本科生毕业论文（设计）

题目：基于远程直接内存访问的高性能内存传输优化

姓　　名	兰 靖
学　　号	18340085
院　　系	计算机学院
专　　业	计算机科学与技术
指导教师	肖侬 (教授)

2022 年 3 月 27 日

基于远程直接内存访问的

高性能内存传输优化

High-Performance Distributed Memory

Transfer Using RDMA

姓　　名	兰 靖
学　　号	18340085
院　　系	计算机学院
专　　业	计算机科学与技术
指导教师	肖侬 (教授)

2022 年 3 月 27 日

学术诚信声明

本人郑重声明：所呈交的毕业论文（设计），是本人在导师的指导下，独立进行研究工作所取得的成果。除文中已经注明引用的内容外，本论文（设计）不包含任何其他个人或集体已经发表或撰写过的作品成果。对本论文（设计）的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。本论文（设计）的知识产权归属于培养单位。本人完全意识到本声明的法律结果由本人承担。

作者签名：

日　　期：　　年　　月　　日

【摘　要】

随着大数据处理、大规模智能模型训练、强化学习等多样化、高负载的计算需求成为学术界和产业界的焦点，作为计算基础设施的分布式计算框架（例如 Mapreduce, Spark, Ray）正在得到更多的关注和更广泛的应用。通常，计算框架软件负责为用户调度计算任务，同时管理和最大限度地利用计算集群的物理资源。上述场景所产生的快速增长的计算需求，正在使计算集群的性能愈发捉襟见肘：一方面，高性能计算 (HPC) 集群正在取代廉价的计算机器以加速高强度计算。值得注意的是，前者所拥有的新型高性能网络，明显地区别于传统集群。以智能网卡为硬件基础，高性能计算机能够在无中央处理器（CPU）的介入下，对机器内存直接读写和传输数据。这种被称为远程直接内存访问（RDMA）的传输机制，相比以太网具有低占用、低延迟、高带宽等技术优势。另一方面，作为基础设施的软件框架，也在面临严峻的性能挑战。当前大多数分布式计算框架仍然缺乏对高性能硬件的支持，特别是无法充分利用集群中的高速网络资源。因此，本研究选取了分布式计算框架 Ray 的核心组件：分布式内存对象数据库 Plasma，首先测试和分析了其使用以太网传输数据而导致的性能瓶颈。然后，在高性能集群上，我们提出了一种支持 RDMA 特性的高吞吐数据传输机制。该机制针对大型数据，提出了基于单边读语义的传输协议，实现了用户态的内存零拷贝。并且该机制能够在运行时，根据数据大小动态地选择传输协议，以获得最佳性能。进一步，我们基于并行计算范式 MPI 构造了分布式、可扩展的多节点传输性能测试。在实验中，我们确定了传输协议选择的最优参数，并且展示了在天河高性能集群上，优化后的内存平台能够在常见大小的对象传输中实现至多 8 倍的吞吐率提升。

关键词： 数据传输，高性能网络，RDMA，键值对数据库，分布式缓存

[ABSTRACT]

Keywords: data transfer, high-performance network, RDMA, key-value database, distributed memory

目录

1 绪论 1
1.1 选题背景与意义 1
1.2 国内外研究现状和相关工作 2
1.3 论文主要研究内容 6
1.4 论文结构与章节安排 7

2 Plasma 分布式内存存储架构和性能分析 8
2.1 Plasma 架构分析 8
2.2 基于套接字的 Plasma 通信机制 10
2.3 Plasma 存储和传输性能分析 11

3 LaTeX 模板配置与使用 12
3.1 使用 Overleaf 编写毕设 12
3.2 编译环境配置 14
3.3 写作环境配置 14
3.4 如何开始写毕业论文（设计） 15

4 简单的使用例子 17
4.1 图像的插入 17
4.2 表格的插入 20
4.3 公式 20
4.4 算法流程图 21
4.5 例子、定理与证明 21
4.6 代码 22
4.7 其他的一些用法 22

5 其他注意事项 24
5.1 关于生僻字 24

6 实验与结果 25

参考文献 26

附录 A 补充更多细节 27

A.1 补充图 27

致谢 28

插图目录

2.1 Plasma 存储架构 8
2.2 基于套接字的 Plasma 通信机制 10

3.1 在 Overleaf 上创建并上传压缩包。 12
3.2 在 Overleaf 上调整编译工具 13
3.3 Overleaf 使用例子 13
3.4 vscode 配置好后的样例 15

4.1 镶嵌在文中的图像 17
4.2 单张图像 18
4.3 并排的多张图像 18
4.4 并排的多张图像加各自的注解 18
4.5 复杂的两列对象的插入 19

A.1 一个配有彩色表格的插图 27

表格目录

2.1 Plasma 客户端接口 9
2.2 Plasma 集群接口 9

4.1 典型的实验对比表格 20
4.2 复杂一些的表格 20

1 绪论

1.1 选题背景与意义

分布式计算框架是一种复杂的平台软件。传统的并行计算范式如信息传递接口（MPI）和分区全局地址空间（PGAS），通常为编程者提供丰富的调用接口和灵活的编程空间。然而，使用这些编程标准的门槛过高：编程者通常需要自己管理多台机器的状态，特别是内存的分配和使用；编程者还需要使用给定的标准原语，精确地规定多个进程之间的通信和协作方式，这通常需要大量时间和精力；另外使用这些范式将导致程序和功能的强耦合——编程者很可能不得不重新编写代码来更新程序的逻辑。因此，现代分布式计算框架通常承担了上述硬件管理的角色，并针对目标任务类型，为用户设计尽可能少、但简单易用的功能接口，来降低集群的使用难度。传统的分布式计算框架，例如 Mapreduce 和 Spark，都是面向大规模数据分析而设计的。然而，近些年以强化学习、复杂工作流等为代表的复杂计算需求，需要更灵活的框架支持。加州大学伯克利分校 RISELab 实验室提出的 Ray 和芝加哥大学 Globus 实验室提出的 Parsl 是其中两个典型。这些框架吸取了云计算领域“函数即服务（Function as a Service, FaaS）”的设计思想，能够细粒度地以函数为单位将任务调度到集群上执行，同时依然对用户隐藏绝大部分实现细节。Ray 通过分布式内存对象存储 Plasma，实现了框架完全自主的集群内存管理，让用户可以专注于实现功能逻辑，无需担心数据的存放和移动。Ray 在保持易用性的前提下，极大地提升了计算框架的灵活性，用户只需要修改几行代码就能将单进程程序扩展到整个集群，进而实现分布式机器学习等复杂模型。

高性能集群（或超算集群），是一种以高端处理器、计算卡、高性能网络、大容量存储为核心硬件的计算集群。随着大数据应用的丰富、超大规模人工智能模型的出现，高性能集群和超级计算机正在变得愈发重要。首先，高性能计算机拥有普通机器不能比拟的计算能力，主要表现为高端的多核处理器和计算卡。而随着大模型时代的到来，新应用对集群网络的需求快速提高，高性能网络所表现出的高带宽、低延迟等特性也逐渐受到了更多的关注。在超算集群中，网络性能上的优势很大程度上来自于对远程直接内存访问（RDMA）机制的支持。然而，要利用 RDMA，用户必须在应用中实现基于 RDMA 的通信机制，而不能依赖于操作系统内核提供的系统调用。因此，基于套接字（Socket）的网络应用大多不能在高性能集群中直接获得显著的性能提升。分布式计算框架 Ray 并不例外，其分布式内

存存储 Plasma 目前仅有对传统 TCP/IP 协议的支持，因而在超算集群上无法发挥出应有的网络性能，进而影响 Ray 运行在超算上的总体性能。

当前，超算集群普遍使用的是英伟达（Nvidia）公司的 Infiniband 高速网络。对于使用 Socket 通信的网络程序，该架构通过“基于 Infiniband 的互联网协议（IPoIB）”实现支持。值得注意的是，这是一种依赖操作系统内核的非原生支持，已经有多个工作表明，其网络性能和直接使用 RDMA 技术相比具有明显差距。因此，本研究的目的是：我们是否能为分布式内存存储 Plasma，提出并实现一种支持 RDMA 机制的内存通信协议，从而让 Plasma 乃至整个 Ray 框架在现代超算集群上获得更好的性能？从超算研究的趋势来说，应用软件和先进超算硬件之间的隔阂，正在逐渐成为大家关注的热点。随着超级计算机和云计算两个领域的融合，会有越来越多的软件运行在高性能集群中。然而，它们中的大部分还没有针对高性能硬件提供软件支持——通过提供软件对高性能硬件的支持，我们能够将这些应用的运行性能提升到全新的水平。

