英特尔中国
制造及能源行业
AI实战手册

intel ai

Contents

目 录

趋势篇

为工业制造企业勾画未来智能蓝图

新技术浪潮正推动着工业制造行业飞速革新，并以人工智能（Artificial Intelligence，AI）技术为代表，引领着第四次工业革命的进程。这一过程中，信息科技的进步、AI 技术的普及以及 5G 通讯技术的成熟，都将加速人类生产方式的迭代，与信息技术相伴的智力型劳动将大规模代替体力劳动，进而驱动行业实现转型升级。

作为“世界工厂”之一的中国，也正在拥抱第四次工业革命和信息化浪潮，迎接新的机遇与挑战，比如技术与市场的快速变化使传统工业制造、能源生产等企业转型需求加剧，压力与动力并存，且主要体现在以下方面：

- **生产成本的增加**：上下游产业的转型升级与环保压力的日益凸显等，使中国制造业成本已经大幅上升。来自波士顿咨询（BCG）的报告《全球制造业的经济大挪移》显示，以美国的制造成本为基准指数 100，中国的制造成本已高达 96[1]。加之经济发展进入新常态后，出口贸易等领域的税收优惠在日益减少，使得无论是中小企业还是大型企业，都面临着生产、管理成本增加的严峻挑战；

- **市场需求的变化**：市场消费需求趋于个性化、差异化、定制化。消费者对于衣食住行的需求，已经快速从“满足温饱”转向对个性定制和差异化的需求，且越来越便利的网购和物流使潮流更迭越来越快。以前单一款式、大批量生产的节奏和模式已难以适应新的消费需求。

新的挑战往往伴随着前所未有的机遇。对于目前的工业领域，要缩减成本，就需要优化生产和流通环节，压缩人力成本和开销，提高良品率；要满足个性化定制化需求，就需要更好地控制每一个生产环节，以更细的颗粒度去优化每道生产工艺，用更弹性的方式去运营生产和协调供应链。这些需求，无疑为 AI 在传统工业制造中的应用提供了丰富的场景。

例如，利用机器视觉、智能预测等技术，不仅能使产品故障率大幅下降，且节约原材料，缩短因设备检测带来的停机时间，更能通过自动化检测帮助企业大幅减少人力成本；另外，通过深度学习等 AI 方法构建的产能预测解决方案，还能帮助企业根据生产效率和市场需求的变化，优化生产工艺和排期。

利用 AI 带来的技术革新、生产效率优化以及运营效率提升，不仅可以帮助传统制造企业从容应对成本和市场带来的挑战，加快产业升级，更可以完成从人力密集型到技术密集型的转换，实现“弯道超车”。也是凭借对产业变革的强大驱动力，AI 在工业制造领域的影响正日益扩大。在质量监控、产量提升、故障监控、维保预测、能源管理、机械臂控制以及市场分析预估等使用场景中，越来越多的 AI 软硬件产品及解决方案正发挥越来越大的作用，市场前景广阔。如图 1-1-1 所示，来自 Tractica 数据表明，到 2025 年，工业制造领域的 AI 投资规模将超过 130 亿美元[2]。

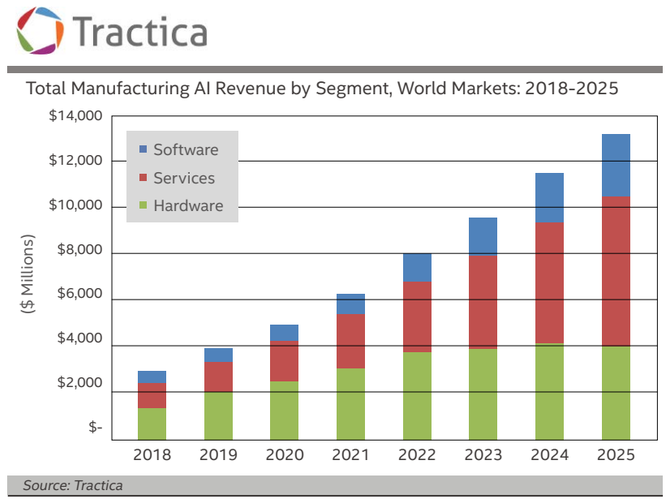


图 1-1-1　工业制造领域的 AI 营业额规模预测

[1] 沿用互联网媒体相关报道：http://www.199it.com/archives/400017.html
[2] 数据及插图引自：https://tractica.omdia.com/newsroom/press-releases/artificial-intelligence-technologies-are-quietly-penetrating-a-wide-range-of-enterprise-applications/

与其他领域相比，工业制造领域涉足信息化和自动化的历史堪称久远，一般规模以上的工业企业都有自己的 IT 数据中心，并部署有企业资源计划系统（Enterprise Resource Planning, ERP）、供应链管理系统（Supply Chain Management, SCM）、客户关系管理系统（Customer Relationship Management, CRM）等多个应用系统。但这些用于业务和生产数据的处理手段都有其局限性，在企业实施智能制造的转型中，由于生产流程、产品规格等不确定性的增加，致使其弊端也愈来愈明显：

- **使用成本高**：传统信息化、自动化方案一般都是软硬件一体的集成型方案，且只能部署在固定生产线上。一旦产品发生变化，就需要进行大规模改造。同时，方案中的工具和设备也需要专人维护，使成本高启；
- **灵活性不足**：传统信息化、自动化方案基本都是根据特定产品进行开发，灵活性较差。设备在进行部署时，需要对产线进行调整，且对安装位置、尺寸以及参数设定等都有严格要求。一旦产线出现变动，方案的适用性也随之下降；
- **开放兼容性差**：传统信息化、自动化方案往往缺乏数据连通接口。即便有接口，不同厂商的设备之间也缺乏统一标准，无法快速形成数据合力，难以供模型训练、推理等所用。

因此，AI 在智能制造中的运用，需要针对 “固定问题” 和 “通用需求” 配备一套市场既有、且经过同行、友商验证过的解决方案。而方案的落地，也需要进一步地将计算力 “下沉”，即尽可能将基于 AI 方法的数据处理和分析能力部署到最接近生产一线的边缘侧，打造直接驱动生产力爆发的强劲引擎。

如图 1-1-2 所示，通过深度学习、机器学习等方法所构建的智能解决方案，可以让部署在生产一线的产能预测、故障预警、瑕疵检测等功能端，与在其它位置部署的云端数据中心、企业知识库、市场营销部门以及其他产线形成数据连通，构建个性化的信息及数据处理分析模式，以便让原先固定的生产调度和检测技术变得更灵活、实时、便捷，为未来勾画具备自我调节、自我适应和自我修复的智能工厂蓝图。

目前，AI 应用在工业制造有两个方向上备受关注，分别是基于机器视觉（Machine Vision）的工业辅助检测和基于时序（Time Series）数据的智能预测。

得益于图像采集硬件、深度学习算法以及边缘计算等技术的蓬勃发展，基于机器视觉的工业辅助检测目前已在电子制造、汽车、纺织等领域获得了广泛的应用。与人工检测相比，其依托先进的镜头技术和成像技术，以及推陈出新的深度学习、机器学习算法，呈现出多项优势：

- **精度更高**：机器视觉具有 256 级，相对于 64 级灰度的人类视觉，无疑识别能力更高，同时还可辨析天体级至微米级的不同目标；
- **光感更宽**：人眼仅可识别 400nm-750nm 范围的可见光，而机器视觉可以轻易地感知红外线、紫外线等不可见光；
- **速度更快**：人眼无法追踪高速物体，而机器视觉系统中的工业相机快门可至微秒级；
- **数据更丰富**：利用机器视觉系统，用户可以留存产线的全部过程数据，方便进行在线和离线分析；
- **稳定性更高**：机器视觉系统没有疲劳，不会出现人工检测中常见的漏检、错检问题。

而基于时序数据的智能预测，则能帮助众多制造、能源领域企业提升产能预测、维护性预测以及智能运维等能力，进而增加收益：

- 企业有能力对每台生产设备的生产能力和健康状况作出更精准、更全面以及更细颗粒度的预估；
- 企业不再通过事后排障来对生产设备实施检修维护，能有效避免因设备故障停机带来的经济损失；
- 企业可通过 IT 智能运维，更好释放 IT 资源潜力，助力业务发展。

这些优势与收益，让基于机器视觉的工业辅助检测以及基于时序数据的智能预测成为了传统工业制造企业实施智能转型的左膀右臂。这一过程中，来自英特尔的至强® 可扩展处理器平台、OpenVINO™ 工具套件英特尔® 发行版（以下简称 “OpenVINO™ 工具套件”）、Analytics Zoo “大数据分析 +AI” 平台等一系列先进产品与技术，为智能方案成功实施和运行提供了坚实基础。通过这些高性能硬件基础设施和软件框架，企业的生产和运营可以实现从自动化向智能化的转变；面对优化管理和运营问题，企业从 “求人” 变成 “求己”，从 “找方案” 变成 “找数据”，进而生产运营也变得更灵活、更弹性、更智能，有能力在第四次工业革命浪潮中乘风破浪，扬帆远航。

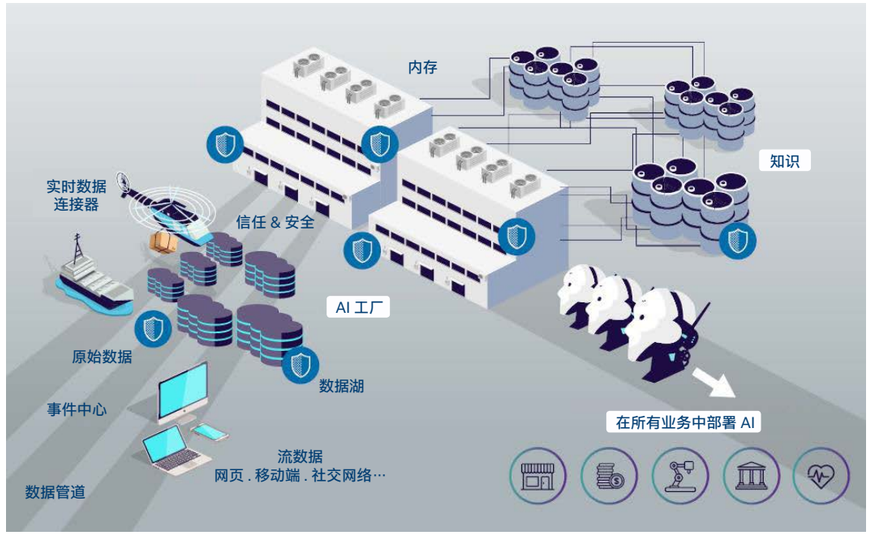


图 1-1-2　AI 在工业制造领域的部署

下一部分，将结合多个工业制造领域的实际案例，围绕 AI 技术在这些典型应用中的部署情况，共同探讨英特尔相关技术与产品在真实场景中的应用和优化方案。

实战篇

助力用户选择更优模型和架构，推动 AI 机器视觉落地智能制造

机器视觉与智能制造

智能制造的全新 “视界”

由互联网大潮掀起的技术进步，推动着智能制造成为传统制造行业面向未来、寻求突破的关键路径。通过融合机器人、大数据、云计算、物联网以及 AI 等多种技术，智能制造凭借更高效、环保和敏捷的特点，成为制造业转型的全新模式。

这其中，自动化系统、机器人等技术帮助智能制造解决了传动和控制问题，而融合了计算机视觉和 AI 技术的机器视觉，则为智能制造带去了明睿的 “慧眼”，在产品检测、条码辨识、外观测量等多个领域都能为制造业带来效率提升。

如图 2-1-1 所示，机器视觉系统的基本架构，是通过工业相机等图像采集装置，将目标转换成图像信号，再通过网络设备传送到后端处理系统。系统根据目标形态、像素分布、亮度、颜色等信息，抽取目标特征，最终得到判别结果，并利用工控机（工业机器人、机械臂、传动轴等）来控制相关设备。

在一些典型的机器视觉应用场景中，例如纺织工厂，可以使用这一方法进行纺织品的瑕疵自动化检测。由于纺织机械运行速度非常高，流水线速度可达数米每秒，漏针、破洞、错针等瑕疵往往在毫米以下，依靠人工识别的方法难以保证检测质量。通过引入机器视觉，纺织产线不仅可以准确地记录缺陷发生的时间与位置，还能与生产控制系统相关联，根据检测情况执行启停。

与传统人工方法相比，机器视觉在精确、客观程度、可重复性、成本以及效率上都有明显的优势，特别是在高速运行的流水线作业中，采用机器视觉的辅助检测方法，可以大幅提升工厂的生产效率和自动化程度。因此，越来越多的企业正开始在产线中引入机器视觉系统，一项数据表明，至 2020 年，全球机器视觉市场总额将至 269 亿美元 [3]。

传统机器视觉亟待与 AI 更深度融合

利用工业化视觉系统来提升自动化生产效能并非新生事物，早在上个世纪就有很多企业开始了这方面的探索与部署。但传统的机器视觉方案存在许多不足和局限，主要体现在以下几个方面：

- **成本昂贵，使用门槛高**：传统的机器视觉方案，往往是由大型生产设备供应商负责开发部署，其算法和软件都以紧耦合方式固化在工业相机等类特定硬件上。一旦检测精度或检测品类需要调整，就需要联系原设备厂商进行升级或改动，而复杂的生产环境带来的大量非标准化特征识别需求，会导致方案调整周期长、成本高；同时，更换设备也会迫使整个生产线停机重启。

- **灵活性差**：传统机器视觉方案往往都基于固定识别模式开发，灵活性较差，导致在部署检测系统时，需要对相关产线进行调整，且对检测对象的位置、尺寸及摆放方向都有严格要求，被检测对象的任何偏离也都会造成检测结果的不可信。同时，由于传统机器视觉方案与硬件紧密耦合，在部署时，需要机械部件配合定位，会占用很大的产线空间，对位置、环境以及温湿度等也都有更高要求，因此很难做到全产线、全流程化部署。

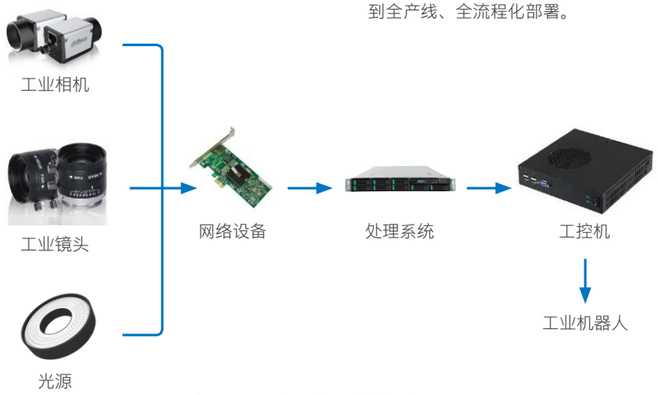


图 2-1-1　一种典型的机器视觉系统架构

[3] https://www.prnewswire.com/news-releases/global-markets-for-machine-vision-technologies-278425321.html

- **开放性兼容性差**：传统机器视觉方案一般都采用软硬一体紧耦合的模式，方案商同时也是设备商。因此，难以在同一套设备上集成多种机器视觉方案，即便工厂对设备实施二次开发，也要付出极高的成本。如果更换设备供应商，那么品控水平还可能会发生差异。

为应对这些问题，近年来，越来越多的企业都尝试将深度学习、机器学习等 AI 方法引入传统的机器视觉方案，并取得了良好的效果。

首先，在成本方面，以深度学习框架为基础的 AI 机器视觉方案是一个完全开放的平台，其核心是开源的深度学习框架和通用的计算设备（例如基于英特尔® 架构的服务器），企业完全可以用开放平台来训练基于自己产线的图像数据，生成满足自身要求的检测算法和应用。当应用需求发生变化后，企业也无需依赖外部资源，可自行在新的数据集上进行模型和算法的迭代优化。同时，完全解耦的软硬件架构也使升级和应用整合变得更容易。具有横向和线性扩展能力的英特尔® 架构平台，配合开源深度学习框架，可弹性满足用户多样化的部署和整合需求，无需重新开发和调整算法。

其次，在灵活性方面，深度学习是完全基于大数据的 AI 数据分析和处理方法。它不需要预设任何模式或框架，只需大量的样本数据和适当的标定，即可实现自学习和生成推理模型。因此，通过深度学习框架，企业可以彻底摆脱设备供应商的束缚，自主采集数据，形成自己的机器视觉应用标准和规范。由此形成的方案源于企业的实际应用需求，独立于设备供应商之外，即便更换设备也不会影响品控。

合理算法模型，助力智能制造事半功倍

精度更高的分类和目标检测算法

工业检测是现代化制造业中不可或缺的流程。传统上，工厂主要通过人眼识别或工具辅助方式进行产品检测，效率低，准确率也不高。AI 机器视觉的出现，能逐一弥补这些不足，为产品检测精度和效率带来成百上千倍的提升。基于机器视觉的工业辅助检测系统需要使用高效、适宜的图像分类、目标检测以及实例分割等深度学习算法。

■ YOLO 算法

传统的 R-CNN、Faster-RCNN 等目标检测算法，由于目标识别和目标定位分离的设计模式，都无法达到工业辅助检测所需的低延迟和实时性。近年来出现的 SSD（Single Shot MultiBox Detector）、YOLO（You Only Look Once）等新算法，将目标识别和目标定位两个步骤合二为一，大幅提升了检测效率。

如图 2-1-2 所示，YOLO 算法将目标检测任务转换成一个回归问题，将每个输入图像都划分成 S × S（例如取为 7 × 7）的网格，每个网格都要预测两个 bounding box 的坐标（x,y,w,h），box 内包含检测目标的置信度（confidence），以及检测目标属于预设类别中每一类的概率。在后续的筛选层中，则会选出合适的 bounding box 作为结果。

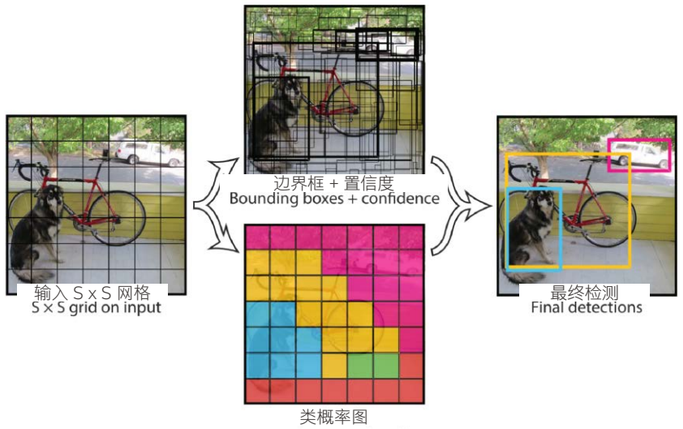


图 2-1-2　YOLO 算法模型

由于将目标识别和目标定位网络合二为一，YOLO 算法在检测速度上有了巨大突破。基础 YOLO 算法的检测速度可达 45 帧 / 秒，超过 24 帧每秒的人眼感知速度，达到了实时性要求，而 Fast YOLO 等新版本，更可将速度提升至 155 帧 / 秒。

更多 YOLO 算法详情，请参阅 You Only Look Once: Unified, Real-Time Object Detection，Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi
https://arxiv.org/pdf/1506.02640.pdf

■ SSD 算法

虽然 YOLO 算法在执行速度上有了巨大提升，但检测准确率相对不足。SSD 算法弥补了这一短板，融合了 YOLO 算法的回归思想以及 Faster R-CNN 算法中的候选框机制，在执行速度和检测准确率上都有不俗表现。

如图 2-1-3 所示，SSD 算法通常基于传统的图像分类网络，例如 VGG、ResNet 等展开（图中为 VGG-16）。经过一系列的卷积层和池化层，得到一定尺寸的特征图，再在这个特征图上进行回归，得到物体的位置和类别。

与其他算法不同的是，SSD 有多个候选框和特征图。在神经网络中，浅层的特征图往往包含更多细节信息，适合较小目标的检测；较深的特征图包含更多全局信息，适合大目标的检测。因此，通过在不同特征图上对不同尺寸的候选框进行回归，SSD 算法在对不同尺寸的物体进行检测时结果更加精准。

更多 SSD 算法详情，请参阅 SSD: Single Shot MultiBox Detector, Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, Alexander C. Berg
https://arxiv.org/pdf/1512.02325.pdf

■ ResNet 算法

残差网络（Residual Net，ResNet）是目前常用的分类模型算法之一。众所周知，在深度神经网络中，网络层数越多，输出的特征表达能力越强。但过深的网络也会带来梯度消失（Gradient Disappearance）问题和退化（Degradation）问题，由此反而引起分类准确率的下降。

ResNet 算法通过创新的结构来有效地解决这一问题。如图 2-1-4 所示，在 ResNet 中可以构成多个残差块结构，其输入与期望输出相等，构成一种恒等映射的关系。通过这样的结构，可让深度神经网络在不断增加深度的同时保持准确率。

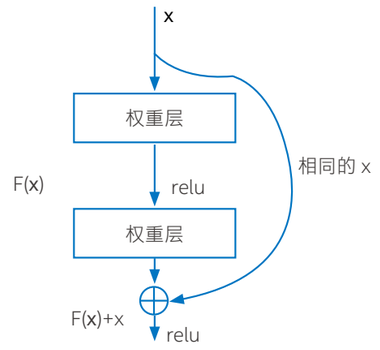


图 2-1-4　ResNet 残差块构造

ResNet50 是 50 层的 ResNet 网络模型，也是最常用的分类模型之一，主要包含了恒等块（Identity Block）和卷积块（Convolutional Block）2 种主要的结构，2 种块的主要差别是近道连接（Shortcut Connection）上是否进行了卷积操作。ResNet50 是由以上两种主要块结构形成的 5 个卷积阶段组成。

更多 ResNet 算法详情，请参阅 Deep Residual Learning for Image Recognition，Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun
https://arxiv.org/pdf/1512.03385.pdf

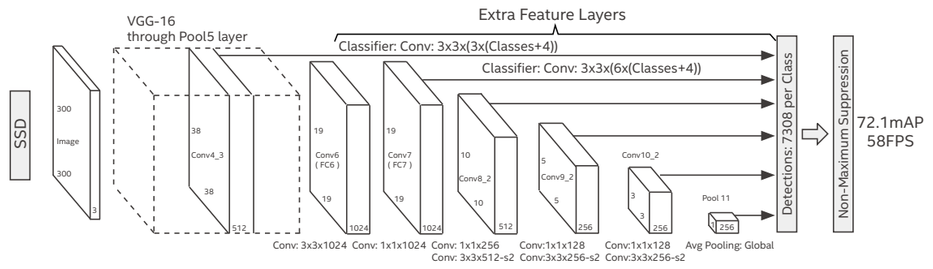


图 2-1-3　SSD 算法网络架构

更轻锐的轻量级算法

随着机器视觉在工业辅助检测场景中的广泛应用，其训练、推理的部署也逐渐不再限于大型数据中心的服务器中。尤其随着5G时代的到来，移动边缘计算（Mobile Edge Computing，MEC）的发展让更多企业和机构倾向于将AI应用向边缘侧部署。因此，机器视觉也需要引入更为轻锐的轻量级目标检测算法模型。

SSDLite + MobileNet V2模型是近年来备受瞩目的轻量级目标检测算法模型之一。通过将轻量级网络Mobilenet V2替换传统SSD算法中的VGG部分，同时将SSD算法中的普通卷积替换为深度可分离卷积（Depthwise Separable Convolution），不仅可有效提升SSD算法的检测准确率，也能够让检测速度有所飞跃，且模型比原本SSD算法缩小数倍。

现在，SSDLite + MobileNet V2模型已集成在Caffe、TensorFlow等经典深度学习框架，企业可方便地使用开放API调用。

更多MobileNet V2算法详情，请参阅MobileNet V2: Inverted Residuals and Linear Bottlenecks，Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, Liang-Chieh Chen
https://arxiv.org/pdf/1801.04381.pdf

缺陷检测中算法选取的常用原则

基于图像数据的缺陷检测通常会用到图像分类、目标检测以及实例分割等深度学习方法。那么，在日常缺陷检测应用中如何选择合适的算法呢？以下是一些常用的原则：

- 需要明确缺陷检测本身是否有定位的需求。如果仅仅是分类，而没有对于目标物体位置的诉求，可以直接采用分类算法进行尝试，利用预训练好的拓扑模型通过模型微调（fine-tuning）的方式应用到自有的数据当中。常用模型有：ResNet、Inception、DenseNet、ShuffleNet等，可根据模型的复杂度和精准度需求来选择尝试；
- 如果缺陷检测本身需要将目标物体的位置标出，可以采用检测或者分割的方式。这两种模型的一个区别是，实例分割是从像素级做目标的分类，而检测只需要给出目标物体的检测框。常用的检测模型有：R-CNN类、YOLO类和SSD类。可根据具体的需求来选择，通常YOLO类和SSD类是满足具有实时性需求，并可面向边缘设备使用的轻量级模型；
- 针对实例分割通常使用Mask-RCNN。这个模型相对较重，更适合于计算能力较为充裕的边缘云或数据中心环境；
- 如果缺陷检测精度要求较高，可能需要多个模型组成一个模型管道，通过多个模型的组合达到最终的检测目的。

