

RK1808 人工智能计算棒使用说明

V1.4.1

toybrick Confidential

目 录

1	简介	3
2	技术规格	4
2.1	最小系统要求	4
3	资料指南	5
4	快速入门指南	6
4.1	环境准备	6
4.2	RKNN-TOOLKIT 安装	6
4.3	设备访问权限修改	8
4.4	运行示例	9
5	WEB 配置介绍	12
5.1	宿主机网络配置	12
5.2	WEB 登录	18
5.3	WEB 页面主要功能介绍	19
6	被动模式开发流程	25
7	主动模式开发	27
7.1	主动模式下的 AI 开发过程描述和介绍	27
7.2	产品部署过程描述和介绍	28
7.3	开发工具 TOYBRICK_DEPLOYC	28

1 简介

RK1808 人工智能计算棒是瑞芯微旗下 Toybrick 系列产品中的一员，计算棒搭载瑞芯微的 RK1808 神经网络处理器，具有低功耗、高性能的特点，可应用于人工智能的各个应用领域。上位机通过 RK1808 人工智能计算棒，即可获得强大的深度学习推理能力。借助 RK1808 人工智能计算棒的强大算力，嵌入式设备可在网络边缘侧构建人工智能算法，使得传统嵌入式设备轻松完成人工智能升级。

RK1808 人工智能计算棒可用于辅助推理计算，也支持通过二次开发完成独立人工智能计算功能。

2 技术规格

技术规格	
CPU	RK1808
内存	1GB LPDDR
存储	8GB EMMC
尺寸	82x31x13mm
接口	USB3.0 Type-A
温度	0°C~40°C

2.1 最小系统要求

- Ubuntu 16.04 或 Windows 7 的 x86 的 64 位计算机
- CPU intel 酷睿 i3
- USB 3.0
- 2 GB RAM
- 4 GB 可用存储空间

3 资料指南

插入 RK1808 人工智能计算棒，PC 上会显示 U 盘设备，U 盘设备的目录结构及文档说明如下：

目录	文档名称	说明
doc	Rockchip_RK1808_AI_Compute_Stick_User_manual_EN.pdf	RK1808 人工智能计算棒使用说明指南
	Rockchip_RK1808_AI_Compute_Stick_User_manual_CN.pdf	
	Rockchip_RK1808_AI_Compute_Stick_Easy_Start_Demo_Guide_EN.pdf	一键运行 yolov3 demo 指南
	Rockchip_RK1808_AI_Compute_Stick_Easy_Start_Demo_Guide_CN.pdf	
driver	ntb	USB ntb windows 驱动
example	run_demo.bat	一键运行 yolov3 示例 Windows 的批处理程序
	run_demo.sh	一键运行 yolov3 示例的脚本
tool		脚本和工具

更多资料请登陆官方 [wiki](http://t.rock-chips.com/wiki.php?mod=view&pid=28) 查看：<http://t.rock-chips.com/wiki.php?mod=view&pid=28>

4 快速入门指南

本章节主要描述 RK1808 人工智能计算棒如何在 Ubuntu18.04 的 PC 上，基于 python3.6 使用 RKNN-Toolkit 快速运行深度神经网络模型 mobilenet_v1 的例子。

4.1 环境准备

- 装有 ubuntu 18.04 操作系统的 intel 酷睿 i3 以上的 x86 的 64 位 PC。
- RK1808 人工智能计算棒。
- 将 RK1808 人工智能计算棒插在 PC 的 USB 接口上，使用 `lsusb` 命令查看结果。如下（其中红色字体部分 2207:0018 即为我们的 RK1808 人工智能计算棒）：

1) 输入命令如下：

```
lsusb
```

2) 执行结果如下：

```
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
```

```
Bus 003 Device 009: ID 2207:0018
```

4.2 RKNN-Toolkit 安装

RKNN-Toolkit 是瑞芯微为用户提供的基于 python 接口编程的模型转换、推理和性能评估的 NPU 开发套件，本节以装有 ubuntu 18.04 操作系统的 x86 的 64 位 PC 为例，说明当 Python 版本为 3.6 时如何安装 RKNN-Toolkit，以下是在终端下执行的命令：

- 安装 Python3.6

```
sudo apt-get install python3.6
```

- 安装 opencv

```
sudo apt-get install -y python3-opencv
```

- 安装 pip3

```
sudo apt-get install python3-pip
```

- 插入 RK1808 人工智能计算棒，PC 上会显示 U 盘设备，U 盘设备的根目录结构如下：



- 执行以下步骤：

1. 在当前用户目录创建 rknn 目录， 并进入 rknn 目录：

```
mkdir ~/rknn  
cd ~/rknn
```

2. 安装 wget：

```
sudo apt-get install -y wget
```

3. 从官方 FTP 文件服务器下载 RKNN-Toolkit

```
wget http://repo.rock-chips.com/python/rknn_toolkit-1.2.1-cp36-cp36m-  
linux_x86_64.whl
```

4. 安装 Python 依赖，tensorflow 最低版本要求 1.11.0，这里以安装 1.14.0 为例：

```
pip3 install --user tensorflow==1.14.0
```

5. 安装 RKNN-Toolkit：

```
pip3 install --user rknn_toolkit-1.2.1-cp36-cp36m-linux_x86_64.whl
```

注：RKNN-Toolkit 包必须与 python 版本一致， ubuntu18.04 缺省默认是 python3.6。

6. 检查 RKNN-Toolkit 是否安装成功。

- 1) 在终端下输入以下命令：

```
python3
```

- 2) 在 python3 运行环境下输入以下代码， 导入 RKNN 模块：

```
from rknn.api import RKNN
```

(1) RKNN 模块导入成功的情况如下：

```
$ python3
>>> from rknn.api import RKNN
>>>
```

(2) 如果导入 RKNN 模块没有报错，然后输入 `quit()` 退出 Python3（跳过以下第（3）步骤）。

(3) 如果导入 RKNN 模块报以下错误，这是由于 RKNN 依赖的 Tensorflow 的安装包使用的是 SSE4.2 指令集，这些指令无法在旧版的 CPU 上运行，请检查并更换一台支持 SSE4.2 指令集的 PC。

```
$ python3
>>> from rknn.api import RKNN
2019-06-25 20:10:25.255397: F
tensorflow/core/platform/cpu_feature_guard.cc:37] The TensorFlow library was
compiled to use SSE4.2 instructions, but these aren't available on your
machine.
```

4.3 设备访问权限修改

1. 插入 RK1808 人工智能计算棒，需要修改其 USB 设备访问权限，操作步骤如下：

1) 返回至 U 盘根目录，拷贝 `tool/update_rk1808_ai_cs_rule.sh` 到当前 HOME 目录的 `rknn` 目录下，修改 `update_rk1808_ai_cs_rule.sh` 为可执行权限。

```
cp tool/update_rk1808_ai_cs_rule.sh ~/rknn -f
chmod +x ~/rknn/update_rk1808_ai_cs_rule.sh
```

2) 返回至 `rknn` 目录，`update_rk1808_ai_cs_rule.sh` 必须以 root 权限执行。

```
cd ~/rknn
sudo ./update_rk1808_ai_cs_rule.sh
```

注：这一步只有在安装的时候需要执行一次，以后都不需要执行。

2. 执行完脚本后，查看 RK1808 人工智能计算棒的 USB 设备编号。

1) 输入以下命令:

```
lsusb
```

2) 执行结果如下:

```
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 003 Device 009: ID 2207:0018
```

注: 其中 Bus 003 Device 009 是计算棒的设备编号。

3. 根据计算棒的 USB 设备编号, 确认 RK1808 人工智能计算棒设备的读写权限

1) 输入以下命令:

```
ls -l /dev/bus/usb/003/009
```

注: 以上的 003/009, 不同的 PC 得到编号可能不一样, 视具体情况而定。

2) 执行结果如下 (正确的读写权限如红色字体部分所示):

```
crw-rw-rw- 1 root root 189, 264 6月 14 16:02 /dev/bus/usb/003/009
```

4.4 运行示例

这里以 mobilenet_v1 为例, mobilenet_v1 示例实现的功能是对一张图片进行特征提取, 并识别这张图片所属分类。

以下是 mobilenet_v1 示例的目录结构及说明如下:



- dataset.txt: 包含测试图片路径的文本文件。
- dog_224x224.jpg: 作为 mobilenet_v1 示例的测试图片。
- mobilenet_v1.tflite: TensorFlow Lite 模型文件。
- mobilenet_v1.rknn: rknn 模型文件。由非 rknn 模型文件 (这里是 TensorFlow

