Accelerating Text-to-image Editing via Cache-enabled Sparse Diffusion Inference

Zihao Yu[1] **Haoyang Li**[1] **Fangcheng Fu**[1] **Xupeng Miao**[2] **Bin Cui**[1]
[1]Peking University [2]Carnegie Mellon University

Abstract

Due to the recent success of diffusion models, text-to-image generation is becoming increasingly popular and achieves a wide range of applications. Among them, text-to-image editing, or continuous text-to-image generation, attracts lots of attention and can potentially improve the quality of generated images. It's common to see that users may want to slightly edit the generated image by making minor modifications to their input textual descriptions for several rounds of diffusion inference. However, such an image editing process suffers from long-standing heuristics and low inference efficiency. This means that the extent of image editing is uncontrollable, and unnecessary editing invariably leads to extra computation. To solve this problem, we introduce Fast Image Semantically Edit (FISEdit), a cached-enabled sparse diffusion model inference method for efficient text-to-image editing. Extensive empirical results show that FISEdit can be $3.4\times$ and $4.4\times$ faster than existing methods on NVIDIA TITAN RTX and A100 GPUs respectively, and even generates more satisfactory images.

1 Introduction

The recent months have witnessed the astonishing improvement of diffusion models. Text-to-image generation is one of the most popular applications of diffusion models, which involves generating realistic images based on textual descriptions. While diffusion models have been shown to generate high-quality images with good coverage of the data distribution, they require substantial computational power to generate high-quality images. Thus, it is an emerging research area to make the generation process of diffusion models more effective and efficient.

Our target scenario. In real-world cases, after an image has been generated, it is common that the user may be unsatisfied with some specific regions of the generated image and desire to make subsequent **semantic edits**. Figure 1 demonstrates a concrete example. The user may wish to interactively edit the image by making minor modifications to the textual description. These modifications involve adding specifications (e.g., adding a "river" or the "fireworks") or replacing words/phrases (e.g., replacing the "river" with "snow mountain"). To be specific, in each round of editing, the user expects to make changes to only a small, specific region (affected region) and keep the rest (non-affected regions) untouched.

Challenges. However, we find that the current implementations fall short in supporting such a semantic editing system effectively and efficiently. First, due to the diversity nature of diffusion models, existing methods tend to make significant changes to the image, and therefore the process suffers from long-standing heuristics. Although there are applications that allow users to provide masks to control the in-painting regions, it would lead to poor user experience. Second, even though only a small region is expected to be modified, existing works typically follow the vanilla inference process of diffusion models, which generates the entire image from scratch, without taking the spatial property into account. Undoubtedly, this is a waste of time and computational resources since the generation of non-affected regions is pointless.

Machine Learning for Systems Workshop at 37th NeurIPS Conference, 2023, New Orleans.

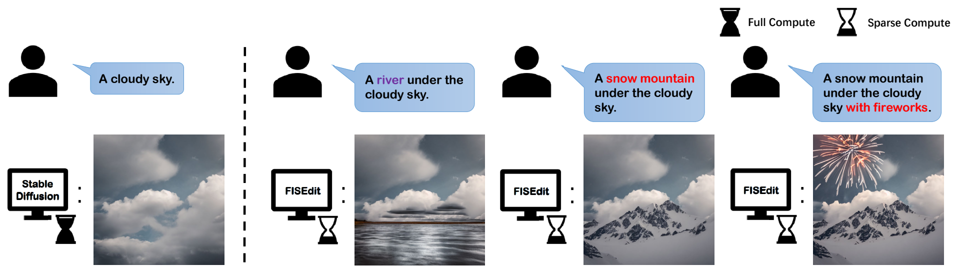
Full Compute
Sparse Compute
A cloudy sky.
A river under the cloudy sky.
A snow mountain under the cloudy sky.
A snow mountain under the cloudy sky with fireworks.
Stable Diffusion
FISEdit
FISEdit
FISEdit

Figure 1: An example of the user's interaction with FISEdit.

