北京航空航天大學
BEIHANG UNIVERSITY

AI for Science导论

人工智能研究院

主　　讲　沙磊

AI4Science

AI for Quantum Mechanics

AI for Density Functional Theory

AI for Small Molecules

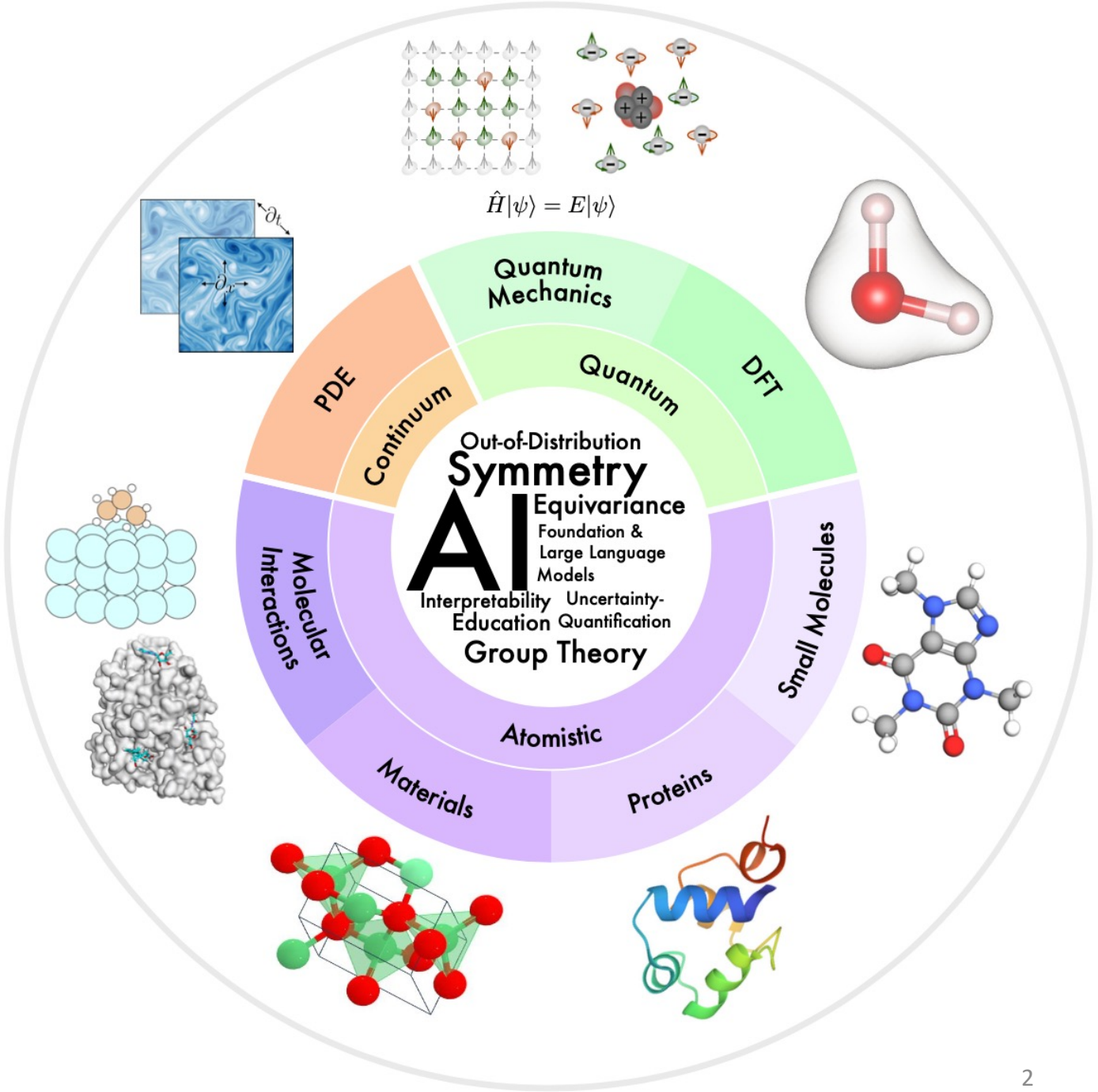
AI for Protein Science

AI for Materials Science

AI for Molecular Interactions

AI for Partial Differential Equations

……



Contents

- 物理
  - Neural ODE
  - 液态神经网络
  - DeepONet
- 生物
  - AI 分子生物概述
  - AI 药物发现
  - AI 解蛋白质
- 总结

Neural ODE

Ordinary Differential Equations (ODEs)

1st order Ordinary Differential Equation:

$$\frac{dx(t)}{dt} = f(x(t), t, \theta)$$

x is a variable we are interested in,

t is (typically) time,

f is a function of x and t, it is the differential,

θ parameterizes f (optionally).

Ordinary Differential Equations (ODEs)

Initial value problem:

$$\frac{dx(t)}{dt} = f(x(t), t, \theta); \quad x(t_0) \text{ is given}; \quad x(t_1) = \ ?$$

Many physical processes follow this template!

Ordinary Differential Equations(ODEs)

Initial value problem:

$$\frac{dx(t)}{dt} = f(x(t), t, \theta); \quad x(t_0) \text{ is given}; \quad x(t_1) = \ ?$$

Solution:

$$x(t_1) = x(t_0) + \int_{t_0}^{t_1} f(x(t), t, \theta)\, dt$$

Example:

$$\frac{dx}{dt} = 2t; \ x(0) = 2; \ x(1) = \ ?$$

$$\begin{aligned} \Rightarrow x(1) &= x(0) + \int_0^1 2t\, dt \\ &= x(0) + (t^2|_{t=1} - t^2|_{t=0}) \\ &= 2 + 1^2 - 0^2 \\ &= 3 \end{aligned}$$

Ordinary Differential Equations(ODEs)

Initial value problem:

$$\frac{dx(t)}{dt} = f(x(t), t, \theta); \quad x(t_0) \text{ is given}; \quad x(t_1) = \ ?$$

Solution:

$$x(t_1) = x(t_0) + \int_{t_0}^{t_1} f(x(t), t, \theta)\, dt$$

What if this cannot be analytically integrated?

Example:

$$\frac{dx}{dt} = 2xt \ ; \ x(0) = 3$$

$$\Rightarrow \int \frac{1}{2x}\, dx = \int t\, dt$$

$$\Rightarrow \frac{1}{2}\log x = \frac{1}{2}t^2 + c_0$$

$$\Rightarrow x(t) = ce^{t^2}$$

$$x(0) = 3 \Rightarrow c = 2$$

$$\therefore x(t) = 2e^{t^2}$$

$$\Rightarrow x(1) = 5.436$$

Ordinary Differential Equations(ODEs)

Initial value problem:

$$\frac{dx(t)}{dt} = f(x(t), t, \theta); \quad x(t_0) \text{ is given}; \quad x(t_1) = \ ?$$

Solution:

$$x(t_1) = x(t_0) + \int_{t_0}^{t_1} f(x(t), t, \theta)\, dt$$

Approximations to $\int_{t_0}^{t_1} f(x(t), t, \theta)\, dt$

i.e. **Numerical Integration:**

- Euler method
- Runge-Kutta methods
- ...

Ordinary Differential Equations(ODEs)

Initial value problem:

$$\frac{dx(t)}{dt} = f(x(t), t, \theta); \quad x(t_0) \text{ is given}; \quad x(t_1) = \ ?$$

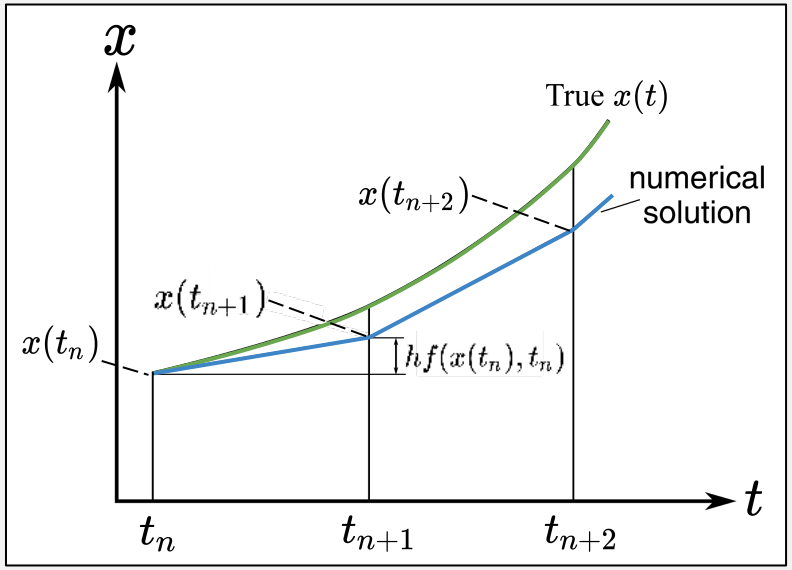
Solution:

$$x(t_1) = x(t_0) + \int_{t_0}^{t_1} f(x(t), t, \theta)\, dt$$

1st-order Runge-Kutta / Euler's method:

$$t_{n+1} = t_n + h \longrightarrow \text{Step size } h$$

$$x(t_{n+1}) = x(t_n) + hf(x(t_n), t_n) \dashrightarrow \text{Update using derivative } f$$



https://guide.freecodecamp.org/mathematics/differential-equations/eulers-method/

Ordinary Differential Equations(ODEs)

Initial value problem:

$$\frac{dx(t)}{dt} = f(x(t), t, \theta); \quad x(t_0) \text{ is given}; \quad x(t_1) = \ ?$$

Solution:

$$x(t_1) = x(t_0) + \int_{t_0}^{t_1} f(x(t), t, \theta)\, dt$$

1st-order Runge-Kutta / Euler's method:

$$t_{n+1} = t_n + h$$
$$x(t_{n+1}) = x(t_n) + hf(x(t_n), t_n)$$

Example:

$\frac{dx}{dt} = f(x,t) = 2xt\ ;\ x(0) = 3;\ x(1) = \ ?$

(Solution: $x(t) = 2e^{t^2}\ ;\ x(1) = 5.436$)

$h = 0.25$

$$\begin{aligned}
x(0.25) &= x(0) + 0.25 * f(x(0), 0) \\
&= 3 + 0.25 * (2 * 3 * 0) \\
&= 3 \\
x(0.5) &= x(0.25) + 0.25 * f(x(0.25), 0.25) \\
&= 3 + 0.25 * (2 * 3 * 0.25) \\
&= 3.375 \\
x(0.75) &= x(0.5) + 0.25 * f(x(0.5), 0.5) \\
&= 3.375 + 0.25 * (2 * 3.375 * 0.5) \\
&= 4.21875 \\
x(1) &= x(0.75) + 0.25 * f(x(0.75), 0.75) \\
&= 4.21875 + 0.25 * (2 * 4.21875 * 0.75) \\
&= 5.8008
\end{aligned}$$

Ordinary Differential Equations(ODEs)

Initial value problem:

$$\frac{dx(t)}{dt} = f(x(t), t, \theta); \quad x(t_0) \text{ is given}; \quad x(t_1) = \ ?$$

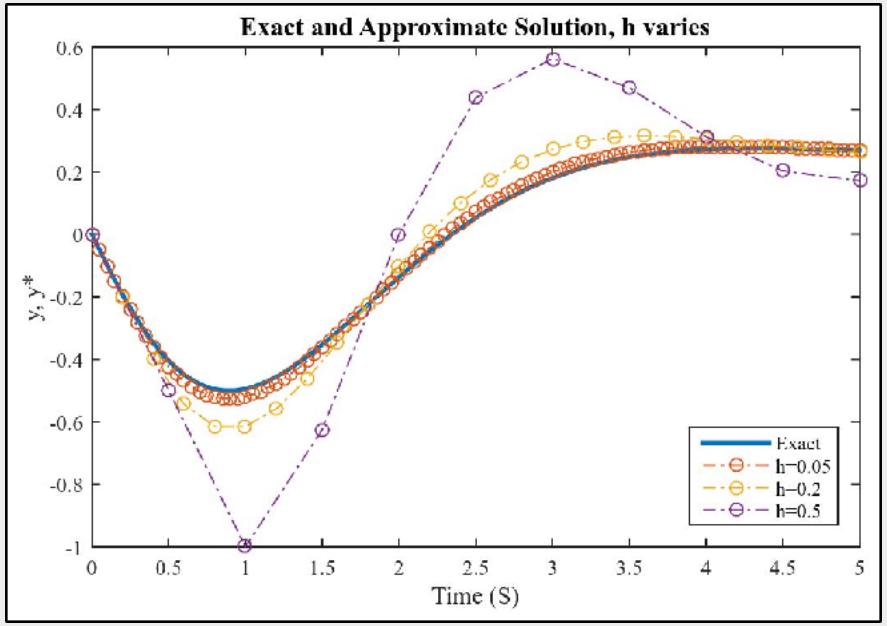
Solution:

$$x(t_1) = x(t_0) + \int_{t_0}^{t_1} f(x(t), t, \theta)\, dt$$

1st-order Runge-Kutta / Euler's method:

$$t_{n+1} = t_n + h$$
$$x(t_{n+1}) = x(t_n) + h f(x(t_n), t_n)$$

Step size matters!



https://lpsa.swarthmore.edu/NumInt/NumIntFirst.html

Ordinary Differential Equations(ODEs)

Initial value problem:

$$\frac{dx(t)}{dt} = f(x(t), t, \theta); \quad x(t_0) \text{ is given}; \quad x(t_1) = \ ?$$

Solution:

$$x(t_1) = x(t_0) + \int_{t_0}^{t_1} f(x(t), t, \theta)\, dt$$

★ **4th-order Runge-Kutta method:**

$$t_{n+1} = t_n + h$$
$$s_1 = f(x(t_n),\ t_n)$$
$$s_2 = f(x(t_n) + \tfrac{h}{2} s_1,\ t_n + \tfrac{h}{2})$$
$$s_3 = f(x(t_n) + \tfrac{h}{2} s_2,\ t_n + \tfrac{h}{2})$$
$$s_4 = f(x(t_n) + h s_3,\ t_n + h)$$
$$x(t_{n+1}) = x(t_n) + \tfrac{h}{6}(s_1 + 2s_2 + 2s_3 + s_4)$$

→ Default ODE solver used in MATLAB: https://blogs.mathworks.com/loren/2015/09/23/ode-solver-selection-in-matlab/

Ordinary Differential Equations(ODEs)

Initial value problem:

$$\frac{dx(t)}{dt} = f(x(t), t, \theta); \quad x(t_0) \text{ is given}; \quad x(t_1) = \ ?$$

Solution:

$$x(t_1) = x(t_0) + \int_{t_0}^{t_1} f(x(t), t, \theta)\, dt$$

$$x(t_1) = \text{ODESolve}(\ f(x(t), t, \theta),\ x(t_0),\ t_0,\ t_1\ )$$

- t_1: Final time
- t_0: Initial time
- $x(t_0)$: Initial value
- $f(x(t), t, \theta)$: Differential
- ODESolve: Any ODE solver of our choice!

Ordinary Differential Equations(ODEs)

Initial value problem:

$$\frac{dx(t)}{dt} = f(x(t), t, \theta); \quad x(t_0) \text{ is given}; \quad x(t_1) = \ ?$$

Solution:

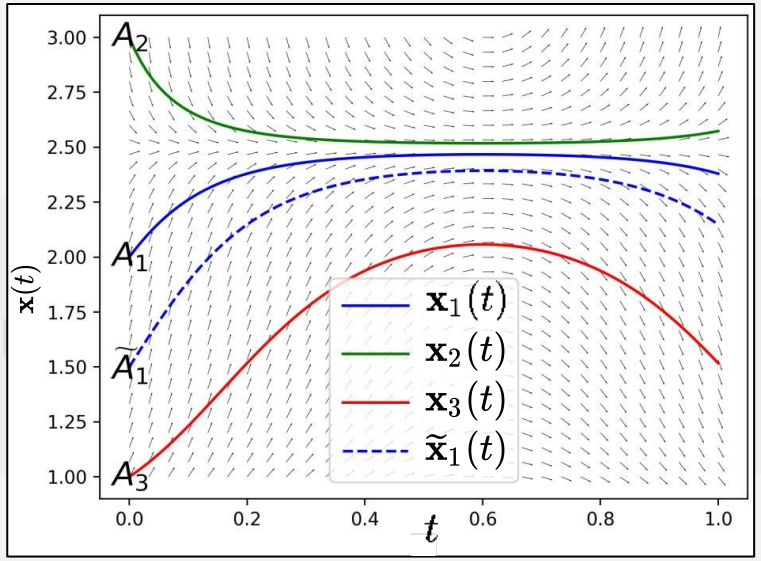
$$x(t_1) = \text{ODESolve}(\ f(x(t), t, \theta),\ x(t_0),\ t_0,\ t_1\ )$$

Fundamental Theorem of ODEs

Suppose f is continuously differentiable.

1. The solution to a differential equation completely fills the space.
2. Solution curves for different solutions **do not intersect**.

Geometrically, $x(t)$ is a flow!



http://faculty.bard.edu/belk/math213/InitialValueProblems.pdf

https://openreview.net/pdf?id=B1e9Y2NYvS

Neural ODEs (Chen et al., NeurIPS 2018)

Initial value problem:

$$\frac{dx(t)}{dt} = f(x(t), t, \theta); \quad x(t_0) \text{ is given}; \quad x(t_1) = \ ?$$

Solution:

$$x(t_1) = \text{ODESolve}(\ f(x(t), t, \theta),\ x(t_0),\ t_0,\ t_1\ )$$

f is a neural network!

Paradigm shift: whereas earlier *f* was pre-defined/hand-designed according to the domain, here we would like to estimate an *f* that suits our objective.

https://arxiv.org/pdf/1806.07366.pdf

Neural ODEs (Chen et al., 2018)

ODEs

$$\frac{d\mathbf{x}(t)}{dt} = f(\mathbf{x}(t), t, \theta)$$

Euler discretization

Vector notation

$$\mathbf{x}_{n+1} = \mathbf{x}_n + h\, f(\mathbf{x}_n, t_n, \theta)$$

https://arxiv.org/pdf/1806.07366.pdf

Residual networks

$$\mathbf{x}_{l+1} = \mathrm{ResBlock}(\mathbf{x}_l, \theta)$$

$$\mathbf{x}_{l+1} = \mathbf{x}_l + g(\mathbf{x}_l, \theta)$$

Skip connection

https://arxiv.org/pdf/1512.03385.pdf

Neural ODEs (Chen et al., 2018)

ODEs

$$\frac{d\mathbf{x}(t)}{dt} = f(\mathbf{x}(t), t, \theta)$$

Euler discretization

$$\mathbf{x}_{n+1} = \mathbf{x}_n + h\, f(\mathbf{x}_n, t_n, \theta)$$

Forward propagation:

$$\mathbf{x}(t_1) = \text{ODESolve}(\, f(\mathbf{x}(t), t, \theta),\ \mathbf{x}(t_0),\ t_0,\ t_1\,)$$

$$L(\mathbf{x}(t_1)) \rightarrow \frac{\partial L}{\partial \theta}$$

How to compute this?

Update θ to reduce L

https://arxiv.org/pdf/1806.07366.pdf

Residual networks

$$\mathbf{x}_{l+1} = \text{ResBlock}(\mathbf{x}_l, \theta)$$

$$\mathbf{x}_{l+1} = \mathbf{x}_l + g(\mathbf{x}_l, \theta)$$

Skip connection

$$\mathbf{y}_{pred} = \text{ResNet}(\mathbf{x})$$

Stacked ResBlocks

$$L(\mathbf{y}_{pred}) \rightarrow \frac{\partial L}{\partial \theta}$$

Update θ to reduce L

https://arxiv.org/pdf/1512.03385.pdf

Neural ODEs (Chen et al., 2018)

ODEs

$$\frac{d\mathbf{x}(t)}{dt} = f(\mathbf{x}(t), t, \theta)$$

Euler discretization

$$\mathbf{x}_{n+1} = \mathbf{x}_n + h\, f(\mathbf{x}_n, t_n, \theta)$$

Forward propagation:

$$\mathbf{x}(t_1) = \text{ODESolve}(\, f(\mathbf{x}(t), t, \theta),\ \mathbf{x}(t_0),\ t_0,\ t_1\,)$$

$$L(\mathbf{x}(t_1)) \rightarrow \boxed{\frac{\partial L}{\partial \theta}}$$

Update θ to reduce L

Back-propagate through the ODE Solver!

High memory cost-

need to save all activations of all iterations of ODESolve.

Can we do better?

Yes.

https://arxiv.org/pdf/1806.07366.pdf

Neural ODEs (Chen et al., 2018)

$$L(\text{ODESolve}(\ f(\mathbf{x}(t), t, \theta),\ \mathbf{x}(t_0),\ t_0,\ t_1\ )) \rightarrow \frac{\partial L}{\partial \theta}$$

Adjoint method (Pontryagin et al., 1962)

$$\text{adjoint } \mathbf{a}(t) = \frac{\partial L}{\partial \mathbf{x}}\ ;\ \frac{d\mathbf{a}}{dt} = -\mathbf{a}(t)^\top \frac{\partial f(\mathbf{x}(t), t, \theta)}{\partial \mathbf{x}}$$

Chain rule:

$$\frac{\partial L}{\partial \mathbf{x}(t)} = \frac{\partial L}{\partial \mathbf{x}(t+\epsilon)} \frac{\partial \mathbf{x}(t+\epsilon)}{\partial \mathbf{x}(t)}$$
$$= a(t+\epsilon) \frac{\partial \mathbf{x}(t+\epsilon)}{\partial \mathbf{x}(t)}$$

Since:

$$\mathbf{x}(t+\epsilon) = \mathbf{x}(t) + \int_t^{t+\epsilon} f(\mathbf{x}(t'), t', \theta) dt'$$

$$\mathbf{a}(\mathbf{t}) = \mathbf{a}(t+\epsilon) \frac{\partial}{\partial \mathbf{x}(t)} \left(\mathbf{x}(t) + \int_t^{t+\epsilon} f(\mathbf{x}(t'), t', \theta) dt' \right)$$
$$= \mathbf{a}(t+\epsilon) + \mathbf{a}(t+\epsilon) \frac{\partial}{\partial \mathbf{x}(t)} \left(\int_t^{t+\epsilon} f(\mathbf{x}(t'), t', \theta) dt' \right)$$

$$\frac{d\mathbf{a}(t)}{dt} = \lim_{\epsilon \rightarrow 0^+} \frac{\mathbf{a}(t+\epsilon) - \mathbf{a}(t)}{\epsilon}$$
$$= -\mathbf{a}(t)^\top \frac{\partial f(\mathbf{x}(t), t, \theta)}{\partial \mathbf{x}(t)}$$

Neural ODEs (Chen et al., 2018)

$$L(\text{ODESolve}(\ f(\mathbf{x}(t), t, \theta),\ \mathbf{x}(t_0),\ t_0,\ t_1\ )) \rightarrow \frac{\partial L}{\partial \theta}$$

Adjoint method (Pontryagin et al., 1962)

$$\text{adjoint } \mathbf{a}(t) = \frac{\partial L}{\partial \mathbf{x}}\ ;\ \frac{d\mathbf{a}}{dt} = -\mathbf{a}(t)^\top \frac{\partial f(\mathbf{x}(t), t, \theta)}{\partial \mathbf{x}}$$

Forward propagation: $\mathbf{x}(t_1) = \text{ODESolve}(\ f(\mathbf{x}(t), t, \theta),\ \mathbf{x}(t_0),\ t_0,\ t_1\ ) \Rightarrow \mathbf{a}(t_1) = \frac{\partial L}{\partial \mathbf{x}(t_1)}$

$$\boxed{\frac{\partial L}{\partial \theta}} = \int_{t_1}^{t_0} -\mathbf{a}(t)^\top \frac{\partial f(\mathbf{x}(t), t, \theta)}{\partial \theta}\, dt$$

?

?

https://arxiv.org/pdf/1806.07366.pdf

Neural ODEs (Chen et al., 2018)

$$L(\text{ODESolve}(\ f(\mathbf{x}(t), t, \theta),\ \mathbf{x}(t_0),\ t_0,\ t_1\ )) \rightarrow \frac{\partial L}{\partial \theta}$$

Adjoint method (Pontryagin et al., 1962)

$$\text{adjoint } \mathbf{a}(t) = \frac{\partial L}{\partial \mathbf{x}}\ ;\ \frac{d\mathbf{a}}{dt} = -\mathbf{a}(t)^\top \frac{\partial f(\mathbf{x}(t), t, \theta)}{\partial \mathbf{x}}$$

Forward propagation: $\mathbf{x}(t_1) = \text{ODESolve}(\ f(\mathbf{x}(t), t, \theta),\ \mathbf{x}(t_0),\ t_0,\ t_1\ ) \Rightarrow \mathbf{a}(t_1) = \frac{\partial L}{\partial \mathbf{x}(t_1)}$

Back-propagation:

$$x(t_0) = \text{ODESolve}(\quad f(\mathbf{x}(t), t, \theta)\quad ,\ \mathbf{x}(t_1),\ t_1,\ t_0)$$

$$\Rightarrow \mathbf{a}(t_0) = \frac{\partial L}{\partial \mathbf{x}(t_0)} = \text{ODESolve}(-\mathbf{a}(t)^\top \frac{\partial f(\mathbf{x}(t), t, \theta)}{\partial \mathbf{x}},\ \frac{\partial L}{\partial \mathbf{x}(t_1)},\ t_1,\ t_0)$$

$$\therefore \boxed{\frac{\partial L}{\partial \theta}} = \int_{t_1}^{t_0} -\mathbf{a}(t)^\top \frac{\partial f(\mathbf{x}(t), t, \theta)}{\partial \theta}\, dt = \text{ODESolve}(-\mathbf{a}(t)^\top \frac{\partial f(\mathbf{x}(t), t, \theta)}{\partial \theta},\ \mathbf{0}_{|\theta|}\ ,\ t_1,\ t_0)$$

Initial value is0

https://arxiv.org/pdf/1806.07366.pdf

Neural ODEs (Chen et al., 2018)

Forward propagation:

$$\mathbf{x}(t_1) = \text{ODESolve}(\, f(\mathbf{x}(t), t, \theta),\ \mathbf{x}(t_0),\ t_0,\ t_1\ )$$

Compute $L(\mathbf{x}(t_1))$.

$$\mathbf{a}(t_1) = \frac{\partial L}{\partial \mathbf{x}(t_1)}$$

Back-propagation:

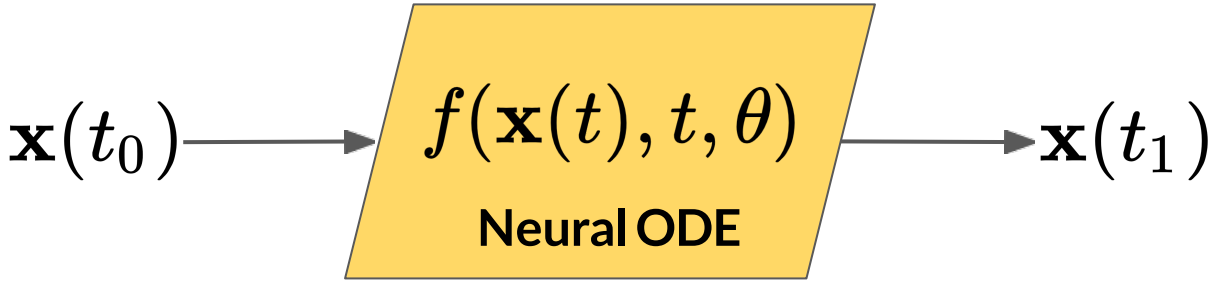
$$\begin{bmatrix} \mathbf{x}(t_0) \\ \frac{\partial L}{\partial \mathbf{x}(t_0)} \\ \boxed{\frac{\partial L}{\partial \theta}} \end{bmatrix} = \text{ODESolve}\left(\begin{bmatrix} f(\mathbf{x}(t), t, \theta) \\ -\mathbf{a}(t)^\top \frac{\partial f(\mathbf{x}(t), t, \theta)}{\partial \mathbf{x}} \\ -\mathbf{a}(t)^\top \frac{\partial f(\mathbf{x}(t), t, \theta)}{\partial \theta} \end{bmatrix}, \begin{bmatrix} \mathbf{x}(t_1) \\ \frac{\partial L}{\partial \mathbf{x}(t_1)} \\ \mathbf{0}_{|\theta|} \end{bmatrix}, t_1, t_0 \right)$$

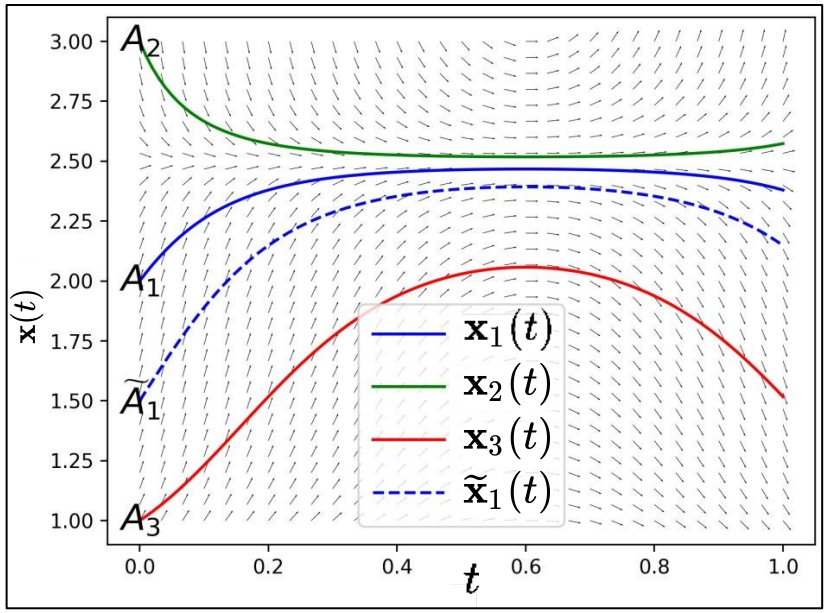
Update θ to reduce L

https://arxiv.org/pdf/1806.07366.pdf

Neural ODEs (Chen et al., 2018)

https://arxiv.org/pdf/1806.07366.pdf





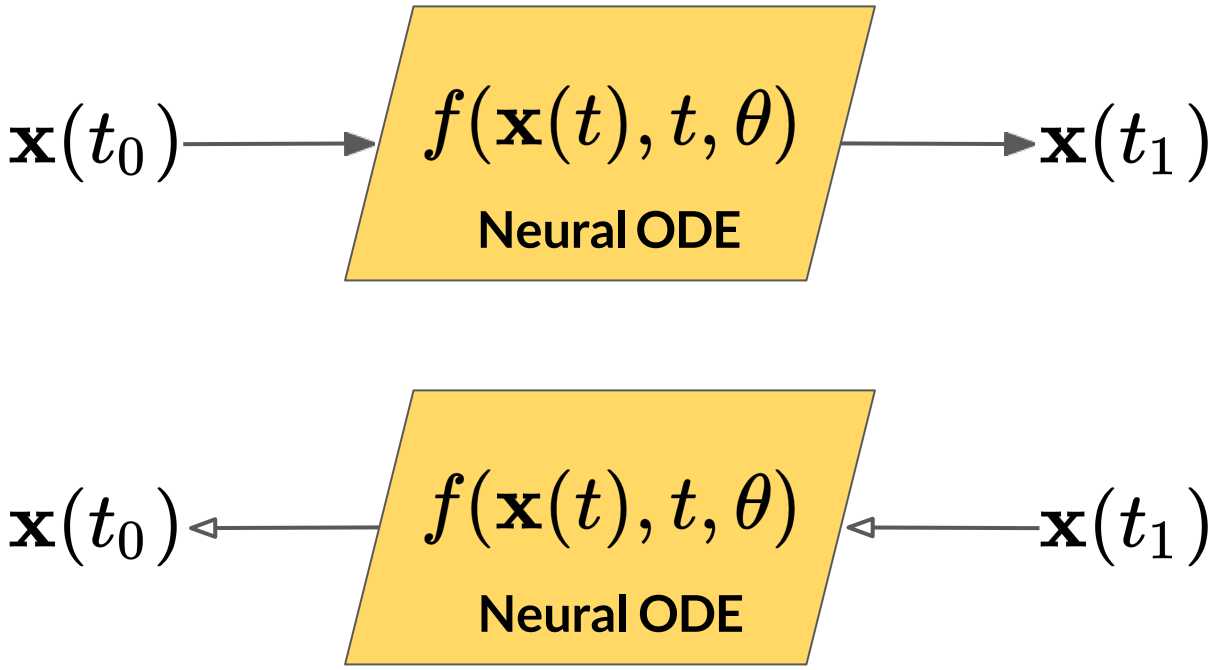
https://openreview.net/pdf?id=B1e9Y2NYvS

Neural ODEs describe a homeomorphism (flow).

- They preserve dimensionality.
- They form non-intersecting trajectories.

Neural ODEs (Chen et al., 2018)

https://arxiv.org/pdf/1806.07366.pdf



Neural ODEs are **reversible** models!
Just integrate forward/backward in time.

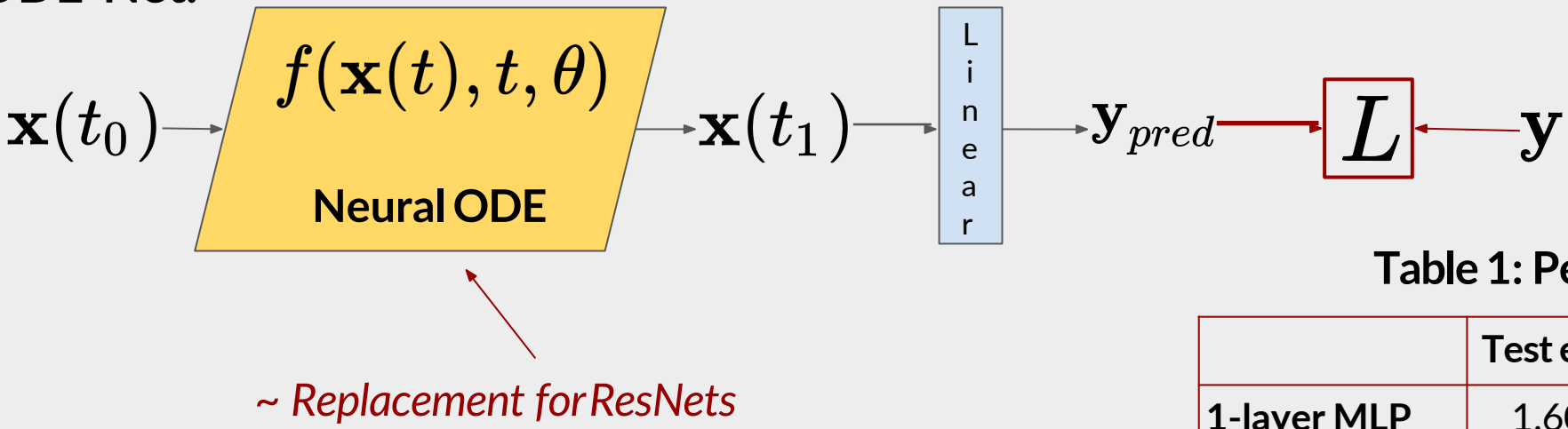
Neural ODEs (Chen et al., 2018)

https://arxiv.org/pdf/1806.07366.pdf

Applications

Supervised Learning | Continuous Normalizing Flows | Generative Latent Models

ODE-Net:



~ Replacement for ResNets

Table 1: Performance on MNIST.

	Test error	# Params	Memory	Time
1-layer MLP	1.60%	0.24 M	-	-
ResNet	0.41%	0.60 M	$\mathcal{O}(L)$	$\mathcal{O}(L)$
RK-Net	0.47%	0.22 M	$\mathcal{O}(\tilde{L})$	$\mathcal{O}(\tilde{L})$
ODE-Net	0.42%	0.22 M	$\mathcal{O}(1)$	$\mathcal{O}(\tilde{L})$

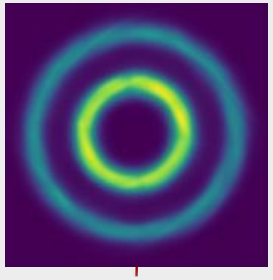
Neural ODEs (Chen et al., 2018)

https://arxiv.org/pdf/1806.07366.pdf

Applications

Supervised Learning | **Continuous Normalizing Flows** | Generative Latent Models

Target distribution

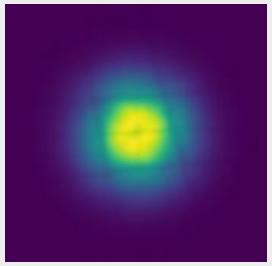


(such as real image manifold)

Sample from target distribution (such as an image)

$\mathbf{x}(t_0)$ ⟷ $f(\mathbf{x}(t), t, \theta)$ **Neural ODE** ⟷ $\mathbf{x}(t_1)$

Noise distribution



Sample from noise distribution (such as Gaussian)

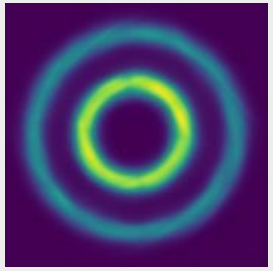
Neural ODEs (Chen et al., 2018)

https://arxiv.org/pdf/1806.07366.pdf

Applications

Supervised Learning | **Continuous Normalizing Flows** | Generative Latent Models

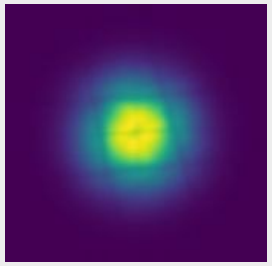
Target distribution



$\mathbf{x}(t_0)$ ⟷ $f(\mathbf{x}(t), t, \theta)$ ⟷ $\mathbf{x}(t_1)$

Neural ODE

Noise distribution



Likelihood estimation

using Change of Variables formula

$$\mathbf{x}_1 = g(\mathbf{x}_0) \Rightarrow \log p(\mathbf{x}_0) = \log p(\mathbf{x}_1) + \log \left| \det \frac{\partial g}{\partial \mathbf{x}_0} \right|$$

Train *f* to maximize the likelihood of the samples from target distribution $\log p(\mathbf{x_0})$

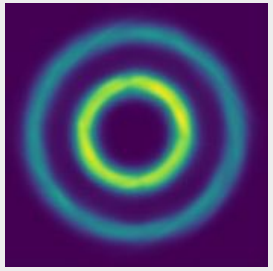
Neural ODEs (Chen et al., 2018)

https://arxiv.org/pdf/1806.07366.pdf

Applications

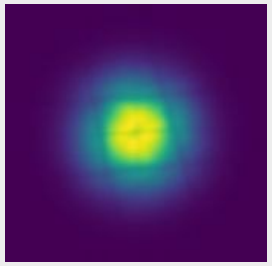
Supervised Learning | **Continuous Normalizing Flows** | Generative Latent Models

Target distribution



$\mathbf{x}(t_0)$ ⟷ $f(\mathbf{x}(t), t, \theta)$ **Neural ODE** ⟷ $\mathbf{x}(t_1)$

Noise distribution



Likelihood estimation

using Change of Variables formula

Generate samples

Sample from the noise distribution, transform it into a sample from the target distribution using the trained Neural ODE.

Neural ODEs (Chen et al., 2018)

https://arxiv.org/pdf/1806.07366.pdf

Applications

Supervised Learning | Continuous Normalizing Flows | **Generative Latent Models**

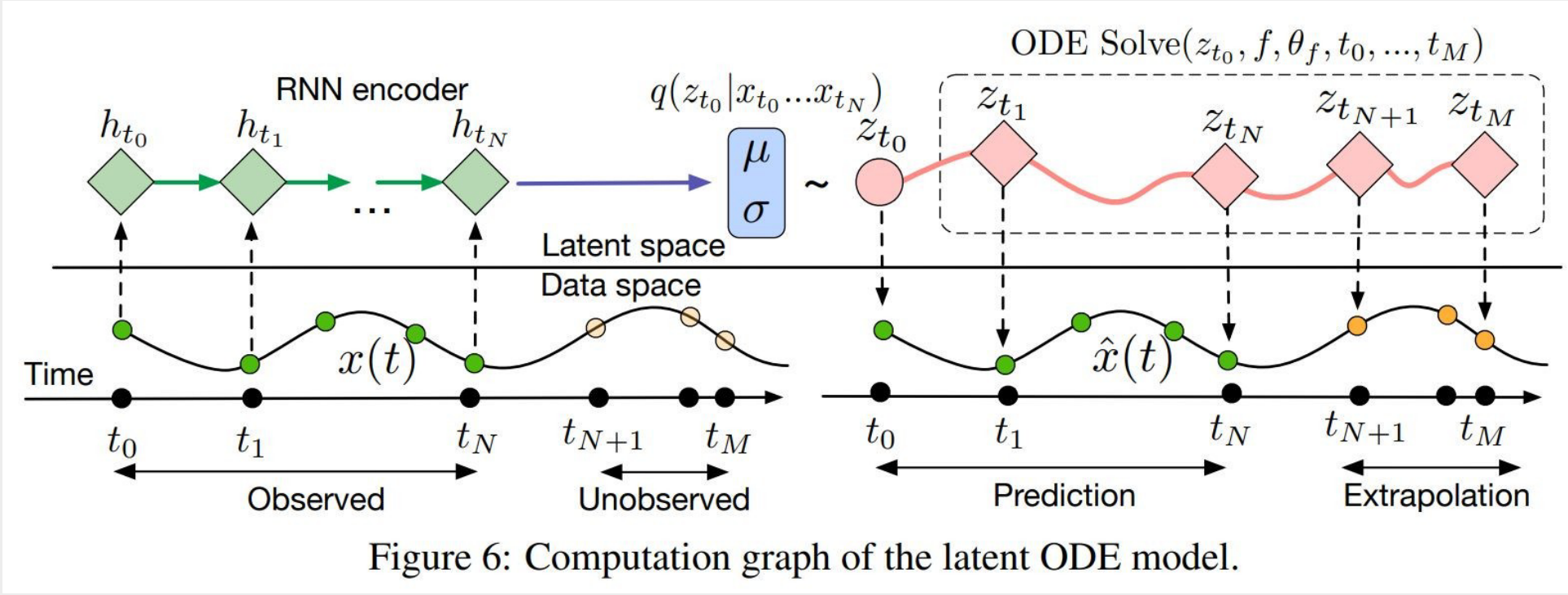


Figure 6: Computation graph of the latent ODE model.

