Qualcomm ThunderSoft®

SNPE Training Part 1

Aaron Yue

2021

Contents

Review

Deep Learning Basics

- Deep learning has two phases
  - Training phase – Process for machine to learn and optimize a model from data
    - Collecting and preparing data
    - Creating a model
    - Training the model
  - Inference phase – Use trained model to predict outcomes from new observations in an efficient way
    - Evaluate model
    - Improving the performance

Frameworks of Deep Learning

- Frameworks of deep learning help developers quickly train and deploy a deep learning model, they provide a set of commonly used function libraries and class libraries for deep learning, and builds a deep network model just like "building blocks."

Framework	Author	Year	Open source	license	Written in	Interface
Caffe	Berkeley	2013	Yes	BSD	C++	Python, MATLAB, C++
TensorFlow	Google	2015	Yes	Apache 2.0	C++, Python, CUDA	Python, C/C++, Java, Go, etc.
MXNet	Apache Software Foundation	2015	Yes	Apache 2.0	Python, C, C++, CUDA	C++, Python, Java, Go, R, etc.
PyTorch	Facebook	2016	Yes	BSD	Python, C, C++, CUDA	Python, C++
Torch	Ronan Collobert, et al.	2002	Yes	BSD	C, Lua	Lua, C
Theano	University of Montreal	2007	Yes	BSD	Python	Python
MATLAB	MathWorks	-	No	-	C, C++, Java, MATLAB	MATLAB
Keras	François Chollet	2015	Yes	MIT license	Python	Python, R
CNTK	Microsoft Research	2016	Yes	MIT license	C++	Python, C++

Inference engine of Deep Learning

- Unlike the deep learning frameworks, which focus mainly on training and deploying on CPU and GPU devices, the inference engine emerges to help models to run on heterogeneous computing platforms (including servers and PCs, mobile devices, DSPs). Tools such as performance optimization, model encryption, and memory analysis are provided.

Inference engine	Author	Year	CPU	GPU	DSP	NPU / VPU
SNPE	Qualcomm	2016	1	1	1	1
TensorRT	NVIDIA	2016	0	1	0	0
OpenVINO	Intel	2018	1	1	0	1
MACE	Xiaomi	2018	1	1	0.5	0
TensorFlow Lite	Google	2017	1	0.5	0	0
Core ML	Apple	2018	1	1	0	1
ONNX	Microsoft/Facebook/Amazon	2017	1	1	0	0

- SNPE supports for a wide range of heterogeneous hardware.
- SNPE can run on embedded devices like cell phone with high performance.

Hardware Preparation : AI Kit

Thundercomm AI Kit Overview

845 SOM

+ IO Board

AI Kit

- Thundercomm AI Kit aims to help developers to evaluate, verify and develop their on-device AI products on SDA845 AI platform, providing AI algorithm SDK, demo applications, tools and cloud service support.

- Target Users: Software developers interested in applications, algorithms or solutions

Thundercomm AI Kit Key Features

Key Features

- Powerful computing platform:
  - CPU: Qualcomm SDA845 processor, 8x Qualcomm® Kryo™ 385 @ 2.8GHz
  - GPU: Adreno 630 GPU
  - DSP: Hexagon 685 DSP
  - New vector engine and heterogeneous computing(CPU, GPU, DSP) for AI
  - ISP: Qualcomm Spectra™ 280 image sensor processor,dual 14-bit ISPs
  - VPU: 4k@60fps or 720@480fps video capture
  - Display: 4K Ultra HD@60fps
- Powerful Expandability
  - Multiple visual solution: built-in MIPI Ultra HD camera, USB 3.0 camera, IP camera(RTSP)
  - Rich interface: 3x USB3.0, Ethernet, Type C, HDMI, SD Card, Micro USB, WiFi / BT

Thundercomm AI Kit Key Features(cont.)

Key Features

- Multi OS Support
  - Android O, Linux
- Optimized Algorithm SDK
  - Face Recognition
  - Age Detection
  - Emotion Detection
  - Gender Detection
  - Object Detection
- Hardware accelerated face detection
- Accelerated electronic image stabilization

