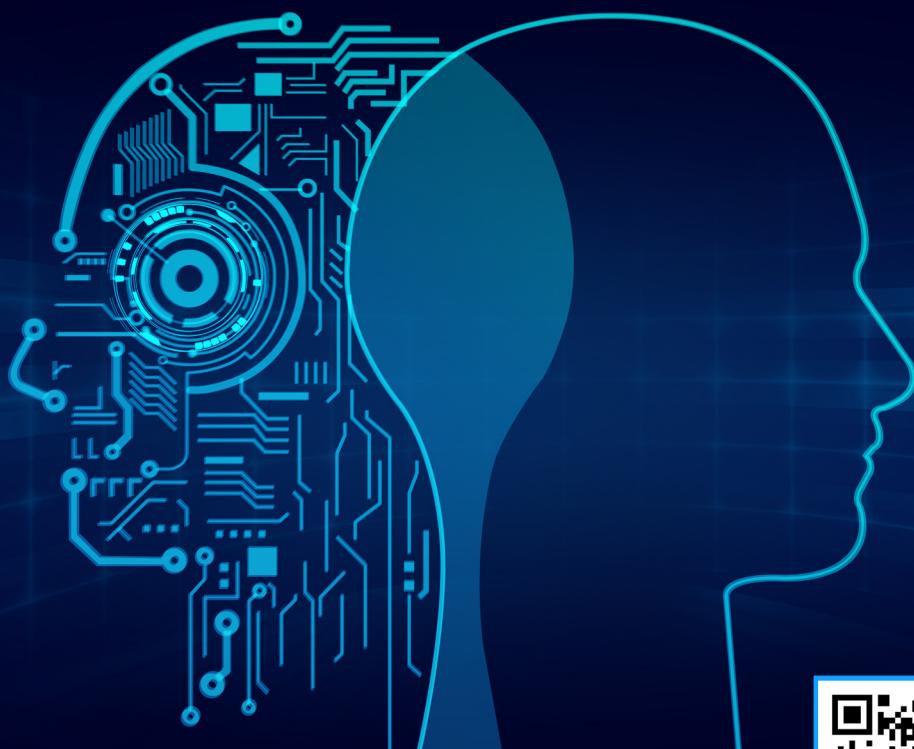


2017年11月刊

A I - F R O N T



关注落地技术，探寻AI应用场景



# 卷首语

**鲍捷**

如今的 AI 落地，核心是工程问题，不是算法问题，更不是“哲学”问题。一定要特别特别“土”，踏踏实实从朴素的运维、数据库、数据清洗做起，从实际的工程中逐步演化。如何按天迭代？如何构造联调系统？如何无标注数据启动？如何分离准确度和召回率要求？如何统一运用规则和统计？如何适应无明确衡量标准的开发？如何设计可演进的数据模式？如何提升数据可理解性？如何逐步提升规则系统的表达力？如何平衡黑箱和白箱模型的优缺点？如何在优雅架构和工期间取舍？等等，这些都是教科书上没有的答案。只有扎扎实实从工程出发，才能实事求是地发展出低成本的、有生命力的 AI 系统。

从能在顶会上发文章到能真正工程落地，中间的细节足够再读两个 PhD。



全球人工智能技术大会 2018

# 助力人工智能落地

2018.01.13 – 01.14 · 北京国际会议中心

人工智能已不再停留在大家的想象之中，各路大牛也都纷纷抓住这波风口，投入AI创业大潮。那么，2017年，到底都有哪些AI落地案例呢？机器学习、深度学习、NLP、图像识别等技术又该如何用来解决业务问题？

由InfoQ举办的AiCon全球人工智能技术大会上，一些大牛将首次分享AI在金融、教育、电商、外卖、搜索推荐、人脸识别、自动驾驶、语音交互等领域的最新落地案例，以及在落地过程中的那些痛点和难点，一些技术细节该如何操作，有哪些避坑经验，应该能学到不少东西。

## 部分演讲嘉宾



颜水成  
360人工智能研究院  
院长及首席科学家



山世光  
中科院智能信息处理  
重点实验室常务副主任  
中科视拓董事长/CTO



袁进辉 (老师木)  
一流科技  
创始人



刘海锋  
京东商城  
总架构师&技术VP



洪亮劫  
Etsy  
数据科学主管



于磊  
携程  
基础大数据产品团队总监



张浩  
饿了么  
技术副总裁



尹大朏  
摩拜单车  
首席科学家

## 精彩案例抢先看

摩拜 | 如何使用人工智能实现单车精细化运营

知乎 | 如何使用机器学习实现News Feed正向  
交互率提升100%

国美 | 推荐引擎与算法持续部署实践

微博 | 深度学习在红豆Live直播推荐系统中的应用

Tutorabc | 大数据和AI之路

饿了么 | 机器学习和运筹优化在外卖行业的应用实践

第四范式 | 如何利用大规模机器学习技术解决

微信小程序 | 商业智能技术应用实践

爱奇艺 | 自然语言处理和视频大数据分析应用

# 8折

限时优惠进行中，每张立减720元

截至2017年12月15日前

团购享受更优惠

邮箱：hedy.hu@geekbang.org 电话：18510377288 (同微信)



购票请联系



扫码关注大会官网  
获取更多大会信息

# AI 前线

InfoQ 中文站 AI 月刊 2017 年 11 月

## 生态评论

5 颜水成：学界与工业界的 AI 研究，有哪些重要不同？

## 重磅访谈

11 商汤科技杨帆：AI 落地的关键是算法闭环

## 落地实践

24 图像算法在电商大促中的应用浅析

35 人工智能在饿了么的应用实践

50 准确率 99%！基于深度学习的二进制恶意样本检测

65 阿里小蜜：电商领域的智能助理技术实践

## 推荐阅读

81 TensorFlow 与深度学习

100 ImageNet 冠军带你入门计算机视觉：监督学习与神经网络的简单实现

# 学界与工业界的 AI 研究：有哪些重要不同？

作者 颜水成



我之前在学术界，现在在工业界。现在不少学界的科学家都到公司里面做研发，通常遇到的一个问题是：在工业界从事研发和以前在学界究竟有哪些不同？很多关心研发的人会有这样的疑问。我想从自己的经历出发，谈一下我的体会，希望能提供一些借鉴。

去年，我们讲到人工智能有“三要素”：算法、算力和数据。从今年开始，我们把场景加入进来，开始用“四元分析”的方式来理解人工智能。

为什么要加入场景？去年大家对人工智能非常热情，包括学校、企业都在讨论。但是，一年过去了，大家在想人工智能到底给我们带来了什么

实实在在的价值？其实，加入场景非常重要的原因是人工智能终究是一种技术，人工智能必须要落实到精准的场景，才有它实实在在的价值。

## 学术界追逐精度的极限

我们现在来看一下，在学术界是怎么做人工智能。因为人工智能的概念实在太大了，现在深度学习最热，那我们就看下在学术界里研究深度学习，会做一些什么事情。

一般情况下，学术界是把问题设立好之后，去思考研究一些新的算法，然后在具体的问题上，力图在精度上达到极限。从深度学习上设计更好的模型结构方面，大家可以看到在过去这些年，像最初的Hinton用最基本的网络结构，到谷歌的GoogleNet，微软的残差网络（ResNet），到今年我们参加比赛所设计的模型，可以看到基础网络结构是推动学术界往前走的核心。但是除了基本的网络结构之外，更大的网络、更深的网络以及不同的网络模型的融合，也是大家追逐精度的常用方法。

另一方面，我们要训练这些网络，可能需要更多的计算资源，比如像图形处理器集群（GPU Cluster），比如说我们希望有更便捷的训练平台，比如说像Caffe、MxNet、Tensorflow等等。当然，更重要的是大家一点点往前推动的同时，积累了很多小的经验，这些经验通过学术报告，通过论文的形式来分享。大家都站在巨人的肩膀上在一步一步往前走。当然，还有怎么样用其它的非标注的数据来提升解决问题的能力。所有的一切都合在一起，在解决具体问题的时候，能够把精度达到极限。

学术界很多时候研究的目的，是要有成果论文发在最顶级的学术杂志上，也希望这些算法能够具有普适性，除了能解决自己的问题，其他人也能借鉴，最好能开源，所有的人都可以去使用，这样就能很好的提升自己在这个领域的影响力。

刚刚说的像深度学习去解决图像识别的很多问题，大家可以看到在过去的几年，错误在一点一点的降低，这正是大家在追逐精度的极限。

## 没有瑕疵的用户体验如何产生？

但是工业界不是这样。工业界要去探索商业，注定要有经济上的考虑，思考盈利模式，那对人工智能的考虑就会不一样。

在工业界里待过就会明白，人工智能本身并不是一个产品，不是单纯靠人工智能就能获得利益，必须要通过与自己的业务和场景相结合，才能发挥它的价值，核心算法只是其中的一个模块而已。无论是往前端走，还是往后端走，还是需要很多不同类型的人，才可以做出一个产品。

最重要的是，人工智能并不是一个静态的东西。比如说训练出来的模型，要用到某个业务场景里面，业务场景里产生新的数据，这些数据进一步提升人工智能模型的能力，再用到场景里面，这是一个闭环和不断迭代的过程。

另一方面，也是很多从学术界到工业界的教授和学者经常很容易犯的一个很严重的错误。就是认为技术在真正推动产品，但其实，用在具体的场景里面，技术只是起到一个非常小的作用，如果说它的贡献大概到30%到40%就不错了。

一个成功的产品，还需要产品工程师和非常多的人，大家一起才能做出一个非常完美的用户体验的产品出来。一个核心点就是我们做技术的人，做研究的人，要明白永远没有完美的算法，算法永远是有瑕疵存在的，我们一定要和场景工程师在一起，通过好的产品设计，把这些算法上的瑕疵避免掉，产生没有瑕疵的用户体验。

比如说有一个很现实的场景，人脸的检测和定位的技术之后，大家都想做一些非常有趣的增强现实的应用。早期的时候，我们特别享受技术有多么牛，比如早期产品的设计模式，会看一张图能不能把我的脸换成刘德华的脸，即使在脸动的时候，在张嘴闭嘴的时候，看起来都像刘德华。可是，很多时候如果产品的定位是这样子，技术永远都不可能做得非常好，为什么呢？人脸的场景，光照条件或者是姿态不一样，就会产生一种烧伤脸的感觉，不会产生很好的效果。但是，像我们，还有国外的一些创业

公司，他们的想法就是没有必要把人脸全都换掉，只要利用人脸定位的技术，可以在脸上加一些花卉，有蝴蝶飞，这样即使人脸定位的技术还不是很完美，还有一些抖动的情况，产生出来的视觉效果，还是可以接受的。这是一个典型的例子，需要算法和产品相互结合才能产生没有瑕疵的用户体验。

我最佩服的应该是Snapchat，他们的技术是做算法的和做工程设计的人在一起，一个一个的效果不停地打磨。他们用的人脸的技术，像分割的技术，像SLAM (simultaneous localization and mapping，即时定位与地图构建) 技术，这些技术都不是完美的。在这种情况下，通过工程师的产品设计，把每一个特效都做的非常有意思，非常酷。

此外，除了考虑用户体验，工业界设计一个产品还会考虑其它方面。比如，当前把视觉，语音和相关的技术用在智能硬件上的时候，可能会想，到底这个产品是不是能满足某种高频的刚需？

我原来在新加坡每年写很多文章，一年写50、60篇的文章都有可能。那时候有一个很明显的特点，在写文章的时候我们会造一个场景，这个场景从用户需求来说，根本就不存在；从写文章的角度来说是有价值的，从产品的角度来说，不一定有价值。工业界还会考虑一款产品用到的技术有没有成熟？比如说家用机器人，可以端茶送水，可以聊天，这是不可能的，技术上还有一个过程。

另外，工业界还会考虑技术成熟了，但有没有壁垒？假设没有技术壁垒的话，今天做一个产品出来，比较前沿的大公司，都有专家团队，你把这个产品做出来立马又失掉了，技术上的壁垒也一定要有。

另外一方面，就是学术界想得最少的：我们做一个场景，一定要有变现的模式。没有一个变现的模式，我们的产品出来了，但是今后挣不了钱，也不可能让这个公司维系下去。这些都是工业界和学术界思考的点不一样的地方。

## 用四元分析来看学界和工业界的区别

总的来说，学界进行人工智能，深度学习的研究，一直是在追求精度和极限。用四元分析的方法来说就非常有意思，即我们的场景和数据确定了，然后设定一个问题，设定一个数据集，假设有足够多的计算机资源，怎么样设计新的算法，让精度能够达到极限？

我们知道有很多的数据集，比如ImageNet，号称人工智能的世界杯；人脸研究界有LFW（Labeled Faces in the Wild，人脸图片的数据库，用来研究不受限的人脸识别问题）；在视频领域有美国组织的TRECVID；语音的话有Switchboard。他们共同特点就是：问题和数据都是确定的，用尽量多的计算机资源，去设计不同的算法，最终是希望达到精度的上限。

但是我们不得不承认，这里面很多的成果是没有办法商业化的。为什么？在ImageNet上，假设训练了1000多层的网络，把9个或更多网络全部合在一起能达成一个很好的精度，在现实的场景下是不可能用这么大的模型和这么多的资源去做一件事情。所以，很多的成果，是假设将来计算能力达到一定的程度，精度能够达到这个上限。

AI研究的另外一个维度是追求用户体验的极限。用四元分析的方法，是把场景和算力固定了。这是什么意思？假设我们要做一个机器人，这个机器人希望它能识别你，这时候场景是确定的。算力确定了是说，这个场景推出的时候，用什么样的芯片和什么样的硬件，其实已经确定了。我们要做的事情是在这样一个确定场景和算力的情况下，怎么样去提升数据和算法，跟具体的应用场景去形成一个闭环，去不断地迭代，去提升它的性能。这跟学术界把场景和数据固定是完全不一样。在这种场景下，可以不停的用收集到的新数据不断提升和优化模型，在数据，算法和场景形成一个闭环。虽然我们能把所有的问题解决，但是在具体的场景下，也有可能逐步地提升它的性能。

这时候做的事情很有意思，要做很多数据的清洗、标注。为了把产品的价格降低，比如用一个很差的CPU就能够去做计算，肯定要不停地去优化模型的速度。另一方面，很多时候，满足这种体验的需求会有一些新的问题出来。

如果我们仔细想一想，学术界多数做的事情是在思考，在想它的极限在哪，主要用脑；工业界并不是强调用脑，而是用心——就是怎么样能把这个场景做出来，并不一定要有非常高大上的算法，就是要从用户使用产品的维度上，让用户感觉这个产品非常好。

学术界和工业界又不是完全割裂的：工业界敢去提某一个产品的设想，是看到了在学术界有一些前沿的成果，可以在工业界来用。同时，工业界也在逐步提炼它的问题，扔给学术界，希望他们去做这种前沿的探索。比如说工业界可以想，三年、五年以后会往哪些方向去推动，他就可以把这些任务推给学术界。

现在有很多公司，在中国和在美国纷纷建立AI实验室，其实有两种目标。一方面是长期希望能瞄准将来前沿的领域，做技术的积累；另一方面是要追求产品更好的落地，所以现在很多公司就建起了自己人工智能的实验室。

在人工智能深度学习的研究，学术界和工业界的差别还是很大的，同时也相互作用，相互增强。学术界和工业界一起合作，研究和产业相结合，一定会把人工智能带上另外一个阶段。

本文系颜水成在最近某论坛的演讲整理稿，经本人确认。

## 关于作者

**颜水成**，奇虎 / 360公司的首席科学家，新加坡国立大学Dean's Chair副教授。他的研究领域包括机器学习，计算机视觉和多媒体。独自或与人合作撰写了几百篇文章，涉及广泛的研究题目。Google Scholar引用超过20,000次，H指数为66。他同时也是ISI 2014, 2015, 2016年的高被引研究者。



扫码阅读“25万年薪的你  
与25万月薪的他，猜头来  
谈你们之间的差别”

# 商汤科技杨帆：AI 落地的关键是算法闭环

作者 蔡芳芳



人脸识别技术，曾经是反乌托邦的科幻小说中出现的想法，现在可能正在成为中国日常生活的一个特色。

广东深圳已经有了人脸识别抓拍行人闯红灯的示范路口，如果你闯红灯的时候被摄像头拍了下来，下次你再试图闯红灯时，你的脸就会出现在街道旁边的显示屏上，显示屏上还会出现一行字：“人脸识别智能抓拍行人闯红灯”。

人脸识别技术已经成为监视领域最有力的新工具之一，地铁站、机场、海关都在使用这项技术。刷脸取款、刷脸支付、刷脸登机等新应用更是层出不穷，刷手机的时代仿佛也才到来没多久，刷脸时代已经来势汹汹。

今年9月下旬，一段被称为“中国天网”监控视频的视频片段在新浪微博和朋友圈里疯传，视频展示了我国最新实时行人检测识别系统，该系统可以实时监测区分出机动车、非机动车和行人，并能准确识别出机动车和非机动车的种类，以及行人的年龄、性别、穿着。而这个系统的背后，其实是商汤科技的Sense Video技术。



主打人脸识别技术的商汤科技成立于2014年10月，其核心创始人汤晓鸥，同时也是香港中文大学教授，领导着计算机视觉实验室，这一特殊的跨界身份似乎也预示了为何商汤科技未来能够横跨学术和商业两界并取得亮眼成绩。商汤科技目前拥有140位博士，2016年ImageNet大规模视觉识别挑战赛中，商汤科技联合香港中文大学一举揽下三项冠军；近日，商汤科技与香港中大-商汤科技联合实验室，继以23篇论文横扫CVPR后，又以20篇论文力压群雄称霸ICCV，在全球顶级视觉学术会议上刮起了一阵中国旋风。而在业界落地方面，商汤科技的产品遍布金融、安防、互联网娱乐、AR、智能手机等多个行业场景，与华为、Qualcomm、中国移动、小米等众多公司都达成了合作。2017年7月，商汤科技获得4.1亿美元B轮融资，成为史上人工智能最高单笔投资，直到11月2日旷视科技获得4.6亿美元C轮融资再度刷新这项纪录。

人脸识别大行其道，不免让人对这项技术及其背后的公司产生了许多好奇。人脸识别技术到底有何门道？它经历了怎样的技术演进历程？各家公司宣传的识别正确率百分之99点几后面的小数点真的有区别吗？人脸识别技术在商汤是如何落地的？它带来的安全性问题如何应对？带着这些问题，InfoQ记者来到了商汤科技（下文统称商汤）在深圳的办公室，对商汤科技联合创始人、副总裁杨帆进行了专访。

## 商汤到底是一家什么样的公司？

提到商汤，大部分人第一反应就是人脸识别，但人脸识别并不足以定义商汤。

在杨帆看来，商汤是一个坚持人工智能原创技术的平台服务提供商，它利用原创的AI技术给不同的行业提供平台化服务、赋能各个行业，让AI技术真正地去改变每个行业。“当然目前来说，我们的工作主要集中在人工智能的计算机视觉，也就是图像和视频分析的这个领域。毫无疑问，人脸作为一种非常特殊且具有极高价值的影象标识，会是整个图像视频分析领域中占比重非常大的一部分。但同时商汤还经常给不同行业提供其他解决方案，涵盖范围会远远超过人脸识别。”

## 计算机视觉技术的发展和突破

### 深度学习使CV真正从学术界走向工业应用

杨帆在计算机视觉技术领域沉浸多年，在微软任职期间，他主要从事计算机视觉、计算机图形学等领域的新技术孵化工作，包括人脸识别、图像物体识别、人像三维重建等；目前商汤的核心技术也是以人脸识别、智能监控、图像识别等为主。作为主导技术落地的负责人，杨帆笑称自己是给公司的研究员们打下手的，但回忆起计算机视觉技术的发展历程，他表示还是有很大的感触。

上世纪90年代末期，有一波所谓的人工智能，或至少是人脸识别的热潮。当时在实验室环境下，人脸识别已经能够达到一个相当不错的结果，

但离实际应用还是有比较大的差距。从2004年杨帆进入微软实习开始，到2010、2011年这段时间内，计算机视觉领域的技术进步一直在持续，但主要还是积累期，整个行业的技术进步相对比较缓慢，基本没有太多新的应用和机会。到了2011–2012年，随着硬件设备计算能力的进步，以及各大公司开始具备收集海量数据的能力，深度学习变得越来越实用，给行业带来了巨大的改变，从那之后计算机视觉技术就进入了一个特别高速的快车道。计算机视觉技术从学术界蔓延到了工业界，在各行各业都有了越来越多广泛的应用，这是外因。

从内因角度来讲，这一轮以深度学习为核心的视觉技术，对数据的依赖更强了，核心技术研发能力提高了，而且最终得到的成果普适性也变好了。杨帆回忆道，“我以前在微软做过一些人脸识别的工作，在深度学习出现之前，你做一个算法能够把肤色的问题解决得很好，但它可能对光线的问题就很难适应。假如你想要一个对光线适应很好的算法，它可能对肤色问题又解决不好，它的技术突破是单点性的突破。”

而今天，伴随着海量数据的应用，很多识别技术会变成一种相对通用的方法论，可以以更低的成本、更短的时间，快速迁移到不同的领域上，这其中的价值非常巨大。随着人工智能技术的发展，虽然它难度依然很高，但是它的不可知性和风险已经大大降低，在这种情况下，就会有越来越多的企业愿意投入力量到这些技术的研发中，从而带来更大的价值。

**以前只有世界顶尖级别的公司才会成立研究院，去做核心技术研究，比如贝尔实验室、微软等。但是今天你会发现完全不一样，我相信未来整个技术在不同行业的落地，对于整个业界生态会有比较大的改变。**

### 基础研究和应用科研，二者不可偏废

业界曾出现一种批评的声音，称现在很多公司和开发者其实对于深度学习的运作原理并不清楚，只知道应用，却不知其所以然。对此，杨帆也有自己的看法。

杨帆表示，学术界有两套观念，一套观念说知其然不知其所以然是离

经叛道、是不对的。对于这个观念，杨帆表示认可，其实现在已经有很多团队，包括商汤也投入力量在进行更加前沿、更加基础性的科研，“这样的基础科研能够指导我们将来在正确的方向上走得更远。”但杨帆认为，基础研究与应用科研，二者不可偏废，完整的科学体系和持续的方向性指引非常重要，但是实证科学也非常 important，企业最终还是要以技术落地的结果说话。

## 脱离场景谈识别正确率毫无意义

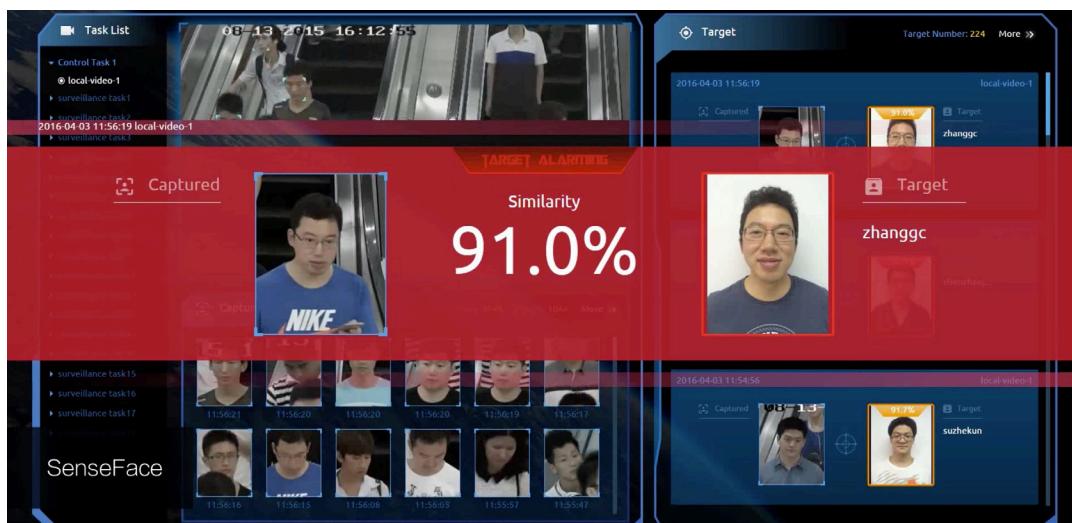
近几年，很多公司在人脸识别技术上投入了大量的研发并取得了亮眼的成绩，其中识别率一直是各家宣传的重点，今年我们能在各类报道中频繁看到各种99%、99.4%、99.8%，如何理解这些识别率中小数点后面数字的差距？

**技术指标是没法一概而论的，任何一个技术指标背后都隐藏了一大堆的假设条件。**

杨帆列举了几个例子，比如在金融场景做1:1的人脸识别，用于互联网金融的注册，这与在家用相册中做人脸识别，也就是把照片集中同一个人的照片找出来，以及在安防场景中，根据模糊的照片在一个海量的逃犯库中找到特定的人，这些场景都是人脸识别，准确率可能都差不多99%、或者99%点几。虽然企业这么宣称，但实际背后蕴含的差异是非常大的，它会有非常多影响因素，所以准确率跟行业背景以及前置假设会是一个强相关的关系。而不同的场景下取得的识别准确率很难做类比。

相比不知前提的识别正确率，更为重要的是，在不同的场景下，企业是不是能够使用原创技术真正地取得突破。在互联网相册的应用场景下，商汤可以说是全世界第一个让计算机的人脸识别超越了人类，而后续很多智能相册的业务和服务都脱胎于这项突破。在杨帆看来，当公司面临一个新的行业场景，和过去的场景不一样且遇到新的挑战的时候，是不是能够率先去形成量变的突破，这才是最重要的。当技术沉淀、数据积累和对业务场景的理解，三者融合在一起的时候，才能帮助公司完成一个真正有价值的、有意义的技术突破。

当识别率达到99%以后，人脸识别技术面临的难点主要在于，如何在不同行业场景中深化这项技术。虽然看上去99%的识别率已经很高了，但不同行业场景对于识别率的要求不同，99%可能只是该技术得以使用的入门条件，比如银行身份认证服务，如今商汤人脸识别的误识别率已经可以做到 $10^{-7}$ 次方，相当于7位银行密码，但在这个场景下也才刚刚得以使用；而安防场景下，照片模糊、有遮挡、角度不佳都给人脸识别带来了更现实的挑战。



“看似同质化很强、很简单的人脸识别，细分的技术场景其实非常复杂，所以脱离场景去谈技术是没有太大意义的，今天能看得到的，包括以安防、手机这样的一些重点行业为代表，对于真正的人脸识别技术的全面深化存在着非常多的挑战，值得我们去攻克。”

## 图像和视频分析比你想像的更复杂

**图像和视频分析其实是一个从功能或者从能力角度来看都比较复杂的技术体系，当我们一项技术落地或深化的时候，它可能需要几个团队合作完成。**

商汤在计算机视觉技术领域的探索工作大致可以分为图像增强、物体检测和分类、算法模型、训练引擎等几个方面。

图像智能化增强是图像和视频分析的第一步，虽然今天照片和视频

的采集设备已经非常好了，但图像和视频的采集还是经常面临困难，比如用红外摄像头以及结构光摄像头，拿到的深度图信息里面的噪音非常大，或者用安防设备拍摄高速运动的物体时会因为运动而导致模糊，因此分析前需要现对这些图像和视频进行智能化的增强和恢复，又叫做Low Level Vision，这在商汤是一项独立的工作，目的在于提升采集到的图像和视频的质量。

而图像和视频的识别及分析又可以细分成多个部分，包括物体检测，知道一个东西在哪里；物体的关键点定位，知道物体的关键轮廓和形状；物体的分类，就是对于找到的物体，能够知道它是什么东西；整个区域的分割，对整个物体的边缘或轮廓有非常清晰的描述。实际上，整个识别体系可能需要分成若干个不同的子领域，在真正的行业应用中，它往往是一些子领域叠加组合的应用。

商汤有专门的团队进行基础研究，比如如何将算法小型化，使之能够在资源受限的移动终端上运行；如何优化算法使之运行得更快；AI核心的训练引擎或操作系统的持续升级和演进；弱监督或无监督学习的研究，包括增强学习、迁移学习等前沿技术。

