物理信息图神经网络及其应用

王石平 教授/博导/智慧地铁福建省高校重点实验室主任

福州大学 计算机与大数据学院

2024/10/28

目录

contents

PART 1

物理信息网络

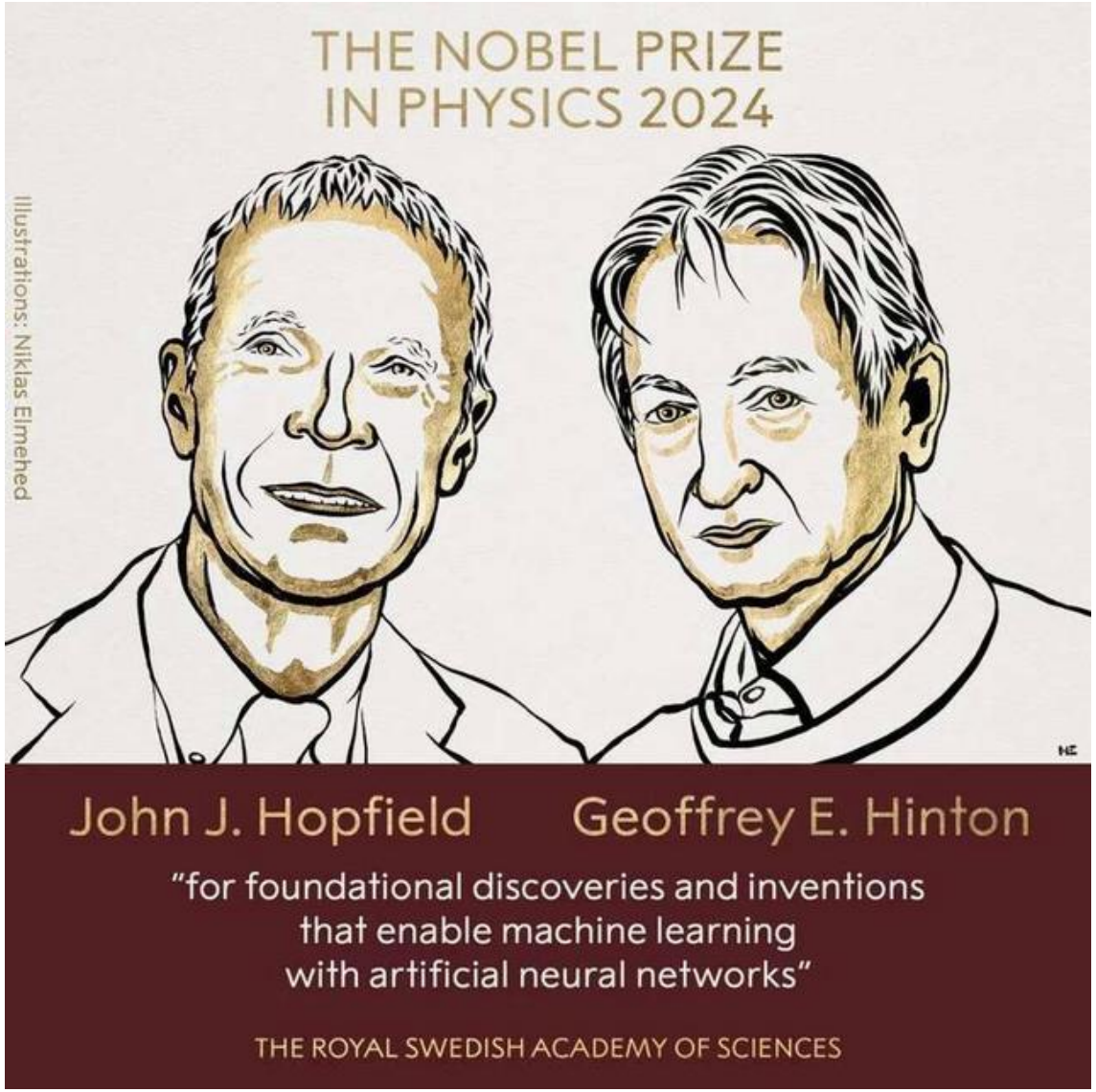
第一部分

目录

contents

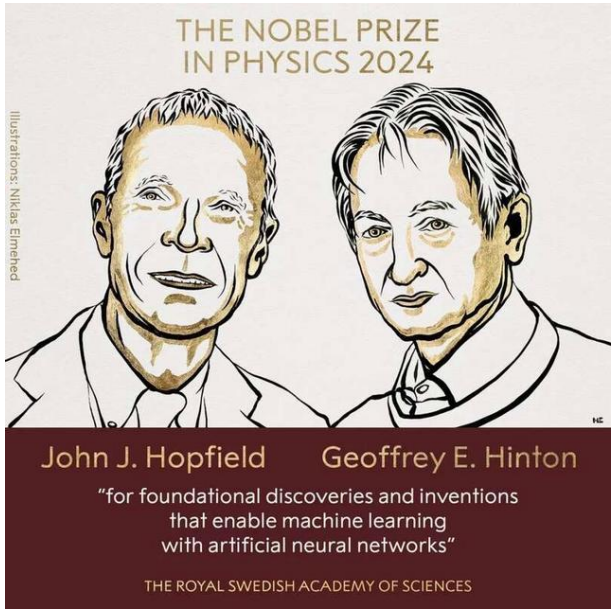
1.1 物理信息网络

2024年诺贝尔物理学奖



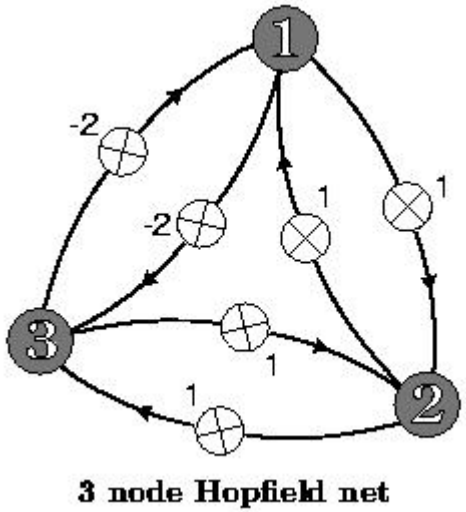
1.1 物理信息网络

2024年诺贝尔物理学奖



Hopfield

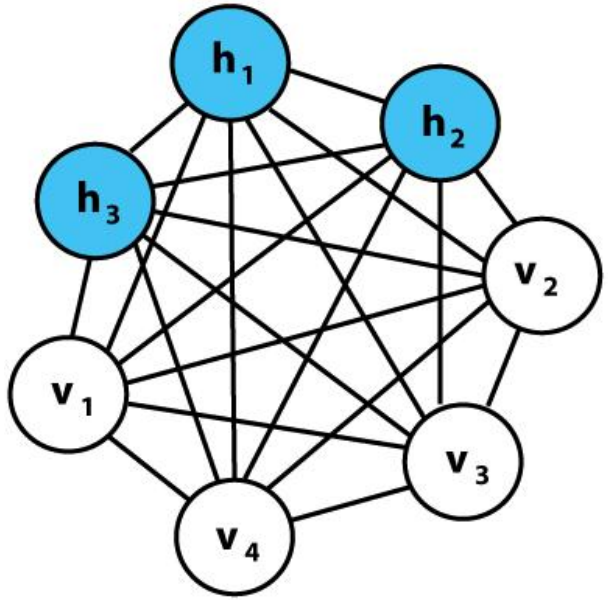
1982年，霍普菲尔德网络



3 node Hopfield net

Hinton

1985年，玻尔兹曼机



1.1 物理信息网络

物理微分方程

物理微分方程是描述自然界中连续物理现象的方程，广泛应用于各个领域，如力学、热学、电磁学、流体力学、量子力学等。微分方程的建立通常是为了描述某个物理系统随时间或空间变化的规律，它将物理量与其导数联系起来，揭示出系统的动态行为。

常见的物理微分方程

常见的微分方程有以下这些：

拉普拉斯方程（Laplace Equation），波动方程（Wave Equation），热传导方程（Heat Equation），薛定谔方程（Schrödinger Equation），纳维-斯托克斯方程（Navier-Stokes Equation）……

■ 拉普拉斯方程

拉普拉斯方程，又名调和方程、位势方程，是一种偏微分方程。因为由法国数学家皮埃尔-西蒙·拉普拉斯首先提出而得名。求解拉普拉斯方程是**电磁学、天文学、热力学和流体力学**等领域经常遇到的一类重要的数学问题，因为这种方程以势函数的形式描写电场、引力场和流场等物理对象（一般统称为“保守场”或“有势场”）的性质。

$$\nabla^2\phi = 0 \text{ 或 } \sum_{i=1}^{n} \frac{\partial^2 u}{\partial x_i^2} = 0$$

拉普拉斯算符(Laplacian) ∇^2 被定义为梯度的散度($\nabla \cdot \nabla$)，有点类似于单变量函数的2阶导数。二阶导数与拉普拉斯算符都是描述速度的变化(加速度)。

■ 一维拉普拉斯方程

$$\frac{\partial^2 u}{\partial x^2} = 0$$

表示在空间上 的“速度”变化率为0，与引入时间维度牛顿第二运动定律$F = ma, F = m\frac{d^2x}{dt^2}$对比，假设$m = 1$，对应于作用力和加速度都等于0情况，蜕化为牛顿第一定律————无外力作用下，物体保持匀速直线运动或静止状态。想象一下，没有开启发动机沿直线高速滑行的太空飞船，也是处于一种均衡状态。

■ 二维拉普拉斯方程

$$\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}=0$$

适用的场景如：室内均衡的气温的Laplacian等于0。Laplacian等于0的函数被称为Harmonic，这种函数描述的场景是：每个点的函数值等于它周围点的平均函数值。皂膜可作为一个直觉的想象。若某点的函数值偏离了周围点的值，均衡就被打破了，偏离的越多，拉普拉斯算子越大，回归均衡的速度越快。

■ 热传导方程

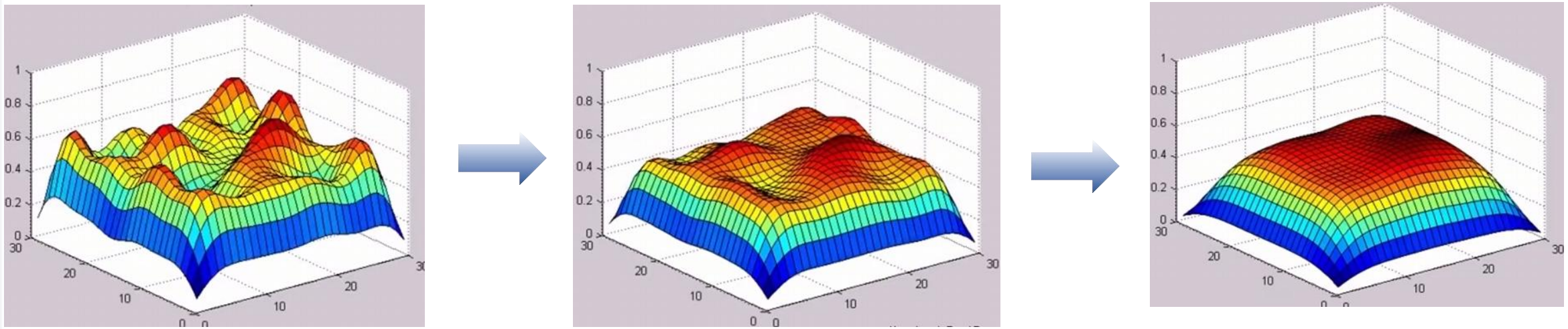
热传导方程由法国物理学家傅里叶（Joseph Fourier）在1822年提出。通过实验观察，傅里叶发现热量从高温区域传递到低温区域，这一现象可以用数学语言来进行表达。傅里叶提出的“傅里叶定律”指出，热流密度与温度梯度成正比，基于这一物理原理，傅里叶推导出了热传导方程。

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x_1^2} + \cdots + \frac{\partial^2 u}{\partial x_n^2}$$

热方程解决了什么问题？简单的说：对于一个温度分布不均匀的物体，通过求解热方程，可以知道它的温度如何随时间和空间的变化而变化，即温度 $u(x,t)$ 关于时间 t 和空间 x 的函数。

■ 热传导方程的含义

含义很简单，可粗略的认为：温度分布不均的物体的某一点温度的变化速度 $\frac{\partial u}{\partial t}$ 取决于该点与相近点温度均值的差 $\nabla^2 u$。换言之，如果某点的温度远高于周围温度，该点温度会快速下降；反之，若某点温度远低于周围温度，其温度将快速上升。



■ 波动方程

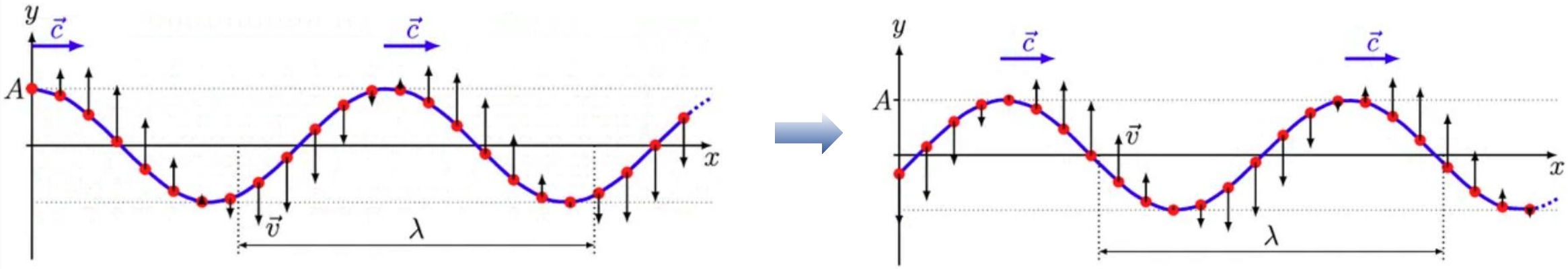
是一种二阶线性偏微分方程，主要描述自然界中的各种的波动现象—正如它们出现在经典物理学中—例如机械波，包括声波、光波、引力波、无线电波、水波、和地震波。波动方程是双曲形偏微分方程的最典型代表，其最简形式可表示为：关于位置x和时间t的标量函数u（代表各点偏离平衡位置的距离）满足：

$$\frac{\partial^2 u}{\partial t^2} = v^2 \nabla^2 u$$

这里v通常是一个固定常数，代表波的传播速率。在常压、20°C的空气中v为343米/秒。

■ 波方程直观理解

波以速度c向右侧传播(1 维)



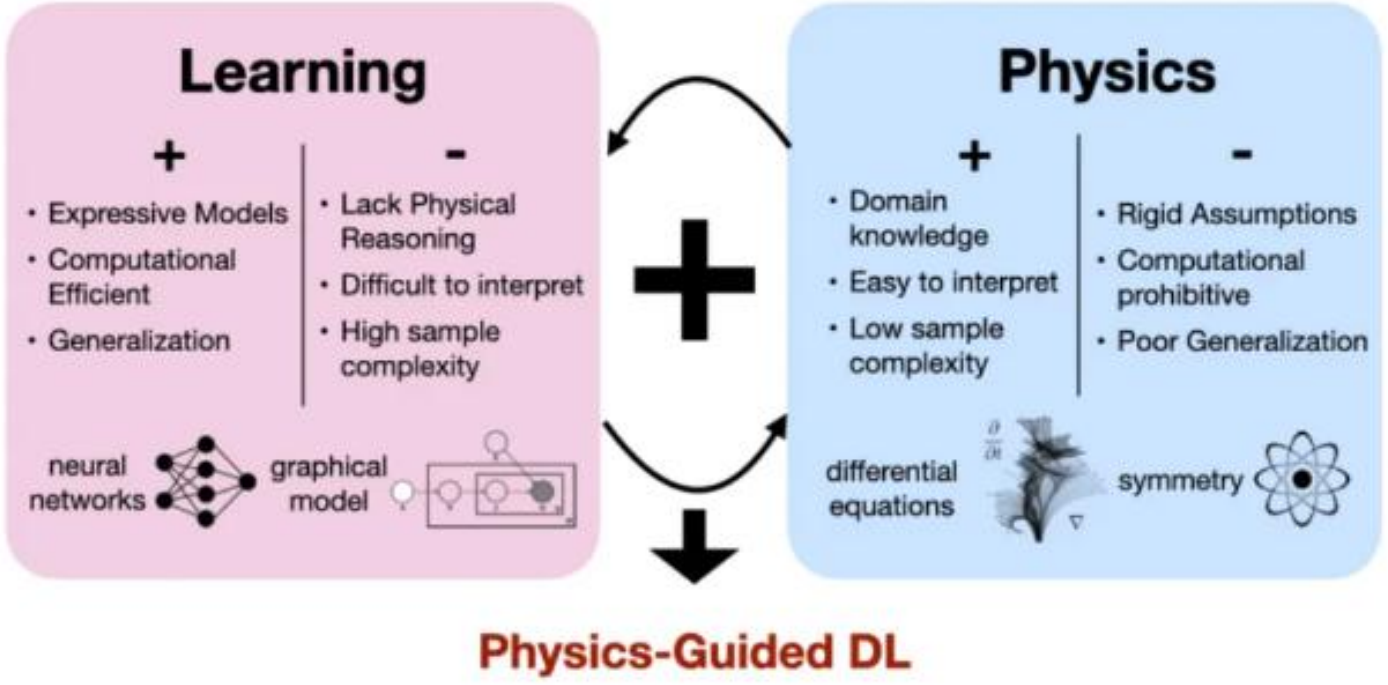
■ 物理信息网络的概念

这些网络结合了物理定律和机器学习方法，特别是神经网络，用于解决涉及物理现象的问题。物理信息网络的核心思想是通过将已知的物理定律（如微分方程、守恒定律等）融入神经网络的设计与训练中，从而提高模型的精度、鲁棒性和可解释性。

物理引导的神经网络最早是在科学计算和数值模拟领域提出的。随着神经网络在数据驱动模型上的成功，人们意识到将已知的物理规律与机器学习方法结合，能够有效提高模型在复杂物理问题中的预测能力。

■ 物理信息网络的优点

物理引导网络概述它结合了基于学习和基于物理的方法的互补优势，在与物理定律相关的同时，产生了高效、富有表现力、可推广的模型。



PART 2

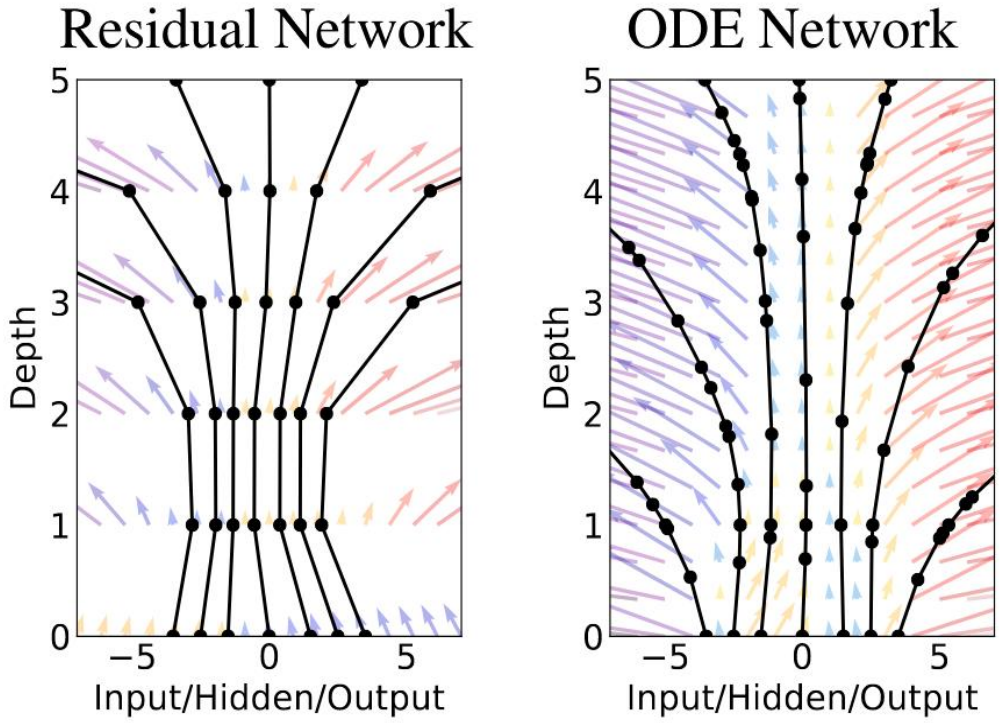
与神经网络的关联

目录

contents

第二部分

2.1.1 什么是Neural ODE？——从离散到连续的动态系统建模



左：残差网络定义有限变换的离散序列。

右图：ODE 网络定义了一个向量场，它不断变换状态。

圆圈代表评估位置。

神经常微分方程（Neural Ordinary Differential Equations，Neural ODEs）是一类深度学习模型，将神经网络的隐藏状态视为由常微分方程（ODE）描述的连续过程。传统的深度学习模型，如前馈神经网络和循环神经网络，依赖于层与层之间的离散转换，而神经ODE则将这种转换视为时间上的连续流动。

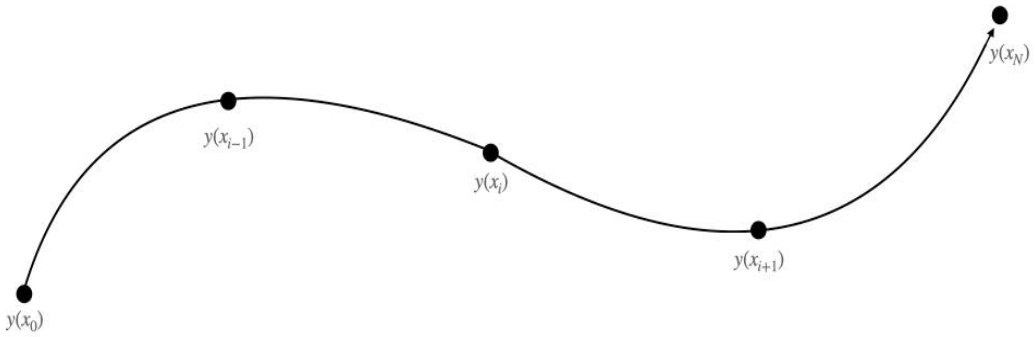
对比残差网络和ODE网络：

Residual Network：是通过残差连接的离散神经网络，适合处理图像、视频等固定层次的任务，设计思想是在深层网络中保持信息流的顺畅。

ODE Network（Neural ODE）：将网络建模转化为连续的微分方程，适合动态系统建模和连续时间的任务，主要优势在于能通过ODE求解器精确描述动态变化。

- Chen, Ricky TQ, et al. Neural ordinary differential equations. In: *NIPS*, 2018.

2.1.2 连续时间建模



一条轨迹
$y(x_0), \ldots, y(x_{i-1}), y(x_i), \ldots, y(x_N)$

ODE（Ordinary Differential Equation，常微分方程）是描述一个变量随另一个变量（通常是时间）变化的数学方程形式为：

$$\frac{dy(t)}{dt} = f(y(t), t)$$

其中，$y(t)$是时间 t 时的状态，方程描述了状态 $y(t)$ 的变化速率 $\frac{dy(t)}{dt}$ 。

残差网络、循环神经网络解码器和归一化流等模型通过将一系列变换组合到隐藏状态来构建复杂的变换：

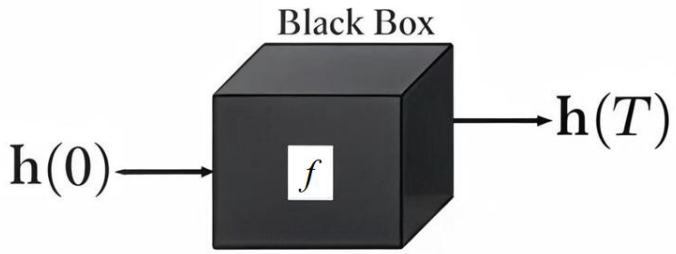
$$\mathbf{h}_{t+1} = \mathbf{h}_t + f(\mathbf{h}_t, \theta_t)$$

当添加更多层并采取更小的步骤时会发生什么？在极限情况下，使用神经网络指定的常微分方程参数化隐藏单元的连续动态：

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

- Chen, Ricky TQ, et al. Neural ordinary differential equations. In：*NIPS*, 2018.

2.1.2 连续时间建模



- 从输入层 $\mathbf{h}(0)$ 开始，可以将输出层 $\mathbf{h}(T)$ 定义为该 ODE 初始值问题在某个时间 T 的解。该值可以通过黑盒微分方程求解器计算，该求解器在必要时评估隐藏单元动力学 f 以确定具有所需精度的解。
- 将 ODE 求解器视为黑盒，并使用伴随灵敏度方法计算梯度。此方法通过及时向后求解第二个增强型 ODE 来计算梯度，并且适用于所有 ODE 求解器。这种方法随问题大小线性扩展，内存成本低，并且明确控制数值误差。

- Chen, Ricky TQ, et al. Neural ordinary differential equations. In: *NIPS*, 2018.

2.1.3 Neural ODE的实现

➢ Forward Process（前向计算过程）

ODE Net的前向计算，就相当于求解一个常微分方程初值问题。

➢ Backward Process（后向计算过程）

考虑优化标量值损失函数 $L(\cdot)$，其输入是 ODE 求解器的结果：

$$L(\mathbf{z}(t_1)) = L\left(\mathbf{z}(t_0) + \int_{t_0}^{t_1} f(\mathbf{z}(t), t, \theta)dt\right) = L(ODESolve(\mathbf{z}(t_0, f, t_0, t_1, \theta))$$

后向过程的目的是为了计算模型参数θ的梯度$\frac{dL(\mathbf{z}(t_1))}{d\theta}$，然后对$\theta$进行更新。

- Chen, Ricky TQ, et al. Neural ordinary differential equations. In：*NIPS*, 2018.