1.2 国内外研究现状和相关工作

1.2.1 分布式计算框架

分布式计算框架是一种复杂的平台软件。在底层，它通过实现任务调度、并发执行、内存管理等基本组件，向用户透明地提供分布式计算的功能；在应用层，它通过设计和规范一系列接口，帮助用户高效地运行特定任务：例如大数据分析（Mapreduce、Spark），分布式机器学习（Ray）等等。Ray 是近年来最受关注的分布式计算框架，它以函数为单位调度任务执行，完全自主地管理内存移动，并且支持异步执行。借助这一平台，目前社区人员已经实现了机器学习（Ray ML）、强化学习（Ray RLlib）、模型部署（Ray Serve）、工作流（Ray Workflows）等上层应用库供用户使用。同时，用户也可以直接在框架上编写任意程序：使用常见的 Python 语法构建出完全分布式的计算程序，而且不会有任何表达能力上的限制。

代码 1.1 Ray 代码示例

```
import ray
ray.init()

@ray.remote
def f(x):
    return x * x

futures = [f.remote(i) for i in range(4)]
```

```
print(ray.get(futures)) # [0, 1, 4, 9]
```

在上述代码片段中，用户通过修饰符”@ray.remote” 将函数定义为远程函数，并在下方执行 4 次。值得注意的是，在 Ray 框架的调度下，这四次调用将会逐一调度到不同的进程上并发执行，因此 Ray 能够透明地为任何函数提供并行计算能力。而分布式内存存储 Plasma，则在系统中持续为并发任务调度所需的数据。在示例代码中，Plasma 将会在集群中查找变量 i 的位置，并将其拉取到进程中供任务使用。最后，Ray 原生支持异步执行，任何函数调用都会立刻返回，给予用户一个凭证（future）——用户可以立刻用这一凭证作为参数继续调用其他函数，即使数据的真实值还没有求得。Ray 的执行引擎动态地构建和解决数据依赖，将凭证替换为真实值，然后继续调用依赖它的函数。异步的特性将使 Ray 充分发挥并发任务的性能。

1.2.2 远程直接内存访问技术（RDMA）

RDMA 全称 Remote Direct Memory Access，即远程直接内存访问，是一种于二十一世纪之后逐渐兴起的新型网络通信技术。其核心思想和直接内存访问（DMA）相似——在早期计算机系统当中，内存读写、移动都需要由中央处理器（CPU）直接操作，从而带来相当的性能开销。而现代计算机系统分离出了内存子系统，将内存相关的负载卸载到子系统的专用硬件上。因此，CPU 只需要在访问开始和结束时同子系统协作，在漫长的访问延迟中可以执行其他计算任务，从而大大提高了计算机的总体性能。RDMA 技术在这一方向上更进一步：支持 RDMA 的现代智能网卡，不仅能从网卡端直接寻址并读写本机内存，还能够和远端的另一个网卡协作，直接读写远端机器的内存空间，从而实现通信。这一过程中 CPU 通常只需极少的介入、甚至完全不需要，因此具有相当明显的技术优势：

1) 旁路内核（Kernel-bypass）。基于 TCP/IP 协议的网络通信已经逐渐不能适应现代高并发、重负载的网络应用。基于高速 SSD 的存储系统、基于内存的缓存和（键值对）数据库都需要在短时间内应对大量并发的 I/O 操作请求，工作表明这些应用的性能瓶颈位于 CPU，而不是 I/O 部分。臃肿的网络栈、用户态-内核态的切换、内核态的数据处理和拷贝等等，是影响网络系统性能的根本原因。智能网卡能代替 CPU 执行内存移动，而且 CPU 和网卡的大多数交互都实现在用户态，大大减轻了 CPU 的负担。
2) 零拷贝（Zero-copy）。零拷贝是当前操作系统领域的一项热门技术，其思想在于提高内存的共享程度，例如减少进程-进程、用户态-内核态内存拷贝的数

量，从而提高 I/O 的整体性能。在支持 RDMA 的应用中，由于内核旁路，一次内存操作往往能省去：用户缓冲到内核缓冲的拷贝、内核缓冲到硬件（驱动）缓冲的拷贝。数据直接在两端主存之间移动，因此产生可观的性能提升。

3) 低延迟、高并发。简化的网络路径、零拷贝等特性大大降低了机器操作远端内存的延迟。在并发应用中，更低的延迟意味着相同时间内更强大的并发处理能力。
4) 异步通信。RDMA 是原生异步的通信机制，进程需要主动访问完成队列（CQ）甚至直接检查内存数据，才能得知通信的发生。这一特性让程序的并发潜力大大提升，但于此同时，编程者需要自己设计同步机制来完成通信，同样增加了编程上的难度。

目前，RDMA 技术是智能网卡技术中较为成熟的一种。硬件支持 RDMA 的网卡通常都具有相当惊人的网络带宽以及其他诱人的硬件特性：目前，Mellanox NDR Infiniband 网卡能够支持高达 400Gb/s 的网络带宽；另外，Infiniband 标准实现了链路层的容错机制，这意味着通常意义上的丢包在 IB 网络上并不存在，大大降低了用户设计通信机制的难度。加上近十年来新型硬件的价格逐渐走低，学术界和工业界争相尝试这一新技术，并已经有了相当多优秀的成果。尽管 RDMA 存在着其他实现方式，例如基于以太网的 RoCE，但本研究中的 RDMA 机制在硬件上基于超算集群中广泛使用的 Mellanox Infiniband 架构。

基于 RDMA 机制的网络编程，目前主流的做法是使用 Verbs 操作原语。在 Infiniband 网络架构中，两端以队列对（Queue Pair，QP）为基本模型进行通信，一次简单的发送-接收可以描述为如下步骤：

1) 两端程序各自创建一个队列对，并借助套接字等基础通信方式，交换队列对的基本信息，建立一对连接。
2) Infiniband 通信的基本单位是一个（发送/接收/完成）事务：接收方通过 Verbs 调用将一个构造好的接收事务压入到接收队列（Receive Queue）中
3) 之后发送方将一个构造好的发送事务压入到发送队列（Send Queue）中。
4) 此时，硬件将为两端处理这一对事务，将本机内存从发送事务指定的内存地址发送到接收事务指定的远端内存地址。
5) 硬件完成一次事务操作后，通常将一个完成队列项（CQE）压入到完成队列（Completion Queue，CQ）中。
6) 用户进程可以在任何时刻通过弹出完成队列的头部来确认（发送/接收）事务执行完成，或者得到操作失败的错误代码。

以上流程实现了类似 TCP/UDP 的双边通信语义，不过，两者仍然有着本质的

区别：RDMA 通信机制是基于事务模型的；以上操作均在用户态完成，无需任何系统调用；数据从一个进程直接发送到另一个进程。除此之外，RDMA 机制还支持使用读/写（Read/Write）等单边通信语义。在执行这些单边操作时，远端应用不需要事先发送接收事务，也不会通过完成队列得知通信的发生。使用单边操作语义将会进一步降低远端 CPU 的介入和负担，从而支持更高强度的并发操作。不过，使用单边语义通常需要引入额外的同步机制以完成通信。

1.2.3 基于 RDMA 技术的内存系统

近年来，针对 RDMA 机制提供的高带宽、低延迟优势，研究人员在多个方向上尝试将其转变为实际应用上的性能提升。特别是在 RDMA 同时支持双边和单边语义的情况下，如何针对实际应用场景，设计合适的通信和协作机制，一直是这一领域研究的重点。从场景的角度来说，目前的主要研究方向是分布式内存系统和高并发的内存系统。

