

# 第六讲：分布式学习、压缩与可解释性

南京大学人工智能学院

申富饶

01

# 分布式学习

# 1.1 分布式学习简介

分布式学习简介

划分方法

通信机制

模型聚合

# 什么是分布式学习

- 随着模型规模、训练数据量等不断增长，单个计算机已经无法完成机器学习训练过程中一些的计算、存储任务。
- 分布式机器学习尝试**用若干廉价的、普通的机器取代单个高性能计算机，来完成和加速机器学习的训练过程**，即利用更多的机器，处理更多的数据。
- **神经网络的分布式学习**主要研究如何使用计算机集群来加速大规模神经网络模型的训练。

# 为什么需要分布式学习

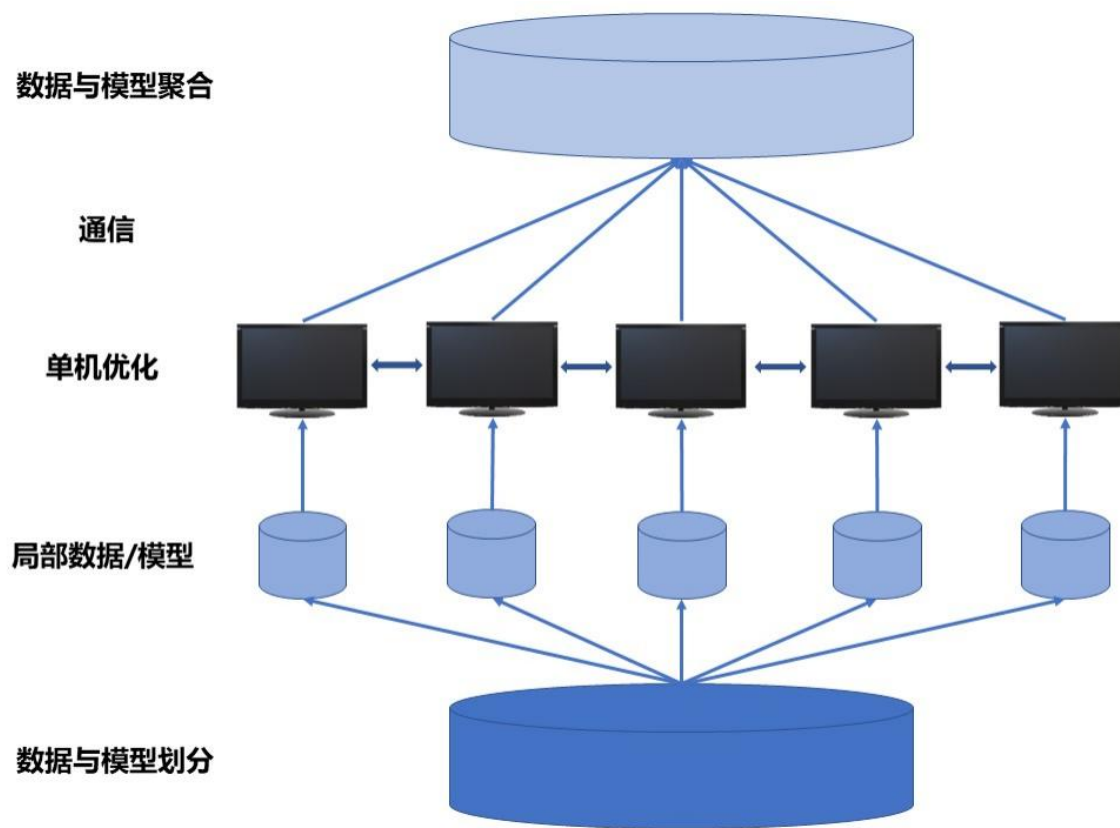
- 神经网络已经为人工智能领域带来了巨大的发展和进步，但训练神经网络模型需要大量的计算。
- 在单个服务器节点（4路消费级GPU）机器上使用 ImageNet 数据集完成一次ResNet-50的训练可能要耗费**数天**时间
- 而借助分布式学习可以通过 GPU 集群将训练时间**降低至几分钟**。

# 为什么需要分布式学习

- 在实际生活中运用神经网络技术解决生产问题时，海量训练数据、问题复杂程度高等诸多挑战是难以避免的，因此必须使用更复杂的神经网络模型来解决问题，这些问题单机训练是无法解决的。
- 因此当面临以下三种情况时，一般会优先考虑使用分布式学习的方法来训练神经网络模型：
  - 第一，计算量太大；
  - 第二，训练数据太多；
  - 第三，模型规模太大。

# 分布式学习的基本流程

- 分布式学习的基本流程可以划分为以下四个主要部分：**数据与模型划分单元**、**单机优化单元**、**通信模块单元**以及**数据与模型聚合单元**。



## 1.2 划分方法

分布式学习简介

划分方法

通信机制

模型聚合

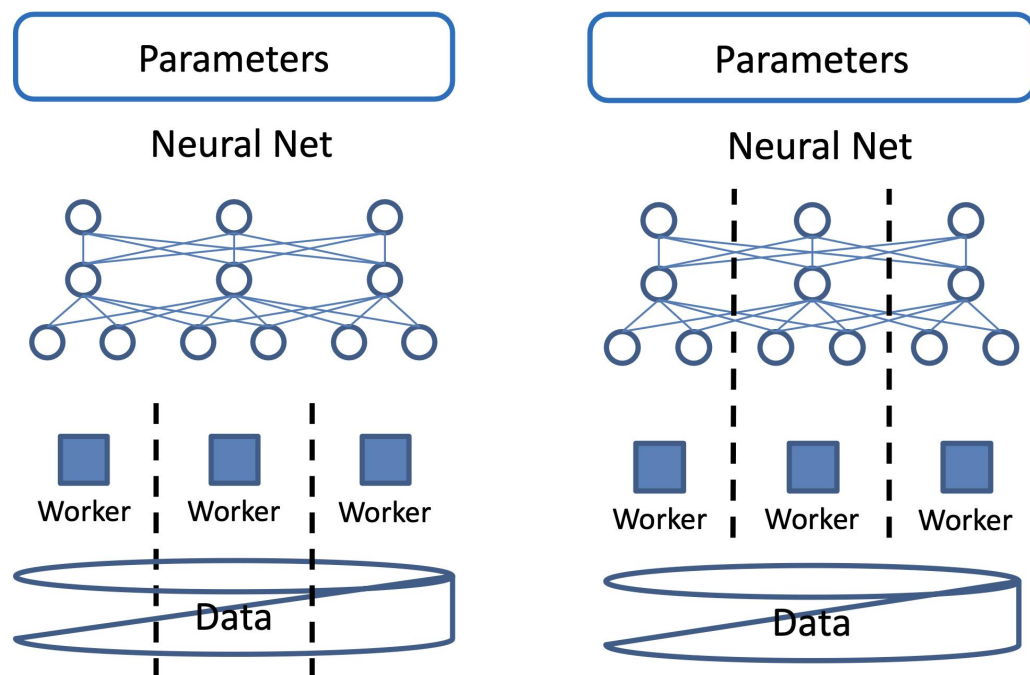


# 常用的划分方法

- 当我们拥有大量训练数据或大规模神经网络模型时，由于存储容量和计算性能受限等原因，通常无法在单个工作节点上完成模型学习，需要先将任务进行划分并分配给各个工作节点。

- 常用的划分方法：

- 数据并行
- 模型并行



# 基于数据的划分

- **数据并行**：当数据规模过大，无法单独存储在一个工作节点上，数据将被分割并分发到每个工作节点。然后，每个工作节点使用分配得到的数据对神经网络模型进行本地训练。
- 要求：尽可能让每台机器上的局部训练数据与原训练数据独立同分布。
- 常用的划分方式有：**随机采样、置乱切分**

# 基于数据的划分

## ➤ 随机采样:

- 把训练数据作为采样的数据源，通过有放回的方式进行随机采样，然后按照每个工作节点的容量为其分配相应数目的训练样本。
- **优点：** 随机采样方法可以保证每台机器上的局部训练数据与原始训练数据是独立同分布的，因此在训练的效果上有理论保证。
- **缺点：** 因为训练数据较大，实现全局采样的计算复杂度比较高；其次，如果随机采样的次数小于数据样本的数目，可能有些训练样本会一直未被选出，导致辛苦标注的训练样本并没有得到充分的利用。

# 基于数据的划分

## ➤ 置乱切分：

- 将训练数据进行乱序排列，然后按照工作节点的个数将打乱后的数据顺序划分成相应的小份，随后将这些小份数据分配到各个工作节点上。每个工作节点在进行模型训练的过程中，只利用分配给自己的局部数据。一定阶段后，再次进行数据打乱和重新分配。
- **比较：**置乱切分方法相比于随机采样方法，虽然数据的分布与原始数据分布略有偏差，但是其计算复杂度比全局随机采样要小很多，而且置乱切分能保留每一个样本，直观上对样本的利用更充分。

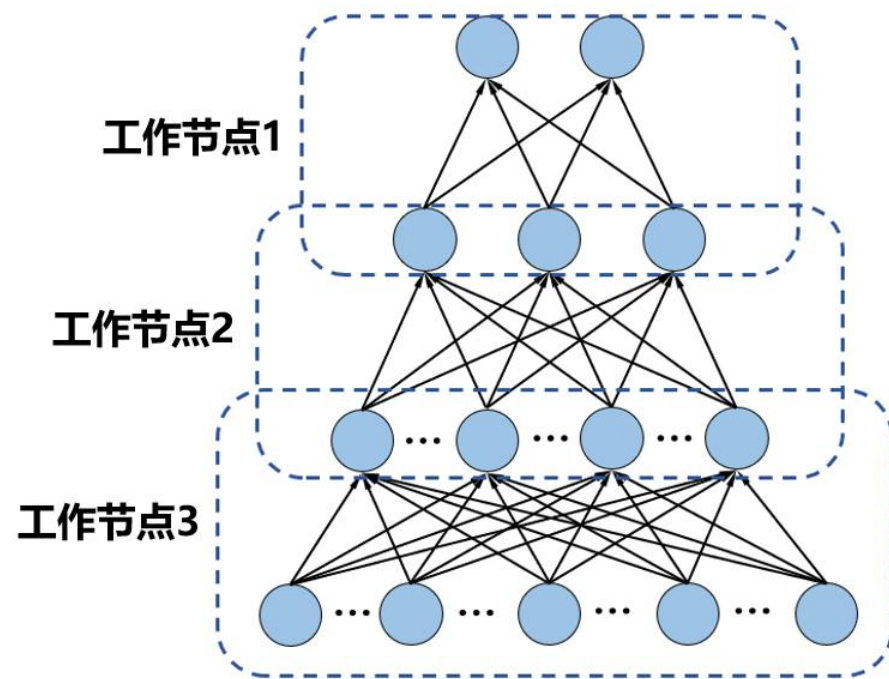
# 基于模型的划分

- **模型划分**：对于规模大、参数多、无法在单个工作节点存储的神经网络模型，有必要将神经网络模型的结构进行划分，并分配给各个工作节点。
- 与简单的线性模型不同，神经网络是高度非线性的，各个工作节点不能相对独立地完成对自己负责的参数的训练和更新，必须依赖与其他工作节点的协作。
- 常用的划分方式有：**横向按层划分、纵向跨层划分、模型随机划分。**

# 基于模型的划分

## ➤ 横向按层划分：

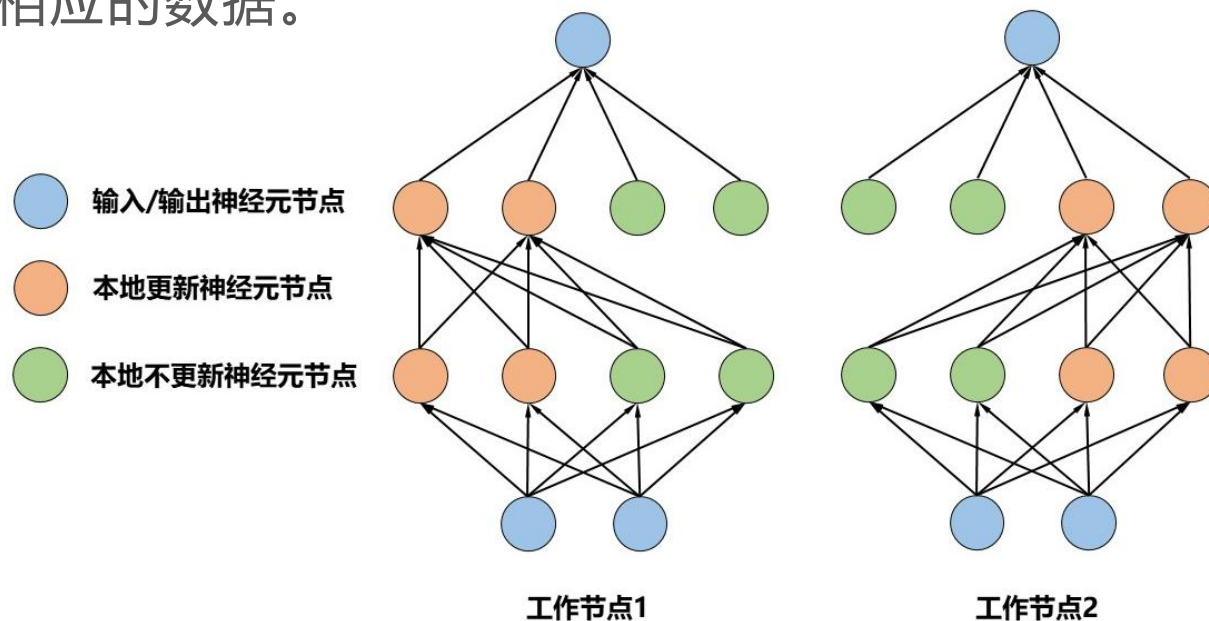
- 将神经网络每两层间的参数、激活函数值和误差传播值存储于一个工作节点。
- 当前传过程需要前一层的激活函数值时，向对应的工作节点发出请求并接收相应的数据，然后再利用这些数据计算本层的前向激活函数值；
- 当后传过程需要后一层的误差传播值时，同样，向对应的工作节点发出请求并接收相应的数据，然后利用这些数据计算本层的后向误差传播值。



# 基于模型的划分

## ➤ 纵向跨层划分：

- 将每一层的参数均等地划分成若干份，每一个工作节点存储由所有层的一部分参数构成的子网络模型。
- 在前传和后传过程中，如果需要子模型以外的激活函数值和误差传播值，则向对应的工作节点发出请求并接收相应的数据。



# 基于模型的划分

## ➤ 模型随机划分：

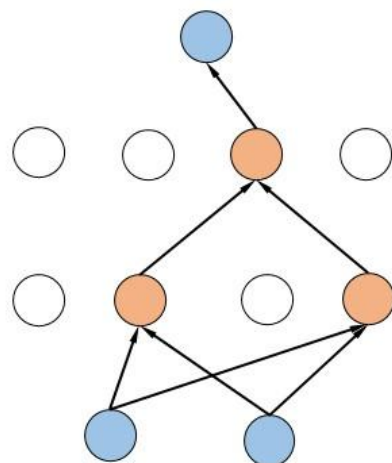
- 人们发现神经网络具有一定的冗余性，可以找到一个规模小很多的子网络（称为**骨架网络**），其效果与原网络差不多。
- 可以首先按照某种准则，在原网络中选出骨架网络，作为公用子网络存储于每个工作节点。除骨架网络之外，每个工作节点还会随机选取一些其他节点存储，以探索骨架网络之外的信息。骨架网络周期性地依据新的网络重新选取，而用于探索的节点也会每次随机选取。



# 基于模型的划分

## ➤ 模型随机划分示例：

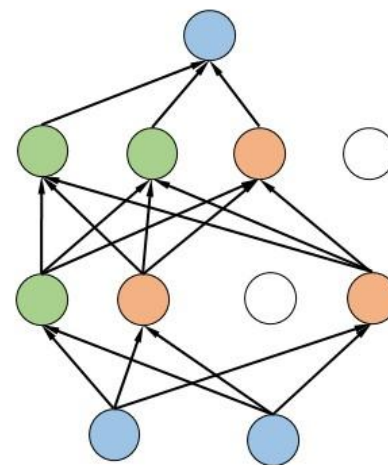
- 输入/输出神经元节点
- 骨架神经元节点
- 本地不更新神经元节点
- 本地更新非骨架神经元节点



