

Jone Gorman AI 学习指南：从入门到 Agent、Physical AI 与 RSI

2026-10-19

目录

1. 认识计算机	2
2. 认识操作系统	2
3. 学习 Python	3
4. 管理开发环境	3
5. 从机器学习走向深度学习	3
6. 理解大语言模型	4
7. 强化学习：从基础到 PPO	5
8. 学习 Agent 与应用	6
9. RAG、记忆与评价	7
10. 多模态、生成模型与世界模型	7
11. Physical AI：从感知到物理行动	7
12. 科学机器学习与 AI for CAE	8
13. RSI：递归自我改进	9
14. 用项目把路线串起来	9
15. 把学习变成可持续的习惯	10

昂贵的课程、密集的新名词，以及不断出现的技术热点，很容易让刚开始接触 AI 的人感到焦虑。这份指南从一个预算有限、计算机基础薄弱的学习者出发，把“我要学 AI”拆成可以逐步完成的事情。

从认识电脑，到写出第一段 Python，再到复现模型和完成一个小项目，每一步都可以留下自己的代码与记录。课程是入口，动手实践才让这些知识真正连接起来。

这份路线以熟悉概念、建立动手能力为目标。先理解 AI、机器学习、深度学习和生成式模型之间的联系，再学习能够调用工具的 Agent、与物理环境交互的系统，以及如何评价和改进它们。不必把所有课程看完再开始项目。

先放好几个概念的位置：AI 是更大的研究领域；机器学习研究从数据或经验中学习的方法；深度学习主要使用多层神经网络；LLM 是语言建模的一类模型。强化学习是一种学习与决策框架，Agent 是围绕目标组织模型、工具与环境的系统，不能把它们都当成同一种模型。

- **从零起步**：依次完成第 1—5 节，能读写 Python、管理环境并解释一个小模型的训练结果。
- **做 Agent 应用**：在基础上学习第 6—9 节，以有引用的问答或可核对的工具任务为第一个项目。
- **走向物理与研究**：选择第 10—13 节的一个方向，再用第 14 节的项目建立完整记录。RSI 在这里指 Recursive Self-Improvement（递归自我改进）。

资料覆盖大学课程、YouTube 讲解、原始论文、作者代码、官方文档和研究博客。YouTube 入口优先来自作者、大学或机构的原始发布，并说明适用阶段和学习目标。视频用于建立直觉，论文用于核对证据，代码和文档用于落实方法。

1. 认识计算机

先理解计算机的基本组成，区分 CPU、内存、存储和 GPU，知道程序、文件以及硬件之间大致如何配合。这一阶段不需要先购买高性能设备，也不需要一次掌握所有计算机知识。

- 计算机入门视频

动手记录：查看自己设备的处理器、内存和磁盘容量，整理常用文件的位置。内存和磁盘存储是不同概念；安装软件占用的主要是存储空间。

2. 认识操作系统

从熟悉的 Windows 开始，逐渐了解操作系统、终端、目录和文件权限。在需要 Linux 的课程或项目里，再学习相应操作。双系统是原文提供的一条实践路线，也可以先在虚拟机或 WSL 中练习。

- 双系统安装教程
- Linux 基础操作

动手记录：在终端中完成查看当前目录、切换目录、创建文件夹和运行程序，并理解自己输入的命令做了什么。

3. 学习 Python

用一门语言建立编程基础，比同时追逐多个工具更容易开始。这里选择 Python，先学会变量、条件、循环、函数、常用数据结构、模块导入和基本报错排查，再进入机器学习代码。

- Python 与 VS Code 环境搭建
- Python 基础课程

动手记录：独立写一个读取文件、处理数据并输出结果的小脚本。跟着视频敲完之后，修改输入或增加一个功能，确认自己能够解释程序的行为。

4. 管理开发环境

项目之间往往需要不同的 Python 和依赖版本。Miniconda 提供较小的初始安装，可以按项目创建环境；随后使用 conda 或环境中的 pip 安装所需依赖。它与 Anaconda 的区别不需要用固定的体积倍数来判断，实际占用取决于安装内容。

- Miniconda 入门教程
- conda 官方环境管理文档

原文以 Python 3.10 为例创建第一个环境。这个版本用于展示操作方式，实际项目应匹配课程或依赖文件中的版本要求。

```
conda create -n aienv python=3.10
conda activate aienv
python --version
python -m pip --version
```

