深圳湾实验室
Shenzhen Bay Laboratory

Advanced
Intelligent
Molecular
Modeling Group
先进智能分子模拟课题组

Cybertron: 基于MindSpore的深度分子模型通用架构

杨奕 博士 副研究员
Dr. Yi Isaac Yang

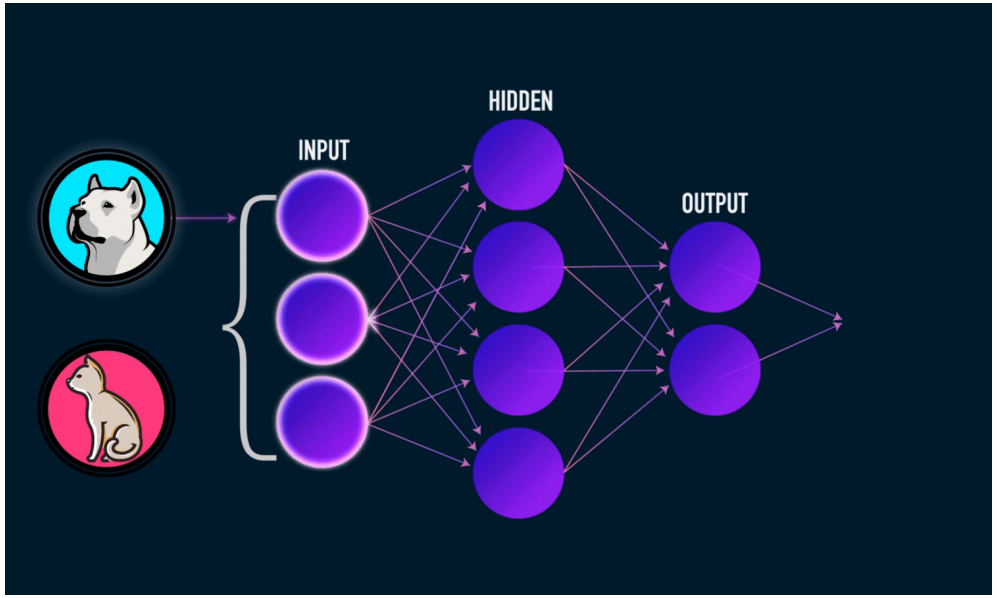
深圳湾实验室 系统与物理生物学研究所
Institute of Systems and Physical Biology, Shenzhen Bay Laboratory

Email: yangyi@szbl.ac.cn

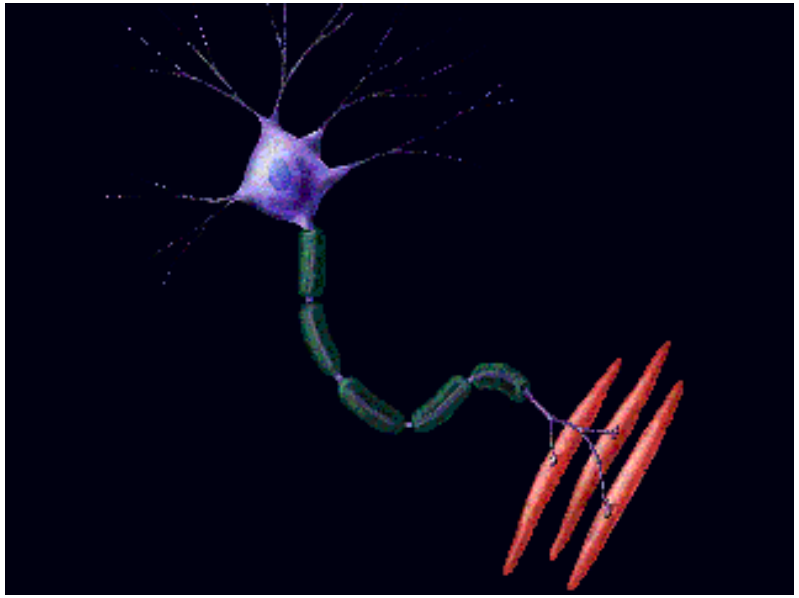
9 May 2022

深圳湾实验室
Shenzhen Bay Laboratory

Artificial Neural Networks (ANN)



Artificial Neural Network

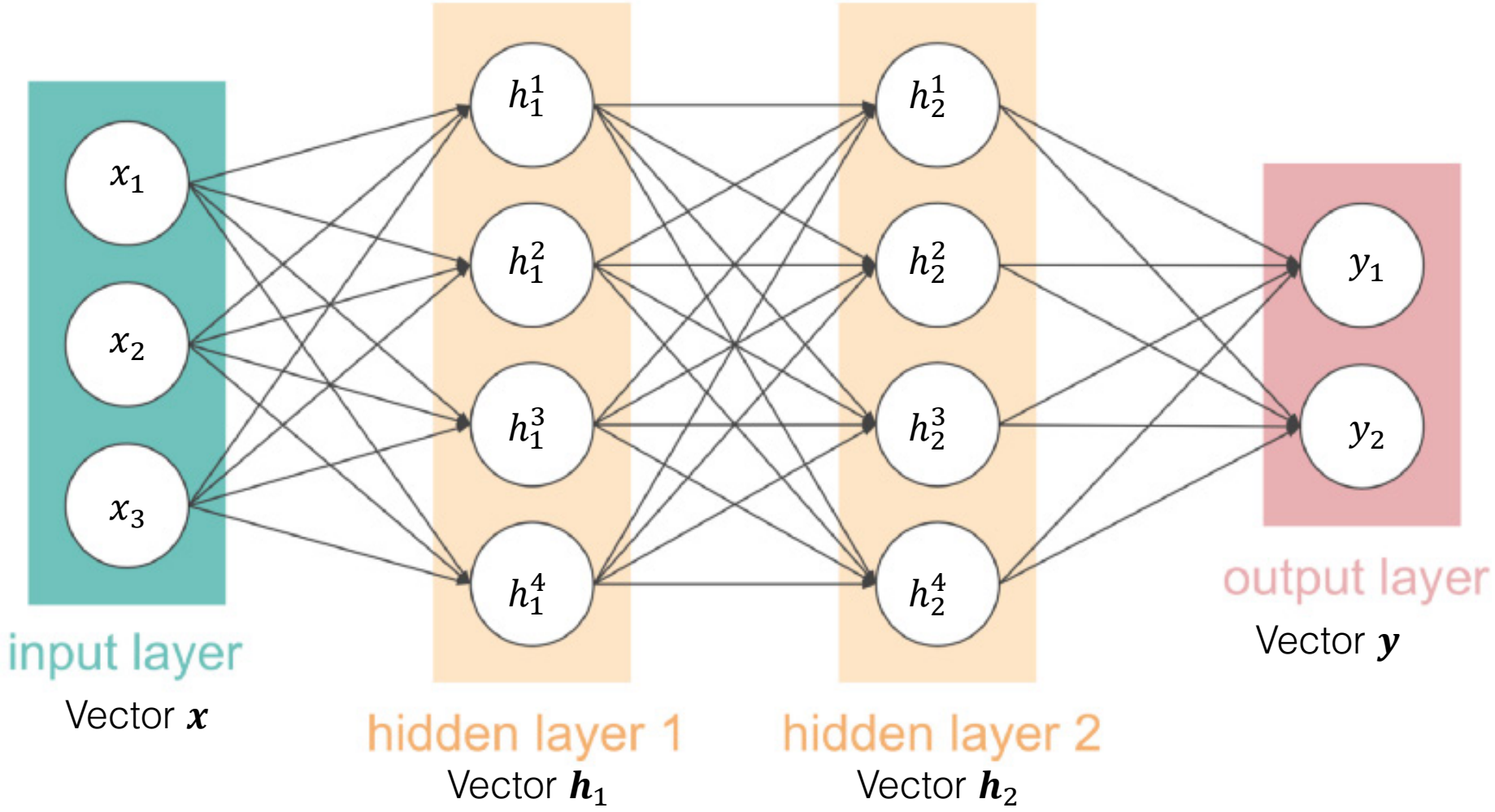


Biological Neural Network

万能近似定理（Universal Approximation Theorems）：
如果一个前馈神经网络具有线性输出层和至少一层隐藏层，只要给予网络足够数量的神经元，便可以实现以足够高精度来逼近任意一个在实数空间上的连续函数。

深圳湾实验室
Shenzhen Bay Laboratory

多层感知机（MLP）



深圳湾实验室
Shenzhen Bay Laboratory

神经网络的前向传播
Forward Propagation

- MLP的前向传播计算：

$$\boldsymbol{z}_i = \boldsymbol{W}_i \cdot \boldsymbol{h}_{i-1} + \boldsymbol{b}_i$$
$$h_i = f_i(z_i)$$

