



Scalable Diffusion Models with Transformers



DiTs(Diffusion Transformers)

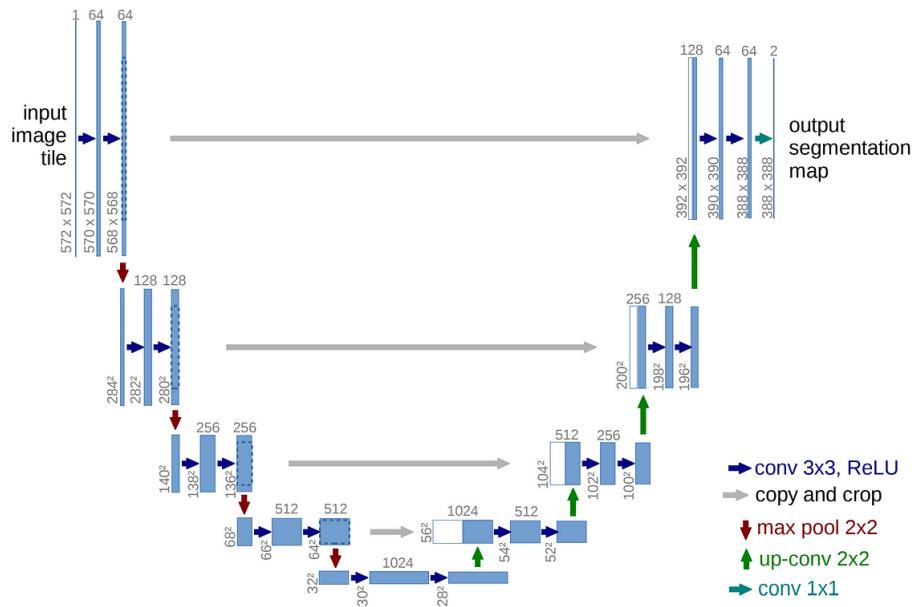
- Transformer 아키텍쳐 기반 diffusion 모델에 대해 연구
 - ↳ 일반적으로 활용되던 U-Net backbone → Transformer

1. Introduction

- 해당 연구는 **트랜스포머**를 기반으로 한 새로운 확산 모델(diffusion model) 클래스에 초점을 맞추고 있음
⇒
Diffusion Transformers(DiT)
 - 이미지 생성에 있어 기존의 U-Net 백본을 대체
- 연구 결과
 - U-Net의 설계 편향이 확산 모델의 성능에 큰 영향을 미치지 않으며, 표준 디자인인 **트랜스포머**로 쉽게 대체될 수 있음을 입증
 - 이미지 생성 영역으로 확장 가능하며, 네트워크 복잡성과 샘플 품질 간의 강력한 상관 관계를 가지며, 우수한 성능을 보임

▼ U-Net

<https://wikidocs.net/148870>



- 이미지 세그멘테이션을 위한 딥러닝 아키텍처로, 인코더와 디코더로 구성됨
 - 인코더: 입력 이미지를 다운샘플링하고 특성을 추출
 - 디코더: 추출된 특성을 이용하여 세그멘테이션 맵을 생성
- U-Net은 작은 파라미터 수로도 효과적인 성능을 보이며, 데이터가 부족한 상황에서도 일반화 성능이 우수함
 - 특히 객체의 경계를 정확하게 분할하는 데 강점을 가지고 있으며, 의료 영상 분석과 같은 다양한 분야에서 사용됨