Thundercomm AI Kit Hardware

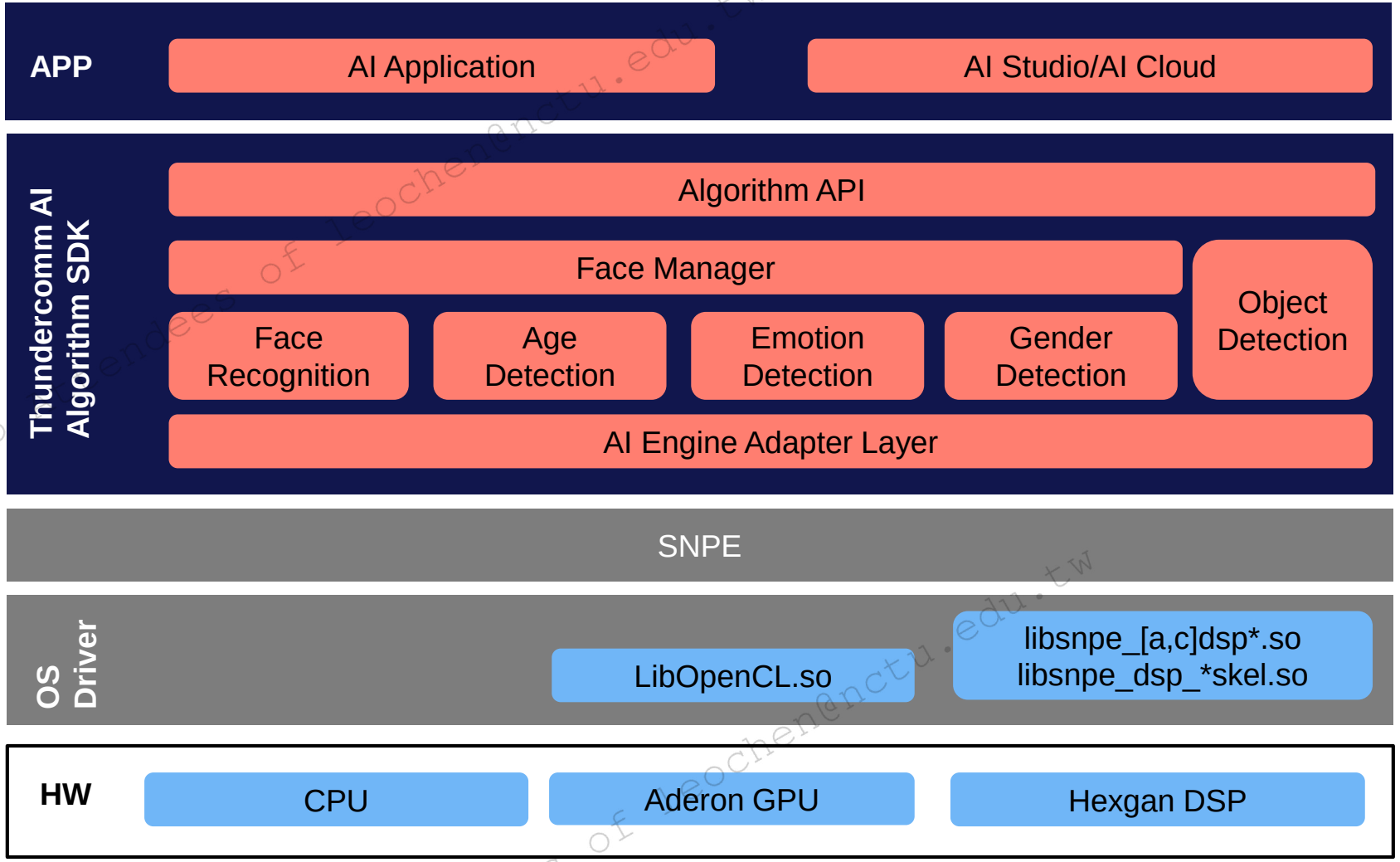
Thundercomm AI Kit Hardware SPEC

Module	Item	Spec
Computing	CPU	Qualcomm Snapdragon 845, Qualcomm® Kryo™ 385 @ 2.8GHz
	GPU	Adreno 630 GPU
	DSP	Hexagon 685 DSP
	ISP	Qualcomm Spectra™ 280 image sensor processor
Memory	RAM	8GB LPDDR4X
	ROM	64GB UFS2.1
	SD Card	Support MicroSD Card
Camera	Built in Camera	8MP, FOV 120 degree;
	External Camera	USB3.0 camera (accessory, not included in release)
Display	HDMI	1x Micro HDMI, 4k Ultra HD
	Type C	4k Ultra HD
Video	VPU	Support for HDR10, HLG, and H.265 (HEVC) 4k@60fps video capture
Audio	Speaker	1x speaker
	Audio out	1x Headphone
	Microphone	1x microphone
Wireless Connectivity	WIFI	2.4G/5G,802.11 a/b/g/n/ac 2x2 MIMO
Wired Connectivity	USB	3x USB3.0 host, 1x 3.0 Type-C
	Ethernet	1x Gigabit.
Debug Port	Micro USB	1x Micro USB (connecting to serial port for debug purpose)
Indicator	LED	RGB Led: Power status, WIFI status, Ethernet status;
Input	Buttons	Power key,Volume up/down key, Camera snapshot key
Sensors	9-axis sensor	Gyroscope & accelerometer & geomagnetic sensor
Power	Power charging	12v DC

