Lecture 11: Diffusion Models for Image Generation (Gen AI)

15.S04: Hands-on Deep Learning
Spring 2024
Farias, Ramakrishnan

You

Create a photorealistic image of graduate students in a class on deep learning

ChatGPT

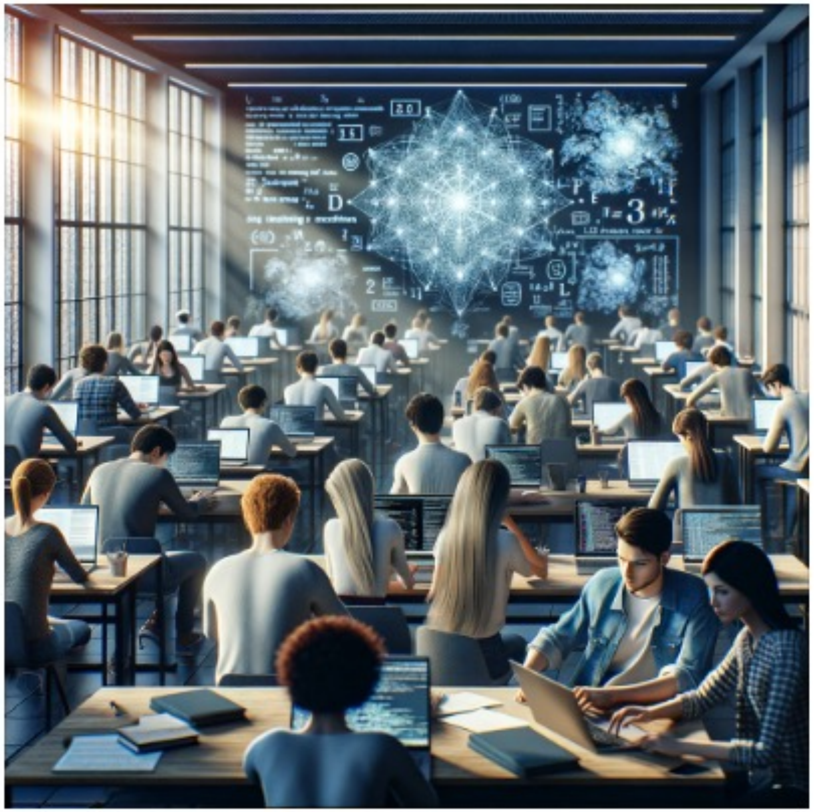


Image generated using ChatGPT from the prompt from the prompt quoted above.
Grad students in classroom image with prompt by instructor: OpenAI (2024). ChatGPT (DALL-E 3).

I've created a photorealistic image of graduate students in a class on deep learning. You can see the students engaging with their laptops and tablets, with a digital blackboard in the front displaying complex diagrams and equations related to deep learning.

Text-to-image

A quaint Italian seaside village with colorful buildings, boats, and the reflection of the setting sun on the water, in the impressionist style of Claude Monet, with visible brush strokes and dappled light.

Text-to-Image

Image generated using Midjourney from the prompt quoted above. Italian seaside image with prompt by Alex Serban on MSPowerUser: Midjourney (2024). Midjourney (V6).

https://mspoweruser.com/best-midjourney-prompts/

Text-to-Video

In an ornate, historical hall, a massive tidal wave peaks and begins to crash. Two surfers, seizing the moment, skillfully navigate the face of the wave.

*Released February 15, 2024 (https://openai.com/sora)

Wave/surfer video with prompt by Brooks, Peebles, et al. on OpenAI: OpenAI (2024). ChatGPT Plus (SORA).
Video generated using SORA from the prompt in an ornate, historical hall, a massive tidal wave peaks and begins to crash. Two surfers, seizing the moment, skillfully navigate the face of the wave.

OpenAI Sora

"… we train text-conditional diffusion models jointly on videos and images of variable durations, resolutions and aspect ratios. We leverage a transformer architecture that operates on spacetime patches of video and image latent codes …"

Wave/surfer video with prompt by Brooks, Peebles, et al. on OpenAI: OpenAI (2024). ChatGPT Plus (SORA).
Video generated using SORA from the prompt in an ornate, historical hall, a massive tidal wave peaks and begins to crash. Two surfers, seizing the moment, skillfully navigate the face of the wave.

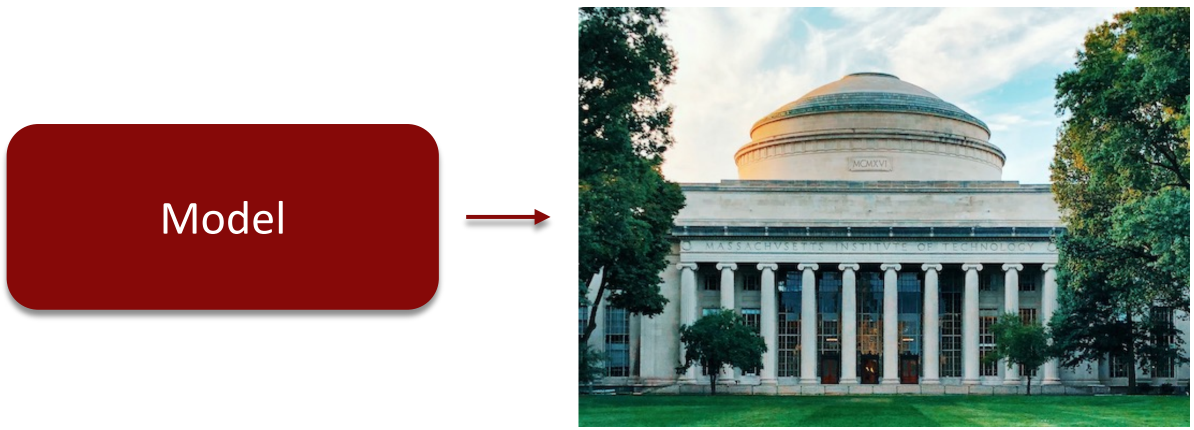
https://openai.com/research/video-generation-models-as-world-simulators

Our journey

- How to generate a good image

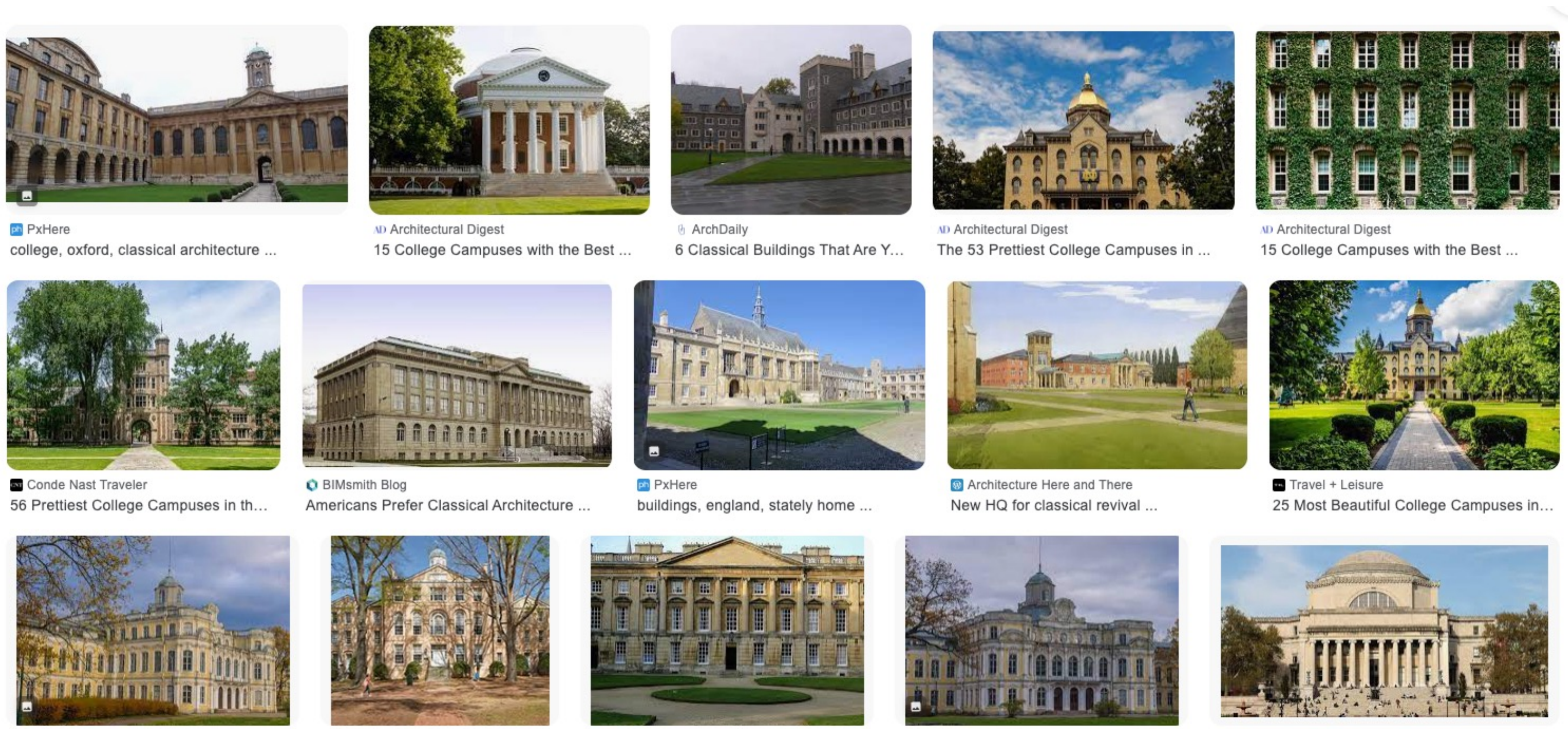
- How to “control” the image generation process with a text prompt

Let’s say we want to build a model that can be used to generate images of stately college buildings



MIT Dome: Photo by Muzammil Soorma on Unsplash.

Assume we have millions of training images at our disposal



Source: Google Image Search with query = "stately college buildings"

Each time we generate an image from the model, we would like it to be different. How can we guarantee this?

Model →

MIT Dome: Photo by Muzammil Soorma on Unsplash.

MIT Dome: Photo by [Muzammil Soorma](#) on [Unsplash](#)

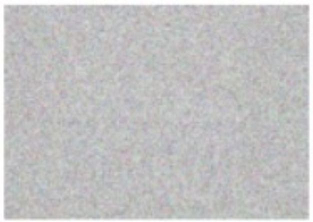
Model →

The Rotunda at UVA image is in the public domain. Source: Wikimedia Commons. https://commons.wikimedia.org/wiki/File:University_of_Virginia_Rotunda_2006.jpg

https://en.wikipedia.org/wiki/Rotunda_%28architecture%29

We will create a “noise” image by setting each pixel value to a random number and input that. Since “noise” images are random, the inputs will vary

“Noise”

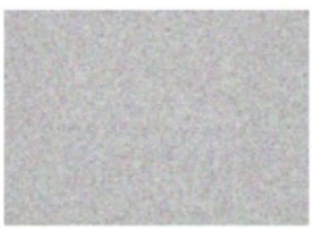


Model

MIT Dome: Photo by Muzammil Soorma on Unsplash.

