# Tests

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Prebuilt NPU Demo | NPU Prebuilt Demo Usage | on VIM gives an error while executing ./detect\_demo\_fb 2 1080p.bmp  ./detect\_demo\_fb 0 1080p.bmp is OK |  |  |
| NPU SDK | NPU SDK Usage | executed on PC, compiled etc.... |  |  |
| NPU Demo Source Code | Application Source Code | on VIM3 board, compiled run... |  |  |
| Model Convert | convert and call your own model through NPU | compiled on PC and run on the VIM3 |  |  |
| NPU Performance | NPU Performance Interface Usage |  |  |  |
| NPU Operation Usage Times | each operation times usage | compiled on PC and run on the VIM3 |  |  |
| Tengine SDK | Tengine SDK Usage |  |  |  |
| Tengine Demo | Tengine SDK Usage |  |  |  |
| KSNN Usage | KSNN Usage | Installed on Khadas and tested an example, code conversion to be tested on PC. |  |  |

* Ubuntu
* npusdk1: yolov3: modified according to the Model Convert
* npusdk1a: yolov3: run again with Model again till “Compile And Get The Case Code” step
* npusdk1b: yolov3: to be run again with Model again with further steps
* npusdk2: mobilenet\_tf: default that generated many files
* npusdk3: yolov3: some further details
* npusdk4: mobilenet\_tf: basic
* npusdk5: mobilenet\_tf: original 2\_export\_case\_code trails
* npusdk6: mobilenet\_tf: edited 2\_export\_case\_code trails
* Khadas
* Applsource1: Application Source Code
* eachoptime
  + op\_test: using npusdk2, mobilenet\_tf
* eachoptime1 (it took me a lot of time to reprodrucing the these multiple text files. At the end I found that I was using Ctrl+R feature to scroll backward on the terminal and was using “NN” for export CNN\_PERF=1 instead of NN\_LAYER\_DUMP=1.)
  + op\_test1a: yolov3, with and without NN\_LAYER\_DUMP=1
  + op\_test3: no normal case files no \*.export.data, took nbg files instead. Have to see how to use the nbg case files.
  + op\_test3b: yolov3, with and without NN\_LAYER\_DUMP=1
  + op\_test5: mobilenet\_tf, with and without NN\_LAYER\_DUMP=1
  + op\_test6: mobilenet\_tf, with and without NN\_LAYER\_DUMP=1

# Some Parameters

--viv-sdk: SDK dir, if one of --pack-\* is given, please specify a folder containing the binary sdk of vSimulator.

--pack-nbg-unify: Pack binary graph for unify driver. (Optional) mutually exclusive with other --pack-\*. If this feature is enabled, two cases will be generated, a unify case and a nbg\_unify case. If no --pack-\* is given, only the unify case will be generated.

--optimize: Optimize the exporting network according to the specified hardware configuration path or configuration name.

(Optional). none – no optimization Default (Default) - If a configuration file or configuration name is not specified, it will use default export rule to export application code. If --pack-\* is given, please specify a configuration file path or configuration name instead of none or Default.

The difference between the two commands:

**Normal\_case:** When loading the model, online compilation may take a long time.

Android platform: supports running normal case directly

Linux platform: does not. If Linux platform is used, 1.

you need to push acuity\_tool\_xx/bin/vcmdtoos directory to the board data directory, and then set the environment variable: export VIVANTE\_SDK\_DIR=/data/vcmdtools. 2.you need to push buildroot\_sdk\_6462/buildroot\_sdk/build/so/drivers\_64\_exportdata or drivers\_32\_exportdata to the board data directory, and then set the environment variable: export LD\_LIBRARY\_PATH=/xxx/xxx/drivers\_64\_exportdata

**NBG case:** This step of on-line compilation has been completed on the PC. nb file can be loaded directly on the board, and the model loading speed is fast.

# Export Variables

Linux：

Get OP usage time:

export VIV\_VX\_DEBUG\_LEVEL=1

export CNN\_PERF=1

Dump result of each layer:

export NN\_LAYER\_DUMP=1, Each file represents a neural network layer.