**大连理工大学本科毕业设计（论文）**

**车联网中的高效雾计算模型研究**

**The Subject of Undergraduate Graduation Project (Thesis) of DUT**

学 院（系）：

专 业：

学 生 姓 名：

学 号：

指 导 教 师：

评 阅 教 师：

完 成 日 期：

大连理工大学

Dalian University of Technology

# 摘 要

近年来，车辆的普及率和车辆智能化的程度不断提高，车载应用得以快速发展。随着不断增长的先进车载服务的出现，满足计算和通信需求的挑战日益突出。现有的解决方案如基于云服务中心和移动蜂窝网络等都是有缺陷的。以上方案虽然能在一定程度上解决终端计算能力不足的问题，但是由于车载应用中有许多延迟敏感型应用服务，以上的方案不能满足低延迟的特性。

雾计算（也称为边计算）通过使用部署在用户附近的本地计算设施来扩展云计算，在雾计算中，数据和应用程序都集中在网络边缘设备上，而不是需要传输存放到云上，这样能有效降低了云核心网络的压力，减少了延迟。基于云服务中心和雾（边缘）计算节点的模式是一种合理的方案来解决车辆本身计算能力不足的问题，并且雾节点基于地理分布更加接近用户能够满足一些低延迟的服务需求。在这个基本框架中，解决雾节点和核心云之间的相互合作和负载分配不仅能够提升用户的QoS，还能够减少能耗节省成本开销，具有巨大的研究意义和价值

因此本文在车联网的应用背景下研究了如何分配任务工作负载到云或雾节点使得能够保证总延迟在一定范围内系统总能耗最低。在本文中，建立了系统的模型。在模型中将上述问题公式化并证明其为NP难解问题，随着系统规模的不断增大，采用枚举暴力求解的方法的开销呈指数增长是不可取的。因此我们将系统分为前端和后端两部分，在前端采用组合模式的方案在满足额定延迟的约束下减少能耗开销，后端采用深度学习方法选择服务设备，最终我们选取了QoS参数延迟和系统能耗作为指标进行模拟仿真，仿真结果证明我们的模型是在能够通过优化工作负载的分配达到最小化系统能耗和时延。

**关键词：雾计算；神经网络；能耗优化**

目 录

[摘 要 I](#_Toc514340165)

[1 绪论 1](#_Toc514340166)

[1.1 研究的背景 1](#_Toc514340167)

[1.2 研究的意义 2](#_Toc514340168)

[1.3 国内外研究现状 3](#_Toc514340169)

[1.4 论文组织结构 3](#_Toc514340170)

[2 相关技术介绍 5](#_Toc514340171)

[2.1 车联网 5](#_Toc514340172)

[2.1.1 车联网的发展 5](#_Toc514340173)

[2.1.2 车联网网络架构 5](#_Toc514340174)

[2.2 深度学习 6](#_Toc514340175)

[2.2.1 深度学习发展历史 6](#_Toc514340176)

[2.2.2 深度学习简介 6](#_Toc514340177)

[2.3 人工神经网络 8](#_Toc514340178)

[2.3.1 神经元 8](#_Toc514340179)

[2.3.2 BP神经网络 10](#_Toc514340180)

[2.4 卷积神经网络 13](#_Toc514340181)

[2.4.1 卷积层 13](#_Toc514340182)

[2.4.2 池化 14](#_Toc514340183)

[2.5 本章小结 15](#_Toc514340184)

[3 模型建立和问题描述 16](#_Toc514340185)

[3.1 排队论模型 17](#_Toc514340186)

[3.1.1 组成部分 17](#_Toc514340187)

[3.1.2 分类和数量指标 19](#_Toc514340188)

[3.1.3 M/M/1 模型 20](#_Toc514340189)

[3.1.4 M/M/c 模型 22](#_Toc514340190)

[3.2 云雾结合系统模型 23](#_Toc514340191)

[3.2.1 系统能耗 23](#_Toc514340192)

[3.2.2 系统延迟 25](#_Toc514340193)

[3.3 约束条件 26](#_Toc514340194)

[3.4 问题描述 27](#_Toc514340195)

[4 解决方案 28](#_Toc514340196)

[4.1 贪婪算法 28](#_Toc514340197)

[4.2 深度学习模型 29](#_Toc514340198)

[4.2.1 输入与输出 29](#_Toc514340199)

[参 考 文 献 30](#_Toc514340200)

# 1 绪论

## 1.1 研究的背景

随着车联网的发展和车辆智能化程度的不断提高，人们对车辆的概念理解逐渐由传统的代步交通工具向高速智能的移动终端而转变。在车辆智能化的过程中需要各种各样的车载应用来满足车主用户的需求，然而由于车载计算能力的限制，许多复杂的应用在车载计算机上执行效果不好(如图像处理)。在没有得到充分的计算能力的支持下，各种车载应用服务都停留在概念阶段无法实际投入到日常使用。

云计算（Cloud Computing）技术是一种新兴的计算模型，其发展历经分布式计算、并行处理、网格计算。通俗来讲，“云”指的是大型数据中心网络中可利用的资源（包括存储、计算及相应软件服务等），云计算服务商利用高性能的计算机群组为用户提供相应计算服务，这实际上改变了传统的网络服务提供模式，也从一定程度上满足了更多人对于计算服务的需求而且更加便利与可靠，是一种更加灵活可控的信息化模式。云计算越来越为人们所熟知。近十年来，云计算充分的向人们展示了它的优越性。服务提供商把特定服务部署在云中，终端设备发送信息给服务，服务完成运算后将结果发回给终端，并将必要数据在云端存储。通过这种形式，云充分满足了终端设备的资源期待，也成为物联网生态系统中不可缺少的一环。正如思科云指数（2013-2018）报告的那样，互联网已经向基于云的结构的方向转向，自2008年以来，大部分互联网流量都发起或终止于云数据中心。

雾计算（边缘计算）是一种新兴概念，2014年由思科首次提出，在传统的用户和云之间新增了雾层，雾层由部署在网络边缘的基于地理分布的雾服务器组成（例如公园，公交总站，购物中心等）。每个雾服务器都是高度虚拟化的计算系统，类似于轻量级云服务器，并配备了大容量数据存储，计算和无线通信设备，通过将计算能力带入到与终端设备非常相近的网络边缘能够提供低延迟的服务。雾计算可理解为本地化的云计算是一种对云计算概念上的延伸。因此通过对原有的基于云的车联网进行扩展能够很好地解决终端计算能力不足的问题。在本文中，我们采用一个云雾结合的计算系统。一方面由于云服务器随着请求的增长能耗和开销较大，因此在云雾结合时考虑能耗是十分必要的。另一方面保证用户服务质量QoS（如延时要求）也是不可忽视的，用户不可容忍的延时将迫使用户选择其他供应商提供的更好的服务。为此，我们系统地研究了云雾结合计算系统中如何分配工作负载实现最小化功耗和延迟的问题。

## 1.2 研究的意义

云中心提供了高度集中的大量资源，然而只有云仍有一些不足。高延迟：离用户较远的距离导致了较高的网络延迟。对实时性要求高的应用难以部署在云中。网络拥塞：根据 Cisco 的预测，到2020年，全球智能设备的数量将达到500亿。相较而言，网络带宽的增长速度远远滞后。如果大量的物联网应用部署在云中，将会有数量庞大的传感器原始数据时刻不断的涌入核心网络，带来核心网络拥塞。 较低可靠性：安全，生命相关的物联网应用，一旦遇到应用失效，数据中心失效，或从终端用户到云平台的任何一段网络失效，都将带来重大的安全隐患。从终端到云的通信通路较长，失效风险较大；而在云中部署服务备份的成本也较高。随着近年来移动流量的增长，将超大容量数据传输到云端不仅给通信带宽造成了沉重负担，而且还给终端用户带来了难以承受的传输延迟和服务质量下降。除了实时交互和低延迟之外，移动用户如今占据主导地位，对移动性的支持和基于地理分布的特性至关重要。因此，云计算和移动应用程序的成功集成成为一项重要任务。为了应对上述挑战，Cisco公司在2014年提出了“雾计算”的概念。

雾是接近地面的云，雾计算和云计算有很多相似。它们都基于虚拟化技术，通过资源池的共享，为多用户提供资源。在 Cisco 的定义中，雾主要使用边缘网络中的设备。这些设备可以是传统网络设备（早已部署在网络中的路由器，交换机，网关等等），也可以是专门部署的本地服务器。一般来说，专门部署的设备会有更多资源，而使用有宽裕资源的传统网络设备则可以大幅度降低成本。这两种设备的资源能力都远小于一个数据中心，但是它们庞大的数量可以弥补单一设备资源的不足。现如今国家在大力发展物联网，物联网发展的最终结果就是将所有的电子设备，移动终端，家用电器等等一切都互联起来，这些设备不仅数量巨大，而且分布广泛，现实的需求为雾计算提供了发展机会。有了雾计算才使得很多业务可以部署。比如：车联网。由于车辆具有移动性，车联网的应用和部署要基于地理分布有多种连接方式可选。车和车，车和接入点，接入点和接入点。

