6. Supplementary Material

6.1. Problem

The current state-of-the-art methods for text-to-image synthesis normally have two sources of randomness: one for the text embedding variability, and the other (noise z given a normal distribution) capturing image variability. Our empirical investigation of some previously published methods reveals that those two sources can overlap: due to the randomness in the text embedding, the noise vector z then does not meaningfully contribute to the variability nor the quality of generated images, and can be discarded. This is illustrated qualitatively in Figure 8 and quantitatively in Figure 9.

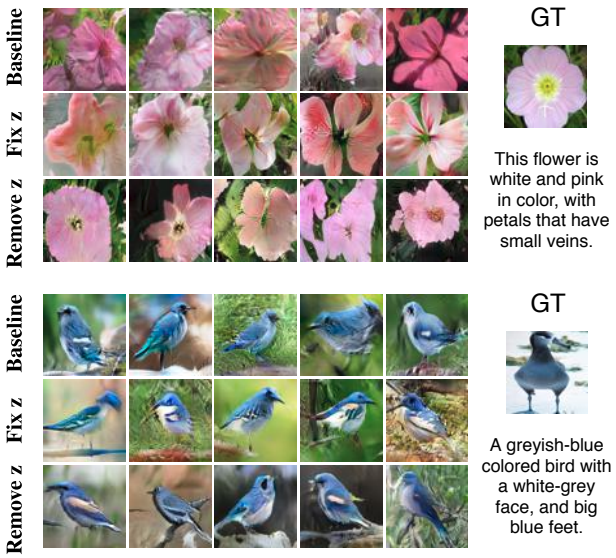

Figure 8: Generated images from previous state-of-the-art method (Baseline), fixing the noise vector z in the baseline method (Fix z) and removing the noise vector z in the baseline method (Remove z). The removal of the randomness from the noise source by either fixing z or removing z does not affect the variability nor the quality of generated images, indicating that the noise vector z has no contribution in the synthesis process.

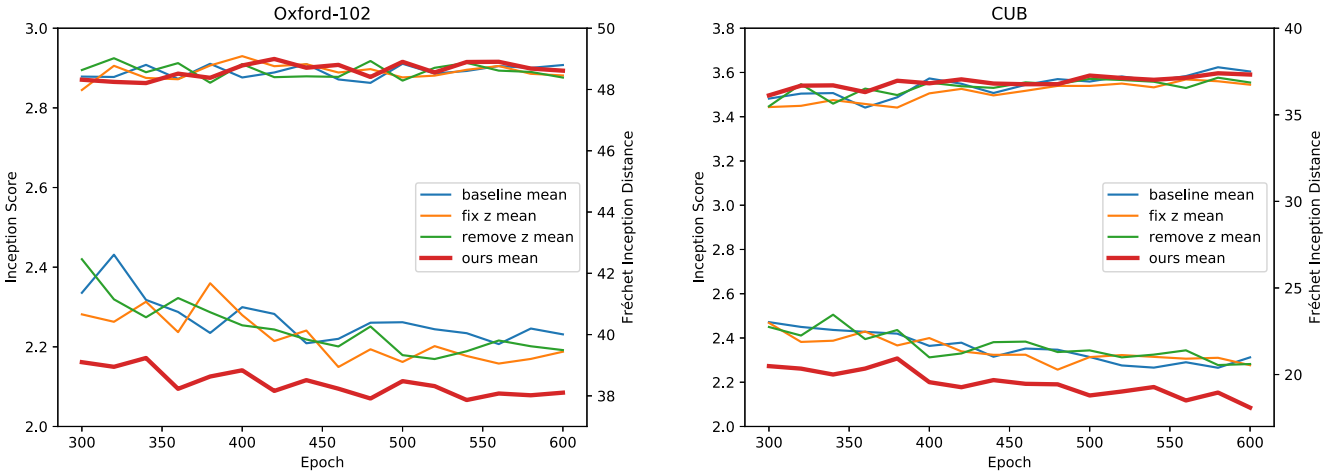

Figure 9: Inception score (left axis, top curves) and FID (right axis, bottom curves) for the baseline method, its variants (fix z and remove z) and our method on Oxford-102 (left) and CUB (right) datasets. Each curve is the mean of three independent experiments. Higher inception score and lower FID mean better performance.

