Build Petalinux 2021.2 with Vitis AI 2.0 and Smartcam to Run Resnet50 Demo

首先需要的環境有

1. Ubuntu 18.04
2. PetaLinux Tools - Installer - 2021.2
3. Kria K26 SOM Board Support Package - 2021.2
4. Vitis-AI Lab 2.0

建置開始

- Step 1:

  下載完 PetaLinux Tools – Installer 後

  安裝 Dependencies

  <command> sudo apt-get install gcc g++ libtinfo5 libncurses5-dev libncursesw5-dev libtool net-tools autoconf xterm texinfo gcc-multilib gawk zlib1g libz1:i386 zlib1g-dev build-essential

  <command> ./petalinux-v2021.2-final-installer.run -d <自訂安裝的路徑>

  - 不能執行請先 sudo chmod -R 777

  <command> source <自訂安裝的路徑>/settings.sh

  以上便安裝完 PetaLinux Tools 與設定好環境變數

(optional)有時候同個版本像是 petalinux 2021.1 有 update 1，有加入新的 Vitis ai layers，會影響到使用，因此會建議更新 petalinux tool

---> From Network:

<command> petalinux-upgrade -u http://petalinux.xilinx.com/sswreleases/rel-v2021/sdkupdate/2021.1_update1/ -p "aarch64" --wget-args "--wait 1 -nH --cut-dirs=4"

---> From Local:

<command> petalinux-upgrade -f <Local eSDK Directory Path> -p "aarch64"

- Step 2: 創建 petalinux project

  <command> petalinux-create -t project -s /<放 kv260 BSP 的路徑>/xilinx-k26-starterkit-v2021.2-final.bsp -n kv260_os(此為專案名稱與資料夾)

  <command> cd ./kv260_os

  <command> ls 後可看見下圖:

```
parallels@parallels-Parallels-Virtual-Platform:~/kria_kv260$ petalinux-create -t project -s ../kria_bsp/xilinx-k26-starterkit-v2
021.1-final.bsp -n kv260_os
INFO: Create project: kv260_os
INFO: New project successfully created in /home/parallels/kria_kv260/kv260_os
parallels@parallels-Parallels-Virtual-Platform:~/kria_kv260$ cd ./kv260_os/
parallels@parallels-Parallels-Virtual-Platform:~/kria_kv260/kv260_os$ ls
components  config.project  hardware  pre-built  project-spec  README  README.hw
parallels@parallels-Parallels-Virtual-Platform:~/kria_kv260/kv260_os$
```

要先將基本的 Petalinux build 起來，後面再加入 accelerated application 和 AI 等等之類的項目

<command> petalinux-build

- 看電腦性能，我筆電 build 了兩小時，建議在 Server 上面 build 完再把 kernel image 載到 windows

◆ Step 3: 將 Vitis AI 2.0 跟一些 application 加入到 Project 中

1. Vitis AI 2.0

原始在 petalinux 2021.2 的 vitis ai 為 1.4，因此我們要先將原始的刪除

<command> cd components/yocto/layers/

<command> sudo rm -r meta-vitis-ai

然後再裝新的 meta-vitis-ai

<command> git clone -b rel-v2021.2 https://github.com/jlamperez/meta-vitis-ai.git meta-vitis-ai

