个人作业报告

SY2121106 胡哲宇

1. 目标模型简介

PyTorch是一个深度学习框架，最初由Facebook开发，包含2个核心功能：支持GPU加速的张量计算、支持自动微分的深度神经网络。张量计算模块以torch.Tensor类为核心，提供了大量计算函数。神经网络模块以torch.nn.Module类为核心，所有神经网络模块都继承自Module，支持主流的神经网络类型和计算模块。

PyTorch中的张量模块以提供丰富的操作函数为目的，不涉及太多业务对象，更适合用面向过程的思维来建模，所以不纳入目标模型。PyTorch中的神经网络模块主要目的是描述一个深度神经网络模型，每个模型内部可以组合多个模块，正好符合对神经网络进行建模的目的，所以将目标模型确定为神经网络模型。

神经网络有许多类型：多层感知机（MLP）、卷积神经网络（CNN）、循环神经网络（RNN）以及最新的Transformer等等。MLP作为最早出现的网络类型只涉及全连接层，模型内容过于简单。CNN、RNN和Transformer都是具备一定复杂度的网络类型，最终我选择了比较熟悉的CNN作为目标模型。

2. 元模型

卷积神经网络在计算机视觉领域有大量应用，通常由卷积层、池化层、激活函数层、线性层等部分组成。图1是相对完整的PyTorch元模型。

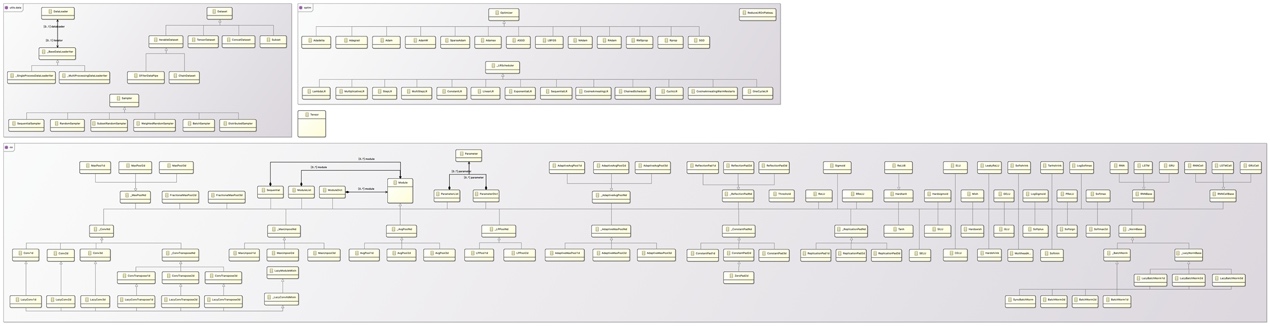


图1 相对完整的PyTorch元模型

从图中可以看到，即使将模型范围限制到CNN，涉及到的模块种类还是非常多的，仅仅激活函数层就有二十多种，如果选择完整的CNN进行建模，不仅难以突出重点（建模过程变成了琐碎的罗列类），而且过于复杂的模型也会对后续的图形建模和模型转换带来困难。因此，我挑选了经典模型中常用到的模块裁剪得到最终的元模型（图2），包括线性层、普通卷积层、最大池化层、平均池化层以及激活函数层，并通过prev和next关系来描述网络结构。

图示

描述已自动生成

图2 裁剪后的CNN元模型

CNN元模型中各Classifier和Association的含义见表1和表2。

表1 Classifier含义说明

|  |  |
| --- | --- |
| **名称** | **含义** |
| NeuralNetwork | 卷积神经网络，包含多个模块 |
| Module | 神经网络中的一个模块，所有模块的父类 |
| LinearLayer | 所有线性层的父类 |
| Identity | 同一变换层 |
| Linear | 线性变换层 |
| Bilinear | 双线性变换层 |
| ConvLayer | 所有卷积层的父类 |
| Conv1d | 一维卷积层 |
| Conv2d | 二维卷积层 |
| Conv3d | 三维卷积层 |
| MaxPoolLayer | 所有最大池化层的父类 |
| MaxPool1d | 一维最大池化层 |
| MaxPool2d | 二维最大池化层 |
| MaxPool3d | 三维最大池化层 |
| AvgPoolLayer | 所有平均池化层的父类 |
| AvgPool1d | 一维平均池化层 |
| AvgPool2d | 二维平均池化层 |
| AvgPool3d | 三维平均池化层 |
| ActivationLayer | 所有激活函数层的父类 |
| Sigmoid | Sigmoid激活函数层 |
| ReLU | ReLU激活函数层 |
| LeakyReLU | LeakyReLU激活函数层 |
| Softmax | Softmax激活函数层 |

表2 Association含义说明

|  |  |
| --- | --- |
| **名称** | **含义** |
| modules | 组合关系，每个神经网络包含多个模块 |
| prev | 前置模块，可能有多个 |
| next | 后置模块，可能有多个 |