6.2. Implementation details

For the encoder $G_{z,c}$, we first extract a 1024-dimension feature vector from a given image $\boldsymbol{x}$ (note that the text embeddings are also 1024-dimension vectors), and apply the reparameterization trick to sample $\hat{\boldsymbol{z}}$ and $\hat{\boldsymbol{c}}$. To reduce the complexity of our models, we use the same discriminator for both $D_{x,z}$ and $D_{x,c}$ in our experiments, thus re-denoted as $D_{(x,z)/(x,c)}$, and the weights are shared between D_x and D_{x,φ_t}. By default, $\lambda = 4$. For the training, we iteratively train the discriminators D_{x,φ_t}, $D_{(x,z)/(x,c)}$, $D_{x,x'}$ and then the generators G_x, $G_{z,c}$ for 600 epochs (Oxford-102 and CUB) or 200 epochs (COCO), with Adam optimization [15]. The initial learning rate is set to 0.0002 and decreased to half of the previous value for every 100 epochs.

6.3. Datasets

The Oxford-102 dataset [23] contains 8,189 flower images from 102 different categories, and the CUB dataset [29] contains 11,788 bird images belonging to 200 different categories. For both datasets, each image is annotated with 10 text descriptions provided by [25]. Following the same experimental setup as used in previous works [26, 32, 35], we preprocess and split both datasets into disjoint training and test sets: $82 + 20$ classes for the Oxford-102 dataset and $150 + 50$ classes for the CUB dataset. For the COCO dataset [20], we use the 82,783 training images and the 40,504 validation images for our training and testing respectively, with each image given 5 text descriptions. We also use the same text embeddings pretrained by a char-CNN-RNN encoder [25].

6.4. Evaluation metrics

Three quantitative metrics are used to evaluate our method: Inception score [28], Fréchet inception distance (FID) [10] and Visual-semantic similarity [35]. Inception score focuses on the diversity of generated images. We compute inception score using the fine-tuned inception models for both Oxford-102 and CUB datasets provided by [32]. FID is a metric which calculates the distance between the generated data distribution and the real data distribution, through the feature representations of the inception network. Similar to previous works [32, 35], we generate $\sim$ 30,000 samples when computing both inception score and FID. Visual-semantic similarity measures the similarity between text descriptions and generated images. We train our neural distance models for 64×64 resolution images similar to [35].

6.5. Choices of reconstruction loss

In addition to the adversarial loss $V_{cycle}(D, G)$ defined in Section 3.3, we also examine the model performance by using either l_1 or l_2 loss for cycle-consistency. However, the degradation is unexpectedly dramatic. Figure 10 shows the generated images using adversarial loss compared with those using l_2 loss, and it is clear that the latter gives much blurrier images. A similar effect is observed when using l_1 loss. The quantitative results are shown in Table 2.

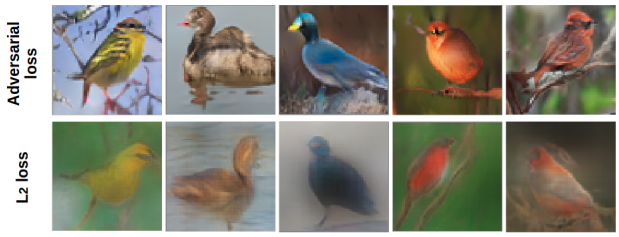

Figure 10: Examples of generated images by using either adversarial loss or l_2 loss for the cycle consistency.

6.6. Visual-semantic similarity

Visual-semantic similarity measures the similarity between text descriptions and generated images. We train our neural distance models for 64×64 resolution images similar to [35] for 500 epochs. In the testing phase, we generate $\sim$ 30,000 images at 64×64 resolution and compute the visual-semantic similarity scores based on the trained neural distance models. The real images are also used to compute the visual-semantic similarity scores as the ground truth. Table 3 shows the comparison of the baseline method, our method and the ground truth. Similar to inception score and FID, we provide mean scores for visual-semantic similarity metric based on three independent experiments at five different epochs (600, 580, 560, 540 and 520). As shown in the table, our method achieves comparable results for visual-semantic similarity compared to the baseline method.

Method	Dataset	
	Oxford-102	CUB
Ground Truth	0.422 ± 0.117	0.382 ± 0.154
HDGAN mean*	0.248 ± 0.136	0.263 ± 0.164
Ours mean*	0.246 ± 0.139	0.251 ± 0.168

* mean calculated on three experiments at five different epochs

Table 3: Visual-semantic similarity.

