

Movidius™ Neural Compute Stick

Getting Started Guide

July 2017 – Revision 1.1

You may not use or facilitate the use of this document in connection with any infringement or other legal analysis concerning Intel products described herein. You agree to grant Intel a non-exclusive, royalty-free license to any patent claim thereafter drafted which includes subject matter disclosed herein.

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest Intel product specifications and roadmaps.

The products described may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Copies of documents which have an order number and are referenced in this document may be obtained by calling 1-800-548-4725 or by visiting: <http://www.intel.com/design/literature.htm>

Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Learn more at <http://www.intel.com/> or from the OEM or retailer.

Intel, Movidius and the Intel logo are trademarks of Intel Corporation in the U.S. and/or other countries.

*Other names and brands may be claimed as the property of others.

Copyright © 2017, Intel Corporation. All rights reserved.

Contents

Terminology.....	4
Reference Documents	4
1.0 Introduction.....	5
1.1 Installation and configuration.....	6
1.1.1 Required materials.....	6
1.1.2 Connecting the NCS to a development computer.....	6
1.1.3 The user forum.....	7
1.1.4 Downloading the Movidius™ SDK	8
1.2 Install and Verify the Movidius Neural Compute Toolkit.....	9
1.3 Install and Verify the Movidius NC API.....	12
1.4 Using the Movidius™ NC Toolkit	13
1.4.1 Make example01.....	14
1.4.2 Make example02.....	15
1.4.3 Make example03.....	16
1.5 Using the Movidius NC API.....	17
1.5.1 Verify the directory structure and contents.....	18
1.6 Deploying on a Raspberry Pi	20
1.6.1 Running the stream_infer.py Example on Raspberry Pi.....	20
2.0 Appendix A: Toolkit Utilities and Examples	22
2.1 Utilities.....	22
2.1.1 Movidius™ Neural Compute compiler.....	22
2.1.2 Movidius™ Neural Compute checker.....	23
2.1.3 Movidius™ Neural Compute profiler	23
2.2 Examples.....	24
2.2.1 top_5_over_a_dataset.....	24
3.0 Appendix B: API Examples.....	25
3.1 C Examples.....	25
3.1.1 ncs-check	25
3.1.2 ncs-threadcheck	26
3.1.3 ncs-fullcheck	27
3.2 Python3 Examples	28
3.2.1 ncs_camera.....	28
3.2.2 stream_infer.....	29
3.2.3 age_gender_classification.....	29
3.2.4 age_gender_example	29

Revision History

Date	Revision	Description
7/19/2017	1.0	Updates for SDK Gold release 1.07.06
7/24/2017	1.1	Updates for SDK 1.07.07 – Raspberry Pi

Terminology

The following table provides the meaning of the abbreviations mentioned in this document, as well as some definitions for some specific terms.

Term	Description
API	Application programming interface
Caffe	A deep learning framework used to develop networks that can be compiled to run on the NCS
CNN	Convolutional neural network
Debian®-based Linux* OS	An Operating System (OS) that uses the Linux* kernel and accepts precompiled packages as a way to install user applications.
Host	System that the NCS is connected to
Inference	The act of comparing input to a network knowledge base, whereon a subject's attributes can be inferred
NCS	Neural Compute Stick
NCS SDK	A software package that contains the Toolkit and API for the NCS
Toolkit	A software used to compile and tune networks for the NCS
VPU	Visual processing unit

Reference Documents

Visit developer.movidius.com for additional documentation and information.

1.0 Introduction

This guide outlines how to starting using the Movidius™ Neural Compute Stick (NCS).

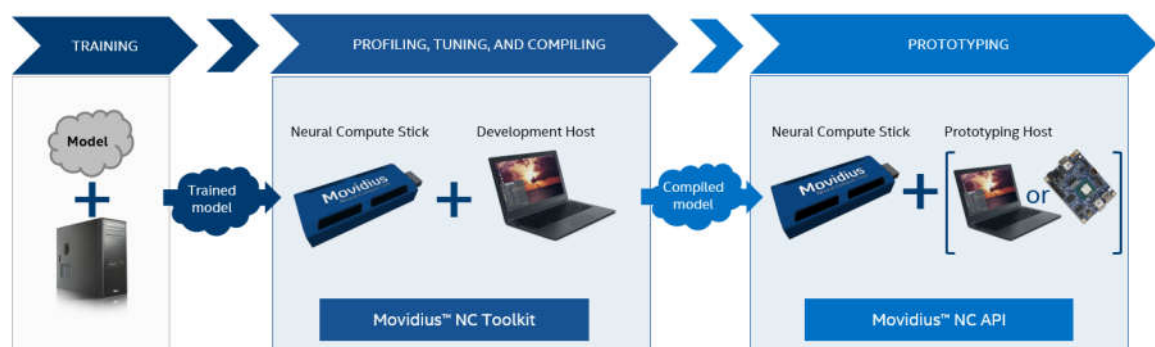
It provides an overview of the hardware, internet URLs to obtain software and general instructions to exercise various example and scenarios that will ensure your installation and configuration were completed successfully.

The NCS is used in two primary scenarios:

- Profiling, tuning, and compiling a Convolutional Neural Network on a development computer (host system) with the **Toolkit** elements of the **Movidius™ Neural Compute SDK**. In this scenario the host system is typically a desktop or laptop machine running Ubuntu 16.04 Desktop (x86, 64 bit)
- Prototyping a user application on a development computer (host system) which accesses the hardware of the NCS to accelerate CNN inferences. The **API** elements of the **Movidius™ Neural Compute SDK**. In this scenario the host system can be a developer workstation or any developer system that runs an operating system compatible with the API.

The following diagram shows the typical workflow for development with the NCS.

The training phase does not utilize the NCS hardware or SDK, while the subsequent phases of “profiling, tuning and compiling” and “prototyping” do require a workstation, NCS hardware and the accompanying **Movidius™ Neural Compute SDK**.



1.1 Installation and configuration

1.1.1 Required materials

- Movidius™ Neural Compute Stick
- Development computer running Ubuntu 16.04 LTS with an available USB 2.0 port
- USB Camera
- The latest Movidius™ Neural Compute SDK containing:
 - The latest Movidius™ NC Toolkit
 - The latest Movidius™ NC API

1.1.2 Connecting the NCS to a development computer

The NCS connects to the development computer over a USB 2.0 High Speed interface. Plug the NCS directly to a USB port on your development computer or into a powered USB hub that is plugged into your development computer.

