

A Multi-Scale Feature Smoothing Algorithm

Algorithm A illustrates the multi-scale feature smoothing, which is the core computational procedure for refining retrieved patch representations within their generation context. This algorithm ensures that retrieved visual elements are spatially and stylistically coherent with the surrounding image content through systematic multi-scale convolution operations.

The algorithm processes each retrieved patch representation $\hat{\mathbf{h}}_i$ independently, applying convolution operations at multiple scales ranging from 2×2 to $Q \times Q$ kernels. For each scale q , the algorithm initializes a temporary feature tensor $\mathbf{M} \in \mathbb{R}^{Q \times Q \times D}$ and an accumulation vector $\hat{\mathbf{h}}_q \in \mathbb{R}^D$. The nested loops over indices m and n systematically extract local patch features from different spatial windows around the target position (i, j) . Each extraction operation $\mathbf{H}_{\text{loc}}^l \leftarrow \mathbf{H}^l[i - m : i + q - m, j - n : j + q - n]$ captures a local neighborhood of size $q \times q$ centered at varying offsets from the target position.

The extracted local features undergo two-stage convolution processing. The first convolution operation $\text{Conv}_{q \times q}^1$ transforms the local patch features into an intermediate representation stored in \mathbf{M}_{mn} , effectively capturing contextual relationships within each local window. The second convolution operation $\text{Conv}_{q \times q}^2$ processes the accumulated intermediate features to produce scale-specific refined representations. This two-stage design enables the algorithm to first capture local contextual patterns and then integrate them into a coherent scale-specific feature representation.

After processing all scales for a given retrieved patch, the algorithm computes the final refined representation by averaging the scale-specific features. The normalization factor $(Q - 1)$ accounts for the number of scales processed, ensuring consistent feature magnitudes across different retrieved patches. This averaging operation effectively combines multi-scale contextual information into a single refined representation that preserves both fine-grained details from smaller kernel sizes and broader contextual patterns from larger kernel sizes. The resulting refined patch representations maintain spatial coherence with the surrounding generation context while preserving the essential visual characteristics of the retrieved content.

B Experiment Setup

B.1 RA-CM3 Implementation Details

Since the pretrained RA-CM3 model is not publicly available, we implement our own version following the methodology described in the original paper to serve as a representative baseline for image-level retrieval-augmented generation. Our implementation uses Janus-Pro as the backbone model to ensure fair comparison with our proposed methods, as both approaches operate on the same foundation architecture.

We construct an image-level retrieval database using the same CC12M [5] and JourneyDB [34] datasets employed for our patch-level retrieval database to maintain consistency in the underlying data distribution. All images in the database are encoded into 512 dimensional vector representations using a pretrained CLIP [27] model. For each training instance in our 50,000 sample training set, we retrieve the most relevant reference image by encoding the corresponding text prompt with the same

Algorithm 1 Multi-Scale Feature Smoothing

Input: Image Representations $\mathbf{H}^l \in \mathbb{R}^{\sqrt{N} \times \sqrt{N} \times D}$, Retrieved Patch Representations $[\hat{\mathbf{h}}_1, \hat{\mathbf{h}}_2, \dots, \hat{\mathbf{h}}_K]$, Next Patch Index (i, j)

Output: Updated hidden states $[\hat{\mathbf{h}}_1, \hat{\mathbf{h}}_2, \dots, \hat{\mathbf{h}}_K]$

```

1 foreach  $\hat{\mathbf{h}}_i \in [\hat{\mathbf{h}}_1, \dots, \hat{\mathbf{h}}_K]$  do
2   for  $q = 2$  to  $Q$  do
3     Initialize tensor:  $\mathbf{M} \leftarrow \mathbf{0} \in \mathbb{R}^{Q \times Q \times D}$  Initialize
4     tensor:  $\hat{\mathbf{h}}_q \leftarrow \mathbf{0} \in \mathbb{R}^D$  for  $m = q$  down to 1 do
5       for  $n = q$  down to 1 do
6         // Extract local patch features
7          $\mathbf{H}_{\text{loc}}^l \leftarrow \mathbf{H}^l[i - m : i + q - m, j - n : j + q - n]$  // Compute smoothed
8         features
9          $\mathbf{M}_{mn} \leftarrow \text{Conv}_{q \times q}^1(\mathbf{H}_{\text{loc}}^l)$ 
10        // Pool smoothed features
11         $\hat{\mathbf{h}}_q += \text{Conv}_{q \times q}^2(\mathbf{M})$ 
12      // Update Retrieved features
13     $\hat{\mathbf{h}}_i \leftarrow \frac{\hat{\mathbf{h}}_q}{Q-1}$ 

