19 AI 框架

清华大学教授胡事民：统一计算图——机器学习框架 Jittor 的创新与探索

整理：智源社区 刘鑽

在第二届北京智源大会 AI 框架专题中，胡事民教授做了主题为《统一计算图： 机器学习框架 Jittor 的创新与探索 》的报告。

胡事民是清华大学计算机系教授，主要研究方向为计算机图形学、虚拟现实、智能信息处理、系统软件等。在 ACM TOG/SIGGRAPH、IEEE TVCG、IEEE CVPR 等重要刊物和国际会议上发表论文 100 余篇。现为中国计算机学会副理事长、亚洲图形学学会副主席，并担任 Computational Visual Media 主编和 CAD 等多个期刊编委。

在报告中，胡事民介绍了他团队研发的机器学习框架——计图（Jittor），它基于“统一计算图”的概念，创新地使用了元算子融合和动态编译技术，目前在多种任务性能上超越国外主流平台。除此之外，计图还在易用性、灵活性以及模型算法覆盖度上做了大量改进。

以下是智源社区编辑整理的胡事民演讲要点。

一、为什么要做 Jittor

鉴于国外主流深度学习框架 Tensorflow 和 Pytorch 已经构建起了完整的产业链和庞大用户群体，那为什么清华**大学要研发“ 计图”（Jittor）这样一个 AI 框架？为了回答这个问题，胡事民首先介绍了深度学习框架的一些背景知识。**

胡事民认为，以深度学习为代表的新一代人工智能技术已经产生非常大的影响，深度学习在某种程度上已经成为科学研究和工程应用的一种新范式，数据驱动的思想已经深入人心。

机器学习框架是人工智能的一种核心技术，它不仅要负责机器学习、模型的训练、推理，管理人工智能应用所需要的大规模数据和模型，而且还要负责底层的计算设备调度和资源申请，所以如下图 1 所示，它是在计算设备之上、在应用之间的一个框架层或者平台层。

图 1： 机器学习框架所处层次

接下来，胡事民回顾了深度学习框架十年来的演进情况。2008 年，图灵奖获得者 Bengio 所领导的实验室提出了第一个比较成熟的框架 Theano。这个框架奠定了很好的基础，但是很遗憾在 2017 年就停止维护了，其主要成员后来参加了 Tensorflow。在 2013 年时，加州伯克利大学博士生贾扬清提出了 Caffe 平台，后来他加盟了谷歌和 Facebook，并在 Facebook 期间发布了颇具影响力的 Caffe2。但真正在工业界、学术界产生重大推动作用的，是 2015 年由谷歌开源的 Tensorflow。因为谷歌的强大影响力，其 2017 年发布的 Tensorflow1.0 版本在学术界一度取得了垄断性的应用地位。众所周知，在 2017、2018 年左右，绝大多数和深度学习相关的论文实验都是用 Tensorflow 来做的。这股势头直到 2018 年 Pytorch 1.0 的出现才发生了变化。Pytorch 最初发布于 2016 年，特色是提出了动态图的概念。到了 2019 年，学术界将近 70% 的文章实验是用 Pytorch 这个平台来做的，由此可一窥动态图思想的主导性地位。

图 2：深度学习框架的十年间演进

胡事民认为，我们国家人工智能研究尽管非常活跃，也取得了很多大的进展，但是整体上的研究布局并不均衡：我们在算法应用方面做得比较多，在学习平台框架方面却做得比较少，而且我们的算法应用研究大多是基于国外平台 Tensorflow 和 Pytorch 等，有被“卡脖子”的风险。这类风险的迹象已经在蔓延，比如华为“清单事件”中，华为被谷歌禁用安卓系统；比如今年 1 月 6 号美国特朗普签署法案，要禁止人工智能软件的出口等。

胡事民指出，“为什么做中国自己的机器学习平台”？首先是自主创新的需要，比如 EDA，中国在 20 世纪 90 年代在 EDA 算法上是非常领先的，但由于没有注重平台研发和发展，导致现在非常被动。第二，做平台也是人才培养的需要，我们需要培养自己的一代新的人工智能领军人才。

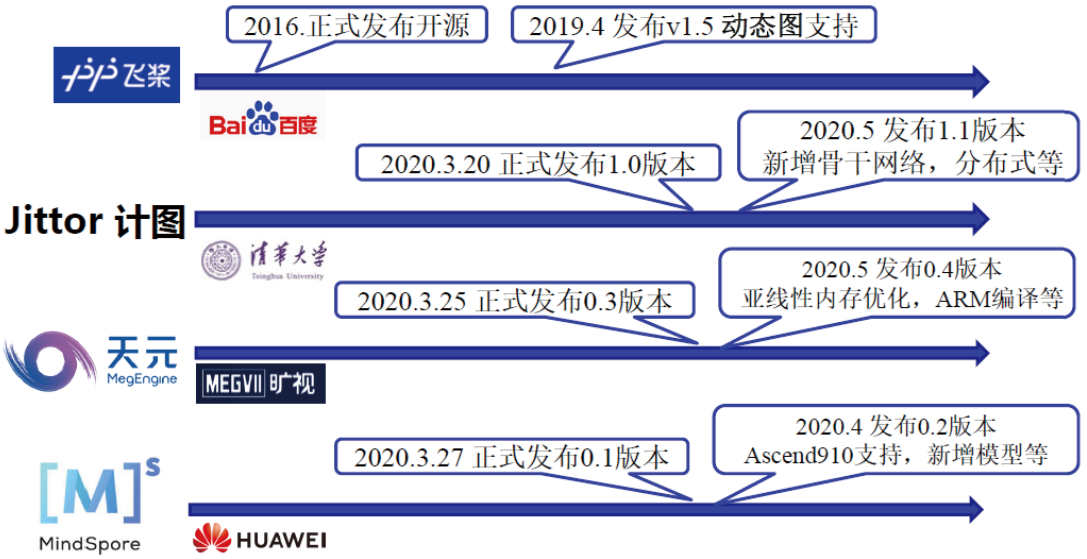


图 3：深度学习框架在中国的进展

那么，构建机器学习平台的难点在哪里？胡事民认为它首先要对机器学习算法本身有深刻的理解，对图形图像应用，也包括自然语言等要有精确的把握。特别重要的是，我们需要对底层系统软件要有很好的掌握。

胡事民介绍，他们团队尽管是做图形学的，但是在视觉、机器人、操作系统方面已经有很多积累。比如从 2009 年开始，团队在操作系统和软件安全方面，已经在包括 ACM、TOCS、USENIX 以及 ATC 等等重要学术会议上发表了 10 多篇文章。这样通过构建机器学习平台，可以使得同学们把机器学习、图形图像应用和软件系统等结合起来。

以上内容，便是计图（Jittor）平台诞生的主要缘起。

二、Jittor 框架的架构

清华计图（Jittor）机器学习框架发布于 2020 年 3 月 20 日，它基于元算子和统一计算图来构建。胡事民介绍，计图目前尽管尚存很多不足，但是已经实现的部分相比两个国际主流平台 Tensorflow 和 Pytorch，性能有 5%–153% 的提升，目前支持 30+ 个骨干网络，27 个主流 GAN 模型，此外还支持图像分类、图像检测、图像分割、机器人强化学习等模型。

如下图 4 所示，这个架构的第一创新点是在系统层引进“统一计算图”。机器学习框架的架构有静态、动态之分，静态图高效，动态图好用，但是这两者很难兼得，计图则将它们融合在一起； 能够实现它的根本原因，在于计图的第二个创新点，即有一套元算子融合的理念： 在机器学习框架里算子非常多，它的开发和优化困难比较大，计图希望通过元算子融合和动态编译优化解决这个根本性的问题。

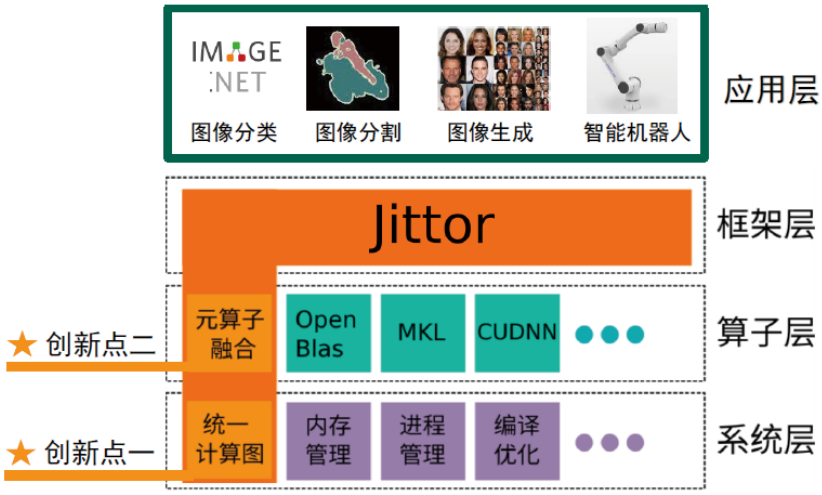


图 4：Jittor 整体架构

2.1 计图平台的六个特性

接下来，胡事民介绍了计图平台的六个特性:

特性一：反向传播闭包。元算子是反向传播闭包，它的反向传播也是元算子，那么元算子融合起来的算子（即合成的算子），它的反向传播也是可以用元算子表达出来的，这样就使得我们能够把前向、反向的计算统一起来。也就是说 **Jittor 可以统一“ 前向”和“ 反向”计算图**。这是非常重要的一点。另外，**计图能够支持高阶导数的计算**，这也是现在很多机器学习框架所欠缺的。

特性二：算子动态编译和运行时优化。用户编写的算子模型在运行的时候，会生成高性能代码，进行动态编译的优化。

特性三：统一内存管理，节省显存。计图把 GPU 和 CPU 统一起来，GPU 内存耗尽时可以用 CPU 弥补，当中可以做数据迁移。

特性四：极简的自定义算子开发功能。提出 Code 算子，用户可以在 Python 中内连 C++，数行代码即可完成高性能算子的开发。

特性五：辅助模型迁移工具。Pytorch 在现在的学术界社区里面已经成为主导的框架，计图希望架构接口能跟它相似，便于学生们上手。同时计图提供一键迁移功能，使得 Pytorch 代码仅通过一键操作便能够方便地转换成计图的代码。

特性六：自有的模型库。除了 2020 年 5 月 30 日发布的 GAN 模型库、6 月底发布的计图分割库模型库，更多模型库比如检测库、3D 点云库、3D 网格库等将陆续发布。

接下来，胡事民重点介绍了计图的两个创新点。

2.2 创新点一：元算子融合

Tensorflow 有 2000 多个算子，Pytorch 有 700 多个算子，这么庞大的算子库的维护和优化非常困难，对用户来说不方便。计图通过对神经网络计算所需要的基本算子进行归纳总结分析，**提出 18 种元算子，并通过算子融合，可以覆盖算法所需要的算子**。这是计图的基本出发点，只需要对元算子做优化，效率很高，也容易开发，可以做统一优化。

对于传统的机器学习，需要要跟一些深度学习算子比如卷积、池化等等打交道来支持应用，而计图的做法是增加一层元算子层，通过 3 种类别的元算子构造深度学习算子再做机器学习应用，包括：**重索引算子、重索引化简算子、元素级算子**。

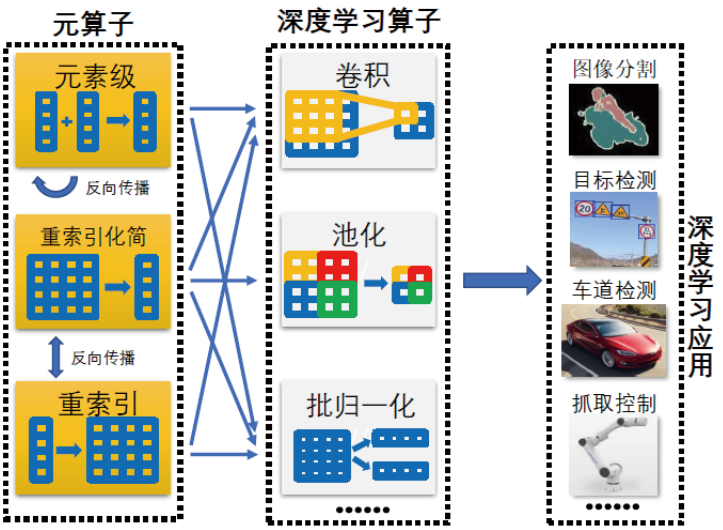


图 5：计图增加的元算子层

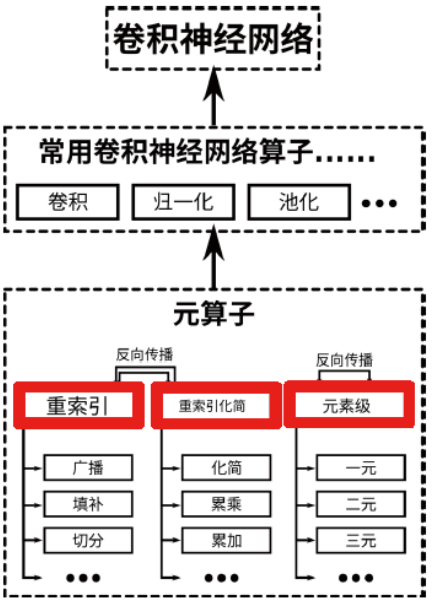


图 6：元算子分成三类

第一类是重索引算子，它是一类一对多映射关系的算子。对于机器学习经常碰到的运算，比如广播、填补、切分，以及对一个向量重复几份变成矩阵等，计图则把这些作为基本的元算子，这就是所谓的重索引算子。

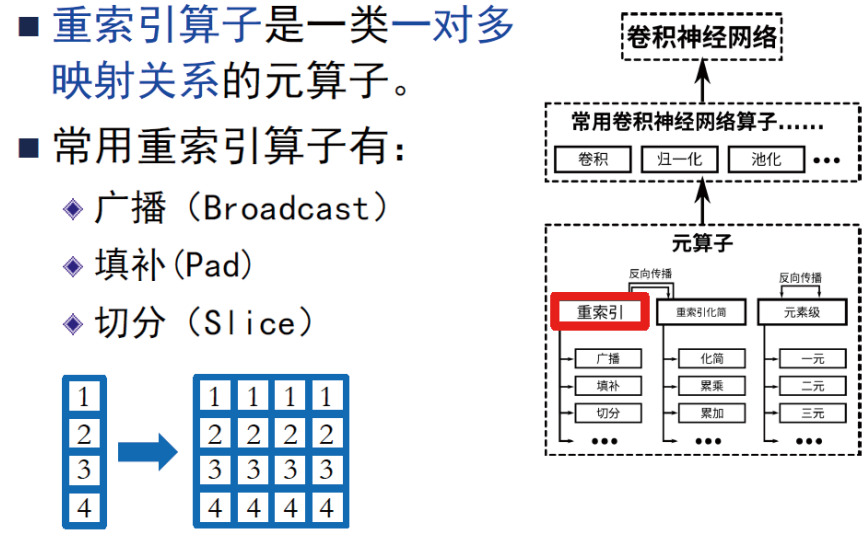


图 7：重索引算子

第二类叫重索引化简算子。比如累加、累乘、取均值操作，或把一个矩阵通过每行累加变成一个向量等经常碰到的运算，将这些操作基本抽象出来后，便可成为计图的一个元算子。

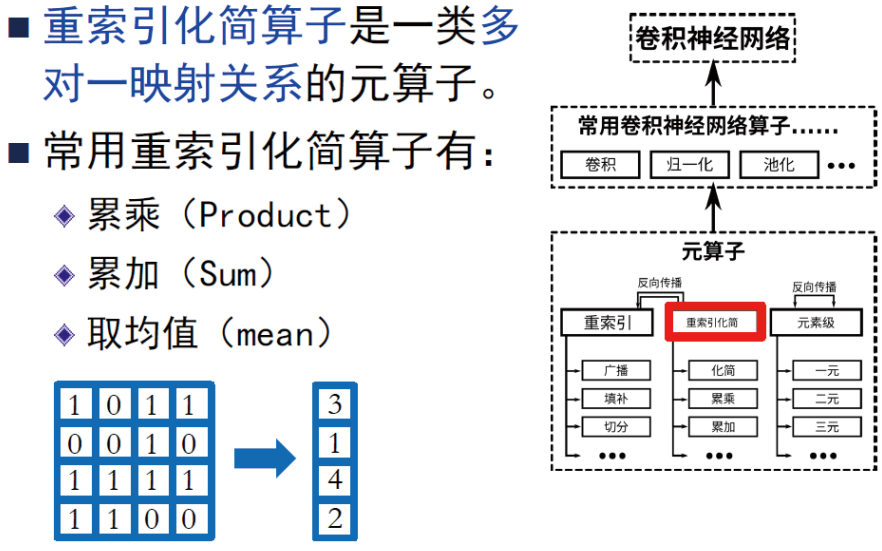


图 8：重索引化简算子

第三种是元素级算子。比如加法算子，把两个元素相加变成新的向量。

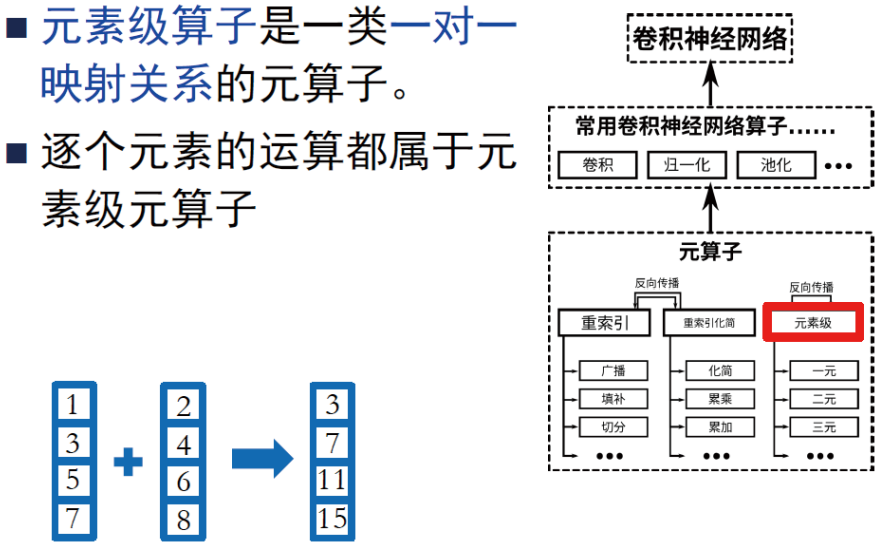


图 9：元素级算子

有了三类元算子之后，还有一个非常重要的特点，就是它们是反向传播闭包的。重索引反向传播是重索引化简，重索引化简的反向传播是重索引，元素级的反向传播还是元素级的，比如加变减。

这样，我们就可以用这三类元算子来构造机器学习的算子，比如卷积层，计图使用广播、下标、重排、乘法和累加实现卷积。

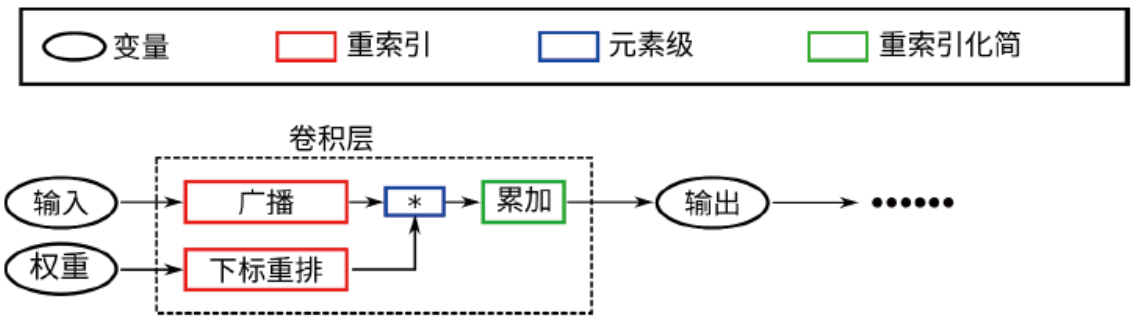


图 10：使用元算子实现卷积层

这样，有了卷积构造之后反卷积自动就有了，反向传播也可以用这个元算子自动构造出来，也是通过广播这种操作，重排就变成化简了。

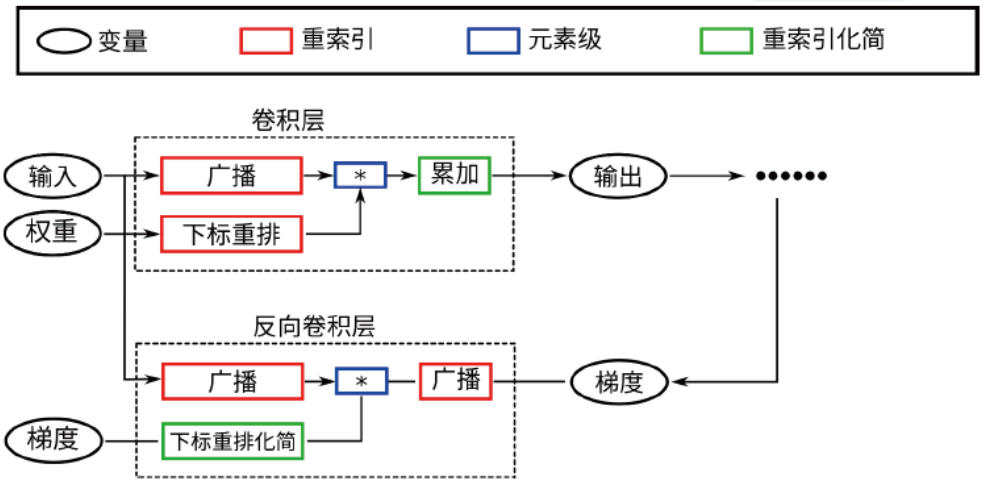


图 11：元算子自动生成反向传播卷积层

那么，计图元算子的完备性如何？胡事民介绍，目前的元算子还无法覆盖全部机器学习的算子，比如在图像分类、图像分割、图像生成方面百分之百都能覆盖，但是目标检测只能做到 96%，实例分割则做到了 93% 等。对不能覆盖的算子，计图讲提供一种 Code 算子工具，以完成 100% 算子覆盖。如下图所示，元算子覆盖率的计算标准，为该应用使用的算子有多少种可以通过元算子实现。

应用类型	图像分类 Resnet	图像分割 Deeplabv3+	图像生成 Cycle Gan
覆盖率	100%	100%	100%

应用类型	目标检测 SSD	实例分割 Mask RCNN	3D点云 PointNet++	3D网格 MeshCNN
覆盖率	96%	93%	87%	80%

注：覆盖率的计算标准为该应用使用的算子有多少种可以使用元算子实现

图 12：元算子的覆盖情况

2.3 创新点二：统一计算图

计算图是深度学习框架里面用来描述模型的数据结构。例如何恺明关于 Resnet 的论文中有一张很长的图，从顶上到底下，就是通过计算图把这个模型展示出来的。下图是一个图像分割的例子，输入图像的目标是生成语义分割，需要通过卷积、激活等等构造模型，这样的结构便是“计算图”。

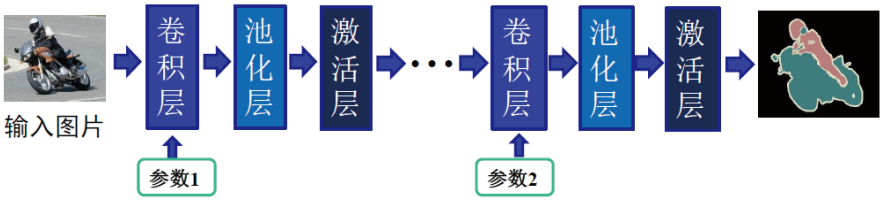


图 13：一个图像分割神经网络的计算图实例

机器学习框架有动态静态之争。早期是静态图，它是把全图一次性发送到计算设备上做优化和计算。比如早期的 Tensorflow 就是这样的。

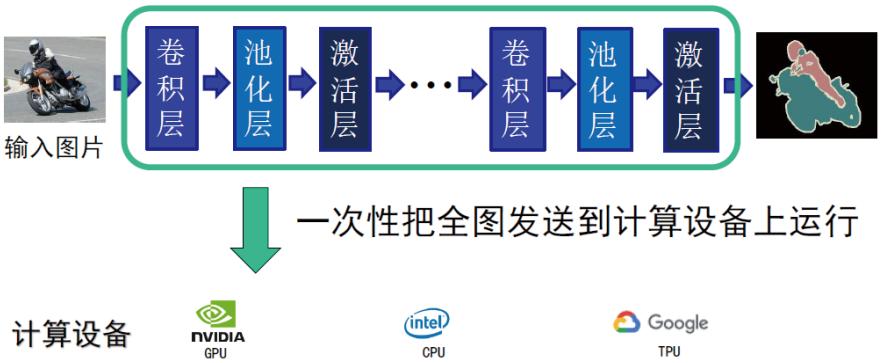


图 14：静态计算图

动态图 Pytorch 的想法是把算子动态发布到计算设备上运行，这种模式下改代码很方便，允许用户对模型做灵活调整，比如加一个卷积层等。因此动态图的出现使得 Pytorch 抓住机遇一下超越了 Tensorflow，成为业界最主流的框架。

