pytorch编程基础与药物发现实战

- 人工智能用于药物发现

目录

pytorch基础知识

- torch基础
- dataset、dataloader、transform
- training loop

pytorch进阶知识

- 集合通信与torch distributed
- 数据并行

毒性预测

- representation of a molecule
- tox21
- training a Neural Network

分子生成

- 分子生成概述
- 单机多卡训练分子GPT

显卡与深度学习

Forward pass:
Compute output

```
def f(w0, x0, w1, x1, w2):
  s0 = w0 * x0
  s1 = w1 * x1
  s2 = s0 + s1
  s3 = s2 + w2
  L = sigmoid(s3)
```

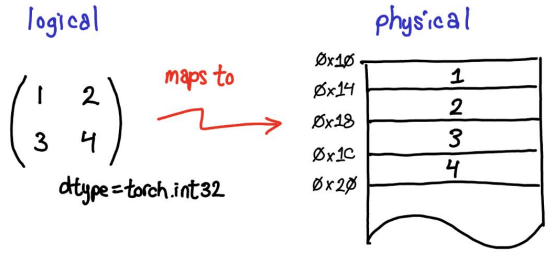
Backward pass:
Compute grads

```
grad_L = 1.0
grad_s3 = grad_L * (1 - L) * L
grad_w2 = grad_s3
grad_s2 = grad_s3
grad_s0 = grad_s2
grad_s1 = grad_s2
grad_w1 = grad_s1 * x1
grad_x1 = grad_s1 * w1
grad_w0 = grad_s0 * x0
grad_x0 = grad_s0 * w0
```

Pytorch基础

Tensor

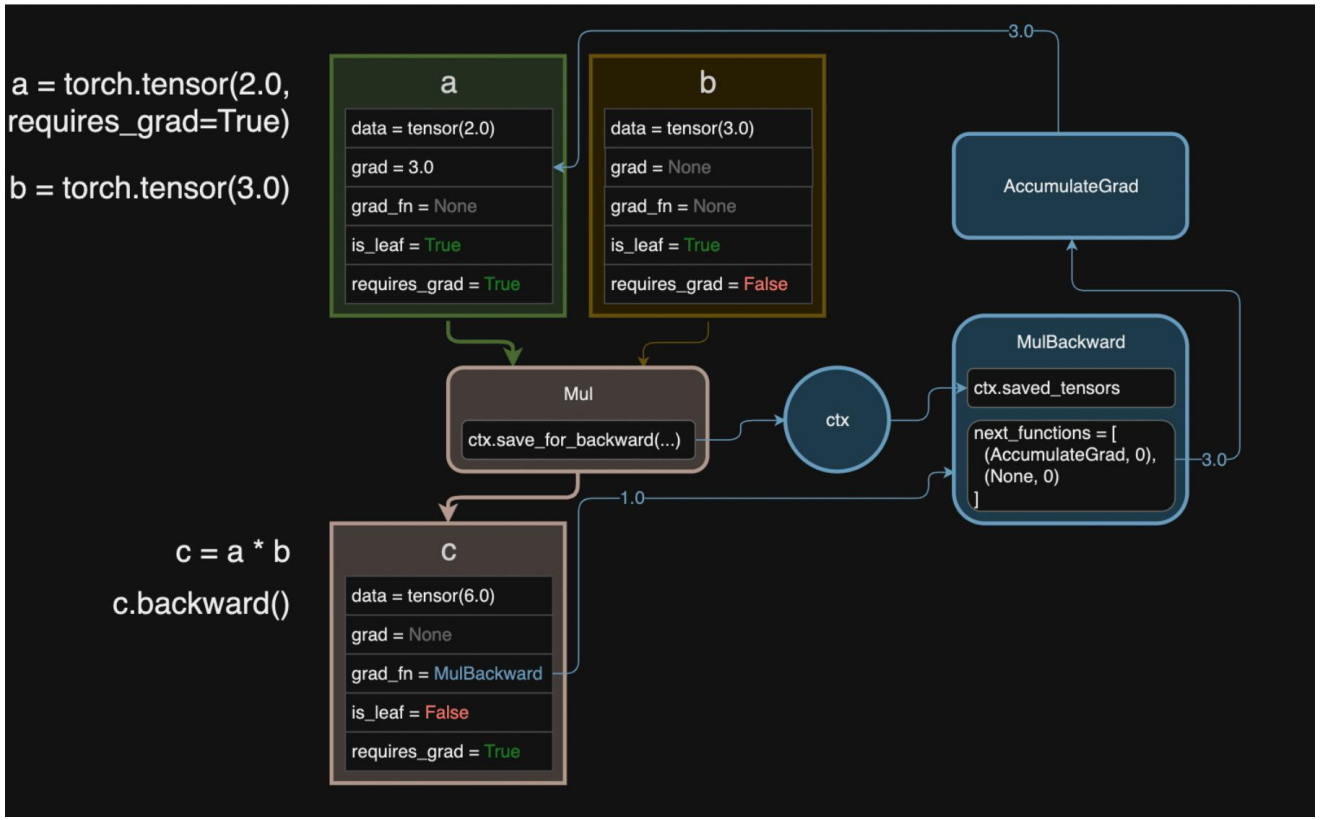
sizes: (2,2)
strides:(2,1)
dtype: int
device: cuda
layout: strided, sparse



Tensor Operations:
torch.mm(x,y)
cpu - xla - cuda etc
float16 bfloat16 float32 int32 etc

```
preds = Net(input_data)
loss = CrossEntropy(preds, labels)
loss.backward()
optimizer.step()
```

Autograd



Pytorch Training Loop

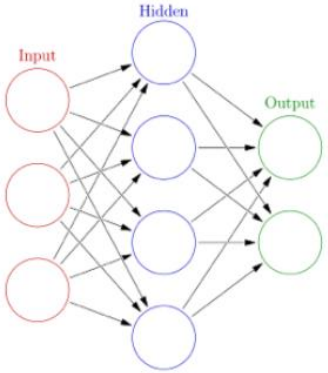
datasets & dataloaders

- *dataset*: 数据怎么从原有介质被读取到内存中作为一个*sample*
- *dataloader*: 将*sample collate*(装钉、整理) 成*batch*

```
preds = Net(input_data)
loss = CrossEntropy(preds, labels)
loss.backward()
optimizer.step()
```

ML algorithms learns a function

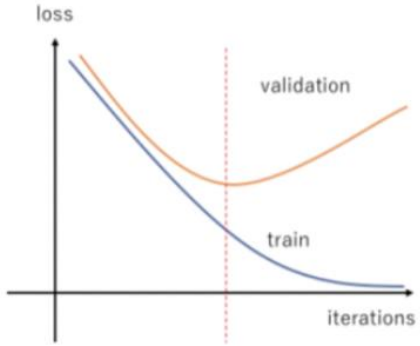
Define function with unkown parameters



Find an objective

$$RMSE = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(h(x^{(i)}) - y^{(i)})^2}$$

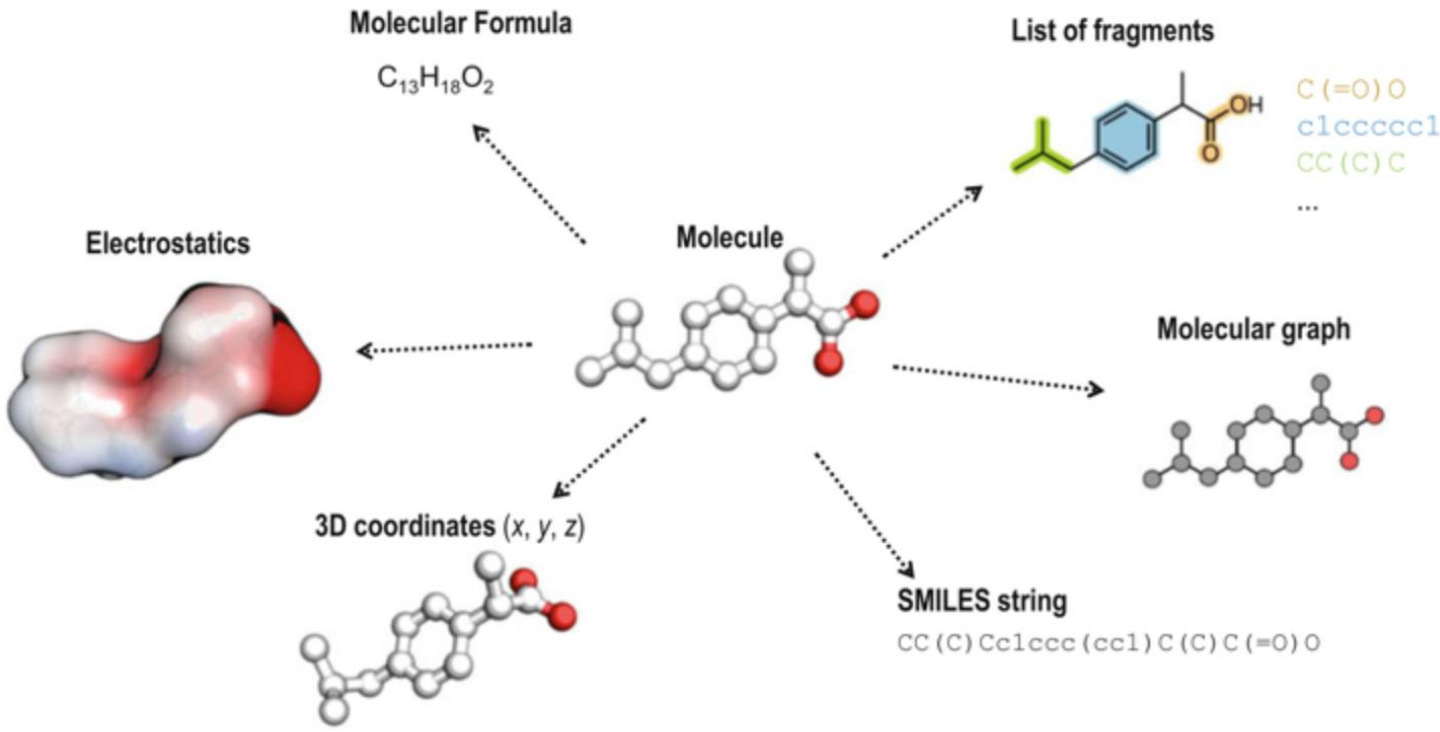
Find a way to optimize it



Successful ones need a large clean diverse dataset

分子表征和特征工程

字符	原子
单词	官能团
组合词	分子砌块
句子	药效团
段落	分子
文章	药物



- 1D String
- 2D Graph
- 3D Geometry
- 3D mesh
- Chemical Images
- Descriptors and Fingerprints

