



第一期

A I E C O S Y S T E M



关注落地技术，探寻AI应用场景



创刊序

为什么我们要“**All Around AI**”？

极客邦科技创始人兼 CEO 霍泰稳

在制定极客邦科技今年的内容战略时，我们团队进行了深入的讨论，最终确定了“**All Around AI**”，简而言之，就是我们所有的内容都围绕着“AI”来进行，AI 优先。其实在技术社区，热点话题永远都不少，云计算、大数据风头正劲，物联网也是蒸蒸日上，更不用提一直高烧不退的移动开发和大前端。但我们还是有一万个理由选择“All Around AI”作为战略重点。

前段时间公司开放日的时候，一个同事问我对 AI 在国内发展的看法，我的回答就是“非常乐观”。从技术层面来看，AI 的发展非常依赖良好的运算能力，和海量数据的支撑。而这两点，在互联网普及近 20 年，云计算普及近 5 年后，已经不再成为问题。尤其是电子商务的发展，中国可以说是全球领先，很难在全球其他地方能看到两小时快递送达，半小时生鲜送达等服务。而在电子商务发展的过程中，也让很多互联网公司都积

累了大量用户的数据，可以基于此去分析整理，进而更加了解用户，为用户提供优化后的服务。

另外一个非常考验 AI 是否有大发展的因素就是算法这些基础技术的积累，原来这块的优势在国外在硅谷，但现在很多优秀的算法人才也在回归，回到国内来工作，其原因就是这些技术在国内更能找到应用的场景。最后就是国内的很多公司对 AI 的重视，百度不用说了，口号直接是“*All In AI*”，将自己全部的身家性命搭上了，陆奇的加盟更是如有神助。阿里也发布了自己的 NASA 计划，通过人工智能、机器智能等技术在未来 20 年为 20 亿人提供服务。腾讯也组建了自己的人工智能实验室，将原来百度研究院的副院长张潼博士请过去做负责人，网罗了全球 50 多名顶尖的科学家。

其实 AI 已经不再那么神秘了，在我们日常生活中的很多地方已经在应用了，比如公益领域，通过人脸识别找寻走失儿童，让很多家庭在多年后还能够破镜重圆；在安全方面，通过人脸识别寻找犯罪分子，竟然将一个在逃十多年现在担任快递小哥的罪犯给捉拿归案；在医疗方面，IBM 的 Watson 表现抢眼，对癌症的识别率好像比专业的医生还要高；在家庭生活中，扫地机器人作为典型代表也越来越普及，释放了许多劳动力等等。

如果将未来发展的趋势分为“软趋势”（不确定的，有待验证的趋势）和“硬趋势”（确定的，必将发生的趋势）的话，我是将 AI 划为硬趋势的行列的。也正因为如此，极客邦科技才会选择全力以赴这个领域，才会将今年的内容战略定义为“*All Around AI*”。

具体来说，极客邦科技接下来会主要通过如下的方式促进 AI 技术在国内的普及和发展：

1. 加大在内容方面的投入，结集公司资深编辑挖掘社区、企业和专家的优质资源，通过文字、视频、活动等各种方式发布。
2. 加大在产品方面的投入，通过极客搜索等互联网的产品，让用户能够非常简单方便地搜索到 AI 相关的内容，不论是我们自己创

作的，还是第三方社区创作的。

3. 加大在商务方面的投入，结集公司资深商务人员，深入 AI 领域领先的企业，了解大家的诉求，有针对性地提供服务。

AI 在此，诸神退位。极客邦科技所有在 AI 方面的努力，都会通过我们的微信公众号“AI 前线”第一时间展示给大家，关注就是支持，咱们一起愉快地玩耍。



扫描二维码

关注 AI 前线

AI 生态

InfoQ 中文站 AI 特刊 2017 年 9 月 第 1 期

生态评论

06 吴恩达：人工智能是新电能

18 Michael Jordan：目前人工智能发展到了什么地步？

重磅访谈

26 专访 ImageNet 冠军颜水成团队，如何将比赛成果在企业中落地？

落地实践

37 格灵深瞳：基于人工智能的新天网

54 机器学习与微博：TensorFlow 在微博的大规模应用与实践

70 深度学习在美团点评推荐平台排序中的运用

企业机器学习平台

88 Twitter 机器学习平台的设计与搭建

吴恩达：人工智能是新电能

作者 Andrew Ng, 译者 杨旸



本文译者在印尼 Manado 拍摄，一种体长 8 厘米的海蛞蝓

本文基于 Andrew Ng 在斯坦福 MSx 论坛的演讲 (Artificial Intelligence is the New Electricity)，经演讲人授权，由 InfoQ 中文站总结并分享。

2017 年 2 月，百度首席科学家、Coursera 的联合创始人 Andrew Ng 在斯坦福 MSx 未来论坛上的一个演讲，吸引了全球的眼球。他认为，人工智能 (AI) 对未来许多行业带来的变革，如同 100 多年前，美国“触电”一样——电对制造、运输、农业（尤其是冷藏）、医疗等等带来了划时代的变革。

AI 驱动着百度的搜索和广告，调度百度外卖的快递员，选择路线，

和预估运送时间。AI 正在彻底改变金融工程，对物流的转变进行了一半，医疗和自动驾驶刚开始，而前景巨大。和“电”带来的变革一样，很难想象哪个行业不会被 AI 改变。

监督学习

驱动百亿的市场容量的，基本上属于同一种 AI：监督学习(Supervised learning)，即用 AI 来确定 A→B 的映射——输入 A 和响应 B 的映射。

- 用 Email 作为输入 A，判断是否是垃圾邮件是响应 B。
- 用图像作为输入，识别这是一千种物体中的哪种？
- 从声音 A 到文字 B，从英文到法文，或从文字到声音。

软件可以学习这些输入 A 到响应 B 的映射——有很多好的工具来让机器学习。比如 50,000 小时的音频和对应的文本，就能让机器学到如何从音频内容转化为文本内容。通过大量的电邮数据和区分垃圾的标签，也可以很快地训练出一个垃圾邮件过滤器。

现在的 AI 还很初级——A 到 B 的映射而已，不过已经推动着很大的市场。百度有很好的算法来预测某用户是否会点击某广告。向受众呈现更相关的广告，能为互联网营销和广告公司带来极大的赚钱机会。这可能是 AI 最赚钱的应用。

在哪些产品里能用到 AI？

产品经理常常希望了解 AI 能实现的，和不能实现的。一个简单的思路是：一般人能在一秒内想出来的事情，现在或很快就可以用 AI 自动实现。

AI 进展最快的领域正是人能做得到的领域。比如自动驾驶。人类能驾驶，所以 AI 也能驾驶。在医学影像阅片和分析上，人类放射科医生能够阅片，所以 AI 也很可能在未来几年内做到。

而人类难以做到的事情，比如预测股市变化，AI 可能也难。

- 原因 1：人类能做的，至少是可行的；
- 原因 2：可以利用人类的数据作为培训样本，比如前面提到的输入

A和响应B；

- 原因3：人类能提供指导。如果AI对某个放射影像的结论有误，设计者可以向医生请教，医生所做的正确结论的原因是什么？进而对AI进行改善。

在 Andrew Ng 所接触到的 80–90% 的 AI 项目中，都遵循这一规律：在人类能做到的领域，AI 的进展更快。很多项目的发展一旦超越人类水准，发展也会变得缓慢。这也带来一个社会矛盾：如果 AI 和人的水平类似，实质上是跟人类竞争。

AI 的发展趋势

AI 已经出现了几十年了，而近五年发展明显加速，为什么？

当以前的机器学习算法性能上升到一定程度，即使再增加数据样本量（前文谈到的输入 A、响应 B 的 A-B 映射），性能改善也很有限。似乎超过一定样本量之后，再多的数据也对算法不起作用。

而过去几年，主要由于 GPU，我们终于实现了能利用这么巨大的数据集的机器学习软件。将数据输入一个小的神经网络，当超过一定性能后，上升变得平缓。而不断地把数据输入一个很大的神经网络时，即使性能上升没有那么快，也会保持上升趋势，随着数据量的增大，不断提高。

因此，要想获得很好的 AI 性能，需要两样东西：

- 很大的A-B映射的数据集；
- 大的神经网络。现在常用的大型神经网络建立在HPC高性能计算集群上。

现在的大型 AI 团队包括机器学习和高性能计算两组人，才能获得足够计算能力。百度 AI 团队里的这两种人员都专注于各自领域，没有人能两者兼备。

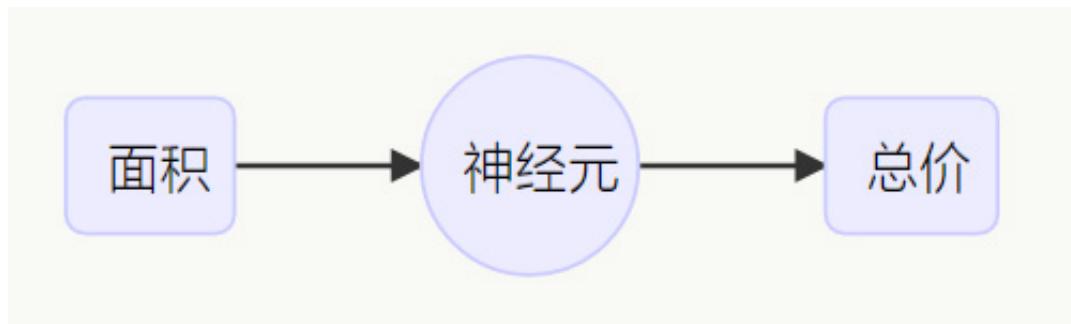
什么是神经网络？有没有可能取代人类大脑？

问题是，我们不清楚人脑如何工作，所以很难造出取代人类大脑的神

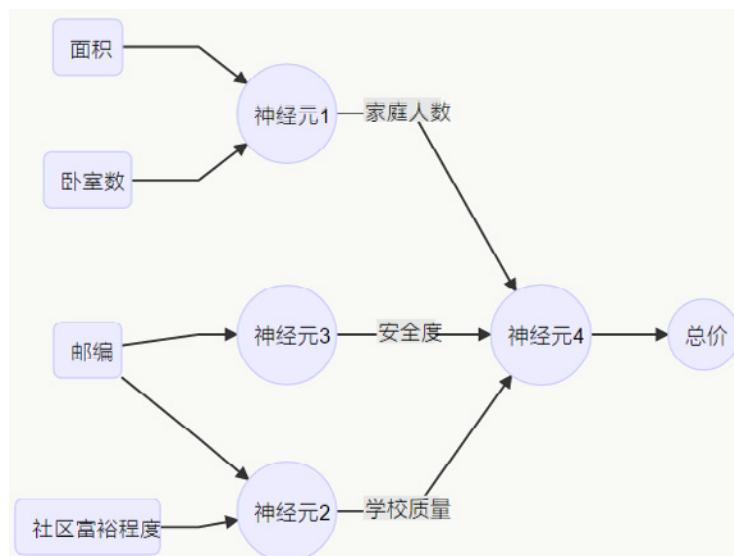
经网络。

什么是神经网络？先看个最简单的神经网络。

如果想输入房屋面积，得到房屋总价，可以用面积 - 总价的一阶近似的线性模型来描述这个神经网络。



或者用更多因素建模，比如通过面积和卧室数，从第一个神经元得到可以支持的家庭人数。再通过所在地址的邮编和社区富裕程度，从第二个神经元得到附近学校的质量。



这就成为一个神经网络。面积、卧室数、邮编、社区富裕程度属于“输入”集合 A，总价属于“响应”集合 B。

好处在于，当训练这样一个神经网络时，用户无需关心中间因素，诸

如家庭人数、安全度、学校质量等，也无需关心每个神经元如何将输入映射到中间结果。只需要给出输入集合 A 和响应集合 B，神经网络将自动形成中间的计算过程和参数。当 A 和 B 的集合足够大，神经网络可以自动算出很多东西。 神经网络看上去非常简单，让很多初学者觉得有点失望，但它确实能解决很多问题。关键在于数据量要够大——几万或几十万个样本本身能提供大量的信息，而软件本身只是一小部分。

如何保护 AI 业务？

AI 研究较前沿的团队都比较开放，常常发布研究成果。百度的 AI 研究论文也没有隐藏什么成果——在人脸识别等论文里，都分享了所有的细节。既然很难把算法本身隐藏起来，如何保护 AI 业务？ 当前稀缺资源有两种，一种是数据，二是人才。获取巨量数据很难，要包括输入 A+ 响应 B。比如语音识别用了 5 万小时的音频来训练，今年准备用 10 万小时，相当于百度 10 年积累的音频。

以人脸识别所用的训练图像数量为例，

- 学术上最常用的基准测试/比赛：1 百万幅；
- 所用图像数最多的计算机视觉对象识别学术论文：1500 万幅；
- 百度用来训练世界上最先进的人脸识别系统：两亿幅！

如果只是 5-10 人的研发团队，很难获得这样规模的数据。百度这样的大企业的经常推出一些新产品不一定是为了营收，而是为了数据，然后通过后续的产品来获得收益。

另一个稀缺资源是人才。AI 的应用需要根据具体业务场景来定制。仅仅下载个开源包，无法解决问题。实际情况下，是否适合用某种垃圾邮件识别或语音识别技术？针对某种场景，机器学习怎么用？ 所以各个公司都在为数据挖掘争夺 AI 人才，来定制 AI 技术，找到所需要的 A 和 B 各自代表什么，怎么找到这些数据和如何调整算法来适应业务场景。

AI 的良性循环

- 先做出某种产品。比如通过语音识别，以语音实现搜索；

- 然后吸引来很多用户，用户产生数据；
- 再通过机器学习，用数据改善产品。

这就形成了 AI 产品的良性循环。最好的产品能获得最多的用户，带来最多的数据，通过现代机器学习体系，能得到最好的 AI，最终让产品变得更好，周而复始。

百度发布新的产品，会特别考虑怎样推动这样的良性循环，会包括相当先进的产品发布策略，比如按地理区域、细分市场等，来更好地推动这个循环。

这种良性循环的理念很早就有了，只是最近变得更加明显。正如前文所述，当数据超过一定规模后，传统 AI 算法无法明显改善 AI 性能，因此数据多的优势不明显，大公司也很难保护自己的 AI 业务。现在数据越多，AI 性能越好，大公司也更容易保护自己的优势。

AI 炒作的非良性循环

许多人担心 AI 会不会取代或威胁人类。有一小部分研究 AI 的人专门从事对“邪恶 AI”的炒作，以获得投资人或政府机构的投资，来研究“反邪恶 AI”。道高一尺，魔高一丈，又进一步推动对“邪恶 AI”的炒作，从而形成非良性循环，非常不健康。

担心 AI 变得邪恶，类似于担心火星的未来人口过剩。现在看不出 AI 将会怎样走偏，因此也谈不上有针对地研究相应措施。研究本身没有问题，不同的研究是好事，但是对邪恶 AI 的研究占用不恰当的资源，就不应该了。两个人，或者 10 个人来研究邪恶 AI 也许没问题，但是现在投资得太多。

AI 对就业的影响

AI 对就业带来的影响更让人担心。有些 AI 项目确实是瞄准了某些人类岗位，而从事这些工作的人并不清楚严重性。硅谷创造了大量财富，但也应该对其造成的问题承担责任，比如造成的失业问题。AI 取代人类岗位的现实问题，更应该引起重视，而不是被邪恶 AI 的炒作转移了注意力。

AI 产品管理

AI 是个让人兴奋的领域，同时也存在一些挑战。如何将 AI 融入公司业务？

产品经理的职责是找到用户喜欢的，而工程师的角色是做出可行的产品。两者共同协作，才能做出理想的产品。

AI 是个新生事物，所以技术公司以前的流程和工作方法，不太适用。硅谷的产品经理和工程师的合作已有一套标准流程。比如开发 APP 时，产品经理先画出线框图，比如 logo，按钮，各个板块等，工程师再写出代码来实现。但是 AI 的 APP 无法通过画线框来描述。通过什么形式，把产品经理头脑里对 AI 产品的功能要求明白地分享给工程师呢？

比如开发语音识别系统，实现语音搜索，有很多改善方向。比如：

- 在嘈杂环境下如何改善，比如车里或咖啡馆；
- 仅改善窄带语音信号；
- 对不同口音改善。

百度发现，产品经理通过数据和工程师沟通，是个较好的办法。产品经理负责提供测试数据集给工程师，比如一万个音频和对应的文字，来说明所关心的问题，工程师也能更明白需要解决的问题。如果这些音频里有大量车辆噪音，工程师就知道车辆噪音是问题。如果是混合了几种不同噪声，工程师也能想办法解决。最糟糕的情况是，产品经理提供的测试数据，并不能代表自己想解决的问题，那就出问题了。

同时，新产品设计的流程有很多，比如想设计一个交流型 AI 机器人：

- 人：“我想叫个外卖”；
- AI：“你喜欢哪种类型餐馆？”
- 人：“川菜”；
- AI：“这些可供选择， xxx, yyy, zzz, ... ”

线框图只能显示对话过程，无法描述所需 AI 的复杂程度等。百度的产品经理和工程师会在一起，写五十种对话。

- 人：“请帮我定一个结婚纪念日的餐馆。”
- AI：“你需要订花吗？”

这时候，工程师会问一些更具体的问题，比如每种场景是否都需要继续提配套产品的问题，比如谈到圣诞节时，是否要问对方要不要买圣诞装饰？一起思考，共同讨论需求和技术，很有效。

对AI的宣传里，有很多吸引眼球的技术，不过它们未必最有用。如何将吸引眼球的技术和产品、业务相结合？软件产业已经有标准流程，比如代码审查、敏捷开发等，如何组织AI的产品工作，有很长的路要走，现在正是考虑这些问题的时候。

短期内，AI有哪些机会？

语音识别正在起飞

最近准确率已经提高到很有用的程度。4-5个月之前，斯坦福大学计算机系教授James Landay、百度、华盛顿大学一起展示了在手机上输入英文和普通话，用语音识别的速度比用手机输入快3倍。去年百度的所有语音识别产品年度环比增长大约100%，现在正是语音识别技术腾飞之时。美国有几个公司做智能语音控制器（Smart Speakers），用语音控制家用设备也会很快推广。相关的操作系统和硬件都会很快发布。

计算机视觉也即将到来

中国的人脸识别发展速度很快。因为中国的手机比笔记本更普及，很多人有手机，而不一定有笔记本。在中国可以仅仅凭手机申请助学贷款。涉及到钱，所以需要先验证身份和很多东西。这加速了人脸识别的发展。通过手机进行人脸识别，作为用生物标识进行身份认证的一种方法，在中国发展很快。

在百度总部，不需要RFID卡进行认证，而是直接刷脸进门，Andrew Ng在YouTube上有一段视频。现在人们对人脸识别技术已经足够信任，并在安全要求较高的场景下使用。

百度在语音识别和计算机识别上的资金投入和数据投入巨大，任何小开发团体远远无法相提并论，也不太可能有其他出乎意料的技术突破。

医疗健康的AI应用

Andrew Ng 对 AI 对医疗健康领域带来的影响很看好。很多现在的放射科医生会被 AI 影响到。如果想在放射科一直工作四十年，不是个好的职业计划。

还有很多垂直领域将受到 AI 的影响，比如金融工程和教育。不过短期之内还不太会对教育产生实质性的影响。

永恒的春天

光从监督学习已经看得出 AI 将如何逐渐改变各个行业。其他的 AI 形式，比如无监督学习、强化学习、迁移学习等等，都还在研究阶段，现在的市场规模较小。

有很多行业会经历几个冬天，然后迎来永恒的春天。AI 经历过两个冬天，现在已经进入永恒的春天。就像硅的春天一样，半导体、晶体管、计算周期这些都将和人类一起发展很久。神经网络和深度学习会繁荣很长时间，一百年或许太远，但一些重要应用改变几个大行业的路线图已经很清晰。

AI 确实正在取代人类的一些岗位。当某些岗位被 AI 取代后，我们需要新的教育系统，来帮助失去工作的人获得新的技能。政府应该为这些愿意学习新技能的人，提供基本收入保障，重新成为劳动者的一员。我们需要新的系统和结构，来让帮助社会向新世界进化。虽然会有新类型的工作，但工作岗位的消失也比以前更快。

一些问题

1. 大公司在数据和人才上有巨大优势，那么创业公司的机会在哪里？投资者可以关注哪种规模的创新？

在语音识别、人脸识别上，小公司非常难与大公司竞争，除非有意料之外的技术突破。同时，也有很多小垂直领域适合创业公司，比如医疗影像。有一些疾病的病例不多，如果有一千张影像，也许就涵盖了所有所需的数据了，一些垂直领域需要的数据量也不大。

另外，AI的机遇非常多，大公司会放弃很多的小的垂直市场，因为精力有限，大的机会还研究不过来。

2. AI 在发明创造上，有哪些进展？

还很早期。AI可以作曲，但这很主观。20年前的技术做出来的曲子有人喜欢，有人不喜欢。有些项目用AI制作图片特效，用特效模仿某画家作品，这些都是小而有趣的领域。现在还看不到有什么技术路线能发明复杂的系统。

3. 如果摩尔定律不再成立，对AI的扩展性有什么影响？

一些高性能计算公司的硬件路线图显示，摩尔定律在单芯片上不再那么有效，但神经网络、深度学习所需的计算类型在未来几年仍然能很好地扩展。SIMD（单指令多数据）让并行化处理负载非常容易。神经网络很容易并行化，加速计算的空间还很大。

AI面对的诸多问题中，许多问题的瓶颈在于数据，也有很多的瓶颈在于计算速度——能便宜地处理数据的速度赶不上获得数据的速度。所以高性能计算的路线图应该包括这方面。

4. 算法是AI里的特殊作料。是否应通过知识产权保护，还是绕过这个问题去设计产品？对机器学习的研究者，是否有和AI产品经理 - 工程师那样类似的流程或良性循环，来实现突破或改善研究流程？

知识产权的问题比较难讲。有些公司申请了大量专利，但是是否真能带来实质性的保护？所以我们往往从如何从战略上思考细节，比如让数据保护自己。

研究机构更偏好新鲜、抢眼球的东西，来发表论文。训练新研究者的办法通常是读很多论文。而大家常常忽视重复论文里的试验的重要性。不

一定要把精力大量用于发明新东西，而花时间重复别人的发布结果也是很好的培训方法。和培训博士生一样：去学习和理解别人的论文，重复别人的试验，争取获得类似的结果，很快你就能产生自己的想法去推动最新的科技。

5. 对希望从事机器人相关工作的机械工程学生，有哪些和 AI、机器人相关的机会比较适合？

很多机械工程背景的人，在 AI 领域很成功。可以上一些计算机 /AI 课程，和 AI 领域的老师聊聊。一些垂直领域存在有趣的 AI 机器人的机会，比如精准农业。Blue River 用计算机视觉来区分不同植物，比如不同品种卷心菜，选择留下哪些，除掉哪些，来提高产量。

