KYBERNETIKA — VOLUME *60* (2024), NUMBER 6, PAGES 819–833

# DDIMCACHE: AN ENHANCED TEXT-TO-IMAGE DIFFUSION MODEL ON MOBILE DEVICES

Wu Qifeng

On June 11, 2024, OpenAI announced a collaboration with Apple to deeply integrate the ChatGPT generative language model into Apple's product lineup. With support from various generative AI models, devices like smartphones will become more intelligent. The text-to-image diffusion model, known for its stable and superior generative capabilities, has gained wide recognition in image generation and will undoubtedly play a crucial role on mobile devices. However, the large size and complex architecture of diffusion models result in high computational costs and slower execution speeds. As a result, diffusion models require high-end GPUs or cloud-based inference, which often raises personal privacy and data security. This paper presents a multiplicative effect joint optimization method for complex models such as diffusion models, enabling efficient execution on mobile devices. The method integrates multiple optimization strategies, leveraging their interactions to create synergies and enhance overall performance. Building on this multiplicative effect joint optimization approach, we have introduced DDIMCache, an enhanced text-to-image diffusion model. DDIMCache maintains image generation quality while achieving optimal speed, generating 512–512 images in approximately 6 seconds. This provides powerful image generation capabilities and an enhanced user experience for mobile users.In addition, as a foundation model, Stable Diffusion supports more applications such as image editing, inpainting, style transfer, and super-resolution, all of which can have a significant impact. The ability to run the model entirely on mobile devices without an internet connection will open up endless possibilities.

*Keywords:* diffusion model, text-to-image, mobile devices

*Classification:* 68T01

## 1. INTRODUCTION

2024 is poised to become a breakthrough year for mobile AI. The global market share of AI-powered smartphones is expected to reach 16 %. Fueled by consumer demand for AI assistants and enhanced mobile AI features, this share is anticipated to surge to 54% by 2028. The strategic partnership between Apple and OpenAI could reshape consumer expectations for mobile AI applications, accelerating the shift toward more personalized, efficient, and diverse AI solutions. In the long term, Apple may develop a fully integrated AI ecosystem, spanning from hardware to applications, centered around

DOI: 10.14736/kyb-2024-6-0819

mobile devices. Apple has stated that its "Apple Intelligence" will decide whether tasks are processed on-device or in the cloud. Simple tasks will be handled on the device, while more computationally demanding ones will utilize cloud-based processing. Apple's AI capabilities can recognize data without collecting it. To ensure users that their data remains local and isn't exploited, Apple has developed a private cloud server with chip-level privacy and security protections.

The text-to-image diffusion model, renowned for its stability and generative capabilities, is poised to become an essential feature on mobile devices. Generative language models, such as ChatGPT, demand considerable computational resources. Diffusion-based text
-to-image generation models [1] generally surpass generative language models in both computational scale and architectural complexity. Consequently, large-scale cloud inference platforms and high-end GPUs are needed to deliver the desired user experience. This approach not only incurs high costs but also raises potential privacy concerns. The challenge of accelerating model inference on mobile devices has garnered significant attention. Deploying diffusion models on mobile devices can drastically cut computational costs, reduce latency, enhance user privacy, and improve scalability. However, limited computing and memory resources on mobile devices pose significant challenges for deploying and running diffusion models. To tackle these challenges, we have introduced DDIMCache, an enhanced text-to-image diffusion model. The key contributions of this paper include:

1. This paper performed an in-depth analysis of the UNet used in the denoising process, identifying architectural redundancies. Based on this, we developed a feature-caching optimization method for diffusion model inference. This method greatly enhances image generation speed with only a slight reduction in quality.

2. This paper presents a multiplicative effect Joint optimization algorithm for complex models such as diffusion models, enabling efficient execution on mobile devices. The method integrates multiple optimization strategies, leveraging their interactions to create synergies and enhance overall performance.

3. This paper propose DDIMCache, a text-to-image diffusion model designed specifically for mobile devices. DDIMCache achieves optimal speed without compromising image quality, generating 512 × 512 images in around 6 seconds.

## 2. RELATED WORK

In their article "World's First On-Device Demonstration of Stable Diffusion on an Android Phone" [2] (Hou and Asghar, 2023), the authors discuss Qualcomm's breakthrough in running the Stable Diffusion model directly on an Android smartphone. In their 2023 study [3], Chen et al. implemented several optimizations that reduced Stable Diffusion' inference latency to under 12 seconds on mobile.

