

云天励飞DeepEye1000开发板快速上手手册

版本：v1.1.2

更新记录

版本	修改日期	修改说明
V1.0.0	2020.7.20	初始版本
V1.0.1	2020.10.13	增加usb模式/net模式切换操作说明
V1.1.0	2020.12.2	更新设备管理工具为desdk，适配v1.5.0版本交付结构
V1.1.1	2021.2.22	增加应用开发章节
	2021.8.20	调整环境搭建描述，由于detvm采用单独docker

目录

1 概述

2 开发套件

2.1 硬件连接

2.2 DEngine交付结构

3 环境搭建

3.1 解压版本

3.2 配置软件环境

4 固件烧录

4.1 DP1000固件烧录

4.1.1 切换到烧录模式

4.1.2 执行烧录

4.1.3 切换回工作模式

4.1.4 版本验证

5 USB和NET模式切换

5.1 版本要求

5.2 连接要求

5.3 切换为NET模式

5.4 切换为USB模式

5.5 NET模式下IP

5.5.1 自动获取IP

5.5.2 手动指定IP

5.6 无串口切换NET模式

6 模型评估

6.1 模型编译仿真

6.2 模型板上评估

6.2.1 单次评估

6.2.2 benchmark评估

7 应用开发

7.1 代码编译

7.2 程序运行

8 FAQ

8.1 烧录或下载过程中出现USB初始化失败

8.2 使用VMWare虚拟机时USB未连接上

8.3 连接时报错Server Bind Port : Address already in use

1 概述

本文档主要介绍如何使用DeepEye1000开发板平台进行模型评估和软件开发。

2 开发套件

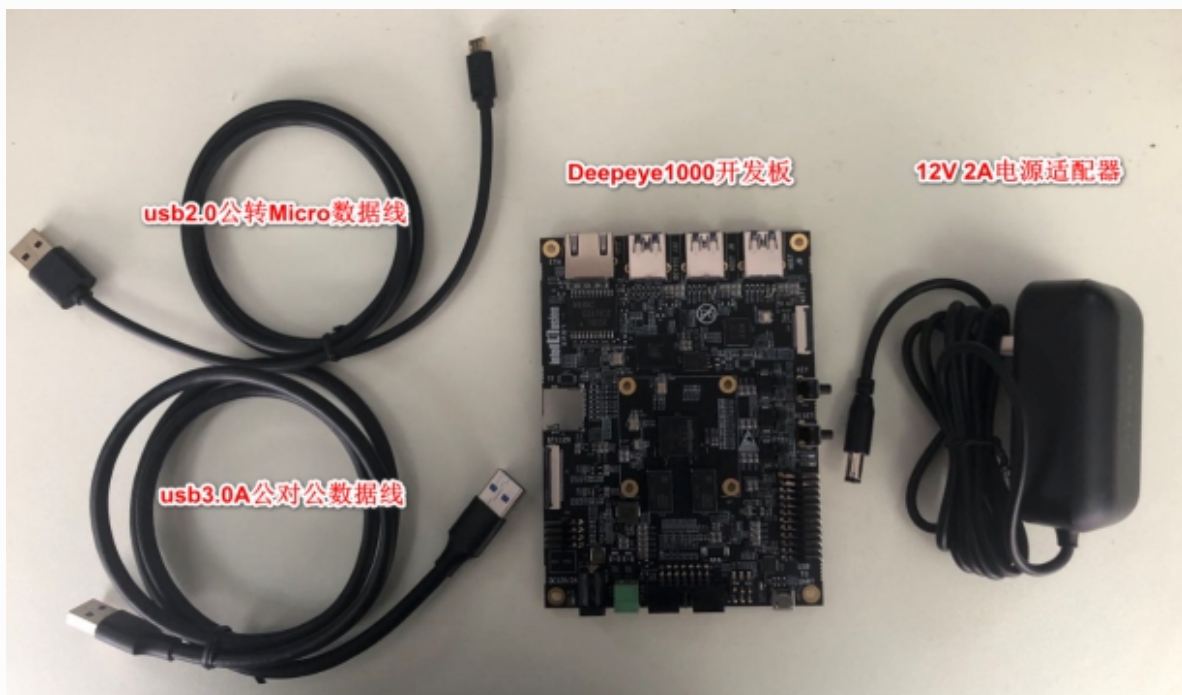
2.1 硬件连接

开发前需要准备以下硬件物料：

- 1) DeepEye1000开发板1块
- 2) 12V 2A电源适配器1个
- 3) usb2.0公转Micro数据线1根
- 4) usb3.0公对公数据线1根