2.1.4 Neural ODE实例

反向传播算法中存在需要保存前向计算过程中的activation的问题，如果计算图很复杂，要保存的activation会很多，导致显存不足，因此可以使用adjoint method（伴随方法）来处理复杂的计算图，其好处就是不需要在forward过程中保存activation也能计算梯度。

adjoint method就是把backward过程看作一个新的常微分方程初值问题（IVP），直接用ODE Solver计算得到梯度值。

Forward过程的ODE：

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

为$\mathbf{z}(t)$创建一个伴随状态：

$$\mathbf{a}(t) = \frac{dL}{d\mathbf{z}(t)}$$

则$\mathbf{a}(t)$的变化过程可以用一个新的ODE表示：

$$\frac{d\mathbf{a}(t)}{dt} = -\mathbf{a}(t)\frac{\partial f(\mathbf{z}(t), t, \theta)}{\partial \mathbf{z}(t)}$$

Backward对应的IVP，初始条件是 $\mathbf{a}(t_1) = \frac{dL}{d\mathbf{z}(t_1)}$

利用ODE Solver求解$\mathbf{a}(t_0)$：

$$\mathbf{a}(t_0) = \mathbf{a}(t_1) + \int_{t_1}^{t_0} \frac{d\mathbf{a}(t)}{dt} dt = \mathbf{a}(t_1) - \int_{t_0}^{t_1} \frac{\partial f(\mathbf{z}(t), t, \theta)}{\partial \mathbf{z}(t)} dt$$

2.1.4 Neural ODE实例

计算相对于参数θ的梯度需要计算取决于$z(t)$和$a(t)$的积分：

$$\frac{dL}{d\theta} = \int_{t_1}^{t_0} \mathbf{a}(t)^{\mathrm{T}} \frac{\partial f(\mathbf{z}(t), t, \theta)}{\partial \mathbf{z}(t)} dt$$

其中的向量雅可比乘积$\mathbf{a}(t)^{\mathrm{T}} \frac{\partial f}{\partial z}$和$\mathbf{a}(t)^{\mathrm{T}} \frac{\partial f}{\partial \theta}$可以通过自动微分有效进行评估。用于求解$\mathbf{z}$, $\mathbf{a}$ 和 $\frac{\partial L}{\partial \theta}$ 的所有积分都可以通过一次调用 ODE 求解器来计算，该求解器将原始状态、伴随和其他偏导数连接到单个向量中。算法 1 展示了如何构造必要的动力学，并调用 ODE 求解器一次性计算所有梯度。

Algorithm 1 Reverse-mode derivative of an ODE initial value problem

Input: dynamics parameters θ, start time t_0, stop time t_1, final state $\mathbf{z}(t_1)$, loss gradient $\partial L / \partial \mathbf{z}(t_1)$

$s_0 = [\mathbf{z}(t_1), \frac{\partial L}{\partial \mathbf{z}(t_1)}, \mathbf{0}_{|\theta|}]$ ▷ Define initial augmented state

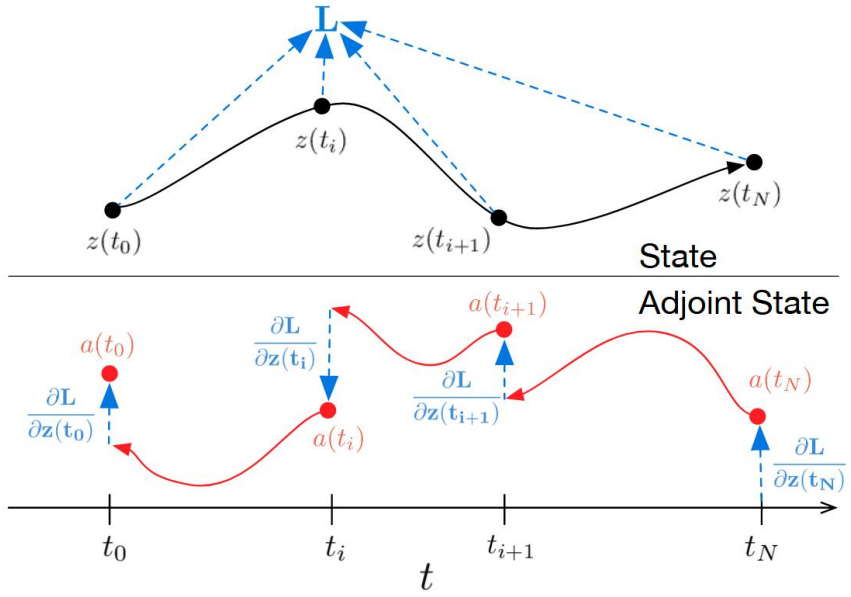
def aug_dynamics($[\mathbf{z}(t), \mathbf{a}(t), \cdot], t, \theta$): ▷ Define dynamics on augmented state

  return $[f(\mathbf{z}(t), t, \theta), -\mathbf{a}(t)^{\mathrm{T}} \frac{\partial f}{\partial \mathbf{z}}, -\mathbf{a}(t)^{\mathrm{T}} \frac{\partial f}{\partial \theta}]$ ▷ Compute vector-Jacobian products

$[\mathbf{z}(t_0), \frac{\partial L}{\partial \mathbf{z}(t_0)}, \frac{\partial L}{\partial \theta}] = \mathrm{ODESolve}(s_0, \text{aug\_dynamics}, t_1, t_0, \theta)$ ▷ Solve reverse-time ODE

return $\frac{\partial L}{\partial \mathbf{z}(t_0)}, \frac{\partial L}{\partial \theta}$ ▷ Return gradients

2.1.4 Neural ODE实例



大多数 ODE 求解器都可以选择多次输出状态 $\mathbf{z}(t)$。当损失取决于这些中间状态时，反向模式导数必须分解为一系列单独的求解，每个连续的输出时间对之间都有一个求解。在每次观察时，必须在相应的偏导数$\frac{\partial L}{\partial \mathbf{z}(t_i)}$的方向上调整伴随函数。图为 ODE 解的逆模式微分。伴随灵敏度法在时间上向后求解增广 ODE。增强系统包含原始状态和损失相对于状态的敏感性。如果损失直接取决于多个观察时间的状态，则必须在损失相对于每个观察的偏导数的方向上更新伴随状态。

- Chen, Ricky TQ, et al. Neural ordinary differential equations. In：*NIPS*, 2018.

2.1.4 Neural ODE优势

- **内存效率**：由于神经ODE只需要存储最终状态（因为伴随方法的应用），相比传统的深度网络，它们更节省内存，后者需要存储反向传播所需的中间状态。
- **灵活的时间步长**：神经ODE可以对不规则时间间隔的数据进行建模，使其适用于数据点不均匀分布的问题，如某些时间序列或物理过程。
- **自适应计算**：ODE求解器可以根据动态的复杂程度自适应地选择函数评估的次数。当动态简单时，可以节省计算量；当动态复杂时，可以提供更精确的评估。
- **融入先验知识**：神经ODE特别适用于物理信息场景，其中已知动力学的微分方程。这些方程可以与数据驱动的模型相结合，产生更具可解释性和物理一致性的预测。

2.2.1 什么是Neural PDE?

Neural PDE（Neural Partial Differential Equation）是一种利用神经网络解决偏微分方程（PDE）问题的技术。这类方法通常结合了神经网络的灵活性和物理系统中的PDE约束，能够处理许多传统数值方法难以解决的问题，如复杂的边界条件、高维问题或不规则几何结构。

偏微分方程描述了连续物理过程的演化，例如流体动力学、热传导和电磁场等。

PDE的一般形式是：

$$F\left(x_1, x_2, \ldots, x_n, u, \frac{\partial u}{\partial x_1}, \frac{\partial u}{\partial x_2}, \ldots, \frac{\partial^2 u}{\partial x_1^2}, \frac{\partial^2 u}{\partial x_1 \partial x_2}, \ldots\right) = 0$$

其中$x_1, x_2, \ldots, x_n$是自变量（通常表示空间和时间坐标）。u是待求解的函数（即解）。

$\frac{\partial u}{\partial x_1}$，$\frac{\partial^2 u}{\partial x_1^2}$ 等是u的偏导数。

2.2.2 偏微分方程(PDE)

常见PDE类型包括椭圆形方程、抛物型方程、双曲型方程。

椭圆型方程常用于描述稳态问题，如静态热传导问题。经典例子是拉普拉斯方程：

$$\nabla^2 u = 0$$

抛物型方程描述随时间演化的过程，典型的例子是热方程：$\frac{\partial u}{\partial t} = \alpha \nabla^2 u$

双曲型方程常用于描述波动或振动，如波动方程：$\frac{\partial^2 u}{\partial t^2} = c^2 \nabla^2 u$

Neural PDE基本数学表达式：

$$\frac{\partial u}{\partial t} = f(u, \nabla u, t)$$

其中u是解，∇u是梯度，t是时间，f是系统的物理模型。

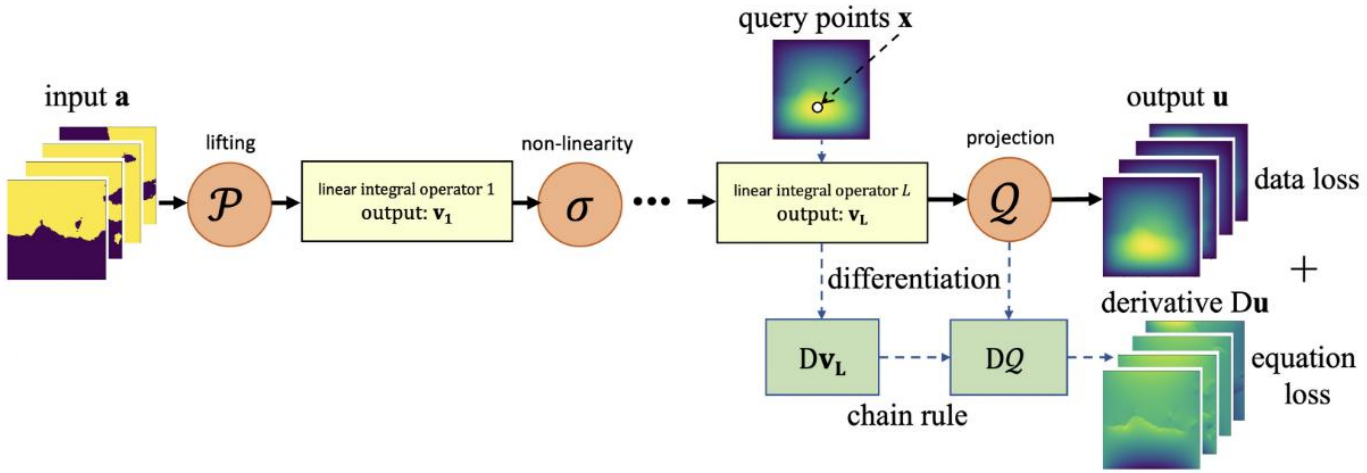
2.2.3 PINN——Neural PDE的实现形式

近年来，神经网络等机器学习技术被用于PDE的近似求解。其中Physics-Informed Neural Networks（PINN）通过结合物理定律与神经网络，提供了一种新颖的PDE求解方式。

PINN框架的主要步骤包括：

- 构建神经网络：定义一个神经网络（通常是全连接网络），将输入的物理变量（如时间、空间坐标）映射为网络输出（PDE解的近似）。
- 定义损失函数：PINN的损失函数包含两部分：数据误差损失（Data Loss）和物理方程损失（Physics Loss）
- 训练过程：在训练过程中，通过反向传播优化网络的权重，使得数据误差和物理方程的残差同时最小化。
- 求解PDE：在训练结束后，神经网络可以用来预测新的输入点上的PDE解。

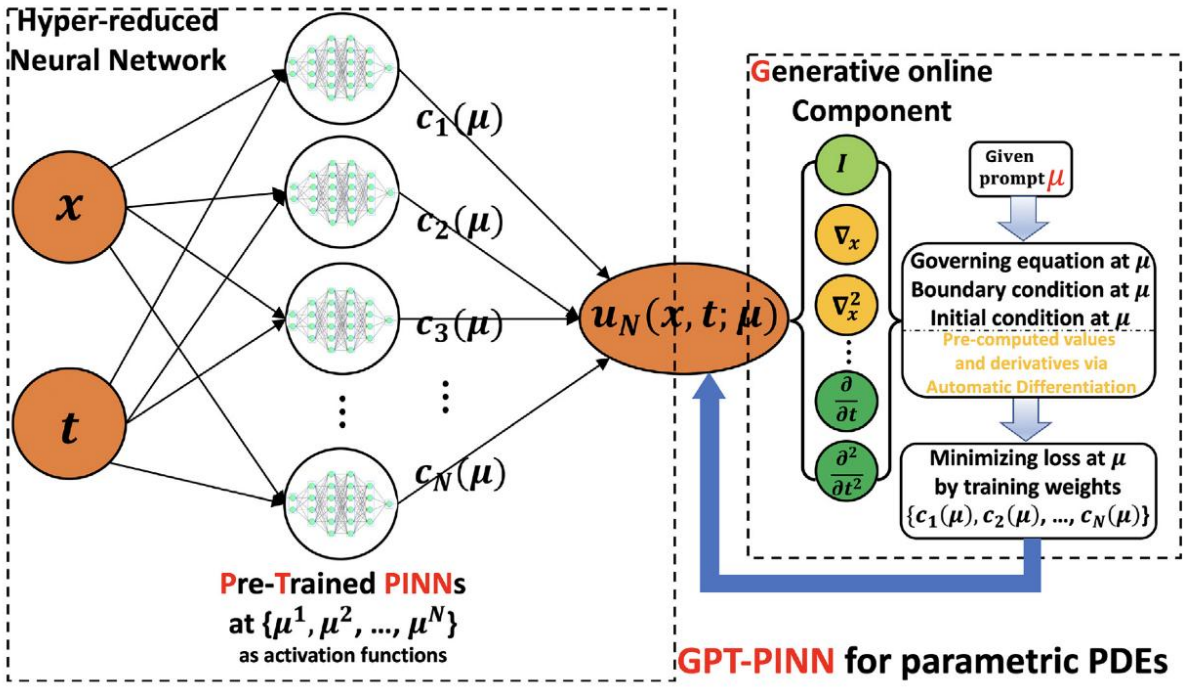
2.2.3 Neural PDE实例——PINO



PINO是一种结合了物理信息和神经网络算子的新方法，用于学习求解参数化偏微分方程（PDEs）的解算子。PINO利用神经算子框架，能够精确地近似给定PDE族的解算子，并且在零样本超分辨率下不降低准确性。PINO 是一种混合方法，它结合了训练数据和物理约束，来学习给定参数化 PDEs 家族的解算子。PINO 使用傅里叶神经算子（FNO）框架，它是一个对任何连续算子的通用逼近器，并且在网格细化的极限中是离散化收敛的。

- Li, Zongyi, et al. Physics-informed neural operator for learning partial differential equations. *ACM/JMS Journal of Data Science*, 2024.

2.2.3 Neural PDE实例——GPT-PINN

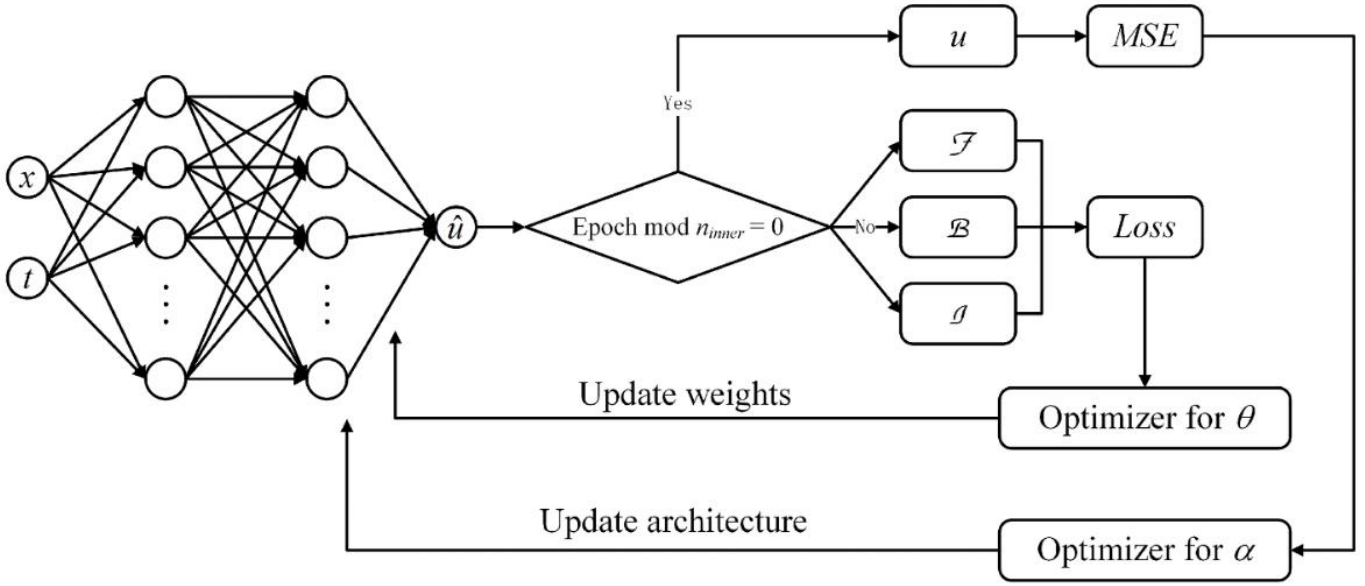


GPT-PINN是一种新型的物理信息神经网络框架，核心思想是结合元学习（meta-learning）和深度学习技术，以提高求解参数化PDEs的效率。

这种方法设计了具有定制激活函数的网络架构，这些激活函数是通过贪婪算法选择的参数值实例化的预训练PINNs。提出了一种全新的元学习范式，即GPT-PINN，用于解决参数化系统中PINNs面临的训练成本和过度参数化的挑战。借鉴了经典的RBM技术，采用策略来显著减小PINNs的规模，并加速使用PINNs求解参数化PDE的过程。

- Chen Y, Koohy S. Gpt-pinn: Generative pre-trained physics-informed neural networks toward non-intrusive meta-learning of parametric pdes. *Finite Elements in Analysis and Design*, 2024.

2.2.3 Neural PDE实例——NAS-PINN



NAS-PINN是一种新型的神经网络架构搜索方法，它用于求解偏微分方程（PDEs）。NAS-PINN通过自动搜索最优的神经网络架构来解决特定的PDEs问题。NAS-PINN通过构建混合操作和引入掩码来实现不同形状张量的相加，将架构搜索问题转化为一个连续的双层优化问题。在PINN框架中引入了空间信息。通过使用数值微分（ND）替代自动微分（AD），成功地将空间信息引入模型中，确保模型得到的解符合物理定律。

- Wang Y, Zhong L. NAS-PINN: neural architecture search-guided physics-informed neural network for solving PDEs. *Journal of Computational Physics*, 2024.

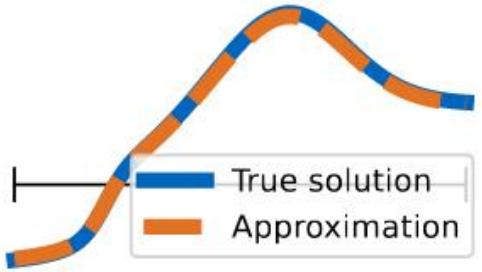
2.2.3 Neural PDE实例

PDE: $-\Delta u = f$

Neural basis: $\phi_1, \phi_2, \phi_3, \ldots$

Ansatz: $\hat{u} = \sum_k c_k \phi_k$

Approximation: $-\sum_k c_k \Delta \phi_k \approx f$



基于神经网络的偏微分方程（PDE）求解方法，提出了一种新的采样方法，通过随机采样内部权重来构建神经网络的基函数，并通过分离变量的方法解决时间相关问题。通过对隐藏层的权重进行采样，提出了一种用于求解偏微分方程（PDE）的新方法。对于静态线性PDE，通过求解一个线性问题来找到解，从而减少了参数的数量。对于一般的时间依赖性PDE，通过在空间中随机采样ansatz函数，将问题简化为求解高维ODE。

2.2.4 Neural PDE应用

Neural PDE方法通过结合数据和物理约束，能够在复杂物理问题的建模和求解上取得优异的表现，特别是在高维问题或不规则几何结构下，Neural PDE展现了出色的适应性。

- **流体力学**：如气流或水流模拟，可以通过Navier-Stokes方程进行建模。
- **热传导**：用于热扩散或温度场计算。
- **电磁场模拟**：解决电磁波传播等问题。
- **材料科学**：预测应力应变的变化。

PART 3

目录

contents

与图神经网络的关联

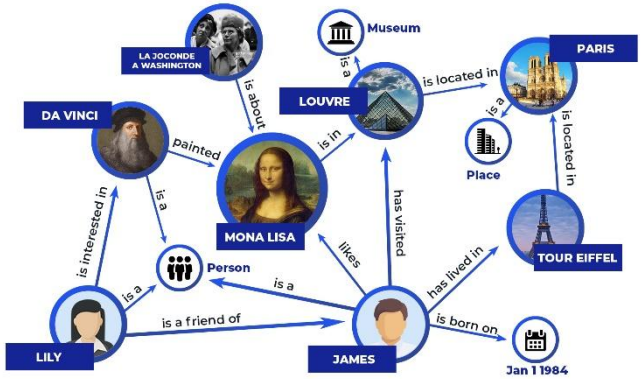
第三部分

3.1 显式图神经网络

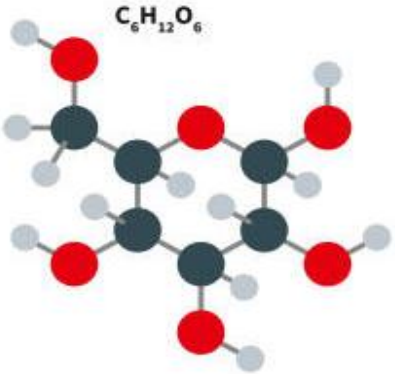
■ 图神经网络研究背景

图神经网络（Graph Neural Networks, GNN）是一类专门用于处理图结构数据的深度学习模型，能够有效捕捉图中节点及其连接关系的复杂信息。因此GNN广泛应用于社交网络分析、推荐系统、化学分子结构预测等领域。

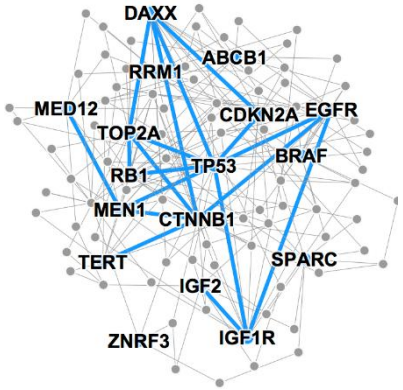
社交网络



知识图谱



分子结构



疾病路径

- Kipf, T. N.; and Welling, M. Semi-supervised Classification with Graph Convolutional Networks. In *ICLR*, 2017.

3.1 显式图神经网络

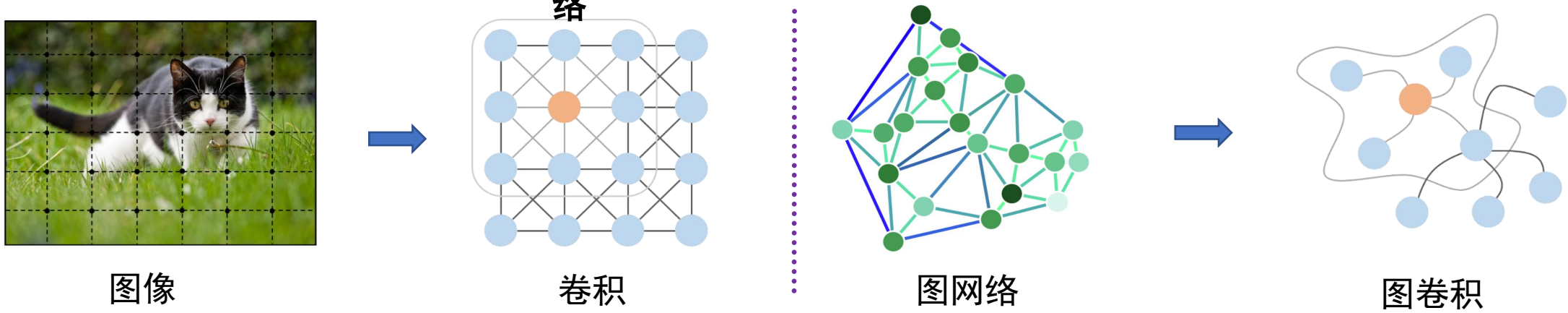
■ 图卷积网络

图卷积网络（Graph Convolutional Network, GCN）是一种扩展卷积操作到图结构数据的模型，它通过节点及其邻居之间的信息传递，生成节点的嵌入表示，图卷积的公式如下：

$$H^{(l+1)} = \sigma\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}\right)$$

H:隐藏层表示，$\tilde{A}$:添加了自连接的无向图的邻接矩阵，$\tilde{D}$:邻接矩阵的度，W:可训练的权重矩阵

卷积神经网络 VS 图卷积网络



- Kipf, T. N.; and Welling, M. Semi-supervised Classification with Graph Convolutional Networks. In *ICLR*, 2017.

图卷积网络面临的挑战

- 在图卷积网络（GCN）中，随着图的层数增加，节点的特征在多次卷积后逐渐趋同，最终导致同一图中的不同节点特征变得无法区分，导致精度下降 ，这类被称为“过平滑”问题。
- 现有的图卷积网络（如GCN和SGC）在特征传播过程中存在效率与表现之间的权衡问题。简单图卷积（SGC）通过减少非线性操作和层数，提高了计算效率，但仍然会受到过平滑问题的影响，简单图卷积公式如下$\hat{Y} = \mathrm{softmax}(S^K XW)$

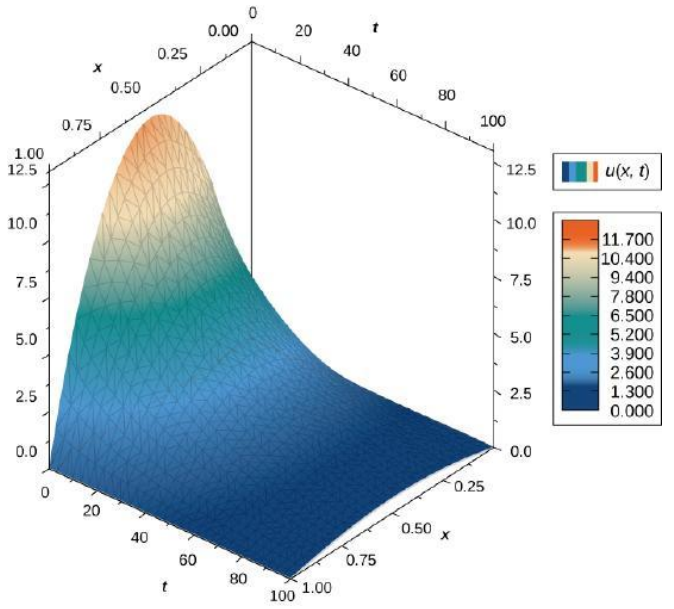
  其中 $S = \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$，$K$表示传播的层数， W为可训练的权重矩阵

- Wu F, Souza A, Zhang T, et al. Simplifying graph convolutional networks. In *ICML*, 2019.
- Wang Y, Wang Y, Yang J, et al. Dissecting the diffusion process in linear graph convolutional networks. In *NIPS*, 2021.