Neural ODEs (Chen et al., 2018)

https://arxiv.org/pdf/1806.07366.pdf

Applications

Supervised Learning | Continuous Normalizing Flows | **Generative Latent Models**

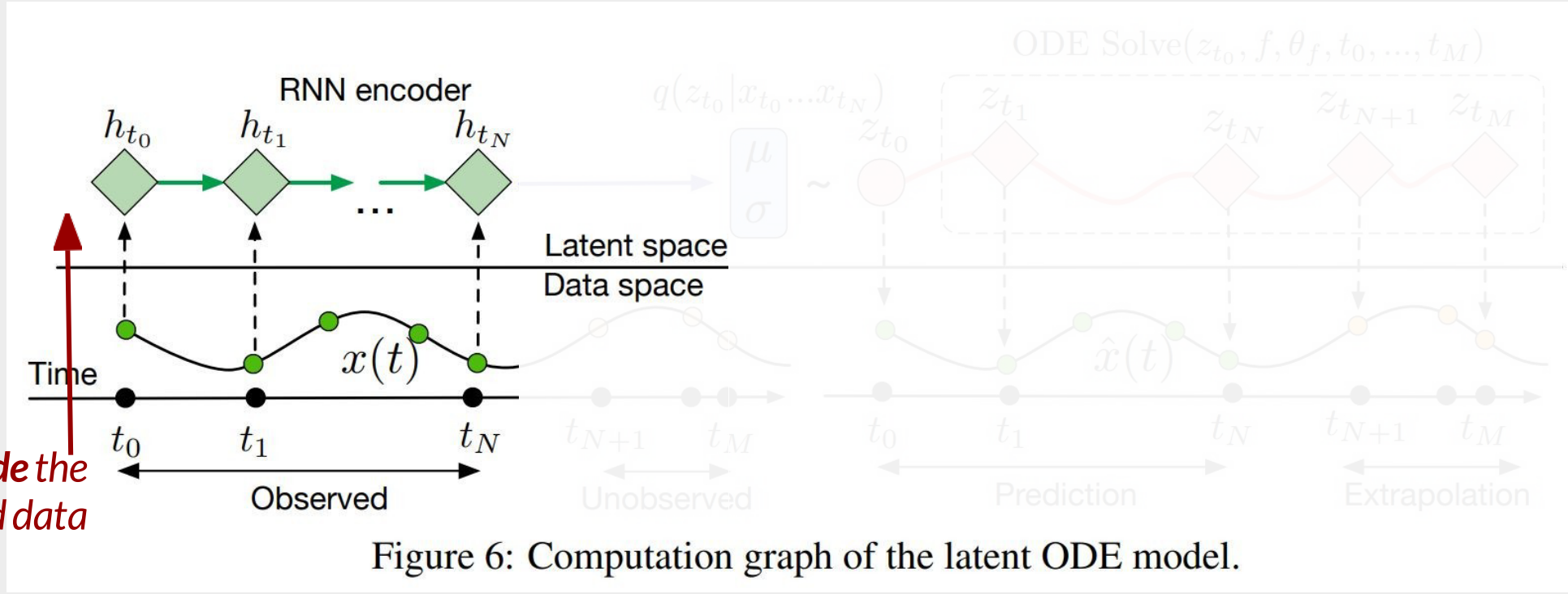


Figure 6: Computation graph of the latent ODE model.

Neural ODEs (Chen et al., 2018)

https://arxiv.org/pdf/1806.07366.pdf

Applications

Supervised Learning | Continuous Normalizing Flows | **Generative Latent Models**

Encode *into a latent distribution (such as Gaussian)*

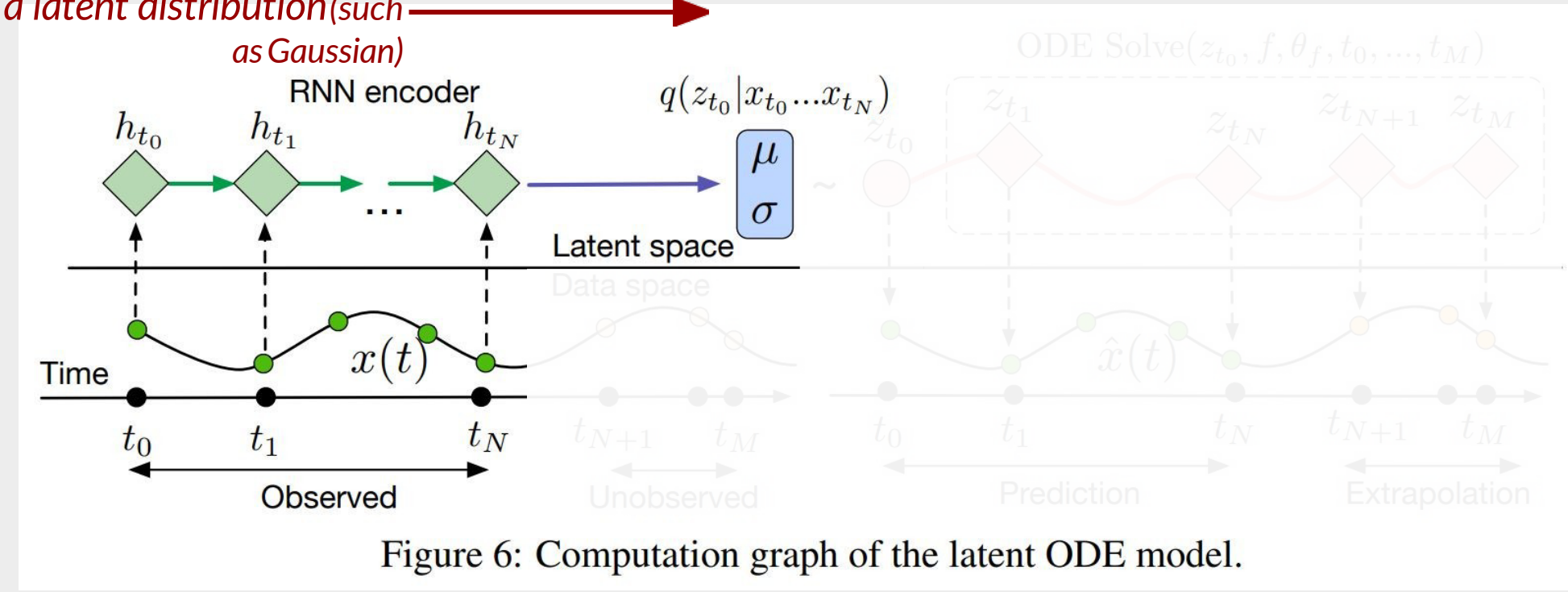


Figure 6: Computation graph of the latent ODE model.

Neural ODEs (Chen et al., 2018)

https://arxiv.org/pdf/1806.07366.pdf

Applications

Supervised Learning | Continuous Normalizing Flows | **Generative Latent Models**

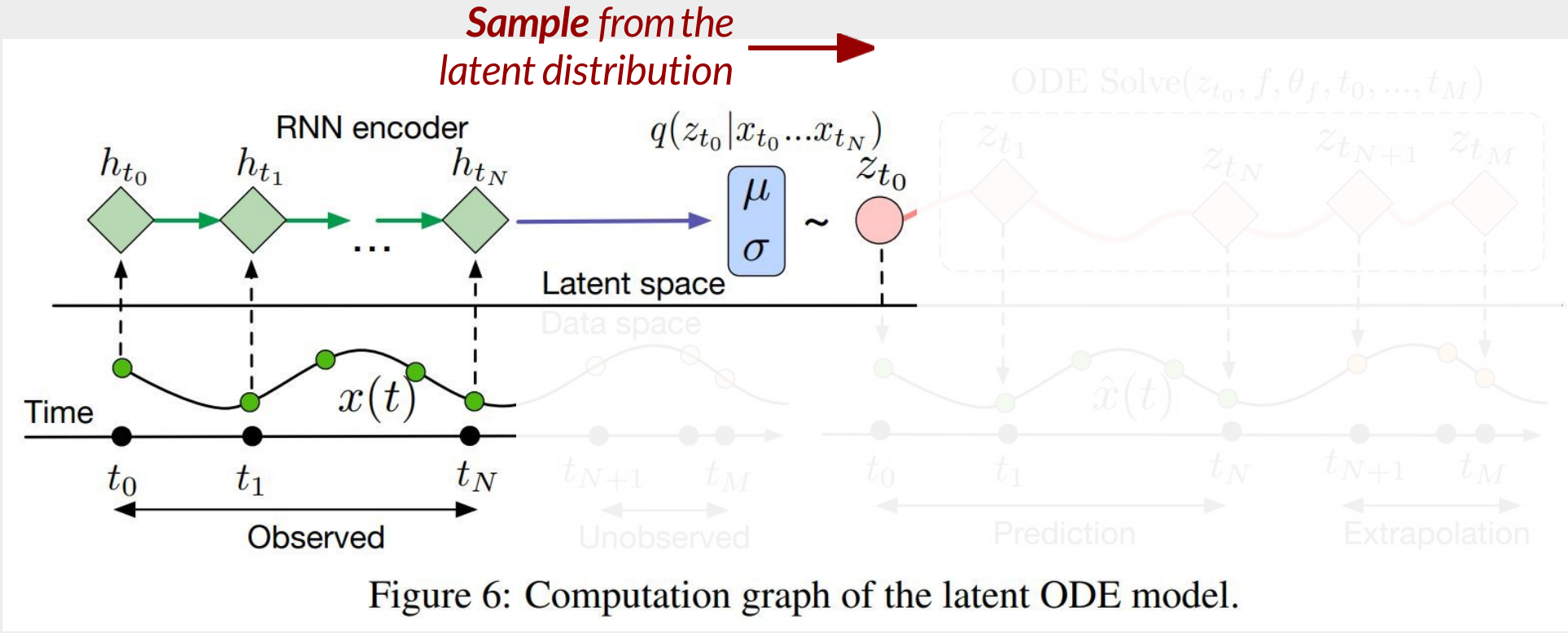


Figure 6: Computation graph of the latent ODE model.

Neural ODEs (Chen et al., 2018)

https://arxiv.org/pdf/1806.07366.pdf

Applications

Supervised Learning | Continuous Normalizing Flows | **Generative Latent Models**

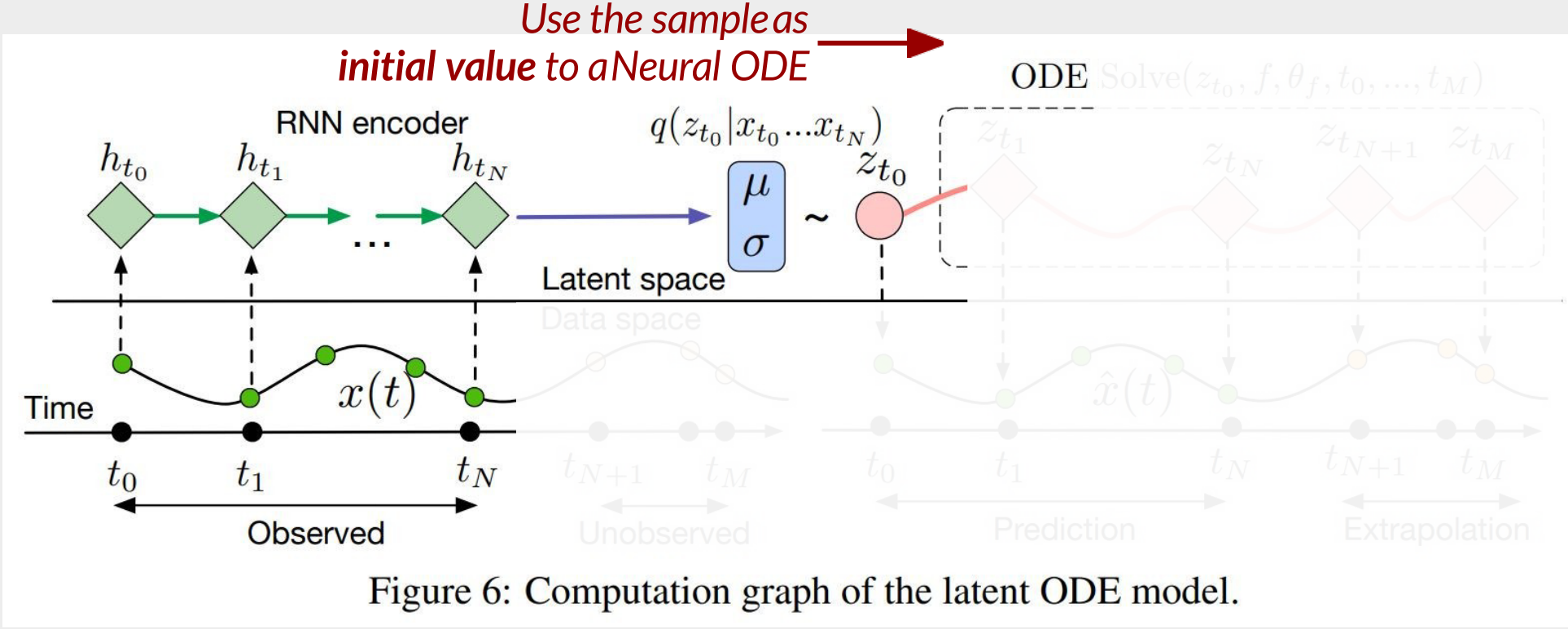


Figure 6: Computation graph of the latent ODE model.

Neural ODEs (Chen et al., 2018)

https://arxiv.org/pdf/1806.07366.pdf

Applications

Supervised Learning | Continuous Normalizing Flows | **Generative Latent Models**

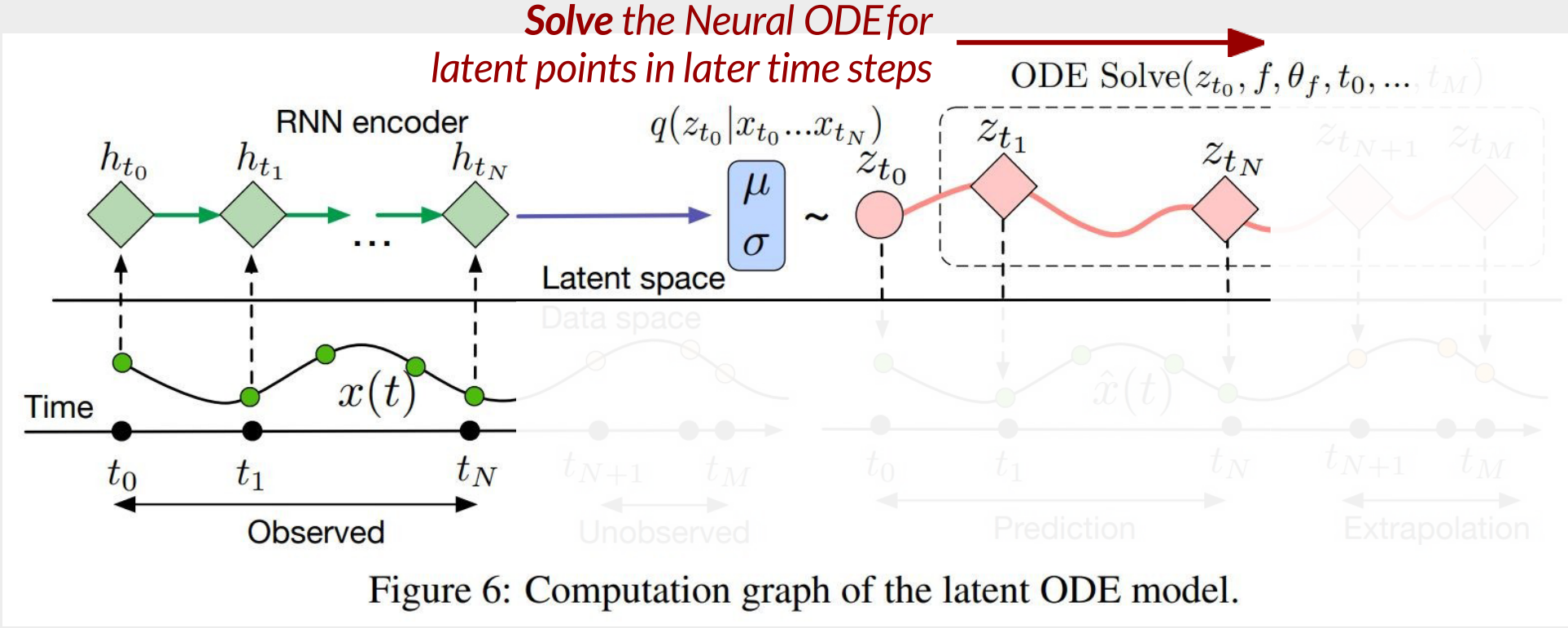


Figure 6: Computation graph of the latent ODE model.

Neural ODEs (Chen et al., 2018)

https://arxiv.org/pdf/1806.07366.pdf

Applications

Supervised Learning | Continuous Normalizing Flows | **Generative Latent Models**

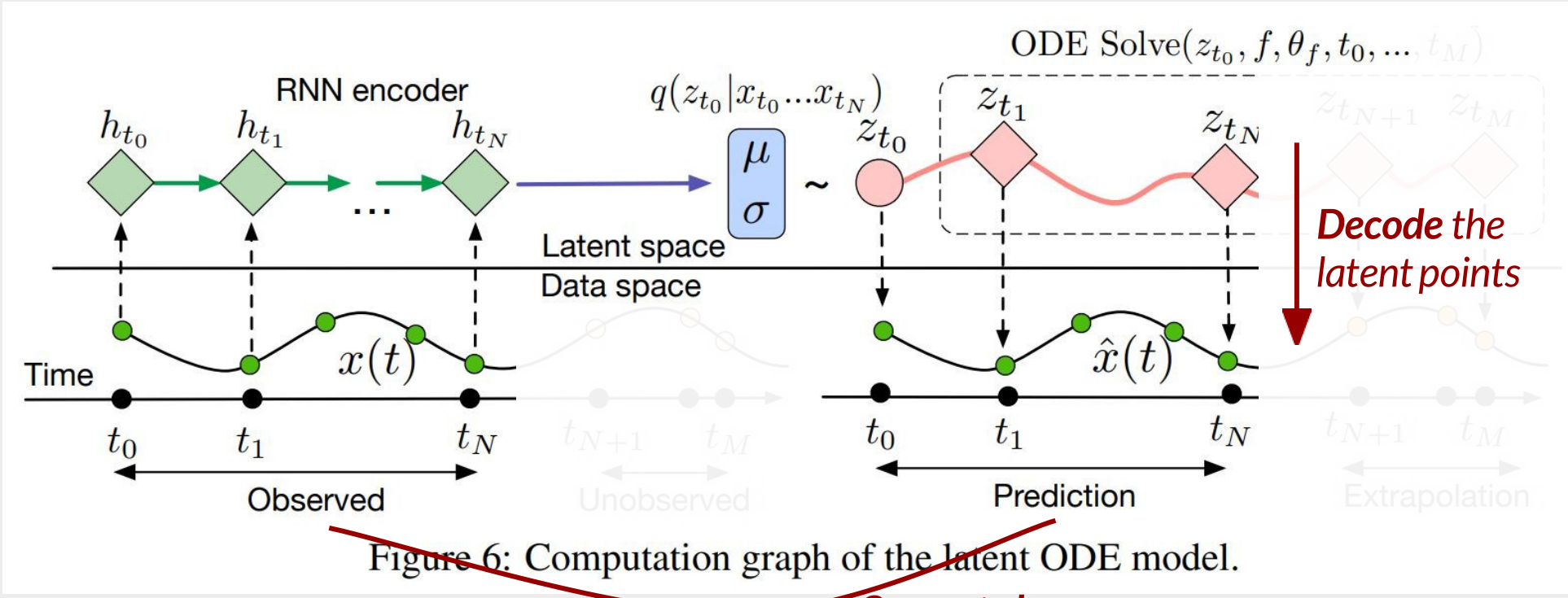


Figure 6: Computation graph of the latent ODE model.

Compute loss

Neural ODEs (Chen et al., 2018)

https://arxiv.org/pdf/1806.07366.pdf

Applications

Supervised Learning | Continuous Normalizing Flows | **Generative Latent Models**

*Solve the Neural ODE for latent points in **future** time steps*

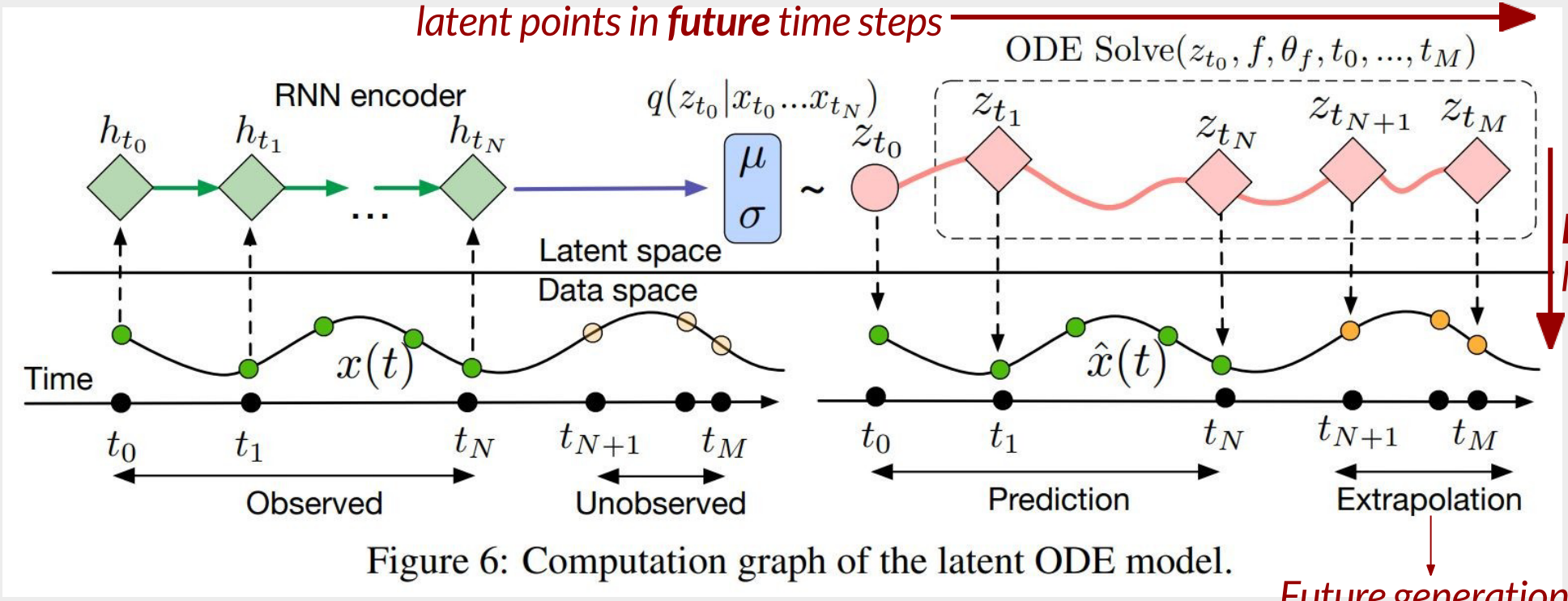


Figure 6: Computation graph of the latent ODE model.

***Decode** the latent points*

Future generation!

Later research

FFJORD: Free-form Continuous Dynamics For Scalable Reversible Generative Models (Grathwohl et al., ICLR 2019)

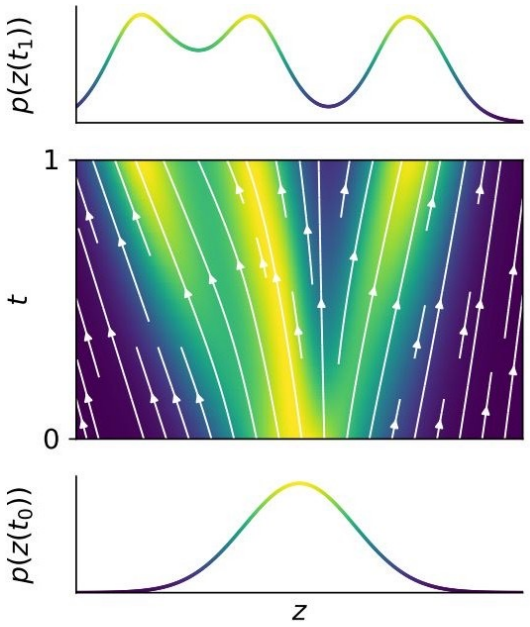


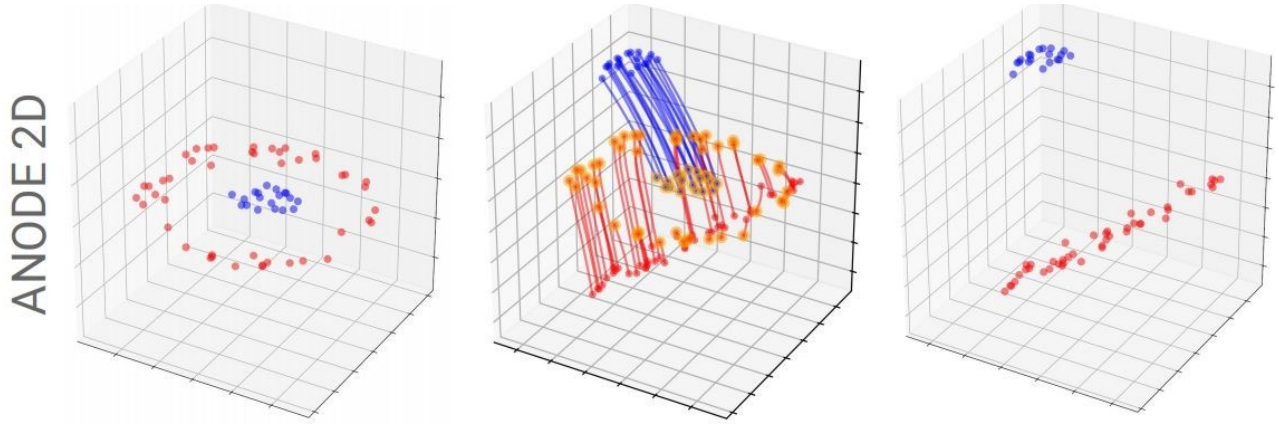
Figure 1: FFJORD transforms a simple base distribution at t_0 into the target distribution at t_1 by integrating over learned continuous dynamics.

- Essentially a better Continuous Normalizing Flow.
- Makes a better estimate for the log determinant term.
- “We demonstrate our approach on high-dimensional density estimation, image generation, and variational inference, achieving the state-of-the-art among exact likelihood methods with efficient sampling.”

https://arxiv.org/pdf/1810.01367.pdf

Later research

Augmented Neural ODEs (Dupont et al., NeurIPS 2019)



- Shows that Neural ODEs cannot model non-homeomorphisms (non-flows)
- **Augments** the state with additional dimensions to cover non-homeomorphisms
- Performs ablation study on toy examples and image classification

https://arxiv.org/pdf/1904.01681.pdf

Later research

ANODEV2: A Coupled Neural ODE Evolution Framework

(Zhang et al., NeurIPS 2019)

$$\begin{cases} z(1) = z(0) + \int_0^1 f(z(t), \theta(t))dt & \text{“parent network”}, \\ \theta(t) = \theta(0) + \int_0^t q(\theta(t), p)dt, \quad \theta(0) = \theta_0 & \text{“weight network”}. \end{cases}$$

- Network weights are also a function of time
- Separate “weight network” generates the weights of the function network ata given time

https://arxiv.org/pdf/1906.04596.pdf

Later research

Latent ODEs for Irregularly-Sampled Time Series

(Rubanova et al., NeurIPS 2019)

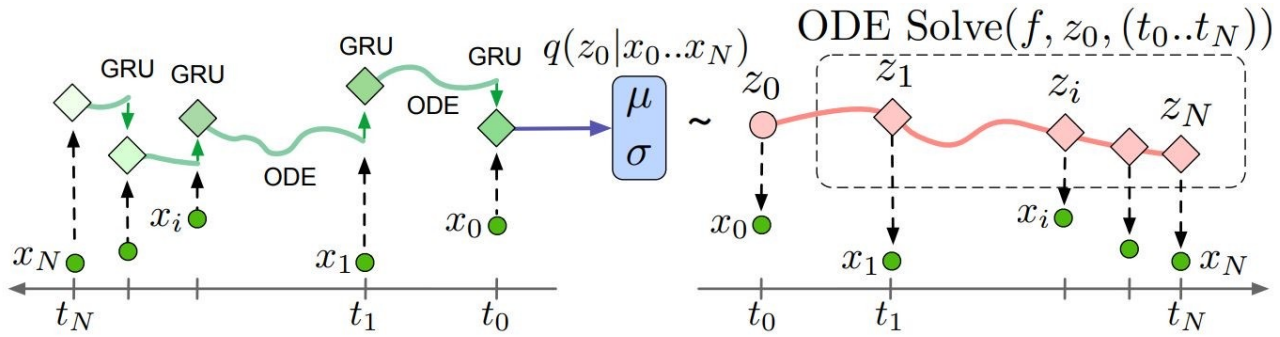


Figure 2: The Latent ODE model with an ODE-RNN encoder. To make predictions in this model, the ODE-RNN encoder is run backwards in time to produce an approximate posterior over the initial state: $q(z_0|\{x_i, t_i\}_{i=0}^N)$. Given a sample of z_0, we can find the latent state at any point of interest by solving an ODE initial-value problem. Figure adapted from Chen et al. [2018].

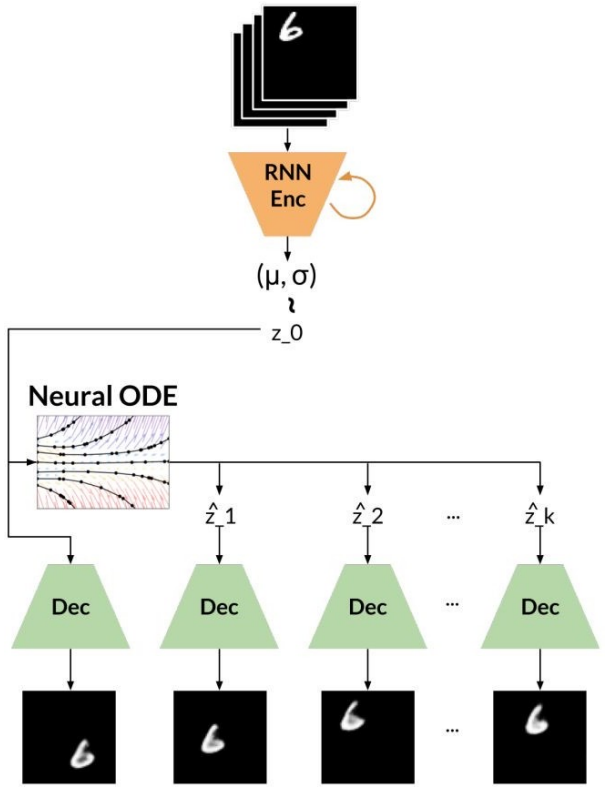
- Improves the generative latent variable framework for irregularly-sampled time series
- Essentially uses an ODE in the encoder where samples are missing
- Shows results on toy data, MuJoCo, PhysioNet

https://arxiv.org/pdf/1907.03907.pdf

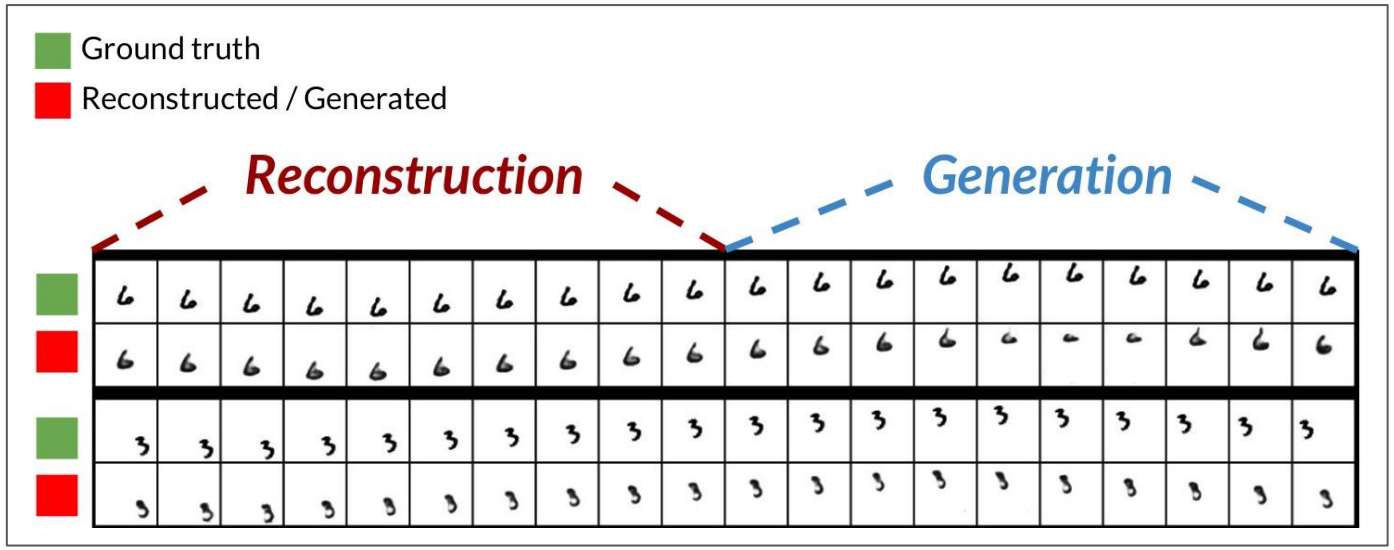
Later research

Simple Video Generation using Neural ODEs

(David Kanaa*, Vikram Voleti*, Samira Kahou, Christopher Pal; NeurIPS 2019 Workshop)



- Video generation as a generative latent variable model using Neural ODEs



https://sites.google.com/view/neurips2019lire/accepted-papers?authuser=0

Later research

ODE2VAE: Deep generative second order ODEs with Bayesian neural networks (Yildiz et al., NeurIPS 2019)

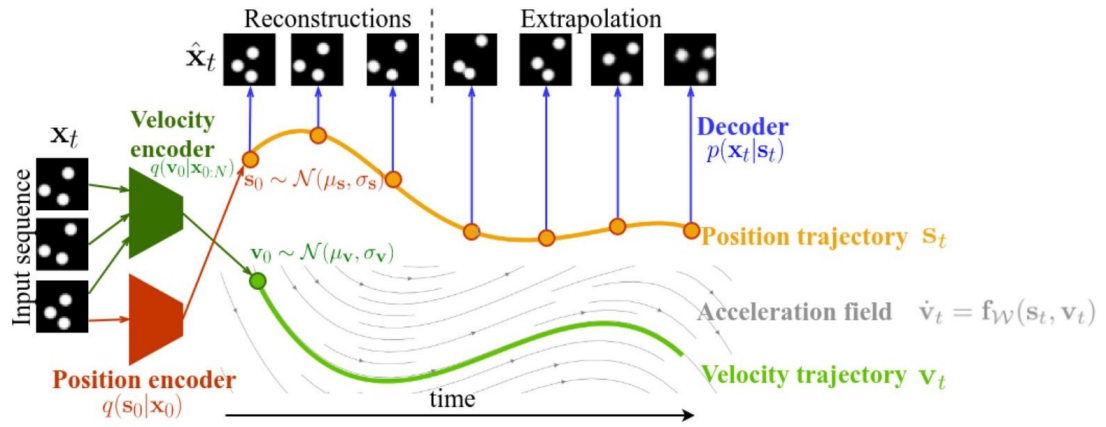


Figure 2: A schematic illustration of ODE^{2}VAE model. Position encoder $(\boldsymbol{\mu}_\mathbf{s}, \boldsymbol{\sigma}_\mathbf{s})$ maps the first item $\mathbf{x}_0$ of a high-dimensional data sequence into a distribution of the initial position $\mathbf{s}_0$ in a latent space. Velocity encoder $(\boldsymbol{\mu}_\mathbf{v}, \boldsymbol{\sigma}_\mathbf{v})$ maps the first m high-dimensional data items $\mathbf{x}_{0:m}$ into a distribution of the initial velocity $\mathbf{v}_0$ in a latent space. Probabilistic latent dynamics are implemented by a second order ODE model $\tilde{\mathrm{f}}_\mathcal{W}$ parameterised by a Bayesian deep neural network $(\mathcal{W})$. Data points in the original data domain are reconstructed by a decoder.

- Uses 2nd-order NeuralODE
- Uses a Bayesian Neural Network
- Showed results modelling video generation as a generative latent variable model using (2nd-order Bayesian) NeuralODE

https://papers.nips.cc/paper/9497-ode2vae-deep-generative-second-order-odes-with-bayesian-neural-networks.pdf

Later research

Neural Jump Stochastic Differential Equations (Jia et al., NeurIPS 2019)

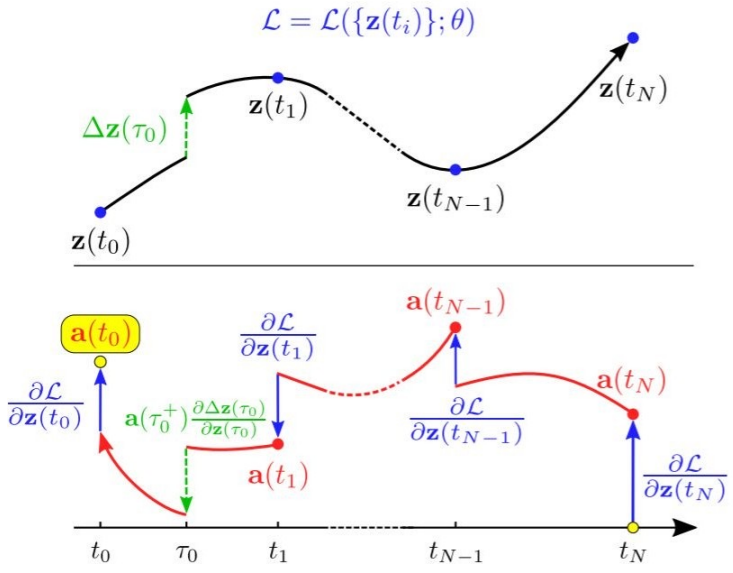


Figure 1: Reverse-mode differentiation of an ODE with discontinuities. Each jump $\Delta\mathbf{z}(\tau_j)$ in the latent vector (green, top panel) also introduces a discontinuity for adjoint vectors (green, bottom panel).

- Models continuous + discrete dynamics of a hybrid system
- Discontinuities are modelled as stochastic events
- Show results on real-world and synthetic point process datasets

https://papers.nips.cc/paper/9177-neural-jump-stochastic-differential-equations.pdf

Later research

Neural SDE: Stabilizing Neural ODE Networks with Stochastic Noise

(Liuet al., 2019)

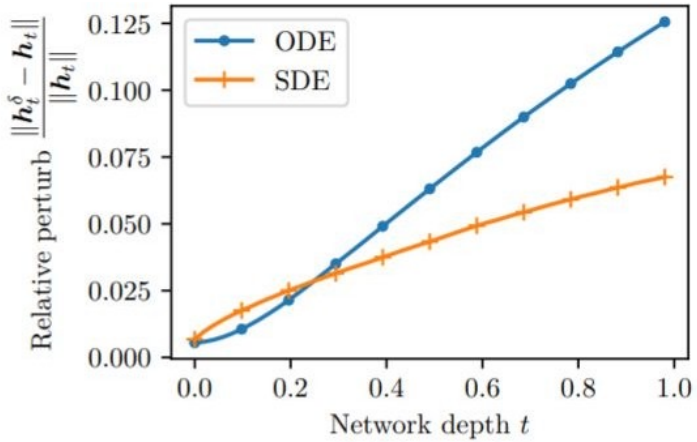


Figure 4: Comparing the perturbations of hidden states, ϵ_t, on both ODE and SDE (we choose dropout-style noise).

- Random noise injection into Neural ODEs
- Adds a diffusion term into the Neural ODE formulation, denoting a continuous time stochastic process
- Makes a case for robustness

https://arxiv.org/pdf/1906.02355.pdf

Later research

On Robustness of Neural Ordinary Differential Equations

(Yan et al., ICLR 2020)

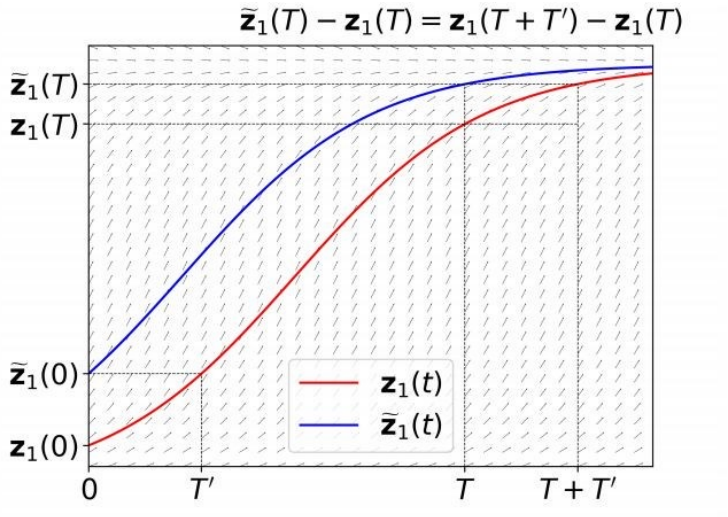


Figure 3: An illustration of the time-invariant property of ODEs. We can see that the curve $\widetilde{\mathbf{z}}_1(t)$ is exactly the horizontal translation of $\mathbf{z}_1(t)$ on the interval $[T', \infty)$.

