Strategy and Experiment

Sizhe Tang

28 March 2022

1 系统建模

1.1 系统建模

设备（车辆）和边缘服务器：d，$s_i \in S$, i 表示边缘服务器中的第几个

推理网络模型：$l_i \in Layer$, 把一个推理模型视为一个有向无环图，前后层顺序依赖，i 表示第 i 层（从 0 开始）共 L 层

分割点：$p_i \in L$, 以模型中的某一层为分割点，前部留在本地，后部卸载到 s。

数据量：d_i: 每一层的数据量大小，即要传输的数据大小

设备/服务器的计算能力：c_d, c_s^i（第 i 个服务器的计算能力, 由 cpu 频率来衡量）

W：带宽；S：信号强度；N：噪声强度；

信号传输率：

$$B = W \cdot \log_2(1 + \frac{S}{N}) \tag{1}$$

传输时间：

$$t_i^{tr} = d_i / B \tag{2}$$

计算时间：

c 为上方描述的设备算力，FLOPs 的计算依赖于预置模型

数据大小（卷积层还考虑窗口大小）———> 计算 FLOPs———>

根据各个设备的算力（cpu 频率等来衡量）———>

$$t_i^{comp} = prediction\_model(data_{size}, layer_{type})/c \tag{3}$$

总时间：

$$t_{total} = \sum_{layer=1}^{i} t_{layer}^{comp_d} + \sum_{layer=i+1}^{L} t_{layer}^{comp_s} + t_i^{tr} \tag{4}$$

优化目标：

$$\begin{aligned} \min \quad & t_{total} && (1)\\ \text{s.t.} \quad & t_i^{tr} < T, \quad \forall l_i \in Layer && (2)\\ & t_{total} < t_{local} && (3)\\ & t_{all} < t_{local} && (4)\\ & c_i^s > c^d \quad \forall s_i \in Server && (5)\\ & && (5) \end{aligned}$$

2 算法

2.1 计算延迟估计模型

是一个简单的神经网络，由几层全连接层组成，用来对模型中每一层的计算延迟进行预估，结果用来表示该层在不同端或边上的计算时间。

该模型的输入与当前层次的种类有关。卷积层：输入输出数据大小，输入输出特征数量；全连接层：输入输出神经元数量；激活函数/正则化：输入神经元数量。

应在车上部署两个训练好的模型，分别估计每个网络层在端、边上的计算延迟。

2.2 分割点选择算法

先通过上述预测模型进行计算时间的预估，再初始化每层的传输时间。然后对每一层在端上的计算时间、边上各服务器的计算时间、传输时间进行求和，通过循环找出最小的总完成时间，以此为分割点。

极端情况即为边界条件：全部在本地或全部卸载:t_{local} 和 t_{all}

3 实验

3.1 场景

将单一车辆的推理任务面向多个边缘服务器进行决策选择，最终选择一个做为卸载对象。

考虑在车联网环境下的路况检测、目标识别。鉴于 YOLO 等网络较为复杂，不适用于模型分割，所以采用 VGG 或 Alexnet 来实现推理。VGG 可以实现图像分类和图像定位，故也与此场景有关。

3.2 实验设置

选择 VGG / Alexnet 网络作为分割对象，数据选择路况实景？还是一个不在路况这一场景下的简单数据集？

实验模拟：现有的两个实验方案：

1、使用 python 中的 rpc（远程进程调用）或 socket 库进行搭建场景；

2、依托 cloudsim 等平台进行模拟