中国也生产和销售很多社交和伴侣机器人，美国还没起怎么起步。

6. 让 AI 和人配合起来的前景如何？很多 AI 应用是基于 AI 自己，如果采用 AI+ 人的混合方案？比如自动驾驶等？

没有统一的规则，应该跟实际情况有关。很多语音识别是为了让人类更高效，比如通过手机。对自动驾驶汽车，可能需要 10–15 秒来转换控制权，因为难让容易分神的人快速接手驾驶，很困难。这种情况下，由 AI 独立控制更安全。所以从使用者角度来讲，人类和 AI 混合的自动化比较困难。

7. 对在线教育而言，主要问题是动机，人们不愿意花那么多时间来学完整个课程。这是不是最大的挑战？其他还有什么挑战？

AI 对在线教育有帮助。个性化的辅导已经谈论了很长时间，Coursera 用 AI 推荐个性化的课程，自动打分，在细节上确实有帮助。但在利用 AI 之前，教育的数字化还有很长的路要走。很多行业都有个规律：先有数据，再有 AI，比如医疗，美国电子病历（EHR）的进展很大。随着电子病历的兴起，影像胶片变成数码图片，这些数字化产生了很多数据供 AI 使用，并产生价值。教育需要先经历数字化，这一阶段还有很多工作要做。

8. 百度如何用 AI 来管理自己的云上数据中心？比如 IT 运维管理的

例子？

两年前，百度做了个项目，可以提前一天自动检测出硬件故障，特别是硬盘故障。这就可以事先拷贝、热插拔进行预防处理。还可以降低数据中心的用电量，负载均衡等，都是很多小细节的改善。

9. 能否举一些例子说明能通过仔细地建模和规划，用 AI 解决的复杂问题？对这些问题，人类可能需要进行长时间的思考。

亚马逊是个很好的例子。它知道我浏览过什么，读过什么，比我太太更了解。电脑对人们看过什么，点击过什么广告更了解，所以在广告方面做得非常好。对于有些任务，计算机可以处理的信息量远远超过人类，并根据规律建模，进行预测，这方面 AI 比人做得更好。

将 AI 融入人类工作的很大一部分，是将一块块的 AI 部分串成一个大系统。比如为了造自动驾驶汽车，要用相机拍摄的图像，雷达等，组成车前方的一幅图，再由监督学习估算和其他车的距离，以及和行人的距离，这只是两个重要的 AI 部件，还需要其他的部件来估计 5 秒后车的位置，行人的方向。还有一个部件来分析，根据行人车辆等不同对象的运动情况，我应该怎么走？然后还需要算方向盘的旋转程度，以此类推。

所以复杂的 AI 系统有很多小 AI 部件，工程人员要知道如何将这种超级学习能力融合到更大的系统里，来创造价值。

10. 产品经理和社会学家、律师等如何协调？比如自动驾驶汽车在撞人前，开发者和 AI 应从驾驶者，还是行人的角度考虑问题？这只是个法律问题，但也有很多类似情况。产品管理者和不同的功能部门的合作时，应该扮演什么角色？

这个问题的一个相似版本是“有轨电车”问题，会产生伦理矛盾。一个电车走到岔道口，继续往前会撞死 5 个人，你可以用扳手将电车扳到另一条轨道，撞死该轨道上的一个人，而你成为凶手，你扳吗？

除了在哲学课里，很少有谁在现实生活里遇到过这个问题，所以，它并不重要。自动驾驶的开发者没去讨论它。实际上，如果谁真正遇到了，可能之前已经犯了其他错误了。自动驾驶处理的问题更实际，和你自己开车一样。比如，对面有个白色的大车，是否能及时刹车？

Michael Jordan：目前人工智能发展到了什么地步？

作者 Andrew Ng，编辑 陈思



2017 腾讯“云 + 未来”峰会今日正式在深圳会展中心拉开帷幕。大会以“连接·智能·未来”为主题，云集了近 20 位全球技术前沿专家，共同探讨云端智能，解码未来用人工智能在云端处理大数据的技术应用与发展。

目前，业内越来越多的谈到的一个关键词就是机器学习，而现在也是机器学习领域的一个大突破和爆发的时期，可以这么说：机器学习的发展使得人工智能领域有了飞跃式的进展。大家在谈到人工智能的时候就会特别关注机器学习领域将会以什么层级的速度向未来发展。“云 + 未来”峰会邀请到的顶尖科学家，伯克利教授、人工智能领域的专家 Michael

Jordan先生进行主题演讲，主题为：“机器学习：创新视角直面挑战”。

以下为演讲全文。

大家早上好！今天非常高兴能够来到这里，感谢大家对我的邀请，我是一个研究者，我是做统计学，包括人工智能研究的，今天我非常高兴来到这里，好像大家都非常的了解我，我也非常高兴能够和腾讯合作，能够看一下我们目前对于未来的发展趋势的预见。

首先我们要非常清晰地了解，什么是可能出现的技术，那些是不可能存在的，而哪些是我们现在所存在的问题，以及未来我们会看到什么样的技术的发展，这就是我今天这个演讲的主要所在，以及包括我们未来的洞见和挑战。



行业现状

首先我们简单了解一下到底目前人工智能行业发展是什么样的。

在 60 年代刚刚出现了“智能”这个词，也是刚刚出现了人工智能这个说法，那时候我们说要建立一个机器人，让它可以和人一样思维，加入到人的世界当中来，那个时候大部分人工智能的电影向大家展示的是机器人最终进入到人的世界中，以及包括我们的云系统、视觉系统，还有我们

的自然语言系统，能够让机器人越来越像一个人。

但是在 80 年代、90 年代出现了另外一个趋势，这个趋势对我们来说也是非常重要的，我们叫做 IA，也就是智能增强技术。那时候我们提到搜索引擎，这也是我们的智能方面的应用，通过智能引擎，我们可以非常快的找到我们所要问的任何问题的答案，这些东西不需要存储在人的大脑中，所以人的智能得到了引擎的支持，帮助我们能够更好地用自然语言来进行增强，电脑可以帮助我通过自然语言的处理，增强我的自然语言的表现，所以我可以通过自然和科技的技术，以及智能技术说多种语言。我们可以看到目前这样一个技术正在发展，而机器帮助我们有了更好的存储能力，能有更好的沟通、交流的能力。

还有一个部分是 IaaS，也就是智能基础设施，这对我们来说是最重要的，现在我们的交通和金融行业，在我们身边的每一个行业、每一个模块，现在都出现了智能化的趋势，我们也发现世界更了解我们了，能够根据我们的需求提供服务，所以在我们前方是有一个系统的，如果你要说云的话，这个系统就是云，这个云变得更加智能，所以它并不是机器人和我们沟通，而是这个云的架构和云的基础设施在和我们沟通。

之前我们大部分的研发都是与机器人，以及包括智能领域的发展，它主要是制约与我们的技术发展，它和人的发展是非常相似的，但是智能是完全不同的。现在我们在这种所谓的智能设施的建立的时候，我们遇到了很多问题，在腾讯也是如此。比如说我们要对相关的大型的设施做出相应的决定，比如说我们要做一个金融、交通，以及包括对人类做出一些医疗决定的时候，作为一个单独的机器，如果要能够仅仅跟周围的信息做决策，这是很不好的，有时候机器了解的信息是不够的，一个机器做出的决策往往是不对的，它没办法意识到我们周围环境的变化。特别是这样的决策如果要影响到大部分人，它更是危险的。

从这个角度来说，我们应该如何分享数据呢？我们可以看到机器人的发展，会帮助我们更好地促进业务的发展，我们能够帮助到公司，能够帮助他们进行数据的传输，能够帮助他们得到更多的竞争优势，但是这并不

是最好的说法。如果能把信息进行分享，比如说诈骗信息，每个公司都能看到这方面的危险，它们会不断地集合，然后我们把问题进行更好的解决，同时大家也不会失去竞争力，每个人都可以从中获益。

我们从技术和思维的角度来说还没有办法完全解决，到底我们的知识在哪里？知识是在每个地方不同重复的，在每个地方会出现不重叠性，以及在发展中不同的问题会得到不同的答案，所以我们如何通过数据在多重的维度当中，既包括在云端，也包括在云边缘得到相应的解答，这是很难做到的。同时我们如何做到公正和多样性，现在还很难做到，我们现在只能用一组系统处理一个数据，但是我们还没办法在多种情景下进行部署。还有一个问题是大多数的机器人出现了安全问题，在我们处理了之后，会出现一些系统的攻击，这是比较大的问题，这是我们要重视的，我们不可能说这是小问题，这些都是大问题，这就是我对智能的一个想法。

未来：可能？不可能？

下面回到我们的机器人、智能发展，包括从人工智能的角度来看，我们看看哪些是可能的，哪些是不可能的。



我们看到机器视觉，在过去几年，我们通过摄像机对场景中的物体进行标识，但是它还是没有办法能够像我们清晰的了解到所有的情况，就像我在这里站在台上，大家在台下，我没办法了解到所有人的注意力在哪里，通过人工智能可以帮我们更好地了解语义，但是现在也没办法做到，语音

识别也是如此，我们现在可以把语音转化成文字，文字也可以转换成语音，在各种语言上都可以实现，但是我们的机器人还没办法帮我们了解听觉、视觉之后的真正的意义。

还有一点就是自然语言的处理，我们可以看到到目前为止，自然语言的处理得结果还没有达到我们需要的发展，我们现在有大量的语言的翻译，但是大部分的语言和语句因为没有办法得到有效的语义的阐述，没办法让我们的受众了解到这个语义的意思，有时候我们问问题仅仅能了解部分的答案，而不能了解全部的答案，对机器人学来说也是如此，我们看到世界上有很多工业可编程的机器人，他们也在和我们沟通，但是它们没办法了解到我们的环境、处境以及我们的情绪，我相信这对我们大家来说，如果我们都觉得机器智能将会无处不在的话，这是不太可能的。

对于我们来说，在过去几年的发展，特别是在机器人的发展上，我们的机器人还只是一个雏形，之后可能会出现一些有效的对话，特别是像这样一个自我导识的机器也会出现，但是智能方面它目前还是比较有限的。

我相信在未来，短期内不会出现太多的像人这样的灵活性和可变化性。也许机器可以了解一些事实，它们看上去非常有知识，但是它们没法真正得到人这样一种高级智能，甚至像小孩一样的高级智能，它没有办法了解抽象思维，没有办法进行抽象的处理，机器人还不能实现这方面的能力。这些机器人就像小孩一样，他们知道一些非常棒的现实，他们知道每条河流、每个国家，但是它们仍然没有很高的智能进行人的抗衡，所以在这方面，我们还是很难看到一个超人类的发展，我们相信这个技术可能要很多年的发展才能够出现。我相信我们真正要关注的不仅仅只是这样一种技术的发展，到目前为止，在我们这代人身上还看不到这种高水平的人工智能的出现。

除此之外，我们即使没有办法进行抽象、识别语义，我们也是非常难接近人的发展的，但是我们仍然要进行等待，让我们了解到通过大量的数据的处理，比如说机器人以及人工智能可以帮助我们大批量的处理数据，能够通过数据了解未来一些事件的走向，同时能够保证我们的数据结果不

不断地提高，同时我们还可以用这个机器人做一些简单的人工工作的处理，但是机器人永远不可能像人这样聪明，同时我们可以看到，我们的人工智能的系统也会有很多的智能，它们知道这个现实，但它们不知道哪些现实是真的，哪些是有可能出现未来的一些颠覆式的发展，所以这个机器人并没有办法实现像人一样的能力，它没有办法引领一个公司的发展，在我们这代人身上，在机器上没法做出这样一个前景化的决定。

应该关注什么

我们到底应该关注什么，应该担心什么呢？如果我们担心这种高度类人化的人工智能的发现，我们应该关注，我们所谓的人工智能看上去很智能，但是它并非如此。比如说在医疗行业中，我们让机器做很多的医学诊断，这是不太可能的，有很多人会因为这种不畅的诊断，可能会剂量出现问题，特别是某种竞争的情况下，如果出现任何问题，这个机器没有办法做出有效的诊断，我们的病人都有可能去世。与此同时，我们真正要关注的是机器人可能会造成大量的工作的流失，以及大多数人因为丢了工作没有办法得到收入。

在过去我们可以看到工业的发展，在七八十年代都是如此，但是在过去 50 年中，人们在不断地调整，现在我们可以看到未来 10 到 20 年，人们没有机会更多的调整，机器人会取代更多的人，获得更多的工作。同时它还可以帮助现有的智能设备的发展，在世界上也有很多人会恶意使用人工智能的系统，如果出现人工智能系统的误用，我相信也会有问题。机器人本身是没有任何恶意要伤害人类的，只是使用这些及其人的人本身含有恶意。

在这里我特别要和大家谈到我们近期对于机器学习的一些比较大的挑战，我相信这些挑战都是我们大家已经意识到的，但是现在还没有解决，如果我们能够确保未来要建立起一个人工智能的系统，我们必须解决这些问题，否则没有办法保证未来人工智能的发展。

首先是我们必须要设计一个系统，这个系统可以带来有意义的经过校

准以后的信息，能够应对一些不确定性，比如说在医疗行业，还有在策略规划的角度，如果你是公司的 CEO，你必须要清楚地了解到，一种做法和另外一种做法之间的差别，你不可能只有一个做法。与此同时，我们还要保证我们的系统能够真正地解释它们自己所做出的决策，如果机器做出了一个决定，我们必须要让机器向我们阐释为什么做这样的决定，是否还有其它的潜在方法。还有我们要找到问题发生的原因。

另外我们要找到一个系统，这个系统可以实现长期目标的追溯，同时可以主动的收集在实现目标相关的数据。

还有一点是实时，我们可以看到很多的数据和机器需要花几天、几个小时来学习这些数据，但是到目前为止，我们的机器学习方面还没有办法能够达到真正的实时操作。

还有在意外情况下怎么办，还有在外部事件上的连接，包括数据和其他的要求，需要和政府的合作，和法律部门、和社会科学家的合作。

这是我们所面临的技术挑战，是需要我们关注的，我们只是做 AI，让这个机器人能够跨过去，或者做计算机视觉，我们需要像工程师一样解决一些问题。

在更广范围的挑战，我相信对我们来讲是更难的，比如说在语义方面，在世界上未来会发生什么，我需要了解什么样的概念，我们在机器学习上，我们讲的更多的是表面的东西，我们需要了解真正的世界上需要什么，了解我们所处的情景，但是它们并不知道我们之前发生了什么，它们是了解我们的一些事实，但是它们并不真正知道我们，我们到底什么时候觉得厌烦，我们不想要这样的互动或者交互。

当然还有云端的互动，这也是挑战非常大的，如果把这个数据放到云上，你需要关注隐私的问题，要看一下实施的问题，同时你还要考虑现实的情况，有时候它可能离我们太远，它不一定是和事实一样的，我们有可能会做出错误的决定，所以我们现在要有更好的方案。当然还有一个不确定性，这也是人类的一个非常重要的特点，围棋的比赛其实并不是一个很好的例子，因为你知道棋盘上的东西，但是人的生活有很多不确定性，比

如果说我不知道今天会发生什么事情，我不知道将来会发生什么，这就是所谓人的一生，这和围棋是不一样的，所以我们需要解决更深层次的人工智能方面的问题。

最后总结一下。我很高兴来到这里和大家讲人工智能，我也期待着看大家在 AI 方面会做什么，看看其它企业会做什么，我们需要一起合作，我觉得这相当于 3000 年以前，两个人一起来建立合作，大家去建大桥、建房子，我们觉得很兴奋，我们要带来新的发展，同时也会面临一些灾难，大楼可能会倒闭等等，因为他们当时没有什么科学。



我们一起创建了土建工程，我们一起在世界上进行分享，我们看看周边的建筑物，世界上大家的想法是一样的，因为这是可信任的，它不会再垮塌。但是人工智能还没有，这需要花几十年努力，所以我们需要一起合作，我们要认真考虑怎么解决这些挑战。

专访 ImageNet 冠军颜水成团队，如何将比赛成果在企业中落地？

作者 吴少杰，Tina，陈思



2017 年 7 月 26 日，计算机视觉顶会 CVPR 2017 同期举行的“超越 ILSVRC” Workshop 上，宣布计算机视觉乃至整个人工智能发展史上的里程碑——ImageNet 大规模视觉识别挑战赛于 2017 年正式结束，也就是说 2017 年是 ImageNet 的最后一届。在 2017 年 ImageNet Large Scale Visual Recognition Challenge 2017 (ILSVRC2017) 的收官比赛中，360 公司与新加坡国立大学合作团队拿下了物体定位任务的冠军。InfoQ 因此联系到颜水成团队，进行了这次采访。

ImageNet 竞赛主要分为物体定位（识别）、物体检测、视频物体检测三个大类。在 ImageNet 举行的八年中，物体识别的精度从最初的

71.8% 上升到 97.7%，识别错误率已经远远低于人类的 5.1%。“ImageNet 重新定义了思维模式，虽然很多人关注模型，但 ImageNet 使我们更关注数据，”ImageNet 创始人之一李飞飞说：“数据改变了我们思考的模式。”截至目前，ImageNet 数据集现在超过 1300 万张图片。

受访者介绍

颜水成，360 公司技术副总裁，首席科学家，人工智能研究院院长。在加入 360 之前，在新加坡国立大学做计算机视觉和机器学习的研究，在 2015 年年底加入 360。新加坡国立大学终身教职，IEEE Fellow, IAPR Fellow 及 ACM 杰出科学家。他的主要研究领域是计算机视觉、机器学习与多媒体分析，发表近 500 篇高质量学术论文，论文引用过 3 万次，H-index 74。2014、2015、2016 三次入选全球高引用学者（TR Highly-cited researchers）。

董健，360 高级数据科学家，前 Amazon 研究科学家。目前主要关注深度学习、强化学习、计算机视觉等方面的科学和技术创新，拥有丰富的大数据、计算机视觉经验。曾经多次领队参加 Pascal VOC、ImageNet 等世界著名人工智能竞赛并获得冠军。博士期间在顶级国际学术会议和杂志上发表过多篇学术论文。从 2015 年年底加入 360 至今，董健作为主要技术人员参与并领导了多个计算机视觉和大数据项目。

将 ImageNet 的成功转化到企业实践

ImageNet 从学界发起，但是赛事一经公布，便有多家科技企业参与，包括：谷歌、百度、微软、360 公司、商汤、海康威视等等。对于一个特别是针对计算机视觉的比赛而言，其结果对于学术界的影响力是比较明显的。学术界追求的是算法的极限。就是说研究的目的是能不能通过新的算法，以及一些新的 trick，使性能达到能期望得到的最高的上限，我们叫做追求精度极限。这个比赛，每一年都希望在原来的极限基础上再次提升。对于学术界来说，它的特点非常像一个百米赛跑，每次长度是一样的。比

如像 ImageNet 竞赛，它的数据从 2010 年到现在，训练集和测试集在分类上基本没有变过，在定位方面也没有变过，那么每年的比赛，都是在做同样的事情，每年都在追求一个新的精度极限，每年极限都有一个更新。

对于工业界来说，当性能在往上提升的时候，就会思考这个精度是不是可以转化到工业界的产品或者服务里面。因为对于有一些问题，如果精度没有达到一个 Threshold（阈值），它就可能不会在工业界里去使用。但如果达到了这个域值，就可以开始去思考，怎么通过算法的精简，让它在工业界具体的产品和服务里面直接用起来。

在 360 内部，我们把人工智能设成四个纬度，一是图像的理解，二是语音的理解，三是语义的理解，四是大数据的理解，或者大数据分析。一个很有趣的事情是，每一次计算机视觉新的算法出现之后，在计算机视觉界的成功会引发这些模型在其他领域进行使用。比如说深度学习里的卷积神经网络，原来是在图像领域用的非常不错，后来有人就把它用到语音和语义里面。计算机视觉错误率的降低，决定它的根基的网络结构，会慢慢的从计算机视觉领域往其他的领域扩展，这样会对整个人工智能领域产生一个整体推动的作用。最早的时候，深度学习的使用点并不是在计算机视觉领域，而是在语音识别领域。语音识别领域因为问题相对要少，而计算机视觉领域要处理的问题的类型、数据类型差异化非常大，是个非常庞大的领域，因此计算机视觉在整个人工智能领域的声望比其他领域更强大一些。

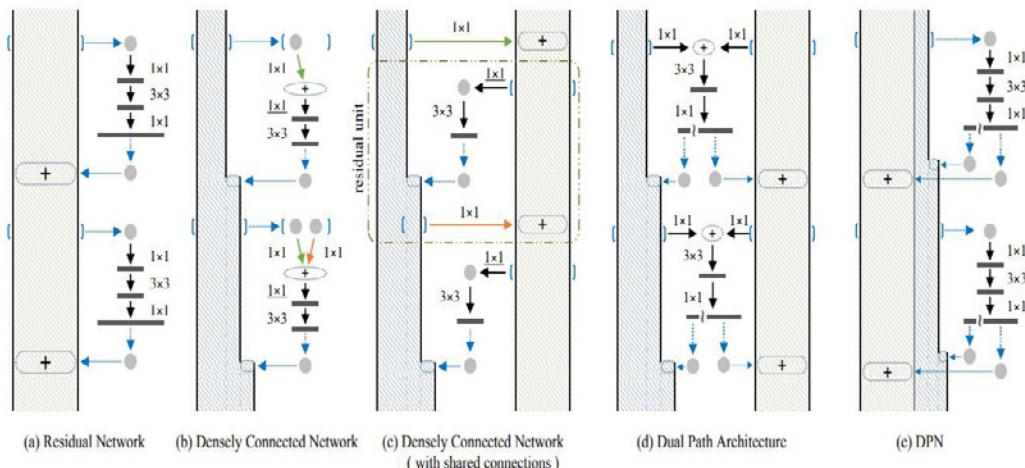
逐年叠加的突破和 DPN 模型的提出

自 2010 年以来，ImageNet 比赛主要包含三类任务，物体定位 (object localization) / 识别 (classification)、物体检测 (object detection)、视频中物体检测 (object detection from video)，还有就是已经取消的物体分割任务。如果只是去看所使用的模型，那么慢慢的大家都非常的相像，都有一个基础模型。这个基础模型一般是从物体分类领域出来的，在这个基础之上，还可以把它拓展成物体的检测和物体的分

割。那么慢慢的在检测和分割上，除了基础模型外后面要叠加的部分，大家都越来越相似，越来越固化，逐渐变成经典。

大家现在的贡献主要是在这个基础模型上的进展。2015年微软拿到比赛冠军的时候，他们也是在基础模型上有很大的一个突破。今年的状况也是非常相象的，主要是在基础模型的一个变化，也就是 DPN 模型的提出。至于怎么做检测，大家用的还是比较标准的框架，可能每年会增加一些特别小的 trick。但是这些 trick 并不是根本性的，根本性的还在基础模型的变化上。

在这次 ImageNet 竞赛中，颜水成团队提出了双通道网络（Dual Path Network）模型。在 ImageNet-1k 分类任务中，DPN 网络不仅提高了准确率，还将 200 层 ResNet 的计算量降低了 57%，将最好的 ResNeXt (64x4d) 的计算量降低了 25%。[论文地址](#)。