Recent research on accelerating model inference on mobile devices has mainly concentrated on fast sampling methods, GPU optimization [3], training improvements [4], quantization techniques [5], model pruning [6], and architecture search [7]. Due to the severely limited computational and memory resources on mobile devices, even though fast sampling methods theoretically offer the greatest potential for acceleration, the resulting performance improvements are often insufficient to ensure a good user experience.In some cases, they may even degrade image generation quality. Therefore, standard optimization algorithms alone are insufficient to achieve both optimal generation speed and maintain image quality on mobile devices with hardware and performance limitations.

To tackle these challenges, this paper presents a joint optimization method for complex models such as diffusion models, enabling efficient execution on mobile devices. The method integrates multiple optimization strategies, leveraging their interactions to create synergies and enhance overall performance.

## 3. APPROACH

### 3.1. Multiplicative effect joint optimization algorithm

When exploring the deployment of neural network models on mobile devices, several key challenges and optimization directions are involved. Due to the limited computational resources and power constraints of mobile devices, efficiently deploying complex neural network models on these devices requires the application of various optimization techniques in combination. Here are some key exploration directions:

1. Quantization techniques reduce memory usage and computational load by converting floating-point weights and activations in a model to lower 16 widths. Although quantization may slightly reduce model accuracy, it significantly lowers computational costs and accelerates inference on mobile devices.

   Pruning techniques reduce model size by removing redundant or less impactful parameters. Common pruning methods include weight pruning, which removes neurons with smaller weights, and structural pruning, which eliminates entire convolutional filters or neuron layers that are less important. Pruning can significantly reduce the number of parameters in the model, thereby lowering memory requirements and computational complexity.

2. Researchers have specifically designed several lightweight neural network architectures for mobile devices, such as TensorFlow Lite, PyTorch Mobile, and Core ML. These architectures adapt to the computing capabilities of mobile devices by reducing parameters and computation.

3. Modern mobile devices are equipped with dedicated AI hardware accelerators, such as NPUs (Neural Processing Units) and DSPs (Digital Signal Processors). These accelerators efficiently handle neural network computations, greatly increasing the speed and efficiency of model inference. GPU Acceleration: Although the GPU performance of mobile devices is limited compared to desktop-level GPUs, using them for parallel computation can still improve the speed of neural network execution.

4. Improving neural network model architectures is a key area of deep learning research, particularly with regard to addressing architectural redundancy in different

application scenarios, which presents a significant optimization opportunity. Architectural redundancy refers to unnecessary layers, nodes, or parameters in the model design that can lead to wasted computing resources, slower inference speed, or excessive memory usage. By optimizing these redundant components, model efficiency can be significantly improved while maintaining or even enhancing performance.

5. By analyzing the inference process, methods such as reducing inference steps or caching intermediate computation results can be used to minimize redundant calculations.

Through the analysis of the deployment and operation process of the neural network model, the multiplicative effect joint optimization algorithm combines the aforementioned multiple optimization techniques, utilizing the strengths of each to complement and share information, achieving a multiplicative effect that surpasses the abilities of any individual method.

We analyze the deployment and execution of the diffusion model on mobile devices.

1. Mobile devices have limited computational power and memory, making it impossible to load all components of the text-to-image diffusion model. In most scenarios, the loss from 8-bit quantization is minimal, resulting in almost no significant difference in image quality while substantially reducing model size.Pruning typically involves a trade-off with model performance, necessitating careful consideration.Both pruning and quantization are essential for the complete operation of diffusion models on mobile devices.

2. Existing deep learning frameworks have been optimized specifically for certain mobile devices. For example, iOS devices are well-supported by Core ML, while Android devices typically support TensorFlow Lite or PyTorch Mobile. A universal deep learning framework for mobile devices has not yet emerged, which is why we do not use specialized learning frameworks.

3. Although AI accelerators and GPU acceleration on mobile devices have played an important role in promoting the efficient operation of deep learning models, their designs are often tailored to specific devices and hardware architectures, resulting in a lack of universal solutions. Therefore, we have also decided to abandon this approach.

4. The application of the UNET architecture in diffusion models does indeed face redundancy issues, especially in image generation tasks. This is because UNET was originally designed for image segmentation, and the requirements for image generation differ from those for image segmentation. Therefore, in image generation applications, the design of the UNET architecture contains some redundancies and offers significant optimization potential.

5. The inference performance of diffusion models is mainly constrained by the number of denoising steps and the computational complexity of each step. We can consider using common fast sampling methods to reduce the number of inference steps.

## 3.2. Pruning and quantization

When loaded into memory, Stable Diffusion 1.5 typically occupies around 10GB of RAM. The limited memory on mobile devices makes it impractical to load all components of Stable Diffusion at once. All subsequent experiments will default to using quantization and pruning.

Due to the significant impact of pruning on diffusion models, which often have complex architectures, preserving details and generating high-quality images is crucial. We adopt a structural pruning approach, removing entire convolutional layers. After pruning, the model is fine-tuned using a small-scale training dataset.

