计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.200900045

DragDL:一种易用的深度学习模型可视化构建系统

汤世征　张岩峰
东北大学计算机科学与工程学院　沈阳 110000
(tangsz1023@qq.com)

摘　要　深度学习在各个领域得到了普遍的应用,但是用户在应用深度学习时仍然面临两方面的问题:1)深度学习有着复杂的理论背景,非专业用户缺乏建模以及调优的背景知识,难以构建性能优化的模型;2)数据预处理、模型训练、预测等过程往往涉及比较复杂的编程实现,给没有程序设计基础的非专业用户在入门时带来了一定的困难。针对以上两点易用性问题,文中提出了一种易用的深度学习模型可视化构建系统DragDL,其目的在于降低用户进行数据预处理、模型训练、监控、在线预测等工作的难度。该系统基于PaddlePaddle框架,支持以拖拽图形算子的方式在画布上搭建深度学习网络结构以及推理预测功能,并将数据预处理操作过程抽象成数据流图展示,以方便用户理解和调试。系统还提供训练过程中的质量监控和性能监控的可视化功能,帮助用户实时观察训练情况。同时,DragDL提供经典模型库帮助用户完成建模任务,支持以微调经典模型的方式构建新的模型,降低用户建模时的难度。DragDL基于集群服务器和Web客户端进行部署,服务器为每个训练任务构建虚拟机服务,并支持大规模异步任务调度,具有一定的并发处理能力。

关键词:深度学习;图形化编程;数据流图;预训练模型;PaddlePaddle
中图法分类号　TP319

DragDL:An Easy-to-Use Graphical DL Model Construction System

TANG Shi-zheng and ZHANG Yan-feng
School of Computer Science and Engineering,Northeastern University,Shenyang 110000,China

Abstract　Deep learning has broad applications in various fields. However, users still need to face problems from two aspects when applying deep learning. First, deep learning has a complex theoretical background, non-professional users lack background knowledge in modeling and tuning. It is difficult for them to build performance-optimized models. Second, modules such as data preprocessing, model training, and prediction often involve more complicated programming implementations, which bring some difficulties in getting started for non-professional users who have no programming skill background. In view of the above two issues of usability, this paper proposes an easy-to-use graphical deep learning model construction system, DragDL. The purpose of DragDL is to reduce the difficulty of data preprocessing, model training, monitoring, online prediction and other tasks for users. The system is based on the PaddlePaddle framework and supports building a deep learning network structure on the canvas by dragging graphical operators, supporting inference and prediction functions, and abstracting the data preprocessing operation process into a dataflow graph, which is convenient for users to understand and debug. The system also provides visualization functions for performance monitoring during the training process. At the same time, DragDL provides a classic model library, which allows users to build new DL network by tuning the existing classic model network. DragDL is deployed based on a centralized server and Web client. The server provides a virtual machine service for submitted tasks and supports large-scale asynchronous task scheduling to have concurrent processing capabilities.

Keywords　Deep learning, Graphical programming, Dataflow graph, Pre-trained model, PaddlePaddle

1　引言

深度学习技术的应用场景非常广泛,例如在安保、教育、制造业等领域都得到了广泛的应用。尽管越来越多的理论和编程框架已经被提出,但是对于初学者用户,在尝试应用深度学习解决问题时,仍然面临以下两点困难:1)深度学习理论的复杂性使用户需要掌握复杂多样的网络层原理以及优化技巧等;2)构建深度学习任务往往需要数据预处理、模型构建、训练等多个过程,其中每一个过程都需要熟练掌握编程技术以及具有一定的编程工作量。特别是对于深度学习的教学来说,掌握一门编程语言(如Python)和一些数学知识(如线性代数)是学习深主爱产技术的基础,需要有一定的编程基础和

到稿日期:2020-09-05　返修日期:2020-12-18
基金项目:国家自然科学基金(61672141);辽宁省重点研发计划(2020JH2/10100037);中央高校基本科研业务费(N181605017,N181604016)
This work was supported by the National Natural Science Foundation of China(61672141), Key R&D Program of Liaoning Province(2020JH2/10100037) and Fundamental Research Funds for the Central Universities(N181605017,N181604016).
通信作者:张岩峰(zhangyf@mail.neu.edu.cn)

数学基础才能够理解和应用深度学习技术。例如，用户使用 Tensorflow 和 Pytorch 构建深度学习任务时，需要完成数据预处理、模型训练、预测、评估等多个步骤以及其中大量代码的设计、编写和调试工作。

