Supplementary: End-to-End Diffusion Latent Optimization Improves Classifier Guidance

Bram Wallace
Salesforce AI
b.wallace@salesforce.com

Akash Gokul
Salesforce AI
agokul@salesforce.com

Stefano Ermon
Stanford University
ermon@cs.stanford.edu

Nikhil Naik
Salesforce AI
nnaik@salesforce.com

This supplementary materials contains

1. Results for improving aesthetic appeal in text-to-image generation (Section 1)
2. Further analysis of personalization results (Section 2)
3. Details of the DOODL optimization process (Section 3)
4. Additional qualitative samples (Section 4)

1. Text-to-Image Generative Aesthetics Results

We investigate using DOODL to guide text-to-image generation using the aesthetic scoring model. The standard algorithm is used with $\lambda_{DOODL} = 0.1$. Qualitative results are shown in Figure 1.

We additionally perform human evaluation using the same comparison method as in the main paper, asking *Please select the image which you think would be preferred visually by the majority of people*. We sample 8 prompts over 16 seeds:

- A surfer catching a wave
- A unicorn in forest
- A stained glass window
- Yosemite valley
- A dramatic photo from the surface of mars
- A cottage in the countryside
- A river at sunrise
- A dog with a chewtoy

Quantitative results against the original generation are shown in Figure 2. We find that DOODL generations are consistently rated as having higher aesthetic appeal vs. the original generations with a win-draw-loss rate of $0.58-0.11-0.31$. We perform the same experiment against baseline classifier guidance with $\lambda_{Baseline} = 10$. We find the signal to be less consistent, with a win-draw-loss rate for DOODL of $0.48-0.07-0.45$ across the 128 comparisions. We hypothesize that this is possibly attributable to the value that the aesthetic model places on vivid colors and contrast; lower-level features that don't neccessarily require the precision of DOODL's approach vs. more approximate control. DOODL also is more prone to warp or deform content than the baseline guidance in performing the model-based optimization, which is typically visually unappealing. Further stabilizing DOODL with respect to the latter point would serve to broaden the above performance gap.

2. Further Analysis of Personalization Results

In the main text, we presented how often the majority (2/3) labelers labeled the dog as appearing to be the original dog in the desired context. The responses labelers chose from were:

1. The second image does not match the prompt, or is highly unrealistic
2. The dog in the second image does not look like the original dog
3. The dog in the second image looks similar to the original dog but there are significant differences
4. The dog in the second image appears to be the original dog

We present several different views of the data, all confirming that DOODL achieves far superior performance to the baselines and opens a door to a new family of guidance-based personalization methods.

Aggregate Statistics 4.5% of the original generations were labeled (4), as opposed to 5.6% for the baseline and 19.4% for DOODL.

Figure 1: Qualitative Aesthetic Results. Prompts: *A unicorn in forest*($\times 3$), *A dog with a chewtoy*, *A river at sunrise*, *A dramatic photo from the surface of Mars*.

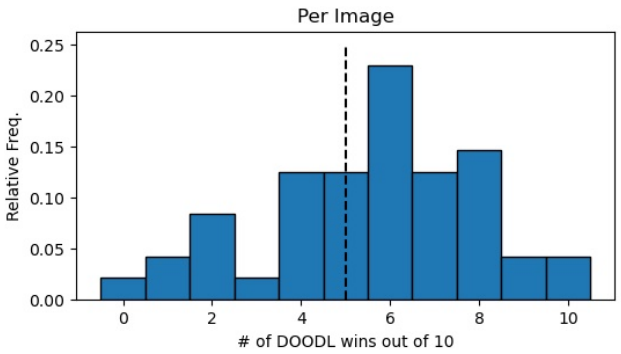

Figure 2: Human Aesthetic Results. 10 independent labelers are instructed *Please select the image which you think would be preferred visually by the majority of people*. A "No preference" response is given which we count as half a response in each direction. The per-image DOODL wins out of 10 is shown. The win-draw-loss rates of DOODL are $0.58 - 0.11 - 0.31$.

Unanimous Agreement Due to the challenging problem and noisiness in the labeling process, very few images were unanimously classified as (4). 3.1% (3/96) were for DOODL and 0 for the other two methods.

Lowering Similarity Bar We visualize the success rates if allowing responses (3) or (4) (so the dog must look similar but differences are allowed) in Table 1.

