我的深度学习之路

作者：沿途的笔记，西安电子科技大学人工智能学院研三学生，发表论文两篇，专利三项，参加遥感领域国际竞赛取得第一名成绩，曾在商汤科技实习，秋招 offer 签在华为 Cloud BU。
GitHub 地址：https://github.com/computervisionlearner/Start_DeepLearning

自 2016 年阿尔法狗问世以来，深度学习的地位无论是在学术界还是在工业届都逐步攀升。从事深度学习岗位的工程师们的薪水也水涨船高，拿今年秋招各大互联网巨头给应届生的待遇便可见一斑。

互联网大厂2019届校招薪酬

来源：100offer

公司	研发			算法		
	白菜价	sp	ssp	白菜价	sp	ssp
阿里	24w	27-30w	60w	26w	29-32w	40-72w
腾讯	20-23w	26-31w	40w+	30w	35-40w	45-70w
百度	22-23w	28w	/	25w	29w	47w
字节跳动	32w	35-41w	个例差距大，原则上不封顶	32w	35-41w	个例差距大，原则上不封顶
美团	22w	26-29w	31-35w	/	26-30w	50w
滴滴	24-27w	30-32w	/	30w	35w	40w
华为	21-24w	22-26w	/	25-29w	31-44w	41-46w
行业平均	22-24w	26-32w	35-60w	25-30w	30-44w	40-72w

重赏之下，必有勇夫，所以很多学数学、机械等其他专业的同学也想转行加入深度学习的大浪潮。与此同时，过剩的人才也导致了该领域的竞争空前激烈，更有甚者传出某企业的 CV 算法岗投岗量与 HC 比达到 100:1。（https://www.zhihu.com/question/286925266）

某四个知名互联网企业简历投递量与 HC 比

	算法岗投递量与HC比	工程岗投递量与HC比
A	10:1	移动端的投递量<HC数量
B	15:1	11:1，其中移动端3.5:1
C	40:1	/
D	100:1（计算机视觉岗）	4:1（移动端）

现在，关于深度学习入门、进阶的教程琳琅满目，也有不少大牛撰写他们的经验感想。但有的都太高端，让很多刚入门的同学们甚至是跨行的同学们望而却步。基于此，本菜鸟将从自己的实际情况出发，写一篇接地气的学习深度学习的历程。

注，本帖旨在非计算机/AI 专业的同学，或刚接触该专业的本科、硕士生。这是一篇偏实用的贴，不适用于理论高超，知识完备的大佬。

0.入门

1).入门的素材要精准简洁

刚接触深度学习的同学可能会“贪多”，搜集整理很多资源，类似于“机器学习从入门到精通”，“XXX 大学机器学习公开课”等。先不说这些资源的完整性，单就内容量来讲，这几十个 G，甚至是几百 G 的学习资源，也会让人晕头转向。因此选择精准的入门教程非常关键。

在这里我推荐的视频资源有：coursera 上的吴恩达大神的机器学习，以及 Hinton 大神的 Neural Networks for Machine Learning。含金量高的书籍有：周志华老师的机器学习，Goodfellow 大神的“花书”《深度学习》。这一部分主要是用于夯实基础，了解相关的概念。虽然本文的主题是深度学习的学习历程，但机器学习与深度学习有着千丝万缕的联系，最好还是要一起看一看。

2).勤做笔记，动手推导

在学习以上推荐的两个视频资源及书籍的时候，会涵盖高等数学、矩阵论、凸优化、概率统计等知识。我们不仅要熟悉上面的理论（如机器学习有：线性回归、逻辑回归、决策树、支撑向量机、贝叶斯、聚类、PCA、稀疏、马尔科夫、强化学习，深度学习有：CNN、LSTM、word2vec、反向传播、梯度下降，ps 感觉深度学习有主见脱离机器学习的趋势），而且对于重要的理论上的数学推导，一定要一步步的验证，能根据书本（或视频）一步步走下来，遇到知识盲点或者遗忘的部分，就**针对性的**查阅资料，力争不放过任何重点。这里需要提醒的是，**不建议先从头开始学习这些数学知识，因为时间周期太长，中途可能仍会遗忘**。

1.确定研究领域，混迹相关网站、论坛

完成上部分的阶段之后，那么恭喜你，进入了学前班水平。此时的你基本算是对机器学习、深度学习有一个较为全面的了解。**上面所提到的资料不要觉得看一遍就完事儿，可以多看几遍，甚至有实战经验之后再反复看，保证你每看一遍都会有新的体会**。