We then applied quantization to the pruned Stable Diffusion 1.5, quantizing the weights to 8-bit precision to reduce memory consumption. Since mobile GPUs lack support for integer matrix multiplications, we converted them to 16-bit floating points before computation.

Finally, when fully loaded into memory, the model occupies approximately 1.8GB, which most mobile hardware can support.

## 3.3. DDIM

The reverse diffusion process [8] in the text-to-image diffusion model denoises random noise $X_t \sim N(0, I)$ to the target distribution by modeling the transition probabilities $q(X_{t-1}|X_t)$. At each step $t$ of the reverse diffusion process, the conditional probability distribution is approximated by the network $\epsilon_\theta(X_t, t)$, and the training is formalized as a noise prediction problem . Therefore,

$$X_{t-1} \sim p_\theta(X_{t-1}|X_t) = \mathcal{N}\left(x_{t-1}; \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(\mathbf{x}_t, t)\right), \beta_t\mathbf{I}\right), \tag{1}$$

where $\alpha_t = 1 - \beta_t$, and $\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$. After multiple iterations, $X_0$ is obtained.

Diffusion models require a thousands of denoising steps. By employing fast sampling algorithms like DDIM (Denoising Diffusion Implicit Models) [9] , the number of sampling steps can be reduced without significantly degrading the quality of the generated output. The method uses deterministic sampling to accelerate inference speed by dozens of times while maintaining the quality of the generated images. Theoretically, it is the most powerful optimization method for diffusion models. In the following experiments, the implementation will be specifically done using Python. Hugging Face's [10] diffusers library provides a relatively simple interface for using Stable Diffusion 1.5. You can generate images by specifying the DDIM sampler.

## 3.4. Cache strategy

In this work, we continue to optimize each inference step of the diffusion model. The network $\epsilon_\theta$, employed in the reverse diffusion process, is typically constructed using a UNet [11] architecture. However, the redundancy of the UNet design has become the main bottleneck for conditional diffusion models. As shown in Figure 1, UNet is a convolutional neural network comprising an encoder, a decoder, and skip connections. The

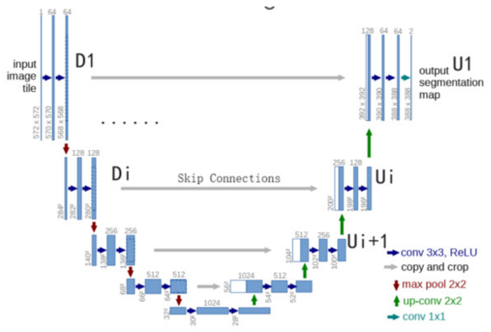


**Fig. 1.** This is UNet Architecture image.

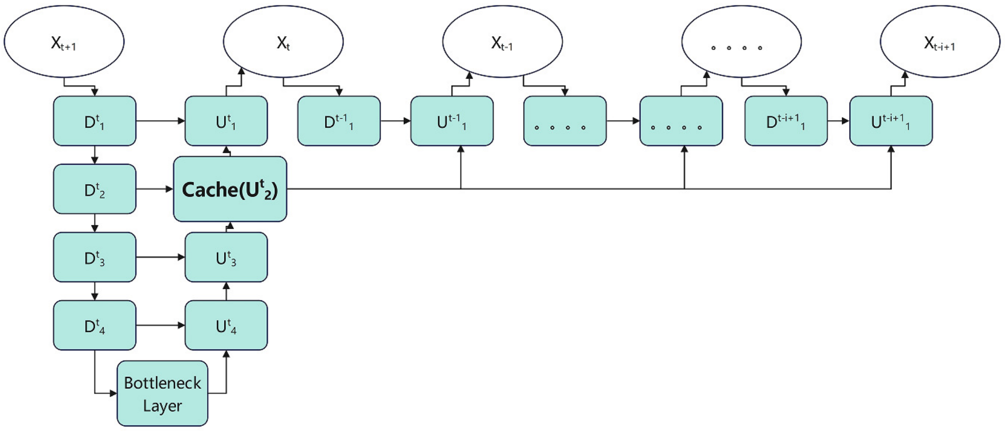


**Fig. 2.** Cache Strategy in the Inference.

encoder down-samples the input image to various sizes, encoding high-level features, while skip connections transmit feature information from the encoder to corresponding decoder layers for feature fusion. This design ensures that high-frequency features, which may be lost during down-sampling, are reintroduced during the decoder's reconstruction of the high-resolution image. The decoder then up-samples the feature maps to reconstruct the final output image.