从第一次工业革命开始,各种不可再生能源便不断被人类开采消耗着:石油、煤炭、天然气等,而如今我们日常生活中不可缺少的电、燃气和汽油都主要由以上能源供养着,据称按照目前的开采使用速度,差不多2050年左右,人类就将面临能源危机,节约能源已经成为国家发展的重要战略。我国2011年用于数据中心总耗电量达700亿千瓦时,已经占到全部用电量的1.5%。通过综合分析我国云计算数据中心的能耗趋势,不难发现云中心的耗电呈现出逐年增长的态势,能耗问题是云计算中巨大的挑战。根据美国LBNL[]公布的数据,云中心耗能主要为服务器设备和制冷系统,分别占比46%和31%。通过引入雾设备能够缓解云的压力,并且由于雾节点是基于地理位置分布的因此能够解决散热的问题,云雾结合的模型本身能够减少一部分能耗开销。但只有这些是远远不能够满足国家节能发展的要求。因此解决既满足用户的服务质量保障延迟在容忍范围,又能有效降低系统的能耗的需求是十分必要的,本文通过在前端采用V2V和V2I两种结合的模式来降低能耗和延迟,后端合理地进行负载分配来满足上述需求。

## 1.3 国内外研究现状

对于云计算而言，国内外有很多关于负载均衡的算法。计算任务分配的算法有多种多样的，其主要可以分为2个方面即静态策略和动态策略。静态算法是指根据服务器的一些系统信息（如进程个数，任务周期等）进行分配。在这种策略下，网络节点仅在任务启动前根据分配策略执行，不能够动态监测节点负载的变化，不具备动态调整的能力。在早期的网络中，由于任务数量和流量少，静态策略的开销小是一种可行的方案。但是随着云计算的不断发展，网络规模不断扩大，每个节点由于负载的不同，执行时间不同，动态调整的能力是不可或缺的。因此静态的策略在实际中并不适用。

动态调整策略与静态的不同，可以根据一段时间内节点的任务数量通过调度算法来平衡节点的负载。与静态相比，动态策略在实际应用中展现出的效果较好。但同时动态策略不得不在监听节点状态上开销。并且监听的时间周期是一个比较难以选择的参数，周期过小不具备有实时性，反之浪费在监听上的开销过大。

综合以上分析，静态策略的适应性较差，开销小易实现。而动态策略则更注重实际应用，具有实时性能够动态地调整。但是以上两方面的算法都是在云上的负载均衡的算法，事实上这些负载均衡的算法并不适用于我们的模型，他们的优化指标一般是QoS延迟（如最快响应时间算法，最少连接数算法）并不考虑系统能耗的问题。在[]中虽然考虑了能耗的问题但是这种方案只能优化车联网中的延迟和能耗，并没有考虑如何优化在后端云雾上的分配。在[] 中也是仅仅考虑了在云上如何进行优化计算卸载并没有涉及到雾层。因此我们所考虑的问题和模型相较与其他研究更加全面合理。

## 1.4 论文组织结构

针对车联网中的雾计算模型，我将论文分为4个部分进行阐述。

第一部分是绪论，在绪论中介绍了本文的研究背景、研究意义和研究现状以及论文的组织结构。

第二部分是相关知识的介绍，主要包括后端的深度学习，卷积神经网络和前端车联网，车辆接入模式的一些简单介绍。

第三部分中我们在车联网中建立的一个云雾结合的模型，并进行相关问题描述。

第四部分是解决方案，主要分为前端和后端两个方面，前端通过车联网中对V2V和V2I的选择来降低能耗，后端采取深度学习的方法来分配任务。

第五部分是仿真实验及结果。

# 2 相关技术介绍

## 2.1 车联网

### 2.1.1 车联网的发展

从时间的角度来看,车联网有着悠久的历史,其起源可以追溯到上个世纪60年代日本开始研究车辆间通信。1980年日本最先研发路车间通信系统RACS(Road Automotive Communication System),建立了车联网的雏形。1997年日本提出Smartway的概念并逐进入到理论研究的阶段。20世纪末到21世纪初美国和欧盟相关项目逐步启动,为车联网部署基础设施,划分频段,车联网进入了发展阶段。到目前为止,车联网仍属于发展阶段。中国车联网的发展始于2009年，在清华大学的支持下，启动了智能汽车道路协调863计划。2012年,中国物联网重大专项"基于物联网的城市智能交通关键技术研究与应用"由广州市交通信息投资有限公司牵头正式启动，从此中国车联网发展走向快车道。2016年,中国工信部批准上海、重庆、武汉、亦庄、广州等多个智能网联汽车示范基地。2017年9月,工信部电信研究院发布了《车联网白皮书(2017年)》预计到2020年基本建成国家车联网产业标准体系。

从应用发展的角度可以分为4个阶段，目前我们处于正在优化第三阶段。第一阶段，提供安防类的紧急救援为主，主要关注整车的安全问题，当出现了事故时可以进行紧急呼救。第二阶段，用安卓手机操作系统，将互联网的内容和服务映射到车机系统，行业内比较知名的如基于安卓的CarLife、基于苹果IOS的CarPlay。第三阶段，基于自主可控的操作系统和基于云端计算的生态出行服务。AliOS跟安卓最大区别就是核心引擎既有云端又有终端，同时，基于AliOS衍生出的创新功能，垂直深入到车辆CAN-BUS总线进行控制，是其他车载操作系统无可比拟的。第四阶段，真正的V2X打通，包括车与车、人与车和基础设施的打通以及更多的想象空间。

### 2.1.2 车联网网络架构

从网络上看，车联网(IOV)系统是一个“端管云”三层体系。

第一层（终端端系统）：端系统是汽车的智能传感器，负责收集和获取车辆的智能信息，感知车辆的状态和环境.它是具有车内通信、车间通信、车网通信功能的通信终端;同时它还让汽车具备寻址和网络可信标识等能力。

第二层（管理系统）：解决车与车(V2V)、车与基础设施（V2I）、车与人（V2H）等的互联互通，实现车辆自组网及多种异构网络之间的通信与漫游，在功能和性能上保障实时性、可服务性与网络连通性。

第三层（云服务系统），车联网是一个云架构的车辆运行信息平台，需要能够满足丰富的车载应用需求，由于终端的计算能力有限，因此需要于后端的云服务系统进行交互以便获得足够的计算资源。

## 2.2 深度学习

### 2.2.1 深度学习发展历史

在最初深度学习被赋予了很多不同的名字，就在其火爆的前几年它还是相对冷门，这给我们造成了深度学习是一个全新领域的假象。事实上，深度学习的起源可以追溯到上个世纪40年代。到目前为止，深度学习的发展经历了3个重要的阶段，上世纪40年代到60年代，控制论中首次提到深度学习概念，深度学习的雏形由此产生。20 世纪 80 年代到 90 年代，深度学习表现为联结主义，直到2006年，深度学习才以如今的名字流行。图2.1给出了深度学习发展历程的展示。

起初，深度学习的思想源于人工神经元模型(MCP)，该模型通过将神经元简化为加权、求和、激活三个过程来模拟人的神经元反映。但这种简单的模型只能处理简单线性分类的问题，因此很长时间无人问津陷入停滞。直到1989年Hinton发明了适用于多层感知器（MLP）的BP算法并采用Sigmoid进行非线性映射，解决了非线性分类和学习的问题，深度学习的发展出现了转折。如今随着数据量的增多以及计算能力的提升，深度学习的优势才逐渐引起人们的重视。目前，在很多领域都能看到深度学习的身影。

### 2.2.2 深度学习简介

深度学习（也称为深度结构学习或分层学习）是众多机器学习方法的一个分支，它是一种使用多个处理层对数据进行复杂多重非线性变换从而抽象数据的算法，可分为监督、半监督、无监督的学习。学习是指让计算机自动调整函数参数以拟合想要的函数的过程，而深度是指多个函数进行嵌套，组成一个多层神经网络，利用非监督逐层训练调整有效地自动调整函数参数。简单的说深度学习就是用机器学习的方法使用多层神经网络进行学习。深度学习主要应用在图片，语音，文字识别方面。深度学习的Imagenet 已经可以识别1000多类的物体。对象识别最受人们关注的自然就是每年都要举行的ImageNet大型视觉挑战赛（ILSVRC）。深度学习在这项比赛上取得了傲人的成绩，每年的比赛我们可以看到都会有不同成都的提高，如图2.2

综上所述，深度学习应用统计学、应用数学、信息论和人脑结构等知识在多层嵌套神经网络架构上进行参数调整，最终达到我们预期的输出结果。在大数据时代的驱动下，随着计算机硬件的发展，处理能力的增强，深度学习的优势将越来越明显，应用的范围也会越来越广泛。