1.2.3.1 分布式内存系统

Infiniband 网络标准和 RDMA 技术最早应用于高性能计算（HPC）领域。俄亥俄州立大学的[] 首次将 RDMA 技术应用于优化消息传递接口（MPI）的通信机制，这也是最早的、较为完善的一个 RDMA 协作机制实现。这一设计针对 RDMA 单边操作所产生的协作困难问题，预先建立了固定映射的发送-接收缓冲区对，以及设计了自描述的消息块，深刻影响了后续 RDMA 通信方案的设计。并且在 MPI 多年的演进中，其 RDMA 通信机制也不断有优化方案提出。

然而，从集群内存空间的角度来看，RDMA 技术的逐渐成熟，让研究者看到了颠覆性、普适性优化的可能性：优异的访问带宽和极低的访问延迟，已经极大程度地拉近了机器内存之间的“距离”，使得集群规模的共享内存逐渐成为可能：[] 发现 IB 网络的通信延迟和 PCIe 总线上的通信延迟处于一个数量级，这说明网络已经不再是集群通信的性能瓶颈。以 RDMA 为核心，HPC 社区和分布式系统社区均提出了一些分布式共享内存的实现方案，从而以较低的性能代价连接起整个集群的内存空间：前者包括 GasNet，OpenSHMEM 等支持分区全局地址空间（PGAS）计算的中间件；后者以 FaRM，CoRM 等系统为代表。这些研究的共同焦点是分布式的事务机制——除了本地内存的读-写冲突之外，还需要解决远端机器读写和本地访存之间的数据冲突。此外，也有研究避开了细粒度的内存管理，如去中心化、可扩展的分布式页表系统 Infiniswap，就通过粗粒度的内存页交换处理集群中存在

的物理内存使用不均的问题。

1.2.3.2 高并发内存系统

在另一个方向上，研究人员希望将 RDMA 低延迟、高并发的特性直接变为应用的性能——内存键值对（key-value）数据库是一个合适的领域。Redis，Memcached 等内存数据库通常被用作一种高速缓存，其他机器对数据库的访问也主要以并发读取为主，因此非常适合使用 RDMA 技术进行优化，特别是使用单边读等通信方式优化单机并发能力。不过，基于单边操作的读取机制难以应对高强度并发中的数据冲突问题，需要借助于更好的哈希函数等其他优化手段才能获得较好的优化效果。因此近期的工作更是提出了双边语义和单边语义混合的通信机制，从而在 CPU 负载、并发能力和编程难度上都获得较好的结果。总体来说，在过去的一段时间，针对应用场景不同，研究人员已经提出了众多不同的 RDMA 通信范式。

1.3 论文主要研究内容

作为分布式计算框架 Ray 的核心组件，Plasma 并没有对 RDMA 通信的支持，因而在超算集群中存在性能提升的空间。该组件兼有分布式、高并发两方面的特性：作为集群内存存储，Plasma 需要运行在集群的每个计算节点上，并且期望实现最大的并发传输能力，以支撑 Ray 快速的任务调度。不过，Plasma 并没有实现细粒度的并发机制，而是进一步依靠 Redis 等外部机制实现。这使得我们的研究重点倾向于优化大型数据对象的传输性能。在本文中，我们为 Plasma 提出了一种原生支持 RDMA 技术的通信机制，并且在现代超算集群上验证了其在各个数据大小的传输上都获得了更优的性能。

这一工作存在以下挑战：

1) 目前 RDMA 编程仍然是极为“小众”的技术，如何能在有限的资料和现有研究帮助下实现高性能的网络通信机制。
2) 如何在尽可能不破坏项目整体结构的情况下，为 Plasma 提供原生 RDMA 通信机制。这要求优化后的程序可以无缝地运行在以太网和 Infiniband 两种网络架构上。
3) 针对 Ray 框架中可能出现的大小不一的数据，如何实现该机制使得 Plasma 能够在尽可能多的大小范围内都能获得最优的网络性能。

下面总结了本工作的主要贡献：

1) 我们通过实验分析了原 Plasma 实现在超算集群上的存储和网络性能，验证

了其无法充分利用 Infiniband 高速网络，从而证明了使用 RDMA 技术优化 Plasma 数据传输的可行性。

2) 我们分别针对小型数据和分布式训练中常见的大型数据，分别实现了基于双边和单边通信的传输机制。针对大数据，单边通信将实现用户态的零拷贝特性，从而降低了 CPU 数据拷贝而导致的负载和占用时间。
3) 我们基于消息传递接口 MPI 实现了分布式、可扩展的数据传输性能测试，比较了优化实现和原实现在各个数据大小上的性能。并且通过实验，我们确定了机制在选择传输方式时的最佳方案，从而获得了最优的整体性能。

1.4 论文结构与章节安排

本文共分为五章，这些章节的内容安排如下：

第一章：绪论。简述了本文的研究背景和意义，简述了本研究的核心技术背景，并介绍了国内外相关工作和研究现状。

第二章：Plasma 分布式内存存储的架构和性能分析。这一章将简要分析 Plasma 的分布式架构，并且通过性能测试和常见内存存储 Redis 进行对比，分析了 Plasma 在传统网络结构上的性能瓶颈。

第三章：基于 RDMA 技术的混合通信机制实现。这一章将详细介绍基于 RDMA 技术的优化方案，并针对大小数据提出混合通信机制的实现。

第四章：实验和分析。这一章将在天河高性能集群上验证优化方案的性能优化，

第五章：总结和展望。这一章将总结本文的主要结果，并且进一步分析后续的工作方向。

2 Plasma 分布式内存存储架构和性能分析

2.1 Plasma 架构分析

Plasma 分布式存储架构由多个进程组成。通过将控制面、数据面上的任务解耦到不同的进程，我们可以较为方便地对其软件架构进行改进。图 2.1展示了 Plasma 集群的组织结构。

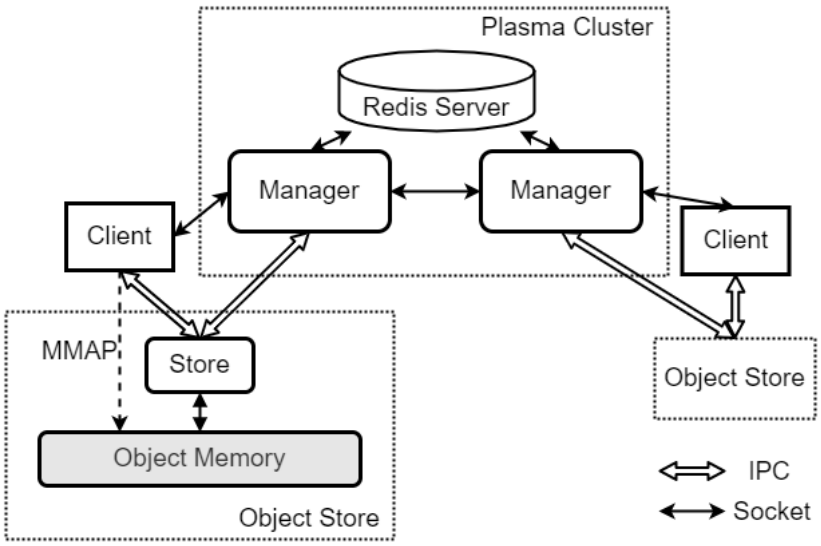


图 2.1 Plasma 存储架构

2.1.1 Plasma 对象存储架构

对象存储（Object Store）是 Plasma 运行的最小单元，即 Plasma 在任意单节点上运行时，可以不需要连接到管理进程（Manager）和 Redis 数据库。图 2.1中左下方部分展示了这一架构。对象存储的基本架构遵循用户端-服务端的特征。Store 进程负责直接操作和管理机器内存。当用户（Client）进程发起创建对象请求时，会将关键元数据——即对象编号（id）、元数据大小、数据大小等通过进程间通信（IPC）发送给 Store 进程。值得注意的是，Plasma 使用 Unix 域套接字（Unix Domain Socket）作为进程间通信方式，这一方式允许用户端以类似套接字的方式，通过操作系统内核与本机内的另一个进程交换数据。