6.7. Relation to previous work on the separation of content and style for text-to-image synthesis

Reed *et al*. [26] have also investigated the separation of content and style information. However, their learning of style is detached from the text-to-image framework, and the parameters of the image generator are fixed during the style learning phase. Their concept of content and style separation is therefore not actually leveraged during the training of the image generator. In addition, their work uses a deterministic text embedding, which cannot plausibly cover all content variations, and as a result, one can assume that information belonging to the content could severely contaminate the style. In our work, we learn style from the data itself as opposed to the generated images. This allows us to learn style while updating the generator and effectively incorporate style information from the data into the generator.

6.8. More examples on text-to-image generation

We provide more text-to-image generation examples in Figure 11. For each method, we generate three groups of images given a certain text description: 1) use a fixed $\boldsymbol{z}$ and sample $\boldsymbol{c}$ from $p(\boldsymbol{c})$ to show the role of $\boldsymbol{c}$ in the generated images; 2) use a fixed $\boldsymbol{c}$ and sample $\boldsymbol{z}$ from $p(\boldsymbol{z})$ to examine how $\boldsymbol{z}$ contributes to image generation; 3) sample both $\boldsymbol{z}$ and $\boldsymbol{c}$ from $p(\boldsymbol{z})$ and $p(\boldsymbol{c})$ respectively to evaluate the overall performance. Note that $p(\boldsymbol{c})$ is conditioned on the text description as defined in Section 3.1. As seen in Figure 11, our model generates better-quality images and gives more meaningful variability in style compared to the baseline, when we keep the content information constant and only sample from the latent variable $\boldsymbol{z}$. This is not surprising due to the fact that the generator has learned to match changes in $\boldsymbol{z}$ with style. Note that for most of the 'easy' tasks (images that are easy for the generator to synthesize), visual comparison does not reveal significant differences between the baseline and our method.

Figure 11: Examples of generated images on Oxford-102 dataset compared with the baseline method using three different strategies: 1) use a fixed $\boldsymbol{z}$ and sample $\boldsymbol{c}$ from $p(\boldsymbol{c})$ to show the role of $\boldsymbol{c}$ in the generated images; 2) use a fixed $\boldsymbol{c}$ and sample $\boldsymbol{z}$ from $p(\boldsymbol{z})$ to examine how $\boldsymbol{z}$ contributes to image generation; 3) sample both $\boldsymbol{z}$ and $\boldsymbol{c}$ from $p(\boldsymbol{z})$ and $p(\boldsymbol{c})$ respectively to evaluate the overall performance. Note that $p(\boldsymbol{c})$ is conditioned on the text description. The conditioning text descriptions and their corresponding images are shown in the left column.

6.9. More examples on disentanglement analysis

In this subsection, we provide more results and discussions on the disentanglement analysis with the Bernoulli constraint (see Section 4.5). Section 6.9.1 gives more interpolation results based on inferred content ($\hat{c}$) and inferred style ($\hat{z}$); Section 6.9.2 and Section 6.9.3 respectively show style transfer results based on real images and synthetic images with specifically engineered styles.

6.9.1 Interpolations on content and style

For interpolations, we provide our trained inference model with two images: the source image and the target image, to extract their projections $\hat{z}$ and $\hat{c}$ in the latent space. As shown in Figure 12, Figure 13, Figure 14 and Figure 15, the rows correspond to reconstructed images of linear interpolations in $\hat{c}$ from source to target image and the same for $\hat{z}$ as displayed in columns. The source images are shown in the top left corner and the target images are shown in the bottom right corner. The figures demonstrate that our proposed dual adversarial inference is able to separate the learning of content and style, and moreover, the learned style $\hat{z}$ indeed represents certain meaningful information. In particular, Figure 12 shows an example of style controlling the number of petals, and the pose of flowers from facing towards upright to facing front; Figure 13 shows a smooth transition of style from a single flower to multiple flowers; Figure 14 shows the changing of the bird pose from sitting to flying; and finally Figure 15 shows the switch of the bird pose from facing towards right to facing towards left and the emergence of tree branches in the background.