这两年有两个非常重要的网络结构，一个是 2016 年的最佳论文，ResNet；一个是 2017 年获得 CVPR 的 Best paper 的 DenseNet。大家在早期研究 DenseNet 的时候，DenseNet 还并没有拿 Best paper。我们的核心成员在研究这两个网络之间的相关性，发现 ResNet 可以解释成 DenseNet 的一个特例，2016 年的 Best paper 是今年的 2017 年的 Best paper 的一个特例。

它的分析是基于我们以前的 Network-in-Network (NiN) 里的 1×1 的卷积。差别是里面有一个 1×1 的卷积，如果在不同的层，用的是同样的参数， DenseNet 就能转化成 ResNet。在这个分析的基础上，我们认为 ResNet 和 DenseNet，虽然是个特例，但是初衷各有优劣。ResNet 不停的去 refine 改善和修正它的特征，但是 DenseNet 相当于不断的去探索新的特征。两种策略都非常重要。举个例子，就是说好像一个公司要往前去发展，一方面你的老员工要不停的自我修炼，要自我提升，这个有点像 ResNet；另一方面，又要不停的去吸收新的血液进来，这些新的成员有新的特性，那么这两部分合在一起，最终的话，能让公司可以稳健的往前走。

基于这种思想，我们就提出一种网络结构，一方面的话，要对原有的的一部分特征进行自我提升，叫做 refinement；另外方面，让一部分网络探索新的特征，然后这两部分特征融合在一起，一层一层的往前走。这地方有一个有趣的问题，有人在问我们，ResNet 既然是 DenseNet 一种特例，那么现在是把这两种网络的特性融合在一起了，那么新的网络是不是一个新的 DenseNet？这个问题很有意思。我和我的学生陈云鹏，想了一个晚上，最后总结，得到的新的网络不能解释成 DenseNet。虽然是把两个网络融合在一起得到一个新的网络，这个网络融合在一起，为什么不是新的 DenseNet？我们举个例子，你有一个等边直角三角形，如 DenseNet 就是一个任意方向的等边直角三角形，ResNet 是某一个特定方向的一个等边直角三角形，两个等边直角三角形拼接好会还是个等边直角三角形，还是 DenseNet。但是我们虽然 ResNet 可以解释成是 DenseNet 一个特例，但是它们两个融合在一起的时候是有一个错位的。类比，两个等边直角三角形如果用长边去做对接，那么出来就是一个正方形，就不再是等边的直角三角形了。DPN 用到了 ResNet 和 DenseNet 的思想，结构拼接的方式比较特殊，所以它出来的网络结构化就已经无法解释成 DenseNet 了。它变成了一种新的网络结构。优势就是充分利用到了修正和改进的特征以及新特征的探索两方面。

ImageNet 的价值和模型关键节点

不只是 ImageNet，还包括做人脸检测和人脸定位，都有各种各样的 benchmark。有一些把测试的样本的标注已经告诉你了，那么就可以不停的不受次数约束地去测试各种算法；还有一种，像 ImageNet 还有一些人脸识别比赛的话，它提供训练数据，校验数据，但是测试数据只提供原始数据，不提供标签。你需要把结果发送到一个服务器上，这个服务器会做结果评价，同时会限制一段时间内能提交的次数，防止不停的去刷参数。如果能不受限制地刷参数，就变得纯粹是一种体力劳动，不是真正意义上的智力活动了。

很多团队，不论是学术界，还是工业界，都希望在这些比赛和 benchmark 上去拿到很好的结果和名次。基于不同的诉求，一种是对他们，比如说融资，有很大的帮助，另一种是为了就是秀肌肉，展现自己的强大实力。我觉得这些都是没有错的。但是刷榜本身是没有价值的，真正的价值在于，刷完榜之后，有没有为这个 community，为所在的领域留下有 Insight 的东西。在 ImageNet 领域，在不同的时间点，留下了很多很不错的基础模型。这些基础模型对于整个领域都是非常有价值的。

最早的一个时间节点是 2012 年，AlexNet 第一次把传统的模型、传统的算法转换到深度学习。2013 年我们参加了 ImageNet 比赛，我们在分类里面取得了第二名。我们当年提出了一个新的模型结构，叫做 Network in Network，实际上我们那年是没有来得及用，但是我们那年在参加 Workshop 的时候，就把 Network in Network 介绍了。2014 年有两个非常重要的模型，一个是 GoogleNet，一个是 VGG Net。GoogleNet 作者把他们的模型叫做 Network in Network in Network，受到 Network in Network 一些启发，特别是大量利用了 Network in Network 中提出的 1x1 卷积。2015 年是微软的 ResNet。2017 年 DPN 的总体效果也挺不错，不少学者在自己领域实验的结果来看具备不错的普适性。

深度学习的低门槛已经能面向普通开发者

深度学习发展到现在，特别在计算机视觉里，大部分东西都已经非常成熟了。甚至可以说现在一个本科一年级学生，他就可以训练出一个一个车辆检测器，或者人脸检测器。也就是说深度学习的门槛不是非常高了。如果说一个企业有一个具体的业务场景，希望得到还不差的性能的话，那么已经没有必要说要拿一个 PHD 去做这件事情了。可以利用开源的平台和一台 GPU 机器，甚至可以运用云计算，把数据放进去，就可以训练出比较不错的模型出来。

那么深度学习的研究，我觉得有两个纬度，一种是追求精度的极限，一种是追求用户体验的极限。追求精度的极限的话，要做的事情是去思考，当前最好的模型在解决具体问题上还有哪些 weakness（弱点）。在这个基础上，再去提出新的模型的结构，在基础模型上去做提升。但是这个非常的困难，为什么？比如说，要想再去超越 ImageNet 比赛最好的性能，所需要的计算机的资源是非常的巨大的，这是第一点。第二点，我觉得对于无论是 ImageNet，还是说人脸识别比赛，通过纯粹算法能提升的精度已经非常少了，基本上都是个位数，或者零点几的提升。那么从这个角度来说，我觉得我们需要有更难的，更有挑战性的数据型集。这也是今年 ImageNet 比赛到了它的收官之战的一个主要原因。

在这个 ImageNet 之前，还有一个比赛，叫做 PASCAL VOC，是欧盟组织的一个比赛，进行了 8 年左右。我们参加了 PASCAL VOC 后三届的比赛。PASCAL VOC 比赛在 2012 年也进入收官之战，我们那年参加了比赛，当时拿了两项比赛的第一。PASCAL VOC 比赛当时进入收官之战的一个主要原因，是因为它的图像标签的数目只有 20 个，跟真实的应用场景有一定的差距。ImageNet 竞赛，现在也要进入收官之战一个主要原因，是因为它的分类的数据集每张图像上只有一个主要物体，跟真实的图片的状况有很大的差距。一般拍的照片不只是一个主要的物体，应该是有很多的 label，是多标签的图像理解。那么 ImageNet 之所以没有弄成非常大

的，多标签的图像分类的一个主要原因，是因为标注量会有大幅的提升。

GAN 和增强学习是深度学习的两个热点

GAN 和增强学习，是当前深度学习的两个热点。增强学习在 Game 领域非常成功，因为它需要有一个 Reward 机制，这个机制在 Game 里面是直接的，从最终的结果里直接能判断出来，但是视觉里面 Reward 的获取不是特别的自然。但是 GAN 的话，对于计算机视觉领域的发展是有基础性的推动作用，特别是对于这种 pixels to pixels 问题，从输入的图像大小，到输出的结果图片的大小是一样的时候。像分割、风格化、超分辨率这些问题，GAN 是非常好的。另外一方面，最近 GAN 用的非常成功的一个地方是用 GAN 去生成一些虚拟图像，这是非常有价值的。我觉得 GAN 还能火几年，还有很多的问题可以做。

计算机视觉技术在 360 的落地

360 最核心的业务是安全。安全一直是整个公司的灵魂和基因，包括 PC 的安全，移动设备上的安全，以及企业的安全。安全产品带来巨大的流量，这个流量就可以直接推动内容和服务的产品，比如 360 导航，360 搜索，360 手机助手，信息流、花椒直播，AR 相机，短视频等等。另外一块，老周从 2012 年开始专注智能硬件，物联网方向。360 专注在三个方面，一个方面是穿戴设备，主要是面对儿童的。如 360 儿童手表，已经第六代了，销量应该是达到了 500 多万台。第二个，家居安全，主要是包括 360 智能摄像头等。第三个是面向出行的安全，主要两款产品，一款产品是 360 的行车记录仪，还有 360 的后视镜。

360 的计算机视觉应用场景，主要是在内容产品和物联网产品。内容上除了支持 360 的图像搜索；还有 360 的短视频，以及花椒直播和相机。用计算机视觉去对图像、视频进行分析、理解，分析出来的内容会作为推荐和搜索的依据。直播和相机则主要是用计算机视觉技术，做美颜、人脸的分析，让拍摄的过程更加好玩，更加有趣。在智能硬件这块的计算机视

觉技术，一方面是人脸的技术，另一方面是人体的技术。人脸的技术主要是在监控摄像头和儿童机器人，使它能够识别家人，识别用户。在出行安全领域，后视镜里面有辅助驾驶的功能，对车辆和车道线进行检测，实现前车碰撞报警，车道偏移报警，前车起动报警。

将人工智能跟信息安全相结合

360 很早就把人工智能和信息安全相结合，行业内第一个具有人工智能技术的杀毒引擎就是 360 的 QVM 引擎，QVM 引擎采用人工智能算法——支持向量机，具备“自学习、自进化”能力，无需频繁升级特征库就能免疫 90% 以上的加壳和变种病毒，从根本上攻克了前两代杀毒引擎“不升级病毒库就杀不了新病毒”的技术难题。现在 360 的安全部门也有一个团队，专门利用深度学习解决很多安全相关的问题。比如说在 2015 年，他们在 Blackhat 大会上曾经有好几篇文章发表了，用深度学习 Data driven 方式去做安全相关的问题。人工智能跟信息安全相结合，肯定是一个趋势。今年 360 会继续去举办互联网安全大会，里面会有一个专门的 keynote speech 讲人工智能和信息安全结合的问题。

计算机视觉未来发展和面临的挑战

对于计算机视觉发展的挑战和前景，团队里的董健回复说，第一个方面，现在很多成功的计算机视觉算法主要是在监督学习领域，需要进行大量的数据标注，但是计算机视觉如果想得到更广泛的应用，还是要考虑更好的利用非监督数据。第二点，现在计算机视觉的算法，在计算资源不受限的环境下，比如有 GPU 集群的情况下，性能是比较有保证的。但我们在很多实际应用中，比如在嵌入式设备中，或者云端部署但用户量非常巨大的情况下，对性能和速度都有非常高的要求。如何将算法进行压缩和加速，也是一个比较大的挑战。当然，这个可能会从软件角度和硬件的角度同时进行推进。

而颜水成对小样本问题给了我们进行了解释。计算机视觉里面，会有

些小样本问题，小样本的问题怎么去解决呢？小样本会有两种情况，一种是样本少，还有一种情况是没有标注数据，但无标注的样本很多。对于第二种，无监督的数据怎么去使用就会非常有价值，就像董健刚才说的，这块会是研究和应用的一个重点。怎么用有两种，一种是在训练模型之前，可以用这种无监督数据做 pre-training，还有一种跟人是非常像的，人对世界当有了一个初步的认识之后，在跟物理世界接触的过程中，在识别时脑海里已有这种识别的模型，在这个模型部署之后，还需要有一种自我学习和自我更新的能力。这种数据没有提供标注，可以认为是一种无监督的方式。

另外，计算机视觉的落地，跟学术界不一样。学术界可以在资源完全不受限的情况下，去做这种事情，去达到精度的极限。但是在工业界，计算机视觉算法的落地，无论是放在云端，还是本地，如嵌入式设备或者手机的设备，降低计算的复杂度，都是有价值的。比如说，假设如果能把人脸识别的计算复杂度降低一半，只要部署一半的资源就可以完成任务，对于公司来说，收益就会有大幅度的提升。除了去追求算法的精度，降低模型计算的复杂度肯定是要重点思考的方向。

深度学习实践建议

实做深度学习图像处理任务时，在实际的应用场景中，Caffe/Caffe2、Tensorflow、PyTorch/Torch 这些工具如何选择，有什么好的建议？

董健：这个问题要看具体的应用场景。如果说的是学术界，或者是偏研究的应用，优先推荐选择 Tensorflow，或者 PyTorch，因为语言是大家比较常用的 Python，开发会比较简单。而且整个网络架构的设计非常灵活，如果想设计一个新的网络结构，代码的开发量会比较小，开发速度会非常快。如果要在实际工业界使用，要看具体的场景，如果是在云端的一个服务，或者说服务的压力不是非常大的话，那像 Tensorflow/mxnet 也是比较合适的，因为它整体部署起来比较容易，效率各方面也能满足要求。如

果在嵌入式平台上开发，因为 Tensorflow 和 PyTorch 相对复杂，速度会更慢一点，而且由于代码量大，导致库的大小也比较大。因此在移动端，像 Caffe/Caffe2 这样比较简单的库更加合适。

对现在各个工具平台的 model zoo，是用 pre-model（预训练模型），还是自我训练模型，有什么建议？

董健：这个问题现在基本上已经达成共识了，就是说如果在这个问题上，你的训练数据已经非常多的话，那你可以直接训练。但是如果你训练数据没有那么多，大家的一个通用做法是，先在一些标准库上，比如 ImageNet，进行预训练，再用预训练模型在具体的问题上进行 fine-tuning。这是一个比较流行也有效的措施。另外，实际使用当中的话，如果你要解决一个实际问题，通常是要根据具体应用设计专门的模型。这个时候，你可能也没有办法直接用网上预训练的模型，但是一般也会进行预训练的操作。

实际生产环境，训练样本偏少，要提高识别准确率，应该在哪些方面多做工作，人工收集样本数据还是通过算法模型标注样本数据？有什么好的建议。

董健：这个也是现在学术界努力解决的一个问题，我们实际在工业使用当中，一个纯 unsupervised 的问题还是比较难解决的，我们更多的是通过算法加速数据标注。具体来说，有点类似于 Active Learning 的思想，先用算法对数据进行一个标注，之后再用人工进行筛查反馈，对数据进行一个修正，目前可能说想完全通过非监督学习来提高精度，难度还是比较大的。

格灵深瞳：基于人工智能的新天网

作者 赵勇，编辑 尚剑



随着平安城市和智慧城市的建设进入智能阶段，公安、交通等部门对于视图大数据的应用产生了强烈需求，他们希望借助大数据分析的能力，进行更快速地案件侦破和更高效的交通管理。格灵深瞳瞄准视图大数据的应用需求，利用海量的数据、先进的深度学习、高性能计算及大数据技术，在产品化的视觉计算处理和数据架构方面进行了相关探索。这次分享将会逐一介绍我们在视图大数据方面所做的一些工作。

视频大数据

今天我们讲的应用对未来的互联网、物联网将会有很大的影响。首先

什么是视频大数据？现在互联网上大致有两类数据，一类是结构化数据，比如说文章、表格里的数字，还有一类数据来自于摄像头、麦克风传进来的图片、视频、语音，这些是非结构化的数据。今天互联网上关于这些数据的处理基本是编解码、直播、转播，比如我发一个视频别人可以看，音乐大家可以听。但是这都不是我们做的，我们做的是对内容的理解，利用人工智能把自然信号转换成结构化的信息放在互联网上来进一步的分析。

有一个美剧叫《疑犯追踪》，讲的是美国有一个神秘的机器，可以把社会上的摄像头和语音通话的内容结构化，把这里面的元数据相关起来形成情报给国家的反恐和犯罪干预提供线索。这种情节在很多美剧里都有，去年还有一部电影《速度与激情 7》也有类似的画面。

这里面无非就是人体检测、跟踪与识别、人脸识别与识别、车辆识别，通过大数据把很多节点连接起来，有了 GIS 的 Service 以后把情报关联起来就形成非常酷的这样一种工具。电视里叫做“上帝之眼”，那么我们离它到底有多远呢？希望讲座之后大家有个清晰的认识。



那么大数据，什么叫“大”？全世界的硬盘公司每生产两块硬盘就有一块硬盘在存储安防、监控的录像，这个录像的数量有多大呢。在北京目前已经有 200 多万个摄像头，它们每一分每一秒都在存储图像，每天都会产生 200 多万天的图像，折合成年的话就是 5000 多年。这些图像已

经产生了，但是我们没有任何能力能去解析、理解这些图像，基本是坏事发生之后我们反过来根据线索去检索阅读里面少量的数据，这个情况以后会发生很大的改变。

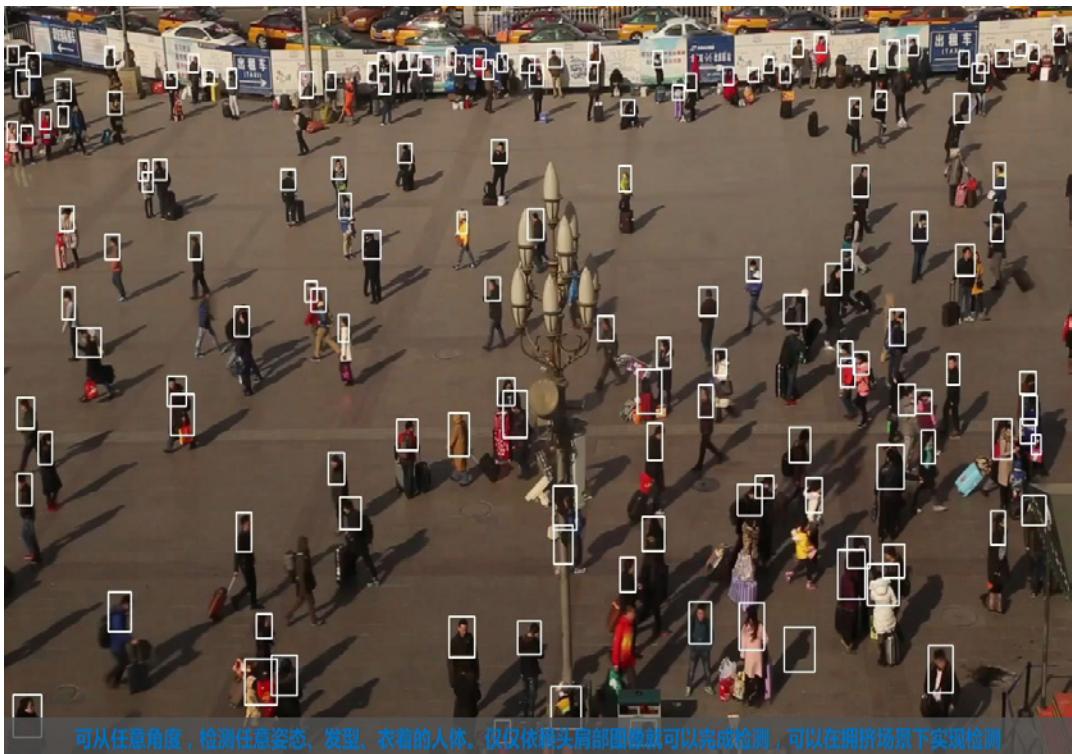
先看一下我们在人工智能方面的一些进展，我们怎么样产生元数据？首先看一下车辆。今天我们的算法可以把一段视频接进来，每一个移动目标，比如汽车，可以看到车牌号码、车型车款、生产年份、车的颜色、基本型号等，分辨率高的话还可以看到驾驶员的状态，有没有系安全带、打电话。



对于人体的目标，这就是一个普通的平安城市的摄像头，走在街头巷尾的时候你可能没有意识到有很多摄像头每天都在拍摄你，我们算法现在可以做到只要你的头部和肩部露出来 1-2s，你一定会被捕捉到，无论穿什么衣服、发型什么样、面对摄像头的角度是什么、长什么样子或者手有没有抬起来、姿态什么样。这是一个近距离的画面，我们来看一个远距离的画面。

这是在北京的火车站的广场上，基本上路过的每个人，只要在画面中

停留超过 1-2s 的时间，都会被我们检测出来。检测出来之后我们会产生一个小小的截图，这个截图会传到后台做二次识别，在二次识别的过程中我们会用很多复杂的人工智能的模型进行分析，可以得到多达几十个标签，比如说年龄、性别、穿衣服的形态、颜色，如果看到正面就包括有没有留胡子、戴口罩、戴眼镜，等等。



人脸识别，在视频里面，如果我们可以得到一个 100×100 以上像素的人脸的话，人脸识别的软件就会抓取人脸的特征放在数据库里进行检索。但是事实上大多数摄像头就算是 1080p，放在任何一个地方，如果它是一个 90 度的指向角，那么 5m 以外就已经达不到这个分辨率了，我们为此专门设计了全新摄像头，可以在 50 米以外可以提供人脸抓取，等效分辨率达到 2 亿像素，比传统高清摄像头高出 100 倍。

三四周以前，我们在办公室园区做了展示，请了科技媒体到我们公司参观，当时我穿着红色 T 恤带着他们参观，其实我们的摄像头在远处偷偷地把每一个人脸都截取下来了。当我们到办公室的时候，把近处的脸捕捉下来，我们的人脸识别系统在后台自动比对每一个人，没有一个人发现

自己的脸被识别错了。我们的数据库现在有 20 万张脸的情况下，可以把每一个人匹配起来。

这是一个网易科技的记者，她贴在自己的朋友圈上，所以我借用一下。其实她当时是站在后面，当时捕捉到的人脸是右边这么一个品质，在 38 米外，但是在室内抓取到的是左边这张脸。

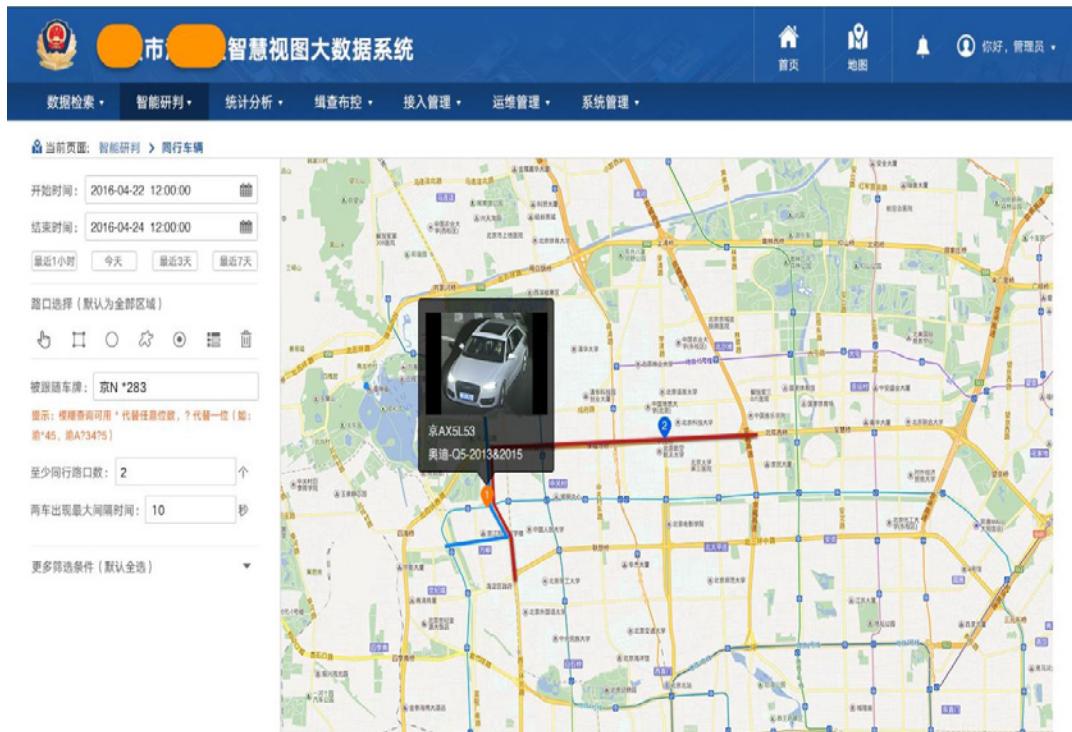


今天人脸识别的精度达到什么程度呢？最近我们在某个省公安厅里面把 3400 万公民身份证照片导进来，有时候警方看到一个嫌疑人，但他不知道是谁，我们比对一下，大概在超过一半的情况下我们搜索结果里的第一名就是他要找的人。今天的人脸识别就到了这样一个程度。

