AI 基础知识

AI 主要包含四个领域：自然语言处理（NLP）、机器学习（ML）、深度学习（DL）、计算机视觉（CV）

前置领域解析

NLP（文字处理基础）

NLP：即自然语言处理（Natuarl Language Processing），*主要研究人类语言活动中，信息成分的发现、提取、存储、加工与传输*，在 AI 模型中，将一句话分为很多个 token 就是由 NLP 完成的。常见的**中文分词工具**包括 jieba、FoolNLTK、HanLP、NLPIR、THULAC、LTP 等

Token：文本的最小单元、可以是一个词，也可以是一个字

Token 的处理分为按词分割和按字分割

按词分割时，结果如下：

```python
"hello world" = ["hello","world"] # 2Token
"你好，AI" = ["你好","，","AI"] # 3Token
```

按字分割时，结果如下：

```python
"hello world" = ["h","e","l","l","o","w","o","r","l","d"] # 10Token
"你好，AI" = ["你","好","，","A","I"] # 5Token
```

该领域的终极目标：**强人工智能**

1. **弱人工智能**：建立一个足够精确的语言数学模型使计算机通过编程来完成自然语言的相关任务。如：听、读、写、说，释义，翻译，回答问题等
2. **强人工智能**：让用户能通过自然语言与计算机自由对话

ML（学习记忆支撑）

机器学习的类型有如下几种：

1. **监督学习（Supervised Learning）**：使用带标签的数据进行训练，模型学习输入到输出的映射关系

   人工标记*数据集*，提供足够的训练资料， 比如从赵本山的小品中提取足够的语料进行训练，便可以获得一个会说小品的 AI

2. **无监督学习（Unsupervised Learning）**：使用不带标签的数据进行训练，模型寻找数据中的模式或结构

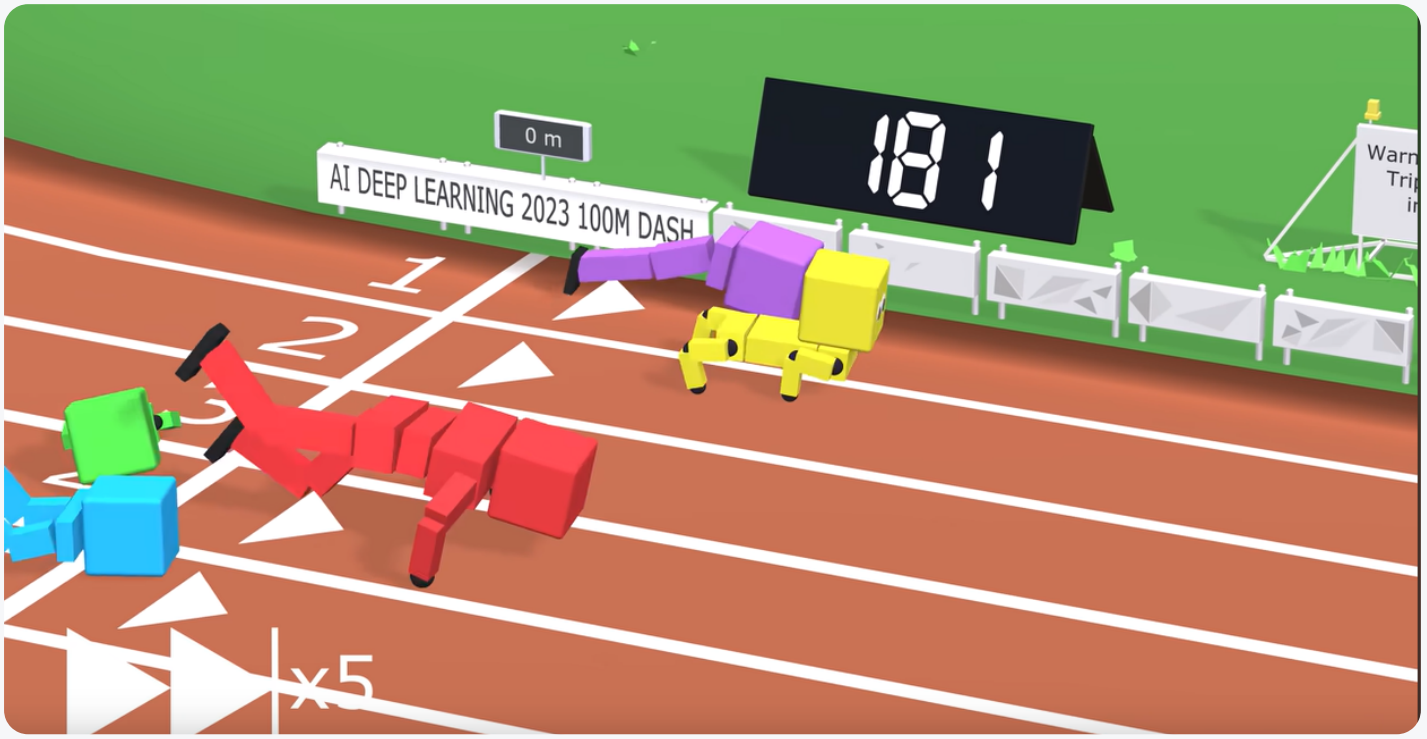
   不标记任何*数据集*，让 AI 找出其中的规律，AI 代替人类探索可能的蛋白质结构属于此类