In essence, to support the semantic editing task effectively and efficiently, there involves two technical issues: (1) *how to automatically and accurately detect the affected region to be revised given the original image and the modified description* and (2) *how to focus on the target region during the generation while skipping the non-affected regions?*

Summary of contributions. In this paper, we develop a brand new ML-based method for the semantic editing system. The technical contributions of this work are summarized as follows.

- We introduce Fast Image Semantically Edit (FISEdit), a framework for minor image editing with semantic textual changes. FISEdit is a direct application to the inference process of pre-trained diffusion models and does not need to train new models from scratch. To support the interactive semantic editing task, FISEdit caches the feature maps of the latest generation process and addresses the two technical issues respectively.
- For the first issue, we present a fine-grained mask generation algorithm to capture the specific region for editing. To be specific, we quantify the distance between the cached and ad hoc feature maps given the modified textual description. Subsequently, we extract the region displaying substantial differences.
- Based on the detected region and the cached feature maps, we propose a sparse updating approach to minimize unnecessary computational burden.
- We thoroughly evaluate the performance of FISEdit with extensive experiments. Compared with existing text-based image editing works, FISEdit can generate high-quality images that are more consistent to the semantic edit. Furthermore, FISEdit outperforms existing works in terms of generation efficiency by $4.4\times$ on NVIDIA TITAN RTX and $3.4\times$ on NVIDIA A100.

2 Related works

As current image editing systems are computationally expensive and time-consuming, especially when dealing with high-resolution images, various methods have been explored to accelerate this process. Firstly, many recent works [1, 2, 3, 4, 5, 6, 7, 8, 9] have successfully reduced the number of iteration steps required for diffusion model inference while maintaining or improving the quality of the generated samples. All these research directions are orthogonal to ours and can be integrated with FISEdit.

Another way to accelerate image editing is reducing computation by exploiting the sparsity of the input data or the model parameters. There are several techniques [10, 11, 12, 13, 14, 15, 16, 17] available for implementing sparse computation. However, directly applying existing techniques such as tiled sparse convolution [13] fails to achieve considerable speedup and image quality in our scenario.

SIGE [18] is a general-purpose sparse inference framework for generative models, which integrates sparse convolution techniques into image editing tasks and makes sufficient use of data sparsity. However, SIGE [18] faces limitations when applied to Stable Diffusion [19] models due to its significant additional memory overhead requirement and inability to support textual changes. Therefore, we will not include SIGE as a baseline in the experimental part. In comparison to SIGE, our proposed framework supports more efficient sparse kernels and addresses its limitations, including batched input restrictions, intermediate activation management and fine-grained mask generation designed to detect areas affected by semantic changes.

