CS3485

Deep Learning for Computer Vision

Lec 22: Image Generation by Prompt

Announcements

- Save the date Nov 29th (Wed) during lecture:
  - Ben Swanson from Ubisoft will be our guest lecturer!
  - He'll come to talk about some of his work in computer vision for health.
  - Make sure to show up and ask him questions!
- Lectures after Mega Quiz:
  - Attendance mandatory throughout.
  - I'll be there to help out on projects for next week's lectures.
  - We'll have our final formal lecture on Tuesday Dec 3rd after thanksgiving
- TA applications are open!

Announcements

- There will be a bunch of candidates to our faculty position giving talks! This Thursday at 4:15-5:45 we'll have Brandon Fain from Duke University with a talk titled:

  Algorithmic Fairness, Moral AI, and Human Alignment

- We'll have 4 more coming after that one, many of them on AI, 4:15-5:45 in Mon, Wed and Fri! Please show up to as many as you can!

(Tentative) Lecture Roadmap

Basics of Deep Learning

- Intro to Computer Vision
- Linear Classifiers and Perceptron
- Multilayer Perceptron
- Optimization in Deep Learning
- Pytorch I – MLPs
- Pytorch II – Images and Regularization
- Convolutional Neural Networks
- Data Augmentation and Deep CNNs
- Transfer Learning and Residual Nets
- Inception Net and what CNNs learn
- Adversarial Examples and Self-supervision

Deep Learning and Computer Vision in Practice

- Intro to MLOps
- Intro to Object Detection
- Fast Object Detection
- Intro to Image Segmentation
- Applications of Detection and Segmentation
- Autoencoders
- Image Generation with GANs
- Advanced GANs
- The Attention Mechanism
- Transformers and ChatGPT
- Image Generation by Prompt
- Misc. Topics

Computer Vision and Language

- Up to now we saw many applications of Deep Learning to CV and NLP tasks, but it has been also applied to many other **data domains**.
- Some of these applications are in fact **multi-modal**, i.e., leverage more than one domain to learn a specific task.
- Today, our goal is to learn a currently very popular multi-modal computer vision task: **image generation by prompt** (also called text-to-image translation), which means generating an image from its textual description.
- Here, we will learn how to use a famous method called **Diffusion** to solve this problem.

Image generated by the prompt "*A photo of an astronaut riding a horse*".

First Ingredient: CLIP

- In this task, we first need to make textual and visual data interact so our text may guide the generation of our desired image.
- One recent popular approach to connect text and image domains is **CLIP** (**C**ontrastive **L**anguage-**I**mage **P**retraining, from a 2021 paper*), which is a self-supervised method that aims to find similar representations for corresponding data in different domains.
- But what does that mean? Assume we have a batch of N images paired with their respective descriptions, e.g.:

$$\{(\text{Image}_1, \text{Text}_1), (\text{Image}_2, \text{Text}_2), \ldots, (\text{Image}_N, \text{Text}_N)\}.$$

- CLIP aims to **jointly train an Image and a Text Encoder Networks** that produce vector outputs (all of dimension 512) $I_1, I_2, \ldots, I_N$ and text embeddings $T_1, T_2, \ldots, T_N$ such that $I_1 \cong T_1, I_2 \cong T_2, \ldots, I_N \cong T_N$ and I_i is as different as possible from T_j for any $i \neq j$.

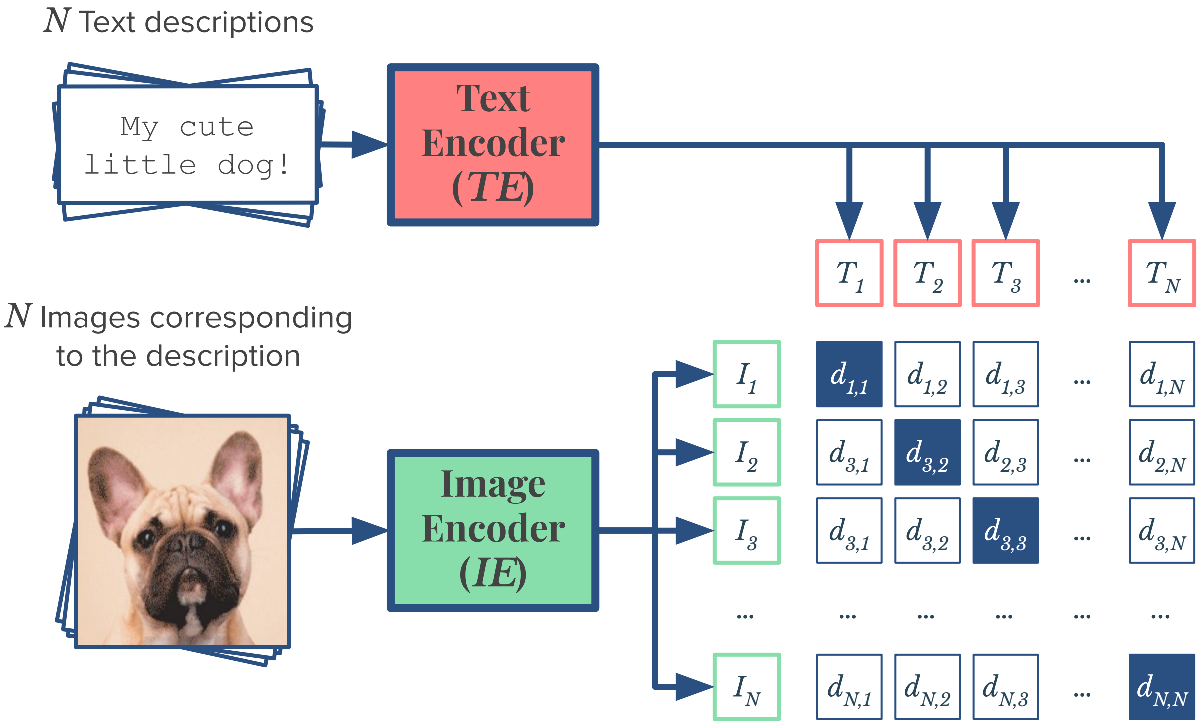
* OpenAI, the creator of CLIP, provided a pedagogical Colab notebook for its use. HuggingFace also makes its pretained CLIP networks easily available for developers.