Plasma 利用 MMAP 映射机制对内存对象操作进行了性能优化。当前常见的内存数据库如 Redis，用户端进程和服务端需要直接交换数据以完成创建、读取等操

作。而 Plasma 利用了 MMAP 机制，使得客户端与服务端能够共享内存空间：

1) 创建对象时，服务端使用 MMAP 映射分配内存空间，将其与一临时文件映射。
2) 服务端通过 IPC，将上述临时文件的文件描述符（file descriptor，fd）传递给用户进程。
3) 用户进程使用相同的文件描述符 fd 再次调用 MMAP。

如此，MMAP 向 Store 进程和用户进程返回同一个内存地址，用户进程能够直接读取、操作数据库中的内存对象。这一架构能够将数据库读取操作的开销优化到 $O(1)$ 时间复杂度，使 Plasma 针对大型对象仍然具有优秀的吞吐能力。Plasma 在对象存储架构中定义了如下操作原语：

表 2.1 Plasma 客户端接口

接口定义	接口描述
plasma_contain(id)	查询对象是否存在
buf ← plasma_create(id, fields, values)	创建参数/数据为 fields/values 的对象
plasma_seal(id)	封装对象使其对其他进程可见
plasma_release(id)	释放对象，引用计数减一（为 0 则删除）
buf ← plasma_get(id, fields)	读取对象
plasma_delete(id)	删除对象
plasma_connect(store_addr, manager_addr)	连接到 Store 和 Manager
plasma_subscribe(id)	等待一个对象在本机被创建

2.1.2 Plasma 集群架构

Plasma 存储将其集群架构单独实现为 Manager 进程，多个进程以及一个 Redis 数据库共同构成其集群架构，图 2.1中上方展示了将 Plasma 连接成集群的架构。Plasma 在集群架构中定义了如下操作原语：

表 2.2 Plasma 集群接口

接口定义	接口描述
buflist ← plasma_fetch(ids)	将 id 列表中的对象拉取到本地存储

管理者（Manager）进程仅处理用户进程发出的一种请求，即输入一个 id 列表，执行一次拉取（fetch）操作：

1) Manager 对列表中的每个 id，从集群中查找对应数据对象的位置。
2) 本地 Manager 与存有这一对象的另一个 Manager 建立连接，拉取数据。

3) Manager 和 Store 交互将数据保存到本地——对 Store 进程来说，Manager 只是普通的 Client 进程。

在这一过程中，Plasma 依赖于 Redis 提供的数据库服务。每当 Client 进程创建并封装数据对象时，它将借助 Manager 与 Redis 服务器建立通信，并更新该对象在集群中的分布。因此 Manager 在拉取数据的过程中（上述步骤 2），必须先从 Redis 得到数据分布，才能开始一次数据传输。

2.2 基于套接字的 Plasma 通信机制

Plasma 实现了基于套接字（Socket）的数据传输机制。通过访问 Redis 服务器获得对象处在的目标节点后，Manager 会逐一尝试建立 Socket 连接，并拉取数据。其通信机制如图 2.2所示：

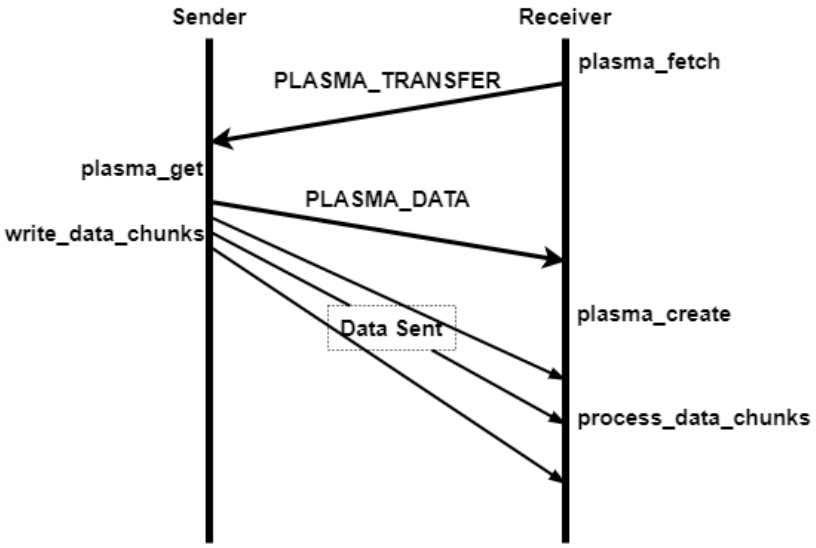


图 2.2 基于套接字的 Plasma 通信机制

发起者（Receiver）首先将一条类别为 PLASMA_TRANSFER 的消息发送给存有数据对象的 Manager 进程。在得知需要发送的对象 id 后，发送方通过向 Store 调用查询操作获得了数据的缓冲区地址。需要注意的是 Plasma 的读取操作具有常数的时间复杂度，因而会马上返回。发送方（Sender）会将查询到的元数据（主要是数据大小）通过 PLASMA_DATA 消息返回给接收方，接收方便会向 Store 创建一个同 id 的对象，获得分配的内存空间。最后双方分别进入发送/接收函数，分批将数据转移到本地缓冲区中。

在 Plasma 实现中，默认一次发送 4kb 大小的数据。

2.3 Plasma 存储和传输性能分析

为了分析 Plasma 存储运行在超算上可能存在的性能瓶颈，我们在天河高性能集群上对 Redis 和 Plasma 进行了存储和传输上的性能测试。

2.3.1 存储性能测试

2.3.2 数据传输测试

3 LaTeX 模板配置与使用

本部分内容将让你能够通过本 LaTeX 模板生成一份可用的 pdf，并为后面修改源码撰写毕设做准备。

首先，我们会展示最简单的方法：直接使用 overleaf 进行编写。然后，我们整理了不同环境下 LaTeX 环境的配置指南与不同写作工具的配置技巧，方便各位同学使用本 LaTeX 模板。最后，我们说明了如何开始编写自己的毕业论文（设计）。

3.1 使用 Overleaf 编写毕设

Overleaf[①]是一个在线的 Latex 文档协作平台。我们不需要配置任何环境，便能够在上面直接使用本模板进行写作。操作步骤如下：

第一步，下载本项目压缩包（从`https://github.com/SYSU-SCC/sysu-thesis/releases`处下载即可），注意需要下载 zip 格式的压缩包。然后，我们在 Overleaf 上新建项目，并上传该压缩包，可参考图 3.1。