MIT Dome: Photo by Muzammil Soorma on Unsplash

“Noise”

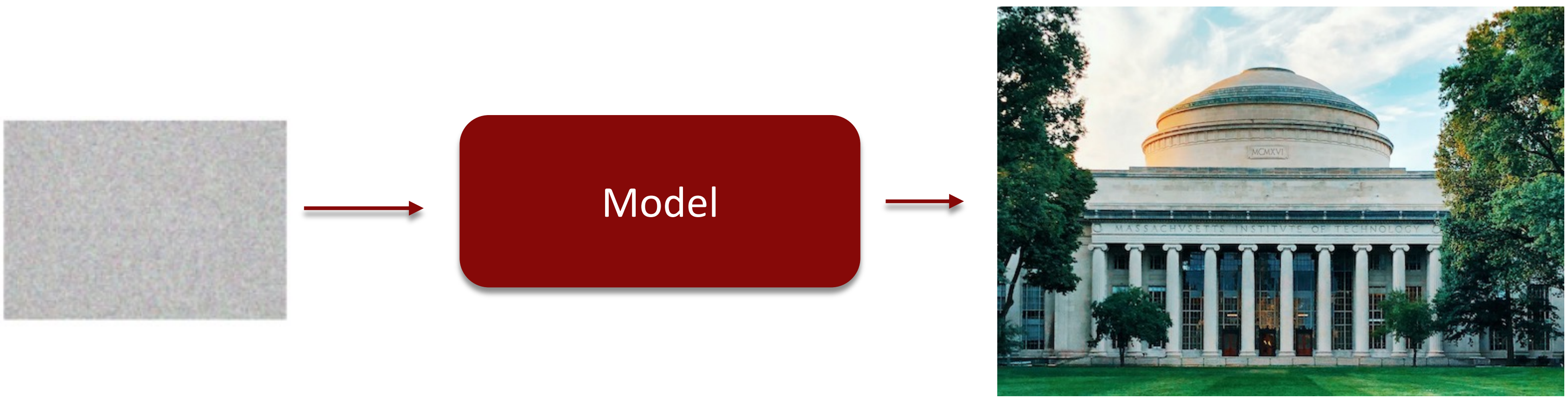


Model

The Rotunda at UVA image is in the public domain. Source: Wikimedia Commons. https://commons.wikimedia.org/wiki/File:University_of_Virginia_Rotunda_2006.jpg

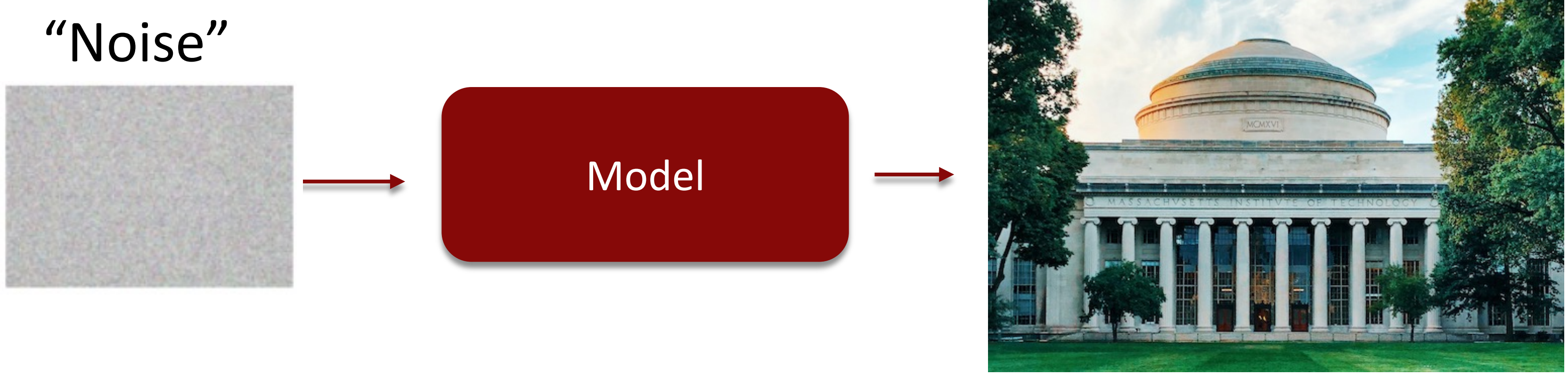
https://en.wikipedia.org/wiki/Rotunda_%28architecture%29

How can we train a model to generate an image from pure noise?



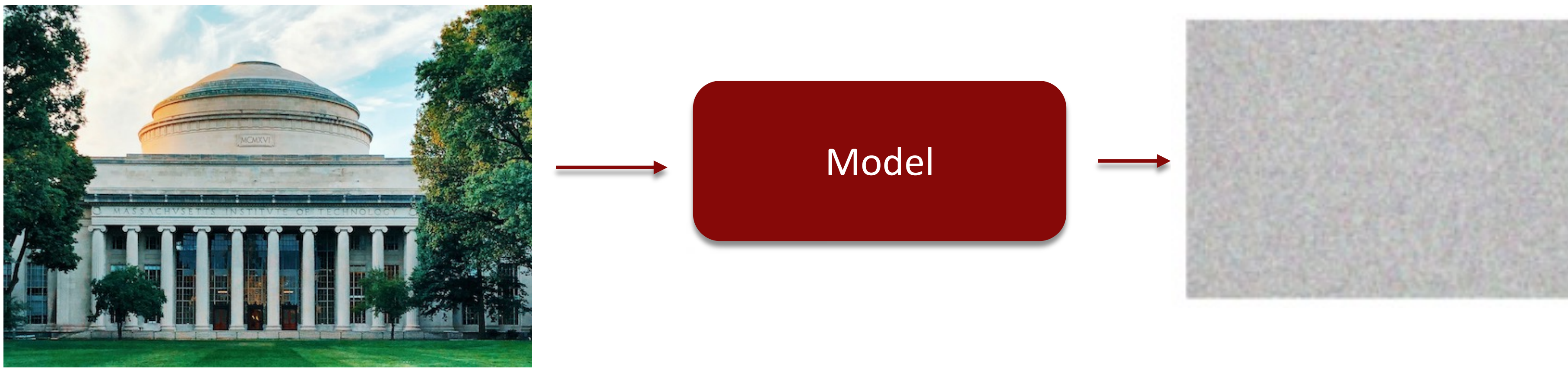
MIT Dome: Photo by Muzammil Soorma on Unsplash.

It is not clear how to do this …



MIT Dome: Photo by Muzammil Soorma on Unsplash.

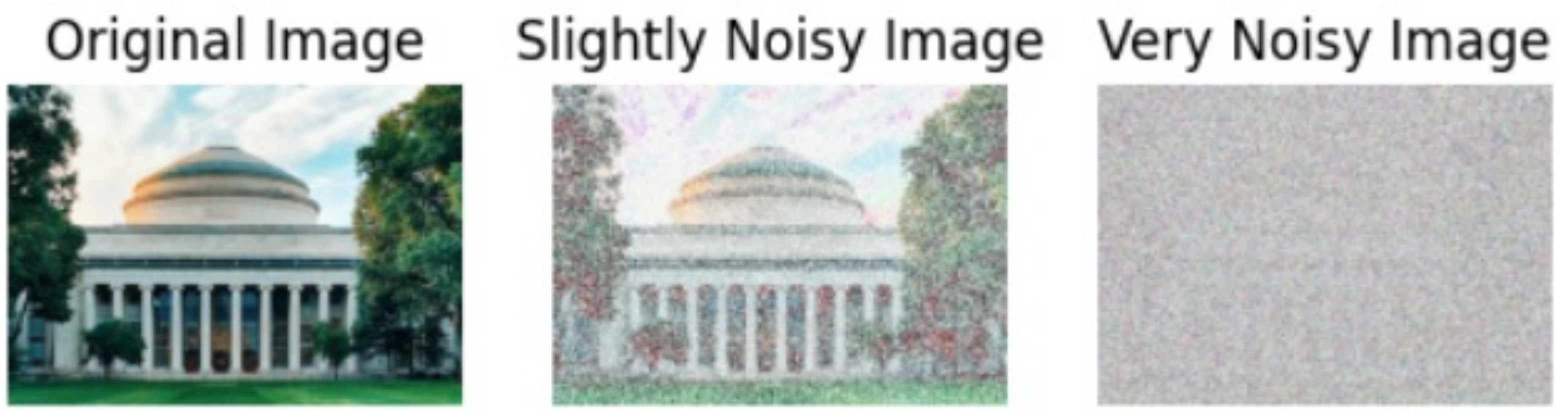
… but how about the reverse?



MIT Dome: Photo by Muzammil Soorma on Unsplash.

Given an image, can we create a "noisy" version of it?

Yes! We know how to add noise to an image!



MIT Dome: Photo by Muzammil Soorma on Unsplash.

Just add random numbers to every pixel!

(Colab link)

By increasing the magnitude of these random numbers, we can make the image noisier

MIT Dome: Photo by Muzammil Soorma on Unsplash.

(Colab link)

That suggests an idea!

We can take each image in the training set and create many noisy versions of it

MIT Dome: Photo by Muzammil Soorma on Unsplash.

We can create (x,y) training data from these images!

?

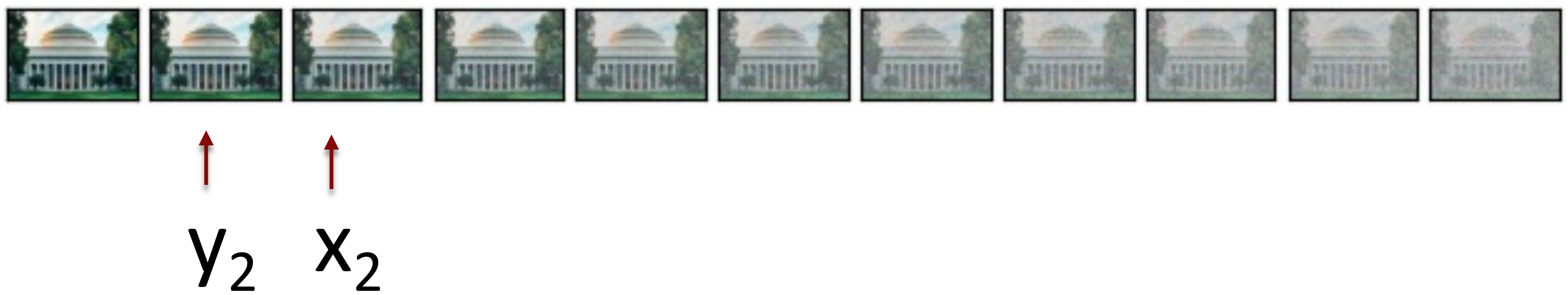
MIT Dome: Photo by Muzammil Soorma on Unsplash.

We can create (x,y) training data from these images!

y_1 x_1

MIT Dome: Photo by Muzammil Soorma on Unsplash.

We can create (x,y) training data from these images!



MIT Dome: Photo by Muzammil Soorma on Unsplash.

We can create (x,y) training data from these images!

MIT Dome: Photo by Muzammil Soorma on Unsplash.

We can create (x,y) training data from these images!

What’s the relationship between x and y?

MIT Dome: Photo by Muzammil Soorma on Unsplash.

We can create (x,y) training data from these images!

y_{10} x_{10}

x = image

y = "less noisy" version of the image

MIT Dome: Photo by Muzammil Soorma on Unsplash.

What can we do with these (x,y) pairs?

x = image

y = "less noisy" version of the image

MIT Dome: Photo by Muzammil Soorma on Unsplash.