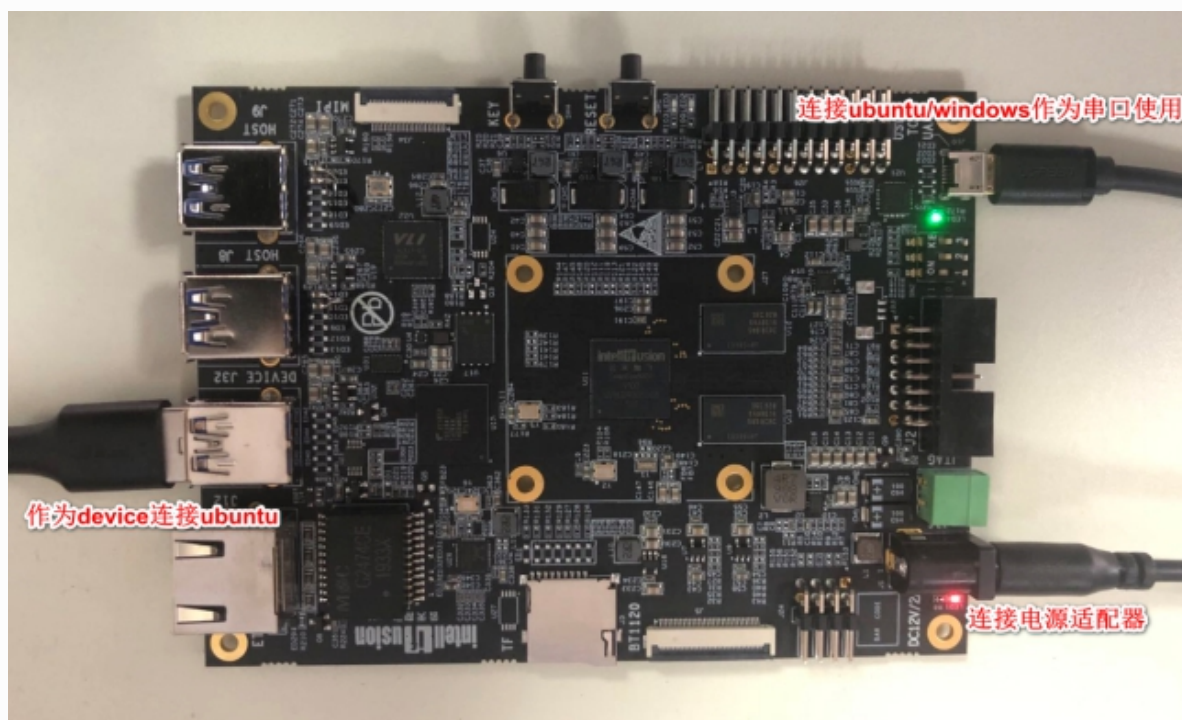
其中1)-3) 包含在DeepEye1000开发板套件中。

如图所示：



开发板提供USB连接和以太网连接2种方式供客户选择，但烧录版本只能使用USB连接。

USB连接方式如图所示：



USB3.0公对公数据线连接开发板的USB DEVICE口，作为版本烧录和通信使用。USB2.0公转Micro数据线连接开发板的Micro USB口，作为串口调试使用，波特率为115200。

当开发板的USB线连接主机后，可通过lsusb命令确认。

```
lubin@ubuntu:~$ lsusb
Bus 001 Device 002: ID 0525:a4a0 Netchip Technology, Inc. Linux-USB "Gadget Zero"
```

如果使用以太网连接，直接将以太网线插入开发板的以太网口，同时拔出USB DEVICE口的USB线。

2.2 DEngine交付结构

DEngine开发包是DeepEye芯片软件开发包的基础结构，所有平台的交付都是在此之上增加或修改部分平台特有的软件库。DEngine版本包解压展开后目录如下：

```
├─ DEngine
│   ├── detvm
│   ├── desdk
│   │   ├── archive
│   │   ├── platform
│   │   ├── cpp
│   │   └── python
│   ├── model
│   ├── data
│   └── tools
```

各子目录说明如下：

目录	内容
detvm	存放DP1000模型编译工具链
desdk	存放DP1000 芯片固件和SDK
archive	存放DP1000 芯片固件，按硬件形态存放
platform	存放主控和芯片侧的库文件和工具，按系统和cpu类型存放
cpp	存放C++头文件和示例源码
python	存放python示例源码
model	存放示例用到的原始模型
data	存放示例用到的数据，如图片
tools	存放平台工具等

3 环境搭建

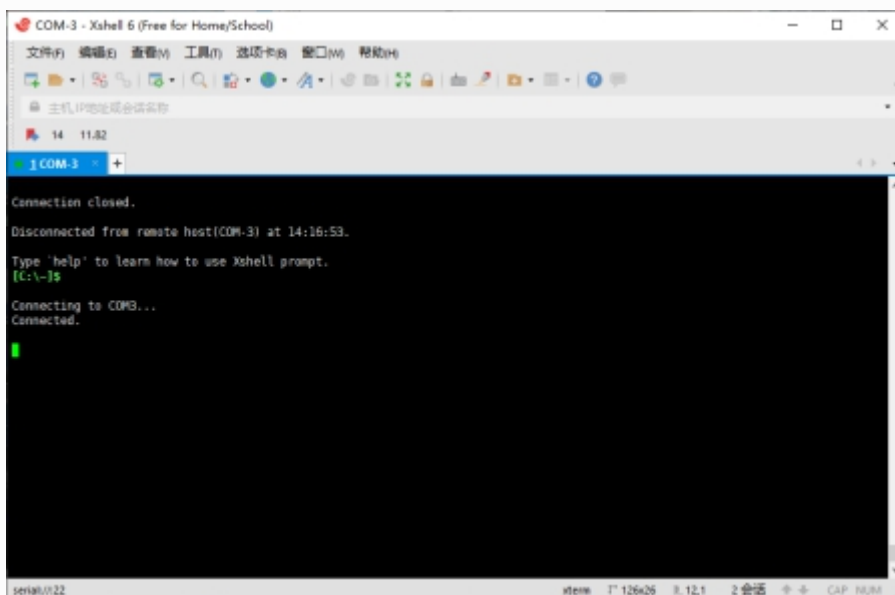
3.1 解压版本

将DEngine压缩包解压到ubuntu x64机器上。为release_packages.sh脚本增加可执行权限，并运行该脚本，输入y，进一步解压缩子压缩包。

```
# cd DEngine
# chmod +x release_packages.sh
# ./release_packages.sh
```