Lite 模型) 经过 RKNN-Toolkit 模型转换生成的该 rknn 模型文件。

- test.py: 示例运行脚本(包含 rknn 模型转换部分)。
- test_inference.py: 示例运行脚本(仅加载 rknn 模型进行推理)。

示例运行步骤如下:

1. 从官方 FTP 文件服务器下载 mobilenet_v1 压缩包, 并解压出来:

```
wget http://repo.rock-chips.com/rk1808/mobilenet_v1.tar.gz
tar xvf mobilenet_v1.tar.gz
```

2. 进入 mobilenet_v1 目录, 并执行 test.py 脚本:

```
cd mobilenet_v1/
python3 test.py
```

3. 得到如下结果:

```
--> config model
done
--> Loading model
done
--> Building model
done
--> Export RKNN model
done
--> Init runtime environment
done
--> Running model
mobilenet_v1
-----TOP 5-----
[156]: 0.8837890625
[155]: 0.0677490234375
[188 205]: 0.00867462158203125
[188 205]: 0.00867462158203125
[263]: 0.0057525634765625

done
--> Begin evaluate model performance
```

```
=====
                                Performance
=====
Total Time(us): 7140
FPS: 140.06
=====

done
```

4. 根据以上执行结果可知，TOP5 表示模型预测出来的排名前 5 的分类的结果，其中 [156] 表示狗的标签，0.8837890625 表示预测为该标签的概率，可以看出预测结果为狗的可能性最大，从测试图片 (dog_224x224.jpg) 可以看出预测结果是准确的。

5 WEB 配置介绍

RK1808 人工智能计算棒支持使用 web 界面进行系统配置，下面介绍如何访问这一配置界面，以及界面的主要功能。

5.1 宿主机网络配置

Windows 7/10 下网络配置：

1. 插入 RK1808 人工智能计算棒。
2. 打开设备管理器，网络适配器模块中，会出现“Remote NDIS based Internet Sharing Device”。

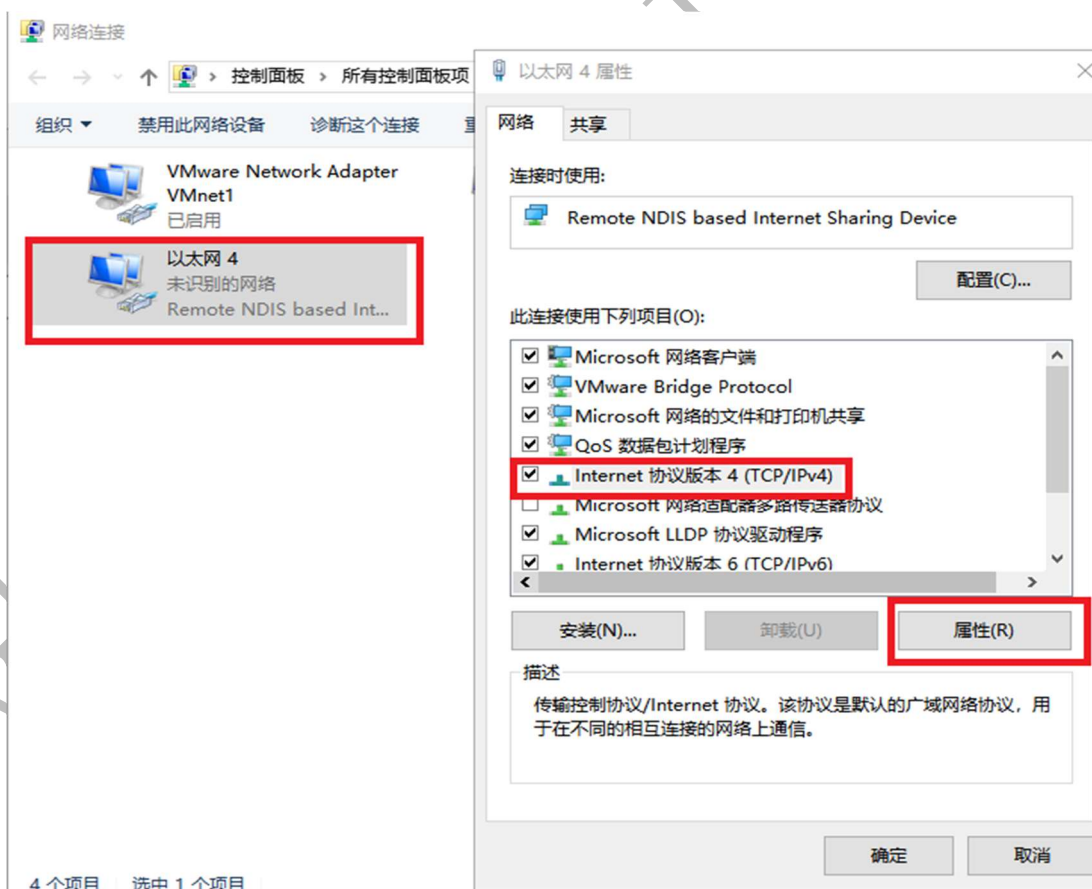


注：有些电脑会出现无法识别的设备，卸载无法识别的设备，等待 RNDIS 驱动重新安装。

3. 打开网络和 Internet 设置，点击“以太网”->“更改适配器选项”。

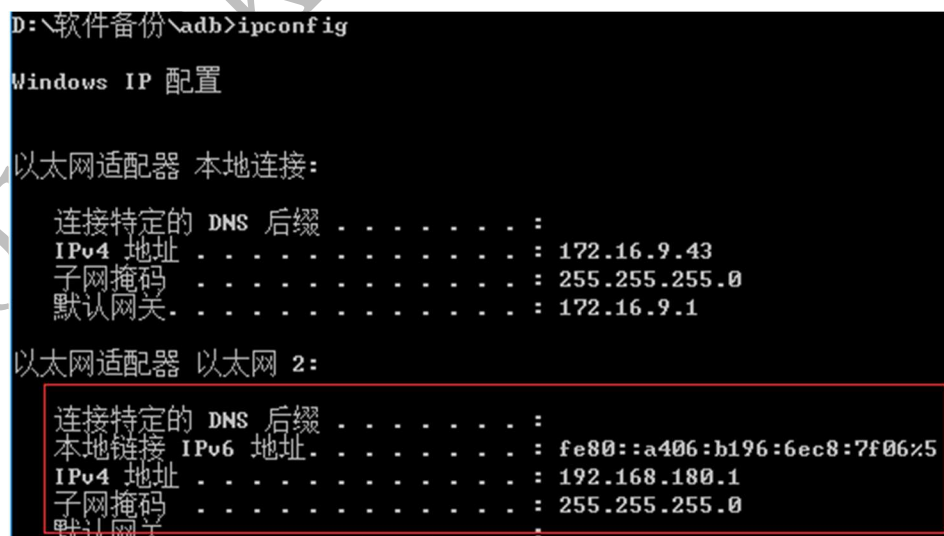


4. 右键“以太网*”（远程 NDIS 兼容设备），->“属性”->“Internet 协议版本 4”，->“属性”，选择“使用下面的 IP 地址”，输入 IP 地址和子网掩码，点击“确认”。





5. 确认 IP 地址。



Linux 下网络配置（以 ubuntu18.04 为例）：

1. 在插入 RK1808 人工智能计算棒之前，先在上位机打开终端，输入命令 ifconfig。

```
wuli@wuli-HP-ProDesk-680-G1-TWR:~$  
wuli@wuli-HP-ProDesk-680-G1-TWR:~$ ifconfig  
eno1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500  
    inet 172.16.9.120 netmask 255.255.255.0 broadcast 172.16.9.255  
    inet6 fe80::1e9:9d1e:37ea:8225 prefixlen 64 scopeid 0x20<link>  
    ether ec:b1:d7:53:8d:7d txqueuelen 1000 (以太网)  
    RX packets 410248 bytes 299235961 (299.2 MB)  
    RX errors 0 dropped 0 overruns 0 frame 0  
    TX packets 133223 bytes 12508587 (12.5 MB)  
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0  
    device interrupt 20 memory 0xf7c00000-f7c20000  
  
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536  
    inet 127.0.0.1 netmask 255.0.0.0  
    inet6 ::1 prefixlen 128 scopeid 0x10<host>  
    loop txqueuelen 1000 (本地环回)  
    RX packets 11570 bytes 915370 (915.3 KB)  
    RX errors 0 dropped 0 overruns 0 frame 0  
    TX packets 11570 bytes 915370 (915.3 KB)  
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0  
  
vmnet1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500  
    inet 192.168.29.1 netmask 255.255.255.0 broadcast 192.168.29.255  
    inet6 fe80::250:56ff:fec0:1 prefixlen 64 scopeid 0x20<link>  
    ether 00:50:56:c0:00:01 txqueuelen 1000 (以太网)  
    RX packets 0 bytes 0 (0.0 B)  
    RX errors 0 dropped 0 overruns 0 frame 0  
    TX packets 3563 bytes 0 (0.0 B)  
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0  
  
vmnet8: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500  
    inet 172.16.108.1 netmask 255.255.255.0 broadcast 172.16.108.255  
    inet6 fe80::250:56ff:fec0:8 prefixlen 64 scopeid 0x20<link>  
    ether 00:50:56:c0:00:08 txqueuelen 1000 (以太网)  
    RX packets 0 bytes 0 (0.0 B)  
    RX errors 0 dropped 0 overruns 0 frame 0  
    TX packets 3555 bytes 0 (0.0 B)  
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

2. 然后插入 RK1808 人工智能计算棒，再次在上位机输入命令 ifconfig，发现多了图中红框这一项。