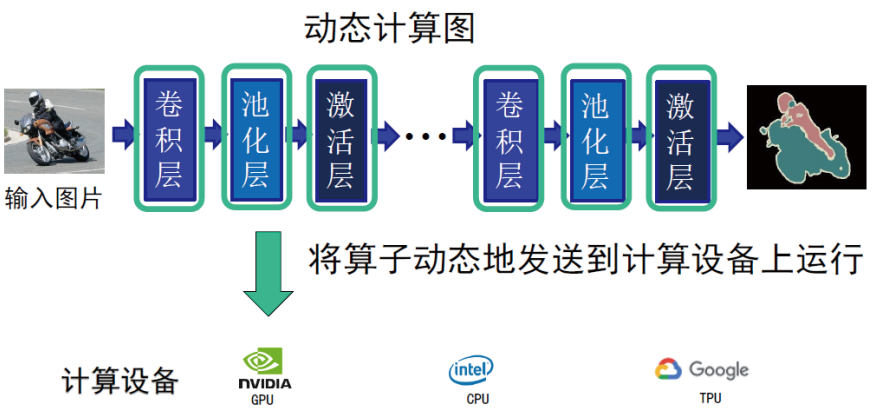


图 15：动态计算图

动、静态图的优点缺点都很明显，静态图的代表是 Tensorflow1.0，优点是高效，缺点是灵活性差；动态图的代表是 Pytorch，包括后来 Tensorflow2.0 也采用了动态图，它的优点是易用、灵活，但是效率比较低。

为了解读上述两者不可兼得的问题，计图提出了“统一计算图”，通过元算子融合的概念，通过反向传播闭包把它们整合在一起。计图希望易用性和动态计算图一样好，效能跟静态图一样好，主要的思路是动态切分，把动态图切成静态子图，对静态子图做优化。

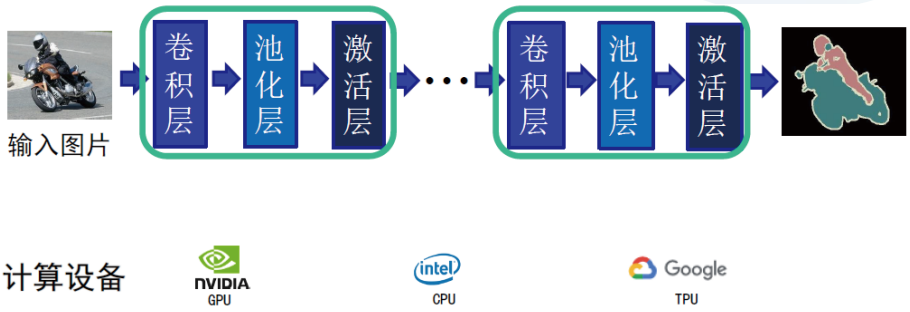


图 16：统一计算图：统一静态计算图和动态计算图

胡事民介绍，除了计算图整合在一起外，**计图实现了多种统一**:

第一是动态图和静态图的统一，易用、同时高效。

第二是前向、反向管理的统一，统一管理前向反向图，支持高阶导数。

第三是 GPU、CPU 内存管理的统一，来突破 GPU 显存限制。

第四是同步异步运行接口的统一，使得数据读取、内存拷贝、模型计算可以同时进行，提升性能。

第五是可以管理多次迭代的计算图，使得架构可以跨迭代的进行融合优化。

下面就是统一计算图把前向和反向计算图两者融合起来的一个示意图。

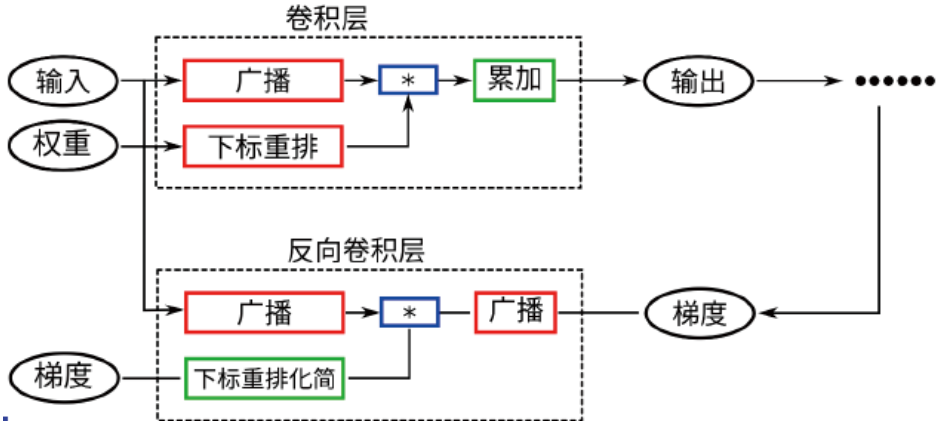


图 17：统一计算图：通过反向传播闭包，统一前向和反向计算图

胡事民指出，传统机器学习框架区分前向和反向传播。统一计算图希望把多次迭代的前向、反向拼接起来，减少大量的重复计算，提升灵活性和对话空间。所以一般来说，应用计图框架，它的计算性能上只要优化好了，一定会比 Pytorch 和 Tensorflow 要快。

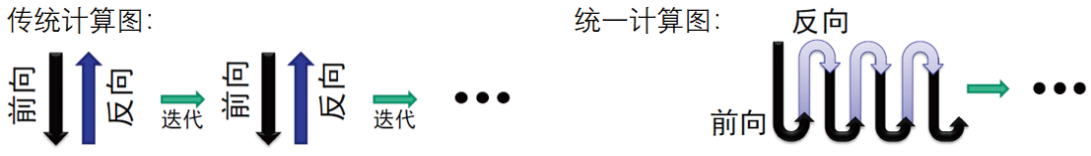


图 18：传统计算图与统一计算图在前向后向传播上的比较

下图对比了几个机器学习框架：自动微分三个平台都是支持的，动态图也都支持了，同步接口 Pytorch 现在没有，统一内存现在前两家也做不到，迭代融合也只有计图能做到。计图是有前瞻性的、创新性的平台。

	Tensorflow	PyTorch	Jittor
自动微分	√	√	√
动态图	√	√	√
同步接口	eager mode √	√	√
异步接口	√	×	√
统一内存	不	×	√
跨迭代融合	×	×	√

图 19：计算图特征比较

三、Jittor 的特点与新进展

接下来，胡事民介绍了计图的五大特色：

1. **动态编译。**在平台里内置元算子编译器，可以把 Python 代码动态编译成高性能 C++ 代码。胡事民认为有些专家评价计图更像一个编译器，这并不正确。因为计图的定位取决于平台是为什么人服务、用户是谁，但它也不排除利用很多软件工程、软件系统的思想来做平台优化。此外，计图还内置了 LLVM 兼容的优化编译遍(Pass)，可以根据硬件的设备自动优化编译代码，对 C++ 代码进一步优化，生成对底层设备友好的底层代码。

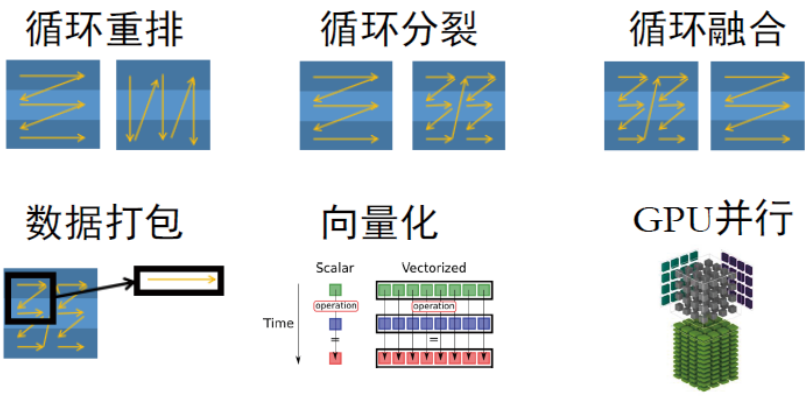


图 20：计图使用动态编译

2. **骨干网络。**计图现在已经支持很多重要的骨干网络，比如 AlexNet、VGG、ResNet、SqueezeNet、Inception、GoogleNet、ShuffleNet、MobileNet、MnasNet 以及最近支持的 Res2Net。如下图所示跟 Pytorch 对比，计算性能在几乎所有骨干网络上都能得到很大的提升。

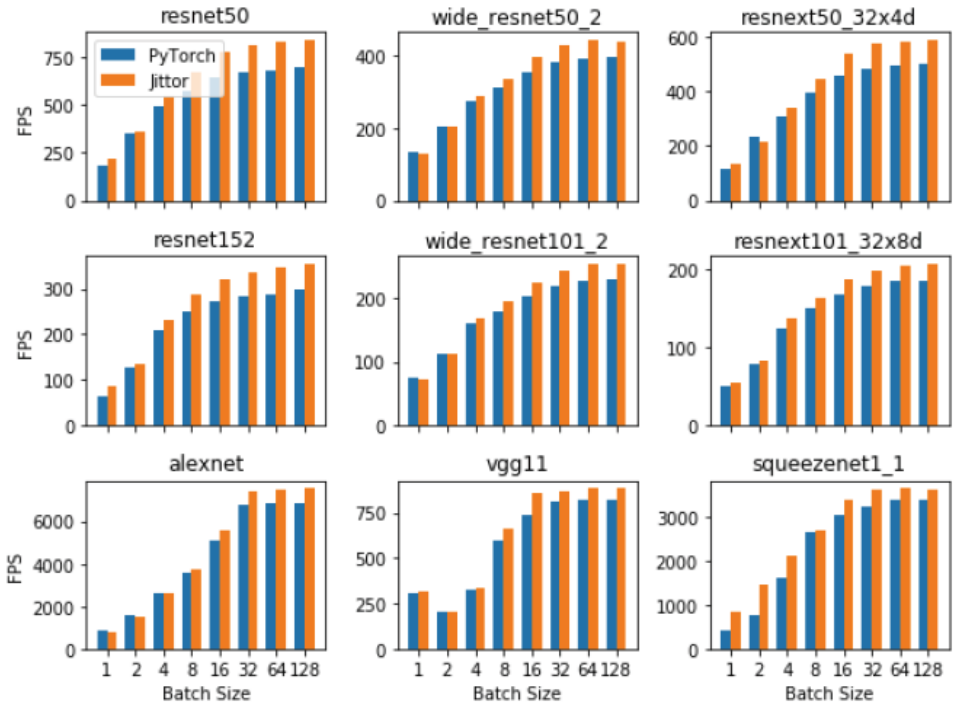


图 21：计图在骨干网络上的性能提升

3. **GAN 模型库。**GAN 网络是非常著名的网络，在 Pytorch 社区里最受欢迎的就是两个 GAN 模型 pix2pix 和 CycleGAN。现在有 27 种主流的 GAN 模型，计图都把它做了移植并做了分析。这些模型影响都非常大，有的引用达一万八千多次。这 27 种模型在计图平台上都能得到很大的性能提升，比如 WGAN–GP 模型，在计图上的性能是原来的 2.83 倍。最著名的是 CycleGAN，在计图上的性能是原来的 1.39 倍，最差的也能做到 1.03 倍。胡事民认为，计图能使得一半以上模型效能提升 1 倍以上，一半的模型指标到 200%，这是非常不容易的事情。

GAN	Year	Cites	GAN	Year	Cites
GAN	2014	18575	StarGAN	2018	869
Deep Convolutional GAN	2015	6186	Coupled GAN	2016	802
Pix2Pix	2017	5161	Energy-Based GAN	2016	734
CycleGAN	2017	4678	BEGAN	2017	693
Wasserstein GAN	2017	4276	PixelDA	2017	670
Conditional GAN	2014	3217	BicycleGAN	2017	457
Wasserstein GAN GP	2017	2822	Enhanced Super-Resolution GAN	2018	365
Wasserstein GAN DIV	2017	2822	Relativistic GAN	2018	184
InfoGAN	2016	1826	DRAGAN	2017	122
Context Encoder	2016	1712	Boundary-Seeking GAN	2017	88
Least Squares GAN	2017	1312	Cluster GAN	2019	36
Adversarial Autoencoder	2015	1193	Semi-Supervised GAN	2018	17
Auxiliary Classifier GAN	2017	1152	Softmax GAN	2017	11
UNIT	2017	971			

图 22：主流 GAN 模型库

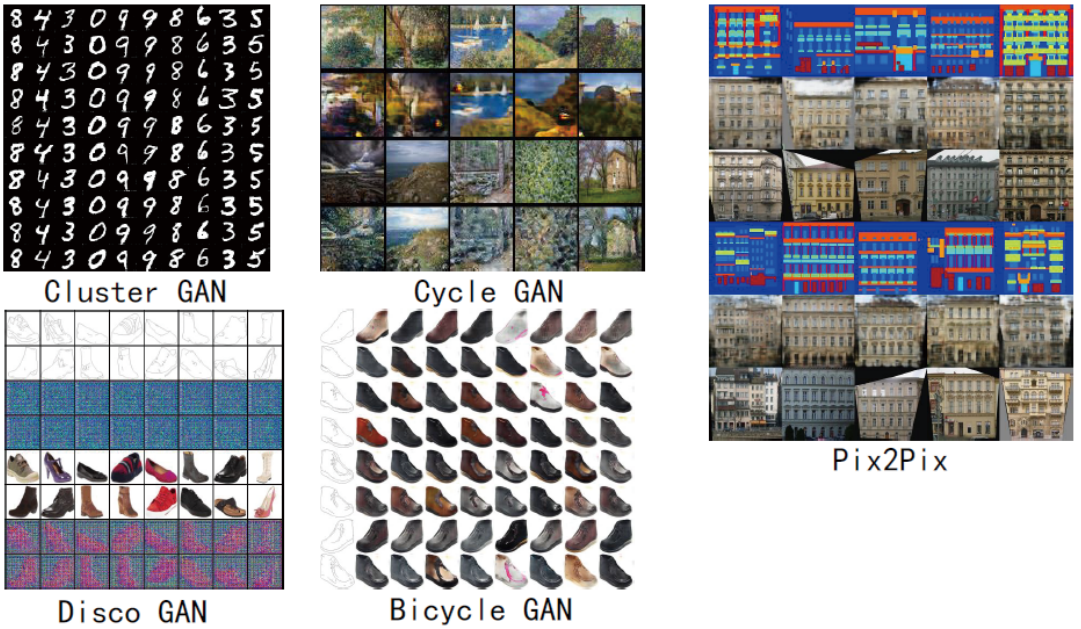


图 23 ： GAN 模型库部分模型生成效果

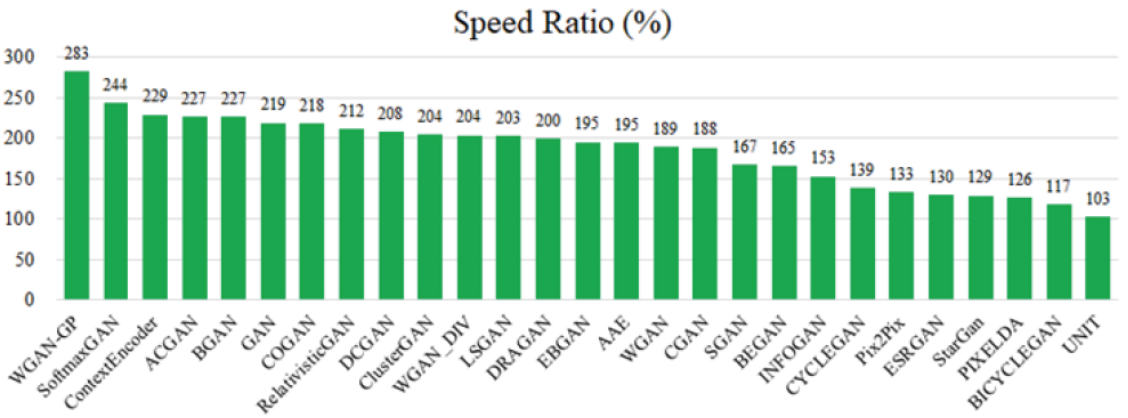


图 24： GAN 模型库性能对比

4. **提供一键转换功能**。从 Pytorch 改到计图并不麻烦，一键操作系统就帮忙把代码改了，原来的工作拿过来都可以用，辅助转换，非常方便。

5. **分布式**。在计图框架下，可以很方便进行多卡训练，编写多机代码。比如单卡训练就一个命令行，多卡训练命令前面只加几个字符来指定设备，再增加前面一小段就行了。

```
# 单卡训练代码
python3.7 -m jittor.test.test_resnet
# 分布式多卡训练代码
mpirun -np 4 python3.7 -m
 jittor.test.test_resnet
# 指定特定显卡的多卡训练代码
CUDA_VISIBLE_DEVICES="2,3" mpirun -np
 2 python3.7 -m jittor.test.test_resnet
```

图 25 ： 单卡训练和分布式训练

接下来，胡事民介绍了计图正在推进的一些工作。

第一，**支持 Res2net**。这是南开大学提出的新骨干网络。这个骨干网络在很多应用上性能非常强，而把它做到计图平台后，除了让它本身性能很强外，在计算效能上还有 10% 的提升。

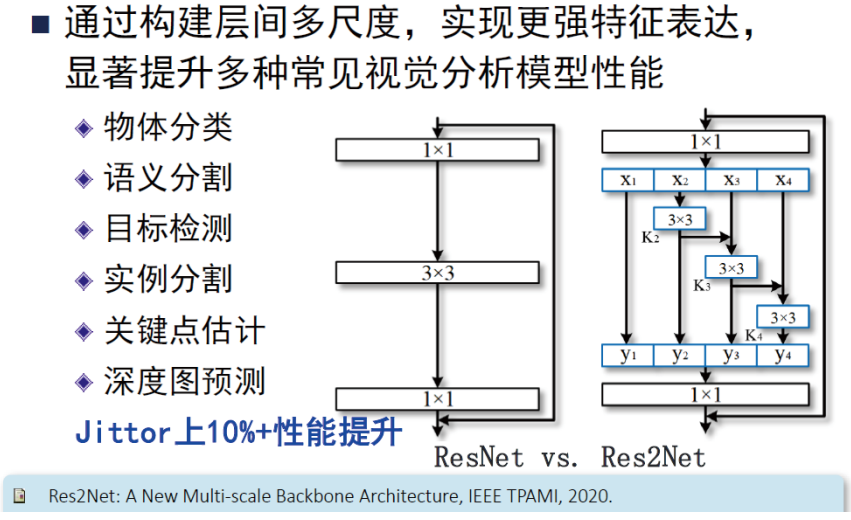


图 26 ： Res2Net 神经网络骨干模型

第二，**实现了 PointNet++**。在实践中发现，计图在分类的准确性上已经超过了原来论文中的数据，从原来的 91.9% 提高到 92.3%。模型代码已公开在 Jittor-oneline-first。

参考资料

计图的参考资源：https://cg.cs.tsinghua.edu.cn/jittor/

亚马逊云服务（AWS）上海 AI 研究院院长张峥：高性能高可用的深图计算平台——亚马逊 DGL 的实践与应用

整理：智源社区 尹朝晖

亚马逊云服务（AWS）上海 AI 研究院院长、上海纽约大学教授（学术休假）张峥在第二届北京智源大会上作了题为《Deep Graph Library an update》的报告。

张峥教授在大规模分布式计算理论与实践、机器学习的交叉领域被公认为经验丰富的世界级专家。他拥有伊利诺伊大学厄巴纳 – 香槟分校的博士学位，毕业后加入惠普中央研究院，是计算机协会会员，也是 SIGOPS APSYS 研讨会和 CHINASYS 研究社区的创始人。张峥教授是开源项目 DGL 以及 MXNet 的发起人之一。2013 年加入上海纽约大学之前，曾在全球领先的科技公司微软亚洲研究院任系统与网络研究部分的副院长。2019 年出任亚马逊云服务（AWS）上海 AI 研究院首任院长。

深度学习和图网络的结合是近年来机器学习领域最有潜力和最有突破性的一个方向。张峥教授认为，如果将深度学习看作一个算法，是没有数据结构做支撑的，而图（Graph）的重要性在于将非结构化的数据变为结构化的数据，给深度学习算法提供了数据结构支撑。

一、基于图网络的深度学习

深度学习和图网络的结合是近年来深度学习领域最有潜力和最有突破性的一个方向，最近一年公开发表的文章里，热度最高的就是图神经网络（如图 1）。

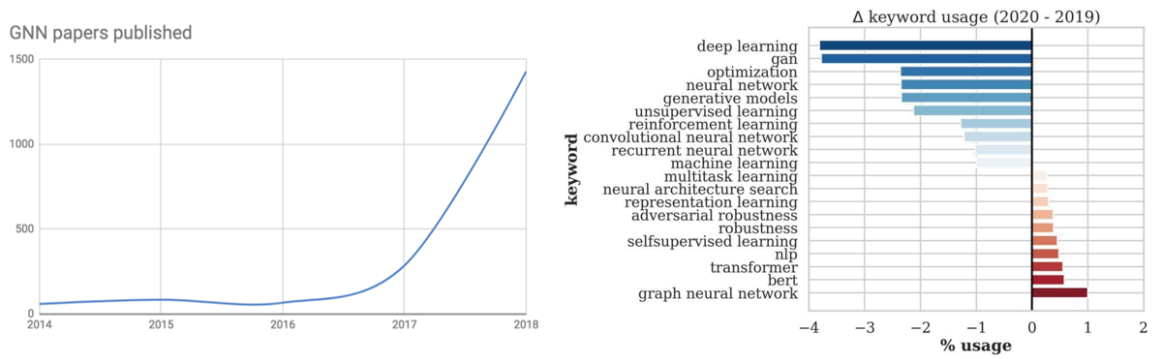


图 1：2014–2018 年历年发表的关于 GNN 的论文数（左）和 2019 年发表论文中关键词分布（右）

接下来张峥用几个例子介绍了图网络，首先以分子网络为例，里面的原子和原子之间的连接就是一种图，这种图比较小，节点数从 100 到 1000。另一种典型的图网络是知识图谱，比如一个人的国籍、教育、工作、获奖等就可以连接成一张图，这种图网络节点数大，从 1 百万到 1 亿之间。节点数更多的是推荐系统，例如同一个用户可能买了多种商品，或者同一个商品被不同的用户购买，也可以构成一个图，这个图还可以和知识图谱结合起来，形成更大的图（图 2）。

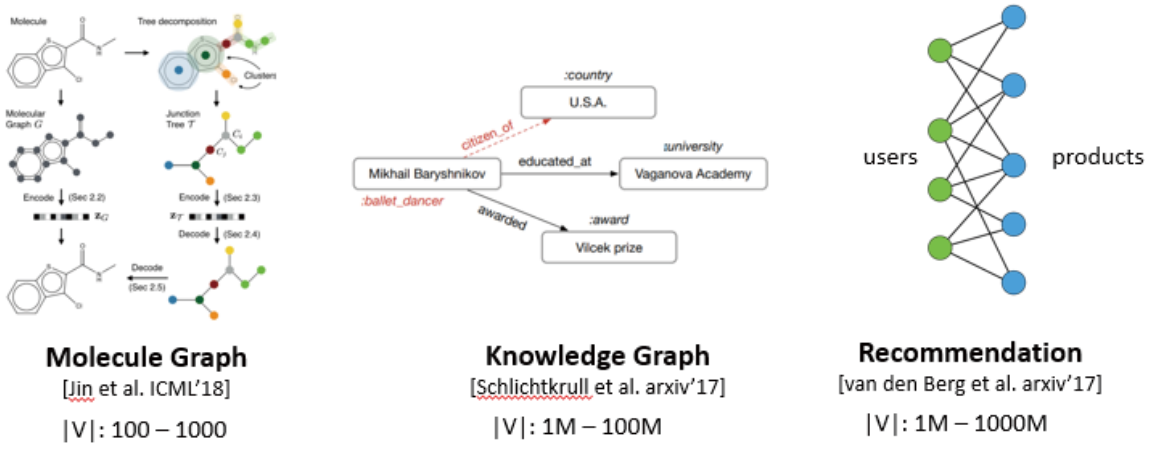


图 2：图网络的 3 个示例

近期，为应对全球新冠病毒 COVID-19，亚马逊上海 AI 实验室联合其他合作伙伴刚刚完成了一项“老药新用”的工作，构建了大规模药物重定位知识图谱 DPKG(Drug Repurposing Knowledge Graph)，以及一套完整的用于药物重定位研究的机器学习工具，以便帮助相关研究人员更有效的对新冠病毒进行药物重定位研究。用于计算某种现成的药和一种新的疾病之间是否存在某种关联，从而判断这种现成的药是否可以被用于治疗相似的疾病（图 3）。这项技术已经获得了广泛应用，并取得了初步成效。

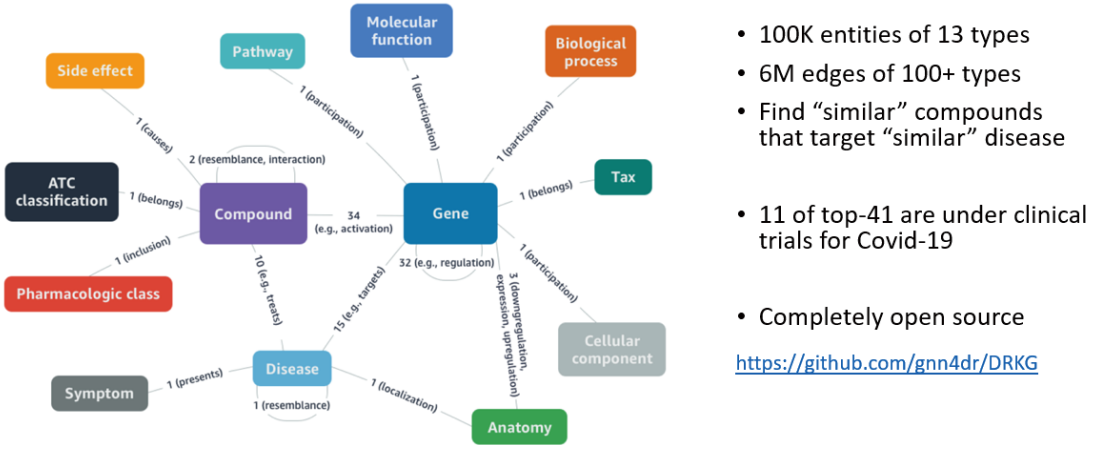


图 3：大规模药物重定位知识图谱 DPKG(Drug Repurposing Knowledge Graph)