In each layer of the UNet, high-level features from the previous layer $U_{i+1}$ are fused with high-frequency features from the corresponding down-sampling block $D_i$ via skip connections. This fusion is represented by the operation $D_i \oplus U_{i+1}$ , where$\oplus$denotes feature concatenation. Here, $U$ represents up-sampling blocks, $D$ represents down-sampling blocks, $D$i denotes the $i$th layer down-sampling block, and $U_{i+1}$ denotes the $(i + 1)$th layer up-sampling block. After feature fusion in the current layer, the result is passed to the next layer for further fusion at a larger scale. $U_{i+1}$ represents high-level features, including semantic and structural information about the image. These high-level features encode slowly changing information, such as the shape, texture, position, and category of objects. On the other hand, $D_i$ provides high-frequency features, including fine edges, texture details, and noise, primarily derived from the down-sampling blocks.

The training process of diffusion models is computationally intensive, requiring forward propagation, backward propagation, and weight updates for each batch of samples. However, the inference process is less demanding, requiring only forward propagation to predict the image without backward propagation or weight updates. The inference process primarily relies on UNet for noise prediction. Leveraging redundant features of the UNet architecture can effectively reduce the computational load during inference.

Below, we introduce the process described in Figure 2. At each denoising step, the network $\epsilon_\theta$ predicts noise based on the parameters $X_t$ and $t$. Removing the predicted noise from $X_t$ results in $X_{t-1}$. During the denoising process, changes in the image between adjacent steps are minimal.

The final stage of noise prediction specifically involves feature fusion in the first layer of the UNet, represented as $D_1 \oplus U_2$. The predicted noise is related only to $D_1$, $U_2$, and the time step $t$. $U_2$ contains high-level features of the generated image, primarily low-frequency information such as high-level semantics and structure, which are unrelated to high-frequency noise. Because $U_2$ changes very little between consecutive steps in the denoising process, recomputing these high-level features at each step results in substantial redundant computation. Therefore, $U_2$ can be cached at each step to accelerate the inference process. On the other hand, $D_1$, which contains high-frequency information like noise, changes significantly between consecutive steps and cannot be cached.

At step $t$ in the denoising process, after processing by the encoder and multiple layers of feature fusion in the UNet, high-level features $U_{t2}$ are obtained. $U_{t2}$ represents the high-level features of the second-layer up-sampling block at step $t$. Since $U_{t2}$ changes little over the subsequent $n$ steps, it can be cached in a cache. The result of feature fusion ($D_1 \oplus U_2$) in the first layer of the UNet is then further processed to obtain $X_t$. In the subsequent denoising step $t-i$ ($i = 1, 2, 3, \ldots, n-1$), the computation process for $U_{t+i,2}$ can be omitted, and the cached $U_{t2}$ value reused for feature fusion in the first layer. This feature caching mechanism can be extended to all steps in the reverse denoising process. The cached features are computed once and reused over the subsequent $n-1$ steps. The process of caching high-level features $U_2$ continues in the next step. Subsequent experiments will analyze the appropriate value for $n$.

## 3.5. DDIMCache

We introduced DDIMCache, an enhanced text-to-image diffusion model, by combining optimizations from quantization,pruning, DDIM, and caching strategies. The following experiments will demonstrate that these optimization algorithms produce a multiplicative effect, significantly reducing computational load and greatly improving efficiency.

## 4. EXPERIMENT

### 4.1. Experimental setup

DDIMCache is built on the foundation of Hugging Face's Stable Diffusion 1.5 [10]. To ensure optimal performance on mobile devices, we performed pruning and quantization optimizations in advance.We selected the widely used MS-COCO 2017 [12] dataset for a comprehensive performance evaluation.We used standard metrics, including latency and FID, for our evaluation. All experiments were conducted on the recently released high-performance Samsung S23 Ultra, which features a Snapdragon 8 Gen 2 processor, Adreno 740 GPU, and 8GB of RAM.

### 4.2. Quantitative analysis

| FID / Latency | Cache $n = 1$ | Cache $n = 2$ | Cache $n = 3$ | Cache $n = 5$ | Cache $n = 6$ | Cache $n = 10$ |
|---|---|---|---|---|---|---|
| DDIM Step=20 | 29.1 / 9.8 | 29.4 / 5.3 | 29.8 / 3.8 | 30.8 / 2.4 | 31.5 / 2.2 | 33.9 / 1.7 |
| DDIM Step=50 | 18.3 / 24.4 | 18.5 / 13 | 18.8 / 9.3 | 19.8 / 6.2 | 20.6 / 5.4 | 24.2 / 4.1 |
| DDIM Step=100 | 18 / 49.4 | 18.2 / 27.3 | 18.4 / 19.4 | 19.5 / 12.1 | 20.1 / 10.8 | 23.7 / 8.3 |
| DDIM Step=1000 | 17.4 / 487.3 | 17.6 / 271.0 | 17.9 / 191.0 | 18.9 / 123.1 | 19.8 / 109.4 | 23.1 / 84.2 |

**Tab. 1.** Benchmark results for the GALAXY S23 ULTRA with each optimization enabled, using MS-COCO 2017 to generate images with a resolution of 512–512. Due to memory constraints, the DDIM and Cache optimizations executed in Stable Diffusion 1.5 have undergone quantization and pruning.