近年来，已经出现一些系统框架提供给用户图形化的接口，以降低用户使用机器学习技术的门槛，如阿里云的 PAI-Studio[1]、中科院计算所的 Easy-ML[2] 等。这些系统将机器学习数据处理和模型构建的过程抽象为有向无环图（Driected Acyclic Graph，DAG），相应的特征工程或模型计算被提炼为一系列计算单元（即算子），并被抽象为有向无环图中的节点，节点间的连线表示操作算子之间的依赖关系，整个计算任务会以 DAG 的形式绘制出来，以帮助用户理解计算逻辑。但是，已有的这些框架在很多方面仍然存在若干不足，如 Easy-ML 中缺乏对深度学习任务的支持，PAI-Studio 系统中仅仅提供 API 编程接口调用 Tensorflow，Pytorch 等框架，缺乏图形化构建模型的支持，初级用户使用起来仍较困难。

针对深度学习框架易用性的问题，本文提出了一种易用的深度学习模型构建系统 DragDL，用于帮助用户完成数据处理、模型构建、训练、预测部署等工作。该系统基于 PaddlePaddle 框架[3]，支持以拖拽图形（GUI）算子方式在画布上构建深度学习任务以及在线推理预测。深度学习计算任务被抽象成包含模型、数据、数据操作等算子的数据流图，利用这些 GUI 算子构建数据流图可便于用户设计深度学习任务，也方便其直观理解处理过程。DragDL 系统也提供训练过程中的质量监控和性能监控的可视化功能，帮助用户实时观察训练情况。同时，DragDL 提供经典模型库，用于帮助用户完成建模任务，用户可以微调网络结构以设计新的网络模型，借助知识迁移提高模型性能，降低用户建模时的难度。最后，为了方便部署，DragDL 基于集群服务器和 Web 客户端方式来部署，服务器为每个训练任务构建虚拟机服务，并支持大规模异步任务调度，具有一定的并发处理能力。

总的来说，相比已有的深度学习框架，DragDL 具有以下特点：

（1）支持便捷的图形化方式，支持拖拽式构建数据流图和深度学习模型；

（2）提供丰富的预训练模型库，允许用户选择多种经典深度模型，在其基础上进行微调和扩展来构建新模型，降低用户建模的难度；

（3）高效的 Web 服务架构，服务器端支持计算任务的异步调度，用户可以基于 Web 端监测训练过程，以方便用户对模型进行调整，且有助于理解和学习。

2 相关工作

近年来涌现出了若干深度学习模型构建框架支持以图形化方式构建复杂的机器学习或深度学习任务，包括阿里云的 PAI-Studio、中科院计算所的 Easy-ML、新加坡国立大学的 Rafiki[4]。这些系统基于数据流编程模型的思想，将高度调优的计算模块或算法封装为数据流图中的算子，并将数据流图绘制在用户界面，用户通过简单拖拽特定算子并设计算子之间的连接关系就能构建出复杂的机器学习任务。

PAI-Studio 提供了可视化构建机器学习任务的功能。PAI-Studio 中封装了大量调优后的机器学习算法，包括时间序列分析、文本分析、图分析等。计算任务可以分布式地运行在阿里云的集群之上，依赖庞大的集群和数据服务来提供高效的计算服务。PAI-Studio 也提供了 Tensorflow，Caffe 等深度学习功能的接口，用户可以指定代码文件和数据路径，同时也支持把建好的模型可视化地展示出来，以帮助用户调试模型。但是，PAI-Studio 只支持已经构建好的模型的可视化展示，即使 PAI-Studio 提供了深度学习接口，但仍然需要手写代码才能搭建模型，并不能支持图形化方式的模型构建。

EasyML 将机器学习任务归纳为数据预处理、特征工程、算法训练、测试等步骤，并将整个过程抽象为 DAG 形式的数据流图。复杂的算法被抽象为算子节点的连接组合，算子可能会包括多个输入端、输出端，可以构建复杂的计算逻辑。EasyML 底层依赖于 Spark，Hadoop 等框架，支持分布式或单机方式运行。EasyML 的数据流抽象给用户构建模型带来了很大便利，但遗憾的是，它并没有针对深度学习模型予以支持。