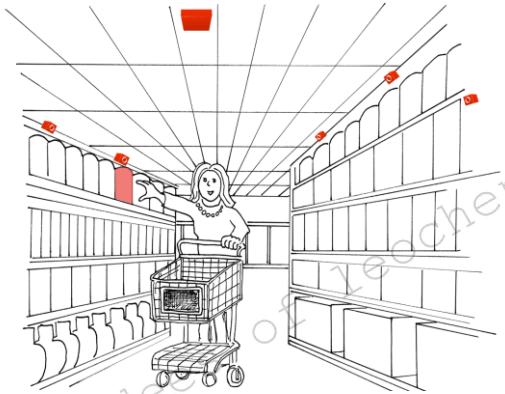
AI Kit SW SPEC on Android

Module	Feature
OS	Android O
AI Framework	Tensorflow, Caffe/Caffe2 ,ONNX,SNPE, Android NN
AI Algorithm SDK	Face Detection & Face Recognition & Emotion Detection & Age Detection & Gender Detection
	Object Detection algorithm
AI Sample Application	Face Recognition & Emotion Detection & Age Detection & Gender Detection algorithm demo
	Objection Detection algorithm demo
Camera	Support USB3.0 Camera, 1080p@30fps, Accessing USB camera with standard camera API in Android SDK
	Ultra HD camera preview, recording and snapshot
	RTSP Client to support IP camera
Graphic	OpenGLES3.2, OpenCL2.0 full

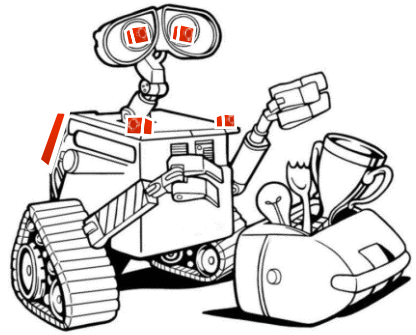
AI Kit SW Architecture on Android



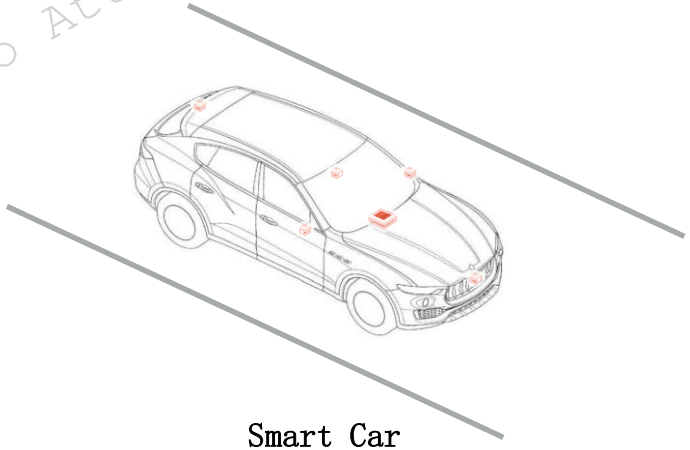
Scenario of Thundercomm AI Kit



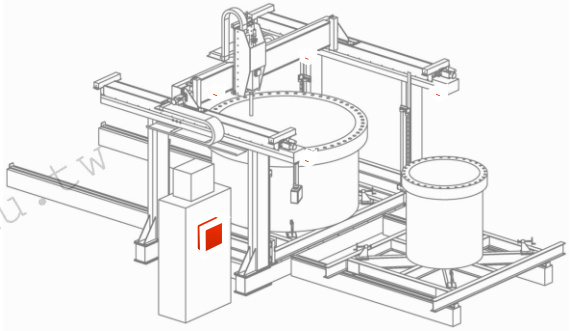
Smart Retail



Smart Robot



Smart Car



Smart Factory

More details, please refer to the AIKIT website: https://www.45smart.com/en/

Software Preparation :
Part 1 SNPE SDK Development Environment Setup

SNPE Development Environment Overview

- Currently the SNPE SDK development environment is limited to Ubuntu, specifically version 14.04, 16.04 and even 18.04.
- x86 architecture support has moved to Ubuntu 16.04 OS from Ubuntu 14.04 after the 1.27.0 release. The setup flow in the following chapters has been tested on Ubuntu 18.04. The current latest version SNPE SDK is v1.35.0 and here it takes SNPE SDK v1.35.0 as an example to show how to setup SNPE SDK development environment.
- The SDK requires either Caffe, Caffe2, ONNX or TensorFlow. This document shows both setup of Caffe and setup of TensorFlow.
- Python 2.7 or Python 3.4 is required for model converting and SNPE SDK environment deployment.
- Android NDK (android-ndk-r17c-linux-x86) is optional and only required to build the native CPP example that ships with the SDK.
- Android SDK (SDK version 23 or above and build tools version 23.0.2 or above) is optional and only required to build the Android APK that ships with the SDK.

SNPE SDK Download Methods

- The SNPE SDK is available to download from CreatePoint
  - 1.Navigate to the Tools section.
  - 2.Filter by “Neural Processing Engine”.
  - 3.Download the latest version.
- The SNPE SDK also can be downloaded from Qualcomm Developer site

  https://developer.qualcomm.com
- SNPE documentation is packaged with the SDK in

  <workspace>/snpe-<version>/doc/html/index.html
- For a full list of prerequisites, dependencies and setup instructions, refer to the documentation packaged with the SDK in

  <workspace>/snpe-<version>/doc/html/setup.html

SNPE SDK Setup

Getting Inception v3

- The Inception v3 TensorFlow model file, and sample image files are prepared for the running TensorFlow classification. The script requires a directory path to the Inception v3 assets (zip file). The script can also optionally download the Inception v3 archive.