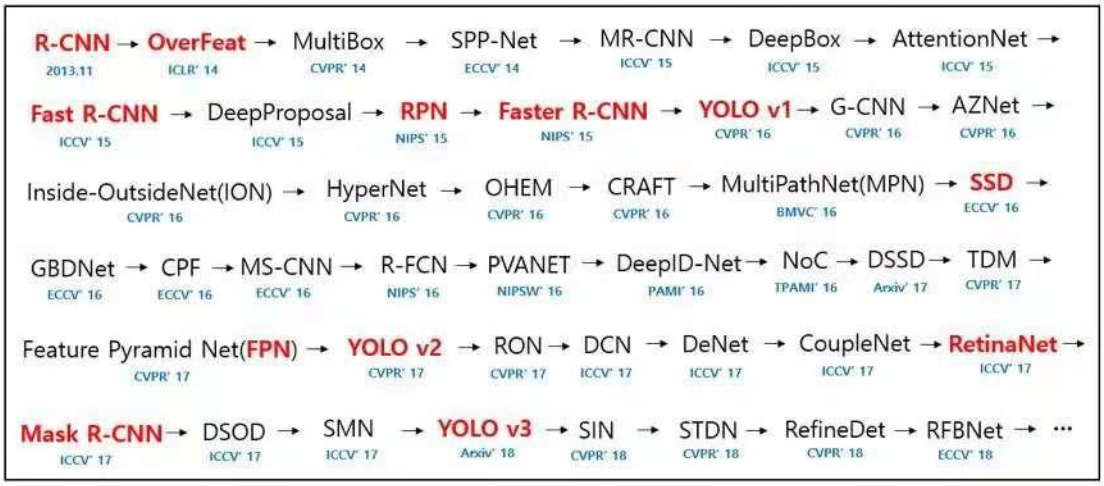
深度学习是一个非常庞大的系统，涵盖的知识面很广。个人觉得学习这一领域，要先找准自己感兴趣的方向，从此方向入手，然后再由点及面（建议以 CV 的目标检测为切入口）。

1)**学习经典主流**深度学习算法

为啥要这样呢？因为“经典主流”的东西，理论完备（因为有的太新的东西，可能只是为了水论文，而非真正有益处。时间积淀下来的东西，才是好东西），积累的资料多，研究的小伙伴多，碰到问题可以找到前辈们相应的解答，而如果一开始就研究小众的东西，可能会陷入闭门造车的尴尬处境。

针对某一方向找一些综述看看，了解该方向的发展脉络，初步了解每个阶段的改进的内容，及当时想到这个改进的初衷。然后再深入一线去看相关论文。以 CV 中的目标检测为例，它的发展历程如下图所示。

目标检测发展史



我们可以看出，现阶段在视觉顶会上关于目标检测的文章就有这么多，如果面面俱到，则不

易消化理解。因此在追踪某一方向的文章时，还要**重点突出**，比如红色字体所表示的算法需要精读，而其他检测算法则有所了解即可。更具体的，了解传统 DMP 算法及主要缺陷，然后 two-stage 思路发展到 RCNN 做出了哪些改进，再到 SPP-Net 做了哪些改进，然后到 Fast-RCNN、Faster-RCNN 又改进了什么，以及 one-stag 思路中 SSD 系列、YOLO 系列之间的差别。