杨帆强调，从计算引擎到数据流程架构，更重要的意义其实不在于数据量，而在于让算法形成一个稳定的闭环。

## 计算机视觉技术如何落地实际产品

### 计算机视觉技术在商汤的落地场景

商汤一直非常关注计算机视觉技术的落地，杨帆在早前的一些分享和演讲中也多次提及技术进步需要与产业需求相结合。据杨帆介绍，计算机视觉技术在商汤的产品和业务中主要包含以下应用场景：

#### 1. 安防

过去对安防的理解主要是公安，其实真正意义上的安防还包括交通、线下的商业场景、小区、学校等，可以涵盖的场景非常大。

## 2. 智能终端

目前智能终端主要指手机，但它未来的形态可能会继续演化，人工智能的技术一定会在这样的终端设备上体现出非常大的价值。

## 3. 互联网视频类应用

随着互联网应用的进一步加深，它会越来越多地从文本转向图像、视频这种更加丰富的多媒体形态的应用，这些年从直播到短视频的爆发都是例子。在这方面，商汤可以给视频类应用的厂商提供非常完整而丰富的高附加值的解决方案。

## 4. 人像身份认证

基于人像的身份认证也是一个非常有价值的工作，它是一个特殊的跨行业的解决方案。这个解决方案现在已经从线上到线下开始极大范围地蔓延。对中国来说，个人公民身份信息的实名制是一个非常重要的诉求，这个诉求能够有效地帮我们在一定程度上解决互联网的安全问题、解决线下的公共安全问题。所有线上的互联网行业应用，到各种线下行业，包括机场、超市、酒店，都会有越来越多的对于个人身份信息核验的强烈需求，商汤在这方面也提供了非常完整的解决方案。

## 5. 自动驾驶

自动驾驶会是未来一个非常大的标杆性的方向，在这个过程中，人工智能技术会是一个非常关键的环节，商汤在这个领域也有一定的投入和规划。

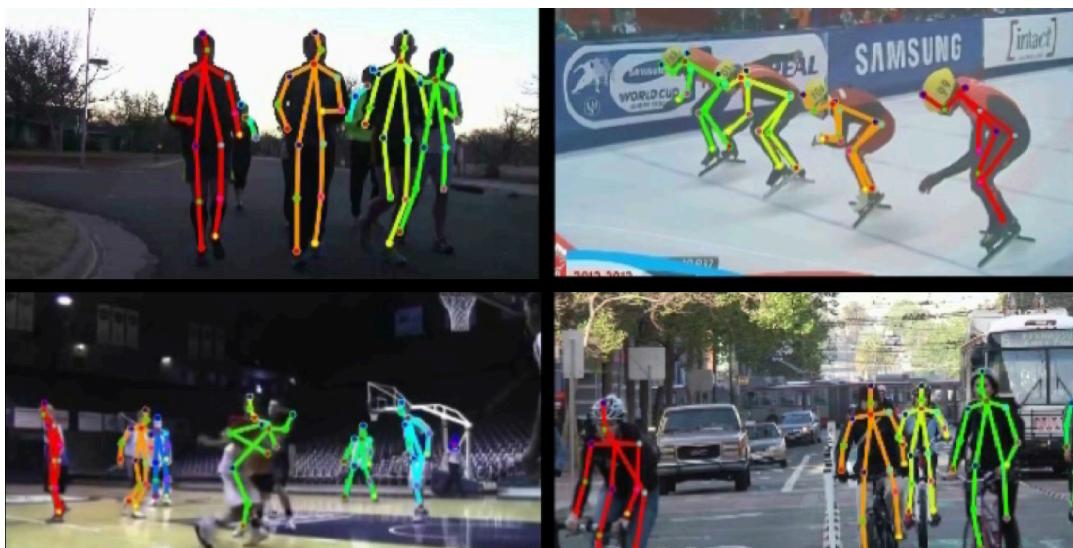
## **商汤安防场景背后的技术支撑**

一款合格的安防产品，背后绝不只靠人脸识别这一项，而是由多项技术共同支撑。

以一个广场级别的安防监控场景为例，其背后涉及的技术主要包括：

- 硬件设备，即摄像头。对于大型广场，一个摄像头无法全面覆盖，因此可能需要全景摄像头和可拉伸的近景摄像头配合，完成人脸或其他图像的采集。

- 采集算法。摄像头中会集成一个人群分析的算法，即通过收集的数据、结合人工规则，了解这个广场现在哪里人流比较密集、哪里人停留时间比较长，然后让负责抓拍和跟进的摄像头重点关注这些区域。
- 人脸识别。接下来就可以在上述区域使用人脸识别的技术，寻找是否有黑名单（比如扒手库）中的人，可以用于反扒。这也是为什么刚才要找人密集的区域、停留时间长的区域，因为这些是高发区。
- 肢体动作捕捉和识别。在寻找特定人员的过程中，需要进行人体姿态的跟踪，通过对这些人的关键动作进行检测和识别，从而判断是否出现偷窃行为。
- 图像增强。如果摄像头采集到的图片模糊了，还会用到图像增强



技术，使图像变得更适合后续步骤分析。

如杨帆所说，真正去看行业落地的时候，往往都是不同的技术叠加和组合的应用，这里面人脸识别和动作识别是最关键的技术，但实际上想把落地场景做好，一定需要多种技术组合。

## 复合型人才是AI落地的关键

杨帆表示，将创新技术转变为实际产品是一条满是荆棘的道路，行之不易，而其中最大的难点，一是如何选对方向和时机，二是如何找到合适的人才。

AI技术落地需要与行业相结合，而如何去选择需要结合的行业就是第一个难题。杨帆说，“如果技术还没有到真正能成功的门槛，比如搜索引擎中的视频搜索，大公司不断积累可能没问题，但如果是一个小的创业公司，把它作为安身立命之本，难以得到回报，可能两年之后就死了。”

杨帆表示，首先需要确认所选择的行业市场是一个真实有效、有规模的刚需市场；其次，需要在市场中真正拿到完整的闭环数据，才能获得持续性的进步；接下来，需要考虑行业当前的技术红线是不是在一个合理的区间内，介入太晚或介入太早，都是会有问题的；最后，在产品落地的过程中，需要考虑如何利用技术门槛期（通常1年到1年半）带来的优势，进一步建立行业壁垒，只有技术壁垒而没有行业壁垒的话，最后从长期来讲还是为他人做嫁衣。

从另一方面来讲，行业落地需要各种综合性的关键技术的整合。行业的需求往往是一些相对模糊的，而且从技术上来看是非常不明确的东西，这时候就需要有人有足够的能力去一一拆解。在杨帆看来，找到或培养一些既有技术背景、又对行业有足够深的理解的人才，是企业实现AI技术落地最关键的一点。他说到，“人才问题、团队组织问题、发展问题，特别是做2B行业，标准化与非标准之间的平衡性掌握，任何一个技术性产品落地会面临的共有问题，做AI技术落地，这些问题一个都不会少，而只会更严重。AI人才是个更大的坑，AI的技术性更深重，从过往来看，它跟行业的结合更弱，所以你想要真正去打磨出一个符合真正行业需求的产品的时候，需要把对行业的理解和对技术的理解融合在一起，这在我看来是最有挑战的，因为过去可能这个世界上基本不存在这样的人，对行业有理解的人很少。”

## 市场增量期，商汤更愿意合作而非竞争

人工智能领域的创业浪潮中，计算机视觉技术（CV）在国内是一个非常火热的方向，呈遍地开花之势。在安防、金融、机器人、医疗、无人驾驶等诸多业务场景都有大批公司在竞争。

安防是商汤非常重要的一个业务场景，也是国内很多计算机视觉初创企业（如旷视科技、依图、云从等）非常看重的市场，更不用说已经在这个领域深耕多年的海康威视。

杨帆认为，安防市场目前正处于高速增长期，从2018到2019年，整个安防市场还会大爆发，爆发速度可能会超过大家的想象。而商汤的定位是依托原创技术去做能力服务平台，去做不同行业的赋能者，这使得商汤更愿意跟行业上下游企业形成合作而不是竞争的关系。

## 人脸识别技术的安全性问题

人脸识别技术多用于安防和金融领域，尤其像银行、支付相关的人脸识别应用对安全性要求特别高。前不久苹果发布会上推出的FaceID也引发了大家对于其是否足够安全的讨论。

杨帆将人脸识别的安全性问题分为两种，一种是人脸识别如何做得更准确，不会误识别；另一种则是如何防御非法攻击，比如通过照片、视频等方式绕开人脸识别。随着数据量的增大以及新算法的迭代演进，人脸识别的准确率一直在不断提升，相对而言，后一个问题面对的挑战更大，这个问题在业界又被称为活体检测问题。

对于金融场景的非法攻击防御，商汤目前的做法主要是通过积累大量的攻击数据，并通过模式分析、光谱分析等方法识别出攻击行为的模式，进而抵挡这些攻击。杨帆解释说：“不管用视频还是照片，其实有很多蛛丝马迹是可以看到的，但这种蛛丝马迹人不一定能够特别好地分辨，当有大量数据的时候机器可以比较好地分辨，比如手机屏幕的反光等。”

苹果FaceID采用的3D人脸识别技术，主要的差异在于采集设备，将采集设备换成3D摄像头之后，能够采集到的图像数据信息更大，除了彩色信

息之外，还会拥有3D的数据信息，而这些深度信息能够使算法进行更好的分析，从而达到更好的人脸识别以及防御攻击的效果。杨帆认为3D采集设备的研发和发展是一个比较明确的行业趋势，商汤未来在这个方向上也会做一些尝试。

## 计算机视觉技术的未来

对于计算机视觉技术目前面临的挑战，杨帆认为主要有三点，第一是如何减少对数据的依赖，而这也是行业内大家达成共识的一个大的方向，目前的图像识别模式对于数据依赖太强，人类识别的时候并不需要这么大量的数据。第二个是整体性能优化，就是如何用更低的计算成本完成智能分析，这对于实用化非常重要。第三个则是理论研究，知其所以然还是很重要的，这样更有助于长期发展。

杨帆认为视频的分析理解是未来计算机视觉比较有前景的研究方向之一。他说，“视频的分析理解，其实大家喊了很多年，到底什么时候算是真正成熟的点，不同的人会有不同的判断，会在不同的时期投入。我个人认为互联网作为一个已经成型的、具有特别大的商业价值的体系链，视频的应用在我看来是太少而不是太多。视频或者说视觉信号的潜在价值是非常大的，因为人和人之间沟通其实视觉信息占非常重要的比例，它的信息含量非常丰富。今天互联网已经形成了非常完整的生态，它对信息的五个环节都有特别好的基础技术支撑，在这种情况下，率先对视频领域做更深的探索和挖掘其实是必经之路。很多线下的行业可能有刚需，互联网上的视频、图像，特别是视频内容分析理解相关的领域，在未来其实还是会有很大的空间，今天能够做的事情还是太少。”

在整个人工智能布局上，计算机视觉的定位是怎样的？

**视觉是最核心的，而且潜在商业价值也是最大的。**

杨帆认为，信息是一切的核心，抛开人工智能，整个IT行业所做的事情就是信息的采集、传输、存储、分析、计算和反馈。而人工智能就是在整个信息环中，机器越来越多地去承担人的角色，可能比人做得更好。人

和人日常进行交互的时候，视觉信息是更加本质的信息，所包含的信息量更大，因此计算机视觉在整个信息形态上是以一个相对高阶的形态存在，对各个环节的技术要求都会更高。一旦在每个环节上逐步具备视觉信息的处理能力之后，它所迸发出来的价值可能会超过今天IT互联网行业所能影响的空间，甚至可能会颠覆人和人、人和这个世界的交互。

在杨帆看来，计算机视觉有一个很重要的点，就是人的眼睛能够分析、感受的电磁波是一个很窄的波段，而机器却识别更宽的波段，比如红外摄像头、近红外摄像头、结构光深度的摄像头。杨帆提出了一个很有趣的问题：“这些摄像头能够把人类所能够看到的、能够处理的波段进一步扩展。那这个东西是不是可以一直扩展下去？如果从这个角度去理解，计算机视觉意味着将来机器可以替代人类，或者它作为人类的助手拥有更加本质的对这个世界的洞察。”

杨帆认为，目前我们设计、使用红外摄像头的方式思路还是从人出发的，依赖于人类经验的辅助和指导，也就是先将红外摄像头所采集到的影像信息，转化成一个人类可理解的影像，然后用机器去理解它。他说：“而下一步，很可能是红外摄像头直接去采集机器可以理解的信息形态，然后机器可以再去扩展。”

# 图像算法在电商大促中的应用浅析

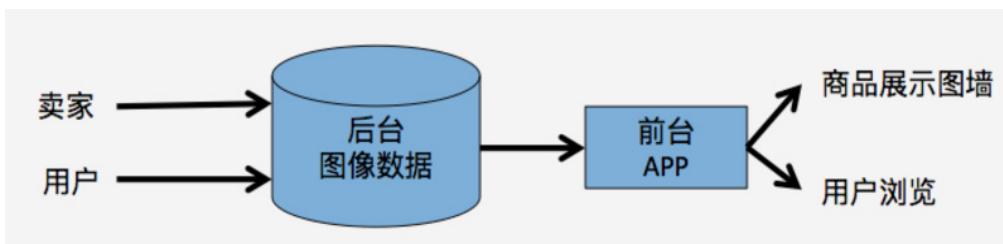
作者 民达 弘量 人可 少湖 任任



## 1. 双 11 大促的业务分析

自 2013 年转型以来，蘑菇街电商平台历经了多次双 11 大促的洗礼。通常而言，大量的商家与商品参与双 11，用户规模相比平时也会剧增。今年双 11，蘑菇街还开辟了微信小程序作为新的支点，希望以此撬动新社交电商战略。平台为用户带来价值的关键是保障商品丰富、价格合理、服务可靠。在此背景下，有很多挑战需要在复杂的业务场景中去应对，其中包括：如何提高商品管理的效率，以及如何改善用户体验。在众多的技术和产品方案中，图像算法作为一项重要能力，运用于电商场景

中，支持上述业务问题的改善。



**图 1. 电商中的图像数据**

如同 1 所示，在电商平台中可以按照业务流向简单地描述图像数据。电商平台从商家或者用户处，获取到不同来源的图像数据，并且存放于后台图像数据库；前台 APP 产品作为面向用户的界面，基于图像数据和业务算法，把商品呈现在用户眼前，主要包括商品展示图墙页面和用户浏览页面。

蘑菇街在实践中，采用了两种类型的图像算法技术支持业务发展。第一个是图像搜索技术，用于后台商品管理和前台搜相似商品；第二个是图像标签技术，应用在商品属性管理的场景中。

## 2. 图像搜索技术的应用

### 2.1 技术原理简介

给定一个图像作为 query 输入，基于内容的图像搜索过程是在指定的图像数据库中检索，找到和 query 相同或者内容相似的图像。蘑菇街的图像搜索应用场景主要集中在商品搜索上，既有移动端的搜相似购物，也有后台运营选品的需求等。

大规模商品图像检索所面临的主要挑战包括几个方面：

1. 图像数据量大，一般电商平台的商品图像包含了主图、SKU 图、商品详情图和用户评论图等，规模达到千万至亿级别。
2. 特征维度高，图像特征是描述图像视觉信息的基础，特征表达能力直接决定了图像检索的检索精度。

1. 响应速度要快，检索系统需要具备可以快速响应用户查询的能力，一般要求检索系统能够满足实时或者准实时的要求。

针对这些挑战，蘑菇街图像搜索技术的工作主要集中在两个方面。

### 图像特征的表达能力

随着深度学习的兴起，利用 CNN 提取图像特征，已成为图像检索领域的共识。商品图片类别众多、背景复杂，如何从丰富的图像信息中提取关键特征依然是很有挑战的问题。图像特征模块主要包含三个重要部分：数据清洗、特征模型设计、模型压缩。

利用 CNN 提取图像特征，关键在分类标签的定义。有文章（ICLR 2017: On the Limits of Learning Representations with Label-based Supervision）指出：模型提取特征能力的上限，不在数据集的大小，而在标签质量。因此，设计监督更强、质量高的标签，更有利于特征的表示。我们的商品标签有两个来源，一个是商品在类目体系中从属的类别，另一个是商家对商品的描述。数据清洗过程主要解决商家打标的标签和图像实际内容不符合的问题。利用自动化图像标签模块，可对商品图片自动打标，辅之以人工矫正。通过这种方式我们累积了数以千万计的样本图像数据，所涉及的标签 label 数目有几千种，从而构建了高质量的训练样本。

特征模型的设计以 ResNet(残差网络) 为基础，根据 ResNet 是浅层网络集成学习的思想（NIPS 2016: Residual Networks Behave Like Ensembles of Relatively Shallow Networks），我们通过设计不同尺度卷积核并拼接（Concat）在一起，提高了浅层网络的表达能力；同时适当控制深度，并改进 ResNet 中影响优化的 Shortcut 结构。试验证明网络的改进是有效的，改进后的网络在实际数据集合上的 top1 accuracy 是 61.8%，而传统的 ResNet-50 是 56.6%。

特征模型部署在 GPU 服务器上，为控制系统的整体响应时间，需要缩短特征提取的时间，因此要对深度学习网络模型进行压缩。压缩算法采用的是（ICLR 2017: Pruning Filters for Efficient ConvNets）所提

到的剪枝策略。具体的做法是：针对每个卷积核计算其绝对值和，然后排序，针对绝对值小的权值和通道进行剪枝。流程中包括两个主要步骤：首先按照一定比例（比如 10%）进行压缩，然后进行模型的 fine-tunning 训练；两者交替迭代进行，直至模型精度的下降超过预设的目标，流程结束。最终我们所获得的特征模型在 GPU 卡 K40 上，单次特征抽取的时间在 40ms 内。

### 近似最近邻查找

鉴于搜索数据库数据量级很大，对每个查询都要计算所有的距离是非常困难的，同时存储数千万图片的高维残差网络特征向量需要耗费巨大的存储空间。为了解决这些问题，采用了近似最近邻算法中的局部优化的乘积量化算法（Product Quantization，PQ），训练得到粗量化质心和细量化质心，粗量化的结果用来建立倒排索引，细量化结果用来计算近似距离。通过这种方法，既能保证图像索引结果的存储需求合理，也能使检索质量和速度达到更好的水平。

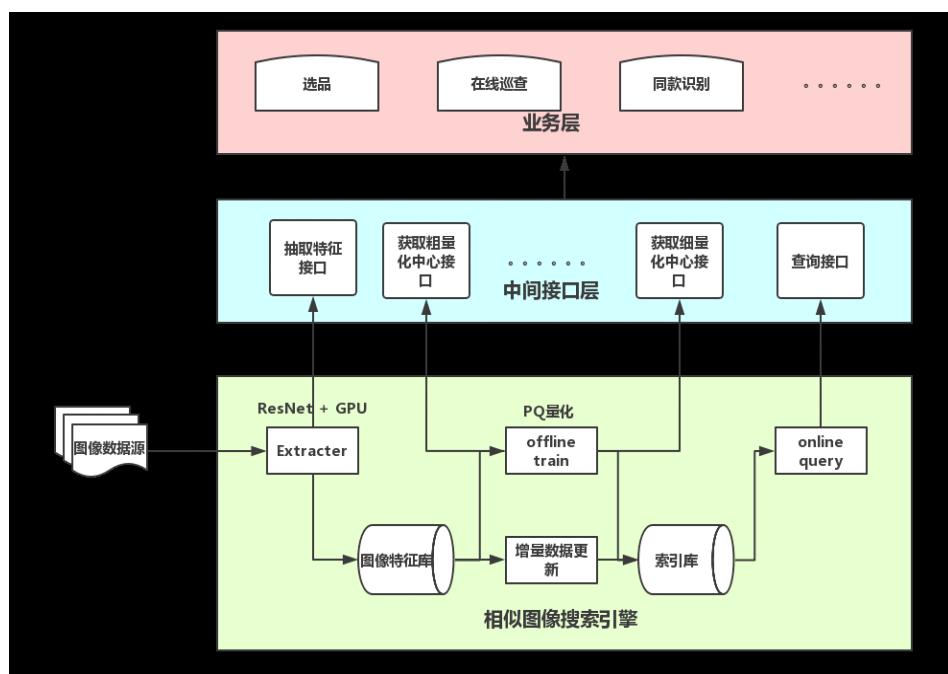


图 2. 图像检索的系统架构

图像检索系统的整体架构如图 2 所示。基于底层的图像搜索算法，通过中间接口层提供给具体的业务使用，提升了相似图像搜索的扩展性，能够快速地响应实际的需求和应用。

## 2.2 应用 1：同款商品识别

电商基础业务中，需要审核商家上传的商品图片。我们在实践中基于相似图像搜索技术，构建了同图识别系统。系统产出了全量图像数据的索引库，基于相似搜索引擎来查询商家图片是否为商品库中的相同图像。根据查询结果，结合业务规则，判断商品是否为同款。图 3 给出了同款商品识别的系统概要图。

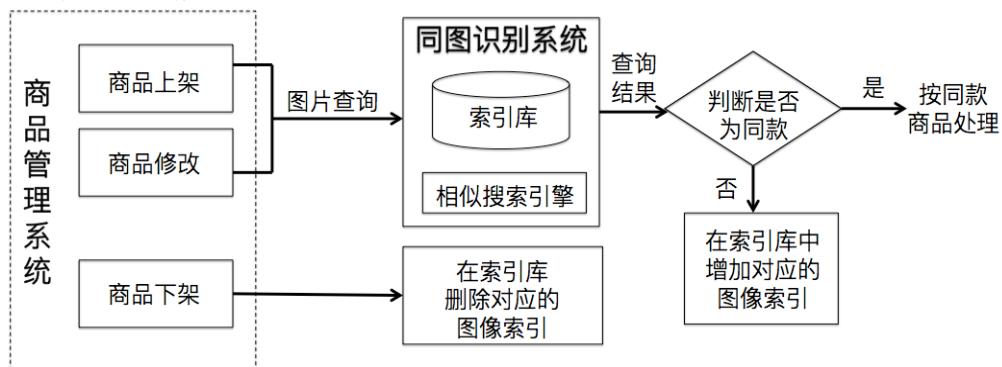


图 3. 结合图像信息的同款商品识别

目前该系统部署在蘑菇街电商平台中，提升了商品管理的效率。在亿级图像索引规模下，系统识别准确率为 99.06%，单张图像查询的整体响应时间为 20ms。

## 2.3 应用 2：移动端搜相似商品

蘑菇街 APP 上提供了搜相似商品的功能。如图 4 所示，其过程是点击单个商品图像右下角的搜索图标，将与该图像相似的同类商品展现给用户。

图 5 展示了系统概要图。在实现过程中，采用了图像搜索技术来承担相似图像查询，从而召回相似商品列表，然后结合业务因素和图像相似

性，进行商品排序。通过该功能，能够提升用户在蘑菇街 APP 上的浏览体验，有利于发现更多相似商品；同时，用户的停留时长也有所增加。

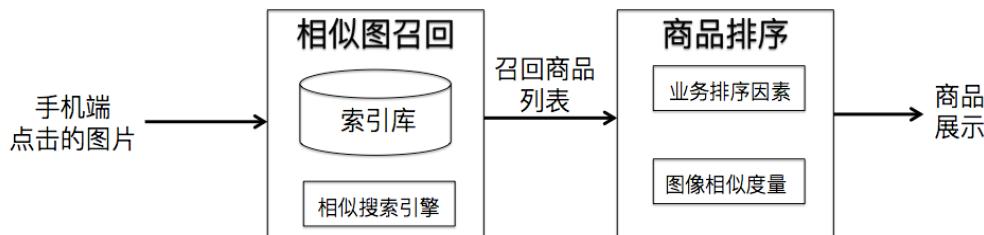


图 4. 搜相似的用户界面，左 (a) 商品展示原图，右 (b) 相似商品列表页面

图 5. 搜相似商品的系统概要图

### 3. 图像标签技术应用

#### 3.1 技术原理介绍

图像标签技术的任务是通过图像算法，自动识别图片内容，如场景、风格、主体名称、颜色、图案等。通常来讲，在电商中对商品图片做图像标签的处理流程如图 6 所示，其中所涉及到的技术模块简述如图6所示。

##### 区域提取

这个阶段利用图像分割技术，将商品主体从背景中分离出来。以服装



**图 6. 图像标签处理流程**

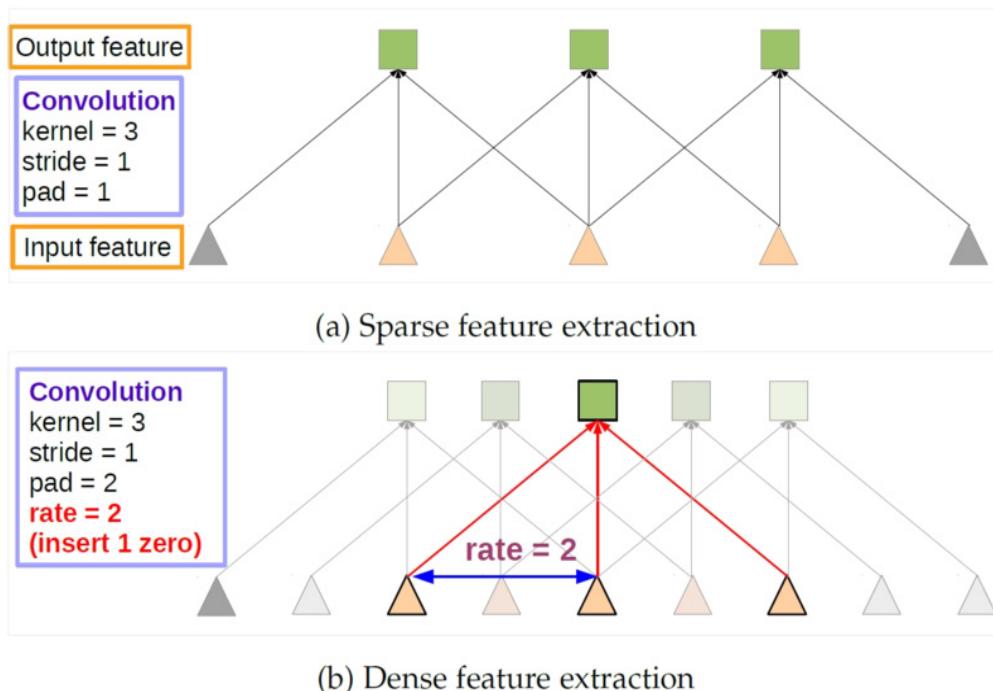
图像为例，区域提取的目标是对图片进行精细化语义分割，排除背景与服装、服装与服装之间的相互干扰。针对服装模特图片，我们通过 Human Parsing 算法，把主要区域提取出来，例如：头肩、上衣、裤子、鞋、包包等。

图像语义分割是图像理解的基础技术，在服饰信息分析、自动驾驶系统（具体为街景识别与理解）、无人机应用（着陆点判断）以及穿戴式设备应用中举足轻重。众所周知，图像是由像素组成，语义分割就是将像素按照图像中表达语义含义的不同进行分组和分割。如图 6 所示，紫色区域表示语义为“上衣”的图像像素区域，荧光蓝代表“下装”的语义区域，军绿色表示“包包”，橙色则表示“鞋子”区域。在图像语义分割任务中，输入为一张  $H \times W \times 3$  的三通道彩色图像，输出则是对应的一个  $H \times W$  矩阵，矩阵的每一个元素表明了原图中对应位置像素所表示的语义类别 (Semantic label)。因此，图像语义分割也称为“图像语义标注”。

在语义分割领域，全卷积网络 (Fully Convolutional Networks, FCN) 推广了原有的 CNN 结构，在不带有全连接层的情况下能进行密集预

测。FCN 使得分割图谱可以生成任意大小的图像，且与图像块分类方法相比提高了处理速度。实际上几乎所有关于语义分割的最新研究都采用了 FCN 结构；不过该框架中的池化层在增大上层卷积核的感受野、聚合背景的同时，却丢弃了部分位置结构信息。

在服装语义分割中，人体结构和服装位置之间的相对关系，对提高分割效果来说至关重要。我们在实际工作中，借鉴了 DeepLab 论文 (DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs. arXiv:1606.00915, 2016) 的空洞卷积 (Atrous Convolution) 思想，保留了池化层中所舍弃的位置结构信息。



