

中国矿业大学计算机学院

21 级本科生课程报告

课程名称: 人工智能工具与平台实践

报告时间: 2023.6.18

学生姓名: 杨学通、黄圳

赵泊远、王珂

学 号: 08213129、12204201

08213122、12204302

专 业: 人工智能

任课教师: 寇旗旗

任课教师评语

任课教师评语(①对课程基础理论的掌握;②对课程知识应用能力的评价;③对课程报告相关实验、作品、软件等成果的评价;④课程学习态度和上课纪律;⑤课程成果和报告工作量;⑥总体评价和成绩;⑦存在问题等):

序号	毕业要求	课程教学目标	考查方式与考查点	占比	得分
1	1.3	目标 1: 掌握人工智能领域软件、硬件系统的搭建及应用原则;掌握硬件的基本功能模块。	考核课程设计题目相关的理论知识,课程报告中硬件基本模块及其功能、使用场景的介绍。	10%	
2	2.1	目标 2: 能够理解信息处理领域工程需求,能够分析系统功能的软件需求,并采用结构化方法描述系统需求。	考核题目需求分析和功能分析,课程报告中相关软件开发所具备的环境、知识及基本功能的介绍	10%	
3	3.2	目标 3: 能够正常的进行硬件的安装、构建虚拟机及搭建运行环境。	课程报告中硬件安装的介绍、软件运行环境搭建的流程	15%	
4	4.3	目标:4: 能够搭建开发环境,综合考虑系统的算法模型和软硬件开发环境,进行合理得方案设计、编程实现、系统测试及对设计方案进行优化。	课程报告中软件开发环境、硬件加载运行环境搭建的流程,模型的训练、编程实现的代码难度和复杂性、设计工作量等	35%	
5	10.3	目标 5: 报告撰写规范,通过成果演示、陈述发言、清晰表达等方式进行有效沟通与交流。	现场验收与答辩 考核成果演示的质量评价,演示成果所涉及的问题答辩。验收报告的结构合理性、内容和图表的正确性。课程验收报告排版的规范性。	30%	
总成绩				100%	

任课教师签字:

年 月 日

目 录

1 概论.....	1
1.1 选题背景及意义	1
1.2 选题内容及目标	2
1.3 小组成员任务分工及工作量明细介绍	2
2 系统总体方案介绍.....	2
2.1 系统的组成及工作原理	2
2.2 系统总体设计及框图	3
3 系统软硬件环境搭建.....	3
3.1 环境部署流程	3
3.2 搭建运行环境	4
3.3 搭建开发环境	8
4 系统软件程序设计.....	10
4.1 模型设计及训练优化	10
4.2 模型的转换与加载	11
5 系统测试.....	11
5.1 系统运行所需第三方依赖安装	11
5.2 系统运行界面截图	15
5.2 系统运行测试结果	16
6 系统设计结果分析及结论.....	17
6.1 系统设计结果	17
6.2 结果分析及结论	18
7 课程设计体会总结.....	18
8 程序源代码及注释.....	18
8.1 原始模型代码	18
8.2 对应的 CFG 代码	30

1 概论

1.1 选题背景及意义

在上个世纪 60 年代，美国 IBM 公司开始进行了对印刷体汉字的模式识别研究工作，1996 年 Casey 和 Nag 用模板匹配法成功的识别出了 1000 个印刷体汉字，在全球范围内，汉字识别开始展开了。而就在这个时候，研究界对手写汉字识别也掀起了高潮。因为汉字在日语中占有一定的地位，手写体汉字识别（HCCR）在一开始是由日本率先尝试研究的，在 80 年代，国内开始了对手写汉字的研究，因为汉语作为我们的母语，汉字主要在我国广泛使用，对汉字的种类、内涵、造字原理国内的掌握情况较透彻，所以关于手写汉字识别的深入研究主要集中在国内。

手写体汉字识别由于数据采集方式不同可以划分为脱机手写体汉字识别和联机手写体汉字识别两大类。联机手写汉字识别所处理的手写文字是书写者通过物理设备(如数字笔、数字手写板或者触摸屏) 在线书写获取的文字信号，书写的轨迹通过定时采样即时输入到计算机中。而脱机手写文字识别所处理的手写文字是通过扫描仪或摄像头等图像捕捉设备采集到的手写文字二维图片。由于识别的对象不同，使得这两类手写识别技术所采用的方法和策略也不尽相同。前者的识别对象是一系列的按时间先后排列的采样点信息，而后者则是丢失了书写笔顺信息的二维像素信息，由于没有笔顺信息，加之由于拍照扫描设备在不同光照、分辨率、书写纸张等条件下，数字化会带来一定的噪声干扰，一般来说，脱机手写文字识别比联机手写文字识别更加困难。

手写汉字识别是一个极具挑战性的模式识别及机器学习问题，主要表现在：

- 1) 书写方式随意，不规正，无法达到印刷体要求；
- 2) 汉字字符级别比较繁杂，极具变化特点；
- 3) 诸多汉字在外形上相似，容易混淆；
- 4) 要求具备庞大的训练数据，但采集困难，特别是随意性、无约束性手写，对应数据库的构建显得力不从心。

可见，手写汉字识别进步空间较大，需要综合各项技术，增加训练样本数据，提升识别率。

一般而言，传统的手写中文单字识别系统主要包括数据预处理、特征提取和分类识别三部分。然而，近些年来，传统的手写汉字识别框架进展并不明显，原地踏步，急需寻找其它的解

决方案。而深度学习正满足了手写汉字识别革新需求。实践证明，在深度学习技术协助下，联机 HCCR、脱机 HCCR 的识别率都大为提升，同原有的识别技术相比进步非常明显。

1.2 选题内容及目标

文字识别在当今大学校园，乃至整个社会层面都有很强的应用性。尤其对于长与文字打交道的人来说，文字识别是其日常办公所必须的。以往的长篇幅文字录入需要大量的手工劳作，不仅需要大量人力输出，而且耗费时间。加入文字识别功能后，让文字录入工作简单化，高效率，低成本。脱机手写体汉字识别，由于存在手写的随意性，相较于识别传统机器文字，复杂性更高，识别难度更大，如果能够充分实现该功能，并保证准确率，将会产生巨大的社会经济效益。

本次实验内容为在个人 PC 上配置 Ubuntu 18.04 虚拟机，部署开发环境，在昇腾 Atlas 200DK 开发板上部署运行环境。将训练好的汉字手写体识别模型加载到运行环境，实现脱机汉字手写体识别。

1.3 小组成员任务分工及工作量明细介绍

小组成员		小组分工	
组长	杨学通	前期	搭建开发环境、搭建运行环境（采用开发环境和运行环境分设）、运行简单样例（检测目标汽车）测试开发和运行环境是否搭建正确；
		中期	由于前期错选安装用户，不能正确配置环境变量，无法安装运行样例所需依赖，故重新制卡、重新搭建环境（开发环境和运行环境合设）；
		后期	<ul style="list-style-type: none"> 选择运行样例（汉字手写体识别），配置运行样例所需依赖（如，opencv、ffmpeg 等），成功运行样例； 安装MobaXterm远程登陆软件，实现样例测试结果可视化； 撰写实验报告。
组员	赵泊远	负责运行代码，测试代码在环境中的运行情况，并加以调适，使得代码适应环境，得出实验最终结果，判断是否与预期一致	
	黄圳	开发环境构建	
	王珂	搭建运行环境	

成员 1 签字：

成员 3 签字：

成员 2 签字：

成员 4 签字：

2 系统总体方案介绍

2.1 系统的组成及工作原理

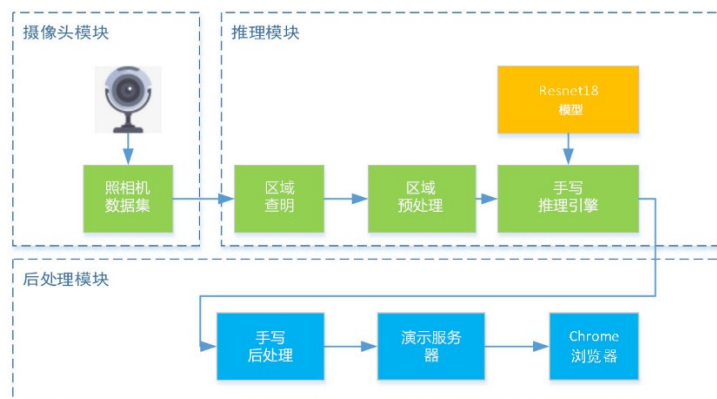
本系统主要分为三大模块：摄像头模块、推理模块（Atlas 200DK 开发板）、后处理模块（PC 端浏览器）。