First Ingredient: CLIP

- It means that we can consider the similarity $d_{i,j}$ between I_i and T_j and find all possible $d_{i,j}$ from the batch.
- For that, we can use the inner product similarity

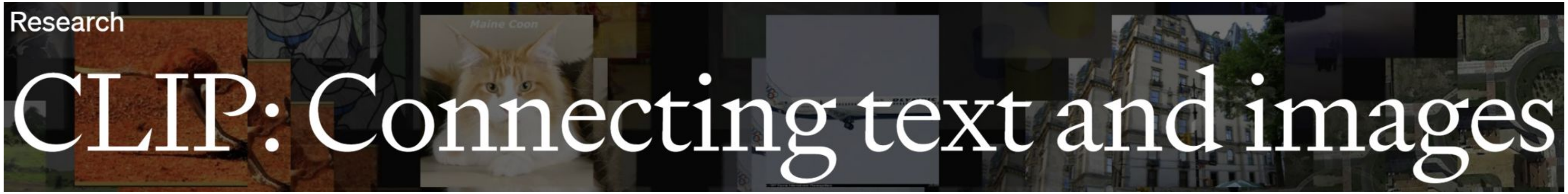
$$d_{i,j} = I_i^{\top} T_j,$$

- One then needs to maximize the diagonal values of an $N \times N$ matrix, while minimizing the other values in it.