**图 7. 图像语义分割中的卷积操作**

丢失的位置结构信息主要由于重复池化和下采样造成，因此我们的网络中移除了最后的若干个最大池化层下采样操作，并对滤波器进行上采样，在非零的滤波器值之间加入空洞，进行空洞卷积。如图 7 所示，(a) 中的常规卷积只能获取到较小感受野上的稀疏特征；(b) 中方法采用空洞

卷积后可以得到较大感受野对应的更丰富特征，对应到服装语义分割也就保留了人体和服装之间的位置结构信息，有助于服装分割效果的提升。

### 属性分类

根据电商业务特点，主要从三个不同的维度来定义标签体系，包括类目、元素、颜色。类目主要描述商品是什么，元素和颜色体现商品有什么样的性质。

我们定义的类目包括服装类、生活类、化妆品，元素范围包括风格、纹理、版型等。以服装为例，标签信息比较复杂，覆盖到服装的多级类目、颜色、领型、衣长等，这导致同一张图存在多个标签。在实际工作中我们聚焦于以下两个方面的属性分类解决方案。

#### 标签的层级关系建模。

电商商品的层级类目结构导致一件商品具有层级化的标签，例如 T 恤：它的一级类目是服装，二级类目是上衣，三级类目是 T 恤，因此一件 T 恤的商品图单类目标签就有 3 个。根据不同标签和相应的数据训练单独获得 3 个模型是一个可行方法，但是系统复杂度过高。

我们采用多级 word tree 结构来实现一个整合多套标签的模型，同时用标签间的关系来约束输出，概率表示为： $P(T\text{ 恤})=P(T\text{ 恤}|\text{上衣})*P(\text{上衣})$ 。根据这个思路设计，用一个模型就能表示出层级关系的多套标签，不仅充分利用了同一批数据，而且通过条件约束之后输出的标签更精准。

#### 标签的多维度特性。

对于同一件商品会有不同维度的信息描述，以 T 恤为例，它有衣长、领型、袖型等维度的信息。因此对于多维度的标签模型，我们采用多任务 (multi-task) 的方法，用一个网络同时提取不同维度的图像特征，能够统一描述图片内容。

在实际系统中，我们利用了百万级别的样本进行模型训练，基础模型的网络结构是残差 18 层网络 (ResNet-18)。业务测试中类目的准确率

是 92.46%，元素分类的准确率是 91.10%。

### 颜色量化分析

用色彩来装饰自身是人类的原始本能，色彩在服饰审美中有着举足轻重的地位，自古至今都是服装三大要素之一，因此服装颜色标签识别是重要部分。通过区域提取过程获得上衣、裤子等服装单品的区域后，对单品图像采用改进的 Mean-Shift 算法进行颜色聚类，得到服装单品的主要颜色占比。

实际应用中，蘑菇街商品图片的颜色会受到拍摄光线和滤镜处理的影响，这给我们的颜色识别带来了挑战，主要表现为两个方面：

**颜色空间的选择：**不同的颜色空间有不同的特点，Lab 色彩空间具有单独的亮度通道，因此为了减小光照对聚类算法的影响、提升颜色聚类的准确度，我们会在 Lab 色彩空间对图片色彩进行一定的矫正，再进行颜色聚类，以尽可能减少光照和滤镜对颜色识别的影响。

**服装色卡定义：**根据蘑菇街服饰颜色分布及商品标签需要，定义了蘑菇街标准的 72 色服装色卡。基于颜色聚类的结果，获取到主体颜色的聚类中心信息，所占比例最大的聚类中心所对应的色卡，被定义为最终输出颜色名称。

## 3.2 应用：商品属性的自动填写

商家在发布新品时，需要填写商品的标题、上传图片，填写商品的属性值，以及详情页信息。当上新量很多的时候，特别是筹备双 11 期间，填写商品信息比较费时，加大了商家的工作量。而图像算法能够在商家上新的环节，通过分析上传的图片，得到图中的关键信息，为商家提供便利。

以服装类目举例，商家上传了商品图片后，我们通过图像标签技术模块，计算得到图中商品的一系列属性信息。例如图 8 所示，这些信息包括：类目（毛呢外套）、袖长（长袖）、版型（收腰）、领型（西装领）、衣长（长款）、风格（韩系）、颜色（藕粉色）等。利用这些信

息，自动帮商家填写好对应的属性，节省了商家选择属性值的时间。当商家发现图像算法识别错误时，可以在自动填写的基础上，对已填写内容进行手动修改。整个流程能够大幅度减少商家上新填写信息所需时间，提升商家的业务效率。

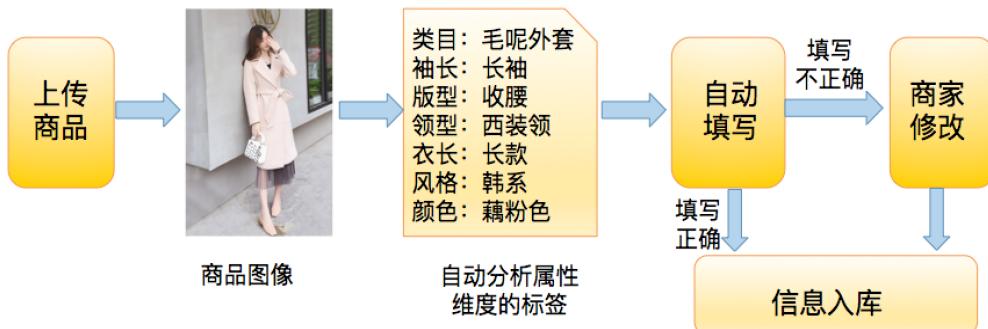


图 8. 商品属性自动填写

## 4. 结语

在实际的业务场景中，图像算法开发是基于应用来驱动的，为保障平台运营和用户体验提供价值。我们的工作，通过图像搜索技术可以自动识别平台上的同款商品，提升后台商品管理的效率；也能够帮助用户发现更多相似商品，改善用户体验。同时，运用图像标签技术，为商家发布新品节省信息填写时间，提升了商家效率。

我们在日常的开发过程中积累图像算法的基础模块，并在双 11 的业务开发中拓展其运用场景；未来将根据业务中的数据变化、场景变化，进行技术迭代开发，从而为不断升级的业务需求提供保障。

致谢：本文工作是蘑菇街图像算法组同学的共同贡献，特此致谢！



# 人工智能在饿了么的应用实践

作者 张浩



## 关于饿了么

大部分人都点过外卖，现在外卖成了中国吃饭的方式，点外卖是什么样的量级，说起来大家吓一跳，中国最大领域是电商，淘宝、京东，其次就是出行行业，滴滴、UBER 紧接是共享单车，这几家公司加起来是一天两三千万订单量左右。在外卖行业大家都知道，这个行业到今天为止已经每天 2500 万单，所以可以想象这个行业在飞速发展。为什么数据和算法起到那么大的作用，因为我们都知道在互联网+ 的情况下，有这么大的单量，至少是在数据行业我们有非常多事情要做。

2500 万单量里面 “饿了么” 是什么样的场景：我们把手机 APP 打开，可以找到自己喜欢的餐厅，大家选择一个餐厅，选择喜欢吃的东西，这个行为虽然是点一个菜，但是实际上跟大家淘宝买衣服，在携程买机票是一样。前面是电商交易平台所交易食品，现在不仅是食品，可以在上面买鲜花，买到药品，同时还有本地的帮买帮送等等，所以电商只是第一部分，电商到了什么规模大家知道吗？，“饿了么” C 端注册用户 2.6 亿，B 端商家目前已经是 130 万，每年是千万级别的定单情况，这个是我们外卖行业的一部分，就是电商交易平台。

### 饿了么简介



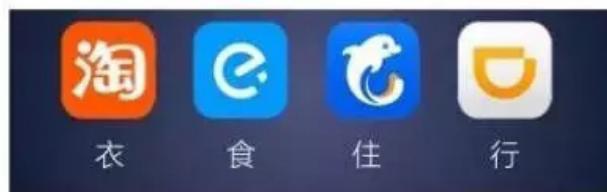
第二部分就是大家也可以看见这张图就是骑手小哥拿着箱子，要么走路，或者骑着电动车。其实这就是本地物流平台，为什么要强调本地，因为我们行业特殊性跟其他物流行业不一样，他们几天时间到达，我们这个行业的本地物流一定是希望 30 分钟送到手里，所以我们设计这个架构的时候就有很大的挑战，这个有一些不同，一会儿讲到算法模型的时候就清楚了，我们是做一些本地的物流，所以时间上有非常严格的限制。

到今天为止的话，我们配送员已经是 300 万，平均每天在任何时刻全国都有 30 万到 40 万的骑手活跃线下，随时准备接单，这个跟

滴滴是一样的运营模式，现在已经覆盖了全国 2 千多个城市。

## 关于 AI @ 饿了么

第二部分是 AI 在 “饿了么” 的应用。这个行业为什么需要人工智能呢？作为本地生活的平台，我们都知道衣食住行是非常需要的。在每个方向都有很多大的商家，他们在技术上的挑战有什么不同，这个一定来自于他们的业务形态。



**淘宝**: 线上以用户/商户为主，线下运单走开放平台，时效性以天计算，超时一般不赔

**携程**: 线上以用户/商户为主，基本没有线下运单

**滴滴**: 线上与线下都以用户/司机为主，运单只有众包模式，超时不惩罚司机，由乘客承担

**饿了么**: 线上以用户/商户为主，线下运单包含自营/团队/众包模式，时效性以分钟计算，超时赔付用户

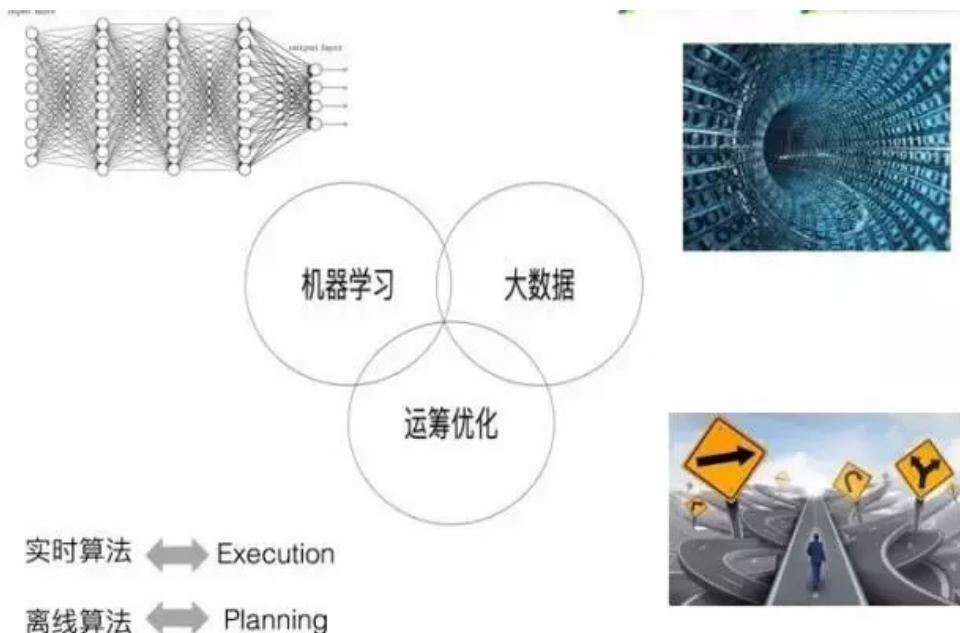
这个跟大家简单过一下：首先是淘宝，我相信大部分都会用淘宝，淘宝是大家在线上买东西最常用的一个平台，里面主要是以用户和商户为主，线下大家知道是同城当日达更多定单是走开放平台，大家下了定单送到大家的手里，这个是三通一达，也可以菜鸟自己，或者顺丰，这个是开放平台，最重要的一点是时效性，通常以天来计算，超时不会有所谓的赔偿，这个是淘宝的情况。

我们看携程，它可以定旅馆，酒店，线上以用户和商户为主，不会有线下订单的。

跟外卖行业特别接近的就是滴滴，其实在国外优步也做的非常好，从业务形态来讲，像滴滴和外卖是非常接近。滴滴线下线上始终

是用户和司机，定单形式也是众包的形式，要么通过加盟商，要么通过司机网上注册来承担运力，超时也不会惩罚司机，因为谁也不会预料到会不会出现车祸，这个时效没有保证，所以说它和饿了么是极其相似的。最后提到“饿了么”和外卖行业。

首先线上以用户和商户为主，线下订单部分比较多，蓝色骑手有一部分是“饿了么”的员工，就是自营，还有团队和加盟商的形式，当然还有一种是众包，比如说今天开一个会，下午还有四个小时，我可以送几单，这是一种众包的形式。时效性以分钟来计算，我们的目标其实在很长时间已经做到了，全国平均半个小时可以把订单送到手里，还有超时赔付，如果30分钟之后超过10分钟没有到，有一个红包的赔偿，超时赔付的压力是比较大的，不过这样对客户来说算是一种服务不足的补偿。



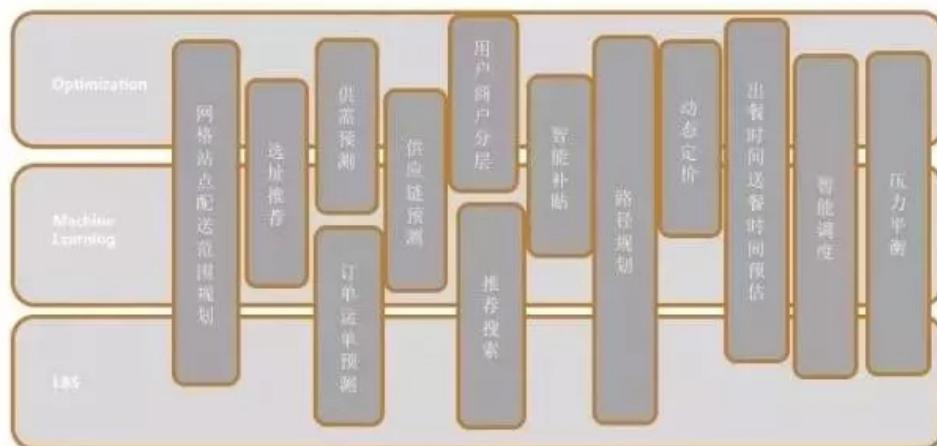
根据上面所述，我们进入了一个大的框架，就是在外卖这个行业是三个大东西，一个是机器学习，运筹优化其实也是跟机器学习密不可分的。讲到其中运筹优化，大数据作为运筹优化的基础起到了非常关键的作用，现在大家看这个图挺有意思，我会多花两分钟讲一讲在业务中的算法问题，大概有三个层面。底层的外卖行业希望30分钟

把食物送到手里，不可能送到二十或者三十公里之外，除非你会飞，否则半个小时不可能送 10 公里。基于这种情况，所有行业都是基于当前打开 APP 定位，定位可能 3 公里或者 5 公里的半径，LBS 保证在运营商做各种推荐或者搜索为基础，再往上两层就是机器学习和优化。所以现在来具体讲一下这三部分。

## 第一：交易

大家可以看一下中间这个模块是用户商户分层，推荐搜索以及智能补贴，这几个大的方向是任何电商都必须做的。

### 饿了么业务中的算法问题



在有很精细的用户画像体制上，我们希望对用户和商户的生命周期做严格的管理，在这个基础上我们做相应的推荐、搜索、补贴，比如说有一个用户进入沉睡期，我们会通过一定的方式对客户进行刺激。

## 第二：线下

当交易行为发生时，我们希望 30 分钟将外卖送到用户手里，这里面涉及到机器学习的规划，我会详细讲智能的调度，也会详细讲到出餐时间和送餐时间的预估，以及动态定价等这几个模块。智能调度

是调度的一部分，我们 30 分钟包括了准备的时间和路上的时间，甚至保证了送到楼下，等电梯到你手上的时间等等，所以 30 分钟有很多不可预估的东西。那么压力平衡是什么意思？大家都知道，线上交易和我们物流是矛盾的，对于线上交易来说我们当然希望订单越多越好，我们希望有上千万的用户一下子几秒钟进来。但是 30 分钟内把订单全部送出去，这个是不太可能一下子解决的问题，为了达到压力平衡，就要保证交易和物流、配送等保持平衡，既达到交易质量，也不损失用户的积极性。

### 第三：底层

讲到这两个之后就是一些底层的东西，现在让我们看上一张图，左边包括选址推荐等等，刚才讲到配送是本地，当一个商家定下准备配送的地方就会画一个圈，比如说我送一个圆圈或者六边形，这个不是随便画的，首先有可能这个地方是高速路或者高架桥，不是每个人平台都是一样，有的用户也有可能老是定便宜订单，我们在网格和站点规划的时候会考虑所有因素，这个涉及很多运筹优化的问题，最后一个例子就会讲到选址和网格规划的问题。简单讲一下我们三个部分包含了我们在人工智能方面所有一些尝试，这对我们业务是非常重要的。

## 关于运筹优化与应用实例

这一部分我会分两种来说，分别是机器学习的应用案例和机器学习应酬优化的案例。

### 案例 1：出餐时间预估

第一个是出餐时间预估，我在这里用滴滴做比较，什么是等待时间预估，比如我们在滴滴场景下下了一个订单，比如说我想去浦东机场，它会告诉你这个车离这里两公里，3 分钟会到，3 分钟就是等待时间预估。“饿了么”相当于下了一个单，大概 20 分钟才能做好，

我希望来的早不如来的巧，作为我平台骑手刚好在 20 分钟就到，如果早了骑手等在那儿是浪费，但是去晚了，就可能订单超过了时间。

### 案例1：出餐时间预估

**饿了么 vs. 滴滴：等待时间预估**

**滴滴：**司机从当前位置前往乘客上车点。一般三公里内车程，时间预估有保障

**饿了么：**餐厅的备餐时间受堂吃人数、餐品类型、烹饪方式、订单大小等因素影响，且备完餐后无通知

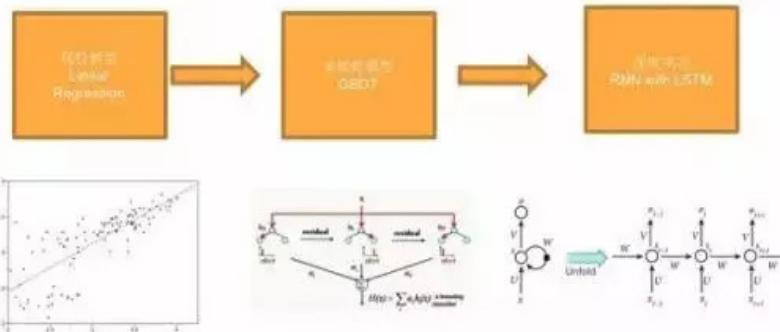
**热卖套餐**

招牌牛腩 (大)	x18
小炒生菜炒饭	x22
关东三鲜豆腐砂锅	x18
山西老陈醋	x24

这个出差时间准确性是关键，当订单完成之后，怎么知道订单花多长时间完成，这个餐厅受很多的因素影响，餐厅的备餐时间和食堂吃的用户数，餐品类型，烹饪方式，订单大小等因素影响，且备完餐后无通知，比如说餐厅客户特别多，平时可能 5 分钟做出来，可能人多了就做不出来，还有产品品类的问题，甚至包括一天的天气各种原因，包括餐厅的出勤率，餐厅厨师请假突然少了几个人，这些都是造成预估不准确的原因之一。我们想过为什么不让餐厅做好了告诉我们，我们就去，这个理论是可行，大家想象一下在餐厅场景里面，厨房是什么样的情况，你想象一个厨师满手都是油，出来点一下这个订单好了，下一个订单了，这个是很难想象的事情，我们没有得到这方面的数据。这个是一个前提，我们的解决方案毫无疑问是机器学习，最简单版本就是线性模型，一开始效果不是特别好，逐渐演进到后面用了的 GBDT，大家都比较熟悉下，在场景下做到平均不是特殊平均，加上出餐时间是 10 分钟，我们可以固定 7 分钟到 13 分钟，这个准确率比较高。

我强调是平均，因为有很多特殊场景，如果厨师出了什么事情，我们也不知道，因为机器学习只能根据过去的事情来预测将来。在突发事件有

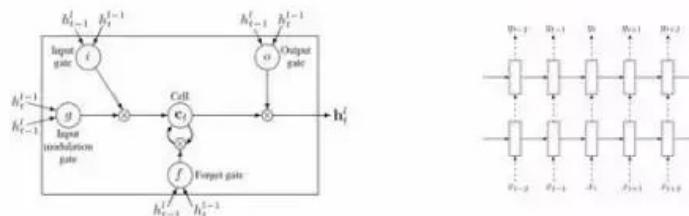
## 出餐时间预估模型演进



一些产品的方案，比如说看到这个餐厅出餐量和订单量并没有呈线性的增长，前面出现了堵塞情况，我们根据数据对平台进行实时调整。

最后现在用的方法是深度学习，我们是用 LSTM，右边这图大家可以看一下这个文章。我们通过时间相关性把预测做的更加准确，毫无

RNN with LSTM Cells



$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \text{sigm} \\ \text{sigm} \\ \text{sigm} \\ \tanh \end{pmatrix} T_{2n,4n} \left( \mathbf{D}(h_t^{l-1}) \right)$$

$$c_t^l = f \odot c_{t-1}^l + i \odot g$$

$$h_t^l = o \odot \tanh(c_t^l)$$

2 layers, 1500 units per layer

65% dropout non-recurrent connections

Batch size 128

Gradient Clip: 10, normalized by batch size

疑问出餐时间一定会跟过去订单有关系，这个不用解释，但是为什么跟未来有关系，我们预估未来 3 到 5 分钟有新的订单，但是跟现有的订单有共同之处，有可能是同样的菜品，有可能是共同的地方，同样的菜品对厨房是一个订单，可以把菜一起做，我们学到了，通过这个模型也可以捕捉这些特征，对订单分配有一定的帮助，同样对订单

打包也有一定的帮助。

## 案例 2：行程时间预估

行程时间的预估就是这样，当订单完成了以后，骑手把订单拿到手里，他会跑到办公室或者家里也好，这个是行程时间的预估，滴滴从 A 点到 B 点，交通方式肯定就是车，而且有大量的地图数据，像高德或者谷歌地图或者百度地图，这些数据会实时上传给服务器。

在这种交通情况预测已经是比较准确，相对“饿了么”场景远远没有那么多的信息，首先骑手是有可能步行，有可能走电梯，或者走上下楼



梯，或者骑电动车，或者换交通工具，这个直接造成了我们在数据搜集是极不准确的，还有一点提到了在楼宇内的交通复杂，这个数据很难获取。我们上班的时候，餐厅和顾客都是在大楼里面，大楼里面没有 GPS 信号或者不太好，我们收到的数据或者定位误差高达几百米。

所以提前时间预估，我们需要把轨迹建立起来，因为时间预估在高德地图或者腾讯百度地图是基于历史数据的，我们第一步做的是历史数据清洗，室内定位不准甚至完全缺失，这个情况下我们想了各种各样的办法，我们用 WIFI 信号，GPS 信号，或者大家互相定位最大程度减少定位缺失的问题，其次即使定位有了，它的位点也是有 GPS 轨迹，也是有很多的噪音，所以需要去噪音，所以需要去噪。我们通

## 行程时间预估与轨迹网络



- 室内定位不准甚至完全缺失
- 骑手位点数据去噪
- 聚类算法找到O点和D点
- 轨迹聚类
- 恶劣天气、活动、高峰、周末与节假日

过定位的算法把相关的时间，把 O 点和 D 点合起来，最后进行轨迹聚类。

### 案例 3：智能分单

滴滴与我们的分单难度不一样，滴滴场景下要配一个司机，最多接两三单。在“饿了么”一个骑手一个包同时背 5 到 10 单，而且订单之间有时间限制，有涉及大量的时效要求。

我讲两个方案，第一个方案是路径规划的问题，很传统的 VRP。

当你给一个订单，在骑手容量和成本固定的情况下，我们需要找到匹配的线路，每个订单承诺时间是不一样的，就是不能超时。默认



饿了么 vs. 滴滴：行程时间预估

滴滴：从A点到B点的交通方式已经确定，且有大量的道路数据，行程时间较为容易估计

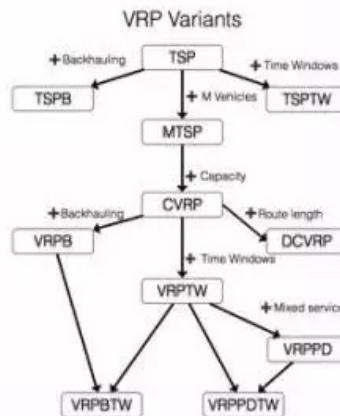
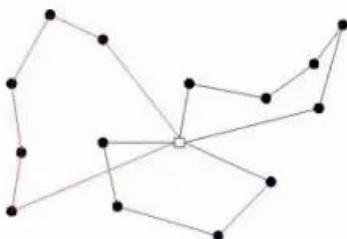
饿了么：骑手步行、等电梯、上下楼梯、电动车等多种交通方式复合进行，且楼宇内交通复杂，数据不易获取



模式下，一个骑手可以同时送 5 到 10 单，每单都有严格时效要求，并且订单在午高峰爆发式增加。

#### 方案1.0:车辆路径规划 (Vehicle Routing Problem, VRP)

输入: 订单, 骑手, 容量, 成本  
输出: 订单和骑手间的匹配以及行走路线  
优化目标: 最小化时间或者行驶距离  
约束条件: 骑手背单数、骑手数量、最晚到达时间等



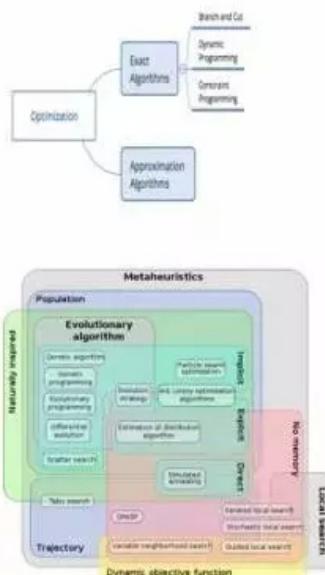
#### 方案 1 就是车辆路径规划

- 输入: 订单, 骑手, 容量, 成本。
- 输出: 订单和骑手间的匹配以及行走路线;
- 优化目标: 最小化时间或者行驶距离;
- 约束条件: 骑手背单数、骑手数量、最晚到达时间等。

我们用了模拟退火算法，模拟退火算法（Simulated Annealing, SA）是基于 Monte-Carlo 迭代求解策略的一种随机寻优算法，通过赋予搜索过程一种时变且最终趋于零的概率突跳性，从而可有效避免陷入局部极小并最终趋于全局最优的串行结构的优化算法。从理论上来说这算法具有概率的全局优化性能，目前已在工程中得到了广泛应用，诸如 VLSI，生产调度、控制工程、机器学习、神经网络、信号处理等领域就是用它来做订单的分配，但是最后结果不是特别好，因为时间预估存在不准备性，在路径规划的时候，先走 A 单还是 B 单，在时间一旦出现误差的情况下，这个路径规划会非常差。

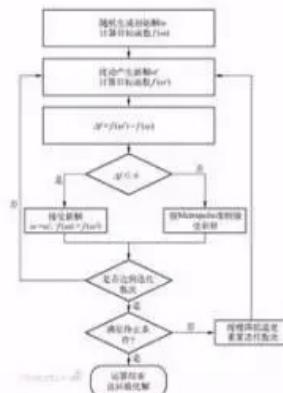
最后用的是第二个算法，也是一种基于大量函数的组合算法。

左下角是个矩阵，每一行是一个订单，每一列是一个骑手，我们



## 模拟退火算法

模拟退火算法(Simulated Annealing, SA)是基于Monte-Carlo迭代求解策略的一种随机寻优算法，通过赋予搜索过程一种时变且最终趋于零的概率突跳性，从而可有效避免陷入局部极小并最终趋于全局最优的串行结构的优化算法。理论上算法具有概率的全局优化性能，目前已在工程中得到了广泛应用，诸如VLSI、生产调度、控制工程、机器学习、神经网络、信号处理等领域。