分子表征和特征工程-例子-pharmacophores

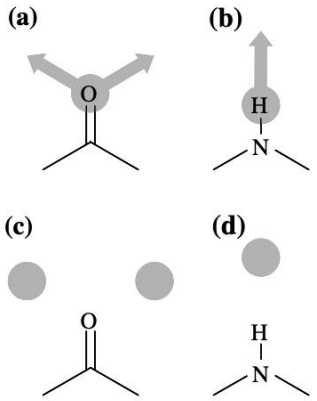


Fig. 2 Hydrogen bond acceptor and donor mappings based on the use of vector features (**a**, **b**) and pure projected points (**c**, **d**)

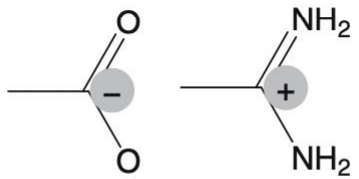


Fig. 4 Negative ionizable and positive ionizable feature mappings

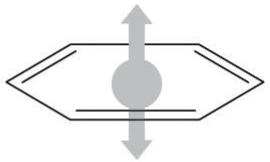
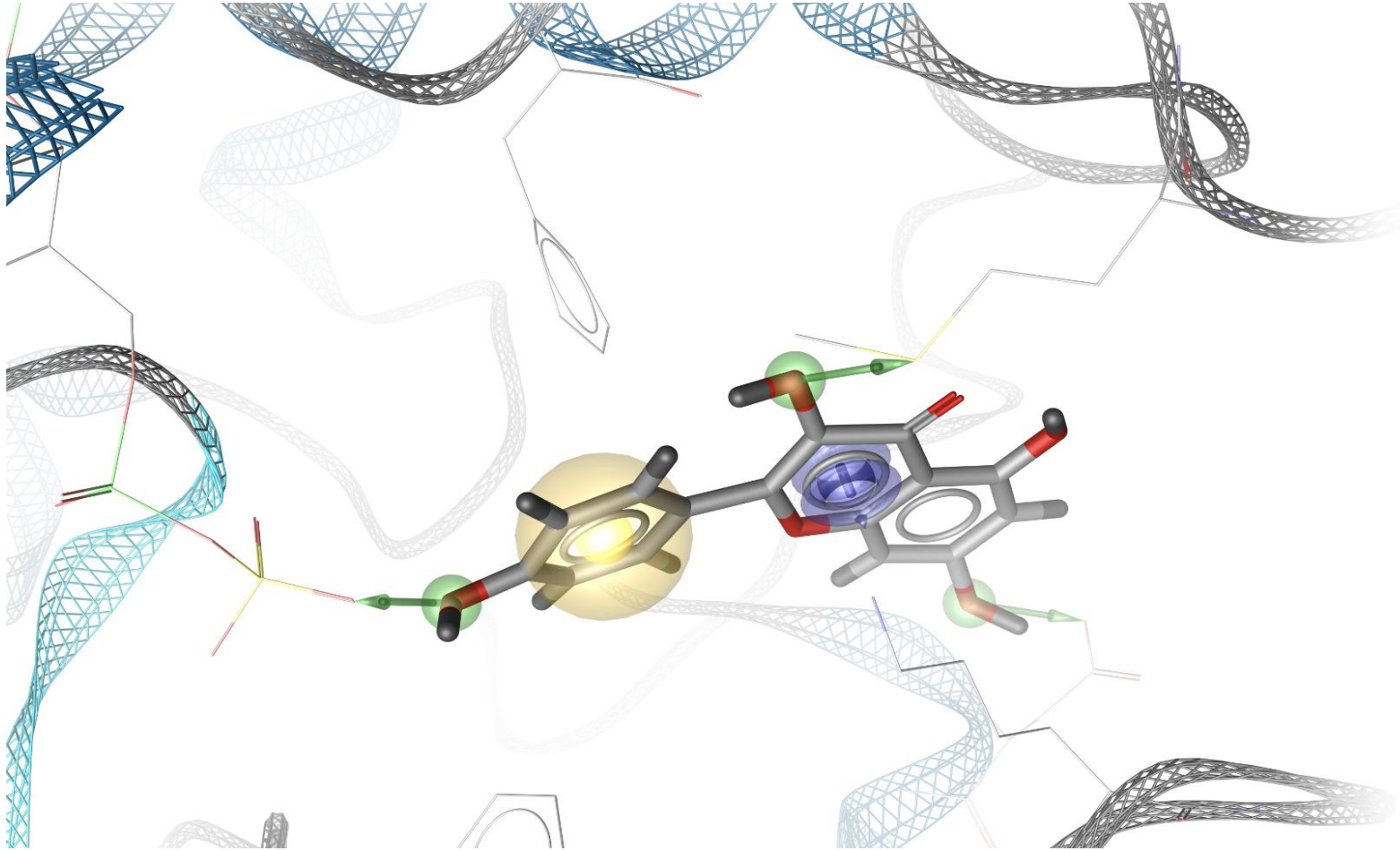


Fig. 5 Aromatic ring feature mapping

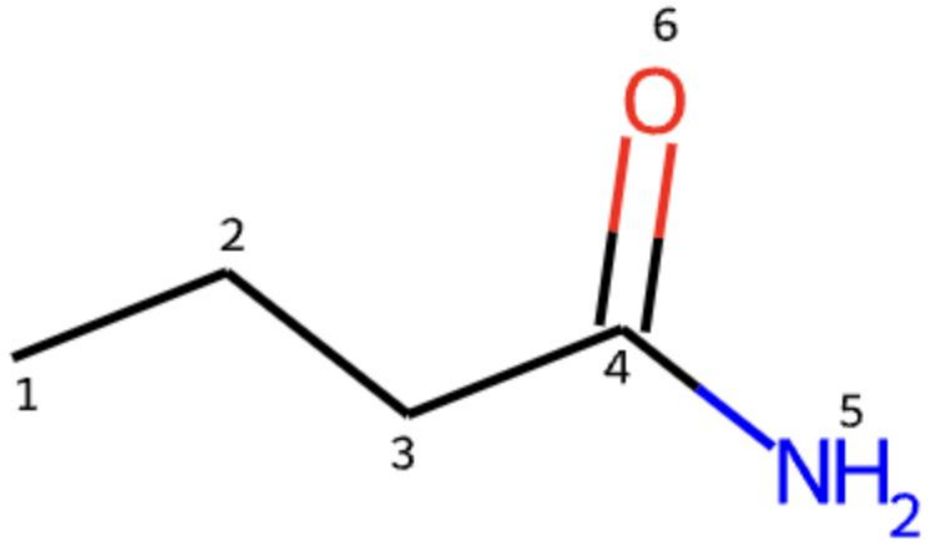


分子表征和特征工程-例子-分子指纹

Morgan-fingerprint

- Number of non-hydrogen immediate neighbors = 3 (atom 3, 5, and 6)
- Valency minus the number of connected hydrogens = 4 (4 - 0)
- Atomic number = 6
- Atomic mass = 12
- Atomic charge = 0
- Number of attached hydrogens = 0
- Is it a part of a ring = 0 (no)

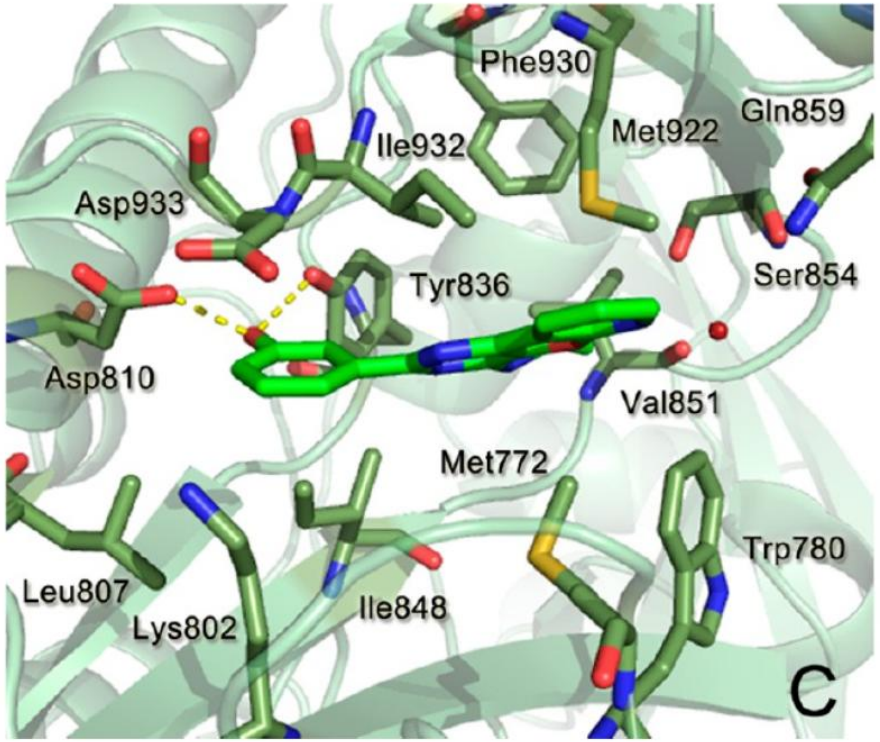
```
identifier=hash((3,4,6,12,0,0,0))print(identifier)# -2155244659601281804
```