摄像头模块：主要负责对前方进行连续地拍照，形成不间断地图片流，并传入推理模块；

推理模块：对摄像头传来的图片流，划分手写体文字所在区域，对划分区域进行预处理后送入手写体推理引擎，利用 Resnet18 模型推理识别手写体内容；

后处理模块：主要负责推理结果的输出。

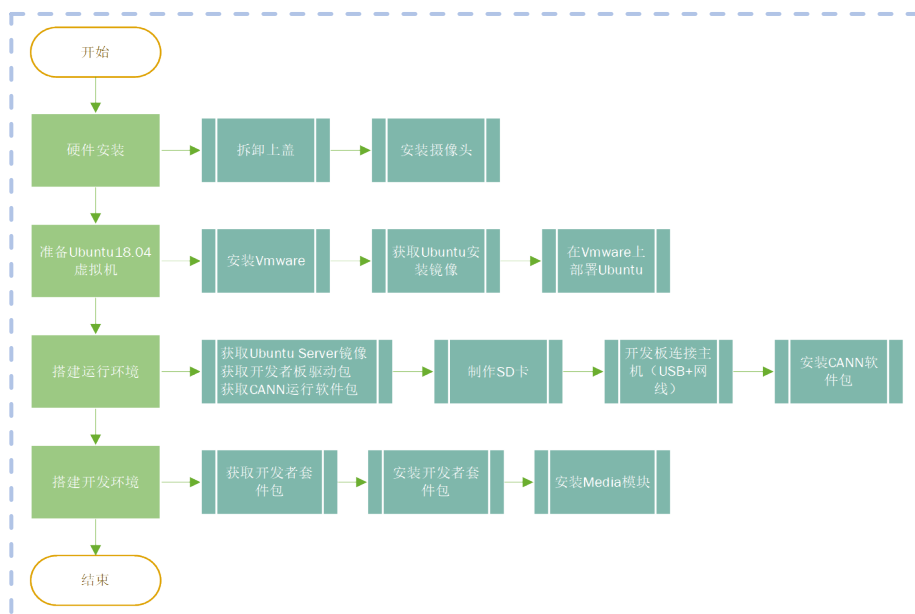
2.2 系统总体设计及框图



3 系统软硬件环境搭建

3.1 环境部署流程

运行环境和开发环境分设



3.2 搭建运行环境

• 获取安装镜像或安装包

获取 Ubuntu-18.04.6-desktop-amd64.iso

A full list of available files, including BitTorrent files, can be found below.
If you need help burning these images to disk, see the [Image Burning Guide](#).

Name	Last modified	Size	Description
Parent Directory		-	
SHA256SUMS	2021-09-16 21:58	202	
SHA256SUMS.gpg	2021-09-16 21:58	833	
ubuntu-18.04.6-desktop-amd64.iso	2021-09-15 20:42	2.3G	Desktop image for 64-bit PC (AMD64) computers (standard download)
ubuntu-18.04.6-desktop-amd64.iso.torrent	2021-09-16 21:46	188K	Desktop image for 64-bit PC (AMD64) computers (BitTorrent download)
ubuntu-18.04.6-desktop-amd64.iso.zsync	2021-09-16 21:46	4.7M	Desktop image for 64-bit PC (AMD64) computers (zsync metafile)
ubuntu-18.04.6-desktop-amd64.list	2021-09-15 20:42	7.8K	Desktop image for 64-bit PC (AMD64) computers (file listing)
ubuntu-18.04.6-desktop-amd64.manifest	2021-09-15 20:36	59K	Desktop image for 64-bit PC (AMD64) computers (contents of live filesystem)

获取开发者板驱动包

产品系列
开发者套件

产品型号
Atlas 200 DK 开发者套件 (型号: 3000)

CANN版本
6.0.0.alpha003

固件与驱动
1.0.13.alpha

一键下载

软件名称	分类	发布日期	说明	大小	操作
A200dk-npu-driver-21.0.4-ubuntu18.04-aarch64-minirc.tar.gz	NPU	2022/04/01		47.1MB	下载

获取 CANN 运行软件包

6.0.0.alpha003
2022/12/12

本版本进行了特性增强，针对PyTorch的部分动态shape网络，若使用了编译好的算子二进制包，会有性能收益。

CPU架构 ☐ 全部 ☒ AArch64 ☐ X86_64

软件包格式 ☐ 全部 ☒ run ☐ targz

一键下载

数字签名验证工具下载

软件名称	说明	操作
Ascend-cann-nnae_6.0.0.alpha003_linux-aarch64.run	ARM平台深度学习引擎软件包，适用于命令行方式安装场景	软件包下载 数字签名下载
Ascend-cann-nnrt_6.0.0.alpha003_linux-aarch64.run	ARM平台推理引擎软件包，适用于命令行方式安装场景	软件包下载 数字签名下载
Ascend-cann-toolkit_6.0.0.alpha003_linux-aarch64.run	ARM平台开发套件软件包，适用于命令行方式安装场景	软件包下载 数字签名下载

• 制作 SD 卡

步骤 1 请将 SD 卡放入读卡器，并将读卡器与 Ubuntu 服务器的 USB 接口连接。

步骤 2 在 Ubuntu 服务器中执行如下命令安装 qemu-user-static、binfmt-support、yam

1、squashfs-tools 与交叉编译器。

```
su root
apt-get update
pip3 install pyyaml
apt-get install qemu-user-static binfmt-support python3-yaml squashfs-tools gcc
-aarch64-linux-gnu g++-aarch64-linux-gnu
```

步骤 3 在 Ubuntu 服务器中以 root 用户执行如下命令创建制卡工作目录。

```
mkdir $HOME/mksd
```

步骤 4 将软件包准备获取的 Ubuntu 操作系统镜像包、开发者板驱动包、CANN 运行软件包上传到制卡工作目录中

步骤 5 在制卡工作目录下执行如下命令获取制卡脚本。

下载制卡入口脚本 “make_sd_card.py”。

```
wget https://gitee.com/ascend/tools/raw/master/makesd/for_1.0.10.alpha/make_sd_card.py
```

下载制作 SD 卡操作系统的脚本 “make_ubuntu_sd.sh”。

```
wget https://gitee.com/ascend/tools/raw/master/makesd/for_1.0.10.alpha/make_ubuntu_sd.sh
```

步骤 6 执行制卡脚本。

以 root 用户执行如下命令查找 SD 卡所在的 USB 设备名称。

```
fdisk -l
```

运行 SD 制卡脚本 “make_sd_card.py”。

```
python3 make_sd_card.py local /dev/sda
```

步骤 7 制卡成功后，将 SD 卡从读卡器取出并插入 Atlas 200 DK 开发者板卡槽。

步骤 8 上电 Atlas 200 DK 开发者板。

• 连接开发板和主机

USB 连接开发板和主机（用于主机和开发板之间互传文件）

步骤一 在主机上安装 RNDIS 虚拟网卡

步骤二 配置虚拟网卡的 IP 地址为 192.168.1.110

步骤三 在命令行中执行如下命令，连接开发板

```
ssh HwHiAiUser@192.168.1.2
```



```
? MobaXterm Personal Edition v23.1 ?
(SSh client, X server and network tools)

► SSH session to HwHiAiUser@192.168.1.2
? Direct SSH      : ✓
? SSH compression : ✓
? SSH-browser     : ✓
? X11-forwarding  : ✓ (remote display is forwarded through SSH)

► For more info, ctrl+click on help or visit our website.

Welcome to Ubuntu 18.04.6 LTS (GNU/Linux 4.19.90+ aarch64)

* Documentation:  https://help.ubuntu.com
* Management:    https://landscape.canonical.com
* Support:        https://ubuntu.com/advantage
Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

Last login: Sun Jun 18 11:08:47 2023 from 192.168.1.110
HwHiAiUser@davinci-mini:~$
```