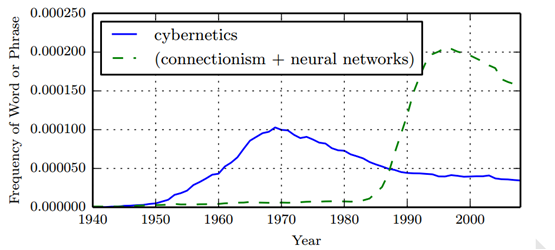


图2.1 深度学习发展历程

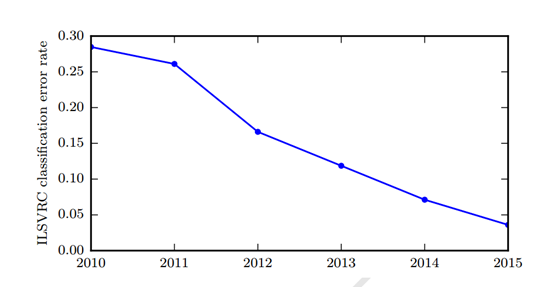


图2.2 ImageNet大规模视觉识别挑战赛中TOP5精度的错误率

## 2.3 人工神经网络

利用机器来模仿人的智能是长期以来人们认识自然、改造自然和认识自身的理想。20年前一提到神经网络，不禁让人联想到生物神经中数以万计的神经元和突触。但是今天人们已经将生物神经系统部分应用于计算机上，由此诞生了人工神经网络。具体来书人工神经网络是一个并行、分布处理结构，它由很多处理单元及通道互联而成，这些处理单元具有局部内存，可以完成局部操作，即它必须仅仅依赖于经过输入联接到达处理单元的所有输入信号的当前值和存储在处理单元局部内存中的值。每个处理单元只有一个单一的输出信号，可以是任何需要的数学模型。这些处理单元就是神经网络中的神经元。与生物神经系统相似，人工神经网络需要经过训练学习后才能对输入信号产生正确的输出，就像我们要通过学习后才能够对外界的刺激尝试正确的反射。

### 2.3.1 神经元

我们的大脑可以视为1000多亿神经元组成的神经网络，如果仅仅靠单个的神经元, 是永远没有办法让我们像今天一样, 完成各种任务, 处理各种复杂的问题。每个神经元如图2.3所示神经元是神经系统的基本单位，其消息传递和处理是一种电化学过程，树突用于接受外界刺激，当刺激达到一定程度即轴突电位达到一定的值则产生动作电位，通过轴突末梢传递给其他神经元，这一过程可以看作为多输入单输出的过程。

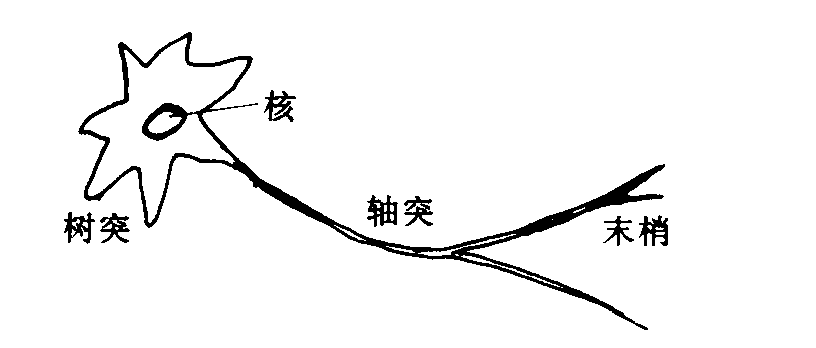


图2.3 大脑神经元示意图

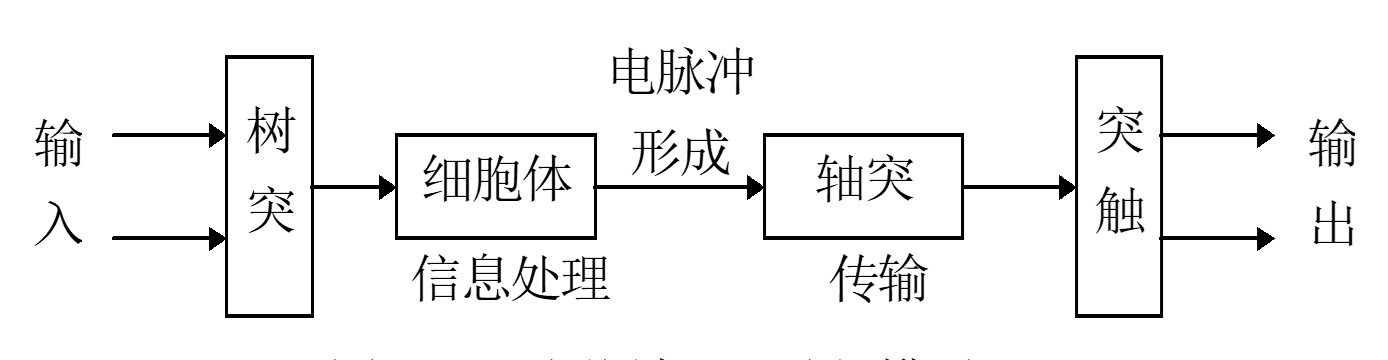


图2.4 神经系统工作流程

通过在生理上的研究人类的神经元结构以及生物神经网络的模型，我们可以总结出生物神经活动具有巨量并行性、信息处理和存储单元结合、自组织自学习等特征。通过这些研究人们提出了人工神经网络种的神经元模型如图2.5所示。

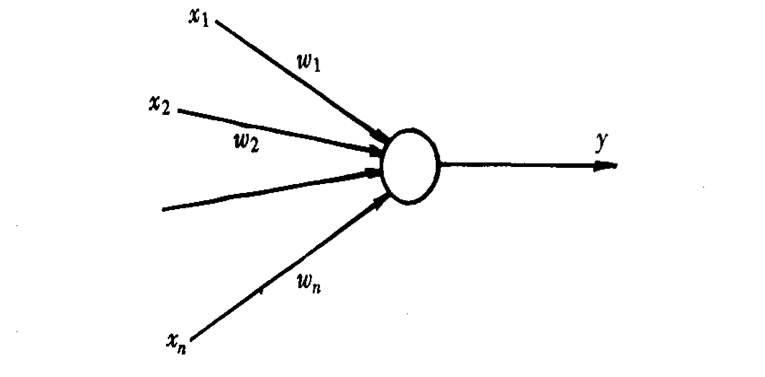


图2.5 神经元的数学模型

从图中可以看出神经元接受n个不同的输入，通过神经元的处理输出结果y，输入和输出关系如下式。

其中 x＝（, … ,）为输入向量，是权系数，为阈值，f是激活函数。激活函数按照类型可以分为线性激活函数、非线性斜面函数、阈值函数、S型函数等通过激活函数可以很好地引入非线性因素，解决纯线性模型解决不了的问题(如异或问题)。一句话概括激活函数: 就是让神经网络可以描述非线性问题的步骤, 是神经网络变得更强大。常见的几种激活函数如图2.6所示

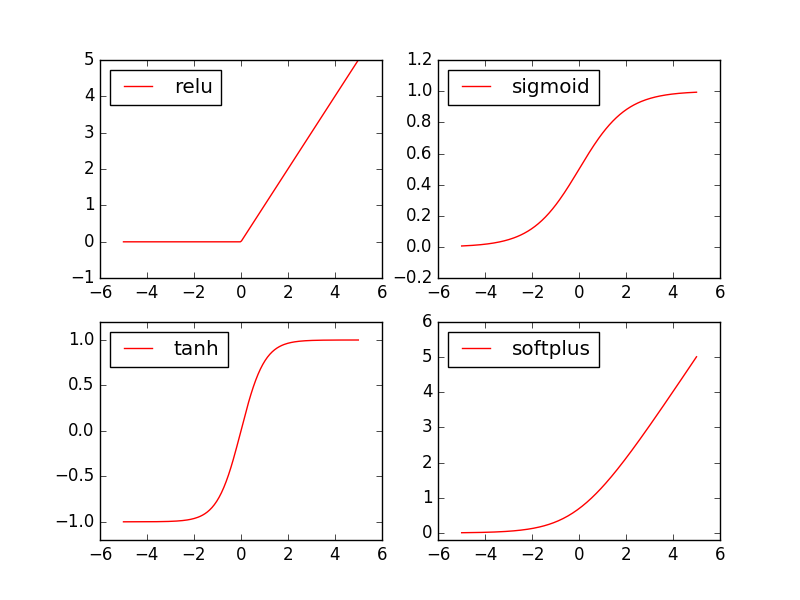


图2.6 常见激活函数

### 2.3.2 BP神经网络

前馈神经网络是一种比较简单的神经网络，在前馈神经网络中，各个神经元按层进行排列，每一层的神经元只与上一层的神经元相连，接受上一层的输出处理后传递给下一层，BP（Back propagation）神经网络是一种常见的前馈神经网络，也是目前应用最广泛的神经网络。BP神经网络之所以是一个前馈神经网络是因为它的主要思想是按照训练误差反向传播给各个层并进行调优，把偏差一级一级向前传递，逐层得到，这就是反馈。BP网络的结构包括输入层、隐含层（一层或多层）、输出层。隐含层包含多个神经单元，每一个隐含层所包含的神经元个数可以是不同的，如下图所示。

