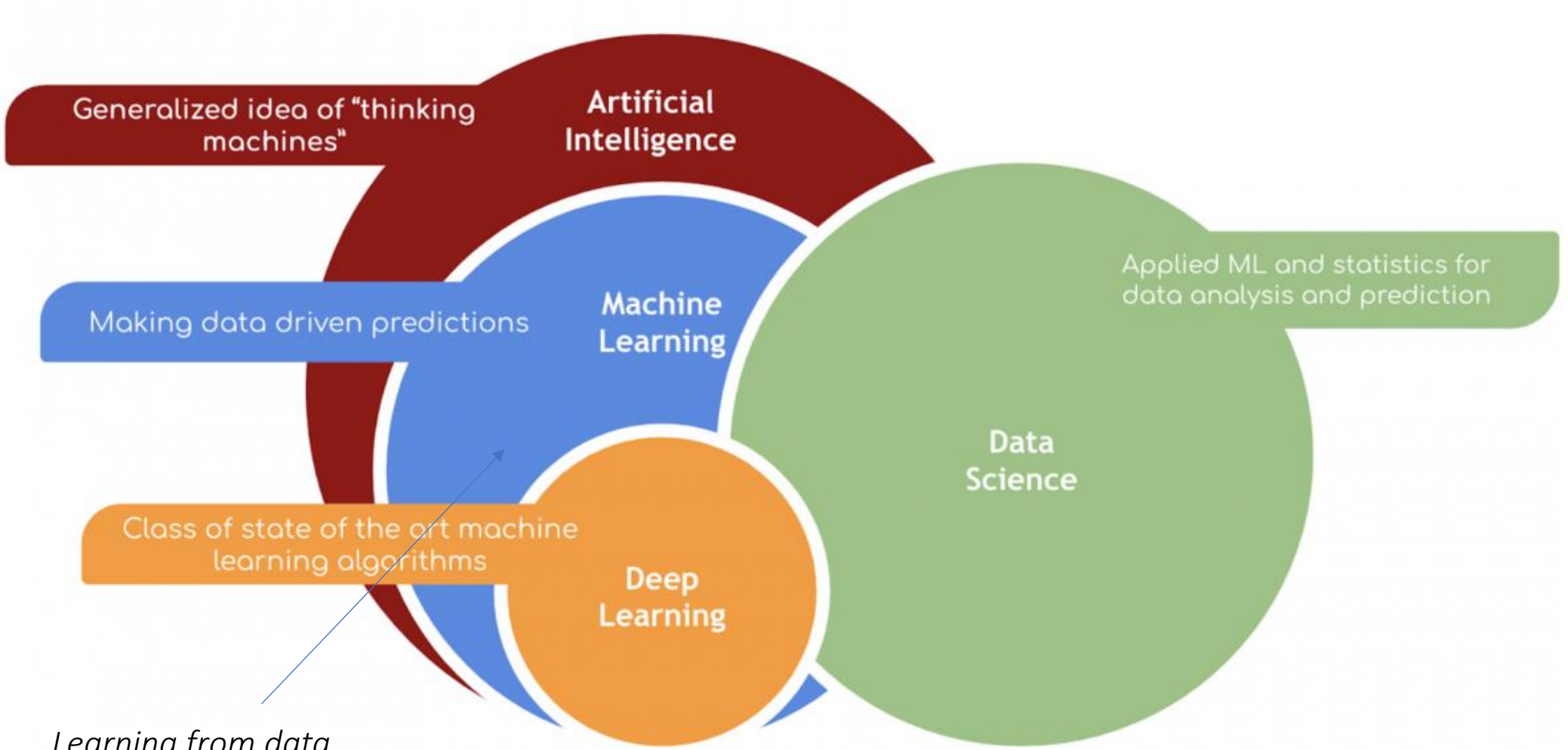
Machine Learning for Bioinformatics & Systems Biology

5. Deep learning strategies & Generative Modelling

Marcel Reinders *Delft University of Technology*
Perry Moerland *Amsterdam UMC, University of Amsterdam*
Lodewyk Wessels *Netherlands Cancer Institute*

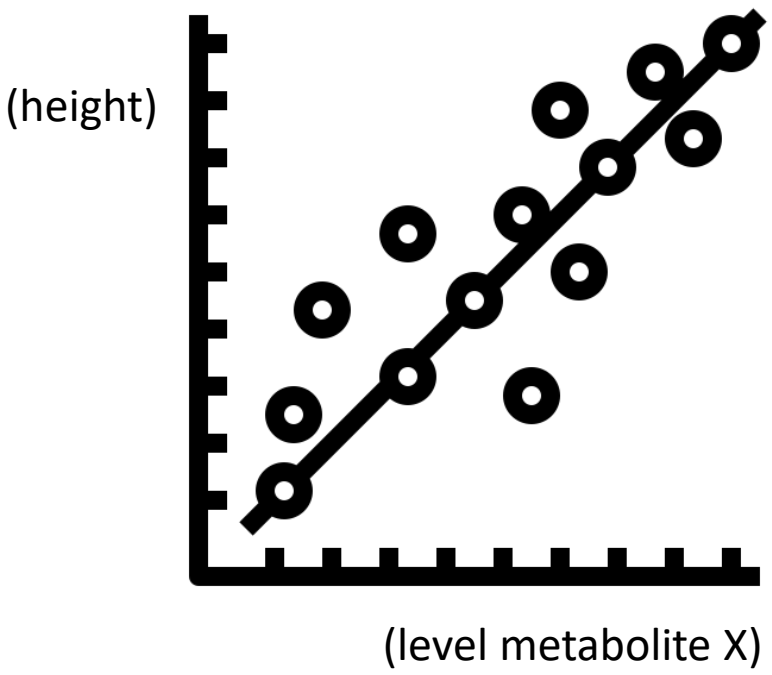
Deep learning strategies

What is Artificial Intelligence?



Learning from data

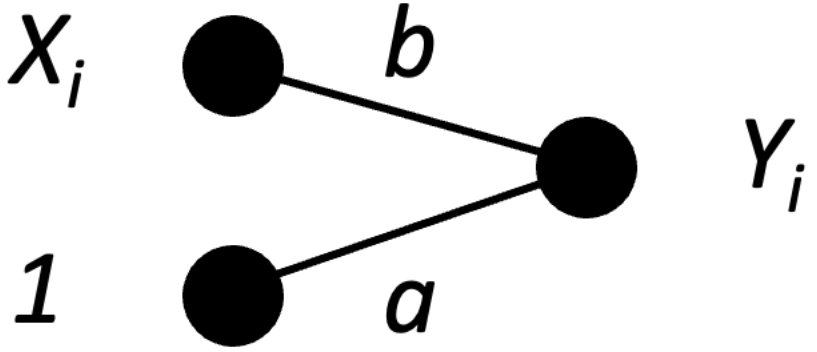
Machine learning : learning from data



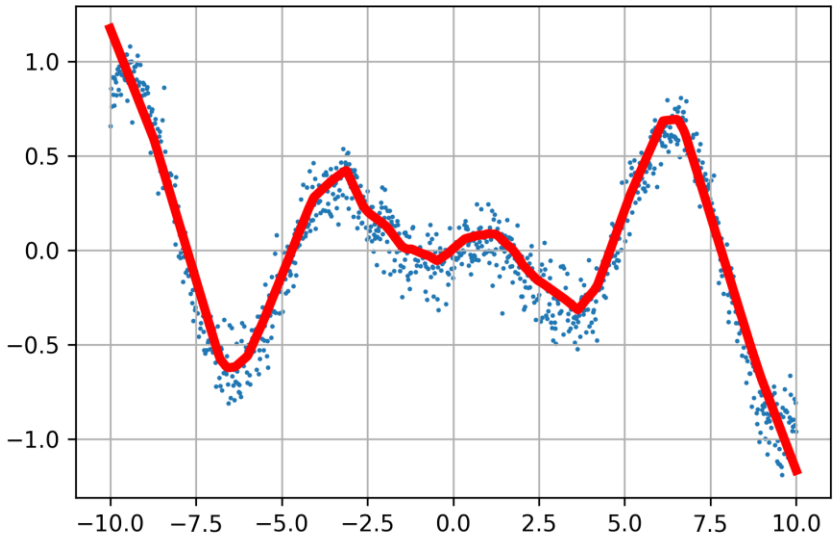
Statistical modeling

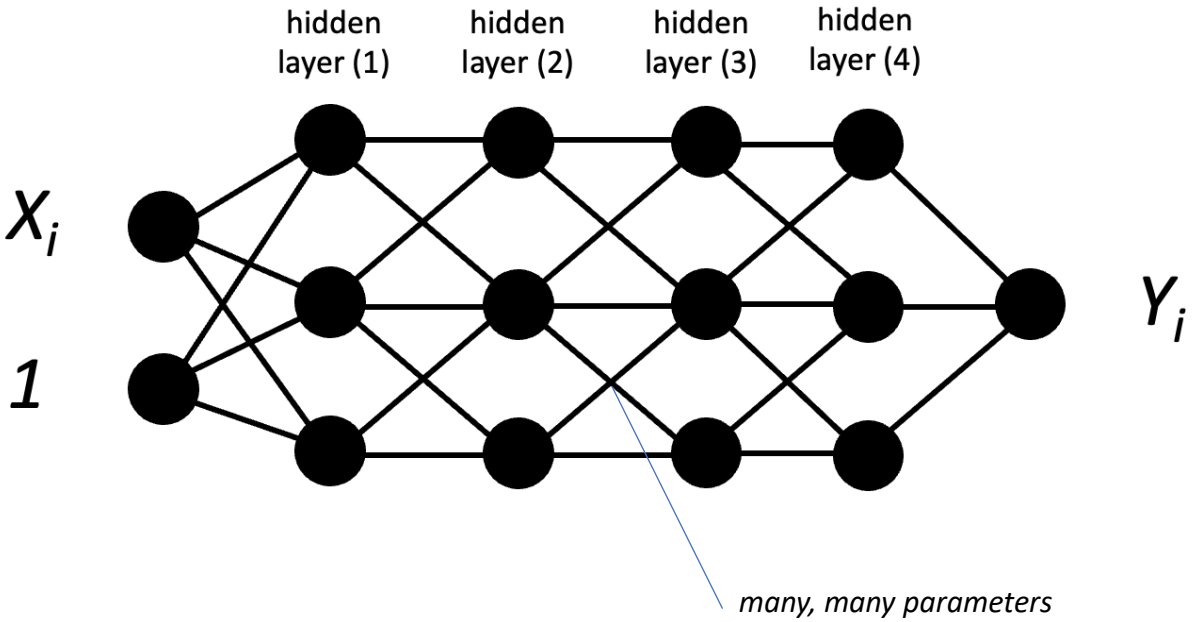
$$Y_i = a + bX_i + e_i$$

Neural network modeling



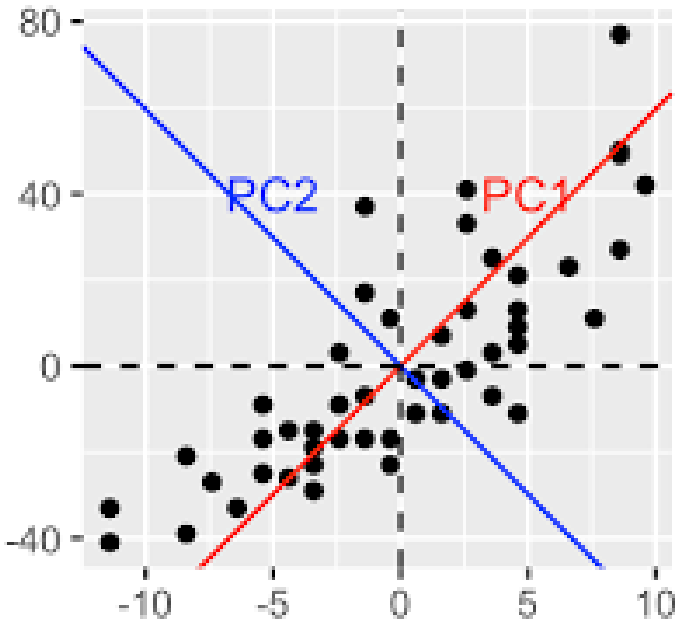
Complex functions, non-linear complex networks



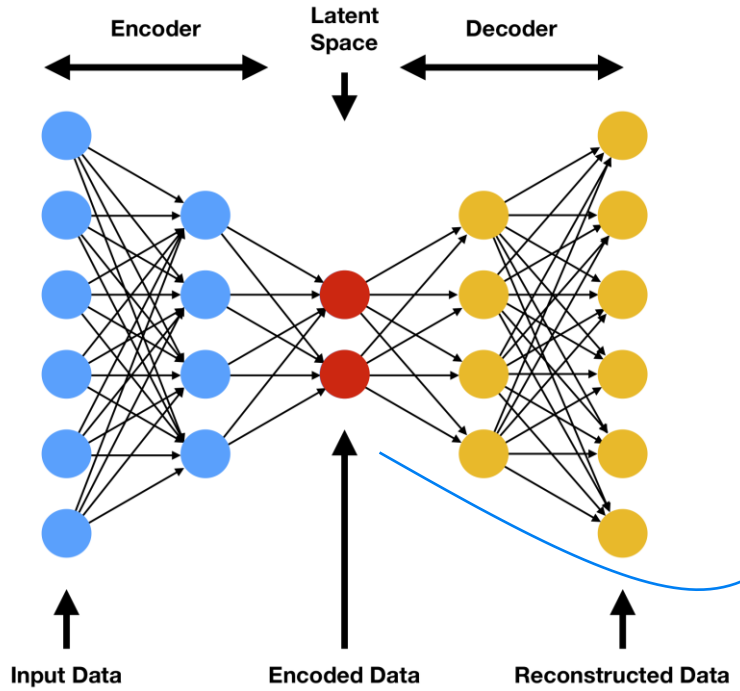


But you need a lot of (labeled) data (Y's)

Reduce dimensionality, autoencoder
Build predictor in reduced space, *embeddings*

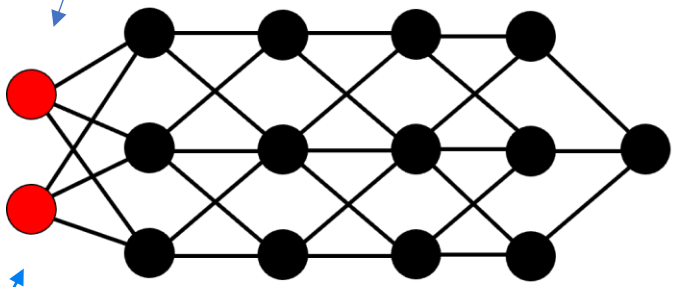


PCA
linear reduction



Autoencoder
non-linear reduction

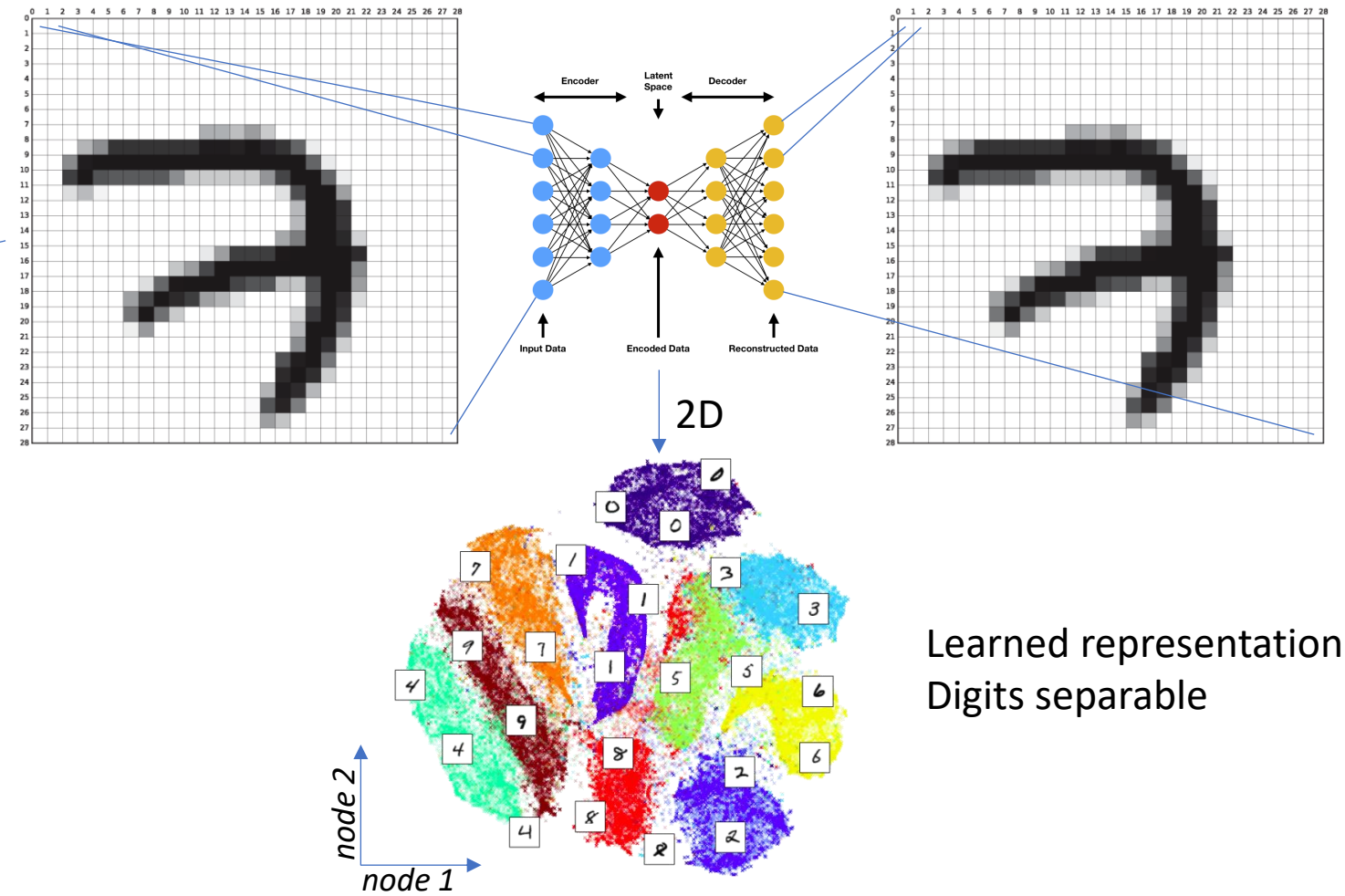
latent space, embedding, learned representation



