DALL-E-2 2022

Tuesday, July 8, 2025 9:46 AM

Hierarchical Text-Conditioned Image Generation with CLIP Latents

Abstract

- Proposes two-stage model:
  1) Prior that generates a CLIP image embedding given a text caption/prompt
  2) Decoder that generates an image conditioned on the input embedding
- Decoder can produce variations of the image, while preserving semantics & style, but preserving non-essential details
- CLIP enables language-guided image manipulations in zero-shot fashion
  - ↳ model performs a task without being explicitly trained on that specific task or dataset ⇒ tests the ability for the model to generalize

Introduction

CLIP Properties:
- Robust to image distribution shift
- Impressive zero-shot capabilities
- Can achieve SOTA performance on vision & language tasks

Diffusion Properties:
- Can achieve SOTA performance in image & video generation tasks
- Leverages a guidance technique — telling the model how to generate the image given a prompt — which improves quality but reduces sample diversity

- DALL-E-2 combines CLIP & Diffusion:
  1) Train a diffusion decoder to invert the CLIP image encoder
     - Inverter is non-deterministic and can produce multiple images from single image embedding
     - Encoder and its approximate inverse (the decoder) enables capabilities beyond text→image translation
     - Can interpolate between input images by inverting interpolations of their image embeddings
     - Advantage of using CLIP is ability to semantically modify images by moving in direction of any encoded text vector
     - Encoding & decoding images also provides a tool to observe which features of the image are recognized or disregarded by CLIP
  2) A prior model
     - ↳ a model that predicts what an image should look like; models the distribution $p(z_{img} | z_{text})$
     - Generates possible CLIP image embeddings from a given text caption
- Performance compared to other text→image models:
  - Comparable in quality to GLIDE, but greater diversity in generations
  - When trained in latent space, DALL·E-2 has comparable performance to autoregressive priors, but more computationally efficient

Method

- Training dataset: pairs of (x: image, y: corresponding caption)
- Given an image x, z_i = CLIP image embedding & z_t = text embedding
- Design model to produce images from captions using two components:
  1) A prior $P(z_i|y)$ that produces CLIP image embeddings z_i conditioned on captions y
  2) A decoder $P(x|z_i, y)$ that produces images x conditioned on CLIP image embeddings z_i (and optionally, text captions y)
- Decoder enables us to invert images given their CLIP embeddings
- Decoder allows us to learn a generative model of the image embeddings themselves

- The prior [illegible]
- If we stack these components, we get a generative model $P(x|y)$ of [illegible]

probability of generating image x given caption y = joint probability of generating x; given prompt y = image x given the decoder and the prior

$$P(x|y) = P(x, z_i|y) = P(x|z_i, y)\,P(z_i|y)$$

($P(x|z_i, y)$: decoder; $P(z_i|y)$: prior)

Decoder

- Uses diffusion to produce images conditioned on CLIP image embeddings (and optionally the text captions)
  - Modifies GLIDE by projecting and adding CLIP embeddings to timestep embedding, and by projecting CLIP embeddings into 4 extra context tokens that are concatenated to the sequence of outputs from GLIDE's text encoder
- We can sample directly from conditional distribution of the decoder, but past papers show that using guidance on the conditioning information significantly improves sample quality
- Enables classifier-free guidance by setting CLIP embeddings at random to zero (or to a learned embedding) 10% of the time, and randomly dropping the text caption 50% of the time during training

→ doesn't use a separate image classifier to guide image generation, unlike early diffusion techniques

- Trains 2 diffusion upsampler models to generate high-res images:
  - one to upsample from 64x64 → 256x256 pixels
  - another to upsample from $256^2 \to 1024^2$ resolution
- Slightly corrupt the conditioning images during training to improve robustness:
  - For $64^2 \to 256^2$, use gaussian blur
  - For $256^2 \to 1024^2$, use a more diverse BSR degradation

→ image corruption pipeline:
1) Blur the original image (with Gaussian filter)
2) Subsample/downsample (reduce resolution)
3) Reintroduce noise (Gaussian)
4) Upsample the image (e.g., using interpolation, with a neural network, or using diffusion)

- Trains on random image crops 1/4 the image's spatial resolution
  e.g., if image is 256^2 pixels, train on 64^2 px random image crops

  Clarification: we do not need to train all four crops. Instead the paper says to train one crop from the image of size $H/2 \times W/2$ from the full image $H \times W$

  Intuition: Learning to paint patches of fur, glass, or grass - not the whole cat landscape at once - but knowing what it's supposed to be

  - Reduces training compute and improves stability
- Only uses spatial convolutions (no attention) and at inference time, directly apply the model at the target resolution
  - They observe this generalizes to a higher resolution
- They found no benefit in conditioning the upsamplers on the caption
  - feeding the caption into the upsampler didn't help the generated samples

Prior

- A decoder can invert CLIP image embeddings z_i to produce images x, but we need a prior model that produces z_i from captions y (so we can generate images from text)
- 2 model options for the prior:
  1) Autoregressive (AR) prior:
     - CLIP image embedding z_i is converted to sequence of discrete codes and predicted autoregressively conditioned on caption, y.
  2) Diffusion prior:
     - Continuous vector z_i is directly modelled using a Gaussian diffusion model conditioned on caption y.
- We can also condition the prior on the CLIP text embedding z_t (because it's a deterministic function)
- Enables classifier-free guidance for the AR and diffusion prior to improve quality, by randomly dropping the text conditioning information 10% of the time during training