- W_i和b_i分别称为“weight”和“bias”，其中$\boldsymbol{W}_i$是一个矩阵而$\boldsymbol{b}_i$是一个向量。
- $f(z)$是一个非线性函数，称为“激活函数”，用于保证网络的非线性。

- 一些典型的激活函数：
- Sigmoid:

$$f(z) = \frac{1}{1+e^{-z}}$$

- ReLU:

$$f(z) = \max\{0, z\}$$

- TanH:

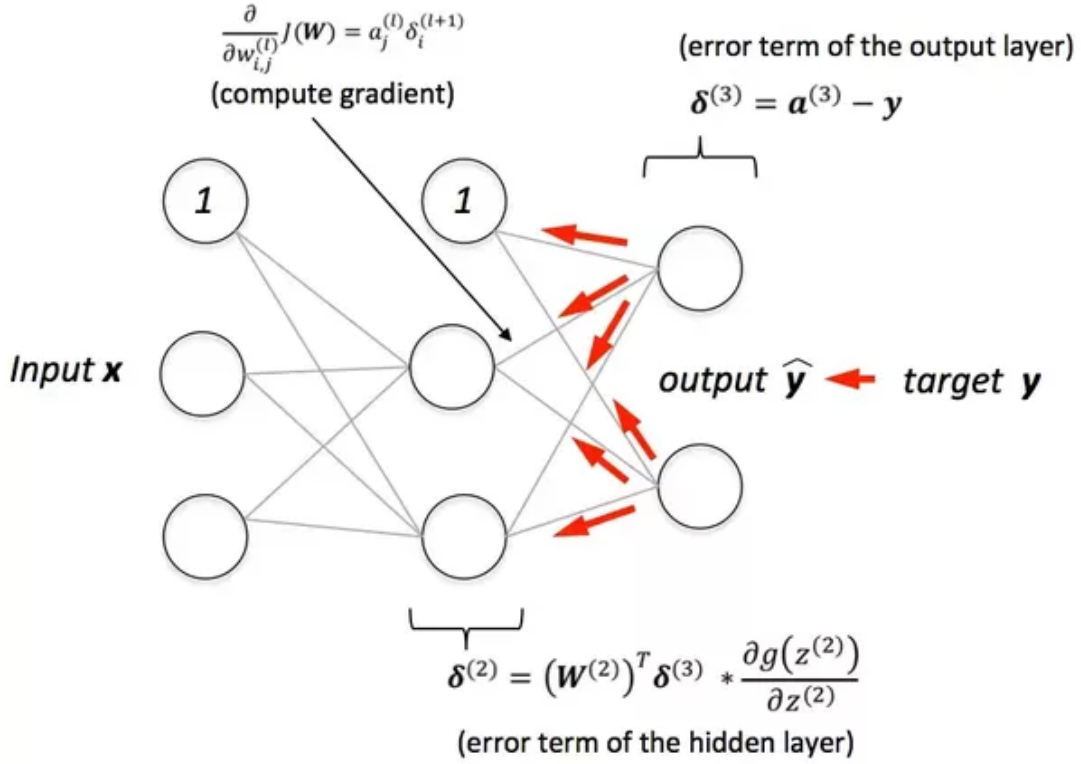
$$f(z) = \tanh(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$$

- Softmax:

$$f(z_i) = \frac{z_i}{\sum_j z_j}$$

深圳湾实验室
Shenzhen Bay Laboratory

梯度的计算
Backward Propagation



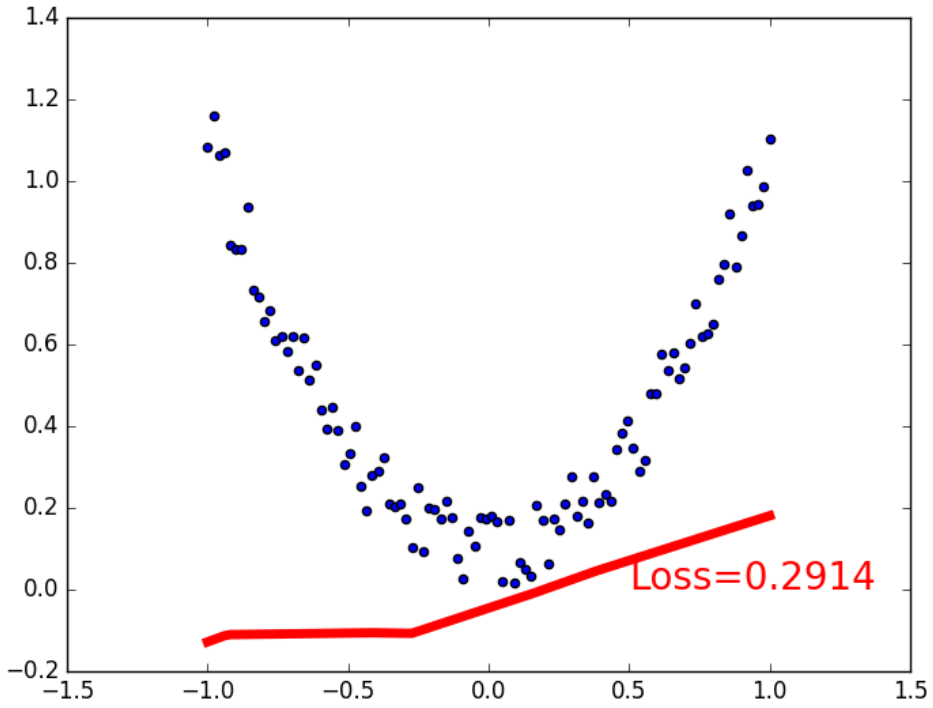
- MLP的第i层 $\boldsymbol{h}^i$相对于输入向量$\boldsymbol{x}$的导数可以通过链式法则从后往前进行计算，这一过程称为“反向传播”。

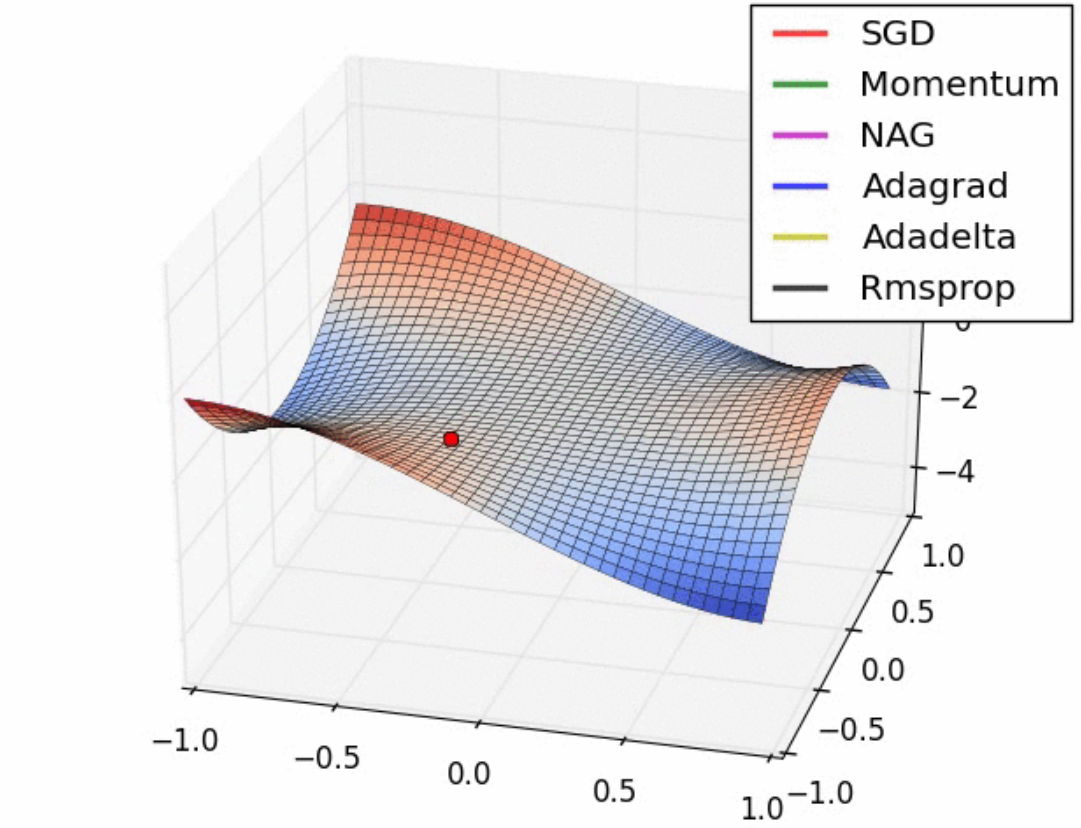
$$\frac{\partial \boldsymbol{h}_i}{\partial \boldsymbol{x}} = \frac{\partial \boldsymbol{h}_i}{\partial \boldsymbol{z}_i} \cdot \frac{\partial \boldsymbol{z}_i}{\partial \boldsymbol{h}_{i-1}} \cdot \frac{\partial \boldsymbol{h}_{i-1}}{\partial \boldsymbol{x}} = f_i'(\boldsymbol{z}_i)\boldsymbol{W}_i \cdot \frac{\partial \boldsymbol{h}_{i-1}}{\partial \boldsymbol{x}}$$

