ENSEMBLING DIFFUSION MODELS VIA ADAPTIVE FEATURE AGGREGATION

Anonymous authors
Paper under double-blind review

ABSTRACT

The success of the text-guided diffusion model has inspired the development and release of numerous powerful diffusion models within the open-source community. These models are typically fine-tuned on various expert datasets, showcasing diverse denoising capabilities. Leveraging multiple high-quality models to produce stronger generation ability is valuable, but has not been extensively studied. Existing methods primarily adopt parameter merging strategies to produce a new static model. However, they overlook the fact that the divergent denoising capabilities of the models may dynamically change across different states, such as when experiencing different prompts, initial noises, denoising steps, and spatial locations. In this paper, we propose a novel ensembling method, Adaptive Feature Aggregation (AFA), which dynamically adjusts the contributions of multiple models at the feature level according to various states (i.e., prompts, initial noises, denoising steps, and spatial locations), thereby keeping the advantages of multiple diffusion models, while suppressing their disadvantages. Specifically, we design a lightweight Spatial-Aware Block-Wise (SABW) feature aggregator that adaptive aggregates the block-wise intermediate features from multiple U-Net denoisers into a unified one. The core idea lies in dynamically producing an individual attention map for each model's features by comprehensively considering various states. It is worth noting that only SABW is trainable with about 50 million parameters, while other models are frozen. Both the quantitative and qualitative experiments demonstrate the effectiveness of our proposed method.

1 INTRODUCTION

Diffusion models (Sohl-Dickstein et al., 2015; Ho et al., 2020) have progressively become the mainstream models for text-guided image generation (Nichol et al., 2021; Ramesh et al., 2022; Saharia et al., 2022; Rombach et al., 2022; Balaji et al., 2022; Xue et al., 2023; Feng et al., 2023), which treats generation as an iterative denoising task. Recently, the open-source stable diffusion (SD) (Rombach et al., 2022) model has prompted the development and release of numerous powerful diffusion models within the open-source community (e.g., CivitAI[1]). These models are typically fine-tuned on various expert datasets, showcasing diverse denoising capabilities. Leveraging multiple high-quality models to dig out better generations is an important research direction, which has not been extensively studied.

Existing methods leverage multiple diffusion models through the weighted merging of model parameters, which can be called the static method. The weights are usually manually set (e.g., Weighted-Merging and MBW) or automatically searched through enumeration (e.g., autoMBW). In contrast, model ensembling, which can be called the dynamic method, often uses dynamic strategies to fuse multiple models at the feature level. Unlike ensembling for classification models (Freund & Schapire, 1995) that usually work after the output, ensembling for diffusion models typically needs to work for each block. However, the denoising capabilities of models vary not only at different blocks but also at different spatial locations. As illustrated in Figure 1, we conduct denoising experiments on different prompts with various initial noises. Then, we plot the proportion of *wins* (i.e., the model with the least error between the predicted noise and the initial noise), for each model in a certain spatial region, to distinguish the denoising capabilities of the models at different states. In

[1] https://civitai.com/

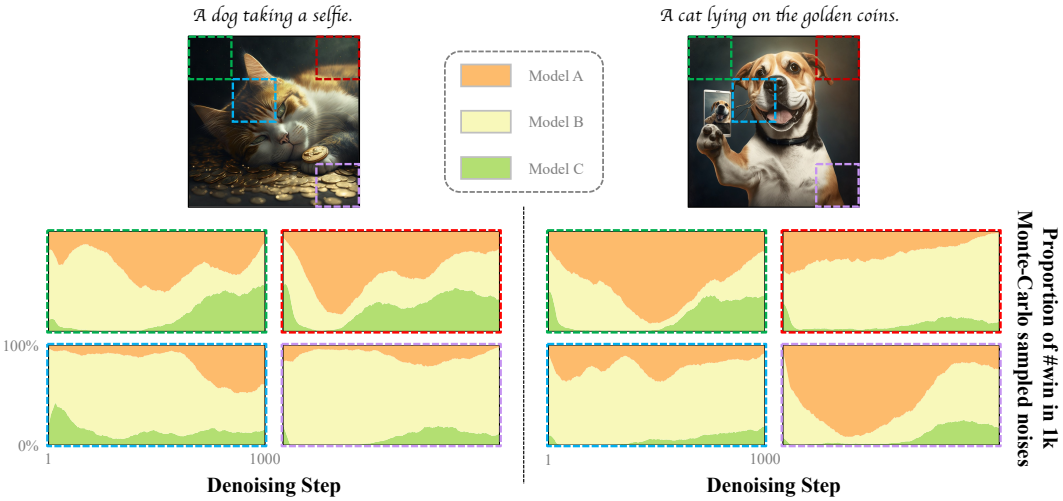


Figure 1: Examples to illustrate the dynamical change of the denoising capabilities across various states. We conduct experiments on different prompts with various initial noises. We then plot the proportion of *wins* (i.e., the model with the least error between the predicted noise and the initial noise), for each model in a certain spatial region.

other words, different prompts, initial noises, denoising steps, and spatial locations can all have a significant impact on the denoising capabilities of diffusion models. This implies that an adaptive method is needed to ensure that each diffusion model dominates the generation at its strongest states.

In this paper, we propose a novel Adaptive Feature Aggregation (AFA) method, which dynamically adjusts the contributions of multiple models at the feature level by taking into account various states, such as prompts, initial noises, denoising steps, and spatial locations. Specifically, we design a lightweight Spatial-Aware Block-Wise (SABW) feature aggregator that adaptively aggregates the block-level intermediate features from multiple U-Net denoisers into a unified one. The core idea of adaptive aggregation lies in dynamically producing an individual attention map for each model's features by comprehensively taking into account the various states. A noteworthy aspect of AFA is that only SABW is trainable with about 50 million parameters, while all other models are frozen.

Our main contributions are summarized as follows:

- We propose an ensembling-based AFA method to dynamically adjust the contributions of multiple models at the feature level.
- We design the SABW feature aggregator that can produce attention maps according to various states to adaptively aggregate the block-level intermediate features from multiple U-Net denoisers.
- We conduct both quantitative and qualitative experiments, demonstrating that our AFA outperforms the base models and the baseline methods in both superior quality and context alignment.

2 Related Work

Text-Guided Image Synthesis. Early works for text-guided image synthesis leverage Generative Adversarial Networks (GAN) (Goodfellow et al., 2014) conditioned on text (Tao et al., 2022; Xu et al., 2018; Zhang et al., 2021a; Zhu et al., 2019). Based on the success of Transformers (Vaswani et al., 2017), many subsequent works reframe text-guided image synthesis as a sequence-to-sequence task Zhang et al. (2021b); Ramesh et al. (2021); Ding et al. (2021); Gafni et al. (2022); Yu et al. (2022). Recently, diffusion models (Sohl-Dickstein et al., 2015; Ho et al., 2020) gradually become mainstream, which treats image generation as an iterative denoising task. By injecting text as a condition into the denoising process, many diffusion-based models achieve significant success in text-guided image synthesis (Nichol et al., 2021; Ramesh et al., 2022; Saharia et al., 2022; Rombach et al., 2022; Balaji et al., 2022; Xue et al., 2023; Feng et al., 2023). Among them, the Latent Diffusion Model (LDM) (Rombach et al., 2022) performs the diffusion and reverse process in the

latent space, instead of the pixel space, which largely reduces the computational burden. Among all the implementations of the Latent Diffusion Model, the stable diffusion model (SD) is the most famous one. Along with its public availability, the open-source community achieves tremendous success, with the emergence of many excellent fine-tuned models. These high-quality models are typically fine-tuned on various expert datasets, showcasing diverse denoising capabilities. In this paper, we aim to leverage multiple such models to achieve stronger text-to-image generation.

Model Merging. For vision or language understanding tasks, the effectiveness of parameter merging (Frankle et al., 2020; Wortsman et al., 2022b; Matena & Raffel, 2022; Ilharco et al., 2022b; Li et al., 2022; Don-Yehiya et al., 2022; Jin et al., 2022) can be interpreted from the perspectives of loss landscape (Wortsman et al., 2022a; Ainsworth et al., 2022; Stoica et al., 2023) and task arithmetic (Ilharco et al., 2022a; Ortiz-Jimenez et al., 2023; Zhang et al., 2023). However, merging-based methods have not been widely studied for generation tasks, especially diffusion-based generation. Recently, many intuitive merging-based methods have emerged. One of the popular methods is Weighted-Merging, which manually determines weights to merge each U-Net (Ronneberger et al., 2015) parameter of multiple SD models. Although the simplicity, Weighted Merging coarsely allocates weight to all U-Net blocks. As an improvement, Merge Block Weighted (MBW) allows for manually setting different merging weights for the parameters of distinct U-Net blocks, which provides a more fine-grained merging strategy. Furthermore, to reduce the reliance on manual weights, autoMBW attempts to automate the merging process, which enumeratively selects the optimal combination of MBW weights by an aesthetic classifier. However, autoMBW is constrained by the performance bottleneck of the aesthetic classifier, and the enumerative selection leads to a huge time consumption to find the optimal settings. Such merging methods can be called the static methods. In this paper, we aim to leverage multiple diffusion models by the dynamic methods, i.e., ensembling.

Model Ensembling. Model ensembling is an effective method to achieve better performance (Zhou, 2012), which has been widely applied in various vision understanding tasks, such as classification (Zhao et al., 2005; Rokach, 2010; Yang et al., 2010), regression (Mendes-Moreira et al., 2012; Ren et al., 2015), clustering (Vega-Pons & Ruiz-Shulcloper, 2011). While fewer works focus on the ensembling of generative models, because of the complexity of image space. Vision-Aided GAN (Kumari et al., 2022) focuses on GANs (Goodfellow et al., 2014), which guides the optimization of a target generator by ensembling pretrained models as a loss. MagicFusion (Zhao et al., 2023) focuses on the diffusion models, which fuses the predicted noises of two expert U-Net denoisers to implement specific applications, such as style transferring and object binding. In this paper, we aim to efficiently ensemble multiple diffusion models to achieve general generation improvements[2].

3 METHOD

3.1 PRELIMINARY

As a type of generative model, the diffusion model (Sohl-Dickstein et al., 2015; Ho et al., 2020) consists of two processes, which are the diffusion process and the reverse process, respectively. In the diffusion process (i.e., the forward process), the Gaussian noises are iteratively added to degrade the images over T steps until the images become completely random noise. In the reverse process, a trained denoiser is used to iteratively generate images from the sampled Gaussian noise.

When training, given an input image $\mathbf{x}_0$ and the additional condition $\mathbf{c}$ (e.g., encoded textual prompts etc.), the denoising loss is defined as

$$\mathcal{L}_{\text{denoise}} = \mathbb{E}_{\mathbf{x}_0,\boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0},\mathbf{I}),\mathbf{c},t} \left\|\boldsymbol{\epsilon}_\theta\left(\mathbf{x}_t,\mathbf{c},t\right)-\boldsymbol{\epsilon}\right\|^2 . \tag{1}$$

Among them, $\mathbf{x}_t = \sqrt{\alpha_t}\mathbf{x}_0 + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}$ is the noisy image at timestep $t \in [1, T]$, where α_t is a predefined scalar from the noise scheduler. $\boldsymbol{\epsilon}$ is the added noise. $\boldsymbol{\epsilon}_\theta$ is the denoiser with the learnable parameters θ, which predicts the noise to be removed from the noisy image.

LDM stands out as one of the most popular diffusion models. It performs the diffusion and reverse process in the latent space which is encoded by a Variational Auto-Encoder (VAE) (Kingma & Welling, 2013; Rezende et al., 2014). In LDM, the U-Net structure (Ronneberger et al., 2015) is used

[2]Note that combining the styles/concepts of multiple diffusion models is also a common goal of ensembling/merging, which is not the interest of our work.

