GENERATIVE AI

Mike Spertus

University of Chicago MPCS 57200

November 12, 2024

GPTS

Another way to make Gen AI apps

- As we have mentioned, models are not an application
- Just like an engine is not an automobile
- Techniques we have used to create an application from a model include
  - Prompt Engineering
  - Langchain
  - Langgraph
  - Streamlit
  - RAG
- While these are techniques for building Gen AI applications
  - They are not Gen AI themselves
- What if we could use Gen AI to make Gen AI applications?

Perspective: AI has been moving up the application stack

- Originally (2010-2022), Deep Learning models were primarily accessed through APIs in traditional programming fashion
- More recently, langchain chains embedded deep learning models into chain workflows
  - Such chains can be thought of as creating an Embedded AI DSL (Domain-Specific Language) in Python and Javascript
- Even more recently, chains have been giving way to agents where an AI model directs the AI workflow using tools like langgraph
  - Again, the AI workflows are embedded in Python or Javascript programs

GPTs: Pushing AI even further up the application stack

- OpenAI recently introduced a conversational interface for creating customized applications
    - https://openai.com/blog/introducing-gpts
- No Python or Javascript at all
    - Just natural language
- Let’s take a look

WHAT IS THE GOAL OF GEN AI TRAINING?

Simple Answer

We train Gen AI models to generate output that looks like the data they were trained on

But what does that mean?

An important but surprisingly difficult questions

Review: Temperature

- If we train an LLM to output the most likely word
  - Temperature = 0
- Any single word that it produces will indeed look like what you would expect in the training set
- But if you look at its output over time
  - It will look very different than the training data
- In the training corpus, the most likely word is not always chosen
- This suggests the following idea

Probability distributions

- A Generative AI system trained on a data set should generate data to match the probability distribution at which it would appear in the training data
    - If "perfectly trained," the LLM would be able to produce output that would be able to fool us into thinking that it was part of the data that was used to train it
- We approached this for LLMs by using a Softmax layer to create a probability distribution to sample outputs from
    - Temperature = 1
- However, we didn't address this when we looked at image generation (encoder-decoder, VAE, etc.)
- As we dive more deeply into image generation in this lecture, we will constantly look through this lens

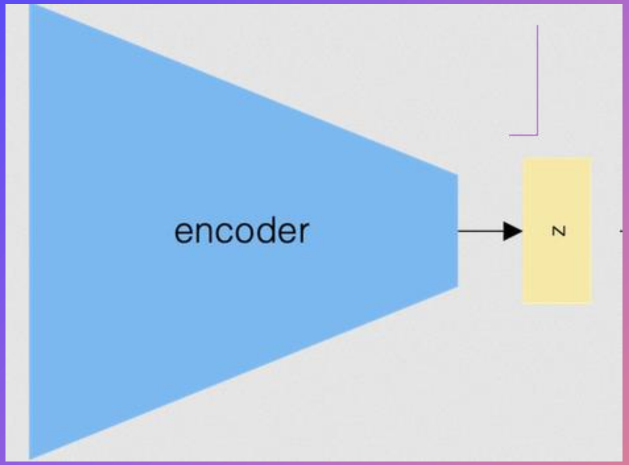
BEYOND TEXT IMAGES

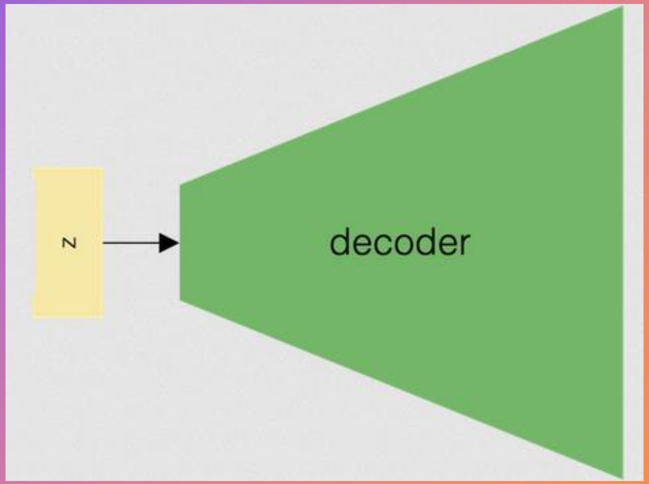
REVIEW OF ENCODER-DECODE FOR IMAGE GENERATION

Encoder/Decoder

An *encoder* embeds its input into a low-dimensional vector space called a *latent space*

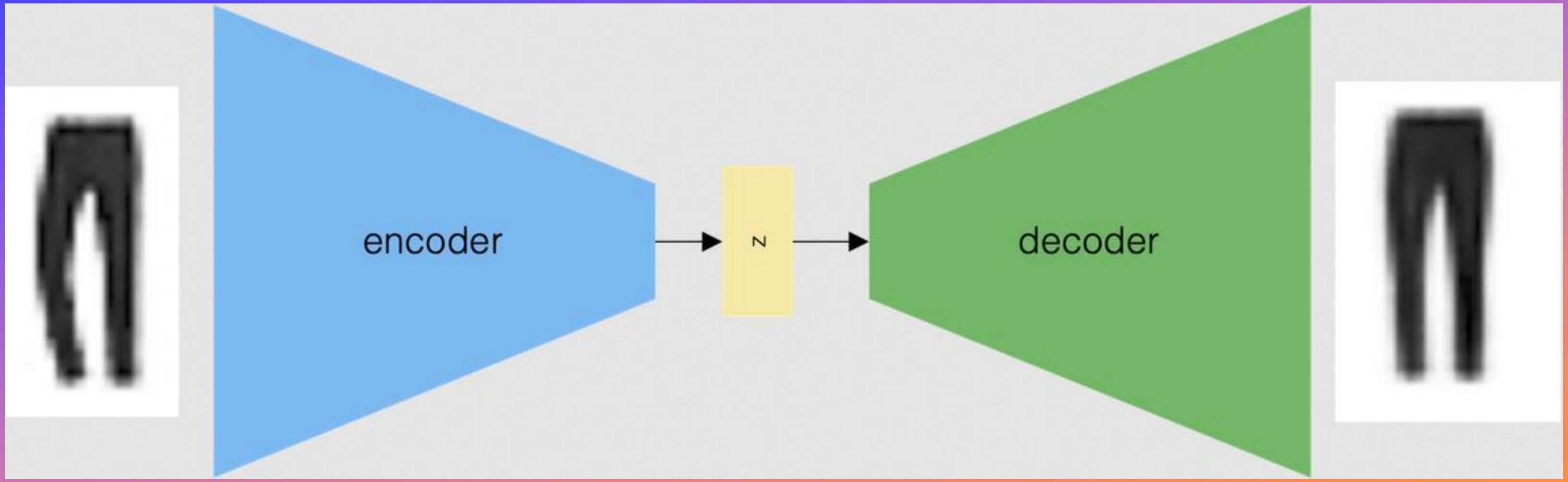
The *decoder* generates an output from a latent space vector





How can we generate a good embedding?

- Put the encoder and decoder together into a single net
- And train the output to be as close as possible to the input
- Since the latent space is a layer of the neural net
- It will have to learn how to efficiently code the inputs in a low-dimensional vector space
- Since the embedding was learned automatically, this architecture is called an *autoencoder*