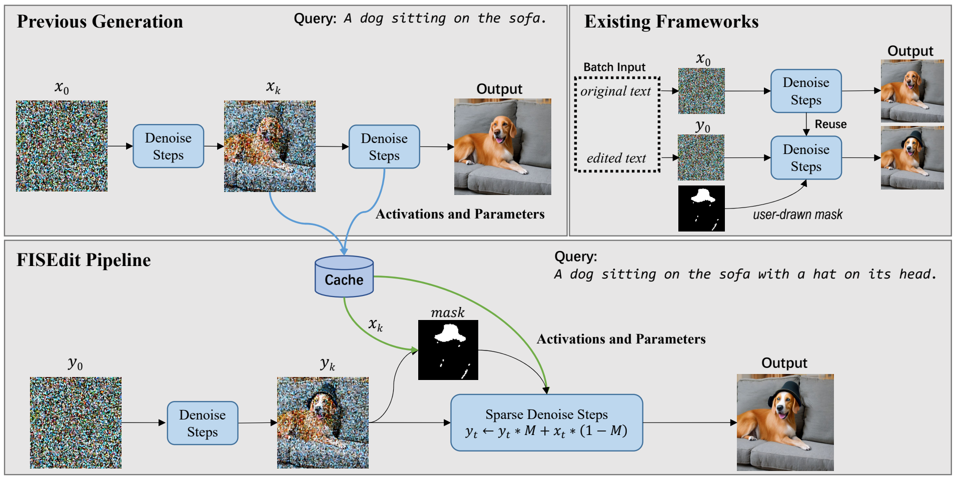

Figure 2: Overview structure of FISEdit. When a query arrives, our system firstly executes k denoise steps, and then generates difference mask according to the output activation of step k. In the remaining denoise steps, the pre-computed results are reused and the latents will be computed sparsely.

3 Method

Figure 2 shows the overview structure of FISEdit. Below, we first introduce a method to detect affected areas and generate difference mask for the subsequent sparse computation. Then we present our improved implementation of sparse computation algorithms.

3.1 Fine-grained mask generation

Difference masks are used in image editing tasks to index changed regions, but existing models have sub-optimal performance when the user cannot draw a precise mask. To address this, we propose a mask generation method, which automatically identifies the image areas that require substantial change according to the modification of textual description.

Target area capture. To detect the affected areas in feature maps, we observe the feature maps in each step of the Stable Diffusion [19] inference process and compare them using two similar textual prompts. Unfortunately, even though we have controlled the random factors, the changed regions remain large and unpredictable. Drawing inspiration from Prompt-to-Prompt [20], these irregular spatial changes can be controlled by enhancing cross-attention [21] module. Specifically, we cache and preserve part of the cross-attention map corresponding to the unchanged text prompts in the calls with new description, and then the changed regions can be structured and easily located through few denoise steps. As demonstrated in Figure 3, the difference caused by textual change can be observed clearly in step 5 to 10 and convert to a mask after finding the proper threshold ϵ via the following formula:

$$diff := Normalize\left(\sum_{t=t_1}^{t_2}|X_t - Y_t|\right),\ t_1, t_2 \in [1, 10]$$

$$A := \{i|diff_{pixel_i} < \epsilon\},\ B := \{j|diff_{pixel_j} \geq \epsilon\},\ N_1 := Card(A),\ N_2 := Card(B)$$

$$\arg\max_{\epsilon\in[0,1]} OTSU(\epsilon) := \frac{N_1 N_2}{(N_1 + N_2)^2}\left(\frac{1}{N_1}\sum_{i\in A} diff_{pixel_i} - \frac{1}{N_2}\sum_{j\in B} diff_{pixel_j}\right)^2$$

Consequently, we no longer need extra diffusion steps to calculate the mask as our previous work did (e.g., DIFFEdit [22]).

Sparsity analysis. We evaluate our method by generating mask for the human-written dataset used by InstructPix2Pix [23], where each sample in the dataset contains an original and manually

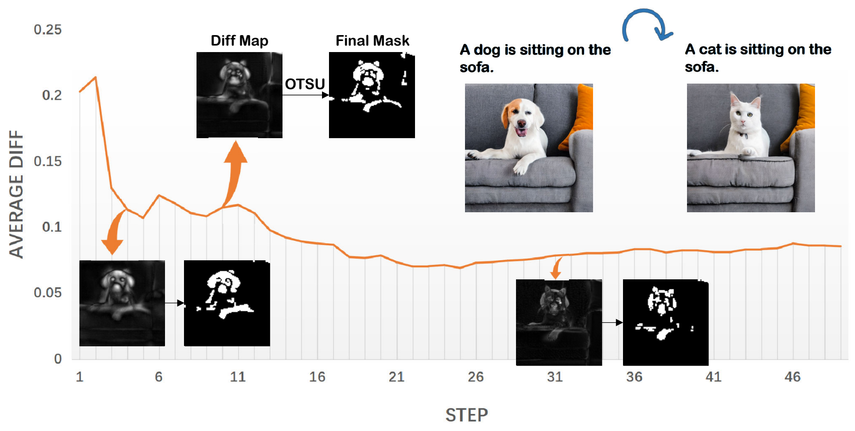

Figure 3: Variation of latent difference with iteration steps.

