Kneron Inc

Document Name: **KL520 NCPU Tutorial**

KL520 NCPU Tutorial

Kneron Inc

Table of Contents

1. Introduction

1.1 Working Flow

To generate the model binary files for firmware and load it into KL520, you need to follow these steps:

(1) converting deep learning models from different deep learning framework (Keras, Tensorflow, Pytorch, Caffe) to ONNX format;

(2) conducting fixed pointer analysis on the selected model and image dataset; batch-compiling the related model files to model binary files and firmware information files;

2. Installation

2.1 Prerequisite

For Windows 10 Home system, please install Docker Toolbox
https://download.docker.com/win/stable/DockerToolbox.exe

For Windows 10 Professional system, please install

https://docs.docker.com/docker-for-windows/install/

For Ubuntu 16.0.4, please follow these instructions

https://docs.docker.com/install/linux/docker-ce/ubuntu/ to install.

2.2 Docker check

Before using this toolchain, you need to start the docker properly.

For windows 10 home users, if you successfully install Docker Toolbox, which is introduced in the Part 2.1, you can search and find Docker Quickstart Terminal in your computer. Everytime you want to use this toolchain, please remember to start Docker Quickstart Terminal. Figure 1 shows the appearance of Docker Quickstart Terminal which is started properly. If the terminal does not show this appearance, it means there is some problem for the installation of Docker.

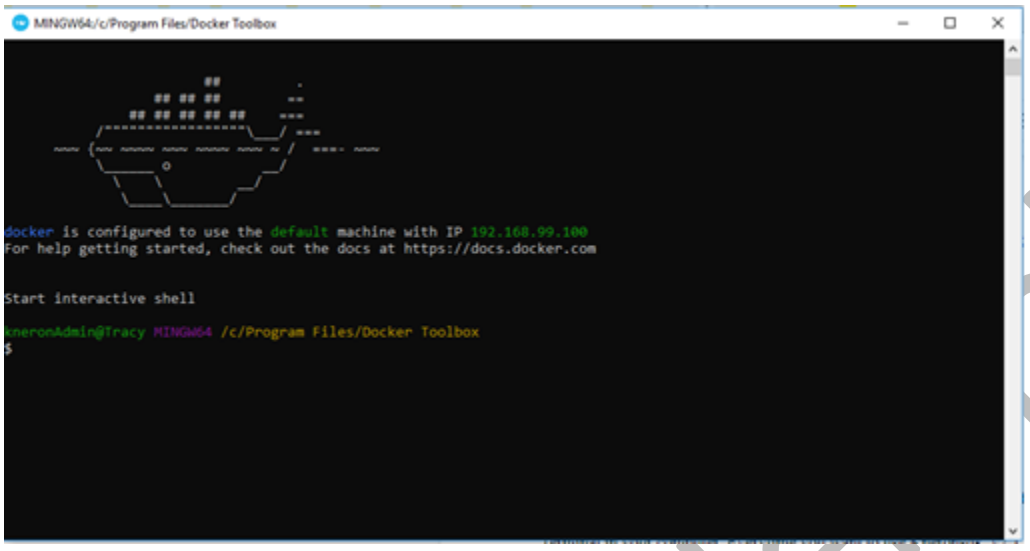

Figure 1 Appearance of Docker Quickstart Terminal

For ubuntu 16.0.4 users, please refer this link

https://www.digitalocean.com/community/questions/how-to-check-for-docker-installation to check whether docker is installed successfully.

3. Tutorial Example

During this tutorial, it will use Keras Onet model as an example. For the window users, you need to type the commands by Docker Quickstart terminal or other related Docker tools, and for the ubuntu 16.0.4 users, you need to type the commands by ubuntu terminal.

3.1 Pull the docker image

Each time when there is a new version of linux command toolchain releasing, you need to pull the latest docker image.

The command to pull the latest image is:

docker pull kneron/toolchain:linux_command_toolchain

After finishing this command, the docker image is saved in your local side, and only when you want to update the image to the latest one, you need to connect to the Internet to repeat this command again. Otherwise, you can finish the following parts without the connection to the Internet.

To check whether the docker image is pulled successfully, type in this command:

```
docker image ls
```

If the image is pulled successfully, you will see the image, mrwhoami/kmc_docker:linux_command_toolchain in the docker image list.

3.2 Start the docker image

Then you can start the docker image you just pulled, and get a docker container to run the toolchain. When you start it, you need to configure a local folder as the one for communicating between your local environment and the container, let's call it as **Interactive Folder** Assume the absolute path of the folder you configure is absolue_path_of_your_folder.

