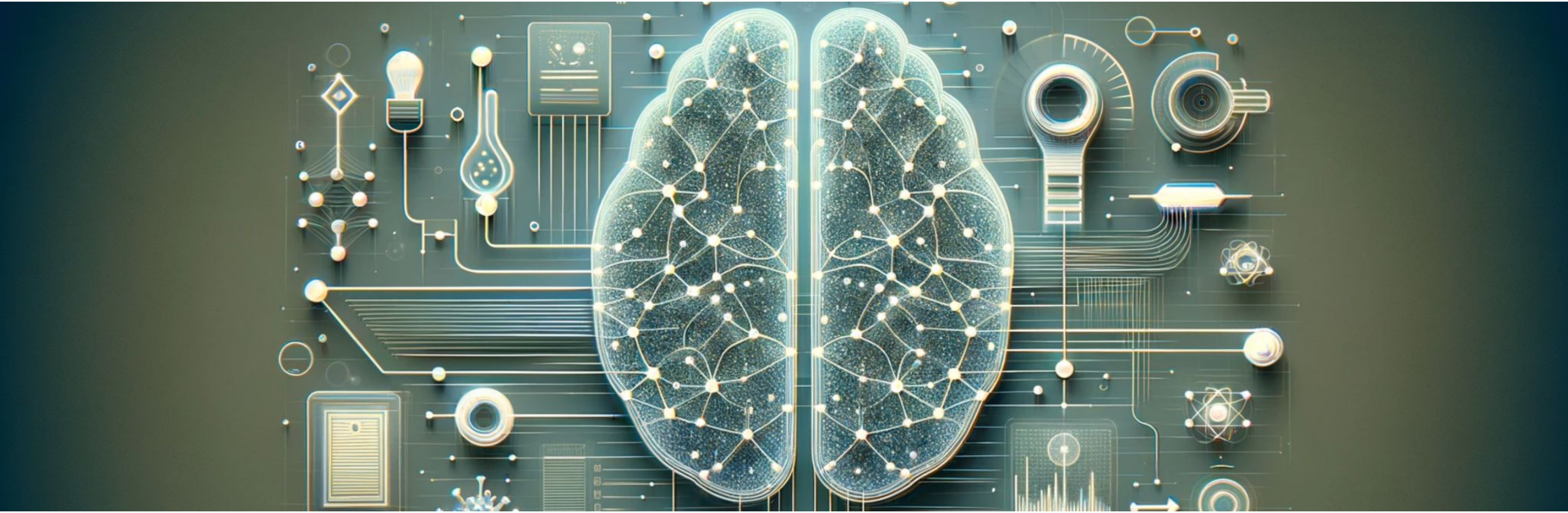
CSE7850/CX4803 Machine Learning in Computational Biology



Lecture 22: ML for Bioimaging Data

Yunan Luo

Dall-E: text-to-image generation (available in ChatGPT)

LU **You**

A photo of an astronaut riding a horse

ChatGPT

How to build such a generative model?

Two steps

Step 1: connecting text and images

an armchair in the shape of an avocado

an illustration of a baby daikon radish in a tutu walking a dog

a professional high quality emoji of a Lovestruck cup of boba

Two steps

Step 1: connecting text and images

an armchair in the shape of an avocado

an illustration of a baby daikon radish in a tutu walking a dog

a professional high quality emoji of a Lovestruck cup of boba

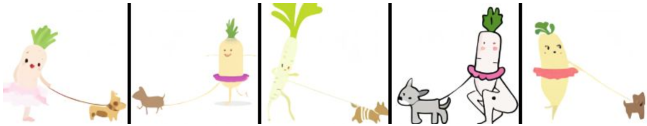
Two steps

Step 1: connecting text and images

Step 2: generate images

an armchair in the shape of an avocado

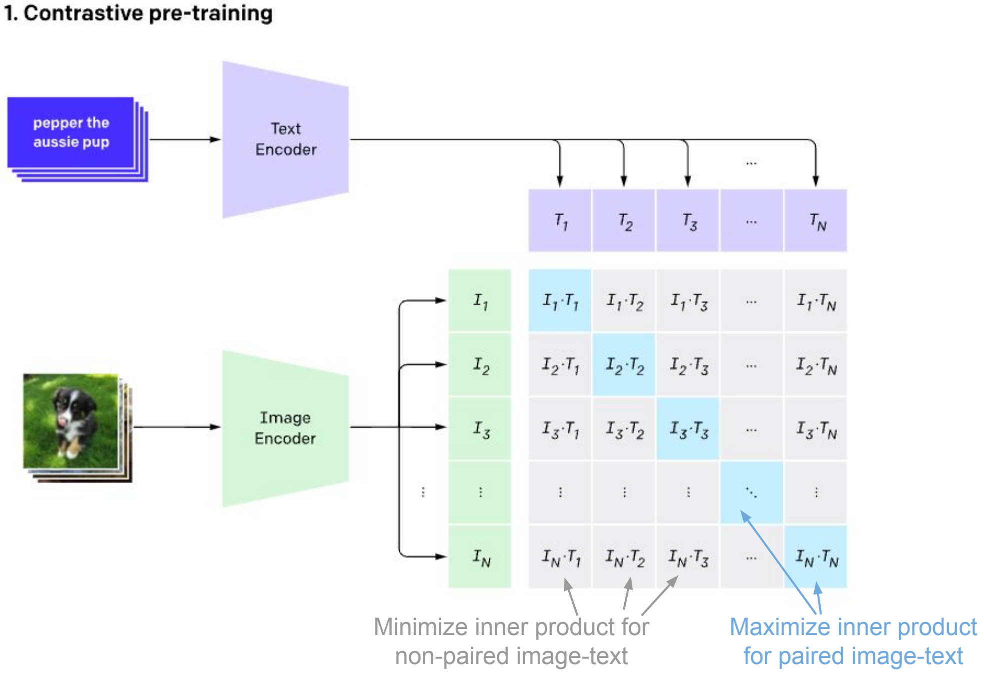
an illustration of a baby daikon radish in a tutu walking a dog