美的工业视觉检测云平台

■ 项目背景

作为白色家电行业领军企业，美的集团（以下简称“美的”）希望通过完整、可复制的产品缺陷检测方案，来完善其智能制造产业链中的关键环节。如前面提到的，由工业相机、工控机以及机器人组成的传统视觉方案存在诸多问题，例如定制化方案开发周期长、成本高，检测内容多样化造成参数标定繁琐、工人使用困难，占用产线空间大，对工艺流程有影响。

因此，美的希望通过新的技术方法来优化和升级检测方案，打造以下能力：

- 对单个检测项目形成通用的推理算法，并可推广至不同产线；
- 可在任何产线上做到无缝部署，不干扰现有生产和工艺流程；
- 在无人工干预情况下做到高鲁棒性，并在全天候高频次下，保证准确率和延迟的稳定；
- 整个检测过程在100毫秒以内完成，识别率达到98%以上。

来自生产一线的海量数据资源，让美的具备了利用AI技术，特别是深度学习方法，来解决上述问题的基础，并通过与英特尔展开深入的技术合作，提升了算法和算力。如图2-1-5所示，美的通过前端高清图像采集、后端训练推理的架构，构建了基于深度学习的工业视觉检测云平台，为旗下各产线提供瑕疵检测、工件标定、图像定位等一系列辅助检测能力。

在这一过程中，英特尔不仅为新方案提供了Analytics Zoo大数据分析和AI平台（https://github.com/intel-analytics/analytics-zoo），来构建从前端数据预处理到模型训练、推理，再到数据预测、特征提取的全流程，还针对美的各生产线的实际检测需求，为新方案选择了轻锐的SSDLite + MobileNet V2算法模型并实施优化，令新方案进一步提升了效果。

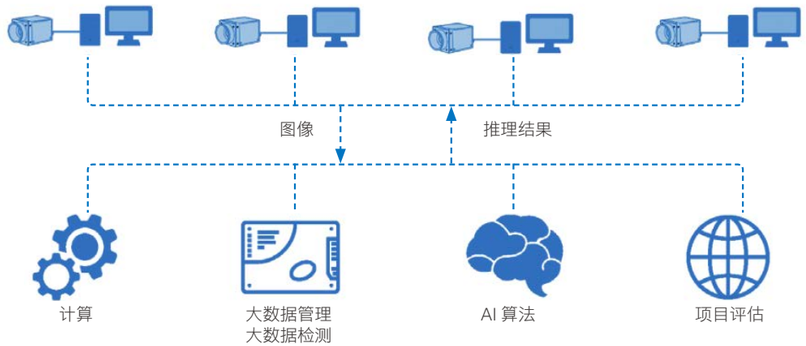


图 2-1-5　美的预设的机器视觉检测云平台架构

■ 基于 Analytics Zoo 的端到端解决方案

如前所述，美的设计的机器视觉检测云平台架构主要由前、后端两部分组成，由工业相机、工控机等设备构成图像采集前端，部署在工厂产线上，经云化部署的英特尔® 架构服务器集群则组成云平台的后端系统。

图 2-1-6　执行微波炉缺陷检测的工业机器人

在前端，执行图像采集的机器人通常装有多个工业相机，或进行远距离拍摄，用于检测有无和定位；或进行近距离拍摄，用于光学字符识别（Optical Character Recognition，OCR）。以微波炉划痕检测为例，如图2-1-6所示，当系统开始工作时，通过机器人与旋转台的联动，先使用远距离相机拍摄微波炉待检测面的全局图像，并检测计算出需要进行OCR识别的位置，再驱动近距离相机进行局部拍摄。相机采集到的不同图像，先由搭载英特尔® 酷睿™ 处理器的工控机进行预处理，根据检测需求确定需要传输到云端后，再将数据传送到后端云服务器，实施深度学习训练和推理。

虽然这一架构并不复杂，但新方案要达到美的希望的灵活、敏捷和高通用性却并非易事。尤其是以端到端方式构建从数据采集、模型训练、算法部署的全流程，如果中间每个环节都由美的自行建设，势必会耗费大量的时间和开发成本，且容易造成软硬件紧耦合和扩展性差的问题。

使用Analytics Zoo来解决以上问题，实为明智之举。Analytics Zoo融合了Apache Spark、TensorFlow、BigDL等多种技术框架，可直接运行在英特尔® 架构服务器构建的大数据集群上，并可通过对英特尔® 至强® 可扩展处理器进行深度优化，充分释放强大的性能潜力。同时，Analytics Zoo所集成的英特尔® 数学核心函数库（Intel® Math Kernel Library，英特尔® MKL）与多线程技术，也帮助美的工业视觉检测云平台大幅提升特征训练、图片预测以及数据批处理等效率。

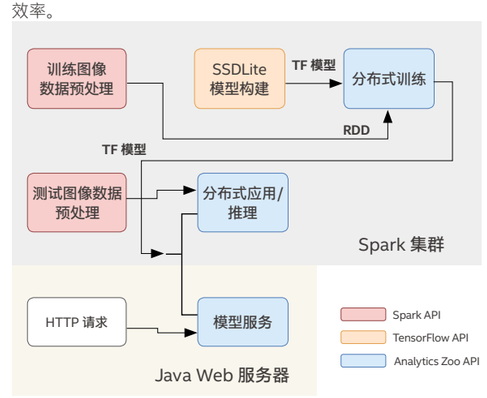


图 2-1-7　基于 Analytics Zoo 的美的新方案流程

通过双方的紧密合作，英特尔帮助美的在其新方案后端的云服务器中，基于 Analytics Zoo 构建了端到端数据分析流水线方案。整个方案流程如图 2-1-7 所示，包括以下几个主要步骤：

1. 通过 Spark，方案以分布式方式处理来自各产线工业相机获取的大量视频和图像。其中，Analytics Zoo 使用 PySpark 从磁盘中读取视频或图像数据并进行预处理，构造出 TensorFlow Tensor 的弹性分布式数据集（Resilient Distributed DataSet，RDD）。整个训练流程可以自动从单个节点扩展到基于英特尔® 架构服务器的大型 Hadoop / Spark 集群，无需修改代码或手动配置。

2. 使用 TensorFlow 目标检测 API 接口，直接构建对象检测模型，例如，可以采用轻量级的 SSDLite + MobileNet V2 模型。

3. 直接使用在第一步中预处理的图像 RDD，以分布式方式在 Spark 集群上训练（或微调）对象检测模型。例如，为了以分布式方式处理缺陷检测流水线的训练数据，方案使用 PySpark 将原始图像数据读取到 RDD 中，然后应用一些变换来解码图像，并提取边界框和类标签。方法如下所示：

```
train_rdd = sc.parallelize(examples_list)
   .map(lambda x: read_image_and_label(x))
   .map(lambda image: decode_to_ndarrays(image))
```

而返回的 RDD（train_rdd）中的每条记录都包含一个 NumPy ndarray 的列表（即图像、边界框、类和检测到的框的数量），它可以直接用于创建 TensorFlow 模型，并在 Analytics Zoo 上进行分布式训练。通过创建 TFDataset（如下所示），可以实现这一功能。

```
dataset = TFDataset.from_rdd(train_rdd,
            names=["images", "bbox", "classes", "num_detections"],
            shapes=[[300, 300, 3],[None, 4], [None], [1]]],
            types=[tf.float32, tf.float32, tf.int32, tf.int32],
```

4. 训练结束后，可以基于与训练流程类似的流水线，直接使用 RDD 评估图像数据集，使用 PySpark、TensorFlow 和 BigDL 在 Analytics Zoo 上，以分布式方式在 Spark 集群上执行大规模模型评估（或推理）；

5. 使用 Analytics Zoo 中 POJO 模式的 API，将整个 Pipeline 轻松地部署于在线 Web 服务中，以实现低延迟的在线服务（例如，Web 服务、Apache Storm、Apache Flink 等）。实现代码如下：

```
AbstractInferenceModel model = new AbstractInferenceModel(){};
model.loadTF(modelPath, 0, 0, false);
List<List<JTensor>> output = model.predict(inputs);
```

更多有关详细信息，请参阅：
https://analytics-zoo.github.io/master/#Programming Guide/inference/

通过这样的方法，新方案可以对预处理过的图像进行识别，提取出需要进行检测的标的物，例如螺钉、铭牌标贴或型号等，并通过不断地迭代分布式训练提高对检测物的识别率。最后，系统会将识别结果传递给机械臂等自动化设备来执行下一步动作。

值得一提的是，英特尔® 至强® 可扩展处理器为新方案提供了另一项关键要素：计算力。部署在该云平台中的英特尔® 至强® 可扩展处理器得到了充分的性能优化，其英特尔® 高级矢量扩展 512（Intel® Advanced Vector Extensions 512，英特尔® AVX-512）等技术得以大展拳脚，以出色的并行计算能力，满足了该云平台在模型训练和模型推理时对算力的严苛需求。

■ 基于英特尔® 架构优化的目标检测算法模型

如前文所述，提升基于机器视觉的工业辅助检测系统的工作效能，关键在于为其选择高效、适宜的目标检测。美的的新方案选择了更适于实时目标检测的 SSDLite + MobileNet V2 模型。

利用 Analytics Zoo，新方案使用 TFDataset 来表示一个分布式存储的记录集合，每条记录包含一个或多个 TensorFlow Tensor 对象。这些 Tensor 被直接用作输入，来构建 TensorFlow 模型。

如以下代码所示，方案通过 TensorFlow Object Detection API 构建了 SSDLite + MobileNet V2 模型：

```
# using tensorflow object detection api to construct model
# https://github.com/tensorflow/models/tree/master/research/object_detection

from object_detection.builders import model_builder

images, bbox, classes, num_detections = dataset.tensors
detection_model = model_builder.build(model_config, is_training=True)
resized_images, true_image_shapes = detection_model.preprocess(images)
detection_model.provide_groundtruth(bbox, classes)
prediction_dict = detection_model.predict(resized_images, true_image_shapes)

losses = detection_model.loss(prediction_dict, true_image_shapes)
total_loss = tf.add_n(losses.values())
```

在模型构建之后，方案首先加载预先训练的 TensorFlow 模型，然后使用 Analytics Zoo 中的 TFOptimizer，通过以下方式对模型进行微调训练：

```
with tf.Session() as sess:
    init_from_checkpoint(sess, CHECKPOINT_PATH)
    optimizer = TFOptimizer(total_loss, RMSprop(LR), sess)
    optimizer.optimize(end_trigger=MaxEpoch(20))
    save_to_new_checkpoint(sess, NEW_CHEKCPOINT_PATH)
```

最终方案在验证数据集上的成效可达 0.97 mAP@0.5。

■ 方案成效

将深度学习的方法引入工业辅助检测领域，不仅让美的工业视觉检测云平台可以快速、敏捷、自动地识别出待测产品可能存在的问题，例如螺钉漏装、铭牌漏贴、LOGO 丝印缺陷等。更重要的是，该云平台能够良好适应非标准变化因素，即便检测内容和环境发生变化，云平台也能很快适应，省去了冗长的新特征识别、验证时间。同时，这一方案也能有效地提高检测的鲁棒性，克服了传统视觉检测过于依赖图像质量的问题。

新方案在美的产线中实际部署后，达到了很好的应用效果。从已有 9 条产线的实际部署测试数据来看，该方案对现有产线的影响几乎为零。同时，由 Analytics Zoo 提供统一的数据分析 + AI 平台，大幅降低了方案进行分布式训练和推理以及提供低延迟在线服务所耗费的人力物力成本。相比传统的工业视觉方案，如图 2-1-8 所示，项目部署周期缩短了 57%，物料成本减少 30%，人工成本减少 70%[4]。

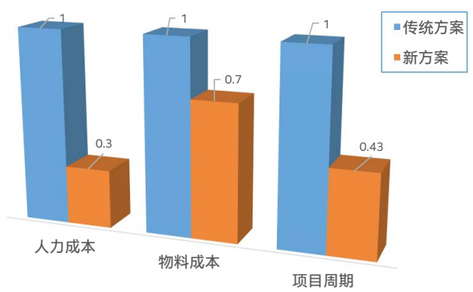


图 2-1-8 归一化的美的工业视觉检测云平台方案成效对比

同时，经英特尔优化的 SSDLite + MobileNet V2 目标检测算法模型也有效提升了方案的执行效率和准确率。来自一线的数据表明，方案对诸多缺陷的识别率达到了 99.98%，推理预测时间从原先的 2 秒缩减到现在的 124 毫秒[5]。

更多示例以及优化细节，请参阅 Github 相关代码：
https://github.com/intel-analytics/analytics-zoo/blob/master/pyzoo/zoo/examples/tfnet/train_lenet.py

打造敏捷自动化的缺陷识别系统

智能工厂中的自动化缺陷识别

良品率是指通过测试的产品与理论产量的比例，良品率越高，意味着设备产能越大，收益率越高，因此良品率一直是制造业的核心指标之一。良品率的提升有赖于工厂产线缺陷检测能力的提升，优秀的自动化缺陷检测系统可以尽早地发现和定位问题，并与生产系统形成正反馈，迭代提升良品率。

传统上，制造企业通过人工方法进行缺陷检测。但这种依靠密集劳动力的方法不仅效率低下，准确率也不高，且抬升人力成本。而在精密制造、医疗、安全设备等高精制造领域，产品中的超细微缺陷，更是超出了人力检测的能力范围。

随着机器视觉技术的蓬勃发展，更多制造企业正利用基于机器学习的机器视觉技术，构建更为敏捷和高效的自动化缺陷识别（Automated Defect Classification，ADC）系统，通过采集高清生产影像，从中计算提取出不同的识别特征来用于缺陷识别和分类，并从缺陷分析中厘清引发问题的根本原因。

利用全新一代英特尔® 至强® 可扩展处理器带来的强劲算力，包括英特尔工厂在内的生产企业正通过 ADC 系统的部署，在晶圆封装、测试等核心生产环节中，实现更高效的缺陷检测能力，大幅提升良品率。

自动化缺陷识别助力英特尔工厂有效提升产品品质

■ 项目背景

晶圆的加工、封装及测试是芯片制造中的关键步骤。英特尔工厂需要在其产线上部署同步缺陷测量（Inline defect Metrology）的流程，从微观的芯片布线中发现极小的瑕疵，并将瑕疵与最优产品设计基线（Optimum Design Baseline）做对比，进而了解整个产线的健康状况。

过去，这项任务由专职员工来完成。但统计数据表明，培养一名员工实现 90% 的检测准确率，需要 6-9 个月的时间，但随着时间的推移，人工检测准确率会从 90% 逐渐下降到 75%-80%。究其原因，一方面是因为疲劳和重复劳动会降低人工检测的生产效率，另一方面，日新月异的工艺进步，也会使员

[4] 该数据来自美的与英特尔合作项目未公开的测试报告：《美的微波炉底板螺钉检测项目成本核算 - 方案对比报告》。
[5] 测试配置：英特尔® 酷睿™ i7-7700T 处理器，64GB 内存；测试流程为：系统读取图片并使用英特尔 ® Analytics Zoo 所集成的 TFNet，舍去前 10 次时延结果，并取接下来 100 次时延结果的平均值。

工的经验无法跟上变化。同时，一些新技术、新工艺的采用，例如图 2-1-9 所示，在晶圆封装流程中，由环氧树脂喷涂工具（左图）引起的缺陷和瑕疵，或焊球相关焊点偏移（右图）带来的问题，尺寸都在微米级甚至更小，使得肉眼辨别非常困难。

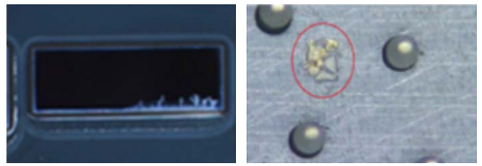


图 2-1-9　英特尔工厂中的细微瑕疵

为此，英特尔工厂在晶圆产线中引入了基于机器视觉的 ADC 系统，并在实践中获得了良好的效果。

■ 基于机器学习方法，构建高效敏捷的自动化缺陷识别系统

英特尔工厂希望利用基于机器视觉技术的 ADC 系统来解决晶圆生产、封装流程中的自动化缺陷检测识别，并将检测结果反馈至生产系统，用于产品质量检测和问题原因分析。

解决方案架构如图 2-1-10 所示，首先，由工控机控制的高清扫描式电子显微镜（Scanning Electron Microscope，SEM）会根据需求，采集高清图像推送至其后的分类模型推理服务器集群。在分类推理服务器上，每张高清图像都会根据对应晶圆层级的模型（每一层级的晶圆都有对应的模型）进行实时标注，找出缺陷瑕疵并进行相应的分类识别。分类识别的结果将被送至分析数据库作分析比对。

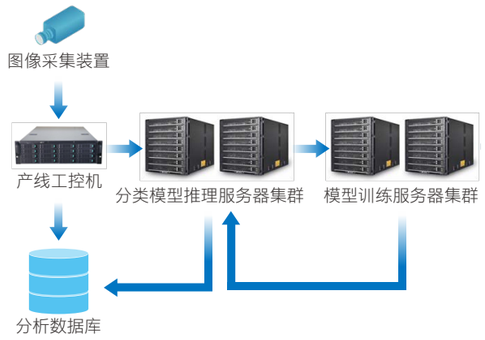


图 2-1-10　英特尔工厂 ADC 系统架构

如果模型识别到新的特征，做过标注的图像数据会被送到更后端的模型训练服务器上进行模型迭代和优化。迭代优化完成之后，更新的模型会被再次推送到前端的分类模型推理服务器上进行升级。无论是数据提取、处理、还是模型训练，方案都采用了分布式的并行处理的架构，可以方便地根据业务需求实现横向扩展。

新方案包括了以下几项技术亮点：

- **数据分类**：在晶圆的各个层级上，有过数以千计的不同缺陷和瑕疵。英特尔全球工厂在进行晶圆检测时，对大量不同的缺陷识别和定义进行了预搜集。在新方案中，这些既有数据被转换成了多种训练模型，并按照技术、生产流程、晶圆层级以及产品等对这些数据进行了大量的预处理，极大提升了推理分析效率；
- **缺陷识别**：方案会根据缺陷来定义识别模型，如果模型的推理结果并未达到预期效果，方案会引入人工干预手段对模型进行微调；
- **特征学习**：根据不同产线的需求，方案加入了基于不同算法的自动特征学习能力，包括传统机器视觉的自动外形标注、CNN 深度学习算法等；
- **原因分析**：通过基于英特尔 ® 架构的处理器实时计算检测出的数千种特征，方案能为工程师反馈带有关键信息的特征，帮助其利用缺陷逆向工程来找到缺陷产生的原因。

整套方案使用了第二代英特尔 ® 至强 ® 可扩展处理器作为训练与推理的计算引擎，其所具备的更多处理器内核和线程，以及全面优化升级的微架构，为方案带去了更强劲的计算力。同时，这一系列处理器所配备的采用矢量神经网络指令（VNNI）的全新英特尔 ® 深度学习加速（Intel® Deep Learning Boost，英特尔 ® DL Boost）在推理速度上有着卓越的表现，与上一代产品相比，性能提升高达 30 倍[6]，有力提升了方案的检测效率。

■ 方案成效

目前，利用 ADC 系统，英特尔工厂可在不升级产线、不添加任何额外设备的情况下，将高效、敏捷的自动化产品缺陷瑕疵检测流程应用于生产、封装测试的多个流程，获得了以下收益：

- 产品良品率得到大幅提升；
- 缺陷检测可靠性更高；
- 有效提高了现有设备的投资回报率；
- 管理人员能更快了解整个生产制造流程的健康状况；
- 快速地自动定位瑕疵发生的原因。

[6] 数据援引自英特尔官网：https://www.intel.com/content/www/us/en/technology-provider/products-and-solutions/xeon-scalable-family/moving-ai-to-the-edge-article.html

“云 - 边 - 端”构建完备 AI 解决方案

网络技术发展驱动 AI 架构革新

以 5G 为代表的新一代网络技术的迅猛发展，在改变人们生活的同时，也使 AI 基础设施架构悄然发生变化，且在制造业中表现尤为明显。如图 2-1-11 所示，传统制造企业产线与 IT 设施分布往往是多层、竖井式的：企业总部的数据中心一般会部署 ERP、SCM、CRM 等核心 IT 系统；分厂、分公司的服务器上会部署生产管理相关的调度系统；而底层的生产线上，则是工控机一类的操作设备。

这样的架构会对 AI 系统的构建和运行带来阻碍。如前所述，在传统 IT 架构中，各层级的线上线下数据流通都是垂直式的，被局限在各个分厂、各个子系统中间，缺乏横向连接，因而无法聚合成为 AI 应用的强效动力引擎；其次，总部数据中心也往往不能与生产一线的自动化操作系统、检测系统实现数据互动，导致优化策略无法实时下达到生产一线，AI 应用难以实现迭代优化。另外，来自车间、产线的海量边缘数据需要进行实时采集与分析，而且对响应时延也有着苛刻的要求，这对传统集中式的数据中心模式也提出了严峻挑战。因此，要推动智能制造系统的发展，必须对 AI 基础架构做出合理调配甚至革新。

推动业务下沉，在网络边缘构建有效的 AI 处理机制是应对这一挑战的有力手段。随着边缘技术的发展与成熟，更多基于边缘、端、云协同的 AI 应用模式正走向前台，并显现出多项优势：

- **缓解网络带宽与数据中心压力**：在制造企业中，设备会产生海量数据供训练和推理使用。通过在边缘侧对数据实施分拣和预处理，能有效减轻网络带宽与数据中心的压力；
- **增强设备响应能力**：边缘计算的近距离服务模式能减少因网络连接和路由不稳等因素带来的长时延。部署在边缘侧和端侧的轻量级模型，可以实时地将推理结果反馈到产线；
- **提升数据安全性**：位于企业生产内网的边缘计算节点，为企业关键性隐私数据的存储和使用提供了基础设施，关键数据可在边缘侧完成推理而无须上传云端，大幅提升了数据安全性。

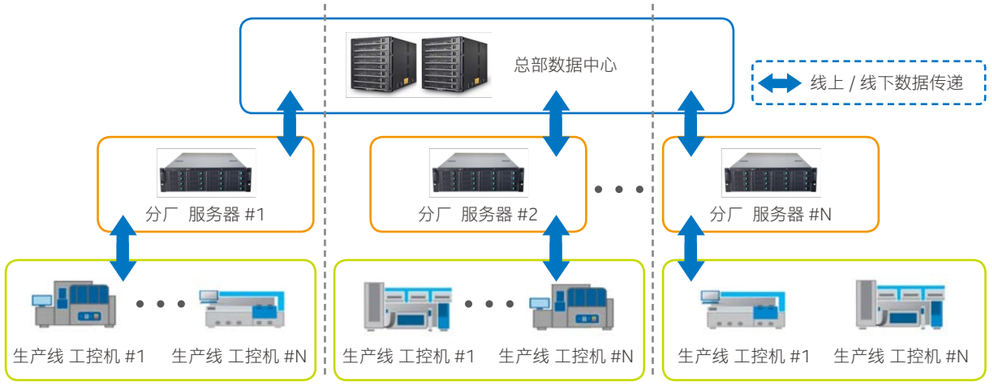


图 2-1-11　传统制造企业产线与 IT 设施分布

动力锂电池全生产流程缺陷检测方案

■ 项目背景

该方案来自某全球领先的锂电池研发和制造企业。面对不断增加的市场需求，该企业积极引入了智能制造技术，对多种锂电池的各个生产环节进行调控与优化，提高生产效率，在保持优异品质的同时突破产能瓶颈。

以该企业的核心产品之一——动力锂电池为例。动力锂电池的基本单元是电芯。每一个完备电芯的生产都必须经过极其严格的褶皱、暗斑、掉料以及绝缘膜异常等瑕疵的缺陷检测，以保证最终产品的可靠性与安全性。但在大规模产线上，如果采用人工检测等传统方式来执行毫米级的缺陷检测，不仅速度慢、耗时巨大，精细和准确度更是无从谈起。即便引入基于工业相机的计算机图像辅助检测等自动化方法，也存在缺乏扩展性和灵活性等问题，无法有效应对新产品导致的新瑕疵形式的检测，限制了产能。

为有效应对以上问题，这家企业在英特尔的支持下，利用 AI 方法构建全新的动力电池缺陷检测方案。通过对产能需求的评估，该企业希望新方案能够达到单条产线 423FPS（Frame Per Second，帧率）的检测速度，同时检测准确率达到 1DPPM（Defect Part Per Million，每百万的缺陷数量）。

新方案一方面根据动力电池产线的实际部署情况，以基于英特尔® 架构的平台为基础，构建云（总部云数据中心）- 边（边缘计算节点）- 端（产线工控机、工业相机）的架构，并引入英特尔® 至强® 可扩展处理器、Analytics Zoo 和 OpenVINO™ 工具套件，以及面向英特尔® 架构优化的 PyTorch 等软硬件，形成端到端的机器视觉缺陷检测方案；另一方面，根据检测场景的差异，方案中也部署了多种不同的深度学习和机器学习算法模型，让检测速度和准确率均获得了显著提升。

