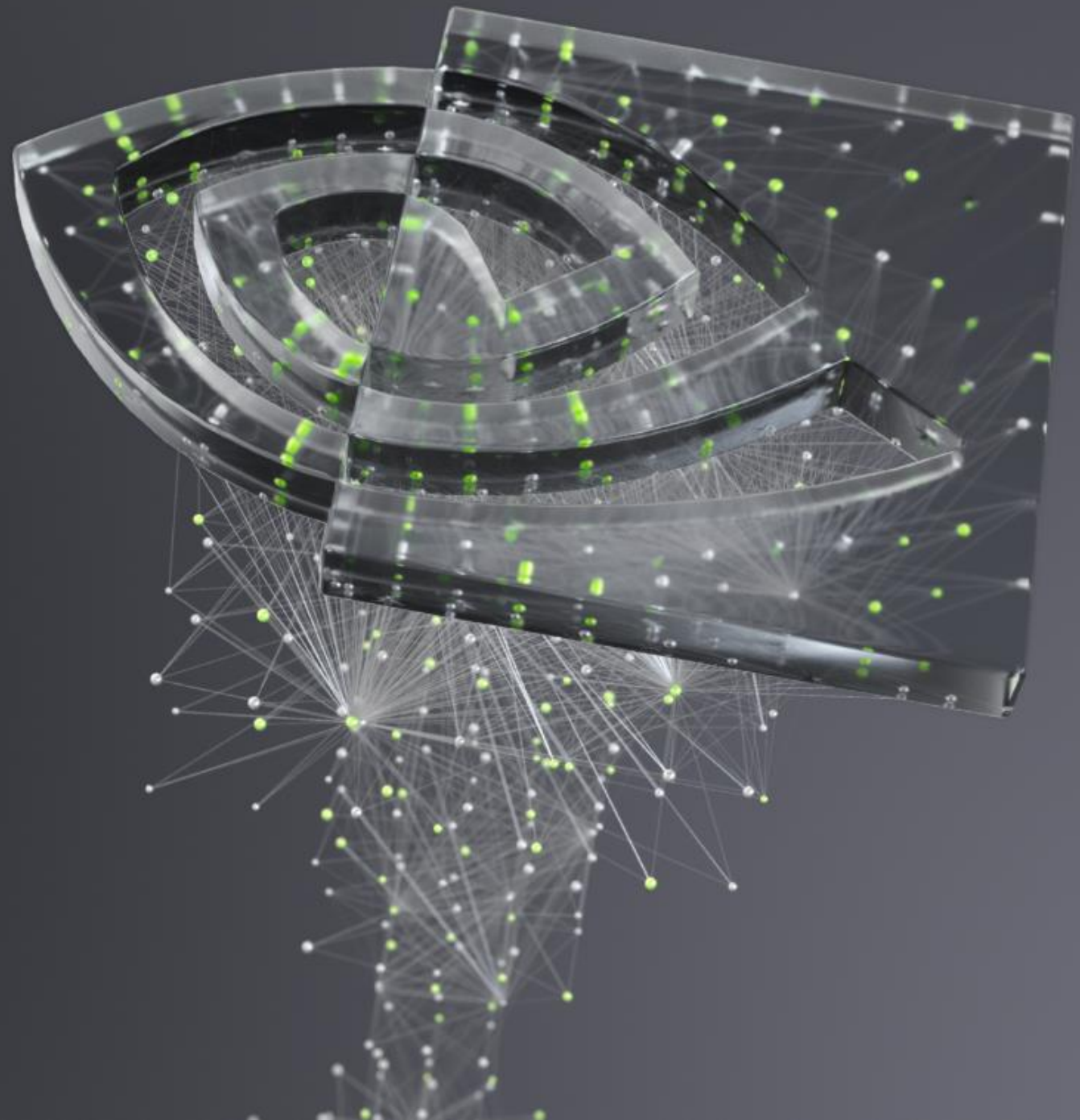


# JETSON PLATFORM

Jetson SA:  Alan Zhang

April 24,  2020

# AGENDA

## Jetson product and marketing

## Why we need cross compile

## How to setup the cross-compile environment

1. Config directly via the environment variable.
2. Config in Makefile.
3. Build your simulation.

## Compile on X86 host and execute on Jetson

1. Cross compile Linux kernel.
2. Cross compile CUDA project.
3. Cross compile Multi Media API based application.
4. Cross compile Deepstream.
5. Cross compile Caffe
6. Cross compile OpenCV.