a professional high quality emoji of a Lovestruck cup of boba

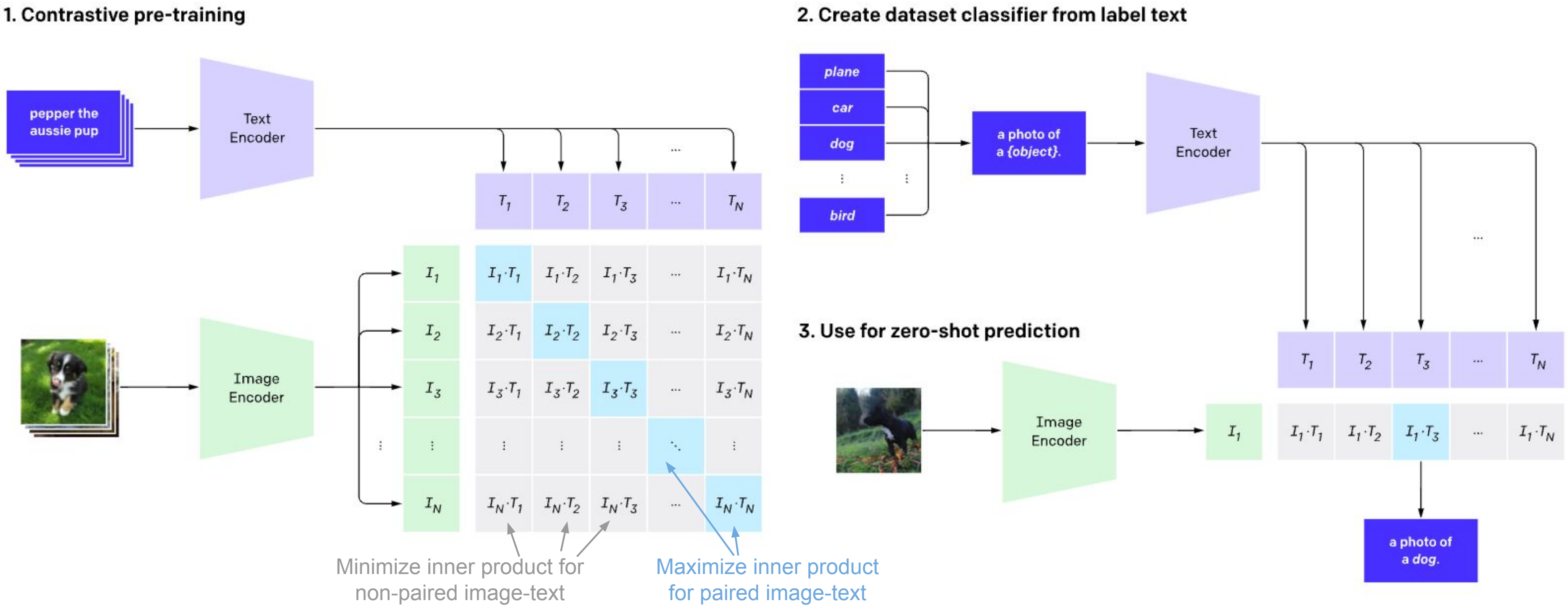
Step 1: Connecting text and images

CLIP (Contrastive Language–Image Pre-training)

