图片包含 游戏机, 镜子, 桌子, 食物

描述已自动生成

本科生毕业论文 (设计)

徽标, 公司名称

描述已自动生成

题 目 **稀疏神经网络的推理性能预测**

学 院 **计算机学院**

专 业 **计算机科学与技术（试验班）**

学生姓名 **林俊卿**

学 号 **2018141461015** 年级 **2018**

指导教师 **段磊 孙广中（中国科学技术大学）**

教务处制表

二〇二二年五月二十日

# **稀疏神经网络的推理性能预测**

计算机科学与技术（试验班）

学生 林俊卿 指导教师 段磊，孙广中

［**摘要**］为了获得高推理性能的稀疏神经网络，神经网络稀疏化方法需要对网络的推理性能进行评估。由于高昂的部署成本，实际过程中常使用代理器来评估稀疏神经网络的推理性能。在部署过程中不同的优化策略会对稀疏神经网络的推理性能产生影响，因此对TensorRT推理框架的部署过程中的算子间融合和算子间并行这两个算子间优化策略进行了验证与分析，提出了基于图结构的稀疏神经网络的推理性能预测模型（pipSG）。pipSG模型包括图特征提取和推理性能预测两个模块，其中图特征提取模块使用图卷积神经网络对稀疏神经网络的模型计算图进行特征提取，能够间接学习TensorRT推理框架部署过程中的算子间优化策略；推理性能预测模块使用多层感知机对稀疏神经网络的推理性能进行预测。使用结构化剪枝算法ABCPruner得到DenseNet、GoogleNet、VGG16和VGG19四种神经网络的稀疏神经网络集，然后使用TensorRT推理框架在Nvidia2080TiGPU上对其进行部署测量得到对应的推理延迟数据集ILDS，在该数据集上验证了pipSG模型的有效性，同时通过消融实验分析pipSG中聚合函数和激活函数对模型精度的影响。

［**关键词**］神经网络稀疏化；推理性能；图卷积神经网络；性能建模

# **Inference performance prediction of sparse neural networks**

Computer Science and Technology

Student: LIN Junqing Advisor: DUAN Lei,SUN Guangzhong

**[Abstract]** In order to obtain sparse neural networks with high inference performance, neural network sparsification methods need to evaluate the inference performance of the network. Due to the high deployment cost, the practical process often uses agents to evaluate the inference performance of sparse neural networks. Different optimization strategies in the deployment process can affect the inference performance of sparse neural networks. Therefore, two inter-operator optimization strategies, inter-operator fusion and inter-operator parallelism, are validated and analyzed in the deployment process of the TensorRT inference framework, and a graph structure-based inference performance prediction model (pipSG) for sparse neural networks is proposed. pipSG model includes graph feature extraction and inference performance prediction. The graph feature extraction module uses a graph convolutional neural network to extract features from the model computation graph of the sparse neural network, which can indirectly learn the inter-operator optimization strategy during the deployment of the TensorRT inference framework. The inference performance prediction module uses a multilayer perceptron to predict the inference performance of the sparse neural network. The sparse neural network sets of four neural networks, DenseNet, GoogleNet, VGG16 and VGG19, are obtained using the structured pruning algorithm ABCPruner, and then the corresponding inference latency dataset ILDS is obtained using the TensorRT inference framework on Nvidia 2080TiGPU for their deployment measurements. The validity of the pipSG model is verified, and the influence of the aggregation function and activation function in pipSG on the model accuracy is also analyzed by ablation experiments.

**[KeyWords]** sparse neural network; inference performance; graph convolutional neural network; performance modeling

**目录**

[1 绪论 1](#_Toc102503393)

[1.1 选题背景 1](#_Toc102503394)

[1.2 研究现状 3](#_Toc102503395)

[1.2.1 神经网络稀疏化方法 3](#_Toc102503396)

[1.2.2 神经网络性能建模方法 4](#_Toc102503397)

[1.3 研究问题 5](#_Toc102503398)

[1.4 论文工作 6](#_Toc102503399)

[1.5 论文结构 6](#_Toc102503400)

[2 推理性能影响因素 8](#_Toc102503401)

[2.1 TensorRT部署流程 9](#_Toc102503402)

[2.2 算子间融合 10](#_Toc102503403)

[2.3 算子间并行性 12](#_Toc102503404)

[3 基于图结构的稀疏神经网络推理性能预测模型 15](#_Toc102503405)

[3.1 图特征提取 16](#_Toc102503406)

[3.2 推理性能预测 17](#_Toc102503407)

[4 实验验证 19](#_Toc102503408)

[4.1 实验环境 19](#_Toc102503409)

[4.1.1 硬件设备 19](#_Toc102503410)

[4.1.2 部署工具 19](#_Toc102503411)

[4.1.3 实现相关 19](#_Toc102503412)

[4.2 数据集 19](#_Toc102503413)

[4.3 评价指标 22](#_Toc102503414)

[4.3.1 均方根误差 22](#_Toc102503415)

[4.3.2 平均绝对百分误差 22](#_Toc102503416)

[4.3.3 5% 以内的准确度 23](#_Toc102503417)

[4.3.4 10% 以内的准确度 23](#_Toc102503418)

[4.4 对比方法 23](#_Toc102503419)

[4.4.1 随机森林回归模型 23](#_Toc102503420)

[4.4.2 K近邻回归方法 23](#_Toc102503421)

[4.4.3 线性回归方法 24](#_Toc102503422)

[4.5 pipSG 相关实验 24](#_Toc102503423)

[4.5.1 pipSG 模型的有效性 24](#_Toc102503424)

[4.5.2 聚合函数的选择 26](#_Toc102503425)

[4.5.3 激活函数的选择 26](#_Toc102503426)

[5 总结与展望 28](#_Toc102503427)

[5.1 工作总结 28](#_Toc102503428)

[5.2 未来展望 28](#_Toc102503429)

[参考文献 30](#_Toc102503430)

[声 明 34](#_Toc102503431)

[致 谢 35](#_Toc102503432)

## **1 绪论**

### **1.1 选题背景**

近年来，神经网络方法高速发展，在图像、语音和自然语言等领域都得到了广泛应用。在不断刷新各领域任务纪录的同时，神经网络模型的规模也越来越大，如OpenAI在2020年发布的GPT-3模型的参数量达到了1750亿，Google在2021年发布的switch transformer模型的参数量更是达到1.6万亿。神经网络模型规模的越来越大也导致了其落地越来越困难：大规模的神经网络模型不仅需要更大的存储空间和更高的能源消耗；同时也使得模型的推理性能降低，这对神经网络在实时性要求较高的应用场景（如自动驾驶、实时目标检测等）上的部署造成一定的困难。

为了解决大规模神经网络模型的高功率、高延迟问题，研究学者们关注到模型压缩这一方法，即通过压缩神经网络模型规模来获得具有更低功率和延迟的模型。Denil等人[1] 指出了神经网络模型高达 95%的模型参数可以由剩余5%的模型参数预测出，为模型压缩的可行性提供了理论支持。神经网络稀疏化方法是模型压缩的一类方法，该方法通过移除冗余参数来降低模型的存储和计算开销，在近年来被广泛研究。神经网络稀疏化方法根据对剪枝粒度的不同可以分为结构化剪枝、半结构化剪枝以及非结构化剪枝，其中结构化剪枝的剪枝粒度最大，其剪枝后的神经网络模型仍是规则的，可以在通用的硬件和计算库下获得推理加速；而半结构化剪枝以及非结构化剪枝则需要实现特定的硬件或计算库来获得模型实际的推理加速。早些年前，神经网络稀疏化过程只关注神经网络的规模大小，而忽视了神经网络的推理性能。但是神经网络的低推理性能会使得用户的体验下降，因此近些年来，研究学者们在对模型进行稀疏化时也将神经网络模型的推理性能作为了优化目标之一来指导模型剪枝。本文使用神经网络的推理延迟作为其推理性能，其中延迟越低推理性能越高。

神经网络模型设计者由于没有相关硬件，或者不具备模型部署相关的底层软硬件知识，在实际模型设计时，更多的是采用如参数量、浮点数计算量（FLOPs）等中间指标作为神经网络模型的推理性能评价指标，但是由于模型最终部署到各式各样的终端中，以及不同部署工具对神经网络模型的不同的优化策略等原因，这些中间指标与最终神经网络模型推理性能并不是简单的线性关系，不能很好的表示真实的神经网络模型推理性能。

