Deep Learning for Computer Vision

Text-Conditioned and Latent Diffusion Models

Vineeth N Balasubramanian

Department of Computer Science and Engineering
Indian Institute of Technology, Hyderabad

भारतीय प्रौद्योगिकी संस्थान हैदराबाद
Indian Institute of Technology Hyderabad

Text-Conditioned Generation

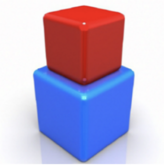
- In the last lecture, we saw classifier and classifier-free guidance for class-based generation. We used an additional input y (class label) to model the conditional distribution $p(x|y)$. Which allows us to generate the desired image given the conditioning signal.
- We can also extend this concept to text and replace the class labels with text sequences for text-to-image generation diffusion models
- We can leverage existing language or vision-language models instead of a classifier to achieve this

Why Text-Conditioned?

- Class-based guidance is incapable of generating complex images
- E.g. "*a dog sitting on the table*", "*an astronaut sitting on a horse on the moon*"
- To enhance the precision of this generation task, we require greater controllability over the diffusion process $\rightarrow$ we require a text-conditioned diffusion model

"a high-quality oil painting of a psychedelic hamster dragon"

"an illustration of albert einstein wearing a superhero costume"



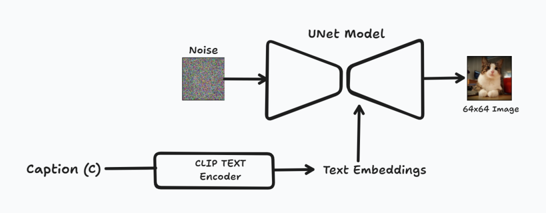
"a red cube on top of a blue cube"

"a stained glass window of a panda eating bamboo"

Image Credit: Nichol et al, GLIDE: Towards Photorealistic Image Generation and Editing with Text-Guided Diffusion Models, ICML 2022

GLIDE from OpenAI

- GLIDE from OpenAI proposed caption and CLIP guidance for the text-conditioned diffusion model, trained on hundreds of millions of paired datasets
- The model is trained by feeding a text prompt into a massive diffusion model as a condition
- 64×64 base diffusion model, which uses UNet architecture
- Token embedding from CLIP text encoder is placed into the class embedding of the UNet model

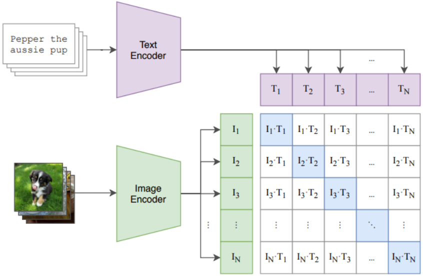


Nichol et al, GLIDE: Towards Photorealistic Image Generation and Editing with Text-Guided Diffusion Models, ICML 2022

Text-Conditioned Diffusion Model

Understanding CLIP, a vision-language foundation model:

- CLIP: Contrastive Language-Image Pre-training.
- Projects given text and image pairs into the same embedding space via a text and image encoder, uses cosine similarity distance function to train model in a contrastive manner
- Can efficiently learn visual concepts in the form of text via natural language supervision



(More in the next week...)

Image Credit: Radford et al, Learning Transferable Visual Models From Natural Language Supervision, ICML 2021

CLIP-Guided Text-Conditioned Diffusion Model

- To use CLIP guidance, we need to re-train CLIP on the noised dataset
- Given an image x and a prompt c, a CLIP model computes the alignment via cosine similarlity between x and c, indicating how similar the image and the prompt are

CLIP-Guided Text-Conditioned Diffusion Model

- To use CLIP guidance, we need to re-train CLIP on the noised dataset
- Given an image x and a prompt c, a CLIP model computes the alignment via cosine similarlity between x and c, indicating how similar the image and the prompt are
- To get the guidance signal, we use CLIP similarity score and calculate the gradient of this score w.r.t the noised image x_t:

$$\hat{\mu}_\theta(x_t|c) = \mu_\theta(x_t|c) + s \cdot \Sigma_\theta(x_t|c)\nabla_{x_t}(f(x_t) \cdot g(c))$$

where:

- $f(x_t)$ - CLIP Image Encoder; $g(c)$ - CLIP Text Encoder
- s - guidance weight (same as γ in classifier and classifier-free guidance)
- $\Sigma_\theta(x_t|c)$ - covariance matrix

CLIP-Guided Text-Conditioned Diffusion Model

- To use CLIP guidance, we need to re-train CLIP on the noised dataset
- Given an image x and a prompt c, a CLIP model computes the alignment via cosine similarlity between x and c, indicating how similar the image and the prompt are
- To get the guidance signal, we use CLIP similarity score and calculate the gradient of this score w.r.t the noised image x_t:

$$\hat{\mu}_\theta(x_t|c) = \mu_\theta(x_t|c) + s \cdot \Sigma_\theta(x_t|c)\nabla_{x_t}(f(x_t) \cdot g(c))$$

  where:

  - $f(x_t)$ - CLIP Image Encoder; $g(c)$ - CLIP Text Encoder
  - s - guidance weight (same as γ in classifier and classifier-free guidance)
  - $\Sigma_\theta(x_t|c)$ - covariance matrix

Note: Without re-training CLIP, performance is sub-optimal

GLIDE: Text-Conditioned Diffusion Model

- Classifier-Free Diffusion Guidance with caption conditioning (rather than class conditioning):

$$\hat{\epsilon}_\theta(x_t|c) = \epsilon_\theta(x_t|\emptyset) + s \cdot (\epsilon_\theta(x_t|c) - \epsilon_\theta(x_t|\emptyset))$$

- CLIP-guided Diffusion Guidance:

$$\mu\epsilon_\theta(x_t|c) = \mu\theta(x_t|c) + s \cdot \sum_\theta (x_t|c)\nabla_{x_t}(f(x_t) \cdot g(c))$$

Caption conditioning worked better in their results

Radford et al, Learning Transferable Visual Models From Natural Language Supervision, ICML 2021

Pixel-Space Diffusion Models: Limitations

- Input and output sizes are same, making network parameters count large
- Requires significant computation power and long time to train
- Slow at sampling or inference time
- Requires a large amount of GPU memory

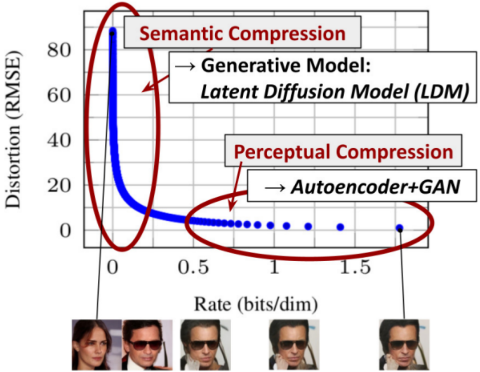
Pixel-Space Diffusion Models: Limitations

- Input and output sizes are same, making network parameters count large
- Requires significant computation power and long time to train
- Slow at sampling or inference time
- Requires a large amount of GPU memory

How can we reduce the computation cost and sampling time for diffusion models?

Towards Latent Diffusion Models

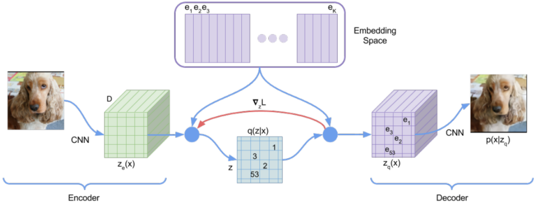
- Rather than train the diffusion model in pixel space, train in latent space!
- Majority of pixels within an image represent insignificant details that may not have semantic relevance
- Latent diffusion models provide faster training and sampling, with lower memory requirements and lower cost of computation