二、DGL 的开发过程、设计和性能

DGL 从 2018 年初开始开发，至 2018 年底 NeurIPS 大会上正式发布了 V0.1 版本，2019 年迭代了 4 个版本。一年半时间从一个学术界的科研成果转变为可以供产业界应用的框架。目前最新的版本是 V0.43，预计今年内会发布 V0.5 版本，增加分布式训练功能。

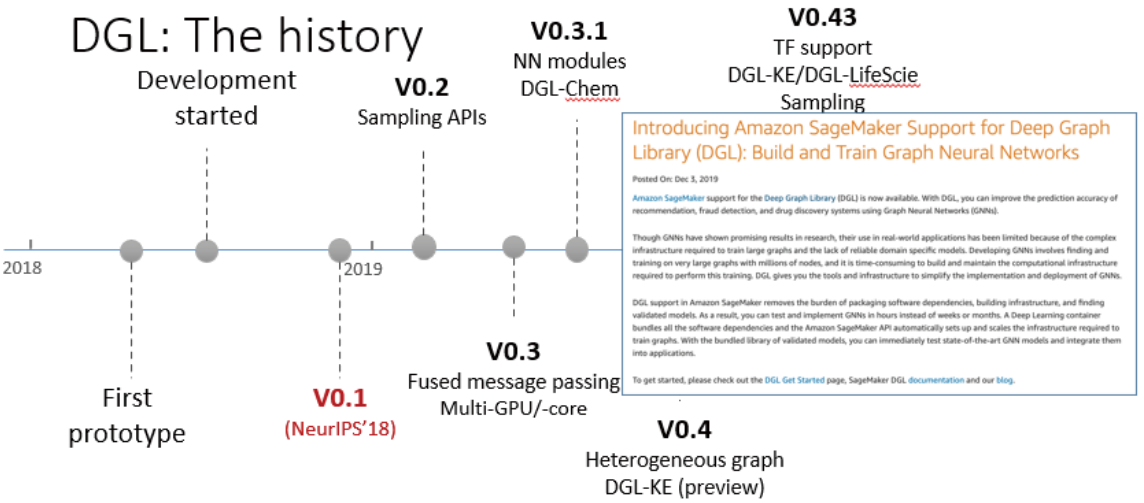


图 4：DGL 的发展历史

DGL 的架构包括三层，从下往上依次为平台 (Platform)，后端 (Backend) 和深图库 (Deep Graph Library)。其中平台层同时支持 PyTorch、MXNet 和 TensorFlow。DGL 开发时有两个主要的目标，第一个目标是针对用户达到前后向兼容，前向兼容方便研究人员快速打磨新的模型，后向兼容意味着可以在不同的神经网络框架下运行。第二个目标是快速和可扩展。

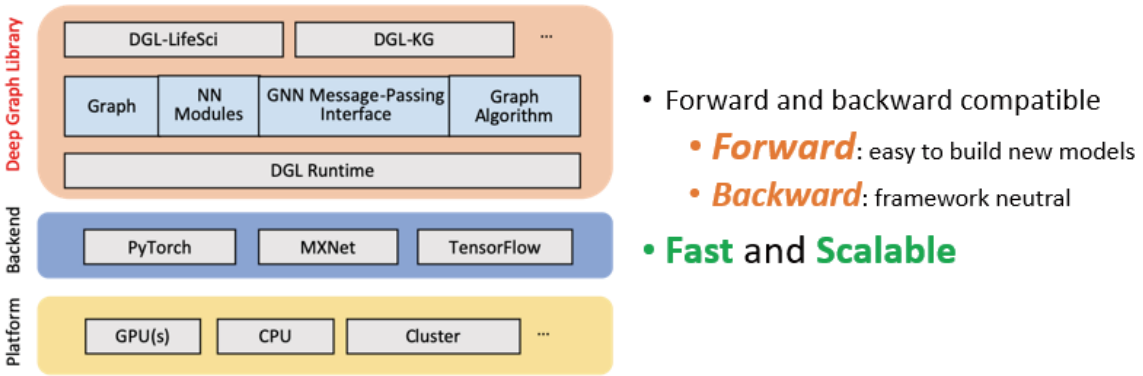


图 5：DGL 的架构

从 DGL 与 PyG 的在性能上的比较来看，DGL 在产业界用得更多，而 PyG 在学术界使用更多。在 CPU 上，DGL 的性能总体优于 PyG；在 GPU 上，凡是用到边的特征的模型，DGL 的表现都比 PyG 好，比较简单的模型两者性能接近。

Dataset	Model	CPU		GPU	
		DGL	PyG	DGL	PyG
Node Classification					
REDDIT	SAGE	**17.82**	99.47	0.453	**0.403**
REDDIT	GAT	**801.22**	OOM	OOM	OOM
OGBN-ARXIV	SAGE	**4.295**	8.389	0.100	**0.098**
OGBN-ARXIV	GAT	**24.12**	43.21	**0.139**	0.234
OGBN-PROTEIN	R-GCN	**156.6**	373.8	**3.707**	OOM
Link Prediction					
ML-100K	GCMC	**0.191**	1.569	0.028	**0.012**
ML-1M	GCMC	**0.625**	40.47	**0.056**	0.103
ML-10M	GCMC	**12.85**	OOM	**0.687**	OOM

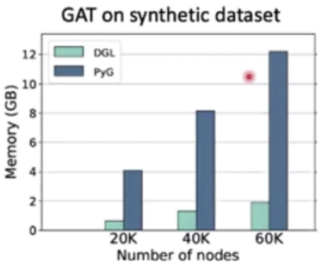


图 6：DGL 与 PyG 的性能比较

DGL 的设计理念是框架中立 (framework neurality)，即在模型中只处理最少和最关键的部分 (例如 DGLGraph)，而不考虑其余部分，例如自动求导、内存管理和稠密张量运算等都交给不同的平台处理。DGL 一开始也没有考虑自己做稀疏张量运算，经过实验，后期也将稀疏张量运算加入到了上述最少和最关键的部分。

为了做到框架中立，模型在不同的框架下跑起来有一些移植的工作量，理论上说性能叶会有区别。但 DGL 做到了模型本身没有变化，而且性能影响很小。DGL 的 Pytorch 的版本与 PyG 相比，在节点较少时，PyG 优于 DGL，而节点较多时，DGL 优于 PyG。与 Tensorflow 原生的图网络 GraphNets 相比，DGL 整体更好 (如图 7)。

Porting effort

GNN Model	Change Type			Change / Total
	I	II	III	
GCN [11]	4	12	14	30 / 103
GAT [13]	4	26	7	37 / 95
GraphSage [12]	4	13	8	25 / 85
GIN [24]	4	4	0	8 / 37
SGC [25]	4	4	4	12 / 50

I. Model class inheritance
II. Sub-modules and parameter initialization
III. Framework-specific operators

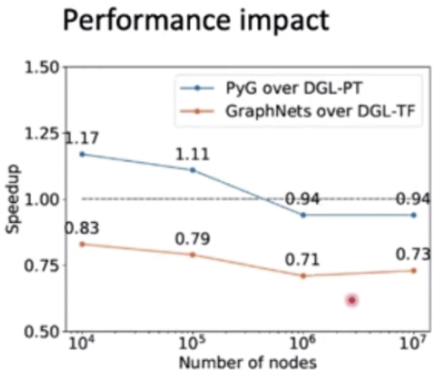


图 7：框架中立的代价

三、强 AI

张峥认为要做到强 AI，需要把大图和小图两类图有机联系在一起。大图是马太效应的结果，有些特别大的图会变成越来越大，比如全中国作为一张微博图或者淘宝支付图。这些图的变化出现较慢，一个图内部的变化量大。另一类是特别小的图，例如一句话、一张照片或者一段视频背后的一个动态变换图，这种图会快速生成。只有把上述两类图结合起来，才能做到强 AI。从这个角度来讲，需要一个数据结构。假如把深度学习看作一个算法，目前缺少一个数据结构，而算法和数据结构是紧密联系在一起的。张峥教授认为，如果将深度学习看作一个算法，是没有数据结构做支撑的，而图 (Graph) 的重要性在于将非结构化的数据变为结构化的数据，给深度学习算法提供了数据结构支撑。

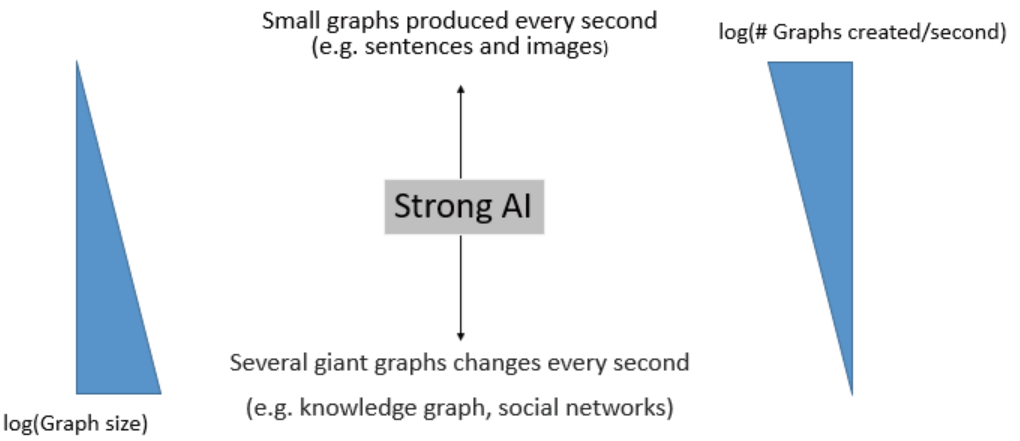


图 8：强 AI 概念图

四、DGL 的更多应用场景

张峥教授团队继 DGL 之后，又开源了一款专门针对大规模知识图谱嵌入表示的新训练框架 DGL–KE，旨在让研究人员和工业界用户方便、快速地在大规模知识图谱数据集上进行机器学习训练任务。DGL–KE 比当前最快的同类系统（如 Facebook 发布 Pytorch–BigGraph）更快。上文提到的 DRKG 就是 DGL–KE 的一个应用。目前国内外多家医药公司已经开始使用 DGL–KE。张峥教授团队近期发布的另一个基于 DGL 的框架是 DGL–LifeSci，这是面向化学和生物领域的 GNN 算法库，可以用于生物制药行业对药分子毒性、水溶性和分子合成路线的研究。DGL–LifeSci 也比现有的框架快 2.5 ～ 13 倍。

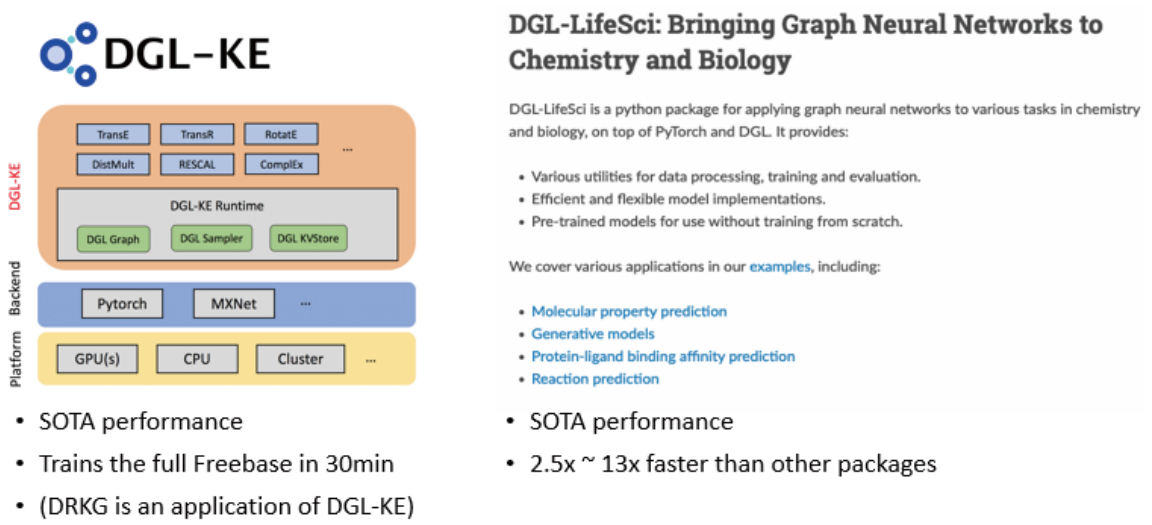


图 9：DGL–KE 和 DGL–LifeSci

阿里巴巴白俊杰：PyTorch——从 AI 研究到生产

整理：智源社区 亓麟

白俊杰是开放式神经网络交换 ONNX 的联合创始人，PyTorch 和 Caffe2 的核心开发者，曾任 Facebook 人工智能技术架构负责人。于 2020 年 2 月加入阿里巴巴，现供职于阿里巴巴阿里云智能事业群，负责大规模 AI 基础相关设施的建设。

本次演讲白俊杰通过解析 PyTorch 的设计理念和部署细节，阐释了 PyTorch 如何帮助用户从研究到生产进行过渡。主要内容包括了 PyTorch 五大设计理念（PyTorch 的 Eager&Graph 双执行模式、动态性、分布式、硬件加速、简化）介绍，以及三大部署策略（快速实验、训练规模化、部署规模化），最后介绍了“PAI 平台”作为阿里云智能框架的重要平台，有哪些重要工作要做。整个演讲不仅有对具体设计理念的描述，也有结合代码段进行的详细说明。

一、PyTorch 五大特性

首先我们来看一下 PyTorch 是什么。PyTorch 是通过许多基础模块集成的深度学习框架，帮助用户从研究到生产的整个过程得以实现，我们可以根据图 2 先来看一下 PyTorch 的五大特性:

- 双执行模式（Eager & Graph–Based Execution）
- 动态神经网络（Dynamic Neural Networks）
- 分布式训练（Distributed Training）
- 硬件加速（Hardware Acceleration）
- 简化复杂运算（Simplicity Over Complexity）

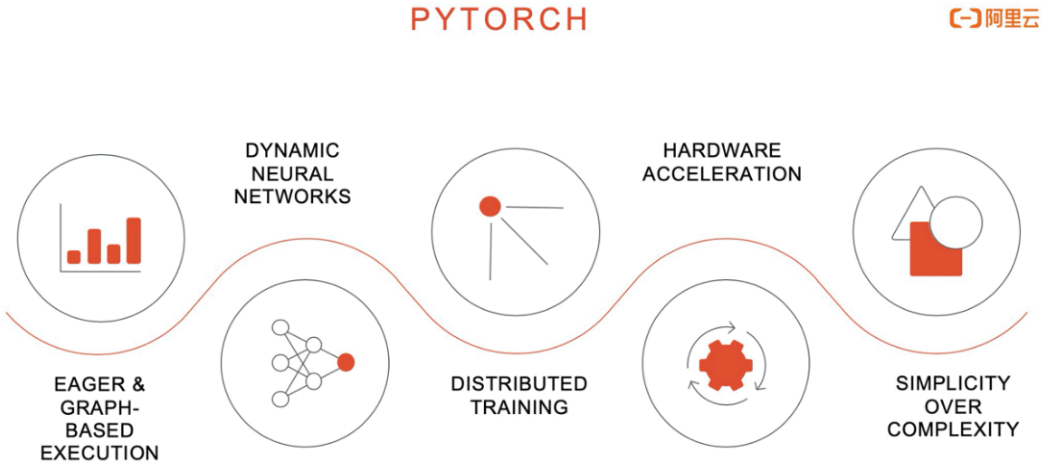


图 1：PyTorch 的五大特性

首先是 Eager 和 Graph 两种执行模式。PyTorch 从设计之初，就支持两种模式的运行方式：动态即时执行模式（Eager–Based Execution）和图模式（Graph–based Execution）。Eager 模式下用户可以即时编写具有良好表

达能力的代码，不需要关心底层如何实现的，只需要把注意力聚焦在如何表达自己的想法即可；PyTorch 同样支持基于图的执行（Graph-based），用户先将数据和计算模式定义在一个计算流模式（Data Flow）下，以获得计算上的高性能收益，同时也可以预先进行算法优化。一个计算过程如果能在 Eager 模式下执行，那么同样也可以在 Graph 模式下执行，这两种执行模式是可以进行平稳过渡的，我们的开发人员花了很多精力去分别开发这两种模式以及两个模式之间的无缝切换。

第二点是模型的动态性。这里的动态是一个泛指概念，而不是简单指数据形状（Data Shape）或者模型结构（Model Structure）的单一的动态性，PyTorch 在整个设计过程中都在考虑如何让框架的使用更加灵活，同时在动态性的前提下又能运行得更快。

第三点是分布式训练。近年来产业界的趋势是模型越来越大，随之带来计算复杂度越来越高给产业要求的高效运算带来了很大的挑战。PyTorch 整个设计过程中，很早就开始支持分布式数据并行的运算模式，最近也加入了专门的模块来支持加速。

第四点是硬件加速。过去几年基于特征或者其他专门特性的硬件设计不断取得进步，PyTorch 最近也和很多合作者共同探索，把 PyTorch 软件模块和特殊硬件进行整合，以达到加速训练过程和推断过程的目的。

第五点是对复杂操作的简化。这是我认为 PyTorch 成功的最重要特点之一，主要体现在 API 是非常简单而统一的，在很多层面上对用户的起到了很大帮助。得益于简单的 API 和程序组织，不光是上手用起来很容易，和其他人进行交流也同样轻松，还可以对程序编写中可能遇到的错误进行预判（prevent from making mistakes）。如果遇到了模型性能不佳，也能很容易的 debug 。

以上是 PyTorch 的几个主要特性，这些特性很好地帮助了用户从科学研究到生产部署的整个过程。一般地，一个 AI 应用的产生一般经过三个流程，分别是:

- 快速实验（Rapid experimentation）。在这个阶段，研究者会产生一些没有过的创新想法，或者想去尝试一些已有的新改进，这时你更多的是和代码“玩耍”，即通过不断“思考——实现”进行迭代，最终 idea 变得清晰并进入下一个阶段;
- 大规模训练（Training at scale）。第二个阶段是在 idea 已经有了清晰的轮廓的基础上，利用更多的数据进行模型的训练，直到模型的准确率或者其他表现能够满足 AI 应用需求后，就会接着考虑进入应用部署阶段;
- 大规模部署（Deployment at scale）。此时模型已经进行了大规模数据的训练，开始部署在真正的业务场景当中，这个阶段会产生一定的业务需求的反馈给研究者，来达到整个应用迭代改进的目的。

当然，时间线并不是在这三个流程中以严格单一的方向流动，而是不断的过渡和迭代、来回切换。比如已经训练好模型并部署上线后，也会根据业务需求或者新的想法来重复整个流程，以达到新一轮的模型迭代，PyTorch 会在整个流程中帮助研究者和开发者顺利的过渡，接下来我们分别看一下这三个流程。

二、快速实验

一般来讲，一个 AI 应用的第一个流程是通过快速实验来得到一个基本可行的方案。实验流程采用 PyTorch 实现的话，又具有以下几个特点:

- 使用灵活、表达清晰 (Flexible and Expressive)。用户只需要专注于实现自己的 idea，不需要去关注框架的其他底层东西；
- 整洁易理解的 API (Clean and intuitive APIs)。这一部分同样是让用户能专注于写代码，可以轻松的通过调用 API 来表达自己的 idea；
- 可组合重用性 (Composability)。PyTorch 中的很多“块 (block)”都是可复用的，在设计自己的模型时候往往继承 nn.Model 这一特性，而设计好的模型本身又是一个 model，这种模块之间的复合非常的容易，也可以很方便和其他人共享模块；
- 丰富的工具库和生态 (Rich Ecosystem of Tools and Libraries)。PyTorch 具有非常丰富的预定义好的模块作为工具箱供开发者调用，同时 PyTorch 也提供了很好的开发者交流的生态环境。

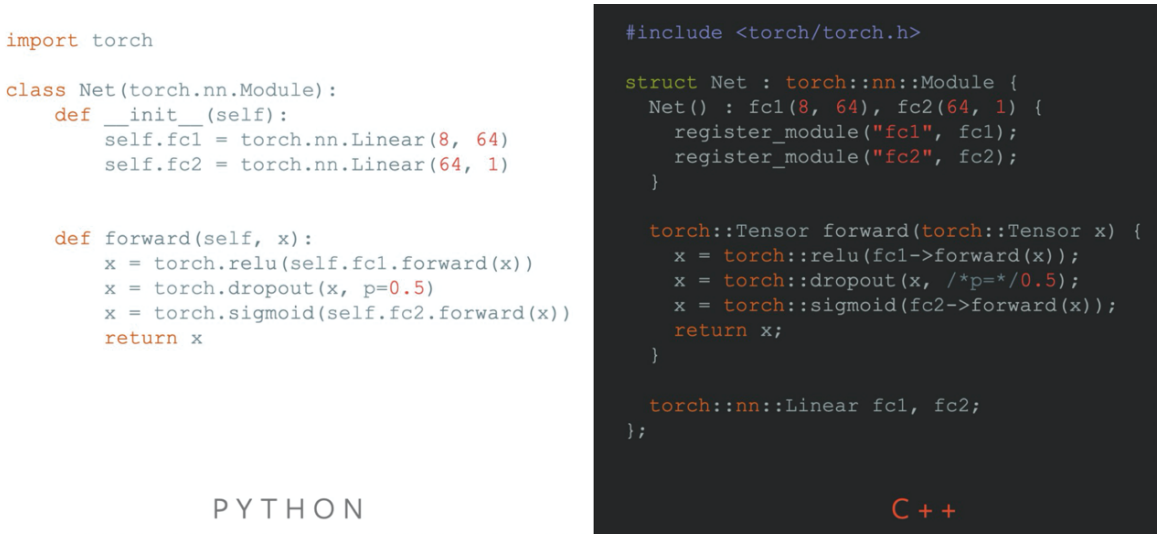


图 2：PyTorch 模型定义的 python 和 C++ 实现

接下来我们来看一个简单的例子，通过 PyTorch 来定义一个简单的神经网络。图 2 是定义模型的代码，可以看到模型本身是一个 nn.Model，其实这一模型本身也是复用了 PyTorch 预定义好的原生 nn.Model，具体例子中的前向传播过程是一个全连接的 nn.Model。也可以在图中看到前向传播过程中，还有 relu 激活、dropout 等过程，这些都是通过 PyTorch 自带的 API 实现的，具有非常的便捷、清晰和高复用性的特点。

定义好的模型，同样可以作为小的模块来被其他更复杂的模型复用。从 PyTorch 的名字也可以看出，最开始实际上提供的是 Python 的 API，但是后来由于用户的反馈和部署需求，后来也提供了 C++ 的 API，C++ 的 API 和 Python 的 API 是完全可以对应匹配的，所以总的来说两个语言版本的 API 没有额外的知识学习开销。

图 3：PyTorch 训练循环的 python 和 C++ 实现

定义完模型之后，图 3 是一个训练循环的实现，可以看到仅通过简单的 data.DataLoader 就可以把 MNIST 数据集导入，然后是优化器和训练循环。下面两行是 Python 语句，用户可以在任何位置插入 Python 语句，具有非常灵活的表达能力。图 3 右边是 C++ 的实现，同样和 Python 的 API 做到了完全的匹配。

刚才提到 PyTorch 能够很容易建立起生态系统，来共享许多可重用的模块和工作。PyTorchHub 就是一个这样的中心化场所，用户可以从中发现和分享可重用模块。如果发布者有工作向分享给大家，只需要在工程下面加入 hubconf.py 配置文件并定义接入点即可。用户的使用更加简单，只需要通过 torch.hub.load 一句话来进行模块的导入即可，也可以传入一些动态参数来调整加载细节。整个生态系统还有许多其他的特性，比如 AX、BoTorch、PyTorch3D、Crypten、FairSeq、HuggingFace、Captum 等等。

三、大规模训练（Training at scale）

经过一些迭代验证，idea 变得可用之后，就需要通过大量数据来训练自己的模型，训练过程也会随着模型变大和数据量变大而耗时增加，我们需要从框架层面给用户一个可以适应大规模训练的能力。