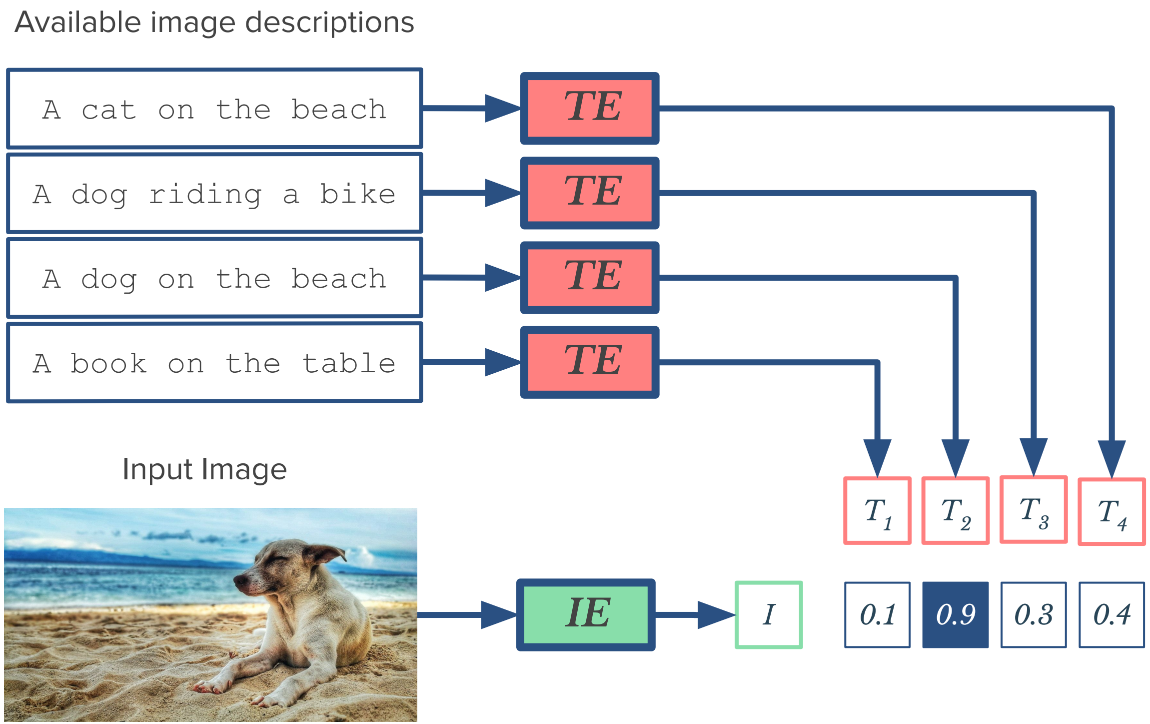
First Ingredient: CLIP

- But which architecture are used for the encoders? The original paper used the following:
  - The Text Encoder (TE) is a standard **Transformer** encoder.
  - The Image Encoder (IE) can be either a ResNet or a **Vision Transformer (ViT)**.
- Some other facts about the training of CLIP:
  - CLIP is trained using a staggering amount of 400 **million image-text pairs**. For comparison, the ImageNet dataset contains 1.2 million images.
  - The final trained CLIP model was trained on 256 V100 GPUs for two weeks. For an on-demand training on AWS, this would cost at least 200**k dollars**!
  - The model uses a batch of $N = 32{,}768$ images for training, meaning that they had to keep a matrix of the size $N \times N$ floats in its RAM memory, which amounts to around 17.5 **Gb**!



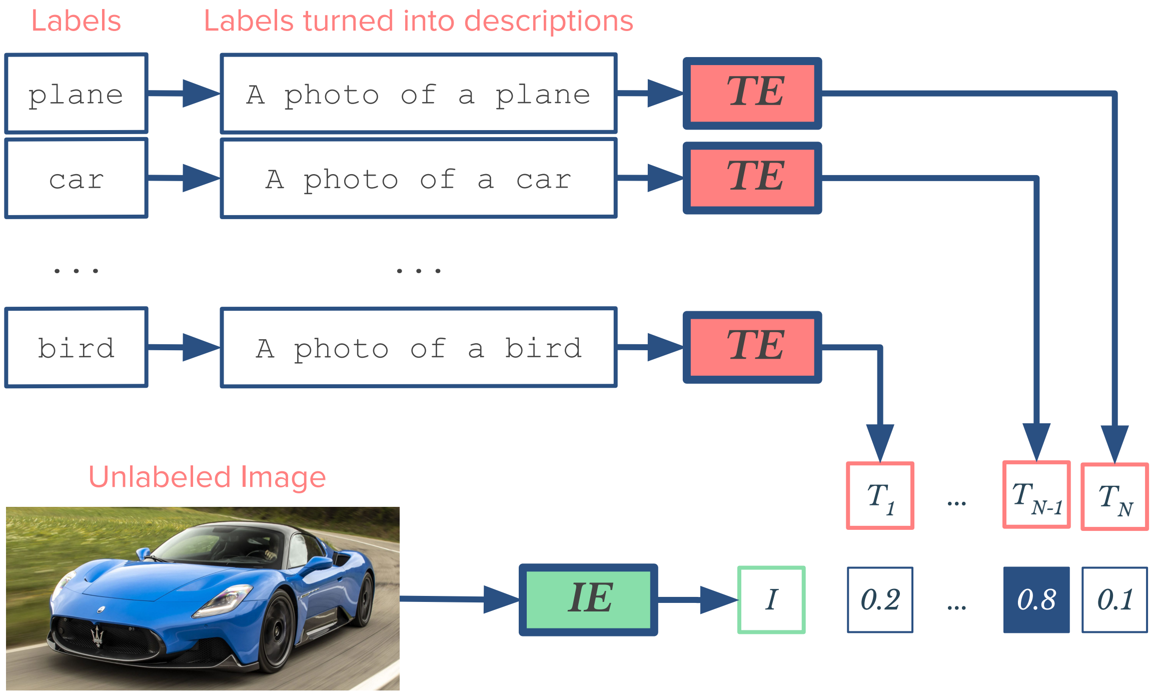
CLIP in Practice: Toy example

- What can we do with it?
- We can use the trained TE and IE to find a description for an image.
- We pass all the available descriptions through TE and our image through IE to find their respective vector representations.
- Then we select the text whose representation is the most similar to the image's.



CLIP in Practice: Zero-shot learning

- CLIP is able to perform **zero-shot learning**: the ability of a model to perform tasks **it was not explicitly trained to do**.
- For example, for **image classification***, one can convert a series of possible class labels, turn them into descriptions and select the that best describes an unlabeled image, according to CLIP.

