

# DaLL-E

## Zero-Shot Text-to-Image Generation

### Abstract

텍스트 to 이미지 생성은 전통적으로 더 좋은 모델 가정을 찾는 것에 집중했다. 이러한 가정은 복잡한 구조, 보조 로스 또는 추가정보를 수반합니다. 우리는 autoregressive의 transformer 기반 방식을 소개합니다. 충분한 데이터를 통해 우리의 접근은 zero-shot에 대해서도 효과적입니다.

### Introduction

우리는 120억개의 파라미터 autoregressive transformer를 2억 5천만개 이미지 텍스트 페어 데이터를 인터넷에서 수집하여 언어를 통해 이미지를 조절할 수 있습니다. MS-COCO dataset의 학습 라벨을 사용하지 않고 평가를 진행했습니다. 그리고 이전 다른 모델의 결과물 보다 90%넘게 더 좋은 결과를 얻었다고 사람들에 의해 선택되었습니다. 우리는 또한 기초 수준의 이미지 to 이미지 번역 또한 가능합니다.



(a) a tapir made of accordion. (b) an illustration of a baby tapir with the texture of an hedgehog in a christmas sweater walking a dog (c) a neon sign that reads "backprop". a neon sign that reads "backprop". backprop neon sign (d) the exact same cat on the top as a sketch on the bottom

*Figure 2. With varying degrees of reliability, our model appears to be able to combine distinct concepts in plausible ways, create anthropomorphized versions of animals, render text, and perform some types of image-to-image translation.*

### Method

우리의 목표는 텍스트와 이미지 토큰을 하나로 만들어 transformer를 autoregressive하게 학습 시키는 것입니다. 그러나 이미지의 픽셀을 직접적으로 사용하는 것은 많은 자원을 필요로 합니다. likelihood 방식은 인접 픽셀간의 관계를 고려하는 경향이 있어서 디테일한 특징 부분을 포착하는데 비용이 소비됩니다. 우리는 이러한 문제를 해결하기 위해 two-stage 학습 방식을 구성합니다.

## Stage 1



Figure 1. Comparison of original images (top) and reconstructions from the discrete VAE (bottom). The encoder downsamples the spatial resolution by a factor of 8. While details (e.g., the texture of the cat's fur, the writing on the storefront, and the thin lines in the illustration) are sometimes lost or distorted, the main features of the image are still typically recognizable. We use a large vocabulary size of 8192 to mitigate the loss of information.

우리는 discrete variational autoencoder(dVAE)로  $256 * 256$  RGB 이미지를  $32 * 32$  격자 이미지 토큰으로 학습시켜 변환시킵니다. 그리고 각각의 토큰은 8192개의 값을 가질 수 있습니다. 이는 transformer의 context size를 192분의 1로 줄이는 효과를 가져다 줍니다.

## Stage 2

우리는 BPE-encoded text 256 토큰을 1024 이미지 토큰과 연결시키고 autoregressive transformer를 결합확률 분포를 학습시킵니다. 이러한 전반적인 과정은 evidence lower bound(ELB)를 최대화시킨다.

$$\ln p_{\theta, \psi}(x, y) \geq \mathbb{E}_{z \sim q_{\phi}(z | x)} (\ln p_{\theta}(x | y, z) - \beta D_{\text{KL}}(q_{\phi}(y, z | x), p_{\psi}(y, z))), \quad (1)$$

$q$ 는 dVAE encoder에서 생성된 이미지 토큰에 대한 분포를 나타냅니다.

$p_{\theta}$ 는 이미지 토큰에 따라 dVAE decoder에서 생성된 RGB 이미지 분포를 나타냅니다.

$p$ 는 트랜스포머에서 텍스트와 이미지 토큰간의 결합 분포를 나타냅니다.

$\beta$ 가 1일 때로 적었지만, 실제로 더 큰 값이 유용합니다.

## Stage One

첫 번째 학습 스텝에서, 우리는 이미지에 대해서만 dVAE를 학습시킨다. 우리는 초기 prior  $p$ 를 uniform categorical distribution로 설정하여 8192 가지의 값을 갖도록 설정한다. 그리고  $32 * 32$  격자 값에 대하여 이를 따라 8192 값으로 파라미터화 시킨다.