Y_i

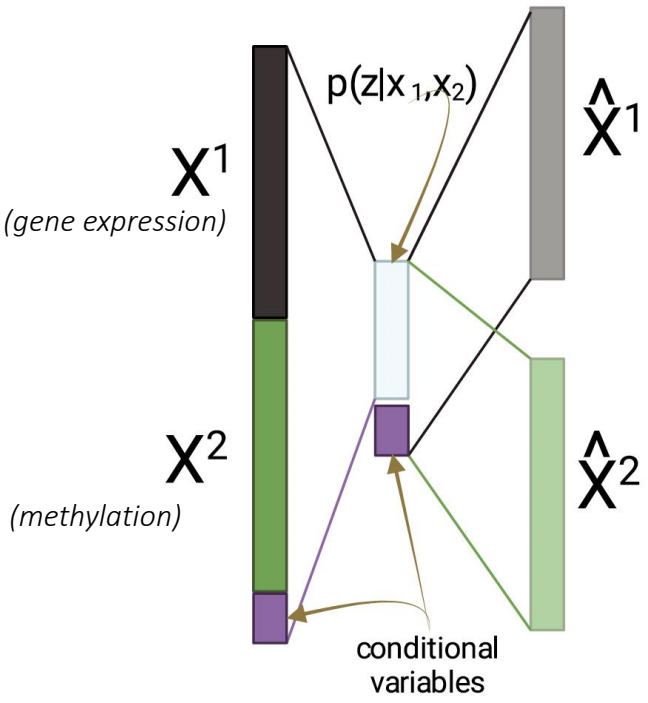
Use learned
representation to predict Y

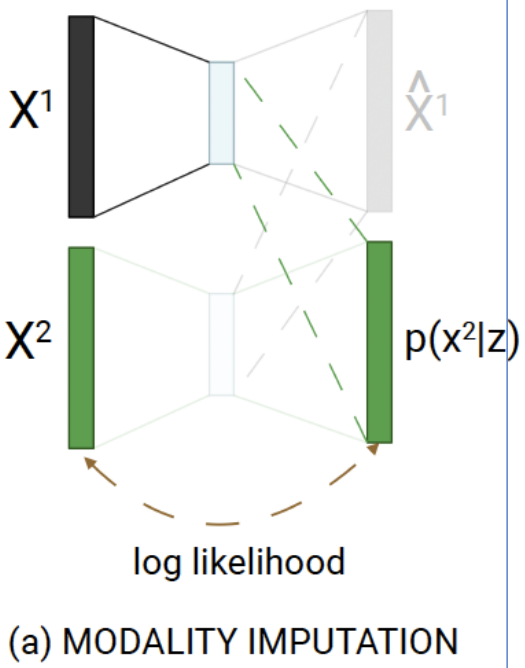
Famous example: MNIST



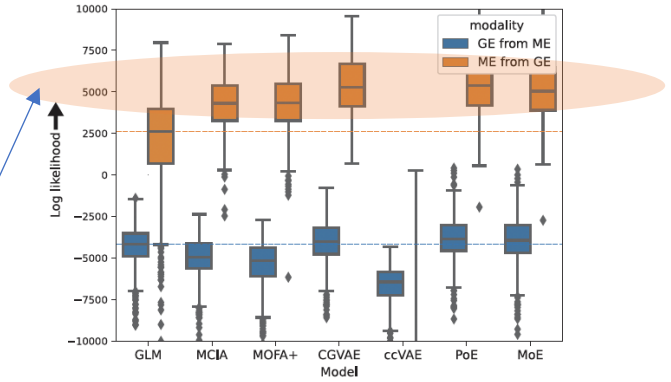
Learned representation
Digits separable

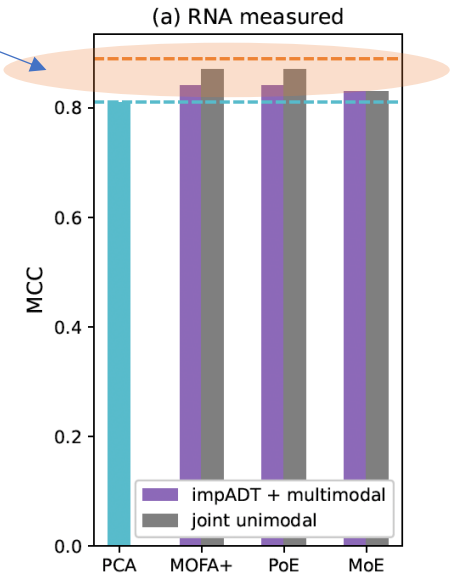
Multi modalities: cross-modality imputation, joint spaces





improvement

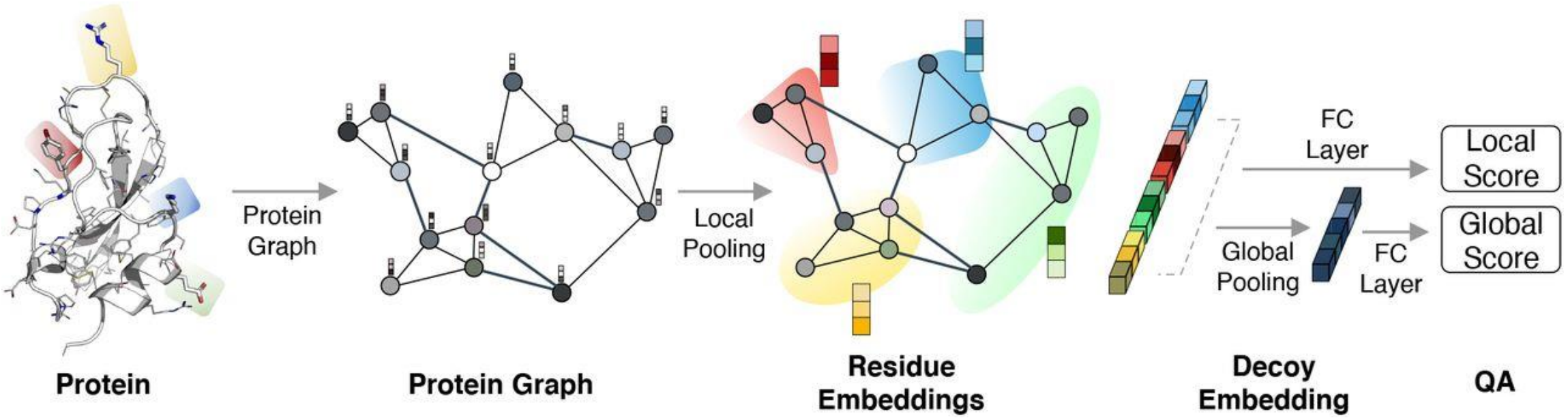




Makrodimitris, Pronk, Abdelaal, Reinders. An in-depth comparison of linear and non-linear joint embedding methods for bulk and single-cell multi-omics, Briefings in Bioinformatics, 2024

Incorporate knowledge into neural network

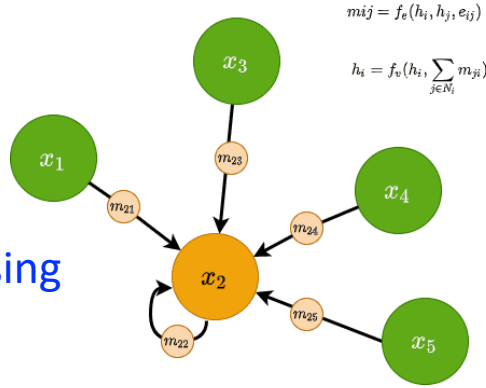
Eg how molecules relate to each other



message passing

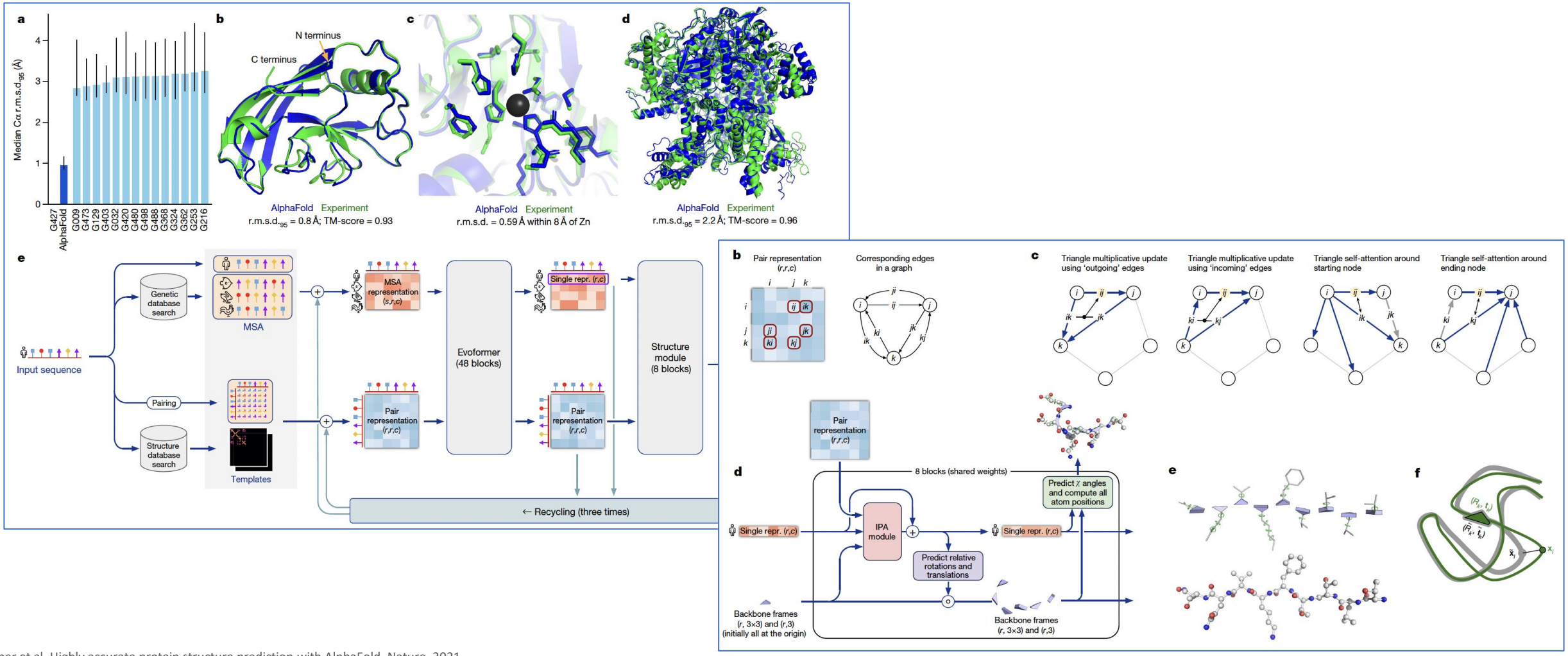
$$m_{ij} = f_e(h_i, h_j, e_{ij})$$

$$h_i = f_v(h_i, \sum_{j \in N_i} m_{ji})$$



Sanyal et al. ProteinGCN: Protein model quality assessment using Graph Convolutional Networks. bioRxiv. 2020

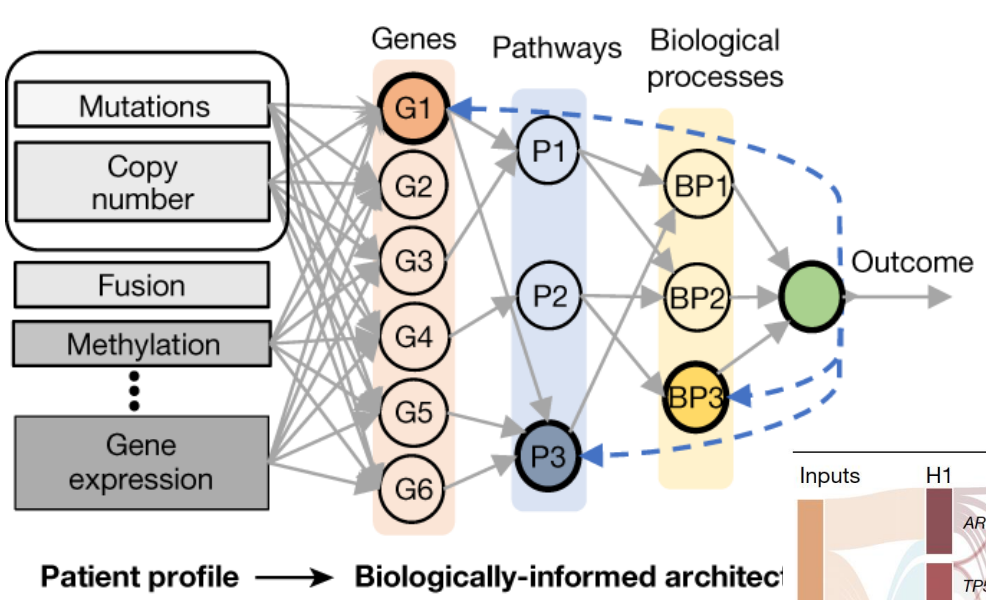
Alfafold: predicting 3D structure of proteins
Based on graph convolutional neural net (GNN)

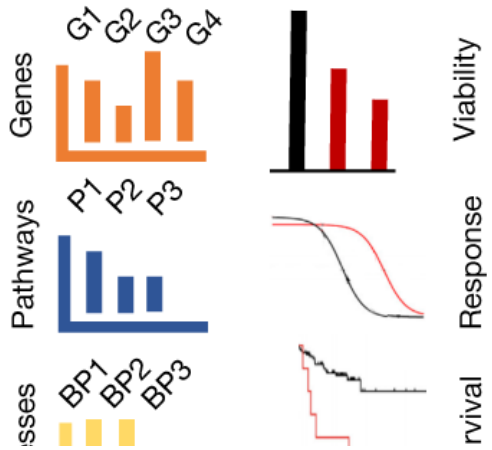


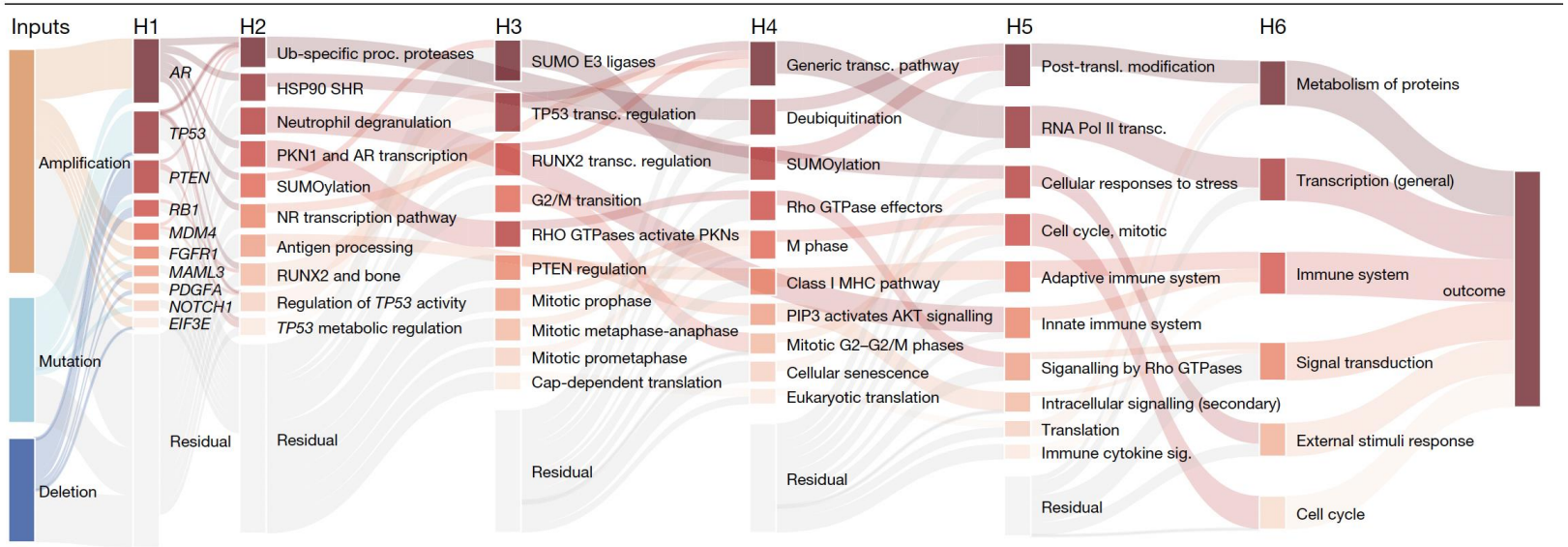
Jumper et al. Highly accurate protein structure prediction with AlphaFold. Nature. 2021.