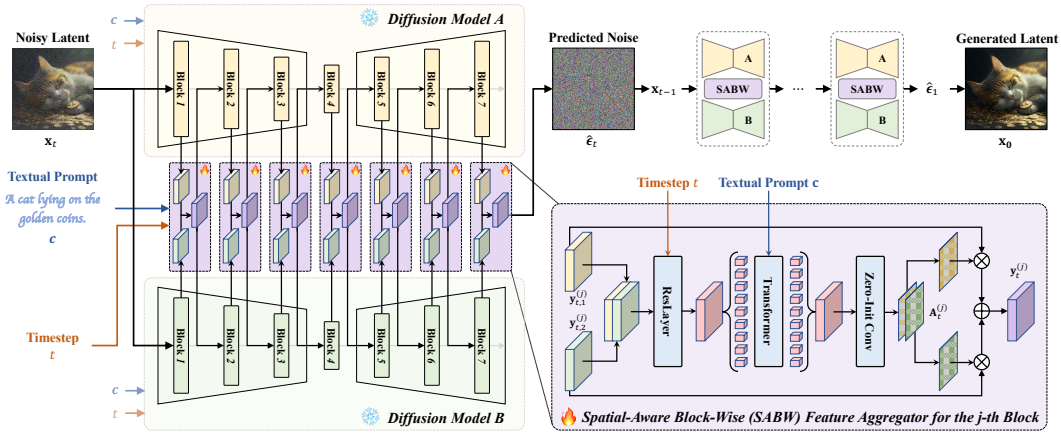


Figure 2: Framework of ensembling multiple diffusion models by our AFA method.

for denoiser, which contains three parts of blocks, which are the down-sampling blocks, the middle block, and the up-sampling blocks, respectively. Each down-sampling block is skip-connected with a corresponding up-sampling block. These blocks are composed of Transformer (Vaswani et al., 2017) layers and ResNet (He et al., 2016) layers. In the Transformer layers, the cross-attention mechanism is employed to incorporate conditional textual prompts, which plays a crucial role in text-guided image generation.

3.2 Adaptive Feature Aggregation

Our proposed Adaptive Feature Aggregation (AFA) ensembles multiple diffusion models that share the same architecture but different parameters. Specifically, SABW is designed to integrate intermediate features from multiple U-Net denoisers at the block level. Figure 2 demonstrates an example of ensembling two diffusion models, and each U-Net denoiser contains seven blocks by SABW. The forward process of AFA can be seen in Alg. 1.

Given N diffusion models to be ensembled, whose U-Net denoisers contain K blocks, let $\boldsymbol{\epsilon}_{\theta_i}$ be the i-th U-Net denoiser, and $\boldsymbol{\epsilon}_{\theta_i}^{(j)}$ be its j-th block. For the timestep t, it transforms the input features $\mathbf{x}_t^{(j)}$ into the output feature by

$$\mathbf{y}_{t,i}^{(j)} = \boldsymbol{\epsilon}_{\theta_i}^{(j)}(\mathbf{x}_t^{(j)}, \mathbf{c}, t) \ . \tag{2}$$

Note that because of the same architecture of all $\boldsymbol{\epsilon}_{\theta_i}$, $\mathbf{y}_{t,i}^{(j)}$ has the same shape, i.e., $\mathbf{y}_{t,i}^{(j)} \in \mathbb{R}^{h_j \times w_j \times c_j}$, where h_j, w_j, and c_j are the height, width, and channels, respectively. To achieve the output feature for the next block, a block $f_\varphi^{(j)}$ of SABW is employed to aggregate $\{\mathbf{y}_{t,i}^{(j)}\}_{i=1}^N$,

$$\mathbf{y}_t^{(j)} = f_\varphi^{(j)}(\{\mathbf{y}_{t,i}^{(j)}\}_{i=1}^N, \mathbf{c}, t) \in \mathbb{R}^{h_j \times w_j \times c_j} \ , \tag{3}$$

where φ is the parameters of the SABW block. As a whole, the j-th ensembled block of these U-Net denoisers can be formulated as

$$\mathbf{y}_t^{(j)} = \mathcal{F}_\Theta^{(j)}(\mathbf{x}_t^{(j)}, \mathbf{c}, t) \ , \tag{4}$$

where Θ is the parameter collection of φ and all the θ_i. It can be deduced that an ensembled block operates with the same functionality as a single block.

3.3 Spatial-Aware Block-Wise Feature Aggregator

The Spatial-Aware Block-Wise (SABW) feature aggregator is learned to aggregate output features of each block from multiple U-Net denoisers. We design SABW to learn spatial attention to reweight the contributions of these U-Net denoisers at each spatial position. Specifically, the spatial attention map $\mathbf{A}_t^{(j)}$ for N U-Net denoisers is achieved by

$$\mathbf{F}_t^{(j)} = g_\varphi^{(j)}(\{\mathbf{y}_{t,i}^{(j)}\}_{i=1}^N, \mathbf{c}, t) \in \mathbb{R}^{h_j \times w_j \times N} \ , \tag{5}$$

Algorithm 1 One forward process of Adaptive Feature Aggregation (AFA).

Input: Timestep t, Condition $\mathbf{c}$, Input latent $\mathbf{x}_t$, U-Net denoisers $\{\boldsymbol{\epsilon}_{\theta_i}\}_{i=1}^{N}$, each of which contains K blocks, and SABW $f_{\varphi}^{(j)}$ for j-th U-Net block.
Output: Predicted noise $\hat{\boldsymbol{\epsilon}}_t$.
Initialize an empty stack $\mathcal{S}$ to store the features from the skip connections;
$\mathbf{x}_t^{(1)} \leftarrow \mathbf{x}_t$;
for block index j from 1 to K **do**
  ▷ *Ensembling the j-th U-Net block by SABW feature aggregator $f_{\varphi}^{(j)}$*
  for U-Net denoiser index i from 1 to N **do**
    Get the output feature $\mathbf{y}_{t,i}^{(j)}$ base on Eq. 2;
  end for
  Get $\mathbf{y}_t^{(j)}$ by aggregating all $\mathbf{y}_{t,i}^{(j)}$ base on Eq. 3 for the next block;
  if $(j+1)$-th block is down-sampling block **then**
    $\mathbf{x}_t^{(j+1)} \leftarrow \mathbf{y}_t^{(j)}$ and push $\mathbf{y}_t^{(j)}$ into $\mathcal{S}$;
  else if $(j+1)$-th block is up-sampling block **then**
    Pop $\mathbf{y}_t^{s}$ from $\mathcal{S}$ and $\mathbf{x}_t^{(j+1)} \leftarrow \text{concat}(\mathbf{y}_t^{(j)}, \mathbf{y}_t^{s})$;
  else
    $\mathbf{x}_t^{(j+1)} \leftarrow \mathbf{y}_t^{(j)}$;
  end if
end for
$\hat{\boldsymbol{\epsilon}}_t \leftarrow \mathbf{x}_t^{(K)}$;
return predicted noise $\hat{\boldsymbol{\epsilon}}_t$.

$$\mathbf{A}_t^{(j)} = \text{softmax}(\mathbf{F}_t^{(j)}) \in [0,1]^{h_j \times w_j \times N} , \tag{6}$$

where, $\mathbf{F}_t^{(j)}$ is the output of the learnable network of SABW g_{φ}. The ensembled output feature is

$$\mathbf{y}_t^{(j)} = \mathcal{F}_{\Theta}^{(j)}(\mathbf{x}_t^{(j)}, \mathbf{c}, t) = \sum_{i=1}^{N} \mathbf{A}_{t,i}^{(j)} \otimes \mathbf{y}_{t,i}^{(j)} , \tag{7}$$

where $\otimes$ is the element-wise multiplication. $\mathbf{A}_{t,i}^{(j)} \in \mathbb{R}^{h_j \times w_j \times 1}$ is the spatial attention map for the i-th U-Net denoiser.

Note that the spatial-attention mechanism of SABW differs to the typical multi-head self-attention mechanism (Vaswani et al., 2017), which simply uses three linear layers to project the input features into three separate spaces(i.e., query, key, and value), and then computes the attention map across all the projected features. Our spatial-attention mechanism is based on an important experimental insight as shown in Figure 1. The divergent denoising capabilities of multiple diffusion models are influenced by the prompts and the denoising steps, and they dynamically vary across different spatial locations. Thus, the projection method needs to account for the input prompt and the current denoising step. And it is sufficient to compute the attention map on each spatial location individually. As a result, better generation performance can be achieved by enabling multiple models to collaborate at different spatial locations under the conditions of the input prompt and the current denoising step.

Specifically, based on the denoising loss (i.e., Eq. 1), the denoising capability[3] of $\boldsymbol{\epsilon}_{\theta}$ for $(\mathbf{x}_0, \mathbf{c})$ at timestep t can be defined as $-\mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0},\mathbf{I})}\|\hat{\boldsymbol{\epsilon}}_t - \boldsymbol{\epsilon}\|^2$, where $\hat{\boldsymbol{\epsilon}}_t = \boldsymbol{\epsilon}_{\theta}(\mathbf{x}_t, \mathbf{c}, t)$. For the spatial location (x, y), the denoising capability is formulated as $-\mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0},\mathbf{I})}\|\hat{\boldsymbol{\epsilon}}_{(x,y)} - \boldsymbol{\epsilon}_{(x,y)}\|^2$, which considers the noise prediction errors at the spatial location (x, y).

For example, as shown in Figure 1, we employ three SD models to present their positional denoising capability on the specific image and textual prompt by Monte-Carlo sampled noises. We demonstrate the results of four patches which indicate that the divergent denoising capabilities of the three models vary at different spatial locations. SABW is learned to incorporate these denoisers to achieve stronger denoising capability at all spatial locations at the block level.

[3]Compared with the denoising loss in Eq. 1, the denoising capability is defined by adding the negative sign, because of a negative correlation between the denoising capability and the denoising loss.

	COCO 2017				Draw Bench Prompts			
	FID ↓	IS	CLIP-I	CLIP-T	AES	PS	HPSv2	IR
Base Model A	13.01	5.65	.6724	.2609	5.4102	21.6279	27.8007	.3544
Base Model B	13.45	5.43	.6775	.2652	5.5013	21.4624	27.7246	.2835
Base Model C	12.32	6.32	.6890	.2566	5.4881	21.8031	27.9652	.3922
Wtd. Merging	10.65	6.93	.6861	.2626	5.4815	21.7272	27.9086	.3909
MBW	11.03	6.51	.6870	.2624	5.4812	21.7201	27.9080	.3922
autoMBW	13.35	5.51	.6772	.2577	5.5056	21.4785	27.8192	.3672
MagicFusion	10.53	6.85	.6751	.2620	5.3431	21.3840	27.8105	.3317
AFA (Ours)	**9.76**	**7.14**	**.6926**	**.2675**	**5.5201**	**21.8263**	**27.9734**	**.4388**

Table 1: Quantitative comparison between AFA with the three base models in Group I (i.e., ER, MMR, and RV) and the baseline methods.