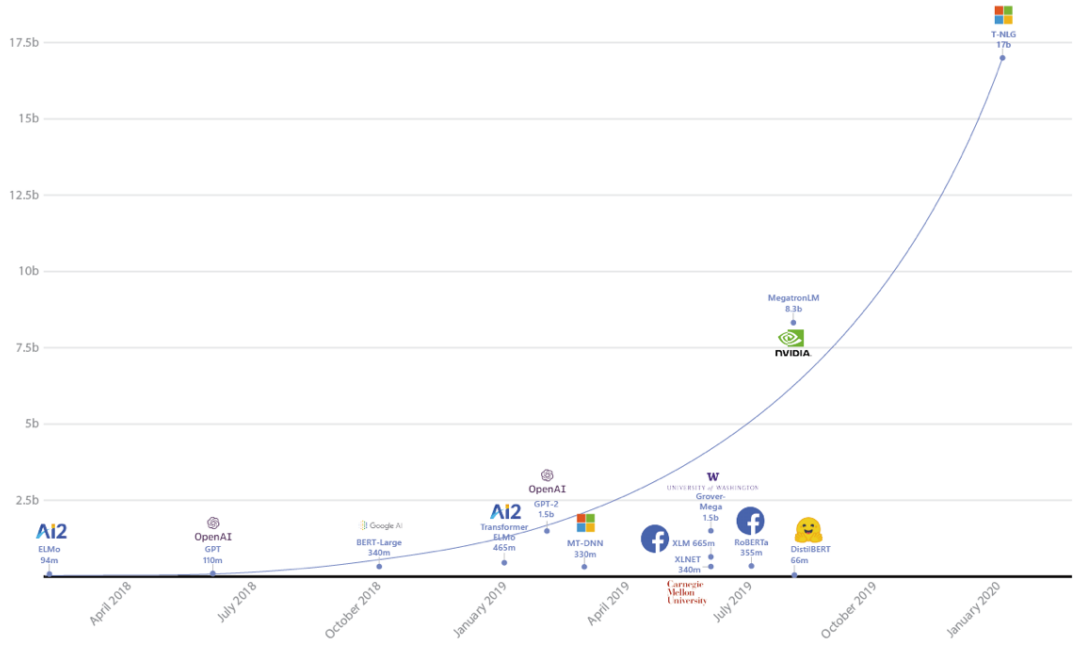


图 4：深度学习模型参数量增长

图 4 可以看出，从业界的趋势来看，模型大小在不断增加，而且是指数型增加。今年年初微软发布了一个参数容量大小 17billion 的模型 Turning NLG，几个月后，OpenAI 就发布了 GPT–3，参数量多了十倍。随着模型大小的增加，计算复杂度也随之增长。

PyTorch 最先支持的是分布式的数据并行，可以在单机上用多个 GPU 以及跨多机进行 GPU 数据的传递，进行联合训练。在 PyTorch 并行运算的背后，其实是综合利用了许多高性能的设计，比如优化网络 IO 跨机器数据传输、反向传递 reduce 等步骤，用起来都非常的简单，只需要把原来的模型通过 DistributedDataParallel 进行并行模型转换即可。易用性仍是遵守模型复用性设计：“模型本身是 nn.Model，经过复用后仍然是 nn.Model ”。代码不需要经过其他任何更改，就可以轻松的分布在多机多 GPU 上进行训练。

刚才提到数据并行的训练方式，随着模型大小的增加模型也不能单独放在一个 GPU 上，这时候就需要进行模型并行。RPC 是在这个需求下面应运而生的一套工具，RPC 是建立在数据并行基础上的一系列工作，例如刚才提到数据并行中的高性能 IO 都可重用在 RPC 上。RPC 定义了两个重要的类型，和一个是 Remote Ref 值代表远程信息的参考值类型，另一个是 features 代表将来可用的值类型。

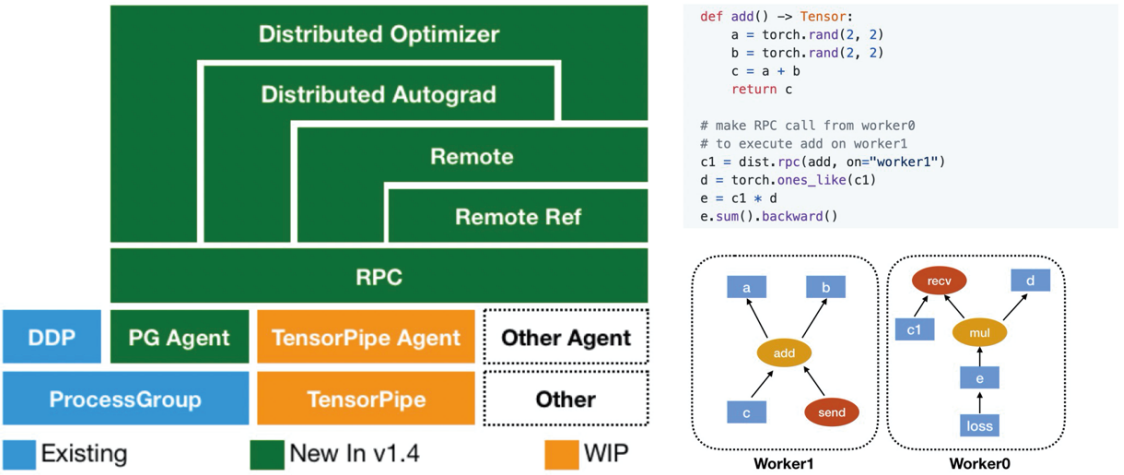


图 5：PyTorch 高性能计算部署

图 5 右边是一段简单的 RPC 代码。可以看出，使用 RPC 的 disk.rpc 命令可以很容易地把一个计算过程远程部署到一个远程工作站，数据经过运算后，把分布式的运算结果进行聚合，然后像本地一样利用 backward 函数传递梯度更新参数。分布式 RPC 在 PyTorch1.4 时候首次发布，而就在 3 个月前也发布了稳定的版本，已经运用在了产品上。

还有 Torch Elastic，这是另一项相关的工作，让用户不用关心网络、机器等底层复杂知识，实现自动地对底层的可用机器设备数量和工作流进行部署。

四、大规模部署（Deployment at scale）

训练完了模型之后需要进行应用部署。在这个阶段要考虑的东西和之前实验及训练阶段是有些不同的。在实验阶段要做的，更多是改一句代码看看 idea 是否如我所愿，但是在部署的时候，已经有一个基本的模型思路了，面临的模型种类和数量少了一个量级，从而更关心系统上的性能、可靠性、特殊硬件适应、自动规范部署等特性。

在这方面，一个很重要的工作就是如何把普通的规范化代码编程图表示的形式，torch.jit 就是这样的工作。

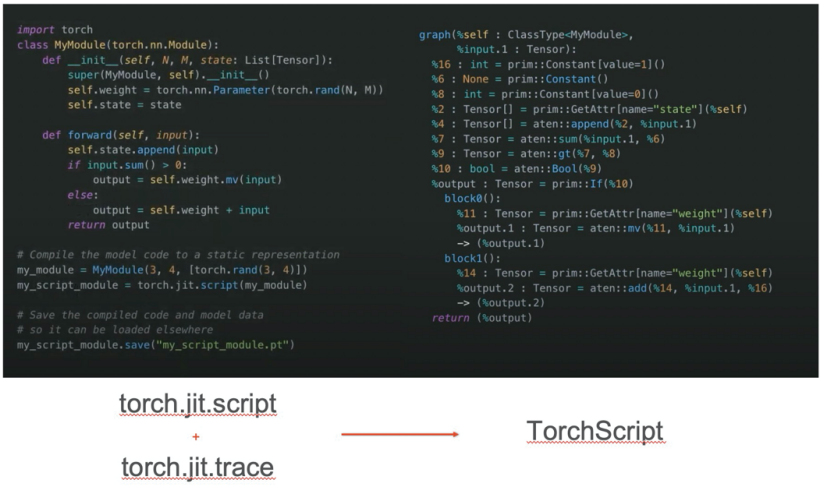


图 6：用于大规模部署的 trace 和 script 两个 API

如图 6 所示，我们有两个 API，分别是 torch.jit.script 和 torch.jit.trace。script 是通过解析 python 的 AST 去静态分析 PyTorch 的 API，得到 PyTorch 的 python 语言图表示。trace 适合另外一种场景，比如 CV 中的模型 RestNet 是一个比较固定的数据形状和模型结构，trace 根据一个基本的简单模型，在一次前向传播的过程中记录下所有运算，进一步生成图表示。trace 和 script 两个 API 得到的结果是可以融合起来的，从经验上来讲我们一般是用两个 API 的结果融合起来的到一个模型，使得既可以用 script 去捕获动态控制流信息，trace 则可以得到一些优化阶段能够利用到的信息。

TorchScript 作为一个图表示，是 PyTorch 作为一个生态系统连接其他高性能库的一个通道，比如现在已经有一些跟 GLOW、ONNX、TVM、TensorRT 的一些结合工作。除了和高性能库结合以外，还跟一些特殊的硬件进行结合，例如 TPU 或者 IPU 等等。

关于简化部署的工作，在轻量的移动端或者服务端显得更加重要，因为这样的设备上运算资源更少，例如内存、显卡的运算能力都会小很多，所以在不损失精度的前提下，把模型压缩，参数减少是一项很重要的工作，PyTorch 的移动端也有一个库可以进行轻量化部署。

刚才说的快速实验、大规模训练、大规模部署几个流程在阿里云的 PAI（Platform of AI）平台上都有很好的接口帮助用户很好地把 PyTorch 用起来。PAI DSW 和 PAI Studio 是帮助用户建模，后者还预设了很多已经验证过的算法； 经过实验验证后的模型，可以使用 PAI DLC 自动化的帮助用户进行大规模的分布式训练； 训练完成后，PAI EAS 有提供部署服务和一些加速技术，帮助用户更高性能的部署自己的模型。

五、结语

白俊杰通过对 PyTorch 的特性介绍，解释了如何帮助用户从科学研究到生产的整个过程进行部署。并根据自身的开发经验，结合具体的三个开发流程，说明了 PyTorch 都有哪些友好的使用方法，部分讲解还结合代码进行了细致的分析。最后通过 PAI 平台的简单介绍，对 PyTorch 前沿的开发方向做了阐述，对 PyTorch 的使用者和开发者有很好的学习和参考价值。

寒武纪副总裁刘道福：智能编程 BANG 语言，释放芯片强大算力！

整理：智源社区 刘鑽

2020 年 6 月 23 日，在第二届北京智源大会 AI 框架专题中，刘道福做了名为《寒武纪智能编程语言—BANG》的主题演讲。

刘道福，寒武纪公司副总裁，2010 年于中国科学技术大学计算机学院获学士学位，2015 年于中国科学院计算技术研究所获博士学位。历任中国科学院计算技术研究所助理研究员、高级工程师和硕士生导师。2016 年 3 月加入寒武纪并任副总裁。刘道福从事人工智能和体系结构交叉领域的研究多年，在人工智能处理器领域有很深的造诣，提出了全球首个通用机器学习处理器架构。

在本次大会上，刘道福分享了寒武纪公司自主研发的 BANG 编程语言及其工具链。BANG 是一种智能编程语言，是整个智能芯片的 SDK 环节里面很重要的一环。它是专门针对寒武纪智能处理器产品架构而设计的，可极大优化寒武纪芯片的通用编程能力并提升用户编程的自由度，方便开发者在寒武纪云、边、端平台上开发和部署。

一、人工智能的发展和应用

刘道福首先简要介绍了互联网人工智能产业的发展和应用。整个人工智能产业非常大，从传统的大数据，到现在的互联网，还有自动驾驶、医疗，各行各业都产生了一些人工智能的应用。

图 1：人工智能的发展与应用

在整个人工智能的应用中，有三个要素，主要包括大数据、计算能力和算法。

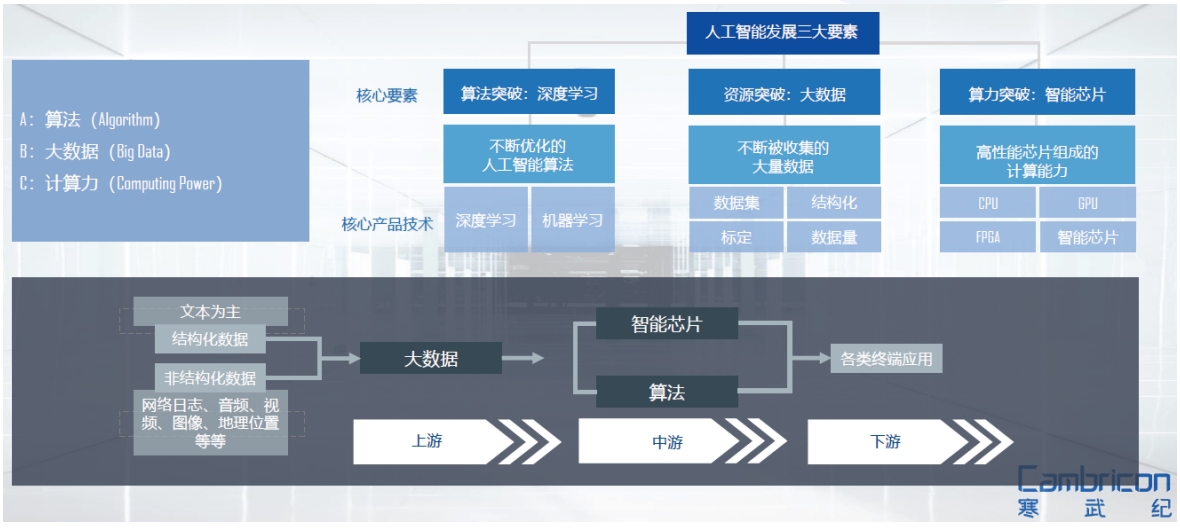


图 2：人工智能三个要素

人工智能的发展有三个浪潮，我们现在处于第三波浪潮。第三波浪潮是通过深度学习或者深度神经网络带起来的。寒武纪是在浪潮比较早的时候就开始做偏底层的研究和智能芯片的研究。最早于 2008 年课题组就开始成立，并且进行了人工智能和处理器架构交叉的研究。

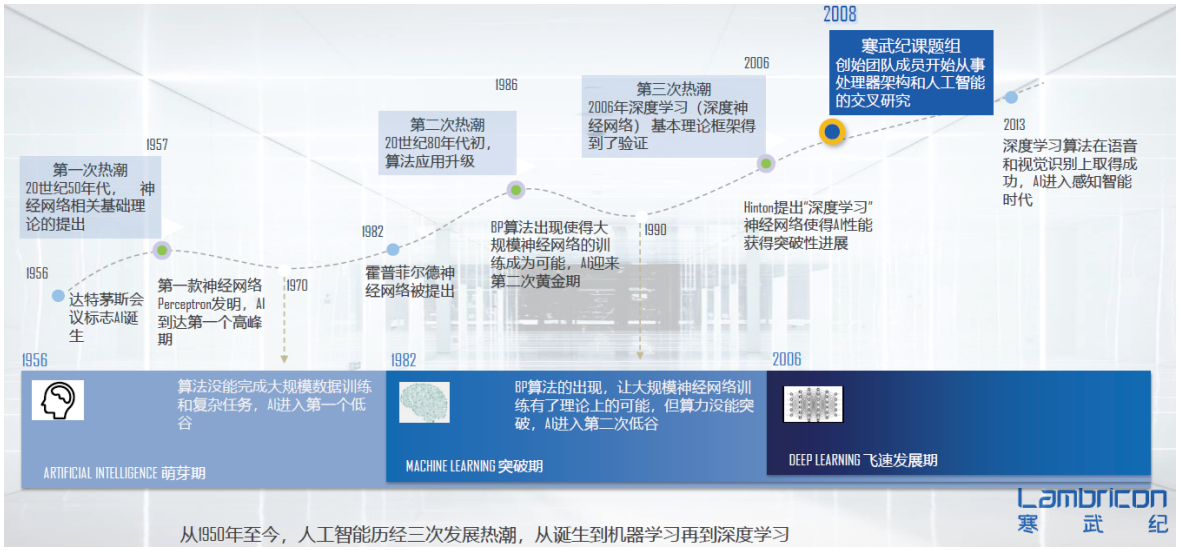


图 3：人工智能三次浪潮

二、传统处理器与智能处理器

刘道福详细介绍了智能处理器与传统处理器的区别。智能处理器是针对机器学习优化的处理器。当前的智能处理器往往是针对深度学习或者网络，从长期来看智能处理器是针对智能应用或是针对机器学习优化的处理器，它与传统处理器的差异，从不同维度来看是不太一样的。

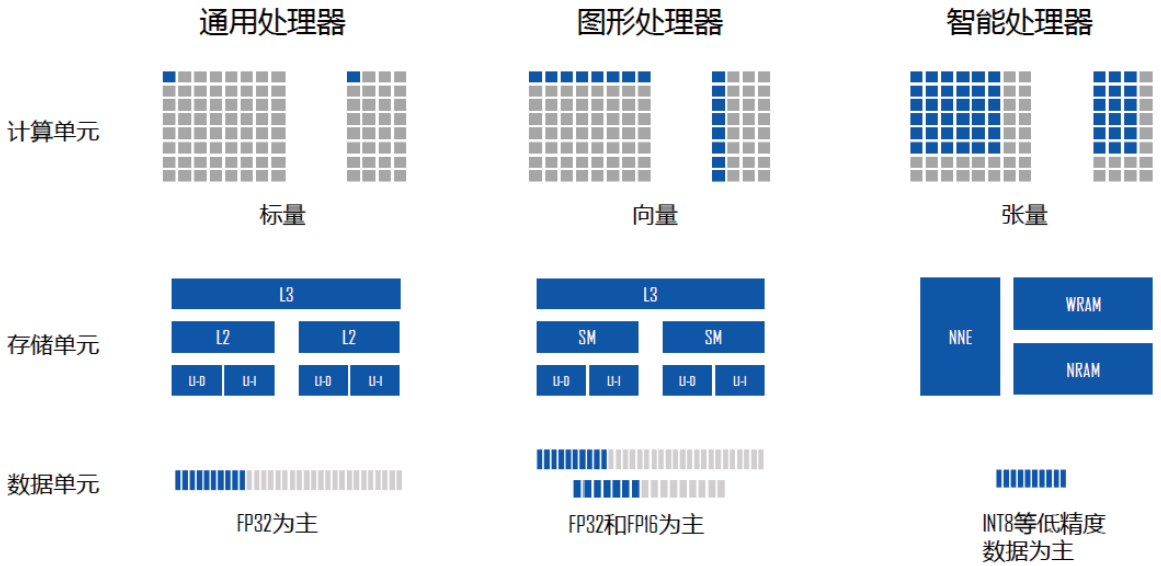


图 4：智能处理器与传统处理器的区别

首先，从计算单元来看。传统通用处理器的计算单元往往都是标量，到后面随着应用的发展也做了一些向量支持，比如 AVX 这样的支持，但是它核心计算能力和控制通路的设计最早都是针对标量的。图形处理器最早是应用于图形处理，一般处理的是向量。智能处理器往往处理的是高维的张量或者矩阵运算，所以它的计算单元计算密度是逐步提升的。图形处理器一条指令能完成一个向量的运算，智能处理器一条指令能够完成一个 tensor 或矩阵的运算。

第二，从存储单元来看。这里的存储单元主要是指片内的存储单元。传统通用处理器一般分成几级 cache，这些 cache 对程序员是透明的，一级 cache 往往分为指令和数据 cache。图形处理器是一个比较模糊的概念，有时候又叫寄存器或者一级 cache，上面会有一些 share memory。有些图形处理器有 L3 的 cache，但是总体来说它的结构对程序员来说有些是透明的，有些是不透明的。现在到了智能处理器，运算器旁边会放一些 SRAM，这些不像 cache 对程序员不透明，程序员是可以直接操作的，通过操作这些 SRAM 优化更好的访存，减少片外带宽的需求。

通用处理器和智能处理器的区别在于，通用处理器即使是 L1 cache 也需要大概 4 拍或者 10 拍的时间去访问，延时还是比较大的。并且每一拍都要经过一级计算器，是需要能耗的。所以即使像 L1 cache 这样的片内存储的能耗和延时都比较大。但是智能处理器的 SRAM 往往紧挨着这个运算器，好处是能耗更低，因为只需要一拍就能够访问，直接从 SRAM 取出来数据马上就进入运算的 pipeline 里面，这样整个能耗会好很多。

第三，从数据方面来看。传统的通用处理器最早的运算主要是以 32 位浮点为主，后面有些场景科学计算有 64 位浮点或者文字处理里面也有一些定点，但总体来说运算器以浮点为主。早期图形处理器其实也是以浮点为主，后面为了 AI 做了半监督的浮点或者一些定点的处理。但是现在智能处理器往往是通过定点的方式来减少能耗或者芯片的面积，以及增加整个芯片的算力。通过低精度方式能让智能处理器的面积降得很小，功耗降得很低，在同样的面积或者同样的功耗下，它的性能提升是一个数量级的提升。

下图包含通用处理器、图像处理器以及智能处理器的具体应用场景、通用性以及代表厂商的信息。总体而言，通用处理器已经非常成熟了，在我们的生活中到处都能见到相关的应用，包括一些设备或者一些场景。图形处理器虽然相对来讲专用一点，但是每个人都可以接触到，如手机、包括 PC 游戏都在应用。智能处理器刚起步不久，现在也已经在一些领域中与我们打交道了，但是这些智能处理的任务，现阶段往往还是在通用处理器和图形处理器上完成的。比如现在的数据报告在整个智能处理服务器里，基于 CPU 的服务器比例有 90%，剩下的 10% 大部分是基于 GPU 的，真正用智能处理器的任务比较少。但在未来当这些智能应用全部切到智能处理器时，它的市场容量会非常大，因为趋势是往智能处理器走的。整个智能处理器针对智能的任务，包括神经网络或者其他的机器学习算法，包括一些 CV 的处理都专门做了架构优化、SDK 优化、软件栈优化，能效相比通用处理器和图形处理器是有优势的。

不同类型处理器比较

	通用处理器	图形处理器	智能处理器
应用	PC, 移动终端，服务器	图形处理和渲染	计算机视觉，语音，NLP，搜索推荐
通用性	通用性强	适用于并行计算	适用于智能处理
代表厂商	Intel, AMD, ARM等	Nvidia, AMD等	传统与新兴半导体厂商

图 5：不同类型处理器比较

三、智能芯片应用场景

刘道福提出，作为芯片厂商，关注的角度**首先是不同应用场景算力的要求**，因为不同算力意味着芯片的尺寸规格会不一样。二是它的数量和容量，就是整个量。从整个市场来说，最大的是未来的物联网的设备，物联网的设备基本可以认为未来所有的物联网设备都具有一定的 AI 能力，因为物联网设备一般都具有传感器。因此，在未来这些智能处理器和传统的 CPU 和 MCU 一样，量是数亿级的。类似现在的 ARM，一年嵌入它 IP 的发货量大概有 60 多亿到 100 多亿，所以未来智能处理器在这会有非常大的量。它相对算力的要求会低很多，往往低于 1 万亿次。

另外一个比较大的应用场景就是以手机为代表的移动消费类电子类的智能处理。当前最主流的是手机，手机有一个特点就是量特别大，因为整个手机一年发货量大概是 10 多亿台。未来可能会有新的消费类设备，包括 AR/VR、智能可穿戴设备以及未来家里电视上的摄像头。原来的体感游戏需要专门的设备，现在电视具有 AI 能力，能处理具有体感游戏所需要的 AI 需求。

再往上是更高算力需求的场景，比如车载大概需要 20–200 TOPS，取决于它的 level，总共有 5 个 level，每往上升一个 level 算力就要翻 5–10 倍。20 TOPS 大概是 level2 自动驾驶所需要的计算能力。自动驾驶这个市场相对来说单设备算力需求比较大，另外它的设备数也非常多，每年全球汽车乘用车发货量有 6000 多万台，所以是非常大的算力市场。

最后一个是现在大家最关注的云端应用场景。目前所有框架都有加速应用，除了像 TensorFlow Lite 这种轻量级的框架，主流的框架加速 AI 处理往往还是偏云端训练的场景。数据中心和云算力往往取决于业务规模，比如互联网厂商业务规模，往往跟整个并发的请求数有关，算力从 POPS 到 EOPS 不等。

图 6：智能芯片广泛的应用场景

智能芯片有这么多应用，但是这些应用往往是千差万别的，不同的应用对程序员来说需要的算子以及硬件架构的抽象都是不一样的。如果要充分发挥智能芯片算力的话对开发者来说有很多挑战，因为对开发者来说把一个智能芯片用好需要一个好的 SDK，但是一个好的 SDK 往往是非常大的工作量和积累。这里面包括几方面，一个是算子的完备度，但即使算子再完备，很多厂商也有自己的一些私有算子，这些算子往往是没有支持的，这时候就存在另外一个问题，这些私有的算子怎么去更好地支持，则在整个环节里需要编程语言去支持。总而言之，**要充分发挥智能芯片的算力是离不开良好的智能编程工具支持的。**

四、传统编程与智能编程

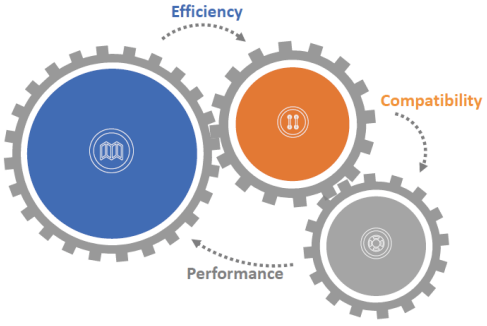
刘道福认为智能编程有三个痛点:

第一个是开发效率，对大部分企业来说，程序员成本往往比硬件成本还高。假如业务规模不够大，花 5 个程序员做半年，可能投入就几百万。如果把 5 个程序员减到 1 个程序员做 1 个月就把这件事情完成，即使性能差一点，就可能多买几十张卡。所以智能编程最大的痛点还是开发效率。开发效率这里面主要考核的点，首先是支持智能计算相关核心算子的数量以及它的友好度，另外是在具体业务上，比如视觉处理或者语言处理，语言有专门的语言处理框架，视觉有专门的视觉处理框架，系统有没有做不同领域的支持。包括自动驾驶也会有类似的框架，需要适配这些应用框架，一方面是开发效率高，一方面在训练时需要适配的是 AI 的框架，但是真正部署的时候需要适配一些领域的框架或者一些场景。