再來要去將預設的 meta-vitis-ai config 路徑刪掉

<command> cd ~/kv260_os

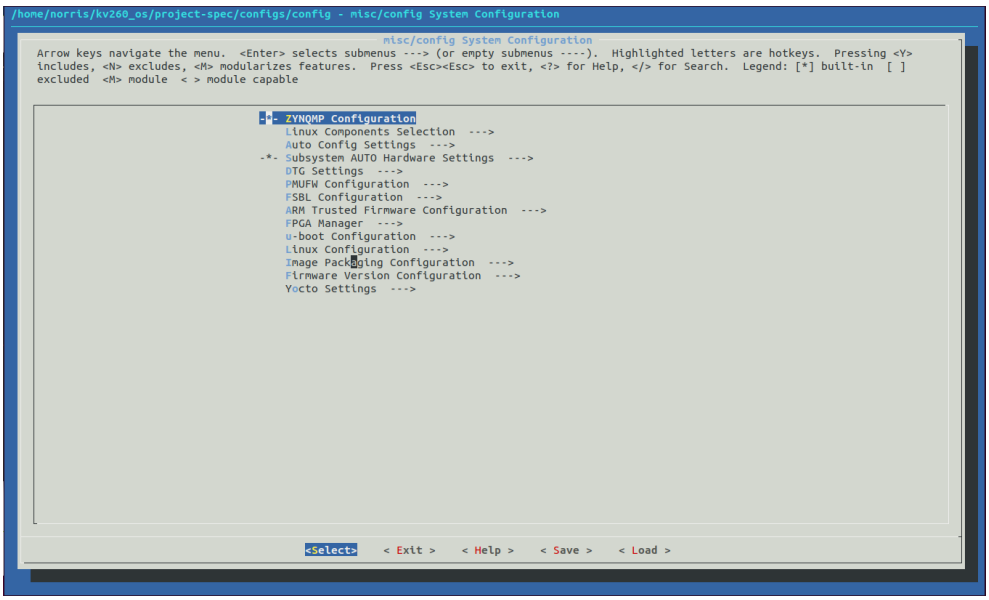
<command> vi build/conf/bblayers.conf

找到 ${SDKBASEMETAPATH}/layers/meta-vitis-ai 刪掉

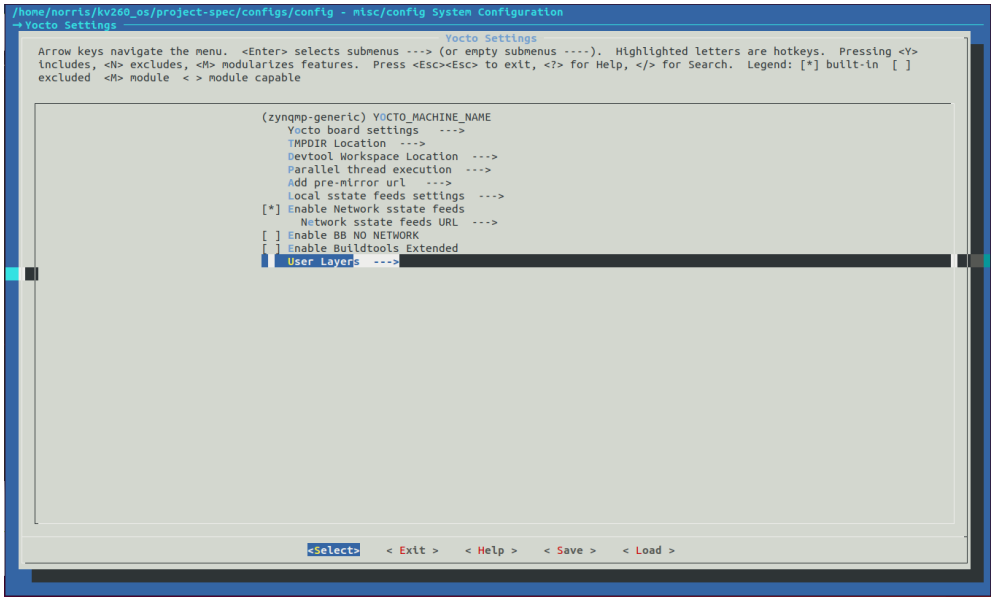
再來需要將新的 meta-vitis-ai 加入到 config 裡面

<command> petalinux-config

出現下圖

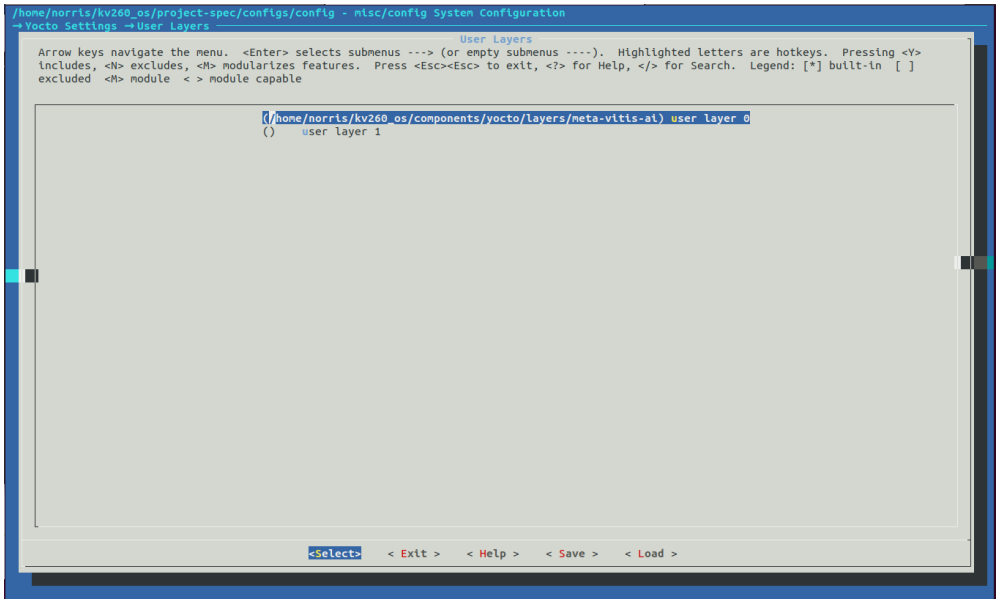


選擇 **Yocto Settings ---> User Layers**



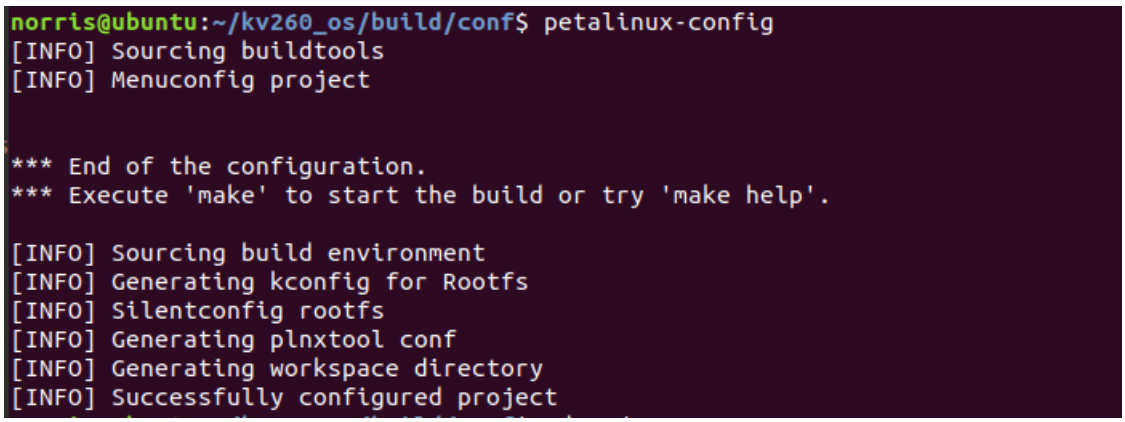
打上如下圖中的這串文字，但前面需依照你自己的路徑

eg. /home/xxx/<project_name>components/yocto/layers/meta-vitis-ai



Save ---> Exit 直到退出整個 GUI

成功會看到下圖



再來，petalinux build 時預設是使用 petalinux-image-minimal

你可以選擇 petalinux-image-minimal.bbappend 或是 petalinux-image-full.bbappend 再透過

```
<command> vi components/yocto/layers/meta-petalinux/recipes-core/images/petalinux-image-minimal.bb
```

添加下列文字

```
IMAGE_INSTALL_append = "packagegroup-petalinux-vitisai-dev \
                        packagegroup-petalinux-vitisai"
```