1. Sigmoid: $f(z) = \frac{1}{1+e^{-z}}$,
$$f'(z) = f(z)[1 - f(z)]$$
2. ReLU: $f(z) = \max\{0, z\}$
$$f'(z) = \frac{f(z)}{f(z) + \epsilon} = \begin{cases} 1 & (z \geq 0) \\ 0 & (z < 0) \end{cases}$$
3. TanH $f(z) = \tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$
$$f'(z) = \tanh'(z) = 1 - \tanh^2(z)$$
4. Softmax: $s_i = f(\boldsymbol{x}) = \frac{e^{z_i}}{\sum_i e^{z_j}}$
$$s_i' = s_i(\delta_{ij} - s_j)$$

深圳湾实验室
Shenzhen Bay Laboratory

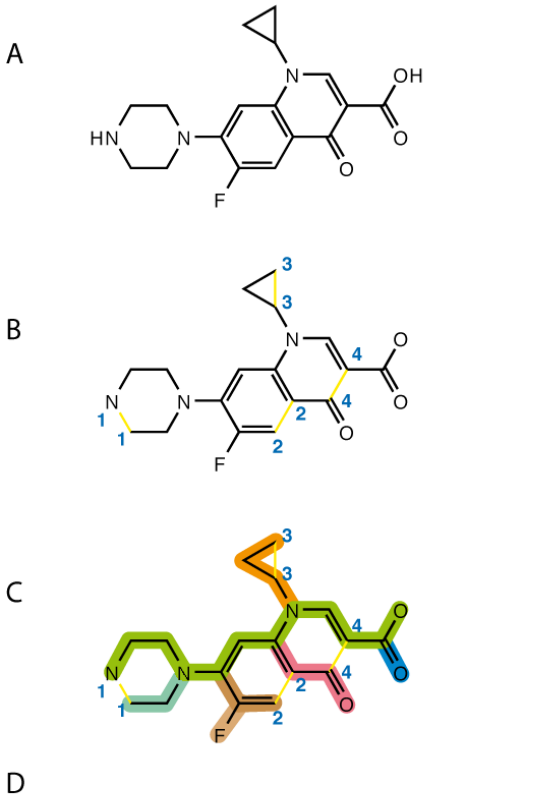
损失函数的优化





深圳湾实验室
Shenzhen Bay Laboratory

使用神经网络描述分子的构型与构象



分子**构型**的SMILES表示法

- 深度分子模型（Deep Molecular Model）：可以根据分子**构象**（原子空间坐标）预测分子性质及其导数（如作用力）神经网络模型。

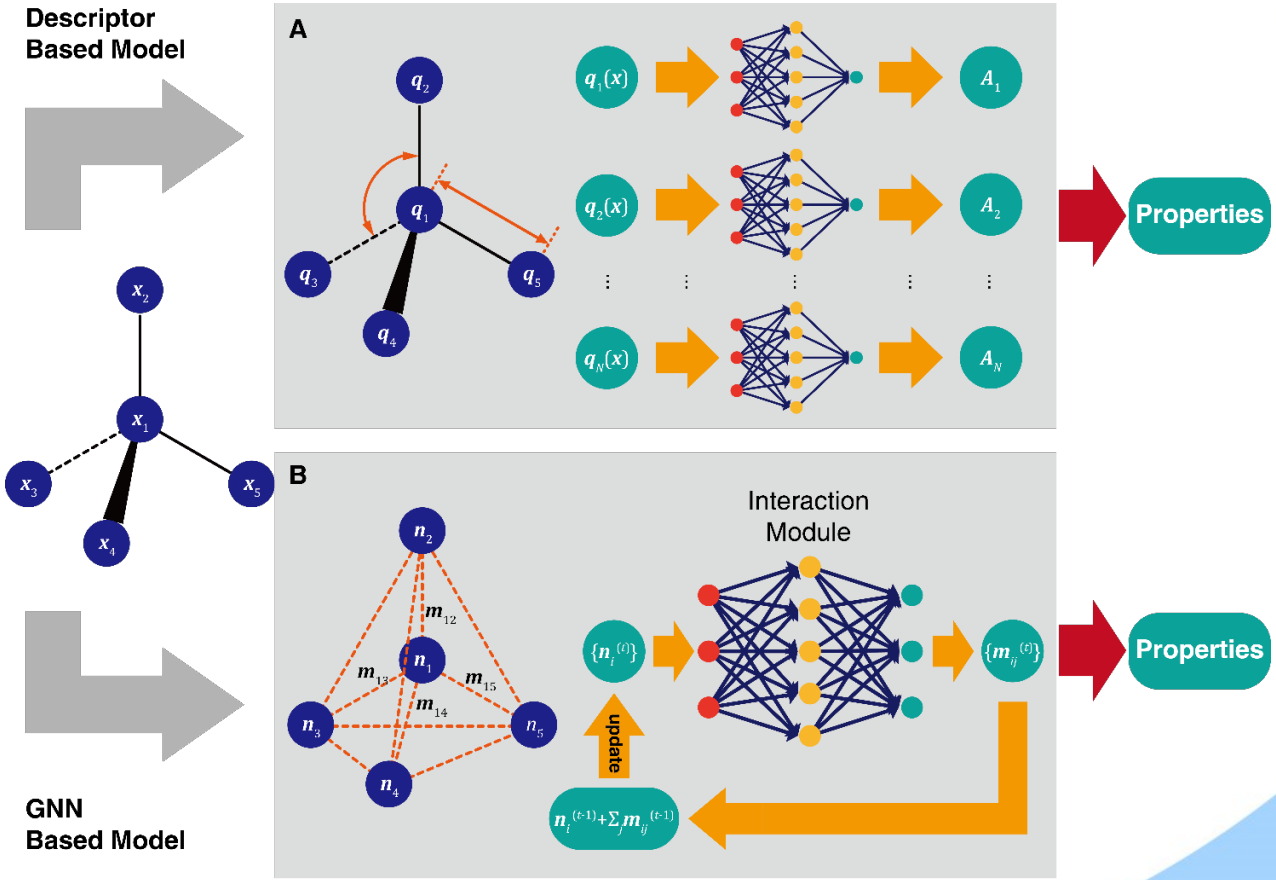
- 深度分子模型需要满足的归纳偏置条件：
  1. **参考系不变性**：网络的输出不因分子的旋转和平移而发生变化
  2. **简并性**：网络的输出不因相同类型的原子交换位置而发生变化
  3. **光滑性**：网络的输出随着分子构象的变化应是光滑的，即网络的输出应保证连续可导
  4. **可迁移性**：网络参数迁移到相似体系后经过少量训练即可使用

- 现有的深度分子模型架构：
  - 基于**描述符**（Descriptor）的模型
  - 基于**图神经网络**（Graph Neural Network）的模型