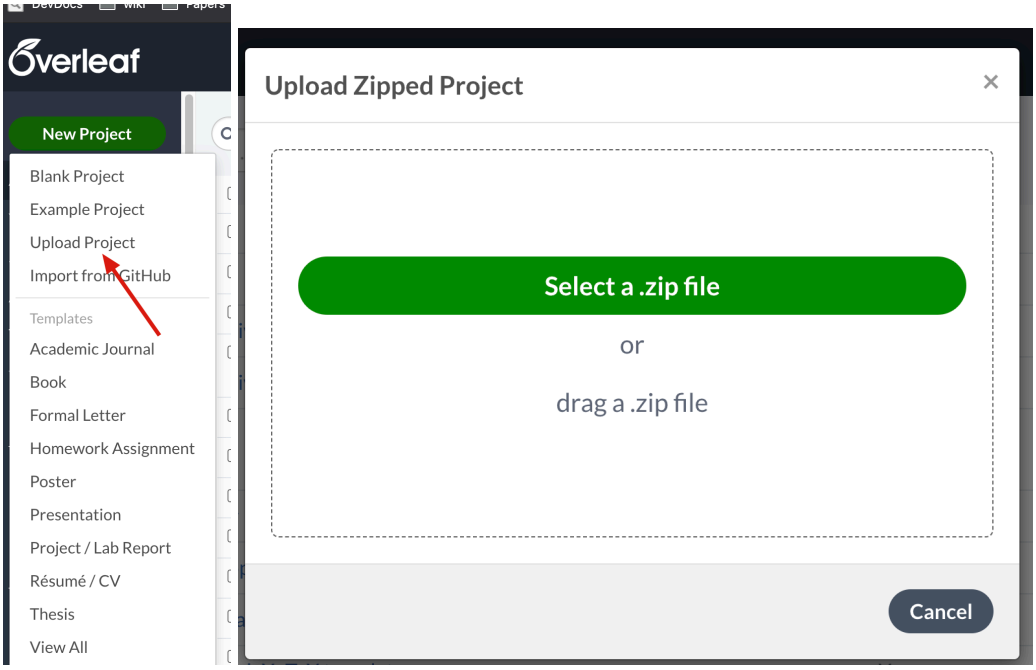


图 3.1 在 Overleaf 上创建并上传压缩包。

第二步，在 Overleaf 的菜单中调整编译工具为 `xelatex`，可参考图 3.2。

第三步，点击编译，得到本 pdf，可以开始修改 pdf 了！最终可见图 3.3。

[①] 网址可见`https://www.overleaf.com/`

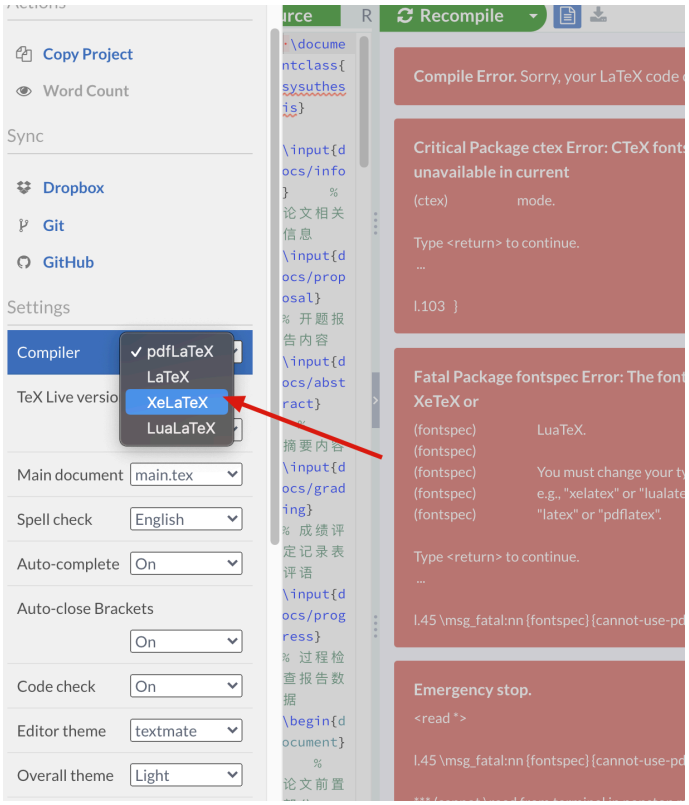


图 3.2 在 Overleaf 上调整编译工具

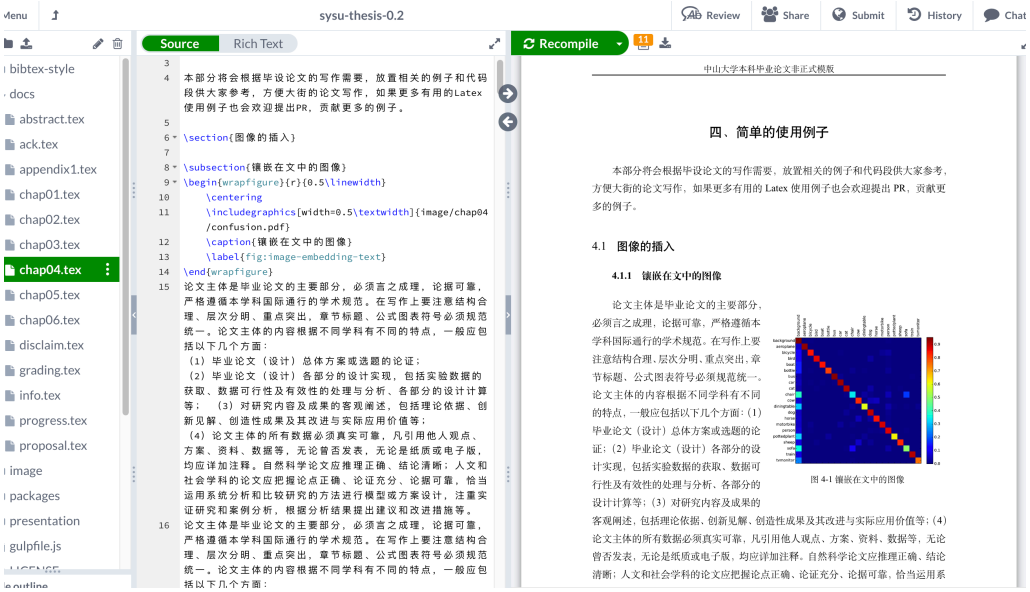


图 3.3 Overleaf 使用例子

3.2 编译环境配置

编译环境配置相对来说比较简单，下载 Tex Live2020 并如同一般的程序一样安装即可。

3.2.1 编译环境配置：Window 篇

在https://mirrors.tuna.tsinghua.edu.cn/CTAN/systems/texlive/Images/上下载 Tex Live2020 并参考教程[①]安装即可。

3.2.2 编译环境配置：Linux 篇

在https://mirrors.tuna.tsinghua.edu.cn/CTAN/systems/texlive/Images/上下载 Tex Live2020 并参考教程[②]安装即可。

3.2.3 编译环境配置：MacOS 篇

在 MacOS 上配置 Latex 的环境，这里我们使用的是 MacTex。

1) https://www.tug.org/mactex/下载 MacTex 安装。
2) 安装步骤：不详细展开，按照图形界面点击即可, 傻瓜式安装。

TIPS：MacTex 文件比较大，有 2G 多，介意的话可以选择 MacTex_Basic 包，只有 100M 以内，但是如果安装 MacTex_Basic，后期可能会遇到各种缺包的问题。

安装完成之后，可以简单测试一下安装是否成功。如可以查看 Texshop 应用是否安装好，或者在命令行测试一下 `xelatex` 命令是否可用。

3.3 写作环境配置

不同的写作工具对应不同的写作环境。这里我们给出几个工具的配置例子以供参考。

3.3.1 模板编译流程

由于 LaTeX 的限制，本模板需要经过四次编译才能生成完整的论文：

1) 先使用 xelatex 编译一次
2) 再使用 bibtex 编译一次

[①] 可以参考https://zhuanlan.zhihu.com/p/58811994
[②] 可以参考https://zhuanlan.zhihu.com/p/55894177

3) 然后使用 xelatex 编译两次

本编译流程已经写在 Makefile 中，修改模板源码后只需要执行 `make pdf` 即可按照该流程进行编译并生成最终的 pdf。

3.3.2 写作环境配置：Visual Studio Code

Visual Studio Code 是微软公司推出的轻量代码编辑器，我们可以做一些简单的配置，便可以用该编辑器修改我们的 LaTeX 模板，并实现一键编译。

1) 安装 Visual Studio Code。

2) 安装 LaTeX Workshop 插件。

本项目的`.vscode/setting.json` 下已经包含了与前面所述编译流程相同的配置。正常配置下，每次修改模板源码后按下保存（Ctrl+S），就能够自动进行编译产生 pdf。效果图如图 3.4所示。

