2025 · REVISED EDITION

TO THE NEW STUDENT

致新生的你

关于学习、科研与保研
给泛计算机专业新生的一份成长路线图

AXI
个人博客精选长文

初版 2025.06
修订 2025.12

CONTENTS

目录

从学习方法到人工智能技术栈，再到科研实践与推免流程。目录中的页码以正文第一页为起点。

READER'S NOTE

阅读说明

> **TL;DR**
>
> 本博客旨在介绍在人工智能时代下，作为泛计算机领域及人工智能领域从业者，特别是本科学生，如何进行日常学习、开展人工智能领域科研以及参与保研流程。对于不同的部分，博客中的展开相对独立，读者可以根据自己的需求选择阅读。
>
> 文中大量内容旨在阐述思考方式与方法论，篇幅较长，同时给出的学习路线实质上较为完整且简洁，读者无须担心。建议完整阅读整篇博客。
>
> 对于西安交通大学的学生，关于学习的更多资讯，如其他学科、学生事务或者生活类内容，请见 西安交大生存指南↗。

当你看到这篇博客的时候，笔者的大三生活已经快要结束了，期末考试已经完结，剩下的只有几篇实验报告需要去写，甚至往远去说，可能将来的去向也已经在脑海中形成了一个模糊的轮廓，一切都将尘埃落定，并且踏上一个新的起点。

而读者们，抱着审视的态度来看这篇文章的 Seniors，以及希望从中获得一些知识的 Juniors，这是我这三年的感受，关于如何在大学中找到自己想要做什么，找到方向以及自己的爱好，假如你恰好喜欢做学问，那么如何做研究，如何看待这一切，在这个奔涌的时代面前如何去适应一切，我会给出一些我的看法。

01 前言

关于向新生做一些介绍和科普的事情，如何去规划自己的大学生活，如何适应大学的节奏，假如你是想要升学，想要「卷」，那么应该按照什么顺序去做什么，这些内容我其实做过很多。

我在不同的班会上给目前大二以及大三的同学们都进行过一些分享，以及在学组的研讨中，也给大一的同学带来了一些自己当时的想法，当然时过境迁，我对一切又有了一些新的视野和新的看法，所以打算写这篇博客。

其实讲实话，之前的一些学习心得或者总体的方法论，我在另一个网站中其实有过一些介绍，就是我搭建的西安交大生存指南 ↗。这其实更像是一个 wiki 一样的内容，里面大大小小塞满了我各种的私货。

在某个时刻，我看到其他学校的类似搭建，于是萌生了这个念头，并且凭藉着当初远超现在的执行力，搭建了那个网站。当时我还相信开源社区的力量，事实上这也确实是一片热忱的土壤，然而总归大多数的人是消费者而非生产者，网站获得了数万的浏览量，但是投稿者寥寥无几。

我逐渐意识到，搭建一个大而全的生存指南网站是不现实的，而我当初的视角也存在一些偏差。实际上对于目前大三年级的同学们来说，甚至可以说 80% 的同学都没有在 Github[1] 上给一个公开的项目提交过 PR[2]，而人们总是对于没有做过的事情充满畏惧，难以参与贡献流程。和计算机非常相关的专业尚且如此，更遑论广大的文科或者理科的专业呢?

因而西安交大生存指南中充满了我的一面之辞，而我的视角总是有限的。相对来说我可以这样定性，我在大学的三年时光中更多的是学会了如何「卷」，而恰好科研以及各种技术也是我的兴趣所在，因此对于更广大的对学习弃之如敝履的同学来说，对于如何学习竞赛和科研的长篇大论并不在他们的兴趣范围内，而如何在这些课本与屏幕之外的丰富生活中获得乐趣，显然并不在我的认知范围之内。

更何况不同专业之间的「卷」度也是有区别的，这取决于专业内部的升学压力以及整个专业的学习节奏。

对于人工智能来说，大一或者大二进入课题组实在是家常便饭的事情，而即使读者甚至整个年级都排斥这种现象的出现，大环境不等人。一方面做出科研成果的门槛在不断下降，一方面全部的学校中具有前沿视野的同学都在进行着率先科研的军备竞赛，在这样的浪潮下，很难有人可以幸免。

然而对于不少的文理或者医学的专业来说，学好课内的知识已经是需要做到的全部，当然这也不意味着轻松。看着我女朋友医学考试的大部头资料，我每一次都不禁感慨，她可以坚持下来每一次复习和考试，简直是一个奇迹。

因此，在一个面向全体学生的百科全书式的内容中大谈进组科研的必要性，显得有些不切实际，且缺乏「泛化性」。

这使我萌生了写作这篇博客的想法，部分学习方法与节奏复用了我在'西安交大生存指南'中的总结，更多内容则面向人工智能专业进行特别撰写，不只是如何学，如何科研，也包括你需要认识到自己需要的是什么，因为人生不只有一条路，甚至说，通往成功的人生也不止一条路。

02 濒临奔溃的本科教学

你好，欢迎来到大学生活。

假如你已经决定正式入学某一所大学，你也就抛下了之前十八年来所积累的大多数的成绩，无论你来自于哪个省份或者哪所重点高中，现在一切退回起跑线，我们不得不重新开始。

事实上，在人的一生之中，除了掌握的知识与能力之外，很少有什么能够始终伴随我们的一生，成绩更是其中贬值的最为迅速的一项。

在西安交通大学中，一个可以透露的统计数据是，再比你们大三届的 2022 级中，大一结束的时候学分绩的中位数才仅有 88 分左右，而在两年之后的 2024 级，这个数字已经跃升至了 90 分以上。考试难度的波动以及教学方案的调整带来的影响确实不可忽略，但是学生们在笔头下的功夫确实肉眼可见的增加了。

在繁重考试以及绩点排名压力的背后，是AI浪潮的席卷。伴随着 ChatGPT[3] 的横空出世，全世界对于人工智能的期冀，连带着学界加快研究节奏的热忱一同高涨。在比你们大五届的2020级，本科生第一作者身份发表顶级会议的最早记录是大三下学期，而且两年之后的 2022 级，这个记录被笔者刷新到了大二下学期。放眼全国，在大一发表顶会甚至 spotlight[4] 或者 oral[5]，也很难说是一件稀罕事。

> ⓘ **为什么要科研**
>
> 假如我们将学生的大学经历定义为以「成材」为目的为期四年的培养，从功利主义的视角来看，一个合理的衡量标准是毕业后薪资。
>
> 以年包计算，且不考虑大多数的离群点，常规大厂的入职门槛通常是 985 硕士学历，通过企业实习的方式构建简历，可以获得 300K 到 500K 不等的年包。而对于通过科研方式构建简历，并且在热门领域具有与企业方向匹配的科研成果的科研岗求职者来说，这个薪资通常可能是 700K。对于方向匹配的博士来说，薪资以 1M 起步也并非是一件少见的事情。我们可以说，大多数你听说到的关于 AI 领域当前的热潮以及对应的薪资高涨都绝非谣言，所以即使不管科研理想，想想真金白银，也会让你足够有动力。
>
> 当然，另外一个不可忽略的要素在于科研对于保研申请的裨益。在硕士学历似乎已经是求职敲门砖的当下，保研对于大多数读者来说是一条相当有诱惑力且几乎必须的路线。相较而言，考研会浪费你的大量时间，在研究生期间需要漫长的时间进行冷启动，从而有更大概率错失后续实习的黄金时期。对于保研简历来说，科研经历毫无疑问是最具竞争力的要素，并且有利于你申请到更加顶尖的学校。

在本科培养方案的规划下（以西安交通大学人工智能专业为例，而大量的传统院校以及传统计算机专业甚至不会介绍人工智能的相关技术），假如读者继续按部就班的一步一步学习，直到大三上学期才会开始介绍计算机视觉[6]以及自然语言处理[7]，而课程的内容涉及的技术多半来自上个世纪或者五六年前，而对于当下最时髦且流行的技术则鲜有提及。事实上在国内即使是最顶尖的大学诸如清华北大，也很难规划出让学生可以在大一发表顶会的培养方案。

因此对于那些希望在学术体系下做出一番成就的读者来说，如何达到这一目标，那个答案已经昭然若揭，脱离这套已经生锈的、难以跟上科研节奏的本科教学体系。

> **关于提前选课的必要性**
>
> 值得注意的是，脱离本科的教学体系完全不等于你有必要进行课程的跨选或者提前选课，这除了给你当前学期增加过多的学习压力，并且浪费你的时间之外，不会有别的帮助。正确的方法是寻找更加高效的学习方法，并且投入恰好的时间在本学期的课程中，使得你可以以「够用」的成绩通过考试，除此之外不再花费任何的时间留恋其间。

03 学问的学问，生活的学问

对于那些早已经对科研跃跃欲试、摩拳擦掌的读者来说，后面的内容定然具有一番参考价值；那些还憧憬着大学美好生活以及多元发展的读者来说，则不免微微皱起了眉头。

笔者对于科研的执着以及描写无疑是对焦虑的渲染，然而也别无他法，笔者的大学三年时光主要就集中在这一系列的探索之中，也难免将其作为重点，而但凡皆为同好的读者看后，必然也能感同身受的给予认同。不过在开始对于如何做学问的学问长篇大论之前，还是有必要先讲讲生活的学问。

一组简单的数据可以作为参考，正常一家三口买菜做饭，一天的开销大约是五十元；一线城市一居室公寓一个月的租金大约是三千元；本专业的同学在本科可以获得的实习工资大约在每天两百元或以上；西交本科应届毕业生的工资在一万元左右。

善于观察与思考的读者不难获得一个答案，即在本专业中获得一个像样的学位，养家糊口，并且培养自己的兴趣爱好并非一件难事，而投身科研之中，消耗大把的时间进行研究，可能会让你放弃几乎全部的课余休息时间，然而结果依然是抽奖以及千军万马过独木桥，换来的薪资确实更高，但这是否是你最后想要的生活依然是一个问号。

每天按部就班接受培养方案的教育，认真听讲做笔记，完成作业，在考试中取得好成绩，这是一种活法；参加机器人竞赛，或者其他的团体活动，和队员一起起早贪黑磨练技术打磨能力，在激烈的角逐中取得优胜，这是一种活法；投身科研之中，加入国内外顶尖课题组，发表论文，交流学术，这是一种活法。

然而对于圆满与美好的定义总非是狭隘的，和好朋友们加入二次元社团，每天在欢声笑语之中度过，在考试前三三两两聚在一起研究如何低分飘过期末考试，然后畅想假期一同的出游，这也是一种活法；或者只是一个人默默的学习，学到够用为止，找一个自己的爱好，又或者干脆就漫无目的，在校园里投下阴影的梧桐树下行走，偶然间走到思源活动中心的南门，那里没有枝干的遮挡，那片湛蓝的天空是那样的动人，霎时间豁然开朗，这也是一种活法。

假如我们为了薪资或者前途发展功利去看，并且选择一种期望值最大的途径，那么几个月后西交将有七八十名都读过本篇博客的同学一同踏入这个学校，手握相同的武功秘籍，陷入水深火热。放眼全国，这个数字可能以百或者以千计算。

不过实际上反直觉的是，利益最大化绝非大多数人对于成功的定义。设想科研这条路走到最后是怎样一副情景，你或许成为高校的教授、企业的 CTO[8]，领导数十甚至上百人的科研团队，每天早上起来一杯咖啡，然后阅读论文、开会、讨论，为你的组织拉取资源，思考将来前沿的方向，然后在忙碌奔波中度过一天，你能否想象你的面孔出现在那个模糊的身影上，那是否是你想要的生活，而且假如不是，如何达到那个你的梦想，这是一个留给每个人思考的问题。

04 Learn to learn

在传统的机器学习话题中，有一个概念叫做元学习，meta learning[9]，有的时候也被称为 learn to learn，学会如何学习。虽然事实上这一领域中的绝大多数内容都是通过梯度传播来改变那些本来固定的超参数，在当下的应用场景也着实有限，然而这个概念本身却一直被学界所重视，并且在此之后的如持续学习[10]等领域也都继承了这一思想。

学习如何去学习这一内容在人们心中是如此被看重，大概还是由于在人的成长与发展中学习是最不可或缺的一环。在本科的教育中学习那些培养方案中的课程，在竞赛中你学习那些课程之外的技术，在科研中学习那些前沿的论文和思想，而哪怕是在生活中，如何交往如何沟通也是关键的一环。