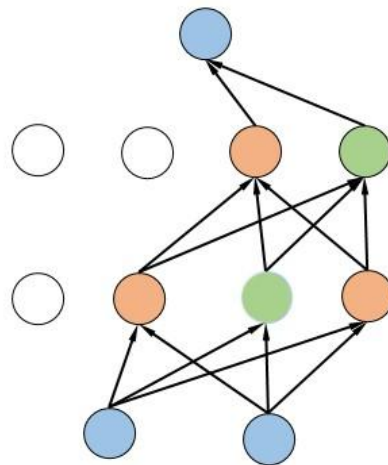
骨架网络

随机探索  
→

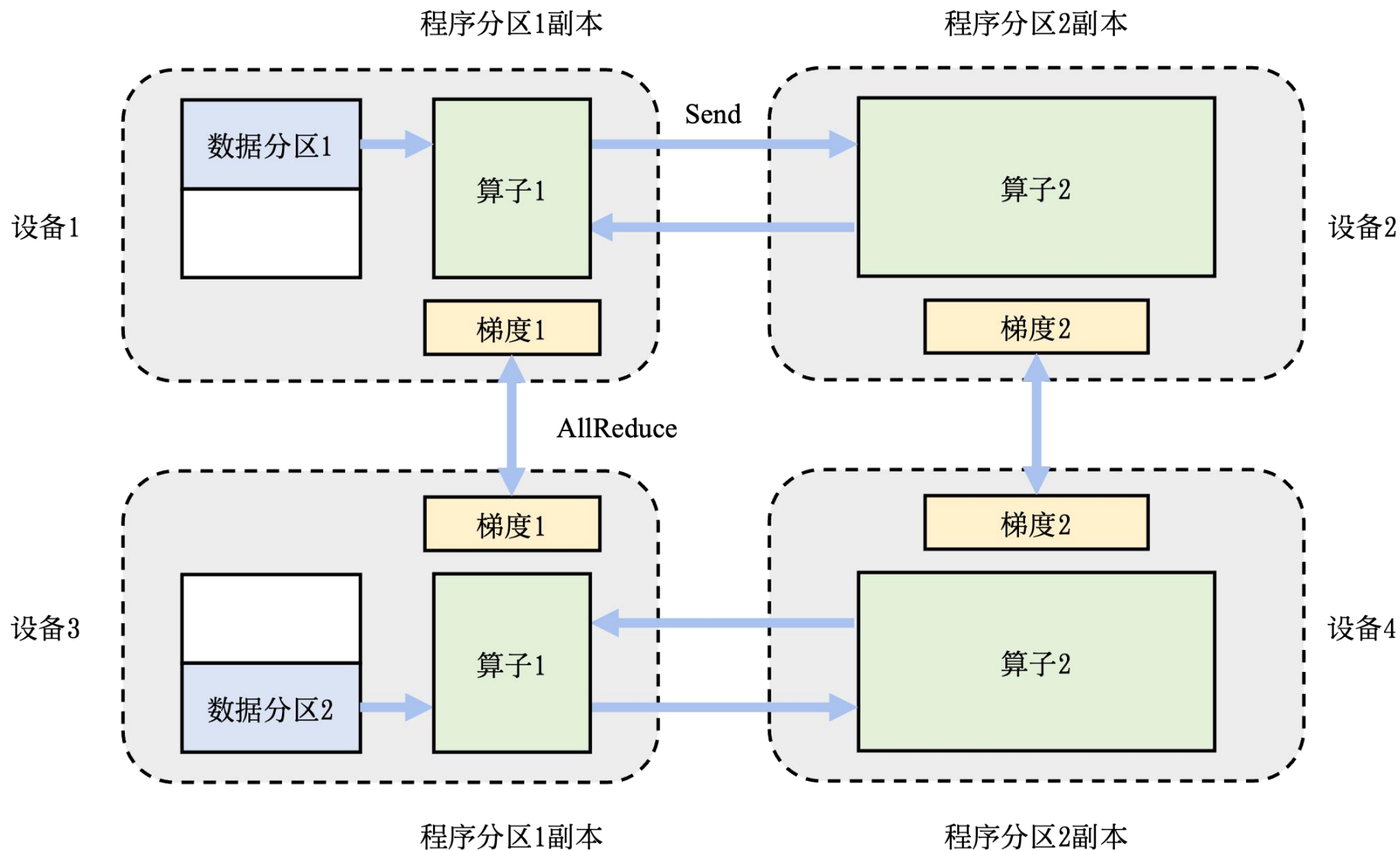
随机子网络1



随机子网络2



# 混合并行：数据并行 + 模型并行



## 1.3 通信机制

分布式学习简介

划分方法

通信机制

模型聚合

# 通信的内容

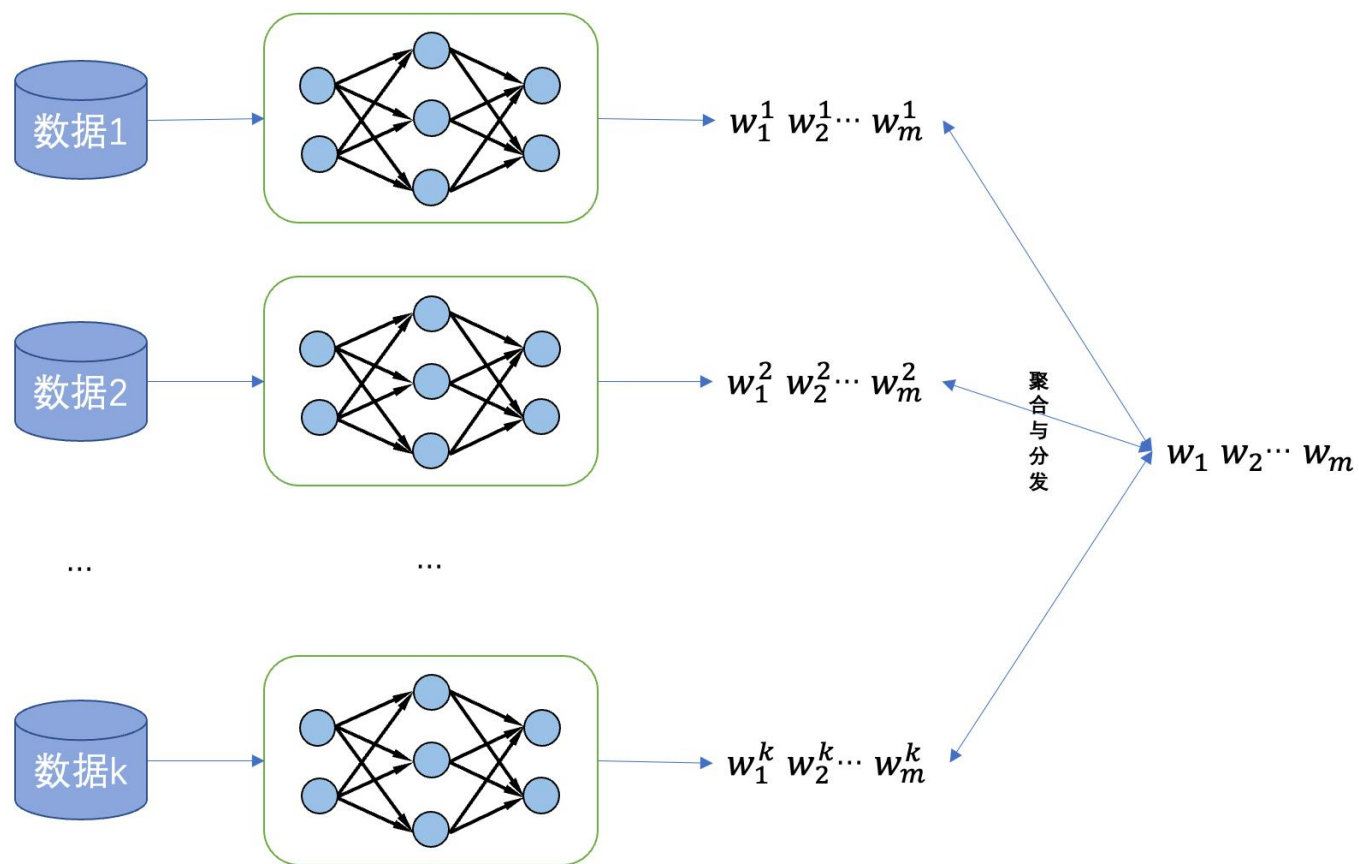
- 通信的内容与划分方法有关：

**基于数据划分的方法：**工作节点各自完成本地的学习任务，然后互相交流各自对模型的修改。因此，在此情形下通信的内容是**模型参数或参数更新**。

**基于模型划分的方法：**各个工作节点利用同一份数据对模型的不同部分进行训练，每个节点要依赖其他节点的中间计算结果。因此，在此情形下通信的内容是**计算的中间结果**。

# 通信的内容

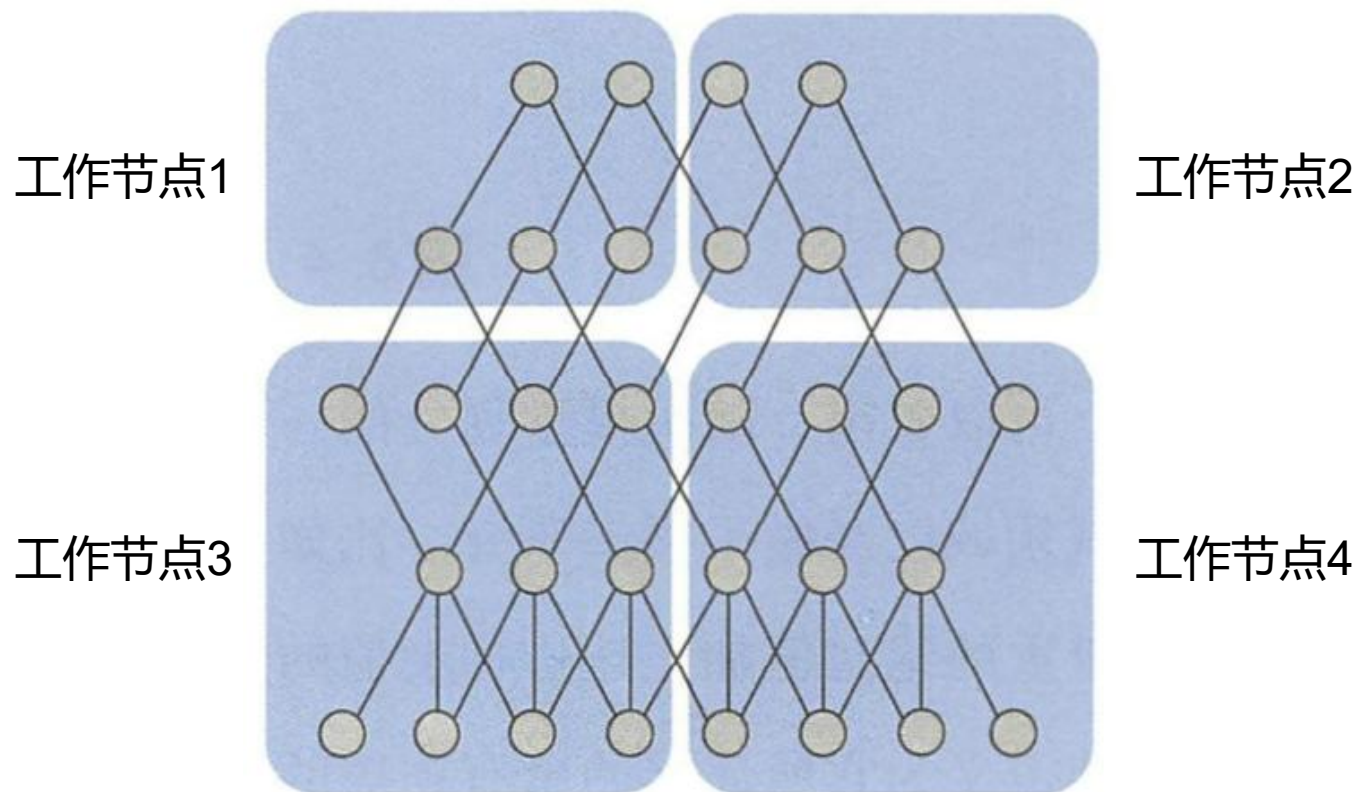
- 参数（或参数的更新）示例



# 通信的内容

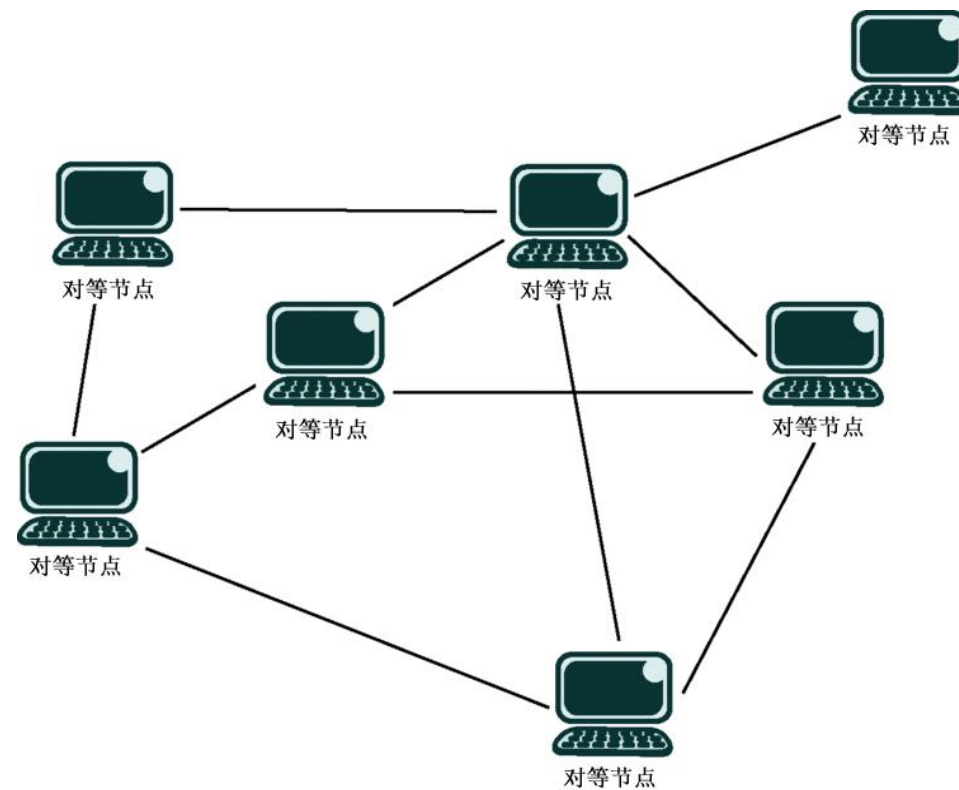
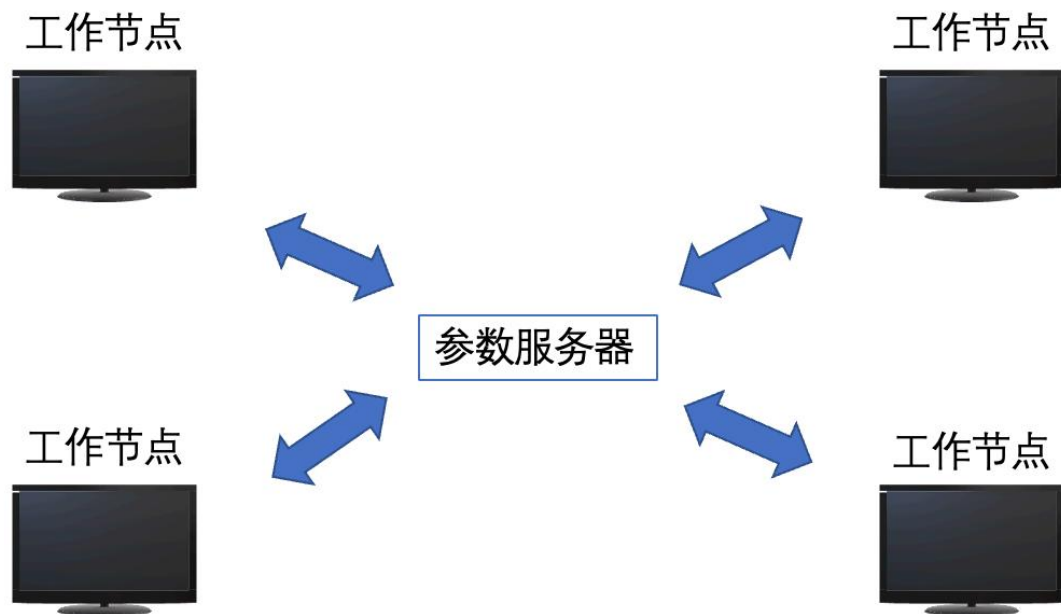
## • 计算的中间结果示例

- 激活函数值
- 误差信息
- 梯度更新
- 等等



# 通信的拓扑结构

通信拓扑结构分为**参数服务器架构**与**去中心化架构**，但在分布式学习中**参数服务器架构**更加常用



# 通信的步调

- **参数服务器**这种通信拓扑将各个工作节点之间的计算相互隔离，取而代之的是工作节点与参数服务器之间的交互。利用参数服务器提供的参数存取服务，各个工作节点可以独立于彼此工作。
- 工作节点对全局参数的访问请求通常分为获取参数（PULL）和更新参数（PUSH）两类。服务器节点响应工作节点的请求，对参数进行存储和更新。
- 有了参数服务器，通信的步调既可以是**同步**的，也可以是**异步**的，甚至是**同步和异步混合**的。



