
FlowDCN: Exploring DCN-like Architectures for Fast Image Generation with Arbitrary Resolution

Shuai Wang
Nanjing University

Zexian Li
Alibaba Group

Tianhui Song
Nanjing University

Xubin Li
Alibaba Group

Tiezheng Ge
Alibaba Group

Bo Zheng
Alibaba Group

Limin Wang✉
Nanjing University, Shanghai AI Lab



Figure 1: **Selected arbitrary-resolution samples (384x384, 224x448, 448x224, 256x256).** Generated from a single FlowDCN-XL/2 model trained on ImageNet 256x256 resolution with CFG = 4.0.

Abstract

Arbitrary-resolution image generation still remains a challenging task in AIGC, as it requires handling varying resolutions and aspect ratios while maintaining high visual quality. Existing transformer-based diffusion methods suffer from quadratic computation cost and limited resolution extrapolation capabilities, making them less effective for this task. In this paper, we propose FlowDCN, a purely convolution-based generative model with linear time and memory complexity, that can efficiently generate high-quality images at arbitrary resolutions. Equipped with a new design of learnable group-wise deformable convolution block, our FlowDCN yields higher flexibility and capability to handle different resolutions with a single model. FlowDCN achieves the state-of-the-art 4.30 sFID on 256×256 ImageNet Benchmark and comparable resolution extrapolation results, surpassing transformer-based counterparts in terms of convergence speed (only $\frac{1}{5}$ images), visual quality, parameters (8% reduction) and FLOPs (20% reduction). We believe FlowDCN offers a promising solution to scalable and flexible image synthesis.

1 Introduction

Image generation is an important task in computer vision research, which is aimed at capturing the inherent data distribution of original image datasets and generating high-quality synthetic images through sampling. Diffusion models [1, 2, 3, 4, 5] have recently emerged as a highly promising foundation for training algorithms in image generation, outperforming GAN-based models [6, 7] and Auto-Regressive models [8] by a significant margin. The evolution of diffusion models is fast, transitioning from discrete forms [1] to SDE-based continuous forms [2, 3, 4, 5, 9]. In a nutshell, diffusion models incrementally degrade an image through a time-dependent stochastic perturbation process and then learn the reverse process to restore the original image from its corrupted state.

Beyond theoretical advancements in diffusion models, the architecture of these models also significantly influences the quality of generated images. Many works [1, 10, 11] in the diffusion domain adopt a standard UNet architecture as the generation backbone, which consists of downsample blocks, upsample blocks, and long residual connections between these components. Inspired by the success of the vision transformer in perception tasks, DiT [12] eliminates the long residual connection in favor of a pure transformer-based architecture. Through rigorous experiments, DiT demonstrates that the UNet inductive bias is not essential for achieving high performance in diffusion models [12]. Meanwhile, PixArt [13, 14] and SD3 [15] venture further by significantly increasing the number of parameters, exploring new frontiers in model architecture and its impact on image generation.

When considering the generation of images at arbitrary resolution, diffusion transformers need to confront at least two primary challenges. The first is the *quadratic computation cost*: the architecture of diffusion transformers employs attention mechanisms to aggregate spatial tokens. Owing to the dense nature of attention computations, high-resolution image generation inevitably leads to significant computation and memory demands, both scaling with $O(n^2)$ complexity. To address the quadratic computation challenge, some methods [16] have adapted recurrent computational strategies from natural language processing. However, these adaptations do not fully capitalize on the strengths of autoregressive tasks and result in slower inference speeds due to the reduced parallelism inherent in RNN-based scanning. The second challenge is *resolution extrapolation*: many diffusion transformers rely on absolute position embedding (APE) [17] to incorporate positional information, introducing it at the onset of the model. This approach forces subsequent layers to become overfitted to the APE for providing positional context to the attention layers, which presents a significant barrier when extrapolating to different resolutions. To address this issue, FiT [18] has turned to Rotary Positional Encoding [19], incorporating RoPE2D to enhance its resolution extrapolation capabilities. Nevertheless, FiT still requires a training pipeline tailored to arbitrary-resolution generation.

In contrast, convolutional models are the most common choice of visual encoders, boasting linear complexity and aggregating spatial features based on relative positions. With the support of modern convolution operators [20, 21, 22], convolutional models have demonstrated comparable performance or even surpassed transformers in perception tasks. This naturally leads us to inquire: *Can modern convolutional networks achieve arbitrary-resolution generation efficiently and outperform transformer counterparts?* To answer this question, we opt for deformable convolution as the basic block for exploration in generation, owing to its superior performance in perception tasks.

Specifically, we propose a novel approach to decouple the scale and direction prediction of deformable convolution, giving rise to a group-wise multiscale deformable convolution block that enables efficient multiscale feature aggregation. By leveraging this block, we introduce FlowDCN, a modern purely convolution generative model that tackles arbitrary-resolution generation. Thanks to the new design of convolutional deformable block, our FlowDCN yields higher flexibility and capability to handle different resolutions with a single model. The experiments demonstrate that FlowDCN consistently surpasses its diffusion transformer counterparts, DiT [12] and SiT [23]. Notably, on the 256x256 ImageNet benchmark, FlowDCN achieves faster convergence, yielding SoTA sFid of 4.30 and FID of 2.13 under 1.5M steps with batch size 256, while exhibiting 20% lower latency, 8% fewer parameters, and approximately 20% fewer floating-point operations (FLOPs). On the 512x512 ImageNet benchmark, FlowDCN achieves 4.53 sFid and 2.44 FID under 100K finetuning steps with batch size 256.

Moreover, our FlowDCN offers a significant advantage in fast arbitrary-resolution generation, as it only requires linear time and memory complexity. Through visualization comparisons, our FlowDCN demonstrates substantially better visual quality even at extremely small sampling steps, such as 3, 4, and 5 steps. To further enhance its visual quality, we propose Scale Adjustment, a simpler

technique for extrapolating resolution to unseen dimensions. Our results show that FlowDCN achieves comparable resolution extrapolation capabilities to highly tailored methods, underscoring its potential for generating high-quality images at various resolutions. The contributions can be summarized as:

- We decouple the scale and direction priors of deformable convolution and propose a Group-wise MultiScale Deformable Block. Building upon this block, we propose FlowDCN, a purely convolution-based generative model with high efficiency.
- On 256x256 ImageNet benchmark, under only 1.5M training steps, our FlowDCN-XL/2 achieves 2.13 FID and SoTA 4.30 sFID with Euler solver and classifier free guidance.
- On 512x512 ImageNet benchmark, under only 100K finetuning steps, our FlowDCN-XL/2 achieves 2.44 FID and 4.53 sFID with Euler solver and classifier free guidance.
- We propose a much simple and efficient resolution extrapolation method, deemed as Scale Adjustment. For arbitrary resolution generation, we achieve comparable results to highly tailored methods.