We can build a de-noising model …

… by running stochastic gradient descent on the training data (as usual)

An “image de-noising” model!

image

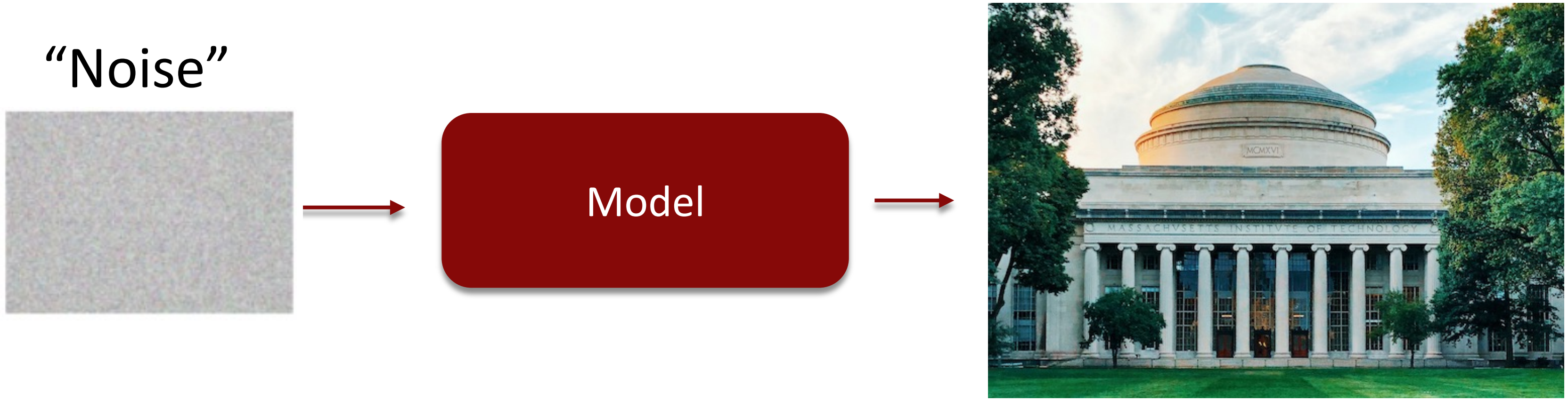
“less noisy” version of the image

x → DNN → y

MIT Dome: Photo by Muzammil Soorma on Unsplash.

After this de-noising model is trained …

… we can solve this problem.



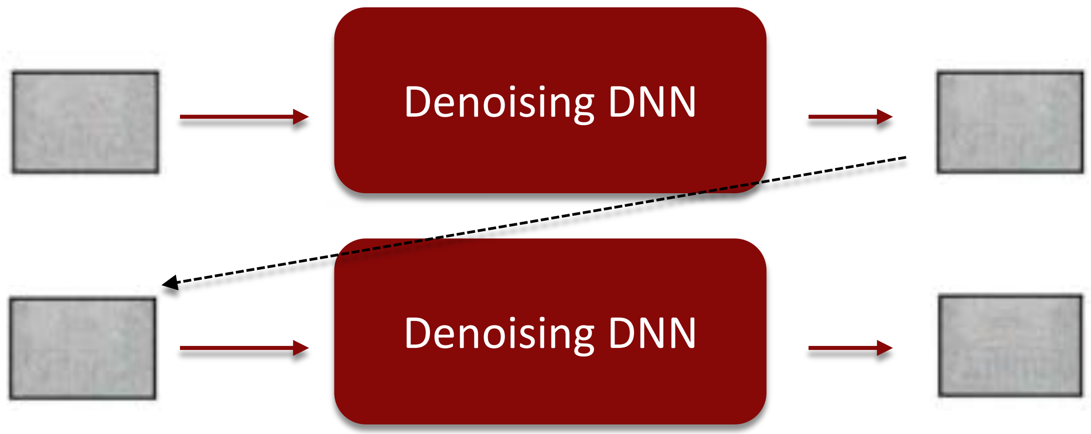
MIT Dome: Photo by Muzammil Soorma on Unsplash.

How?

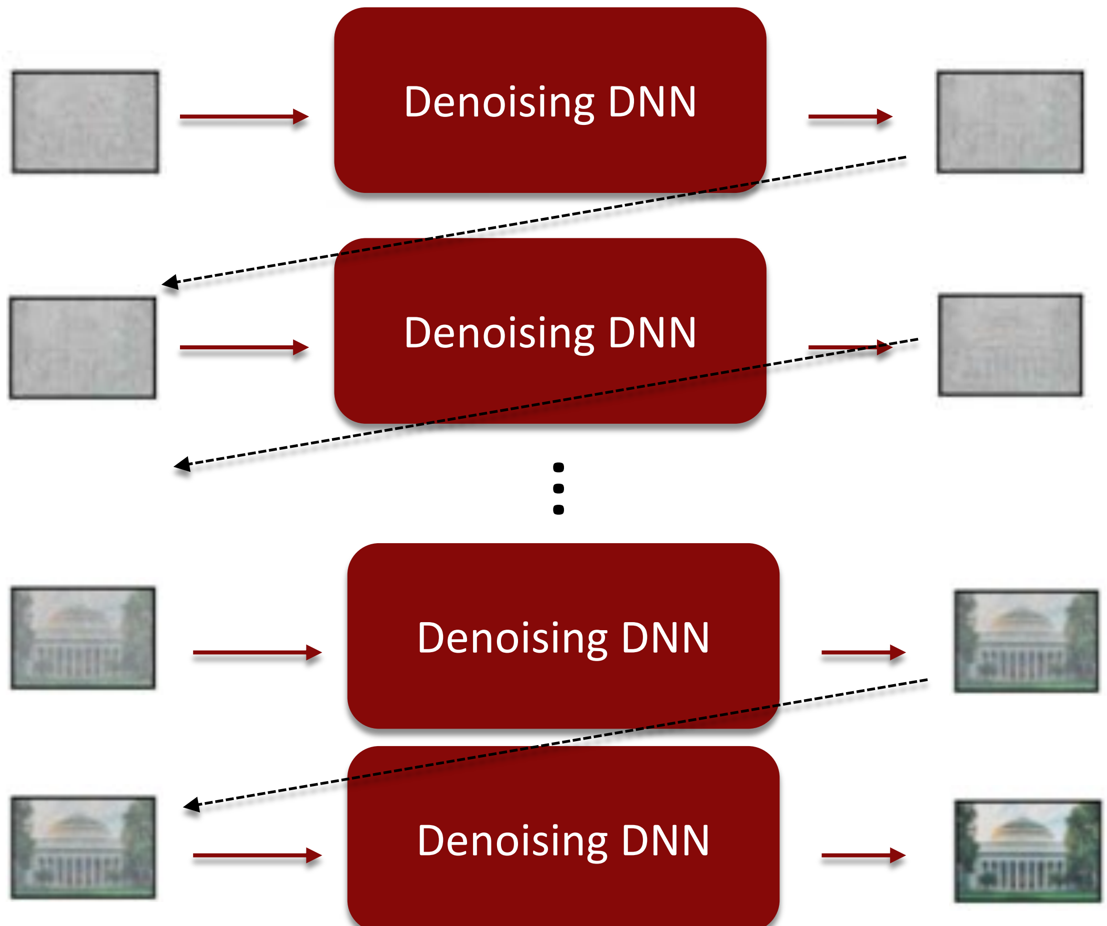
Start with “pure” noise and <u>repeatedly</u> denoise it!

Start with “pure” noise and <u>repeatedly</u> denoise it!

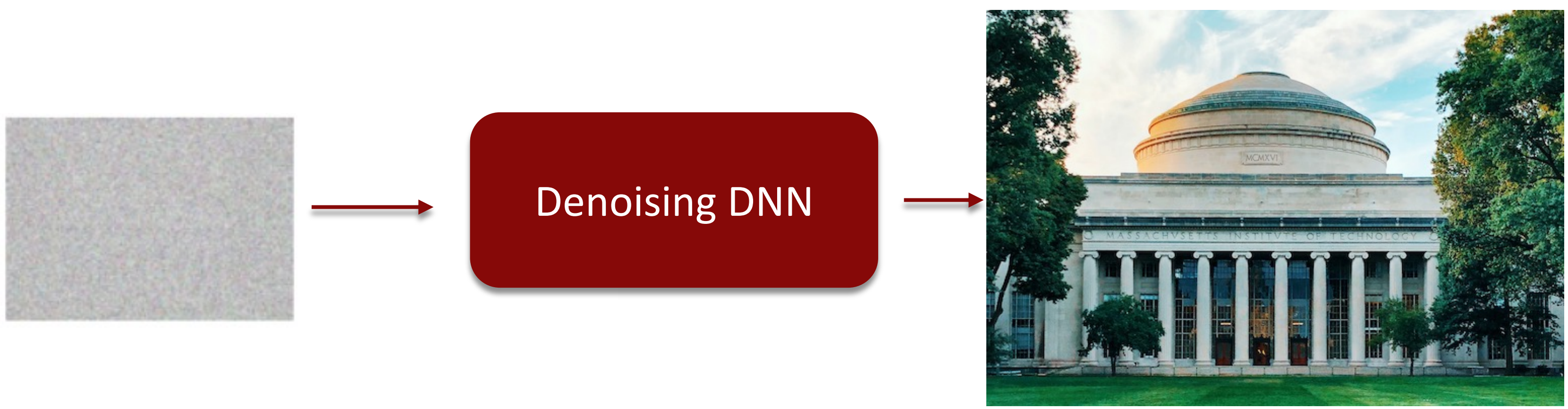
Start with "pure" noise and repeatedly denoise it!



Start with "pure" noise and repeatedly denoise it!



The model will generate a sequence of “less noisy” images. The final one is the “answer”.



MIT Dome: Photo by Muzammil Soorma on Unsplash.

This is called a *diffusion* model

Deep Unsupervised Learning using Nonequilibrium Thermodynamics

Jascha Sohl-Dickstein JASCHA@STANFORD.EDU
Stanford University

Eric A. Weiss EAWEISS@BERKELEY.EDU
University of California, Berkeley

Niru Maheswaranathan NIRUM@STANFORD.EDU
Stanford University

Surya Ganguli SGANGULI@STANFORD.EDU
Stanford University

https://arxiv.org/pdf/1503.03585.pdf

Key Improvement: Instead of training the model to predict the “less noisy” version of the image, we ask it to predict the “noise” and then subtract the noise from the input

x → DNN → ϵ

image

(“noise”)

“less noisy” version of the image

$$y = x - \epsilon$$

Denoising Diffusion Probabilistic Models

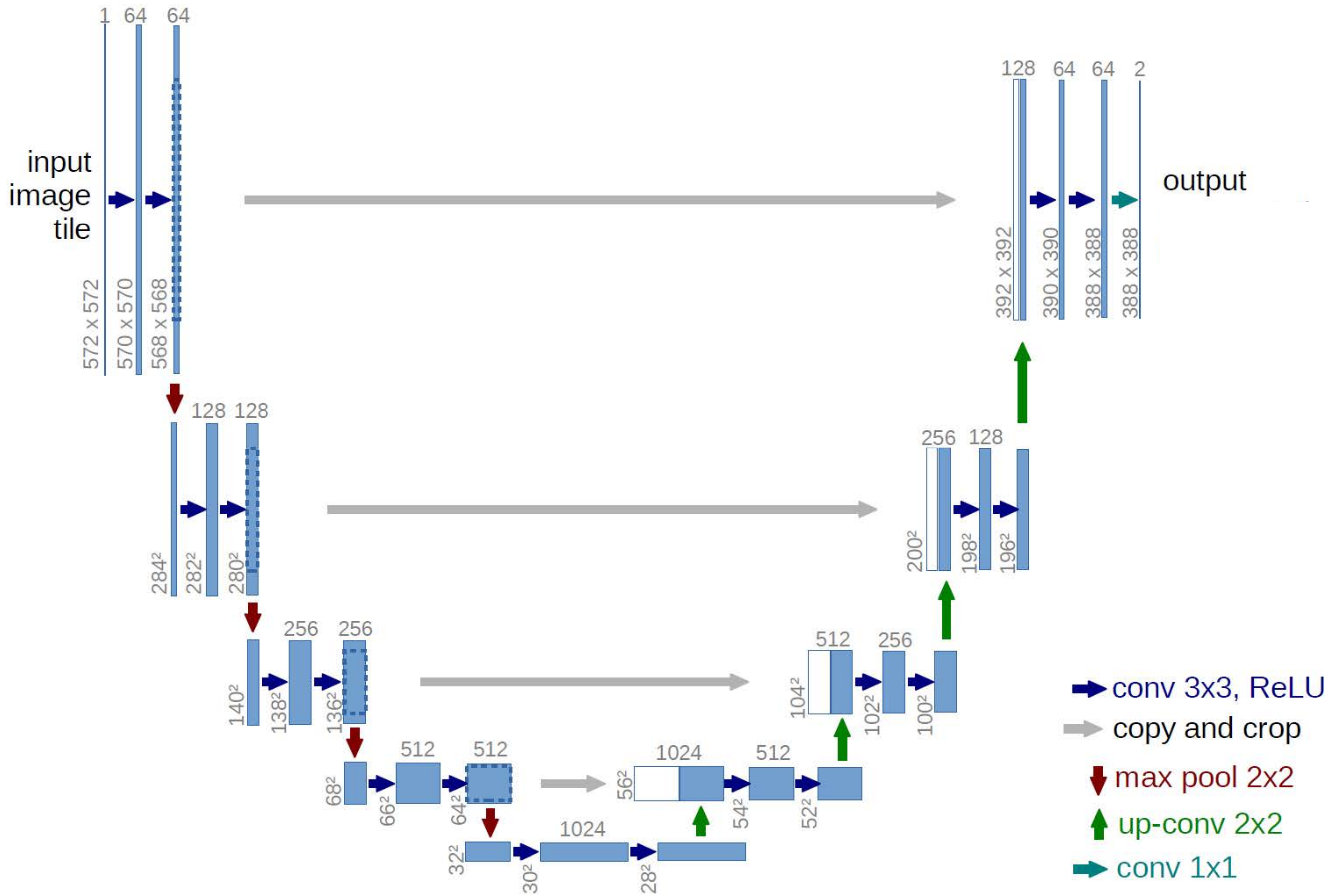
Jonathan Ho
UC Berkeley
jonathanho@berkeley.edu