```

muli@wuli-HP-ProDesk-680-G1-TWR:~$ ifconfig
eno1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 172.16.9.120 netmask 255.255.255.0 broadcast 172.16.9.255
    inet6 fe80::1e9:9d1e:37ea:8225 prefixlen 64 scopeid 0x20<link>
    ether ec:b1:d7:53:8d:7d txqueuelen 1000 (以太网)
    RX packets 413671 bytes 299653869 (299.6 MB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 133854 bytes 12591302 (12.5 MB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
    device interrupt 20 memory 0xf7c00000-f7c20000

enp0s20u11: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet6 fe80::c803:409f:4aac:24eb prefixlen 64 scopeid 0x20<link>
    ether de:00:f8:9a:b6:ca txqueuelen 1000 (以太网)
    RX packets 46 bytes 1576 (1.5 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 396 bytes 99557 (99.5 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

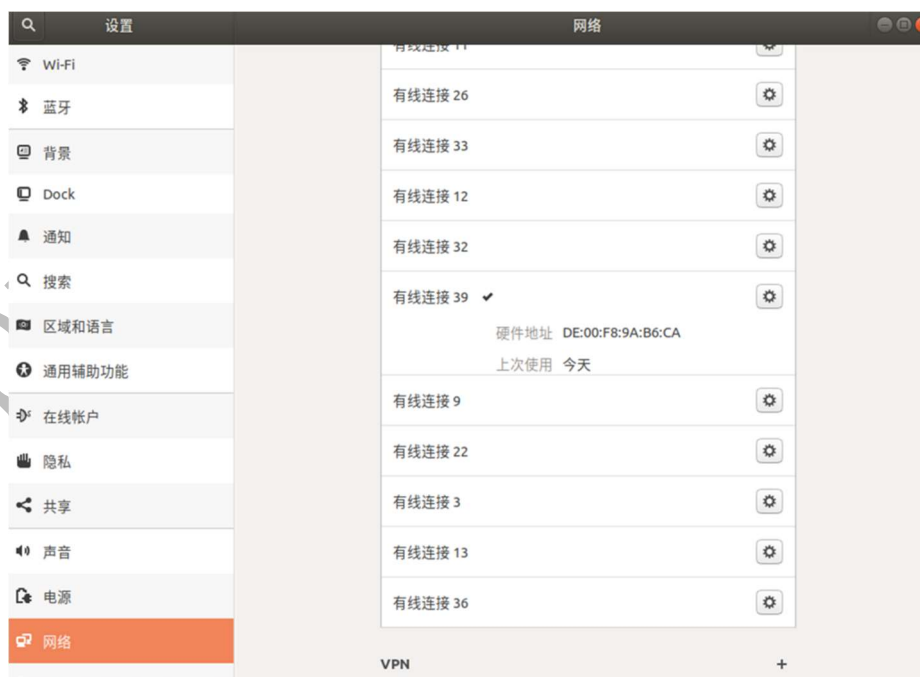
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (本地环回)
    RX packets 12126 bytes 965472 (965.4 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 12126 bytes 965472 (965.4 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

vmnet1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.29.1 netmask 255.255.255.0 broadcast 192.168.29.255
    inet6 fe80::250:56ff:fec0:1 prefixlen 64 scopeid 0x20<link>
    ether 00:50:56:c0:00:01 txqueuelen 1000 (以太网)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 3625 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