深圳湾实验室
Shenzhen Bay Laboratory

深度分子模型

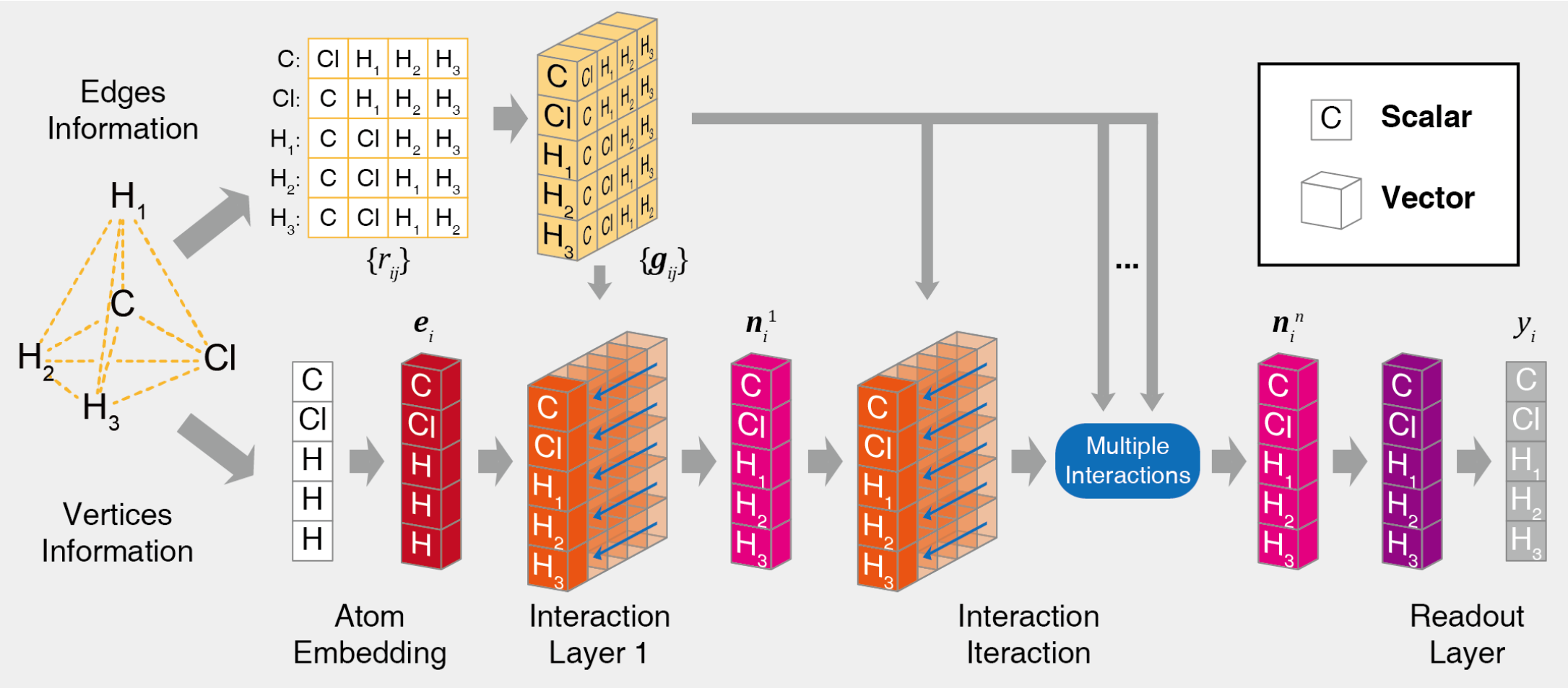
- 基于描述符（Descriptor）的模型：
  - BPNN (2007)
  - ANI-1 (2017)
  - TensorMol (2018)
  - DPMD (2018)
- 基于图神经网络GNN（Graph Neural Network）的模型：
  - DTNN (2017)
  - MPNN (2017)
  - SchNet (2018)
  - PhysNet (2019)
  - DeepMoleNet (2021)
  - MolCT (2021)



Zhang, J.; Lei, Y.-K.; Zhang, Z.; *et al.* A Perspective on Deep Learning for Molecular Modeling and Simulations *J. Phys. Chem. A* **2020**, *124*, 6745

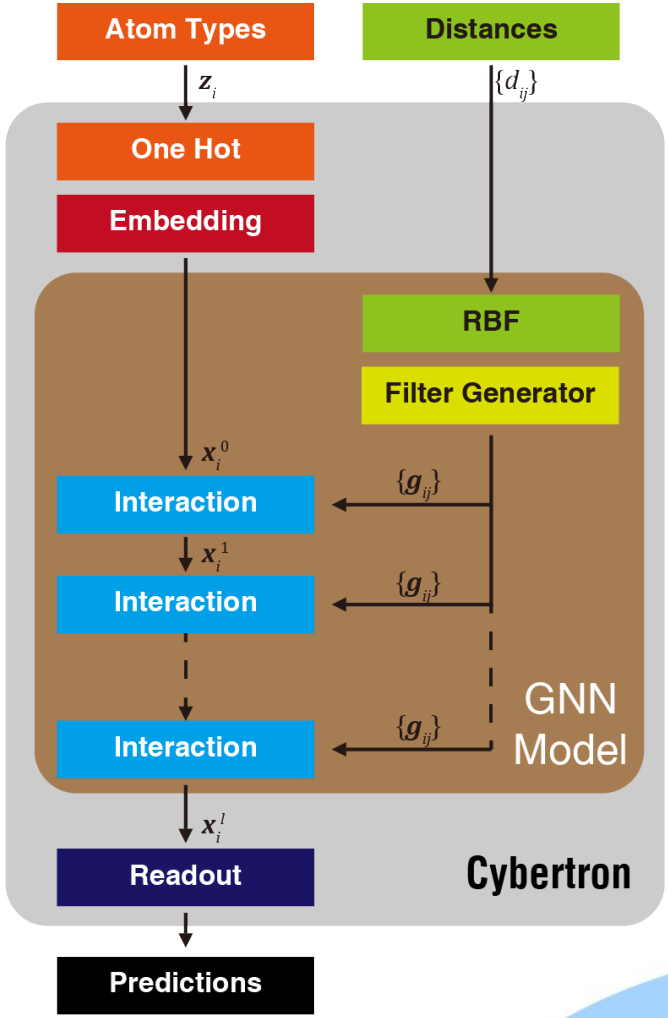
深圳湾实验室
Shenzhen Bay Laboratory

基于GNN的深度分子模型



深圳湾实验室
Shenzhen Bay Laboratory

Cybertron：通用图神经网络分子模型架构

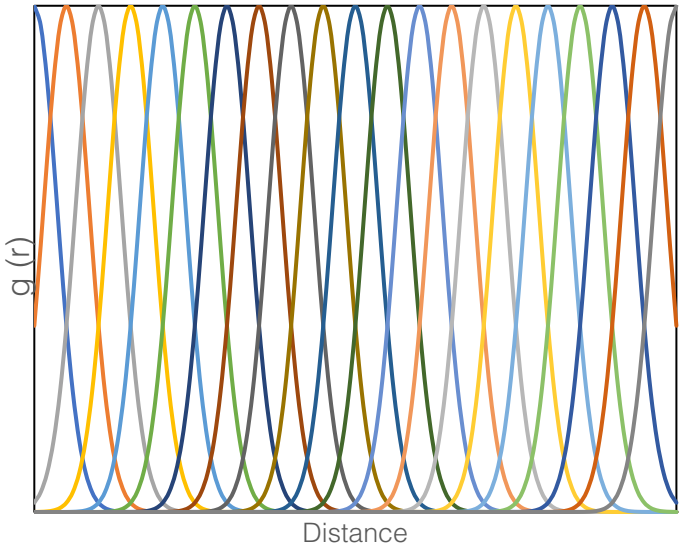


- 输入：
  - 节点坐标（R）
  - 节点类型（Z）
  - 原子类型（A）
  - 原子间距（D）
  - 邻居表（N）
  - 邻居表mask（n）
  - 键连信息（B）
  - 键连mask（b）
- 输出：预测性质（E）
- 输出对节点坐标的导数：作用力（F）
- 模型：SchNet、PhysNet、MolCT
- Readout类型：
  - Atomwise
  - Graph

深圳湾实验室
Shenzhen Bay Laboratory

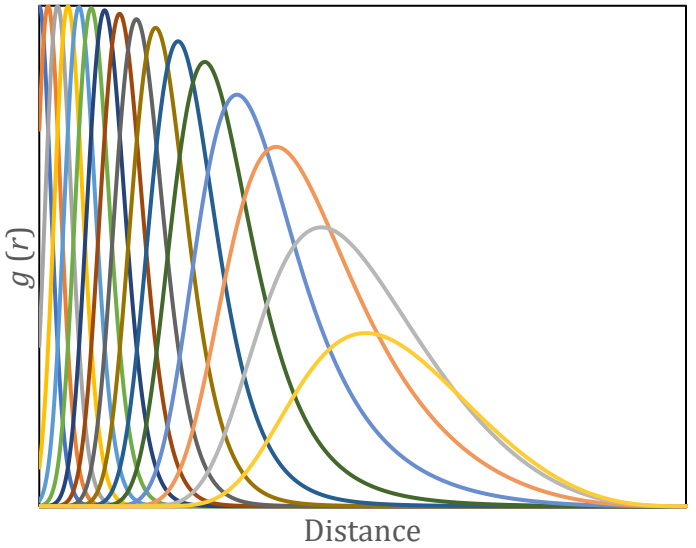
Radical Basis Functions （RBF）

SchNet



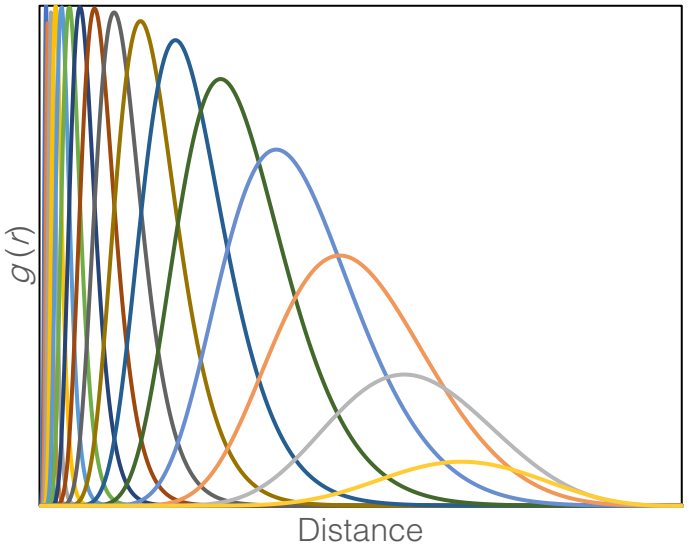
$$g_k(d_{ij}) = \exp\left[-\gamma\left(d_{ij} - \mu_k\right)^2\right],$$
$$\mu_k \in \{0, d_{\mathrm{Max}}\}$$