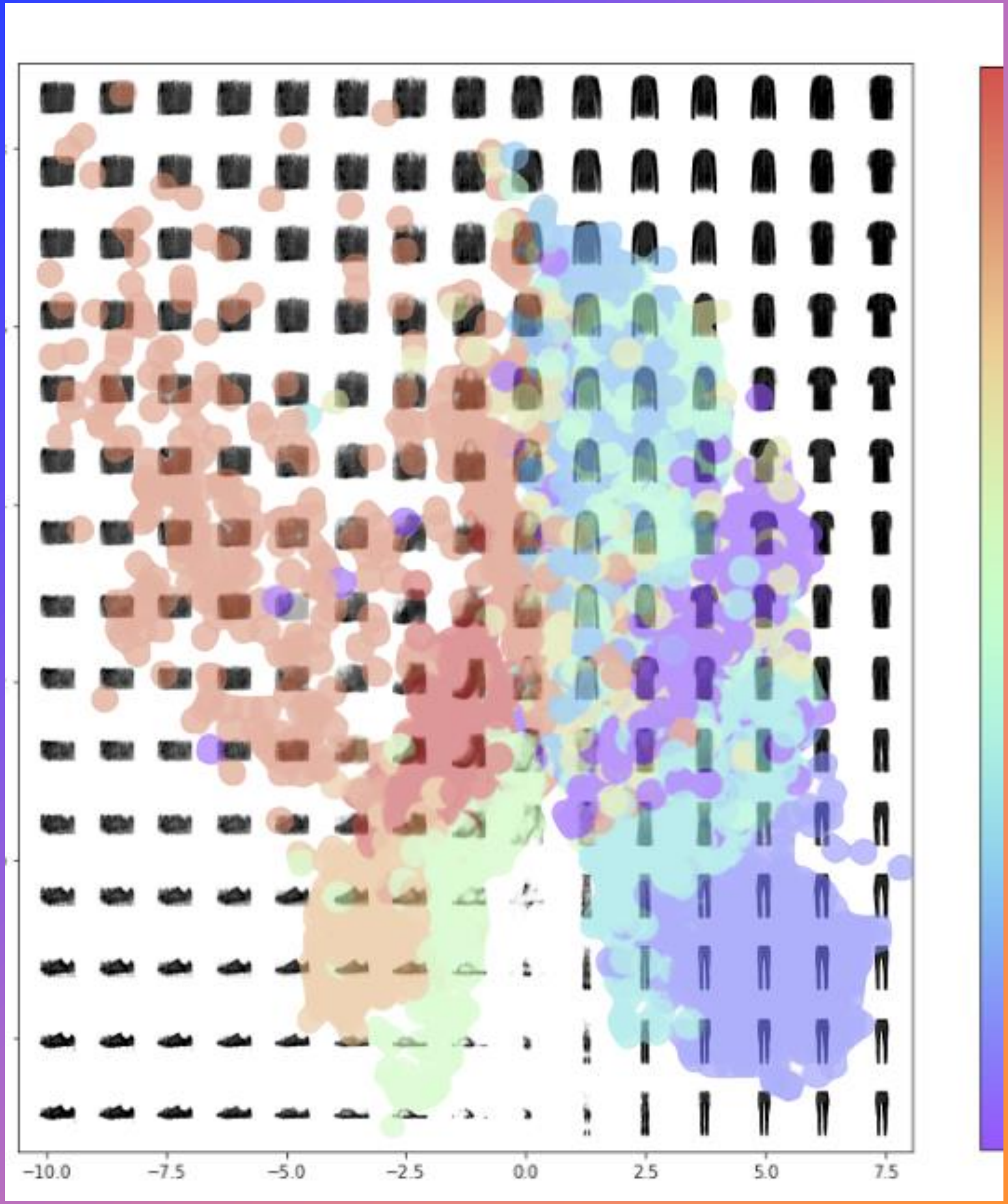


Generative AI

- While we train the encoder and decoder together
- We often run them separately
- The decoder can be used to generate original outputs
- By choosing a new point in the latent space
- This is hard to understand just by talking about it
- So let’s walk through an example

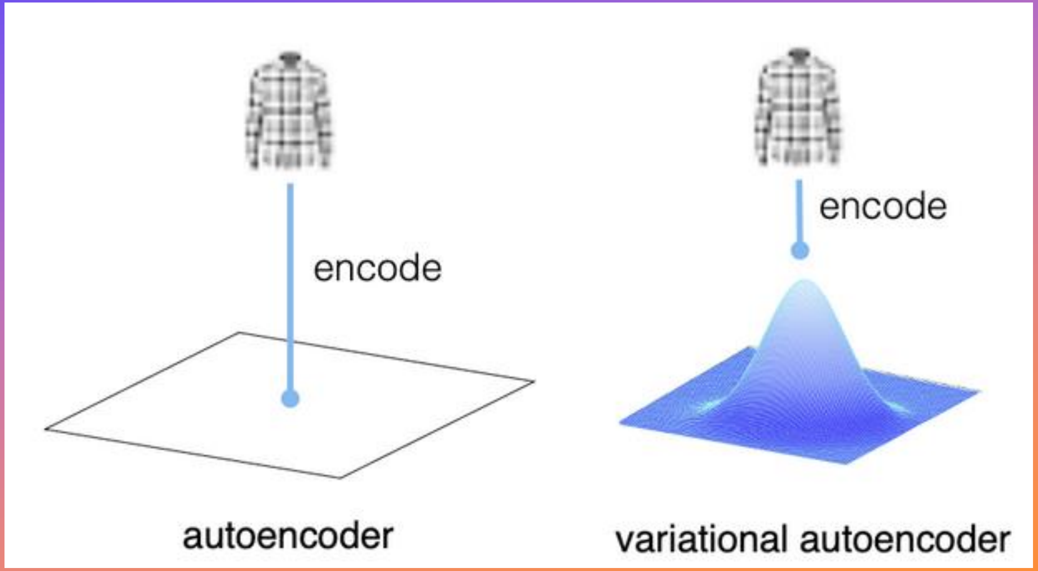
Let’s take a closer look at our encoder output

- If you look very closely
- You can see a lot of “Frankenstein-y” output as you abruptly transition from one clothing type to another
- Also, large regions of the latent space were never touched during training
- So who knows what they’ll generate

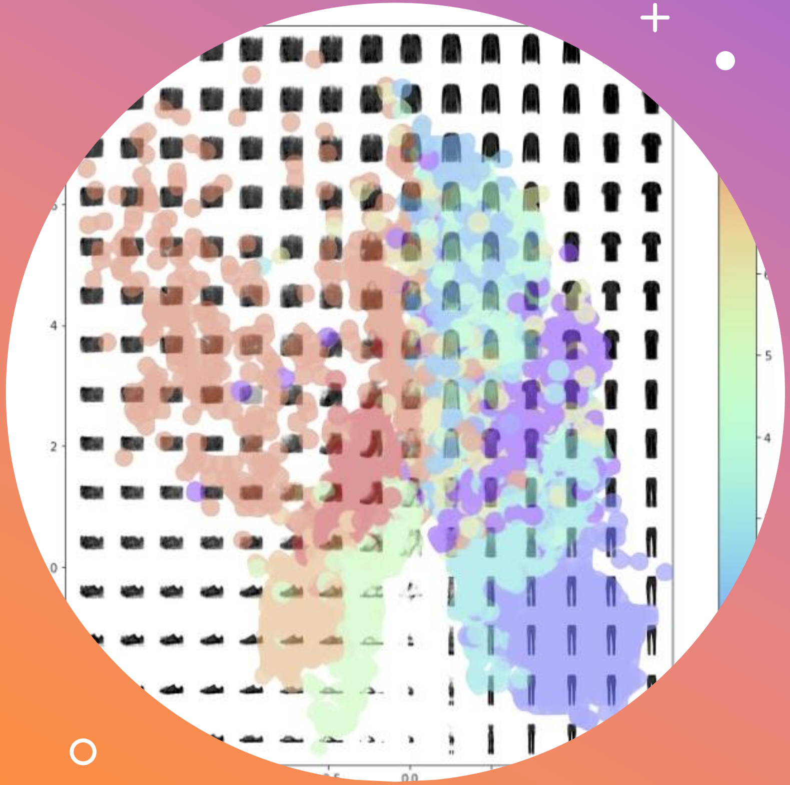


Variational autoencoder

- We'd like to apply some sort of smoothing to make the transitions less discontinuous
- A *variational autoencoder* adds gaussian noise to the generated vector, smoothing out the transitions