网线连接开发板和主机（用于开发板通过共享主机的网络的方式，接入互联网）

步骤一 进入开发板 root 用户

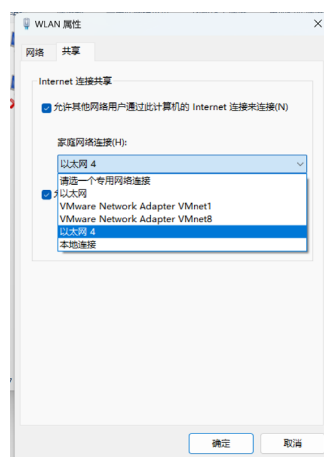
步骤二 执行以下命令，配置 eth0 的网关地址 192.168.0.101

vi /etc/netplan/01-netcfg.yaml

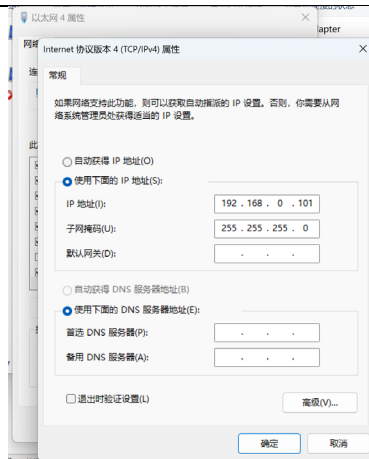
```
network:
  version: 2
  renderer: networkd
  ethernets:
    eth0:
      dhcp4: no
      addresses: [192.168.0.2/24]
      gateway4: 192.168.0.101
      nameservers:
        addresses: [8.8.8.8]
        addresses: [114.114.114.114]

    usb0:
      dhcp4: no
      addresses: [192.168.1.2/24]
      # gateway4: 192.168.1.101
      nameservers:
        addresses: [8.8.8.8]
        addresses: [114.114.114.114]
```

步骤三 在 Windows 中共享网络给开发板（以太网 4）



步骤四 配置以太网 4 的 IP 地址为 192.168.0.101



```
Last login: Sun Jun 18 11:10:07 2023 from 192.168.1.110
HwHiAiUser@davinci-mini:~$ ping www.baidu.com
PING www.baidu.com (36.152.44.96) 56(84) bytes of data:
64 bytes from 36.152.44.96 (36.152.44.96): icmp_seq=1 ttl=55 time=67.4 ms
64 bytes from 36.152.44.96 (36.152.44.96): icmp_seq=2 ttl=55 time=53.9 ms
64 bytes from 36.152.44.96 (36.152.44.96): icmp_seq=3 ttl=55 time=50.2 ms
64 bytes from 36.152.44.96 (36.152.44.96): icmp_seq=4 ttl=55 time=51.8 ms
64 bytes from 36.152.44.96 (36.152.44.96): icmp_seq=5 ttl=55 time=50.2 ms
```

• 安装软件包

步骤一 执行如下命令为安装包增加可执行权限。

```
chmod +x *.run
```

执行如下校验安装包的一致性和完整性。

```
./Ascend-cann-nnrt_{software version}_linux-aarch64.run --check
```

执行如下命令进行离线推理引擎包的安装。

```
./Ascend-cann-nnrt_{software version}_linux-aarch64.run --install
```

步骤二 配置环境变量。

```
./usr/local/Ascend/nnrt/set_env.sh
```

```
./etc/bash_completion
fi
fi
export LD_LIBRARY_PATH=/usr/lib64/aicpu_kernels/0/aicpu_kernels_device:/usr/lib64/aicpu_kernels/0/aicpu_kernels_device/sand_box

#用于设置python3.9.7库文件路径
export LD_LIBRARY_PATH=/usr/local/python3.9.7/lib:$LD_LIBRARY_PATH
#如果用户环境存在多个python3版本，则指定使用python3.9.7版本
export PATH=/usr/local/python3.9.7/bin:$PATH

export CPU_ARCH='arch'
export THIRDPARTY_PATH=/usr/local/Ascend/thirdparty/${CPU_ARCH} #代码编译时链接第三方库
export PYTHONPATH=${THIRDPARTY_PATH}/acllite:$PYTHONPATH #设置pythonpath为固定目录
export INSTALL_DIR=/usr/local/Ascend/ascend-toolkit/latest #CANN软件安装后文件存储路径

export PATH=/home/HwHiAiUser/.local/bin:$PATH

export LD_PRELOAD=/usr/local/python3.9.7/lib/python3.9/site-packages/torch/lib/libgomp-d22c30c5.so.1:$LD_PRELOAD

./usr/local/Ascend/ascend-toolkit/set_env.sh
```

3.3 搭建开发环境

• 获取安装包



• 安装 CANN 软件

步骤一 安装 OS 依赖。

安装 gcc，make 以及 python 依赖软件

```
apt-get install -y gcc g++ make cmake zlib1g zlib1g-dev openssl libsqlite3-dev
libssl-dev libffi-dev unzip pciutils net-tools libblas-dev gfortran libblas3
```

步骤二 安装 python3.7.5。

```
wget https://www.python.org/ftp/python/3.7.5/Python-3.7.5.tgz
tar -zxvf Python-3.7.5.tgz
cd Python-3.7.5
./configure --prefix=/usr/local/python3.7.5 --enable-loadable-sqlite-extensions --
enable-shared
make
sudo make install
```

步骤三 设置 python3.7.5 环境变量。

```
#用于设置 python3.7.5 库文件路径
export LD_LIBRARY_PATH=/usr/local/python3.7.5/lib:$LD_LIBRARY_PATH
#如果用户环境存在多个 python3 版本，则指定使用 python3.7.5 版本
export PATH=/usr/local/python3.7.5/bin:$PATH
```

步骤四 安装依赖

```
pip3 install --upgrade pip
pip3 install attrs
```

```
pip3 install numpy
pip3 install decorator
pip3 install sympy
pip3 install cffi
pip3 install pyyaml
pip3 install pathlib2
pip3 install psutil
pip3 install protobuf
pip3 install scipy
pip3 install requests
pip3 install absl-py
```

步骤二 安装开发套件包（以 root 用户安装）。

进入套件包所在路径，执行如下命令为安装包增加可执行权限。

```
chmod +x *.run
```

执行如下校验安装包的一致性和完整性。

```
./Ascend-cann-toolkit_{version}_linux-x86_64.run --check
./Ascend-cann-toolkit_{version}_linux-aarch64.run --check
```

执行以下命令安装软件。

```
./Ascend-cann-toolkit_{version}_linux-x86_64.run --install
./Ascend-cann-toolkit_{version}_linux-aarch64.run --install
```

步骤三 配置交叉编译环境。

分设场景下，开发环境架构为“x86_64”，运行环境架构为“aarch64”，所以在开发环境中安装交叉编译器进行应用程序的交叉编译。

执行以下命令安装交叉编译器

```
apt-get install g++-aarch64-linux-gnu
```

步骤四 配置环境变量。

```
. /home/HwHiAiUser/Ascend/ascend-toolkit/set_env.sh
```

• 安装 Media 模块

步骤一 环境变量设置示例如下。

```
export INSTALL_DIR=${HOME}/Ascend/ascend-toolkit/latest
```

步骤二 在\${INSTALL_DIR}路径下创建“driver”目录

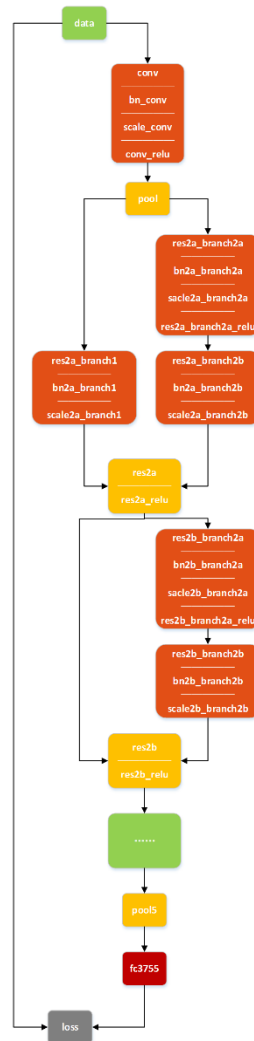
```
mkdir -p ${INSTALL_DIR}/driver
```

步骤三 将媒体应用程序编译依赖的相关文件从板端环境中拷贝到开发环境的 $\{\text{INSTALL_DIR}\}/\text{driver}$ 目录下。

```
sudo scp -r HwHiAiUser@X.X.X.X:/usr/lib64/libmedia_mini.so ${INSTALL_DIR}/driver/
sudo scp -r HwHiAiUser@X.X.X.X:/usr/lib64/libsllog.so ${INSTALL_DIR}/driver/
sudo scp -r HwHiAiUser@X.X.X.X:/usr/lib64/libc_sec.so ${INSTALL_DIR}/driver/
sudo scp -r HwHiAiUser@X.X.X.X:/usr/lib64/libmmpa.so ${INSTALL_DIR}/driver/
sudo scp -r HwHiAiUser@X.X.X.X:/usr/local/Ascend/include/peripheral_api.h ${INSTALL_DIR}/driver/
```

4 系统软件程序设计

4.1 模型设计及训练优化



4.2 模型的转换与加载

• 部署 ATC 工具

步骤一 设置公共环境变量

以 root 用户安装 Ascend-cann-toolkit 包

```
. /usr/local/Ascend/ascend-toolkit/set_env.sh
```