PhysNet



$$\mu_k \in \{\exp(-\alpha), 1\}, \qquad g_k(d_{ij}) =$$
$$\phi(d_{ij}) \exp\left\{-\beta_k\left[\frac{K}{2}\frac{\exp\left(-\alpha\frac{d_{ij}}{d_{\max}}\right) - \mu_k}{1 - \exp(-\alpha)}\right]^2\right\}$$

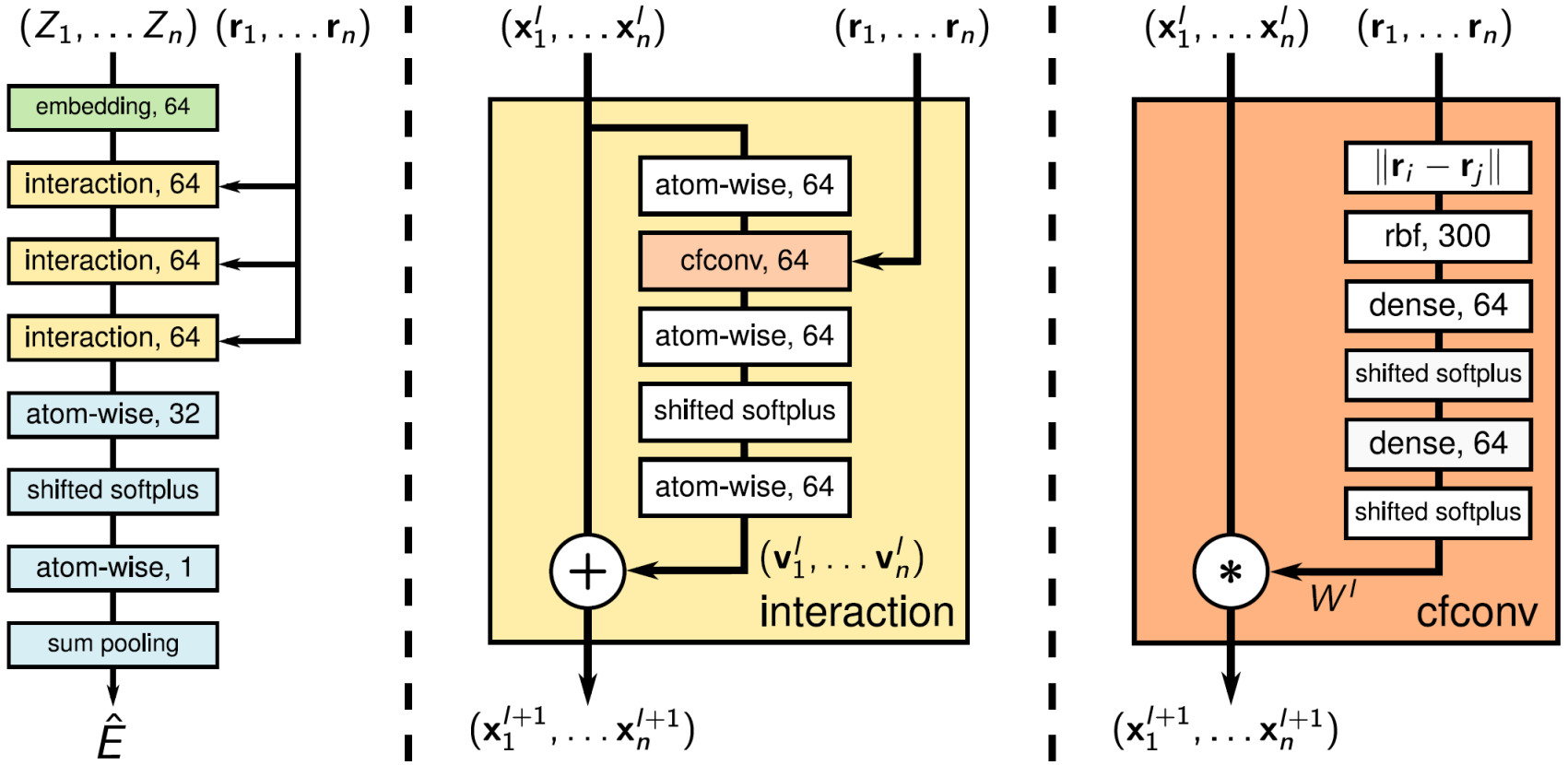
MolCT



$$\mu_k \in \left\{\log\left(\frac{d_{\mathrm{Min}}}{d_{\mathrm{Max}}}\right), 0\right\}, \qquad g_k(d_{ij}) =$$
$$\phi(d_{ij}) \exp\left\{-\frac{\left[\log\left(\frac{d_{ij}}{d_{\mathrm{Max}}}\right) - \mu_k\right]^2}{2\sigma^2}\right\}$$

$$\phi(d_{ij}) = \begin{cases} 1 - 6\left(\frac{d_{ij}}{d_{\mathrm{cutoff}}}\right)^5 + 15\left(\frac{d_{ij}}{d_{\mathrm{cutoff}}}\right)^4 - 10\left(\frac{d_{ij}}{d_{\mathrm{cutoff}}}\right)^3, & d_{ij} < d_{\mathrm{cutoff}} \\ 0, & d_{ij} \geq d_{\mathrm{cutoff}} \end{cases}$$