■ 图热方程（Graph Heat Equation）

图热方程（GHE）是一种模拟热量在图中随时间扩散的过程。其背后的核心思想是，节点的特征信息会像热量一样，从一个节点传播到相邻的节点，并逐渐扩散到整个图。用数学方式表示，图热方程可以被描述为一种偏微分方程，其形式类似于经典的热传导方程，表示随时间变化的特征扩散。

$$\begin{cases} \frac{d\mathbf{X}_t}{dt} & = -\mathbf{L}\mathbf{X}_t, \\ \mathbf{X}_0 & = \mathbf{X}, \end{cases}$$



X_t 表示在时间 t 时刻的节点特征表示

$$\mathbf{L} = \widetilde{\mathbf{D}}^{-\frac{1}{2}}\left(\widetilde{\mathbf{D}} - \widetilde{\mathbf{A}}\right)\widetilde{\mathbf{D}}^{-\frac{1}{2}} = \mathbf{I} - \mathbf{S}$$

是图的拉普拉斯矩阵，负责捕捉图中节点之间的邻接关系及其拓扑结构。

- Spielman D. Spectral graph theory. *Combinatorial scientific computing*, 2012.

■ 解耦图卷积（Decoupled Graph Convolution）

由于图热方程是一个连续动力学，因此在实践中可以利用数值方法来对它进行求解。而SGC 可以看作是图热方程的粗略有限差分。通过前向欧拉方法对图热微分方程求解可以得到：

$$\hat{X}_{t+\Delta t} = \hat{X}_t - \Delta t L \hat{X}_t$$
$$\hat{X}_{t+\Delta t} = \hat{X}_t - \Delta t (I - S) \hat{X}_t$$
$$\hat{X}_{t+\Delta t} = \left[(1 - \Delta t) I + \Delta t S\right] \hat{X}_t$$

通过K 次前向步骤的更新，将在终端时间得到最终的特征表示：

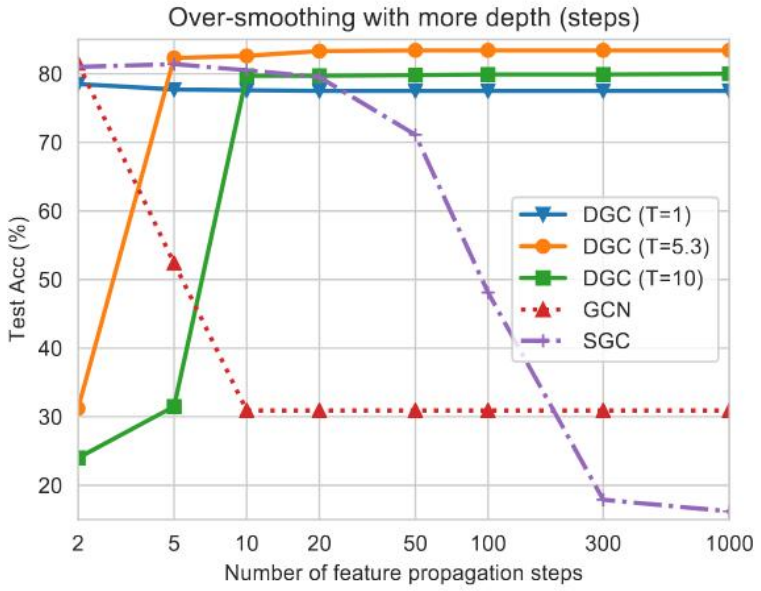
$$\hat{\mathbf{X}}_T = [\mathbf{S}^{(\Delta t)}]^K \mathbf{X}$$

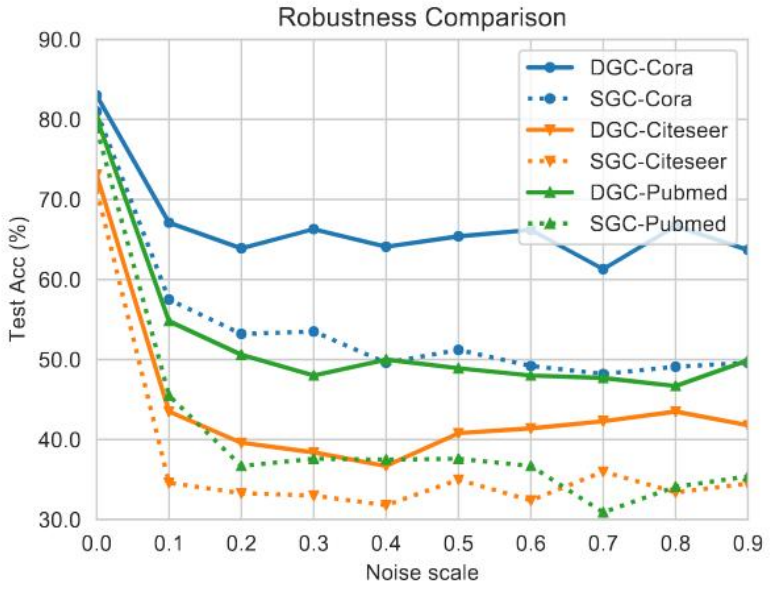
其中 $\mathbf{S}^{(\Delta t)} = (1 - \Delta t)\mathbf{I} + \Delta t \mathbf{S}$ ，T为终端时间，K为传播次数，$\Delta t = T/K$

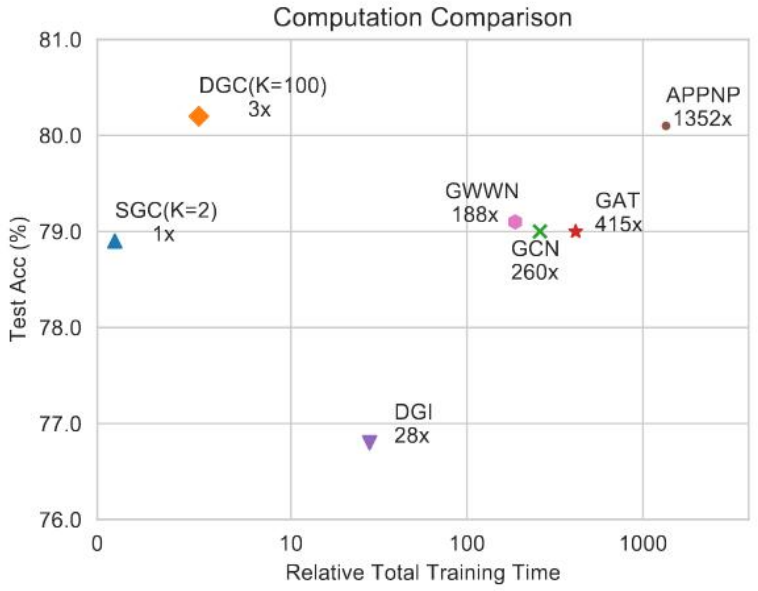
• Wang Y, Wang Y, Yang J, et al. Dissecting the diffusion process in linear graph convolutional networks. In *NIPS*, 2021.

■ 解耦图卷积（Decoupled Graph Convolution）

基于图热方程的DGC可以有效的缓解过平滑现象，在噪声环境下体现出良好的鲁棒性，并且大大的提升了运算效率。







- Wang Y, Wang Y, Yang J, et al. Dissecting the diffusion process in linear graph convolutional networks. In *NIPS*, 2021.

■ 状态空间模型（State Space Model）

状态空间模型是一种序列模型，通常被称为线性时不变系统，它将输入序列$x(t)$映射到响应序列$y(t)$中。具体来说，SSM 使用潜在状态 $h(t)$，演化参数**A**和投影参数**B**、**C**进行更新：

$$h'(t) = \mathbf{A}\, h(t) + \mathbf{B}\, x(t),$$
$$y(t) = \mathbf{C}\, h(t).$$

为缓解深度学习设置中求解上述微分方程的难度，通常采用参数Δ对上述系统进行离散化处理：

$$h_t = \bar{\mathbf{A}}\, h_{t-1} + \bar{\mathbf{B}}\, x_t,$$
$$y_t = \mathbf{C}\, h_t,$$

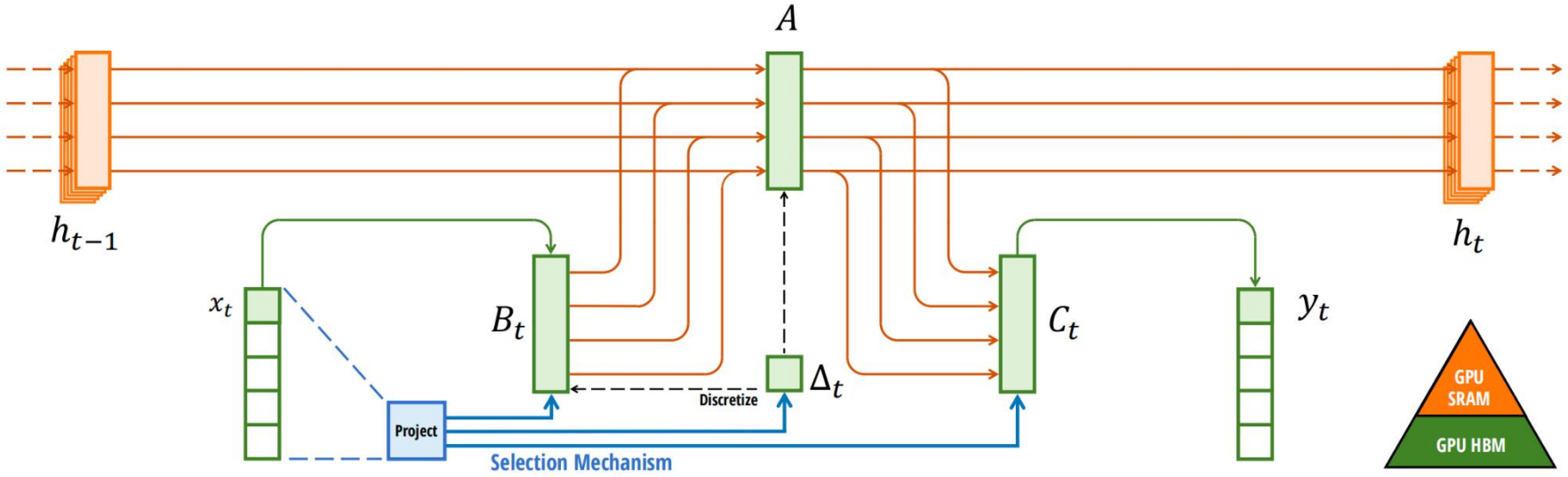
其中 $\bar{\mathbf{A}} = \exp(\Delta \mathbf{A}), \bar{\mathbf{B}} = (\Delta \mathbf{A})^{-1} (\exp(\Delta \mathbf{A} - I)) \cdot \Delta \mathbf{B}.$

- Behrouz A, Hashemi F. Graph mamba: Towards learning on graphs with state space models. In *KDD*, 2024.

■ 图曼巴网络（Graph Mamba Network）

由于在图数据的非因果结构与顺序数据模型不兼容的问题上。现有的模型在处理复杂的长程依赖图结构时效率较低，缺乏有效的选择机制，因此难以对图数据进行高效学习。

Mamba状态转移示意图：



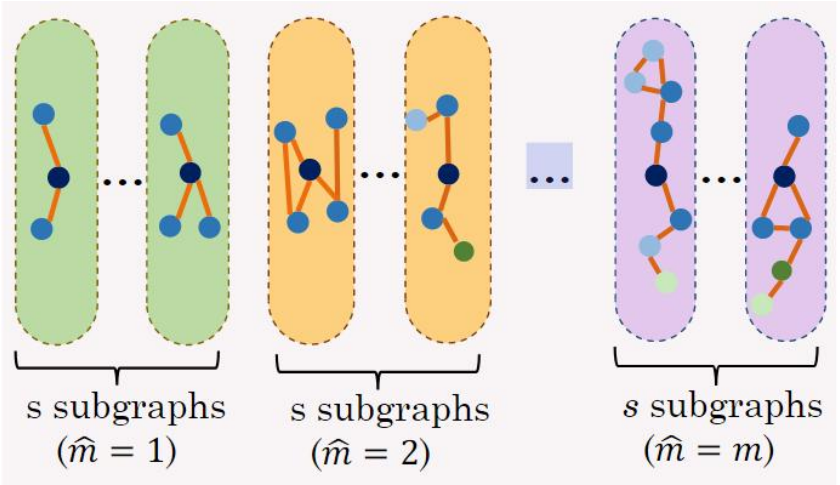
- Behrouz A, Hashemi F. Graph mamba: Towards learning on graphs with state space models. In *KDD*, 2024.

■ 图曼巴网络（Graph Mamba Network）

长序列 Mamba 显示了较长序列的性能改进， Graph Mamba通过为每个节点进行随机游走，并从其邻居节点构建令牌子图序列，这些序列随后用于特征编码，子图的长度为m。

$$G\left[T_{\hat{m}}(v)\right] = G\left[\bigcup_{i=0}^{M} T_{\hat{m},i}(v)\right]$$

$$G[T_0(v)], \underbrace{G[T_1^1(v)], \ldots, G[T_1^s(v)]}_{s \text{ times}}, \ldots, \underbrace{G[T_m^1(v)], \ldots, G[T_m^s(v)]}_{s \text{ times}}$$

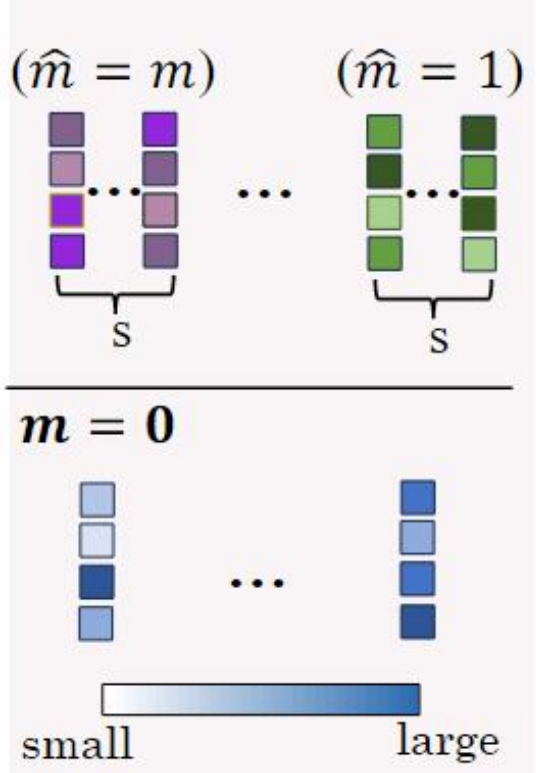


并对每种不同步长的子图重复选择 s 次。

• Behrouz A, Hashemi F. Graph mamba: Towards learning on graphs with state space models. In *KDD*, 2024.

■ 图曼巴网络（Graph Mamba Network）

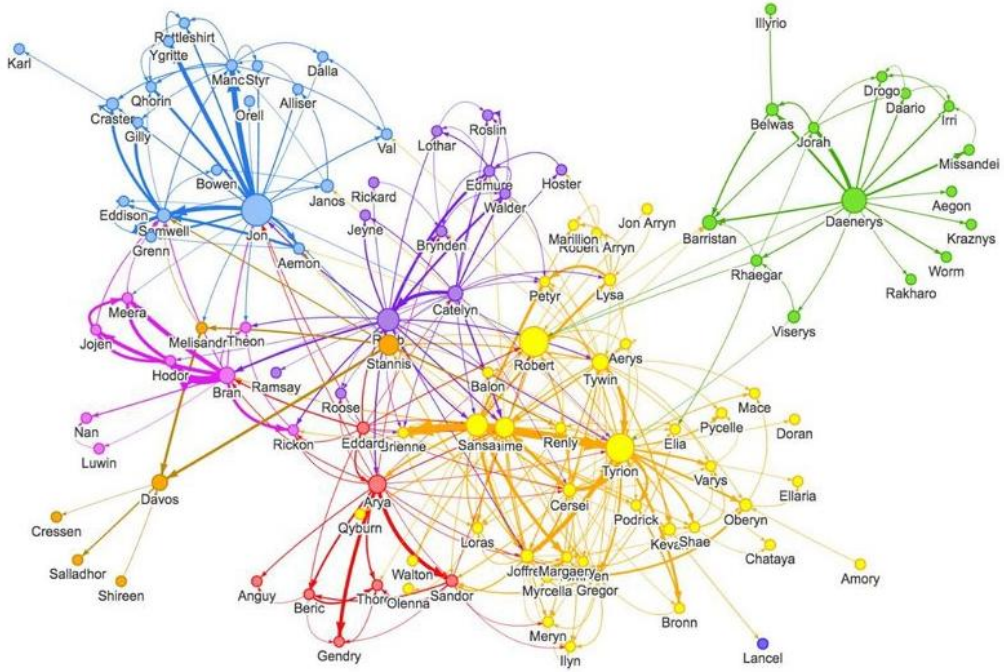
- 节点的度及子图的步长很大程度上反应一个子图及节点的信息量，由于步长小的标记对上下文的信息较少，而步长大的标记具有几乎整个序列的信息。
- 图曼巴网络则根据子图的距离信息将其从外向内排序，以便较内层的子图能够携带关于全局结构的信息，而较外层的子图则能够更加依赖局部信息。
- 考虑到SSM属于循环模型并且需要有序输入，而图结构数据没有任何顺序并且需要编码器的配合处理。为此，图曼巴网络并使用两个循环扫描模块分别扫描两个不同子图排序方向的数据。



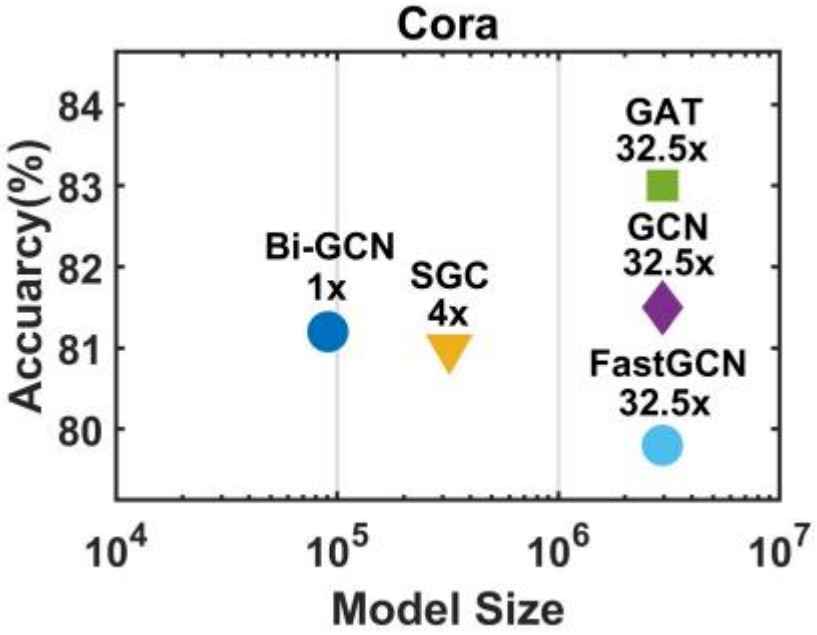
• Behrouz A, Hashemi F. Graph mamba: Towards learning on graphs with state space models. In *KDD*, 2024.

3.2 隐式图神经网络（均衡图神经网络）

■ **显式图神经网络存在的问题：**

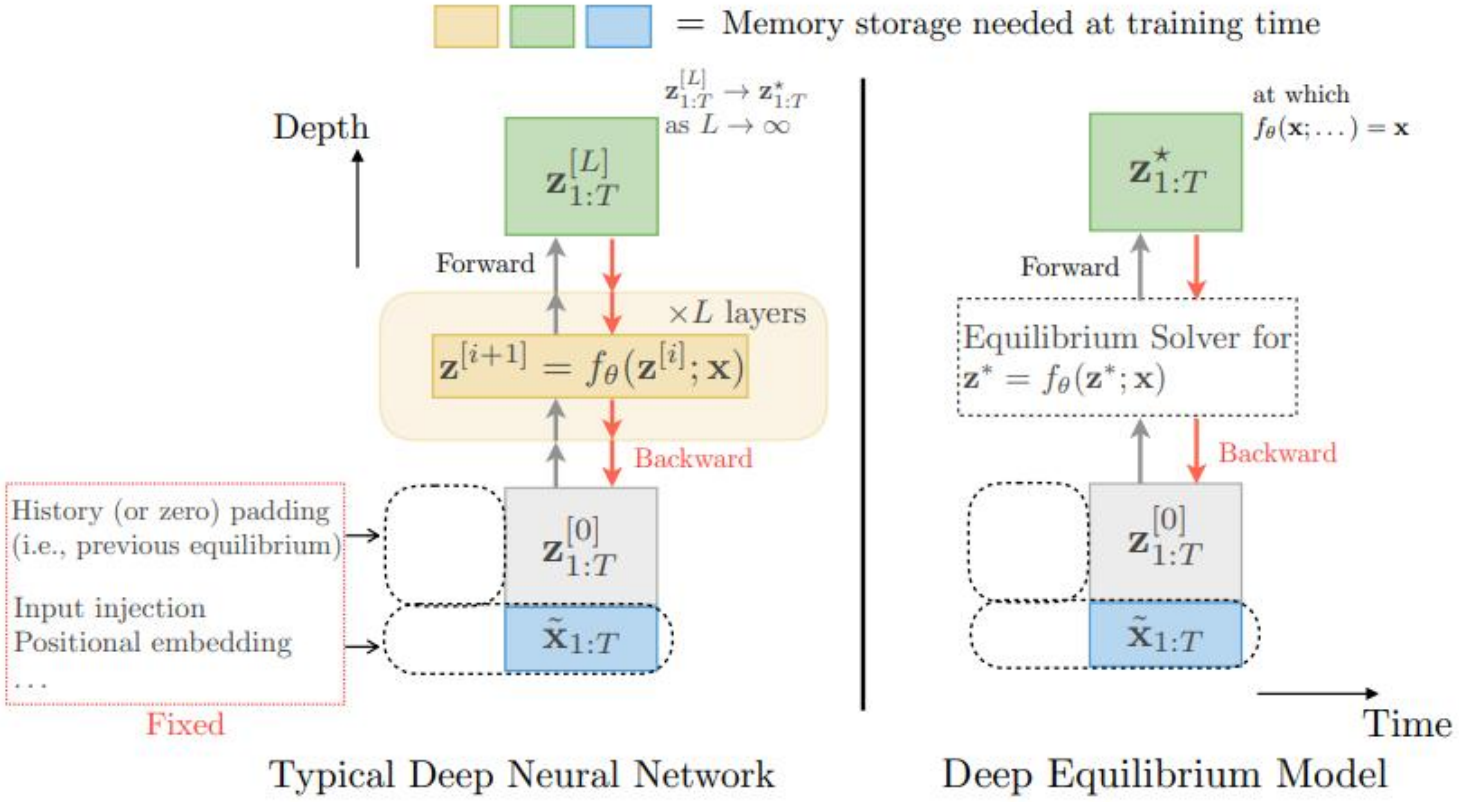


1. 高阶信息难以捕获



2. 爆炸的网络参数

3.2 隐式图神经网络



显示神经网络 vs 隐式神经网络

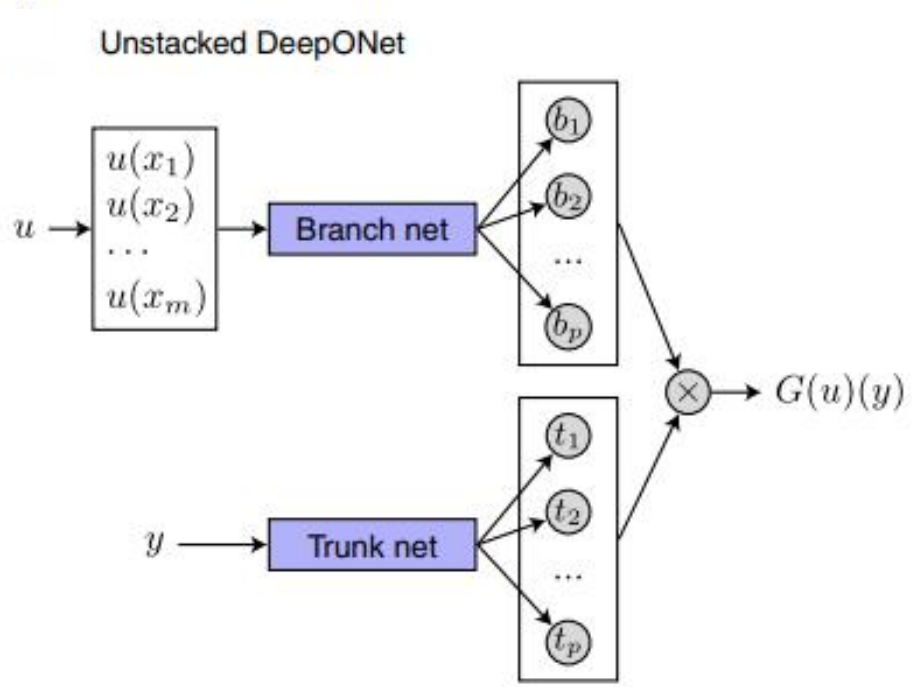
- Bai, Shaojie, J. Zico Kolter, and Vladlen Koltun. Deep equilibrium models. In: *NIPS*, 2019.