- Ablation study on adversarial attacks on ODE-Nets
- Introduces new regularization term to improve robustness

https://arxiv.org/pdf/1910.05513.pdf, https://openreview.net/pdf?id=B1e9Y2NYvS

Later research

Approximation Capabilities of Neural ODEs and Invertible Residual Networks (Zhang et al., ICML 2020)

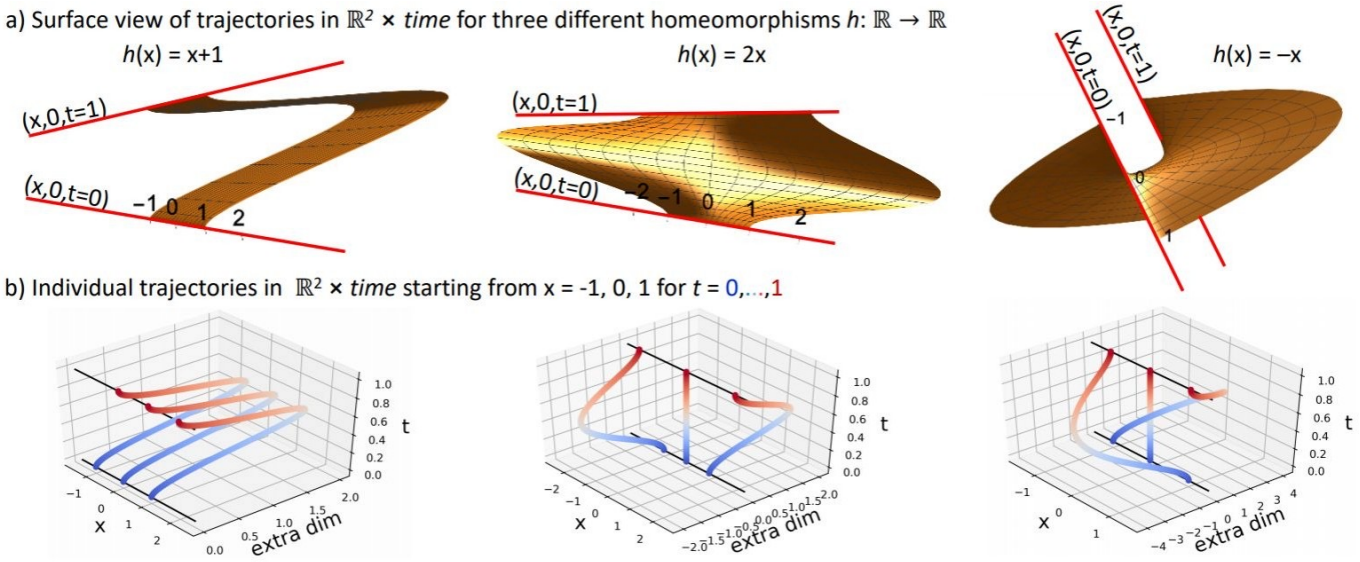


Figure 1: Trajectories in $\mathbb{R}^{2p}$ that embed an $\mathbb{R}^p \to \mathbb{R}^p$ homeomorphism, using $f(\tau) = (1 - \cos \pi\tau)/2$ and $g(\tau) = (1 - \cos 2\pi\tau)$. Three examples for $p = 1$ are shown, including the mapping $h(x) = -x$ that cannot be modeled by Neural ODE on $\mathbb{R}^p$, but can in $\mathbb{R}^{2p}$.

- Provides guarantees on modelling capability of homeomorphisms v/s the capacity of the Neural ODE

https://arxiv.org/pdf/1907.12998.pdf

Later research

How to Train Your Neural ODE : the world of Jacobian and kinetic regularization (Finlay et al., ICML 2020)

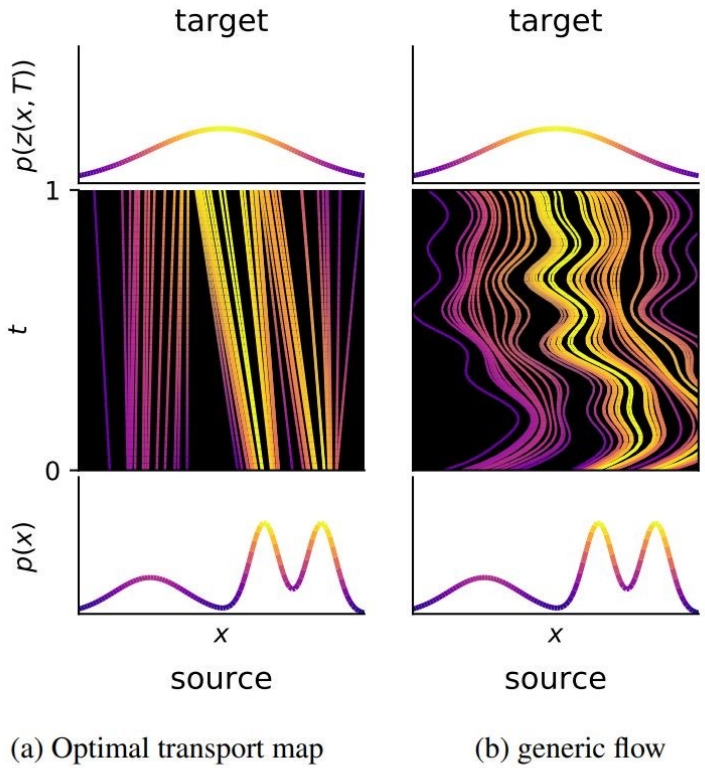


Figure 1. Optimal transport map and a generic normalizing flow.

- Makes a link between the flow in Neural ODEs and optimal transport
- Introduces two new regularization terms to constrain flows to straight lines
- Speeds up training of Neural ODEs

https://arxiv.org/pdf/2002.02798.pdf

Later research

Scalable Gradients for Stochastic Differential Equations
(Li et al., AISTATS 2020)

Diffusion model

- Generalizes the adjoint method to stochastic dynamics defined by SDEs : “stochastic adjoint sensitivitymethod.”
- PyTorch Implementation of Differentiable SDE Solvers: https://github.com/google-research/torchsde

https://arxiv.org/pdf/2001.01328.pdf

Liquid Time-Constant Network

What is a time-continuous neural network?

$$\frac{dx}{dt} = f_{(n,k,l_{type})}(x(t), I(t), \theta)$$

Number of layers

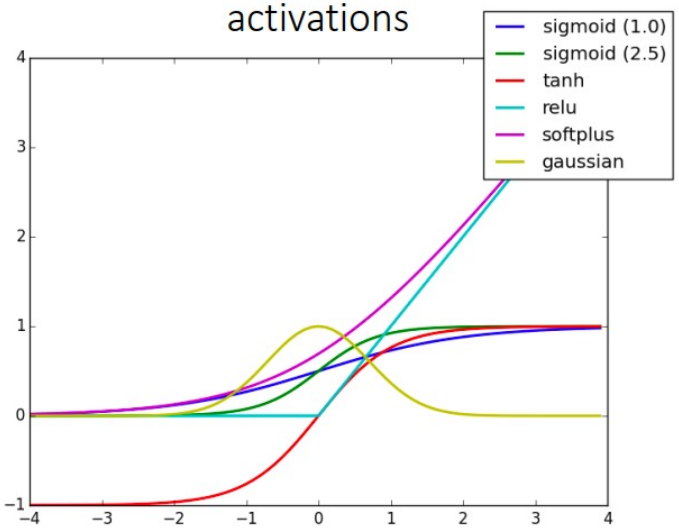
Width

Hidden state

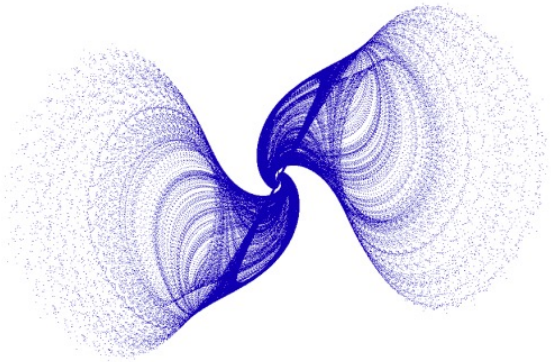
Inputs

Model parameters

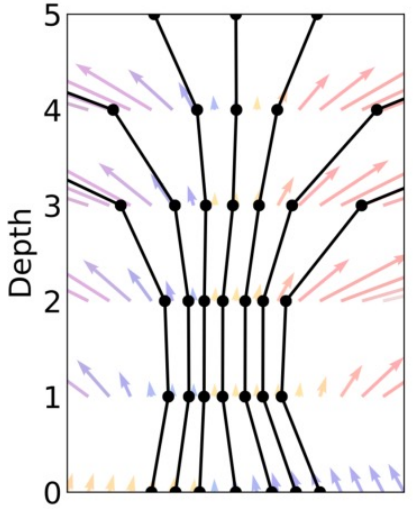
activations



Dynamical systems



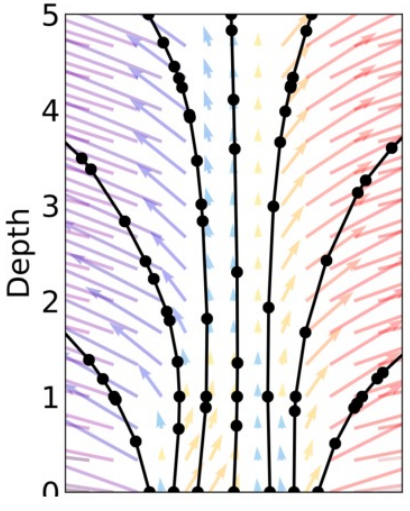
Residual Network



$$\mathbf{h}_{t+1} = \mathbf{h}_t + f(\mathbf{h}_t, \theta_t)$$

He et al.
CVPR 2016

ODE Network



$$\frac{d\mathbf{h}(t)}{dt} = f(\mathbf{h}(t), t, \theta)$$

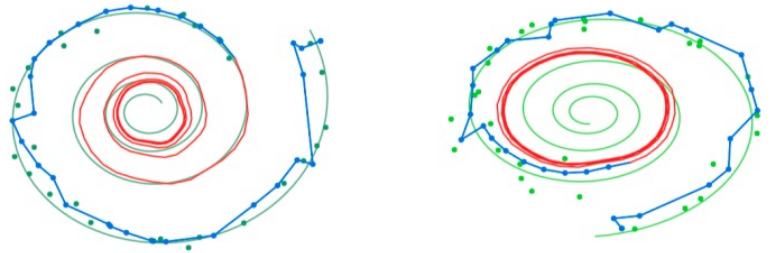
Chen et al.
NeurIPS 2018

Figure Credit: Chen et al. NeurIPS 2018

What is a time-continuous neural network?

Standard Recurrent Neural Network (RNN)
Hopfield 1982

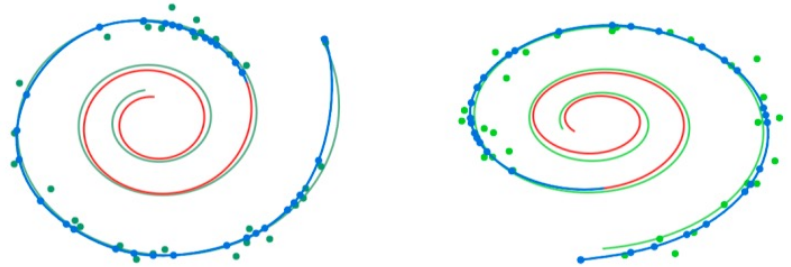
$$x(t+1) = f(x(t), I(t), t\,; \theta)$$



(a) Recurrent Neural Network

Neural ODE
Chen et al. NeurIPS, 2018

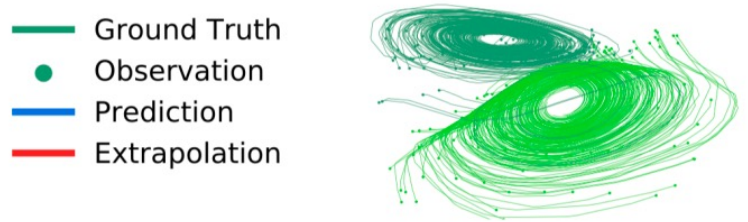
$$\frac{dx(t)}{dt} = f(x(t), I(t), t\,; \theta)$$



(b) Latent Neural Ordinary Differential Equation

Continuous-time (CT) RNN
Funahashi et al. 1993

$$\frac{dx(t)}{dt} = -\frac{x(t)}{\tau} + f(x(t), I(t), t\,; \theta)$$



Liquid Time-Constant Networks

$$d\mathbf{x}(t)/dt = -\mathbf{x}(t)/\tau + \mathbf{S}(t) \qquad \mathbf{S}(t) \in \mathbb{R}^M$$

$$\mathbf{S}(t) = f(\mathbf{x}(t), \mathbf{I}(t), t, \theta)(A - \mathbf{x}(t))$$

$$\frac{d\mathbf{x}(t)}{dt} = -\left[\frac{1}{\tau} + f(\mathbf{x}(t), \mathbf{I}(t), t, \theta)\right]\mathbf{x}(t) + f(\mathbf{x}(t), \mathbf{I}(t), t, \theta)A$$

"Liquid" = variable

LTCs have stable state and time-constant

System time-constant

$$\frac{d\mathbf{x}(t)}{dt} = -\left[\frac{1}{\tau} + f(\mathbf{x}(t), \mathbf{I}(t), t, \theta)\right]\mathbf{x}(t) + f(\mathbf{x}(t), \mathbf{I}(t), t, \theta)A \qquad \tau_{sys} = \frac{\tau}{1+\tau f(\mathbf{x}(t),\mathbf{I}(t),t,\theta)}$$

Theorem 1. *Let x_i denote the state of a neuron i within an LTC network identified by Eq. 1, and let neuron i receive M incoming connections. Then, the time-constant of the neuron, τ_{sys_i}, is bounded to the following range:*

$$\tau_i/(1 + \tau_i W_i) \leq \tau_{sys_i} \leq \tau_i, \qquad (4)$$

Theorem 2. *Let x_i denote the state of a neuron i within an LTC, identified by Eq. 1, and let neuron i receive M incoming connections. Then, the hidden state of any neuron i, on a finite interval $Int \in [0, T]$, is bounded as follows:*

$$min(0, A_i^{min}) \leq x_i(t) \leq max(0, A_i^{max}), \qquad (5)$$

Liquid Time-Constant Networks are Universal Approximators

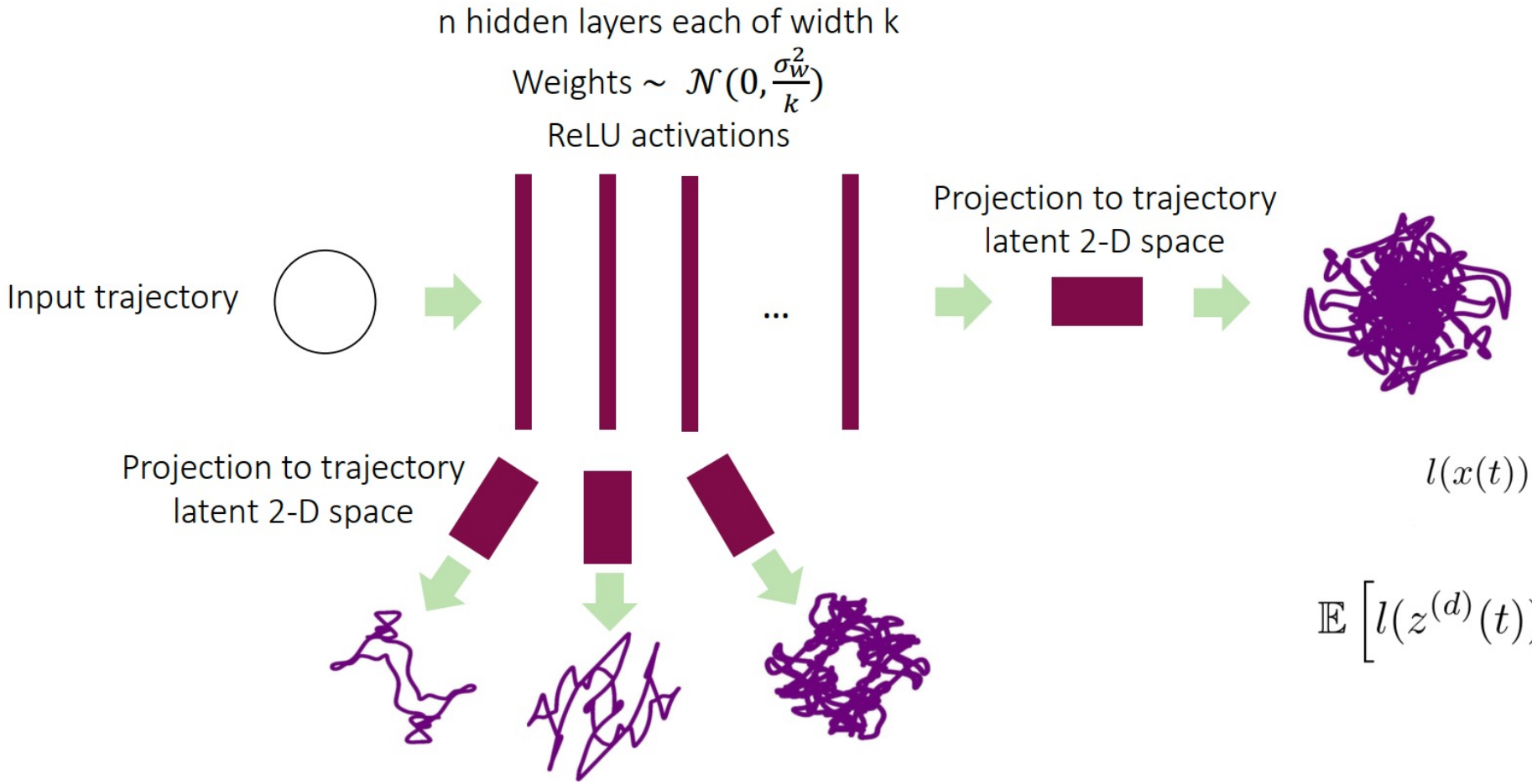
$$\frac{d\mathbf{x}(t)}{dt} = -\left[\frac{1}{\tau} + f(\mathbf{x}(t), \mathbf{I}(t), t, \theta)\right]\mathbf{x}(t) + f(\mathbf{x}(t), \mathbf{I}(t), t, \theta)A$$

Theorem 3. *Let $\boldsymbol{x} \in \mathbb{R}^n$, $S \subset \mathbb{R}^n$ and $\dot{\boldsymbol{x}} = F(\boldsymbol{x})$ be an autonomous ODE with $F : S \to \mathbb{R}^n$ a C^1-mapping on S. Let D denote a compact subset of S and assume that the simulation of the system is bounded in the interval $I = [0, T]$. Then, for a positive ϵ, there exist an LTC network with N hidden units, n output units, and an output internal state $\boldsymbol{u}(t)$, described by Eq. 1, such that for any rollout $\{\boldsymbol{x}(t) | t \in I\}$ of the system with initial value $x(0) \in D$, and a proper network initialization,*

$$max_{t \in I} |\boldsymbol{x}(t) - \boldsymbol{u}(t)| < \epsilon \qquad (6)$$

Expressivity

Raghu et al. 提出了一种新的关于神经网络表达能力的measurement



$$l(x(t)) = \int_t \left|\left| \frac{dx(t)}{dt} \right|\right| dt$$ Arc-length

$$\mathbb{E}\left[l(z^{(d)}(t)) \right] \geq O\left(\frac{\sigma_w \sqrt{k}}{\sqrt{k+1}} \right)^d l(x(t))$$

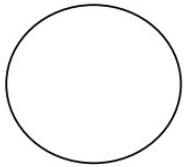
[Raghu et al. ICML 2017]

[Raghu et al. ICML 2017] Trajectory Length as a measure of expressivity of Deep networks

Expressivity

Let's implement the trajectory space for time-continuous models

Input trajectory



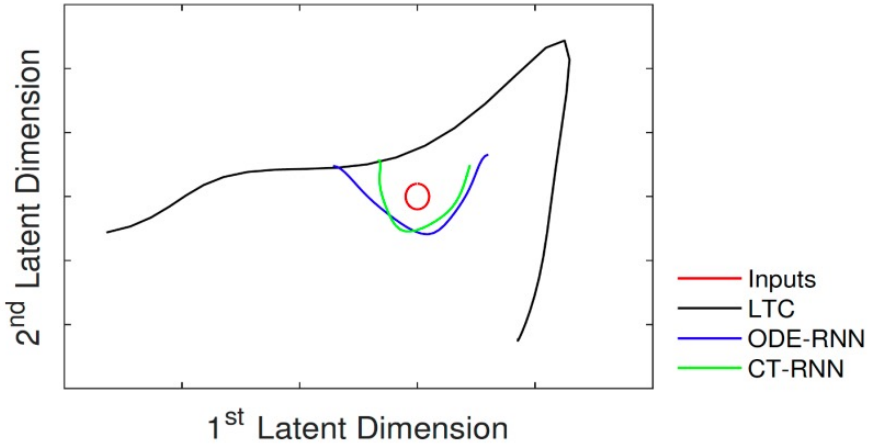
LTC
CT-RNN
Neural ODE

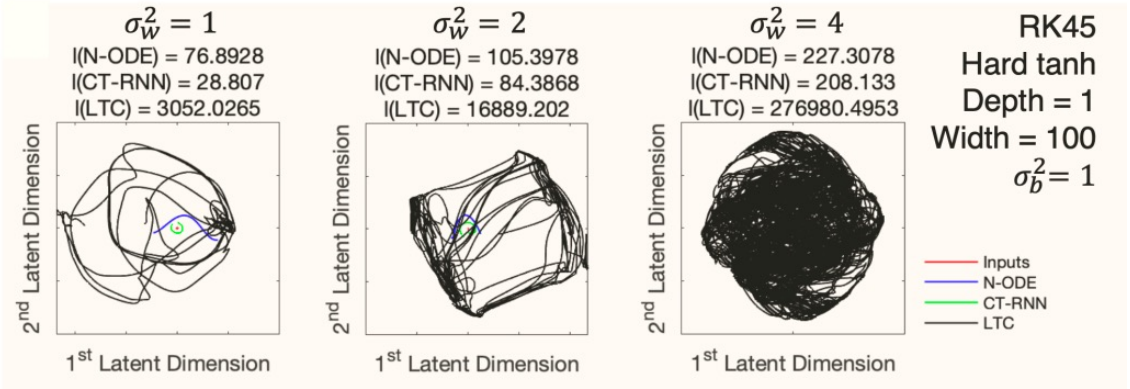
Weights ~ $\mathcal{N}(0, \frac{\sigma_w^2}{k})$

activation functions:
ReLU, tanh, logistic sigmoid

Projection to a latent trajectory 2-D space

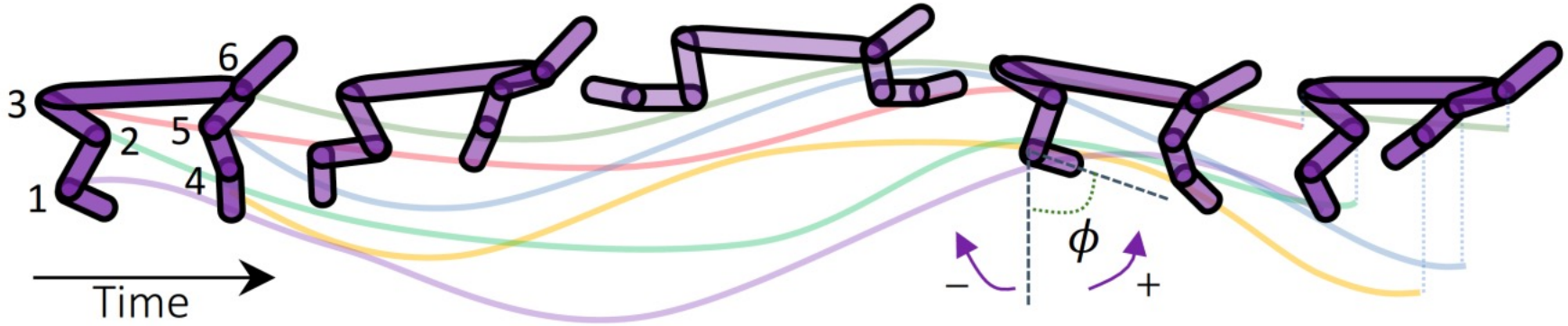
PCA





Performance

- LTCs in modeling physical dynamics



17 input observations | 6 control outputs | ϕ = joint angle

Table 6: Sequence modeling Half-Cheetah dynamics n=5

Algorithm	MSE
LSTM	2.500± 0.140
CT-RNN	2.838± 0.112
Neural ODE	3.805 ± 0.313
CT-GRU	3.014± 0.134
LTC (ours)	**2.308± 0.015**

Performance

- LTCs in modeling real-life time series data

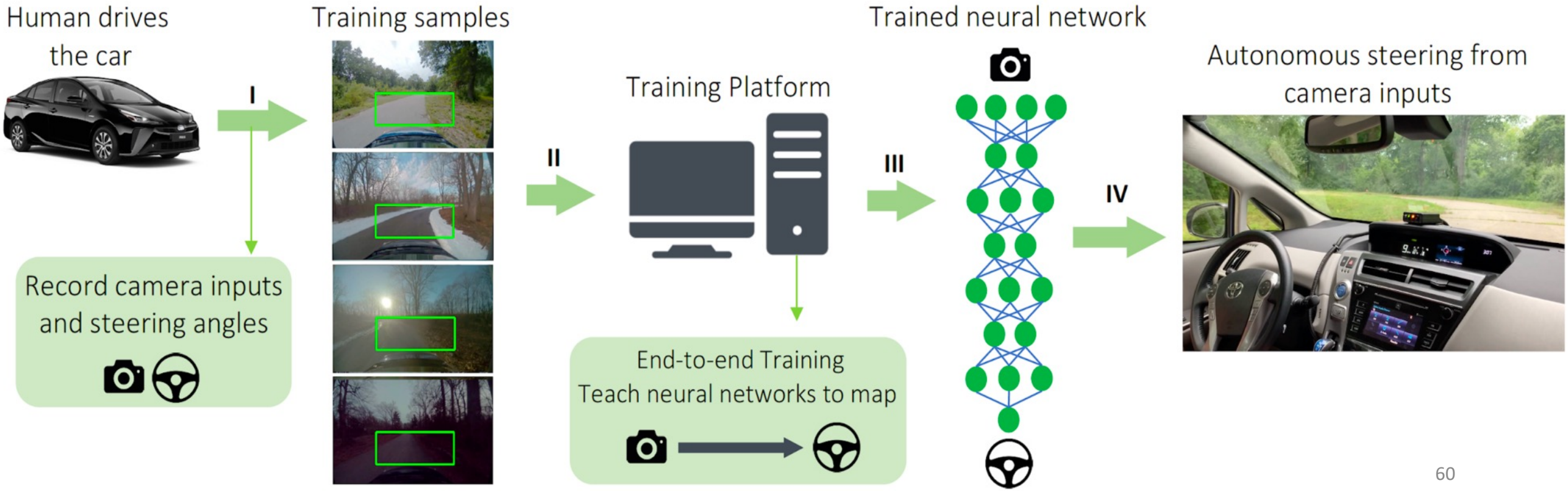
Table 3: **Time series prediction** Mean and standard deviation, n=5

Dataset	Metric	LSTM [28]	CT-RNN [47]	Neural ODE [6]	CT-GRU [38]	LTC (ours)
Gesture	(accuracy)	64.57% ± 0.59	59.01% ± 1.22	46.97% ± 3.03	68.31% ± 1.78	**69.55% ± 1.13**
Occupancy	(accuracy)	93.18% ± 1.66	94.54% ± 0.54	90.15% ± 1.71	91.44% ± 1.67	**94.63% ± 0.17**
Activity recognition	(accuracy)	95.85% ± 0.29	95.73% ± 0.47	**97.26%** ± 0.10	96.16% ± 0.39	95.67% ± 0.575
Sequential MNIST	(accuracy)	**98.41%** ± 0.12	96.73% ± 0.19	97.61% ± 0.14	98.27% ± 0.14	97.57% ± 0.18
Traffic	(squared error)	0.169 ± 0.004	0.224 ± 0.008	1.512 ± 0.179	0.389 ± 0.076	**0.099** ± 0.0095
Power	(squared-error)	0.628 ± 0.003	0.742 ± 0.005	1.254 ± 0.149	**0.586** ± 0.003	0.642 ± 0.021
Ozone	(F1-score)	0.284 ± 0.025	0.236 ± 0.011	0.168 ± 0.006	0.260 ± 0.024	**0.302** ± 0.0155

[28] Hochreiter et al. 1997
[6] Chen et al. NeurIPS, 2018
[38] Moser et al. Arxiv, 2017
[47] Rubanova et al. NeurIPS 2019

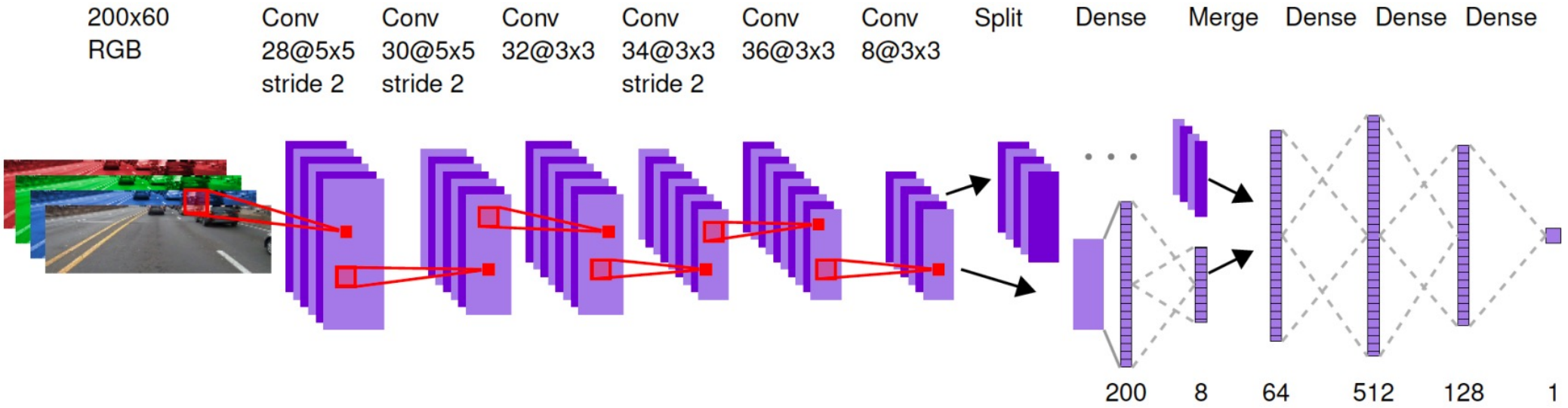
Performance

- High-fidelity autonomy by LTCs
- end-to-end learning



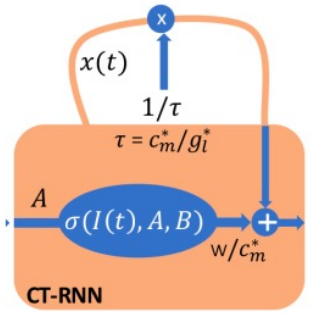
LTCs: Performance

- High-fidelity autonomy by LTCs - end-to-end learning

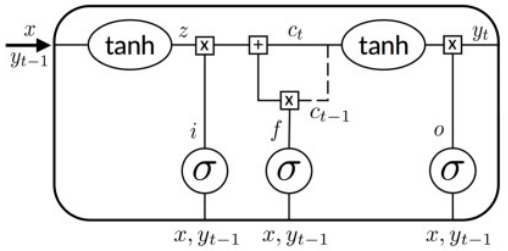


What if we replace the fully connected layers by a recurrent neural network?

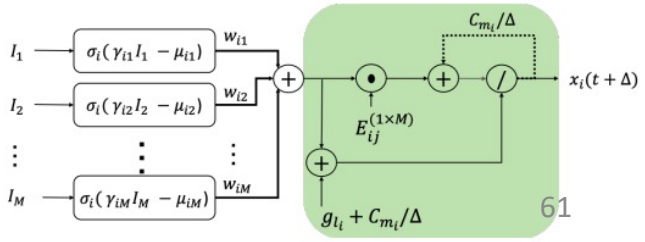
ODE-RNN



LSTMs

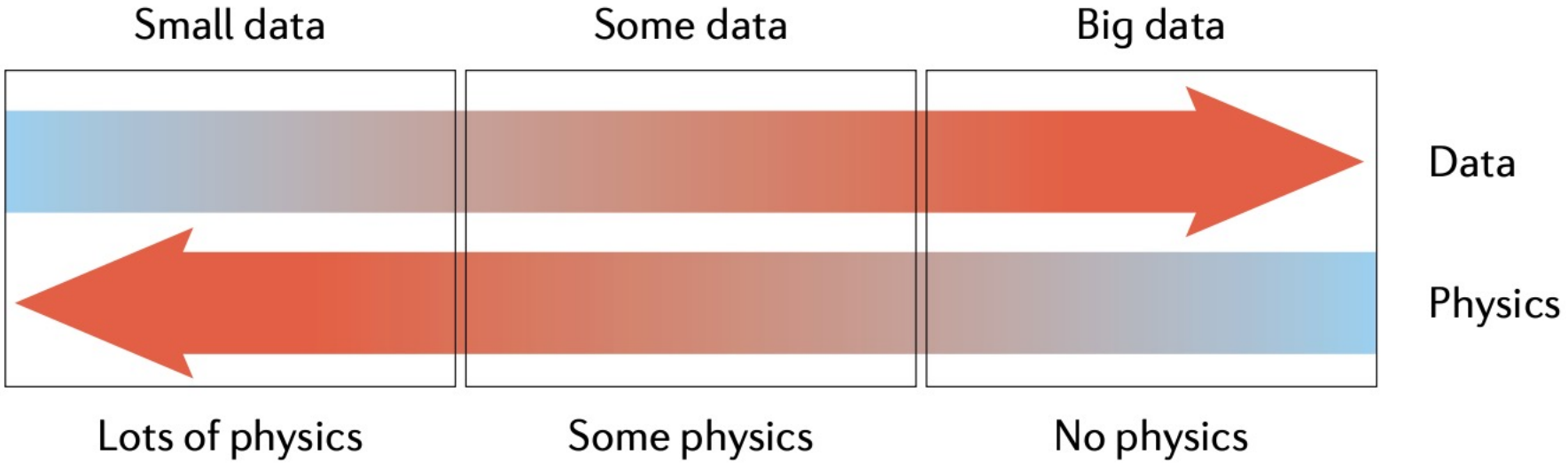


LTC-based Networks?



DeepONet

Data & physics



- Lots of physics—Forward problems
  - Finite difference/elements 有限元方法
- Some physics—Inverse problems
  - Multi-fidelity learning (多精度学习)
  - Physics-informed neural network(PINN)[1]
  - DeepM&Mnet[2]
- No physics -- System identification/discovery
  - Operator Learning(DeepONet)

[1] Karniadakis, George Em, et al. "Physics-informed machine learning." *Nature Reviews Physics* 3.6 (2021): 422-440.
[2] Cai, S., Wang, Z., Lu, L., Zaki, T. A., & Karniadakis, G. E. (2021). DeepM&Mnet: Inferring the electroconvection multiphysics fields based on operator approximation by neural networks. *Journal of Computational Physics*, *436*, 110296.

From function to operator

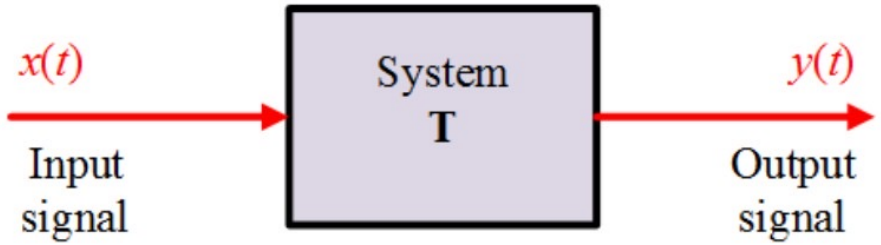
⇒ Can we learn operators via neural networks?

⇒ How?

- Function: $\mathrm{R}^{d_1} \longrightarrow \mathrm{R}^{d_2}$
  - e.g., image classification:

5

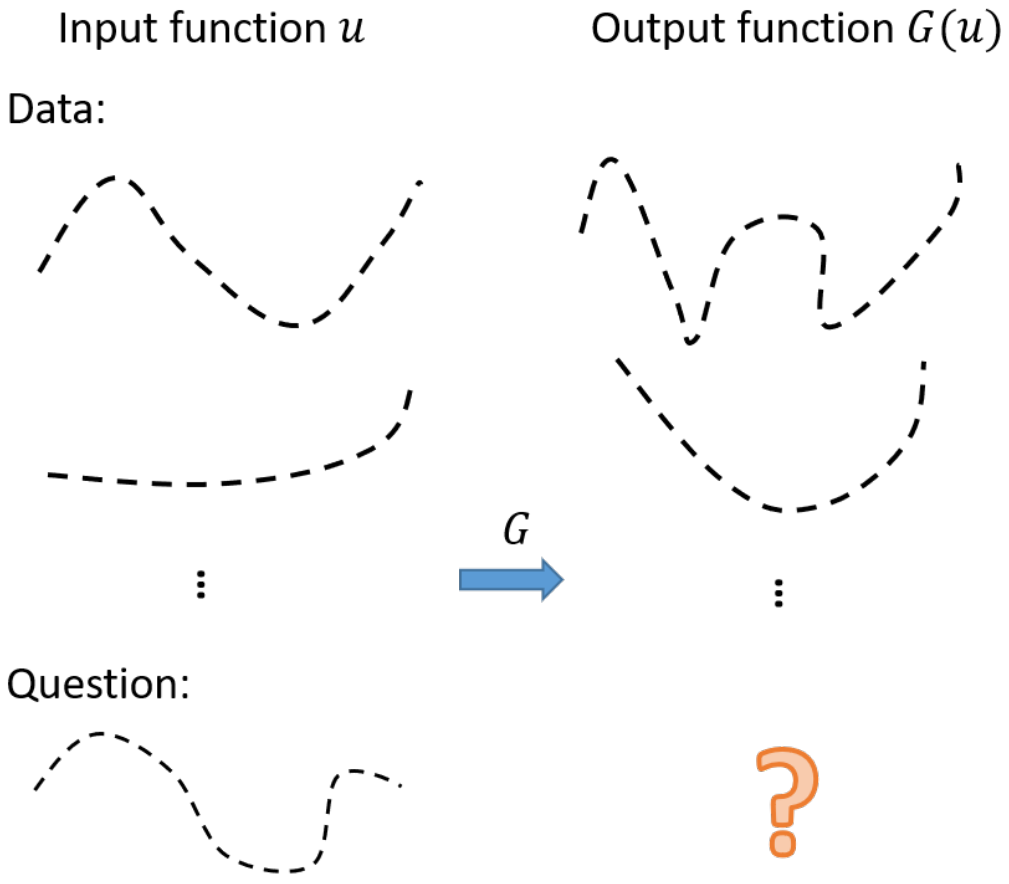
- Operator: function (∞-dim) ⟶ function (∞-dim)
  - e.g., derivative (local): $x(t) \mapsto x'(t)$
  - e.g., integral (global): $x(t) \mapsto \int K(s,t)x(s)ds$
  - e.g., dynamic system:
  - e.g., biological system
  - e.g., social system



Problem setup

$$G : u \in V \subset C(K_1) \mapsto G(u) \in C(K_2)$$

$$G(u) : y \in \mathbb{R}^d \mapsto G(u)(y) \in \mathbb{R}$$



Universal Approximation Theorem for Operator

$G : u \in V \subset C(K_1) \mapsto G(u) \in C(K_2)$

$G(u) : y \in \mathbb{R}^d \mapsto G(u)(y) \in \mathbb{R}$

Theorem (Chen & Chen, 1995)

Suppose that σ is a continuous non-polynomial function, X is a Banach Space, $K_1 \subset X$, $K_2 \subset \mathbb{R}^d$ are two compact sets in X and $\mathbb{R}^d$, respectively, V is a compact set in $C(K_1)$, G is a continuous operator, which maps V into $C(K_2)$.

Then for any $\epsilon > 0$, there are positive integers n, p, m, constants $c_i^k, \xi_{ij}^k, \theta_i^k, \zeta_k \in \mathbb{R}$, $w_k \in \mathbb{R}^d$, $x_j \in K_1$, $i = 1, \ldots, n$, $k = 1, \ldots, p$, $j = 1, \ldots, m$, such that

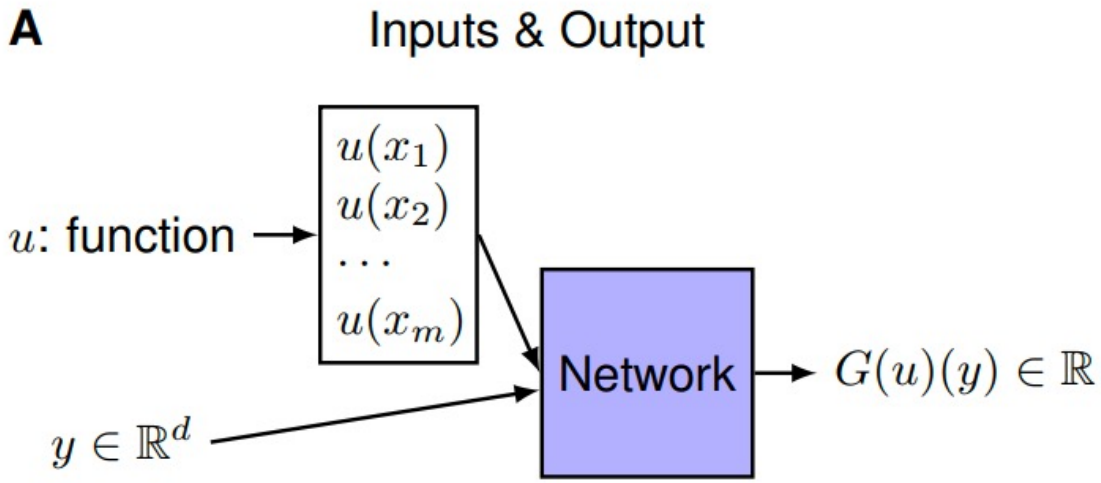
$$\left| G(u)(y) - \sum_{k=1}^{p} \sum_{i=1}^{n} c_i^k \sigma \left(\sum_{j=1}^{m} \xi_{ij}^k u(x_j) + \theta_i^k \right) \sigma(w_k \cdot y + \zeta_k) \right| < \epsilon$$

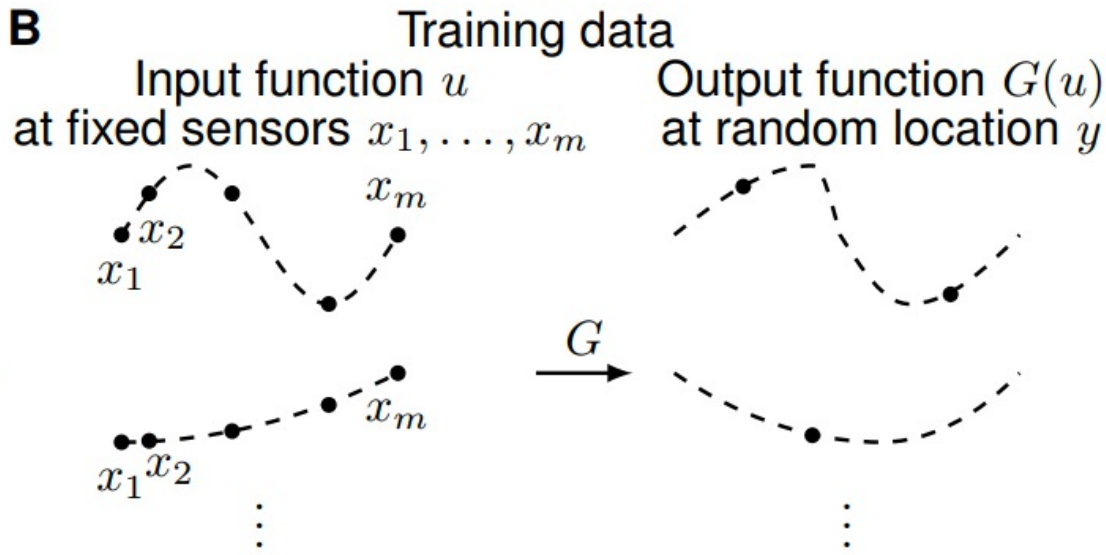
holds for all $u \in V$ and $y \in K_2$.