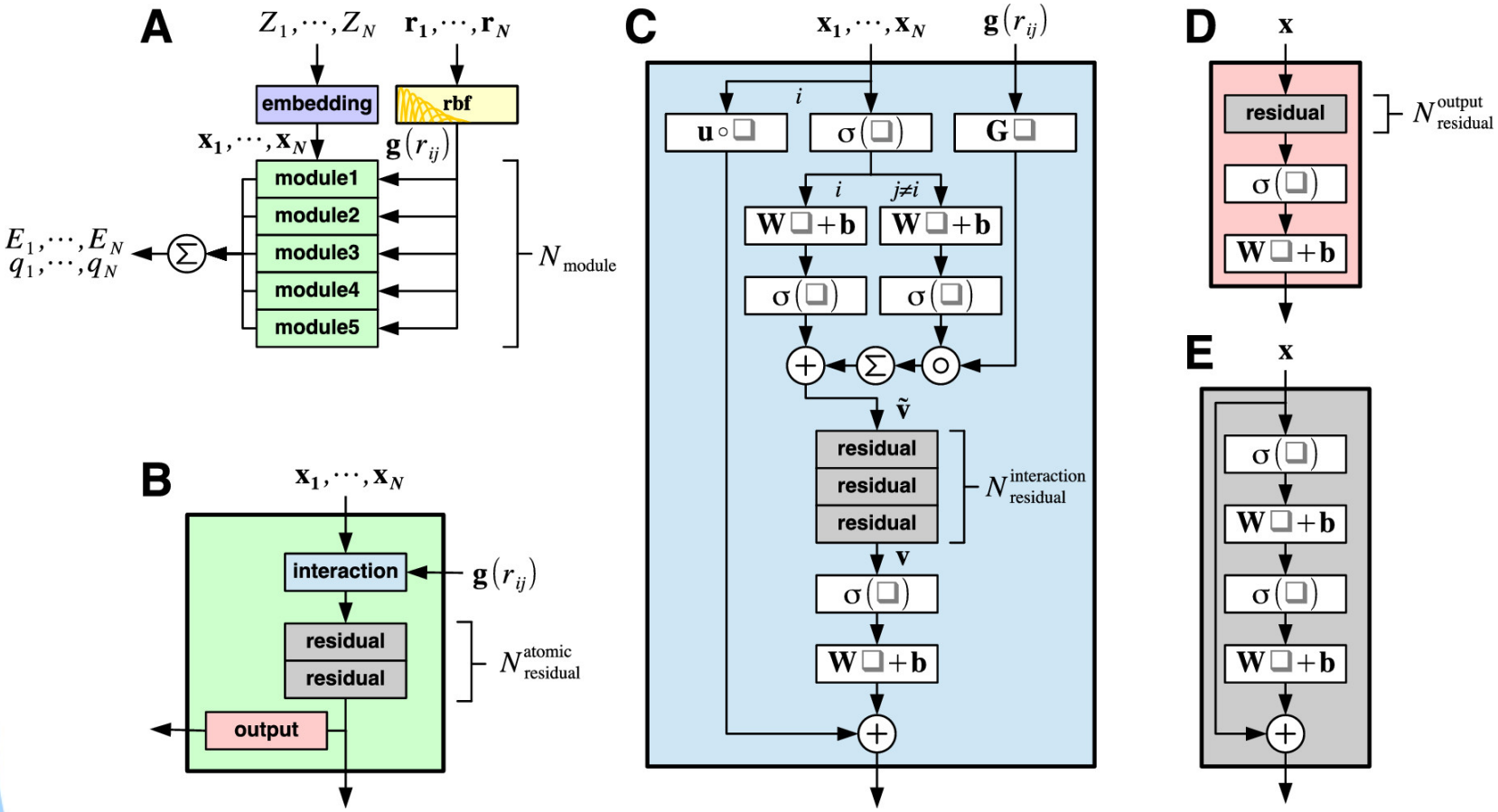
SchNet结构



- Atom-wise:
$$\boldsymbol{x}_n^{l+1} = \boldsymbol{W}_{aw}^l * \boldsymbol{x}_n^l + \boldsymbol{b}_{aw}^l$$
- Dense block:
$$\boldsymbol{h}_{ij}^1 = \sigma\left[\boldsymbol{W}_d^1 * \boldsymbol{g}(d_{ij}) + \boldsymbol{b}_d^1\right]$$
$$\boldsymbol{h}_{ij}^2 = \sigma\left[\boldsymbol{W}_d^2 * \boldsymbol{h}_{ij}^1 + \boldsymbol{b}_d^2\right]$$
- 卷积操作：
$$\boldsymbol{x}_n^{l+1} = \left(\boldsymbol{H} * \boldsymbol{x}_n^l\right)_n = \sum_k^{N-1} \boldsymbol{x}_k^l \circ \boldsymbol{h}_{nk}$$
- 最终输出：
$$y_n = \boldsymbol{W}_{aw}^{last} * \boldsymbol{x}_n^{last} + \boldsymbol{b}_{aw}^{last}$$

Schütt, K. T.; Sauceda, H. E.; Kindermans, P.-J.; Tkatchenko, A.; Müller, K.-R.; SchNet – A deep learning architecture for molecules and materials. *J. Chem. Phys.* **2018**, *148*, 241722.

PhysNet结构



- Residual Block:

$$\boldsymbol{x}_n^{l+2} = \boldsymbol{x}_n^l + \boldsymbol{W}_{res}^{l+1} * \sigma\left[\boldsymbol{W}_{res}^l * \sigma\left(\boldsymbol{x}_n^l\right) + \boldsymbol{b}_{res}^l\right] + \boldsymbol{b}_{res}^{l+1}$$

- Module Layer:

$$\boldsymbol{x}_n^{l+1} = \boldsymbol{u}_m \circ \boldsymbol{x}_n^l + \boldsymbol{W}_{m0}^l * \sigma\left(\boldsymbol{v}_n^l\right) + \boldsymbol{b}_{m0}^l$$

$$\boldsymbol{v}_n^l = \sigma\left[\boldsymbol{W}_{m1}^l * \sigma\left(\boldsymbol{x}_n^l\right) + \boldsymbol{b}_{m1}^l\right] + \sum_{k\neq n} \boldsymbol{G}_m * \boldsymbol{g}(d_{nk}) \circ \sigma\left[\boldsymbol{W}_{m2}^l * \sigma\left(\boldsymbol{x}_k^l\right) + \boldsymbol{b}_{m2}^l\right]$$

- Output block:

$$\boldsymbol{y}_n^m = \boldsymbol{W}_{\text{out}}^m * \sigma\left(\boldsymbol{x}_n^l\right) + \boldsymbol{b}_{\text{out}}^m$$

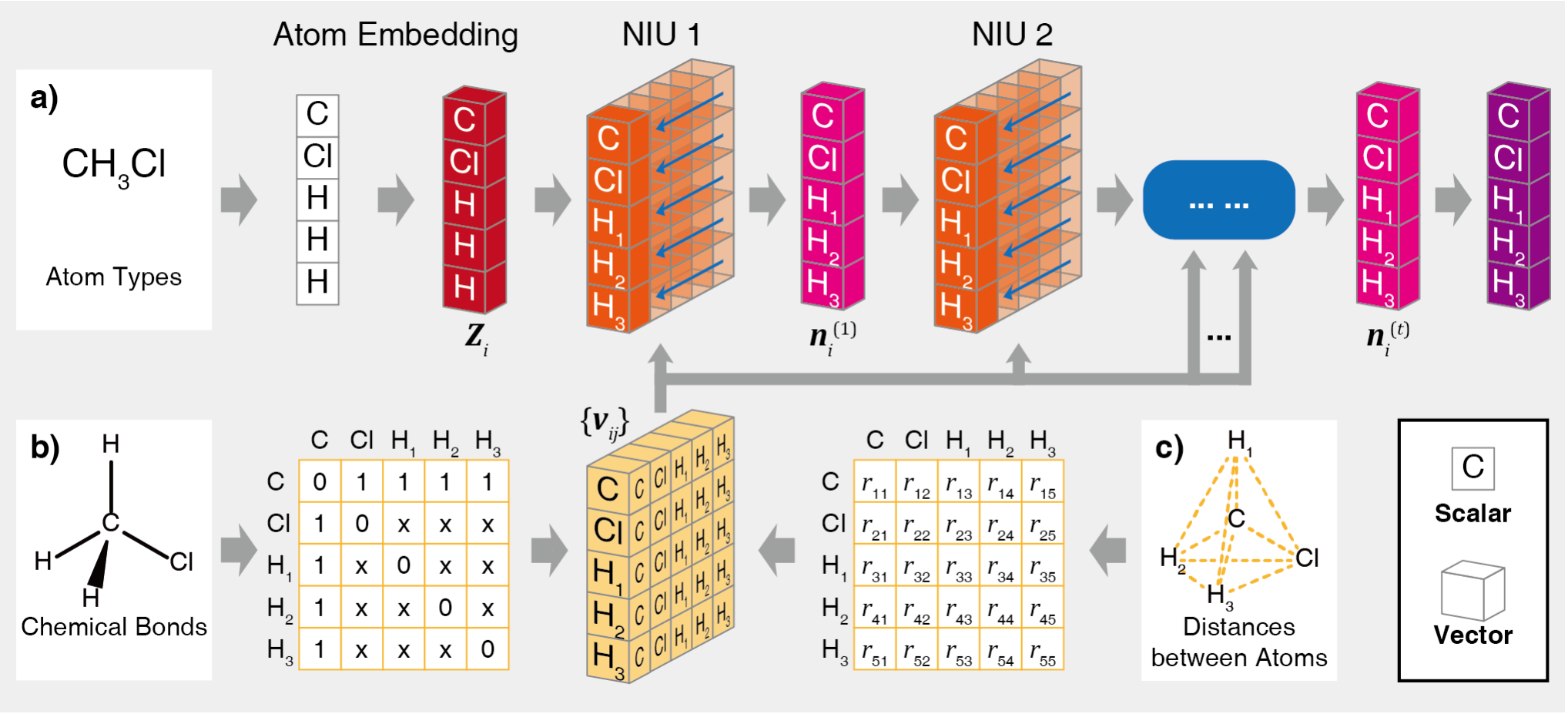
- Final Prediction:

$$\boldsymbol{y}_n^{\text{fin}} = \boldsymbol{s}_n \circ \sum_m^{N_{module}} \boldsymbol{y}_n^m + \boldsymbol{z}_n$$

Unke, O. T.; Meuwly, M.; PhysNet: A Neural Network for Predicting Energies, Forces, Dipole Moments, and Partial Charges. *J. Chem. Theory Comput.* **2019**, *15*, 3678.