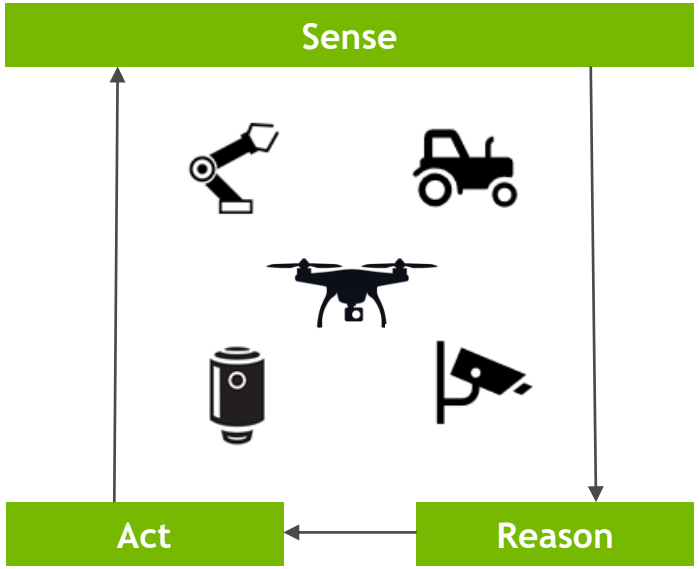
## Where can I get the prebuild version and how to build on jetson?