modified caption for an image. We find that the average edited size for these samples is less than 30%, reflecting an astonishing fact that most image regions can be reused and numerous calculations can be potentially saved. However, when the feature map is compressed to low-resolution, sparsity decreases correspondingly. Consequently, we adopt the sparse computation for high-resolution inputs only, and forgo those with low-resolution.

3.2 Sparse computation

We focus on the U-Net [24] modules since they are the most time-consuming and influential components in the image generation process. The U-Net architecture is composed of several attention and ResNet [25] modules. Based on the observation in the previous section, it is practicable to perform sparse calculations on a small proportion of the whole image in the above modules. We discuss our optimization on these modules in the appendix.

4 Discussion

Conclusion Existing image editing methods fail to exploit the similarity between previous and current diffusion model calls, wasting a large amount of computation. To sufficiently reuse prior results, we have solved the three challenges: difference detection, sparse computation and pre-computed data management. The experimental results indicate that our method can accurately edit the details of images while achieves a considerable speedup compared to vanilla image editing methods.

Limitations. Our method has a poor performance when editing a low-resolution images (e.g., 256×256). This is because, the image changes caused by semantic modification do not exhibit adequate sparsity in low-resolution images. However, GANs [26, 27] hold a competitive position in low-resolution image generation, and have less computational overhead compared to diffusion models, meaning that large image generation is the main challenge for real-world deployment.

Moreover, although we have optimized the memory overhead of image editing for diffusion model, about 1GB additional memory usage is still demanded in every iteration steps. But with FISEdit, this additional memory overhead can be on any of CPU, disk, or GPU, which is tolerable for model deployment in real-world use. In addition, as research on a few-step sampler for diffusion models keeps moving forward, we believe that FISEdit will have better performance in the future.

Real-world Applicability. With our method, it is possible to design a cache system for a real-world text-to-image service. Specifically, when a user query that contains a text is received, we can search for the most similar text-image pairs stored in the cache and process this query sparsely and incrementally, thereby achieving better system throughput.

References

[1] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *NeurIPS*, 2020. 2

[2] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *ICLR*, 2021. 2

[3] 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. *ArXiv preprint*, 2022. 2

[4] 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*, 2022. 2

[5] Daniel Watson, Jonathan Ho, Mohammad Norouzi, and William Chan. Learning to efficiently sample from diffusion probabilistic models. *ArXiv preprint*, 2021. 2

[6] Chenlin Meng, Ruiqi Gao, Diederik P Kingma, Stefano Ermon, Jonathan Ho, and Tim Salimans. On distillation of guided diffusion models. *ArXiv preprint*, 2022. 2

[7] Zhifeng Kong and Wei Ping. On fast sampling of diffusion probabilistic models. *ArXiv preprint*, 2021. 2

[8] Zhisheng Xiao, Karsten Kreis, and Arash Vahdat. Tackling the generative learning trilemma with denoising diffusion gans. In *ICLR*, 2022. 2

[9] Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. In *ICLR*, 2022. 2

[10] Patrick Judd, Alberto Delmas, Sayeh Sharify, and Andreas Moshovos. Cnvlutin2: Ineffectual-activation-and-weight-free deep neural network computing. *ArXiv preprint*, 2017. 2

[11] Xuanyi Dong, Junshi Huang, Yi Yang, and Shuicheng Yan. More is less: A more complicated network with less inference complexity. In *CVPR*, 2017. 2

[12] Xingyu Liu, Jeff Pool, Song Han, and William J. Dally. Efficient sparse-winograd convolutional neural networks. In *ICLR*, 2018. 2