在神经网络稀疏化过程中获取准确的模型推理性能需要较大的成本。首先，神经网络的设计空间十分庞大，在对神经网络进行稀疏化时需要考虑大量的稀疏神经网络，如对于一个拥有 L 层的神经网络进行结构化剪枝，则它的稀疏模型的假设空间为，其中为神经网络第 j 层的通道数目；其次，为了降低系统波动等原因带来的数据误差，每个稀疏神经网络模型需要运行数十上百次后取推理性能的平均值，这往往需要数秒甚至更多的时间开销；最后稀疏神经网络模型部署的需要了解不同硬件、不同部署工具，这意味着神经网络模型设计者需要一定的学习成本。考虑到以上原因，将稀疏神经网络模型在不同部署工具部署到不同硬件上获得模型实际的推理延迟的成本过于高昂，因此建立稀疏神经网络的推理性能预测模型来代替实际的部署测量过程具有极大的应用价值。在给定的模型结构和特定的硬件或推理框下，稀疏神经网络性能预测模型可以预测出给定的模型结构的推理性能，为神经网络稀疏化过程提供可靠的参考。

构建一个准确的稀疏神经网络推理性能预测模型需要考虑神经网络在部署过程中存在的优化策略，将各类优化策略考虑到预测模型中，从而提高模型的准确度。神经网络模型从设计到应用一般可以分为以下几个阶段：模型构建、训练与验证以及部署。其中神经网络模型可以使用Pytorch、tensorflow等神经网络框架来构建，在这一过程中，神经网络框架会对模型的计算图进行一定的优化，提升模型的推理性能；神经网络构建完成后对模型进行训练与验证使其满足准确性要求。在部署过程中为了满足计算设备的性能要求，需要对训练好的模型进行压缩及优化来降低模型规模和提高模型推理性能。针对不同的硬件设备则需要使用不同的部署工具，如OpenVINO部署工具主要是针对神经网络模型在intel设备，如CPU、GPU上的部署；TFLite部署工具是针对于Tensorflow神经网络模型在边缘设备或手机上的部署；TensorRT推理框架是Nvidia提供的一个针对其相关GPU设备的部署工具。部署工具会对神经网络模型进一步优化，使用多种优化手段使得神经网络在不损失或损失部分精度情况下提高神经网络模型的性能。不同部署工具对应着不同的应用场景，模型部署完之后需要考虑模型的精度和性能是否达到要求，如果没有达到要求，则重复以上步骤。

不同的部署工具对神经网络模型有不同的优化策略。如TensorRT推理框架中存在层间融合、张量融合、量化、核自适应、动态张量内存、多流执行等优化策略。但现有的推理性能预测模型的建模方法往往只考虑了算子间融合的这一优化策略，而忽视了其他的优化策略。而随着设备的计算能力的快速增长，单个算子的计算量往往无法充分利用计算设备的计算能力，算子间并行策略可以调度多个算子并行执行来提高计算设备的利用率，因此算子间并行优化策略也受到了广泛关注，如TensorRT中的层间融合和张量融合将神经网络层之间的算子融合为一个更大算子，多流执行优化可以将多个算子分配到不同流中执行，TensorRT通过这两种方式来实现算子间的并行；MXNet中的依赖引擎将多个算子同时调度到GPU中不同的流执行来实现算子间并行；IOS调度器是Ding等人[2] 提出的一个算子调度器，通过对多算子的自动化调度来实现算子间的并行执行；以及Ma等人提出的RAMMER[3]编译框架将将数据流图中的算子和底层硬件抽象成更小的调度单元和执行单元，在多个执行单元上执行多个调度单元来兼顾了算子内并行和算子间并行。

本文将推理延迟作为稀疏神经网络推理性能的表示，对稀疏神经网络在TensorRT推理框架的推理性能进行建模，在建模过程中考虑TensorRT推理框架中的算子间融合和算子间并行优化策略这两个算子间优化策略。

### **1.2 研究现状**

神经网络稀疏化方法和神经网络性能建模方法这两个领域一直受到研究人员的关注，涌现出了大量的解决方法。接下来，本章对神经网络稀疏化方法和和神经网络性能建模方法的国内外研究现状进行一定的介绍。

#### 游戏画面 低可信度描述已自动生成**1.2.1 神经网络稀疏化方法**

图 1.1 不同粒度的剪枝结构示意图

神经网络稀疏化方法通过移除神经网络的部分连接，在保证模型精度的同时降低神经网络的存储和计算开销，进而降低神经网络的功耗和延迟。根据对剪枝的元素的约束条件，可以将神经网络剪枝方法分为非结构化、结构化和非结构化剪枝，如图1.1所示。其中非结构化剪枝使得神经网络的拓扑结构发生了变化，需要设计特定的算法或硬件来支持这种稀疏的运算；结构化剪枝只改变神经网络的通道数目（滤波器数目）而不改变网络的拓扑结构；半结构化剪枝的剪枝约束处于非结构化剪枝和结构化剪枝之间。

**非结构化剪枝**

非结构化剪枝对剪枝的元素没有任何约束，可以对神经网络中任何位置进行剪枝从而获得更高的稀疏度。早在1990年LeCun 等人[4] 就提出了最优脑优化方法，在对角假设和极限假设下，模型参数的显著性仅利用误差函数关于参数的导数中的二阶项进行近似，在剪枝过程中移除神经网络中显著性低的参数，从而得到一个稀疏神经网络。考虑到对角假设的不适用性，Hassbi等人[5] 针对OBD进行优化提出了最优脑手术方法。非结构化剪枝不仅在全连接神经网络上受到关注，针对卷积和循环神经网络的非结构剪枝方法也受到了广泛关注。Han等人[6]提出使用模型剪枝、训练量化和霍夫曼编码的三阶段流水线式深度压缩方法对卷积神经网络进行压缩。Narang等人[7] 将流水线式方法扩展到了循环神经网络上。Liu等人[8]使用二值掩码对全连接、卷积和循环神经网络进行稀疏化，在训练过程中使用神经元式或过滤式可训练阈值进行动态细粒度剪枝和恢复，同时层级剪枝率随学习率变化。

**结构化剪枝**

非结构化剪枝会给神经网络模型在推理过程带来不规则的内存访问，因此非结构化剪枝在现有的计算库和硬件不能获得很好的推理加速，甚至于起反作用。而结构化剪枝在通道和滤波器的粒度上对模型进行剪枝，剪枝后的神经网络的网络结构仍是规则的，可以使用现有的计算库对剪枝后的稀疏网络进行推理较好的加速。但在粗粒度上的剪枝会减去一些必要的信息，从而使得模型的精度损失较多。Li等人[9]通过计算通道重要性来移除神经网络中的重要性低的通道及其连接的特征图，大大降低了计算成本。然而，Liu等人[10]发现不管是否继承权值，剪枝后的网络都可以达到相同的精度；Lin等人[11]和Liu等人[12]基于这个想法分别提出了ABCPruner和LACP方法，使用进化算法来选择最优的CNN网络结构。由于RNN基本结构相互交织，独立地删除这些结构可能会导致它们的维度不匹配，然后导致无效的循环单位，Wen等人[13]提出了LSTM的ISS结构（内部的本征稀疏结构）及其学习方法，能够高效的加速RNN的推理速度。

**半结构化剪枝**

半结构化剪枝相较于非结构化剪枝具有更加规则化的稀疏结构，能够提高硬件推理速度；同时剪枝粒度较结构化剪枝低，因此能更好的保持模型准确率。但仍需要自己对稀疏网络进行一定的优化以获得较高的推理加速。Yao等人[14]提出了Balanced-Sparsity方法，将参数矩阵均分成多个块，对每个块进行平衡稀疏，最后通过实现特定的 GPU 计算对模型进行推理加速。与Balanced-Sparsity方法中权重组由同一输出通道内部的连续权重组成的不同，Yu等人[15]提出的Cross-channelintragroup（CCI）方法中权重组是跨不同的输出通道所组成的；同时提出了一种能够克服基于CCI-Sparsity(或BalancedSparsity)结构且权值较小的网络训练的困难的方案。Guo等人[16]设计了tile -wise的稀疏模式，在tile层面保持一个规则的模式，以便高效执行，但允许在全局尺度上进行不规则的、任意的剪枝，以保持较高的准确性。

#### **1.2.2 神经网络性能建模方法**

根据神经网络的性能建模粒度，现有的神经网络性能建模方法可以分为：算子级性能建模、层级粒度性能建模和模型级的性能建模。接下来本文将从基于不同粒度对神经网络性能建模方法的研究现状进行介绍。

**算子级性能建模**

算子级性能建模首先对算子的性能进行建模，然后再将模型中所有的算子的性能聚合起来作为模型的性能。Cai等人[17,18]使用算子的类型、输入输出特征图的大小、核大小、卷积步长、膨胀系数等特征来构建算子的性能模型；然后将网络的延迟建模为神经网络架构中各个算子的和。Wu等人[19]为了估计网络结构的延迟，测量搜索空间中每个算子的延迟，构建一个算子延迟的查询表，然后在算法执行的过程中通过查找网络各个算子的延迟将其相加作为网络总的延迟。基于算子的方案假设了各个算子的执行是独立的且顺序的，但是多数情况下，这个假设并不成立。在模型部署过程中，往往会对计算图进行优化，进行算子融合等操作。Zhang等人[20]将网络划分成多个kernel，每个kernel为一个算子或多个算子的融合，预测kernel的延迟性能，最后将每个kernel的性能相加作为网络的性能。