如此 Vitis ai 2.0 的部分就結束了

2. Add FPGA Firmware

直接加入現有的 Firmware 即可

```
<command> cd ~/kv260_os
<command> git clone -b xlnx_rel_v2021.2 https://github.com/Xilinx/kv260-firmware
```

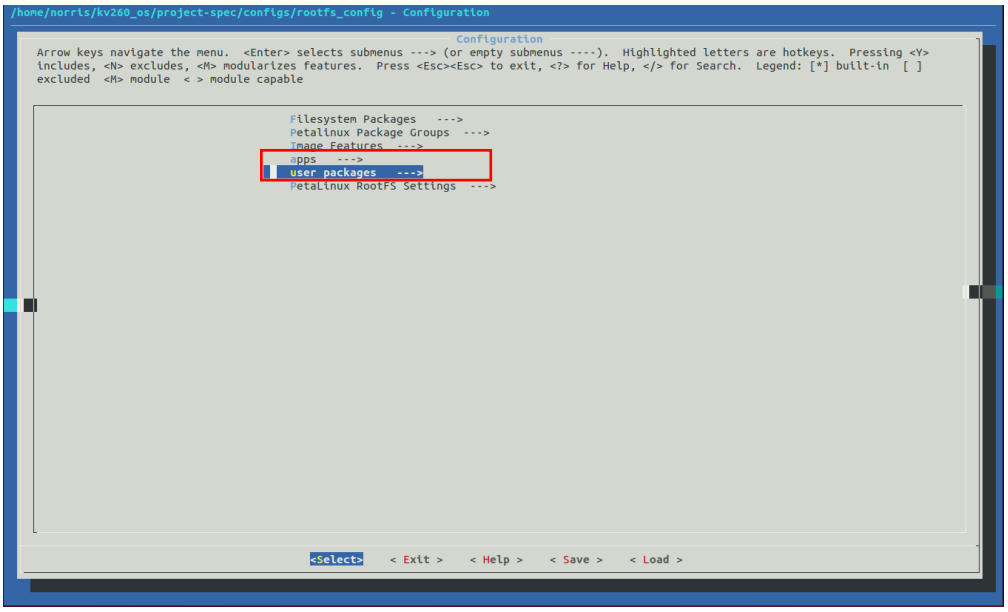
再來各自加入 firmware

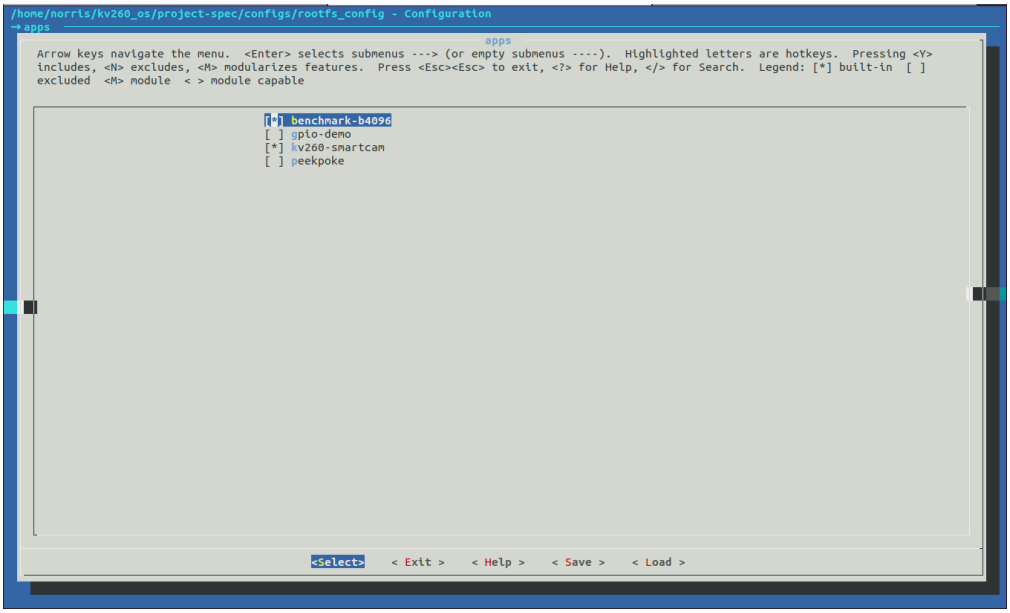
```
# benchmark-b4096.bb 這條可不必，因為用的是 B4096 的 DPU
<command> petalinux-create -t apps --template fpgamanager -n benchmark-b4096 --enable --srcuri "./kv260-firmware/benchmark-b4096/kv260-benchmark-b4096.bit ./kv260-firmware/benchmark-b4096/kv260-benchmark-b4096.xclbin ./kv260-firmware/benchmark-b4096/shell.json ./kv260-firmware/benchmark-b4096/kv260-benchmark-b4096.dtsi"
```