And the start command is:

```
docker run -it --rm -v absolute_path_of_your_folder:/data1
kneron/toolchain:linux_command_toolchain
```

For example, if the absolute path of the path folder you configure is /home/kneron/Document/test_docker, and then the related command is

```
docker run -it --rm -v /home/kneron/Document/test_docker:/data1
kneron/toolchain:linux_command_toolchain
```

After running the start command, you'll enter into the docker container. Then, copy the example materials to the Interactive Folder by the following command:

```
cp -r /workspace/examples/* /data1/
```

3.3 Converter

3.3.1 Keras to ONNX

The command format for converting keras model to onnx model is:

python /workspace/onnx-keras/generate_onnx.py -o absolute_path_of_output_model_file absolute_path_of_input_model_file -O -C

For Onet example here, the detailed command is:

python /workspace/onnx-keras/generate_onnx.py -o /data1/onet-0.417197.onnx /data1/keras/onet-0.417197.hdf5 -O -C

For tiny_yolo_v3 example, the detailed command is:

python /workspace/onnx-keras/generate_onnx.py -o /data1/yolov3-tiny-224.h5.onnx /data1/keras/yolov3-tiny-224.h5 -O -C

There might be some warning log when running this problem, and you can check whether the convert works successfully by checking whether the onnx file is generated.

3.3.2 Tensorflow to ONNX

/workspace/scripts/tf2onnx.sh absolute_path_of_input_model_file absolute_path_of_output_onnx_model_file name_of_input_layer:0 name_of_output_layer:0

For the provided example model: mnist.pb

/workspace/scripts/tf2onnx.sh /data1/tensorflow/model/mnist.pb /data1/mnist.pb.onnx Placeholder:0 fc2/add:0

3.3.3 Pytorch to ONNX

python /workspace/scripts/pytorch2onnx.py absolute_path_of_input_model_file channel_number model_input_width model_input_height absolute_path_of_output_model_file

For the provided example model: resnet34.pth

python /workspace/scripts/pytorch2onnx.py /data1/pytorch/models/resnet34.pth 3 224 224 /data1/resnet34.onnx

3.3.4 Caffe to ONNX

python /workspace/onnx-caffe/generate_onnx.py -o absolute_path_of_output_onnx_model_file -w absolute_path_of_input_caffe_weight_file -n absolute_path_of_input_caffe_model_file

For the provided example model: mobilenetv2.caffemodel

python /workspace/onnx-caffe/generate_onnx.py -o /data1/mobilenetv2.onnx -w /data1/caffe/models/mobilenetv2.caffemodel -n /data1/caffe/models/mobilenetv2.prototxt

3.4 FpAnalyser and Batch-Compile

This part is the instructions for batch-compile, which will generate the binary file requested by firmware.

3.4.1 Fill the input parameters

Fill the simulator and emulator input parameters in the /data1/batch_compile_input_params.json in Interactive Folder. Please refer on the FAQ question 4 to fill the related parameters.

And the follow will give two examples for how to configure the batch_compile_input_params.json.

(1) tiny_yolo_v3

```
{
    "input_image_folder": ["/data1/caffe/images"],
    "img_channel": ["RGB"],
    "model_input_width": [224],
    "model_input_height": [224],
    "img_preprocess_method": ["yolo"],
    "input_onnx_file": ["/data1/yolov3-tiny-224.h5.onnx"],
    "command_addr": "0x30000000",
    "weight_addr": "0x40000000",
    "sram_addr": "0x50000000",
```

```
    "dram_addr": "0x60000000",
    "whether_encryption": "No",
    "encryption_key": "0x12345678",
    "model_id_list": [19],
    "model_version_list": [1]
}
```


(2) tiny_yolo_v3 and Onet


```
{
    "input_image_folder": ["/data1/caffe/images", "/data1/keras/n000645"],
    "img_channel": ["RGB", "L"],
    "model_input_width": [224, 48],
    "model_input_height": [224, 48],
    "img_preprocess_method": ["yolo", "kneron"],
    "input_onnx_file": ["/data1/yolov3-tiny-224.h5.onnx", "/data1/onet-0.417197.onnx"],
    "command_addr": "0x30000000",
    "weight_addr": "0x40000000",
    "sram_addr": "0x50000000",
    "dram_addr": "0x60000000",
    "whether_encryption": "No",
    "encryption_key": "0x12345678",
    "model_id_list": [19, 20],
    "model_version_list": [1, 1]
}
```


3.4.2 Running the programs

For running the compiler and ip evaluator:

cd /workspace/scripts && ./fpAnalyserBatchCompile.sh.x

And a folder called batch_compile will be generated in Interactive Folder, which stores the result of the fpAnalyer and batch-compile.