Ajay Jain
UC Berkeley
ajayj@berkeley.edu

Pieter Abbeel
UC Berkeley
pabbeel@cs.berkeley.edu

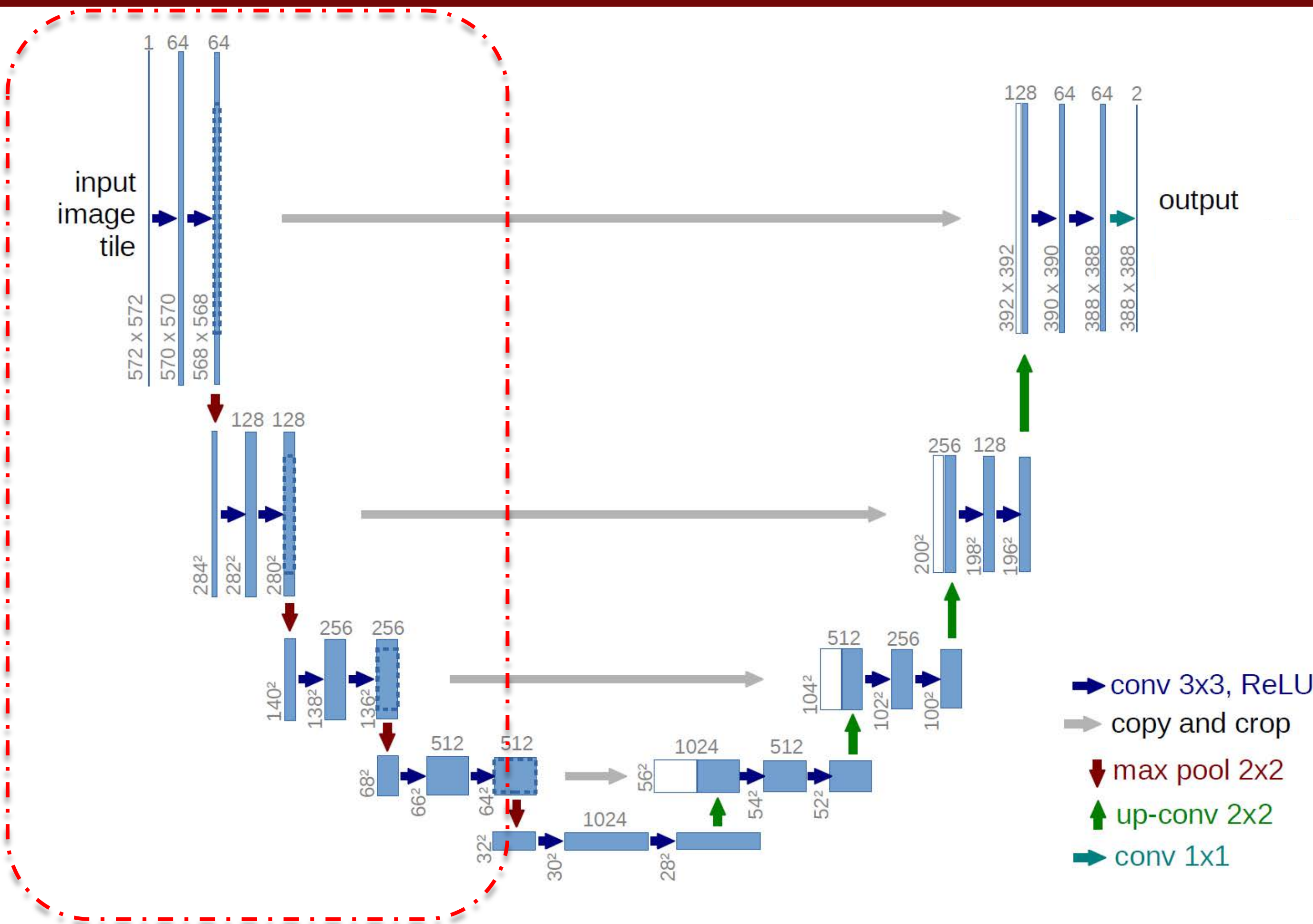
https://arxiv.org/abs/2006.11239

The U-Net architecture is commonly used for image-to-image models like the diffusion model



https://lmb.informatik.uni-freiburg.de/people/ronneber/u-net

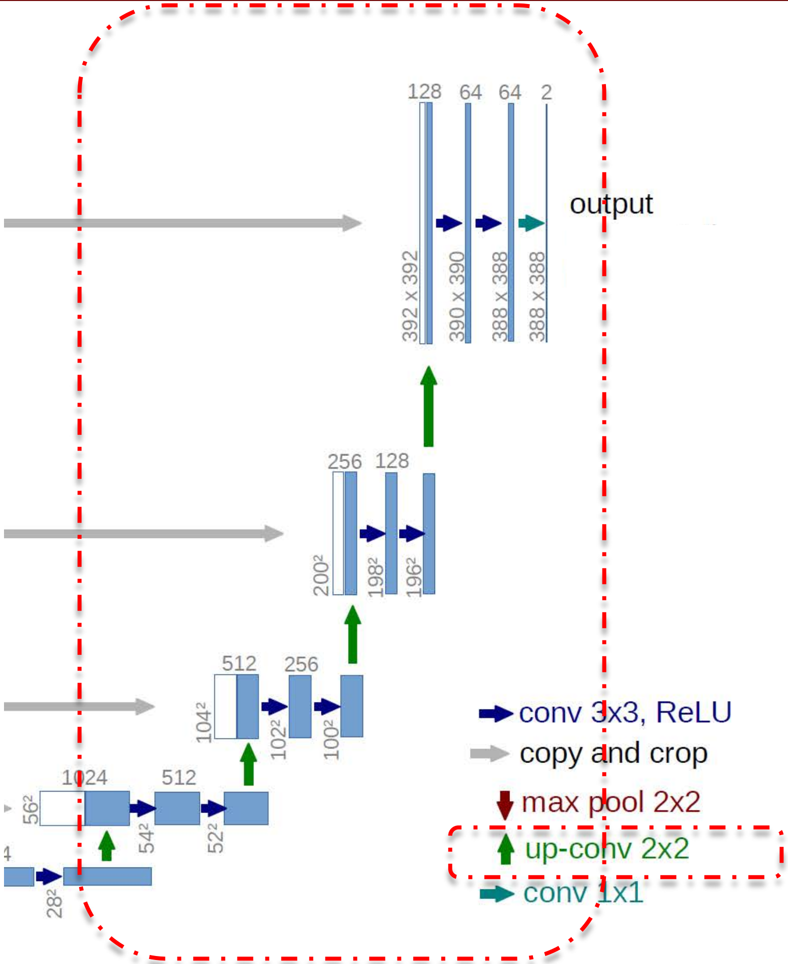
The “left half” of the U uses standard convolutional and pooling layers



The “right half” of the U uses a new kind of convolutional layer

We need to “enlarge” back to the original size.

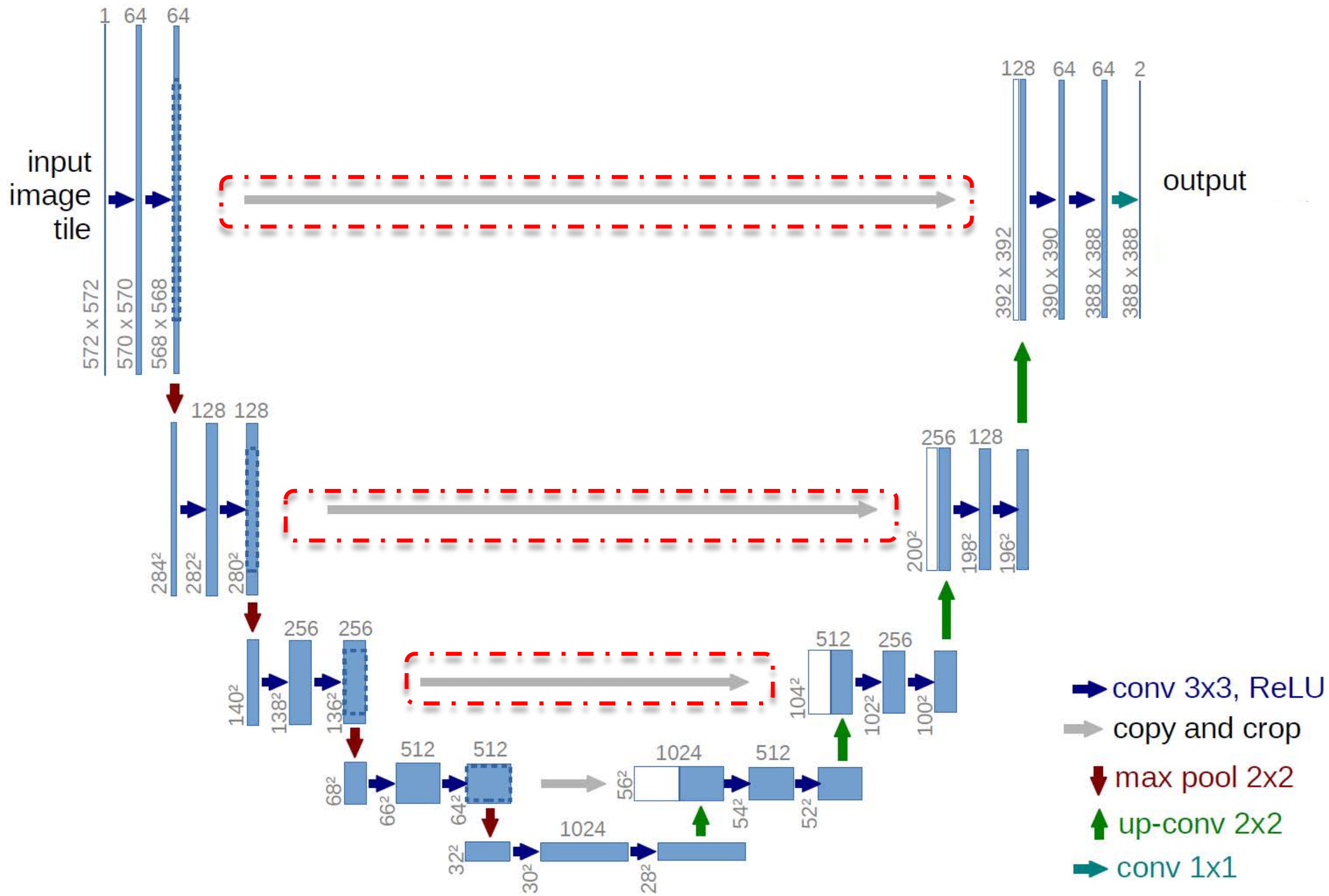
We use an ‘inverse’ of the convolution layer called an Up-Convolution (or De-Convolution) layer*



https://lmb.informatik.uni-freiburg.de/people/ronneber/u-net

*implemented as `Conv2DTranspose` in Keras (see 9.2 in the textbook)

The U-Net uses cross-connections as well (like residual connections)



https://lmb.informatik.uni-freiburg.de/people/ronneber/u-net

Our journey

- How to generate a good image
- How to “control” the image generation process with a <u>text prompt</u>

Intuition

- We want to take the text prompt into account when generating the image
- If we had a “rough” image that corresponds to the text prompt, we can feed it in and try to denoise it. But we don’t!