vmnet8: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 172.16.108.1 netmask 255.255.255.0 broadcast 172.16.108.255
    inet6 fe80::250:56ff:fec0:8 prefixlen 64 scopeid 0x20<link>
    ether 00:50:56:c0:00:08 txqueuelen 1000 (以太网)
    RX packets 0 bytes 0 (0.0 B)

```

3. 打开系统设置，选择网络，找到第二步中新添那一项 Mac 地址相同的设备，点击右上角设置。



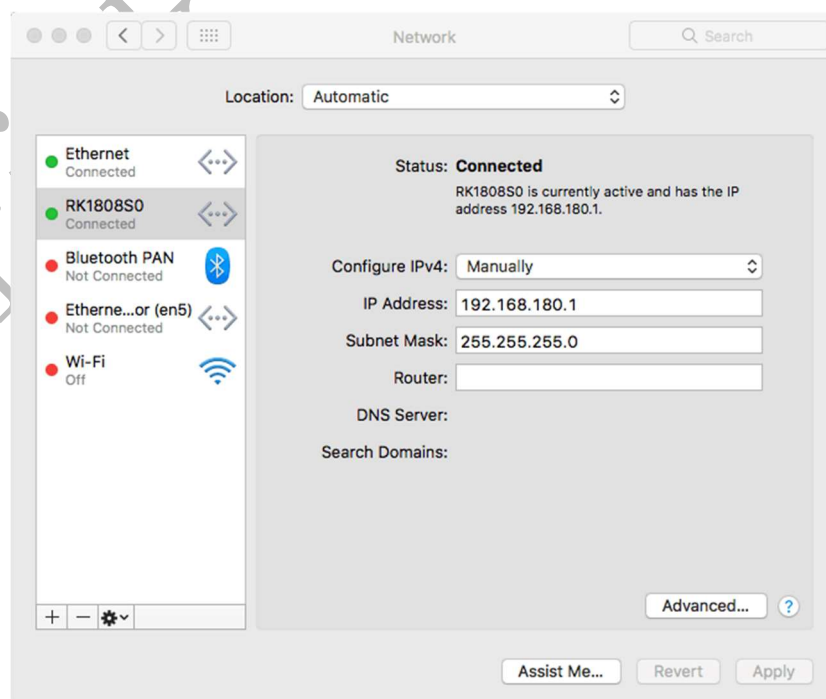
4. 按照下图，选择 ipv4, 手动，添加新的 ip -- 192.168.180.1 255.255.255.0, 点

击应用后，即完成网络配置。



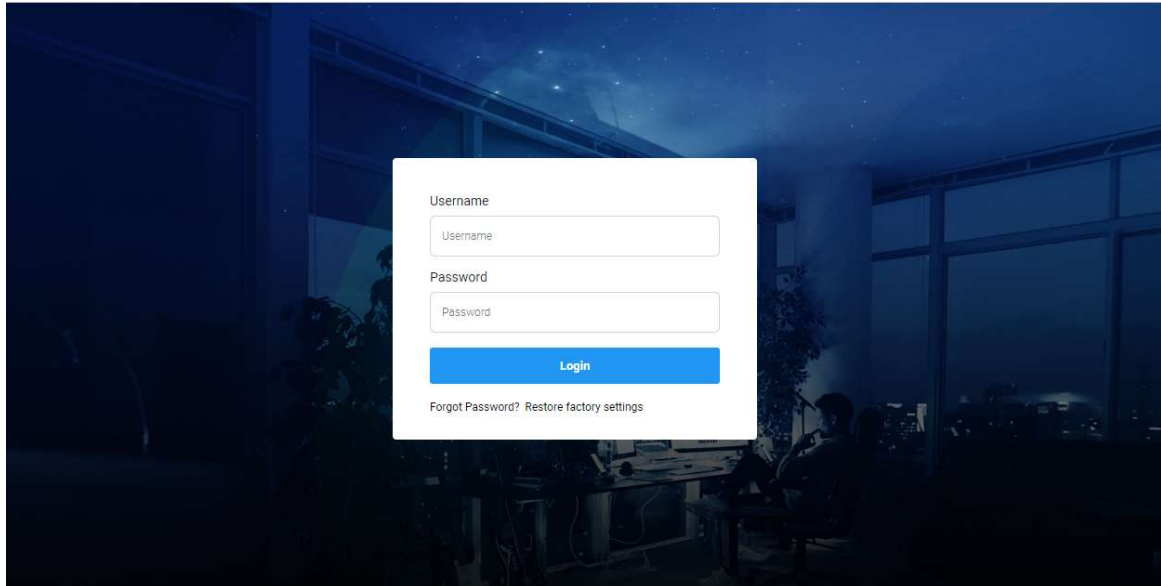
MAC OS 下网络配置：

1. 打开“System Preference”，选择“Network”。
2. 选择 RK1808S0，配置 ip 地址

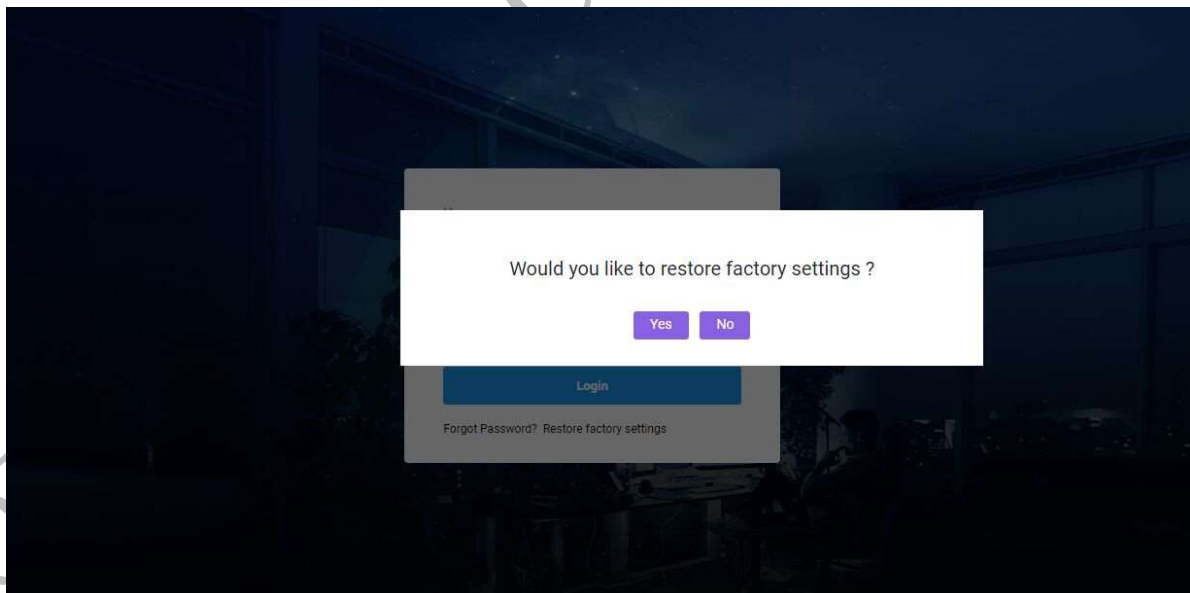


5.2 WEB 登录

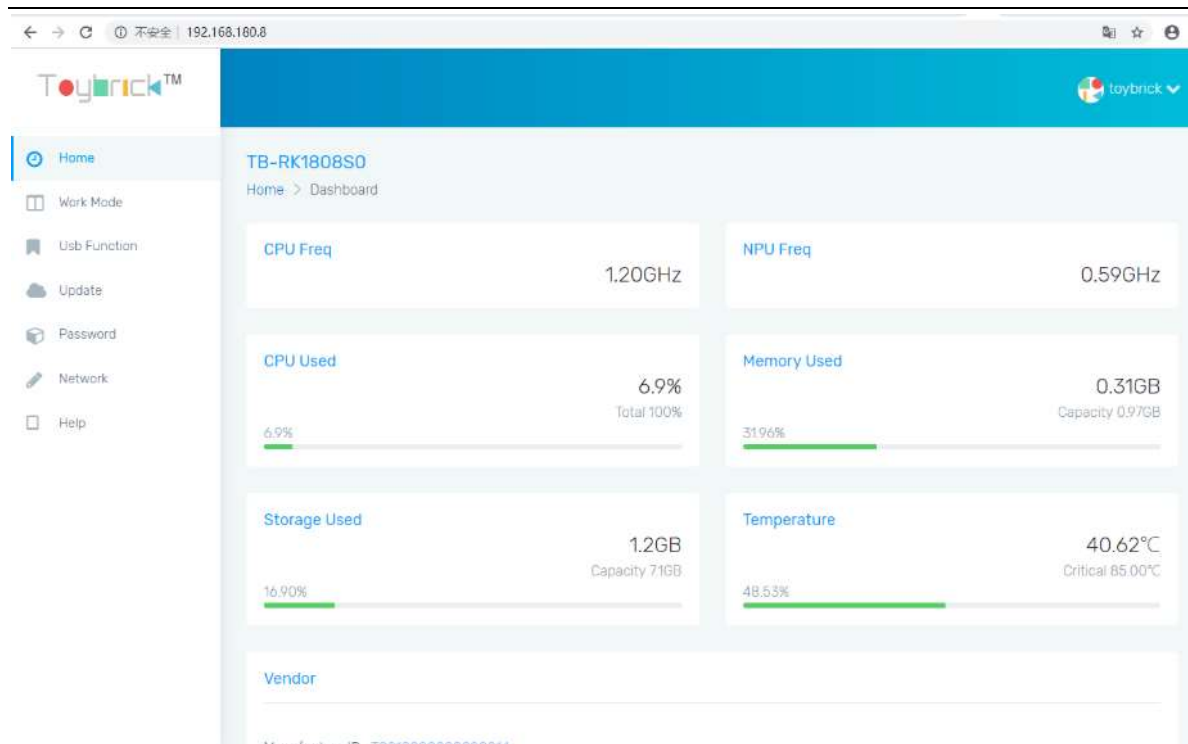
1. 网络配置完成后，浏览器输入 <http://192.168.180.8> 即进入 RK1808 人工智能计算棒登录页面。



若忘记密码，请点击下方链接，此操作不仅会重置密码，而且会将计算棒 web 配置恢复至出厂设置，请谨慎操作：

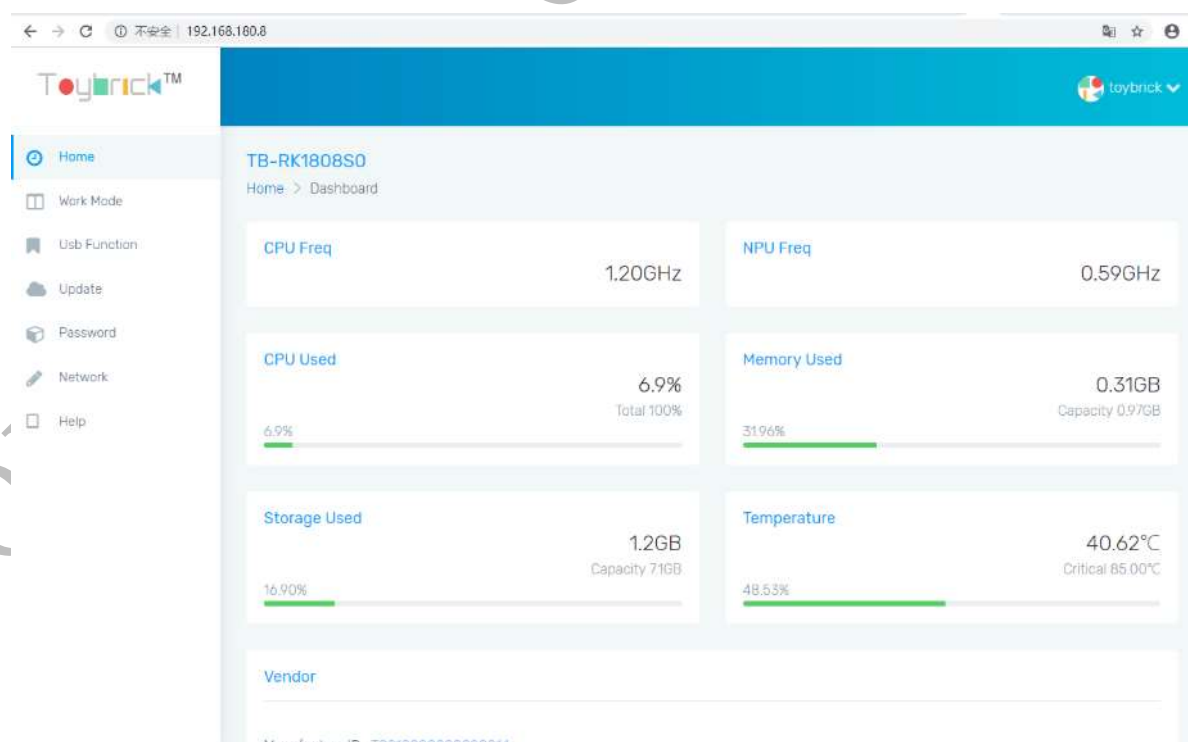


2. 输入用户名，密码登录(默认用户名和密码均为 toybrick)，即可进入 Home 页面。