希望通过一些规则和一些机器学习的算法算出来，右边是一个定单匹配的结果。

## 方案2.0:基于代价函数的优化问题

## VRP方案遇到的挑战:

- 骑手的送餐习惯
- 路径规划的不规则性
- **时间预估的不准确性: 出餐时间, 行程时间**
- 基础数据的准确性

最优匹配就是 KM 算法。调度算法的演进最早是 VRP。

后来是 KM 算法，但是这个基础框架界定了以后，还有很多工作量需要做。订单实际上有相似性，因为订单是可以打包的，一个人稍微等几分钟，也许这个订单出来跟那个订单很相似的性质，就是去同一个地方，就可以把订单给同一个人拿走。所以订单打包和吸水是我们做的第一件事情，但是订单靠什么规则在高峰期和非高峰期的时候是不一样的，存在两个方向的路和两个方向的夹角不一样的地方，所以定单匹配模型是在 2.2 版本之上做出来的，用机器学习通过历史



输入:  $N$ 个订单,  $N$ 个骑手  
输出: 订单包和骑手间的匹配  
优化目标: 最小化代价

	$O_1$	...	$O_j$	...	$O_n$
$O_1$	0.4	0.6	0.4	0.6	0.6
...	0.6	0.6	0.6	0.6	0.6
$O_j$	0.4	0.6	0.6	0.6	0.6
...	0.6	0.6	0.6	0.6	0.6
$O_n$	0.4	0.6	0.6	0.4	0.6

输入代价矩阵  $P$

	$D_1$	...	$D_j$	...	$D_n$
$D_1$	0	1	0	0	0
...	1	0	0	0	0
$D_j$	0	0	1	0	0
...	0	0	0	1	0
$D_n$	0	0	0	0	1

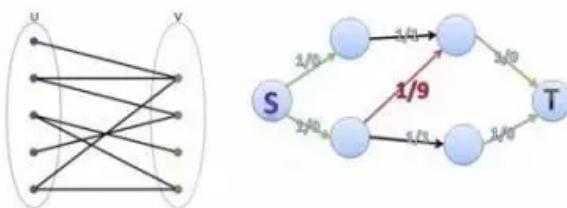
订单匹配矩阵  $X$

### 最优匹配-KM算法

KM算法求的是完备匹配下的最大权匹配：在一个二分图内，左顶点为 $X$ ，右顶点为 $Y$ ，现对于每组左右连接 $X_iY_j$ 有权 $W_{ij}$ ，求一种匹配使得所有 $W_{ij}$ 的和最大。

Kuhn-Munkres算法流程：

- (1) 初始化可行顶标的值；
- (2) 用匈牙利算法寻找完备匹配；
- (3) 若未找到完备匹配则修改可行顶标的值；
- (4) 重复(2) (3) 直到找到相等子图的完备匹配为止。

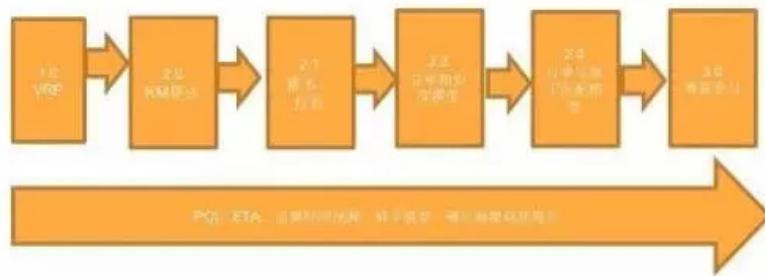


数据来训练，在这里我们也碰到一些挑战，由于在不同的站点配送员习惯不一样，我们推广的时候会遇到一个问题，在 A 站点大家觉得是 OK，但是在 B 站点不行，我们现在做到千站千面的东西，根据类似的站点历史过去分担一些情况，我们把这些模型用来做训练，做到类似的站点它有类似分单的方式。所以不会出现说你特别不喜欢这个分单的方式，多多少少有一点类似性，所以做到了 2.3 这个版本。现在做的版本就是增强学习，我们根据实时的情况来进行动态地调整。

餐厅选址就不详细讲了，我们自己其实也和商家开始合作开一些餐厅，我们都希望选最好的地方，餐厅覆盖最多的用户，菜品不一

样，用户群不一样，所以这个选址是很重要的。

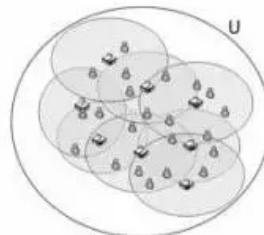
## 总结



我今天主题是应用实践，我个人感受，我做机器学习做了十几年，工作挑战是来自于基础数据的完整性和准确性。刚才讲到数据不准确，餐厅不规则的情况，我们无法知道一些准确的情况，我们花了

寻找一个区域内最优的餐厅选址，最小化选址的固定成本、最大化潜在的**GMV**（考虑商圈附近的用户群特征）  
**facility location problem (FLP)**

$$\begin{aligned}
 & \min \sum_{i \in \mathcal{F}, j \in \mathcal{C}} c_{i,j} x_{i,j} + \sum_{i \in \mathcal{F}} f_i y_i \\
 \text{s.t. } & \sum_{i \in \mathcal{F}} x_{i,j} = 1 \quad \forall j \in \mathcal{C} \\
 & x_{i,j} - y_i \leq 0 \quad \forall i \in \mathcal{F}, j \in \mathcal{C} \\
 & x_{i,j}, y_i \geq 0 \quad \forall i \in \mathcal{F}, j \in \mathcal{C}
 \end{aligned}$$



大量时间来做基础数据的调整。第二点我讲到算法的提升和对人的行为的理解比较重要，因为在外卖行业都需要人去执行，以前人工分配通过打电话，有大量沟通在里面，现在机器一下子分摊了，他们难以理解，而且机器考虑全局最优而不是局部最优，人是做不到这点。在算法提升和产品运营综合起来，才能把这个事情最后推下去让大家形成习惯。第三点优化算法与机器学习在我们行业是相辅相成的，不仅

## 总结

本地生活场景有很多的算法挑战

1. 基础数据的完整性和准确性
2. 算法的提升和对人的行为的理解
3. 优化算法与机器学习的结合

是机器学习，跟重要的是我们在这么短时间怎么样把人力分布最好，在最少的时间情况下把订单完成。谢谢大家！

## 作者介绍

**张浩**，饿了么技术副总裁，负责人工智能与大数据建设。带领团队将机器学习应用在物流调度、压力平衡、美食推荐等场景，通过数据挖掘建立完整的数据运营体系，用数据和智能驱动业务发展。拥有十余年机器学习、数据挖掘、分布式经验，曾任滴滴研究院高级总监、美国 Uber 大数据部、LinkedIn 搜索与分析部、Microsoft Bing 语音组等从事机器学习与大数据工作。

# 准确率 99%！基于深度学习的二进制恶意样本检测

作者 吴睿



瀚思科技成立于 2014 年，按行业划分我们是一家安全公司。但和大家熟知的卖杀毒软件或者防火墙的传统安全公司不同。瀚思科技帮助各种中大型企业搭建安全大数据的分析平台，平台上应用的安全分析策略深度结合了多种机器学习算法，最终帮助企业定位与揭示各种安全问题。所以我们自己定位是一家安全 + 大数据 +AI 的公司。

在瀚思科技首席科学家万晓川的带领下，瀚思算法部门紧追时下 AI 领域最前沿的技术突破，并尝试应用在安全领域当中。今天是主题月的第三讲，接下来我会为大家分享我们是如何利用深度学习来做二进制恶意样本检测的。

## Agenda

1. 病毒样本检测的各种技术
2. 为何用深度学习
3. 怎么用深度学习
4. 经验教训和下一步规划

agenda 如上图所示，考虑我们主要的观众是大数据和 AI 方向的，我会首先介绍一下病毒检测的技术沿革，各种技术的优劣和取舍。然后我会说明为什么我们认为深度学习可以很好的应用于病毒检测。具体我们是怎么应用深度学习的，用的什么网络，中间有什么技巧。最后是实际操作后，我们得到的一些经验教训，以及下一步的发展规划。

### 病毒样本检测的各种技术

- 按数据源分为：基于内容、基于行为
- 对检测方法分为：基于特征码、基于规则、基于算法
- 常见的组合例子：
  - 内容 + 特征码 = 传统病毒引擎
  - 行为 + 规则 = 沙箱
  - ( 网络 ) 行为 + 算法 = NTA

如图所示，我们可以笼统的把病毒检测的各种技术根据两个不同的维度来进行区分。一是根据数据源可以分为基于内容和基于行为两种。二是根据检测方法分为：基于特征码、基于规则、基于算法。

基于内容的就是一般所谓的静态分析，病毒样本不需要实际执行起来。安全人员直接打开文件查看二进制码或者反汇编后来分析源代码都算静态分析。熟悉安全领域的朋友会注意到这里有个加壳的问题，加壳的样本，尤其是复杂的壳，要做反汇编其实不容易。内容结合特征码就是传统的病毒引擎的原理，我们日常在都在使用。依赖

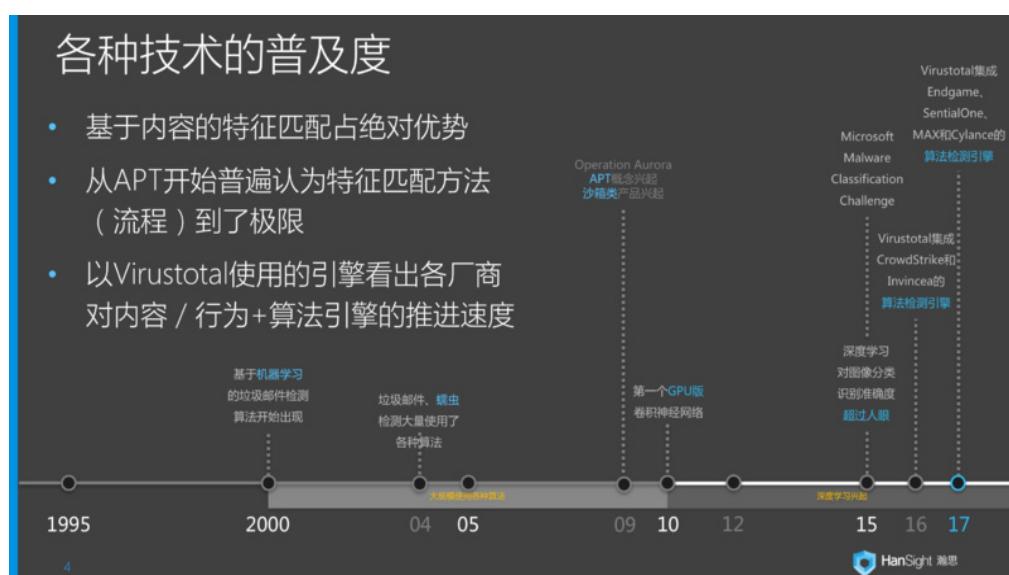
安全人员给出精准匹配的特征码，匹配迅速，但是就如我们的杀毒引擎一样，需要定期更新病毒库。

基于行为的就是一般所谓的动态分析，需要在实际或者通过虚拟化的方法把病毒样本执行起来。通过考察样本对操作系统各种资源的操作来构建特征和分析。某些大病毒家族，比如勒索软件，因为操作高调，通过动态分析非常好识别。

基于规则的方法在静态分析领域用得很少，算是补充。但是在沙箱领域非常常见，因为病毒行为很容易写出规则来。但实际的情况是病毒在沙箱内运行的时间短，最多就是 30 分钟，往往 10 分钟都不到，导致其行为暴露不够充分。

运用分类算法基于行为特征来检测看似不错，但是行为特征少是一个明显的缺陷。所以很多时候往往是混合了动态的和静态的特征来构建。

运用网络行为加算法来分析就是目前比较火也很有前途的 NTA (Network Traffic Analytics)。NTA 既融合了传统规则，也结合了机器学习，通过监测网络的流量、连接和对象来识别恶意样本产生的行为。再配合上质量好的威胁情报，能产生高信息熵的特征，特别适合 Botnet 这一类的病毒。



在历史沿革和普及度上，我们需要知道在业界中，基于特征码（也就是内容特征匹配）的技术仍然占有压倒性优势。统计发现，只要能拿到对应的病毒样本，各大杀毒引擎厂家都能在 3 小时左右部署特征码。但这个工业流程到达极限以后，厂商在后台拿到新样本的速度很难再提高（APT 也难拿到样本），要么是拿到了也处理不过来（新病毒的产生速度一直在提高，有报告统计出每 4.2 秒就有一个新病毒产生）。一般我们普遍认为，在 APT 开始大量出现后，特征匹配的方法（流程）到了极限。

如上图时间轴，考察 Virustotal 中集成的新引擎，可以看出各个厂商应用内容 / 行为 + 算法模式的推进速度。安全厂商普遍使用机器学习算法是从 2000 年开始的垃圾邮件检测，从最初的简单贝叶斯到 04 年的 SVM。我们安全行业采用算法的速度并不落后于其他行业，衡量标准是安全厂商的数量。而学术界更早就有用算法解决安全问题的文章。前面提到 09 年后 APT 兴起，伴随着是沙箱类产品的兴起。2010 年第一个 GPU 版的卷积神经网络出现，在图像识别领域飞速发展，网络变得越来越深，越来越大。到 15 年，基于深度学习的图像分类识别，其准确度已经超越了人眼。同年，微软开办了第一届恶意软件分类大赛（Microsoft Malware Classification Challenge）。虽然当年拿下第一名的队伍后来被爆出投机取巧了（后面还会讲到细节），但是 MS 的竞赛让很多厂商学到了实际的样本分类算法应该怎么做。到 16, 17 年的时候，Virustotal 已经集成了诸如 CrowdStrike、Invincea、Endgame 之类的基于算法的检测引擎了。

我们再快速做个对比。基于行为的也好，基于内容的也好，方法本身并无明显的优劣，但是部署场景有着明显的区别。瀚思做的是 2B 端，所以如果做行为分析，只能依赖于把样本放入沙箱里跑，没法从已经感染了病毒的一个客户实际的电脑上收集到病毒实际的行为。同时现在反沙箱技术实在是过于普及，倚赖沙箱技术会导致病毒检测率

## 行为+规则 / 算法 vs. 内容+算法

- 方法本身并无明显优劣之分
- 部署场景有明确差别：
  - 行为：需要依赖沙箱、或者终端等能捕抓到行为的机制。但反沙箱技术日益流行。能覆盖各种样本。
  - 内容：看到信息没有行为全面，容易被加壳技术干扰。不支持非文件类型的样本。
- HanSight因为没有终端产品线，加上做沙箱出身了解其弱点，所以选择了内容+算法的机制。

的天花板很低。如果做内容分析，又容易被加壳技术干扰。对应非文件类型的样本，如寄存在内存中的木马等无法支持。考虑到瀚思没有终端检测的产品线，加上技术团队做沙箱出身了解其弱点，最终选择了内容 + 算法的路线。另外 2B 还有个优势是能看到样本的额外信息（安全设备日志），比如是从那个网站下载的，是哪封邮件的附件，这些可以帮助我们降低误报率。

## 为啥用深度学习

- 效果足够好：很多领域效果远远超过非深度学习算法
- 样本性质适合：深度学习长于处理单一类型数据
- 样本量够多：更多样本 = 更好深度学习检测率
- 避免人工特征选取，只需设计好网络拓扑
  
- 但是，如何转换样本文件变成深度学习擅长的图像、语音、文字领域？
- 具体选用深度学习的CNN、RNN、GAN、强化学习还是其他？

第二部分我们讲讲为什么深度学习适用于病毒样本分类的问题。

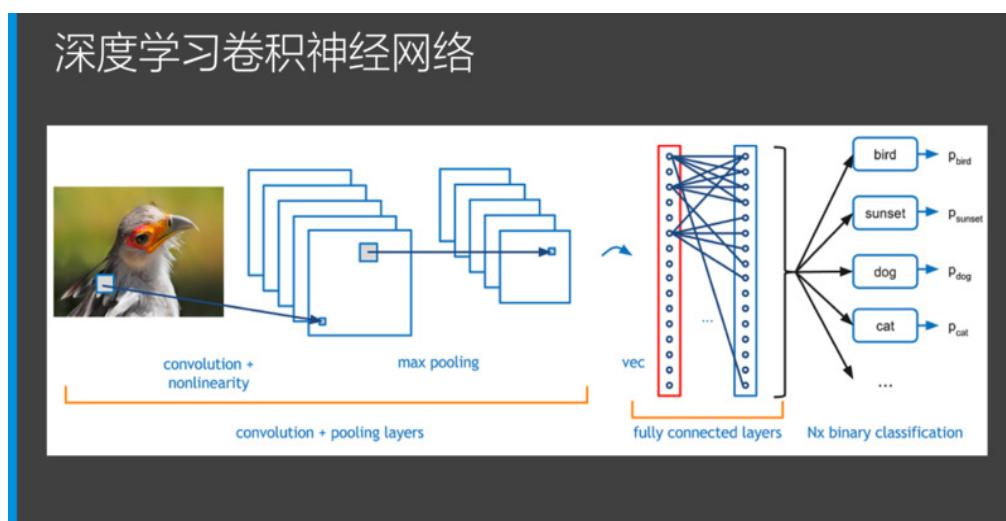
首先是效果足够好，在图像识别、语音识别、机器翻译等领域的效果都远远超过非深度学习的算法。第二是样本的性质合适，深度学习就是擅长处理单一类型的数据。对应我们的输入都是二进制文件。

第三要求足够的样本，而样本越多准确的也越高。病毒的样本呢，根据 Symantec 在今年 Q2 的统计，单日样本可以达到 300 万。第四，可以避免人工去选择特征，只要一开始就设计好网络结构，深度学习会自动学习到重要的特征。再对比静态分析，比如用 N-gram，特征数量会轻松突破百万，再乘以上面的样本数，机器学习是必然选择。

总得来说，深度学习的优点（超多特征，无需人工特征，样本越多越好）都非常适合二进制病毒样本分类这个领域。目前我们也尝试应用深度学习到其他安全场景中（基于 NLP 的智能运维），但因为安全系统属于机器学习的下游，一般安全领域不容易直接产生对机器学习新思路突破，经常是借鉴上游的突破（如图像识别），所以还有很多安全领域等待我们去应用新的机器学习算法。

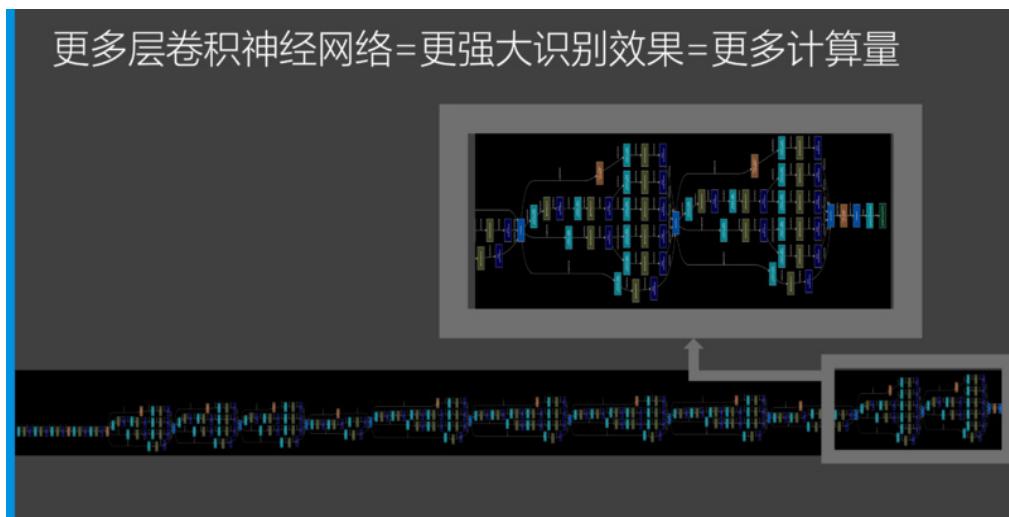
唯一的缺点就是深度学习目前还相对黑盒，研发人员和客户都还不容易理解，相关人员也不好招聘。

那么问题来了，我们的输入是文件，深度学习的输入是图像、语言、文字。中间如何转化？选取什么样的网络结构？



什么是深度学习？2012 年的 ImageNet 大赛中，Alex Krizhevsky 凭借 AlexNet 一举把分类误差的记录从 26% 降到了

15%，震惊了世界。而 AlexNet 的核心就是上图中的卷积神经网络。网络的输入是图像，输出是该图像对应的分类。顾名思义，卷积神经网络包含多个卷积池化单元，其中包括了应用卷积核的卷积层，进一步降维的池化层。做完特征检测后，接入全连接层，观察输出特征与哪一分类最为接近。最后通过一个分类器得到分类的结果。

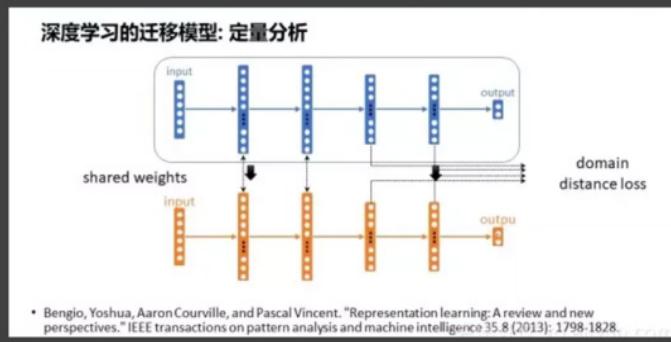


一般笼统的说法，越深的网络就代表更好的分类结果。但目前最新的一些网络模型都参考了 resnet 的理念，也就是很多个少数层的 block 组合而成，理念类似 ensemble method，看 block 的构成和 block 的组合。对于这个样本分类来说，我们前后换过几个 resnet 变种，目前部署的模型是基于 inception-v4，但目前正在实验 densenet，因为同等检测率下，参数更少计算更快。在客户环境下部署，我们希望尽量降低对硬件的需求。从我们实验的效果看，从早的 resnet 到新的 inception-v4，迁移学习后差别只在 5% 内。所以今后我们研究的重点在速度了，网络选择是第二位。

之前我们提了一个问题，我们的输入是二进制样本，而卷积神经网络的输入是图像，怎么办？其实这个问题有两层含义，第一，我们需要把二进制样本通过一种方法转换为图像。第二，我们需要借助图像分类的经验甚至是特征模型来帮助我们做病毒的分类。

## 卷积网络迁移学习

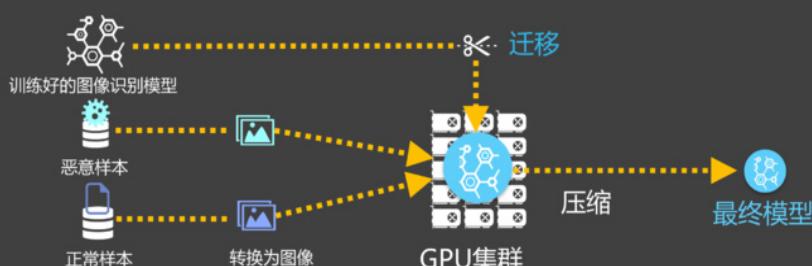
- 将从一个环境中学到的知识用来帮助新环境中的学习任务



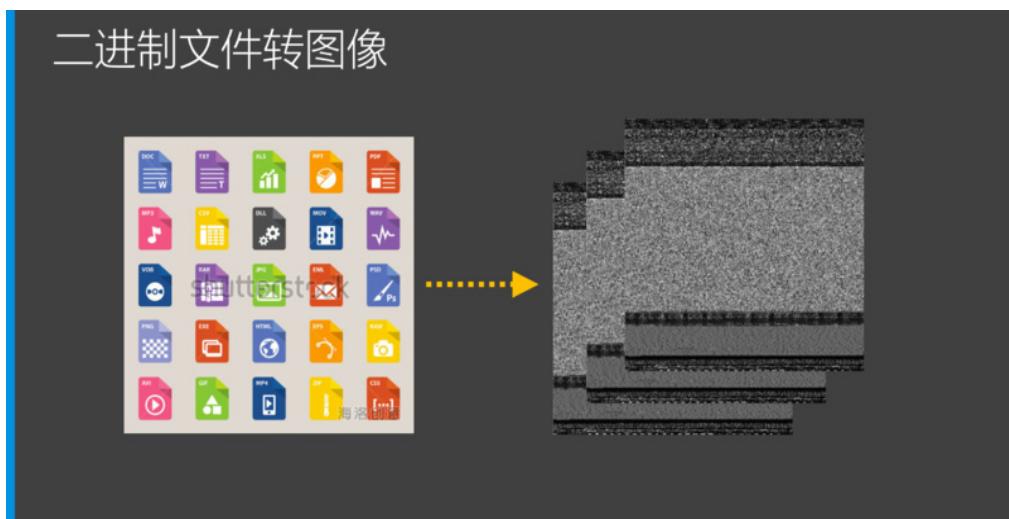
那么具体讲解我们的工作流之前，我们先引入一个概念：迁移学习。

在今年的 CCAI 上，香港科技大学教授杨强又跟大家分享了什么是深度学习的迁移学习，这里借用他一张 ppt。现在我们有两个近似领域 A 和 B，通过深度学习自动提取特征以后，我们发现两个领域在浅层网络中的低级特征其实是可以共享的，而在深层网络中因为领域不同而对应了不同的高级特征。一种普遍的情况是领域 A 有大量的标记数据，有优良的特征模型已经生成。而领域 B 数据量比较小。那么通过共享特征，我们可以应用迁移学习将适用于大数据的领域模型 A 借用过来，再通过领域 B 的标签数据去训练高级特征和分类，从而在领域 B 上实现更好的分类效果。

## 流程



所以我们的流程如图所示。将正负样本按 1: 1 的比例转换为图像。将 ImageNet 中训练好的图像分类模型作为迁移学习的输入。在 GPU 集群中进行训练。我们同时训练了标准模型和压缩模型，对应不同的客户需求（有无 GPU 环境）。



流程中比较核心的算法其实在文件到图像的转换。因为常规的网络一般能输入的尺寸也就是 300 x 300 上下，也就是 9K 左右的规模。而病毒样本的大小平均接近 1M，是远远大于这个尺寸。图像领域的常规转换方法就是缩放，或者用 pyramid pooling。这两者我们实

## 怎么处理加壳？

- 和原先预想的不一样：有些类型的壳对文件进行转换前后有一定对应关系
- 特别复杂的壳没有这种对应关系，深度学习记住的是壳本身的特质
- 解决方法：
  - 人工对样本进行加壳产生新样本
  - 复杂壳使用非深度学习方法过滤，比如威胁情报

验效果都非常低差，AUC 在 0.6 左右。所以来我们又设计了一个很复杂的 pooling 算法处理大尺寸文件。

前面我们介绍过，加壳的样本会对基于内容的分析造成影响。实验中我们发现文件 A 和 B 经过加壳后转图像。肉眼看过去，A 和 B 的相识度会比以前更高。就好比 PS 中马赛克一张猫和一张狗的图片，马赛克强度越高，两图片处理后看起来越相似。但马赛克（加壳）强度低的话，其实处理后的图片和处理前的图片有一定的高纬度映射关系（加壳前后有对应关系）。这种关系，实验看起来深度学习网络能够分辨。当然强度大就无能为力了。

假设算法目前是分辨猫狗照片，不管有没有马赛克。但训练照片中，只有猫照片有马赛克处理，狗没有。所以人工产生狗马赛克图片（人工对样本加壳），让算法有更强分辨能力，至少是对弱马赛克后的图片。

## 迁移学习双模型

- Inception-v4：默认模型，更高检测率
- SqueezeNet：压缩模型，部署给不能安装GPU的客户
- 两模型迁移时候都去掉了最顶和最底几层

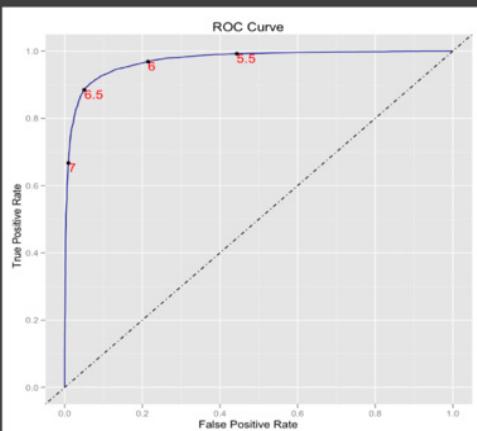
高强度马赛克的话，算法只能记住高强度马赛克后的特征，只是如果有额外信息，比如图片出现在邮件正文有猫或者狗的字样，就能辅助我们判断（引入其他信息）。