Intuition

- We want to take the text prompt into account when generating the image
- If we had a "rough" image that corresponds to the text prompt, we can feed it in and try to denoise it. But we don't!
- But what if we had an embedding for the prompt that is "close" to the embeddings of all the images that correspond to that prompt?

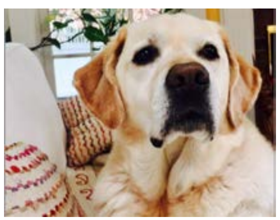
Intuition

- We want to take the text prompt into account when generating the image
- If we had a "rough" image that corresponds to the text prompt, we can feed it in and try to denoise it. But we don't!
- But what if we had an embedding for the prompt that is "close" to the embeddings of all the images that correspond to that prompt?
- If we feed this embedding to our de-noising model, maybe it can use it to "steer" the generation process? After all, this embedding is already "close" to the embedding of the image we want to generate?

We will describe an approach to calculate an embedding for any text prompt that is “close” to the the embeddings of the images that correspond to that prompt

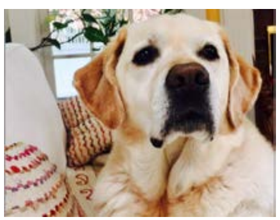
Let’s say we have an image and a caption. How can we compute embeddings from them?

A yellow Labrador retriever on a cushion → ? →

 → ? →

We can use standard architectures!

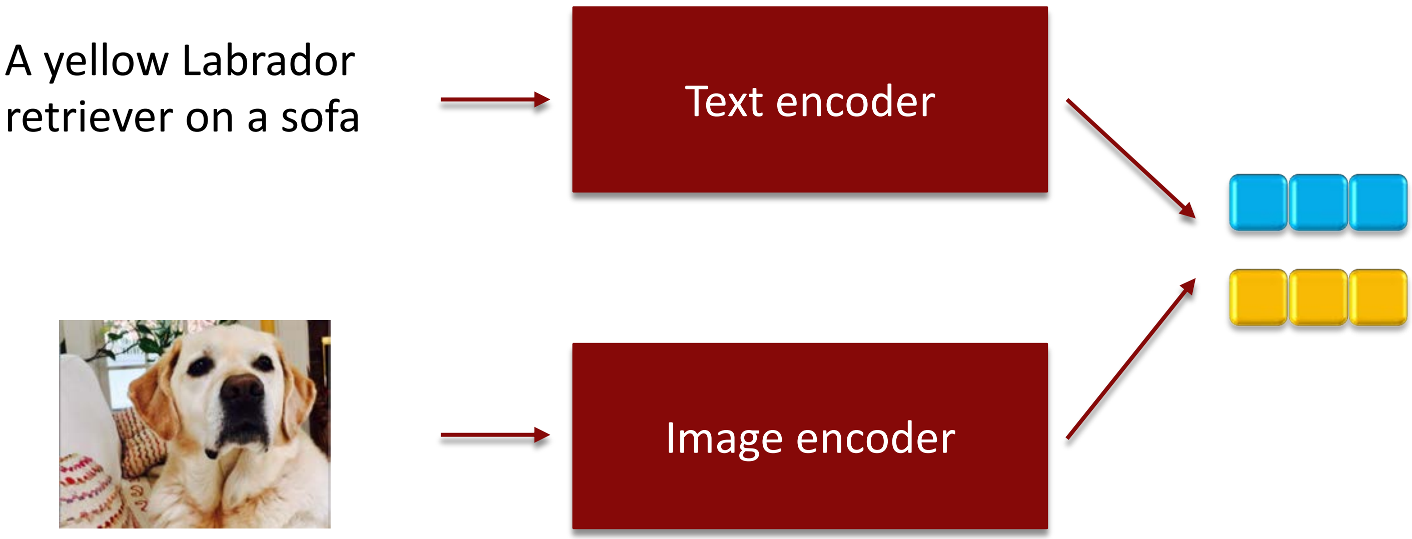
A yellow Labrador retriever on a cushion → Text encoder (like BERT) →



→ Image encoder (like ResNet) →

But these embeddings need to satisfy <u>two</u> important requirements

Given an image and a caption that describes it, the embeddings should be "close" to each other



Given an image and an unrelated caption, we want their embeddings to be "far" from each other

A train rounding a bend with beautiful Fall foliage all around

Text encoder

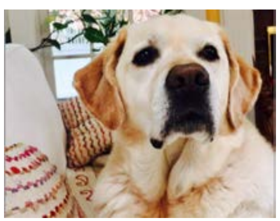


Image encoder

This ensures that the text embedding and the image embedding are referring to the same underlying concept i.e., "live" in the same "concept space"

This ensures that the text embedding and the image embedding are referring to the same underlying concept i.e., “live” in the same “concept space”

i.e., it ensures that the embedding for any text prompt is “close” to the embeddings of the images that correspond to that prompt

Question: How can we tell how “close” two embeddings are?

Question: How can we tell how "close" two embeddings are?

Answer: Their cosine similarity

OK, we know how to measure "closeness".

How can we compute embeddings that satisfy the two requirements?

OpenAI built a model called CLIP (Contrastive Language-Image Pretraining) to solve this problem

Learning Transferable Visual Models From Natural Language Supervision

Alec Radford [*1] Jong Wook Kim [*1] Chris Hallacy [1] Aditya Ramesh [1] Gabriel Goh [1] Sandhini Agarwal [1]
Girish Sastry [1] Amanda Askell [1] Pamela Mishkin [1] Jack Clark [1] Gretchen Krueger [1] Ilya Sutskever [1]

https://arxiv.org/pdf/2103.00020.pdf

How CLIP works

1. Set a 12-layer, 8-head Transformer Causal Encoder stack as the text encoder. Set ResNet-50 as the image encoder.*
2. Initialize with random weights

*with some small tweaks

How CLIP works

1. Set a 12-layer, 8-head Transformer Causal Encoder stack as the text encoder. Set ResNet-50 as the image encoder.*

2. Initialize with random weights

3. Grab a batch of <image, caption> pairs

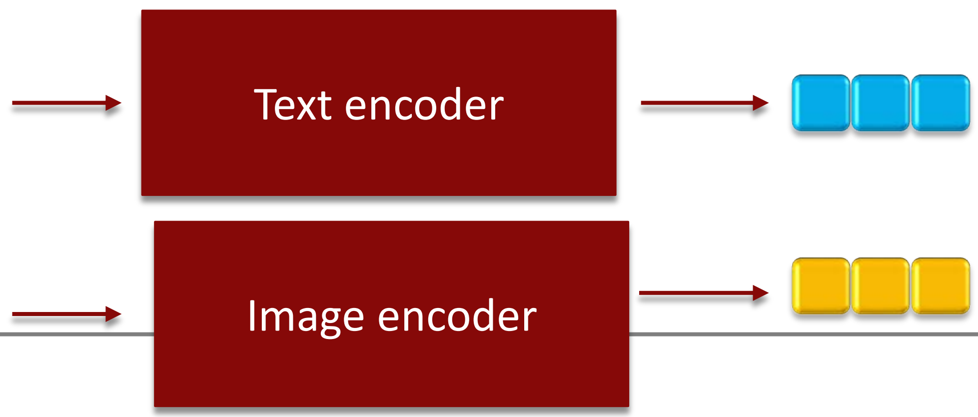
A yellow Labrador retriever on a sofa

A blue jay with an acorn in its mouth

A train rounding a bend with beautiful Fall foliage all around

How CLIP works

1. Set a 12-layer, 8-head Transformer Causal Encoder stack as the text encoder. Set ResNet-50 as the image encoder.*
2. Initialize with random weights
3. Grab a batch of <image, caption> pairs
4. Run the images through the image encoder and the captions through the text encoder and get embeddings



How CLIP works

1. Set a 12-layer, 8-head Transformer Causal Encoder stack as the text encoder. Set ResNet-50 as the image encoder.*
2. Initialize with random weights
3. Grab a batch of <image, caption> pairs
4. Run the images through the image encoder and the captions through the text encoder and get embeddings
5. With the embeddings, calculate the "closeness score" (i.e., cosine similarity) for <u>every</u> image-caption pair

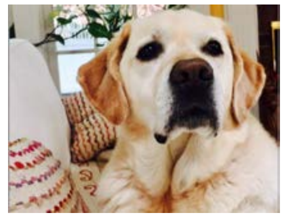
Image-Caption Closeness Scores

A yellow Labrador retriever on a cushion

A blue jay with an acorn in its mouth

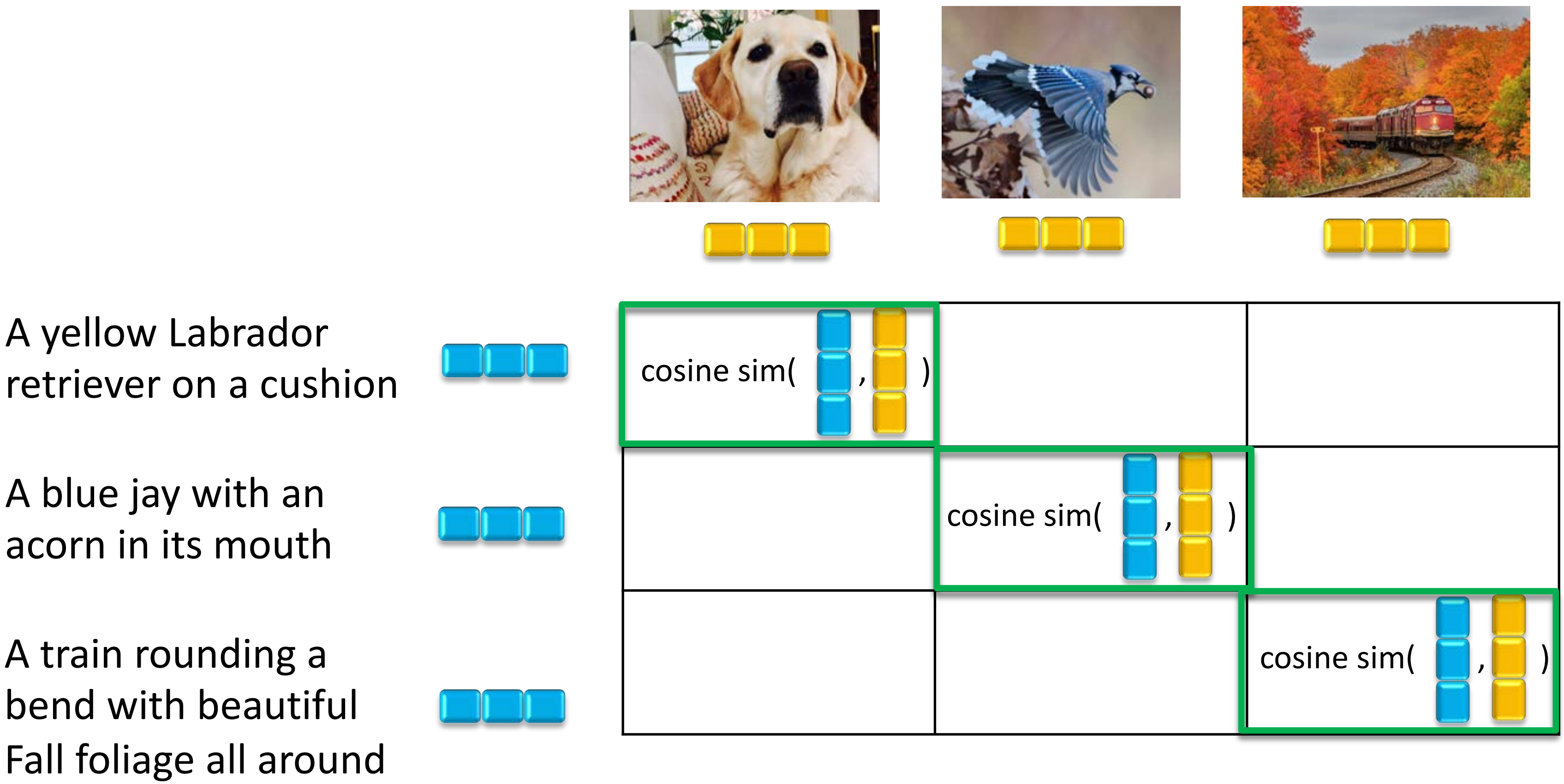
A train rounding a bend with beautiful Fall foliage all around

