

# DL/ML的学习路径问题，目前暂时ML的学习路径没有解决？

高效快速进步的方法：

1，作息+锻炼

2，周计划内复盘&反思改进

3，多和同行/大佬交流讨论+学习：

（1）没有对话的氛围以及条件：就主动创造对话，比如说研究生的上课环节等，各种技术交流平台等——主动去stay foolish&stay hungry，去引发话题（主动引导导向自己的问题）

线下：课题组内外、学院内外（只要是做同一个领域的，管他什么学科）

线上：各种技术博客交流网站、以及各种交流群

动态的思考以及交流的角度

（2）从现有的资料（文献、github code库）中去复现：

多种文献、多种code实际上就是静态的多种观点，可以从复现&引用中去思考每个人的思考角度以及方法等

4，讨论交流有个rough idea之后，立马执行just do it！——》细节可以后面边做边学边完善

建议添加到周计划中

有个问题：就是搞ML的话，使用R还是python，有区别吗

主要是DL肯定是用python，然后pytorch用的比较多（DL中肯定还会用到ML相关，按理来说python ML统一在一起肯定也好）

当然主要是我R中的数据科学tidyverse用的比较多了，如果ML用R也没问题。

总结就是：

涉及到生信pipeline的时候（主要是统计+可视化），用R；

涉及到ML+DL+其他泛型编程任务（广义），用python

其实选择哪个语言更加取决于你的思考方式是什么。

比如说你来自于计算机或者工程背景，那么Python就适合你，因为它起初就是以程序员的角度被创造出来，因此你会更容易理解它。

而如果你来自于统计背景，那毫无疑问R更适合你，很多的统计学出身的人会更喜欢R对待数据的方式。

Python is unquestionably more straightforward as a language in general. However, it's fundamentally a general-purpose scalar language, not a vector-data language like R or a matrix language like matlab. That fact makes the type of data manipulation and analysis that is meat and potatoes in R less convenient in Python. "Hello world" is easier in Python, but real data analysis is easier in R. I use Python for general programming, but it's just not worth the trouble to force Python to pretend to be R for data analysis, econometrics, or statistics. Python is way, way behind in all forms of data analysis. For example, Python is only now considering basic ideas like "missing" values being different from "not a number" values, which the creators of R thought of and ...  
Read more

不过对于现在的工作或者生产环境，其实掌握Python或者R的任意一种就足够了，因为编程语言不过是你思考方式的载体，而这个载体无论是C++，还是Java或者Python以及R都不是关键的，因为代码转换+对于ChatGPT+来说是一件非常简单的事情。

不过理论是理论，实际是实际，因为要编写脚本，考虑到以后岗位（工业界计算机类的岗位至少python是最低要求）

所以python的还是要兼顾

另外注意一下对应的平替模块即可：

```
tidyverse
```

数据科学全套流程，吊打

```
pandas
```

```
mlr3verse
```

全面领先的机器学习框架，吊打

```
sklearn
```

### 医学生建议先学习python还是R？



**林大郎**，颜值是第一生产力？

学r。其它领域肯定是人生苦短我学python。但是医生，乃至相关统计学科的人。就要学r，尤其医生既要看病还要搞科研，时间本来就有限。搞科研要快速的能做探索性分析+eda(和做出漂亮图表发文章，不是做爬虫做网页，那是跨行了。单说eda，基本都是传统的统计学，而这方面r社区无论是前沿的还是小众的统计学工具都能找的到，读一读文档就能开箱即用。反观python，你要不然等人从r社区把你想用的工具翻译成python包。要不然就你自己造轮子。又或者你在python调用r2py一类的转接口，那你为啥不直接用r。对于医生的使用场景来说python很多优势是冗余的，而r的很多被用python的人吐槽的缺点，对计算机方面相对不是那么专业的人来说却是可以忽略甚至是优点。就好比，很多人吐槽r-base+不支持引用，所有赋值都是copy，速度太慢。但问题是一个只关心计算结果的用户，甚至他们数据量顶多也就百万级别，在当前的硬件条件下，一点速度和内存效率的损失带来的是低的学习成本从而快速把注意力放在自己业务的核心问题上，何乐而不为呢。