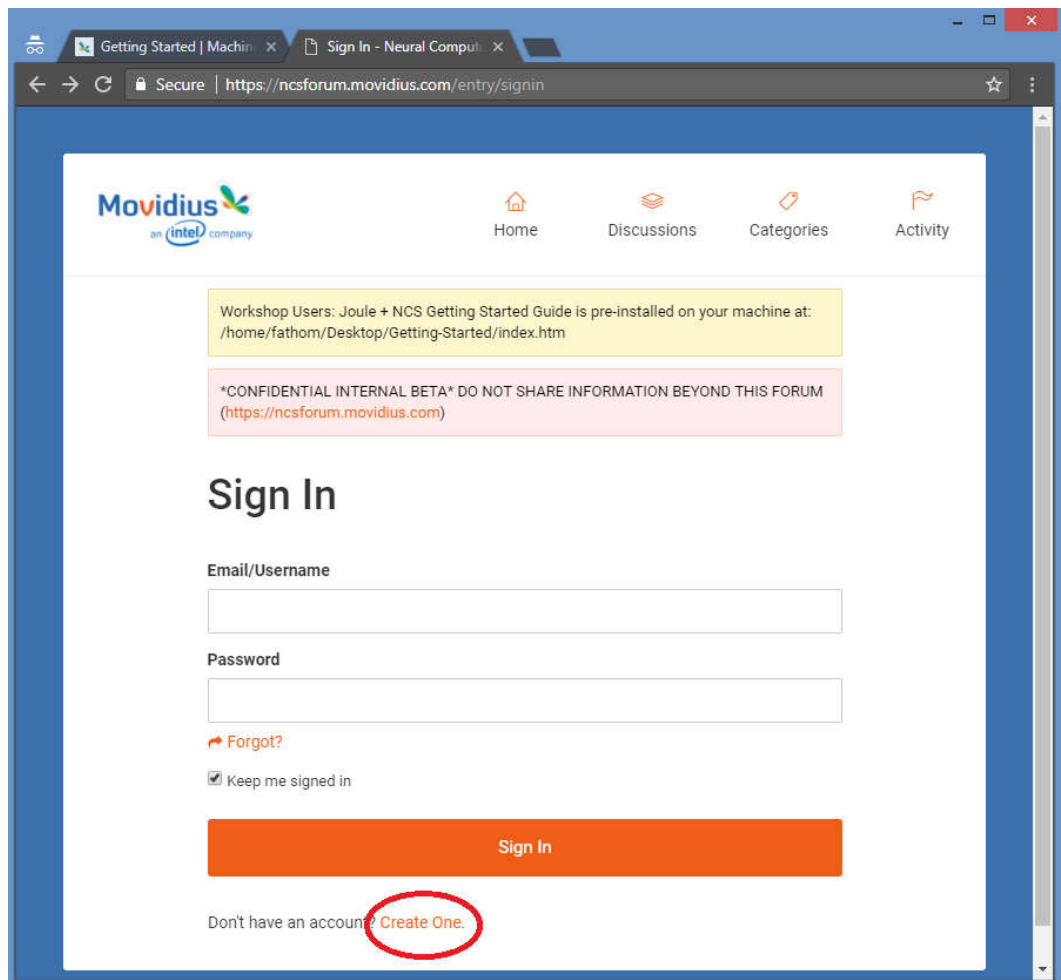
Figure: Connecting the Movidius™ NCS



1.1.3 The user forum

While creating a forum account is not required to download and install the SDK, doing so will enable participation in conversations where ideas and solutions are shared.

If you want to create an account click the sign in button and then the “Don’t have an account? Create one” link.



The screenshot shows a web browser window with two tabs: "Getting Started | Machine..." and "Sign In - Neural Comput...". The address bar shows the URL "https://ncsforum.movidius.com/entry/signin". The page features the Movidius logo (an Intel company) and navigation links for Home, Discussions, Categories, and Activity. Two informational boxes are present: a yellow one stating "Workshop Users: Joule + NCS Getting Started Guide is pre-installed on your machine at: /home/fathom/Desktop/Getting-Started/index.htm" and a pink one with a disclaimer: "*CONFIDENTIAL INTERNAL BETA* DO NOT SHARE INFORMATION BEYOND THIS FORUM (https://ncsforum.movidius.com)". The main heading is "Sign In". Below it are input fields for "Email/Username" and "Password". There is a "Forgot?" link, a checked "Keep me signed in" checkbox, and a large orange "Sign In" button. At the bottom, the text "Don't have an account?" is followed by a red circle around the "Create One." link.

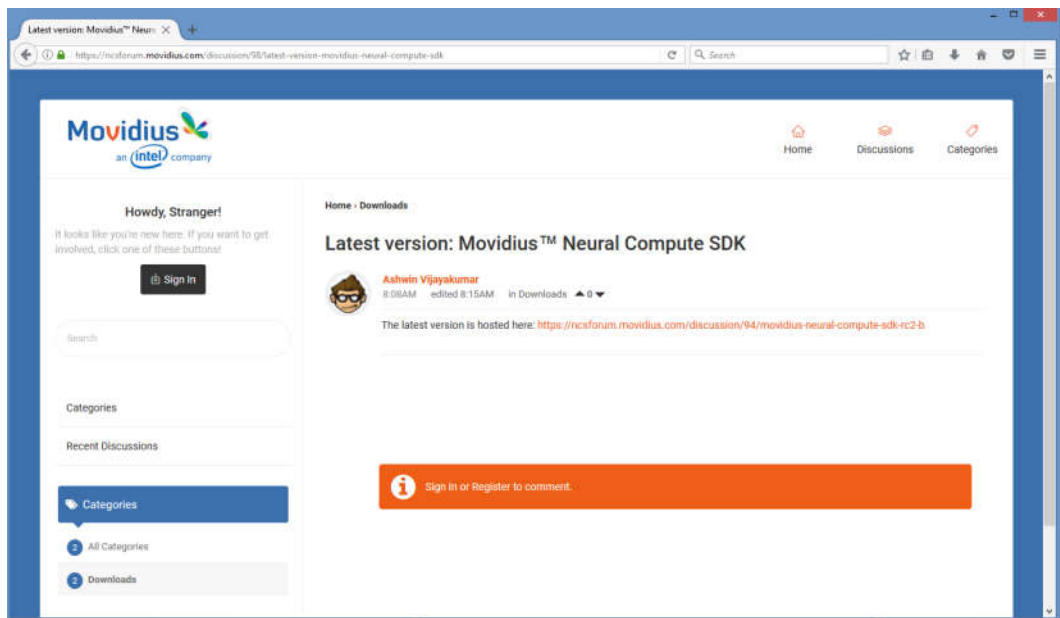
1.1.4 Downloading the Movidius™ SDK

The Movidius™ SDK provides the software tools, development libraries, and samples that enable the NCS. An active internet connection is required to download and install the SDK. The SDK is distributed via the Movidius™ NCS User Forum

<https://ncsforum.movidius.com>

You can get the latest release from this direct link:

<https://ncsforum.movidius.com/discussion/98/latest-version-movidius-neural-compute-sdk>