ELB는 이제 최적화하기에 어려워진다.  $q$ 가 이산분포이기에 우리는 기울기를 최대화시키기 위해 다시 파라미터화를 시킬 수 없다. 이러한 문제를 해결하기 위해 online cluster assignment procedure + straight-through estimator를 사용한다. 대신 우리는 gumbel-softmax relaxation을 사용하여  $q$  대신  $q^r$ 를 사용했다. 이는  $r \rightarrow 0$ 이 될수록 불안정해진다. 그리고 likelihood는 log-laplace distribution을 사용하여 평가되어진다. relaxed ELB는 Adam을 사용하여 최대화시킨다.

학습을 안정시키기 위해 우리는 다음과 같은 방식을 사용한다.

1.  $r$ 을 16분의 1로 줄이며 학습을 진행시키면 relaxed validation ELB와 true validation ELB사이에 차이가 가까워진다.
2. 인코더의 끝 부분과 디코더의 앞 부분에  $1*1$  convolution을 사용하여 receptive field 크기를 줄였을 때 true ELB로 더 잘 일반화 한다.
3. 인코더와 디코더의 resblocks에 작은 상수를 곱하면 초기에 안정적으로 학습이 가능하다. 우리는 KL weight  $\beta = 6.6$ 로 증진시키면 학습에서의 오류를 줄일 수 있다.

## Stage Two

우리는  $\varphi$ 와  $\theta$ 를 고정하고 텍스트와 이미지 토큰에 대한 사전 분포를 ELB를 최대화시키기 위해 학습을 진행한다. 이미지와 텍스트 쌍에서 우리는 텍스트를 BPE encode를 통해 16384 vocabulary로 최대 256 tokens를 만든다. 그리고 이미지는 8192 크기의 vocabulary로  $32 \times 32 = 1024$  tokens를 생성한다. 이미지 토큰은 dVAE encoder logits에서 argmax sampling을 통해 얻어진다. 최종적으로 텍스트와 이미지 토큰은 하나의 데이터로 연결된다.

### Zero-Shot Text-to-Image Generation

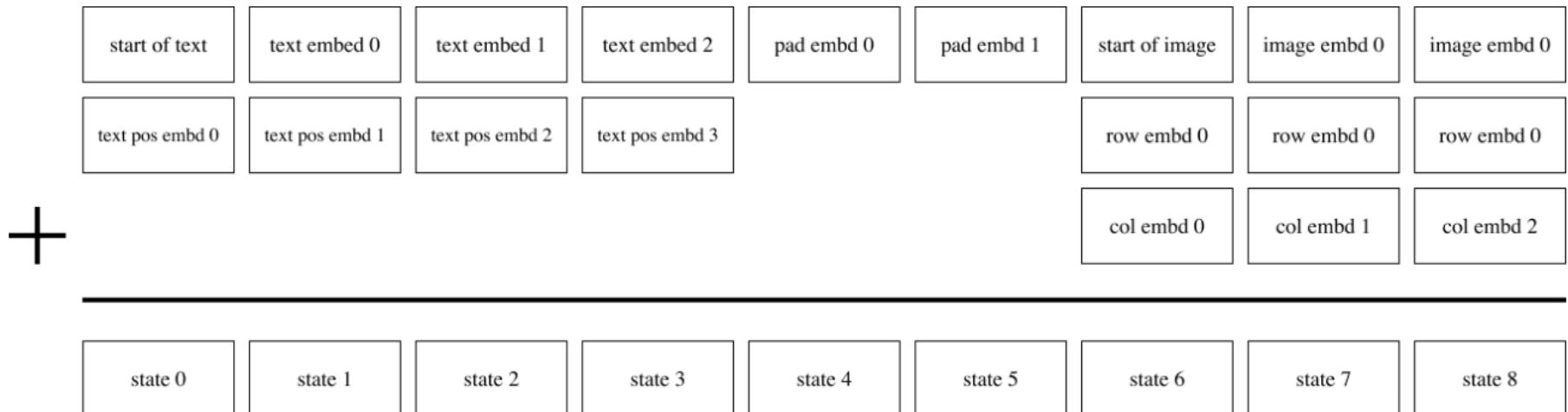


Figure 10. Illustration of the embedding scheme for a hypothetical version of our transformer with a maximum text length of 6 tokens. Each box denotes a vector of size  $d_{\text{model}} = 3968$ . In this illustration, the caption has a length of 4 tokens, so 2 padding tokens are used (as described in Section 2.2). Each image vocabulary embedding is summed with a row and column embedding.