Rafiki 封装了多种深度学习模型，并且提供了编程接口以方便开发者使用深度学习服务。开发者提交数据和任务配置之后，提交的任务在分布式集群上运行。Rafiki 也提供了分布式超参优化方式加快模型的训练速度，参数服务器中利用当前最优的模型参数来填充其余模型副本。Rafiki 可以在一个学习任务中同时训练多个模型，在计算资源有限的情况下，会用强化学习选择其中几个模型进行预测，以优化多个模型的预测速度。但是，Rafiki 缺乏易用的 GUI 界面，一定程度上影响了在配置模型、数据处理方法方面的灵活性。

PAI-Studio 和 Easy-ML 借助 DAG 图来表示计算任务的逻辑关系，这是一种非常好的可视化方式，系统在设计时基于 DAG 模型支持用拖拽方式去构建计算任务非常易用。此外，该系统考虑到了深度学习模型能够凭借强大的特征提取能力来减少用户进行数据处理的工作量，图形化方式能够给用户提供更为高效便利的体验。

3 系统总体架构

本节概述 DragDL 系统的整体架构。如图 1 所示，DragDL 基于 Web 的 C/S 架构提供服务，下文分别介绍客户端和服务器端。

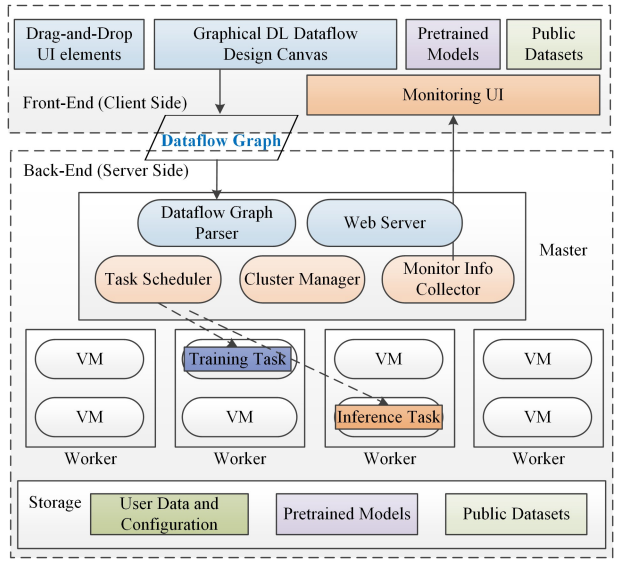

图 1 DragDL 系统架构

Fig. 1 System architecture of DragDL

系统客户端(前端)基于 Web 技术设计,支持用户通过在数据流图画布上拖拽 GUI 算子以构建数据处理逻辑以及深度学习模型。系统提供了一系列预训练的深度学习模型,让模型构建的过程更加方便,并且提供了一些公开数据集以便于用户测试调试模型。此外,系统还提供 GUI 界面用于可视化训练过程和测试集上的性能表现。客户端通过 Restful 方式与服务器进行通信,客户端上生成的数据流图被提交到服务器端。

服务器端(后端)采用 Master-Slave 的分布式集群模型,Master 节点提供了 Web 服务、数据流分析器、任务调度器以及监控数据收集器等几个模块。其中,数据流图分析器主要从用户提交的数据流图中解析出数据变换操作、选用的预训练模型以及数据集等关键信息,并将这些信息转换成代码调用。由于 PaddlePaddle 抽象了丰富的算子操作,具有良好的封装性,适合对应到合理粒度的 GUI 算子,因此 DragDL 基于 PaddlePaddle 框架进行实现,并将用户提交的数据流图转换成 PaddlePaddle 风格的 Python 代码。根据数据预处理流的信息可以解析出依赖关系,并转换生成、连接对应的处理对象。预训练模型的信息则转换成 PaddleHub[5] 代码,负责完成模型的加载和生成。

被定义好的训练或预测任务被任务调度器分发到 worker 节点上,worker 节点一般运行在分布式的 Docker 容器上,作为 VM 由集群管理器完成资源的调度和分配。监控信息收集器则会在训练过程中从 worker 上收集训练、预测信息(包括性能指标信息、异常、执行时长等)反馈给用户。DragDL 同时采用分布式存储以及数据库来维护预训练模型、用户数据集以及配置数据。接下来将介绍系统的核心特征。

4 数据流图模型

为了解决计算任务中复杂的计算模块及控制依赖关系的表达问题,DragDL 采用数据流图模型表示任务逻辑,图中的节点表示不同的计算操作,操作之间的依赖关系(计算的输入输出)使用节点之间的连线进行表达[6]。原始数据从起点开始输入数据流图中,经过不同节点施加不同的变换,并依次传递到下一节点,数据流图中的终点输出 出目标数据。