- The Inception v3 assets are listed below:
  - inception_v3_2016_08_28_frozen.pb.tar.gz - https://storage.googleapis.com/download.tensorflow.org/models/inception_v3_2016_08_28_frozen.pb.tar.gz

SNPE SDK Setup (cont.)

- Running "python $SNPE_ROOT/models/inception_v3/scripts/setup_inceptionv3.py -h" will show the usage description.
- usage: $SNPE_ROOT/models/inception_v3/scripts/setup_inceptionv3.py [-h] -a ASSETS_DIR [-d] [-r RUNTIME] [-u]
- Prepares the inception_v3 assets for tutorial examples.
- required arguments:
  - -a ASSETS_DIR, --assets_dir ASSETS_DIR directory containing the inception_v3 assets
- optional arguments:
  - -d, --download Download inception_v3 assets to inception_v3 example directory
  - -r RUNTIME, --runtime RUNTIME Choose a runtime to set up tutorial for. Choices: cpu, gpu, dsp, aip, all. 'all' option is only supported with --udo flag
  - -u, --udo Generate and compile a user-defined operation package to be used with inception_v3. Softmax is simulated as a UDO for this script.

SNPE SDK Setup (cont.)

- Run the script to download model and set up to run on CPU:
  - python $SNPE_ROOT/models/inception_v3/scripts/setup_inceptionv3.py -a ~/tmpdir –d
- After the script is complete the prepared Inception v3 assets are copied to the $SNPE_ROOT/models/inception_v3 directory, along with sample raw images, and converted SNPE DLC files with additional optimizations as applicable.

SNPE SDK Setup (cont.)

Getting AlexNet

- AlexNet Caffe model files (prototxt and caffemodel), and sample image files are prepared for running Caffe model. The script requires a directory path to the AlexNet assets. The script can also optionally download the AlexNet assets.

- The AlexNet assets are listed below:

  - deploy.prototxt - https://raw.githubusercontent.com/BVLC/caffe/master/models/bvlc_alexnet/deploy.prototxt

  - bvlc_alexnet.caffemodel - http://dl.caffe.berkeleyvision.org/bvlc_alexnet.caffemodel
    caffe_ilsvrc12.tar.gz - http://dl.caffe.berkeleyvision.org/caffe_ilsvrc12.tar.gz

  - caffe_ilsvrc12.tar.gz - http://dl.caffe.berkeleyvision.org/caffe_ilsvrc12.tar.gz

SNPE SDK Setup (cont.)

- Usage: $SNPE_ROOT/models/alexnet/scripts/setup_alexnet.py [-h] -a ASSETS_DIR [-d]
- Prepares the AlexNet assets for tutorial examples.
- required arguments:
  - -a ASSETS_DIR, --assets_dir ASSETS_DIR directory containing the AlexNet assets
- optional arguments:
  - -d, --download Download AlexNet assets to AlexNet assets directory
- E.g.
  - python $SNPE_ROOT/models/alexnet/scripts/setup_alexnet.py -a ~/tmpdir -d
- After the script is complete the prepared AlexNet assets are copied to the $SNPE_ROOT/models/alexnet directory, along with sample raw images, and converted SNPE DLC files.
- More details about environment setup, please refer to the document:
  - <workspace>/snpe-<version>/doc/html/setup.html

SNPE SDK Folder Structure

```
├── android
│   ├── platform-validator.aar
│   ├── psnpe-release.aar
│   └── snpe-release.aar
├── benchmarks
│   ├── alexnet_sample.json
│   ├── common_utils
│   ├── config_help.json
│   ├── snpe_bench.py
│   └── snpebm
├── bin
│   ├── aarch64-android-clang6.0
│   ├── aarch64-linux-gcc4.9
│   ├── aarch64-oe-linux-gcc8.2
│   ├── arm-android-clang6.0
│   ├── arm-linux-gcc4.9sf
│   ├── arm-oe-linux-gcc8.2hf
│   ├── check_python_depends.sh
│   ├── dependencies.sh
│   ├── envsetup.sh
│   └── x86_64-linux-clang
├── doc
│   └── html
├── examples
│   ├── android
│   ├── NativeCpp
│   └── Python
├── include
│   └── zdl
```

```
├── lib
│   ├── aarch64-android-clang6.0
│   ├── aarch64-linux-gcc4.9
│   ├── aarch64-oe-linux-gcc8.2
│   ├── arm-android-clang6.0
│   ├── arm-linux-gcc4.9sf
│   ├── arm-oe-linux-gcc8.2hf
│   ├── dsp
│   ├── python
│   └── x86_64-linux-clang
├── LICENSE.txt
├── models
│   ├── alexnet
│   ├── inception_v3
│   └── mnist
├── NOTICE.txt
├── REDIST.txt
├── ReleaseNotes.txt
└── share
    ├── dlcviewer
    └── SnpeUdo
```