Image-Caption Closeness Scores



A yellow Labrador retriever on a cushion	cosine sim(,)	cosine sim(,)	cosine sim(,)
A blue jay with an acorn in its mouth	cosine sim(,)	cosine sim(,)	cosine sim(,)
A train rounding a bend with beautiful Fall foliage all around	cosine sim(,)	cosine sim(,)	cosine sim(,)

We want these scores to be as high as possible



We want to:

maximize the sum of the green cells

Equivalent loss function:

minimize [– sum of green cells]

We want to:
maximize the sum of the green cells

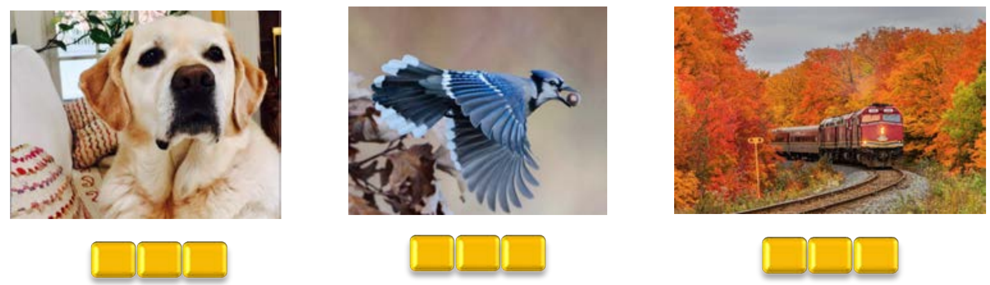
Equivalent loss function:
minimize [– sum of green cells]

Will this loss function do the trick?

Will this loss function do the trick?

No. The optimizer can simply ignore the input, make all the embeddings the same and achieve a perfect cosine similarity of 1.0 for all pairs of embeddings!

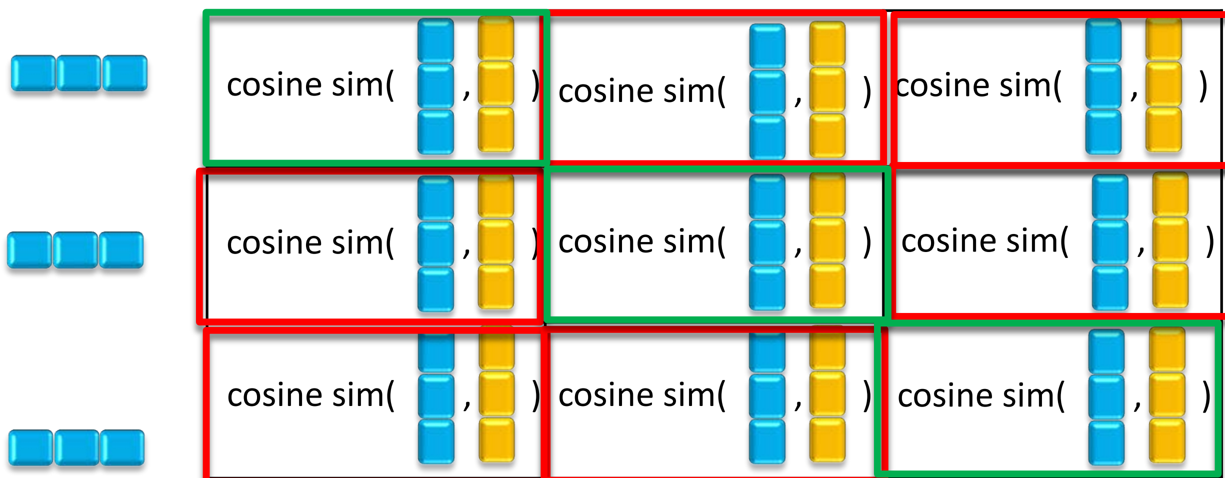
We want the scores of the off-diagonal cells to be as small as possible as well



A yellow Labrador retriever on a cushion

A blue jay with an acorn in its mouth

A train rounding a bend with beautiful Fall foliage all around



We want to:

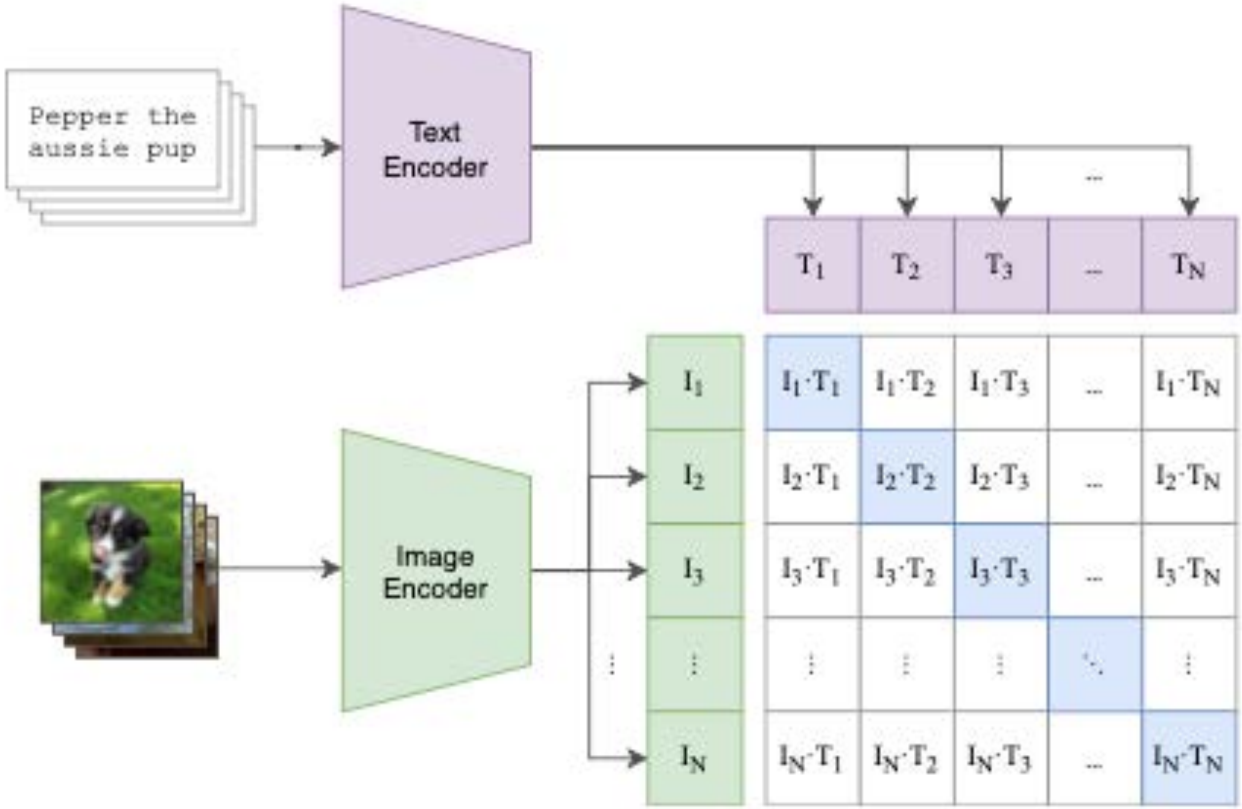
maximize the sum of the green cells

and *minimize* the sum of the red cells

Equivalent loss function:

minimize [sum of red cells – sum of green cells]

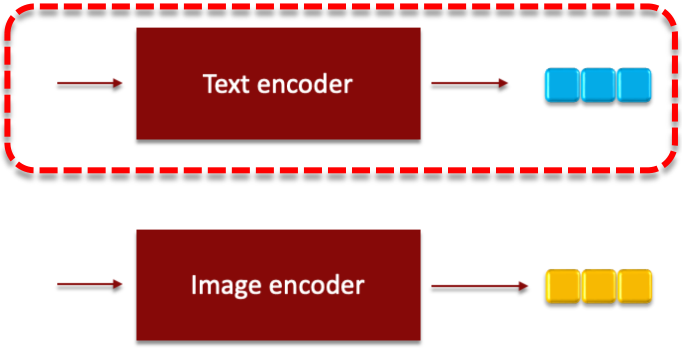
Here's the official picture from the CLIP paper 🙂



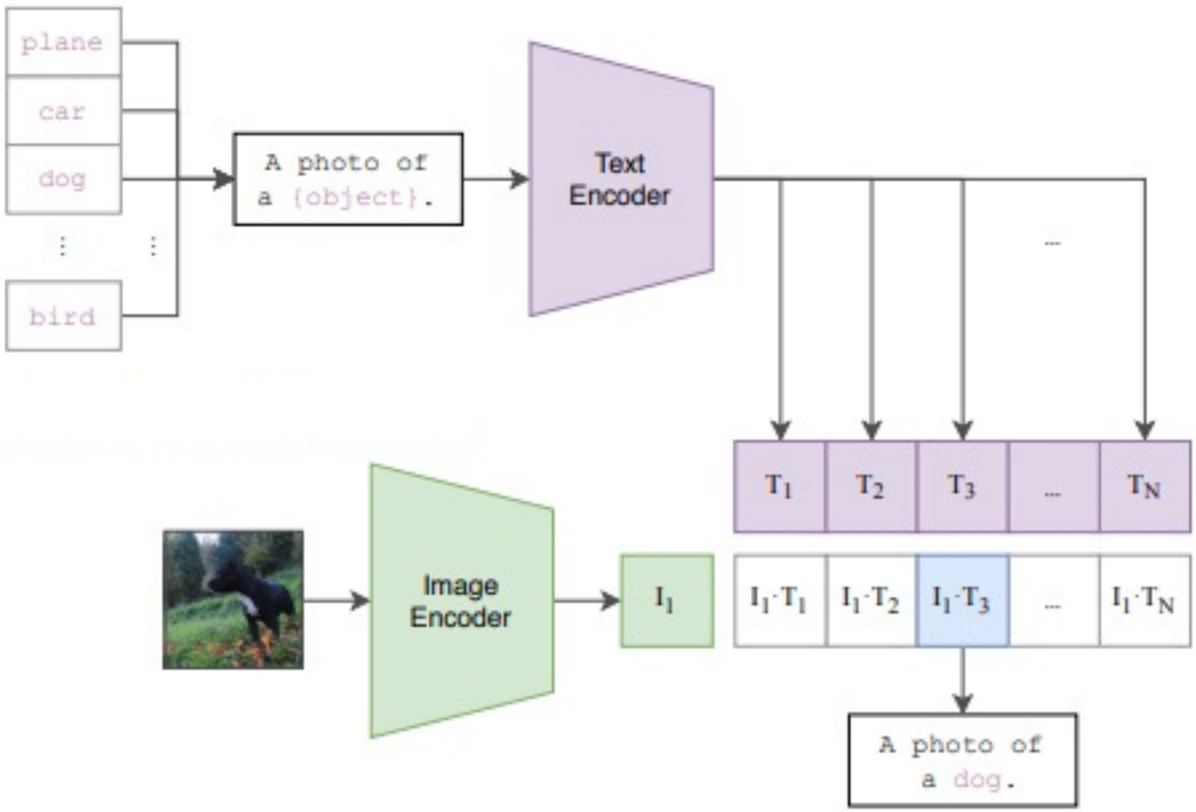
https://arxiv.org/pdf/2103.00020.pdf

By minimizing this loss function on a training dataset of 400 million <image, caption> pairs scraped from the Internet, CLIP was built!

