Workshop: Building end-to-end ML workflows with Arm

Gian Marco Iodice, Tech Lead ACL, Arm

Wei Xiao, Principal Evangelist AI Ecosystem, Arm

Hands-on session pre-requisites

Install VNC (Virtual Network Computing) viewer

In order to control the Raspberry desktop interface through your laptop, it is necessary to install VNC viewer (https://www.realvnc.com). This will allow to control the Raspberry Pi UI using the screen, keyboard and mouse of your laptop.

Note: VNC server is already included with Raspbian OS and it requires to be launched before establishing the connection from your laptop.

VNC viewer is supported on various platforms including Windows, Linux, MacOS, Chrome OS, Android and many more. For this workshop, we recommend installing VNC viewer on Chrome browser through the Chrome Web Store.

https://www.realvnc.com/en/connect/download/viewer/

Chrome browser

1. Install VNC® Viewer for Google Chrome™ from Chrome web store
2. Open Chrome browser
3. In the navigation bar, type:

- chrome://extensions/

4. In the search extensions field, look for VNC viewer and run it

Default login for Raspbian OS

- **User**: pi
- **Password**: raspberry

Using the serial terminal

Windows

1. Download and install the **PuTTY** terminal
    - https://www.putty.org
2. Execute (as administrator) **putty.exe**
3. Under **Connection type**, select:
    - **Serial**
4. In the Serial line field, enter the **COM#** for your USB serial port. You can find the **COM#** number from the **Device Manager** under the section **"Ports (COM & LPT)"**
5. In the **Speed field**, type:
    - **115200**
6. Click Open.
7. If you cannot see the login screen, press Enter multiple times

MacOS

1. Launch the Terminal app from **Spotlight** (**cmd + space**).
2. In the **Terminal**, enter the command:
    - **$ ls /dev/cu.usbserial-***
3. From the list of devices, copy the name of the device that contains **cu.usbserial**.
4. In the **Terminal**, enter the command:
    - **$ screen /dev/cu.usbserial-*<name> 115200**

    This command will start the serial communication with the Raspberry Pi

5. If you cannot see the login screen, press Enter multiple times

Linux

1. Open the terminal (**ctrl +alt + t**)
2. In the Terminal, enter the command:
    - **$ sudo apt-get install screen**

    Note: Skip if already installed
3. In the **Terminal**, enter the command:
    - **$ ls /dev/ttyUSB***
4. From the list of the devices, copy the name of the device that contains **ttyUSB** (I.e. ttyUSB0)
5. In the **Terminal**, enter the command:
    - **$ sudo screen /dev/ttyUSB<number> 115200**
6. If you cannot see the login screen, press Enter multiple times

Backup

Install Raspbian OS on Raspberry Pi 4

https://www.raspberrypi.org/documentation/installation/installing-images/

Note: Arm Cortex-A72 is an aarch64 architecture but Raspbian OS is based on an armv7 (aarch32) filesystem. This means that the Arm Compute Library will not be able to call the optimized routines for aarch64

VNC for Raspberry Pi

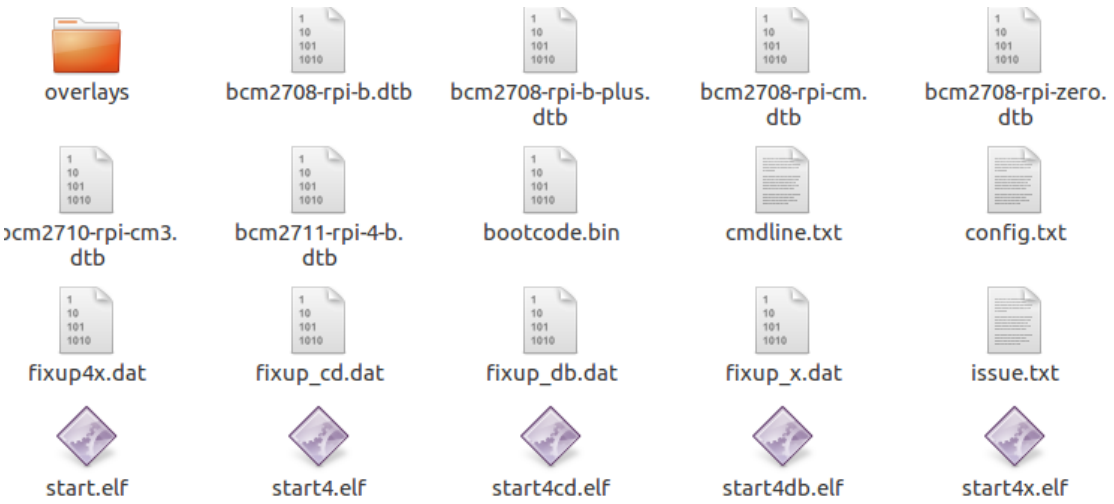
https://www.raspberrypi.org/documentation/remote-access/vnc/