按照上面介绍的硬件连接方式将开发板的USB和串口连接到主机上。

在主机上打开串行口工具，配置为115200 8N1，连接上芯片的串口，如图所示：



3.2 配置软件环境

应用开发可在linux参见《云天励飞DEngine集成开发指南》环境搭建章节的描述搭建环境，如果选择在DEngine_docker中运行，直接按照说明安装并进入docker。如果选择不在DEngine_docker中运行，需要自行安装依赖软件，可参考/DEngine/tools/env下的环境配置脚本。示例需要依赖的软件主要有：

软件名	版本	说明
cmake	3.10	用于编译c++示例代码
gcc/g++	4.8.5	用于编译c++示例代码
opencv	3.1.0	用于图片读取、解码、打框和保存，主要在图片检测、视频检测示例中使用
ffmpeg	4.1.4	用于视频取流显示，主要在视频检测、视频检测跟踪示例中使用
gstreamer	1.14.5	用于VideoInput标准Node取流，如自行取流可不依赖
python	3.6	用于python推理示例
opencv-python	4.1.2	用于python推理示例
decorator	4.4	用于python推理示例
xlsxwriter	1.3.7	用于生成benchmark统计表格

非DEngine_docker环境下设置步骤：

1. 板上运行需要使用root权限，使用sudo su切换到root用户。
2. 配置平台环境变量，以linux-x64平台为例

```
# export TARGET_TYPE=host
# export TARGET_OS=linux
# export TARGET_CPU=x64
```

3. 运行env.sh脚本，将根目录映射到/DEngine

```
# sh env.sh
```

4. 切换根目录，执行：

```
# cd /DEngine
```

无论是否是DEngine_docker环境：

1. 如果需要重新烧录固件，参见固件烧录章节。如果不需要，跳过此步骤。
2. 如果准备使用usb连接芯片，通过usbprop.sh脚本调用设备管理工具desdk生成usb配置文件usbprop.ini，参数为连接的芯片数。如果只连接了1个开发板，则参数为1，以此类推。执行命令，usbprop.ini文件将生成在/DEngine目录下。

```
# sh tools/usbprop.sh 1 -F
```

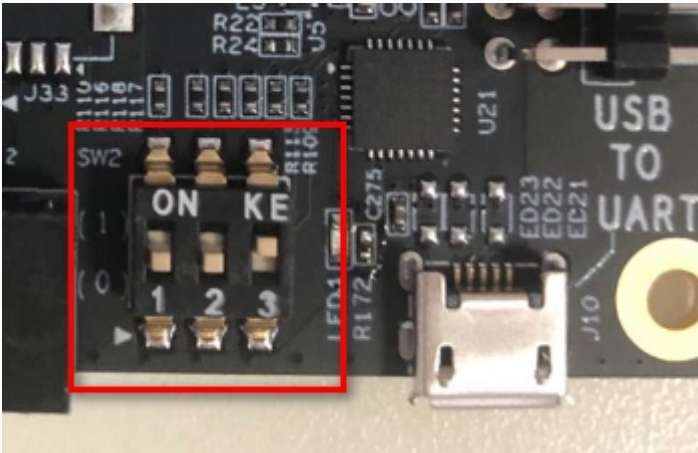
注：-F参数在没有gpio或smbus对芯片进行控制时使用，表示强制生成，可以控制时不建议使用。

4 固件烧录

4.1 DP1000固件烧录

4.1.1 切换到烧录模式

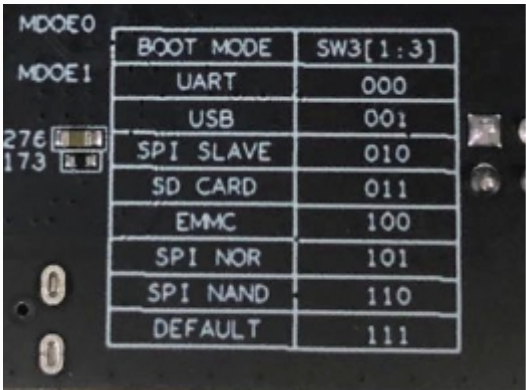
拨码开关位于DeepEye1000的USB转串行口位置处，如图所示：



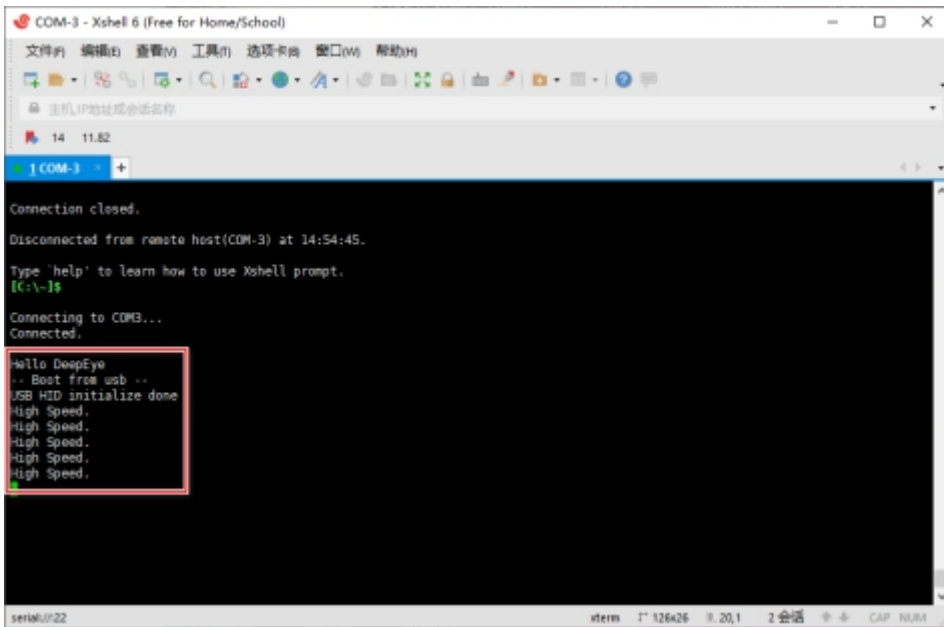
拨码开关由三位组成，由左至右分别由高到低排列。拨码开关位于黄线以上代表1，位于黄线以下代表0。三个拨码开关共可表示8种启动模式。拨码开关明细如表所示：