添加如下环境变量

```
export LD_LIBRARY_PATH=/usr/local/Ascend/ascend-toolkit/latest/<arch>-linux/
devlib:$LD_LIBRARY_PATH
```

步骤二 设置 Python 相关环境变量

#如果用户环境存在多个 python3 版本，则指定使用 python3.7.5 版本

```
export PATH=/usr/local/python3.7.5/bin:$PATH
```

#设置 python3.7.5 库文件路径

```
export LD_LIBRARY_PATH=/usr/local/python3.7.5/lib:$LD_LIBRARY_PATH
```

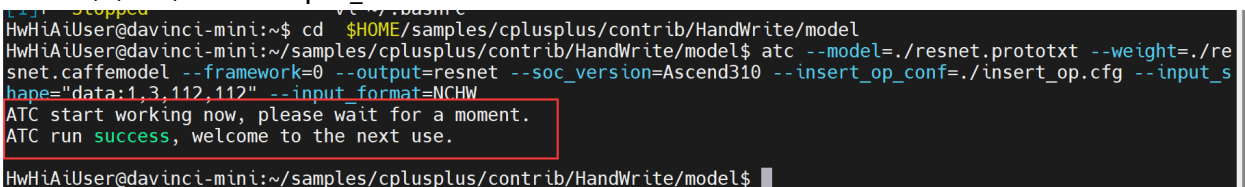
步骤三 设置环境变量，指定是否开启单线程或多线程编译。

```
export MULTI_THREAD_COMPILE=0
```

• 模型转换

在样例所在文件夹的/model 路径下执行以下命令进行模型转换

```
atc --model=./resnet.prototxt --weight=./resnet.caffemodel --framework=0 --output=
resnet --soc_version=Ascend310 --insert_op_conf=./insert_op.cfg --input_shape="dat
a:1,3,112,112" --input_format=NCHW
```



```
HwHiAiUser@davinci-mini:~$ cd $HOME/samples/cplusplus/contrib/HandWrite/model
HwHiAiUser@davinci-mini:~/samples/cplusplus/contrib/HandWrite/model$ atc --model=./resnet.prototxt --weight=./re
snet.caffemodel --framework=0 --output=resnet --soc_version=Ascend310 --insert_op_conf=./insert_op.cfg --input_s
hape="data:1,3,112,112" --input_format=NCHW
ATC start working now, please wait for a moment.
ATC run success, welcome to the next use.
HwHiAiUser@davinci-mini:~/samples/cplusplus/contrib/HandWrite/model$
```

5 系统测试

5.1 系统运行所需第三方依赖安装

系统运行所需依赖有 presentagent, ffmpeg+acllite

• 配置所需环境变量

开发环境

步骤一 以安装用户在开发环境任意目录下执行以下命令，打开.bashrc 文件。

```
vi ~/.bashrc
```

步骤二 在文件最后一行后面添加如下内容。

```
export CPU_ARCH=aarch64
```

```
export THIRDPART_PATH=/usr/local/Ascend/thirdpart/${CPU_ARCH} #代码编译时  
链接第三方库
```

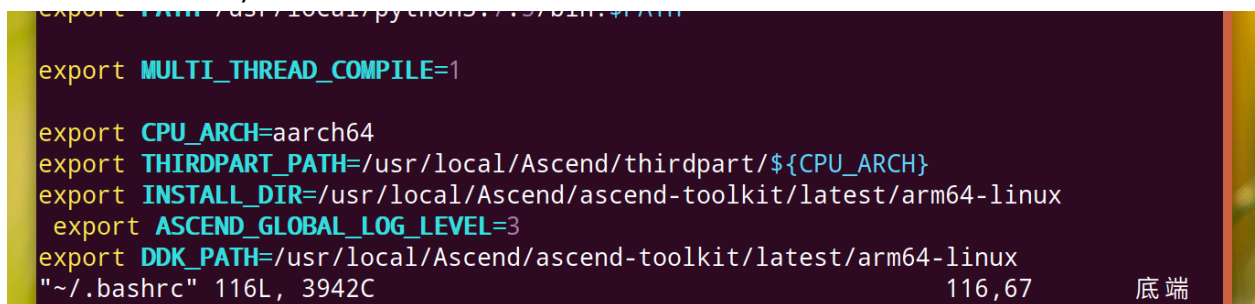
```
export INSTALL_DIR=/usr/local/Ascend/ascend-toolkit/latest/arm64-linux
```

步骤三 执行命令保存文件并退出。

```
:wq!
```

执行命令使其立即生效。

```
source ~/.bashrc
```

A terminal window with a dark background and light-colored text. It shows the following commands being entered and executed: export CPU_ARCH=aarch64, export THIRDPART_PATH=/usr/local/Ascend/thirdpart/\${CPU_ARCH}, export INSTALL_DIR=/usr/local/Ascend/ascend-toolkit/latest/arm64-linux, export ASCEND_GLOBAL_LOG_LEVEL=3, and export DDK_PATH=/usr/local/Ascend/ascend-toolkit/latest/arm64-linux. The prompt shows the file ~/.bashrc is 116 lines long and 3942 characters. The cursor is at line 116, column 67. The bottom right corner shows the text '底端'.

运行环境

步骤一 以安装用户在运行环境任意目录下执行以下命令，打开.bashrc 文件。

```
vi ~/.bashrc
```

步骤二 在文件最后一行后面添加如下内容。

```
export CPU_ARCH=`arch`
```

```
export THIRDPART_PATH=${HOME}/Ascend/thirdpart/${CPU_ARCH} #代码编译时  
链接第三方库
```

```
export LD_LIBRARY_PATH=${HOME}/Ascend/thirdpart/${CPU_ARCH}/lib:$LD_LIBRA  
RY_PATH #运行时链接库文件
```

```
export INSTALL_DIR=${HOME}/Ascend/ascend-toolkit/latest #CANN 软件安装后文  
件存储路径
```

步骤三 执行命令保存文件并退出。

```
:wq!
```

执行命令使其立即生效。

```
source ~/.bashrc
```

```
export CPU_ARCH=`arch`
export THIRDPART_PATH=/usr/local/Ascend/thirdpart/${CPU_ARCH} #代码编译时链接第三方库
export PYTHONPATH=${THIRDPART_PATH}/acllite:$PYTHONPATH #设置pythonpath为固定目录
export INSTALL_DIR=/usr/local/Ascend/ascend-toolkit/latest #CANN软件安装后文件存储路径
```

• 安装 presentagent

步骤一 安装 protobuf 相关依赖

```
# 安装 protobuf 相关依赖
sudo apt-get install autoconf automake libtool
# 安装 pip3
sudo apt-get install python3-pip
# 安装 presentserver 启动所需要的 python 库。若安装失败，请自行更换 python 源。
python3.6 -m pip install --upgrade pip --user
python3.6 -m pip install tornado==5.1.0 protobuf Cython numpy
python3.7 -m pip install tornado==5.1.0 protobuf Cython numpy
```

步骤二 安装 protobuf

```
运行环境为 arm。执行以下命令安装 protobuf
# 下载 protobuf 源码
cd ${HOME}
git clone -b 3.13.x https://gitee.com/mirrors/protobufsource.git protobuf
cp -r protobuf protobuf_arm
# 首次编译安装 protobuf，生成 x86 架构的 protoc 文件
cd protobuf
./autogen.sh
./configure
make -j8
sudo make install
cd $HOME/protobuf_arm
./autogen.sh
./configure --build=x86_64-linux-gnu --host=aarch64-linux-gnu --with-protoc=pr
otoc --prefix=${THIRDPART_PATH}
make -j8
make install
```

步骤三 生成 proto 文件并安装 presentagent。

```
cd $HOME/samples/cplusplus/common/presenteragent/proto
sudo ldconfig
```



```
protoc presenter_message.proto --cpp_out=./
# 安装 presenteragent
cd ..
make -j8
make install
# 拷贝相关 so, 其中 X.X.X.X 为运行环境 ip 地址。
sudo scp -r ${THIRDPART_PATH}/* HwHiAiUser@192.168.1.2:${THIRDPART_PATH}
```

- 安装 ffmpeg+acllite

步骤一 下载并安装 x264。

```
# 下载 x264
cd ${HOME}
git clone https://code.videolan.org/videolan/x264.git
cd x264
# 安装 x264
./configure --enable-shared --disable-asm
make
sudo make install
sudo cp /usr/local/lib/libx264.so.164 /lib
```

步骤二 下载并安装 ffmpeg。

```
# 下载 ffmpeg
cd ${HOME}
wget http://www.ffmpeg.org/releases/ffmpeg-4.1.3.tar.gz --no-check-certificate
tar -zxvf ffmpeg-4.1.3.tar.gz
cd ffmpeg-4.1.3
```