<https://www.zhihu.com/search?type=content&q=机器学习用R还是python>

轮子自然不用重复造，但是就算拿造好的轮子，也不是每个人都能组装成汽车的。

2021-05-01

● 回复



**iknowyouknowso**

真相了😂

2023-04-03

● 回复



**别拿钢钉当大炮**

真理

本来就是这样的啊，不止python,所有编程都这样的，因为你调用的包里也调用了一堆别人的包😏

这就是面向对象的核心思想和终极目标-调包

太对了，真香（封装好的）。自己再折腾也没封装好的好用😏😏所以，做好两点就够了。

一是封装好的有哪些，能做什么，熟悉进出。熟悉它们的调用。

二是做好自己的衔接代码，把各种库包块串好能用。

2022-07-02

● 回复 ❤️ 12

**舞雩;**

...

发论文需要创新，工程只看效果😂

## 研一在读，代码完全不会，怎么入门深度学习？



younger

其实根本不用学，如果你以后不从事机器学习这方面的工作（比如我，现在依旧在做后端开发），你只要能看懂代码就行，就是你找一篇领域内比较出名，并且开源的作品，克隆下来看代码，不懂的就问gpt，然后通过改这个代码来实现自己的想法，我就是这样做的，并且我看大部分人都是这样，没必要花太多时间在这上面。

当然，这都建立在你本科是学计算机的基础上，你学过java或者c的话，python其实也很容易懂。

编辑于 2024-09-12 18:11

## 机器学习如何快速入门啊？



周小小， 视频图像处理/模式识别/机器学习/情感计算/很菜但长得帅

看西瓜书，看csdn已经能让你水毕业了，你还想快速啊...我现在有套快速方法，复现论文贼快，但是告诉你你也学不会啊。

建议按照别人的建议，从西瓜书入门，最重要的是，快速学习有一个地方很关键，弄不明白的地方，不要纠结，动手写代码，从结果倒推原理，更重要的是，所有的一切实验不以教程里的小数据集为基础，而是以你毕业论文的数据集为基础，在学习的过程中累积对你毕业论文领域的了解。

第二，不要专门花一学期去学Python，要学会看各个库给的文档，或者ctrl+左键进去看定义和说明，反正我是没有专门学过Python，只要学过其他语言，肯定很容易上手。

这样应该是我作为普通人能理解的最极限的快速了吧。

### \*\*还有DL的学习路线：

李宏毅DL（包括HW吃透，理论）——》小土堆pytorch同时学习框架工具（pre-实践）+对照直接上手pytorch官方文档——》直接上手读生信方面的DL论文（参考本人CSDN上的寻找code博客）+github model复现内化学习。

至于是否要插入李沐动手深度学习（pytorch实现版，github）+使用pytorch实现各种DL模型（github phil WANG），看自己实际需求；

## 整体上可以偏向李宏毅网课+李沐动手DL（pytorch版本）\*\*

法就会很自闭。工具的话：推荐先学Python，然后是pytorch（拒绝一切花里胡哨的，官方文档是最新最权威的，建议直接看官方文档）。tensorflow、mxnet、paddle啥的用到再说，没必

好的，谢谢，现在就在看基本原理，然后准备上手师兄留下的代码....对了，请问下如果将输入三轴加速度数据改为六轴数据的话数据处理是不是要作大改动呀🤔

2021-08-22

回复 喜欢



**匿名用户** 提问者 ▶ **周小小**

...

大佬，那一般网上找算法代码都是上csdn和github吗？最近学keras和sklearn有点吃力，看师兄代码感觉都是调参。。。

2021-09-07

回复 喜欢



**周小小** 作者 ▶ **匿名用户**

...

嗯了，代码能抄就抄，除非你像我这么熟练了，可以比较快的复现别人论文，抄完代码后，针对你自己的数据改一改，预处理一定要写好，这个没得抄，然后再是调参。预处理比调参重要，预处理决定下限，调参决定上限。

2021-09-07

回复 喜欢

ML的学习路线：其实这里指的应该是传统机器学习，就是一些统计算法之类  
吴恩达理论网课——》（同时python的基础语言进行复建，不用另外花时间学，直接用到复建即可，不要浪费时间）——》问题是实践呢？

主要是缺一个可以调包sklearn的实践锻炼（demo等project）



wzl 作者

...