[13] Mengye Ren, Andrei Pokrovsky, Bin Yang, and Raquel Urtasun. Sbnet: Sparse blocks network for fast inference. In *CVPR*, 2018. 2, 7

[14] Shaohuai Shi and Xiaowen Chu. Speeding up convolutional neural networks by exploiting the sparsity of rectifier units. *ArXiv preprint*, 2017. 2

[15] Liu Liu, Lei Deng, Xing Hu, Maohua Zhu, Guoqi Li, Yufei Ding, and Yuan Xie. Dynamic sparse graph for efficient deep learning. In *ICLR*, 2019. 2

[16] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. *ArXiv preprint*, 2019. 2

[17] Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. Token merging: Your vit but faster. *ICLR*, 2023. 2

[18] Muyang Li, Ji Lin, Chenlin Meng, Stefano Ermon, Song Han, and Jun-Yan Zhu. Efficient spatially sparse inference for conditional gans and diffusion models. *NeurIPS*, 2022. 2

[19] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, 2022. 2, 3

[20] Amir Hertz, Ron Mokady, Jay Tenenbaum, Kfir Aberman, Yael Pritch, and Daniel Cohen-Or. Prompt-to-prompt image editing with cross attention control. *ArXiv preprint*, 2022. 3, 8

[21] Hezheng Lin, Xing Cheng, Xiangyu Wu, and Dong Shen. Cat: Cross attention in vision transformer. In *2022 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE, 2022. 3, 7

[22] Guillaume Couairon, Jakob Verbeek, Holger Schwenk, and Matthieu Cord. Diffedit: Diffusion-based semantic image editing with mask guidance. *ICLR*, 2023. 3, 8

[23] Tim Brooks, Aleksander Holynski, and Alexei A Efros. Instructpix2pix: Learning to follow image editing instructions. *ArXiv preprint*, 2022. 3, 8

[24] 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*. Springer, 2015. 4

[25] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 4

[26] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *CVPR*, 2019. 4

[27] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 2020. 4

[28] Yuxin Wu and Kaiming He. Group normalization. In *ECCV*, 2018. 7

[29] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, JMLR Workshop and Conference Proceedings, 2015. 7

[30] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017. 7

[31] Chenlin Meng, Yutong He, Yang Song, Jiaming Song, Jiajun Wu, Jun-Yan Zhu, and Stefano Ermon. Sdedit: Guided image synthesis and editing with stochastic differential equations. In *ICLR*, 2022. 8

[32] Narek Tumanyan, Michal Geyer, Shai Bagon, and Tali Dekel. Plug-and-play diffusion features for text-driven image-to-image translation. In *CVPR*, 2023. 8

[33] Gaurav Parmar, Krishna Kumar Singh, Richard Zhang, Yijun Li, Jingwan Lu, and Jun-Yan Zhu. Zero-shot image-to-image translation. *ArXiv preprint*, 2023. 8

[34] Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, et al. Laion-5b: An open large-scale dataset for training next generation image-text models. *ArXiv preprint*, 2022. 8

[35] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *ICML*, 2021. 8

[36] Rinon Gal, Or Patashnik, Haggai Maron, Amit H Bermano, Gal Chechik, and Daniel Cohen-Or. Stylegan-nada: Clip-guided domain adaptation of image generators. *ACM Transactions on Graphics (TOG)*, 2022. 8

[37] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *NeurIPS*, 2017. 8

[38] Jack Hessel, Ari Holtzman, Maxwell Forbes, Ronan Le Bras, and Yejin Choi. CLIPScore: A reference-free evaluation metric for image captioning. In *EMNLP*, 2021. 8

A Sparse modules

A.1 Adaptive pixel-wise sparse convolution.