3. **半监督学习（Semi-Supervised Learning）**：结合少量带标签的数据和大量不带标签的数据进行训练。

   介于监督学习和无监督学习之间，*数据集*有少量标注，但人类标记的数据集将为机器学习提供大致方向

4. **强化学习（Reinforcement Learning, RL）**：通过与环境的互动学习策略，以最大化累积奖励

   游戏 AI、机器人行走、机器人走迷宫属于此类

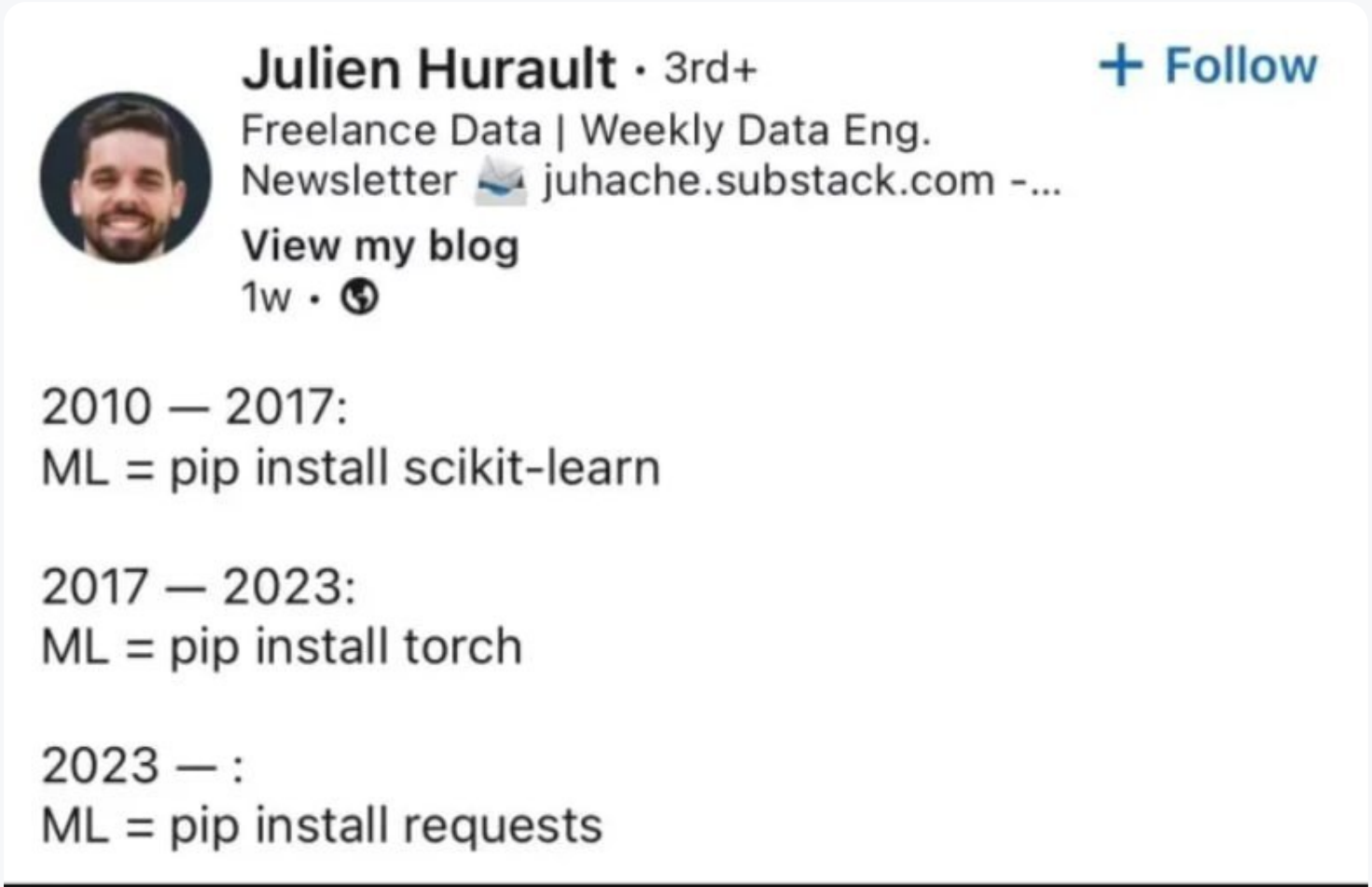
   

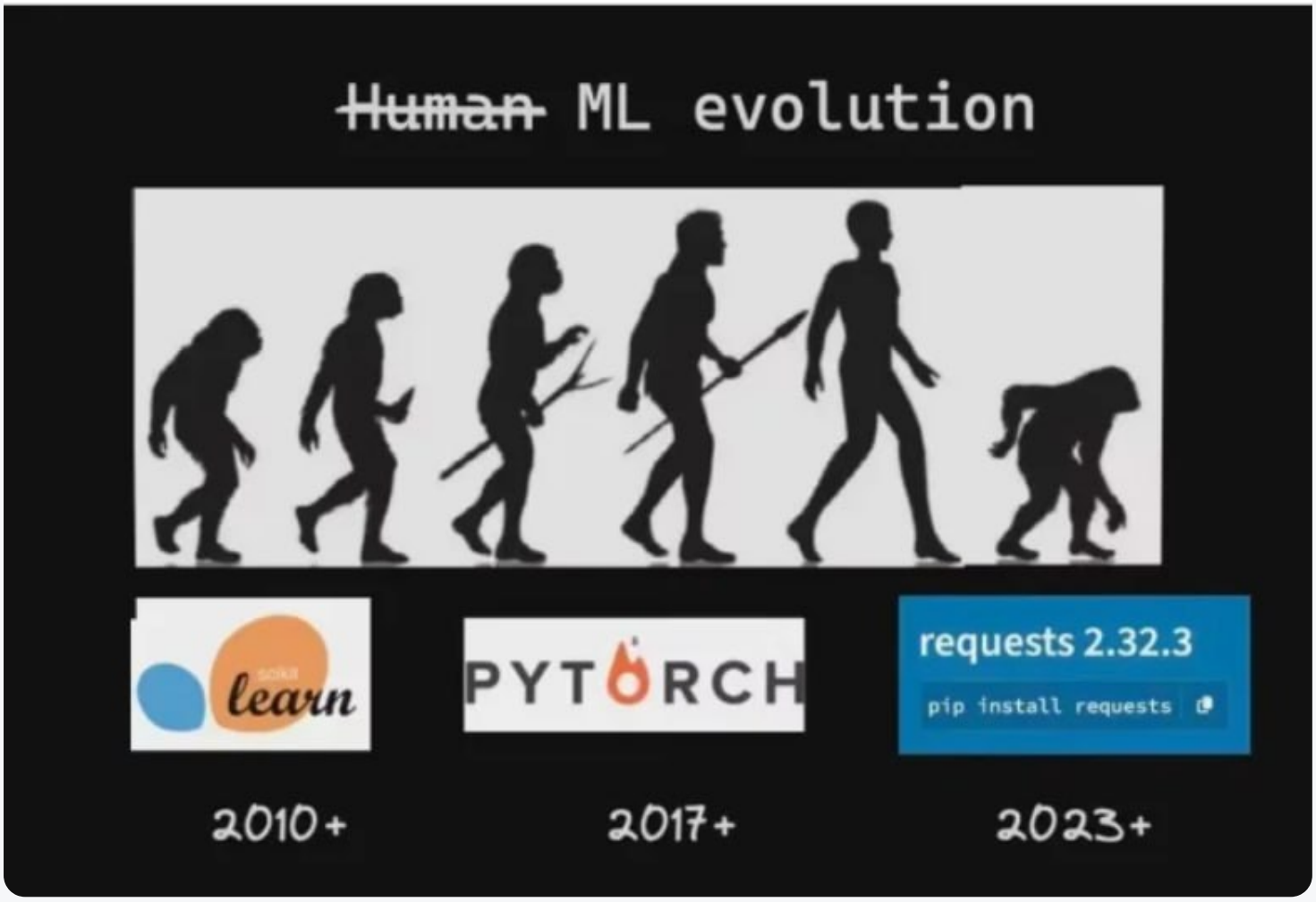
   很久以前看过的视频，通过强化学习让 AI 自己探索走路方式，当时对这个印象很深刻