在这个章节中笔者会介绍几个关键的学习心法，一些是在课程学习中适用的，一些则在更广泛的学习场景中都具有价值，希望可以给读者带来一定的启发和帮助。

信息差

笔者在大一结束的时候，有不少的新生前来向我请教，包括学习社团生活科研竞赛的种种经验，我当时便放出过这个论断，大学最大的差距不是天赋，不是努力，而是信息差。

不同于高中生成长路线的单一，及拜诸多辅导机构所赐，网络上或现实中均充斥着大量的有用信息，大学的信息依然繁杂，但是质量却参差不齐。

笔者在一个没有很多人指导的环境下独自摸索，踩过很多坑，也做过很多错事，回头来看，很多的努力和付出并不是必要的，而在一个追求快节奏的时代里，如何从这些庞杂的信息中识别出路径，并且在诸多路径中挑选到最短的那一条，毫无疑问是一个有价值且意义重大的学问。

在后续具体的学习方法中，将会有笔者这些年来积累的信息的汇总，而在这里让我们着眼于这抽象的思想，简单思考下如何减小信息差，并且获得到更多的信息。

对于绝大多数的减少信息的方法，我们可以粗略划分为两种类型，一种是提问，一种是搜索。

提问的艺术

当你明确了你的疑问，明确了你想要获得怎样的信息，提问是最快的解决问题的方法。提问的关键有三点，如何形成正确的问题，如何找到可以回答问题的人，以及如何整理答案，并且从中获得你需要的内容。

正确的问题

排除少数的有偿咨询之外，绝大多数的提问是无偿的，而这往往意味着提问者没有付出任何代价就获得了自己想要的内容而被提问者则没有收获任何价值，而付出了时间和自己的知识。在这种不平衡之下，意味着提问者需要向被提问者表达最基本的尊重。提问的艺术 ↗是起初源于黑客社区的册子，里面讲述了新手如何向老鸟获得答案的方法，而这其中的内容极具参考价值，笔者也强烈推荐每一个读者去将其通篇阅读，这会为你将来的提问之旅带来更好的结果。

简单来说，提问的艺术中要求提问者需要向被提问者准确描述自己的状况，自己的需求以及目前已经尝试的方法，并且在获得自己需要的答案或者获得了帮助，即使并不是自己需要的内容后都表达感谢，这是最基本的尊重的体现。

对于那些乐于回答别人问题的人来说，他们很喜欢一个好问题，并且愿意为之付出大量的时间，因为这不仅可以帮助到提问者，也可以帮助被提问者整理自己的思绪，对这个问题产生更深刻的认知。然而相对的，假如读者希望被提问者可以花费十分钟回答自己的问题，毫无疑问自己也有必要花费十分钟来准备自己的问题，这才是公平的体现。

「我具有不错的课内成绩，现在正在抉择是参加机器人竞赛还是直接参与科研，我在网络上搜索了一些科研的知识，感觉具有比较高的上手门槛，而且领域繁多，我不知道如何选择，所以想要询问你的意见」，并且在提问结束后说上一句谢谢，听上去总比「怎么变成像你一样的人」，并且在获得回复之后已读不回听上去要好的多。

正确的人

相比起来形成一个正确的问题，这种只需要勤加练习，并且付出时间的事情来说，找到合适的人总是一件更难的挑战。

对于人的信息的积累是一件需要持之以恒的事情，并且鲜有捷径可走。了解每一个人擅长做什么，对什么的理解深刻，并且在哪些方面可以帮到自己。在日常的交流中读者需要细心注意这些内容，并且记在心中。

一个好的开始是在阅读读完这篇博客之后，对于自己不懂的地方向笔者提问，不过在这里依然有一个小小挑战，毕竟有的时候你不仅需要知道你需要向谁提问，还需要知道你怎样才能和他取得联系。

搜索与整合

除了积极的提问，从搜索引擎以及各路论坛讨论中获得信息也是一项十分关键的技能，而甚至对于一些技术问题来说，也只有在那些深奥的技术文档和技术论坛中才能获得答案。这是一项十分后期的技能，不得通过这种方式获得信息的时候，意味着你在这个领域已经打磨良久，并且有必要具备更 Senior 的能力了。而在此之前一些视频平台或者诸如知乎的社区，也可以给你理想的答案。

在三年以前，这个回复显然是唯一的正解，但是时过境迁，伴随大语言模型的兴起，面对搜索这个困难的开放性问题，显然有了不一样的答案，那提问与搜索集合成一体的究极答案，向大模型提问。

得益于大语言模型经过的大量预训练[11]，海量的互联网信息被它整合在自己的权重中，这意味着你可以通过日常的对话，就像和一个好朋友聊天一样，获得绝大多数你需要的信息。

有的时候你需要学会辨别信息获取的边界，那在互联网上广泛存在但是可能稍显晦涩的内容使大语言模型输出的强项，而假如这些信息更具时效性或者出现频率较低，则可能传统的提问与搜索才是更好的答案。比如说如何学习 Python 这门语言，向一位及时已经深谙到多年的学长进行数小时的提问，倒不如和大模型在对话中持续学习来的更有效率；而西安交通大学人工智能专业历届的保研比例和分数线，无论你用何种口吻或者要求向模型提问，他都很难给出正确的答案。

同时不容忽视的是大模型的幻觉问题，也就是模型输出的内容往往不一定是完全真实的，对于那些可以通过实践验证正确性的内容，最好的做法应当是立刻将之付诸实践，并且检查正误；而对于那些包含事实性内容的答案来说，将模型的输出粘贴到搜索引擎并进行二次确认，毫无疑问是更好的做法。

读者至少需要学会如何向 DeepSeek ↗ 或者 Qwen ↗ 这样的国产模型提问，进一步了解如何访问并获得国际上最前沿的模型服务，如 ChatGPT ↗ 或者 Gemini ↗，则是更加进阶的做法。当然，这些内容会在后续的详细阐述中有更具体的体现。

持续学习

假如读者喜欢学习新技术，那么描绘无数种广袤的前景是不难的事情。读者可以学习传统的前端后端框架[12]，搭建非常炫酷的网页（比如本博客就是在模板的基础上加入了大量自己的修改形成的）；也可以深入 Linux 系统[13]，折腾 ArchLinux[14]，搭建美观且实用的开发与生产环境；也可以苦练算法，了解那些前人创造的精妙思想，并且在算法竞赛上一较高低；或者学习大模型的一系列技术栈，训练自己的模型，并且在科研上有所建树。同样的路还有无数条，学习计算机图形学[15]了解渲染的机理；学习 CPP[16] 或者 C#[17] 之后用 Unity[18] 或者 Unreal[19] 搭建自己的独立游戏，甚至参加 GameJam[20]，和其他开发者们即兴创作；甚至是研究 TCS[21] 或者计算机体系结构，有趣的知识是无穷的，等待着你的探索。

经过了笔者的一系列报菜名中，那些热爱学习的读者心中不难涌现出一团烈火，想要用自己有限的大学时光，充分了解这些技术，把自己发展成一个更强大的人。但是在经历了无数学弟学妹的请教以及后续的观察之后，笔者不得不得出一个结论，在绝大多数时候，兴趣所带来的学习动力都是不能持久的，然而只有持久的学习才能让你真正掌握某个技能。而这一切的关键在于正反馈。

记笔记是一种正反馈

记笔记是笔者推荐的一种极佳的学习方法。

笔者不了解读者在高中期间对于记笔记的看法，例如是否会在复习的时候回顾这些笔记，或者认为记笔记是否是一种累赘，但是在高中的生活中，笔者从来没有记过一行笔记。

尽管如此，笔者依然推荐在大学期间使用记笔记作为最核心的学习方法。

一种广泛的论调是，大学的学习以自学为主，这一点是不可置否的，而在自学的过程中，能否让自己获得持续学习的动力，这成为了自学过程中最关键的一环。

一种常见的现象是，每每开学，无论之前在高中，还是如今的大学，都有很多同学向笔者咨询学习相关的事宜，仿佛准备大干一场，在学业中一展宏图。而在开学两个月之后，一般他们的朋友圈或空间便被二次元或者现充生活占据了，而全然不见当初的拼搏模样。有时候学习的难点并不在于如何高效的学习，反而在于更为基础的一点：如何坚持学下去。

假如并非十分自律的人，在学习的过程中，毫无疑问，正反馈是十分重要的。适当的正反馈不但可以让读者有学习的动力，也可以让读者保持学习的节奏。

令人遗憾的是，在大学中，正反馈往往不是如此的常见。在不经常存在考试，甚至课程质量良莠不齐的大学当下，在学习中获得的反馈是十分难得的。更加经常的情况是，你有数不尽的内容可以学习（学完课内可以拓展课外知识），但是每每学习结束之后你并不知道自己的学习是否为自己带来了改变，这与高中阶段学习

少量的内容，但是在重复性的刷题以验证自己的学习成果不同，而一味地向前冲有时会带来迷茫，从而让读者失去兴趣。

关于笔记的用处，不排除一些人通过记笔记起到加深记忆，以及获得一手的复习资料的作用，但是在此之外，记笔记也是一种记录，如日记一般，可以告诉读者：“我今天学习了很多内容”。

故也便不难阐述记笔记带来的正反馈了。事实上，正反馈的缺失往往来自于难以对自己的学习成果带来一种定量的总结，这是显而易见的，因为并不存在大量的考试，而学习的内容也远远多于考试的内容。但是通过频繁的记笔记，笔记的字数以及内容的积累可以带来一种最为直观的反馈，尤其是当读者自己的笔记从千字到万字再到数十万字的时候，便可以很明显的感知到，如今的自己已经今非昔比。

在后文中，笔者会介绍自己使用的工具流。也就是 Obsidian ↗，以及使用 Markdown[22] 进行记录笔记。使用 Markdown 进行笔记的记录是一件非常常见的做法，这一属于程序员的记录语言提供了极佳的记录效率以及内容的规整性，而在如今的编程世界中，基本上全部的文档，包括本篇博客的背后都是 Markdown 正在书写。

一条精心设计的路线

尽管我们有了记笔记这一带来正反馈的利器，但是在学习的过程中依然可能存在不少的挫折，使得读者难以集中精神，或者最后也会半途而废。怀着满腔热血冲进一个自己喜欢的领域，即使自己一无所知，但是闷头苦学，这种想法毫无疑问是不可取的。

事实上每一个领域，伴随着它涉及的知识与你目前掌握的知识面的交集越来越少，越来越多的内容将涉及不断地自我递归式的学习，即对于自己不懂的内容进行查询，然后再将查询内容中不懂的内容进行查询，以此类推，直到了解整体的知识脉络。这种频繁的递归式学习只会让你的学习负担越来越大，而难以对于整体的知识脉络有一个清晰的认识。

我们推荐在一些局部使用递归式学习，这有助于你在小范围内规避多余的学习内容，而聚焦在必要的内容上，但是当将视野放大到整个领域之后，我们需要的是如庖丁解牛般的游刃有余，将领域的大块知识进行分解，了解自己需要掌握的知识边界，再自底向上进行学习。

没有了解全貌，就难以进行笔记的记录，至少大多数人都难以提起兴致记录笔记。人们往往因为内心清楚自己对于这部分知识还没有完全掌握，便断定自己梳理出来的记录也一定毫无章法可言，从而拒绝进行笔记的记录，而笔记的缺失有进一步带来了正反馈的缺失，从而影响了学习的节奏。

在学习之前，读者有必要详细了解自己需要学习的领域。就像之前所说到的那样，你需要通过提问或者搜索来让自己先了解某个领域的学习内容，以及如何阶段式的完成一个一个目标。尽管其中的很多中间状态并不是我们目的所在，但是阶梯式的划分有助于成就感的增加，而且在前人基础上总结出来的阶梯会使得学习更具条理性，并进一步降低学习的门槛。在后文中会详细介绍人工智能这类专业如何进行详细的阶梯式分解，并且假如上且有余力，给出一些其他的领域的介绍。

听课是一种糟糕的学习方法

行文至此，读者们已经开始逐渐接近笔者的思想了，我们在追求的是一种高效的学习方法，并且可以长久进行学习，而在这一语境下，听课是一种糟糕的学习方法，也是一种平庸的学习方法。

这并非一种暴论，笔者并不是高傲地站在一种高位者的视角上，将那些只会听课的学生们评价为庸才与平庸之辈，事实上他们之中不少人都天赋异禀，但是学习的方法或许不是最本质的。我们需要意识到，学习这一过程本质上就是接受知识、提出疑问、获得答案的过程，并且在这个过程中不断迭代自己的认知，从而获得更好的理解。提出疑问以及获得答案的过程往往是通过自己的悟性以及之前学习的知识，包括提问以及请教他人，而接受知识的速度往往很大程度上取决于你受到的教育的知识密度。

回想一下课堂的场景，即使是最为经验丰富的老师，也不可避免地要解决一个难题，他需要将文字甚至符号化的知识进行口述，并且在课堂中再现接受知识、提出疑问、获得答案的循环，这一循环需要关照课堂里面的每一个学生，自然也包括某一位悟性比较差，并且距离前排比较远的同学。优秀的教师在教授知识的时候总是娓娓道来深入浅出，并且受到同学们的广泛好评，但是读者需要注意的是，我们在追求的本质上只有一件事情，那便是尽可能高效的获取知识，而上述的过程显然不是最快的。