3.4.3 Get the result

In Interactive Folder, you'll find a folder called compiler, which contains the output files of the fpAnalyer and batch-compile. If you have questions for the meaning of the output files, please refer to the FAQ question 5.

4. FAQ

1. Fails in the step of FpAnalyser

When it shows the log "mystery", it means there are some customized layser in the model you input, which are not support now;

When it shows the log "start datapath analysis", you need to check whether you input the proper image pre-process parameters.

2. Other unsupported models

This version of SDK doesn't support the models in the following situations:

(1) Too large models

3. The functions KDP520 NPU supports

Layers/Modules	Functions/Parameters	Spec.
Convolution	Convolution kernal dimension:	1x1 up to 11x11
	Stride	1,2,4

	Padding:	0-15
	Depthwise Conv	Yes
	Deconvolution	Use Upsampling + Conv
Pooling	Max pooling 3x3	stride 1,2,3
	Max pooling 2x2	stride 1,2
	Ave Pooling 3x3	stride 1,2,3
	Ave Pooling 2x2	stride 1,2
	global ave pooling	Support
	global max pooling	Support
Activation	ReLu	Support
	Leaky ReLU	Support
	PReLU	Support
	ReLU6	Support
Other processing	Batch Normalization	Support
	Add	Support
	Concatenation	Support
	Dense/Fully Connected	Support
	Flatten	Support

3. How to configure the batch_compile_input_params.json?

By following the above instructions, the batch_compile_input_params.json will be saved in Interactive Folder.

Please do not change the parameters' names.

The parameters in batch_compile_input_params.json are:

(1) input_image_folder

The absolute path of input image folder for fpAnalyser. Since that batch-compile can compile more than one models together, the order of the input_imgae_folder is related to the order of input_onnx_file.

(2)img_channel

Options: L, RGB

The channel information after the input image is preprocessed. L means single channel. Input for fpAnalyer. Same as (1), the order of the img_channel is related to the order of input_onnx_file.

(3)model_input_width

The width of the model input size.Input for fpAnalyer. Same as (1), the order of the model_input_width is related to the order of input_onnx_file.

(4)model_input_height

The height of the model input size. Same as (1), the order of the model_input_height is related to the order of input_onnx_file.

(5)img_preprocess_method

Options: kneron, tensorflow, yolo, caffe, pytorch. Same as (1), the order of the img_preprocess_method is related to the order of input_onnx_file.

The image preprocess methods, input for fpAnlayer, and the related formats are following:

“kneron”: RGB/256 - 0.5,

“tensorflow”: RGB/127.5 - 1.0,

“yolo”: RGB/255.0

“pytorch”: (RGB/255. -[0.485, 0.456, 0.406]) / [0.229, 0.224, 0.225]

“caffe”(BGR format) BGR - [103.939, 116.779, 123.68]

(6)input_onnx_file

The abosolute path of the onnx file, which works as the input file for fpAnalyser. Since that the batch-compile can compile more than one models at a time. The order of the model in the array of input_onnx_file decides the model binary's order in all_models.bin

(7)command_addr

Address for command, input for compiler.

(8)weight_addr

Address for weight, input for compiler.

(9)sram_addr

Address for sram, input for compiler.

(10)dram_addr

Address for dram, input for compiler.

(11)whether_encryption

Option: Yes, No

Whether add encryption on the bin files generated by compiler, input for compiler.

(12)encryption_key

Encryption key for bin files, input for compiler.

(13)id_list

This is the ids for the models you input in the parameter input_onnx_file, and it will decide the order of the model in firmware. And the id_list you input should be the same as the one in SDK.

The supported model ids by SDK have been listed in chapter 5.6.2. If a new model will be supported by SDK, assign a unique id incrementally for it. Kneron reserves model ids from 0 to 1000. The ids after 1000 are for customers.

(14)version_list

This is the version number for the models you input in the parameter input_onnx_file. Currently there is no version checking mechanism in SDK. Version number 1 is used for all models.

4. What's the meaning of the output files of batch-compile?

The result of 3.4 FpAnalyser and Batch-Compile is generated at a folder called batch_compile at Interactive Folder, and it has two sub-folders called fpAnalyser and compiler.

In fpAnalyser subfolder, it has the folders with name format as input_img_txt_X, which contains the .txt files after image preprocessing. The index X is the related to the order of model file in FAQ qustion 4 (6), which means the folder input_img_txt_X is number X model's preprocess image text files.

In compile subfolder, it will have the following files: all_model.bin, fw_info.bin, temp_X_ioinfo.csv. The X is still the order of the models.

- all_model.bin and fw_info.bin are for firmware to use;

-temp_X_ioinf