Incorporate knowledge into neural network

Eg how genes relate to each other



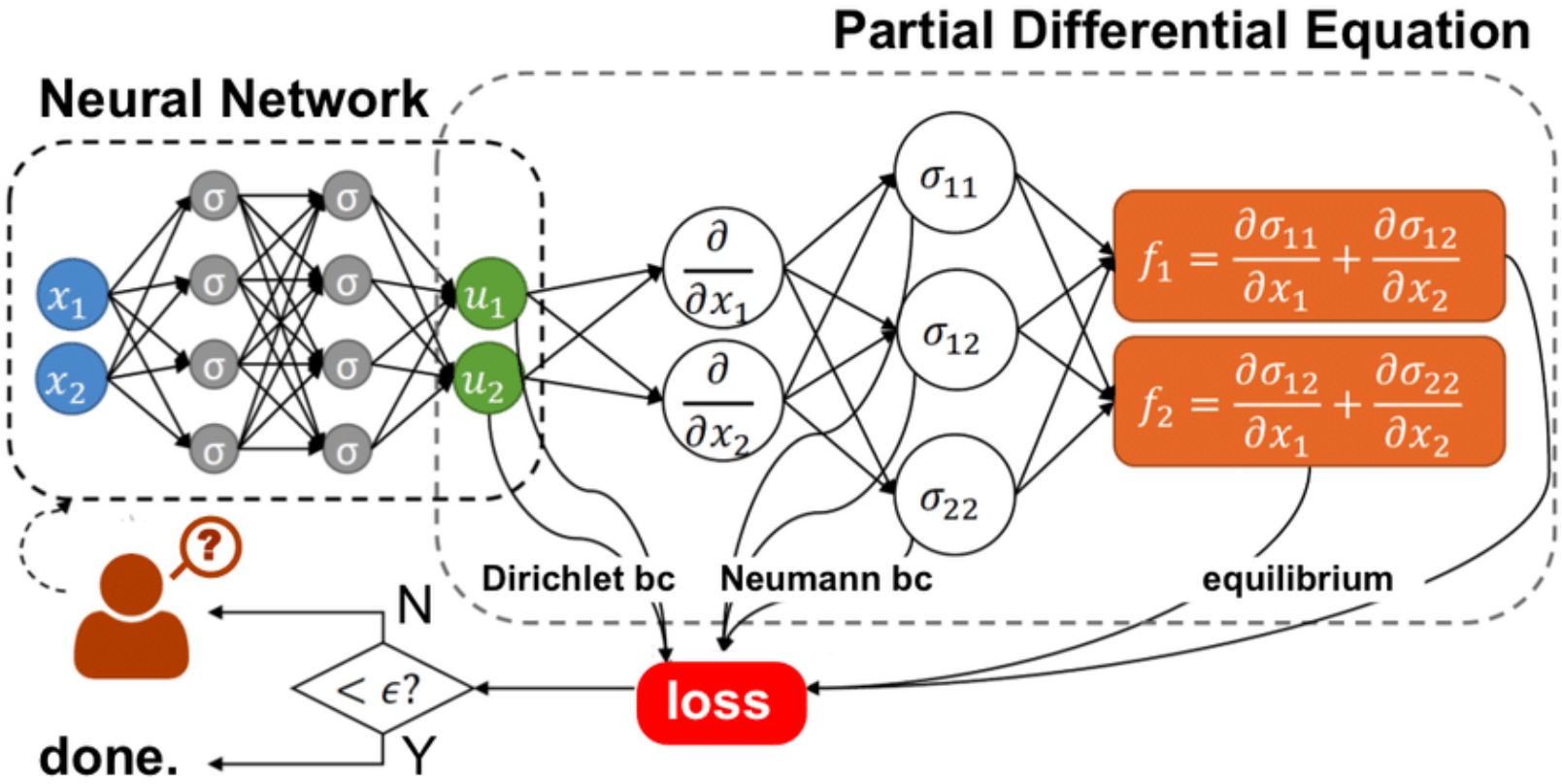


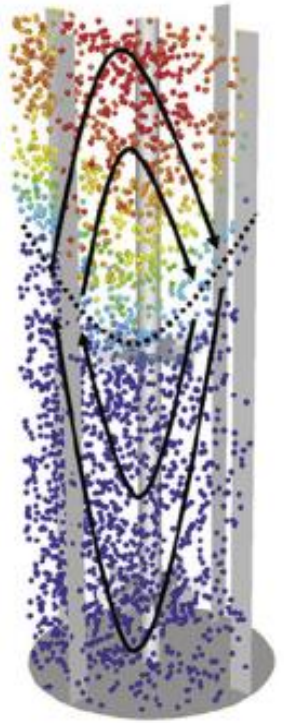


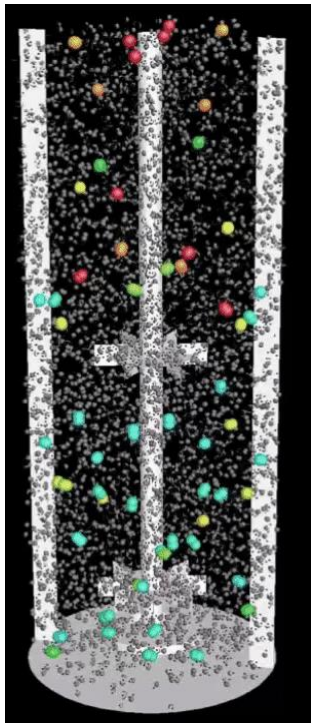
Elmarakeby et al. Biologically informed deep neural network for prostate cancer discovery. Nature. 2021.

Incorporate knowledge into neural network

Eg on physical constraints on output

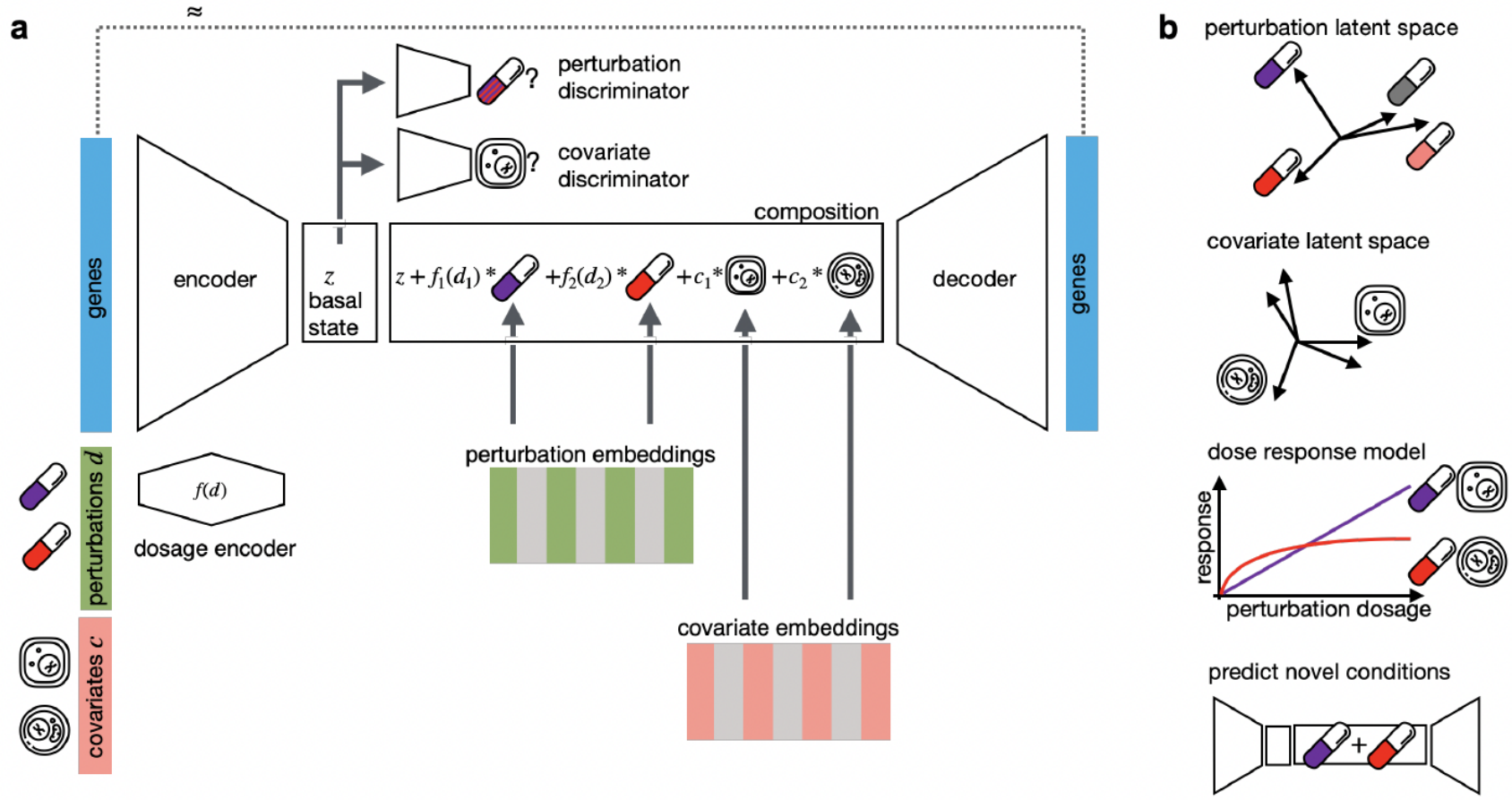






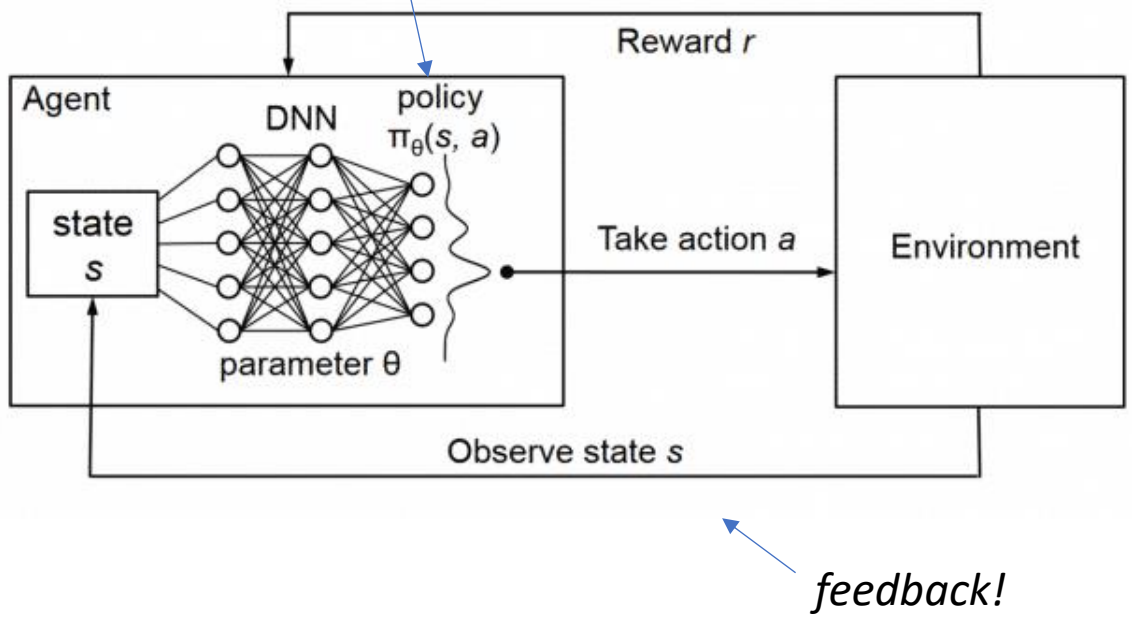
Peng et al. Multiscale Modeling Meets Machine Learning: What Can We Learn? Archives of Computational Methods in Engineering. 2020

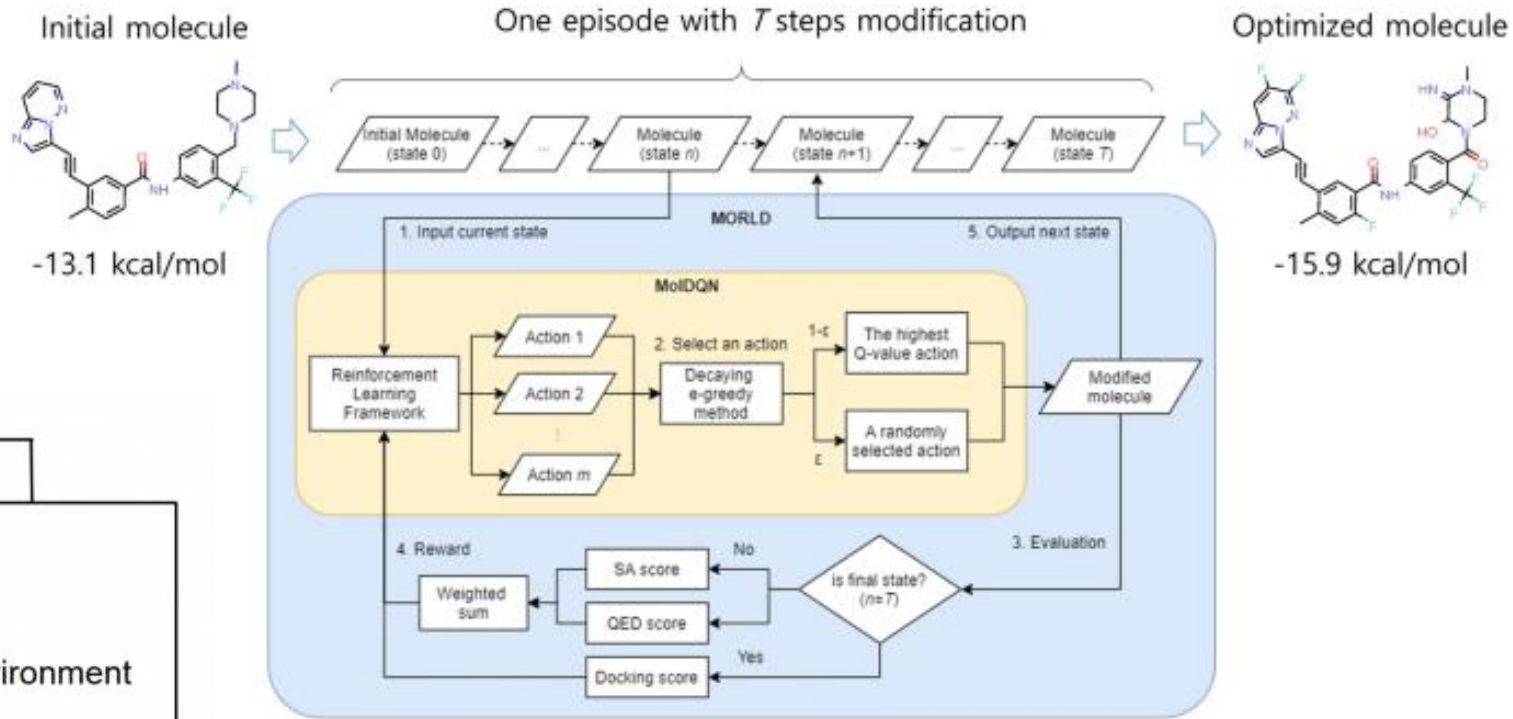
Learn cellular drug/perturbation response

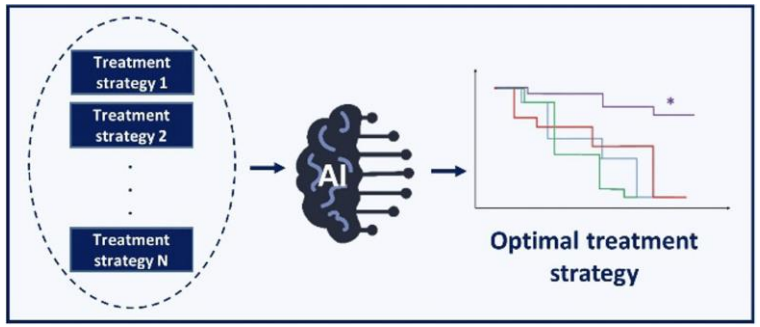


Alessandro Palma, Fabian J Theis, Mohammad Lotfollahi. Predicting cell morphological responses to perturbations using generative modelling. Nature Communications. 2025