Tiled sparse convolution algorithm was proposed in SBNet [13] to handle sparse data within ResNet modules. However, as we mentioned in Section 3.1, sparsity will decrease when processing low-resolution feature maps, and this is particularly obvious in tiled sparse convolution layers. Therefore, directly applying existing technique fails to achieve considerable speedup and image quality in our scenario.

Given that the efficiency of this algorithm is largely contingent on the distribution of difference masks and the size of split blocks, the constant block-gather strategy used in SBNet doesn't seem plausible. Therefore, it is essential to select a compatible block size with the given mask. We introduce Adaptive Pixel-wise Sparse Convolution (APSC) to ensure tiled sparse convolution achieves desired effect when dealing with capricious mask. We denote (h, w), $N^m_{h,w}$ as the block's spatial size and the number of blocks gathers using given mask m respectively, and assume the kernel size is (h_k, w_k). We search for the optimal division strategy heuristically via the following approximation:

$$\underset{h,w\in\{2,4,8,16,32\}}{\arg\min} \quad f(h,w) := (h - h_k + 1) * (w - w_k + 1) * N^m_{h,w}$$

All the variables mentioned above can be determined during the preprocessing stage, ensuring that APSC does not introduce any additional computational costs to model inference.

A.2 Approximate normalization.

Since some normalization layers such as group normalization [28] entail calculating the mean and variance of the feature map in spatial dimension, it is infeasible to use sparse feature map in these layers. Inspired by batch normalization [29], which uses the mean and variance during training for model inference, it is plausible to use approximate methods in normalization layer. With the assumption that a tiny textual modification will not affect the distribution of the whole image, we are able to cache the mean and variance computed in the previous image generation and directly reuse them to facilitate the sparse inference process. Therefore, without the reliance on the statistics of the input feature map, sparse computation is available in all normalization layers.

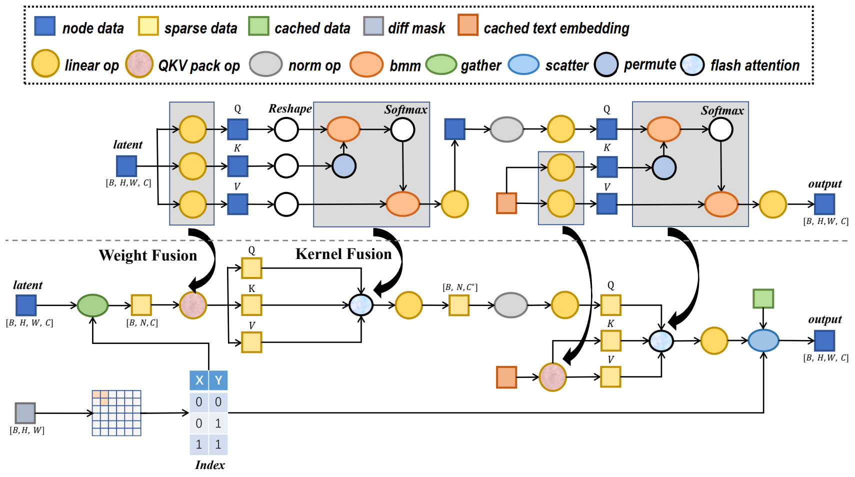

Figure A1: Overall improvement of our attention module.

A.3 Approximate attention.

Recent diffusion models have incorporated self-attention [30] and cross-attention [21] modules to enhance the image quality and the correlation between image and text in text-to-image tasks, resulting in computational bottlenecks. Readers might suspect whether it is possible to utilize the sparsity of edited areas in attention layers, since the whole feature map is required to be used in self-attention layers in order to learn the long-term relevance among pixels. Intuitively, if we only calculate the results using non-zero regions in difference mask, the output will be inevitably deviated from the accurate result. Fortunately, such precision errors can be controlled to a tolerable range through some accuracy compensation measures. Specifically, we only apply sparse computation in the self-attention layers with a high-resolution input feature map. In these layers, non-zero regions account for only a small proportion of the entire feature map, and their influence on the unaffected area is consequently negligible, particularly after the SoftMax layer.

