



# **Guide of Tool Chain Installation and Use**

**ver 1.0.0**

## **content**

1	Arm Cross Compile Tool Chain Installation .....	2
1.1	Tool Chain Decompression.....	2
1.2	Tool Chain Installation.....	2
1.3	Environment Variables Settings.....	2
1.4	Tool Chain Installation Test .....	3
2	Model Conversion Tool Installation .....	3
2.1	Tool Chain Decompression.....	3
2.2	Configuration File Description .....	4
2.3	Tool chain Installation .....	4
2.4	Tool Chain Installation Test .....	4
3	Instructions for the use of tool chains.....	5
3.1	Arm Compile Steps.....	5
3.2	Model Conversion steps .....	5

## **Release History**

ver 1.0.0	2019/03/16	First version content
-----------	------------	-----------------------



# 1 ARM CROSS COMPILE TOOL CHAIN INSTALLATION

This SDK provides the arm-hisiv300-linux cross compile toolchain. The installation steps are as follows:

## 1.1 Tool Chain Decompression

The arm cross compile toolchain and its installer are located in directory *tools/arm\_toolchain*, enter this directory and decompress it. The commands are as follows:

```
cd tools/arm_toolchain/  
tar -xvf arm-hisi300-linux.tar.bz2
```

## 1.2 Tool Chain Installation

Enter the folder *arm-hisi300-linux/* after it's decompression and use the script to install the toolchain. The commands are as follows:

```
chmod +x cross.v300.install  
sudo ./cross.v300.install
```

## 1.3 Environment Variables Settings

If your computer is 64-bit, you need to install 32-bit compiler compatibility packages:

```
sudo aptitude install lib32z1  
sudo aptitude install lib32stdc++6-4.8-dbg
```

Import the path of the tool chain into environment variable, the path can be viewed as follows:



```
cd /opt/hisi-linux/x86-arm/arm-hisiv300-linux/target/bin  
pwd
```

Add the following statement at the end of the file *bashrc*:

```
vi ~/.bashrc  
export PATH=/opt/hisi-linux/x86-arm/arm-hisiv300-linux/target/bin:$PATH
```

Ensure that environmental variable settings take effect:

```
source ~/.bashrc
```

## **1.4 Tool Chain Installation Test**

Execute the following commands to view the version of the toolchain:

```
arm-hisiv300-linux-gcc -v
```

If “*gcc version 4.8.3 20131202 (prerelease) (Hisilicon\_v300)*” appears at the tail of the version description, the toolchain installation is successful.

# **2 MODEL CONVERSION TOOL INSTALLATION**

## **2.1 Tool Chain Decompression**

The model conversion tool and its installer are located in directory *tools/mv\_toolchain*, enter this directory and decompress it. The commands are as follows:

```
cd tools/mv_toolchain/  
tar -xvf model_conversion.tar.gz
```



## 2.2 Configuration File Description

Enter the folder after decompression and the development environment requirements document is as follows:

— *install-ncsdk.sh*

#script to install all packages and libs on your system, including caffe-ssd cpu version and tensorflow.

— *uninstall-ncsdk.sh* # Uninstall

— *ncsdk.conf* # ncsdk config file.

— *requirments.txt* # python3 dependencies.

— *requirments\_apt.txt* # ubuntu system dependencies.

## 2.3 Tool chain Installation

Use the script to install the toolchain. The commands are as follows:

*sudo ./install-ncsdk.sh*

The installation log will be written into *setup-logs*.

## 2.4 Tool Chain Installation Test

Execute the following commands to view the version of the toolchain:

*mvNCCompile -v*

If “*mvNCCompile v02.00, Copyright @ Movidius Ltd 2016*” appears at the head of the version description, the toolchain installation is successful. Use “*-h*” command to see the help message:

*mvNCCompile -h*

### 3 INSTRUCTIONS FOR THE USE OF TOOL CHAINS

#### 3.1 Arm Compile Steps

Developers need to use arm cross-compiler tool chain to compile their own projects, generate executable files and run it on the embedded board. In this SDK sample program, *Makefile* shows how to use arm cross compiler to generate executable files.

For example, enter folder *examples/detect/*, use “*make*” to compile and link, you will see the compile steps like follows:

```
“ arm-hisiv300-linux-g++ -c xxx.c -o xxx.o  
arm-hisiv300-linux-g++ -o xxx xxx.o -lpthread ../../libs/SIM_CAM_lib.a ”
```

*xxx.c* is user-developed programs, *xxx.o* is object-file, *xxx* is the final executable file.

#### 3.2 Model Conversion steps

Developers need to use *mvNCCompile* tool chain to generate a model file which can be read and run on Movidius. This model file could be named as “*graph*” and it contains the structure and weight information of the neural network model trained by developers.

The often-used usage and parameters of *mvNCCompile* are as follows:

<b>-h</b>	# Show help
<b>-w WEIGHTS</b>	# Model weight (Caffe only), *.caffemodel*
<b>-in INPUTNODE</b>	# Name of input node, default: Caffe: *data*, TF: *input*
<b>-on OUTPUTNODE</b>	# Name of output node, default: Caffe: Last layer, TF:
<b>-o OUTFILE</b>	# Output path of SimCam graph
<b>-s NSHAVES</b>	# Number of Movidius Shave, recommend 6, default 1



For example, enter folder *examples/convert/*, use script *compile.sh* to compile, you will see the compile steps like follows:

***“mvNCCompile xxx.prototxt -w yyy.caffemodel -o graph -s 6 ”***

*xxx.prototxt* is the structure file of network, *yyy.caffemodel* is the weight file of network.

Document ***“Train\_Model.pdf”*** describes how to modify and generate these two files.

Document ***“SIMCAM\_API.pdf”*** describes how to read and run *graph* on board.

In folder *examples/models/*, we provide other models for different recognition.

***Labelmap.prototxt*** describes the classifications of the model.