BOOT MODE	SW3[1:3]	本文档需要关注的启动模式
UART	000	
USB	001	USB方式烧录版本使用此模式
SPI SLAVE	010	
SD CARD	011	
EMMC	100	正常启动使用此模式
SPI NOR	101	
SPI NAND	110	
DEFAULT	111	

其中，拨码开关明细表在DeepEye1000开发板的背面有印刷，如图所示：



将拨码开关拨到001状态，进入USB模式，按Reset键重启开发板，此时串口输出如下：



4.1.2 执行烧录

通过burn.sh脚本使用desdk工具进行烧写，参数为连接的芯片数。如果只连接了1个开发板，则为1或不填。

```
# sh tools/burn.sh
```

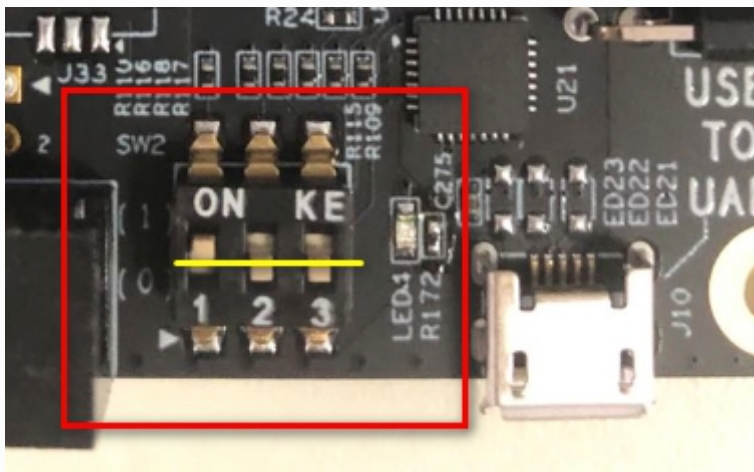
等待烧录过程结束。烧写成功后会提示如下图“download dp1000 -> ok”字样显示，烧写失败会提示 fail 字样。

```
-----  
---USB_DOWNLOAD_TOOL V1.3---  
-----  
[0x1400002]: ..... download dp1000 -> OK
```

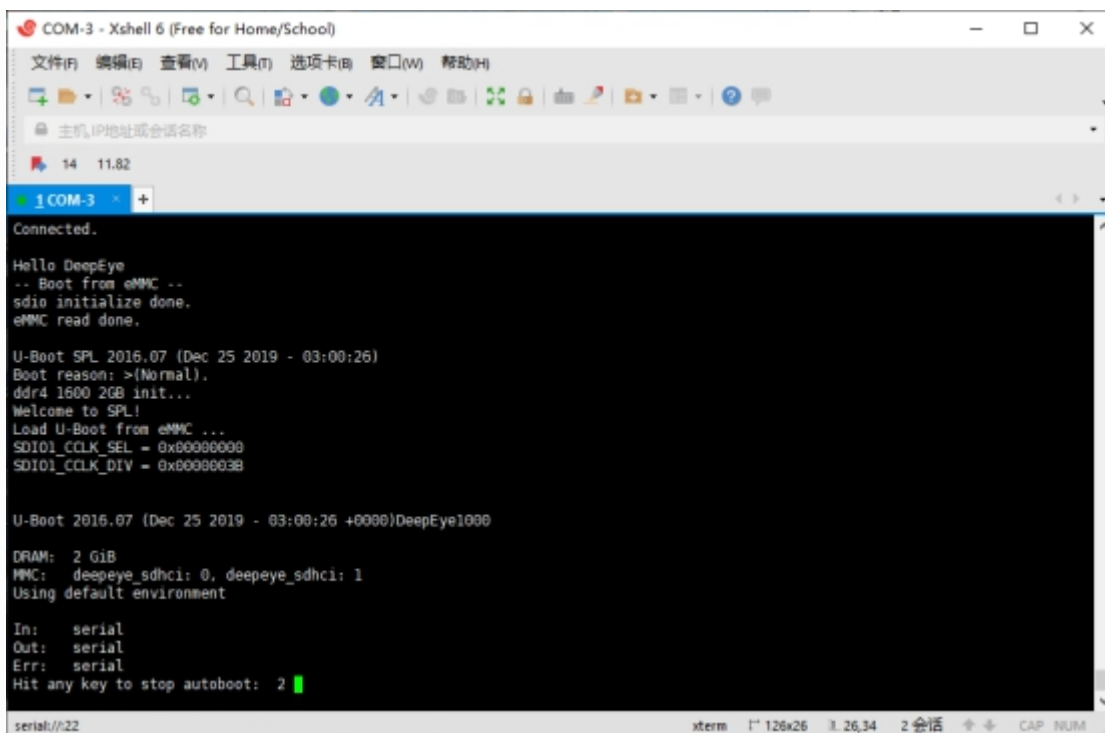
4.1.3 切换回工作模式

当烧录完成后，需要将拨码开关切换至EMMC启动模式，切换方法如下：

将拨码开关拨到100状态，如图所示：



按Reset键重启开发板，此时串口一开始输出如下：



COM-3 - Xshell 6 (Free for Home/School)