■ “云 - 边 - 端”协同，构建基于机器视觉的缺陷检测平台

为构建高性能的缺陷检测平台，双方首先从基础架构入手，根据总部云数据中心、各产线的生产管理系统、各类检测设备在缺陷检测流程中的不同作用，以及所处的不同场景带来的特定需求，设计出“云 - 边 - 端”协同的方案。“云”端的总部数据中心，可以利用强大的计算能力和来自各产线的丰富数据，根据生产场景需要进行集中化的模型训练，再将训练好的模型发布给“边缘”和“端”侧。“边缘”计算节点部署在分厂或产线服务器中，主要包括推理服务器、模型管理器以及模型仓库等组件，用于较重模型的推理，并将推理结果推送至产线质量控制系统中。“端”则位于工厂内每条生产线上，主要执行图像采集、预处理、预分类及轻量级推理工作。

这一架构经部署后对提升缺陷检测效率效果显著。如图 2-1-12 所示，首先，方案采用了分层推理的方案。从前文可知，无论哪种目标检测算法，都会耗费庞大的算力和带宽（用于数据传输）资源；且离产线越远，检测时延就越高。在新方案中，端侧系统采用开源的 OpenCV 计算机视觉库对采集的图像流实施预处理，并将预分类等简单工作负载部署在基于英特尔® 酷睿™ i5/i7 处理器的工业 PC 上，且使用轻量级模型进行推理，将结果直接反馈回产线，应用效率极高。

对于目标检测、图像分割等较“重”的工作负载，则通过边缘计算节点完成。这些节点由基于第二代英特尔® 至强® 可扩展处理器的服务器（集群）构建，可以从云数据中心调取合适的模型和参数，并通过英特尔提供的统一大数据分析及 AI 平台 Analytics Zoo 来构建分布式的推理方案。

而云端数据中心则主要承担高强度模型训练、推理以及管理职责。除了由基于第二代英特尔® 至强® 可扩展处理器的服务器构成高性能计算集群外，云端还配备了可扩展的中心存储数据库，存储各类中间过程数据以及最终模型和参数。

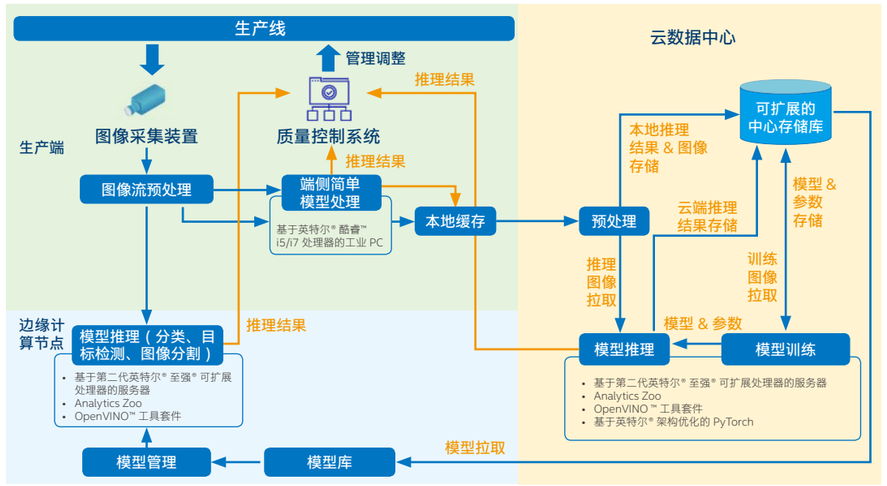


图 2-1-12　工业视觉平台系统架构图

经过本地预处理的海量图像流汇集到云端后，方案采用 Labelme 标注工具对数据集中的图像进行标注，并将标注后的特征类别及位置信息传输到计算集群中进行训练和推理。Labelme 工具不仅可以标注各种形状，还具备图像分类、目标检测、场景分割、实例分割、视频标注等功能，可以很好覆盖动力电池缺陷检测的范围。更重要的是，该工具支持像素级的细粒度标注，有助于提升标注效率与准确度。

值得一提的是，云端的算力虽然充沛，但其远离产线，实时性会受到一定影响。新方案在云端引入了面向英特尔® 架构优化的 PyTorch 框架，以及 OpenVINO™ 工具套件来进一步加速推理过程。原生 PyTorch 深度学习框架内置了强大的视觉工具包 torchvision，包含目前流行的数据集、模型结构和常用的图片转换工具，可轻松应对各种图像检测场景。新框架不仅继承了原生 PyTorch 简洁、灵活的特点，还引入面向深度神经网络的英特尔® 数学核心函数库（Intel® Math Kernel Library for Deep Neural Networks, 英特尔® MKL-DNN），其包含的高度矢量化、线程化的构建模块，能有效提高框架在基于英特尔® 架构的处理器上的运行速度，配合 OpenVINO™ 工具套件所提供的模型优化器、指令集优化等功能，令新方案获得了非常好的推理性能。

最后，Analytics Zoo 的引入使“云 - 边 - 端”协同架构的运行变得更为顺畅。这一架构将 Spark、PyTorch、OpenVINO™ 工具套件以及其它软件和框架，无缝集成到同一管道中，有助于新方案将数据存储、数据处理以及训练推理的流水线整合到统一的基础设施，不仅大幅提升新方案的部署效率、资源利用率和可扩展性，也能减少硬件管理以及系统运维成本。

■ 针对不同检测场景，采用适宜检测算法

在这家全球领先的锂电池生产制造企业的动力电池产线中，有三种主要的动力电池缺陷检测场景：绝缘膜间隙检测、正负极偏差检测以及绝缘膜异常问题检测。不同的场景对检测环境、检测速度、检测精度以及检测参数都有不同的要求。通过缜密的技术分析，英特尔帮助该企业针对不同检测场景部署了不同的目标检测模型。

■ 绝缘膜间隙检测

绝缘膜是电池充放电时锂离子传输的重要介质，其间隙过大或过小都会影响电池的性能，因此在生产中需要严格把控绝缘膜的间隙范围。但如图 2-1-13 所示，绝缘膜的厚度仅为毫米级别，对检测精度提要求高。

图 2-1-13　绝缘膜间隙检测图

英特尔在方案中建议采用 Mask R-CNN 目标检测模型，来实现精细的绝缘膜间隙检测流程。Mask R-CNN 模型是 Faster RCNN 算法模型的一个分支，特点是可对检测目标实施逐像素的分类，进而确定图像中检测目标的类别和位置，并对其进行分割，尤其适合精密检测场景的使用。采用 Mask R-CNN 模型对图片进行像素级分类，分割出检测边缘，再通过 OpenCV 测量实现产线所需的 0.3-3.9mm 的测量需求，超过该范围即可确定为缺陷电池。

■ 正负极偏差检测

在动力锂电池生产过程中，正极片、绝缘膜、负极片三层材料会叠压在一起进行卷绕，正常的电池正负极需交替出现，且个数一定。如图 2-1-14 所示，图片中细长的为阴极，粗的为阳极。如果出现单个极连续出现或者个数不符情况，电池即可被视为存在缺陷，需及时进行自动纠偏调整来控制质量，这对实时性的要求非常高，处理延迟要求在数十毫秒内。

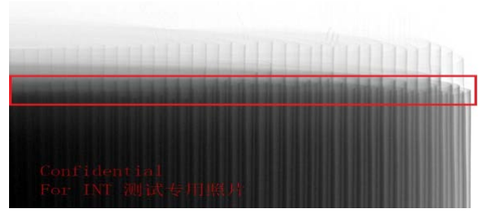


图 2-1-14　正负极片偏差检测图

英特尔在方案中建议采用轻量级快速目标检测模型——YOLO v3来进行正负极偏差检测。如前文（第14页“YOLO算法”部分）所述，YOLO算法模型的主要特点就是检测速度高，而 YOLO v3模型作为其轻量级进阶版本，在检测准确率和推理速度上有了进一步的提升，尤其适用于诸如动力电池产线正负极偏差检测所需的实时性和小型目标敏感检测。

■ **绝缘膜异常问题检测**

绝缘膜异常问题检测主要用于避免动力电池中的绝缘膜异常，导致正负极接触而引发短路事故。如图 2-1-15 所示，绝缘膜非常薄，因此该检测对精细度和准确率要求非常高。在经典的深度神经网络中，网络层数越多，能够提取到的图像特征越丰富，也更符合该类检测的需求。但随着网络深度的增加，退化（Degradation）问题也随之产生，即准确率会先上升直至饱和，如果继续增加深度，准确率反而会下降。

图 2-1-15　绝缘膜破损、丢失、褶皱问题检测实例

ResNet 可有效解决这一问题。其由多个残差块和恒等映射块拼接而成，与一般深度神经网络相比，能有效避免深层网络的梯度消失和退化问题。因此，英特尔在方案中建议采用经典的 ResNet50 残差网络实施训练。

■ 混合学习方法和迁移学习训练，提升检测效率和准确率

经过产线检测实践发现，通过单一的深度学习方法获得更优的准确率与召回率，需要手工对 logits 进行适当的调整。这无疑给整个检测过程带来了一定的不确定性，并增加了使用难度。为此，英特尔在方案中推荐采用机器学习中的支持向量机（Support Vector Machine, SVM）分类器与 ResNet50 残差网络一起，组成混合模型来实施检测，同样也可以达到类似的优化效果。SVM 分类器能够依据支持向量与分类超平面间隔最大化的原则，通过多次训练迭代，寻求最优的分类超平面来实现数据分类。

针对绝缘膜异常检测中的多分类（multiple-class）问题，SVM 能将其分解为多个二分类问题，再构造多个分类器来解决。ResNet50+SVM 的组合方案，不仅很好地解决了绝缘膜破损、丢失、褶皱等异常问题的检测难题，还大幅提升了检测效率和准确率。

模型的检测效率和准确度除了与选择合适的模型相关外，还需要有充足的训练数据。一般情况下，要满足实用要求，数据集量级需达到百万级甚至千万级。但在实际产线中，如此高量级的数据集很难收集，且采用大数据集从头训练也需要耗费大量时间及资源。

针对这种矛盾，英特尔在新方案推荐采用迁移学习训练（Transfer Learning for Training）方法。这一方法可以根据已有的预训练源模型进行微调（fine-tuning），将源模型的初始参数重新配置，直接从最后一层或最后几层开始重新训练，这样只需依靠少量数据集作为样本，并在训练集中加入曾经预测错误的图片，就可以提升模型在新样本中对于该类别的训练准确度。通过调整，这一措施得出的模型精度甚至可以和采用大数据集从头开始训练的模型相媲美。

■ 基于英特尔® 架构的代码示例

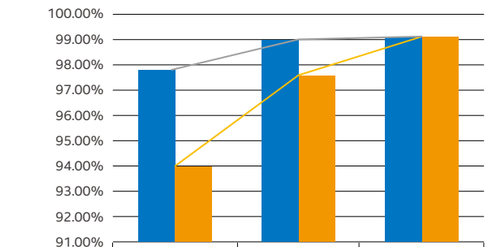
为了帮助合作伙伴更好地探索优化方案，英特尔也从其解决方案的实际需求出发，给出了多样化的配置优化代码示例。以下是使用 PyTorch，对 ResNet50 残差网络做模型微调的参考代码：

```
import torch
from torchvision import datasets, models, transforms
import torch.utils.data as data
from torch.optim import lr_scheduler
import os
import time

# train the model and save the best model
def train_save(model, loss_func, optimizer, scheduler, n_epochs):
    start = time.time()
    best_acc = 0.0
    best_weights = model.state_dict()
    print('start training.......')
    for epoch in range(n_epochs):
        print('*'*30)
        print('epoch:{}'.format(epoch))
        for phase in ['train', 'val']:
            epoch_loss = 0.0
            epoch_acc_num = 0
            if phase == 'train':
                model.train()
            else:
                model.eval()
            for X_batch, y_batch in data_loaders.get(phase):
                X_batch.to(device)
                y_batch.to(device)
                prediction = model(X_batch)
                loss = loss_func(prediction, y_batch)
                if phase == 'train':
                    optimizer.zero_grad()
                    loss.backward()
                    optimizer.step()
                epoch_loss += loss.item() * X_batch.size(0)
                epoch_acc_num += torch.sum(torch.max(prediction, dim=1)[1] == y_batch)
            epoch_loss = epoch_loss / len(image_datasets.get(phase))
            epoch_acc = epoch_acc_num.item() / len(image_datasets.get(phase))
            if phase == 'train':
                scheduler.step()
            elif phase == 'val' and epoch_acc > best_acc:
                best_acc = epoch_acc
                best_weights = model.state_dict()
            print('{}: loss:{:.2f}, accuracy:{:.4f}'.format(phase, epoch_loss, epoch_acc))
    end = time.time()
    print('*' * 30)
    print('finish training.......')
    print('training time is {:.4f} s'.format(end-start))
    print('best accuracy is {:.4f}'.format(best_acc))
    model.load_state_dict(best_weights)
    return model

if __name__ == '__main__':
    data_path = 'path to your data'
    # define your data transformation
    data_transforms = {
        'train': transforms.Compose([transforms.RandomResizedCrop(size=224), transforms.RandomHorizontalFlip(),
                                     transforms.ToTensor(),
                                     transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])]),
        'val': transforms.Compose([transforms.Resize(size=256), transforms.CenterCrop(size=224), transforms.ToTensor(),
                                   transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])])}

    image_datasets = {x: datasets.ImageFolder(os.path.join(data_path, x), transform=data_transforms.get(x)) for x in
                      ['train', 'val']}
    data_loaders = {x: data.DataLoader(dataset=image_datasets.get(x), batch_size=4, shuffle=True, num_workers=4) for x in
                    ['train', 'val']}
    device = torch.device('cuda:0' if torch.cuda.is_available() else 'cpu')

    # load pretrained resnet50 model
    model_ft = models.resnet50(pretrained=True)
    fc_in_features = model_ft.fc.in_features
    loss_func = torch.nn.CrossEntropyLoss()
    model_ft.fc = torch.nn.Linear(fc_in_features, 2)
    torch.nn.init.xavier_normal_(model_ft.fc.weight, gain=torch.nn.init.calculate_gain('relu'))
    model_ft.to(device)
    optimizer = torch.optim.SGD(params=model_ft.parameters(), lr=0.001, momentum=0.9)
    scheduler = lr_scheduler.StepLR(optimizer=optimizer, step_size=7, gamma=0.1)
    model_ft = train_save(model_ft, loss_func, optimizer, scheduler, 25)
```

■ 方案成效

创新的架构以及适宜检测算法的运用，使该企业的电池生产全流程缺陷检测方案一上线，就获得了良好的效果。实际部署后，单条产线的检测速度和准确度都超过了预期指标。

在满足产线所需的检测精度和检测速度之外，新方案在目标检测模型的创新应用上也获得显著效果。以 ResNet50 残差网络和 SVM 分类器的混合模型在绝缘膜异常问题检测场景中的使用效果为例，在验证测试中，先以 1,000 张图片作为样本集，在 ResNet50 模型中进行模型微调得到基准值（97.85% 的准确率和 94% 的召回率），然后在此基础上分别进行参数调整，以及使用 ResNet50+SVM 的混合模型进行训练。验证结果如图 2-1-16 所示，在 ResNet50 模型中进行参数调整优化后，可将准确率提高至 99%，将召回率提高至 97.56%，而加入 SVM 分类器后，更是将准确率提升至 99.12%，将召回率提升至 99.16%，检测精度提升显著。[7]



	基准值：	ResNet50 参数调整优化：	ResNet50+SVM 分类器：
准确率	97.85%	99%	99.12%
召回率	94%	97.56%	99.16%
准确率趋势	97.85%	99%	99.12%
召回率趋势	94%	97.56%	99.16%

图 2-1-16　混合模型在准确率与召回率上的效果

英特尔软硬件工具

OpenVINO™ 工具套件

在智能制造系统中，AI 的部署与应用需要支持日趋多样和复杂的算法模型和网络，而要保持模型变换的准确性并非易事。同时，在基于机器视觉的智能制造方案中，也需要高性能的计算机视觉库工具予以支撑。由英特尔推出的 OpenVINO™ 工具套件，可帮助制造行业用户更有效地加速深度学习的推理以及部署，缩短开发时间。主要功能特征如下：

- 内置深度学习部署工具包（Deep Learning Deployment Toolkit，DLDT），包含模型优化器（Deep Learning Model Optimizer）和推理引擎（Deep Learning Inference Engine）两个核心深度学习组件；
- 对传统 OpenCV、OpenCL™ 图像处理库进行了指令集优化，并融合了英特尔® 优化视觉库（Intel® Photography Vision Library）以及英特尔® Media SDK；
- 采用通用的 API 接口，可在基于英特尔® 架构的处理器、英特尔® Movidius™ VPU、现场可编程门阵列（Field-Programmable Gate Array，FPGA）等设备运行；

通过引入 OpenVINO™ 工具套件，基于深度学习的 AI 应用可获得巨大的性能增益。一项数据表明，OpenVINO™ 工具套件

[7] 测试配置为：处理器：双路英特尔® 至强® 6240 处理器，2.60GHz，18 核心 36 线程，超线程技术开启，睿频技术开启；内存：256GB DDR4；存储：英特尔® 固态盘 SC2BB480G7；操作系统：CentOS Linux release 7.7.1908 (Core)；编译器版本：GCC 7.3.0

可为深度学习推理带来多达 19 倍的性能提升[8]。下面简单介绍一些 OpenVINO™ 工具套件在 AI 解决方案中的应用实例：

■ **加速深度学习部署**

为了帮助用户加速深度学习部署，OpenVINO™ 工具套件提供了 DLDT，包括模型优化器和推理引擎两个核心组件以及一系列 DEMO 和优化工具。目前 OpenVINO™ 工具套件支持 Caffe、TensorFlow、MXNet 等主流深度学习框架，并预置了 SSD 检测模型，人脸识别检测等算法模型。

DLDT 的工作流程如图 2-1-17 所示。首先，系统需要为 AI 应用使用的深度学习框架（例如 Caffe 等）配置模型优化器，用于模型训练；其次，系统通过运行模型优化器，根据训练后的网络拓扑、权重和偏差值以及其他可选参数，生成模型的优化中间表示（Intermediate Representation，IR）文件，包括 bin（经训练的数据文件）和 xml（描述网络拓扑的文件）两种格式的文件；然后，通过 OpenVINO™ 工具套件提供的推理引擎来验证 AI 应用，并使用目标环境中的推理引擎以 IR 格式对模型进行测试；最后，在 AI 应用中集成推理引擎，进而在目标环境中部署模型。

■ **FP32/INT8 模型转换**

与 FP32 模型相比，INT8 模型具有更小的数值精度和动态范围，在保证信息损失最小化的前提下，将训练和推理的模型转化为 INT8 型，可以大幅降低计算压力，提升目标检测一类 AI 应用的效率。

OpenVINO™ 工具套件最新版本可以提供 FP32 到 INT8 模型转换功能，且支持基于第二代英特尔® 至强® 可扩展处理器所集成的英特尔® 深度学习加速。该技术对 INT8 有着良好的支持，可大幅加快低精度数值的运算速度。

OpenVINO™ 工具套件能将训练好的开放神经网络交换（Open Neural Network Exchange，ONNX）格式的模型进行转换和优化，生成 FP32 格式的 IR 文件；而后通过校准工具将 FP32 格式的文件转换为 INT8 格式的 IR 文件。为了保证信息损失最小化，上述的转换过程中会使用一个小批量的验证数据集，并且会将转换量化过程中的统计数据存储下来，以便确保后续的推理精度不受影响。

更多 OpenVINO™ 工具套件信息，可参阅本手册技术篇相关内容，或访问 https://software.intel.com/zh-cn/openvino-toolkit

基于英特尔® 技术支持的机器视觉全流程方案

如图 2-1-18 所示，基于机器视觉构建的工业辅助检测方案可以分为需求分析、终端数据采集部署、大数据管理、模型训练、推理部署以及测试 / 评估 / 迭代等主要流程。英特尔通过性能高效的硬件、灵活的产品形态，以及贴近用户场景的技术合作模式，为基于机器视觉的工业辅助检测方案全流程的构建、优化和落地提供卓而有效的支持。

在终端数据采集部署阶段，通过与英特尔开展的技术合作，用户可以采用分布式的数据预处理，优化特征提取模型，使影像读取时间缩短数倍；在模型训练过程中，由英特尔推出的 Analytics Zoo 平台提供了端到端、分布式的模型训练系统，可有效缩短用户部署时间；而在推理部署阶段，基于英特尔® 架构的 Movidius™ 视觉处理器以及 FPGA 产品都能为模型带来强劲的算力支持。尤其是最新的第二代英特尔® 至强® 可扩展处理器，其集成的英特尔® 深度学习加速技术，对 INT8 有着更好的支持，能大幅提升方案的模型推理速度，使预测时间从原先秒级缩短至毫秒级。

图 2-1-17　OpenVINO™ 工具套件加速深度学习部署流程

[8] 数据援引自英特尔官网：
https://software.intel.com/zh-cn/articles/a-guide-for-setting-up-docker-based-openvino-development-environment-with-ubuntu-system

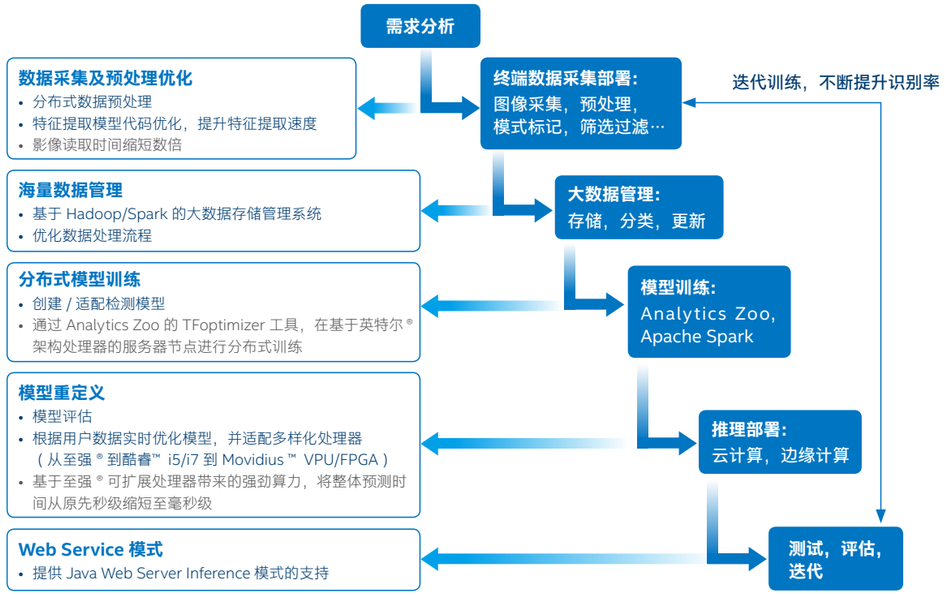


图 2-1-18　基于英特尔® 技术支持的机器视觉全流程

软硬件建议配置

以上基于机器视觉的智能制造解决方案的构建，可以参考如下基于英特尔® 架构的平台完成，环境配置如下：

名称	规格
处理器	双路英特尔® 至强® 6240 处理器或更高
基础频率	2.60GHz
核心/线程	18/36
HT	On
Turbo	On
内存	DDR4/192G（12 * 16GB 2666 MT/s）
硬盘	Intel SSDSC2BB480G7
BIOS	SE5C620.86B.02.01.0009.092820190230

名称	规格
操作系统	CentOS Linux release 7.6.1810
Linux内核	3.10.0-957.el7.x86_64
工作负载	CNN Classification/Object Detection
编译器	GCC 4.8.5 or GCC7 Higher
框架	PyTorch/Tensorflow

小结

随着面向图像处理和目标检测的深度学习、机器学习方法越来越成熟，利用机器视觉方法构建高效的智能制造辅助检测系统，正逐渐成为传统制造行业智能化转型的重要契机。但与大学、实验室以及科研机构中的 AI 研究不同的是，在制造业、金融、医疗等行业生产环境中落地的 AI 应用所面临的环境更为复杂，对实时性、精度要求也更高。为此，企业需要对 AI 算法模型的选择、训练方法的运用，以及基础设施架构的构建做出更多评估和优化，才能得到事半功倍的良好解决方案。

为此，英特尔与众多合作伙伴一起，充分评估一线生产场景的实际状况，一方面为不同场景选择合理的算法，满足检测系统在速度和精度上的需要；另一方面，也推动边缘计算等创新基础设施架构在 AI 应用中的运用，并取得了良好的成效。未来，英特尔还计划与更多合作伙伴一起，推动更多先进的软硬件产品与 AI 技术相融合，针对机器视觉在智能制造中的应用，推出更多高性价比的解决方案。

基于时间序列开展智能预测，助力企业排障增效，提升运营效能

基于时间序列的智能预测

时间序列预测方法

随着人工智能、大数据、工业物联网等技术的蓬勃发展，基于AI技术的智能预测方法正在传统制造、能源、金融、医疗等行业中发挥越来越重要的作用。所谓预测，是根据历史及当前状况，以一定逻辑对事物的未来发展趋势进行预想和判断。合理的预测方法，能有效帮助企业对生产经营资源进行调配和优化。

例如，市场部门可根据销售数据和市场变化实施预测，对销售方案做出优化；生产部门可根据设备日志来预测流水线的产能和故障情况，进而调整未来一段时间内的生产或检修计划。可见，构建良好的预测模型，能切实帮助企业优化产能、实现降本增效。