3.2 隐式图神经网络

$$\begin{cases} \frac{\mathrm{d}}{\mathrm{d}x}\mathbf{s}(x) = \mathbf{g}(\mathbf{s}(x), u(x), x) \\ \mathbf{s}(a) = \mathbf{s}_0 \end{cases}$$

网络拟合

$$G(u)(y) \approx \sum_{k=1}^{p} \underbrace{b_k(u(x_1), u(x_2), \ldots, u(x_m))}_{\text{branch}} \underbrace{t_k(y)}_{\text{trunk}}$$



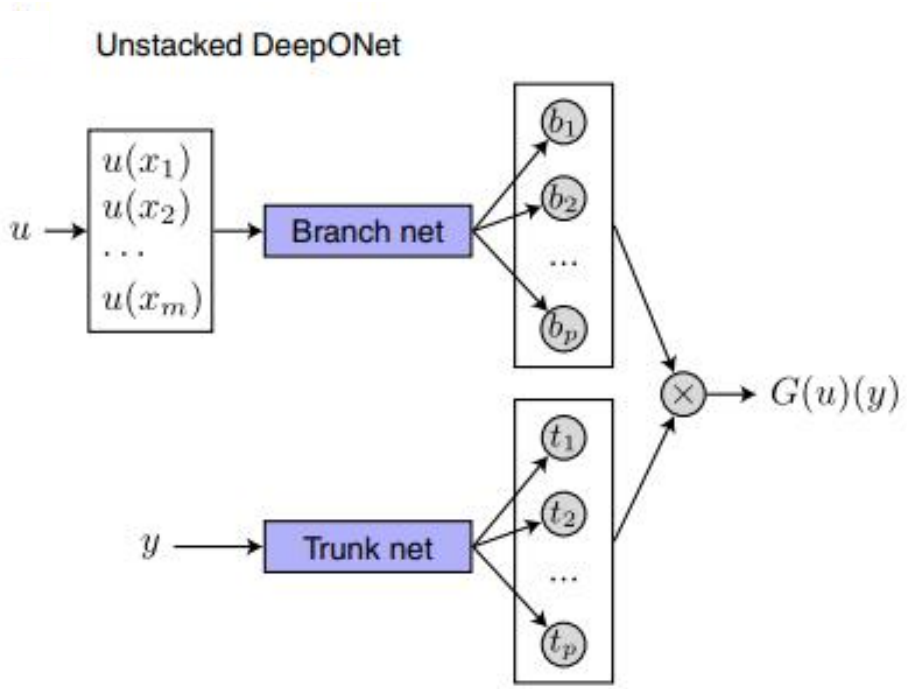
对于一个偏微分方程，任何的边界或初始条件，

这个网络都能获得这个偏微分方程的解

DeepONets 网络结构

- Lu, Lu, et al. Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 2021.

3.2 隐式图神经网络



优化策略

1. 大多数稳态偏微分方程的解可以表示为非线性算子的不动点

2. 使用不动点方程求解器直接求解稳态偏微分方程的解

3. 使用隐式算子层求解无限深度不动点，网络训练参数大大降低

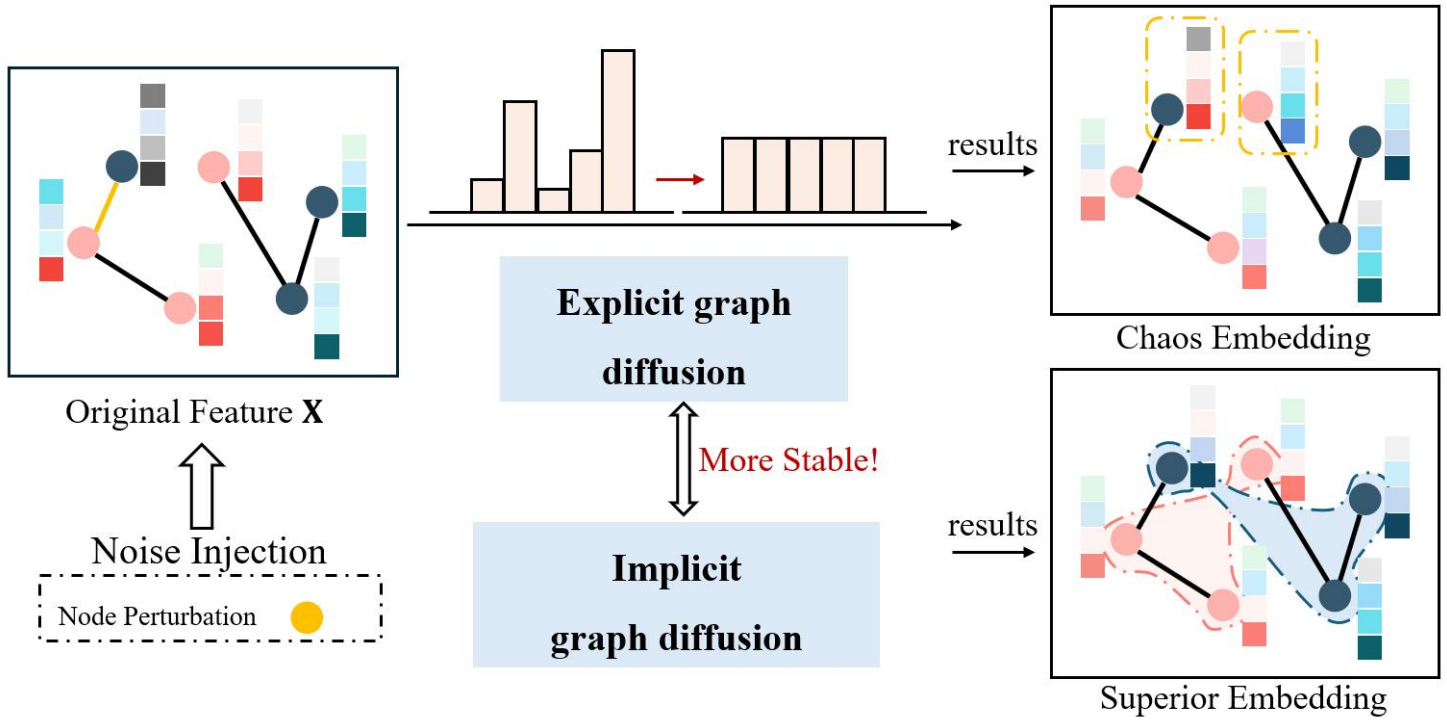
- Marwah, Tanya, et al. Deep equilibrium based neural operators for steady-state PDEs. In: *NIPS*, 2023.

3.2 隐式图神经网络

$$\begin{cases} \frac{\mathrm{d}}{\mathrm{d}x}\mathbf{s}(x) = \mathbf{g}(\mathbf{s}(x), u(x), x) \\ \mathbf{s}(a) = \mathbf{s}_0 \end{cases}$$

$$\begin{cases} \frac{d\mathbf{X}}{dt} = -\mathbf{L} \cdot \mathbf{X}_{(t)} \cdot \mathbf{W}_{(t)}, \ s.t. \ t > t_0 \\ \mathbf{X}_{(t_0)} = \Phi(\mathbf{X}) \end{cases}$$

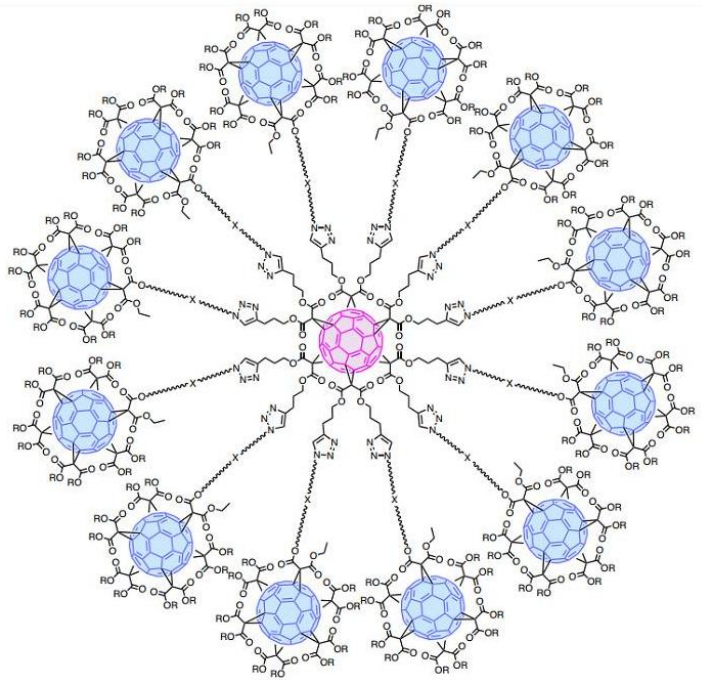
$$\begin{cases} \frac{d\mathbf{X}}{dt} = -\mathbf{L} \cdot \mathbf{X}_{(t)} \cdot \mathcal{P}(\mathbf{W}), \ s.t. \ t > t_0 \\ \mathbf{X}_{(t_0)} = \Phi(\mathbf{X}) \end{cases}$$

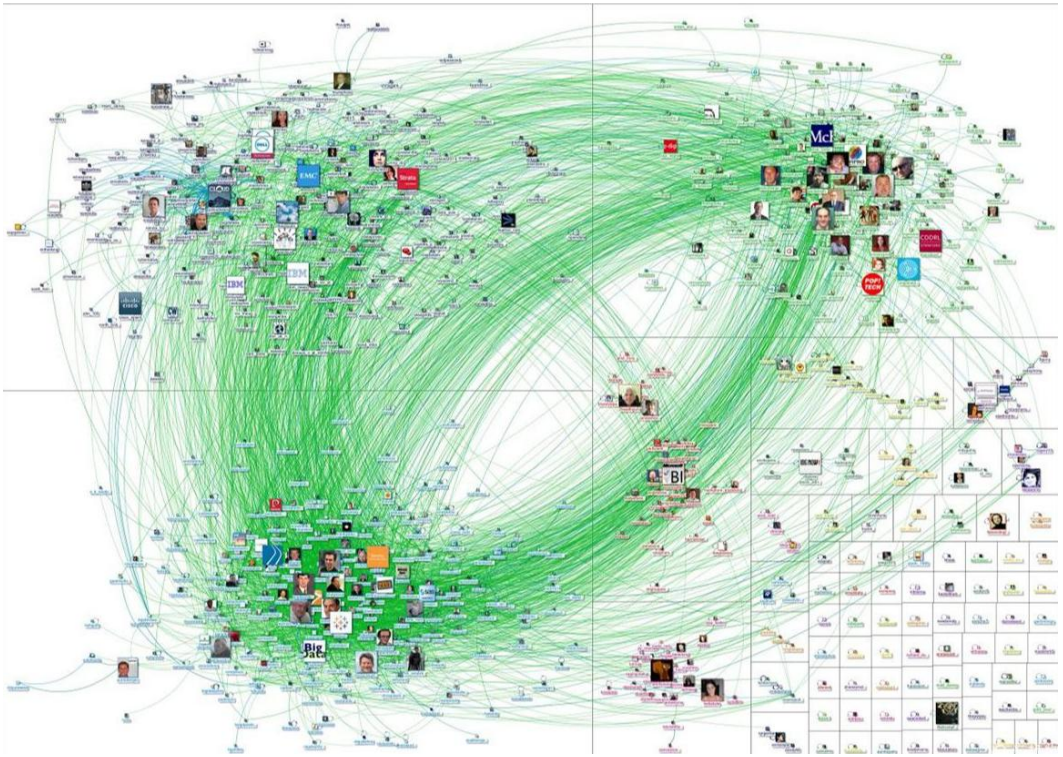


隐式图神经网络更具稳定性！

3.2 隐式图神经网络

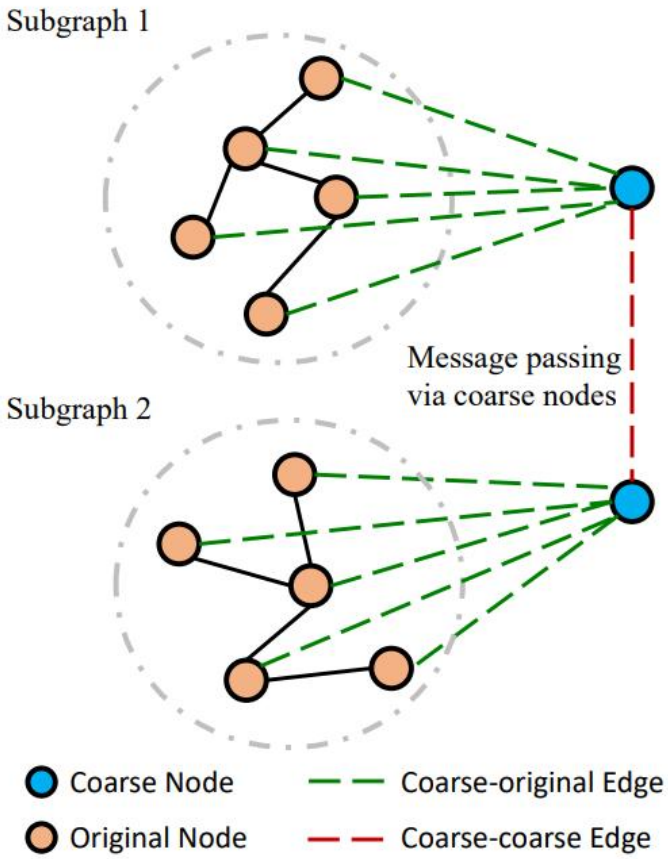
■ **大规模图结构数据如何处理？**





通过将深层前馈图神经网络转为隐式图神经网络大大降低了网络参数量，但依然存在大规模图数据无法处理的问题。医疗和社交网络的图节点的规模可能在百万、千万级。

3.2 隐式图神经网络



Proposition 4. *The proposed stochastic solver is an unbiased estimator of the equilibrium Z^* of the forward pass, i.e., the expectation of the approximated equilibrium $\hat{Z}^*$ is the same as that of the true equilibrium Z^* : $\mathbb{E}\left[\hat{Z}^*\right] = Z^* = \sum_{k=0}^{\infty} \gamma^k g(W)^k f(X, \mathcal{G}) S^k$, under the condition that $\sum_{k=t+1}^{\infty} \gamma^k g(W)^k f(X, \mathcal{G}) S^k \frac{1}{\alpha^{k-t}}$ exists.*

通过粗化处理，使用随机求解器对子图进行训练，得到的表示和使用完整图进行训练能够取得近似的值！

1. 图节点粗化

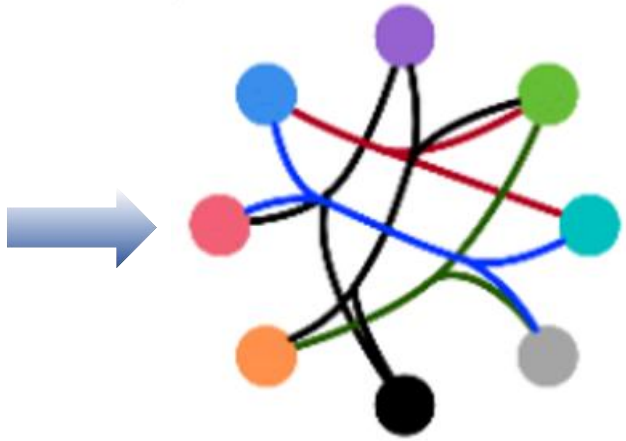
2. 随机求解器

- Liu, Juncheng, et al. "Scalable and Effective Implicit Graph Neural Networks on Large Graphs." in *ICLR*, 2024.

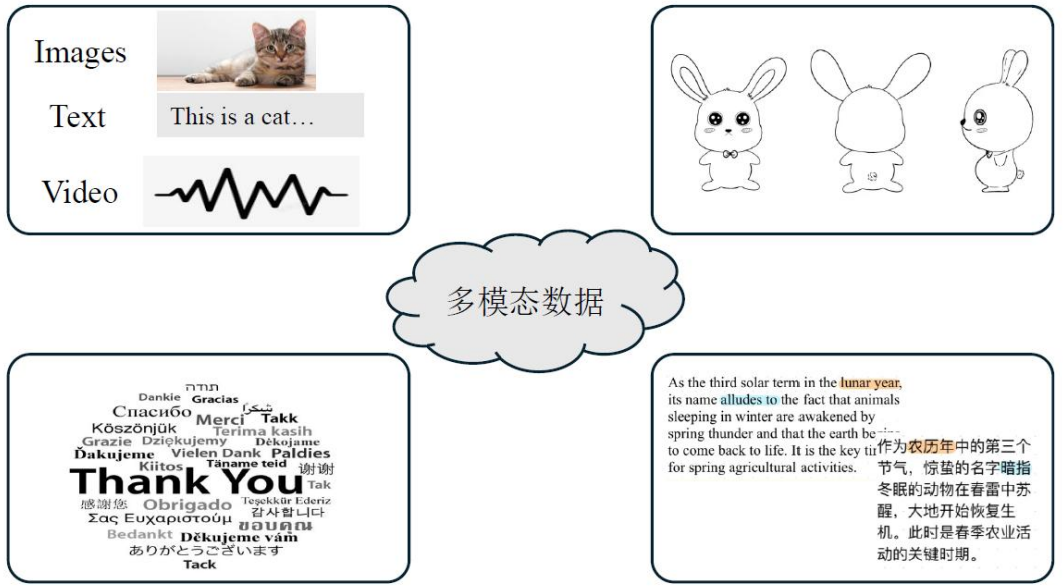
3.2 隐式图神经网络

现有隐式图神经网络的不足：

图结构数据



线性隐式图网络



1. 网络表示能力有一定不足

2. 难以处理多模态数据

如何将构建一个更具表示能力的隐式图神经网络并用于多模态数据处理？

3.2 跨模态非线性的隐式图网络

针对问题1：受谱图理论的启发，我们构建了一个非线性的隐式图网络

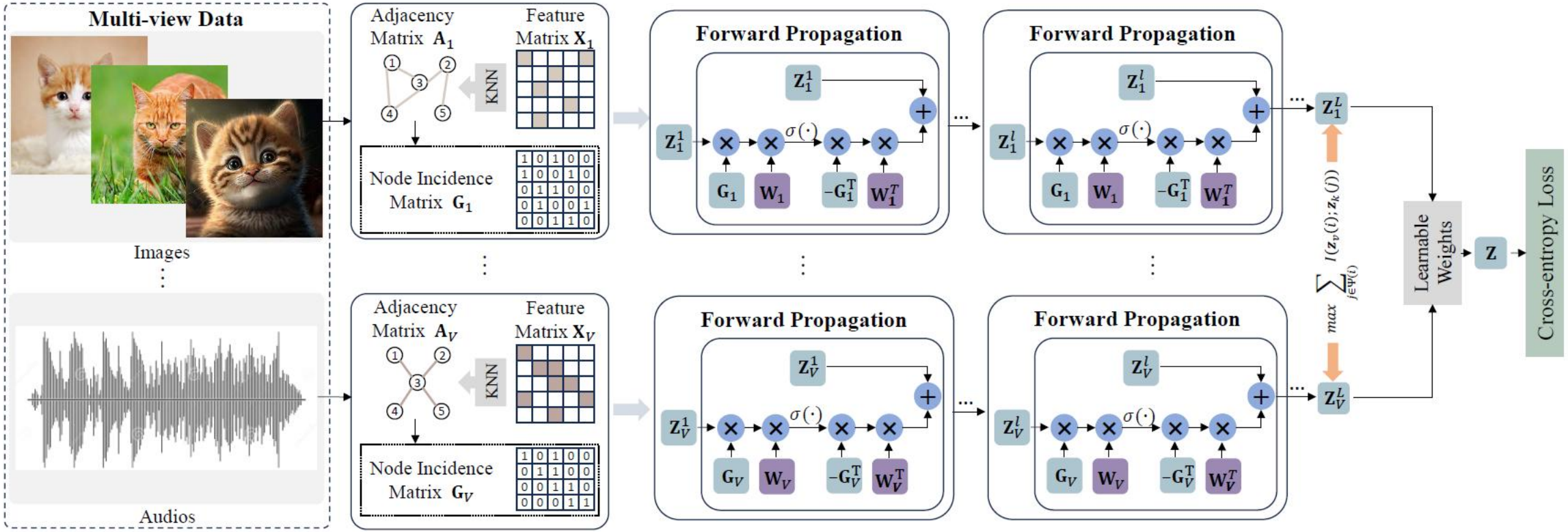
$$\frac{\partial \mathbf{Z}_v^{(t)}}{\partial t} = -\mathbf{G}_v^{\mathsf{T}} \sigma(\mathbf{G}_v \mathbf{Z}_v^{(t-1)} \mathbf{W}_v^{(t)}) \mathbf{W}_v^{(t)^{\mathsf{T}}}$$

针对问题2： 定义了半监督场景下的跨模态一致性，实现了跨模态融合

Definition 1. *(Semi-supervised Cross-view Consistency) Two view-specific (v and k) representations of i-th node $\mathbf{z}_v(i)$ and $\mathbf{z}_k(i)$ are consistent if $min(I(\mathbf{z}_v(i); \mathbf{z}_k(i^{'}))) > max(I(\mathbf{z}_v(i); \mathbf{z}_k(j)))$ for any $i^{'} \in \Psi(i)$ and $j \notin \Psi(i)$, where $\Psi(i)$ denotes the set of nodes with the same label as node i.*

- Pi Y, Wu Y, Huang Y, et al. Inhomogeneous Diffusion-Induced Network for Multiview Semi-Supervised Classification. *IEEE Transactions on Neural Networks and Learning Systems*, 2024.

3.2 隐式图神经网络



用于多模态场景下的隐式图神经网络结构

- Pi Y, Wu Y, Huang Y, et al. Inhomogeneous Diffusion-Induced Network for Multiview Semi-Supervised Classification. *IEEE Transactions on Neural Networks and Learning Systems*, 2024.

PART 4

相关应用

第四部分

目录

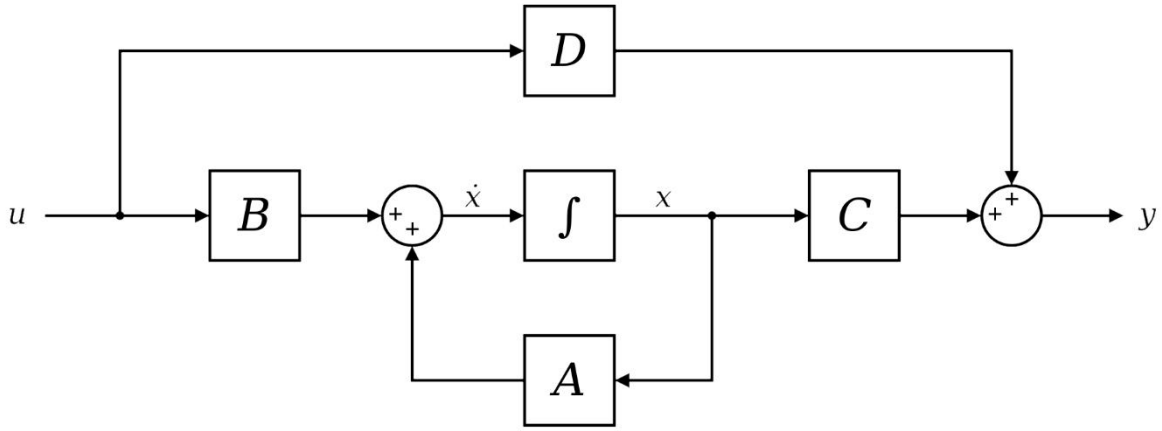
contents

4.1 自然语言处理

状态空间模型(State Space Model, SSM) 是一类数学模型，用于描述一个系统的状态如何随时间演变，适合用于处理动态系统，通过一系列状态变量来表征系统当前的状态以及如何在下一时间步转移。

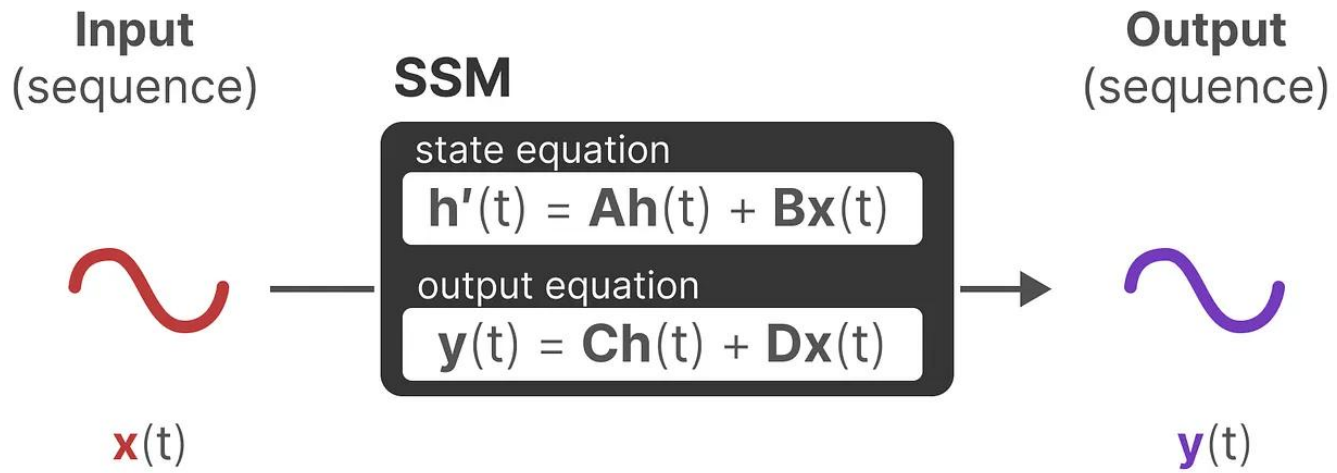
状态空间模型由以下两个方程组成:

- **状态方程:** $x'(t) = Ax(t) + Bu(t)$
- **观测方程:** $y(t) = Cx(t) + Du(t)$



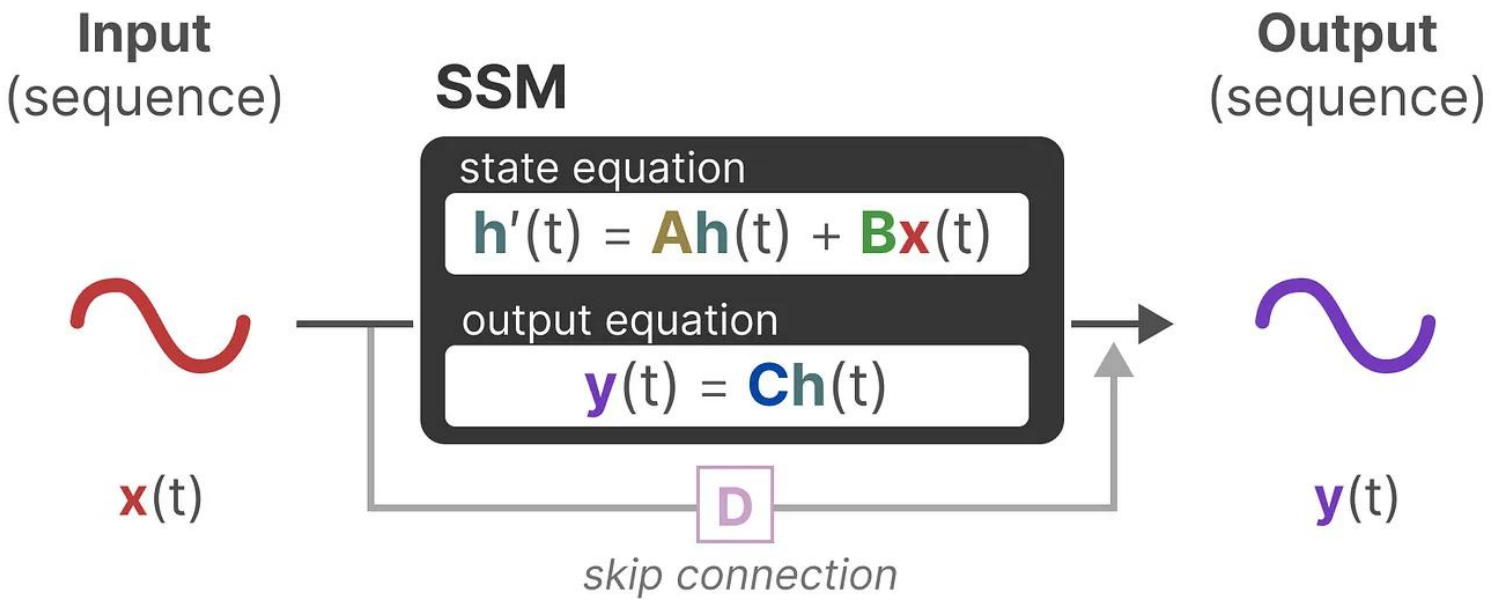
4.1 自然语言处理

SSM不使用离散序列，而是将连续序列作为输入并预测输出序列，A, B, C, D矩阵为可学习的参数，学习后，A, B, C, D固定不变，我们称为线性时不变系统(*Linear time-invariant system, LTI*)。但后续的改进版本mamba中这四个矩阵随着输入不同而变化。



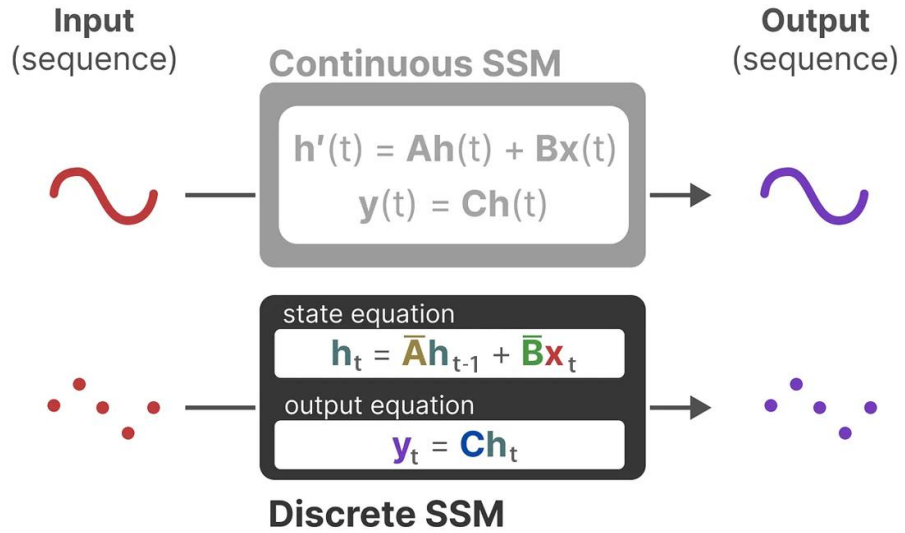
4.1.1 自然语言处理

在深度学习中，D 矩阵类似*skip connection*，因此 SSM 也可以被视作如下形式:



4.1.1 自然语言处理

为了在实际计算中使用，特别是在机器学习任务中，通常需要将连续时间的状态空间模型转化为离散时间的形式。这是因为在数值计算中，计算机无法直接处理连续时间的系统。大部分实际问题中的数据都是离散的（例如，图像、音频、时间序列等），因此需要将连续的数学模型转化为离散形式，以便在计算机上进行处理。

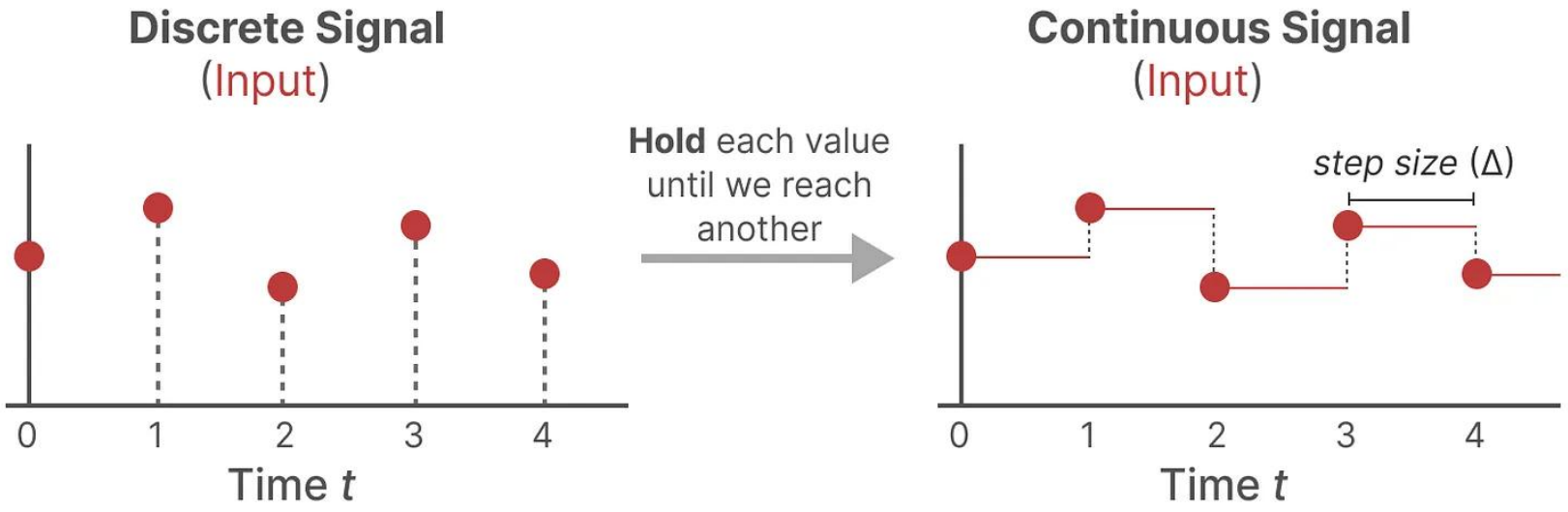


Mamba中采用的的是零阶保持技术(Zero-order hold technique)，下面简单介绍一下ZOH:

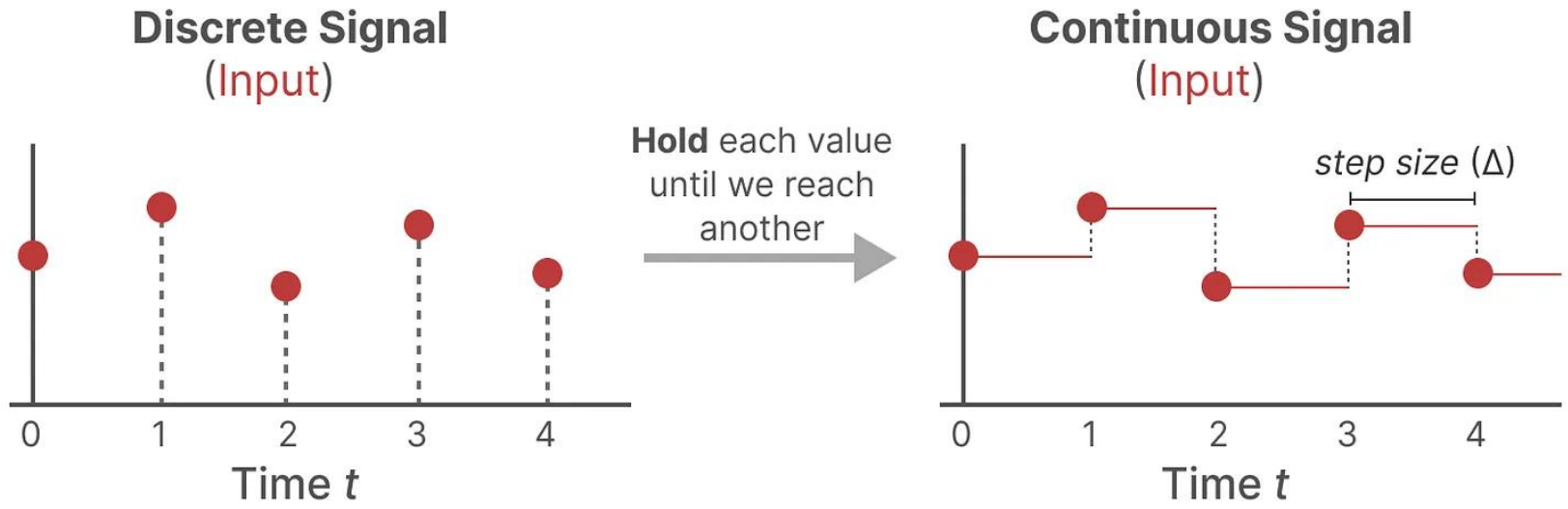
$$\overline{A} = \exp(\Delta A) \qquad \overline{B} = (\Delta \mathrm{A})(\exp(\Delta \mathrm{A}) - \mathrm{I}) \cdot \Delta B$$

4.1 自然语言处理

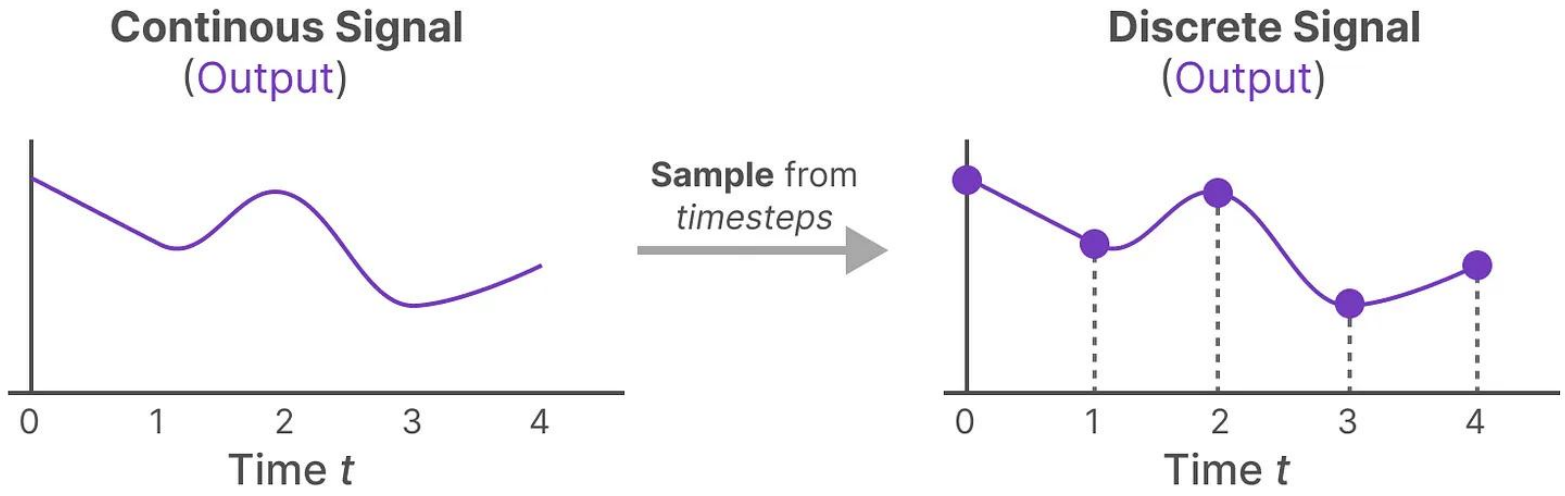
ZOH，全称为零阶保持器（Zero-Order Hold），是一种用于将离散信号转化为连续信号的常用方法。ZOH的原理是保持输入信号在每个采样周期内的值不变，直到下一个采样周期开始。换句话说，在采样之间，信号值保持恒定，从而形成了一系列阶梯状的信号输出。



4.1 自然语言处理



1. 每次收到离散信号时，我们都会保留其值，直到收到新的离散信号，如此操作导致的结果就是创建了 *SSM* 可以使用的连续信号，
2. 保持该值的时间由一个新的可学习参数表示，称为步长(step size)——Δ，它代表输入的阶段性保持(resolution)
3. 有了连续的输入信号后，便可以生成连续的输出，并且仅根据输入的时间步长对值进行采样



这些采样值就是我们的离散输出，针对A，B 按如下方式做零阶保持，能够从连续SSM转变为离散SSM。

$$\overline{A} = \exp(\Delta A) \qquad \overline{B} = (\Delta \mathrm{A})(\exp(\Delta \mathrm{A}) - \mathrm{I}) \cdot \Delta B$$

$\overline{A}$和$\overline{B}$为离散后的模型参数，离散SSM可由如下方程表示：

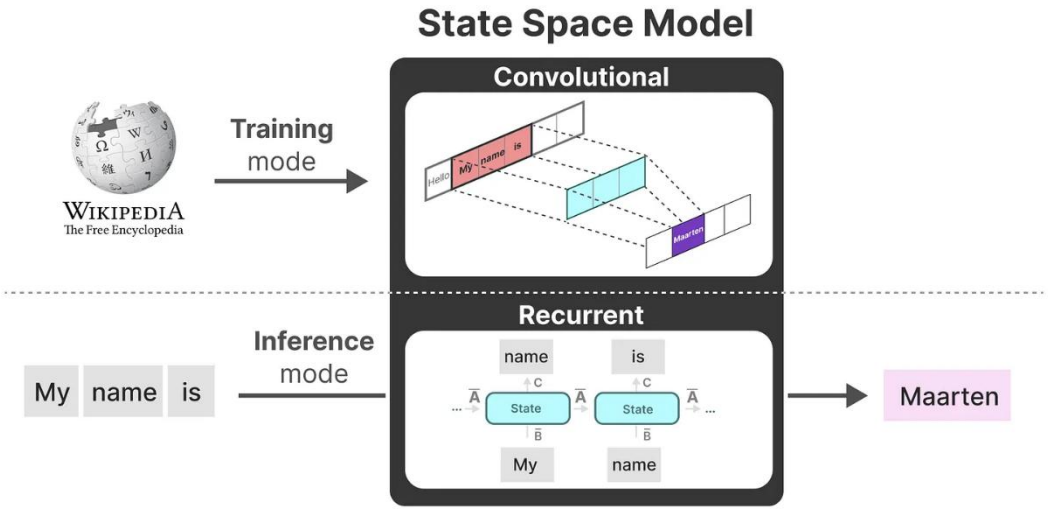
$$h_t = \overline{A} h_{t-1} + \overline{B} x_t$$

$$y_t = Cht$$

4.1 自然语言处理

离散化后的SSM公式可以表示为两种形式，递归和卷积。如下所示，公式(2a)是离散状态空间方程的递归形式，通过不断递推，我们可以得到卷积形式，如公式(3a)所示。

$$h'(t) = Ah(t) + Bx(t) \quad (1a)$$
$$y(t) = Ch(t) \quad (1b)$$
$$h_t = \overline{A}h_{t-1} + \overline{B}x_t \quad (2a)$$
$$y_t = Ch_t \quad (2b)$$
$$\overline{K} = (C\overline{B}, C\overline{AB}, \dots, C\overline{A}^k\overline{B}, \dots) \quad (3a)$$
$$y = x * \overline{K} \quad (3b)$$



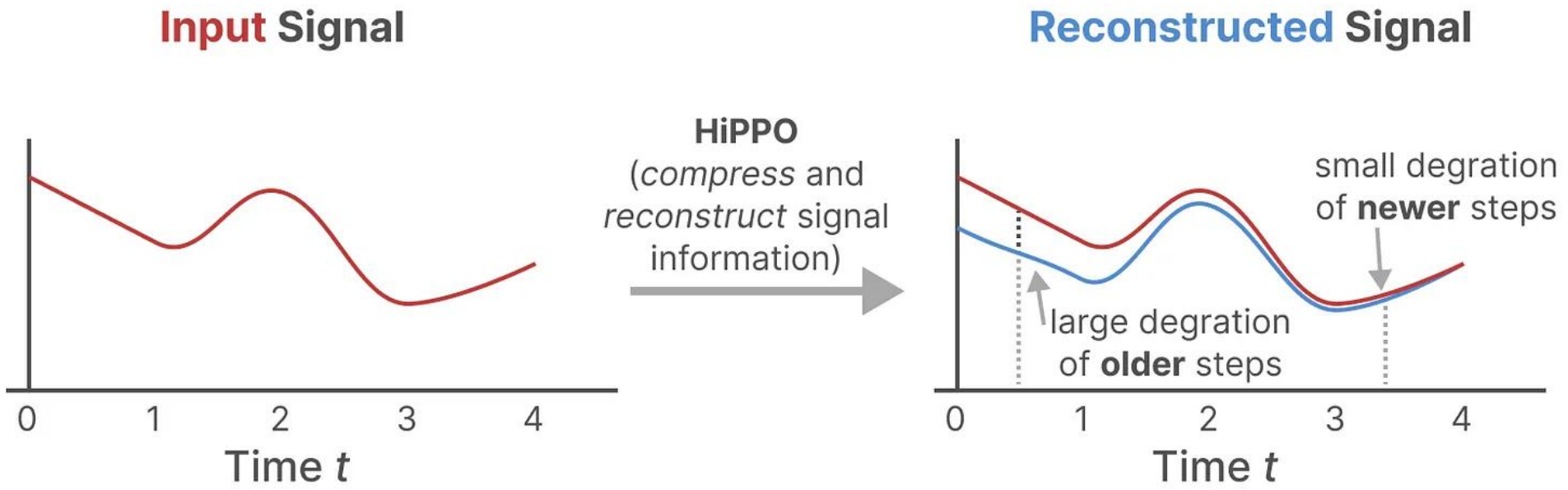
- 卷积结构表示：方便并行训练
- 递归结构表示：方便快速推理

总之，这类模型可以非常高效地计算，可以通过递归或卷积实现，且在序列长度上具有线性或接近线性的扩展性。

- Gu, A. and Dao, T. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.

4.1 自然语言处理

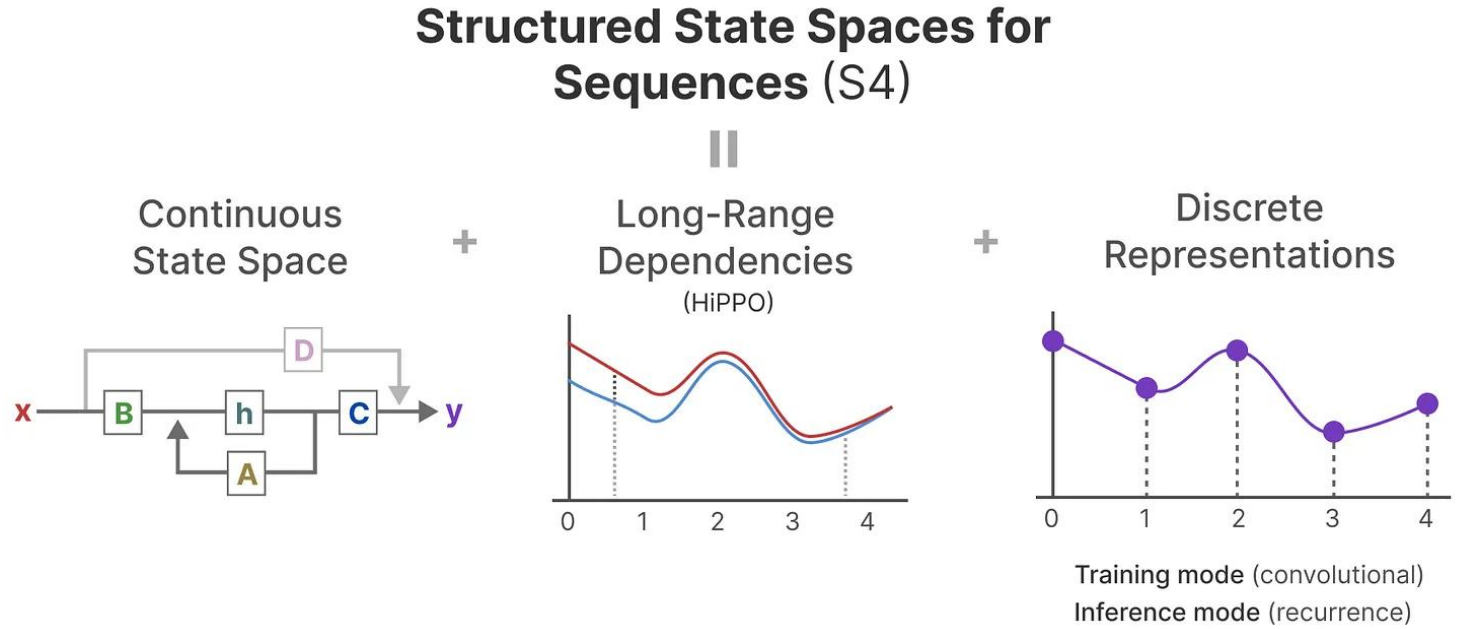
在自然语言处理任务中，捕捉长距离依赖关系是至关重要的，但由于序列的长度增长会导致存储和计算成本急剧增加。HIPPO矩阵是一种高阶多项式投影操作符，它能够在有限的存储空间内高效地解决序列建模中的长程依赖问题。



- Albert Gu, Tri Dao, Stefano Ermon, Atri Rudra, and Christopher Ré. “HIPPO: Recurrent Memory with Optimal Polynomial Projections”. In: *NIPS*, 2020.

4.1 自然语言处理

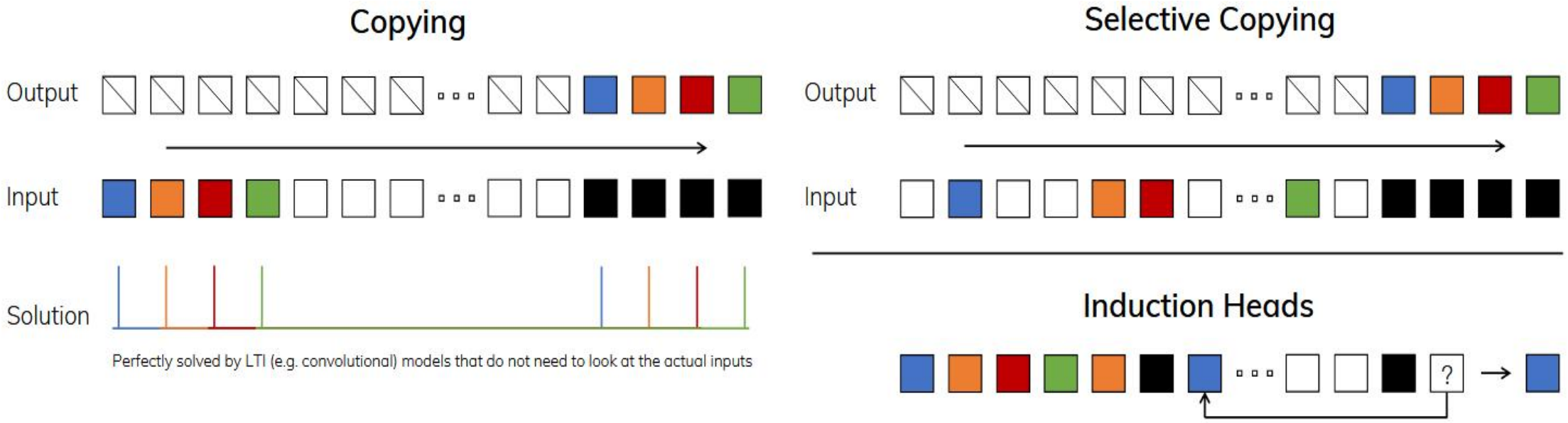
结构化状态空间模型(Structured State Spaces for Sequences ，S4) 优点在于能捕捉长程依赖(通过HIPPO)，并且可以通过卷积并行训练。但是S4依赖于线性、时不变(LTI)的系统假设，而自然语言的复杂结构和非线性依赖超出时不变系统的能力，无法有效处理复杂语境下的某些任务。比如选择性复制任务和归纳头任务。



- Albert Gu, Karan Goel, and Christopher Ré. Efficiently Modeling Long Sequences with Structured State Spaces. In: *ICLR*, 2022.