我想讲一下 person of interest 的概念，原意是关键人，你对谁感兴趣。对于我们每个人来说都有自己的 person of interest，对于警方来说，那肯定是逃犯。对于爸爸妈妈来说，就是他们的孩子，对于老师来说就是学生，对于公司来说就是企业的员工和客户。我希望通过我今天的介绍，大家可以想象到，当未来人脸识别技术精确到一定程度以后，大规模运用起来会产生什么样的威力。

业务案例分析

现在做几个简单的业务案例的分析。这里可能的业务非常多，我就缩小到一个例子，比如说车辆追踪。



我们给客户提供了同行车辆分析的功能，主要是发现一些隐藏的团伙作案。有些人发觉自己被追踪了，或者一个案件发生了，怎么发生的，他会觉得之前肯定某些程度上被跟踪过。但怎么样发现谁追踪了你，他们在哪些地方追踪了你。

有一个功能就是你可以把被追踪者的车牌号码输进去，有时候可能是一个模糊搜索，大概是这么一个号码输进去，你可以定一个追踪的标准，比如说至少跟你两个路口，或者跟你间隔小于 10 秒钟之内，我们会在数据库里基于物体被检测的时间、空间的信息，给出来一个轨迹，你就知道这个人跟我这么远，他是谁，车牌号码是什么，这个信息可以帮助警方推测出来可能作案的凶手是谁。警方也会好奇他有没有同犯，是不是在一起

行动的，通过这个也可以给他提供一个精确的线索。

另外一个功能是以图搜图。

为什么要有这个功能？我已经有了车牌号码，是这个车固定的 ID，永远性的，但是对于犯罪分子来说这不是真的，如果是预谋犯罪的话一定会对车进行一定程度的伪装。这个现象非常普遍。不光是犯罪分子想要伪装自己的车牌，其实“良家妇女”也想伪装自己的车牌。比如说北京有个现象叫“克隆车”，你家只有一辆车，你想再给老婆买一辆车，有一个方法就是买一模一样的车套你们自己家的牌，除非查一下引擎号才会比对上，但这个工作量太大了。

怎么去找到这种套牌车呢？万一是犯罪分子套你的牌，干了坏事就栽到你的头上了。

我们先说以图搜图，如果我们不能信任一个车牌的话，光凭这个车的长相怎么把这个车找出来，很多车是大众车型，同一颜色型号的车可能有很多人在开。其实这个车长什么样子不光是这个车的标准长相，包括你的年检标志怎么贴的，这对于一个车来说相当于是一个指纹信息，每个人把



The screenshot shows the Beijing Smart Vision Big Data System interface. At the top, there is a navigation bar with icons for home, map, and user information. Below the navigation bar, there are several menu items: '数据检索' (Data Search), '智能研判' (Intelligent Judgment), '统计分析' (Statistical Analysis), '缉查布控' (Surveillance and Control), '接入管理' (Access Management), '运维管理' (Operation and Maintenance), and '系统管理' (System Management). On the left side, there is a sidebar with a list of recent vehicle detections:

- 京N 6UT45, 海淀区功德寺桥鼎红旗舰店门口, 2016-09-21 10:22:36
- 京N N2479, 海淀区颐和园路国际关系学院, 2016-09-21 10:21:02
- 京A V7626, 海淀区学院路清华科技创业大厦, 2016-09-21 10:20:18
- 渝B CX993, 海淀区学院路航空航天大学东南门, 2016-09-21 10:18:56
- 京N A4L88, 海淀区明杰西路大钟寺中坤广场, 2016-09-21 10:17:22
- 京N 00GY6, 海淀区皂君庙路双柳树南里, 2016-09-21 10:16:45

The main area of the interface displays a map of Beijing with several blue location markers. A specific car is highlighted with a callout box containing its license plate number, '渝BCX993', and model, '奥迪-A4L-2013_2016'. To the right of the map, there is a large inset image showing a close-up of the same silver Audi A4L car from a different angle.

年检标贴到车上的时候都是不一样的，车里有什么挂件，座位有什么装饰，有些时候这些都是关键信息，可以帮助在茫茫车海里把要找的车找出来。

上面这个也是以图搜图通过局部特征找到车牌号码被隐藏起来或者是被套牌的信息。

您输入一辆车，我们在空间里把所有长的像的车全部找出来，然后你可以在这些里面慢慢做一些选择把你感兴趣的找出来。再回到套牌车分析的角度上来，当一个车被套了的时候，除非这两辆车是一直行驶在一起的，不然你们的轨迹一定是不一样的，虽然你可以骗过单个的节点，但你骗不过地理信息，你骗不过时间空间的整合，比如说某一个时间你开车在朝阳区，但是在海淀区又出现了一个同样的车牌，那就很清楚了，一定有一辆是假的，那么当你有轨迹信息可以分析时，你再分析的话也就不难了，你问一下这个车主平时在哪个地方生活的时候就可以知道哪辆车是假的了，当假车再次出现时就会快速被定位了。

轨迹重现就是，当我怀疑一辆车的时候，我可以输入它的车牌号，就会得到在指定时间内的轨迹，经过分析你就可以得到这个人的出行规律、

The screenshot displays the Beijing Smart View Big Data System's vehicle tracking and trajectory reconstruction feature. At the top, there is a header bar with the system's logo, a search bar containing '市' (City), and navigation links for '首页' (Home), '地图' (Map), and a user profile. Below the header, a menu bar includes '数据检索' (Data Search), '智能研判' (Intelligent Judgment), '统计分析' (Statistical Analysis), '缉查布控' (Surveillance and Control), '接入管理' (Access Management), '运维管理' (Operation and Maintenance), and '系统管理' (System Management). The main content area shows a map of Beijing with a red line tracing a vehicle's trajectory. A black rectangular box covers the license plate area of the vehicle. The license plate information is displayed in a callout box: '京AV7626 东风悦达·起亚-2006'. To the left of the map, a list of tracked vehicles is shown:

- 京A V7626, 海淀区巴沟桥西红康路御园路口, 2016-09-21 10:22:36
- 京A V7626, 海淀区颐和园国际关系学院, 2016-09-21 10:21:02
- 京A V7626, 海淀区成府路清华科技创业大厦, 2016-09-21 10:20:18
- 京A V7626, 海淀区学院路航空航天大学东南门, 2016-09-21 10:18:56
- 京A V7626, 海淀区明杰西路大钟寺中坤广场, 2016-09-21 10:17:22
- 京A V7626, 海淀区皂君庙路双榆树南里, 2016-09-21 10:16:45

落脚点，就可以把背后隐藏的犯罪动机找出来。这是一个应用场景。

计算引擎和数据处理架构

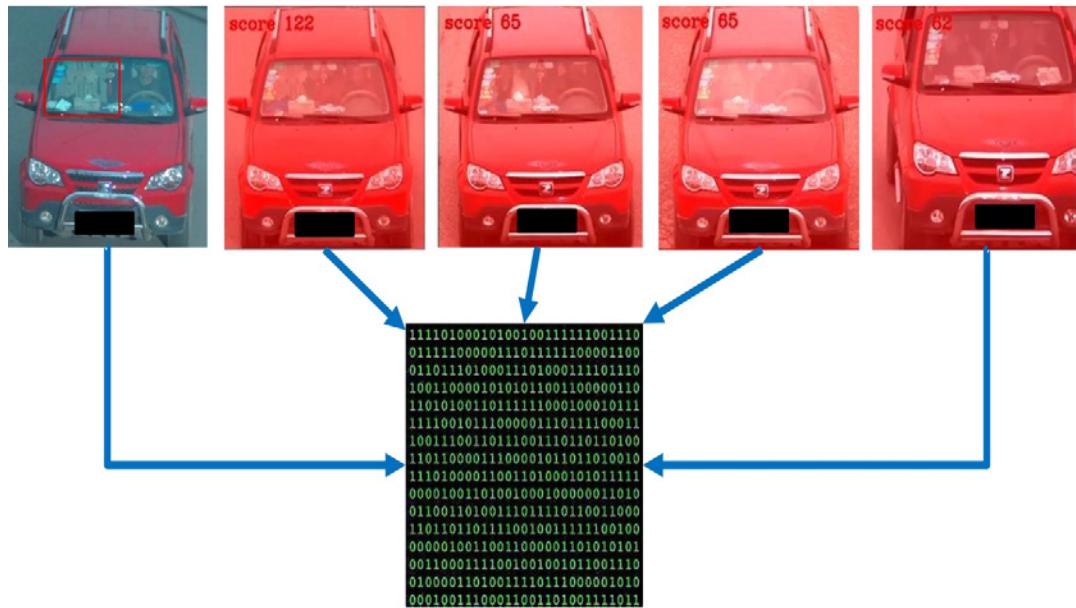
我们看一下背后支持这些应用场景的计算引擎和数据处理的架构是什么样子的。



车牌号	主品牌	型号	型号年份	车身颜色	年检标	细分车型	安全带	副驾驶
京AB1234	奥迪	A6L	2014	黑色	有	小型三厢乘用车	是	无

首先说一下元数据的产生，车辆目标结构化。通过算法生成一些结构化的信息，包括车牌号码、主品牌、型号年份。新款和老款会有细微差别的。颜色、年检标的状态、司机有没有带安全带、副驾驶有没有坐人等。这些标签放到数据库里是直接可以被搜索和分析的。

另外的事情就是视觉特征的向量化，同一个物体，即使是一个汽车，大多数都是长的一样的。但是开车的时候不同的摄像头看到的多少会有不同，角度、光线等等。要搜索它的时候最好把复杂的视觉表达变成一个特征，而这个特征在各种维度都需要具有不变性。特征无非就是一个标准的向量，比如说 128 个字节，如果这个事情通过算法做好的话，最终把视觉的表达变成一个标准向量，比较标准向量就可以了。



人体也是同样的，我们看到每一个人大概会生成年龄、性别、穿着、上半身、下半身、颜色款式、有没有背包、有没有打伞、有没有戴眼镜等等不同的图像，这些都是标准结构化目标。比如说这个人，她穿着红色的衣服和裙子，她路过不同的地方都会产生不同的姿态、不同的视角，当时的动作也不一样。

而且有些摄像头颜色比较灰、有些比较鲜艳，算法就是要把每一个人的表达最终生成一个统一的向量，这个向量是可以比较精确的去比较的，能够把这个个人算出来的。这个过程怎么做？我们用了很多深度学习的算法。模型会把他的每一个身体部分进行解析出来，就算肢体动作不一样，仍然可以得到一个比较稳定的向量特征。

在过去几年进展最快的就是人脸。今天的人脸识别技术当然也有结构化的部分，根据一张脸就可以推测出来他的年龄、性别、有没有留胡子、戴口罩、戴眼镜，等等。

对于人脸的特征向量化就是这样一个概念，对于同一个人，他随着年龄的增长发生了一些变化，这些变化有可能是面部器官上，有的是发型上，不同时候表情也不一样，那我们希望是最终得到一个标准的向量特征，而这个特征是极其稳定的，无论在什么时候都能把这个个人对比出来。

今天的视觉算法可以对任何一个人脸可以把它抽象成一个 512 个字节的特征向量。无论什么时候拍到，向量是很稳定的，在同一个视觉空间的话，同一个人无论做什么表情，最终我们从截图中得到的特征都是你的，它是一个稳定的特征。

数据规模(以车辆大数据为例)



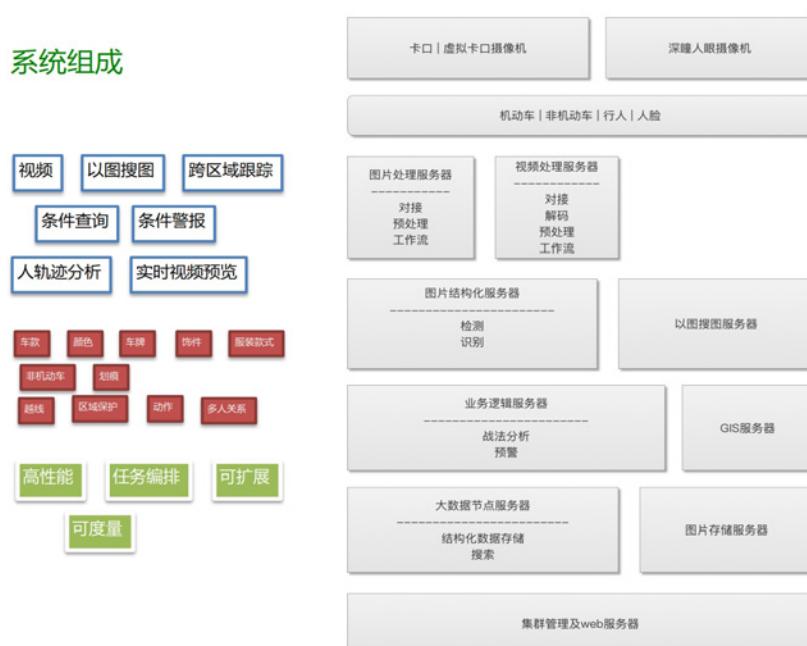
有了摄像头网络，每一天看到每一个目标，进来就生成目标机构化的数据以及视觉特征的向量，这些向量它们的维度并不大，就存到一个数据库里，伴随被存进来的还有摄像头的地点和时间，我们就有了大量的元数据。

车辆大数据的话，以一线城市为例，比如北京或者天津，在城市里面有些摄像头是卡口摄像头，卡口摄像头就是那种架设在龙门架上的摄像头，它的分辨率很高，专门针对一个车道进行捕捉，光学条件非常理想。

北京和天津这样的城市基本有几千个这样的卡口，每天捕捉上亿张照片。我们大概需要 10 到 20 台服务器就可以把所有车辆照片识别出来，天津这样的城市每天会有 3000 多万辆汽车照片被我们识别掉。如果加上视频监控的点位，北京道路有 20 万个摄像头，这 20 万个监控摄像头现在还没有结构化的过程，如果连上我们的服务器的话我们会把这些摄像头里面的数据解析出来，会产生上百亿的图片。在北京不过就几百万辆车，

你可能会被上千个摄像头拍到，一个车会有上千张照片，这样就需要几万台这样的服务器将其转化成可以被大数据挖掘的元数据。

我们的系统组成大概是包括视频、以图搜图、跨区域跟踪、条件查询、条件报警、人轨迹分析、实时视频预览，有各种各样的结构化的信息能够被收集起来。整个体系包括卡口 / 虚拟卡口摄像机、格灵深瞳人眼摄像机，可以得到机动车、非机动车、行为、人脸的结构化的信息，还有它们的视觉向量。



我们有些图片是来自于其他的服务器，这里面有对接和预处理的工作，对于视频来说也有来自于各种型号比较老的各种摄像头，对于图片我们就做一些目标的检测和识别。然后我们有以图搜图的服务器，还有不同的业务逻辑服务器还有 GIS 服务器等，所有这些产生的结果最后都放在大数据的节点服务器上，最后很多应用都建设在这些服务器上。

整个人工智能处理的过程当中数据是非常重要的。

- 首先处理的数据量特别大；
- 第二，深度学习技术，它跟过去的机器学习最大差别是什么？

我个人来看，以前做算法的人差不多每个人都是一個领域的专家，每

个领域产生的自然信号非常熟悉，他设计一套算法把信号转化成特征，这个时候选择一种非常合适的机器学习算法，往往是统计的方法，把这个体系连通起来调解，当你把这个体系设计完成时就有了一个算法的概念在这了。

对于任何一个算法来讲，这个算法设计师闭上眼睛就能想象里面的环节是怎么样工作的。但是这种传统的机器学习方法性能一直不太理想。

直到神经网络算法的出现，神经网络是从哺乳动物神经网络架构里面得到的启示，进而设计了人工的所谓神经网络。每个神经元只做很简单的运算，但是当这个巨型网络建立起来以后，如果你可以做一个很好的训练，数据量足够大，甚至能够将上亿参数的模型训练出来时，它的能力就非常强，在很多领域都超过了人的能力。这种方式的伟大之处就在于，把以前设计算法的过程变成了训练模型的过程，使得机器学习的流程从设计的过程变成训练的过程。

但是要把这个模型训练好的话需要大量的数据，比如说做人脸识别，要是想设计我刚才描述的精度的模型我们可能需要几百万人，每个人需要从他年轻到老，从白天到黑夜，不同表情的数据。这个过程当中数据的获取本身是一个问题，这些数据该怎么标注呢？最开始都是人工的方式去标注，我们公司有一个众包的体系，我们把标注数据放在网络上，每一天都有上千人在不停的标注这个数据，通过审核之后入库重新训练一个新的模型再推出一个产品。

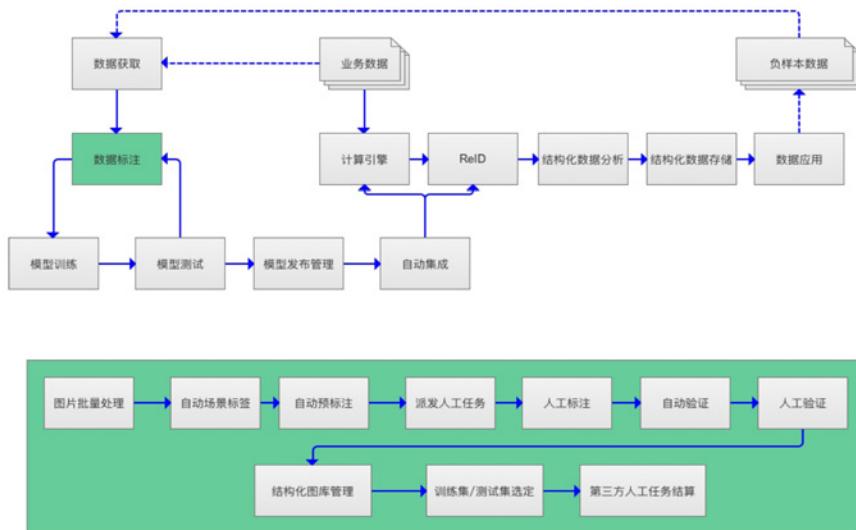
我们看一下数据的环路是什么样的。

首先我们通过某种方式获取数据，进行标注，然后把标注好的数据作为素材去训练模型，这个模型训练之后经过测试如果这个模型比之前的模型好就把这个模型发布出去形成产品，这个产品就以计算引擎的方式出现。

然后业务的数据（客户的各种摄像头的数据）过来，就被计算引擎生成结构化数据、Reld 数据等等。应用过程当中必然会出现错误，而这个错误有时候我们可以感知，有时候不能感知到。有时候我们的算法模型自己就对这个结果不自信，不确定这个结果是对的，有时候产生错误的数据，就

在客户那被提醒出来，这些数据回流。

数据闭环



做新产品的时候所有数据都被标注，第一个版本数据精度是 98%，那么这个产品推向市场的时候就会有 2% 的数据是错的，然后 2% 的数据回来我们又去训练，然后精度就变成 99.8%，这时候就是 0.2% 的数据是错的，依此类推只要业务模型可以流动起来，模型、产品、数据三个模型可以流动起来，以后标注的数据量就越来越少了，并且以前自己的模型会参与到自动标注的过程中。当图片进来的时候首先我们的自动标注系统先产生标注，然后根据结果派发人工的任务，不是很确定的结果由人去标，去验证，验证完之后就进入结构化图片的管理体系，去训练新的模型，产生新的产品，不断的迭代。

什么是计算引擎？所有的原始数据变成结构化的信息，变成一个可以被搜索的向量的流程是什么？

首先我们有任务管理器，有模型的流水线。完成一个任务往往需要很多个模型，比如说人脸识别，我们基本上很少去用一个整体的方式去识别人脸，通过一张人脸进来之后或者一张照片进来之后先做大概的人脸检测，人脸大概在哪个区域，这时候做个细致的人脸检测，你的眉毛、鼻子、嘴

角等在什么地方，大概能产生六七十个定位的点，有了这些点之后，把这些点进行正则化，比如把脸摆正，这时候对眼睛、眉毛、鼻子、嘴巴每个部件分别产生一个深度学习的模型去识别它，最后的分数是根据每个部件识别的结果综合得到的。这就是为什么有的人戴着口罩，还能通过眼睛把你识别出来。

计算引擎

模型：

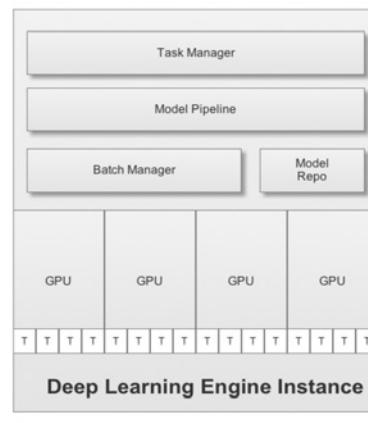
- 检测
- 跟踪
- 识别
- ReID

工程：

- model pipeline
- lifecycle
- model repo
- dynamic load
- compute metric
- better batching

业务逻辑：

- data model
- rpc



所以我们有很多模型就需要有一个流水线把它们组织起来。这里还有一个批处理的概念在这，什么意思呢？一般深度学习网络里面会牵扯到很多参数，有很多层次，每个层次有大量的参数载入进来，一个训练好的人脸识别模型动不动就有上千万的数据在里面。所以如果可以同时识别 10 张、100 张脸，分批量处理的时候，模型的处理效率就高得多。但是这样又有一个 latency 的问题，所以要处理好效率和 latency 的问题，现在基本上大多数的深度学习模型都是在 GPU 上运行的。

同时我们对计算也要进行一些分层，核心计算获取尽可能多的元数据，将业务逻辑后置，我们所有的原始数据进来以后，经过编解码、抽帧、检测、去重，把目标最清晰的一张图片拿到后台去做检测识别，生成它的向量。这些元数据生成结构化信息、向量以后被存储在服务器里，我们所有应用服务都是基于这些数据库在后台去建立的。