Problem setup

$$G : u \in V \subset C(K_1) \mapsto G(u) \in C(K_2)$$

$$G(u) : y \in \mathbb{R}^d \mapsto G(u)(y) \in \mathbb{R}$$



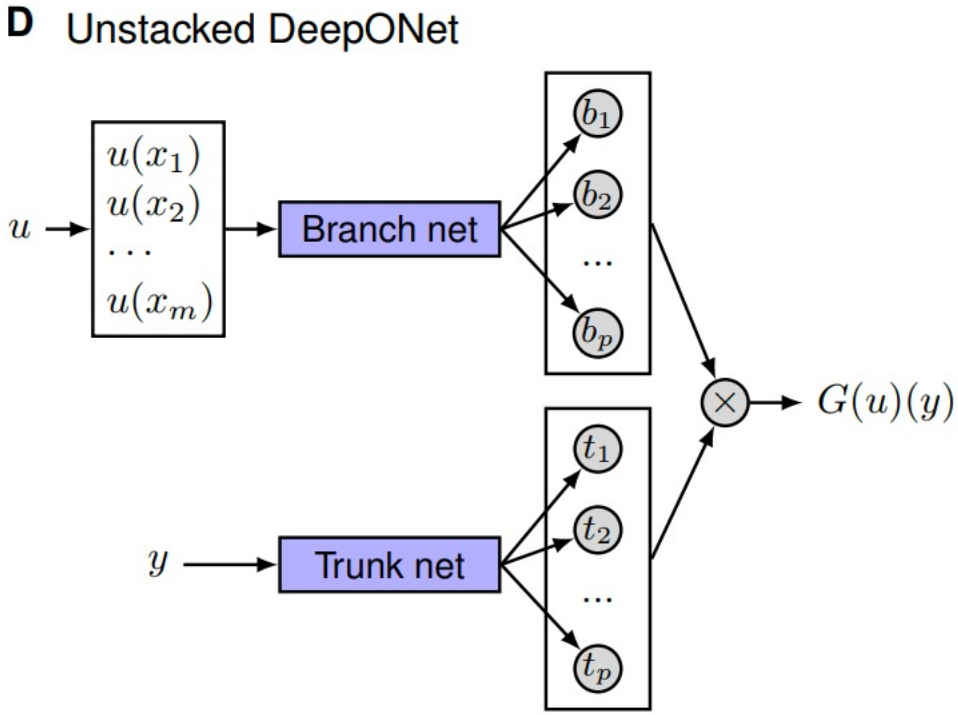


- 输入： u(.) at $\{x_1, x_2, \ldots, x_m\} \in \mathbb{R}^m, y \in \mathbb{R}^d$

- 输出： $G(u)(y) \in \mathbb{R}$

Deep operator network (DeepONet)

$$G(u)(y) \approx \sum_{k=1}^{p} \underbrace{b_k(u)}_{\text{branch}} \underbrace{t_k(y)}_{\text{trunk}}$$



Idea: $G(u)(y)$: a function of y conditioning on u

$t_k(y)$: basis function of y

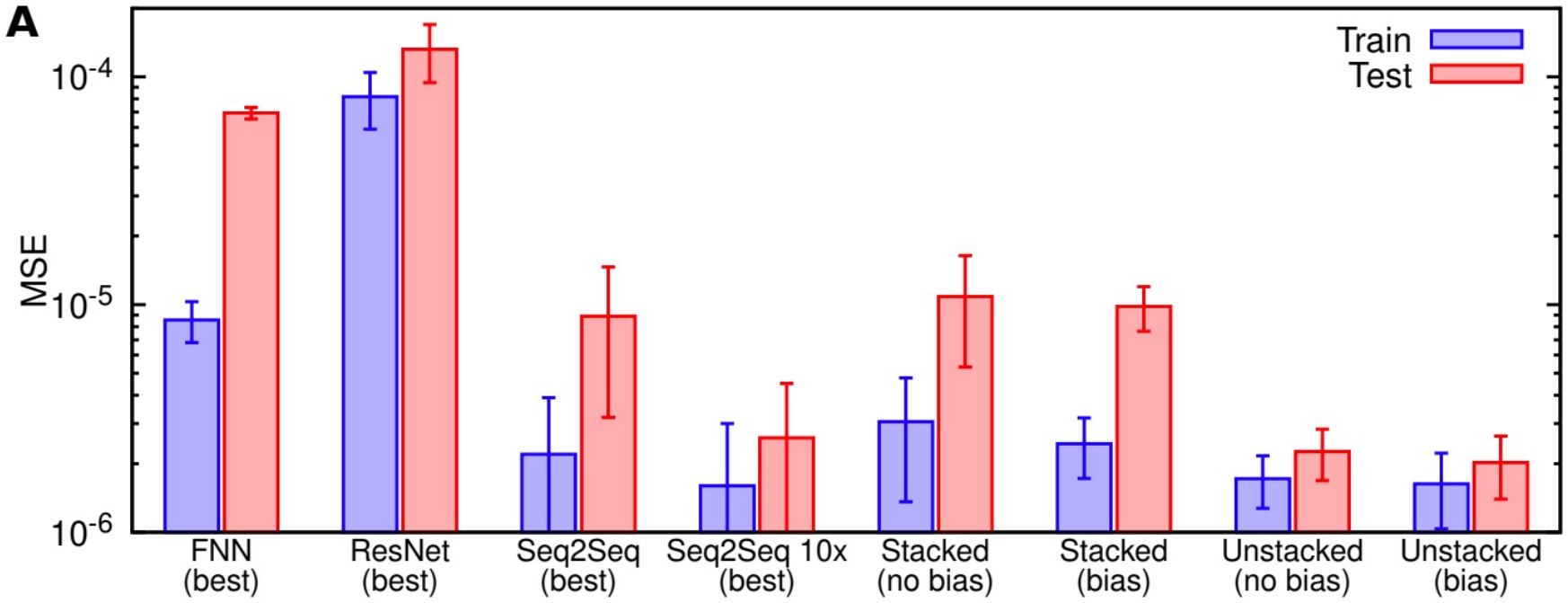
$b_k(u)$: u-dependent coefficient, i.e., functional of u

Explicit operator: A simple ODE case

$$\frac{ds(x)}{dx} = u(x), \quad x \in [0,1], \quad s(0) = 0$$

$$G : u(x) \mapsto s(y) = s(0) + \int_0^y u(\tau) d\tau$$

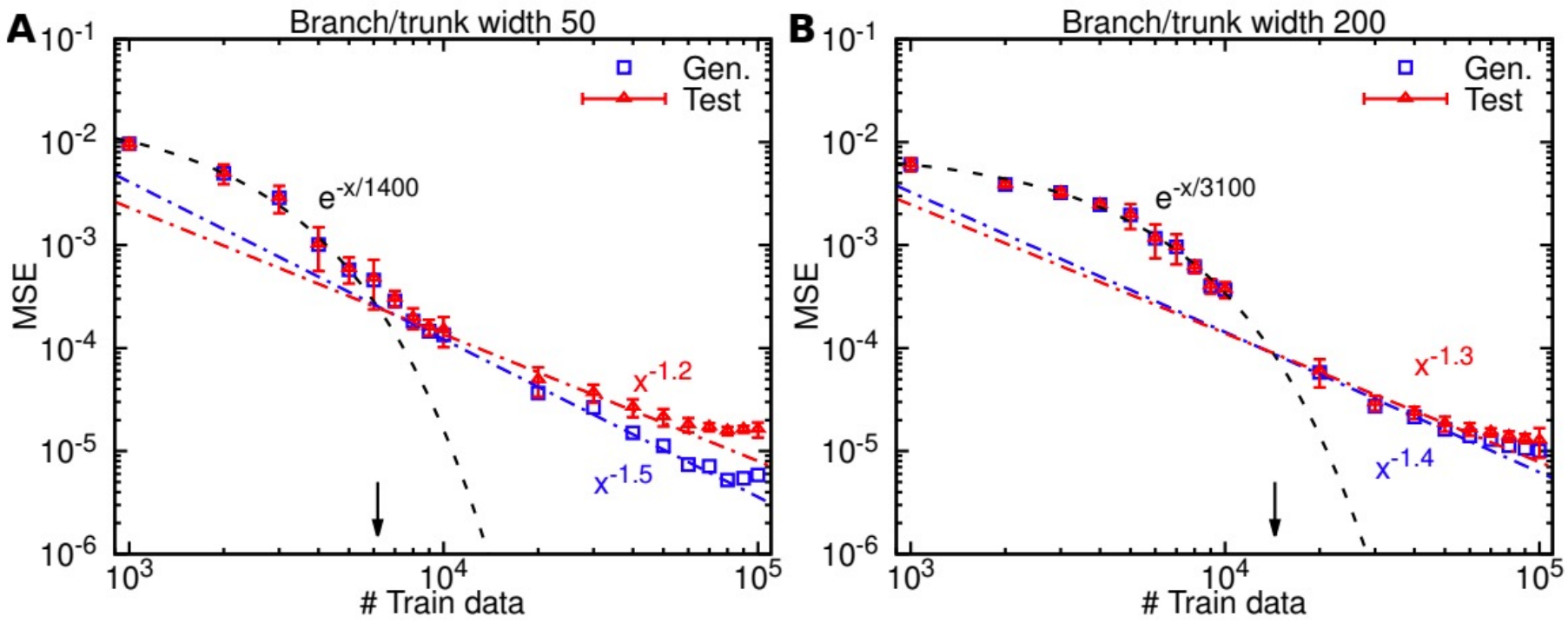
Very small generalization error!



Implicit operator: Gravity pendulum with an external force

$$\frac{ds_1}{dt} = s_2, \quad \frac{ds_2}{dt} = -k \sin s_1 + u(t)$$

$$G : u(t) \mapsto \mathbf{s}(t)$$



Test/generalization error:

- small dataset: exponential convergence
- large dataset: polynomial rate
- larger network has later transition point

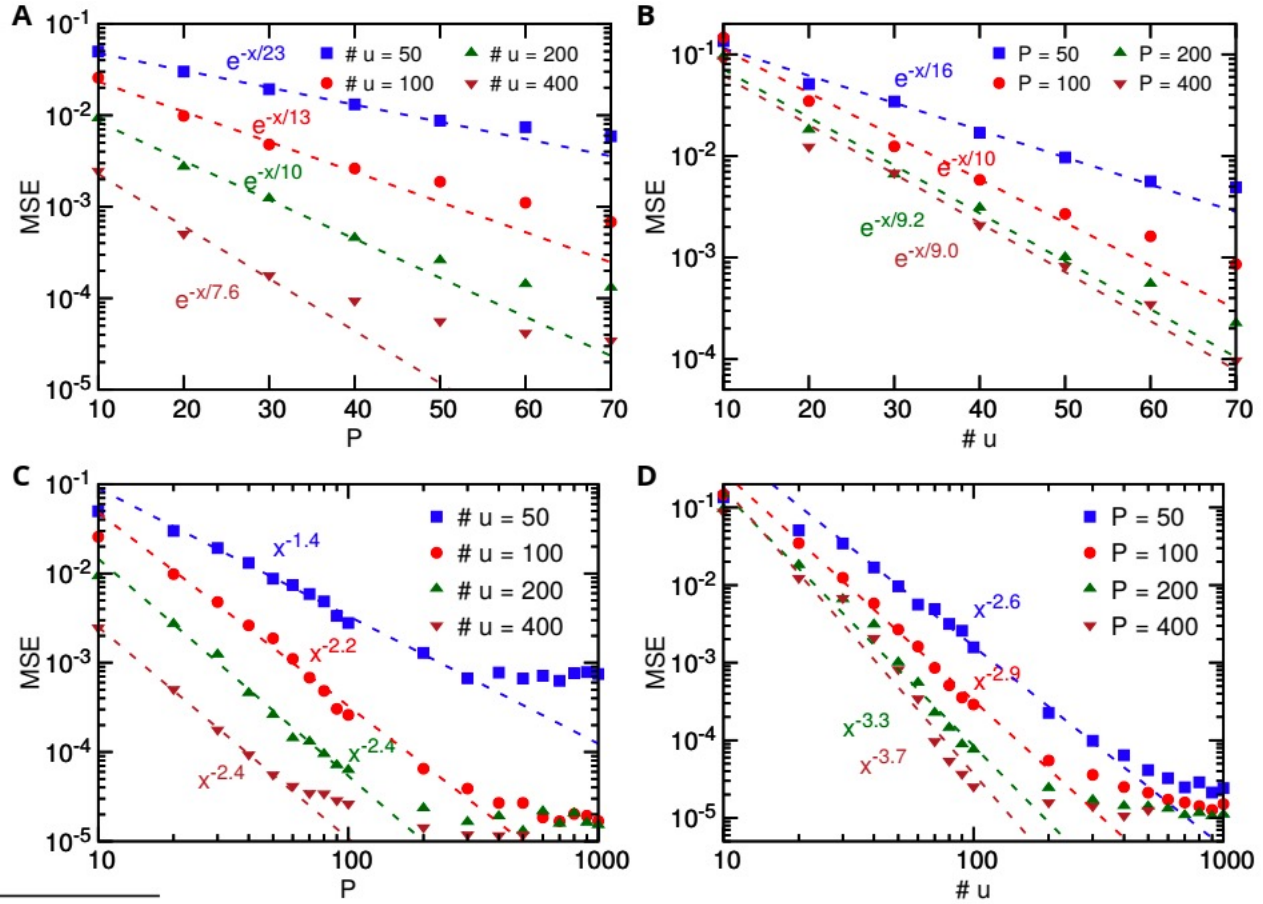
Implicit operator: Diffusion-reaction system

$$\frac{\partial s}{\partial t} = D\frac{\partial^2 s}{\partial x^2} + ks^2 + u(x), \quad x \in [0,1], t \in [0,1]$$

$$\#\ \text{Training points} = \#u \times P$$

Small dataset:
Exponential convergence

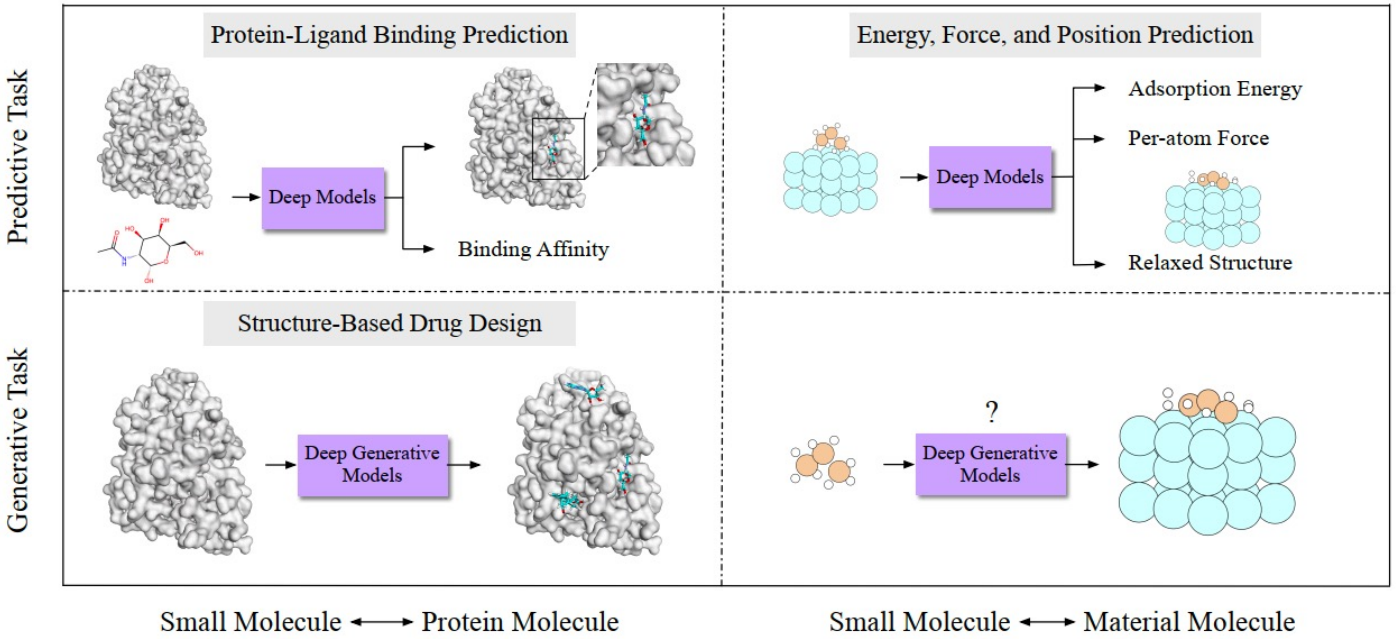
Large dataset:
Polynomial convergence



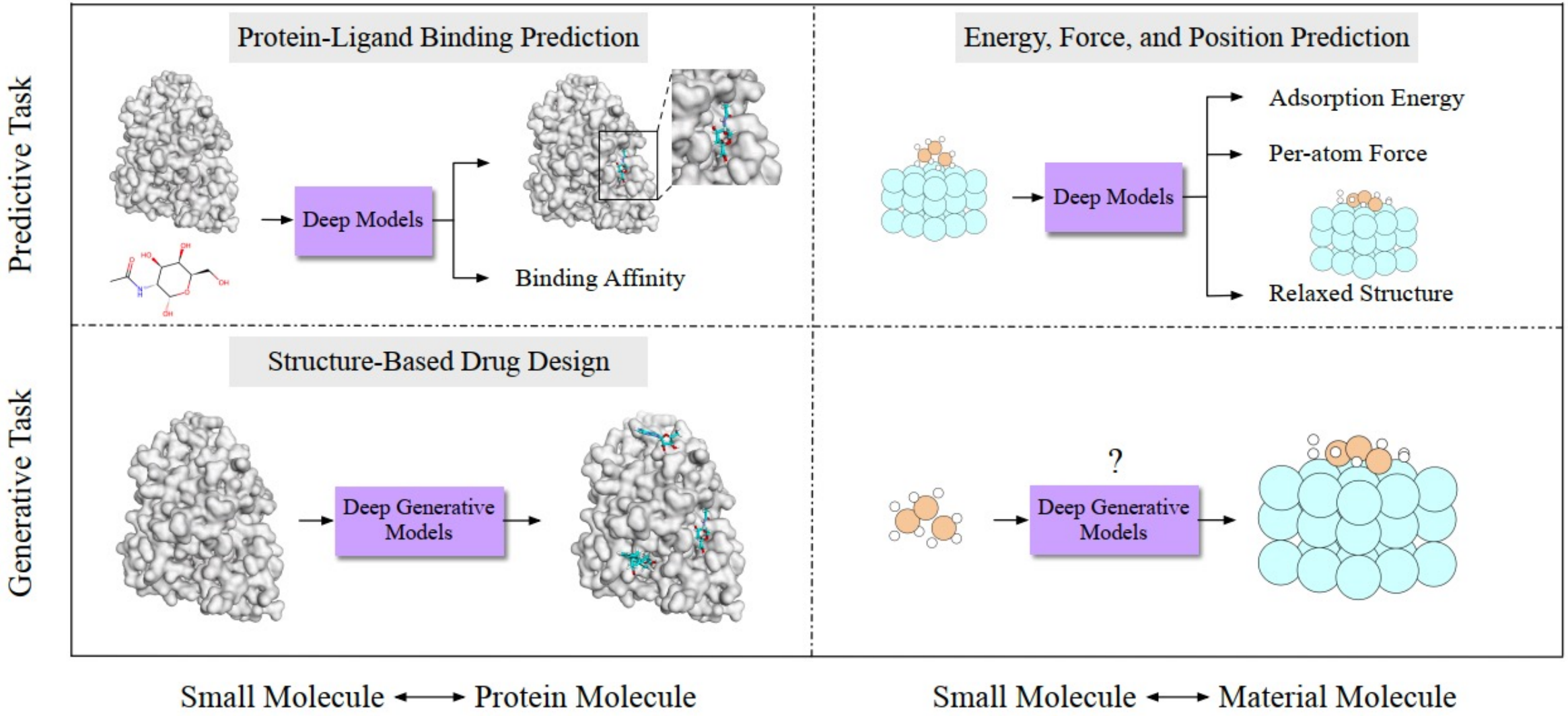
生物

人工智能用于分子交互

- 致力于利用人工智能来洞察治理分子间交互的分子机制
- 具有推进我们对分子交互理解和为生命科学与材料科学范围内的各种挑战提供实际解决方案的巨大潜力
- 对于分子-蛋白质和分子-材料的交互,我们将现有的任务分为预测任务和生成任务（基于任务的性质,而不是执行任务的方法）



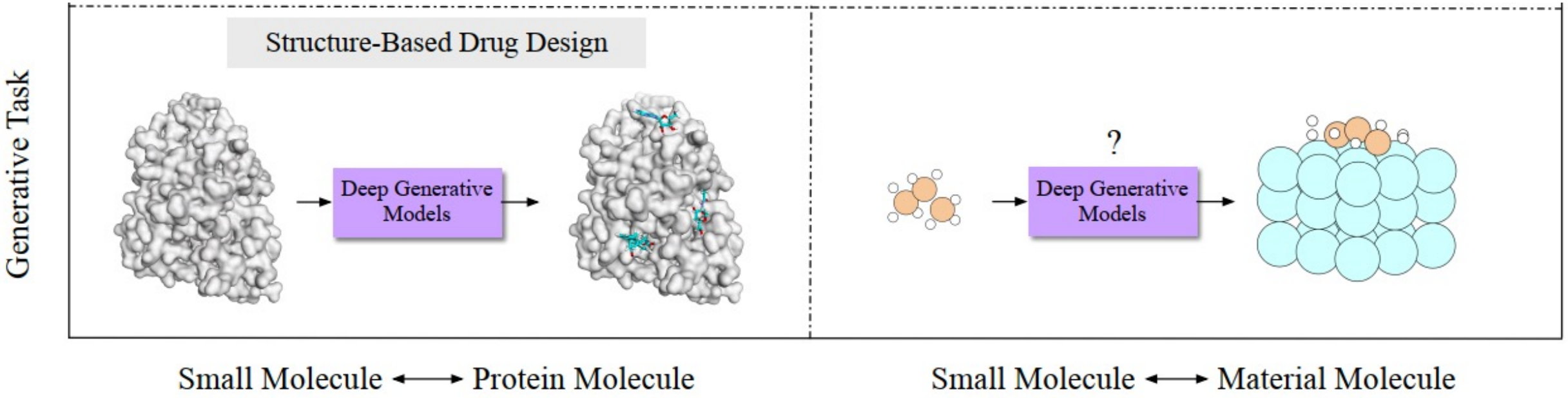
人工智能用于分子交互


Predictive Task
Generative Task
Protein-Ligand Binding Prediction
Deep Models
Binding Affinity
Energy, Force, and Position Prediction
Deep Models
Adsorption Energy
Per-atom Force
Relaxed Structure
Structure-Based Drug Design
Deep Generative Models
?
Deep Generative Models
Small Molecule ⟷ Protein Molecule
Small Molecule ⟷ Material Molecule

人工智能用于分子交互

- 预测任务:
  - 对于分子-蛋白质:结合(对接)预测
    - 结合位势预测
    - 结合亲和力预测
  - 对于分子-材料:
    - 预测吸附能(S2E)
    - 预测每原子力(S2F)
    - 预测松弛终态结构(IS2RS)
    - 预测松弛状态下的吸附能(IS2RE)

人工智能用于分子交互



- 生成任务
  - 对于分子-蛋白质:基于结构的药物设计
  - 对于分子-材料:仍未被探索,存在潜在机遇

Protein-Ligand Binding Prediction

- 任务设定
  - 分子对接简介：
    - 定义：给定蛋白质结构和配体分子图。
    - 目标：预测配体与蛋白质结合的原子位置。
  - 结合力预测简介：
    - 定义：预测配体与蛋白质的结合亲和力。

人工智能用于分子交互

- 对接过程：
  - 已知对接位置：在已知的袋（pocket）中预测结合位置。
  - 盲对接：没有对接位置的先验知识。
  - 性能指标：成功 = 高比例的正确预测。
- 结合力预测：
  - 结合亲和力的预测：配体与蛋白质结合的强度。
  - 表示结合与未结合状态的频率。
  - 输入可以是蛋白质-配体结合结构或蛋白质结构和配体的分子图

形式表示

- 配体结构表示：$\mathrm{M} = (A, E, C)$
  - $A = [\boldsymbol{a}_1, \cdots, \boldsymbol{a}_n]$：所有分子中的原子属性
  - $E = [\boldsymbol{e}_1, \cdots, \boldsymbol{e}_l]$：边特征，例如化学键类型和长度
  - $C = [\boldsymbol{c}_1, \cdots, \boldsymbol{c}_n] \in R^{3\times n}$：三维坐标矩阵
- 蛋白质/袋结构表示：$\mathrm{P} = (B, S)$
  - $B = [\boldsymbol{b}_1, \cdots, \boldsymbol{b}_m]$：节点特征，包括氨基酸或原子类型
  - $S = [\boldsymbol{s}_1, \cdots, \boldsymbol{s}_m] \in R^{3\times m}$：α-碳或结构中的原子坐标

形式表示

- 对接目标：采样对接姿势的分布
  - 表示为 $p_{\text{pose}}(C|B, S, A, E)$
- 方法：
  - 直接建模分布 $p_{\text{pose}}(C|B, S, A, E)$ 并采样
  - 预测 k 个配体构象的几何，使得 $f_{\text{pose}}(B, S, A, E) \rightarrow [C_1, \cdots, C_k]$
- 结合力预测目标：估计结合亲和力或提供排名
  - 表示为 q：$q \in \mathrm{R}$ 或 $q \in \mathrm{Z+}$ 或 $q \in [0, 1]$
- 任务表示为 $f_{\text{strength}}(A, E, C, B, S) \rightarrow q$
- 在结合强度预测中,可能不会给出配体或蛋白质/口袋的几何信息。

挑战

- 预测配体结合姿态：
  - 描述玻尔兹曼分布的全局模式。
  - 理想情况：稀疏支持度、生成模型复制。
- 难点：在大量可信的构型中找到能量最低的姿态。

- 配体对接 vs. 蛋白质结构预测：
  - 对接缺乏大量序列数据进行演化约束。
  - 蛋白质结构预测之前的成功源于数据可用性。
- 技术挑战：可访问训练数据有限（约 20k 样本）、清理复杂数据困难，影响质量保证。
- 训练稳健的亲和力预测器存在困难

任务中的对称性

① O(n)是由旋转和映射组成的n维正交群；
② SO(n)是一个只包含旋转的特殊正交群；
③ E(n)是由旋转、映射和平移组成的n维欧几里得群；
④ SE(n)是一个由旋转和平移组成的特殊的欧几里得群；
⑤ Lie李群是其元素构成可微流形的群。

- 对称性考虑：
  - 对接：$SE(3)$-等变任务，保持蛋白质-配体姿态的旋转和平移。
  - 形式表达：$p_{\text{pose}}(g \triangleright C | B, g \triangleright S, A, E) = p_{\text{pose}}(C | B, S, A, E)$ 和 $f_{\text{pose}}(B, g \triangleright S, A, E) = g \triangleright f_{\text{pose}}(B, S, A, E)$。
- 结合力预测：$SE(3)$-等变任务，对系统的旋转和平移不会影响预测。
- 形式表达：$f_{\text{strength}}(A, E, g \triangleright C, B, g \triangleright S) = f_{\text{strength}}(A, E, C, B, S)$。
- $\triangleright$：群操作（类比函数操作。，f∘x ⇔ f(x)）
- g: SE(3)上的平移旋转 操作

Drug Discovery is a Long and Expensive Process

- It takes more than 10 years and $2.5B to develop a new drug.

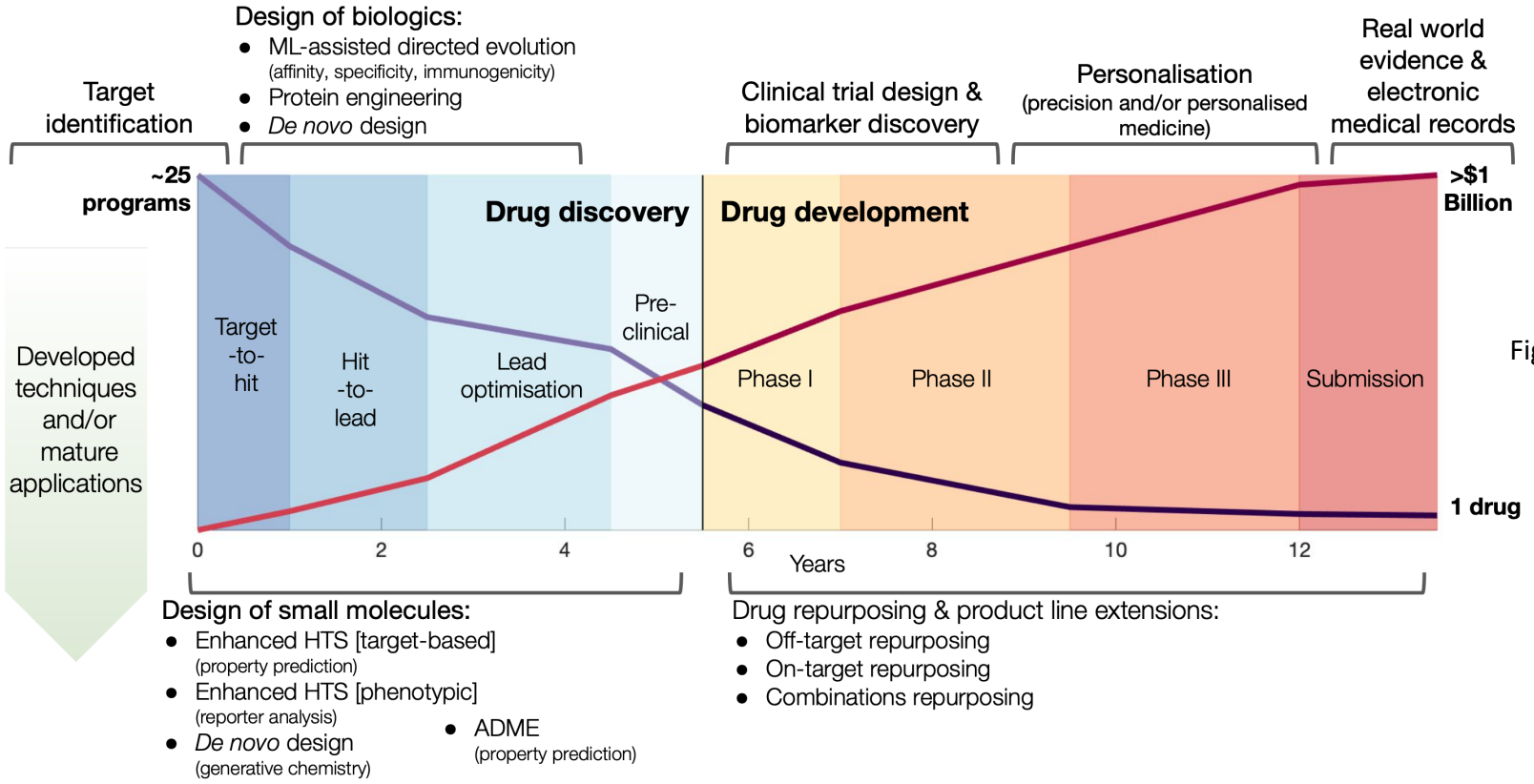
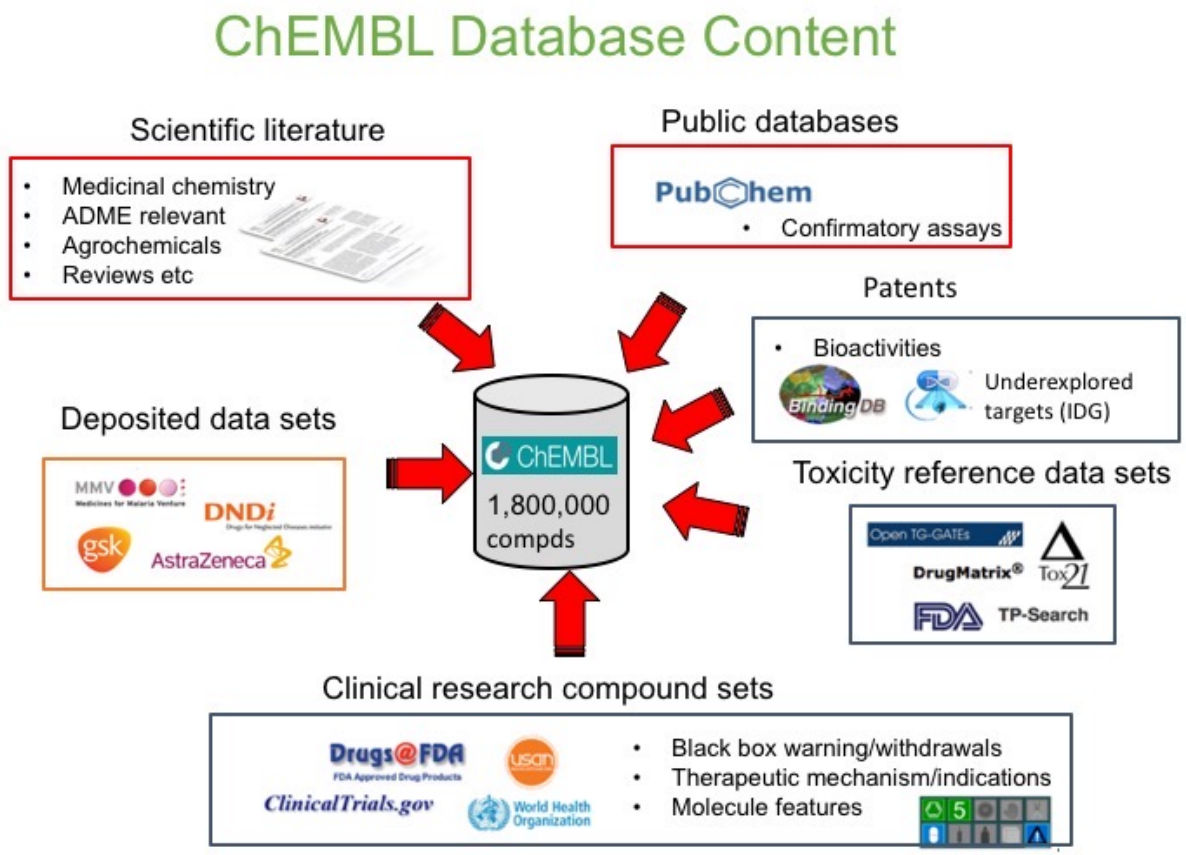


Figure from Gaudelet et al.

Big Opportunity for Artificial Intelligence

- A huge amount of data is generated in the biomedical domain



Many Data are Graph-Structured

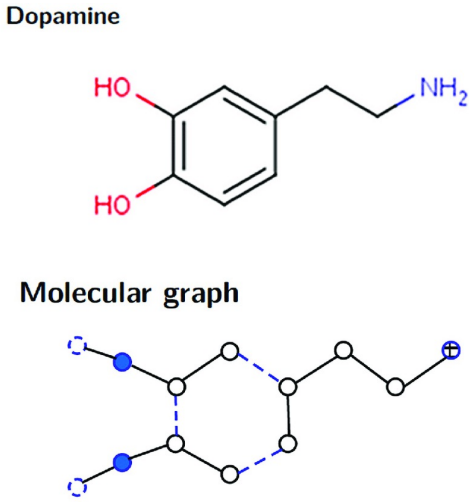


Fig. 1: Small Molecules

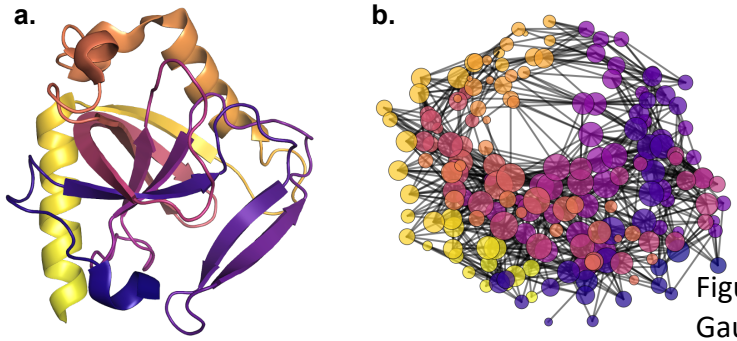


Figure from Gaudelet et al.

Figure 7: Illustration of **a.** a protein (PDB accession: 3EIY) and **b.** its graph representation derived based on intramolecular distance with cut-off threshold set at 10Å.

