Reusing Computation in Text-to-Image Diffusion for Efficient Generation of Image Sets

Supplementary Material

A. Further Method Details

We provide pseudocode for our hierarchical diffusion algorithm. Our inputs are as follows: τ is a hyperparameter controlling the tradeoff between savings and quality, K is the total number of diffusion steps, C is our set of clusters obtained through agglomerative clustering, C^{scores} are the corresponding c^{score}s for each cluster in C, and Y is our set of all text prompts. The additional variables in line 18 are standard diffusion variables: a_k and b_k are scheduler coefficients, ϵ_θ is the diffusion model, and σ_{k-1} is a time-step-dependent standard deviation. Note, this algorithm assumes that $\tau \leq max_c(c^{score})$ otherwise a dummy parent node is required above the root node with a corresponding $c^{score} > \tau$. Our method returns the image from step K for each leaf cluster containing an individual prompt.

Algorithm 1 Hierarchical Diffusion.

```
1: Inputs: $\tau, K, C, C^{\text{scores}}, Y$
2: Let $\phi(k) = \tau \cdot \left(1 - \frac{k}{K}\right)$ assign scores to each diffusion
   step $k \in 1...K$ s.t. the scores of the $K$ steps are evenly
   spaced over the interval $[\tau, 0]$.
3: for $k$ in $1...K$ do:
4:     $C_{\text{k}} \leftarrow []$
5:     for $y$ in $Y$ do:
6:         $c' = \underset{c \in C}{\text{argmin}}\, c^{\text{score}}$ s.t. $[\mathcal{P}\text{arent}(c)]^{\text{score}} \geq \phi(k) \wedge$
   $y \in c$
7:         if $c'$ not in $C_{\text{k}}$ then
8:             $C_{\text{k}}$.append($c'$)
9:         end if
10:    end for
11:    for $c$ in $C_{\text{k}}$ do
12:        if k==1 then
13:            $\mathbf{x}^c_{k-1} \sim N(0, \mathbf{I})$
14:        else
15:            $\mathbf{x}^c_{k-1} \leftarrow \mathbf{x}^{\mathcal{P}\text{arent}(c)}_{k-1}$
16:        end if
17:        $\bar{y} = \text{mean}(y : y \in c)$
18:        $\mathbf{x}^c_k \sim \mathcal{N}\left(a_k \mathbf{x}^c_{k-1} - b_k \epsilon_\theta(\mathbf{x}^c_{k-1}; k-1, \bar{y}), \sigma^2_{k-1}\mathbf{I}\right)$
19:    end for
20: end for
21: return $\{\mathbf{x}^{\{y\}}_K : y \in Y\}$
```

B. Additional Experiments

Experiments on Additional Models. Due to the coarse-to-fine generation properties exhibited by models conditioned on a text-image prior, our approach works best on the Kandinsky and Karlo models which are trained in this manner (See Fig. 4). However, we can still achieve savings using more standard models such as SD 1.5 and models using the more modern DiT architecture and Flow Matching scheduler such as FLUX. While these models generate high frequency details earlier on in the diffusion process and thus leave less room for shared compute (see Fig. 2), we still achieve moderate savings over standard diffusion inference. For comparable quality results (better VQA scores in $\approx 50\%$ of the samples), we save up to **28.25%** and **23.73%** on compute when using SD 1.5 and FLUX, respectively. We report results for FLUX on the Style Variations dataset. SD 1.5 results are reported in the table below for all datasets.

Model	Dataset	GenAI B.	Prompt T.	Style V.	Animals
SD 1.5	Savings ↑	10.4%	28.25%	23.55%	11.70%
	Win % ↑	48%	50%	49%	50%

Diversity of Generations. Examining sample generations from both our approach and standard diffusion in Figures 1 and 6, we observe that our generations qualitatively display comparable diversity to the results from standard diffusion. Since our approach leverages similarities across prompts to save compute in the generation process, when used at small scales, our generations can exhibit high similarity between examples (see Fig. 8). However, as we increase the size of our datasets such as with the GenAI Bench or Prompt Templates datasets, our diversity significantly increases. To quantitatively evaluate generation diversity, we randomly sample 100 images from each generated dataset and compute the CLIP cosine similarity between all possible pairs. The reported result is the mean similarity. On larger datasets (GenAI Bench, Prompt Templates, and Animals which all have $\geq$ 500 examples), our method achieves comparable diversity to the standard approach. While our results on the Style Variations dataset are less diverse, this is expected since the dataset contains only 100 examples and the highly structured and similar prompts naturally lend themselves to less diverse generations.

Dataset	GenAI B.	Prompt T.	Style V.	Animals
Std. CLIP Diversity ↓	0.6082	0.6508	**0.8071**	0.6652
Our CLIP Diversity ↓	0.6027	0.6493	0.8374	0.6552

Visualization of Mean Embedding. We analyze the mean cluster embedding by using it to guide image generation.

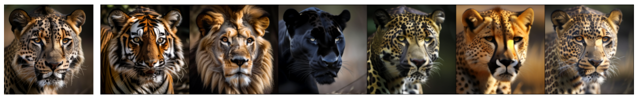

Figure 10. De-noised mean embedding (left) and its children using the Kandinsky model.

As we can see in Fig. 10 (zoom in for details), even though this embedding (left) is not associated with any particular prompt, it yields an image corresponding to a reasonable visual mixture of its children. In practice, this “mean image” is never generated (fully denoised) – we show it here as a tool to visualize the semantics of that embedding.