



Amlogic

nnEngine 工具介绍


Revision: 0.1

Release Date: 2022-02-10

Copyright

© 2022 Amlogic. All rights reserved. No part of this document may be reproduced, transmitted, transcribed, or translated into any language in any form or by any means without the written permission of Amlogic.

Trademarks

 , and other Amlogic icons are trademarks of Amlogic companies. All other trademarks and registered trademarks are property of their respective holders.

Disclaimer

Amlogic may make improvements and/or changes in this document or in the product described in this document at any time.

This product is not intended for use in medical, life saving, or life sustaining applications.

Circuit diagrams and other information relating to products of Amlogic are included as a means of illustrating typical applications. Consequently, complete information sufficient for production design is not necessarily given. Amlogic makes no representations or warranties with respect to the accuracy or completeness of the contents presented in this document.

Contact Information

- Website: www.amlogic.com
- Pre-sales consultation: contact@amlogic.com
- Technical support: support@amlogic.com

更改记录

版本 0.1 (2022-02-10)

第一版。

Confidential!
nick@khadas.com
2022-03-17 19:26:35

目录

更改记录.....	ii
1. nnEngine 简介.....	1
2. nnEngine 使用方式介绍	2
2.1 导入 nn_engine python 库	2
2.2 nn_engine API 接口介绍.....	2
2.2.1 amlnn = AMLNN().....	2
2.2.2 amlnn.init_runtime().....	2
2.2.3 amlnn.set_input()	3
2.2.4 amlnn.inference().....	3
2.2.5 amlnn.destroy().....	3
3. Demo 运行方法介绍	4

1. nnEngine 简介

nnEngine 是 Amlogic 开发的一套基于 nnsdk 开发包的模型推理引擎，利用 ADB 协议实现嵌入式设备端和服务器端的数据传输和模型运行控制。用户可通过提供的 aml_nn_engine python 库实现模型推理和性能指标获取功能。

https://github.com/Amlogic-NN/AML_NN_SDK/tree/master/Tools/nnengine

关于 nnsdk 开发包的相关介绍请见《DDK_6.4.8.7_SDK_V1.9.4 API》。

Confidential!
nick@khadas.com
2022-03-17 19:26:35

2. nnEngine 使用方式介绍

2.1 导入 nn_engine python 库

```
import sys
sys.path.append('aml_nn_engine_37') #指定 aml_nn_engine 库文件路径
```

注：需选择对应 python 版本的 aml_nn_engine 库文件，若版本不匹配可能出现导入失败的问题。

2.2 nn_engine API 接口介绍

1. 初始化 amlnn 类

```
amlnn = AMLNN()
```

2. 模型初始化

```
amlnn.init_runtime(config.linux_path, config.nb_path, config.model_type)
```

3. 设置模型输入

```
amlnn.set_input(config.input_name, config.input_data, config.input_type, config.in  
put_w, config.input_h, config.input_c, config.run_cyc, config.modetype, config.profile)
```

4. 模型前向处理并获取输出结果

```
output=amlnn.inference()
```

5. 模型卸载

```
amlnn.destroy()
```

2.2.1 amlnn = AMLNN()

API	amlnn = AMLNN()
功能	初始化 AMLNN 类

2.2.2 amlnn.init_runtime()

API	amlnn.init_runtime(config.linux_path, config.nb_path, config.model_type, config.device_type)
功能	设置模型输入及相关控制参数
参数	config.linux_path: 设置设备工作路径
参数	config.nb_path: 模型 nbg 文件的路径及名称
参数	config.model_type: 设置模型框架类型，目前已支持 caffe, tensorflow, tensorflowlite, darknet, onnx, keras, pytorch
参数	config.device_type: 指定设备操作系统类型，分为 linux 和 android

示例：

```
amlnn.init_runtime("/media/nn/", mobilenetv1_be.nb, tensorflow, linux)
```

2.2.3 amlnn.set_input()