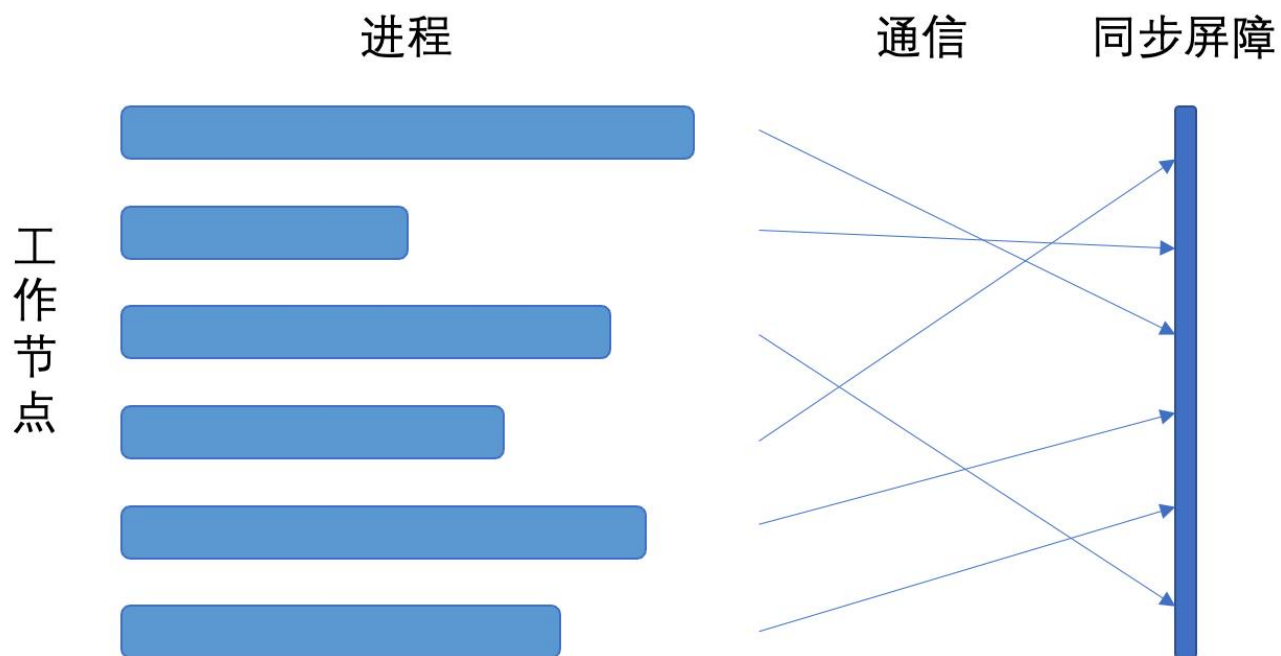
# 通信的步调

- **去中心化架构**中各工作节点之间直接互相通信，无需中心节点参与。
- 梯度信息在工作节点之间互相传递，所有节点都能获得全局聚合后的梯度信息。
- 去中心化架构中通信的步调通常是**同步的**，一些研究论文提出了异步或半同步的变种，但需要复杂的协调机制支持。

# 通信的步调

## ➤ 同步通信

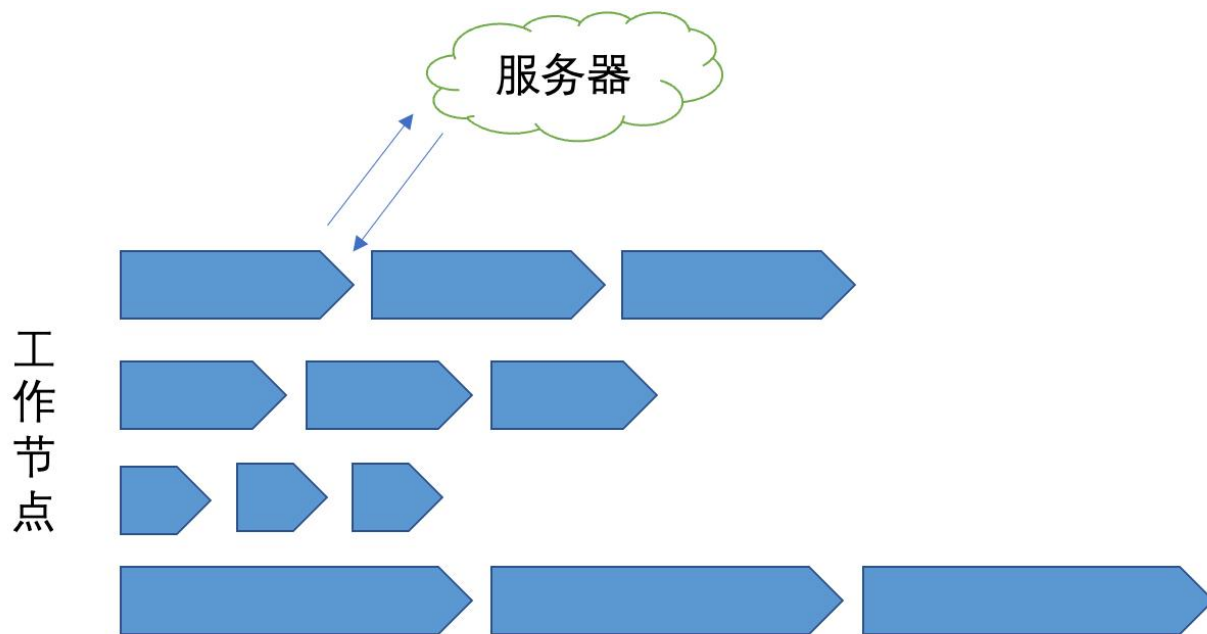
当集群中的一个工作节点完成本轮迭代后，需要等待集群中的其他工作节点都完成各自的任务，才能共同进行下一轮迭代。



# 通信的步调

## ➤ 异步通信

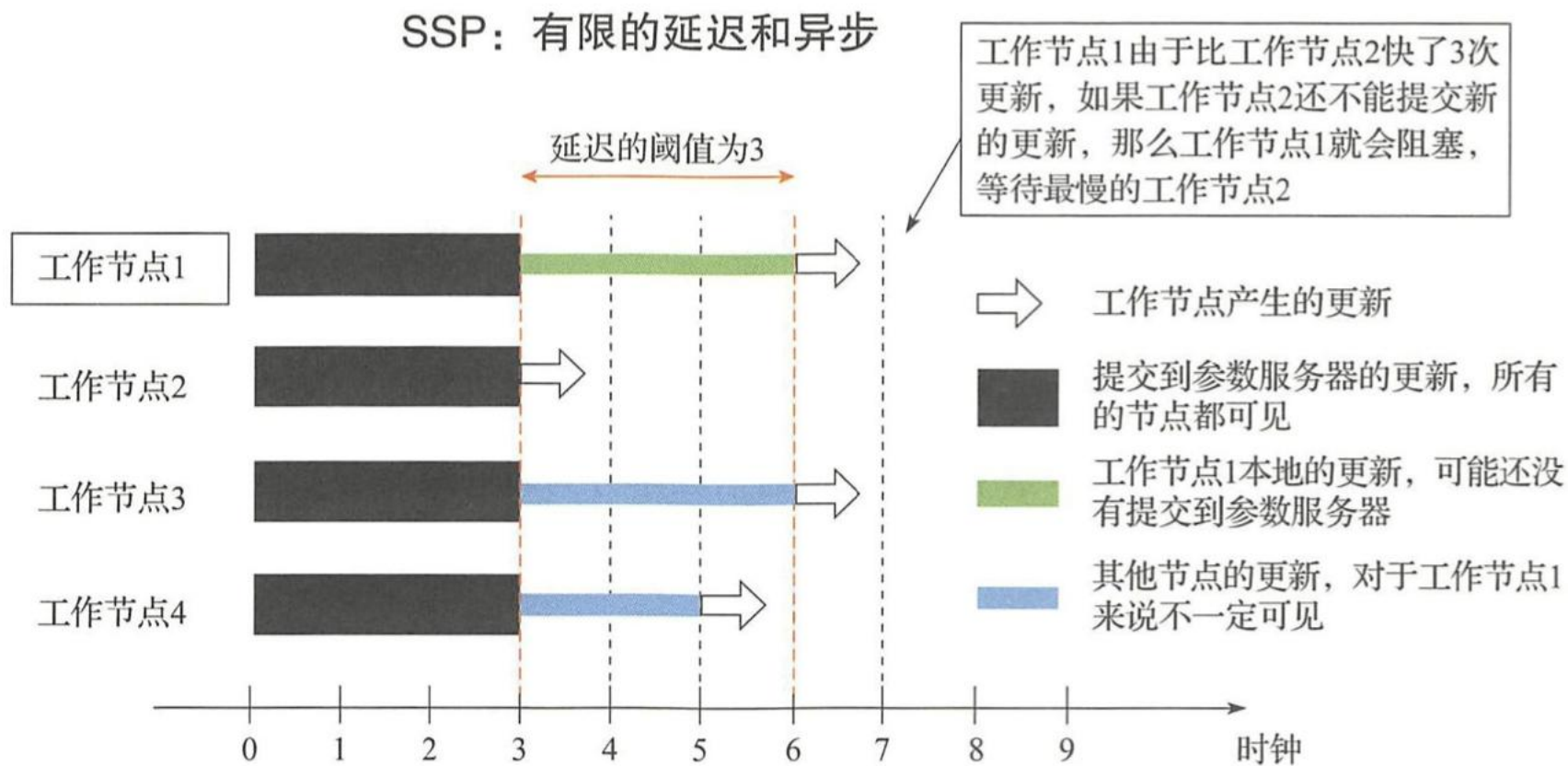
当集群中的一个工作节点完成本轮迭代后，无须等待集群中的其他工作节点，就可以继续进行后续训练，因此系统效率可以大大提高。



# 通信的步调

## ➤ 同步和异步的平衡

延时同步并行 SSP：控制最快和最慢节点之间相差的迭代次数不超过预设的阈值



# AI框架使用的通信库

- **Message Passing Interface (MPI)**

- MPI 常用于在计算集群、超算上编写程序，比如传统科学计算的并程序序。MPI 接口兼容性好，通信功能丰富，在深度学习框架中主要用于 CPU 数据的通信。
- MPI是一个开放接口，有多种实现的库，一种广泛使用的开源实现是 Open MPI。一些硬件厂商也提供针对硬件优化的实现。

- **NVIDIA Collective Communication Library (NCCL)**

- NCCL 常用于多个 GPU 之间通信的实现。它专为Nvidia的计算卡和网络优化，能带来更低的延迟和更高的带宽。

## 1.4 模型聚合

分布式学习简介

划分方法

通信机制

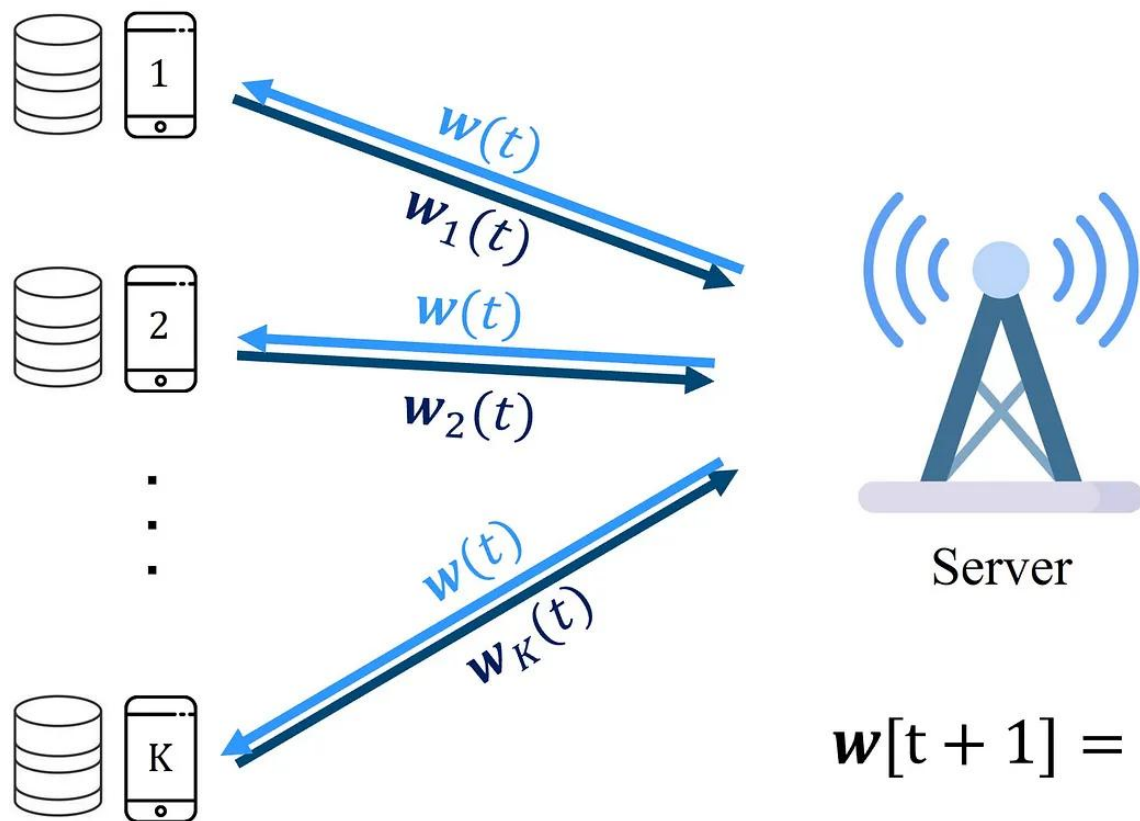
模型聚合

# 常用的模型聚合方法

- **聚合**是分布式学习特有的逻辑，当划分的数据使用分布式训练后，需要对多个设备的本地模型进行聚合。
- 有效的聚合往往会带来更好的加速效果。通常我们对聚合方法有以下需求：
  - 聚合本身的时间代价比较少，这样给整个学习流程带来的额外负担较小；
  - 聚合算法合理、有效，整体的收敛性仍然能保持与单机算法大体一致。
- 常用的模型聚合方法有：**基于模型加和的聚合、基于模型集成的聚合**

# 基于模型加和的聚合

**基于模型加和的聚合方法**是指在将来自不同工作节点的模型或者模型更新进行加权求和。



$$\mathbf{w}[t+1] = \frac{1}{K} \sum_{i=1}^K \mathbf{w}_i[t]$$



# 基于模型加和的聚合

- 但是，并非所有情况下这类基于模型参数加和的聚合方法都是可取的。事实上，只有在凸优化问题中这种简单加和的手段才能保证训练性能。
- 具体地，**假设损失函数关于模型参数是凸函数**，于是以下不等式成立：

$$l(g(\bar{w}; x), y) = l\left(g\left(\frac{1}{K} \sum_{k=1}^K w^k; x\right), y\right) \leq \frac{1}{K} \sum_{k=1}^K l(g(w^k; x), y)$$

- 其中，**左端**是参数平均后的模型  $\bar{w}$  对应的损失函数取值，**右端**是各个局部模型的损失函数值的平均值。这说明，在凸优化问题中，平均模型的性能不会低于原有各个模型性能的平均值。

# 基于模型集成的聚合

- 但是，对于非凸的神经网络，由于损失函数关于模型参数非凸。所以，上述不等式将不再成立，模型的性能在参数平均后也不再具有保证。
- 为了解决这个问题，研究者提出了**基于模型集成的聚合方法**。

# 基于模型集成的聚合

- 虽然神经网络的损失函数关于模型参数是非凸的，但是它关于模型的输出一般是凸的。这时利用损失函数的凸性可以得到如下不等式：

$$l \left( \frac{1}{K} \sum_{k=1}^K g(w^k; x), y \right) \leq \frac{1}{K} \sum_{k=1}^K l(g(w^k; x), y)$$

- 其中，**左端**是对不同局部模型的输出进行平均后对应的损失函数取值，**右端**是局部模型的损失函数值的平均值。所以，如果**对局部模型的输出进行加和平均**，所得到的预测结果要好于局部模型预测结果的平均值。
- 这种对模型输出进行加和平均的方法称为**模型集成(ensemble)**。通过集成多个模型的预测结果，可以取得比单个模型更好的性能。

# 基于模型集成的聚合

- 虽然模型集成在模型精度上有更好的保障，但是模型集成后产生的新模型参数量是局部模型  $K$  倍。考虑到模型聚合会在分布式学习的迭代算法中多次发生，这很可能导致**模型规模爆炸**的问题。
- 人们提出了一种用压缩方法来降低集成模型规模的算法，称为 “**集成一压缩**” 的模型聚合方法。利用模型压缩技术（比如知识蒸馏），获得与局部模型大小相同的新模型作为最终的聚合结果。
- 这种集成一压缩的聚合方法，既可以通过集成获得性能提升，又可以在整个机器学习的迭代过程中保持全局模型的大小基本不变。