**层级性能建模**

相较于算子级性能建模，层级性能建模可以缓解算子之间的相关性带来的预测误差。Cai等人[21] 首先使用函数建模CNN中的卷积层、全连接层和池化层，然后据此对网络性能进行估计。Yang等人[22]通过构建带有预先测量的每一层资源消耗的分层查找表；在执行算法时查找每一层的表，将每一层的度量值相加估计网络的资源消耗。

**模型级性能建模**

与将网络性能看成多个模块性能之和不同，模型级性能建模通过构建网络在不同平台上的性能模型来预测性能，可以有效降低在其他粒度下性能建模中的由算子融合、编译器优化等因素带来的误差，可以更好的预测网络在不同平台上的性能。Yao等人[23] 使用网络结构配置来表示网络，构造了树形线性回归模型来预测神经网络结构性能，使用特征向量（执行时间非线性的原因）来构建二叉树，然后使用解释向量来线性拟合性能。Dudziak等人[24] 提出了BRP-NAS方法，使用GNN来抽取网络的特征，节点特征包括算子类型、节点类型，然后使用这些特征来预测神经网络性能。Ganesan[25]等人将网络的各层特征进行拼接作为整个网络的特征来进行性能建模。Gao等人[26]提出的DNNPerf不仅捕获算子级特征，还捕获硬件内的隐藏因素作为图节点和边的特征，然后使用基于注意力机制的节点-边编码器对计算图中的节点进行编码，然后聚合所有节点特征作为神经网络计算图的特征表达，随后使用预测模型对神经网络模型的性能进行预测。

### **1.3 研究问题**

本文主要对如何对神经网络结构化稀疏方法后的模型推理性能进行建模这一问题进行研究。接下来本文针对该问题存在着的挑战和解决方案进行介绍：

（1）如何获取准确的稀疏神经网络推理延迟。神经网络的推理延迟容易受到多方面因素的影响，如系统波动、GPU冷启动等。本文收集稀疏神经网络的推理延迟时通过冷启动和多次测量取平均这两种方法来减少收集的模型推理延迟存在的误差。

（2）如何表达稀疏神经网络。神经网络推理性能建模方法基于不同粒度可以分成算子级性能建模、层级性能建模和模型级性能建模。其中算子级性能建模将神经网络看作一系列算子的并集，将所有算子的推理延迟之和作为模型的总延迟；层级性能建模将神经网络看作网络层的并集；大部分模型级的性能建模使用神经网络的总体特征，如参数量、FLOPs、层数等来表达神经网络。但是这些表达方式忽视了算子间的拓扑结构，不能对算子间的推理优化策略进行准确的建模。而神经网络模型的计算图能够反应各个算子之间的拓扑顺序，同时计算图中的节点还包含各个算子的特征信息，因此本文使用稀疏神经网络模型的计算图来表达稀疏神经网络模型。

（3）如何提取稀疏神经网络模型计算图特征。传统的深度学习方法在语音、文本、图像等欧几里得结构数据上已经取得了极大的成功，但是对于计算图这类图数据并不能有效的进行处理。图数据是非欧几里得结构，图上的节点和边的数目是易变的，同时每个节点的相邻节点的数量也是不同的，这使得针对欧几里得结构数据进行学习的神经网络模型在图数据上的应用遇到了困难。研究人员扩展了传统深度学习的方法使得深度学习能够应用到图数据上，设计了能够处理图数据的图神经网络，如图卷积神经网络、图注意力神经网络等。本文使用图卷积神经网络来处理稀疏神经网络模型的计算图。

（4）如何在推理性能建模过程中考虑TensorRT推理框架中的算子间优化策略。本文使用图神经网络在对模型计算图特征进行提取的过程中隐式的学习TensorRT推理框架中的算子间优化策略。

### **1.4 论文工作**

本文的主要工作和贡献如下：

（1）分析了TensorRT推理框架在进行部署过程中存在的模型推理性能影响因素：介绍了TensorRT推理框架的部署流程，以及对其在部署过程中存在的算子间融合和算子间并行这两个算子间优化策略进行了分析与验证。

（2）提出了在TensorRT推理框架下基于图结构的稀疏神经网络推理延性能预测模型（Prediction model of inference performance of sparse neural network based on graph structure，pipSG），使用图卷积神经网络提取稀疏神经网络模型的特征表达，然后使用多感知机来预测稀疏神经网络模型的推理延迟。图卷积神经网络通过对稀疏神经网络的模型计算图表示进行学习，其在计算图上的相邻节点能够隐式的学习TensorRT推理框架对神经网络模型中算子间的优化策略。

（3）通过实验验证了pipSG模型的有效性，使用稀疏神经网络模型计算图来表达稀疏神经网络模型的相较于使用总体特征有着更好的表现，同时通过消融实验分析pipSG模型中所使用的聚合函数和激活函数。

### **1.5** 论文结构

本文共有五章，论文组织架构如下：

第1章首先介绍了论文的选题背景和研究现状，然后阐述了论文的研究问题、工作内容，最后对论文的组织架构进行了一定的说明。

第2章介绍了稀疏神经网络使用TensorRT推理框架进行部署存在的模型推理性能影响因素（TensorRT推理框架的部署流程以及其在部署过程中存在的算子间优化策略）；

第3章针对TensorRT推理框架的算子间优化策略提出了基于图结构的稀疏神经网络推理延性能预测模型pipSG；

第4章介绍了论文使用的数据集ILDS的收集流程，在该数据集上验证了pipSG模型的有效性以及pipSG模型中部分组件对模型性能的影响；

第5章对当前工作进行了总结与展望。

## **2 推理性能影响因素**

了解对稀疏神经网络的推理延迟产生影响的因素（如部署工具的部署流程以及其在部署过程中所使用的优化策略等），能够对稀疏神经网络的推理性能进行更好的建模。因此本文在推理性能预测模型构建过程中通过考虑推理性能影响因素来进一步提高模型的准确性。

图示, 日程表

描述已自动生成本章首先介绍了稀疏神经网络模型在TensorRT推理框架下的部署流程，了解稀疏神经网络的部署过程中优化发生的步骤。然后对TensorRT推理框架中的算子间融合和算子间并行两个优化策略进行分析与验证，以Inception模型（Inception模型图如图2.1所示）为例对TensorRT推理框架的部署过程中存在的算子间优化策略进行介绍与分析；同时构造特定的神经网络模型结构数据集来验证TensorRT推理框架是否存在算子间融合和算子间并行这两种优化策略。由于不同部署工具在神经网络部署过程中使用了不同的优化策略，在考虑如何对稀疏神经网络的推理性能进行建模时对特定的部署工具的优化策略进行验证和分析，针对特定部署工具中的优化策略进行建模，能够使得在该部署工具下稀疏神经网络推理性能预测模型拥有更高的准确性。

图 2.1 Inception 模型图

### **2.1 TensorRT部署流程**

TensorRT推理框架的部署流程如图2.2所示，可以分成两步：第一步为优化完成训练的模型，首先获得一个训练好的神经网络模型文件，然后将其通过解析等方式输入到TensorRT优化器中做解析得到一系列推理引擎，每个引擎对应着一个核函数；然后对所得到的引擎进行优化，并将其进行序列化到内存或文件得到优化后的流图。第二步为使用Runtime部署优化的流图，模型推理过程中读优化的流图时并将其反序列化得到TensorRT Runtime引擎以供使用；然后在神经模型推理执行时需要创建上下文，分配预先的资源，使用引擎和上下文对神经模型进行推理。其中TensorRT推理框架对神经网络模型的部署优化主要发生在TensorRT优化器中的优化步骤和Runtime部署过程中，如层间融合、张量融合、量化等发生在TensorRT优化器中的优化步骤中，降低单个引擎的推理延迟；多流执行可以发生在Runtime部署过程中，通过同时调用多个引擎来降低神经网络模型总的推理时间。