Software Preparation :
Part 2 SNPE Application Development Tools

SNPE Application Development Tools List

- The following tools are recommended for SNPE application development.
  - a) Android Studio 3.4.1 or later
  - b) Android SDK 27
  - c) Android NDK r17c
  - d) OpenCV 4.1.0

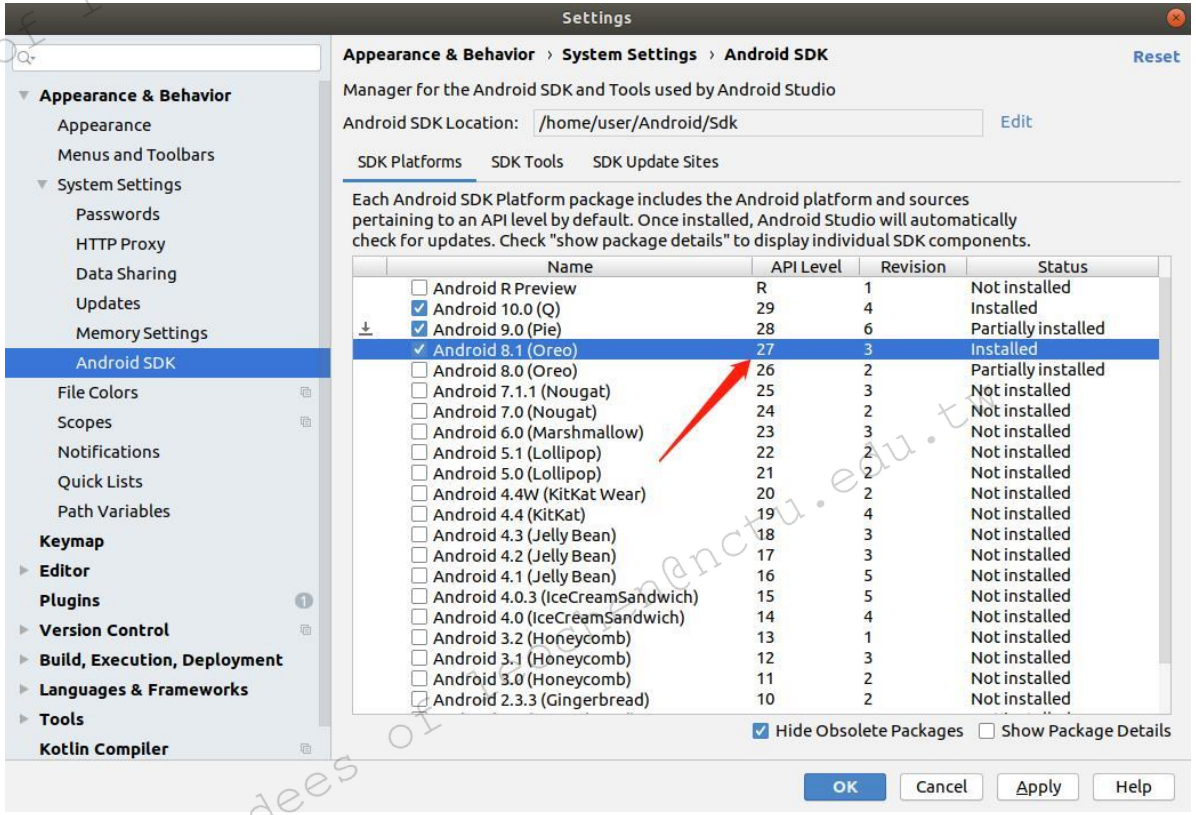
Download Android Studio

- Android studio can be downloaded from the following websites:
  - https://developer.android.google.cn/studio#downloads
  - http://www.android-studio.org/index.php/download
- Choose the version you want to download.
  - E.g. android-studio-ide-183.5522156-linux.tar.gz
- Download link:
  - https://dl.google.com/dl/android/studio/ide-zips/3.5.3.0/android-studio-ide-191.6010548-linux.tar.gz
- Decompress the package:
  - tar -zxvf android-studio-ide-183.5522156-linux.tar.gz
- Run Android studio:
  - cd PATH_OF_ANDROID_STUDIO/bin/
  - ./studio.sh

Download Android SDK 27

Now, Android studio is opened. Click File -> Settings... -> Appearance & Behavior -> System Settings -> Android SDK.

- Click Edit Choose a location you want to save the SDK assets
- Tick API Level 27 and then Apply.



Download and Setup Android NDK r17c

- You will need the Android NDK to build the Android C++ executable. The tutorial assumes that you can invoke 'ndk-build' from the shell.
- Android NDK r17c is recommended.
- You can download Android NDK by the method below:

  a) https://developer.android.google.cn/ndk/downloads/

  b) Press NDK Revision History and find relative version NDK package

  c) Agree the terms and conditions appear

  d) Choose the specific version to download

- Click android-ndk-r17c-linux-x86_64.zip to download.