一定的数学功底是指高数1、线性代数和随机过程能及格。按照我的理解，你的需求应该是正确的调包。这个可以快速入门，scikitlearn包含了绝大部分机器学习算法，看看文档和demo应该很快能入门。深度学习也差不多，在GitHub找相应算法的实现，改改也是可以用的

2019-07-10

回复 1



小秋 提问者

...

谢指教。你的理解是对的。我只想将已有的代码套入我的数据里，至于他们怎么写出来的，我没有力气去学了。

2019-07-10

回复 喜欢



小秋 提问者

...


谢指教，首先基础医学生不是医学生，我本科是制药工程，高等数学，概率论数理统计学过一些，虽然已经忘得差不多了。然后，我对编程语言的开发不求甚解，我只想把我的数据应用在已有的代码里就行了。🙏

2019-07-10

回复 1

## 入门机器学习/深度学习要多长时间？



叶紫枫 , 浙江大学 计算机科学与技术硕士在读

第一个月李沐<sup>+</sup>动手学深度学习，b站有视频，一个月过完 + 打两个Kaggle作业，顺便看看其他AI<sup>+</sup>相关书籍陶冶一下情操。

第二个月找一个你这个细分领域方向的SOTA<sup>+</sup>，或接近SOTA的工作，必须要有能跑起来的源代码，在它基础上看代码，改代码（改变量名、函数名这种你看着不顺眼的，主要是为了熟悉），调调参数，改改loss。4周精读4篇前沿论文，对论文中没见过的概念和知识点递归式<sup>+</sup>学习。

然后你对这套代码比较熟悉之后，对这个领域有个大概了解后，去看这个领域之前发展的脉络，早期的经典工作，这个领域其他和你比较接近的分支的论文【简单】看一下。看个十篇论文下来，肯定就对细分领域有基本理解了，可能也有几个idea了，先从简单的idea试起，改一点小结构，想一个小聪明机制。

深度学习，在2023年这个时间点几乎是完全的技术学科而非理论学科。实践大于一切，看书用处有限。

编辑于 2023-11-28 10:36





He小鹿

只有我觉得沐神讲的太高深了么

2023-11-28 · 四川

回复 5



吴从周

李沐最大的价值是他讲到的所有东西都有现成、能直接跑通的代码。讲原理他可能有没解释明白的地方（他也不太care这个），但是一旦给了代码，理论上你一定能完全清楚所有机制

02-15 · 福建

回复 5

刚刚研一，深度学习直接上花书吗？需要先学机器学习吗？感觉东西很多，不知道该怎么去学，有没有学习路线？



子春之酒，强烈推荐《周四推理俱乐部》三部曲

首先，不建议直接上花书，深度学习重点还是在工程应用，而不是理论推导。个人觉得通过上手感受和熟悉比看书快速和有效得多。

机器学习还是要有基本概念的，所以推荐通过吴恩达+的机器学习系列去学习掌握，但是可以抛弃很多不必要的部分，重点关注神经网络+这部分内容。这些系列视频在小破站上都有，原始网址是 [Machine Learning Specialization - DeepLearning.AI](https://www.coursera.org/learn/machine-learning/specialization/deeplearning-ai)。

接着就推荐李沐大神的动手学深度学习系列了，绝对适合初学者入门掌握如何运用深度学习技术，B站官方号是跟李沐学AI的个人空间-跟李沐学AI个人主页-哔哩哔哩视频。这本书的官方网站是《动手学深度学习》— [动手学深度学习 2.0.0 documentation \(d2l.ai\)](https://d2l.ai/)。