DL（深度学习、机器学习的加强版）

从属于机器学习，暴力理解就是 pytorch

在 openAI 的 ChatGPT3.0 铺开以后，深度学习所需标准下放，于是就有了下面这张图：





注：requests 是 python 最常用的请求库，不适用 AI 训练和学习

> **为什么是 ChatGPT3.0？**
>
> ChatGPT3.0 发布是 AI 技术在国外大规模铺开的**重要里程碑**
>
> 更早期的都是一小群人在玩图像生成、文字生成。如 2022 年，NovelAI 在国内有一定程度的传播，但生成的图像普遍存在严重问题（人们对 AI 生成的都是残次品这样的偏见也是在那个时期出现的）
>
> 该模型的出现降低了人们对 AI 生成内容的偏见，也是此时 ChatGPT 开始大规模传播

CV（部分非结构化数据处理）

视频有各种各样的编码，MPEG、H.26X 等等，图片也是如此。AI 和人一样，看不了二进制形态的视频和图片，所以需要解码后进行**特征提取**、**目标检测**、**行为识别**等一系列处理，才能理解内容。

openCV 是最常见的计算机视觉库。

名词解释

结构化数据：二维数据表，有严格组织结构的二维数据，如数据库、csv

非结构化数据：各类 office 文件、图片、视频等非二维数据表

半结构化数据：机器可读，但非二维数据表，如 json toml xml log 等

AI 解析

参数

AI 参数模型（或权重）是用来描述神经网络中每个节点的数值

B 是 billion（十亿）的缩写，70B 便表示该模型包含 700 亿以上的参数

量化

量化是指将模型参数和计算方式从高精度转换为低精度，以减少模型的存储和计算需求

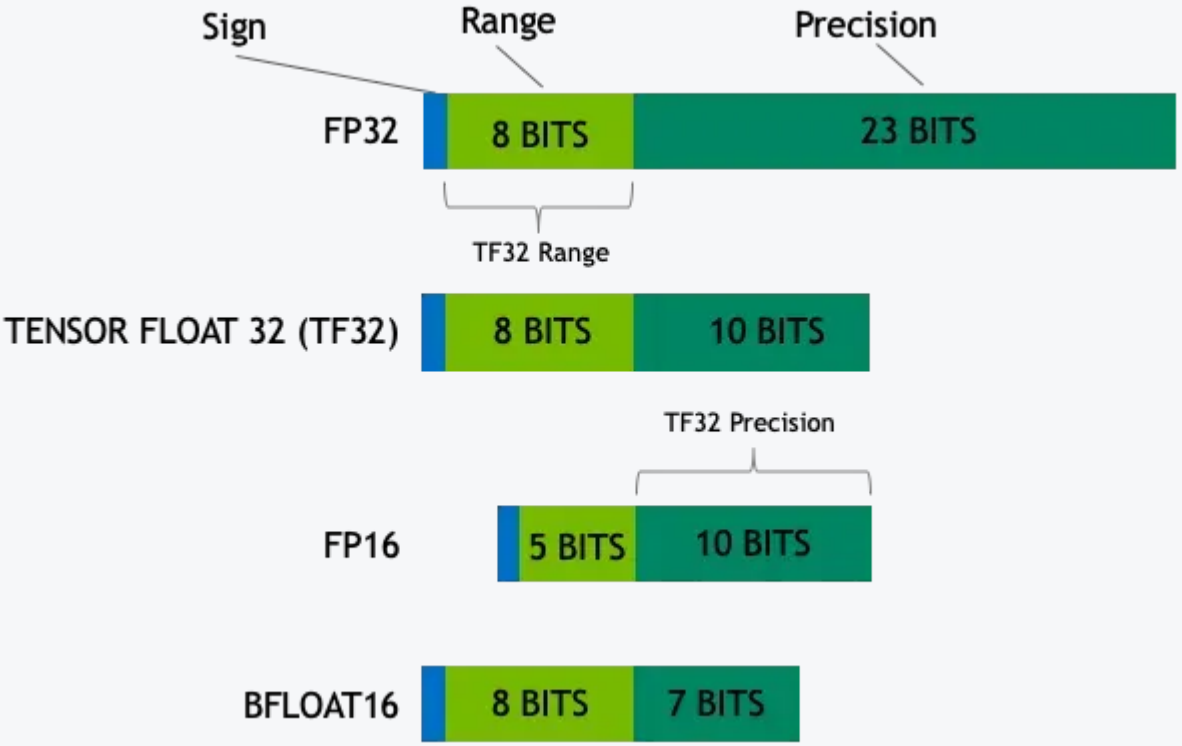
量化涉及计算机底层存储数据的方式：**整形和浮点数**

每个参数都是通过整形或浮点数加载到显存中的，因此**要运行模型，首先需要将模型装载到显存中**

量化相关数据结构

