



中國海洋大學
OCEAN UNIVERSITY OF CHINA

实验报告

单衍喆

2024-9-14

GitHub地址: <https://github.com/Venuss1/course.git>



目录

1	<u>实验内容</u>	1
2	<u>实验设计</u>	1
2.1	调试及性能分析	1
2.1.1	gdb调试c语言及c++程序	1
2.1.2	基本命令	2
2.2	元编程	2
2.2.1	Cmake构建流程	2
2.2.2	Cmake构建项目	3
2.2.3	makefile基本语法	4
2.3	大杂烩	4
2.3.1	修改键位映射	4
2.3.2	MarkDown纯文本编辑	5
2.3.3	vpn	6
2.4	pytorch应用	6
2.4.1	transform	6
2.4.2	DataLoader	7
2.4.3	train epoch	7
2.4.4	loss function	8
2.4.5	optimizer	8
3	<u>实验心得</u>	10

1 实验内容

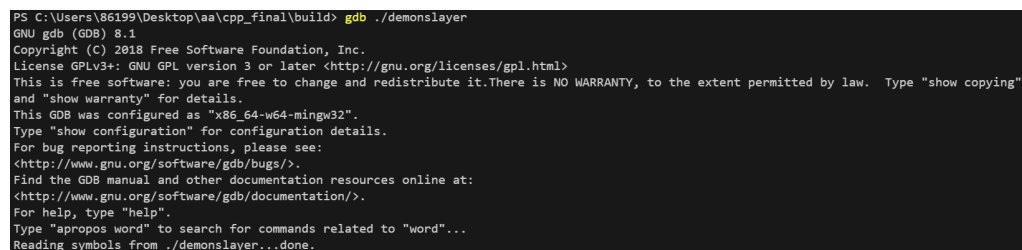
1. 调试及性能分析
2. 元编程
3. 大杂烩
4. pytorch应用

2 实验设计

2.1 调试及性能分析

2.1.1 gdb调试c语言及c++程序

1. 编译可执行程序
2. 启动gdb



```
PS C:\Users\86199\Desktop\aa\cpp_final\build> gdb ./demonslayer
GNU gdb (GDB) 8.1
Copyright (C) 2018 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it. There is NO WARRANTY, to the extent permitted by law. Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-w64-mingw32".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from ./demonslayer...done.
```

图 1: 启动gdb

3. 设置断点
4. gdb调试程序

2.1.2 基本命令

表 1: 基本命令

简写	全称	作用
b	break	设置断点
r	run	运行程序
n	next	继续执行
f	finish	结束当前函数
c	continue	继续运行程序
p	print	打印

2.2 元编程

2.2.1 Cmake构建流程

1. 确定构建目录:
“Out-of-source”，构建文件放在源代码目录之外的独立目录中
2. 使用CMake生成构建文件:

```
CMake -DCMAKE_BUILD\ _TYPE=Release ..
```

3. 编译和构建:

```
make <your_project_name>
```

4. 清理构建文件:

```
make clean
```

2.2.2 Cmake构建项目

- 规定最低的版本:

```
cmake minimum required(VERSION 3.12)
```

- 构建项目名称:

```
project(cpp final)
```

- 设置变量:

```
set(mainTargetName "demonlayer")
```

- 自定义命令:

```
add_custom_command(  
  <type> <name> <COMMAND> <WORKING_DIRECTORY> <OUTPUT> <DEPENDS>  
  [<VERBOSE>]  
  [<COMMENT>]  
)
```