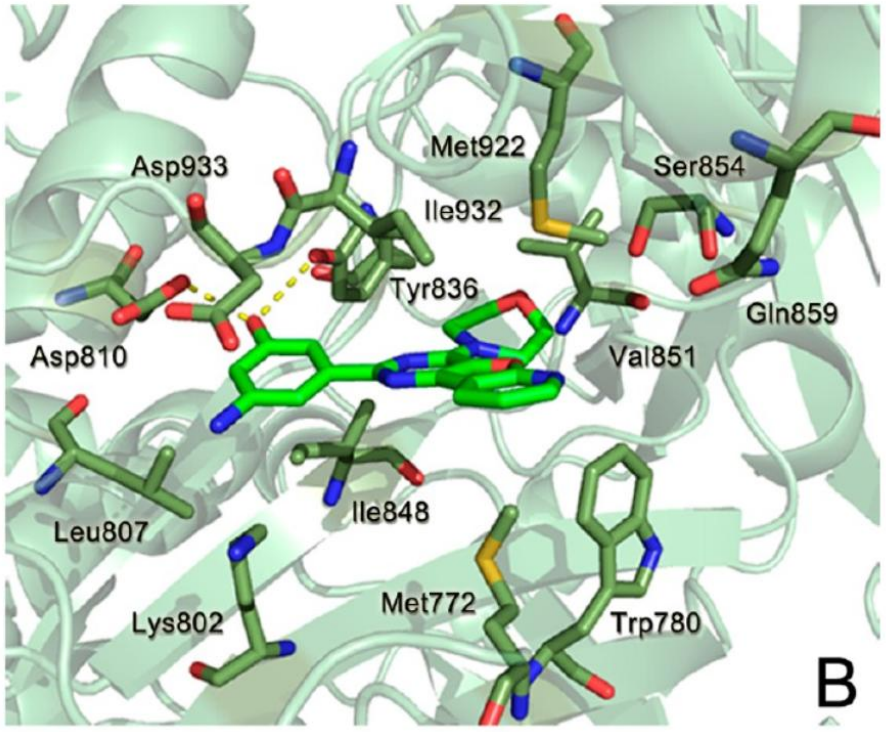


https://rdkit.org/docs/Cookbook.html

什么是药物？

活性 与 安全性





分子属性预测-ADMET

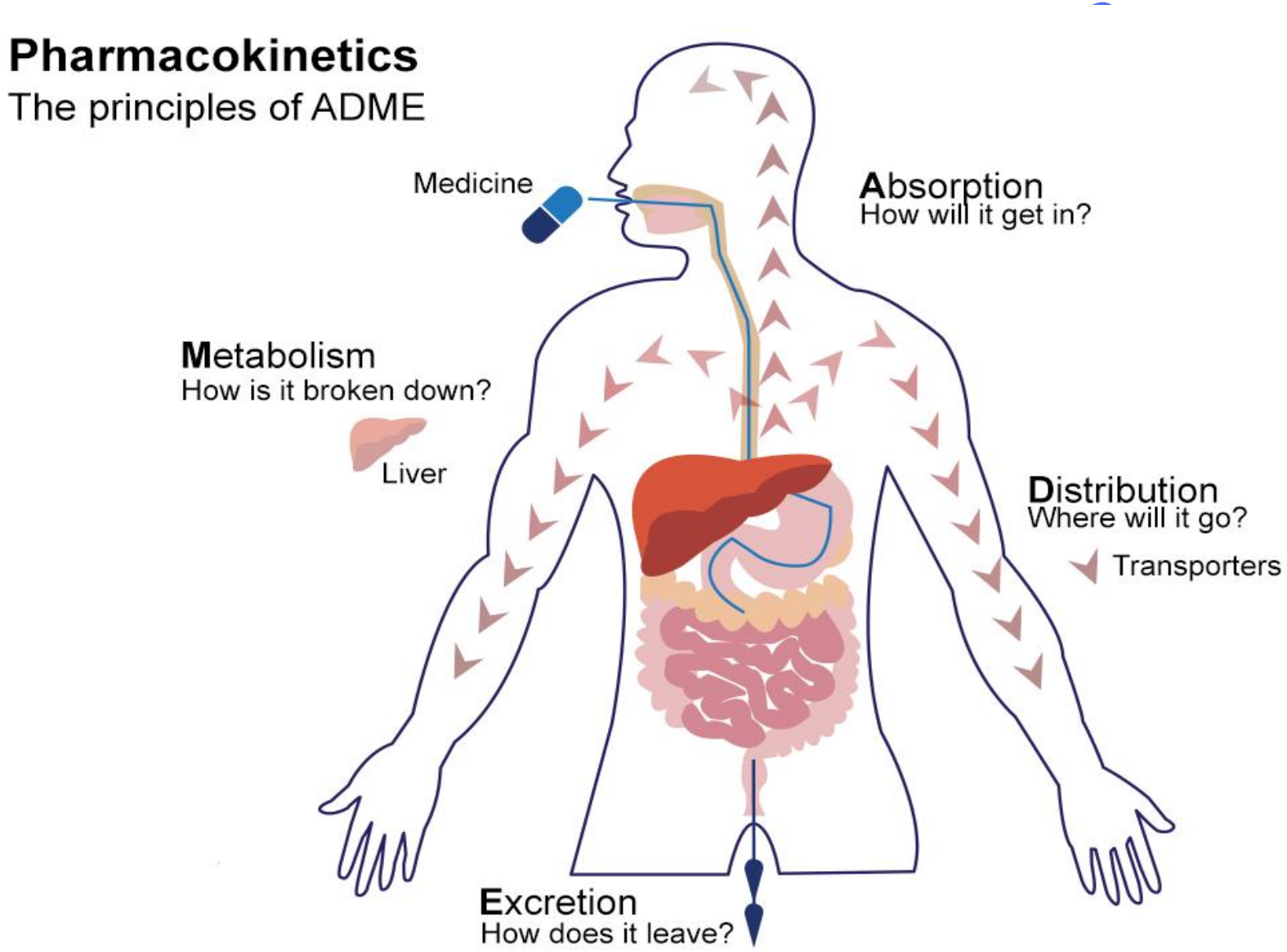
A：吸收
D：分布
M：代谢
E：排泄
T：毒性

Nuclear Receptor Signaling Panel

Assay	SDF	SMILES
AR		
AhR		
AR-LBD		
ER		
ER-LBD		
aromatase		
PPAR-gamma		

Stress Response Panel

Assay	SDF	SMILES
ARE		
ATAD5		
HSE		
MMP		
p53		



https://tripod.nih.gov/tox21/challenge/

实战I-深度学习进行分子属性预测

思考

数据的限制：
1.内插与外推：目前的深度学习模型是内插能力强，所以需要大量的数据（data-hungry）
2.足量的数据不现实，深度学习的起点是imagenet

模型的限制：
1.从第一性原理来讲，到底有用什么去表示？
2. 3D等变性与模型表征能力的矛盾
3. foundation model学习的表征也许是个promise的方向但是无监督的目标需要突破

Pytorch分布式训练

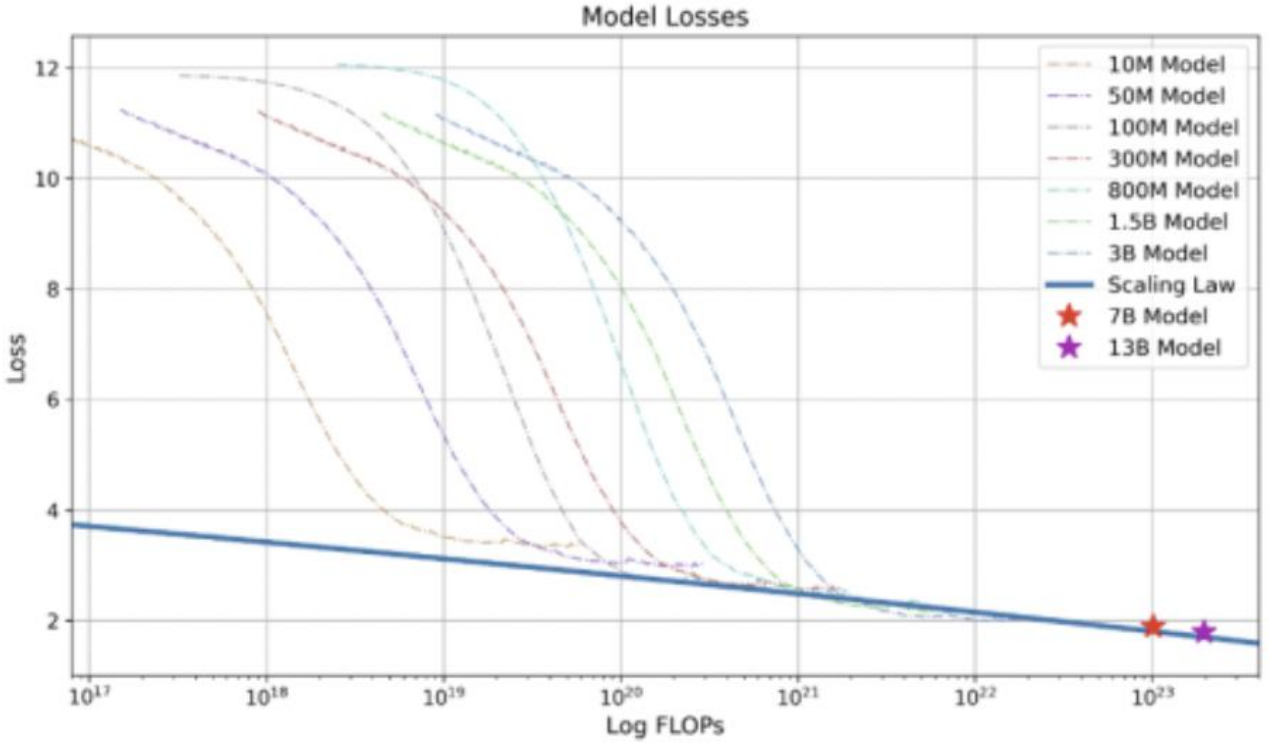
The ***bitter lesson*** is based on the historical observations that

1) AI researchers have often tried to build knowledge into their agents,

2) this always helps in the short term, and is personally satisfying to the researcher, but

3) in the long run it plateaus and even inhibits further progress, and

4) breakthrough progress eventually arrives by an opposing approach based on scaling computation by search and learning.



http://www.incompleteideas.net/IncIdeas/BitterLesson.html

大人，时代变了。。。

性能参数	V100 PCIe	A100 80GB PCIe	A800 80GB PCIe	H100 80GB PCIe
微架构	Volta	Ampere		Hopper
FP64	7TFLOPS	9.7TFLOPS		26 TFLOPS
FP32	14TFLOPS	19.5TFLOPS		51 TFLOPS
FP16 Tensor Core		312TFLOPS		756.5 TFLOPS
INT8 Tensor Core	62 TOPS	624 TOPS		1513 TOPS
GPU 显存	32/16GB HBM2	80GB HBM2e		80GB
GPU 显存带宽	900 GB/s	1935GB/s		2TB/s
最大热设计功耗 (TDP)	250 瓦	300 瓦		300-350W
多实例 GPU		最多 7 个 MIG 每个 10GB		
外形规格		PCIe 双插槽风冷式 或单插槽液冷式		PCIe 双插槽风冷式
互连技术	NVLink：300 GB/s PCIE：32 GB/s	搭载 2 个 GPU 的 NVIDIA” NVLink” 桥接器：600GB/s PCIe 4.0：64GB/s	搭载 2 个 GPU 的 NVIDIA” NVLink” 桥 接器：400GB/s PCIe 4.0：64GB/s	NVLink：600GB/s PCIe 5.0：128GB/s