INT4、INT8、FP16、BF16、TF32、FP32 等都是用于深度学习的数值格式，它们和常规的整形、浮点数类似，但又有不同

下图是各类浮点数的量化：



- Sign 存储正负，占用 1 个比特
- Range 存储数值范围，拥有的比特越大能表示的范围越大
- Precision 存储精度，拥有的比特越多越精准

  FP16 使用 16 位比特来表示一个参数，这 16 比特是要实打实装进显存里面的，假设需要装载 1.5B 的模型，那么就是 15 亿个这样的单元装进显存，并需要配套的 cpu、内存和硬盘

  由图类推可知： FP32 一个单元占用 32 bit，BFLOAT16 一个单元占用 16 bit，TF32 比较特殊，一个单元占用 19 个 bit，介于 2 byte 和 3 byte 中间

降低精度的优缺点

不同的显存占用影响性能

除浮点数之外还有 Int8 Int4 的量化方式，**占用更小，但模型精度也更差**

因此将 FP32 浮点数参数量化为 Int8，可以**减少约四分之三的存储空间**（每个参数占用从 32 bit 降低至 8 bit）

人吃不饱饭会不干活，AI 吃不饱虽然不会拒绝干活，但幻觉、复读、答非所问出现的概率那

就不得而知了

非结构化数据的解析能力需要一定规模的模型才能支持

精度与参数如何取舍？

参数数量决定了思考问题的**广度**，量化**精度**决定了思考问题的**深度**，但 1 字节 / 参数以下（也就是比 Int4 还差的量化）的量化会导致非常明显的性能下降

llama.cpp 输出模型的格式

对应于不同类型的量化，有不同的表达方式，其遵循如下规律：

“q”+ 用于存储权重的位数（精度）+ 特定变体

在上面内容的基础上，可以引出更加细致的优化方案

更为详细的量化方法

> `X` 是量化位数，也就是参数占用的比特数（和 fp16、fp32 后面的数字是一个东西）
>
> `Y` 表示对称/非对称量化，0 表示对称量化，1 表示非对称量化
>
> 带 **_K** 后缀的，属于 K 系列量化方法