DragDL 的计算任务中含有 3 类算子:1)数据集算子,为训练或预测任务提供数据输入;2)数据操作算子,完成数据的预处理任务;3)模型算子,代表计算任务所使用的深度模型,学习如何基于数据预测结果。计算任务的执行过程等同于源数据从数据集节点中输入,经过不同的操作算子完成不同的转换操作,最终被加载到模型中进行训练或预测任务。数据流图使用边表示操作之间的依赖关系,在输出端即得到目标数据。将 F 考虑成数据变换的操作集合,其中某一操作 $f \in F$,f 可以是一个函数,接受一种形式的数据输入,经过 f 中的处理逻辑转化为特定的输出形式。在数据处理的过程中可以构建一个操作序列来表示数据的处理过程,如 $S=\{f_1, f_2, \cdots, f_n\}$,其中 n 为需要操作的个数,算子 f_i 接受来自 f_{i-1} 的输入作为输入,并将计算后的输出传递到 f_{i+1} 的输入端。

以图像处理应用为例,通过研究常用的图像处理操作[7],DragDL 逐一实现常用的图像处理相关的数据预处理算子,集成到"算子库"中供用户编辑数据流图时使用。表 1 列出了系统中提供的部分常用的图像处理相关的算子。

表 1 图像处理相关的部分数据流算子

Table 1 Dataflow operators for image processing

算子名称	作用
Resize	图像长宽调整
Normalize	归一化处理
RandomAffine	随机仿射变换
ColorJitter	颜色抖动
RandomHorizontalFlip	随机水平翻转
Lambda	自定义 Lambda 方法

客户端在 Dataflow Canvas(数据流设计画布)模块中集成了 AntV 图编辑库[8],为用户提供编辑数据流图的功能,用户在 Web 客户端的数据流设计画布上拖拽数据集、算子、模型等对应的 GUI 节点,并设置节点属性,完成数据流图的创建。提交任务时客户端将输出一个描述数据流图属性信息的 JSON 文件给服务器端,该 JSON 文件中包括图中算子的节点信息,以及节点之间的依赖关系,如图 2 所示。

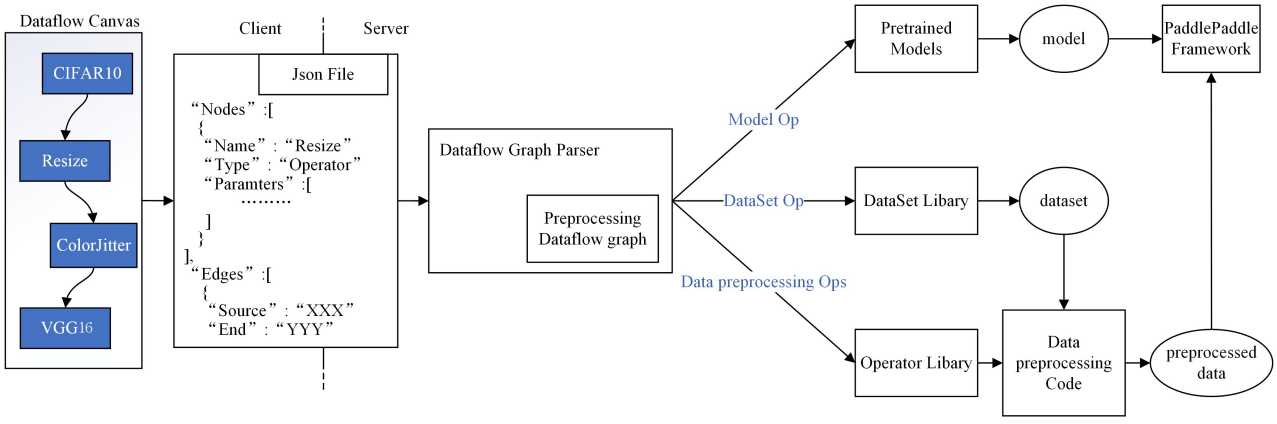

图 2 数据流图处理过程

Fig. 2 Process of dataflow graph processing

服务器端主要包含四大模块进行数据流图的解析任务:1)预训练模型库(Pretrained Models)中集成了多种预训练模型,它基于 PaddleHub 实现,支持多种经典的深度学习模型,用户可以基于这些经典模型的网络结构进行微调和扩展,设计新的网络模型结构;2)算子库(Operator Libary)中通过对 Resize,Lambda 等多种底层操作的封装,提供种类丰富的数据操作算子;3)数据集库(DataSet Libary)中设有内置数据集,在数据集加载数据时进行算子的调用,完成数据处理工