Reinforcement learning
Learning strategies



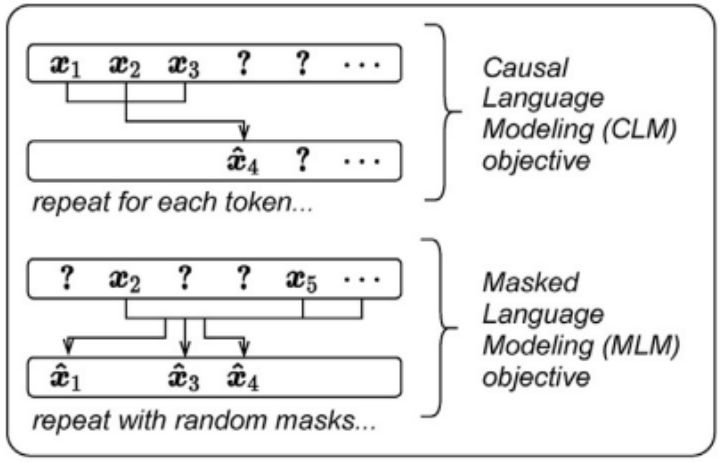




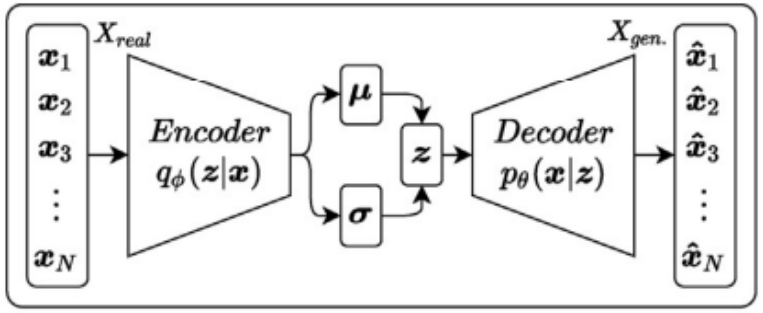
Generative Modelling

Deep generative model families

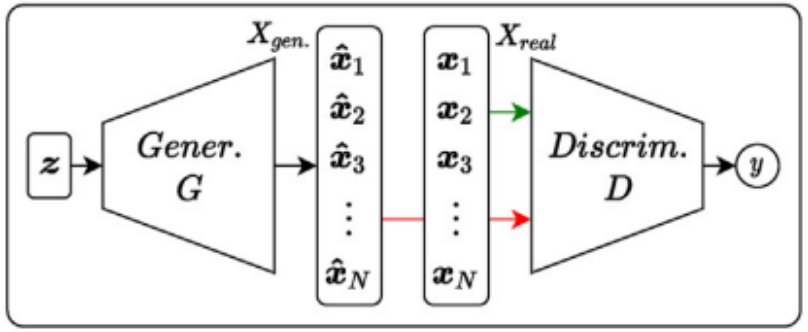
Autoregressive language models (LMs)



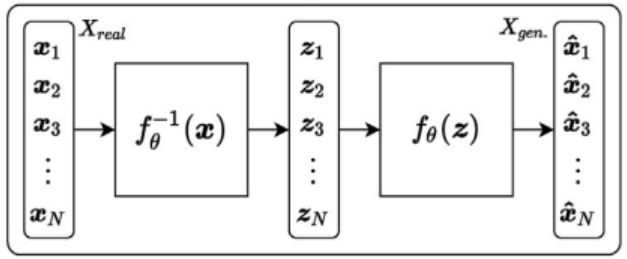
Variational AutoEncoders (VAEs)



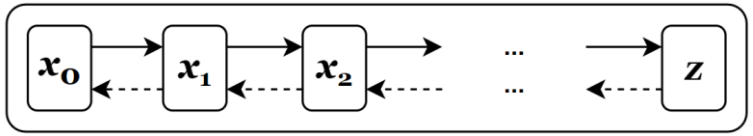
Generative Adversarial Networks (GANs)



Normalising Flows (NFs)



Diffusion Models (DMs)



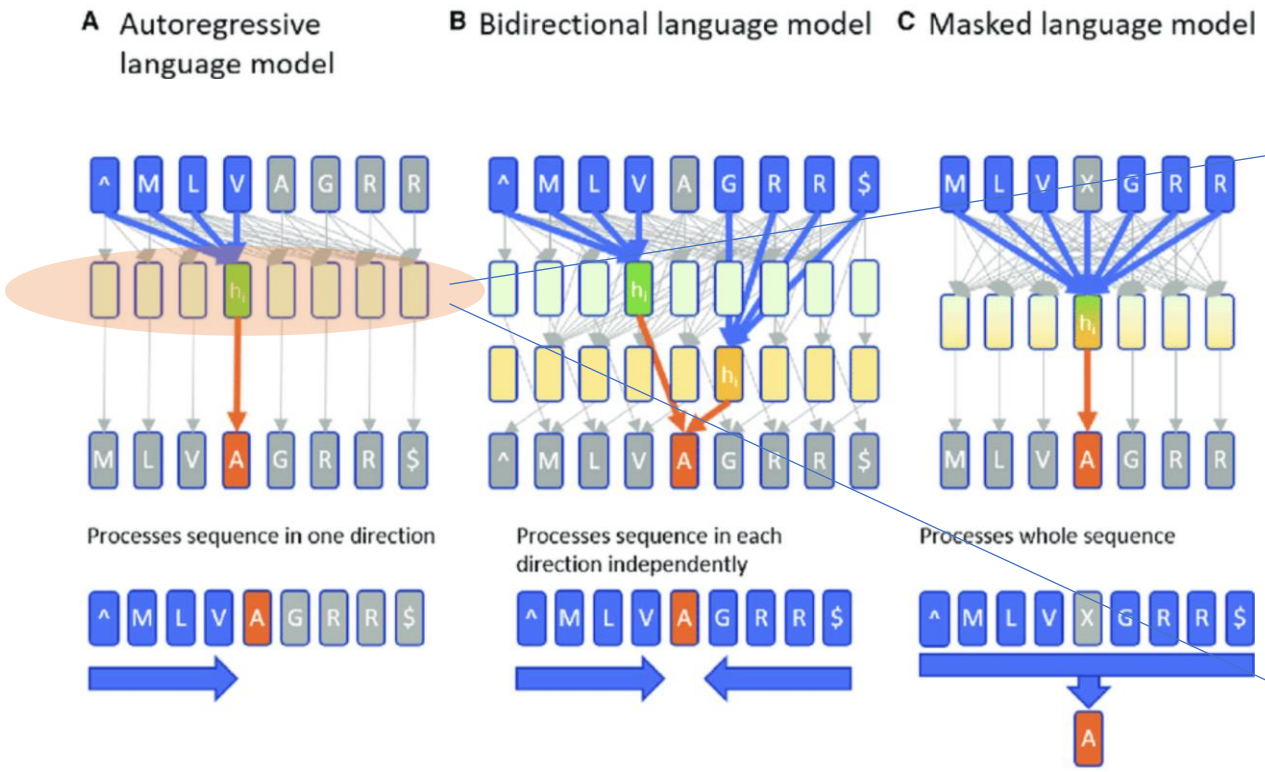
Strokach, A., & Kim, P. M. (2022). Deep generative modeling for protein design. *Current opinion in structural biology*, *72*, 226-236.

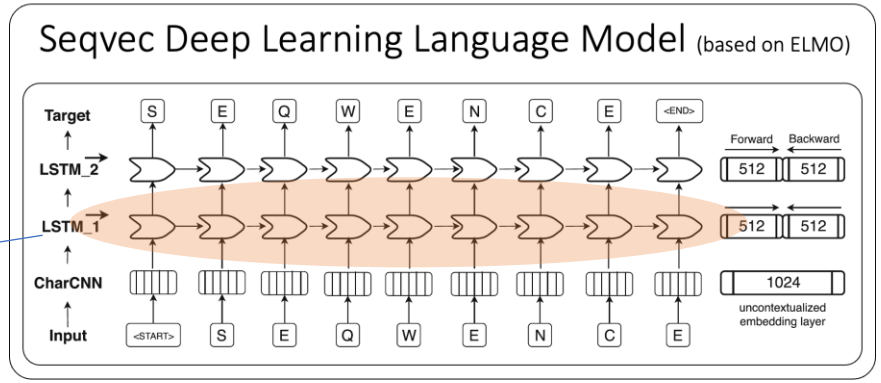
Applications in Bioinformatics

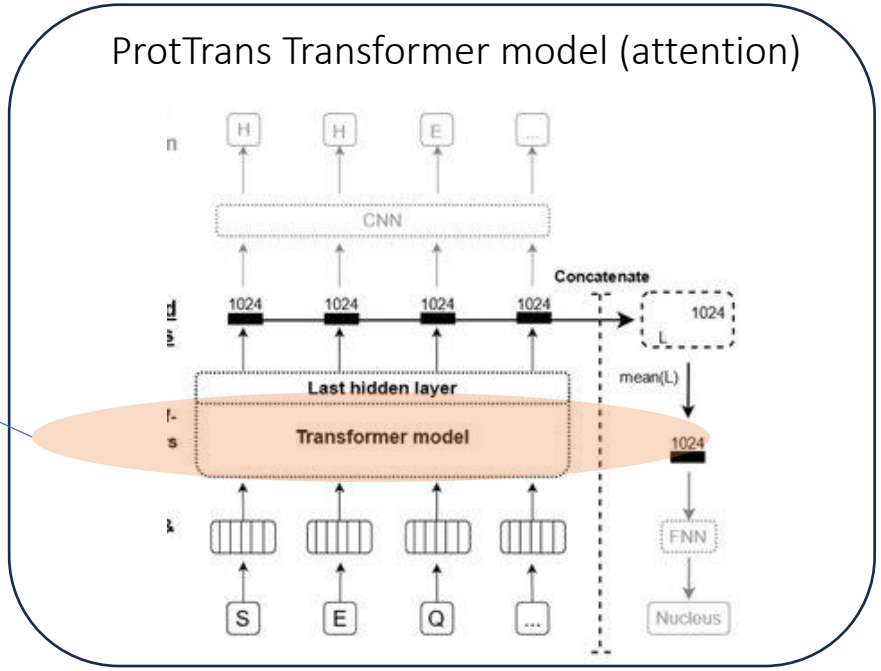
Autoregressive language models (LMs)	e.g., protein design (ProtGPT2, ProGen, …), drug discovery (molGPT, …), DNABert, …
Variational AutoEncoders (VAEs)	e.g., protein design, drug discovery, representation learning in transcriptomics, multimodal data integration, …
Generative Adversarial Networks (GANs)	e.g., data generation/augmentation across different bioinformatics domains (scRNA-seq data, DNA sequences, Protein sequences), …
Diffusion Models (DMs)	e.g., protein folding, protein and small molecule generation, protein-ligand interaction modeling, data analysis (cryo-EM, gene expression), single-cell image, …
Normalising Flows (NFs)	and many more…

Autoregressive Language Model

Autoencode sequences (Language models)

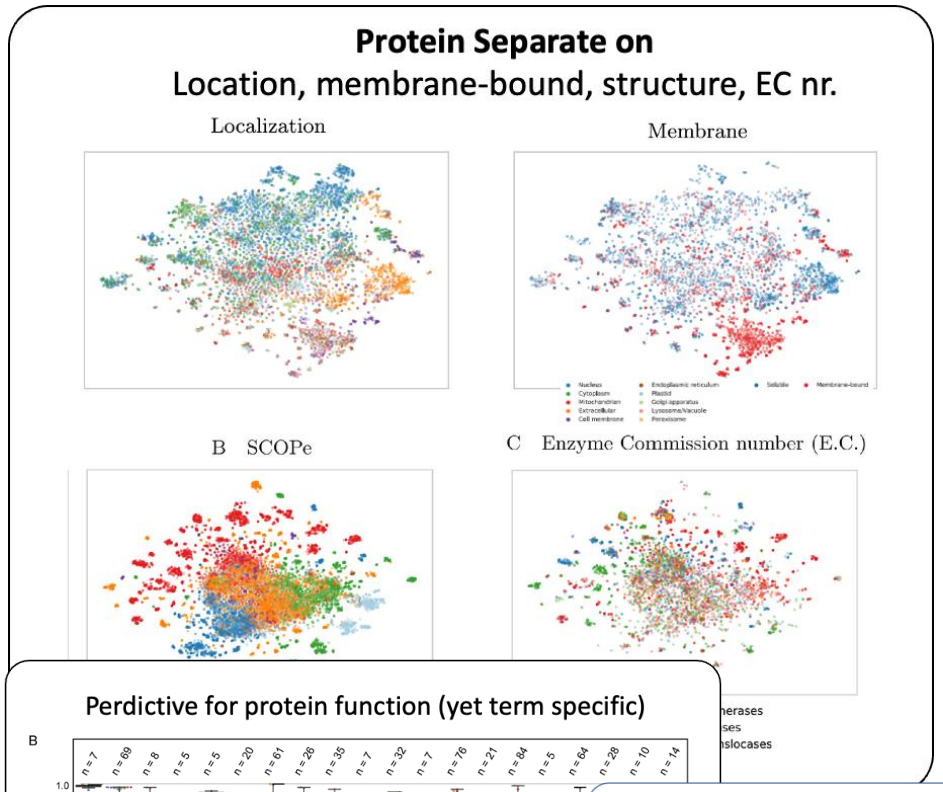


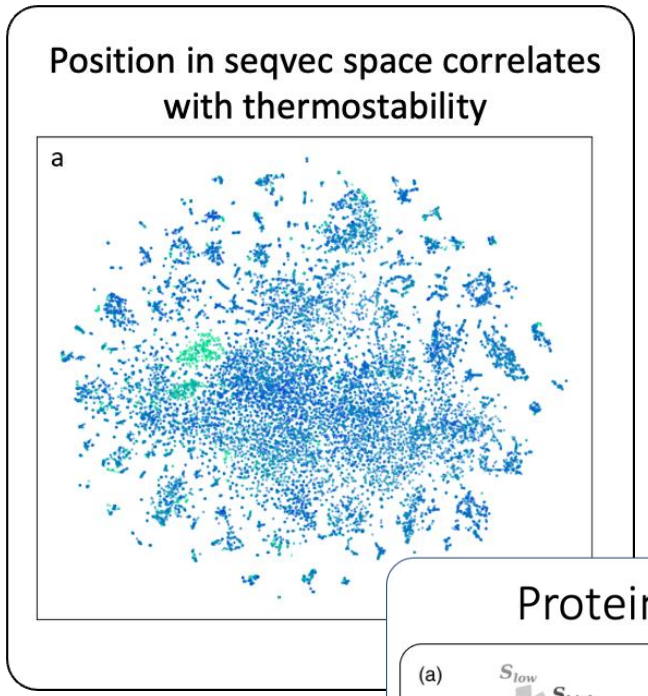


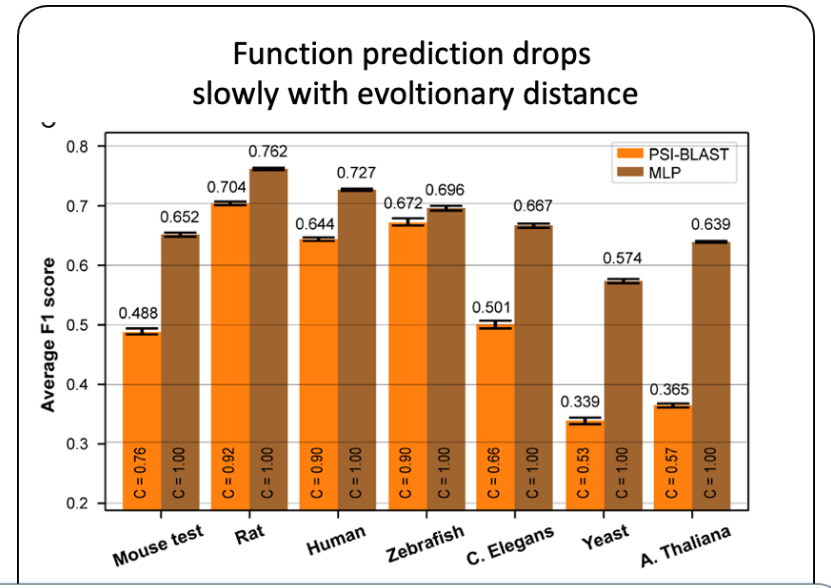


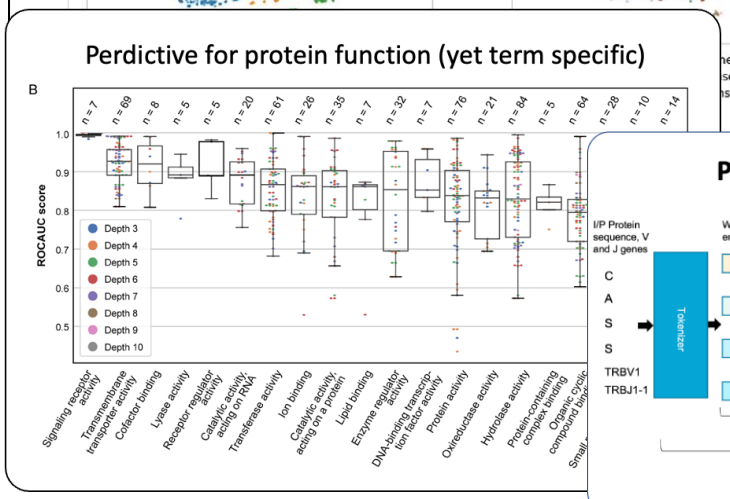
Bepler and Berger. Learning the protein language: Evolution, structure, and function. Cell systems. 2021.
Elnagger et al. ProtTrans: Toward Understanding the Language of Life Through Self-Supervised Learning. IEEE PAMI. 2022
Heinzinger et al. Modeling aspects of the language of life through transfer-learning protein sequences. BMC Bioinformatics. 2019