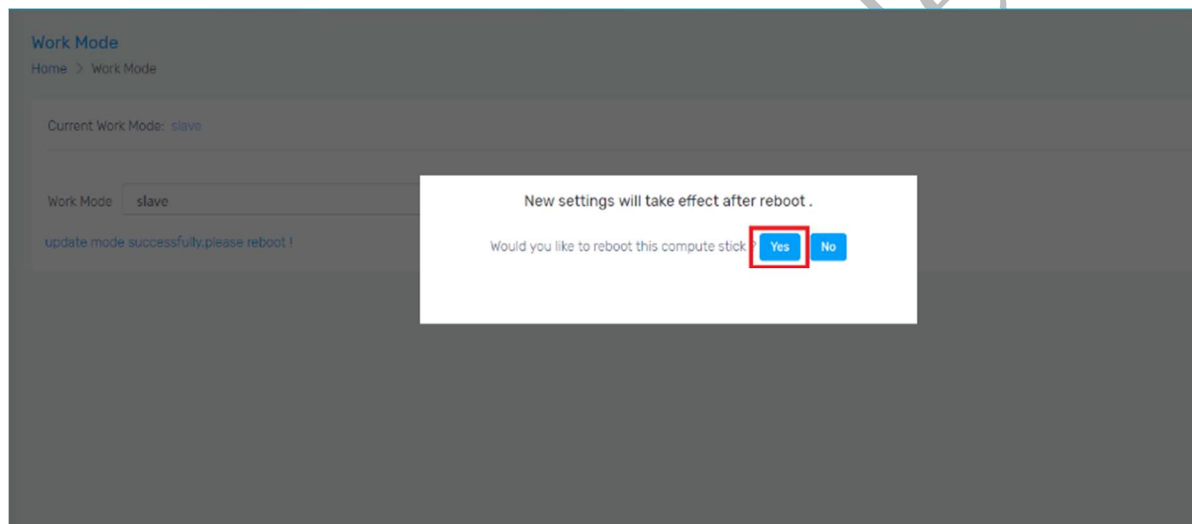
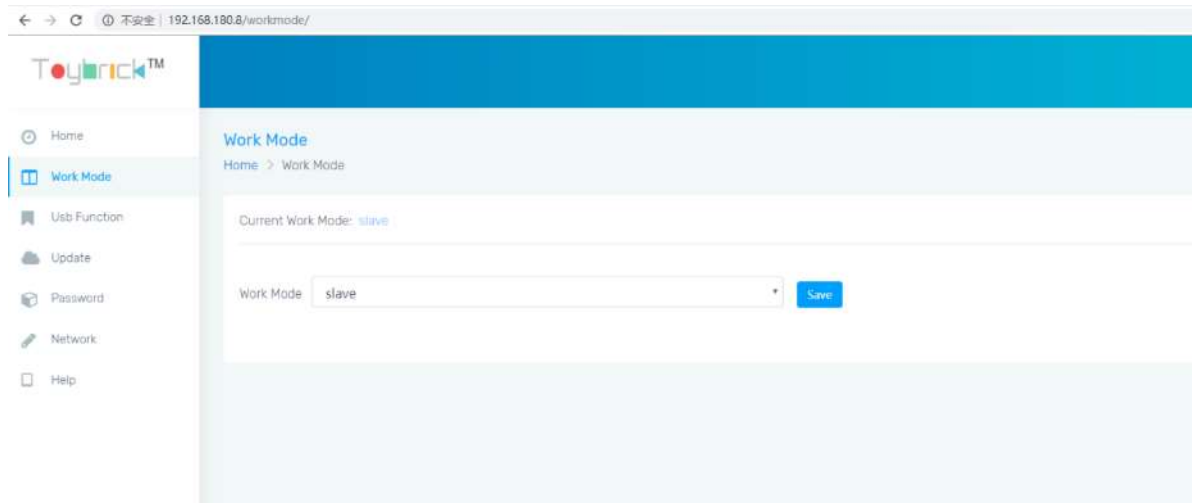


5.3 WEB 页面主要功能介绍

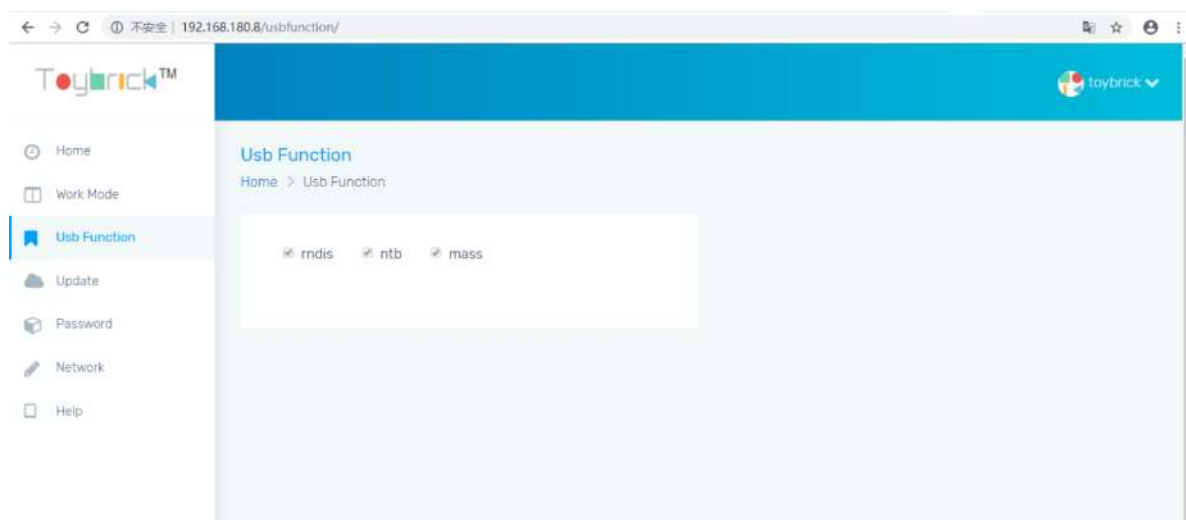
1. Home: 显示 RK1808 人工智能计算棒的主要系统信息和配置信息。



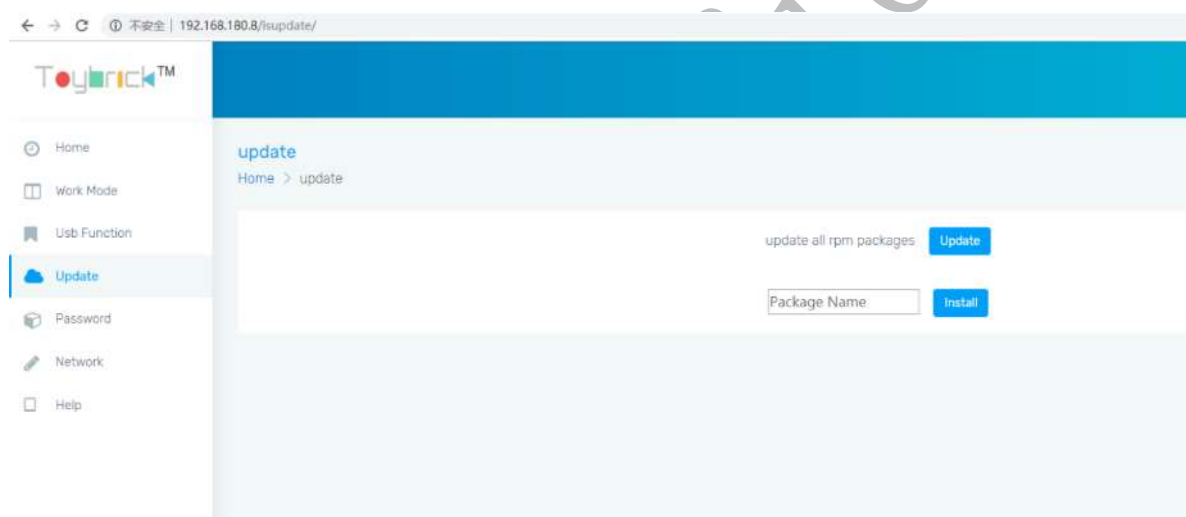
2. Work Mode:工作模式为主动或者被动模式，可进行切换。



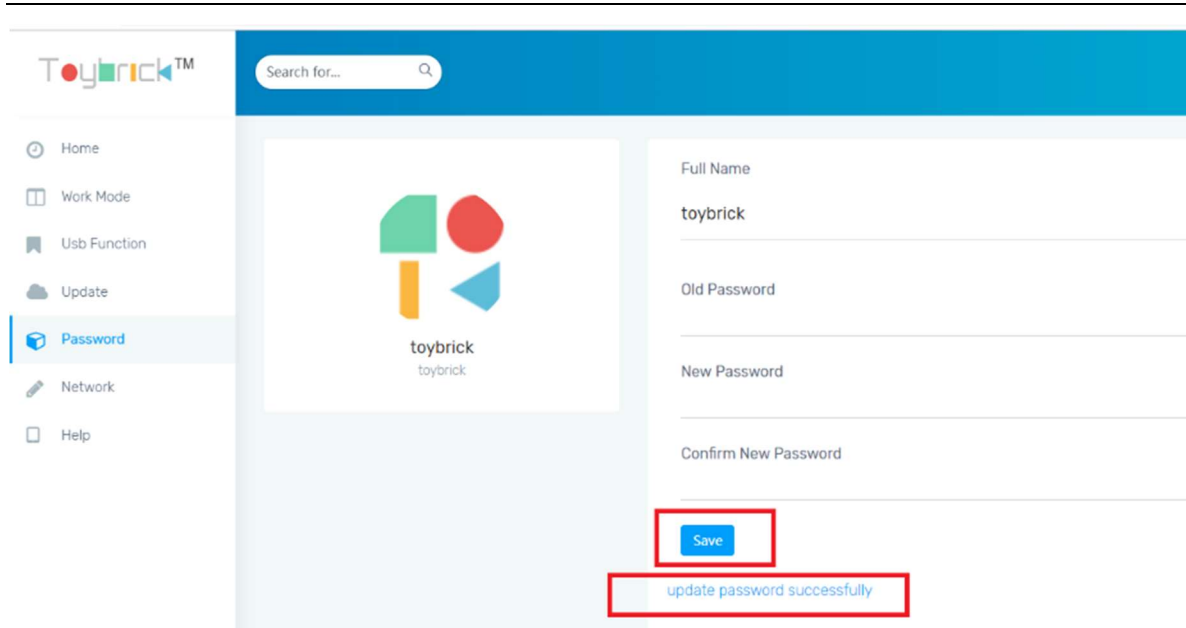
3. Usb Function:RK1808 人工智能计算棒的 usb 功能(rndis/ntb/mass)。



4. Update:更新, 安装 RK1808 人工智能计算棒 RPM 包。



5. Password:可在此处修改登录密码, 修改后需重新登录, 出现图中蓝色字样即修改成功。



Toybrick™

Search for...

Home

Work Mode

Usb Function

Update

Password

Network

Help

Full Name

toybrick

Old Password

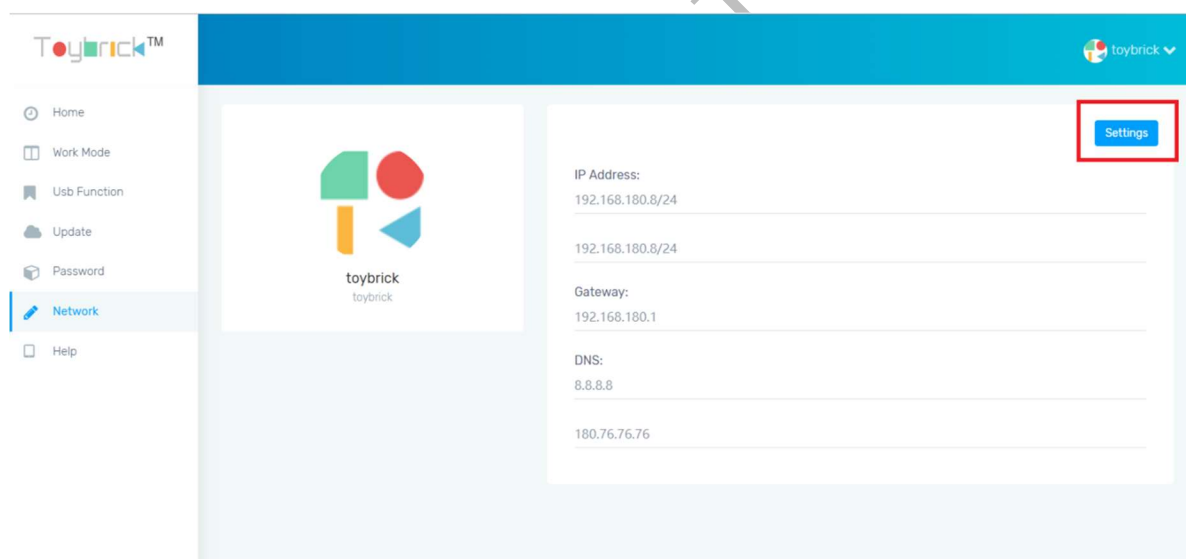
New Password

Confirm New Password

Save

update password successfully

6. Network:RK1808 人工智能计算棒网络配置页面显示当前的 IP，网关，DNS 信息。



Toybrick™

toybrick

Home

Work Mode

Usb Function

Update

Password

Network

Help

Settings

IP Address:

192.168.180.8/24

192.168.180.8/24

Gateway:

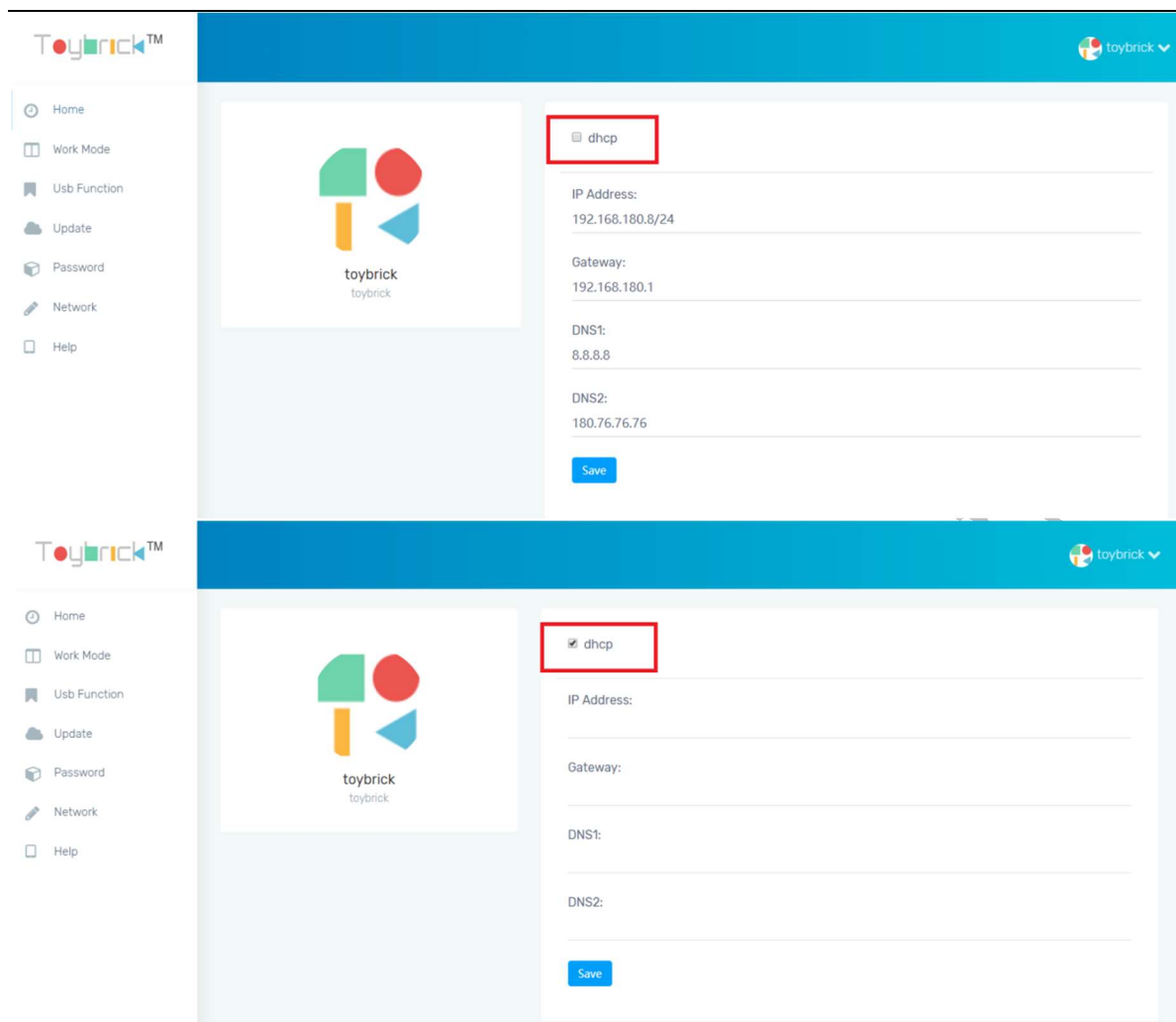
192.168.180.1

DNS:

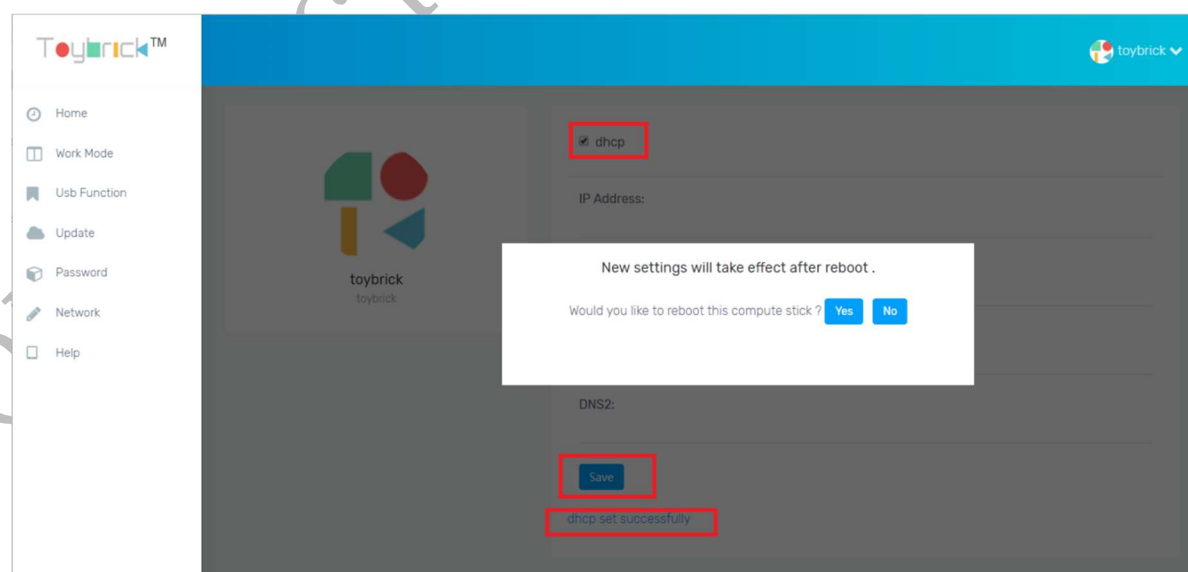
8.8.8.8

180.76.76.76

7. 点击右上角的 setting 按钮进入配置页面，配置静态 ip，网关，DNS 信息。

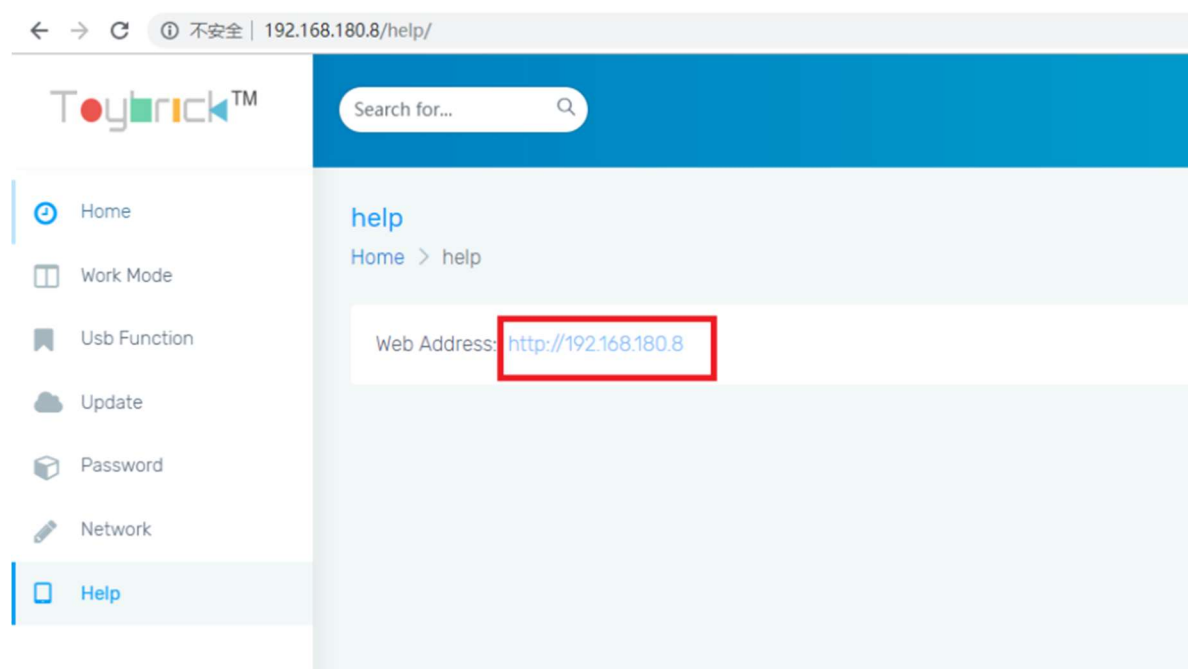


8. 若勾选 dhcp，上位机需配置桥接网络，如何配置上位机桥接网络请详见 [wiki](#) 说明。



9. 配置完成后，按 save 按钮，根据提示点击重启生效。

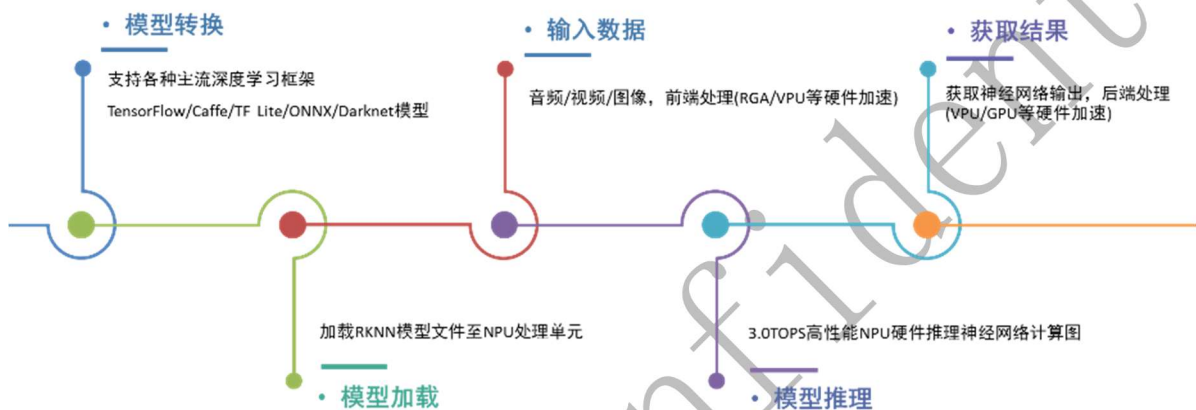
10. Help:RK1808 人工智能计算棒 web 配置的帮助信息。



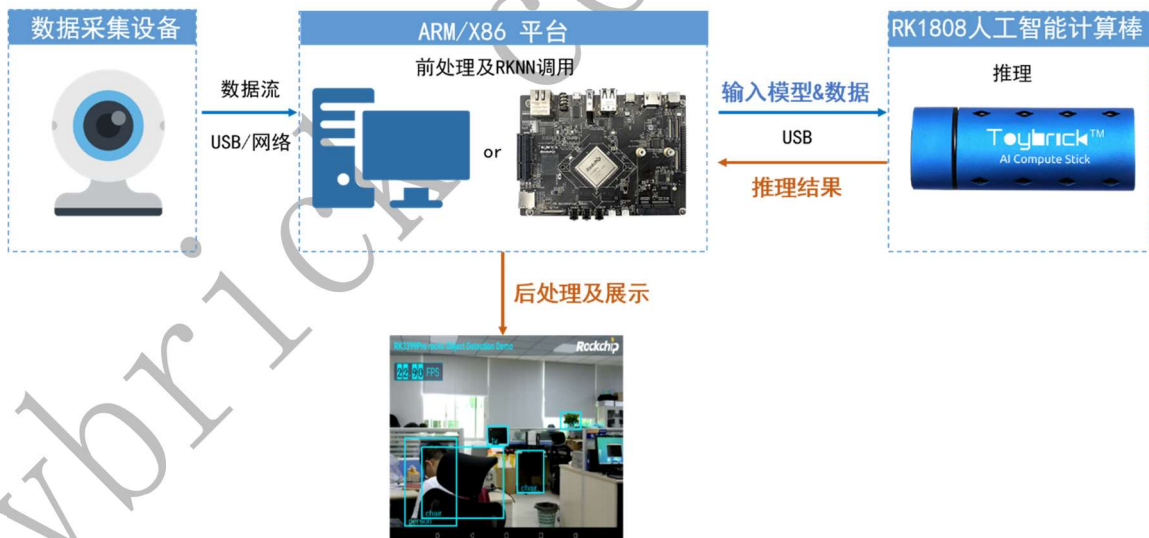
6 被动模式开发流程

被动模式下，RK1808 人工智能计算棒是一个通用 AI 加速器。主机端通过 RKNN-Toolkit 将模型及前处理后的数据传输给 RK1808 人工智能计算棒，RK1808 人工智能计算棒完成推理，并把结果返回主机端，主机端进行后处理以及显示等操作。

开发流程图如下：



整体数据流图如下：



被动模式下，有提供 python 和 C 两套 API 供上位机编程使用。

- Python 编程上位机需安装 RKNN-Toolkit，<http://repo.rock-chips.com/python/>链接下，提供了 RKNN-Toolkit 安装包，用户可以使用这些

安装包安装 RKNN-Toolkit。RKNN-Toolkit 的具体开发使用请参见 [《Rockchip User Guide RKNN Toolkit》](#)。RKNN-Toolkit 的更多文档请参考链接 http://repo.rock-chips.com/rk1808/rknn-toolkit_doc/。