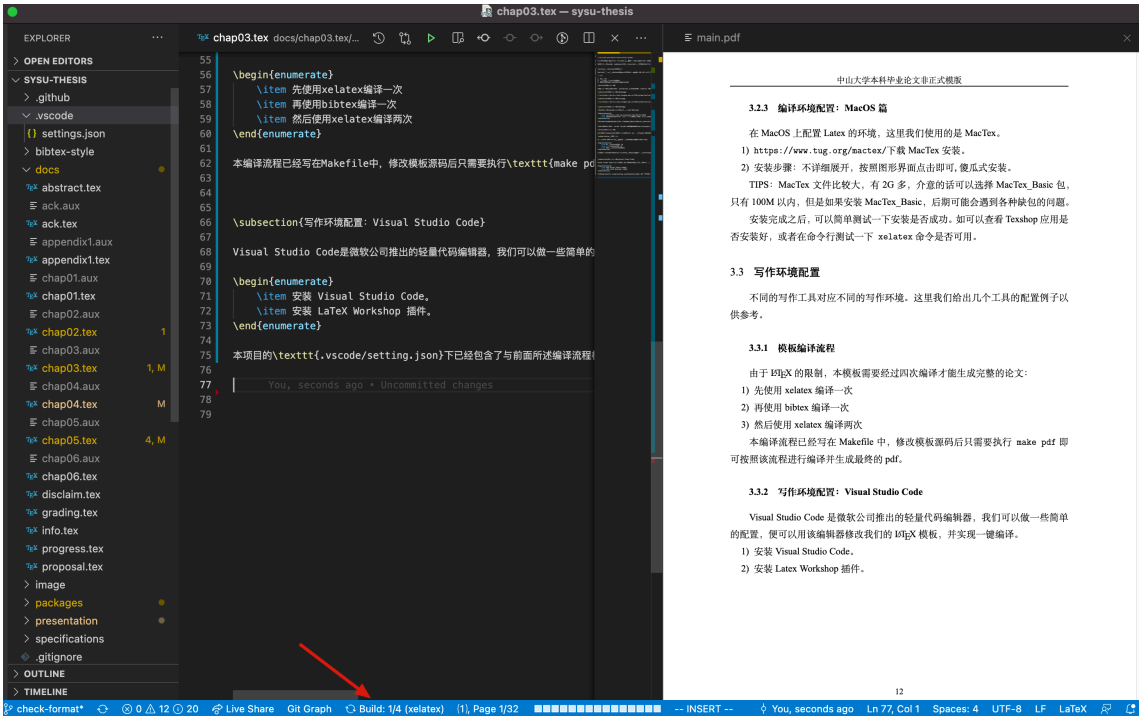


图 3.4 vscode 配置好后的样例

3.4 如何开始写毕业论文（设计）

首先将所有个人信息，包括学号、姓名、专业、论文题目等，在`./docs/info.tex` 中逐项进行更新。

然后我们再编辑`./docs/abstract.tex` 补充论文摘要。

到了论文主体部分,我们可以自行编辑./docs/chap01.tex,./docs/chap02.tex 等文件进行编辑。如果章数不够，可以自行修改 main.tex 增加新的章节。

当论文主体编写完成后，我们再编辑./docs/ack.tex 作为论文致谢。

4 简单的使用例子

本部分将会根据毕设论文的写作需要，放置相关的例子和代码段供大家参考，方便大家的论文写作，如果更多有用的 Latex 使用例子也会欢迎提出 PR，贡献更多的例子。

4.1 图像的插入

4.1.1 镶嵌在文中的图像

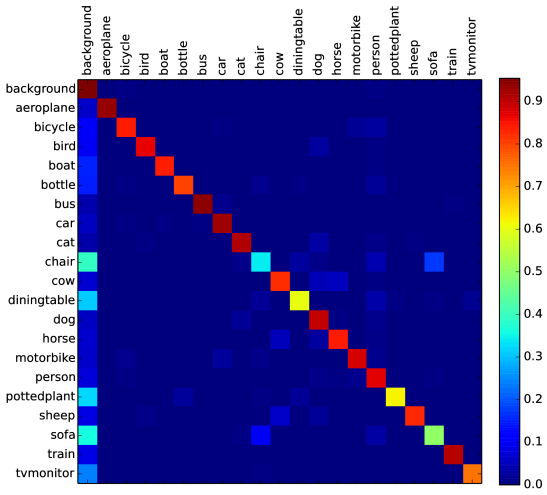


图 4.1 镶嵌在文中的图像

论文主体是毕业论文的主要部分，必须言之成理，论据可靠，严格遵循本学科国际通行的学术规范。在写作上要注意结构合理、层次分明、重点突出，章节标题、公式图表符号必须规范统一。论文主体的内容根据不同学科有不同的特点，一般应包括以下几个方面：（1）毕业论文（设计）总体方案或选题的论证；（2）毕业论文（设计）各部分的设计实现，包括实验数据的获取、数据可行性及有效性的处理与分析、各部分的设计计算等；（3）对研究内容及成果的客观阐述，包括理论依据、创新见解、创造性成果及其改进与实际应用价值等；（4）论文主体的所有数据必须真实可靠，凡引用他人观点、方案、资料、数据等，无论曾否发表，无论是纸质或电子版，均应详加注释。自然科学论文应推理正确、结论清晰；人文和社会学科的论文应把握论点正确、论证充分、论据可靠，恰当运用系统分析和比较研究的方法进行模型或方案设计，注重实证研究和案例分析，根据分析结果提出建议和改进措施等。论文主体是毕业论文的主要部分，必须言之成理，论据可靠，严格遵循本学科国际通行的学术规范。在写作上要注意结构合理、层次分明、重点突出，章节标题、公式图表符号必须规范统一。论文主体的内容根据不同学科有不同的特点，一般应包括以下几个方面：（1）毕业论文（设计）总体方案或选题的论证；（2）毕业论文（设计）各部分的设计实现，包括实验数据的获

取、数据可行性及有效性的处理与分析、各部分的设计计算等；（3）对研究内容及成果的客观阐述，包括理论依据、创新见解、创造性成果及其改进与实际应用价值等；（4）论文主体的所有数据必须真实可靠，凡引用他人观点、方案、资料、数据等，无论曾否发表，无论是纸质或电子版，均应详加注释。自然科学论文应推理正确、结论清晰；人文和社会学科的论文应把握论点正确、论证充分、论据可靠，恰当运用系统分析和比较研究的方法进行模型或方案设计，注重实证研究和案例分析，根据分析结果提出建议和改进措施等。

4.1.2 单张图像的插入

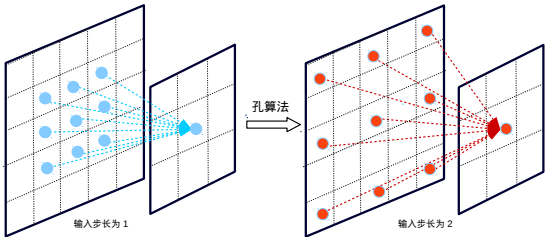


图 4.2 单张图像

4.1.3 多张图像的并排插入

图 4.3 并排的多张图像

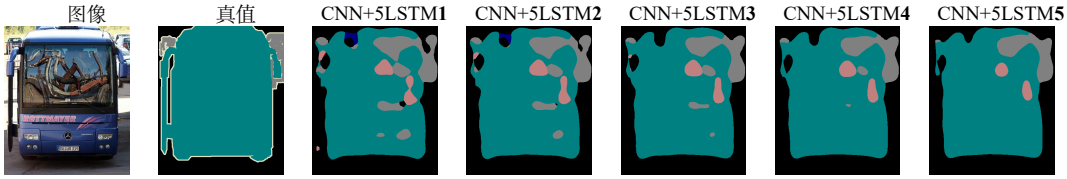
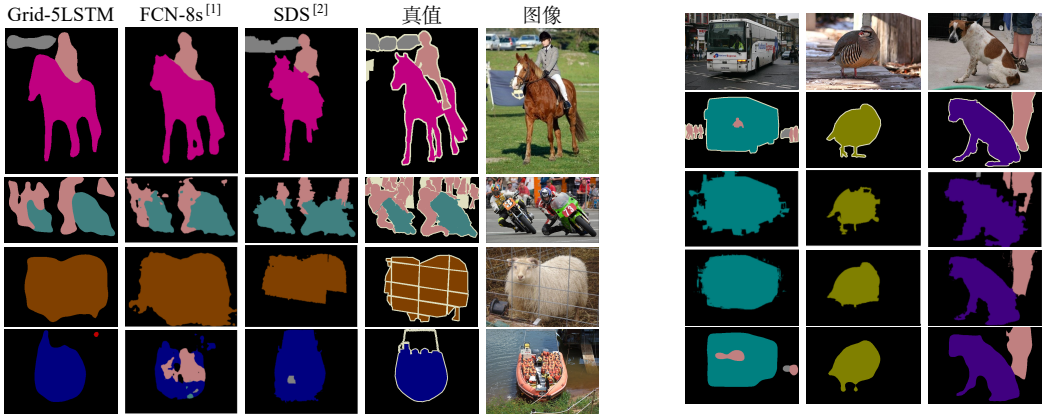