作，系统也设计了数据集接口以便用户自定义数据集；4）数据流解析器（dataflow parser）通过协调其余模块来完成对数据流图表示文件的解析任务，输出表示数据处理和模型网络计算逻辑 PaddlePaddle 代码，交付底层的 PaddlePaddle 框架进行训练。在数据流图的解析过程中，解析器会首先从客户端传递过来的表示数据流图的 JSON 文件中解析出不同类型的算子节点以及算子节点之间的依赖关系，并调用数据、算子、模型相关模块的接口来完成从数据流图到 PaddlePaddle 代码的转换。

如上文所述，DragDL 将为每个客户端提交的数据流图处理任务创建一个 DataflowGraph 实例。图 3 给出了 DataflowGraph 类的核心功能，当被实例化时，首先调用 init()方法从 JSON 文件中解析出数据集节点（通过 parseDataSet 方法），并解析出数据操作算子和它们之间的依赖关系（通过 parseOps 方法），最后还要解析出所采用的模型（通过 parseModel 方法）。目前的模型配置仅支持已经在系统中注册过的模型（包括经典模型库和用户自定义注册的模型），如果用户需要自定义模型，可以在系统注册。接下来生成一个数据流图的执行过程程序，根据节点的类别调用 DataSet Libary，Operator Libary，Pretrained Models 等模块提供的接口来完成数据流图到代码的翻译。被调度器调度执行时，调用数据流图对象中的 run()方法则可以依次加载数据集（data），并施加算子流中的数据预处理操作（ops）和模型操作（model）。数据流图中的数据预处理流程通过解析器翻译成 Python 代码，用于处理数据集，并保持和标签的对应关系。解析后的模型算子操作交由 PaddlePaddle 框架进行加载并初始化，之后 PaddlePaddle 将前面预处理得到的数据集（包括训练数据和标签）作为输入开始训练。

```
Class DataflowGraph:
  def __init__(json_file):
    // initilize dataset from JSON file
    dataset=parseDataSet(json_file)
    // initilize operator dependences from JSON file
    ops=parseOps(json_file)
    //initilize model by PaddleHub
    model=parseModel(json_file)
  //main entry point to invoke dataflow processing
  def run():
    tmp=data(dataset)
    for op in ops:
      tmp=op(tmp)
    model(tmp)
```

图 3 DataflowGraph 类伪代码

Fig. 3 Pseudocode of DataflowGraph

训练过程由基于 Web 的监控平台展示，关于图形化的监控平台将在后文进行介绍。训练好的模型将保存在服务器存储系统中，用户可以在监控平台上查看训练准确率。在之后的推理过程中，用户可直接调用保存的模型进行推理预测。

5 经典模型复用与预训练模型支持

模型构建过程中涉及复杂的深度学习理论，模型复用则可以减小用户在通用网络模块上的设计难度。为了在模型复用时确定模型参数的作用边界并复用功能类似的模块和网络，以方便网络层替换时不影响其他功能模块的网络层参数，DragDL 将整个模型网络分为若干部分。以图像分类应用为例，可以将某个图像分类网络分为两大部分：特征提取网络和分类网络。特征提取网络包含若干与特征提取功能相关的网络层，常见的卷积神经网络中浅层的卷积核倾向于提取点、颜色等基础特征，后继的卷积层通过相互作用逐渐学习到线段、形状等抽象特征[9]。而分类网络包含若干与分类判断功能相关的网络层，基于特征提取网络层输出的特征向量作为输入进行推理判断，例如全连接层进行分类判断，RNN 层进行序列分析。

系统通过复用经典模型中特征提取网络的结构为用户提供模型构建支持。模型构建时仅复用经典模型中特征提取的网络部分，等价于对经典模型中原始的分类网络结构进行截断，如图 4 所示。根据目标任务的特点（如分类的类别）构建合适的分类网络，连接特征提取网络和分类网络，进而构成完整的目标模型。