2 Preliminary

2.1 Linear-based Flow Matching

Flow matching [4, 5] is a simple but powerful diffusion family. We incorporate linear-based flow matching as the training framework for its simplicity. Given the image sampled x from training distributions and the noise ϵ sampled from a Gaussian distribution, linear-based flow matching forward process interpolate x_t with x and ϵ using the following equation:

$$x_t = tx + (1 - t)\epsilon. \quad (1)$$

The velocity field of linear-based flow matching [4, 5] is defined as Eq. (2). We train our FlowDCN to predict the time-dependent velocity field between x and ϵ :

$$v_t(x_t) = x - \epsilon. \quad (2)$$

During training, the flow matching objective directly regresses the target velocity:

$$\mathcal{L}_v = \int_0^1 \mathbb{E}[\|v_\theta(x_t, t) - v_t(x_t)\|^2] dt. \quad (3)$$

For sampling, the common ODE/SDE solver *e.g.* Euler method, Heun method can be employed.

2.2 Deformable Convolution Revisited

Given an image feature $\mathbf{x} \in \mathbb{R}^{H \times W \times D}$, deformable convolution predicts the deformable field $\Delta\mathbf{P}(\mathbf{x}) \in \mathbb{R}^{H \times W \times G \times K \times 2}$ and the dynamic weights $\mathbf{W}(\mathbf{x}) \in \mathbb{R}^{H \times W \times G \times K}$ from the image feature $\mathbf{x}$. Specifically, H and W represent the height and width of the feature spatial shape, D is the feature channel, K is the number of sampling points, and G is the number of groups in the deformable convolution operation. The deformable field and dynamic weights are computed as Eq. (4):

$$\Delta\mathbf{P}(\mathbf{x}) = \mathbf{W}_{\text{deformable}}^T \mathbf{x} + \mathbf{b}_{\text{deformable}}, \quad (4)$$

$$\mathbf{W}(\mathbf{x}) = \mathbf{W}_{\text{weight}}^T \mathbf{x} + \mathbf{b}_{\text{weight}}. \quad (5)$$

For a specific group g in deformable convolution, the sampling position is determined by the base feature position p_0 , sampling position prior p_k , and predicted deformable Δp_k from $\Delta\mathbf{P}(\mathbf{x})$ for the k -th sampling point. The dynamic weight w_k is provided from $\mathbf{W}(\mathbf{x})$. The deformable convolution aggregates K sparse spatial features according to the sampling location and dynamic weight as following:

$$\mathbf{y}^g(p_0) = \sum_{k=0}^K w_k^g \mathbf{x}^g(p_0 + p_k + \Delta p_k(\mathbf{x})), \quad (6)$$

$$\mathbf{y} = \text{concat}(\mathbf{y}^1, \mathbf{y}^2, \dots, \mathbf{y}^G). \quad (7)$$

The predefined spatial position prior p_k is initialized from the regular convolution, commonly using $(-1, -1), (-1, 0), \dots, (0, 0), \dots, (1, 1)$ as the predefined value.

Deformable convolution introduces long-range dependencies and dynamic aggregation into regular convolutions, bridging the gap between convolution and multi-head self-attention [24]. Thus, deformable convolution shares the efficiency merit of convolution and the dynamics merit of the attention mechanism. In most scenarios, DCN-like architectures are more powerful than common CNNs, we provide comparison experiments of DCN and CNN of flow matching training. Notably, deformable convolution directly predicts the dynamic weights and only aggregates limited features from spatial locations, enjoying a relatively sparse computation diagram. A deformable convolution operator only requires $\frac{4KHW C}{G}$ FLOPs for computation when employing bilinear sampling to aggregate features.

3 Method

3.1 Multi-Scale Deformable Convolution

The original deformable convolution has been widely adopted in hierarchical model architectures [25, 24, 20] for perception tasks. However, these models typically progressively downsample the feature maps to increase the reception field growth rate. In contrast, image generation tasks require outputs with more high-frequency details and low-level information. From this perspective, progressively downsampling features would lead to the loss of high-frequency details. One possible solution is to introduce long residual connections to generation models [11, 10]. However, in practice, this approach demands caching image features from the encoder part, which increases peak memory usage during model inference.

To strike a balance between receptive fields and high-frequency details, we propose decoupling the deformable field into scale and direction, and introduce a novel multiscale deformable convolution. Unlike previous deformable convolutions, our approach assigns different scale priors to different groups.

Decoupling deformable field to direction and scale. The original deformable convolution directly regresses the deformable field to learn an unbounded and adaptive sampling point generator. However, the vast image spatial range poses a challenge to the learning process, as it leads to unstable regression of the deformable range when extrapolating from local neighbors to distant feature locations. We tackle this problem by decoupling the direction and scale of the deformable field. Specifically, we reorganize the sampling point formulation in Eq. (9).

$$s(\mathbf{x}) = S_{\max} * \text{sigmoid}(\mathbf{W}_s^T \mathbf{x}), \quad (8)$$

$$p = p_0 + s(\mathbf{x}) * (p_k + \Delta p_k(\mathbf{x})), \quad (9)$$

where $s(\mathbf{x})$ is the learnable scale predicted from the image feature $\mathbf{x}$. $S_{\max}$ is the max scale value of the given deformable convolution, we leave it as a hyper-parameter only related to input resolution, thus we can manually tune it according to input resolution, details are placed in Sec. 3.3.

Group-wise multi-scale deformable convolution. To keep high-resolution feature maps and own a large reception field growth rate, we propose to assign different scale priors to different groups. This allows deformable groups with large scale priors to aggregate long-dependency features, while those with small scale priors aggregate short-dependency features as follows:

$$s^g(\mathbf{x}) = S_{\max} * \text{sigmoid}(\mathbf{W}_s^T \mathbf{x} + s_0^g), \quad (10)$$

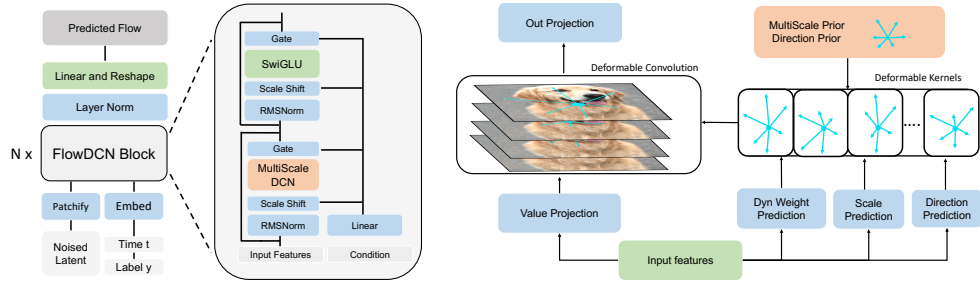
$$p = p_0 + s^g(\mathbf{x}) * (p_k + \Delta p_k(\mathbf{x})). \quad (11)$$