图 4.4 并排的多张图像加各自的注解

4.1.4 两列图像的插入



(a) 左边的图像

(b) 右边的图像

图 4.5 复杂的两列对象的插入

4.2 表格的插入

表 4.1 典型的实验对比表格

Method	Pixel Acc.	Mean Acc.	Mean Iu.
Liu 等人[3]	76.7	-	-
Tighe 等人[4]	78.6	39.2	-
FCN-16s[1]	85.2	**51.7**	39.5
Deeplab-LargeFOV[5]	85.6	51.2	39.7
Grid-LSTM5	**86.2**	51.0	**41.2**

表 4.2 复杂一些的表格

Method	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	shep	sofa	train	tv	mIoU.
CNN	72.6	29.6	70.2	53.1	65.1	81.0	74.3	79.8	25.0	64.8	47.8	69.5	66.2	65.2	74.2	42.1	69.6	38.8	74.4	58.6	62.5
CNN+**1**LSTM	71.5	30.6	70.5	53.8	64.9	82.4	77.1	79.5	25.1	65.8	47.8	71.5	64.6	67.0	74.0	43.9	69.6	38.6	74.9	59.4	63.0
CNN+**2**LSTM	76.1	32.6	72.1	57.0	65.3	83.6	75.4	81.7	24.7	69.3	47.5	72.3	68.9	69.5	74.7	41.5	69.8	38.3	77.8	62.1	64.3
CNN+**3**LSTM	77.7	32.3	72.6	60.0	68.3	85.5	78.5	82.3	25.3	71.1	49.7	71.5	69.7	70.8	75.9	47.9	71.2	38.9	80.2	61.7	65.8
CNN+**4**LSTM	79.1	**33.7**	**73.6**	**62.0**	**70.4**	85.5	**80.9**	83.7	**24.1**	70.7	45.7	73.7	69.6	72.1	75.6	47.2	**76.0**	37.3	80.5	62.2	66.4
CNN+**5**LSTM	**79.9**	33.6	**73.6**	61.7	68.0	**88.5**	**80.9**	**84.0**	23.6	**71.3**	**49.7**	**73.1**	**71.3**	**72.9**	**76.4**	**48.9**	75.1	**38.1**	**84.5**	**63.8**	**67.2**
CNN+**5**LSTM†	84.8	36.4	82.0	69.4	73.0	87.2	81.8	86.1	34.5	82.4	53.1	81.5	77.4	79.0	81.3	54.8	81.1	47.0	84.3	67.3	72.3

4.3 公式

没有编号的公式

$$\boldsymbol{z}^{(l)} = \boldsymbol{W}^{(l)}\boldsymbol{a}^{(l-1)} + \boldsymbol{b}^{(l)}$$
$$\boldsymbol{a}^{(l)} = f(\boldsymbol{z}^{(l)})$$

公式中含有中文

$$\begin{aligned} \text{像素准确率} &= \sum_{i=1}^{n_{cl}} n_{ii} / \sum_{i=1}^{n_{cl}} t_i \\ \text{平均像素准确率} &= \frac{1}{n_{cl}} \sum_{i=1}^{n_{cl}} (n_{ii}/t_i) \\ \text{Mean IU} &= \frac{1}{n_{cl}} \sum_{i=1}^{n_{cl}} \frac{n_{ii}}{t_i + \sum_j^{n_{cl}} n_{ji} - n_{ii}} \end{aligned} \tag{4.1}$$

公式中含有矩阵

$$\mathbf{H} = \begin{bmatrix} I * \boldsymbol{x}_i \\ \mathbf{h} \end{bmatrix} \tag{4.2}$$

每行后面都有编号的公式

$$\frac{\partial}{\partial W_{ij}^{(l)}} J(\boldsymbol{W}, \boldsymbol{b}; \boldsymbol{x}, y) = \frac{\partial J(\boldsymbol{W}, \boldsymbol{b}; \boldsymbol{x}, y)}{\partial z_i^{(l+1)}} \cdot \frac{\partial z_i^{(l+1)}}{\partial W_{ij}^{(l)}} = \delta_i^{(l+1)} a_j^{(l)} \tag{4.3}$$

$$\frac{\partial}{\partial b_i^{(l)}} J(\boldsymbol{W}, \boldsymbol{b}; \boldsymbol{x}, y) = \frac{\partial J(\boldsymbol{W}, \boldsymbol{b}; \boldsymbol{x}, y)}{\partial z_i^{(l+1)}} \cdot \frac{\partial z_i^{(l+1)}}{\partial b_i^{(l)}} = \delta_i^{(l+1)} \tag{4.4}$$

4.4 算法流程图

算法 4.1: 梯度下降算法

输入: m 个训练样本

对于 $l = 1$ **转到** n_l **进行** 初始化: $\Delta \boldsymbol{W}^{(l)} = 0,\ \Delta \boldsymbol{b}^{(l)} = 0$;

对于每个 训练样本 **进行**

  对于 $l = 1$ **转到** $n_l - 1$ **进行** 前向传播:

  $\boldsymbol{z}^{(l+1)} = \boldsymbol{W}^l a^l + \boldsymbol{b}^l$,$\boldsymbol{a}^{(l+1)} = f(\boldsymbol{z}^{(l+1)})$;

  输出误差计算: $\delta^{(n_l)} = \frac{\partial}{\partial \boldsymbol{z}^{(n_l)}} J(\boldsymbol{W}, \boldsymbol{b}; \boldsymbol{x}, y)$;

  对于 $l = n_l - 1$ **转到** 1 **进行** 后向传播: $\delta^{(l)} = \left((\boldsymbol{W}^{(l)})^T \delta^{(l+1)}\right) f'(\boldsymbol{z}^{(l)})$;

  对于所有 层 l **进行**

    计算梯度: $\nabla_{\boldsymbol{W}^{(l)}} J(\boldsymbol{W}, \boldsymbol{b}; \boldsymbol{x}, y) = \delta^{(l+1)} (\boldsymbol{a}^{(l)})^T$

    $\nabla_{\boldsymbol{b}^{(l)}} J(\boldsymbol{W}, \boldsymbol{b}; \boldsymbol{x}, y) = \delta^{(l+1)}$;

    累加梯度: $\Delta \boldsymbol{W}^{(l)} \leftarrow \Delta \boldsymbol{W}^{(l)} + \nabla_{\boldsymbol{W}^{(l)}} J(\boldsymbol{W}, \boldsymbol{b}; \boldsymbol{x}, y)$;

    $\Delta \boldsymbol{b}^{(l)} \leftarrow \Delta \boldsymbol{b}^{(l)} + \nabla_{\boldsymbol{b}^{(l)}} J(\boldsymbol{W}, \boldsymbol{b}; \boldsymbol{x}, y)$;

对于所有 层 l **进行**

  更新权重: $\boldsymbol{W}^{(l)} \leftarrow \boldsymbol{W}^{(l)} - \alpha \left[\frac{1}{m} \Delta \boldsymbol{W}^{(l)}\right]$

  $\boldsymbol{b}^{(l)} \leftarrow \boldsymbol{b}^{(l)} - \alpha \left[\frac{1}{m} \Delta \boldsymbol{b}^{(l)}\right]$

4.5 例子、定理与证明

例 4.1　这是一个例子, 用以验证特殊环境的字体成功更改为楷体.

