

AI前线

2019年08月刊

AI - FRONT



关注落地技术，探寻AI应用场景



AiCon

全球人工智能与机器学习技术大会

AI 商业化下的技术演进

Geekbang> InfoQ

演讲专题

机器学习应用和实践

搜索推荐与算法

NLP应用和实践

计算机视觉

AI工具与框架

智能语音交互

智能金融

大数据框架应用

知识图谱

AI+教育

AI前沿产品

AI芯片

会议：11月21-22日

地址：北京国际会议中心

7折报名立减1440元

咨询热线：18514549229



目录

卷首语

工业级深度学习已经进入2.0阶段	I
------------------------	---

生态评论

AI走向自我毁灭？DeepMind一年亏损40亿到底意味着什么	1
自动驾驶趋冷，RoboTaxi渐热	6

重磅访谈

独家专访 AI 大神贾扬清：我为什么选择加入阿里巴巴？	9
-----------------------------------	---

企业机器学习平台

华泰证券基于 Kubernetes 构建 AI 基础平台的落地实践	21
---	----

推荐阅读

Tensorflow 2.0到底好在哪里？	30
-----------------------------	----

精选论文导读

基于深度学习的推荐系统效果遭质疑，它真的有带来实质性进展吗？	40
--------------------------------------	----

卷首语

工业级深度学习已经进入2.0阶段

作者：阿里资深算法专家 朱小强

机器学习和深度学习在互联网行业核心业务应用实践可以分为两个阶段。

第一个阶段是从 2000 年到 2015 年，可以将它粗浅地定义为基于大规模机器学习的上一代技术体系。这个技术体系在最初的十年里，机器学习相关的技术研发还是凤毛麟角，只有一些简单的应用。后来的 5 年成为了机器学习技术在工业界发展的黄金时间，包括百度、谷歌等国内外巨头都是在这个时间段内开始大力推动并引领了机器学习技术的规模化落地。最初工业界试图应用机器学习技术来解决问题的时候，发现学术界推出的各种复杂的算法模型对于工业界并不适用，因为工业界的数据规模实在太大了，复杂的模型根本训练不起来。为了将这些算法模型落地，工业界付出了很大的代价，其中一个关键要素是大规模分布式机器学习架构，例如基于参数服务器（Parameter Server）的并行训练系统就是这个阶段的典型代表作。此外，这期间也伴随着大数据架构如 Hadoop、Spark 等的兴起，为机器学习技术在工业界的实际应用起到了重要推动力。

第二个阶段始于 2015 年年底 2016 年初，以广告、推荐和搜索为代表的互联网公司开始发现，新一轮爆发于学术界和传统 AI 领域（如语音、图像等）的深度学习浪潮，也给互联网技术带来了全新的机会。过去的大规模机器学习，模型本身相对来讲比较固化和简单，还是偏重以人工先验设计加工的数据模式为主，但这一代的深度学习技术带来了更彻底的变革。

首先深度学习本身的模型容量更大、变化更丰富，可以针对具体的场景数据进行模型的自由定制；其次，模型的设计变得更加简单，基于标准化的深度学习训练框架可以非常容易地实现一个全新的算法模型并进行大规模的分布式训练，普通的算法工程师就能轻松完成这个过程。而在以前的大规模机器学习阶段，需要非常强大的专业团队花费大量时间才能定制式地研发出一个工业级可用的新模型算法；此

外，当模型训练出来之后，现在已经有一个非常体系化的流水线来支持任意复杂算法模型快速部署到线上提供服务。

这是这一代深度学习技术带来的巨大变革：算法结合算力的突破，从离线到在线、从数据到业务系统到背后整个算法体系，整个都串起来形成了一个体系化的 Pipeline。而且相比大规模机器学习时代，这个 Pipeline 的复杂性更高，但迭代进化的速度更快。

深度学习的技术爆发为互联网广告等行业带来了数十亿甚至百亿规模的营收提升，这是非常巨大的业务突破。但是，深度学习带来的提升并不是毫无代价的。深度学习的爆发就像超新星一样，突然一下子很亮，但非常快就会变成黑洞，变成黑洞之后，它会吞噬什么呢？

深度学习短短 2-3 年时间，就迅速地把工业界上一代十几年建立起来的整个体系存量，包括数据、系统、架构、算力等全部吃掉了。这就导致我们不得不面临这样一个问题：我们曾经引以为傲地认为是算力推动了这一轮 AI 的突破性发展，但今天，至少在工业级应用场景下，算力已经成为了阻力。其背后更深层次的原因，是由 GPU 带来的单点算力红利已经基本消失，相应地嵌入 GPU 算力的工业系统架构依然原地踏步、发展不适配。

如何重新设计过去这个持续了十几年的系统架构，是头部企业已经开始重点关注的问题。以广告场景举个例子，现有系统架构普遍遵循“匹配 - 召回 - 海选 - 粗排 - 精排 - 重排 - 策略机制”的链路体系，这个链路每一步都在考虑如何在算力约束下，尽可能地通过算法的手段去逼近最优效果、减少折损。然而，深度学习伴随的 GPU 算力升级，某种程度上已经可以打破这种体系。接下来工业级深度学习将进入 2.0 阶段，而这一阶段面临的问题本质上不是深度学习变了，也不是算力变了，算力不可能天天呈爆炸式发展，而会慢慢形成一个台阶并至少暂时停留一段时间。今天这个阶段面临的核心问题是，当前深度学习仍然跑在为上一代大规模机器学习模型需求而构建的那套系统架构之上。但过去的那套架构已经不太适合如今数据、算法和算力背后的需求。如今，算力和算法模型逐渐成为企业新的基础设施，如何面向业务场景的需求和数据的特性，对算力、算法和系统架构之间做协同设计（co-design），才是今天需要去思考的更彻底的一次变化。

工业级深度学习发展到今天，仅单独依赖工业界或者学术界、甚至仅依赖某几个头部公司或实验室来推动已经远远不够。工业界擅长将成熟的技术落地，学术界擅长对问题的抽象、定义和基础性研究，但今天大量的问题和数据在工业界，工业界不同规模公司的奔跑速度和进度也差异很大。下一代工业级深度学习需要工业界与学术界共同推进。

AI走向自我毁灭？DeepMind一年亏损40亿到底意味着什么

作者 | Gary Marcus

译者 | 核子可乐

策划 | 陈思

Alphabet 公司旗下的人工智能研究 DeepMind 正把大笔资金投入围棋及其它经典游戏的 AI 解决方案身上，可是 DeepMind 在盈利方面却是连年亏损的状态。在本文作者看来：这种**只有投入、没有产出**的状态，可能会危及 AI 领域的资金健康。

Alphabet 公司旗下的 DeepMind 单在过去一年中就烧掉了**5.72 亿美元（约 40 亿人民币）**。这背后到底隐藏着怎样的真相？

作为世界上规模最大的人工智能研究机构之一，DeepMind 在过去三年当中持续亏损，总资金投入已经超过 10 亿美元。在未来 12 个月内，DeepMind 还打算额外投入超过 10 亿美元预算。

这是否意味着 AI 正在走向自我毁灭？

并不尽然。**研究就是个花钱的事儿**，而 DeepMind 每年都在推动大量研究性工作。虽然涉及的金额很大，至少要比以往任何 AI 研究工作都要费钱，但与那些科学领域顶级规模的项目相比，这样的数目还远不足以说明问题。大型强子对接机每年的维护费用就高达 10 亿美元，而发现希格斯玻色子的总成本预计将超过 100 亿美元。当然，真正的机器智能（亦被称为人工通用智能，AGI）这种既能驾驶《星际迷航》中的战舰、又能分析自然语言中内容的伟大成果，其实现成本也绝对便宜不了。

但是，DeepMind 连年上涨的亏损额度仍是个值得关注的问题：**2016 年其亏损额为 1.54 亿美元，2017 年为 3.41 亿美元，2018 年则为 5.72 亿美元**。在我看来，其中包含三个核心问题：

- DeepMind 在科学层面是否找到了正确方向？

- 从 Alphabet 的角度来看，这种规模的投资是否合理？
- 如此巨大的亏损，又会给整个 AI 市场造成怎样的影响？

先来看第一个问题，我们确实有理由提出质疑。DeepMind 把大部分鸡蛋都放进了同一个篮子里，也就是深度强化学习技术。这项技术主要用于在模型的识别当中把深度学习与强化学习结合起来，基于奖励信号实现学习能力，例如让系统学会在游戏中获得更高分数、或者在象棋等游戏中努力取胜。

DeepMind 在 2013 年凭借着一篇振奋人心的论文闯出了名号，这篇论文阐述了如何训练单一神经网络系统以游玩多种雅达利游戏——包括《突围》以及《太空入侵者》——而且获得等同甚至超越人类的成绩。这篇论文可以说是工程界的一场巡回演出，亦成为 DeepMind 在 2014 年 1 月接受谷歌收购的关键催化剂。这项技术的进一步发展，推动 DeepMind 在围棋以及《星际争霸》这款知名游戏中接连取得辉煌的胜利。

但问题在于，这种技术对于环境的限制非常具体甚至是苛刻。例如，在玩《突围》时，一点点微小的变化——例如将镜头焦点移动几个像素，其性能就会急剧下降。DeepMind 的《星际争霸》方案同样限制重重：在单一地图上使用同一“种族”进行对战时，其成绩确实优于人类；但在其它地图使用不同“种族”时，成绩就要差得多。要想切换操作风格，我们必须得从头开始重新训练这套系统。

从某种程度上讲，深度强化学习是一种经过涡轮增压的记忆系统；其能够提供很棒的实际表现，但对于工作内容的理解却非常肤浅。因此，这类系统缺乏充分的灵活性，因此无法根据周遭环境的变化——哪怕是极其微小的变化——做出自我调整。（DeepMind 最近公布的肾脏疾病检查方案，也因为类似的原因而受到质疑。）

深度强化学习还需要大量数据的支持——例如以自我对局的方式完成数百万盘围棋对弈。这远远超过人类精通围棋所需要的训练量，而且整个过程往往困难且成本高昂。只有谷歌这样的科技巨头才能为其提供充足的计算机资源，这也意味着在面对诸多现实世界中的问题时，用户往往无法依靠单一计算机获得类似的效果。据估算，AlphaGo 的整个训练成本高达 3500 万美元，其消耗的能量足以支持 12760 个人类大脑在三天内不眠不休地工作。

但这里讨论的只是经济意义。真正的问题是，正如 Ernest Davis 和我在即将出版的《重启 AI (Rebooting AI)》一书中的讨论，我们能否信任现在的这些 AI。目前，深度强化学习只在良好的受控环境下具备可信性；这一点对围棋来说不是什么问题，毕竟无论是 2000 年前还是现在，围棋棋盘都没发生过什么变化。但是，我们显然不敢直接把它投入到现实问题的处理当中。

鲜有商业成功

部分原因在于，目前很少有哪些现实问题能够像 DeepMind 所关注的游戏那样拥有严格的限制条件，因此时至今日 DeepMind 仍然拿不出任何成规模的深度强化学习商业应用。相比之下，Alphabet 已经在 DeepMind 身上砸下约 20 亿美元（包括 2014 年收购 DeepMind 的 6.5 亿美元）。但在宣传之外的直接财务回收方面，去年 DeepMind 仅带来了约 1.25 亿美元收入，其中一部分还来自在 Alphabet 内部应用深度强化学习以降低冷却成本所带来的电费节约。

适用于围棋的功能，也许并不适用于 DeepMind 原本打算解决的其它挑战性问题——例如癌症与清洁能源。无独有偶，IBM 公司当初凭借着 Watson 计划在电视问答节目当中大放异彩，但该项目在医学诊断方面则表现不佳。虽然能够在某些病例当中取得良好效果，但 Watson 在其它一些病例中却折戟沉沙，有时甚至会在心脏病发作这类症状上犯错——即使是刚刚接触医学的新生，也不至于如此粗心。

当然，这可能只是个时间问题。至少自 2013 年以来，DeepMind 一直致力于研究深度强化学习技术，但科学进步确实很少能够在短时间内转化为实际产品。DeepMind 以及其它研究机构，也许终将找到一种可行的方法，推动深度强化学习产生更深刻也更稳定的结果，或者是将其与其它技术结合起来——但这只是一种可能性。深度强化学习最终可能被证明像是当初的晶体管一样，成为一项彻底改变整个世界的实验室内发明；或者也有可能永远代表一种学术上的好奇心，唯一的价值就是强调人类“为问题寻求解决方案”的意志。我个人的猜测，是这项技术可能介于这两者之间——成为一种有用且广泛存在的工具，但没有能力改变世界。

虽然 DeepMind 目前的战略成果可能达不到很多人的预期，我们也不该随意对其加以指摘。深度强化学习虽然不一定是通往真正人工智能的坦途，但 DeepMi

nd 本身仍是一家值得尊敬的组织，其运营严密、资金充足且拥有几百名高学位人才。围棋、雅达利以及《星际争霸》的成功，也帮助其吸引到更多后续人才。如果 AI 的风向标发生变化，DeepMind 无疑能够很好地适应不同的方向。就目前来看，还鲜有研究机构能够与 DeepMind 相匹敌。

与此同时，考虑到 Alphabet 的庞大资源背景，每年 5 亿美元似乎也不是太大的问题。Alphabet 公司非常睿智地决定在 AI 身上下注——包括 Google Brain 项目——其本身也在快速增长。Alphabet 可能会以多种方式调整自家 AI 成果的组合形式，但作为一家年收入高达 1000 亿美元的企业，其从搜索到广告推荐的所有业务几乎都依赖于人工智能，因此在这方面长期支持几个大项目并不是什么难事。

对于过度炒作的担忧

至于最后一个问题，我们恐怕很难判断 DeepMind 的经济学理念会给整个 AI 领域带来怎样的影响。如果炒作远高于交付，那么这有可能引发“人工智能寒冬”，甚至导致支持者不再投资。而如果 DeepMind 的亏损额度在接下来几年中仍然不断增长，就连 Alphabet 也有可能被迫选择退出。这并不只是资金的问题；到目前为止，DeepMind 也拿不出像样的财结果。这意味着到了特定时期，投资者也许将不得不重新调整自己对于 AI 技术的立场。

麻烦缠身的不只是 DeepMind。几年之前承诺的许多进展——例如无人驾驶汽车以及能够理解人类对话的聊天机器人，目前都没有实现。Mark Zuckerberg 曾在 2018 年 4 月向美国国会保证，AI 技术将很快解决虚假新闻问题。但是话好说、事难做，整个社会对于 AI 的观点，最终仍然取决于 AI 到底能够交出怎样的答卷。

就目前而言，真正的机器智能更多是种炒作而非实际方案。虽然 AI 在广告与语音识别等有限几个领域取得了重大进展，但可以肯定的是，AI 的发展还有很长的路要走。我们不能否认 AI 在对大数据集进行分析方面带来的助益，而且即使只有目前的水平，AI 也已经是一种非常强大的工具。虽然企业界也许对未来的 AI 不再倾力投入，但 AI 也绝对不会彻底销声匿迹。