llama.cpp 提供如下量化方式：

- `q2_k`：将 Q4_K 用于 attention.vw 和 feed_forward.w2 张量，Q2_K用于其他张量。
- `q3_k_l`：将 Q5_K 用于 attention.wv、attention.wo 和 feed_forward.w2 张量，否则 Q3_K
- `q3_k_m`：将 Q4_K 用于 attention.wv、attention.wo 和 feed_forward.w2 张量，否则 Q3_K
- `q3_k_s`：将Q3_K用于所有张量
- `q4_0`：原始量化方法，4 位。
- `q4_1`：精度高于q4_0但不如q5_0。但是，与 q5 模型相比，推理速度更快。
- `q4_k_m`：将 Q6_K 用于一半的 attention.wv 和 feed_forward.w2 张量，否则Q4_K
- `q4_k_s`：将Q4_K用于所有张量
- `q5_0`： 原始量化方法，5位。精度更高，资源使用率更高，推理速度更慢。
- `q5_1`：精度高于q5_0但不如q6_k。但是，与 q6 模型相比，推理速度更快。

- `q5_k_m`：将 Q6_K 用于一半的 attention.wv 和 feed_forward.w2 张量，否则Q5_K
- `q5_k_s`：将Q5_K用于所有张量
- `q6_k`：将Q8_K用于所有张量
- `q8_0`：与16位浮点数几乎无法区分。资源使用率高，速度慢。不建议大多数用户使用。

hugging face 提供了一整套量化方式，如下表所示（删除了已废弃的量化方式）：

类型	描述
F64	64 位标准 IEEE 754 双精度浮点数。
I64	64 位定宽整数。
F32	32 位标准 IEEE 754 单精度浮点数。
I32	32 位定宽整数。
F16	16 位标准 IEEE 754 半精度浮点数。
BF16	16 位缩短版的 32 位 IEEE 754 单精度浮点数。
I16	16 位定宽整数。
Q8_K	8 位量化，每个块有 256 个权重。仅用于量化中间结果。所有 2-6 位点积均为此量化类型实现。权重公式：w = q * block_scale。
I8	8 位定宽整数
Q6_K	6 位量化，超级块有 16 个块，每个块有 16 个权重。权重公式：w = q * block_scale（8 位），结果为 6.5625 位/权重。
Q5_K	5 位量化，超级块有 8 个块，每个块有 32 个权重。权重公式：w = q * block_scale（6 位）+ block_min（6 位），结果为 5.5 位/权重。
Q4_K	4 位量化，超级块有 8 个块，每个块有 32 个权重。权重公式：w = q * block_scale（6 位）+ block_min（6 位），结果为 4.5 位/权重。
Q3_K	3 位量化，超级块有 16 个块，每个块有 16 个权重。权重公式：w = q * block_scale（6 位），结果为 3.4375 位/权重。
Q2_K	2 位量化，超级块有 16 个块，每个块有 16 个权重。权重公式：w = q * block_scale（4 位）+ block_min（4 位），结果为 2.5625 位/权重。
IQ4_NL	4 位量化，有重要性矩阵，超级块有 256 个权重。
IQ4_XS	4 位量化，有重要性矩阵，超级块有 256 个权重。其大小为 4.25 位 / 权重。
IQ3_S	3 位量化，有重要性矩阵，超级块有 256 个权重。其大小为 3.44 位 / 权重。

类型	描述
IQ3_XXS	3 位量化，有重要性矩阵，超级块有 256 个权重。其大小为 3.06 位 / 权重。
IQ2_XXS	2 位量化，有重要性矩阵，超级块有 256 个权重。其大小为 2.06 位 / 权重。
IQ2_S	2 位量化，有重要性矩阵，超级块有 256 个权重。其大小为 2.50 位 / 权重。
IQ2_XS	2 位量化，有重要性矩阵，超级块有 256 个权重。其大小为 2.31 位 / 权重。
IQ1_S	1 位量化，有重要性矩阵，超级块有 256 个权重。其大小为 1.56 位 / 权重。
IQ1_M	1 位量化，有重要性矩阵，超级块有 256 个权重。其大小为 1.75 位 / 权重。