# NVIDIA JETSON

## Software-Defined AI Platform

### AI at the Edge

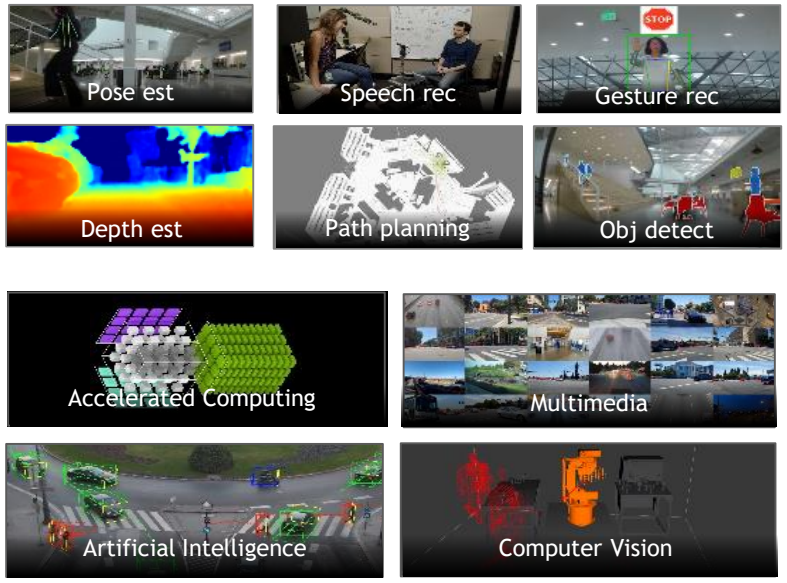
Sensor Fusion & Compute Performance




JETSON COMPUTER

### SOFTWARE DEFINED

SDK, Design Tools, Libs, GEMs

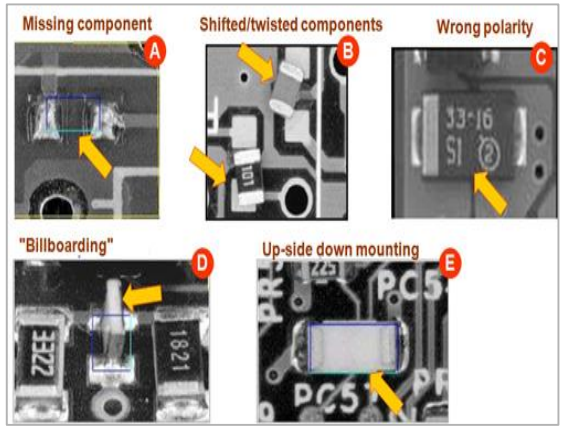



Jetpack SDK · CUDA · TensorRT · TensorFlow · ONMX · ROS

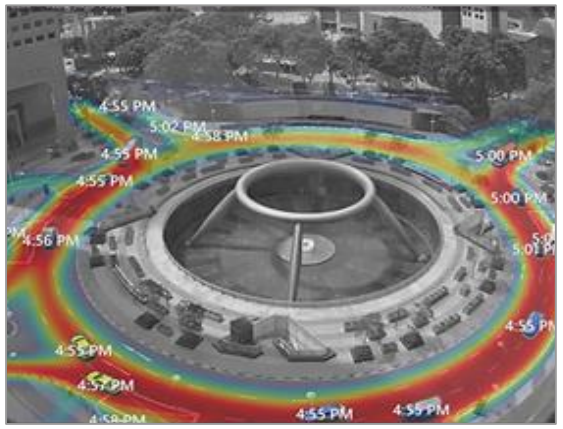
### ECOSYSTEM

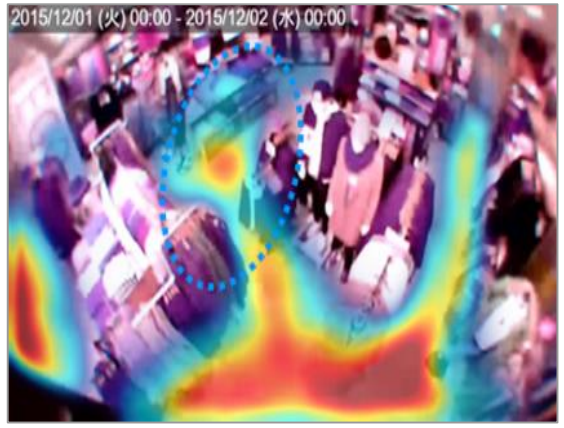
Expertise, Time to Market

# INTELLIGENT VIDEO ANALYTICS FOR EFFICIENCY AND SAFETY

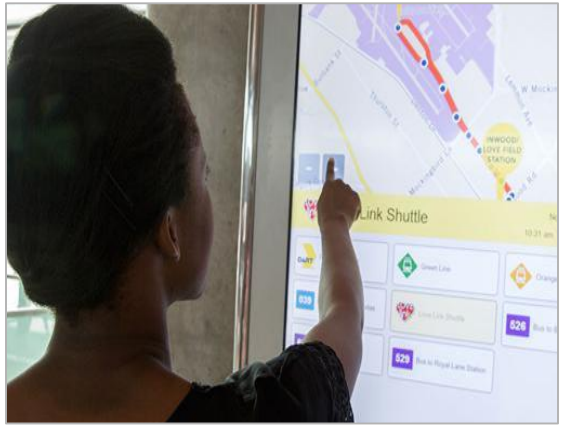




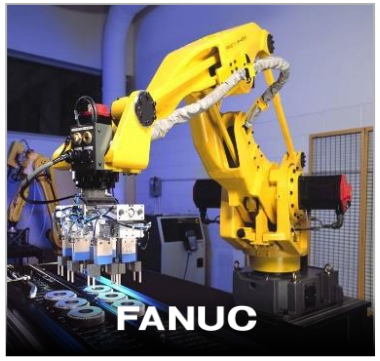






# JETSON ROBOTICS & LOGISTIC



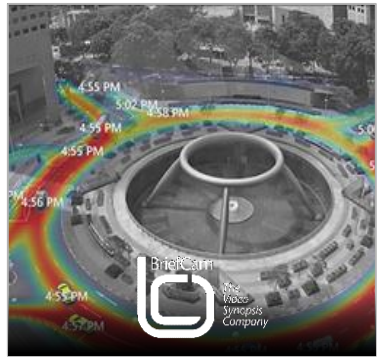

Industrial


Aerospace/Defense


Construction


Agriculture


Healthcare



Smart City

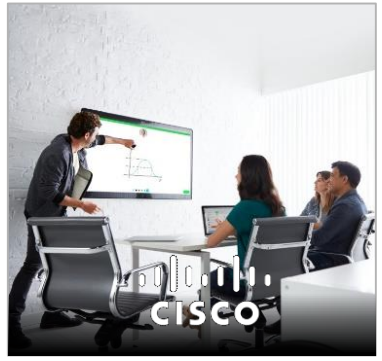

Retail


Logistics


Delivery

Inspection


Service