流程中提过我们训练了双模型，一个的 Inception-V4，一个是 SqueezeNet。Inception-V4 是目前较为先进的模型，有最好的实验结果，训练和 inference 的速度也可以接受。而 SqueezeNet 是压缩模型，参数数量只有 AlexNet 的  $1/50$ ，虽然准确度稍差，但检测速度快很多，专为不能提供 GPU 环境的客户设计。另外针对这两个模型做迁移学习的时候，我们都替换掉了最顶和最低几层。

简单说一下测试结果，AUC 可以达到 0.985，误报率小于  $1/1000$ ，检测的速度目前可以达到 150M/ 天。

## 测试结果

- AUC = 0.985
- 检测速率：百万/天



## 经验教训

- 深度学习卷积神经网络能取得和沙箱类似的检测效果
- 检测模型具有更好通用型，不更新仍能发挥效用
- 更多数据=更好效果
- 混合其它方法降低误报率
- 不能完全把深度学习当成黑箱，需要分析其机制

再与各位分享一些经验教训。上面提到检测率和误报率已经同等于沙箱检测的水平。而基于二进制文件的深度学习无需沙箱环境（无需在客户处部署沙箱）。深度学习模型记住的是病毒二进制文件中的有效特征，而不是特征码（特征码由专家选取，对应唯一病毒样本），所以具有更好的通用性。在实际测试中，即使一个月不更新模型，对新衍生的病毒样本也有较高的识别能力。

测试发现，对不同规模的样本进行测试。更大的数据集，有更高的准确度。这个深度学习本身的性质是一致的。再者不能完全把深度学习当成黑箱，而是需要分析其机制，至少要观察哪些样本的哪些特征比重较大。

今年 5 月爆发的 WannaCry 席卷了全球 90 多个国家，造成

## 早期WannaCry样本VirusTotal扫描结果

Engine	Signature	Version	Update				
Ad-Aware		3.0.3.794	20170210	Jiangmin	-	16.0.100	20170210
AegisLab		4.2	20170210	K7AntiVirus	-	9.250.22376	20170210
AhnLab-V3		3.8.3.16811	20170210	K7GW	-	9.251.22386	20170210
ALYac		1.0.19	20170210	Kaspersky	-	15.0.1.13	20170210
Anti-AVL		1.0.0.1	20170210	Kingssoft	-	2013.8.14.323	20170210
Arcabit		1.0.0.795	20170210	Malwarebytes	-	2.1.1.115	20170210
Avast		8.0.1489.320	20170210	McAfee	-	6.0.6.653	20170210
AVG		16.0.0.4756	20170210	McAfee GW-Edition	-	8.200.1.100	20170210
Avira		8.3.3.4	20170210	Microsoft	-	1.1.13407.0	20170210
AVWare		1.5.0.42	20170210	MicroWorld-eScan	-	12.0.250.0	20170210
Baidu		1.0.0.2	20170210	NANO-Antivirus	-	1.0.70.15039	20170210
BitDefender		7.2	20170210	nProtect	-	2017.02-	20170210
Bkav		1.3.0.8471	20170210	Panda	-	10.02	20170210
CAT-QuickHeal		14.00	20170210	Qihoo-360	-	4.6.4.2	20170209
ClamAV		0.99.2.0	20170210	Rising	-	1.0.0.1120	20170210
CMC		1.1.0.977	20170210	Sophos	-	28.0.0.1	20170210
Comodo		26569	20170210	SUPERAntiSpyware	-	4.98.0	20170210
CrowdStrike	malicious_confidence_68% (D)	1.0	20170210	Symantec	Ransom.Locky	5.6.0.1032	20170210
Cynet		5.4.16.7	20170210	Tencent	-	1.2.0.0	20170210
DrWeb		7.0.27.12160	20170210	TheHacker	-	1.0.0.1	20170210
Emsisoft		4.0.0.834	20170210	TotalDefense	-	6.8.0.5.1282	20170209
Endgame	malicious (moderate confidence)	0.0.2	20170208	TrendMicro	-	37.1.62.1	20170210
ESET-NOD32		14914	20170210	TrendMicro-HouseCall	-	9.740.0.1012	20170210
F-Prot		4.7.1.166	20170210	VBA32	-	9.900.0.1004	20170210
F-Secure		11.0.19100.45	20170210	VIPRE	-	3.12.26.4	20170210
Fortinet		5.4.233.0	20170210	V3Robot	-	55870	20170210
GData		25	20170210	Yandex	-	2014.3.20.0	20170210
Ikarus		0.1.3.4	20170210	Zillya	-	5.5.1.3	20170209
Invicrea		6.2.2.24419	20170203	Zoner	-	2.0.0.3202	20170210
		6.2.2.24419	20170203			1.0	20170210

## 意外发现：模型对HTML的检测率

- 偶然发现用二进制模型对恶意HTML也有很高检测率
- 然后发现正常HTML也有很高误报率
- 定位判断原因是1千万恶意样本中有很多包含HTML内容（广告软件等等），所以深度学习把HTML特征也包含进去
- 排除方法是正常样本中加入HTML
- 深度学习黑箱化的缺点：不清楚起作用主要是哪些特征

了很恶劣的影响。上图贴出了 VirusTotal 上各家病毒引擎对早期 WannaCry 样本的扫描结果。我们可以看到，只有 CrowdStrike 和 Endgame 两家使用机器学习为核心的病毒引擎能够将其识别为可疑。这也印证了我们所说的泛化能力强。

另外有一个意外发现是，我们的模型对恶意的 HTML 检测率也很高。但同时对正常的 HTML 样本有很高的误报率。定位发现原因是训练集的恶意样本中包含很多 HTML 内容，被深度学习抽取成了特征。优化方法很简单，只需要在正常样本中加入一定的 HTML 就可以平衡。

前面提到微软比赛中第一名的方案是有问题的。他的方案是用 N-gram 产生数万特征，然后用 XGboost 来做分类。赛后有人发现，

## 下一步规划

- 更深入的检测机制研究
- 文档类型病毒片段检测
- 病毒类型检测
- 脱壳后检测
- 映入长空间跨度特征

微软提供病毒样本时是按病毒分类放在不同路径下，而路径字符是包含在样本中，并被分类器判断成了重要特征。这才使其分类准确度优于其他选手。

上面两个案例都再次提醒我们，不能把机器学习黑箱化。

下一步，我们还会深入到网络中，继续探索具体的检测机制。我们还会测试其他的样本类型，比如文档类型。目前我们的输出只是一个二元判断，那安全人员可能希望可以进一步给出病毒类型，甚至是家族归属。方案层面，除了目前使用的二进制码转低纬度图片 +CNN 的方法，我们也在测试另一套方案，考察二进制码在长空间跨度下的特征，并应用 LSTM。

今天分享先到这，感谢大家！下周还会有我们瀚思高颜值的大数据架构师为大家分享基于 Flink 的超大规模实时流应用。想加入我们成都或者南京 site 做大数据 / 算法开发的，不要犹豫，直接邮件 hr@hansight.com。对机器学习和用户行为分析感兴趣的也欢迎直接来勾搭我。

## 问答环节

### Q1: 在文件加壳加密混淆的情况下，系统如何处理？

答：处理加壳样本的方法我在分享中 P12 已经直接讲到了。

Q2: 现在病毒传播很多靠社会工程学，是用户的正常行为，而且病毒也越来越以获利为目的，不搞破坏，重视隐藏。如果病毒使用

## gfw 对抗方的各种流使量无特征方式，检测系统如何应对呢？

**答：**现在某些病毒，在行为层面隐藏的比较好，基于行为的检测可能不适用了。那么我也讲到了，基于深度学习的这套检测技术，是基于二进制码的，是直接针对内容的。因为病毒的恶意行为总需要通过某些代码触发，我们即可以通过内容来抽取对应特征并识别。

## Q3：基于深度学习的检测结果的可解释性很低。如果出现了误报，除了 whitelist，有没有其他更好的办法？

**答：**首先我们的误报率是低于千分之一的。另外分享中我提到了，我们可以结合外部信息，比如威胁情报来产生高信息熵的结果。再者因为我们是做 2B，对需要二次确认的样本还可以放在我们大数据平台下做前后事件的追溯。

## Q4：可以详细说一下是如何将二进制文件转换为图像的吗？文件大小不一，是如何处理的？还有就是二进制文件有分段处理吗？

**答：**我们使用改造后的 pooling 算法，一次性转换，没有分段。

## Q5：对于样本 label，除了借助第三方如 VT 扫描结果，有没有其他更好的方法？

**答：**我们的样本 label 包括 VT 和安全部门自己产生的。

## Q6：您 PPT 中提到了标准模型和压缩模型，可以详细解释下这两个模型的差异吗？

**答：**简单回答，大的网络参数多，准确度高，但无论是 train 还是 inference 的速度都很慢。我们的企业客户很多又没有 GPU 可以提供。为了提速，我们可以通过剪枝或者选择小网络来训练，只保留几十分之一的参数，缺只损失 5% 左右的准确度。这也是最近深度学习的一个发展方向。

## Q7：PPT 中有提到替换最顶和最低几层，这个又是什么意思呢？具体如何替换的呢？

**答：**输入层的替换主要是对接我们转换后的样本。输出层的替换主要是分类器我们用了 lightGBM。

**Q8：请问您的方法中是不是只能检测已知的恶意文件，泛化能力这一块大体是个什么水平。**

**答：**如上面介绍的，深度学习模型记住的是病毒二进制文件中的有效特征，而不是特征码，所以具有更好的通用性。即使一个月不更新模型，对新衍生的病毒样本也有较高的识别能力。

**Q9：我对这个结果还有点疑惑，理由如下：深度学习本质还是个 $f(x)$ ，效果好的前提是训练和测试的分布有较强一致性，而迁移学习现在进展缓慢。我之前也做过一段时间安全，安全领域现实数据可以说五花八门，一般训练集几乎无法做到跟真实环境分布一致，基于分类的做法我当时感觉更多是学术价值，当然对于 dga 这种分布明显的是有效的。**

**答：**我们本身就是基于真实环境中的病毒样本来做训练的。安全团队每月都会拿到千万的最新病毒样本，我们也会持续更新模型。这个最终是会应用到客户环境的，并不是一个纯研究性的课题。

**Q10：请问一下，训练好的模型，可以在移动端上运行或自行改良吗？也就是不断应对持续升级的病毒样本。**

**答：**我们还没有测试过在移动端上运行。但是模型是会动态更新的。

## 讲师介绍

**吴睿**，瀚思科技高级算法专家，用户行为分析专家，毕业于 University of Missouri - Columbia 计算机系。



扫码阅读“基于Flink流  
处理的动态实时超大规模  
用户行为分析”

# 阿里小蜜：电商领域的智能助理技术实践

作者 陈海青



## 智能人机交互领域的介绍

### 行业分类及目前的应用状况

在全球人工智能领域不断发展的今天，包括 Google、Facebook、Microsoft、Amazon、Apple 等互联公司相继推出了自己的智能私人助理和机器人平台。

智能人机交互通过拟人化的交互体验逐步在智能客服、任务助理、智能家居、智能硬件、互动聊天等领域发挥巨大的作用和价值。因此，各大公司都将智能聊天机器人作为未来的入口级别的应用在对

待。今天随着市场的进一步发展，聊天机器人按照产品和服务的类型主要可分为：客服，娱乐，助理，教育，服务等类型。

图 1 截取了部分聊天机器人。

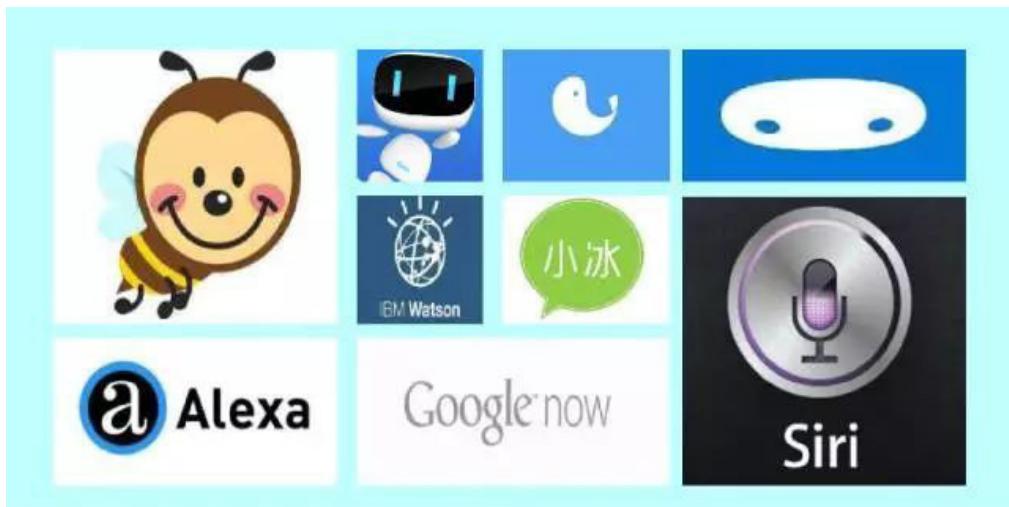


图 1 一些 chat-bot 的汇总

## 阿里小蜜在电商领域的状况

2015 年 7 月，阿里推出了自己的智能私人助理 – 阿里小蜜，一个围绕着电子商务领域中的服务、导购以及任务助理为核心的智能人机交互产品。通过电子商务领域与智能人机交互领域的结合，带来传统服务行业模式的变化与体验的提升。

在去年的双十一期间，阿里小蜜整体智能服务量达到 643 万，其中智能解决率达到 95%，智能服务在整个服务量（总服务量 = 智能服务量 + 在线人工服务量 + 电话服务量）占比也达到 95%，成为了双十一期间服务的绝对主力。

## 电商领域下阿里小蜜的技术实践

### 阿里小蜜技术的 overview

智能人机交互系统，俗称：chatbot 系统或者 bot 系统。

图 2 是人机交互的流程图：



**图 2 人机交互的流程**

核心是 NLU(自然语言理解)，通过对话系统处理后，最后通过自然语言生成的方式给出答案。一段语言如何理解对于计算机来说是非常有难度的，例如：“苹果”这个词就具备至少两个含义，一个是水果属性的“苹果”，还有一个是知名互联网公司属性的“苹果”。

### 意图与匹配分层的技术架构体系

在阿里小蜜这样在电子商务领域的场景中，对接的有客服、助理、聊天几大类的机器人。这些机器人，由于本身的目标不同，就导致不能用同一套技术框架来解决。因此，我们先采用分领域分层分场景的方式进行架构抽象，然后再根据不同的分层和分场景采用不同的机器学习方法进行技术设计。首先我们将对话系统从分成两层：

**意图识别层：**识别语言的真实意图，将意图进行分类并进行意图属性抽取。意图决定了后续的领域识别流程，因此意图层是一个结合上下文数据模型与领域数据模型不断对意图进行明确和推理的过程；

**问答匹配层：**对问题进行匹配识别及生成答案的过程。在阿里小蜜的对话体系中我们按照业务场景进行了 3 种典型问题类型的划分，并且依据 3 种类型会采用不同的匹配流程和方法：

- **问答型：**例如“密码忘记怎么办？” → 采用基于知识图谱构建 + 检索模型匹配方式

- 任务型：例如“我想订一张明天从杭州到北京的机票”→ 意图决策 +slots filling 的匹配以及基于深度强化学习的方式
- 语聊型：例如“我心情不好”→ 检索模型与 Deep Learning 相结合的方式

图 3 表示了阿里小蜜的意图和匹配分层的技术架构。

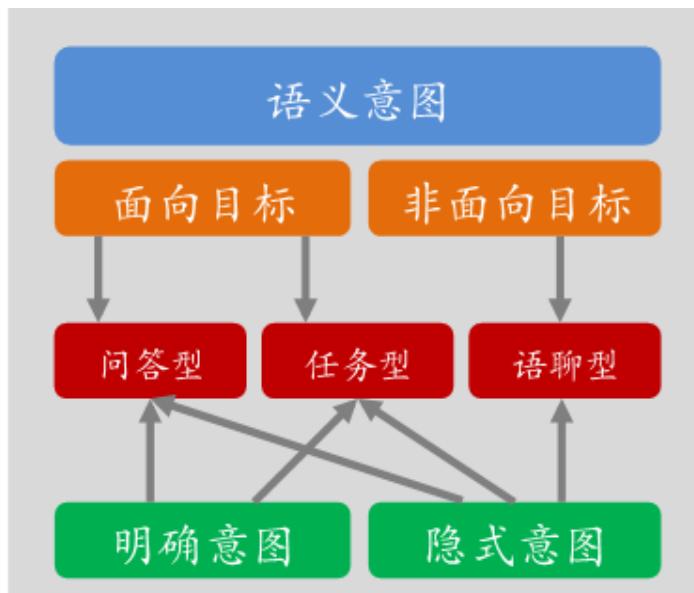


图 3 基于意图匹配分层的技术架构

意图识别介绍：结合用户行为 deep-learning 模型的实践

通常将意图识别抽象成机器学习中的分类问题，在阿里小蜜的技术方案中除了传统的文本特征之外，考虑到本身在对话领域中存在语义意图不完整的情况，我们也加入了用实时、离线用户本身的行为及用户本身相关的特征，通过深度学习方案构建模型，对用户意图进行预测，具体如图 4。

在基于深度学习的分类预测模型上，我们有两种具体的选型方案：一种是多分类模型，一种是二分类模型。多分类模型的优点是性能快，但是对于需要扩展分类领域是整个模型需要重新训练；而二分类模型的优点就是扩展领域场景时原来的模型都可以复用，可以平台

进行扩展，缺点也很明显需要不断的进行二分，整体的性能上不如多分类好，因此在具体的场景和数据量上可以做不同的选型。



图 4 结合用户行为的深度学习意图分类

小蜜用 DL 做意图分类的整体技术思路是将行为因子与文本特征分别进行 Embedding 处理，通过向量叠加之后再进行多分类或者二分类处理。这里的文本特征维度可以选择通过传统的 bag of words 的方法，也可使用 Deep Learning 的方法进行向量化。具体图：

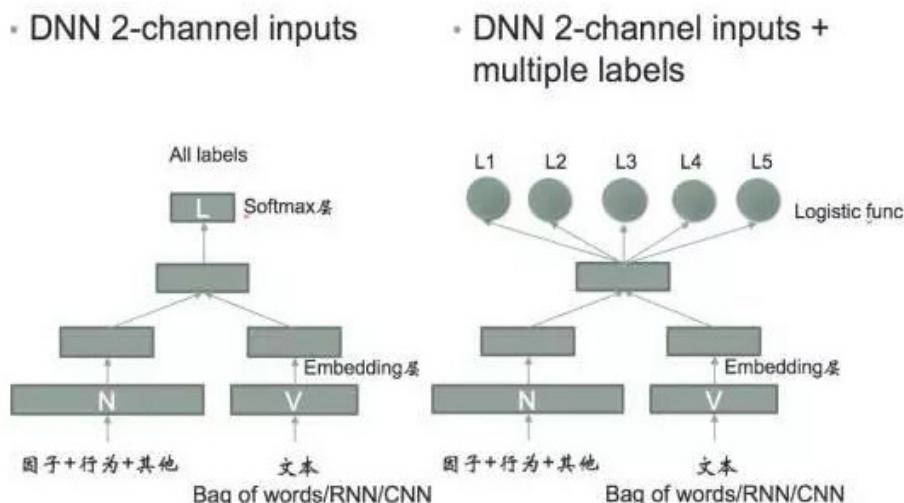


图 5 结合用户行为的深度学习意图分类的网络结构

匹配模型的 overview：介绍行业 3 大匹配模型

目前主流的智能匹配技术分为如下 3 种方法：

- 基于模板匹配 (Rule-Based)

- 基于检索模型 (Retrieval Model)
- 基于深度学习模型 (Deep Learning)

在阿里小蜜的技术场景下，我们采用了基于模板匹配，检索模型以及深度学习模型为基础的方法原型来进行分场景（问答型、任务型、语聊型）的会话系统构建。

## 阿里小蜜的 3 大领域场景的技术实践

### 智能导购：基于增强学习的智能导购

智能导购主要通过支持和用户的多轮交互，不断的理解和明确用户的意图。并在此基础上利用深度强化学习不断的优化导购的交互过程。图 6 展示了智能导购的技术架构图。



图 6 智能导购的架构图

这里两个核心的问题：

- a) 在多轮交互中理解用户的意图。
- b) 根据用户的意图结果，优化排序的结果和交互的过程。

下面主要介绍导购意图理解、以及深度增强学习的交互策略优化。

智能导购的意图理解和意图管理

智能导购下的意图理解主要是识别用户想要购买的商品以及商品对应的属性，相对于传统的意图理解，也带来了几个新的挑战。

第一，用户偏向于短句的表达。因此，识别用户的意图，要结合用户的多轮会话和意图的边界。

第二，在多轮交互中用户会不断的添加或修改意图的子意图，需要维护一份当前识别的意图集合。

第三，商品意图之间存在着互斥，相似，上下位等关系。不同的关系对应的意图管理也不同。

第四，属性意图存在着归类和互斥的问题。

针对短语表达，我们通过品类管理和属性管理维护了一个意图堆，从而较好的解决了短语表示，意图边界和具体的意图切换和修改逻辑。同时，针对较大的商品库问题，我们采用知识图谱结合语义索引的方式，使得商品的识别变得非常高效。下面我们分别介绍下品类管理和属性管理。

- 基于知识图谱和语义索引的品类管理

智能导购场景下的品类管理分为品类识别，以及品类的关系计算。下图是品类关系的架构图。



**图 7 品类管理架构图**

- 品类识别：

采用了基于知识图谱的识别方案和基于语义索引及 dssm 的判别

模型。

a) 基于商品知识图谱的识别方案：

基于知识图谱复杂的结构化能力，做商品的类目识别。是我们商品识别的基础。

b) 基于语义索引及 dssm 商品识别模型的方案：

知识图谱的识别方案的优势是在于准确率高，但是不能覆盖所有的 case。因此，我们提出了一种基于语义索引和 dssm 结合的商品识别方案兜底。

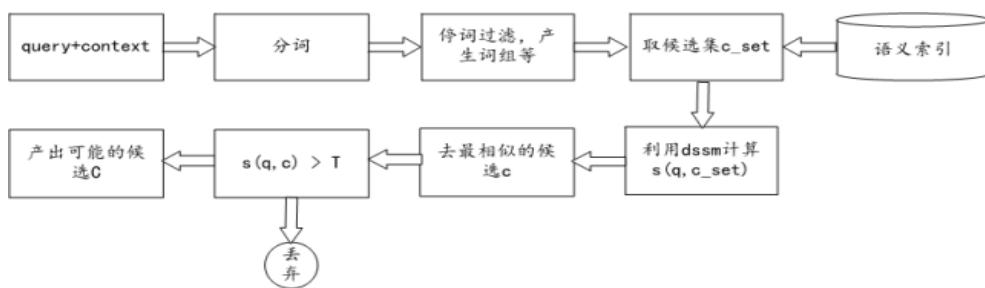


图 8 基于语义索引和 dssm 的商品识别方案

- 语义索引的构造：

通常语义索引的构造有基于本体的方式，基于 LSI 的方式。我们用了一种结合搜索点击数据和词向量的方式构造的语义索引。主要包括下面几步：

第一步：利用搜索点击行为，提取分词到类目的候选。

第二步：基于词向量，计算分词和候选类目的相似性，对索引重排序。

- 基于 dssm 的商品识别：

`dssm` 是微软提出的一种用于 `query` 和 `doc` 匹配的有监督的深度语义匹配网络，能够较好的解决词汇鸿沟的问题，捕捉句子的内在语义。本文以 `dssm` 作为基础，构建了 `query` 和候选的类目的相似度计算模型。取得了较好的效果，模型的 `acc` 在测试集上有 92% 左右。

样本的构造：训练的正样本是通过搜索日志中的搜索 `query` 和点

击类目构造的。负样本则是通过利用 query 和点击的类目作为种子，检索出来一些相似的类目，将不在正样本中的类目作为负样本。正负样本的比例 1: 1。

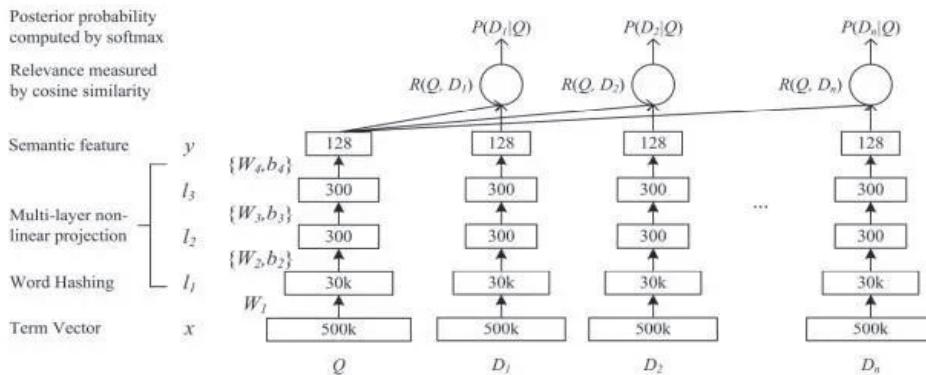


图 9 dssm 模型的网络结构图

- 品类关系计算：

品类关系的计算主要用于智能导购的意图管理中，这里主要考虑的几种关系是：上下位关系和相似关系。举个例子，用户的第一个意图是要买衣服，当后面的意图说要买水杯的时候，之前衣服所带有的属性就不应该被继承给水杯。相反，如果这个时候用户说的是要裤子，由于裤子是衣服的下位词，则之前在衣服上的属性就应该被继承下来。

- 上下位关系的计算 2 种方案：

- 采用基于知识图谱的关系运算。
- 通过用户的搜索 query 的提取。

相似性计算的两种方案：

- 基于相同的上位词。比方说小米，华为的上位词都是手机，则他们相似。

- 基于 fast-text 的品类词的 embedding 的语义相似度。

基于知识图谱和相似度计算的属性管理

下图是属性管理的架构图：

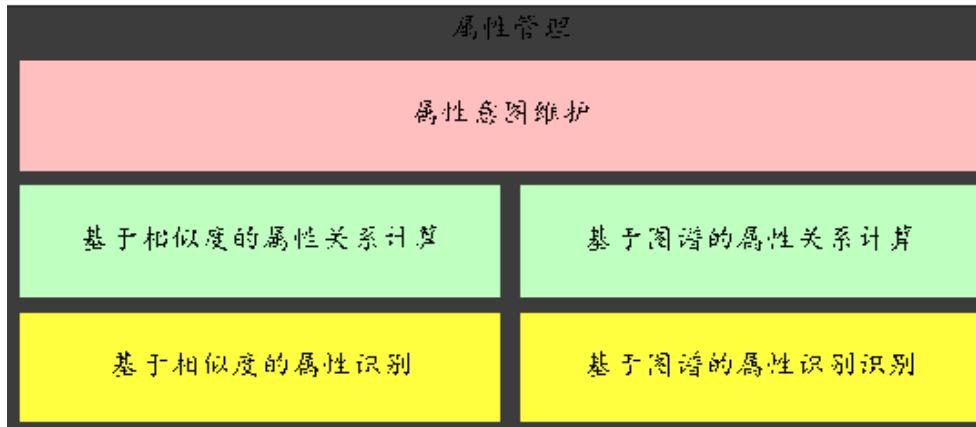


图 10 属性管理架构图

整体上属性管理包括属性识别和属性关系计算两个核心模块，思路和品类管理较为相似。这里就不在详细介绍。

### 深度强化学习的探索及尝试

强化学习是 agent 从环境到行为的映射学习，目标是奖励信号（强化信号）函数值最大，由环境提供强化信号评价产生动作的好坏。agent 通过不断的探索外部的环境，来得到一个最优的决策策略，适合于序列决策问题。图 11 是一个强化学习的 model 和环境交互的展示。