Fig. 2: Proteins

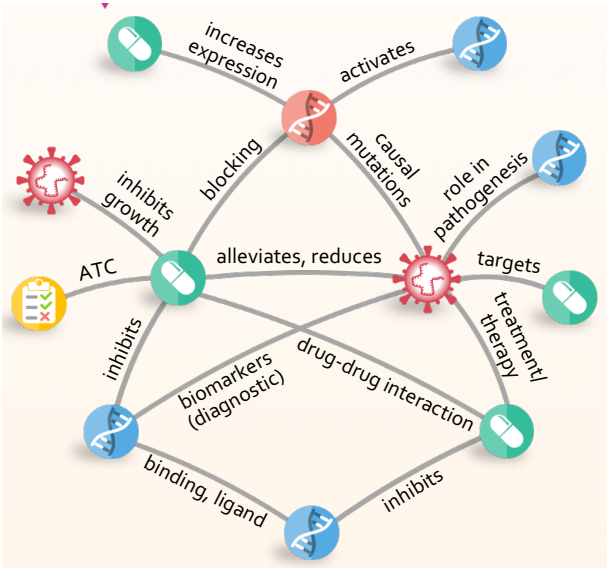
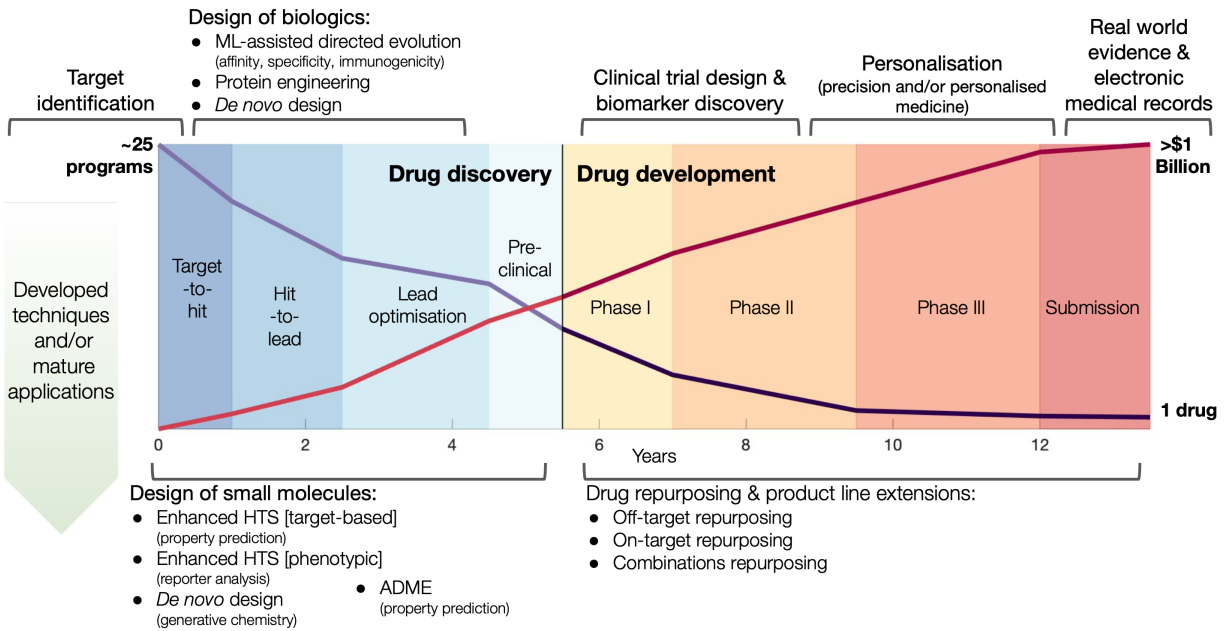


Figure from Zeng et al. 201

Fig. 3: Biomedical Knowledge Graphs

Research Problems (1): De Novo Drug Discovery



Molecule Property Prediction

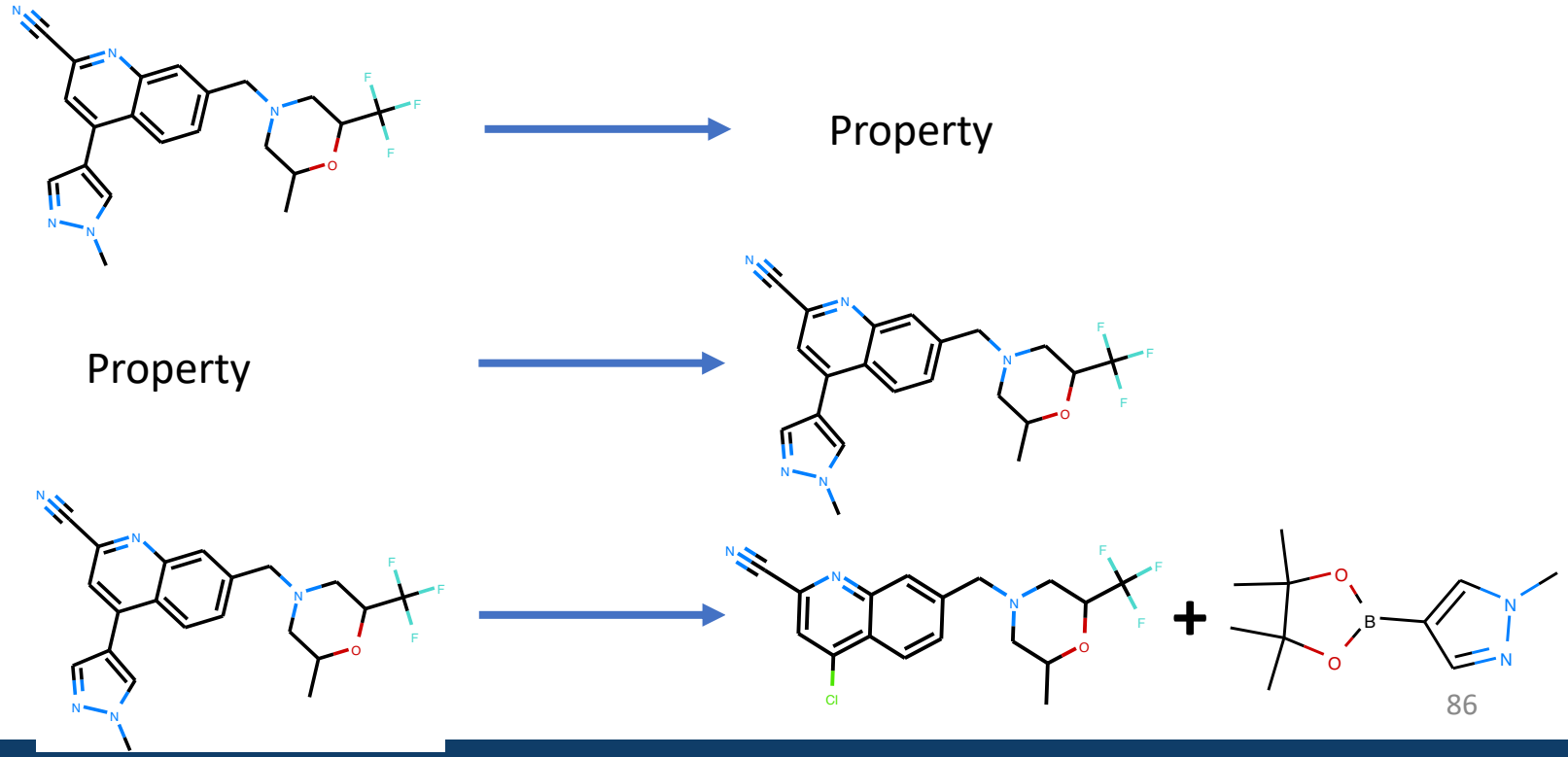
分子性质预测

De Novo Molecule Design and Optimization

De Novo 分子设计与优化

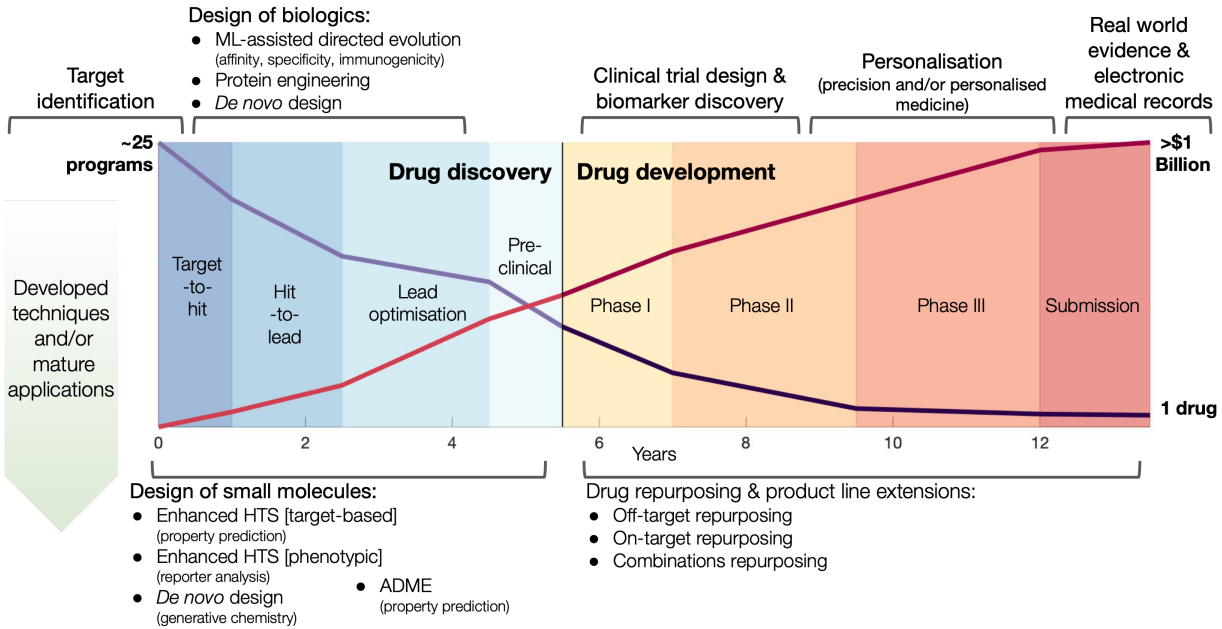
Retrosynthesis Prediction

回溯合成预测



Research Problems (2): Drug Repurposing

- Predict the **links** between **drugs** and **diseases** on biomedical knowledge graphs
  - **Reasoning** on biomedical knowledge graphs



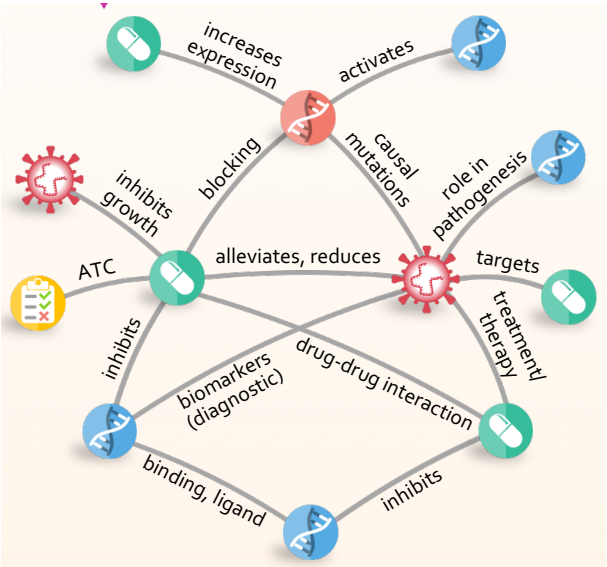
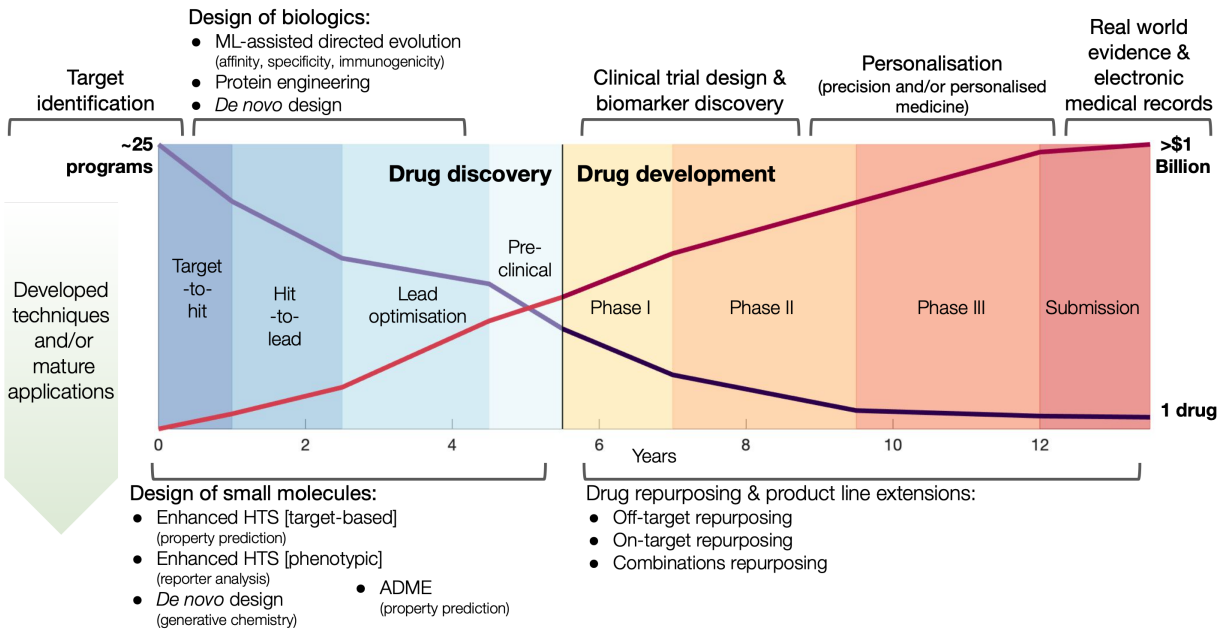
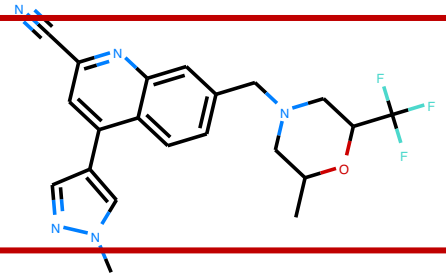


Fig. 3: Biomedical Knowledge Graphs

Research Problems (1): De Novo Drug Discovery



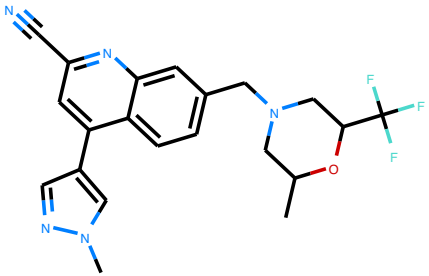
Molecule Property Prediction



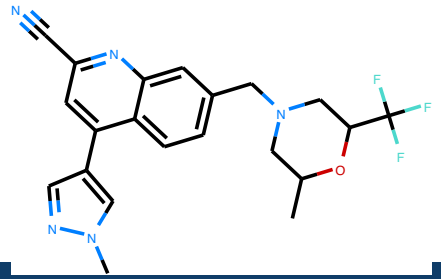
→ Property

De Novo Molecule Design and Optimization

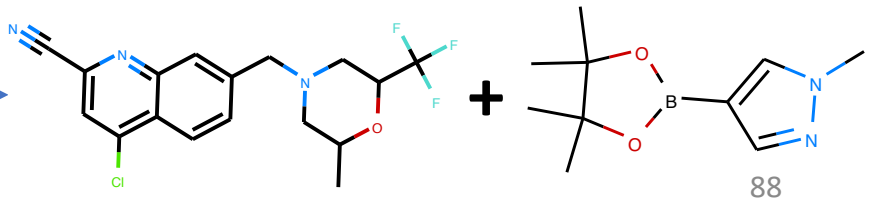
Property →



Retrosynthesis Prediction



→



Graph Neural Networks

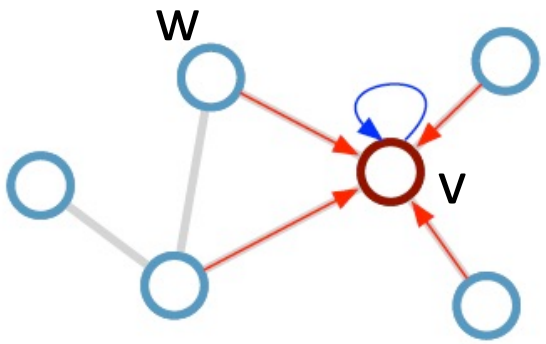
- Techniques for learning node/graph representations
  - Graph convolutional Networks (Kipf et al. 2016)
  - Graph attention networks (Veličković et al. 2017)
- Neural Message Passing (Gilmer et al. 2017)

MESSAGE PASSING: $M_k(h_v^k, h_w^k, e_{vw})$

AGGREGATE : $m_v^{k+1} = \text{AGGREGATE}\{M_k(h_v^k, h_w^k, e_{vw}) : w \in N(v)\}$

COMBINE : $h_v^{k+1} = \text{COMBINE}(h_v^k, m_v^{k+1})$

READOUT: $g = \text{READOUT}\{h_v^K : v \in G\}$



InfoGraph: Unsupervised and Semi-supervised Whole-Graph Representation Learning (Sun et al. ICLR'20)

- 基于图神经网络的监督方法需要大量标记数据进行训练。
- 在药物发现领域，标记数据的数量非常有限。
  - 大量未标记的数据（分子）是可用的。
- 如何以无监督或半监督的方式有效地学习整个图表示呢？

Fanyun Sun, Jordan Hoffman, Vikas Verma and Jian Tang. **InfoGraph: Unsupervised and Semi-supervised Graph-Level Representation Learning via Mutual Information Maximization**. ICLR'20.

InfoGraph: Unsupervised Whole-Graph Representation Learning (Sun et al. ICLR'20)

- 最大化整个图表示 $H_\phi(G)$ 与所有子结构 h_ϕ^i 表示之间的互信息。
  - 确保图表示捕获了所有子结构中的主要信息
- K层图神经网络：

$$h_v^{(k)} = \text{COMBINE}^{(k)}\left(h_v^{(k-1)}, \text{AGGREGATE}^{(k)}\left(\left\{\left(h_v^{(k-1)}, h_u^{(k-1)}, e_{uv}\right) : u \in \mathcal{N}(v)\right\}\right)\right)$$

- 总结每个节点i处的局部结构信息：

$$h_\phi^i = \text{CONCAT}(\{h_i^{(k)}\}_{k=1}^K)$$

- 总结整个图的信息：

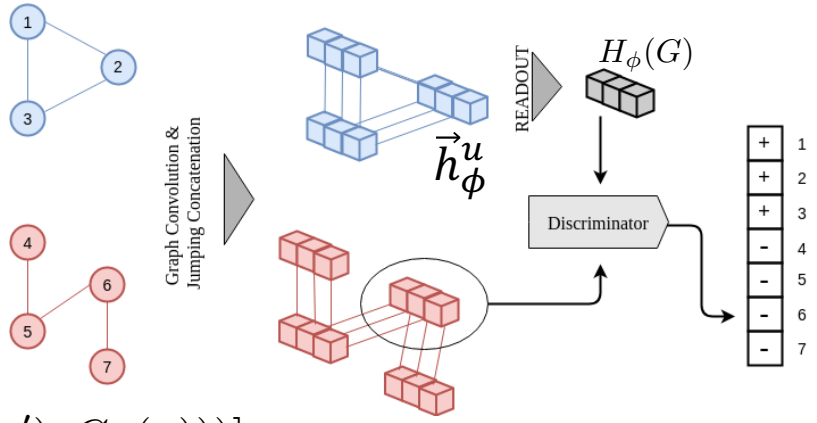
$$H_\phi(G) = \text{READOUT}(\{h_\phi^i\}_{i=1}^N)$$

Fanyun Sun, Jordan Hoffman, Vikas Verma and Jian Tang. **InfoGraph: Unsupervised and Semi-supervised Graph-Level Representation Learning via Mutual Information Maximization**. ICLR'20.

InfoGraph: Unsupervised Whole-Graph Representation Learning

- 最大化整个图表示 $H_\phi(G)$ 与所有子结构 $\vec{h}_\phi^u$ 表示之间的互信息。

$$\hat{\phi}, \hat{\psi} = \arg\max_{\phi,\psi} \sum_{G \in \mathbf{G}} \frac{1}{|G|} \sum_{u \in G} I_{\phi,\psi}(\vec{h}_\phi^u; H_\phi(G)).$$



- 我们用 Jensen-Shannon MI estimator:

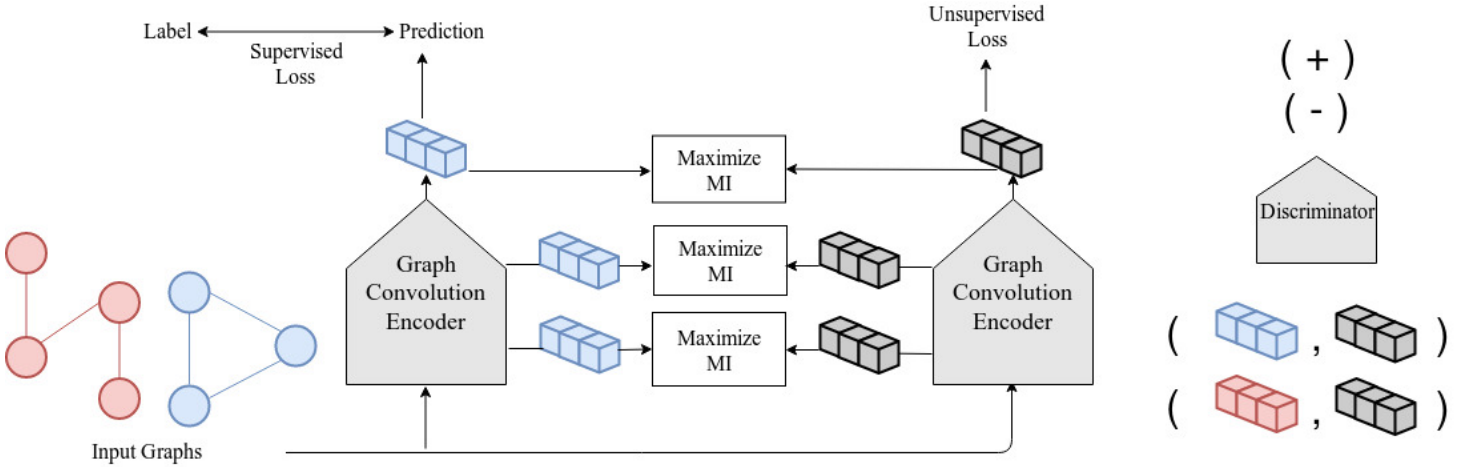
$$I_{\phi,\psi}(h_\phi^i(G); H_\phi(G)) :=$$
$$\mathbb{E}_{\mathbb{P}}[-\mathrm{sp}(-T_{\phi,\psi}(\vec{h}_\phi^i(x), H_\phi(x)))] - \mathbb{E}_{\mathbb{P}\times\tilde{\mathbb{P}}}[\mathrm{sp}(T_{\phi,\psi}(\vec{h}_\phi^i(x'), G_\phi(x)))]$$

- Where x is an input sample, x' is a negative graph sample, $\mathrm{sp}(z) = \log(1 + e^z)$, $T(\,,\,)$ is a neural network

InfoGraph*: Semi-supervised Graph Representation Learning

- 两种不同的编码器用于监督和无监督任务。
- 在所有级别（或层次）上最大化由这两个编码器学习的表示之间的互信息。

$$\mathrm{L}_{\text{total}} = \sum_{\mathrm{i}=1}^{|\mathbb{G}^L|} \mathrm{L}_{\text{supervised}}(\mathrm{y}_\phi(\mathrm{G_i}), \mathrm{o_i}) + \sum_{j=1}^{|\mathbb{G}^L|+|\mathbb{G}^U|} \mathrm{L}_{\text{unsupervised}}(h_\varphi(G_j); H_\varphi(G_j)) - \lambda \sum_{j=1}^{|\mathbb{G}^L|+|\mathbb{G}^U|} \frac{1}{|G_j|} \sum_{k=1}^{K} I(H_\phi^k(G_j); H_\varphi^k(G_j))$$



Results on Graph Classification and Regression

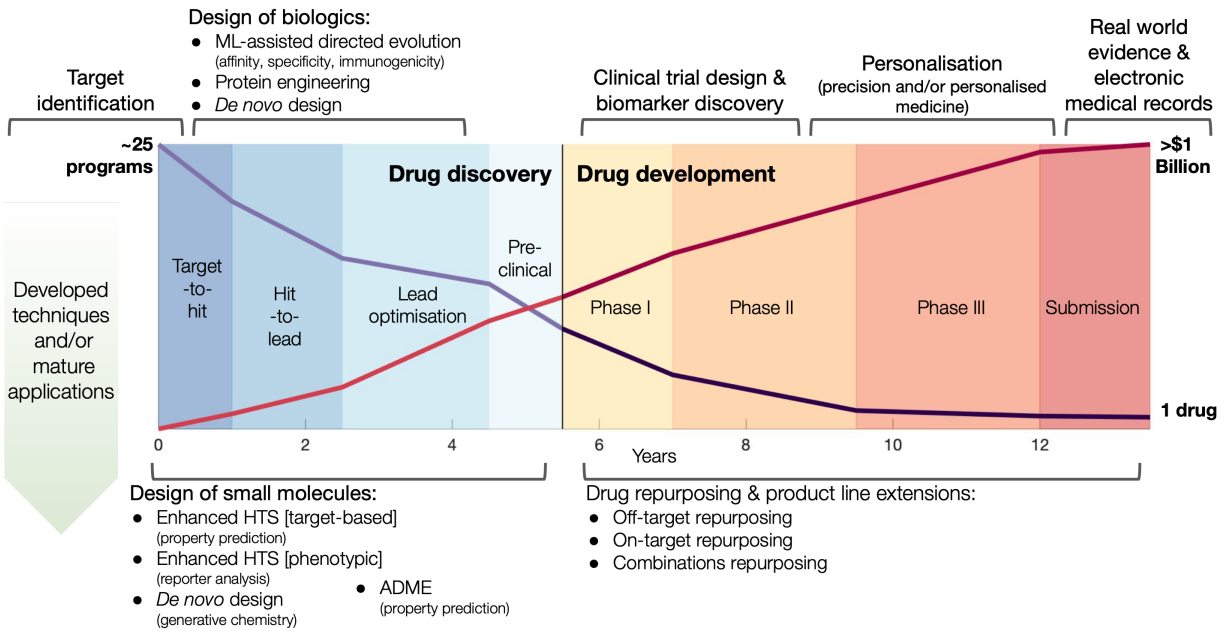
Dataset	MUTAG	PTC-MR	RDT-B	RDT-M5K	IMDB-B	IMDB-M
(No. Graphs)	188	344	2000	4999	1000	1500
(No. classes)	2	2	2	5	2	3
(Avg. Graph Size)	17.93	14.29	429.63	508.52	19.77	13.00
Graph Kernels						
RW [14]	83.72 ± 1.50	57.85 ± 1.30	OMR	OMR	50.68 ± 0.26	34.65 ± 0.19
SP [3]	85.22 ± 2.43	58.24 ± 2.44	64.11 ± 0.14	39.55 ± 0.22	55.60 ± 0.22	37.99 ± 0.30
GK [55]	81.66 ± 2.11	57.26 ± 1.41	77.34 ± 0.18	41.01 ± 0.17	65.87 ± 0.98	43.89 ± 0.38
WL [54]	80.72 ± 3.00	57.97 ± 0.49	68.82 ± 0.41	46.06 ± 0.21	72.30 ± 3.44	46.95 ± 0.46
DGK [68]	87.44 ± 2.72	60.08 ± 2.55	78.04 ± 0.39	41.27 ± 0.18	66.96 ± 0.56	44.55 ± 0.52
MLG [28]	87.94 ± 1.61	$\mathbf{63.26 \pm 1.48}$	> 1 Day	> 1 Day	66.55 ± 0.25	41.17 ± 0.03
Other Unsupervised Methods						
node2vec [17]	72.63 ± 10.20	58.58 ± 8.00	-	-	-	-
sub2vec [1]	61.05 ± 15.80	59.99 ± 6.38	71.48 ± 0.41	36.68 ± 0.42	55.26 ± 1.54	36.67 ± 0.83
graph2vec [38]	83.15 ± 9.25	60.17 ± 6.86	75.78 ± 1.03	47.86 ± 0.26	71.1 ± 0.54	$\mathbf{50.44 \pm 0.87}$
InfoGraph	$\mathbf{89.01 \pm 1.13}$	61.65 ± 1.43	$\mathbf{82.50 \pm 1.42}$	$\mathbf{53.46 \pm 1.03}$	$\mathbf{73.03 \pm 0.87}$	49.69 ± 0.53

Table 1: Graph classification accuracy with unsupervised methods

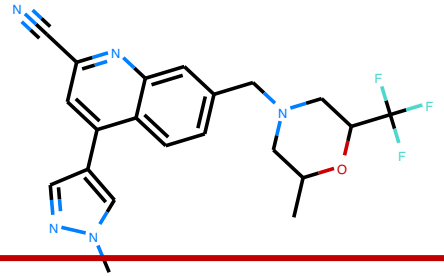
Target	Mu (0)	Alpha (1)	HOMO (2)	LUMO (3)	Gap (4)	R2 (5)	ZPVE(6)	U0 (7)	U (8)	H (9)	G(10)	Cv (11)
MAE	0.3201	0.5792	0.0060	0.0062	0.0091	10.0469	0.0007	0.3204	0.2934	0.2722	0.2948	0.2368
Semi-Supervised	Error Ratio											
Mean-Teachers	1.09	1.00	**0.99**	1.00	**0.97**	0.52	0.77	1.16	0.93	0.79	0.86	0.86
InfoGraph	1.02	0.97	1.02	**0.99**	1.01	0.71	0.96	0.85	0.93	0.93	0.99	1.00
InfoGraph*	**0.99**	**0.94**	**0.99**	**0.99**	0.98	**0.49**	**0.52**	**0.44**	**0.58**	**0.57**	**0.54**	**0.83**

Table 2: Results of semi-supervised experiments on QM9 data set.

Research Problems (1): De Novo Drug Discovery



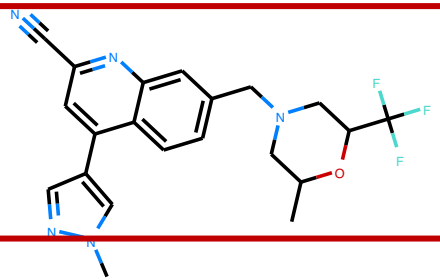
Molecule Property Prediction



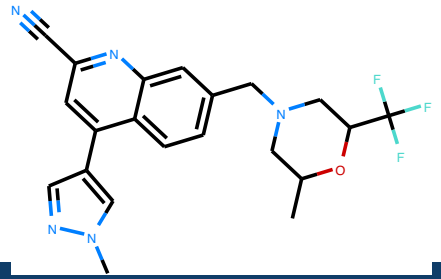
Property

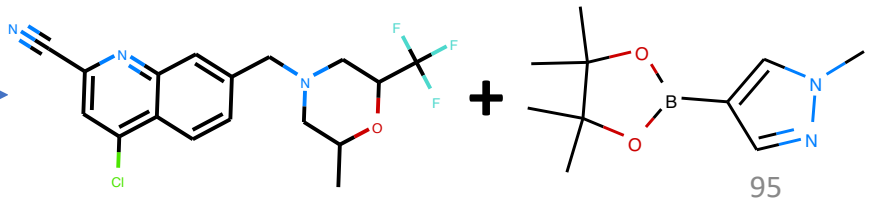
De Novo Molecule Design and Optimization

Property



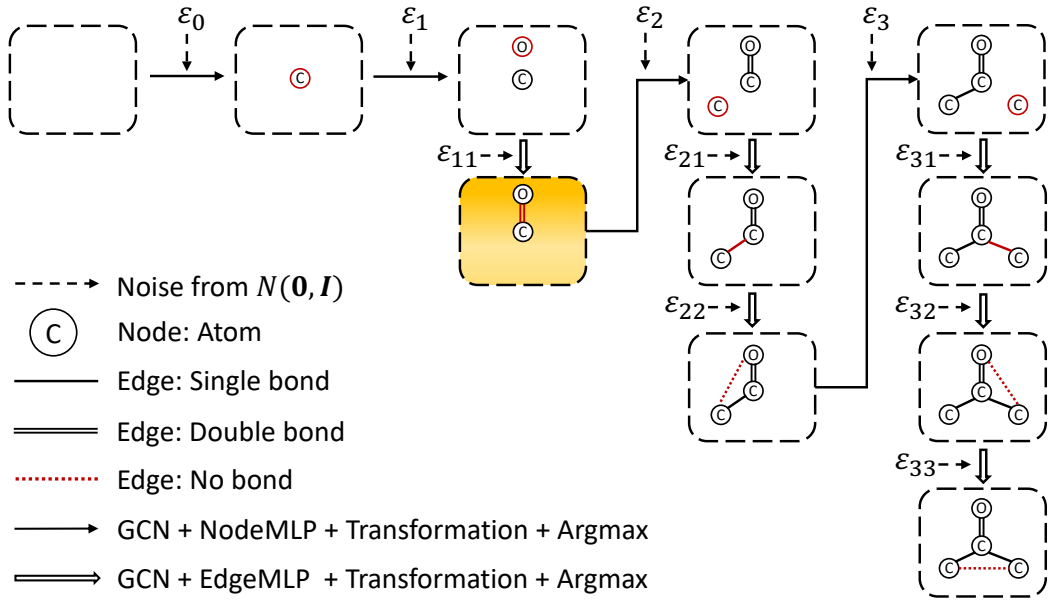
Retrosynthesis Prediction





GraphAF: an Autoregressive Flow for Molecular Graph Generation (Shi & Xu ICLR'20)

- 将图生成视为一个顺序决策过程
  - 在每一步中，生成一个新的原子
  - 确定新原子与已有原子之间的化学键



(a) Sampling Framework

Chence Shi, Minkai Xu, Zhaocheng Zhu, Weinan Zhang, Ming Zhang, and Jian Tang.
"GraphAF: a Flow-based Autoregressive Model for Molecular Graph Generation." ICLR'20.

Normalizing Flows (Dinh et al. 2016)

- 定义了从基础分布（例如高斯分布）到观测空间的可逆映射。

$$f\colon \mathcal{Z} \to \mathcal{X}$$

Data space $\mathcal{X}$ Latent space $\mathcal{Z}$

Inference

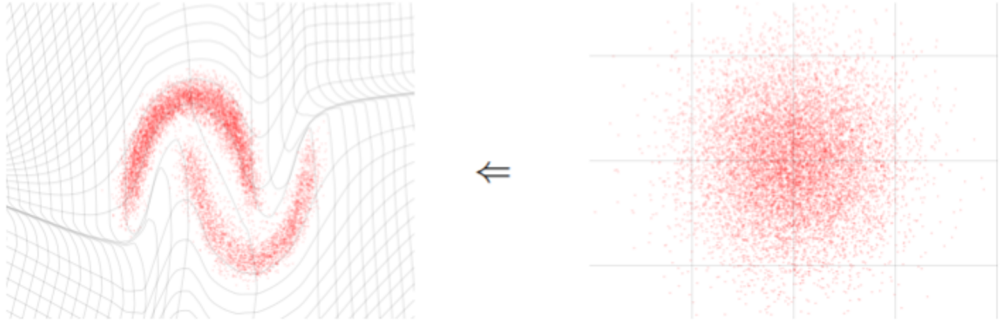
$x \sim \hat{p}_x$

$\mathrm{z} = f(x)$

Generation

$z \sim p_z$

$x = f(z)$



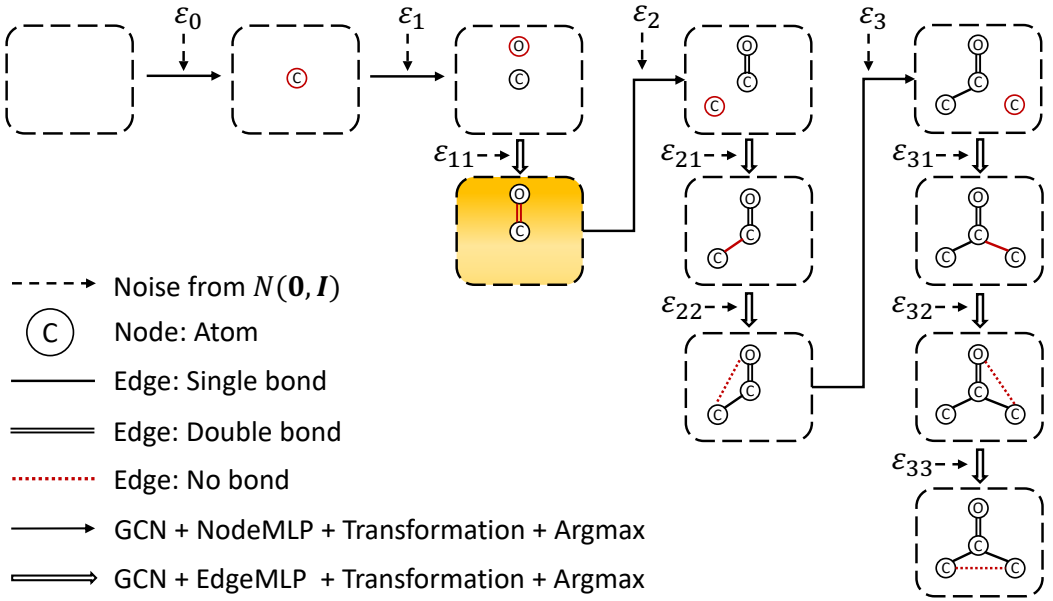
Change-of-Variables

$$p_{\mathcal{X}}(x) = p_{\mathcal{Z}}\left(f_\theta^{-1}(x)\right)\left|\det \frac{\partial f_\theta^{-1}(x)}{\partial x}\right|$$

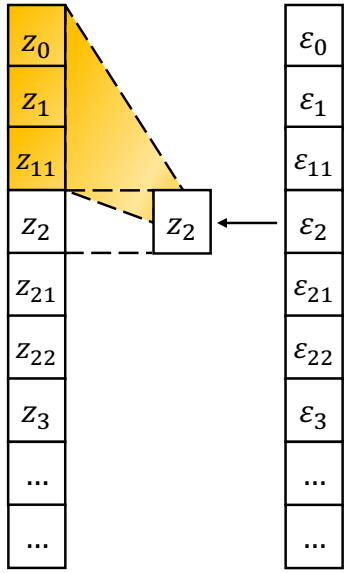
Density estimation using Real NVP (2016)

GraphAF: an Autoregressive Flow for Molecular Graph Generation

- 按BFS顺序对图进行遍历
  - 将每个图形转换为节点和边的序列
- 为基础分布（高斯分布）到观测（图节点和边序列）定义了可逆映射



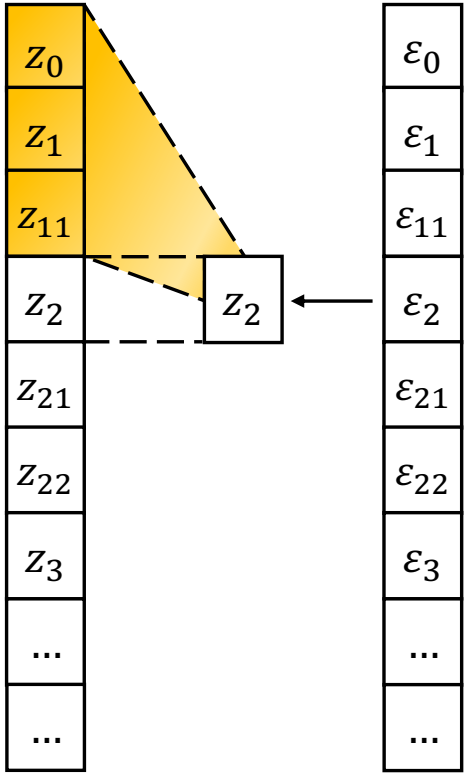
(a) Sampling Framework



(b) Autoregressive Flow

Advantages of GraphAF

- 具有强大的数据密度建模能力
  - 得益于正规化流（normalizing flow）框架
- 训练（从 z 到 ϵ）：并行
  - 高效的训练过程
- 采样（从 ϵ 到 z）：顺序
  - 有效捕获图结构
  - 能够整合化学规则

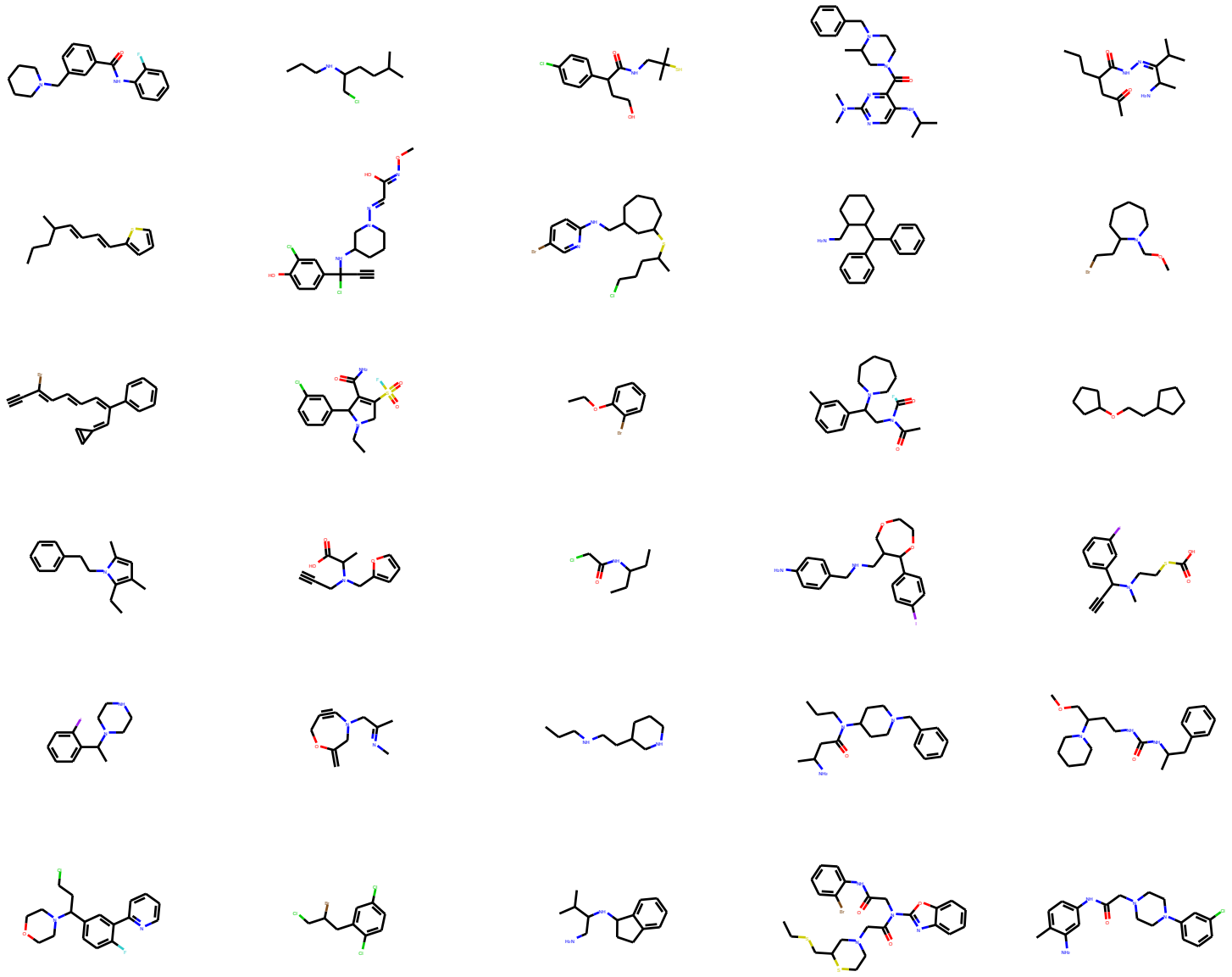


(b) Autoregressive Flow

Molecule Generation

- Training Data: ZINC250K
  - 250K drug-like molecules with a maximum atom number of 38
  - 9 atom types and 3 edge types

Method	Validity	Validity w/o check	Uniqueness	Novelty	Reconstruction
JT-VAE	100%	—	100%‡	100%‡	76.7%
GCPN	100%	20%†	99.97%‡	100%‡	—
MRNN	100%	65%	99.89%	100%	—
GraphNVP	42.60%	—	94.80%	100%	100%
GraphAF	100%	68%	99.10%	100%	100%



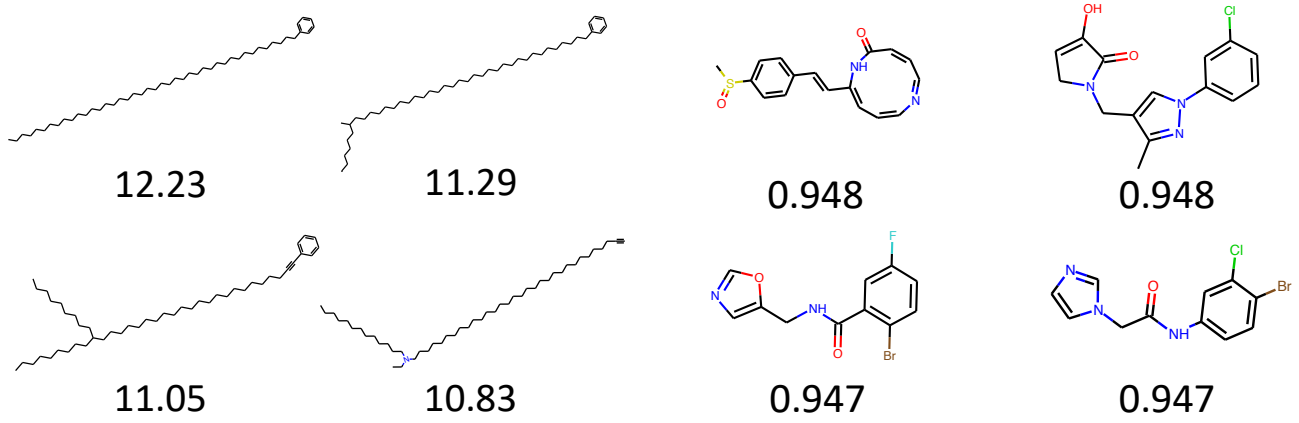
Goal-Directed Molecule Generation with Reinforcement Learning

- 微调生成策略，使用强化学习来优化生成分子的特性
- **状态**：当前子图 G_i
- 动作：生成一个新的原子(i.e. $\mathrm{p}(X_i|G_i)$)或一个新的化学键 $(\mathrm{p}(A_{ij}|G_i, X_i, A_{i,1:j-1}))$.
- 奖励设计：分子的特性（最终奖励）和化学有效性（中间和最终奖励）

Molecule Optimization

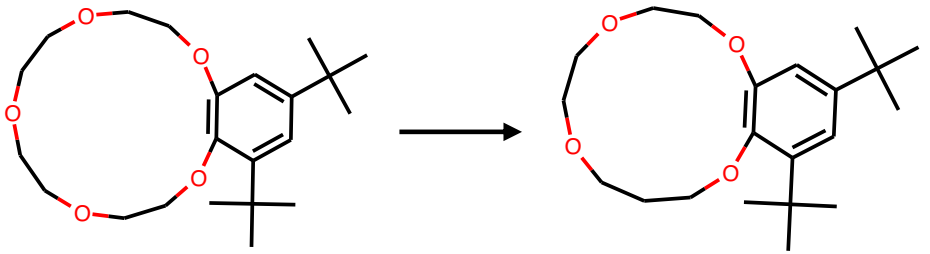
- Properties
  - Penalized logP
  - QED (druglikeness)