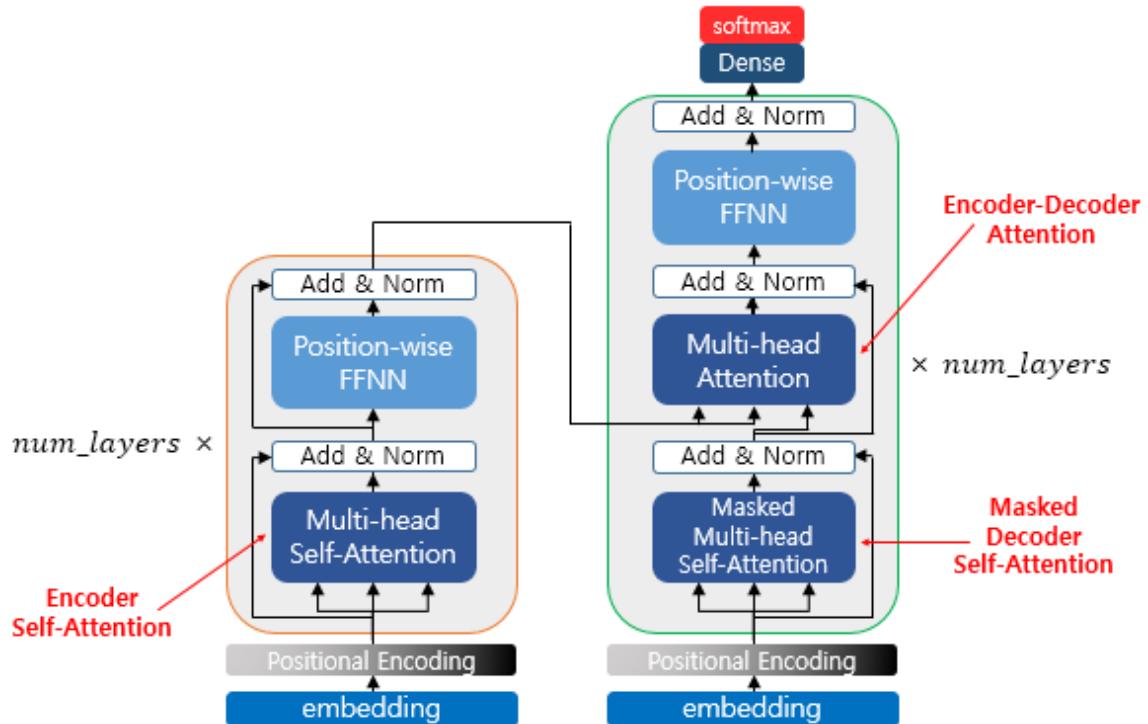
2. Related Work

Transformers

- 언어, 비전, 강화 학습 및 메타 학습과 같은 영역별 아키텍처를 대체해 왔음
 - 언어 영역
 - 모델 크기, 훈련 컴퓨팅 및 데이터를 점점 더 증가시킴으로써 놀라운 스케일링 특성을 보였음
 - 일반적인 자기 회귀 모델 및 ViTs로도 사용되었음
 - 픽셀을 자기 회귀적으로 예측하는 데 사용되었으며, 이산 코드북에서도 자기 회귀 모델 및 가려진 생성 모델로 훈련되었음
 - 또한, DDPMs에서 비공간 데이터를 합성하기 위해 탐색되었음

⇒ 해당 논문에서는 이미지 확산 모델의 backbone으로 사용될 때의 transformer의 스케일링 특성을 연구

▼ Transformers



<https://wikidocs.net/31379>

- 딥러닝 모델 아키텍처로, 주로 자연어 처리와 시퀀스 관련 작업에 사용됨
 - self-attention 메커니즘을 기반으로 하며, 입력 시퀀스의 모든 요소가 서로 상호 작용할 수 있음
 - 인코더-디코더 구조를 사용하여, 입력 시퀀스를 처리하여 출력 시퀀스를 생성
 - 또한 positional encoding을 사용하여 단어의 순서 정보를 전달하고, multi-head attention을 통해 다양한 관점에서 입력을 바라볼 수 있음
- 자연어 처리뿐만 아니라 이미지 생성, 비전 작업 등 다양한 분야에서도 사용되고 있음
 - 대규모 데이터와 계산 가능성의 증가로 인해 많은 관심을 받고 있으며, 많은 작업에서 우수한 성능을 보여주고 있음

Denoising Diffusion Probabilistic Models(DDPMs)

- 확산 모델과 점수 기반 생성 모델은 이미지 생성 분야에서 탁월한 성과를 보이며, 기존의 SOTA였던 생성적 적대 신경망(**GANs**)을 능가하는 경우가 많았음
- 특히, 최근의 확산 모델 개발은 향상된 샘플링 기술과 함께 이루어졌음
 - 이들 모델에는 일반적으로 **convolutional U-Net**이 백본 아키텍처로 사용되었음
- 최근의 연구에서는 어텐션 기반의 새로운 효율적인 아키텍처가 도입되었음

Architecture complexity

- 이미지 생성 분야에서는 보통 아키텍처의 복잡성을 매개변수 수로 평가
 - 그러한 경우 이미지 해상도와 같은 다른 요소들을 고려하지 못하는 한계가 있음
- 해당 논문에서는 이론적인 **Gflops**를 사용하여 모델 복잡성을 분석
 - 복잡성의 최적 메트릭은 여전히 논란 중이며, 응용 프로그램에 따라 다를 수 있음

▼ Gflops

- Gflops는 초당 수행되는 10억 개의 부동 소수점 연산을 나타내는 지표로, 컴퓨터의 성능을 측정하는 중요한 값임
- 주로 CPU 또는 GPU와 같은 프로세서의 성능을 나타내는데 사용되며, 높은 Gflops 값은 더 빠른 연산 속도와 높은 성능을 의미
- 특히, 딥러닝과 같이 대규모 데이터 처리나 복잡한 연산이 필요한 작업에서 중요하게 사용됨

3. Diffusion Transformers

3-1. Preliminaries

Diffusion formulation

- Diffusion model은 실제 데이터 x_0 에 점진적으로 noise를 적용하는 **forward process**를 가정
 - 실제 데이터에 점진적으로 노이즈를 추가하는 과정

$$q(x_t|x_0) = \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t}x_0, (1 - \bar{\alpha}_t)I)$$

- 재매개변수화(reparameterization) 과정을 통해 표준화

$$x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1 - \bar{\alpha}_t}\epsilon_t, \quad \epsilon_t \sim \mathcal{N}(0, I)$$