4.1 自然语言处理

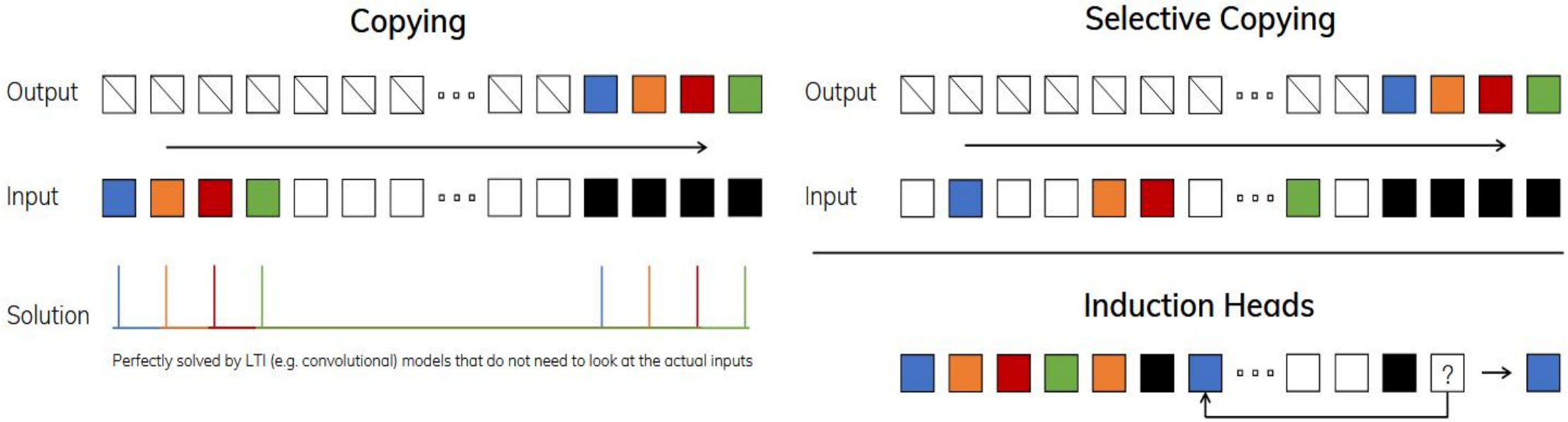
- **选择性复制任务**相对于流行的复制任务，改变了需要记忆的标记的位置。该任务要求具备内容感知推理能力，以能够记住相关的标记（有颜色的）并过滤掉无关的标记（白色的）。
- **归纳头任务**是一种被广泛假设用来解释大型语言模型（LLMs）大部分上下文学习能力的机制，该任务要求上下文感知推理能力，以在适当的上下文中知道何时生成正确的输出（黑色的）。



- Martin Arjovsky, Amar Shah, and Yoshua Bengio. Unitary Evolution Recurrent Neural Networks. In: *ICML*, 2016.

4.1 自然语言处理

这些任务揭示了线性时不变（LTI）模型的失效模式。从递归的角度来看，LTI 模型的恒定动态（公式（2）中的（$\overline{A}$，$\overline{B}$）转换）无法让它们从上下文中选择正确的信息，或者以依赖输入的方式影响沿着序列传递的隐藏状态。从卷积的角度来看，已知全局卷积可以解决基本的复制任务，因为它只需要时间感知能力，但在选择性复制任务中则存在困难，原因在于它们缺乏内容感知能力。更具体地说，输入到输出之间的间隔是变化的，无法通过静态卷积核进行建模。



- David W Romero, Anna Kuzina, Erik J Bekkers, Jakub M Tomczak, and Mark Hoogendoorn. CKConv: Continuous Kernel Convolution For Sequential Data. *arXiv preprint arXiv:2102.02611*, 2021.

4.1 自然语言处理

构建序列模型的一个基本原则是选择性，或者说上下文感知的关注或过滤输入到序列状态的能力。

将选择机制纳入模型的一种方法是让影响序列交互的参数依赖于输入。

Algorithm 1 SSM (S4)

Input: $x : (\mathrm{B}, \mathrm{L}, \mathrm{D})$
Output: $y : (\mathrm{B}, \mathrm{L}, \mathrm{D})$
1: $\boldsymbol{A} : (\mathrm{D}, \mathrm{N}) \leftarrow$ Parameter
▷ Represents structured $N \times N$ matrix
2: $\boldsymbol{B} : (\mathrm{D}, \mathrm{N}) \leftarrow$ Parameter
3: $\boldsymbol{C} : (\mathrm{D}, \mathrm{N}) \leftarrow$ Parameter
4: $\Delta : (\mathrm{D}) \leftarrow \tau_\Delta(\text{Parameter})$
5: $\overline{\boldsymbol{A}}, \overline{\boldsymbol{B}} : (\mathrm{D}, \mathrm{N}) \leftarrow \text{discretize}(\Delta, \boldsymbol{A}, \boldsymbol{B})$
6: $y \leftarrow \text{SSM}(\overline{\boldsymbol{A}}, \overline{\boldsymbol{B}}, \boldsymbol{C})(x)$
▷ Time-invariant: recurrence or convolution
7: **return** y

Algorithm 2 SSM + Selection (S6)

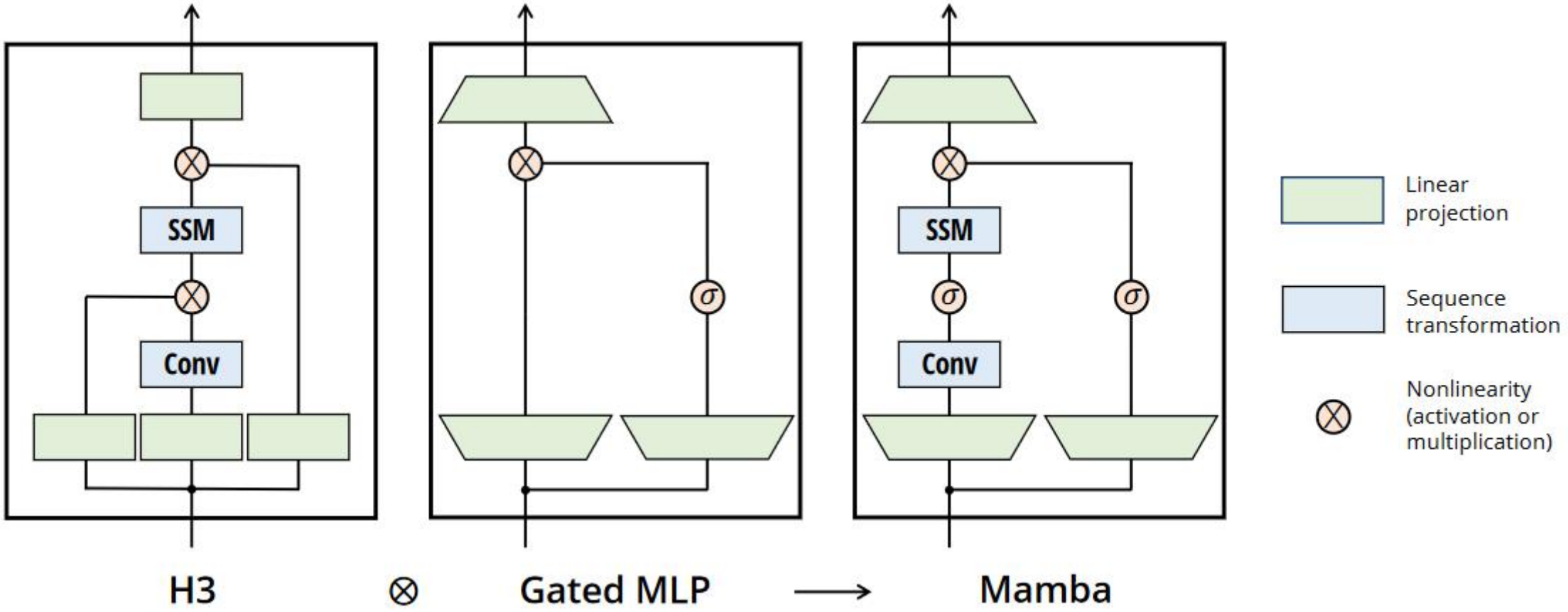
Input: $x : (\mathrm{B}, \mathrm{L}, \mathrm{D})$
Output: $y : (\mathrm{B}, \mathrm{L}, \mathrm{D})$
1: $\boldsymbol{A} : (\mathrm{D}, \mathrm{N}) \leftarrow$ Parameter
▷ Represents structured $N \times N$ matrix
2: $\boldsymbol{B} : (\mathrm{B}, \mathrm{L}, \mathrm{N}) \leftarrow s_B(x)$
3: $\boldsymbol{C} : (\mathrm{B}, \mathrm{L}, \mathrm{N}) \leftarrow s_C(x)$
4: $\Delta : (\mathrm{B}, \mathrm{L}, \mathrm{D}) \leftarrow \tau_\Delta(\text{Parameter} + s_\Delta(x))$
5: $\overline{\boldsymbol{A}}, \overline{\boldsymbol{B}} : (\mathrm{B}, \mathrm{L}, \mathrm{D}, \mathrm{N}) \leftarrow \text{discretize}(\Delta, \boldsymbol{A}, \boldsymbol{B})$
6: $y \leftarrow \text{SSM}(\overline{\boldsymbol{A}}, \overline{\boldsymbol{B}}, \boldsymbol{C})(x)$
▷ Time-varying: recurrence (*scan*) only
7: **return** y

$s_B(x) = \mathrm{Linear}_N(x)$、$s_C(x) = \mathrm{Linear}_N(x)$、$s_\Delta(x) = \mathrm{Broadcast}_D(\mathrm{Linear}_1(x))$，且$\tau_\Delta = \mathrm{softplus}$，其中$\mathrm{Linear}_d$ 是维度 d 的参数化投影。

通过以上算法对比可以发现，这些参数现在有一个长度维度L，这意味着模型已经从时不变变为时变。

4.1 自然语言处理

Mamba架构如下。



4.1 自然语言处理

Model	Arch.	Layer	Acc.
S4	No gate	S4	18.3
-	No gate	S6	**97.0**
H3	H3	S4	57.0
Hyena	H3	Hyena	30.1
-	H3	S6	**99.7**
-	Mamba	S4	56.4
-	Mamba	Hyena	28.4
Mamba	Mamba	S6	**99.8**

Table 1: (**Selective Copying.**) Accuracy for combinations of architectures and inner sequence layers.

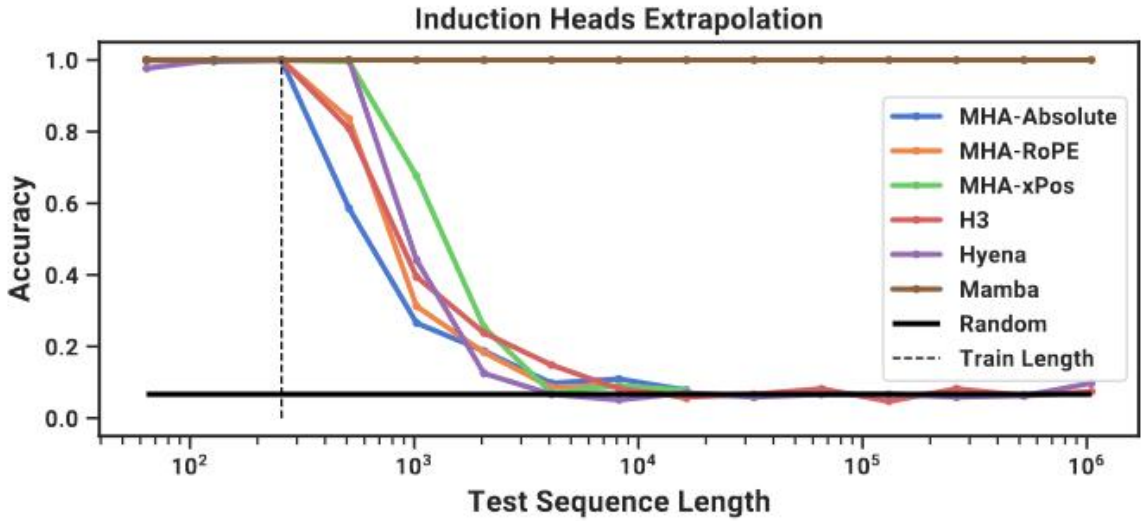
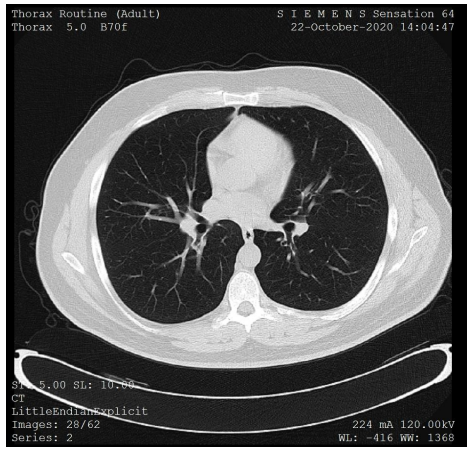
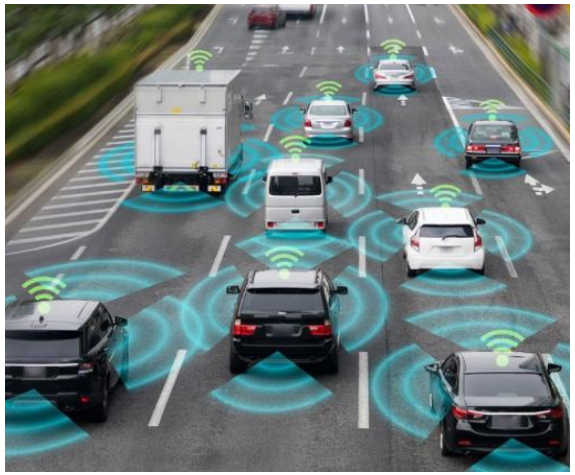


Table 2: (**Induction Heads.**) Models are trained on sequence length $2^8 = 256$, and tested on increasing sequence lengths of $2^6 = 64$ up to $2^{20} = 1048576$. Full numbers in Table 11.

4.2 计算机视觉

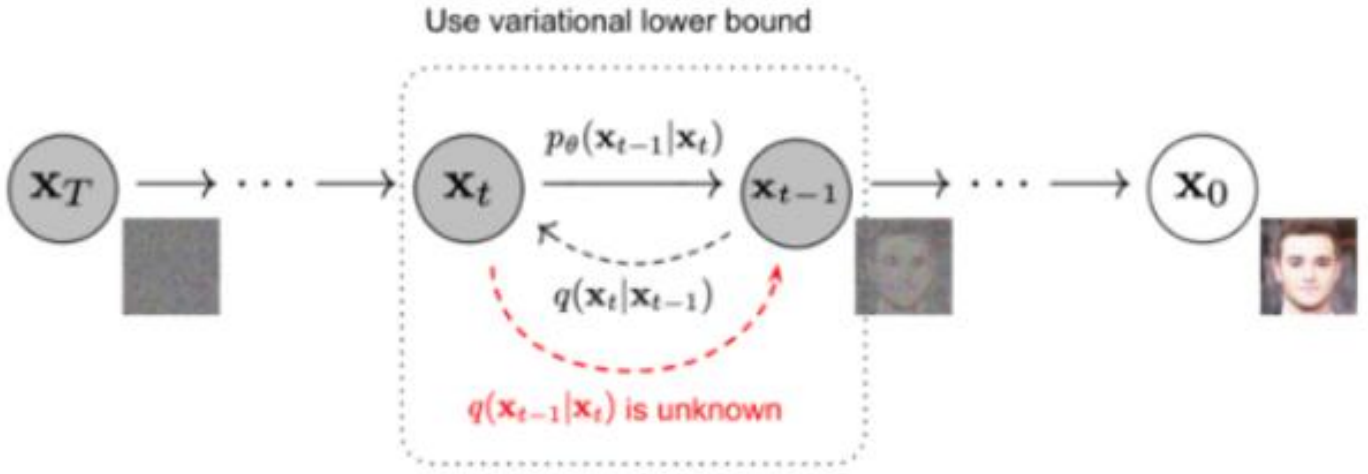
- 计算机视觉是人工智能和计算机科学的一个分支，旨在让使计算机能够理解和解释视觉信息（如图像和视频）的领域。其目标是模拟人类的视觉系统，以自动化分析和理解视觉数据。
- 计算机视觉应用领域有：医疗影像分析，自动驾驶，人脸识别，图像和视频监控





4.2 扩散模型

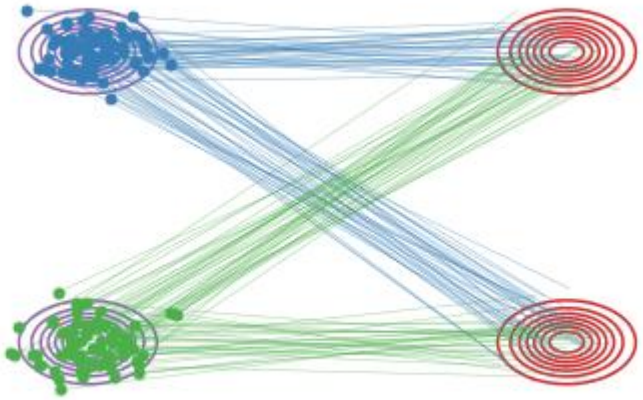
- 扩散模型是一种生成模型，通过逐步添加噪声到数据中，再通过逆向过程去噪以生成新样本。
- 扩散模型的基本过程分为两个阶段分别是：前向过程与反向过程
- 优势：高质量生成，多样性，训练稳定性



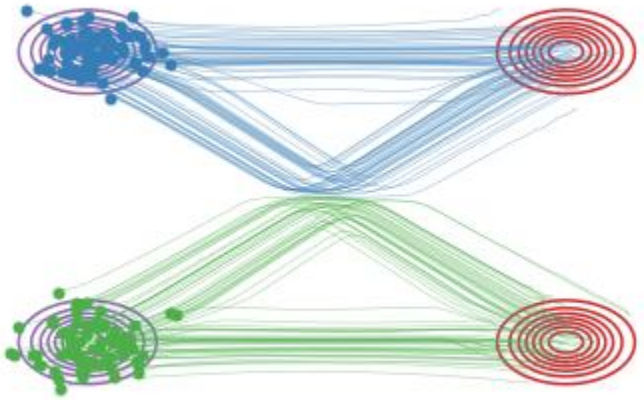
- Croitoru, F. A., Hondru, V., Ionescu, R. T., & Shah, M. Diffusion models in vision: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.

4.2 扩散模型

- 扩散模型的基本过程分为两个阶段分别是：前向过程与反向过程
- 优势：高质量生成，多样性，训练稳定性



Linear interpolation



Rectified Flow

- Croitoru, F. A., Hondru, V., Ionescu, R. T., & Shah, M. Diffusion models in vision: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.

4.2 扩散过程的具体说明

前向扩散过程： 前向过程是将真实数据逐步添加噪声，直到变为纯噪声

使用以下公式可以直接获取特定时间步长t下的噪声分布

每一步都依赖于前一步的输出，使得模型能够在每一步中控制噪声的添加量。

$$x_t = \sqrt{\overline{\alpha_t}}x_0 + \sqrt{1-\overline{\alpha_t}}\epsilon$$

4.2 扩散过程的具体说明

反向扩散过程：从纯噪声开始，逐步去噪以恢复原始数据的分布。 由于该结果不可直接计算，主要通过训练神经网络ϵ来逼近它。通过不断迭代，最终将噪声样本转化为清晰的图像。训练目标损失函数：

$$x_t = \sqrt{\overline{\alpha}_t} x_0 + \sqrt{1 - \overline{\alpha}_t}\epsilon$$

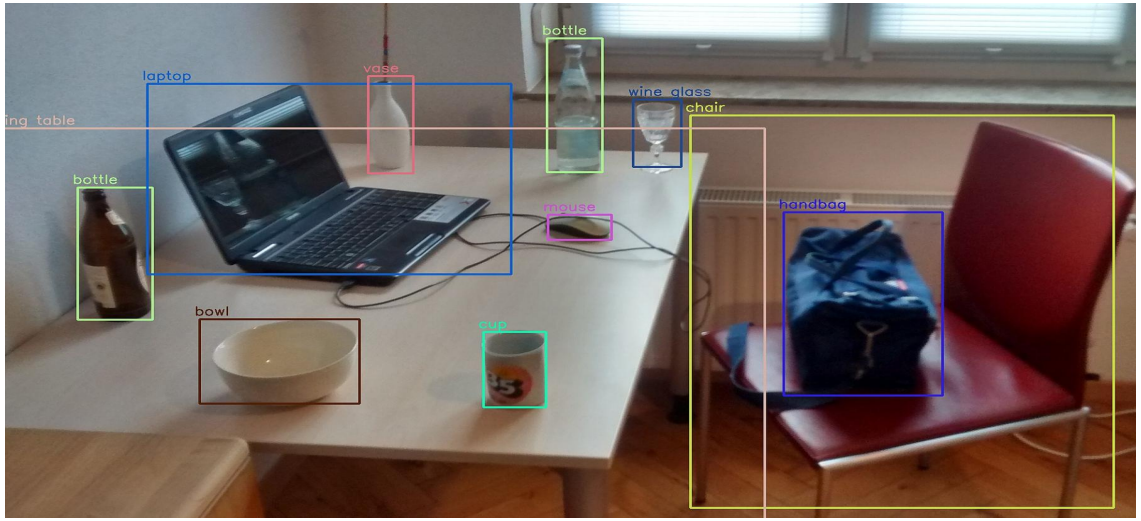
$$L_{simple} = E_{t,x_0,\varepsilon}[||\varepsilon - \varepsilon_\theta(x_{t,t})||^2]$$

4.2 扩散模型在目标检测中的应用

目标检测是计算机视觉的一项关键任务，旨在识别图像中的目标对象并定位其位置。该任务面临挑战：检测精度、速度以及对小物体和遮挡物体的识别能力。针对这个问题扩散模型可以实现以下功能：

数据增强：使用扩散模型生成合成图像，增加数据集的多样性。

目标定位：利用扩散模型可以通过去噪过程更清晰地突出目标，使得后续的定位算法（如边界框回归）能够更准确地识别目标。



- Luo, R., Song, Z., Ma, L., Wei, J., Yang, W., Yang, M. Diffusiontrack: Diffusion model for multi-object tracking. In: *AAAI*, 2024.

4.2 扩散模型在自动驾驶领域的应用

自动驾驶的目标是：获得一个可以应对复杂条件下的自动驾驶模型。

针对这个问题扩散模型可以实现以下功能：

数据增强：生成多样化的合成图像，模拟不同天气、时间和环境条件下的驾驶场景，从而增强自动驾驶模型的训练数据。

图像去噪：针对传感器数据（如相机图像）常常受到噪声影响，扩散模型可以用于去噪。

- Harvey, W., Naderiparizi, S., Masrani, V., Weilbach, C., Wood, F. Flexible diffusion modeling of long videos. In：*NIPS*, 2022.

4.2 扩散模型在图像生成中的应用

任务目标： 利用算法生成的图像应具有高质量、真实感和多样性。

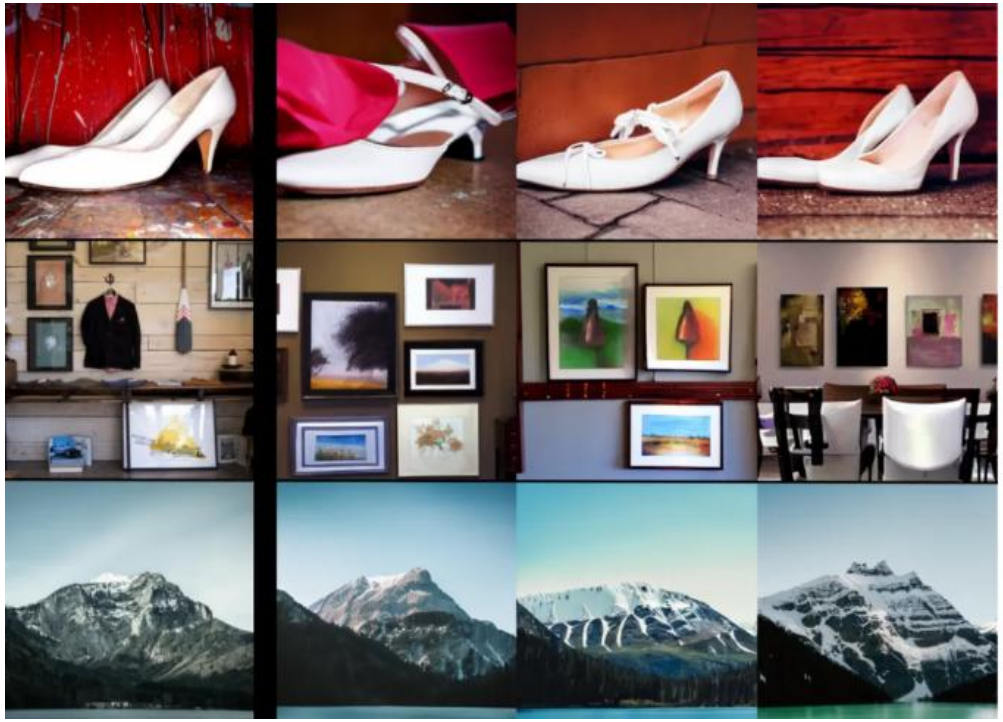
流程：噪声添加过程：将真实图像逐步添加噪声，直到图像变得不可识别，形成一个高维噪声分布。

去噪过程：从随机噪声开始，通过训练的网络逐步去噪，生成与训练数据相似的新图像。

- Ho, Jonathan; et al. “Cascaded diffusion models for high fidelity image generation”. *The Journal of Machine Learning Research*, 2022.