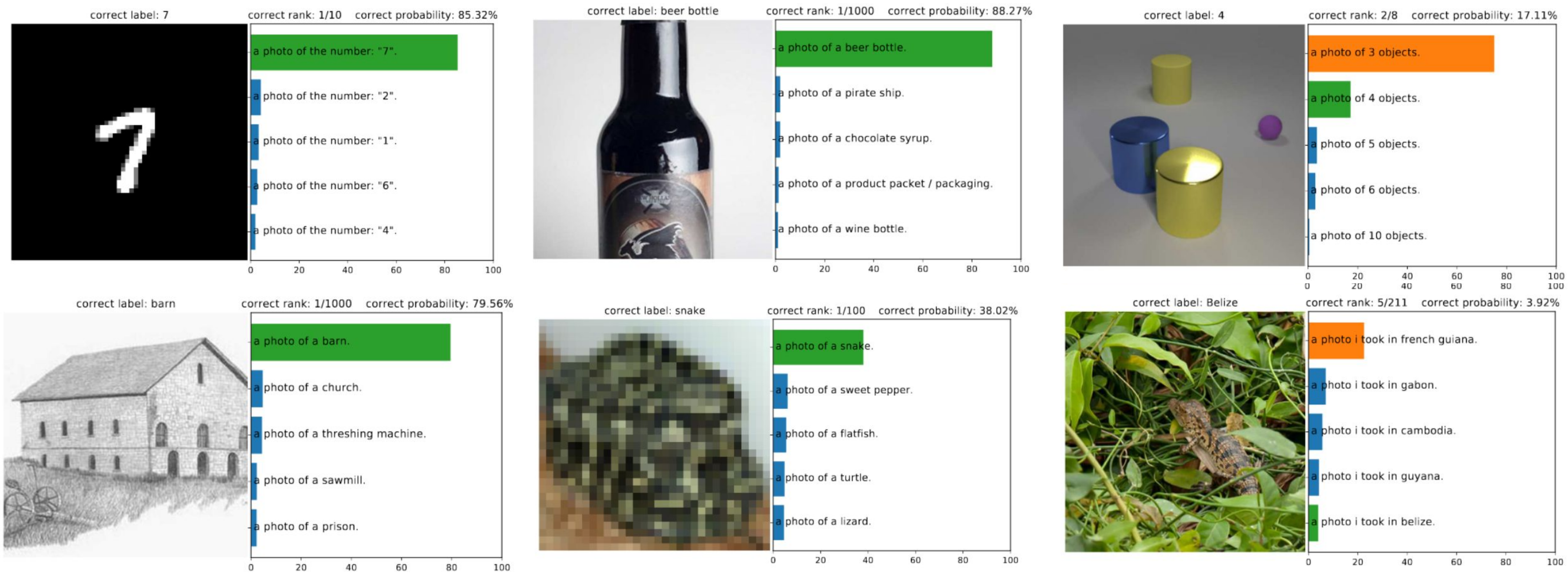


* Note that CLIP was not trained specifically for image classification.

Go over this colab notebook in class:
https://colab.research.google.com/github/openai/clip/blob/master/notebooks/Interacting_with_CLIP.ipynb

CLIP in Practice: Zero-shot learning

- Here are a few CLIP's Zero-shot learning results* (check paper for more):



* They show the top-5 prediction per image. The ground truth label is colored green while an incorrect prediction is colored orange.

CLIP in Practice: Text-Driven Image Manipulation

- CLIP had also been used in connection with StyleGAN for **text-based image manipulation**:

CLIP in Practice: Text-Driven Image Manipulation

- How does it work? The first step is to embed the input image I into the StyleGAN space (like in this 2019 paper* and in facemorph) to find a vector z_i. We hope that, if z_i is given to the generator G in StyleGAN, we get an image similar to I.
- Then, starting from z_i and having CLIP's trained encoders at our disposal, we'd find another z_o, with (where `text` is the input text):

Input Image

Image generated by StyleGAN with z_i.

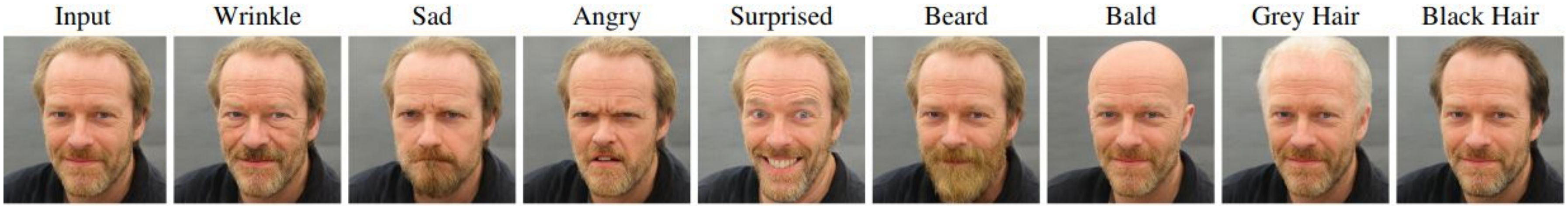
$$z_o = argmax_z \, [IE(G(z))]^T [TE(\texttt{text})]$$

- This means that we'd like StyleGAN to generate a latent vector whose corresponding image has an encoding that is very similar to the text description according to CLIP.

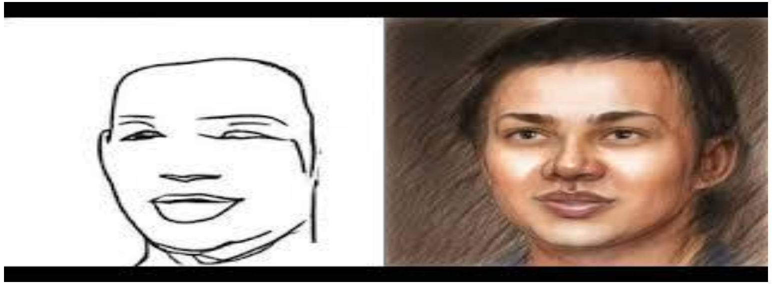
* I am simplifying here for better understanding. Check the original paper and/or ask me about the remaining details, if you'd like.