我个人的预测？

十年之后，我们会得出结论，深度强化学习这几年来被高估了，并导致其它不少重要的研究途径遭到忽略。在强化学习领域投入的每一美元，都占用了其它研发经费的额度——例如本应花在人类谁知科学中的资金。机器学习研究人员现在经常会问，“机器如何利用大量数据来优化复杂问题？”但我们真正关心的是，“儿童为什么能够以远低于现有 AI 系统的数据与处理量，顺利掌握自然语言并了解现实世界？”如果我们能在后一个问题上多花点时间、金钱和精力，也许我们与人工通用智能的距离才真的不再遥远。

自动驾驶趋冷，RoboTaxi渐热

作者 | 陈思

2019 年 8 月 10 日，百度自动驾驶小巴“阿波龙”项目被曝出主力人员已撤出、产品已经停止推广的消息。据知情人士透露，由于项目的投入产出比跟预想有所出入，所以目前项目暂停推广与销售。

但是，也有业内人士认为：百度“阿波龙”项目更像是一次测试，其真正目的是在给未来具有极大潜力的 RoboTaxi 市场开疆拓土。

自动驾驶领域从 2018 年末开始趋于冷静，产业落地难、技术仍存在短板等问题让个人自动驾驶车辆的研发进展放缓，百度 Apollo、谷歌 Waymo、Uber 等知名大厂都在不断调整业务方向，以适应行业的最新变化。有意思的是，他们把目标出奇一致地放在了同一个方向上——RoboTaxi。

为什么要发展 RoboTaxi？

RoboTaxi，一眼看过去就像是把“robot”和“taxi”两个单词拼了起来，亦写作“Robo-Taxi”或者“Robo-Cab”。翻译成中文意思就是：自驾车或无人驾驶出租车，是一种自动驾驶汽车服务。随着自动驾驶技术的进步，一些车厂或是自动驾驶解决方案提供商开始转变思路：从让每一个用户都有一辆自动驾驶车辆，变成了提供自动驾驶车辆服务。

说起造成这种思路转变的原因，每家公司都有不同的说法，但是总结起来主要集中在以下几点：

1. 技术问题。就目前的技术成熟度来看，自动驾驶车辆的打造还是有些困难的，虽然有各种解决方案提供商表示可以“一键”将车辆升级为自动驾驶，但实际操作上还是存在着不少的问题；
2. 成本问题。自动驾驶方案提供商与车厂共同面临着成本的问题，以目前公众对待自动驾驶车辆的态度，除了少部分技术爱好者、好奇者，大部分用

户都处于官网的态度，因此打造一辆无人驾驶汽车能有多少用户真的愿意掏钱购买，仍然是个未知数；

3. 安全问题。自动驾驶车辆最令用户担心的就是安全问题，普通用户若购买后发生了事故，责任划分、事故评估标准、惩罚机制等等在全球范围内都缺少相关法律或者政策的支持。

综上，在一些厂商看来，推出自动驾驶服务是最佳的解决方案，一一对应下来，推出 RoboTaxi 的优势也有这样三点：

1. 虽然技术上仍然存在短板，但是相比“一键”升级，通过解决方案提供商与车厂的合作，直接造一辆自动驾驶车辆的技术复杂度相对而言更小，成熟度也更高；
2. 既然大部分用户不愿意单独购买无人车，那么与车厂合作推出乘车服务确实是最合适的解决方案，用户既可以体验到无人驾驶的便利，也省去了是否要自己购买一辆的担忧；
3. 个人用户存在对于操作不熟悉导致误操作的情况，一旦发生事故则会对追责等后续工作造成困难。如果直接交给服务提供商，可通过直接与服务提供商对接进行责任划分及处理。

国外发展现状

国外的 RoboTaxi 服务主要集中在北美。

2016 年 8 月，从麻省理工学院拆分出的 NuTonomy 成为首个推出 RoboTaxi 服务的公司，地点在新加坡的某个限定区域内。一个月以后，Uber 也推出了同类服务。

2017 年初，谷歌旗下自动驾驶公司 Waymo 于凤凰城开始大规模的 RoboTaxi 测试；同年 5 月，Waymo 宣布与 Lyft 共同推行 RoboTaxi 服务。

2017 至 2018 年，宝马、大众、通用、福特等车厂联合英特尔、英伟达、谷歌、Lyft 等等公司纷纷入局 RoboTaxi，而这也是自动驾驶最火热的一个时段。

2019 年 4 月，特斯拉创始人埃隆·马斯克更是公开表示：要在**2020 年建立特斯拉 RoboTaxi 网络**，作为特斯拉网络“**RoboTaxi**”服务的一部分，运营车辆的车主每年可以获得高达 30,000 美元的收入。马斯克已将目光投向**自主移动即服务（MaaS）**市场，在特斯拉宣布加息后的一次电话会议中，马斯克指出：**RoboTaxi 最终可能将公司推向市值 5000 亿美元**。

值得一提的是：就像“2020 年实现全面自动驾驶”的承诺一样，大部分公司把 2025 年设置为全面实现 RoboTaxi 的目标年限。

在亚洲范围内，除了新加坡，日本也有一些公司正在进行 RoboTaxi 测试，由 DeNA 和 ZMP 的合资企业计划为 2020 年奥运会提供自动驾驶出租车服务。

中国新玩家

热闹的 RoboTaxi 市场，怎能缺少中国玩家的身影？

2019 年 6 月，百度在拿下了在长沙运营 RoboTaxi 服务的许可，这也就意味着：百度极有可能成为国内第一批运营的公司之一。

2019 年 8 月 5 日，滴滴出行宣布旗下自动驾驶部门升级为独立公司，专注于自动驾驶研发、产品应用及相关业务拓展。据了解，RoboTaxi 也是滴滴自动驾驶公司主要发展的业务之一，作为国内出行服务提供商，滴滴在该领域似乎有着先天的优势。

2019 年 8 月 7 日，自动驾驶初创公司文远知行宣布与华南最大的出租车公司、广州公交集团旗下的广州市白云出租汽车集团有限公司，以及科学城（广州）投资集团有限公司组建合资公司——文远粤行 WeRide，专注 RoboTaxi，共同布局自动驾驶新出行。

在不久的将来，RoboTaxi 领域或许会出现一场“大混战”，究竟是具有先天市场优势的出行公司能抢得先机，还是拥有解决方案的科技公司能够后来居上？一切都是未知，但同样令人期待，相信科技与服务的完美结合，才能给用户带来极致的出行体验。

独家专访 AI 大神贾扬清：我为什么选择加入阿里巴巴？

作者 | 蔡芳芳

刚满 35 周岁的贾扬清是出生于浙江绍兴上虞的青年科学家，是业内主流 AI 框架 Caffe 的创始人、TensorFlow 的作者之一、PyTorch 1.0 的共同创始人，是全球最受关注的 AI 科学家之一。他曾任谷歌大脑研究科学家、Facebook AI 架构总监，于 2019 年 3 月正式加入阿里巴巴。在阿里巴巴，贾扬清的新 Title 是阿里巴巴集团副总裁、阿里云智能计算平台事业部总裁，不过内部更为流行的叫法是“阿里计算平台掌门人”。在 7 月 24 日于上海世博中心召开的阿里云峰会·上海 开发者大会现场，InfoQ 记者非常荣幸得到了对贾扬清进行一对一专访的机会，这也是贾扬清加入阿里之后首次接受国内媒体专访。

真正跟贾扬清近距离接触后笔者发现，这位被很多人称为“AI 架构大神”的 80 后青年科学家，更像一位温柔且平易近人的邻家“学霸”，虽然技能全面碾压但丝毫没有架子。加入阿里以来，贾扬清一直忙于了解集团覆盖范围极广的各项产品和业务，近两个月才开始在一些重要活动上以新身份亮相。他在访谈中直言，阿里非常大、方向非常多，短短几个月还未能全部了解完。虽然离职时多家媒体均报道贾扬清的 base 地是阿里硅谷研究院，但由于团队基本都在国内，贾扬清加入阿里后在杭州待的时间更多。新的身份给他带来了许多挑战，忙到没时间理发、一个月倒三次时差、参加活动大半天没吃上饭。但身为阿里集团副总裁的同时，贾扬清依然是典型的技术人，在聊到他最熟悉的 AI 技术和平台时会因为兴奋而语速加快。

在这次访谈中，贾扬清向我们透露了他加入阿里的原因，并对他目前在阿里主要负责的工作做了详细说明，他不仅回顾了过去 6 年 AI 框架领域发生的变化，也分享了自己对于 AI 领域现状的观察和对未来发展的思考。结合自己的经验，贾扬清也给出了一些针对 AI 方向选择和个人职业发展的建议，对于 AI 从业者来说有不少可借鉴之处。

从 Facebook 到阿里巴巴

被内部称为“阿里计算平台掌门人”的贾扬清目前直接领导阿里云智能计算平台事业部，而计算平台事业部同时负责大数据和人工智能两大平台，其中大数据方面包括 Flink、Spark 以及从阿里自己做起来的 MaxCompute 大数据平台，人工智能平台则包括底层资源管理、中间层 AI 框架开发等一系列工作。

相比原来在 Facebook 所负责的 AI 框架和 AI 平台相关工作，贾扬清现在在阿里所负责的工作范围更加宽泛。用贾扬清自己的话说，原来在 Facebook 他更多是大数据的用户，只是在 AI 训练的时候需要大数据平台提供支持；而在阿里计算平台事业部，他需要同时支持人工智能和大数据这两块。在贾扬清看来，阿里云智能计算平台事业部是全球少数的几个把大数据和人工智能放在一起的部门，但他认为大数据和人工智能本身就是紧密结合的，因此这两块放在一个事业部做非常有道理。

过去几年深度学习得以快速发展，很大程度上要归功于数据，而今天的人工智能在一定程度上其实可以说是“数据智能”，即 AI 需要大量的数据才能够提炼出来我们所谓的模型。于是，大规模人工智能可以归纳为两种计算模式，第一个是“智能计算”，就像 AI 工程师熟知的训练、模型迭代等，另外一个“数据计算”，就是怎么样把大量的数据灌到人工智能训练和推理的链路里面。“数据计算”是大数据一直以来擅长的事情，把这两个计算结合到一块的时候，大量的数据处理跟高性能的数据链路，再加上现在人工智能的算法、高性能计算等一系列技术，才能把整个的解决方案给做出来。从这个角度来说，贾扬清认为人工智能跟大数据是天生结合在一起的。

除了技术范围不同，从 Facebook 到阿里巴巴也给贾扬清带来了另一个新挑战，那就是岗位角色的变化。早期还是研究员的时候，贾扬清只需要关注技术，后来升任 Facebook AI 架构总监后他转型技术管理，到现在成为计算平台事业部总裁，贾扬清需要管理技术、产品和业务，后两者对于他来说是更大的挑战，但也是很有意思的挑战。贾扬清告诉 InfoQ 记者：“就像开源要商业化落地一样，技术最后也需要经过产品和商业的锤炼，这是一定程度上我来阿里的原因。另外，云原生接下

来是整个 IT 产业大势所趋的方向，在云上，技术肯定会有新一轮的进化，这个时候对于我来说是接受新挑战的一个非常好的时机。”

与阿里一起将开源进行到底

除了上文提到的 Title，贾扬清如今也同时支持阿里巴巴的一系列开源工作，因此他在这次开发者大会上所做的演讲主要也是围绕开源这个话题来展开。对于开发者社区工作，贾扬清有非常大的热情，他希望自己加入阿里之后能够更好地推动国内开源的发展。

在采访中，贾扬清向我们详细介绍了阿里巴巴接下来在开源层面的几个重点策略。阿里巴巴希望在以下三个方面做比较完整的梳理并和社区合作推进：

1. 底层操作系统，阿里有飞天操作系统底座，同时也应用过很多像 Linux 这样本身就开源的系统，前段时间阿里就刚刚发布了自己的 Alibaba Cloud Linux 2 OS，接下来阿里会考虑如何将自己在这方面的能力以最优的方式贡献给社区。另外一块是再往上层的云原生，这一层可以广义地叫做操作系统，在一定程度上跟应用比较相关，比如 AI 平台、大数据平台，在上云的情况下可以认为他们也是大规模云原生的操作系统底座。接下来阿里在 K8s 等云原生系统上也会深入跟一些开源组织合作，加强 Flink、Spark 等开源大数据产品的输出。
2. 前端，相比底层系统的开发模式，前端更偏向设计和交互，蚂蚁金服的 AntDesign 是其中做得非常好的一个代表项目，在前端领域非常受欢迎。
3. 工具层也是阿里巴巴一直以来非常感兴趣的一个方向，比如怎么用开源的项目和解决方案，来帮助开源的开发做得更有效率，这其中又包括测试部署工具，源代码管理工具，项目交流平台，等等。这一块国外社区相对来说做得更好，因此未来如何在国内推动建立更好的工程师交流平台和文化，也是阿里巴巴开源层面关注的重点。

在贾扬清看来，应用开源项目的时候，跟社区的紧密程度往往决定了整个应用的成败。过去业界常常出现一种情况，每个公司都对开源社区版本有所谓的改动，最后社区的能力和公司的能力分别发展，很难融合回去。但今天大家已经从过去的

经验教训中学到了与社区对接更好的方法。以计算平台事业部当前投入大量精力运营的实时计算引擎 **Apache Flink** 项目为例，一方面，公司会深入参与到一些功能的开发和优化当中，另一方面，大家会把软件上的能力和系统架构上的思路都贡献到开源社区，避免每个人把开源拿到公司内部自己又改一通这种做法，进而推动开源项目更好地发展。对于公司来说的话，反过来也能够更加有效地利用开源最新的成果。

除了集团层面的开源策略，对 **Apache Flink** 开源社区，贾扬清也透露了接下来的一些规划。

目前 **Flink** 对于流计算的使用场景已经支持得非常好了，接下来阿里依然会从用户的需求出发，继续对 **Flink** 做优化和改进。同时，阿里会继续开展 **Flink Forward** 大会这样的活动，通过这些活动和开发者互动，向大家传达 **Flink** 接下来的 **Roadmap** 等；另外也希望通过这种方式和社区对接，来获得更多的关于 **Flink** 接下去应该怎么走的系统设计，包括 **Roadmap** 上的一些输入。贾扬清表示，对一个比较健康的开源项目来说，自己的系统设计跟用户需求的输入这两方面都是必不可少的。所以很多开源项目都会有一个对应的开发者大会作为跟开发者互动的媒介，**Flink** 也不会例外。

AI 技术的现状和未来

Caffe 是“之前的代表作”

作为 AI 架构大神，很多人知道贾扬清是因为他写的 AI 框架 **Caffe**，但那也已经是 6 年前的事情了。在聊到 **Caffe** 这个代表作时，贾扬清特意停下来强调，“是之前的代表作”。因为 6 年前还没有框架可以满足需求，所以贾扬清为了解决论文研究中遇到的问题自己开发了 **Caffe**。贾扬清认为，那个时候还是相对比较刀耕火种的时代，大家更多只是出一些科研论文，对于工程上软件框架怎么做、怎么适配都比较早期。但随着深度学习的兴起和发展，工业界的输入越来越多，如今像 **TensorFlow**、**PyTorch** 等主流 AI 框架在解决图像、自然语言处理、语音等一系列建模问题时，已经比当年做得好很多，现阶段主流框架要解决的问题比 6 年前贾扬清自己研发 **Caffe** 框架时想要解决的问题也更大了。