Figure A1 shows the overview of our implementation of attention modules. Masked regions are collected before being input into the subsequent self-attention and cross-attention layers, utilizing sparse data exclusively within the attention layers to avoid unnecessary data movement.

B Experiment

B.1 Experimental design

We implement our system based on the HuggingFace's diffusers[1], which is a generic framework for training and inference of diffusion models. We clone this project and integrate it with our self-developed sparse inference engine[2]. The source code has been made publicly available at https://github.com/Hankpipi/diffusers-hetu.

Baselines. We compare our method with the following baselines, and all the implementations for them can be found in HuggingFace's diffusers. All the baselines and our method use the pre-trained weights and configurations of the Stable Diffusion v2 model[3] (e.g., the output image size is 768×768), and more details about our evaluation configurations can be found in our supplementarial materials.

- *Vanilla stable diffusion text-to-image pipeline (SDTP).* A simple attempt to edit image is to fix the internal randomness (e.g., the initial latent) and regenerate using the edited text prompt.
- *Other image editing systems*. Stable diffusion in-painting pipeline (SDIP), Prompt-to-Prompt [20], SDEdit [31], InstructPix2Pix [23], DIFFEdit [22], PPAP [32] and Pix2Pix-Zero [33].

Datasets & Metrics. Following prior works [23, 31], we select LAION-Aesthetics [34] as our evaluation dataset. We use Image-Image similarity [35], Directional CLIP similarity [36], FID [37] and CLIP Accuracy [38] to evaluate the quality of image editing.

B.2 Main results

Image quality. We show the quantitative results in Figure A2 and the qualitative results in Figure A3. We find that image-image similarity is competing with CLIP accuracy and directional CLIP similarity, which means that increasing the degree to which the output images correspond to a desired text will reduce their consistency with the input image. In order to show the tradeoff between two metrics, we have varied representative parameters for the methods. It can be inferred from these metrics that our method can better locate and modify changing areas as well as reflect the meaning of the text.

Model efficiency. We present efficiency comparison results in Table A1. We observe that all baselines struggle to generate or edit images efficiently since they cannot leverage the sparsity to avoid unnecessary computations. Worse, InstructPix2Pix [23] even encounter both computation and memory bottlenecks. In contrast, our method can accurately edit the details of image, and also reduces the computation of diffusion models by up to $4.9\times$ when the image size is 768×768 and the edit size is 5%. Eventually, we accelerate text-to-image inference by up to $4.4\times$ on NVIDIA TITAN RTX and $3.4\times$ on NVIDIA A100 when the edit size is 5%.

[1]https://github.com/huggingface/diffusers
[2]https://github.com/Hankpipi/Hetu/tree/diffusers
[3]https://huggingface.co/stabilityai/stable-diffusion-2

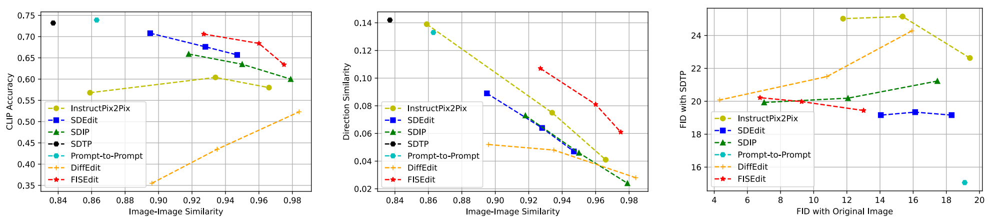

Figure A2: Quantitative results of the images generated by baselines and our method, demonstrating the trade-off between consistency with the input image and consistency with the semantic change. We vary InstructPix2Pix's image guidance scale between $[1.0, 2.5]$, SDEdit's strength between $[0.5, 0.75]$, DIFFEdit's strength between $[0.5, 1.0]$, and edited size between $[0.25, 0.75]$ for SDIP and our method.