集合通信

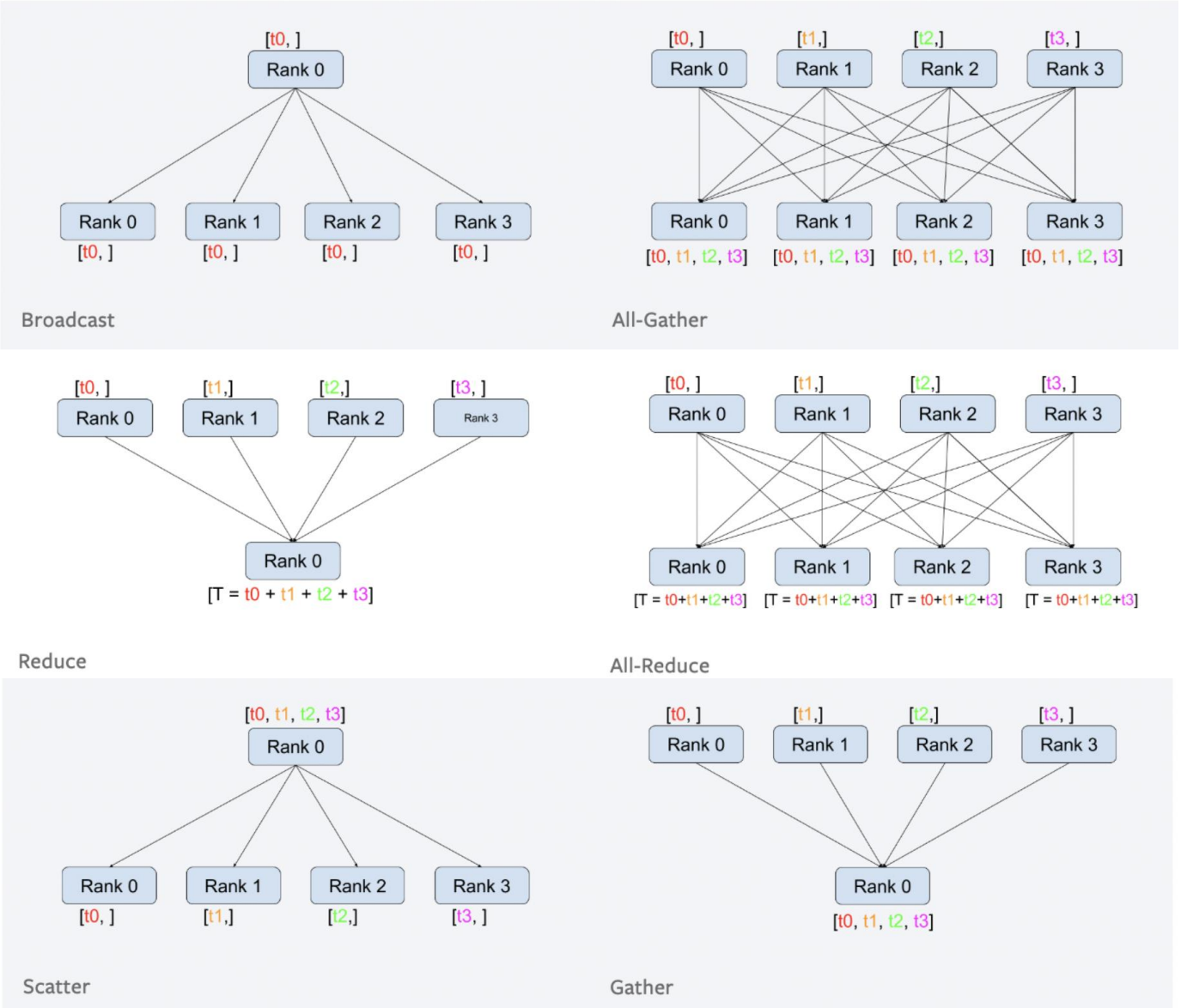
process：
集合通信视角的

rank：
进程对应的序号

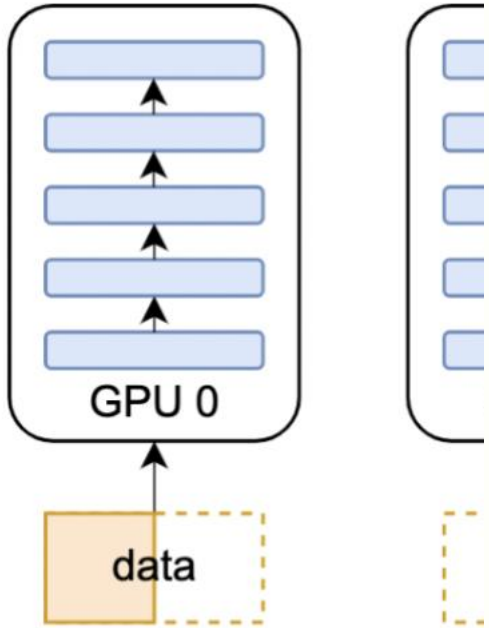
process group：
进程组，用于更细粒度的操作参与集合通信的进程

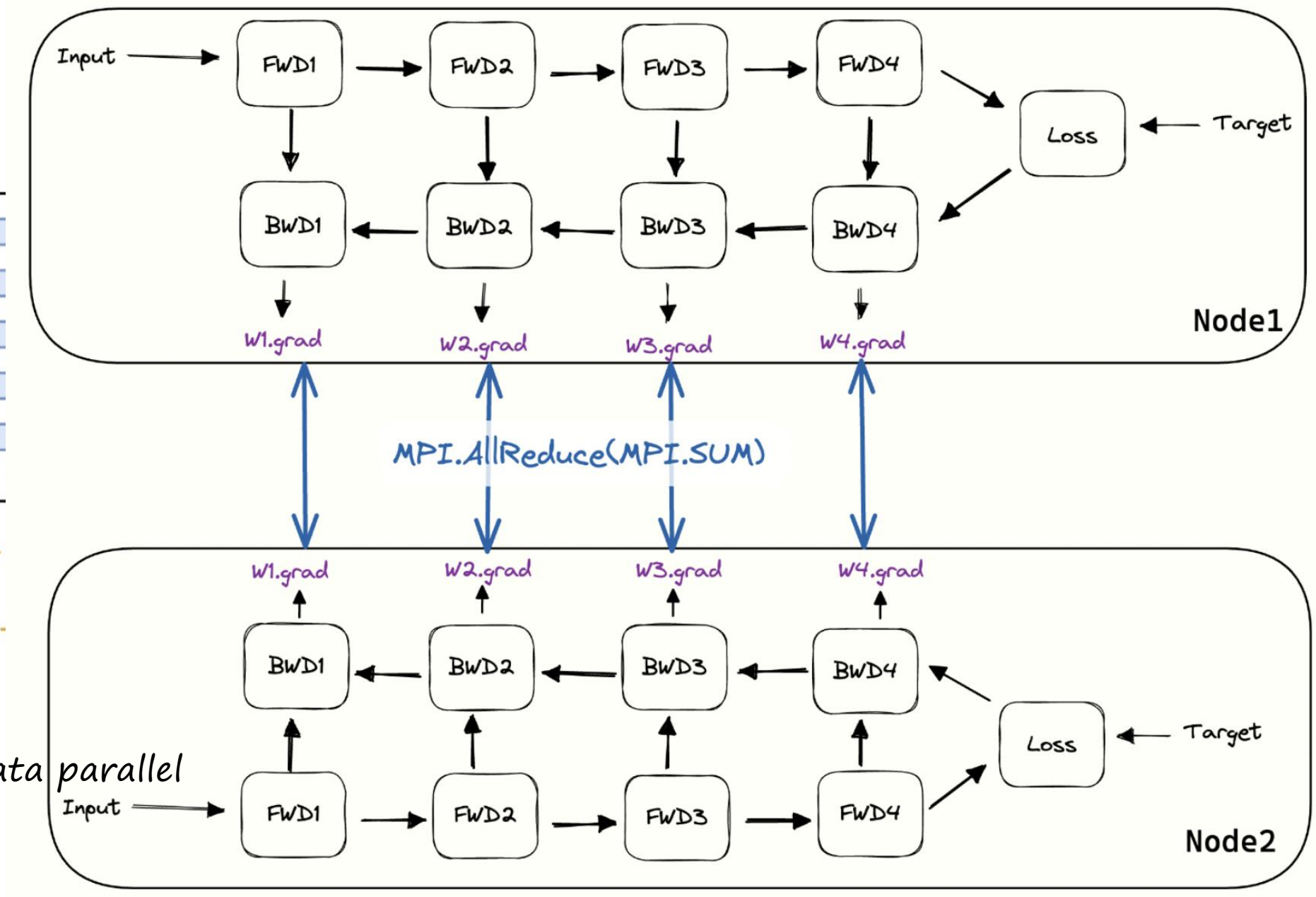
算子：集合通信会抽象成几个算子完成大部分的通信任务

集合通信库： 集合通信的不同实现，gloo、nccl 、hccl 、accl



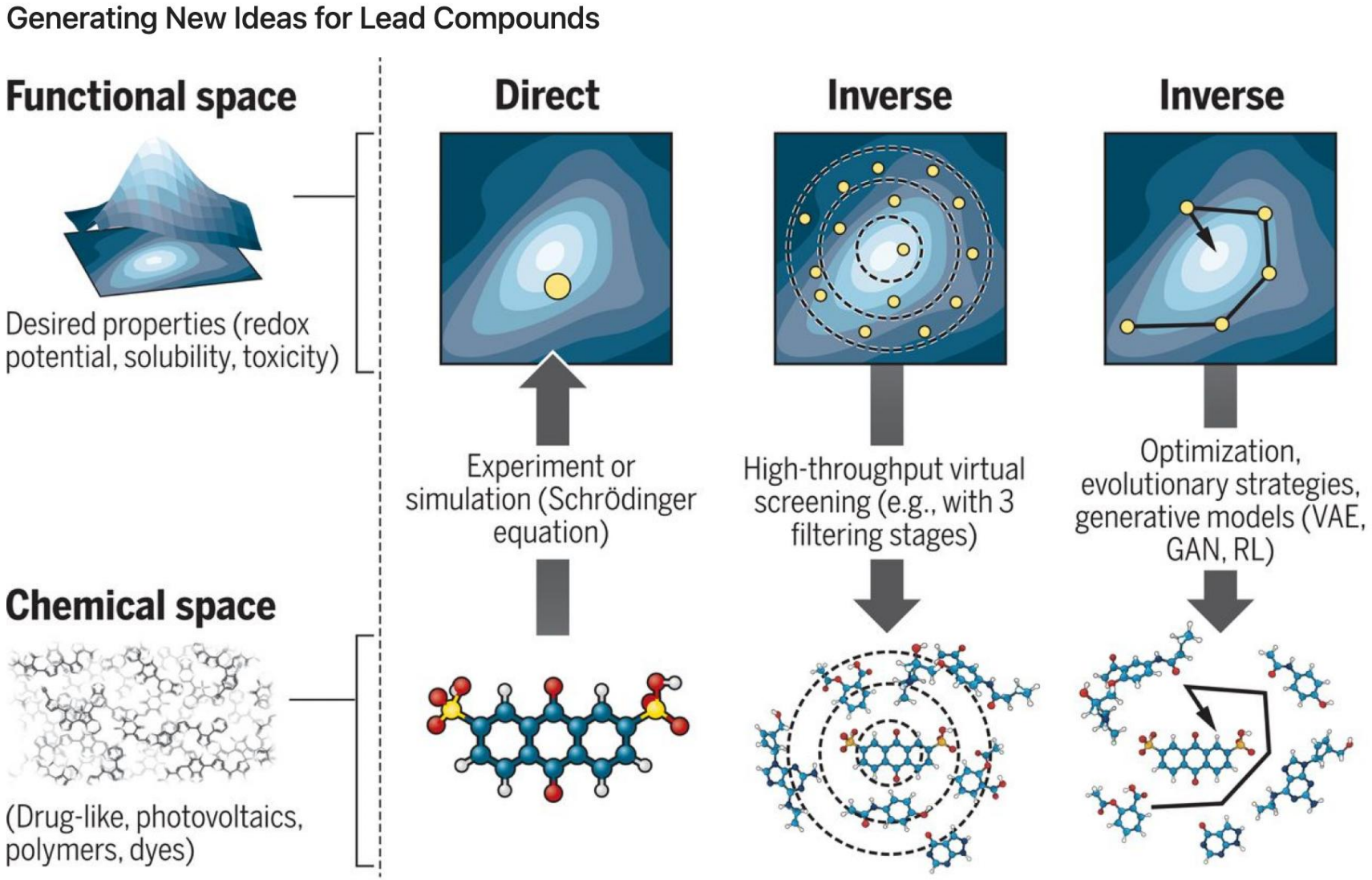
数据并行





torch distribued data parallel

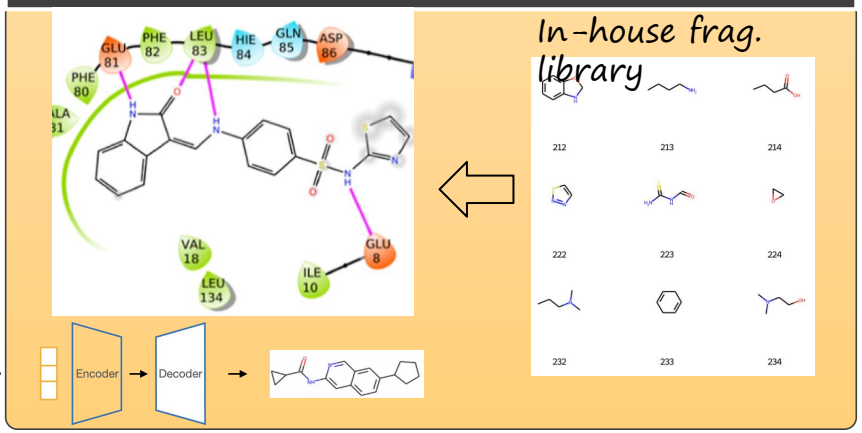
分子生成motivation



de novo scaffold construction and design

Generate diverse lead candidates directed by discovered binding pattern

interaction directed molecular generation



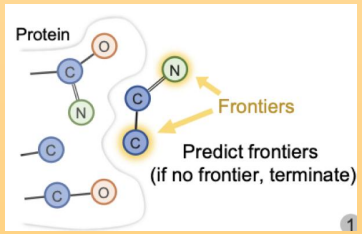
In-house frag. library

tractable chemical starting point

Explore vast chemical space
Discover drug-like hits

drug likeness directed molecular generation

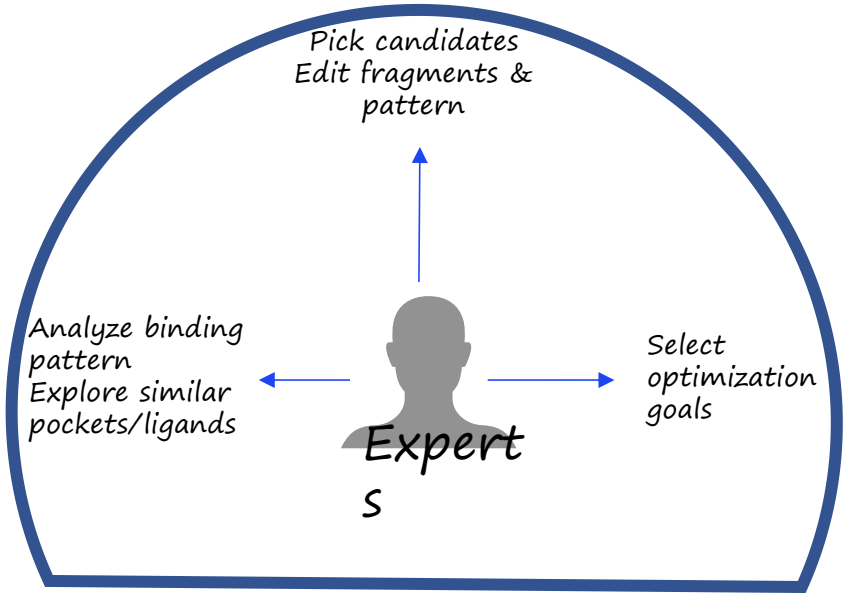
virtual screening



Evaluators

- docking
- similarity
- rule filters
- ...

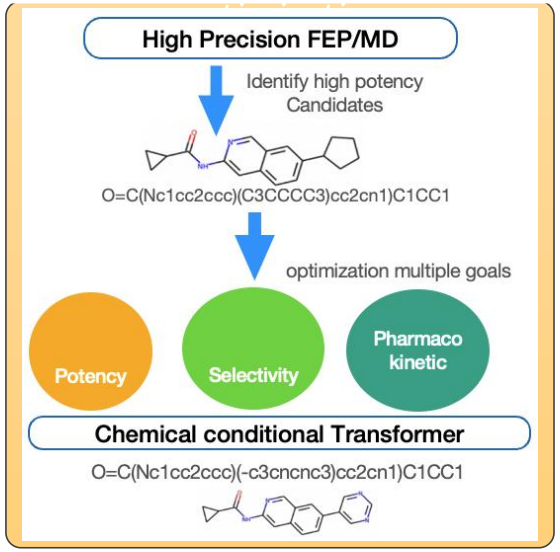