贾扬清将 Caffe、Theano、Torch 定义为上一代 AI 框架，这些早期框架带有很强的学术界的烙印，做的更多的是针对科研方向的一些尝试。而如今的第二代 AI 框架如 TensorFlow、PyTorch 已经把框架的概念扩得更宽，不光只是做深度学习的神经网络建模，更多的是怎么设计一个通用的科学计算引擎，同时探索编译器、软硬件协同设计和更复杂的建模的上层框架等方向。“6 年前所说的框架在今天看来，可能只是整个软件栈里面很窄的一部分。”

Caffe 推出之初，贾扬清希望它能成为“机器学习和深度学习领域的 Hadoop”，现在回过头来再看当时定下的目标，贾扬清觉得其中有一些巧合特别有意思。当初定下这个目标，是希望 Caffe 能够像 Hadoop 一样广泛普及，没想到后来二者真的在发展路线上经历了类似的情况。从大数据的角度来说，今天大数据已经从 Hadoop 进化到 Spark、Flink 这样更为复杂的引擎。当年 Hadoop 主要的计算模式就是 MapReduce，而如今的 Spark、Flink 都使用了更复杂的做法。比如 Spark 使用 Directed Acyclic Graph (DAG) 对 RDD 的关系进行建模，描述了 RDD 的依赖关系，它的计算模型比 MapReduce 更加灵活。Caffe 跟现在的新一代框架相比也有类似的情况，Caffe 等之前的框架的设计非常针对神经网络，它的系统设计中有个叫 Network 的概念，然后里头还有一个 Layer 的概念，做前向和后向计算，比较固化，有点像 MapReduce 一样。而今天的主流框架如 TensorFlow、PyTorch、Caffe2 等，使用的是更加通用的计算图模型。可以说 AI 框架和大数据框架经历了类似的发展历程，都有一个从第一代向第二代往前进化的情况。

AI 框架应该关注重复造轮子之外的挑战

在早前阿里一次内部演讲中，贾扬清表达了这样一个观点，他认为“AI 框架的同质化说明技术的挑战在其他更广泛的方向”。对于做出如此判断的理由，贾扬清向 InfoQ 记者做了更详细的说明。

贾扬清将当前 AI 框架的用途归结为最重要的两点，一是支持在框架之上简单地建模，也可以叫做开发的灵活性；另一点是实现更高效的计算，因为一旦把 AI 算法应用到工程上，基础架构的效率会变得非常重要，比如推荐系统可能要跑在几万台甚至几十万台机器上，性能优化就必须做好。当前大多数框架都在朝这两个方向努力，包括 TensorFlow 2.0 加入了 Eager Mode 和 PyTorch 1.0 将旧版本的

PyTorch 和 Caffe 合并，都是在逐渐解决前面提到的这两个问题，补齐自己的短板。其实目前已有的框架都在互相学习和借鉴，大家要解决的问题已经开始逐渐变得清晰和明朗化，大家的设计也在朝着同一个方向走。“这个时候，从一定程度上来说，重新造一个轮子到底有多大的意义呢？这是 AI 工程师需要深入考虑的问题。”

更深地说，几年前，大家说到 AI 的时候有点将 AI 等同于 AI 框架这样的情况，但到了今天，整个 AI 工程的解决方案做出来，其实框架是里面很薄的一部分。贾扬清认为 AI 框架就好像计算机编程语言，比如 C++ 是一个比较好用的语言，但光有它是不够的，框架真正能体现价值的原因，在于它有非常好的生态，而且有很多的科学计算和外部服务等。所以从框架开始，往上跟往下都有非常多新的战场或者说更多的领域需要我们关注。

往下可能包括系统上的创新，比如怎么做高性能计算、怎么做软硬件协同设计等；往上做的话，

框架本身可能没有做太多大规模训练、模型迭代等工作的完整工具链。因此阿里现在关注的第一个是拥抱框架，第二是把 AI 整个链路做出来，比如前段时间阿里开源了一个 MNN 引擎，它可以让我们更好地在手机端、嵌入式端运行模型。另外，阿里还有一个开源项目叫 XDL，XDL 的一个想法是怎么构建大规模稀疏化的推荐系统，稀疏化建模是很多通用框架上非常缺的一层：在基本的框架上面，需要有一个高层的抽象，或者是更加跟业务相关的工具平台来解决这个问题。为什么大规模稀疏的系统有用呢？因为所有的推荐系统都是跟它有关系的，比如说阿里巴巴怎么样来做推荐，用户在不同的新闻网站上面感兴趣的新闻是什么，都涉及到稀疏数据，所以这一块光有通用计算框架解决不了，AI 端到端的工程需要在整个栈上发力。

软硬件协同设计

在贾扬清看来，AI 编译器是接下来比较有趣并且非常重要的一个研究方向。首先，深度学习框架之前往往需要手写各种算子的实现，如果有一个新硬件版本出来，这些函数往往需要重新优化；其次，做优化的时候，这些手写函数到底是不是最优的，即使是专家，是否能穷尽所有可能的办法找到性能最优解，都是不一定的。

而像 XLA、TVM 这样的 AI 编译器就是在解决这些问题。

AI 编译器属于软硬件协同设计的范畴，旨在最大化芯片能力。当前新的芯片产品层出不穷，我们原来这种来一个硬件在上面手写设计软件的模式已经开始逐渐的跟不上了。现在的应用越来越复杂，结构也越来越复杂，我们也不知道手写的设计是不是最优的，这个时候就开始要考虑怎么做 AI Guided Compilation 或 Performance Guided Compilation，把硬件的能力跟软件的灵活性更好地结合在一起。以 TVM 这个项目为例，它可以在运行时的时候，通过计算模式跟硬件特征来设计或者生成最优的运行代码。这些都是软硬件协同设计正在探索的方向，已经变成比框架更有意义的方向，无论从科研还是应用上来说都是如此。

但软硬件协同设计要做好难度很大，需要对体系结构有丰富经验的人才行，这样的工程师很少而且培养也非常困难。如果我们能够将体系结构建模成一个可优化的问题，那么机器学习就可以派上用场了。

AI 与计算机系统架构

Jeff Dean 等人从 2018 年 3 月开始发起 SysML 会议，聚焦于机器学习 / 深度学习相关的硬件基础设施和计算机系统。那么 AI 到底能够给计算机系统架构带来哪些新的机会？贾扬清认为可以从两个方面来看。

Jeff Dean 在提到 SysML 的时候，其实提过这样一个概念，就是 Machine Learning for Systems and Systems for Machine Learning。今天我们做的更多的是 System for Machine Learning，指的是当机器学习有这样一个需求的时候，我们怎么去构建一个系统来满足它的需求。另一方面，在计算机系统构建的过程中，我们还可以考虑怎么通过机器学习的方法跟数据驱动的方法来优化和设计系统，解决原来系统设计对人的经验的依赖问题，这是 Machine Learning for System 可以解决的事情，不过目前这方面还处于相对比较早期的探索阶段，也是人工智能接下来还需要突破的瓶颈之一。

AI 商业化落地

工业界落地是贾扬清现阶段重点关注的另一个方向。与初入 AI 领域的时候相

比，贾扬清认为现在 AI 领域最大的一个变化就是工业界对于 AI 的应用需求和算法输入变得越来越多了。最早在 2000 年初的时候，业内大家会有一个感觉：机器学习在 80% 的时候能够以 80% 的准确率解决 80% 的问题。这其实意味着它没有跨过可以实际商用的门槛。但到了现在，深度学习在不同领域取得了相当亮眼的成果，很多算法的准确度提升到了可以落地应用的阈值之上，如此一来工业界开始大量地使用算法，反过来又推动算法进一步发展，越来越多工业界的场景需求被反馈到科研上，同时也使 AI 研发人群数量有了一个非常大的增长。

但与此同时，人工智能在实际落地的时候依然存在行业壁垒。如何把人工智能的通用性做好，使 AI+ 行业能够真正地推动下去，这在贾扬清看来是人工智能领域目前面临的另外一个瓶颈。

从科研的角度来说，人工智能领域也存在不少待解决的问题，比如智能的本质到底是什么，我们现在更多在做的还是预测问题，那么怎么做因果关系和因果推理，怎么做可解释的人工智能，等等。

如何选择 AI 研究方向？

最近知乎上有一个问题非常火爆，题主的问题是“当前（2019 年）机器学习中有哪些研究方向特别的坑？有哪些小方向实用性很差或者很难做？或者有哪些小方向是只有圈子内的人能发？”我们把这个问题也抛给了贾扬清，请他来回答。不过扬清大神表示预测未来太不靠谱，当初他刚开始搞机器学习的时候大家都认为神经网络肯定没戏，“所以这事谁也说不清楚。但有一点是比较肯定的：**不要再写老框架了，而是要看看新的方向。**”

贾扬清直言，如果现在再去写个 Caffe 框架就没意思了，但如果你是想看到 TensorFlow 和 PyTorch 的长处和短处，并写出一个不管是编程语言还是系统角度跟他们有差异化优点的更好的框架，还是可以做。总之就是：**不要 Follow，而要想些新的问题。**比如谷歌最近有一个叫做 JAX 的项目就非常受欢迎，首先它能够非常自然地跟 Python 结合到一起，同时应用底层编译器的能力来做优化，这些是科研上很有意思的新方向。虽然贾扬清不认为 JAX 会马上替换掉 TensorFlow，也不认为它一定能解决所有问题，但它确实是一个很好的探索方向，就像 2008 年的

Theano、2013 年的 Caffe 一样，是值得一看的新东西。

另外，贾扬清也表示最近看到偏网络调参的论文有点多，他认为通过手工调参使性能得到一点提升这样的研究价值正在逐渐下降。研究人员更应该去关注是否有更好的方法论来实现自动网络调优。也就是说“越偏手工的科研方向是越低价值的方向，越能够提炼出通用的方法论并用到大规模系统上的研究可能更加有意思。”

也谈大数据计算平台

当前深度学习其实还是非常依赖大数据量，随着互联网和终端设备的快速发展，产生的数据不仅量大而且变化也非常快，那么如何快速将最新的数据输入进来、处理并生成更为精准的算法模型？这给 AI 基础设施，包括大数据计算平台，提出了新的要求。

早前像 Hadoop 这样的系统是将计算、存储都放在一起，后来业内开始更多提倡计算与存储分离，便于实现弹性的扩缩容。当时说计算跟存储分离，更多在说的是存储跟大数据计算，而今天我们又有了新的计算，就是 AI 训练带来的异构计算。贾扬清将对应的新系统称为存储、数据计算和科学计算三者分离：存储主要解决分布式存储大量数据、稳定性、Throughput 这一系列的问题；数据计算解决怎么做数据的预处理、数据的清洗、数据的变换这一类问题；科学计算解决怎么利用硬件的性能来快速解决大量数学表达式的运算和计算模式问题。这样一个系统，怎么进行模块化设计，怎么将不同模块有机地联合起来，这是对于大数据系统设计提出的新挑战。

对于大数据计算平台未来需要重点发力的方向，贾扬清认为主要是如何进一步对接场景、提高效率以及优化用户体验等。他表示，大数据计算现在主要有四个场景，第一个是传统的批计算，第二个是像 Flink 这样的流计算，第三个是怎么做秒级甚至毫秒级的交互式查询，第四个是怎么把大数据链路跟 AI 打通，来做大规模的智能模型的训练和部署。

流计算这几年越来越受重视，Flink 一直主打流批统一，对此，贾扬清也有自己的看法。

批和流在大数据领域面对的场景还是很不一样的，我的观点是，批和流这两个场景是根据底层的系统设计自然出现的，不同的引擎有些擅长批，有些擅长流，Flink 在流上面就是无出其右的，做批流统一是为了给用户提供更加完整的体验。

现在大家经常讨论批和流统一的问题，其实是因为有比较实际的背景。大家更多场景是以流计算为主，有时候需要有一些批计算的应用，但不需要太高的效率。这个时候如果完全整个换一套引擎，开销太大，这时候引擎就要考虑怎么去补齐短板，为应用提供一个端到端的体验，不需要在计算过程中因为不同需求把所有数据都挪一遍。在批流统一上，我认为 Flink 还是会继续加强自己在流计算上面领先的地位，同时对于批计算、交互式查询做补齐，使用户在一个以流为主、相对比较综合的场景下，就能够很快地构建自己的解决方案。

现在我们看见越来越多的流计算、交互式计算这两个应用场景出现。至于未来是以流计算为主还是以批计算为主，我个人觉得接下来很长一段时间这两类计算还是会同时存在，而且针对不同的场景优化其实还是比较独特的，很难用一套来解决所有的问题。

给 AI 从业者的建议

作为 AI 科学家中的佼佼者，贾扬清在短短几年里完成了从工程师、研究人员到技术管理者的蜕变，在职业发展生涯的不同阶段，大神也会面临不同的挑战。在采访的最后一小段时间，贾扬清跟我们分享了他从自己的职业发展生涯总结的几点小经验，希望对大家有所帮助。

持续学习，多和同行交流

AI 技术发展日新月异，对于开发者、工程师来说，最头疼的可能是每天都会不断涌现出新的技术、新的框架、新的算法模型等，一不留神可能自己掌握的知识就过时了，而且这也使研究方向的选择变得越来越困难（有多难、能否出新的成果、

有没有别人做过了等都要考虑)。

贾扬清强调，AI 领域的工程师一定要主动吸收学习新的信息和技术成果，持续进行知识迭代。像 HackerNews、Reddit 的 Machine Learning 板块等都是很好的信息渠道。现在也有很多媒体在推动 AI 领域前沿一些新想法、新成果和新的科研方向的传播，这也是一件很好的事情。

至于研究方向的选择，贾扬清认为两条原则就够了，一个是兴趣，另一个是多和同行交流。

让自己成为技术多面手

正如前面提到观点，在贾扬清看来，现在 AI 框架已经不再是 AI 工程师的桎梏了，AI 工程师的岗位职责也会相应的发生一些变化，未来大家需要把自己的关注点更多地放在跟实际应用场景相结合上。接下来最大的机遇是怎么把 AI 真正落地，对于工程师来说，他就需要从单纯只做 AI 转变成具备全栈能力，包括怎么对接 IoT 设备、怎么把 AI 能力做到汽车上去等。工程师除了要加强自己的 AI 能力以外，也要把自己的技术能力培养得更加多面手一些。

贾扬清坦言，在一定程度上，其实“AI 工程师”这个称谓有点被滥用了，更多还是因为 AI 比较热而催生出来的角色。

“其实今天没有只是做 Java 的 Java 工程师，因为 Java 只是一个工具，AI 其实也只是一个工具，更重要的还是你拿它来做什么。就像每个工程师都得会编程，或者说每个工程师都得会基本的一些工程能力，AI 也是一项基本的工程能力。”

对于普通开发者，不管是哪个领域的的开者，未来都可以学一学 AI 的应用，不一定需要学到知道怎么构建 AI 框架这个程度，只需要学会怎么把 AI 当成 Excel、Java 这样的工具来用，这是增强自己能力的一个比较有意思的方向。对于专门做 AI 的工程师或者科研人员来说，更多的还是需要自己的领域深挖。