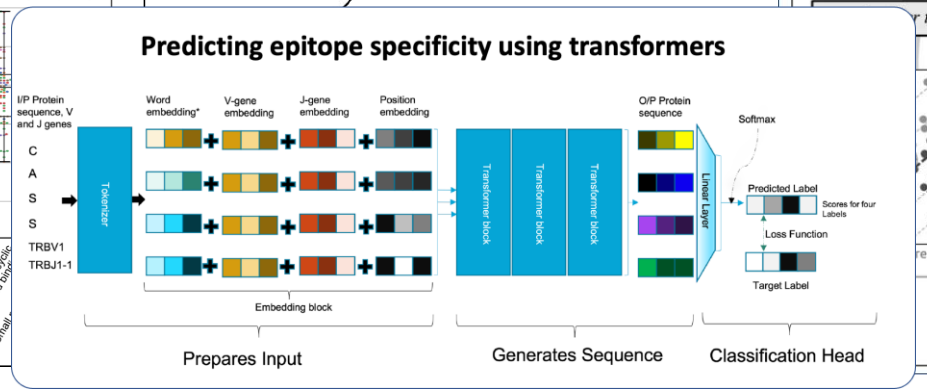
Autoencode sequences (Language modes)
Protein embeddings, function prediction, redesign

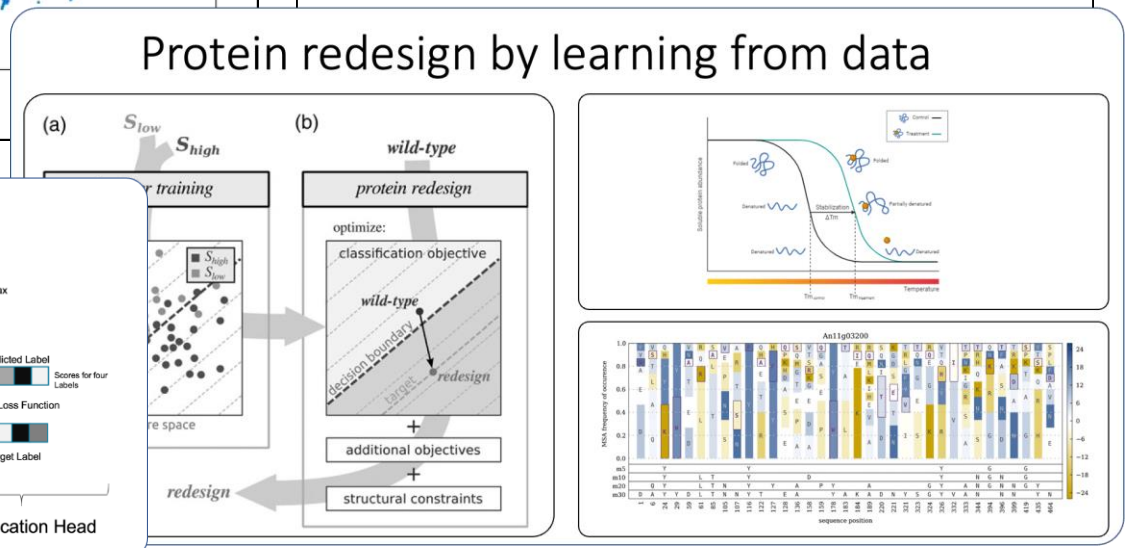






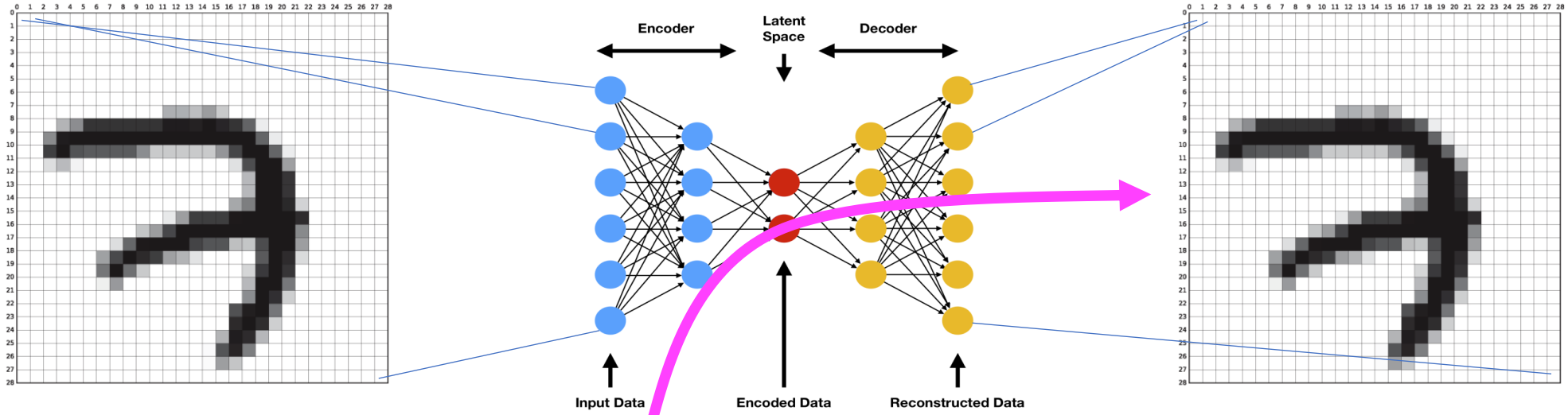




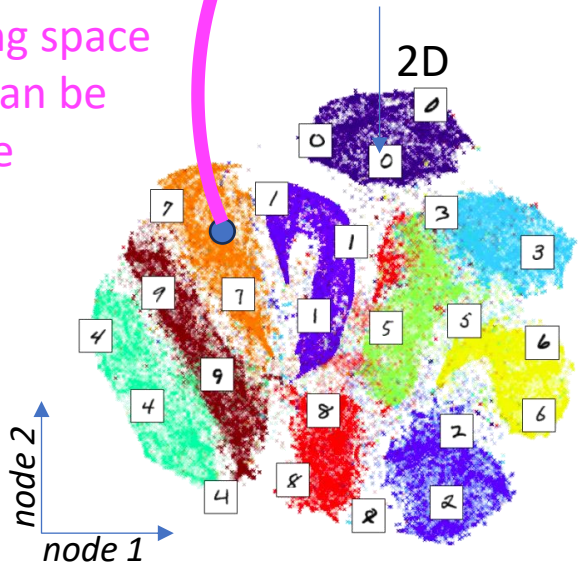


Variational Autoencoder

Revisit Autoencoder



Sampling from embedding space **generates** samples that can be decoded and look like the samples in input space



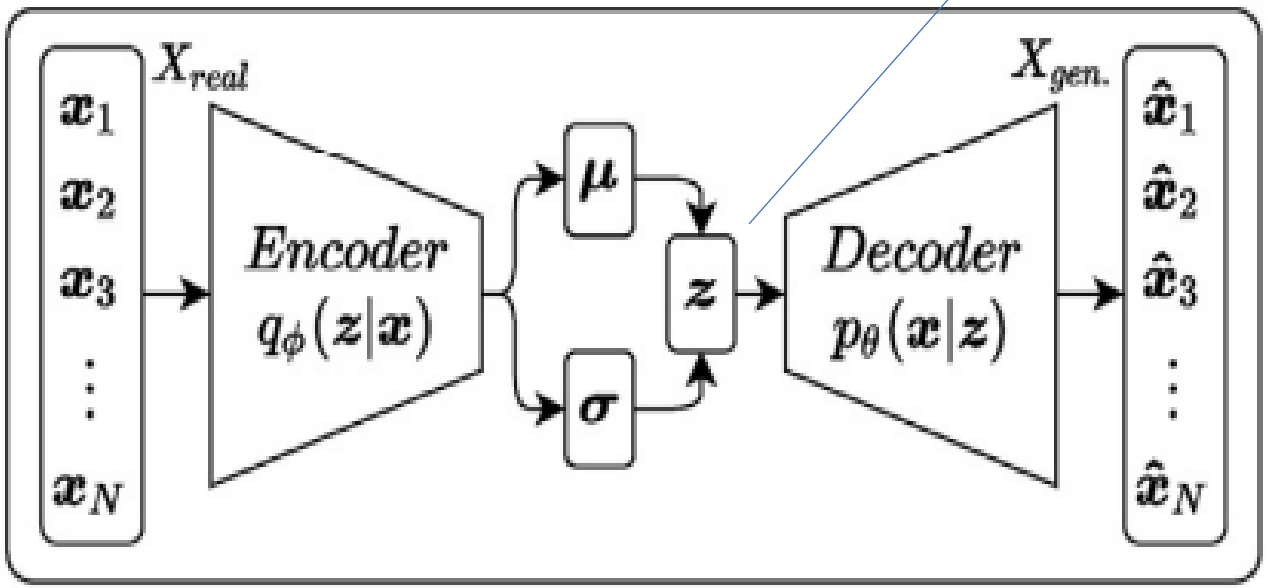
Learned representation
Digits separable

Enforce this behavior more:
Require that samples close to embedded sample should generate the same sample

Probabilistic encoder

Choice prior can be anything, Gaussian, Laplacian, Student t, Mixture of gaussians, etc.

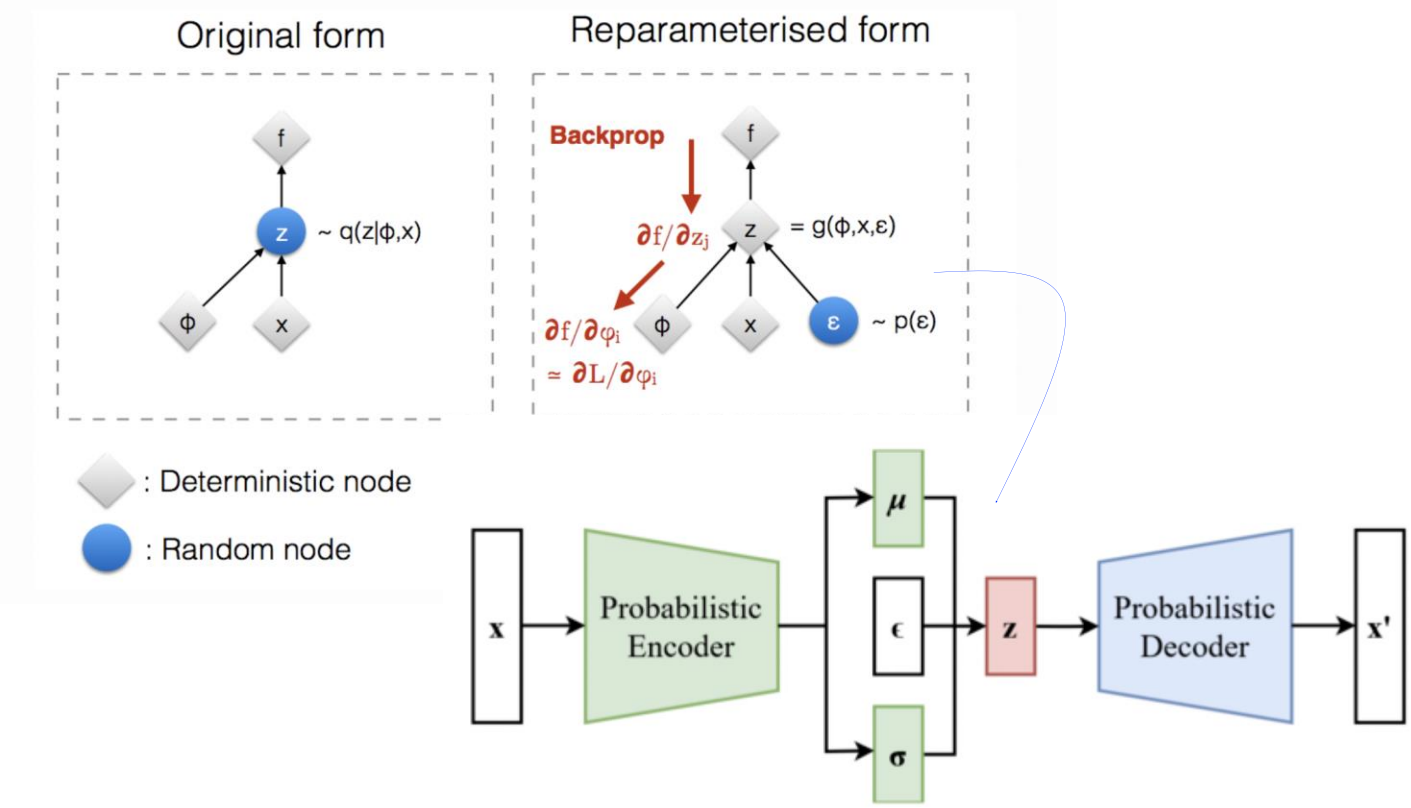
$$z \sim \mathcal{N}(\mu(\mathbf{x}), \sigma^2(\mathbf{x})) \quad \equiv \quad \mathbf{z} = \mu(\mathbf{x}) + \sigma(\mathbf{x})\epsilon, \epsilon \sim \mathcal{N}(0,1)$$



inference model $q_\phi(\mathrm{z}|\mathrm{x})$ generative model $p_\theta(\mathrm{x}|\mathrm{z})$

Backpropagation problem

- Backpropagation cannot flow through the random node
- Solve by reparameterization



Instead of sampling from

$$z \sim q_\phi(z \mid x)$$

Sample from

$$\epsilon \sim N(0,1)$$

And linear transform

$$z = \mu + \sigma \odot \epsilon$$

Alternatively, sampling from

$$z \sim N(\mu, \sigma)$$

Is the same as sampling from

$$\epsilon \sim N(0,1)$$

And setting

$$z = \mu + \sigma \odot \epsilon$$

https://blog.bayeslabs.co/2019/06/04/All-you-need-to-know-about-Vae.html

What to optimize

- For each sample ***z***, there will be two variables ***μ*** and ***σ*** *(defining Gaussian)*
- Accumulation of all Gaussian distributions becomes the original distribution ***P***(***x***)

$$P(x) = \int_z P(z)P(x|z)dz$$

- P(x), likelihood of the data, the bigger the better

$$\text{Maximum } L = \sum_x \log P(x)$$

LogP(x)

$$\log P(x) = \int_z q(z|x) \log P(x) dz \quad = logP(x) \boxed{\int_z q(z|x) dz}$$

equals 1 (trick)

$$= \int_z q(z|x) \log \left(\frac{P(z,x)}{P(z|x)} \right) dz$$

$$= \int_z q(z|x) \log \left(\frac{P(z,x)}{q(z|x)} \frac{q(z|x)}{P(z|x)} \right) dz$$

$$= \int_z q(z|x) \log \left(\frac{P(z,x)}{q(z|x)} \right) dz + \boxed{\int_z q(z|x) \log \left(\frac{q(z|x)}{P(z|x)} \right) dz}$$

$$= \int_z q(z|x) \log \left(\frac{P(x|z)P(z)}{q(z|x)} \right) dz + \boxed{KL\left(q(z|x)||P(z|x)\right)}$$

Kullback Leiber divergence, always bigger then zero

$$\boxed{\log P(x) \geq \int_z q(z|x) \log \left(\frac{P(x|z)P(z)}{q(z|x)} \right) dz \quad = \text{ELBO}}$$

Maximizing P(x) equal to maximizing ELBO

https://medium.com/geekculture/variational-autoencoder-vae-9b8ce5475f68

$$\text{ELBO} = \int_z q(z|x)\log\left(\frac{P(x|z)P(z)}{q(z|x)}\right)dz$$

$$= \int_z q(z|x)\log\left(\frac{P(z)}{q(z|x)}\right)dz + \int_z q(z|x)\log P(x|z)dz$$

$$= -KL\left(q(z|x)||P(z)\right) + \int_z q(z|x)\log P(x|z)dz$$

maximize — *Maximize*

$$-KL\left(q(z|x)||P(z)\right) = \int_z q(z|x)\left(\log P(z) - \log q\left(z|x\right)\right)dz$$

$$= -\frac{1}{2}\sum_{i=1}^{J}\left(1 + \log\left(\sigma_i^2\right) - \mu_i^2 - \sigma_i^2\right)$$

prior on z

$$\int_z q(z|x)\log P(z)\,dz = \int_z N\left(z;\mu,\sigma^2\right)\log N\left(z;0,I\right)dz$$

$$\int_z q(z|x)\log q\left(z|x\right)dz = \int_z N\left(z;\mu,\sigma^2\right)\log N\left(z;\mu,\sigma^2\right)dz$$

$$\text{Maximum}\int_z q(z|x)\log P(x|z)dz$$

P(x|z) (Decoder's output) given that ***q(z|x)*** (Encoder's output) is as high as possible. Similar to AutoEncoder's loss function(reconstruction error):

$$\left(x - \hat{x}\right)^2$$

Some variations (many more)