Feedbacks



Lead Optimization

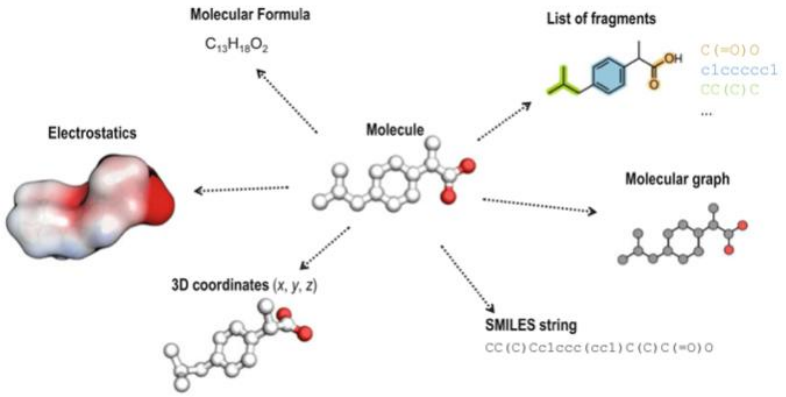
Multi-goal directed lead optimization

Goal directed molecular



分子生成概览

Representation



机器如何表示一个分子?

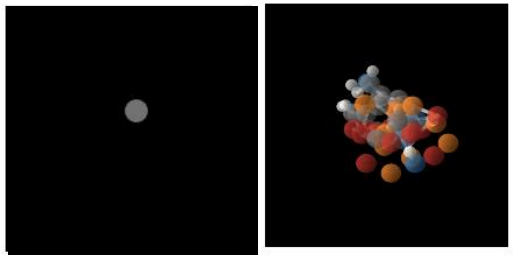
- 1D String
- 2D Graph
- 3D Geometry
- 3D mesh
- Chemical Images
- Descriptors and Fingerprints

Algorithm

Deep Generative Model

1) autoregressive
2) VAE
3) normalizing flow
4) diffusion
5) score-based
6) energy-based

etc

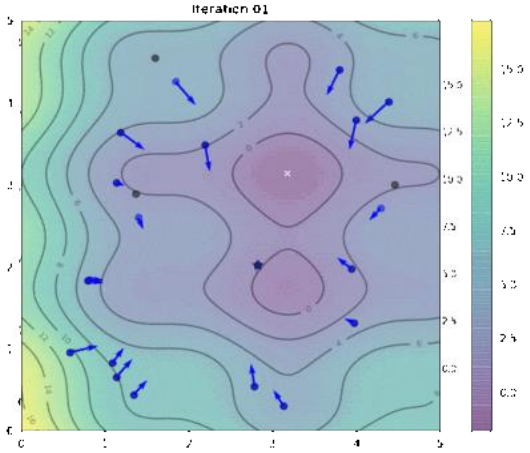


Auto-regressive Diffusion

Combinatorial Optimization Methods

1) Evolutionary algorithm
2) reinforcement learning

etc



Metric

- **Generation Metric**
  - Validity
  - Uniqueness
  - Novelty
  - Diversity
- **Distribution Metric**
  - FCD
  - Fragment/Tanimoto/Scaffold Similarity
- **Optimization Goals**
  - QED
  - LogP
  - Affinity
  - GSK3β \ JNK3 \ DRD2
- **3D conformer Evaluation**
  - RMSD
  - Coverage&Matching