The table 1 presents the performance metrics of the base model Stable Diffusion 1.5 after applying quantization and pruning, across various cache Step and DDIM sampling Step. The metrics include FID scores, which assess image generation quality, and latency, which reflects the image generation speed in seconds. The first row represents the baseline model with varying cache step sizes applied. The cache step size, denoted as n, takes values of 1, 2, 3, 5, 6, and 10. When $n = 1$, it is equivalent to not using the caching strategy. The first column represents the baseline model with varying sampling step sizes applied.The sampling step size, denoted as step, takes values of 20, 50, 100, and 1000. When step = 1000, it is equivalent to not using the Fast Sampling Method. When $n = 1$ and step=1000, it is equivalent to executing the original base model with only quantization and pruning applied. The values below the diagonal line in each cell represent the latency for image generation by the model under the corresponding cache step size n and sampling step size step, with the latency measured in seconds. For the quality assessment of images generated by diffusion models, the FID metric is one of the very commonly used evaluation indicators. Regarding the reasonable range of FID values [13], for image generation tasks, an FID score between 5 and 20 can be considered a very good result. An FID score above 20 may require further model improvement. These values, of course, can vary depending on the specific dataset and task, but they can generally serve as a reference range.

Speedup and efficiency are two metrics used to measure the performance improvement of optimized algorithms. Speedup measures the degree of performance enhancement achieved by the optimized algorithm. It represents the factor by which the execution time of the algorithm improves after optimization compared to the execution time before optimization. The formula for calculating speedup is:

$$S(n) = \frac{T_s}{T_p(n)} \tag{2}$$

Where: $S(n)$ is the speedup under parameter $n$, $T_{\mathrm{s}}$ is the execution time before the algorithm is optimized, and $T_p(n)$is the execution time of the optimized algorithm under parameter n. The efficiency of an algorithm is a measure of how effectively the algorithm utilizes resources (n) after applying an optimization technique. It indicates how efficiently the algorithm uses resources (n) in the process of improving performance. The formula for calculating efficiency is still based on speedup:

$$E(n) = \frac{S(n)}{n} \tag{3}$$

Where:$S(n)$ is the speedup under parameter n, $E(n)$represents the efficiency under parameter n, and n denotes the amount of resources used.

### 4.2.1. Quantization and pruning

Without quantization and pruning, mobile devices' memory limitations prevent the full execution of Stable Diffusion. This paper focuses on model performance on mobile devices, ablation experiments for these methods will not be conducted. As seen in the Table 1. When $n = 1$ and step=1000, it is equivalent to executing the original base model with only quantization and pruning applied.In this case, the model's image generation latency is 487 seconds, with an FID of 17.4. The model's latency is significantly higher than typical values, and the FID is also larger than usual. This is due to the significantly lower computational performance of mobile platforms compared to desktop platforms. Regarding differences in image generation quality, models on mobile devices often need to be quantized and pruned. It may degrade image generation quality.

### 4.2.2. DDIM

As seen in the second column of Table 1, where the cache $n = 1$, indicating that no caching strategy is applied, and only the DDIM is used. As the sampling step size decreases from 1000 to 50, the latency significantly decreases from 487 seconds to 24.4 seconds. Meanwhile, the FID score gradually rises from 17.4 to 18.3, indicating a slight reduction in image generation quality. When the sampling step size is further reduced to 20, the latency continues to decrease to 9.8 seconds, but the FID score increases sharply to 29.1. As expected, DDIM is feasible on mobile platforms.DDIM offers an effective method to reduce the computational burden of diffusion models while maintaining quality image generation.When Step=50, the FID is 18.3, which falls within an acceptable range. However, the latency is 24.4 seconds, far exceeding the image generation delay for a good user experience.Next, we will separately evaluate the speedup and efficiency for different sampling Step of DDIM.

| | DDIM Step=100 | DDIM Step=50 | DDIM Step=20 |
|---|---|---|---|
| factor =1000/Step | DDIM factor=10 | DDIM factor=20 | DDIM factor=50 |
| S(factor) | 9.9 | 19.9 | 49.7 |
| E(factor) | 0.99 | 0.99 | 0.99 |

**Tab. 2.** Performance evaluation of DDIM.

Table 2 can be calculated from Table 1 using Equations 2 and 3. In DDIM, as the Step increases, the algorithm's performance decreases. To more clearly express the performance variation in DDIM, we introduce a parameter equivalent to Step: factor, where factor = 1000/Step. Here, 1000 represents the total number of sampling steps, so the factor indicates the reduction factor relative to the total Steps of DDIM.