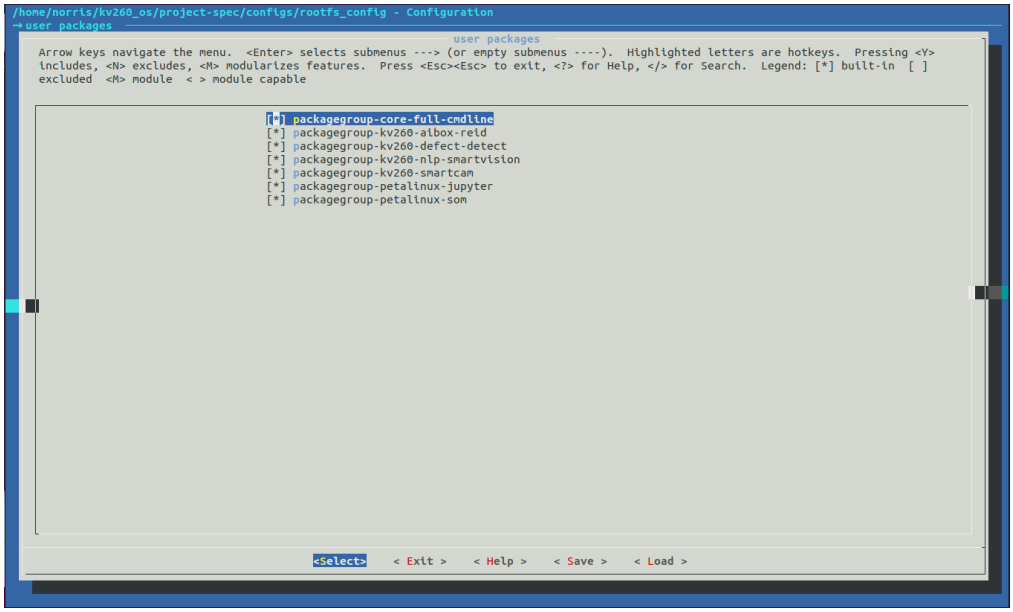
```
# smartcam.bb smartcam 中用的是 B3136 的 DPU
<command> petalinux-create -t apps --template fpgamanager -n karp-smartcam --enable --srcuri "./kv260-firmware/smartcam/kv260-smartcam.bit ./kv260-firmware/smartcam/kv260-smartcam.xclbin ./kv260-firmware/smartcam/shell.json ./kv260-firmware/smartcam/kv260-smartcam.dtsi"
```

最後啟用 root config 來勾選 image 要使用的應用

```
<command> petalinux-config -c rootfs
```


/home/norris/kv260_os/project-spec/configs/rootfs_config - Configuration
Configuration
Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty submenus ----). Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to exit, <?> for Help, </> for Search. Legend: [*] built-in [ ] excluded <M> module < > module capable
Filesystem Packages --->
Petalinux Package Groups --->
Image Features --->
apps --->
user packages --->
PetaLinux RootFS Settings --->
<Select> < Exit > < Help > < Save > < Load >