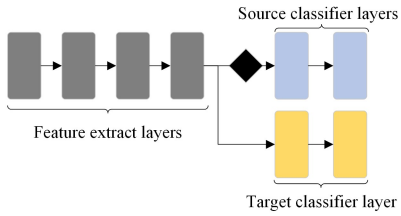

图 4 经典模型的复用

Fig. 4 Reuse of classical models

DargDL 为了实现对经典模型的复用设计了 Net 类，伪代码如图 5 所示，当进行模型构建时将会实例化 Net 类对象。Net 的 init()方法主要由 PaddleHub 获取经典模型中的特征提取网络部分，build_net()方法将根据参数构建全连接层和 SoftMax 层，并将其连接到特征提取网络上，用于分类推断。DragDL 基于 PaddleHub 实现了对多种强大的经典模型和预训练模型的支持，如 ResNet[10]，VGG[11]等。图 6 给出了一个对 VGG16 网络结构进行复用、使其适应于某一含有 108 个对象的分类任务的例子，在复用的过程中将保留池化层、卷积层等与特征提取密切相关的网络结构，并将其连接到为完成新任务而构建的分类网络。

```
Class Net:
  def __init__(pretrained_model,hidden_units,nums_classes)
    //get feature layers of pretrained_model by PaddleHub feature_layer=
      PaddleHub.load_featureLayer(pretrained_model)
  //construct classify layers and connect to the feature layers def build_net
    ():
    //connect feature layer and fc layer (for classify)
    connected_layer=feature_layer
    if hidden_units is not None:
      connected_layer=fc(feature_layer,hidden_units,'relu')
    softmax=fc(connected_layer,18,'softmax')
    return softmax
```

图 5 Net 类伪代码

Fig. 5 Pseudocode of Net

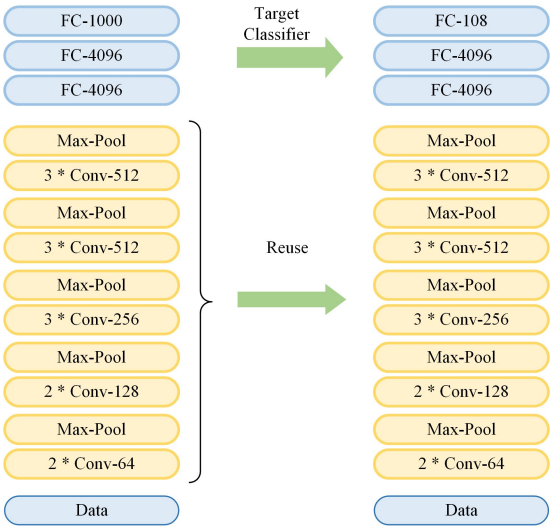

图 6 基于 VGG 构建新模型

Fig. 6 Building a new model based on VGG

DragDL 将训练过程分为初始化和微调两个阶段。初始化时，考虑源域 $D_s=\{x_i,y_i\}_{i=1}^{n}$ 和目标域 $D_t=\{x_j,y_j\}_{j=n+1}^{n+m}$，$x$ 和 y 分别表示领域上的数据和标签信息，假定它们的特征空间相同、数据分布不同，即：

$$\begin{cases} X_s=X_t \\ P_s(x_s)\neq P_t(x_t) \end{cases} \tag{1}$$

其中，X_s 和 X_t 分别代表源域及目标域的特征空间，$P_s(x_s)$ 和 $P_t(x_t)$ 则分别表示源域及目标域的数据分布[12]。使用 PaddleHub 中保存的预训练模型参数去初始化特征提取网络部分，分类网络部分仅仅使用随机初始化方式，复用预训练模型中的参数同时也能够将预训练模型强大的特征提取能力迁移到新的分类任务中[9]。训练则可根据源域与目标域的差异性选用不同的微调方式：1）微调分类网络层的方式，如图 7(a)所示，PaddlePaddle 冻结特征提取网络的参数，使其不参与优化过程，由于已使用预训练模型参数进行初始化，这部分网络仍具有一定的特征提取能力，训练时仅仅优化分类网络部分的参数，模型结构的复杂度降低使得这种方式能够降低因用户数据量不足而导致的过拟合风险[13]；2）微调模型的方式，如图 7(b)所示，PaddlePaddle 将会在训练过程中优化所有的模型参数，包括特征提取网络，这种方式能在用户数据与预训练数据集的相似度不高时，提高模型在新任务下的特征提取能力。