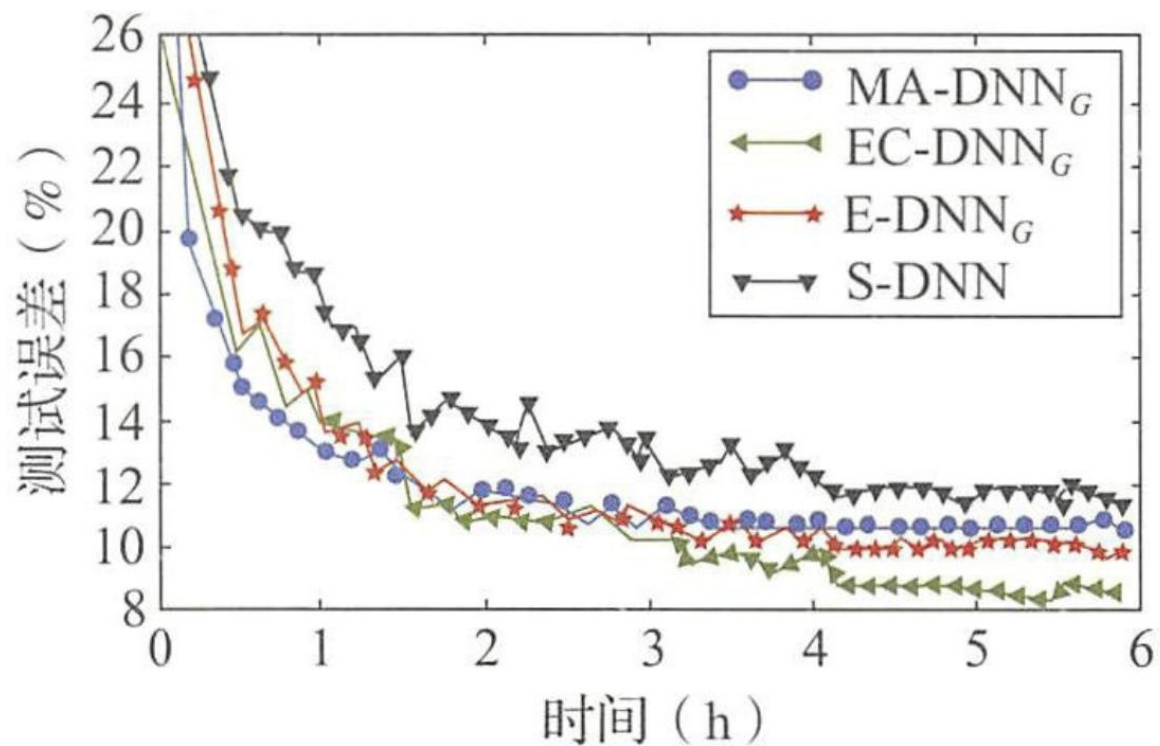
# 基于模型集成的聚合

## 一种集成-压缩算法流程：

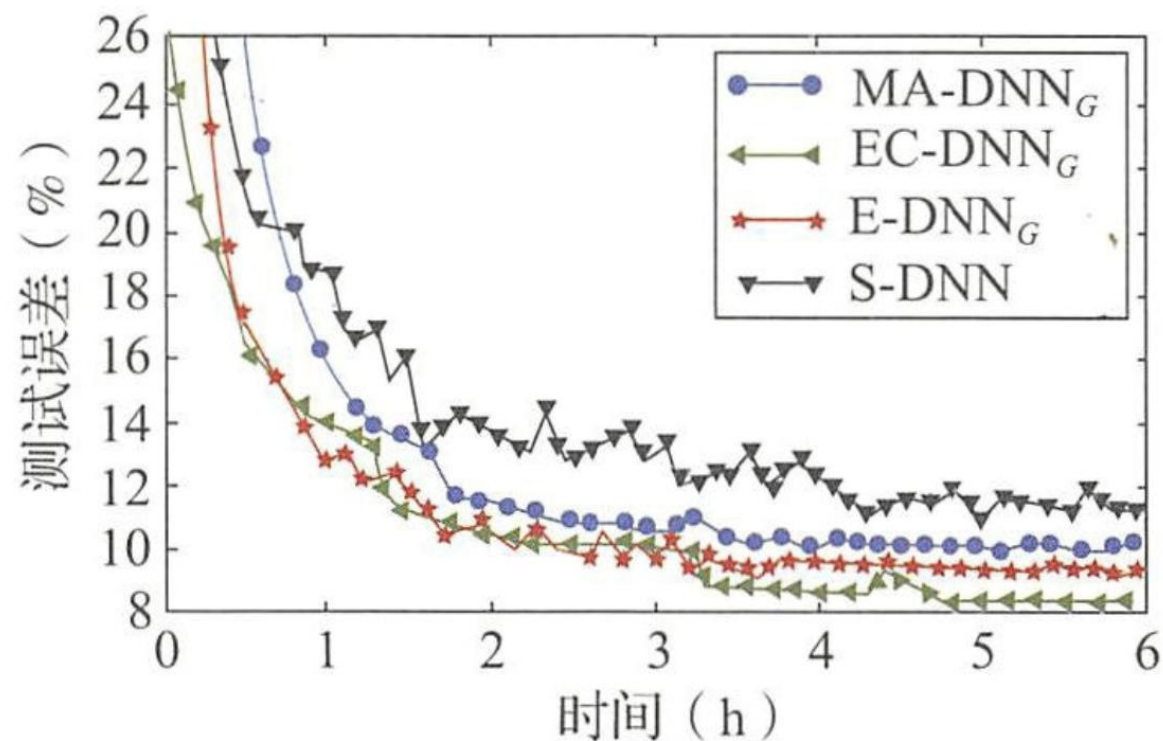
- (1) 各个工作节点依照本地局部数据训练出局部模型；
- (2) 工作节点之间相互通信获取彼此的局部模型，集成所有的局部模型得到集成模型，并对（一部分）局部数据使用集成模型进行再标注；
- (3) 利用模型压缩技术，结合数据的再标注信息，在每个工作节点上分别进行模型压缩，获得与局部模型大小相同的新模型作为最终的聚合结果。

# 在CIFAR数据集上对比不同聚合方式

- S-DNN: 表示模型仅在一个 GPU 上进行训练
- E-DNN: 表示基于模型集成的聚合方法
- MA-DNN: 表示基于模型加和平均的聚合方法
- EC-DNN: 表示基于模型集成-压缩的聚合方法



a)  $K=4$ , CIFAR-10



b)  $K=8$ , CIFAR-10

## 02

# 神经网络压缩

## 2.1 神经网络压缩

重点：神经网络模型巨大且存在冗余

神经网络压缩

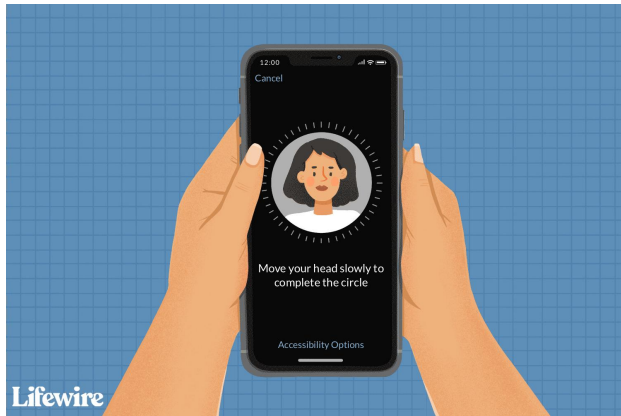
压缩方法

网络剪枝

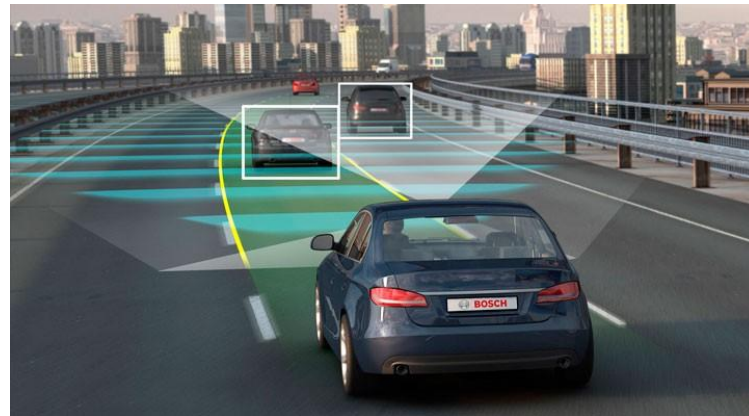


# 神经网络压缩的意义

- 深度学习模型，尤其是深度神经网络，已经成功地应用到计算机视觉任务中
- 大多数的深度神经网络都是在具有GPU计算显卡的电脑甚至大型服务器上训练和部署的。但有一些应用场景，要求神经网络部署在移动设备上
- 移动设备上并不具备拥有良好计算性能和较大内存的GPU计算卡



智能手机上的人脸识别系统



汽车上的自动驾驶系统



智能家居上的语音识别系统

# 神经网络压缩的意义

- **内存占用**: 深度神经网络中有着数以百万计的权重参数，部署一个训练好的深度神经网络模型需要较多的内存。
- **计算速度**: 相比于GPU计算卡和具有较高计算性能的硬件设备，移动设备的计算能力是有限的。然而，深度神经网络模型对一张图像进行计算所需要的浮点数运算次数十分庞大。

神经网络模型	占用内存	浮点数运算次数
AlexNet	232.5MB	$0.7 \times 10^9$
VGGNet	526.4MB	$15.4 \times 10^9$
GoogleNet	26.3MB	$1.5 \times 10^9$
ResNet	97.2MB	$4.1 \times 10^9$
DenseNet	110.2MB	$8.0 \times 10^9$

# 神经网络压缩的意义

- **能量消耗:** 由于利用深度神经网络对图像进行处理需要庞大的计算量，使用这些模型的过程中将会产生大量的能量消耗。手机或者其他一些便携式设备的电量是有限的，所以在这些设备上使用神经网络模型将会使得电量消耗极快。



# 神经网络压缩的意义

神经网络压缩使大语言模型在消费级硬件运行成为可能，中小企业或个人都可以部署自己的大模型

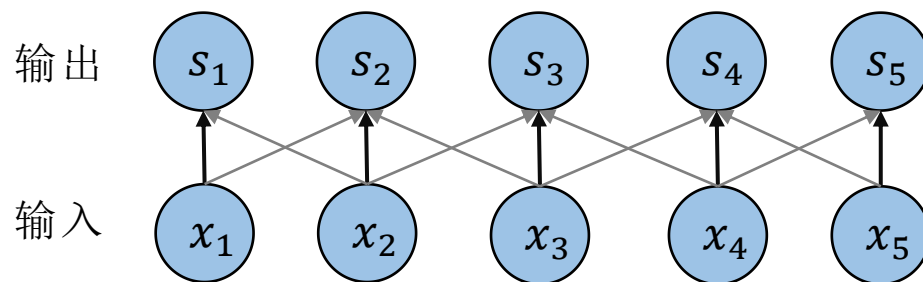
压缩前后的DeepSeek R1对比

指标	全参数模型（671B）	7B 压缩版
语言生成质量	高（如专业级藏头诗）	基础文本生成易出错
逻辑推理能力	复杂问题准确率高	简单任务尚可，复杂任务受限
显存需求	1.34TB（FP16）	8-16GB（INT8）
部署成本	需 16×NVIDIA H100	单卡 RTX 3060 即可
适用场景	国家级科研、AGI	移动端、中小企业

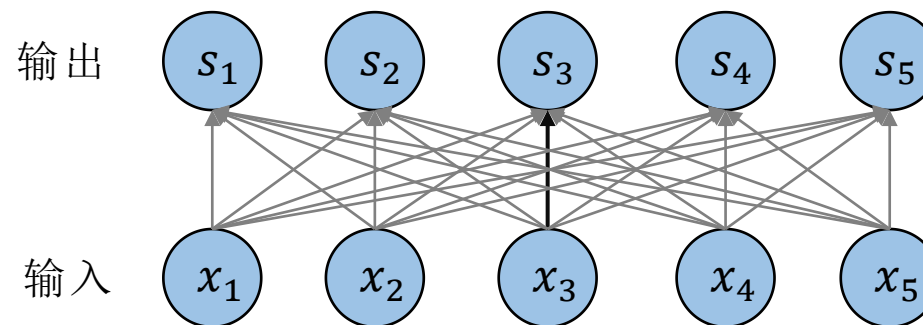
# 神经网络压缩的定义

- 研究表明，神经网络模型中是存在冗余的，可以通过模型压缩的方法来减小神经网络的存储消耗和计算时间
- 卷积神经网络中的权值共享和稀疏连接，是对全连接网络的压缩

卷积神经网络



全连接神经网络



# 神经网络压缩的定义

- 神经网络压缩是指通过改变网络结构或利用量化、近似的方法来减少网络的参数或存储空间，在不大幅损失模型性能的情况下，降低网络计算代价和存储空间。
- 在学术界和工业界都有广泛的研究，有利于神经网络模型部署在轻量级设备上，提高推理速度，使得深度学习能够更广泛的应用于日常生活中，提高生活和工作智能化程度。

## 2.2 压缩方法

参数共享

低秩因子分解

知识蒸馏

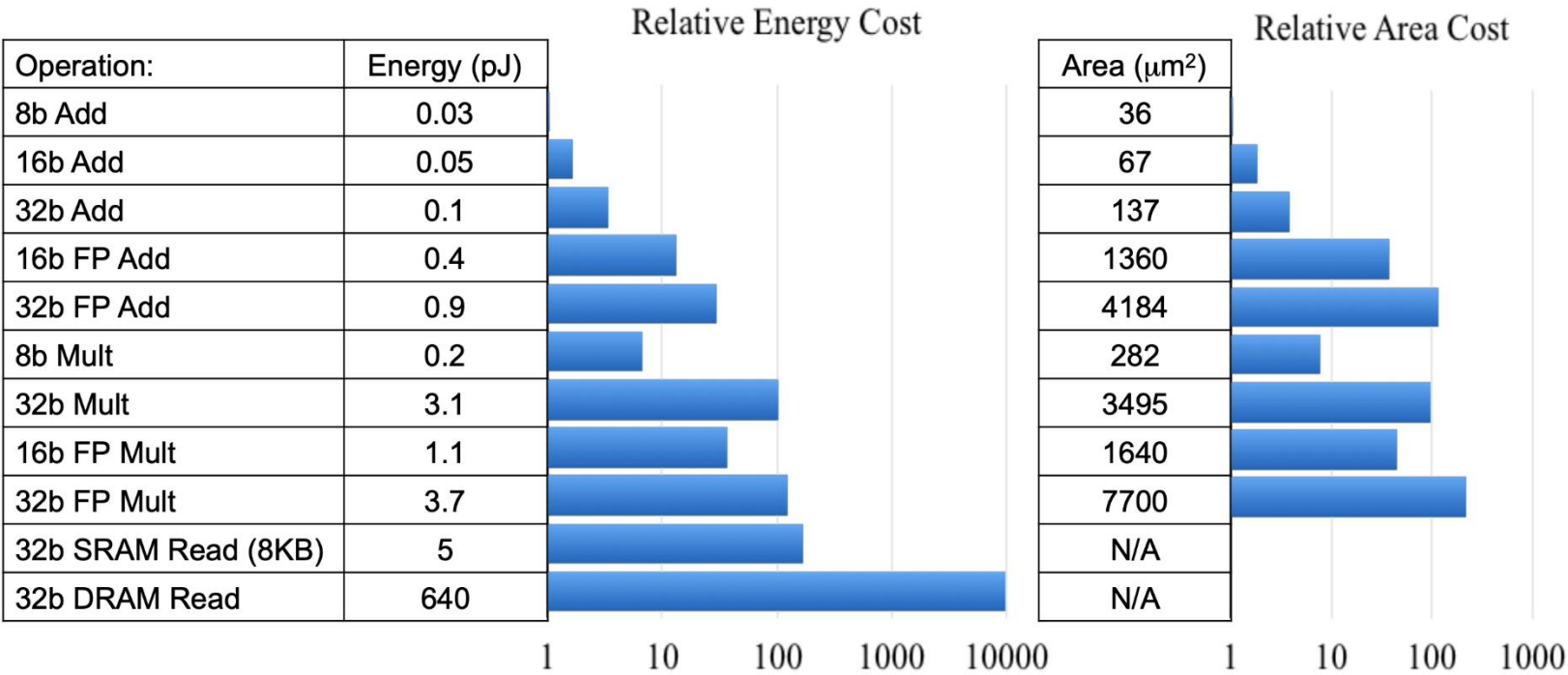
神经网络压缩

压缩方法

网络剪枝

# 模型量化

- 神经网络量化是指使用更小的比特宽度来存储原来的数据，例如用8比特整数值来表示原来的32比特浮点数。
- 模型量化可以在损失少量精度的前提下对模型进行压缩，低精度数操作的能耗以及硬件面积大小要比高精度数操作要少好几个数量级。