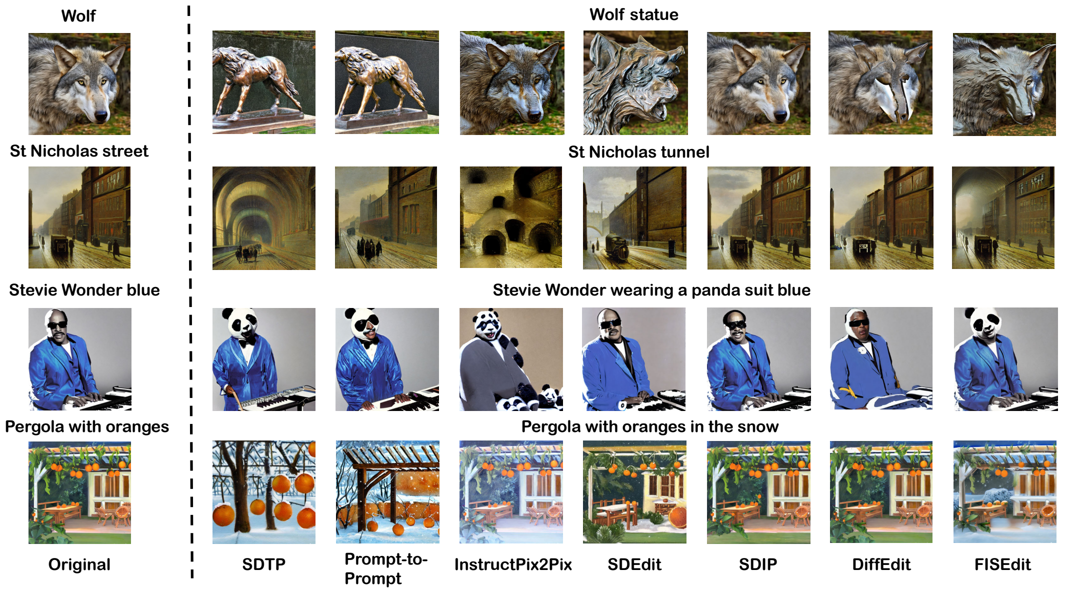

Figure A3: Qualitative results of the images generated by baselines and our method.

Method	Property		Efficiency						
	No Extra Training	Partially Edit	Edit Size	MACs		TITAN RTX		A100 PCle 40GB	
				Value	Ratio	Value	Ratio	Value	Ratio
SDTP	✓	×							
SDIP	×	✓	-	1901G	1.0×	1.83it/s	1.0×	3.86it/s	1.0×
Prompt-to-Prompt	✓	×							
pix2pix-zero	✓	×	-	-	-	OOM	-	OOM	-
PPAP	✓	×							
InstructPix2Pix	×	×	-	4165G	0.5×	OOM	-	2.29it/s	0.6×
SDEdit	✓	×	-	1521G	1.2×	2.28it/s	1.2×	4.82it/s	1.2×
DIFFEdit	✓	✓	-	3041G	0.6×	1.15it/s	0.6×	2.42it/s	0.6×
FISEdit	✓	✓	30%	835G	2.3×	4.06it/s	2.2×	9.57it/s	2.5×
			15%	485G	3.9×	5.64it/s	3.1×	11.6it/s	3.0×
			5%	**386G**	**4.9**×	**8.14it/s**	**4.4**×	**13.1it/s**	**3.4**×

Table A1: Property and efficiency evaluation of each method. We use the Multiply-Accumulate Operations (MACs) of U-Net to measure computational cost and the number of U-Net calls completed per second as a measure of speed, since U-Net calls are the computational bottleneck of the above models. Due to the fact that the inference speed of the baseline models will not be affected by the size of modifications, we do not discuss edit size for them.