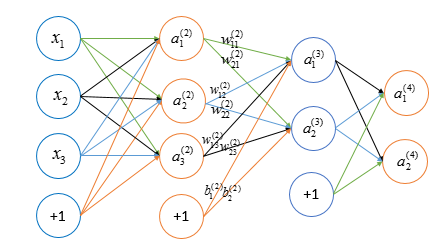


图2.5 BP神经网络的结构

BP学习算法：算法的伪代码如表2.1所示。BP学习算法的流程是首先对权值进行初始化，然后依次输入n个学习样本，通过前向传播得到输出，反向传播求解各个层的误差，按照权重修改公式修改各层的权值和阈值，按照新的权值和阈值再次计算，达到最小误差或最大学习次数时停止学习。基本BP算法包括输入的前向传播和误差的反向传播两个过程。

|  |
| --- |
| 算法：BP 算法 |
| 输入：训练数据集D={()}； |
| 学习率 |
| 开始： |
| 1： 随机初始化连接权重和神经元阈值 |
| 2： 循环 |
| 3： for (,) D |
| 4： 通过当前的参数计算 |
| 5： 计算误差E = |
| 6： 求偏导数，计算每层对误差的影响 |
| 7： 更新参数 |
| 8： end for |
| 9： 直到满足终止条件 |

表2.1 BP算法

前向传播：从样本集中去出一个样本（X，Y），将X输入到网络中，经过各层传递得到实际输出结果。在上图所示的网络结构中，从输入层到隐含层有 通过激活函数sigmoid有。继续传播，从隐含层到有，同样的，通过激活函数sigmoid有。继续前向传播从最后一层隐含层到输出层有=对输出层激活后有。至此前向传播完毕。

反向传播：神经网络中学习的本质是对连接各个层的权重进行调整，而对权重调整的过程是在反向传播中实现的。反向传播的具体操作是将网络输出的数据和真实的输出数据的误差反向传递到各个层，以便调整各个层的权重值。我们还是以图2.5所示的网络结构为例，对反向传播算法进行解析。反向传播首先要在前向传播的基础上计算误差

采用欧式距离公式的平方作为网络输出和样本输出之间的误差。反向传播将改误差传播到各个层，首先我们以距离结果较近的权值为例，如果我们想知道权重 对整体误差产生了多少影响，可以通过链式法则的方式用整体误差对权重求偏导。

其中：

因此通过链式法则可以求解出的值。我们用来表示输出层的误差，则对权重的求导可进一步表示为

通过计算出误差对权重的偏导值，我们可以通过下面的公式进行更新权重

通过不断地使用梯度下降法进行迭代更新权重，反向传播算法能够使得误差不断减小最终达到要求。

## 2.4 卷积神经网络

卷积神经网络(Convolutional Neural Network,CNN)是一种前馈神经网络，主要应用于图像识别系统中。在输入层和输出层之间，卷积神经网络由一个或多个卷积层和最后的全连接层构成，中间包括激活和池化层。卷积层由若干个卷积单元组成，卷积层主要进行卷积运算，通过反响传播算法对参数进行优化。卷积层的目的是提取出不同特征，第一层卷积只能提取一些低级的特征，更多层能从低级特征中迭代提取更复杂的特征。而池化层会对卷积层所得到的特征进行降维稀疏处理，减少数据运算量。卷积神经网络的操作流程如图2.6所示

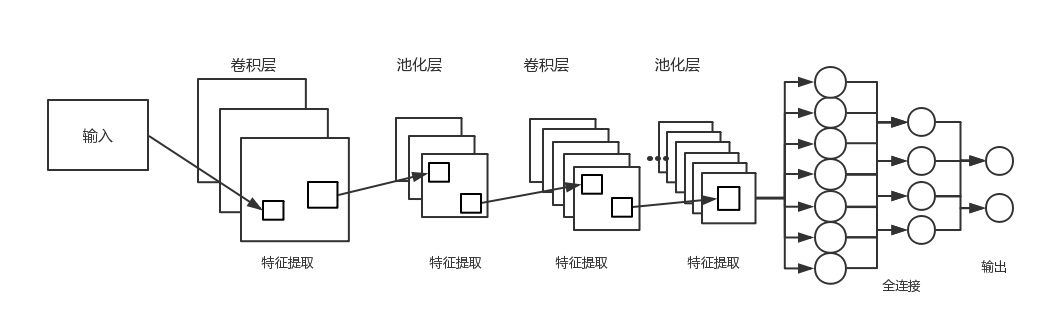


图2.6 卷积神经网络架构

### 2.4.1 卷积层

卷积层的目的是提取出数据的特征。卷积层主要通过卷积运算提取特征，卷积运算是一种加权求和的线性运算，如果在二维上，有两个的函数和 。和的卷积就是一个新的函数 。通过下式得到:

通过公式(2.1)不难看出在二维上卷积运算是以函数为权重，以为中心进行加权求平均的操作。其离散的形式如公式2.2所示。

卷积层有3个重要的概念分别是共享权重、稀疏连接和平移等变性。下面我们对这三个感念进行详细的阐述。

以的二维数据为例，如果要生成一张的特征，以神经网络中全连接的方式需要 个参数，如果数据长度和宽度较大，需要的参数个数太多以至于无法进行有效地训练计算。因此在卷积层中定义了感受视野，通过一个带有卷积核的感受视野扫描生成的下一层神经元矩阵称为一个特征映射图(feature map)。在同一个特征映射图中的神经元使用的卷积核是相同的(一个卷积核对应一个特征图)，因此这些神经元具有共享的权重。由于有感受视野的存在即隐藏层的神经元并不是与全部的输入层相连而是与感受视野范围内的输入层相连进行卷积运算。如图2.3所示，CNN层与层之间的连接并不全是全连接，而是稀疏连接，它的目的就是大幅度减少参数个数。当输入发生改变，输出自然也会发生相同的变化，具备平移等变性。平移等变性是由参数共享形成的。

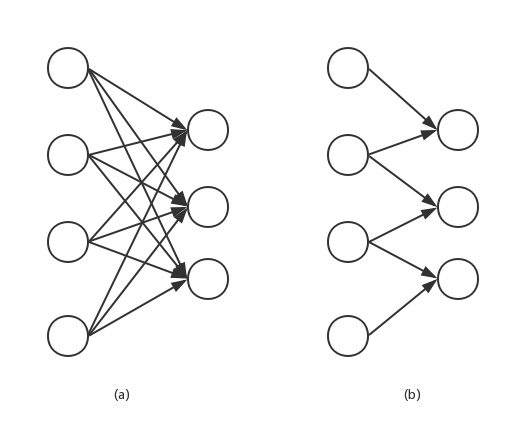


图2.3 稀疏连接，（a）表示全连接的结果，（b）使用感受视野为2的卷积核产生的结果

### 2.4.2 池化

在通过卷积获得了特征之后，下一步我们希望利用这些特征去做分析。理论上讲，人们可以用所有提取得到的特征去训练分类器，但这样做面临计算量的挑战。因此在处理之前我们要对提取出的特征映射图进行池化，池化的目的是减少输入到分类器中的特征。池化的种类包含最大值池化、均值池化等。

如果人们选择连续范围作为池化区域，并且只是池化相同的隐藏单元产生的特征，那么，这些池化单元就具有平移不变性。这就意味着即使图像经历了一个小的平移之后，依然会产生相同的 (池化的) 特征。

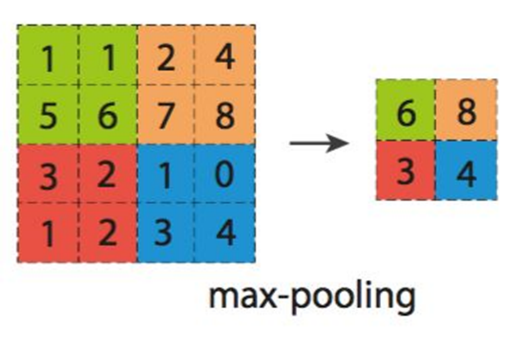


图2.3 池化

## 2.5 本章小结

在本章中我们主要对车联网和深度学习的基础知识和发展历史进行了介绍。学习任一门知识都应该先从其历史开始，把握了历史，也就抓住了现在与未来。本章重点对本文后续要采用的卷积神经网络进行了详细的介绍。通过了解卷积计算和卷积层的权重共享、稀疏连接和池化层的一些池化方法对卷积神经网络的架构和工作方式有了一定的了解。通过这些介绍能够更好地理解后续的模型。