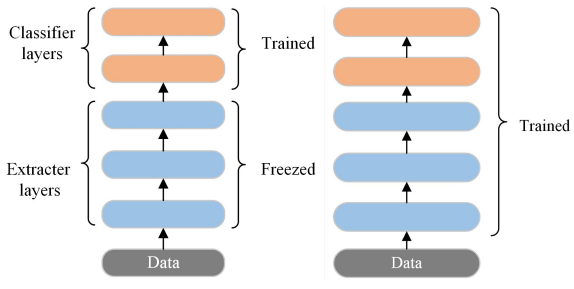

图 7 微调策略的区别

Fig. 7 Differencre of finetune strategy

6 服务端异步任务调度

系统提供了 Web 服务，用于处理用户执行训练、在线预测等任务的请求。但训练、预测等计算过程非常消耗计算资源和时间，而单机的任务承载量有限，并且用户请求到系统响应将会存在非常大的交互时延[14]，因此，在执行训练、预测任务时需要将计算单元与 Web 应用进行解耦。DragDL 采用异步执行的方式，由单独的 worker 负责执行长时间计算任务，降低服务器的压力，提高对用户请求的响应速度。如图 8 所示，一旦用户的训练等长时间计算任务被提交，将由任务调度模块将任务发送到任务队列中，同时返回唯一的任务 ID，用于对异步任务进行控制和监测。多个 worker 采用轮询的方式从队列中获取执行任务，训练或预测完成时将结果返回任务队列或持久化到本地磁盘。系统启动初期将启动若干个 worker 节点来不断监听任务队列，用户提交的任务在解析后通过任务队列传递到 worker 节点中进行计算。各个 worker 节点监听到队列事件之后将会根据自己的负载情况决定是否接纳新任务，如 worker 节点中运行的任务数低于设定的阈值，worker 节点将从队列中拉取任务并开辟新的进程执行任务，执行过程中 worker 将过程相关日志持久化到对应的 log 文件中，同时保存 Check-Point 到本地磁盘中便于任务失败后的重试。用户发出任务请求时，服务器将首先检查任务是否处于已完成状态(Failed, Successed)，若处于已完成状态，则将直接从数据库和本地文件中读取相应的数据；若暂时显示未完成状态，则将会通过异步任务模块与任务队列进行通信，更新任务执行的状态或者发送相应的任务控制信息(终止、重试)，若任务完成，则信息收集器将会收集 log 信息并持久化，以便之后的查询。

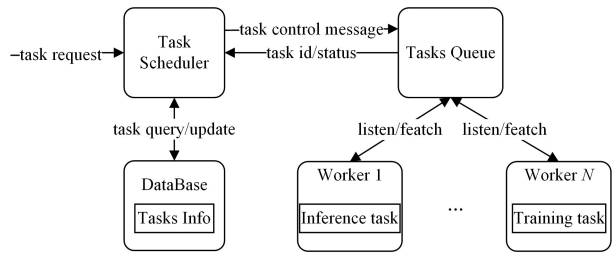

图 8 异步任务调度模型

Fig. 8 Asynchronous task scheduling model

7 系统前端展示

本节将介绍 DragDL 系统的前端设计。系统前端界面中主要有：1）数据流图编辑界面；2）任务管理界面；3）任务监测及在线预测界面。

如图 9 所示，数据流图编辑界面中左侧为菜单栏，提供数据集、数据处理、模型 3 类算子，中间为支持拖拽的图编辑画布，点击节点之后可以弹出节点编辑栏。通过算子节点可以获取数据集信息，也可以设置数据集配置，点击数据预处理算子可以指定数据预处理的参数(如 resize，重新指定图片大小)，也可以指定模型算子的超参数等。算子之间通过连接线相连，体现算子操作之间的依赖关系，即绘制完成了数据流图。

图 9 数据流图设计画布

Fig. 9 Canvas for dataflow design

用户可以从菜单栏中选择数据集(用户数据集将与系统内置的数据集一同列出)，并拖拽不同的操作算子组建数据处理流，点击算子节点可以在右侧弹出的节点编辑栏中输入算子的参数或者 Lambda 表达式，用户构建的数据流将作用于数据集中的样本数据。预训练模型可以拖拽到界面中并配置 BatchSize、是否训练特征提取层等超参数。系统将数据集按一定比例划分为训练集、验证集和测试集，用户构建好的数据流图将被服务器解析为具体的功能调用模块，模型则被延迟到训练任务开始时进行构建，同时训练时系统将会使用预训练模型的参数初始化特征提取层，并根据用户的选择(是否冻结特征提取层)进行微调训练。用户在数据流图编辑界面构建完成数据流和模型后，将会在个人中心看到任务列表，任务列表中显示出用户目录下的任务。