3. 图形建模

本部分需要为元模型中的每个元素定义一个图形表示，并实现相应的新增、删除操作，从而实现通过图形界面操作来方便地进行建模的目的。元模型中元素的图形表示（映射关系）如表3所示，每种层用不同颜色的矩形表示，层与层之间的先后关系用箭头表示。

表3 元模型与图形建模语言映射关系

|  |  |
| --- | --- |
| **元模型** | **图形建模语言** |
| Identity | 浅蓝色矩形 |
| Linear | 紫色矩形 |
| Bilinear | 浅灰色矩形 |
| Conv1d | 浅绿色矩形 |
| Conv2d | 橙色矩形 |
| Conv3d | 浅紫色矩形 |
| MaxPool1d | 浅红色矩形 |
| MaxPool2d | 蓝色矩形 |
| MaxPool3d | 深蓝色矩形 |
| AvgPool1d | 深巧克力色矩形 |
| AvgPool2d | 深灰色矩形 |
| AvgPool3d | 深绿色矩形 |
| Sigmoid | 深橙色矩形 |
| ReLU | 深紫色矩形 |
| LeakyReLU | 深红色矩形 |
| Softmax | 深黄色矩形 |

此处用经典卷积神经网络模型AlexNet来展示图形建模语言的效果，AlexNet开创了用CNN做图像识别的先河，并在ILSVRC中取得冠军。AlexNet的结构如图3所示。自动生成的图形模型如图4所示。

图示, 工程绘图

描述已自动生成

图3 AlexNet网络结构

演示视频链接：<https://github.com/hzy1721/ParkShare/blob/master/%E7%AC%AC2%E6%AC%A1%E4%BD%9C%E4%B8%9A/SY2121106_%E8%83%A1%E5%93%B2%E5%AE%87/Sirius%E5%BB%BA%E6%A8%A1%E6%BC%94%E7%A4%BA.mp4>

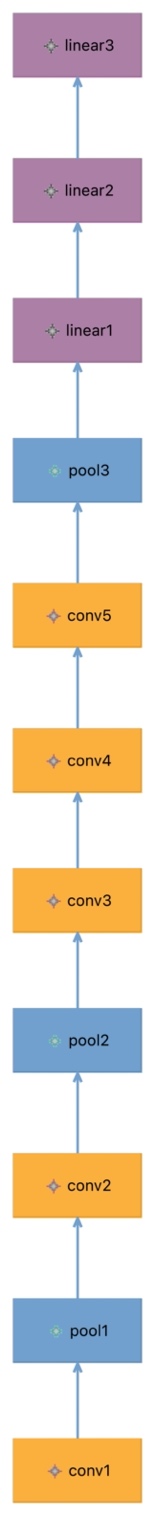


图4 AlexNet图形模型

4. 模型转换

ONNX的全称是Open Neural Network Exchange，是一个表示深度学习模型的标准，理论上不同框架（比如PyTorch、TensorFlow、MXNet等）的模型都可以导出为该标准定义的模型格式，所以我选择了ONNX作为模型转换的目的模型。由于源模型是经过裁剪的，目的模型也需要进行裁剪，最终得到如图5所示的目的元模型。

图形用户界面, 文本, 应用程序

描述已自动生成

图5 裁剪后的ONNX元模型

其中Graph和Node的含义如表4和表5所示。

表4 Graph各字段含义

|  |  |  |
| --- | --- | --- |
| **名称** | **类型** | **说明** |
| name | string | 模型的名称 |
| node | Node[] | 结点列表 |
| input | string[] | 输入列表 |
| output | string[] | 输出列表 |

表5 Node各字段含义

|  |  |  |
| --- | --- | --- |
| **名称** | **类型** | **说明** |
| name | string | 结点的名称 |
| input | string[] | 输入列表，每个元素必须对应于另一个结点的输出或者对应于Graph的输入 |
| output | string[] | 输出列表，每个元素必须是新引入的或者对应于Graph的输出 |
| op\_type | string | 结点类型 |

编写对应的ATL文件后，为了测试转换效果，使用之前提到的AlexNet模型作为转换的源模型，生成对应的ONNX模型。

转换后的xmi文件部分内容如下：

<?xml version="1.0" encoding="ISO-8859-1"?>

<Graph xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI" xmlns="onnx" name="AlexNet">

<node name="conv1" op\_type="pytorch!Conv2d">

<input>graph input</input>

<output>conv1->pool1</output>

</node>

<node name="pool1" op\_type="pytorch!MaxPool2d">

<input>conv1->pool1</input>

<output>pool1->conv2</output>

</node>

……

<node name="linear2" op\_type="pytorch!Linear">

<input>linear1->linear2</input>

<output>linear2->linear3</output>

</node>

<node name="linear3" op\_type="pytorch!Linear">

<input>linear2->linear3</input>

<output>graph output</output>

</node>

<input>graph input</input>

<output>graph output</output>

</Graph>