# 3 模型建立和问题描述

在本节中我们将详细阐述在车联网下我们建立的满足一定延迟的最小化能耗云雾结合模型。如图3.1所示我们所考虑的系统框架包含：一组路侧单元(RSUs)，距离用户侧较近的雾层设备，距离用户较远的云服务中心。路侧单元接受来自用户的请求并通过本地局域网(LAN)将他们转发到一组N个雾设备上，在模型中RSU扮演着用户接口的角色。雾节点可以处理一些对延迟敏感的轻量级请求，对于运算量较大的请求或者说当前雾设备处理请求较多无法及时处理，雾设备可以将他们转发到其他雾设备或者通过广域网(WAN)转发到云服务器上进行处理。每个云服务器都承载着许多同类的计算机器。从雾设备到云服务器通过广域网传输，由于广域网从雾到云跨越的地理区域较大，因此传输延迟不能被忽略，应该考虑通信延迟和带宽。此外云节点或雾节点对请求的处理延迟也应该被加入到请求的总延迟中。因此在延迟方面我们主要考虑从车发送请求到RSU的延迟、请求的处理延迟（云和雾）、请求发送到云服务器上的延迟。至于为什么不考虑请求返回给用户的延迟，因为一般来说请求的结果数据是远远小于请求上传的数据的，在[]这篇文章有中详细的解释。在能耗方面，我们主要考虑车辆发送请求到RSU所消耗的能量、云服务器消耗的能量、雾设备消耗的能量。表3.1中总结了接下来我们要用到的公式符号。

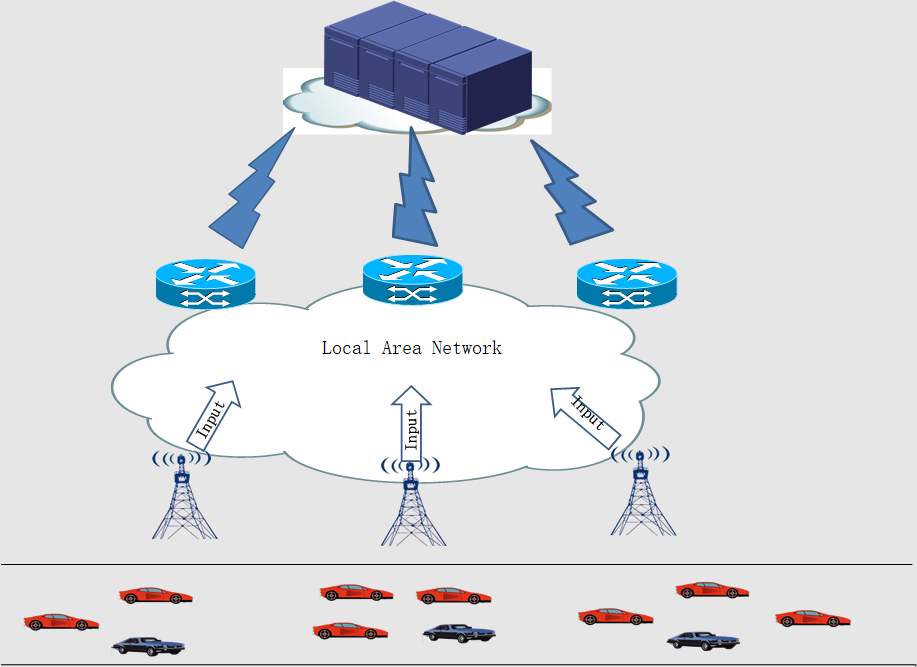


图3.1 雾云计算系统的整体架构图

|  |  |
| --- | --- |
| **符号** | **定义** |
| **i,N,****N** | **索引，集合大小，雾设备集合** |
| *j,M,M* | **索引，集合大小，云服务器集合** |
| *r,R,R* | **索引，集合大小，路侧单元集合** |
|  | **从雾设备i到云服务器j的请求转发速率** |
|  | **云服务j上的工作负载** |
|  | **路侧单元接受到的总请求** |
|  | **分配给雾层设备的工作量** |
|  | **分配给云服务中心的工作量** |
|  | **能耗** |
|  | **延迟** |
|  | **雾设备i的服务率** |
|  | **云服务器j中计算机的CPU频率** |
|  | **云服务器j的开机状态** |
|  | **21维矩阵，用于选择传输模式**  **是V2V还是V2I** |
|  | **云服务器j上的机器数量** |
|  | **从雾设备i到云服务器j的通信延迟** |
|  |  |

**表3.1 公式符号说明**

## **3.1 排队论模型**

在我们的日常生活中，经常会出现排队的现象如去超市排队结账，购票等。排队论是研究和解决具有排队现象的问题而发展起来的一门应用数学分支。排队论的基本思想是在1910年由丹麦电话工程师A.K.埃尔朗提出，起初是为了解决自动电话设计问题。排队论是一门研究服务系统中排队现象随机规律的学科，现如今广泛应用在通信、网络、交通中，通过对排队论模型的研究使得队列长度和等待时间可以预测。

### 3.1.1 组成部分

排队系统模型可以分为三大基本组成部分分别为“输入过程”，“排队规则”，“服务机构”。如下图所示，下面我们将详细描述这三大组成部分。

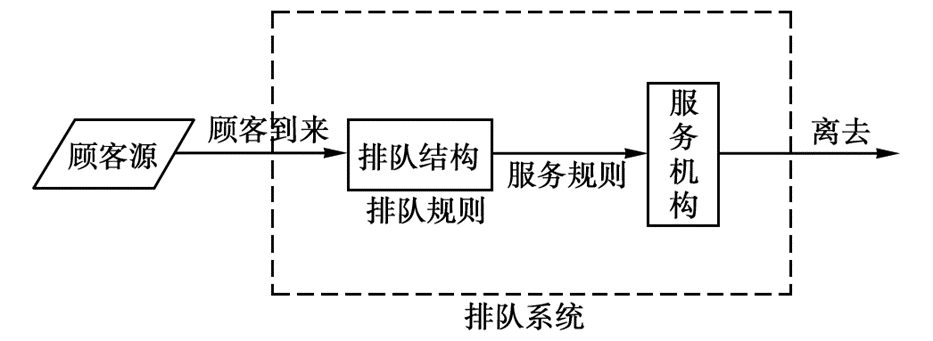


图3.2 排队论模型图

输入过程：输入过程即顾客到达的方式，顾客的总体的组成可能是有限的，也可能是无限的。顾客相继到达的时间间隔可以是确定的，也可以是随机的，对于随机的情形，要知道单位时间内的顾客到达数或相继到达的间隔时间的概率分布。输入过程可以是平稳的（描述相继到达的间隔时间分布和所含参数（如期望值、方差等）都是与时间无关的），否则成为非平稳的，我们研究的是平稳的过程。

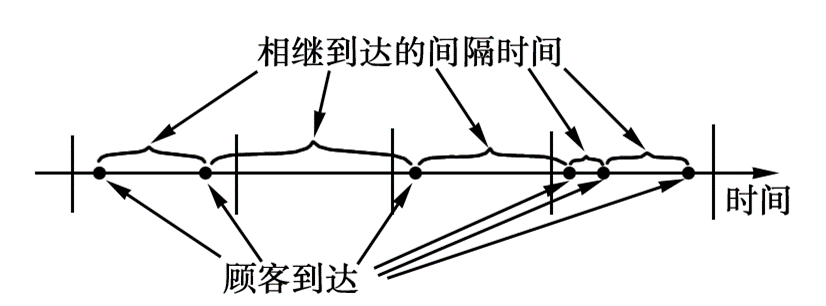


图3.3 到达时间示意图

排队规则：排队规则可以分为损失制、等待制、混合制三种。在损失制下，如果顾客到达排队系统时，所有服务台都已被先来的顾客占用，那么他们就自动离开系统永不再来。典型例子是，如电话拔号后出现忙音，顾客不愿等待而自动挂断电话，如要再打，就需重新拔号，这种服务规则即为损失制。等待制是指当顾客来到系统时，所有服务台都不空，顾客加入排队行列等待服务。例如，排队等待售票，故障设备等待维修等。混合制是上面两种的结合，即允许排队，但又不允许队列无限长下去，具体策略可以是队长有限、等待时间有限或逗留时间有限。

服务机构可以没有服务员，也可以有一个或多个服务员(服务台、通道、窗口等)从数量上说，服务台有单服务台和多服务台之分。在有多个服务台的情形中，可以是平行排列的，也可以是前后排列的，或混合排列的。对于等待制，服务规则又可以分为先到先服务（FCFS），后到先服务（LCFS），随机服务和有优先权的服务。