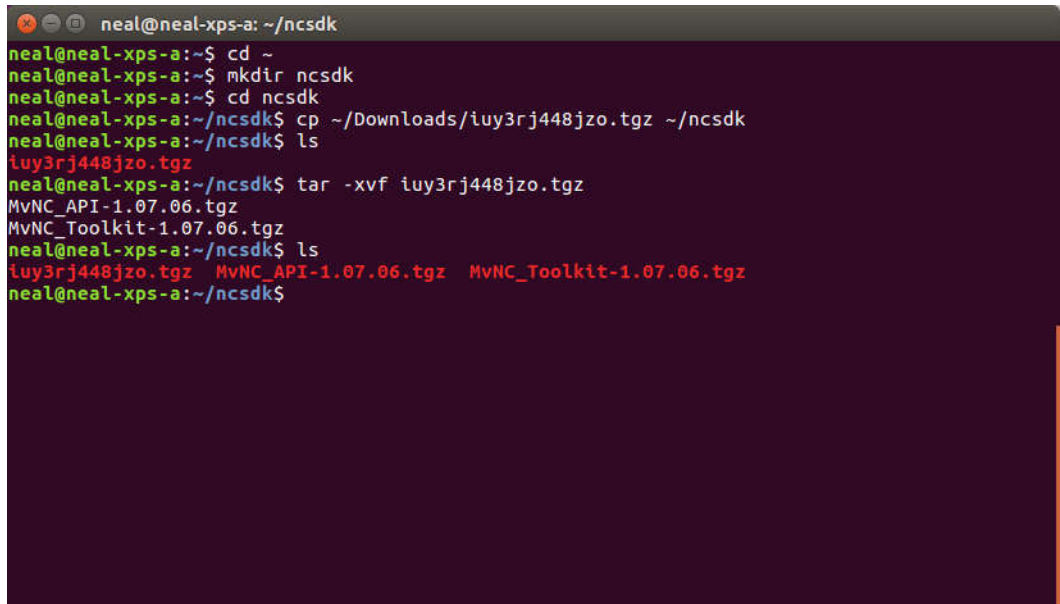


When you have navigated to the forum post linked above, click on the link to go to the specific post for the latest version. The SDK will be available for download as an attachment to the post and will be downloaded as a single, compressed archive file.

Note: When downloading from the forum the file name may be changed by the browser unless you right click and save as. In the next image the file is named *iuy3rj448jzo.tgz*

After the archive is downloaded, copy it to a directory of your choice such as ~/ncsdk. Next expand the archive with the tar command below. This will create two new archive files, one for the toolkit and one for the API.

```
$ cp ~/Downloads/ iuy3rj448jzo.tgz ~/ncsdk  
  
$ cd ~/ncsdk  
  
tar -xvf <downloaded sdk file>
```



```
neal@neal-xps-a: ~/ncsdk  
neal@neal-xps-a:~$ cd ~  
neal@neal-xps-a:~$ mkdir ncsdk  
neal@neal-xps-a:~$ cd ncsdk  
neal@neal-xps-a:~/ncsdk$ cp ~/Downloads/iuy3rj448jzo.tgz ~/ncsdk  
neal@neal-xps-a:~/ncsdk$ ls  
iuy3rj448jzo.tgz  
neal@neal-xps-a:~/ncsdk$ tar -xvf iuy3rj448jzo.tgz  
MvNC_API-1.07.06.tgz  
MvNC_Toolkit-1.07.06.tgz  
neal@neal-xps-a:~/ncsdk$ ls  
iuy3rj448jzo.tgz  MvNC_API-1.07.06.tgz  MvNC_Toolkit-1.07.06.tgz  
neal@neal-xps-a:~/ncsdk$
```

1.2 Install and Verify the Movidius Neural Compute Toolkit

The Movidius™ NC Toolkit provides tools to enable rapid tuning, validation and profiling of Convolutional Neural Networks. The Toolkit also includes a tool to compile CNNs to binary graph files that the Neural Compute Stick can load and execute from within a user's software application.

Before you install the Toolkit you should update your system with these commands

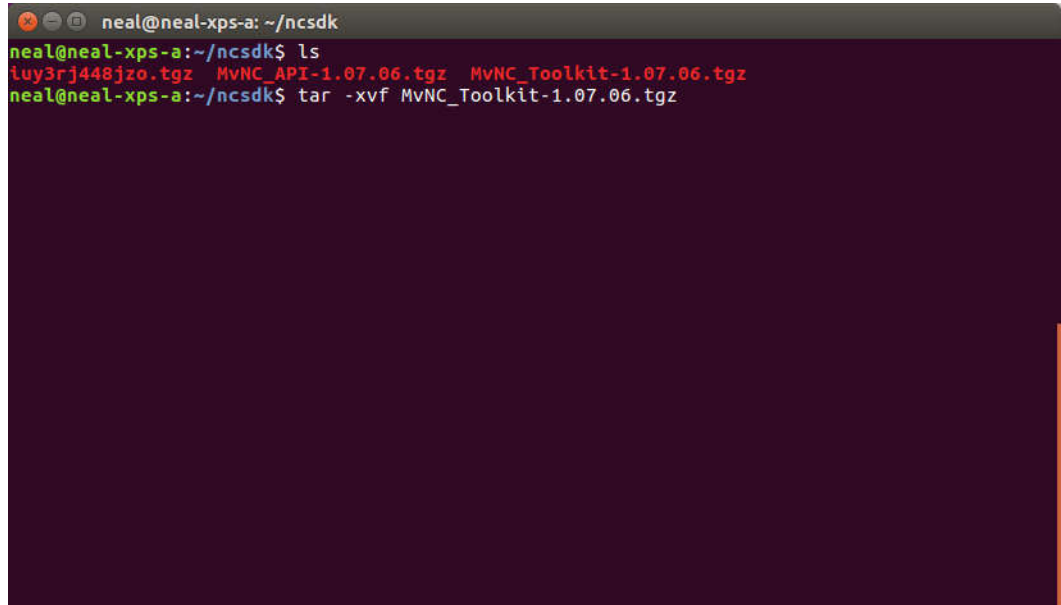
```
$ sudo apt-get update  
  
$ sudo apt-get upgrade
```

After the SDK has been downloaded and extracted, the Toolkit archive will be created. Follow the steps below to install the Toolkit. Note that your development host computer will need to be connected to the internet for the installation to work properly.

Expand the Toolkit archive with:

```
$ tar -xvf <MvNC_Toolkit file name>
```

`tar -xvf <toolkit archive filename>`

A terminal window with a dark background and light-colored text. The prompt is 'neal@neal-xps-a: ~/ncsdk'. The user enters 'ls' and the output shows three files: 'tuy3rj448jzo.tgz', 'MvNC_API-1.07.06.tgz', and 'MvNC_Toolkit-1.07.06.tgz'. The user then enters 'tar -xvf MvNC_Toolkit-1.07.06.tgz' and the terminal shows a large block of output, which is mostly obscured by a dark rectangular area, likely representing a large amount of text or a screenshot of a long output stream.

```
neal@neal-xps-a: ~/ncsdk
neal@neal-xps-a:~/ncsdk$ ls
tuy3rj448jzo.tgz  MvNC_API-1.07.06.tgz  MvNC_Toolkit-1.07.06.tgz
neal@neal-xps-a:~/ncsdk$ tar -xvf MvNC_Toolkit-1.07.06.tgz
```

After the toolkit has been extracted run the following commands to install it.

```
$ cd bin
```

```
$ ./setup.sh (This may take 15 minutes or more)
```

The setup.sh script will prompt for a password and also prompt you for a location to install Caffe.