From the observation of Table 2, it can be concluded that as the sampling Step decreases and the step reduction factor increases, the algorithm's speedup improves significantly, while the efficiency remains stable. This indicates that the Fast Sampling Method are able to maintain high efficiency while enhancing computational speed.

#### 4.2.3. Cache strategy

The fifth row of Table 1 shows that when the sampling step size is set to 1000, the fast sampling method is not applied, and only the cache strategy is used. The caching method efficiently reduces the computational burden of diffusion models.As the caching step increases (from $n = 1$ to $n = 10$), the image generation latency significantly decreases from 487 seconds to 84 seconds. The FID gradually increases from 17.4 to 23.1, indicating a slight decline in image generation quality. When $n = 6$, the FID is 19.8, which remains within an acceptable range. However, the latency is 109 seconds, far exceeding the image generation delay for a good user experience. Next, we will separately evaluate the speedup and efficiency for different n of the Cache Strategy.

| | Cache ($n = 2$) | Cache ($n = 3$) | Cache ($n = 5$) | Cache ($n = 6$) | Cache ($n = 10$) |
|---|---|---|---|---|---|
| S(n) | 1.80 | 2.55 | 3.96 | 4.47 | 5.8 |
| E(n) | 0.9 | 0.85 | 0.79 | 0.75 | 0.58 |

**Tab. 3.** Performance evaluation of Cache strategy.

From Table 1, Table 3 can be calculated using Equations 2 and 3. As observed in Table 3, the speedup of the cache strategy increases significantly as the n grows. However, the acceleration rate diminishes with larger values of n, indicating that while a larger cache n improves computational speed, its effect on acceleration lessens progressively.

#### 4.2.4. Multiplicative-effect optimization algorithm

The remainder of Table 1 presents the image generation performance after applying cache strategy and fast sampling method to the baseline model.As the sampling step size decreases from 100 to 20 and the cache n increases from 2 to 10, latency significantly reduces from 271 seconds to 1.7 seconds. Meanwhile, the FID score gradually increases from 17.6 to 33.9, indicating a decrease in image generation quality.

We compare the performance of the DDIM with the Cache strategy. Comparing Table 2 and Table 3, DDIM shows greater speed gains and maintains stable efficiency across various settings. The Cache strategy achieves higher efficiency with smaller values of n, but as n increases, although speed improves, efficiency decreases. The DDIM outperforms the Cache strategy in terms of both performance and stability. Neither optimization method alone is sufficient to achieve the low latency required for a satisfactory user experience in image generation, while maintaining acceptable image quality. However, by an multiplicative-effect optimization algorithm, this goal can be achieved.

We propose DDIMCache, a text-to-image diffusion model designed specifically for mobile devices. As shown in Table 1, with cache step size of $n = 5$ and sampling step size of Step=50, the optimized implementation achieves 512–512 images generation latency of 6.2 seconds and an FID score of 19.8. In hardware and performance-constrained mobile environments, the image generation latency falls within the range of a satisfactory user experience, and the image quality is acceptable.

### 4.3. DDIMCache

Table 4 compares the Stable Diffusion 1.5 with the DDIMCache proposed in this paper for generating 512x512 images across four different scenarios. The first row displays images generated by Stable Diffusion 1.5 on a desktop computer with an RTX 3090 GPU, while the second row shows images generated by the DDIMCache on a Samsung S23 Ultra using the same prompt.

The results indicate that the images generated by DDIMCache are highly similar to those of Stable Diffusion 1.5, with some differences. However, the images generated by DDIMCache still align with the text prompts provided, maintaining consistency and relevance between the image content and descriptions. The model can still understand and accurately generate images based on the text prompts. The performance gap is attributed to DDIMCache's use of quantization, pruning, fast sampling, and feature caching, which causes some loss of image information during generation. Despite this, the difference between images generated by DDIMCache and the true image distribution remains within an acceptable range. This balance between performance and quality is particularly valuable in mobile applications requiring fast image generation with limited computational resources.