从一线开发到管理者：学会后退一步，成就他人

贾扬清刚刚从一线开发转到管理岗位的第一年，依旧埋头写代码，产出的代码量在组里头不是第一就是第二，但这使他在对团队的支持和培养方面的投入捉襟见肘。对于一线开发这或许是个不错的表现，但对于一个管理人员来说就未必了。因为作为管理者，他并没能给团队的成长提供足够多的价值。人的精力是有限的，哪怕他当时工作时间从上午九点到晚上十二点，甚至更晚，也无法做到兼顾写代码和支持团队。

这一年发生的事情对贾扬清的触动很大，他后来才意识到作为管理者真正需要做的是帮助团队其他人最大化自己的能力，而不是说像单纯搞技术开发的时候那样自己一个人往前冲就可以了。管理者要学会自己往后退一步，提供必要的指导和赋能，相信别人并为他们创造空间，让一线的同学能够得到更多的锻炼，心态得从自己钻技术转变为支持团队 **Scale** 技术。

华泰证券基于 Kubernetes 构建 AI 基础平台的落地实践

作者 | 刘赐民

策划 | 蔡芳芳

华泰证券一直将信息化建设作为公司发展的主推力，目前公司正在推动 AI+、AI 中台等战略。公司内部的 AI 基础平台已经落地，作为量化平台、数据科学平台等业务平台的基座，承载了华泰证券目前以及未来深度学习等领域的工程孵化。

目前 AI 基础平台底层基于 Kubernetes 和大数据平台，支持 Job 驱动、Pipeline、DAG、分布式计算框架等功能；同时支持异构计算，包括 Kubernetes 集群、Spark 集群等。本文暂只描述 AI 基础平台上层设计部分，后期会另外撰文聊聊如何打通大数据平台获取行情数据等资讯为量化平台、数据科学平台提供数据支持。

AI 基础平台项目出发点

随着大数据和人工智能技术的发展，人工智能相关技术已在华泰证券内部多个业务场景里取得创新应用，包括精准营销、量化交易、智能投顾、智能诊股、营销反欺诈、相似 K 线等场景；类似应用案例都需要依托于海量金融、产业、行业相关数据，并通过数据挖掘、机器学习、深度学习等相关技术来实现。这类计算密集型业务，一般都需要使用 GPU 服务器，在没有 AI 基础平台时，各个算法团队需要自购或申请自己团队独占的服务器资源。但是这种各自为战的模式会因为资源管理、服务器运维、资源争抢等问题致使各个团队效率降低。在这种背景下，需要建设统一的集群资源管理来解决这些问题。

华泰证券的 AI 基础平台基于海量数据，支持多用户、超高频率训练 / 预测 / 回测、主流深度学习框架，提供机器学习、深度学习所需的统一运行环境。各个算法团队可以将精力从资源、数据等方面抽出来，集中精力在自己团队的核心业务上。



华泰证券信息技术部数据科学中心平台团队研发的 AI 基础平台定位于承载公司内部所有一站式人工智能平台所需的底层平台，基于底层高性能计算资源、海量数据和资源调度能力，并持续集成主流深度学习 / 机器学习框架，能够对外提供通用机器学习 / 深度学习所需基础平台环境。平台主要功能包括：深度融合大数据平台和 AI 基础平台；为上层数据科学开发平台、量化平台的数据处理、模型训练、模型预测、模型发布提供底层支撑；为 AI 算法工程师以及业务团队提供便捷高效的深度学习框架技术。平台在提供多样化服务的同时还需具备稳定性、自动分布式化、算法兼容性、弹性扩缩容、易用性、资源利用率提升这几大特点。AI 基础平台是满足各种业务场景下机器学习、深度学习需求的通用基础服务平台。

选型考虑

从大部分用户的角度出发，最迫切的需求是保证离线跑批任务的资源隔离性，离线任务运行时不出现资源争抢，还需要为用户提供方便快捷的调试环境，比如常用的 Jupyter Notebook、Jupyter Lab 等等。由于历史原因，大部分的算法工程师习惯于单机单卡或单机多卡，因此为了让算法工程师们无感知迁移到机器学习平台，除了要为算法工程师们提供回测支持，还需要提供个人调试环境，个人调试环境中

需要将数据持久化存储以供开发人员频繁读取写入。

根据以上重点进行架构选型的研究，底层使用容器技术来保障离线计算任务的资源隔离性，同时又可以保障个人调试环境的隔离性，还可以提高单服务器上的资源使用率。不仅如此，对于不同的开发人员，常用的机器学习框架不同，例如 Sklearn、TensorFlow、Pytorch 等，每种框架还有不同的版本分支，因此引进容器镜像来保障不同开发人员的需求是非常必要的，也是机器学习平台的基本功能。

选定容器技术后，如何支持、编排、多资源混合调度也是需要考量的。早期内部使用自研开发的容器调度编排工具，但是无法有效支持混合资源管理，所以转型过程中考虑引入业内活跃开源组件，通过与开源社区紧密联系保障架构的可持续演进。机器学习平台天生就和大数据强相关，因此 Yarn 作为大数据平台的调度核心是我们可选方向之一。但是 Yarn 对容器、GPU 的友好性比较差，同时业内 Kubernetes 已经成为容器编排、调度的事实标准，因此选择 Kubernetes 作为我们 AI 基础平台的底层支撑也就顺理成章了。

华泰证券的 AI 基础平台服务于量化平台、数据科学开发平台，因此天然需要支持异构框架，除了需要对接 Kubernetes 集群，还需要同时对接 Spark 集群实现离线学习。随着分布式计算的逐步成熟，以及深度学习模型的快速增长，业务对于在线深度学习的需求越来越强，因此华泰证券 AI 基础平台需要支持在线分布式机器学习，为用户带来更快的模型迭代速度，同时通过流数据及时修正模型参数，真正形成模型开发、发布、修正的闭环。因此我们在原生 Kubernetes 集群和 Spark 集群基础上实现了 JOB 类型任务、在线服务类型任务、SideCar 机制。同时由于原生的 Kubernetes 是通过内部 namespace 实现资源配额而无法实现跨集群资源配额限制，因此我们的 AI 基础平台额外实现了自身的配额管理，面对多集群实现统一的用户资源隔离。

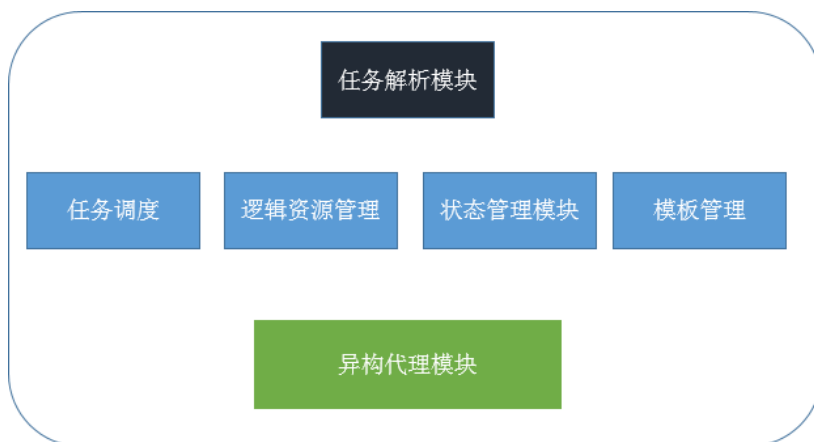
架构分析

AI 基础平台主要包括几大模块：

- 任务解析模块，负责将上游提供的 Job 任务进行解析，包括 Kubernetes 类

型任务、Spark 类型任务、DAG 类型任务、Batch Job 类型任务，等等；

- 任务调度模块，负责两级调度功能；
- 逻辑资源管理模块，负责统筹多集群、异构集群的资源管理；
- 状态管理模块，负责对 Job 任务进行状态探测和管理；
- 模板管理模块，负责抽象出可复用模板，实现模板复用；
- 异构代理模块，负责跨多集群、跨异构集群的对接。

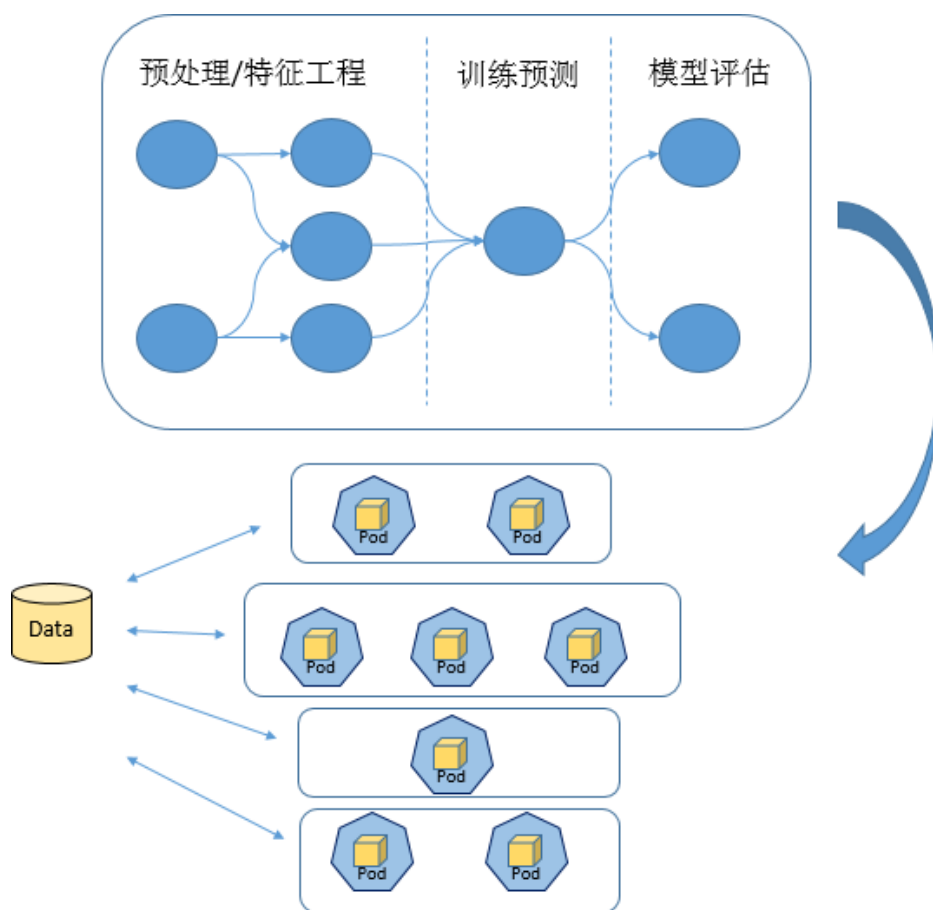


跨集群资源管理

针对跨集群管理用户组资源配额的需求，我们采用了逻辑资源管理，将异构集群内的所有资源进行逻辑分割，面对不同的用户组分配不同的资源配额，主要是为了整合不同集群的资源，不需要面对单个集群资源过大导致迁移困难问题，同时单集群迁移时可以实现在线不中断服务。



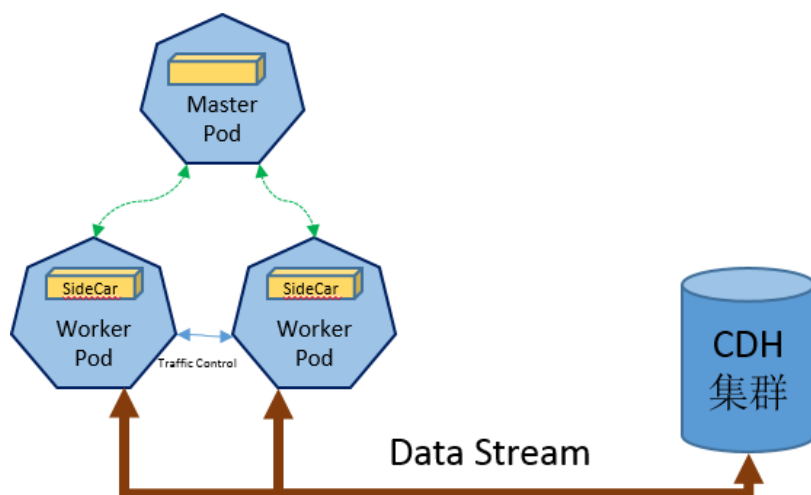
面对机器学习中最基础的任务调度问题，我们采用了任务调度模块，支持单次任务调度和定时任务调度，同时在面对机器学习的可视化建模，数据的抽取、转换、加载（ETL）等业务复杂情况下，需要基于 DAG 的调度管理。一个完整的 DAG 主要包括数据读取 / 数据预处理、特征工程、模型训练、模型预测、模型评估、模型固化。任务调度模块通过对任务中的 Node 信息解析，转化为上下文强依赖的任务，以此保证 DAG 的有向无环图特性。



Kubernetes 上的 Master-Worker 与 SideCar 模式

类似于 Spark on K8s 模式，一个 Job 任务发送到 AI 基础平台，将会在经过前置分解后，交由 Master Pod 进行整个 Job 任务的处理。同时 Worker Pod 使用 SideCar 模式与 Master Pod 保持心跳和状态同步，可以实现在无侵入客户应用

容器情况下完成定制化改造。Sidecar 在 Kubernetes 中是一个辅助容器的概念，和主容器跑在同一个 pod 中。同时 Sidecar 由我们平台团队自己提供，对应用户只需要提交自己的应用镜像，定义输入输出和特殊参数等。同时基于该 Master-Worker 与 SideCar 模式，可以方便地支持分布式框架。AI 基础平台可以定制化提供 SideCar 实现不同用户的需求，例如支持 CDH 环境 Kerberos 认证获取行情数据、特殊流量管控等等，而不需要用户修改自己的应用镜像。



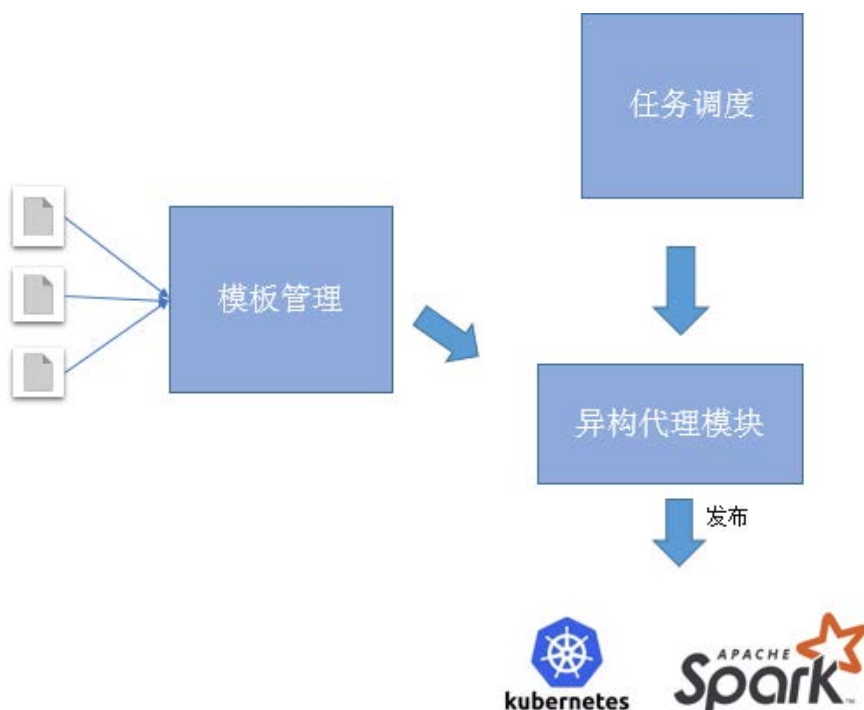
整个流程如下：

1. 用户向 AI 基础平台提交 Job 任务
2. AI 基础平台将 Job 任务分解，解析
3. 根据任务类型与逻辑资源管理判断提交任务的集群
4. 通过异构代理模块在 Kubernetes 集群中初始化 Master Pod
5. Master Pod 根据任务需求在集群中孵化 Worker Pod
6. Worker Pod 通过 SideCar 容器初始化工作上下文，与 Master Pod 交互
7. Worker Pod 内应用完成后，由 Master Pod 控制 SideCar 实现有状态容器管理，优雅退出