# 模型量化

## 模型量化的作用：

- 更少的存储开销和带宽需求。即使用更少的比特数存储数据，有效减少应用对存储资源的依赖。
- 更低的功耗。移动 8bit 数据与移动 32bit 浮点型数据相比，前者比后者高 4 倍的效率，而在一定程度上内存的使用量与功耗是成正比的。
- 更快的计算速度。新型处理器都支持 8bit 数据的快速计算，如果是二值量化则更有优势。

二值化网络可以视为量化方法的一种极端情况：所有的权重参数取值只能为  $\pm 1$ ，也就是使用 1 比特来存储权重和特征。

# 二值化

- 二值化网络如何训练呢？

1. 权重初始化为浮点

2. 前向传播 Forward Pass:

- a. 利用 $\text{sign}()$ 函数把权值 $w_{t-1}$ 量化为  $+1/-1$ , 记为 $w_b$ ;
- b. 利用  $w_b$  来计算前向传播, 由二值权重与输入进行乘法、卷积等运算（实际上只涉及加法）, 获得输出。

3. 反向传播 Backward Pass:

- a. 基于二值权重计算梯度 $\frac{\partial C}{\partial w_b}$ , 把梯度更新到浮点权重 $w_{t-1}$ 上, 得到新权重 $w_t$
- b. 训练结束: 把权重永久性转化为 $+1/-1$ , 供推理使用。

# 低秩因子分解

- 低秩分解方法是通过将高秩张量分解成低秩张量来减少模型的大小和内存的访问次数。张量分解可以被应用到卷积层和全连接层上，并且可以与网络量化方法相结合。
- 低秩：若矩阵 $X$ 是一个 $m$ 行 $n$ 列的数值矩阵， $rank(X)$ 是 $X$ 的秩。假如 $rank(X)$ 远小于 $m$ 和 $n$ ，则称 $X$ 是低秩矩阵。

# 低秩因子分解

- 形式化定义：给定权重矩阵  $W \in \mathbb{R}^{m \times n}$ ，若能将其表示为若干个低秩矩阵的组合，即

$$W = \sum_{i=1}^n M_i,$$

其中  $M_i \in \mathbb{R}^{m \times n}$  为低秩矩阵，其秩为  $r_i$ ，并满足  $r_i < \min(m, n)$ ，  
则每一个低秩矩阵都可分解为小规模矩阵的乘积，

$$M_i = G_i H_i,$$

其中  $G_i \in \mathbb{R}^{m \times r_i}$ ， $H_i \in \mathbb{R}^{r_i \times n}$ 。

当  $r_i$  取值很小时，便能大幅降低总体的存储开销

# 低秩因子分解

- 常见的分解方法：SVD分解、CP分解、Tucker分解等
- **SVD分解**（文献：A singularly valuable decomposition: the SVD of a matrix）

在稀疏参数矩阵中，其权重向量大部分分布在一些低秩的子空间中，通常可以用少数几个基向量来重构模型参数矩阵。

假设矩阵 $A$  是一个 $m \times n$ 的矩阵，则矩阵 $A$ 可以分解为 $U$ 、 $\Sigma$ 、 $V$ 三个矩阵，即

$$A = U\Sigma V^T$$

其中， $U$ 是一个 $m \times m$ 的矩阵； $\Sigma$ 是一个 $m \times n$ 的对角矩阵，其主对角线上的元素不为0，其他元素为0，主对角线上元素为矩阵的奇异值； $V$ 是一个 $n \times n$ 的矩阵。且 $U$ 和 $V$ 都是正交矩阵。满足以下公式：

$$U^T U = I, V^T V = I$$

# 低秩因子分解

- SVD分解

如果矩阵 $A$ 由权重矩阵得来，通常不是满秩矩阵，即存在冗余数据，可以将矩阵 $A$ 中非0的 $r$ 个奇异值保留，则矩阵 $A$ 的分解公式可以转化为：

$$A_{m \times n} = U_{m \times m} \Sigma_{m \times n} V_{n \times n}^T = U_{m \times r} D_{r \times r} V_{r \times n}^T$$

进一步，如果存在极小的奇异值，则可以当作噪声省略，剩下 $k$ 个奇异值，分解公式可以进一步转化为

$$A_{m \times n} = U_{m \times r} D_{r \times r} V_{r \times n}^T \approx U_{m \times k} D_{k \times k} V_{k \times n}^T$$

将矩阵 $U$ 和 $D$ 合并，进一步转化为

$$A_{m \times n} \approx U_{m \times k} D_{k \times k} V_{k \times n}^T = X_{m \times k} V_{k \times n}^T$$

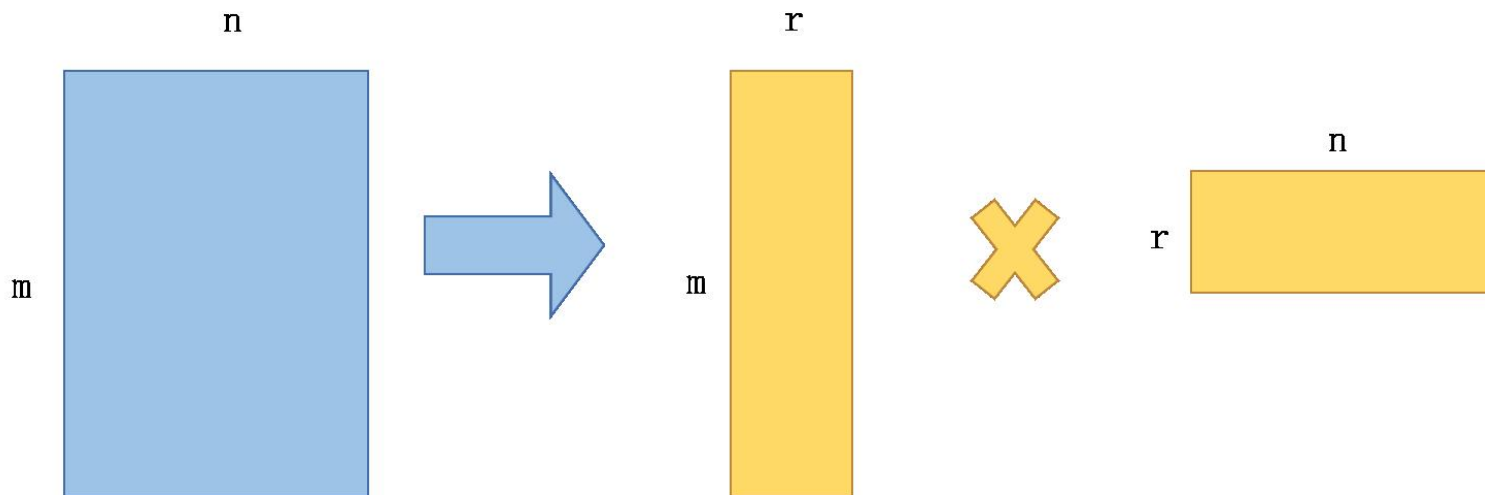
# 低秩因子分解

- SVD分解

经过一系列转化，矩阵 $A$ 的分解公式为：

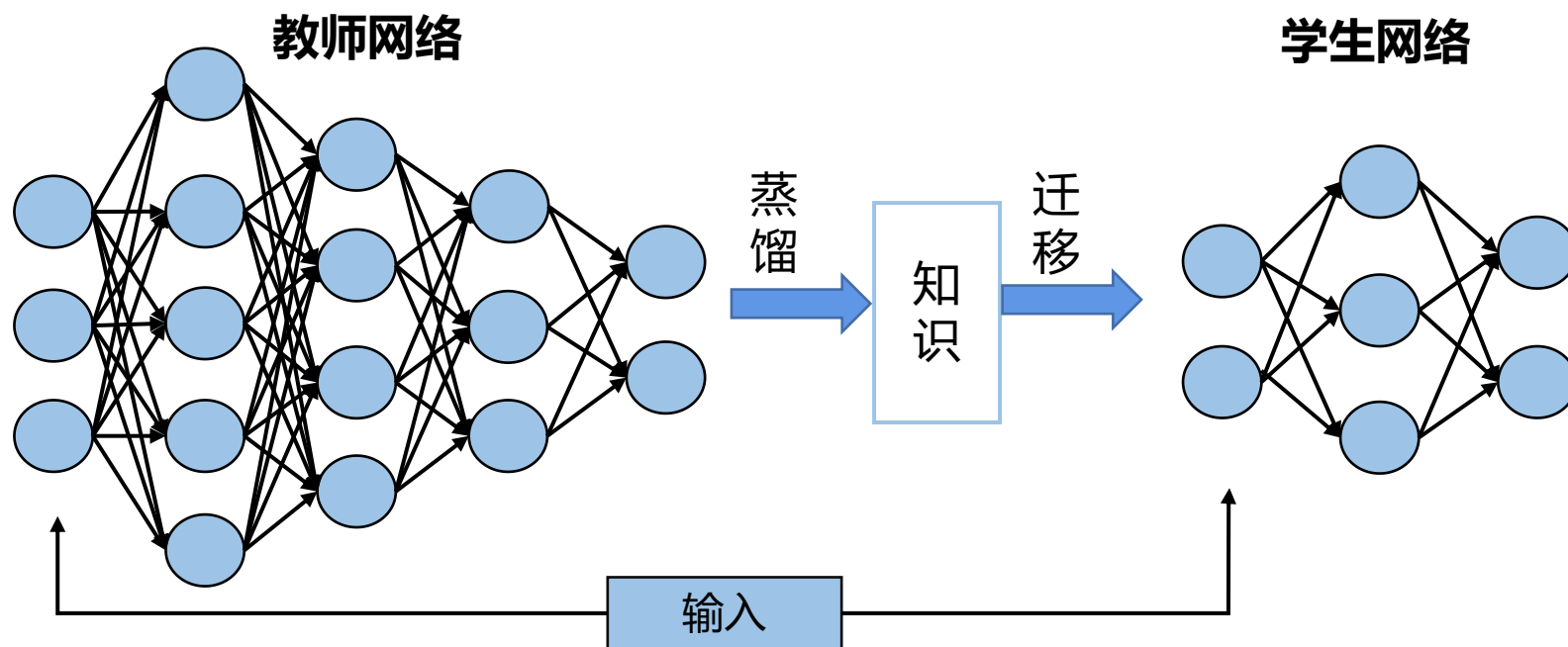
$$A_{m \times n} \approx U_{m \times k} D_{k \times k} V_{k \times n}^T = X_{m \times k} V_{k \times n}^T$$

模型参数量从 $m \times n$ 转化为 $k \times (m + n)$ ，当 $k$ 足够小时，模型参数量得到明显的下降。  
例如，当 $m = 10, n = 10, k = 2$ 时，分解后参数量变为原来的40%。



# 知识蒸馏

- 知识蒸馏是从知识传输演变而来的，它的目标是生成一个和大模型性能相当的小模型
- 大模型往往复杂、学习能力强，被称为**教师模型**，是知识的输出者；  
小模型往往参数量小、学习能力弱，被称为**学生模型**，是知识的接受者
- 实现方式是训练一个学生网络去模仿教师网络的行为



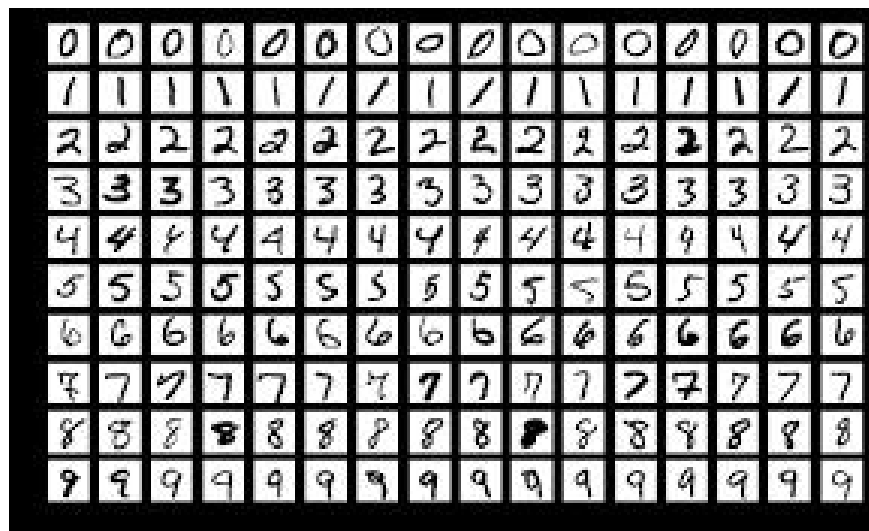


# 知识蒸馏

- 经典之作 Distilling the Knowledge in a Neural Network. Hinton G, Vinyals O, Dean J.
- 机器学习最本质的目的是训练出一个泛化能力强的模型。因为不可能收集到所有数据，因此通常，模型的训练目标是在有限的训练数据集上建模。
- 知识蒸馏的思想是，在已经有一个泛化能力强的教师网络后，可以利用教师网络来蒸馏训练学生网络，直接让学生网络去学习教师网络的泛化能力。
- **高效的迁移泛化能力方法**：使用softmax层输出的类别概率来作为 “soft target”

# 知识蒸馏为什么有效

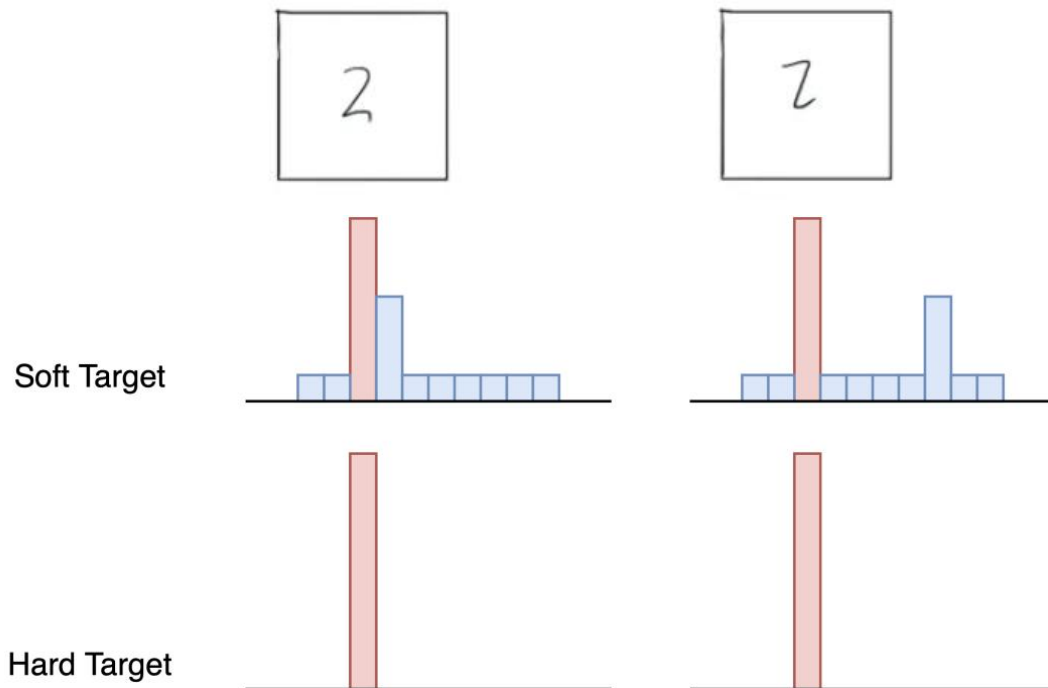
- 传统的训练 (hard targets) : 对ground truth (类别标签) 求极大似然
- 知识蒸馏的训练 (soft targets) : 用教师模型的类别概率作为soft targets
- Softmax层输出的类别概率信息包含了大量的信息, 比如某些负标签的概率远远大于其他负标签, 蕴含的信息量更大