Specifically, we initialize the scale priors with Eq. (12) and initialize $\mathbf{W}_s$ with zeros to obtain linearly increased $\text{sigmoid}(s_0^g)$ along group axis:

$$s_0^{g+1} = \log\left(\frac{g}{G-g}\right). \quad (12)$$

3.2 Flow-based Deformable Convolutional Generative Model

We introduce our novel diffusion generation architecture, dubbed FlowDCN. Rather than directly adopting tailored architectures for image generation, such as long residuals and normalization techniques, we aim to explore the generative capabilities of deformable convolution-based architectures



(a) **FlowDCN Architecture.** Our FlowDCN consists of stacked MultiScaleDCN blocks and SwiGLU blocks. We also employ RMSNorm to stabilize training. (b) **MultiScale DCN Block.** Dynamic weight and scale& direction deformable field are predicted from input features, then merged with priors to form the deformable kernels to extract features.

Figure 2: **The Architecture of Our FlowDCN and MultiScale DCN Block.**

in a faithful manner. To this end, we deliberately discard long residual connections and opt to build a pure convolution-based generative model, preserving the unique characteristics of DCN-like models as much as possible. For training and sampling, we leverage the powerful flow matching algorithm to align our model with the state-of-the-art SiT [23].

Deformable convolution generative model. The model architecture is illustrated in Fig. 2a. We aim to build a pure DCN-like generative model to explore the generation ability of DCN-like [20] architectures. To match the base resolution of model input with DiT [12] and SiT [23], we similarly patchify the noisy input via convolution. Inspired by DiT [12], we inject the timestep and label conditions through adaLN-Zero [12, 26]. The basic block is formulated as Eq. (13). Drawing inspiration from LLaMA [27, 28], we replace vanilla FFN and LayerNorm with SwiGLU and RMSNorm, respectively. Note we also provide FFN and LayerNorm version FlowDCN for fair comparisons:

$$\mathbf{x}_1 = \mathbf{x} + \text{AdaLN}(\mathbf{y}, \mathbf{t}, \text{MultiScale-DCN}(\mathbf{x})), \quad (13)$$

$$\mathbf{x}_2 = \mathbf{x}_1 + \text{AdaLN}(\mathbf{y}, \mathbf{t}, \text{SwiGLU}(\mathbf{x}_1)). \quad (14)$$

3.3 Arbitrary Resolution Sampling

We denote the training resolution as $H_{\text{train}} \times W_{\text{train}}$ and the inference resolution as $H_{\text{test}} \times W_{\text{test}}$. Notably, our FlowDCN is capable of handling arbitrary resolution that differs from the training resolution. As a reminder, the multiscale deformable convolution block aggregates features based on predicted scales and directions according to Equation (Eq. (9)). In practice, the predicted scale of the multiscale deformable convolution layer is typically fitted to match the training resolution distribution. However, this limits the reception fields of the image features when encountering unseen resolution, ultimately hurting the global semantic consistency [29, 30]. To improve the global semantic consistency, we propose adjusting the scaling factor based on the relative ratio between the training resolution and inference resolution.

Adjust S_{max} to match inference resolution. As shown in Eq. (10), S_{max} controls the maximum sampling range in multiscale deformable convolution. As discussed in Sec. 3.1, we treat it as a resolution-dependent hyperparameter. It is straightforward to observe that scaling S_{max} with the relative aspect ratio between train size and inference size could match the reception field between train and inference:

$$s_h^g(\mathbf{x}) = \text{sigmoid}(\mathbf{W}_s^T \mathbf{x} + s_0^g) \cdot S_{\text{max}} \cdot \frac{H_{\text{test}}}{H_{\text{train}}}, \quad (15)$$

$$s_w^g(\mathbf{x}) = \text{sigmoid}(\mathbf{W}_s^T \mathbf{x} + s_0^g) \cdot S_{\text{max}} \cdot \frac{W_{\text{test}}}{W_{\text{train}}}. \quad (16)$$

Operator	Runtime (ms) of Input Shape $H \times W \times G \times D$			
	$16 \times 16 \times 16 \times 64$	$16 \times 16 \times 16 \times 128$	$32 \times 32 \times 16 \times 64$	$32 \times 32 \times 16 \times 128$
Attention (Math SDP)	0.92/2.1	1.16/2.71	10.7/28.8	12.4/35.8
Attention (Flash SDP) [34]	0.62/N	1.47/N	4.98/N	14.4/N
DeformConv(DCNv4 [20])	0.77/1.00	1.0/2.1	2.8/4.4	3.9/8.3
DeformConv(Shm)	0.56/0.81	1.1/1.5	2.7/3.9	5.0/7.3
DeformConv(Triton-lang)†	0.83/0.89	0.95/1.1	3.4/3.8	4.0/4.8

Table 1: **Op-level benchmark on standard input shape of Diffusion backbone task.** FP16/FP32 results are collected on Nvidia A10 GPU. We use 32 batch sizes for benchmarking. † indicates our Triton-lang [32] implementation of DCNv4. N indicates implementation is not available.

Models	FID↓	sFID↓	IS↑	Kernel	FID↓	sFID↓	IS↑	p_k	$s(x)$	FID↓	sFID↓	IS↑
SiT-S/2	7.42	4.47	8.7	4	5.88	4.6	8.89	fixed	fixed	5.6	4.58	8.90
FlowDCN-S/2	5.47	4.35	8.89	9	5.47	4.35	8.89	fixed	learn	5.47	4.35	8.89
w/o MultiScale	5.72	4.42	8.85	16	5.39	4.54	8.93	learn	fixed	6.01	4.43	8.85
w/o PriorInit	5.68	4.49	8.9	32	5.13	4.43	9.05	learn	learn	5.63	4.37	8.89

(a) **Comparisons with SiT.** Our FlowDCN outperforms SiT by a significant margin. (b) **KernelSize K of FlowDCN.** large kernel size produces better results than small one. (c) **Deformable fields learning setting.** Default achieves best results.

Table 2: **Ablation Studies and Comprasion with other flow-based method on 32x32 CIFAR Dataset.** In order to fully align with SiT [23], here we replace our SwiGLU and RMSNorm with FFN and LayerNorm armed in SiT, respectively. **Bold** font indicates the default setting.