트랜스포머는 64개의 self-attention layers에서 이미지 토큰이 모든 텍스트 토큰을 참조하는 디코더만 있는 모델이다. 모델에서 사용되는 self-attention mask는 총 3가지 종류가 있다.

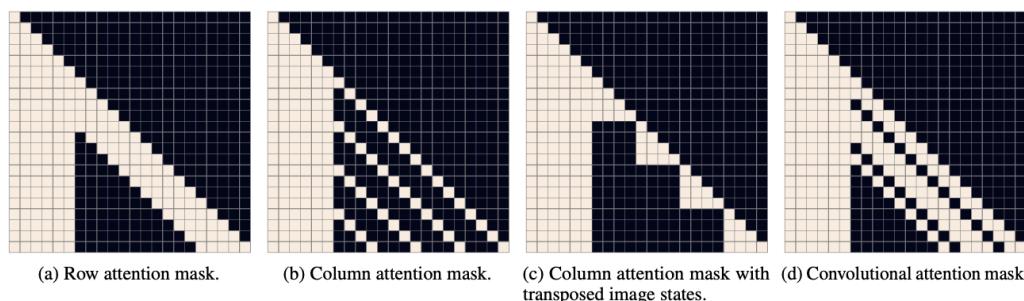


Figure 11. Illustration of the three types of attention masks for a hypothetical version of our transformer with a maximum text length of 6 tokens and image length of 16 tokens (i.e., corresponding to a  $4 \times 4$  grid). Mask (a) corresponds to row attention in which each image token attends to the previous 5 image tokens in raster order. The extent is chosen to be 5, so that the last token being attended to is the one in the same column of the previous row. To obtain better GPU utilization, we transpose the row and column dimensions of the image states when applying column attention, so that we can use mask (c) instead of mask (b). Mask (d) corresponds to a causal convolutional attention pattern with wraparound behavior (similar to the row attention) and a  $3 \times 3$  kernel. Our model uses a mask corresponding to an  $11 \times 11$  kernel.

텍스트 to 텍스트 어텐션에 대해서는 표준의 마스크 그리고 이미지 to 이미지에 대한 어텐션은 row, column, convolution attention 마스크를 사용한다. 우리는 text 토큰을 256개로 제한을 두었다. 그렇기에 텍스트와 이미지 토큰 사이 padding이 필요한 공간에 어떻게 해야하는가에 대해서는 그 사이를  $-\inf$ 를 설정하는 방안이 있다. 그러나 우리는 위치마다 각기 다른 special padding token을 사용해 학습을 진행했다. 이 토큰은 text token이 사용 불가능할 때 사용한다. 이러한 방식은 validation loss를 높게 만들지만 out-of-distribution captions에서는 잘 작동했다.

우리는 텍스트와 이미지 토큰에 대해 각 배치 데이터의 수에 따라 크로스 엔트로피를 정규화시켰다. 우리는 이미지 모델링에 집중했기에 text에는 8분의 1 그리고 이미지에는 8분의 7 loss를 두었다. 목적 함수는 adam + exponentially weighted iterate averaging을 사용한다.

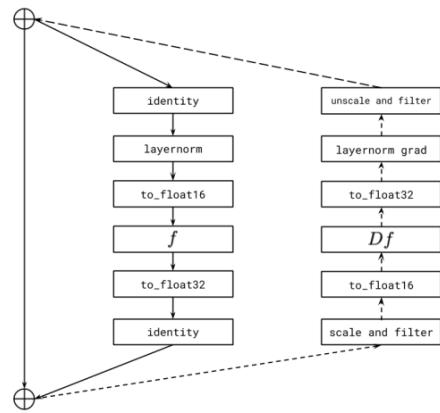
## Data Collection

우리의 12억개의 파라미터에 주된 실험은 MS-COCO에 확장인 330만개의 이미지 텍스트 페어 쌍 데이터인 Conceptual Captions에서 이루어졌다. 파라미터를 120억개로 키우기 위해 우리는 2억 5천만개의 텍스트 이미지 페어 쌍 데이터를 인터넷에서 추출해 생성했다. 이 데이터는 MS-COCO를 포함하지 않지만 Conceptual Captions와 YFCC100M의 일부분은 포함한다. MS-COCO는 나중에 만드렸기 때문에 우리의 학습 데이터는 MS-COCO에서 일부 겹칠 수 있다. 우리는 이를 조절하기 위해 봤지만 딱히 있지는 않았다.