步骤三 安装 ffmpeg

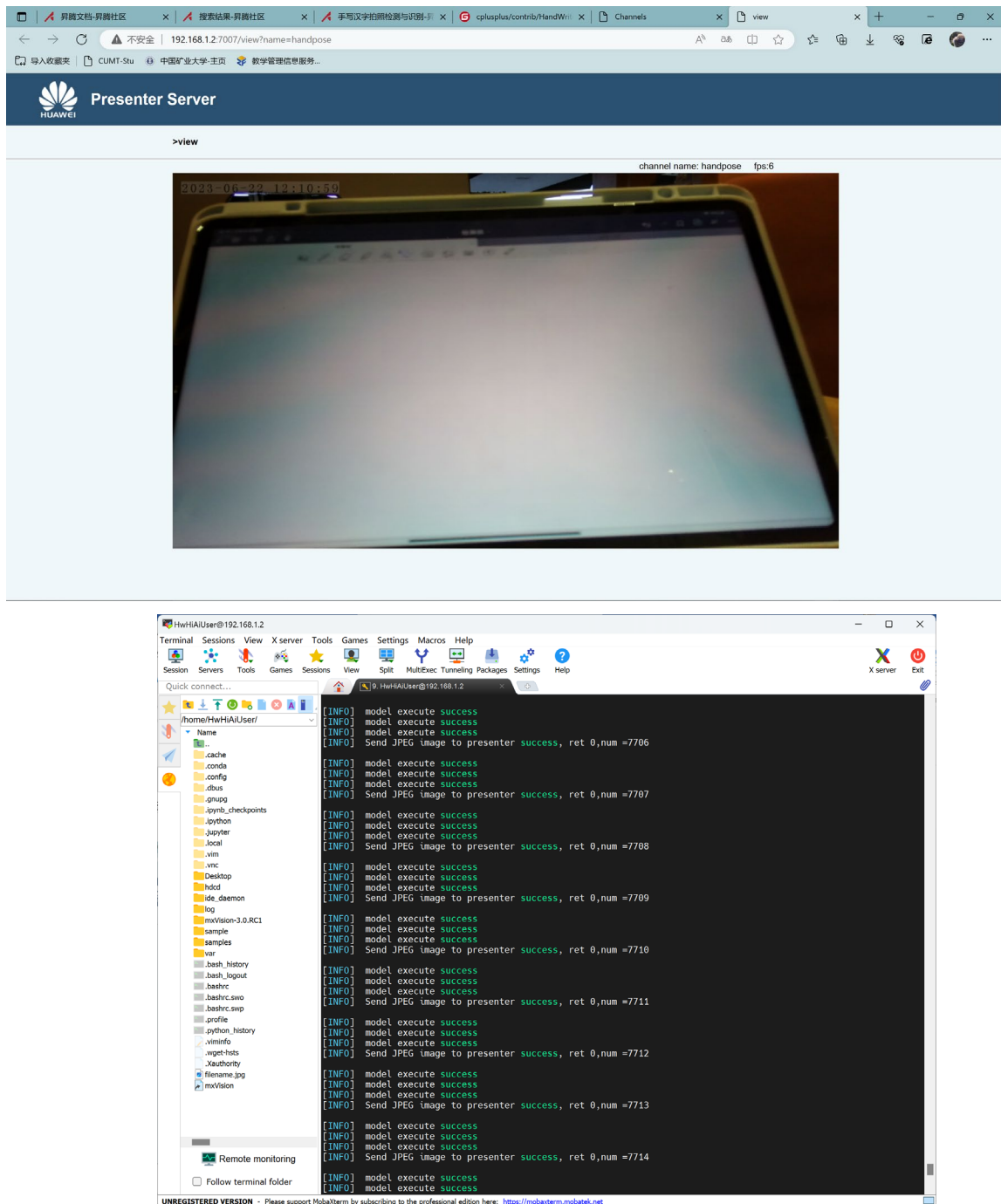
```
运行环境为 arm。执行以下命令安装 ffmpeg
./configure --enable-shared --enable-pic --enable-static --disable-x86asm --cross
-prefix=aarch64-linux-gnu- --enable-cross-compile --arch=aarch64 --target-os=li
nux --enable-libx264 --enable-gpl --prefix=${THIRDPART_PATH}
make -j8
make install
```

步骤四 安装 acllite 并将结果文件拷贝到运行环境。

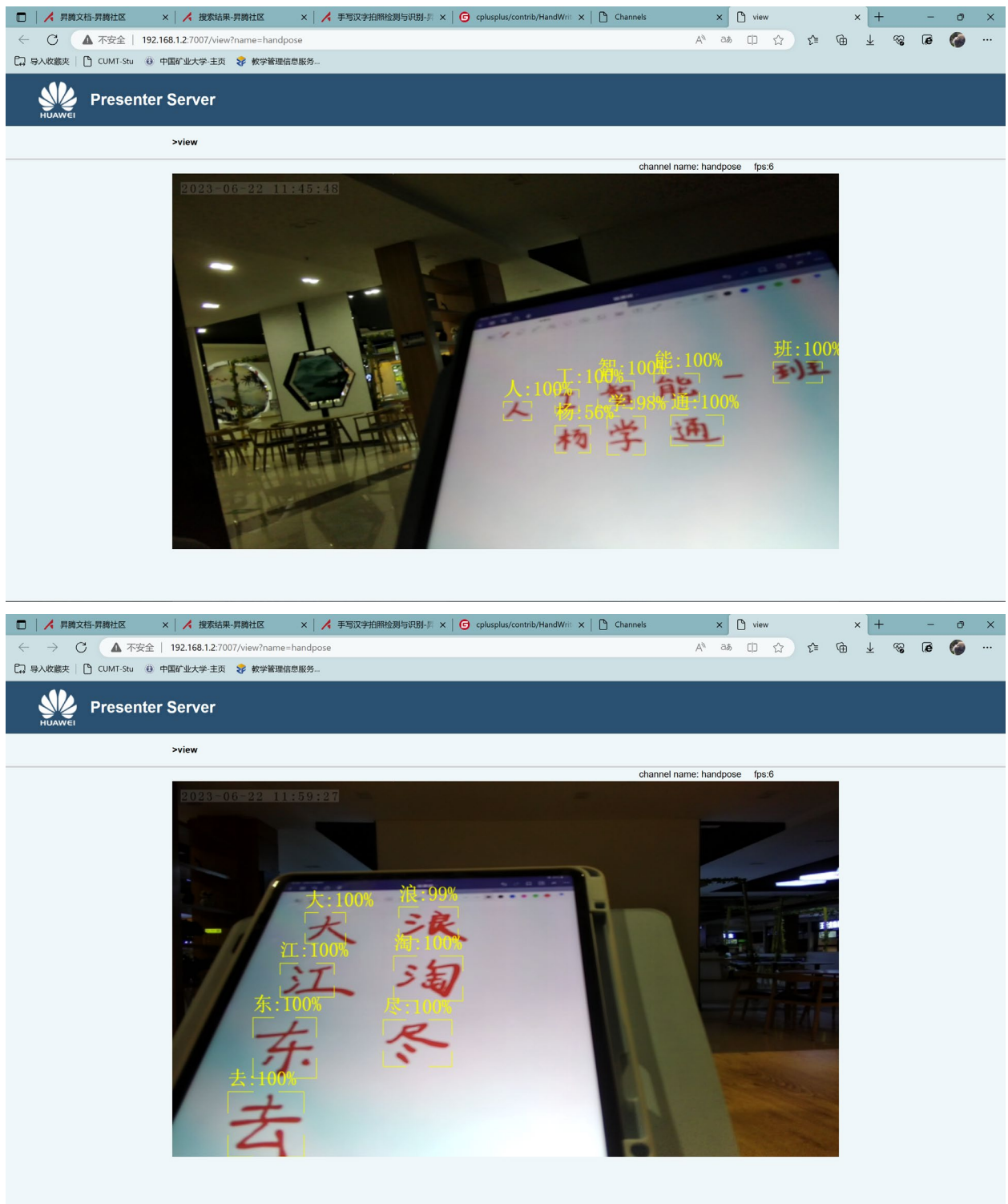
```
# 下载源码并安装 git
cd ${HOME}
sudo apt-get install git
git clone https://gitee.com/ascend/samples.git
```