MNIST手写数字识别数据集

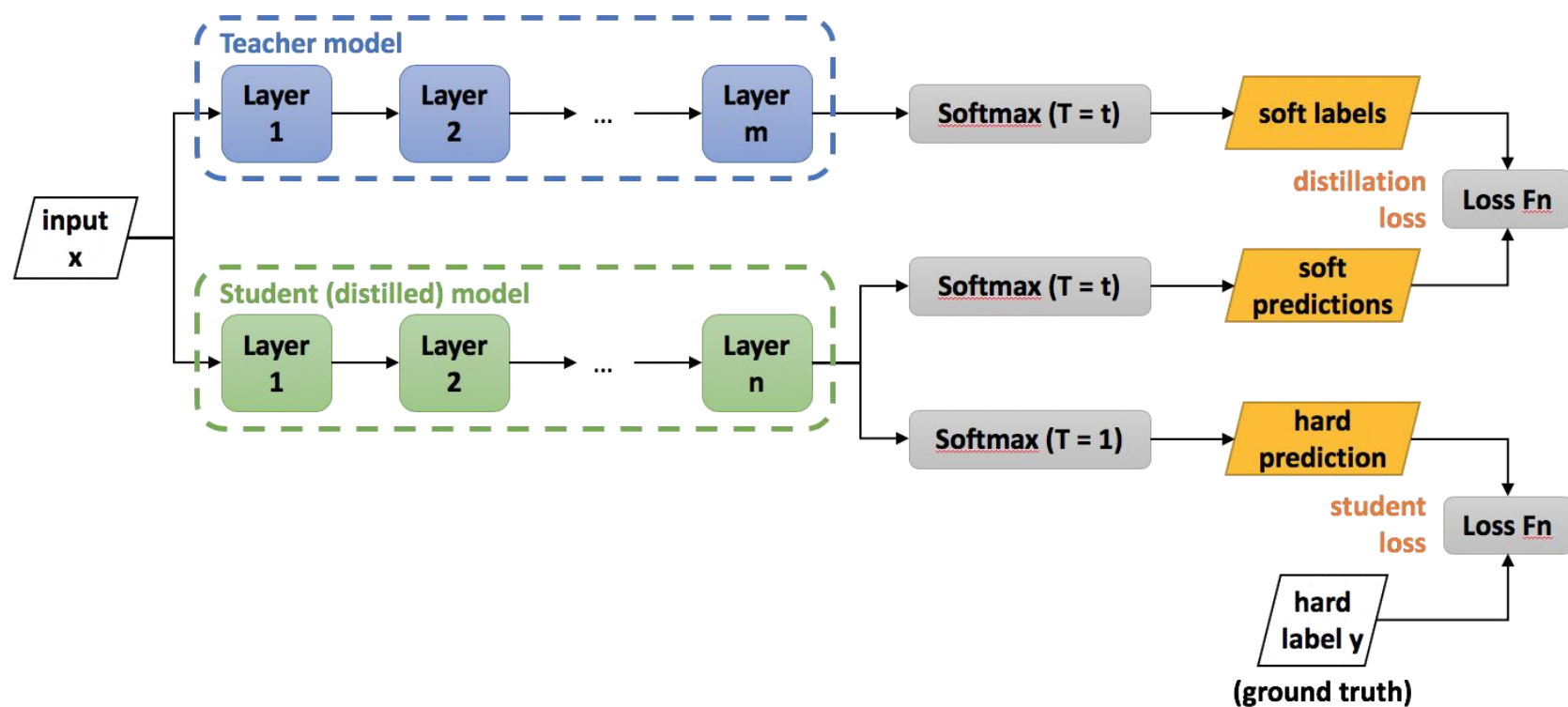
# 知识蒸馏为什么有效

- 左边的“2”形似“3”，右边的“2”形似“7”，对应的hard target都是类别2
- 但是soft target类别概率是不同的，左边“2” softmax输出值中“3”对应的概率次高，右边“2” softmax输出值中“7”对应的概率次高
- 因此，soft target比hard target蕴含的信息更多，且soft target分布的熵相对高时，其soft target蕴含的知识越丰富



# 知识蒸馏训练过程

- 第一步：使用hard targets（类别标签）训练教师网络
- 第二步：在高温T下，蒸馏教师网络的知识到学生网络
  - 温度T下的softmax函数  $q_i = \frac{\exp(z_i/T)}{\sum_j \exp(z_j/T)}$



# 知识蒸馏训练过程

- 第二步：在高温 $T$ 下，蒸馏教师网络的知识到学生网络

目标函数  $L = \alpha L_{soft} + \beta L_{hard}$

$$L_{soft} = - \sum_j^N p_j^T \log(q_i^T), \text{ 其中 } p_i^T = \frac{\exp(v_i/T)}{\sum_k^N \exp(v_k/T)}, \quad q_i^T = \frac{\exp(z_i/T)}{\sum_k^N \exp(z_k/T)}$$

$$L_{hard} = - \sum_j^N c_j \log(q_i^1), \text{ 其中 } q_i^1 = \frac{\exp(z_i)}{\sum_k^N \exp(z_k)}$$

$p_i^T$ : 教师网络的softmax( $T=t$ )输出值,  $q_i^T$ : 学生网络的softmax( $T=t$ )输出值

- 教师网络也有一定的错误率,  $L_{hard}$ 损失函数可以有效降低错误被传播给学生网络的可能性

# 神经网络压缩方法

压缩方法	方法描述	缺点
参数共享	减少对性能不敏感的冗余参数	<ol style="list-style-type: none"><li>1. 低精度量化特别是二值化的精度损失比较大</li><li>2. 更难以收敛，参数不灵活</li></ol>
低秩分解	使用矩阵/张量分解去估计有信息量的参数	<ol style="list-style-type: none"><li>1. 分解的操作不易执行，其计算消耗是昂贵的</li><li>2. 目前的算法是逐层进行的，因此不能全局压缩</li><li>3. 分解后，模型的收敛难度增加</li></ol>
知识蒸馏	通过从大模型中蒸馏知识训练一个紧致的神经网络	<ol style="list-style-type: none"><li>1. 依赖教师模型的性能；</li><li>2. 参数量减少导致的精度损失比较明显。</li><li>3. 通常仅应用于分类模型</li></ol>

## 2.3 网络剪枝

神经网络剪枝的优势

结构化剪枝和非结构化剪枝

剪枝中的不同思路

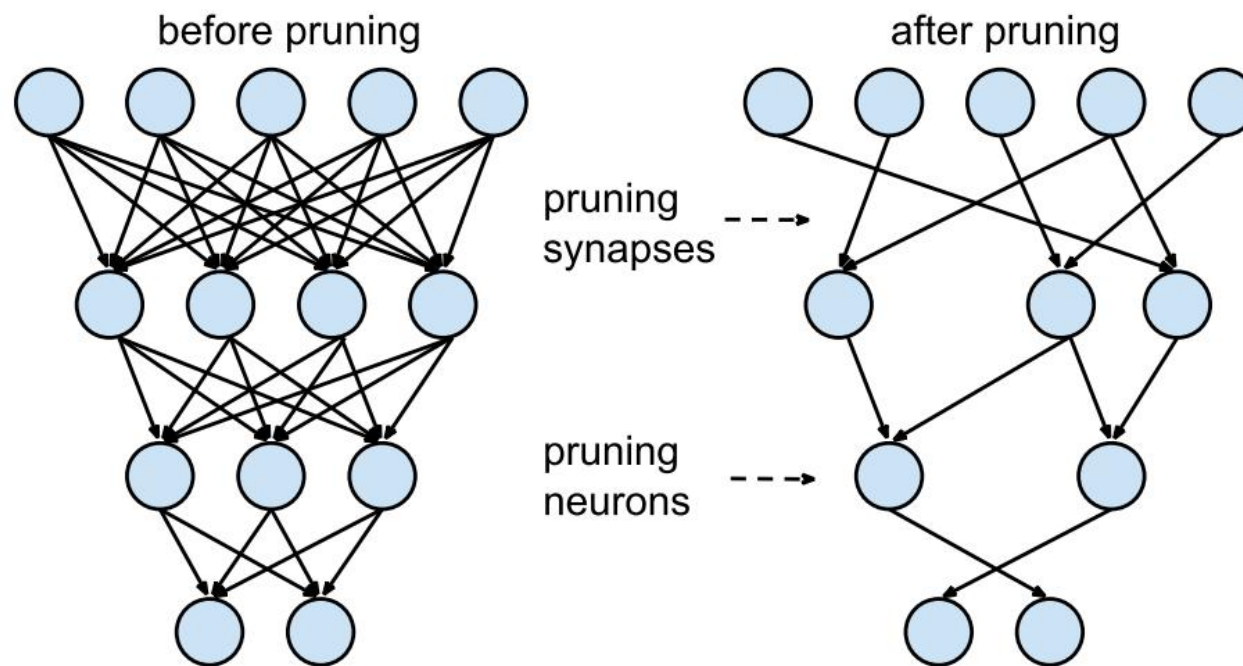
神经网络压缩

压缩方法

网络剪枝

# 神经网络的剪枝

**神经网络剪枝 (pruning)**：去掉神经元中对最终性能没有影响或者影响较小的连接





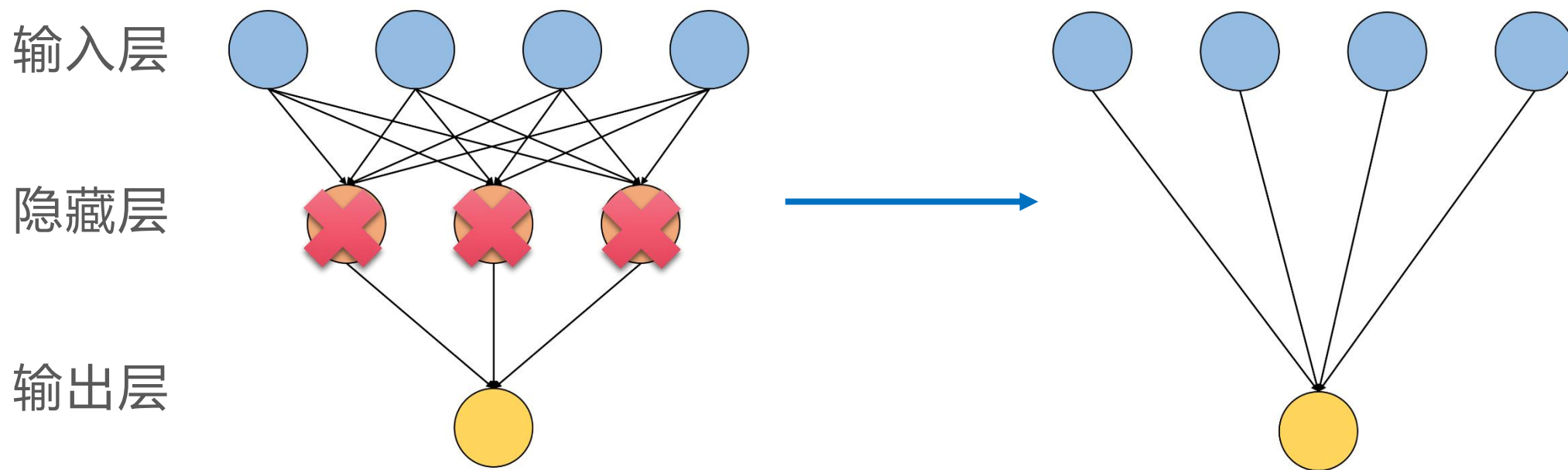
# 为什么选择神经网络的剪枝？

- 剪枝的**普适性是最高的**：不受网络结构或任务的限制，对任意网络都可以达到轻量化的效果
- 剪枝的缺陷通常和使用的剪枝方法相关。但总的来说，剪枝方法一般不会从头训练一个新的模型，都是从预训练模型出发。如果没有可用的预训练模型，那么剪枝任务需要从训练一个完整的模型开始。

# 结构化剪枝

根据剪枝粒度的不同，神经网络剪枝一般可以分成结构化和非结构化剪枝

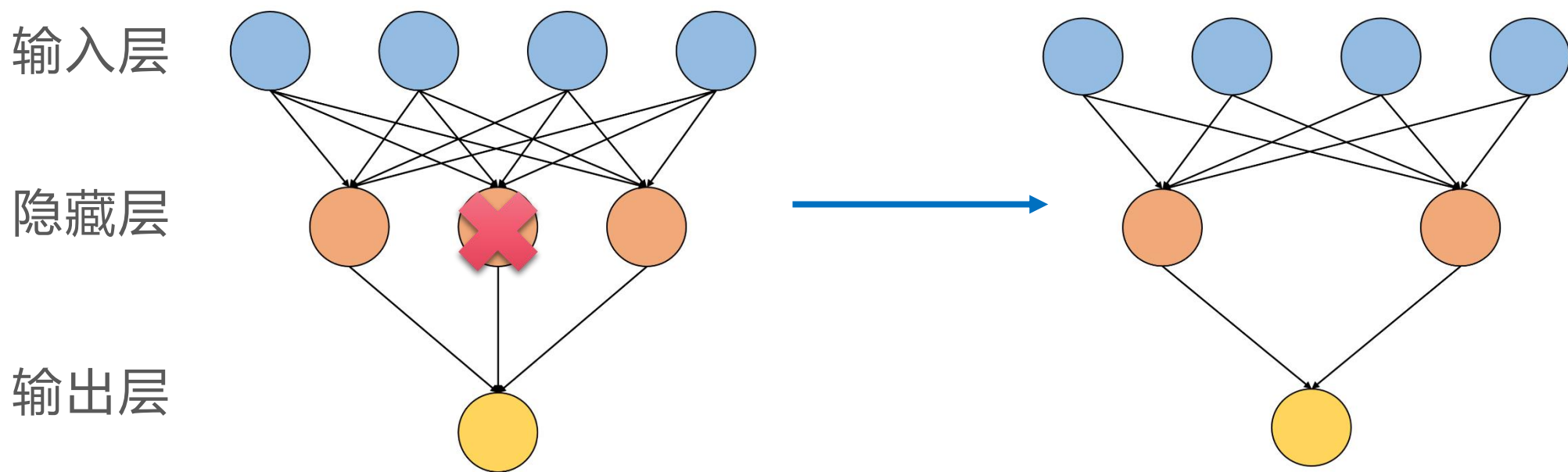
**结构化剪枝**是精确到层或通道(卷积神经网络)的剪枝策略



# 非结构化剪枝

根据剪枝粒度的不同，神经网络剪枝一般可以分成结构化和非结构化剪枝

**非结构化剪枝**是可以精确到神经元的剪枝策略



# 剪枝中的不同思路

- 剪枝算法的核心，其实是**重要度度量**，如何判断通道或神经元的重要程度：
- **范数信息**：参数的范数信息上，普遍的假设是更大的范数会具有更多的信息，倾向于修剪那些具有更小范数值的参数
- **稀疏性**：目标是修剪具有相似信息的通道，从而形成一种参数更少、特征提取能力相当的稀疏结构
- **重建误差**：基于重建的准则直接关注输出特征，其主要思想是最小化重建误差。因此，剪枝模型可以视为预训练模型的最佳近似

# 基于范数信息的剪枝

L1 Regularization

$$\text{Cost} = \sum_{i=0}^N (y_i - \sum_{j=0}^M x_{ij} W_j)^2 + \lambda \sum_{j=0}^M |W_j|$$

L1范数

L2 Regularization

$$\text{Cost} = \sum_{i=0}^N (y_i - \sum_{j=0}^M x_{ij} W_j)^2 + \lambda \sum_{j=0}^M W_j^2$$

L2范数

Loss function

Regularization  
Term

# 基于范数信息的剪枝

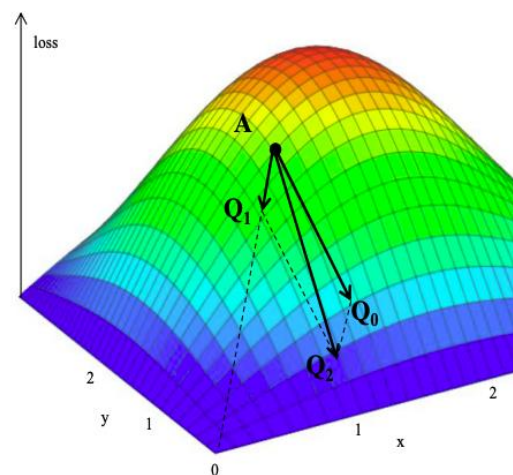
- L1 正则化：使原最优解的元素产生不同量的偏移，并使某些元素为0，从而产生稀疏性  
与之类似的还有Dropout的操作，可以随机的将一些连接设置为0，达到缓解过拟合的效果。这两种方法的思路，其实就是非结构化剪枝的思路

- 最新的方法中，会利用L1或L2正则化，结合其他的重要度度量来达到模型剪枝的效果，具体可以参考：

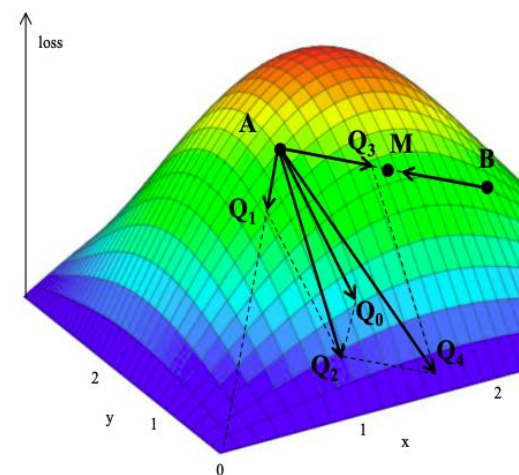
Han, S., Pool, J., Tran, J., & Dally, W. (2015). Learning both weights and connections for efficient neural network. Advances in neural information processing systems, 28.