深圳湾实验室
Shenzhen Bay Laboratory

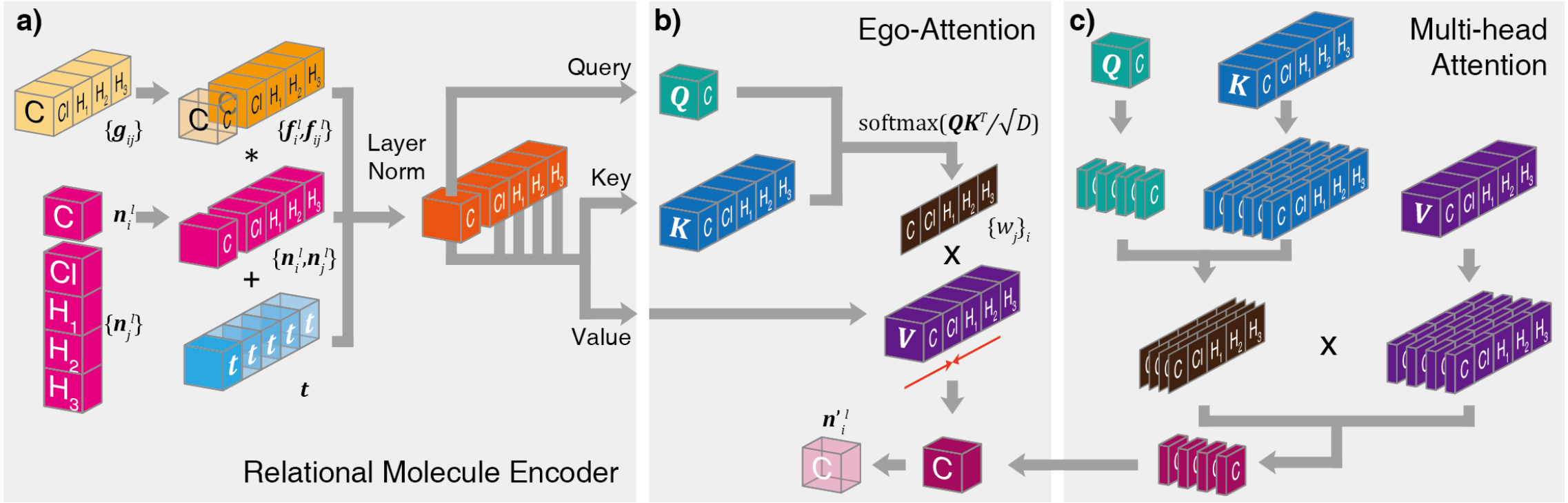
MolCT模型



Zhang, J.; Zhou, Y.; Lei, Y.-K.; Yang, Y. I.; Gao, Y. Q., *Molecular CT: Unifying Geometry and Representation Learning for Molecules at Different Scales*. arXiv:2012.11816 [cs.LG]

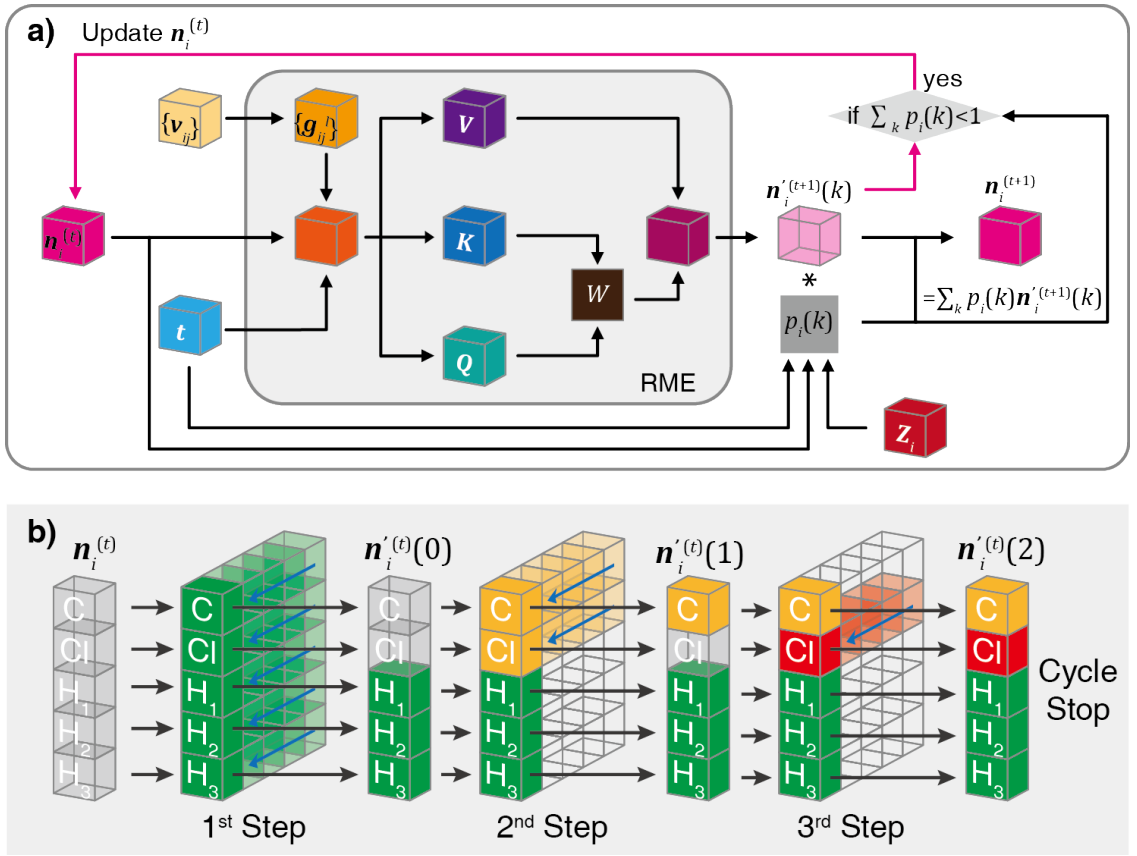
深圳湾实验室
Shenzhen Bay Laboratory

MolCT模型



深圳湾实验室
Shenzhen Bay Laboratory

Neural Interaction Unit



深圳湾实验室
Shenzhen Bay Laboratory

GNN深度分子模型的输入域输出

- GNN深度分子模型的输入是每个原子的类型Z_i和笛卡尔坐标$\boldsymbol{R}_i$，其中Z_i使用整型来区分不同类型的原子。
- 网络的原始输出是每个原子的“表示”（representation）$\boldsymbol{V}_i$，之后需要经过“Readout”函数得到最终的输出

$$\{\boldsymbol{R}_i; Z_i\} \rightarrow \text{Network} \rightarrow \{\boldsymbol{V}_i\} \rightarrow \text{Readout} \rightarrow E$$

- 总体而言有两种不同的Readout模式：
  - Atomwise Readout，适用于“所预测的物理量可以分配到每一个原子”的情况：

$$\{\boldsymbol{V}_i\} \rightarrow \text{Decode} \rightarrow \{E_i\} \rightarrow \text{Aggregate} \rightarrow E$$

  - Graph Readout 或 Set2Set Readout，适用于“预测的物理量属于整个分子的性质”的情况

$$\{\boldsymbol{V}_i\} \rightarrow \text{Aggregate} \rightarrow \boldsymbol{V}_{\text{agg}} \rightarrow \text{Decode} \rightarrow E$$

深圳湾实验室
Shenzhen Bay Laboratory

训练数据集的标准化与归一化及神经网络的尺度变换（Scale Shift）

- 由于神经网络的能够有效预测的数值范围有限，所以在使用神经网络的拟合相应数据的时候，需要对训练用的数据集的标签进行标准化或归一化操作：
  - 标准化（z-score）操作：指将训练数据集缩放为平均值为0，标准差为1的操作：

$$y' = \frac{y - \langle y \rangle}{\sigma}$$

  - 归一化（normalization）操作：指将训练数据集映射到$[0,1]$或$[-1,1]$区间：
    - 最大值最小值归一化：$y' = \frac{y - \min(y)}{\max(y) - \min(y)}$
    - 均值归一化：$y' = \frac{y - \langle y \rangle}{\max(y) - \min(y)}$
    - 最大绝对值归一化：：$y' = \frac{y}{|\max(|y|)|}$
  - 如果数据集大致满足高斯分布，则建议使用标准化操作，否则建议使用归一化操作
- 无论是标准化还是归一化操作，都是对数据集进行某种尺度变换（Scale Shift），即$y' = (y - b)/a$，那么在训练完成之后，使用模型进行推理时，则需要用同样的系数对网络模型进行尺度变换：

$$y = ay' + b = af(x) + b$$

深圳湾实验室
Shenzhen Bay Laboratory

神经网络输出的scale，shift和type_ref

- 一般神经网络的数值输出范围非常有限，但是需要预测数据的数据范围一般较大，所以需要对神经网络的输出进行scale和shift。
- 在训练之前通常需要对分析训练数据的数据范围，一般取训练数据的平均值作为shift，标准差作为scale
- 在使用深度分子模型对能量数据的进行训练时，如果训练数据集包含不同的分子，可以使用type_ref
- 所谓type_ref，是指每种元素的单个原子在真空中的能量基准值，这样在训练过程中就可以在基准数值的基础上加上网络的输出作为预测值：