以上就是我对个人入门的初浅建议了，至于往后如何发展，还是看自己的想法。希望能帮到你。

我认为ML还是有必要学的：

尤其是生信是一个和生物学联系非常密切的领域，绝对非常依靠biology的domain knowledge，

所以：



## 现如今深度学习这么火，机器学习还有研究的必要吗？



易显维，系统之神与我同在，竞赛top选手，全栈算法研发

当然有研究的必要啊，它是两种不同的方式，相当于是深度学习，这个东西叫做[联结主义+](#)，它是一种神经网络来实现机器学习的相关方法，但是呢机器学习当中呢，还有很多其他的东西也是很有用的呀，像[决策树+](#)，啊像这个常见的各大竞赛平台上面刷榜单的算法，什么xgboost啊，GBDT之类的方法。

他每一个算法都不是万能的。也不是你有了神经网络之后，传统的机器学习就没有必要了，你就说结构化的数据挖掘，这个问题当中表格数据的预测当中，以机器学习传统的机器学习尤其是特征工程这一套方法还是很有用的。其实知乎上面也有一个讨论说，为什么在这种表格的机器学习上面神经网络模型到现在是打不过啊，传统的树模型的，你可以去看一下这个问题的答案。

另外一个呢，就是深度学习，实际上解决的问题呢是一个。叫做自动表征的问题。

但是在很多的业务当中啊，你要构建有用的特征，真还不是靠深度学习能够学到这个有用的特征的，它特征呢，它是人对业务知识的一种理解。如果说这种业务知识的理解能够靠神经网络自动的去学习的话，已经实现了[强人工智能+](#)。

我以前在银行的时候遇到过一个什么问题呢，你比如像你要做风控，你风控里面有一个重要的特征，就是那个得分卡，实际上得分卡呢，是业务人员根据它的业务经验学出来的或者叫做设计出来的。特征工程方式，那这个就和业务有关联关系。

你没有干过这一行，你是不知道他这个得分卡是怎么算出来的。

你又比如说我之前还做过一个什么业务呢，还要算那个厄你的驾驶行为风险，这是一次比赛的那当中价值行为风险的输入呢，有速度经纬度之类的特征，但是你要怎么样衡量这个速度呢？后来我想了一下呢，其实你的速度快不快，跟你的周围的车开的快不快是有关系的，那这样的话就要把地图整个的切片，所以呢，你直接把这个速度那一列放到神经网络当中去学的话就学不到什么东西了，他需要人对这个数据有一定的理解。

编辑于 2022-10-15 15:54

## 学深度学习需要把机器学习学一遍嘛？



经常肚子疼，普通打工仔

不用，直接学就行了，深度学习理论很简单，看斯坦福那个课就可以。机器学习内容太多了，而且很多模型原理比深度学习复杂得多，很难搞懂。

深度学习主要复杂在结构，各种魔改网络结构，光是比较新的Transformer类这个分支，下面衍生模型几十上百种。这个就按照应用场景看就行了，不可能都学完。

所以ML的实践项目问题：

## 准研一，导师让学习pytorch，之前没接触过机器学习，是直接学pytorch还是先学机器学习？



Zeurd，狂读paper中的cv打工仔

只问一个，想读博么？如果要读博，建议别看什么机器学习西瓜书了，研三上申博士，研二结束至少得发个两篇吧，论文送审也很长时间的，研一基本就得有思路开始准备做了。

说实话，计算机这个东西，大家都开源的，而且时代飞速发展，和基础学科不一样，那些东西你十年二十年都不会变的，学了就学了，而且你也不会让一个数学系研究生从研一开始学高数不是。在计算机领域，不跟着最前沿的东西看，你从基础学，追不上最前沿的，那谈何发论文？去年clip刚出的时候，我们组里想跟进做点工作的，结果上海封市了，停了两个月，clip已经不行了，dalle2出了，又过了两个月，chatgpt+也出了，这些年ai界的成果是以月计的，甚至前年vit刚出的时候，跟进都是按天出的，你说你还从头学机器学习，哪里这么多时间给你补基础？

且不说深度学习基本都开源的，你拿了模型之后，搭积木的结构改起来并不吃力，也不算深度学习现在基本用的都是transformer+的结构，你之前的cnn，lstm+读来读去其实能用到现在的东西并不多，更别说机器学习里的算法都是sklearn集成的，一行代码就解决了，你在那儿吭哧吭哧搞半天效果还不如别人调库。