Autoregressive

estimate p_{data} from $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(n)} \sim p_{data}(\mathbf{x})$
Now we introduce function approximation: learn θ so that $p_\theta(x) \approx p_{data}(x)$

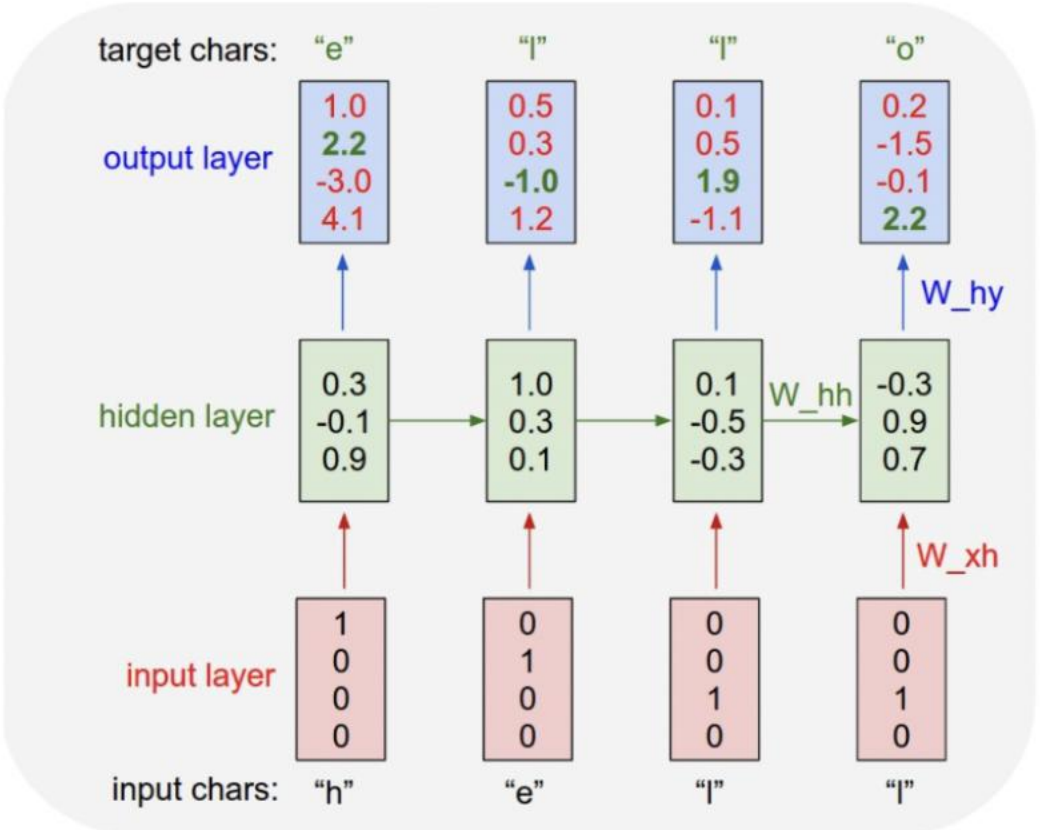
Maximum likelihood: given a dataset $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(n)}$, find θ by solving the optimization problem

$$\arg\min_\theta \ \text{loss}(\theta, \mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(n)}) = \frac{1}{n}\sum_{i=1}^{n} -\log p_\theta(\mathbf{x}^{(i)})$$

$$\text{for all } \theta, \quad \sum_{\mathbf{x}} p_\theta(\mathbf{x}) = 1 \quad \text{and} \quad p_\theta(\mathbf{x}) \geq 0 \quad \text{for all } \mathbf{x}$$

$$\mathrm{P}(A_n \cap \ldots \cap A_1) = \mathrm{P}(A_n | A_{n-1} \cap \ldots \cap A_1) \cdot \mathrm{P}(A_{n-1} \cap \ldots \cap A_1)$$

$$\log p_\theta(\mathbf{x}) = \sum_{i=1}^{d} \log p_\theta(x_i \mid \text{parents}(x_i))$$



https://github.com/pengxingang/Pocket2Mol

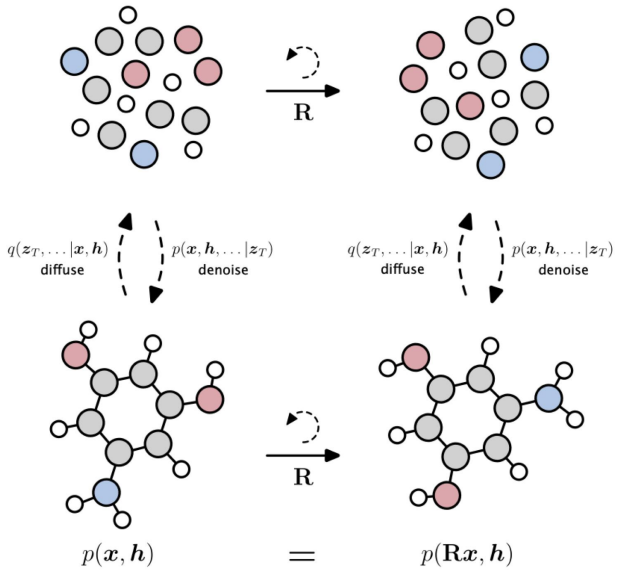
Diffusion

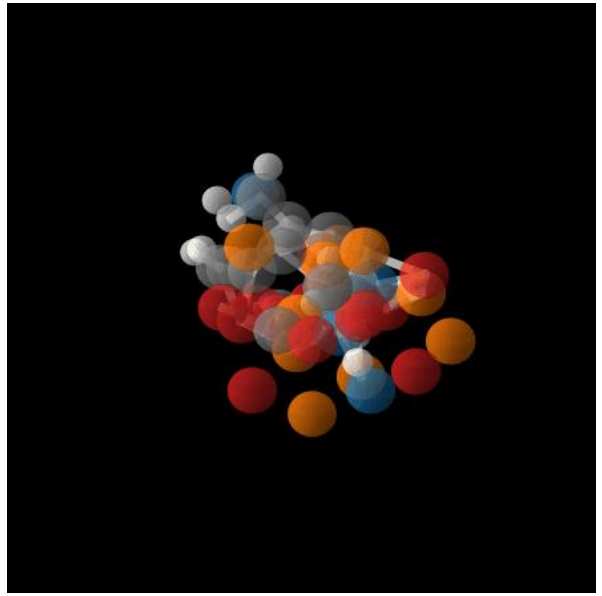
$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x_t}; \sqrt{1-\beta_t}\mathbf{x_{t-1}}, \beta_t\mathbf{I})$$

Data

Noise

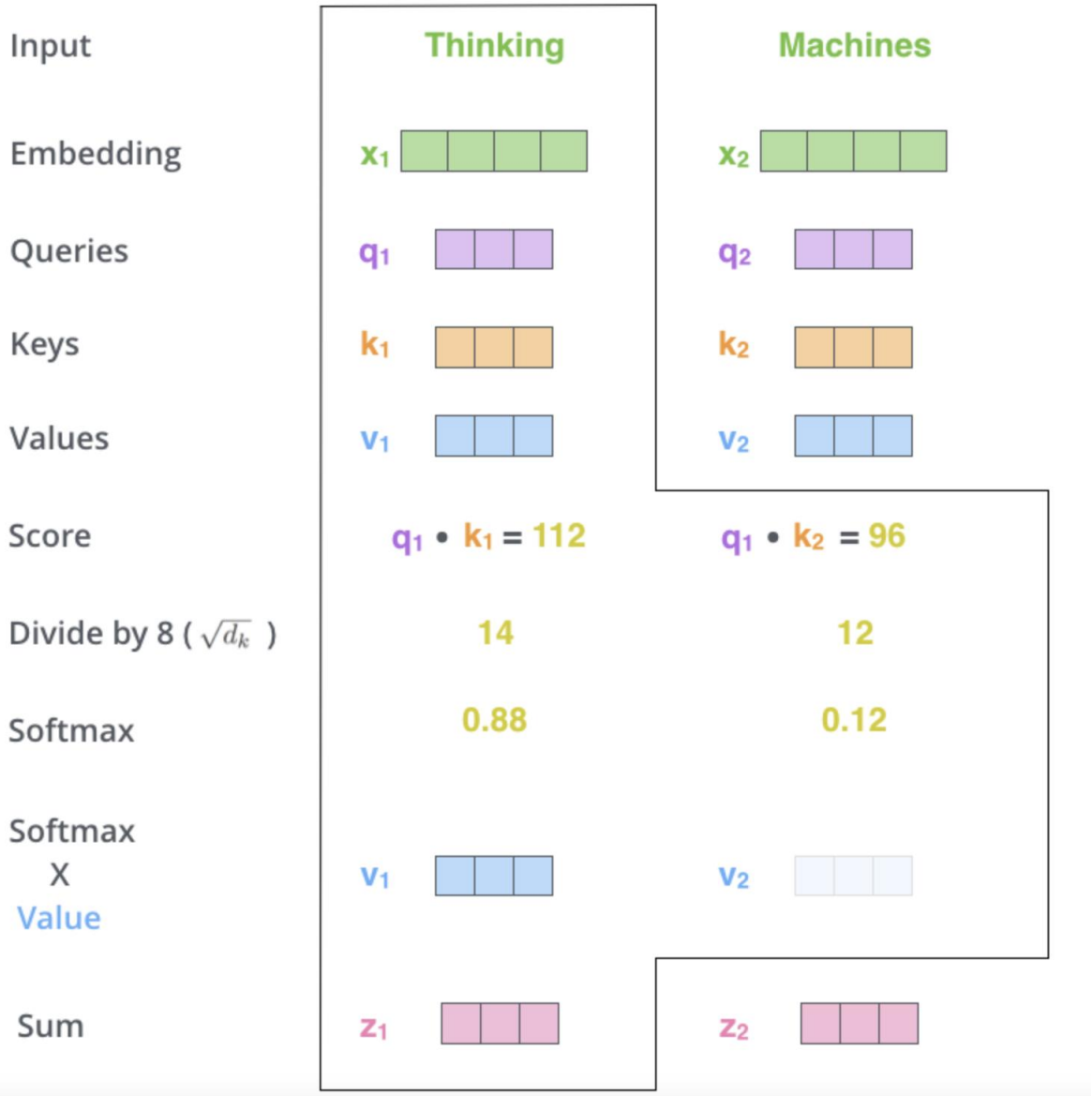
$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \sigma_t^2\mathbf{I})$$