4.2 扩散模型在图像到图像的转换中的应用

同样的方式也适用于图像到图像的合成，但是图像的转换一半需要输入样本图像作为参考图像。生成的图像在语义和视觉上与作为参考的图像相似。这个过程在概念上类似于基于风格的生成式对抗网络模型，然而，它在保留图像的语义结构方面做得更好。



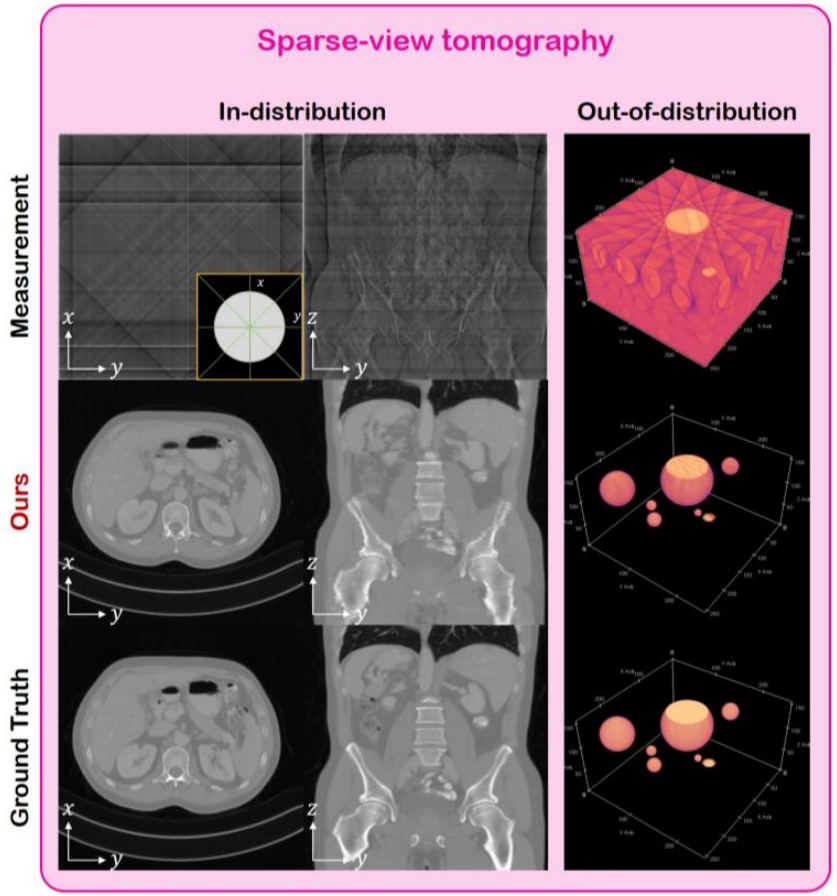
- M Xia, Y Zhou, R Yi, et al. A Diffusion Model Translator for Efficient Image-to-Image Translation, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

4.2 扩散模型在2D模型到3D模型生成中的应用

任务目标：利用简单的2D图生成的3D图像。

流程：

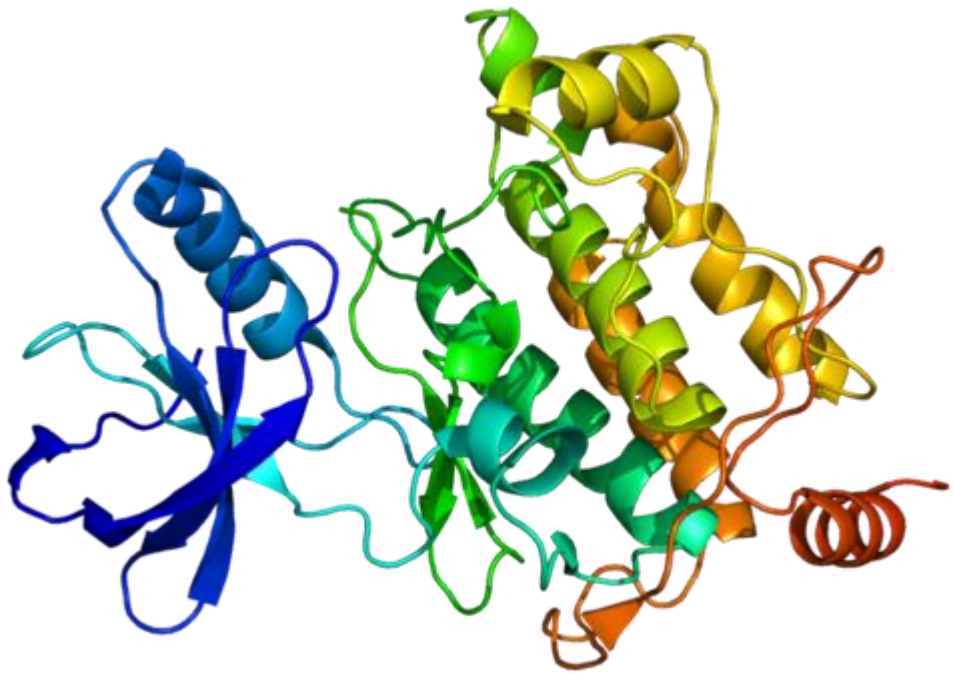
- 通过输入2D图像及其对应3D模型，对网络架构进行训练。
- 输入2D图像，从随机噪声开始，通过训练后的网络逐步去噪，从而生成3D模型。
- 对生成的3D模型进行后期处理。



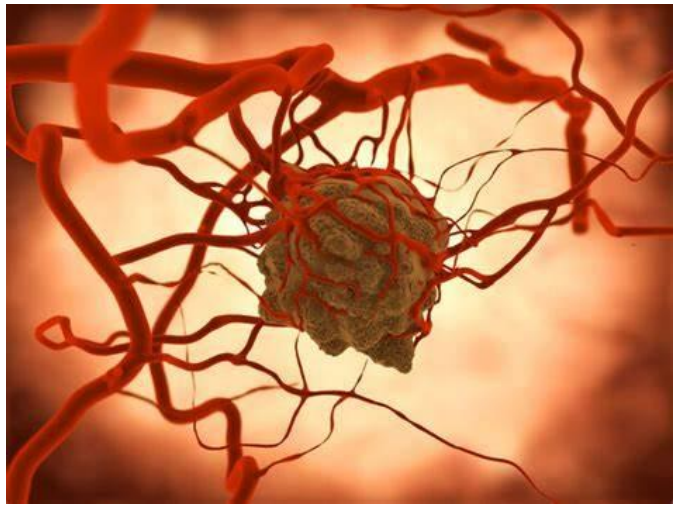
- Karnewar, A., Vedaldi, A., Novotny, D., Mitra, N. J. Holodiffusion: Training a 3d diffusion model using 2d images. In: *CVPR*, 2023.

4.3 蛋白质分子检测

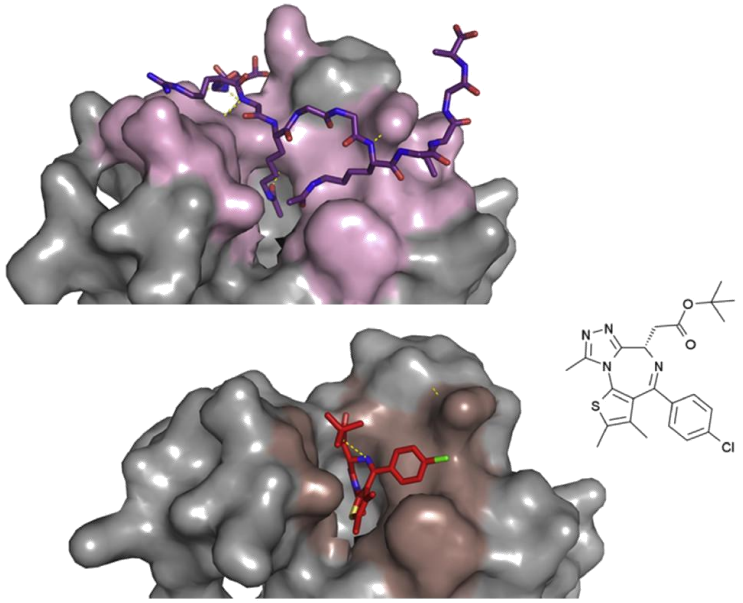
- 通过精确**检测和分析**蛋白质分子，不仅可以深入理解生物过程中的关键机制，还能帮助识别疾病的生物标志物，推动**早期诊断和个性化医疗**的发展。
- 在药物研发中，蛋白质分子检测技术用于**筛选药物靶点、监控药物与蛋白质的相互作用**，从而加速新药的开发过程，提升药物的有效性和安全性。这些技术为推动精准医疗和生物技术创新提供了坚实的基础。



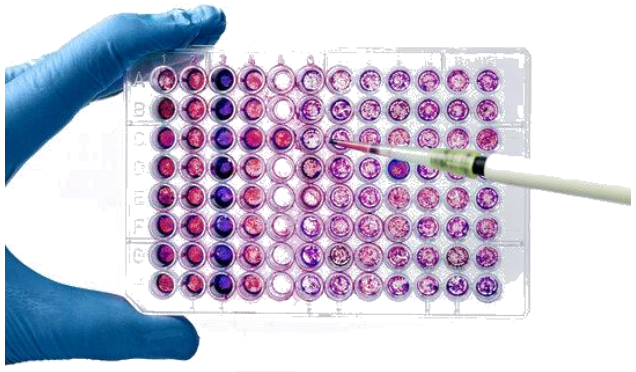
4.3 蛋白质分子检测背景



疾病诊断与监测



药物靶点发现与药物研发



精准医疗

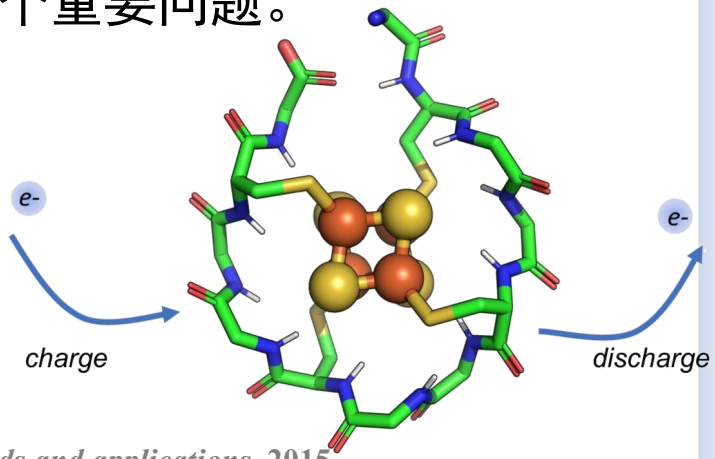
蛋白质分子检测在生物医学和药物研发等领域具有极高的应用价值。

4.3 蛋白质分子检测任务–蛋白质定量分析

任务目标：蛋白质定量分析旨在测量样品中某种或某些特定蛋白质的含量。

面临挑战：

- **噪声干扰：**样品中存在的其他分子或残留物质可能会干扰定量分析的准确性。
- **灵敏度限制：**对于低丰度的蛋白质，常规检测方法可能难以检测，需结合高灵敏度技术。
- **高通量分析：**面对复杂的生物样品，如何提高检测速度和通量是一个重要问题。



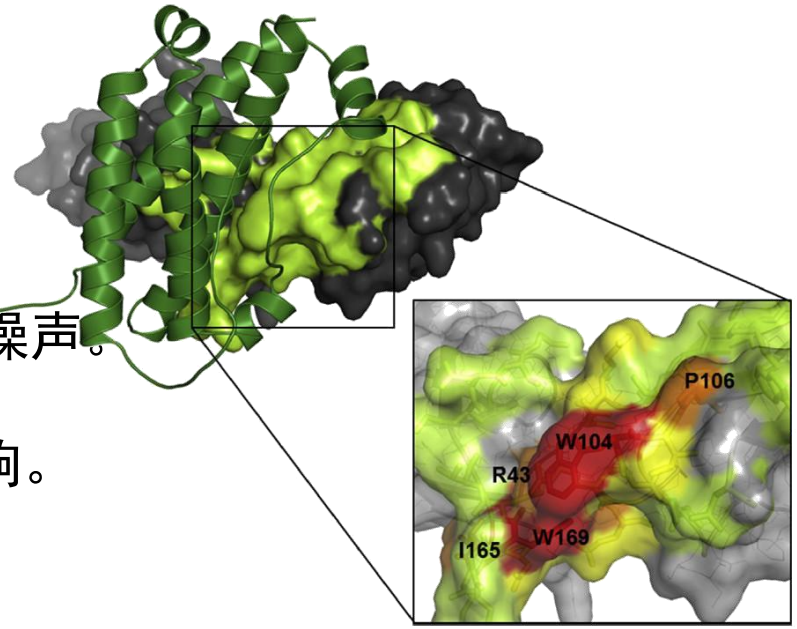
- Pagala V R, High A A, Wang X, et al. Quantitative protein analysis by mass spectrometry. *Protein-protein interactions: Methods and applications*, 2015.

4.3 蛋白质分子检测任务–蛋白质相互作用研究

任务目标： 蛋白质相互作用在细胞生物学中至关重要。蛋白质并不是孤立工作的，它们通过与其他分子（如其他蛋白质、核酸、小分子）相互作用来调控细胞功能。

面临挑战：

- **特异性和灵敏度：**研究蛋白质相互作用的特异性需要确保检测信号来源于真实的相互作用，而不是实验引入的噪声。
- **动态性：**蛋白质相互作用是动态的，受细胞环境或时间的影响。



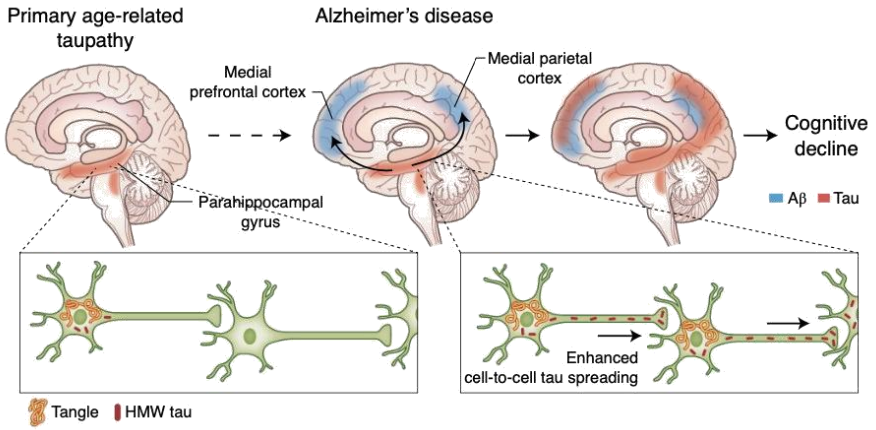
- Katz C, Levy-Beladev L, Rotem-Bamberger S, et al. Studying protein–protein interactions using peptide arrays. *Chemical Society Reviews*, 2011.

4.3 蛋白质分子检测任务–蛋白质结构与功能的检测

任务目标： 蛋白质的结构决定其功能，解析蛋白质的三维结构是理解其生物功能的基础。许多疾病的发生都与蛋白质的结构异常有关（如阿尔茨海默病中的蛋白质错误折叠）。

面临挑战：

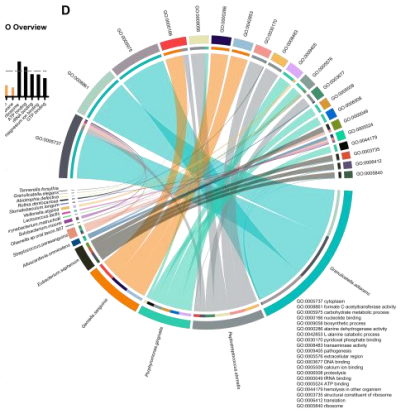
- **复杂性和多样性：**蛋白质具有动态性和多构象，许多蛋白质在不同环境下可能表现出不同的结构，给检测带来挑战。
- **计算资源需求：**解析大型蛋白质结构需要大量计算资源，尤其是使用冷冻电镜或NMR技术。



- Lisewski A M, Lichtarge O. Rapid detection of similarity in protein structure and function through contact metric distances. *Nucleic Acids Research*, 2006.

4.3 蛋白质分子检测任务-蛋白质组学分析

任务目标： 蛋白质组学是一种高通量的分析手段，旨在全面分析样品中所有蛋白质的种类、丰度、修饰状态及其动态变化。蛋白质组学分析在系统生物学和药物开发中具有重要价值，有助于揭示生物系统的整体功能，并筛选出潜在的疾病标志物。



面临挑战：

- **数据复杂性**：蛋白质组学产生大量复杂的数据，如何进行有效分析和解释是一个挑战。
- **技术敏感性**：蛋白质组学方法需要高度敏感的仪器，以确保低丰度蛋白质的可靠检测。

- Gilchrist A, Au C E, Hiding J, et al. Quantitative proteomics analysis of the secretory pathway. *Cell*, 2006.

4.3 扩散过程与蛋白质分子传输

蛋白质在**许多检测**方法中（如电泳和色谱）都会经历扩散过程，
微分方程帮助建模和预测这种扩散行为。

$$\frac{\partial D}{\partial t} = D\nabla^2 C$$

C:蛋白质浓度　∇：扩散系数 $\nabla^2 C$：浓度的空间梯度（拉普拉斯算子）

在 **电泳** 技术中，蛋白质在电场和扩散的双重作用下迁移。通过扩散方程，
可以模拟蛋白质迁移的扩散过程，优化分离条件，以提高分辨率和检测精度。

4.3 蛋白质折叠与Fokker-Planck方程的关联

蛋白质的三维结构直接决定其功能，而蛋白质折叠是从无序的线性链转变为功能性三维结构的动态过程。

$$\frac{\partial P(x,t)}{\partial t} = -\nabla \cdot \left(P(x,t)v(x)\right) + D\,\nabla^2 P(x,t)$$

$P(x,t)$: 蛋白质在构象空间中的概率分布
$v(x)$: 蛋白质折叠过程中的驱动力
D: 扩散系数

蛋白质折叠动力学：通过Fokker-Planck方程，研究人员可以建模蛋白质的折叠路径，预测蛋白质如何从无序状态折叠成其稳定的三维结构。这对于理解蛋白质功能具有重要意义，特别是在折叠错误与疾病（如阿尔茨海默病）相关的研究中。

- Piana S, Lindorff-Larsen K, Shaw D E. Protein folding kinetics and thermodynamics from atomistic simulation. *Proceedings of the National Academy of Sciences*, 2012.

4.3 蛋白质热稳定性与热传导方程的关联

蛋白质的功能往往受温度的影响，特别是蛋白质在高温下可能发生 **热变性**，导致其功能丧失。研究蛋白质的热稳定性时，热量的传导和扩散是关键因素。

$$\frac{\partial T}{\partial t} = \alpha \nabla^2 T$$

T: 温度　　α: 热扩散系数

差示扫描量热法（DSC）：通过热传导方程，研究人员可以分析蛋白质在温度变化下的行为，如熔解温度、热稳定性和变性过程。这些信息对蛋白质的应用（如药物稳定性）和功能研究至关重要。

- Menczel J D, Judovits L, Prime R B, et al. Differential scanning calorimetry. *Thermal analysis of polymers: Fundamentals and applications*, 2009.

4.3 微分方程在高通量分析中的应用

1. **动力学分析：**
    1. 使用 **Michaelis-Menten 动力学方程** 描述高通量药物筛选中分子间的结合动力学，为优化实验条件提供支持。
2. **扩散模型：**
    1. 在微流体芯片中的高通量分析过程中，**扩散方程** 用于模拟蛋白质或药物分子的迁移行为，优化检测效率。
3. **数据处理模型：**
    1. 使用 **偏微分方程（PDE）** 建模样品信号的变化，减少噪声对检测结果的干扰，提高数据分析的准确性。

4.4 智慧交通

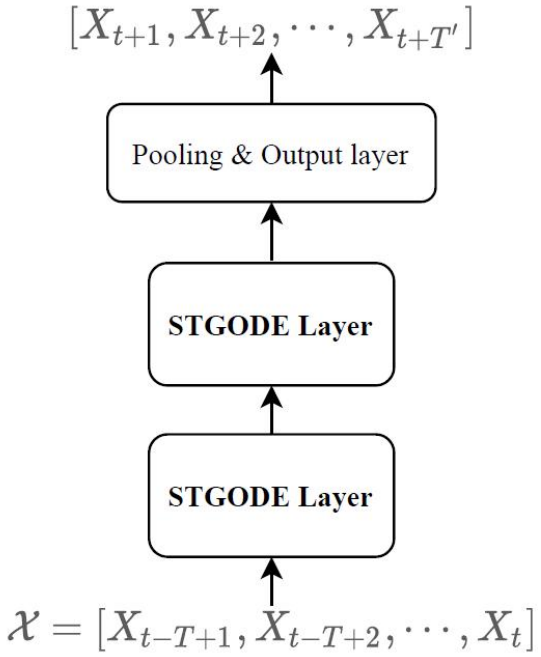
■ **现有研究存在问题：**

- 大多数方法将空间模式和时间模式分开建模，不考虑它们之间的相互作用，限制了模型的表示能力。
- 神经网络的层数越多，一般性能越好，而 GNN 从深度中获益甚微。很少有工作考虑在时空预测中网络深度的重要性，但这对于捕获长程依赖性至关重要。

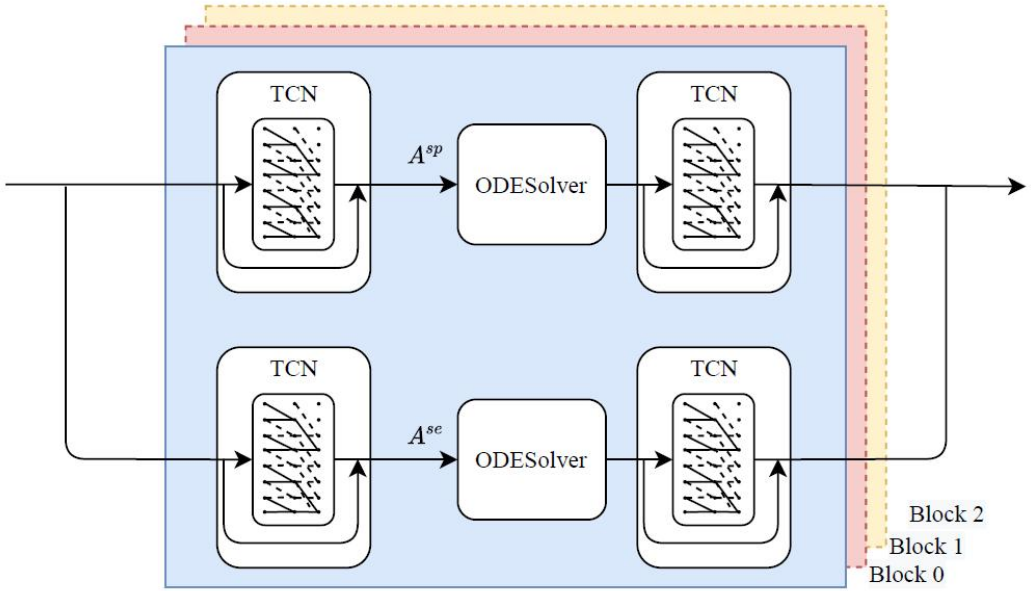
- 根据交通流的空间连通性和语义相似性构建空间邻接矩阵和语义邻接矩阵。
- 引入具有残差连接的连续 GNN 以避免过度平滑问题，建模长距离空间-时间依赖关系。具有残差连接的离散层可以视为微分方程（ODE）的离散化，可以推导出连续图神经网络（CGNN）。
- 构建空间-时间张量，同时考虑空间和时间模式，模拟复杂的空间-时间交互。

4.4 时空图常微分方程网络STGODE

问题描述：给定在交通网络 $\mathcal{G}$ 上观察到的张量 $\mathcal{X}$，交通预测的目标是从历史 T 观察到的学习一个映射函数 f，用于预测未来的 T' 交通观察。 $[X_{t-T+1}, X_{t-T+2}, \cdots, X_t; \mathcal{G}] \xrightarrow{f} [X_{t+1}, X_{t+2}, \cdots, X_{t+T'}]$



STGODE 网络的架构



STGODE 块细节

- Fang Z, Long Q, Song G, Xie K. Spatial-temporal graph ODE networks for traffic flow forecasting. In: *KDD*, 2021.