深圳湾实验室 Shenzhen Bay Laboratory

损失函数

- 只有能量的损失函数：
  - MSE Loss：$L = \left(E^{\text{pred}} - E^{\text{label}}\right)^2$
  - MAE Loss：$L = \left|E^{\text{pred}} - E^{\text{label}}\right|$
- 包含力和能量的损失函数：
  - SchNet形式：$L = \rho\left(E^{\text{pred}} - E^{\text{label}}\right)^2 + \frac{1}{N_{\text{atoms}}}\sum_i^{N_{\text{atoms}}}\left(\boldsymbol{F}_i^{\text{pred}} - \boldsymbol{F}_i^{\text{label}}\right)^2$
  - PhysNet形式：$L = w_E\left|E^{\text{pred}} - E^{\text{label}}\right| + \frac{w_F}{3N_{\text{atoms}}}\sum_i^{N_{\text{atoms}}}\sum_j^3\left|F_{i,j}^{pred} - F_{i,j}^{\text{label}}\right|$
  - 新形式：

$$L = w_E\left(\frac{E^{\text{pred}}}{N_{\text{atoms}}} - \frac{E^{\text{label}}}{N_{\text{atoms}}}\right)^2 + \frac{w_F}{N_{\text{atoms}}}\sum_i^{N_{\text{atoms}}}\left(d_F\boldsymbol{F}_i^{\text{pred}} - d_F\boldsymbol{F}_i^{\text{label}}\right)^2$$

$$d_F = \frac{\sigma\left(E^{\text{label}}\right)}{N_{\text{atoms}}\left|\text{mod}\left(F^{\text{label}}\right)\right|}$$

深圳湾实验室
Shenzhen Bay Laboratory

MindSpore基本设置

- 两种模式：
  - Graph模式：静态图
  - PyNative模式：动态图
- Tensor类型：
  - `t = Tensor(numpy.ndarray,mindspore.dtype)`
- Dataset数据集格式：
  - `ds = dataset.NumpySlicesDataset({'R':numpy_data['R'],'z':numpy_data['z'],'E':numpy_data['E']},shuffle=True)`
- 设置batch_size:
  - `ds = ds.batch(batch_size,drop_remainder=True)`
- 设置优化器：
  - `optim = nn.Adam(params=net.trainable_params(),learning_rate=1e-3)`

- 设置网络模型：
  - `mod = SchNet(dim_feature=128)`
- 设置Readout:
  - `readout = AtomwiseReadout(n_in=128,n_out=1,activation='swish')`
- 设置Cybertron：
  - `net = Cybertron(mod, readout=readout)`
- 设置loss：
  - `loss_network = WithLabelLossCell('RZE',net,nn.MAELoss())`
- 设置训练模型：
  - `model = Model(loss_network,optimizer=optim)`
- 开始训练：
  - `model.train(n_epoch,ds_train)`

深圳湾实验室
Shenzhen Bay Laboratory

不同的数据集

- 数据集：
  - 训练集（Train set）：用于训练模型
  - 验证集（Validation set）：用于监测训练过程的进度
  - 测试集（Test set）：用于检验模型训练后的效果
- 在Cybertron中使用验证集：
  1. 设置验证集：

```
ds_valid = ds.NumpySlicesDataset({'R':valid_data['R'],'z':valid_data['z'],'E':valid_dat
a['E'][:,idx]},shuffle=False)
```

  2. 在model中设置验证参数：

```
model = Model(loss_network,optimizer=optim,eval_network=eval_network,metrics=
{eval_mae:MAE([1,2]),eval_loss:MLoss(0)})
```

  3. 设置TrainMoniter：

```
record_cb = TrainMonitor(model, outname, eval_dataset=ds_valid)
```

  4. 开始训练：

```
model.train(n_epoch,ds_train,callbacks=[record_cb])
```

深圳湾实验室
Shenzhen Bay Laboratory

Cybertron参数

- 输入参数：
  - positions `(mindspore.Tensor[float], [B, A, D])  # R`
  - atom_types `(mindspore.Tensor[int], [B, A])  # Z`
  - pbc_box `(mindspore.Tensor[float], [B, D])  # C`
  - distances `(mindspore.Tensor[float], [B, A, N])  # D`
  - neighbors `(mindspore.Tensor[int], [B, A, N])  # N`
  - neighbor_mask `(mindspore.Tensor[bool], [B, A, N])  # n`
  - bonds `(mindspore.Tensor[int], [B, A, N])  # B`
  - bond_mask `(mindspore.Tensor[bool], [B, A, N])  # b`
- 初始化参数：
  - **atom_types**：如果在初始化时赋值，则在计算过程中使用这一数值作为固定的**atom_types**
  - **num_atoms**：如果在初始化时不给出**atom_types**，则需要设定（最大的）原子的数目

深圳湾实验室
Shenzhen Bay Laboratory

Tutorial 0: 数据预处理

- 要点：
  1. 分子体系数据集的内容
  2. 数据的分析
  3. 数据集的输出
  4. Atomwsie scaleshift与Graph scaleshift的区别

深圳湾实验室
Shenzhen Bay Laboratory

Tutorial 1: 基本操作（一）

- 要点：
  1. Cybertron初始化
  2. 训练数据集的导入
  3. 损失函数的设置
  4. 优化器和学习率
  5. MindSpore训练模型的设置
  6. 基本的callback: LossMonitor()和ModelCheckpoint()
  7. 使用Model.train进行训练

深圳湾实验室
Shenzhen Bay Laboratory

Tutorial 2: 基本操作（二）

- 要点：
  1. Model与Readout的设置
  2. 预测以能量为代表的广度量一般需要使用AtomwiseReadout
  3. Atomwsie scaleshift与Graph scaleshift的区别

深圳湾实验室
Shenzhen Bay Laboratory

Tutorial 3: 归一化数据集与验证数据集的使用

- 要点：
  1. 不同数据集的用法与设置
  2. 使用归一化数据集进行训练不需要对输出本身进行Scale和Shift
  3. 在mindspore.Model中设置metrics，从而在训练过程输出需要的中间量
  4. 使用TrainMonitor保存训练过程中重要的数据
  5. 使用Model.eval检验模型
  6. 可以为EvalCell设置Scale和Shift从而获得正确的验证结果

深圳湾实验室
Shenzhen Bay Laboratory

Tutorial 4：参数与超参数的读取（一）

- 要点：
  1. 使用load_hyperparam读取超参数
  2. 根据超参数初始化Cybertron
  3. 使用load_checkpoint读取模型参数
  4. 在TrainMonitor中设置best_ckpt_metrics以保存训练过程中的最佳参数
  5. 如果不需要继续训练，只需用load_checkpoint将模型读入模型即可进行模型检验
  6. 如果需要继续训练，需要对优化器使用load_param_int_net

深圳湾实验室
Shenzhen Bay Laboratory

Tutorial 5：多任务训练（一）

- 要点：
  1. 使用白化的数据集，readout本身不能设置scale和shift
  2. WithLabelLossCell可以自动进行白化（do_scaleshift），但需要设置scale和shift
  3. 如果WithLabelLossCell使用自动白化，WithLabelEvalCell则需要开启自动缩放（do_scaleshift=Ture），并使用相同的scale和shift

深圳湾实验室
Shenzhen Bay Laboratory

Tutorial 6：多任务训练（二）

- 要点：
  1. 如果readout相同，可以通过设置dim_output的方法实现多任务训练

深圳湾实验室 Shenzhen Bay Laboratory

Tutorial 7：对力进行拟合

- 要点：
  1. 当对原子种类和顺序不变的，可以在Cybertron模型声明时提前输入原子类型（Z），这样在训练过程中就不必输入原子类型
  2. 损失函数使用WithForceLossCell，需分别为能量和力设置损失函数的形式
  3. 如果同时对能量和力进行拟合，需要为能量和力相应的权重
  4. 传统的损失函数存在的最大问题是能量和力的单位不统一：
     - SchNet形式： $L = \rho\left(E^{\text{pred}} - E^{\text{label}}\right)^2 + \frac{1}{N_{\text{atoms}}}\sum_i^{N_{\text{atoms}}}\left(\boldsymbol{F}_i^{\text{pred}} - \boldsymbol{F}_i^{\text{label}}\right)^2$
     - PhysNet形式： $L = w_E\left|E^{\text{pred}} - E^{\text{label}}\right| + \frac{w_F}{3N_{\text{atoms}}}\sum_i^{N_{\text{atoms}}}\sum_j^3\left|F_{i,j}^{pred} - F_{i,j}^{\text{label}}\right|$

     这里为了消除尺寸单位不同给力的数据带来的影响，引入力的平均单位位移：

$$\langle s_F \rangle = \frac{\langle E \rangle}{\langle |F| \rangle} = \frac{\langle E \rangle}{\langle |\partial E / \partial \boldsymbol{R}| \rangle}$$

     则新的损失函数为：

$$L = w_E\left(E^{\text{pred}} - E^{\text{label}}\right)^2 + \frac{w_F}{N_{\text{atoms}}}\sum_i^{N_{\text{atoms}}}\left[\langle s_F \rangle\left(\boldsymbol{F}_i^{\text{pred}} - \boldsymbol{F}_i^{\text{label}}\right)\right]^2$$

深圳湾实验室
Shenzhen Bay Laboratory

Tutorial 8：参数与超参数的读取（二）

- 要点：
  1. 参见tutorial 4