构建预测方法的重要方法之一，是根据预测目标的当前和历史数据，来推断其未来变化和趋势。因此，数据的时序性（Time Series）特性在方案中尤为重要。时序数据是指按时间顺序记录的数据列，其数据的表征口径必须一致。典型的时序数据包括服务器日志、生产流水、销售记录等。如下表1所示，在时序数据中，通常会有一列专门的数据类型timestamp对时间戳进行标识。

timestamp	server	city	cpu	memory
2019-07-26T09:55:01Z	server1	Guangzhou	0.1	0.2
2019-07-26T16:56:11Z	server2	Lanzhou	0.2	0.1
2019-07-27T11:56:11Z	server1	Guangzhou	0.2	0.2
2019-07-28T18:56:11Z	Server3	Nanjing	0.2	0.3

表1 某数据中心服务器日志

基于时序数据的预测方法，传统上更多是基于经验和规则的判断，准确率和时效性不高。现在通过AI方法，例如回归、聚类、决策树等模型的引入，企业得以在海量时序数据中厘清头绪，并结合科学的逻辑推理算法来做出预测，使其效率和准确率大为提升。目前，常见的时间序列预测算法有自回归滑动平均模型（Autoregressive Moving Average Model，ARMA）、差分整合移动平均自回归模型（Autoregressive Integrated Moving Average model，ARIMA）、递归神经网络（Recurrent Neural Networks，RNN）、卷积神经网络（Convolutional Neural Networks，CNN）、长短期记忆（Long Short-Term Memory，LSTM）、门控循环单元（Gated Recurrent Unit，GRU）以及XGBoost等。

■ 基于时间序列的AI预测解决方案

如图2-2-1所示，基于时间序列的AI预测解决方案目前正广泛地用于产量预测、供应链预测、销量预测、智能运维、电力负载预测等多个领域。其中，用于设备生产效率提升的产能预测、用于设备故障检测的维护性预测及保障企业IT设施健康运行的智能运维，是目前备受瞩目的几个应用方向。

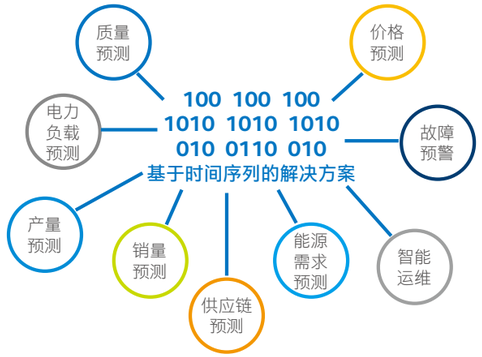


图2-2-1 基于时间序列的预测解决方案应用方向

在企业向智能制造转型的过程中，迫切需要生产线和设备能更灵活地应对市场个性化、差异化和定制化的需求。由于设备的产能预测可以直接影响一个企业的供应链管理和生产排程，因此传统制造、能源等企业亟待利用产能预测方法，对每台设备的生产能力和健康状况做出精准、全面和细颗粒度的预估。

利用智能化的维护性预测（Predictive Maintenance，PdM）方法来开展故障预警和设备检测，企业可大大减少由机器故障带来的高昂成本，延长设备的工作时间和使用寿命，降低保养和维护成本，提升产量。目前，维护性预测正越来越受到企业关注，如图2-2-2的研究数据表明，至2022年，全球维护性预测市场规模可达109亿美元[9]。

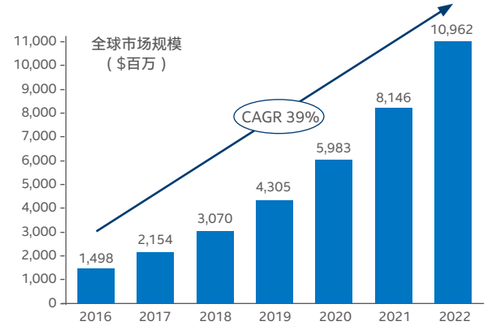


图2-2-2 全球维护性预测市场规模预测

[9] 数据援引自IoT Analytics相关报告：https://iot-analytics.com/report-us11-billion-predictive-maintenance-market-by-2022/

智能运维（AI Operations，AIOps）是通过 AI 方式，对企业 IT 设施的运行状态，例如处理器与内存使用率、磁盘吞吐量等进行监控、检测与维护，并及时做出反应。传统的运维方法一般是依靠经验以及专家规则等来制定自动化策略。但随着制造、能源等企业的业务与 IT 设施的绑定日趋紧密，企业对 IT 运维的实时性、可靠性以及预测前瞻性等方面都提出了更高的要求。

■ 现有预测方法的不足

以上预测方案多数是由设备生产厂商提供，逻辑也大多是基于规则或决策树制定的专家系统。随着制造、能源等传统企业进一步向数字化、智能化转型，传统预测方案也暴露出以下的局限性：

- **扩展性不足**：即便是同一型号的设备，不同企业的使用情况也不同，面对新问题、新需求时，传统的专家系统逻辑就会显得呆板和不适应。如果基于设备自身设定来做故障预判或产能预测策略，势必会造成停产时间加长、产能浪费、设备利用不充分等问题。

- **缺乏快捷的集成方法**：在工业制造领域，可以运用于时间序列预测方案的机器学习和深度学习算法越来越多，包括随机森林（Random Forest，RF）、LSTM、XGBoost 以及许多改造优化过的算法。这些日新月异的算法和模型需要被快速集成、部署和应用，这对传统工业制造企业相对薄弱的 IT 资源而言，无疑是一个艰巨的任务。

- **覆盖性不足**：在工业制造领域涉及的时序数据中，相当一部分属于稀疏数据，即数据的宏观特征命中率非常低。例如在故障预测中，符合故障特征的样本可能仅占所有样本的百万分之一甚至千万分之一。在这种情况下，一些传统模型的训练和推理效率会较低，且容易过拟合。

为帮助企业有效应对这些问题，英特尔与众多合作伙伴一起，从生产应用的实际需求出发，基于创新的深度学习、机器学习框架以及先进的软硬件产品，助力企业构建全新的智能预测方案。

常见时间序列预测算法

选择合适的时间序列预测算法模型，需要考虑的因素包括：

- 时序数据的质量；
- 维度的多寡；
- 是否有周期性；
- 是否有相关性。

目前常见的算法包括：

- 基于统计分析的时间序列预测算法；
- 基于深度学习的时间序列预测算法；
- 基于机器学习的时间序列预测算法。

每种算法都有自己的特点，对承载平台的系统性能要求也各不相同。企业需要根据自身的实际情况进行横向对比，找出最适宜的算法模型。

■ 基于统计分析的时间序列预测算法

时间序列预测算法在工业制造领域并非新鲜事物。在上世纪，许多工矿、能源企业就已经着手利用传统的数学统计分析模型来预测产量、仓储状态，或者用于对齿轮、轴承等机械零件进行故障诊断和分析。其中较为有名的包括 ARMA、ARIMA 等。

ARMA 模型是传统基于统计分析的时间序列预测算法中，处理平稳时间序列的重要方法。如名称所示，其由自回归模型（Auto-regressive，AR）和移动平均模型（Moving-Average，MA）组成。它的基本原理是将预测指标和 timestamp 组成的数据序列作为一个随机序列。这一序列不仅体现了原始数据在时间上的延续性，也体现了影响因子随时间变动的规律。假设影响因子为：i_1，i_2，i_3 … i_n，噪声为 d，则预测对象的观测值 O 可以表示为：

$$O = A_0 + A_1 i_1 + A_2 i_2 + A_3 i_3 + \cdots + A_n i_n + d$$

而其随时间变动的表示式为：

$$O_t = A_0 + A_1 i_{t-1} + A_2 i_{t-2} + A_3 i_{t-3} + \cdots + A_n i_{t-n} + d_t$$

其中d_t可表示为：

$$d_t = b_0 + b_1 d_{t-1} + b_2 d_{t-2} + \cdots + b_m d_{t-m} + f_t$$

ARIMA 模型是在 ARMA 模型的基础上增加了差分阶数（Integrated, I）。由于 ARMA 模型要求时序数据呈正态分布，均值、方差和自协方差均要求为常数，因此当数据处于不稳定状态时，例如呈持续上升或持续下降的，就可以采用 ARIMA 模型。一般地，这一模型也会被写成 ARIMA（p, d, q），其中 p 代表自回归阶数、d 代表差分次数，而 q 则代表移动平均阶数。

从上述几种模型的特性可知，基于统计分析的时间序列预测算法一般用于处理线性数据。对于序列中的一些季节性规律和变化，或者有数据缺失等情况尚难以有效应对，用户只能手工处理。同时，每个序列的模型都要用户手工维护，灵活性不高，因此这类模型一般只用于目标种类较少、维度单一，且准确率要求不高的场景。

■ 基于机器学习的时间序列算法

机器学习方法也是时间序列预测解决方案中常见的算法，尤其是各类基于树的机器学习方法，常用的有 XGBoost、随机森林以及 SVM 等。

随机森林是以决策树为基础的机器学习算法。决策树的基本思想如图 2-2-3 所示，作为一个树形结构，决策树构建时会将数据划分为具有相似值的子集，其每一个非叶节点都是一个特征属性的测试，并由这些测试来产生许多分支，每个分支就是某个值域的输出子集。而最终的每个叶子，就是输出预测结果的数据。

随机森林用随机的方式构建一个相互不关联的决策树森林，当有一个新的样本进入随机森林，就让每一棵决策树都进行一次判断，计算样本的分类，然后看被哪一类选择最多，从而得到预测样本的归类。

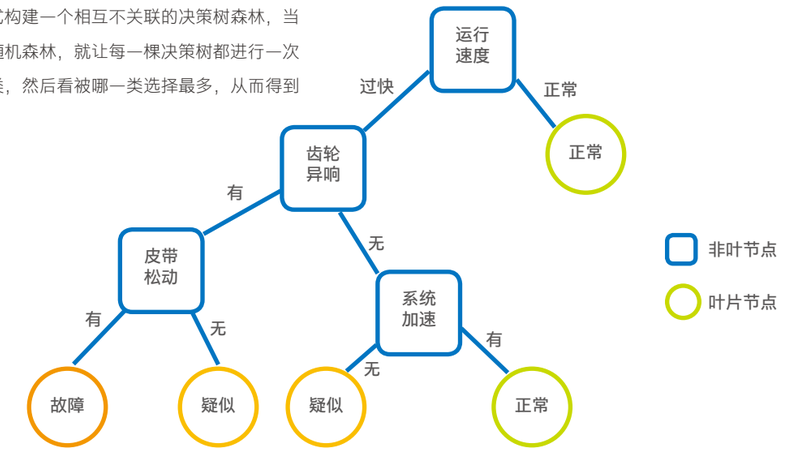


图 2-2-3　决策树示意图

XGBoost 是一种 boosting 的集成学习方法，是由大量分类回归树（Classification And Regression Tree，CART）集合而成的强分类器。其核心思想，就是通过不断进行特征分裂来生成新的分叉树，每添加一个树，其实就是学习一个新函数来拟合上次预测的残差。因此，XGBoost 目标函数可以定义为：

$$y_i = \sum_j q_j x_{ij}$$

当有k个样本时，其第n轮的模型预测结果为：

$$y_i^{(n)} = \sum_{k=1}^{n} f_k(x_i) = y_i^{(n-1)} + f_n(x_i)$$

在一些设备故障率较低的场景，大多数训练数据都是正常行为，正常：故障数据比例超过了 100~1,000 万：1，此时沿用深度学习方法可能造成训练时间长且效果不佳的问题。由于随机森林、XGBoost 等机器学习算法对高维数据以及非平衡数据有着良好的预测性能，因此在故障预警等可维护性预测领域中有着良好的应用。

■ 基于深度学习的时间序列算法

各类深度学习方法是目前时间序列预测方案中的常见算法，其中，RNN 又以其良好的时序亲和性而倍受关注。典型的 RNN 模型会对上一时刻的输入与当前状态进行结合，再进行输出。由于 RNN 模型结构是线性化时序关系，因此很容易造成长期

依赖问题（即预测结果与非常久远的时序输入相关，在工程上很难实现，造成长期记忆失效）。因此，还可以使用 RNN 的两种重要的衍生模型，即 LSTM、GRU 来避免这一问题。

如图 2-2-4 所示，LSTM 网络通过 3 个特别的 “门” 结构设计来大幅提升记忆时长。这一门结构是 Sigmod 函数和位乘法的结合体，其中 Sigmod 函数会输出一个 0 到 1 之间的值来描述当前可通过该结构的输入信息量，当输出为 1 时，可通过全部信息，反之则阻挡所有信息。GRU 网络是 LSTM 的优化版，它将 3 个 “门” 结构合并为 2 个，在计算方法上也有所不同。

LSTM 等深度学习网络对既包含长周期、又包含短周期的时序数据有着良好的训练和预测效果。例如在光伏发电场景中，用户需要对太阳能功率输出进行预测，数据中既包括了季节影响等长周期，也包括昼夜、云的移动等短周期，此时采用 LSTM、GRU 等算法模型，可以获得事半功倍的效果。与基于统计分析的时间序列预测方法相比，基于深度学习的时间序列预测方法优势如表 2 所示：

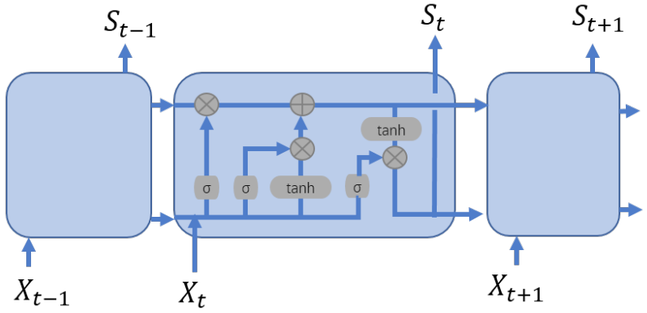


图 2-2-4　LSTM 网络结构

	传统时间序列（例如 ARIMA）	深度学习时间序列（例如 LSTM）
技术上的区别	• 线性模型 • 只能利用历史数据 • 只能手工发现一些季节性规律和变化 • 手工处理丢失数据 • 很难利用外部数据例如天气，温度等 • 对每个序列都要手工维护模型	• 可利用任何结构化的和非结构化的数据 • 可以自动学习季节性和其他规律趋势 • 自动处理丢失的数据和异常 • 可以轻松加入外加的信息和数据 • 可以一次性处理所有序列同时学习序列之间的关联性
适用场景	• 适合月，星期的序列 • 适合目标种类比较少的情况	• 可以处理任何颗粒度，例如每秒、每分钟、到每年 • 对处理种类，序列没有限制
训练时间	长	短
准确率	低	高

表 2　基于深度学习的时间序列预测方法优势

基于深度学习算法的预测流程

通常，如图 2-2-5 所示，时间序列预测解决方案包括数据准备、数据导入、数据预处理、模型定义、拟合与预测以及过拟合判断等步骤。

■ 数据准备

高质量的原始数据是实施时间序列预测的基础。在工业制造、能源生产等领域，数据采样率通常比较高，常见设备每秒采样在几次到十几次之间，数据量比较大，且数据时间跨度较长。在这种情况下，模型选择可以采用交叉验证（Cross Validation）的方式来对数据进行分组，训练组与测试组可以设一个比例，例如 60%-40%。

数据准备阶段的另一项重要工作是对数据质量进行校验。由于工业制造、能源生产等领域的工作环境往往比较恶劣，导致很多返回的传感数据是空白甚至是错误的，这些数据集会对最后的预测结果造成影响。如果数据缺失不多，可以用插值等方法进行弥补，如果缺失或错值过多，就必须放弃这组数据。实践中，常用的方法是从数据完整性、准确性等角度进行量化打分，到达一定分值后才执行下一步流程。

■ 数据导入

在数据导入阶段，需要考虑多维度、多变量数据的处理。当输入数据具有多维度、多变量特征，且这些多变量之间还有一定的依赖关系时，就需要对数据进行维度转换。最简单的方法是调用最主要的变量进行处理。

例如在设备保养中，发动机、水利发电机等都有大量参数，如温度、湿度、电压、电流等，用于监控设备的运行状况。在数据导入之前，需要把多维度转换成单维度进行处理。最简单的做法是选择一个与设备状况最为密切相关的参数，比如电流。

■ 数据预处理

在大多数情况下，时间序列都可以转换为监督学习。在转换过程中，会把不需要的数据剔除，把空缺的数据补充上，然后将数据序列进行归一化和标准化处理。

■ 模型定义

接下来需要选择模型参数。模型参数的选择将直接影响模型运行的性能，参数包括了迭代数量、迭代序列长度、隐含层节点数等，一般采取的方法是逐一找出最优参数。

■ 拟合与预测

拟合和预测过程是根据模型进行训练和推理的过程，需要根据场景需要，设定适当的损失函数、epochs 以及 batch_size。

■ 过拟合判断

所谓过拟合，是指 AI 学习时选择的模型包含参数过多，以致出现这一模型对已知数据预测很好，但对未知数据预测很差的现象。对是否过拟合，可以通过学习曲线来判断。

基于英特尔® 架构的代码示例

为帮助用户更好地基于英特尔® 架构开展时间序列 AI 预测方法的研究与探索，英特尔已根据不同的用户应用场景，在多种模型算法上给出了相应的优化代码示例，以下是使用 TensorFlow 框架的 LSTNet 代码示例：

```
import numpy as np
import keras
from keras.layers import Input, Dense, Conv1D, GRU, Dropout, Flatten, Activation, Masking
from keras.layers import concatenate, add, Lambda
from keras.models import Model, Sequential
from keras.optimizers import Adam
import keras.backend as K
import tensorflow as tf

class LSTNet(object):
    def __init__(self, args, dims):
        super(LSTNet, self).__init__()
        self.P = args.window
        self.m = dims
        self.hidR = args.hidRNN
        self.hidC = args.hidCNN
        self.hidS = args.hidSkip
        self.Ck = args.CNN_kernel
        self.skip = args.skip
        self.pt = int((self.P-self.Ck)/self.skip)
        self.hw = args.highway_window
        self.dropout = args.dropout
        self.output = args.output_fun
        self.lr = args.lr
        self.loss = args.loss
        self.clip = args.clip

    def _customized_loss(self, y_true, y_pred):
        return K.mean(K.square(y_true[:,0]-y_pred[:,0]),axis=-1)
```

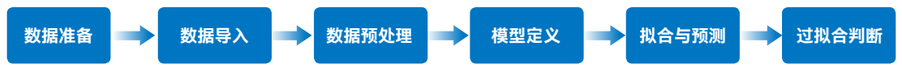


图 2-2-5　时间序列预测解决方案基本流程

```
        r = GRU(self.hidR)(c)
        r = Lambda(lambda k: K.reshape(k, (-1, self.hidR)))(r)
        r = Dropout(self.dropout)(r)

        # skip-RNN
        if self.skip > 0:
            s = Lambda(lambda k: k[:, int(-self.pt*self.skip):, :])(c)
            s = Lambda(lambda k: K.reshape(k, (-
1, self.pt, self.skip, self.hidC)))(s)
            s = Lambda(lambda k: K.permute_dimensions(k, (0,2,1,3)))(s)
            s = Lambda(lambda k: K.reshape(k, (-
1, self.pt, self.hidC)))(s)
            s = GRU(self.hidS)(s)
            s = Lambda(lambda k: K.reshape(k, (-
1, self.skip*self.hidS)))(s)
            s = Dropout(self.dropout)(s)
            r = concatenate([r,s])

        res = Dense(self.m)(r)

        # highway
        if self.hw > 0:
            z = Lambda(lambda k: k[:, -self.hw:, :])(x)
            z = Lambda(lambda k: K.permute_dimensions(k, (0,2,1)))(z)
            z = Lambda(lambda k: K.reshape(k, (-1, self.hw)))(z)
            z = Dense(1)(z)
            z = Lambda(lambda k: K.reshape(k, (-1, self.m)))(z)
            res = add([res, z])

        if self.output != 'no':
            res = Activation(self.output)(res)

        model = Model(inputs=x, outputs=res)
        model.compile(optimizer=Adam(lr=self.lr, clipnorm=self.clip), loss=s
elf._customized_loss)
        return model

class LSTNet_multi_inputs(object):
    def __init__(self, args, dims):
        super(LSTNet_multi_inputs, self).__init__()
        self.P = args.window
        self.m = dims
        self.hidR = args.hidRNN
        self.hidC = args.hidCNN
        self.hidS = args.hidSkip
        self.Ck = args.CNN_kernel
        self.skip = args.skip
        self.pt = args.ps
        self.hw = args.highway_window
        self.dropout = args.dropout
        self.output = args.output_fun
        self.lr = args.lr
        self.loss = args.loss
        self.clip = args.clip

    def _customized_loss(self, y_true, y_pred):
        return K.mean(K.square(y_true[:,0]-y_pred[:,0]),axis=-1)

    def make_model(self):
        input1 = Input(shape=(self.P, self.m))

        # CNN
        conv1 = Conv1D(self.hidC, self.Ck, strides=1, activation='relu') # f
or input1
        conv2 = Conv1D(self.hidC, self.Ck, strides=self.Ck, activation='relu
') # for input2
        conv2.set_weights(conv1.get_weights()) # at least use same weight
        c1 = conv1(input1)
        c1 = Dropout(self.dropout)(c1)

        # RNN
        r1 = GRU(self.hidR)(c1)
        r1 = Dropout(self.dropout)(r1)

        # Input2: long-term time series with period
        input2 = Input(shape=(self.pt*self.Ck, self.m))
```

```
        # CNN
        c2 = conv2(input2)
        c2 = Dropout(self.dropout)(c2)

        # RNN
        r2 = GRU(self.hidS)(c2)
        r2 = Dropout(self.dropout)(r2)

        r = concatenate([r1,r2])
        res = Dense(self.m)(r)

        # highway
        if self.hw > 0:
            z = Lambda(lambda k: k[:, -self.hw:, :])(input1)
            z = Lambda(lambda k: K.permute_dimensions(k, (0,2,1)))(z)
            z = Lambda(lambda k: K.reshape(k, (-1, self.hw)))(z)
            z = Dense(1)(z)
            z = Lambda(lambda k: K.reshape(k, (-1, self.m)))(z)
            res = add([res, z])

        if self.output != 'no':
            res = Activation(self.output)(res)

        model = Model(inputs=[input1, input2], outputs=res)
        model.compile(optimizer=Adam(lr=self.lr, clipnorm=self.clip), loss=
self._customized_loss)
        return model
```

为了进一步提升训练的效果，自定义了多项式的学习率下降规则，示例如下：

```
class PolynomialDecay():
    def __init__(self, maxEpochs=100, initAlpha=0.01, power=1.0):
        # store the maximum number of epochs, base learning rate,
        # and power of the polynomial
        self.maxEpochs = maxEpochs
        self.initAlpha = initAlpha
        self.power = power

    def __call__(self, epoch):
        # compute the new learning rate based on polynomial decay
        decay = (1 - (epoch / float(self.maxEpochs))) ** self.power
        alpha = self.initAlpha * decay

        # return the new learning rate
        return float(alpha)
```

根据不同的用户的目标，还可以自定义评价标准，让模型更加契合用户的评价指标：

```
def calcE(X):
    return 1 - 2 * np.sum(np.abs(X["y"]/(X["y"]+X["y_pred"]) -
0.5) * np.abs(X["y"]-X["y_pred"])/np.sum(np.abs(X["y"]-X["y_pred"])))
```

用时间序列预测未来的产能

Analytics Zoo 提供良好的时间序列预测方案集成环境

如前所述，随着 AI 算法研究的深入，更多机器学习和深度学习算法正被应用到工业制造领域的智能预测方案中，例如 LSTM、RNN、GRU、随机森林、梯度提升迭代决策树(Gradient Boosting Decision Tree，GBDT) 、XGBoost 等。

针对不同的使用场景，选择不同的算法模型、框架以及配套的软硬件基础设施，包括数据存储处理平台、计算平台等，都会对预测效率和准确率造成影响。因此在实际的方案部署中，企业需要根据需求的变化，选择、调整或优化不同的算法模型以及相应的系统架构和软硬件基础设施。

由英特尔推出的开源 Analytics Zoo “大数据分析 +AI” 平台，可以无缝地将 Apache Spark、TensorFlow、Keras 等软件与框架集成到一个统一的体系，方便扩展到大型 Apache Hadoop/Spark 集群，并结合 Analytics Zoo 集成的多个功能组件，用于时间序列预测解决方案所需的分布式训练或预测。同时，在 Analytics Zoo 新版本中，还集成了自动机器学习 (AutoML) ，可以进一步提升时序数据处理的效率。

现在，英特尔正与合作伙伴一起，基于 Analytics Zoo 构建用于风电、光伏场景的时间序列功率预测解决方案，以及用于设备维护的自动故障检测方案，并在实践中取得了良好的效果。

金风慧能以 AI 推动新能源预测实践

■ 项目背景

风电、光伏等清洁可再生能源日趋受到青睐，但风速、日照、气温等环境因素，会给发电设备的运行效率、设备安全等带来很大影响。因此，对风机等设备的输出功率开展预测，不仅有利于调度系统合理调整和优化发电计划，更能在减少火电占比，减少环境污染的同时，保证电网功率输出平稳。

传统上，电力企业会根据历史数据及生产经验来实施功率预测，但这一方法准确率低、波动性大。为解决这些问题，中国领先的新能源数字化、智能化专业服务提供商北京金风慧能技术有限公司 (以下简称 “金风慧能”) 正与英特尔一起，利用分布式深度学习方法，结合实时气象预报数据，构建全新的分布式深度学习功率预测算法平台 , 来提升功率预测算法的准确性、稳定性、适应性及开发效率。