```

CLIP model, extracting the [CLS] token as the text representation, and computing cosine similarity scores between the text representation and all image representations in the database. The image with the highest similarity score is selected as the retrieved reference. Each retrieved image is then processed through the quantized autoencoder from Janus-Pro to obtain image tokens $[v_1, \dots, v_N] = \mathcal{Z}(\theta_{Enc}(I))$, which are subsequently encoded into 2048 dimensional vector representations in the language model’s latent space using the image embedding and aligning layers in Janus-Pro. These retrieved image representations are concatenated with the text embeddings of the input prompts to form the augmented input for training the retrieval-enhanced model, which is the same training strategy used in RA-CM3.

During inference, given a text prompt for image generation, we follow the same retrieval process used in training. The input prompt is encoded using the CLIP text encoder, and we compute cosine similarity with all images in the database to identify the most relevant reference image. The retrieved image is processed through the same pipeline to obtain its representation in the language model’s latent space. This representation is then prepended to the text prompt embedding to provide the model with both textual and visual context for generation. The augmented input is fed into the fine-tuned Janus-Pro model to generate the output image following the standard autoregressive generation procedure.

B.2 Show-o Implementation Details

Our patch-based autoregressive retrieval augmentation methods can be theoretically adapted to any model that generates images through discrete tokens. To demonstrate this generalizability, we implement both DAiD and FAiD on the Show-o [42] model, which generates images through a masked token decoding process rather than strict left-to-right autoregression. Show-o decodes multiple image tokens simultaneously at each time step by converting masked tokens to specific image tokens based on a learned probability matrix. This fundamental difference in generation strategy necessitates several architectural adaptations to effectively incorporate our patch-based retrieval mechanisms while maintaining the model’s inherent generation capabilities.

DAiD on Show-o The implementation of DAiD on Show-o requires three key modifications to accommodate its non-autoregressive generation strategy. First, instead of constructing retrieval queries from upper-left neighboring patches as in autoregressive models, we utilize all eight surrounding patches to form the h -hop neighborhood representation for each target token position (i, j) . This comprehensive neighborhood encoding is computed as $[\mathbf{V}_{(i-1)(j-1)} : \mathbf{V}_{(i-1)(j)} : \mathbf{V}_{(i-1)(j+1)} : \mathbf{V}_{(i)(j-1)} : \mathbf{V}_{(i)(j+1)} : \mathbf{V}_{(i+1)(j-1)} : \mathbf{V}_{(i+1)(j)} : \mathbf{V}_{(i+1)(j+1)}]$, where missing positions are filled with zero vectors $\mathbf{0}$. Second, to mitigate retrieval noise arising from sparse neighborhood information in early time steps, we apply patch-level retrieval only during the final half of Show-o’s decoding process when sufficient contextual information is available. Third, since Show-o simultaneously predicts tokens for all patch positions at each time step rather than sequentially, we perform retrieval for all patch positions concurrently. At each qualifying time step t , for every patch position (i, j) in the partially generated image, we extract the eight-neighborhood representation as the retrieval query and obtain the top- K most similar patches $[\hat{\mathbf{v}}_1^{(i,j)}, \hat{\mathbf{v}}_2^{(i,j)}, \dots, \hat{\mathbf{v}}_K^{(i,j)}]$ from our database. We then construct position-specific retrieval distributions $D_{\text{retrieval}}^{(i,j)} \in \mathbb{R}^{|\mathcal{Z}|}$ using the same softmax formulation over retrieval distances as described in the main paper. These retrieval distributions are merged with Show-o’s predicted distributions for each patch position using the weighted average $D_{\text{merge}}^{(i,j)} = (1 - \lambda) \cdot D_{\text{model}}^{(i,j)} + \lambda \cdot D_{\text{retrieval}}^{(i,j)}$, where λ controls the retrieval influence across all positions.