一个有趣的事实是，假如说读者在期末周进行过考试前的突击，对于那些广泛存在在各个专业之间的通识课程，网络上往往充斥着无数的速通课，那些仅需不到三四个小时的视频便可以概括你横跨整个学期，高达四个学分的大课，并且假如选择恰当，几乎可以 cover 老师课堂上讲解的 99% 的内容。这个现象从侧面反映了常规课堂中知识密度之低，其中充斥着提问、重复的回顾、对于大量练习题的讲解甚至还有课堂小测。

事实上，我们对于知识的掌握程度的需求其实更加朴素，掌握知识的大概，并且在需要运用这一知识的时候，直到如何搜索关键词，再次搜索出来的时候知道如何去用，毕竟实践是一场开卷而非闭卷考试。用一个更加恰当的比喻，我们只需要学习到可以抄懂答案的程度，而不是独立解决难题的程度，在大多数时候就已经足够。

一种笔者推荐的最高效的学习方法当然是看书，书籍中包含最凝练的符号化表达，而同时，一个令人惊喜的事实是，虽然说那些符号乍一看令人生畏，但是假如沉下心来仔细阅读，事实上一切内容也并没有那么难懂。

05 人工智能的学习路线

看完了上述啰嗦的长篇大论，那些对于看似读者早已经了解的技巧和知识的反复强调，我们终于来到了这些充满干货的环节，尽管实际上在我看来，上述的心法才是重点。

那些可能你认为是废话的内容，正是因为我在反复强调，所以你有必要在学习过程中反复回看，注意自己是否真的做到了，而无论如何，在一段时间过后，再度回看你的学习过程，也很难不由衷的感慨这些内容在学习过程中的重要性。

回归正题，让我们正式开始学习路线的分享，我们假设你是一名之前仅受到高中教育，没有接触过计算机相关内容的大一新生，以下内容你只需要按照顺序去学便可以从中获得收获，而对于那些更有经验的人来说，以下内容也可以作为一个 checklist，看看有什么内容是你还没有涉及的，假如有什么更好的选择和建议，欢迎评论补充。

值得一提的是，由于是学习路线而非大而全的指南，因此比如说在练习某些课程的时候，我会首先给出我推荐的内容正常情况下，你需要把推荐内容学完即可，而不需要涉及任何拓展知识。它们的知识点大致都是同质化的，假如想要展开研究或许可以，不过在这个节奏下，先掌握必须的要点，并写在实践中查漏补缺或许才是真理。

当然在此之前，**你需要学会如何访问谷歌**，假如你不知道方法，问问身边的朋友，这是一切的起点。

ⓘ 笔者的能力以及局限性

一位在领域中久负盛名的专家的分享往往最具价值，而当你听一个相对意义上的「失败者」对于自己的经验侃侃而谈，那么可能需要谨慎甄别了。因此读者有必要也有权利了解笔者个人的水平，从而认识到笔者的局限性。也就是，笔者对于如何达到笔者这个水平以及稍微更上一层楼（即分享笔者在这一过程中哪里可以改进）具有较大的话语权，但是如果读者已经是这个领域的专家并且寻求更深层次的讨论，那么本章节的内容也就相对只具备参考价值了。

对笔者当前的阶段来说，目前笔者在本科期间发表过两篇顶会一作论文，分别是大二的 ECCV 以及大三的 CVPR。在发表 CVPR 之后，笔者对于发论文的兴趣几乎消磨殆尽，并且开始追求更加系统化的成熟探索，而非对于比较显然的 Idea 进行雕花。了解什么内容是重要的，去做对领域有意义的事情，大型的开源项目或者技术报告，这是笔者目前的追求，并且已经作为核心贡献者发表了具身智能领域的一篇还算优质的技术报告。从对于学术的理解上，笔者对于人工智能的某个领域（即具身智能）十分了解，并且在其他领域也进行过非常广泛的阅读，在 insight 上面通过和别人的讨论也可以具有一定的深度。笔者其中一个明显的不足在于笔者目前的绝大多数工作都是所谓的独狼型科研，也就是在论文的整个过程中，笔者承担了绝大多数的工作，而并没有和合作者均摊任务，并且达成更良性循环的合作关系。

在科研中，假如硬要和考试应试对比，不可忽视的一个重点便是在于其允许的团队以及资源不平衡的现象。即使你的个人水平很强，但是你一个人孤军分站同时没有资源，而另外一个团队只有七八人甚至十几人同时工作，即使他们人均水平远不如你，但奈何人数众多，且资源广集，最后项目的成果自然远好于你，那么他的论文可以中稿而你的不能，也自然无处说理。对于科研来说，最终的效果以及方法是一切的关键，但是至于你是怎么得出这个方法，或者你在背后付出多少努力，这些东西还是留到成功之后在 talk 上漫谈时不经意间讲出吧。因而如何通过社交结识更多具有能力的合作伙伴，通过合作，充分实现你和他人的创意，做出更多有影响力的工作，并且利用共同的计算资源以及其他资源，这是一门不得不学的学问，相应的经验，欢迎大家在评论区补充。

一切开始之前

哦，我的朋友，等下，在一切开始之前，让我们明确我们的目标，我们是不是真的喜欢科研。

假如你不喜欢科研，那么好的，这并不妨碍你继续看下去。从绝对功利的角度来看，科研依然是当下提升个人竞争力最具性价比的方式，可以在短时间内让你学习更加超前的知识，从而以更好的视角回看课内以及其他的内容，起到事半功倍的效果。

当然，假如你喜欢科研，也不妨看看谢赛宁老师在 CVPR 上给出的 talk，Research as an Infinite Game ↗，用心体会，给自己一些更加长远的动力。同时，一些心法也同样重要，Omar Khattab 在 Github 的 Blog，On Impactful AI Research ↗ 讲述了如何做出有影响力的研究，言之有理，值得细品。

那么接下来，不要脱离你的主线，沿着这些前人的道路走下去，然后让我们开始 from scratch 的 Deep Learning 学习之旅~

基础数理知识

无论是什么理工科，事实上都是建立在最基础的数理知识之上，也就是以高等数学、线性代数以及概率论为代表的这三门科目，他们也是每个专业在大学期间的必修课。

网络上存在着大量的相关课程，假如有条件的话可以观看 MIT 的相关课程，即 高等数学-18.01 ↗、高等数学-18.02 ↗、线性代数-18.06 ↗。

事实上尽管在网络上对于国内教材的批判呼声很大，但是对于了解以及学习来说，其实无论是哪些教材或者课程并没有本质的区别，只要认真学都可以熟练运用。有的时候我们要将课程知识当作工具而非一种艺术，对于这种起步时需要学习的基础知识尤其如此，在课程学习中患上「收集癖」，还没怎么开始学习就在挑选课程中看花了眼，不如花上一周先学完再说。

在这里笔者推荐 宋浩 ↗ 的相关课程，属于按照国内课程的教学大纲进行的常规讲解，尽管说不上鞭辟入里，但是胜在详细。同时对于线性代数这门可以理解为几乎最重要的数学课程，同样推荐读者观看 3Blue1Brown 的相关讲解 ↗。

从本质上来说，你只需要了解这三门课程中的部分内容，即高等数学中多维的积分以及微分、线性代数中矩阵的乘法与求逆以及概率论中的贝叶斯相关内容，从事实上就已经可以读懂大多数科研的论文以及思想了。当然，对于一些涉及数学更多的研究领域，你需要专门了解这个领域，问问领域中的专家，特定地学习更多的数学知识。

令人欣喜的是，在当下学习人工智能的门槛确实令人发指地低，我们更多时候（尤其是在科研的起步阶段，以论文发表而非更加具有科研理想的深度探索为目的）在通过工程手段以及枚举法搭建模型，这并不需要那么多的专业知识。

认识你的电脑

对于电脑小白来说，一般还是建议首先使用 Windows 电脑，假如没有玩游戏的需求，购买一个商务本即可，一般来说对于较重的渲染或者作业任务，在后期都是直接在服务器下运行，而不需要本地的计算资源。

本地的办公环境还是要保证一定的轻便性，而由于在此之前绝大多数的读者对电脑的使用环境均是在 Windows 下进行，并且网络上对于 Windows 的资料远大于其他系统，因此在绝大多数情况下，笔者均建议电脑小白选择 Windows 作为自己第一台电脑的操作系统。

假如读者已经领悟了便捷开发的重要性，并且具备一定的电脑使用能力，或许可以考虑使用 Mac。与此同时在本电脑上在后续也可以选择加装 Linux 作为双系统，以进一步提高生产力，并且贴近实际的开发环境，然而这一部分内容并不属于新手，将在后续展开说明。

对于之前电脑接触不多的读者来说，可以阅读 你缺失的那门计算机课 ↗ 来进行一些基础知识的扫盲，其中的前几个篇章都很有阅读的必要，而超越篇作为科普读物也尚且可以。假如读者不是对自己电脑的基本功绝对自信，也同样建议扫一遍目录，学习一下基础使用。

对于电脑的基本熟悉流程，预期在三天之内完成。

Markdown，记录你的笔记

在正式学习某些专业知识或者编写程序之前，了解并学会使用 Markdown 可以说是十分重要的技能。

简单来理解一下 Markdown 是什么。按照官方的说法，Markdown 是一种轻量化的标记语言，将带有特殊标记的纯文本转化为可以直观显示的 HTML 或者 PDF 文件。因为其便捷的特性，事实上当前互联网中绝大多数编程的文档，Github 上面的 README，甚至是本篇教程，都是使用 Markdown 完成的。

以下是一个简单使用 Markdown 的示例：

```
## 这是一个二级标题

### 这是一个三级标题

- 这是一个列表
- 这是一个列表
```

可以被渲染为：

我们不难简单理解 Markdown 流行的原因。在了解 Markdown 之前，读者可能接触最多的两类记录文本的内容分别是记事本以及 Word。记事本没有任何的渲染功能，作为临时的记录尚可，但是一旦要整理成可视化更好的文档便有点难以胜任了；而 Word 虽然功能强大，但是第一在于微软将其闭源，一般来说必须使用其软件才可以打开对应的文件，并不具备这类软件的机器上难以显示。同时相较于简单的纯文本，Word 对于正常的文字记录需求有些太重了。

读者不妨回忆下自己使用 Word 的需求，一般来说也就只有改变字体，以及设置不同的标题和插入图片超链接，大量 Word 涉及的丰富功能往往为出版行业使用（事实上在这一方面，LaTeX 是更好的工具，将在后面进行介绍），而我们需要的其实很简单，完全没有必要等待漫长的软件打开时间以及安装庞大的软件。当我们渲染文件的需求移动到了服务器上时，这一劣势变得越发明显，

通过 Markdown 官网 ↗，读者可以了解 Markdown 的全部语法，虽然说是全部，但实际上也非常简单，基本上两三次之后就可以熟练使用。

尽管这些标记符号十分清晰，一个可以实时将这些内容渲染为可视化的文档的软件依然有必要存在，在这里推荐 Typora ↗，作为专业的 Markdown 编辑工具，可以从外面导入不同的外观模板，同时对于导出 PDF 可以做到所见即所得；而 Obsidian ↗ 相较于 Typora，导出的内容和软件中的呈现并不完全一致，不过其对于双向链接的支持，可以使得你使用特殊的语法连接不同的文档，并且构建属于你自己的知识库，这也是上述提到的记笔记软件中笔者的首选推荐。

作为我自己的一个参考，我同时使用这两个软件，在面对常规的课程作业的需求时我使用 Typora 撰写文档并且导出美观的文件，而整理个人笔记及自身的一些记录时则使用 Obsidian。

VSCode，Not IDE

在学习了基础的数理知识，并且了解了如何整理自己的知识库之后，接下来我们终于可以来了解首选也是唯一的那一把属于程序员的瑞士军刀，VSCode ↗。

假如读者刚刚开始接触编程，往往不难经常听说一个词汇，也就是所谓的 IDE，对于 C++ 来说，Visual Studio ↗ 以及 CLion ↗ 都是属于这类的内容。在 IDE 中，软件内置了需要开发使用到的全部环境以及众多额外的高端开发工具，但是缺点也很明显，太重。试想一下一个软件巨大，功能繁杂，并且就大多数功能你都用不到的开发环境，我们还是提出那个基本的需求，极简的满足我们要求的开发环境，所以全世界绝大多数成员的首选，VSCode 应运而生。

要是用某一个词汇去形容 VSCode，那么文本编辑器可能是比较贴切的之一，VSCode 并不支持任何的编程语言，也就是在你下载了它之后，你并不能直接在里面编辑并且运行 C，也不能直接运行 Python，本质上它只是一个文本编辑器而已，而恰好我们说的这些编程语言的文件都是直接的文本，只是后缀名不同。