> **名词解释**
>
> 位 == 比特
>
> 权重 == 参数

类型前面的字母有如下含义：

- **IQ**：代表"Importance Quantization"（重要性量化），使用重要性矩阵来改进量化结果
- **Q**：代表"Quantization"（传统量化）

> 同样为量化， 重要性量化通过重要型矩阵优化模型量化
>
> **重要性矩阵（Importance Matrix）**：其能根据不同权重的重要性来调整量化的精度，可以达到不显著降低模型精度的情况下，对重要性较低的权重进行更高程度的量化，而对重要性较高的权重进行较低程度的量化

K 字母有如下含义：

- **K**：块（block），通常与块内的量化方法相关

  其特点为**层次化**，即将用户的复杂问题分解为小问题去分析，**适合在旧硬件、Mac 电脑或纯 CPU 推理的情况下对模型进行量化**

I K 都是量化优化，I 是通过优化参数组织方式、K 是通过参数分组进行优化，二者不可共存

F 字母有如下含义：

- **F**：浮点数（Floating-point）
- **BF**：脑浮点（Brain Floating-point），优化过的浮点数

S、N、L 字母有如下含义：

- **S**：小块（shrunken），小块量化

- **XS**：超小块（extra small block），超小块量化
- **XXS**：超超小块（extra extra small block），即更小块的量化
- **NL**：非线性（nonlinear）方法的量化
- **M**：中等块（medium）
- **L**：大块（large），大块量化

> 这几类都表示参数被**分块量化**了，区别在于参数块的大小，虽然优化了显存，其对调度要求更高，计算效率也存在一定下降

unsloth 提供了一些特殊量化的模型

模型名称	模型格式	模型大小（G）	预估显存（G）
DeepSeek-R1-UD-IQ1_S	gguf（分卷）	141	180
DeepSeek-R1-UD-IQ2_XXS	gguf（分卷）	197.5	250
DeepSeek-R1-Zero-Q2_K_XS	gguf（分卷）	223.9	280

AI 模型的存储方式

1. safetensors：高速型
   - 安全性高，无代码注入风险。加载速度快，支持内存映射，适合加载大型模型
   - **只能在 Hugging Face 上执行**
2. gguf：笨重型、高效型
   - 高效推理设计，支持多种量化方案，适合在资源受限设备上运行
   - 强制需求 llama.cpp、高精度量化占用大
3. ckpt：训练特化型
   - 保存模型参数和优化器状态，支持训练过程的中断与恢复
   - 有代码注入风险，存在**安全问题**
4. bin：通用型
   - 适配框架多、通用性强

- 需要自定义逻辑读取和保存

5. pth：开发型、训练特化型
    - 可直接对接 pytorch
    - 不能直接用于其他框架，存在**安全问题**

仅推理推荐使用 gguf 和 bin

gguf 模型文件结构示意图

模型选型与推荐配置

> 依据量化精度为 bf16，模型格式为 gguf，系统为信创系统进行估算

根据上面的配置，可以通过估算什么配置可以装下模型：

模型参数大小（B）	所需显存（G）	所需内存（G）	所需核心数（个）	所需存储（GB）	节点间高速网络（Gbps）	高性能网卡与配套交换机
671	1800	512	64	4000	200	需要
70	256	512	64	2000	0	不需要
32	128	512	32	1000	0	不需要
14	64	256	32	500	0	不需要
8	32	128	32	300	0	不需要
7	32	128	32	300	0	不需要
1.5	32	64	16	300	0	不需要

> 该估算仅供推理使用，无法完成训练
>
> 低于模型最低配置是无法部署的

其他影响因素

1. 网络速度影响集群推理效率
2. 显卡算力影响 token 输出速度
3. 存储影响模型的启动速度、模型参数大小也对存储需求有影响