Module 形式管理

支持模板定义，简化了用户提交任务的参数复杂度。不同用户组所需的模板不同，例如量化平台某些团队所需的模板里面需要访问某些特定数据集，需要初始化

一些高频、低频因子数据，而另一些团队对于数据源的需求不同，但是对于整个团队而言基本上这些定制化的需求都相同。因此通过模板实现大部分配置通用化，可以让用户更专注于自身业务。

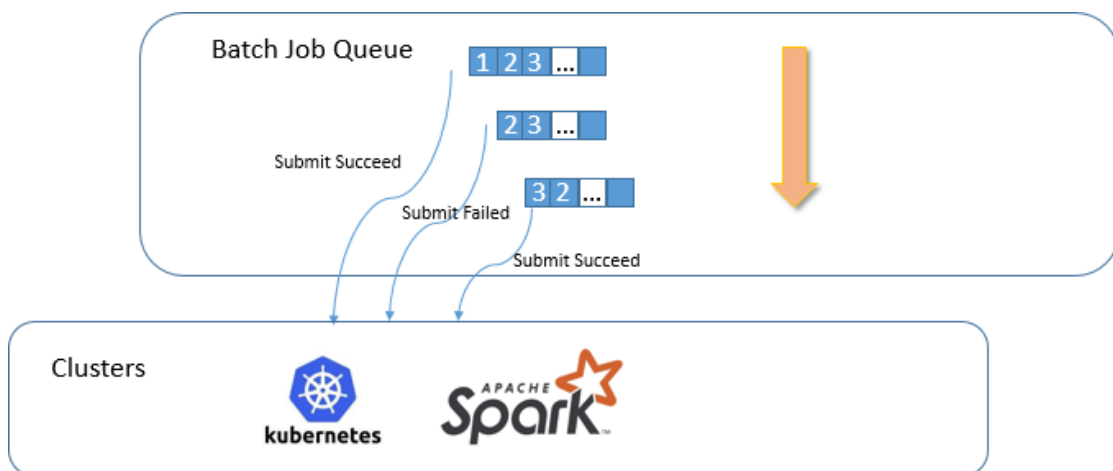


二级调度：让 Kubernetes 集群支持 Batch 类型

Kubernetes 默认的 Kube-scheduler 通过 list watch Kube-apiserver 获取需要 bind node 的 Pod，利用全集群串行，保障资源不会发生冲突。但是该串行是以 Pod 为粒度的，当出现前置 Batch Job 资源不够，而后置 Batch Job 某些 pod 资源满足，很可能会导致整个集群的资源死锁，比如两个 Batch Job 单独提交可以完成调度，但是同时提交则会每个 Batch Job 占用一些资源而无法同时启动。在许多机器学习的分布式框架中，比如 Ps-Worker，需要所有 worker 都启动后，训练才能正常进行。

为了解决无法支持 Batch Job 任务调度的问题，AI 基础平台采用了二级调度的方法。其核心思想是将一个 Batch Job 的所有 Pod 当做整体来调度，当该 Batch Job 所有 Pod 都能启动，才进行下一个 Batch Job 的调度。同时会动态调整 Batch Job 在调度时候的优先级，考虑在集群资源不足时优先保障资源需求较少的 Job

b 完成任务，提高用户的体验。从而避免了任务间的资源死锁，提高了资源的利用率。后期打算参考 Kube-batch 队列的机制，不同队列直接可以设置优先级，优先级高的队列的任务会优先得到调度，队列还可以设置权重，权重高的队列分配到的资源会更多。



目前华泰证券内部的 Spark 集群主要使用 Yarn 来支持资源调度，由于 Yarn-client 模式下 Application 发生故障时，直接在 Client 端查看日志、定位错误较为方便。因此目前华泰内部 Spark 集群主要采用该模式。而 AI 基础平台二次调度模式对于 Yarn-client 模式同样可以无缝支持，即当该 Batch Job 所有 Executor 都能启动，才进行下一个 Batch Job 的调度。

未来展望

对于机器学习和深度学习来讲，算力集中化和工程平台化是未来持续演进的方向。脱离以前的烟囱式发展，赋能平台用于管理不断爆发式增长的算力，持续提高资源利用率，降低算力使用成本，对算法用户便捷可用，是 AI 基础平台的持续演进方向。

未来华泰证券的 AI 基础平台将会支持更多的通用分布式训练框架，比如 Ray，以帮助用户进行代码分布式改造。由于证券行业的特殊性，AI 基础平台对数据的访问控制，对流量策略会有很强的安全性考量，因此考虑利用 Service Mesh 实现

全平台的访问策略管理。AI 基础平台最重要的是稳定性、性能和安全，因此下一阶段希望能将基于 Job 粒度的监控和任务状态异常探测加入其中。

作者介绍:

刘赐民，曾就职于中兴通讯、苏宁易购，一直从事云平台、大数据平台相关工作，目前是华泰证券数据科学中心平台团队的资深工程师，也是 Kubernetes、CNI、Kata-container 等项目的 contributor。

Tensorflow 2.0到底好在哪里？

作者 | Paul

译者 | 王强

策划 | 蔡芳芳



TensorFlow 现在已经不仅是一个单纯的工具包了，而是发展成为了一个平台，在易用性、分布式训练和部署等方面都取得了长足的进步。

如今已经没有人质疑机器学习和深度学习的重要性了。数十年来这一行业让人们见识过无数承诺、骗局和失望，时至今日两大技术终于带来了众多实际应用。机器学习或深度学习应用离充分完善还有很长的路要走，但现有的成果已经非常喜人了。

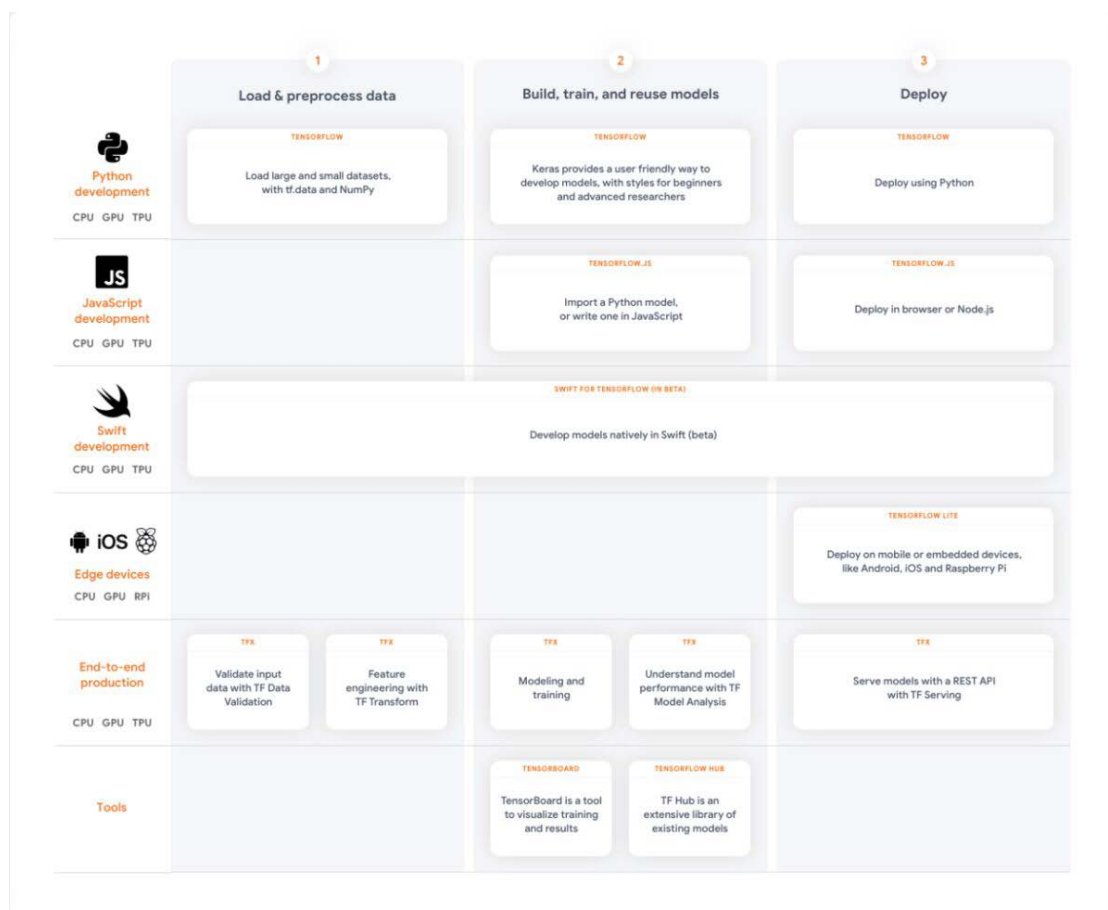
在所有优秀的机器学习和深度学习框架中，TensorFlow 是最成熟的，在研究论文中被引用最多（就算排除谷歌员工的引用也是如此），也有着最出色的生产实践案例。它可能不是最容易学习的框架，但比起它 2016 年刚发布时的情况，现在的

TensorFlow 对新人要友好得多。TensorFlow 还是许多谷歌服务的基础。

TensorFlow 2.0 网站将该项目描述为“端到端开源机器学习平台”。实际上 TensorFlow 已进化成为一个更全面的“工具、库和社区资源生态系统”，可帮助研究人员构建和部署人工智能助力的应用。

TensorFlow 2.0 有四大组成部分：

- TensorFlow 核心，一个用于开发和训练机器学习模型的开源库；
- TensorFlow.js，一个用于在浏览器和 Node.js 上训练和部署模型的 JavaScript 库；
- TensorFlow Lite，一个轻量级库，用于在移动和嵌入式设备上部署模型；
- TensorFlow Extended，一个在大型生产环境中准备数据、训练、验证和部署模型的平台。



TensorFlow 2.0 生态系统包括对 Python、JavaScript 和 Swift 的支持，以及对云、浏览器和边缘设备的部署支持。TensorBoard（可视化）和 TensorFlow Hub（模型库）都是很有用的工具。TensorFlow Extended（TFX）则支持端到端生产流水线。

在以前的文章中，我曾评测过 [TensorFlow r0.10（2016）](#) 和 [TensorFlow 1.5（2018）](#)。

这些年来，TensorFlow 逐渐发展为基于数据流图的机器学习和神经网络库，拥有较高的学习曲线和一个底层 API。对普通人来说 TensorFlow 2.0 已经不再那么难学了，2.0 版本还拥有一个高级 Keras API，支持在 JavaScript 中运行、在移动和嵌入式设备上部署以及在大型生产环境中运行。

TensorFlow 的竞争对手包括 Keras（可能使用除 TensorFlow 之外的其他后端）、MXNet（与 Gluon）、PyTorch、Scikit-learn 和 Spark MLlib。最后两个主要是机器学习框架，缺乏深度学习的相关设施。

你可以同时使用多种方案。在单条流水线中同时使用多个框架是非常合理的，例如使用 Scikit-learn 准备数据并使用 TensorFlow 训练模型。

TensorFlow 核心

TensorFlow 2.0 的设计重点就是简洁易用，它的新特性包括 Eager Execution、直观的高级 API 以及在任何平台上灵活构建模型等更新。前两个特性值得深入研究。

Eager Execution

Eager Execution 意味着 TensorFlow 代码被定义后会立即运行，而不是先将节点和边缘添加一个图上，稍后再在一个会话中运行——后者是 TensorFlow 原来使用的模式。例如，TensorFlow r0.10 早期版本的“Hello, World!”脚本如下所示：

```
1 $ python
2 ...
3 >>> import tensorflow as tf
4 >>> hello = tf.constant('Hello, TensorFlow!')
5 >>> sess = tf.Session()
```



```
6 >>> print(sess.run(hello))
7 Hello, TensorFlow!
8 >>> a = tf.constant(10)
9 >>> b = tf.constant(32)
10 >>> print(sess.run(a + b))
11 42
12 >>> exit()
```

注意这里使用的 `tf.Session()` 和 `sess.run()`。在 TensorFlow 2.0 中，Eager Execution 模式是默认的，如下例所示。

Setup and basic usage

```
from __future__ import absolute_import, division, print_function, unicode_literals

!pip install -q tensorflow==2.0.0-beta1
import tensorflow as tf

import cProfile
```

In Tensorflow 2.0, eager execution is enabled by default.

```
tf.executing_eagerly()
```

```
True
```

Now you can run TensorFlow operations and the results will return immediately:

```
x = [[2.]]
m = tf.matmul(x, x)
print("hello, {}".format(m))
```

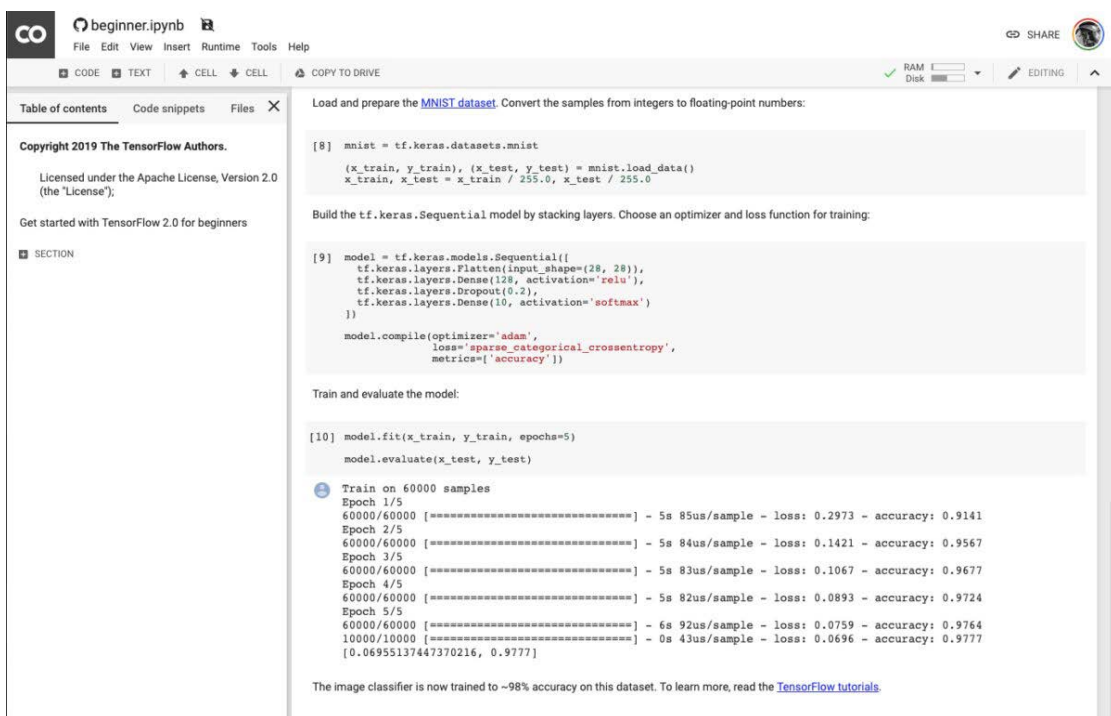
```
hello, [[4.]]
```

Enabling eager execution changes how TensorFlow operations behave—now they immediately evaluate and return their values to Python. `tf.Tensor` objects reference concrete values instead of symbolic handles to nodes in a computational graph. Since there isn't a computational graph to build and run later in a session, it's easy to inspect results using `print()` or a debugger. Evaluating, printing, and checking tensor values does not break the flow for computing gradients.