VAES ARE POWERFUL, BUT



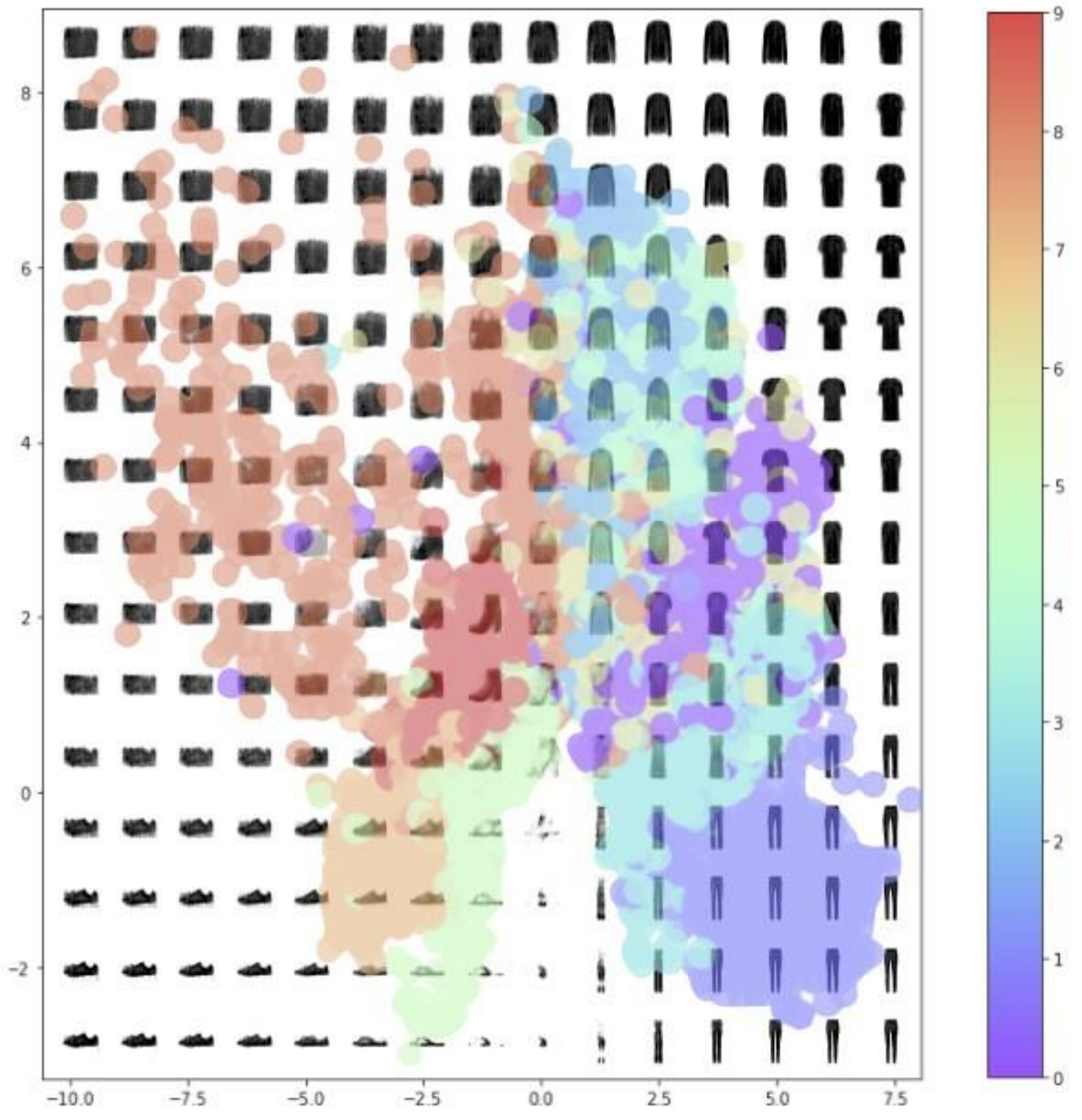
They also have their problems

They tend to produce blurry images

- We saw this in our toy VAE
- But it remains a problem even for industrial strength ones
- The cause of this is still being debated

Their output is incorrectly distributed

- In Generative AI, you may want to be able to generate data that looks like the training set
- Consider training data like MNIST fashion
- In VAE, the inputs are mapped into latent space
- If I select a point in latent space, I may generate something that looks like it could come from the training data, but
- It may not if I choose a point in latent space that isn't close to the training data
- Furthermore, if 10% of the data consists of shirts, there is no reason to think that 10% of the latent space encoding is in the "shirt region," so shirts will likely be over or underrepresented when generating from the latent space



NORMALIZING FLOWS

Normalizing flows address these issues

- In a normalizing flow
- The encoder and decoder are exact inverses of each other on the training data
- So the output is exactly the same as the input for the training data
- The probability distribution of training points is mapped to a normal Gaussian (bell curve) in latent space
  - So normal sampling of latent space will generate data that looks like the training data
- In the illustration on the next slide, the training data is the 2 dimensional "two means" training set
  - In real life, it would be, say, 65536 dimensional for 256x256 grayscale images
  - The latent space is distributed like a Gaussian bell curve around the origin
  - Gaussian sampling from it produces data that looks like the original
  - https://arxiv.org/pdf/1605.08803.pdf