4.4 邻接矩阵

■ **空间邻接矩阵：** $A_{ij}^{sp} = \begin{cases} \exp\left(-\frac{d_{ij}^2}{\sigma^2}\right) & , \text{if} \exp\left(-\frac{d_{ij}^2}{\sigma^2}\right) \geq \epsilon \\ 0 & , \text{otherwise} \end{cases}$

d_{ij}：节点 i 和节点 j 间距离　　σ^2、ϵ：控制矩阵 A_{sp} 稀疏性的阈值

■ **DTW（动态时间规整）：** $D(i,j) = dist(x_i, y_j) + \min\left(D(i-1,j), D(i,j-1), D(i-1,j-1)\right)$

时间序列 $X=(x_1,x_2,\cdots,x_i)$、$Y=(y_1,y_2,\cdots,y_j)$　　$dist(x_i,y_j)$：某种距离度量

$D(i,j)$：子列 X 和 Y 之间的最短距离　　$DTW(X,Y)=D(m,n)$：X 和 Y 之间的最终距离

■ **DTW 距离定义语义邻接矩阵：** $A_{ij}^{SE} = \begin{cases} 1, DTW(X^i, X^j) < \epsilon \\ 0, \quad \text{otherwise} \end{cases}$

X_i：节点 i 的时间序列　　ϵ：决定邻接矩阵的稀疏程度

4.4 STGODE 学习框架

$$\mathcal{H}_l = \sum_{i=0}^{l} \left(\mathcal{H}_0 \times_1 \widehat{A}^i \times_2 U^i \times_3 W^i\right)$$

$$\Downarrow$$

$$\frac{\mathrm{d}\mathcal{H}(t)}{\mathrm{d}t} = \mathcal{H}_0 \times_1 \widehat{A}^{t+1} \times_2 U^{t+1} \times_3 W^{t+1}$$

$$\Downarrow$$

$$\frac{\mathrm{d}\mathcal{H}(t)}{\mathrm{d}t} = \mathcal{H}(t) \times_1 \ln \widehat{A} + \mathcal{H}(t) \times_2 \ln U + \mathcal{H}(t) \times_3 \ln W + \mathcal{H}_0$$

$$\Downarrow$$

$$\frac{\mathrm{d}\mathcal{H}(t)}{\mathrm{d}t} = \mathcal{H}(t) \times_1 (\widehat{A} - I) + \mathcal{H}(t) \times_2 (U - I) + \mathcal{H}(t) \times_3 (W - I) + \mathcal{H}_0$$

$$\Downarrow$$

$$\mathcal{H}(t) = ODESolve\left(\frac{\mathrm{d}\mathcal{H}(t)}{\mathrm{d}t}, \mathcal{H}_0, t\right)$$

$\mathcal{H}_l \in \mathbb{R}^{N \times T \times F}$：第 l 层节点隐藏嵌入的时空张量

$\times_i$：在模式 i 上的张量-矩阵乘法

$\hat{A}$：正则化邻接矩阵　　U：时间变换矩阵

W：特征变换矩阵　　$\mathcal{H}_0$：GNN 的初始输入

4.4 STGODE 学习框架

■ 时间卷积块

$$H_{tcn}^{l}=\begin{cases}X\\ \sigma(W^{l}*_{d^{l}}H_{tcn}^{l-1}),l=1,2,\ldots,L\end{cases}$$

$X\in\mathbb{R}^{N\times T\times F}$：TCN 输入　　W^{l}：第 l 个卷积核

$H^{l}_{tcn}\in\mathbb{R}^{N\times T\times F}$：TCN 第 l 层输出，

时间卷积采用指数扩张率 $d^{l}=2^{l-1}$

■ 损失函数（Huber 损失）

$$L(Y,\hat{Y})=\begin{cases}\frac{1}{2}(Y-\hat{Y})^{2} & ,|Y-\hat{Y}|\le\delta\\ \delta|Y-\hat{Y}|-\frac{1}{2}\delta^{2} & ,\text{otherwise}\end{cases}$$

δ：超参数，控制对异常值的敏感度

■ 实验数据集

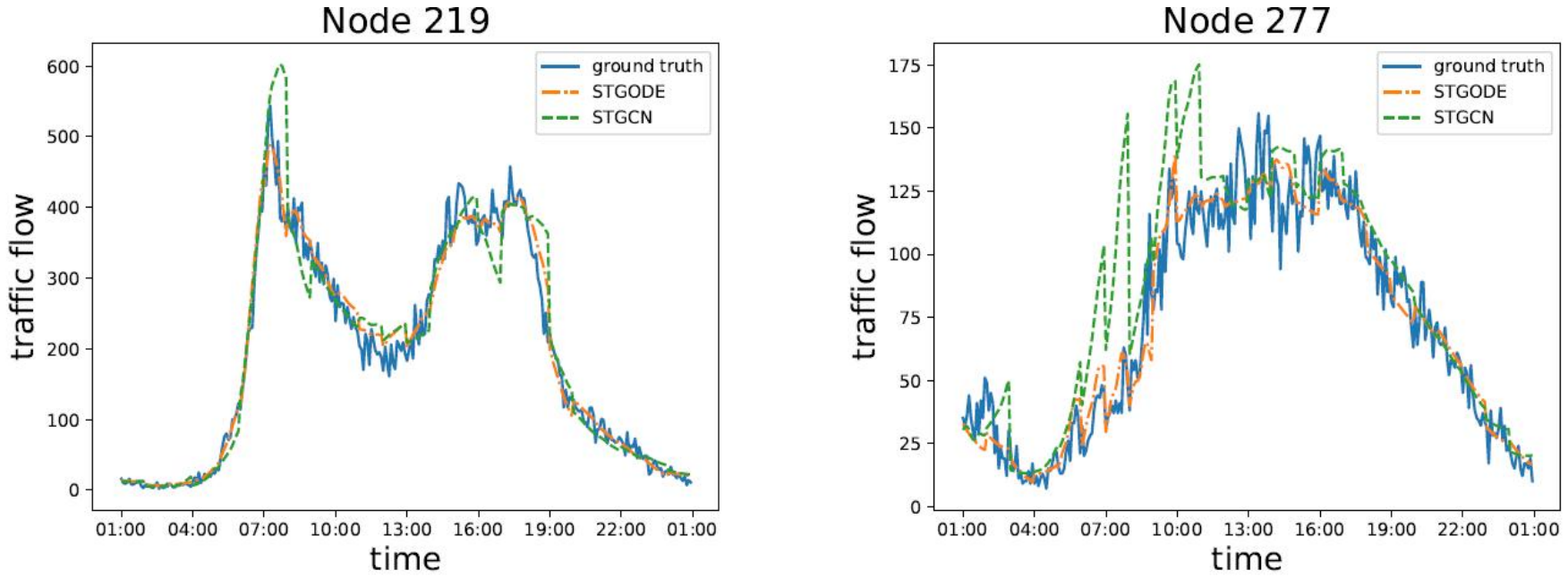
Datasets	#Sensors	#Edges	Time Steps
PeMSD7(M)	228	1132	12672
PeMSD7(L)	1026	10150	12672
PeMS03	358	547	26208
PeMS04	307	340	16992
PeMS07	883	866	28224
PeMS08	170	295	17856

数据集由加利福尼亚州交通部性能测量系统（PeMS）实时每 30 秒收集。数据集按照 6:2:2 的比例分为训练集、验证集和测试集。使用一小时的历史数据来预测接下来 60 分钟的交通状况。

4.4 空间-时间图 ODE 网络用于交通流量预测

Dataset	Metric	ARIMA	STGCN	DCRNN	ASTGCN(r)	GraphWaveNet	STSGCN	**STODE**
PeMSD7(M)	RMSE	13.20	7.55	7.18	6.87	6.24	5.93	**5.66**
	MAE	7.27	4.01	3.83	3.61	3.19	3.01	**2.97**
	MAPE	10.38	9.67	9.81	8.84	8.02	7.55	**7.36**
PeMSD7(L)	RMSE	12.39	8.28	8.33	7.64	7.09	6.88	**5.98**
	MAE	7.51	4.84	4.33	4.09	3.75	3.61	**3.22**
	MAPE	15.83	11.76	11.41	10.25	9.41	9.13	**7.94**
PeMS03	RMSE	47.59	30.42	30.31	29.56	32.77	29.21	**27.84**
	MAE	35.41	17.55	17.99	17.34	19.12	17.48	**16.50**
	MAPE	33.78	17.43	18.34	17.21	18.89	16.78	**16.69**
PeMS04	RMSE	48.80	36.01	37.65	35.22	39.66	33.65	**32.82**
	MAE	33.73	22.66	24.63	22.94	24.89	21.19	**20.84**
	MAPE	24.18	14.34	17.01	16.43	17.29	13.90	**13.77**
PeMS07	RMSE	59.27	39.34	38.61	37.87	41.50	39.03	**37.54**
	MAE	38.17	25.33	25.22	24.01	26.39	24.26	**22.99**
	MAPE	19.46	11.21	11.82	10.73	11.97	10.21	**10.14**
PeMS08	RMSE	44.32	27.88	27.83	26.22	30.04	26.80	**25.97**
	MAE	31.09	18.11	17.46	16.64	18.28	17.13	**16.81**
	MAPE	22.73	11.34	11.39	10.6	12.15	10.96	**10.62**

4.4 空间-时间图 ODE 网络用于交通流量预测



从道路网络选择两个节点进行案例研究。*STGODE*预测结果与地面真相相比明显更接近。在正常情况下，模型生成平滑的预测，忽略小振荡以对抗噪声。但当出现突变时，*STGODE*能够快速响应。*STGODE* 能够利用更远距离地理邻居和语义邻居的特征信息，准确捕捉实时动态并过滤无效信息，而 *STGCN* 作为浅层网络，对附近邻居的敏感度较低，因此表现不稳定。

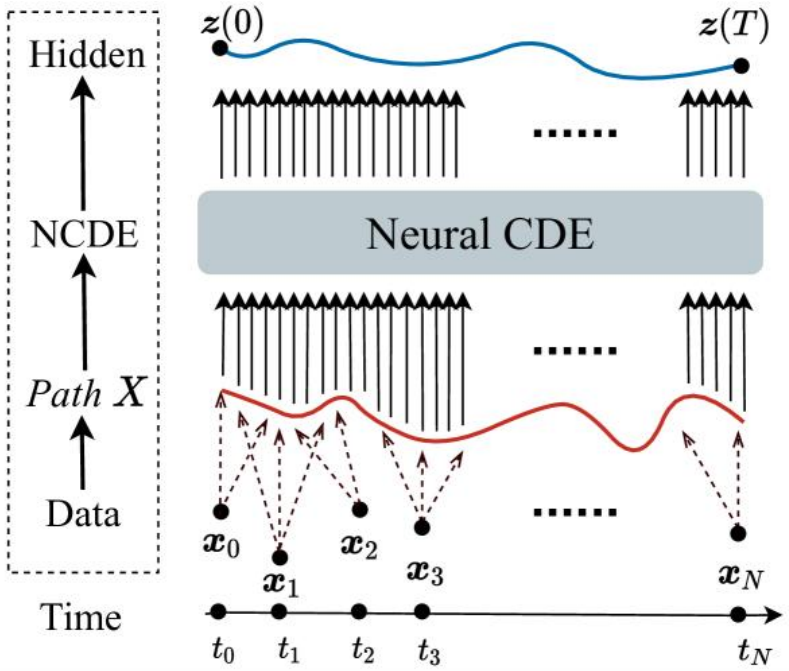
4.4 时空图神经控制微分方程STG-NCDE

基于神经控制微分方程的方法(NCDEs)：

$$z(T) = z(0) + \int_0^T f\left(z(t); \theta_f\right) dX(t)$$
$$= z(0) + \int_0^T f\left(z(t); \theta_f\right) \frac{dX(t)}{dt} dt.$$

$X(t)$：一个连续路径，其值位于巴拿赫空间中

$\boldsymbol{z}$ (t) 的整个轨迹通过路径 X 随时间进行控制

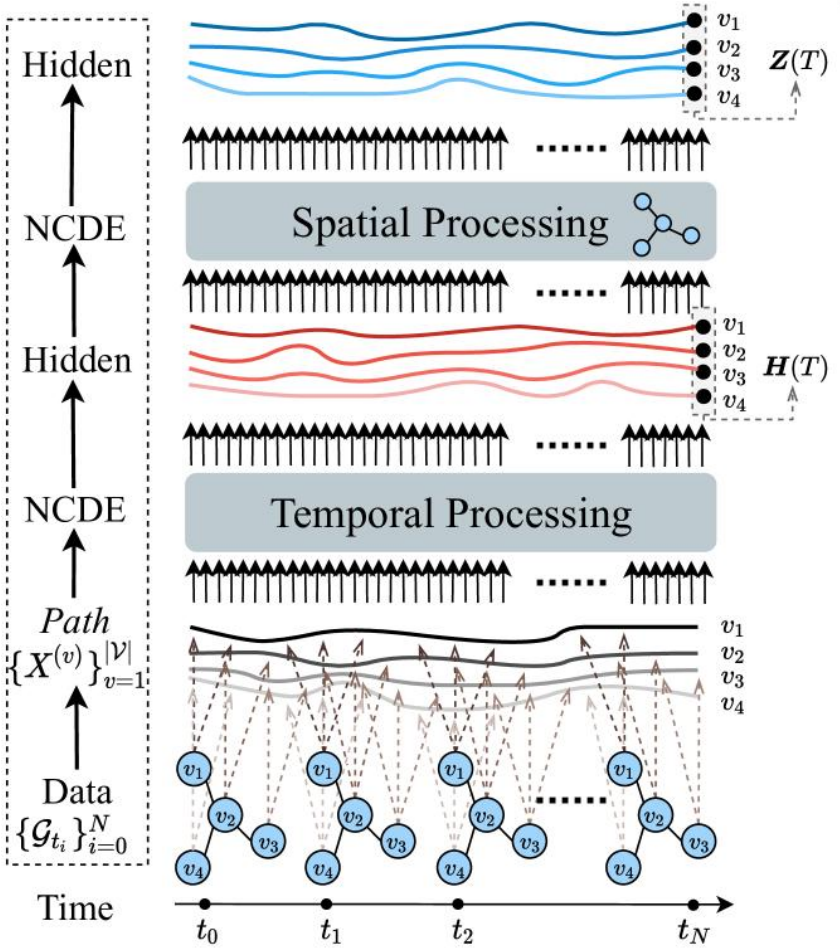


原始 *NCDE* 处理时间序列的整体工作流程

- Choi J, Choi H, Hwang J, Park N. Graph neural controlled differential equations for traffic forecasting. In: *AAAI*, 2022.

4.4 时空图神经控制微分方程STG-NCDE

- **预处理步骤**：为每个节点创建一条连续路径 $X^{(v)}$。给定一个离散的时间序列 $\{\boldsymbol{x}_i\}_{i=0}^{N}$，原始 *NCDE* 运行插值算法来构建其连续路径。分别对每个节点应用相同的方法，创建一组路径，表示为 $\{X^{(v)}\}_{v=1}^{|V|}$。
- **主要步骤**：对 $\{X^{(v)}\}_{v=1}^{|V|}$ 联合应用空间和时间处理方法，并考虑其图连接性。设计一个 NCDE 模型，模型在空间和时间处理方面都配备了图处理技术。然后，得出每个节点 v 的最后一个隐藏向量 $\boldsymbol{z}^{(v)}(T)$，以及最后输出层预测 $\hat{\boldsymbol{y}}^{(v)} \in \mathbb{R}^{S \times M}$。收集所有节点在 $\mathcal{V}$ 中的预测后，得到预测矩阵 $\hat{\boldsymbol{Y}} \in \mathbb{R}^{|\mathcal{V}| \times S \times M}$。



STG-NCDE 的整体工作流程

4.4 时空图神经控制微分方程STG-NCDE

- **时间处理** $\mathrm{H}(T)=\mathrm{H}(0)+\int_0^T f\left(\mathrm{H}(t);\theta_f\right)\frac{d\mathrm{X}(t)}{dt}dt$

- **空间处理** $\mathrm{Z}(T)=\mathrm{Z}(0)+\int_0^T g\left(\mathrm{Z}(t);\theta_g\right)f(\mathrm{H}(t);\theta_f)\frac{d\mathrm{X}(t)}{dt}dt$

$Z(t)\in\mathbb{R}^{|\mathcal{V}|\times\dim(\boldsymbol{z}\ (\mathrm{v}))}$ 是在将所有 v 的隐藏轨迹 $z(v)$ 堆叠后生成的矩阵

- **增强的*ODE*** $\frac{d}{dt}\begin{bmatrix}\mathrm{Z}(t)\\ \mathrm{H}(t)\end{bmatrix}=\begin{bmatrix}g(\mathrm{Z}(t);\theta_g)f(\mathrm{H}(t);\theta_f)\frac{d\mathrm{X}(t)}{dt}\\ f(\mathrm{H}(t);\theta_f)\frac{d\mathrm{X}(t)}{dt}\end{bmatrix}$ $\hat{\mathrm{y}}^{(v)}=z^{(v)}(T)\mathrm{W}_{output}+b_{output}$

$\boldsymbol{W}_{\mathrm{output}}\in\mathbb{R}^{\dim(\boldsymbol{z}\ (\mathrm{v})(\mathrm{T}))\to\mathrm{S}\times\mathrm{M}}$ $\boldsymbol{b}_{\mathrm{output}}\in\mathbb{R}^{\mathrm{S}\times\mathrm{M}}$ 输出层可训练权重和偏置

- ***L*¹ 损失函数作为训练目标**

$$\mathcal{L}=\frac{\sum_{\tau\in\mathcal{J}}\sum_{v\in\mathcal{V}}\|\mathbf{y}^{(\tau,v)}-\hat{\mathbf{y}}^{(\tau,v)}\|_1}{|\mathcal{V}|\times|\mathcal{T}|}$$

$\mathcal{T}$ 是训练集　　τ 是训练样本

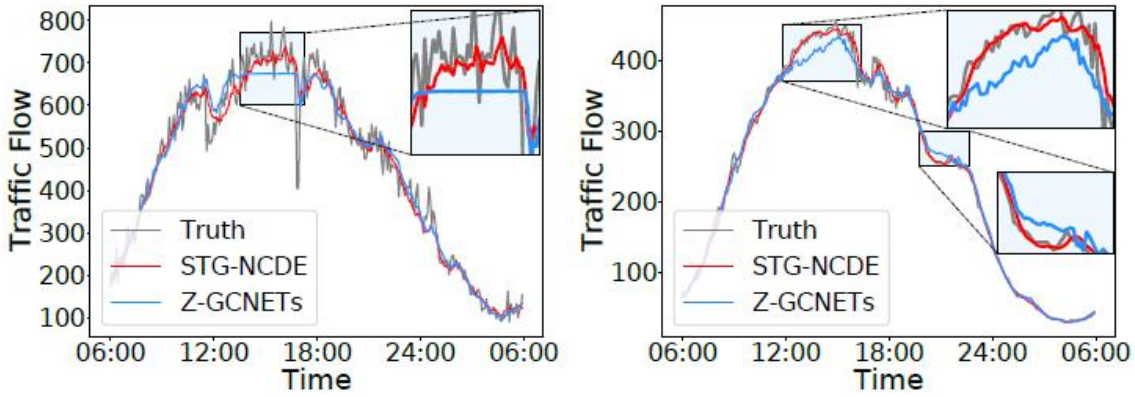
$\boldsymbol{y}\ (\tau,\ v)$ 是 τ 中节点 v 的真实值

4.4 用于交通预测的图神经可控微分方程

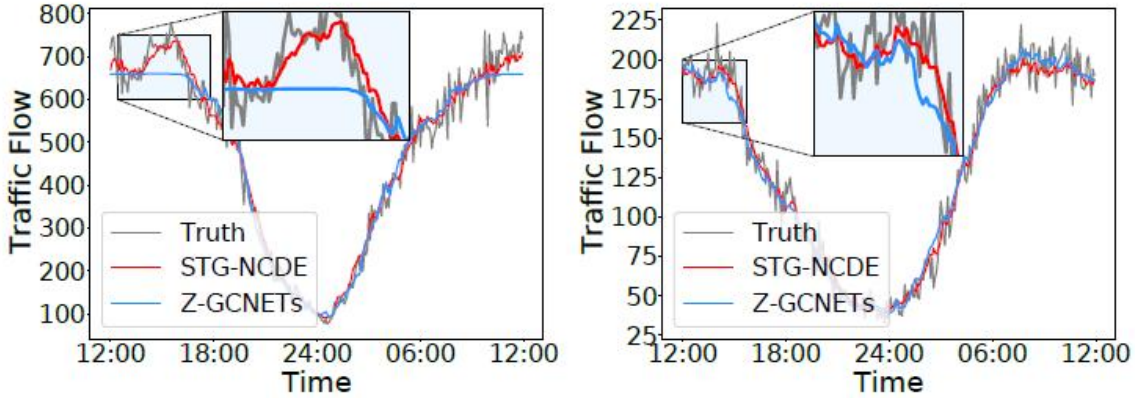
Dataset	$\lvert\mathcal{V}\rvert$	Time Steps	Time Range	Type
PeMSD3	358	26,208	09/2018 - 11/2018	Volume
PeMSD4	307	16,992	01/2018 - 02/2018	Volume
PeMSD7	883	28,224	05/2017 - 08/2017	Volume
PeMSD8	170	17,856	07/2016 - 08/2016	Volume
PeMSD7(M)	228	12,672	05/2012 - 06/2012	Velocity
PeMSD7(L)	1,026	12,672	05/2012 - 06/2012	Velocity

Model	PeMSD3			PeMSD4			PeMSD7			PeMSD8			PeMSD7(M)			PeMSD7(L)		
	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE
HA	31.58	52.39	33.78%	38.03	59.24	27.88%	45.12	65.64	24.51%	34.86	59.24	27.88%	4.59	8.63	14.35%	4.84	9.03	14.90%
ARIMA	35.41	47.59	33.78%	33.73	48.80	24.18%	38.17	59.27	19.46%	31.09	44.32	22.73%	7.27	13.20	15.38%	7.51	12.39	15.83%
VAR	23.65	38.26	24.51%	24.54	38.61	17.24%	50.22	75.63	32.22%	19.19	29.81	13.10%	4.25	7.61	10.28%	4.45	8.09	11.62%
FC-LSTM	21.33	35.11	23.33%	26.77	40.65	18.23%	29.98	45.94	13.20%	23.09	35.17	14.99%	4.16	7.51	10.10%	4.66	8.20	11.69%
TCN	19.32	33.55	19.93%	23.22	37.26	15.59%	32.72	42.23	14.26%	22.72	35.79	14.03%	4.36	7.20	9.71%	4.05	7.29	10.43%
TCN(w/o causal)	18.87	32.24	18.63%	22.81	36.87	14.31%	30.53	41.02	13.88%	21.42	34.03	13.09%	4.43	7.53	9.44%	4.58	7.77	11.53%
GRU-ED	19.12	32.85	19.31%	23.68	39.27	16.44%	27.66	43.49	12.20%	22.00	36.22	13.33%	4.78	9.05	12.66%	3.98	7.71	10.22%
DSANet	21.29	34.55	23.21%	22.79	35.77	16.03%	31.36	49.11	14.43%	17.14	26.96	11.32%	3.52	6.98	8.78%	3.66	7.20	9.02%
STGCN	17.55	30.42	17.34%	21.16	34.89	13.83%	25.33	39.34	11.21%	17.50	27.09	11.29%	3.86	6.79	10.06%	3.89	6.83	10.09%
DCRNN	17.99	30.31	18.34%	21.22	33.44	14.17%	25.22	38.61	11.82%	16.82	26.36	10.92%	3.83	7.18	9.81%	4.33	8.33	11.41%
GraphWaveNet	19.12	32.77	18.89%	24.89	39.66	17.29%	26.39	41.50	11.97%	18.28	30.05	12.15%	3.19	6.24	8.02%	3.75	7.09	9.41%
ASTGCN(r)	17.34	29.56	17.21%	22.93	35.22	16.56%	24.01	37.87	10.73%	18.25	28.06	11.64%	3.14	6.18	8.12%	3.51	6.81	9.24%
MSTGCN	19.54	31.93	23.86%	23.96	37.21	14.33%	29.00	43.73	14.30%	19.00	29.15	12.38%	3.54	6.14	9.00%	3.58	6.43	9.01%
STG2Seq	19.03	29.83	21.55%	25.20	38.48	18.77%	32.77	47.16	20.16%	20.17	30.71	17.32%	3.48	6.51	8.95%	3.78	7.12	9.50%
LSGCN	17.94	29.85	16.98%	21.53	33.86	13.18%	27.31	41.46	11.98%	17.73	26.76	11.20%	3.05	5.98	7.62%	3.49	6.55	8.77%
STSGCN	17.48	29.21	16.78%	21.19	33.65	13.90%	24.26	39.03	10.21%	17.13	26.80	10.96%	3.01	5.93	7.55%	3.61	6.88	9.13%
AGCRN	15.98	28.25	15.23%	19.83	32.26	12.97%	22.37	36.55	9.12%	15.95	25.22	10.09%	2.79	5.54	7.02%	2.99	5.92	7.59%
STFGNN	16.77	28.34	16.30%	20.48	32.51	16.77%	23.46	36.60	9.21%	16.94	26.25	10.60%	2.90	5.79	7.23%	2.99	5.91	7.69%
STGODE	16.50	27.84	16.69%	20.84	32.82	13.77%	22.59	37.54	10.14%	16.81	25.97	10.62%	2.97	5.66	7.36%	3.22	5.98	7.94%
Z-GCNETs	16.64	28.15	16.39%	19.50	31.61	12.78%	21.77	35.17	9.25%	15.76	25.11	10.01%	2.75	5.62	6.89%	2.91	5.83	7.33%
STG-NCDE	**15.57**	**27.09**	**15.06%**	**19.21**	**31.09**	**12.76%**	**20.53**	**33.84**	**8.80%**	**15.45**	**24.81**	**9.92%**	**2.68**	**5.39**	**6.76%**	**2.87**	**5.76**	**7.31%**
Only temporal	20.44	32.82	20.03%	26.31	40.97	17.95%	28.77	44.39	12.60%	20.83	32.55	13.01%	3.34	6.68	8.41%	3.54	7.03	8.89%
Only spatial	15.92	27.17	15.14%	19.86	31.92	13.35%	21.72	34.73	9.24%	17.58	27,76	11.27%	2.77	5.40	7.00%	2.99	5.85	7.60%

4.4 用于交通预测的图神经可控微分方程

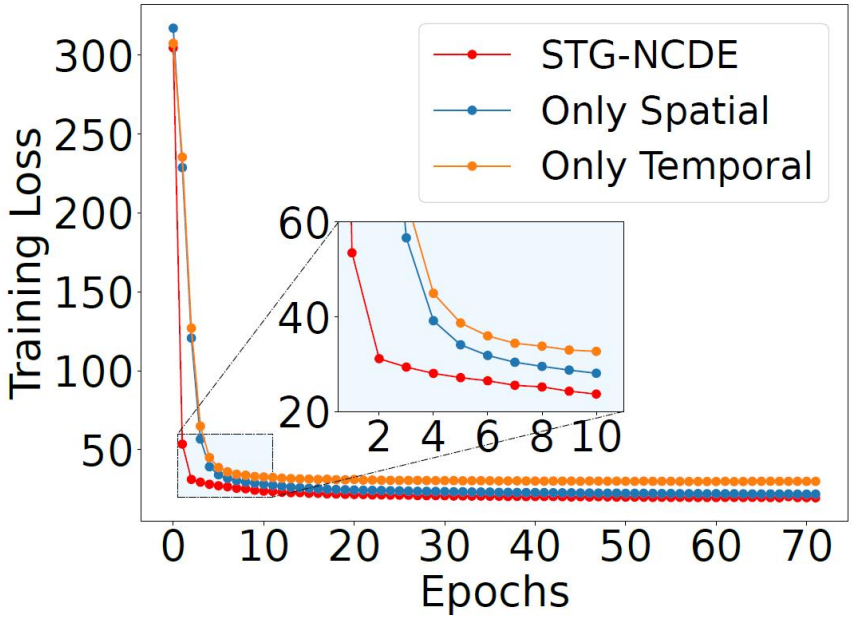


(a) Node 111 in PeMSD4　　(b) Node 261 in PeMSD4



(c) Node 9 in PeMSD8　　(d) Node 112 in PeMSD8

消融实验



节点 111 和 261（分别对应节点 9 和 112）是 PeMSD4（分别对应 PeMSD8）中的两个交通流量最高的区域。

4.4 用于交通预测的图神经可控微分方程

■ 不规则交通预测

实际上，交通传感器可能会损坏，有些区域在一定时间内无法收集数据。为反映这种情况，独立地为每个节点随机丢弃 10%到 50%的感应值。

Model	Missing rate	MAE	RMSE	MAPE
STG-NCDE		19.36	31.28	12.79%
Only Temporal	10%	26.26	40.89	17.66%
Only Spatial		19.73	31.67	13.20%
STG-NCDE		19.40	31.30	13.04%
Only Temporal	30%	26.86	41.73	18.35%
Only Spatial		19.83	31.95	13.29%
STG-NCDE		19.98	32.09	13.48%
Only Temporal	50%	28.15	43.54	19.14%
Only Spatial		20.14	32.30	13.30%

PeMSD4

Model	Missing rate	MAE	RMSE	MAPE
STG-NCDE		15.68	24.96	10.05%
Only Temporal	10%	21.18	33.02	13.26%
Only Spatial		16.85	26.63	11.12%
STG-NCDE		16.21	25.64	10.43%
Only Temporal	30%	21.46	33.37	13.57%
Only Spatial		18.46	29.03	12.16%
STG-NCDE		16.68	26.17	10.67%
Only Temporal	50%	22.68	35.14	14.11%
Only Spatial		17.98	28.12	11.87%

PeMSD8

谢谢

福州大学 计算机与大数据学院

智慧地铁福建省高校重点实验室

https://rail.fzu.edu.cn/