/home/norris/kv260_os/project-spec/configs/rootfs_config - Configuration
→ apps
apps
Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty submenus ----). Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to exit, <?> for Help, </> for Search. Legend: [*] built-in [ ] excluded <M> module < > module capable
[*] benchmark-b4096
[ ] gpio-demo
[*] kv260-smartcam
[ ] peekpoke
<Select> < Exit > < Help > < Save > < Load >


/home/norris/kv260_os/project-spec/configs/rootfs_config - Configuration
→ user packages
user packages
Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty submenus ----). Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to exit, <?> for Help, </> for Search. Legend: [*] built-in [ ] excluded <M> module < > module capable
[*] packagegroup-core-full-cmdline
[*] packagegroup-kv260-aibox-reid
[*] packagegroup-kv260-defect-detect
[*] packagegroup-kv260-nlp-smartvision
[*] packagegroup-kv260-smartcam
[*] packagegroup-petalinux-jupyter
[*] packagegroup-petalinux-som
<Select> < Exit > < Help > < Save > < Load >

一樣儲存後退出，會看到

```
norris@ubuntu:~/kv260_os$ petalinux-config -c rootfs
[INFO] Sourcing buildtools
[INFO] Silentconfig project
[INFO] Generating kconfig for Rootfs
[INFO] Menuconfig rootfs


*** End of the configuration.
*** Execute 'make' to start the build or try 'make help'.

[INFO] Generating plnxtool conf
[INFO] Successfully configured rootfs
```

- **Step 4: Build Petalinux Image**

  <command> petalinux-build -c petalinux-image-minimal

- **Step 5: Create SD Card Image**

  <command> petalinux-package --boot --u-boot --dtb images/linux/u-boot.dtb --force

  <command> petalinux-package -wic

  如上若無法成功開機，可再嘗試以下方法

```
<command> petalinux-config

# change Image Packaging Configuration -> Root filesystem type  to INITRD
# change Image Packaging Configuration -> INITRAMFS/INITRD Image Name (NEW) to
petalinux-image-minimal

<command> petalinux-build -x mrproper
<command> petalinux-build

# then in the next command rename ramdisk.cpio.gz.u-boot to rootfs.cpio.gz.u-boot
<command> petalinux-package --wic --bootfiles "rootfs.cpio.gz.u-
boot,boot.scr,Image,system.dtb"
```