文件(F) 编辑(E) 查看(V) 工具(T) 选项卡(T) 窗口(W) 帮助(H)

主机, IP地址或会话名称

14 11.82

1 COM-3

Connected.

```
Hello DeepEye
-- Boot from eMMC --
sdio initialize done.
eMMC read done.

U-Boot SPL 2016.07 (Dec 25 2019 - 03:00:26)
Boot reason: >(Normal).
ddr4 1600 2GB init...
Welcome to SPL!
Load U-Boot from eMMC ...
SDIO1_CLK_SEL = 0x00000000
SDIO1_CLK_DIV = 0x0000003B

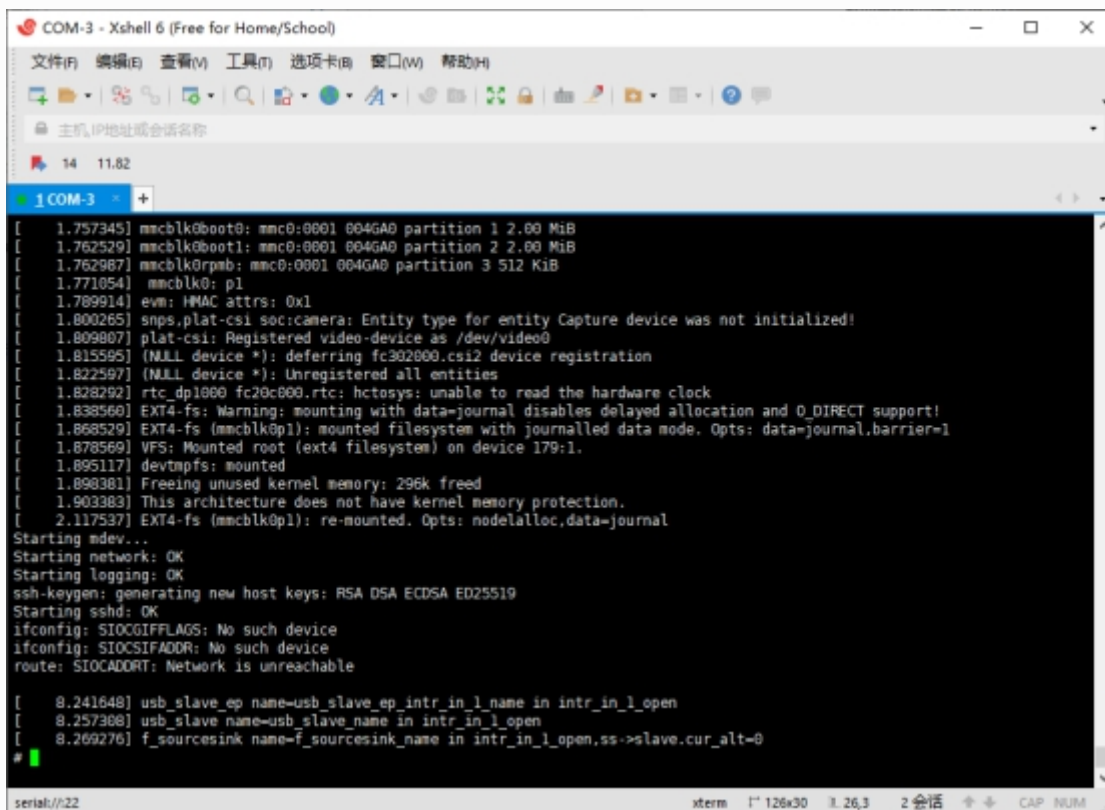
U-Boot 2016.07 (Dec 25 2019 - 03:00:26 +0000)DeepEye1000

DRAM: 2 GiB
MMC:  deepeye_sdhci: 0, deepeye_sdhci: 1
Using default environment

In:  serial
Out: serial
Err: serial
Hit any key to stop autoboot:  2
```

serial://22 xterm 126x26 2,26,34 2 会话 CAP NUM

启动完成时输出如下：



COM-3 - Xshell 6 (Free for Home/School)

文件(F) 编辑(E) 查看(V) 工具(T) 选项卡(T) 窗口(W) 帮助(H)