2)解决问题、总结积累

遇到问题从论坛、博客中寻找答案

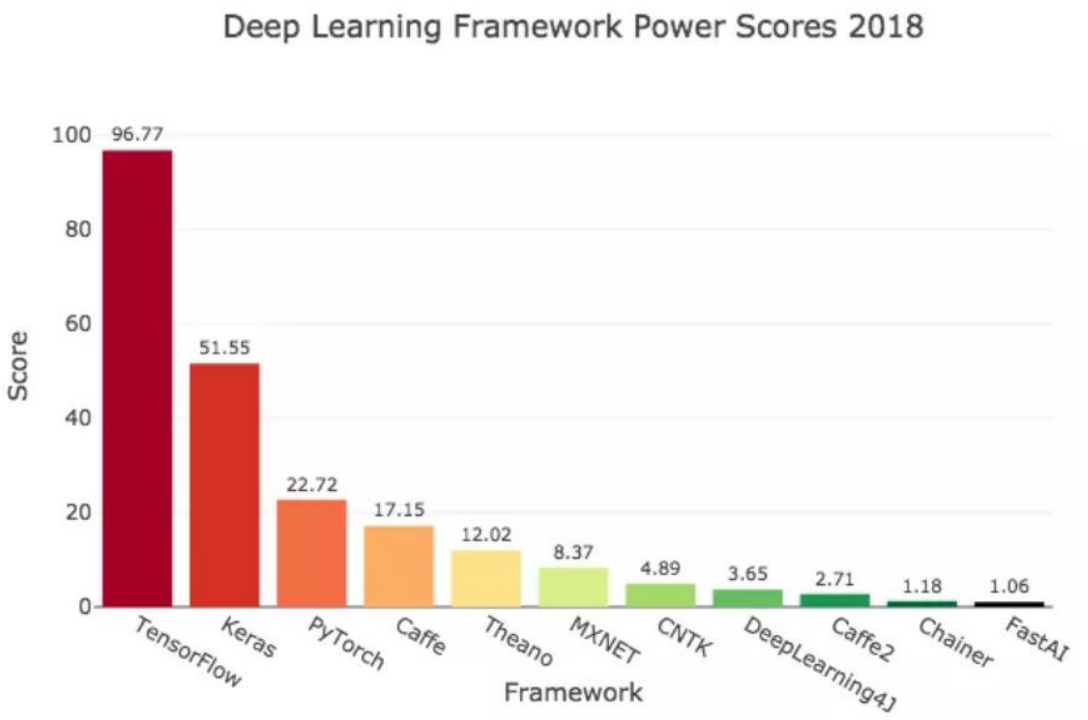
对于前一部分提到的，需要重点精度的论文，我们总会有一些细枝末节不明白，因为我们自身水平有限，而且论文受到篇幅限制，作者所以不可能面面俱到。所以这就需要我们从两个方向入手解决，一是在相关论坛搜集高手们对该论文的解读，另一个是阅读论文开源代码（这个我们在后面再细说）。要做到善用谷歌、百度搜索工具，你会发现，你遇到的问题，别人可能早就遇到并且得到解决，这样就事半功倍啦。

总结梳理算法细节

每当一个遇到的问题解决之后，应该做一些总结。这些总结**一来是巩固理解，加深印象，二来是找出论文各部件之间的差异**，分析这个差异是“改进”造成的差异，还是由于对主体算法“因地制宜”而造成的差异。用人话举个例子吧，对于 YOLO 和 YOLOv2 之间，bbox 的回归计算方法是不一样的，我们要体会一下作者做这种改进的初衷，不断去靠近这些大神们的思路，对于我们今后做自身的研究大有帮助。另外，Faster-RCNN 和 YOLOv2 虽然均是有事先 Anchor Boxes 来对 bbox 做回归，但是回归的计算方式也不一样，这就是由于这两个 Anchor Boxes 的由来不同导致的，Faster-RCNN 是由三种宽高比、三种尺寸形成的 Anchor Boxes，而 YOLOv2 则是由先验框聚类得到的。将这些细枝末节以图表，文字的形式总结起来（可以写在 CSDN 或 GitHub 上，也可以写在纸质版笔记本上），能够有助于我们找到一些算法的改进空间或者一些新的灵感。

2.选择合适的框架

对于重要的论文文章，光看是远远不够的，我们还得学习其工程实现。一来是编程复现论文算法有助于加深理解，二来是可以将我们所学知识转化为生产力。下图是某机构对各大深度学习框架的综合评估。



这里重点推荐 TensorFlow 和 PyTorch，因为它们不仅灵活性高，而且背后有强大的团队维护，可查阅的文档也很丰富。

当选定一个深度学习框架之后，**应当通过具体工程实例去学习，而非直接去阅读源码**。遇到一些常用的或者复杂的函数用法，要总结到个人博客中。更重要的一点是，要时常混迹于 GitHub 之中，将 star/fork 数量较多的，且与自己研究方向相近的代码 clone 下来去练习，学习大佬们的编程风格。（讲真，好的编程习惯写出来的代码简直是一个非常愉悦的艺术享受，这里切记自己闭门造车，一定要多读大神们的代码）

这里附上 TensorFlow 中文学习文档（这是我当年学的，可能距现在年代有些久远了，http://wiki.jikexueyuan.com/project/tensorflow-zh/tutorials/mnist_pros.html）。当对 TensorFlow 有一定了解之后，可以看看《TensorFlow 实战 Google 深度学习框架》，然后再看看《TensorFlow 实战》（注意别弄错了顺序，前面那本书稍微简单点，后面那本书主要是 demo）。待到 TensorFlow 能够灵活运用的时候，我们就得学习调参技巧了，这里推荐魏秀参（个人主页：http://lamda.nju.edu.cn/weixs/?AspxAutoDetectCookieSupport=1）的大作《解析深度学习网络——深度学习实战手册》（当初看这本书的时候，感觉自己的很多想法都与大神在这本书中提到的结论一致，感觉能与大神想到一起，真是十分开心）。