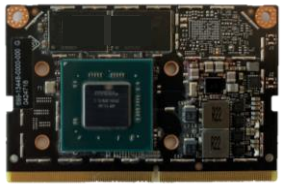



Collaboration

# THE JETSON FAMILY

## for AI at the Edge and Autonomous System designs

**JETSON NANO**
0.5 TFLOPS (FP16)



5 - 10W
45mm x 70mm

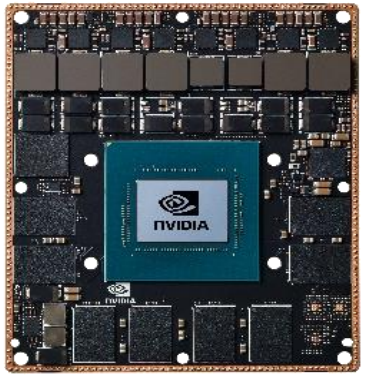
**JETSON TX2 series**
1.3 TFLOPS (FP16)

7.5 – 15W*
50mm x 87mm

**JETSON Xavier NX**
6 TFLOPS (FP16)
21 TOPS (INT8)

10 - 15W
45mm x 70mm

**JETSON AGX XAVIER series**
11 TFLOPS (FP16)
32 TOPS (INT8)



10 – 30W
100mm x 87mm

AI at the edge — Fully autonomous machines

**Same software**

Listed prices are for 1000u+ | Full specs at developer.nvidia.com/jetson

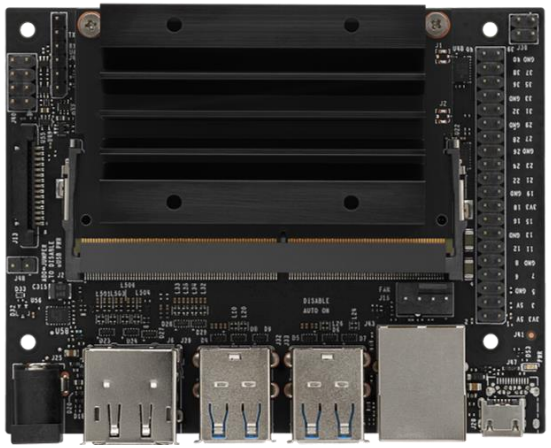
* TX2i: 10-20W


## Jetson AGX Xavier Developer Kit

- Developer Kit User Guide
- L4T Documentation
- Module Datasheet
- OEM Design Guide

## Jetson TX2 Developer Kit

- Developer Kit User Guide
- L4T Documentation
- Module Datasheet
- OEM Design Guide