VSCode 实际的工作原理是将你本地配置好的环境和它自身的插件结合在一起，你在本地具有 Python 环境，同时你在 VSCode 中安装了 Python 插件，那么你就可以具有直接运行程序，实现代码高亮，错误提示在内的一系列功能。

诸如此类，我们不难想象这样一个场景，当我们在本地分别为各种语言配置好了环境，并且为我们的 VSCode 安装了插件，那么一个兼容无数编程语言的强大 IDE 就诞生了，而且还很简洁。当然假如你有额外的需求，也不用担心，再安装一个插件就好，VSCode 具有极其庞大的社区，开源社区为其制作了无数的插件，可以覆盖你的绝大多数需求。

关于配置环境，尽管我们如此描述，并且形容了 VSCode 的诸多好处，在 Windows 配置环境对于新手也绝非一件简单的事情（当后续我们介绍 Linux 的时候这一差距会变得更直观），因此在这里给出教程链接。对于 C++，可以见 这个链接 ↗；而对于 Python，我们建议你配置 miniconda，见 这个链接 ↗。

Chat with LLM

大模型出现之前学习一门新语言或者写一个项目有很多方法，你可能需要查阅文档或者论坛去找到一些在浅层搜索中难以发现的使用逻辑，然后再进行漫长的代码编写和 debug，其中可能还包括大量的重复性工作，但是大模型在出现之后，一切逻辑都被改变了。

对于国产的模型，读者可以尝试使用 Qwen ↗ 作为首选，或者使用 Deepseek ↗，但是不得不承认的是当前 Deepseek 具有极大的幻觉率。在有条件的情况下，笔者依然推荐读者使用海外模型，对于处理大量内容编写，Gemini ↗（Google 的模型）是一个好的选择，而 OpenAI ↗ 的 GPT 系列在本博客写作的当下对于回复长内容显得极其懒惰，但是响应速度以及使用体感上尚可。这两个海外模型的会员费用都是二十美元一个月，对于学生党算不上便宜，但是在我的视角来看绝对是物超所值。

同时，伴随着大模型在代码编程领域的深度应用，代码补全插件应运而生，也就是在你输入了几行代码之后，模型会尝试理解并且预测你将来想写的程序，并且给出提示，此时你只需要按下 Tab 键就可以把它的提示应用到你的文件中。不过笔者在体验了诸多 VSCode 提供的代码补全插件之后，要是硬要说推荐，Cursor ↗ 是唯一的选择。

Cursor 并非一个插件，而是一个单独的软件，它的逻辑和 VSCode 完全相同，因为它实际上就是 VSCode 的套壳。VSCode 完全开源，也就是任何人都可以在它的基础上进行进一步的修改，而 Cursor 就是其中之一。它在其中内置了他们设计的代码补全以及写程序的流程框架，这套十分强大的流程远超任何代码补全插件的体验。要是用某种量化去描述，代码补全插件可以将我的编程效率提升一倍，而 Cursor 可以到三四倍，并且可以使我具有之前并没有能力做的，开发不了解的语言的小型项目的能力。

事实上本博客的大多数代码也是在 Cursor 的帮助下完成了重构。Cursor 每月可以有一些免费额度，同时付费会员是二十美金每月，同样物超所值，并且假如你只有二十美元，我建议你为 Cursor 消费，而不是大模型会员。

学会与大模型交流是一件十分重要的事情，尤其是在一切博客或者教程的撰写者都带有一定的 bias 的前提下，很有可能在你作为初学者阅读的过程中就忽然发现了自己难以理解的全新词汇，在此之前你需要在网上搜索资料，并且尝试理解，而如今你只需要打开网页，并且询问模型：「用浅显易懂的话来解释一下这个词汇的含义」，一切就迎刃而解，包括对本博客以上以及以下的内容都可以这样对待。

氛围编程，编程是一种思想

在诸如 Cursor 这样的 AI 代码编辑器兴起之后，一个新的概念开始出现，也就是氛围编程。作为氛围编程，无外乎就是使用文字的提示让模型去替你生成程序，但是其背后却隐藏着一个深刻的思想，那就是程序的语言本身只是外在，但背后的思想是统一的。

没有读者没有学过任何编程语言的情况下，理解这些抽象内容显然是复杂的。事实上不同的语言确实有不同的特性，但是对于基础的使用来说，其实都是通用的。比如他们可能都有变量和逻辑语句，都有类以及函数，都可以分文件编写。这些共同的特性组成了程序设计的思想暗线，也就是如何组织你的功能，如何将不同的模块进行封装，使得整体的项目便于维护以及继续开发，而这才是体现编程功底的地方所在。至于语言以及具体的语法，模型能完成其中的大多数，而尽管更多的需求要你在进行语言的学习，但**语言只是工具而思想不变**。

回到具体的语言学习，再根据上面的教程配置好开发环境后，读者大概率会面临本科教学中三大常见语言之一，也就是 C、C++ 或者 Python。在这里笔者并不会给出任何的教程，而是建议读者直接询问大模型并且获得答案。

编程是互联网生产力的根本所在，也是众多大模型企业集中力量去攻关的场景之一，每一个模型都无外乎具有速通你的本科语言课程的能力，而与模型进行交互式的问答，效果好过任何的单方面视频课程是显而易见的。假如你是需要满足课内需求，可以直接将你的教学大纲发给大模型；而假如只是想要去学习，「我想要学习 xx 语言，我现在是零基础，请先列出一个详细的教学大纲，然后一步一步教会我」，这是一个好的提问，也是一个好的开始。

在古早时期，对于大模型的使用存在大量的所谓提示词，可以进一步强化模型的能力，也就是所谓的提示词工程[23]，而目前随着模型的能力不断变强，往往不需要提示词或者任何精心的设计，就可以让模型解决你的问题，因此也无需担心竭虑。

遇到报错，直接将信息复制或者截图发给模型；遇到看不懂的地方直接询问；你想要的功能也可以直接让它去写，对于初学者的需求来说，模型不会出任何问题。不过读者也不要完全依赖于模型，而是要在其中尝试自己学到一些内容，因为到了更深的层次之后，模型的能力会逐渐不适应，此时个人能力的体现就至关重要了。

你唯一在深度学习中需要掌握的语言

在了解了编程的抽象概念之后，如何选择编程语言也是一个问题。对于深度学习以及人工智能领域的从业者来说，第一门语言的选择是显然的，即使用 Python。如果你从某些地方看到对于 C++ 的吹捧或者某些稀奇古怪的奇技淫巧，尽管笔者曾经也是 C++ 的忠实拥趸，但是有必要强调的是，这些内容均没有任何的价值。使用 Python，是唯一的选择。

在此基础上，我们有必要简单描述一下在主流深度学习领域中对于编程部分的技术栈，在领域的早期，各种的学习框架以及各类内容几乎是百花齐放，但是伴随着几年的迭代，社区趋于稳定，框架的选择也逐渐收敛。我们使用 MiniConda 或者 uv 进行环境管理，PyTorch 进行模型的搭建以及训练框架的搭建，对于更加现代的领域，诸如 LLM, VLM 或者 VLA，我们均使用 Transformers 库，这是一个 Huggingface 搭建的对于模型进行进一步封装的库。在此基础上，如果你对于 LLM 以及 VLM 存在一些部署的需求，目前的主流框架是 vLLM。

如果你不了解 Huggingface，这是一个面向模型的 Github 类似的内容，使用者可以上传自己的模型权重或者数据集（数据集的概念相当广泛，这几乎就是一个网盘）到 Huggingface，以实现开源以及全球的分发，除此之外 Huggingface 也形成了全球最大的人工智能尤其是模型社区。当然，如果你不了解 Github 是什么，在上面我们进行了解释，你也可以直接翻到文章底部的脚注寻找 Github 词条，或者问问大模型。

Linux (and Git)

当你看到这一章的时候，你应该已经熟练掌握了如何使用 Windows 系统，并且已经学完了一门语言，这时候开始实践这一篇章才是合适的。当然，在此之前，先看一看介绍来了解一下详情也是可以的。

有一种说法，Windows 是「弱智」的操作系统，Mac 是设计者的操作系统，而 Linux 是开发者的操作系统。前两者的描述并不准确，但最后一个却十分的恰当。市面上基本上全部的服务器都是用 Linux 进行管理，包括将来你在人工智能领域中训练模型所使用的服务器，因此学习 Linux 的基本使用是非常重要的一件事情。

笔者推荐读者可以在自己的 Windows 电脑上安装 Ubuntu[24] 双系统，我 之前的博客 非常详细的讲解了这些内容；同时直接租赁一个服务器也是值得推荐的，市面上现在大多数的显卡服务器都是按量付费，1小时也就几块钱，随用随停倒也不算很贵；而与此同时，网络上使用虚拟机安装 Ubuntu 的教程也很多。

在这里简单解释 Ubuntu 和 Linux 的关系，本质上我们在说 Linux 系统的时候，实际上是在描述使用了 Linux 内核的一系列操作系统，这些操作系统在内核的基础上提供了图形化支持以及各种各样的包，使得其成为一个真实可以供开发使用的系统。在众多操作系统中，Ubuntu 以其易用性脱颖而出成为了深度学习研究者的首选，不出意外在将来的时间里读者只会见到这一个操作系统，因此也没有必要学习其他来节外生枝。但可以简单提一句的是，对于一些更加极客的开发者来说，他们会使用 Arch[14]。

当你开始使用 Linux 操作系统，这意味着需要熟练使用命令行操作，简单向大模型提问，「向我介绍一下 Ubuntu 操作系统常见的命令行」，就可以获得你想要的答案，因此在这里不展开介绍。事实上，绝大多数时候你还是在图形化界面中按照 Windows 的操作逻辑进行操作，只有在需要运行你的代码的时候，你需要用命令行连到服务器，并使用 `cd` 移到路境下，`conda` 激活环境，并且 `python` 运行程序。

此时也可以向推荐另一个计算机扫盲的网站 ↗，这其中讲解了一些计算机的其他常识，同时简单介绍了 `git` 这个工具，这一版本管理的工具将在日后派上巨大用处，而廖雪峰专门为其 撰写的教程 ↗ 也值得推荐。

Git 是一个非常重要的工具，关于如何进行版本管理、多人协作以及参与开源社区，可以说某种程度上，在任何的编程项目中初始化一个 git 仓库都是很正常的事情，上述的扫盲网站中关于 Git 的章节 ↗ 笔者尤为推荐，这是你几乎必须了解的概念以及必须掌握的技巧。

当然，事实上上述的全部需求，在实战中一步一步向大模型提问也可以解决，所以实践出真章，在实践中学习也是一个好的选择。同时，对于想要进一步了解 Linux 系统的读者来说，阅读 Linux 101 ↗ 是一个很不错的选择。

机器学习与计算机视觉是否真的必要

笔者可以简单给出结论，机器学习以及传统的计算机视觉在大多数时候并不会起到什么作用，或者说这些内容已经通过理论完善，并且具有一定的上限。尽管有的时候可以为你设计新的算法带来启发，但是并不在我们学习的最短路线中，不过这里给出参考资料。

对于系统学习机器学习的书籍，相较于网络较火的西瓜书，笔者更加推荐李航老师的《统计学习方法》。而传统的计算机视觉相关的内容，则推荐冈萨雷斯的《数字图像处理》。

同时，尽管这些内容有的时候依然实用，但是多半在实际的使用中，我们都是以使用者的需求而非创新性的追求来看待。好消息是 Python 的众多库，如机器学习的 scikit-learn 以及计算机视觉的 OpenCV，其实都支持了这些算法，因此往往只需要简单向大模型进行提问，并且调动这些就可以实现对应的功能。

一些反对者或者保守主义者不免担心，跳过了这些打基础的环节，是否会导致将来自己学习一些东西的时候不太牢靠。尽管在笔者的视野中并没看到这样的案例，但是进行适当的学习倒也无所谓，只是看到互联网上繁多的课程，没有必要因此看花了眼，选中几个课程并且进行快速的学习即可。想当初笔者在视频网站中比较不同视频合集介绍算法的异同点并且都看了一遍，如今回过头来看，除了浪费时间，没有得到任何的帮助。

尽管传统机器学习与计算机视觉方法在当下并非主流，但在实际工程应用中依然有着广泛的使用，例如工业检测、图像处理、边缘设备部署等场景；而在理论研究领域，它们更是构建现代 AI 的基础。如果你对工程落地或算法原理感兴趣，这些内容仍然值得系统学习。

深度学习

到了深度学习的领域，说白了，你正式开始入门了，需要更深度的理解各个方面的知识，并且融会贯通。当然，这只是一个开始。

假如说你擅长英语，那么在这里推荐你观看 CS231n ↗，这门经典的计算机视觉课程现如今已经涵盖了对于计算机视觉以及泛深度学习中的绝大多数基础内容，对于初学者来说绝对是最好的教材。当然，假如你并不擅长英语，这门课程也依然推荐，请在磨砺你的英语水平的同时观看（顺带一提，YouTube 提供了实时字幕翻译的功能）。