API	amlnn.set_input(config.input_name,config.input_data,config.input_type,config.input_w,config.input_h,config.input_c,config.run_cyc,config.modetype,config.profile)
功能	修改输出物理地址对应 DMA buffer 的起始指针，实现 output 的快速切换
参数	config.input_name: 输入文件路径及名称，支持 jpeg,tensor 和 binary 格式文件
	config.input_data: 输入原始数据，例如利用 opencv 读取 bmp 图像的数据
	config.input_type: 数据前处理类型，rgb 表示采用 RGB24 的模型预处理操作；tensor 表示采用 tensor 的模型预处理操作；raw 表示不进行数据预处理，直接将输入作为模型的输入。
	config.input_w: 输入数据宽度信息
	config.input_h: 输入数据高度信息
	config.input_c: 输入数据通道信息
	config.run_cyc: 模型 run_graph 循环次数，通常用于测试模型的运行时间
	config.modetype: 用户自有模型请选择 99,将模型输出数据直接返回，用户可在 python 端进行后处理操作
	config.profile: 获取 profile 信息，可通过设置成'all'获取 fps 和带宽信息，设置 'runtime'获取模型 fps 信息，设置'bw'获取模型带宽信息

注：

Set_input 支持 jpeg, tensor, binary 格式文件输入，此时 config.input_name: 设置为对应的文件名称，并将 config.input_data 设置为 None，config.input_type 设置为指定的格式码，并设置好输入数据的宽度/高度/通道信息。

Set_input 支持输入各种格式图像解析后的数据，此时 config.input_name: 设置为 None，并将 config.input_data 设置为输入数据。用户可自行进行数据前处理并将输入类型设置为 rawdata，若用户希望利用 nnsdk 的预处理方案可将输入图像解析为 RGB24 或 tensor 格式的数据，将输入类型设置为 RGB24 或 tensor。

此版本 android 平台暂不支持获取模型带宽信息，将在下一版本中添加该功能。

示例：

```
amlnn.set_input('2242243.jpg',None,'rgb',224,224,3,1,0,all)
```

2.2.4 amlnn.inference()

API	output=amlnn.inference(out_size)
功能	获取模型推理结果
参数	out_size: sockect 传输数据尺寸信息
返回值	output: 模型输出结果

output 为模型经过 npu 推理得到的运算结果，可基于 python 端添加后处理代码验证模型精度是否正常。out_size 参数默认为 4096，若输出 size 大于 4096 需指定具体的数据。如果是多输出模型，需要设置输出 size 的总和。

2.2.5 amlnn.destroy()

API	amlnn.destroy()
功能	卸载模型，释放相关资源

3. Demo 运行方法介绍

当前 nnEngine 支持运行在 windows10 操作系统和 python3.7 环境下。

windows 环境配置：

1. PC 端和开发板通过 USB 接口连接，并确保 ADB 可正常识别到指定设备。

shell 中执行命令: adb devices -l, 可正常显示目标设备; 执行命令: adb shell, 并在使用 cmd 或 powershell 执行 adb shell 确保 adb 功能正常

2. 配置 python 环境,

```
pip3 install numpy==1.20.1
pip3 install opencv-python==4.5.4.58
pip3 install easydict==1.9
```

- a) windows 平台环境准备

windows 环境下使用 cmd 或 powershell 进入 demo 存放路径下，执行 python demo.py 即可在 shell 中获取分类模型的最终结果。

执行操作前需要将在板端创建 '/media/nn' 文件夹，将 nnEngine 可执行文件放置在该路径下。

- b) Android 平台准备

执行操作前在 shell 中为 android 添加对应权限，然后在板端创建 '/media/nn' 文件夹，将 nnEngine 可执行文件放置在该路径下。

```
adb root
adb remount
adb shell
setenforce 0
```

- c) 启动 NNEngine 服务（二选一即可）

方法一：板端执行: ./nnEngine

方法二：PC 端执行: python .server.py

- d) 运行测试 demo

PC 端执行: python .demo.py 即可得到输出结果

物体分类 demo

```
output=amlnn.inference(4096)
for j in range(10+4,10+4+5):
    print("preceived top",j-4," is:",struct.unpack('f', struct.pack('4B', output[4*j], output[4*j+1],
    output[4*j+2],output[4*j+3]))[0])
for i in range(4,7):
    print("profile",i-4,"is:",struct.unpack('f',struct.pack('4B', output[4*i], output[4*i+1], output[4*i+2],
    output[4*i+3]))[0])
```

物体检测 demo


```
output=aml.nn.inference(425*13*13)    #output size is 425*13*13=71825
list_out=list(output[(4+10)*4:(71825+10+4)*4])#10:存放 profile 相关数据,4:socket 信息头
lista = []
out_lenth = (int)((len(list_out) )/4)
for i in range(out_lenth):
    lista.append(struct.unpack('f',struct.pack('4B',list_out[4*i],list_out[4*i+1],list_out[4*i+2],list_out[4*i+3]))[0])
```

Confidential!
nick@khadas.com
2022-03-17 19:26:35