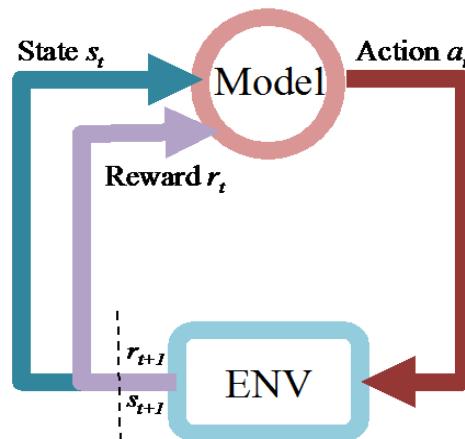


图 11 env-model 的交互图

深度强化学习是结合了深度学习的强化学习，主要利用深度学习强大的非线性表达能力，来表示 agent 面对的 state 和 state 上决策逻辑。

目前我们用 DRL 主要来优化我们的交互策略。因此，我们的设定是，用户是强化学习中的 env，而机器是 model。action 是本轮是否出主动反问的交互，还是直接出搜索结果。

状态 (state) 的设计：

这里状态的设计主要考虑，用户的多轮意图、用户的人群划分、以及每一轮交互的产品的信息作为当前的机器感知到的状态。

```
state= ( intent1, query1, price1, is_click,query_item_sim, ..., power, user_inter, age)
```

其中 intent1 表明的是用户当前的意图，query1 表示的用户的原始 query。price1 表示当前展现给用户的商品的均价，is\_click 表示本轮交互是否发生点击，query\_item\_sim 表示 query 和 item 的相似度。power 表示是用户的购买力，user\_inter 表示用户的兴趣，age 表示用户的年龄。

- reward 的设计：

由于最终衡量的是用户的成交和点击率和对话的轮数。因此 reward 的设计主要包括下面 3 个方面：

- a) 用户的点击的 reward 设置成 1
- b) 成交设置成  $[1 + \text{math.log(price} + 1.0)]$
- c) 其余的设置成 0.1

DRL 的方案的选型：

这里具体的方案，主要采用了 DQN，policy-gradient 和 A3C 的三种方案。

智能服务：基于知识图谱构建与检索模型的技术实践

智能服务的特点：有领域知识的概念，且知识之间的关联性高，并且对精准度要求比较高。

基于问答型场景的特点，我们在技术选型上采用了知识图谱构建 + 检索模型相结合的方式来进行核心匹配模型的设计。

知识图谱的构建我们会从两个角度来进行抽象，一个是实体维度的挖掘，一个是短句维度进行挖掘，通过在淘宝平台上积累的大量属于以及互联网数据，通过主题模型的方式进行挖掘、标注与清洗，再通过预设定好的关系进行实体之间关系的定义最终形成知识图谱。基本的挖掘框架流程如下：

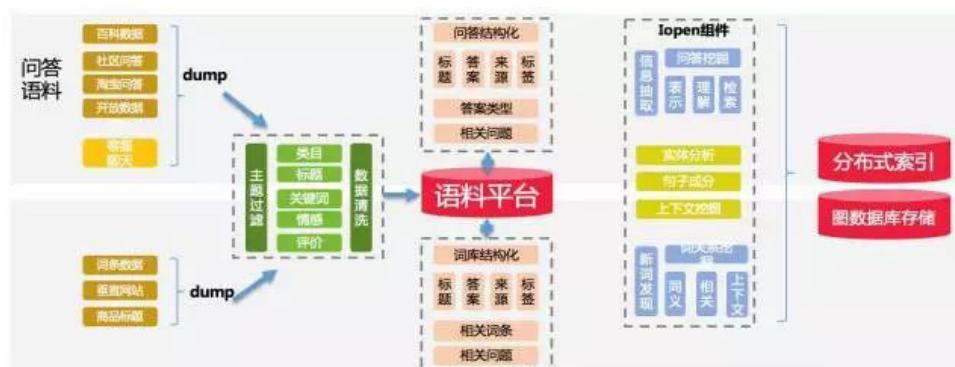


图 12 知识图谱的实体和短语挖掘流程

挖掘构建的知识图谱示例如图 13：

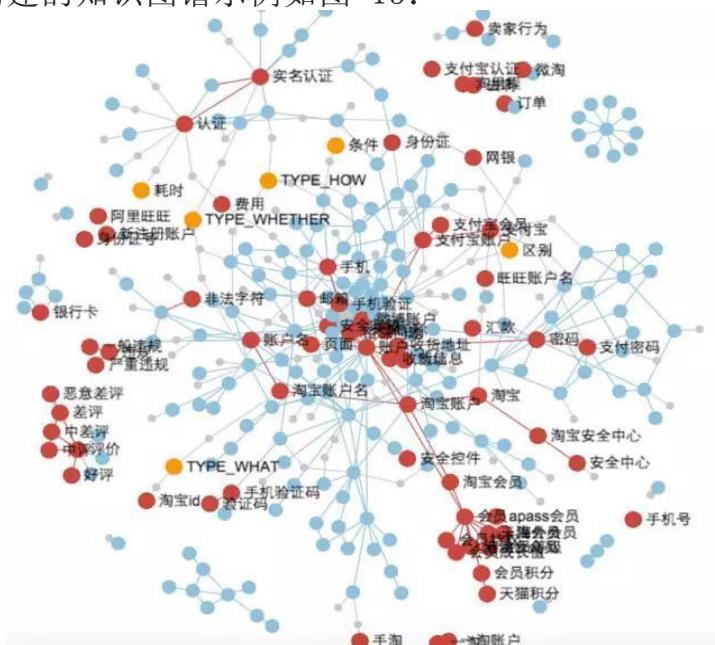


图 13 具体的知识图谱的示例

基于知识图谱的匹配模式具备以下几个优点：

1. 在对话结构和流程的设计中支持实体间的上下文会话识别与推理；
2. 通常在一般型问答的准确率相对比较高（当然具备推理型场景的需要特殊的设计，会有些复杂）。

同样也有明显的缺点：

1. 模型构建初期可能会存在数据的松散和覆盖率问题，导致匹配的覆盖率缺失；
2. 对于知识图谱增量维护相比传统的 QA Pair 对知识的维护上的成本会更大一些。

因此我们在阿里小蜜的问答型设计中，还是融入了传统的基于检索模型的对话匹配。

其在线基本流程分为：

1. 提问预处理：分词、指代消解、纠错等基本文本处理流程；
2. 检索召回：通过检索的方式在候选数据中召回可能的匹配候选数据；
3. 计算：通过 Query 结合上下文模型与候选数据进行计算，通过我们采用文本之间的距离计算方式（余弦相似度、编辑距离）以及分类模型相结合的方式进行计算；
4. 最终根据返回的候选集打分阈值进行最终的产品流程设计。

离线流程分为：

- 知识数据的索引化；
- 离线文本模型的构建：例如 Term-Weight 计算等。

检索模型整体流程如图 14。

智能聊天：基于检索模型和深度学习模型相结合的聊天应用

智能聊天的特点：非面向目标，语义意图不明确，通常期待的是语义相关性和渐进性，对准确率要求相对较低。



图 14: 检索模型的流程图

面向 open domain 的聊天机器人目前无论在学术界还是在工业界都是一大难题，通常在目前这个阶段我们有两种方式来做对话设计：

一种是学术界非常火爆的 Deep Learning 生成模型方式，通过 Encoder-Decoder 模型通过 LSTM 的方式进行 Sequence to Sequence 生成，如图 15：

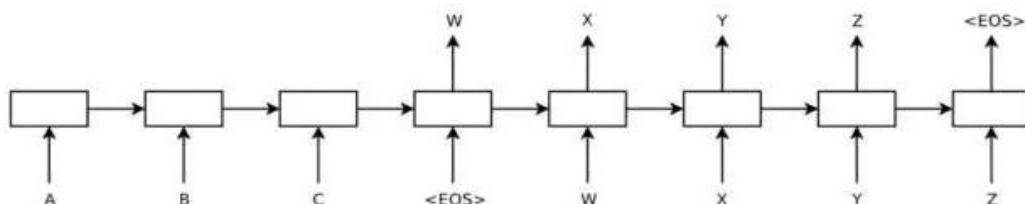


图 15 seq2seq 网络结构图

- GenerationModel(生成模型)：

优点：通过深层语义方式进行答案生成，答案不受语料库规模限制  
缺点：模型的可解释性不强，且难以保证一致性和合理性回答

另外一种方式就是通过传统的检索模型的方式来构建语聊的问答匹配。

- RetrievalModel(检索模型)：

优点：答案在预设的语料库中，可控，匹配模型相对简单，可解释性强

缺点：在一定程度上缺乏一些语义性，且有固定语料库的局限性

因此在阿里小蜜的聊天引擎中，我们结合了两者各自的优势，将两个模型进行了融合形成了阿里小蜜聊天引擎的核心。先通过传统的检索模型检索出候选集数据，然后通过 Seq2Seq Model 对候选集进行 Rerank，重排序后超过制定的阈值就进行输出，不到阈值就通过 Seq2Seq Model 进行答案生成，整体流程图 16：

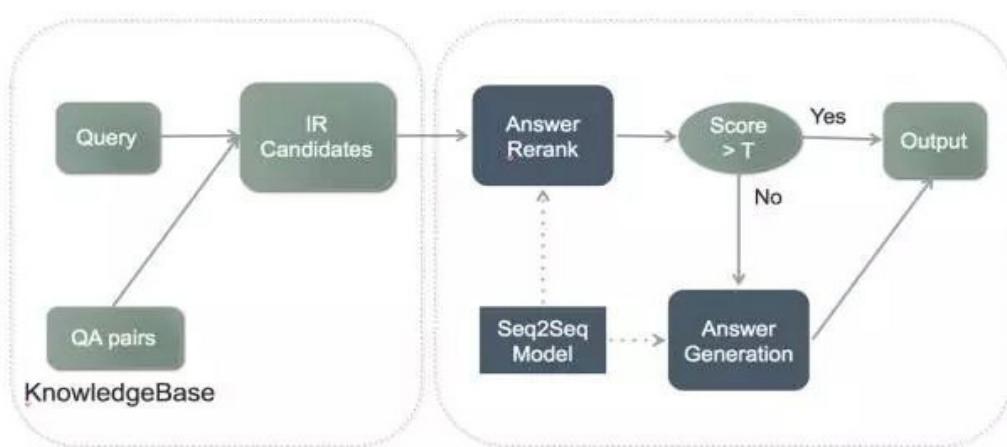


图 16 小蜜的闲聊模块

## 未来技术的发展与展望

目前的人工智能领域任然处在弱人工智能阶段，特别是从感知到认知领域需要提升的空间还非常大。智能人机交互在面向目标的领域已经可以与实际工业场景紧密结合并产生巨大价值，随着人工智能技术的不断发展，未来智能人机交互领域的发展还将会有不断的提升，对于未来技术的发展我们值得期待和展望：

1. 数据的不断积累，以及领域知识图谱的不断完善与构建将不断助推智能人机交互的不断提升；
2. 面向任务的垂直细分领域机器人的构建将是之后机器人不断爆发的增长点，open domain 的互动机器人在未来一段时间还需要不断提升与摸索；

3. 随着分布式计算能力的不断提升，深度学习在席卷了图像、语音等领域后，在 NLP(自然语言处理) 领域将会继续发展，在对话、QA 领域的学术研究将会持续活跃。

在未来随着学术界和工业界的不断结合与积累，期待人工智能电影中的场景早日实现，人人都能拥有自己的智能“小蜜”。

## 作者简介

**陈海青**，阿里巴巴智能服务事业部资深技术专家，在阿里从事智能人机交互领域相关的工作和研究 8 年，带领团队构建了阿里巴巴智能交互机器人系统。本文来自陈海青在“携程技术沙龙——人机语义交互 AI”上的分享。



扫码阅读“李维博士：  
NLP 助力电商智能化的台  
前幕后”

# Google Brain 工程师演讲实录： TensorFlow 与深度学习

作者 周玥枫



## Who am I

我的名字叫做周玥枫，我是 Google Brain 的工程师，我现在做 TensorFlow 和 TensorFlow 分布式的开发以及使用机器学习来优化 TensorFlow 的研究项目。

今天首先跟大家分享深度深入学习的例子，然后再跟大家简单介绍一下什么是 TensorFlow，以及 TensorFlow 一些最新特性，包括即将公开甚至还没有完成一些的特性，如果有时间的话，我会花一些篇幅着重介绍新的特性。最后的时间我会简要介绍一下 Google Brain 两个研究项目。

## Machine Learning

今天，我们看到机器学习已经改变了我们的世界，机器科学家用深度学习的方法来检测糖尿病和视网膜病变，其中检测视网膜病变能达到 95% 的准确率，甚至超过眼科专家 91% 的准确率。机器学习实现了机器和人类专家相媲美的准确率。



同时机器学习也可以用在自动驾驶方向，可以让交通更加安全和高效。



其次，机器学习能够跨越语言的障碍，实现更加便捷的沟通和交

流，我们知道传统的机器翻译系统需要把不同语言词组对应起来，通过一些复杂的甚至手写的规则，把一种语言转换为一种语言，这种系统非常庞大且不利于维护，而且准确度不够高，所以最近兴起了一种基于神经网络的方法，我们将其用 TensorFlow 实现出来，用这种方法来缩小机器和人类翻译的差距，能够使翻译更加准确和自然。

## 神经机器翻译



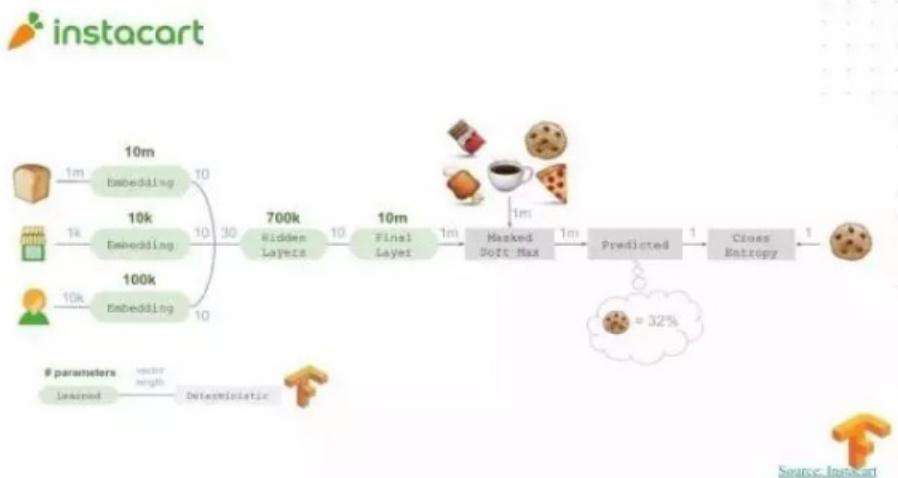
同样的，机器学习还可以给人类带来愉悦，可以实现自动修改照片、突出人物的前景、背景虚化等功能，我们很快可以在手机上看到这个功能。

## 相机效果

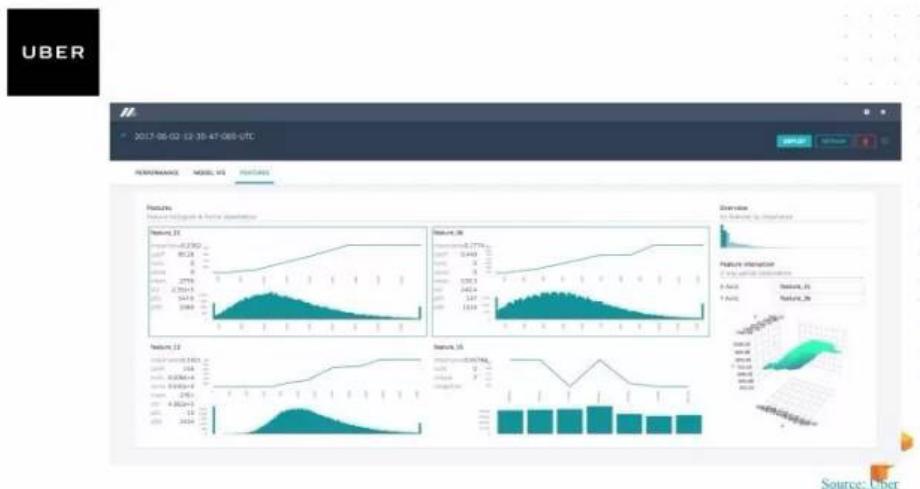


接下来看看机器学习在工业界的应用，第一个例子是INSTACART，

它是做杂货当天送货服务的，顾客通过网络应用程序从当地许多零售商选出想要的商品并且购买杂货。这个软件的客户非常多。客户在购买时面临一个问题，就是从数百万计零售商店或者商品中选出自己想要的物品，所以 INSTACART 为了让购物者更快地找出想要的商品，用 TensorFlow 建立了一套深度学习模型用来最有效地排序商品列表，这种方法能大大省下购物者寻找商品的时间。



第二个例子就是 UBER ， UBER 用 TensorFlow 和基于 TensorFlow 的开源项目来构建一个叫做“米开朗基罗”的系统，这是一个内部使用的机器学习平台，谷歌希望利用这个平台让内部使用 AI 就像他们请求乘车一样的方便。这个系统涵盖了从数据管理、



数据获取和模型训练、评估、部署等方面，而且这个系统不但支持 TensorFlow 深度学习，还支持其他机器学习的模型。

第三个例子是 KEWPIE，它用 TensorFlow 搭建了人工智能系统用来改善婴儿食品的质量，对食物产品进行人工智能分析，这样可以识别出产品中可以接受的成分并且剔除产品中不能接受的成分，这样保证了婴儿食品的质量。



Source: Google Blog

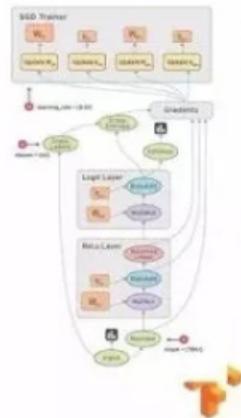
## What Is TensorFlow

而实现上述这一切所有的基础框架就是 TensorFlow。

我们在 2015 年末开源了 TensorFlow，希望把它做成能够服务所有人的机器学习平台。我们想要将它做成一个快速灵活的、生产环境就绪的框架。它可以很方便可以做研究，也可以很快部署到生产环境当中。TensorFlow 本质上是一个大规模的运算框架，它的运算被抽象成一张运算矢量图。就像在这边看到一张运算图一样，上面的节点代表运算或者状态。当我们完成了一些运算或者结束了一些状态的时候，我们的数据就从一个节点流到另外一个节点。这个图可以用任何语言来构建，当这张图构建完之后，我们把它传到 TensorFlow 核心当中进行编译，进行优化然后来执行。

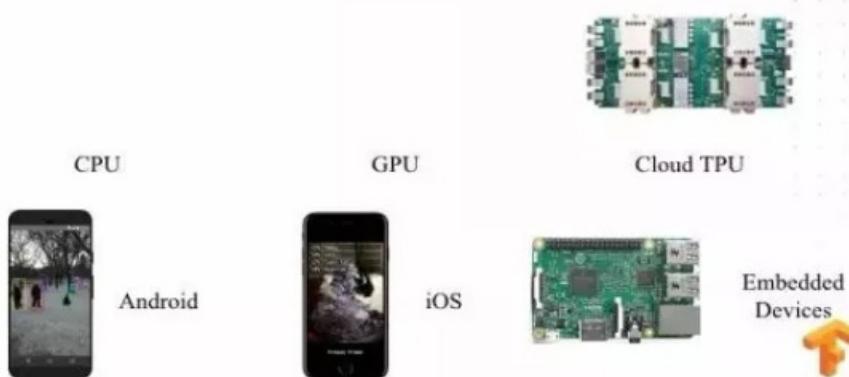
## TensorFlow的运算图

- Computation is defined as a graph
- Nodes represent computation or states
  - Can be run on any devices
- Data flow along edges
- Graph can be defined in any language
- Graph would be compiled and optimized



TensorFlow 也支持很多硬件平台，从最初的 CPU、GPU，到即将发布 CLOUD CPU，还有对安卓、iOS 的支持，甚至对嵌入式设备的支持。

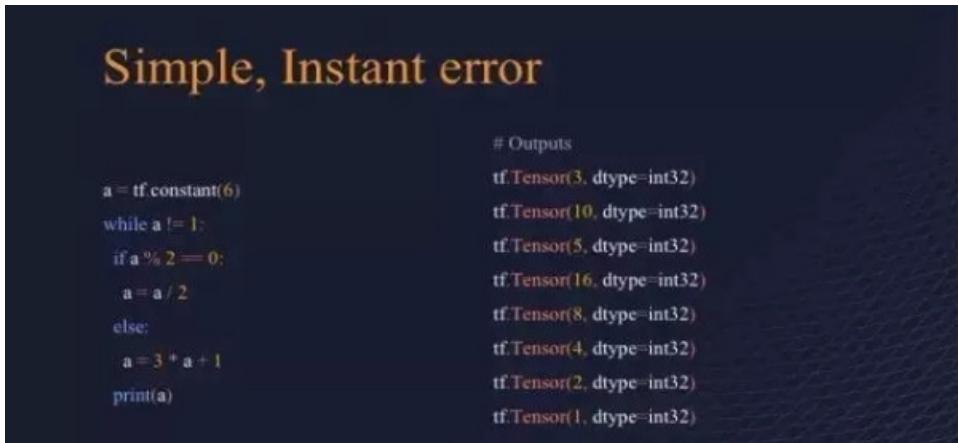
## TensorFlow支持各种硬件平台



我们将 TensorFlow 开源到 Github 上面后，过去两年兴起了许多围绕 TensorFlow 活跃的开源社区，现在我们有 67,000 多个 star，有 17,000 多个 Github 项目名字当中包括 TensorFlow。TensorFlow 不断出现在各种大学课程和在线课程里面，很多大学也正在开发基于 TensorFlow 的课程，除此之外我们也发布了 [TensorFlow 中文网站](#)，大家把它可以当做入门 TensorFlow 的初级教程。

## New Feature of TensorFlow

现在我们看一下 TensorFlow 的最新特性。首先是 Eager Execution，Eager Execution 是一种新的编程模式，我在之前一张幻灯片中展示了一个基于 TensorFlow 的静态图。



The slide has a dark background with orange text. The title 'Simple, Instant error' is at the top. Below it is a code snippet and its output:

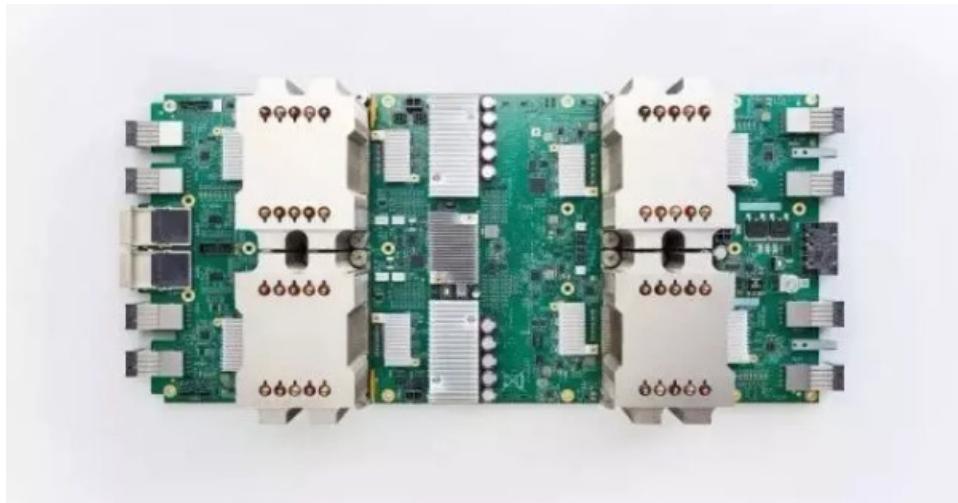
```
a = tf.constant(6)
while a != 1:
    if a % 2 == 0:
        a = a / 2
    else:
        a = 3 * a + 1
    print(a)

# Outputs
tf.Tensor(3, dtype=int32)
tf.Tensor(10, dtype=int32)
tf.Tensor(5, dtype=int32)
tf.Tensor(16, dtype=int32)
tf.Tensor(8, dtype=int32)
tf.Tensor(4, dtype=int32)
tf.Tensor(2, dtype=int32)
tf.Tensor(1, dtype=int32)
```

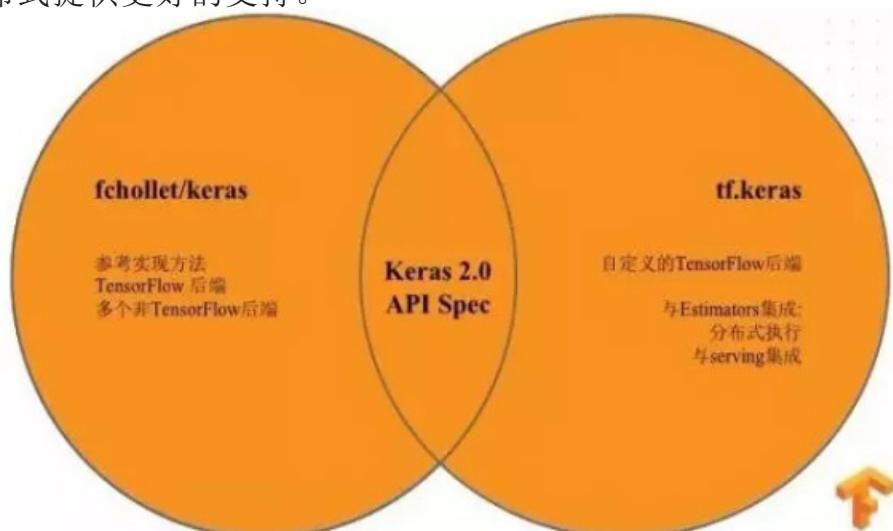
Eager Execution 解决了静态图中一些问题，解决了什么问题呢？首先它可以少写很多代码，就像上图一样。其次，通过 Eager Execution 写代码可以立刻发现它的错误，相对之前来说可以大大提高编写代码查错的效率。第三是可以用 Tensor 来编写控制流，就不需要用 TF 来做循环甚至做判断。最重要一点是如果用其他语言编写这张图的话，再把这图传到 TensorFlow 核心中相当于编写了另外一种代码。看到这个幻灯片就是简单的例子，充分说了 Eager Execution 的简单之处。

今年的 Google I/O 大会宣布了第二代 TPU，我们第二代 TPU 既可以做推理也可以作训练。一个 TPU 可以实现很高的词典运算。我们甚至可以把很多代 TPU 联合起来成为一个就像超级计算机一样的计算核心。在第二代 TPU 的帮助下，我们可以在 20 小时内全部训练出 RESNET-50 的模型，以前如果只做一个 TPU 的训练，可能要花一周的时间来训练这个模型。今天第二代 TPU 即将发布到 Google Cloud，并且推出供大家使用。

下面讲一下 TensorFlow 上层 API，除了神经网络训练速度，大家还关注如何更加方便实现用 TensorFlow 上层 API 来创建神经



网络。Keras 是其中一个 API , 它支持很多的后端。相信很多观众都用过 Keras , 从本质上来说 Keras 更加像一种 API 开发规范。TensorFlow 有一个 TF 就是 Keras , 但是它只是 API 规范实现的一种方式, 使用的是一个自定义 TensorFlow 后端, 有了这个后端, 我们可以让 Keras 与 Estimators 或者 Serving 整合起来, 这样会对分布式提供更好的支持。



还有一个在TensorFlow里面介绍的概念, 叫做Estimators , 这是一个比较轻量化, 并且在谷歌内部生产环境中广泛使用的API 其中 Estimators 提供了很多模型供大家使用, 叫做Canned Estimator,