- diffusion model은 이러한 재매개변수화 과정의 역연산을 학습
 - reverse process(forwarding 이전 상태로 되돌리는 작업)
- $p_\theta(x_{t-1}|x_t) = \mathcal{N}(\mu_\theta(x_t), \Sigma_\theta(x_t))$
- 신경망은 p_θ 의 통계를 예측하는 데 사용됨
- reverse process 모델은 x_0 의 log-likelihood의 변동 하한(VLB)으로 학습되며, 이는 다음과 같이 줄일 수 있음

$$\mathcal{L}(\theta) = -p(x_0|x_1) + \sum_t \mathcal{D}_{\text{KL}}(q^*(x_{t-1}|x_t, x_0) \parallel p_\theta(x_{t-1}|x_t))$$

- q^* 와 p_θ 는 모두 Gaussian Dist
 \rightarrow
 D_{KL} 은 두 분포의 평균과 공분산으로 평가 가능
- μ_θ 를 noise 예측 네트워크 ϵ_θ 로 다시 parameterize하면 예측된 noise $\epsilon_\theta(x_t)$ 와 샘플링된 ground-truth Gaussian noise ϵ_t 사이의 단순 평균 제곱 오차를 사용하여 모델 학습 가능

$$\mathcal{L}_{\text{simple}}(\theta) = \|\epsilon_\theta(x_t) - \epsilon_t\|_2^2$$

- 그러나 학습된 reverse process 공분산 Σ_θ 로 diffusion model을 학습하려면 전체 D_{KL} 항을 최적화해야 함
 \Rightarrow ADM의 접근 방식 채택
 - L_{simple} 로 ϵ_θ 를 학습하고 전체 L 로 Σ_θ 를 학습
 - p_θ 가 학습되면 $x_{t_{\max}} \sim N(0, I)$ 을 초기화하고

$x_{t-1} \sim p_\theta(x_{t-1}|x_t)$ 를 샘플링하여 새 이미지 샘플링 가능

Classifier-free guidance

- 클래스 레이블 c 와 같은 추가 정보를 입력으로 사용

- 이 경우 reverse process는 $p_\theta(x_{t-1}|x_t, c)$ 가 되며, 여기서 ϵ_θ 와 Σ_θ 는 c 로 컨디셔닝됨
- 이 설정에서 classifier-free guidance를 사용하여 샘플링 절차가 $\log p(c|x)$ 가 높은 x 를 찾도록 장려할 수 있음
- 베이즈 정리
 - $\log p(c|x) \propto \log p(x|c) - \log p(x)$
 $\Rightarrow \nabla_x \log p(c|x) \propto \nabla_x \log p(x|c) - \nabla_x \log p(x)$
- score function으로의 해석
 - DDPM 샘플링 절차는 다음과 같이 $p(x|c)$ 가 높은 샘플 x 로 유도 가능

$$\hat{\epsilon}_\theta(x_t, c) = \epsilon_\theta(x_t, \emptyset) + s \cdot \nabla_x \log p(x|c) \propto \epsilon_\theta(x_t, \emptyset) + s \cdot (\epsilon_\theta(x_t, c) - \epsilon_\theta(x_t, \emptyset))$$
 - 이때 $s > 1$ 은 guidance의 척도
 - $c = \emptyset$ 으로 diffusion model을 평가하는 것
 \Rightarrow 학습 중에 c 를 임의로 삭제하고 학습된 “null” 임베딩 \emptyset 으로 대체하여 수행
- Classifier-free guidance는 일반 샘플링 기술에 비해 상당히 개선된 샘플을 생성하는 것으로 널리 알려져 있으며 이러한 추세는 DiT 모델에도 적용됨