Note: After Toolkit setup.sh has finished be sure to open a new terminal session if you haven't already. This is also indicated at the end of the setup.sh output as shown below.

```
neal@neal-xps-a: ~/ncsdk/bin
-- Installing: /opt/movidius/caffe/build/install/python/caffe/io.py
-- Installing: /opt/movidius/caffe/build/install/python/caffe/coord_map.py
-- Installing: /opt/movidius/caffe/build/install/python/caffe/imagenet
-- Installing: /opt/movidius/caffe/build/install/python/caffe/imagenet/ilsrvrc_2012_mean.np
y
-- Installing: /opt/movidius/caffe/build/install/python/caffe/pycaffe.py
-- Installing: /opt/movidius/caffe/build/install/python/caffe/classifier.py
-- Up-to-date: /opt/movidius/caffe/build/install/python/caffe/proto
-- Installing: /opt/movidius/caffe/build/install/python/caffe/proto/caffe_pb2.py
-- Installing: /opt/movidius/caffe/build/install/python/caffe/proto/__init__.py
-- Installing: /opt/movidius/caffe/build/install/python/caffe/_caffe.so
-- Set runtime path of "/opt/movidius/caffe/build/install/python/caffe/_caffe.so" to "/opt
/movidius/caffe/build/install/lib:/usr/lib/x86_64-linux-gnu/hdf5/serial/lib"
Configuring device udev rules
0+1 records in
0+1 records out
378 bytes copied, 2.2195e-05 s, 17.0 MB/s
Adding user 'neal' to 'users' group
Augmenting PYTHONPATH environment variable
Setup is complete.
The PYTHONPATH environment variable was added to your .bashrc as described in the Caffe doc
umentation.
Keep in mind that only newly spawned terminals can see this variable!
This means that you need to open a new terminal in order to be able to use the toolkit.
Please provide feedback in our support forum if you encountered difficulties
neal@neal-xps-a:~/ncsdk/bin$
```

In the new terminal execute the following command to verify that your \$PYTHONPATH includes the python directory inside your caffe installation directory.

Output should be similar to the following:

```
$ echo $PYTHONPATH
:/opt/Movidius/caffe/python:
```

if your \$PYTHONPATH is not set, run the following command:

```
$ source ~/.bashrc
$ cd <toolkit bin dir>/data
```

The command below downloads caffe models for the example code:

```
$ ./dlnets.sh
```

1.3 Install and Verify the Movidius NC API

The Movidius™ NC API provides user software applications with an Application Programming Interface to take advantage of the Movidius™ NCS features.

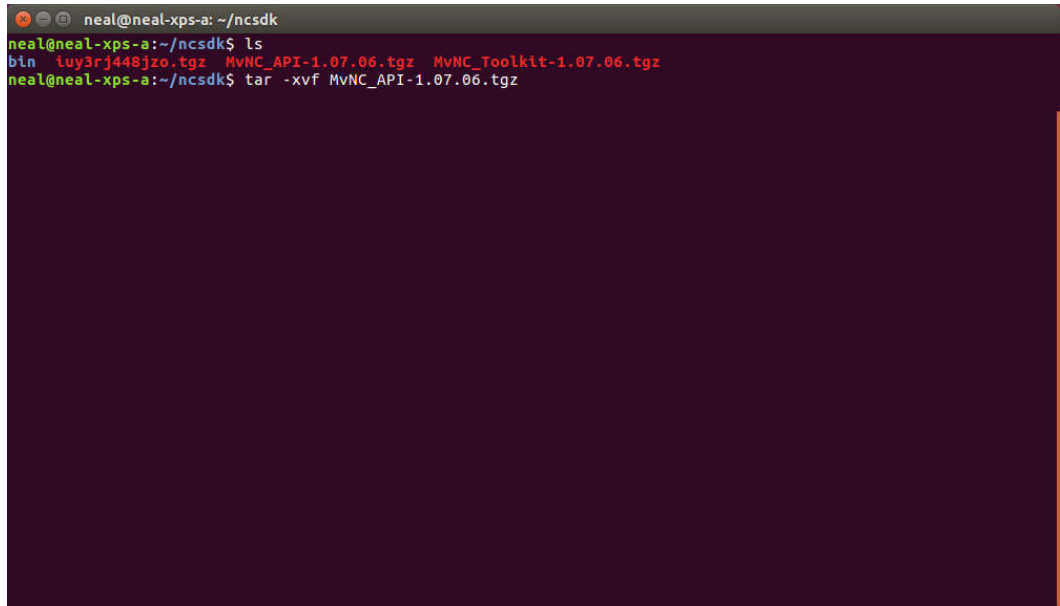
After the SDK has been downloaded and extracted, the API archive will be created. Follow the steps below to install the API.

Note: Your development host computer will need to be connected to the internet for the installation to work properly.

Note: The Toolkit must be setup prior to the API.

Expand the API archive with this tar command to expand the ncapi directory.

```
$ tar -xvf <API archive file name>
```



```
neal@neal-xps-a: ~/ncsdk
neal@neal-xps-a:~/ncsdk$ ls
bin iuy3rj448jzo.tgz MvNC_API-1.07.06.tgz MvNC_Toolkit-1.07.06.tgz
neal@neal-xps-a:~/ncsdk$ tar -xvf MvNC_API-1.07.06.tgz
```

After the API archive has been expanded follow the steps below to install it.

```
$ cd ncapi
$ ./setup.sh (this will take a few minutes to finish.)
```

1.4 Using the Movidius™ NC Toolkit

Now that the Toolkit is installed you have access to the three main tools it provides:

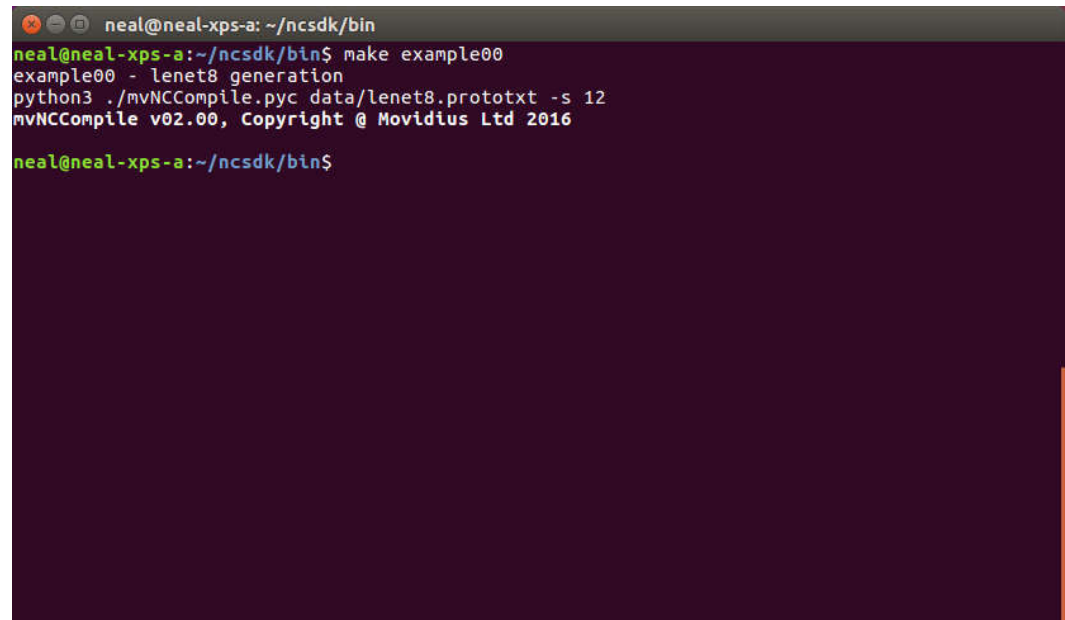
mvNCProfile.pyc, and mvNCCheck.pyc, mvNCCompile.pyc,