- 迭代:

```
foreach(<var> <item> <condition>)  
  <statement>  
endforeach()
```

2.2.3 makefile基本语法

- 编译命令:

```
hello.c main.c test.c
gcc main.c test.c
```

- 目标文件

```
main.o: main.c
gcc -c main.c
```

- 伪命令

```
.PHONY:clean
clean:
rm -f *.o hello
```

- 变量

targets = hello

2.3 大杂烩

2.3.1 修改键位映射

使用AutoHotKey修改键位映射

1. 下载安装AutoHotKey
2. 配置脚本文件: 交换将capslock大写锁定键映射到esc
3. 运行脚本文件
4. 配置开机启动项

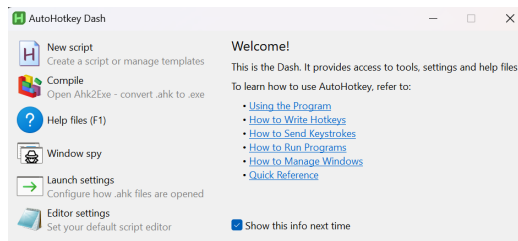


图 2: autoHotKey

2.3.2 Markdown纯文本编辑

- 标题

标题1

标题2

- 段落

<p>

这是一个段落。

<p>

- 链接

链接文字

- 图片

- 代码块

```
<pre>  
<code>  
这是一些代码。  
</code>  
</pre>
```

2.3.3 vpn

1. 下载安装clash for windows

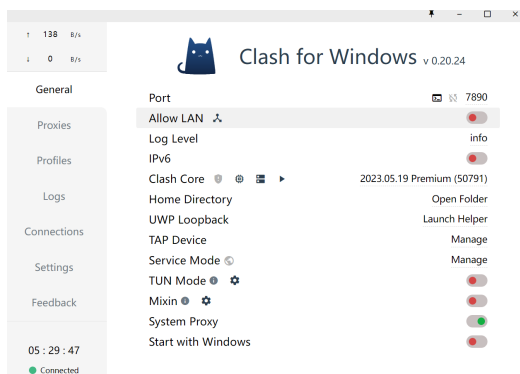


图 3: clash for windows

2. 配置profiles

3. 打开system Proxy

2.4 pytorch应用

2.4.1 transform

帮助用户在训练和推理过程中对数据进行预处理、数据增强、数据转换等操作

```
data_transform = {  
    "train": transforms.Compose([transforms.RandomResizedCrop(224),
```



```

        transforms.RandomHorizontalFlip(),
        transforms.ToTensor(),
        transforms.Normalize([0.485, 0.456,
"val": transforms.Compose([transforms.Resize(256),
        transforms.CenterCrop(224),
        transforms.ToTensor(),
        transforms.Normalize([0.485, 0.456, 0.40

```

2.4.2 DataLoader

帮助用户自动加载训练数据，并行处理数据，以及控制数据读取的速度和策略

```

train_loader = torch.utils.data.DataLoader(train_dataset,
        batch_size=batch_size, shuffle=True,
        num_workers=nw)

```

2.4.3 train epoch

核心循环

```

epochs = 3
best_acc = 0.0
save_path = './resNet34.pth'
train_steps = len(train_loader)
for epoch in range(epochs):
    net.train()
    running_loss = 0.0
    train_bar = tqdm(train_loader, file=sys.stdout)

    for step, data in enumerate(train_bar):
        images, labels = data

```

```

optimizer.zero_grad()
logits = net(images.to(device))
loss = loss_function(logits, labels.to(device))
loss.backward()
optimizer.step()

running_loss += loss.item()

```

2.4.4 loss function

用于衡量模型预测和真实值之间的差距

- MSE（均方误差）

```

def mse_loss(y_true, y_pred):
    return np.mean((y_true - y_pred) ** 2, axis=1)

```

- Cross-Entropy（交叉熵）

```

def cross_entropy_loss(y_true, y_pred):
    return -np.sum(np.log(y_pred) * y_true, axis=1)

```

2.4.5 optimizer

用于最小化损失函数，从而实现模型的训练和优化

- Adam（自适应矩阵梯度下降）

```

def adam(params, grads, lr=0.001, beta1=0.9, beta2=0.999, epsilon=1e-8):
    m = params.data.mean()
    v = params.grad.data.mean()

```

```

for i in range(params.size):
    m_hat = beta1 * m + (1 - beta1) * params.data[i]
    v_hat = beta2 * v + (1 - beta2) * grads.data[i]
    params.data[i] = -lr * m_hat / (np.sqrt(v_hat) + epsilon)
    v[i] = beta2 * v_hat
return params

```

- SGD（随机梯度下降）

```

def adam(params, grads, lr=0.001, beta1=0.9, beta2=0.999, epsilon=1e-8):
    m = params.data.mean()
    v = params.grad.data.mean()
    for i in range(params.size):
        m_hat = beta1 * m + (1 - beta1) * params.data[i]
        v_hat = beta2 * v + (1 - beta2) * grads.data[i]
        params.data[i] = -lr * m_hat / (np.sqrt(v_hat) + epsilon)
        v[i] = beta2 * v_hat
    return params

```

3 实验心得

通过本次实验，我学会了运用pytorch进行深度学习的基本步骤和基本参数的设置，为之后人工智能的学习打下基础；

我学会了用gdb调试c语言及c++代码的基本步骤，学会了用gdb调试寻找代码问题；

了解了CMake构建项目的基本流程及CMake的基本命令，学会了makefile的基本语法；

学会了许多和计算机有关的知识，包括键盘映射，vpn网络设置，GitHub的使用及纯文本编辑工具Markdown。