新方案采用了基于时序数据的分布式深度学习预测算法。但是，新的算法模型在提升预测效率和准确性的同时，也给方案的构建以及后续的训练与预测等环节带来挑战。在英特尔的助力下，金风慧能引入基于大数据的 Analytics Zoo 分布式深度学习平台，完成了从数据预处理、特征工程搭建、模型训练等分布式框架的建设，并针对数据的时序性特性进行了大量优化。最后对分布在全国各重点区域的场站进行了测试，获得良好效果。

■ 基于时序数据的深度学习功率预测方案

智能功率预测一般可分为数据采集、存储、治理，训练、推理等流程。部署在风机、光伏面板、电网等处负责数据采集的边缘传感器，会把温度、功率、发电量等既有数据通过网络传送数据中心进行存储，而后在数据治理模块中进行有效数据的筛选，并使用选定模型对数据进行训练和推理，最终得到预测数据并用于生产中。

如图 2-2-6 可以看到，金风慧能的功率预测是一个端到端的数据处理管道。数据需要从部署在边缘的物联网 (Internet of Things，IoT) 设备采集，并在存储后通过数据治理等环节，再采用特定的模型完成训练，最终实现 AI 的应用落地。由于 IoT 设备的不稳定性，原始数据的质量通常无法保证，数据治理中数据标准的建立将是至关重要的环节。

图 2-2-6　AI 功率预测流程

金风慧能既有的功率预测方法是通过特定算法模型，对近期的环境参数、功率、发电量等数据样本进行训练和推理。虽然这一方法可对较近的时间点（15-30 分钟内）准确预测，但随着预测时间的增加而导致预测准确率会降低。对于电力生产所需的功率预测系统而言，最常见的超短期预测也需要系统能预测 4 小时内的功率输出，这意味着，预测系统需要在未来 16 个时间点（每 15 分钟计为一个时间点）上，都保持较高的预测准确率和稳定性。

新方案采用了面向时序数据的深度学习模型来构建功率预测方案。在方案构建之初，金风慧能首先尝试使用传统的 RNN（LSTM）这类经典的深度学习算法模型。但在实践中发现模型的预测效果并不尽如人意。究其原因，更多是来自于数据本身，即原有数据的缺失率非常高，且缺失数据的填充方法对于最终结果有直接影响。

因此，在英特尔研发团队的帮助下，金风慧能选择了在时序预测中更为准确的长短期时间序列网络（Long and Short term Time series Network，LSTNet）深度学习算法模型。如 2-2-7 所示，这一面向时序数据的算法模型融合了三种不同的网络拓扑，即用于短期非线性时序特征的 CNN、用于长期非线性时序特征的 RNN/LSTM/RNN-Skip 网络以及用于线性预测的 AR 网络。模型首先采用 CNN 中的空洞卷积抽取多特征维度的时序特征，将特征输入到 RNN 网络获得长、短期的时序特征，结合自回归等线性预测一同预测最后的功率值，满足了金风慧能对本项目的性能期望。

更多 LSTNet 算法详情，请参阅 Modeling Long- and Short-Term Temporal Patterns with Deep Neural Networks，Guokun Lai, Wei-Cheng Chang, Yiming Yang, Hanxiao Liu
https://arxiv.org/pdf/1703.07015.pdf

基于这一算法，金风慧能构建了全新的功率预测解决方案。如图 2-2-8 所示，首先，方案会将来自风机或光伏的数据导入开源 Kafka 流处理平台上，然后进行数据 ETL（Extract-Transform-Load）处理（包括数据预处理和流数据处理过程），接下来系统进行模型训练和预测，得到的权重、参数和预测结果会置入数据库，随着新数据的不断加入和历史数据的不断积累，模型可以实现满足不同周期的迭代优化，实现闭环的 AI 应用部署。

此外，由英特尔提供的 Analytics Zoo 帮助新方案构建了统一的端到端分布式架构，在提高系统开发部署效率和可扩展性的同时，也在时序数据分析方面拥有独特的功能和优势。

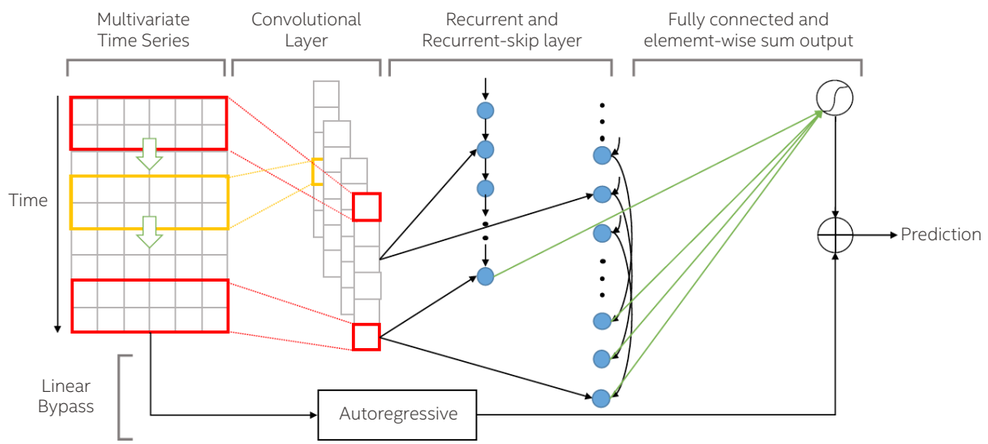


图 2-2-7　LSTNet 深度学习框架

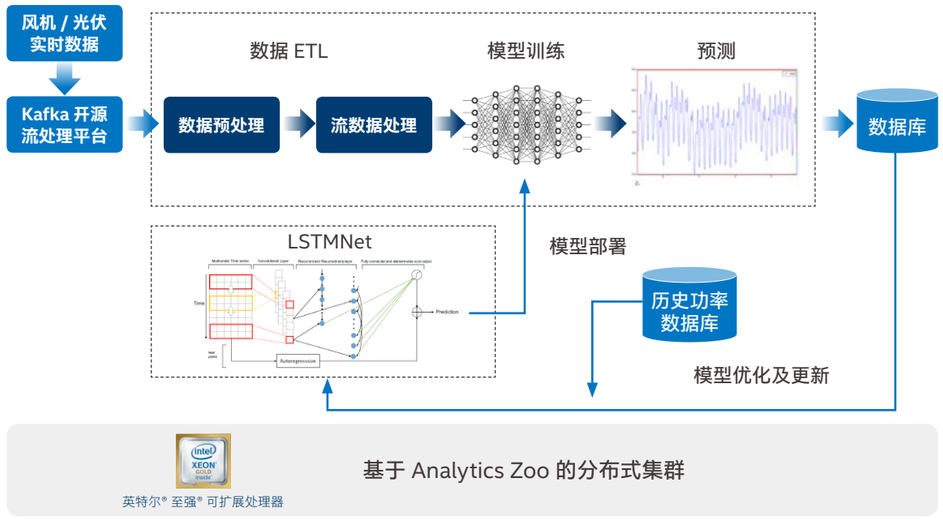


图 2-2-8　采用 LSTNet 模型的功率预测方案

如图 2-2-9 所示，Analytics Zoo 能够帮助金风慧能将新方案中的 Spark、TensorFlow、Keras 以及其它软件和框架无缝集成到同一管道中，有助于金风慧能将数据存储、数据处理以及训练推理的流水线整合到统一的基础设施，来大幅提升方案的部署效率、资源利用率和可扩展性，并减少硬件管理以及系统运维成本。

同时，Analytics Zoo 还能将英特尔提供的众多底层加速库，英特尔 ® MKL、英特尔 ® MKL-DNN 等应用到上层功率预测方案的优化中去，并可将 TensorFlow、Keras 模型透明地扩展到大数据集群，让用户能更方便地在训练或推理方案中采用分布式架构。

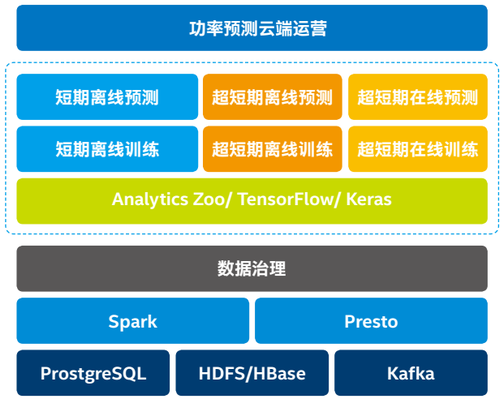


图 2-2-9　基于 Analytics Zoo 的分布式功率预测架构

针对时序数据分析，Analytics Zoo 预置了丰富的功能组件，包括：

- 功率预测常见的深度学习 / 机器学习模型：LSTNet、LSTM、Encoder-Decoder、MTNet、ARIMA 等等；
- 功率预测中常用的数据预处理和特征工程：Datetime features、Time diff、Log-transform、Rolling window 等等；
- 功率预测中普遍的异常探测方法：Percentile、Distribution-based、Uncertainty based、Autoencoder 等等。

通过这些组件，Analytics Zoo 能对不同时序分析应用，例如时序预测、异常检测、时序表征学习、时序聚类等，提供完整的解决方案。

除此之外，最新版本的 Analytics Zoo 针对于时间序列数据提供了 AutoML（自动机器学习）方法，使之能够进行自动化特征选择、模型选择和超参调优等，可以减少建模人员的工作量。随着更多的时序网络模型的加入，Analytics Zoo 还将继续提升处理时间序列数据的灵活性和效率，大幅度地减轻建模人员的工作压力，并能保证模型的推理与训练效果。

■ 方案成效

为验证基于时序数据的深度学习功率预测方案在实际场景中的运行表现，金风慧能与英特尔一起，在全国多个需要重点优化的光伏测试场站进行了测试。以月为周期，在每一个测试的光伏场中在单小时内，使用 30,000 条记录对 LSTNet 模型进行 5,000 次迭代优化，并在 50 毫秒内获得未来 2 小时的功率预测数据。如图 2-2-10 所示，在预测准确率上，新方案超越了原方案的 59%，达到了 79.41%，而在训练速度上，新方案的训练时间远低于原方案的 4 小时，仅为 1 小时 [10]。

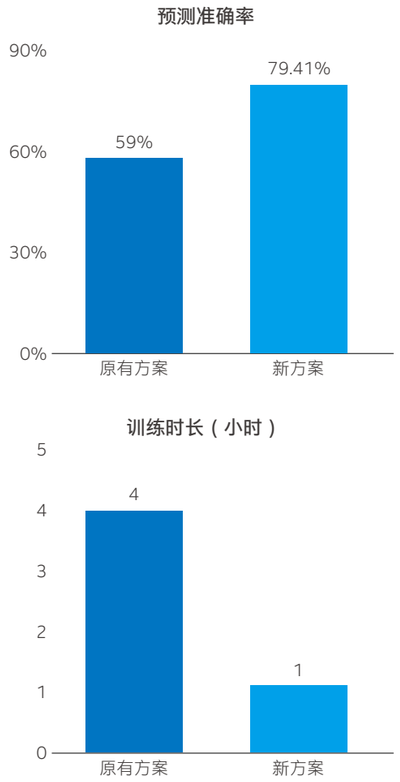


图 2-2-10　金风慧能功率预测新旧方案效能对比

一如上述测试取得的良好成效，通过双方的共同努力，新方案也在实践中取得了丰硕的成果。经初步估计，在一个 30 兆瓦的风电场中引入 AI，如果提升大约 20% 的预测率，可以间接减少上百吨的碳排放 [11]，从而为践行绿色环保的新能源理念提供了强有力的支撑。

面向时序数据的维护性预测

创新机器学习算法，提升维护性预测效能

时序数据是预测性维护方法中的重要数据基础，以往的维护性预测解决方案通常使用聚类分析、回归分析等经典数据挖掘方法来处理。近年来，基于机器学习算法模型的新方案逐渐成熟并受到用户的青睐。总体而言，基于机器学习方法的预测性维护主要通过两个分支方法来实现，即分类方法和回归方法。针对这两类方法，诸多经典机器学习算法，例如 LR、SVM、XGBoost 等，都被广泛地用于构建智能预测性维护方案。

对制造、能源等领域的企业而言，数据一般来自机器和传感器数据。这类数据的特征，一方面是存在较大不平衡性，尤其在一些精密仪表或者齿轮轴承等核心部件上，故障样本与非故障样本的比例往往达到几千万比一；另一方面是数据往往有着高维特征。在这种高维且稀疏的数据特征下，一些传统的机器学习方法可能面临训练推理时间长、准确率低等问题。

而企业对于齿轮轴承这些核心部件的故障预警，实时性要求非常高，甚至需要达到毫秒级。因此，预测模型需要在获得短短数组或数十组时序数据后，即得到预测结果，这对以往各类经典的机器学习算法模型无疑提出了极大的挑战。为此，英特尔与南京智谷人工智能研究院（以下简称 “智谷研究院”）一起，开展创新算法的研究，并在远景能源齿轮故障预测方案的实践中，获得了良好效果。

南京智谷人工智能研究院是依托计算机软件新技术国家重点实验室（南京大学），由南京大学人工智能学院人才团队与南京经济技术开发区管理委员会签约共建。南京大学周志华教授为研究院学术顾问。研究院着眼产业经济发展需求，面向人工智能领域前沿技术、共性技术和应用技术开展研发，充分发挥研究院及南京大学各方面的资源优势，实现科研技术的成果转化。

南京大学 LAMDA 团队隶属于计算机软件新技术国家重点实验室和南京大学计算机科学与技术系。团队在周志华教授的带领下，专注于机器学习、数据挖掘、模式识别等领域的研究与创新，成为国内乃至全球首屈一指的 AI 研究团队。

[10] 测试配置为：处理器：英特尔® 至强® 金牌 6130 处理器；内存：192GB DDR4 2666MHz；操作系统：CentOS 7.6；Spark版本：2.4.3。
[11] 数据来源：金风慧能的内部统计。

面向时序数据的创新算法模型

■ 深度森林模型

决策树是机器学习中常用的分类预测模型，但传统的决策树模型在学习容量上非常有限，一般仅能做到几层模型。在面对高维数据时，模型容量不足以承载足够的训练数据来进行表征学习。

吸纳了深度学习的多层处理结构思想后，来自南大的 LAMDA 团队提出了一种全新基于决策树的集成方法，被称为多粒度级联森林（Multi-Grained Cascade Forest，gcForest）。该模型设计了一种新的级联结构来进行表征学习。如图 2-2-11 所示，级联的每个层级包括了两个随机森林（黑色表示）和两个完全随机树木森林（蓝色表示）。假设训练中有三个要预测的类，每个森林都将输出三维类向量，并在联接后作为下一层级的原始输入。

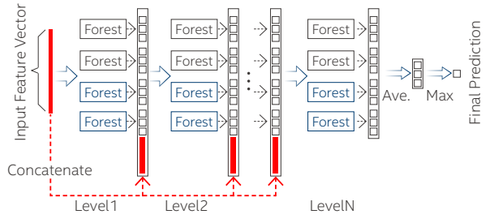


图 2-2-11　深度森林 gcForest 模型结构

与其他算法相比，由于级联层级能根据需要进行调节，这使得小规模数据集在gcForest模型中也有着不俗表现。以齿轮运行数据为例，仅需要少量时序数据（几组到十几组）就可以得到准确率较高的预测模型。而针对数据中的高维特性，该模型也可通过多粒度扫描（Multi-Grained Scanning）来进一步提升其表征学习能力。同时，gcForest所需的超参数更少，且具有很好的鲁棒性。

针对分类场景的对比测试结果如图2-2-12所示，在预测准确率上，gcForest模型有着不亚于深度神经网络、随机森林及逻辑回归等模型的表现，且预测效率更高。

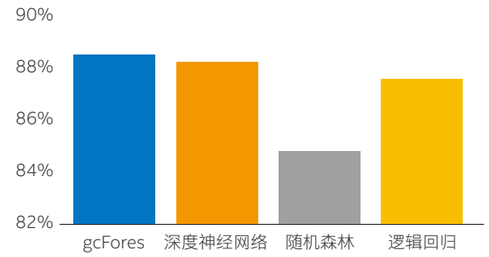


图 2-2-12　gcForest 模型与其他 AI 模型的对比测试

更多 gcForest 模型详情，
请参阅 Deep Forest: Towards An Alternative to Deep Neural Networks，Zhi-Hua Zhou, Ji Feng
https://arxiv.org/pdf/1702.08835v2

在 gcForest 模型的基础上，LAMDA 团队又进一步提出了基于决策树集成方法的自编码器（auto-encoder) eForest，以及可进行表征学习的多层梯度提升决策树（Multi-Layered Gradient Boosting Decision Trees，mGBDT）两种新的算法模型。

eForest 能使 gcForest 利用树决策路径所定义的最大相容规则（MCR）来重构原始模式，从而进一步提升预测准确率和预测速度。尤其是在齿轮运行日志一类的文本化数据处理上，用户仅使用 10%的输入比特，就能令模型以很高的精度重建原始数据 [12]。

而 mGBDT 则充分融合了树集成（Tree Ensembles）的优秀性能，以及分层分布式表示带来的表征学习能力，其使用梯度增强决策树作为每层的构建块，并可通过目标传播变体来共同优化训练过程。

[12] 数据引自J. Feng and Z.-H. Zhou. AutoEncoder by forest. In: Proceedings of the 32nd AAAI Conference on Artificial Intelligence (AAAI'18) , New Orleans, LA, 2018.

■ 基于英特尔® 架构的处理器为新模型提供强劲算力

深度森林主要是基于多层级的树架构来构建，其每一层均可视为一系列的决策树。因此，它需要并行在多棵树上执行任务，即更多地进行并行决策计算，而这正是拥有众核、高频能力的基于英特尔® 架构的处理器所擅长的。

在深度森林的并行任务处理机制下，只要增加处理器的内核数，即可同步带来训练效率的线性增长，这让英特尔擅长的多核架构大有用武之地。同时，基于英特尔® 架构的处理器更高的处理器时钟频率也大幅提升了训练和推理速度。第二代英特尔® 至强® 可扩展处理器不仅集成多达 56 个处理器内核，112 个线程，微架构也进行了全面升级优化，配备了更快、效率更高的高速缓存来提升处理效能，并可支持高达 36TB 的系统级内存容量。其配备的英特尔® AVX- 512，可提供更宽的矢量计算功能。由此可见，第二代英特尔® 至强® 可扩展处理器能为深度森林训练任务提供强劲的计算力，也能应对深度森林多任务并行处理的需求。

同时，第二代英特尔® 至强® 可扩展处理器对英特尔® 傲腾™ 持久内存有着良好的支持。深度森林需要在多棵决策树上并行执行任务，因此需要大量内存空间用于中间过程的迭代。传统动态随机存取存储器（Dynamic Random Access Memory，DRAM）不仅相对价格昂贵，单位内存容量也有限。通过引入第二代英特尔® 至强® 可扩展处理器与英特尔® 傲腾™ 持久内存的组合，可以很好满足这一需求，帮助 AI 平台轻松应对训练所需的 TB 级内存容量。

值得一提的是，采用 VNNI 的全新英特尔® 深度学习加速也令第二代英特尔® 至强® 可扩展处理器在推理速度上有着耀眼的表现，与上一代产品相比，性能提升高达 30 倍[13]，有力提升了方案的应用效率。

有了基于英特尔® 架构的处理器提供的强大计算力，以及优化的软件和编译器的助力，深度森林所蕴含的潜力可以被深入地探索和发掘，在一项利用该处理器开展的基于决策树的工作任务中，训练速度曾被提升了数百倍[14]。

创新算法助力远景能源构建风机齿轮故障预测方案

■ 项目背景

作为人们最为熟悉的清洁、可再生能源之一，风力发电正在能源格局中呈现更重要的角色。来自国家能源局的数据显示，2019 年上半年，全国风电发电量同比增长 11.5%[15]。作为全球领先的清洁科技企业，远景能源以智能风机、智慧风场、分布式风电等一系列先进风电设备引领着风电行业的高速发展。

风电机组是典型的旋转机械设备，齿轮是风机的核心设备，其故障是风机运行中最常见的问题。传统上，企业是根据 VDI3834 等故障检测标准，利用压电加速传感器一类设备感知齿轮故障，发现问题后由工作人员赶去现场检查维修，但此时故障往往已经对设备造成了危害。随着风电规模的日益扩大，人员检修的效率和成本显然成为风电企业进一步提升风场运营效率的瓶颈。为此，远景能源准备为用户提供更为智能的齿轮故障预测方案来解决这一问题。

利用振动信号来判别风机齿轮运行状态是目前常用的检测方法。齿轮箱在发生故障时，故障特征会带来时域信号的幅值以及其他数据的变化。利用机器学习算法可以对这类时序数据进行分析判断，进而有效判断齿轮箱的故障情况，进行及时处理，避免造成停机、甚至设备损坏带来重大经济损失。因此，新方案需要具有很高的实时性，要求算法模型必须具有 “快” 和 “准” 的特点。

■ 基于机器学习的齿轮故障预测方案

英特尔与南京智谷为远景能源设计的基于机器学习的齿轮故障预测方案主要分为两个核心环节：特征提取和分类模型。前者用于将原始的时序齿轮振动信号数据转化为训练样本，后者是通过一个创新的集成学习模型进行故障预测。

特征提取

特征提取主要分为以下三个步骤：

■ 普通特征提取

首先需要进行数据正规化和直方图特征化，即得到正规化的原始频域信号——正规化信号，以及直方图特征化后的数据——直方图信号。

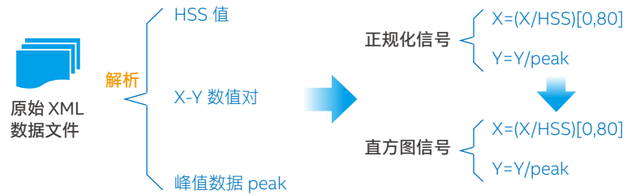


图 2-2-13　数据正规化和直方图特征化

如图 2-2-13 所示，系统首先从原始 XML 数据文件中解析出 HSS 值（轴承原生转速）、频域数据 X-Y 数值对以及峰值数据 peak。其中数据正规化是将 X 值除以 HSS 值，且只取 0-80 区间内的值，Y 值除以峰值数据 peak；而直方图特征化是指将数据正规化后的 X 轴进一步按 0.1 切分，得到 800 个区间，然后将每个区间内的 Y 值求和，由此得到一个长度为 800 的向量。

在获得正规化信号和直方图信号后进行特征设计。在方案中，一共需要设计 9 组特征，分别为：

特征组（1）：在正规化信号上，判断 peak 频率是否为某故障频率的倍数，或故障频率是否为 peak 频率的倍数，若是则为 1，否不是则为 0。此特征维度为 1；

特征组（2）：分别提取故障频率及其 2-10 倍频的 Y 值；以及故障频率及其 2-10 倍频左右一倍边频范围内的 Y 值，并计算标准差。若该通道内包含 f 个故障频率，则该特征维度为 20*f；

特征组（3）：在直方图信号上，计算信号的熵，即：

$$y_i = \hat{x_i}2/\left(\hat{x_1}2+\cdots+\hat{x_{800}}2\right)$$

$$\mathrm{H} = -\backslash\mathrm{sum\_i} = \hat{1}800 y_i * \log\left(y_i\right)$$

此特征维度为 1；

特征组（4）：然后在直方图信号上，进行 haar、db1、coif1、bior1.1 四种小波变换，分别使用变换后的上、下边带信号进行逆小波变换，由此得到 8 组重构信号，特征维度为 8*800=6400；

特征组（5）：对于特征组（4）中重构的信号，分别计算信号的熵，此特征维度为 8；

特征组（6）：在正规化信号上，计算 Y 值的均值和方差，此特征维度为 2；

特征组（7）：在正规化信号上，提取故障频率及其 2-3 倍频左右 0.5 倍边频范围内的 Y 值并计算熵值。若该通道包含 f 个故障频率，则该特征维度为 f；

特征组（8）：在直方图信号上，计算

$$y_i = \left|x_i - x\_\{i+1\}\right|$$

$$\mathrm{H\_diff} = -\backslash\mathrm{sum\_i} = \hat{1}800 y_i * \log\left(y_i\right)$$

此特征维度为1；

特征组（9）：在直方图信号上，计算

$$y_i = \left|x_i - 0.5 * x\_\{i+1\} - 0.5 * x\_\{i-1\}\right|$$

$$\mathrm{H\_diff} = -\backslash\mathrm{sum\_i} = \hat{1}800 y_i * \log\left(y_i\right)$$

此特征维度为1；

因此，在普通特征提取步骤中，一共可得到的特征维度为：14+21*f+6400。

■ 普通特征拼接

为了得到通道层级的特征向量，需要将属于相同通道的测点特征进行拼接，从而得到通道层级的特征向量。例如在某设备前端通道的单个测点上，采集的 XML 文件中包含 BPFO、BPFI、BSF、FTF 四个故障频率，那么该测点的数据提取出的样本特征维度即为 14+21*4+6400=6498，而在训练分类器时实际使用了其中的 98 维特征（重构信号的 6,400 维特征在训练分类器中一般不使用）。如图 2-2-14 所示，如已知设备前端通道包含了 4 个这样的测点，因此拼接后设备前端样本的特征维度为 98*4=392。