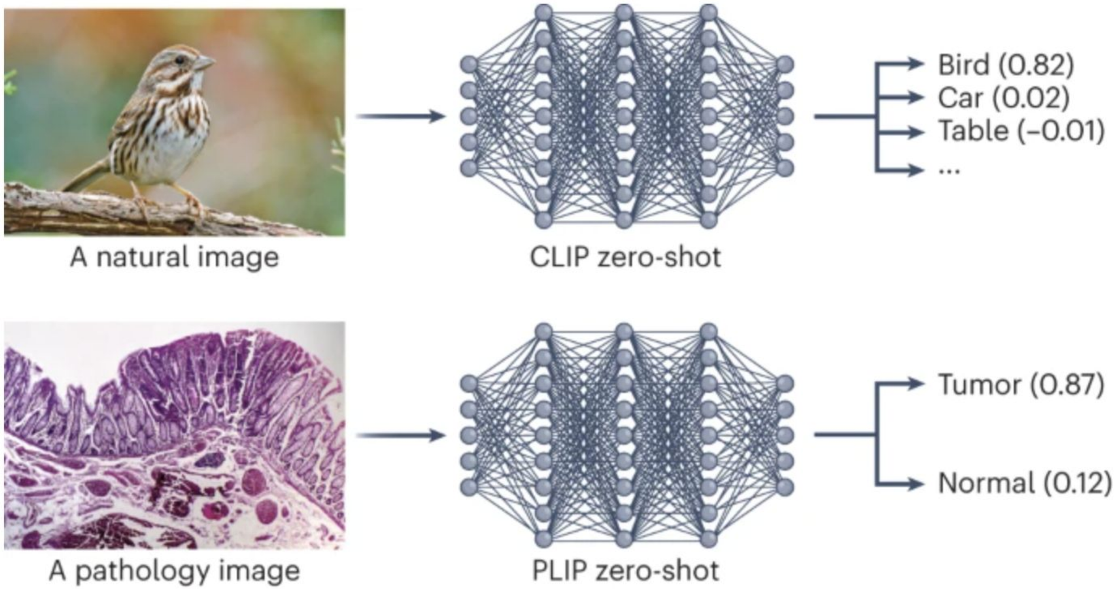


Step 1: Connecting text and images

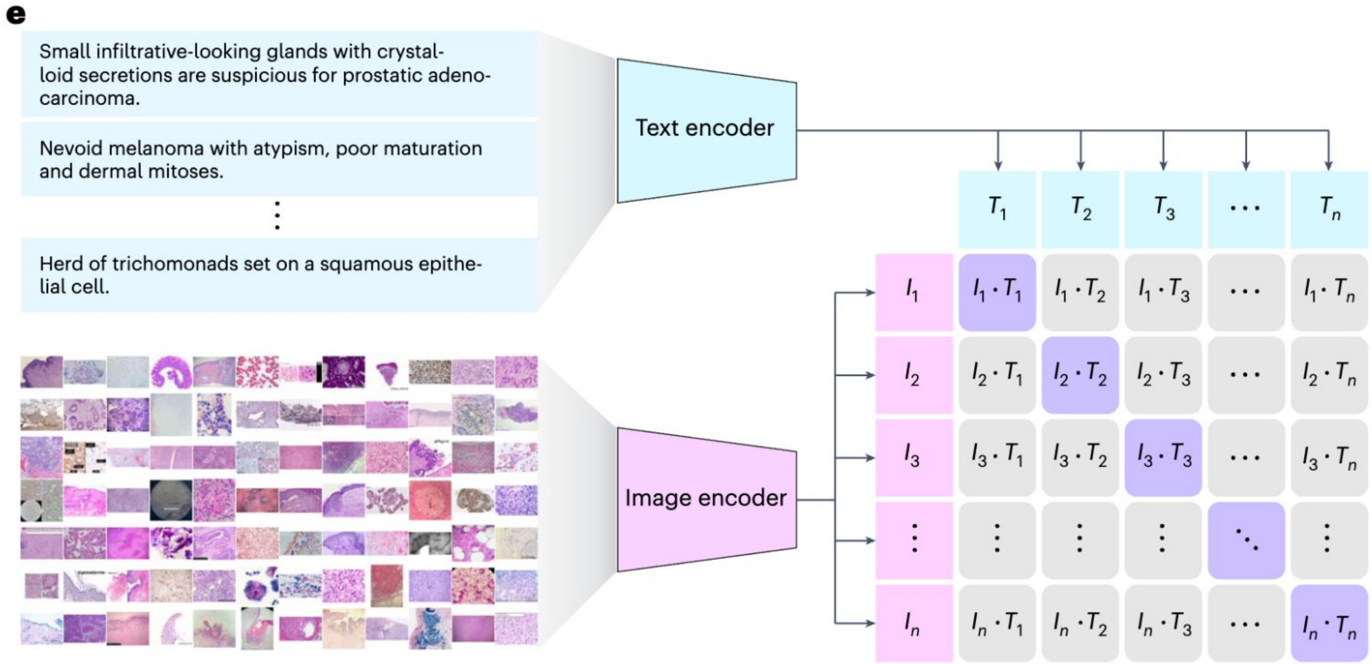
CLIP (Contrastive Language–Image Pre-training)