动手记录：确认终端、编辑器和安装依赖所使用的是同一个环境，记录项目的 Python 版本及依赖配置。

5. 从机器学习走向深度学习

进入 AI 之后，数学、代码和实验会一起出现。遇到暂时不理解的推导，可以先把一个例子运行起来，再回头连接公式、输入输出与代码实现。不同课程的节奏不同，可以交叉阅读，但不必强求短时间内看完所有内容。

YouTube：直觉、数学与代码

- **入门直觉：**3Blue1Brown 的 But what is a neural network?。先理解网络、损失和参数调整，学完画出一个两层网络的输入输出关系。
- **随用随补的数学：**3Blue1Brown 的 Vectors 与 The essence of calculus。分别连接向量与矩阵运算、导数与变化率；随后自己计算一个简单函数的梯度。
- **第一段训练代码：**Karpathy 的 Building micrograd，配合作者课程页。需要基本 Python 和导数概念，目标是理解计算图、反向传播与梯度下降。

- **系统概览：**MIT 6.S191 的 2025 年第一讲 与 课程及实验。适合有初步线性代数和微积分基础后建立全景，再挑一个实验运行。

训练之外，还要认识数据质量、训练 / 验证 / 测试划分、过拟合、损失函数与评价指标。一个小项目的起点可以是线性模型或简单 MLP；保留基线，才能判断增加复杂度是否值得。

课程与实践材料

- 李沐：动手学深度学习
- Abhishek Thakur: Approaching (Almost) Any Machine Learning Problem, 作者仓库
- AAAMLP 中文翻译
- RethinkFun 深度学习教程

Git、GitHub 与设备

写代码的同时学习版本管理：Git 用来记录修改，GitHub 可以托管和分享仓库。先用现有设备运行适合其规模的例子，再根据项目需求判断是否需要 GPU。原文中的装机视频作为硬件知识补充保留。

- GitHub 介绍
- Git 介绍
- 硬件茶谈装机教程

动手记录：完成一个有数据划分、评价指标和结果记录的模型实验，并将自己的实现提交到 Git 仓库。

6. 理解大语言模型

从 Transformer 开始，把注意力机制、张量形状和前向传播联系起来。逐行实现一个教学模型，再阅读 Llama 等架构，能够让论文中的模块不再只是名词。这里的学习顺序是路线安排，不是对技术新旧或性能的排名。

YouTube：从直觉到完整实现

- **先看结构：**3Blue1Brown 的 Transformers, the tech behind LLMs，用来建立 token、向量和注意力的视觉直觉。
- **再写模型：**Karpathy 的 Let's build GPT，在掌握张量、自动求导后逐步实现小型语言模型。
- **逐层复现：**Umar Jamil 的 Coding a Transformer from scratch，配合作者实现仓库，核对模型、训练与推理怎样连接。
- **补上输入表示：**Karpathy 的 Let's build the GPT Tokenizer，理解文本怎样变成 token，以及分词为什么会影晌上下文和计算。
- **工程进阶：**Stanford CS336 (2025) 的首讲 与 课程作业。在熟悉 Python、PyTorch、数学和基础机器学习后，再深入数据、训练、系统与评价。

Transformer 与 Llama

- Attention Is All You Need 讲解
- Transformer 的 PyTorch 实现
- Llama 架构解析
- Llama 的 PyTorch 实现

模型结构

- RoPE: 旋转位置编码
- MoE: 混合专家
- RMSNorm

微调、推理与效率

LoRA 是参数高效微调方法，可以用于监督微调。KV Cache、FlashAttention 和 PagedAttention 则从不同角度处理注意力计算、推理或内存管理，学习时要区分它们解决的问题。

- LoRA 微调
- KV Cache
- FlashAttention
- PagedAttention

从预训练到后训练

把训练目标和参数更新方式分开看：预训练建立基础模型能力；监督微调（SFT）用示例学习目标行为；偏好优化或强化学习用不同形式的反馈调整模型。LoRA 描述一种参数高效更新方式，可以与不同训练目标组合。反复生成更长的答案本身并不是训练。

进阶入口是 Hugging Face LLM Course、Smol Course 的指令微调单元，以及 DPO 原始论文。先能解释数据样本、损失和最终评价，再运行一个教学规模实验。

动手记录：在教学规模上实现并解释注意力模块，记录关键张量的形状；阅读后续技术时，写清它改变了哪个环节、引入了哪些条件。

7. 强化学习：从基础到 PPO

先理解状态、动作、奖励、策略和价值函数，再学习算法与代码。PPO 是这条路线重点展开的一个方法；算法的实际效果取决于任务、实现和评价设置，不能把它概括为所有问题上的“绝对最优”。