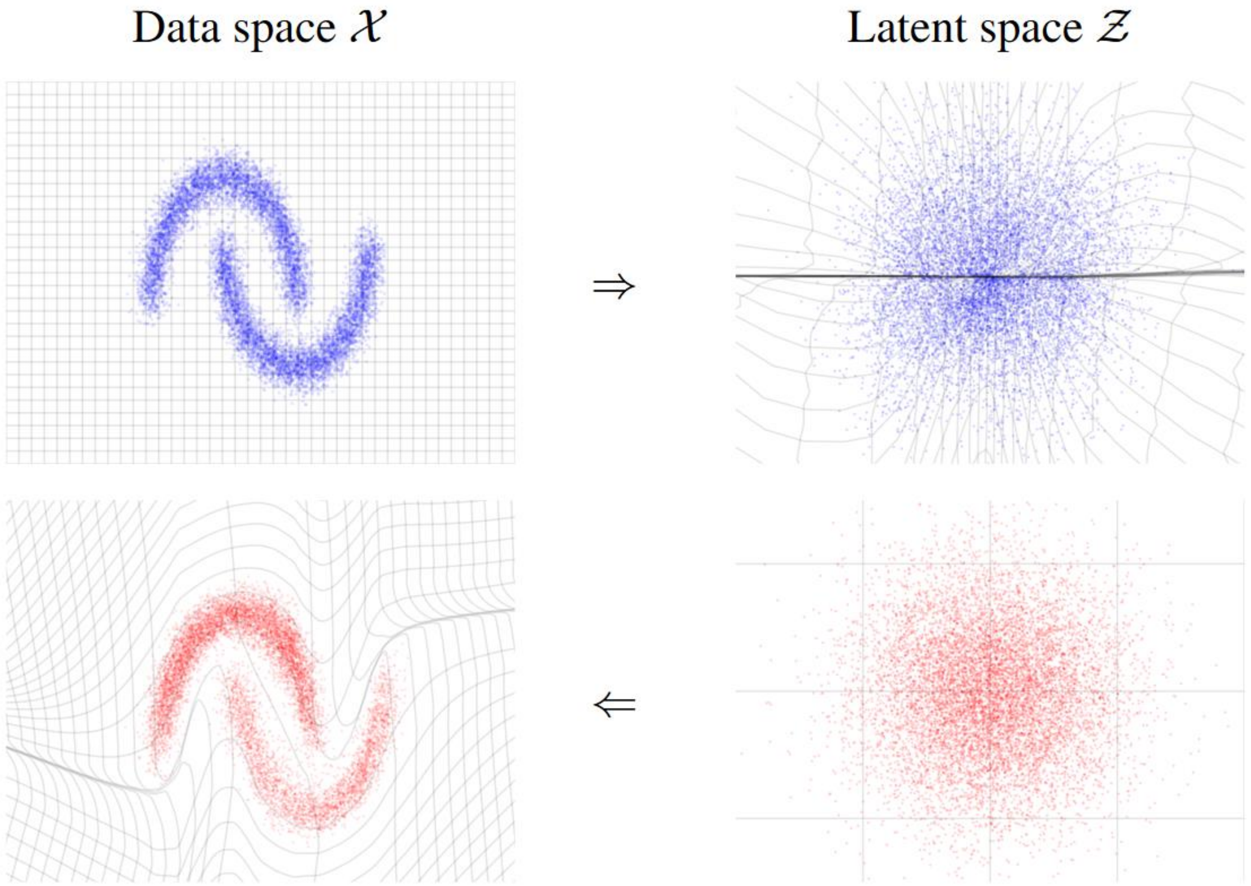
Illustration

Inference

$$x \sim \hat{p}_X$$
$$z = f(x)$$

Generation

$$z \sim p_Z$$
$$x = f^{-1}(z)$$



NOW LET’S TRY IT IN A NOTEBOOK

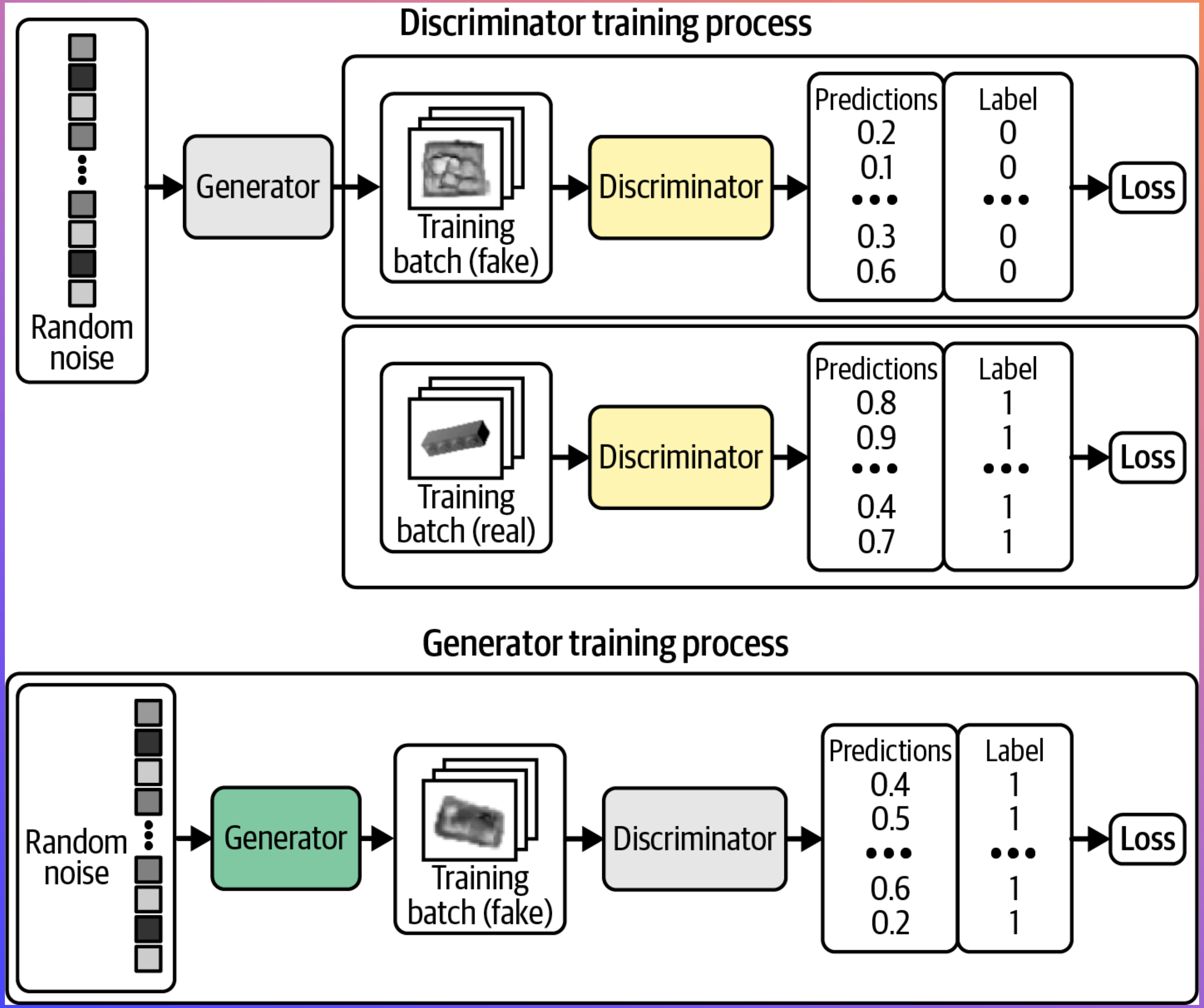
GENERATIVE ADVERSARIAL NETS

Key idea

- As we said above, the goal of Gen AI is to produce data that can fool us into thinking it came from its training set
- So in addition to a generative model that produces data
- Let’s train a discriminative recognizer model to distinguish generated data from the original training set
- And have the generator and discrimator train each other!

Example: Image Recognition

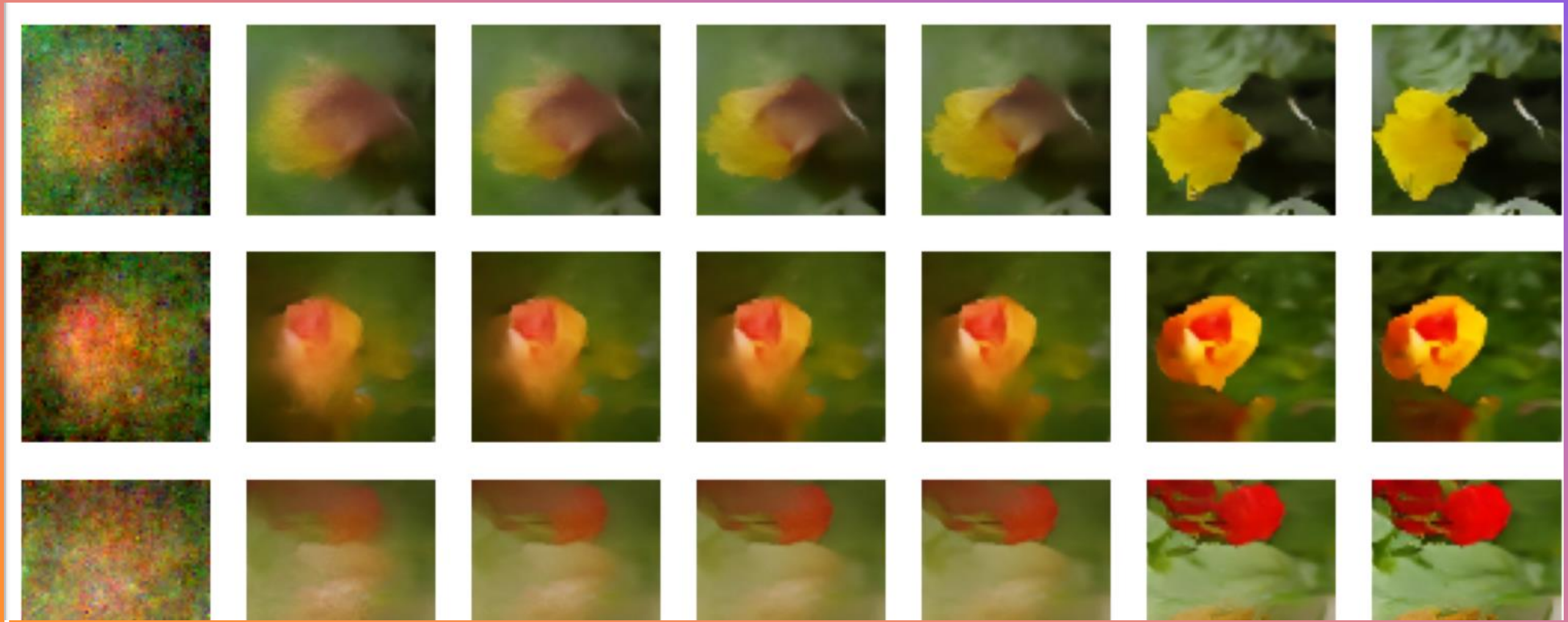
- The generator creates fake images, which are intermixed with real images
- In alternate training runs
    - The discriminator/critic trains itself to tell which is which
    - The generator trains itself to fool the discriminator
- You can even start out with random images


Discriminator training process
Random noise
Generator
Training batch (fake)
Discriminator
Predictions 0.2 0.1 ••• 0.3 0.6
Label 0 0 ••• 0 0
Loss
Training batch (real)
Discriminator
Predictions 0.8 0.9 ••• 0.4 0.7
Label 1 1 ••• 1 1
Loss
Generator training process
Random noise
Generator
Training batch (fake)
Discriminator
Predictions 0.4 0.5 ••• 0.6 0.2
Label 1 1 ••• 1 1
Loss

GENERATIVE ADVERSARIAL NET

DEMO

DIFFUSION MODELS

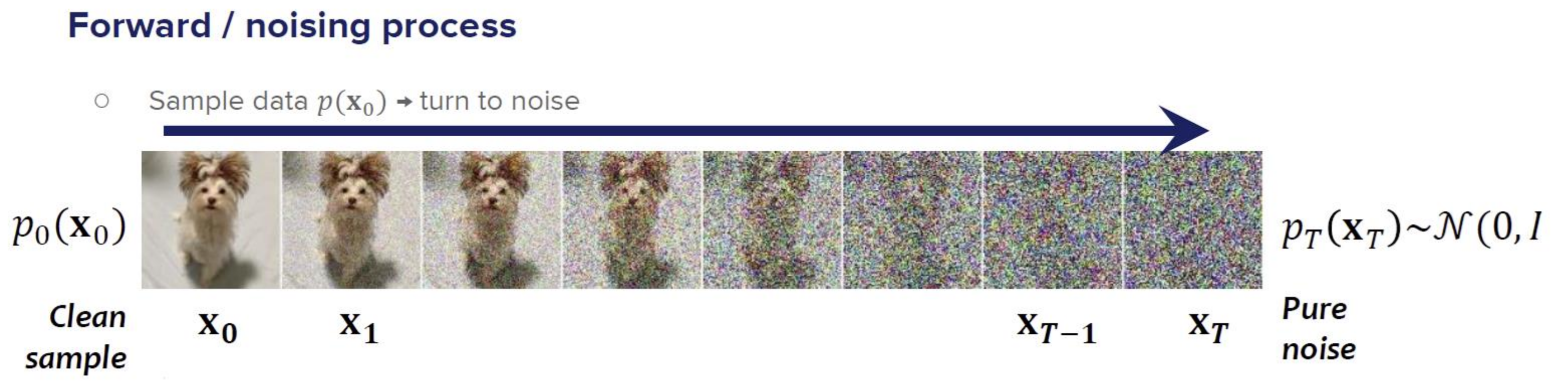


The goal of diffusion models

- Suppose we have a lot of pictures of flowers
- And we want to generate more pictures of flowers similar to the ones we have
- With very high quality
- But not necessarily similar to any particular one
- A diffusion model is a great choice