TensorFlow 2.0 中的 Eager Execution 示意。这个笔记本可以在谷歌 Colab 中运行，或者在安装好预设的 Jupyter 笔记本中也能运行。

tf.keras

前面的两个示例都使用了底层 TensorFlow API。“活用 TensorFlow 2.0 指南”则使用了高级别的 `tf.keras` API 取代了旧的底层 API；这将大大减少你需要编写的代码量。你只需要每层写一行代码就能构建 Keras 神经网络，如果能善用循环结构的话需要的代码就更少了。下面的示例演示了 Keras 数据集和顺序模型 API，它们运行在谷歌 Colab 中；谷歌 Colab 用来运行 TensorFlow 样本和实验很方便（还是免费的）。请注意，Colab 除了 CPU 外还提供了 GPU 和 TPU 实例。



```
Load and prepare the MNIST dataset. Convert the samples from integers to floating-point numbers:

[8] mnist = tf.keras.datasets.mnist
(x_train, y_train), (x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0

Build the tf.keras.Sequential model by stacking layers. Choose an optimizer and loss function for training:

[9] model = tf.keras.models.Sequential([
    tf.keras.layers.Flatten(input_shape=(28, 28)),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(10, activation='softmax')
])

model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

Train and evaluate the model:

[10] model.fit(x_train, y_train, epochs=5)
model.evaluate(x_test, y_test)

Train on 60000 samples
Epoch 1/5
60000/60000 [=====] - 5s 85us/sample - loss: 0.2973 - accuracy: 0.9141
Epoch 2/5
60000/60000 [=====] - 5s 84us/sample - loss: 0.1421 - accuracy: 0.9567
Epoch 3/5
60000/60000 [=====] - 5s 83us/sample - loss: 0.1067 - accuracy: 0.9677
Epoch 4/5
60000/60000 [=====] - 5s 82us/sample - loss: 0.0893 - accuracy: 0.9724
Epoch 5/5
60000/60000 [=====] - 6s 92us/sample - loss: 0.0759 - accuracy: 0.9764
10000/10000 [=====] - 0s 43us/sample - loss: 0.0696 - accuracy: 0.9777
[0.06955137447370216, 0.9777]

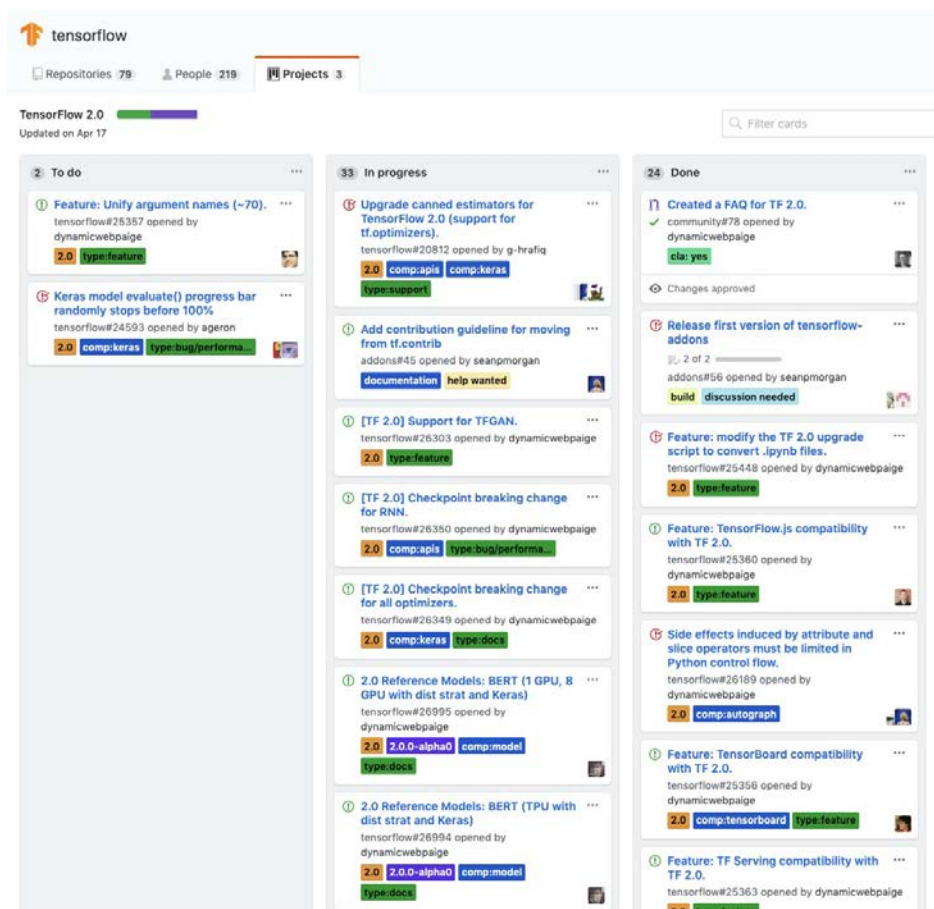
The image classifier is now trained to ~98% accuracy on this dataset. To learn more, read the TensorFlow tutorials.
```

上图是一个 TensorFlow 笔记本，用于训练基本的神经网络对 MNIST 手写数字图像进行分类。这是在谷歌 Colab 上运行的 TensorFlow 示例笔记本。注意这里使用了 `tf.keras.datasets` 来提供 MNIST 图像。

过渡到 TensorFlow 2.0

在撰写本文时，从 TensorFlow 1.14 到 TensorFlow 2.0 的过渡还在进行中，如

下面的截图所示。目前有两个待办事项、23 个正在进行的任务以及 34 个已完成的任务。正在进行的任务中既有完全没有进展的（可能是因为当事人退出），也有几乎完成的（代码已进入存储库主分支，但尚未经过审查和部署）。



从 TensorFlow 1.14 到 TensorFlow 2.0 的过渡状态可通过以下网址跟踪：<https://github.com/orgs/tensorflow/project/4>。此截图生成于 6 月 21 日，但请注意该页面自 4 月 17 日以来都未更新。

将模型升级到 TensorFlow 2.0

就像很多开源项目的大版本更新一样，TensorFlow 2.0 对 API 引入了许多重大更改，需要你随之升级你的代码。所幸我们有一个随 TensorFlow 2.0 自动安装的 Python 代码升级脚本，还有一个给无法升级的 API 符号用的兼容模块（compat.v1），只需使用字符串替换即可。运行升级脚本后，你的程序可能会在

TensorFlow 2.0 上运行，但是会引用 `tf.compat.v1` 命名空间，你得在有空的时候处理一下以保持代码清洁。此外，你可以将 GitHub repos 上的 [Jupyter 笔记本](#) 升级到 TensorFlow 2.0。

使用 `tf.function`

Eager Execution 模式的缺点是可能会损失一些编译和执行流程图的性能。有一种方法可以在不完全关闭 Eager Execution 模式的情况下恢复性能，它就是 `tf.function`。

基本上，当你使用 `@tf.function` 注释一个函数时后者将被编译成一个图，它和它调用的任何函数就（可能）更快地执行、支持在 GPU 或 TPU 上运行，还支持导出到 `SavedModel`。`tf.function` 的一项便利的新功能是 [AutoGraph](#)，它自动将 Python 控制流语句编译为 TensorFlow 控制操作。

分布式训练

以前我研究 TensorFlow 时有两种方法可以运行分布式训练：使用异步参数服务器，或使用第三方的 Horovod 项目——该项目是同步的并使用 all-reduce 算法。现在新版有五种原生的 TensorFlow 分布式策略，以及一个用于选择所需策略的 API，`tf.distribute.Strategy`；它允许你跨多个 GPU、多台计算机或多个 TPU 分发训练。此 API 还可用于在不同平台上分发评估和预测。

Training API	MirroredStrategy	TPUStrategy	MultiWorkerMirroredStrategy	CentralStorageStrategy	ParameterServerStrategy
Keras API	Supported	Support planned in 2.0 RC	Experimental support	Experimental support	Supported planned in post 2.0
Custom training loop	Experimental support	Experimental support	Support planned in 2.0 RC	Support planned in 2.0 RC	No support yet
Estimator API	Supported	Supported	Supported	Supported	Supported

TensorFlow 现在支持五种原生分布式策略，TensorFlow 2.0 beta 还对三种训练 API 提供不同级别的支持。

[TensorFlow.js](#)

[TensorFlow.js](#) 是一个用于在 JavaScript 中开发和训练机器学习模型并在浏览器或 Node.js 中部署它们的库。还有一个基于 TensorFlow.js 的高级库 [ml5.js](#)，使用户无需直接面对复杂的张量和优化器。

[TensorFlow.js](#) [运行在浏览器中](#)，支持移动设备和桌面设备。如果你的浏览器支持 WebGL 着色器 API，TensorFlow.js 就可以使用它们并利用 GPU 计算能力，带来相比 CPU 后端多达 100 倍的加速效果。在配备 GPU 的计算机上，[TensorFlow.js](#) [演示](#)在浏览器中运行得非常快。

在 Node.js 上，TensorFlow.js 可以使用已安装的 TensorFlow 版本作为后端，或者运行基本的 CPU 后端。CPU 后端是纯 JavaScript 的，并行优化不够充分。

你可以在浏览器上运行官方 [TensorFlow.js](#) [模型](#)、[转换 Python 模型](#)、[使用迁移学习](#)来用你自己的数据自定义模型，以及直接在 [JavaScript](#) 中构建和训练模型。

[TensorFlow Lite](#)

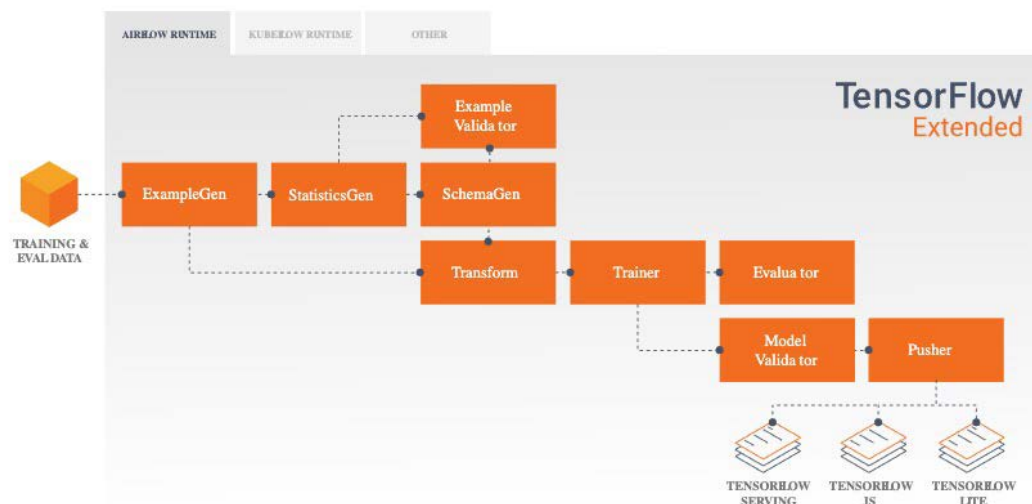
[TensorFlow Lite](#) 是一个用于设备上推断的开源深度学习框架。它目前为 iOS、ARM64 和树莓派构建了模型。

TensorFlow Lite 的两大组件分别是解释器和转换器。解释器用来在许多不同的硬件类型上运行特别针对优化的模型。转换器则将 TensorFlow 模型转换为高效形式供解释器使用，并可引入优化以改善程序体积和性能。预先训练的模型有图像分类、对象检测、智能回复、姿势估计和语义分割等类型。还有用于手势识别、图像分类、对象检测和语音识别的示例应用。

[TensorFlow Extended](#)

[TensorFlow Extended \(TFX\)](#) 是用于部署机器学习生产流水线的端到端平台。你训练好一个模型就要考虑使用 TFX 了。其流水线包括[数据验证](#)、[功能设计](#)、[建模](#)、[模型评估](#)、[服务推断](#)以及在线平台、移动原生平台和 JavaScript 平台的部署管

理。下图显示了 TFX 流水线的组件是如何组合在一起的。



TensorFlow Extended 示意图。

Swift for TensorFlow

[Swift for TensorFlow](#) 是一个用于深度学习和可微分编程的下一代（并且仍不稳定）平台。它有一个用于训练的高级 API，看起来很像 Python TensorFlow，但它也支持使用 `@differentiable` 属性自动微分构建到 Swift 编译器的一个 fork 中。Swift for TensorFlow 可以导入和调用 Python 代码，使开发者更容易从 Python TensorFlow 过渡过来。

TensorFlow 工具

目前有七种工具支持 TensorFlow。它们分别是 TensorBoard，TensorFlow 图的一组可视化工具；TensorFlow Playground，一个可调节的在线神经网络；CoLab，又名 Colaboratory，一个免费的在线 Jupyter 笔记本环境；What-If 工具，可用于探索和调试 TensorBoard、CoLab 或 Jupyter 笔记本中的模型；ML Perf，流行的机器学习基准测试套件；XLA（加速线性代数），一种用于线性代数的特定领域编译器，可优化 TensorFlow 计算；和 TFRC（TensorFlow 研究云），一个由 1000 多个云

TPU 组成的集群，研究人员可以申请免费使用。

总的来说，TensorFlow 2.0 测试版已经在很多方面取得了很大进展。在 `tf.keras` API 和 Eager Execution 模式的帮助下，新版核心框架更易于学习、使用和调试。你可以有选择地将要编译的函数标记为图形。有五种方法可以进行分布式训练和推断。

新版有一套完整的组件，也就是 TFX，用于构建从数据验证到推断模型管理的全套机器学习流水线。你可以在浏览器或 Node.js 上运行 TensorFlow.js，还可以在移动设备和嵌入式设备上运行 TensorFlow Lite。最后，Swift for TensorFlow 将为模型构建开辟新的可能性。

原文链接：<https://www.infoworld.com/article/3405641/tensorflow-2-review-easier-end-to-end-machine-learning.html>

基于深度学习的推荐系统效果遭质疑，它真的有带来实质性进展吗？

作者 | Maurizio Ferrari Dacrema

译者 | 马卓奇

策划 | 蔡芳芳

深度学习已经成为推荐系统领域的首选方法，但与此同时，已有一些论文指出了目前应用机器学习的研究中存在的问题，例如新模型结果的可复现性，或对比实验中基线的选择。这篇论文发表在推荐系统顶级会议 ACM RecSys 2019 上，作者对过去几年在顶级会议上发表的 18 种 top-n 推荐任务的算法进行了系统分析。

作者发现，这些算法中只有 7 种算法可以合理复现算法结果，而其中 6 种方法都被经典的启发式算法所超越，例如基于最近邻或基于图的方法。作者通过这篇论文揭示了当前机器学习领域的一些潜在问题，并呼吁大家改进该领域的科学实践。本文是 AI 前线第 86 篇论文导读。

Are We Really Making Much Progress? A Worrying Analysis of Recent Neural Recommendation Approaches

Maurizio Ferrari Dacrema, Paolo Cremonesi, Dietmar Jannach

(Submitted on 16 Jul 2019)

Deep learning techniques have become the method of choice for researchers working on algorithmic aspects of recommender systems. With the strongly increased interest in machine learning in general, it has, as a result, become difficult to keep track of what represents the state-of-the-art at the moment, e.g., for top-n recommendation tasks. At the same time, several recent publications point out problems in today's research practice in applied machine learning, e.g., in terms of the reproducibility of the results or the choice of the baselines when proposing new models. In this work, we report the results of a systematic analysis of algorithmic proposals for top-n recommendation tasks. Specifically, we considered 18 algorithms that were presented at top-level research conferences in the last years. Only 7 of them could be reproduced with reasonable effort. For these methods, it however turned out that 6 of them can often be outperformed with comparably simple heuristic methods, e.g., based on nearest-neighbor or graph-based techniques. The remaining one clearly outperformed the baselines but did not consistently outperform a well-tuned non-neural linear ranking method. Overall, our work sheds light on a number of potential problems in today's machine learning scholarship and calls for improved scientific practices in this area. Source code of our experiments and full results are available at: [this https URL](https://github.com/mferrari/recommender-systems).