	COCO 2017				Draw Bench Prompts			
	FID ↓	IS	CLIP-I	CLIP-T	AES	PS	HPSv2	IR
Base Model A	12.12	5.66	.6849	.2623	5.5641	21.8013	28.0183	.4238
Base Model B	12.41	5.59	.6818	.2580	5.5027	21.7249	28.0343	.4202
Base Model C	12.05	5.95	.6638	.2642	5.5712	21.4936	27.8089	.3367
Wtd. Merging	11.53	6.56	.6824	.2631	5.5756	21.7516	28.0014	.4387
MBW	12.06	6.42	.6826	.2632	5.5772	21.7487	28.0029	.4396
autoMBW	12.39	5.62	.6774	.2588	5.5478	21.5135	27.9873	.3513
MagicFusion	11.63	7.13	.6790	.2640	5.4674	21.4270	27.9608	.4194
AFA (Ours)	**10.27**	**7.42**	**.6855**	**.2717**	**5.5798**	**21.8059**	**28.0371**	**.4892**

Table 2: Quantitative comparison between AFA with the three base models in Group II (i.e., AR, CR, and RCR) and the baseline methods.

To implement g_φ in Eq. 5, SABW applies a ResNet (He et al., 2016) layer to introduce timesteps by adding the time embedding into the feature map, and a Transformer (Vaswani et al., 2017) layer to introduce textual prompts by the cross-attention,

$$\mathbf{Y}_t^{(j)} \leftarrow \text{concat}(\{\mathbf{y}_{t,i}^{(j)}\}_{i=1}^N) \in \mathbb{R}^{h_j \times w_j \times (N \cdot c_j)} , \tag{8}$$

$$\mathbf{H}_t^{(j)} \leftarrow \text{ResLayer}(\mathbf{Y}_t^{(j)} + \gamma(t)) \in \mathbb{R}^{h_j \times w_j \times d} , \tag{9}$$

$$\mathbf{O}_t^{(j)} \leftarrow \text{TransformerLayer}(\mathbf{H}_t^{(j)}, \mathbf{c}) \in \mathbb{R}^{h_j \times w_j \times d} , \tag{10}$$

$$\mathbf{F}_t^{(j)} \leftarrow \text{ZeroInitConv}(\mathbf{O}_t^{(j)}) \in \mathbb{R}^{h_j \times w_j \times N} , \tag{11}$$

where γ is used to map the timestep into the time embedding with the same shape of $\mathbf{Y}_t^{(j)}$. d is the hidden state dimension. ZeroInitConv$(\cdot)$ is the convolution layer initialized by zero, which leads to equal attention weights for all denoisers before training. We aspire for the model to commence training from a completely equitable state.

3.4 Training

Given the ensembled denoiser $\mathcal{F}_\Theta$, the training object is same as the denoising loss in Eq. 1,

$$\mathcal{L} = \mathbb{E}_{\mathbf{x}_0, \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \mathbf{c}, t} \left\| \mathcal{F}_\Theta \left(\mathbf{x}_t, \mathbf{c}, t \right) - \boldsymbol{\epsilon} \right\|^2 . \tag{12}$$

Note that the parameters Θ contains both the parameters of SABW φ and the parameters of all U-Net denoisers $\{\theta_i\}_{i=1}^N$, where φ are learnable, and $\{\theta_i\}_{i=1}^N$ are frozen.

A well-trained AFA ensembles multiple U-Net denoisers to enhance the denoising capability in every block when experiencing different prompts, initial noises, denoising steps, and spatial locations. It leads to smaller denoising errors and stronger generation performance.

3.5 Inference

The inference of AFA starts from the sampled Gaussian noise. Then, the diffusion scheduler (e.g., DDIM (Song et al., 2020), PNDM (Liu et al., 2022), DPM-Solver (Lu et al., 2022a;b), etc.) is applied to generate images with multiple denoising steps.

For each inference step, the noise prediction relies on the technique of Classifier-Free Guidance (CFG) (Ho & Salimans, 2022), which is formulated as $\hat{\boldsymbol{\epsilon}}_t^{\text{pred}} = \hat{\boldsymbol{\epsilon}}_t^{\text{uc}} + \beta_{\text{CFG}}(\hat{\boldsymbol{\epsilon}}_t^{\text{c}} - \hat{\boldsymbol{\epsilon}}_t^{\text{uc}})$. Among them, $\hat{\boldsymbol{\epsilon}}_t^{\text{pred}}$, $\hat{\boldsymbol{\epsilon}}_t^{\text{uc}}$, $\hat{\boldsymbol{\epsilon}}_t^{\text{c}}$, and β_{CFG} are the predicted noise, the predicted noise with condition, the predicted noise without condition, and a guidance scale, respectively. The latent for the next step is $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\mathbf{x}_t - \frac{\sqrt{1-\alpha_t}}{\sqrt{\alpha_t}}\hat{\boldsymbol{\epsilon}}_t^{\text{pred}}$. Finally, the generated image is achieved from the latent by a VAE decoder.

4 Experiments

4.1 Experiment Settings

Base Models. We select six popular models with the same architecture from SDv1.5 in CivitAI. There are 12 down-sampling blocks, 1 middle block, and 12 up-sampling blocks for each model.

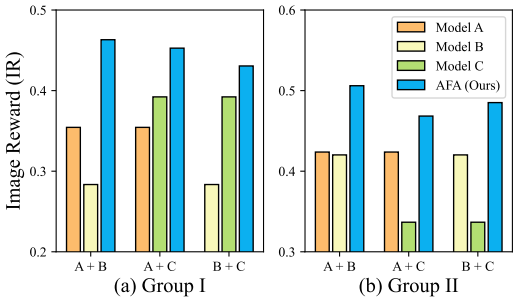


Figure 3: Quantitative comparison between AFA with the two base models.

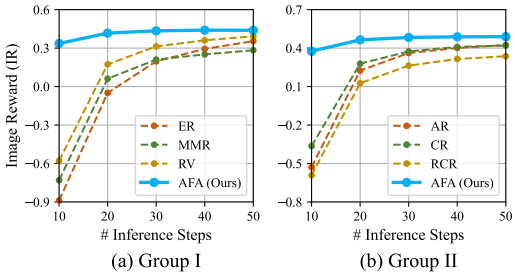


Figure 4: Effect of varying inference steps.

These models are randomly divided into two groups, with three models in each group. The models from the same group will be ensembled.

- **Group I** contains EpicRealism (ER), MajicMixRealistic (MMR), and RealisticVision (RV);
- **Group II** contains AbsoluteReality (AR), CyberRealistic (CR), and RealCartoonReslistic (RCR).

Training. AdamW (Loshchilov & Hutter, 2017) is used as the optimizer with a learning rate of 0.0001 and a weight decay of 0.01. Note that only the parameters of SABW are optimized, while those of U-Net denoisers are frozen. Our AFA framework is trained on 10,000 samples from the dataset JourneyDB (Pan et al., 2023) for 10 epochs with batch size 8. To enable CFG, we use a probability of 0.1 to drop textual prompts.

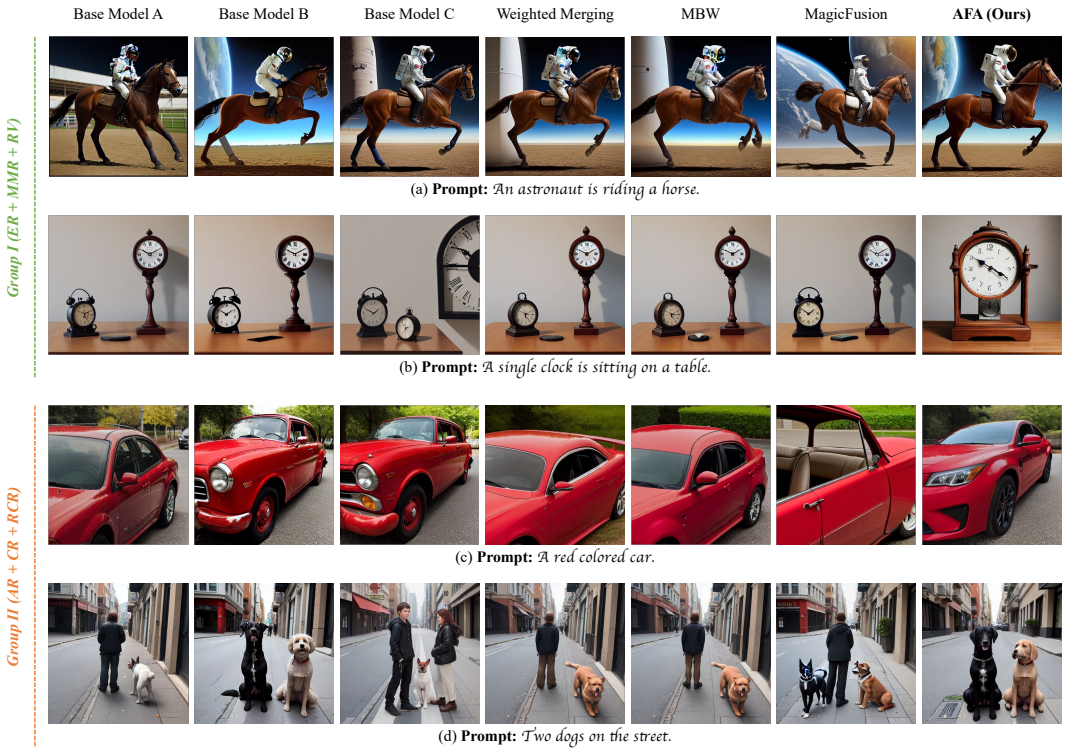
Evaluation Protocols. For a fair comparison, all the methods generate 4 images by DDIM Ho et al. (2020) for 50 inference steps. The CFG weight β_{CFG} is set to 7.5. We evaluate the models with two datasets, which are COCO 2017 (Lin et al., 2014) and Draw Bench Prompts (Saharia et al., 2022), respectively. For COCO 2017, we apply four metrics, which are Fréchet Inception Distance (FID), Inception Score (IS), CLIP-I, and CLIP-T, respectively. For Draw Bench Prompts,we apply 4 evaluation metrics, which are AES, Pick Score (PS) (Kirstain et al., 2023), HPSv2 (Wu et al., 2023), and Image Reward (IR) (Xu et al., 2023), respectively. More details can be found in Appendix B.

Baselines. We compare our AFA with several methods, including three merging-based methods (i.e., Weighted-Merging, MBW, and autoMBW) and one ensembling-based method (i.e., MagicFusion (Zhao et al., 2023)). The evaluation of these baselines follows the protocols.

4.2 Quantitative Comparison

The results of the two evaluation protocols are summarized in Table 1 and Table 2. Generally, compared with base models, performance improvements are achieved by our AFA. For COCO 2017, after being ensembled by AFA, FID, IS, CLIP-I, and CLIP-T exhibit enhancements under both Group I and Group II. This indicates that the quality of our generated images is more closely with the dataset images. Additionally, the contexts of our generated images align more closely with those of the dataset images and corresponding captions. For Draw Bench Prompts, all four metrics show improvements under both Group I and Group II. The improvement on AES suggests that the aesthetics of our generated images surpass those generated by the base models. The improvements on PS, HPSv2, and IR suggest that the general generation capability can be boosted by applying AFA. Overall, the quantitative results validate the effectiveness of our AFA in ensembling text-to-image diffusion models.

While MagicFusion attains impressive performance in style-transferring and object-binding by ensembling two U-Net denoisers, it fails to achieve superior general generation when ensembling multiple U-Net denoisers. The merging-based methods, such as Weighted Merging, MBW, and autoMBW, do not always produce models with superior generation capabilities. For instance, under Group II, Weighted Merging achieves a better IR compared to the base models, but under Group I, it yields a lower IR. This could be attributed to the fixed contributions of merged models, which are determined by a set of predefined merging weights. Consequently, these methods fail to effectively



(a) **Prompt:** *An astronaut is riding a horse.*

(b) **Prompt:** *A single clock is sitting on a table.*

(c) **Prompt:** *A red colored car.*

(d) **Prompt:** *Two dogs on the street.*

Figure 5: Qualitative comparison between AFA with the base models and the baselines.

facilitate collaboration between models to compensate for each other's weaknesses and improve their generative capabilities. Compared with these baselines, our AFA consistently achieves better performance, demonstrating its superiority.