The idea of diffusion models

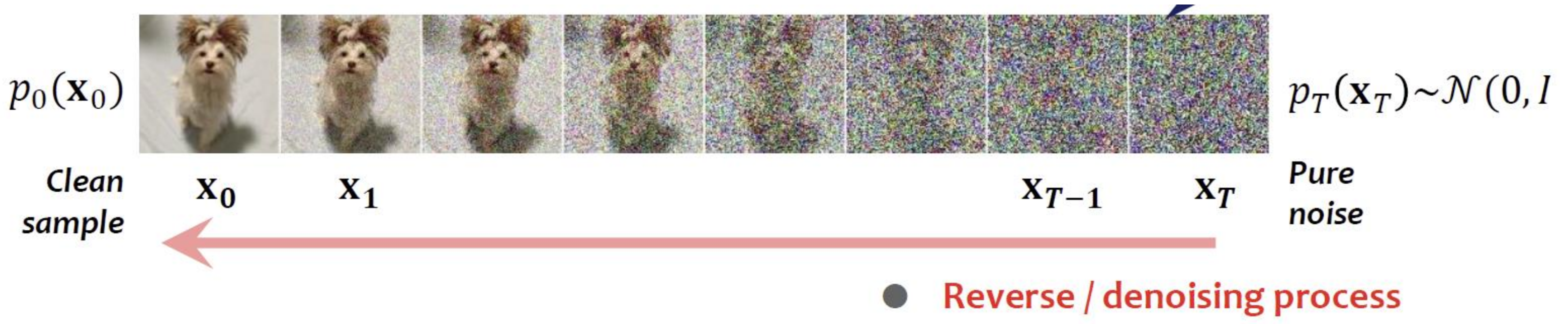
- The encoding process the opposite of autoencoder
- Instead of trying to extract the key features
- We gradually diffuse them away
  - like a drop of ink dissolving in water until it is spread throughout
- Note: Latent space is the same size as data space



Forward / noising process
Sample data $p(\mathbf{x}_0)$ → turn to noise
$p_0(\mathbf{x}_0)$
Clean sample
$\mathbf{x}_0$
$\mathbf{x}_1$
$\mathbf{x}_{T-1}$
$\mathbf{x}_T$
$p_T(\mathbf{x}_T) \sim \mathcal{N}(0, I$
Pure noise

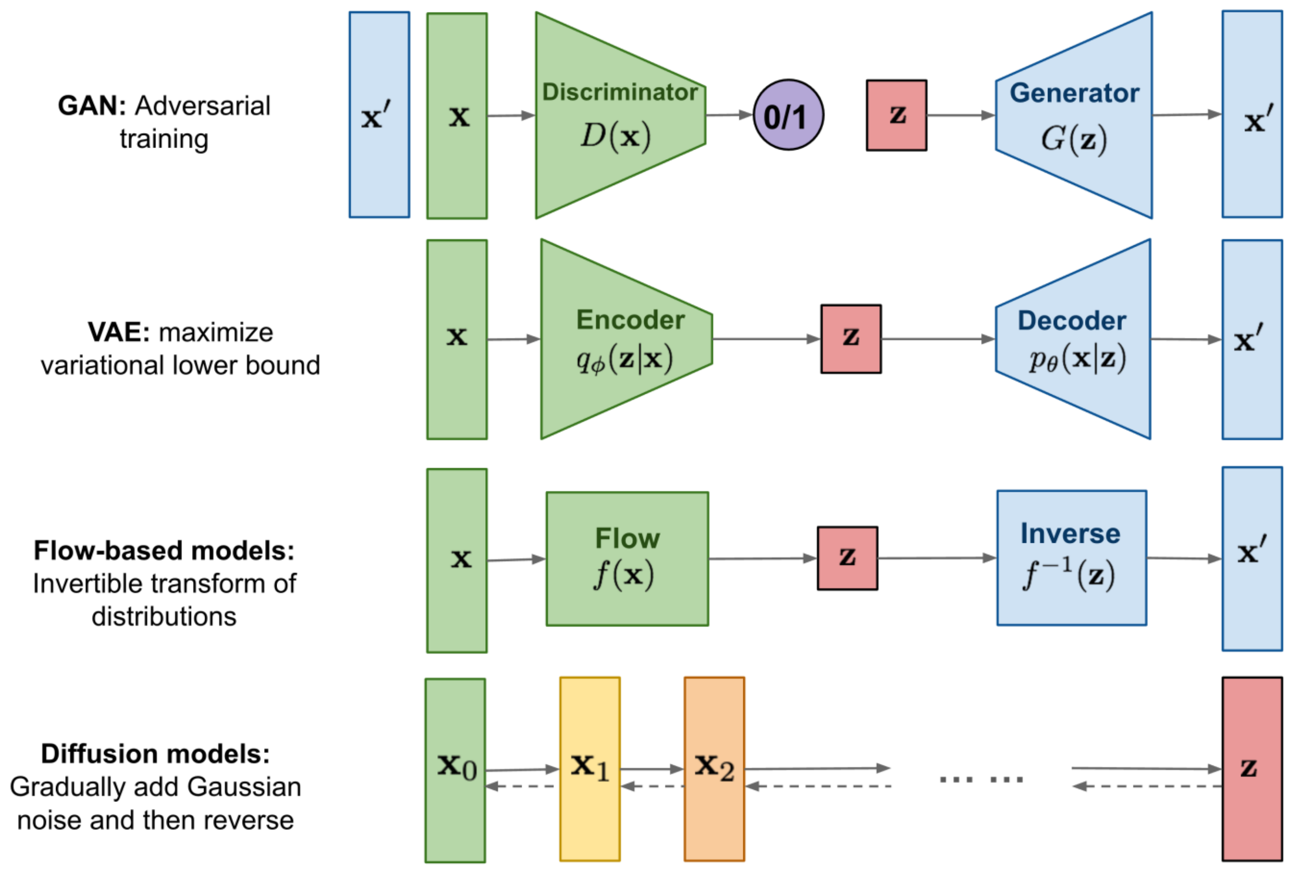
The idea of diffusion models

- Only the decoding process is trained
- To “denoise” each step
- Since the “noising” lost information, the decoder might have to randomly make some guesses as to which trajectory to follow
  - but that’s what we want