CLIP in Practice: Text-Driven Image Manipulation

- This is the basic approach explained in StyleCLIP (published in 2021). The authors also provide an implementation in Replicate.com (a site similar to HuggingFace).



- In StyleGAN-NADA (also published in 2021), the authors elaborate StyleCLIP's technique to fast image domain adaptation (like translating a sketch drawing to its final result, for example). They also provide an implementation you can play with it.



Results of StyleGAN-NADA for domain adaptation

CLIP in Practice: Text-Driven Image Manipulation

- Here a few more results of StyleGAN-NADA (you can try it yourself here):

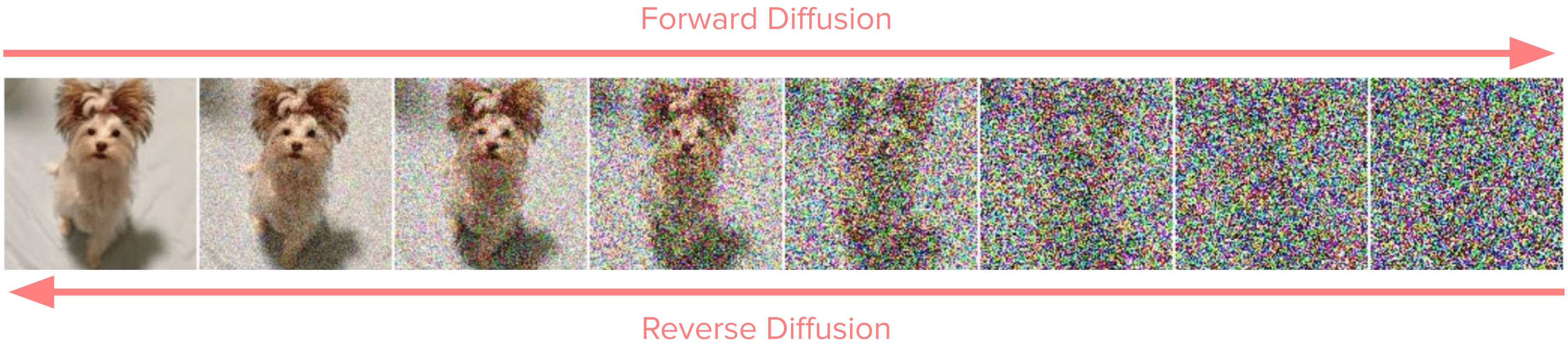
Input Image

Exercise (*in pairs*)

- Today is the last day, so let’s just have fun! Go play with StyleCLIP and StyleGAN-NADA implementations o Replicate. Try out the various available parameters in those models and try to understand what they are responsible for.

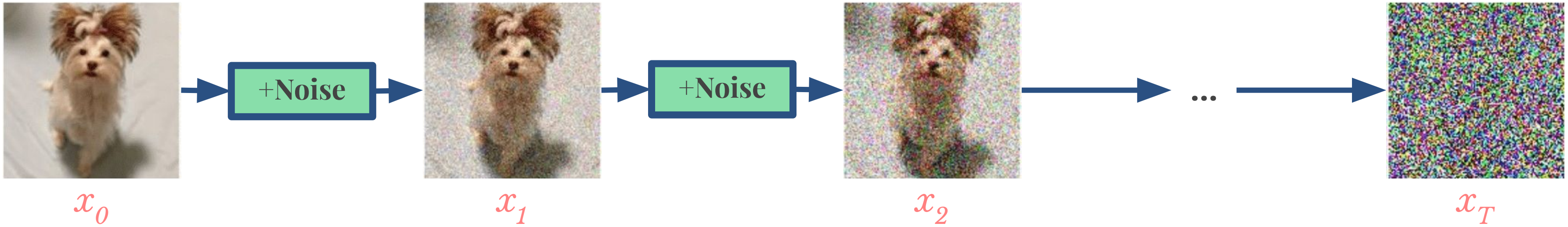
Second Ingredient: Diffusion Models

- Our attempt here to generate images from text is based on **Diffusion Models**, which consist of two processes:
  - A **forward diffusion** process adds noise to a training image, gradually in T steps, turning it into an uncharacteristic noise image.
  - A **reverse diffusion**, which attempts to, starting from a noisy image, recover a realistic image.
- We'll try to mimic this process so to learn how to generate images from noise.