- β-VAE : Balance between reconstruction loss and KL term, learn to entangle (β>1)

$$\mathrm{L}(\theta, \phi; x) = -\mathbb{E}_{q(z|x)}[log\, p(x|z)] + \beta\, D_{KL}(q(z|x) \,||\, p(z))$$

- β-Total Correlation VAE (β-TCVAE) : Additionally penalizing the total correlation between the latent variables, more statistically independent latent variables

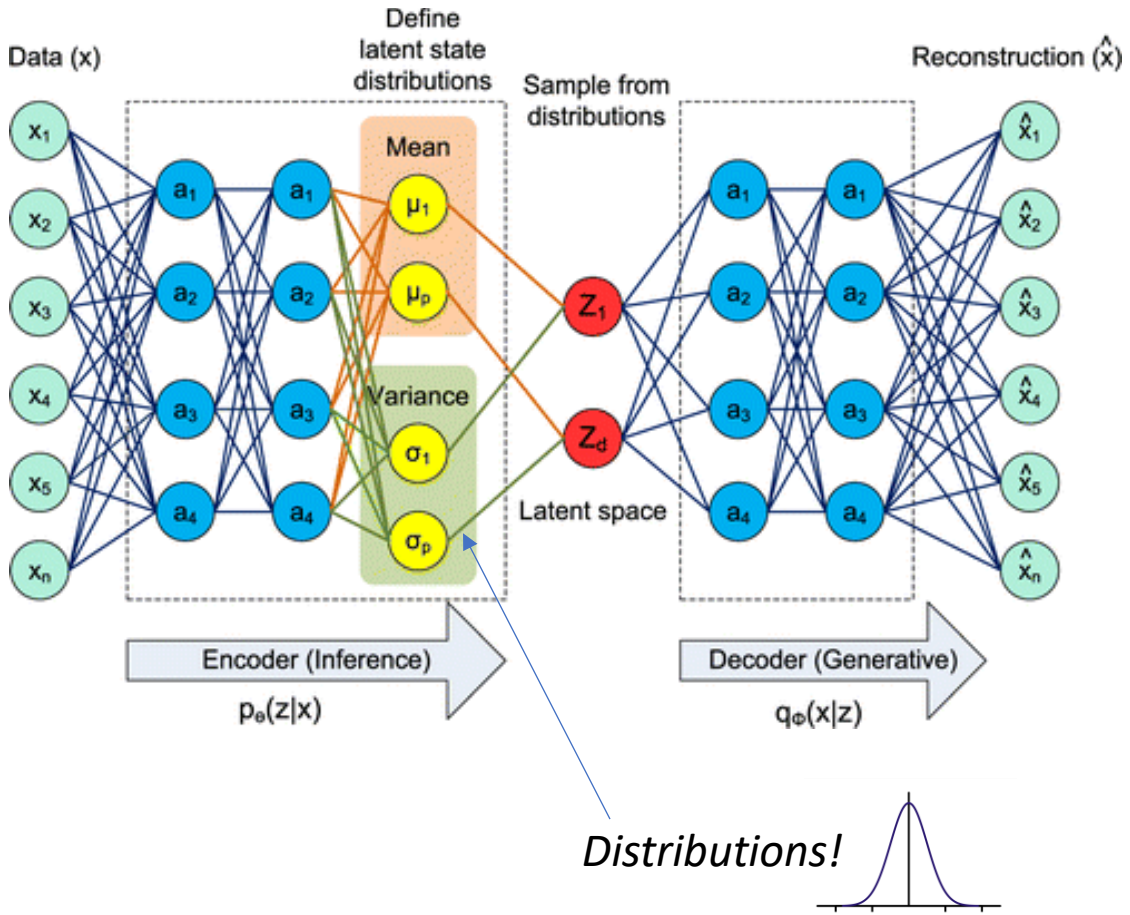
mutual information between the data variable and latent variables

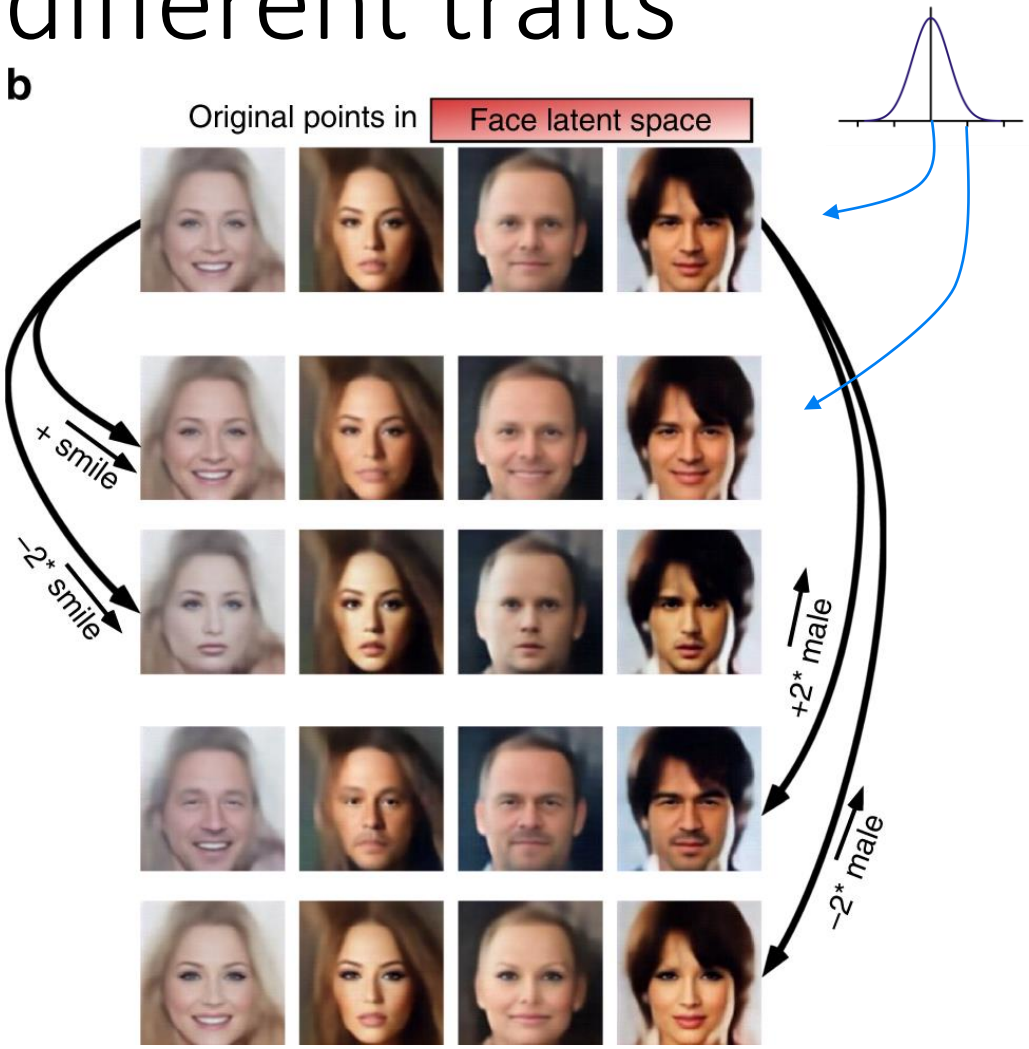
$$\mathrm{L}(\theta, \phi; x, \alpha, \beta, \gamma) = -\mathbb{E}_{q(z|x)}[log\, p(x|z)] + \alpha\, D_{KL}(q(z,x)||q(z)p(x))$$
$$+\beta\, D_{KL}(q(z)||\, \Pi q(z)) + \gamma\, \Sigma D_{KL}(q(z)||p(z))$$

dependence term

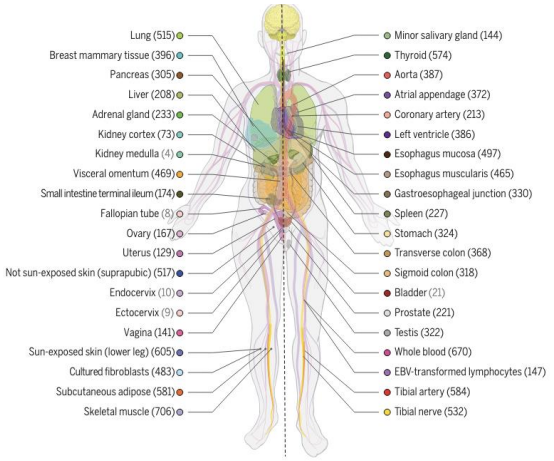
prevent latent variable to diverge from prior

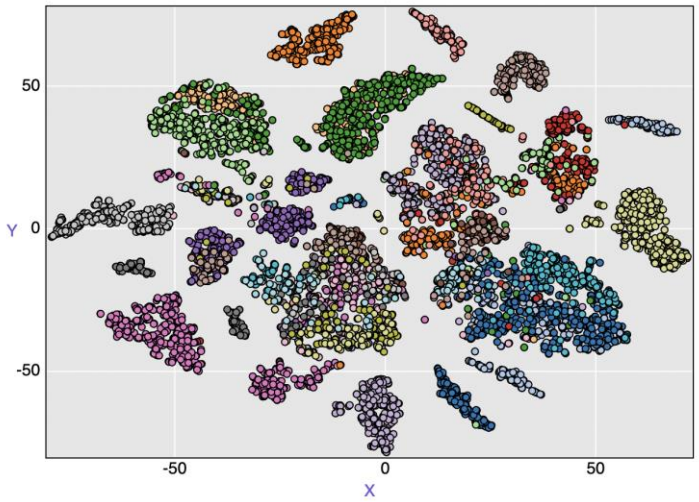
Autoencoder with generating distribution
Latent dimensions can vary different traits



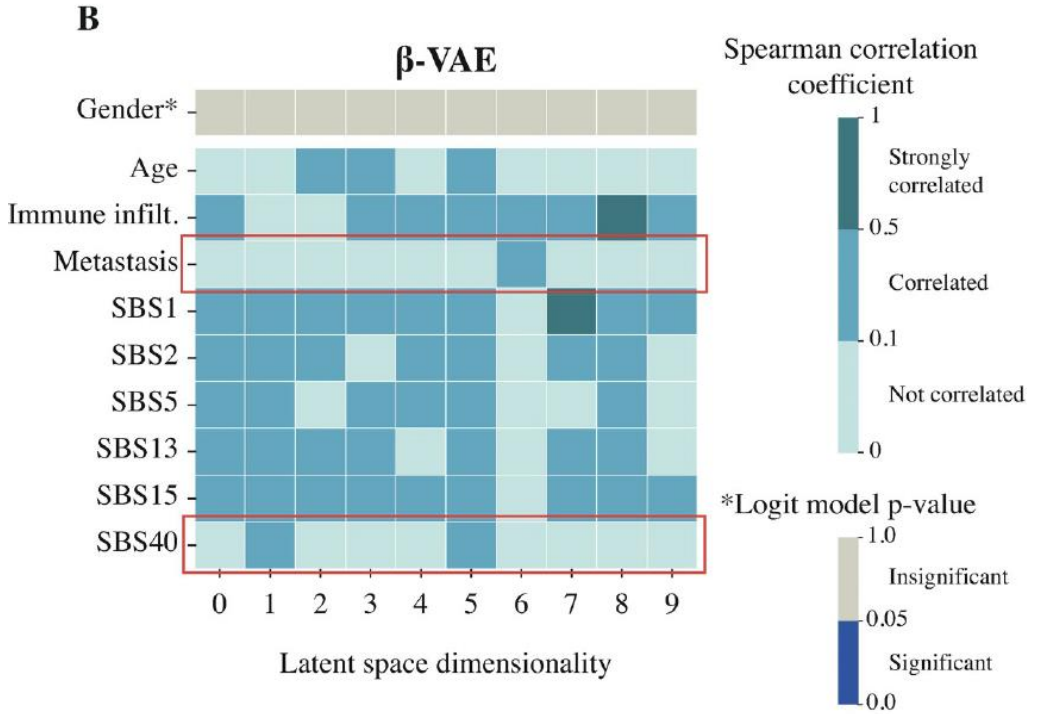


Example: Beta-VAE on GTEX data ; latent factors are disentangled and correlate with data features





17,382 samples with 56,200 genes representing 30 different tissue types



Diffusion model

Idea: Recursively add noise (the diffusion)

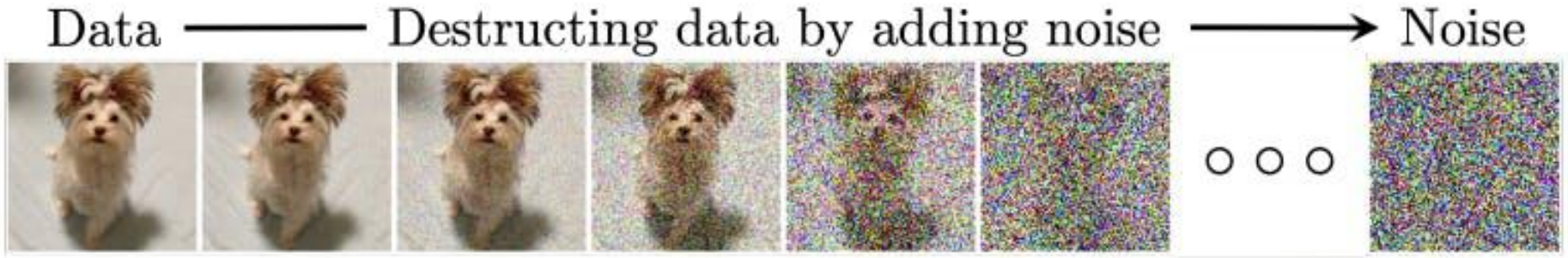
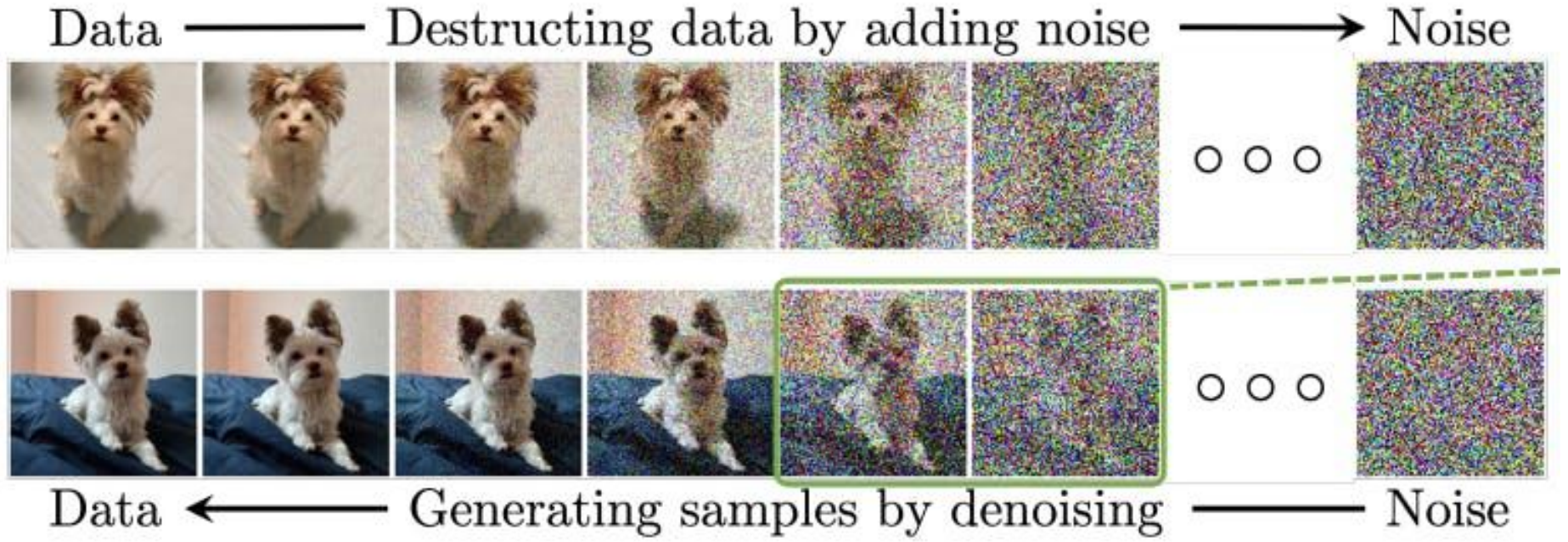


Image is **transformed** to TV static (white noise)

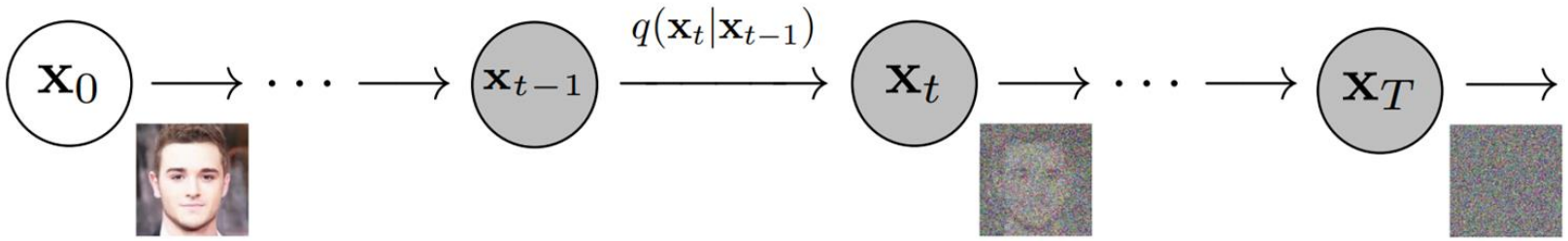
Idea: What if we could reverse this process?