计算的性价比，产品本身不是奢侈品，有的时候它最基本的价值反而是能为更多人服务的，所以这个过程当中你必须得做一些取舍。

计算引擎

模型：

- 检测
- 跟踪
- 识别
- ReID

计算的分层：

- 核心计算获取尽可能多的元数据，将业务逻辑后置，灵活可配置
- 故事：黑名单、白名单、关键人筛选

工程：

- model pipeline
- lifecycle
- model repo
- dynamic load
- compute metric
- better batching

计算的性价比：

- 产品不是奢侈品，放弃最贵的
- 故事：骨架识别

业务逻辑：

- data model
- rpc

这里我们有一个故事，是骨架识别。

我们有一个算法识别你的骨架。它有什么意义呢？比如说有的客户提需求，我想知道谁跟人打架了，谁在招手，每一帧图象都进行识别，识别他的全身胳膊腿全身 28 个关节在什么位置，把每一帧都做这种运算，每个人哪个时刻做什么事你就都知道了，基于骨架模型就可以知道他是在招手还是打架。

但这个成本非常高，因为这个运算需要我们对视频中的每一帧每一个人的每个关节去做处理。做出来这个产品之后在市场上推的时候其实它是一个很小众的市场，有这种需求的客户比如说银行（对金库里每一个工作人员进行分析）、监狱（对关键地方的犯人进行分析）。但是我们没有办法把这样的产品推向普罗大众。因为大多数人在马路上，警方关注的问题不是动作不是行为而是身份，而识别出来你的身份不需要每一帧进行识别，找一些关键帧就行了。所以这对我们来说是一个教训，就是针对市场上主要客户的需求提供一些共同的属性出来。

最后介绍一下我们公司，我们公司叫格灵深瞳，是 3 岁半的创业公司，

在北京，现在在颐和园附近。经过了两三年努力之后我们的技术积累基本上进展到了能解决车的问题、人的问题、脸的问题，我们觉得可以对世界上最重要的物体在很高的可靠性下了解他们，这个物体就是人。

今天我们的核心技术使得我们有能力可以为很多客户解决不同问题。我们目前主要客户是国家安全机关，比如反恐、公安这些客户。就像我最开始说的一样，我相信每个人都有自己的 person of interest，比如有美容院、银行联系我们，他们希望他们的 VIP 在走近商店门口的时候就可以先识别出来，问我们能不能提供这样的产品，这是我们下一步要推出的产品。

作者简介

赵勇，博士毕业于美国布朗大学计算机工程系，专攻计算机视觉（Computer Vision）和运算影像学（Computational Photography）。2013年创办北京格灵深瞳信息技术有限公司，目前担任创始人和董事长职位。2016年初，格灵深瞳与前英特尔中国研究院院长吴甘沙、北理工无人驾驶冠军团队负责人姜岩一同成立了专注于无人驾驶的公司驭势科技，赵勇出任董事长。

机器学习与微博：TensorFlow 在微博的大规模应用与实践

作者 何沧平



TensorFlow 在微博业务中有丰富的应用场景，文字、图片、视频，各具特色。微博机器学习平台集成 TensorFlow 服务，支持分布式训练，在广告点击预测应用中，本轮分享的主讲人何沧平积累了一些 TensorFlow 优化经验，在 8 月 3 日晚 AI 前线社群分享活动中，他将自己的这些经验分享给了大家。

TensorFlow 在微博的应用场景

今天的分享内容由虚到实，由概括到具体。

微博的日活和月活数量都在增长，移动端比例很高，占 91%。2017

年 4 月份的财报显示，营收同比增涨 67%，一个重要原因就是移动端抢到了用户的碎片时间。

截止2017年Q1，微博日活跃用户1.54亿，微博月活跃用户3.4亿



微博里随处可见推荐信息：推荐的新闻、推荐的视频、推荐的新账号。最重要还有推荐的广告。

用户登录以后，立刻就要计算推荐什么内容。拿推荐广告来说，备选的广告数以万计，需要排序给出最可能被点击的几条广告。

如果切中用户的购买需要，广告就不再是打扰。



垃圾严重影响用户体验，色情、暴力、反动、低俗，宁可错杀不可漏网，十分严格。

人工智能反垃圾的目标是提高准确度、降低成本。

图像质量也是用户体验的基本要求。



微博反垃圾系统月度识别评论信息超过**6600万条**，准确率达**95%**

升级后，预计月度识别评论信息量超过**1亿条**，准确率可升至**99%**

用户可以容忍不感兴趣的图片，但很难容杂乱的图像。

例如左边的美女图，看起来赏心悦目，手机上刷过，即使不停下细看，也不会反感。

右边的图片，里面也是美女，但加上文字之后，立刻变得杂乱，版式与酒店里的小卡片相仿。很可能被认定为骗子。



明星是微博制胜的法宝。明星是公众人物，话题多、热度高、影响力大。明星粉丝狂热，消费力强。

为粉丝推荐她他喜欢的明星的行程、事件、各种评价，粉丝爱看。甚至明星代言的广告，粉丝可能都会喜欢。停留在微博的时间越长，有意无意浏览的广告就越多。正确识别明星就很重要了，如果不巧推荐了用户讨厌的明星，可能就没了刷微博的心情。

明星脸识别是微博的特色，有海量的明星图片，也有巨大的识别需求。

明星脸识别有特别的困难：常用人脸识别研究所用的照片表情、造型

较少，不同人之间的差别较大。而明星表情丰富，造型多变，无论男女都化妆！不少人妆容近似，有些整容脸连人脑都傻傻分不清，计算机就更难分清了。

明星识别



上部的图片可能归属两个及以上类别，因此称为“兼类”。

图像分类



图片、视频分类的最终目的都是为了关联广告。喜欢旅游的用户就给她他推荐旅游景点、线路、酒店、机票、户外装备等。

如果广告能够切中用户本来就要买的物品，就不必费尽心机说服用户购买不必要的商品，只需要将购买场所由一个地点（网站、实体店）转移到另一个地点，将购买时间由将来转移到现在，将商品品牌由 A 切换为 B。这样广告效果自然会好很多，点击率高，用户还不反感。

例如，印度电影《三个白痴》中几次提到太空笔，我当时就特别想买一支，在京东上搜了半个小时。如果能够提前识别到这个广告点，并在播放过程中推荐购买链接，很可能立即就下单了。



这是一支太空笔
This is an astronaut pen

但是，图像分类难，视频精细分类更难，又不得不分。短视频（5分钟以内）方兴未艾，变现模式还不成熟，处于烧钱阶段。相对于文本、图片，短视频的带宽成本更高，消耗的用户时间更多。如果关联广告的转化率不高，入不敷出，无法长久。

TensorFlow 在微博的应用技术 & 案例

务虚内容结束，下面是具体点的技术。

微博机器学习平台承担了离线训练和在线预测任务。微博实时产生的文本、图片、视频显示后转入后台，用于提取特征、离线训练。

越来越多的业务使用深度学习方法，TensorFlow/Caffe 框架被集成

进来。

离线训练主要使用 GPU 机群。由于业务增长过快，计算机群有一部分来自阿里云。

TensorFlow机群 vs HPC机群



以上完全是个人理解。

对规模巨大的训练任务，TensorFlow 提供了分布式的模式。

TensorFlow 分布式计算与 HPC 的 MPI (Message Passing Interface) 分布计算区别很大。用过 MPI 的人都知道，MPI 进程相互平等，保证没有瓶颈进程。MPI-IO 也设计得每个主机都能均匀分担 IO 压力。MPI 进程上的计算任务也要求均匀划分，保证各个进程的计算进度保持一致。MPI 进程之间也只交换数据块的边界，尽量减少网络流量，压缩通信时间。

TensorFlow 的分布式计算设计得简单粗暴。

若干参数服务器 (parameter server) 和若干劳工 (worker) 组成一个机群 (cluster)，劳工承担运算任务，将每步运算得到的参数 (权重和 bias) 提交给参数服务器，参数服务器将来自所有 worker 的参数合并起

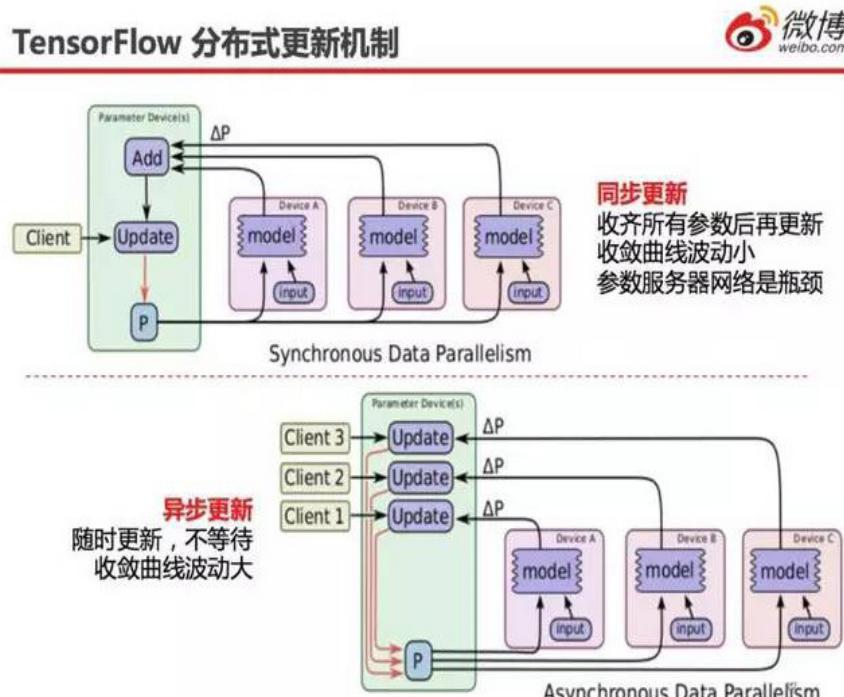
来，得到全局参数，然后将全局参数发送给劳工。劳工在全局参数的基础上继续下一步运算。

TensorFlow 采用主从模式，参数服务器是瓶颈。每步都要传递所有的参数，网络流量太大，假设每个劳工上参数占用内存 1GB，机群包含 1 个参数服务器和 10 个劳工，那么每个迭代步将产生 20GB 的网络流量，按照 10GbE 网络计算，通信时间至少需 16 秒。而实际上，每个 batch 数据的运算时间可能还不足 1 秒，模型参数占用的内存可能远大于 1GB。从理论分析来看，TensorFlow 分布式运算的效率不如 MPI。

有人说深度学习只是高性能计算的一个特殊应用，我认为不是这样。

如图中表格所列，TensorFlow 机群就与 HPC 机群有重大区别。

HPC 机群的 3 大特点：高性能计算芯片（高端 CPU、GPU）、高速网络、并行存储。TensorFlow 机群只需要其中的 1 个：高端 GPU。



劳工在一批数据上训练得到 ΔW 和 Δb （合称为 ΔP ），称为一步训练。

如上图所示，所有的劳工（Device A/B/C）在完成一步训练后，暂停训练，将自己得到的 ΔP 发送到参数服务器（Parameter Device）。参数

服务器一直等待，直到来自所有的劳工的参数变化量 ΔP 都接收成功。参数服务器将所有的 ΔP 相加取平均，然后用这个均值更新旧参数（更新公式请参见随机梯度算法），得到新参数 P ，接着将 P 发送给所有的劳工。劳工在接收到新参数 P 以后，才进行下一步的训练。

与用 1 台服务器训练相比，用 N 台劳工同时训练 + 同步更新参数等价于将 batch 的规模扩大了 N 倍。具体来说，如果用 1 台服务器时，每步训练采用 100 张数字图片 (batch=100)，那么用 4 个劳工得到的参数变化量 (即 ΔP) 同步更新，就相当于每步训练采用 400 张数字图片 (batch=400)。从而，参数变化得更平稳，收敛更快。

同步更新也有缺点：整体速度取决于最慢的那个劳工。如果劳工之间的软硬件配置差别较大，有明显的速度差异，同步更新计算速度较慢。

为了避免劳工有快有慢造成的等待，TensorFlow 提供了异步更新策略。

如图下部所示，当有一个劳工训练得到一个参数变化量 ΔP 时，不妨假设是图中的 Device A，该劳工立即将 ΔP 发送给参数服务器。参数服

异步更新大致正确



```

* cangping -cangping@h10779131:~/simple_dist -expect -f ~/project/wtool/c... cangping -cangping@h10779132:~/simple_d... cangping@h10779132:~/simple_d...
cangping@h10779131:~/simple_dist -expect -f ~/project/wtool/c... cangping@h10779132:~/simple_d... cangping@h10779132:~/simple_d...
localstep= 10092 global_step_value = 10091 Extracting MNIST_data/t10k-images-idx3-ub
localstep= 10093 global_step_value = 10092 Extracting MNIST_data/t10k-labels-idx1-ub
localstep= 10094 global_step_value = 10093 I tensorflow/core/distributed_runtime/mas...
localstep= 10095 global_step_value = 10094 c963c5db0c57398f with config:
localstep= 10096 global_step_value = 10095 allow_soft_placement: true
localstep= 10097 global_step_value = 10096 I tensorflow/core/distributed_runtime/mas...
localstep= 10098 global_step_value = 10097 5a96a68e799cdd8b with config:
localstep= 10099 global_step_value = 10098 allow_soft_placement: true
localstep= 10100 global_step_value = 10099
localstep= 10100 global_step_value = 10099 accuracy = 0.9219
localstep= 10101 global_step_value = 10100 localstep= 1 global_step_value = 10100
localstep= 10102 global_step_value = 10101 localstep= 2 global_step_value = 10276
localstep= 10103 global_step_value = 10102 localstep= 3 global_step_value = 10279
localstep= 10104 global_step_value = 10103 localstep= 4 global_step_value = 10281
localstep= 10105 global_step_value = 10104 localstep= 5 global_step_value = 10283
localstep= 10106 global_step_value = 10105 localstep= 6 global_step_value = 10286
localstep= 10107 global_step_value = 10106 localstep= 7 global_step_value = 10288
localstep= 10108 global_step_value = 10107 localstep= 8 global_step_value = 10291
localstep= 10109 global_step_value = 10108 localstep= 9 global_step_value = 10293
localstep= 10110 global_step_value = 10109 localstep= 10 global_step_value = 10296
localstep= 10111 global_step_value = 10110 localstep= 11 global_step_value = 10298
localstep= 10112 global_step_value = 10111 localstep= 12 global_step_value = 10300
localstep= 10113 global_step_value = 10112 localstep= 13 global_step_value = 10302
localstep= 10114 global_step_value = 10113 localstep= 14 global_step_value = 10305
localstep= 10115 global_step_value = 10114 localstep= 15 global_step_value = 10307
localstep= 10116 global_step_value = 10115 localstep= 16 global_step_value = 10309

```

务器接收到来自劳工 Device A 的 ΔP 后，不等待其它的劳工，立即用 ΔP 更新全局参数，得到全局参数 P，紧接着将 P 发送给劳工 Device A。劳工 Device A 接收到全局参数 P 后，立即开始下一步训练。

由异步更新参数的过程可知，它等价于只用 1 台服务器训练：都是每次用一小批（batch）图像训练更新参数，只是各批数据的上场顺序不能事先确定，上场顺序由劳工的随机运行状态确定。

刚开始运算时，劳工 0（左边）先算了 10100 步（对应 localstep），此后劳工 1（右边）才开始运算。这说明，在异步运算模式下，劳工之间确实不相互等待。劳工 0 和劳工 1 都运算到了全局第 10100 步（global_step_value），说明运算的剖分并不十分准确。

异步更新大致正确



```
cangping --cangping@10779131:~/simple_dist -- expect -f ~/project/wttool/com... cangping@10779132:~/simple_dist -- expect -f ~/project/wttool/com...
cangping@10779131:~/simple_dist -- expect -f ~/project/wttool/com... cangping@10779132:~/simple_dist -- expect -f ~/project/wttool/com...
localstep= 11680 global_step_value = 13091 localstep= 1421 global_step_value = 13107
localstep= 11681 global_step_value = 13093 localstep= 1422 global_step_value = 13109
localstep= 11682 global_step_value = 13095 localstep= 1423 global_step_value = 13111
localstep= 11683 global_step_value = 13097 localstep= 1424 global_step_value = 13113
localstep= 11684 global_step_value = 13099 localstep= 1425 global_step_value = 13115
localstep= 11685 global_step_value = 13101 localstep= 1426 global_step_value = 13117
localstep= 11686 global_step_value = 13103 localstep= 1427 global_step_value = 13119
localstep= 11687 global_step_value = 13105 localstep= 1428 global_step_value = 13121
localstep= 11688 global_step_value = 13107 localstep= 1429 global_step_value = 13123
localstep= 11689 global_step_value = 13109 localstep= 1430 global_step_value = 13125
localstep= 11690 global_step_value = 13111 localstep= 1431 global_step_value = 13127
localstep= 11691 global_step_value = 13113 localstep= 1432 global_step_value = 13129
localstep= 11692 global_step_value = 13115 localstep= 1433 global_step_value = 13131
localstep= 11693 global_step_value = 13117 localstep= 1434 global_step_value = 13133
localstep= 11694 global_step_value = 13119 localstep= 1435 global_step_value = 13134
localstep= 11695 global_step_value = 13121 localstep= 1436 global_step_value = 13135
localstep= 11696 global_step_value = 13123 localstep= 1437 global_step_value = 13136
localstep= 11697 global_step_value = 13125 localstep= 1438 global_step_value = 13137
localstep= 11698 global_step_value = 13127 localstep= 1439 global_step_value = 13138
localstep= 11699 global_step_value = 13129 localstep= 1440 global_step_value = 13139
localstep= 11700 global_step_value = 13131 localstep= 1441 global_step_value = 13140
localstep= 11700 global_step_value = 13131 localstep= 1442 global_step_value = 13141
localstep= 11700 global_step_value = 13131 accuracy = 0.9238 localstep= 1443 global_step_value = 13142
localstep= 11701 global_step_value = 13142 localstep= 1444 global_step_value = 13144
localstep= 11702 global_step_value = 13144 localstep= 1445 global_step_value = 13146
localstep= 11703 global_step_value = 13146 localstep= 1446 global_step_value = 13148
localstep= 11704 global_step_value = 13148 localstep= 1447 global_step_value = 13150
localstep= 11705 global_step_value = 13149 localstep= 1448 global_step_value = 13152
localstep= 11706 global_step_value = 13151 localstep= 1449 global_step_value = 13154
localstep= 11707 global_step_value = 13153
```

14

2 个劳工都执行了第 13142、13144、13146、13148 步，但都没有执行 13143、13145、13147 这 3 步。这说明 Tensorflow 异步更新的任务指派会随机出错，并不是绝对不重不漏。所幸随机梯度法对更新顺序没有要求，少量的错误对最终计算结果影响不大。

同步更新模式不能真正地同步执行，将程序杀死的时候，2 个劳工执

行完的步数相差很多。劳工 0 本地执行了 11023 步之后，全局步数竟然只有 7072，肯定出错了。

同步更新有待修正



```
* cangping -> cangping@h10779131:~/simple* cangping -> cangping@h10779132:~/simple_dist -- expect -f ~/project/
i= 11001 global_step_value = 7050 i= 3103 global_step_value = 7050
i= 11002 global_step_value = 7051 i= 3104 global_step_value = 7051
i= 11003 global_step_value = 7052 i= 3105 global_step_value = 7052
i= 11004 global_step_value = 7053 i= 3106 global_step_value = 7053
i= 11005 global_step_value = 7054 i= 3107 global_step_value = 7054
i= 11006 gl
i= 11007 gl
i= 11008 gl
i= 11009 gl
i= 11010 gl
i= 11011 gl
i= 11012 global_step_value = 7061 i= 3113 global_step_value = 7060
i= 11013 global_step_value = 7062 i= 3114 global_step_value = 7061
i= 11014 global_step_value = 7063 i= 3115 global_step_value = 7062
i= 11015 global_step_value = 7064 i= 3116 global_step_value = 7063
i= 11016 global_step_value = 7065 i= 3117 global_step_value = 7064
i= 11017 global_step_value = 7066 i= 3118 global_step_value = 7065
i= 11018 global_step_value = 7067 i= 3119 global_step_value = 7066
i= 11019 global_step_value = 7068 i= 3120 global_step_value = 7067
i= 11020 global_step_value = 7069 i= 3121 global_step_value = 7068
i= 11021 global_step_value = 7070 i= 3122 global_step_value = 7069
i= 11022 global_step_value = 7071 i= 3123 global_step_value = 7070
i= 11023 global_step_value = 7072 i= 3124 global_step_value = 7071
Traceback (most recent call last):
  File "dist_mnist_sync.py", line 93,
    tf.app.run()
  File "/usr/lib/python2.7/site-packages/tensorflow/python
  in run
      _sy
File ISSUE地址 https://github.com/tensorflow/tensorflow/issues/9596 _ags_passthrough)
      in main
      _, global_step_value = sess.run([
          _ags_passthrough))
      in main
      _, global_step_value = sess.run([
          train_op, global_step
          : batch vs})
      in main
      15
```

网络上也有人报告了这个错误：<https://github.com/tensorflow/tensorflow/tensorflow/issues/9596>，TensorFlow 开发者已经确认这是一个漏洞，但尚未修复。

公式预警……

以 MNIST 手写数字识别为例，上部分公式迭代一步就使用所有 n 个样本。

下部公式将所有样本分割成若干批次（batch）。

TensorFlow 的异步更新，就是不同的劳工使用不同的小批训练样本来更新权重和 bias，不能事先确定每个劳工的更新顺序。具体举例：假设有 2 个劳工执行训练任务，劳工 0 负责更新奇数批次样本 b1/b3/b5…b499，劳工 1 负责更新偶数批次样本 b2/b4, …, b500。

由于各种随机因素，样本的使用顺序可能是 b1àb3àb5àb2àb7àb4à…

异步更新等价于同步更新

1次用所有样本训练

$$C = \frac{1}{n} \sum_{i=1}^n h(w, b; x_i, y_i) + \frac{\lambda}{2n} \|w\|^2$$

$$\Delta w = \frac{\partial C}{\partial w} = \frac{1}{n} \sum_{j=1}^n \frac{\partial h(w, b; x_j, y_j)}{\partial w} + \frac{\lambda}{2n} \frac{\partial \|w\|^2}{\partial w}$$

$$\Delta b = \frac{\partial C}{\partial b} = \frac{1}{n} \sum_{j=1}^n \frac{\partial h(w, b; x_j, y_j)}{\partial b}$$

$$w = w - \eta \Delta w \quad b = b - \eta \Delta b$$

分批次训练 (n=50000, m=100)

$$\Delta w = \frac{\partial C}{\partial w} = \frac{1}{m} \sum_{j=(i-1)m+1}^{im} \frac{\partial h(w, b; x_j, y_j)}{\partial w} + \frac{\lambda}{2m} \frac{\partial \|w\|^2}{\partial w}, i = 1, 2, \dots, \frac{n}{m}$$

$$\Delta b = \frac{\partial C}{\partial b} = \frac{1}{m} \sum_{j=(i-1)m+1}^{im} \frac{\partial h(w, b; x_j, y_j)}{\partial b}, i = 1, 2, \dots, \frac{n}{m}$$