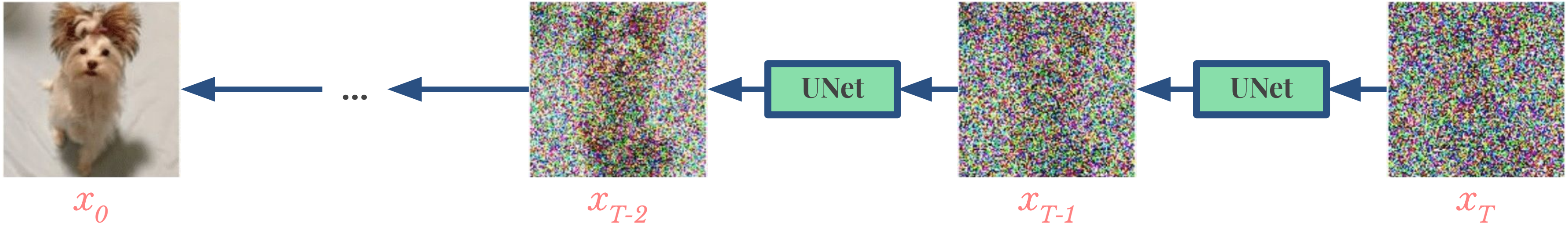


Second Ingredient: Diffusion Models

- Reproducing the forward process is simple: at each step, simply add Gaussian noise:



- The reverse process (that removes noise) is not as straightforward, but we can use **denoising networks** (such as UNets) on various (x_i, x_{i+1}) image pairs, to do that job:



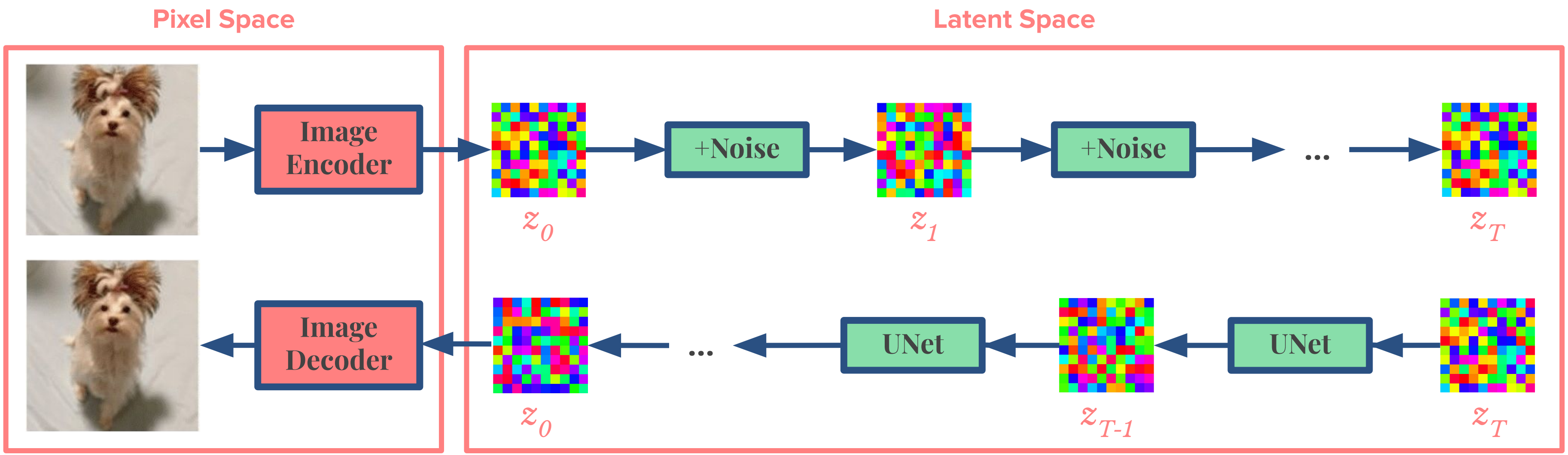
Second Ingredient: Diffusion Models

- The idea is inspired in thermodynamics (via a 2015 paper) and is the basis for what became known as **Diffusion Model (DM)**, published in 2020*.
- DM for deep learning is a very beautiful theory with compelling results, overcoming some of the limitations of GAN-based image generation (it is **not prone to mode collapse** for example).
- The main drawback of DM is its complexity and learning speed. Originally, it used $T = 1000$, which means that it trained 1000 different UNets!
- **Latent Diffusion Models (LDM)**, published in 2021, overcame this issue by training these UNets on smaller sized latent image representations.

Images (256×256) of faces generated by the original diffusion process algorithm.

Second Ingredient: Diffusion Models

- The idea is to train an Autoencoder* and used its (much smaller) latent space for diffusion:

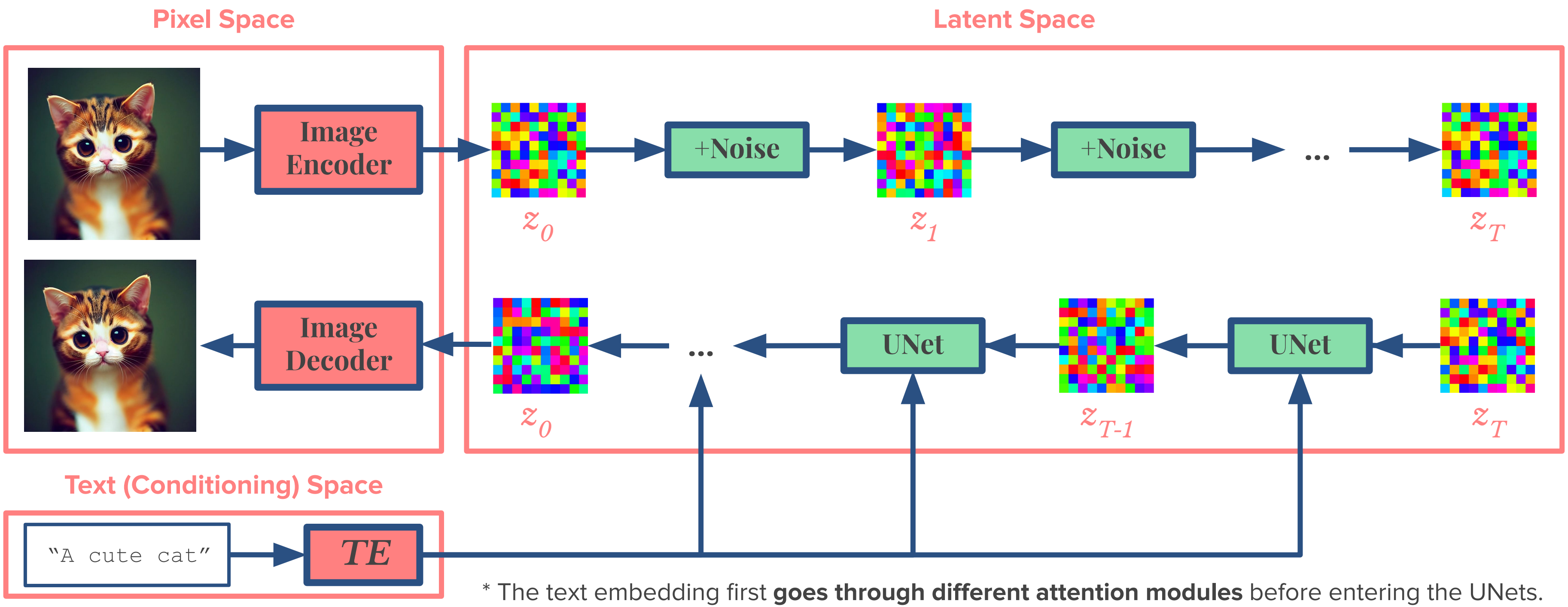


- Besides the speed-up, DLM also introduced added a feature that allowed conditional information (such as, but not limited to, text) to the generation pipeline.

* The authors used an image compression technique that is more elaborated than our simple Autoencoder, but the idea is similar.

Second Ingredient: Diffusion Models

- They input CLIP's text embedding into each UNet along with their respective latent vector*.

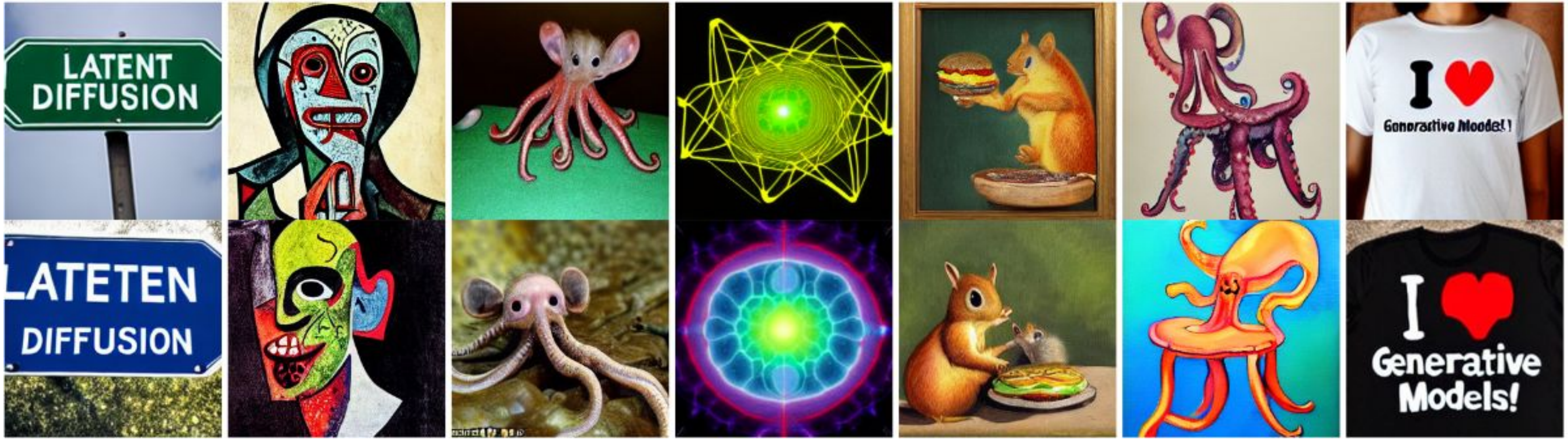


* The text embedding first **goes through different attention modules** before entering the UNets.