As depicted in Figure 3, we use AFA to ensemble two models from Group I or Group II, and present the IR scores of base models and the ensembled models on Draw Bench Prompts. The results show that AFA significantly improves the performance when ensembling two U-Net denoisers. The results presented in Table 1, Table 2, and Figure 3 indicate that AFA can be effectively applied to various model combinations and can be extended to accommodate different numbers of models. Meanwhile, it consistently achieves superior generation performance compared to a single base model.

4.3 Qualitative Comparison

As shown in Figure 5, we present a qualitative comparison of the results from the base models, some of the baseline methods, and our AFA, which allows us to make several insightful observations. Firstly, AFA can generate images with better aesthetics. For example, in Figure 5 (a) and (c), AFA can generate images with improved composition and finer details, compared to both the base models and the baseline methods. Secondly, AFA excels in achieving superior context alignment. For example, in Figure 5 (b), all the base models and the baseline methods generate images containing more than *one clock*, despite the textual prompt specifying *a single clock*. In contrast, only our AFA generates the image with just one clock, accurately reflecting the provided context. Thirdly, AFA can focus on the correct context of the base models and drop out the incorrect context. For example, in Figure 5 (d), only the base model B generates the image with *two dogs*, while the baseline methods do not fully trust the base model B, and generate the images with *people*. Only our AFA fully trusts the base model B, and generates the image with *two dogs*, which aligns with the textual context.

4.4 Model Study

Visualization of Attention Maps. As shown in Figure 6, we select to visualize the attention maps of the first and the penultimate (i.e., 24-th) blocks, both of which are learned by ensembling models

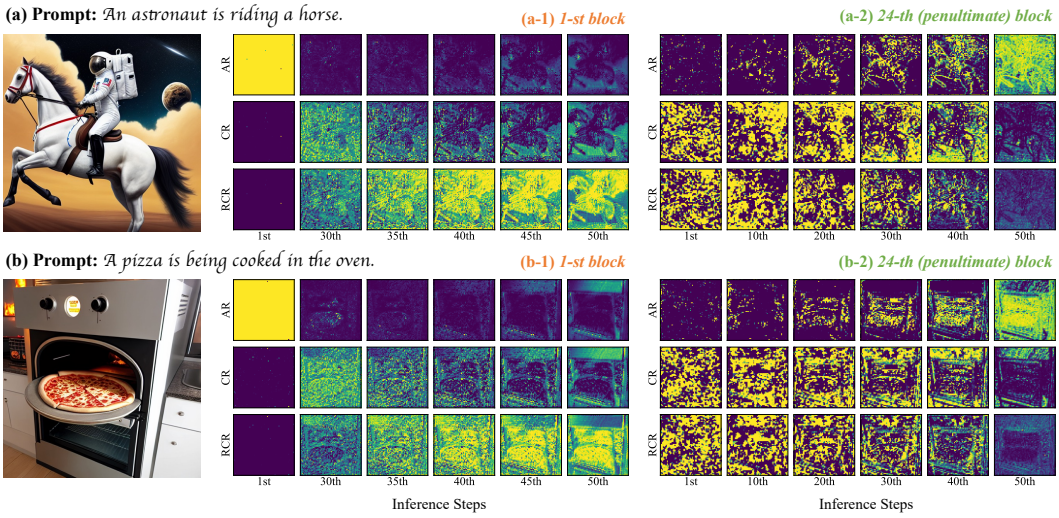


Figure 6: Visualization of the learned attention maps. In each attention map, the position with a lighter color means a higher attention weight. Conversely, the position with a darker color means a lower attention weight.

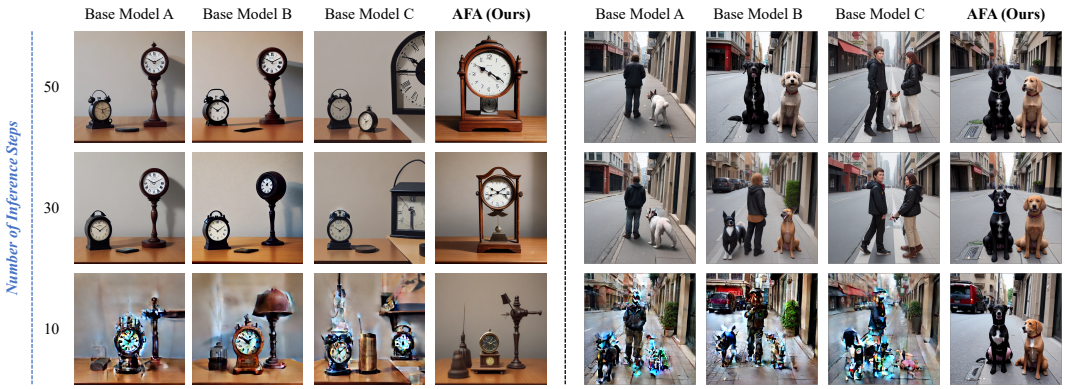
in Group II through our AFA. For the first block, the learned attention maps seem to concentrate solely on the feature map of AR at the first inference stage. However, at the last few inference steps (i.e., from the 30th to the 50th step), the roles of the feature maps from different models begin to diverge. The attention maps tend to highlight the contextual body (i.e., the *astronaut* and the *horse* in Figure 6 (a), and the *pizza* and the *oven* in Figure 6 (b)) in the feature map of RCR, while focusing on the background elements in the feature maps of AR and CR. For the penultimate block, as the inference progresses, the learned attention maps gradually emphasize the contextual body within the feature map of AR, while concentrating on the background within the feature maps of both CR and RCR. Overall, the visualization of the learned attention maps indicates that our AFA can effectively ensemble diffusion models based on the context and the timesteps.

Effect of Fewer Inference Steps. As shown in Figure 4 and Figure 7, we investigate the resilience of our AFA to a reduction in the inference steps. Figure 4 clearly shows a significant drop in the performance of the base models when the number of inference steps is reduced. However, the performance of our AFA remains almost stable when the number of inference steps is reduced from 50 to 20, and only experiences a slight decline when the number of inference steps is reduced to 10. Additionally, in Figure 7, we can find that the quality of the images generated by the base models deteriorates with the reduction of inference steps. In contrast, our AFA stably generates high-quality images. Overall, it indicates that greater tolerance for fewer inference steps will be achieved after ensembling by AFA. Furthermore, while a single inference step may require a proportional increase in time, the total time consumed by the entire inference process does not escalate in the same manner.

Ablation Study. As shown in Table 3, we conduct the ablation study to investigate the necessity of each component within our AFA. Firstly, we conduct experiments with the following settings. (1) *Only Ensembling Last Block*: The AFA ensembles only the last block, leaving all other blocks unaltered. (2) *Block-Wise Averaging*: Each block is ensembled by averaging the output feature maps, rather than using SABW. (3) *Noise Averaging*: Only the last block is ensembled by averaging the output feature maps (i.e., predicted noises). The observed performance declines in these settings suggest that the design of the AFA effectively contributes to the ensembling

	IR (Group I)	IR (Group II)
AFA (Full Model)	**.4388**	**.4892**
Only Ensembling Last Block	.4176	.4374
Block-Wise Averaging	.4001	.4372
Noise Averaging	.3919	.4355
w/o Spatial Location	.4044	.4610
w/o Timestep	.4235	.4622
w/o Textual Condition	.4163	.4559

Table 3: Ablation study of AFA.



(a) **Prompt:** *A single clock is sitting on a table.* (b) **Prompt:** *Two dogs on the street.*

Figure 7: Qualitative comparison under different inference steps.

of diffusion models, thereby leading to improved generation. Next, we conduct an ablation study on the spatial location from AFA, which learn only a single attention score (i.e., a scalar) for the feature of each U-Net denoiser, as opposed to an attention map. The observed decrease in performance suggests that learning a spatial attention map to adaptively adjust the weights of various U-Net denoisers in the level of spatial location is beneficial for the ensemble of different diffusion models. Finally, we individually ablate the introduced timestep and the textual condition. The decreasing performance indicates that both the timestep and textual condition play crucial roles for AFA.

Discussion on Computational Efficiency. (1) *Ensembling Methods.* Compared with another ensembling method (i.e., MagicFusion (Zhao et al., 2023)), the computational efficiency of our AFA for a single inference step aligns closely with it. Both AFA and MagicFusion necessitate running all base models in one inference step. Although SABW in AFA introduces additional parameters, which leads to more computations, it has fewer than the base models (about 1/16). To be precise, our AFA may be marginally less efficient than MagicFusion on single inference steps, but the difference is practically negligible. (2) *Merging Methods.* Because the various models are collapsed into one model, the computational efficiency of the merging methods equals that of a single base model. For the base models and the merging methods, a single image generation takes about 7 seconds with 50 inference steps. When ensembling three such base models, a single generation will take about 22 seconds with the same inference steps. Compared with ensembling, the computation efficiency of one inference step of the merging methods is significantly higher. However, as we discussed in *Effect of Fewer Inference Steps* of Section 4.4, AFA shows remarkable tolerance for fewer inference steps. When limited to 20 inference steps, the performance of AFA does not significantly decline (as shown in Figure 4), while the time consumed is reduced to about 8.8 seconds. This suggests that AFA has comparable computational efficiency with that of the base models or the merging methods throughout the entire inference process.

5 Conclusion

In this paper, we aim to ensemble multiple diffusion models to achieve better generation performance. To this end, we propose the Adaptive Feature Aggregation (AFA) method, which dynamically adjusts the contributions of multiple models at the feature level according to different prompts, initial noises, denoising steps, and spatial locations. Specifically, we design a lightweight Spatial-Aware Block-Wise feature aggregator that produces attention maps to adaptively aggregate the block-level intermediate features from multiple U-Net denoisers. The quantitative and qualitative experiments demonstrate that AFA achieves improvements in image quality and context alignment.

Limitation. A significant limitation of our AFA is that a single inference step may necessitate a proportional increase in inference time, resulting in lower computational efficiency for that step. Fortunately, due to its high tolerance for fewer inference steps, the total time consumed by the entire inference process does not escalate in the same manner. This results in a computational efficiency that is on par with that of the base models or the merging methods.

REFERENCES

AbsoluteReality. https://civitai.com/models/81458?modelVersionId=108576.

AES. https://laion.ai/blog/laion-aesthetics/.

Samuel K Ainsworth, Jonathan Hayase, and Siddhartha Srinivasa. Git re-basin: Merging models modulo permutation symmetries. *arXiv preprint arXiv:2209.04836*, 2022.

autoMBW. https://github.com/Xerxemi/sdweb-auto-MBW.

Yogesh Balaji, Seungjun Nah, Xun Huang, Arash Vahdat, Jiaming Song, Karsten Kreis, Miika Aittala, Timo Aila, Samuli Laine, Bryan Catanzaro, et al. ediffi: Text-to-image diffusion models with an ensemble of expert denoisers. *arXiv preprint arXiv:2211.01324*, 2022.

Zewen Chi, Li Dong, Shaohan Huang, Damai Dai, Shuming Ma, Barun Patra, Saksham Singhal, Payal Bajaj, Xia Song, Xian-Ling Mao, et al. On the representation collapse of sparse mixture of experts. *NeurIPS*, 2022.

CyberRealistic. https://civitai.com/models/15003?modelVersionId=89680.

Ming Ding, Zhuoyi Yang, Wenyi Hong, Wendi Zheng, Chang Zhou, Da Yin, Junyang Lin, Xu Zou, Zhou Shao, Hongxia Yang, et al. Cogview: Mastering text-to-image generation via transformers. *NeurIPS*, 2021.