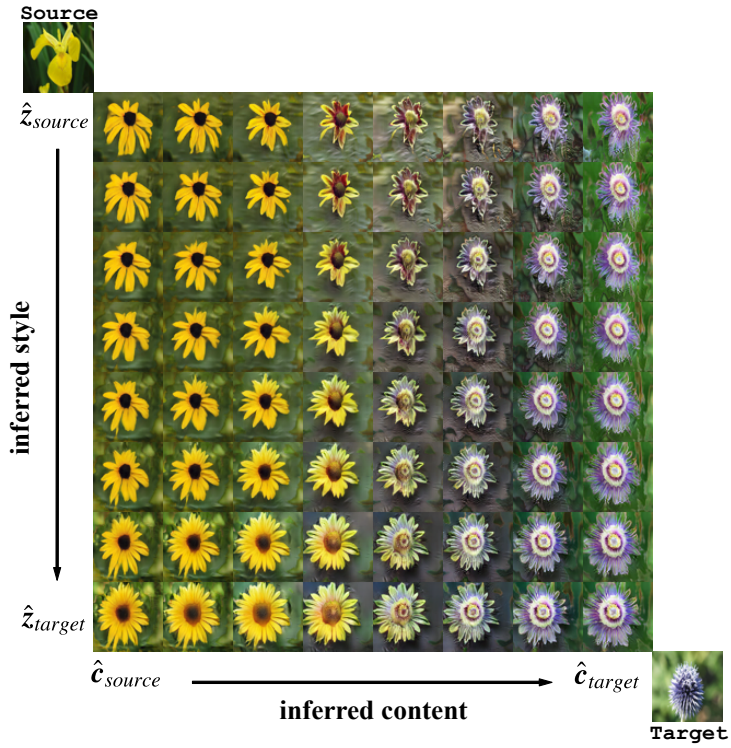

Figure 12: Example of inferred style controlling the number of petals, and the pose of flowers from facing towards upright to facing front.

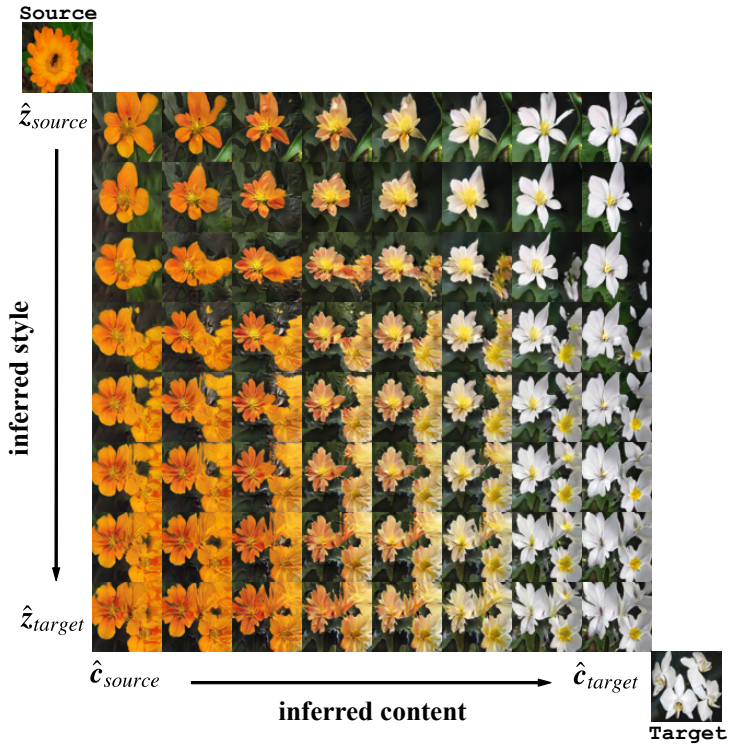

Figure 13: Example of inferred style controlling the number of flowers, from a single flower to multiple flowers.