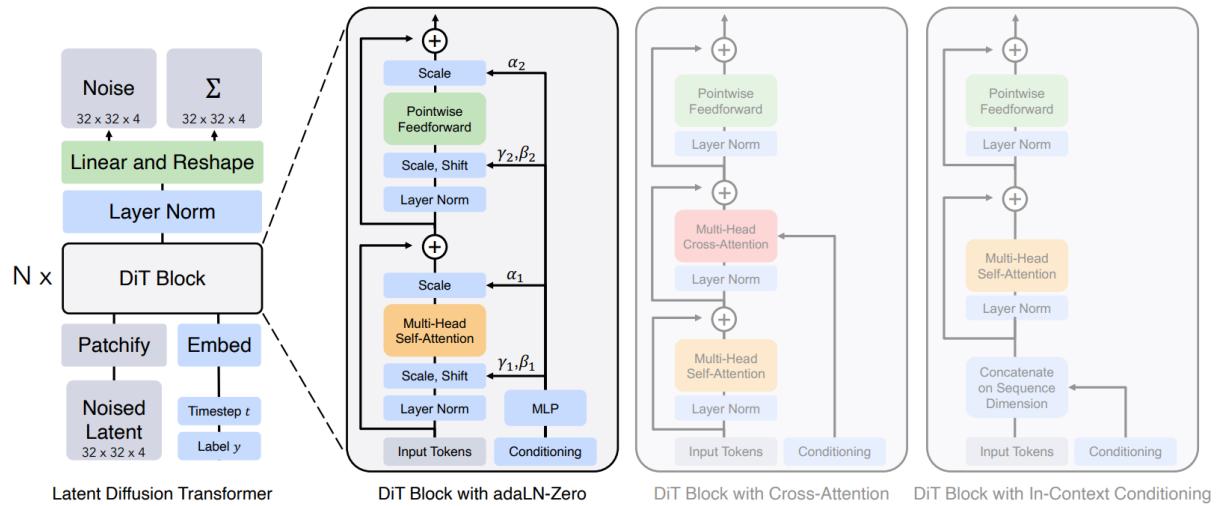
Latent diffusion models

- Latent diffusion model(LDM)은 2단계 접근 방식으로 고해상도 픽셀 space에서 직접 diffusion model을 학습하는 것의 계산 복잡도를 계산하고자 하였음
 1. 학습된 인코더 E 를 사용하여 이미지를 더 작은 space의 표현으로 압축하는 오토인코더를 학습
 2. 이미지 x 의 diffusion model 대신 표현 $z = E(x)$ 의 diffusion model을 학습(E 는 고정)
- 이후 diffusion model에서 표현 z 를 샘플링하고 학습된 디코더 $x = D(z)$ 를 사용하여 이미지로 디코딩하여 새 이미지를 생성
- LDM은 ADM과 같은 픽셀 space diffusion model의 Gflops의 일부를 사용하면서 우수한 성능을 달성
- 본 논문에서는 latent space에 DiT를 적용

- 기존 convolution VAE와 Transformer 기반 DDPM을 사용하여 이미지 생성 파이프라인 생성

3-2. Diffusion Transformer Design Space

Overview

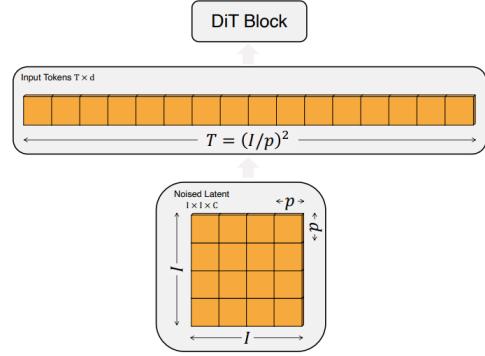


목표

- 스케일링 속성을 유지하기 위해 가능한 한 표준 Transformer 아키텍처에 충실하는 것
- 이미지의 DDPM을 학습하는 것에 초점
→ 일련의 패치에서 작동하는 ViT 아키텍처 기반

Patchify

- 입력에 각 패치를 선형으로 삽입하여 입력을 차원 d 의 T 개의 토큰 시퀀스로 변환
 - 이후 표준 ViT의 주파수 기반 위치 임베딩(사인-코사인 버전)을 모든 입력 토큰에 적용
- patchify에 의해 생성된 토큰 T 의 수는 패치 크기 hyperparameter p 에 의해 결정됨



- p 를 절반으로 줄이면 T 가 4배가 되고, 따라서 전체 Transformer의 Gflops는 적어도 4배가 됨

DiT block design

- noise가 있는 이미지 입력 외에도 diffusion model은 때때로 timestep t , 클래스 레이블 c , 자연어 등과 같은 추가 조건부 정보를 처리
- 저자들은 조건부 입력을 다르게 처리하는 다음과 같은 Transformer block의 4가지 변형에 대해 연구하였음

1. In-context conditioning

- 단순히 t 와 c 의 벡터 임베딩을 입력 시퀀스에 두 개의 추가 토큰으로 추가하여 이미지 토큰과 다르지 않게 취급
 - ViT의 cls 토큰과 유사하며, 수정 없이 표준 ViT 블록을 사용할 수 있음
- 마지막 블록 이후 시퀀스에서 컨디셔닝 토큰을 제거
 - 무시할 수 있는 새로운 Gflops를 모델에 도입

2. Cross-attention block

- t 와 c 의 임베딩을 이미지 토큰 시퀀스와 별도로 길이가 2인 시퀀스로 concat
- Transformer block을 수정
 - multi-head self-attention block 다음에 추가 multi-head cross-attention layer를 포함하도록
 - LDM에서 클래스 레이블로 컨디셔닝하는 데 사용하는 것과 유사
- Cross-attention은 대략 15%의 오버헤드로 모델에 가장 많은 Gflops를 추가

3. Adaptive layer norm(adaLN) block

- transformer block의 표준 레이어
→ adaptive layer norm(adaLN)

- 차원별 scaling 및 shift 파라미터 γ 와 β 를 직접 학습하는 대신 t 와 c 의 임베딩 벡터 합계에서 회귀
- 세 가지 블록 디자인 중 adaLN은 최소한의 Gflops를 추가하므로 가장 컴퓨팅 효율적
- 또한 모든 토큰에 동일한 feature를 적용하도록 제한되는 유일한 컨디셔닝 메커니즘