DIFFUSION MODEL DEMO

Putting it together



https://lilianweng.github.io/posts/2021-07-11-diffusion-models/

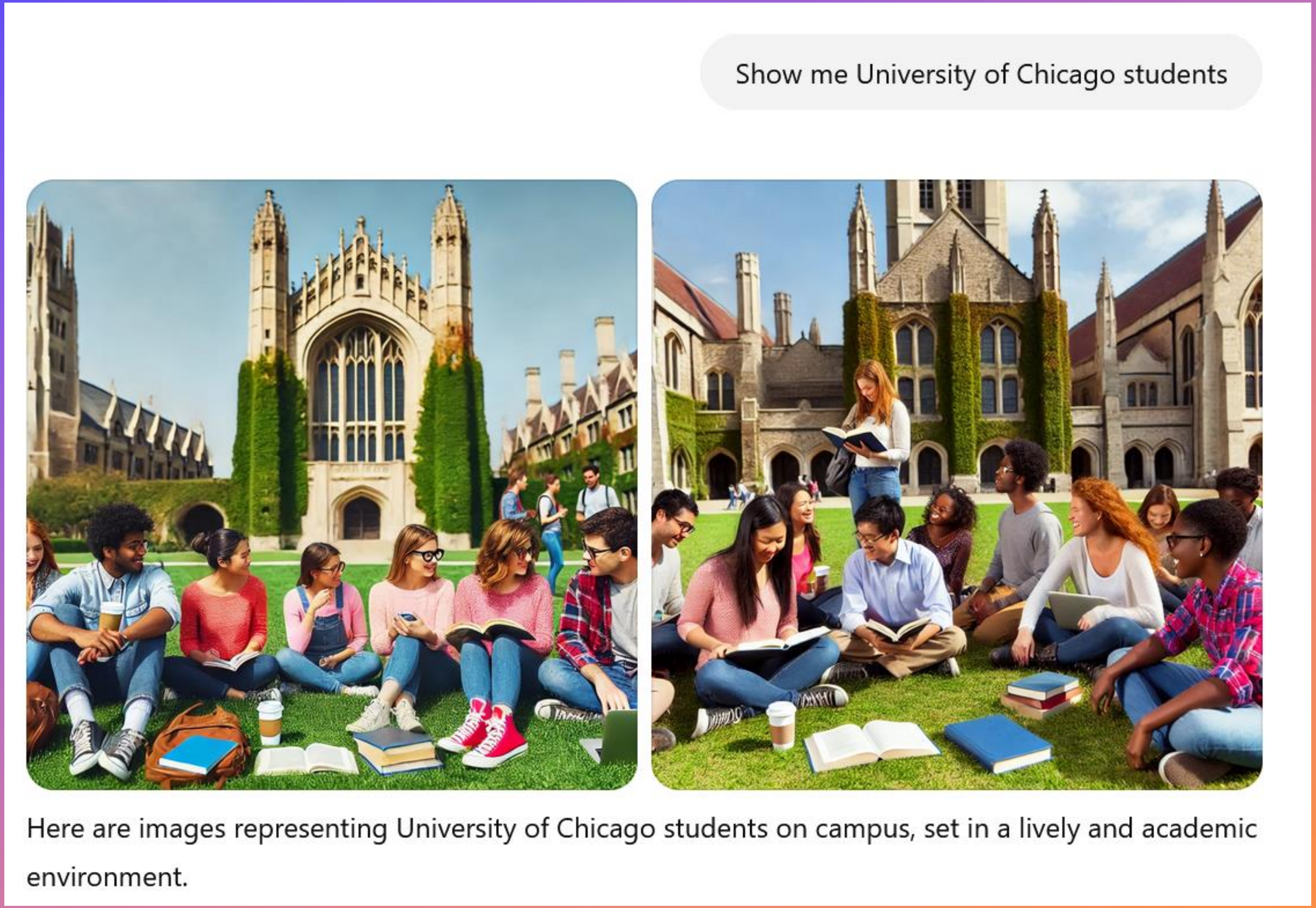
IMAGES AND TEXT

Dall-E

Technology to Product

- We have learned a lot of Image Generation technologies
  - Variational Autoencoders
  - GANs
  - Normalizing Flows
  - Diffusion Models
- How can we put Generative AI technologies like these together into complete products
- Let’s look at the overwhelmingly successful DALL-E Text-to-Image generators and see

BEYOND IMAGES: IMAGES AND WORDS



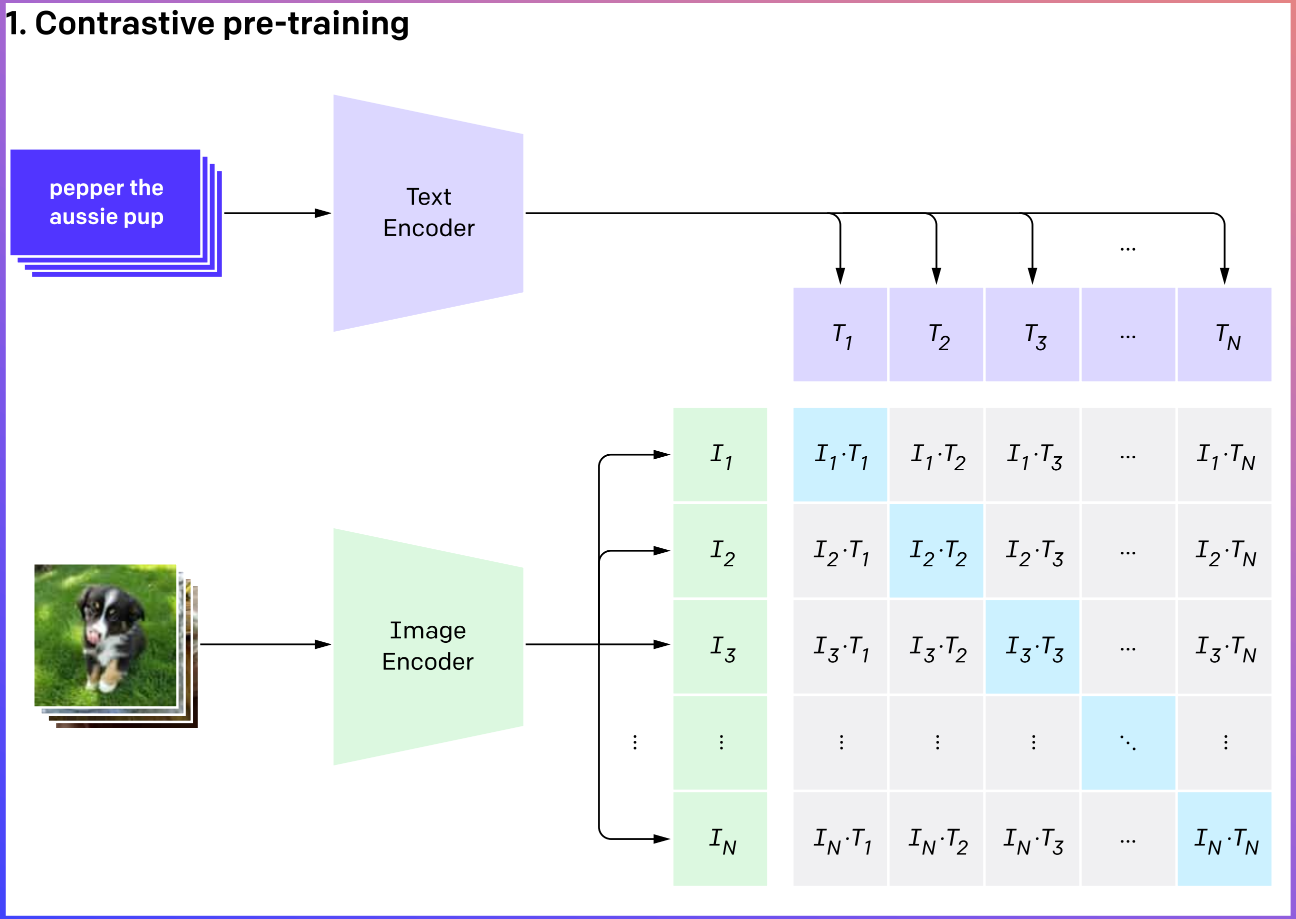
Show me University of Chicago students

Here are images representing University of Chicago students on campus, set in a lively and academic environment.

Text-to-image: CLIP

- **C**ontrastive **L**anguage-**I**mage **P**re-training was used for DALL-E 2
- CLIP trains 2 encoders on hundreds of millions of pictures and their captions
  - Encoder-encoder model? ☺
- One encoder embeds captions
- The other encoder embeds images
- The loss function is based not just on maximizing the cosine similarity of the matching caption and image embeddings are
- But also on minimizing the cosine similarity of a caption embedding from the embedding of non-corresponding images
  - This is why it is called “Contrastive”
- This aids in learning what parts of images are key to their descriptions
- https://arxiv.org/abs/2103.00020

CLIP IN PICTURES


1. Contrastive pre-training
pepper the aussie pup
Text Encoder
Image Encoder
T_1 T_2 T_3 … T_N
I_1 I_2 I_3 ⋮ I_N
$I_1 \cdot T_1$ $I_1 \cdot T_2$ $I_1 \cdot T_3$ … $I_1 \cdot T_N$
$I_2 \cdot T_1$ $I_2 \cdot T_2$ $I_2 \cdot T_3$ … $I_2 \cdot T_N$
$I_3 \cdot T_1$ $I_3 \cdot T_2$ $I_3 \cdot T_3$ … $I_3 \cdot T_N$
$I_N \cdot T_1$ $I_N \cdot T_2$ $I_N \cdot T_3$ … $I_N \cdot T_N$

https://arxiv.org/abs/2103.00020

Remember how we couldn't make Dall-E put a cart before a horse?

- That is an artifact of how CLIP is trained
- "CLIP is typically *not* incentivized to preserve information about the relative positions of objects, or information about which attributes apply to which objects. CLIP would therefore have a hard time distinguishing between, say, an image of a red cube on top of a blue cube and another image in which the positions of the two objects are swapped.

  The reason for this is the nature of the CLIP training objective: CLIP is only incentivized to learn the features of an image that are sufficient to match it up with the correct caption (as opposed to any of the others in the list). Unless it receives a counterexample (i.e., a caption that mentions a blue cube on top of a red cube), CLIP will not learn to preserve information about the objects' relative positions."
  -- Aditya Ramesh, co-creator of DALL-E 2
- http://adityaramesh.com/posts/dalle2/dalle2.html