Below, we detail the differences for Stable Diffusion 1.5 and DDIMCache:

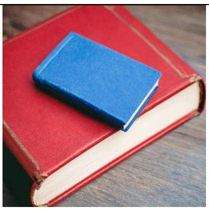
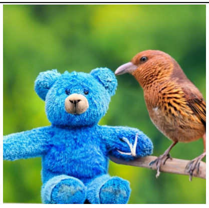
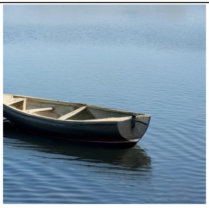
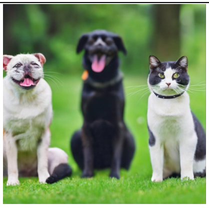
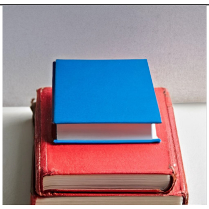
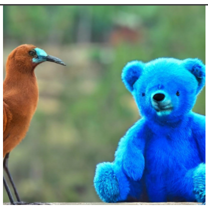
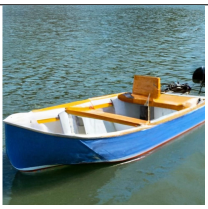
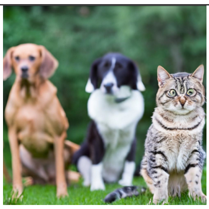
1. Technical Differences:
   Stable Diffusion 1.5 uses the full UNet architecture and employs standard fast sampling methods. The memory usage is about 10GB and requires a high-end GPU (such as an RTX 3090) on a desktop computer. DDIMCache has undergone quantization and pruning to reduce the model's memory footprint, introducing feature caching methods and the DDIM, which significantly reduces sampling steps and computational load. The sampling steps and cache steps can be flexibly adjusted as needed. The memory usage is about 1.8GB, making it runnable on mobile devices with decent computational power.

2. Performance Comparison:
   On a desktop computer with an RTX 3090 GPU, Stable Diffusion 1.5 generates 512x512 images with an average latency of 5.7 seconds and an FID of 4.6. Running the Stable Diffusion 1.5 model on the memory- and compute-constrained Samsung S23 Ultra is impractical. Using the DDIMCache with $n = 5$ cache steps and 50 sampling steps, the optimized diffusion model can be deployed and run on the Samsung S23 Ultra, generating 512x512 images with an average latency of 6.2 seconds and an FID of 19.8.

Below is an introduction to suitable and unsuitable scenarios for DDIMCache:

Best-Suited Scenarios:

1. Real-time image generation on mobile devices

| | A small blue book sitting on a large red book. | A brown bird and a blue bear. | A vessel propelled on water by oars, sails, or an engine. | One cat and two dogs sitting on the grass. |
|---|---|---|---|---|
| Stable Diffusion1.5 Latency=5.7s FID=4.6 |  |  |  |  |
| DDIMCache Latency=6.2s FID=19.8 |  |  |  |  |

**Tab. 4.** Comparison of Stable Diffusion and DDIMCache on Image Generation.

2. Scenarios sensitive to latency
3. Environments with limited computational and memory resources
4. Scenarios requiring fast response times for user interaction

Less-Suited Scenarios:

1. Scenarios with abundant computational resources and no speed constraints
2. Scenarios requiring precise image detail reproduction

For scenarios requiring quick preview generation, more aggressive optimization parameters can be used. For final output images, increasing the sampling steps or reducing the cache interval can improve image quality. Overall, DDIMCache is an optimization solution for mobile scenarios, particularly well suited for applications requiring fast image generation with constrained resources. By sacrificing some image quality, DDIMCache significantly improves performance, enabling the deployment of diffusion models on mobile devices. The decision to use DDIMCache should depend on specific application requirements, hardware conditions, and user experience considerations.

## 4.4. Algorithm comparison

Although some research exists on deploying large models on mobile devices, the number of studies is still limited, and the specific research data only includes latency and optimization methods. Next, Table 5 will be used to analyze and compare existing optimization methods, and several trends for deploying and running large models on mobile devices in the future will be presented.

1. All methods employ fast sampling techniques, which, as previously mentioned, are the most effective for optimizing diffusion models. Fast sampling can increase inference speed by several orders of magnitude while maintaining image quality. Such specialized optimization methods tailored to specific models will play a crucial role.

| Method | Latency | Fast Sampling | Fid | Main Optimization |
|---|---|---|---|---|
| HOU & ASGHAR | $\sim$ 15S | 20 steps | Not mentioned | Qualcomm AI Stack |
| CHEN ET AL. | $\sim$ 12S | 20 steps | Not mentioned | GPU-aware optimization |
| DDIMCache | $\sim$ 6S | 50 steps | 19.8 | Multiplicative Effect Joint Optimization |

**Tab. 5.** Comparison of different optimization implementations on GALAXY S23.

2. Hou and Asghar's approach was the first to deploy the Stable Diffusion model on Android smartphones, leveraging the Qualcomm AI Stack for optimization, enabling the generation of 512x512 pixel images on the GALAXY S23 in 15 seconds.Qualcomm combines its hardware advantages with neural network models, algorithms, and software into a unified AI stack, assisting developers in creating, optimizing, and deploying AI applications. The Qualcomm AI Stack plays a role in acceleration, but this general hardware and software integrated optimization approach does not have an advantage compared to other specialized optimization methods tailored to specific models.