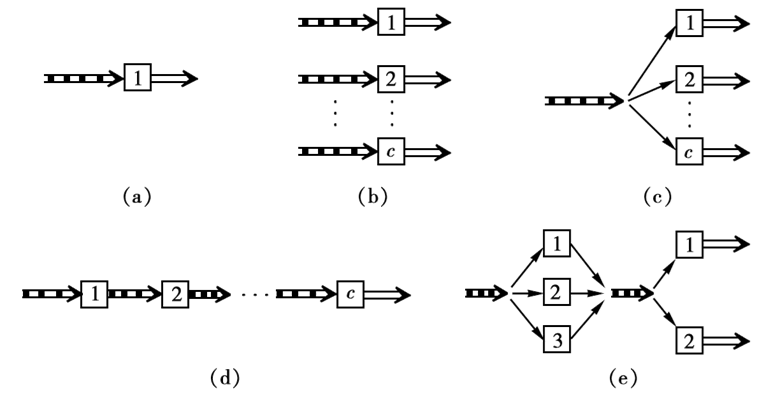


图3.4 服务台的各种排列方式

### 3.1.2 分类和数量指标

相继顾客到达间隔时间的分布、服务时间分布、服务器个数是构成排队模型的三个主要特征指标。根据这三个特征可以通过Kendall记号对排队模型进行分类X/Y/Z,其中X表示相继到达间隔时间的分布；Y表示服务时间的分布；Z表示并列的服务台的数目。常用的分布有负指数分布（M）、确定型分布（D）、爱尔朗分布（E）。在此基础上加上后三项可以将其扩充为X/Y/Z/A/B/C，其中前三项意义不变。后三项中A代表系统容量限制N，或称等待空间容量。如系统有N个等待位子，则 0< N <∞。当 N =0 时，说明系统不允许等待，即为损失制。N =∞ 时为等待制系统，此时∞般省略不写。N为有限整数时，表示为混合制系统。B代表顾客源数目m。分有限与无限两种，∞表示顾客源无限，此时一般∞也可省略不写。C代表服务规则，如先到先服务(FCFS)，后到后服务(LCFS)，优先权服务(PR)等。当后三项为∞/∞/FCFS可以默认省略。在排队论中有一些数量指标定义如下表

|  |  |
| --- | --- |
| **指标** | **定义** |
| **平均队长(Ls)** | **在系统中的顾客数（包括正被服务的顾客 和排队等待的顾客）的期望值** |
| **平均排队长(Lq)** | **系统中排队等候服务的顾客数的期望值** |
| **平均逗留时间(Ws)** | **一个顾客在系统中的停留时间期望值** |
| **平均等待时间(Wq)** | **一个顾客在系统中排队等待的时间的期望值** |
| **忙期** | **从顾客到达空闲服务机构到该服务机构再次空闲的时间长度。即服务机构连续忙的时间长度** |
| **系统的状态概率** | **指系统中的顾客数为n的概率** |
|  | **服务率，单位时间完成服务顾客的个数** |
|  | **到达率，单位时间内到达的顾客数量** |
|  | **服务强度，到达率与服务率的比值** |
|  | **时刻t系统状态为n的概率** |

**表3.1 公式符号说明**

逗留时间（Ws）和等待时间（Wq）是排队论中主要研究的两个指标，也是顾客最关心的QoS的指标，对于顾客而言通常希望完成同样一件事情的时间越短越好。

### 3.1.3 M/M/1 模型

在M/M/1模型中我们假设顾客到达的时间间隔满足参数为的负指数分布，服务时间满足参数为的负指数分布。负指数分布在日常生活中比较常见，如网站访问的时间间隔、婴儿出生的时间间隔等，它用于描述泊松过程中事件之间的时间的概率分布具有无记忆性。在M/M/1模型中假设服务机构是单台服务器，客户源是无限的，顾客相互独立，并且满足<，否则服务队列将会无限增长，无法求解。M/M/1中状态转移图如图3.5所示

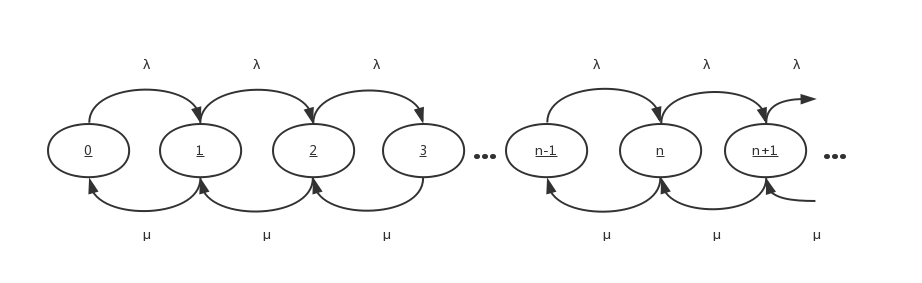


图3.5 M/M/1 状态转移图

根据状态转移图，考虑在时刻**t+**，系统中有n个顾客的四种情况如下表3.2所示

|  |  |  |  |
| --- | --- | --- | --- |
| **情况** | **t时刻顾客数** | **在区间[t,t+**] | **时刻t+**的顾客数 |
| **A** | **n** | **到达0 离去0** | **n** |
| **B** | **n+1** | **到达0 离去1** | **n** |
| **C** | **n-1** | **到达1 离去0** | **n** |
| **D** | **N** | **到达1 离去1** | **n** |

表3.2 时刻**t+**的四种情况

根据以上四种情况，我们可以对系统内的顾客数列出式3.1

整理可得

当时，考虑根据上式得到,由于所以有

由公式（3.3）可得出结论得到了系统状态得递推公式。通过对求和可得

根据上式可以求解出,。进一步我们对求期望可得到系统内的平均顾客数Ls。

根据等比数列求和公式可求得

根据little 公式可求得顾客平均逗留时间和顾客平均等待时间

至此我们通过排队论中的M/M/1 模型推导得出了模型的重要指标如顾客平均逗留时间等

### 3.1.4 M/M/c 模型

在上一小节中，我们提到了M/M/1模型，然而日常生活中并不完全是只有一个服务机构，存在多个服务机构的情况是十分普遍的。例如银行排队叫号系统，在银行中有很多窗口可以提供服务，顾客到达后叫号排队等待，在这个案例中由于银行存在多个窗口可以提供服务，因此并不能够满足M/M/1模型中的假设服务机构只有一个的条件。在M/M/c模型中，规定各个服务窗口之间的工作是相互独立的并且平均服务率相同，即

如果将所有服务窗口看成一个整体服务机构，整个服务机构的平均服务率为

在M/M/c模型中令,这样才不会排成无限的队列，我们称为系统的服务强度。M/M/c的状态转移图如图3.6所示

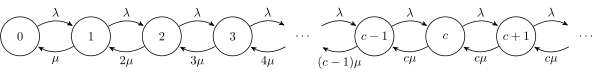


图3.6 M/M/c状态转移图

同样的，我们可以根据状态转移图来得到状态转移方程

用递推法来求解上述差分方程，可得状态转移概率，在此省略推导过程，得到各项指标如下

由此可以看出M/M/c模型是不同于c个M/M/1结合的，我们还以银行叫号服务系统为例。如果排队方式不变，顾客到达后不通过叫号系统而是在每个窗口前各排成一队，并且不换队，这就形成了3个M/M/1子系统。通过实际经验和求解结果可以得出单队列M/M/c在各个指标方面要优于3个M/M/1子系统。

## **3.2 云雾结合系统模型**

在我们的系统模型中，我们考虑要在满足一定的延迟下最小化系统能耗，因此我们把系统分为能耗和延迟两部分分别来计算。

### 3.2.1 系统能耗

在我们的模型中考虑的能耗有雾层设备，云服务器，车辆到RSU的传输能耗，下面我们将详细的介绍这些概念在我们模型中的定义。

（1）雾设备能耗：雾装置i的计算能耗可以通过CPU频率来衡量。我们用一个函数来近似模拟雾设备的能耗，如公式3.7所示，其中A,B预先选定的正数参数。

（2）云服务器能耗：如前所述，每个云服务器搭载着许多同类型的计算机，因此我们可以做出位于同一服务器上的这些机器CPU频率是相同的假设。我们用来表示云服务器j所包含的机器数。如此，云服务器的能耗可以表示为每台机器的能耗与云服务器所包含的机器的个数的乘积。我们通过将每个机器的CPU频率作为参数用一个函数来近似模拟机器的能耗，，其中和是非负数并且P在[2,3]范围内取值。在某个特定的时刻一些云服务器可能会接电开机来处理计算请求，而一些服务器可能没有请求需要处理，因此处于待机省电的状态。我们用一个二元变量来标识服务器的状态。具体来说，表示服务器上的机器处于待机状态， 表示服务器上的机器处于工作状态。因此，云服务器j的能耗可以表示为