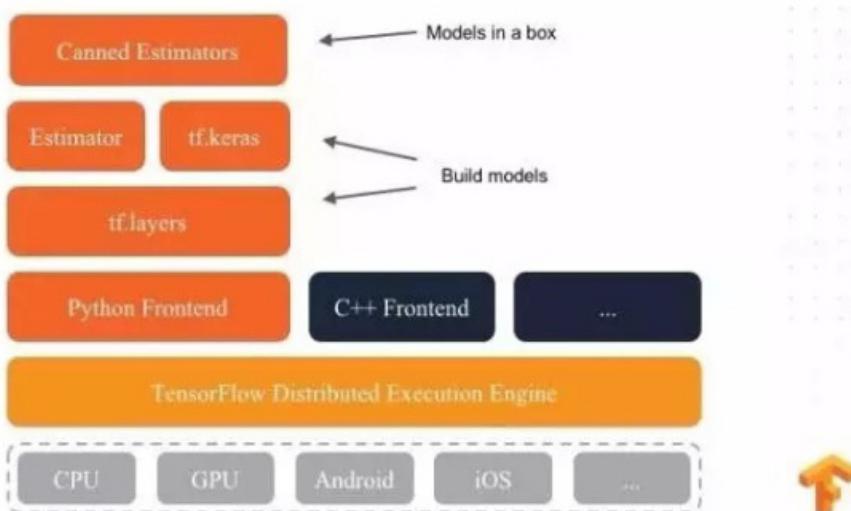
他们的关系是这样的：Estimators 和 tf.keras 上层封装了一个 Canned Estimator，可以用其来封装成各种模型。

## Estimators

实现了：

- 训练loop.
- checkpointing
- 内置与TensorBoard的集成
- 输出到 tensorflow/serving

Canned Estimator

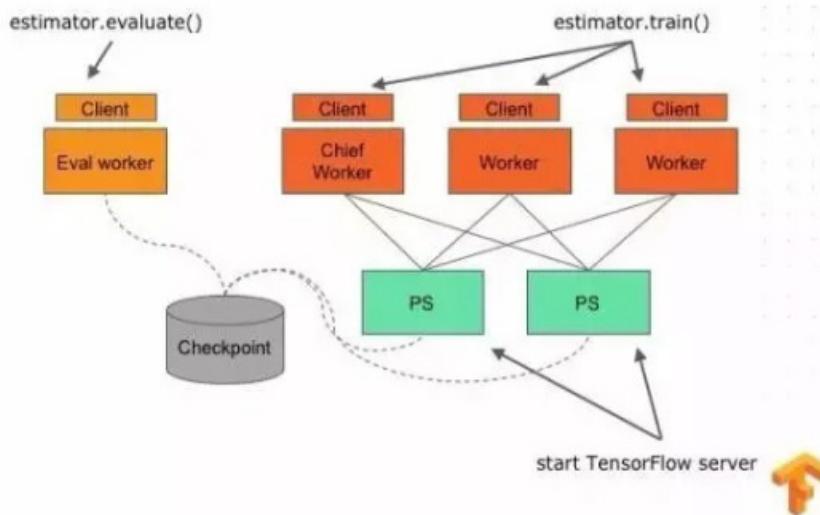
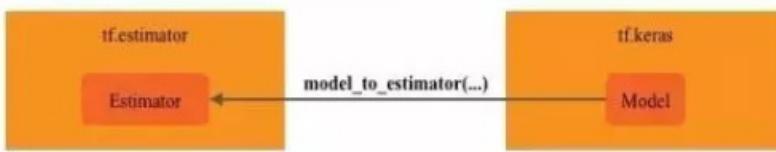


如果你们习惯于 Keras 的接口的话，我们提供了一个上层 API 转换的工具，叫做 `model_to_estimator`，一旦你有一个编译好的 Keras 模型就可以调用这个 `model_to_estimator` 来获取一个 Estimator，从而将 Keras 转换成了 Estimator。

Estimator 还提供了分布式训练的接口，如果你用 TensorFlow 来做分布式训练的话，你就可能熟悉我们的分布式模式。我们的 Estimator 很好地提供了对分布式训练的支持，只要写一份单机的代码，它就可以帮你构建好在不同机器上的执行的程序，训练的程序只要调用 `Estimator.train` 就能完成这一执行过程，只要调用它的

`Estimator.evaluate`，整个集群就可以开始训练了。

## Keras/Estimator 集成



大家可以看一下这些 API 的文档：TF，KERAS，TFLAYERS 等等，我们还发布了一个改进过的程序员指南在官网上，希望大家去看一下。

下一个特性是 TensorFlow Lite，TensorFlow Lite 是跑在移动设备上的 TensorFlow 的一个子集。现在移动设备无处不在，并且越来越重要。在移动设备上，我们可以在野外判断这个狗是什么品种或者判断这个植物有没有病害，利用人工智能都可以在移动设备做一些应用，所以我们推出了 TensorFlow Lite.

为什么很多时候要在移动设备上做？除了刚才说的那些应用场景，为什么要做移动设备的推理？这是因为我们时常需要在一些特殊

## Check out...

tf.keras

tf.layers

tf.estimator

New Programmer's Guide:

[www.tensorflow.org/programmers\\_guide](http://www.tensorflow.org/programmers_guide)

BlogPost: <https://goo.gl/RyLuUw>

环境下做一系列的推理，很多时候，尤其在野外，我们的网络带宽非常的低，网络延迟非常大。如果每次推理都向远程服务器发送请求，那对移动设备的电池能力要求很高。虽然现在市面上对移动设备能够完成推理有迫切的需求，但是其中存在很多的挑战，最主要的挑战是因为移动设备的内存、计算资源以及带宽等等受到了限制。从编程角度来讲，因为平台抑制性很高，开发越来越复杂，比如说在安卓上，我们可以使用 CPU 或者指令等方式编写底层代码，在 IOS 上又有自己一些平台和工具，这种不同平台的工具让我们的硬件以及 API 的开发，甚至存在不同的 API 让我们的开发变得更复杂，所以我们设计了 TensorFlow Lite。

## Why on-device?



Offline



Low-bandwidth



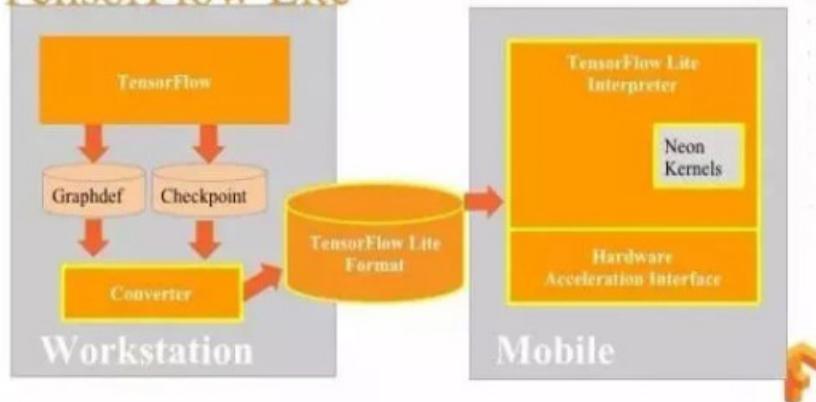
Latency



Power

相比 TensorFlow Lite 的话，TensorFlow 主要关注一些大型的设备。TensorFlow Lite 让小型的设备应用更加效率，现在我们通过一个小的例子看 TensorFlow Lite 如何工作的。

## TensorFlow Lite



这个是 TensorFlow Lite 生命周期，首先我们以标准方式来运行 TensorFlow，运行结束之后，我们得到 Graphdef 和 Checkpoint，我们通过 TensorFlow Lite 提供一个转换器，把它转换成 TensorFlow Lite 的模型格式。有了这个 TensorFlow Lite 格式的模型之后，我们就可以把它转移到移动设备当中。接入 TensorFlow Lite 显示器就能在移动设备加载这个模型。如果我们的显示器直接调度 NeonKerels，如果是在其他设备上，还可以利用硬件加速器接口来定义自己对自己硬件的一些优化。

下一个特性就是 Input Pipeline，不管是初学者还是专家都会对 Input Pipeline 感兴趣，因为 Input Pipeline 使用起来非常痛苦。

### Feeding

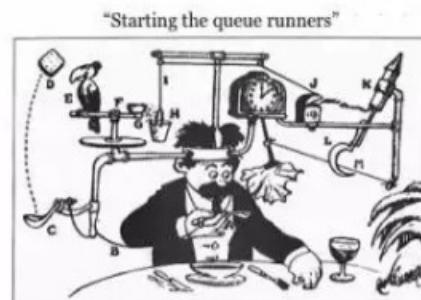
```
sess.run(...  
        feed_dict={x: features,  
                   y: labels})  
  
All the flexibility of Python...  
...and all the performance
```



它主要有两种模式，一种是 Feeding，它优点是可以用 python 灵活处理零距，但是性能很差。而且单线程跑，每一步训练都要等待数据处理的完成，这个对 GPU 来说效率非常低。另外一种方式效率高一点，就是把数据处理变成一系列的操作，这使用一个 Queues 作为数据存放的临时空间，我们把预处理好的数据和一些中间预处理数据放在 Queues 里面，通过 python 来控制 Queues 的输入和控制。但是有一个问题，这个 python 有一个权值解释器的锁，所以它让这个 Queues 输入性能受到很大的限制。

## Queues

```
files = string_input_producer(...)  
record = TFRecordReader().read(files)  
parsed = parse_example(record, ...)  
batch = batch(parsed, 32)  
  
Uses TensorFlow ops to perform  
preprocessing, but driven by client  
threads.
```



还有一个问题就是，我们写的数据处理模块没有办法得到常用，也没有办法在训练的时候更改输入数据。所以我们开发了一套 Input Pipeline，因为种种原因，所以把它设计成惰性的列表。因为我们的数据很多长得类似，而且数据量可以比较大，所以可以把它定义成 LAZY，我们把它定义成惰性列表之后，可以很自然用函数编程语言中的 map 和 filter 来定义预处理管道，我们在很多语言当中都可以看到 map 和 filter 的操作。现在我们看一下它的接口，我们还有第二个接口叫做 Iterator，可以方便把 elements 数据取出来。就像很多一般的编程语言里面的 Iterator 一样，我们可以对这个 Iterator 配置不同的数据，PPT 上这就是一个例子，大家可以看一下。

## Input pipelines = lazy lists

Functional programming to the rescue!

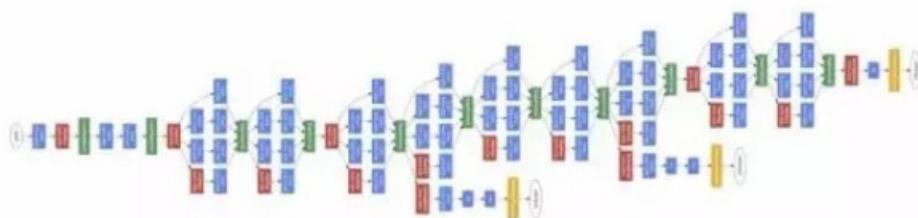
Data elements have the same type

Dataset might be too large to materialize all at once... or infinite

Compose functions like map() and filter() to preprocess

## Learn To Learn

在神经网络解决问题的时候，神经网络也会给我们带来一些新的问题，就是我们设计神经网络架构需要投入大量的专业知识和时间投资，比如这个就是谷歌图象分类的一个 .NET 的架构，这个神经网络架构设计从最初的卷积的架构到现在复杂的架构，经过研究人员多年的实验不断重复、完善，才得以达到这么复杂的模型。



我们与其让科研人员不断在电脑面前，为什么不用强大计算资源，让机器自动寻找好的神经网络架构？在谷歌我们用一种方法，就是用 RNN 循环神经网络来生成一个子网络，这个想法的来源是因为我们可以把一个神经网络对应成一个训练化的一个个序列，RNN 非常擅长解决这类问题，所以我们用 RNN 来生成一个子网络，我们把这个子网络用真实数据进行训练。通过训练准确度的好坏来更新 RNN 的控制器。在它下一次迭代当中，RNN 就会输出更高精度的子网络。这是 RNN 转接架构的例子，这个例子看上去错综复杂，其实也不难理解。

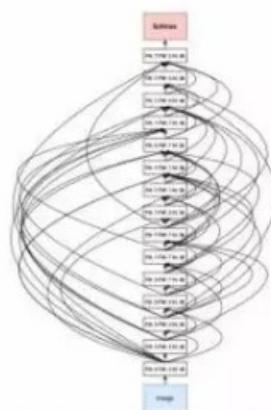
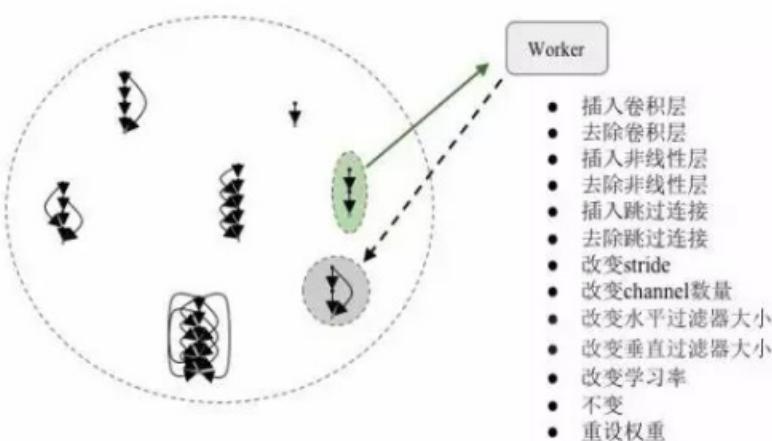


Figure 7: Convolutional architecture generated by our model, where the search space does not have nodes in preceding layers. PW is filter weight and N is number of filters.



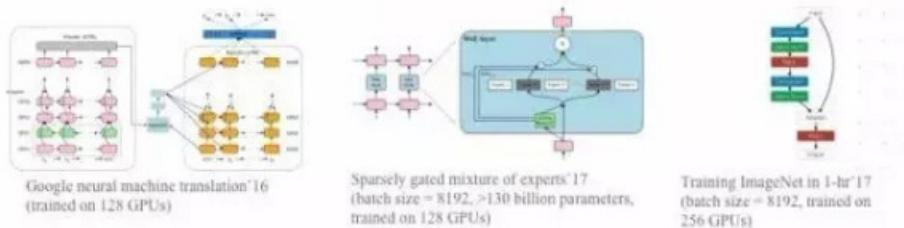
我们也尝试用同样的方法来生成优化函数，然后把生成的优化函数和常用的函数来进行对比，从这张表可以看到生成的函数超过了现有函数的精度，但是这些生成的函数非常的不直观，所以我们就开始想，有没有更加直观的方法来处理 learn 2 learn 这个问题，于是我们想到了进化的算法，或者说遗传算法。既然一个简单单细胞生物都可以进化到非常复杂的、有独立思考多细胞生物，那么我们可以不可以把同样理论用到模型的构建上来，于是我们就设计了这样一种算法，在每个时间的开始，我们建立了含有一千个训练好模型的种群，每一步进化从一千个模型中随机选两个，比较他们的准确率，准确率低的模型就会被扔掉，准确率高的模型会留下来，而且模型会繁殖，他们繁殖的方式就是给模型创建一个副本，并且简单修改这个模型，看上去像基因变异一样，我们会训练这个变异的副本然后放在种群当中。



这个是分布式训练的细节。模型变异有很多种，我们刚才提到结构的一些变化，甚至也有一些会保持不变，但是我们会对它多训练一下，或者重新训练一下，因为初始化变异非常重要。变异化的选择是均匀分布概率的模型。

## Why Device Placement?

Trend towards many-device training, bigger models, larger batch sizes



我们看一下进化时间的最新进展，图中横轴代表的是时间，纵轴是准确率，里面每个点代表种群当中一个模型，灰色点是被淘汰的模型。右上方蓝色的点是存活下来的模型，训练三天之后就在这个位置。训练三天之后就可以得到一个比较高的准确率。在训练十天之后，可以看到准确度达到停止期，也就是我们找到了个比较理想的模型。

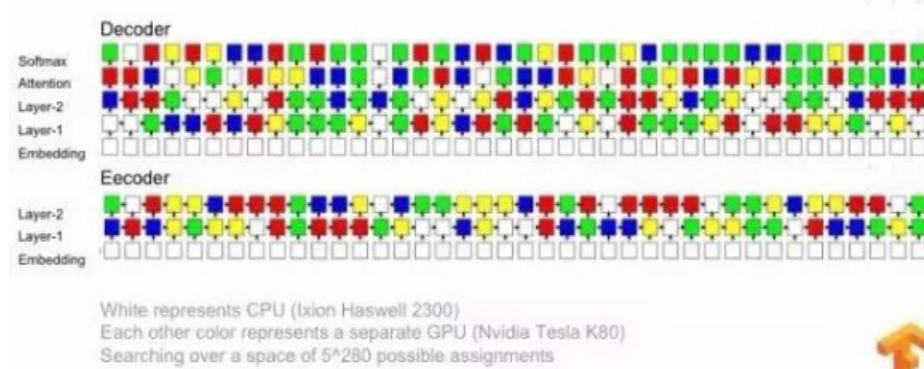
## Device Placement

我们看一下另外一个研究项目叫做 Device placement，他是用强化学习的方法来做 Device Placement，当前机器学习使用了非常多的模型，也使用非常多的数据，这个要求我们必须有很多的设备共同训练这个模型。

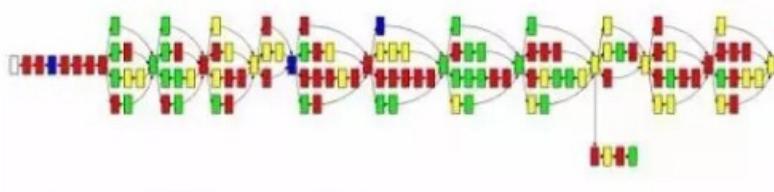
比如说看到这个翻译的模型就非常大。我们通常使用 128 个 GPU 来训练，我们先简单介绍一下 Device placement，Device placement 就是把计算和设备对应起来，目前我们都是用一些人工的方法，甚至简单的算法。我们设置算法需要我们对设备有充分的了解，而且对模型有充分

的了解，这个算法并不是简单从一个模型推广到另外一个模型。但是目前非常时髦的做法，都是将基于这些规则的系统转化为，变成基于机器学习的系统，所以我们也用了类似的方法。我们用强化学习的方法来解决这个 Device placement 的模型，我们受 learn 2 learn 方法的启发来创建一个类似的模型。

### Learned Placement on NMT



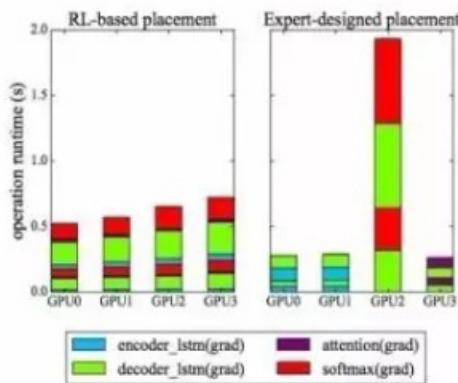
### Learned Placement on Inception-V3



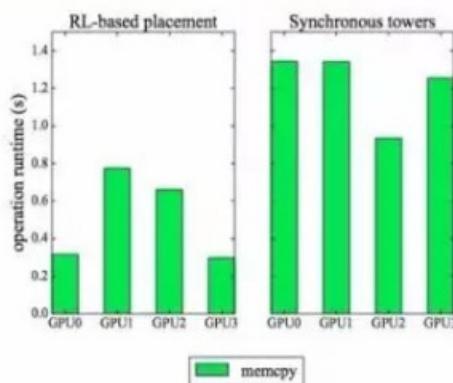
我们有一个网络，这个网络以神经网络作为输入，同时告诉一个网络有多少计算资源，这个网络就会告诉我们 Neural Model，我们跑一下这个放置好的 Neural Model，根据运行的时间来调整一下神经网络，我们用类似的机器翻译的架构。因为模型比较大，比较慢，我们采用了数据并行方式来做训练。

之后我们看一下训练出来的结果，这是在一个神经翻译系统 Device placement 的结果，上面白色的点代表是在 CPU，在 CPU 运行的节点，下面彩色点代表在不同 GPU 运行的节点，尽管取得了 20% 左右的提高，但是还是非常不利于理解。现阶段这个在输入端采用了 CPU，到后面都是采用了 GPU。

### Profiling Placement on NMT



### Profiling Placement on Inception-V3



最后两张图表示我们在神经翻译系统上，每个 GPU 运算非常平衡，而右边人类专家设计的运算非常不平衡，在 GPU2 花了非常长的时间，在其他 GPU 花了很少的时间，但是这个也是可以理解，因为专家设计一般只考虑到一部分。在 Inception V3 我们的结果不是非常平衡。但是

可能是因为 Inception V3 当中有一些过分的依赖，我们的结果仍然在 Inception V3 有总体的运行时间上的提高。后来我们做一些分析发现因为在数据拷贝期间，我们花了更少的时间。所以总体对它有一个运行时间的提高。

## The End

我们讲了两个例子，我们探索如何用算法或者机器学习来优化机器学习的模型，并且取得了一定的进展，我们相信在不久的将来，如果有更好的计算资源，以后的所有架构将是由电脑的生成，谷歌已经给大家走出了探索的第一步，希望大家都可以参与其中，我讲到这里，谢谢大家！

# ImageNet 冠军带你入门计算机视觉：监督学习与神经网络的简单实现

作者 董健



近几年，人工智能的浪潮席卷了整个科技圈。Google，Facebook，微软，百度等全球最顶尖的科技公司都将目光转向了人工智能，并将之视为今后的战略重心。随着人脸识别，辅助驾驶，AlphaGo等应用的不断涌现，基于学习的计算机视觉（learning based vision）正在越来越多的改变我们的生活。本系列文章，将逐步介绍这些看似神奇的系统背后的视觉算法原理。

本文是整个系列的第一篇文章，将会简单介绍一下计算机视觉的发展，以及监督学习、神经网络的基本原理。最后的实践部分，会用TensorFlow给出之前介绍算法的一个简单实现。

## 计算机视觉的发展

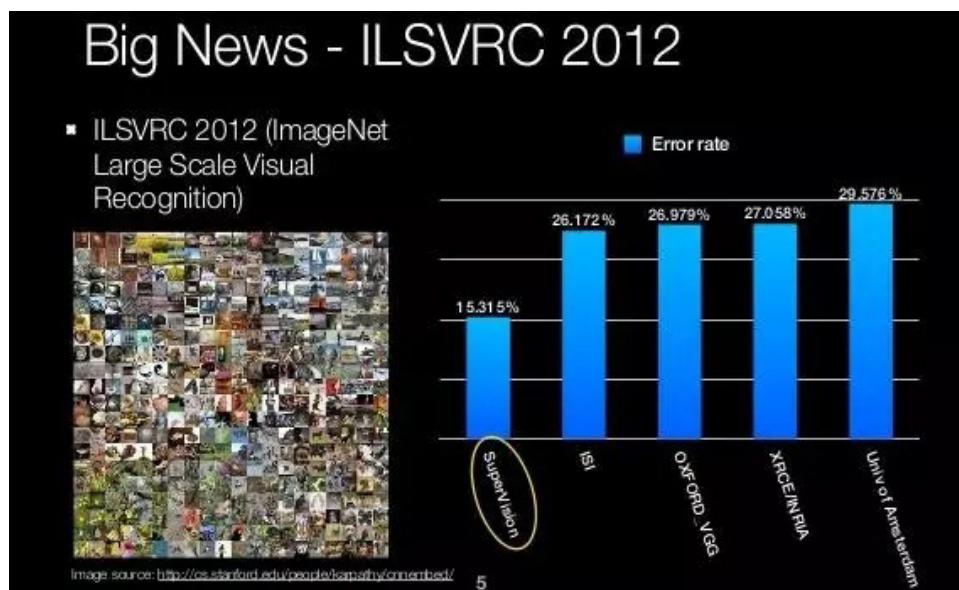
什么是计算机视觉？首先我们看一下维基百科的定义：

Computer vision is an interdisciplinary field that deals with how computers can be made for gaining high-level understanding from digital images or videos.

简单来说，计算机视觉就是让机器能自动的理解图片或视频。

计算机视觉的起源可以追溯到1966年，当时MIT著名的工智能专家 Marvin Minsky给他的本科学生留了一个暑期作业-“Link a camera to a computer and get the computer to describe what it saw”。虽然人可以很容易的理解图片，但事实证明，让计算机理解图片远比我们一开始想象的复杂。

早期的计算机视觉研究，由于计算资源和数据的原因，主要集中在几何和推理。上世纪90年代，由于计算机硬件的不断发展和数字照相机的逐渐普及，计算机视觉进入了快速发展期。这期间的一大突破是各种人工设计特征的涌现，例如SIFT，HOG等局部特征。这些特征相对原始像素具有对尺度，旋转等的鲁棒性，因此得到了广泛的应用，催生了如图像拼接、图像检索、三位重建等视觉应用。另一大突破是基于统计和机器学习的方法的流行。随着数字照片的不断普



及，大规模的数据集也相伴而生，基于学习的计算机视觉（Learning based Vision），由于可以通过大量数据自动学习模型参数，渐渐的成为了主流。

随着计算能力的不断进步和海量互联网数据的产生，传统的基于人工特征和SVM/boosting等简单机器学习算法的视觉技术遇到了瓶颈。因此，工业界和学术界都在探索如何避免繁琐的人工特征设计同时加强模型的拟合性能，从而进一步利用海量数据。深度学习很好的满足了这一需求，因此在视觉领域得到了非常广泛的应用。2010年之后，计算机视觉逐渐进入了深度学习的时代。标志性的事件是ImageNet 2012比赛。这次比赛中，基于深度学习的算法大大超过了经过精心设计的传统算法，震惊了整个学术界，进而带动了深度学习在其他领域中的应用。这次比赛也被看成是深度学习在整个人工智能领域复兴的标志性事件。

目前，除了三维重建等low-level vision问题，基于深度学习的算法，在大多数视觉问题上的性能已经远远超过了传统算法，因此本系列文章会重点介绍基于深度学习的计算机视觉算法。

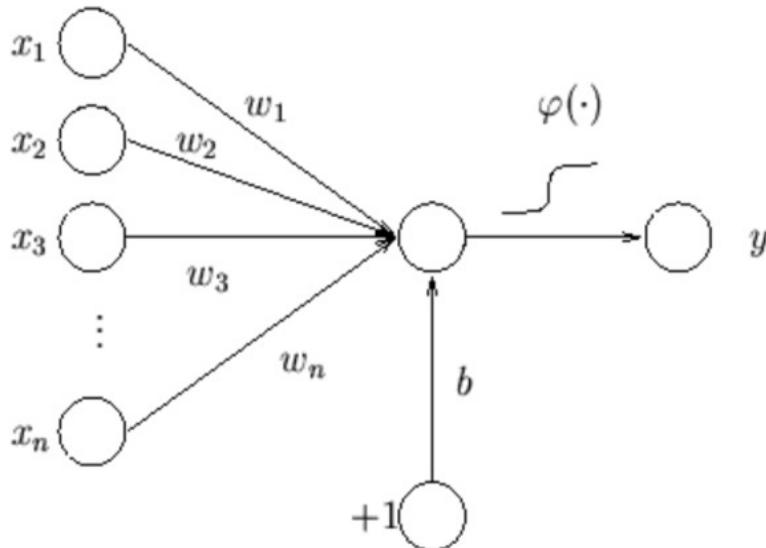
## 神经网络 (neural network)

神经网络（NN），简单来说就是神经元组成的网络，是最早出现，也是最简单的一种深度学习模型。其他很多更复杂的算法比如卷积神经网络，深度增强学习中的许多概念都来源于神经网络。因此，我们在这篇文章中先介绍一下神经网络的原理。要理解神经网络，我们需要先了解什么是神经元。

### 神经元 & 感知器

神经元（neuron）是神经网络的最小单位。每个神经元将多个输入映射到一个输出。如图所示，神经元的输出是输入的加权和加上偏置，再通过一个激活函数。具体可以表示成：

$$y = \varphi(\sum_i^n (w_i * x_i) + b) = \varphi(\mathbf{w} \cdot \mathbf{x} + b)$$