4 Experiments

We conduct experiments on 32x32 CIFAR10 and 256x256 ImageNet datasets. The training batch size is set to 256. Similar to SiT [23] and DiT [12], we use Adam optimizer [31] with a constant learning rate 0.0001 during the whole training. We do not adopt any gradient clip techniques for fair comparison. For 32x32 CIFAR10 dataset, we train our model for 25000 steps. As for 256x256 ImageNet dataset, we train for 1.5M steps. We use 8×A100 GPUs as the default training hardware.

Efficient deformable convolution implementation. Although DCNv4 [20] proposes a much faster deformable convolution implementation, it is not tailored for image generation input shape. For resolution below 512×512 , there are fewer spatial tokens (only 16×16 tokens for 256×256 resolution) to fully utilize sparse computation strengths, thus DCNv4 exhibits even worse latency compared to attention. To remedy high latency of deformable convolution for low-resolution scenery, we decide to leverage shared memory to reduce the latency of random sampling in deformable convolutions, deemed as DeformConv(shm). We place the performance benchmark at Tab. 1. For high-resolution scenery, We also re-implement DeformConv(DCNv4) in Triton-lang as DeformConv(Triton-lang) to leverage the strengths of compiler [32, 33] to find suitable hyperparameters.

4.1 32x32 CIFAR Dataset

The CIFAR10 dataset[35], comprising 50,000 32x32 small-resolution images from 10 distinct class categories, is considered an ideal benchmark to validate the design of our MultiScale deformable block due to its relatively small scale. We select SiT-S/2 as a comparison baseline, as it also leverages the flow-matching framework. For sampling, we employ the Euler stochastic solver with 1000 sampling steps to generate images. We report the FID [36], sFID [37], and Inception Score [38] as the primary metrics to evaluate the performance of our model.

Compare with baseline SiT. We summarise the metrics of our FlowDCN and SiT in Tab. 2a. Our FlowDCN achieves 5.47 fid, surpassing its counterpart SiT with 2.0 fid margins. Additionally, our model performs slightly better in terms of sFID and Inception Scores, further demonstrating its superiority.

Group-wise multiscale design. As showed in Tab. 2a, we denote the variant of FlowDCN that uses vanilla deformable convolution instead of Multiscale deformable convolution as *w/o MultiScale*. Notably, the absence of group-wise multiscale deformable convolution leads to a 0.25 FID performance degradation. This result demonstrates the effectiveness and power of our proposed group-wise multiscale mechanism.

Prior initialization. By default, we manually initialize the direction priors with predefined grids $\{(-1, -1), (-1, 0), \dots (0, 0), \dots (1, 1)\}$, and initialize the scale priors with linearly increased scale along group axis. We also experiment with randomly initialized direction and scale priors in Tab. 2a, donated as *w/o PriorInit*. Random initialization shows slight performance degradation.

Sampling points. In Tab. 2b, We train FlowDCN with varying kernel sizes K and observe that the performance consistently improves as the number increases. Specifically, using 32 points to aggregate features, FlowDCN achieves a FID score of 5.13 and an sFID score of 4.43. However, to maintain a relatively sparse pattern, we choose $K = 9$ as the default setting, striking a balance between performance and computational efficiency.

Fixed direction priors. In Tab. 2c, we present the results of training FlowDCN with different prior learning settings. Notably, we find that the fixed direction prior p_k in Eq. (9) achieves better results compared to the learnable direction prior. We hypothesize that the learnable direction prior may cause the learning of the deformable field to become unstable, leading to inferior performance.

Learnable relative scale. In the $s(\mathbf{x})$ column of Tab. 2c, the notation "*learn*" indicates that we predict a relative scale of the deformable fields in addition to the learnable scale priors s_0^g ($\mathbf{W}_s^T \mathbf{x}$ in Eq. (10)), whereas "*fixed*" does not predict the relative scales $s(\mathbf{x})$ in the deformable field. Learning a relative scale for each feature in Tab. 2c achieves better results of 5.47 FID.

4.2 256×256 ImageNet Dataset

Based on our analysis, we select the MultiScale deformable convolution with a kernel size of $K = 9$ as the basic block for our Imagenet experiments. Our default setting involves fixing the direction priors and learning relative scales from the deformable field. We manually initialize the direction priors with predefined grids and initialize the scale priors with linearly increased scales. To generate images, we employ an Euler-Maruyama solver with 250 steps for stochastic sampling. We report the FID, sFID, Inception Score, and Precision & Recall as the primary metrics to evaluate the performance of our model.

Model	FLOPs (G)	Params (M)	Latency(ms)	FID↓	sFID↓	IS↑
SiT-S/2	6.06	33	0.026	57.64	9.05	24.78
SiT-S/2 †	6.06	33	0.026	57.9	8.72	24.64
FlowDCN-S/2	4.36 (-28%)	30.3 (-8.1%)	0.027	54.6	8.8	26.4
SiT-B/2	23.01	130	0.084	33.5	6.46	43.71
SiT-B/2 †	23.01	130	0.084	37.3	6.55	40.6
FlowDCN-B/2	17.87 (-22%)	120 (-7.6%)	0.076	28.5	6.09	51
w/o RMS & SwiGLU	17.88 (-22%)	120 (-7.6%)	0.072	29.1	6.13	50.4
DiT-L/2	80.71	458	0.291	23.3	-	-
SiT-L/2	80.71	458	0.291	18.8	5.29	72.02
FlowDCN-L/2	63.51 (-21%)	421 (-8.0%)	0.254	13.8	4.69	85
DiT-XL/2	118.64	675	0.387	19.5	-	-
SiT-XL/2	118.64	675	0.387	17.2	5.07	76.52
FlowDCN-XL/2	93.24 (-21%)	618 (-8.4%)	0.303	11.3	4.85	97

Table 3: **Image generation metrics comparisons between SiT [23], DiT [12] under 400k training steps budgets.** All metrics are calculated from the sampled 50k images under 250 Euler SDE sampling steps without classifier-free guidance. †: reproduced result. Latency(ms) is the 1-NFE latency and collected from Nvidia A10 GPU with 16 batchsize under float32.

Metrics comparison with baseline SiT. We present the performances of different-size models at 400K training steps in Tab. 8. From Small to XL-size models, our FlowDCN model family consistently outperforms its counterpart DiT [12] and SiT [23] with significant margins. Without RMS/SwiGLU, our FlowDCN-B/2 degrades with 0.6 FID gains but still surpasses SiT by a large margin. In addition to its superior performance and convergence speed, our FlowDCN also boasts a remarkable 8% reduction in parameters and at least 20% reduction in FLOPs compared to DiT/SiT. This demonstrates that our FlowDCN surpasses vision transformer-based generation models in multiple aspects.