Enabling Remote Access (SSH) on Raspbian OS

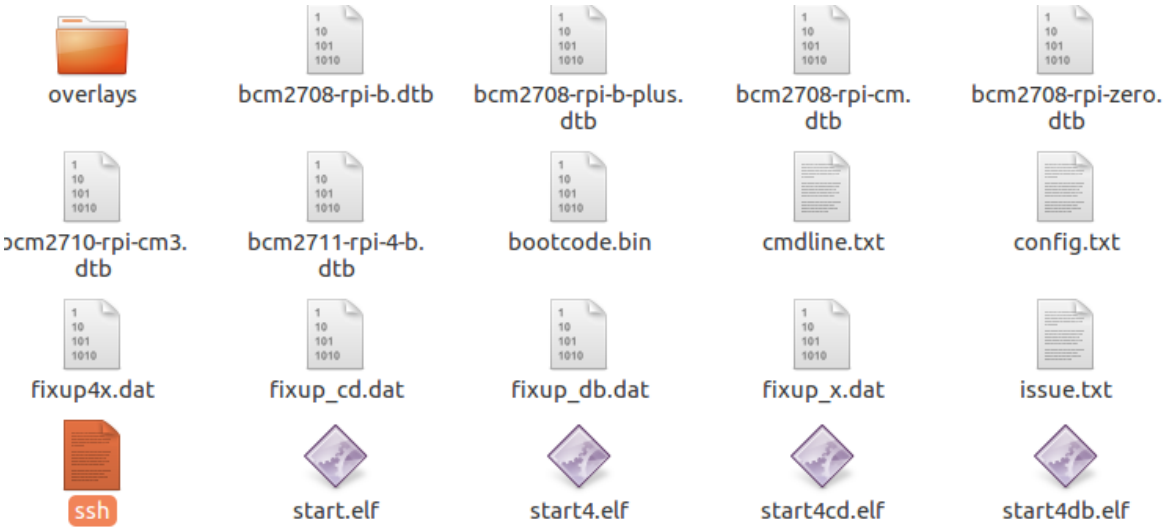
By default, Raspbian OS has the remote access disabled.

In order to allow SSH connections with wired ethernet cable, you need to:

1. Insert the sdcard with Raspbian OS in your laptop
2. Open the boot directory of the card



3. Write an empty txt file (without extension) named **"ssh"**

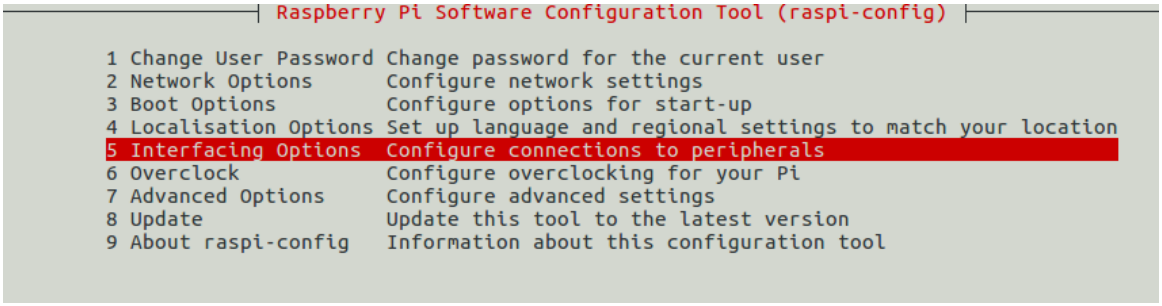


Enabling Serial Communication

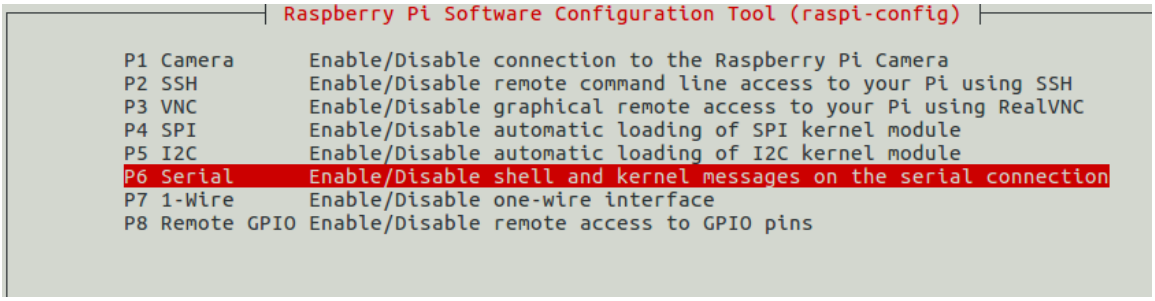
By default, Raspbian OS has the serial communication disabled.