4. adaLN-Zero block

- ResNet에 대한 이전 연구들에서는 각 residual block을 항등 함수로 초기화 하는 것이 유익하다는 것을 발견하였음
 - Diffusion U-Net 모델은 유사한 초기화 전략을 사용하여 residual 연결 전에 각 블록의 최종 convolution layer를 0으로 초기화
- γ 와 β 를 회귀하는 것 외에도 DiT 블록 내의 residual 연결 직전에 적용되는 차원별 scaling 파라미터 α 도 회귀
 - 모든 α 에 대해 영벡터를 출력하도록 MLP를 초기화
⇒ 전체 DiT 블록을 항등 함수로 초기화
- adaLN 블록과 마찬가지로 adaLNZero는 무시할 수 있는 Gflops를 모델에 추가

Model size

Model	Layers N	Hidden size d	Heads	Gflops ($I=32, p=4$)
DiT-S	12	384	6	1.4
DiT-B	12	768	12	5.6
DiT-L	24	1024	16	19.7
DiT-XL	28	1152	16	29.1

- hidden dimension 크기 d 에서 각각 작동하는 일련의 N 개의 DiT 블록 적용
- ViT에 이어 $N, d, \text{attention head}$ 를 공동으로 확장하는 표준 Transformer 구성을 사용
 - 특히 DiT-S, DiT-B, DiT-L, DiT-XL의 네 가지 구성을 사용
- 0.3에서 118.6 Gflops까지 다양한 모델 크기와 flops 할당을 다룸

→ 확장 성능 측정 가능

Transformer Decoder

- 최종 DiT 블록 이후에는 이미지 토큰 시퀀스를 출력 noise 예측과 출력 대각 공분산 예측으로 디코딩
 - 이를 위해 표준 선형 디코더를 사용하여 각 토큰을 $p \times p \times 2C$ 텐서로 선형 디코딩
- 마지막으로 예측된 noise와 공분산을 얻기 위해 디코딩된 토큰을 원래 공간적 레이아웃으로 재정렬

4. Experimental Setup

데이터셋

- ImageNet(256×256, 512×512)

학습

- learning rate: 1×10^{-4} (warm-up은 사용 x)
- optimizer: AdamW
- weight decay 사용 x
- batch size: 256
- data augmentation: horizontal flip
- EMA decay: 0.9999

Diffusion

- VAE: Stable Diffusion의 VAE 사용(downsample factor 8)
- t_max=1000
- Noise schedule: 선형, 1×10^{-4} 에서 2×10^{-2}

5. Experiments

DiT block design

- 4가지 블록 디자인 변형을 통해 Gflop **DiT-XL/2** 모델을 훈련
 - in-context(119.4 Gflops)

- cross-attention(137.6 Gflops)
- adaptive layer norm(adaLN, 118.6 Gflops)
- adaLN-zero(118.6 Gflops)
- 훈련 중 FID 측정

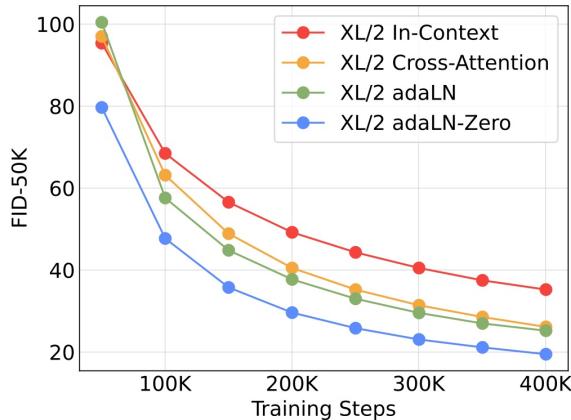
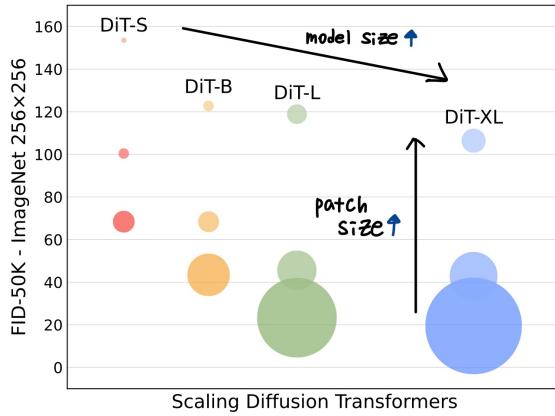


Figure 5. **Comparing different conditioning strategies.** adaLN-Zero outperforms cross-attention and in-context conditioning at all stages of training.