图 2.2 TensorRT推理框架工作流程

### 图示 描述已自动生成**2.2 算子间融合**

图 2.3 Inception 算子融合后结构可能情况

核函数（kernel function）为GPU上运行的基本单位。CPU通过调用核函数来实现对 GPU资源的利用。但是每次调用核函数时都存在一定的额外开销，如对数据流图进行调度、在CPU与GPU之间进行数据传输以及启动核函数等操作。将稀疏神经网络模型中每个算子对应一个核函数，这会使得稀疏神经网络在推理过程中频繁调用核函数，从而带来大量的额外开销使得稀疏神经网络模型的推理性能降低。如果将多个算子融合成一个核函数来降低推理过程中核函数的调用次数则可以有效的缓解核函数频繁调用所带来的性能开销（如图2.1中conv算子、bn算子以及relu激活函数，在大部分部署工具中可以通过算子间融合优化策略将其合并成一个CBR结构，也就是说，它们在推理过程中可以从之前的三次核函数调用，缩减为融合后的一次调用），图2.3为Inception进行算子融合后可能的模型结构图，其从先前的19次调用降到了7次调用。除了将多个算子融合成一个大算子来进行算子间的融合外，部署工具对神经网络模型进行部署时还可能将算子划分成更小的执行单元，通过调用多个小的执行单元来间接实现算子间的融合，如Ma等人提出的 RAMMER[3] 编译框架将算子与计算资源都划分成更小的执行单元，从而在调用过程中间接实现算子间融合和算子间并行。

为了提高模型的推理性能，绝大多数的部署工具在对神经网络模型进行部署时会进行算子间融合优化，但是由于不同部署工具针对不同的设备，其采取算子间融合方式可能不同。同时大多数部署工具是闭源的，我们很难从官方文档中获取部署工具中具体算子间融合方式，因此需要通过其他手段来对具体的算子间融合方式进行验证与分析，如 nn-meter 通过比较两个算子单独运行与一起运行下的推理延迟来判断这两个算子在该硬件（部署工具）下能否进行融合，但这种方法无法获取如RAMMER实现的算子间融合的实现方式。接下来本文首先对TensorRT推理框架的算子间融合优化策略的存在性进行验证。

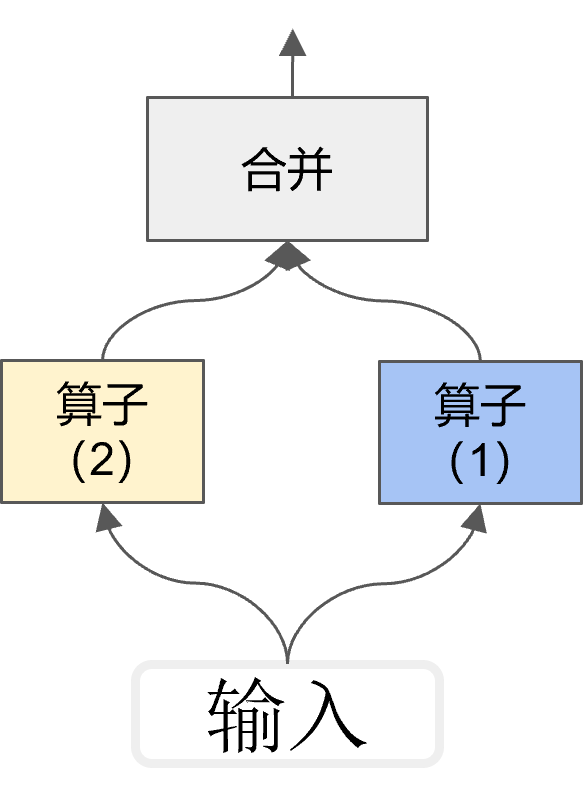
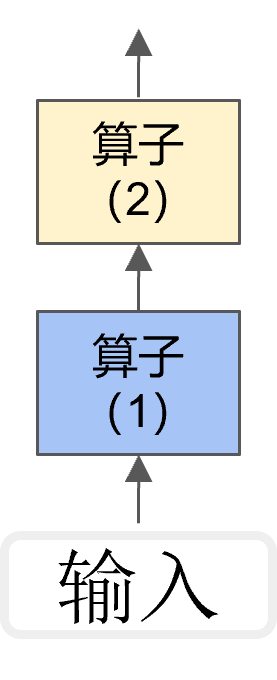


图 2.4 算子间融合实验图 图 2.5 算子间并行实验图

为了对TensorRT推理框架中算子间融合这一优化策略的存在性进行验证。本文首先构建一个如图2.4所示的神经网络模型类，然后使用不同的算子参数（如算子类型、输入输出通道数目、核大小、步长等）来实例化这个神经网络类后得到一系列拥有两个算子的神经网络模型，最后分别测量组成的神经网络模型的推理延迟和两个组成算子的推理延迟yall、y1、y2 得到对应算子间融合验证数据集。以神经网络组成算子之间较低的推理延迟（*min*(*y*1*,y*2)）为横轴，算子间融合后降低的推理延迟（*y*1+*y*2−*y*all）为纵轴作散点图，结果如图2.6所示，当两个算子的融合程度高时，即具有较低推理延迟的算子将完全融合到具有较高推理延迟的算子中，即降低的推理延迟应该与两个算子中较低的推理延迟相近。该实验中，本文针对了conv2d与conv2d、batchnormalize、maxpool2d、avgpool2d、relu五种算子组合的算子间融合进行研究分析，每种组合选取500个随机样本。

图表

低可信度描述已自动生成

图 2.6 算子间融合实验散点图

从图中我们可以看出，con2d算子在batchnormalize算子和relu算子两种组合下的散点图中散点在直线y=x附近，即降低的推理延迟与组成算子之间较低的推理延迟相近，这说明了conv2d算子与batchnormalize算子和relu算子的融合程度高。而conv2d算子与con2d算在、maxpool2d算子和avgpool2d算子三种组合下，降低的推理延迟在y=0直线附近，即组成的神经网络模型几乎没有降低推理延迟，也就是说conv2d算子与conv2d算子、maxpool2d算子、avgpool2d算子之间不能进行有效的融合。由此我们可以得出结论，TensorRT推理框架中存在算子间融合这一优化策略，对其具体的算子间融合方式，本文将通过图卷积神经网络来间接的学习。

### **2.3 算子间并行性**

神经网络模型在推理过程中不仅在算子内存在并行性，而且在算子间也存在并行性，如图2.3所示，3\*3CBR算子与5\*5CBR算子可以在1\*1CBR算子执行后并行执行而不影响神经网络模型最终的输出结果，因此神经网络模型在部署过程中可以充分挖掘神经网络模型中算子间存在的并行性，从而提高神经网络模型的推理性能。

神经网络模型的单个算子的计算量往往不能充分利用GPU上的计算资源，容易导致GPU计算资源的浪费。为了更好地利用GPU资源和提高神经网络模型的推理性能，近些年来，在神经网络模型的推理过程中同时运行多个算子这项技术（即算子间并行）得到了一定的研究。如TensorRT推理框架中的层间融合、多流执行优化和kernel自适应调用等部署优化策略可以同时执行将神经网络模型中的多个算子，充分利用GPU的计算资源；MXNet中的依赖引擎可以使用GPU中的多流执行来调度多个算子实现算子间并行；Ding等人[2]提出的调度器IOS，对多个算子自动化并行调度实现算子间并行；Ma等人提出的RAMMER[3]编译框架将计算设备和算子划分成更小的单位，在推理过程中并行调用更小的执行单元实现算子间并行。

图表, 散点图

描述已自动生成同算子间融合部分类似，本文针对TensorRT推理框架的算子并行性优化的存在性也进行研究，为本文提出的推理性能预测模型提供一定的理论依据。本文首先构建如图2.5所示的神经网络模型类，算子1和算子2在推理过程中可以并行执行。使用不同的算子参数来构建不同的神经网络模型（算子1和算子2都为conv2d算子），然后分别对神经网络模型的推理延迟以及其中的两个算子的推理延迟进行测量得到推理延迟 yall、*y*1、*y*2 得到算子间并行验证数据集。以神经网络组成算子之间较高的推理延迟（max(y1,y2)）横轴，组成的神经网络模型的推理延迟（yall）为纵轴做散点图，结果如图2.7所示。

图 2.7 算子间并行实验散点图

如果两个算子在推理过程的并行程度高，即两个组成算子可以高效的并行执行，具有较低推理延迟的算子的推理时间被具有较高推理延迟的算子的推理时间覆盖，组成的神经网络模型的推理延迟将与组成算子之间较高的推理的延迟相近，散点位于直线y=x附近。从图中我们可以发现存在部分散点位于 y=x附近，即神经网络模型的推理延迟与组成算子中较高的推理延迟相近，也就是说TensorRT推理框架在对多个算子进行调度时会考虑到算子之间的并行性，尽可能的将多个算子并行执行来提高神经网络模型的推理性能。从图中可以看出距离直线y=x较远的散点的横坐标（即组成算子中较高的推理延迟）较大，这是由于算子的推理延迟越高意味着其需要更多的计算资源，而硬件设备的计算资源是有限的，当多个算子的计算需求过大时往往无法进行有效的并行，因此在推理性能预测模型构建过程中，不能简单的在存在并行性的多个算子中选取最大的算子推理延迟作为多个算子的推理延迟，还应该考虑算子的计算需求、算子规模等条件来判断算子间的并行程度。