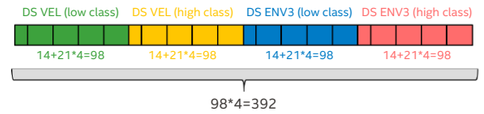


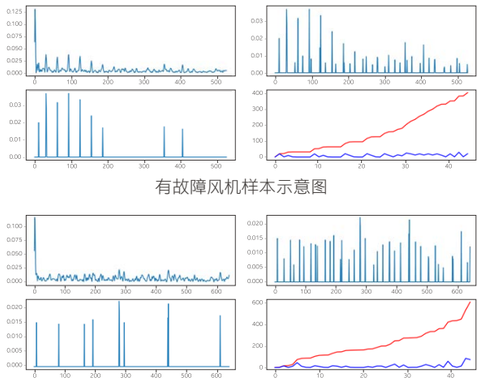
图 2-2-14　测点特征拼接

[13] 数据援引自英特尔官网：https://www.intel.com/content/www/us/en/technology-provider/products-and-solutions/xeon-scalable-family/moving-ai-to-the-edge-article.html
[14] 数据来自对LAMDA团队未公开的内部测试报告。
[15] 数据来自国家能源局数据统计报告：http://www.nea.gov.cn/2019-07/26/c_138259422.htm，http://www.nea.gov.cn/2019-08/23/c_138330885.htm

■ 高阶特征提取

最后，方案进行倍频 Zero Ratio（零比例）特征的高阶特征提取，其分为以下 6 个步骤：

1. 按比例遍历不同的窗口大小；
2. 对于可能的窗口取最大值，从而得到可能的峰值；
3. 取一定数量（例如 15 个）的峰值；
4. 统计这些峰值的索引值，并计算两两间的差值绘制曲线，如图 2-2-15 中的红色曲线；
5. 计算红色曲线的差值，得到图 2-2-15 中蓝色曲线，蓝色曲线的零值比例（红色曲线水平部分的比例）即 Zero Ratio；
6. 对于所有窗口的 Zero Ratio 取最大值、平均值和方差等统计量作为特征。



有故障风机样本示意图

无故障风机样本示意图

图 2-2-15　高阶特征提取示例

集成分类模型

风机齿轮故障预测，主要是采用分类方法来实现。为了提高学习系统的泛化能力。新方案采用了融合多种机器学习方法的创新集成算法模型，其中包括了 LR、RF 和 XGBoost 三种基础分类器。

如图 2-2-16 所示，齿轮故障集成分类模型首先将训练集通过 EasyEnsemble 算法得到 N 个采样集，并分别对这些采样集进行基础模型训练。尔后，使用 LR、RF 和 XGBoost 三种分类器得到各个采样集的预测结果。从图中可以看到，方案中采用了分层分类方法，其中第一层分类器使用高阶倍频特征，区分度很高，有故障模式的样本基本可以抓到，召回也率很高；第二层分类器使用高阶倍频特征以及其余的统计特征（例如峰值大小），可以排除第一层分类器误报的样本，进一步提高准确率。通过这种分层集成模型，可以有效地提升模型的预测质量。

■ 方案成效

通过在多个风场进行的部署验证，远景能源新的基于机器学习的齿轮故障预测方案充分展现了“快”和“准”的特点。在预测实时性方面，新方案中的模型在学习过大量故障案例后，在现场无须从零开始学习，通过小规模的样本集，例如 10~15 条时域信号即可快速发现齿轮故障；而在预测准确率上，在某风场的现场预测中，召回率和准确率均达到了 90% 以上 [16]。

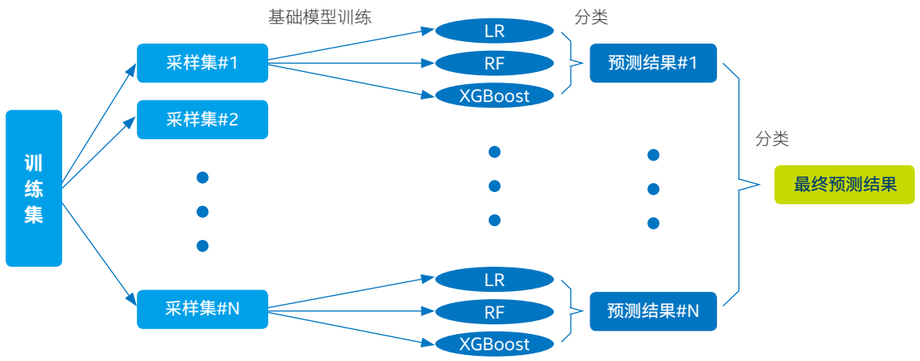


图 2-2-16　齿轮故障集成分类模型

[16] 数据所使用的测试配置：双路英特尔® 至强® 金牌 6148 处理器，内存：DDR4, 2666，256GB；硬盘：4*480GB 英特尔® 固态盘 S4510；操作系统：CentOS 7.3.1611；系统优化库：英特尔® Parallel Studio XE; 编程语言：Python

宝信以无监督的深度学习方法构建设备故障自动检测方案

■ 项目背景

作为国内领先的工业软件行业应用解决方案和服务提供商，上海宝信软件股份有限公司（以下简称“宝信软件”）一直致力于推动新一代信息技术与制造技术融合发展，引领中国工业化与信息化的深度融合，为工业制造、钢铁、能源等行业提供一流的 IT 技术服务。

设备异常检测是工业制造企业中最普遍，也是最重要的工作流程之一。传统上，企业通过定期停机检修维护、提前更换设备零部件等操作来避免设备出现异常。为了帮助企业提升生产效率，降低因检修停机或频繁更换部件带来的经济损失，宝信软件结合大数据以及 AI 技术，推出过多种设备振动信号异常检测解决方案。但这类解决方案通常以传感器信息作为数据源，其应用在真实场景中尚存在问题。一方面，传感器往往探测频率较高，数据量很大，大型企业每天接受到的数据量可能达到 TB 级；另一方面，数据类型也五花八门，导致企业难以为数据打“标签”。换句话说，在旧的解决方案中，并没有一个明确的目标变量，而是需要系统自己去纷繁复杂的数据中寻找答案。

因此，宝信软件在构建新的解决方案时面临两个挑战：一是如何结合工业制造企业的 IT 状况以及海量数据，建立良好便捷的方案集成环境；二是如何为方案选择合理的无监督方法。为此，宝信软件与英特尔开展深度技术合作，基于 Analytics Zoo，用无监督的深度学习方法构建全新的设备故障自动检测方案。

■ 方案与成效

如图 2-2-17 所示，新方案由“数据预处理”和“LSTM 检测模型”两个主要模块组成。以伺服电机的全寿命周期数据为例，来自设备传感器的数据进入数据预处理阶段，首先要进行特征提取，针对每秒数据提取均方根、峰度、峰值以及小波包分解得到的各频段能量值等，一共 11 个特征；然后进行特征预处理，包括小波去噪、标准化和滑动平均处理等过程。

由于设备数据都是无标签数据，为了保证检测精度，宝信软件为其设计了 LSTM 检测模型。模型由三个 LSTM 层和一个密集层组成，并采用每 50 个点训练下一个点的方式进行训练。整个模型通过 Analytics Zoo 中提供的 Keras API 来创建，并最终得到异常检测结果。通过在多个实际场景中的验证测试，如图 2-2-18 所示，模型最终成功预测了设备的异常点（橙色线为预测值，红点为异常情况），表明新方案检测效果良好。

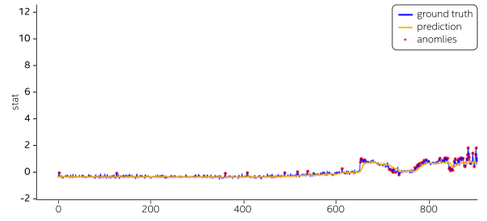


图 2-2-18　宝信软件预测效果

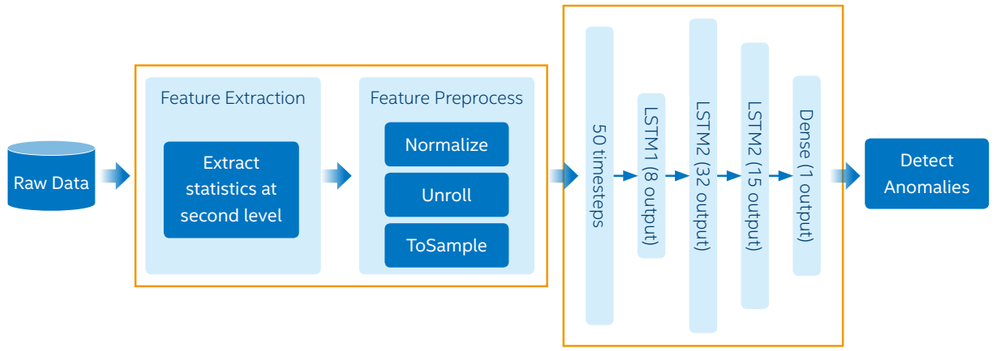


图 2-2-17　宝信软件设备故障自动检测方案架构

基于时序数据，推动智能运维发展

基于 IT 健康分析的智能运维新趋势

随着 IT 技术的发展与普及，各类信息化系统正日趋成为工业制造领域不可或缺的基础能力，推动着业务部门、工厂产线的高效运行，因此对信息化系统的高质量运维，是工业制造企业保持高效生产的关键。传统上，企业 IT 运维是通过系统指标的变化，例如处理器 / 内存使用率、磁盘吞吐量等，由人工判断系统是否存在问题与隐患。随着企业信息化系统日趋多样，加之软硬件平台不再紧密耦合，单一系统可能存在多个厂商的硬件与服务，因此信息化系统的复杂程度正呈指数化增长，给工业制造企业的 IT 部门带来了巨大的挑战。

虽然许多企业已经部署了自动化的运维监控系统，并基于专家规则对异常指标进行告警，但这种借助经验构建的系统，无法让运维人员通过告警信息即时进行原因分析。而通过专家后期采集、分析整套系统的数据，又往往需要很长时间。同时，由于技术能力的差异，专家对监控指标的分析结论也会有差别，为系统异常的及时甄别与处理埋下隐患。

在英特尔与南京基石数据技术有限责任公司（以下简称“基石数据”）看来，要根本解决这一问题，需要从运维生态入手，针对设备日志、运行数据等时序数据建模，建立“IT 健康分析”系统来发现存在的系统隐患，进而推送给运维部门或者专门的优化部门进行优化改进，并能够通过经验积累实现自我优化。

如图 2-2-19 所示，借助基于时序数据构建的“IT 健康分析”系统，企业一方面可以及时发现系统隐患，并通过常态优化或架构优化来予以修复；另一方面，模型也可对信息化系统的运维维保进行支撑，并通过运维经验和最佳实践的积累，来不断自我完善和优化。而要构建这样的系统，工业制造企业需要解决三个方面的问题。

- 引入高精度、高效率的智能分析模型作为系统核心，减少运维系统对人力，尤其是对专家的依赖；
- 为智能分析模型提供强有力的硬件基础设施，特别是高性能计算力的输入；
- 以合理的系统架构设计，保证充分的计算、分析能力能“下放”到运维一线。

基于以上分析，英特尔与基石数据首先引入 AI、大数据等技术，通过机器学习、深度学习方法来建立智能模型，对复杂的时序指标数据进行分析并判别系统的运行状态。基于海量数据训练出的智能分析模型，不仅有着更胜于专家系统的准确率和效率，更能大幅减少运维工作所需的人力，提高运维效率。一项统计数据表明，通过智能模型来预测系统状态变化，工作可以在秒级内完成，且分析准确率超过 98%[17]。

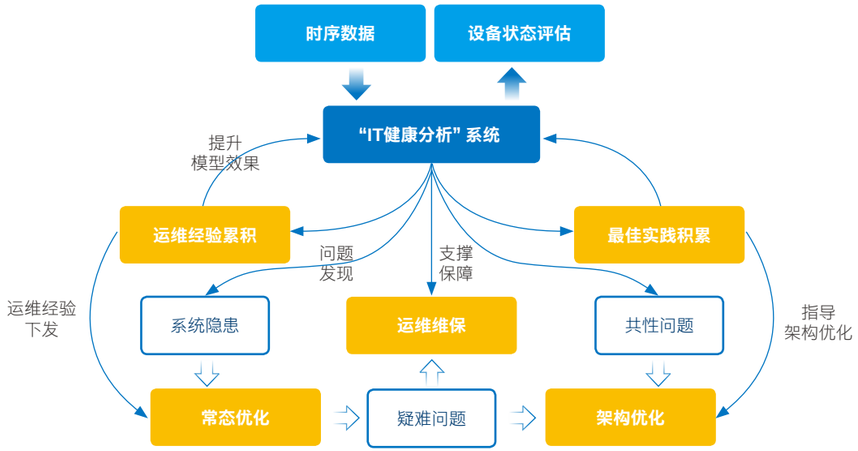


图 2-2-19 “IT 健康分析”模型架构

[17] 数据援引自南京基石数据技术有限责任公司《D-SMART运维知识自动化系统技术白皮书》：https://www.dfcdata.com.cn/index.html

为了使“IT 健康分析”系统和智能分析模型发挥更大效能，英特尔为之提供了多种先进软硬件产品与框架，为智能分析模型的训练推理过程提供强劲算力和工具。同时，新方案还引入了“云边协同”的新架构，一方面，通过就近部署智能分析模型，提升运维能力的实时性；另一方面，利用云端的专家知识库，对发现的问题进行闭环管理，展开问题溯源与优化方案编制，并将优化方案反馈回现场。

现在，这一全新的系统方案正广泛地在“数据库健康状态评估”、“网络安全风险预警”等实际场景中开展实践，并取得了良好的应用效果。

基石数据以机器健康模型，提升企业数据库运维效率

■ 项目背景

作为 IT 系统的核心组件之一，数据库的健康对于企业信息化系统的高效运行至关重要。传统上，运维工程师需要通过数据库管理系统（Database Management System，DBMS）等工具，以人工方式对数据库进行统一的管理、控制和调配。但这种方式既繁琐又缺乏效率，尤其当企业信息化系统变得更为复杂，且与业务紧密关联时，配置优化效果将直接影响企业生产的效率。以产线自动化监控系统为例，通过高清摄像头采集的产线图像需要在数据库中暂存后再送至后端处理，在这种高吞吐量的场景中，如何设置数据库的缓存机制，如何在阻塞发生前启动相关 Session 的处理等，都会直接影响该产线的生产效率和产品品质。

基石数据推出的机器健康模型能有效应对以上挑战。这一模型利用数据库丰富的时序化监控数据，例如连接状态、处理器 / 内存使用率、磁盘读写时延、缓存大小、等待时间等，通过机器学习或深度学习的方法进行训练，并得到合理的数据库健康预测得分，进而帮助运维人员制定相应策略。

同时，这一健康预测方法，也是英特尔与基石数据合作开展的“IT 健康分析”系统在数据库智能运维领域的重要落地，部署在边缘的数据库健康预测系统所得到的预测结果，可以与云端的 D-Smart 运维知识自动化系统形成交互，对方案实施迭代优化。

■ 基石数据机器健康模型方案描述

基石数据机器健康模型方案基本架构如图 2-2-20 所示，贴近电网管线、电力生产等一线部署的机器健康模型，由数据预处理、模型训练 & 验证以及预测系统几部分组成，可以使用训练数据，通过特定算法训练模型，并利用测试数据对模型效果进行验证，迭代优化模型。最终的预测结果将传送到位于云端的 D-Smart 运维知识库，并可以对接内外部专家系统、厂商支撑、系统优化团队以及专门的 IT 系统健康管理团队，根据预测结果对数据库状况进行分析，开展进一步优化。

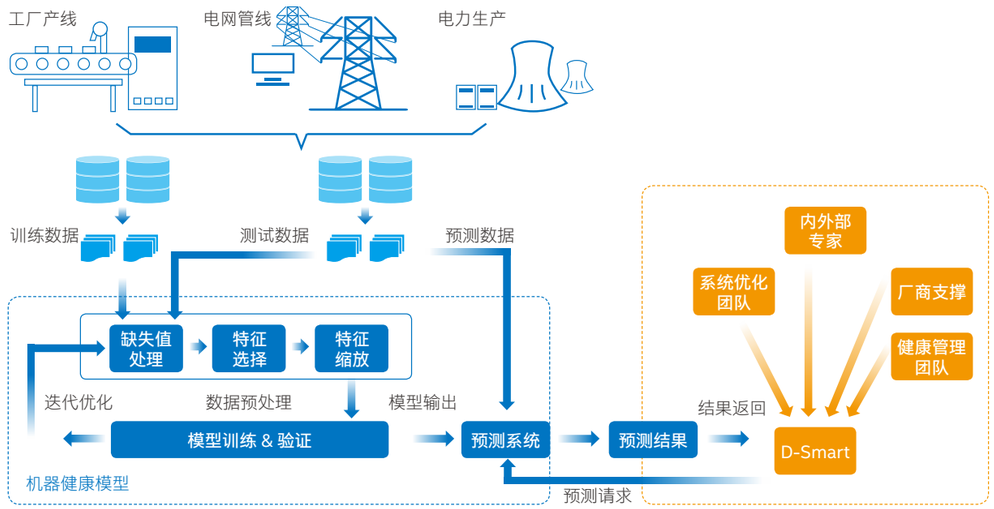


图 2-2-20 基石数据机器健康模型系统架构

机器健康模型会对数据库当前的健康状态进行评价打分，并预测未来一段时间内的健康得分。因此如图2-2-21所示，模型的输入数据X包括了会话连接状态、处理器/内存使用率、磁盘读写时延、缓存大小等具有时序特征的数据库监控数据，输出Y则是数据库的健康得分，包括当前分数和未来时间的预测分。分数为百分制，如96分。模型需要通过健康得分（标签）来调整优化模型的参数，因此模型采用的是监督学习的方法。

模型首先从一线数据中获得供训练和测试使用的数据集，这些数据已经预先打好标签，并按照 80%:20% 的训练与测试比例进行划分。如图2-2-22所示，方案中针对数据库运维的健康模型使用了7类68个维度的指标，并预先设定了各个指标的健康度得分。由此，系统可以得到一组以时间序列排列的数据库健康得分数据。

	指标项	指标名称	值
1	2180000	连接状态	连接成功
2	2180001	Oracle数据库日志分析结果	5
3	2180017	latch miss	215539
4	2180018	latch gets	3888253454
5	2180019	gc cr blocks received	195384
6	2180020	gc current blocks received	129479
7	2180021	SUM3 gc cr blocks served gc current blocks se···	280792
8	2180022	SUM4 gcs messages sent ges messages sent	1651606
9	2180023	global enqueue get time	43736
10	2180024	SUM6 global enqueue gets async global enque···	20713126
11	2180025	gc cr block receive time	2419
12	2180026	gc cr block flush time	214
13	2180027	gc current block flush time	207
14	2180028	msgs sent queue time	13499
15	2180029	msgs sent queued	812429
16	2180030	gcs msgs process time	9735

图 2-2-21　机器健康模型输入输出设计

在获取数据集之后，系统首先进行缺失值处理。数据集中的缺失值会带来噪声，从而对最后的预测结果造成偏差，因此方案采用了平均值填充或上下值填充的方式来予以处理。前者是将均值填入缺失值，后者是将前一个值或后一个值填入缺失值，不同缺失值填充方法会对预测结果造成差异，一般建议每行数据如果缺失率小于0.6则填充平均值。示例代码如下：

```
#平均值填充
fs.missing_stats.head()
fs.plot_missing()

#上下值填充
data.isnull().any().any()
data = data.fillna(method='ffill')
data = data.fillna(method='bfill')
data.isnull().any().any()
```

模型分项
数据库RAC
数据库并发执行
数据库总体状况
操作系统
数据库命中率
数据库负载
数据库IO

	指标	指标子项	总分
1	Global Cache Average CR Get Time		1
2	Global Cache Blocks Lost		1
3	Global Cache Average Current Get Tim		1
4	Global Cache Blocks Corrupted		1
5	Avg global enqueue get time		1
6	Avg GES message process time		1
7	Avg GCS message process time		1
8	Avg message sent queue time		1

	timestamp	score	2180200	2180204	2180205	21802								
0	2018-09-28 17:30:00	96.75	99.99	0.0167	0.00	0.								
1	2018-09-28 17:33:00	96.75	99.99	0.0333	0.01	0.								
2	2018-09-28 17:36:00	96.75	99.99	0.0333	0.00	0.								
3	2018-09-28 17:39:00	96.75	99.99	2.1333	0.00	0.								
4	2018-09-28 17:42:00	96.75	99.99	0.0333	0.00	0.00	0.00	0.13	0.4167	NaN	...	1.133	0.208	1.0	NaN
5	2018-09-28 17:45:00	96.75	99.99	0.1333	0.00	0.00	0.00	0.00	0.4000	NaN	...	1.566	0.208	1.0	NaN

图 2-2-22　获取训练数据集

特征选择是数据预处理环节中的重要步骤，进行合理的特征选择可以降低维度，查找和选择最有用的特征，提高模型的可解释性。另外，特征选择还能减少不必要的计算量，加快训练速度，同时降低模型方差，提高泛化效果。

在特征选择过程中，首先需要查找高度相关的特征，在机器学习方法中，这类特征可能会导致模型在测试集上的泛化能力下降。其次是计算特征的重要性。示例代码如下：

```
#查找高度相关的特征
collinear_features = fs.ops['collinear']
fs.record_collinear

#计算特征重要性
fs.plot_feature_importances(threshold = 0.99, plot_n = 20)
fs.identify_low_importance(cumulative_importance = 0.99)
train_final = fs.remove(method = 'all', keep_one_hot = False)
train_final.shape
```

在使用梯度下降一类的机器算法中，如果能保证不同特征的取值在相同或相近的范围内，比如都处于 0-1 之间，那么梯度下降算法会收敛的很快。因此在数据预处理的最后，方案对数据进行了特征缩放处理。

经过预处理的数据集需要选择合适的算法进行训练，方案根据数据库时序数据的特点选择多种算法进行了比较，包括支持向量回归（Support Vactor Regression，SVR）算法、RNN-LSTM 算法，GBDT 算法、XGBoost 算法以及随机森林算法等。通过验证比较表明，在时序化的数据库健康预测环境中，XGBoost 以及随机森林算法的预测准确度和效率较高。

在上述过程中，机器健康模型选择了英特尔® 至强® 可扩展处理器来为整个训练推理过程提供强劲算力。这一系列的处理器不仅集成了更多的内核和线程，对微架构也进行了全面升级优化，并配备了更快、效率更高的高速缓存来提升处理效能。同时，其集成的英特尔® DL Boost 技术，对 INT8 数值类型数据有着更好的支持，可大幅提升方案的模型推理速度。

■ 方案成效

通过在多个电力系统生产环境中的实际部署，验证了采用 XGBoost 或随机森林算法的机器健康模型可对数据库健康状况进行有效预测。如图 2-2-23 所示，上图是模型预测结果，下图是实际情况，两者的均方误差（Mean Squared Error，MSE）为 0.28，而在采用 XGBoost 算法的情况下，均方误差可进一步缩减到 0.2[18]。同时，得益于基于英特尔® 架构的处理器的强大算力，两种算法的训练时间均在数秒内，满足了工业制造企业预测实时性的要求。

利用机器健康模型，及以其为核心的 “IT 健康分析” 系统，基石数据针对某电力企业省级公司的 20 多套系统进行了 IT 健康分析巡检，仅在一个多月时间里，就发现问题 143 个，并全部完成溯源工作。同时，利用预测结果，用户还通过系统配置调整、SQL 调整、参数调整等方法，提升了系统性能，使一体化电量与线损系统、数据管理服务、结构化数据中心等与业务息息相关的核心信息化系统的健康分，由不足 80 分上升至 90 分以上，获得了从管理层到生产一线的一致好评。

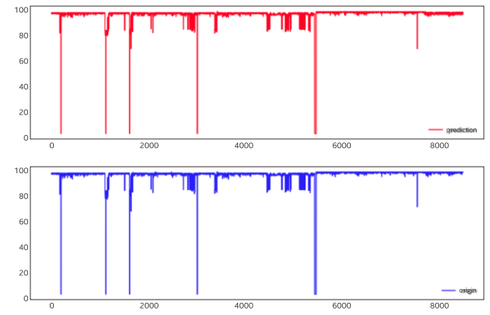


图 2-2-23　使用随机森林算法的机器健康模型预测结果

[18] 数据所使用验证测试配置：处理器：英特尔® 至强® 金牌6140处理器；内存：192GB DDR4 2666MHZ。

英特尔软硬件工具

集成 AutoML 框架的 Analytics Zoo

传统机器学习模型中的数据预处理、模型优化等工作，一般都需要富有经验的数据科学家来完成。为了进一步提高实施效率，近来年逐渐受到关注的自动机器学习（AutoML）框架，正尝试从特征工程、模型构建、超参优化等方面来实现机器学习的自动化流程。如前所述，由英特尔推出的开源 Analytics Zoo “大数据分析 +AI” 平台，在为时间序列预测解决方案提供统一便捷的集成环境之余，也在其新版中集成了基于开源 Ray 分布式框架的 AutoML 框架，使特征生成、模型选择和超参数调优等流程实现了自动化，可进一步提升时间序列预测效率。

