学习笔记 by 赵韶涵

任务一：手搓手势识别模型

1.Q&A

Q1.模型开发通常包含哪些步骤，每个步骤要做什么，它的作用又是什么？

A:

1. 数据收集：收集训练模型所用数据。训练数据的质量和数量会直接影响模型的性能。
2. 数据清洗与预处理：清理不适于训练的数据，并进行格式转换、归一化等处理。作用是确保数据的质量，减少噪声对模型的影响，提高训练效果。
3. 模型选择：选择适合解决当前问题的模型类型。不同的模型在不同任务中表现各异，选择合适的模型能够确保训练效果。
4. 模型训练：使用训练数据集对模型参数进行调整，让其学习数据中的模式，从而提高模型精度，降低损失。
5. 模型评估：使用验证数据集对每轮训练结果进行评估，检查其性能。确保其在未见过的数据上也能保证性能，防止过拟合。
6. 模型优化：调整模型的超参数或者改善模型结构，以进一步提升其性能
7. 模型测试：使用测试数据集对模型进行最终评估，检验其泛化能力。

Q2.手势识别模型的pipeline是什么？

1. 数据读取：从数据集中读取图片及其标签
2. 图像预处理：调整图像大小并将其标准化
3. 划分数据集：将数据集划分为训练集，验证集，测试集供之后使用。
4. 模型构建：确定合适的模型架构
5. 模型训练：使用反向传播对模型进行多轮训练，以逐步提升其预测能力
6. 模型评估：使用测试数据集对模型进行评估，检验其性能。
7. 模型调参：调整模型超参数以及模型架构，以提升其性能。

Q3.目前手势识别的模型有哪些？哪一些模型适合部署在硬件上？部署到硬件上的难点有哪些？

1. 常见的手势识别模型：CNN架构如VGG，ResNet等；以及YOLO模型等
2. 适合部署到硬件上的模型：轻量级的CNN模型如MobileNet, EfficientNet等；Tiny YOLO, TensorFlow Lite等等

3. 部署到硬件上的难点：

- 计算资源有限：嵌入式设备通常计算能力和内存等比较有限，需要轻量化的模型
- 实时性要求：若需要实时推理能力，需要在保证精准度的同时提高推理速度。
- 功耗限制：一些嵌入式设备由电池供电，需要保证模型的效率。
- 模型兼容性：不同的硬件架构可能需要一些模型转换工具等以匹配其固件。
- ...

2.学习过程中遇到的一些问题

1.确保数据处理方式适用于数据集

最开始时我使用的是`datasets`模块里的`ImageFolder`类来加载数据，但这种方式只适用于ImageFolder形式组织的数据集，使用这种方式加载Kaggle数据集出来的数据标签混乱，最终我采取了适于Kaggle数据集的数据加载方式得以正常加载。

2.模型训练过程的一些报错信息

```
RuntimeError                              Traceback (most recent call
last)
Cell In[12], line 17
     16 for inputs, labels in train_loader:
---> 17     inputs, labels = inputs.to('cuda'), labels.to('cuda')
     18     model.to('cuda')

RuntimeError: CUDA error: device-side assert triggered
CUDA kernel errors might be asynchronously reported at some other API
call, so the stacktrace below might be incorrect.
For debugging consider passing CUDA_LAUNCH_BLOCKING=1.
Compile with TORCH_USE_CUDA_DSA to enable device-side assertions.
```

解决过程：之后检查错误是由于数据标签超出范围，由于输出层设置的是6个类别，但却把10个类别的数据都读取了，修改数据范围后依然报错，最后发现是由于`CrossEntropyLoss`要求标签是从 0 开始的整数，而数据集标签为1-6。最终解决方法是读取Lable时将其减一。

3.Debug时用的环境变量

```
export CUDA_LAUNCH_BLOCKING=1
```

在通常的情况下，CUDA 操作是异步执行的，主机代码会在 CUDA 内核启动后继续执行，而不等待其完成。设置`CUDA_LAUNCH_BLOCKING=1`会使得所有 CUDA 调用变为同步执行，也就是说，主机代码会等待每个 CUDA 操作完成后才继续执行下一步操作。