Shachar Don-Yehiya, Elad Venezian, Colin Raffel, Noam Slonim, Yoav Katz, and Leshem Choshen. Cold fusion: Collaborative descent for distributed multitask finetuning. *arXiv preprint arXiv:2212.01378*, 2022.

Nan Du, Yanping Huang, Andrew M Dai, Simon Tong, Dmitry Lepikhin, Yuanzhong Xu, Maxim Krikun, Yanqi Zhou, Adams Wei Yu, Orhan Firat, et al. Glam: Efficient scaling of language models with mixture-of-experts. In *ICML*, 2022.

David Eigen, Marc'Aurelio Ranzato, and Ilya Sutskever. Learning factored representations in a deep mixture of experts. *arXiv preprint arXiv:1312.4314*, 2013.

EpicRealism. https://civitai.com/models/25694?modelVersionId=134065.

William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *JMLR*, 23(120):1–39, 2022.

Zhida Feng, Zhenyu Zhang, Xintong Yu, Yewei Fang, Lanxin Li, Xuyi Chen, Yuxiang Lu, Jiaxiang Liu, Weichong Yin, Shikun Feng, et al. Ernie-vilg 2.0: Improving text-to-image diffusion model with knowledge-enhanced mixture-of-denoising-experts. In *CVPR*, 2023.

Jonathan Frankle, Gintare Karolina Dziugaite, Daniel Roy, and Michael Carbin. Linear mode connectivity and the lottery ticket hypothesis. In *International Conference on Machine Learning*, pp. 3259–3269. PMLR, 2020.

Yoav Freund and Robert E Schapire. A desicion-theoretic generalization of on-line learning and an application to boosting. In *European conference on computational learning theory*, pp. 23–37. Springer, 1995.

Oran Gafni, Adam Polyak, Oron Ashual, Shelly Sheynin, Devi Parikh, and Yaniv Taigman. Make-a-scene: Scene-based text-to-image generation with human priors. In *ECCV*, 2022.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *NeurIPS*, 2014.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

Jack Hessel, Ari Holtzman, Maxwell Forbes, Ronan Le Bras, and Yejin Choi. Clipscore: A reference-free evaluation metric for image captioning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 7514–7528, 2021.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *NeurIPS*, 2020.

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. *arXiv preprint arXiv:2212.04089*, 2022a.

Gabriel Ilharco, Mitchell Wortsman, Samir Yitzhak Gadre, Shuran Song, Hannaneh Hajishirzi, Simon Kornblith, Ali Farhadi, and Ludwig Schmidt. Patching open-vocabulary models by interpolating weights. *NeurIPS*, 2022b.

Xisen Jin, Xiang Ren, Daniel Preotiuc-Pietro, and Pengxiang Cheng. Dataless knowledge fusion by merging weights of language models. *arXiv preprint arXiv:2212.09849*, 2022.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Yuval Kirstain, Adam Polyak, Uriel Singer, Shahbuland Matiana, Joe Penna, and Omer Levy. Pick-a-pic: An open dataset of user preferences for text-to-image generation. *arXiv preprint arXiv:2305.01569*, 2023.

Nupur Kumari, Richard Zhang, Eli Shechtman, and Jun-Yan Zhu. Ensembling off-the-shelf models for gan training. In *CVPR*, 2022.

Yunsung Lee, JinYoung Kim, Hyojun Go, Myeongho Jeong, Shinhyeok Oh, and Seungtaek Choi. Multi-architecture multi-expert diffusion models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 13427–13436, 2024.

Margaret Li, Suchin Gururangan, Tim Dettmers, Mike Lewis, Tim Althoff, Noah A Smith, and Luke Zettlemoyer. Branch-train-merge: Embarrassingly parallel training of expert language models. *arXiv preprint arXiv:2208.03306*, 2022.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014.

Luping Liu, Yi Ren, Zhijie Lin, and Zhou Zhao. Pseudo numerical methods for diffusion models on manifolds. *arXiv preprint arXiv:2202.09778*, 2022.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps. *NeurIPS*, 2022a.

Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver++: Fast solver for guided sampling of diffusion probabilistic models. *arXiv preprint arXiv:2211.01095*, 2022b.

Simian Luo, Yiqin Tan, Longbo Huang, Jian Li, and Hang Zhao. Latent consistency models: Synthesizing high-resolution images with few-step inference. *arXiv preprint arXiv:2310.04378*, 2023.

MajicMixRealistic. `https://civitai.com/models/43331?modelVersionId=176425`.

Michael S Matena and Colin A Raffel. Merging models with fisher-weighted averaging. *NeurIPS*, 2022.

MBW. `https://github.com/bbc-mc/sdweb-merge-block-weighted-gui`.

Joao Mendes-Moreira, Carlos Soares, Alípio Mário Jorge, and Jorge Freire De Sousa. Ensemble approaches for regression: A survey. *Acm computing surveys (csur)*, 45(1):1–40, 2012.

Alex Nichol, Prafulla Dhariwal, Aditya Ramesh, Pranav Shyam, Pamela Mishkin, Bob McGrew, Ilya Sutskever, and Mark Chen. Glide: Towards photorealistic image generation and editing with text-guided diffusion models. *arXiv preprint arXiv:2112.10741*, 2021.

Guillermo Ortiz-Jimenez, Alessandro Favero, and Pascal Frossard. Task arithmetic in the tangent space: Improved editing of pre-trained models. *arXiv preprint arXiv:2305.12827*, 2023.

Junting Pan, Keqiang Sun, Yuying Ge, Hao Li, Haodong Duan, Xiaoshi Wu, Renrui Zhang, Aojun Zhou, Zipeng Qin, Yi Wang, et al. Journeydb: A benchmark for generative image understanding. *arXiv preprint arXiv:2307.00716*, 2023.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4195–4205, 2023.

Dustin Podell, Zion English, Kyle Lacey, Andreas Blattmann, Tim Dockhorn, Jonas Müller, Joe Penna, and Robin Rombach. Sdxl: Improving latent diffusion models for high-resolution image synthesis. *arXiv preprint arXiv:2307.01952*, 2023.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pp. 8748–8763. PMLR, 2021.

Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In *International Conference on Machine Learning*, pp. 8821–8831. PMLR, 2021.

Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 1(2):3, 2022.

RealCartoonReslistic. `https://civitai.com/models/97744?modelVersionId=104496`.

RealisticVision. `https://civitai.com/models/4201?modelVersionId=130072`.

Ye Ren, PN Suganthan, and N Srikanth. Ensemble methods for wind and solar power forecasting—a state-of-the-art review. *Renewable and Sustainable Energy Reviews*, 50:82–91, 2015.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International conference on machine learning*, pp. 1278–1286. PMLR, 2014.

Carlos Riquelme, Joan Puigcerver, Basil Mustafa, Maxim Neumann, Rodolphe Jenatton, André Susano Pinto, Daniel Keysers, and Neil Houlsby. Scaling vision with sparse mixture of experts. *NeurIPS*, 2021.

Lior Rokach. Ensemble-based classifiers. *Artificial intelligence review*, 33:1–39, 2010.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, 2022.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pp. 234–241. Springer, 2015.

Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, et al. Photorealistic text-to-image diffusion models with deep language understanding. *NeurIPS*, 2022.

SDv1.5. `https://huggingface.co/runwayml/stable-diffusion-v1-5`.

Sheng Shen, Zhewei Yao, Chunyuan Li, Trevor Darrell, Kurt Keutzer, and Yuxiong He. Scaling vision-language models with sparse mixture of experts. *arXiv preprint arXiv:2303.07226*, 2023.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pp. 2256–2265. PMLR, 2015.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *ICLR*, 2020.

George Stoica, Daniel Bolya, Jakob Bjorner, Taylor Hearn, and Judy Hoffman. Zipit! merging models from different tasks without training. *arXiv preprint arXiv:2305.03053*, 2023.

Ming Tao, Hao Tang, Fei Wu, Xiao-Yuan Jing, Bing-Kun Bao, and Changsheng Xu. Df-gan: A simple and effective baseline for text-to-image synthesis. In *CVPR*, 2022.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *NeurIPS*, 2017.

Sandro Vega-Pons and José Ruiz-Shulcloper. A survey of clustering ensemble algorithms. *International Journal of Pattern Recognition and Artificial Intelligence*, 25(03):337–372, 2011.

Zijie J Wang, Evan Montoya, David Munechika, Haoyang Yang, Benjamin Hoover, and Duen Horng Chau. Diffusiondb: A large-scale prompt gallery dataset for text-to-image generative models. *arXiv preprint arXiv:2210.14896*, 2022.

Weighted-Merging. `https://github.com/hako-mikan/sd-webui-supermerger`.

Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, et al. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *International Conference on Machine Learning*, pp. 23965–23998. PMLR, 2022a.

Mitchell Wortsman, Gabriel Ilharco, Jong Wook Kim, Mike Li, Simon Kornblith, Rebecca Roelofs, Raphael Gontijo Lopes, Hannaneh Hajishirzi, Ali Farhadi, Hongseok Namkoong, et al. Robust fine-tuning of zero-shot models. In *CVPR*, 2022b.

Xiaoshi Wu, Yiming Hao, Keqiang Sun, Yixiong Chen, Feng Zhu, Rui Zhao, and Hongsheng Li. Human preference score v2: A solid benchmark for evaluating human preferences of text-to-image synthesis. *arXiv preprint arXiv:2306.09341*, 2023.

Jiazheng Xu, Xiao Liu, Yuchen Wu, Yuxuan Tong, Qinkai Li, Ming Ding, Jie Tang, and Yuxiao Dong. Imagereward: Learning and evaluating human preferences for text-to-image generation. *arXiv preprint arXiv:2304.05977*, 2023.

Tao Xu, Pengchuan Zhang, Qiuyuan Huang, Han Zhang, Zhe Gan, Xiaolei Huang, and Xiaodong He. Attngan: Fine-grained text to image generation with attentional generative adversarial networks. In *CVPR*, 2018.

Zeyue Xue, Guanglu Song, Qiushan Guo, Boxiao Liu, Zhuofan Zong, Yu Liu, and Ping Luo. Raphael: Text-to-image generation via large mixture of diffusion paths. *arXiv preprint arXiv:2305.18295*, 2023.

Pengyi Yang, Yee Hwa Yang, Bing B Zhou, and Albert Y Zomaya. A review of ensemble methods in bioinformatics. *Current Bioinformatics*, 5(4):296–308, 2010.

Jiahui Yu, Yuanzhong Xu, Jing Yu Koh, Thang Luong, Gunjan Baid, Zirui Wang, Vijay Vasudevan, Alexander Ku, Yinfei Yang, Burcu Karagol Ayan, et al. Scaling autoregressive models for content-rich text-to-image generation. *arXiv preprint arXiv:2206.10789*, 2022.

Han Zhang, Jing Yu Koh, Jason Baldridge, Honglak Lee, and Yinfei Yang. Cross-modal contrastive learning for text-to-image generation. In *CVPR*, 2021a.

Han Zhang, Weichong Yin, Yewei Fang, Lanxin Li, Boqiang Duan, Zhihua Wu, Yu Sun, Hao Tian, Hua Wu, and Haifeng Wang. Ernie-vilg: Unified generative pre-training for bidirectional vision-language generation. *arXiv preprint arXiv:2112.15283*, 2021b.

Jinghan Zhang, Shiqi Chen, Junteng Liu, and Junxian He. Composing parameter-efficient modules with arithmetic operations. *arXiv preprint arXiv:2306.14870*, 2023.