3. Details on Multicrop in DOODL Optimization

Here we precisely describe the multicrop augmentation used in the DOODL optimization process. We employ the same MakeCutouts class as used in the diffusers library[1], with code in Figure 3. We use `cut_power`=0.3. The size of the square crop is sampled by generation a random value $r \sim U(0,1)$ and crop size $ModelInputSize + (OriginalImageSize - ModelInputSize)r^{CutPower}$. A crop of this size is then uniformly selected from the image.

4. Additional Qualitative Results

Additional qualitative results are given in the below listed figures.

- Figure 4 for Drawbench generations
- Figure 5 for additional FGVC results
- Figures 6 to 9 for randomly chosen aesthetic editing on COCO from the result set used in human evaluation for Section 5.3.
- Figures 10 to 17 for further personalization results across source images and target captions with random seeds.

References

[1] Patrick von Platen, Suraj Patil, Anton Lozhkov, Pedro Cuenca, Nathan Lambert, Kashif Rasul, Mishig Davaadorj, and Thomas Wolf. Diffusers: State-of-the-art diffusion models. 2, 3

Method	Original	Baseline Clf. Guidance	DOODL
Aggregate Statistics	12.2%	28.5%	52.1%
Majority Agreement	6.2%	25%	52.1%
Unanimous Agreement	0%	8.3%	25%

Table 1: Success rates for personalization with lowering criteria to *The dog in the second image looks similar to the original dog but there are significant differences*

```
class MakeCutouts(nn.Module):
    def __init__(self, cut_size, cut_power=1.0):
        super().__init__()

        self.cut_size = cut_size
        self.cut_power = cut_power

    def forward(self, pixel_values, num_cutouts):
        sideY, sideX = pixel_values.shape[2:4]
        max_size = min(sideX, sideY)
        min_size = min(sideX, sideY, self.cut_size)
        cutouts = []
        for _ in range(num_cutouts):
            size = int(torch.rand([]) ** self.cut_power * (max_size - min_size) + min_size)
            offsetx = torch.randint(0, sideX - size + 1, ())
            offsety = torch.randint(0, sideY - size + 1, ())
            cutout = pixel_values[:, :, offsety : offsety + size, offsetx : offsetx + size]
            cutouts.append(F.adaptive_avg_pool2d(cutout, self.cut_size))
        return torch.cat(cutouts)
```

Figure 3: Multicrop python code from [1] examples.