FAiD on Show-o The adaptation of FAiD to Show-o involves both training and inference modifications to accommodate the model’s masked token generation process. During training, we prepare the training dataset by applying Show-o’s noise injection process to generate intermediate noisy representations at each time step, which serve as ground truth targets for the denoising process. For each training instance, we save these intermediate representations and apply patch-level retrieval to obtain relevant patches for all time steps. The training objective remains consistent with the standard Show-o formulation, but with augmented input representations that incorporate retrieved patch information. We insert FAiD modules into every L/b decoder layers of Show-o’s Φ model, where each module processes all patch positions simultaneously rather than focusing on a single next token. At each qualifying time step and for each FAiD-equipped layer l , we construct the 2D spatial

representation $\mathbf{H}^l \in \mathbb{R}^{\sqrt{N} \times \sqrt{N} \times D}$ from the current hidden states and perform multi-scale feature smoothing for all patch positions. For each position (i, j) and its corresponding retrieved patches $[\hat{\mathbf{h}}_1^{(i,j)}, \hat{\mathbf{h}}_2^{(i,j)}, \dots, \hat{\mathbf{h}}_K^{(i,j)}]$, we apply the convolution operations $\{\text{Conv}_{2 \times 2}, \text{Conv}_{3 \times 3}, \dots, \text{Conv}_{Q \times Q}\}$ to capture contextual patterns at multiple scales. The refined representations are computed as $\hat{\mathbf{h}}_k^{(i,j)} \leftarrow \sum_{q=2}^Q \text{softmax}(\Omega)_q \cdot \hat{\mathbf{h}}_{k,q}^{(i,j)}$, where $\hat{\mathbf{h}}_{k,q}^{(i,j)}$ represents the output of the $q \times q$ convolution for patch k at position (i, j) . The final augmented representation for each position is calculated as $h_{ij}^{(l+1)} = h_{ij}^l + \Delta h_{ij}^l + \sum_{k=1}^K s_k^{(i,j)} \hat{\mathbf{h}}_k^{(i,j)}$, where Δh_{ij}^l represents the standard transformer layer updates including self-attention and feed-forward components, and $s_k^{(i,j)}$ are position-specific compatibility scores computed through learned linear projections. During inference, we follow the same procedure but apply retrieval and feature blending only during the final half of the generation time steps to ensure sufficient contextual information is available for effective patch integration.

B.3 Training Setup

Training Datasets For model training, we utilize two large-scale image-caption datasets: CC12M [5] and Midjourney-v6 [3]. From the training sets of these datasets, we randomly sample a total of 50,000 image-caption pairs (25,000 from each dataset) to fine-tune our model. Each image is encoded into 576 patch features and corresponding image tokens with the same image tokenizer [35] employed in the Janus-Pro model. For each image patch, we further retrieve the top- K image tokens from our retrieval database that exhibit similar neighborhood relationships. Consequently, each training instance comprises: (1) a textual image caption that serves as the conditioning input, (2) a sequence of 576 image tokens representing the ground-truth image, where each token is paired with K relevant image tokens retrieved from the database based on similar contextual features.

Training Details For the implementation of our FAiD approach, we fine-tune two pre-trained text-to-image generation models using the training dataset of 50K text-image pairs that we constructed. We select Janus-Pro-1B [8] and Show-o [42] as our base models. The fine-tuning process is conducted on 4 NVIDIA A100 (80GB) GPUs with a global batch size of 256 for a single epoch. We utilize the AdamW optimizer without weight decay, incorporating a 10% linear warm-up schedule followed by a constant learning rate of $2e-4$.