1 引言

在短短几年内，深度学习技术在推荐系统算法研究中占据了主导地位。随着人们对机器学习的兴趣普遍增加，发表论文数量越来越多，以及深度学习在视觉或语言处理等其他领域的成功，人们可以预见，这些工作也会为推荐系统领域带来实质

性的进展。然而，在机器学习的其他应用领域中，所取得的进展并不总是如预期的那样好。

在推荐系统领域，即使是最新的推荐方法，在大多数情况下也不能超越经典的方法（例如基于近邻的方法）。这些关于在应用机器学习中取得的真正进展的问题并不是最新提出的，也与基于深度学习的研究无关。早在 2009 年，研究人员通过对 ad-hoc 检索任务的算法分析得出结论：尽管该领域内发表了许多论文，但这些论文中提到的改进并没有“累积”。

这种现象的出现有不同因素，包括（i）基线方法较弱；（ii）用较差的方法作为新的基线；以及（iii）比较或复现其他论文结果具有困难性。第一个问题在于方法对比时基线的选择。有时对于给定的任务和数据集，选择的基线太弱，或基线参数没有得到适当的调整。有时基线是从新提出的算法簇中选择的，例如，一个新的深度学习算法只与深度学习基线进行比较。这种方法强制传播了弱基线。此外，随着论文的不断发表，越来越难跟上最先进基线的发展。

除了基线的问题外，另一个挑战是研究人员使用各种各样的数据集、评估方法、性能度量和数据预处理步骤，因此很难确定哪种方法在不同的应用场景中是最好的。当研究人员不公开源代码和数据时，这个问题尤其突出。虽然现在越来越多的研究人员会公布算法的源代码，但这并不是通用规则，即使顶级会议或顶级期刊也没有这样的要求。而且即使发布了代码，有些代码也是不完整的，并不包括数据预处理、参数调整或评估程序。

最后，另一个问题可能普遍存在于应用机器学习的研究实践。缺少审稿人，或对论文作者的不当激励，会刺激某些特定类型的研究。以及研究领域对抽象精确性度量的过度关注，或者只关心机器学习研究中“顶级期刊能发表的”内容。

这篇论文中，作者的目标是阐明上述问题是否也存在于基于深度学习的推荐算法领域。作者主要关注以下两个问题：

（1）可复现性：该领域的近期研究有多少是可复现的（通过合理方法）？

（2）进展：与相对简单但经过良好调整的基线方法相比，近期研究取得了多

少实际性进展？

为了回答这些问题，作者进行了一项系统的研究。作者从 KDD、SIGIR、WWW 和 RecSys 这四大顶会中找到了 18 篇 top-n 推荐任务中基于深度学习的相关论文。

第一步，对于公开源代码和实验数据集的论文，作者尝试复现论文中报告的结果。最后，仅有 7 篇论文可以复现结果。

第二步，作者重新执行了原始论文中报告的实验，但在比较中增加了额外的基线方法。出乎意料的是，研究显示，在绝大多数被调查的方法中（7 个方法中有 6 个方法），所提出的深度学习方法均被经典的基线方法所超越。另一个方法中，即使是个性化的基线方法（向每个人推荐最受欢迎的项目），在某些评价指标下的表现也是最好的。

该论文的第一个贡献在于评估了该领域论文的可复现程度，论文的第二个贡献在于提出一个与机器学习的当前研究实践相关的更深远的问题。

2 研究方法

2.1 收集可复现论文

作者收集了 2015 年至 2018 年期间出现在以下四个会议中的长论文：KDD、SIGIR、WWW 和 RecSys。如果一篇论文（a）提出了一种基于深度学习的技术，（b）关注 top-n 推荐任务，那么就算作一篇相关论文。经过筛选，作者收集了 18 篇相关论文。

下一步，作者尝试复现这些论文中报告的结果。作者尽可能多地依赖论文原作者自己提供的源代码和实验中使用的数据。理论上说，应该可以只使用论文中的技术描述来复现已发表的结果。但实际上算法和评估程序的实现包含许多微小细节，可能会对实验结果产生影响。因此，作者尝试从原作者那里获得所有相关论文的代码和数据。如果满足以下条件，则认为论文是可复现的：

- 源代码可用，或者只需要少量的修改即可正常运行。

- 原论文中至少有一个数据集可用。另一个要求是，原论文中使用的训练 - 测试划分方法是公开的，或者可以根据文中的信息重构。

否则，则认为论文是不可复现的。根据该标准，可复现的论文列表如表 1 所示：

Conference	Rep. / Non-rep.	Reproducible
KDD	3/4 (75%)	[17], [23], [48]
RecSys	1/7 (14%)	[53]
SIGIR	1/3 (30%)	[10]
WWW	2/4 (50%)	[14], [24]
Total	7/18 (39%)	
<i>Non-reproducible:</i> KDD: [43], RecSys: [41], [6], [38], [44], [21], [45], SIGIR: [32], [7], WWW: [42], [11]		

总的来说，只有大约三分之一的论文可复现。

2.2 评价方法

测量方法

在这项工作中，作者通过分解原始代码来复现论文，以应用与原论文中相同的评估过程。分解的方式是将训练、超参数优化和预测的代码与评估代码分离。并且将评估代码也用于基线方法。

基线

作者在实验中考虑了以下基线方法：

- **TopPopular:** 一种非个性化的方法，向每个人推荐最流行的项目。
- **ItemKNN:** 基于 k 近邻 (kNN) 和 item-item 相似度的传统协同过滤方法。
- **UserKNN:** 一种基于邻域的协同用户相似性方法。
- **ItemKNN-CBF:** 一种基于邻域内容过滤 (CBF) 的方法，通过使用项目内容特征 (属性) 计算项目相似性。

- **ItemKNN-CFCBF**: 基于项目相似性的混合 CF+CFB 算法。
- **P3 α** : 一种简单的基于图的算法, 实现了用户和项目之间的随机行走。
- **RP3 β** : P3 α 的另一个版本。

3 DNN 方法与基线对比实验

3.1 协作存储网络 (Collaborative Memory Networks, CMN)

CMN 方法在 SIGIR18 会议上提出, 将记忆网络和神经注意力机制与隐因素和邻域模型结合。CMN 作者将该方法与不同的矩阵分解和神经推荐方法, 以及 ItemKNN 算法进行了比较。采用了三个数据集用于评估: Epinions、CiteULike-a 和 Pinterest。原论文给出了最优参数, 但没有提供如何调整基线实验的信息。点击率和 NDCG 是原论文采用的评价指标。原论文报告的结果表明, CMN 在所有的度量标准上都优于其他的基线方法。

CMN 所有数据集上的实验都是可复现的。对于简单基线进行的额外实验, 作者针对点击率度量优化了基线参数。在三个数据集上的实验结果如表 2 所示。

CiteULike-a				
	HR@5	NDCG@5	HR@10	NDCG@10
TopPopular	0.1803	0.1220	0.2783	0.1535
UserKNN	0.8213	0.7033	0.8935	0.7268
ItemKNN	0.8116	0.6939	0.8878	0.7187
P ³ α	0.8202	0.7061	0.8901	0.7289
RP ³ β	0.8226	0.7114	0.8941	0.7347
CMN	0.8069	0.6666	0.8910	0.6942
Pinterest				
	HR@5	NDCG@5	HR@10	NDCG@10
TopPopular	0.1668	0.1066	0.2745	0.1411
UserKNN	0.6886	0.4936	0.8527	0.5470
ItemKNN	0.6966	0.4994	0.8647	0.5542
P ³ α	0.6871	0.4935	0.8449	0.5450
RP ³ β	0.7018	0.5041	0.8644	0.5571
CMN	0.6872	0.4883	0.8549	0.5430
Epinions				
	HR@5	NDCG@5	HR@10	NDCG@10
TopPopular	0.5429	0.4153	0.6644	0.4547
UserKNN	0.3506	0.2983	0.3922	0.3117
ItemKNN	0.3821	0.3165	0.4372	0.3343
P ³ α	0.3510	0.2989	0.3891	0.3112
RP ³ β	0.3511	0.2980	0.3892	0.3103
CMN	0.4195	0.3346	0.4953	0.3592

结果表明，在优化基线方法之后，CMN 在任何数据集上都不能取得最好的表现。对于 CiteULike-a 和 Pinterest 数据集，至少有两种个性化基线方法在任何度量上都优于 CMN 方法。基本上所有的个性化基线方法都比 CMN 效果好。对于 Epinions 数据集，出乎意料的是，原始文献中没有提及的 TopPopular 方法在很大程度上优于所有其他算法。在这个数据集上，CMN 确实比基线方法要好。因此，CMN 在这个相对较小且非常稀疏的数据集上的成功，可能与数据集的特殊性或 CMN 的受欢迎度（popularity）偏置有关。分析表明，与其他数据集相比，Epinions 数据集的受欢迎程度的分布确实更加不均匀（基尼指数为 0.69，而 CiteULike 基尼指数为 0.37）。

3.2 基于元路径上下文的推荐方法（Metapath based Context for REcommendation, MCRec）

MCRec 方法发表在 KDD18，是一个基于元路径的模型，它利用辅助信息实现 top-n 推荐任务。原文献作者在三个小数据集（MovieLens100k、LastFm 和 Yelp）上对不同复杂度的各种模型，以及 MCRec 的四个变体进行了基准测试。原文献通过创建 80/20 随机训练测试划分，进行 10 次交叉验证。选择 MF 和 NeuMF 作为基线。但只有 MovieLens 数据集提供了数据划分，原文献没有给出基线超参数调参的具体信息。原文献采用的评价指标为精确度、召回率和 NDCG。但是论文中实现的 NDCG 方法较为奇怪，所以作者采用了标准的 NDCG 评价程序。

	PREC@10	REC@10	NDCG@10
TopPopular	0.1907	0.1180	0.1361
UserKNN	0.2913	0.1802	0.2055
ItemKNN	0.3327	0.2199	0.2603
P ³ α	0.2137	0.1585	0.1838
RP ³ β	0.2357	0.1684	0.1923
MCRec	0.3077	0.2061	0.2363

表 3 表明，当正确设置传统的 ItemKNN 方法时，该方法在所有性能指标上都优于 MCRec。原始论文除了使用一种不常见的 NDCG 方法外，作者还发现了其他潜在的方法学问题。如前所述，MF 和 NeuMF 基线的超参数没有针对给定数据集进行优化，而是取自原始论文。此外，通过查看提供的源代码，可以看到作者报告的是不同 epoch 中选择的最佳结果，这是不恰当的。

3.3 协同变分自动编码器（Collaborative Variational Autoencoder, CVAE）

CVAE 方法发表在 KDD18，该模型以无监督的方式从内容数据中学习深度隐表示，并从内容和排序中学习项目和用户之间的隐式关系。

该方法在两个比较小的 CitULike 数据集（135K 和 205K 次交互）上进行评估，分别测试了这两个数据集的稀疏版本和密集版本。原文献中的基线实验包括三个最

新的深度学习模型以及协同主题回归（CTR）。每个方法的参数都是基于验证集进行调整的。采用不同的列表长度（50 至 300）的召回率作为评价指标。采用随机数据划分，重复 5 次测量。

	REC@50	REC@100	REC@300
TopPopular	0.0044	0.0081	0.0258
UserKNN	0.0683	0.1016	0.1685
ItemKNN	0.0788	0.1153	0.1823
$P^3\alpha$	0.0788	0.1151	0.1784
$RP^3\beta$	0.0811	0.1184	0.1799
ItemKNN-CFCBF	0.1837	0.2777	0.4486
CVAE	0.0772	0.1548	0.3602

原文献作者共享了代码和数据集。通过对基线进行微调，得到了表 4 所示的稠密 CiteULike-a 数据集的结果。对于最短的列表长度 50，即使大多数纯 CF 基线方法在这个数据集上也优于 CVAE 方法。在较长的列表长度下，ItemKNN-CFCBF 方法获得了最佳结果。稀疏 CiteULike-t 数据集上也得到了类似的结果。一般来说，在列表长度为 50 时，ItemKNN-CFCBF 在所有测试配置中始终优于 CVAE。只有在更长的列表长度（100 及以上）时，CVAE 才能在两个数据集上超越基线方法。总的来说，只有在某些配置中，并且很长且相当不常见的推荐截止阈值下 CVAE 才优于基线。然而，这种列表长度的使用是不合理的。

3.4 协同深度学习（Collaborative Deep Learning, CDL）

上述的 CVAE 方法将 KDD15 中经常引用的 CDL 方法作为其基线之一。CDL 是叠置去噪自动编码器（SDAE）和协同滤波联合学习的概率前馈模型。原文献中的评估表明，与 CTR 方法相比，CDL 方法的表现较好，尤其是在稀疏数据情况下。

	REC@50	REC@100	REC@300
TopPopular	0.0038	0.0073	0.0258
UserKNN	0.0685	0.1028	0.1710
ItemKNN	0.0846	0.1213	0.1861
$P^3\alpha$	0.0718	0.1079	0.1777
$RP^3\beta$	0.0800	0.1167	0.1815
ItemKNN-CBF	0.2135	0.3038	0.4707
ItemKNN-CFCBF	0.1945	0.2896	0.4620
CDL	0.0543	0.1035	0.2627