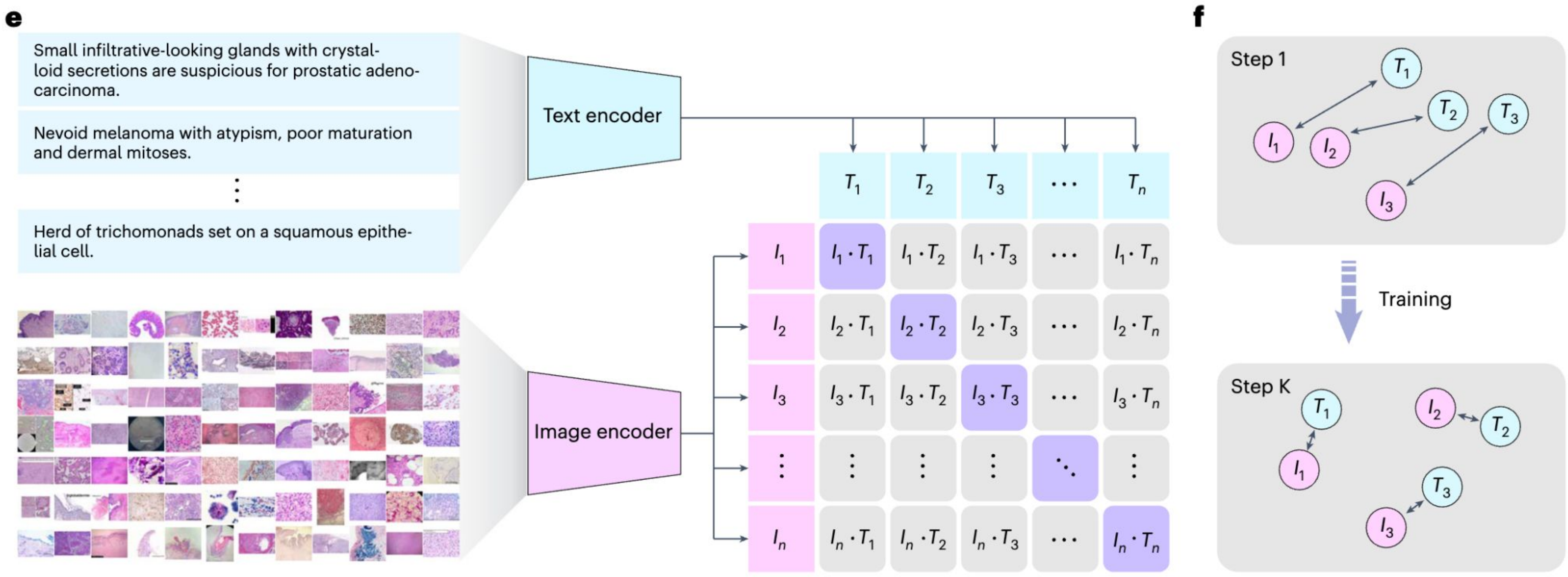
Today’s paper



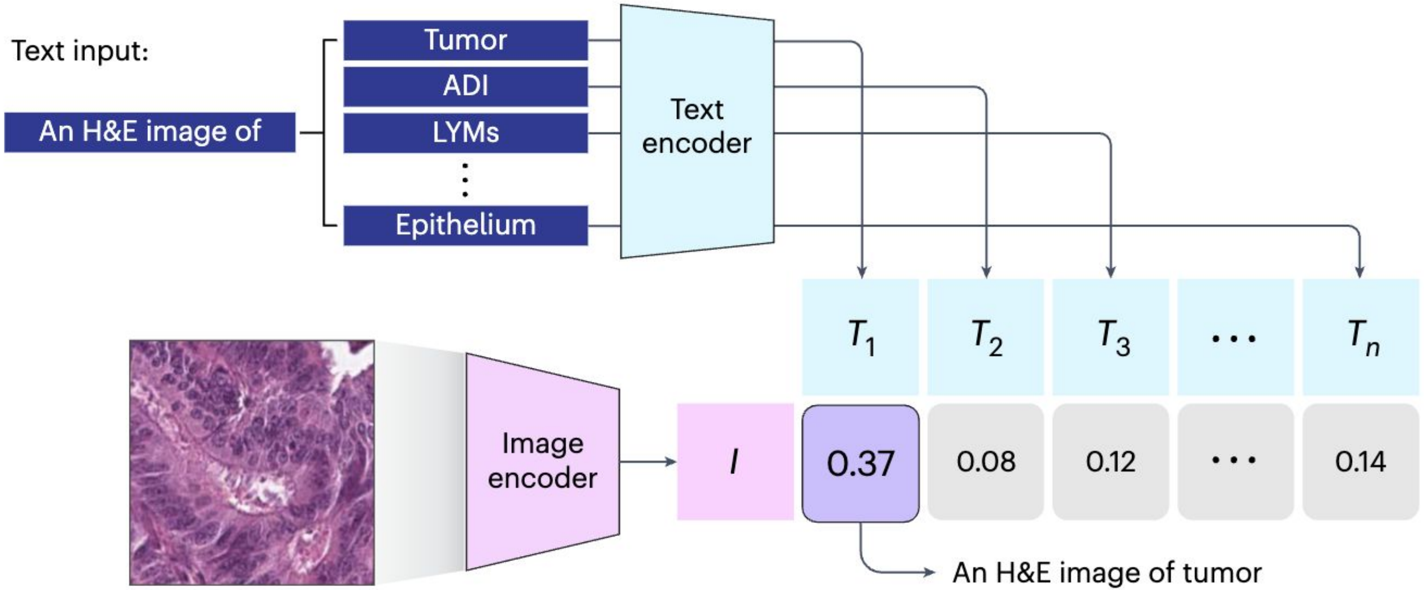
Today's paper



Today’s paper



Today's paper



Two steps

Step 1: connecting text and images

Step 2: generate images

an armchair in the shape of an avocado

an illustration of a baby daikon radish in a tutu walking a dog

a professional high quality emoji of a Lovestruck cup of boba

Step 2: generate images

an armchair in the shape of an avocado

DALL-E (2021)

DALL-E (2023)

Diffusion Models

Ho et al. Denoising diffusion probabilistic models (DDPM), Neurips 2020.
Song et al. Score-based generative modeling through stochastic differential equations, ICLR 2021.
Bao et al. Analytic-DPM: an Analytic Estimate of the Optimal Reverse Variance in Diffusion Probabilistic Models, ICLR 2022.
Bao et al. Estimating the Optimal Covariance with Imperfect Mean in Diffusion Probabilistic Models, ICML 2022.
Rombach et al. High-resolution image synthesis with latent diffusion models. CVPR, 2022.