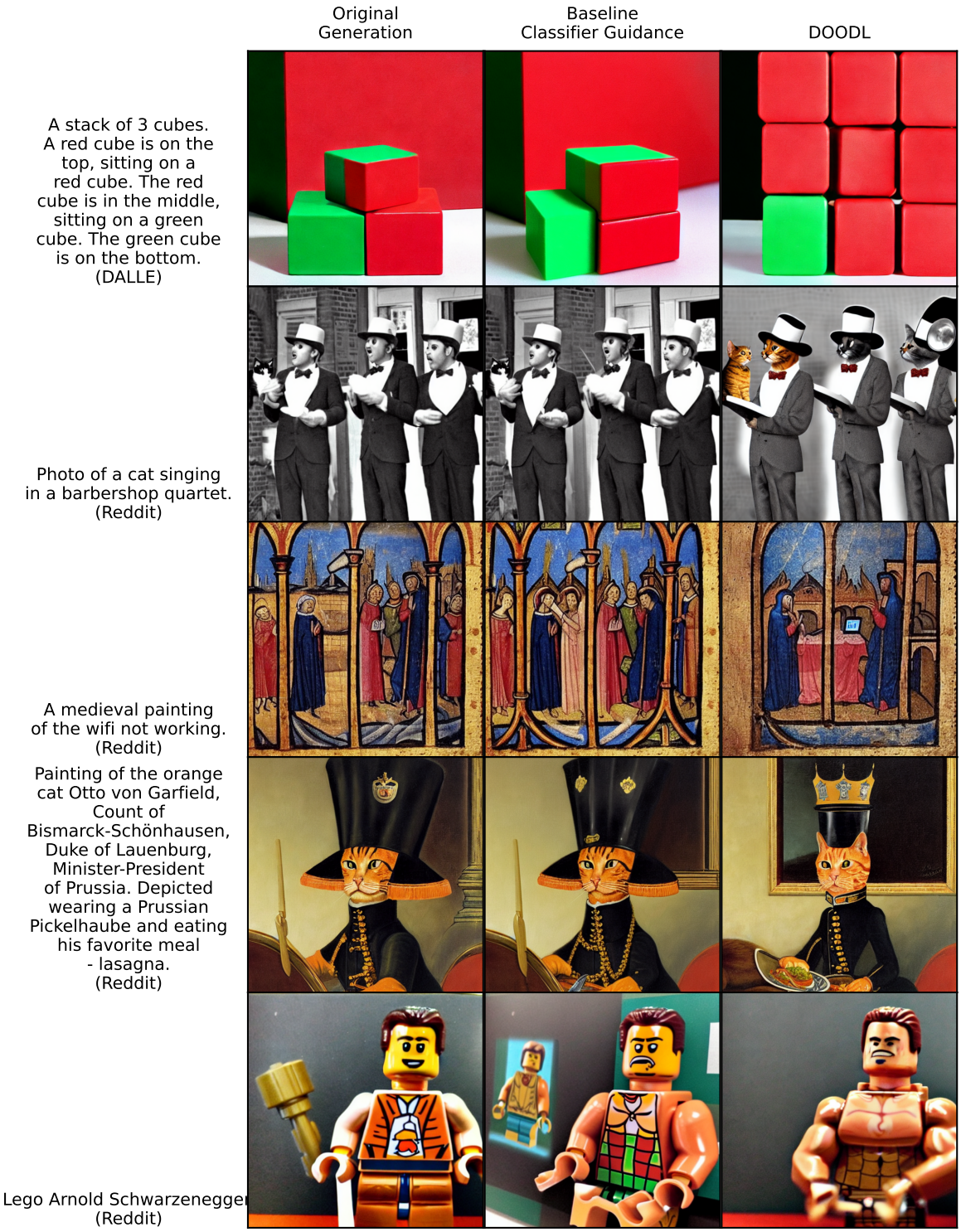

Figure 4: Additional Drawbench results

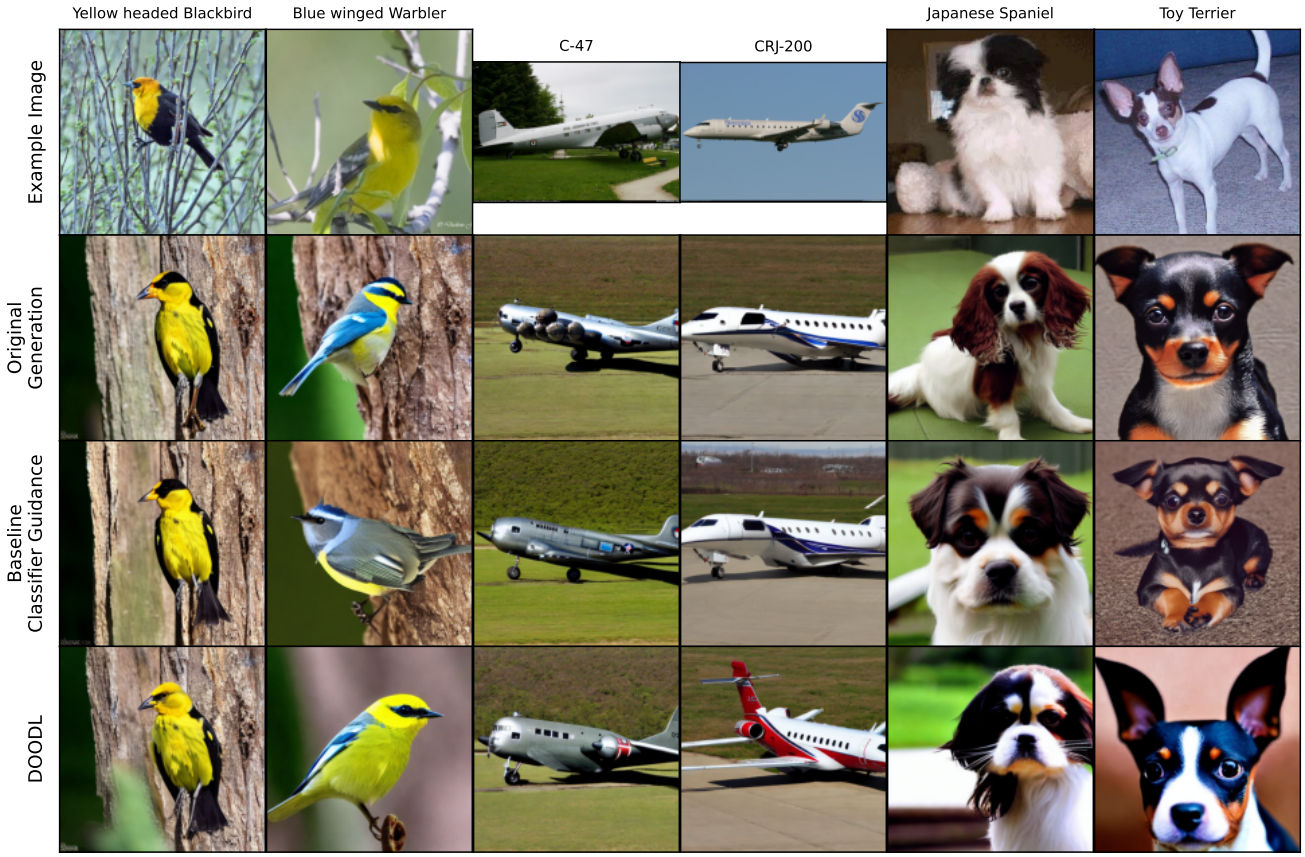

Figure 5: Additional FGVC results

Figure 6: Additional random COCO aesthetic editing results (1). No caption information is used