Then we can draw random noisy image, and generate an image like our training data

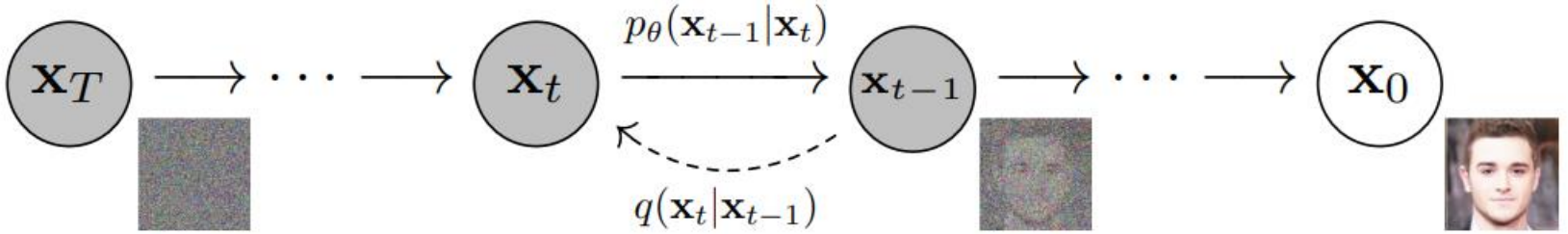
Diffusion Probabilistic Models

- **Forward diffusion process:** Markov chain gradually adds noise to data to obtain approximate posterior



$q(x_T|x_{T-1}) \approx N(0,1)$

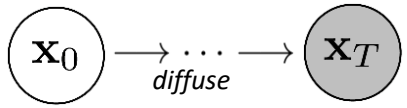
- **Generative diffusion process:** use the Markovian assumption to learn the reverse process



$p(x_T) = N(0,1)$

This we are going to do by trained NN

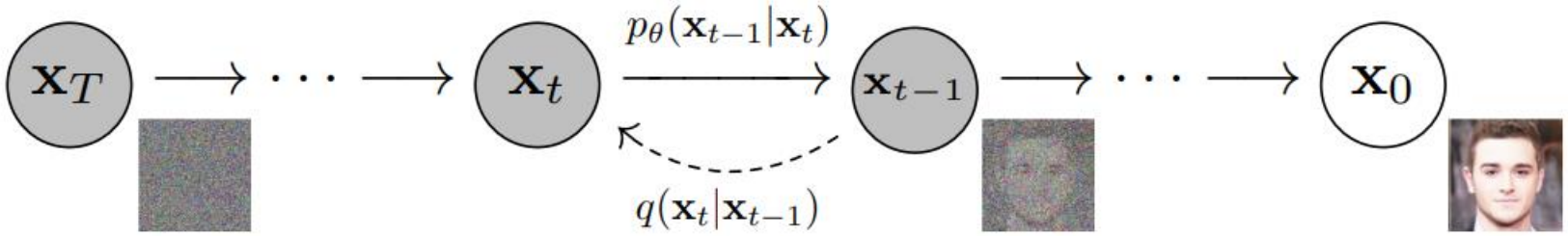
J. Ho, *et al.* "Denoising diffusion probabilistic models." In *NeurIPS* 2020



Intuitive

θ *refers to model*

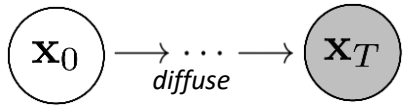
- Need to learn $q(\mathbf{x}_t|\mathbf{x}_{t-1})$ which is our model $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$



- Provide any NN architecture with $(\mathbf{x}_{t-1}, \mathbf{x}_t)$ and learn to predict $\mathbf{x}_{t-1}$ from $\mathbf{x}_t$

1) Want to optimize that for all diffusion steps
2) Need to do that for many diffusions, and for all time steps (to be able to generate from noise)

more formal
(sorry, some more math)



The forward process (diffusion process) (2)

The posterior distribution after T steps

Corrupt data by sampling from a multivariate Gaussian distribution mean centered around the previous state

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

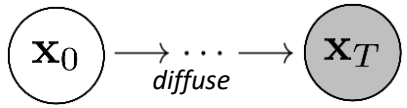
The product of each state at time *t* given the previous state *t-1*

Where $\beta_1, \ldots, \beta_T$ is a variance schedule (either learned or fixed)

Too large β will corrupt image too quickly. Very difficult to undo

Too small will take a long time to learn

Tutorials and math derivations:
https://cvpr2022-tutorial-diffusion-models.github.io/
https://towardsdatascience.com/understanding-diffusion-probabilistic-models-dpms-1940329d6048
https://lilianweng.github.io/posts/2021-07-11-diffusion-models/



The forward process (diffusion process) (3)

- Property of the forward process is that it admits sampling x_t at an arbitrary timestep *t* in closed form (*one step instead of sequence of steps*)

$$q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I})$$

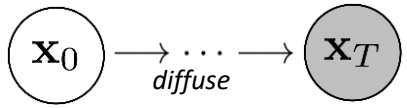
$$\alpha_t := 1 - \beta_t \quad \bar{\alpha}_t := \prod_{s=1}^{t} \alpha_s$$

- Allows to rewrite, *reverse step*(!) (remember *q* was defined with: $q(\mathbf{x}_t|\mathbf{x}_{t-1})$

$$q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_{t-1}; \tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0), \tilde{\beta}_t\mathbf{I})$$

$$\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0) := \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\mathbf{x}_0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{x}_t \qquad \tilde{\beta}_t := \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$$

Available, can be used for training!



The reverse process (reverse diffusion process)

The parameterised Markov chain mapping noise back to image

Different time steps are associated with different noise levels

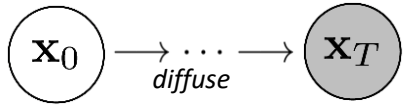
$$p_\theta(\mathbf{x}_{0:T}) := p(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) := p(\mathbf{x}_T) \prod_{t=1}^{T} \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t))$$

$p(x_T) = q(x_T|x_0) \approx N(0, 1)$

These need to be learned

Note: Reverse diffusion transition distribution depends only on the previous timestep (Markov):

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) := \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t))$$



Reverse Markov transitions (1)

- Reverse Markov transitions:

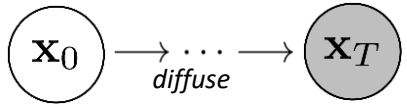
$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) := \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t))$$

- Find the one that maximize the likelihood of the training data

$$L_{vlb} = L_0 + L_1 + ... + L_{T-1} + L_T$$

- Use Kullback-Leibler Distance to measures distance between two distributions

$$L_{t-1} = D_{KL}(q(x_{t-1}|x_t, x_0) \,||\, p_\theta(x_{t-1}|x_t))$$



Reverse Markov transitions (2)

- Simplify: Set the variances equal to variances in forward process schedule :

$$\Sigma_\theta(x_t, t) = \sigma_t^2 \mathbb{I} \qquad \sigma_t^2 = \beta_t$$

- Reverse Markov transition becomes:

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) := \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t) := \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \sigma_t^2 \mathbf{I})$$

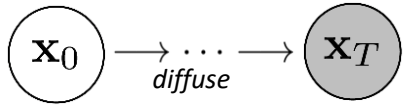
- Allows to transform (rewrite) Kullback-Leiber distance to:

known *model that we want*

$$L_{t-1} \propto ||\tilde{\mu}_t(x_t, x_0) - \mu_\theta(x_t, t)||^2$$

- The most straightforward parameterization of $\mu_\theta(x_t, t)$ is a model that predicts the $\tilde{\mu}_t(x_t, x_0)$ forward process posterior mean (known/training!)

$$\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0) := \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\mathbf{x}_0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{x}_t$$



Some rewriting

- Reparametrize forward step $q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I})$

$$\mathbf{x}_t(\mathbf{x}_0, \boldsymbol{\epsilon}) = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon} \qquad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

- Rewrite posterior mean

rewrote x_0

$$\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0) \longrightarrow \tilde{\boldsymbol{\mu}}_t\left(\mathbf{x}_t(\mathbf{x}_0, \boldsymbol{\epsilon}), \frac{1}{\sqrt{\bar{\alpha}_t}}(\mathbf{x}_t(\mathbf{x}_0, \boldsymbol{\epsilon}) - \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon})\right)$$

$$\longrightarrow \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t(\mathbf{x}_0, \boldsymbol{\epsilon}) - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}\right)$$

$\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0) := \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\mathbf{x}_0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{x}_t \qquad \alpha_t := 1-\beta_t \quad \bar{\alpha}_t := \prod_{s=1}^{t} \alpha_s$

Prediction model, sampling becomes

- Remember

$$\mu_\theta(x_t, t) = \tilde{\mu}_t(x_t, x_0)$$

- Becomes now

$$\mu_\theta(x_t, t) = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(\mathbf{x}_t, t)\right)$$

our model

- ϵ_θ is available function approximator intended to predict ϵ from $\mathbf{x}_t$: **LEARN**
- *To sample* $\mathbf{x}_{t-1} \sim p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$ *to compute:*

$$\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(\mathbf{x}_t, t)\right) + \sigma_t \mathbf{z} \qquad \mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

phew
(now algorithm ☺)

DPM training and sampling

Algorithm 1 Training

1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \text{Uniform}(\{1, \dots, T\})$
4: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: Take gradient descent step on
$$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t) \right\|^2$$
6: **until** converged

predicted noise

$\mathbf{x}_t$, the noised image at timestep t

Algorithm 2 Sampling

1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ — sample from prior
2: **for** $t = T, \dots, 1$ **do**
3: $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
4: $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \mathbf{z}_\theta(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$
5: **end for**
6: **return** $\mathbf{x}_0$

predicted noise

add noise back for stochasticity (and ensure noise remains Gaussian)

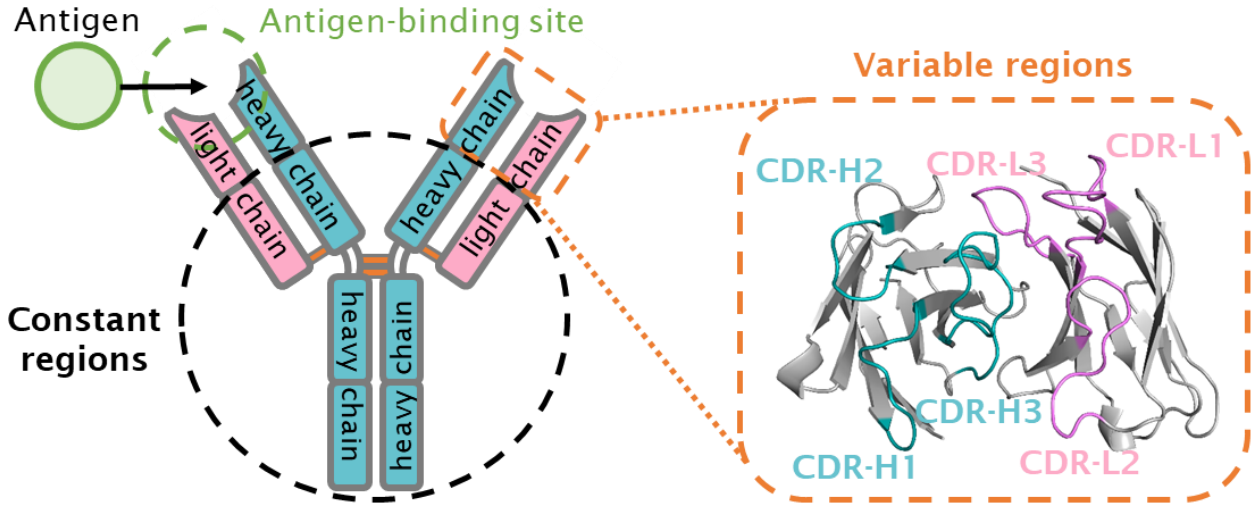
J. Ho, *et al.* "Denoising diffusion probabilistic models." In *NeurIPS* 2020

Example Diffusion Model
Generating Antibodies

Introduction to antibodies

- Antibody (**Ab**) or Immunoglobulin (Ig)
  Y-shaped protein produced by the immune system in response to **antigens (Ag)**

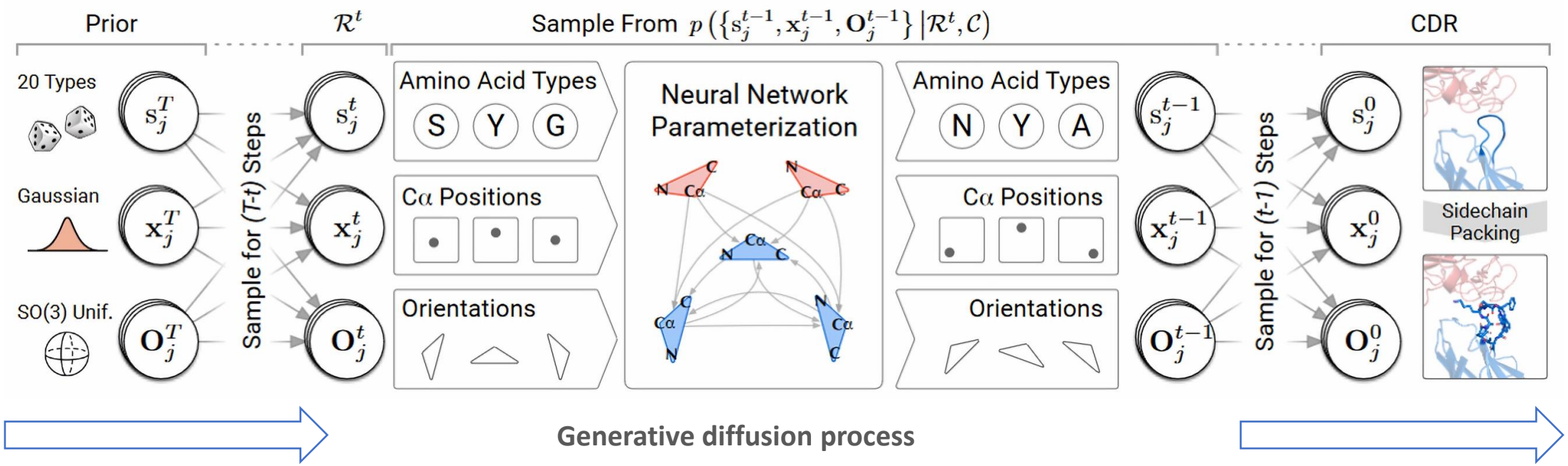
- Two **heavy** (H) and two **light** (L) chains
- **Constant** regions (mostly H)
- **Variable** regions (H/L)
  Antigen-binding site (paratope)
  Framework region + CDR loops
  CDR-H3 loop → most variable



- Monoclonal antibody (**mAb**) → engineered in the lab

Diffusion model for antibody design – DiffAb

- In antibody design, diffusion models used to **generate CDR sequences and structures**
- Condition on **bound antibody-antigen complex**
- **Multimodal** → amino acid types, Cα atom coordinates, and orientations in SO(3)



S. Luo, *et al*. "Antigen-specific antibody design and optimization with diffusion-based generative models." In *NeurIPS* 2022