从上式中可以看出云服务器的能耗主要和服务器内机器数量和机器的工作状态有关。

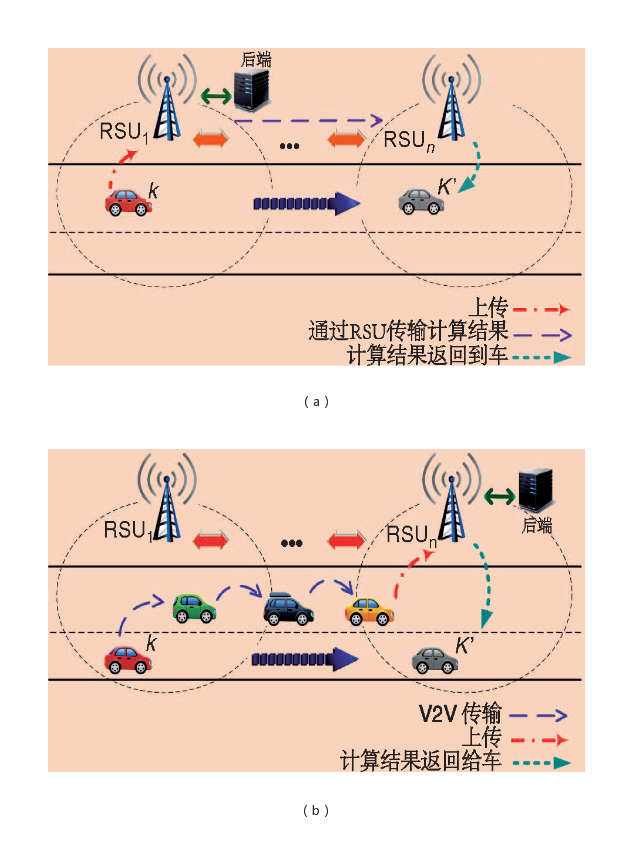


图3.7 V2I（a）和V2V（b）模式

（3）从车到RSU的传输能耗：网络中的物理基础设施主要包括路侧单元(RSUs)、雾层的处理节点(路由器)和车辆。令集合R、U分别代表路侧单元和雾层处理节点，其中R={1，…，R}，U={1，…，U}。从车辆到路侧单元有两种传输模式，一种是车辆直接与路侧单元进行通信，通过RSU转发后端计算结果至车行驶到的RSU,另一种通过预测请求处理的时间，通过车辆间接转发到请求处理后车辆即将到达的RSU范围内。我们把通过RSU进行转发的模式称为V2I模式，把通过车辆间接传递的模式成为V2V模式，具体如图3.7所示。我们可以计算出通过V2I模式传输的能耗消耗如下

其中和分别代表从源路测单元到目的路侧单元所需要经过的路侧单元（RSU）的跳数和每一次相邻RSU之间转发的能耗。

此外，与基于基础设施RSU接入服务不同，V2V模式是通过车辆自组织网络，在车辆与车辆之间进行传递，在此种模式下能耗是远远低于V2I模式的。在这种模式下从车辆用户u到路侧单元r 的传输能耗可以被定义为

其中代表了从车辆u到距离为j的目的路侧单元所需要中间车辆转发的跳数，是一个随着j的增大而不断增大的参数。注意j代表了从当前车辆所在的路侧单元范围到目的路侧单元范围所需要的跳数（即中间相隔的RSU个数），因此j>1意味着采用预测模型来进行传输。从车辆到RSU的传输能耗可以被归纳为

综上所述，通过我们的建模分析，可以将我们模型中的系统能耗归纳三部分，并对每一部分进行详细的阐述给出了合理的定义，并用数学公式进行量化。

### 3.2.2 系统延迟

在我们的模型中考虑的延迟主要包括三部分，分别是从车辆传输到目的路侧单元的时间、请求处理的时间（云和雾）、请求从雾节点传输到云上的时间。下面我们将详细介绍在我们的模型中如何定义这些变量。

（1）雾设备处理延时：我们假设雾设备按照排队系统来处理请求。对于某个雾设备，请求到达的速率为,服务率为，雾设备符合M/M/1模型则根据公式3.4 可以求解出请求的平均逗留时间，因此雾设备的处理定义如延时下

(2) 云服务器处理延时：云服务器的处理延时可以分为等待延时和处理延时，对于云服务器j首先我们要计算出其服务率 ，其中K是请求所需的平均转数。我们假定云服务器j中的机器服务率相同，我们将云服务器建模成符合M/M/c的排队模型，根据排队论公式3.6，可以计算出请求的逗留时间。

(3) 从车辆到RSU的传输延迟：对于车辆u和路侧单元r通过V2I模式传输延时；在V2V模式下传输延时可以归纳为，其中代表V2V模式内平均通过车辆转发每一跳的平均延时。代表了到距离为j的 RSU需要V2V传输的总跳数。因此从车辆到目的RSU传输的延迟可以近似为

其中是或代表选择传输的模式。

(4) 从雾到云服务器的传输延迟：我们用记录从雾节点i到云服务器j的延迟，并用二元变量表示请求从雾节点i分配到云服务器j，那么从边界节点到云的传输延迟如下

## 3.3 约束条件

对于我们所提出的模型而言，参数在许多方面具有一定的约束，主要分为雾设备参数的约束、云服务参数的约束、工作负载参数的约束。

雾设备的约束：对于雾设备来说由于物理资源的限制处理能力是有限的，我们用来代表雾设备i最大计算能力，用代表雾设备cpu最大转速则有

云服务器的约束：对于云服务器j，我们用和分别代表所有服务器的最大和最小CPU频率，则有

除此之外每个云服务器j内的机器数是一个整数并且不能超过上界

对于每个云服务器其状态是一个二元变量即

工作负载的约束：我们用L来表示从前端RSUs发送过来的所有请求，这些请求首先被发送到处在网络边缘的雾设备上。因此满足

我们用X,Y分别来表示分配给云服务和雾设备的工作负载则有。来自前端的请求可以被划分为两部分分别被云服务器和雾节点进行处理，对于每个云服务器工作负载为从每个雾节点分配到云的累计。

根据3.13 可以得出

对于车辆而言在V2V传输模式下，数据可以预先传送到距离当前RSU j跳以外的RSU，为了满足低延迟，跳数j具有一个上限值

## 3.4 问题描述

我们提出了车联网中云雾结合的模型希望能够在满足延迟的的前提下，最小化系统能耗。我们将系统延迟归纳如下：

系统能耗包括雾层、云服务器和车辆传输能耗。

因此可以将我们所提出的问题公式化归纳为式3.18

如上所述，式3.18是非线性规划问题，是NP难解问题。因此在这种情况下，随着规模的增大求解时间呈指数增长，求解时间是不可接受的。

# 4 解决方案

首先，我们将整个系统分为2部分：前端（车和RSUs）、后端（雾节点和云服务器）。前端的开销主要是车和RSUs通信之间的能耗和延时即和。我们引用一个基于预测的V2V、V2I结合模型来优化前端的能耗问题。车辆在发送请求之前，对请求返回的时长进行预估并假设当前是直行道路，RSU按照一定的间隔分布，那么可以根据汽车行驶速度来计算出请求返回时汽车所行驶到的RSU编号。通过预测出目的RSU来计算V2V和V2I两种不同模式的开销，并选取开销较小的模式。

后端包含多个雾节点和云服务器，我们首先采用一个贪婪的算法，对计算请求进行卸载分配。通过这种贪婪算法我们可以用我们的云雾结合模型和仅有云中心的模型在能耗方面、延迟方面进行对比。我们还提出了一种基于卷积神经网络的解决方案。下面我们将对这两种方案进行详细的介绍。

## 4.1 贪婪算法

贪心算法在每一步选择中，都选择当时最佳的情况作为决策，决策一旦做出便不可更改。因此通过贪婪算法可以得到局部最优的解。在我们当前的模型中可以对每一个请求进行贪婪求解并将它们合并起来计算系统的延时和能耗，其伪代码如表4.1所示

|  |
| --- |
| 算法：贪婪算法 |
| 输入：总请求集合Q={q1,q2,…, qn} |
| 服务端列表 S={s1,s2,…, sn} |
| 开始： |
| 1 ： for q Q： |
| 2 ： for s ： |
| 3 ： q->s将请求q分配到服务端s |
| 4 ： if 满足约束条件： |
| 5 ： 计算能耗 |
| 6 ： if 能耗<当前最小能耗： |
| 7 ： 更新最小能耗、记录分配结果 |
| 8 ： end for |
| 9： 更新服务端参数 |
| 10： end for |

表4.1贪婪算法伪代码

通过表4.1 可以看出在贪婪算法中我们将一段时间内的请求加入到请求集合中，并且对于每一个请求我们都将其放置到当前最优位置。这种算法所求出的结果虽然不是最优解，但其结果不会太差。

## 4.2 深度学习模型

尽管贪婪算法在一定程度上能够求解原有问题，但其优化的程度是不够的。因此我们提出了一种深度学习的模型来进一步进行优化，通过深度学习模型，我们可以为一组请求找打一组合适的服务方，并按照一定的规则进行请求分配。

### 4.2.1 输入与输出

我们所考虑的系统模型如图1 所示，在我们的模型中共有n个可供选择的边界计算节点和云服务器来处理来自前端的请求。因此每个设备可以记录前H个时期的参数（能耗、时延、到达率），我们用这些信息作为深度学习模型的输入，深度学习模型采用CNN模型。卷积神经网络包含特征提取和分类两大部分，在特征提取部分，利用卷积层对低值特征进行提取，池化层用于减少参数的个数以此加快网络的计算速度。，具体如图4.1（a）所示。