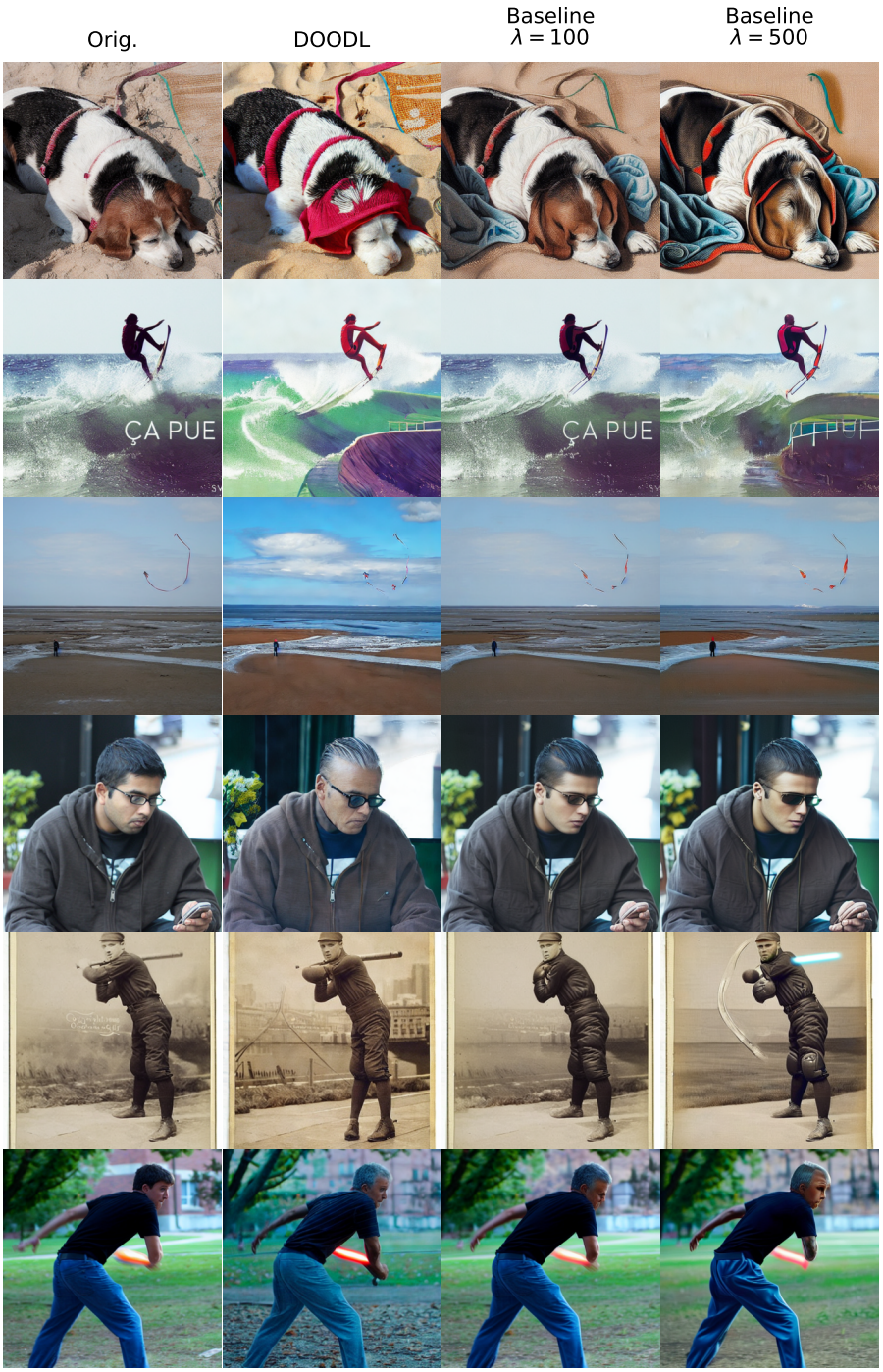

Figure 7: Additional random COCO aesthetic editing results (2). No caption information is used

Figure 8: Additional random COCO aesthetic editing results (3). No caption information is used