数据混洗(shuffle) → 批次顺序无关紧要 → 异步更新

16

因为样本的批次划分本身就是随机的，这样乱序更新仍然是随机的，对最终结果没有什么影响。

TensorFlow 同步更新时，对所有劳工得到的梯度求平均，然后更新权重和截距。仍然假设有 2 个劳工，它们分别训练第 1 批和第 2 批样本得到梯度 Δw_1 和 Δb_1 截距分别为 Δw_2 和 Δb_2 ，同步之后的梯度如图中所示。

从而，同步更新等价于一次使用 $2m$ 个训练样本，正则化系数和 batch 大小都扩大为原来的 2 倍而已。如果劳工数量很多（例如 20 个），那么同步更新就等价于一次使用 2000 个训练样本，与划分 batch 的初衷不符。因此，建议不要使用同步更新。[注意公式里红色的（2m）](#)

下面是一个具体优化案例。

CTR (Click-Through-Rate, 点击通过率) 是营收的关键。

对候选广告按点击可能性排序，然后插入到用户信息流之中。

deepCTR 不完全是特征工程，输入层与隐层的连接关系也是不全连接。

千亿样本数据近百 TB，为提高效率，采用多人推荐过的 TensorFlow

异步更新等价于同步更新2/2

2个劳工同步更新

$$\begin{aligned}\Delta w &= \frac{1}{2}(\Delta w^1 + \Delta w^2) \\ &= \frac{1}{2} \left(\frac{1}{m} \sum_{j=1}^m \frac{\partial h(w, b; x_j, y_j)}{\partial w} + \frac{\lambda}{2m} \frac{\partial \|w\|}{\partial w} + \frac{1}{m} \sum_{j=m+1}^{2m} \frac{\partial h(w, b; x_j, y_j)}{\partial w} + \frac{\lambda}{2m} \frac{\partial \|w\|}{\partial w} \right) \\ &= \frac{1}{(2m)} \sum_{j=1}^{2m} \frac{\partial h(w, b; x_j, y_j)}{\partial w} + \frac{(2\lambda)}{2(2m)} \frac{\partial \|w\|}{\partial w} \\ \Delta b &= \frac{1}{2}(\Delta b^1 + \Delta b^2) \\ &= \frac{1}{2} \left(\frac{1}{m} \sum_{j=1}^m \frac{\partial h(w, b; x_j, y_j)}{\partial b} + \frac{1}{m} \sum_{j=m+1}^{2m} \frac{\partial h(w, b; x_j, y_j)}{\partial b} \right) \\ &= \frac{1}{(2m)} \sum_{j=1}^{2m} \frac{\partial h(w, b; x_j, y_j)}{\partial b}\end{aligned}$$

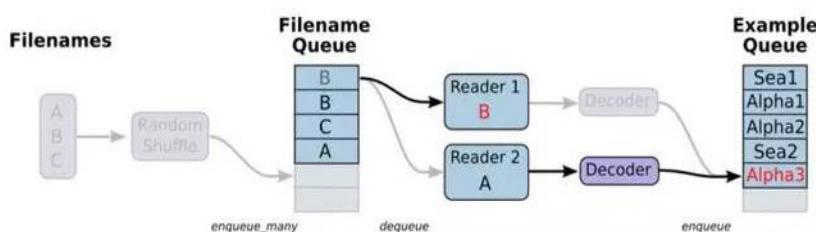
- 等价于一次使用 $2m$ 个训练样本，正则化系数和batch大小都扩大为原来的2倍
- 劳工数量很多（例如200个）时，等价于一次使用2000个样本，与划分batch的初衷不符
- 建议不要使用同步更新。

17

队列。

个人理解，队列的设计初衷很好（如图中表格所示），但实际性能很差，GPU利用率只有5%。查找原因发现，程序卡在线程同步操作上，而这个线程同步就来自于TensorFlow队列。于是尝试用别的方式读取训练样本文件。

TensorFlow 队列读文件



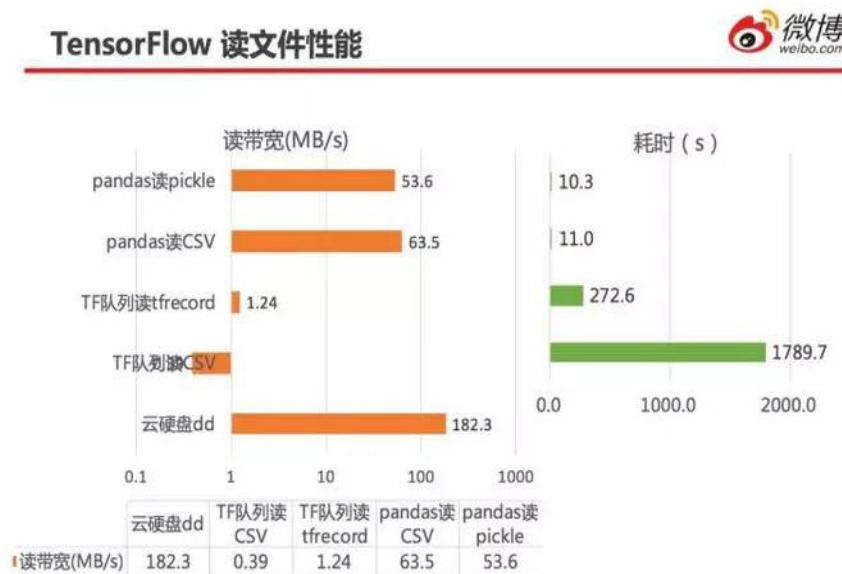
解决的问题

- 单个文件循环读
- 多个文件循环读
- 超大文件不同1次取入内存
- 均匀混洗
- 多个线程同时读取

带来的问题

- 读取性能差，GPU绝大部分时间空闲
- GPU利用率5%

左图横轴采用对数坐标。



队列读以 CSV 带宽只有极限带宽的 $1/467$, 队列读取 tfrecord 格式文件带宽提升至 1.24MB/s , 提高至 3.2 倍。由于 tfrecord 格式文件较小, 读完一个文件的耗时降低至 $15\%(272.6/1789.9)$ 。

用 pandas 读取文件带宽达到极限带宽的 35%。最终舍弃 TensorFlow 队列, 选用 pandas 读 CSV 文件。



当 CSV 文件小于 1/3 内存时，直接用 pandas 一次性读入内存。不用 tf 队列，数据混洗就要程序员自己完成，所幸不麻烦。

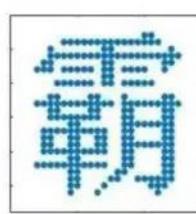
对大于内存 1/3 的文件，直接拆分成多个小文件。需要程序员自行保证均匀使用各个小文件。

最后给各位汇报一个小游戏。

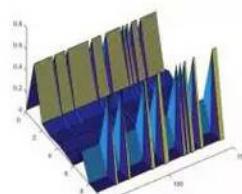
PS：如何判断不认识的类别



MNIST 手写数字
简单 CNN 就能达到 99.2%
的准确度
https://www.tensorflow.org/get_started/mnist/pros



CNN 识别结果：1
0~9 的 softmax 值
[0.0280/0.5445/0.0668/0.03
46/0.0457/0.0318/0.0123/0.
2258/0.0029/0.0077]



字库文件 HZK24S
中第 50-149 这 100
个汉字的分类概率

用 MNIST 训练得到的 CNN 网络来识别汉字，“霸”字被识别为 1。这点很容易理解，得到的 CNN 网络只有 10 个类别，不得不在 0~9 个数字中选一个。

因为“霸”字与任何数字都不像，识别为任何数字的“概率”应该都不太大吧，例如都小于 0.2（随便说的数值）。可是实际情况却是这样：0~9 分类对应的概率差别很大，最大接近 0.8，最小接近 0，卷积网络识别汉字的时候不会犹豫不决，错得十分坚定。

从这个小实验里可以发现几个问题：

图像的特征究竟是什么？如果有，如何用这些特征来区分不认识的图像（比如这个例子里的汉字）？

如何控制一个网络的泛化能力？这个例子中的泛化能力看起来太强了，以致于把汉字都识别成数字了。目前看来，CNN 的泛化能力几乎是听

天由命。

Softmax 后的值真的代表概率吗？看起来它们仅仅是和为 1 正数。概率本质的随机性体现在哪里呢？

这些问题，我还没有想明白，这里提出来，请各位朋友批评指教。

问答环节

问题 1：队列读取性能差是否是由于设置 cache 的样本数问题？

回答：cache 基本没有影响。batch_size 会有影响，最关键还是线程锁的问题。

问题 2：（反垃圾）这一步的准确率怎么算的？是模型准确率吗？

回答：这个涉及到业务，不便透露。可以私下交流。

问题 3：千亿级别 feature 没有模型并行吗？感觉模型单机放不了，不能数据并行。

回答：数据并行，因此研究分布式运算。

问题 4：1 亿条评论的话，你怎么判断分类器是否分正确了？还是说这里的准确率只是在测试集上的准确率？

回答：业务上具体做法不便透露。这里提醒一下，微博有举报、屏蔽功能。

问题 5：微博的 TensorFlow 环境配置，资源管理和分布式计算是用的第三方平台吗？还是自己封装的？

回答：资源管理和分布式计算尝试过几种方案，开源软件 + 自行定制。多种机群，安全级别和管理方式不完全一样，因此资源管理方式（网络、存储、权限）也不一样。

问题 6：会考虑评价 GPU 的利用率吗？比如用 deepbench 测？有什么 GPU 提升利用率的经验分享？

回答：GPU 利用率是成本核算的重要指标，很重视。查看 GPU 利用率比较简单：命令行 nvidia-smi，英伟达还有专门的库，提供轻量级的 C、Java 等接口。

提高 GPU 利用率经验：如果显存能装得下，尽量使用 1 个模型训练；设定显存使用量（例如 0.5），将 2 个及以上作业放在同一个 GPU 上。IO 性能差的话，会导致数据供应不上，从而 GPU 利用低。PPT 中 deepCTR 优化案例就是这个情况。batch 太小、权重矩阵过小，都会导致不能充分利用 GPU 的大量核心（通常有 4000~5000 个），利用率低。

问题 7：如果在庞大的 csv 上训练，用 tf 队列和用 spark df 制作生成器的效果有比对过么？

回答：目前没有对比过 tf 队列和 spark df。

作者简介

何沧平，微博研发中心算法工程师，目前负责建设深度学习平台。对高性能计算（HPC）较熟悉，著有《OpenACC 并行编程实战》一书。

深度学习在美团点评推荐平台排序中的运用

作者 潘晖



美团点评搜索推荐团队借鉴了 Google 在 2016 年提出的 Wide & Deep Learning 的思想，并基于自身业务的一些特点，在大众点评推荐系统上做出了一些思考和尝试，本文是他们的实践经验的总结。

美团点评作为国内最大的生活服务平台，业务种类涉及食、住、行、玩、乐等领域，致力于让大家吃得更好，活得更好，有数亿用户以及丰富的用户行为。随着业务的飞速发展，美团点评的用户和商户数在快速增长。在这样的背景下，通过对推荐算法的优化，可以更好的给用户提供感兴趣的内容，帮用户更快速方便的找到所求。我们目标是根据用户的兴趣及行为，向用户推荐感兴趣的内容，打造一个高精准性、高丰富度且让用户感

到欣喜的推荐系统。为了达到这个目的，我们在不停的尝试将新的算法、新的技术进引入到现有的框架中。

1. 引言

自 2012 年 ImageNet 大赛技惊四座后，深度学习已经成为近年来机器学习和人工智能领域中关注度最高的技术。在深度学习出现之前，人们借助 SIFT、HOG 等算法提取具有良好区分性的特征，再结合 SVM 等机器学习算法进行图像识别。然而 SIFT 这类算法提取的特征是有局限性的，导致当时比赛的最好结果的错误率也在 26% 以上。卷积神经网络（CNN）的首次亮相就将错误率一下由 26% 降低到 15%，同年微软团队发布的论文中显示，通过深度学习可以将 ImageNet 2012 资料集的错误率降到 4.94%。

随后的几年，深度学习在多个应用领域都取得了令人瞩目的进展，如语音识别、图像识别、自然语言处理等。鉴于深度学习的潜力，各大互联网公司也纷纷投入资源开展科研与运用。因为人们意识到，在大数据时代，更加复杂且强大的深度模型，能深刻揭示海量数据里所承载的复杂而丰富的信息，并对未来或未知事件做更精准的预测。

美团点评作为一直致力于站在科技前沿的互联网公司，也在深度学习方面进行了一些探索，其中在自然语言处理领域，我们将深度学习技术应用于文本分析、语义匹配、搜索引擎的排序模型等；在计算机视觉领域，我们将其应用于文字识别、图像分类、图像质量排序等。本文就是笔者所在团队，在借鉴了 Google 在 2016 年提出的 Wide & Deep Learning 的思想上，基于自身业务的一些特点，在大众点评推荐系统上做出的一些思考和取得的实践经验。

2. 点评推荐系统介绍

与大部分的推荐系统不同，美团点评的场景由于自身业务的多样性，使得我们很难准确捕获用户的兴趣点或用户的实时意图。而且我们推荐的

场景也会随着用户兴趣、地点、环境、时间等变化而变化。点评推荐系统主要面临以下几点挑战。

- 业务形态多样性：除了推荐商户外，我们还根据不同的场景，进行实时判断，从而推出不同形态的业务，如团单、酒店、景点、霸王餐等。
- 用户消费场景多样性：用户可以选择在家消费：外卖，到店消费：团单、闪惠，或者差旅消费：预定酒店等。

针对上述问题，我们定制了一套完善的推荐系统框架，包括基于机器学习的多选品召回与排序策略，以及从海量大数据的离线计算到高并发在线服务的推荐引擎。推荐系统的策略主要分为召回和排序两个过程，召回主要负责生成推荐的候选集，排序负责将多个算法策略的结果进行个性化排序。

召回层：我们通过用户行为、场景等进行实时判断，通过多个召回策略召回不同候选集。再对召回的候选集进行融合。候选集融合和过滤层有两个功能，一是提高推荐策略的覆盖度和精度；另外还要承担一定的过滤职责，从产品、运营的角度制定一些人工规则，过滤掉不符合条件的 Item。下面是一些我们常用到的召回策略。

- User-Based 协同过滤：找出与当前 User X 最相似的 N 个 User，并根据 N 个 User 对某 Item 的打分估计 X 对该 Item 的打分。在相似度算法方面，我们采用了 Jaccard Similarity。

$$\text{sim}(x, y) = \frac{r_x \cap r_y}{r_x \cup r_y}$$

r_x, r_y 分别表示用户对Item集合的打分。

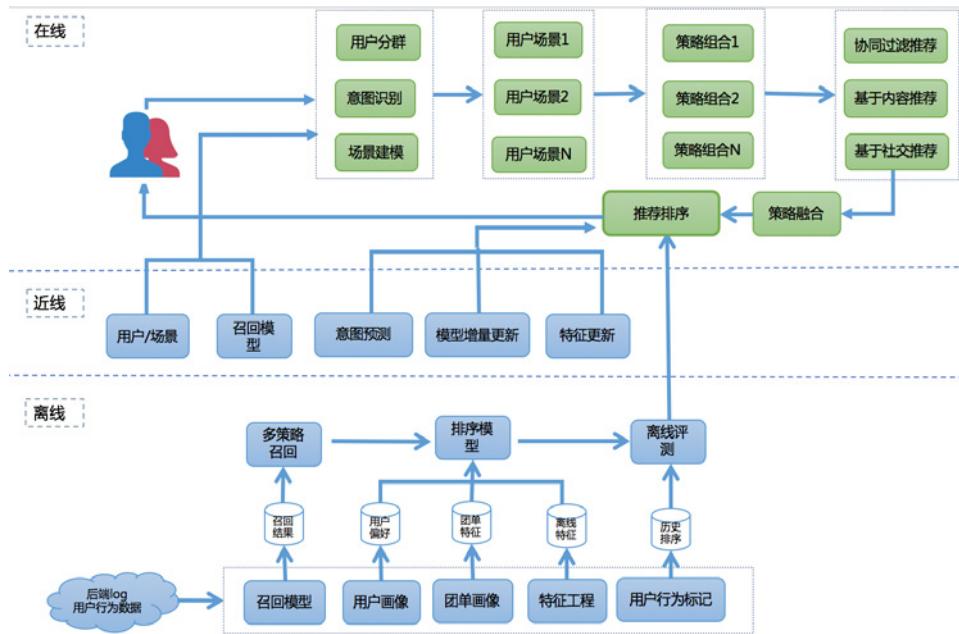
- Model-Based 协同过滤：用一组隐含因子来联系用户和商品。其中每个用户、每个商品都用一个向量来表示，用户 u 对商品 i 的评价通过计算这两个向量的内积得到。算法的关键在于根据已知的用户对商品的行为数据来估计用户和商品的隐因子向量。
- Item-Based 协同过滤：我们先用 word2vec 对每个 Item 取其隐含空间的向量，然后用 Cosine Similarity 计算用户 u 用过的每一个 Item 与未用过 Item i 之间的相似性。最后对 Top N 的结果进行召回。
- Query-Based：是根据 Query 中包含的实时信息（如地理位置信息、WiFi 到店、关键词搜索、导航搜索等）对用户的意图进行抽象，从而触发的策略。
- Location-Based：移动设备的位置是经常发生变化的，不同的地理位置反映了不同的用户场景，可以在具体的业务中充分利用。在推荐的候选集召回中，我们也会根据用户的实时地理位置、工作地、居住地等地理位置触发相应的策略。

排序层：每类召回策略都会召回一定的结果，这些结果去重后需要统一做排序。点评推荐排序的框架大致可以分为三块。

- 离线计算层：离线计算层主要包含了算法集合、算法引擎，负责数据的整合、特征的提取、模型的训练、以及线下的评估。
- 近线实时数据流：主要是对不同的用户流实施订阅、行为预测，并利用各种数据处理工具对原始日志进行清洗，处理成格式化的数据，落地到不同类型的存储系统中，供下游的算法和模型使用。
- 在线实时打分：根据用户所处的场景，提取出相对应的特征，并利用多种机器学习算法，对多策略召回的结果进行融合和打分重排。

具体的推荐流程图如下。

从整体框架的角度看，当用户每次请求时，系统就会将当前请求的数



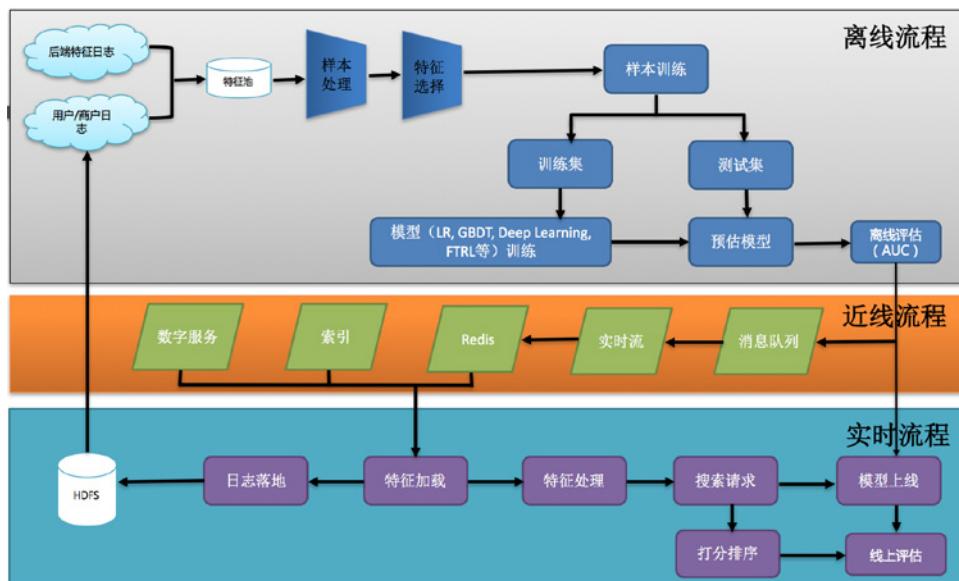
据写入到日志当中，利用各种数据处理工具对原始日志进行清洗，格式化，落地到不同类型的存储系统中。在训练时，我们利用特征工程，从处理过后的数据集中选出训练、测试样本集，并借此进行线下模型的训练和评估。我们采用多种机器学习算法，并通过线下 AUC、NDCG、Precision 等指标来评估他们的表现。线下模型经过训练和评估后，如果在测试集有比较明显的提高，会将其上线进行线上 AB 测试。同时，我们也有多种维度的报表对模型进行数据上的支持。

3. 深度学习在点评推荐排序系统中应用

对于不同召回策略所产生的候选集，如果只是根据算法的历史效果决定算法产生的 Item 的位置显得有些简单粗暴，同时，在每个算法的内部，不同 Item 的顺序也只是简单的由一个或者几个因素决定，这些排序的方法只能用于第一步的初选过程，最终的排序结果需要借助机器学习的方法，使用相关的排序模型，综合多方面的因素来确定。

3.1 现有排序框架介绍

到目前为止，点评推荐排序系统尝试了多种线性、非线性、混合模型



等机器学习方法，如逻辑回归、GBDT、GBDT+LR 等。通过线上实验发现，相较于线性模型，传统的非线性模型如 GBDT，并不一定能在线上 AB 测试环节对 CTR 预估有比较明显的提高。而线性模型如逻辑回归，因为自身非线性表现能力比较弱，无法对真实生活中的非线性场景进行区分，会经常对历史数据中出现过的数据过度记忆。下图就是线性模型根据记忆将一些历史点击过的单子排在前面。



从图中我们可以看到，系统在非常靠前的位置推荐了一些远距离的商户，因为这些商户曾经被用户点过，其本身点击率较高，那么就很容易被系统再次推荐出来。但这种推荐并没有结合当前场景给用户推荐出一些有新颖性的 Item。为了解决这个问题，就需要考虑更多、更复杂的特征，比如组合特征来替代简单的“距离”。

特征。怎么去定义、组合特征，这个过程成本很高，并且更多地依赖于人工经验。

而深度神经网络，可以通过低维密集的特征，学习到以前没出现过的一些 Item 和特征之间的关系，并且相比于线性模型大幅降低了对于特征工程的需求，从而吸引我们进行探索研究。

在实际的运用当中，我们根据 Google 在 2016 年提出的 Wide & Deep Learning 模型，并结合自身业务的需求与特点，将线性模型组件和深度神经网络进行融合，形成了在一个模型中实现记忆和泛化的宽深度学习框架。在接下来的章节中，将会讨论如何进行样本筛选、特征处理、深度学习算法实现等。

3.2 样本的筛选

数据及特征，是整个机器学习中最重要的两个环节，因为其本身就决定了整个模型的上限。点评推荐由于其自身多业务（包含外卖、商户、团购、酒旅等）、多场景（用户到店、用户在家、异地请求等）的特色，导致我们的样本集相比于其他产品更多元化。我们的目标是预测用户的点击行为。有点击的为正样本，无点击的为负样本，同时，在训练时对于购买过的样本进行一定程度的加权。而且，为了防止过拟合 / 欠拟合，我们将正负样本的比例控制在 10%。最后，我们还要对训练样本进行清洗，去除掉 Noise 样本（特征值近似或相同的情况下，分别对应正负两种样本）。