  建立好的 Image 會是 images/linux/petalinux-sdimage.wic

- **Step 6: 使用 balenaEtcher 燒錄至 SD 卡**

- **Step 7: 將 SD 卡放置到 KV260 上並開機**

- **Step 8: 回到 Ubuntu，開始進行 Vitis AI 的 Docker 建置**

  1. 安裝 Docker

     <command> sudo apt-get install docker.io

  2. 將 Docker 每次執行時都需要 sudo 的指令關閉

<command> sudo chmod 666 /var/run/docker.sock

3. 下載 Vitis AI 2.0 的 Docker Image

<command> docker pull xilinx/vitis-ai:2.0.0

4. 下載 Vitis AI 的 github 檔案

<command> git clone -b v2.0 https://github.com/Xilinx/Vitis-AI.git

5. 下載官方 Vitis AI 2.0 的 Lab

Vitis-AI Lab 2.0

6. 到剛剛 git clone 完的 Vitis-AI 資料夾下，並執行

<command>./docker_run.sh xilinx/vitis-ai:2.0.0

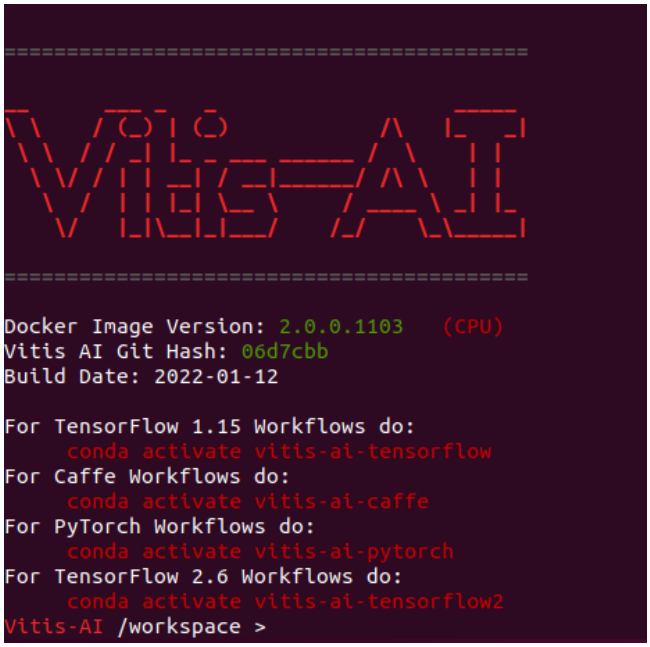
若出現

sed: can't read ./setup/docker/dockerfile/PROMPT.txt: No such file or directory

的錯誤，請修改 docker_run.sh 中的文字為下圖

```
#!/bin/bash
# Copyright 2020 Xilinx Inc.

sed -n '1, 5p' ./setup/docker/dockerfiles/PROMPT.txt
read -n 1 -s -r -p "Press any key to continue..." key

sed -n '5, 15p' ./setup/docker/dockerfiles/PROMPT.txt
read -n 1 -s -r -p "Press any key to continue..." key

sed -n '15, 28p' ./setup/docker/dockerfiles/PROMPT.txt
read -n 1 -s -r -p "Press any key to continue..." key

sed -n '28, 61p' ./setup/docker/dockerfiles/PROMPT.txt
read -n 1 -s -r -p "Press any key to continue..." key

sed -n '62, 224p' ./setup/docker/dockerfiles/PROMPT.txt
read -n 1 -s -r -p "Press any key to continue..." key

sed -n '224, 308p' ./setup/docker/dockerfiles/PROMPT.txt
read -n 1 -s -r -p "Press any key to continue..." key


confirm() {
  echo -en "\n\nDo you agree to the terms and wish to proceed [y/n]? "
  read REPLY
  case $REPLY in
    [Yy]) ;;
    [Nn]) exit 0 ;;
    *) confirm ;;
  esac
    REPLY=''
}
```

7. 成功進入到 Vitis AI 的 Docker 環境後會長這樣



執行

<command> conda activate vitis-ai-caffe

進入到 caffe 的環境中

- Step 9: 設定 cross-compilation 的系統環境

先跳出 Docker 環境，並 cd 到 Vitis-AI/setup/mpsoc 下執行 sdk

<command> cd Vitis-AI/setup/mpsoc
<command> sudo ./sdk-2021.2.0.0.sh

然後輸入個位置裝你的 sdk

安裝完畢後，source 一下 sdk 的環境

<command> . /<sdk 位置>/environment-setup-cortexa72-cortexa53-xilinx-linux

- Step 10: 在 Ubuntu 上 build Resnet50 的執行檔

<command> cd Vitis-AI/demo/VART/resnet50
<command> bash -x build.sh

- Step 11: Quantize(量化)Resnet50 的模型

1. 再回到 caffe 的環境下

<command> conda activate vitis-ai-caffe

2. 解壓縮下載下來的 Vitis AI Lab 的 zip 檔案，並把當中的 vai_q_c 複製到 Vitis-AI 的根目錄下

<command> cp -r training/vai_q_c Vitis-AI

更改 shell 檔來進行 quantize

<command> vim Vitis-AI/vai_q_c/lab/1_caffe_quantize_for_edge.sh

calib_iterc 和 test_iter 後的 10 改成 2

<command> vim Vitis-AI/vai_q_c/lab/cf_resnet50_imagenet_224_224_7.7G_2.0/float/trainval.prototxt

```
 9   phase: TRAIN
10   transform_param {
11     mirror: true
12     crop_size: 224
13     mean_value: 104
14     mean_value: 107
15     mean_value: 123
16   }
17   image_data_param {
18     source: "images/val.txt"
19     root_folder: "images/"
20     batch_size: 64
21     shuffle: true
22   }
23 }
24 layer {
25   name: "data"
26   type: "ImageData"
27   top: "data"
28   top: "label"
29   include {
30     phase: TEST
31   }
32   transform_param {
33     crop_size: 224
34     mean_value: 104
35     mean_value: 107
36     mean_value: 123
37   }
38   image_data_param {
39     source: "images/val.txt"
40     root_folder: "images/"
41     batch_size: 20
```