We can use CLIP's text encoder by itself. We can send in any text and get an embedding that is close to the embedding of any image described by that text



BTW, CLIP can be used for zero-shot image classification



https://arxiv.org/pdf/2103.00020.pdf

Next: We will see how to “control” the image generation process using the CLIP text embedding of a text prompt

Recall the training (x,y) pairs we assembled to train the noise-to-image diffusion model

y_1 x_1 y_{10} x_{10}

x = image

y = "less noisy" version of the image

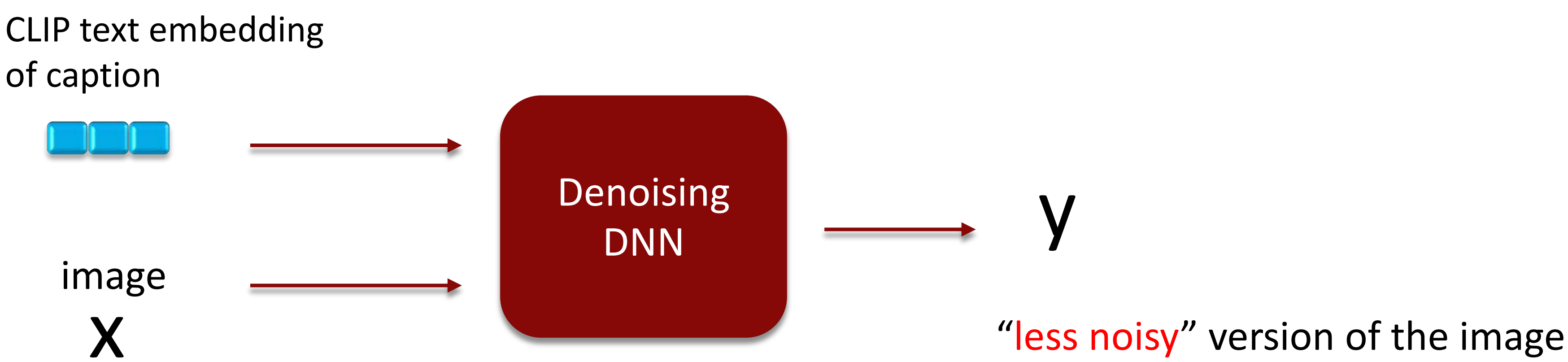
What if expand x to include its caption …

x = [image, CLIP text embedding of caption]

y = “less noisy” version of the image

MIT Dome: Photo by Muzammil Soorma on Unsplash.

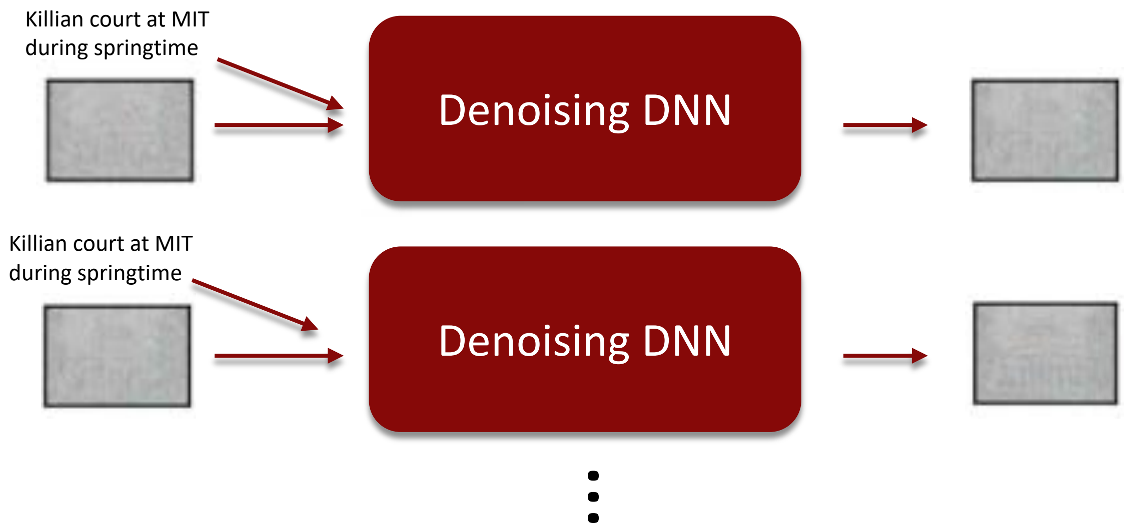
… and train the model like before!



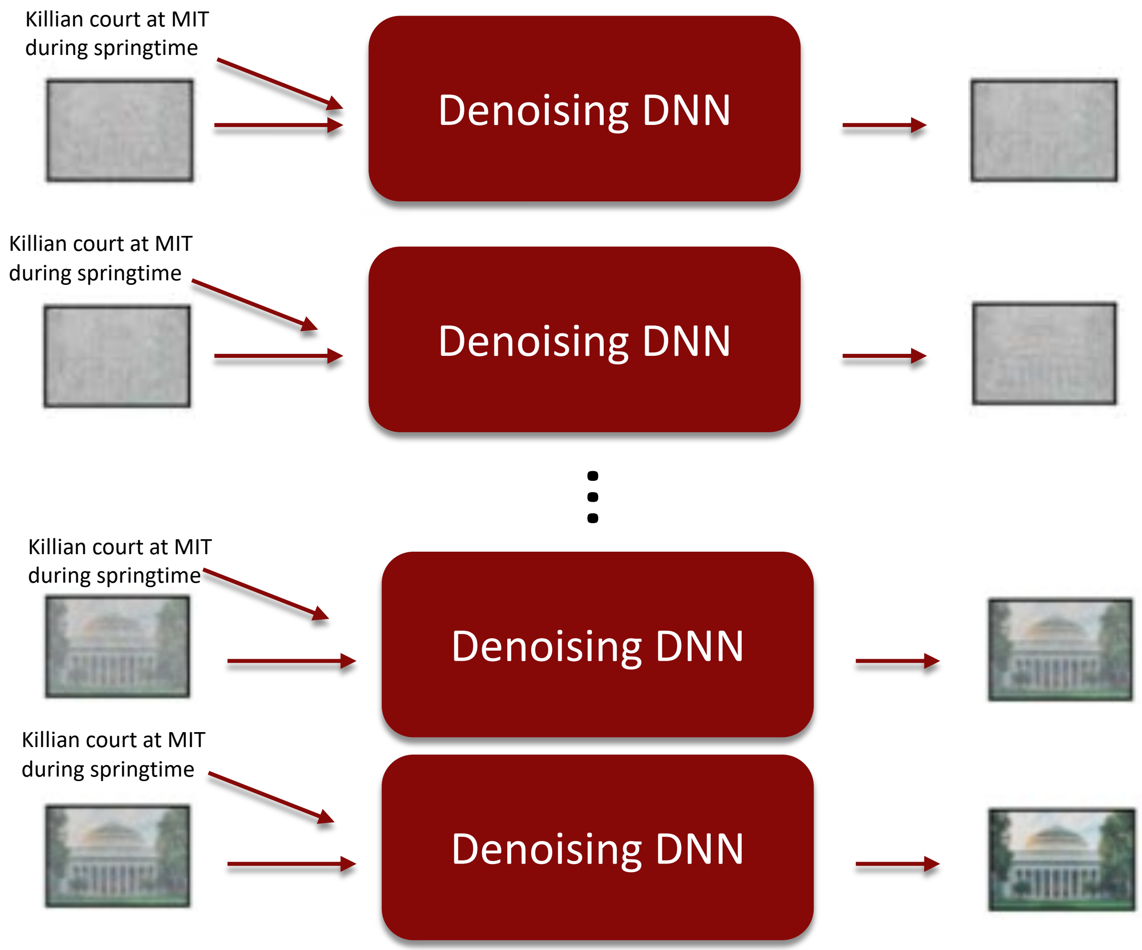
After this model is trained …

… start with "pure" noise and a prompt and repeatedly denoise it!

… start with “pure” noise and a prompt and repeatedly denoise it!

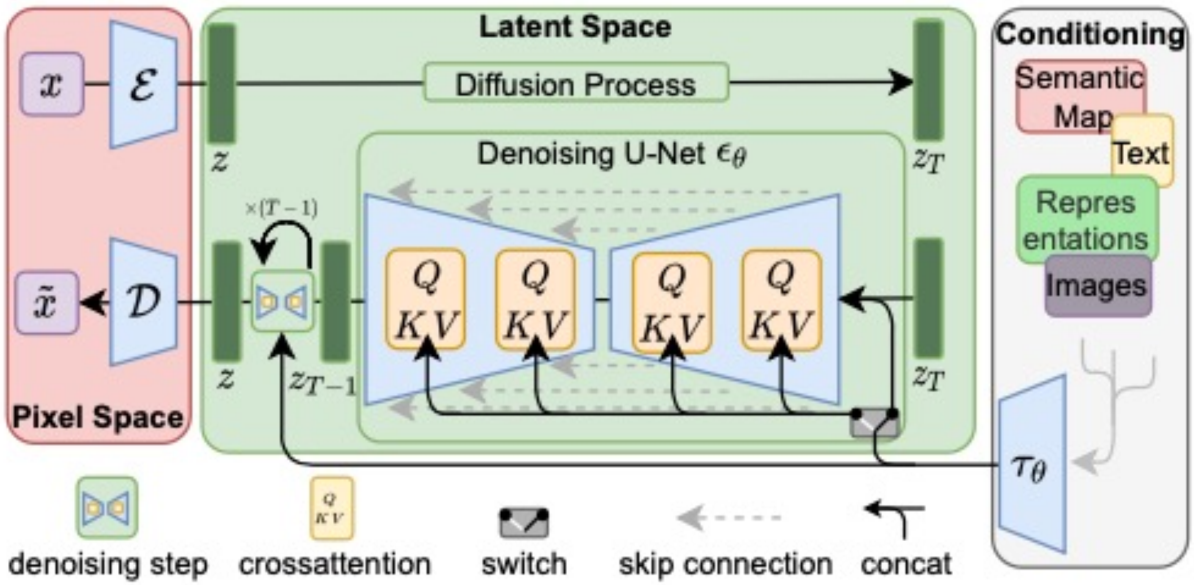


… start with "pure" noise <u>and a prompt</u> and repeatedly denoise it!