3.实战演练，paper 复现

当自己的编程能力、使用深度学习框架的能力达到一定水平后，我们要进入第三阶段，那就是复现之前看过的经典论文。刚进入第三阶段时，全靠自己来复现论文对于我们来讲可能有些困难。因此，我们可以先学习论文作者或者热心同行们开源的代码。具体步骤是：
1.先总体浏览开源代码，比对代码与论文描述是否一致，代码是否对自己更进一步理解论文有所帮助。
2.调试代码。建议初学者不要囫囵吞枣，只是“看”代码，还要一步步调试，看各个函数的

用法，学习各种编程小技巧，验证每步的结果与自己在“大脑编译”的结果是否一致。
3.动手复现。复现之前，可以先根据开源的代码画一个程序框图，然后根据框图逐次实现。然后将自己编写的代码调通，对比开源代码看看有哪些差距。一般差距主要体现在：代码的简洁性和功能的完整性。
4.重复第 2、3 步，直到自己能一目十行阅读开源代码，能掌握每个地方的编程技巧，复现出来的代码与开源代码基本一致，且能与论文对应的上的时候，这篇文章可以算是吃透了。

4.专注自己的东西

当自己复现的论文达到一定数量之后，编程水平会得到一个飞跃的提升。这时，我们要将自己从一个“学习者”的角色转换到“模仿者、改进者”的角色。一方面，我们要整合自己已经吃透的论文（包括代码），另一方面，我们要与时俱进跟进前沿研究，跟着大佬们“分一杯羹”。

经典文章和新论文两手抓
到了这个阶段，我们就不仅仅只读经典的文章了，还要开始接触新领域，新文章。我们千辛万苦学这么多东西，最终就是要有产出。对于很多年前的研究课题，它们只能帮助我们夯实基础，提升基本功，并不能有助于我们发表属于自己的论文或者专利。因为这些经典主题，早就被前辈同行们嚼烂了，只剩下骨头渣了。

碎片化阅读
当我们对深度学习领域坚持学习到一段时间之后，总会收藏一些自己感兴趣的论坛、专栏，以及大牛们的个人主页，他们会分享一些最新的经验或者研究成果。对于这些经验成果，我们可以选取自己感兴趣的或者与所研究领域相关的部分重点阅读（而不是全盘通读，这样太浪费时间）。比如前一阵子吴恩达分享的《Machine Learning Yearning》里面有些实战经验就很值得学习。

学会辨别新出来的好论文
对于年代较为久远的论文，我们可以通过文章的索引量或者开源代码 star 数量的多少来判断其是否值得读。对于新出炉的论文，我们可以去相关论坛看大神们的推荐（如关注一些知乎上的大 V 们，他们会经常在专栏里推荐一些新出炉的好文章），还有一些高质量的公众号，论坛等（这里重点推荐 paperweekly，链接 http://www.paperweekly.site/home）。将这些新文章里的 idea 与自己储备的基本知识相互碰撞，寻找一切可能产生的关联，或者找到新文章 idea 的不足，自己能加以改进。

发表属于自己的文章
当经典文献和新文章积累到一定程度时，我们一定会有一些自己的想法。这时候就需要我们扎实的代码功底，将我们自己的想法实现出来。对于新手，我觉得有这样几种情况可以视为自己的成果发表成论文：新问题用老方法，老问题用新方法，改进他人的新方法。然后将自己的 idea 及实验结果向导师汇报，让导师帮忙将自己的 idea“润色加工”，就可以愉快的发 paper 啦。

小结
本贴是对我个人学习深度学习历程的一个总结，由于本人水平有限，理论知识不强，因此与

各路大神的经验贴相比，更偏实际应用些。如果自己时间充裕，且导师愿意放人的话，可以去一些科研氛围浓厚的企业实习，如商汤科技、Face++、地平线、阿里达摩院、腾讯 AI Labs 等。笔者曾有幸在商汤研究院实习过一段时间，当真是人才济济，被组内的小伙伴渊博的知识，敏捷的头脑所折服，对提升自己的见识水平大有裨益。（但如果找的企业太偏向业务，我个人认为还不如呆在实验室多做做科研，多发几篇论文来的实在）。在我看来，对于普通人来讲，我们很难在理论上有很高的造诣或者突破。我们在大神们创造的理论框架下，千锤百炼、精益求精，就一定能有所收获。

我整理过的一些重要书籍，LeetCode 刷题代码，以及面试经验等，详见
https://github.com/computervisionlearner/Start_DeepLearning