<command> sh 1_caffe_quantize_for_edge.sh

即開始進行 quantize

◆ Step 12: Compile Resnet50 的模型為 xmodel

回到 Ubuntu 的 Vitis AI 上，啟動 caffe 環境

此處因為使用的開發版為 KV260，因此需要更改使用的 DPU

<command> vim Vitis-AI/vai_q_c/lab/2_caffe_compile_for_edge.sh

EDGE_TARGET=ZCU102 改成 KV260

<command> vim /opt/vitis_ai/compiler/arch/DPUCZDX8G/KV260/arch.json

4096 的部分改成 3136

執行

<command> sh 2_caffe_compile_for_edge.sh

完成 Resnet50 compile

◆ **Step 13：將 Ubuntu 上編譯好的執行檔、資料以及模型丟到 KV260 上**

回到 KV260 上，先建立資料夾

<command> mkdir -p Vitis-AI/demo/VART/resnet50

<command> mkdir -p /usr/share/vitis_ai_library/models/resnet50/

➢ **執行檔 Resnet50 和 label 檔**

<command> cd <Ubuntu 上的 Vitis-AI>/demo/VART/resnet50/

<command> scp -r resnet50 words.txt <KV260 的 username>@<IP>:~/Vitis-AI/demo/VART/resnet50/

➢ **要安裝在 KV260 上的 VART (Vitis AI Run Time Library)**

<command> cd <Ubuntu 上的 Vitis-AI>/setup/

<command> scp -r mpsoc <KV260 的 username>@<IP>:~/

➢ **要在 KV260 上 inference 的圖片 (圖片 zip 檔案請到此下載)**

KV260 上

<command> wget https://www.xilinx.com/bin/public/openDownload?filename=vitis_ai_runtime_r2.0.0_image_video.tar.gz

<command> tar -zxvf vitis_ai_runtime_r2.0.0_image_video.tar.gz -C ~/Vitis-AI/demo/VART/

➢ **KV260 要運行的 xmodel**

<command> cd <Ubuntu 上的 Vitis-AI>/vai_q_c/lab/cf_resnet50_imagenet_224_224_7.7G_2.0/vai_c_output_KV260/

<command> scp -r md5sum.txt meta.json resnet50.xmodel <KV260 的 username>@<IP>:/usr/share/vitis_ai_library/models/resnet50/

◆ **Step 14：在 KV260 上安裝 VART 並運行 Demo**

安裝 VART

<command> cd mpsoc/VART

<command> sh bash target_vart_setup.sh

啟用 karp-smartcam

沒這步等於沒啟用到 DPU

<command> sudo xmutil unloadapp

<command> sudo xmutil loadapp karp-smartcam

運行 Demo

<command> cd ~/Vitis-AI/demo/VART/resnet50/

<command>./resnet50 /usr/share/vitis_ai_library/models/resnet50/resnet50.xmodel

成功就會看到

```
xilinx-k26-som-2021_2:~/Vitis-AI/demo/VART/resnet50$ ./resnet50 /usr/share/vitis_ai_library/models/resnet50/resnet50.xmodel
WARNING: Logging before InitGoogleLogging() is written to STDERR
I0803 23:23:28.675163  1632 main.cc:292] create running for subgraph: subgraph_conv1

Image : 001.jpg
top[0] prob = 0.982862  name = brain coral
top[1] prob = 0.008503  name = coral reef
top[2] prob = 0.006622  name = jackfruit, jak, jack
top[3] prob = 0.000544  name = puffer, pufferfish, blowfish, globefish
top[4] prob = 0.000330  name = eel

(Classification of ResNet50:1632): Gtk-WARNING **: 23:23:29.052: cannot open display:
```

無法出現圖的情況還未解決

~End~