Android NDK, Revision 17c (June 2018)

Platform	Package	Size (Bytes)	SHA1 Checksum
Windows 32-bit	android-ndk-r17c-windows-x86.zip	608358310	5bb25bf13fa494ee6c3433474c7aa90009f9f6a9
Windows 64-bit	android-ndk-r17c-windows-x86_64.zip	650626501	3e3b8d1650f9d297d130be2b342db956003f5992
Mac OS X	android-ndk-r17c-darwin-x86_64.zip	675061485	f97e3d7711497e3b4faf9e7b3fa0f0da90bb649c
Linux 64-bit (x86)	android-ndk-r17c-linux-x86_64.zip	709387703	12cacc70c3fd2f40574015631c00f41fb8a39048

Download and Setup Android NDK r17c (cont.)

- E.g. the download link is:
  - https://dl.google.com/android/repository/android-ndk-r17c-linux-x86_64.zip
- Extract the package downloaded:
  - $unzip android-ndk-r17c-linux-x86_64.zip
  - Directory android-ndk-r17c is generated
- Add NDK binaries path to the environment
  - $export NDK_PATH=Path to android-ndk-r17c
  - $export PATH=$PATH:$NDK_PATH
- Test whether the ndk-build environment has been installed normally.
  - $which ndk-build
  - The path to android-ndk-r17c will be printed

Download and Setup OpenCV Android Libraries

- An excellent computer vision tool called OpenCV which provides variety of image processing methods is recommended here. The android application demo also depends on it.
- Please follow this link below to download OpenCV libraries, and the version 4.1.0 has been tested successfully:
  - https://opencv.org/releases.html
  - Click Android and then it jumps to another website with downloading the OpenCV package.
  - Download link: E.g. https://jaist.dl.sourceforge.net/project/opencvlibrary/4.1.0/opencv-4.1.0-android-sdk.zip

- Unzip the package downloaded:
  - $unzip opencv-4.1.0-android-sdk.zip

Add OpenCV and SNPE Libraries, Head Files Paths in CMakeLists.txt

```
set(OpenCV_DIR /home/USER_NAME/workspace/tools/opencv/OpenCV-android-sdk_410/sdk/native/jni)
set(SNPE_LIB ${CMAKE_SOURCE_DIR}/src/main/jniLibs)
find_package(OpenCV 4.1.0 REQUIRED)
message(STATUS "OpenCV library status:")
message(STATUS "    version: ${OpenCV_VERSION}")
message(STATUS "    libraries: ${OpenCV_LIBS}")
message(STATUS "    include path: ${OpenCV_INCLUDE_DIRS}")
include_directories(
        ${CMAKE_SOURCE_DIR}/src/main/cpp/include
        ${CMAKE_SOURCE_DIR}/src/main/cpp/Classification
        #SNPE head files path
        ${CMAKE_SOURCE_DIR}/src/main/cpp/thirdparty/zdl
        #OpenCV head files path
        ${OpenCV_INCLUDE_DIRS})
target_link_libraries( # Specifies the target library.
        native-lib
        libSNPE                    #SNPE libs needed
        ${OpenCV_LIBS}    #OpenCV libs needed
```

An Object Detector Demo

Converting Tensorflow MobilenetSSD Model to DLC Format

Tensorflow MobilenetSSD model

Tensorflow Mobilenet SSD frozen graphs come in a couple of flavors. The standard frozen graph and a quantization aware frozen graph. The following example uses a quantization aware frozen graph to ensure accurate results on the SNPE runtimes.

Prerequisites

- The quantization aware model conversion process was tested using Tensorflow v1.14 however other versions may also work. The CPU version of Tensorflow was used to avoid out of memory issues observed across various GPU cards during conversion.

Setup the Tensorflow Object Detection Framework

The quantization aware model is provided as a TFLite frozen graph. However SNPE requires a Tensorflow frozen graph (.PB). To convert the quantized model, the object detection framework is used to export to a Tensorflow frozen graph.

- mkdir ~/tfmodels
- cd ~/tfmodels
- git clone https://github.com/tensorflow/models.git
- Checkout a tested object detection framework commit (SHA)
- git checkout ad386df597c069873ace235b931578671526ee00

Follow these installation instructions to setup the Tensorflow object detection framework:

- https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/installation.md

Tensorflow MobilenetSSD model(cont.)

- Download the quantization aware model
- A specific version of the Tensorflow MobilenetSSD model has been tested: ssd_mobilenet_v2_quantized_300x300_coco_2019_01_03.tar.gz
  - wget http://download.tensorflow.org/models/object_detection/ssd_mobilenet_v2_quantized_300x300_coco_2019_01_03.tar.gz
- After downloading the model extract the contents to a directory.
  - tar xzvf ssd_mobilenet_v2_quantized_300x300_coco_2019_01_03.tar.gz
- Export a trained graph from the object detection framework
- Follow these instructions to export the Tensorflow graph:
  - https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/exporting_models.md
- or modify and execute this sample script
- Create this file, export_train.sh, using your favorite editor. Modify the paths to the correct directory location of the downloaded quantization aware model files.