- adaLN-Zero 블록은 가장 효율적인 계산 방식으로 in-context 및 cross-attention에 비해 낮은 FID를 보여주었음
 - 40만 번의 훈련 반복에서 adaLN-Zero 모델의 FID는 in-context 모델의 거의 절반 수준이었음
→ 조건 메커니즘이 모델 품질에 중요한 영향을 미친다는 것을 보여줌
 - 초기화도 중요
 - 각 DiT 블록을 항등 함수로 초기화하는 adaLN-Zero가 일반적인 adaLN을 크게 능가하였음
- ⇒ 이후 모든 모델은 adaLN-Zero DiT 블록을 사용

Scaling model size and patch size

- 모델 설정(S, B, L, XL)과 패치 크기(8, 4, 2)를 바꿔가며 12개의 DiT 모델을 훈련시킴



- 트랜스포머를 더 깊고 넓게 만들면 훈련의 모든 단계에서 상당한 FID 개선이 이루어짐
- 또한, DiT가 처리하는 토큰 수를 단순히 확장하여 파라미터를 대략 고정시킴으로써 훈련 중에 상당한 FID 개선이 관찰됨

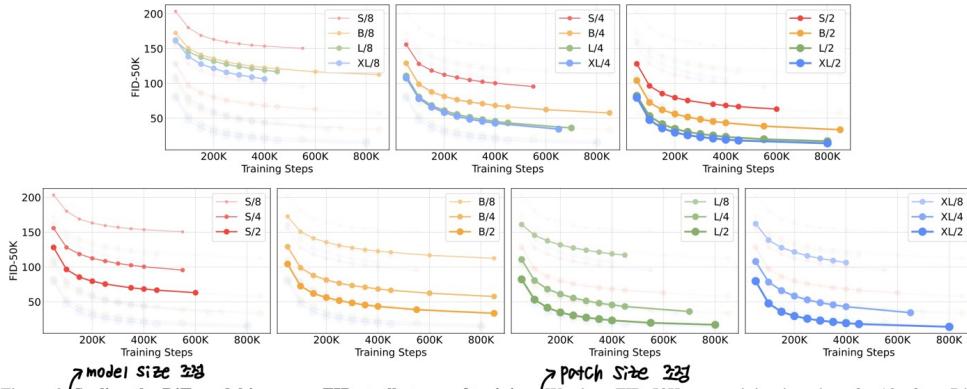


Figure 6. *Scaling the DiT model improves FID at all stages of training.* We show FID-50K over training iterations for 12 of our DiT models. *Top row:* We compare FID holding patch size constant. *Bottom row:* We compare FID holding model size constant. Scaling the transformer backbone yields better generative models across all model sizes and patch sizes.

DiT Gflops are critical to improving performance

- 매개변수 개수는 DiT 모델의 품질에 결정적인 영향을 미치지는 x
 - 모델 크기를 일정하게 유지하고 패치 크기를 줄일 때 트랜스포머의 총 매개변수는 사실상 변경되지 않으며(실제로는 약간 감소하기는 하는데..뭐..), Gflops만 증가
- ⇒ 모델 Gflops의 확장이 성능 향상의 핵심 요소임
- 다양한 DiT 구성 중 총 Gflops가 유사한 경우(ex. DiT-S/2와 DiT-B/4) 유사한 FID 값을 얻는다는 것을 입증

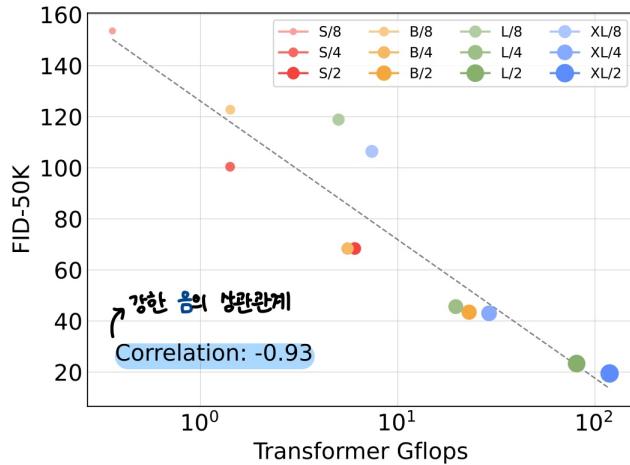


Figure 8. Transformer Gflops are strongly correlated with FID.
We plot the Gflops of each of our DiT models and each model's FID-50K after 400K training steps.

- 모델 Gflops와 FID-50K 사이에 강한 음의 상관 관계가 있으며, 추가 모델 계산이 개선된 DiT 모델에 중요한 요소임을 시사
- 또한, 이러한 경향이 Inception score와 같은 다른 metric에서도 유지되는 것을 관찰함

Large DiT models are more compute-efficient

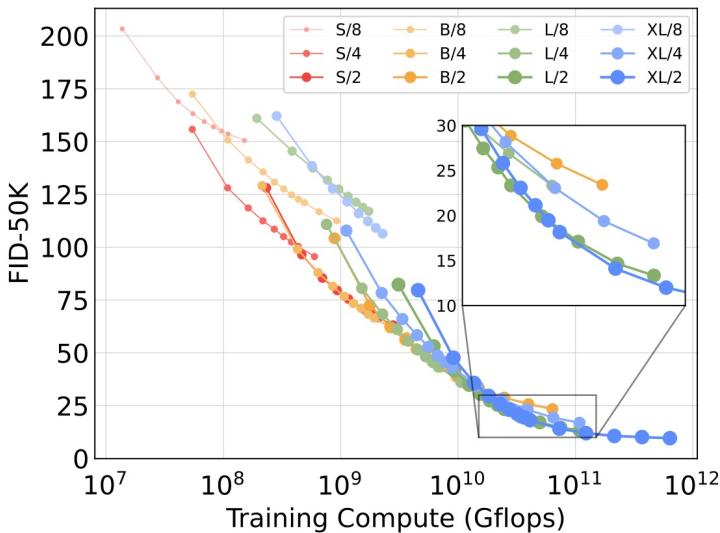


Figure 9. Larger DiT models use large compute more efficiently. We plot FID as a function of total training compute.