## **3 基于图结构的稀疏神经网络推理性能预测模型**

由于TensorRT推理框架不是开源的，基于算子的推理性能预测模型在建模过程中难以对TensorRT推理框架中存在的算子间优化策略（如算子间融合、算子间并行等）考虑周全；如nn-meter方法仅考虑了算子间融合对神经网络模型的推理性能的影响，而忽视了其他的算子间优化策略所带来的影响。现有的模型级性能建模方法往往使用神经网络的层数、参数量等神经网络模型的总体特征来对神经网络的性能进行建模，但是这些总体特征难以表达神经网络模型中算子及其之间的拓扑关系，仅使用这些总体特征无法对TensorRT推理框架在部署过程中的算子间的优化策略进行学习。

文本

中度可信度描述已自动生成为了对TensorRT推理框架中的算子间优化策略进行学习表达，本文提出了基于图结构的稀疏神经网络推理性能预测模型（pipSG模型），pipSG模型框架图如图3.1所示。pipSG 模型是一个模型级的稀疏神经网络推理性能建模方法。由前文所述，TensorRT推理框架中存在算子间融合和算子间并行这两种算子间优化策略，因此在构建针对TensorRT推理框架的稀疏神经网络模型的推理性能预测模型时需要考虑对算子特征及算子间拓扑关系进行表达。稀疏神经网络模型的计算图中每个节点表示一个算子的调用，节点之间的边表示算子之间的执行依赖，也就是说计算图包含了稀疏神经网络模型中各个算子的特征描述以及算子之间的拓扑关系，因此pipSG模型使用稀疏神经网络的计算图作为特征来对稀疏神经网络的推理性能进行建模。

图 3.1 pipSG 模型框架图

但是图数据是非欧几里得数据，其大小形状是可变的，传统的深度学习方法难以对图数据进行有效的处理。由于图神经网络将深度学习方法扩展到了图数据上，pipSG 模型使用图卷积神经网络来处理稀疏神经网络模型计算图，提取各个算子的特征作表达，然后聚合将所有算子的特征表达作为稀疏神经网络模型计算图的特征表达，间接地学习TensorRT推理框架在部署过程中存在的算子间优化策略；然后将计算图的特征表达作为多层感知机的输入特征来预测稀疏神经网络模型的推理延迟。

pipSG模型包含两个模块：图特征提取模块和推理性能预测模块，接下来本文对pipSG模型中图特征提取模块和推理性能预测模块进行一一介绍。

### 图片包含 图形用户界面 描述已自动生成**3.1 图特征提取**

图3.2 图卷积神经网络示意图

pipSG模型中图特征提取模块使用图卷积神经网络[27] 提取稀疏神经网络模型计算图的特征表达，图卷积神经网络示意图如图3.2所示。图卷积神经网络将深度学习中的卷积操作扩展到了图数据中，能够高效的处理图数据，通过聚合节点的相邻节点的特征来学习图的拓扑结构，在图节点分类与预测、整图分类等任务上都取得很好的效果。图卷积神经网络通过聚合相邻节点的特征表示将节点之间的拓扑信息融合到当前节点的特征表示中，该过程可以形象地表达成

其中H(l+1)、H(l)分别表示第l+1层、第l层的隐藏单元，A为节点的连接关系，W为卷积权重，*σ*为激活函数，f为映射函数。

图卷积神经网络的推理步骤可以分成三步：第一步是对节点自身特征信息进行抽取变换，然后将其发送给相邻节点，进行节点间通信；第二步是接收来自相邻节点的特征信息，然后使用卷积操作聚合所有相邻节点的特征信息（及当前节点的特征信息），将节点局部结构信息融合到当前节点的特征表达中；第三步将非线性变换作用到节点的特征信息来提高模型的表达能力。TensorRT推理框架基于算子的局部空间结构进行算子间融合优化，通过考虑具有依赖关系（即连接关系）的多个算子之间是否能够进行融合，从而将多个算子融合成一个算子进行执行；稀疏神经网络模型计算图是一个有向无环图，其边的连接关系表示了算子之间的依赖关系，通过对计算图的节点添加自环，图卷积神经网络在推理步骤中第二步可以聚合当前算子与其相邻算子的节点特征来对算子间融合这一优化策略进行一定的表示，从而学习TensorRT推理框架中对算子间融合的优化策略。TensorRT推理框架同样基于算子的局部空间结构进行算子间并行优化，算子间并行将不存在依赖关系的多个算子之间并行执行来提高模型的推理性能，图卷积神经网络在进行特征聚合是会将当前算子节点特征所有相邻算子节点（即所有的前序算子节点）的特征进行聚合，而算子间的并行性一般发生在算子的前序算子之间，因此图卷积神经网络可以对TensorRT推理框架中的算子间并行进行表达与学习；同时图卷积神经网络通过对当前节点的特征信息进行抽取变化和非线性变化来表达学习，能够对算子内部优化，如量化等进行学习。因此，图卷积神经网络使用稀疏神经网络模型的计算图对网络模型在TensorRT推理框架的推理过程进行很好的特征表达，从而对下游任务（即稀疏神经网络模型推理性能的预测）有着极大的帮助。

### 背景图案 描述已自动生成**3.2 推理性能预测**

图 3.3 多层感知机示意图

pipSG模型中推理延迟预测模块中使用多层感知机对稀疏神经网络模型的推理性能进行预测。多层感知机由输入层、输出层以及至少一层的隐藏层三部分组成，其中层与层之间是全连接的，如图3.3所示；除了输出层外，每一层的输出都作为下一层的输入。多层感知机可以在除输入层外的每个节点使用非线性激活函数（如ReLU、Sigmoid、Tanh和LeakyReLU等）来提高模型的非线性表达能力。pipSG模型以图特征提取模块中得到稀疏神经网络模型计算图的特征表达作为多层感知机的输入，然后输出稀疏神经网络模型的推理延迟。

## **4 实验验证**

本章验证了pipSG模型的有效性以及分析了聚合函数和激活函数对预测模型准确度的影响。接下来本章将从实验环境、数据集、评价指标、对比方法和相关实验设计五个方面进行叙述。

### **4.1 实验环境**

#### **4.1.1 硬件设备**

**Nvidia GeForce RTX 2080 Ti**： 2080 Ti使用了图灵架构以及 GDDR6显存（拥有11GB 容量），具有68个流处理器、4352 个CUDA核以及544个张量核。其具有较高的计算能力，其单精度浮点计算性能达到了13.4/14.2TFLPOS，因此在稀疏神经网络部署过程中需要充分利用稀疏神经网络模型中的并行性来获得更高的推理性能。

#### **4.1.2 部署工具**

**TensorRT 推理框架**：TensorRT推理框架是Nvidia提供的一个高性能的深度神经网络模型部署工具。TensorRT推理框架为神经网络模型在Nvidia各种GPU硬件平台进行部署，对神经网络模型的部署过程中提供多个优化策略，如层间融合和张量融合、量化、kernel自适应、多流执行等，从而使得神经网络的推理延迟更低、吞吐量更高。本文使用Pytorch深度学习框架下对稀疏神经网络模型进行定义，在对稀疏神经网络模型的推理延迟进行策略前需要使用 torch2trt开源工具将torch模型转换成TensorRT推理框架所需要的trtmodule模型格式，然后在TensorRT推理框架下进行推理部署。

#### **4.1.3 实现相关**

**Pytorch 深度学习框架**：Pytorch是一个具有神经网络模型定义、训练以及模型保存和加载等功能的深度学习框架。Pytorch相较于其他深度学习框架更加简洁，其源码只有Tensoflow的1/10左右。本文使用Pytorch对稀疏神经网络以及图神经网络进行定义与训练。

**DGL 框架**：DGL 是一个用于实现图神经网络的 Python 开源工具包，能够在Pytorch、tensorflow等深度学习框架上对图神经网络进行构建与训练。本文使用DGL框架来对所使用的图卷积神经网络进行构建与训练。

### **4.2 数据集**

本文在Nvidia GeForce RTX 2080 Ti机器上收集神经网络模型在稀疏方法中产生的稀疏神经网络模型通过TensorRT推理框架部署的推理延迟，数据集收集流程如图4.1所示。本文所采用的稀疏算法为ABCPruner[11]，ABCPruner是一种结构化剪枝的算法，其通过选择不同层之间的通道数目作为神经网络的剪枝过程。因此本文通过对各层选择不同通道数目来模拟剪枝过程，首先对于每个神经网络类型定义一个Pytorch神经网络模型类，然后通过不同参数来实例化这个神经网络模型类得到稀疏神经网络模型数据集，使用torch2trt开源工具将得到的torch模型转化成trtmodule格式，最后使用TensorRT推理框架进行部署，机器上在测量稀疏神经网络模型的推理延迟。