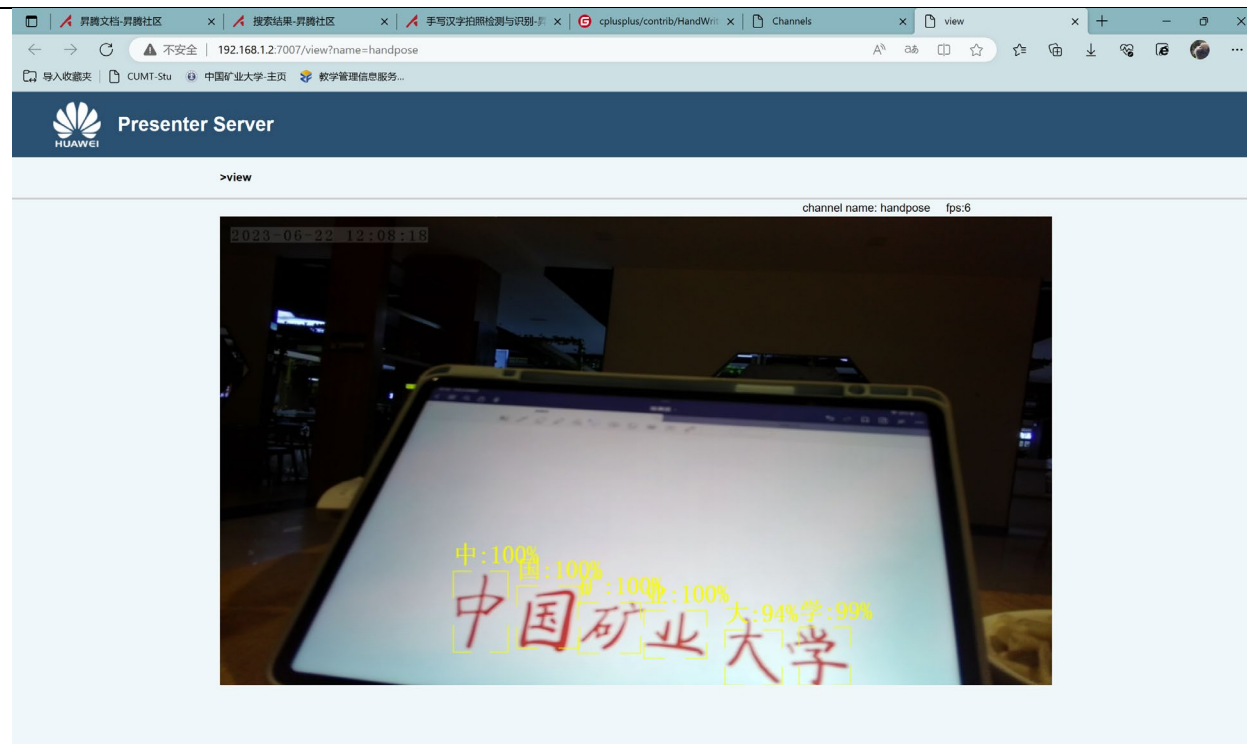
```
# 编译并安装 acllite
cd ${HOME}/samples/cplusplus/common/acllite/
make
make install
# 拷贝相关 so, 其中 X.X.X.X 为运行环境 ip 地址。
sudo scp -r ${THIRDPART_PATH}/* HwHiAiUser@X.X.X.X:${THIRDPART_PATH}
```

5.2 系统运行界面截图



5.2 系统运行测试结果

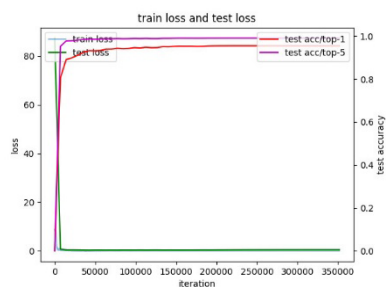




6 系统设计结果分析及结论

6.1 系统设计结果

实现了基于 ResNet 网络的汉字手写体识别模型，使用的数据集为 HITHCD-2018 数据集的子数据集。HITHCD-2018 是哈尔滨工业大学收集的、用于手写汉字识别（HCCR）的大型数据库，有超过 5346 书写者书写，是目前规模最大、字类最多的数据库。



上图为 ResNet 在 3755 类手写汉字训练集下的 accuracy 和 loss，我们发现模型具有很高的 accuracy 和较小的 loss。根据实际模型在 Atlas 200DK 上的运行结果来看，该样例能够准确的识别生活中的大多数手写体汉字识别率很高，对于个别的生僻汉字，识别效果不佳。

6.2 结果分析及结论

由于该样例是利用树莓派摄像头对前方文字进行连续拍照，故而该样例对环境光线、手写墨迹、环境对比度等因素要求很高。在实验过程中，当光线不充足时，会出现严重的不识别问题。说明样例在对图片进行预处理，划分文字所在区域的方面还有一定的不足。根据模型只能识别红色字体来看，推断应当为色域范围太小所致。应适当改变色域部分的参数，以提高文字区域识别率。

7 课程设计体会总结

通过本次实验，我们实现了在 Atlas 200DK 上运行汉字手写体识别样例。从部署开发环境到最终实现运行样例，我们经历了很多挫折与失败。

课程初期，我们对 Ubuntu 系统毫不了解，对其系统的使用需要在不断尝试中摸索经验，期间走了不少弯路，甚至卸载了系统重新安装。

由于没有掌握整个开发过程的总体框架，我们选择了错误的 CANN 软件安装用户，导致后续环境变量配置错误，第三方依赖无法安装的错综复杂的问题。好在后期，我们及时意识到此问题，重新制卡，部署开发环境和运行环境，为样例的运行提供先决条件。

我们选择了比较不错的运行案例，既用到了 Atlas 200DK 开发板，又用到了树莓派摄像头，充分利用了手头已有的资源。当然也处理了比较关键的几个硬件方面的问题：如何安装树莓派摄像头，如何处理开发板与树莓派摄像头的不识别问题。虽然这些问题很基础，但由于我们第一次接触，对之知之甚少，差点导致我们整个样例的失败。

总结来说，本次利用 Atlas 200DK 开发板，运行简单的人工智能案例，让我们收获很多，我们充分了解嵌入式人工智能开发的整个过程，这对我们后期学习人工智能方面的知识有了更加实在具体的参考。

8 程序源代码及注释

8.1 原始模型代码

```
name: "ResNet-18"                                     type: "Input"
                                                         top: "data"
layer {                                                 input_param { shape: { dim: 64 dim: 3
  name: "data"                                         dim: 112 dim: 112 } }
```

```

}

layer {
  bottom: "data"
  top: "conv1"
  name: "conv1"
  type: "Convolution"
  convolution_param {
    num_output: 64
    kernel_size: 7
    pad: 3
    stride: 2
    weight_filler {
      type: "msra"
    }
    bias_term: false
  }
}

layer {
  bottom: "conv1"
  top: "conv1"
  name: "bn_conv1"
  type: "BatchNorm"
}

layer {
  bottom: "conv1"
  top: "conv1"
  name: "scale_conv1"
  type: "Scale"
  scale_param {
    bias_term: true
  }
}

layer {
  bottom: "conv1"
  top: "conv1"
  name: "conv1_relu"
  type: "ReLU"
}

layer {
  bottom: "conv1"
  top: "pool1"
  name: "pool1"
  type: "Pooling"
  pooling_param {
    kernel_size: 3
    stride: 2
    pool: MAX
  }
}

layer {
  bottom: "pool1"
  top: "res2a_branch1"
  name: "res2a_branch1"
  type: "Convolution"
  convolution_param {
    num_output: 64
    kernel_size: 1
    pad: 0
    stride: 1
    weight_filler {
      type: "msra"
    }
    bias_term: false
  }
}

layer {
  bottom: "res2a_branch1"
  top: "res2a_branch1"
  name: "bn2a_branch1"
  type: "BatchNorm"
}

layer {
  bottom: "res2a_branch1"
  top: "res2a_branch1"
  name: "scale2a_branch1"
  type: "Scale"
  scale_param {
    bias_term: true
  }
}

```

```

    }
  }

  layer {
    bottom: "pool1"
    top: "res2a_branch2a"
    name: "res2a_branch2a"
    type: "Convolution"
    convolution_param {
      num_output: 64
      kernel_size: 3
      pad: 1
      stride: 1
      weight_filler {
        type: "msra"
      }
      bias_term: false
    }
  }

  }

  layer {
    bottom: "res2a_branch2a"
    top: "res2a_branch2a"
    name: "bn2a_branch2a"
    type: "BatchNorm"
  }

  }

  layer {
    bottom: "res2a_branch2a"
    top: "res2a_branch2a"
    name: "scale2a_branch2a"
    type: "Scale"
    scale_param {
      bias_term: true
    }
  }

  }

  layer {
    bottom: "res2a_branch2a"
    top: "res2a_branch2a"
    name: "res2a_branch2a_relu"
    type: "ReLU"
  }

  }

  layer {
    bottom: "res2a_branch2a"
    top: "res2a_branch2b"
    name: "res2a_branch2b"
    type: "Convolution"
    convolution_param {
      num_output: 64
      kernel_size: 3
      pad: 1
      stride: 1
      weight_filler {
        type: "msra"
      }
      bias_term: false
    }
  }

  layer {
    bottom: "res2a_branch2b"
    top: "res2a_branch2b"
    name: "bn2a_branch2b"
    type: "BatchNorm"
  }

  layer {
    bottom: "res2a_branch2b"
    top: "res2a_branch2b"
    name: "scale2a_branch2b"
    type: "Scale"
    scale_param {
      bias_term: true
    }
  }

  layer {
    bottom: "res2a_branch1"
    bottom: "res2a_branch2b"
    top: "res2a"
    name: "res2a"
    type: "Eltwise"
    eltwise_param {
      operation: SUM
    }
  }