计算机能不停发展的原因就是有一个理念，不要重复造轮子，所以前人不断把东西集成简化，就是希望后人可以在他们的肩膀上走更远，你在哪儿说不行，我就要自己造轮子，我还要连路也一起修了。那我再说一句，你造完轮子过了几天，真的还记得轮子怎么造嘛？

我不知道为啥这么多老师都让学生从基础学起来，我现在研二，研一的时候上学期在学[运筹学](#)<sup>+</sup>，后来里面的数学有点过于难了，琢磨了半年没想到啥写论文的idea，下学期转学深度学习，两个月啃西瓜书，把所有的公式手推了一遍，可以自己算[svm](#)<sup>+</sup>，两个月学深度学习基础，发现花书上的代码都work不了，各种版本问题导致的bug，然后自己开始[github](#)<sup>+</sup>找基础模型自己写，从头搭了[resnet](#)<sup>+</sup>，vgg，googlenet等等一堆基础模型，但是有什么用？等我研二真的开始做科研了，准确的说是研二组里来了个比较热心的博士，会主动教学怎么做科研之后，我才发现去年一年干的事都是在浪费时间，研一新入学的学弟一个暑假看得已经比我前面了，他们聊的都是什么st，[扩散模型](#)<sup>+</sup>，什么多模态任务，我就觉得太可惜了，要是早一年知道，我现在这时候应该已经写了三四篇论文了，而不是和现在一样，刚刚投完一篇第二篇刚开始写。

### 学深度学习有没有必要把传统机器学习算法原理都学一遍？



**哥廷根数学学派**，与现代信号处理，机器学习，深度学习，故障诊断那些事

如果你是研究导向，那没必要把传统机器学习算法原理都学一遍；你说了你是就业导向，对发paper没太大兴趣，那估计要学很多机器学习的东西，因为面试肯定要问这些，毕竟目前机器学习在工业界应用还是非常广的。

- 1.在实际生产环境中，样本获取的成本有时候还是蛮高的，
- 2.很多领域对可解释性的要求较高。
- 3.不是每个公司都有很强的算力的，算法的使用场景是经不起等待的。

发布于 2023-01-16 22:52

## 机器学习和深度学习哪个才是大势所趋？



山新

有没有大佬来解释一下机器学习和深度学习在应用效果上的区别？

私这样认为：

1. 机器学习更多的是关于线性代数<sup>+</sup>、概率论、统计数学方面的知识，旨在挖掘数据高维度的信息，难点在于数学。
2. 深度学习更多的是训练数据的堆叠、硬件的堆叠，旨在通过暴力的搜索和训练将数据的高维信息展现在w、b等参数上面，重点在于训练速度。

严格来说深度学习是机器学习的一个分支。个人认为：深度学习并不需要复杂的数学推导公式，很适合小白入门，只需要了解个大概便能理解整个神经网络训练<sup>+</sup>的过程。

以本人狭隘的了解看来，目前深度学习的发展在于：神经网络模型<sup>+</sup>的搭建、硬件设备的加速。

建议直接看sklearn文档上手，如果是深度学习就看pytorch的tutorial。可以配一本统计学习方法查阅(好像有第三版了)，或者去搜一下动手学深度学习的pytorch版。  
喜欢数学的话，可以看foundation of machine learning。

所以可以直接上手使用ML的demo

综上：

**DL：李宏毅DL网课（理论）——》结合HW，小土堆pytorch+pytorch官网文档——》（李沐动手DL pytorch版，github pytorch实现各种DL模型 phil wang）——》生信DL论文（找原code）+github model复现内化**

**ML：**

**吴恩达ML理论——》sklearn官方文档调包+实际项目实践？**

**or**

**sklearn官方文档调包实践（吴恩达理论碎片化t补全）**

我觉得恰恰相反，python我是学过的，那就应该把基础python语法当做是边用边复建，所以我应该直接上手python的ML.