主机, IP地址或会话名称

14 11.82

1 COM-3

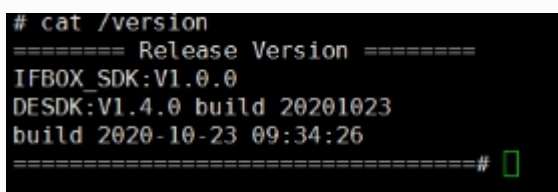
```
[ 1.757345] mmcblk0boot0: mmc0:0001 004GA0 partition 1 2.00 MiB
[ 1.762520] mmcblk0boot1: mmc0:0001 004GA0 partition 2 2.00 MiB
[ 1.762987] mmcblk0rpmb: mmc0:0001 004GA0 partition 3 512 KiB
[ 1.771054] mmcblk0: p1
[ 1.789914] evm: HMAC attrs: 0x1
[ 1.800265] snps,plat-csi soc:camera: Entity type for entity Capture device was not initialized!
[ 1.809807] plat-csi: Registered video-device as /dev/video0
[ 1.815595] (NULL device *): deferring fc302000.csi2 device registration
[ 1.822597] (NULL device *): Unregistered all entities
[ 1.828292] rtc_dp1000 fc20c000.rtc: hctosys: unable to read the hardware clock
[ 1.838560] EXT4-fs: Warning: mounting with data=journal disables delayed allocation and O_DIRECT support!
[ 1.868529] EXT4-fs (mmcblk0p1): mounted filesystem with journaled data mode. Opts: data=journal,barrier=1
[ 1.878560] VFS: Mounted root (ext4 filesystem) on device 179:1.
[ 1.895117] devtmpfs: mounted
[ 1.898381] Freeing unused kernel memory: 296k freed
[ 1.903383] This architecture does not have kernel memory protection.
[ 2.117537] EXT4-fs (mmcblk0p1): re-mounted. Opts: nodevalloc,data=journal
Starting mdev...
Starting network: OK
Starting logging: OK
ssh-keygen: generating new host keys: RSA DSA ECDSA ED25519
Starting sshd: OK
ifconfig: SIOCGIFFLAGS: No such device
ifconfig: SIOCSIFFADDR: No such device
route: SIOCADDRT: Network is unreachable

[ 8.241648] usb_slave_ep name=usb_slave_ep_intr_in_1_name in intr_in_1_open
[ 8.257368] usb_slave_name=usb_slave_name in intr_in_1_open
[ 8.269276] f_sourcesink name=f_sourcesink_name in intr_in_1_open,ss->slave.cur_alt=0
#
```

serial://22 xterm 126x30 2,26,3 2 会话 CAP NUM

4.1.4 版本验证

芯片侧串口输入“cat /version”可以看到芯片的版本号。如下图所示：



```
# cat /version
===== Release Version =====
IFBOX_SDK:V1.0.0
DESDK:V1.4.0 build 20201023
build 2020-10-23 09:34:26
=====#
```


5 USB和NET模式切换

5.1 版本要求

desdk v1.4.0及以上版本。

5.2 连接要求

dp1000上USB线和网线不能同时连接。

以USB模式启动时，需要连接USB线，拔掉网线。

以NET模式启动时，需要连接网线，拔掉USB线。

默认情况下，dp1000以USB模式启动。

5.3 切换为NET模式

1. 可选择直接在dp1000端或者从Host端发起切换。

- 。如果在dp1000端切换，运行命令：

```
# switch_usb.sh net
```

- 。如果从Host端切换，进入/DEngine/tools目录执行：

```
# sh switch_usb.sh net
```

2. 拔掉USB线，连接网线，重启dp1000。
3. Desdk RPC Server启动时，打印如下，切换成功：

```
2658399: Set dsp log level [4, 4]
2698851: Stop kernel log.
2700064: Start RPC server using NET, ip: 192.168.33.136, port: 9200
[ 88.218149] random: crng init done
```

表示RPC Server启动时使用的IP地址为192.168.33.136，端口为9200

4. 主控侧TARGET_CH需要修改为net

```
# export TARGET_CH=net
```

5. 修改主控侧/DEngine/net.cfg文件中的url为开发板的实际地址，示例中均使用这个文件配置IP，也可在初始化代码中手动指定。

5.4 切换为USB模式

1. 可选择直接在dp1000端或者从Host端发起切换。

- 。如果在dp1000上端切换，运行命令：

```
# switch_usb.sh usb
```

- 。如果从Host端切换，以只有一个设备为例，首先编辑主控端平台bin目录下的startup.cfg文件，修改Device 0的IP和端口与设备启动RPC Server时使用的IP和端口一致，如下图所示：

```
{  
    "device_cfg" : [  
        {  
            "dev_id" : 0,  
            "url" : "192.168.33.136",  
            "port" : 9200  
        },  
    ],  
}
```

然后进入/DEngine/tools目录执行：

```
# sh switch_usb.sh usb
```

2. 拔掉网线，连接USB线，重启dp1000
3. Desdk RPC Server启动时，打印如下，切换成功：

```
2687367: Set dsp log level [4, 4]  
2727798: Stop kernel log.  
2728206: Start RPC server using USB.  
dsp1: SetDspLogLevel: dsp 1 level 2 ok.
```

4. 主控侧TARGET_CH需要修改为空（代表默认usb）

```
# export TARGET_CH=
```

5.5 NET模式下IP

5.5.1 自动获取IP

通过串口设置dp1000上的/root/config/desdk.cfg文件中connect.ip_mode字段值为auto。

```
"connect.ip_mode": "auto"
```

dp1000启动后，会打印出自动获取到的IP地址，并使用该地址启动Desdk RPC Server。

注：自动获取的IP在重启后可能发生变化。

5.5.2 手动指定IP

通过串口设置dp1000上的/root/config/desdk.cfg文件中connect.ip_mode字段为manual，并设置connect.ip、connect.netmask、connect.gateway三个字段。例如：

```
"connect.ip": "192.168.33.241",  
"connect.netmask": "255.255.255.0",  
"connect.gateway": "192.168.33.1",  
"connect.port": "9200"
```

dp1000启动后，会用指定的IP、掩码、网关设置系统网络，并使用该IP地址启动Desdk RPC Server。

5.6 无串口切换NET模式

如果dp1000侧没有串口，只能在主控侧进行配置。另外，因为看不到启动过程中的打印，也不能使用 auto IP模式，否则Desdk RPC Server启动后，也无法获知dp1000所使用的IP地址。只能事先选择一个空闲的IP，在主机侧使用工具配置到dp1000上。

dp1000侧初始以USB连接模式启动。手动配置dp1000的IP地址步骤如下：

1. 修改desdk.cfg中的配置

在Host侧，准备一个dp1000上使用的desdk.cfg文件。修改connect.ip_mode字段为manual，并设置connect.ip、connect.netmask、connect.gateway三个字段。