在任务监测界面，系统将可视化出任务训练过程中的误差、准确度的变化趋势，以及任务配置、运行状态等信息，用户可以根据训练过程中的指标信息判断模型的性能，进而选择重新训练或进行预测，如图 10 所示。当训练任务执行完成后，可以在监测界面看到模型在测试集上的性能表现，并通过在线预测接口上传数据进行推理预测，预测结果将以饼状图的形式返回给用户。

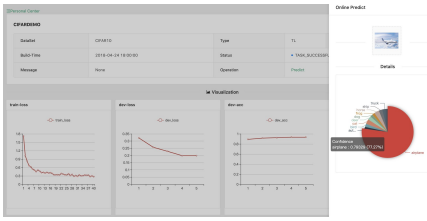

图 10 在线监控和预测界面

Fig. 10 Online monitoring and predicting interface

结束语 DragDL 支持用户在可视化场景下进行数据处理、模型构建、模型训练、在线预测等一系列过程。DragDL 使用数据流图抽象深度学习任务构建的过程，使得用户在画布上拖拽算子便可处理数据、构建模型等。相比复杂的程序设计过程，这极大地降低了用户使用时的难度。同时，DragDL 在数据流图中为用户提供了丰富的数据、模型相关算子，以满足用户构建任务时的需求。经典模型的复用降低了用户自定义深度模型的难度，并结合迁移学习技术提高了模型的训练效率。计算任务的异步执行方式则为系统提供了一定的并发处理能力，同时提高了系统的响应速度。任务调度则为用户屏蔽了任务执行的底层细节，使用户方便启动、停止任务等。此外，DragDL 为用户提供了训练可视化的功能，用户能够随时查看任务的实时状态、模型性能等，方便用户做出进一步的决策。在线预测功能的实现则降低了用户模型部署的压力，极大地方便了用户使用模型来解决实际问题。

参 考 文 献

[1] PAI Studio[OL]. https://data.aliyun.com/pai/studio.

[2] GUO T, XU J, YAN X, et al. Ease the process of machine learning with dataflow[C]// Proceedings of the 25th ACM International Conference on Information and Knowledge Management (CIKM 2016). 2016: 2437-2440.

[3] PaddlePaddle[OL]. https://www.PaddlePaddle.org.cn/.

[4] WANG W, GAO J, ZHANG M, et al. Rafiki: Machine learning as an analytics service system[J]. Proceedings of the VLDB Endowment, 2018, 12(2): 128-140.

[5] PaddleHub[OL]. https://www.PaddlePaddle.org.cn/hub.

[6] ABADI M, BARHAM P, CHEN J, et al. Tensorflow: A system for large-scale machine learning[C]// Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 2016). 2016: 265-283.

[7] SONKA M, HLAVAC V, BOYLE R. Image processing, analysis, and machine vision[M]. United States: Cengage Learning, 2014.

[8] AntV[OL]. https://antv.vision/.

[9] YOSINSKI J, CLUNE J, BENGIO Y, et al. How transferable are features in deep neural networks? [C]// Proceedings of Advances in Neural Information Processing Systems (NIPS 2014). 2014: 3320-3328.

[10] HE K, ZHANG X, REN S, et al. Deep residual learning for image recognition[C]// Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2016). 2016: 770-778.

[11] SIMONYAN K, ZISSERMAN A. Very deep convolutional networks for large-scale image recognition[J]. arXiv: 1409.1556, 2014.

[12] LONG M, ZHU H, WANG J, et al. Deep transfer learning with joint adaptation networks[C]// International Conference on Machine Learning. 2017: 2208-2217.

[13] HOWARD J, RUDER S. Universal Language Model Fine-tuning for Text Classification[C]// Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics. 2018: 328-339.

[14] CRANKSHAW D, WANG X, ZHOU G, et al. Clipper: a low-latency online prediction serving system[C]// Proceedings of the 14th USENIX Conference on Networked Systems Design and Implementation. 2016: 192-209.

TANG Shi-zheng, born in 1994, postgraduate. His main research interests include data mining and deep learning.

ZHANG Yan-feng, born in 1982, professor, Ph.D supervisor, is a senior member of China Computer Federation. His main research interests include big data mining, large-scale machine learning and distributed systems.