## Mixed-Precision Training

GPU memory를 절약하며 처리량을 늘리기 위해 대부분의 파라미터와, 업데이트, 활성화 함수 등은 16-bit 형식으로 저장된다. 그리고 activation checking point를 저장하고 역전파 과정의 resblocks 내에서 activations를 재연산한다. 16-bit 형식의 10억개의 파라미터를 발산

없이 학습시키는 것이 어려운 부분이었다. 16 bit 기울기에서 학습시키는 것의 문제는 언더플로우 때문으로 보였다. 우리는 이를 해결하기 위해 per-resblock gradient scaling을 사용한다.



*Figure 4. Illustration of per-resblock gradient scaling for a transformer resblock. The solid line indicates the sequence of operations for forward propagation, and the dashed line the sequence of operations for backpropagation. We scale the incoming gradient for each resblock by its gradient scale, and unscale the outgoing gradient before it is added to the sum of the gradients from the successive resblocks. The activations and gradients along the identity path are stored in 32-bit precision. The ‘filter’ operation sets all Inf and NaN values in the activation gradient to zero. Without this, a nonfinite event in the current resblock would cause the gradient scales for all preceding resblocks to unnecessarily drop, thereby resulting in underflow.*

모델의 깊이가 깊어질 수록 기울기 같은 점점 더 작아진다. 그리고 이는 16-bit의 형식으로 구성되어 있어 쉽게 0의 값으로 반올림 되어진다. 이런 현상을 underflow라 말한다. 이를 해결하기 위해 mixed precision을 학습한다. 그러나 mixed precision은 5bit로 구성되어져 있다. 이는 언어 모델에서 충분하지만 text to image에서는 충분하지 않다. 따라서 우리는 모델에 각각의 resblock마다 다른 gradient scale을 사용한다. 위의 그림에서 실선은 순전파 점선은 역전파를 나타낸다. residual block의 기울기는 들어갈 때 scale로 기울기를 키운 후 더해지는 지점으로 갈 때 다시 scaling을 취소하여 더하는 방식이다. 이를 통해 underflow 현상을 극복하고자 한다.

## Distributed Optimization

우리의 120억개의 파라미터를 가진 모델은 24GB를 사용한다. 이는 NVIDIA의 V100을 넘는 수치이다. 이를 해결하기 위해 parameter sharding을 사용한다. 이는 compute-intensive 연산으로 내부 머신 소통간 야기되는 지연을 없애준다.

모델을 학습하기 위한 클러스터에서의 bandwidth의 거리는 같은 GPU 내의 bandwidth 보다 짧다. 이는 gradient 취합 과정의 비용이 절감되어진다. 또한 powerSGD를 사용하여 기울기를 압축시킨다. 이와 같은 과정으로 우리는 아래와 같은 효과를 얻을 수 있다.

역전파 과정에서 기울기를 축적시켜 메모리를 절약한다.

Error buffer를 0으로 만드는 instance를 최소화 시킨다.?

수치의 안전성을 더한다.

Underflow를 회피할 수 있다.