Method	Penalized logP				QED			
	1st	2nd	3rd	Validity	1st	2nd	3rd	Validity
ZINC (Dataset)	4.52	4.30	4.23	100.0%	0.948	0.948	0.948	100.0%
JT-VAE (Jin et al., 2018)	5.30	4.93	4.49	100.0%	0.925	0.911	0.910	100.0%
GCPN (You et al., 2018a)	7.98	7.85	7.80	100.0%	**0.948**	0.947	0.946	100.0%
MRNN[1] (Popova et al., 2019)	8.63	6.08	4.73	100.0%	0.844	0.796	0.736	100.0%
GraphAF	**12.23**	**11.29**	**11.05**	100.0%	**0.948**	**0.948**	**0.947**	100.0%

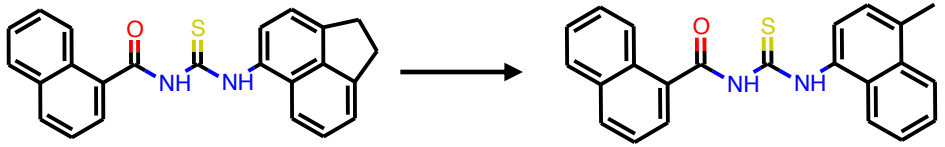


(a) Penalized logP optimization (b) QED optimization

Constrained Optimization



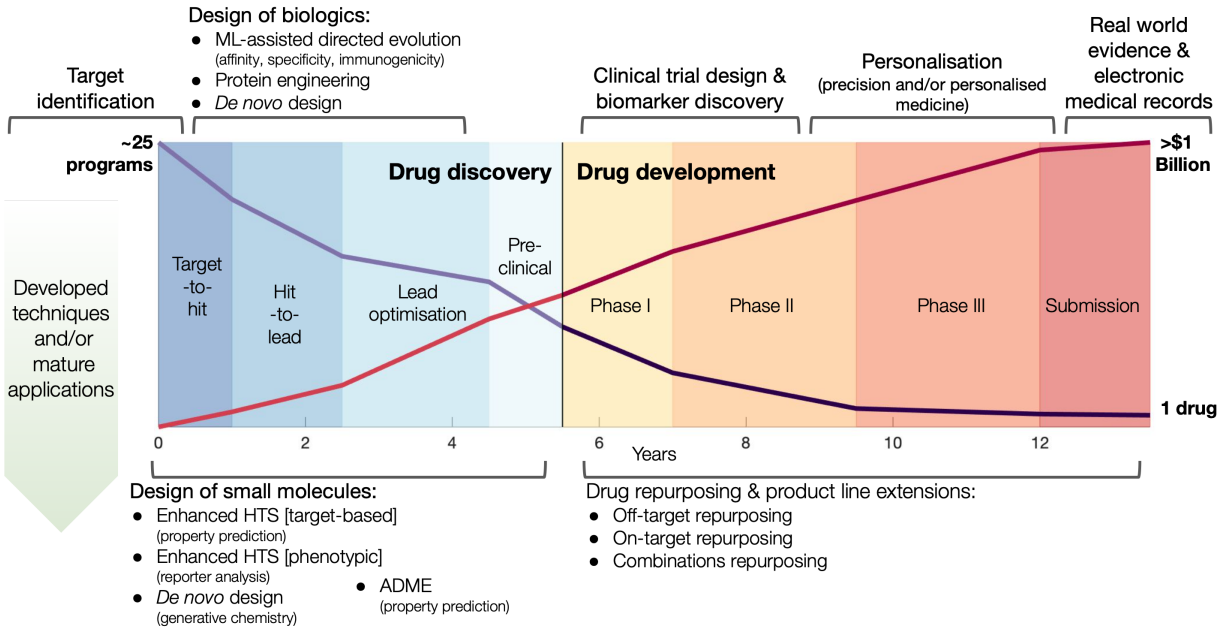
-30.21 -22.87



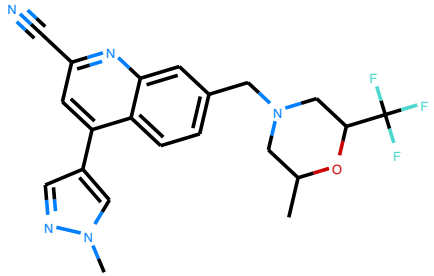
-14.32 3.58

(c) Constrained optimization

Research Problems (1): De Novo Drug Discovery



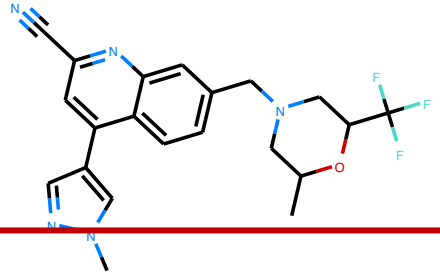
Molecule Property Prediction



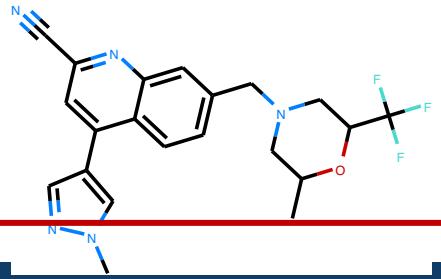
Property

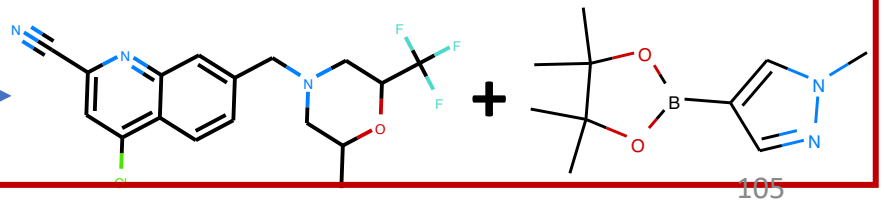
De Novo Molecule Design and Optimization

Property



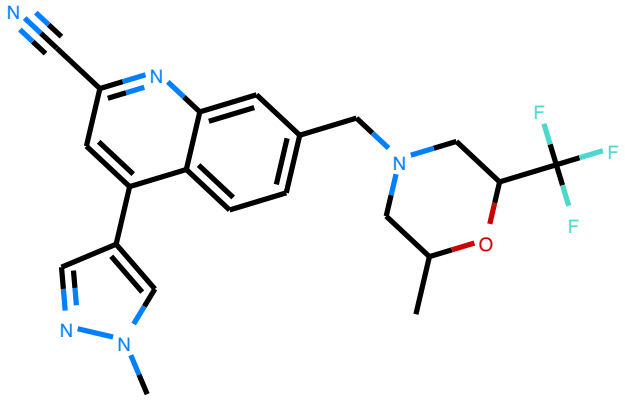
Retrosynthesis Prediction





Retrosynthesis Prediction

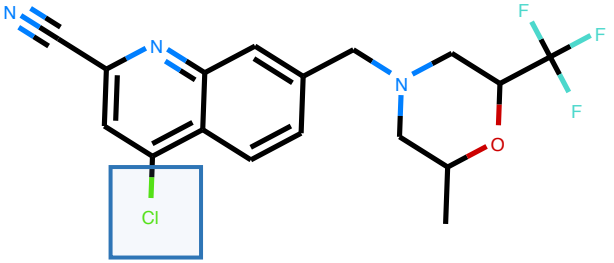
- 设计了分子结构后，如何合成它呢？
- 回溯合成规划/预测
  - 确定一组反应物以合成目标分子



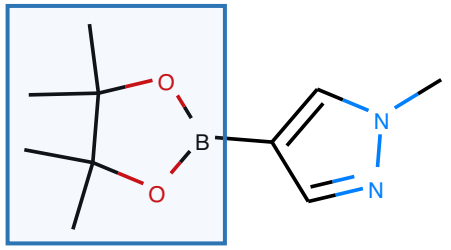
Product (Given)

Predict Reactants

Reaction Type
(optional)



Reactant A



Reactant B

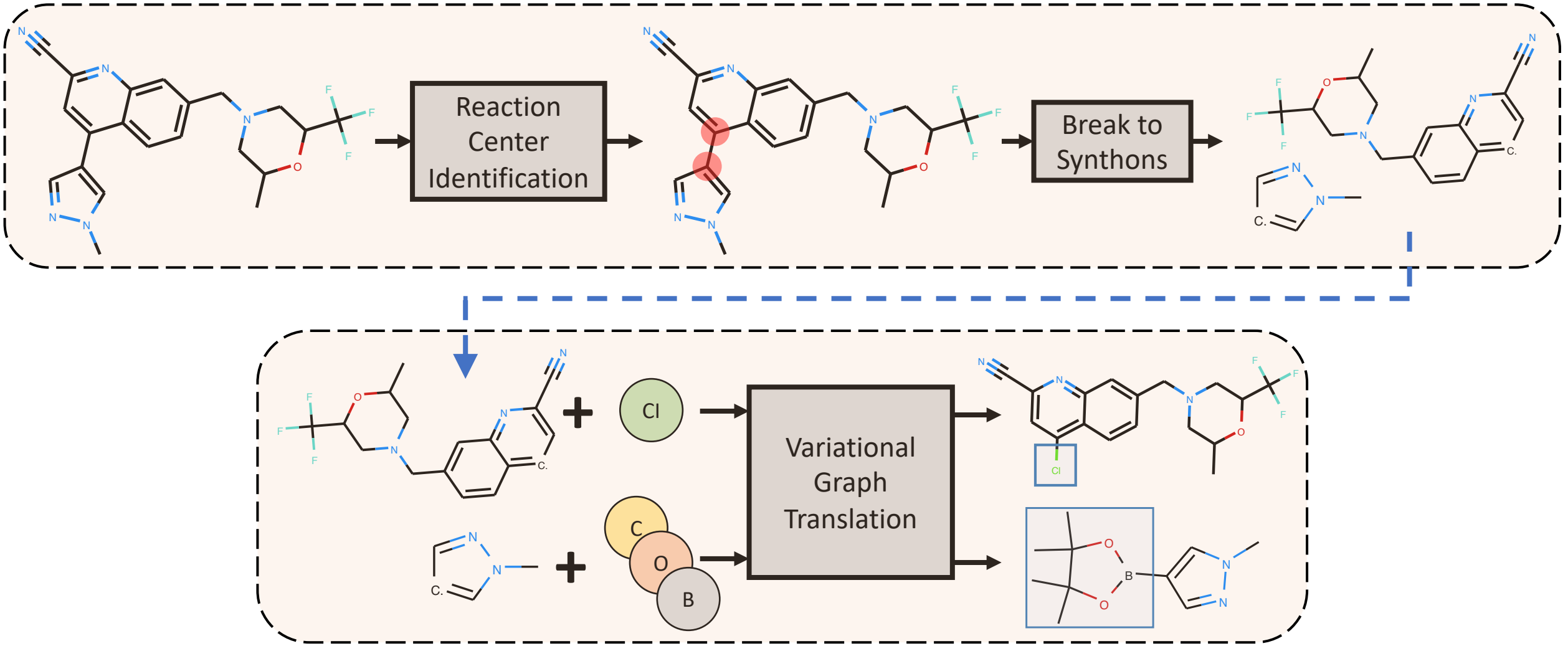
106

A Graph to Graphs Framework for Retrosynthesis Prediction (Shi et al. 2020)

- 每个分子被表示为一个分子图
- 将问题描述为一个图（产物分子）到一组图（反应物）的转换
- 整个框架分为两个阶段
  - 反应中心识别
  - 图翻译

Chence Shi, Minkai Xu, Hongyu Guo, Ming Zhang and Jian Tang. **A Graph to Graphs Framework for Retrosynthesis Prediction**. ICML, 2020.

The G2Gs Framework (Shi et al. 2020)



Shi et al., 2020, A Graph to Graphs Framework for Retrosynthesis Prediction

Research Problems (2): Drug Repurposing

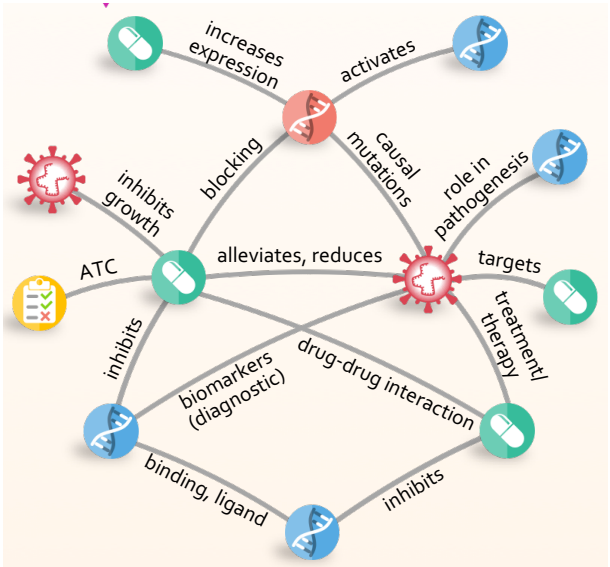
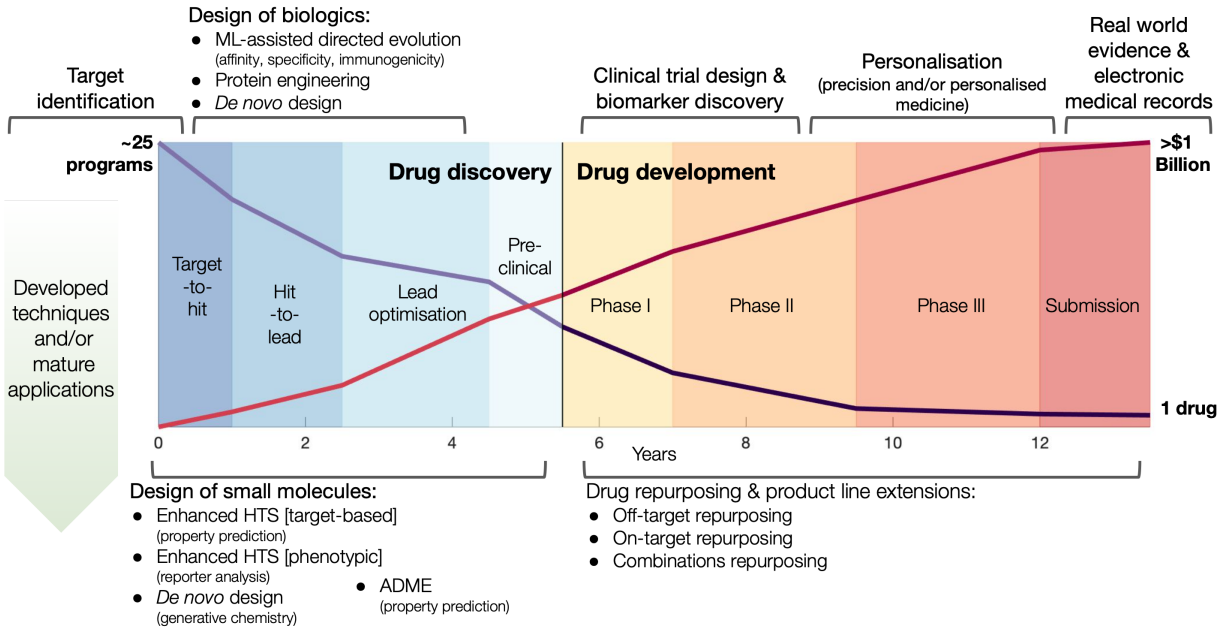


Fig. 3: Biomedical Knowledge Graphs



- 在生物医学知识图谱上预测药物与疾病之间的联系，并进行推理。

Reasoning on Knowledge Graphs

- Reasoning in continuous space---knowledge graph embedding-based methods: **RotatE** (Sun et al. ICLR' 2019)
- Symbolic logic reasoning: **pLogicNet** (Qu et al. NeurIPS'2019)
- Learning logic rules for reasoning on knowledge graphs: **RNNLogic** (Qu et al. ICLR'2021)

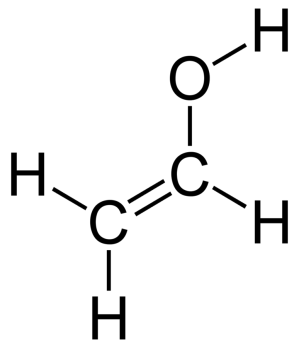
- Sun, Zhiqing, Zhihong Deng, Jian-Yun Nie, and **Jian Tang** et al. "Rotate: Knowledge graph embedding by relational rotation in complex space." *ICLR'2019*.
- Qu, Meng, and **Jian Tang**. "Probabilistic logic neural networks for reasoning." *Advances in Neural Information Processing Systems*. 2019.
- Qu, Meng*, Chen, Junkun*, Xhonneux Louis-Pascal, Bengio Yoshua, and **Tang, Jian**. "RNNLogic: Learning Logic Rules for Reasoning on Knowledge Graphs." ICLR'2021.

Next Step: Going Beyond 2D Graphs to 3D Structures

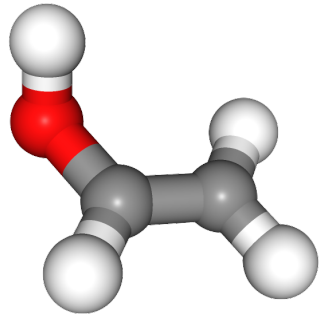
- 分子的更加自然和内在的表征：3D构象
  - 确定其生物和物理活性
  - 例如，电荷分布、空间位阻限制和与其他分子的相互作用

C1CO

1D SMILES



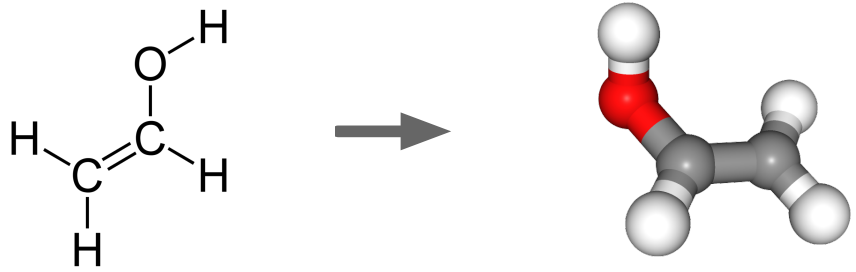
2D Graph



3D Conformation

Conformation Prediction

- 对于大多数分子，它们的三维结构并不可用
  - 如何预测有效和稳定的构象？
  - 每个原子被表示为其三维坐标



Our Solution (Xu and Luo et al. 2020)

- 基于正规化流的灵活生成模型 $p_\theta(\boldsymbol{R}|\mathcal{G})$
  - 将成对距离 d 视为中间变量
  - 首先基于 $\mathcal{G}$ 生成距离 d，即$p_\theta(\boldsymbol{d}|\mathcal{G})$
  - 然后基于 d 和 $\mathcal{G}$ 生成构象（conformations），即$p_\theta(\boldsymbol{R}|\boldsymbol{d},\mathcal{G})$

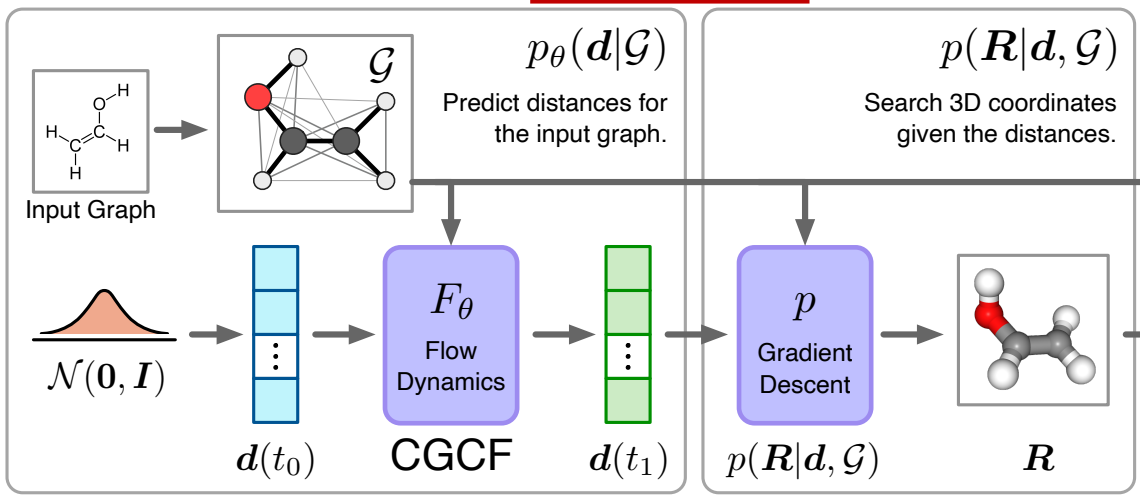
$$p_\theta(\boldsymbol{R}|\mathcal{G}) = \int p(\boldsymbol{R}|\boldsymbol{d},\mathcal{G}) \cdot p_\theta(\boldsymbol{d}|\mathcal{G})\,\mathrm{d}\boldsymbol{d}$$

Minkai Xu*, Shitong Luo*, Yoshua Bengio, Jian Peng, Jian Tang. **Learning Neural Generative Dynamics for Molecular Conformation Generation**. ICLR'21

Distance Geometry Generation $p_\theta(\boldsymbol{d}|\mathcal{G})$

- 条件图连续流（CGCF）
  - 在分子图 $\mathcal{G}$ 的条件下，定义了一个基础分布与成对原子距离 d 之间的可逆映射
  - 利用神经常微分方程（ODEs）定义了距离 d 的连续动态：

$$\boldsymbol{d} = F_\theta(\boldsymbol{d}(t_0), \mathcal{G}) = \boldsymbol{d}(t_0) + \int_{t_0}^{t_1} f_\theta(\boldsymbol{d}(t), t; \mathcal{G}) \mathrm{d}t, \quad \boldsymbol{d}(t_0) \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$$

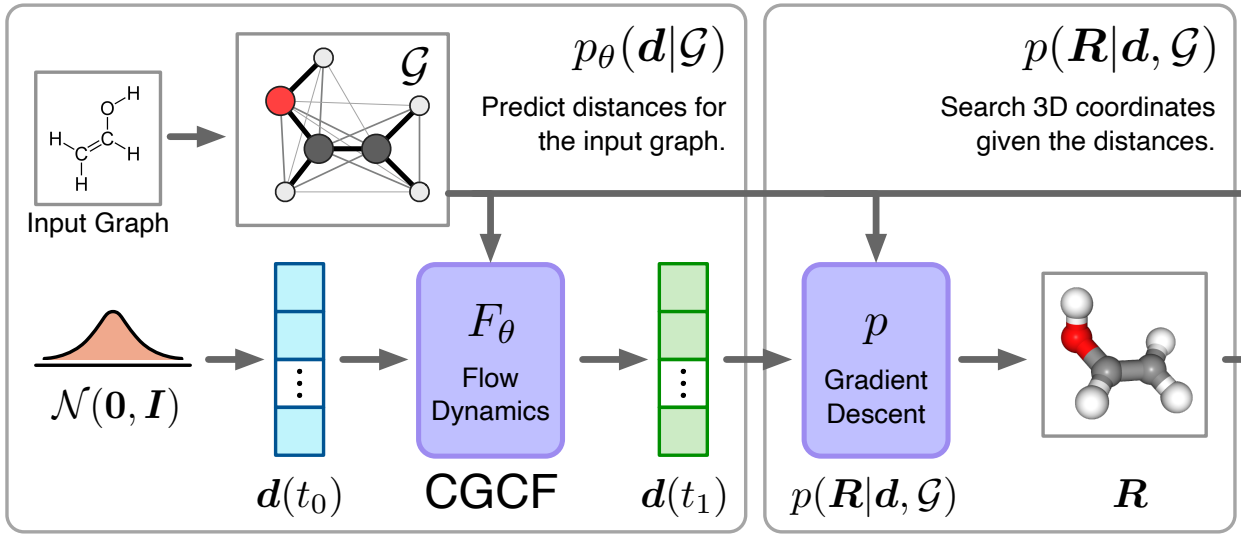


Conformation Prediction $p(\boldsymbol{R}|\boldsymbol{d},\mathcal{G})$

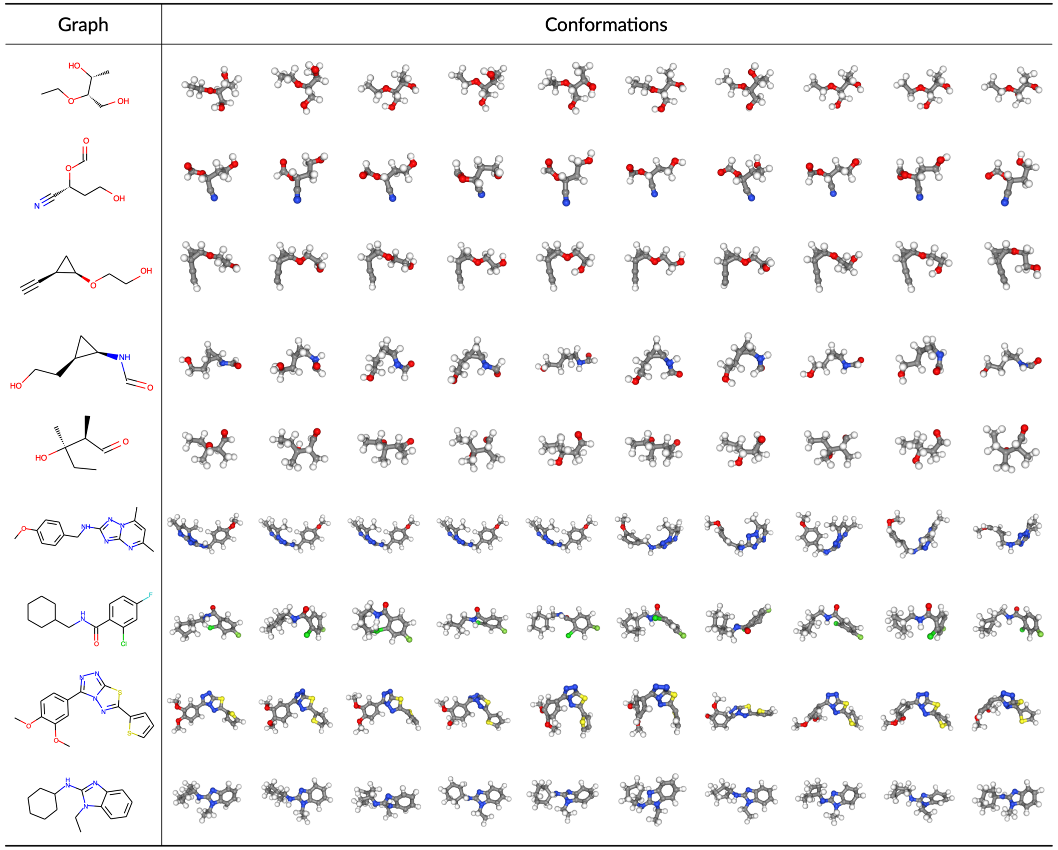
- 在给定分子图 $\mathcal{G}$ 和成对原子距离 d 的情况下，定义了构象 R 的分布

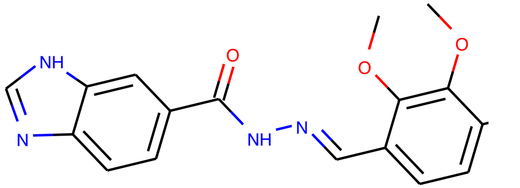
$$p(\boldsymbol{R}|\boldsymbol{d},\mathcal{G}) = \frac{1}{Z}\exp\Big\{-\sum_{e_{uv}\in\mathcal{E}}\alpha_{uv}\big(\|\boldsymbol{r}_u-\boldsymbol{r}_v\|_2-d_{uv}\big)^2\Big\}$$

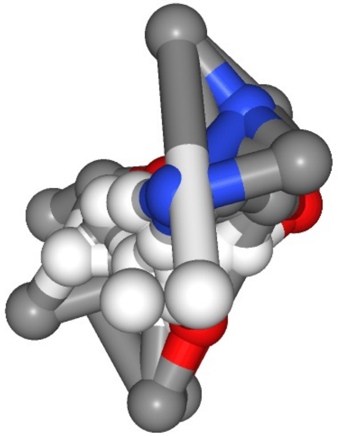
- 试图找到满足距离约束的构象 R

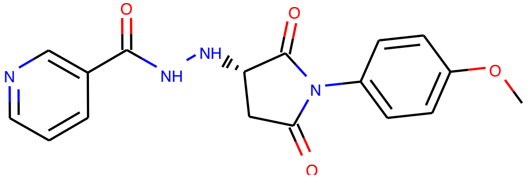


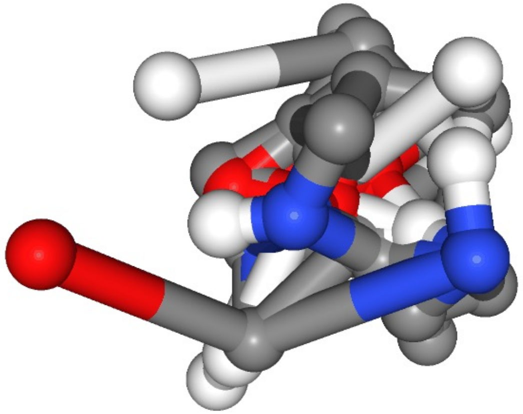
Examples









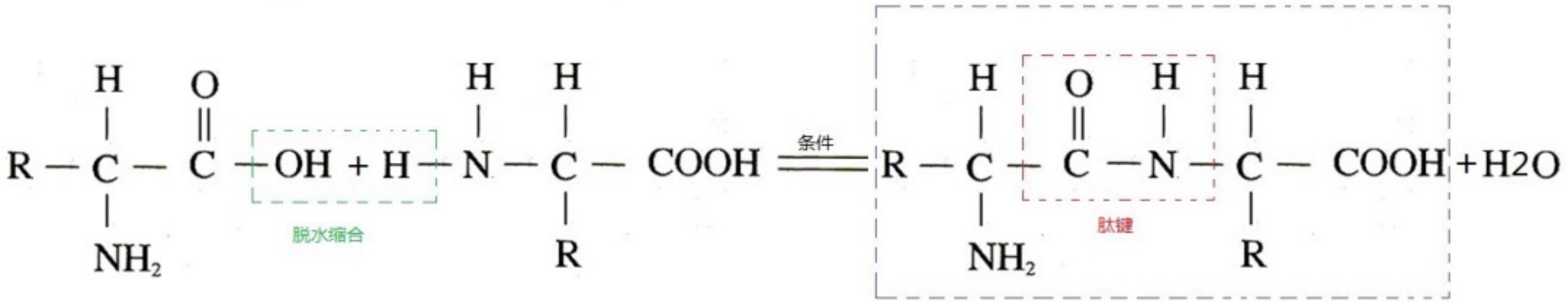


蛋白质结构预测的任务介绍和所需要的基础知识

- 蛋白质结构预测
  - 输入:氨基酸序列
  - 输出:组成蛋白质的每个原子的三维坐标
- 氨基酸组成
  - 氨基酸的核心是碳原子,碳原子有4个键位,分别连接:羧基(COOH)、氨基(NH2)、氢原子(H)、特定基团(R)
  - 常见的R基团有20种,再加上未知的基团(UNK),总共可以视为21个词的词汇表
  - 可以像对文本词向量化那样,对氨基酸序列进行向量化表示

肽键

- 氨基酸的羧基(COOH)和氨基(NH2)可以发生脱水缩合,形成肽键(CONH)
- 多个氨基酸通过肽键连接,形成肽链
- 肽链中的氨基酸连接后,失去了H2O,所以称为氨基酸残基
- 生物学论文中常见的residue指的就是氨基酸残基



在自然语言处理中的应用

- 可以将蛋白质结构预测视为序列建模问题
- 对氨基酸残基进行词向量表示
- 应用自然语言处理模型进行结构预测

氨基酸链

- 多个氨基酸通过肽键顺序连接,形成氨基酸链
- 氨基酸链表示蛋白质的一级结构
- 一级结构就是氨基酸连接顺序的线性表示
- 骨架链和侧链
  - 氨基酸中的NH2-CH-COOH 结构称为骨架链（backbone）,是氨基酸的固定共同部分
  - 氨基酸中的R基团称为侧链,可以不同,决定氨基酸类型
  - 骨架链确定氨基酸链的连接方式;侧链的不同组合则决定蛋白质功能

二级结构

- 一级结构折叠形成不同的规则片段,称为二级结构
- 常见二级结构:α螺旋、β折叠、无规随机卷曲
- 折叠后规则的片段，周期性的结构构象
  - 挨着的肽键会形成某一个角度的螺旋上升（螺旋）
  - 一级结构上挺远的片段通过氢键折叠在一起（折叠）
  - 松散的结构（coil）
  - 某个位置发生了90度以上的急转弯，又记录为转角
- 表示折叠后的周期性和规律性结构

二级结构

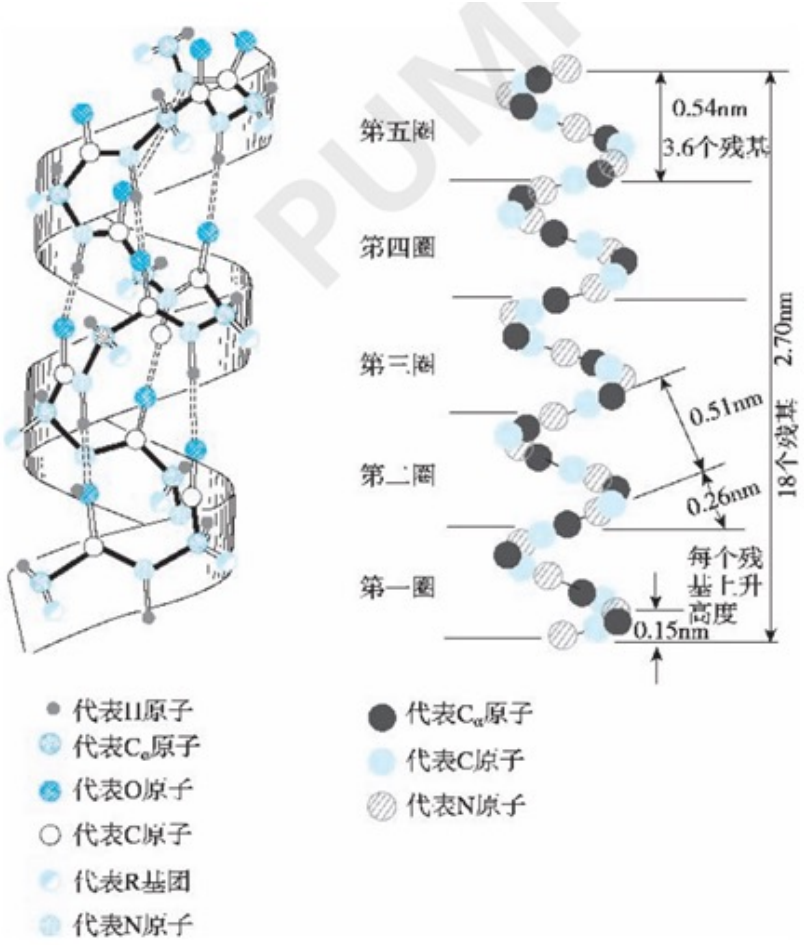


图 1-7 右手 α 螺旋

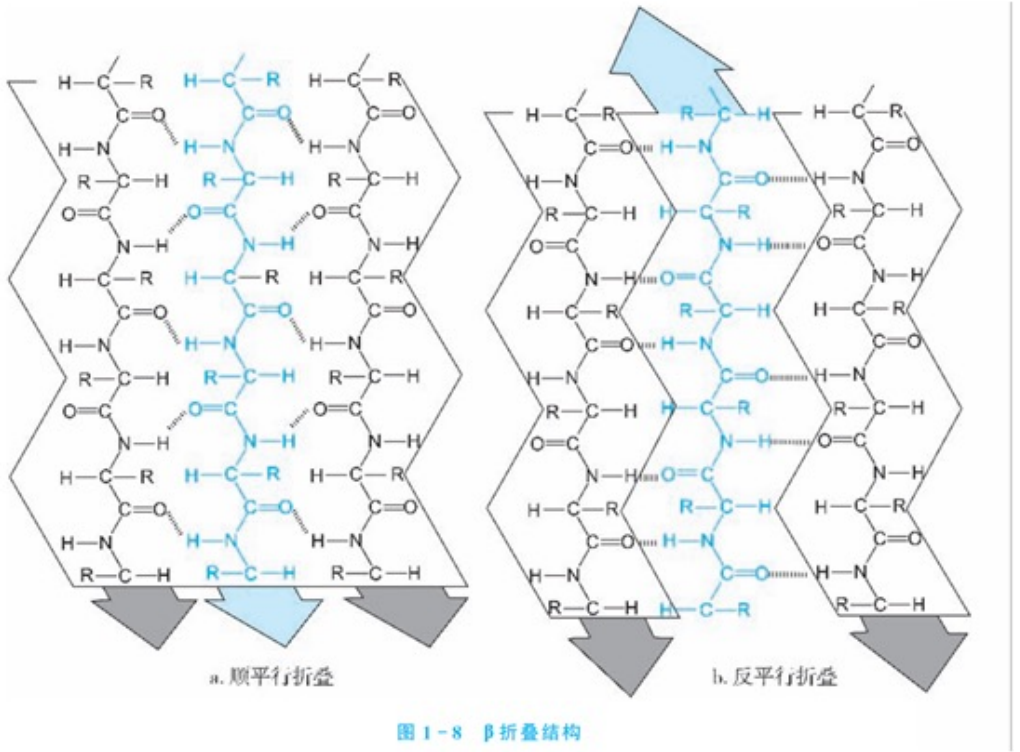


图 1-8 β 折叠结构

三级结构

- 在考虑各侧链的作用力后,氨基酸链折叠形成完整的三维立体结构
- 三级结构是蛋白质实际的立体形状,决定其功能
- 反映各侧链的化学作用力,如二硫键形成、无规结构的具体形状等
  - 二硫键把某些位置拉近了
  - coil结构到底松散成了什么具体形状
  - 某些作用力又使得结构更复杂地变化了

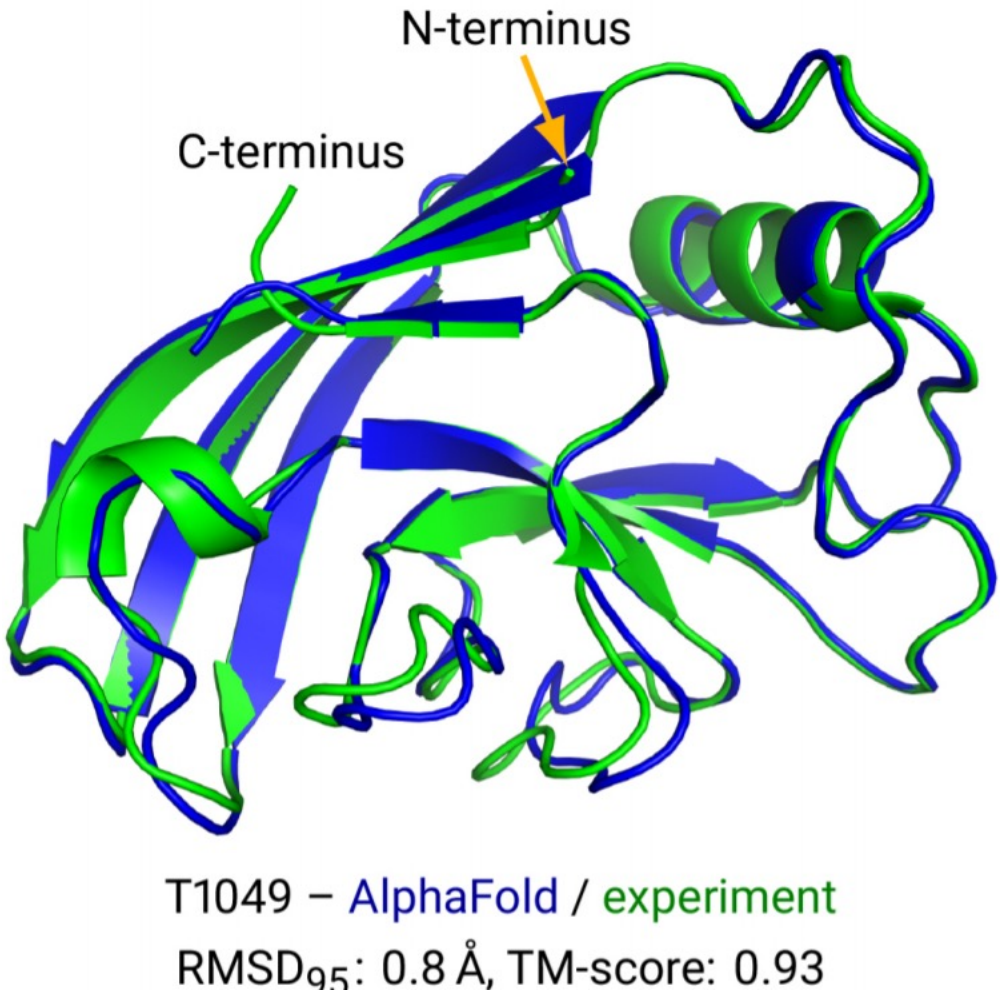
四级结构

- 两个以上的蛋白质分子通过分子间作用力连接,形成蛋白质复合物
- 这种蛋白质之间的连接过程称为对接(docking)
  - 是信号分子发生作用，蛋白质发挥功能，药物和蛋白质结合等生化反应里的重要模拟
- 四级结构常见于许多重要生化过程中,如信号转导等

结构等级预测

- 生物学认为低级结构决定高级结构
- 输入氨基酸序列,可以预测出空间结构
- 对NLP来说,输入序列预测结构,类似文本任务
  - 预测二级结构就是序列标注，对每个氨基酸分类在螺旋区域，还是折叠等等，就好比在BERT的最后一层做一个八分类。
  - 三级结构预测则复杂得多，需要预测每个氨基酸甚至原子的三维坐标