Second Ingredient: Diffusion Models

- With this simpler approach, they are able to "quickly" train a 1.45 billion parameter model and generate the following 256×256 images with the prompts:

"A street sign that reads 'Latent Diffusion'" | *"A zombie in the style of Picasso"* | *"An image of an animal half mouse half octopus"* | *"An illustration of a slightly conscious neural network"* | *"A painting of a squirrel eating a burger"* | *"A watercolor painting of a chair that looks like an octopus"* | *"A shirt with the inscription: 'I love generative models!'"*



Second Ingredient: Diffusion Models

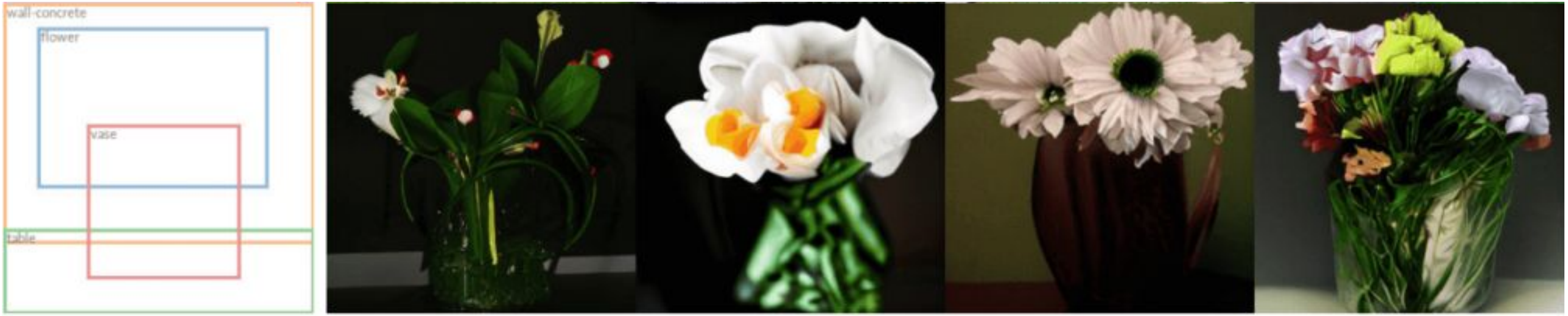
- The number of steps in this diffusion process is crucial to generate realistic images.

- With DLM, we can also condition the generation with data other than textual by replacing the Text Encoder. We can condition it on segmentation maps, for example:

Second Ingredient: Diffusion Models

- The authors showed that DLM can be used in other imaging tasks, such as in painting:

- Or image generation from bounding boxes:



Stable Diffusion

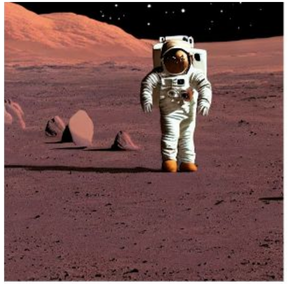
- DLM eventually became known as **Stable Diffusion** and as the basis for Stability AI, the company that is commercializing this algorithm.
- Stable Diffusion became very popular at creating beautiful art! Lexica* is a website where you can search over its creations and prompts!

* You can play with a pretrained generative code of lexica in HuggingFace here.

Other popular Text-to-Image approaches

- Besides Stable Diffusion, two other approaches tackle the text-to-image task:
  - **DALL·E** (under DALL·E 2): announced by OpenAI in April 2022 in a blog post, uses a diffusion model conditioned on CLIP image embedding, but much further details were not disclosed.
  - **Midjourney** (under versions v1 to v5): created by an independent lab of the same name, the underlying technology is speculated to be based on Stable Diffusion, but it wasn't made public. Creators can use the via a Discord channel.

Stable Diffusion



DALL·E 2

Midjourney

"Alone astronaut on Mars, mysterious, colorful, hyper realistic"

Stable Diffusion

DALL·E 2

Midjourney

"Dark alley at night 4k raining aesthetic"

Stable Diffusion in Practice

- HuggingFace created the Diffusers library, where you have access to pretrained diffusion models.
- It's pretty easy to load and run them! First load what they call a diffusion pipeline from a pretrained diffuser:

```
# First install these libraries via !pip install diffusers transformers
from diffusers import DiffusionPipeline

model_id = "runwayml/stable-diffusion-v1-5"
pipeline = DiffusionPipeline.from_pretrained(model_id)
```

- Then, come up with a prompt and send it through the pipeline as following:

```
prompt = "An astronaut riding a horse"
image = pipeline(prompt).images[0]
```

Stable Diffusion in Practice

Click here to open code in Colab

- And after a few seconds (although it may take some minutes depending on your machine), here is your result!
- In that diffuser pipeline you can set:
  - How many steps you want in your inference (the lower, the quicker),
  - How much closely the inferred image should follow the prompt,
  - The model version and quality of your output.
- HuggingFace also has many good tutorials and codes for you to get started with Diffusion!

Video: Art in the AI Era

Video: A humane AI?