AutoML 框架主要由 FeatureTransformer、Model、SearchEngine 和 Pipeline 等组件构成，其中：

- FeatureTransformer 定义了特征工程流程，包括了特征生成、特征缩放和特征选择等操作；
- Model 定义了模型以及所使用的优化算法；
- SearchEngine 用于搜索 FeatureTransformer 和 Model 的最佳超参数组合，是控制模型训练过程的核心；
- Pipeline 配置了 FeatureTransformer 与 Model 的最佳端到端数据分析流水线，可反复加载使用。

基于这些组件，AutoML 框架可面向时间序列数据处理等场景，为时序预测（TimeSequencePredictor）和时序管道（TimeSequencePipeline）等应用提供支持。

典型的时间序列预测 AutoML 执行流程如图 2-2-24 所示，首先通过参数调整的 FeatureTransformer 和 Model 会送入 SearchEngine 进行实例化，SearchEngine 随后会借助 Ray Tune 在右侧集群中进行多轮的试验（trail jobs），每轮试验都会使用不同的超参数组合进行特征工程以及模型训练。最后系统会选出最优的一组超参数和模型的组合（best model/parameters），然后送入 Pipeline 供后续的时间序列预测训练与推理所用。

进行时间序列预测训练时，首先要初始化一个 TimeSequence Predictor 对象，然后调用该对象的 fit 方法自动进行机器学习训练，并最终得到一个 TimeSequencePipeline 对象，参考代码如下：

```
from zoo.automl.regression.time_sequence_predictor import TimeSequencePredictor

tsp=TimeSequencePredictor( dt_col="datatime",
                           target_col="value",
                           extra_features_col=None,
                           feature_seq_len=1)
pipeline = tsp.fit(train_df,
                   metric="mean_squared_error",
                   recipe=RandomRecipe(num_samples=100),
                   distributed=True)
```

而获得最优的超参数和模型组合后，可将其送入 Pipeline 保存，以供后续训练和推理使用，参考代码如下：

```
pipeline.save("/tmp/saved_pipeline/my.ppl") #save

from zoo.automl.pipeline.time_sequence import load_ts_pipeline

pipeline = load_ts_pipeline("/tmp/saved_pipeline/my.ppl") #load
rs = pipeline.evaluate(test_df, metrics=["r_square"]) #evaluation
result_df = pipeline.predict(test_df) #inference
pipeline.fit(newtrain_df, epoch_num=5) #incremental training
```

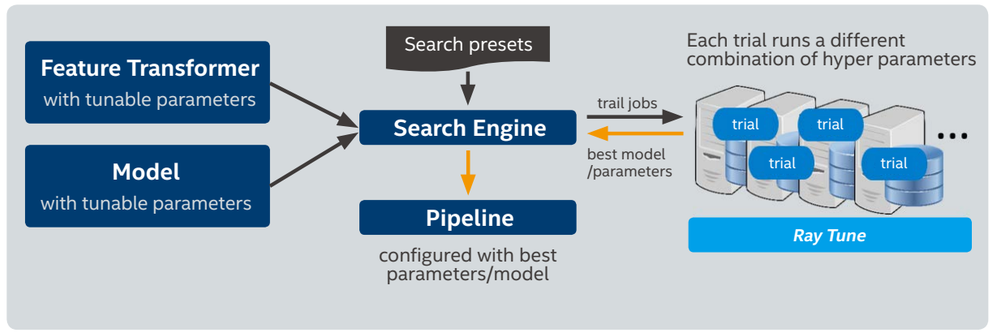


图 2-2-24　典型时序预测 AutoML 执行流程

更多的代码、Demo 和文档，请参阅：
在 Analytics Zoo Repo 中的 branch @ https://github.com/intel-analytics/analytics-zoo/tree/automl
AutoML 自述文档 @ https://github.com/intel-analytics/analytics-zoo/blob/automl/pyzoo/zoo/automl/README.md
Demo 手册 @ https://github.com/intel-analytics/analytics-zoo/blob/automl/apps/automl/nyc_taxi_dataset.ipynb

软硬件建议配置

以上基于时间序列的智能预测解决方案的构建，可以参考如下基于英特尔® 架构平台完成，环境配置如下：

名称	规格
处理器	双路英特尔® 至强® 6240 处理器或更高
基础频率	2.60GHz
核心/线程	18/36
HT	On
Turbo	On
内存	DDR4/192G (12 * 16GB 2666 MT/s)
硬盘	INTEL SSDSC2BB480G7
BIOS	SE5C620.86B.02.01.0009.092820190230

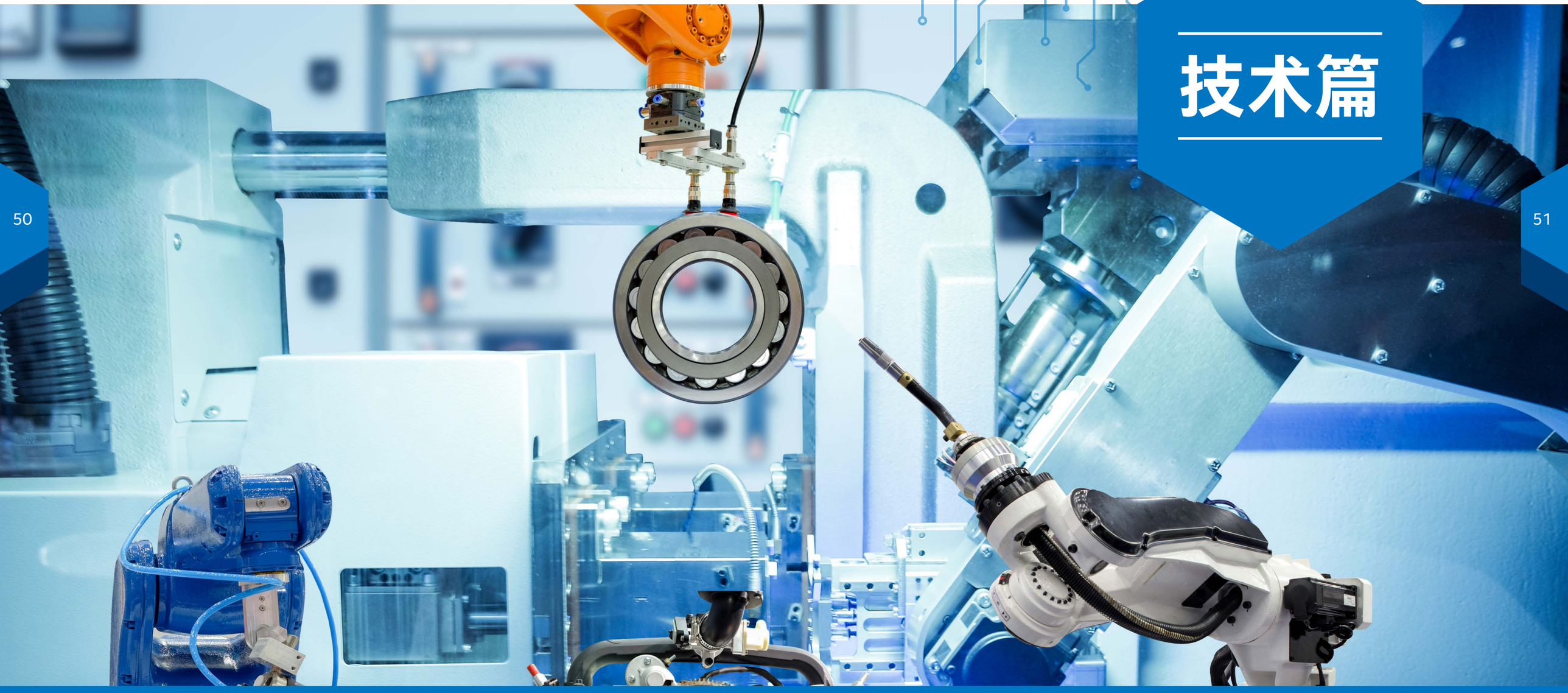
名称	规格
操作系统	CentOS Linux release 7.6.1810
Linux内核	3.10.0-957.el7.x86_64
工作负载	CNN Classification/Object Detection
编译器	GCC 4.8.5 or GCC7 Higher
框架	PyTorch/TensorFlow

小结

智能预测是传统制造、能源等领域的企业实施智能化转型的关键路径之一。通过智能预测方法，企业既可以对设备的功率、产能实施有效预测，进而有的放矢制定生产计划，合理实施资源调度；也可以对设备故障做到提前发现、提前排障，降低停机停产带来的经济损失；同时对 IT 设施开展的智能运维还能有效提升企业 IT 资源的利用效率，减少运维成本。

对时序数据的高效处理是构建各领域中智能预测方案的核心环节。得益于各类深度学习、机器学习算法模型的不断完善，以及由基于英特尔® 架构的处理器提供的强劲算力引擎，和 Analytics Zoo 等先进软硬件产品和框架带来的系统集成和加速能力，众多企业正构建起高效的基于时间序列的智能预测方案，并在一系列部署实践中取得了不俗表现。

未来，英特尔还计划与众多合作伙伴一起，根据产业的不同需求，以先进的软硬件产品激发出更多创新 AI 算法蕴含的潜力，为企业带去更具实时性、可用性和准确性的智能预测方案。



技术篇

第二代英特尔® 至强® 可扩展处理器

第二代英特尔® 至强® 可扩展处理器专为数据中心现代化变革而设计，提供比前代产品高出 25%-35% 的性能[1]，且具备多项新特性，提升了灵活性与安全性，增强了内存性能，能够帮助用户提高各种基础设施、企业应用及技术计算的运行效率，打造性能更强的敏捷服务和更具价值的功能，进而改善总体拥有成本（Total Cost of Ownership, TCO），提升生产力。

英特尔® 至强® 金牌处理器 6200 系列，特别是主流的英特尔® 至强® 金牌 6248 处理器、英特尔® 至强® 金牌 6240 处理器、英特尔® 至强® 金牌 6230 处理器，作为英特尔® 至强® 可扩展处理器平台的中流砥柱，能够支持更高的内存速度、增强的内存容量和四路可扩展性，并在性能、高级可靠性和硬件增强型安全技术方面取得了显著改进，且针对要求苛刻的主流数据中心、多云、网络和存储等工作负载进行了优化，能够适应更复杂、更多样化的应用场景。此外，英特尔® 至强® 金牌 6200 系列首次支持双 FMA 通道，意味着 FMA 性能提升了 2 倍[2]。

此外，第二代英特尔® 至强® 可扩展处理器集成深度学习加速技术（矢量神经网络指令 VNNI），扩展了英特尔® AVX-512，赋予平台更多、更强的 AI 能力，可加速人工智能和深度学习推理，并针对工作负载进行了优化。这使其拥有了集成 AI 加速能力的 CPU 架构。基于这一架构，大多数推理工作被集成在工作负载或应用程序中，让用户可以获得加速带来的性能和更高的灵活性等优势，在以数据为中心的时代，帮助在多云与智能边缘之间高效进行无障碍性能切换，以及 AI 开发与应用。

作为新一代至强® 可扩展平台的“核心”，第二代英特尔® 至强® 可扩展处理器支持英特尔® 傲腾™ 持久内存这一全新产品类别。而英特尔® 傲腾™ 持久内存通过与第二代英特尔® 至强® 金牌以及铂金处理器搭配，可以作为 DRAM 的有力补充，来显著提高系统性能，加速工作负载处理和服务交付。

功能特性：

- 更高的每核性能：多达 56 核（9200 系列）和多达 28 核（8200 系列），在计算、存储和网络应用中，为计算密集型工作负载提供更高的性能和可扩展性。
- 基于 VNNI 的英特尔® 深度学习加速（英特尔® DL Boost）技术：提高了在 CPU 上运行人工智能推理的表现，与上一代产品相比，性能提升高达 30 倍[3]，有助于从数据中心到边缘，充分支持 AI 部署和应用。
- 业界领先的内存和存储支持，更大的内存带宽 / 容量：支持英特尔® 傲腾™ 持久内存，与传统 DRAM 结合使用可支持高达 36TB 的系统级内存容量；内存带宽和容量提高 50%[4]，每路支持 6 个内存通道和多达 4TB DDR4 内存，速度高达 2933 MT/s（1 DPC）。还支持英特尔® 傲腾™ 固态盘和英特尔® QLC 3D NAND 固态盘，对于数据密集型的工作负载，突破性的内存和存储内存创新可以显著提高其效率和性能。
- 英特尔® Infrastructure Management 技术（英特尔® IMT）：该资源管理框架能够将英特尔的多种能力结合起来，有效支持平台级检测、报告和配置。
- 面向数据中心的英特尔® Security Libraries（英特尔® SecL-DC）：该软件库和组件实现了基于英特尔硬件的安全功能。

作为至强® 平台的创新之作，第二代英特尔® 至强® 可扩展处理器，基于突破的设计，从平台层面融合计算、内存、存储、网络以及安全等功能，并将它们提升到了新的高度。

[1] https://www.intel.cn/content/www/cn/zh/technology-provider/products-and-solutions/xeon-scalable-family/2gen-data-centric-computing-article.html
[2]、[3]、[4] https://www.intel.cn/content/www/cn/zh/products/docs/processors/xeon/2nd-gen-xeon-scalable-processors-brief.html

适用于人工智能应用的第二代英特尔® 至强® 可扩展处理器

* 仅在特定处理器上受支持。

	英特尔® 至强® 金牌处理器（6200 系列）	英特尔® 至强®金牌处理器（6200 系列）						英特尔® 至强® 铂金处理器（8200 系列）	英特尔® 至强® 铂金处理器（9200 系列）
		英特尔® 至强® 金牌 6230 处理器	英特尔® 至强® 金牌 6230R 处理器	英特尔® 至强® 金牌 6240 处理器	英特尔® 至强® 金牌 6240R 处理器	英特尔® 至强® 金牌 6248 处理器	英特尔® 至强® 金牌 6248R 处理器		
普适的性能和安全性									
支持的最大内核数	24 核	20 核	26 核	18 核	24 核	20 核	24 核	28 核	56 核
支持的最高频率	4.4 GHz	3.90 GHz	4.00 GHz	3.90 GHz	4.00 GHz	3.90 GHz	4.00 GHz	4.0 GHz	3.8 GHz
支持的 CPU 路数	多达 4 路	多达 4 路	多达 2 路	多达 4 路	多达 2 路	多达 4 路	多达 2 路	多达 8 路	多达 2 路
英特尔® 超级通道互连（UPI）	3	3	2	3	2	3	2	3	4
英特尔® UPI Speed	10.4 GT/s	10.4 GT/s	10.4 GT/s	10.4 GT/s	10.4 GT/s	10.4 GT/s	10.4 GT/s	10.4 GT/s	10.4 GT/s
英特尔® 高级矢量扩展 512（英特尔® AVX-512）	2 FMA	2 FMA	2 FMA	2 FMA	2 FMA	2 FMA	2 FMA	2 FMA	2 FMA
支持的最高内存速度（DDR4）	2933 MT/s	2933 MT/s	2933 MT/s	2933 MT/s	2933 MT/s	2933 MT/s	2933 MT/s	2933 MT/s	2933 MT/s
每路支持的最高内存容量*	1 TB，2 TB，4.5 TB	1 TB	1 TB	1 TB	1 TB	1 TB	1 TB	1 TB，2 TB，4.5 TB	3.0 TB
16 Gb DDR4 DIMM 支持	●	●	●	●	●	●	●	●	●
采用矢量神经网络指令（VNNI）的英特尔® 深度学习加速（英特尔® DL Boost）	●	●	●	●	●	●	●	●	●
英特尔® 傲腾™ 持久内存模块支持*	●	●	●	●	●	●	●	●	
英特尔® Omni-Path 架构（独立式PCIe* 卡）	●	●	●	●	●	●	●	●	●
英特尔® QuickAssist 技术（集成在芯片组中）	●	●	●	●	●	●	●	●	
英特尔® QuickAssist 技术（独立式PCIe* 卡）	●	●	●	●	●	●	●	●	●
英特尔® 傲腾™ 固态盘	●	●	●	●	●	●	●	●	●
英特尔® 固态盘数据中心家族（3D NAND）	●	●	●	●	●	●	●	●	●
PCIe 3.0	●	●	●	●	●	●	●	●	●
英特尔® QuickData技术（CBDMA）	●	●	●	●	●	●	●	●	●
非透明桥（NTB）	●	●	●	●	●	●	●	●	●
英特尔® 睿频加速技术 2.0	●	●	●	●	●	●	●	●	●
英特尔® 超线程技术（英特尔® HT 技术）	●	●	●	●	●	●	●	●	●
节点控制器支持	●	●	●	●	●	●	●	●	●

* 仅在特定处理器上受支持。

了解更多第二代英特尔® 至强® 可扩展处理器信息，请访问：
https://www.intel.cn/content/www/cn/zh/products/processors/xeon/scalable.html

英特尔® 傲腾™ 固态盘与基于英特尔® QLC 3D NAND 技术的英特尔® 固态盘

英特尔® 傲腾™ 固态盘和采用英特尔® QLC 3D NAND 技术的英特尔® 固态盘以创新的存储架构，助力数据中心面向未来，加速变革与跨越。

作为英特尔在固态盘产品线上的高端成员，英特尔® 傲腾™ 固态盘采用创新的 3D XPoint™ 存储介质，并结合了一系列的先进系统内存控制器、接口硬件和软件技术，在低延迟、高稳定等多方面均有上佳表现，可帮助消除数据中心存储瓶颈，并允许使用更大型、更经济实惠的数据集，进而加快应用程序速度、降低延迟敏感型工作负载的事务处理成本，并改善数据中心的 TCO。英特尔® 傲腾™ 固态盘这一更全面、更优秀，也更为均衡的 IT 基础设施能力，无疑能够为数据密集型的 AI 模型训练和推理带来更高的效率。以英特尔® 傲腾™ 固态盘 DC P4800X 为例，其具有高达 55 万 IOPS 的随机读写能力，低至 10 微秒的读写延迟，可更好地应对多用户、高并发场景，帮助应用方案获得更强的性能表现。同时，其写入寿命（Drive Writes Per Day, DWPD）高达 60，远超 NAND 固态盘，能够赋予存储系统更长的生命周期，也带来了更佳的经济性。

英特尔® 固态盘采用具有突破意义、可信的 3D NAND 技术来提升存储经济性，进而为替代传统硬盘（HDD）提供了性价比更高的选择，能够帮助用户改善体验、提升应用与服务的性能，并降低 TCO。作为固态盘中的“新兴生力军”，英特尔® 固态盘 D5-P4320 系列依托英特尔领先的 64 层 3D NAND 技术，可使得 QLC 固态盘单盘容量达到 7.68TB（TeraByte，万亿字节），从而有效应对数据中心等基础设施用户对“大容量”存储的需求。同时，其随机读取的 IOPS 高达 42.7 万，通过与第二代英特尔® 至强® 可扩展处理器搭配，尤其适用于 AI 训练等应用场景中对于“一写多读”的性能需求，为支持复杂的多样化工作负载提供了高效性、高稳定性和低能耗的存储框架。

了解更多信息，请访问：

- https://www.intel.cn/content/www/cn/zh/products/memory-storage/optane-memory/optane-memory-h10-solid-state-storage.html
- https://www.intel.cn/content/www/cn/zh/products/memory-storage/solid-state-drives/data-center-ssds.html

开源的统一的大数据分析+AI平台 Analytics Zoo

Analytics Zoo 是一个统一的大数据分析与 AI 开源平台，是为方便用户开发基于大数据、端到端的深度学习应用而推出。

Analytics Zoo 可帮助用户在 Apache Spark/Flink 和 Ray 之上，实现分布式的 TensorFlow、Keras、PyTorch 和 BigDL 程序，以及日后可能需要支持的其它框架等，并将其无缝集成到一个管道之中；且可将这些模型透明地扩展到成百上千节点规模的大数据集群，进行分布式训练或推理，从而进一步简化了 AI 解决方案开发，且无需额外的专用基础设施。

为了提高计算性能，Analytics Zoo 融合了多种软件库，如英特尔® MKL 和英特尔® MKL-DNN。在硬件方面，它基于英特尔® 至强® 处理器平台，充分释放第二代英特尔® 至强® 可扩展处理器已集成的向量和深度学习指令，可大幅提高训练和推理速度。

将数据存储和处理流水线整合到一个统一的基础设施中，而无需移动数据的好处显而易见——不仅可提高开发部署效率和可扩展性、减少硬件管理和开发者学习新语言的时间、提高开发部署效率、资源利用率和灵活性，且降低总拥有成本，还不会影响计算效率与性能。开发人员需要的只是在扩展其 AI 解决方案时，充分利用 Analytics Zoo 提供的丰富特性和功能，以及多种分析和 AI 工具，即可实现大数据分析和 AI 的高效融合与部署、应用。

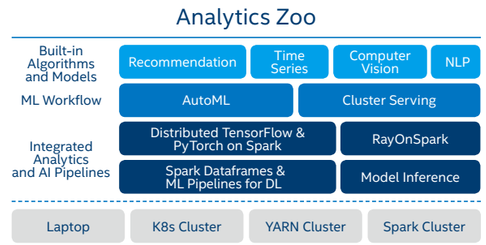
Analytics Zoo 平台所支持的众多 AI 框架中，BigDL 是英特尔自行研发和开源的。BigDL 是一个基于 Apache Spark 的分布式深度学习框架，可以无缝、直接运行在现有的 Apache Spark 和 Hadoop 集群之上，而不需要对集群做任何修改。基于 BigDL，开发者可以使用 Scala 或 Python 语言编写深度学习应用程序，并可以充分发挥 Spark 群集在可扩展方面的强大能力，推动大数据分析与 AI 的融合。在过去几年已经熟悉和启用 BigDL 的用户，可以通过 Analytics Zoo 直接调用 BigDL，无缝切换。

Analytics Zoo 技术特性：

- **端到到的流水线，应用 AI 模型（TensorFlow、PyTorch、OpenVINO™ 工具套件等）到分布式的大数据上**
  - 以 Spark 代码为 TensorFlow 或 PyTorch 实现分布式的训练和预测
  - 在 Spark ML 流水线中，支持原生的深度学习（TensorFlow / Keras / PyTorch / BigDL）
  - 通过 RayOnSpark，在大数据集群上直接运行 Ray 的程序
  - 提供 Plain Java/Python APIs（TensorFlow/PyTorch/BigDL/OpenVINO™）以服务于 Model Inference
- **高抽象的 ML 工作流实现机器学习任务的自动化**
  - Cluster Serving 实现自动化分布式（TensorFlow/PyTorch/Caffe/OpenVINO™）的模型推理
  - 可扩展的 AutoML 服务于时序数据分析的预测
- **内建的模型服务于推荐系统，时序分析，计算机视觉和自然语言处理应用**

使用 Analytics Zoo 的理由：

- 可以轻松地将 AI 模型（例如 TensorFlow、Keras、PyTorch、BigDL、OpenVINO™ 工具套件等）应用于分布式大数据上；
- 可以通过“零”代码更改将 AI 应用程序从一台笔记本电脑透明地扩展到大型集群；
- 可以将 AI 流水线部署到现有的 YARN 或 K8S 集群，而无需对集群进行任何修改；
- 可以使应用机器学习的过程自动化（例如特征工程，超参数调整，模型选择，分布式推理等）。



了解更多信息，请访问：
https://software.intel.com/zh-cn/blogs/2018/09/10/analytics-zoo-unifying-analytics-ai-for-apache-spark?elq_cid=4287274&erpm_id=7282583

英特尔® 数据分析加速库

作为人工智能的一个分支，机器学习现在正获得极大的关注，基于机器学习的高级分析也越来越流行，其原因在于，与其它分析方法相比，机器学习能够帮助 IT 人员、数据科学家、各种业务团队及其组织迅速释放优势。并且机器学习提供了许多新的商用和开源解决方案，为开发人员提供了一个丰富的生态系统。同时开发人员可以选择各种开源机器学习库，比如 Scikit-learn，Cloudera 和 Spark MLlib 等。

英特尔® 数据分析加速库（英特尔®DAAL）

英特尔为行业用户部署机器学习，也推出了一套高性能系统化方案，涵盖处理器、经优化的软件和开发人员支持，以及强大的生态系统等丰富资源。

机器学习需要强劲的计算能力。英特尔® 至强® 处理器提供了一个可扩展的基准，专门用于满足机器学习所特有的高度并行工作负载，及其对内存和架构（网络）的需求。在英特尔的一项测试中，该处理器使系统训练时间减少了 50 倍 [1]。

此外，英特尔还提供了软件支持，包括：

- 在英特尔® 至强® 处理器上优化的库、语言以及构件模块，oneDNN 和**英特尔® 数据分析加速库（Intel® Data Analytics Acceleration Library，英特尔® DAAL）**，以及面向英特尔® 架构优化的 Python 分发包。
- 可简化开发的优化框架，包括 Apache Spark、Caffe、Torch 和 TensorFlow。英特尔支持开源软件和商用软件，使用户能够迅速利用市场上可获得的最新处理器和系统功能。