Tensorflow MobilenetSSD model(cont.)

export_train.sh

- #!/bin/bash
- INPUT_TYPE=image_tensor
- PIPELINE_CONFIG_PATH=<path_to>/ssd_mobilenet_v2_quantized_300x300_coco_2019_01_03/pipeline.config
- TRAINED_CKPT_PREFIX=<path_to>/ssd_mobilenet_v2_quantized_300x300_coco_2019_01_03/model.ckpt
- EXPORT_DIR=<path_to>/exported
- pushd ~/tfmodels/models/tfmodels/research
- python object_detection/export_inference_graph.py \
- --input_type=${INPUT_TYPE} \
- --pipeline_config_path=${PIPELINE_CONFIG_PATH} \
- --trained_checkpoint_prefix=${TRAINED_CKPT_PREFIX} \
- --output_directory=${EXPORT_DIR}
- popd

Make the script executable

- chmod u+x export_train.sh

Run the script

- ./export_train.sh

This should generate a frozen graph in <path_to>/exported/frozen_inference_graph.pb

Convert the frozen graph using the snpe-tensorflow-to-dlc converter.

- snpe-tensorflow-to-dlc --graph <path_to>/exported/frozen_inference_graph.pb -i Preprocessor/sub 1,300,300,3 --out_node detection_classes --out_node detection_boxes --out_node detection_scores --dlc mobilenet_ssd.dlc --allow_unconsumed_nodes

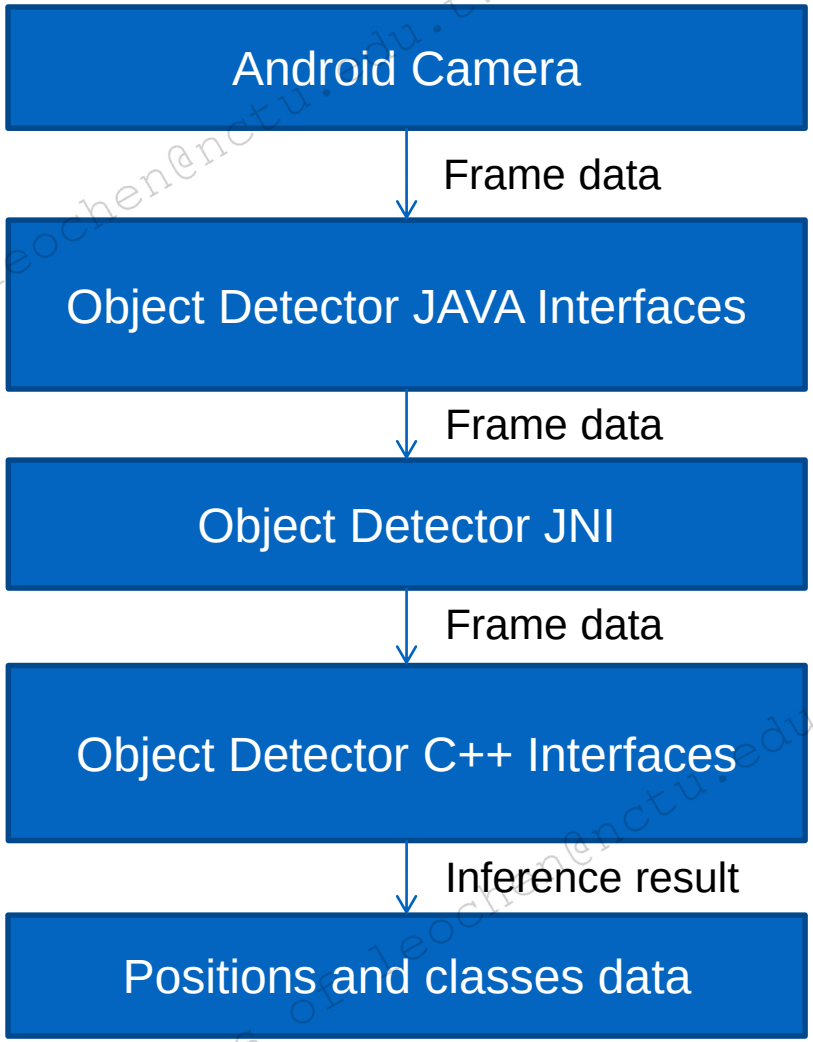
Tensorflow MobilenetSSD model(cont.)

- After SNPE conversion you should have a mobilenet_ssd.dlc that can be loaded and run in the SNPE runtimes.
- The output layers for the model are:
  - Postprocessor/BatchMultiClassNonMaxSuppression
  - add
- The output buffer names are:
  - (classes) detection_classes:0 (+1 index offset)
  - (classes) Postprocessor/BatchMultiClassNonMaxSuppression_classes (0 index offset)
  - (boxes) Postprocessor/BatchMultiClassNonMaxSuppression_boxes
  - (scores) Postprocessor/BatchMultiClassNonMaxSuppression_scores