与此同时，假如说一定要选择一个中文教程，在这里照例推荐李沐的 动手学深度学习 ↗，同时并不建议观看其中任何的动手部分以及比赛部分，而且快速过掉全部知识点。至此你便了解了深度学习的基础知识，可以投身于特定领域以及代码的学习了。

与此同时，假如说你对于 LLM 具有浓厚的兴趣，在这里同样推荐 CS336 ↗，这些内容看上去更加的专业，需要更多的时间仔细学习。

在这里笔者建议读者自行实现几个程序，比如说经典的 MNIST 识别数字，或者可以手动实现一个 ViT 来识别 ImageNet，这个环节可以在AI的辅助下完成，来确保自己已经可以熟练融入编码中带有AI协助的工作流程。不过必须说明的是，这些代码说到底只是练手而已，而实际的项目往往从需求中产生，在此之前先学一学如何了解一个新领域吧。

论文阅读 and so on

伴随着人工智能技术的发展，人工智能所能涉及的领域也越来越多，从本来的图像处理以及自然语言变到了如今越发宽广的模样，包括大语言/多模态模型、具身智能、3DV、图像生成以及其他的无数的领域，而再向下细分更是有无穷多的分枝。从这样一个庞杂的枝杈中找到自己感兴趣的方向并非一件易事，读者不妨给自己放一到两周的假，在大量的新闻中了解AI领域的各个方面，包括但不限于各类新闻和公众号的推送，以及各类的每日论文推荐，并且尝试从中找到自己感兴趣的方向。在这里给出一些推荐的信息渠道。

当读者找到了一个领域之后，往往就需要开始了解这个领域相关的论文，而弄懂一个领域在做什么，核心思想就是弄清楚当前的前沿工作都是用什么架构的模型，在怎样的数据上训练，并且存在哪些应用场景。人工智能本质上作为实践学科，是用方法解决某个问题，因此搞清楚这个关系就明白了大半，接下来就需要开始关注方法的细节，并且进一步精读论文了。

阅读论文的方法依然可以参考李沐的论文精读系列，只是众多论文不一定和你的口味，你只需要了解阅读的节奏，并且在熟悉一个领域之后开始加快这一节奏，同时他们的论文串讲做得非常不错，值得一听。

尽管前面给出了不少的捷径，但是到了最后这里，笔者却难以给出更快的方法，本身对于领域内方法的理解，就是在大量的阅读以及大量的实验基础上积累得到的，二者缺一不可，而在读者尚且没有进行实验的条件下，阅读也自然就成了唯一的途径。

对于如何获得需要阅读的论文列表，大多数流行的领域在 Github 上均存在对应的所谓 awesome list，如 多模态大模型 ↗、具身智能 ↗ 或者 Unified Models ↗，也就是领域中其他人为这个领域收集的论文推荐列表。这些列表往往大而全，但是并不是每一篇都有必要精读，一个很好的参考是读者可以前往 谷歌学术 ↗ 查看论文的引用情况。同时读者也需要紧密关注大量人工智能相关资讯，如公众号机器之心、新智元以及量子位。关注 arxiv ↗ 上的最新论文也是一个好的习惯，同时 papers.cool ↗ 是一个每日推荐论文的网站，值得每日上去刷一刷。

而当你阅读了大量论文之后，可以前往一些租卡平台，诸如 autodl ↗，使用按量计费的方法租卡来跑一些实验，将自己看过的领域内比较前沿的论文跑一跑，从而增加自己的上手经验。

实际上任何研究者都是从新手做起的，所以完全无需为难情绪，尽管实际上在这个介绍篇幅并不是很大的练手环节，才是篇章中可能带来最多挫折的部分，但是现在不同的领域有不同的代码体系，大量的经验只可意会不可言传，都是十分分散的，也是读者也只能在实践中自己摸索了。

不过假如读者在摸索之后获得了提升，并且按照笔者上述的建议进行了近期工作的汇总，将自己如何解决问题记录了下来，笔者自然也欢迎读者将自己的博客发在评论区，供后来者参考。笔者当前在具身智能领域进行了大量仿真器的开发，相关内容都在本站中汇总成的博客，假如读者恰好对这个领域有兴趣也可以前去一观。

加入课题组

在学习了大量的基础知识之后，加入课题组自然也有相关的一番学问，当然事实上，笔者并没有加入过很多课题组，但是在与很多朋友的交流中也有一些心得，并且总结了一些经验。

就像是把大象放到冰箱里分三步一样，加入课题组也有类似的三个步骤，找到想要加入的课题组，准备简历，申请与面试。

寻找课题组

找到想要加入的课题组往往是最令人烦恼的一个环节，因为剩下的两部分只是不知道如何做好，但是如何去做大致上还是有一个思路的，然而对于课题组来说，这毕竟是一个长期的考虑和决定，找到合适的往往是困难的。

以往在加入课题组之后，浪费大量时间进行打杂，甚至将时间浪费在老师评优所需要参加的竞赛中，这些例子比比皆是，而如何对于目标的老师进行检索和筛选，笔者本人有几个方法。

对于检索来说，两个常见的方法分别是在论文阅读中积累或者请教前辈。

在读者找到了感兴趣的领域，并且广泛阅读论文之后，往往对于领域内的工作会有一些自己的初步见解，并且和以简单的对于不同工作的质量进行划分。一个有趣的现象是，读者有时会发现一些优质的工作，总集中在某些课题组中进行发表，或者至少挂在某位老师名下，这通常从侧面反映了课题组对于科研的 taste 以及课题组的学术水平，可以成为是否是一个好的课题组的标准。在笔者认识的大多数精通科研的朋友们中，一个广泛的共性是大家对于科研圈的各种脉络发展以及很多人的科研主线均具有着广泛的了解。具体来说在讨论到某篇工作的时候，往往不止于这篇工作本身提出的创新点，包括核心作者本人在这篇工作之前的内容，某某工作和这篇的思想如出一辙，本身是在遵循一条一致的路线。对于这类脉络的理解，在短期内可以帮助读者了解课题组的好坏，而在长期更可以帮助读者甄别合作者的能力以及相对应的人际关系。

向前辈请教则是另一个很直观的解决思路。本校的同学可以很清楚地说出他了解到的部分老师是如何对待学生的，哪些老师是有能力，哪些老师没有资源；而对于这个课题组里面的学长来说，则可以进一步了解其中的细节，老师的为人处事以及合作模式。就像前面提到的，按照提问的智慧，礼貌的向他们咨询，并且表达感谢。值得注意的是，能够提供这些信息的前辈往往是老师的利益相关。在校的学生不敢对老师进行锐评，

课题组中的学生更是如此，甚至向你推荐一些老师，这是否意味着其他老师的能力不够，往往容易得罪人。这一事实导致大多数人并不愿意在线上提供咨询，一个合理的借口，和前辈在校园里转一转，喝一杯咖啡或者请他出去吃一顿饭，线下的交流往往更加私密，更容易咨询到你需要的信息。

通过咨询的方式可以了解到一定范围内能力尚可的老师，而那些往往能产出优质工作的课题组则门槛更高，只要你有足够的勇气，完全可以申请他们的课题组，就算失败也不会有什么损失，不过假如你决定退一步讲，那么一个问题是在若干能力尚可的老师中如何选择那位相对更好的，也就是一种对于能力的评价策略。

简单介绍一下对于老师的职位，排除各种头衔不谈，一般来说，主要还是教授副教授以及助理教授。全世界的教职制度基本都是如此，向上的升迁是需要时间积累的，而不只是成果足够，因此一个现象是，现在在学术圈中一批更具影响力，并且在工业界也颇有人脉的学者，往往都是某高校的助理教授。在一般的选择上笔者也更推荐读者前往这些老师的课题组，一方面这些老师更加年轻，甚至依然奋斗在学术一线，和自己之前就读的高校或者工业界有着紧密的联系，而近些年来出色的学者往往也有着更加亮眼的成果，这些成果在现如今有着深远的影响力，并且可以为他带来学界以及工业界的合作和资源；另一方面，这些老师和学生的代沟更小，课题组中的人员关系积累也更少，相对纯粹而没有勾心斗角，甚至还可以有机会成为老师的开门大弟子。对于绝大多数的老师，论文引用量以及项目的影响力是不深入了解下评判他们能力的唯一标准，这方面读者可以前往谷歌学术查看，通常两到三千引用已经算是比较厉害的老师，更有甚者可以更多。而对于更加年长的老师来说，五千以上的引用往往意味着他们可以在这个学校里站稳脚跟，并且提供稳定的资源，也是可以投奔的对象。

准备简历

在选择了合适的老师之后，下一个目标就是撰写简历，这其中自然也有不少的学问。通常来说简历的撰写使用 PPT 或者 LaTeX，而很少使用 Word，LaTeX 可以在 overleaf 上进行编辑，这是一个在线免费的 LaTeX 编辑器，支持在线编译。大多 GitHub 上的简历模板可以下载为压缩包，并且直接导入。

简历在编辑的过程中，自然也有轻重缓急之分，姓名在内的个人信息放在最上方，之后是教育经历，接下来依次摆放科研项目以及其他奖项，在另一方面也反映了通常来说老师对于学生的期望。科研成果以及科研能力往往是排在首位的，其次是工程能力，再然后才是其他从侧面可以体现出学生个人水平的事项。一般来说比较前期的简历可以尽量排满一页，如果有所超出，就要对内容进行适当的取舍以及压缩。对于科研以及项目的描述，往往从两个核心点出发，一方面是项目实现了什么，这里主要凸现相较于以往内容的不同以及更加量化的进展，比如说相较于之前 SOTA 模型提升了 10 个点，而非简单将项目或者论文的摘要让人工智能进行压缩，放一堆没有任何信息量的空话上去。简历在某种程度上体现了你的表达能力，而老师在相关领域内也已经足够资深，并不是使用套话空话可以骗过去的类型，因此还是拿出真东西说话；另一个方面则是突出自己在项目中的贡献点，自己在其中参与了哪些部分，并且实现了什么核心内容，这些可以体现你的能力。其他奖项则仅列举名称，没有必要详细阐述。

对于在此之前没有相关经历的同学，也就是完成了上述我讲述的这些内容，但是没有参加实际项目，事实上一封套瓷信也可以说明一切，无需附上简历。

套瓷信主要分为几个部分。首先自我介绍，并且表达在老师课题组进行科研实习的意向，以及对于老师做研究领域的兴趣；接下来进行简短的自我介绍，例如之前的科研和项目经历，或者说自己的成绩排名；再之后表现自己对于领域的了解，例如阅读过哪些论文，或者假如已经进行过科研训练，是否完成了一些项目，并

且有一些深入的理解，在这部分非常简洁的题，并且再次表达对于老师课题组的兴趣；最后客套收尾，很高兴认识老师，并且主动约会议，如老师假如有时间可以加微信并且约会议进行详谈，同时留下自己的联系方式。

面试相关

假如老师确实要和你进行面试，那么恭喜你进入了最后一个阶段。事实上大多数老师是很愿意要能力尚可的实习生的，面试多半也以聊天为主，而不是进行考核，但是还是要进行适当的准备，例如如何实现 attention 这种老生常谈的问题。

在面试的过程中，投其所好是一个关键，对于已经有科研产出的读者来说，自己的产出与老师的方向是否 match，如果 match，如何让老师认可你的工作；假如没有产出或者不 match，如何同样向你陶瓷的老师阐述你的能力是关键。一个细节是在面试或者聊天的时候不要关注我的方法做到了 XX，除非你做出了绝对领先于领域内大多数人的工作，不然即使是顶会，老师们本质上更加关注的是你的能力。这是一个很有趣的现象，在简历上一般的建议是对于你的项目给出量化的体现，比如说超过 SOTA 4.2%，但是到了面试，老师更关注你的个人能力，即，能否在老师的课题组中继续达到上个项目的完成度甚至更好。

完成了上述的步骤之后，读者就可以加入课题组，并且正式开始科研实习了。

完成第一篇论文

在加入课题组之后，下一件事情就是完成自己的第一篇论文了。

一般来说我们认为，进组之后的状态存在两种，一种是老师分配了一名或者多名师兄带你（具体的形式可能是让师兄本人和你对接，或者安排你进入师兄的课题中，看看能否帮上一些忙），另一种则是老师希望你自己独立完成一个课题。值得一提的是，在这个过程中大多数时候实际上并不存在一个所谓的状态，是被你的老师放养了。课题组并不是某种贴心的论文/科研辅导机构，你需要主动联系老师以及师兄，并且寻找自己在其中可以做些什么事情，而并不是等待他们贴心地找上你。