定理 4.1 **(定理例子)**　这是一个定理。

推论 4.1 **(推论例子)**　这是一个推论。

引理 4.1 **(引理例子)**　这是一个引理。

这里我们先给出定理 4.2

定理 4.2 **(中山大学毕业论文模板定理)**　中山大学 LaTeX 毕业论文模板[6] 可以用于写各种证明。

下面我们对定理 4.2进行证明:

证明 下面我们开始证明：

由本定理的证明可见，我可以引用定理 4.1和引理4.1以及推论4.1来证明我这个 LaTeX 可以用来写各种证明。

定理 4.2得证。 □

4.6 代码

本模版支持在论文中插入代码片段，或直接从源码文件进行插入。例如，在论文中插入代码片段的效果为：

```
    def func():
    print("hello world")
    with open('./output.txt', 'w') as f:
    L = f.readlines()

    if __name__ == "__main__":
    # this is a comment line
    func()
```

也可在行内插入代码片段，例如：Python 中重载加法运算符的函数为`__add__`，类的标识符为`class`。此外，还可直接插入代码文件，例如插入`./code/demo.cpp`的效果为：

```
#include <iostream>
int main()
{
    ::std::cout << "hello world" << ::std::endl;
    return 0;
}
```

4.7 其他的一些用法

4.7.1 子章节编号

4.7.1.1 更小的章节

更小的章节编号也是支持的。

可以如此引用章节：

- 章 4
- 节 4.7

- 小节 4.7.1
- 小小节 4.7.1.1

4.7.2 列表的使用

这是一个无序列表

- 引用文献[1]
- 引用文献作者Long et al.
- 引用文献年份2015
- 字体变红，**粗体**，斜体，下划线。

这是一个有序列表

1) 索引前面的节 4.3、图像4.5、表格4.1
2) 加脚注①

① 测试一下脚注和 URL `http://cs231n.github.io/transfer-learning/`

5 其他注意事项

5.1 关于生僻字

测试生僻字

🄵🄵🄵🄵🄵🄵🄵🄵🄵🄵🄵🄵🄵🄵🄵🄵🄵🄵🄵🄵🄵🄵🄵🄵麈🄵🄵🄵🄵🄵🄵🄵🄵🄵🄵🄵🄵🄵🄵🄵

6 实验与结果

参考文献

[1] LONG J, SHELHAMER E, DARRELL T. Fully convolutional networks for semantic segmentation[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. [S.l.: s.n.], 2015: 3431-3440.

[2] HARIHARAN B, ARBELÁEZ P, GIRSHICK R, et al. Simultaneous detection and segmentation[M]//Computer vision–ECCV 2014. [S.l.]: Springer, 2014: 297-312.

[3] LIU C, YUEN J, TORRALBA A. Sift flow: Dense correspondence across scenes and its applications[J]. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 2011, 33(5): 978-994.

[4] TIGHE J, LAZEBNIK S. Finding things: Image parsing with regions and per-exemplar detectors[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. [S.l.: s.n.], 2013: 3001-3008.

[5] CHEN L C, PAPANDREOU G, KOKKINOS I, et al. Semantic image segmentation with deep convolutional nets and fully connected crfs[C/OL]//ICLR. 2015. http://arxiv.org/abs/1412.7062.

[6] CHEN S, HUANG J, YONGFENG W. 中山大学毕业论文 Latex 模板 [EB/OL]. 2020. https://github.com/SYSU-SCC/sysu-thesis.

附录 A 补充更多细节

A.1 补充图

A.1.1 补充图

这是附录内容，应该用宋体小四号字体。

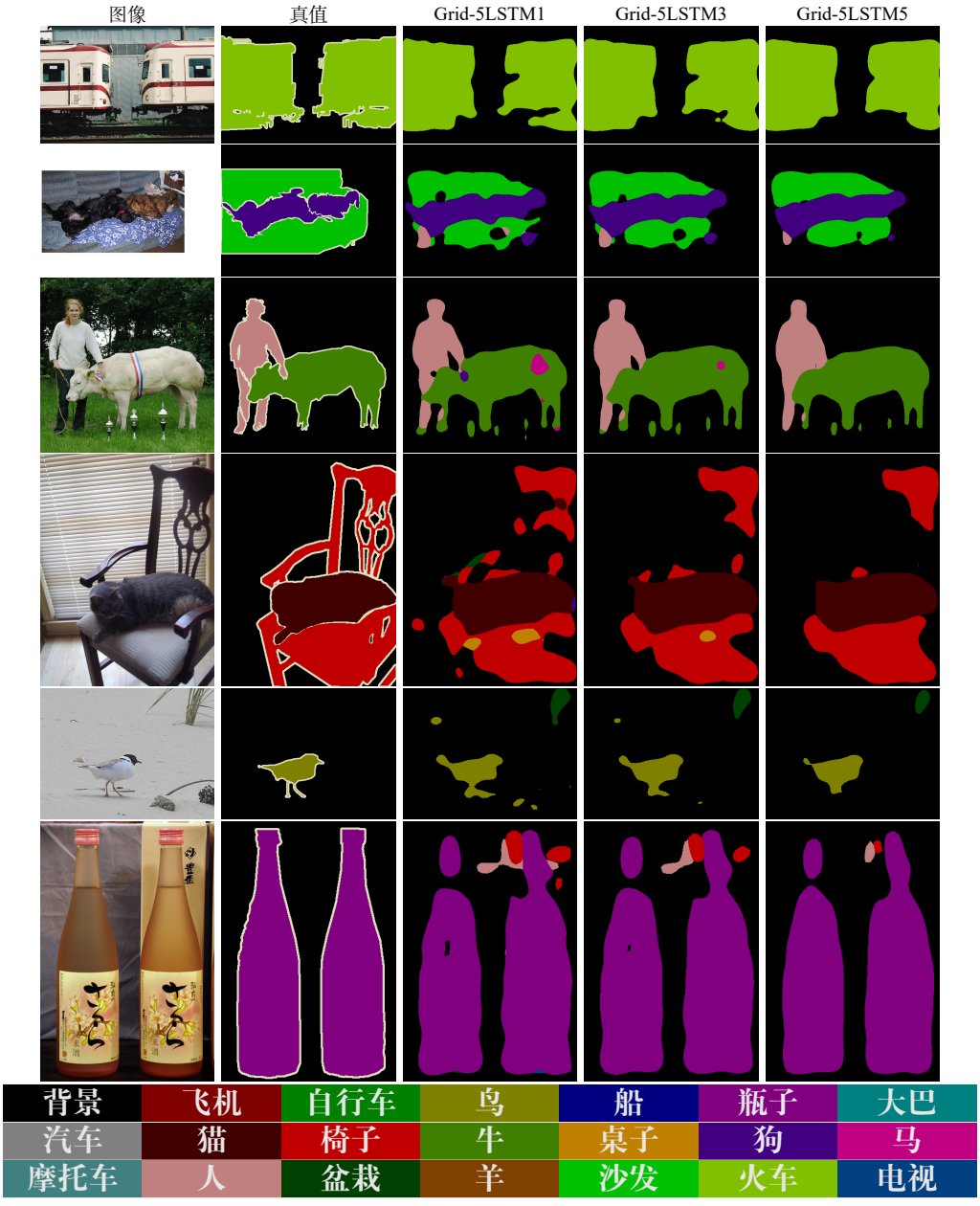


图 A.1 一个配有彩色表格的插图

致谢

四年时间转眼即逝，青涩而美好的本科生活快告一段落了。回首这段时间，我不仅学习到了很多知识和技能，而且提高了分析和解决问题的能力与养成了一定的科学素养。虽然走过了一些弯路，但更加坚定我后来选择学术研究的道路，实在是获益良多。这一切与老师的教诲和同学们的帮助是分不开的，在此对他们表达诚挚的谢意。

首先要感谢的是我的指导老师王大明教授。我作为一名本科生，缺少学术研究经验，不能很好地弄清所研究问题的重点、难点和热点，也很难分析自己的工作所能够达到的层次。王老师对整个研究领域有很好的理解，以其渊博的知识和敏锐的洞察力给了我非常有帮助的方向性指导。他严谨的治学态度与辛勤的工作方式也是我学习的榜样，在此向王老师致以崇高的敬意和衷心的感谢。

最后我要感谢我的家人，正是他们的无私的奉献和支持，我才有了不断拼搏的信心和勇气，才能取得现在的成果。

王小明

2022 年 3 月 27 日