Jing Zhao, Heliang Zheng, Chaoyue Wang, Long Lan, and Wenjing Yang. Magicfusion: Boosting text-to-image generation performance by fusing diffusion models. *arXiv preprint arXiv:2303.13126*, 2023.

Ying Zhao, Jun Gao, and Xuezhi Yang. A survey of neural network ensembles. In *2005 international conference on neural networks and brain*, volume 1, pp. 438–442. IEEE, 2005.

Zhi-Hua Zhou. *Ensemble methods: foundations and algorithms*. CRC press, 2012.

Minfeng Zhu, Pingbo Pan, Wei Chen, and Yi Yang. Dm-gan: Dynamic memory generative adversarial networks for text-to-image synthesis. In *CVPR*, 2019.

A Details about Parameters

The parameters of AFA are concentrated in the SABW module. The SABW module comprises three components, which are a ResNet layer, a Transformer layer, and a convolution layer, respectively. Among them, the number of parameters in both the ResNet layer and the convolution layer varies depending on the number of ensembled models. For instance, when AFA ensembles two diffusion models, SABW contains approximately 42.425 million parameters. With each additional diffusion model ensembled, the parameter count of SABW increases by about 2.707 million. Therefore, when three diffusion models are ensembled, the total number of the trainable parameters is 45.132 million, and with four models, it rises to approximately 47.839 million. In conclusion, we estimate that AFA encompasses close to 50 million trainable parameters.

B Details about Evaluation Protocols

COCO 2017 comprises 118,287 and 5,000 image-caption pairs in the test and validation sets. All the models generate images with a resolution of 256×256. We apply four metrics to evaluate the generation performance, which are Fréchet Inception Distance (FID), Inception Score (IS), CLIP-I, and CLIP-T, respectively. FID and IS are applied to the test set, while CLIP-I and CLIP-T are applied to the validation set. Both FID and IS assess the quality of generated images. Note that lower FID means better quality. CLIP-I is the similarity between the CLIP (Radford et al., 2021) images embeddings of generated images and that of images from the image-caption pairs. CLIP-T is the CLIPScore (Hessel et al., 2021) between the generated images with the captions.

Draw Bench Prompts contains 200 evaluation prompts. All the models generate images with a resolution of 512×512. We apply 4 evaluation metrics, which are AES, Pick Score (PS) (Kirstain et al., 2023), HPSv2 (Wu et al., 2023), and Image Reward (IR) (Xu et al., 2023), respectively. All three metrics evaluate the performance by a model that simulates human preferences. The evaluations are conducted 20 times to ensure statistical significance. And the averaged metrics are reported.

C Effect of More Training Samples

We conduct an experiment to assess the influence of an increased number of training samples on the performance of AFA. For this purpose, we vary the number of training samples from 0 to 18,000, increasing the increment of 2,000. As depicted in Figure 8, the trend of the lines initially rises and then stabilizes after reaching 10,000 training samples. This suggests that approximately 10,000 training samples are sufficient for AFA. More training samples do not bring greater performance. It may be due to the inherent limitations in the generative capabilities of the base models.

D More Qualitative Comparisons

More qualitative comparisons between our AFA and the base models can be found in Figure 9.

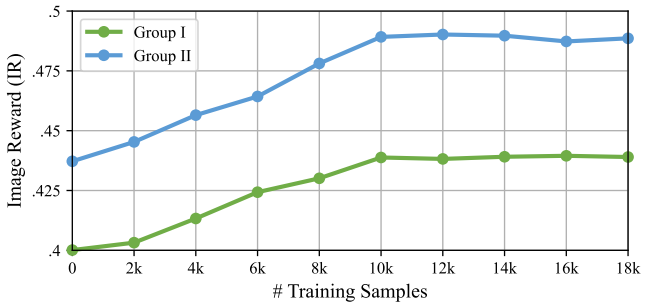
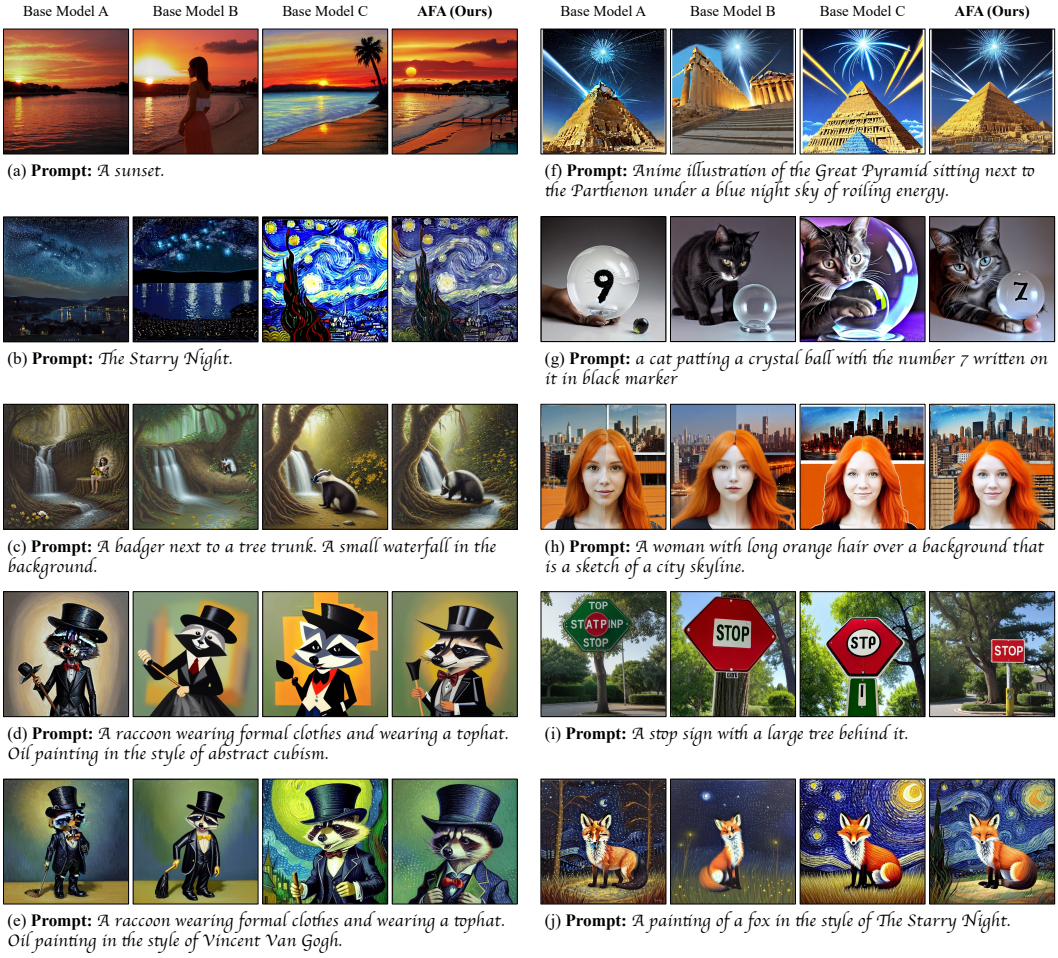


Figure 8: Effect of varying training samples.



(a) **Prompt:** *A sunset.*

(b) **Prompt:** *The Starry Night.*

(c) **Prompt:** *A badger next to a tree trunk. A small waterfall in the background.*

(d) **Prompt:** *A raccoon wearing formal clothes and wearing a tophat. Oil painting in the style of abstract cubism.*

(e) **Prompt:** *A raccoon wearing formal clothes and wearing a tophat. Oil painting in the style of Vincent Van Gogh.*

(f) **Prompt:** *Anime illustration of the Great Pyramid sitting next to the Parthenon under a blue night sky of roiling energy.*

(g) **Prompt:** *a cat patting a crystal ball with the number 7 written on it in black marker*

(h) **Prompt:** *A woman with long orange hair over a background that is a sketch of a city skyline.*

(i) **Prompt:** *A stop sign with a large tree behind it.*

(j) **Prompt:** *A painting of a fox in the style of The Starry Night.*

Figure 9: More qualitative comparisons between our AFA and the base models.

E Ensembling by Mixture-of-Experts

Another intuitive approach to ensembling multiple models involves the use of the Mixture-of-Experts (MoE) method Shen et al. (2023); Eigen et al. (2013); Riquelme et al. (2021); Fedus et al. (2022); Du et al. (2022); Chi et al. (2022). The MoE framework incorporates multiple models, each designated as an *expert*. Given an input, MoE employs a routing mechanism to determine the most suitable expert. The input is then processed by the selected expert, and its output is considered as the output of the entire model. The number of parameters in MoE proportionally increases with the number of ensembled experts. However, the parameters of MoE are sparse, indicating that not all parameters are utilized during the inference process. Despite ensembling multiple models, only one of them is activated during inference, which makes MoE efficient.

Several text-to-image generation methods leverage MoE-based diffusion models, where multiple denoisers are ensembled along the timesteps. In these methods, only one denoising expert is activated at each timestep. For instance, ERNIE-ViLG 2.0 (Feng et al., 2023) and eDiff-I (Balaji et al., 2022) divide all the timesteps into blocks, each of which consists of consecutive timesteps and is assigned to a denoising expert. Building on this idea, MEME (Lee et al., 2024) introduces denoisers with different architectures, tailored to distinct timestep intervals. In these MoE-based methods, the denoising capability of each denoiser is restricted into some specific consecutive timesteps through large-scale full-parameters training, which is not suitable to the task of model ensembling.

Inspired by the MoE-based diffusion models discussed above, an intuitive MoE-based ensembling method is to employ a trained router to select a base denoiser for the current timesteps. We refer to this as the *Denoiser-Level MoE* method. Furthermore, to mirror the concept of our AFA, we propose another strategy of *Block-Level MoE* method, which uses a router to select the next block. The key difference between the Block-Level MoE method and our AFA lies in their operations: AFA manipulates the outputs of all blocks, whereas MoE directly acts on the input. We will introduce the details of Block-Level MoE method, as the Denoiser-Level MoE method can be considered a special case of *Block-Level MoE* method, where the selection occurs only before the first block.

Given N diffusion models, we employ a router $r_\zeta^{(j)}$ to determine which model's block will be used, where ζ denotes the learnable parameters. Specifically, $r_\zeta^{(j)}$ output the logits of N models,

$$\mathbf{l}_t^{(j)} = r_\zeta^{(j)}(\mathbf{x}_t^{(j)}, \mathbf{c}, t) \in \mathbb{R}^N \ . \tag{13}$$

During training, the probability for the i-th model is defined by the Gumbel softmax,

$$p_i = \frac{\exp((l_{t,i}^{(j)} + g)/\tau)}{\sum_{k=1}^{N} \exp((l_{t,k}^{(j)} + g)/\tau)} \ , \tag{14}$$

where, $l_{t,i}^{(j)}$ is the i-th element of $\mathbf{l}_t^{(j)}$. g is the Gumbel noise. τ is the temperature coefficient. To address the issue of back-propagation incapability, we apply the reparameterization trick. Given the probability distribution $\mathbf{p} = [p_i]_{i=1}^N$, the reparameterized distribution is

$$\mathbf{p}' = \text{onehot}(\mathbf{p}) + \mathbf{p} - \text{sg}(\mathbf{p}) \ . \tag{15}$$

Among them, onehot$(\cdot)$ is the one-hot function, which sets the maximum probability to 1 and the other to 0. sg$(\cdot)$ is the stop gradient function. During the forward process, the distribution is defined as $\mathbf{p}' = \text{onehot}(\mathbf{p})$. However, during the backward process, the distribution of $\mathbf{p}' = \mathbf{p}$ is used to compute the gradients. The output of the j-th block is

$$\mathbf{y}_t^{(j)} = \sum \mathbf{p}' \otimes \text{concat}(\{\mathbf{y}_{t,i}^{(j)}\}_{i=1}^N) \ . \tag{16}$$

From Eq. 16, we can infer that the forward time is proportional to the number of ensembled models during training. The training loss is the denoising loss.