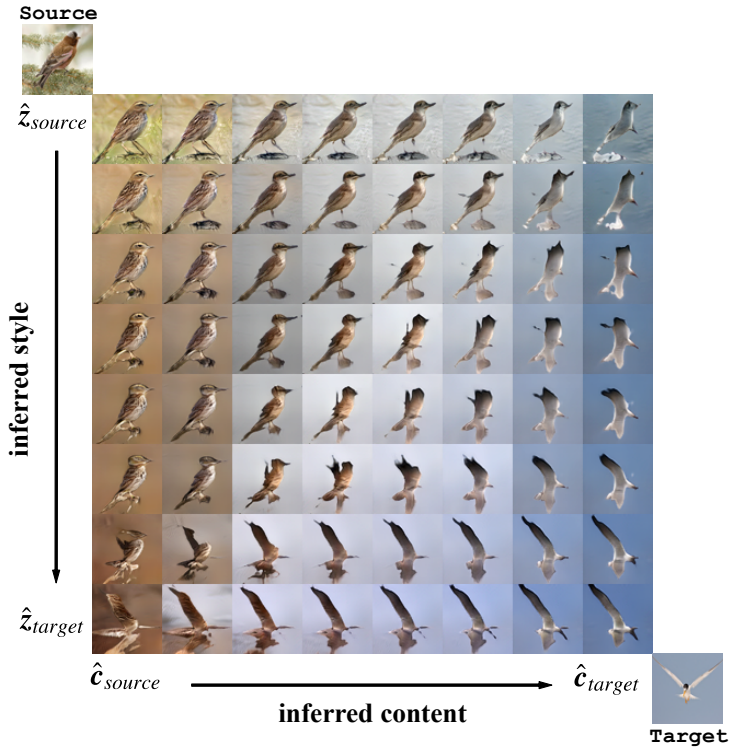

Figure 14: Example of inferred style controlling the pose of birds from sitting to flying.

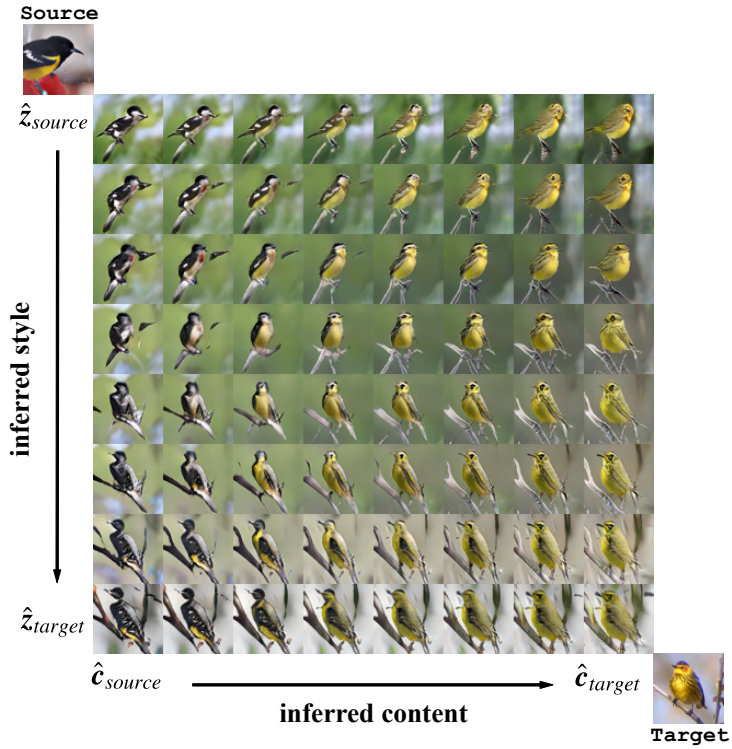

Figure 15: Example of inferred style controlling the pose of birds from facing towards right to facing towards left and the emergence of tree branches in the background.

6.9.2 More style transfer results

Here, we provide more style transfer results in Figure 16 (Oxford-102) and Figure 17 (CUB). Each column uses the same style inferred from a style source, and each row uses the same text description as the content source. The style sources that are shown in the top row, and the corresponding real images for the content sources are shown in the leftmost column.

Figure 16: Disentangling content (in rows) and style (in columns) on Oxford-102 dataset by using content sources from text descriptions. The style sources are shown in the top row, and the corresponding real images for the content sources are shown in the leftmost column.

Figure 17: Disentangling content (in rows) and style (in columns) on CUB dataset by using content sources from text descriptions. The style sources are shown in the top row, and the corresponding real images for the content sources are shown in the leftmost column.