第二个就是高性能，高性能也是毋庸置疑的，客户选用硬件或者编程平台，包括软件硬件平台，往往是追求更高性能和更低成本的。

第三个是兼容性，就是希望现在写的代码是可以移植的，包括它的功能和性能都是能保证的，而不是在不同的硬件上它的功能需要重新去开发。或者换了一个新的硬件，它的性能差异非常大。比如同样是云端的场景，换了一个硬件，它的性能差异很大，这样对业务会有影响。比如对互联网业务往往有延时要求，希望整个业务从

请求到处理完做到 7 毫秒以内或者更低延时，假如性能差异非常大的话，就会出现一定问题。



开发效率
- 支持智能计算核心算子级别的高层语义，如高维矩阵运算的算子
- 支持特定领域智能任务编程，如现在非常热门的语音识别和自动驾驶类

高性能
- 支持底层智能硬件的计算特性
- 支持底层智能硬件的存储特性
- 支持低位宽运算单元，如半浮点和定点

兼容性
- 支持功能可移植性
- 支持性能可移植性
- 支持不同智能硬件的共性特征

传统编程模式和语言无法解决上述痛点

图 7：智能编程的三大痛点

使用传统的编程模式或者编程语言，在人工智能芯片领域是很难直接解决这些问题的。这里面有三个问题：

第一，语义鸿沟。传统编程语言用来描述的是标量的计算和优化，和后面一些向量的计算和优化，但总体来说传统的编译器侧重的是控制流的优化，而不是本身怎么提高并行度的优化。现在智能编程往往是一个 Tensor 或者矩阵级的优化，或者更高维张量的优化，当然有些应用会有向量优化。这意味着设计编译器时更多考虑的不是解决向量级操作的优化或者控制流依赖，而是解决更宏观的矩阵之间的依赖或者一个图的依赖，或者大的 Tensor 之间的依赖，而不是传统的标量或者控制流的依赖。

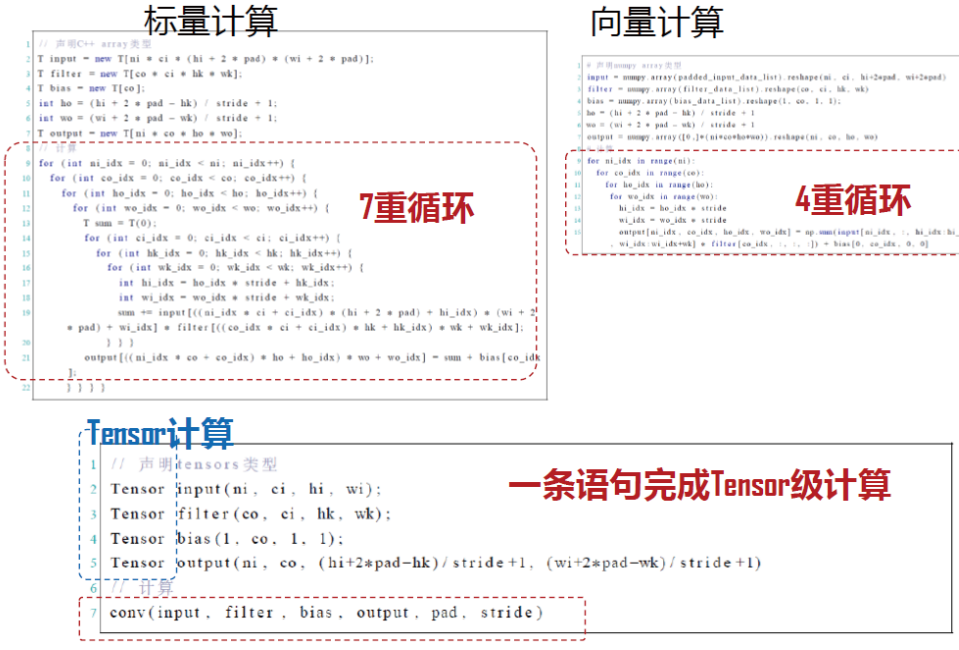


图 8：传统编程模式的问题 – 语义鸿沟

第二，硬件鸿沟。首先，在 cache 或者片内存储方面，传统处理器尤其 CPU 针对程序员是透明的，但智能芯片对程序员是可见的，这意味着传统编译器可能比较少操作 cache，是通过硬件的 cache 替换算法去维护 cache 的存储，包括它里面应该存什么数据、什么时候替换出去。但现在什么时候把数据替换出去什么时候把数据 load 进来都需要程序员或者编译器去做，这对编译器提出比较高的挑战，意味着编译器对未来的数据访存需要有一个比较好的预测。

其次，传统硬件都是高位宽标量，现在智能芯片往往是低位宽张量，所以两个优化方向不太一样。传统考虑是怎么把标量或者控制优化好，现在张量是把图或者张量之间的依赖去解决好，包括访存和计算怎么去并发。当然，这在传统的 CPU 里访存和计算并发也会有一些优化的手段。

最后，传统的硬件往往是乱序执行，所以它需要一些其他的优化手段，比如使用保留栈把整个 pipeline 填满。现在的智能芯片往往更多是顺序执行，顺序执行意味着假如下面 load 出来的指令需要依赖前面的话，原来乱序执行的有些指令可能会先去跑硬件，顺序执行的话硬件就是在干等的，所以需要编译器对未来的数据和计算的依赖分析得更好。当然，智能编程相对来说本身在编译器的友好度方面就做得更好。比如人工智能的应用里面，它的访存往往是连续的、可预测的，比如输入神经元往往是一块连续的存储，权重也是连续的存储，包括中间结果也是一些连续的存储，这些连续的存储，用户的访存行为往往是可预测的，包括未来用到哪些数据，也是通过调度能提前预测的，所以这方面可以通过一些预测的方法去做更好的执行和优化。

图 9：传统编程模式的问题 – 硬件鸿沟

第三，平台鸿沟。传统的基于加速器的编程往往是只做一些算子。做算子有几个问题：第一个，算子一般只是在某个硬件平台或者某个加速器上可用，换一个硬件平台它就没法用。另外一个问题是，算子通常跟不上算法发展，或者很多用户希望保护它的核心资产，算法或者算子是不希望暴露的，不希望厂商或者芯片厂商把算子做了，希望自己有能力去做算子，这就依赖编程语言去解决这个需求。所以智能编程语言需要提供良好的可移植性，包括功能和性能的可移植性。**理想的编程语言需要抽取不同硬件平台的共同特征，并在此基础上提取性能关键特征作为语言特性提供给用户**。

为什么需要智能编程语言？刘道福针对这个问题的回答就是，为了解决以上传统编程模式的三大问题：语义鸿沟导致的开发效率低，硬件鸿沟导致的执行率低，平台鸿沟导致的可移植性差。

刘道福认为在智能编程语言方面需要做的一些具体工作，包括在设计的时候需要考虑的，包括一些常量、宏、内置变量、IO 操作语句、标量、向量计算语句、张量、计算语句、语法控制和数据类型，它和传统编译器总的来说是类似的，但是重点会放在张量计算、张量依赖，以及访存和计算的依赖上。

图 10：智能编程语言

另外，智能编程语言模型有以下特点：一是异构编程，一般是 host 执行基本的控制流程，device 执行相应的核心计算。二是它有一堆内核函数。三是编译器支持需要做任务划分、数据通信、同步以及内建运算支持。四是它的调度策略相比传统的处理器采用更粗粒度的策略，并且通过队列方式来管理执行流。

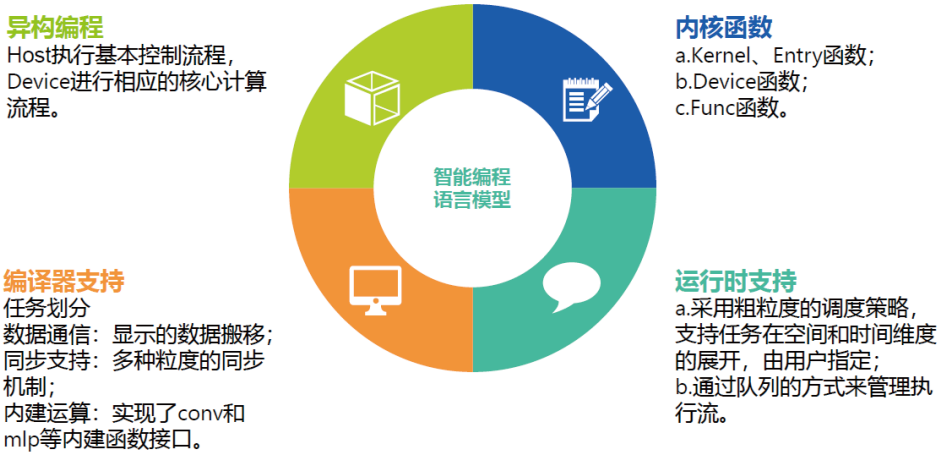


图 11：智能编程语言模型

随后刘道福对整个智能编程语言的框架做了一个总结：从硬件的抽象架构映射到一个编程模型，然后通过编程语言去描述这个编程模型，以及提供给应用一个编程接口，最后这些编程语言以及一些其他的算子会集成到编程的框架里面。**通过智能编程语言及其框架，提供一种高开发效率、高性能和高可移植的编程方法。**

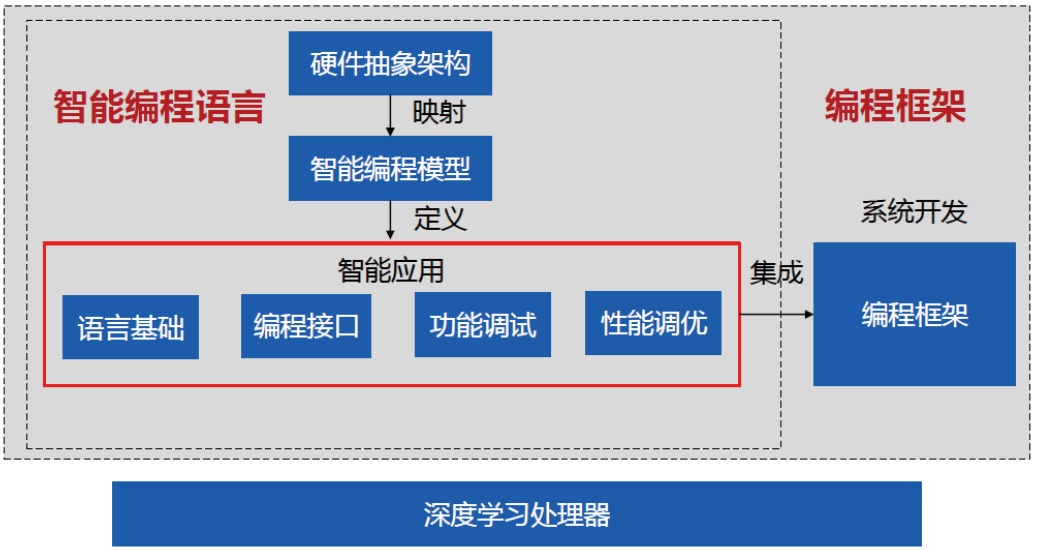


图 12：智能编程语言框架

五、寒武纪的智能编程 BANG 语言

最后，刘道福对寒武纪的智能编程语言 BANG 进行了介绍。这是一个针对寒武纪自己资源系列的芯片和加速卡的编程语言。BANG 的含义是寒武纪大爆炸，旨在希望这个编程语言使用起来更加方便。它主要的目标是希望做到高效的开发效率和超强的执行性能，以及业务程序的通用性和用户编程的灵活性。

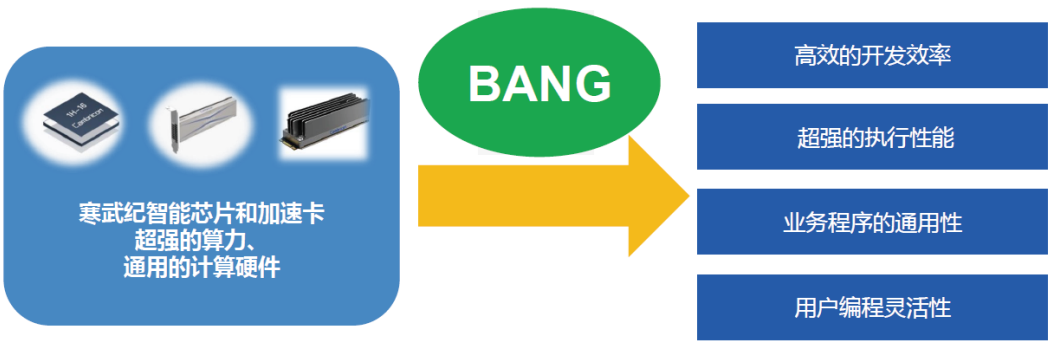


图 13：专为寒武纪智能处理器而设计的编程语言 –BANG

这个编程语言覆盖了云端芯片、边缘芯片以及端侧的 IP，所以它在寒武纪本身的芯片上可移植性是非常好的。未来会逐渐把编程语言往外开放出去，让更多硬件也有机会去使用此编程语言。

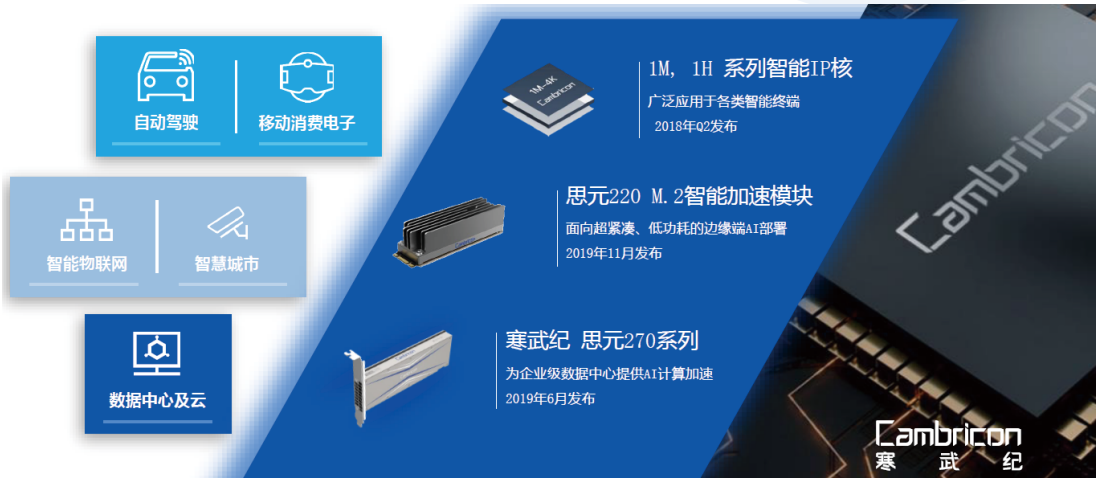


图 14：BANG 语言“云、边、端”全覆盖

这个编程语言对于寒武纪的最大价值，是可以让编程更加灵活。传统的基于算子的编程，假如客户有些算子 SDK 不支持，就需要额外的开发。编程模式包括以下几种方式： 第一是直接调用高性能算子去运行一些离线库； 第二个是过高性能的 CNML 机器学习库实现编程； 第三个是用户最容易接受的，即直接通过框架编程，用户可以直接通过 CNML 以及 BANG 语言进行混合编程，提高编程的通用性。BANG 语言的特点是对硬件做了一个比较好的抽象，所以它能够很好地把硬件能力发挥出来。

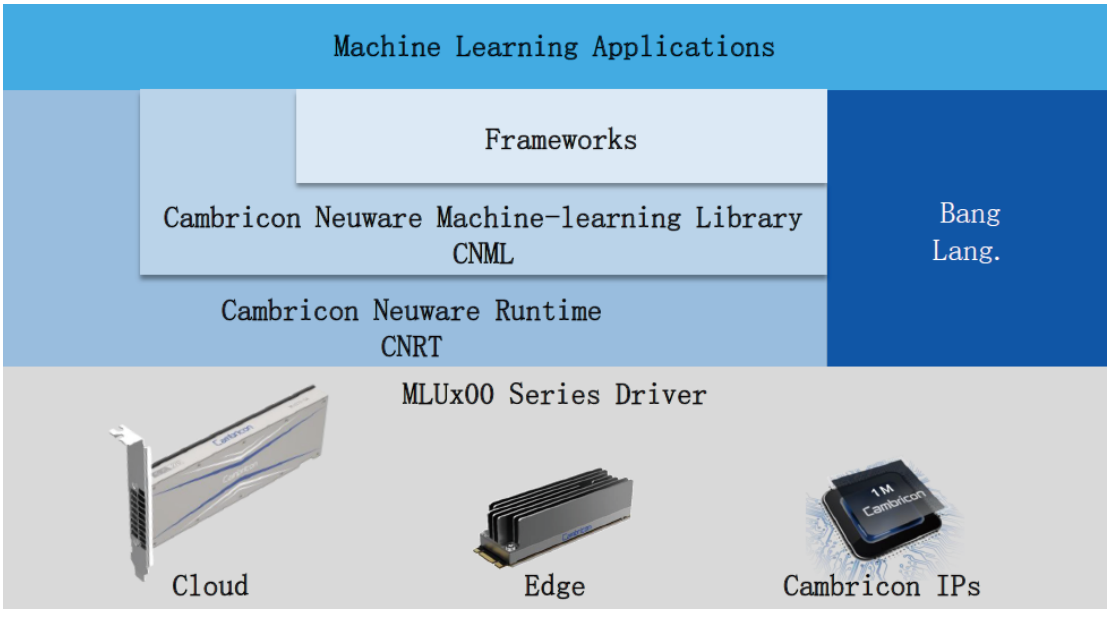


图 15：BANG 语言及其工具链

下图是 BANG 语言的一些具体特性。它支持了多种内置的数据类型，有自定义的算子，以及提供了一个统一高效的编程接口。它提供了一个异构编程的模型，以及支持单核多核的编程。它还支持和高性能库进行混合编程。因为在很多应用当中，大家希望能够将现有的高性能库利用起来。

图 16：BANG 语言特性

下面是刘道福团队做的一个统计，相比直接用高性能库或者直接手写高性能库，使用 BANG 语言开发时间只需要原来的 10%，但它整个性能相比手写算子或者汇编级的算子能够达到 85% 以上。

10%开发周期，85%极致性能

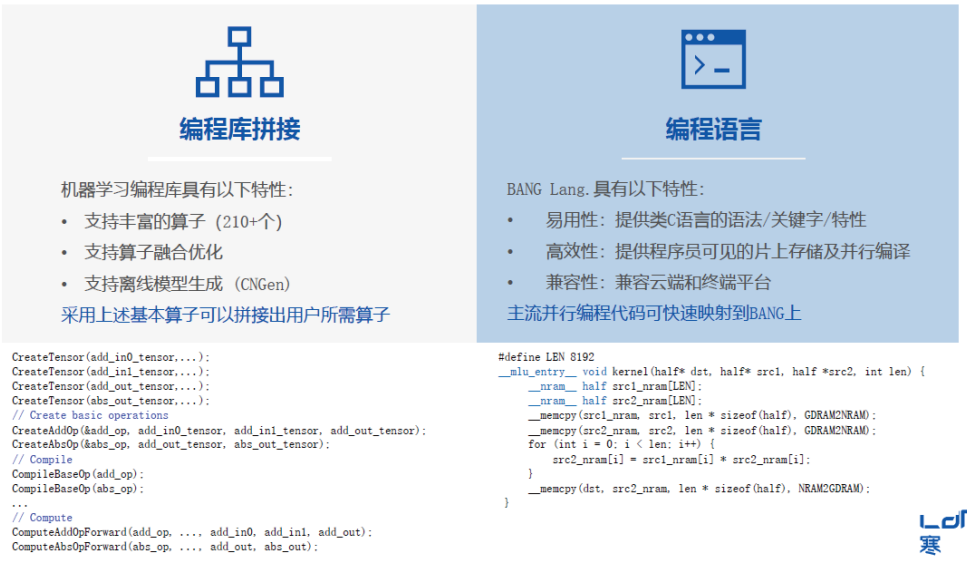
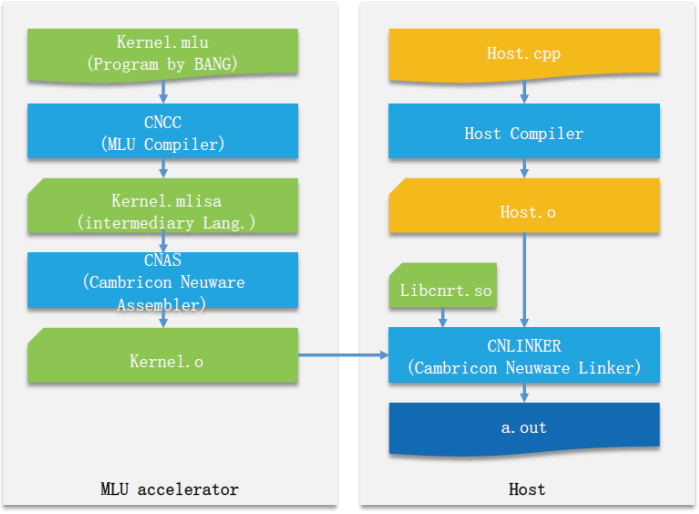


图 17：BANG 高性能库提高研发效率

另外，刘道福的团队对整个编译的流程也进行了优化。自动软件流水，全新的寄存器分配算法，全局指令调度，以及精确的程序依赖分析、配置文件引导优化、地址指针推理及优化，这些常用的编译优化技术也用到 BANG 语言的编译器里面。



编译器优化，提升BANG性能

- 自动软件流水
- 全新的寄存器分配算法
- 全局指令调度
- 精确的程序依赖分析
- 配置文件引导的优化
- 地址指针推理及优化
- 数据类型（如半精度浮点数）计算优化

图 18：compiler 工具链编译流程和优化

最后，刘道福谈到寒武纪开发设计了自己的调试器 CN-GDB，它支持 BANG 语言的调试，并且支持单核多核的调试。实际应用的程序有部分跑在 CPU，有部分跑在加速器上，调试器支持 CPU 和加速卡调试的透明切换。另外，CN-GDB 也支持和 CNML 混合调试。最后调试器是基于 GDB 的，会兼容 GDB 的命令。

图 19：BANG 语言调试器 CN-GDB

六、结语

刘道福的这场报告，向我们展示了智能处理器的优势，以及专门针对寒武纪智能处理器产品架构而设计的 BANG 语言及其工具链。它的主要特点有： 支持多种内置数据类型、自定义算子、统一高效编程接口、提供异构编程模型、支持单核多核编程、高性能库 (CNML) 混合编程。用户不仅可以直接使用 BANG 语言编写 AI 程序，而且可以使用 BANG 语言和寒武纪高性能库进行混合编程，最大限度地释放芯片的强大算力。它的目标是

提供高效的开发效率、超强的执行性能，以及业务程序的通用性和用户编程的灵活性。

附一：问答环节

主持人（陈文光）：BANG 语言抽象出来的硬件的线程模型和存储层次是怎样的?