During inference, only the model that achieves maximum logits will be used to process the block input,

$$\mathbf{y}_t^{(j)} = \boldsymbol{\epsilon}_{\theta_k}^{(j)}(\mathbf{x}_t^{(j)}, \mathbf{c}, t) \ , \quad k = \arg\max_{i \in [N]} \ l_{t,i}^{(j)} \ . \tag{17}$$

The forward time of the ensembled model is the same as that of a single model during inference.

To implement the router $r_\zeta^{(j)}$, we apply a ResNet layer to introduce timesteps by adding the time embedding into the feature, and a Transformer layer to introduce textual prompts by the cross-attention. Specifically, given the input for the j-th block (i.e., $\mathbf{x}_t^{(j)} \in \mathbb{R}^{h_j \times w_j \times c_j}$), the output logits for N models (i.e,, $\mathbf{l}_t^{(j)}$) are achieved by

$$\mathbf{h}_t^{(j)} \leftarrow \text{ResLayer}(\mathbf{x}_t^{(j)} + \gamma(t)) \in \mathbb{R}^{h_j \times w_j \times d} \ , \tag{18}$$

$$\mathbf{o}_t^{(j)} \leftarrow \text{TransformerLayer}(\mathbf{h}_t^{(j)}, \mathbf{c}) \in \mathbb{R}^{h_j \times w_j \times d} \ , \tag{19}$$

$$\mathbf{l}_t^{(j)} \leftarrow \text{Linear}(\text{AvgPool}(\mathbf{o}_t^{(j)})) \in \mathbb{R}^N \ . \tag{20}$$

Although the time required for a single forward pass in the MoE methods is approximately equivalent to that of a single model, they face challenges in performing parallel computations when generating multiple images simultaneously. This limitation becomes evident during classifier-free guidance. If the router selects two different models to process the conditioned and unconditioned inputs, MoE can only execute two sequential forward passes, rather than concurrently processing these two scenarios within a single forward pass. Moreover, when generating K images at once, the MoE method has a high probability of necessitating $2K$ forward passes. While these $2K$ inputs can be divided among N models, this only reduces the complexity associated with the number of forward passes from $O(K)$ to $O(N)$. This reduction still represents a substantial computational demand.

The quantitative comparison among the base models, the MoE methods, and our AFA is summarized in Table 4. AFA demonstrates superior performance compared to the MoE methods. This may be because the base denoisers have well-balanced denoising capabilities across all timesteps, and selecting a specific denoiser for a given timestep does not significantly enhance the overall denoising capability of the ensembled model. Furthermore, we compare AFA and the MoE methods using fewer inference steps (i.e., 20 steps). As shown in Table 5, unlike our AFA, the MoE methods do not demonstrate robustness with fewer inference steps, performing similarity to the base models.

The efficiency comparison among the base models, the MoE methods, and our AFA is summarized in Table 6. Firstly, the MoE methods have a similar number of parameters and training TFLOPs as our AFA. This is because the MoE methods still require all base model to be available for selection, and during training, each base model must perform a forward pass. Secondly, during inference, the MoE methods exhibit lower TFLOPs compared to AFA, as only one base model is activated at each timestep. However, despite this advantage in TFLOPs, the inference time of the MoE methods is comparable to that of AFA due to their inability to leverage parallel inference effectively. Moreover, the MoE methods do not support fewer inference steps. In contrast, when AFA reduces the number of inference steps to 20, it still outperforms the MoE methods, even when they use 50 inference steps, while also achieving lower TFLOPs and faster inference. This highlights the efficiency and effectiveness of our AFA.

	Group I				Group II			
	AES	PS	HPSv2	IR	AES	PS	HPSv2	IR
Base Model A	5.4102	21.6279	27.8007	.3544	5.5641	21.8013	28.0183	.4238
Base Model B	5.5013	21.4624	27.7246	.2835	5.5027	21.7249	28.0343	.4202
Base Model C	5.4881	21.8031	27.9652	.3922	5.5712	21.4936	27.8089	.3367
Denoiser-Level MoE	5.4727	21.6893	27.5345	.4019	5.5638	21.5531	27.7843	.4380
Block-Level MoE	5.5037	21.7690	27.8749	.4037	5.5718	21.7636	27.9842	.4463
AFA (Ours)	**5.5201**	**21.8263**	**27.9734**	**.4388**	**5.5798**	**21.8059**	**28.0371**	**.4892**

Table 4: Quantitative comparison between AFA with the base models and the MoE methods.

	Group I				Group II			
	AES	PS	HPSv2	IR	AES	PS	HPSv2	IR
Denoiser-Level MoE	1.2310	2.9435	4.1739	.0184	2.4556	3.6723	5.3415	.0237
Block-Level MoE	1.6854	3.4573	5.3583	.0098	2.5734	4.2368	4.7252	.0344
AFA (Ours)	**5.4951**	**21.8093**	**27.7347**	**.4191**	**5.5322**	**21.7830**	**27.9775**	**.4803**

Table 5: Quantitative comparison of AFA and the MoE methods with 20 inference steps.

F Ensembling More Models

To evaluate the scalability of our AFA, we apply it to ensemble a larger number of base models. Specifically, we select and ensemble several base models with best performance, and test IR metrics in Draw Bench Prompts. As shown in Table 7, ensembling more base models leads more performance improvements. Additionally, increasing the number of the ensembled base models enhances the tolerance for fewer inference steps.

			1 image		2 images		4 images	
	# params.†	T-TLOPs	I-TFLOPs	Times (s)‡	I-TFLOPs	Times (s)‡	I-TFLOPs	Times (s)‡
Base Model*	859.52M	–	70.24	2.94	140.45	5.61	280.88	6.87
Denoiser-Level MoE*	2581.01M	6.36	72.37	5.36	144.75	10.37	289.48	20.81
Block-Level MoE*	2621.03M	6.43	74.41	5.39	148.82	10.43	297.65	21.00
AFA (50 inf. steps)	2621.03M	6.42	218.74	8.98	437.48	15.57	874.96	21.33
AFA (20 inf. steps)	2621.03M	6.42	86.61	3.62	173.22	6.41	346.45	8.74

Table 6: Efficiency Comparison between AFA with the base models and the MoE methods when ensembling three base models to generate images with a resolution of 512×512. T-TFLOPs and I-TFLOPs denote TFLOPs for training and inference, respectively. †Note that only the parameters of denoisers are considered. ‡Note that all the inference times are evaluated in our A6000 environment over 20 runs. *Note that both the base models and MoE methods are evaluated with 50 inference steps, because they cannot achieve comparable performance with fewer inference steps.

	1 base model	2 base models	3 base models	4 base models	5 base models	6 base models
50 inference steps	0.4238	0.5003	0.4892	0.4967	0.5042	0.5347
40 inference steps	0.4065	0.4834	0.4874	0.4906	0.4984	0.5310
30 inference steps	0.3593	0.4791	0.4846	0.4853	0.4993	0.5294
20 inference steps	0.2103	0.3641	0.4803	0.4840	0.4975	0.5304
10 inference steps	-0.5255	0.1844	0.4013	0.4285	0.4423	0.5135

Table 7: Performance (IR) comparison across different numbers of ensembled base models and varying inference steps.

G Comparison on Fewer Inference Steps

We compare our AFA with the baseline methods under fewer inference steps. As shown in Table 8, our AFA not only outperforms the baseline methods at higher inference steps but also demonstrates significantly better performance under fewer inference steps. This highlights AFA's superior tolerance to the fewer inference steps compared to the baselines.

	Group I					Group II				
inference steps	50	40	30	20	10	50	40	30	20	10
Wtd. Merging	0.3909	0.3104	0.2451	-0.0535	-0.7536	0.4387	0.4123	0.2942	0.1046	-0.5105
MBW	0.3922	0.3041	0.2345	0.0341	-0.7731	0.4396	0.4094	0.2893	-0.0031	-0.5524
autoMBW	0.3672	0.3158	0.2510	-0.4173	-0.8046	0.3513	0.3545	0.2502	-0.0841	-0.5841
MagicFusion	0.3317	0.3245	0.3104	0.2349	0.0175	0.4194	0.3995	0.3408	0.1951	0.0818
AFA (Ours)	**0.4388**	**0.4389**	**0.4238**	**0.4191**	**0.3575**	**0.4892**	**0.4874**	**0.4846**	**0.4803**	**0.4013**

Table 8: Quantitative comparison (IR) on fewer inference steps.

H Efficiency Comparison

We compare the efficiency of merging methods and ensembling methods. As shown in Table 9, when using the same 50 inference steps, ensembling methods (e.g., MagicFusion and AFA) do not have an efficiency advantage over a single base models or merging methods. The number of model parameters, TFLOPs, and inference time all increase linearly with the number of base models.

However, thanks to AFA's high tolerance for fewer inference steps, it can achieve similar performance with reduced steps, as demonstrated in Table 7, while the baseline methods fail to maintain performance, as shown in Table 8. Therefore, under fewer inference steps, AFA achieves efficiency in terms of TFLOPs and inference time comparable to that of a single base models and merging methods.

	2 base models			3 base models			6 base models		
	# params.†	TFLOPs	Times (s)‡	# params.†	TFLOPs	Times (s)‡	# params.†	TFLOPs	Times (s)‡
Merging Method*	859.52M	70.24	2.94	859.52M	70.24	2.94	859.52M	70.24	2.94
MagicFusion*	1719.04M	138.66	6.01	2578.56M	206.88	9.13	5157.12M	411.53	22.41
AFA (50 inf. steps)	1761.47M	148.64	6.28	2621.03M	218.74	8.98	5210.37M	430.08	23.15
AFA (30 inf. steps)	1761.47M	94.24	3.21	–	–	–	–	–	–
AFA (20 inf. steps)	–	–	–	2621.03M	86.61	3.62	–	–	–
AFA (10 inf. steps)	–	–	–	–	–	–	5210.37M	103.68	4.58

Table 9: Efficiency Comparison between the merging methods and the ensembling methods when generating an image with a resolution of 512×512. †Note that only the parameters of denoisers are considered. ‡Note that all the inference times are evaluated in our A6000 environment over 20 runs. *Note that both the merging methods and MagicFusion are evaluated with 50 inference steps, because they cannot achieve comparable performance with fewer inference steps.

I GENERALITY ON OTHER ARCHITECTURES

To evaluate the generality of our AFA on other diffusion architectures, we select three SDXL (Podell et al., 2023) models and two FLUX .1 [dev][4] models from CivitAI.

SDXL consists two U-Nets: the denoising U-Net and the refining U-Net. Both U-Nets shares a similar architecture of SDv1.5, but contain larger blocks. Consistent with the method used for ensembling SDv1.5 models, we employ the SABW module to aggregate features from each block.

FLUX .1 [dev] is a DiT-based model (Peebles & Xie, 2023). To ensemble models with this architecture, we also utilize the SABW module to aggregate features from each DiT Transformer block.

The selected SDXL models are NoobAI-XL[5], iNiverse-Mix[6], and epiCRealism-XL[7]. The selected FLUX .1 [dev] models are PixelWave[8] and VerusVision[9].

As shown in Table 10, when ensembling models with the SDXL or FLUX .1 [dev] architectures, the performance of the ensembled model surpasses that of the individual base models, demonstrating the generality of our AFA on different model architectures. However, the observed improvement is relatively small, likely because the base models already exhibit strong performance, and ensembling leads to only marginal gains. Additionally, due to the large size of the base models, ensembling multiple such models may be less practical.