## Sample Generation

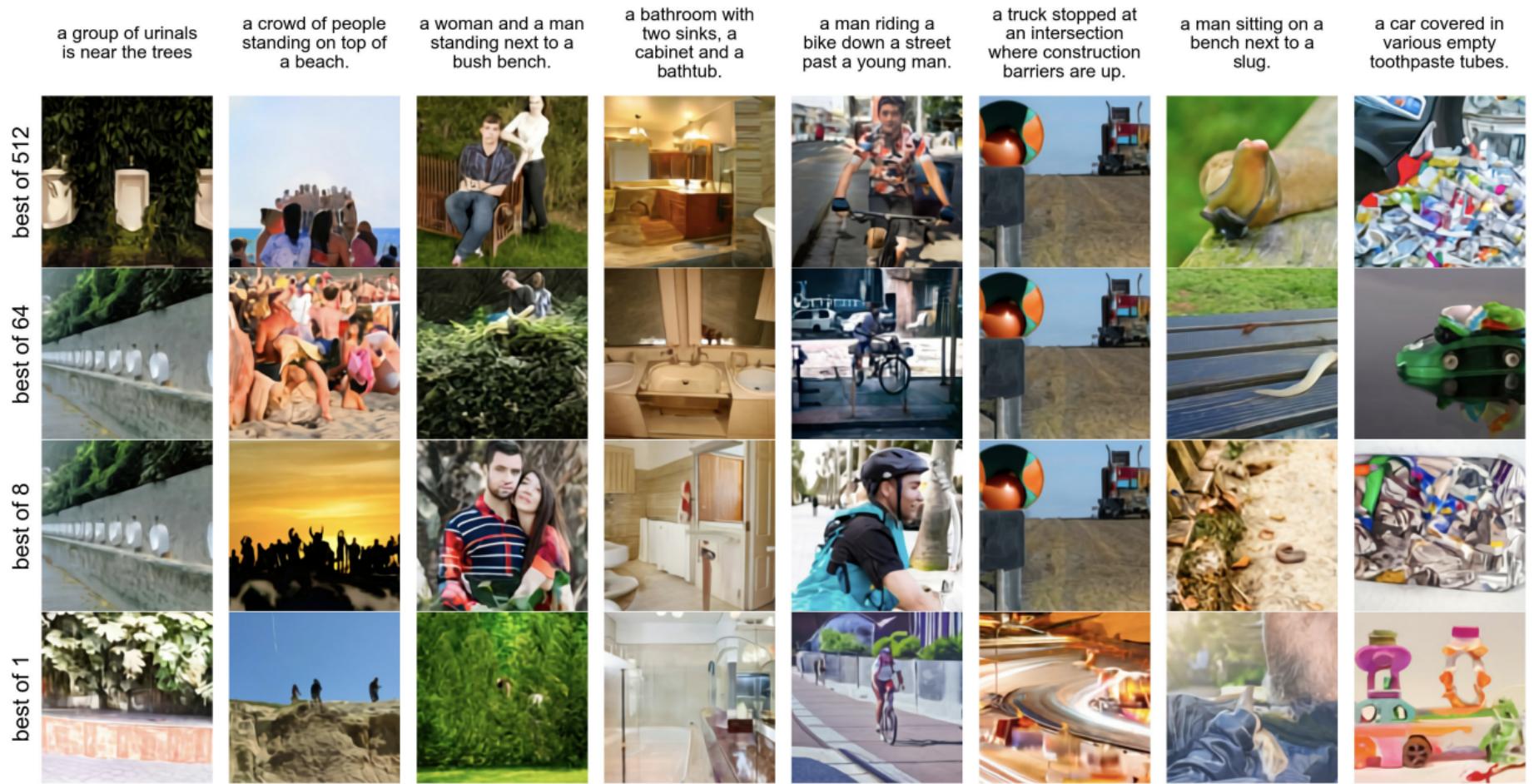


Figure 6. Effect of increasing the number of images for the contrastive reranking procedure on MS-COCO captions.

transformer로부터 나온 결과를 사전 학습된 contrastive model을 사용해 reranking 하여 이미지를 sampling한다.