刘道福：BANG 抽象出来的存储模型有这么几块存储，一块用来存储神经元的，这块程序员和编译器都可以见到，另外存储的是 weights，这一块 RAM 大家也是可以看到的。在我们的硬件抽象里面 weights 的 RAM 往往 add 的是一些矩阵的运算，神经元的 RAM 一般会把神经元的值广播到一个神经网络处理的 engine 里面去做。

整个线程模型做得会稍微复杂一点，因为整体线程是一个比较粗粒度的线程，我们的处理器核是一个比较大的处理器核，所以往往一个线程处理的是整个网络。

当然，我们也支持一些多核协作的编程，比如这里面支持两种模式，一种是数据变形，一种是模型变形。数据变形就是不同的核处理不同的数据，模型变形是指不同的核处理同一个网络。同一个网络不同的层可能划分到不同的处理器核上去要运行。

主持人（陈文光）：听上去不完全是 cuda，因为 BANG 的内存抽象分了 weight 和神经元。

刘道福：稍微不一样。

主持人（陈文光）：至少名字上不一样，实现起来应该还会有不一样的地方，就是你怎么摆放这个数据，怎么去做数据通路到处理器上，然后到并行的执行。

刘道福：比如 GPU，我理解它里面很多东西，比如有些是分 global 的、share memory 的，我们层次是不一样的，我们的层次有些有接近的，最后一层现在在多核里面，为了多核之间的共享数据也有一些需要 memory，这和 GPU 的需要 memory 又比较接近，但是处理器内部结构还是不一样的，存储层次差异比较大。

另外，GPU 现在有个问题，就是它原来的 cuda 是为了给原来的科学计算做的，所以它相对来说更多是效量的加速，所以 cuda 编程模型这一套是针对向量的，对新的 Tensor code 支持还是比较有限的。我们希望 BANG 编程语言一开始就能支持 Tensor code 或者 NPU 等这样加速器的编程。cuda 原来是面向科学计算的，它更多还是向量的，比较少直接处理矩阵这样的维度。

附二：更多关于 BANG 语言的参考资料

寒武纪推出 BANG 语言，高效编程模式释放智能芯片强劲性能
http://www.cambricon.com/index.php?m=content&c=index&a=show&catid=127&id=29
智能计算系统 AICS 视频课程
https://www.bilibili.com/video/BV1WE411A7tv/?spm_id_from=333.788.videocard.1

旷视研究院田忠博：国产深度学习框架的技术探索——天元设计之道

整理：旷视科技

深度学习框架是现代人工智能算法模型应用开发的基本支撑框架。目前，国外主流深度学习框架基于开源开放的生态环境，构建起了较为完整的产业链和庞大的用户群体。要推动中国人工智能更好的发展，国产深度学习框架也必须要在前沿技术的探索中坚持前行。并且要基于产业生态和用户群体的扩展，持续加大资源的投入力度。

2020 年 3 月，旷视开源了在内部工程实践 6 年的天元深度学习框架，Alpha 版包含 35 万行代码，主打“动静合一”、“训推一体”、“灵活高效”、“兼容并包”的四大特性。在第二届北京智源大会上，天元深度学习框架正式发布其 Beta 版，更新至 47 万行代码。针对天元深度学习框架的四大特性，有了持续的强化升级，针对性能、易用性、兼容性以及算法模型覆盖度上，也做了大量优化和改进。

在北京智源大会的“AI 框架”专题论坛上，旷视研究院高级技术总监，天元框架负责人田忠博分享了深度学习框架的核心设计理念，以及在旷视内部，工程化实践 6 年之久的天元框架，在开源之后，从技术社区层面获得了丰富的技术反馈，带来了对深度学习框架未来发展的深入思考。

以下为田忠博分享的精彩内容实录:

一、天元深度学习框架总览

深度学习或者说 AI 是由三个最主要的部分组成：数据、算法、算力，这三方面共同支撑了深度学习和 AI 的应用，也就是人们常说的人工智能三要素。旷视有一套人工智能生产力平台 Brain++，该“三位一体”体系，包含三个系统 MegData、MegEngine、MegCompute，能够极致地将数据、算法、算力融合起来，实现 AI 能力的快速迭代，构筑强大的 AI 护城河。

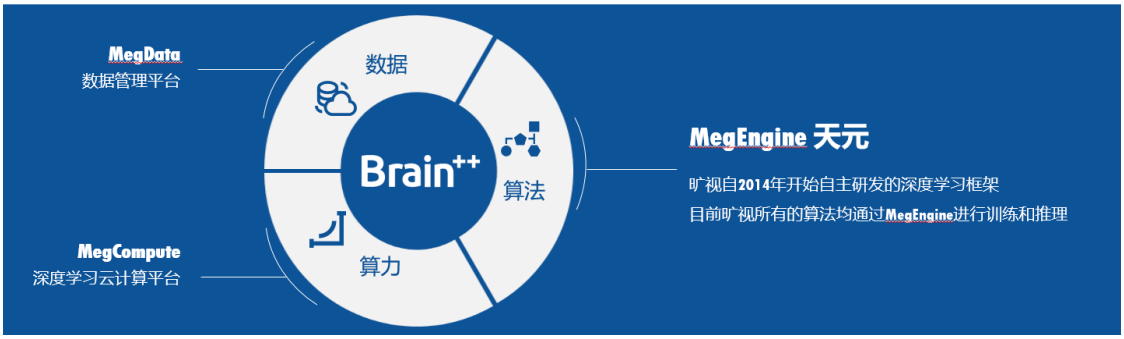


图 1：旷视 Brain++“三位一体”

数据存储与计算平台搭建是 AI 应用的硬件成本中心。在数据存储方面，MegData 通过数据加速能力，使得企业仅用约 10% 的全闪存储集群，搭配低价 HDFS 存储即可达到 100% 全闪存储集群的储存性能。在算力分配方面，MegCompute 通过提供多种训练工具及算力分配优先级机制，得以动态分配所有 GPU，将通常 20%GPU 闲置率大幅缩减至 4% 以内。

而今天主要介绍的 MegEngine 天元，是旷视 2014 年开始自主研发的深度学习框架，这个框架在今年 3 月份正式对外开源。旷视内部所有的算法、模型，都是基于 MegEngine 天元深度学习框架进行训练和推理的。包括旷视几次参加国际比赛冠军的模型，都是在这个系统上进行研发的。

天元凭借其训练推理一体、动静合一、兼容并包、灵活高效的强大性能，将模型开发流程缩短至训练、保存、推理载入执行三步，从而避免了训练、推理模型配置不一致，支持算子不同等问题，有效地保证了能训练就能推理，避免返工，平均节省产品从实验室原型到工业部署 90% 的流程，真正实现小时级的转化能力。

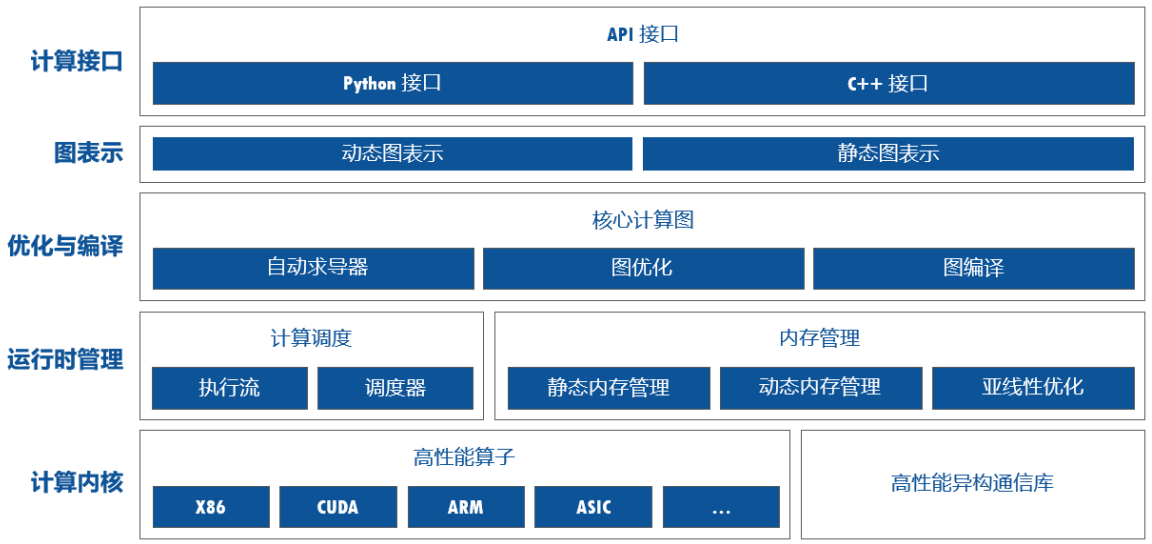


图 2：天元总体架构

上面这张图，就是天元的总体架构，从上到下分为五个层次。最上面 API 接口层，是方便开发者使用这个接口进行编程，这里主要提供了 Python 和 C++ 的接口。接口层下面是图表示层，这里既有动态图的表示方法，也有静态图的表示方法，两者有所不同。

下一层是优化和编译层，核心组件是核心计算图部分。这个部分包含自动求导器自动求导的部分，包含图优化和图编译的机制，使得它能够对上层的动态图和静态图都有所支撑。

在优化和编译层下面，是对运行时、运行态的管理。这个运行时分计算方面的调度和内存方面的管理，在计算调度方面，通常会把不同的设施抽象成为流，然后会有一个调度器统一调度。在内存部分的设计，我们更倾向于用静态方式做内存管理，同时也支持动态内存管理方案，这是天元特有的亚线性内存优化机制，后面会详细讲到。

最下面是对计算和通信的管理，在计算内核层，有高性能的算子库，支持 X86、CUDA、ARM、ASIC，同时有高性能异构通信库，来支持大规模的分布式学习和分布式训练。

二、天元框架的核心设计理念

1. 动静合一

天元深度学习框架的设计之初，就是要做一个动静合一的计算图。天元计算图是以算子为中心，这个图中展示了天元一个以算子为中心的图具体的样子：

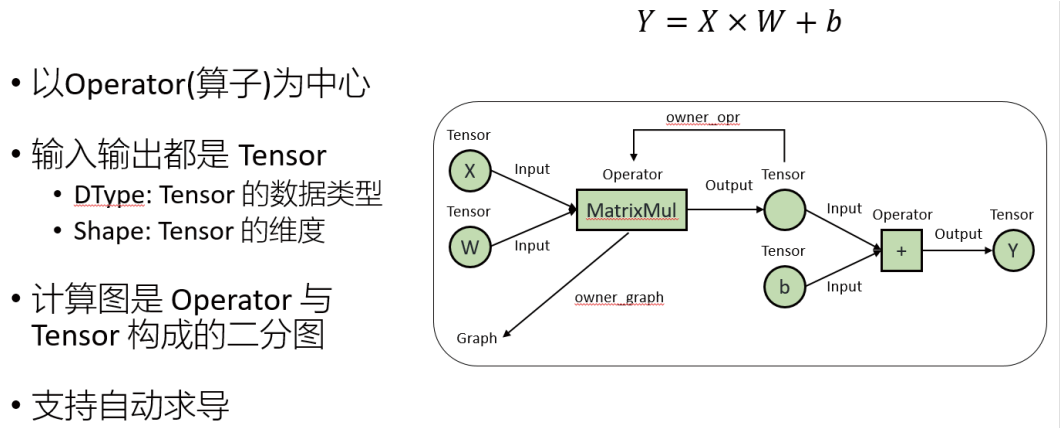


图 3：计算图

这里写的是 X × W+B 的部分，其中圆圈部分都是 Tensor，方框部分是算子，图是以算子为核心进行组织和调度的。每个算子的输入和输出都是 Tensor，Tensor 上有标注数据类型的，也有标注维度的，算子和 Tensor 对计算图构成了二分图。

同时我们用便捷的方法，通过 input、output、owner-graph 等方式，将算子前后串连起来，找到大家所需要的辨别图的方式。基于这个图机制，还可以进行自动地求导，求导机制在后面也会详细介绍。

要想支持动静态图，核心一点在于 Shape，因为动态图和静态图有所不同，动态图可能时时刻刻都知道 Tensor 的值，而静态的图更多时候拿到的是 Tensor 的表示或者代表，所以这里面动静态图的能力上有一定的 Gap。

这里特别提到天元的一个设计，就是 Symbolic Shape。即允许一个 Tensor 的 Shape 是另外一个计算得到的 Symbolic，Shape 也可以是一个 Tensor，天元可以以非常简单的方式来实现 Maxout。这里可以看到一个展开图是这样的，这里有一个对 Shape 计算的部分：

- Shape 也是 Tensor
  - Maxout: `y = x.reshape(x.shape[0], x.shape[1] // k, k).max(axis=2)`

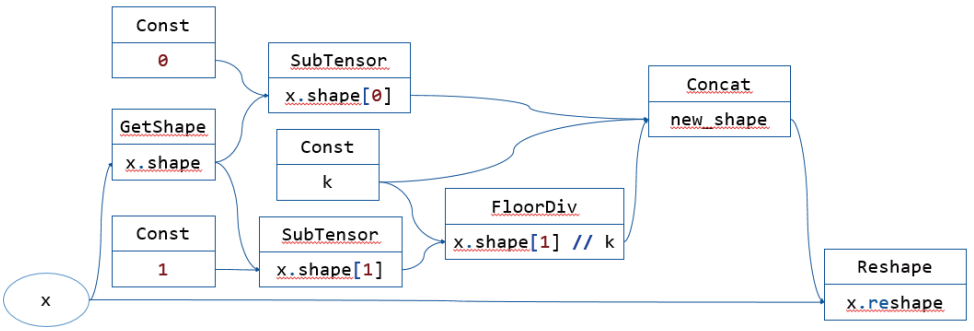


图 4：Symbolic Shape

通过这样的方式，Y 和 Xshape 在这里绑定在一起，我们给定不同 X 的值，可以使任意 Shape 变化，这时候 Y 得到正确的 shape。使得扩展静态图的能力和动态图一样，对于不同输入，动态变化的范式。在这个基础上，真正实现动静合一。

在这里面我们定义动态图是什么？其实是“Define by Run”，就是通过一次执行实际应用计算，这种方式很灵活，便于科研计算。但是静态图也有自己的优势，它虽然是一个 Define and Run，需要先以符号化方式，定义网络执行模式，然后再执行。但是，这样也会使得我们有更多的优化机会，有更多进行编译部署的能力，便于我们把网络更好的放到生产环境下进行推理。

动态图和静态图都有各自的优势，但它们的实现模式又非常不一样。除了刚才提到的 Symbolic Shape 以外，还需要很多方法去弥合动态图与静态图之间的差异。天元深度学习框架通过这样的设计，实现了内置动态和静态的转化。就像下面这张图中的代码，很难说清它是一个静态图还是动态图，只要对这个函数做一个 Trace，就可以控制这个部分是以静态图的方式，或是以动态图的方式来运行。

- 动态图
  - Define by Run
  - 便于科研实验
- 静态图
  - Define and Run
  - 便于生产部署
- 内置动静态转换
- 动静态混合编程
- 探索更多执行模式的可能性

```python
import megengine.functional as F
from megengine.jit import trace

trace.enabled = True

@trace
def train_func(data, label, *, opt, net):
    pred = net(data)
    loss = F.cross_entropy_with_softmax(pred, label)
    opt.backward(loss)
    return pred, loss

for step, (data, label) in enumerate(data_loader)
    pred, loss = train_func(data, label, optimizer, net)
    optimizer.step()
```

图 5：动静合一

此外，天元还可以支持在一个环境内进行动静态的混合编程。天元团队也在寻找更多执行模式的可能性，不仅仅是动态图和静态图，还希望通过多种模式的计算，使得大家找到自己在科研和生产中最好的范式，能够把研究和生产部署无缝对接起来。

2. 灵活高效的优化机制

深度学习框架本身是一个计算系统，计算系统的性能是非常重要的，所以天元希望让用户感受到它非常灵活的同时，也能保持更高的效率。业界早期的深度学习框架设计，是有不同的粗细粒度取向。像 Caffe 会选择非常粗粒度的算子，而 Theano 就会更倾向细粒度的算子。

粗细粒度也是各有优势，比如粗粒度算子，往往表达更简单一些，更易于进行性能的优化。但相对来说会丢失细粒度算子所具有的灵活性，即可以随意用细粒的算子，组装出新的算法。

分类	例子	优势	代表框架
粗粒度	$y = BatchNorm(x)$	简单，易于性能优化	Caffe
细粒度	$y = \frac{x - mean(x)}{std(x)}$	灵活	Theano

- 天元的设计理念:
  - 倾向**灵活性**：需要在框架设计之初就考虑进去
  - **多层次多粒度**的算子，简化计算表达
  - 使用**图优化**技术来持续提升性能

图 6：算子粒度

天元深度学习框架的设计，是倾向于支持更好的灵活性。因为对灵活性的要求，所以粒度的考量，要在框架设计之初就做深入的规划，它影响了整个框架的表达能力和表达方式。天元并不想简单地选择一个极细粒度或一个极粗粒度的方式，而是尝试用多层次多粒度的算子，既可以让用户做细粒度的灵活表达，也提供比较粗粒度的算子，让用户实现自己计算方式的时候更加简单。

- 计算图改写
  - 常量折叠
  - 冗余计算消除
  - 延迟广播
  - ……
- 算子融合
  - Convolution & Bias & Activation
  - Muliply & Add
  - ……
- 计算图切分与转换
  - TensorRT
    - 自动识别可转换子图
    - 自动转换子图到 TensorRT
- 计算图编译
  - CodeGen 实现自动算子生成
    - Halide 支持
    - NVRTC 支持

图 7：计算优化

那么如何协调粗粒度和细粒度算子之间的关系？天元采用图优化技术，进行算子融合来提升性能。这里做了很多计算优化，比如做计算图改写： 常量折叠，冗余计算消除，延迟广播； 做算子融合、也会做计算图切分与转

换、计算图编译，比如自动识别可转换子图，让一部分子图转换到 TensorRT，帮助加速在 GPU 上的部署计算。也会基于 Haidde 支持和 NVRTC，提高它的性能。

除了计算部分，天元还在内存方面进行优化。内存优化的一个难点，是要在计算执行初期，就知道每一个部分到底要占用多少内存。这样要求我们必须静态地推导出每一个 TensorShape 的信息，预先估计好每一个 TensorShape 的大小，通过这样深度的优化内存和显存的占用。

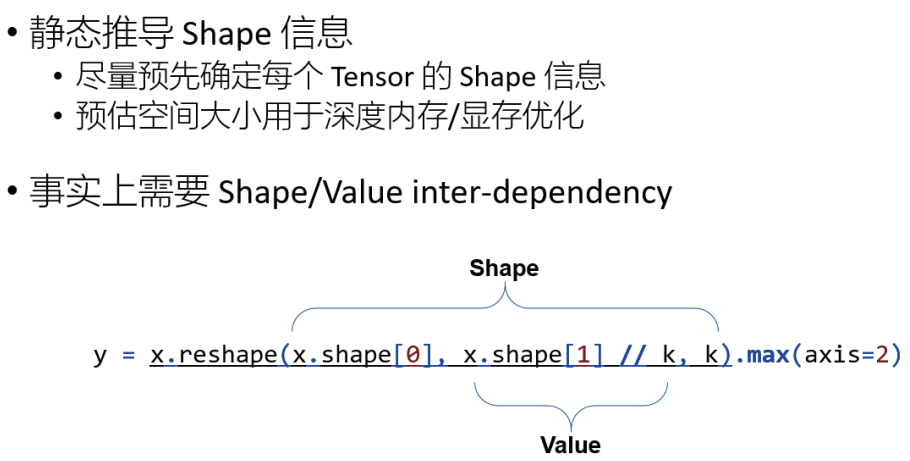


图 8：静态内存优化

天元中的一项功能，是当它需要 Shape/Value 时，一个 Tensor 的值和另外一个 TensorShape 是绑定的。拿刚才举例，中间 Tensor 计算结果影响了最终 Y 的 Shape，这样做静态内存推导的时候，既做 Shape 的也做了 Value 的。这部分内容的详细实现代码，都已经开源了。大家可以在 Github 上下载看到，Shape 和 Value 这部分静态推导的代码。

天元在 GitHub 上的项目地址：https://github.com/MegEngine/MegEngine

当得到预估好的静态内存大小后，可以做 Reuse 和 Forwad。比如下面这张图中，橘色部分可以用原来的内存去计算。如果它只有读，或者它实际上是不断可以本地进行更新，就可以显著减少内存和显存的占用。这一点不仅对于推理来说非常重要，对于训练来说也非常重要，因为可以让我们训练更大的模型。

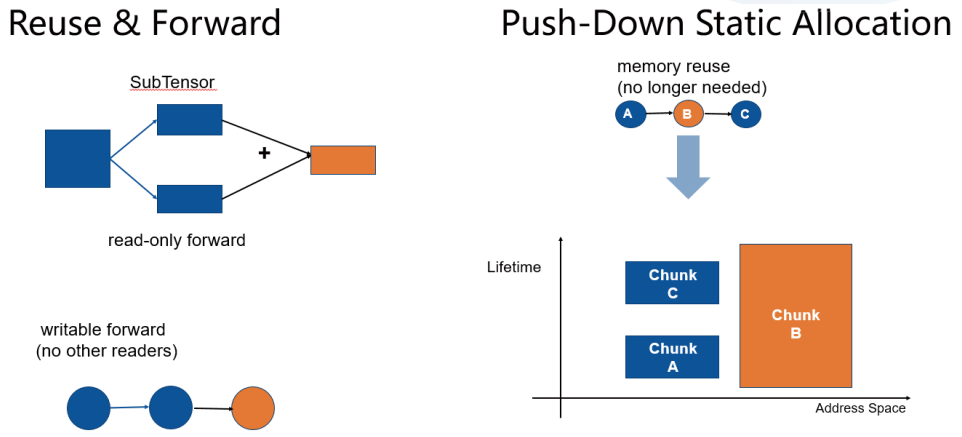


图 9：静态内存优化，Reuse&Forward，以及 Push-down 两种策略

另外，天元还实现了自己研发的 Push-down 内存分配策略，它会根据计算的大小，对内存增块进行排列。通过统筹的 push-down 规划，可以看到上面图中，C 的内存空间其实可以复用 A 的空间，也就是 A 和 C 同时可以复用一块空间，通过这样的方法可以把内存占用深度地优化起来。

3. 亚线性内存优化技术

针对前面提到的，天元独特的亚线性的内存优化技术，它的本质上是用计算来换显存。即在部分地方不会保留所有的中间结果，只保留部分 checkpoint，当这个地方反向要用它的时候再同期进行计算。归纳起来，它整个内存的使用，就是 checkpoint 加上最大的单块 block，这个计算方法比线性要好很多，内存占用可以达到平均根解。

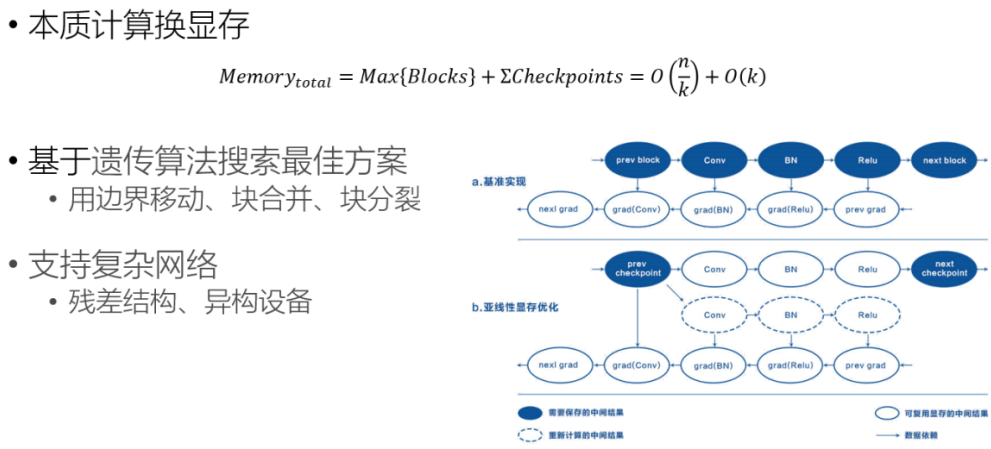


图 10：亚线性内存优化

这是 2016 年陈天奇的一篇论文“Training Deep Nets with Sublinear Memory Cost”中提到的方法，但是原来的论文中有一个问题，它假定整个训练网络必须是一条直链，所以它在真正的网络中几乎不能使用。天元的研发团队认为这项技术非常好，所以做了一个算法，是基于遗传算法去搜索，一旦网络有分岔的时候，用边界移动跨合并或者跨分类的方法，去找那些在这样一个复杂网络中，可以作为 checkpoint 的位置，而且它是内存

优化最佳的方案。

这部分的详细介绍，我们前一段时间在旷视研究院的微信公众号上，有一篇针对这个算法的深入解读，对亚线性内存优化技术感兴趣的用户，可以看我们相关的文章。通过亚线性内存优化技术，天元深度学习框架具备了支持复杂网络、残差结构、异构设备、包括节点不在同一个设备上的方法。

《天元亚线性显存优化技术》https://mp.weixin.qq.com/s/N-bjcUEF4cQbH5vT0RM9CA

4. 训练推理一体化，从研究无缝走向产业

训练推理一体化，是天元核心设计理念之一。训练推理一体化有诸多优势，首先要一张图上表达训练、表达推理，这要求我们在做计算图的时候，不要区分前向还是反向，下图中可以看到，以此为例：

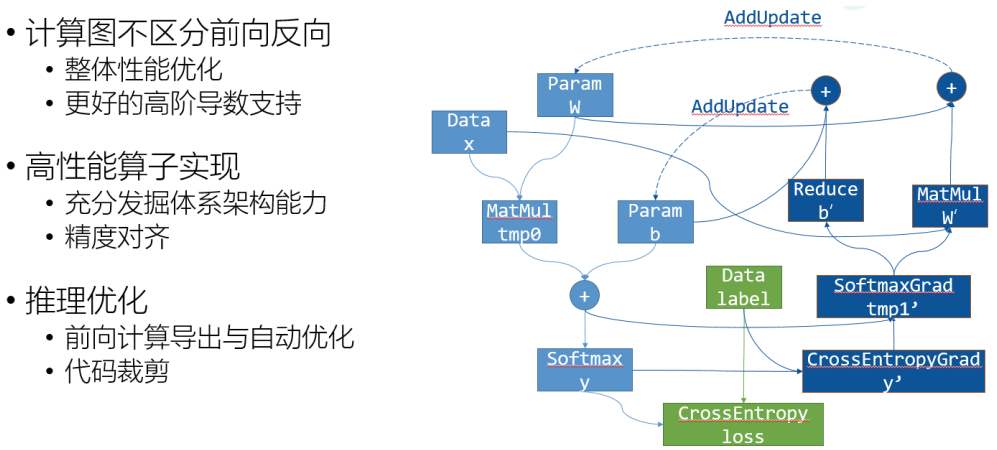


图 11：训推一体

浅蓝色的是正向，中间是橘红色，深蓝是反向，我们可以把它放在一起，这样做的优势是可以对它进行整体的性能优化，可以跨越前向和反向的边界进行算子融合。另外，可以更好的对反向部分继续进行反向，有更好的高阶导数支持能力。

通过高性能算子的实现，充分发掘体系架构的能力。精度对齐，使得我们的研究成果能够精度无损的变成我们的产品。在推理部分也可以通过只保留它的前向部分，而进行代码的裁减，然后进行优化，保持推理部分的能力。

对于量化网络的支持，已经在天元最新的 Beta 版中发布。量化网络需要训练和推理的深度配合，比如 QAT 量化感知训练，是在训练时就知道要进行量化，提前考虑了量化的噪音。再比如后量化，也需要使用原来的训练模型。还有阈值可考虑的量化，量化中的阈值是由训练来定的。

这些例子的目标都是为了进行更好的推理，为推理做的优化，往往需要和训练进行深度整合。因此，训练到推理一体化的方式，就非常适合这些场景。

《天元推理性能极致优化综述》https://mp.weixin.qq.com/s/FKlibmuNrpSnR7tXlXs8Pg

从训推一体化的代码框架中可以看到，在天元深度学习框架中，训练推理非常容易做 training 转化，量化后就能够拿到最好精度的网络，同时性能达到最佳。这个中间过程，不需要再进行复杂的转化和变更就能完成。

5. 支持大规模分布式训练

天元深度学习框架希望更好的支持大规模分布式训练。因此使用的是 MegRay 统一通信库，还有 NCCL、UCX 多种后端实现，支持多种网络协议。像 TCP、ROCE、GPU DIRECT，针对网络设施进行加速，非常简单和直接，也很容易试验和扩展新算法。

在天元框架中，做分布式是希望把这种集合通信，在框架中作为原生的并且可导的算子。这样可以非常方便地进行数据并行、模型并行或混合并行。SYNC BN 或 FC+Softmax 也更容易完成，更易于优化，维护性很好。系统内部集成了像 ParamPack，把小的参数打包做计算。天元框架可以通过这样的技术，很大程度上保持几乎线性。

三、天元深度学习框架的未来开发路线图

2020 年 3 月，天元正式发布了 Alpha 版本，今天在北京智源大会上，正式发布了 Beta 版本 0.5.0，引入全系列的 ARM CPU 的支持，刚才提到的量化和低比特支持也同步放出，希望这个版本对大家有更多的帮助，帮助大家用新的算法在工业界进行算法训练。

图 12：开发路线图

天元在 9 月份计划发布 1.0 的正式版本，能够覆盖主流的计算设备，把框架的动态能力进一步升级，整体优化训练推理的全流程。

天元项目 Github 地址：https://github.com/MegEngine/MegEngine

四、总结 & 答疑

总结报告的内容，天元做深度学习框架的核心设计理念是：

- 具有强大的动静合一的计算图机制，更好的进行训练和推理；
- 灵活高效的机制，能够使它跑得更快；
- 训练推理一体化的方式，简化流程；
- 面向大规模分布式深度学习，这是 AI 和深度学习的未来。

问答环节：

AI 框架专题论坛的最后，主持人清华大学陈文光教授，提出了参会者们最关注的 3 个问题:

Q：天元最开始是在旷视内部使用，后来开源了。现在旷视内部大量自己的应用，也都是用天元来开发吗?