图 4.1 数据集收集流程

本文考虑了三种类型的神经网络进行稀疏化，分别为GoogleNet、DenseNet、VGG，其中VGG考虑VGG16和VGG19两个变体。针对这四种神经网络模型，本文共收集了 16000个稀疏神经网络模型的推理性能样本，其中每种神经网络模型各4000个样本。在测量过程中以大小为（32,3,32,32）的张量作为稀疏神经网络模型的输入来测量稀疏神经网络模型的推理延迟。为了减少模型在推理执行过程由于硬件、系统波动等因素带来的误差，在测量过程中对每个稀疏神经网络模型执行200次的推理过程，然后取平均值作为该稀疏神经网络模型的推理延迟。GPU在不使用并且不在持续模式（persistence mode）时会自动将其能量状态降低到一个非常低的水平，这会给测量稀疏神经网络的推理延迟带来极大的影响，即GPU 的冷启动问题。为了减少GPU冷启动带来的推理延迟测量误差，本文在收集稀疏神经网络推理延迟数据时会先执行50次的推理过程来预热（warm up）GPU硬件。

表 4.1 算子特征

|  |  |  |
| --- | --- | --- |
| 特征 | 特征含义 | 取值范围 |
| op\_type | 算子的类型 | [Conv, BatchNormalization  MaxPool, AveragePool, Concat  Relu, Pad, Reshape, Gemm] |
| in\_channel | 输入通道数目 | [1, 1000] |
| out\_channel | 输出通道数目 | [1，1000] |
| kernel\_size | kernel 大小 | [1, 3, 5] |
| stride | 步长 | [1, 2, 3] |
| pads | 补位 | [0, 1, 2, 3] |
| input\_shape | 输入特征图形状 | 维度为 4 的张量 |
| output\_shape | 输出特征图形状 | 维度为 4 的张量 |
| parms | 参数量 | [0, +∞] |
| FLOPs | 浮点计算数 | [0, +∞] |

为了获取稀疏神经网络模型的计算图，本文首先将每个稀疏神经网络模型从torch格式转换成onnx格式，然后使用开源工具nn-meter中的模型计算图提取转化工具来提取 onnx格式的稀疏神经网络模型的计算图，将其保存成json文件，json文件中包含各个算子的特征描述以及各个算子之间的连接关系。nn-meter工具会将空节点自动删除，对计算图进行一定的删减。在对图神经网络进行训练之前，需要将存储在json文件中的计算图转换成DGL中使用的数据集，同时为每个节点加上自环。本文在json文件中提取op\_type、in\_channel、out\_channel、kernel\_size、stride、pads、input\_shape、output\_shape、parms以及 FLOPs 作为稀疏神经网络模型计算图中算子节点的特征。各个特征的含义及其取值范围如表4.1所示。同时本文收集了稀疏神经网络模型的总体特征（如表4.2所示，神经网络模型的层数、模型的浮点计算数、参数量、激活函数数目、卷积层数目、归一化层数目、全连接层数目、神经元数目），用于对比方法的训练。

通过以上步骤本文得到了在TensorRT推理框架上关于稀疏神经网络模型的推理延迟数据集（inference latency dataset for sparse neural network model，ILDS）。本文将ILDS数据集按照7：3的比例划分成训练集和测试集。

表 4.2 网络模型总体特征

|  |  |
| --- | --- |
| 特征 | 含义 |
| Layers | 神经网络模型的层数 |
| FLOPs | 浮点计算数 |
| Params | 参数量 |
| acs | 激活函数数目 |
| convs | 卷积层数目 |
| bns | 归一化层数目 |
| fcs | 全连接层数目 |
| nerouns | 神经元数目 |

### **4.3 评价指标**

本文使用均方根误差、平均百分比误差、5%以内的准确度以及 10%以内的准确度四种评价指标对各类预测模型的精确度进行度量。

#### **4.3.1 均方根误差**

均方根误差是指预测值与真实值之差平方的期望值的算术平方根，即均方误差的平方根，缩写为RMSE。均方根误差越小，说明预测构建的模型拥有更好的准确度。RMSE计算公式为：

#### **4.3.2 平均绝对百分误差**

平均绝对百分误差是指预测值与真实值之差与真实值的比（即相对误差）的期望值，其缩写为MAPE。MAPE的范围是[0,+∞)，同RMSE相同，其值越小表示该模型的预测性能越好。MAPE 计算公式为：

#### **4.3.3 5% 以内的准确度**

5%以内的准确度（±5%）统计了预测值与真实值的误差在 5%以内的准确度。其计算公式为：

，其中)为指示函数，当时为1，时为0。

#### **4.3.4 10% 以内的准确度**

10%以内的准确度（±10%）与±5%类似，不同的是±10%统计的是预测值的误差在

10%以内的准确度。其计算公式为：

，其中)为指示函数，当时为1，时为0。

### **4.4 对比方法**

#### **4.4.1 随机森林回归模型**

随机森林回归模型是一种bagging集成方法，本文使用CART回归树作为随机森林回归模型的基估计器。CART回归树使用超平面将特征空间划分成多个子空间，每个子空间对应一个特定输出，回归树的每个节点对应着“是”和“否”，将空间一分为二。最后多个回归树的输出决定随机森林回归模型的输出。随机森林中的随机性体现在两方面：一方面是样本的随机性，随机森林在构建回归树的过程中从训练集随机抽取一定数量的样本进行构建，而非所有数据；另一方面是特征的随机性，在构建过程中随机选取部分特征作为候选特征对回归树进行构建。本文使用sklearn开源工具包来实现随机森林回归，其中设置回归树数量为100，随机种子为42。同时本文选取稀疏神经网络的总体特征作为随机森林回归模型的特征，具体见表4.2。

#### **4.4.2 K近邻回归方法**

K近邻回归方法步骤为：对新输入的实例，在给定的训练集中寻找距离其最近的K个点，然后取这K个实例的标签平均值作为新输入的实例的标签。K 近邻回归的优点在于其不用学习，只需要在预测时查找相邻实例的值即可。K 近邻回归方法中最重要的是选取合适的K值和距离的度量方式，K值越大意味着模型越简单，如当K取N（N为数据集规模）时，模型输出数据集中所有样本的平均值，K值越小意味预测所使用的领域越小，模型容易过拟合；常用的距离度量方式有：欧式距离、曼哈顿距离、余弦距离等，使用不同的度量方式会对模型的精度产生影响。通过实验，本文选择5作为K值，采用欧式距离作为度量方式。同时K近邻回归模型的特征与随机森林相同也为稀疏神经网络的总体特征，即表4.2。

#### **4.4.3 线性回归方法**

线性回归方法使用线性回归函数对自变量和因变量之间关系进行建模。其目标是找到一组参数使得函数与真实值y尽可能接近，即找到一个超平面来拟合函数f(x)。线性回归函数可以使用最小二乘法或梯度下降法进行拟合。本文采用最小二乘法来对线性回归模型进行学习。线性回归方法中所使用的特征为稀疏神经网络模型的总体特征，即表4.2。

### **4.5 pipSG 相关实验**

pipSG模型使用了一个三层的图卷积神经网络模型作为图特征提取模块，其中图卷积神经网络中第一层是具有24个输入通道和512个输出通道的图卷积层，第二层和第三层是输入输出通道都为512的图卷积层，其中使用ReLU函数作为图卷积层的激活函数；最后聚合所有节点特征作为稀疏神经网络计算图的特征表示（本文使用sum聚合函数将所有节点特征求和最为计算图的特征表示）；使用具有一层隐藏层的多层感知机作为推理延迟预测模块，其中输入层为经过图特征提取模块后得到的512维的计算图特征，隐藏层具有 512个神经元，输出层为稀疏神经网络的推理延迟，选取sigmoid函数多层感知机作为激活函数。

在模型训练之前，需要对图数据进行预处理。首先将json文件转换成图数据，然后对每个节点上的特征进行一定的处理：枚举型数据进行独热编码、数据进行归一化处理。枚举型数据不能直接用于训练过程中，因此在训练前需要对枚举型数据进行独热编码；同时由于节点特征的取值范围不同以及存在不同的数据类型，若将其直接输入图卷积神经网络进行特征提取，会导致某一维特征的数值爆炸从而影响模型的预测效果，如参数量的取值往往是输出通道的数万倍，因此需要对计算图中的节点特征进行归一化处理。但是对每个计算图单独归一化会导致不同计算图中节点特征不相匹配，因此本文使用所有计算图的所有计算节点特征来训练一个归一化模型，然后再使用该归一化模型对每个计算图中节点特征进行归一化。本文使用标准差标准化归一化方法（StandardScaler）来对计算图的节点特征进行归一化。

pipSG模型是一种端到端的方案，将图特征提取模块与推理延迟预测模块作为一个整体进行优化与训练。在训练过程中使用MSE作为损失函数评估模型的训练误差，使用adam优化器对模型参数进行更新，以学习率为0.0003、batchsize为64对pipSG模型训练了300 次迭代后得到训练模型。