## Experiments

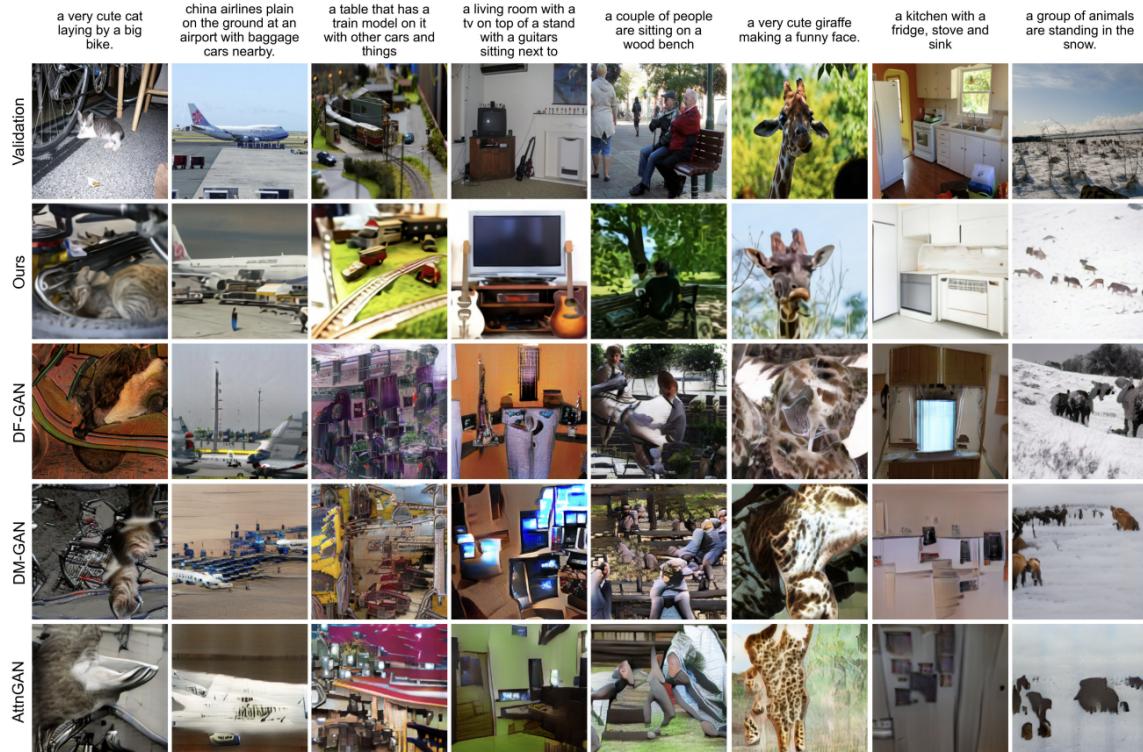
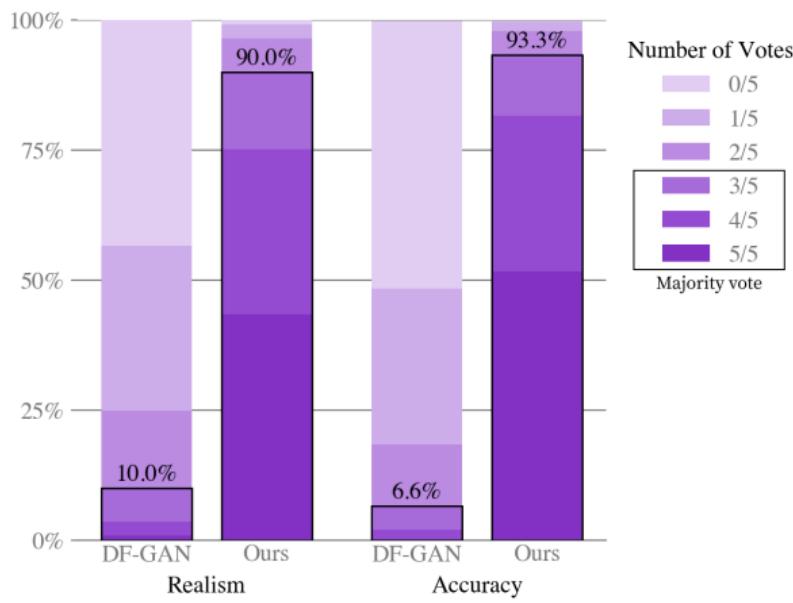