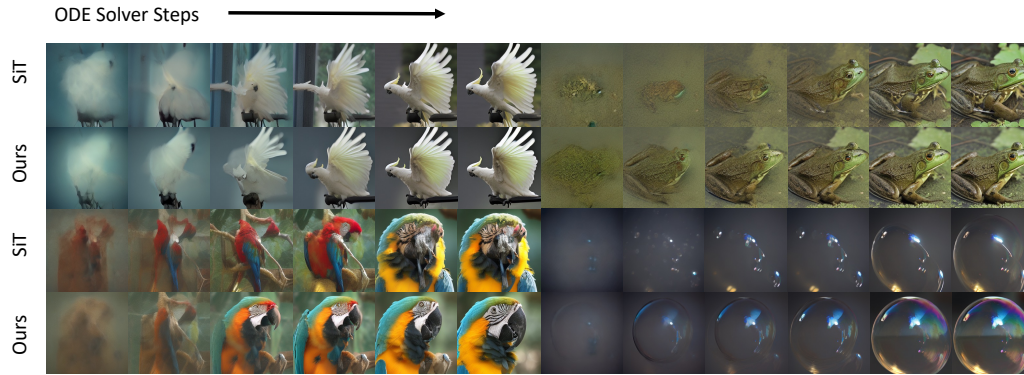


Figure 3: **Visualization Comparison with SiT.** *Best viewed zoomed-in.* We sample both our FlowDCN-XL/2 and SiT-XL/2 with Euler ODE solver under 2, 3, 4, 5, 8, 10 steps using the same latent noise. At the fewer steps sampling scenery, our FlowDCN generates slightly clearer and higher-quality images.

ImageNet 256×256 Benchmark

Generative Models	Long Residuals	Total Images(M)	Total GFLOPs	FID ↓	sFID ↓	IS ↑	P ↑	R ↑
ADM-U [10]	✓	507	3.76×10^{11}	7.49	5.13	127.49	0.72	0.63
CDM [39]	✓	-	-	4.88	-	158.71	-	-
LDM-4 [40]	✓	213	2.22×10^{10}	10.56	-	103.49	0.71	0.62
DiT-XL/2 [12]	✗	1792	2.13×10^{11}	9.62	6.85	121.50	0.67	0.67
DiffusionSSM-XL[16]	✗	660	1.85×10^{11}	9.07	5.52	118.32	0.69	0.64
SiT-XL/2[23]	✗	1792	2.13×10^{11}	8.61	6.32	131.65	0.68	0.67
FlowDCN-XL/2	✗	384	3.57×10^{10}	8.36	5.39	122.5	0.69	0.65
Classifier-free Guidance								
ADM-U[10]	✓	507	3.76×10^{12}	3.60	-	247.67	0.87	0.48
LDM-4 [40]	✓	213	2.22×10^{10}	3.95	-	178.22	0.81	0.55
U-ViT-H/2 [11]	✓	512	6.81×10^{10}	2.29	-	247.67	0.87	0.48
DiT-XL/2 [12]	✗	1792	2.13×10^{11}	2.27	4.60	278.24	0.83	0.57
DiffusionSSM-XL [16]	✗	660	1.85×10^{11}	2.28	4.49	259.13	0.86	0.56
SiT-XL/2[23]	✗	1792	2.13×10^{11}	2.06	4.50	270.27	0.82	0.59
FiT-XL/2[18]	✗	450	-	4.27	9.99	249.72	0.84	0.51
FlowDCN-XL/2 (cfg=1.375; ODE)	✗	384	3.57×10^{10}	2.13	4.30	243.46	0.81	0.57
FlowDCN-XL/2 (cfg=1.375; SDE)	✗	384	3.57×10^{10}	2.08	4.38	257.53	0.82	0.57
FlowDCN-XL/2 (cfg=1.375; ODE)	✗	486	4.52×10^{10}	2.01	4.33	254.36	0.81	0.58
FlowDCN-XL/2 (cfg=1.375; SDE)	✗	486	4.52×10^{10}	2.00	4.37	263.16	0.82	0.58

Table 4: **Image generation quality evaluation of and existing approaches on ImageNet 256 × 256.** Total images by training steps × batch size as reported, and total GFLOPs by Total Images × GFLOPs/Image. P refers to Precision and R refers to Recall.

Comparison with other generative models. We report the final metrics of FlowDCN-XL/2 at Tab. 4. Our FlowDCN achieves much faster convergence speed with nearly $\frac{1}{5}$ total images compared its *No-Long-residuals* counterparts. Additionally, using Euler ODE solver and classifier-free guidance with 1.375, our FlowDCN obtains SoTA 4.30 sFID and 2.13 FID results. Training for extra 400k steps, FlowDCN will be further improved to 2.00 FID. As sFID reflects the spatial structure quality [37], better sFID shows our FlowDCN captures better structure distributions. We notice that the IS metric is lower than other models, however, there is an improvement trend along with training iterations.

Visual quality comparison with baseline SiT. We sample both our FlowDCN-XL/2 and SiT-XL/2 with Euler ODE solver for 2, 3, 4, 5, 8, 10 steps, employing the same latent noise for both models. Notably, at the fewer steps sampling scenario, our FlowDCN generates slightly clearer and higher-quality images. We place the generated images at Fig. 3 and Appendix.

4.3 512 × 512 ImageNet Dataset

As training on high-resolution images consumes much more resources, we opt to fine-tune 100k steps from the same model trained on 256 × 256 resolution setting of 1.5M steps (corresponding to

Class-Conditional ImageNet 512×512					
Model	FID↓	sFID↓	IS↑	Precision↑	Recall↑
BigGAN-deep [6]	8.43	8.13	177.90	0.88	0.29
StyleGAN-XL [7]	2.41	4.06	267.75	0.77	0.52
ADM [10]	23.24	10.19	58.06	0.73	0.60
ADM-U [10]	9.96	5.62	121.78	0.75	0.64
ADM-G [10]	7.72	6.57	172.71	0.87	0.42
ADM-G, ADM-U	3.85	5.86	221.72	0.84	0.53
DiT-XL/2 [12]	12.03	7.12	105.25	0.75	0.64
DiT-XL/2-G [12] (cfg=1.50)	3.04	5.02	240.82	0.84	0.54
SiT-XL/2-G [23] (cfg=1.50)	2.62	4.18	252.21	0.84	0.57
FlowDCN-XL/2(cfg=1.375, ODE-50)	2.76	5.29	240.6	0.83	0.51
FlowDCN-XL/2(cfg=1.375, SDE-250)	2.44	4.53	252.8	0.84	0.54

Table 5: **Benchmarking class-conditional image generation on ImageNet 512×512.** Our FlowDCN-XL/2 is fine-tuned for 100k steps from the same model trained on 256×256 resolution setting of 1.5M steps