在这里回扣上一章节的内容，事实上在选择课题组的过程中，老师所进行的课题是否收敛也是很关键的一个评判指标。笔者的第一段实习很难说不坎坷，老师打算开展一个新的方向，并且让笔者探索，这在某一方面意味着整个课题组中实际上只有笔者一人有经验。哪些代码是好用的，有没有具体的调参技巧，哪些论文是有必要阅读的，这些经验往往只有同课题组中更加资深的老师或者前辈才可以给你指导，而假如老师研究的领域并不集中，这或许意味着你会看到不同领域的师兄彼此没有交集，所能获得的帮助也就更加分散，这并非有助于长远发展的。

回到正题，在两种参与课题的情况中，一般来说第一种是更加合理且幸运的结果，也就是有一名前辈带领你进行第一次科研实践，并且以核心身份参与到他的课题中。在这里我们会先简要说明整个科研的流程，并且在之后强调剩余需要关注的重点。不过在此之前，你需要分辨前辈让你做的事情是否是核心参与者需要进行的，而不是亲切的 PUA 你帮助课题组标注一些数据，或者进行其他的打杂工作，这些内容让你不能参与到课题的核心迭代流程中，从而不仅因为没有参与核心工作而作者位次较低，也很难从中学到任何内容。

形成 Idea

一般来说，一个课题的诞生从 insight 开始，也就是对于领域中发现了某个关键问题，或者对于有的问题找到了更好的解决思路，或者基于某些理解可以进行一些针对性的探索，并且从探索的结果中寻找思路。

在参与课题的过程中，读者有必要跟随前辈的思路，积极参与对于整体模型的迭代，并且在这个领域中尝试寻找其他有待解决的问题。在这里举例一个笔者所参与的领域的例子，也就是 VLA Manipulation 这个方向，即使用模型控制机器人（主要是机械臂）进行操作任务，例如分拣桌面上的物品。

在对于之前的 VLA，在实验的过程中不难发现，对于抓取任务来说，很多时候机械臂确实已经到达了物体的上方，但是在向下抓取的时候，因为高度不够导致没有成功抓取，从而导致了非常大的失败率。这里就呼应了前面提到的找到一个关键问题，也就是当模型只有 RGB 输入的时候，缺乏对于深度的感知，从而使得任务失败率很高。在遇到问题之后就去分析导致问题的原因，并且可以思考最直接的方式去解决，在这个问题中，最好的方式当然是接入一个深度编码器（当然事实上这里只是一个例子，类似的方法已经有其他人做过，笔者才不会心疼去讲这个）。在往更加细致分析，VLA 目前常见的架构是 VLM+Diffusion 拼接在一起的模型，本身进行端到端训练，可以简单理解前面的 VLM 负责任务理解以及各种的语义理解，甚至包括 grounding，而后面的 Diffusion 则关注于动作本身。这时需要关注的重点就在于基于领域中的方法以及一些常识进行分析，应该如何更好地实现自己的 idea。对于这个问题来说，VLM 大多数输出的是语义信息，这意味着其失去了更加细致度的表达，将深度信息作为 VLM 的输入，并且希望 VLM 将这一信息传递到后面显然是更加不具有效率的，而直接将深度编码和 VLM 输出拼接在一起作为 Diffusion 的 condition 虽然是更加直接且有效的方法。这时候一个最基础的新方法就已经出现了，剩下的对于读者来说就只需要实现算法并且形成一篇论文了。

关于代码

在拥有了 idea 之后，之后就是要进行代码的实现了。在这里一般来说对于第一篇论文，还是建议基于以前的方法的代码进行改进，在他们的基础上加入自己的创新点。一个事实是，现如今绝大多数的论文的要求不只是算法有效，更需要超过当前领域中的全部论文，达到 SOTA 的性能，以往的论文除了自己的创新点，但也包括不少的调试技巧。直接在他们的代码基础上进行修改，可以确保你的方法带来的性能改进是在当前最好的基线上进行上下波动。而假如自己从零开始进行实现，极大的概率是忽略了一些实际影响深远的技巧，从而导致难以对齐性能。

这里同样有必要提及此时此刻的代码能力，在此之前我们建议读者掌握 Python 以及 Linux 的使用方法，并且在学习深度学习的过程中了解 pytorch 的内容。假如读者处于一些主流领域，那么在拥有这些基本功，并且熟练了解 Git 以及 GitHub 之后，同样有必要了解 Hugging face 以及 Transformer 库，在一些比较传统的领域中，这些内容没有被推广，但是更多的领域内它们已经成为了绝大多数代码的基石。

不同的领域包含了不同领域细分的需要掌握的技巧，这些内容就需要读者自己去练习了，假如参与其他人的课题，代码是一件工作量很大的事，而重点在于了解自己的能力边界，并且在能力边界之内尽力去参与，而不是揽下大多数自己似乎可以去做的事情，但是最后却发现没有兑现自己的承诺。

进行实验

完成了方法的实现之后，剩下的就是漫长的实验过程，其自然包括了对于性能的调试。事实上一个悲伤的事情是，排除方法确实很有效这一情况，在大多数时候对于略微无效的方法，可以通过调试技巧使得性能超越以前的方法，假如读者认为就是有必要的，或者说大多数时候这都是有必要的，那么在性能不是最好或者不够理想的情况下，不断地在实验中迭代是必须的。

实验过程有两个重点需要注意。

其一是需要详细的记录实验每一步的 setting。一个事实是在论文写作的过程中包含 ablation study 这个环节，你需要客观分析你的方法在整体的性能提升中带来的增益，一些更加细节的参数带来的影响，以及假如这是一个更加偏向于探索，而非制作一个模型的论文来说，阐述此时此刻模型内部的运作机制，使用一些定性或者定量的实验来证明你的观点，这些也是必要的。这意味着那些即使在调试过程中性能不理想的内容，既然可以作为实验结果放到正文之中。

一个简单的例子是关于学习率的敏感性，在调试的过程中，按照数量级对于学习率进行缩放是常见的调参技巧，假如说进行了反复的调试，但是性能没有什么变化，那么这些实验结果就可以作为「模型对学习率不敏感，非常稳定」的佐证；假如说在调试中有了明显的变化，那么则可以阐述如何更好地选择学习率。

其二是在实验过程中依然关注哪些问题是存在的，老实说之前的创新点确实可能不大，后续我们会阐述针对有限的创新应该如何形成一篇论文，在这里明显更好的结果是发现新的问题，并且对于方法进行更多的补充。

论文写作

在完成了主要实验的同时就可以进行论文的写作了，这也是形成一篇论文中最后的重要环节。写作主要从三个方面进行讲解，分别是故事线、创新点以及作图。

一种调侃是当今的写论文就是在讲故事，这从某方面侧面反映了事实。所谓讲故事以及所谓论文的故事线，完全不是一个贬义词。本身论文是事关学术的，这意味着在论文中，你不是在讲述那些你费尽心思写下的代码机制是如何实现的，或者各种调参中你是多么的辛苦，你是如何使用一个 torch 的类做到了某个模块，而是需要赋予整个论文一条故事主线。继续下面对于深度的例子，这条主线可能是当前深度信息的缺失导致了模型抓取精度的丧失，从而引出如何将深度信息更合理的编码到模型中的问题，并且讲述自己论文的贡献点。

所以这里就是第二点了，也就是如何阐述论文的贡献点，一般来说论文的贡献点都在三条左右，其中可能包括一点是在当前所有的测评基准中都获得了 SOTA 的性能，但是另外两件往往事关方法本身。如何从巧妙的角度找到切入点，并且贴合故事进行讲述，这是需要技巧的。

对于上面提到的 idea 来说，我们之前讲到合理的深度输入位置，可以带来更好的改进，这里就已经是一个关键创新点了，而在此基础上需要衍生出下一个点。

例如，如果使用目前已有的深度编码器，这意味着模型都是在通用数据上进行训练的，而我们其实希望我们的模型可以聚焦于机器人领域的深度信息，这些深度信息是具有特征的，比如说对于腕部视角来说，一因为相机固连在机械臂的夹爪上，夹爪部分的深度不变，而这些可以作为锚定的信息，可以为模型带来更加准确的深度估计，那么自然也可以提出一个类似于机器人域上的深度信息微调的贡献点；又或者就像上面提到的

内容一样，同样出于让模型适配机器人数据，那是不是对于最后任务上的训练过程，依然可以不冻结深度编码器，从而自然引出了是否可以设计一个多阶段训练的策略。诸如此类的方法还有很多，可以充实你的框架。

最后一个要点在于作图，作图实际上是论文呈现中非常重要的一个环节。一种说法是，好的作图可以让你不看论文就理解方法，甚至性能的图表也可以让你不看文字就理解他想表达什么，这是一个需要长期锻炼的能力，多看多积累审美。在这里推荐使用 PPT 进行作图。

值得一提的是，读者在阅读论文的过程中，不少的领域中通常会有极其复杂的作图，若干的模块交织在一起，看上去十分有创新点。然而需要怯魅的是，事实上排除一些确实需要工程实现的论文，例如某些 Agent 论文，绝大多数这类论文实际上反而是没有核心的创新点以及故事线，而是在堆砌一些看上去不明觉厉的模块，最后也难以进行非常详细的分析，对于他们如何达成了最后的性能。

在论文的写作过程中，一个重点在于反复向前辈以及老师寻求反馈，并且进行迭代。记录他人的指导，并且详细的落实，甚至在部分有分歧的地方咨询他们的指示，在大多数时候他们都是正确的。

作为科研中新手，读者的第一次论文写作可能过于理想化，例如对于一些自己方法的 feature 进行了 overclaim，一个想法是这样讲才会更加抓人眼球或者令人印象深刻，而前辈的建议则不建议这么说。这实际上都是从以往的经验出发，过分的 overclaim 反而会以来审稿人的质疑，就像过分的遣词造句会使得文章晦涩难懂。

适当的故事是有助于表达的，但却在于如何将你的实际方法串联起来，并且形成完整的表述，而并非去讲云里雾里的新名词。让别人能够立刻看懂论文才是第一要务，也是一篇好论文的核心素养，你需要在实验部分进行细致的分析，或者在方法部分展现有趣的设计，而非堆砌废话。

与此同时，丁霄汉在 Writing AI Conference Papers: A Handbook for Beginners ↗ 中给出了一个更加聚焦到细节的论文写作流程，感兴趣的读者可以参考。

投稿流程

形成论文之后，下一个重点就在于投稿了，但事实上一般来说选择投稿的会议往往在正式论文形成之前，从而导致你需要在会议投稿截止之前进行经典的 rush，从而按时完成你的提交。

在这里简单介绍人工智能领域的投稿概况，排除诸如 NSC 这种最顶级的刊物，在人工智能领域的一个普遍现象是会议的含金量大于期刊，这是因为人工智能领域的迭代速度过快，而期刊的审稿周期往往更长，因此周期更短的会议更受研究者的喜爱，从而形成了这样的格局。甚至到如今大家同样愿意将论文投稿到预印本平台，让论文可以第二天就发布，从而进行宣传。毕竟预印本平台也可以算引用，影响力即是远远大于会议中稿（毕业要求除外）。

目前在国内存在的评价标准是 CCF 评级（SCI 分区并不在人工智能内被认可，假如有人和你强调他的论文是 SCI 某区，这往往意味着他的论文含金量较低），分为 ABC 三个等级，一般来说 A 才是所谓的顶会。当然在不同的领域中也有所例外，而 A 中也并非全都是好的。例如在机器人领域的 IROS 评级为 C，更加广泛的，ECCV 作为计算机视觉三大会，评级为 B；而 ICLR 作为顶级会议，甚至没有评级。当然也存在包括 TPAMI 在内的不少顶刊，认可度依然很高。

论文投稿首先在注册，会议中通常需要在网上搜索会议的信息进入主页，然后前往对应的平台进行注册，注册过程往往是免费的，之后你需要在截止日期之前投稿的论文。在论文正文的提交同时还可以提交附录，包括文本形式的补充材料，以及甚至可以有视频以及网页，当然在其中需要注意的是双盲，也就是你不能向审稿人透露你的身份，即使是无意的泄露也会导致被直接拒稿，即 desk reject。一些会议的补充材料需要在正文截止之后的一周内提交，剩下的可能需要同时提交。

在投稿完成之后就只需要等待结果了，第一轮审稿意见由审稿人给出，一般一篇论文会被分配三到四名审稿人，他们会在对论文给出意见的同时进行打分，之后你可以对他们的意见进行回复，解释他们认为的不足并且提供补充以及修改，也就是所谓的 rebuttal。不少会议要求回复在一页 pdf 之内，这意味着这一要求依然存在：保持表述的准确与简洁。