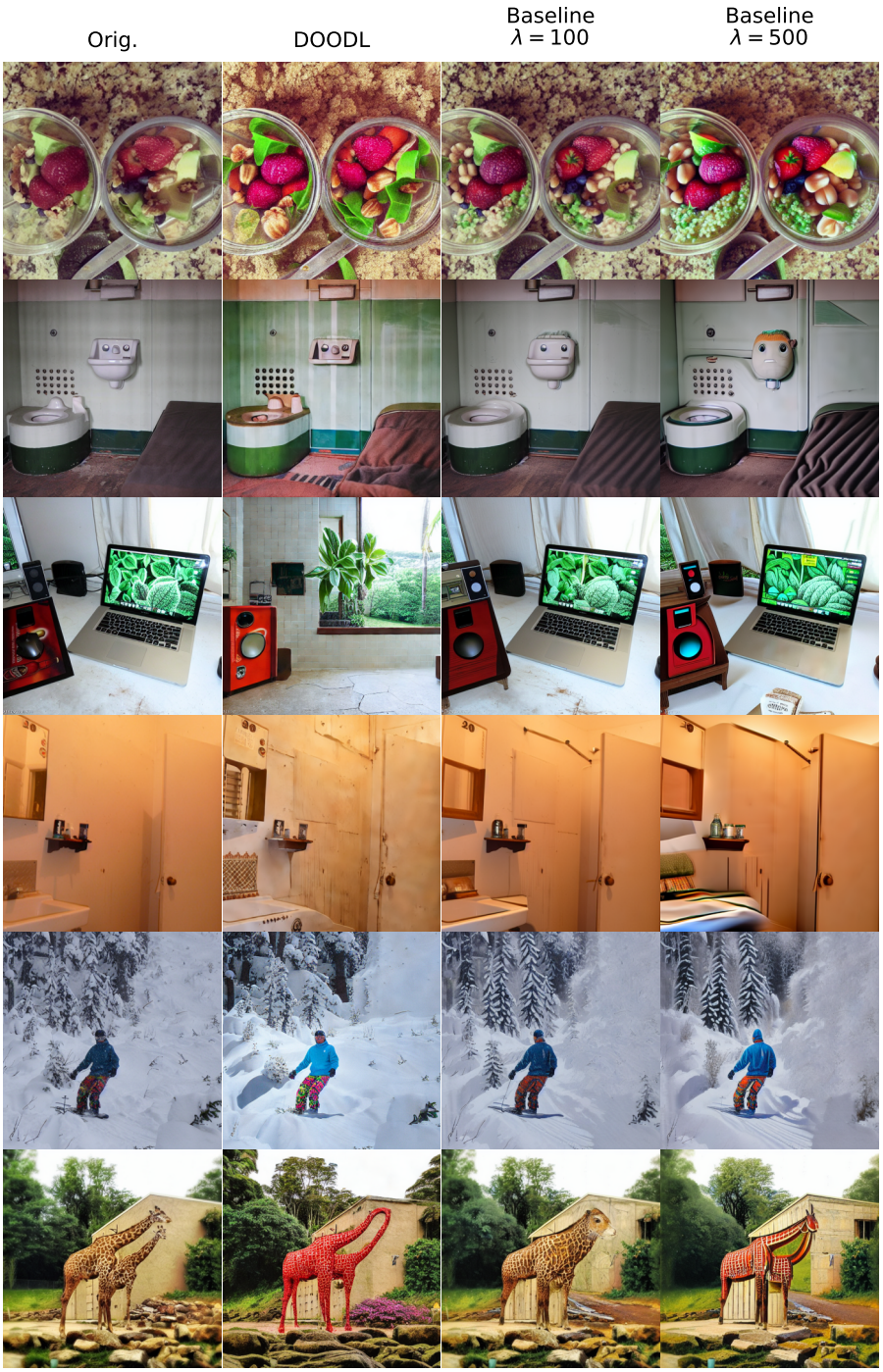

Figure 9: Additional random COCO aesthetic editing results (4). No caption information is used