作者复现了 CDL 的研究结果，得出了表 5 中密集型 CiteULike-a 数据集的结果。不足为奇，在前一节中优于 CVAE 的基线也优于 CDL，而且对于短列表长度而言，纯 CF 方法优于 CDL 方法。然而，当列表长度超过 100 时，CDL 具有更高的召回率。通过对比 CVAE 和 CDL 的结果，作者发现新提出的 CVAE 方法确实优于 CDL 方法，这表明 CAVE 方法的确取得了进展。然而在大多数情况下，这两种方法的表现都不如简单的基线方法。

3.5 神经协同过滤（Neural Collaborative Filtering, NCF）

基于神经网络的协同过滤方法在 WWW17 会议上提出，通过用一种可以从数据中学习任意函数的神经网络结构代替了内积来推广矩阵分解。该方法在两个数据集（MovieLens1M 和 Pinterest）上进行了评估，分别包含 100 万和 150 万次交互。在评价过程中采用了“留一法”。原文献结果表明，当使用点击率和 NDCG 作为评价指标时，NeuMF（NCF 的变体）比现有的矩阵因子分解模型更为有利。

实验结果如表 6 所示。在 Pinterest 数据集上，个性化基线方法在所有评价标准上都比 NeuMF 稍微好一些，或者表现一致。对于 MovieLens 数据集，NeuMF 的结果几乎与最佳基线 $RP^3\beta$ 相同。

	Pinterest			
	HR@5	NDCG@5	HR@10	NDCG@10
TopPopular	0.1663	0.1065	0.2744	0.1412
UserKNN	0.7001	0.5033	0.8610	0.5557
ItemKNN	0.7100	0.5092	0.8744	0.5629
$P^3\alpha$	0.7008	0.5018	0.8667	0.5559
$RP^3\beta$	0.7105	0.5116	0.8740	0.5650
NeuMF	0.6915	0.4879	0.8657	0.5447
	Movielens 1M			
	HR@5	NDCG@5	HR@10	NDCG@10
TopPopular	0.3043	0.2062	0.4531	0.2542
UserKNN	0.4916	0.3328	0.6705	0.3908
ItemKNN	0.4829	0.3328	0.6596	0.3900
$P^3\alpha$	0.4811	0.3331	0.6464	0.3867
$RP^3\beta$	0.4922	0.3409	0.6715	0.3991
NeuMF	0.4980	0.3445	0.6725	0.4011
PureSVD	0.5377	0.3802	0.6896	0.4294

由于 MovieLens 数据集被广泛用于评估新模型，因此作者使用基本矩阵分解方法（此处称为 pureSVD）进行了额外的实验。优化参数后，作者发现 pureSVD 确实比基线方法好，而且在这个数据集上也明显优于 NeuMF。

3.6 光谱协同过滤（Spectral Collaborative Filtering, SpectralCF）

SpectralCF 发表在 RecSys18 上，采用光谱图理论的概念，旨在专门解决冷启动问题。该方法在三个公共数据集（MovieLens1m、HetRec 和 Amazon Instant Video）上进行评估，并采用了多种基线方法，包括最近的神经网络方法和因子分解和排序技术。实验采用 80/20 训练 - 测试随机划分，并使用不同截止点的召回率和平均精度（MAP）作为评价指标。

对于 MovieLens 数据集，原文献作者共享了使用的训练和测试数据集以及代码。对于其他数据集，数据划分没有公布，因此作者按照文中的描述自己创建了划分方式。

对于 HetRec 和 Amazon Instant Video 数据集，所有的基线方法，包括 TopPopular 方法，在所有度量指标上都优于 SpectralCF。然而，在原文献提供的 MovieLens 数据划分上运行代码时，SpectralCF 比所有的基线都要好很多。

	Cutoff 20		Cutoff 60		Cutoff 100	
	REC	MAP	REC	MAP	REC	MAP
TopPopular	0.1853	0.0576	0.3335	0.0659	0.4244	0.0696
UserKNN CF	0.2881	0.1106	0.4780	0.1238	0.5790	0.1290
ItemKNN CF	0.2819	0.1059	0.4712	0.1190	0.5737	0.1243
$P^3\alpha$	0.2853	0.1051	0.4808	0.1195	0.5760	0.1248
$RP^3\beta$	0.2910	0.1088	0.4882	0.1233	0.5884	0.1288
SpectralCF	0.1843	0.0539	0.3274	0.0618	0.4254	0.0656

因此，作者分析了 MovieLens 数据集公布的训练测试划分，发现测试集中项目的受欢迎程度的分布与随机抽样带来的分布非常不同。然后，作者使用自己的数据划分方式对 MovieLens 数据集进行分割，并且优化了数据分割的参数，以确保公平比较。实验结果如表 7 所示。当使用原始论文中描述的数据分割时，MovieLens 数据集的结果与其他两个数据集的实验结果一致，即在所有配置中，SpectralCF 的性能都比基线方法差，甚至 TopPopular 的表现也比它好。

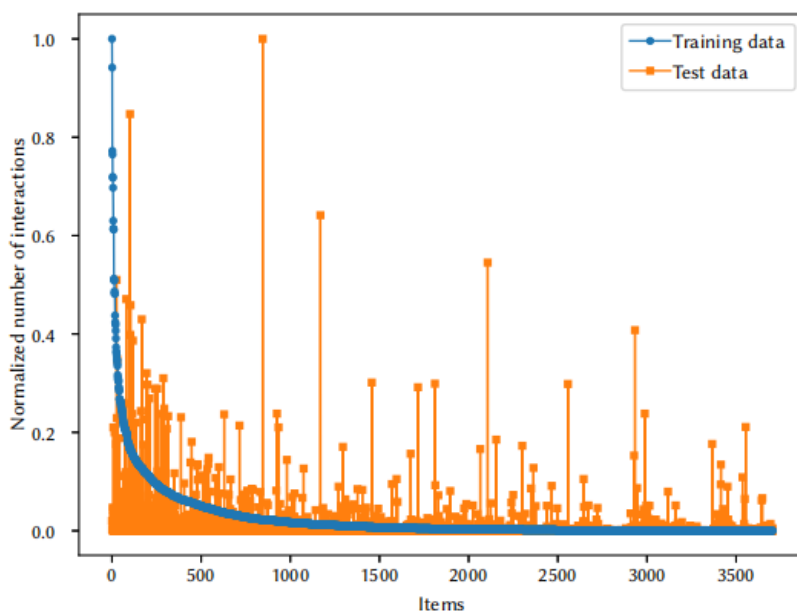


图 1 显示了数据划分问题。蓝色数据点显示训练集中每个项目的归一化受欢迎程度值，最流行的项目的值为 1。在随机划分的情况下，橙色点将非常接近相应的蓝色点。然而，这里测试集中许多项目的受欢迎程度值相差很大。无论是训练还是测试，随机划分的数据集的基尼指数都在 0.79 左右，而所提供的测试集的基尼指数要高得多（0.92），这意味着该分布比随机划分具有更高的受欢迎度偏差。

3.7 变分自动编码器协同过滤（Variational Autoencoders for Collaborative Filtering, Mult-VAE）

Mult-VAE 是一种基于变分自动编码器的隐反馈协同过滤方法。这项工作发表在 WWW18 上。原论文在 3 个二值化数据集上评估该方法，这些数据集包含原始电影评分或歌曲播放计数。实验中采用的基线包括 2008 年的矩阵分解法、2011 年的线性模型和最近的神经网络方法。根据论文，所提出的方法的召回率和 NDCG 结果通常比最佳基线高出 3% 左右。

通过使用它们的代码和数据集，作者发现所提出的方法确实比非常简单的基线技术更好。其准确率比最佳基线高 10% 到 20%。Mult-VAE 是作者经过检查后发现的唯一一个更复杂的方法优于基线技术的方法。

为了验证 Mult-VAE 优于复杂的非神经模型，作者将加权矩阵因子分解方法和线性模型 SLIM 的参数针对数据集 MovieLens 和 Netflix 进行了优化。表 8 显示了在 Netflix 数据集上的实验结果。

	REC@20	REC@50	NDCG@100
TopPop	0.0782	0.1643	0.1570
ItemKNN CF	0.2088	0.3386	0.3086
$P^3\alpha$	0.1977	0.3346	0.2967
$RP^3\beta$	0.2196	0.3560	0.3246
SLIM	0.2551	0.3995	0.3745
Mult-VAE	0.2626	0.4138	0.3756

在 NDCG 评价指标方面，Mult-VAE 和 SLIM 之间的差异非常小。然而，在召

回率方面，与 SLIM 相比，Mult-VAE 的改进似乎是可靠的。作者在不同的截止长度下进行了额外的评估，结果见表 9。表 9 表明，当使用 NDCG 作为优化目标和度量指标时，SLIM 和 Mult-VAE 之间的差异在这个数据集中消失了，SLIM 有时甚至会稍好一些。对于 MovieLens 数据集，也可以观察到类似的现象。因此，在这种特殊情况下，通过神经网络方法获得的进展只是部分的，并且取决于所选择的评价指标。

	NDCG@20	NDCG@50	REC@100	NDCG@100
SLIM	0.2473	0.3196	0.5289	0.3745
Mult-VAE	0.2448	0.3192	0.5476	0.3756

4 讨论

4.1 可复现性和可扩展性

按理说，在应用机器学习领域建立可复现性要比在其他科学和计算机科学的其他子领域容易得多。当研究人员提供他们的代码和使用的数据时，每个人都应该能够或多或少地复现出相同的结果。而且如今的研究人员通常使用公共软件或学术机构提供的软件，因此其他研究人员应该更容易在非常相似的条件下重复实验。

然而，这篇论文表明，算法可复现性的程度实际上并不高。与过去相比，已经有更多的人开始共享核心算法的代码，这可能也是因为可复现性已成为会议论文的评价标准。但是大部分情况下，用于超参数优化、评价、数据预处理和基线的代码是不公开的。这使得其他人很难确认论文报告的结果。

而许多方法的计算复杂性也为复现实验带来了挑战。到 2019 年，已经是 Netflix 发布 1 亿条评分数据集的 10 年之后，研究人员常用的依然是仅包含几十万条评分的数据集。即使对于小数据集，采用 GPU 计算，超参数优化也需要几天甚至几周时间。当然，本文中讨论的基于近邻的方法也存在可扩展性问题。然而，通过适当的数据预处理和数据采样机制，在学术和工业环境中也可以确保这些方法的可扩展性。

4.2 进展评价

最近提出的几种神经网络方法尽管计算复杂，但是其性能却不如在概念上或计算上更简单的方法。因此，至少对于本文所讨论的方法来说，该领域基于深度学习方法的真实进展情况尚不明确。

正如论文所分析的，这种“伪进展”的一个主要原因是基线方法的选择和缺乏对基线方法参数的适当优化。在大多数被研究的方法中，原始论文没有给出足够的基线优化的信息。在有些论文中还发现了数据划分和某些评价标准的实现上存在错误。

另一个有趣的发现是，最近的一些论文使用神经协同过滤方法（NCF）作为其最先进的基线之一。然而，根据作者的分析，这种方法在部分数据集上的表现还不如简单的基线方法。

另一个阻碍评估该领域进展的原因在于研究人员使用的各种数据集、评估协议、度量标准和基线实验。例如，从数据集角度，作者发现了 20 多个公开数据集，以及多个 MovieLens 和 Yelp 数据集的变体，大部分数据集只在一两篇论文中使用。并且研究人员使用了各种度量（精度、召回率、平均精度、NDCG、MRR 等）以及各种评估程序（例如，随机保持 80/20、留一法、每个正项 100 条负项、或 50 项负项）。然而，在大多数情况下，这些选择不合理的。实际上，度量的选择应该取决于应用的环境。例如，在某些应用中，推荐项目的前几项至少需要有一个相关项，这时应该使用基于排序的度量，如 MRR。在其他领域，当目标是向用户显示尽可能多的相关项时，高召回率可能更为重要。除了度量标准的选择不明确之外，这些论文通常也没有解释度量的截止长度，从 top-3、top-5，甚至到几百个元素。

然而，这些现象与基于深度学习的推荐方法无关，在神经网络时代之前也存在这种现象。但是机器学习研究人员对精确度量和寻找“最佳”模型的强烈关注推动了这种发展。在目前的研究实践中，通常认为如果一种新的方法可以在一至两个标准度量上，在一至两个公共数据集上优于现有的一组算法，就已经足够了。然而，使用哪种评估度量和哪些数据集却是任意选择的。

这些现象指出了根本问题，即该领域的研究不受任何假设的指导，也不以解决

给定问题为目标。追求更高的准确度成为了该领域研究的主导方向，但是大家甚至还不清楚准确度的轻微提升是否能够为推荐系统的消费者或提供者带来一定的价值。事实上，许多研究工作表明，更高的准确度并不一定能转化为更好的推荐结果。

5 总结

在这项工作中，作者对各大顶会的最新基于神经网络的推荐算法进行了系统分析。分析表明，已发表论文的可复现程度仍然不高。此外，实验证明，这些基于深度学习的方法均被经典的启发式算法所超越。作者认为，基于神经网络的推荐算法为该领域所带来的实际进展并不明确，作者希望该领域的算法贡献评估能出现更严格和更好的研究实践。

知乎上关于这篇论文也有一些讨论：

如何看待 RecSys2019 上的一篇文章认为现有 DNN-based 推荐算法带来的基本上都是伪提升？

<https://www.zhihu.com/question/336304380/answer/759069150>

查看论文原文：

<https://arxiv.org/abs/1907.06902>



2019.10.17-19 上海·宝华万豪酒店

识别二维码查看大咖实践>>

工程效率

腾讯云上基于 Service Mesh 的后台环境管理实践

黄俊 腾讯 | 高级专项测试工程师

DevOps转型之工程效率提升

王全伦 华为云 | 首席解决方案架构师

用户增长

淘宝用户增长探索与实践

李志勇 阿里巴巴 | 高级技术专家

智能运维

Facebook 大数据模块快速部署和实时更新

冯翼 Facebook | 高级性能架构工程师

架构演进

城市大脑的架构演进

王涵 阿里云 | 技术专家

闲鱼从零到千万 DAU 的应用架构演进

夏朝锋 阿里巴巴 | 技术专家

数据分析

大数据在趣头条的演进: Kafka 读写分离、Hadoop 治理、机器学习平台

虞沐 趣头条 | 大数据部技术总监

Spark SQL 在字节跳动数据仓库领域的优化实践

郭俊 字节跳动 | 数据仓库架构负责人

人工智能

高性能网络通信框架释放 AI 算力的实践

刘一鸽 第四范式 | 基础架构负责人

新零售

NBF: 新零售服务开放的 Serverless 架构与深度实践

冯微峰 (诸葛瑾) 阿里巴巴 | 高级技术专家

编程语言

Scala 和反应式架构

王石冲 字节跳动 | 大数据工程师

音视频

5G 网络下的 8K VR 直播技术解读

张涛 爱奇艺 | 资深工程师