A: 是的。旷视内部所有的算法、模型，都是基于 MegEngine 天元深度学习框架进行训练和推理的。包括旷视几次参加国际比赛冠军的模型，也都是在这个系统上进行研发的。

Q：3 月份天元开源了，提供给外部的用户和自己使用，还是有很大不同，为了提高开发者的使用效率，降低学习新框架的成本，都做了哪些工作?

A: 上面的分享中，展示了一些天元框架代码的样子，大家可以很清楚的看到代码的 Pythonic 风格。总体来说，在最开始的 2013、2014 做的时候，代码不是这样的。为了这次开源，天元的技术团队特意修正和修改了几乎所有的接口，帮助大家更容易上手使用。从 2014 年开始到现在，天元框架更新迭代的变化很大，包括整个行业的审美和使用方式都发生了很多变化，作为框架的开发者，我们还是要保持对最新趋势潮流的敏感。

Q：开源工作需要持续投入大量资源，从旷视的角度讲，做一个自己用的工具可以理解，那开源的目的又是什么?

A: 我们希望做好生态，今天主要分享的天元深度学习框架，是 Brain++ 三位一体中的一个部分。我们希望能够把旷视好的想法和整个体系的技术沉淀，放在产业的角度让大家更多去使用，帮助大家更快获得 AI 的能力。同时，我们也认为，像这样的一个大规模的系统，更需要时间来检验，需要和用户、和生态合作伙伴一起不断迭代，甚至在竞争中来夯实自己的核心价值，这是源于旷视技术信仰中的一个部分，我们相信好的技术一定能够得到大家的认可。

北京一流科技袁进辉：OneFlow——极致高效的深度学习框架！

文稿整理：智源社区 刘鑽

在第二届北京智源大会“AI 框架”专题论坛中，袁进辉博士做了名为《深度学习框架 OneFlow 的探索与实践》的主题演讲。

袁进辉，北京一流科技有限公司创始人，中关村数智产业联盟副理事长，之江实验室天枢开源开放平台架构师。袁进辉是清华大学计算机系博士、博士后，师从中国人工智能泰斗、清华人工智能研究院院长张钹院士，曾获得 2018 年清华大学计算机系“优秀博士学位论文获得者”称号。袁进辉曾担任微软亚洲研究院主管研究员，负责微软下一代深度学习平台研究开发，荣获微软亚洲研究院院长特别奖； 曾发明世界最快的主题模型训练算法 LightLDA； 曾多次在 IEEE TCSVT、ACMMM、CVPR、WWW 等国外核心期刊和顶级会议上发表重要论文，主持及参与过多项国家自然科学基金项目。

在本次大会上袁进辉博士分享了一流科技自主研发的 AI 深度学习框架 OneFlow。目前业界主流的开源深度学习框架有国外 Google 的 TensorFlow、Facebook 的 PyTorch 等，国内也有百度的 PaddlePaddle、天元的 MegEngine 等。作为一个框架研究的后来者，一流科技这样一个创业公司有什么自己的特色之处呢？针对这个问题，袁进辉的分享主要围绕以下两个方面进行： **第一，从一流科技角度来说，框架还有什么新的可探索的技术？第二，作为框架后来者通过什么样的策略有可能脱颖而出？**

一、目前深度学习横向扩展的难题

袁进辉首先从深度学习领域的新算法谈起。他指出，深度学习崛起的最初几年经常在算法上给我们带来很多新鲜的东西，比如像早期的 ResNet 等，算法创新是非常活跃的。但是从 2018 年开始，给人们带来眼前一亮感觉的算法创新变少了。**袁进辉认为目前比较吸引眼球的进步更多的是把原来的算法用在更大规模的数据和模型上，依赖于计算力的提升，也就是人们常说的“ 大力出奇迹”的方式。**

图 1：深度学习的计算力需求

在自然语言处理领域中，2018 年谷歌发布了 BERT 模型，我们本以为它的模型足够大，但是之后又出现了 OpenAI 的 GPT-2、微软的 Turing 模型，特别是最近几个月发布的 GPT-3，它的模型已经达到 1700 亿的参数量。

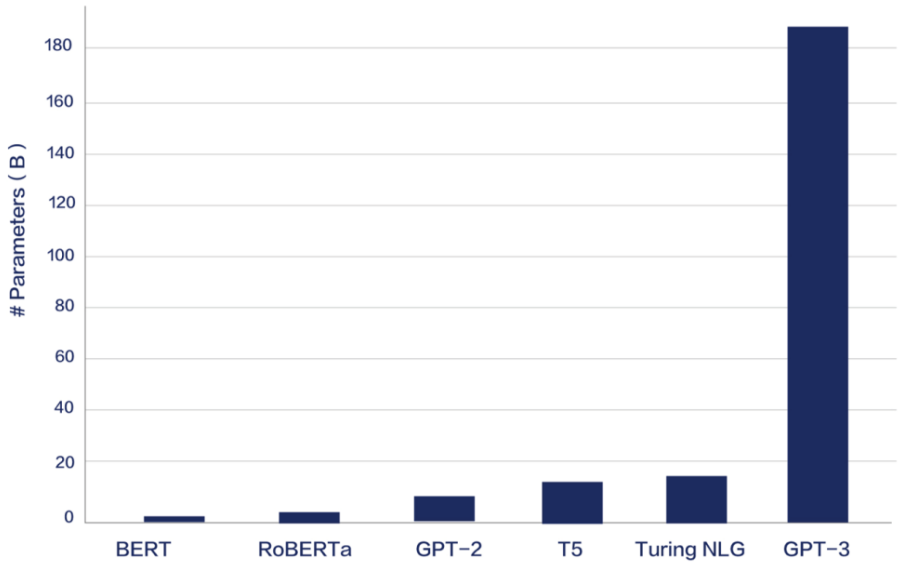


图 2：深度学习大模型的挑战

这个算力需求怎么解决呢？计算量非常大，一种途径是做更专门的芯片，比如从 CPU 到 GPU 过渡，再到各种 AI 专用的 ASIC 芯片。但是芯片技术面临物理限制。单纯靠芯片技术的发展来获得百倍千倍的算力，现在看上去不太可能，所以现实中更多地会把很多芯片通过高速互联的手段做成一个集群，也就是通过横向扩展的方式去获得更大的算力。但是横向扩展也会遇到很像单个芯片内部存在的内存墙问题。现在主流芯片都采用冯诺依曼架构，执行的过程中涉及数据搬运以及数据计算的过程，一般来说，数据搬运通常跟不上数据计算，微观层面叫内存墙。宏观也有这个问题，AI 芯片或者 GPU 本身算的非常快，而且访问自己片上的内存带宽非常高，但是访问其它芯片上的数据或其它机器上的数据时，带宽就比片上内存带宽低一到两个数量级，这通常被称作网络墙。

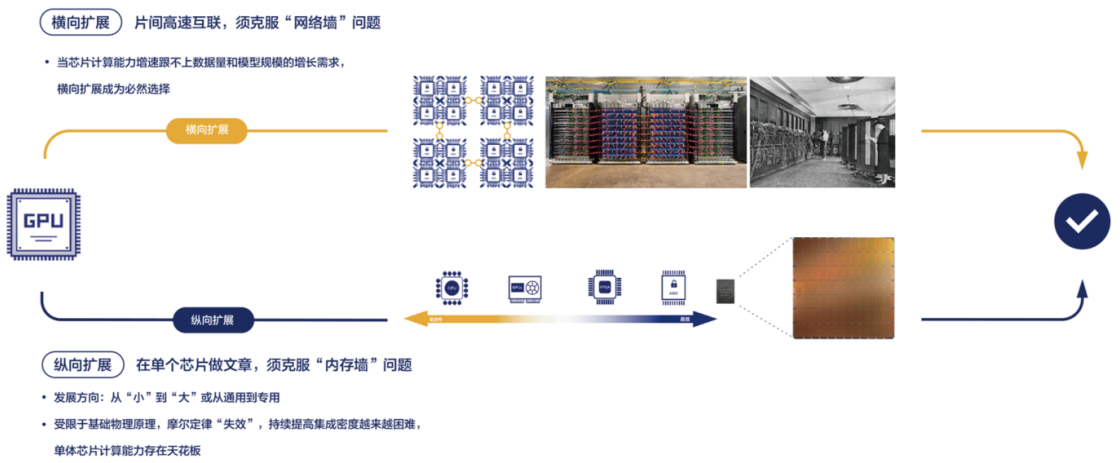


图 3：深度学习的横向扩展难题

很多人以为算力问题在硬件上可以通过集群手段解决，现在很多超算计算机也是这个思路。但是，**袁进辉认为：深度学习计算任务的特点导致即使有这么强的硬件和这么高速的互联带宽，比如 NVLink 等等，如果软件框架本身有不足的话还是会成为瓶颈。**

袁进辉指出通过集群横向扩展到很大规模时会遇到两个难题:

第一是易用性的问题。算法科学家平常写的神经网络是个逻辑图，这个图怎么映射到成百上千 GPU 上去，对不同神经网络结构，最优的并行方式是不一样的。目前已有的框架对数据做切分，也就是数据并行，做得非常漂亮。但是当模型参数量很大时，数据并行非常低，这种情况应该使用模型并行。但现在主流框架不支持模型并行。

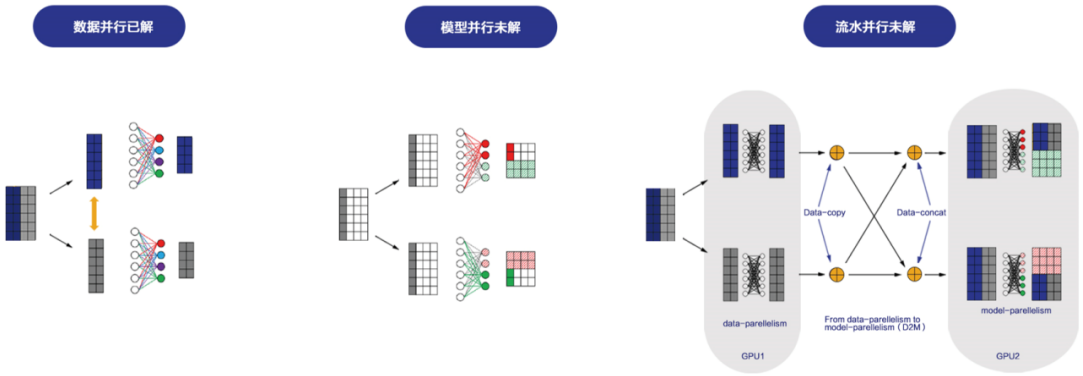


图 4：横向扩展的易用性问题

第二是运行时效率。GPU 本身是异构的计算，它本身处理速度比 CPU 快 10 倍。深度学习中的梯度下降算法是接近流式的计算，计算粒度非常小，所以每个小片段的计算在 GPU 上是几毫秒、十几毫秒、百毫秒的量级。在瞬息万变小粒度计算任务情况下，怎样让每个参与其中的计算设备都不停歇地高效地进行计算，就需要我们把所谓的控制开销，比如资源管理、排队、调度、传输都尽可能降低到极致。这就是深度学习框架在大规模集群上追求速度而面临的一个挑战。

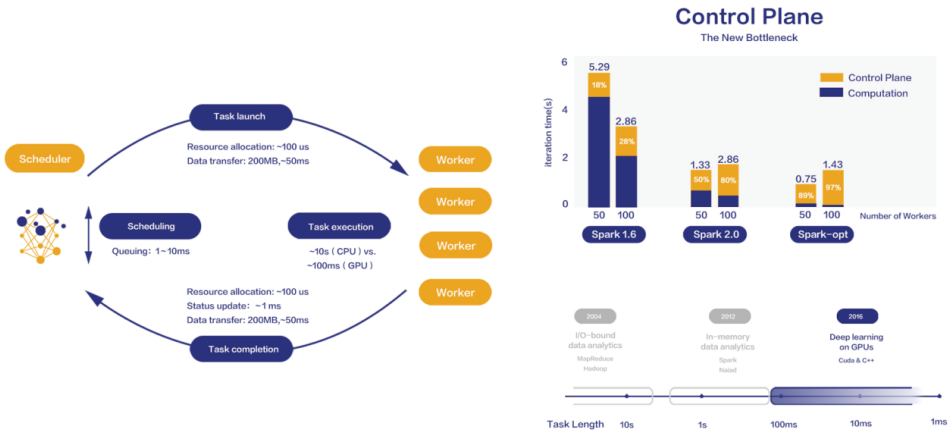


图 5：横向扩展的运行时开销

二、OneFlow 怎样解决横向扩展的难题

接下来袁进辉重点介绍了一流科技的 OneFlow 是从什么角度攻克这些难题的。

2.1 OneFlow 核心设计理念

如图 6 所示，其中有 4 个小图。左上图主要是阐释**通过编译器自动判断和分析一个神经网络最优的并行方式是什么。**如果最优并行方式是对数据做切分，那就用现在很多框架使用的 AllReduce 方式。如果模型参数量很大，那就对参数做切分，做模型并行，对很多神经网络层次进行切分，做复杂的流水管理。

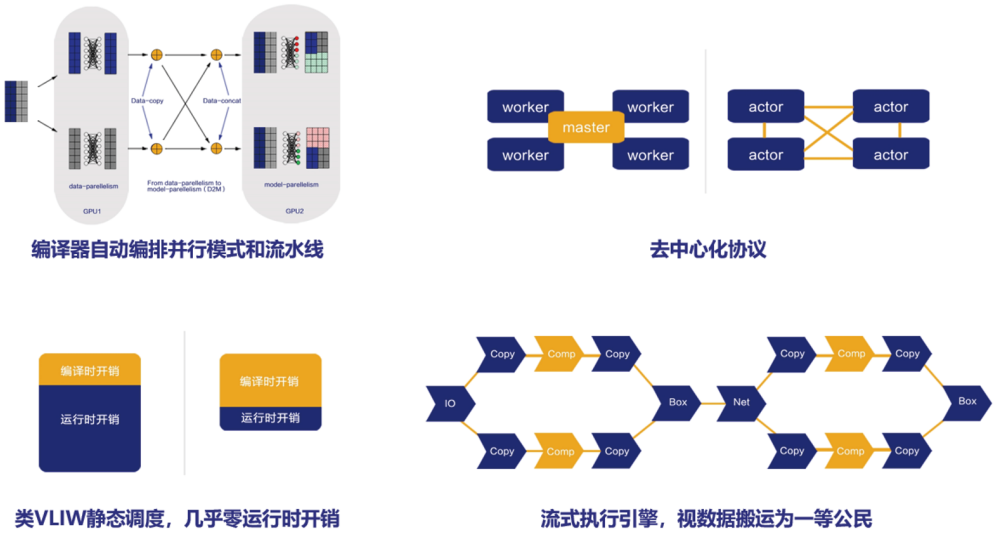


图 6 OneFlow 核心设计理念

左下图指出了深度学习框架里运行时开销是横向扩展时的瓶颈所在，有什么办法可以把运行时开销极致压缩呢？一个方法是可以稍微牺牲一下编译时的开销，把很多调度的管理策略，如内存、带宽、计算的管理策略，**所有能在运行之前得到的信息调度的策略全在编译时做到**。虽然编译时开销长了一点，但运行开销将大大缩减。

右上图是讲深度学习框架**做充分的静态分析之后**，不需要像现在很多大数据系统以及框架执行时有一个中心的 master 命令 worker 去做，master 和 worker 之间有频繁的控制消息的传递，上下游执行体 actor 之间可以直接通信并自驱地判断在什么时机做什么事情，即**做到去中心化调度**。

还有一个特色是因为在比较多的 GPU 参与的异构系统里数据搬运的开销非常显著，不管是跨机器还是机器内部搬运的开销都非常显著，这时候需要在调度时预先发现，先把数据搬运和计算作为同等重要的角色来做编排和调度。而在已有框架处理计算图时只看到计算而没有看到搬运，搬运是在运行时碰运气，能掩盖住就掩盖住，掩盖不住就接受了，充满随机性，这是不可接受的。所以 **OneFlow 在做图分析的时候是把数据搬运视为一等公民来做的**。

2.2 OneFlow 动态图机制

那么动态图怎样做到静态图效率几乎一样呢？问题在于动态图和静态图分析时，编译器看到的信息是不一样的，静态图看到的是全局信息，动态图只能看到眼前一小块、当前一个小小片段的上下文，所以它能做的分析有限。

对此袁进辉打了一个形象的比方，他说，静态图就像已经铺好路了，形状和数据推导就是这个道路，只管开车就行了。动态图是一边铺路一边开车，怎样让铺路不限制开车的速度就是里面的一个关键问题。

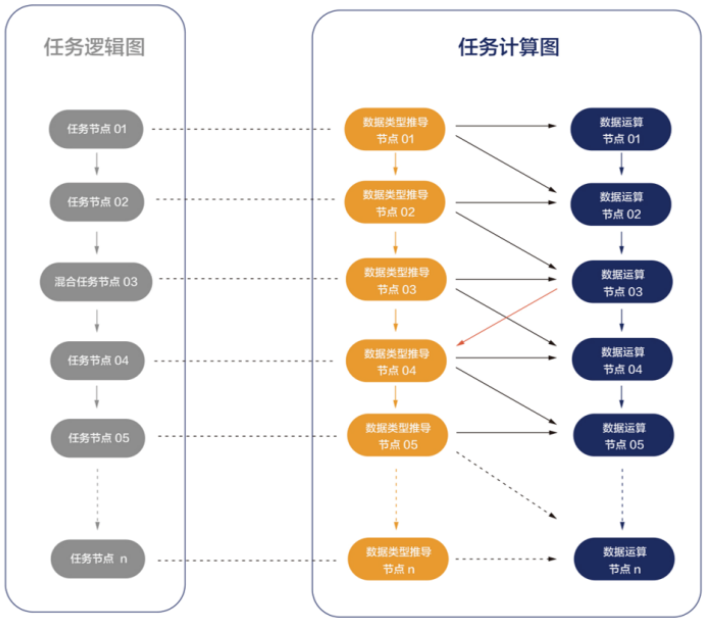


图 7：OneFlow 动态图机制

2.3 OneFlow 核心架构

与大多数框架类似的是，OneFlow 框架架构最上层有一系列模型库、Python 的接口层、支持动态图 define and run、静态图 define and run，底层是算子实现以及对编译器的调用。中间两层是与其它框架差异比较大的部分。从下图中我们可以看到 **OneFlow 的编译时比较厚，因为它做的事情比较多，运行时会非常薄。如果能在编译时做到很充分的工作，运行时就可以非常简单和高效。**

图 8：OneFlow 核心架构

2.4 人有我优，人无我有

接着，袁进辉比较了在模型吞吐率上 OneFlow 与其它国内外深度学习框架比较的评测结果。图 9 是一个来自第三方机构做的测试。我们可以看到，CNN 的 Benchmark 和 BERT 的 Benchmark 在这个比较简单的任务上如 BERT 上都能做到 90 多分，但 OneFlow 仍然可以推动进一步的提高，OneFlow 做到了“人有我优”。

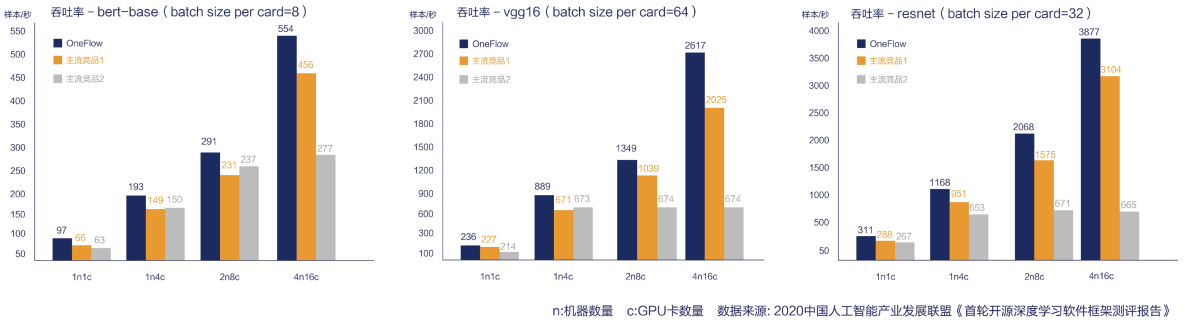


图 9：OneFlow 人有我优：数据并行

袁进辉指出，目前的开源框架直接套用解决不了一些大规模问题。如千万级的人脸识别需要模型并行和流水并行，以及工业级的广告系统 deep and wide 模型需要很大的 embedding table 的模型运行，如果不进行深度定制，开源框架也是不支持的。而 OneFlow 天生就支持数据并行、模型并行和混合并行，已经在安防领域成功实现千万级人脸识别。OneFlow 做到了“人无我有”。

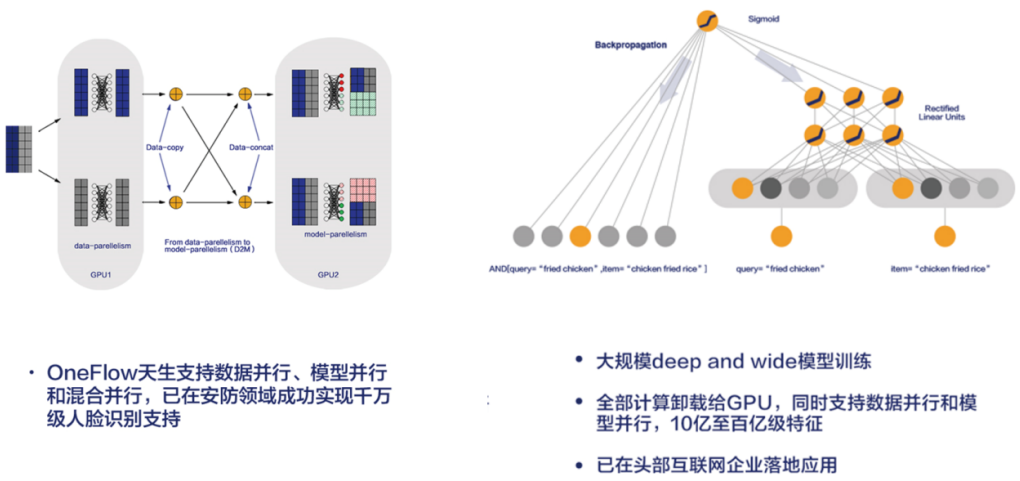


图 10：OneFlow 人无我有：大模型支持

接着袁进辉谈到如果要将横向扩展做到非常理想的状态，会带来很大的经济优势。英伟达推出新款芯片 A100，单看它的单精度浮点计算比上一代 GPU 提高的倍数不到 2 倍，显存提高的倍数也不到 2 倍，但是价格很贵，每块达到了 2 万美金。同样的钱买消费级显卡大概能够买 16 个 2028Ti。如果软件能够让多机多卡容易用，又有线性加速比，单精度浮点计算和显存都差不多提高了 10 倍。同样的 2 万块钱，不考虑电费、服务器费用，它能带来很大的收益。能不能做到这么好的横向扩展，关键取决于这个框架的能力。

图 11：横向扩展的经济优势

2.5 一站式深度学习平台 OneBrain

前面讨论比较多的是效率问题，一流科技除了解决效率的问题还要解决易用性和完备的问题，即框架要在什么环境下去使用。一流科技提供了一套非常完整的一站式的深度学习平台，用 K8S 和 docker 管理各种各样用户的作业、数据集、模型以及各种各样框架的镜像。