Text-to-image generation

Input

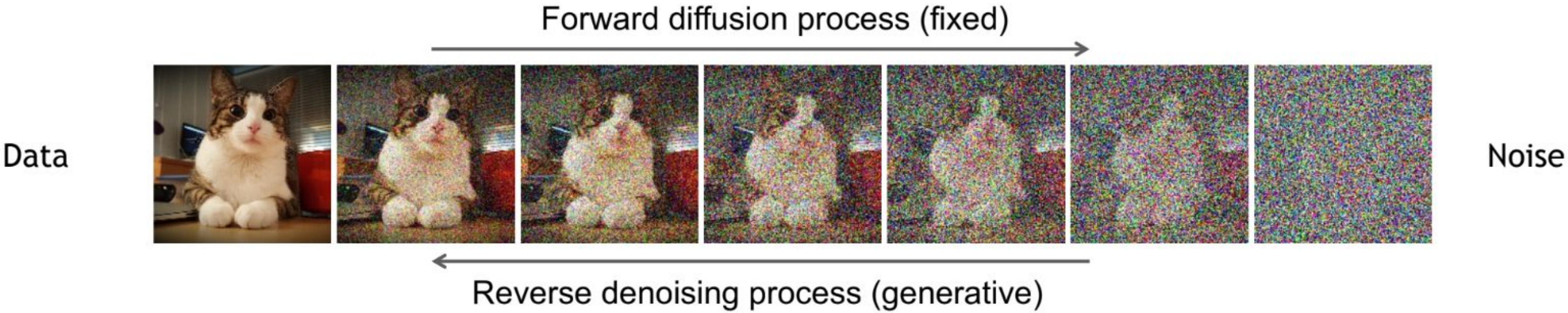
An astronaut riding a horse in photorealistic style.

Output

Diffusion models

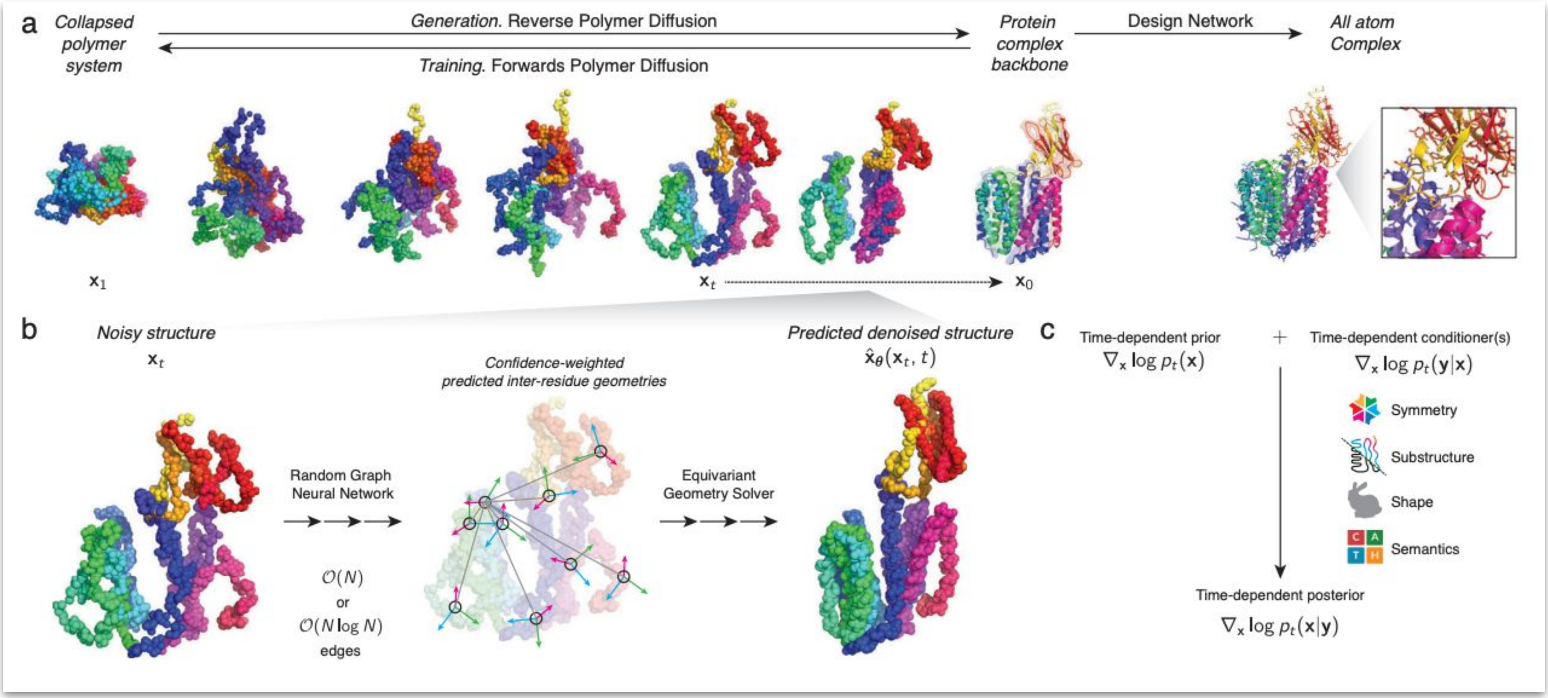
Denoising diffusion models consist of two processes:

- Forward diffusion process that gradually adds noise to input
- Reverse denoising process that learns to generate data by denoising



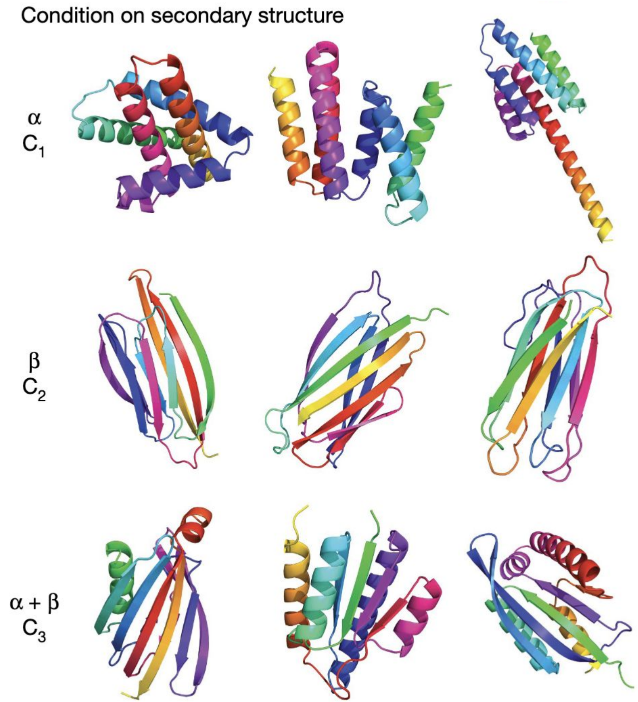
Slides credits: https://cvpr2022-tutorial-diffusion-models.github.io/