B.4 Evaluation Benchmarks and Metrics

To comprehensively evaluate our proposed methods, we adopt multiple widely used benchmarks that assess different aspects of image generation quality:

- **GenEval** [13] is a benchmark designed to evaluate models' ability to understand and generate images based on specific attributes and relationships described in text prompts. It comprises multiple categories such as single object generation, two-object composition, counting, colors, positioning, color attribution and so on. Performance is measured as the percentage of generated images that correctly align with the text descriptions.
- **DPG-Bench** [17] (Detailed Prompt Generation Benchmark) evaluates how well image generation models handle detailed prompts with complex requirements, covering categories such as global image quality, entity generation, attribute accuracy, relationship modeling, and other complex generation tasks. Scores are reported as percentages.
- For the **Midjourney-30k benchmark** [38], we employ three complementary metrics to evaluate the quality of generated images, including (1) Fréchet Inception Distance (FID) [16], which measures the statistical similarity between the distributions of generated and real images in the feature space of a pre-trained Inception network; (2) CLIP-MMD (CMMD) [19], which measures the distance between real and generated images using CLIP embeddings and the Maximum Mean Discrepancy, and is specifically designed to better align with human perception of image quality and addresses several limitations of FID, including poor sample efficiency and incorrect normality assumptions; and (3) Fréchet Wavelet Distance (FWD) [37], which measures the distance between real and generated images in the wavelet packet coefficient space. FWD captures both spatial and frequency information without relying on pre-trained networks, making it domain-agnostic and robust to

³<https://huggingface.co/datasets/brivangl/midjourney-v6-llava>

domain shifts across various image types. For all three metrics, lower scores indicate higher-quality image generation, with both CMMD and FWD particularly effective in capturing distortions in generated images in ways that better correlate with human judgements.

C Experiment Results and Discussion

C.1 Accuracy of Patch-based Autoregressive Retrieval

To assess the effectiveness of our patch-level autoregressive retrieval mechanism, we conduct a comparative analysis between the top- K retrieved image tokens and the ground-truth tokens to be generated. Specifically, we randomly sampled 1,000 instances from our training set, each comprising 576 image tokens and $576 \times k$ retrieved tokens. To demonstrate the accuracy of the retrieved image tokens, for each ground-truth image token, we also randomly sample a vocabulary code as non-relevant tokens. Using the shared codebook, we transform all image tokens into vector representations and compute the l_2 distances between each ground-truth image token and its top- K retrieved counterparts. Similarly, we also compute the mean of the l_2 distance between each ground-truth token and the randomly sampled tokens. As shown in Figure 6, the l_2 distance between retrieved tokens and ground-truth image tokens is significantly smaller than the distance between randomly sampled tokens and ground-truth tokens. As k increases, the distance between the k -th retrieved token and the ground-truth token also increases, demonstrating the effectiveness of the retrieval approach and our assumption that image patches with similar neighbors usually exhibit inherent similarities.

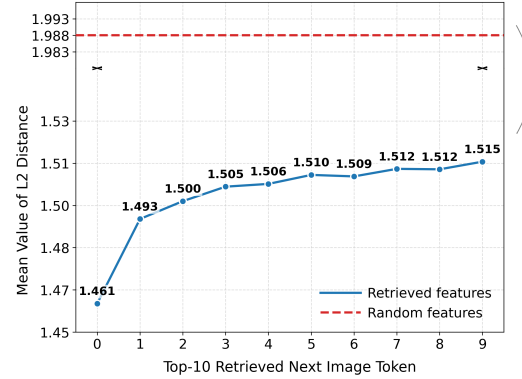


Figure 6: l_2 distance between ground-truth tokens and top-10 retrieved tokens (blue line) compared to randomly sampled tokens (red dashed line). The curved arrow indicates a broken y-axis that accommodates the large gap between the retrieved token and the random token baseline.