AR Tricks:
- To train & sample the AR more efficiently, they use PCA to reduce dimensionality
  - the rank of the CLIP vector space is drastically reduced when training CLIP with SAM while slightly improving evaluation metrics
  - [illegible] information when shrinking from 1024 → 319 principal components

- they preserve nearly all

- After applying PCA, they order principle components by decreasing eigenvalue magnitude, quan...
  each of the 319 dimensions into 1024 discrete buckets, and predict the resulting sequence with a Transformer with casual attention mask.
  - Results in 3x reduction in # of tokens predicted during inference

Diffusion Prior Tricks:

- Train decoder-only Transformer with casual attention mask on a sequence consisting of (in order):
  1) encoded text: from tokenizer (N tokens x d dimensions)
  2) CLIP text embedding: vector $z_t \in \mathbb{R}^d$ (same dimension)
  3) embedding for diffusion timestep: vector encoding current diffusion time step (e.g., Sinusoidal or learned) of d dimension
  4) noised CLIP image embedding: one vector $z_t \in \mathbb{R}^d$ containing the current noisy image latent the model is trying to denoise
  5) final embedding whose Transformer output is used to predict unnoised CLIP image embedding: learned placeholder value (like [MASK]), whose output is used to predict clean CLIP image embedding

→ Goal: Given a text caption, generate CLIP image embedding z_i for the caption.
- This embedding will be later fed into the diffusion decoder to generate actual image

- Full sequence: $[\text{text token}_{1:d,1:N}, z_t \in \mathbb{R}^d, \text{time step} \in \mathbb{R}^d, z_{i_{noised}} \in \mathbb{R}^d, \text{target} \in \mathbb{R}^d]$

- They don't condition the diffusion prior on z_i: z_t contrary to the AR prior; instead, they improve sampling quality by generating two samples of z_i and selecting the one with higher dot product with z_t.

- Instead of using ϵ-prediction (↳ noise prediction) like in the DDPM paper, they train the model to predict unnoised z_i directly and use MSE-loss on this prediction:

Formula 1: $L_{prior} = \mathbb{E}_{t\sim[1,T], z_i^{(t)}\sim q_t}\left[\|f_\theta(z_i^{(t)}, t, y) - z_i\|^2\right]$

- L_{prior}: diffusion loss prior
- $\mathbb{E}_{t\sim[1,T], z_i^{(t)}\sim q_t}$: expected value over random timestep and noisy latent sampled from forward noising process
- $f_\theta(z_i^{(t)}, t, y)$: joint predicted (NN) distribution of the noisy CLIP embedding, the timestep, and the text caption
- z_i: the CLIP image embedding

Meaning: To train the diffusion prior, we teach a Transformer to predict the original CLIP image embedding from a noisy version, using the text caption and the noise level as guidance, by minimizing the squared residual of it's prediction and true embedding.

Purpose: Generates the CLIP image embedding, which represents what the image should look like.

Image Manipulations

- Our approach allows us to decode any image x into a 2-part latent representation (z_i, x_T) that's sufficient for the decoder to produce an accurate reconstruction.
  - z_i: CLIP image embedding
  - x_T: pure-noise image

  - Given a pure-noise image and instructions, the decoder can generate the image and it will be conditioned to generate the image according to the caption because it's encoded in the CLIP text embedding.

- z_i is obtained by encoding the image with the CLIP image encoder
- x_T is obtained by applying DDIM inversion to x using the decoder, while conditioning on z_i
- 3 types of image manipulations that this representation enables:
  1) Variations
  2) Interpolations
  3) Text Diffs

Variations

- Given an input x, we can produce related images sharing same essential content, but vary in other ways
- To do this, we apply decoder to the 2-part representation using DDIM with $\eta > 0$ for sampling.
- To do this, we apply decoder to the 2-part representation using DDIM with $\eta > 0$ for sampling. ...deterministic and reconstructs the original image x

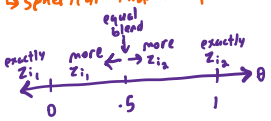
- when $\eta = 0$, the decoder becomes deterministic
- larger η values introduce more variety, but are centered still around x
- as η increases, the variations tell us what information was captured by CLIP image embedding and what was lost.

Interpolations

- Enables blending of 2 images x_1 & x_2
- To do this, we rotate between their CLIP embeddings z_{i_1} and z_{i_2} using spherical interpolation, yielding intermediate CLIP representations $z_{i_\theta} = \text{slerp}(z_{i_1}, z_{i_2}, \theta)$ as θ is varied from 0 to 1.

$\theta \in [0,1]$

↳ spherical linear interpolation:



Text Diffs

- key advantage of using CLIP compared to other models for image representations is that images & text embedded in same latent space, enabling language-guided manipulations (text diffs)
- To modify an image to reflect a new text description y, we 1) use its CLIP text embedding z_t as well as the CLIP text embedding z_{t_0} of a caption describing the current image.

2) we then compute a text diff vector $z_d = \text{norm}(z_t - z_{t_0})$

3) Then, we rotate between the image CLIP embedding z_i and the text diff vector z_d using spherical interpolation
   - this yields intermediate CLIP representations $z_\theta = \text{slerp}(z_i, z_d, \theta)$ where θ is increased linearly from 0 to a maximum valve that's typically in $[.25, .5]$.
- we produce final outputs by decoding the interpolates z_θ and fixing the base DDIM noise to x_T throughout the entire trajectory.

Probing the CLIP Latent Space

- Decoder provides unique opportunity to explore CLIP's latent space by allowing us to directly visualize what the CLIP image encoder is seeing