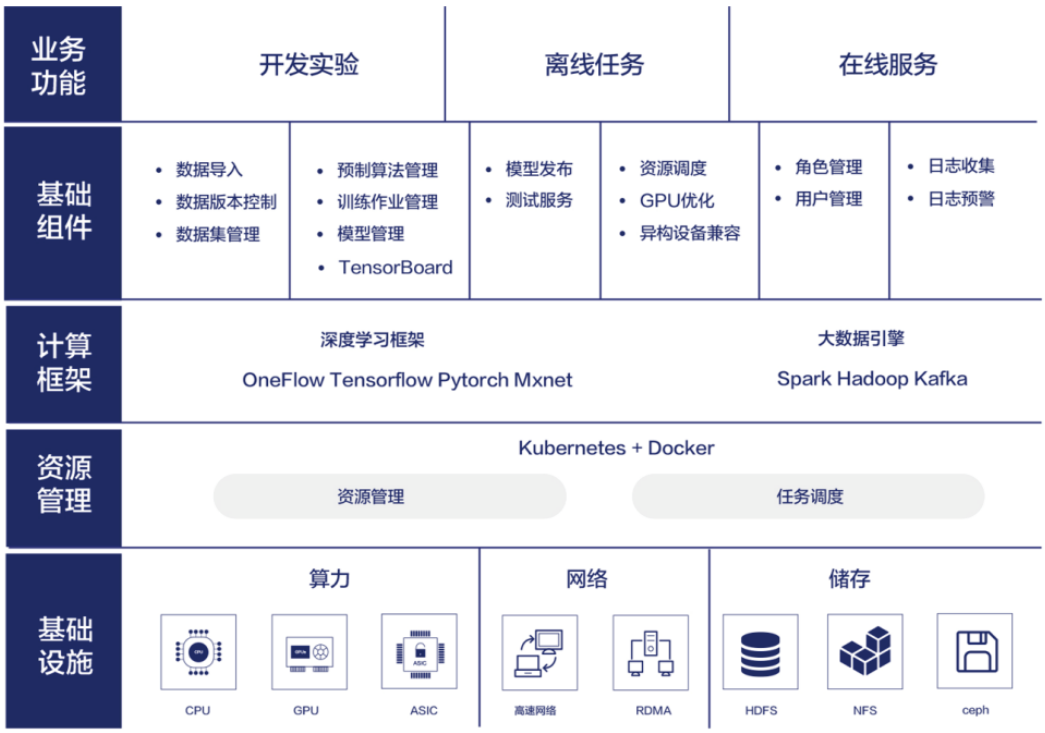


图 12：OneFlow 一站式深度学习平台

三、OneFlow 的差异化优势

框架发展到现在为止不到 10 年时间，也经历了一些迭代，前期主要是学术界在研发框架，到后面基本是工业主导，特别是从 TensorFlow 一统天下，到后面 PyTorch 崛起，现在到 2020 年还有什么可以发挥的地方？袁进辉认为这是每一位深度学习框架从业者需要思考的问题。

图 13：深度学习框架盘点

下面是深度学习框架市场份额的调研，都是采用第三方的调研数据。比如找工作最流行使用什么框架，GitHub 框架的使用情况，以及学术界发表论文使用框架的情况。可以看到 PyTorch 基本和 TensorFlow 用户量不相上下。

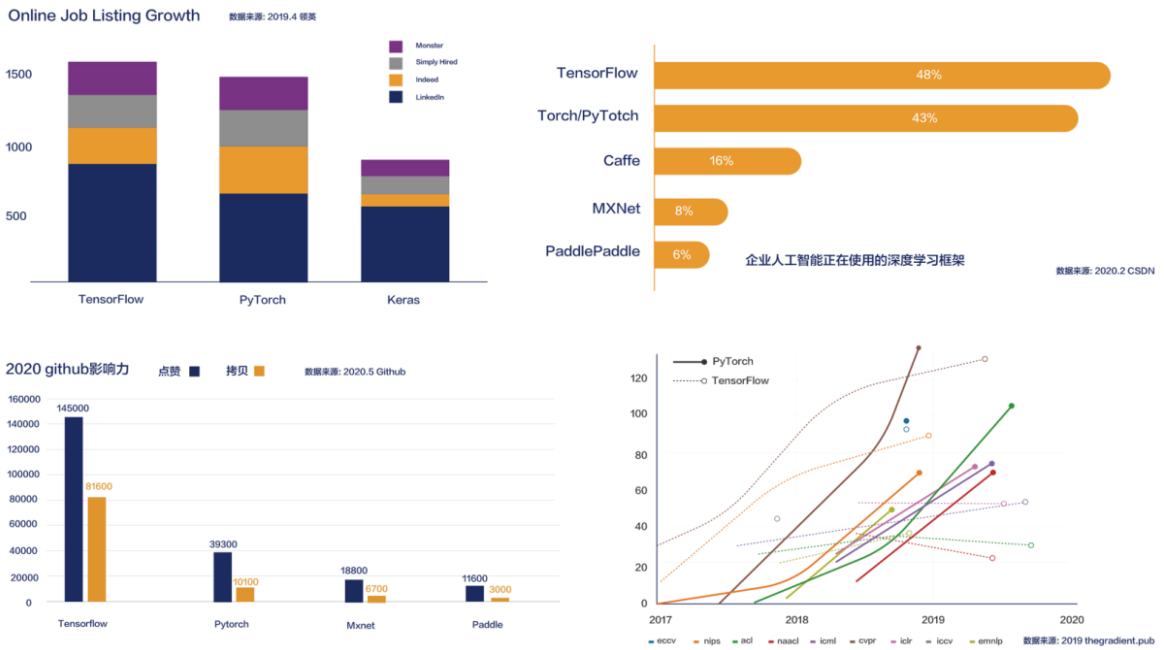


图 14：深度学习框架市场份额

从框架的关键用户指标来说，袁进辉认为最重要的有三个维度，一个是完备性，即用户想要的各种各样工具是不是都有，训练的、部署的、可视化的、推理的等等。另外一个是易用性，众所周知，PyTorch 做得非常极致，但是还有一个技术挑战很高的维度还没有被攻克，那就是高效性。效率又分单设备效率和集群层面的效率。单设备效率更多的是在编译器层面，集群层面的效率就是横向扩展的效率，它主要涉及分布式。**分布式里面数据并行相对来说是解决得比较好的，但更大参数量需要的模型并行和流水并行是现在开源框架还没解决的地方，一流科技想找的突破点是在这个地方。**

图 15：关键用户体验指标

下图是各个框架在完备性、易用性和高效性这几个维度上的优劣对比，袁进辉希望 OneFlow 开源之后给大家留下的印象是它在效率上做得非常出色。OneFlow 开源也在紧锣密鼓做准备，预计 OneFlow 在一个月之内就会和大家见面。袁进辉指出，**一个新的框架需要有差异化的竞争优势，才有可能在市场上立足**。

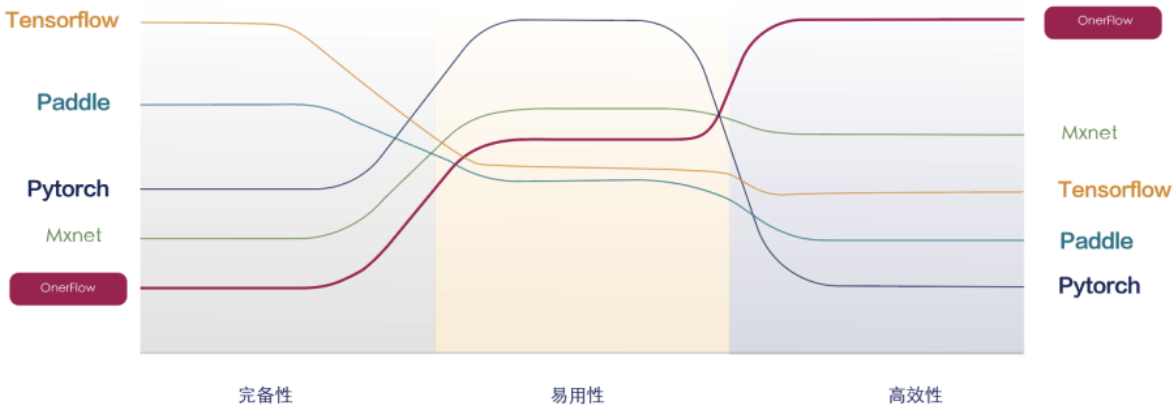


图 16：差异化竞争优势

为什么 OneFlow 首先从效率上攻克呢？袁进辉谈到，就像爬珠穆朗玛峰一样，有两个路线，一个是中国境内的北坡，一个是尼泊尔的南坡，从北坡的中国境内刚开始较为容易，但是 7000 米以上就非常困难。南坡在低高

度处有风险，但再高一些就容易跨越。

从框架角度来说，如果从效率先入手，就有点像从南坡去爬，从完备性和易用性角度去攻克的话就有点像从北坡去爬。对比一下就有点像各个框架的高度现在是不一样的，TensorFlow 和 PyTorch 爬的高度是非常高的，OneFlow 的高度现在是低的。但袁进辉认为后面 OneFlow 爬坡的容易程度会好一些，相信有机会更早爬到最高峰。

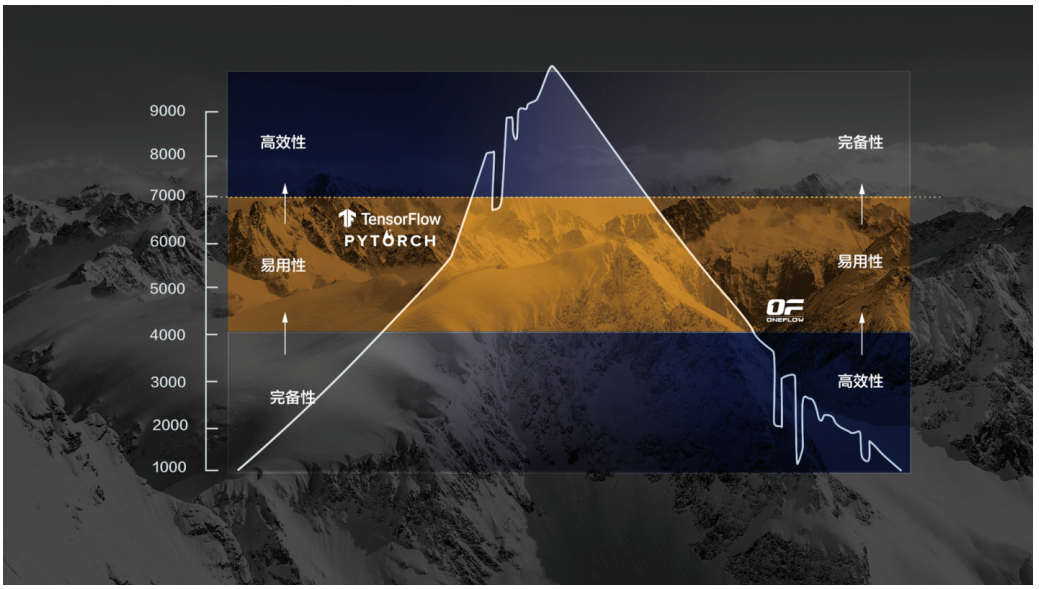


图 17：OneFlow 从高效性走向完备性

最后，袁进辉希望这个框架要交给开源社区的开发者去评价和筛选，也有行业一些公认的 Benchmark。就像擂台赛一样，很多常用的模型和很多常用的框架，可以放在擂台上进行比较。论输赢是次要的，更重要的是通过比较能够在技术的发展上得到更多启发，最后逐渐形成一个共识。比如框架从易用角度来说是不是一定要像 PyTorch 这么去做，从效率角度是不是一定要像 OneFlow 这个角度去做，大家在技术上可以形成螺旋式的上升。

四、结语

袁进辉的这场报告，向我们清晰地展示了一流科技 OneFlow 怎样解决目前深度学习横向扩展遇到的难题。相比 TensorFlow 和 PyTorch，OneFlow 从数据并行、模型并行和流水并行等多个方面入手，做充分的静态分析，视数据搬运为一等公民，使编译时较厚，运行时较薄，在效率方面做到了最极致。OneFlow 即将开源，高效性是它相对其它深度学习框架最大的差异化优势，让我们拭目以待。

附一：问答环节

主持人（陈文光）：如果你希望别人用你的东西，而且以性能打动他，有一种方式是兼容性做得特别好，完全不改程序，不费什么力气拿过来就可以用。如果换成你的框架就要花很多力气去改代码，那么这个时候你只提供性能优势是不是足够吸引用户来用？比如如果你能提供 5 倍性能，那大家愿意去改一改代码，如果你能提高 10 倍性能，大家愿意去重写代码。目前从你的角度，在分布式上、性能上和现有框架取得的优势大概是什么量级

的？你觉得这个量级是不是足够影响大家使用你的框架？

袁进辉：这个问题非常好，这个分两类，一类是像 CNN BERT 确实是 10%、百分之几十的效率，动力不太大。后面提到的需要模型并行的场景，比如工业级广告推荐，非常大规模的人脸识别，还有 OpenAI 的 GPT-3 都是需要模型并行的。基于现有的开源框架来实现这个的难度还是很大的。一方面是正确性的问题，还有一个是效率问题，使用 OneFlow 相信会比改写已有的框架容易很多，所以这个有一定的动力。再一个，在使用接口上，现在越来越趋于一致了，比如大家发现 PyTorch 容易用，所以在 Python 层面大家都非常相像，而且中间层的表示大家也是非常相似的。所以在上层接口层面相信它是呈收敛趋势的，大家越来越像，框架之间的迁移会越来越方便，这是我相信的趋势。

附二：更多关于 OneFlow 的参考资源

30 名工程师，历时 1300 天打造，又一“国产”AI 框架开源了

https://mp.weixin.qq.com/s/XolkGVL_dwzUOusHMz1uIA

TensorFlow 和 PyTorch 迎来了“后浪”

https://mp.weixin.qq.com/s/OptQtjr5nfkN_nb9cUB36Q

挑战 TensorFlow、PyTorch,“后浪”OneFlow 有没有机会

https://mp.weixin.qq.com/s/9JYPsb7exUgTrL8EfpRiYg

《中关村导刊》对一流科技进行报道

https://mp.weixin.qq.com/s/StrbHUr7CFTAxtori1xanQ

老师木：有了 Tensorflow，为什么我们还需要另外一个深度学习平台框架 | GIAC 访谈

https://mp.weixin.qq.com/s/uzoEMK2QuRsvSWN6n_WyHA

让 AI 简单且强大：深度学习引擎 OneFlow 背后的技术实践

https://zhuanlan.zhihu.com/p/63410526

圆桌论坛：我们是否还需要更多的编程框架？

整理：智源社区 张鼎盛

在第二届北京智源大会“AI 框架”论坛圆桌讨论环节中，七位领域内的专家齐聚线上，围绕“人工智能编程框架未来的技术创新”这个主题发表了各自独到的见解，他们分别是：上海纽约大学和亚马逊的张峥老师、清华大学胡事民老师、北京大学的董豪老师、阿里集团的白俊杰博士、袁进辉博士、寒武纪的刘道福博士、旷视科技的田忠博老师。智源学者清华大学教授陈文光担任主持人。

在圆桌讨论中，嘉宾们盘点了 AI 编程框架近些年来重要的进展，罗列了这个领域内重要的技术创新，如 TensorFlow、Pytorch 等，并结合自己在工业界或学术界的切身体会，阐述其对学术界和工业界的极高参考价值。下面是嘉宾们的精彩观点摘要——

问题：在智能编程框架这个领域，后续面临最重要的技术创新是什么？

张峥：我觉得挑战在于模型上还是有很多可以突破的地方，我自己做结构化的数据，但计算机在输入端读取的信号是没有结构的。举个例子，判别图片里是狗咬人还是人咬狗，也就是说狗和人的关系，这比判别其中是有人还是有狗重要的多。换句话说所有数据都是有语义和有结构的，如果没有把数据中的结构挖掘出来，深度学习是没有意义的。

我认为最重要的不在框架而在模型和数学上，这也是我诟病现在所有深度学习框架的一个原因，以为学习了 Python 或者学习了某一种框架就懂了深度学习，这其实是非常不好的，深度学习最后的动力在数学。平台造成了假象，让人们以为学习了 Pytorch，Tensorflow 等框架就懂了深度学习，其实这离真相还有很远。

胡事民：我们为什么要做框架？第一个，深度学习做科学工程冲击太大了，传统学科都被重新定义和思考。这是一个激动人心的时代，一切都在变革。这种时候框架的重要性是毫无疑问的，如果不用框架，什么都从头自己写，这是不现实的。第二个，如果中国没有自己的框架是肯定不行的。为什么中国这么多企业想做框架？华为如果不做，一旦收到美国的钳制，那就不行了。旷视科技做得这么好，如果没有自己的天元，一旦美国停止他使用 TensorFlow，那么大楼的底柱就塌了，所以这个是必须要做的。但怎样超越它是很难的。我们的窗口期很小，因为编程框架这个领域已经做得非常大了，TensorFlow 成功之后引起其他公司的高度警惕，像 Facebook、微软这些公司要抵制 TensorFlow，为什么后来 Pytorch 能够后来居上，一方面是动态图导致了原理上的创新，另一方面是各个公司或者集团之间利益的关系也是很大的。

所以我觉得如果想要超越，那么第一技术上要创新，如果我们重复现在的工作可以解决“卡脖子”问题，但是无法超越它，要想超越必须进行技术创新。框架的本质是计算图的问题，因为它是数据结构。对于计算图，利用算子融合把计算图整合起来是一个机会，他们各位嘉宾谈的很多观点都非常好，咱们现在中国国内是百家争鸣，希望有些尝试和努力能够成功，给中国的科技创新留下一些机会。这是一个竞争激烈的时代、激动人心的时代，但是窗口期非常小。作为清华大学的一份子，我们从技术创新角度来做是有价值的，也是人才培养重要一环，最终这些学生会走向业界，成为人工智能下一代骨干。

董豪：现在依然是百家齐放的时候，随着新的 AI 硬件出现，中美科技战，我们国内必须要发展自己的 AI 框架、一套 AI 的体系，这里面也会有很多的机会。我也很认同胡老师关于培养下一代人才的观点，人才培养体系非常重要。

技术方面，我觉得我们如果只是做一些增量性的工作，很难突围，我们需要投入更多努力来探索下一代框架，因为只有颠覆的技术才能快速地改变现在的状况。这里面可微分编程语言也慢慢地开始了，直接采用编译器的中间表达代替了现有的计算图引擎来表示计算（模型），这种方式可以充分利用现有编译器沉淀的优化技术去提升性能。另一方面，目前单机训练大家的效率都差不多了，下一步框架原生支持大模型（e.g., GPT-3）切分，自动分布式训练的将会是重点方向。

在这个过程中，我们也要注重生态的建设，虽然我们一直在讨论技术，但是我觉得生态反而是我们应该更加要关注的一点。现在国外的框架生态很强，而且国外巨头企业的投入很大，这样的话如果我们没有颠覆性技术的话，能够做得比较好的概率就很低了。我们今天有这么多企业、大学在一起，联合起来是很有希望的，现在已经很难一两个人去做这个事情了。

白俊杰：第一点是前端建模会往更纯粹的 Differentialable Programming Language（可微分编程语言）方向发展，趋势是从第一代的 Caffe，到 Tensorflow 出来，然后到 Pytorch，这个发展越来越灵活的，更靠近真正的编程语言，正好这个编程语言被选中了是 Python，但是在表达上越来越靠近图灵完备的编程语言。

刚才几位老师提到，现在深度学习研究里面出现一个问题是模型学习能力比较低，模型的规模在逐渐变大，如果在框架上不能支持更加灵活表达能力的话，会阻碍人工智能这个领域的探索发展。

比如像 TensorFlow 之前有一个工作是 swift，这个工作它尝试用 swift 做成一个完全可微分的语言，Pytorch 现在还是有些局限，可以求导的只是在张量上的操作，更原始的 Python 没有表达能力。框架在接下来几年如果能够把这个做好，是一个更能吸引用户的点。

第二个点，从我最近工作见到的例子来讲，框架如何帮助用户更好地从研究到生产进行过渡，也是一个非常重要的点。我们知道从生态上来说，从研究到落地，一个商业模式走向盈利然后反哺商业上去推动研究，这个过程是很重要的。

总结一下，框架创新上比较重要的两个点，一个是表达能力更趋向于完整可导的语言，第二个是从研究到落地怎么能更快速，我觉得是一个值得被开发的点。

袁进辉：我特别同意刚才胡老师说的观点，一个新的框架，比如 OneFlow，要想脱颖而出的话是一定要有创新的。同时也非常同意陈老师说的，框架研发里面人才是非常关键的，在座各位专家都做研究，应该都认同这样的观点，即创新通常不是堆钱和堆人就能必然发生的，有的时候它是有偶然性的，在满足一定条件的情况下才可能发生在少数人身上。所以对于国产框架来说，很多人参与、很多厂家参与是一个好事，从多个角度来做的成功率会提高。

回到在技术上还有什么可做的，我认为在效率上有很多可做的点，一个是在宏观层面，就是与分布式相关，在

宏观层面怎么解决网络墙的问题。还有一个是在微观层面，现在有很多学者和公司在研究自动代码生成，怎么实现只用数学的逻辑描述这个运算是什么，然后编译期能自动生成 GPU 上最快的或 CPU 上最快的二进制代码，我也觉得这是一个非常有挑战的问题。

刘道福： 我先从产品角度来说，产品吸引用户的话往往要有些亮点，一个框架一定程度上是个产品，那它的用户或者客户是谁呢？我觉得有这么几类： 最大用户是开发者，开发者追求的是易用，尤其在训练这块，易用的框架生态会建立得非常快。如果做到易用，那么对前沿算法和新结构的支持必须比较好。性能反而在前期对开发者来说不是那么在乎。 对训练这一块，我理解易用是比较重要的，因为大家关注的训练是有关开发效率，更少关注的是硬件成本或者其他的东西。但是对于推理来说，假如一个框架要支持推理的话，大家追求的是性能。因为推理业务尤其对一个大的企业来说，它的部署量往往是非常大的，假如能够把性能做好，做 5 个百分点能节省 5% 的硬件成本，这其实是非常大的收益。

大家对于训练和推理这两个维度的关注是不一样的，未来训练和推理会慢慢地分开，比如训练侧重于通用，能把效率做得很高。并且从技术上来说，训练往往是逐层的，推理往往要做各种层融合优化，所以我觉得训练和推理可能会有两个这样的框架。

但这里面就有一个问题，生态的起源往往是起源于训练，很多算法或者这些东西最早做出来的模型是这些开发者开发出来的模型，他往往可能是基于训练框架去做的，所以怎么把训练好的模型在推理框架上更好的执行，中间会有一些新的挑战。就是你做好一个东西，怎么让它更好地兼容一些东西，这里面需要有一些中间表示，能把这些东西都对接过来。

另外，我觉得在终端现在有个不一样的需求，大家从创业角度，在终端做个专门的框架或者引擎是有价值的，因为终端的产品和形态差异化特别大，它导致现在终端的人在开发 AI 应用是苦不堪言的。

我理解无论是训练、推理还是终端，会有分出来的需求，这些需求导致往下发展会有分支，分支完了以后对开发者是一个很友好的事，对我们做芯片的人来说，我们是希望这些框架都合在一起，我不需要支持那么多框架。

田忠博： 我的想法是最终回归到价值上来，因为旷视有自己的技术信仰、价值务实，我们最终还要回到我们做这个框架，做这些技术创新真正给用户带来什么。我特别同意道福的一点，我们的用户有两类，一类是研究者，一类是研发者，在旷视我们能够知道深度学习从产品到落地有多少环节、多少困难，我们做这个框架是为了自己更好，我们也希望这些经验能帮助更多人。

如果从现状来看，深度学习框架在价值维度上有三点:

第一点，我们希望它更快，首先性能肯定是大家非常关注的一点，性能意味着成本，意味着更好更大的模型，意味着的更多更好的商业机会。
第二点是灵活与易用，希望支持更多模型，支持更多范式，把更新技术尝试引进，而不是一个僵化的体系。
第三点，更规范。现在看到深度学习硬件设施雨后春笋般的蹦出来，这里面蕴涵着大量的机会。

从现状来看，这三点是我们衡量当前价值的点。我觉得在这方面，不管是大的框架、小的框架，是大公司还是

小公司，谁能提供价值，谁就能够有机会，而不是一个垄断的行业。
对于未来的看法，我更同意张峥老师的观点，它充满不确定性，现在回过头来看产业基础，并不是那么扎实。比如深度网络，它会对我们的框架有怎样的需求和冲击，这是不知道的，现在的框架并没有很好地利用这样的变化。

陈文光：我觉得现在还是在百花齐放的阶段，不管是模型、算法，还是理论，都有它的空间。现在的这个领域还不成熟，对于新的系统来说还是有机会的，不管从哪个角度去做它，都有不同的切入方向，比如易用性、性能、对新模型的支持，都有很多可以开发的地方，这个领域还是有很多机会的。

今天非常高兴请到了 7 位圆桌论坛的嘉宾，从各个角度、各个方向对这个编程系统做了不同的探讨，当然，这只是很小的一步，但是今天各位嘉宾的发言让我非常有收获，从张峥强调算法和模型的开放性，到语言，到工业界上面真正的需要，到生态要怎么建设，具体到我们自己应该做什么，到我们分布式能不能做好，这里面其实有非常多的技术突破和未来的发展机会。我们这个智源讨论会应该是每年一期的，今年很高兴讨论了这样一些问题，希望明年有这个问题再来讨论时，会有更多更好的成果，像张峥的想法一样，能够给整个社区、整个社会发展提供我们研究者应有的贡献。