- 在 <http://repo.rock-chips.com/rk1808/rknn-api/> 链接下，有供上位机 C 编程使用的库文件和头文件。C 编程的具体开发使用请参见 [《Rockchip User Guide RKNN API》](#)。上位机在执行编译出来的 C 语言的可执行程序前，需要先运行 npu_transfer_proxy 和计算棒进行通信，npu_transfer_proxy 的下载链接为 http://repo.rock-chips.com/rk1808/npu_transfer_proxy/。

Windows 下的 RKNN-Toolkit 使用需要预先安装 ntb 驱动，ntb 驱动请从 <http://repo.rock-chips.com/rk1808/driver/windows/ntb/> 链接地址下载，ntb 驱动安装请参考 [wiki](#)。

更多被动模式开发资料请登陆官方论坛查看：<http://t.rock-chips.com/>

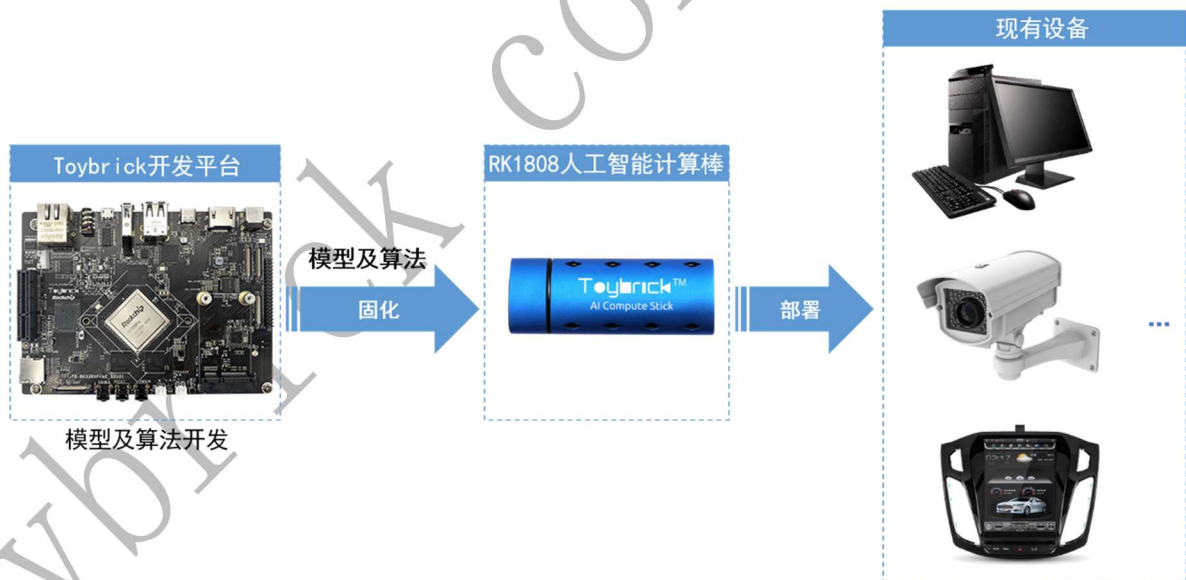
7 主动模式开发

主动模式下，RK1808 人工智能计算棒是一个专用 AI 应用模块。RK1808 人工智能计算棒作为主动设备，模型及算法固化在 RK1808 人工智能计算棒中，上位机只需通过 USB 口向计算棒输入数据（例如图片和视频流），RK1808 人工智能计算棒自动完成数据的前处理、推理、后处理，然后把处理结果通过 USB 口输出给上位机。

为了方便用户通过 USB 口传输数据，RK1808 人工智能计算棒会把 USB 口虚拟成网卡等标准设备，用户只需通过标准设备接口的操作（例如网络的 socket 编程）即可完成对 RK1808 人工智能计算棒数据的输入和输出。

RK1808 人工智能计算棒的 linux 操作系统为 fedora，用户可以通过 ssh 登录 fedora 系统进行开发和调试，root 根用户的密码为 toybrick，普通用户 toybrick 的密码为 toybrick。