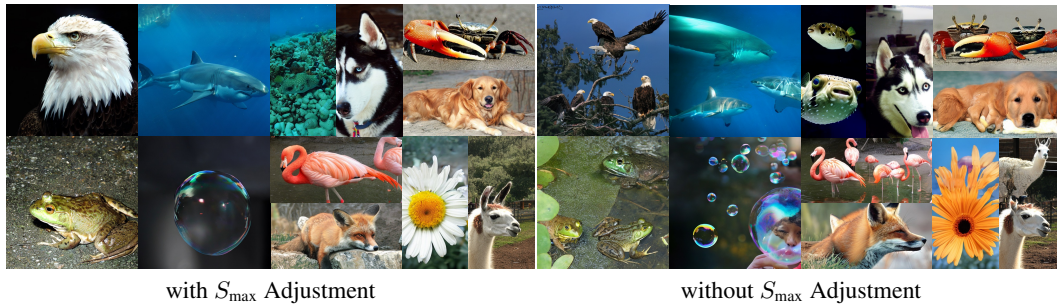


Figure 4: **Visualization Comparison about $S_{\max}$ Adjustment.** Here are the 512×512 , 256×512 and 512×256 , three type resolution images. We employ the same latent noise as start, sampling with Euler SDE solver for 250 steps. With $S_{\max}$ Adjustment, sampled images consistently looks better.

FlowDCN with 384M training images of Tab. 4). Although fine-tuned with limited 100k steps, our FlowDCN demonstrated powerful performance.

Comparison with other generative models. We report the final metrics of FlowDCN-XL/2 on 512×512 ImageNet Dataset at Tab. 5. Our FlowDCN achieves much better FID and sFID performance compared to its counterparts. Using Euler SDE solver with 250 steps and classifier-free guidance with 1.375, our FlowDCN obtains 4.53 sFID and 2.44 FID results. Using Euler ODE solver with 50 steps and classifier-free guidance with 1.375, our FlowDCN obtains 5.29 sFID and 2.76 FID result. As shown in Tab. 5, our FlowDCN achieves better sFID and captures better spatial structure distributions.

4.4 Arbitrary Resolution Extension

For the resolution extrapolation evaluation, we follow the setting in FiT. We select 320×320 and 224×448 as the evaluation arbitrary resolution. It is worth noting that our FlowDCN can handle arbitrary resolution within a reasonable range, the reasonable range is determined by the training setting and training dataset. As our primary goal is to explore DCN-like architectures in universal image generation, we do not intend to enhance the resolution extrapolation nature by data processing. Therefore, we do not employ any multiple aspect ratio training techniques like FiT[18]. Instead, we directly use the FlowDCN model trained on the center-cropped 256×256 ImageNet dataset for arbitrary resolution extension experiments, showcasing the model’s inherent capabilities. Moreover, we provide resolution extension experiments with various aspect ratio training techniques in the Appendix.

Metric comparsion. We report the evaluation results on Tab. 6. For Base-size models, our FlowDCN-B/2 achieves much better results on 320×320 resolution, with 34.4 FID and 35.7 FID using $S_{\max}$ adjustment, outperforming FiT and DiT with a large margin. On 224×448 resolution, our FlowDCN-

Method	320×320 (1:1)			224×448 (1:2)		
	FID↓	sFID↓	IS↑	FID↓	sFID↓	IS↑
DiT-B	95.5	108.7	18.4	109.1	110.7	14.0
DiT-B _{EI}	81.5	62.3	21.0	133.2	72.5	11.1
DiT-B _{PI}	72.5	54.0	24.2	133.4	70.3	11.7
FiT-B	61.4	30.7	31.0	44.7	24.1	37.1
FiT-B _{vYaRN}	44.8	38.0	44.7	41.9	42.8	45.9
FiT-B _{vNTK}	57.3	31.3	34.0	43.8	26.3	39.2
FlowDCN-B/2	34.4	27.2	52.2	71.7	62.0	23.7
+ $S_{\max}$ Adjust	35.7	29.3	51.2	81.1	40.2	21.1

(a) Metrics Results on Base-Size Models

Method	320×320 (1:1)			224×448 (1:2)		
	FID↓	sFID↓	IS↑	FID↓	sFID↓	IS↑
ADM-G,U [10]	9.39	9.01	162	11.34	14.5	146
LDM-4 [40]	6.24	13.21	220	8.55	17.62	186
UViT-H/2 [11]	7.65	16.30	208	67.1	42.92	45.5
MDT-G [41]	383	136	4.24	365	142.8	4.91
DiT-XL/2 [12]	9.98	23.57	225	94.94	56.06	35.7
FiT-XL/2 [18]	5.42	15.41	252	7.9	19.63	215
FlowDCN-L/2	5.99	9.71	238	12.8	17.9	168
FlowDCN-XL/2	5.86	13.5	275	12.9	20.6	184

(b) Metrics Results on Large-Size Models

Table 6: **Benchmarking resolution extrapolations on ImageNet dataset.** On the Base-size Models benchmark, our FlowDCN achieves much better results on 320x320 resolution and comparable results on 224x448 resolution. On the Large-Size Models benchmark, our flowDCN shows comparable extrapolation performance to SoTA models.

B/2 achieves comparable results. Note our FlowDCN not employs any various aspect ratio training in Tab. 6, so we believe our FlowDCN-B/2 can achieve better results when incorporating such training augmentations. For large-size models, we report our FlowDCN-L/2 and FlowDCN-XL/2 with $S_{\max}$ adjustment in Tab. 6b, our model shows comparable results to SoTA models. Meanwhile, we notice that FiT performs poorly on 256x256 resolution in Tab. 4, which we hypothesize is due to resolution-related data augmentation hurting the fitting power of original resolution distributions. Furthermore, as FiT employs the 256×256 reference statistics from ADM Eval Suite[10] to evaluate all resolution(even for 224×448), we suspect this evaluation paradigm is unreasonable.

Visual quality comparison of $S_{\max}$. In Tab. 6a, We notice FlowDCN-B/2 with $S_{\max}$ adjustment does not exhibit better results than directly generating images, we hypothesize that FID and sFID are low level visual quality assessments, not reflecting semantic visual quality. So we also provide the visualization comparisons of our FlowDCN-XL/2 with and without $S_{\max}$ Adjustment in Fig. 4 and Appendix. With $S_{\max}$ Adjustment, generated images consistently look better. But not all the cases demand $S_{\max}$ Adjustment, some images like the bubble and the husky case in Fig. 4, still look good even without $S_{\max}$ Adjustment. More comparison examples can be found in the Appendix.