6.9.3 Style interpolations with synthetic style sources

To further validate the disentanglement of content and style, we artificially synthesize images that have certain desired known styles, *e.g.*, a flower or multiple flowers located in different locations (top left, top right, bottom left or bottom right), and perform linear interpolation of $\hat{z}$ from two different style sources. As shown in Figure 18, the flower can move smoothly from one location to another, and the number of flowers can grow smoothly from one to two, suggesting that a good representation of style information has been captured by our inference mechanism.

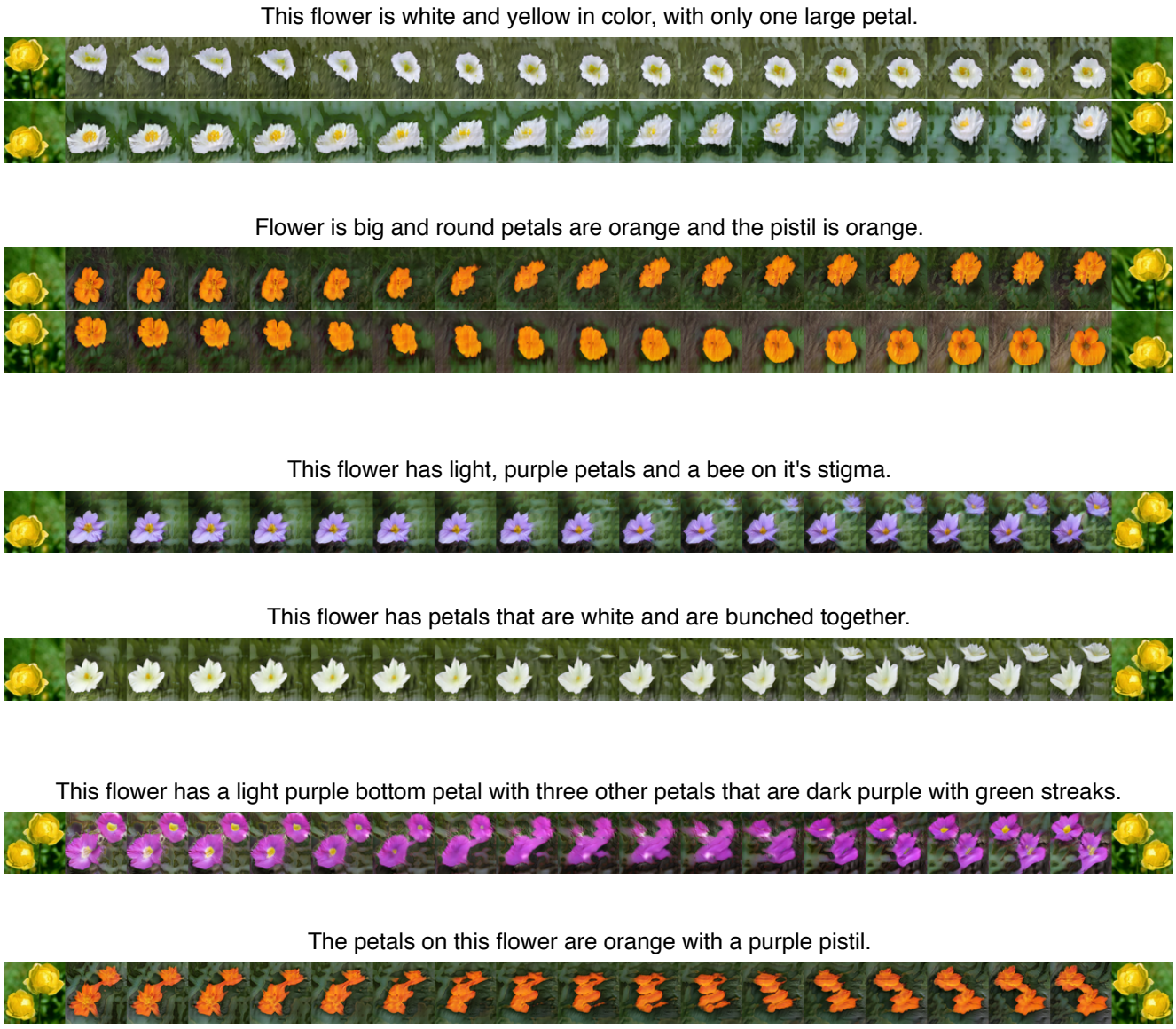

Figure 18: Style interpolations with synthetic style sources: the moving flowers and the growing quantities of flowers.