DALLE 2

A Preprint

1 Main

1.1 General

Training dataset consists of pairs (x; y) where x is image and y is a corresponding captions. Given an image x, z_i, z_t - CLIP image and text embbedings respectively. Authors suggest method of generating image from captions using 2 stages:

- a prior $P(z_i|y)$ that produce CLIP image embeddings z_i conditioned on captions y
- A decoder $P(x|z_i, y)$ that produces images x conditioned on CLIP image embeddings z_i (optionally captions y)

The decoder allows us to invert images given their CLIP image embeddings, while the prior allows us to learn a generative model of the image embeddings themselves.

1.2 Decoder

They generate images using a diffusion model, where the generation is guided by a CLIP image embedding — a 512-dimensional vector that represents either a real image or a text prompt (via CLIP's joint embedding space). Sometimes, they also use the original text as an additional input.

$$\epsilon_\theta(x_t, t|z_{\text{CLIP}}, \text{text})$$

where:

- x_t - noisy image at timestep t
- $z_{\text{CLIP}} \in R^{512}$ is the CLIP embedding
- text - optional caption
- ϵ_θ predicts noise to remove

They take the timestep embedding (which encodes the current noise step t) and add to it a version of the CLIP embedding that's been linearly transformed to the same size.

$$e_t = e_t + W_{z_{\text{CLIP}}}$$

They take the CLIP embedding, project it into 4 separate "tokens" (vectors), and attach them to the sequence of text tokens generated by GLIDE's text encoder.

Let the projected CLIP embedding be reshaped into 4 tokens:

$$\mathbf{Z}_{\text{CLIP-tokens}} = \text{reshape}(W'\mathbf{z}_{\text{CLIP}}) \in \mathbb{R}^{4\times d}$$

Let the text encoder output be:

$$\mathbf{H} = (\mathbf{h}_1, \ldots, \mathbf{h}_n) \in \mathbb{R}^{n \times d}$$

Then the full conditioning sequence is the concatenation:

$$(\mathbf{z}_1, \ldots, \mathbf{z}_4, \mathbf{h}_1, \ldots, \mathbf{h}_n) \in \mathbb{R}^{(n+4) \times d}$$

To prepare the model for guidance at inference time, they simulate "unconditional" cases during training. Specifically:

10% of the time, CLIP embedding is replaced with 0 (or a special vector),

50% of the time, text is removed.

This allows the model to learn both conditional and unconditional versions of the task.

They use two more diffusion models:

First upsampler: from 64×64 $\rightarrow$ 256×256

Second upsampler: from 256×256 $\rightarrow$ 1024×1024

Each one is trained to increase image resolution while keeping visual consistency.

They deliberately damage the low-res images during training, so the upsamplers learn to fix flaws and don't overfit.

First upsampler: uses Gaussian blur

Second upsampler: uses BSR degradation (a more realistic corruption pipeline: blur, noise, resize, etc.)

To save compute, they don't train the upsampler on full-size 1024×1024 images. Instead, they crop out smaller parts (e.g., 256×256) and train on those.

Why it works: The model still learns how to upsample locally, and generalizes to full resolution during inference.

1.3 Prior

Algorithm 1 DALL·E 2 Decoder Pipeline: From CLIP Embedding to 1024×1024 Image

Require: CLIP embedding $\mathbf{z}_{\text{CLIP}} \in \mathbb{R}^{512}$, optional text caption
Ensure: Final image $x_0^{\text{hi-res}} \in \mathbb{R}^{1024\times1024\times3}$
(Optional) Encode caption to token sequence: $\mathbf{H} = (\mathbf{h}_1, \ldots, \mathbf{h}_n)$
Project $\mathbf{z}_{\text{CLIP}}$ into:
(a) Token embeddings: $\mathbf{Z}_{\text{CLIP-tokens}} = \text{reshape}(W'\mathbf{z}_{\text{CLIP}}) \in \mathbb{R}^{4\times d}$
(b) Timestep bias: $\mathbf{z}_{\text{proj}} = W\mathbf{z}_{\text{CLIP}}$
Form conditioning sequence: $\mathbf{C} = (\mathbf{z}_1, \ldots, \mathbf{z}_4, \mathbf{h}_1, \ldots, \mathbf{h}_n)$

Stage 1: 64×64 Base Image Generation
for $t = T \to 1$ **do**
    Sample noise image $x_t \sim \mathcal{N}(0, I)$
    Compute denoised prediction:

$$\epsilon_{\text{guided}} = (1 + w) \cdot \epsilon_\theta(x_t, t, \mathbf{C}) - w \cdot \epsilon_\theta(x_t, t, \emptyset)$$

    Sample x_{t-1} using reverse DDPM step
end for
$x_0^{\text{low}} \leftarrow x_0$ ▷ Generated 64×64 image

Stage 2: 256×256 Upsampling
Corrupt x_0^{low} with Gaussian blur: $\tilde{x}_0^{\text{low}}$
for $t = T \to 1$ **do**
    Sample noise x_t
    Predict noise: $\epsilon_\theta(x_t, t \mid \tilde{x}_0^{\text{low}})$
    Sample x_{t-1}
end for
$x_0^{\text{mid}} \leftarrow x_0$ ▷ Generated 256×256 image

Stage 3: 1024×1024 Upsampling
Corrupt x_0^{mid} with BSR degradation: $\tilde{x}_0^{\text{mid}}$
Train on 256×256 crops of targets to reduce compute
for $t = T \to 1$ **do**
    Sample noise x_t
    Predict noise: $\epsilon_\theta(x_t, t \mid \tilde{x}_0^{\text{mid}})$
    Sample x_{t-1}
end for
$x_0^{\text{hi-res}} \leftarrow x_0$ ▷ Generated 1024×1024 image

Algorithm 2 DALL·E 2 Prior: Generating CLIP Image Embedding from Caption

Require: Caption y, CLIP text encoder $\text{CLIP}_{\text{text}}$
Ensure: Predicted image embedding $\hat{\mathbf{z}}_i \in \mathbb{R}^{512}$

```
Compute CLIP text embedding: z_t ← CLIP_text(y)
Option 1: Autoregressive (AR) Prior
Compute target image embedding z_i
Apply PCA: z_i^PCA ∈ R^319
Quantize each component into 1024 bins ⇒ sequence {q_1, ..., q_319}
Compute dot product: s = z_i · z_t
Quantize s ⇒ discrete token q_s
Prepare prefix tokens from y, z_t, and q_s
Train Transformer with causal mask to predict {q_1, ..., q_319}
Sampling from AR Prior
Encode prefix as above
Predict q_1, ..., q_319 autoregressively
Dequantize and apply inverse PCA: ẑ_i
Option 2: Diffusion Prior
Train diffusion model on z_i using noising schedule q_t
for t = T → 1 do
    Sample noise: z_i^(t) ~ N(0, I)
    Form input sequence:
    Text tokens from y
    CLIP text embedding z_t
    Timestep embedding e_t
    Noised z_i^(t)
    Placeholder token for prediction
    Use Transformer output to predict ẑ_i^(0)
end for
Inference with Dot Product Re-ranking
Generate 2 samples: ẑ_i^(1), ẑ_i^(2)
Compute ŝ_1 = ẑ_i^(1) · z_t, ŝ_2 = ẑ_i^(2) · z_t
Return the one with higher score: ẑ_i = arg max_ẑ(ŝ_1, ŝ_2)
```