#### **4.5.1 pipSG 模型的有效性**

本文将 pipSG 模型与随机森林回归、KNN 回归、线性回归三种方法进行比较，分别在四种稀疏神经网络集以及它们的合集五种情况进行实验分析。结果如表4.3所示。

从表4.3中我们可以看出在稀疏神经网络合集中，pipSG模型在5%误差下的准确度为79%，比使用总体特征中最好的算法随机森林回归高了9.7%，同时pipSG模型在五种**表 4.3 不同方法预测结果**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 指标 | pipSG | RFR | KNN | LR |
| AllNet | RMSE(ms) | **0.069** | 0.116 | 0.123 | 0.143 |
| MAPE(%) | **2.758** | 4.365 | 4.616 | 5.380 |
| ± 5% | **0.790** | 0.693 | 0.678 | 0.598 |
| ± 10% | **0.970** | 0.868 | 0.856 | 0.813 |
| DenseNet | RMSE(ms) | **0.043** | 0.070 | 0.077 | 0.063 |
| MAPE(%) | **1.194** | 1.902 | 2.056 | 1.686 |
| ± 5% | **0.999** | 0.967 | 0.953 | 0.977 |
| ± 10% | 1.000 | 1.000 | 0.998 | 1.000 |
| GoogleNet | RMSE(ms) | **0.033** | 0.051 | 0.055 | 0.046 |
| MAPE(%) | **0.450** | 0.695 | 0.753 | 0.623 |
| ± 5% | 1.000 | 1.000 | 1.000 | 1.000 |
| ± 10% | 1.000 | 1.000 | 1.000 | 1.000 |
| VGG16 | RMSE(ms) | **0.078** | 0.130 | 0.136 | 0.127 |
| MAPE(%) | **4.460** | 6.945 | 7.312 | 6.986 |
| ± 5% | **0.634** | 0.444 | 0.433 | 0.434 |
| ± 10% | **0.931** | 0.771 | 0.751 | 0.770 |
| VGG19 | RMSE(ms) | **0.091** | 0.148 | 0.156 | 0.144 |
| MAPE(%) | **4.293** | 6.538 | 6.838 | 6.328 |
| ± 5% | **0.632** | 0.473 | 0.447 | 0.483 |
| ± 10% | **0.942** | 0.793 | 0.775 | 0.812 |

情况下都取得最好的推理延迟预测结果。这是由于图卷积神经网络在提取神经网络特征的过程中综合考虑了TensorRT推理框架在稀疏神经网络部署过程中使用的算子间融合以及算子间并行等算子间的优化策略，而随机森林回归等方法只是使用了神经网络的总体特征，无法表示算子间的拓扑关系，从而无法反映算子间的优化策略。

同时观察结果可以看出，所有方法在GoogleNet和DenseNet上的结果比VGG16和VGG19好。出现该结果的可能的原因是GoogleNet和DenseNet的模型规模较大导致其推理延迟较高，其中GoogleNet的推理延迟在5ms以上，DenseNet的推理延迟在3ms之上，从而使得模型的推理延迟更有区分度；而VGG16和VGG19的模型规模较小，其推理延迟在2ms之下，导致了其区分度较小，从而使得其结果较差，但pipSG模型在10%的误差下也分别取得了 93.1%和94.2%的准确度。

#### **4.5.2 聚合函数的选择**

为了得到稀疏神经网络模型计算图的特征表示需要对所有节点特征进行聚合，选择合适的聚合函数可以使得模型拥有更高的精度。常见的聚合函数有sum、mean、max三种，其中sum 聚合函数将所有节点特征求和，mean聚合函数将所有节点特征求和后取平均，max聚合函数取所有节点特征的每一维的最大值。本节对三种特征聚合函数进行了分析，实验结果如表4.4所示，sum聚合函数情况下在指标 RMSE、MAPE 和 10%误差下的准确度都比另外两个聚合函数好，在5%的误差下的准确度也与mean聚合函数相近。sum聚合函数将所有算子的节点特征求和作为稀疏神经网络计算图的特征表示，这个步骤与将所有算子的延迟相加作为网络模型的延迟相对应，因此相较于其他两种聚合函数的结果更好。本文选取 sum 聚合函数来聚合计算图中的节点特征。

表 4.4 不同聚合函数的实验结果

|  |  |  |  |
| --- | --- | --- | --- |
|  | sum | mean | max |
| RMSE(ms) | **0.066** | 0.069 | 0.093 |
| MAPE(%) | **2.622** | 2.695 | 3.552 |
| ± 5% | 0.819 | **0.822** | 0.751 |
| ±10% | **0.977** | 0.967 | 0.912 |

#### **4.5.3 激活函数的选择**

不同激活函数会影响多层感知机模型的精度，因此本节对pipSG模型中推理延迟预测

模型中的激活函数进行分析，考虑了Sigmoid、ReLU、LeakyReLU、Tanh四种激活函数。四个激活函数的公式如下：

在保持其他参数不变的情况下，推理预测模块分别使用四种激活函数运行300次迭代，最后在测试集上测试不同激活函数下pipSG 模型的准确度。实验结果如表4.5所示，从表中可以看出使用sigmoid激活函数的pipSG模型的准确度比使用其他三种激活函数的准确度更高，因此本文最终选择 sigmoid 函数作为推理性能预测模块的激活函数。

表 4.5 不同激活函数的实验结果

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Sigmoid | ReLU | LeakyReLU | Tanh |
| RMSE(ms) | **0.066** | 0.069 | 0.069 | 0.067 |
| MAPE(%) | **2.622** | 2.758 | 2.765 | 2.675 |
| ± 5% | **0.819** | 0.790 | 0.802 | 0.812 |
| ±10% | **0.977** | 0.970 | 0.965 | 0.964 |

## **5 总结与展望**

本章主要对论文的主要工作进行总结，以及指出针对论文存在的不足并探讨未来可能的解决方案。

### **5.1 工作总结**

本文对TensorRT推理框架下的稀疏神经网络推理性能建模进行了分析与研究。首先探讨了TensorRT推理框架中对稀疏神经网络推理性能影响因素，介绍TensorRT部署流程后对其部署过程中存在的算子间优化策略（算子间融合和算子间并行）进行了一定的分析，并通过实验对其在TensorRT推理框架上的存在性进行了验证。

考虑到现有方法难以对稀疏神经网络部署过程中存在的算子间优化策略进行周全的考虑，本文提出pipSG模型来缓解这一问题，pipSG模型由图特征提取和推理性能预测这两个模块组成，其中图特征提取模块使用三层的图卷积神经网络进行构建，推理性能预测模块使用具有一层隐藏层的多层感知机进行构建。稀疏神经网络的模型计算图包含网络算子间的拓扑结构以及算子自身的特征，因此本文考虑使用稀疏神经网络的计算图作为其特征表达。针对传统深度学习难以处理图数据这一问题，本文使用图卷积神经网络对稀疏神经计算图的特征进行提取来隐式学习TensorRT中的算子间优化策略，随后使用多层感知机对提取出的稀疏神经网络特征进行预测得到相应的推理延迟。

本文在2080Ti和TensorRT推理框架上收集了DenseNet、GoogleNet、VGG16 和 VGG19 这四种神经网络在使用结构化剪枝算法ABCPruner 后得到的稀疏神经网络模型集的推理延迟数据，得到了ILDS数据集，为了获得更准确的推理延迟数据，本文使用预热和多次测量取平均这两种策略。

接下来本文在ILDS这一数据集上验证了pipSG模型的有效性，pipSG模型相较于随机森林回归方法、KNN方法和线性回归方法具有更高的准确度，在稀疏神经网络合集中，pipSG模型在5%误差下的准确度为79%，比使用总体特征中最好的算法随机森林回归高了9.7%；然后通过实验对图特征提取模块中的聚合函数和推理延迟预测模块中的激活函数的选择进行分析，其中pipSG模型在使用sum聚合函数和sigmoid激活函数的情况下具有更高的准确度。

### **5.2 未来展望**

本文总体来说达到了预期，但仍存在一些不足，具体如下：

1. 本文仅考虑了神经网络稀疏化方法中的结构化剪枝算法，因此在后续的研究将对非结构剪枝算法和半结构剪枝算法得到的稀疏神经网络的推理性能进行建模。
2. 本文目前所收集的数据集覆盖面窄，因此在未来将进一步的扩展推理延迟数据集，收集在不同的硬件和推理框架下的稀疏神经网络推理性能数据，在扩展的数据集上验证pipSG模型的有效性。
3. 本文未对pipSG模型的可扩展性进行研究，本文提出的pipSG模型仅对单个神经网络模型稀疏化下的推理延迟进行预测，无法对未见过的神经网络类型的推理延迟进行预测，因此未来将考虑使用迁移学习、预训练模型等方法提高模型的可扩展性，使其能够对未见神经网络进行有效的预测，降低对新神经网络模型稀疏化的推理性能预测模型的训练成本。

## 参考文献

1. DENIL M, SHAKIBI B, DINH L, et al. Predicting Parameters in Deep Learning[A]. BURGES C J C, BOTTOU L, GHAHRAMANI Z, et al. Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States. 2013: 2148–2156.
2. DING Y, ZHU L, JIA Z, et al. IOS: Inter-Operator Scheduler for CNN Acceleration[J/OL].