Figure 10: Additional personalization results

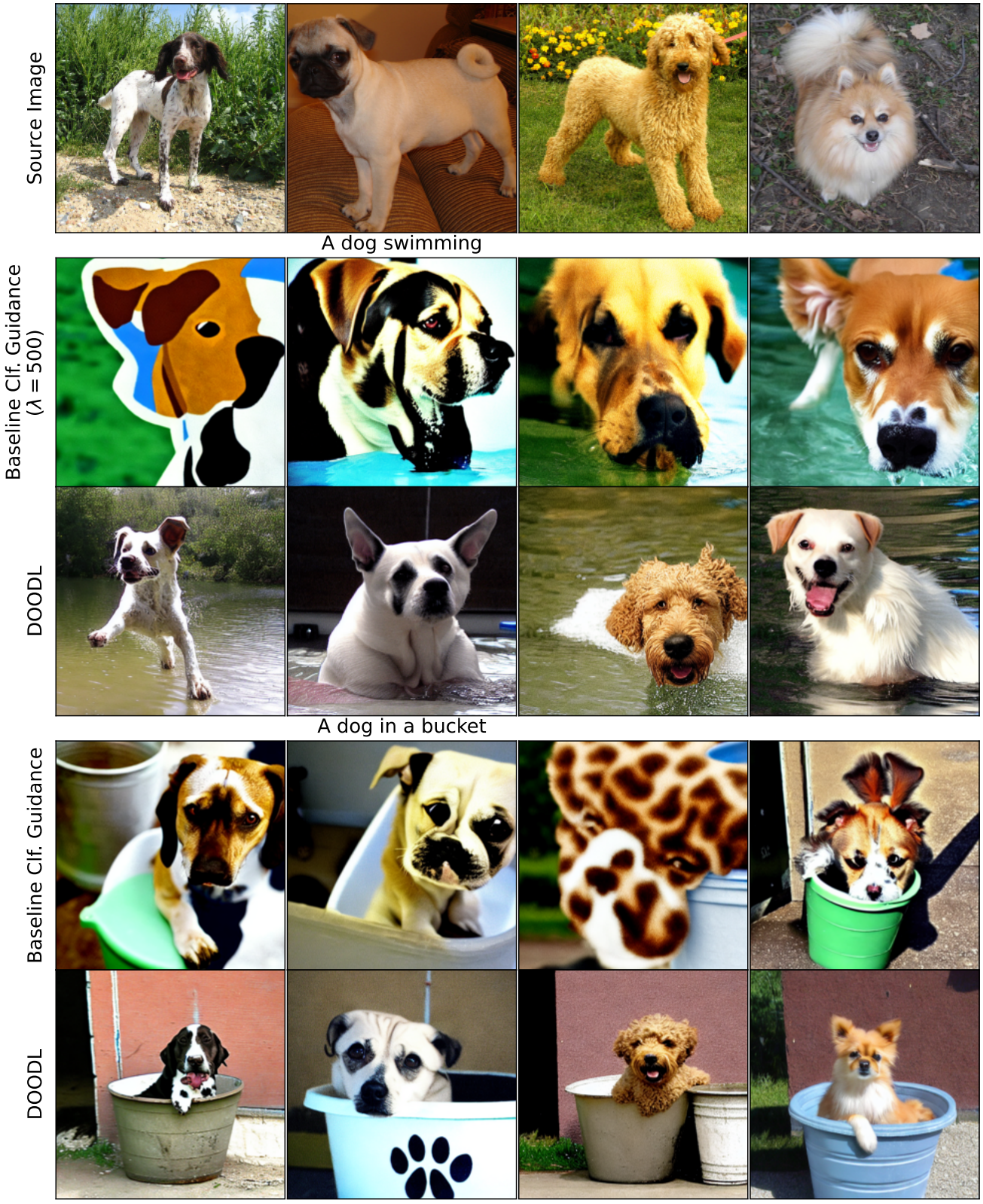

Figure 11: Additional personalization results

Figure 12: Additional personalization results

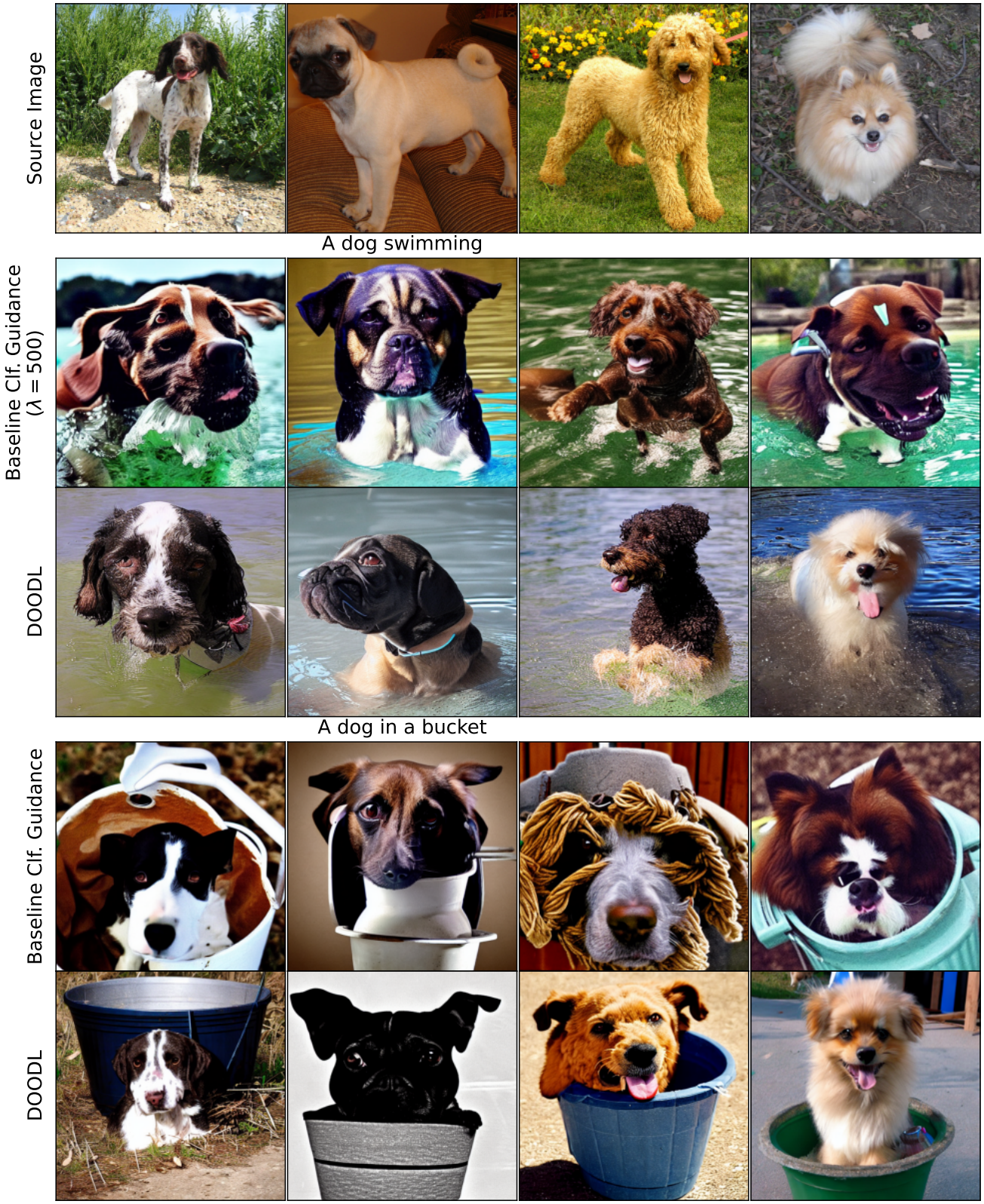

Figure 13: Additional personalization results