本质上来说，审稿人的意见仅供参考，最后的是否接受的决定则由 meta reviewer 决定，但是一般来说依然有必要进行参考，一个基准是评分是否都在 borderline 以上，也就是没有负分，对于五分制的会议来说，意味着每一个分数都要大于等于三。

一个好消息是审稿人在看到你的回复之后会再次修改分数，并且在此之后交给 meta reviewer 进行决定。

大多数论文的整体流程就是如此，在过程中你当然需要决定何时继续课题，这个课题已经难以进行或者需要撤稿，在中稿之后你才需要正式进行注册，也就是进行缴费。一般来说大多数的会议都需要至少一名作者进行正式注册，而同时你不需要让全部作者都注册。注册之后就意味着你的论文可以被正式接收了，在此之后你需要提交一个正式版本，称之为 camera ready，然后你就不需要管了。线下参加会议并非必须的，但是确实是可以选择。

以上就是一次完整科研的流程了。

在这里再次回到一开始的命题，关于是否第一个课题就是自己独立完成，其实本身和参与别人的课题没有非常大的区别，但是确实会带来更多的坎坷，这意味着上面的事情内容大多数都需要你独立完成，花费更多时间思考以及试错，不过在结束之后也可以为你带来更多的成长。

而在第一个课题结束之后，接下来就是在继续对于这个课题内你发现的问题开始下一段课题，或者重新在更广泛的领域中去寻找新的问题。

中稿之后

在论文的中稿一起投稿之后，实际上也并不意味着万事大吉，假如你确实心向科研，想要在其中做出一番事业，那么明显还有更多的事情是你需要关注的。

在程序员界存在一个广泛的故事，关于程序员与产品经理的斗争，在科研中作为初学者，并且手动实现了不少的内容之后，读者不难更加关注于代码能力，而对于那些指使别人工作的人嗤之以鼻，然而项目管理合作能力同样非常重要。

按照笔者的理解，学界存在从低到高不同的能力，这些能力做到极致都可以具有深远的影响力，当然之所以一些拥有所谓高级能力的人还会受到你的鄙夷，主要还是因为他们实际上哪一点都没有做好。

首先就是工程能力，这包括代码以及其他的计算机基础知识在内的一系列能力，比如你是否可以快速用程序实现一些表述清晰但没有开源的算法，或者是否掌握在大规模集群上训练模型的方法。工程能力可以说是一切的基石，可以加速你的方法迭代速度，甚至并行不同的项目。

其次就是对于整个领域的理解能力，这包括一些直觉，当前的领域中哪些问题是核心的，当前的问题是否可以通过某些方法解决，当前的参数应该向哪些方向调试，当前的模型中哪些组件是更加必要的。这些内容一些可以通过广泛阅读其他人的博客来进行领悟，而更多的则是建立在自己广泛的积累、大量的论文阅读以及大量的实验的基础之上的。

最后则是项目管理能力。实际上在科研领域中的管理型人才简直少之又少，不少人可以在多人之间进行一次正常的合作，但是更广泛的并行则难以处理。一个正常的现象是，伴随着对于领域的理解深入，读者会越发发现领域实际上千疮百孔，存在若干有待解决的问题，而其中也有不少的问题是读者有 idea 可以去解决的。Idea 的形成速度会更快，而在 insight 的帮助下，一些有价值的 idea 可以被筛选出来，但是却没有时间实现。这时候如何 involve 更多的人力，说服他们在做有价值的事情，让他们加入你的项目，而非合作，管理更多并行的项目，充分利用每一个人的能力，这是一门很讲究的学问。

在中稿之后，可以说你的科研之路才刚刚开始，在领域中不断探索，并且关注这三个能力，对于那些做的比你好的人，请教，或者观察他们是如何做到的，并且不断锻炼自己的能力，这就是一切的核心。

去做更重要的事情

假如读者的目的只是通过论文来完成老师的毕业标准或者通过保研考核，那么按照上面的内容进行按部就班的循环就已经足以解决读者可能遇到的全部问题了，然而笔者依然在这里希望介绍一些更多的内容，假如说你有志于科研，或者至少有志于通过科研寻求更长远的发展，那么这些内容的了解是必须的。

数量并不重要，在大多数时候是这样的。读者在大多数时候，很容易从各种地方听说所谓的 3/4/5A 的本科生横空出世，从而心生自卑以及焦虑，但是一个反直觉的现实是，数量并不重要。对于一个研究者来说，一年产出两篇论文，大概率还是会被认为水平高超，但是假如是五篇，那么水论文的帽子多半就要扣在你的头上了。

通过充足的合作而挂上部分的论文，从而为自己争取一定的引用量，这种方法固然是可取的，但是对于自己的科研取舍来说，读者需要充分的思考以得出结论，在自己所在的领域，什么是更重要的事情。

The Bitter Lesson ↗ 告诉我们，从本质上人工智能给的发展是在数据科学以及摩尔定律的基础上得到的发展，而非人工设置的先验以及方法。而文章中的那句话依然保有启发性：

> We want AI agents that can discover like we can, not which contain what we have discovered. Building in our discoveries only makes it harder to see how the discovering process can be done.

在数据科学的基础上开展的最大规模的统计机器学习，也就是预训练，毫无疑问是这个时代中最为重要的内容，也正是因为围绕预训练的见解将直接从数据本身得到。读者在从事任何的领域的时候，有必要清晰的认知到自己所从事领域的边界，频繁思考 The Bitter Lesson 会有助于读者保持清醒，清晰地认识到自己目前所做

的事情多半是比较 toy 的研究，从而距离本质相去甚远。这种思考模式可以让你频繁跳脱出当前领域的局限性，并且追求一些真正有意义的工作，从而形成自己的代表作。

在深度学习表层的构成中，数据和方法作为了各自的半边天。当你研究方法的时候，这些方法能否本质上提升计算效率，让模型可以在更短的时间内吃掉更多数据，并且可以有足够的容量，又或者更好的 training recipe；当你研究数据的时候，能否解决当前领域内数据的缺陷。这些内容往往是读者需要频繁关注的，也是领域内任何人都认为重要的。

回到数量本身，一般认为一名出色的研究者需要在他的博士生涯中做出大约三篇代表作，所以不着急，慢慢来，多思考，去做更重要的事情。

一条可能的时间线

既然给出了上述全部的学习内容，那么在这里同样可以给出一条可能的最卷的时间线来给读者进行参考，值得注意的是，这一时间线非常具有挑战性，对于大多数同学来说可以说是难以完成的，在这里仅作为参考，事实上读者可以与本时间线具有半年以内的误差。

在当前的学术环境下，论文投稿的命中与否有相当的随机性，没有任何一篇论文能确保绝对被录用。因此，一篇稿件往往需要经历反复修改与数次投稿的“滚动（Rolling）”过程，从投出到获得结果，单次周期通常为三到四个月。一篇质量正常的论文，想要成功发表，经历一到三次投稿是常态。

如果我们设定一个目标——在大二学年结束时，能拥有一篇被顶级会议录用的论文，那么便可以此为基准倒推出一条清晰的路径。要实现这一目标，按照两到三次的滚动投稿周期来保守计算，首次投稿的时间点最晚不能迟于大二上学期的期末。这意味着，你必须在整个大二上学期就已在课题组内进行实质性的科研工作。顺理成章地，你最晚需要在大一结束后的暑假申请并进入一个合适的课题组。

当然，这里需要澄清一个事实：对于本校学生而言，校内科研经历并非申请外校顶尖课题组的绝对前提。本校的声誉在多数国内顶尖高校（如清北华五）中仍具备一定影响力，因此，部分学业排名顶尖的学生，即使没有任何科研经历，也可能仅凭优异的成绩在大一结束后直接申请到这些高校的优秀课题组进行远程实习。不过，这条捷径对个人能力的要求更高，它意味着你必须在短短一年内，不仅掌握所有必需的基础知识，还要在某个特定领域建立起超越同龄人的学术视野。

具体来说，这趟旅程始于入学前的那个暑假。你需要在此时到大一上的前半学期，系统性地学完数理基础和Python编程。在学期的后半段，则应开始深入学习 CS231n等核心课程，将重心放在掌握扎实的理论与基本的编程实践能力上。进入大一下学期后，重心将全面转向实践探索。你需要花费几周到一个月的时间进行大量调研，以了解热点、找到兴趣，并在此后将主要精力投入到该领域的论文阅读中。与此同时，熟练使用Linux系统、租用服务器练习命令行、尝试拉取并跑通领域的开源项目代码，这些工程能力的培养同样至关重要，它们将为你进入课题组后提供显著的先发优势。

在这一切紧锣密鼓的准备中，切不可忽视课内成绩。尽管它在长远发展中的权重或许不是最高的，但依然是各项申请中无法回避的考察项。你不必苛求自己始终维持年级第一，但将整体成绩稳定在年级前20% 是非常有必要的。

当一切准备就绪，进入大二，你的科研之路便正式步入了正轨。在课题组中，加上初期的探索，你大概需要三到六个月来完成并投出自己的第一篇工作。投稿之后，切勿停下脚步等待结果，而应立即投身于后续的课题中，让研究的节奏持续下去。

理想情况下，当大二结束时，你已经拥有了一篇录用论文。以此为敲门砖，你便可以开始申请自己真正心仪的、计划在博士期间长久投入的课题组，并争取在暑假前往进行一段线下实习。线下实习的体验是远程无法比拟的，它能让你更深入地融入团队，与导师和同门建立更紧密的关系。

随之而来的大三，便是正常的科研产出阶段，你需要不断地产出新工作，并积极参加夏令营以获得正式的录取意向。最终，在大四进入推免季时，按部就班地填报系统，完成这趟漫长旅程的最后一步。

最后必须重申，这终究只是一条仅供参考的时间线。每个人的旅程都会有顺境与波折，这都是正常的。你完全无需为此感到紧张，只需按照这样一个大致的节奏，不疾不徐，稳步前行，就一定能获得很好的结果。

06

保研二三事

假如读者事先了解过大学升学的激烈竞争，保研自然也就是一个绕不过去的话题了。不需要参加考试就可以获得研究生或者直博生的资格，并且可以直接升往外校，听上去如此的诱人。

事实上，假如我们从功利主义的角度去划分升学，无非就是两种方向，一个是科研，一个是就业。这里的科研包括高校教职也包括企业的科研岗位，就业则是类似成为大厂的程序员。在绝大多数情况下，假如参加考研，那么走的路多半是偏向于就业而非科研，上述所讲述的漫长的学习道路自然也没有必要亲自体验一番，而是可以先享受快乐的三年大学时光，之后在最后的时间内准备考研一鸣惊人；而假如想要从事科研的方向，那么保研是一条必经之路。

关于保研，计算机保研交流群（绿群）是一个公益组织，包含数千人以及大量建议与实时信息，读者可以 前去了解 ↗。

保研流程

读者往往会听到很多名词，比如保研本校或者保研外校，或者是保研或者直博，但是假如粗略的拆解，一切保研流程，在中国大陆都可以被分为两个环节：获取保研资格以及获得意向学校的Offer。

实际上无论保研还是直博，讲究的都是一个你情我愿，也是自己的学校愿意放人，而对面的学校愿意收人。所谓自己学校愿意放人，也就是学校每年从国家收到的保研名额，会按照一定比例分配给每个专业的同学，而这个分配过程通常按照成绩进行排序，适配西交的排序流程将在后续说明。在获得保研资格之后，本质上你是获得了进入国家推免系统的资格，并且可以在里面填报志愿。而所谓的对方愿意收人，则是在系统中是否被录取，这并不是因为你需要在系统中提交一些资料并被对方认可，而是你要参加对方组织的招生活动，一般称之为夏令营以及预推免，在线下进行。

所谓夏令营或者预推免，可以理解为在先后两段时间组织的线下统一选拔考试，本质上除了举办时间之外没有任何区别。这些考试分为几个阶段，全部通过并且优胜之后会获得 Offer。首先你需要填报系统，之后对方学校的老师或者教务处就会进行一轮筛选，并且向合格的同学发送邮件邀请参加线下活动，也是所谓的入营。这意味着你在对方学校的填报系统中需要提交提供吸引人的材料，这里面成绩是最基础的，而科研或者别的项目以及竞赛则是重中之重。

在此之后，根据不同学校的安排，可能会存在笔试、机试或者面试中的一种或者多种考核，必须按照一定的规则选拔，轻松脱颖而出的人家获得优秀营员以及 offer。

当然，事实上这些活动也有客观主观之分，也就是所谓的强 com 以及弱 com，即招生委员会在整个选拔工程中的权重。com 越强，意味着招生委员会的权重越大，考核越依赖你的纸面实力以及笔头实力，事先联系老师并不能帮助你获得offer；com 越弱，意味这个考核中老师的权重越大，先联系老师并且聊得来之后后就可能能获得Offer，老师可以在考核的任何阶段捞你，也就越依赖你的综合能力，比如说科研。或者甚至你已经事先联系了老师，并且在他的课题组中进行了很长时间的科研实习，那么在老师明确给出口头Offer之后，夏令营基本上只是走一个流程了。