CoRR, 2020, abs/2011.01302.

1. MA L, XIE Z, YANG Z, et al. Rammer: Enabling Holistic Deep Learning Compiler Optimizations with rTasks[A]. 14th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2020, Virtual Event, November 4-6, 2020. [S.l.]: USENIX Association, 2020: 881–897.
2. LECUN Y, DENKER J S, SOLLA S A. Optimal Brain Damage[A]. TOURETZKY D S. Advances in Neural Information Processing Systems 2, [NIPS Conference, Denver, Colorado, USA, November 27-30, 1989]. [S.l.]: Morgan Kaufmann, 1989: 598–605.
3. HASSIBI B, STORK D G. Second Order Derivatives for Network Pruning: Optimal Brain Surgeon[A]. HANSON S J, COWAN J D, GILES C L. Advances in Neural Information Processing Systems 5, [NIPS Conference, Denver, Colorado, USA, November 30 - December 3, 1992]. [S.l.]: Morgan Kaufmann, 1992: 164–171.
4. HAN S, MAO H, DALLY W J. Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding[J]. Fiber, 2015, 56(4): 3–7.
5. NARANG S, DIAMOS G, SENGUPTA S, et al. Exploring Sparsity in Recurrent Neural Networks[A]. 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings. [S.l.]: OpenReview.net, 2017.
6. LIU J, XU Z, SHI R, et al. Dynamic Sparse Training: Find Efficient Sparse Network From Scratch With Trainable Masked Layers[A]. 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020. [S.l.]: OpenReview.net, 2020.
7. LI H, KADAV A, DURDANOVIC I, et al. Pruning Filters for Efficient ConvNets[A]. 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings. [S.l.]: OpenReview.net, 2017.
8. LIU Z, SUN M, ZHOU T, et al. Rethinking the Value of Network Pruning[A]. 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019. [S.l.]: OpenReview.net, 2019.
9. LIN M, JI R, ZHANG Y, et al. Channel Pruning via Automatic Structure Search[A]. BESSIERE C. Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020. [S.l.]: ijcai.org, 2020: 673–679.
10. LIU J, SUN J, XU Z, et al. Latency-aware automatic CNN channel pruning with GPU runtime analysis[J]. BenchCouncil Transactions on Benchmarks, Standards and Evaluations, 2021, 1(1): 100009.
11. WEN W, HE Y, RAJBHANDARI S, et al. Learning Intrinsic Sparse Structures within Long Short-Term Memory[A]. 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings. [S.l.]:

OpenReview.net, 2018.

1. YAO Z, CAO S, XIAO W, et al. Balanced Sparsity for Efficient DNN Inference on GPU[A]. The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The ThirtyFirst Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019. [S.l.]: AAAI Press, 2019: 5676–5683. [15] YU Z, WANG C, WU Q, et al. Cross-Channel Intragroup Sparsity Neural Network[J/OL].

CoRR, 2019, abs/1910.11971.

1. GUO C, HSUEH B Y, LENG J, et al. Accelerating sparse DNN models without hardwaresupport via tile-wise sparsity[A]. CUICCHI C, QUALTERS I, KRAMER W T. Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, SC 2020, Virtual Event / Atlanta, Georgia, USA, November 9-19, 2020. [S.l.]: IEEE/ACM, 2020: 16.
2. CAI H, ZHU L, HAN S. ProxylessNAS: Direct Neural Architecture Search on Target Task and Hardware[A]. 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019. [S.l.]: OpenReview.net, 2019.
3. CAI H, GAN C, WANG T, et al. Once-for-All: Train One Network and Specialize it for Efficient Deployment[A]. 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020. [S.l.]: OpenReview.net, 2020.
4. WU B, DAI X, ZHANG P, et al. FBNet: Hardware-Aware Efficient ConvNet Design via Differentiable Neural Architecture Search[A]. IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019. [S.l.]: Computer Vision Foundation / IEEE, 2019: 10734–10742.
5. ZHANG L L, HAN S, WEI J, et al. nn-Meter: towards accurate latency prediction of deeplearning model inference on diverse edge devices[A]. BANERJEE S, MOTTOLA L, ZHOU X. MobiSys ’21: The 19th Annual International Conference on Mobile Systems, Applications, and Services, Virtual Event, Wisconsin, USA, 24 June - 2 July, 2021. [S.l.]: ACM, 2021: 81–93.
6. CAI E, JUAN D, STAMOULIS D, et al. NeuralPower: Predict and Deploy Energy-Efficient Convolutional Neural Networks[J/OL]. CoRR, 2017, abs/1710.05420.
7. YANG T, HOWARD A G, CHEN B, et al. NetAdapt: Platform-Aware Neural Network Adaptation for Mobile Applications[A]. FERRARI V, HEBERT M, SMINCHISESCU C, et al. Lecture Notes in Computer Science, Vol 11214: Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part X. [S.l.]: Springer, 2018: 289–304.
8. YAO S, ZHAO Y, SHAO H, et al. FastDeepIoT: Towards Understanding and Optimizing Neural Network Execution Time on Mobile and Embedded Devices[A]. RAMACHANDRAN G S, KRISHNAMACHARI B. Proceedings of the 16th ACM Conference on Embedded Networked Sensor Systems, SenSys 2018, Shenzhen, China, November 4-7, 2018. [S.l.]: ACM, 2018: 278–291.
9. DUDZIAK L, CHAU T C P, ABDELFATTAH M S, et al. BRP-NAS: Prediction-based NAS using GCNs[A]. LAROCHELLE H, RANZATO M, HADSELL R, et al. Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual. 2020.
10. GANESAN V, SELVAM S, SEN S, et al. A Case for Generalizable DNN Cost Models for Mobile Devices[A]. IEEE International Symposium on Workload Characterization, IISWC 2020, Beijing, China, October 27-30, 2020. [S.l.]: IEEE, 2020: 169–180.
11. GAO Y, GU X, ZHANG H, et al. Runtime Performance Prediction for Deep Learning Models with Graph Neural Network[R]. [S.l.]: Technical Report MSR-TR-2021-3. Microsoft, 2021.
12. KIPF T N, WELLING M. Semi-Supervised Classification with Graph Convolutional Networks[A]. 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings. [S.l.]: OpenReview.net, 2017.

## 声 明

本人声明所呈交的学位论文是本人在导师指导下进行的研究工作及取得的研究成果。

据我所知，除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得四川大学或其他教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示谢意。本学位论文成果是本人在四川大学读书期间在导师指导下取得的，论文成果归四川大学所有，特此声明。

学位论文作者（签名）

论文指导教师（签名）

2022 年 月 日

## 致 谢

毕业设计的完成，为我的大学生活划上了一个圆满的句号。

首先我在这里向养育我的父母表示深深地思念和感谢，在向关心我，帮助我的老师和同学表示感谢，正是有你们的支持，我才能得以顺利的完成我的毕业设计。

在去年11月份，我就开始了我的毕业设计，在这几个月里，我得到了段磊老师以及科大的孙广中老师和孙经纬老师的三位老师的关心和教导，我向三位老师表示深深的谢意；感谢陈小霞同学在论文写作过程中的帮助；同时感谢罗晨、焦威等舍友，李涛、李哲玮、李浩斌、杨佳智等同学在大学四年里的陪伴，能够与你们共同进步是我的幸运。

感谢四川大学在这四年里对我的熏陶和培养，川大的校训是：海纳百川，有容乃大。青春广场上的层出不穷的活动为我的大学生活平添了一份彩色；各式各样的选修课程让我对世界大致的轮廓有了一定的了解；深入浅出的专业课更是让我具备了一个扎实的计算机科学基础。

最后感谢自己在大学四年里的坚持与努力。