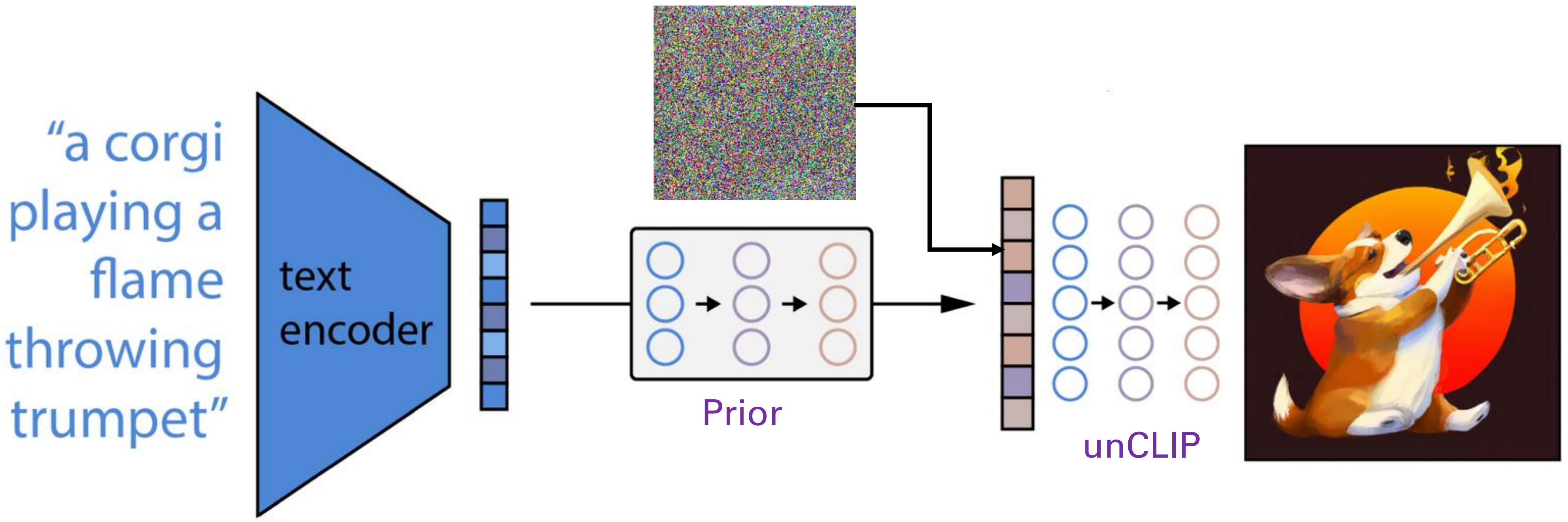
Text-to-Image generation Creating an image

- So far, our “encoder-encoder” architecture embeds image and text in a latent space
- To get images, we‘d like to invert the image encoder to generate an image from a point in latent space
  - Just like we did with clothing and faces!
- This decoder is called unCLIP
- Loosely, it is trained by fixing the weights of the CLIP image encoder and training the encoder/decoder to reproduce the original image (Again, just like before!)
- The decoder is a diffusion model, but since we want it to be able to generate a particular image, not a random image like we did with earlier diffusion models the training inputs include both
  - Successive noisings of the image (like usual for a diffusion model)
  - The CLIP embedding of the image

Text-to-Image generation
The prior

- The caption for an image and the original image won’t embed in exactly the same place in latent space
  - Remember our contrastive training
- So they train another neural net called a *prior* to predict the embedding of an image from the embedding of its caption to give a better approximation of the best point in latent space than just using the encoding of the prompt (which would be usable but not as good)

Putting it all together



Adapted from https://www.assemblyai.com/blog/how-dall-e-2-actually-works/. See link for original image

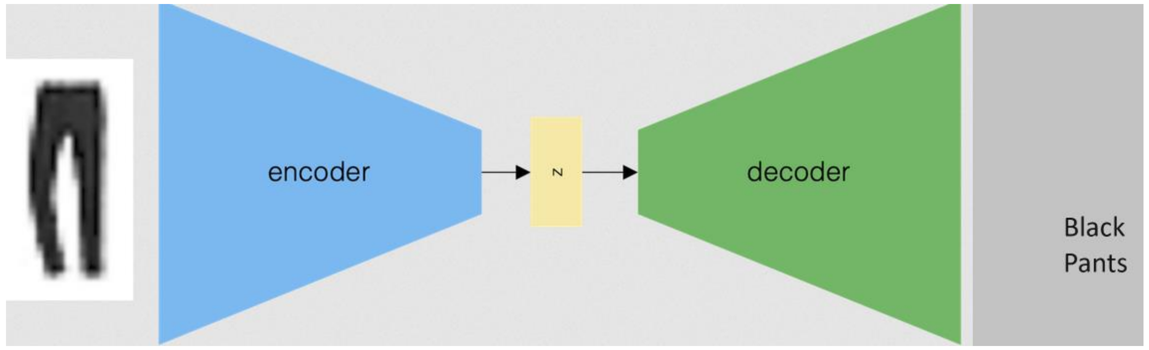
- Of course, since unCLIP is a diffusion model, it can produce many different candidate images (just choose a different example of white noise)
- Generate several and choose the image whose image embedding is closest to the prior

Dall-E 3

- The above is a description of Dall-E 2
- Let’s review the DALL-E 3 paper
- Improving Image Generation with Better Captions
  - https://cdn.openai.com/papers/dall-e-3.pdf
- Key insight: Random captions on the internet are not necessarily very good
- Generating better captions for images could improve the whole process
- How do we generate captions?

Captioning service

- The idea is simple
- We'll just sketch the main idea
- Use a caption dataset like CoCo
- Train with images in encoder and captions in decoder
- Often use a pretrained encoder



DALL-E 3 paper

- The text and images on the following slides are excerpted or paraphrased from the DALL-E 3 Paper unless otherwise noted
  - If there is a mistake, you can assume it is mine
- https://cdn.openai.com/papers/dall-e-3.pdf

DALL-E 3: Approach

- In this work, we propose a new approach to addressing prompt following: caption improvement. We hypothesize that a fundamental issue with existing text-to-image models is the poor quality of the text and image pairing of the datasets they were trained on, an issue that has been pointed out in other works such as Jia et al. (2021). We propose to address this by generating improved captions for the images in our dataset. We do this by first learning a robust image captioner which produces detailed, accurate descriptions of images. We then apply this captioner to our dataset to produce more detailed captions. We finally train text-to-image models on our improved dataset.

Adapting LLMs for captioning

- As we learned in previous lectures, LLMs are typically trained to predict the next word with an autoregressive cross-entropy loss function (Θ represents the parameters we are trying to train
  - $L(t) = \sum_j \log P(t_j | t_{j-k}, \ldots, t_{j-1}; \Theta)$
- "To turn this language model into a captioner, you need only to condition on the image. The challenge here is that images are composed of many thousands of pixel values. Conditioning on all of this information is exceptionally inefficient with our current neural networks, so we need a compressed representation space. Conveniently, CLIP provides just this. Thus, given a pre-trained CLIP image embedding function $F(i)$, we augment our language model objective as follows"
  - $L(t, i) = \sum_j \log P(t_j | t_{j-k}, \ldots, t_{j-1}; z_j; F(i); \Theta)$
- (t, i) are text and image pairs from our training set

Train and *voila*!

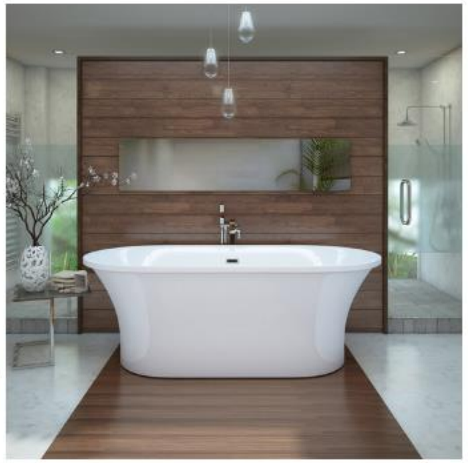
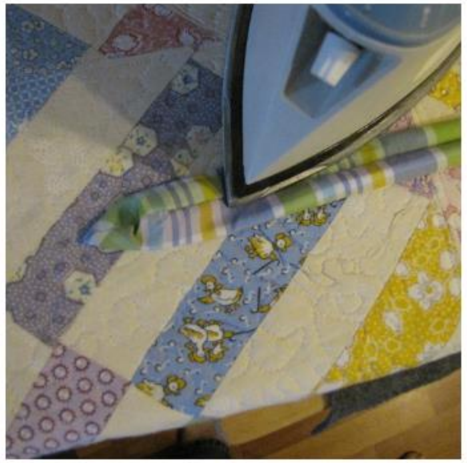
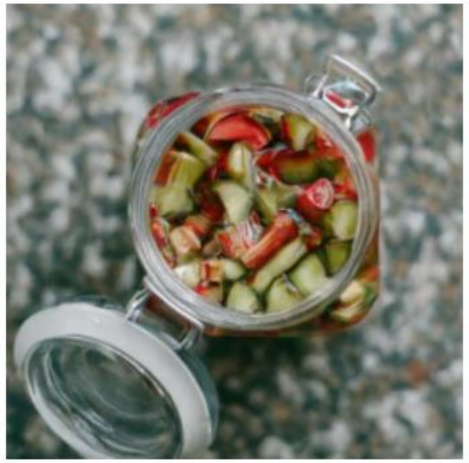
Image			
Alt Text	now at victorian plumbing.co.uk	is he finished...just about!	23 (19 of 30) 1200
SSC	a white modern bathtub sits on a wooden floor.	a quilt with an iron on it.	a jar of rhubarb liqueur sitting on a pebble background.
DSC	this luxurious bathroom features a modern freestanding bathtub in a crisp white finish. the tub sits against a wooden accent wall with glass-like panels, creating a serene and relaxing ambiance. three pendant light fixtures hang above the tub, adding a touch of sophistication. a large window with a wooden panel provides natural light, while a potted plant adds a touch of greenery. the freestanding bathtub stands out as a statement piece in this contemporary bathroom.	a quilt is laid out on a ironing board with an iron resting on top. the quilt has a patchwork design with pastel-colored strips of fabric and floral patterns. the iron is turned on and the tip is resting on top of one of the strips. the quilt appears to be in the process of being pressed, as the steam from the iron is visible on the surface. the quilt has a vintage feel and the colors are yellow, blue, and white, giving it an antique look.	rhubarb pieces in a glass jar, waiting to be pickled. the colors of the rhubarb range from bright red to pale green, creating a beautiful contrast. the jar is sitting on a gravel background, giving a rustic feel to the image.