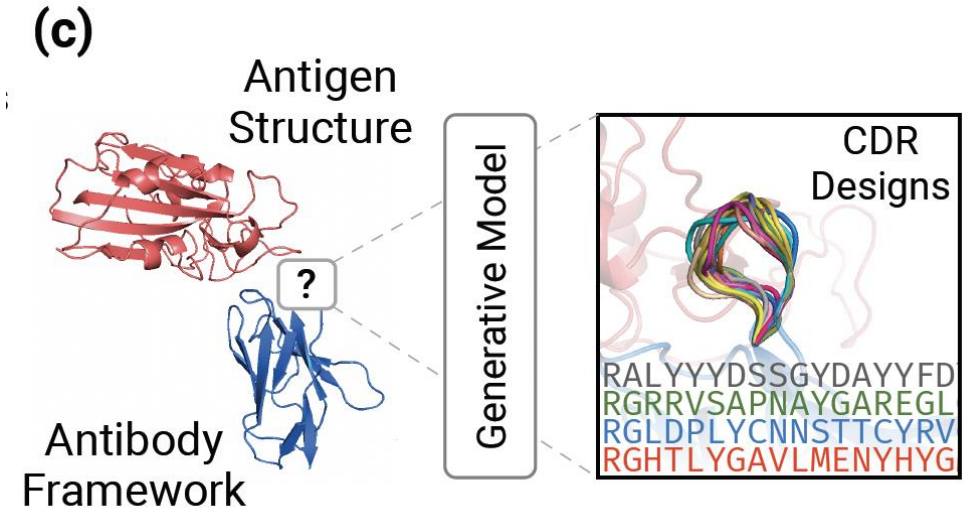
DiffAb – Diffusion processes

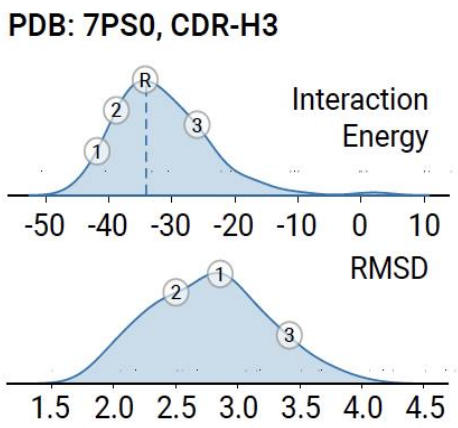
Feature	**Forward diffusion process ($t = 0, \dots, T$). Distributions q**	**Prior distribution (sampling from $t = T$)**	**Generate diffusion process ($t = T, \dots, 0$). Neural network parameterization**
Amino acid types	Multinomial distribution: $q(s_j^t \mid s_j^0) = \text{Multinomial}\left(\bar{\alpha}^t \cdot \text{onehot}(s_j^0) + \frac{1-\bar{\alpha}^t}{20} \cdot \mathbf{1}\right)$	Uniform distribution over 20 classes	***F*** → MLP decoder to predict the probabilities of the 20 amino acids
Cα coordinates	Normal distribution: $q(\mathbf{x}_j^t \mid \mathbf{x}_j^0) = N\left(\mathbf{x}_j^t \mid \sqrt{\bar{\alpha}^t} \cdot \mathbf{x}_j^0, (1-\bar{\alpha}^t)\mathbf{I}\right)$	Standard normal distribution	***G*** → MLP decoder to predict the coordinate deviation wrt the current orientation in the local frame
Orientations in SO(3)	Isotropic Gaussian distribution in SO(3): $q(\mathbf{O}_j^t \mid \mathbf{O}_j^0) = IG_{\text{SO}(3)}\left(\mathbf{O}_j^t \mid \text{ScaleRot}\left(\sqrt{\bar{\alpha}^t} \cdot \mathbf{O}_j^0\right), 1-\bar{\alpha}^t\right)$	Uniform distribution over SO(3)	***H*** → MLP decoder to predict the so(3) vector that is converted to orientation matrix

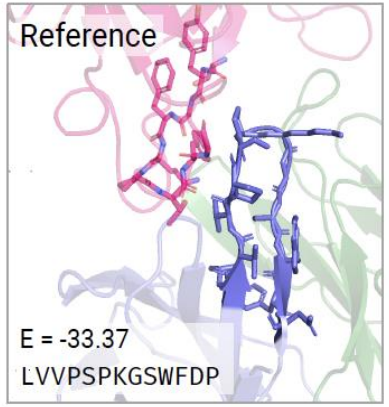
- ***F, G, H*** share an encoder (3D attention layers) of single and pairwise features from the previous timestep
- These networks are **equivariant to the rotation and translation** of the overall structure

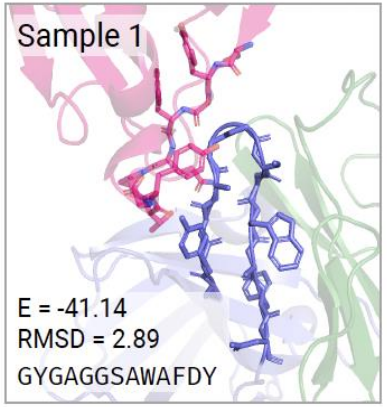
S. Luo, *et al*. “Antigen-specific antibody design and optimization with diffusion-based generative models.” In *NeurIPS* 2022

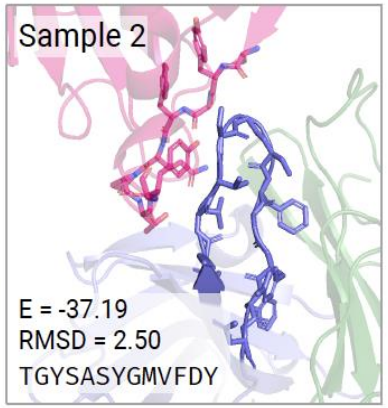
Result: Generate new antibodies

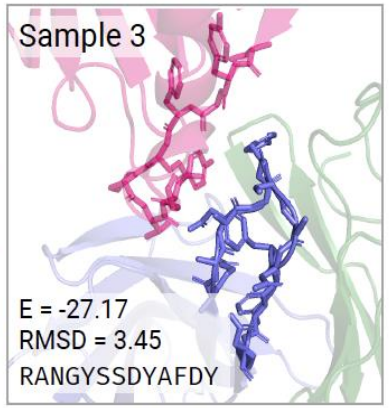










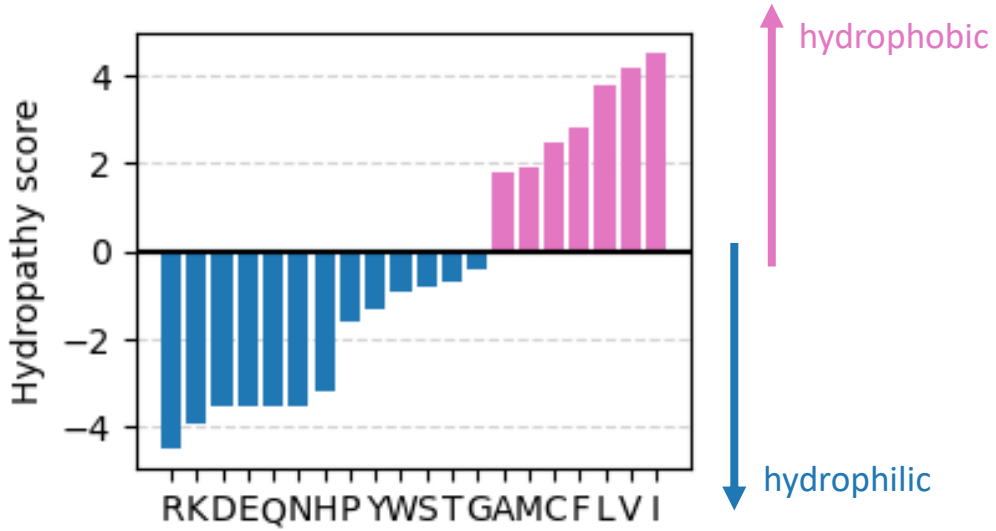


S. Luo, *et al*. "Antigen-specific antibody design and optimization with diffusion-based generative models." In *NeurIPS* 2022

Developability properties

- **Which properties** are essential for antibody developability?
- How can we **calculate or predict** these properties?

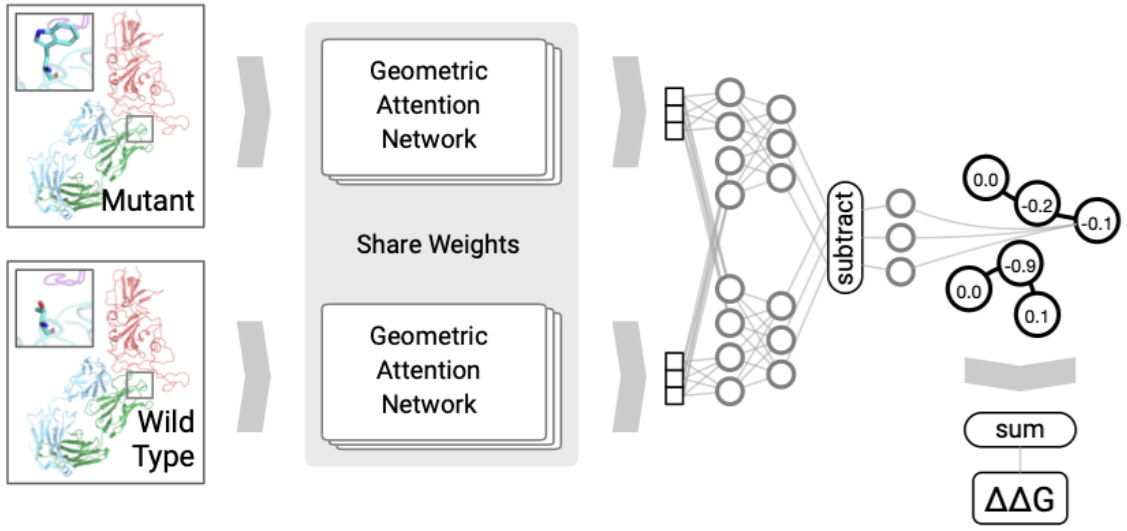
1. **Hydropathy score** (proxy for solubility)



J. Kyte and R.F. Doolittle. "A simple method for displaying the hydropathic character of a protein." *Journal of Molecular Biology*, 1982.

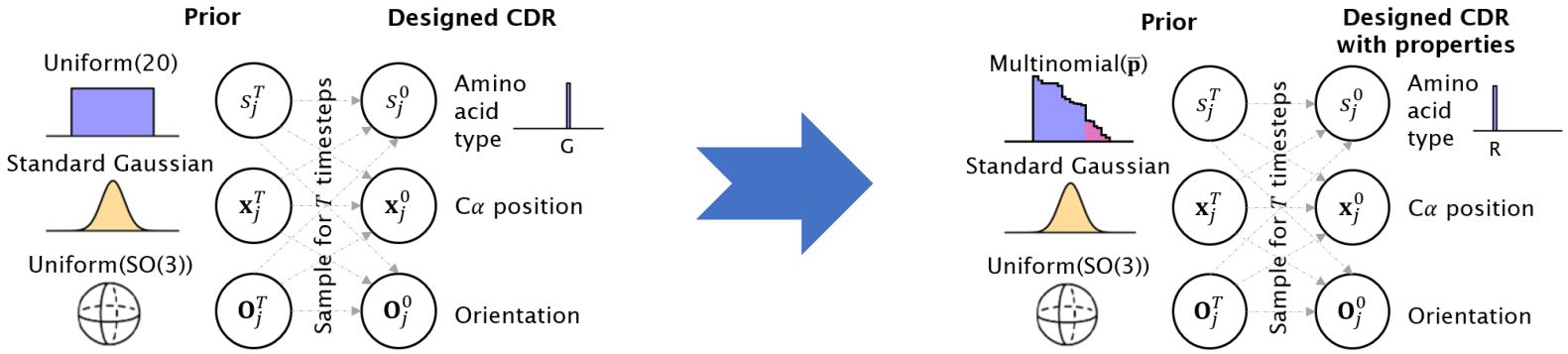
2. **Folding energy** (ΔG)

Predictor of changes in binding energy upon mutation (ΔΔG) for protein-protein complexes



S. Shan, *et al*. "Deep learning guided optimization of human antibody against SARS-CoV-2 variants with broad neutralization." *Proceedings of the National Academy of Sciences*, 2022.

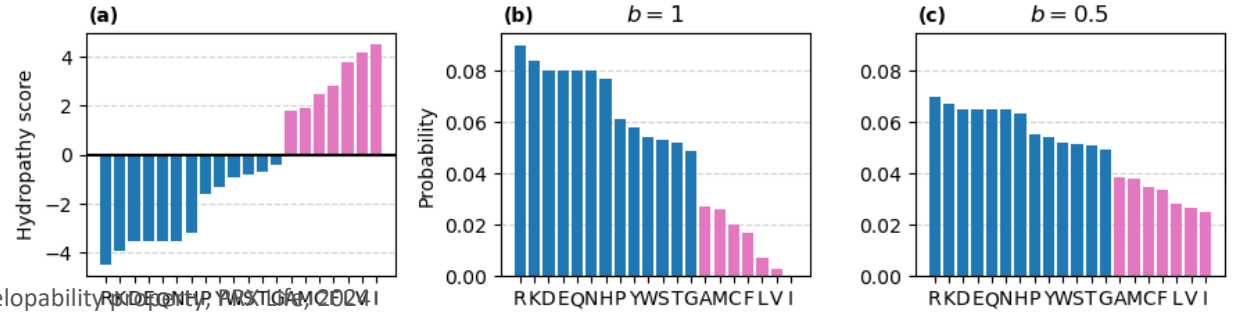
Proposed guidance methods: **Property-aware prior**



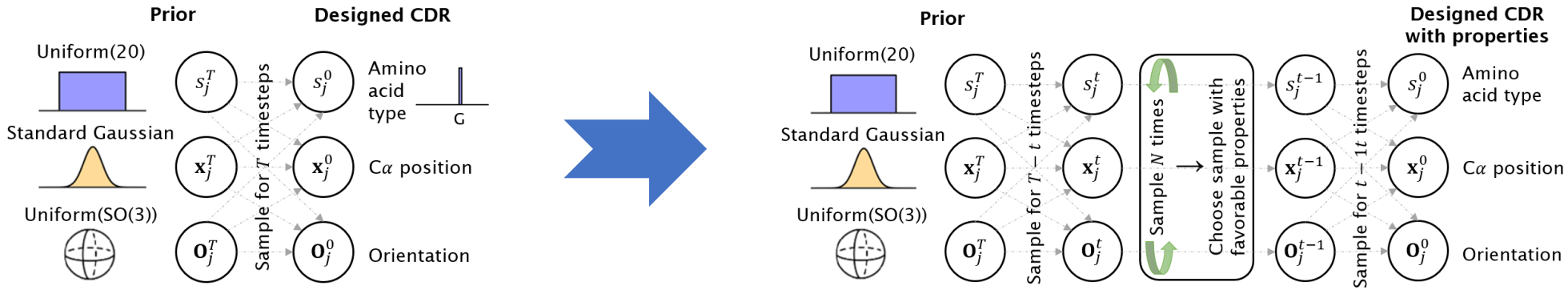
- Sample from informative prior:

$$s_j^T \sim \text{Multinomial}(\bar{\mathbf{p}}) = (1-b)\cdot \text{Uniform}(20) + b \cdot \text{Multinomial}(\mathbf{p})$$

- Probabilities $\mathbf{p} = [p_1, \ldots, p_{20}]$ for **hydropathy**



Proposed guidance methods: **Sampling by property**

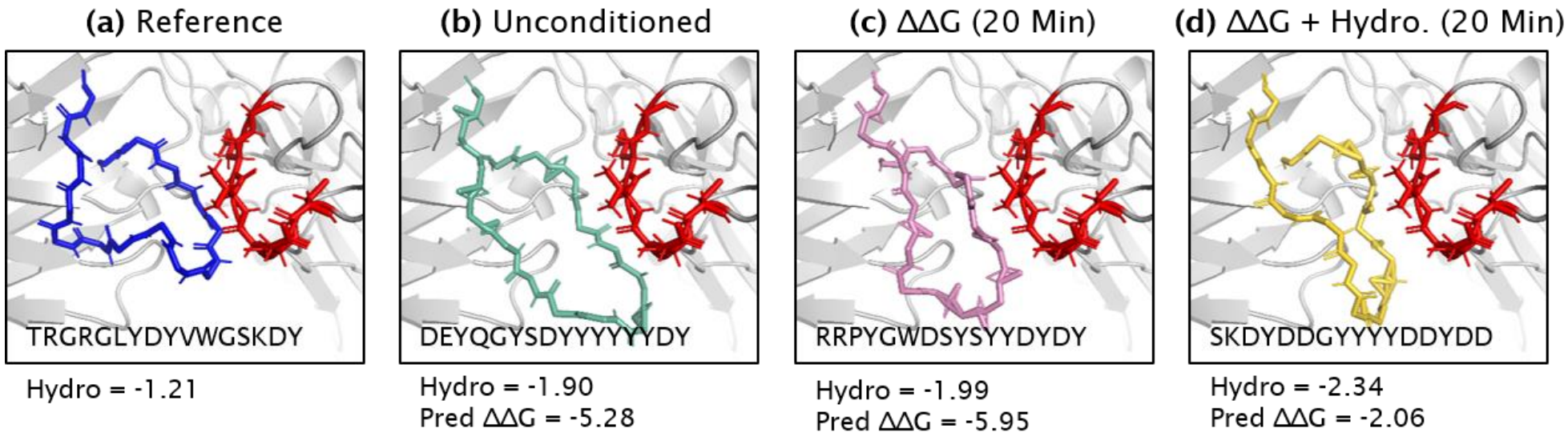


- At each **generation timestep**:
  1. Sample N times
  2. Select the sample with minimum property value

- With multiple properties, **select Pareto optimal** (minimum of the sum)
- Properties: **hydropathy score and predicted ΔΔG**

Pareto optimal solutions

- Designs in the Pareto frontier
- Different CDR sequences lead to **similar structures** compared to the reference with **improved hydropathy and predicted ΔΔG**



Villegas-Morcillo, Weber, Reinders, Guiding diffusion models for antibody sequence and structure co-design with developability property, PRX Life, 2024

END