- 王树森：深度强化学习
- PPO 教程
- The 37 Implementation Details of Proximal Policy Optimization

YouTube 进阶： Berkeley CS285（2023）官方播放列表 与 课程网站。这条线把策略梯度放回更大的决策框架，也包含模仿学习、基于模型的方法和离线强化学习；先完成一个小型 RL 实验，再选择相关章节。

复现 PPO 时，把实现细节与实验记录一起学习：观察训练曲线，记录随机种子、超参数和运行环境。原文提到的 TensorBoard、Weights & Biases（W&B）可以用于实验可视化；先建立能保存、能回看的记录，再追求图表的丰富程度。

原文还收录了一个 W&B 网络连接经验帖，作为社区补充材料；这里保留帖子入口，移除了个人分享参数。帖子内容与可访问性以平台实际页面为准。

区分推理框架与强化学习框架

- vLLM 官方文档：主要面向大模型推理和服务，也可以作为强化学习系统中的生成组件。
- verl 项目仓库 与 官方文档：面向大语言模型强化学习。

动手记录：先完成一个小规模 PPO 实验，保留配置和学习曲线，再去理解 LLM 强化学习系统中的训练、生成与评价分别由哪些组件承担。

8. 学习 Agent 与应用

当你已经能阅读 Python、管理依赖并理解模型基本调用方式，就可以学习 Agent 如何连接工具和外部环境。这是原文路线中的应用入口，可以用一个范围明确的小项目把前面的知识串联起来。

- Hugging Face AI Agents Course：课程入口
- 原文收录的课程直播页面

动手记录：选择一个可以核对结果的任务，实现一次工具调用，记录输入、工具返回和最终输出，再检查模型是否确实使用了返回的信息。

YouTube 与系统设计

- 了解课程：Hugging Face 的 Agents Course 介绍与问答。用来建立学习范围，再进入课程单元。
- 观察执行过程：Hugging Face 的 Agent process 演示，配合 首个 Agent 教程。这是执行过程演示，完整实现以教程为准。
- 理解系统边界：阅读 Anthropic 的 Building Effective Agents。预先定义步骤的流程与由模型动态决定下一步的 Agent，适合的任务和成本不同。

一个可运行的 Agent 至少需要明确目标、可用工具、执行状态和停止条件。先实现“选择工具—调用—观察—继续或结束”的小循环，加入输入校验、错误返回、重试上限和运行日志，再考虑复杂编排。

MCP 提供模型应用连接工具和数据的协议。它可以统一接口，但规划、权限和结果核验仍需要在 Agent 系统中设计。实践时可以只接一个可控工具，用参数错误、工具超时和空结果检验系统行为。

9. RAG、记忆与评价

先把“让模型回答”变成“用证据完成任务”。RAG 原始论文 将生成与外部检索结合。练习时可以把自己的公开笔记切分、索引、检索，再要求答案指向真实段落；检索到相似文字和给出正确答案，是两个需要分别检查的环节。

YouTube 实践：LangChain 的 RAG From Scratch 播放列表，配合 官方示例仓库。适合已经能调用模型和读写文件后学习索引、检索与生成；框架接口以当前文档为准。

把上下文、持久化记忆与训练区分开：把信息写入文件或数据库，能改变下次运行可访问的资料，并不等同于改变模型参数。先写清楚每条记忆从哪里来、何时有效以及怎么检索，再增加自动记忆功能。

用小而固定的题集比较直接回答、RAG 和工具 Agent，记录答案正确率、证据是否支持结论、工具是否真实执行，以及耗时与 token。把修改系统时看到的题和最后验收的题分开。Hugging Face 最终项目 提供了把 Agent 放进基准评价的课程入口。

动手记录：为 20 个问题保存参考答案、来源段落、实际运行轨迹和成本，加入资料缺失、相互矛盾和问题超出资料范围的情况。这个小题集是学习练习，不是行业性能认证。

10. 多模态、生成模型与世界模型

多模态学习关心不同形式的数据如何连接，例如图像、文本、音频和动作。先尝试“模型看到了什么”，再研究“它能预测什么”：识别图片内容、生成图像或视频、预测环境状态，以及选择动作，是不同的任务。

YouTube 基础：MIT 6.S191 的 Deep Generative Modeling（2025），结合 课程材料 建立生成建模概览；需要神经网络与概率的基本知识。