# 基于稀疏性的剪枝

- 将输出通道数聚类成 $k$ 类，然后将每一类中的通道，按照C-SGD的方法，逐渐向一个点靠近，使得每一类中的所有通道，都变成相同的值。最后，只要在每一类中选取第一个，就可以达到剪枝的效果。
- 由于这个方法没有将通道归零，因此也不需要微调。



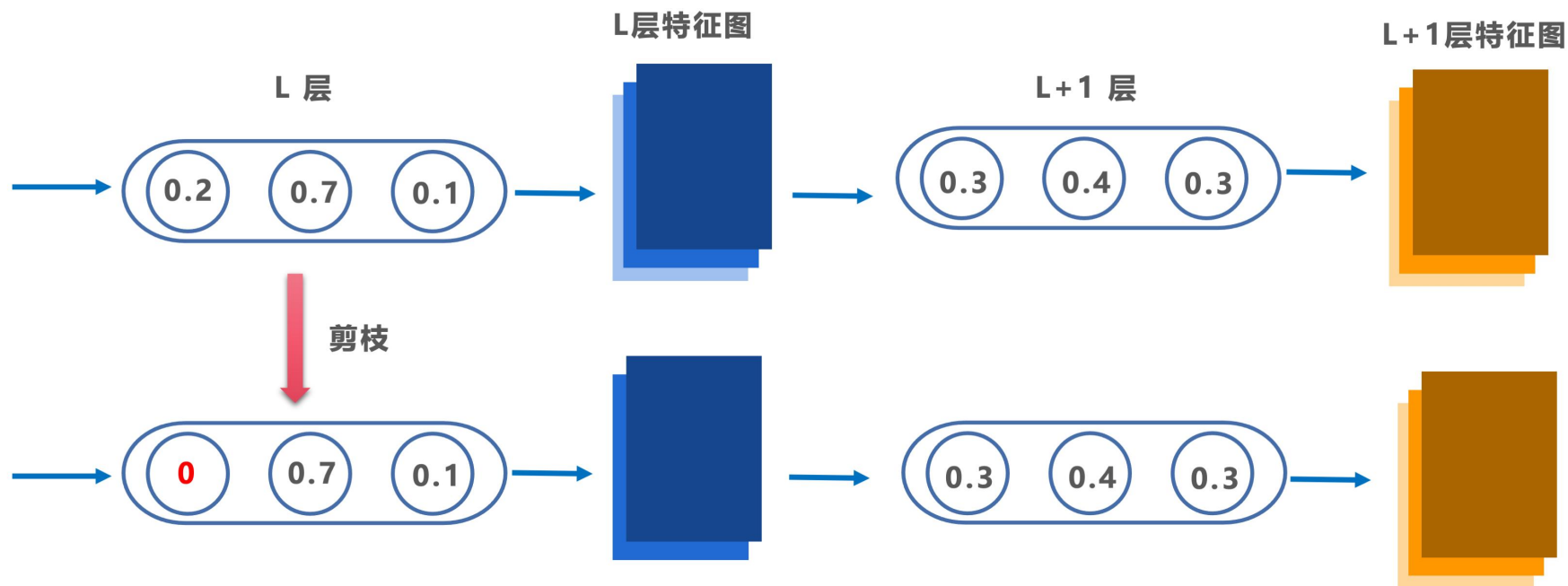
(a) Normal weight decay.



(b) Centripetal constraint.

# 基于重建误差的剪枝

- 如果能找到  $L$  层输出特征图的一个子集，使得  $L+1$  层的输出特征图发生仅可忽略的变化，则认为该子集以外的其他神经元是可以去掉的。
- 但找到这个子集是一个NP-hard问题，一种方案是使用贪心算法来找到次优解。





03

## 神经网络的可解释性

## 3.1 可解释性简述

可解释性的定义

可解释性的必要

可解释性简述

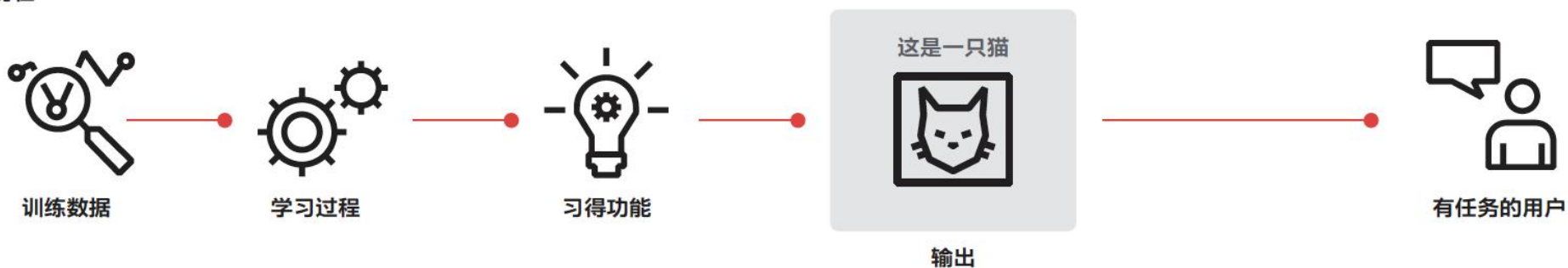
可解释性研究概述

可解释性的研究方向

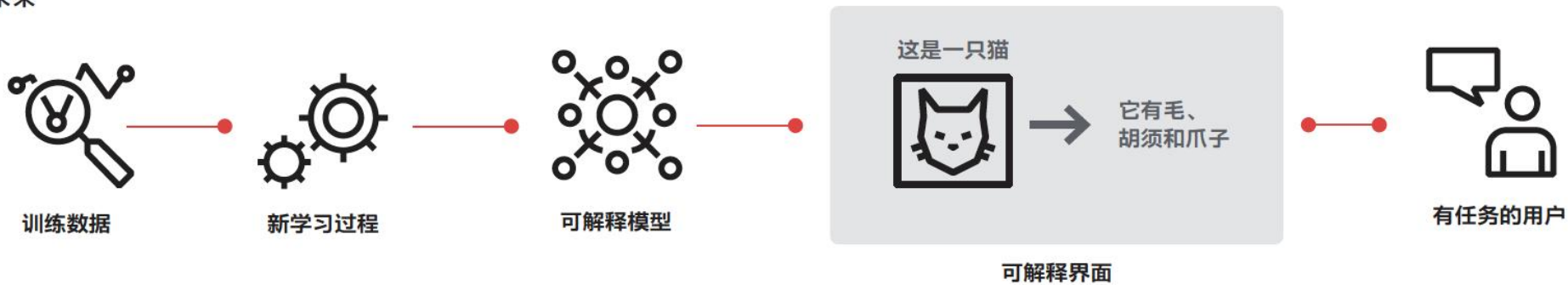
# 神经网络的可解释性

- 为什么是这个结论，为什么不是其他的结论？
- 什么时候模型会成功，什么时候会失败？
- 什么时候可以信任模型，如何去得知模型的错误并纠正错误？

现在



未来



# 神经网络的可解释性

➤ 一个较为人熟知的神经网络可解释性的定义：

“Interpretability of DNN is the ability to provide **explanations** in **understandable terms** to a human.” (F Doshi-Velez & B Kim, 2017)

➤ Explanations: 指需要某种语言来描述和注解。理想情况下最好的解释是严谨的数学符号-逻辑规则；但多数情况并不追求完全的解释。

➤ Understandable terms: 指构成解释的基本单元，不同的领域需要不同的领域术语来组成解释的基本单元，如自然语言处理中的单词，视觉领域中的图像块。

## 3.2 可解释性研究概述

可解释性的定义

可解释性的必要

可解释性简述

**可解释性研究概述**

可解释性的研究方向

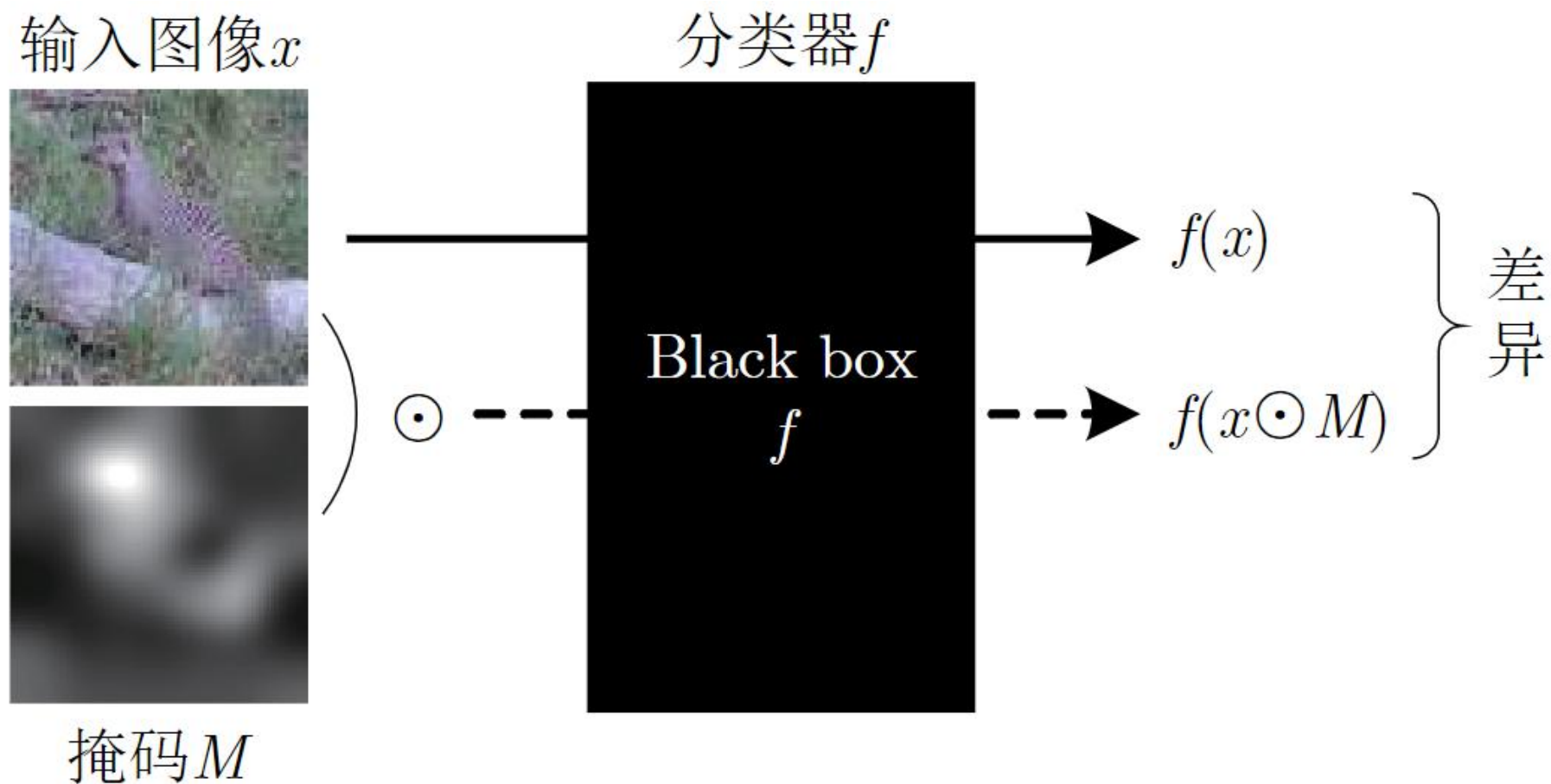
# 基于扰动的方法

基于扰动的方法将神经网络模型看作一种因果过程，通过修改输入（原因）观察输出（结果）的变化情况，从而得到被修改的输入对输出的影响大小。

考虑输入图像  $x \in R^d$ ，分类器  $f$ ，输出结果为  $f(x) \in R^k$ ，其中类别  $c$  的softmax分数为  $f^c(x)$ ，扰动方法使用删除、遮挡等方式处理输入图像，观察指定类别  $c$  的  $f^c(x)$  变化。

若分数下降较大，即被处理区域（像素/图像块）对  $f^c(x)$  的影响较大，重要性也较大。

# 基于扰动的方法



基于扰动的方法原理图

# 基于扰动的方法

## 基于扰动的方法分类：

- 简单扰动
- 有意义的扰动
- 生成式扰动

**简单扰动**使用固定或者随机的掩码去依次遮挡图像的各区域，观察遮挡后的预测结果，分数下降越大表明此时被遮挡区域对于该类别越重要，从而根据重要程度生成基于像素块的显著图。

其中掩码的填充像素一般为灰值或者随机值，不具有具体的意义。



# 基于扰动的方法

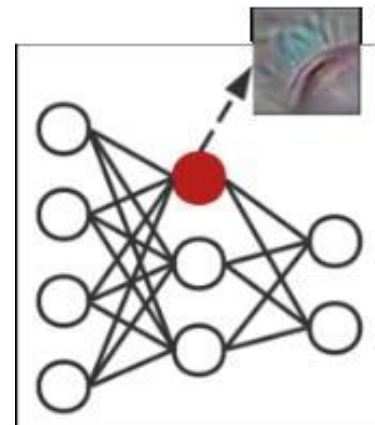
基于扰动的方法：

- 适用于黑盒模型，不止是CV任务
- 无需接触网络权重和中间特征值
- 只有结果一致性，没有过程一致性
- 依赖于尝试性方法，缺乏理论支持

# 理想样本

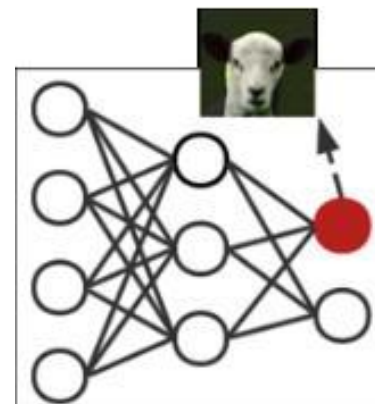
## ➤ 理想样本：

理想样本的解释方法旨在展示神经网络中的神经元学习到的特征，即找到能使指定神经元达到最大激活值的理想样本。



## ➤ 真实样例：

真实样例的解释方法是指网络单元从输入样本中寻找一个或一组样本，使得网络单元的激活程度最高，以这类输入样本作为该网络单元感兴趣样本的代表。



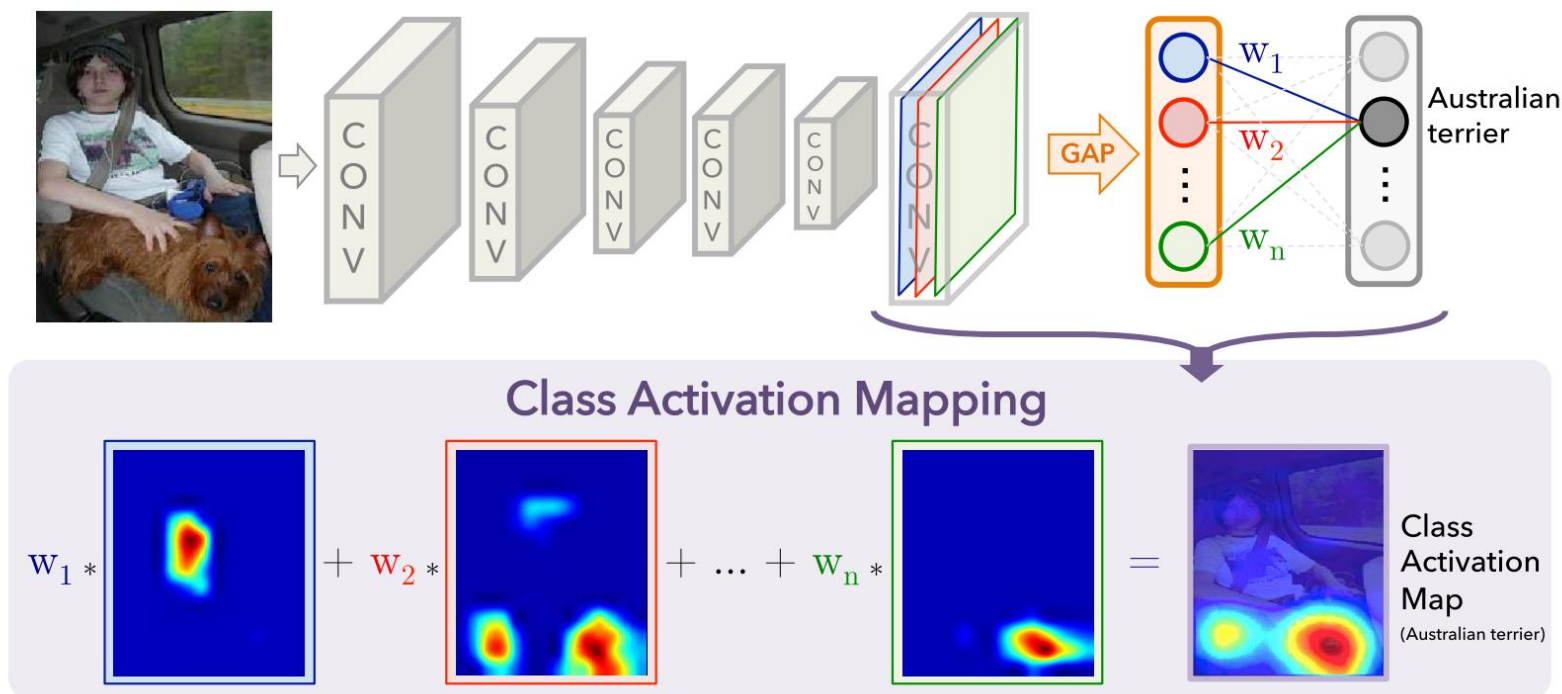
# 类激活映射方法

类激活映射通过生成类激活图 (Class activation map, CAM) 来可视化 CNN的关注区域，类激活图使用区域级的特征高亮方式，以突出与特定类别最相关的区域。