英特尔® DAAL 是一套旨在帮助数据科学家和分析师们快速建立从数据预处理，到数据特征工程、数据建模和部署的端到端软件方案。它提供了建立机器学习和分析所需的各种数据分析及算法所需的高性能构建模块。目前已经支持线性回归、逻辑回归、LASSO、AdaBoost，贝叶斯分类器、支撑向量机、K 近邻、Kmeans 聚类、DBSCAN 聚类、各种决策树、随机森林、Gradient Boosting 等经典机器学习算法。这些算法经过高度优化，可在英特尔® 处理器上实现高性能，如中国一家领先的大数据分析技术和服务提供商，使用这些资源已将多个数据挖掘算法提高了 3 到 14 倍 [2]。

为了开发人员在基于英特尔环境中的机器学习应用中更加方便地使用英特尔® DAAL，英特尔开源了整个项目：https://github.com/intel/daal，并针对不同的大数据使用场景，提供全内存的、流式的和分布式的算法支持。比如 DAAL Kmeans 可以很好地和 Spark 结合，在 Spark 集群上进行多节点聚类。另外，英特尔® DAAL 提供了 C++、Java 和 Python 接口。

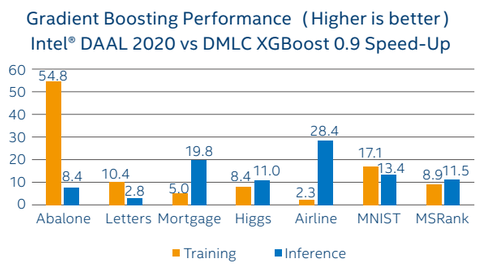
DAAL4py

为了更好地支持 Python 广泛应用最 Scikitlearn，英特尔® DAAL 提供了非常简单的 Python 接口 DAAL4py（开源地址：https://github.com/IntelPython/daal4py），它可以和 Scikitlearn 无缝的结合，在底层提供机器学习的算法加速。开发者无需修改 Scikitlearn 代码，即可利用自动向量化和多线程化的优势。目前 DAAL4py 在 Scikitlearn 中支持算法有：

- sklearn. 线性回归，sklearn. 岭回归，逻辑回归
- PCA
- KMeans
- pairwise_ 距离
- SVC（SVM 分类）

英特尔优化的 XGBoost

XGBoost 是一个基于递进 Gradient Boosting 的机器学习开源库，被广泛应用于各种分类和决策业务中。为了进一步加强其性能，英特尔优化和开源了代码库 [3]，最新的优化成果已经集成到 XGBoost 1.0 及之后的版本。相比 XGBoost 0.9 版，新版本性能提升 2 倍以上，最高达 54 倍。[4]



了解更多信息，请访问：https://software.intel.com/en-us/daal

[1、2] https://software.intel.com/zh-cn/articles/meritdata-speeds-up-a-big-data-platform
[3] Performance optimizations for Intel CPUs : https://github.com/dmlc/xgboost/pull/3957/files
[4] https://software.intel.com/daal

英特尔®深度神经网络库（oneDNN）

英特尔® 深度神经网络库（其前身是面向深度神经网络的英特尔® 数学核心函数库，Intel® MKL-DNN）是一款面向深度学习应用的开源性能增强库，也是英特尔为了帮助开发人员充分利用英特尔® 架构，推进深度学习的研究和应用而创建的基础库。（源代码地址：https://github.com/intel/mkl-dnn）

oneDNN 作为专为在英特尔® 架构上加快深度学习框架的运行速度而设计的一个性能增强库，包含了高度矢量化和线程化的构建模块，支持利用 C 和 C++ 接口实施深度神经网络，具备广泛的深度学习研究、开发和应用生态系统。目前已支持：TensorFlow、 PyTorch、MXNet、Caffe、Spark BigDL、OpenVINO ™ 工具套件等丰富的深度学习软件产品。

为了有效提升深度学习模型在英特尔® 架构基础设施上的运行速度，以及提升各类神经网络中其他性能敏感型应用的效率，oneDNN 提供了众多优化的深度学习运行和操作基元，可应用于不同的深度学习框架，以确保通用构建模块的高效实施。这些模块包括：

- 矩阵乘法和卷积：1D/2D/3D, Winograd 2D
- RNN 基元；
- 内积；
- 池化：最大、最小、平均；
- 标准化：跨通道局部响应归一化（LRN），批量归一化；
- 激活：修正线性单元（ReLU）；
- 数据操作：多维转置（转换）、拆分、合并、求和和缩放。

这些高效的函数模块可以应用于广泛的深度学习模型，如：

应用类型	拓扑结构
图像识别	AlexNet, VGG, GoogleNet, ResNet, MobileNet
图像分割	FCN, SegNet, MaskRCNN, U-Net
体积分割	3D-Unet
目标检测	SSD, Faster R-CNN, Yolo
机器翻译	GNMT
语音文字识别	DeepSpeech, WaveNet
对抗网络	DCGAN, 3DGAN
强化学习	A3C

为大幅提升了深度学习在 CPU 上的性能，英特尔还和众多开源社区合作，把该库集成进各种深度学习框架。如早在 2016 年，经过 oneDNN 优化的 Caffe，利用英特尔® 至强® 处理器 E5-2697 v3，相对于原始的 Caffe 性能获得高达 10 倍的提升 [1]。在 2019 年，经过优化后的 ResNet-50 也在英特尔® 至强® 铂金 9282 处理器上实现了每秒 7,736 张图像的领先性能 [2]。

oneDNN 目前已成为众多深度学习框架在 CPU 上运行时的基本配置。开发者可在深度学习框架的安装和应用中，直接获取 oneDNN 带来的性能提升。

了解更多信息，请访问：

- https://software.intel.com/zh-cn/articles/intel-mkl-dnn-part-1-library-overview-and-installation
- https://software.intel.com/zh-cn/articles/introducing-dnn-primitives-in-intelr-mkl

[1] https://software.intel.com/es-es/node/604830?language=en
[2] https://www.intel.com/content/dam/www/public/us/en/images/diagrams/rwd/xeon-scalable-max-inference-rwd.png

OpenVINO™ 工具套件

OpenVINO™ 工具套件是英特尔推出的一款加速深度学习推理及部署的软件工具套件，用以加快高性能计算机视觉处理和应用。该工具允许异构执行，支持 Windows 与 Linux 系统，以及 Python/C++ 语言，能够有效推进计算机视觉技术在从智能摄像头、视频监控、机器人，到智能交通、智能医疗等领域的深入应用。

本工具套件提高了计算机视觉解决方案的性能，缩短了开发时间，简化了从英特尔提供的丰富硬件选项中获得效益的途径，而这些选项可以提高性能、降低功耗并最大化硬件利用率——让用户可以低资源获得高收益，并为新的产品设计提供个性化空间。

通过基于深度卷积神经网络（CNN），扩展英特尔® 硬件（包括加速器）的工作负载，使得 OpenVINO™ 工具套件可依托英特尔® 架构处理器集成的显卡（Integrated GPU）、FPGA、英特尔® Movidius™ VPU 等芯片，来增强视觉系统的功能和性能。最新发布的 OpenVINO ™ 版本已能支持第二代英特尔® 至强® 可扩展处理器，并通过英特尔® AVX-512 以及采用 VNNI 的英特尔® DL Boost 技术来提升推理性能，可帮助客户在不改变软件的基础上，快速完成硬件产品升级和算法移植，从而助其在边缘侧快速实现高性能计算机视觉与深度学习应用的开发：

- 在英特尔® 架构平台上提升计算机视觉相关深度学习性能达 19 倍以上[1]；
- 释放 CNN-based 的网络在边缘设备的性能瓶颈；
- 对 OpenCV、OpenXV 视觉库的传统 API 实现加速与优化；
- 基于通用 API 接口在 CPU、GPU、FPGA 等设备上运行；
- 基于英特尔® 平台优化的 OpenVINO™ 工具套件主要包括深度学习部署工具包和传统的计算机视觉工具包两大部分。深度学习部署工具包括模型优化器（Model Optimizer）和推理引擎（Inference Engine）两个核心组件；
- 模型优化器：将给定的模型转化为标准的中间表示（Intermediate Representation，IR），并对模型优化，支持 ONNX、TensorFlow、Caffe、MXNet、Kaldi 等深度学习框架；
- 推理引擎：支持硬件指令集层面的深度学习模型加速运行，支持的硬件设备：CPU、GPU、FPGA、VPU。

同时，对传统的 OpenCV 图像处理库也进行了指令集优化，实现了性能与速度的显著提升。计算机视觉工具包括：

- OpenCV（3.3 版本）：预编译 OpenCV 和全新英特尔® 优化视觉库（Intel® Photography Vision Library），具有人脸检测 / 识别、眨眼检测、微笑检测等功能；
- OpenVX：基于图形的 OpenVX 实现，支持传统的 CV 操作和 CNN 原语。支持 Khronos OpenVX 神经网络扩展 1.2；
- 杂项：包括 OpenCL™ 驱动程序和运行库，以及媒体驱动程序，以简化与英特尔® Media SDK 和英特尔® SDK OpenCL™ 应用程序一起工作的计算机视觉 SDK，英特尔® MKL-DNN、CLDNN 都包含在其中，不需要单独下载。

另外，作为旨在快速、有效地在多个应用中实施计算机视觉和深度学习而推出的工具包，基于英特尔® 平台优化的 OpenVINO ™ 工具套件目前提供预先转换的 Caffe、TensorFlow、MXNet 模型的 MO 文件，如 VGG-16, VGG-19、Squeezenet、Resnet、Inception、CaffeNet、SSD、Faster-RCNN、FCN8 等，还具备超过 20 个预先训练的模型。软件开发人员和数据科学家等可以利用这些工具，快速实现个性化的深度学习应用，且可以使用 OpenCV、OpenVX 的基础库，去创建特定的算法，进行定制化和创新型应用的开发。

OpenVINO™ 工具套件在英特尔平台上让视觉成为现实，已帮助众多用户轻松开发和快速部署计算机视觉应用程序，在多种深度学习应用场景展示了人工智能解决方案所蕴藏的巨大潜力。

了解更多信息，请访问：
https://software.intel.com/zh-cn/openvino-toolkit

提供跨平台工具，支持计算机视觉和深度学习推理加速

[1] https://software.intel.com/en-us/articles/a-guide-for-setting-up-docker-based-openvino-development-environment-with-ubuntu-system

面向英特尔® 架构优化的TensorFlow

面向英特尔® 架构优化的 TensorFlow，是英特尔为了应对在 CPU 上运行深度学习模型面临的性能挑战而推出的优化版，能够确保深度学习类工作负载在各种情况下都可利用英特尔® MKL-DNN 基本运算单元高效运行。

为了显著提升性能，英特尔还持续采用其他举措对 TensorFlow 进行了优化。

计算图优化

英特尔推出了大量计算图优化通道，以便在 CPU 上运行时，将默认的 TensorFlow 操作替换为英特尔优化版本，确保用户能运行现有的 Python 程序，并在不改变神经网络模型的情况下提升性能；同时，消除不必要且昂贵的数据布局转换，以及通过将多个运算融合在一起，确保在 CPU 上高效地重复使用高速缓存，并处理支持快速向后传播的中间状态。

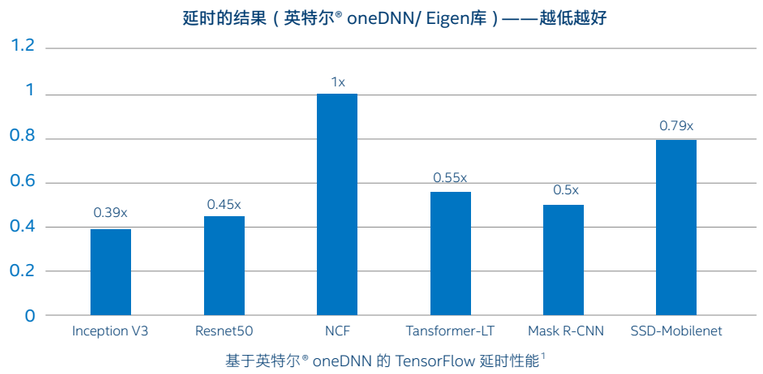
这些计算图优化措施进一步提升了性能，且没有为 TensorFlow 编程人员带来任何额外负担。此外，数据布局优化是一项关键的性能优化措施。此前，将来自 TensorFlow 本地格式的数据布局转换运算插入内部格式，在 CPU 上执行运算，并将运算输出转换回 TensorFlow 格式时，会因为转换造成性能开销的问题。而今利用英特尔® MKL 优化运算完全执行的子图，可消除子图运算中的转换。自动插入的转换节点能在子图边界执行数据布局转换，并通过另一个关键优化，即融合通道，将多个运算自动融合为高效运行的单个英特尔® MKL 运算。

其他优化

为确保在多种深度学习模型上实现最高的 CPU 性能，英特尔有针对性地调整了众多 TensorFlow 框架组件。比如，使用 TensorFlow 中现成的内存池分配器开发了一款自定义内存池分配器，确保其与英特尔® MKL 共享相同的内存池（使用英特尔® MKL imalloc 功能），从而不必过早地将内存返回至操作系统，避免了昂贵的页面缺失和页面清除。此外，对多个线程库（TensorFlow 使用的 pthread 和英特尔® MKL 使用的 OpenMP）页进行了深度优化，以使其能共存，避免了互相争抢 CPU 资源的情况，提升了资源的综合利用率。

了解更多信息，请访问：

- https://www.intel.ai/tensorflow/?_ga=2.231295069.330745958.1563951842597697079.1551333838&elq_cid=4287274&erpm_id=7282583
- https://www.intel.ai/improving-tensorflow-inference-performance-on-intel-xeon-processors/#gs.v0kayg



基于英特尔® oneDNN 的 TensorFlow 延时性能[1]

[1] 系统配置：英特尔® 至强® 铂金 8180 处理器 @2.50GHz; OS: CentOS Linux 7（Core）; TensorFlow 源代码 : https://github.com/tensorflow/tensorflow; TensorFlow 版本号 : 355cc566efd2d86fe71fa9d755ceabe546d577a

面向英特尔® 架构优化的Python分发包

面向英特尔® 架构优化的 Python 分发包，是一个由英特尔主导开发，功能强大的软件开发工具套件，提供了编写 Python 原生扩展所需的一切，包括 C 和 Fortran 编译器、数学库和分析器，并且集成了多个高性能数据分析和数学库，如 NumPy、SciPy、Scikit-learn、Pandas、Jupyter、matplotlib、mpi4py。

英特尔® Python 分发包是英特尔® Parallel Studio XE 的重要工具集之一，具备多项特性与高效率：

- 通过提供开箱即用的 uMath、NumPy、SciPy 和 Scikit-learn 等工具，加速包括数字、科学、数据分析、机器学习等在内的计算密集型应用；
- 集成英特尔® 性能库（如英特尔® MKL），内置最新的矢量化指令，如英特尔® AVX-512 和多线程指令、Numba 和 Cython，并应用对多线程构建模块库英特尔® TBB 的可组合并行，解锁 Python 基于多核处理器的并行应用功能，进而在基于英特尔® 架构的平台上提升 Python 程序运行性能，保证良好的系统兼容性，且不需要对代码进行任何更改；
- 支持 Python2.7、Python3.6 和最新一代英特尔® 架构处理器，提供优化的 TensorFlow、Caffe 等深度学习库和机器学习库，如支持向量机（SVM）和 K-means 预测、随机森林和 XGBoost 算法等，便于面向科学计算和机器学习等工作负载构建和扩展生产就绪算法。

经过基准测试，对比英特尔® Python 分发包与其它开放源码 Python 中 Scikit-learn 工具包（一个广泛用于数值计算、科学计算和机器学习的工具包）的效率，显示面向英特尔® 架构优化的 Python 指标有显著提升（见下图，效率越高，函数速度越快，与本机 C 语言速度越接近），如英特尔® Python 分发包 K-means 聚类、线性回归等算法的效率可以达到英特尔® DAAL 中 C 语言效率的 90%。

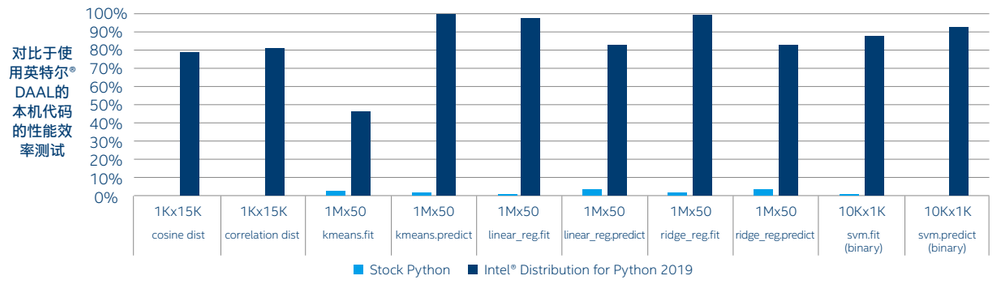
部署简单、易于使用也是英特尔® Python 分发包的一大特性——使用 Conda 软件包管理器和 Anaconda 云，用户只需执行一个命令，既可安装核心 Python 环境：

- conda install intelpython3 -c intel

此外，英特尔® Python 分发包是预先构建的二进制文件，也可以通过 pip、Docker images、YUM 和 APT repos 等各种渠道获得。

了解更多信息，请访问：

https://software.intel.com/en-us/distribution-for-python



英特尔的优化提升了 scikit-learn 程序包的效率，使其更加接近于本机代码在英特尔® 至强® 处理器上的运行速度

面向英特尔® 架构优化的 PyTorch

PyTorch 是一款开源的深度学习库，旨在帮助数据科学家和开发人员提高深度学习的训练和推理效率，具备高度的灵活性、易用性和训练、推理高速度，也深受业界青睐。

PyTorch 依托简洁、灵活的结构，提供了多项优异特性。比如，提供自动求导功能，使得模型搭建简单快捷；采用动态模型，在执行过程的任意环节都可以进行调试，大幅提升了易用性；能够自动计算梯度，并更新网络参数；训练方式灵活，允许用户自定义通讯方式，以实现新的通讯算法；提供了 Python、C++ 混合前端，可用 Python 前端做训练，而用 C++ 前端做部署。此外，PyTorch 具备强有力的社区支持，资源丰富。

为了基于 CPU 获得高性能、高效率，英特尔推出面向英特尔 ® 架构优化的 PyTorch。该优化版利用英特尔® MKL，使其能够充分运用 CPU 的并行计算能力和向量化技术，优来化矩阵乘法性能；采用英特尔® MKL-DNN，使其通过 CPU 的并行计算能力及向量化技术，并高效利用 CPU 片上缓存，来最大限度提高卷积、标准化等深度学习中常用运算性能。针对深度学习网络的层次或节点级的计算图特性，在节点级，英特尔优化了各种层，如卷积、矩阵乘法、ReLU、池等，最大限度地减少数据传输，并确保有效使用 SIMD 指令、执行单元、寄存器和内存缓存层次结构；在计算图级，英特尔优化措施使用多种数据顺序策略和层融合，来优化节点组，例如，将 ReLU 融合为卷积，以便数据在寄存器中的最后一个卷积周期仍执行 ReLU 操作。此外，借由英特尔® MKL-DNN API 关键的 DNN 层，还将英特尔 ® MKL-DNN 集成到 PyTorch 后端。

基于多项深度优化，面向英特尔® 架构优化的 PyTorch 经测试表明，基于英特尔® 至强® 铂金 8280 处理器，在 ResNet50、Faster R-CNN 和 RetinaNet 上的性能分别提高分别 7.7 倍、47 倍和 23.6 倍。[1]

此外，使用面向英特尔® 架构优化的 PyTorch 版本，通常不需要用户修改原有 Pythorch 脚本和代码。

了解更多信息，请访问：
https://software.intel.com/content/www/cn/zh/develop/articles/intel-and-facebook-collaborate-to-boost-pytorch-cpu-performance.html?wapkw=Pytorch&elq_cid=4287274&erpm_id=7282583

[1] https://software.intel.com/content/www/cn/zh/develop/articles/intel-and-facebook-collaborate-to-boost-pytorch-cpu-performance.html?wapkw=Pytorch&elq_cid=4287274&erpm_id=7282583

本手册涉及的专业词汇表

英文全称	英文缩写	中文全称
AI Operations	AIOps	智能运维
Artificial Intelligence	AI	人工智能
Automated Defect Classification	ADC	自动化缺陷识别
Auto-Regressive	AR	自回归模型
Autoregressive Integrated Moving Average Model	ARIMA	差分整合移动平均自回归模型
Autoregressive Moving Average Model	ARMA	自回归滑动平均模型
Classification And Regression Tree	CART	分类回归树
Convolutional Block		卷积块
Convolutional Neural Networks,	CNN	卷积神经网络
Cross Validation		交叉验证
Customer Relationship Management	CRM	客户关系管理系统
Database Management System	DBMS	数据库管理系统
Deep Learning Deployment Toolkit	DLDT	深度学习部署工具包
Defect Part Per Million	DPPM	每百万的缺陷数量
Depthwise Separable Convolution		深度可分离卷积
Dynamic Random Access Memory	DRAM	动态随机存取存储器
Enterprise Resource Planning	ERP	企业资源计划系统
Extract-Transform-Load	ETL	
Field-Programmable Gate Array	FPGA	现场可编程门阵列
Fine-tuning		模型微调
Frame Per Second	FPS	帧率
Gated Recurrent Unit	GRU	门控循环单元
Gradient Boosting Decision Tree	GBDT	梯度提升迭代决策树
Gradient Disappearance		梯度消失
Identity Block		恒等块
Inference Engine		推理引擎
Inline Defect Metrology		同步缺陷测量
Intel® Advanced Vector Extensions 512	Intel® AVX -512	英特尔® 高级矢量扩展 512
Intel® Math Kernel Library	Intel® MKL	英特尔® 数学核心函数库
Intel® Math Kernel Library for Deep Neural Networks	Intel MKL-DNN	面向深度神经网络的英特尔 ® 数学核心函数库
Intel® Photography Vision Library		英特尔® 优化视觉库
Intermediate Representation	IR	中间表示

英文全称	英文缩写	中文全称
Internet of Things	IoT	物联网
Long and Short term Time series Network	LSTNet	长短期时间序列网络
Long Short-Term Memory	LSTM	长短期记忆
Machine Vision		机器视觉
Mean Squared Error	MSE	均方误差
Mobile Edge Computing	MEC	移动边缘计算
Model Optimizer		模型优化器
Moving-Average	MA	移动平均模型
Multi-Grained Cascade Forest	gcForest	多粒度级联森林
Multi-Grained Scanning		多粒度扫描
Multiple-Class		多分类
Multi-Layered Gradient Boosting Decision Trees	mGBDT	多层梯度提升决策树
Open Neural Network Exchange	ONNX	开放神经网络交换
Optical Character Recognition	OCR	光学字符识别
Optimum Design Baseline		最优产品设计基线
Predictive Maintenance	PdM	维护性预测
Random Forest	RF	随机森林
Recurrent Neural Networks	RNN	递归神经网络
Residual Net	ResNet	残差网络
Resilient Distributed DataSet	RDD	弹性分布式数据集
Scanning Electron Microscope	SEM	高清扫描式电子显微镜
Shortcut Connection		近道连接
Single Shot MultiBox Detector	SSD	
Supply Chain Management	SCM	供应链管理系统
Support Vactor Regression	SVR	向量回归
Support Vector Machine	SVM	支持向量机
Time Series		时间序列
TimeSequencePipeline		时序管道
TimeSequencePredictor		时序预测
Transfer Learning for Training		迁移学习训练
Tree Ensembles		树集成
You Only Look Once	YOLO	

免责声明:

性能测试中使用的软件和工作负荷可能仅在英特尔微处理器上进行了性能优化。诸如 SYSmark 和 MobileMark 等测试均系基于特定计算机系统、硬件、软件、操作系统及功能。上述任何要素的变动都有可能导致测试结果的变化。请参考其他信息及性能测试（包括结合其他产品使用时的运行性能）以对目标产品进行全面评估。更多信息，详见 www.intel.com/benchmarks。

在特定系统的特殊测试中测试组件性能。硬件、软件或配置的差异将影响实际性能。当您考虑采购时，请查阅其他信息来源评估性能。关于性能和基准测试程序结果的更多信息，请访问 www.intel.com/benchmarks。

英特尔技术特性和优势取决于系统配置，并可能需要支持的硬件、软件或服务得以激活。产品性能会基于系统配置有所变化。没有任何产品或组件是绝对安全的。更多信息请从原始设备制造商或零售商处获得，或请见 intel.com。

优化声明：英特尔编译器针对英特尔微处理器的优化程度可能与针对非英特尔微处理器的优化程度不同。这些优化包括 SSE2、SSE3 和 SSSE3 指令集和其他优化。对于非英特尔微处理器上的任何优化是否存在、其功能或效力，英特尔不做任何保证。本产品中取决于微处理器的优化是针对英特尔微处理器。不具体针对英特尔微架构的特定优化为英特尔微处理器保留。请参考适用的产品用户与参考指南，获取有关本声明中具体指令集的更多信息。

声明版本：#20110804

没有任何产品或组件是绝对安全的。

描述的成本降低情景均旨在在特定情况和配置中举例说明特定英特尔产品如何影响未来成本并提供成本节约。情况均不同。英特尔不保证任何成本或成本降低。

英特尔并不控制或审计第三方数据。请您审查该内容，咨询其他来源，并确认提及数据是否准确。

加速 AI 实践，请访问:

官网
Intel.cn/ai

微博
@ Intel Business

微信
英特尔商用频道