2. 上传desdk.cfg到dp1000上指定目录，在Host侧执行命令：

```
# sh tools/transfer.sh -i 0 -u config/desdk.cfg /root/config/desdk.cfg
```

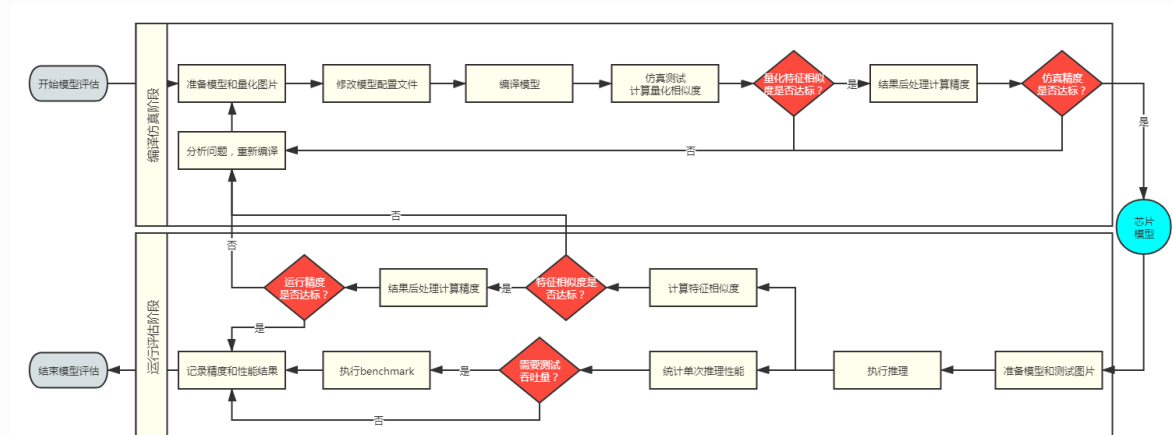
3. 配置dp1000以net连接模式启动，在Host侧执行命令：

```
# /DEngine/tools/switch_usb.sh net
```

4. dp1000上拔掉USB线，连接网线，重启

6 模型评估

用户需要对模型在DeepEye芯片上运行的精度和性能进行评估，以确定是否能满足业务需求。典型的模型评估过程如图所示：



其中，对于一些以特征为主要结果的模型，特征相似度已经能很好的表达模型的精度，也可以不需要进行后处理和精度评估。对于一些特征相似度无法正确表达模型精度的模型，如含有择优、排序和特殊计算等导致特征相似度差异较大的模型，必须要使用后处理和精度来进行评估。

注：开发板上的模型评估建议使用NET模式，避免由于使用虚拟机等USB不太稳定的因素导致评估问题。切换NET模式的方法参见USB和NET模式切换章节。

6.1 模型编译仿真

模型编译仿真只能在ubuntu x64机器的docker中进行，具体操作参见《云天励飞DETVM工具链快速上手手册》。

6.2 模型板上评估

模型板上评估有单次评估、benchmark评估2种方式，用户可自行选择需要的方式。

6.2.1 单次评估

单次评估指从主控侧通过python接口单次调用模型推理，获得模型各部分运行的时间以及运行结果的方法。

操作步骤：

1. 将编译好的芯片模型文件和ini配置文件、model_info.json文件拷贝到运行平台的对应文件夹下，配置文件中的[run]分录下配置正确的芯片模型目录netbin_folder_path和测试图片路径img_path，示例中芯片模型目录为/DEngine/model/dp1000/caffe_squeezenet_v1.1。
2. 进入模型目录/DEngine/model/dp1000/caffe_squeezenet_v1.1，执行：

```
# /DEngine/run.sh caffe_squeezenet_v1.1.ini
```

运行结果如下：

```
-----Info-----
Name                Time(millisecond)    Percent
time_total          231.0187530517578
time_nnp             188.8653358282445    81.75%
time_dsp             35.96525047510097    15.57%
time_crgh            2.7103333691368383    1.17%
time_sched           3.4778333792755136    1.51%
time_kcf             0.0                  0.00%

batch               1
nnp_GOPs            116.76555
nnp_use_ratio        184.00%
fps                 4.328652920120119

Testing loop 0 over.
MODEL_RUN_SUCCESS
```

其中，time_total是芯片上模型的单次运行时间。

注：如果不是自己编译而是直接下载模型，可能没有参考数据tvm_fix_out.bin，导致特征值相似度similarity可能为0。如果需要关注需要自行编译模型并使用相同的输入进行编译和测试。

6.2.2 benchmark评估

benchmark评估指对模型在芯片上的精度和推理性能进行测试和统计，使用户对DeepEye芯片当前配置下的推理能力有定量的了解，帮助用户根据需要进行合理的配置参数。benchmark工具的具体介绍参见《云天励飞DESDK Benchmark工具使用手册》。

示例操作步骤：

1. 模型的默认编译配置为nnp_number = 4，为低延迟模式。跑benchmark时一般使用吞吐量模式，需要将caffe_squeezenet_v1.1.ini中的nnp_number改为1，重新编译。将编译好的芯片模型文件和ini配置文件、model_info.json文件拷贝到运行平台的对应文件夹下，示例中和ini和json文件与原始模型放在同一文件夹下，编译好的模型放在下一级caffe_squeezenet_v1.1目录中。

```
# 指定nnp数
nnp_number = 1
```

