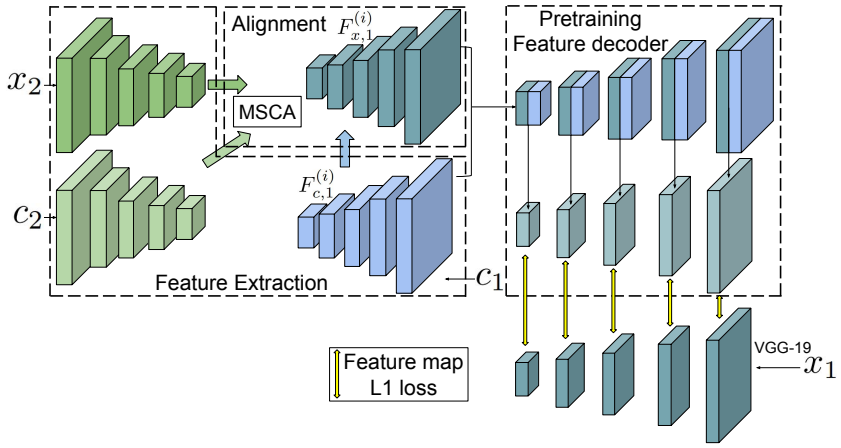

Fig. 8. The details of the auxiliary feature decoder for feature extractor and MSCA pretraining (dash block on the right). At each scale i, the image features map $F_{x,1}^{(i)}$ and the segmentation features map $F_{c,1}^{(i)}$ are concatenated and feed to a 1×1 convolution layer to predict the VGG-19 features map of x_1.