https://arxiv.org/abs/2203.17003
https://yang-song.net/blog/2021/score/

Transformer & GPT



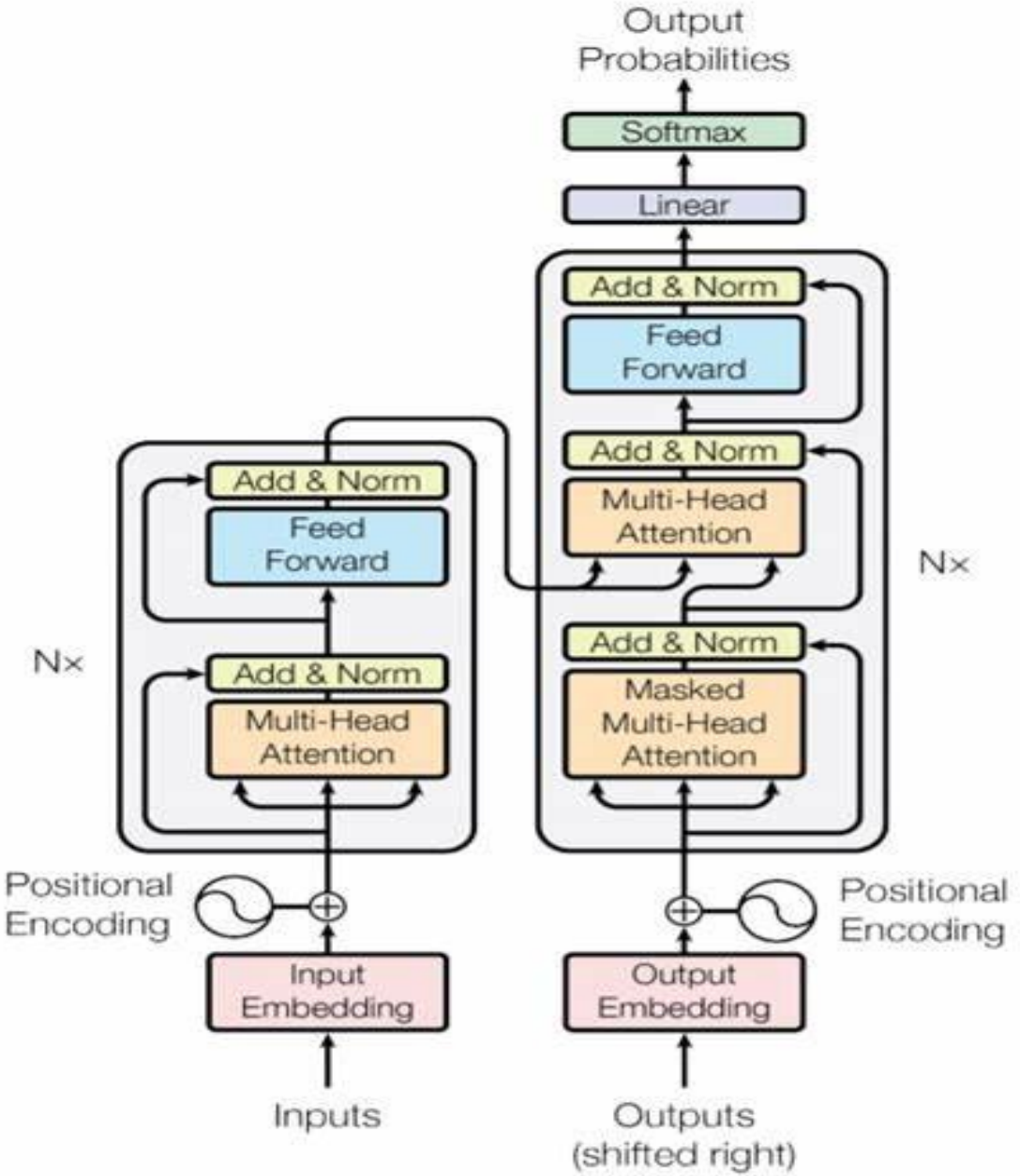


Figure 1: The Transformer - model architecture.

https://jalammar.github.io/illustrated-transformer/

实操2-单机多卡的训练 MolGPT

```
python single_train.py
```

```
python test_allreduce.py
```

```
torchrun --nnodes 8 --node_rank $node_rank
nproc_per_node=8 --master_addr=0.0.0.0 --
master_port=25678 dist_train.py
```

参数说明：
节点数
节点-rank
每个节点几个process
master addr： 0.0.0.0 即本机
端口随意

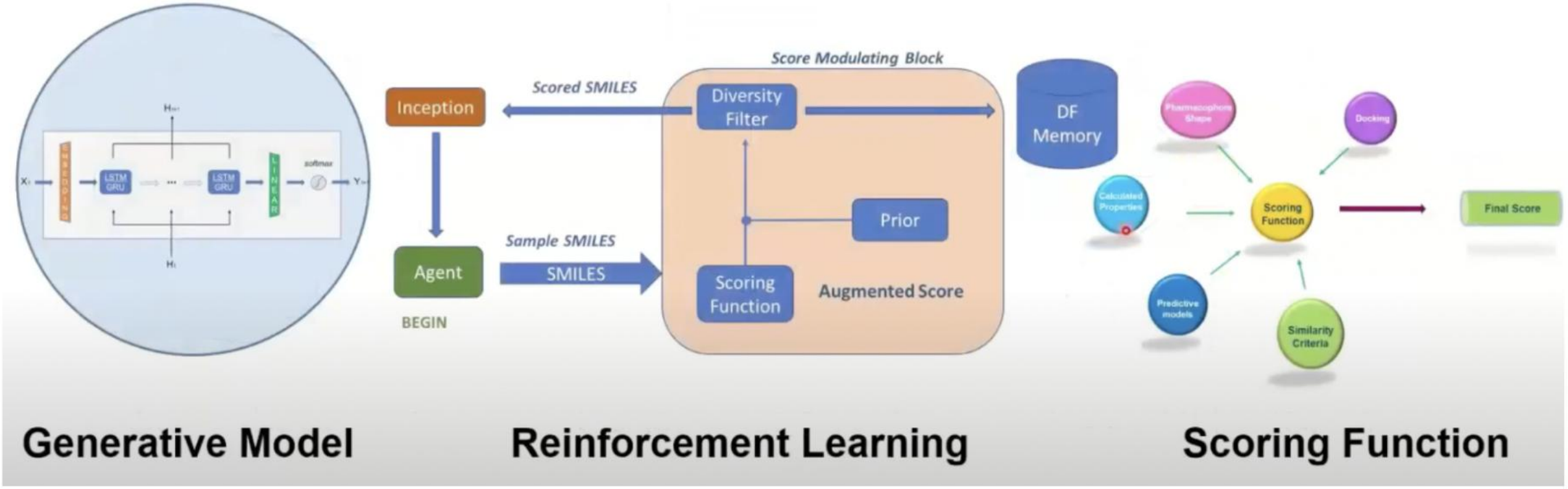
很多影响利用率的因素，会使你的**实验效率**完全不一样
tweak:
batch size
data loader num-workers
pin memory
non-blocking
改成奇怪的数字比如256->237

Goal-directed 生成

conditional modeling

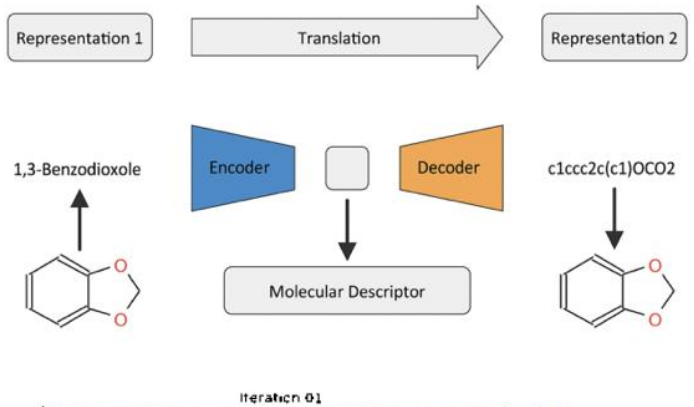
几乎所有的生成式模型都可以变成conditional的生成式模型。

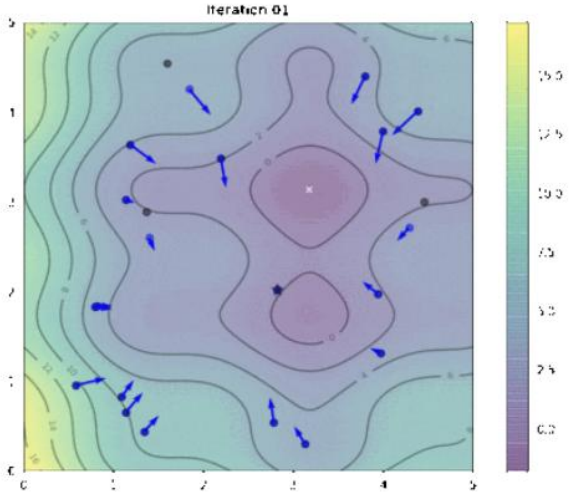
强化学习



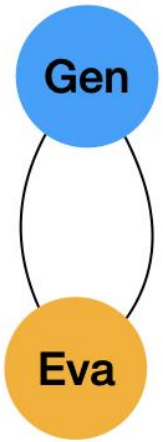
COM: Molecule Swarm Optimization

怎么生成？怎么评价？怎么优化？

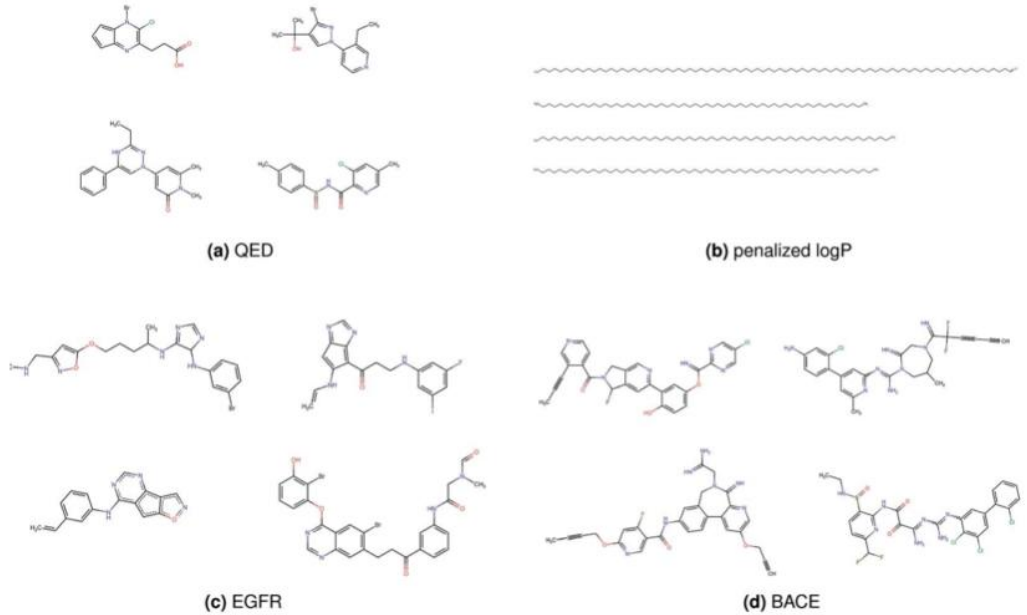




$$V^i(t+1) = wV^i(t) + c_1r_1(pbest^i - X^i(t)) + c_2r_2(gbest - X^i(t))$$



QED
logP
EGFR
BACE
Rewards for substructures



BACE experiment:

1.run 100 steps optimization with 200 restarts

2.needs a **QSAR** model achieves a Spearman correlation coefficient of 0.7 on real data

3.can recover if **similarity to c** is treated as reward

Already Local minimum

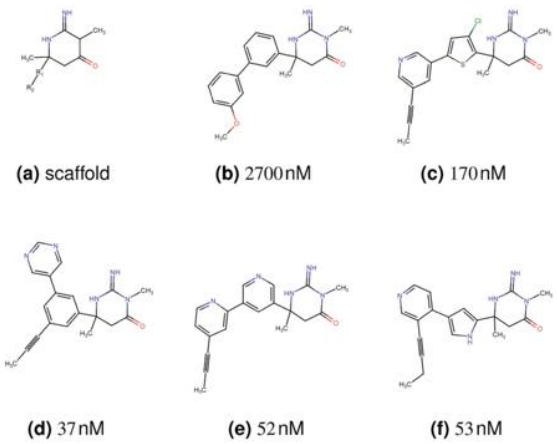
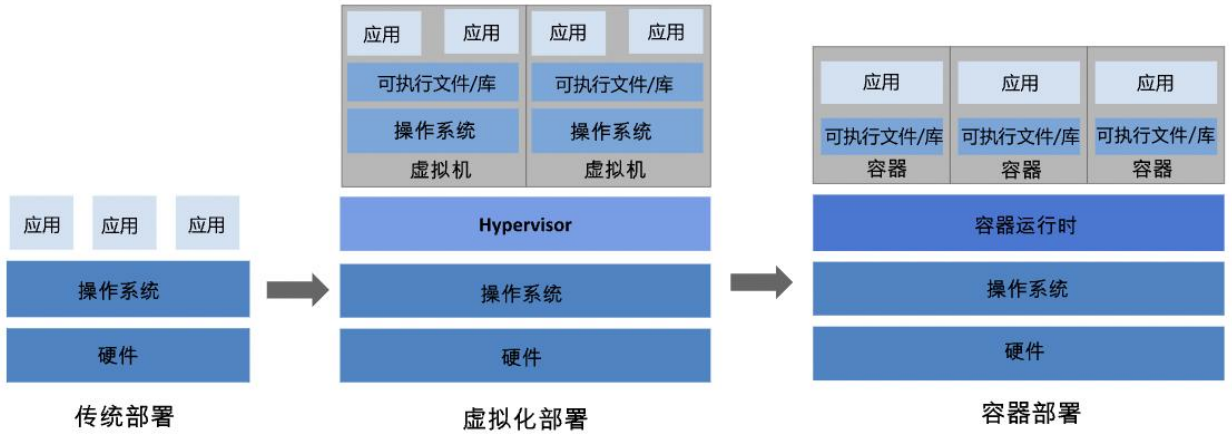


Fig. 3 (a) Iminopyrimidinone scaffold fixed in the optimization. R1 and R2 are aromatic rings. (b) Starting point of the optimization. (c) Best reported compound by Stamford *et al.* (d–f) Top 3 compounds found by our method. All compounds are depicted with their predicted binding affinity to asparyl protease b-site APP cleaving enzyme-1 (BACE1).

机器学习平台-容器编排

kubenetes




```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  selector:
    matchLabels:
      app: nginx
  replicas: 2 # 告知 Deployment 运行 2 个与该模板匹配的 Pod
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2
        ports:
        - containerPort: 80
```

存储编排、网络编排
自动完成计算的资源分配：你告诉 Kubernetes 每个容器需要多少 CPU 和内存 (RAM)
为可扩展性设计：在不改变上游源代码的情况下为集群添加功能。

剩下还有一系列功能如服务发现和负载均衡、自动部署和回滚、水平扩缩等，目前很多互联网公司承载业务也会选择k8s。

https://kubernetes.io/zh-cn/docs

机器学习平台-镜像

你的应用程序
各种依赖-如hf、flash attetion
pytorch
cudadnn，nccl，cuda-sparse
cuda
cuda-runtime
cuda-driver，web-adaptor-driver，rdma
硬件

```
FROM nvcr.io/nvidia/pytorch:23.12-py3

USER root
RUN apt-get update
RUN apt-get install sudo

WORKDIR /workspace
COPY flash-attention /workspace/flash-attention
RUN pip uninstall -y flash_attn && cd /workspace/flash-attention && MAX_JOBS=8 python setup.py install
RUN pip install numpy==1.24.4
RUN pip install nltk
RUN pip install transformers==4.38.2
```

docker build . -t gzy-hub/training-mlm-2312:1.0.0-beta2

docker run -it --gpus '"device=1,2,3,5"' -v /mnt/inspire:/mnt/inspire gzy-hub/training-mlm-2312:1.0.0-beta2 /bin/bash

sudo docker push gzy-hub/training-mlm-2312:1.0.0-beta2

平台提供了官方镜像，在平台的官方镜像pip然后保存

https://catalog.ngc.nvidia.com/orgs/nvidia/containers/pytorch

机器学习平台-存储

HPC并行存储：网络链接的存储空间，
POSIX接口就像用自己的本地文件系统一样。不过一些操作比如list是比较慢的。

空间-项目 个人文件 共享文件
df -h

S3对象存储：比如oss，以buckt和对象id作为存储的，易扩展、备份、迁移是存储大规模训练数据的首选

ossutil ls oss://xx/temp/1003.mp3

可上网区可下载

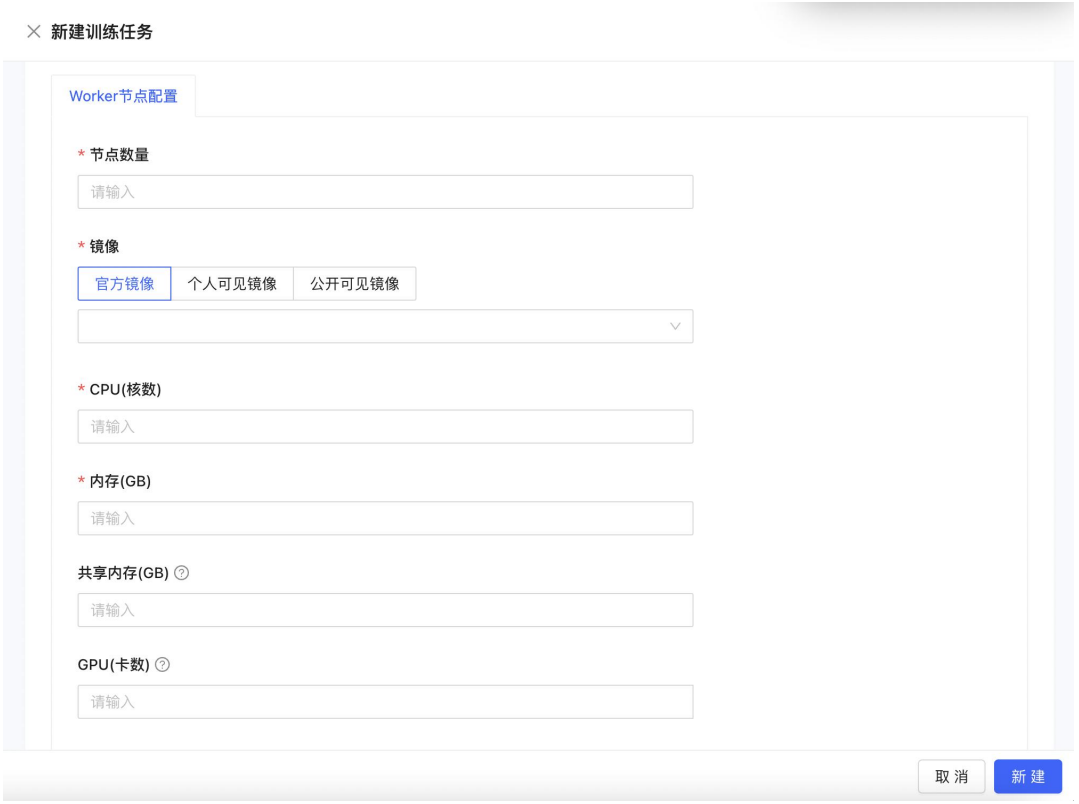
机器学习平台-任务(K8s Operators)

交互式建模

- 一般有jupyter notebook和webide等开发工具
- 可以上传少量的数据、代码等
- 可以用于任务开发、镜像等
- 可能会大量占用造成资源浪费
- 会有自动回收机制，4小时利用率低自动回收

分布式训练

- 管理训练任务的生命周期，分配资源、初始化镜像、组成网络、失败保留等
- 通过环境变量的方式告知应用程序当前的环境



torchrun --nnodes ${PET_NNODES} --node_rank ${PET_NODE_RANK} --nproc_per_node={用户自己填} --master_addr=${MASTER_ADDR} --master_port=${MASTER_PORT}

大模型时代常见训练技术

Activation checkpointing

Gradient accumulation

Offload

Mixed Precision

算子优化

算子融合

高效attention

高效通信

分布式训练

	切分维度
zero	优化器参数、grads
PP	模型
TP	模型
SP	序列
MOE	FLOPS不变增加参数量

Thanks !