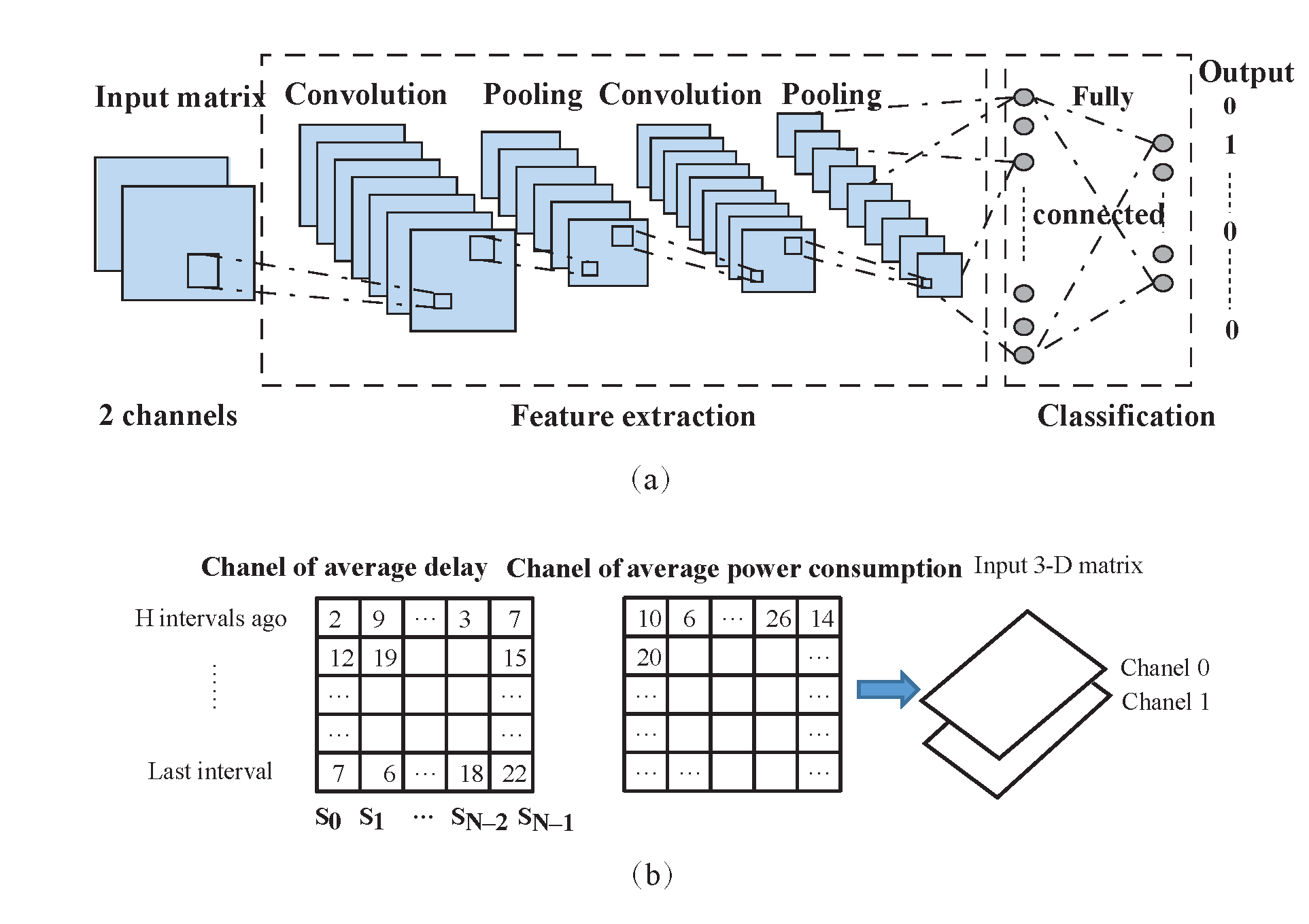


图4.1 提出的CNN结构

如前所述，我们把服务方的前H个时期的记录信息作为输入，每个服务方记录了H个时期如图4.1（b）所示，通过我们将具有2通道的H\*N的矩阵作为输入数据，采用有监督的学习方法进行训练，以此找到合适的服务方，并按照所选出的服务方来将请求进行分配。因此我们深度学习模型的输出是长度为N的向量，其中值为1位置的索引代表所选服务器的编号。

### 4.2.2 深度学习过程

在这部分中，我们关注如何应用CNN来计算选择服务器，全部过程可以分为3部分：初始化阶段、训练阶段、运行阶段。下面我们将对这三个阶段进行详细的阐述。

初始化阶段：在初始化阶段，我们需要获得数据来训练我们的CNN网络，在上一节中我们采用了有监督的学习方法来训练CNN。因此初始化阶段的目标是获得有标签的数据包含输入数据和对应的输出向量。为了获得这种训练数据，我们采用枚举求解的方法来求解最优解作为训练数据，这种方法在小规模网络中的耗时是可以接受的，但我们可以对这些数据进行预先求解。另一种方法是在我们所考虑的网络中运行传统的贪婪算法。通过这两种方法可以记录在时间段内的服务端能耗和延时以及最优解所选择的服务器。

训练阶段：在训练阶段我们用我们所获得的数据来训练CNN 网络，训练主要包含2个部分：参数的随机初始化、参数的调优。对参数的初始化是很有必要的，不对其初始化可能会减慢收敛速度，影响最终结果等。初始化过程采用normal高斯分布初始化，均值为0，方差根据不同的激活函数定义具体在仿真实验中。对于前馈神经网络，参数调优依赖于误差的反向传播，具体的调优算法有sgd和adam等。由于adam能够自适应收敛快，因此在本文中我们选取adam作为优化算法。训练的伪代码如表4.2所示

|  |
| --- |
| 算法：训练伪代码 |
| 输入：（x,y）={() |t=1,2,…,m },， |
| 输出： |
| 开始： |
| 1 ： for () (x,y)： |
| 2 ： predict = forward() |
| 3 ： loss = loss\_fun(predict,) |
| 4 ： backward() Fine-tune |
| 5 ： end for  6 ：return |

表4.2 训练过程伪代码

执行阶段：在运行阶段，所有服务方需要记录他们在一段时间内的平均延迟和能耗并且周期性地将他们发送给边界节点（可能是路由器）。这样每个边界节点可以将这些信息输入到CNN网络中前向传播计算得到选出的服务方，并将请求分配到所选出的服务器。然而并非每次执行CNN阶段都能选择出合适的服务器，有时选择出的服务器不能满足约束条件，当这种情况发生时则采用贪婪的算法作为补偿。整个深度学习的流程图如图

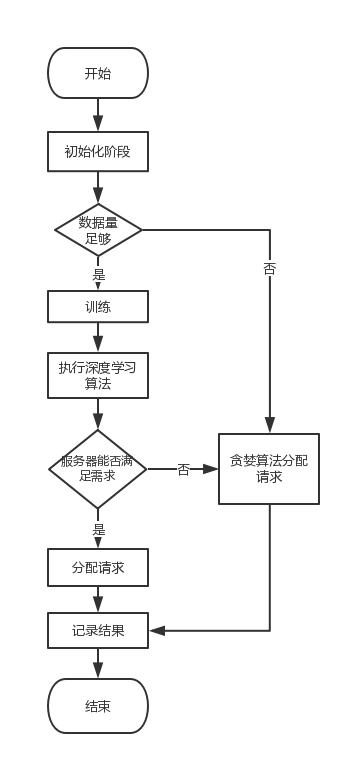


图4.2 深度学习流程图

# 5 仿真实验

在本章中，我们将会在我们提出的模型下用我们的解决方案进行仿真实验。通过仿真对比，可以看出我们的解决方案的优化程度。在本章中我们分别对前端和后端进行仿真。仿真所用的系统是Linux，使用python编程语言，用到的模块有pytorch、numpy、sklearn等。

## 5.1 预测模型

如前所述，在第3章和第4章中，我们提到了在前端我们采用一个基于预测的V2V、V2I模式结合的模型。在这个模型中我们对延迟和能耗定义如3.2.1和3.2.2所示，在这个模型中我们假设车辆延直线行进，并且路边按照一定间隔部署RSUs。

参 考 文 献

|  |  |
| --- | --- |
| 文献类型 | 标志代码 |
| 普通图书 | M |
| 会议录 | C |
| 汇编 | G |
| 报纸 | N |
| 期刊 | J |
| 学位论文 | D |
| 报告 | R |
| 标准 | S |
| 专利 | P |
| 数据库 | DB |
| 计算机程序 | CP |
| 电子公告 | EB |

[1] Zhang, Ke, et al. "Mobile-Edge Computing for Vehicular Networks: A Promising Network Paradigm with Predictive Off-Loading."[J] in *IEEE Vehicular Technology Magazine* 12.2 (2017): 36-44.

[2] Mao, Bomin, et al. "Routing or computing? the paradigm shift towards intelligent computer network packet transmission based on deep learning."[J] in IEEE Transactions on Computers 66.11 (2017): 1946-1960.