激活函数  $\varphi$  有各种不同的形式。如果使用 step 函数，那神经元等价于一个线性分类器：

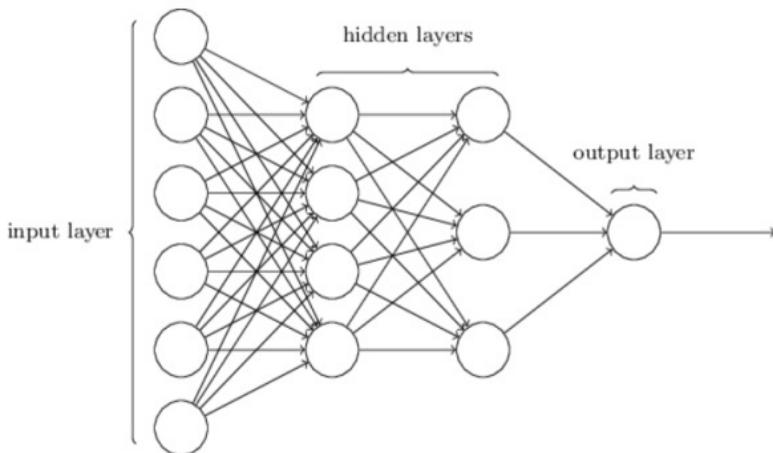
$$y = \begin{cases} 1 & \text{if } w \cdot x + b > 0 \\ 0 & \text{else} \end{cases}$$

这个分类器在历史上被称为感知器（Perceptron）。

## 多层神经网络

单层的感知器只能解决线性可分的问题。但实际上绝大多数问题都是非线性的，这时单层感知器就无能为力了。为此，我们可以把单个的neuron组成网络，让前一层neuron的输出做为下一层neuron的输入。组成如下图所示的神经网络。

由于非线性激活函数的存在，多层神经网络就有了拟合非线性函数的能力。由于历史的原因，多层神经网络也被称为 multilayer perceptrons (MLP)。



神经网络具有拟合非线性函数的能力。但是为了拟合不同的非线性函数，我们是否需要设计不同的非线性激活函数和网络结构呢？答案是不需要。universal approximation theorem 已经证明，前向神经网络是一个通用的近似框架。简单来说，对常用的sigmoid, relu等激活函数，即使只有一层隐藏层的神经网络，只要有足够多的神经元，就可以无限逼近任何连续函数。在实际中，浅层神经网络要逼近复杂非线性函数需要的神经元可能会过多，从而提升了学习的难度并影响泛化性能。因此，我们往往通过使用更深的模型，从而减少所需神经元的数量，提升网络的泛化能力。

## 机器学习的基本概念

深度神经网络是深度学习中的一类算法，而深度学习是机器学习的一种特例。因此，这一节我们在机器学习的一般框架下，介绍模型训练相关的基本概念，及其在Tensorflow中的实现。相关概念适用于包括NN在内的机器学习算法。

## 机器学习的常见问题

常见的机器学习问题，可以抽象为4大类问题：

- 监督学习

- 非监督学习
- 半监督学习
- 增强学习

根据训练数据是否有label，可以将问题分为监督学习（所有数据都有label），半监督学习（部分数据有label）和非监督学习（所有数据都没有label）。增强学习不同于前3种问题，增强学习也会对行为给出反馈（reward），但关注的是如何在环境中采取一系列行为，从而获得最大的累积回报。监督学习是目前应用最广泛，也是研究最充分的机器学习问题，本文接下来将重点介绍监督学习。

## 监督学习

在监督学习中，给定N个训练样本 $\{(x_1, y_1), \dots, (x_N, y_N)\}$ ，我们的目标是得到一个从输入到输出的函数： $f: X \rightarrow Y$ 。实际上，我们通常不会直接优化函数 $f$ ，而是根据问题的具体情况，选择一组参数化的函数 $f_\theta$ 将优化函数 $f$ 转换成优化参数 $\theta$ 。

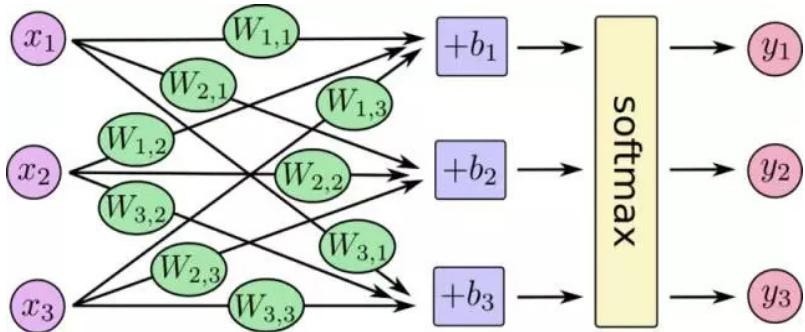
常见的分类问题和回归问题，都是监督学习的一种特例。线性分类器，深度神经网络等模型，都是为了解决这些问题设计的参数化后的函数。为了简单，我们以线性分类器 $y = f_\theta(x) = w^T x + b$ 为例，需要优化的参数 $\theta$ 为 $(w, b)$ 。

## 损失函数

为了衡量函数的好坏，我们需要一个客观的标准。在监督学习中，这个评判标准通常是一个损失函数 $L: Y \times Y \rightarrow \mathbb{R}$ 。对一个训练样本 $(x_i, y_i)$ ，模型的预测结果为 $\hat{y}$ ，那对应的损失为 $L(y_i, \hat{y})$ 。损失越小，表明函数预测的结果越准确。实际中，需要根据问题的特点，选择不同的损失函数。

二分类问题中，常用的logistic regression采用sigmoid + cross entropy作为loss。对常见的loss，tensorflow都提供了相应的函数。

```
loss = tf.nn.sigmoid_cross_entropy_with_logits(labels=labels, logits=y)
```



对多分类问题，如上图所示，我们可以将二分类的线性分类器进行扩展为N个线性方程： $y_i = \sum_j W_{i,j} x_j + b_i$ 。然后通过：

$$\text{softmax}(x)_i = \frac{\exp(x_i)}{\sum_j \exp(x_j)}$$

进行归一化，归一化后的结果作为每一类的概率。因此，多分类通常使用 softmax + cross entropy 作为损失函数。

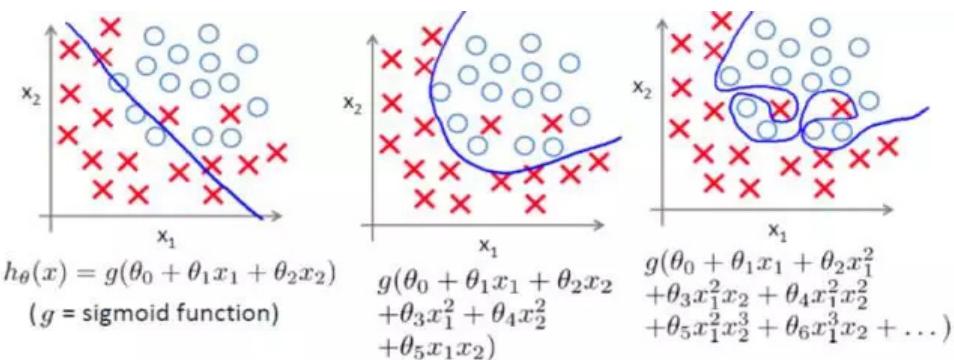
```
loss = tf.nn.softmax_cross_entropy_with_logits(labels=labels, logits=logits)
```

可以证明，对于二分类问题，采用 sigmoid cross entropy 和 softmax cross entropy 作为 loss，在理论上是完全等价的。此外，实际中通常采用直接计算 softmax cross entropy 而不是先计算 softmax，后计算 cross entropy，这主要是从数值稳定性的角度考虑的。

## 损失最小化和正则项

在定义了损失函数之后，监督学习问题可以转化为最小化实验损失  $\frac{1}{n} \sum_{i=1}^n L(y_i, f(x_i))$ 。实际中，为了保证模型的拟合能力，函数  $f$  的复杂度有时会比较高。如最图中最右边情况所示，如果训练样本数较少或者 label 有错误时，直接最小化实验损失而不对  $f$  加限制的话，模型容易过拟合。

因此，在样本数量较少或者标注质量不高的情况下，需要额外添加正则项 (regularizer)，保证模型的泛化能力。实际中根据不同需



求，可以选择不同的正则项。在神经网络当中，比较常见的是 $L_2$  norm 正则项： $R(w) = \frac{1}{2} \sum_{i=1}^n w_i^2$

在 tensorflow 中，通常有两种方法添加正则项。一种是根据需要，自己实现相应的 regularization loss，然后和其他 loss 相加进行优化。这种方法可以实现比较复杂的 regularizer。

```
weight_decay = tf.multiply(tf.nn.l2_loss(weights), wd, name='weight_loss')
```

对于常见的正则项，也可以使用 tensorflow 自带的功能，对相应的变量进行正则化。然后将系统生成的所有 regularization loss 和其他 loss 相加进行优化。

```
tf.contrib.layers.apply_regularization(tf.contrib.layers.l2_regularizer(wd),
weights)
tf.losses.get_regularization_losses()
```

## 梯度下降和反向传播

在定义了损失函数和正则项之后，最终正则化后的 loss 为：

$$\underline{\text{loss}}(\theta) = \frac{1}{n} \sum_{i=1}^n L(y_i, f(x_i)) + \alpha R(w)$$

有了 loss 函数，相应的参数可以通过标准的梯度下降算法进行优化求解。例如对线性分类器中的，可以通过如下的公式进行迭代更新：

$$w \leftarrow w - \eta \left( \alpha \frac{\partial R(w)}{\partial w} + \frac{\partial L(w^T x_i + b, y_i)}{\partial w} \right)$$

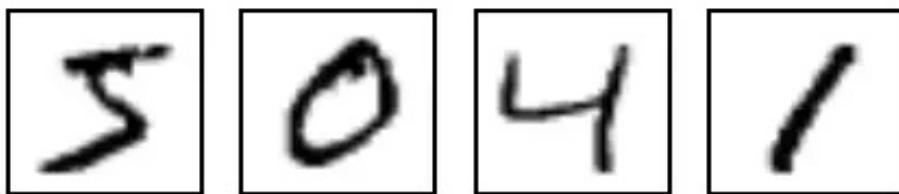
通过设定合适的 learning rate，参数会逐步收敛到局部/全局最优解。

反向传播可以看成梯度下降在神经网络中的一个推广，也是通过最小化loss函数，计算参数相对于loss的梯度，然后对参数进行迭代更新。具体的推导因为篇幅原因在这里略过了。在tensorflow中，只需要指定loss函数和步长（learning rate），optimizer可以自动帮我们完成梯度下降/反向传播的过程：

```
tf.train.GradientDescentOptimizer(learning_rate).minimize(loss)
```

## OCR 实战

最后本文通过一个OCR的例子，展示如何用Tensorflow实现Softmax分类器和MLP分类器。实验数据集采用著名的数字识别数据集MNIST。该数据集包含了60000张训练图片和10000张测试图片。数据集中的每一张图片都代表了0-9中的一个数字，图片的尺寸为 $28 \times 28$ 。



## Softmax分类器

我们首先实现一个简单的Softmax分类器。由于Tensorflow运行构建好的网络的过程比较复杂，为了提高开发效率和代码的复用性，我们将代码分成了3个主要模块，分别是Dataset模块，Net模块和Solver模块。

### 模型结构

对每一个Net类里，我们需要实现三个函数：

- `inference`

我们在inference函数中定义网络的主体结构。在tensorflow中，变量用`tf.Variable`表示。因为Softmax分类器是一个凸函数，

任何初始化都可以保证达到全局最优解，因此我们可以简单的将W和b初始化为0。Softmax分类器可以简单的通过一个矩阵乘法  $y = \text{tf.matmul}(\text{data}, W) + b$  后接一个tf.nn.softmax函数实现。

- `loss`

按照之前介绍的，为了保证数值稳定性，我们直接采用直接计算tf.nn.softmax\_cross\_entropy\_with\_logits的方式。

- `metric`

在训练完模型后，我们需要在validation或test集合上验证模型的性能。在测试集比较大时，我们无法一次得到模型在整个测试集上的结果，需要将测试集分成小的batch，在每个batch上进行测试，之后将每个batch的结果汇总起来。为此，tensorflow提供了tf.metrics模块，可以自动完成对每个batch进行评价，并将所有的评价汇总的功能。在这个例子里，我们是解决分类问题，因此可以使用tf.metrics.accuracy计算分类的准确率。

```
class Softmax(Net):  
    def __init__(self, **kwargs):  
        self.output_dim = kwargs.get('output_dim', 1)  
        return  
  
    def inference(self, data):  
        feature_dim = data.get_shape()[1].value  
        with tf.name_scope('weights'):  
            W = tf.Variable(tf.zeros([feature_dim, self.output_dim]))  
        with tf.name_scope('biases'):  
            b = tf.Variable(tf.zeros([self.output_dim]), name='bias')  
        with tf.name_scope('y'):  
            y = tf.matmul(data, W) + b  
        with tf.name_scope('probs'):  
            probs = tf.nn.softmax(y)  
        return {'logits': y, 'probs': probs}  
  
    def loss(self, layers, labels):  
        logits = layers['logits']
```

```

with tf.variable_scope('loss'):
    loss = tf.reduce_mean(
        tf.nn.softmax_cross_entropy_with_logits(labels=labels, logits=logits))
return loss

def metric(self, layers, labels):
    probs = layers['probs']
    with tf.variable_scope('metric'):
        metric, update_op = tf.metrics.accuracy(
            labels=tf.argmax(labels, 1), predictions=tf.argmax(probs, 1))
    return {'update': update_op, 'accuracy': metric}

```

Dataset

In versions of TensorFlow before 1.2, we recommended using multi-threaded, queue-based input pipelines for performance. Beginning with TensorFlow 1.2, however, we recommend using the `tf.contrib.data` module instead.

从Tensorflow1.2开始，Tensorflow提供了基于`tf.contrib.data`的新API。相比原来基于`QueueRunner`和`Coordinator`的API，代码结构简洁了很多。所以我们在`Dataset`类中采用了新的API，实现数据读取。

```

class MNIST(Dataset):
    def __init__(self, **kwargs):
        self.data_dir = kwargs.get('data_dir', None)
        self.split = kwargs.get('split', 'train')
        self.count = kwargs.get('count', None)
        self.buffer_size = kwargs.get('buffer_size', 10000)
        self.batch_size = kwargs.get('batch_size', 50)
        if self.split not in ['train', 'validation', 'test']:
            raise ValueError('unsupported dataset mode!')
        # download mnist data
        images, labels = load_dataset(self.data_dir, self.split)
        # build dataset
        dataset = tf.contrib.data.Dataset.from_tensor_slices((images, labels))
        if self.buffer_size is not None:

```

```

dataset = dataset.shuffle(buffer_size=self.buffer_size)
dataset = dataset.repeat(self.count)
dataset = dataset.batch(self.batch_size)
with tf.name_scope('input'):
    self._iterator = dataset.make_one_shot_iterator()
    self._batch = self._iterator.get_next()

def batch(self):
    return self._batch

def shape(self):
    return self._iterator.output_shapes

```

我们首先读取了numpy.array格式的mnist数据集images, labels。然后通过tf.contrib.data.Dataset.from\_tensor\_slices将之转换成tf.contrib.data.Dataset格式。

之后我们可以设置对Dataset的遍历次数（None代表无限次），batch\_size以及是否对数据集进行shuffle。最后，我们采用最简单的make\_one\_shot\_iterator()和get\_next()，得到网络的基本数据单元batch。按默认配置，每个batch含有50张图和其对应的label。

## Solver

最后我们介绍Solver类。Solver类主要包含五个函数：

- `build_optimizer`

因为网络比较简单，这里我们选用最基本的随即梯度下降算法tf.train.GradientDescentOptimizer，并使用了固定的learning rate。

- `build_train_net`、`build_test_net`

这两个函数的作用类似，都是将Dataset中的数据和Net中的网络结构串联起来。在最后我们调用tf.summary.scalar将loss添加到summary中。tensorflow提供了强大的可视化模块tensorboard，可以很方便的对summary中的变量进行可视化。

- `train_net`

在train\_net的开头，我们完成了Graph, Saver, summary等模块

的初始化。然后通过`summary_writer.add_graph(tf.get_default_graph())`，将网络结构打印到`summary`中。

之后初始化`tf.Session()`，并通过`session.run`运行对应的操作。在tensorflow使用了符号式编程的模式，创建Graph的过程只是完成了构图，并没有对数据进行实际运算。在Session中运行对应的操作时，才真正对底层数据进行操作。

- `test_net`

和`train_net`类似，`test_net`主要完成了各种模块的初始化，之后读取模型目录文件下的`checkpoint`文件中记录的最新的模型，并在测试集中进行测试。

```
class BasicSolver(Solver):  
    def __init__(self, dataset, net, **kwargs):  
        self.learning_rate = float(kwargs.get('learning_rate', 0.5))  
        self.max_steps = int(kwargs.get('max_steps', 2000))  
        self.summary_iter = int(kwargs.get('summary_iter', 100))  
        self.summary_dir = kwargs.get('summary_dir', 'summary')  
        self.snapshot_iter = int(kwargs.get('snapshot_iter', 100000))  
        self.snapshot_dir = kwargs.get('snapshot_dir', 'cache')  
        self.dataset = dataset  
        self.net = net  
  
    def build_optimizer(self):  
        with tf.variable_scope('optimizer'):   
            train_op = tf.train.GradientDescentOptimizer(self.learning_rate).minimize(  
                self.loss)  
        return train_op  
  
    def build_train_net(self):  
        data, labels = self.dataset.batch()  
        self.layers = self.net.inference(data)  
        self.loss = self.net.loss(self.layers, labels)  
        self.train_op = self.build_optimizer()  
        for loss_layer in tf.get_collection('losses') + [self.loss]:  
            tf.summary.scalar(loss_layer.op.name, loss_layer)
```

```
def build_test_net(self):  
    data, labels = self.dataset.batch()  
    self.layers = self.net.inference(data)  
    self.metrics = self.net.metric(self.layers, labels)  
    self.update_op = self.metrics.pop('update')  
    for key, value in self.metrics.iteritems():  
        tf.summary.scalar(key, value)  
  
def train(self):  
    self.build_train_net()  
    saver = tf.train.Saver(tf.trainable_variables())  
    init_op = tf.global_variables_initializer()  
    summary_op = tf.summary.merge_all()  
    summary_writer = tf.summary.FileWriter(  
        os.path.join(self.summary_dir, 'train'))  
    summary_writer.add_graph(tf.get_default_graph())  
    with tf.Session() as sess:  
        sess.run(init_op)  
        for step in xrange(1, self.max_steps + 1):  
            start_time = time.time()  
            sess.run(self.train_op)  
            duration = time.time() - start_time  
            if step % self.summary_iter == 0:  
                summary, loss = sess.run([summary_op, self.loss])  
                summary_writer.add_summary(summary, step)  
                examples_per_sec = self.dataset.batch_size / duration  
                format_str = ('step %6d: loss = %.4f (%.1f examples/sec)')  
                print(format_str % (step, loss, examples_per_sec))  
                sys.stdout.flush()  
            if (step % self.snapshot_iter == 0) or (step == self.max_steps):  
                saver.save(sess, self.snapshot_dir + '/model.ckpt', global_step=step)  
  
def test(self):  
    self.build_test_net()  
    saver = tf.train.Saver()  
    init_op = [  
        tf.global_variables_initializer(),
```

```

        tf.local_variables_initializer()
    ]
summary_op = tf.summary.merge_all()
summary_writer = tf.summary.FileWriter(
    os.path.join(self.summary_dir, 'test'))
summary_writer.add_graph(tf.get_default_graph())
with tf.Session() as sess:
    sess.run(init_op)
checkpoint = tf.train.latest_checkpoint(self.snapshot_dir)
if not os.path.isfile(checkpoint + '.index'):
    print("[error]: can't find checkpoint file: {}".format(checkpoint))
    sys.exit(0)
else:
    print("load checkpoint file: {}".format(checkpoint))
    num_iter = int(checkpoint.split('-')[-1])
saver.restore(sess, checkpoint)
while True:
    try:
        sess.run(self.update_op)
    except tf.errors.OutOfRangeError:
        results = sess.run([summary_op] + self.metrics.values())
        summary = results[0]
        metrics = results[1:]
        for key, metric in zip(self.metrics.keys(), metrics):
            print("{}: {}".format(key, metric))
        summary_writer.add_summary(summary, num_iter)
        break

```

下图是tensorboard中可视化的网络结构，和loss的统计。可以看到，tensorboard对我们进行分析提供很好的可视化支持。

最终程序的输出结果如下：

```

step 100: loss = 0.3134 (116833.0 examples/sec)
step 200: loss = 0.4800 (113359.6 examples/sec)
step 300: loss = 0.3528 (114410.9 examples/sec)
step 400: loss = 0.2597 (105278.7 examples/sec)
step 500: loss = 0.3301 (106834.0 examples/sec)
step 600: loss = 0.4013 (115992.9 examples/sec)

```

```
step    700: loss = 0.3428 (112871.5 examples/sec)
step    800: loss = 0.3181 (113913.7 examples/sec)
step    900: loss = 0.1850 (123507.2 examples/sec)
step   1000: loss = 0.0863 (125653.2 examples/sec)
step   1100: loss = 0.2726 (105703.2 examples/sec)
step   1200: loss = 0.4849 (115736.9 examples/sec)
step   1300: loss = 0.2986 (100582.8 examples/sec)
step   1400: loss = 0.2994 (103973.8 examples/sec)
step   1500: loss = 0.2626 (102500.1 examples/sec)
step   1600: loss = 0.0996 (107712.0 examples/sec)
step   1700: loss = 0.2523 (114912.4 examples/sec)
step   1800: loss = 0.3264 (105703.2 examples/sec)
step   1900: loss = 0.2911 (114975.4 examples/sec)
step   2000: loss = 0.2648 (132312.4 examples/sec)
accuracy: 0.9194999993324
```

可以看到，一个简单的线性模型可以达到92%的准确率。我们猜测数字识别这个问题应该不是线性可分，因此使用更复杂的非线性分类器应该可以得到更好的结果。

## MLP分类器

由于我们采用了模块化的设计，各个模块直接基本是解耦合的。因此将Softmax分类器替换成MLP非常容易，我们只需要重新实现Net层就可以了。

```
class MLP(Net):
    def __init__(self, **kwargs):
        self.output_dim = kwargs.get('output_dim', 1)
        return
    def inference(self, data):
        with tf.variable_scope('hidden1'):
            hidden1 = linear_relu(data, 128)
        with tf.variable_scope('hidden2'):
            hidden2 = linear_relu(hidden1, 32)
        with tf.variable_scope('softmax_linear'):
```

```
y = linear(hidden2, self.output_dim)
probs = tf.nn.softmax(y)
return {'logits': y, 'probs': probs}

def loss(self, layers, labels):
    logits = layers['logits']
    with tf.variable_scope('loss'):
        loss = tf.reduce_mean(
            tf.nn.softmax_cross_entropy_with_logits(labels=labels, logits=logits))
    return loss

def metric(self, layers, labels):
    probs = layers['probs']
    with tf.variable_scope('metric'):
        metric, update_op = tf.metrics.accuracy(
            labels=tf.argmax(labels, 1), predictions=tf.argmax(probs, 1))
    return {'update': update_op, 'accuracy': metric}

def linear_relu(x, size, wd=0):
    return tf.nn.relu(linear(x, size, wd), name=tf.get_default_graph().get_name_scope())

def linear(x, size, wd=0):
    weights = tf.get_variable(
        name='weights',
        shape=[x.get_shape()[1], size],
        initializer=tf.contrib.layers.xavier_initializer())
    biases = tf.get_variable(
        'biases', shape=[size], initializer=tf.constant_initializer(0.0))
    out = tf.matmul(x, weights) + biases
    if wd != 0:
        weight_decay = tf.multiply(tf.nn.l2_loss(weights), wd, name='weight_loss')
        tf.add_to_collection('losses', weight_decay)
    return out
```

为了证明MLP的效果，我们构造了一个含有2层hidden layer的神经网络。代码结构和Softmax分类其大致一样，就不做过多解释了。因为线性层在网络中多次出现，我们将他抽象为一个可以复用的函

数。另外，为了让graph在tensorboard中的可视化效果更好，我们将相关的变量和操作，通过with `tf.variable_scope('hidden1')`：放置在同一个variable\_scope下面。这样所有相关变量和操作在tensorboard都会收缩成一个可以展开的节点，从而提供更好的可视化效果。

最终的网络结果和运行结果如下所示：

```
#  
step    100: loss = 0.4675 (49113.6 examples/sec)  
step    200: loss = 0.2348 (53200.2 examples/sec)  
step    300: loss = 0.1858 (51922.6 examples/sec)  
step    400: loss = 0.1935 (49554.6 examples/sec)  
step    500: loss = 0.2634 (51552.4 examples/sec)  
step    600: loss = 0.1800 (51871.2 examples/sec)  
step    700: loss = 0.0524 (51225.0 examples/sec)  
step    800: loss = 0.1634 (50606.9 examples/sec)  
step    900: loss = 0.1549 (56239.0 examples/sec)  
step   1000: loss = 0.1026 (54755.9 examples/sec)  
step   1100: loss = 0.0928 (51871.2 examples/sec)  
step   1200: loss = 0.0293 (50864.7 examples/sec)  
step   1300: loss = 0.1918 (54528.1 examples/sec)  
step   1400: loss = 0.1001 (48725.7 examples/sec)  
step   1500: loss = 0.1263 (50003.6 examples/sec)  
step   1600: loss = 0.0956 (54176.0 examples/sec)  
step   1700: loss = 0.1012 (52025.6 examples/sec)  
step   1800: loss = 0.3386 (53471.5 examples/sec)  
step   1900: loss = 0.1626 (54641.8 examples/sec)  
step   2000: loss = 0.0215 (54528.1 examples/sec)  
accuracy: 0.970499992371
```

可以看到，使用了包含2层hidden layer的简单神经网络，分类的准确率已经提升到了97%，大大优于简单的线性分类器。证明了模型选择对最终性能的重要影响。

完整代码[下载](#)。

## 作者简介

**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。



在微信上关注我们



## InfoQ

国内最好的原创技术社区，一线互联网公司核心技术人员提供优质内容。订阅 InfoQ，看全球互联网技术最佳实践。做技术的不会没听过 QCon，不会不知道 InfoQ 吧？——冯大辉  
从事技术工作，或有兴趣了解 IT 技术行业的朋友，都值得订阅。——曹政



关注「InfoQ」回复“二叉树”，看十位大牛的技术初心，不同圈子程序员的众生相。



### 聊聊架构

以架构之“道”为基础，呈现更多的务实落地的架构内容。



关注「聊聊架构」

和百位架构师共聊架构



### 细说云计算

探讨云计算的一切，关注云平台架构、网络、存储与分发。这里有干货，也有闲聊。



关注「细说云计算」

回复“群分享”，  
看云计算实践干货分享文章



### AI前线

提供最新最全AI领域技术资讯、一线业界实践案例、业界技术分享干货、最新AI论文解读。



关注「AI前线」

回复“AI”，下载《AI前线》系列迷你书



### 前端之巅

紧跟前端发展，共享一线技术，不断学习进步，攀登前端之巅。



关注「前端之巅」

回复“京东”，看京东如何做网站前端监控



### 移动开发前线

关注移动开发领域最前沿和第一线开发技术，打造技术分享型社群。



关注「移动开发前线」

回复“群分享”，看移动开发实践干货文章



### 高效开发运维

常规运维、亦或是崛起的DevOps，探讨如何IT交付实现价值。



关注「高效开发运维」

回复“DevOps”，四篇精品文章领悟DevOps

北 京 | 伦 敦 | 纽 约 | 旧 金 山 | 圣 保 罗 | 上 海 | 东 京

# QCon 全球软件开发大会2018

主办方 **Geekbang** 极客邦科技 **InfoQ**

## [北京站]

北京·国际会议中心

演讲：2018年4月20–22日 培训：2018年4月18–19日

纵览20大热门专题

# 最低优惠7折进行时

现在报名每张立减2040元

团购享受更多优惠 截至2017年12月31日

大会官网：[www.qconbeijing.com](http://www.qconbeijing.com)  
访问官网获取更多前沿技术趋势

如有任何问题，欢迎咨询  
电话：15110019061  
微信：qcon-0410