Figure 14: Additional personalization results

Figure 15: Additional personalization results

Figure 16: Additional personalization results

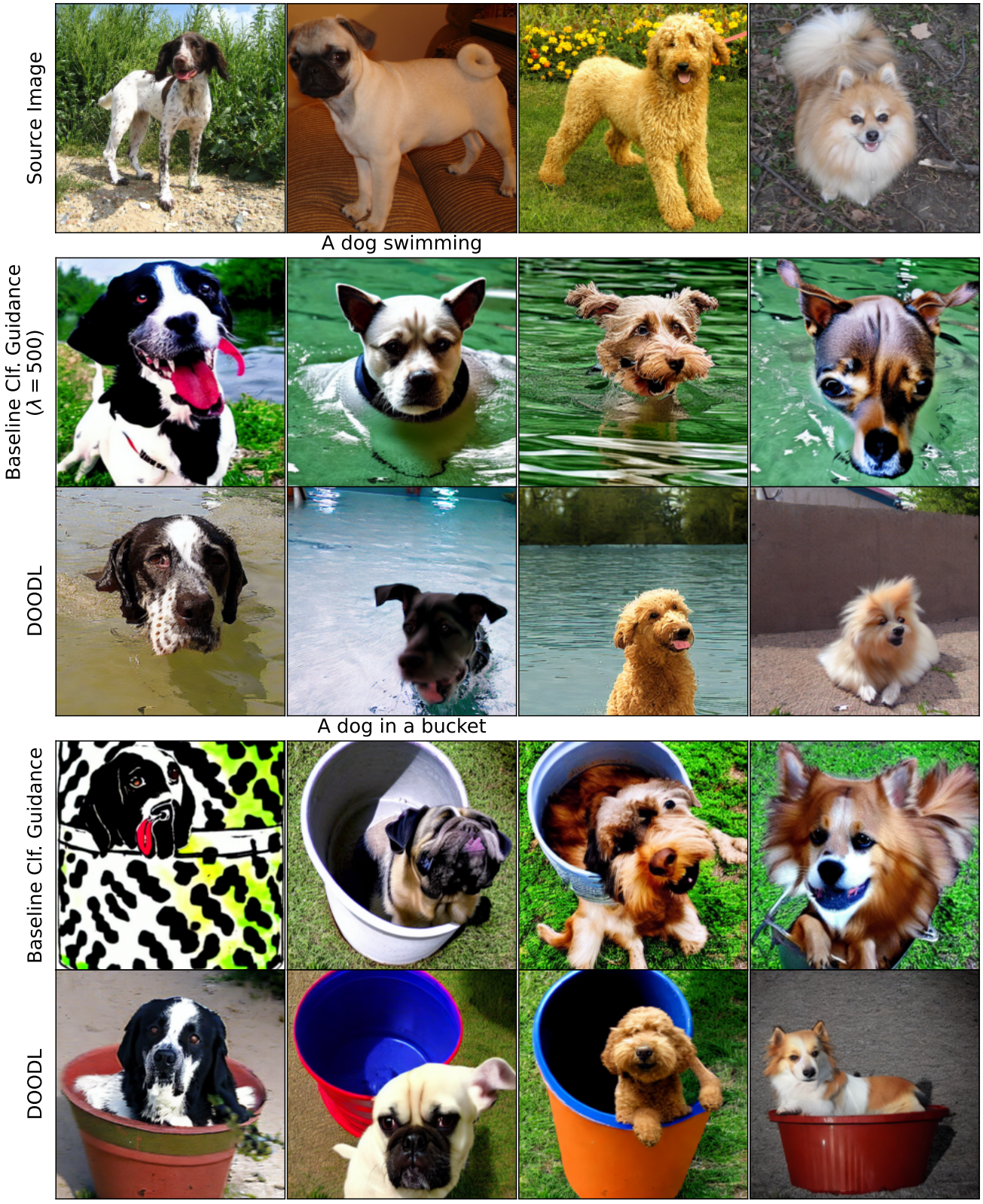

Figure 17: Additional personalization results