作者进阶讲解：Danijar Hafner 的 Unsupervised Intelligent Agents。在理解强化学习后，关注世界模型怎样参与学习和决策，再回到论文及代码核对细节。

世界模型的一条实践路线是学习环境动态，再利用预测进行决策。DreamerV3 作者项目页 展示了学习环境模型并在想象轨迹中改进行为的方法。阅读时同时追踪观察、动作、潜在状态、奖励和策略，避免只看生成视频的观感。

动手记录：在简单仿真环境中比较一步预测误差与多步滚动误差，再测同样控制任务的成功率。视频看起来合理，不足以证明模型能准确处理接触、长程预测或闭环控制；学习目标是分别测量这些能力。

11. Physical AI：从感知到物理行动

这里用 Physical AI 指向与物理环境交互的智能系统：从传感器获取观察，经状态估计或策略选择产生动作，再接受环境反馈。机器人是直观入口，但同时需要理解控制频率、执行器限制、碰撞、接触和环境变化。

YouTube 与动手入口

- **概念入口**：NVIDIA Developer 的 Introduction to Physical AI & Robotics at NVIDIA，先建立物理交互与仿真训练的概念，产品演示与验证结果分开阅读。
- **机器人实践**：Hugging Face 的 LeRobot Tutorials 播放列表，配合 LeRobot 当前文档。先理解数据记录、模仿学习和策略评价；旧视频中的硬件及安装步骤，以当前文档核对。
- **仿真训练**：从 Isaac Lab 文档 了解环境、观测、动作、奖励和任务配置。先运行一个现成的小任务，再改一个参数；进入前核对所需硬件和软件条件。

模仿学习利用示范数据学习行为，强化学习则围绕环境反馈优化策略。完成同一个任务时，可以比较两条路线的数据来源、代价、失败形式和对环境变化的适应情况，而不只比较一段成功演示。

动手记录：先在仿真中完成一个到达或推物任务，再改变质量、摩擦或传感噪声，记录成功率与失败轨迹。仿真表现不能直接代替真机测试；从仿真到真实系统需要重新验证模型、时延和接触条件。

12. 科学机器学习与 AI for CAE

Physical AI 与“把物理知识用于机器学习”有交集，但两个词不等价。工程仿真方向还需要认识微分方程、边界条件、数值离散和守恒关系，再判断数据驱动模型怎样与求解器配合。

YouTube：物理约束与算子学习

- **方向概览**：Steve Brunton 的 Physics Informed Machine Learning 播放列表，适合有微积分、线性代数和简单数值方法基础后选学。
- **物理约束**：Physics-Informed Neural Networks，观察方程残差与边界条件怎样进入学习目标，再复现一个一维例子。
- **算子学习**：Fourier Neural Operator，先理解学习输入函数到输出函数映射的目的，再阅读 FNO 原始论文。

这几条方法可以放在同一个问题下比较：PINN 将方程等信息用于训练约束；神经算子学习一类输入到解的映射；MeshGraphNets 则研究基于网格图的物理动态学习。它们的输入、训练数据和泛化假设不同，不是更换模型名称就能直接互相替代。NVIDIA PhysicsNeMo 可作为科学机器学习的工程工具入口。

动手记录：选一维热传导或弹性小问题，先保存可复核的数值解，再训练一个小代理模型。按完整工况划分训练与测试，比较未见参数或边界条件下的场误差、边界满足情况与推理成本；计时同时注明数据生成、训练和推理是否包含在内。

对于接触和结构问题，进一步检查局部峰值、网格变化与长时间误差累积。可视化中的平滑结果不等于工程精度，模型误差与原始数值模型误差也要区分。

13. RSI: 递归自我改进

这里采用一个便于学习的判断问题：一次改进，是否还增强了系统产生下一轮改进的能力？阅读 RSI 研究时，写清“谁提出修改、改了什么、谁评价、结果保留在哪里”，会比只看“自我进化”的名称更有帮助。

先区分四类机制

机制	主要改变什么	首先核对什么
测试时反馈与修订	答案、上下文	输出是否改善
从反馈训练	模型参数	独立测试表现
Agent 自修改	代码、工具、运行逻辑	真实执行与泛化
程序进化搜索	候选程序与档案	目标与约束验证

这张表是阅读框架。Self-Refine 展示无需额外训练的反馈与修订；从 AI 反馈学习 则包含训练过程。反复改答案、持久化记忆和权重更新，需要分别描述。

Darwin Gödel Machine 研究 Agent 自修改；所引 论文实验 使用冻结的基础模型，改进编码 Agent 的设计。AlphaEvolve 使用生成、自动评价与进化搜索改进候选程序。有限任务上的提升，不能单独证明任意领域的持续加速改进。