可以参考源友们的建议以及经验等

另外在某水上也遇到了同样在生化组搞计算机的，不过是研0

如题，楼主准备从头开始撸机器学习。（研零开始是不是有点晚）

主要是因为组内导师纯放养，整个组除了我是电子信息其他同学全是生物/材料/化学准备自救一下，当然也有可能自救失败。

不过先学一学再说吧😭😭😭😭起码努力努力

或者源u们有什么建议吗。

先坐地铁回闵行了，回去后记录今天的学习内容

另外，诚邀懂这方面的同学与我一起学习，进步，也避免踩坑！

这两天反反复复想了很多出路，包括跟着组里的生物学长做一些生物实验。（因为我们实验室数据都不好要而且组里的同学不太愿意分享数据）

最后还是决定先学习机器学习，然后再自己读读文章，实在不行学完代码出去找实习。只能说活下去还是很艰难。

今天把源友推荐的所有学习深度学习的方法都看了一遍，真的非常感谢各位源友，最后我还是选择中文的李沐学长的动手深度学习。

然后就是从零开始，今天给电脑装了conda，装了pytorch包，然后我的cuda是12.3，但是上面只有12.1和12.4，后来才知道包是向下兼容的哈哈哈哈哈。

今天大致是搜索了学习方法，资源，选择了适合自己的李沐学长的课程，然后安装了环境和包，明天开始码代码！

但是浏览各种网站，攻略，配环境真的累死了哎，感觉一天的精力在通勤和浏览攻略后就不剩多少了😓

如果lz租云服务器的话，不用怎么配环境的。（不是打广告啊👉只是安利一下本懒人的做法

day2: 今天找两个老师聊了聊，一个老师让我看一看一个2017年的，2020年的俩ECCV的文章，然后要求我复现代码，把pytorch换成别的框架复现一遍，另一个老师手里倒是有一个项目（公司的项目），但是我觉得我的能力不足，目前没有想法。然后老师让我学会用git，于是我搜了搜教程的基本上是学会了git怎么用。

下午跟实验室师兄要数据，师兄是做生化的，拉着我开了他带的prp组会（线下），然后跟我说数据不需要处理，绝了

刚刚跟前位老师发了邮件说尝试一下，目前还没消息，不知道老师最后会不会同意

在包图坐了一天，腰酸背痛的，但是感觉自己一天效率好低啊，说是要看李沐，结果今天一天就看完了前言。。。大部分时间都在git上和和老师/师兄交流上。

看着要复现的文章真的头大，这么多文件夹，labml\_nn里这么多我可能一眼看去怎么跑起来都不太完全理解，待我研究研究

day3:

早上，让我代码复现的老师跟我说可以让我用他们实验室的算力，然后我问怎么申请，老师就没回话了，可能是需要等明天？

下午去徐汇开了大组会，然后看到老师的头发已经开始白了，老师在组会上跟我们说他今年真的很忙，太多行政上的事要处理，也许他真的是个好的老师，只是我没赶上时候。又或者他真的是个很好的老师，可惜没有办法指导我。很遗憾。

晚上从徐汇回来开始看李沐的书，2.1数据操作看完了，然后写了个jupyter文件，明天再接再厉，争取把第二章看完！

偶然看到和DL相关的：



输入输出怎么读取和保存，预训练怎么加载和调整，train，test 怎么调整，怎么做可视化

其实比

L G1\_Michigan:

在网络结构里简单改改卷积层，加点简单的attention之类的

要简单吧，因为上面那些都是所有框架差不多的（datamodule、tensorboard之类的），下面这些才是对网络真正产生影响的改动，怎么改会有效不仅需要对问题足够的理解，有时还需要科班知识（比如cs231n）。不然乱改就纯炼丹了 😊

pytorch的文档多看看就行

更简单的方法是在gpt上传代码，然后让它告诉你每个地方是做什么的（也可以问它怎么改）

另外建议用逐步调试的方式跟着代码走一遍

建议尝试利用copilot+chatgpt开始一点点复现顶会论文中设计的模型（直接参考部分封装完整的代码可能很难提高自己码力），然后尝试借助类似mmdet这样框架写点自己的项目，就我个人体验copilot对调参、可视化之类的接口封装得比我手写的强多了 😊 很值得学学

copilot在bing或google搜索github student developer pack，使用交大邮箱通过学生认证后就可免费使用，在vscode, clion都有插件