2. 准备编译好的芯片模型文件和对应的benchmark.json配置文件，示例中已配置好benchmark.json文件内容如下：

```
{
  "device_num":1,
  "test_suites":
  [{
    "test_suite_name":"normal",
    "accuracy_datasets_path":"/DEngine/data/datasets/ILSVRC2012",
    "synset_path":"/DEngine/data/datasets/synset_1000.txt",
    "val_path":"/DEngine/data/datasets/ILSVRC2012_val.txt",
    "fps_datasets_path":"/DEngine/data/datasets/ILSVRC2012",
    "models_root":"/DEngine/model/dp1000",
    "thread_list":[1],
    "batch_list":[1,2,4],
    "nnp_num_list":[1,2,4],
    "resource_num_list":[4,4,4],
    "topn":5,
    "time_criteria": 10.0,
    "num_criteria": 50,
    "accuracy_test": false,
    "fps_test": true,
    "input_quesize": 32,
    "test_models":
    {
      "caffe_squeezenet_v1.1": {"do_test":true, "shape":[227,227],
"rgb":true,  "resource_num_list":[8,8,8]}
    }
  ]
}
```

3. 进入desdk/tools/benchmark目录，执行：

```
# /DEngine/run.sh benchmark.py
```

4. benchmark运行结果如图所示



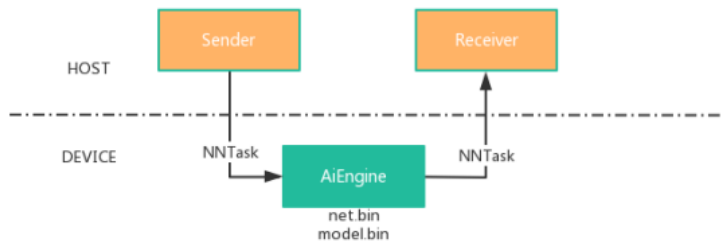
模型名称	测试集	精度	线程数	Batch	RRP	RES	芯片时延(ms)	芯片吞吐(Fps)	RRP利用率	主控时延(ms)	主控吞吐(Fps)	RRP1	RRP2	RRP3	RRP4	RRP5
caffe_squeezenet_v1.1	ILSVRC2012	1.732870	1	1	1	8	205.8898939	98.87284861	0.385777593	995.744	98.87284861	0.56591175	0	0	0	1.486175206
			2	2	2	8	213.1560989	197.4339449	0.381773108	230.464	169.8358451	0.39345644	0	0	0	2.936765812
			4	4	4	8	205.9299927	315.4678938	0.304840863	281.042	169.7881533	0.497430938	0.501430993	0.443261378	0.446276389	5.877650804

7 应用开发

本章节主要介绍如何快速使用desdk开发应用程序，详细介绍请参考《云天励飞DESDK应用开发指南》。

由于DeepEye1000是协处理器，为提升DeepEye1000与主控间的传输效率，充分利用芯片资源，desdk提供了基于Graph的流编程框架。每个操作节点定义为1个Node（也称Graph算子），Node之间通过pin相连，构成一个Graph。待处理的数据从整个Graph的入口pin输入，从出口pin获得结果。

本章以一个简单的模型推理应用为例，Graph如下图所示：



图示绿色模块为desdk标准算子，橙色模块为自定义算子或基于标准算子的扩展算子。

1. Sender算子构造模型输入发送到标准推理算子。
2. Receiver算子接收推理结果。

7.1 代码编译

1. 进入/DEngine/desdk/cpp目录
2. 如果需要交叉编译需要将编译链放到/opt目录下，并修改CMakeList中的配置
3. 执行 ./build.sh 生成 model_pred 可执行程序（安装到/DEngine/desdk/cpp/example/model_pred/host/bin目录）
4. 需要生成芯片侧库的应用执行./build_device.sh（本例不需要）

7.2 程序运行

1. 查看 /DEngine/model/dp1000/caffe_squeezenet_v1.1 目录是否存在，是否包含 net.bin 和 model.bin。
2. 进入/DEngine/desdk/cpp/example/model_pred/host/bin
3. 执行/DEngine/run.sh ./model_pred
4. 观察打印，主控将持续打印100条带有"<====AiEngine result"字样的推理结果。

8 FAQ

8.1 烧录或下载过程中出现USB初始化失败

问题：如下图所示，使用root权限运行出现USB初始化失败

解决方法：确保USB硬件连接正常，并按以下流程每执行一步都尝试一下是否能成功

- ```
rmmod usbtest
find /lib/modules -name usbtest.ko | sudo xargs rm
```

2. 重启芯片开发板，解决芯片侧可能异常问题
3. 如果在docker内退出docker重新进入，如果不在docker内kill掉之前未退出的进程，解决主控侧可能未完全退出的进程占用usb口问题

问题：使用Windows+虚拟机VMWare的方式时，有时出现USB设备未找到导致的错误，可能是USB未连接到虚拟机导致

检测到新的 USB 设备

选择您希望将 DeepEye 连接到的位置

☐ 连接到主机

☒ 连接到虚拟机

虚拟机名称

Ubuntu16.04.06

☒ 记住我的选择，以后不再询问

确定 取消

## 8.3 连接时报错 Server Bind Port : Address already in use

问题：常见于模型评估时，连接芯片打印Try to connect server...后出现Server Bind Port : Address already in use，原因是上一次连接没有正常结束，端口被占用

解决方法：kill掉执行进程后重试。如果是用model\_convert工具或者python直接推理的，先使用top查看活动的python3进程id号，在使用kill -9 id号杀死进程。如果无法确认是哪个进程，在docker内的话可以推出docker重进，不在docker内的话可以重启主控。