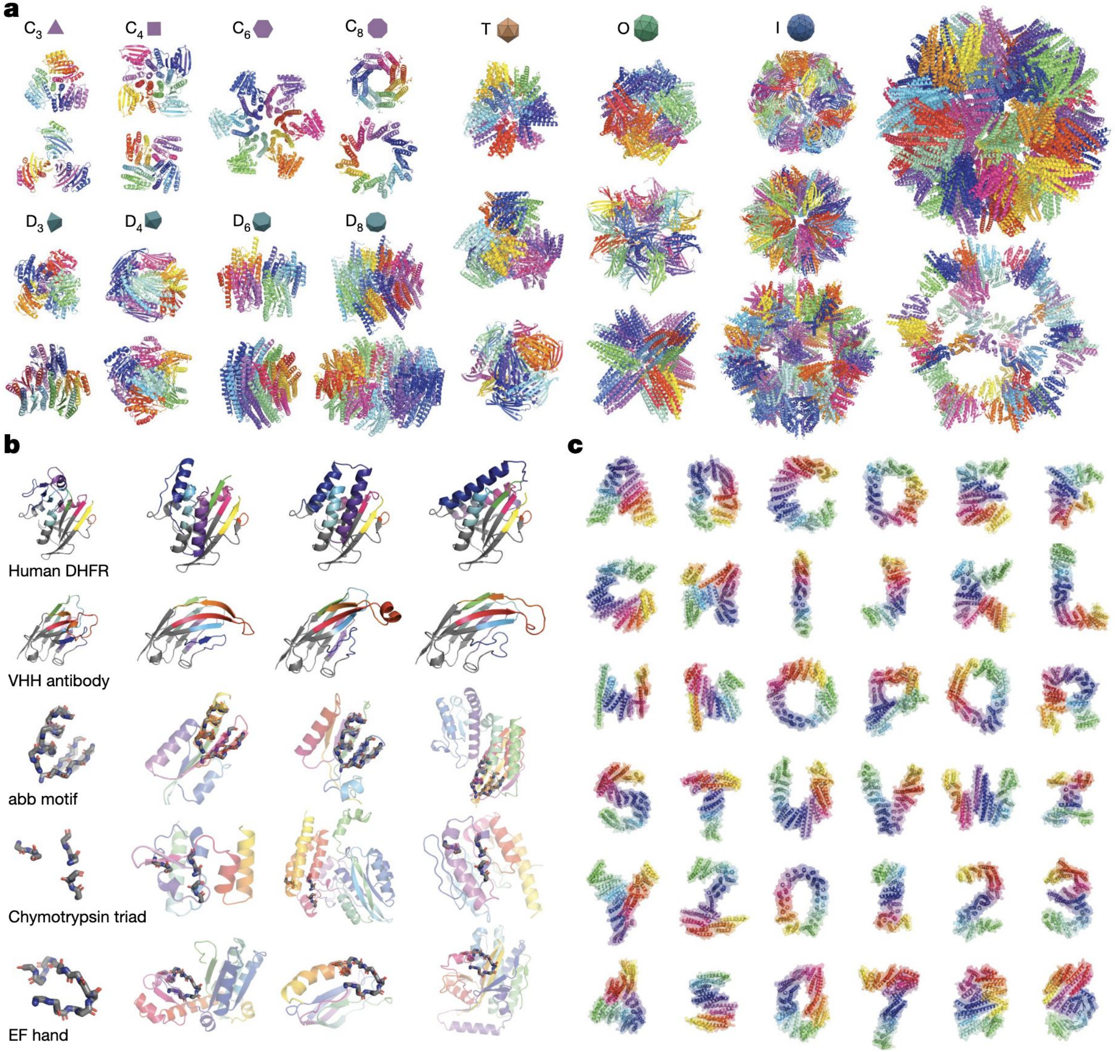
Application: Protein Design



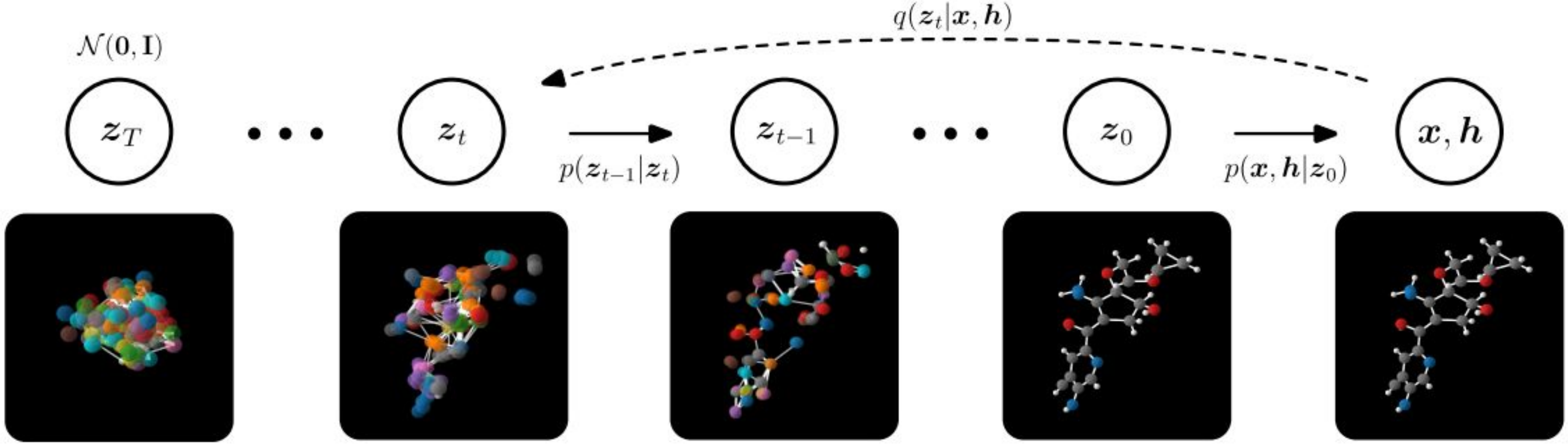
Ingraham et al., “Illuminating protein space with a programmable generative model”, bioRxiv, 2022

Paper #2 last lecture





Application: Drug Design



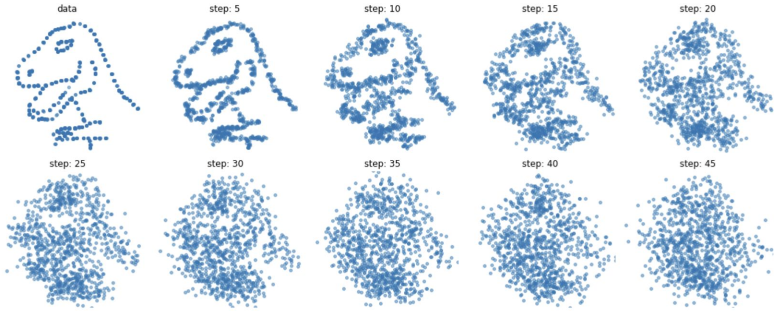
Hoogeboom et al., “Equivariant Diffusion for Molecule Generation in 3D”, ICML, 2022

Simple code demo

https://github.com/tanelp/tiny-diffusion

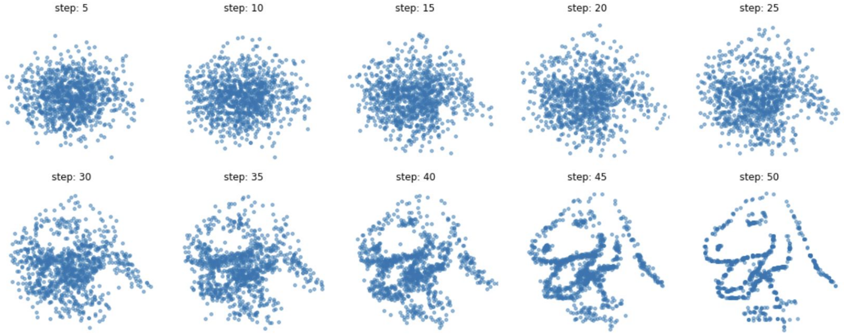
Forward process

A visualization of the forward diffusion process being applied to a dataset of one thousand 2D points. Note that the dinosaur is not a single training example, it represents each 2D point in the dataset.



Reverse process

This illustration shows how the reverse process recovers the distribution of the training data.