Plug in the Movidius™ NCS to an available USB port on host system and perform the following commands to compile a prototxt file into a graph file that applications built with the API can load onto the NCS for hardware accelerated inferences.

```
$ cd <toolkit directory>/bin
```

```
$ make example00
```

You should see the following output:

A terminal window screenshot showing the execution of 'make example00'. The prompt is 'neal@neal-xps-a: ~/ncsdk/bin'. The output shows 'example00 - lenet8 generation', followed by 'python3 ./mvNCCompile.pyc data/lenet8.prototxt -s 12', and 'mvNCCompile v02.00, Copyright @ Movidius Ltd 2016'. The prompt returns to 'neal@neal-xps-a:~/ncsdk/bin\$'.

```
neal@neal-xps-a: ~/ncsdk/bin
neal@neal-xps-a:~/ncsdk/bin$ make example00
example00 - lenet8 generation
python3 ./mvNCCompile.pyc data/lenet8.prototxt -s 12
mvNCCompile v02.00, Copyright @ Movidius Ltd 2016
neal@neal-xps-a:~/ncsdk/bin$
```

1.4.1 Make example01

Enter the following command:

```
$ make example01
```

This will create bin/output_report.html with profile details for the GoogleNet as well as the following output on the terminal.

```
neal@neal-xps-a: ~/ncsdk/bin
neal@neal-xps-a:~/ncsdk/bin$ make example01
echo "example01 - GoogleNet profiling"
example01 - GoogleNet profiling
python3 ./mvNCProfile.pyc data/googlenet.prototxt -s 12
mvNCProfile v02.00, Copyright @ Movidius Ltd 2016

USB: Transferring Data...
Time to Execute : 114.28 ms
USB: Myriad Execution Finished
Time to Execute : 94.0 ms
USB: Myriad Execution Finished
USB: Myriad Connection Closing.
USB: Myriad Connection Closed.
Network Summary

Detailed Per Layer Profile
Layer      Name                                     MFLOPs    Bandwidth MB/s    time(ms)
=====
0      conv1/7x7_s2                            236.028      2508.67          5.62
1      pool1/3x3_s2                             1.806      1440.52          1.06
2      pool1/norm1                              0.000       712.37           0.54
3      conv2/3x3_reduce                          25.690       405.32           0.96
4      conv2/3x3                                693.633      315.43          11.59
5      conv2/norm2                               0.000       795.01           1.44
6      pool2/3x3_s2                             1.355      1493.75          0.77
7      inception_3a/1x1                          19.268       462.52           0.67
8      inception_3a/3x3_reduce                    28.901       399.90           0.81
9      inception_3a/3x3                          173.408       324.83           4.63
10     inception_3a/5x5_reduce                     4.817       791.58           0.37
11     inception_3a/5x5                          20.070       852.28           0.73
12     inception_3a/pool                          1.355       685.40           0.42
13     inception_3a/pool_proj                      9.634       556.86           0.54
14     inception_3b/1x1                          51.380       468.30           0.95
15     inception_3b/3x3_reduce                     51.380       474.46           0.94
16     inception_3b/3x3                         346.817       271.07           7.92
17     inception_3b/5x5_reduce                     12.845      1096.07           0.36
18     inception_3b/5x5                        120.422       580.66           2.32
19     inception_3b/pool                          1.806       696.19           0.55
20     inception_3b/pool_proj                     25.690       682.73           0.61
21     pool3/3x3_s2                               0.847      1307.68           0.55
22     inception_4a/1x1                          36.127       390.54           0.91
23     inception_4a/3x3_reduce                     18.063       573.13           0.47
24     inception_4a/3x3                          70.447       322.64           2.07
25     inception_4a/5x5_reduce                      3.011      1032.97           0.19
26     inception_4a/5x5                          7.526       616.02           0.31
27     inception_4a/pool                          0.847       632.08           0.28
28     inception_4a/pool_proj                     12.042       662.47           0.36
29     inception_4b/1x1                          32.113       287.77           1.21
30     inception_4b/3x3_reduce                     22.479       377.30           0.80
31     inception_4b/3x3                         88.510       309.01           2.63
32     inception_4b/5x5_reduce                      4.817       837.93           0.26
33     inception_4b/5x5                        15.053       374.62           0.80
34     inception_4b/pool                          0.903       614.64           0.31
35     inception_4b/pool_proj                     12.845       551.18           0.46
```


1.4.2 Make example02

Enter the following command:

```
$ make example02
```

Similar to example01, this will create a profile report for another network (lenet8.)

You should see the output below and get a new bin/output_report.html:

```
neal@neal-xps-a: ~/ncsdk/bin
neal@neal-xps-a:~/ncsdk/bin$ make example02
echo "example02 - lenet8 profiling"
example02 - lenet8 profiling
python3 ./mvNCProfile.pyc data/lenet8.prototxt -s 12
mvNCProfile v02.00, Copyright @ Movidius Ltd 2016

USB: Transferring Data...
Time to Execute : 5.54 ms
USB: Myriad Execution Finished
Time to Execute : 4.76 ms
USB: Myriad Execution Finished
USB: Myriad Connection Closing.
USB: Myriad Connection Closed.
Network Summary

Detailed Per Layer Profile
Layer      Name                               MFLOPs    Bandwidth MB/s    time(ms)
=====
0          conv1                             5.530     939.54             0.33
1          pool1                             0.014     352.86             0.07
2          conv2                             4.301     397.78             0.58
3          pool2                             0.004     177.42             0.04
4          ip1                               0.002     2072.92            0.41
5          ip2                               0.001     473.40             0.02
6          softmax                           0.000     0.78               0.03
=====
Total inference time                                     1.48
=====

Generating Profile Report 'output_report.html'...
neal@neal-xps-a:~/ncsdk/bin$
```