这样做的好处是可以更容易定位和调试 CUDA 相关的错误，因为错误会在发生的地方立即报告，而不会延迟到后续的 API 调用时才报错。不过，缺点是在调试时性能会有所下降，因为同步执行会引入额外的等待时间。

4.训练模型时，GPU占用时高时低，总是跑不满的原因

- 数据加载瓶颈：如果数据加载或预处理速度跟不上 GPU 的计算速度，GPU 会等待 CPU 完成这些任务，从而导致 GPU 使用率下降。可以尝试使用多线程数据加载。
- 小批量训练：如果使用的批量大小（batch size）较小，GPU 的计算资源可能无法完全被利用。增加批量大小可以提高 GPU 的利用率，前提是显存允许。

5.最后部署时，使用不了摄像头

由于我是在WSL上进行开发，最后使用OpenCV获取摄像头数据时出现错误

```
[ WARN:0@501.018] global cap_v4l.cpp:999 open VIDEOIO(V4L2:/dev/video0):
can't open camera by index
[ERROR:0@501.020] global obsensor_uvc_stream_channel.cpp:158
getStreamChannelGroup Camera index out of range
```

使用`ls /dev/`发现没有`video0`设备，在Powershell里使用`usbipd`命令将摄像头挂载后依然没有`video0`设备，然而，使用`lsusb`命令可以检测到USB视频设备。

使用`lsmod | grep videodev`命令检查摄像头驱动是否加载，提示：

```
modprobe: FATAL: Module videodev not found in directory
/lib/modules/5.15.153.1-microsoft-standard-WSL2
```

查阅资料发现：WSL2不支持直接访问物理摄像头设备，因此，videodev模块无法在WSL2中加载。

使用我的ubuntu系统来运行代码，发现不能使用cuda，检查了解到是由于nvidia驱动未能正确安装，一直报错，由于时间原因就暂时先不处理，只能先用windows系统拍照，然后再读取照片文件进行识别。

.....

3.一些学习笔记

见`note`文件夹

4.不同超参数的准确率变化（使用同一CNN模型, num_classes=6）

num_epoch	learning_rate	Loss	Accuracy
5	0.01	0.0123	99.72%
5	0.05	1.7929	16.39%
5	0.001	0.0027	99.78%

num_epoch	learning_rate	Loss	Accuracy
10	0.001	0.0001	99.94%

5.不同网络架构的准确率变化（使用同一超参数5/0.001, num_classes=6）

类型	Loss	Accuracy	备注
CNN	0.0027	99.78%	3卷积层 2全连接层
CNN	0.0132	99.67%	2卷积层 1全连接层
CNN	0.0294	99.67%	1卷积层 1全连接层
MLP	0.1546	95.89%	3全连接层
MLP	0.3492	89.72%	1全连接层

任务二: 初步学习ESP32S3

1.一些学习心得

1.将usb设备挂载到WSL上的方法

1. 安装 USBIPD-WIN 项目
2. 在PowerShell中运行`usbipd list`查看状态及BUSID等信息
3. 使用`usbipd bind --busid <busid>`来将设备设置为共享状态
4. 使用`usbipd attach --wsl --busid <busid>`来将设备附加到WSL中
5. 使用`usbipd detach --busid <busid>`将设备断开

2.烧录时的JTAG, UART, DFU如何选择

- JTAG: 适合复杂的调试需求（如逐行调试和监控寄存器），需要硬件调试器。
- UART: 常用于简单的串行通信和日志输出，适合日常开发和调试。
- DFU: 适合在不依赖现有固件或操作系统的情况下进行固件更新，尤其适合恢复设备。

3.使用流程（命令行）

1. 环境变量：`. $HOME/esp/esp-idf/export.sh`
2. 设置目标芯片：`idf.py set-target esp32s3`
3. 运行工程配置工具：`idf.py menuconfig`
4. 编译工程：`idf.py build`
5. 烧录到设备：`idf.py -p PORT flash`
6. 监视输出：`idf.py -p PORT monitor`