It is worth noting that our AFA cannot be applied to ensembling models with different architectures, as the misalignment of block features makes block-wise aggregation challenging. We view ensembling models with diverse architectures as a promising direction for future research.

	SDXL				FLUX .1 [dev]			
	AES	PS	HPSv2	IR	AES	PS	HPSv2	IR
Base Model A	6.1948	21.7854	28.0452	.6314	6.2341	21.8843	28.0958	.7341
Base Model B	6.1363	21.7783	28.0531	.6328	6.2346	21.8593	28.1003	.7324
Base Model C	6.1852	21.7752	27.9984	.6339	–	–	–	–
AFA (Ours)	**6.2190**	**21.7959**	**28.0894**	**.6342**	**6.2490**	**21.8931**	**28.1194**	**.7370**

Table 10: Quantitative comparison of AFA with different diffusion architectures.

[4] https://blackforestlabs.ai/announcing-black-forest-labs/

[5] https://civitai.com/models/833294/noobai-xl-nai-xl?modelVersionId=1070239

[6] https://civitai.com/models/226533/iniverse-mixsfw-and-nsfw?modelVersionId=608842

[7] https://civitai.com/models/277058/epicrealism-xl?modelVersionId=1074830

[8] https://civitai.com/models/141592/pixelwave?modelVersionId=992642

[9] https://civitai.com/models/883426/verus-vision-10b?modelVersionId=988886

J DISTILLATION TO FEWER INFERENCE STEPS

While our AFA demonstrates robustness to fewer inference steps, we aim to further distill the ensembled model into significantly fewer steps. Specifically, we explore using LCM (Luo et al., 2023) to achieve this. In our experiments, we kept the parameters of the denoisers frozen while training only the parameters of the SABW modules. However, as shown in Table 11, this approach is unsuccessful. A possible reason for this failure is that the frozen parameters may have impeded the distillation process.

	Original Ensembled Model		Distilled Ensembled Model			
Inference Steps	50	20	20	4	2	1
IR	0.4892	0.4803	0.4793	-0.5846	-0.5594	-0.5952

Table 11: Experimental results (IR) for distilling the ensembled model with AFA into fewer inference steps using LCM.

K EVALUATION ON MORE DATASETS

To validate the generality of our AFA, we evaluate it on additional datasets, which are DiffusionDB (Wang et al., 2022), JourneyDB (Pan et al., 2023), and LAION-COCO[10], respectively. We randomly selected 50,000 samples from each dataset to compare the performance of AFA against the base models and the baseline methods[11].

The quantitative comparisons on the three additional datasets are presented in Table 12, Table 13, and Table 14, respectively. The results demonstrate that our AFA consistently outperforms both the base models and the baseline methods, highlighting the generality and effectiveness of our approach.

	Group I								Group II							
	FID ↓	IS	CLIP-I	CLIP-T	AES	PS	HPSv2	IR	FID ↓	IS	CLIP-I	CLIP-T	AES	PS	HPSv2	IR
Base Model A	14.78	6.88	.6433	.2987	5.557	23.4113	29.9126	.4354	14.31	7.02	.6578	.3124	5.612	23.5527	30.1521	.4413
Base Model B	15.01	6.93	.6513	.3010	5.456	23.1123	29.8131	.4223	14.98	7.01	.6618	.3128	5.502	23.3128	30.0035	.4307
Base Model C	15.13	6.87	.6441	.2988	5.478	23.3423	29.9341	.4339	15.02	6.94	.6391	.2983	5.481	23.3420	29.9413	.4291
Wtd. Merging	17.41	6.37	.6214	.2776	5.213	23.2975	28.9938	.4007	16.98	6.81	.6231	.2984	5.501	23.2139	28.7310	.4147
MBW	16.67	6.81	.6378	.2843	5.367	23.3002	28.8393	.4115	15.20	6.99	.6392	.2931	5.493	23.4412	29.0012	.4216
autoMBW	16.66	6.76	.6342	.2663	5.377	23.3874	28.8432	.4293	15.17	7.00	.6381	.2891	5.551	23.3584	29.8413	.4298
MagicFusion	14.77	6.98	.6561	.3123	5.674	23.4132	29.9241	.4440	14.21	7.02	.6641	.3125	5.611	23.5049	29.9941	.4391
AFA (Ours)	**12.32**	**7.39**	**.6765**	**.3691**	**5.773**	**23.5132**	**30.0946**	**.4763**	**13.98**	**7.11**	**.6712**	**.3211**	**5.698**	**23.6130**	**30.1931**	**.4712**

Table 12: Quantitative comparison in Group I and II on DiffusionDB.

	Group I								Group II							
	FID ↓	IS	CLIP-I	CLIP-T	AES	PS	HPSv2	IR	FID ↓	IS	CLIP-I	CLIP-T	AES	PS	HPSv2	IR
Base Model A	17.80	5.63	.5324	.2484	4.520	19.2066	24.5997	.3606	17.21	5.75	.5378	.2513	4.577	19.1089	24.5196	.3633
Base Model B	18.17	5.72	.5379	.2501	4.466	18.9965	24.5139	.3458	18.01	5.72	.5428	.2588	4.482	18.9069	24.3972	.3483
Base Model C	17.98	5.62	.5270	.2434	4.507	19.1876	24.5743	.3554	17.87	5.61	.5169	.2398	4.445	18.9523	24.2945	.3443
Wtd. Merging	20.98	5.26	.5170	.2314	4.288	19.1237	23.8558	.3327	20.55	5.50	.5108	.2452	4.455	18.8782	23.3782	.3399
MBW	19.74	5.58	.5218	.2365	4.399	19.1381	23.7068	.3371	18.35	5.74	.5221	.2378	4.484	19.0703	23.5806	.3465
autoMBW	20.07	5.54	.5204	.2227	4.429	19.1960	23.6664	.3508	18.04	5.70	.5182	.2325	4.501	18.9612	24.2498	.3510
MagicFusion	17.76	5.76	.5412	.2591	4.672	19.2147	24.5909	.3666	16.93	5.72	.5414	.2573	4.550	19.0955	24.3816	.3570
AFA (Ours)	**14.86**	**6.04**	**.5587**	**.3063**	**4.763**	**19.3324**	**24.7316**	**.3849**	**16.72**	**5.78**	**.5482**	**.2646**	**4.651**	**19.1986**	**24.5480**	**.3990**

Table 13: Quantitative comparison in Group I and II on JourneyDB.

[10] https://laion.ai/blog/laion-coco/

[11] Note that for JourneyDB, we ensured the selected samples are distinct from those in the training set.

	Group I								Group II							
	FID ↓	IS	CLIP-I	CLIP-T	AES	PS	HPSv2	IR	FID ↓	IS	CLIP-I	CLIP-T	AES	PS	HPSv2	IR
Base Model A	13.82	7.82	.7328	.3432	6.260	26.5012	33.8765	.4962	13.26	8.06	.7549	.3559	6.402	26.8726	34.4362	.5080
Base Model B	14.12	7.89	.7397	.3467	6.164	26.1997	33.7818	.4754	13.87	8.05	.7583	.3617	6.286	26.5950	34.2635	.4910
Base Model C	14.08	7.73	.7280	.3371	6.197	26.4257	33.8805	.4886	13.84	7.90	.7284	.3379	6.254	26.6099	34.1549	.4879
Wtd. Merging	16.46	7.24	.7101	.3209	5.886	26.3613	32.8551	.4573	15.79	7.76	.7149	.3415	6.271	26.5130	32.8277	.4762
MBW	15.45	7.68	.7243	.3275	6.071	26.3787	32.6951	.4688	14.11	8.07	.7344	.3327	6.311	26.7870	33.1185	.4860
autoMBW	15.61	7.67	.7218	.3101	6.092	26.4808	32.6464	.4852	13.98	8.00	.7328	.3280	6.343	26.6503	34.0794	.4946
MagicFusion	13.80	7.92	.7477	.3594	6.425	26.5094	33.8799	.5048	13.13	8.05	.7632	.3632	6.421	26.8431	34.2797	.5024
AFA (Ours)	**11.66**	**8.36**	**.7716**	**.4241**	**6.582**	**26.6673**	**34.1098**	**.5331**	**12.84**	**8.16**	**.7717**	**.3735**	**6.552**	**26.9677**	**34.4844**	**.5405**

Table 14: Quantitative comparison in Group I and II on LAION-COCO.

L Ensembling Models with Highly Correlated Features

As the number of the base models increases, reliance on multiple base models may leads to overfitting to correlated features. To evaluate the robustness of our AFA against highly correlated features, we design an experiment. Specifically, we selected one high-quality base model and one low-quality base model. Using AFA, we ensemble the high-quality model with several low-quality models to evaluate whether the ensembled model will perform toward that of the low-quality model due to the dominance of the correlated features from the low-quality model.

As shown in Table 15, although the performance the ensembled model slightly declines when ensembling with large number of low-quality models, it still outperforms the high-quality model. This demonstrates that our AFA exhibits strong robustness to highly correlated features.

	AES	PS	HPSv2	IR
High-Quality Base Model	5.5641	21.8013	28.0183	.4238
Low-Quality Base Model	5.5013	21.4624	27.7246	.2835
AFA (1 High-Quality Model + 1 Low-Quality Model)	5.6013	21.9341	28.3485	.4746
AFA (1 High-Quality Model + 2 Low-Quality Models)	5.6000	21.9348	28.3398	.4739
AFA (1 High-Quality Model + 3 Low-Quality Models)	5.5974	21.8248	28.3399	.4730
AFA (1 High-Quality Model + 5 Low-Quality Models)	5.5831	21.8019	28.2035	.4593

Table 15: Quantitative comparison of ensembling models with high-correlated features.

M Training with Dataset of Lower Quality

In this paper, we train our AFA using a high-quality dataset, JourneyDB. This raises the concern of whether AFA's high performance is primarily attributed to the quality of the training dataset. To address this, we also train AFA using LAION-COCO, a dataset of lower quality compared to JourneyDB.

As shown in Table 16, the model trained on LAION-COCO performs similarly to the one trained on JourneyDB, indicating that AFA is not sensitive to the quality of the training dataset. Its consistent performance across datasets of varying quality highlights its robustness and generalization ability. Regardless of whether the training dataset is high-quality or low-quality, AFA effectively leverages the features of the base models to deliver stable and reliable results.

	Group I								Group II							
	FID ↓	IS	CLIP-I	CLIP-T	AES	PS	HPSv2	IR	FID ↓	IS	CLIP-I	CLIP-T	AES	PS	HPSv2	IR
JourneyDB	9.76	7.14	.6926	.2675	5.5201	21.8263	27.9734	.4388	10.27	7.42	.6855	.2717	5.5798	21.8059	28.0371	.4892
LAION-COCO	9.74	7.16	.6930	.2664	5.5093	21.8257	27.9781	.4390	10.29	7.39	.6861	.2711	5.5789	21.8077	28.0363	.4889

Table 16: Quantitative comparison in Group I and II trained on JourneyDB and LAION-COCO.

N MORE EXPERIMENTAL DETAILS

We train our AFA in an environment equipped with 8 NVIDIA V100 GPUs, each with 32GB of memory. When ensembling two base models, the GPU memory consumption is about 11GB under FP16 precision and a batch size of 1. When ensembling three base models, the memory consumption increases to 14GB under the same settings. For six base models, the memory consumption reaches about 30GB. Due to the frozen parameters of the base models, training our AFA demands relatively modest GPU memory, even when ensembling a larger number of base models.