至此读者也就不难理解专业保研的大致情况了，而说到底，学校给予保研名额依靠的是你的绝对成绩，而意向学校给予的Offer，则更多看的是自己的综合实力。

保研细则

对于人工智能专业来说，我们作为实验班具有50%的保研率，这个比例是相较于专业内普招的同学，少年班以及其他项目则是单独计算。保研排名中的成绩由两部分组成：智育分，也就是在大学前三年的必修课考试中获得的成绩，按照每个课程的学分进行加权平均后得到的数字；而德育分则每年有基础的七十分，每学年结算一次，包括一个详细的加分规则，例如在竞赛中获奖可以累计最高加十分，参加学校组织的集体活动可以累计最高加三分。

对于比较卷德育分的专业来说，往往很多同学的成绩会超过九十分以上，而根据目前的观察来说，在本专业中成绩取得超过八十五分，就已经很不错了。在前三学年全部成绩都已经确定之后，学院的教务处会统计排名并且进行公示，其中智育分和德育分按照九比一加权。

值得一提的是两个可以翻盘的例外。其中之一是参加学校认可的部分科技类竞赛，并且获得国家级奖项，根据团队中的贡献比例可以获得智育分的加分，根据奖项以及贡献排名分数也有不同；而出国交换也可以获得三分加分。由于这一加分是直接加在学分绩的总分上，不难想象每一分的权重其实很大，即使在其他科目上落后同学很多分数，关键的加分也可以直接翻盘。

两种流派

从此我们不难梳理两种主要的保研流派。

第一种即主要参与强 com 的夏令营，你需要在前期的学习过程中保持成绩的优异，例如年级前五，同时适当参与一些竞赛来充实你的简历。同时你需要在平时经常联系代码能力，甚至参加 ICPC 在内的一系列算法竞赛来锻炼你的算法能力，在强 com 中，机试是最常见的考核内容。科研是可供选择的，可以作为加分项，而过于提前联系老师也是如此，因为强 com 重点还是在于硬实力能否通过层层选拔。这种类型适合就业向的选择，也就是大多数时候攻读硕士学位，不需要提前进组参与科研，而是关注老师是否可以放实习等内容。

第二种即主要参与弱 com 的夏令营，甚至只参加一个夏令营，即你早已经提前进组的老师所在的课题组给你发的 offer 的夏令营。这方面的发展方式则如上面提及的发展路线一样，重点在于凭借科研能力提前进组，并且获得老师的 return offer。对于这一类来说，关键的重点完全在于科研，成绩甚至仅需要保持在可以获得预推免资格即可。在这里同样给出简历上各种内容的含金量评级，与老师 match 的顶会论文 >> 顶会论文 ≈ ICPC 金牌 > 特定领域的特定比赛（如机器人领域的 RM/RC 对于对应领域的老师） >> 其他的水赛（包括各种互联网+/挑战杯/数学建模）。

当然，在这里一个事实是，出于对于教务处的安排，人工智能专业的专业实习横穿各大高校夏令营，并且难以请假，同时专业内部也并没有算法竞赛的风气，导致强 com 的路线在最近更是少之又少。而放眼更加广泛的范围内，弱 com 也在逐渐占据主流，毕竟老师希望招收和自己更加 match 而且磨合了一段时间的学生，这也是本文出于笔者个人的推荐，即按照弱 com 路线进行发展。

07 结语

在经历了三个多月的思考以及迭代之后，从 2025 年的六月份到九月份，我终于完成了这篇漫长的博客。

长久以来，我一直受到开源精神的影响，和非常多的同学交流学习感悟以及方法。信息是一个很有趣的东西，它具有极高的价值，但是却可以进行不断地复制，并且分享给更多的人，而在分享的过程中，我也可以受益。就像之前提问的智慧中提到的，我也喜欢一个好的问题，再教授别人一些细节知识的同时，他们的提问可以帮助我更好的思考，并且凝练出更加关键的信息。

与此同时，在与众多同学进行若干交流之后，我发现大多数的内容都是具有一定共性的，那些通用的经验，为什么不以一种更加通用的形式进行传播？于是有了西安交大生存指南。再时候的故事就像前言中提到的那样，一个泛而全的指南某种程度上是不切实际的，而聚焦在我擅长的领域，帮助与我更加相关的那一批人，这篇博客随之诞生。

三年时间说不上长，但在其中摸爬滚打也很难说不坎坷。不得不承认，即使掌握了一些方法，我在学习的过程中仍走了不少弯路，而后来者假如按照迭代出的最短路径继续前进，能否获得更辉煌的成就，这是我由衷好奇的。

假如你刚刚读到这篇文章，并且有志于此，那就继续努力向前探索吧，毕竟时间尚早。

APPENDIX

Change Logs

1. 2025-12-25. 修订了博客的前半段，增加了对于技术栈的描述，同时优化了文章的结构。
2. 2025-10-04. 在朋友的建议下，在文章中添加了更多的外链，它们指向了同样优秀的博客或者文章。
3. 2025-09-29. 补全了剩余的内容，包括对于科研过程的详细展开论述。对于错别字等的修正。
4. 2025-06-16. 最初的文章，将西安交大生存指南中的内容，以及关于人工智能专业详细的学习路线进行了整合，并且添加了一些关于保研以及科研的内容。

注释

GitHub 是一个基于 Git 的代码托管平台，由微软旗下的 GitHub 公司运营。它提供分布式版本控制与源代码管理功能，支持协作开发、问题追踪、文档编写和持续集成等流程。作为全球最大的开源社区，GitHub 汇集了数量众多的开源项目，是开发者进行代码共享、协同开发与技术交流的重要平台。 ↑

PR（Pull Request） 是一种代码协作机制，主要用于在 GitHub 等平台上提交代码修改请求。开发者在分支上完成修改后，可以通过 PR 请求将这些更改合并到主项目中。PR 支持审查、讨论、测试等流程，是开源项目和团队协作中管理代码变更的核心手段之一。 ↑

ChatGPT 是由 OpenAI 开发的大型语言模型，基于 GPT（Generative Pre-trained Transformer）架构。它能够理解自然语言并生成连贯、有逻辑的文本回复，广泛应用于问答、写作辅助、编程帮助等领域。ChatGPT 是目前最知名的人工智能对话系统之一，代表了当前自然语言处理技术的前沿水平。 ↑

Spotlight 是部分顶级学术会议（如 NeurIPS、ICLR）采用的一种论文展示形式，介于口头报告（Oral）和墙报展示（Poster）之间。被选为 Spotlight 的论文通常具有较高的评价，会获得在主会场短时段内向全体与会者展示的机会。这种形式既提升了论文的可见度，也体现了其在同行评审中的认可度。 ↑

Oral 是顶级学术会议中的最高级别论文展示形式，表示该论文被评为最具影响力或最具创新性的少数成果之一。Oral 通常安排在大会主会场，由作者进行约 10 至 15 分钟的口头报告，并接受现场提问。这类论文往往代表了当年研究的前沿方向或关键突破。 ↑

计算机视觉（Computer Vision）是人工智能的一个重要分支，旨在使计算机具备“看”与“理解”图像和视频的能力。其研究内容包括图像识别、目标检测、三维重建、视频分析等，广泛应用于自动驾驶、人脸识别、医疗影像分析、工业检测等领域。计算机视觉结合了图像处理、模式识别、机器学习等多种技术，是推动智能系统感知世界的核心方向之一。 ↑

自然语言处理（Natural Language Processing，NLP）是人工智能领域的一个核心方向，研究如何使计算机理解、生成和处理人类语言。NLP 涉及语言建模、语义理解、文本生成、机器翻译、对话系统等任务，广泛应用于搜索引擎、智能客服、文本分析、语言模型等场景。随着大规模预训练模型的发展，NLP 在语言理解与生成方面取得了显著进展。 ↑

CTO（Chief Technology Officer），即首席技术官，是企业中负责技术战略与研发管理的高级管理职位。CTO 通常负责公司的技术方向制定、关键技术选型、研发团队建设与技术体系架构等工作，在技术型公司中常与产品、业务密切协作。其角色介于技术专家与战略决策者之间，是推动技术落地与创新的重要核心人物。 ↑

Meta Learning（元学习）是一种“学习如何学习”的机器学习方法，旨在使模型能够在面对新任务时以极少的数据迅速适应。与传统机器学习方法依赖大量训练样本不同，元学习通过在多个任务上进行训练，学习通用的学习策略或初始化，从而提高模型在低资源场景下的泛化能力。 ↑

持续学习（Continual Learning 或 Lifelong Learning）是指模型在不断接收新任务或新数据的过程中，能够保持对已有知识的记忆，并持续学习新知识的能力。这一研究方向旨在解决传统模型在训练新任务时容易遗忘旧任务（即“灾难性遗忘”）的问题。 ↑

预训练（Pre-training）是指在大规模数据集上对模型进行初步训练，以学习通用的特征表示或语言知识，然后再在特定任务上进行微调（Fine-tuning）。这种方法最早广泛应用于自然语言处理和计算机视觉领域，显著提高了模型在下游任务中的性能。预训练使得模型能够在面对数据稀缺任务时仍具有良好的泛化能力，是现代深度学习中构建强大基础模型的核心策略之一。 ↑

前端框架是用于构建网页界面和交互效果的工具，常见的有 React、Vue 等，负责用户在浏览器中看到和操作的部分；后端框架则用于处理服务器端的逻辑，如数据存储、接口响应等，常见的有 Django、Express 等，负责支撑整个应用的功能实现。 ↑

Linux 系统是一种基于 Unix 设计思想开发的自由开源操作系统，具有高稳定性、高安全性和良好的可定制性，广泛应用于服务器、开发环境、嵌入式设备等领域，是现代互联网和云计算基础设施的重要组成部分。 ↑

Arch Linux 是一个轻量、灵活、以滚动更新为特征的 Linux 发行版，强调极简主义、用户控制与 KISS（Keep It Simple, Stupid）理念。它适合有一定技术基础的用户，提供几乎纯净的系统环境，用户通过手动配置构建出完全符合自身需求的系统。 ↑ ↑

计算机图形学（Computer Graphics）是研究如何使用计算机生成、表示和处理图像与几何信息的学科，涵盖图形建模、渲染、动画、图像处理等内容，广泛应用于游戏、电影特效、虚拟现实、CAD、科学可视化等领域。 ↑

C++（CPP） 是一种通用的编程语言，兼具过程式和面向对象编程特性，以性能高、可操作性强著称，广泛应用于系统开发、游戏引擎、图形渲染、嵌入式系统等对效率要求极高的领域。 ↑

C# 是由微软开发的现代化、面向对象的编程语言，运行在 .NET 平台上，语法风格与 C++ 和 Java 接近，广泛用于桌面应用、Web 开发、企业级软件以及 Unity 引擎中的游戏开发。 ↑

Unity 是一款由 Unity Technologies 开发的跨平台游戏引擎，以易用性强、上手快和良好的社区支持著称，广泛应用于 2D/3D 游戏开发、虚拟现实、交互式应用和教育可视化等领域，支持使用 C# 进行脚本编程。 ↑

Unreal（全称 Unreal Engine）是由 Epic Games 开发的高性能游戏引擎，以强大的图形渲染能力、蓝图可视化编程系统和跨平台支持著称，广泛应用于游戏开发、影视制作、虚拟现实和建筑可视化等领域。 ↑

GameJam 是一种限时游戏开发活动，参与者需在规定时间内（通常为 24 至 72 小时）围绕指定主题独立或协作完成一款可玩的游戏，旨在激发创意、锻炼开发能力，并促进游戏开发者之间的交流与合作。 ↑

TCS（理论计算机科学） 是研究计算本质及其基本原理的计算机科学分支，涵盖计算模型、复杂性理论、算法设计与分析、可计算性理论、自动机与语言理论等内容，是整个计算机科学的数学基础和理论核心。 ↑

Markdown 是一种轻量级标记语言，用于使用纯文本格式快速编写格式化内容，如标题、列表、链接、图片和代码块。它语法简洁，易于阅读和编写，广泛应用于文档写作、博客撰写、代码说明和 README 文件等场景。 ↑

提示词工程（Prompt Engineering）是指设计和优化输入内容（提示词）以引导大语言模型生成符合预期输出的技术方法。它涉及语言表述、上下文控制、格式设计等技巧，是有效利用生成式 AI 的关键手段之一。 ↑

Ubuntu 是一个基于 Debian 的开源 Linux 发行版，以易用性和社区支持著称，适合个人用户、开发者和服务器部署使用。它由 Canonical 公司维护，提供定期更新和长期支持版本，是最流行的桌面 Linux 系统之一，也广泛用于云计算和容器化环境。 ↑

— END —