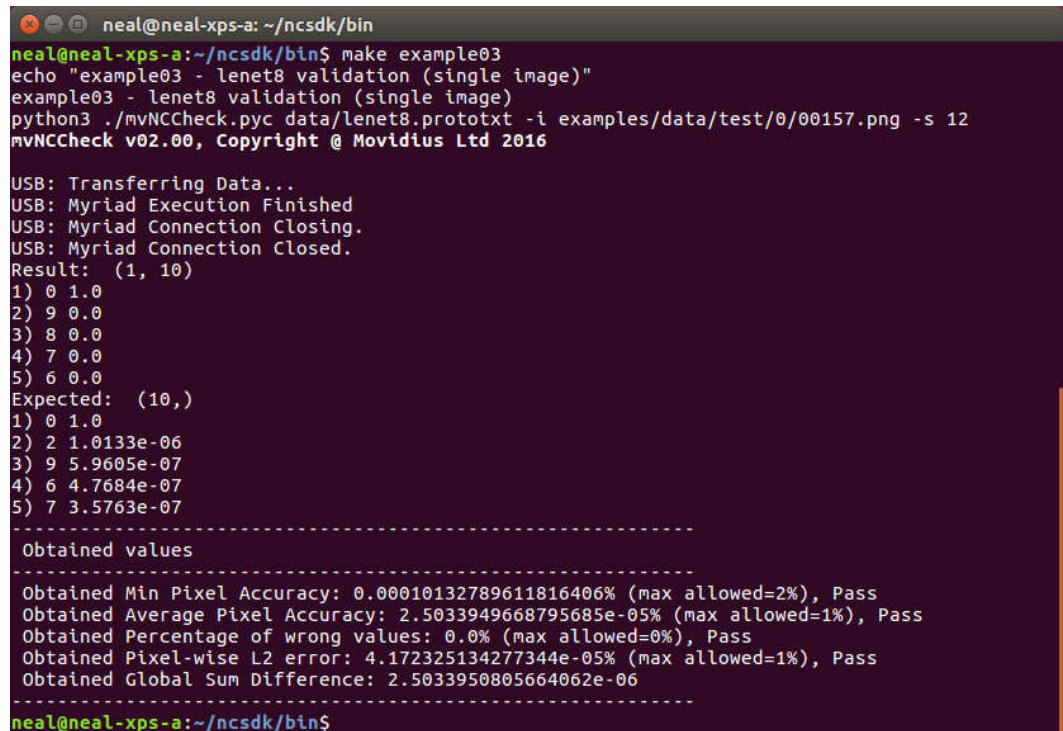
1.4.3 Make example03

Enter the following command:

```
$ make example03
```

This is checking for the correctness of network results for the lenet8 network

You should see output similar to the image shown below



```
neal@neal-xps-a: ~/ncsdk/bin
neal@neal-xps-a:~/ncsdk/bin$ make example03
echo "example03 - lenet8 validation (single image)"
example03 - lenet8 validation (single image)
python3 ./mvNCCheck.pyc data/lenet8.prototxt -i examples/data/test/0/00157.png -s 12
mvNCCheck v02.00, Copyright @ Movidius Ltd 2016

USB: Transferring Data...
USB: Myriad Execution Finished
USB: Myriad Connection Closing.
USB: Myriad Connection Closed.
Result: (1, 10)
1) 0 1.0
2) 9 0.0
3) 8 0.0
4) 7 0.0
5) 6 0.0
Expected: (10,)
1) 0 1.0
2) 2 1.0133e-06
3) 9 5.9605e-07
4) 6 4.7684e-07
5) 7 3.5763e-07
-----
Obtained values
-----
Obtained Min Pixel Accuracy: 0.00010132789611816406% (max allowed=2%), Pass
Obtained Average Pixel Accuracy: 2.5033949668795685e-05% (max allowed=1%), Pass
Obtained Percentage of wrong values: 0.0% (max allowed=0%), Pass
Obtained Pixel-wise L2 error: 4.172325134277344e-05% (max allowed=1%), Pass
Obtained Global Sum Difference: 2.5033950805664062e-06
-----
neal@neal-xps-a:~/ncsdk/bin$
```

For more information on the tools in the Toolkit, take a look at the NCS Toolkit Documentation which you can download from the Movidius™ NCS user forum here <https://ncsforum.movidius.com>.

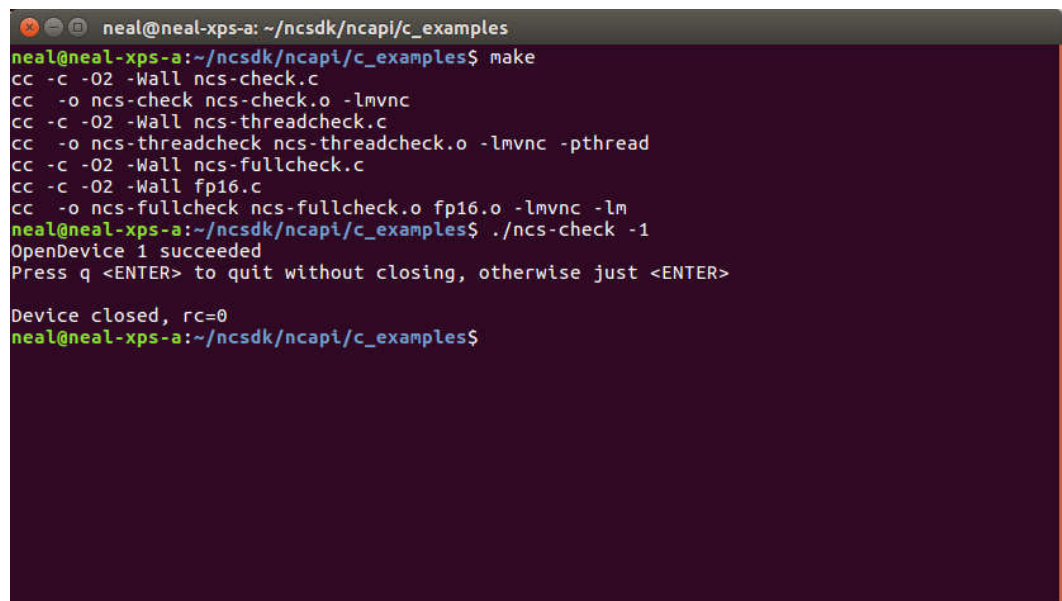
Try the other examples and options shown in the Toolkit documentation.