5 Conclusion

In this paper, we have presented FlowDCN, a novel deformable convolutional network for arbitrary-resolution image generation. Our FlowDCN model leverages the strengths of both group-wise multiscale deformable convolutions and linear flow to generate high-quality images of various resolutions with high flexibility. Through extensive experiments, we demonstrate that FlowDCN outperforms the state-of-the-art transformer-based counterparts in terms of performance, convergence speed, and computational efficiency. Additionally, our model exhibits strong resolution extrapolation capabilities, achieving comparable results to previous models on arbitrary resolution without any additional training techniques. We believe that FlowDCN has a great potential to become a powerful tool for a wide range of image generation tasks and applications.

Limitations and Future Works

Our current implementation of deformable convolution backward is inefficient to be on par with Attention. Our primary focus remains on optimizing the training speed. Once we have made significant strides in training optimization, we plan to scale up our FlowDCN to accommodate larger model parameters and higher training resolution, paving the way for more advanced explorations.

Acknowledgement

This work is supported by the National Key R&D Program of China (No. 2022ZD0160900), the National Natural Science Foundation of China (No. 62076119), the Fundamental Research Funds for the Central Universities (No. 020214380119), the Nanjing University-China Mobile Communications Group Co., Ltd. Joint Institute, and the Collaborative Innovation Center of Novel Software Technology and Industrialization.

References

- [1] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020. [2](#)
- [2] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020. [2](#)
- [3] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. *Advances in Neural Information Processing Systems*, 35:26565–26577, 2022. [2](#)
- [4] Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*, 2022. [2](#), [3](#)
- [5] Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022. [2](#), [3](#)
- [6] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018. [2](#), [9](#)
- [7] Axel Sauer, Katja Schwarz, and Andreas Geiger. Stylegan-xl: Scaling stylegan to large diverse datasets. In *ACM SIGGRAPH 2022 conference proceedings*, pages 1–10, 2022. [2](#), [9](#)
- [8] Huiwen Chang, Han Zhang, Lu Jiang, Ce Liu, and William T Freeman. Maskgit: Masked generative image transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11315–11325, 2022. [2](#)
- [9] Michael S Albergo, Nicholas M Boffi, and Eric Vanden-Eijnden. Stochastic interpolants: A unifying framework for flows and diffusions. *arXiv preprint arXiv:2303.08797*, 2023. [2](#)
- [10] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021. [2](#), [4](#), [8](#), [9](#), [10](#)
- [11] Fan Bao, Shen Nie, Kaiwen Xue, Yue Cao, Chongxuan Li, Hang Su, and Jun Zhu. All are worth words: A vit backbone for diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22669–22679, 2023. [2](#), [4](#), [8](#), [10](#)
- [12] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4195–4205, 2023. [2](#), [5](#), [6](#), [7](#), [8](#), [9](#), [10](#)
- [13] Junsong Chen, Jincheng Yu, Chongjian Ge, Lewei Yao, Enze Xie, Yue Wu, Zhongdao Wang, James Kwok, Ping Luo, Huchuan Lu, et al. Pixart- α : Fast training of diffusion transformer for photorealistic text-to-image synthesis. *arXiv preprint arXiv:2310.00426*, 2023. [2](#)
- [14] Junsong Chen, Chongjian Ge, Enze Xie, Yue Wu, Lewei Yao, Xiaozhe Ren, Zhongdao Wang, Ping Luo, Huchuan Lu, and Zhenguo Li. Pixart- σ : Weak-to-strong training of diffusion transformer for 4k text-to-image generation. *arXiv preprint arXiv:2403.04692*, 2024. [2](#)
- [15] Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. *arXiv preprint arXiv:2403.03206*, 2024. [2](#)
- [16] Jing Nathan Yan, Jiatao Gu, and Alexander M Rush. Diffusion models without attention. *arXiv preprint arXiv:2311.18257*, 2023. [2](#), [8](#)
- [17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. [2](#)
- [18] Zeyu Lu, Zidong Wang, Di Huang, Chengyue Wu, Xihui Liu, Wanli Ouyang, and Lei Bai. Fit: Flexible vision transformer for diffusion model. *arXiv preprint arXiv:2402.12376*, 2024. [2](#), [8](#), [9](#), [10](#)
- [19] Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024. [2](#)
- [20] Yuwen Xiong, Zhiqi Li, Yuntao Chen, Feng Wang, Xizhou Zhu, Jiapeng Luo, Wenhai Wang, Tong Lu, Hongsheng Li, Yu Qiao, et al. Efficient deformable convnets: Rethinking dynamic and sparse operator for vision applications. *arXiv preprint arXiv:2401.06197*, 2024. [2](#), [4](#), [5](#), [6](#)
- [21] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11976–11986, 2022. [2](#)
- [22] Xiaohan Ding, Xiangyu Zhang, Jungong Han, and Guiguang Ding. Scaling up your kernels to 31x31: Revisiting large kernel design in cnns. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11963–11975, 2022. [2](#)

- [23] Nanye Ma, Mark Goldstein, Michael S Albergo, Nicholas M Boffi, Eric Vanden-Eijnden, and Saining Xie. Sit: Exploring flow and diffusion-based generative models with scalable interpolant transformers. *arXiv preprint arXiv:2401.08740*, 2024. 2, 5, 6, 7, 8, 9
- [24] W Wang, J Dai, Z Chen, Z Huang, Z Li, X Zhu, X Hu, T Lu, L Lu, H Li, et al. Internimage: Exploring large-scale vision foundation models with deformable convolutions. *arxiv. arXiv preprint arXiv:2211.05778*, 2022. 4
- [25] Xizhou Zhu, Han Hu, Stephen Lin, and Jifeng Dai. Deformable convnets v2: More deformable, better results. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9308–9316, 2019. 4
- [26] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *Proceedings of the IEEE international conference on computer vision*, pages 1501–1510, 2017. 5
- [27] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023. 5
- [28] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shriti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023. 5
- [29] Yingqing He, Shaoshu Yang, Haoxin Chen, Xiaodong Cun, Menghan Xia, Yong Zhang, Xintao Wang, Ran He, Qifeng Chen, and Ying Shan. Scalecrafter: Tuning-free higher-resolution visual generation with diffusion models. In *The Twelfth International Conference on Learning Representations*, 2023. 5
- [30] Shen Zhang, Zhaowei Chen, Zhenyu Zhao, Zhenyuan Chen, Yao Tang, Yuhao Chen, Wengang Cao, and Jiajun Liang. Hidiiffusion: Unlocking high-resolution creativity and efficiency in low-resolution trained diffusion models. *arXiv preprint arXiv:2311.17528*, 2023. 5
- [31] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 6
- [32] Philippe Tillet, Hsiang-Tsung Kung, and David Cox. Triton: an intermediate language and compiler for tiled neural network computations. In *Proceedings of the 3rd ACM SIGPLAN International Workshop on Machine Learning and Programming Languages*, pages 10–19, 2019. 6
- [33] Chris Lattner and Vikram Adve. Llvm: A compilation framework for lifelong program analysis & transformation. In *International symposium on code generation and optimization, 2004. CGO 2004.*, pages 75–86. IEEE, 2004. 6
- [34] Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in Neural Information Processing Systems*, 35:16344–16359, 2022. 6
- [35] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. *Advances in neural information processing systems*, 2009. 6
- [36] 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. *Advances in neural information processing systems*, 30, 2017. 6
- [37] Charlie Nash, Jacob Menick, Sander Dieleman, and Peter W Battaglia. Generating images with sparse representations. *arXiv preprint arXiv:2103.03841*, 2021. 6, 8
- [38] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. *Advances in neural information processing systems*, 29, 2016. 6
- [39] Jonathan Ho, Chitwan Saharia, William Chan, David J Fleet, Mohammad Norouzi, and Tim Salimans. Cascaded diffusion models for high fidelity image generation. *Journal of Machine Learning Research*, 23(47):1–33, 2022. 8
- [40] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022. 8, 10
- [41] Shanghua Gao, Pan Zhou, Ming-Ming Cheng, and Shuicheng Yan. Masked diffusion transformer is a strong image synthesizer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 23164–23173, 2023. 10