In order to allow serial login, you need to:

1. Open the **Terminal** on your Raspberry Pi and type:
   - $ sudo raspi-config
2. Go down to **Interfacing options**



3. Click on **Serial** and select **Yes**



4. Reboot your Raspberry Pi

Launch VNC viewer from Chrome Browser

1. Open Chrome browser
2. In the navigation bar, type:
    - chrome://extensions/
3. Look for VNC® Viewer for Google Chrome™ and execute the App

SSH on Linux or MacOS

https://www.raspberrypi.org/documentation/remote-access/ssh/unix.md

Cross-compile ArmNN/ArmNN SDK

https://developer.arm.com/solutions/machine-learning-on-arm/developer-material/how-to-guides/cross-compiling-arm-nn-for-the-raspberry-pi-and-tensorflow

Cross-compile Compute Library (Development branch) Linux host

Before you begin, we recommend reading the Arm Compute Library documentation available at https://arm-software.github.io/ComputeLibrary/v19.08/

For the performance analysis hands-on session, the Arm Compute Library development branch must be cloned from the Linaro repository in order to access the experimental benchmark framework.

The benchmark framework allows to profile the performance of all the examples included with the library.

Note: The experimental benchmark framework is not available in the official release of Compute Library on GitHub.

1. Open https://mlplatform.org/contributing/ and click on the Arm Compute Library repository (https://review.mlplatform.org/#/admin/projects/ml/ComputeLibrary)

Development Repositories

Project	Repository
Arm NN	https://review.mlplatform.org/#/admin/projects/ml/armnn
Arm Compute Library	https://review.mlplatform.org/#/admin/projects/ml/ComputeLibrary
Arm Android NN Driver	https://review.mlplatform.org/#/admin/projects/ml/android-nn-driver

2. Clone the project with Git

Note: For Cross-compiling the Arm Compute Library, the cross-compiler version on your host machine must match the compiler version on your Raspberry Pi.

If you want to know what compiler version is used on Raspbian OS, you can follow the instructions below:

1. Open the **Terminal** on your Raspberry Pi and type:
   - **$ g++ -v**

The GCC version should be returned by this command.

```
pi@raspberrypi:~$ g++ -v
Using built-in specs.
COLLECT_GCC=g++
COLLECT_LTO_WRAPPER=/usr/lib/gcc/arm-linux-gnueabihf/8/lto-wrapper
Target: arm-linux-gnueabihf
Configured with: ../src/configure -v --with-pkgversion='Raspbian 8.3.0-6+rpi
with-gcc-major-version-only --program-suffix=-8 --program-prefix=arm-linux-g
dir=/usr/lib --enable-nls --enable-bootstrap --enable-clocale=gnu --enable-l
e-libquadmath --disable-libquadmath-support --enable-plugin --with-system-zl
 --with-float=hard --disable-werror --enable-checking=release --build=arm-li
Thread model: posix
gcc version 8.3.0 (Raspbian 8.3.0-6+rpi1)
```

The GNU toolchain required for this workshop is the **arm-linux-gnueabihf** (32 bit) **8.3.0** available at the following link:

https://www.linaro.org/downloads/)

Linaro Toolchain

The flatest Arm release of the pre-built GNU cross-toolchain for Cortex-A GCC 8.3-2019.03 is now available on the Arm Developer website.

Linaro provides monthly GCC source archive snapshots of the current Linaro GCC release branch, as well as quarterly releases of pre-built Linaro GNU cross-toolchain binary archives.

The following tables provide direct access to the most common Linux and bare-metal ABI variants of the Linaro binary cross-toolchain quarterly releases. Both x86_64 Linux and Mingw32 (MS Windows compatible) host binaries are provided:

Latest Linux Targeted Binary Toolchain Releases

arm-linux-gnueabihf	*32-bit Armv7 Cortex-A, hard-float, little-endian*	Release-Notes	Binaries	Source
armv8l-linux-gnueabihf	*32-bit Armv8 Cortex-A, hard-float, little-endian*	Release-Notes	Binaries	Source
aarch64-linux-gnu	*64-bit Armv8 Cortex-A, little-endian*	Release-Notes	Binaries	Source

Once the toolchain has been installed as described in the Arm Compute Library documentation, you can compile the library using the following instruction:

$ scons -j6 arch=armv7a examples=1 asserts=1 opencl=0 gles_compute=0 neon=1 timers=1 debug=0 Werror=0 validation_tests=0 benchmark_tests=0

All the executables will be in the folder build

How to use Streamline

https://developer.arm.com/docs/100769/0605/introduction-to-streamline-performance-analyzer