Rombach et al, High-Resolution Image Synthesis with Latent Diffusion Models, CVPR 2022

Understanding the VQ-VAE

- VQ-VAE uses discrete latent variables instead of continuous normal distribution in VAEs
- Encoder network takes image x and encodes into z_e
- VQ layer takes z_e and samples embedding from a dictionary based on distance $\nabla_z L$ to output z_q
- Decoder network takes z_q and outputs x' to recreate input x



Oord et al, Neural Discrete Representation Learning, NeurIPS 2017

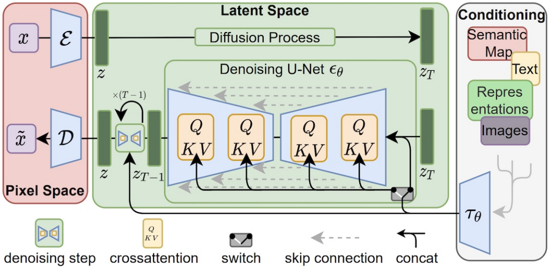
Understanding the VQ-VAE

- Loss Function:
$$\mathcal{L} = \log p(x|z_q(x)) + \|\text{sg}[z_e(x)] - e\|^2 + \beta\|z_e(x) - \text{sg}[e]\|^2$$
- **Reconstruction loss:** $\log p(x|z_q(x))$ is used to optimize the encoder and decoder networks
- **Embedding space loss:** $\|\text{sg}[z_e(x)] - e\|^2$, a dictionary learning algorithm uses an l_2 error to move the embedding vectors e_i towards the encoder output (sg represents stop gradient)
- **Commitment loss:** $\beta\|z_e(x) - \text{sg}[e]\|^2$ is used to control the volume of embedding space and make sure the encoder commits to an embedding
- β is a hyperparameter that controls how much to weigh

Oord et al, Neural Discrete Representation Learning, NeurIPS 2017

Latent Diffusion Models (LDMs)

- To achieve LDMs:
  - First compress image x using VQ-VAE encoder ϵ into a lower dimensional latent embedding z
  - Then, z is fed into the U-Net model to learn to generate latent (i.e., compressed representations) of image which are then decoded into image $\hat{x}$ via the VQ-VAE decoder D
  - To make it a conditional model, a T_θ encoder is used to encode (text, label or image) into embedding and pass it to cross-attention layers of UNet as a guidance signal



Rombach et al, High-Resolution Image Synthesis with Latent Diffusion Models, CVPR 2022

Training LDMs

- **Step 1:** Train a VQ-VAE model (or use pre-trained model) on the dataset
- **Step 2:** Train T_θ encoder (or use pre-trained model) on the dataset
  - In LDM training, both VQ-VAE Encoder and Decoder are frozen
- **Step 3:** Train Latent Diffusion UNet model and τ_θ encoder model using the loss function:

$$\left|\epsilon - \epsilon_\theta(z_t, t, \tau_\theta(y))\right|^2$$

- Jointly optimize both τ_θ and ϵ_θ using above loss function

Popular Diffusion Models

- Stable Diffusion: https://huggingface.co/spaces/stabilityai/stable-diffusion
- DALLE-2: https://github.com/lucidrains/DALLE2-pytorch
- Imagen: https://imagen.research.google/
- SORA: https://github.com/hpcaitech/Open-Sora
- DreamBooth: https://huggingface.co/docs/diffusers/en/training/dreambooth
- ControlNet: https://huggingface.co/docs/diffusers/en/using-diffusers/controlnet

Homework

Readings

- Lilian Weng, What are Diffusion Models?
- Text-to-Image: Diffusion, Text Conditioning, Guidance, Latent Space
- (YouTube video) OpenAI GLIDE: Towards Photorealistic Image Generation and Editing with Text-Guided Diffusion Models
- (YouTube video) VQ-VAE — Everything you need to know about it.