Let's take a quick look inside the

Denoising DNN

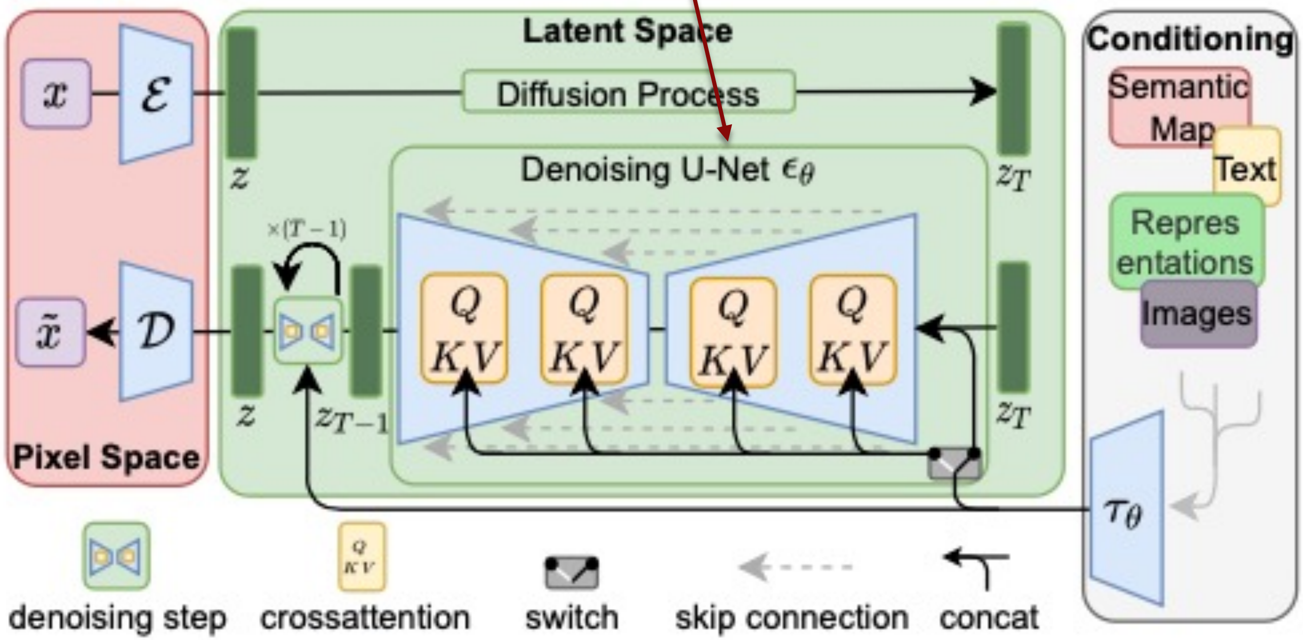


High-Resolution Image Synthesis with Latent Diffusion Models

Robin Rombach[1] * Andreas Blattmann[1] * Dominik Lorenz[1] Patrick Esser[R] Björn Ommer[1]
[1]Ludwig Maximilian University of Munich & IWR, Heidelberg University, Germany [R]Runway ML
https://github.com/CompVis/latent-diffusion

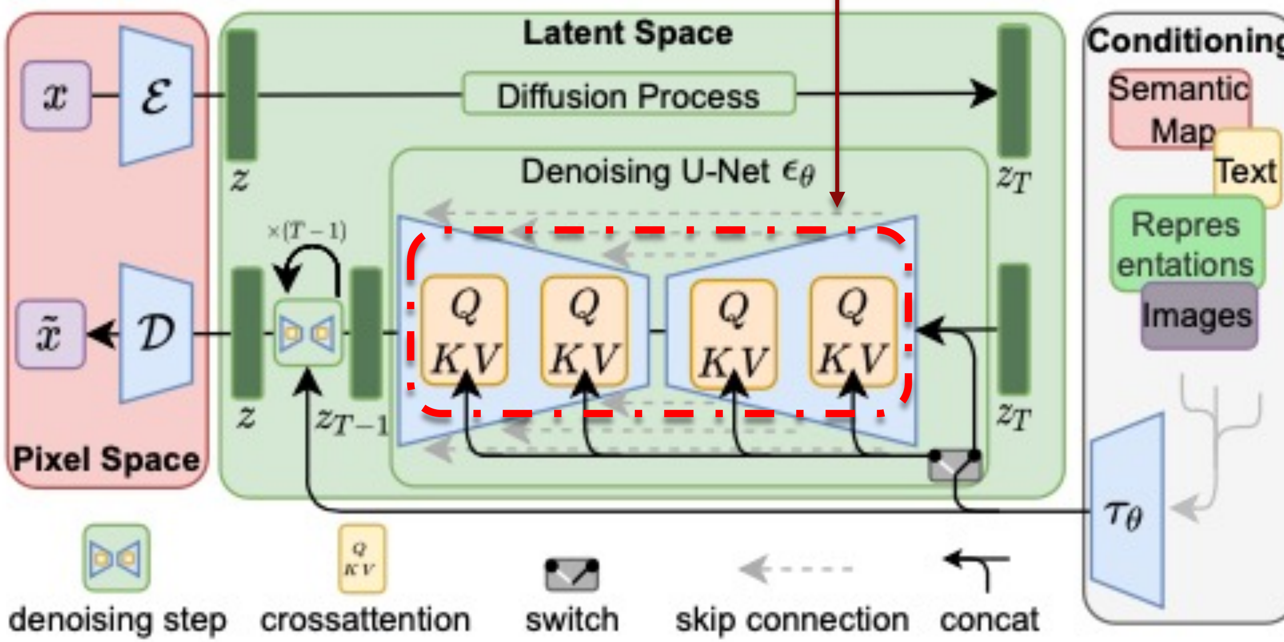
https://arxiv.org/abs/2112.10752

(1) Notice that it uses the U-Net architecture



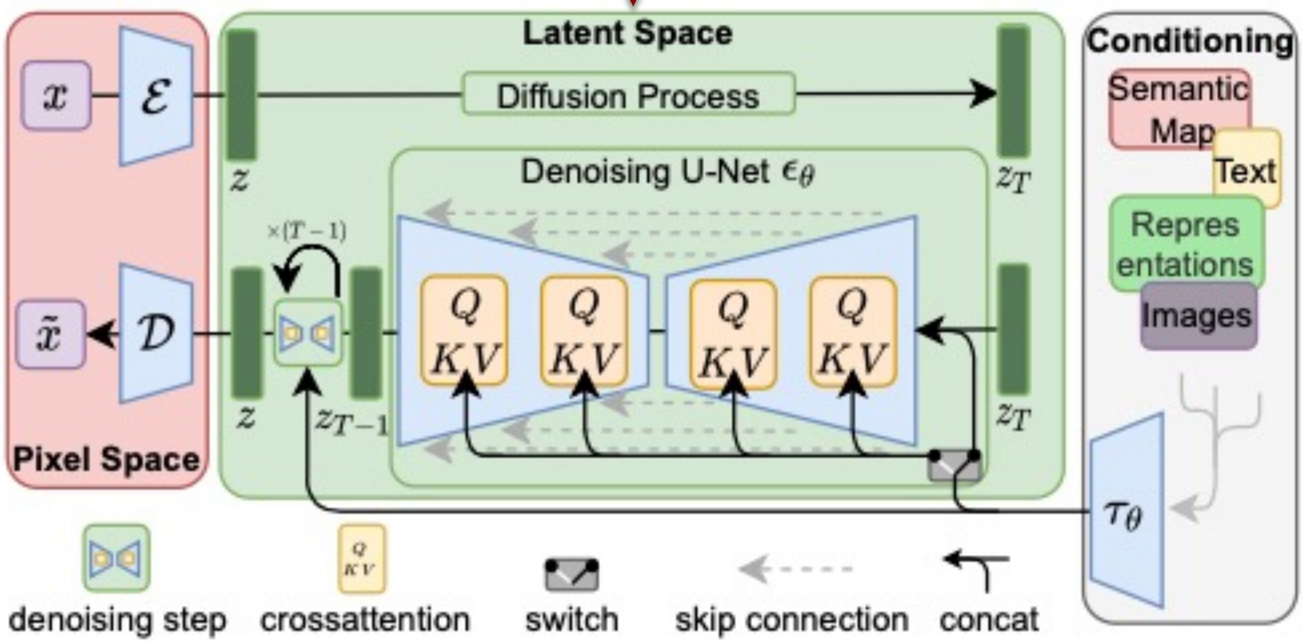
https://arxiv.org/abs/2112.10752

(2) The CLIP embedding of the text prompt is "woven" into the U-Net through an attention mechanism (the Q, K, V should look familiar 🙂)



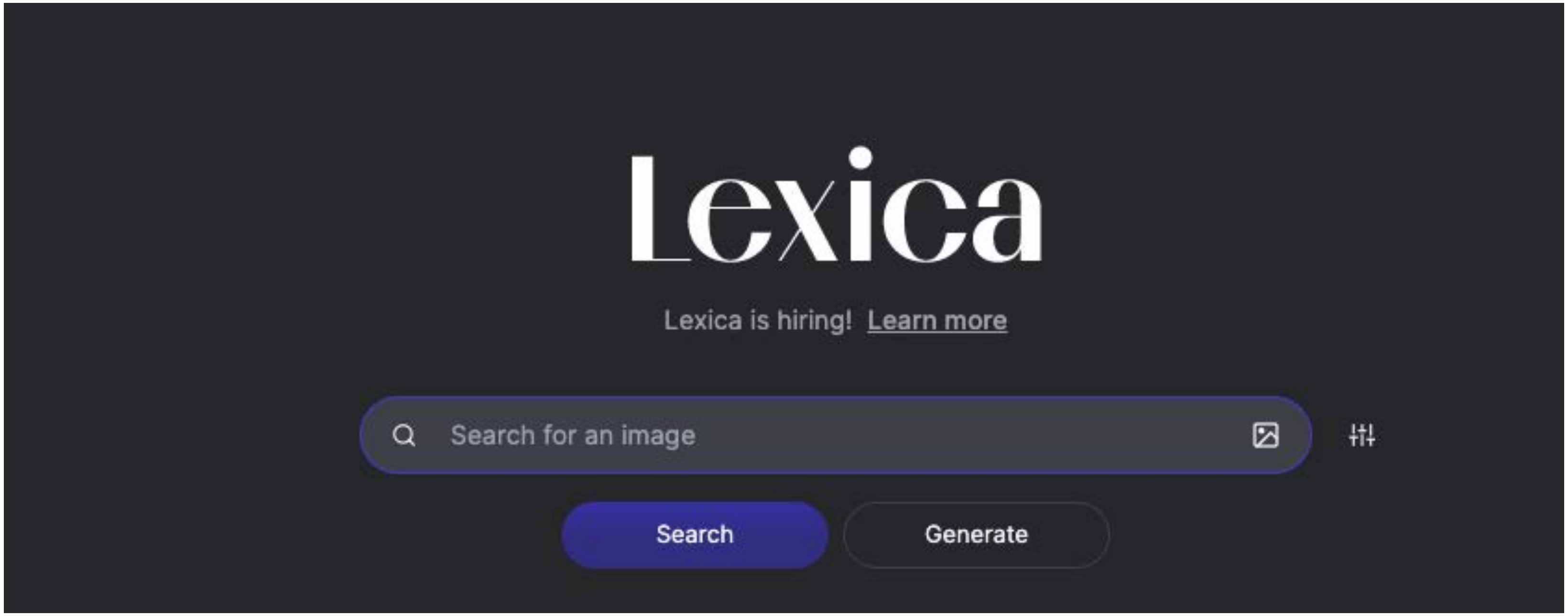
https://arxiv.org/abs/2112.10752

(3) Instead of working with images in "pixel space", the diffusion process operates in "latent space", and this speeds things up dramatically



https://arxiv.org/abs/2112.10752

These models are transforming the art world …



https://lexica.art/

… but their applicability is far-reaching.

Example: Protein design using diffusion models

A.I. Turns Its Artistry to Creating New Human Proteins

Inspired by digital art generators like DALL-E, biologists are building artificial intelligences that can fight cancer, flu and Covid.

A protein diffusion model doing unconditional generation, converting noise into plausible structures. Video by Namrata Anand

Much as DALL-E leverages the relationship between captions and photographs, similar systems can leverage the relationship between a description of what the protein can do and the shape it adopts.

Researchers can provide a rough outline for the protein they want, then a diffusion model can generate its three-dimensional shape.

https://www.nytimes.com/2023/01/09/science/artificial-intelligence-proteins.html

Colab

We will use the Diffusers library

Diffusers

🤗 Diffusers is the go-to library for state-of-the-art pretrained diffusion models for generating images, audio, and even 3D structures of molecules. Whether you're looking for a simple inference solution or want to train your own diffusion model, 🤗 Diffusers is a modular toolbox that supports both. Our library is designed with a focus on usability over performance, simple over easy, and customizability over abstractions.

https://huggingface.co/docs/diffusers/index

Lightning Tour of HuggingFace Computer Vision Models

Colab link