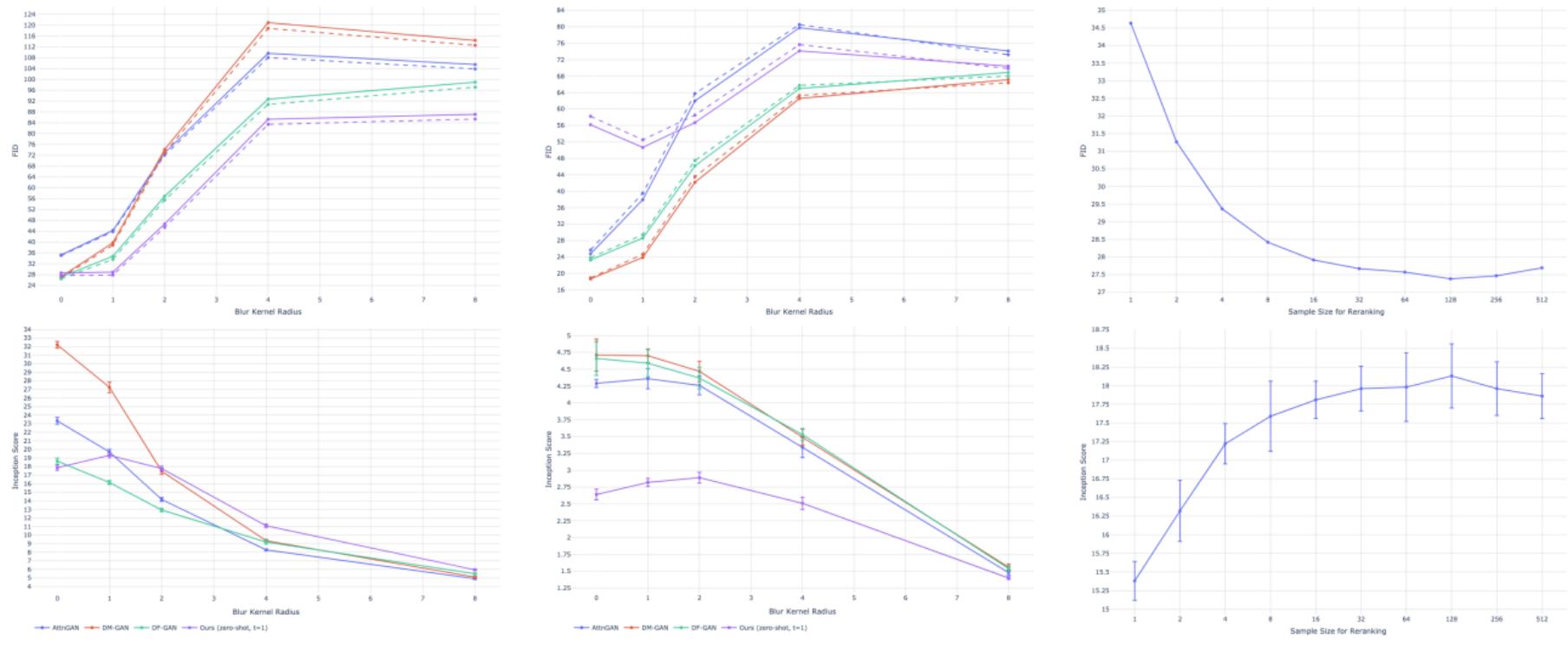
Figure 3. Comparison of samples from our model to those from prior approaches on captions from MS-COCO. Each of our model samples is the best of 512 as ranked by the contrastive model. We do not use any manual cherrypicking with the selection of either the captions or the samples from any of the models.

위 사진은 zero-shot 평가를 다른 모델들과 함께 평가한 것이다.



**Figure 7.** Human evaluation of our model (evaluated zero-shot without temperature reduction) vs prior work (DF-GAN) on captions from MS-COCO. In a best-of-five vote, our model’s sample was chosen as the most realistic 90.0% of the time, and was chosen as the image best matching a shared caption 93.3% of the time.

또한, 사람들의 직접적인 평가로 정성 평가를 진행했고 이는 우리의 모델이 압도적이었다.



**Figure 9.** Quantitative results on MS-COCO and CUB. Solid lines represent FID computed against the original validation sets, and dashed lines represent FID computed against validation sets with overlapping images removed (see Section 3.2). For MS-COCO, we evaluate all models on a subset of 30,000 captions sampled from the validation set. For CUB, we evaluate all models on all of the unique captions in the test set. We compute the FID and IS using the DM-GAN code, which is available at <https://github.com/MinfengZhu/DM-GAN>.

우리의 모델은 dVAE에서 차원을 감소시켜 진행하여 디테일한 정보를 놓치는 경우가 있다. 그래서 이를 알아보기 위해 validation data에 gaussian noise를 주어 나타나는 결과를 살펴 본다. 가우시안 노이즈를 주면 디테일한 정보들이 정답에서 삭제되기 때문에 효과를 볼려고 하는 것 같다. 또한 sampling의 개수를 늘리시 성능이 올라가는 것을 확인할 수 있다.

그리고 달리는 우리가 생각하지 못한 방식 또한 일반화 시킴을 볼 수 있다. 예를 들어 “a tapir made of accordion...”을 넣는다면 아코디언으로 만든 맥(동물)이 나온다. 이는 높은 단계의 추상적인 이미지를 만드는 능력이 있음을 보입니다.



## Conclusion

우리는 autoregressive model에 기반한 text to image의 간단한 방식을 살펴봤다. 우리는 도메인 특화나 zero-shot이나 다양한 곳에서 큰 규모의 학습이 일반화를 잘 해주는 것을 보였다.