```

```

    }
  }

  layer {
    bottom: "res2a"
    top: "res2a"
    name: "res2a_relu"
    type: "ReLU"
  }

  layer {
    bottom: "res2a"
    top: "res2b_branch2a"
    name: "res2b_branch2a"
    type: "Convolution"
    convolution_param {
      num_output: 64
      kernel_size: 3
      pad: 1
      stride: 1
      weight_filler {
        type: "msra"
      }
      bias_term: false
    }
  }

  }

  layer {
    bottom: "res2b_branch2a"
    top: "res2b_branch2a"
    name: "bn2b_branch2a"
    type: "BatchNorm"
  }

  layer {
    bottom: "res2b_branch2a"
    top: "res2b_branch2a"
    name: "scale2b_branch2a"
    type: "Scale"
    scale_param {
      bias_term: true
    }
  }
}

layer {
  bottom: "res2b_branch2a"
  top: "res2b_branch2a"
  name: "res2b_branch2a_relu"
  type: "ReLU"
}

layer {
  bottom: "res2b_branch2a"
  top: "res2b_branch2b"
  name: "res2b_branch2b"
  type: "Convolution"
  convolution_param {
    num_output: 64
    kernel_size: 3
    pad: 1
    stride: 1
    weight_filler {
      type: "msra"
    }
    bias_term: false
  }
}

layer {
  bottom: "res2b_branch2b"
  top: "res2b_branch2b"
  name: "bn2b_branch2b"
  type: "BatchNorm"
}

layer {
  bottom: "res2b_branch2b"
  top: "res2b_branch2b"
  name: "scale2b_branch2b"
  type: "Scale"
  scale_param {
    bias_term: true
  }
}

layer {

```



```

    bottom: "res2a"
    bottom: "res2b_branch2b"
    top: "res2b"
    name: "res2b"
    type: "Eltwise"
    eltwise_param {
        operation: SUM
    }
}

layer {
    bottom: "res2b"
    top: "res2b"
    name: "res2b_relu"
    type: "ReLU"
}

layer {
    bottom: "res2b"
    top: "res3a_branch1"
    name: "res3a_branch1"
    type: "Convolution"
    convolution_param {
        num_output: 128
        kernel_size: 1
        pad: 0
        stride: 2
        weight_filler {
            type: "msra"
        }
        bias_term: false
    }
}

layer {
    bottom: "res3a_branch1"
    top: "res3a_branch1"
    name: "bn3a_branch1"
    type: "BatchNorm"
}

layer {
    bottom: "res3a_branch1"

    top: "res3a_branch1"
    name: "scale3a_branch1"
    type: "Scale"
    scale_param {
        bias_term: true
    }
}

layer {
    bottom: "res2b"
    top: "res3a_branch2a"
    name: "res3a_branch2a"
    type: "Convolution"
    convolution_param {
        num_output: 128
        kernel_size: 3
        pad: 1
        stride: 2
        weight_filler {
            type: "msra"
        }
        bias_term: false
    }
}

layer {
    bottom: "res3a_branch2a"
    top: "res3a_branch2a"
    name: "bn3a_branch2a"
    type: "BatchNorm"
}

layer {
    bottom: "res3a_branch2a"
    top: "res3a_branch2a"
    name: "scale3a_branch2a"
    type: "Scale"
    scale_param {
        bias_term: true
    }
}

layer {
    bottom: "res3a_branch1"
    top: "res3a_branch2a"
    name: "res3a_branch2a"
    type: "Convolution"
    convolution_param {
        num_output: 128
        kernel_size: 3
        pad: 1
        stride: 2
        weight_filler {
            type: "msra"
        }
        bias_term: false
    }
}

layer {
    bottom: "res3a_branch2a"
    top: "res3a_branch2a"
    name: "bn3a_branch2a"
    type: "BatchNorm"
}

layer {
    bottom: "res3a_branch2a"
    top: "res3a_branch2a"
    name: "scale3a_branch2a"
    type: "Scale"
    scale_param {
        bias_term: true
    }
}

```

```

    bottom: "res3a_branch2a"
    top: "res3a_branch2a"
    name: "res3a_branch2a_relu"
    type: "ReLU"
  }

  layer {
    bottom: "res3a_branch2a"
    top: "res3a_branch2b"
    name: "res3a_branch2b"
    type: "Convolution"
    convolution_param {
      num_output: 128
      kernel_size: 3
      pad: 1
      stride: 1
      weight_filler {
        type: "msra"
      }
      bias_term: false
    }
  }
}

layer {
  bottom: "res3a_branch2b"
  top: "res3a_branch2b"
  name: "bn3a_branch2b"
  type: "BatchNorm"
}

layer {
  bottom: "res3a_branch2b"
  top: "res3a_branch2b"
  name: "scale3a_branch2b"
  type: "Scale"
  scale_param {
    bias_term: true
  }
}

layer {
  bottom: "res3a_branch1"
  bottom: "res3a_branch2b"
  top: "res3a"
  name: "res3a"
  type: "Eltwise"
  eltwise_param {
    operation: SUM
  }
}

layer {
  bottom: "res3a"
  top: "res3a"
  name: "res3a_relu"
  type: "ReLU"
}

layer {
  bottom: "res3a"
  top: "res3b_branch2a"
  name: "res3b_branch2a"
  type: "Convolution"
  convolution_param {
    num_output: 128
    kernel_size: 3
    pad: 1
    stride: 1
    weight_filler {
      type: "msra"
    }
    bias_term: false
  }
}

layer {
  bottom: "res3b_branch2a"
  top: "res3b_branch2a"
  name: "bn3b_branch2a"
  type: "BatchNorm"
}

layer {
  bottom: "res3b_branch2a"
  top: "res3b_branch2a"
  name: "scale3b_branch2a"
  type: "Scale"
  scale_param {
    bias_term: true
  }
}

layer {
  bottom: "res3b_branch1"
  bottom: "res3b_branch2a"
  top: "res3b"
  name: "res3b"
  type: "Eltwise"
  eltwise_param {
    operation: SUM
  }
}

```

```

    type: "Scale"
    scale_param {
        bias_term: true
    }
}

layer {
    bottom: "res3b_branch2a"
    top: "res3b_branch2a"
    name: "res3b_branch2a_relu"
    type: "ReLU"
}

layer {
    bottom: "res3b_branch2a"
    top: "res3b_branch2b"
    name: "res3b_branch2b"
    type: "Convolution"
    convolution_param {
        num_output: 128
        kernel_size: 3
        pad: 1
        stride: 1
        weight_filler {
            type: "msra"
        }
        bias_term: false
    }
}

layer {
    bottom: "res3b_branch2b"
    top: "res3b_branch2b"
    name: "bn3b_branch2b"
    type: "BatchNorm"
}

layer {
    bottom: "res3b_branch2b"
    top: "res3b_branch2b"
    name: "scale3b_branch2b"
    type: "Scale"
    scale_param {
        bias_term: true
    }
}

layer {
    bottom: "res3a"
    bottom: "res3b_branch2b"
    top: "res3b"
    name: "res3b"
    type: "Eltwise"
    eltwise_param {
        operation: SUM
    }
}

layer {
    bottom: "res3b"
    top: "res3b"
    name: "res3b_relu"
    type: "ReLU"
}

layer {
    bottom: "res3b"
    top: "res4a_branch1"
    name: "res4a_branch1"
    type: "Convolution"
    convolution_param {
        num_output: 256
        kernel_size: 1
        pad: 0
        stride: 2
        weight_filler {
            type: "msra"
        }
        bias_term: false
    }
}

layer {
    bottom: "res4a_branch1"
    top: "res4a_branch1"
    name: "bn4a_branch1"
    type: "BatchNorm"
}