主动开发模式总体流程图：



7.1 主动模式下的 AI 开发过程描述和介绍

主动模式下开发的 AI 程序分为两个部分：计算棒 AI 程序和上位机 AI 应用程序。

- 模型预先部署在计算棒的存储上，计算棒 AI 程序初始化环境并加载模型，

启动 socket server，接收上位机推送过来的数据，进行推理，然后把推理的结果返回给上位机。

- 上位机 AI 应用程序采集数据（如抓取摄像头数据），通过 socket client 把数据推送给计算棒，同时通过 socket client 接收计算棒返回的处理结果，并做进一步的处理（如显示）。

主动模式下的 RK1808 人工智能计算棒上 RKNN API 调用参考 [《Rockchip RK1808 Developer Guide Linux RKNN》](#)。

RK1808 人工智能计算棒上已经提供了 RKNN API C/C++语言所需的库和头文件，同时也预装了 RKNN API 的 python 3.6 库，用户可以在 RK1808 人工智能计算棒上开发部署 C/C++或 python 的主动 AI 程序。

7.2 产品部署过程描述和介绍

1. 把模型及算法固化于 RK1808 人工智能计算棒，并设置固化于 RK1808 人工智能计算棒中的程序开机自启动。
2. RK1808 人工智能计算棒插入目标设备，如网络摄像头设备、PC 机、无人机、智能小车等。
3. 目标设备上运行上位机服务程序，显示处理结果。
4. 具体案例详见 [wiki](#) 教程中 yolov3 主动模式案例。

7.3 开发工具 toybrick_deployc

为了方便客户开发调试和部署，另外提供一套主动模式的开发工具 toybrick_deployc，该工具只能运行在 toybrick 开发平台上，toybrick_deployc 工具的具体使用方法请参考 [wiki](#)。

更多主动模式开发资料请登陆官方论坛查看：<http://t.rock-chips.com/>