1.5 Using the Movidius NC API

Now that you have installed the API, connect the Movidius™ NCS to an available USB port on a host system and enter the following commands:

```
$ cd <ncapi directory>/c_examples  
  
$ make  
  
./ncs-check -1
```

You should see output similar to the image shown below



```
neal@neal-xps-a: ~/ncsdk/ncapi/c_examples  
neal@neal-xps-a:~/ncsdk/ncapi/c_examples$ make  
cc -c -O2 -Wall ncs-check.c  
cc -o ncs-check ncs-check.o -lmvnc  
cc -c -O2 -Wall ncs-threadcheck.c  
cc -o ncs-threadcheck ncs-threadcheck.o -lmvnc -pthread  
cc -c -O2 -Wall ncs-fullcheck.c  
cc -c -O2 -Wall fp16.c  
cc -o ncs-fullcheck ncs-fullcheck.o fp16.o -lmvnc -lm  
neal@neal-xps-a:~/ncsdk/ncapi/c_examples$ ./ncs-check -1  
OpenDevice 1 succeeded  
Press q <ENTER> to quit without closing, otherwise just <ENTER>  
  
Device closed, rc=0  
neal@neal-xps-a:~/ncsdk/ncapi/c_examples$
```

Make sure you have a camera connected to the developer computer, a built-in laptop web-camera will often work for this.

```
$ cd <ncapi directory>
```

1.5.1 Verify the directory structure and contents

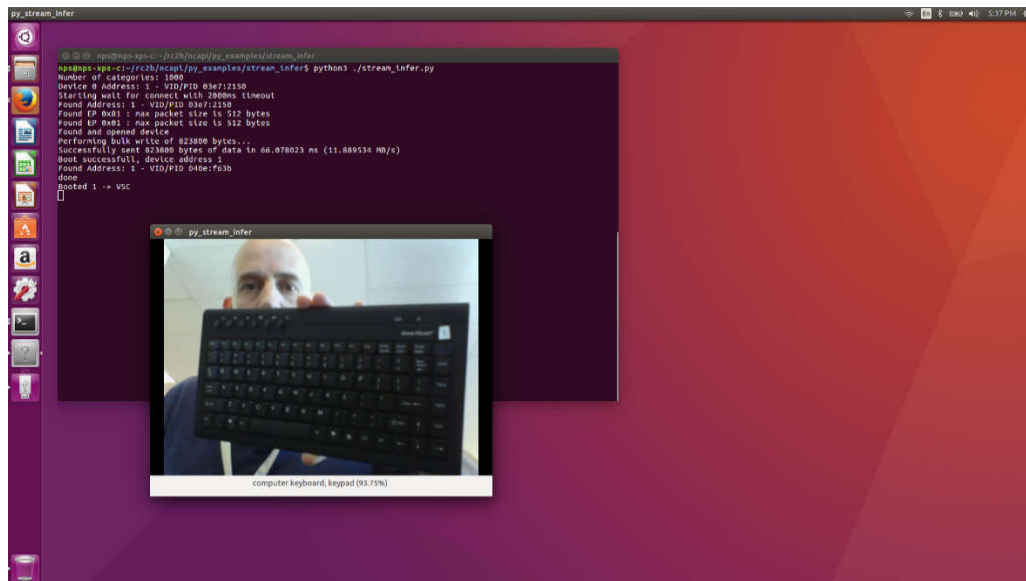
Look in the <sdk directory>/ncapi/networks/SqueezeNet directory and verify that the following files exist there:

- categories.txt
- graph
- inputsize.txt
- NetworkConfig.prototxt
- readme.txt
- squeezenet_v1.0.caffemodel
- stat.txt

If these files don't exist then run the following commands

```
$ cd <ncapi directory>/tools
$ ./get_models.sh
$ ./convert_models.sh
$ cd <ncapi directory>/py_examples/stream_infer
$ python3 ./stream_infer.py
```

You will see the example running as shown in the image below:



Place different items in front of the camera and the example will identify them.

Review the readme.html file in the stream_infer directory to get an understanding of how the sample works.

Note: Read through the API User Guide which you can download from the Movidius User Forum at <https://ncsforum.movidius.com> and take a look at the other examples provided with the API.

See appendix B for more information on API examples.

1.6 Deploying on a Raspberry Pi

Currently, the API must initially be installed on the same development computer as the Toolkit. This machine is also assumed to be running the Ubuntu 16.04 Desktop for x86-64 operating system. The previous sections in this guide describe that process.

After you have completed the initial API installation and have it working alongside the toolkit on your development computer, you may want to deploy it onto a different target machine.

The target machines are not limited to Ubuntu x86 64 bit platforms as the API ships with libraries, headers and other development files required for deploying on all API compatible operating systems which includes Raspberry Pi 3.

See the API Document from your SDK release for details on how to deploy or develop on target machines such as the Raspberry Pi.

1.6.1 Running the stream_infer.py Example on Raspberry Pi

Although the API Document provides the general details for target deployment, here are the basic high level steps you can follow to get the stream_infer example from the API to run on a Raspberry Pi.

- Update your Raspberry Pi with these commands:
 - `$ sudo apt-get update`
 - `$ sudo apt-get install`
- install .deb packages for Raspbian (as in API Document)
 - Copy the <sdk directory>/ncapi/redis/pi_jessie directory on your Ubuntu development computer to ~/pi_jessie dir on the Raspberry Pi
 - Install the packages with this command:
`$ sudo dpkg -i ~/pi_jessie/*.deb`
- create ~/ncapi/py_examples on the Raspberry Pi
- copy the <sdk directory>/ncapi/py_examples/stream_infer directory on your Ubuntu developer computer to your Raspberry Pi so that it is here:
~/ncapi/py_examples/stream_infer
- edit ~/ncapi/py_examples/stream_infer/stream_infer.py on your Raspberry Pi to change this line
SINK_NAME = "xvimagesink"

to this:

SINK_NAME = "glimagesink"

- copy <sdk directory>/ncapi/networks directory on your Ubuntu developer computer to your Raspberry Pi so that it is here:
~/ncapi/networks
- Install the following packages on the Raspberry Pi with these commands:
 - \$ sudo apt install gstreamer-1.0
 - \$ sudo apt install python3-gst-1.0
 - \$ sudo apt-get install gir1.2-gstreamer-1.0
 - \$ sudo apt-get install gir1.2-gst-plugins-base-1.0
- Run the example on the Raspberry Pi:
 - \$ cd ~/ncapi/py_examples/stream_infer
 - \$ python3 stream_infer.py

2.0 *Appendix A: Toolkit Utilities and Examples*

Note: All Toolkit utilities and samples must be executed from the Toolkit's bin directory.

To run the included examples and utilities you will need to have specific networks available in the Toolkit bin/data directory. The `dlnets.sh` script in that directory will download these networks.

Run the following command from the Toolkit bin/data directory prior to running Toolkit utilities and examples:

```
$ dlnets.sh
```

2.1 Utilities

2.1.1 Movidius™ Neural Compute compiler

The compiler is used to create an optimized binary graph file for the NCS.

Usage:

```
$ python3 ./mvNCCompile.pyc <network.prototxt> [--help] [-w  
<weights file>] [-s <number of shaves>] [-in <input node name>]  
[-on <output node name>] [-is <image width> <image height>] [-  
o <path>]
```

See the Toolkit documentation for more usage details.

Example command:

```
$ python3 ./mvNCCompile.pyc ./data/lenet8.prototxt -w  
./data/lenet8.caffemodel -s 12 -o ./lenet8_graph
```

2.1.2 Movidius™ Neural Compute checker

The checker runs a single inference on the NCS, allowing for the calculation of classification correctness.

A sample data set can be found in the bin/examples/data/test directory.

Usage:

```
$ python3 ./mvNCCheck.pyc <network.prototxt> [--help] [-w  
<weights file>] [-s <number of shaves>] [-in <input node name>]  
[-on <output node name>] [-is <image width> <image height>] [-i  
<image>] [-id <expected id>] [-S <scale factor>] [-M <number  
or numpy mean file>]
```

See the Toolkit documentation for more usage details.