# 类激活映射方法

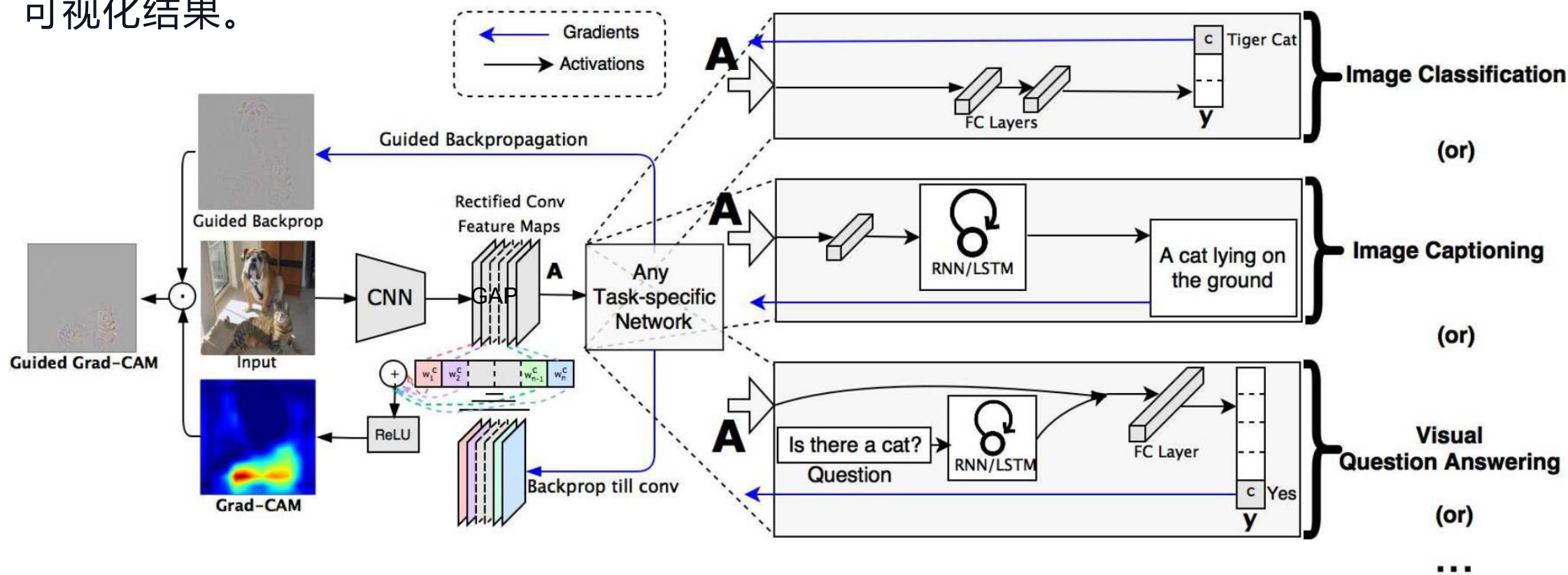
- CNN在的卷积层有标记检测到物体的位置的能力，但这种定位能力因为使用全连接层而丢失。CAM中使用Global Average Pooling (GAP)代替全连接层，再额外训练一个线性分类层，从而保留了卷积层的定位能力。
- 将图像输入模型得到类激活图  $H^c = \sum_k w_k^c A_k^L$ ，其中  $w_k^c$  为第c个类别的连接权重， $A_k^L$ 为最高层特征图（第L层）的第k个通道的特征图。





# Grad-CAM

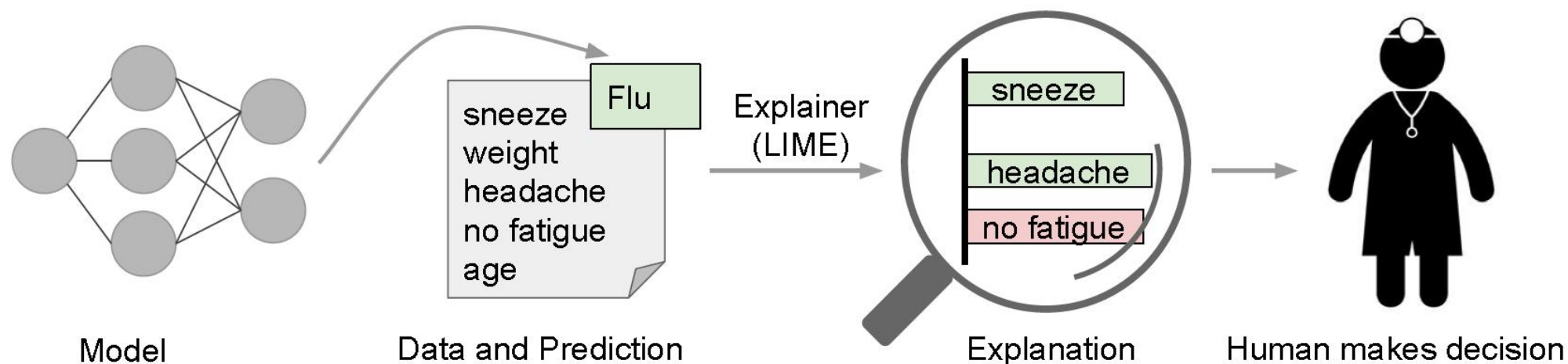
- Grad-CAM里特征图的权重由梯度计算而非训练得到。结合Guided Backpropagation, 生成既能够显示类别相关区域又能够展示这些区域细节的可视化结果。



# 模型无关的解释方法

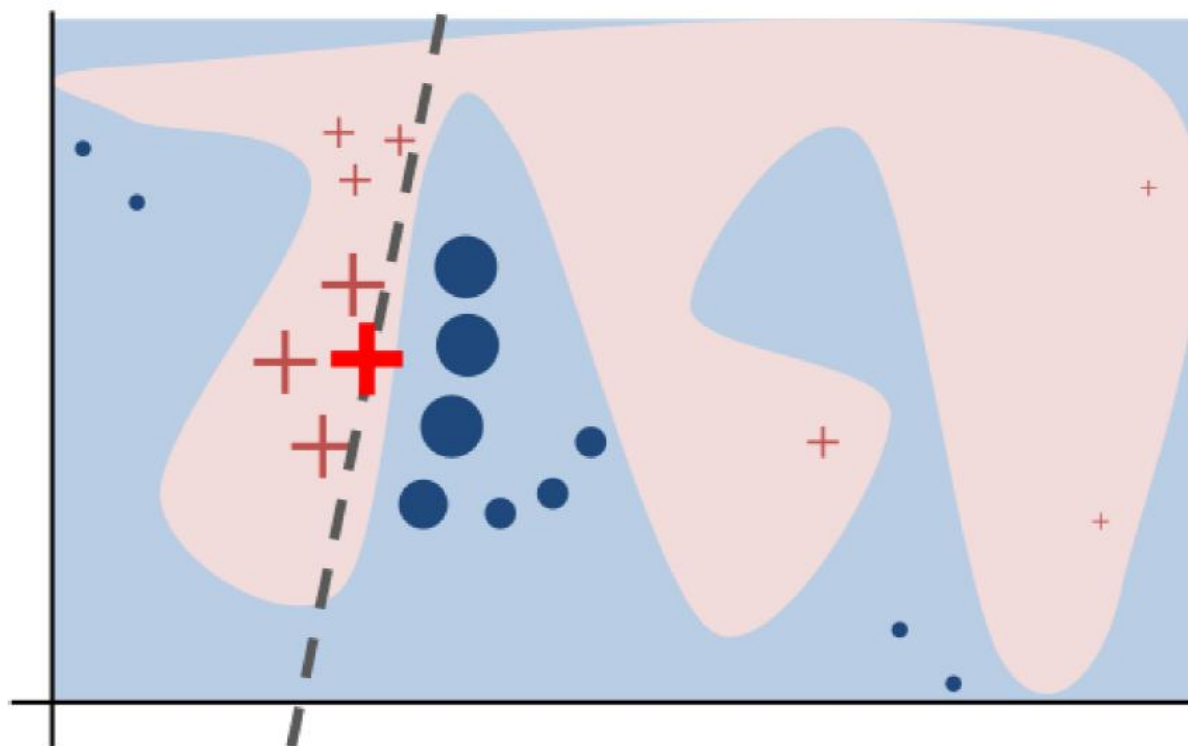
- Locally Interpretable Model-agnostic Explainer (LIME)

思路：通过构造可解释模型，拟合**任意分类器**对某个样本的输出，从而发现哪些特征导致了这样的输出。



# 模型无关的解释方法

- 在 $x$ 的附近, 可以用线性模型 $g$ 近似模型 $f$ 的决策边界, 并通过 $x$ 附近的采样点解释 $x$ 。
- LIME 通过扰动实例 $x$ 进行采样, 输入原模型得到标签, 训练稀疏线性回归模型解释实例。



## 3.3 可解释性研究方向

可解释性简述

可解释性研究概述

可解释性的研究方向



# 神经网络可解释性研究方向

## (1) 探索神经网络原理：

- 基于理想样本的解释旨在发现网络单元的学习信息，如深度神经网络的神经元；
- 基于真实样例的解释通过识别有代表性的样本来解释模型，有助于更好地理解数据集以及其对模型的影响；
- 基于单一输入的解释提供有关单个预测结果的解释信息，例如通过显著图可视化哪些像素与模型最相关以达到其决策；
- 基于多个输入的解释提供对模型行为更为一般的理解，如识别不同的预测策略等。

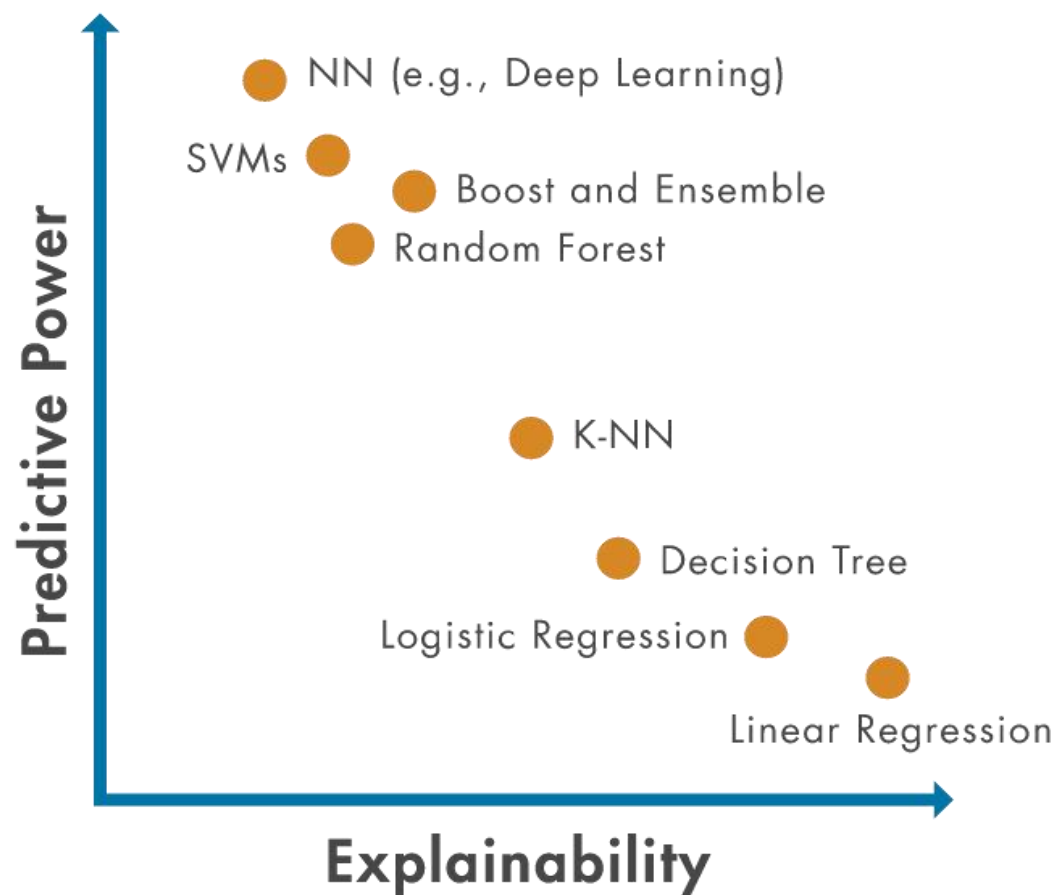
# 神经网络可解释性研究方向

## (2) 优化神经网络结构：

- 结合解释算法与正则化对模型进行改进；
- 根据解释算法发现的模型的内部信息，进行模型压缩和修剪；
- 通过可视化对模型进行诊断，构建更优的模型架构；
- 对模型错误预测情况提供解释，进而深入了解模型的故障原因；
- 识别数据集偏差为模型泛化提供帮助，使模型适用于更多任务。

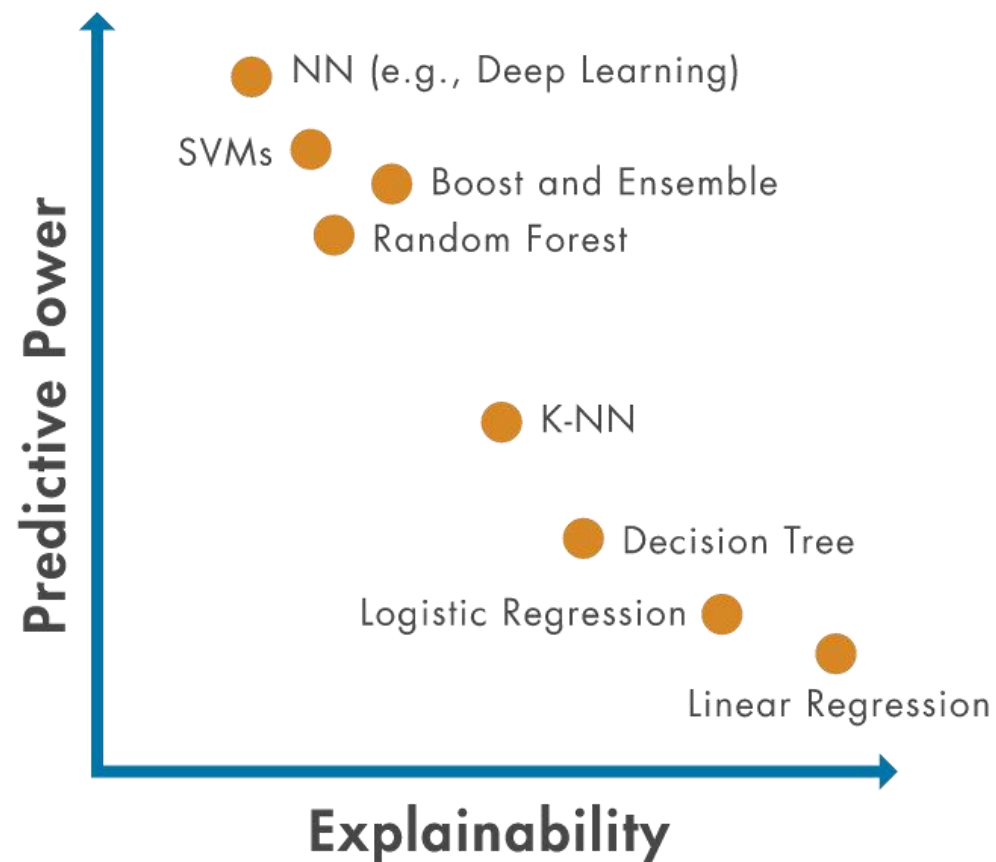
# 神经网络可解释性面临的挑战

- (1) 通常情况下，模型的可解释性与预测能力具有一定的冲突；
- (2) 缺乏统一评估标准；
- (3) 解释算法正确性无法保证。



# 神经网络的可解释性

- (1) 通常情况下，模型的可解释性与预测能力具有一定的冲突；
- (2) 缺乏统一评估标准；
- (3) 解释算法正确性无法保证。



# Q&A

Questions and Answers