- 모든 DiT 모델의 총 훈련 계산에 따른 FID를 플롯
 - 훈련 계산을 모델 Gflops × 배치 크기 × 훈련 단계 × 3으로 추정

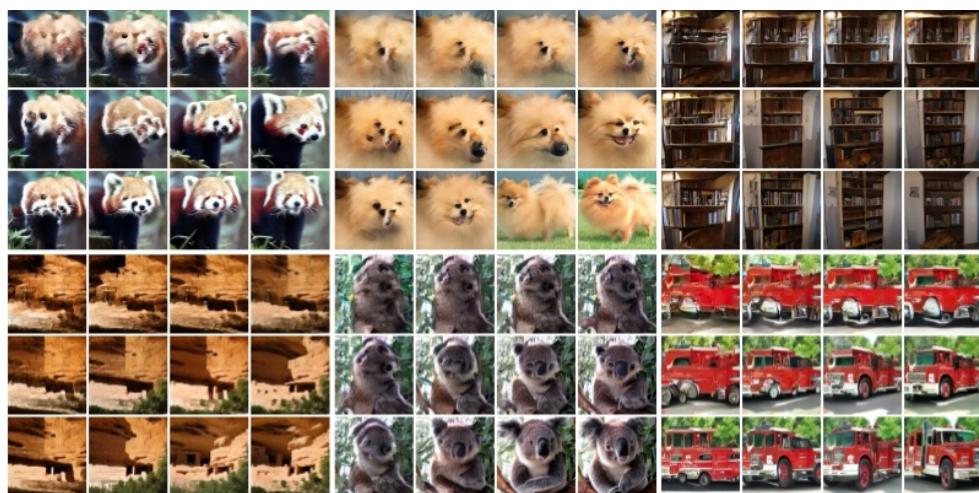
- 왜 3?

→ 역전파가 순방향 패스보다 두 배 계산 부하가 크다고 가정

⇒ 작은 DiT 모델은 긴 훈련 시간에도 불구하고, 더 적은 단계로 훈련된 큰 DiT 모델에 비해 계산 비효율적임

Visualizing scaling

- 모델 크기와 토큰 수를 함께 스케일링하면 시각적 품질이 현저히 향상되는 것을 확인할 수 있음



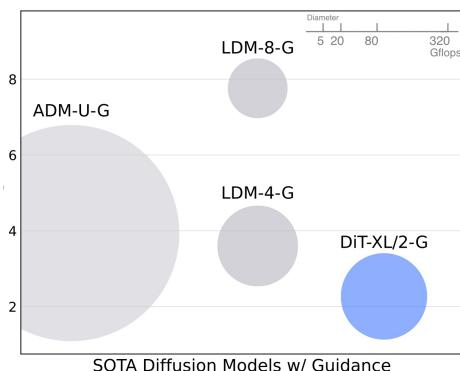
5-1. State-of-the-Art Diffusion Models

256 x 256 ImageNet

Class-Conditional ImageNet 256×256					
Model	FID↓	sFID↓	IS↑	Precision↑	Recall↑
BigGAN-deep [2]	6.95	7.36	171.4	0.87	0.28
StyleGAN-XL [53]	2.30	4.02	265.12	0.78	0.53
ADM [9]	10.94	6.02	100.98	0.69	0.63
ADM-U	7.49	5.13	127.49	0.72	0.63
ADM-G	4.59	5.25	186.70	0.82	0.52
ADM-G, ADM-U	3.94	6.14	215.84	0.83	0.53
CDM [20]	4.88	-	158.71	-	-
LDM-8 [48]	15.51	-	79.03	0.65	0.63
LDM-8-G	7.76	-	209.52	0.84	0.35
LDM-4	10.56	-	103.49	0.71	0.62
LDM-4-G (cfg=1.25)	3.95	-	178.22	0.81	0.55
LDM-4-G (cfg=1.50)	<u>3.60</u>	-	247.67	0.87	0.48
DiT-XL/2	9.62	6.85	121.50	0.67	0.67
DiT-XL/2-G (cfg=1.25)	3.22	5.28	201.77	0.76	0.62
DiT-XL/2-G (cfg=1.50)	2.27	4.60	278.24	0.83	0.57

Table 2. Benchmarking class-conditional image generation on ImageNet 256×256. DiT-XL/2 achieves state-of-the-art FID.