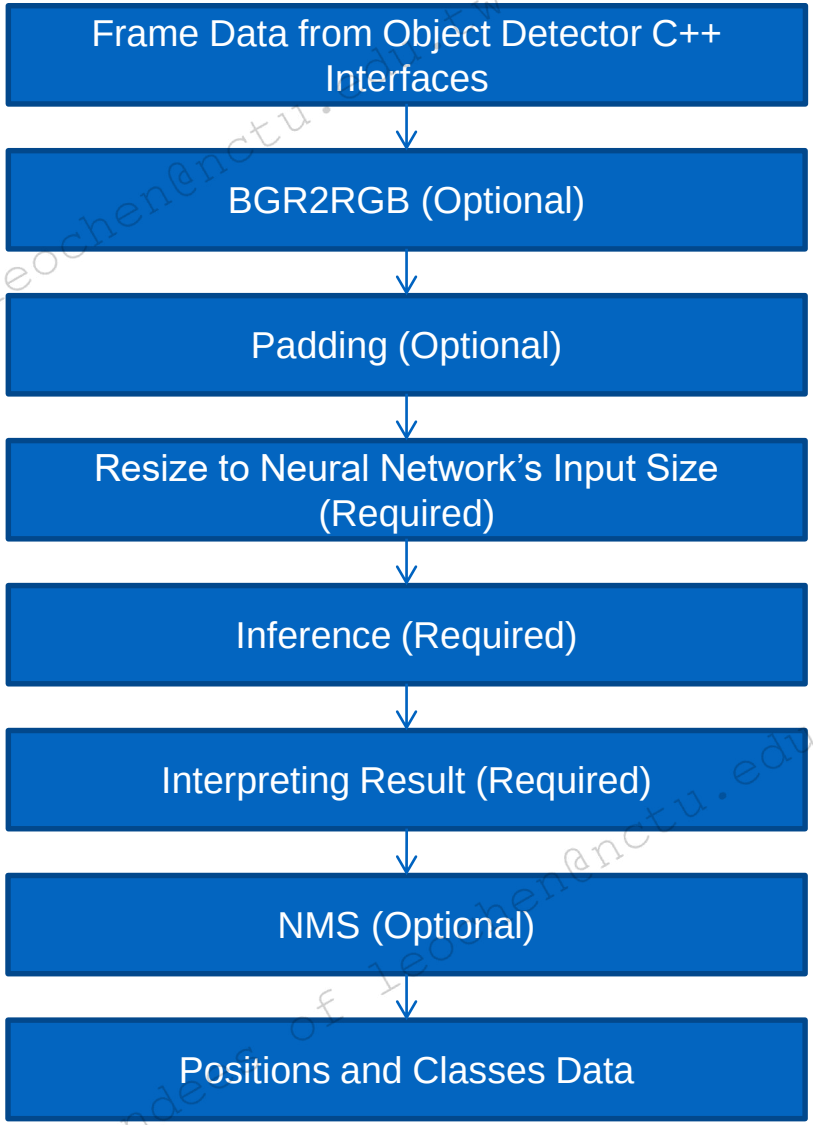
AI demos - Object Detector Demo

Run Demo

Object Detection Demo Flow



Data Flow during Preprocessing, Inference and Postprocessing



Run Demo

Solve AI Problem Skills

Choose Different Runtimes for Different Algorithms

- Different runtimes characteristics:
  - CPU --> High Accuracy, may be very slow and can’t meet requirments.
  - GPU --> High Accuracy.
  - DSP --> Lower Accuracy than GPU / CPU but generally the accuracy won’t descend much.
- Key points to focus:
  - Choose different runtimes combinations to find a best one.
  - CPU load for different conditions. (Snapdragon Profiler / Android Profiler)

Cloud / PC AI vs Embedding AI

- Difference:
- 1. Cloud / PC may have better hardware resource while the hardware resource is limited for embedding side.
- 2. Embedding AI can do things without network connection.

- Optimizing directions:
- 1. Try optimizing networks to reduce amount of parameters.
- 2. Use heterogeneous processor to inference speed to meet requirement.

Homework

Test inference speed with different Runtimes

Runtimes	Time Consuming Per Time (ms)	FPS(Frame Per Second)
CPU		
GPU		
DSP		

Accuracy changes?

Test inference speed with different Runtimes

Runtimes	Time Consuming Per Time (ms)	FPS(Frame Per Second)
CPU	220	4.55
GPU	46.7	21.4
DSP	13.0	76.9

Accuracy changes?

QDN (Qualcomm Developer Networks)

- A) SNPE SDK
- https://developer.qualcomm.com/software/qualcomm-neural-processing-sdk
- B) SNPE index page
- https://developer.qualcomm.com/docs/snpe/index.html
- C) SNPE release notes
- https://developer.qualcomm.com/software/qualcomm-neural-processing-sdk/release-notes
- D) Different versions of SNPE SDK to download
- https://developer.qualcomm.com/software/qualcomm-neural-processing-sdk/tools
- E) The Forum or raise a case in the website of Qualcomm CreatePoint
- https://developer.qualcomm.com/forums/software/qualcomm-neural-processing-sdk

Q & A

ThunderSoft®

Edge Computing & On-Device AI Enabler

Qualcomm

Thank You