4.获取串口路径

```
ls /dev/tty*
```

2.学习过程中遇到的一些问题

1.在 VSCode 中使用 ESP-IDF 扩展来构建 Arduino 项目

我使用的ESP32S3-CAM板子的商家提供的代码是在Arduino IDE中运行的，但我最初进行开发时使用的是VS Code里的ESP-IDF扩展，因此我就查阅一些资料来在 VSCode 中使用 ESP-IDF 扩展来构建 Arduino 项目，过程中也遇到一些棘手的问题。例如：

```
CMake Error at /home/panda/esp/esp-idf/tools/cmake/build.cmake:589
(message):
  ERROR: Because no versions of espressif/arduino-esp32 match >3.0.2,
<3.0.3
  || >3.0.3,<3.0.4 || >3.0.4,<4.0.0

   and espressif/arduino-esp32 (3.0.2) depends on idf (>=5.1,<5.2),
espressif/arduino-esp32 (>=3.0.2,<3.0.3 || >3.0.3,<3.0.4 || >3.0.4,<4.0.0)
requires idf (>=5.1,<5.2).

  And because espressif/arduino-esp32 (3.0.3) depends on both idf
  (>=5.1,<5.2) and idf (>=5.1,<5.2), espressif/arduino-esp32 (>=3.0.2,
<4.0.0)
  requires idf (>=5.1.0,<5.2.0).

  So, because no versions of idf match >=5.1.0,<5.2.0

   and project depends on espressif/arduino-esp32 (^3.0.2), version
solving failed.
```

最初我使用的是ESP-IDF的5.4.0版本和arduino-esp32的3.0.4版本。然而，arduino-esp32目前的最新版本是3.0.4，仅能支持到ESP-IDF的5.1.4版本，因此我只能给ESP-IDF降级。

降级过程遇到的问题由于篇幅在此不多赘述...

最终，由于用ESP-IDF扩展构建Arduino项目过程缺失的库，格式内容等报错实在太多，我放弃了这个途径...

2.调用摄像头时的引脚声明问题

使用ESP-IDF自带的例子并不能正常调用摄像头，了解后发现是引脚声明有误，向商家要来了示例代码，在示例代码中找到了引脚定义的部分，将其复制到代码中即可正常使用摄像头。

3.无法访问串口设备

```
[Errno 13] could not open port /dev/ttyACM0: [Errno 13] Permission denied:
'/dev/ttyACM0'
Connection to /dev/ttyACM0 failed. Available ports:
/dev/ttyACM0
```

该报错提示为权限问题，解决方法：将用户添加到 dialout 组（未成功），或运行`sudo chmod 666 /dev/ttyACM0` 临时更改设备权限

4.未进入编程模式

```
A fatal error occurred: Failed to connect to ESP32-S3: No serial data
received.
For troubleshooting steps visit:
https://docs.espressif.com/projects/esptool/en/latest/troubleshooting.html
```

ESP32-S3 开发板在烧录固件时通常需要进入编程模式。有些板子会自动处理这个过程，但有些需要手动操作。

1. 按住 BOOT 按钮
2. 在按住 BOOT 按钮的同时，短按RESET按钮
3. 松开 BOOT 按钮。此时，开发板应该进入编程模式，准备接收固件。

.....

任务三：模型轻量化部署在ESP32S3上

将模型参数转化为ESP-DL格式

1. 将模型导出为 ONNX 格式

```
torch.onnx.export(model, input, "model.onnx")
```

2. 使用 ESP-DL 工具将 ONNX 转换为 ESP-DL 格式

```
python onnx_to_espdl.py --input model.onnx --output model_espdl
```

3. 在 ESP-DL 项目中集成模型

```
#include "model_espdl.h"

// 初始化和推理代码
esp_dl_model_t model;
esp_dl_load_model(&model, model_espdl_data);
```

非常抱歉，其他内容由于时间安排不当还未能学习完成，望谅解...