gpt要么可以找老板问问愿不愿意掏20刀的订阅费，要么gpt3.5也能凑合用（最近发现win11集成的copilot好像还挺好用）

我有一个问题，就是很多时候根本不知道代码可以这样写。查文档只能是知道某个函数或者类去看怎么用，解决**怎样写**的问题。如果不知道某个操作，也无处查文档。

我知道的解决方案只有大量看别人的代码，问问源u还有什么别的方法吗 😊

gpt.....把你的需求输进去

你不一定要知道实现方式，只要知道需求就行了。

gpt不一定是完全正确的，但是可以根据gpt的思路去进一步编写或者搜索。

numpy pandas这种基本能百分百解决，比较冷门的包就只能看个大概了



我给一个自己试过感觉很实用的建议（适合我这种菜鸟）：

- 首先，带着目的去学习，而不是从头看 python 的书或者看文档。比如 lz 现在至少有一个课题的方向，这个方向内，找到一个**代码较为规范**的项目，把它作为自己之后实验的 pipeline 参考；
- 其次，功底很差的情况下，没别的，就是从 main 开始，自顶向下，**一行一行抄**。抄的过程不是单纯的复制粘贴，而是你每一行代码，都要尽可能地深入思考：（1）它为什么这么写？如果是你想实现这个目的，你会怎么写，或者怎么优化？（2）它用到了什么语法或者技巧，是值得反复用的？这时候就要记录下来作为自己的代码模板；如果抄的时候遇到实在不明白的地方，由于有 chatgpt 的帮助，你可以一行一行地让它帮你解释。如果是很依赖上下文的代码段，可以先放一放，等第二遍/第N遍写的时候再解决
- 【为什么还会有第二遍/第 N 遍】因为项目是一个整体，至少几千行代码，一次从头到尾抄写完，弄明白是不可能的，很多东西都是越往后写，越对前面的处理豁然开朗。加上自己之后可能对这个项目修修剪剪，添加自己的东西，要做好多写几遍的心理准备。
- 抄了一段时间后，至少已经自己写过一个完整的模块了。这时候，**一定要运行，自己 debug**。这个过程是提升能力的重点！要**模块化地 debug**，因为这样更方便定位哪里抄错了。这里的抄错，就帮你找到了值得关注的细节，比如有没有 to(device) 呀，这些自己写的时候容易忽略的小细节。有时候，遇到版本变化或者其它本地环境，数据格式的问题，你将会不得不自己去改写代码，这时候要坚持下去，努力尝试和思考，直到跑通一部分。不要放弃，解决每一个 bug 都特别特别有成就感。
- 【关于 debug】不要心急，最好是能把每次关键的 input, output 可视化一下，比如 dataset, dataloader 每次取出来的数据格式和样子，输入到网络里的又是什么样子，等等。这样就能理解为什么别人的项目里会加入什么 view, squeeze, asint, 切片之类，很难具象化的东西。
- 最后，坚持把整个项目抄写完，全部**跑通**。其中数据的处理，每个模块的输入输出等等，它是怎么封装的，这些宏观上的问题都要弄清楚。这时候，至少这个项目你已经非常熟悉了，而且对 pipeline 有了一个非常踏实细致的了解。

上述过程，只跑通一个项目，对能力的提升都是巨大的。我花了三个月，用上面的方法完整地理解了两个大项目，感受就是写代码非常爽，一点也不畏惧什么复现，融合这些。多动手就不焦虑啦。

不管啥方向应该有个两三年内影响很大的 backbone 文章，大家都在用都在魔改的。你直接把他 github 项目 clone 下来，代码吃透，然后再魔改就行了。第一篇文章这样，第二篇文章在第一篇的代码基础上魔改，依此类推

你写新的文章总有 baseline 吧，开源的 baseline 就是你学习的对象，根据论文去找最核心的类，然后对照论文描述去读懂它，不要直接看代码不看论文，不然你可能很难看明白在做什么。最后代码一般也是以这个 baseline 为蓝图去魔改，很多函数不需要自己重新写，顶多根据需要调试一下。