Example command:

```
$ python3 ./mvNCCheck.pyc ./data/lenet8.prototxt -w  
./data/lenet8.caffemodel -s 12 -i  
./examples/data/test/5/00015.png -id 5 -S 255 -M  
./data/imagenet_mean.npy
```

2.1.3 Movidius™ Neural Compute profiler

The profiler provides a detailed stage-by-stage breakdown of network performance.

Usage:

```
$ python3 ./mvNCProfile.pyc <network.prototxt> [--help] [-w  
<weights file>] [-s <number of shaves>] [-in <input node name>]  
[-on <output node name>] [-is <image width> <image height>]
```

See the Movidius NC Toolkit documentation for more usage details.

Example command:

```
$ python3 ./mvNCProfile.pyc ./data/lenet8.prototxt -w  
./data/lenet8.caffemodel -s 12
```

2.2 Examples

2.2.1 top_5_over_a_dataset

This is an example user script for calculating the top-5 accuracy of a network over a given dataset.

Usage:

```
$ python3 ./examples/top_5_over_a_dataset.py [<data  
directory>] [<network.prototxt>] [<weights file>]
```

Example commands:

```
$ python3 ./examples/top5_over_a_dataset.py  
  
$ python3 ./examples/top_5_over_a_dataset.py  
./examples/data/test ./data/lenet8.prototxt  
./data/lenet8.caffemodel
```


3.0 *Appendix B: API Examples*

3.1 C Examples

C examples must be built before they can be run. To build these examples, enter the following command in the `ncapi/c_examples` directory:

```
$ make
```

3.1.1 `ncs-check`

This example opens the device, allocates a graph, sends some random data representing an input, and gets the result. When it has done this `<count>` number of times, it deallocates the graph and closes the device.

Usage:

```
$ ncs-check [-l<loglevel>] -1 (try one device)
$ ncs-check [-l<loglevel>] -2 (try two devices)
$ ncs-check [-l<loglevel>] [-c<count>] <network directory>
```

The `<loglevel>` can be 0 for no log output (default), 1 for errors only, or 2 for verbose log output.

The `<count>` is the number of times to run inferences (default 2).

The `<network directory>` should be a directory containing `graph`, `stat.txt`, `categories.txt`, and `inputsize.txt`. Sample networks can be found in the `ncapi/networks` directory.

Example commands:

These example commands must be called from the `ncapi/c_examples` directory.

```
$ ./ncs-check -1
$ ./ncs-check -l2 -c3 ../networks/SqueezeNet
```

3.1.2 ncs-threadcheck

This example executes the same tasks as ncs-check but uses a threaded approach.

Usage:

```
$ ncs-threadcheck [-l<loglevel>] [-c<count>] <network  
directory>
```

The <loglevel> can be 0 for no log output (default), 1 for errors only, or 2 for verbose log output.

The <count> is the number of times to run inferences (default 2).

The <network directory> should be a directory containing graph, stat.txt, categories.txt, and inputsize.txt. Sample networks can be found in the API ncapi/networks directory.

Example command:

This example command must be called from the ncapi/c_examples directory.

```
$ ./ncs-threadcheck ../networks/SqueezeNet
```

3.1.3 ncs-fullcheck

This example executes the same tasks as ncs-check but requires an input image.

Usage:

```
$ ncs-fullcheck [-l<loglevel>] [-c<count>] <network directory>  
<image>
```

The <loglevel> can be 0 for no log output (default), 1 for errors only, or 2 for verbose log output.

The <count> is the number of times to run inferences (default 2).

The <network directory> should be a directory containing graph, stat.txt, categories.txt, and inputsize.txt. Sample networks can be found in the API ncapi/networks directory.

The <image> is a path to an input image. Sample images can be found in the bin/images directory.

Example command:

This example command must be called from the ncapi/c_examples directory.

```
$ ./ncs-fullcheck ../networks/SqueezeNet  
../images/512_Ball.jpg
```

3.2 Python3 Examples

3.2.1 ncs_camera

This script performs inferences on streaming video using GStreamer.

Usage:

```
$ python3 ncs_camera.py [-h | --help] [--v4l2-src <V4L2  
source> | --src <video source> | --picture-src <picture  
source>] [-g <network directory>] [-d <device>] [--log-level  
<loglevel>] [-v] [--opengl]
```

<V4L2 source> is the v4l2 source device name (e.g. /dev/video0) (default: /dev/video0)

<video source> is the video source (default: None).

<picture source> is the filename of an input picture (default: None).

<network directory> is a directory containing the graph, categories.txt, inputsize.txt and stat.txt files (default: ../networks/GoogLeNet).

<device> is the name of the NCS device to use for inference (default: 1).

<loglevel> is the API logging level (0 = none, 1 = errors, 2 = verbose) (default: 0).

The verbose option (-v, --verbose) prints out additional information (default: False).

The opengl option (--opengl) will cause this script to use OpenGL instead of Xv extension for preview (default: False)

Example commands:

These example commands must be called from the ncapi/python_examples directory.

```
$ python3 ./ncs_camera.py  
  
$ python3 ./ncs_camera.py -g ../networks/SqueezeNet -v
```

3.2.2 stream_infer

This script performs inference on streaming video using GStreamer. See the readme in the `ncapi/py_examples/stream_infer` directory for more detailed usage information and a code walkthrough.

Usage:

```
$ python3 stream_infer.py
```

Example command:

This example command must be called from the `ncapi/py_examples/stream_infer` directory.

```
$ python3 ./stream_infer.py
```

3.2.3 age_gender_classification

This script performs inference on a sample image using either the Age or the Gender network. The included sample image is located at `ncapi/py_examples/image.jpg`.

This script requires OpenCV for Python. To easily install OpenCV Python bindings, you can use the `install_opencv.sh` script found in the `ncapi/py_examples` directory.

Usage:

```
$ python3 age_gender_classification.py <1 | 2>
```

Note: Use 1 for Age or 2 for Gender.

Example commands:

These commands must be called from the `ncapi/py_examples` directory.

```
$ python3 ./age_gender_classification.py 1
```

```
$ python3 ./age_gender_classification.py 2
```

3.2.4 age_gender_example

This script performs inference on a sample image using either the GoogLeNet, AlexNet, or SqueezeNet network. The included sample image is located at `ncapi/images/cat.jpg`.

This script requires OpenCV for Python. To easily install OpenCV Python bindings, you can use the `install_opencv.sh` script found in the `ncapi/py_examples` directory.

Usage:

```
$ python3 classification_example.py <1 | 2 | 3>
```

Note: Use 1 for GoogLeNet, 2 for AlexNet, or 3 for SqueezeNet

Example commands:

These commands must be called from the `ncapi/py_examples` directory.

```
$ python3 ./classification_example.py 1
```

```
$ python3 ./classification_example.py 2
```

```
$ python3 ./classification_example.py 3
```