AlphaFold蛋白预测实例

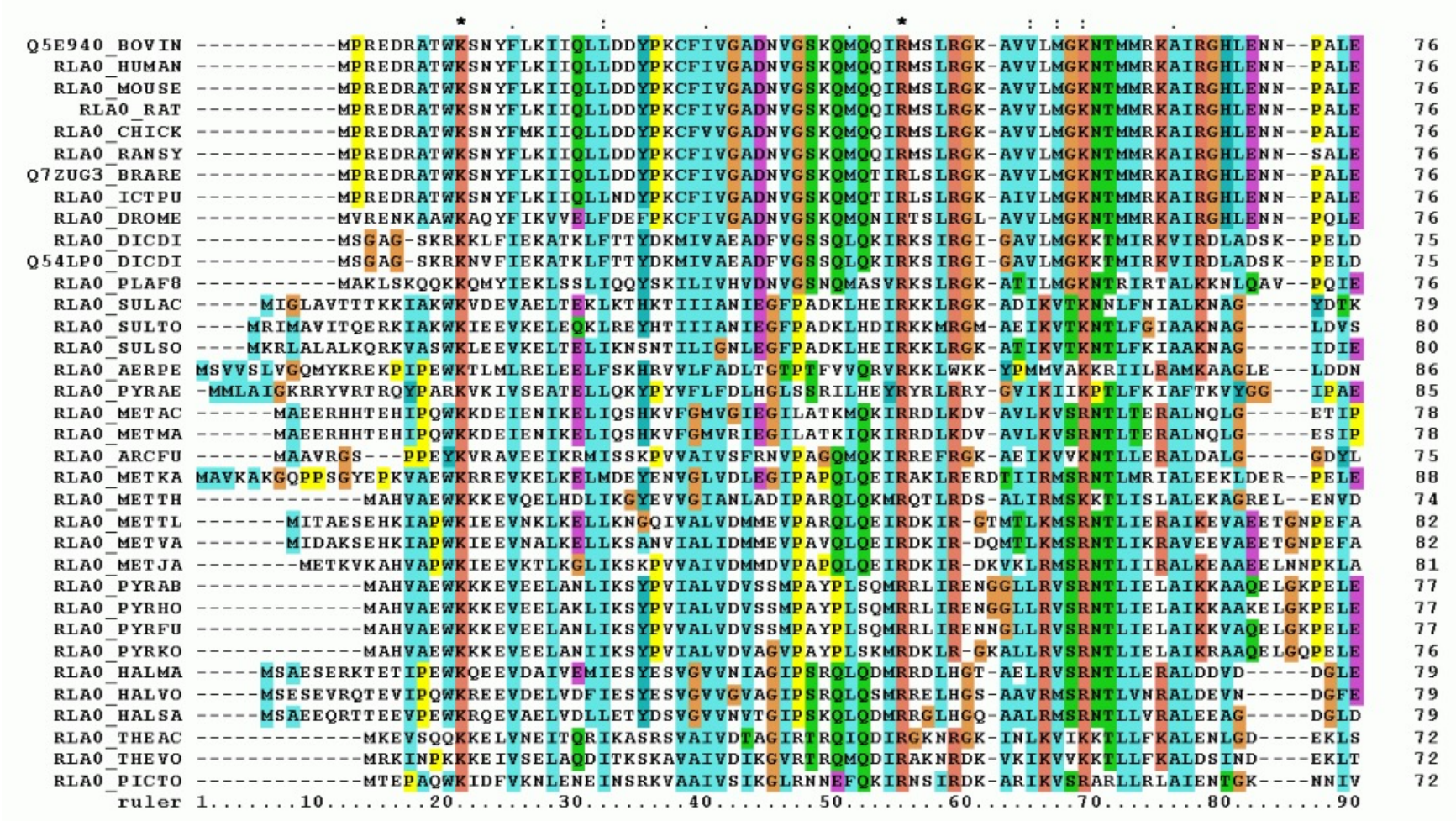


- 螺旋的部分就是螺旋（右上部分）
- 宽箭头部分是折叠（图片中间和左上部分）
- 松散的线是coil部分（图片的右下部分）
- 二级结构可以部分描述蛋白质形状
- 但精确表达形状仍需要三级结构的三维坐标
- 坐标原点和方位可任意变换,表达同一蛋白质

MSA

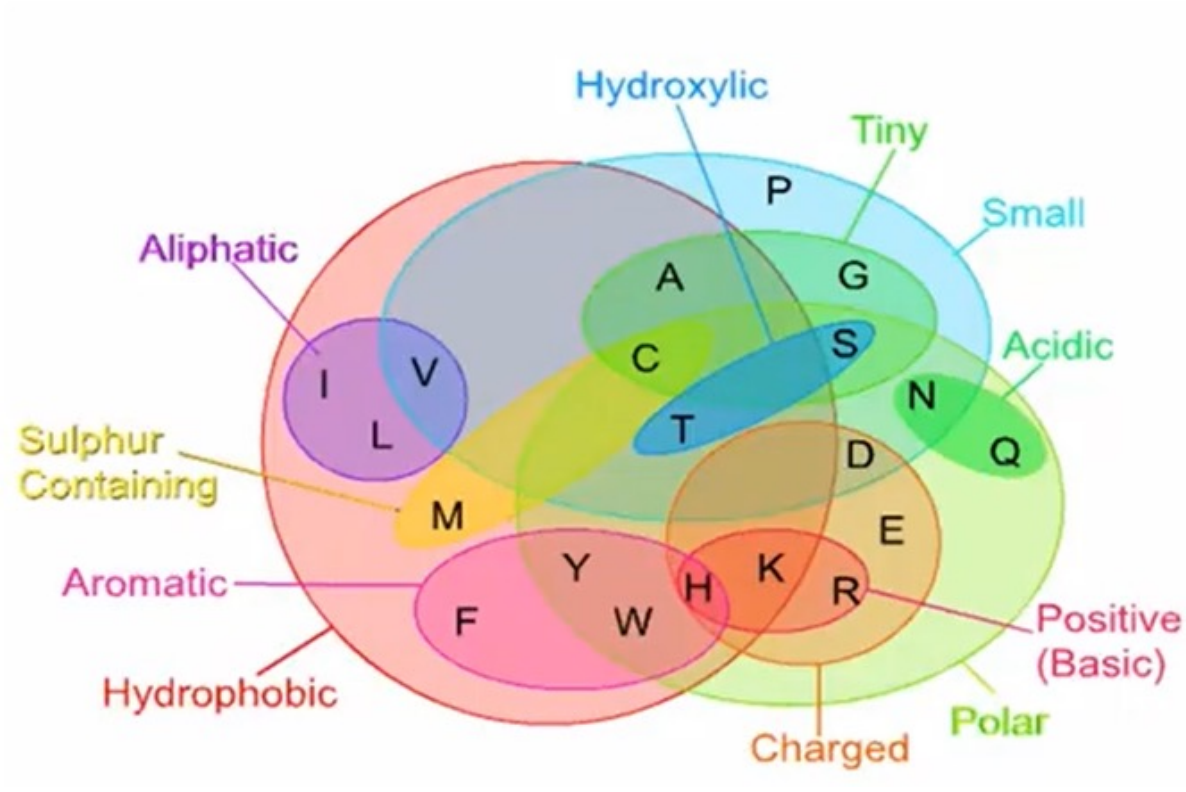
- 把多条同源氨基酸序列进行比对排列
- 先将序列进行对齐
  - 相同氨基酸直接对齐
  - 性质相似氨基酸也可对齐
- 再处理无法对齐的区域
  - 性质不同的氨基酸不对齐
  - 序列中也可能存在缺失或插入
- MSA目的是找到同源序列的共性
  - 发现保守区域
  - 推断蛋白质结构和功能等信息

MSA


Q5E940_BOVIN ----------MPREDRATWKSNYFLKIIQLLDDYPKCFIVGADNVGSKQMQQIRMSLRGK-AVVLMGKNTMMRKAIRGHLENN--PALE 76
RLA0_HUMAN ----------MPREDRATWKSNYFLKIIQLLDDYPKCFIVGADNVGSKQMQQIRMSLRGK-AVVLMGKNTMMRKAIRGHLENN--PALE 76
RLA0_MOUSE ----------MPREDRATWKSNYFLKIIQLLDDYPKCFIVGADNVGSKQMQQIRMSLRGK-AVVLMGKNTMMRKAIRGHLENN--PALE 76
RLA0_RAT ----------MPREDRATWKSNYFLKIIQLLDDYPKCFIVGADNVGSKQMQQIRMSLRGK-AVVLMGKNTMMRKAIRGHLENN--PALE 76
RLA0_CHICK ----------MPREDRATWKSNYFMKIIQLLDDYPKCFVVGADNVGSKQMQQIRMSLRGK-AVVLMGKNTMMRKAIRGHLENN--PALE 76
RLA0_RANSY ----------MPREDRATWKSNYFLKIIQLLDDYPKCFIVGADNVGSKQMQQIRMSLRGK-AVVLMGKNTMMRKAIRGHLENN--SALE 76
Q7ZUG3_BRARE ----------MPREDRATWKSNYFLKIIQLLDDYPKCFIVGADNVGSKQMQTIRLSLRGK-AVVLMGKNTMMRKAIRGHLENN--PALE 76
RLA0_ICTPU ----------MPREDRATWKSNYFLKIIQLLNDYPKCFIVGADNVGSKQMQTIRLSLRGK-AIVLMGKNTMMRKAIRGHLENN--PALE 76
RLA0_DROME ----------MVRENKAAWKAQYFIKVVELFDEFPKCFIVGADNVGSKQMQNIRTSLRGL-AVVLMGKNTMMRKAIRGHLENN--PQLE 76
RLA0_DICDI ----------MSGAG-SKRKKLFIEKATKLFTTYDKMIVAEADFVGSSQLQKIRKSIRGI-GAVLMGKKTMIRKVIRDLADSK--PELD 75
Q54LP0_DICDI ----------MSGAG-SKRKNVFIEKATKLFTTYDKMIVAEADFVGSSQLQKIRKSIRGI-GAVLMGKKTMIRKVIRDLADSK--PELD 75
RLA0_PLAF8 ----------MAKLSKQQKKQMYIEKLSSLIQQYSKILIVHVDNVGSNQMASVRKSLRGK-ATILMGKNTRIRTALKKNLQAV--PQIE 76
RLA0_SULAC -----MIGLAVTTTKKIAKWKVDEVAELTEKLKTHKTIIIANIEGFPADKLHEIRKKLRGK-ADIKVTKNNLFNIALKNAG-----YDTK 79
RLA0_SULTO ----MRIMAVITQERKIAKWKIEEVKELEQKLREYHTIIIANIEGFPADKLHDIRKKMRGM-AEIKVTKNTLFGIAAKNAG-----LDVS 80
RLA0_SULSO ----MKRLALALKQRKVASWKLEEVKELTELIKNSNTILIGNLEGFPADKLHEIRKKLRGK-ATIKVTKNTLFKIAAKNAG-----IDIE 80
RLA0_AERPE MSVVSLVGQMYKREKPIPEWKTLMLRELEELFSKHRVVLFADLTGTPTFVVQRVRKKLWKK-YPMMVAKKRIILRAMKAAGLE---LDDN 86
RLA0_PYRAE -MMLAIGKRRYVRTRQYPARKVKIVSEATELLQKYPYVFLFDLHGLSSRILHEYRYRLRRY-GVIKIIKPTLFKIAFTKVYGG---IPAE 85
RLA0_METAC ------MAEERHHTEHIPQWKKDEIENIKELIQSHKVFGMVGIEGILATKMQKIRRDLKDV-AVLKVSRNTLTERALNQLG-----ETIP 78
RLA0_METMA ------MAEERHHTEHIPQWKKDEIENIKELIQSHKVFGMVRIEGILATKIQKIRRDLKDV-AVLKVSRNTLTERALNQLG-----ESIP 78
RLA0_ARCFU ------MAAVRGS---PPEYKVRAVEEIKRMISSKPVVAIVSFRNVPAGQMQKIRREFRGK-AEIKVVKNTLLERALDALG-----GDYL 75
RLA0_METKA MAVKAKGQPPSGYEPKVAEWKRREVKELKELMDEYENVGLVDLEGIPAPQLQEIRAKLRERDTIIRMSRNTLMRIALEEKLDER--PELE 88
RLA0_METTH ------------MAHVAEWKKKEVQELHDLIKGYEVVGIANLADIPARQLQKMRQTLRDS-ALIRMSKKTLISLALEKAGREL--ENVD 74
RLA0_METTL -------MITAESEHKIAPWKIEEVNKLKELLKNGQIVALVDMMEVPARQLQEIRDKIR-GTMTLKMSRNTLIERAIKEVAEETGNPEFA 82
RLA0_METVA -------MIDAKSEHKIAPWKIEEVNALKELLKSANVIALIDMMEVPAVQLQEIRDKIR-DQMTLKMSRNTLIKRAVEEVAEETGNPEFA 82
RLA0_METJA --------METKVKAHVAPWKIEEVKTLKGLIKSKPVVAIVDMMDVPAPQLQEIRDKIR-DKVKLRMSRNTLIIRALKEAAEELNNPKLA 81
RLA0_PYRAB ------------MAHVAEWKKKEVEELANLIKSYPVIALVDVSSMPAYPLSQMRRLIRENGGLLRVSRNTLIELAIKKAAQELGKPELE 77
RLA0_PYRHO ------------MAHVAEWKKKEVEELAKLIKSYPVIALVDVSSMPAYPLSQMRRLIRENGGLLRVSRNTLIELAIKKAAKELGKPELE 77
RLA0_PYRFU ------------MAHVAEWKKKEVEELANLIKSYPVVALVDVSSMPAYPLSQMRRLIRENNGLLRVSRNTLIELAIKKVAQELGKPELE 77
RLA0_PYRKO ------------MAHVAEWKKKEVEELANIIKSYPVIALVDVAGVPAYPLSKMRDKLR-GKALLRVSRNTLIELAIKRAAQELGQPELE 76
RLA0_HALMA -----MSAESERKTETIPEWKQEEVDAIVEMIESYESVGVVNIAGIPSRQLQDMRRDLHGT-AELRVSRNTLLERALDDVD-----DGLE 79
RLA0_HALVO -----MSESEVRQTEVIPQWKREEVDELVDFIESYESVGVVGVAGIPSRQLQSMRRELHGS-AAVRMSRNTLVNRALDEVN-----DGFE 79
RLA0_HALSA -----MSAEEQRTTEEVPEWKRQEVAELVDLLETYDSVGVVNVTGIPSKQLQDMRRGLHGQ-AALRMSRNTLLVRALEEAG-----DGLD 79
RLA0_THEAC ------------MKEVSQQKKELVNEITQRIKASRSVAIVDTAGIRTRQIQDIRGKNRGK-INLKVIKKTLLFKALENLGD----EKLS 72
RLA0_THEVO ------------MRKINPKKKEIVSELAQDITKSKAVAIVDIKGVRTRQMQDIRAKNRDK-VKIKVVKKTLLFKALDSIND----EKLT 72
RLA0_PICTO ------------MTEPAQWKIDFVKNLENEINSRKVAAIVSIKGLRNNEFQKIRNSIRDK-ARIKVSRARLLRLAIENTGK----NNIV 72
ruler 1.....10.....20.....30.....40.....50.....60.....70.....80.....90

MSA

- 根据氨基酸的性质相似程度如图，我们对任意两个氨基酸之间的对齐进行打分



Amino Acids

A alanine (ala)
R arginine (arg)
N asparagine (asn)
D aspartic acid (asp)
C cysteine (cys)
Q glutamine (gln)
E glutamic acid (glu)
G glycine (gly)
H histidine (his)
I isoleucine (ile)
L leucine (leu)
K lysine (lys)
M metioneine (met)
F phenyalanine (phe)
P proline (pro)
S serine (ser)
T threonine (thr)
W trytophan (trp)
Y tyrosine (tyr)

MSA

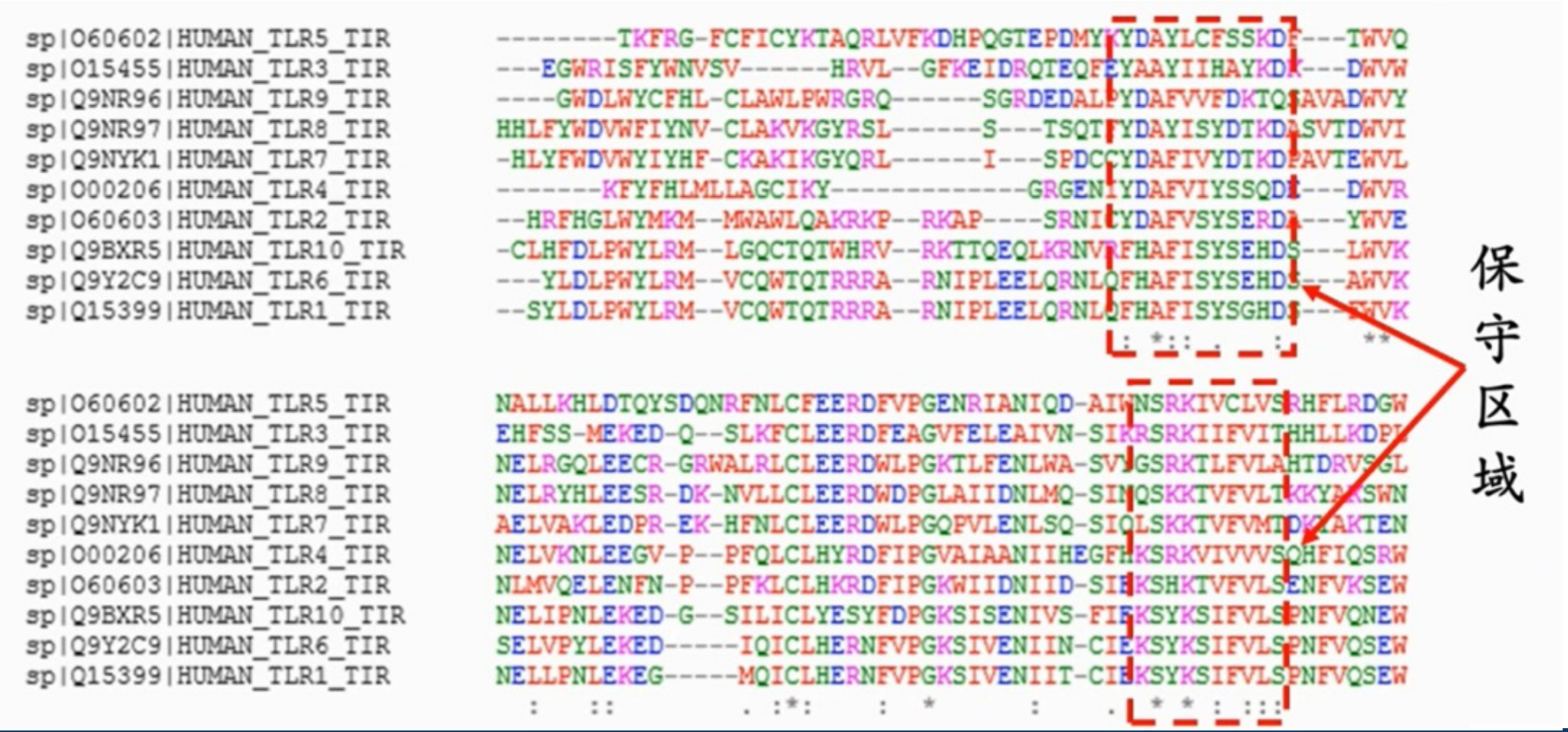
- 打分结果

```
#=======================================
#
# Aligned_sequences: 2
# 1: HBA_HUMAN
# 2: HBB_HUMAN
# Matrix: EBLOSUM62
# Gap_penalty: 10.0
# Extend_penalty: 0.5
#
# Length: 149
# Identity:      65/149 (43.6%)
# Similarity:    90/149 (60.4%)
# Gaps:           9/149 ( 6.0%)
# Score: 292.5
#
#
#=======================================

HBA_HUMAN         1 MV-LSPADKTNVKAAWGKVGAHAGEYGAEALERMFLSFPTTKTYFPHF-D     48
                    || |:|.:|:.|.|.|||| :..|.|.|||.|:.:.:|.|:.:|..| |
HBB_HUMAN         1 MVHLTPEEKSAVTALWGKV--NVDEVGGEALGRLLVVYPWTQRFFESFGD     48

HBA_HUMAN        49 LS-----HGSAQVKGHGKKVADALTNAVAHVDDMPNALSALSDLHAHKLR     93
                    ||     .|:.:||.||||||..|.::.:|||:|::....:.||:||..||.
HBB_HUMAN        49 LSTPDAVMGNPKVKAHGKKVLGAFSDGLAHLDNLKGTFATLSELHCDKLR     98

HBA_HUMAN        94 VDFVNFKLLSHCLLVTLAAHLPAEFTPAVHASLDKFLASVSTVLTSKYR    142
                    |||.||:||.:.|:..||.|...||||.|.|:..|.:|.|:..|..||.
HBB_HUMAN        99 VDPENFRLLGNVLVCVLAHHFGKEFTPPVQAAYQKVVAGVANALAHKYH    147
```

	C	S	T	P	A	G	N	D	E	Q	H	R	K	M	I	L	V	F	Y	W	
C	9																				C
S	-1	4																			S
T	-1	1	5																		T
P	-3	-1	-1	7																	P
A	0	1	0	-1	4																A
G	-3	0	-2	-2	0	6															G
N	-3	1	0	-2	-2	0	6														N
D	-3	0	-1	-1	-2	-1	1	6													D
E	-4	0	-1	-1	-1	-2	0	2	5												E
Q	-3	0	-1	-1	-1	-2	0	0	2	5											Q
H	-3	-1	-2	-2	-2	-2	1	-1	0	0	8										H
R	-3	-1	-1	-2	-1	-2	0	-2	0	1	0	5									R
K	-3	0	-1	-1	-1	-2	0	-1	1	1	-1	2	5								K
M	-1	-1	-1	-2	-1	-3	-2	-3	-2	0	-2	-1	-1	5							M
I	-1	-2	-1	-3	-1	-4	-3	-3	-3	-3	-3	-3	-3	1	4						I
L	-1	-2	-1	-3	-1	-4	-3	-4	-3	-2	-3	-2	-2	2	2	4					L
V	-1	-2	0	-2	0	-3	-3	-3	-2	-2	-3	-3	-2	1	3	1	4				V
F	-2	-2	-2	-4	-2	-3	-3	-3	-3	-3	-1	-3	-3	0	0	0	-1	6			F
Y	-2	-2	-2	-3	-2	-3	-2	-3	-2	-1	2	-2	-2	-1	-1	-1	-1	3	7		Y
W	-2	-3	-2	-4	-3	-2	-4	-4	-3	-2	-2	-3	-3	-1	-3	-2	-3	1	2	11	W
	C	S	T	P	A	G	N	D	E	Q	H	R	K	M	I	L	V	F	Y	W	

MSA

- 对于任意两个氨基酸序列，可以通过补空位，左右移动位置等等，使得匹配的全局得分达到最高，此时我们就得到了两条氨基酸序列的对齐。
- 同理,可以对多条序列进行多序列比对



- 现在我们拥有很多的MSA搜索工具，基于一些局部匹配算法，我们输入一条想要分析的氨基酸序列，可以很快地在氨基酸序列库中搜到相似的氨基酸序列并进行对齐。

MSA的意义

- 多序列比对的主要目的是共进化分析找到保守区域（保守区域即上图中对齐的很好的部分）和其他特征
- 假设找到的相似蛋白质来自同一祖先
- 生物体内常发生突变
- 如果突变点在不重要区域
  - 对结构和功能影响不大
- 如果突变点在关键区域
  - 可能导致功能变化
  - 影响生物存活
- 因此保守区域可能是发生功能的关键区域

- 我们想要得到的是同源蛋白质，而氨基酸相似性搜索只能搜到相似的蛋白质
- 分析这些蛋白质是不是同一个祖先进化来的很困难
- 相似蛋白质就假设他们都是同源共进化的蛋白质来使用。
- 实际上，同源的蛋白质可能因为不同的进化方向而很不相似，而非同源的蛋白质也可能趋同进化变的相似。
- 相似度过高或过低的序列意义不大
- 一般取相似度30-90%的序列

CASP

- Critical Assessment of protein Structure Prediction (CASP), 蛋白质结构预测技术的关键测试
- 自1994年以来每两年进行一次的全球范围内的蛋白质结构预测竞赛
- 目的是更好预测和破解蛋白质三维结构

为啥深度学习研究蛋白质结构很重要

- 分析蛋白质功能
  - 蛋白质的三维结构决定了其生理功能的实现方式
  - 为了充分理解一个蛋白质的生理功能,必须先预测或测定其三维结构
  - 通过蛋白质三维结构,可以确定功能区域,如活性中心、结合位点、抗原表位等
  - 对于酶或其他具有催化活性的蛋白质,其活性中心具高度保守性,而外围部分可变性较大
  - 因此,改变不同氨基酸残基并研究对蛋白质功能的影响,可以依据结构预测结果进行有针对性的设计

为啥深度学习研究蛋白质结构很重要

- 药物设计
  - 开发新药物的第一步是确定靶点蛋白
    - 即哪个蛋白质/哪个基因出了问题，需要去抑制这个靶点还是去激活这个靶点
  - 针对靶点进行对应小分子药物的设计
    - 模拟体内的小分子再修改其功能
    - 修改已有相似药物的功能
    - 根据靶点的三维结构去设计对应结构可以结合上的小分子
    - 对已知可合成的小分子进行高通量筛选
    - 对这一步中所有可能的小分子进行缩小范围，比如判断其毒性，跨模型，在体内会残留多久，会不会对其他器官/蛋白质产生损害等等
    - 实验和临床，从试管和小动物开始，逐步到人体试验
  - 如果能提前预测靶点蛋白质的三维结构,将大大提高药物设计的效率
  - 从而缩短新药研发周期,减少制药公司前期的时间与成本投入

整个过程需要花费10到15年去研发一款新药，花费5到10亿美元或者更多

为啥深度学习研究蛋白质结构很重要

- 实验测定困难
  - 通过实验手段测定蛋白质三维结构非常困难昂贵。要获得一个蛋白质的精确三维结构,科学家需要花费数月乃至数年时间进行结晶、NMR等多种实验。但许多蛋白质难以结晶,难以用实验手段解析。
  - 近年冷冻电镜被广泛用于结构测定,但仪器昂贵,技术门槛高,获得一个三维结构仍意味着一篇高影响期刊论文。
  - 而AlphaFold2实现快速准确预测蛋白质三维结构,对结构生物学意义重大,被认为达到诺奖级别成果。

为啥深度学习研究蛋白质结构很重要

- 理解生命
  - DNA、RNA和蛋白质编码和传递生命的信息
  - 分析这些生物分子序列,可以加深理解生命的奥秘
  - 预测蛋白质三维结构是理解生命语言的重要一环
  - 也可以模拟病毒结构,探索生命起源等问题

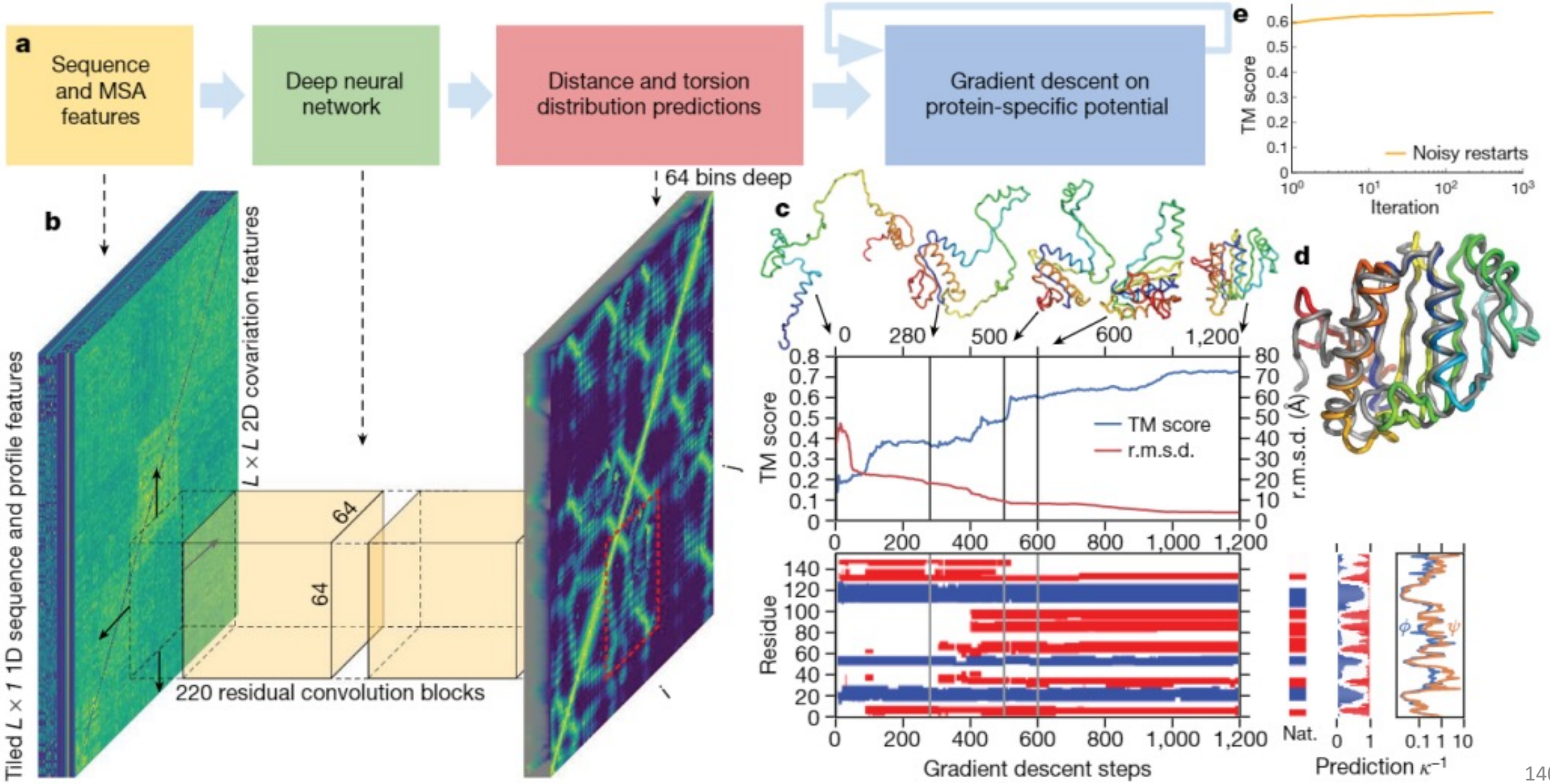
传统蛋白结构预测

- 同源建模法。对于给定的氨基酸序列，检索相同蛋白质家族中已知结构的模板，然后做MSA分析，很相同的部分保留模板结构，不相似的部分再根据能量方程（根据物理学知识，势能越低结构越稳定）修改
- 穿线法。已知的蛋白质结构约十万个，不同结构拓扑的仅有1393个，很少有找不到相同结构拓扑的。在已知序列的各个拓扑结构中，使用能量方程，对给定氨基酸序列选取最低最稳定的拓扑结构当模板

传统蛋白结构预测

- 从头计算法，如果前两种做出来的结构都很差，找不到能用的模板，就可以根据蛋白质的三维结构决定于自身的氨基酸序列，并且处于最低自由能状态的原则，直接计算最低自由能的结构
- 综合法。把给定的氨基酸序列分成一个一个的片段，每个片段使用上面三种方案中最好的方案，然后再拼回来。

AlphaFold-1


a
Sequence and MSA features
Deep neural network
Distance and torsion distribution predictions
Gradient descent on protein-specific potential
b
64 bins deep
L × L 2D covariation features
Tiled L × 1 1D sequence and profile features
64
64
220 residual convolution blocks
i
j
c
0
280
500
600
1,200
TM score
r.m.s.d. (Å)
TM score
r.m.s.d.
Residue
Gradient descent steps
d
e
TM score
Noisy restarts
Iteration
Nat.
Prediction κ⁻¹

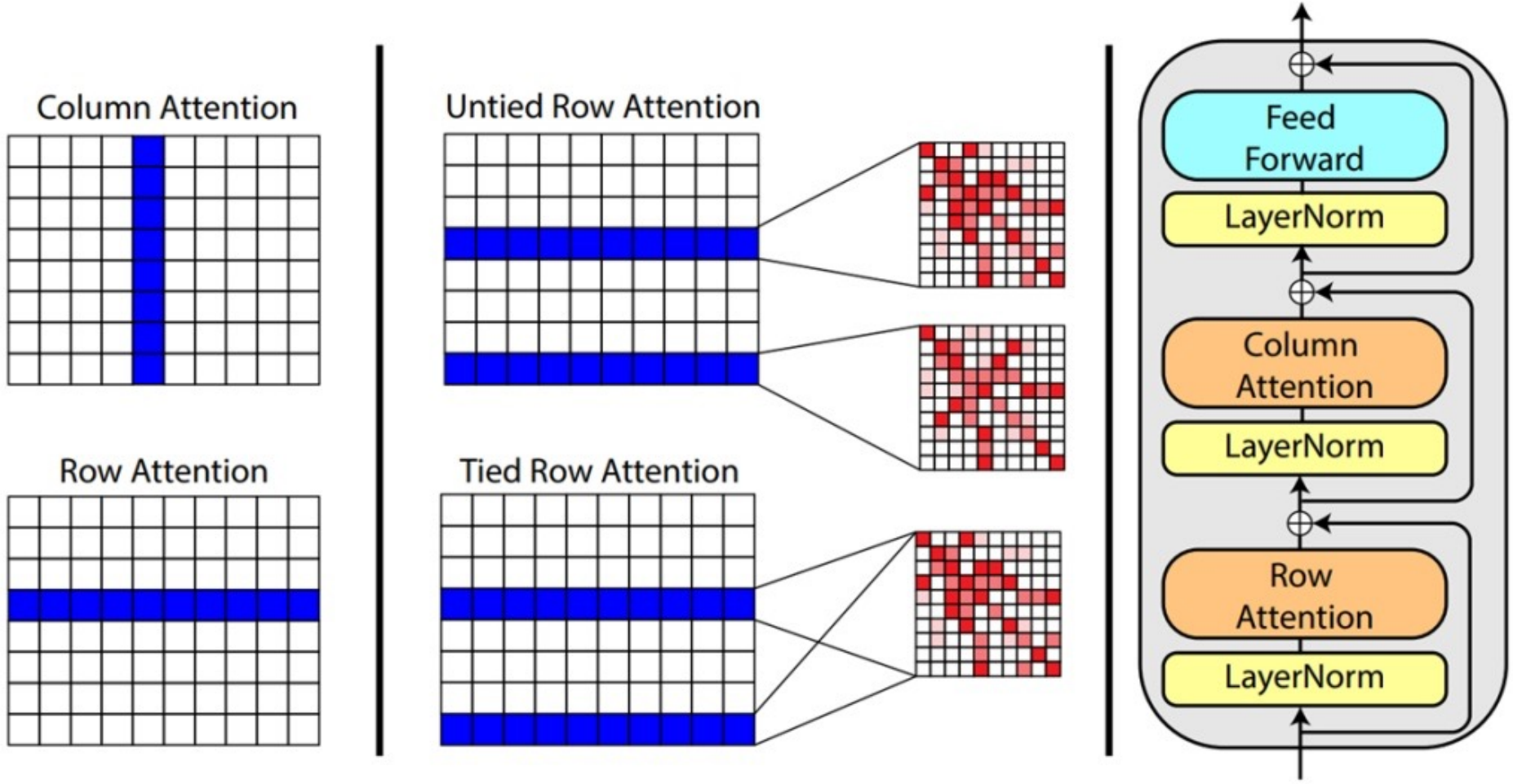
AlphaFold-1

- 输入：MSA feature
- 特征：使用生物学家的传统算法得到
- 输出：氨基酸两两间的距离
- 这个神经网络的输出很好理解，因为三维结构里每个原子的坐标是可以随意旋转和平移的，但是无论如何平移旋转，任意两个氨基酸直接的距离是确定的。
- 神经网络预测出来两两氨基酸之间的距离，而我们最终需要的是氨基酸内部所有原子的距离，因此需要上采样到backbone主链原子间的距离。然后拿到backbone原子距离之后，再喂给能量方程的程序包去补足侧链上原子的距离，然后反复迭代优化势能函数去计算出每个原子的坐标。
- AlphaFold1中，并不是end to end去训练的
  - 喂进来的输入，是传统算法得到的MSA特征值，预测的输出是两两残基间的距离而非最终的三维坐标，运算过程中大量使用生物学家提供的计算程序包。即使如此，在CASP13中，仍然领先其他模型很多，开启了AI进军生物大分子界的序幕。

其他竞品算法

- ESM，MSA Transformer等等
- 在氨基酸序列上，进行BERT、XL-NET、BART等等预训练任务，然后进行二级、三级结构预测、蛋白质性能预测等等任务的finetune。
- 这些工作往往汇报无监督的二级结构预测结果，方法是在预训练后最后一层的attention分布上加一个线性层，使用几个蛋白质去训练这个线性层，然后预测，本质上是few shot
- 比较亮眼的一个工作是MSA Transformer，效果突出，在AlphaFold2开源之前，被认为是AlphaFold2里的MSA部分的一个很好猜测拟合

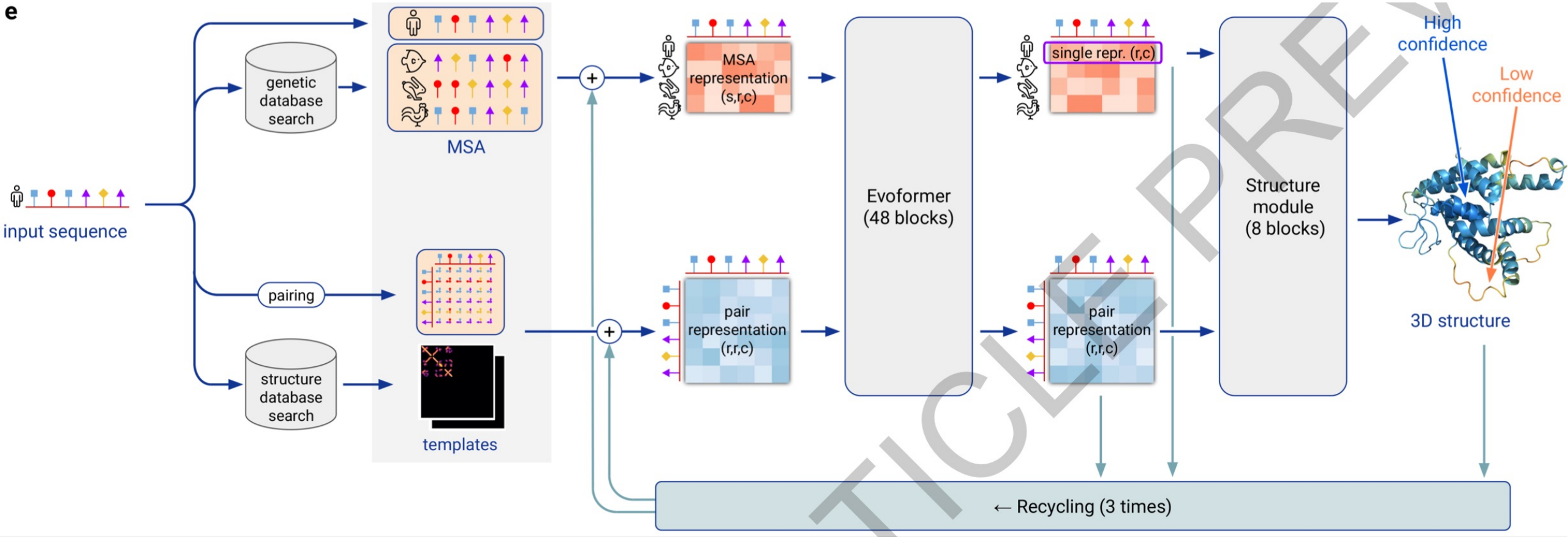
MSA transformer



MSA transformer

- 输入：MSA
- 每层的注意力层被拆分为了横向attention和纵向attention。
- 横向attention就是每个氨基酸序列里的self-attention
- 纵向attention是相同位置的去看其他氨基酸序列里是否被替换了氨基酸还是大家都相同。
- 最好的下游任务结果出现在，横向的attention所有的氨基酸序列共享attention分布，即，使用整个MSA的横向attention平均值作为每个蛋白质的attention。这就意味着，MSA Transformer的结论可以被认为是，单个氨基酸序列的信息表达太差了（比如之前的工作），还不如使用整个MSA的平均值去表达更通用的信息。