同时，推荐业务作为整个 App 首页核心模块，对于新颖性以及多样性的需求是很高的。在点评推荐系统的实现中，首先要确定应用场景的数据，美团点评的数据可以分为以下几类：

- 用户画像：性别、常驻地、价格偏好、Item 偏好等。
- Item 画像：包含了商户、外卖、团单等多种 Item。其中商户特征包括：商户价格、商户好评数、商户地理位置等。外卖特征包括：外卖平均价格、外卖配送时间、外卖销量等。团单特征包括：团单适用人数、团单访购率等。

- 场景画像： 用户当前所在地、时间、定位附近商圈、基于用户的上下文场景信息等。

3.3 深度学习中的特征处理

机器学习的另一个核心领域就是特征工程，包括数据预处理，特征提取，特征选择等。

- 特征提取： 从原始数据出发构造新的特征的过程。方法包括计算各种简单统计量、主成分分析、无监督聚类，在构造方法确定后，可以将其变成一个自动化的数据处理流程，但是特征构造过程的核心还是手动的。
- 特征选择： 从众多特征中挑选出少许有用特征。与学习目标不相关的特征和冗余特征需要被剔除，如果计算资源不足或者对模型的复杂性有限制的话，还需要选择丢弃一些不重要的特征。特征选择方法常用的有以下几种。

表1:常用的特征选择方法

计算每一个特征与响应变量的相关性	通过计算皮尔逊系数和互信息系数计算相关性，再通过排序选择特征。
构建单个特征的模型	通过模型的准确性为特征排序，借此来选择特征
通过L1正则项来选择特征	因为L1具有稀疏解的特性，因此具备特征选择的特性。同时也可用L2交叉验证。
训练能够对特征打分的预选模型	Random Forest和Logistic Regression等都能对模型的特征打分，通过打分获得相关性后再训练最终模型
通过特征组合后再来选择特征	对用户id和用户特征最组合来获得较大的特征集再来选择特征
通过深度学习来进行特征选择	通过深度学习具有自动学习特征的能力，从深度学习模型中选择某一神经层的特征后，用来进行最终目标模型的训练。

特征选择开销大、特征构造成本高，在推荐业务开展的初期，我们对于这方面的感觉还不强烈。但是随着业务的发展，对点击率预估模型的要求越来越高，特征工程的巨大投入对于效果的提升已经不能满足我们需求，于是我们想寻求一种新的解决办法。

深度学习能自动对输入的低阶特征进行组合、变换，得到高阶特征的特性，也促使我们转向深度学习进行探索。深度学习“自动提取特征”的优点，在不同的领域有着不同的表现。例如对于图像处理，像素点可以作

为低阶特征输入，通过卷积层自动得到的高阶特征有比较好的效果。在自然语言处理方面，有些语义并不来自数据，而是来自人们的先验知识，利用先验知识构造的特征是很有帮助的。

因此，我们希望借助于深度学习来节约特征工程中的巨大投入，更多地让点击率预估模型和各辅助模型自动完成特征构造和特征选择的工作，并始终和业务目标保持一致。下面是一些我们在深度学习中用到的特征处理方式：

3.3.1 组合特征

对于特征的处理，我们沿用了目前业内通用的办法，比如归一化、标准化、离散化等。但值得一提的是，我们将很多组合特征引入到模型训练中。因为不同特征之间的组合是非常有效的，并有很好的可解释性，比如我们将“商户是否在用户常驻地”、“用户是否在常驻地”以及“商户与用户当前距离”进行组合，再将数据进行离散化，通过组合特征，我们可以很好的抓住离散特征中的内在联系，为线性模型增加更多的非线性表述。组合特征的定义为：

$$\phi_k(x) = \prod x_i^{c_{ki}}, c_{ki} \in (0, 1)$$

3.3.2 归一化

归一化是依照特征矩阵的行处理数据，其目的在于样本向量在点乘运算或其他核函数计算相似性时，拥有统一的标准，也就是说都转化为“单位向量”。在实际工程中，我们运用了两种归一化方法：

Min-Max。

Min 是这个特征的最小值，Max 是这个特征的最大值。

Cumulative Distribution Function (CDF)：CDF 也称为累积分布函数，数学意义是表示随机变量小于或等于其某一个取值 x 的概率。其

$$x' = \frac{x - \min}{\max - \min}$$

公式为：

$$x' = \int_{-\infty}^x f(x) dx$$

在我们线下实验中，连续特征在经过 CDF 的处理后，相比于 Min-Max，CDF 的线下 AUC 提高不足 0.1%。我们猜想是因为有些连续特征并不满足在 (0, 1) 上均匀分布的随机函数，CDF 在这种情况下，不如 Min-Max 来的直观有效，所以我们在线上采用了 Min-Max 方法。

3.3.3 快速聚合

为了让模型更快的聚合，并且赋予网络更好的表现形式，我们对原始的每一个连续特征设置了它的 super-liner 和 sub-liner，即对于每个特征 x ，衍生出 2 个子特征：

$$x^2$$

$$\sqrt{x}$$

实验结果表示，通过对每一个连续变量引入 2 个子特征，会提高线下 AUC 的表现，但考虑到线上计算量的问题，并没有在线上实验中添加这 2 个子特征。

3.4 优化器 (Optimizer) 的选择

在深度学习中，选择合适的优化器不仅会加速整个神经网络训练过程，并且会避免在训练的过程中困到鞍点。文中会结合自己的使用情况，对使

用过的优化器提出一些自己的理解。

3.4.1 Stochastic Gradient Descent (SGD)

SGD 是一种常见的优化方法，即每次迭代计算 Mini-Batch 的梯度，然后对参数进行更新。其公式为：

$$\nu_t = \mu \nabla_{\theta} J(\theta)$$

缺点是对于损失方程有比较严重的振荡，并且容易收敛到局部最小值。

3.4.2 Momentum

为了克服 SGD 振荡比较严重的问题，Momentum 将物理中的动量概念引入到 SGD 当中，通过积累之前的动量来替代梯度。即：

$$\nu_t = \gamma \nu_{t-1} + \mu \nabla_{\theta} J(\theta)$$

$$\theta = \theta - \nu_t$$

相较于 SGD，Momentum 就相当于在从山坡上不停的向下走，当没有阻力的话，它的动量会越来越大，但是如果遇到了阻力，速度就会变小。也就是说，在训练的时候，在梯度方向不变的维度上，训练速度变快，梯度方向有所改变的维度上，更新速度变慢，这样就可以加快收敛并减小振荡。

3.4.3 Adagrad

相较于 SGD，Adagrad 相当于对学习率多加了一个约束，即：

$$\theta_{t+1,i} = \theta_{t,i} - \frac{\mu}{\sqrt{\sum g_{t,i}} + \epsilon}$$

Adagrad 的优点是，在训练初期，由于 g_t 较小，所以约束项能够加

速训练。而在后期，随着 g_t 的变大，会导致分母不断变大，最终训练提前结束。

3.4.4 Adam

Adam 是一个结合了 Momentum 与 Adagrad 的产物，它既考虑到了利用动量项来加速训练过程，又考虑到对于学习率的约束。利用梯度的一阶矩估计和二阶矩估计动态调整每个参数的学习率。Adam 的优点主要在于经过偏置校正后，每一次迭代学习率都有个确定范围，使得参数比较平稳。其公式为：

$$\theta_{t+1} = \theta_t - \frac{\mu}{\sqrt{v_t^1 + \epsilon}} m_t^1$$

其中：

$$m_t^1 = \frac{m_t}{1 - \beta_1^t}$$

$$v_t^1 = \frac{v_t}{1 - \beta_2^t}$$

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$$

小结

通过实践证明，Adam 结合了 Adagrad 善于处理稀疏梯度和

Momentum 善于处理非平稳目标的优点，相较于其他几种优化器效果更好。同时，我们也注意到很多论文中都会引用 SGD，Adagrad 作为优化函数。但相较于其他方法，在实践中，SGD 需要更多的训练时间以及可能会被困到鞍点的缺点，都制约了它在很多真实数据上的表现。

3.5 损失函数的选择

深度学习同样有许多损失函数可供选择，如平方差函数（Mean Squared Error），绝对平方差函数（Mean Absolute Error），交叉熵函数（Cross Entropy）等。而在理论与实践中，我们发现 Cross Entropy 相比于在线性模型中表现比较好的平方差函数有着比较明显的优势。其主要原因是在深度学习通过反向传递更新 W 和 b 的同时，激活函数 Sigmoid 的导数在取大部分值时会落入左、右两个饱和区间，造成参数的更新非常缓慢。具体的推导公式如下。

一般的 MSE 被定义为：

$$C = \frac{1}{2}(a - y)^2$$

其中 y 是我们期望的输出， a 为神经元的实际输出 $a = \sigma(Wx + b)$ 。由于深度学习反向传递的机制，权值 W 与偏移量 b 的修正公式被定义为：

$$\frac{\partial C}{\partial W} = (a - y)\sigma'(a)x^T$$

$$\frac{\partial C}{\partial b} = (a - y)\sigma'(a)$$

因为 Sigmoid 函数的性质，导致 $\sigma'(z)$ 在 z 取大部分值时会造成饱和现象。

Cross Entropy 的公式为：

$$H(y, a) = - \sum y_i \log(a_i)$$

如果有多个样本，则整个样本集的平均交叉熵为：

$$H(y, a) = -\frac{1}{n} \sum \sum y_{i,n} \log(a_{i,n})$$

其中 n 表示样本编号，i 表示类别编号。如果用于 Logistic 分类，则上式可以简化成：

$$H(y, a) = -\frac{1}{n} \sum y \log(a) + (1 - y) \log(1 - a)$$

与平方损失函数相比，交叉熵函数有个非常好的特质：

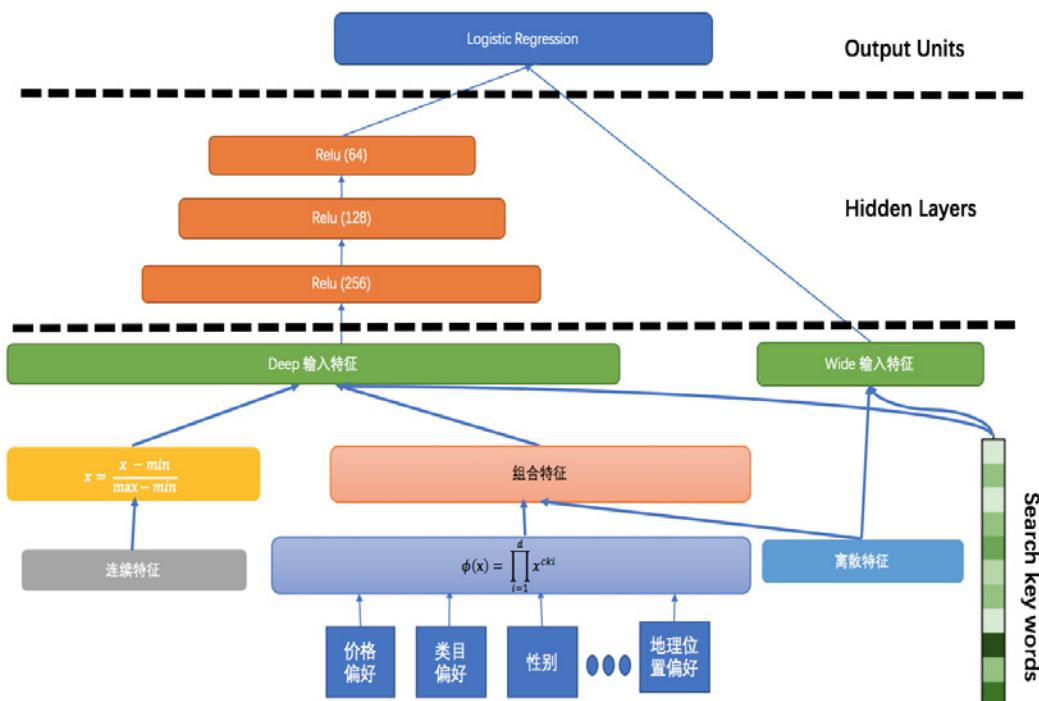
$$H(y, a) = \frac{1}{n} \sum (a_n - y_n) = \frac{1}{n} \sum (\sigma(z_n) - y_n)$$

可以看到，由于没有了 σ' 这一项，这样一来在更新 w 和 b 就不会受到饱和性的影响。当误差大的时候，权重更新就快，当误差小的时候，权重的更新就慢。

3.6 宽深度模型框架

在实验初期，我们只将单独的 5 层 DNN 模型与线性模型进行了比对。通过线下 / 线上 AUC 对比，我们发现单纯的 DNN 模型对于 CTR 的提升

并不明显。而且单独的 DNN 模型本身也有一些瓶颈，例如，当用户本身是非活跃用户时，由于其自身与 Item 之间的交互比较少，导致得到的特征向量会非常稀疏，而深度学习模型在处理这种情况时有可能会过度的泛化，导致推荐与该用户本身相关较少的 Item。因此，我们将广泛线性模型与深度学习模型相结合，同时又包含了一些组合特征，以便更好的抓住 Item–Feature–Label 三者之间的共性关系。我们希望在宽深度模型中的宽线性部分可以利用交叉特征去有效地记忆稀疏特征之间的相互作用，而在深层神经网络部分通过挖掘特征之间的相互作用，提升模型之间的泛化能力。下图就是我们的宽深度学习模型框架：



在离线阶段，我们采用基于 Theano、Tensorflow 的 Keras 作为模型引擎。在训练时，我们分别对样本数据进行清洗和提权。在特征方面，对于连续特征，我们用 Min-Max 方法做归一化。在交叉特征方面，我们结合业务需求，提炼出多个在业务场景意义比较重大的交叉特征。在模型方面我们用 Adam 做为优化器，用 Cross Entropy 做为损失函数。在训练期间，与 Wide & Deep Learning 论文中不同之处在于，我们将组

合特征作为输入层分别输入到对应的 Deep 组件和 Wide 组件中。然后在 Deep 部分将全部输入数据送到 3 个 ReLU 层，在最后通过 Sigmoid 层进行打分。我们的 Wide & Deep 模型在超过 7000 万个训练数据中进行了训练，并用超过 3000 万的测试数据进行线下模型预估。我们的 Batch - Size 设为 50000，Epoch 设为 20。

4. 深度学习线下 / 线上效果

在实验阶段，分别将深度学习、宽深度学习以及逻辑回归做了一系列的对比，将表现比较好的宽深度模型放在线上与原本的 Base 模型进行 AB 实验。从结果上来看，宽深度学习模型在线下 / 线上都有比较好的效果。具体结论如下。

表2:不同模型之间AUC对比

算法版本	AUC
Base Model	68.85%
Deep Learning (256 hidden units)	69.98%
Wide & Deep without Cross feature (256 hidden units)	70.65%
Wide & Deep with Cross feature (256 hidden units)	71.81%

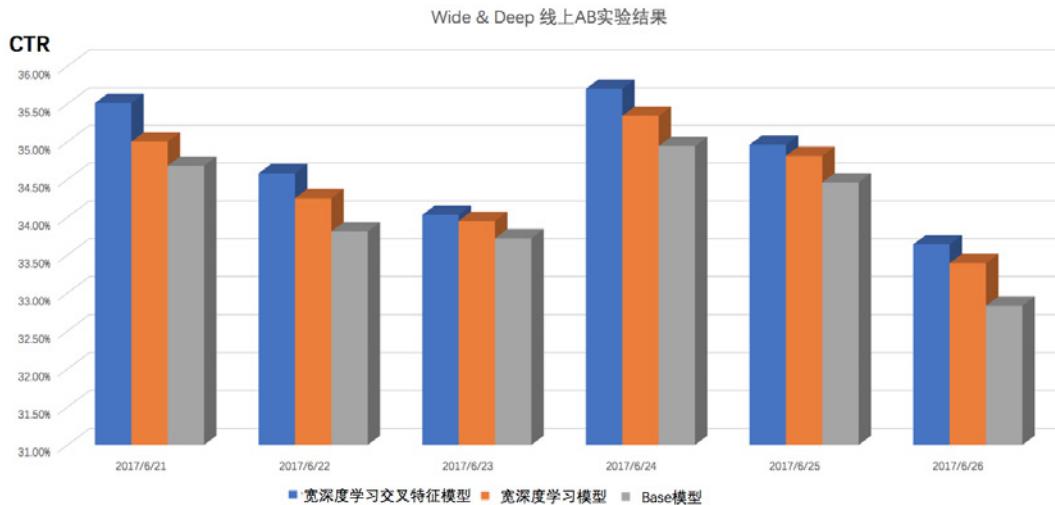
随着隐藏层宽度的增加，线下训练的效果也会随着逐步的提升。但考虑到线上实时预测的性能问题，我们目前采用 256->128->64 的框架结构。

表3:不同隐藏层之间，AUC对比

隐藏层	(Wide & Deep)AUC with cross features
512 ReLU	72.21%
256 ReLU	71.81%
256 ReLU -> 128 ReLU -> 64 ReLU	71.66%

下图是包含了组合特征的宽深度模型与 Base 模型的线上实验效果对

比图。



从线上效果来看，宽深度学习模型一定程度上解决了历史点击过的团单在远距离会被召回的问题。同时，宽深度模型也会根据当前的场景推荐一些有新颖性的 Item。



5. 总结

排序是一个非常经典的机器学习问题，实现模型的记忆和泛化功能是

推荐系统中的一个挑战。记忆可以被定义为在推荐中将历史数据重现，而泛化是基于数据相关性的传递性，探索过去从未或很少发生的 Item。宽深度模型中的宽线性部分可以利用交叉特征去有效地记忆稀疏特征之间的相互作用，而深层神经网络可以通过挖掘特征之间的相互作用，提升模型之间的泛化能力。在线实验结果表明，宽深度模型对 CTR 有比较明显的提高。同时，我们也在尝试将模型进行一系列的演化：

1. 将 RNN 融入到现有框架。现有的 Deep & Wide 模型只是将 DNN 与线性模型做融合，并没有对时间序列上的变化进行建模。样本出现的时间顺序对于推荐排序同样重要，比如当一个用户按照时间分别浏览了一些异地酒店、景点时，用户再次再请求该异地城市，就应该推出该景点周围的美食。
2. 引入强化学习，让模型可以根据用户所处的场景，动态地推荐内容。

深度学习和逻辑回归的融合使得我们可以兼得二者的优点，也为进一步的点击率预估模型设计和优化打下了坚实的基础。

作者简介

潘晖，美团点评高级算法工程师。2015 年博士毕业后加入微软，主要从事自然语言处理的研发。2016 年 12 月加入美团点评，现在负责大众点评的排序业务，致力于用大数据和机器学习技术解决业务问题，提升用户体验。

Twitter 机器学习平台的设计与搭建

作者 郭晓江



本文简单介绍一下 Twitter 机器学习平台的设计与搭建，也希望从规范化机器学习平台的角度来主要讲一些我们在这个过程中所遇到的各种坑，以及我们做的各种的努力，也希望能对大家有一点用处。

咱们今天下午的专题是“大数据”专题，机器学习和大数据是密不可分的。如果我们将数据比作一座金矿，机器学习就是挖掘金矿的工具。俗话说：顺势而为。那么机器学习在近些年来也是发展越来越好，应用越来越广，我认为主要得益于以下几个趋势：

1. Data Availability

我们可以获得的数据量越来越大，一会在下一张 slide 中也会讲到如

果数据量越大，我们模型的质量会有显著提高；

2. Computation Power 越来越强

比如最近出现的云计算、GPU、TPU 等等。在上世纪九十年代其实神经网络的理论就已经有了，也就是深度学习的这些理论已经有了，但是当时并没有火起来，甚至一度被学术界认为这是一个没有前途的方向，就是因为当时这个 computation power 没有到位。

随着近些年来这些方面的不断提高，使得我们可以训练更复杂的模型。大家都知道如果你没有太多的数据或者 computation power 不够多，则只能在一个小数据集上做训练，如果模型非常复杂就会出现过度拟合(Overfit)。所以只有当我们把这些问题全都克服之后，我们才可以训练更复杂的模型，得到一些更好的结果；

3. Development in Algorithms

整个算法的发展，会有无数机器学习的研究者，他们不断地去 push the boundary of machine learning。

从大数据和模型的表现关系来看，在十几年前有两个研究者他们将当

More Data or Better Model?