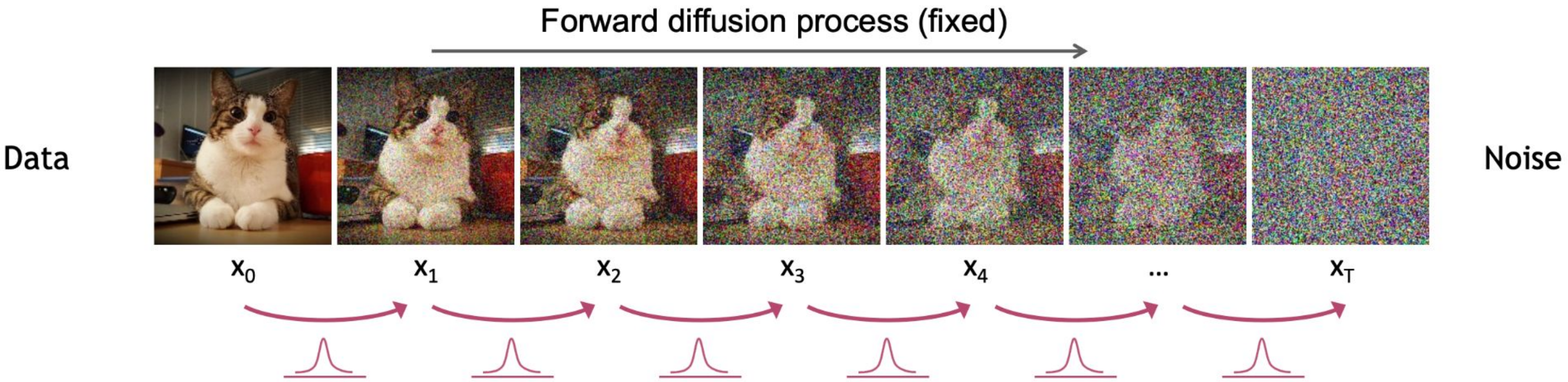
Forward Diffusion Process

```python
def add_noise(self, x_start, x_noise, timesteps):
    s1 = self.sqrt_alphas_cumprod[timesteps]
    s2 = self.sqrt_one_minus_alphas_cumprod[timesteps]

    s1 = s1.reshape(-1, 1)
    s2 = s2.reshape(-1, 1)

    return s1 * x_start + s2 * x_noise
```

The formal definition of the forward process in T steps:



$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x_t}; \sqrt{1-\beta_t}\mathbf{x_{t-1}}, \beta_t\mathbf{I}) \quad \Rightarrow \quad q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1})$$

(joint)

Reverse Denoising Process

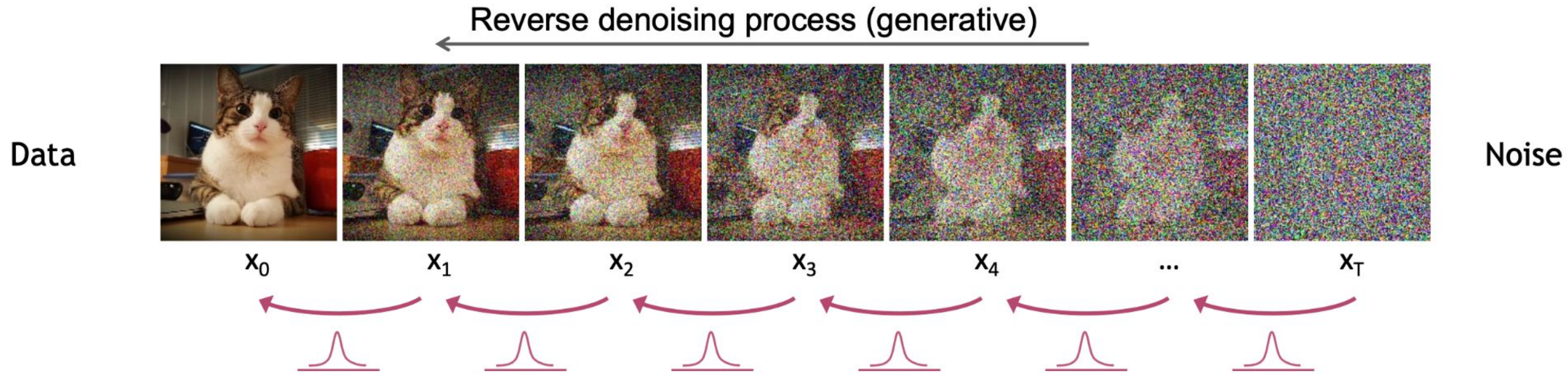
```python
def step(self, model_output, timestep, sample):
    t = timestep
    pred_original_sample = self.reconstruct_x0(sample, t, model_output)
    pred_prev_sample = self.q_posterior(pred_original_sample, sample, t)

    variance = 0
    if t > 0:
        noise = torch.randn_like(model_output)
        variance = (self.get_variance(t) ** 0.5) * noise

    pred_prev_sample = pred_prev_sample + variance

    return pred_prev_sample
```

Formal definition of forward and reverse processes in T steps:



$$p(\mathbf{x}_T) = \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I})$$
$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \underbrace{\mu_\theta(\mathbf{x}_t, t)}_{\text{Trainable network (U-net, Denoising Autoencoder)}}, \sigma_t^2\mathbf{I})$$

$$\Rightarrow \quad p_\theta(\mathbf{x}_{0:T}) = p(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$$