Alphafold2



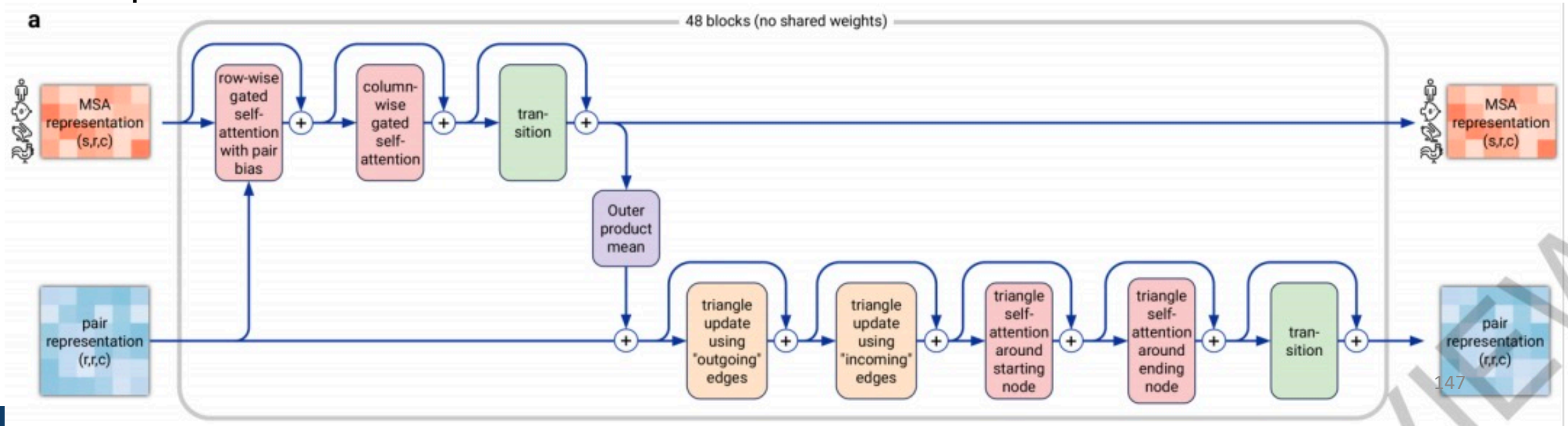
Alphafold2

不是每个残基一个隐状态，是每个残基之间都有一个隐状态来描述他俩之间的关系，我们把每对儿残基之间的隐状态叫做pairwise features

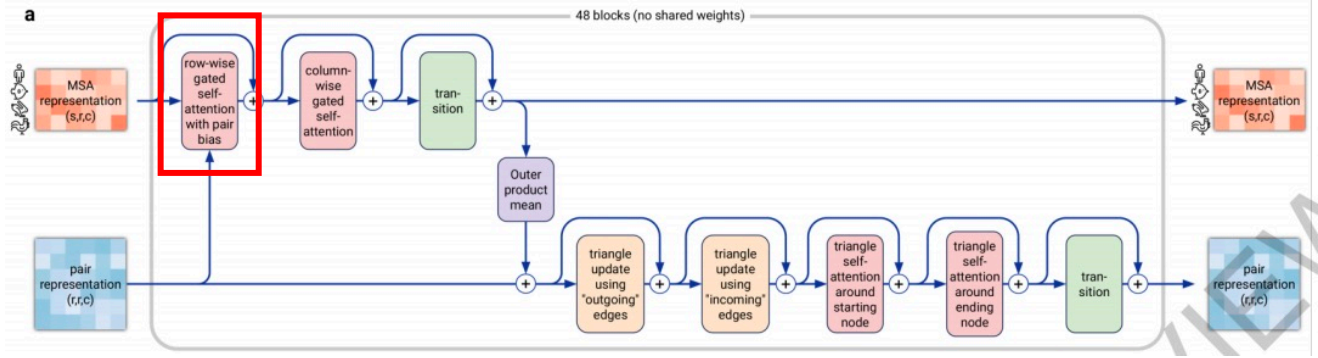
- Alphafold2提出了一个新的结构去同时嵌入MSA和残基-残基对的特征
- 新的输出表示去确保准确的端到端训练，以及新的辅助loss。
- 在finetune训练之前，AlphaFold2首先预训练
  - 在MSA上使用BERT任务遮盖住一些氨基酸再还原回来
  - 使用自蒸馏，自估计的loss去自监督学习
    - 先用训好的模型在只有氨基酸序列的数据上生成预测结果，然后只保留高确信度的，然后使用这个数据预训练，在训练时把输入加上更强的drop out和mask，来增大学习难度，去预测完整信息时高确信度的结果
- 结构由两部分组成，Evoformer和结构模块（Structure Module）
  - Evoformer输入MSA，模板，自己的氨基酸序列，输出MSA信息和残基-残基对关系建模。
  - 结构模块中，丢掉MSA中的其他氨基酸序列，只保留目标的那一条，然后再加上pairwise features，去计算更新backbone frames，预测所有氨基酸的方位和距离，肽键的长度和角度，氨基酸内部的扭转角度等。

Evoformer

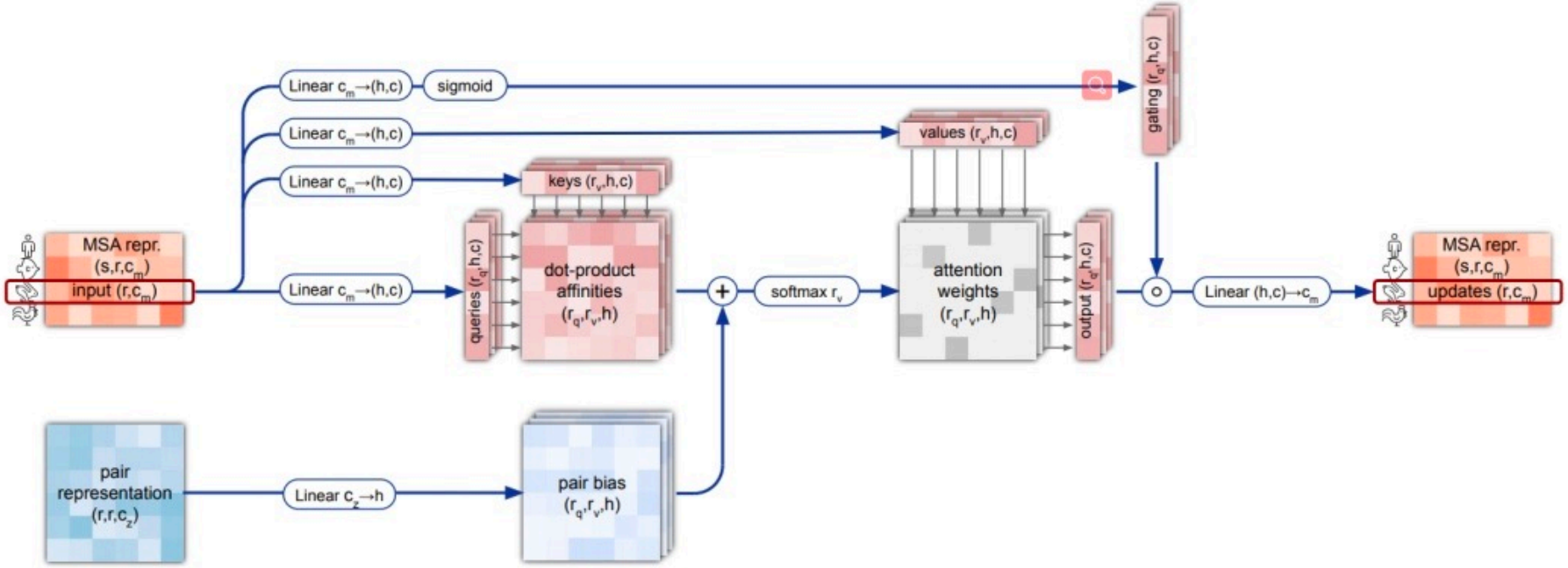
- ***Evoformer***即进化former，进化魔改Transformer
- 用来计算MSA和pairwise features。
  - 匹配到的模板信息仅仅通过一些特征提取的函数被塞进了MSA里面当成给目标氨基酸序列做更新时参考使用的数据
  - 输入MSA和pairwise features，通过很多注意力层，最终输出MSA和pairwise features



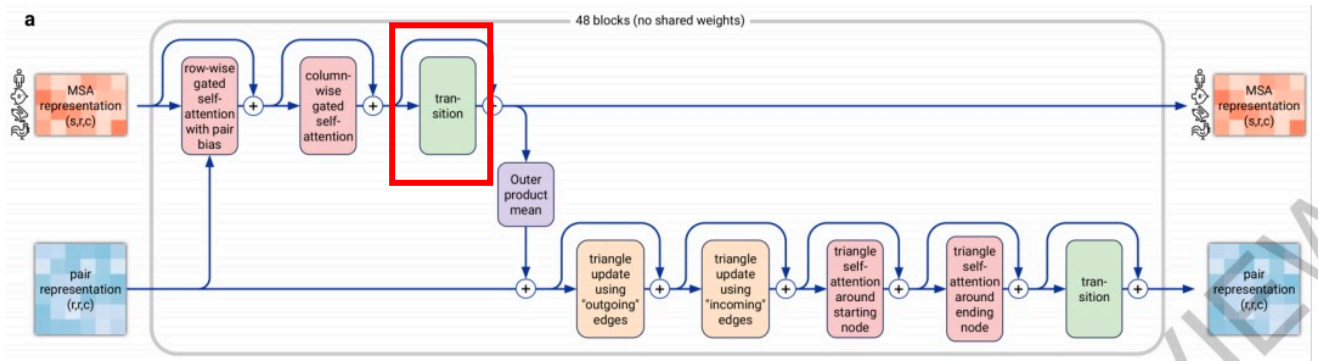
Evoformer



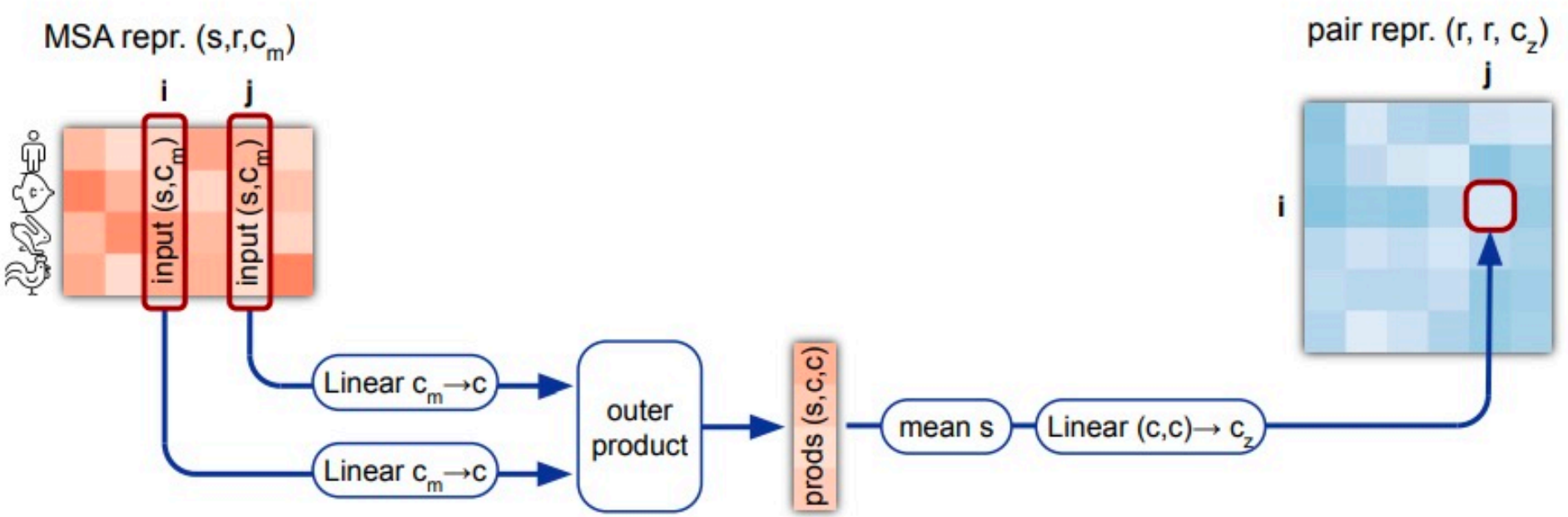
- 有横向attention去往自己当前氨基酸序列的所有位置打attention
  - 加了使用pairwise features作为后attention上的一个bias
- 有纵向的attention去看其他氨基酸序列的相同位置稳定性和突变有多少



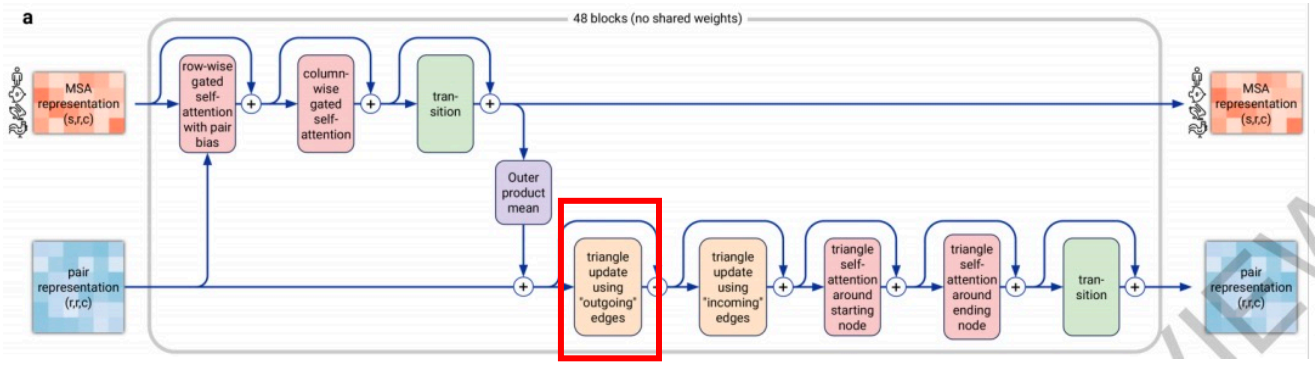
Evoformer



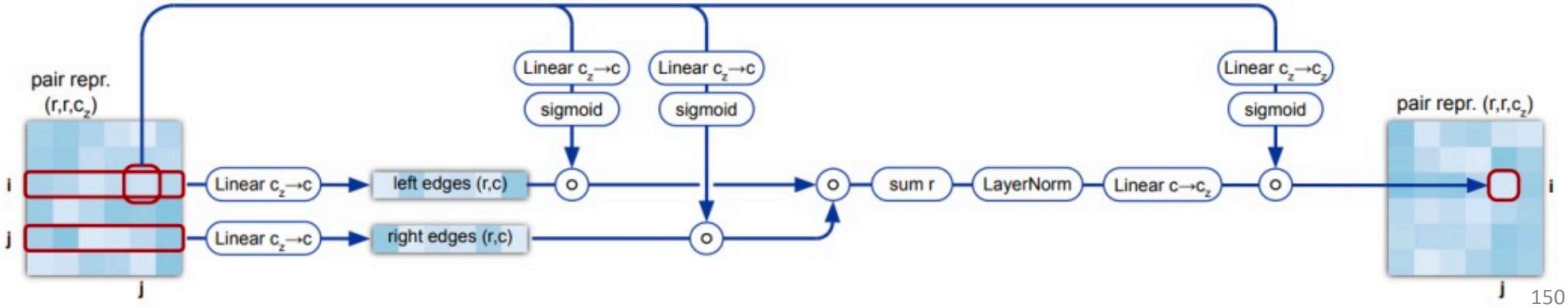
- 最后还有一个transition层，其实就是两层的MLP，中间塞一个relu。transition层的输出结果，通过计算外积，然后求平均，去计算出pairwise features



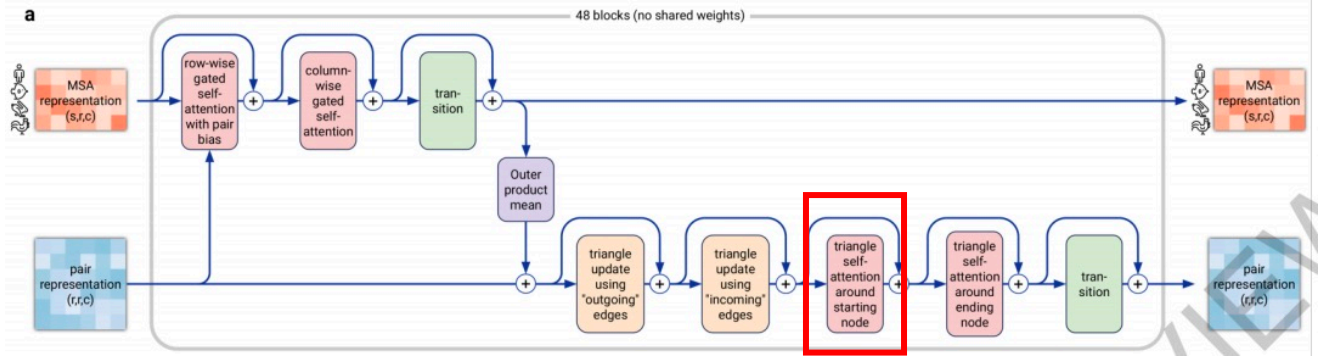
Evoformer



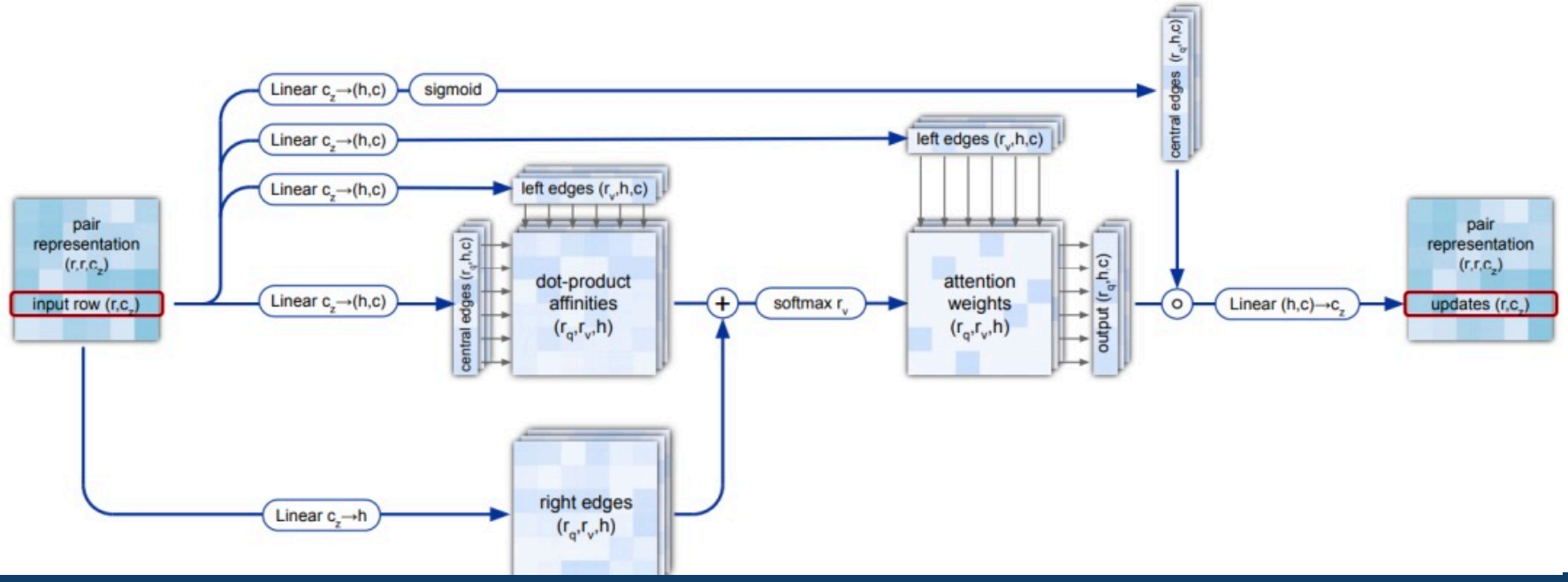
- 三角乘法更新(Triangular multiplicative update)。
- Motivation：这个矩阵描述的是任意两个残基之间的距离等关系，距离关系不是自由的，应该满足三角不等式
  - 相邻的三个边应该满足两边之和大于第三边
  - So, pairwise feature的更新使用了三角乘法更新
  - 对于每个边，都会接收到和他组成三角形的任意两个其他边带来的更新



Evoformer

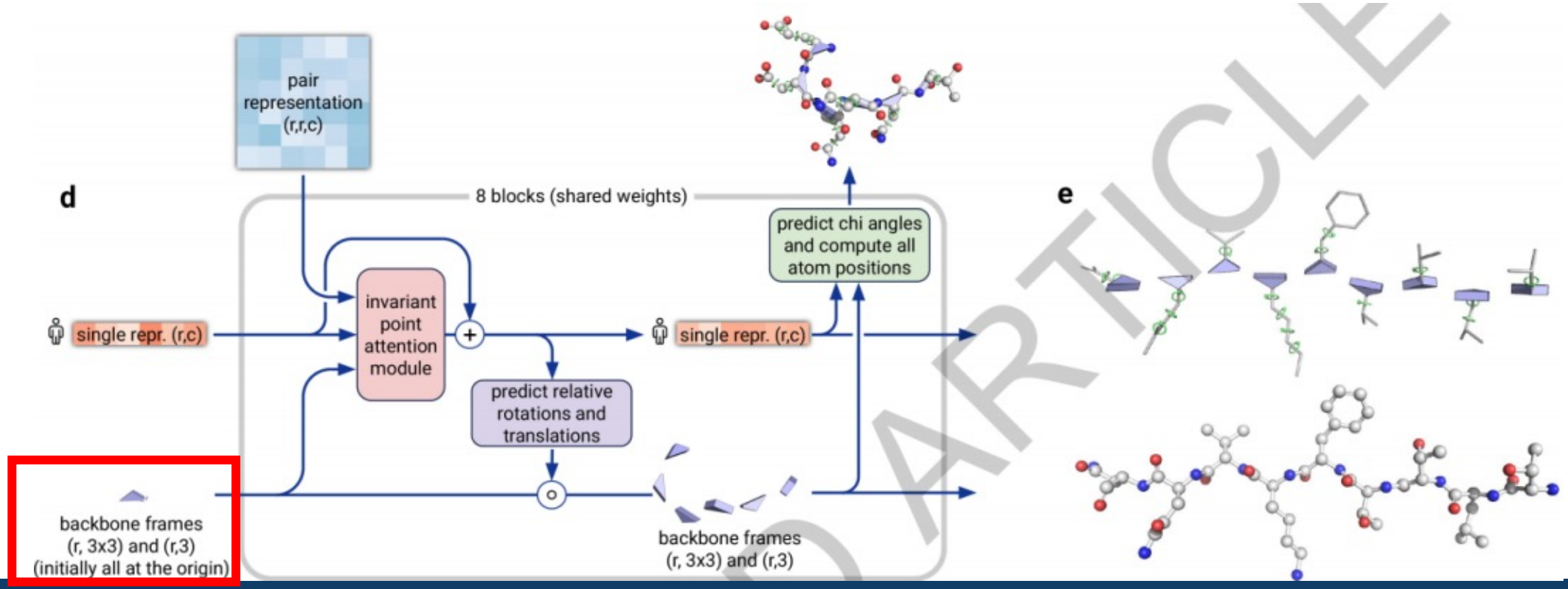


- 三角自注意力层（Triangular self-attention）



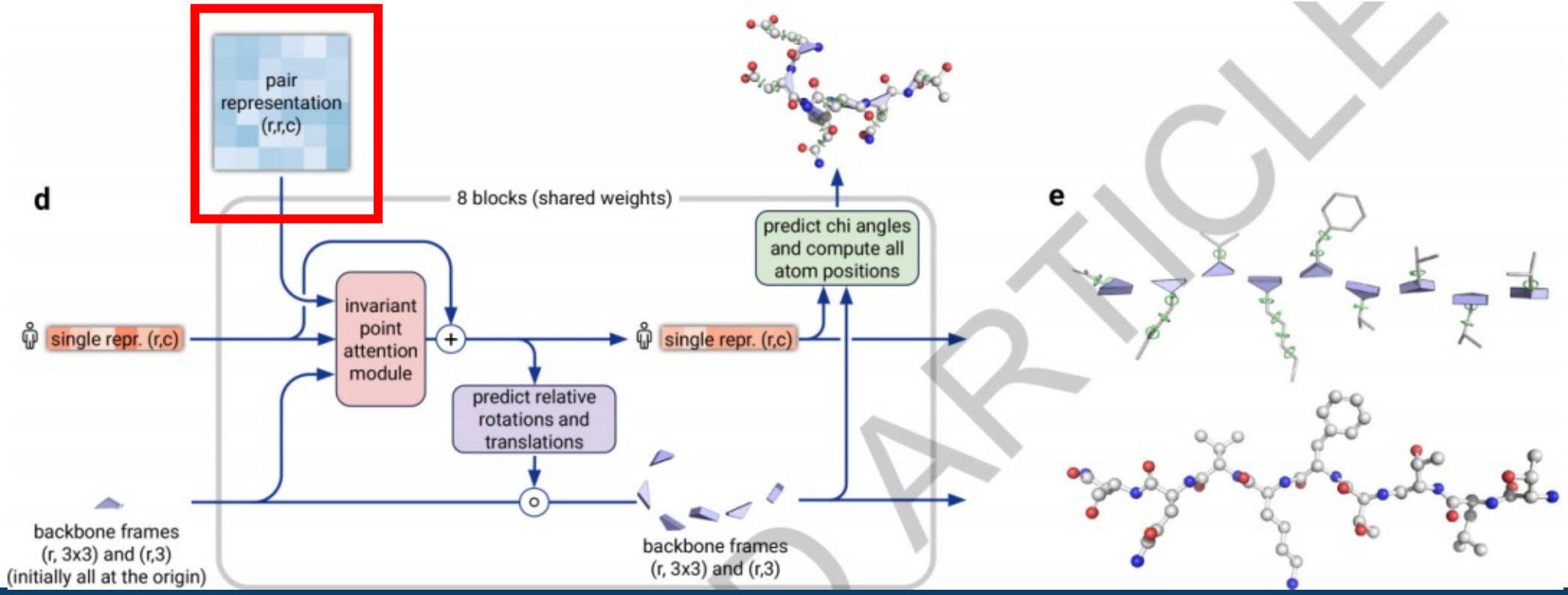
Structure module

- backbone frames使用“黑洞初始化”，这个黑洞初始化把所有的残基都放在全局坐标的坐标原点，然后在后面的每一层计算推走多远



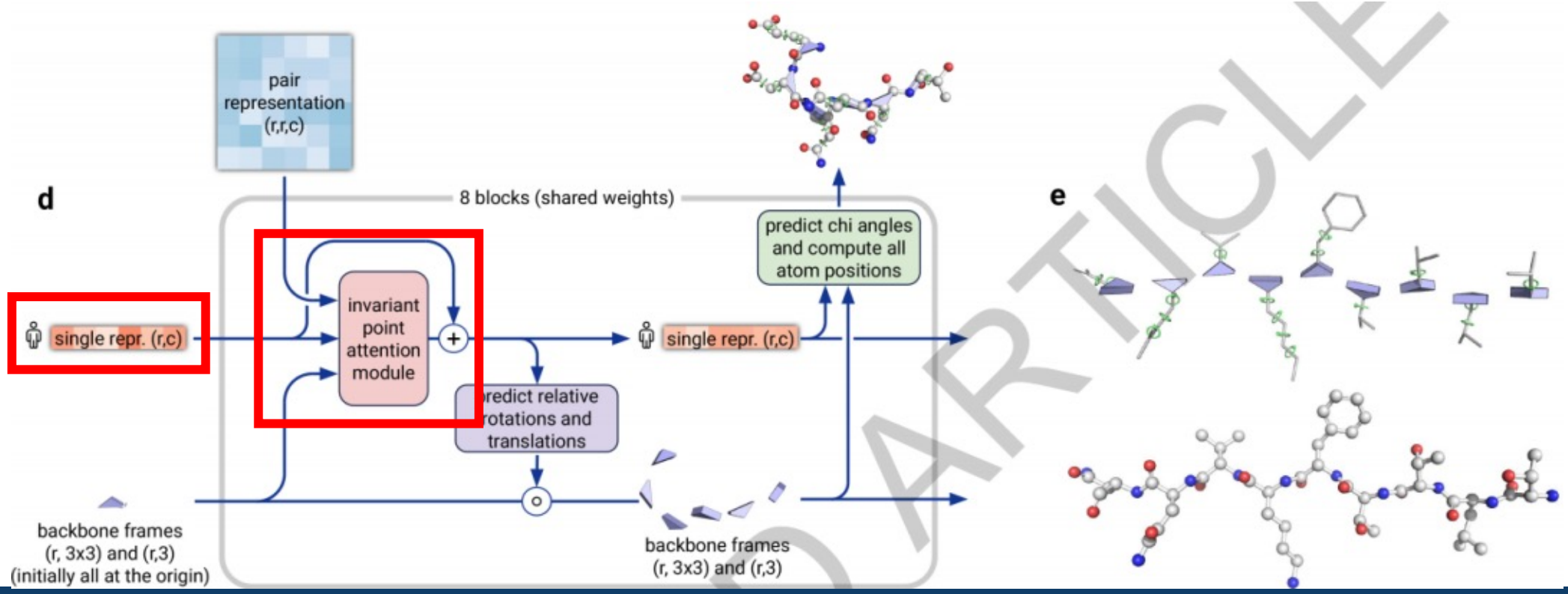
Structure module

- pair features只用来在第一个attention子层中计算bias辅助更新single repr，他们永远不变



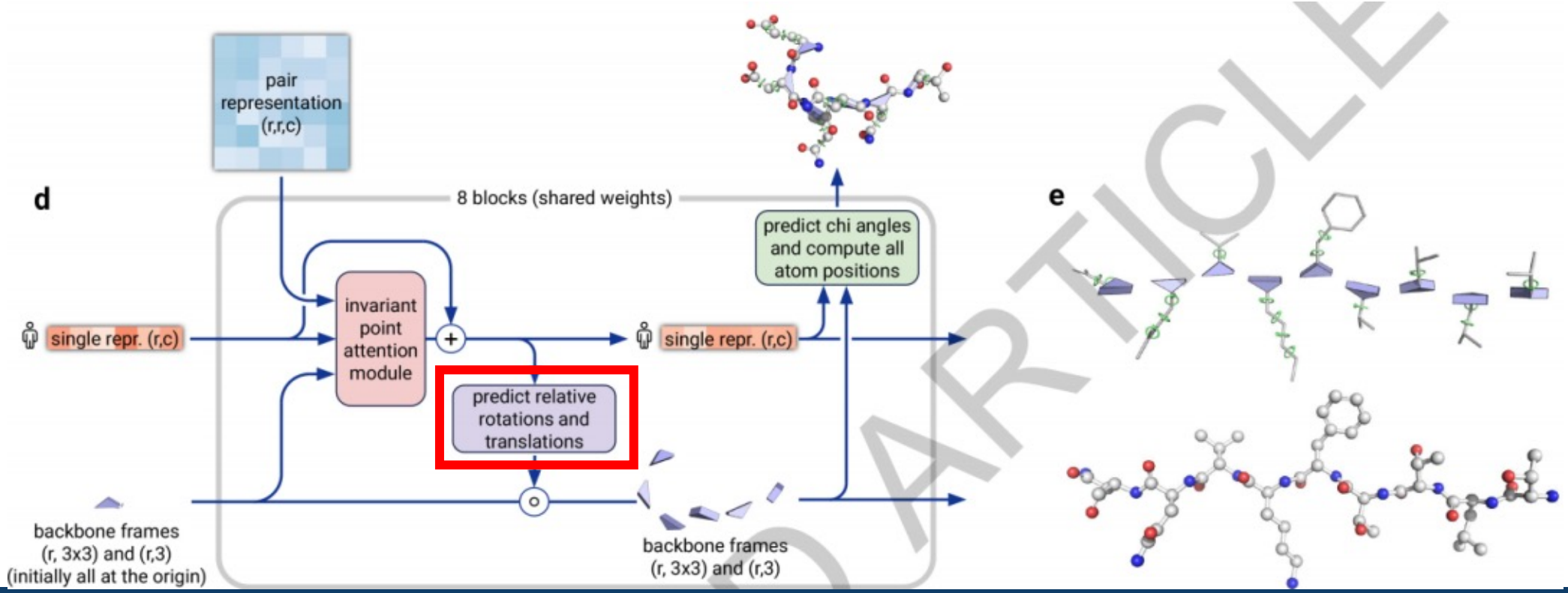
Structure module

- 使用MSA中的第一条而丢弃剩余部分：即只保留目标的这个氨基酸序列，丢掉检索到的其他MSA，称作single repr
- Invariant Point Attention Module子层中，更新single repr



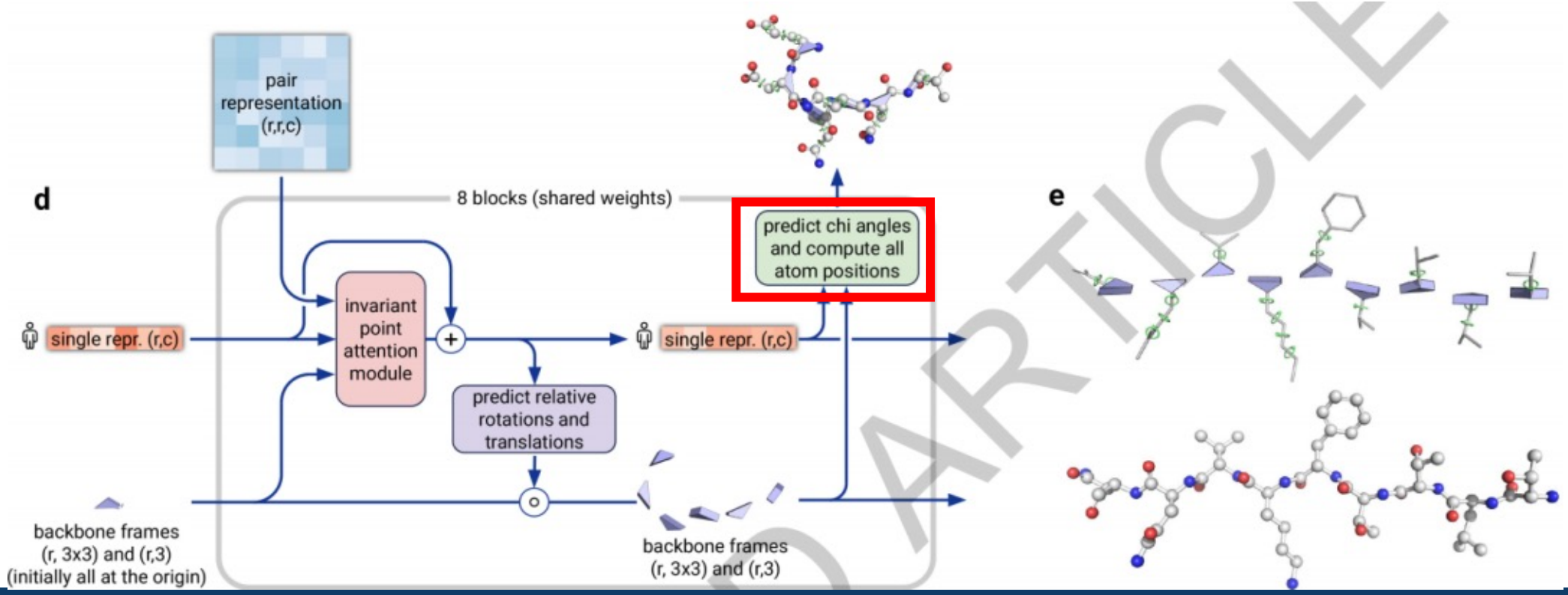
Structure module

- Predict Relative Rotation and Translation子层中，single repr被映射为具体的update frames去更新backbone frames

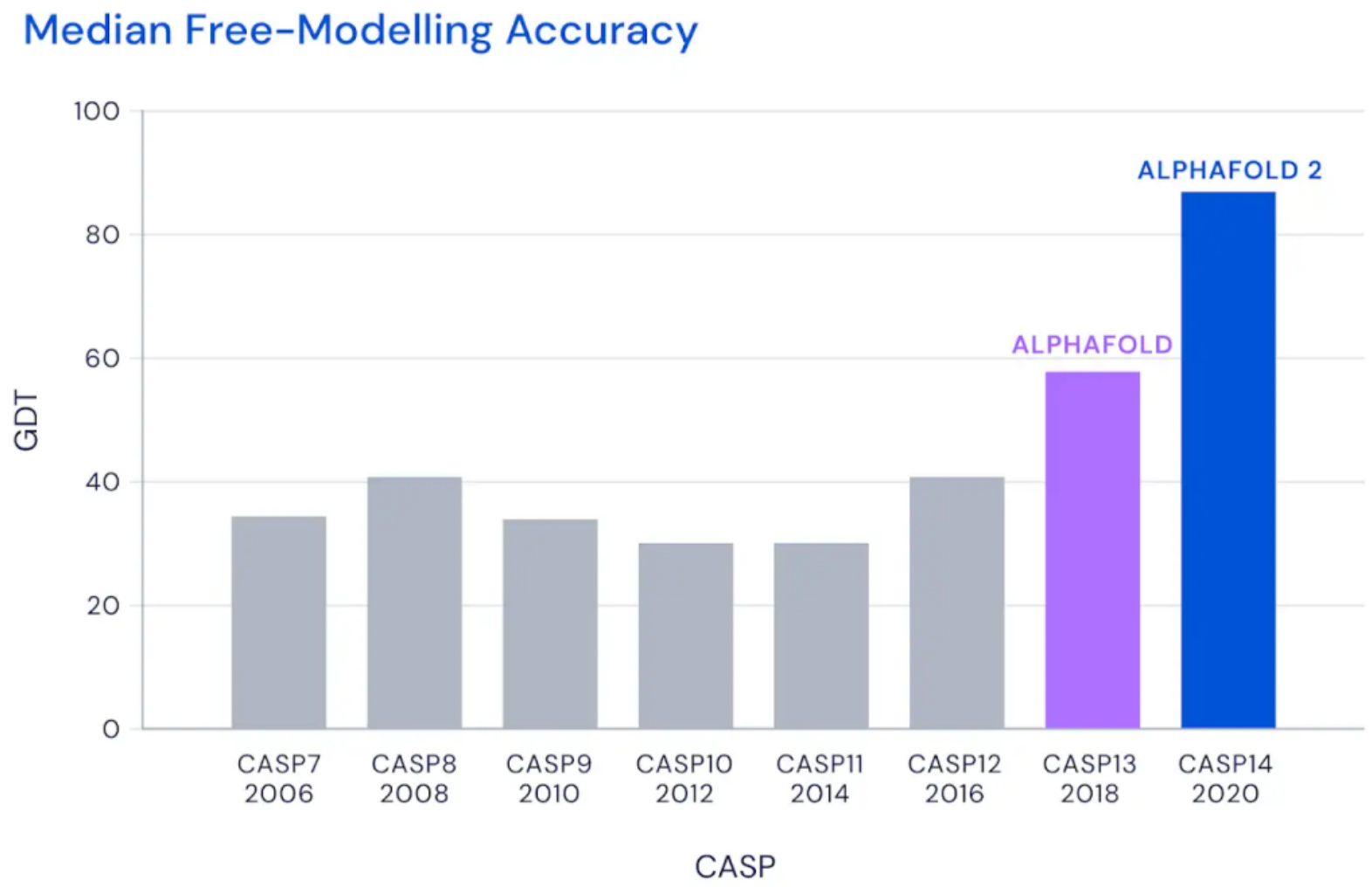


Structure module

- 对于每个残基，我们预测它和相邻残基之间肽键的角度和距离，这个角度和距离是自由的
- 而为了确定每个原子的坐标，残基内部也有若干扭转角度的参数，残基内部确定原子位置仅由这些扭转自由度确定。



效果展示


Median Free-Modelling Accuracy
100
80
60
40
20
0
GDT
ALPHAFOLD 2
ALPHAFOLD
CASP7
2006
CASP8
2008
CASP9
2010
CASP10
2012
CASP11
2014
CASP12
2016
CASP13
2018
CASP14
2020
CASP

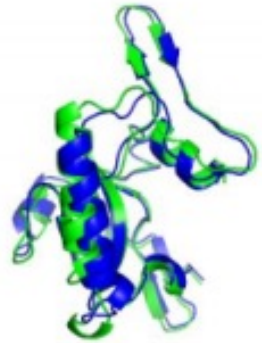
Protein example: T1044 (RNA Polymerase)

- Folding as a single long chain
- Long-chain-trained model trained after the submission

Individual domains

T1041

T1042



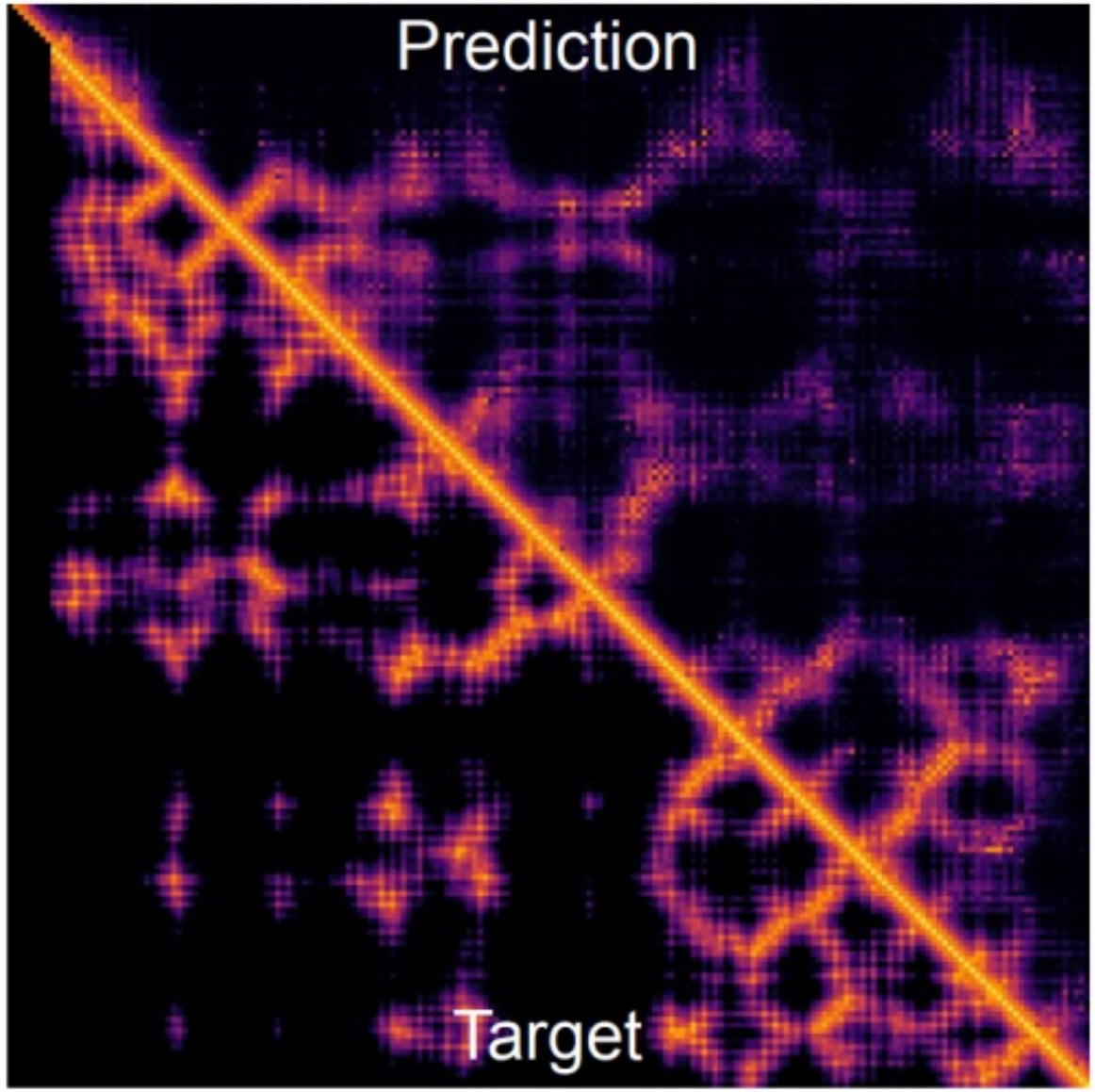
T1043

6VR4: Leiman, P.G., et al. Virion-packaged DNA-dependent RNA polymerase of crAss-like phage phi14:2 (CASP target). (To be published.)

Ground truth
Prediction

Model interpretability - T1038

T1038



6YA2: Bahat, Y., et al. First structure of a glycoprotein from enveloped plant virus. (To be published.)

Thank you!