YouTube、论文与代码阅读

- **作者技术讲座**：Robert Lange 的 ShinkaEvolve — AutoML Seminar，由 作者讲座目录 链接。适合在理解 Agent、评价和进化搜索后学习；这是作者在研讨会的讲解，不是 DGM 专题视频。
- **跟踪一处真实修改**：阅读 DGM 作者仓库，追踪父 Agent、代码差异、评价结果和保留决定，而非只读最终分数。
- **先核验产物**：AlphaEvolve 结果验证仓库 提供发现结果的验证材料。复核结果与复现完整搜索系统是不同任务。

动手记录：用固定模型和受保护的独立测试，比较一次生成、三轮反馈修订、同预算多次独立采样。记录正确率、真实工具轨迹、token、耗时和失败；再允许修改一个受限工具实现，检查收益是否能迁移到未见题目。评价器与测试答案不放进候选程序可改的区域。

这是理解“改进对象”和“评价可信度”的小实验。即使结果变好，也要区分额外推理预算、记忆、代码修改与模型训练的贡献；要支持更强的 RSI 主张，还需专门测量后续改进能力。

14. 用项目把路线串起来

选一个方向做完整，再扩展第二个。下面是可以自行完成的学习项目，不是已经运行得到的结果。

- **有证据的资料 Agent**：读取一组公开文档，回答时返回段落引用，并通过一个工具核对数值。保留检索命中、答案正确性、引用一致性和成本记录。

- **物理任务策略**：在小型仿真中完成到达或推物，比较不同随机种子及环境扰动下的成功率，保留失败视频与环境配置。
- **工程代理模型**：用已验证求解器生成热传导或弹性数据，对未见工况做预测。报告场误差、局部误差和包含哪些阶段的计算成本。
- **可追溯的改进循环**：让系统提出工具代码修改，经过独立测试后决定是否保留。在固定预算下，与原始工具和额外采样比较，保留每次版本差异。

每个项目至少交付问题定义、数据与划分、可运行代码、基线、评价方法、资源记录和失败分析。学习目标是能解释系统为何得到这些结果，以及它在哪些条件下还不可靠。

15. 把学习变成可持续的习惯

持续获取信息的入口

- **大学课程与作业**：从 MIT 6.S191、Stanford CS336 和 Berkeley CS285 获取结构化知识、讲义与实践任务，按自己的基础选择章节。
- **原始论文与讨论**：通过 arXiv 机器学习分类 追踪预印本，通过 OpenReview 查看对应投稿及可见的评审讨论，分别核对论文版本与发表状态。
- **会议论文集**：在 PMLR 和 NeurIPS Proceedings 查找正式论文及补充材料，围绕一个具体问题向前追溯基础方法。
- **作者代码与工程文档**：从论文或项目主页进入作者仓库，读取 README、依赖、示例和版本记录；用 PyTorch 文档 等原始 API 文档核对实际实现。
- **实验室与作者文章**：将本指南中的 DeepMind、Sakana AI、NVIDIA 和作者项目页作为研究线索，再对照论文、代码、实验条件与失败案例理解结论。
- **社区发现入口**：Hugging Face Daily Papers 可以帮助发现材料。热度、转发和评论用于发现问题，技术判断仍回到原始证据。

读一项新工作时，先用自己的话写出问题、输入输出、基线、评价与限制，再挑一个最小结果复现。留下来源链接、版本、代码提交和运行记录，这样同一个知识点就能在课程、论文和实现之间相互核对。

英语阅读和听力能够帮助你直接接触课程、论文和官方文档。遇到翻译不清楚的内容时，回到原文、代码和实际运行结果，往往比继续收集更多二手解释有效。

原文提到的学习者与创作者包括：

- **Umar Jamil**：算法实现与论文讲解。
- **Andrej Karpathy**：神经网络和语言模型的代码实践。
- **Mu Li (李沐)**：深度学习课程、论文阅读与工程实践。

围绕自己感兴趣的问题检索资料、阅读文档、复现代码，也向他人清楚地描述问题并寻求帮助。保留环境、修改和结果记录，让学习过程可以回看和继续。

LLM 是这条路线的重要部分，AI 的范围也更广。随着技术热点变化，信息检索、理解问题和动手实现的能力，仍然能够支撑下一次学习。

勇敢开始，与志同道合的人交流，保持专注，也保持好奇。**Get your hands dirty.**