Figure 3 – Examples of alt-text accompanying selected images scraped from the internet, short synthetic captions (SSC), and descriptive synthetic captions (DSC).

After training the captioner, what happens to image generation?

- “The above experiments suggest that we can maximize the performance of our models by training on a very high percentage of synthetic captions. However, doing so causes the models to naturally adapt to the distribution of long, highly-descriptive captions emitted by our captioner.

  Generative models are known to produce poor results when sampled out of their training distribution. Thus, to extract the maximum potential out of our models, we will need to exclusively sample from them with highly descriptive captions”

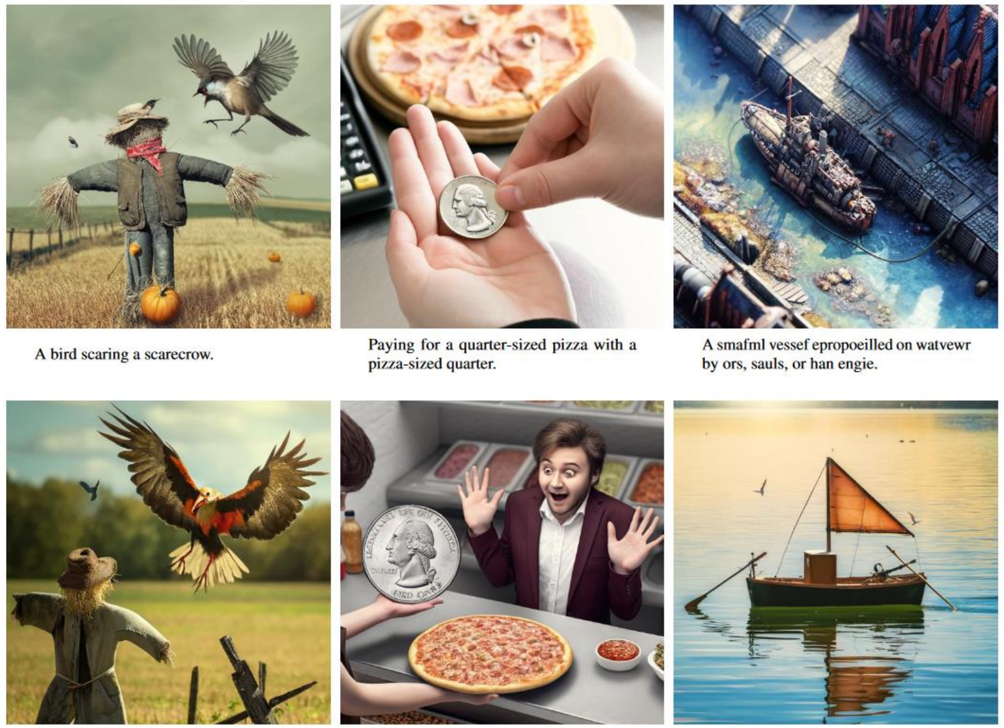
Only works with detailed prompts? Hmm

- Fortunately, this is a solvable problem with recent breakthroughs in large language models. Models like GPT-4[14] have become exceptionally good at tasks that require imagination, such as telling stories and writing poems. It stands to reason that they might also be good at coming up with plausible details in an image description.

  Indeed, given a prompt such as the one found in [Prompt Engineering slide below], we found that GPT-4 will readily "upsample“ any caption into a highly descriptive one. To demonstrate how this approach might be useful, we perform this procedure on the captions from the drawbench dataset[24] and visualize the results in Table 7.

  As can be seen [on the next slide], utilizing a LLM to "upsample" captions can be used to not only add missing details, but also to disambiguate complex relationships which would be hard for a (relatively) small image generation model to learn. The end result is that the model will often correctly render images that it would have otherwise gotten wrong

Image generation with and without ChatGPT "upsampling" prompts



A bird scaring a scarecrow.

Paying for a quarter-sized pizza with a pizza-sized quarter.

A smafml vessef epropoeilled on watvewr by ors, sauls, or han engie.

A large, vibrant bird with an impressive wingspan swoops down from the sky, letting out a piercing call as it approaches a weathered scarecrow in a sunlit field. The scarecrow, dressed in tattered clothing and a straw hat, appears to tremble, almost as if it's coming to life in fear of the approaching bird.

A person is standing at a pizza counter, holding a gigantic quarter the size of a pizza. The cashier, wide-eyed with astonishment, hands over a tiny, quarter-sized pizza in return. The background features various pizza toppings and other customers, all of them equally amazed by the unusual transaction.

A small vessel, propelled on water by oars, sails, or an engine, floats gracefully on a serene lake. the sun casts a warm glow on the water, reflecting the vibrant colors of the sky as birds fly overhead.

Figure 6 – Effect of using "upsampled" drawbench captions to create samples with DALL-E 3. Original drawbench captions on top, upsampled captions on bottom. Images are best of 4 for each caption.

Prompt engineering to create the upsampled prompt

Following is the prompt we give to GPT-4 before feeding it an image caption for "upsampling".

```
You are part of a team of bots that creates images . You work with an assistant bot that will draw anything you say in square brackets . For example ,
outputting " a beautiful morning in the woods with the sun peaking through the trees " will trigger your partner bot to output an image of a forest
morning , as described . You will be prompted by people looking to create detailed , amazing images . The way to accomplish this is to take their
short prompts and make them extremely detailed and descriptive .

There are a few rules to follow :
- You will only ever output a single image description per user request .
- Sometimes the user will request that you modify previous captions . In this case , you should refer to your previous conversations with the user \
  and make the modifications requested .
- When modifications are requested , you should not simply make the description longer . You should refactor the entire description to integrate the \
  suggestions .
- Other times the user will not want modifications , but instead want a new image . In this case , you should ignore your previous conversation with\
  the user.
- Image descriptions must be between 15 -80 words . Extra words will be ignored .

{" role ": " user " , " content ": " Create an imaginative image descriptive caption or modify an earlier caption for the user input : " a man holding a
sword \""} ,
{" role ": " assistant " , " content ": " a pale figure with long white hair stands in the center of a dark  forest , holding a sword high above his \
   head . the blade glows with a blue light , casting a soft glow on the trees and bushes surrounding him ."} ,
{" role ": " user " , " content ": " Create an imaginative image descriptive caption or modify an earlier caption for the user input : " make the light red "} ,
{" role ": " assistant " , " content ": " a pale figure with long white hair stands in the center of a dark forest ,  holding a sword high above his \
   head .   the blade glows with a red light , casting a warm glow on the trees and bushes surrounding him ."} ,
{" role ": " user " , " content ": " Create an imaginative image descriptive caption or modify an earlier caption for the user input : \
   " draw a frog playing dominoes "} ,
{" role ": " assistant " , " content ": " a frog sits on a worn table playing a game of dominoes with an elderly raccoon . the table is covered in a \
   green cloth , and the frog is wearing a jacket and a pair of jeans . The scene is set in a forest , with a large tree in the background ."}
```

BEYOND IMAGES: 3D

3D data

- University of Chicago’s 3DL lab run by Hana Ranocka
- Is laser-focused on 3D generation and deep learning
- It turns out that generating 3D data is surprisingly different than generating 2D images
- Let’s look at Hana Ranocka’s ICCV23 keynote to understand why and what can be done about it

NEXT WEEK SPECIAL GUEST

Virtual Lecture with a special guest

Will be on zoom

Don’t come to classroom

HW 7-1

- Build your own GPT on a topic of your choice
- Analyze what it does well and what it doesn’t
- Include some screenshots or conversations in your submission
- Be prepared to present it in class if asked
- **Note:** This will require using the ChatGPT website (chat.openai.com) and may require a subscription. While many of you have such a subscription, some of you do not. If you need a subscription for a month, let me know and we will help

HW 7-2: Optional

- Create your own image captioning service
  - See slide 44 for some hints