C.2 Hyperparameter Optimization

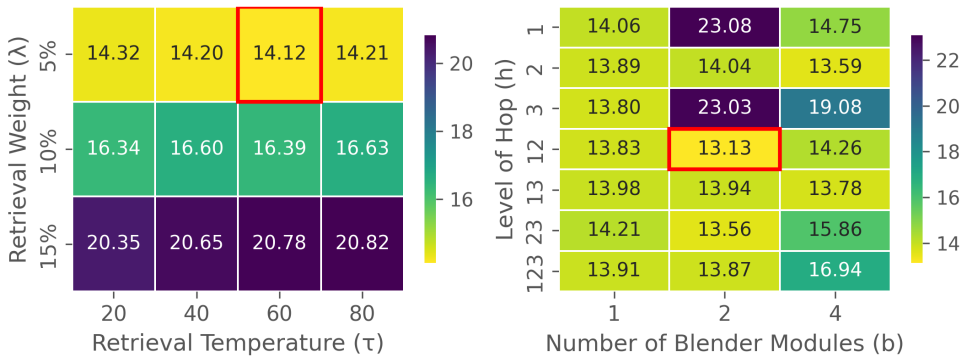


Figure 7: Hyperparameter optimization results for DAiD and FAiD on FID scores. Left: FID scores for DAiD across different combinations of retrieval temperature τ and merging weight λ . Right: FID scores for FAiD across varying levels of hop h and numbers of blender modules b . All experiments conducted on the Midjourney-10K benchmark, with optimal configurations highlighted by red borders.

Both DAiD and FAiD require careful optimization of distinct sets of hyperparameters. For DAiD, we optimize the retrieval temperature τ and merging weight λ , which control the retrieval-based

1026 probability distribution sharpness and the balance between retrieval and model predictions, respec-
1027 tively. For FAiD, we focus on the level of hop (h) and number of blender modules (b), determining
1028 the spatial context incorporated during retrieval and extent of feature blending. To identify optimal
1029 configurations, we conducted a systematic ablation study on the Midjourney-10K benchmark using
1030 Fréchet Inception Distance (FID) as the performance metric.

1031 Figure 7 presents the FID scores for DAiD across different combinations of λ and τ , and for FAiD
1032 across varying levels of (h) and (b), where composite hop levels such as “12” indicate combined
1033 use of multiple hop distances. Analysis of the DAiD results reveals that performance degrades
1034 as λ increases, suggesting that modest integration of retrieval information enhances performance
1035 while excessive reliance impairs generative flexibility. The retrieval temperature τ demonstrates
1036 less pronounced effects, though a moderate value of 0.6 provides marginal benefits. For FAiD,
1037 configurations incorporating multiple hop levels generally outperform single hop levels, with the “12”
1038 configuration yielding optimal results. Regarding blender modules, an intermediate value consistently
1039 delivers the best performance, implying that moderate feature blending optimizes the incorporation of
1040 retrieved patches while avoiding both under-utilization and over-smoothing. Based on this analysis,
1041 we selected $\lambda = 0.05$ and $\tau = 0.6$ for DAiD, and hop levels “12” with 2 blender modules for FAiD,
1042 achieving FID scores of 14.12 and 13.13, respectively. These configurations effectively harness
1043 retrieval information while preserving the generative strengths of the underlying Janus-Pro model, as
1044 demonstrated by their superior performance on the benchmark.

1045 D Limitations

1046 While our AR-RAG framework demonstrates strong performance across multiple benchmarks,
1047 several limitations should be acknowledged. First, our approach relies on discrete image tokenization
1048 and targets discrete token-based models, so it may not be directly applied to continuous diffusion
1049 models operating in latent spaces. Second, due to computational resource limitations, our retrieval
1050 database remains smaller than billion-scale databases. This limitation may introduce visual pattern
1051 biases, as the database may not fully capture the diversity of real-world visual patterns, potentially
1052 affecting the generation of underrepresented visual elements. Third, our implementation focuses
1053 exclusively on 2D image generation. While the underlying patch-based retrieval concept could
1054 theoretically extend to other structured generation tasks such as 3D point cloud generation, we have
1055 not explored these applications.

1056 E Broader impacts

1057 We propose a novel retrieval-augmented approach to enhance existing image generation models.
1058 Our method is both highly efficient and readily adaptable to a wide range of applications, making it
1059 valuable for both academic research and industrial deployment. However, as our approach builds upon
1060 existing generative models, it may inherit their biases and could potentially produce inappropriate
1061 outputs in the absence of additional safety mechanisms. Furthermore, the large-scale retrieval
1062 database may contain unsafe or undesirable content, which can be reflected in the retrieved image
1063 patches. To ensure safe deployment in real-world scenarios, additional safeguards and filtering
1064 measures are necessary to mitigate these risks.