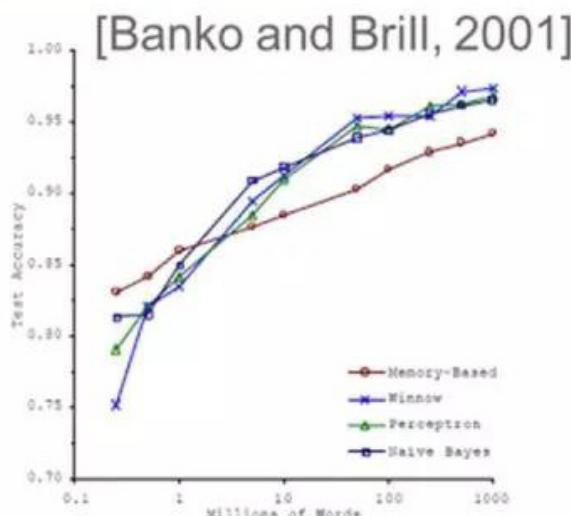


Figure 1. Learning Curves for Confusion Set Disambiguation

时几个机器学习比较常用的算法，在一个具体的机器学习的问题上做了一个实验：

这张图的横轴是数据量，即训练数据的数据量，它是一个指数的规模（Scale）。最左边的刻度应该是 10 万个数据点、100 万个数据点和 1000 万个数据点以此类推；纵轴是模型的表现，即训练出来模型的质量。

大家可以非常清楚地在图中看到，当数据量很小的时候，例如 10 万个数据点时这几个算法的质量非常差，当数据量逐渐增大的时候，模型的质量显著地提高，而且任何一个算法在大数据量时的表现都比任何一个算法在小数据级的表现下要好很多。当然这是在某一个具体的机器学习问题上面做的实验，但是我觉得它有一定的推广价值。它给我们的启示是：如果机器学习的平台架构不够规模化，只能在小数据级上做训练，哪怕你算法做得再好也是徒劳，不如先解决规模化的问题，先在大数据上能够做这样一个训练，然后在算法上再做提高。

说到 Twitter，机器学习在 Twitter 是非常重要的。我们有研究表明：大概 80% 的 DAU 都是直接和机器学习相关产品相关的，90% 的营收来源于广告，而广告完全是由机器学习来支持的。我们现在做的机器学习平台支持了 Twitter 很核心的业务，包括：

- ads ranking (广告排序) ;
- ads targeting;
- timeline ranking (feed ranking) ;
- anti-spam;
- recommendation;
- moments ranking;
- trends。

Twitter 的机器学习规模也非常大，我们拿广告来举例子，每天在 Twitter 大概是做 10 个 trillion 量级的广告预测，每个模型的 weights 个数大概是 10 个 million 的量级，每个 training example 大概是有几千到 1 万个 features，每一个数据点上有这么多，整个 Feature Space 大

大概是百亿的量级，训练的数据也是 TB 量级，所以大家可以看到对机器学习平台的挑战是非常大的。

机器学习在 Twitter 有比较独特的一点是 Realtime（实时性），Twitter 本身的产品非常的 realtime，Twitter is all about realtime，like news、events、videos、trends，比如大家去 Twitter 上更多地是更新自己的状态，或者是看一些新闻，去了解一些最新的动态；广告商也会根据产品的特点去投放一些广告，他们往往投放的广告持续的时间都非常短，比如就是一个事件，如 NBA 总决赛，三个小时内做一个广告投放，所以要求我们机器学习的模型就必须根据实时的 traffic 的情况来不断地做调整和变化。否则，如果我们每天训练和更新一次模型，这样的速度就实在是太慢了，所以我们也是投入非常多精力做了一个规模化的在线学习的系统。你在 Twitter 上点击任何一个广告，那么在百毫秒的量级的延迟内，我们的模型就会更新。

下面我简单地过一下机器学习在 Twitter 上几个具体的产品应用。

1. Ads Ranking

它的具体问题是当你上了 Twitter，我后面有 1 千个或者 1 万个广告可以展示给你，我到底展示哪个广告给你你最可能感兴趣？因为 Twitter 采取的是 CPC (Cost Per Click) Model，只有你点击了广告，广告商才会给我们钱，如果你只是看了广告，不感兴趣没有点，广告商是不会给我们钱的，所以选取最合适的广告不光是给用户更好的用户体验，同时也是 Twitter 盈利的关键；

2. Timeline Ranking (Feed Ranking)

将你的时间轴进行排序，把最好的 Tweet 能放在比较靠上的位置，这样容易被看得到；

3. Recommendation

推荐你可能最感兴趣的人；

4. Anti-Spam

比如抓僵尸粉，或者是 Abuse Detection，例如大家在 Twitter 上骂起来了，要做一些检测并且把它隐藏掉，还有 NSFW Detection 基本上是鉴别一些黄色图片之类的。

大家也可以看到，机器学习平台面临的挑战其实主要是规模化的挑战。规模化我认为主要是两方面：

- 一方面是组织架构上的规模化，我们作为机器学习平台的组如何更好地去支持这样七八个 Twitter 的核心产品，并且能够让我们的 client team（我们的用户）能够非常快地进行 prototype（产品迭代）；

- 另一方面是整个系统的规模化：一方面你的离线训练如何能更快，在线预测如何能够更快；还有一方面，当我们的用户把整个 pipeline 搭建起来之后，他们要不断优化整个 pipeline，我们有没有足够的工具支持我们这些用户做到这一点（我说的用户是 Twitter 内部的这些产品团队）。我们怎么让我们的用户非常快速地进行迭代和实验？

一、组织结构的规模化

我们相信我们的用户真正了解他们具体做的事情、他们的产品和他们的问题，所以我们采取的合作模式是：

- 我们开发各种的工具、很多的框架，去定义 feature（特征）、transform（变换）、model（模型）等等的格式，然后把工具交给我们的用户，让他们从特征提取到离线训练、如果这个模型好再推到在线生产环境当中、以至后面持续不断地优化提高，在整个过程中我们希望把每一步都做到足够的简便。同时我们还对于一些新的用户提供一些 onboarding 的支持；

- 我们的 client team 负责做特征提取的，因为只有他们了解具体的自己的问题，知道什么样的信号可以对他们的的问题有更好的提升。

我们也会把整个特征的共享做到非常好，比如其他 team 有一个很好的特征，你可以非常快地加入你的模型中进行实验。同时我们的 client team 他们负责去 own 和 maintain training pipeline 和 serving

runtime，例如 on call 完全不是我们的事，完全由 client team 来做。

二、系统的规模化

主要分几个方面：

1. 准备数据，既然要进行模型训练当然要把数据准备好；
2. 离线的训练，会有 workflow management；
3. Online Serving（在线服务），比如模型训练好了，要推到市场环境中去，要可以承受这种 high QPS、low latency 这些要求，还要做 A/B testing，在 1% 的数据上先做一些实验，然后过一段时间，真正它在实际的 traffic 上更好的话我们就把它 launch 到 100%。与此同时还做很多工具，来帮助我们的用户更好地去理解他们的数据和模型，以及还有一些工具去做比如参数的扫描、特征的选择等等。

三、准备数据

首先，我们要做的是统一数据格式，定义一个数据格式很简单，但是我觉得意义非常重大，就像秦始皇统一六国以后先统一度量衡，因为只有大家用一样的格式大家才能彼此互相沟通、交流和分享。

举个例子：比如某个产品团队在他们在模型中加了某一种信号或特征，结果特别好，我是做广告的，我想把数据拿过来用，如果数据的格式都不一样，我还得过去去研究你们这个组的数据格式到底是什么样子的，我们怎么转换成我们的格式，有非常非常多时间浪费在这个地方，这是我们希望解决的，Enable feature sharing across teams and make machine-learning platform iteration very easy.

我们的特征向量的格式，其实本质上是 feature identifier to feature value mapping，它支持 4 种 dense types：

- Binary；
- Continuous；
- Categorical；

- Text

2 种 sparse feature types:

- SparseBinary;
- SparseContinuous

为了去优化效率，我们 feature identifier 是用 64 位 feature id 存的，这个 feature id 是 feature name 的一个 hash。之所以这样做，是因为如果你在训练的过程或者在你的生产环境中，你去操作很多个 string 的话，特别费 CPU，所以我们采取的方式是使用 feature id，而在别的地方存一个 feature id to feature name 的 mapping。比如我们存在数据仓库中的数据都是 feature id 的，但是每个机器学习的数据集旁边我们都会存一个 metadata，就是 feature id to feature name 的 mapping。

说到数据准备，大家可以想象一下：如果让一个数据科学家用语言描述怎么样准备他的数据，往往这个数据科学家在非常短的时间比如 1 分钟时间之内就可以描述清楚：比如我们把这个 production 中 scribe 的数据拿过来，然后和另外一个数据集做 join，然后做一些 sampling 和 transform，然后写到 persistent storage 里面去。

我们开发了一套 DataAPI，对机器学习的数据集以及数据集的操作在很高的层次上做的一些抽象，当你用我们这一套 API 去描述你想对数据进行操作过程时，这个代码就跟你描述出来你要做什么事情和我们期望达到的效果一样的简单，我们希望这使得我们大部分的 machine-learning task 在训练过程中的数据准备都能够通过 20 行或者 30 行代码就搞定。

它是基于 Scala 的 API，一个 fluent 的 interface，并且在整个过程中去保证我们的数据正确，以及刚才我们说的 feature id to feature name mapping 的 metadata 是 keep consistency。之后我们简单看一小段代码，举一个例子来让大家感受一下，这是用 Scala 写的，看这个代码的话，其实从代码上你完全就能明白我要做一件怎样的事情。

首先我是从 FeatureSource 里面读出了我机器学习里的数据集，并

Example

```
val tweetTopic = TweetMediaClassificationFeatureSource().read
  DailySuffixFeatureSource(args("input"))
    .read
    .filter(0.1)
    .transform(discretizer)
    .joinWithSmaller(SharedFeatures.TWEET_ID, tweetTopic, new LeftJoin)
    .write(DailySuffixFeatureSink(args("output")))
```

1. Take my dataset whose path given by “input”
2. Sample it by 10% randomly
3. Discretize with the given discretizer
4. Left join with media label on tweet id
5. Dump the result to path given by “output”

存在了 tweetTopic 这样一个变量里，然后我再从另外一个地方，从我的一个 input path 里读出另外一个数据集，并且把他 filter/sample by 10% randomly，然后用我给定的 discretizer 来进行 transform，然后把它和我刚才的 tweetTopic 数据集进行 join，它们的 join key 是 tweet id，并且使用的是 LeftJoin，最后我再把这个数据集写出去，这样我整个过程就准备好了。

其实读一下代码你会发现整个代码其实是非常好懂的，在写代码的过程其实就像在描述。我们的目标就是希望算法工程师在写这个代码的时候就像描述他们想做的事情一样。比如我们对数据的位置、格式等进行抽象。不管你的数据到底在哪里，比如你的数据可以在 hdfs 上，可以在 database 里，可以在很多其他地方，但是在这个 API 里，其实都抽象在 FeatureSource 里，然后用 read 就可以把它读出来，所以用户在使用的时候是不需要操心数据到底存在哪里等等这类事情。

四、Trainer

我们也提供了很多的 trainer，让我们的用户把这些 trainer 进行一定的组合作为他们的 offline training pipeline。首先是 large scale

logistic regression learner，我们有两个解决方案：

1. Vowpal Wabbit

是 John Langford 开源的 C++ trainer；

2. Lolly

是 Twitter 内部开发的基于 JVM 的 online learning trainer，因为 Twitter 整个 stack（技术栈）都是基于 JVM 的，比如 Java、Scala，所以我们开发了这个 learner 会和 Twitter Stack 会结合地更好一些。

在 discretizer 方面我们都比较标准了，像 Boosting tree (GBDT、AdaBoost)、Random forest、MDL discretizer 等；

在 Deep Learning 方面，我们是基于 torch 做的，也有一些 Deep Learning 的 libraries。

五、PredictionEngine

刚刚提到 Twitter 这种实时性是非常非常重要的，所以我开发了一个在线学习的一个引擎，叫 PredictionEngine，这是专门为 Large scale online SGD learning 来做的，我们整个广告包括我们的 Feeds Ranking 都是用的这个 PredictionEngine。

在 offline training 其实整个 PredictionEngine 简单地包一层 application layer；在 online serving 的时候，PredictionEngine 包一层 online service layer，加这个 layer 去处理一些像 RPC 等等这方面的东西，它基本的架构是：

- 第一层是 Transform，用户可以去定义或者用我们提供的 transform 来对 feature vector（特征向量）进行一定的变换；
- 第二层是 Cross，Cross 的意思是我可以把我的特征分组，比如分成四五组，然后第一组和第二组所有特征进行 Cross，比如在广告上这个好处是可以把 advertiser id，即把每个广告商的 id 分到一组，把其它的 features 分到第二组，然后第一组和第二组

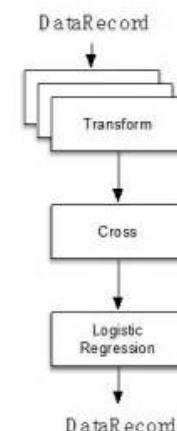
—Cross，其实effectively给每一个广告商一个personalized feature，这是非常有效的；

- 第三层是Logistic Regression；

这个Architecture一方面很方便地让我们进行在线的学习，同时在 transform layer 和 cross layer 我们也加进去了足够多的这种 nonlinearity（非线性）元素。如果只是简单的 logistic regression，那是线性的，效果并没有那么好，于是我们加了 transform layer 和 cross layer 会解决一些非线性变换的问题。

PredictionEngine

- Large scale online SGD learning
- Used in both offline training and online serving
- Architecture
 - Transform: MDL, Decision tree
 - Feature crossing
 - Logistic Regression: Vowpal Wabbit or in-house JVM learner



那么对 PredictionEngine 我们做了非常多的优化，在这个地方我会详细地讲。

1. 我们希望减少序列化和反序列化的代价

第一点是 model collocation，model collocation 是什么意思呢？就比如在广告的预测中，我们预测的不是一个概率，即用户有多少可能性去点击这个广告，我们可能是预测很多个概率，比如用户可能转发这个 tweet 的概率，用户点击这个的 tweet 里面的链接的概率，或者是用户点

击了这个链接还购买的概率，或者用户直接把这个广告叉掉的概率。

对于一个用户和广告的这么一个 pair，我们会预测很多个概率，比如你要预测 5 个概率，本来是应该去做 5 次 RPC call 的，但是我们会把这五个模型都放在一个 physical container 里面，这样的话一个 call 过去，我可以在 5 个模型中都进行计算并把 5 个 prediction 都给你返回，这是第一个优化。

第二点是 Batch request API，还是拿广告问题举例，对于一个用户我要去评估可能成百上千甚至上万的广告的数量，对于任何一个用户和这个广告的 pair，其实用户的特征其实都是一样的，所以有一个 Batch 的 API 的话，我可以 amortise cost for user feature；

2. 我们希望减少 CPU 的 Cost

也是做了几方面的优化。

- 所有的feature identifier全都是用id而不是feature name；
- Transform sharing：刚才可以看到PredictionEngine里面，第一步是做transform，由于我们有model collocation可能有五六个模型，但其实可能有些模型他们的transform是一样的，所以在这个层面上我们不要做重复的transform，如果不同的model的Transform都是一样的话，我们就把它识别出来并且只做一次；
- 最后是feature cross done on the fly，因为feature cross其实是特征从几百个变到几千个甚至几万个的过程，比如原始特征几百个，cross之后特征数量可能大量增加。如果这时候我们把cross完的feature的再存到我们的内存中去，这个cross就太大了，即使只是对这个cross后的结果扫描一遍的代价都非常地大，所以要做成on the fly的cross。

3. Training/Serving throughput

在整个在线学习过程之中，它的瓶颈在于最后 trainer 的模型 update，在 update 模型的时候就要对这个模型加锁。如果不优化的话，

只能有一个线程来对整个模型进行更新。如果你的模型特别大，比如我们每一个模型都是上 GB (Gigabyte) 的，在这个情况下就会严重的影响 training throughput。

所以我们的优化会对整个模型进行 sharding，比如用多线程。比如有 10 个线程，每个线程分别负责这个模型的十分之一，算出来整个模型的 update 的时候把它切成 10 块，扔到 10 个 queue 或 buffer 里面去，让这 10 个线程来更新自己相应的模型的那一块，所以只是需要每一块的 worker 自己更新自己那块的时候对那块进行加锁就可以了；

第二个是把 training 和 prediction 分离，因为在线学习的话，我们需要在线去响应很多的请求，如果每一个模型、每一个 instance 里面都有一个 training 都在做在线学习其实是很重复的。比如你有 1 千个 instances 都在做在线学习，并且都在做实时响应请求，1 千个 instances 里面的 training 部分是冗余的，所以我们会把 training 这部分单独拿出来作为 training service，定期会把这个模型的更新去放到一个 queue 里面，然后 fanout 到所有的 predition service instance 里面去；

第三个是弹性负载，比如我们的 client 端要 call 我们的 prediction service 的时候，我们会在 client 端加一个检测请求延迟，当我们在 client 端检测到 prediction service 不堪重负，这个时候我们会动态地减少对 prediction service 的请求，以保证我们的 prediction service 是良性和健康地运转的。

因为大家知道每天的流量会有周期变化，比如某些时段流量特别高，某一些时段比如在夜里流量相对比较低。通过弹性负载动态调整的机制，比如等到白天上午十点或者晚上八点特别忙的时候，我们可以做到对每一个用户评估少一点的广告，比如评估 2000 个广告；如果是到半夜，每一个用户可以评估多一点的广告，如 1 万个广告。这样动态地去保证 CPU 的使用率都是在固定的 level 上。这个 Level 的制定是要考虑不同数据中心之间的 failover，比如数据中心挂了，所有的这些 traffic 都要

failover 到某一个数据中心，然后还要留一点余量，所以我们一般 CPU 的 utilization 是在 40% 左右。

4. Realtime feedback

在线学习很重要一点是 feedback 一定要及时，但是有一个很不好解决的问题，如果用户他点了这个广告，这是正向的反馈你马上能知道，但是用户没有点这个广告你这事就不能马上知道，说不定用户过五分钟以后才点呢。

常用的解决方式是：我把这个广告先存起来，然后等十五分钟，看看用户有没有点，如果用户在十五分钟内点了，我们就说这个用户点了，这是一个 positive training example，然后把它发到在线学习服务中去；如果用户没有点，这就是 negative training example。

这样的问题就是说我们会有十五分钟的延时，这一点是非常不好的，所以我们做了一个优化：每当我们展示一个广告的时候，我们马上给在线学习服务发一个 negative training example，当成一个用户没有点击的事件，然后当用户后面真正去点了这个广告的话，那时我们会对这个事情进行一定的修正，这样就保证所有的事件实时性都非常高，是没有延迟的。

5. Fault tolerance

我们的模型可能有几千个 instances，这些 instances 经常地挂。我们需要每隔一段时间对我们的模型进行一个 snapshot，如果某一个 instance 挂了，另外一个重新启动的时候，它会去把最新最近的 model snapshot load 进来，然后再开始进行在线学习。

还有一个问题是 anomaly traffic detection，因为在线学习十分危险，因为上游数据任何的错误都会马上影响到这个模型的质量。举个例子，比如你有个 pipeline，有两个 queue，一个 queue 专门发 positive training example，另一个是发 negative training example，结果你的 positive 的 queue 给挂了，这样在线学习的模型一直只能接到 negative training example，于是模型的预测在非常短的时间内整个模型就全乱

了，像 Twitter 这种公司肯定都是有非常严格的 on call 制度，但是 on call 不能解决问题，为什么呢？当 on call 被 page 的时候，5 分钟之后打开电脑去进行干预那个时候就已经晚了，所以我们是需要做 anomaly traffic detection 做到在线学习当中，如果一旦发现 traffic 这个构成发生了很严重的变化，我们会马上停止训练。当然还是要 page on call 啦，然后让 on call 来进行人工干预解决。

六、Tooling

刚才说的是在线学习，我们还给用户提供很多工具，这些工具是为了帮助我们用户很方便地区对整个模型进行研究或者做一些改进。这个工具叫 Auto Hyper-parameter Tuning，就是变量的自动选择。机器学习的模型尤其包括像深度学习模型都有很多的变量。往往大家选变量的时候都是拍脑袋，比如我觉得这个 learning-rate 应该是多少然后放进去，好一点的呢就暴力搜一下，或者有些就是 Random Search。

但是我们基于贝叶斯做了自动的 hyper-parameter 选择，我们会根据之前不同的 parameter setting 所跑出来的模型的结果去计算：我下一个 parameter 选择什么使得在期望意义下我对目标函数的提高会做到最大，而不像无头苍蝇一样到处去搜，而是充分地利用已经模型跑出来的数据和 performance 来选择下一步尝试的参数。

其他的 tooling，比如：

- workflow management：就是整个 offline 的训练，你需要对它进行监测，需要可复现，可以互相地分享；
- Insight 和 Interpretation：我们不希望我们的用户用我们的东西是一个黑盒，所以我们也会搞一些 tool 帮助他们看数据、看模型、去分析特征的权重和贡献等等；
- Feature selection tool：进行简单地 forward/backward 的 greedy search。

七、Work in Progress

我们的机器学习也是在不断的探索之中，这是我们努力的一些方向：

1. 最主要的方向是我们要平衡规模化和灵活性的问题。因为规模化和灵活性往往是非常矛盾的，如果你用一些 R、Matlab、Scikit-Learn 等等一些工具，它们很多东西做得不错，灵活性是可以的，但是在这些工具上要搞规模化，这个事情是非常困难的。

反过来如果要规模化，你的系统要做的非常非常专，要针对某些情况做出很多优化，但是这种情况下就会影响算法的发挥。举个例子，我们的 PredictionEngine 分三层，第一层 Transform、第二层 Cross、第三层是 Logistic Regression，如果说我想试一点别的框架和步骤，和这个假设如果不是那么一样的话，可能你就没有办法用我们的这个工具。

所以，规模化和灵活性一直是一个非常冲突和难以平衡的问题，也是大家在不断在这方面做更多的努力，我们也期望用一些 torch-based 的 large scale 机器学习，因为 torch 在灵活性方面是足够的，如果我们能够解决规模化的问题就会非常好。

2. 我们也会尝试把深度学习的一些东西在广告或者是 feeds 流上做一些实验，虽然在业界现在成功的并不多，只有 Google 他们声称在这个方面做得还可以；

3. 为我们的用户提供更好的工具，比如 visualization 和 interactive exploration。

Q&A

主持人：好，谢谢晓江，今天你讲的这一场非常爆满。我想替大家问你几个问题，第一个问题是，你们做数据和推荐这一块，能直观的给几个数据来度量一下对你们 Twitter 业务的价值吗？

郭晓江：刚才我可能提到了，Twitter 90% 的营收来自广告，所有广告都是机器学习支撑的。我四年前进 Twitter 的时候，广告组的规模的确

刚刚起步，当时那一代机器学习的架构也非常简单，模型也非常的简陋。在我们上了大规模的在线学习的东西之后，把整个 Twitter 的营收提高了大概 30% 左右，这对于 Twitter 是几十亿美金的 business，30% 是一个非常非常可观的数字，一般模型能提高 1%、2% 就已经非常不错了。

在一些基础架构的革新使得我们可以在大规模的数据上面进行训练，比如模型的复杂度和 feature 的规模都扩大了好几个数量级，的确会对我们整个 business 和整个模型的质量都有显著地提高。

主持人：30% 的提升，很酷的数字，很困难的动作，所以在场的 CTO、架构师加油整，我们数据的价值很大。第二个问题，你提到在架构上踩过很多的坑，这四年整个过程中，你觉得最难的地方是什么？

郭晓江：因为我们是一个和业务结合蛮紧密的组织，我觉得其实最难的事情是要统一所有组的机器学习的架构。因为在最开始，可能每个组都有自己的一套机器学习的东西，但是如果大家都自己一套东西，互相的 share 就非常的困难，所以我们花了非常多努力，比如就非常简单的数据特征格式的定义，要推到所有的组都相当困难。

我觉得很多时候这个挑战更多还是来自于非技术的那一块，也就是说服大家用我们的这一套系统。也许因为我做技术，我觉得技术的东西还好，但是非技术的东西要推的话确实会稍微难一些。我们现在大家都用一套机器学习平台，之后我们把学习平台进行一些更新换代，其实基本上代码一般的生命周期也就是两年左右，过两年我们又有一套新的东西来更好地替代，这时候大家都用这一套，我们替代的时候也会方便很多，所以我觉得这是一个很大的挑战吧。

主持人：好，最后一个问题是，你们 Twitter 的这件事情做得很牛，架构也做的很牛，你们有没有更多的文档或者知识能在网上跟大家分享吗？

郭晓江：我们现在并没有太多对外开放的文档，我们也考虑过有一些东西能不能 open source。因为机器学习和业务结合实在太紧密了，它和整个公司的技术战略也非常紧密，我要开源这个东西，可能所有依赖的东

西都得开源，而且机器学习这个东西更新实在太快了，所以我们花了好多时间，把这套开源了，实际上更好的东西已经出来了，所以暂时我们这方面还没有考虑。

作者简介

郭晓江，四年前加入 Twitter，先后供职于广告组和机器学习平台组。在广告组设计和构建了 ads ranking 后端平台，之后从无到有领导团队搭建了 Twitter 的机器学习平台，应用在广告推荐、Timeline Ranking、反欺诈等多个产品中，是每年几十亿美元的营收与内容推荐背后的核心。本科毕业于清华电子工程系，硕士毕业于斯坦福电子工程系。

Geekbang

极客邦科技

整合全球优质学习资源，帮助技术人和企业成长

InfoQ

lieke

EGO

EXTRA GEEKS' ORGANIZATION
NETWORKS

StuQ

lieke

技术媒体

职业社交

职业教育