3. Chenet al. proposed various GPU-specific optimizations that generate 512x512 pixel images in under 12 seconds. Compared to other methods with the same sampling steps, they achieved a latency advantage,indicating that the GPU acceleration methods play a significant role

4. DDIMCache proposed in this paper significantly reduces the generation latency to approximately 6 seconds while maintaining high image quality, clearly outperforming other methods. By utilizing multiple optimization algorithms tailored to specific models, this approach maximizes the strengths of each method, enabling them to complement and exchange information, ultimately achieving synergistic effects that exceed the performance of any individual method. This could be a key approach for deploying large models on mobile devices in the future.

## 5. CONCLUSION

We propose DDIMCache, a text-to-image diffusion model designed specifically for mobile devices.DDIMCache achieves optimal speed, generating 512 × 512 images in around 6 seconds.In hardware and performance-constrained mobile environments, DDIMCache effectively maintains image generation quality and achieves optimal performance.This provides mobile users with powerful image generation capabilities and an enhanced user experience.

## ACKNOWLEDGEMENT

This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors. Funding for this research was covered by the author(s) of the article.

(Received September 30, 2024)

## REFERENCES

[1] R. Rombach, A. Blattmann, D. Lorenz et al.: High-resolution image synthesis with latent diffusion models. In: Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition 2022, pp. 10684–10695. DOI:10.1109/CVPR52688.2022.01042

[2] J. Hou and Z. Asghar: World's first on-device demonstration of stable diffusion on an android phone. Qualcomm *24* (2023). https://www.qualcomm.com/news/onq/2023/02/worlds-first-on-device-demonstration-of-stable-diffusion-on-android

[3] Y. H. Chenm R. Sarokin, J. Lee et al.: Speed is all you need: On-device acceleration of large diffusion models via gpu-aware optimizations. In: Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition 2023. pp. 4651–4655. DOI:10.1109/CVPRW59228.2023.00490

[4] Y. Shang, Z. Yuan et al.: Post-training quantization on diffusion models. In: Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition 2023. pp. 1972–1981. DOI:10.1109/CVPR52729.2023.00196

[5] X. Li, Y. Liu, L. Lian et al.: Q-diffusion: Quantizing diffusion models. In: Proc. IEEE/CVF International Conference on Computer Vision 2023: pp. 17535–17545. DOI:10.1109/ICCV51070.2023.01608

[6] X. Ma, G. Fang, and X.Wang: Llm-pruner: On the structural pruning of large language models. Adv. Neural Inform. Process. Systems *36* (2023), 21702–21720.

[7] Y. Li, G. Yuan, Y. Wen et al.: Efficientformer: Vision transformers at mobilenet speed. Adv. Neural Inform. Process. Systems *35* (2022), 12934–12949.

[8] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan et al.: Deep unsupervised learning using nonequilibrium thermodynamics. In: International Conference on Machine Learning PMLR, 2015, pp. 2256–2265.

[9] Jiaming Song, Chenlin Meng, and Stefano Ermon: Denoising diffusion implicit models. 2020. In: arXiv preprint: 2010.02502

[10] S. M. Jain: Hugging face. Introduction to transformers for NLP: With the hugging face library and models to solve problems. Apress, Berkeley 2022, 51–67. DOI:10.1007/978-1-4842-8844-3_4

[11] O. Ronneberger, P. Fischer, T. Brox U-net: Convolutional networks for biomedical image segmentation. Medical image computing and computer-assisted interventional MICCAI 2015. In: Proc. 18th international conference, Munich 2015, part III 18. Springer International Publishing, pp. 234–241.

[12] T. Y. Lin, M. Maire, S. Belongie et al.: Microsoft coco: Common objects in context. Computer Vision'ECCV 2014. In: Proc. 13th European Conference, Zurich 2014, Part V 13. Springer International Publishing 2014, pp. 740–755.

[13] A. Q. Nichol and P. Dhariwal: Improved denoising diffusion probabilistic models. In: International Conference on Machine Learning, PMLR 2021, pp. 8162–8171.

*Wu Qifeng, Gannan University of Science and Technology, Ganzhou, JiangXi, 341000. P. R. China.*
*e-mail: 38896500@qq.com*