```

```

    }
    layer {
      bottom: "res4a_branch1"
      top: "res4a_branch1"
      name: "scale4a_branch1"
      type: "Scale"
      scale_param {
        bias_term: true
      }
    }
    layer {
      bottom: "res3b"
      top: "res4a_branch2a"
      name: "res4a_branch2a"
      type: "Convolution"
      convolution_param {
        num_output: 256
        kernel_size: 3
        pad: 1
        stride: 2
        weight_filler {
          type: "msra"
        }
        bias_term: false
      }
    }
    layer {
      bottom: "res4a_branch2a"
      top: "res4a_branch2a"
      name: "bn4a_branch2a"
      type: "BatchNorm"
    }
    layer {
      bottom: "res4a_branch2a"
      top: "res4a_branch2a"
      name: "scale4a_branch2a"
      type: "Scale"
      scale_param {
        bias_term: true
      }
    }
  }
}

    bias_term: true
  }
}

    layer {
      bottom: "res4a_branch2a"
      top: "res4a_branch2a"
      name: "res4a_branch2a_relu"
      type: "ReLU"
    }

    layer {
      bottom: "res4a_branch2a"
      top: "res4a_branch2b"
      name: "res4a_branch2b"
      type: "Convolution"
      convolution_param {
        num_output: 256
        kernel_size: 3
        pad: 1
        stride: 1
        weight_filler {
          type: "msra"
        }
        bias_term: false
      }
    }

    layer {
      bottom: "res4a_branch2b"
      top: "res4a_branch2b"
      name: "bn4a_branch2b"
      type: "BatchNorm"
    }

    layer {
      bottom: "res4a_branch2b"
      top: "res4a_branch2b"
      name: "scale4a_branch2b"
      type: "Scale"
      scale_param {
        bias_term: true
      }
    }
  }
}

```

```

    }

    layer {
      bottom: "res4a_branch1"
      bottom: "res4a_branch2b"
      top: "res4a"
      name: "res4a"
      type: "Eltwise"
      eltwise_param {
        operation: SUM
      }
    }
  }

  layer {
    bottom: "res4a"
    top: "res4a"
    name: "res4a_relu"
    type: "ReLU"
  }

  layer {
    bottom: "res4a"
    top: "res4b_branch2a"
    name: "res4b_branch2a"
    type: "Convolution"
    convolution_param {
      num_output: 256
      kernel_size: 3
      pad: 1
      stride: 1
      weight_filler {
        type: "msra"
      }
      bias_term: false
    }
  }

  layer {
    bottom: "res4b_branch2a"
    top: "res4b_branch2b"
    name: "res4b_branch2b"
    type: "Convolution"
    convolution_param {
      num_output: 256
      kernel_size: 3
      pad: 1
      stride: 1
      weight_filler {
        type: "msra"
      }
      bias_term: false
    }
  }

  layer {
    bottom: "res4b_branch2b"
    top: "res4b_branch2b"
    name: "bn4b_branch2b"
    type: "BatchNorm"
  }

  layer {
    bottom: "res4b_branch2a"
    top: "res4b_branch2a"
    name: "scale4b_branch2a"
    type: "Scale"
    scale_param {
      bias_term: true
    }
  }

  layer {
    bottom: "res4b_branch2a"
    top: "res4b_branch2a"
    name: "res4b_branch2a_relu"
    type: "ReLU"
  }

  layer {
    bottom: "res4b_branch2a"
    top: "res4b_branch2b"
    name: "res4b_branch2b"
    type: "Convolution"
    convolution_param {
      num_output: 256
      kernel_size: 3
      pad: 1
      stride: 1
      weight_filler {
        type: "msra"
      }
      bias_term: false
    }
  }

  layer {
    bottom: "res4b_branch2b"
    top: "res4b_branch2b"
    name: "bn4b_branch2b"
    type: "BatchNorm"
  }

```



```

        bottom: "res5a_branch2a"
        top: "res5a_branch2a"
        name: "scale5a_branch2a"
        type: "Scale"
        scale_param {
            bias_term: true
        }
    }
}

layer {
    bottom: "res5a_branch2a"
    top: "res5a_branch2a"
    name: "res5a_branch2a_relu"
    type: "ReLU"
}

layer {
    bottom: "res5a_branch2a"
    top: "res5a_branch2b"
    name: "res5a_branch2b"
    type: "Convolution"
    convolution_param {
        num_output: 512
        kernel_size: 3
        pad: 1
        stride: 1
        weight_filler {
            type: "msra"
        }
        bias_term: false
    }
}

layer {
    bottom: "res5a_branch2b"
    top: "res5a_branch2b"
    name: "bn5a_branch2b"
    type: "BatchNorm"
}

layer {
    bottom: "res5a_branch2b"
    top: "res5a_branch2b"
    name: "scale5a_branch2b"
    type: "Scale"
    scale_param {
        bias_term: true
    }
}

layer {
    bottom: "res5a_branch1"
    bottom: "res5a_branch2b"
    top: "res5a"
    name: "res5a"
    type: "Eltwise"
    eltwise_param {
        operation: SUM
    }
}

layer {
    bottom: "res5a"
    top: "res5a"
    name: "res5a_relu"
    type: "ReLU"
}

layer {
    bottom: "res5a"
    top: "res5b_branch2a"
    name: "res5b_branch2a"
    type: "Convolution"
    convolution_param {
        num_output: 512
        kernel_size: 3
        pad: 1
        stride: 1
        weight_filler {
            type: "msra"
        }
        bias_term: false
    }
}

layer {
    bottom: "res5b_branch2a"
    top: "res5b_branch2a"
    name: "bn5b_branch2a"
    type: "BatchNorm"
}

layer {
    bottom: "res5b_branch2a"
    top: "res5b_branch2a"
    name: "scale5b_branch2a"
    type: "Scale"
    scale_param {
        bias_term: true
    }
}

```

```

    top: "res5b_branch2a"
    name: "bn5b_branch2a"
    type: "BatchNorm"
  }

  layer {
    bottom: "res5b_branch2a"
    top: "res5b_branch2a"
    name: "scale5b_branch2a"
    type: "Scale"
    scale_param {
      bias_term: true
    }
  }

  layer {
    bottom: "res5b_branch2a"
    top: "res5b_branch2a"
    name: "res5b_branch2a_relu"
    type: "ReLU"
  }

  layer {
    bottom: "res5b_branch2a"
    top: "res5b_branch2b"
    name: "res5b_branch2b"
    type: "Convolution"
    convolution_param {
      num_output: 512
      kernel_size: 3
      pad: 1
      stride: 1
      weight_filler {
        type: "msra"
      }
      bias_term: false
    }
  }

  layer {
    bottom: "res5b_branch2b"
    top: "res5b_branch2b"
    name: "bn5b_branch2b"
    type: "BatchNorm"
  }

  layer {
    bottom: "res5b_branch2b"
    top: "res5b_branch2b"
    name: "scale5b_branch2b"
    type: "Scale"
    scale_param {
      bias_term: true
    }
  }

  layer {
    bottom: "res5a"
    bottom: "res5b_branch2b"
    top: "res5b"
    name: "res5b"
    type: "Eltwise"
    eltwise_param {
      operation: SUM
    }
  }

  layer {
    bottom: "res5b"
    top: "res5b"
    name: "res5b_relu"
    type: "ReLU"
  }

  layer {
    bottom: "res5b"
    top: "pool5"
    name: "pool5"
    type: "Pooling"
    pooling_param {
      kernel_size: 3
      stride: 1
      pool: AVE
    }
  }

  layer {
    bottom: "res5b_branch2b"
    top: "res5b_branch2b"
    name: "bn5b_branch2b"
    type: "BatchNorm"
  }

```



```

bottom: "pool5"                                type: "xavier"
top: "fc3755"                                    }
name: "fc3755"                                bias_filler {
type: "InnerProduct"                            type: "constant"
param {                                          value: 0
    lr_mult: 1                                    }
    decay_mult: 1                                }
}                                                }
param {                                          layer {
    lr_mult: 2                                    name: "prob"
    decay_mult: 1                                type: "Softmax"
}                                                bottom: "fc3755"
inner_product_param {                          top: "prob"
    num_output: 3755                            }
    weight_filler {

```

8.2 对应的 cfg 代码

```

aipp_op {
related_input_rank : 0
src_image_size_w : 112
src_image_size_h : 112
crop : false
aipp_mode : static
input_format : RGB888_U8
csc_switch : false
rbuv_swap_switch : true
}

```

[参考文献]

- [1] Atlas 200 DK 开发者套件 (1.0.10.alpha) Media API
- [2] Atlas 200 DK 开发者套件 (1.0.10.alpha) 环境部署
- [3] Atlas 200 DK 开发者套件 (1.0.10.alpha) 外围设备驱动接口
- [4] HandWrite Atlas 200DK