A. Model Details

Model	Layers N	Hidden size d	Groups
FlowDCN-S	12	384	6
FlowDCN-B	12	768	12
FlowDCN-L	24	1024	16
FlowDCN-XL	28	1152	16

Table 7: **Details of FlowDCN models.** We follow DiT for the Small (S), Base (B), Large (L) and XLarge (XL) model configurations.

B. Comparisons between FlowCNN and FlowDCN on ImageNet 256×256

The relationship between DCN and common CNN. As Eq. (6) states, DCN introduces a deformable field $\Delta p(x)$ and dynamic weight $w(x)$. When all features shares the same static weight instead of dynamic, and deformable field $\Delta p(x)$ degrades to zeros, DCN degenerates to common CNN. Therefore, in most scenarios, DCN-like architectures are more powerful than common CNNs. Furthermore, the *fix* p_k in Tab. 2c indicates that we freeze the p_k (not the deformable field $\Delta p(x)$) and initialize it with a predefined grid.

Why not try a common CNN architecture. In many computer vision tasks, traditional CNNs have been outperformed by transformers, so we opted to explore the modern, advanced CNN variant, Deformable Convolutional Networks (DCN). Additionally, we conducted a small experiment where we replaced the DCN block in FlowDCN with standard 3x3 and 5x5 group-wise convolution blocks.

Model	layers	groups	channels	Params (M)	FID	sFID	IS
SiT-S/2	12	6	384	33.0	57.64	9.05	24.78
FlowCNN-3x3	12	8	512	49.1	59.0	10.7	27.4
FlowCNN-5x5	12	6	384	33.1	63.0	10.9	23.6
FlowDCN-S/2	12	6	384	30.3	54.6	8.8	26.4

Table 8: **Image generation metrics comparisons between SiT, FlowDCN and FlowCNN under 400k training steps budgets.**

Method	256×256 (1:1)			320×320 (1:1)			224×448 (1:2)			160×480 (1:3)		
	FID↓	sFID↓	IS↑	FID↓	sFID↓	IS↑	FID↓	sFID↓	IS↑	FID↓	sFID↓	IS↑
DiT-B	44.83	8.49	32.05	95.47	108.68	18.38	109.1	110.71	14.00	143.8	122.81	8.93
DiT-B + EI	44.83	8.49	32.05	81.48	62.25	20.97	133.2	72.53	11.11	160.4	93.91	7.30
DiT-B + PI	44.83	8.49	32.05	72.47	54.02	24.15	133.4	70.29	11.73	156.5	93.80	7.80
FiT-B	36.36	11.08	40.69	61.35	30.71	31.01	44.67	24.09	37.1	56.81	22.07	25.25
FiT-B + VisionYaRN	36.36	11.08	40.69	44.76	38.04	44.70	41.92	42.79	45.87	62.84	44.82	27.84
FiT-B + VisionNTK	36.36	11.08	40.69	57.31	31.31	33.97	43.84	26.25	39.22	56.76	24.18	26.40
FlowDCN-B	28.5	6.09	51	34.4	27.2	52.2	71.7	62.0	23.7	211	111	5.83
FlowDCN-B (+VAR)	23.6	7.72	62.8	29.1	15.8	69.5	31.4	17.0	62.4	44.7	17.8	35.8
+ $S_{\max}$ Adjust	23.6	7.72	62.8	30.7	19.4	68.5	37.8	22.8	54.4	53.3	22.6	31.5

Table 9: **Benchmarking resolution extrapolations on ImageNet with various aspect ratio training.** VAR indicates various aspect ratios training. We follow the same evaluation pipeline of FiT without using CFG.

C. Resolution Extension with Various Aspect Ratios Training

While FlowDCN, trained on fixed-resolution images, is capable of generating images of arbitrary resolution within a reasonable aspect ratio range, its performance can be improved by adopting variable aspect ratio (VAR) training instead of a fixed 256x256 resolution. To ensure a fair comparison with FiT, which inherently uses VAR, we train a FlowDCN-B/2 model from scratch using VAR techniques. We evaluate our model using the same pipeline and reference batch as FiT, without CFG.

NeurIPS Paper Checklist

1. Claims

Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

Answer: [Yes]

Justification:

Guidelines:

- The answer NA means that the abstract and introduction do not include the claims made in the paper.
- The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
- The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
- It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. Limitations

Question: Does the paper discuss the limitations of the work performed by the authors?

Answer: [Yes]

Justification:

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. Theory Assumptions and Proofs

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [NA]

Justification: not include theoretical results

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. Experimental Result Reproducibility

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification:

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general, releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
 - (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
 - (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
 - (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
 - (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. Open access to data and code

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [No]

Justification: We plan to opensource our code and implementation later.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so “No” is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. Experimental Setting/Details

Question: Does the paper specify all the training and test details (e.g., data splits, hyper-parameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification:

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. Experiment Statistical Significance

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [No]

Justification: Running experiments demands a lot of resources and time.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.

- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. Experiments Compute Resources

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification:

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. Code Of Ethics

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics <https://neurips.cc/public/EthicsGuidelines?>

Answer: [Yes]

Justification:

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. Broader Impacts

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification:

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to

generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.

- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. Safeguards

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: Current models are only trained on small datasets.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. Licenses for existing assets

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification:

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, paperswithcode.com/datasets has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. New Assets

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification:

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. Crowdsourcing and Research with Human Subjects

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification:

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification:

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.