- 분류기 없는 가이드라인을 사용할 때, DiT-XL/2는 모든 이전 확산 모델을 능가하며 이전 최고의 FID-50K인 LDM의 3.60을 2.27로 낮춤
- DiT-XL/2(118.6 Gflops)가 잠재 공간 U-Net 모델인 LDM-4 (103.6 Gflops)보다 계산 효율적이며, ADM(1120 Gflops)나 ADM-U(742 Gflops)와 같은 픽셀 공간 U-Net 모델보다 상당히 더 효율적임



- 또한 DiT-XL/2가 LDM-4 및 LDM-8과 비교하여 모든 테스트된 분류기 없는 가이드 스케일에서 더 높은 recall 값을 달성한다는 것을 관찰

512 x 512 ImageNet

- 동일한 하이퍼파라미터를 사용하여 3백만 번의 반복 동안 ImageNet에서 512x512 해상도로 새로운 DiT-XL/2 모델을 훈련시킴

Class-Conditional ImageNet 512×512					
Model	FID↓	sFID↓	IS↑	Precision↑	Recall↑
BigGAN-deep [2]	8.43	8.13	177.90	0.88	0.29
StyleGAN-XL [53]	2.41	4.06	267.75	0.77	0.52
ADM [9]	23.24	10.19	58.06	0.73	0.60
ADM-U	9.96	5.62	121.78	0.75	0.64
ADM-G	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	12.03	7.12	105.25	0.75	0.64
DiT-XL/2-G (cfg=1.25)	4.64	5.77	174.77	0.81	0.57
DiT-XL/2-G (cfg=1.50)	3.04	5.02	240.82	0.84	0.54

Table 3. Benchmarking class-conditional image generation on ImageNet 512×512. Note that prior work [9] measures Precision and Recall using 1000 real samples for 512 × 512 resolution; for consistency, we do the same.

- 이전의 모든 확산 모델을 능가하여, 이전의 최고 FID를 개선하는 성과를 거뒀음
- 심지어 토큰의 수가 늘어난 상황에서도 계산 효율적임

5-2. Scaling Model vs Sampling Compute

- 더 많은 샘플링 계산을 사용하여 작은 모델 계산 DiT가 더 큰 모델을 능가할 수 있는지 여부를 탐구
 - 모든 12개의 DiT 모델에 대해 40만 번의 훈련 단계 후에 이미지 당 [16, 32, 64, 128, 256, 1000]개의 샘플링 단계를 사용하여 FID를 계산

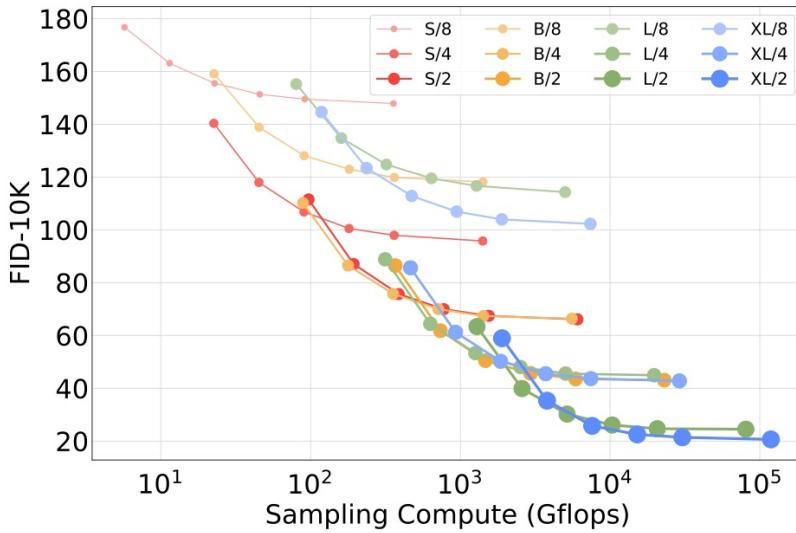


Figure 10. Scaling-up sampling compute does not compensate for a lack of model compute. For each of our DiT models trained for 400K iterations, we compute FID-10K using [16, 32, 64, 128, 256, 1000] sampling steps. For each number of steps, we plot the FID as well as the Gflops used to sample each image. Small models cannot close the performance gap with our large models, even if they sample with more test-time Gflops than the large models.

⇒ 일반적으로, 샘플링 계산을 확장하는 것은 모델 계산 부족을 보상할 수 없음

6. Conclusion

- Diffusion Transformers (DiTs)
 - 이전 U-Net 모델을 능가하며 트랜스포머 모델 클래스의 탁월한 확장 특성을 계승한 확산 모델용 간단한 트랜스포머 기반 백본
- 본 논문에서 유망한 스케일링 결과를 고려할 때, 향후 연구에서는 DiTs를 더 큰 모델과 토큰 수로 확장해 나가는 것이 필요
 - 또한 DiT를 DALL-E 2와 Stable Diffusion과 같은 텍스트 to 이미지 모델의 백본으로 탐구할 수 있음