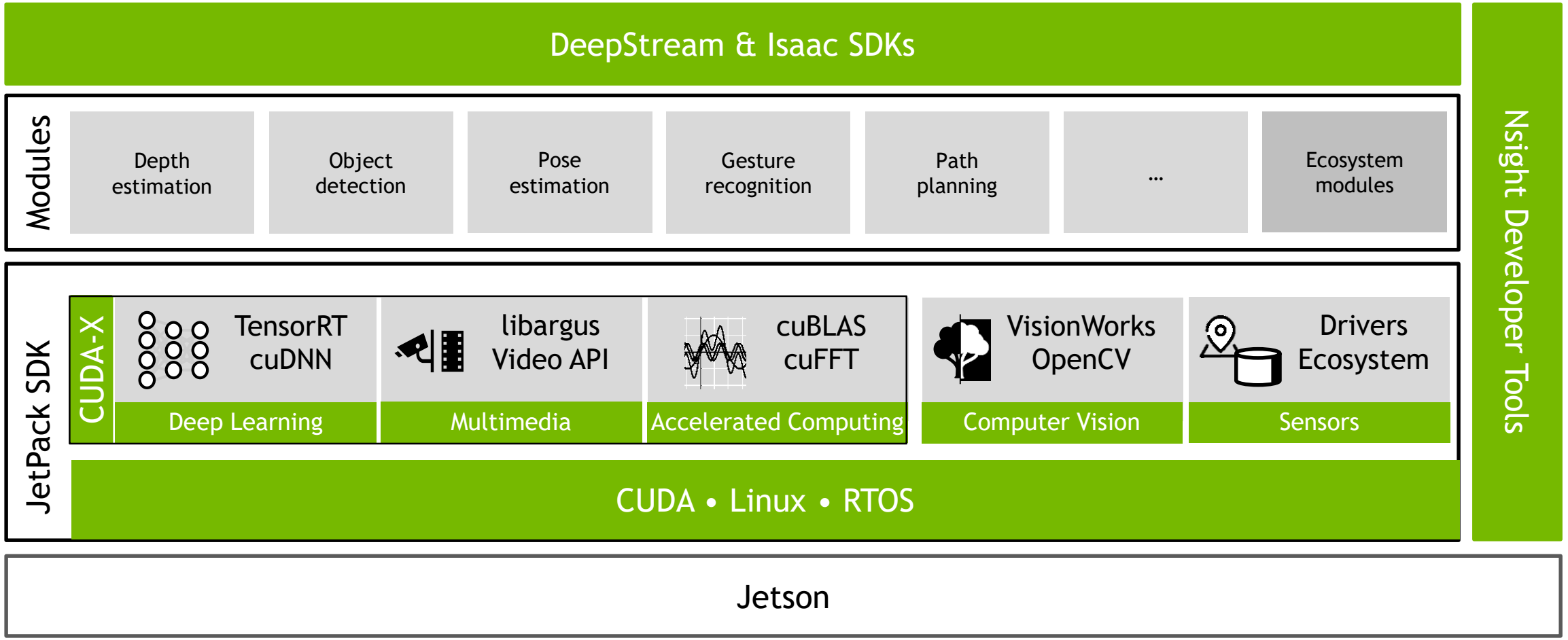


## Jetson Nano Developer Kit

- Getting Started
- Developer Kit User Guide
- L4T Documentation

| | JETSON NANO | JETSON TX2 | JETSON XAVIER NX | JETSON AGX XAVIER |
|---|---|---|---|---|
| **GPU** | 128 Core Maxwell<br>0.5 TFLOPs (FP16) | 256 Core Pascal<br>1.3 TFLOPS (FP16) | 384 Core Volta<br>21 TOPs (INT8) | 512 Core Volta + NVDLA<br>10 TFLOPS (FP16)<br>32 TOPS (INT8) |
| **CPU** | 4 core ARM A57 | 6 core Denver and A57<br>(2x) 2MB L2 | 6 core Carmel ARM CPU<br>(3x) 2MB L2 + 4MB L3 | 8 core Carmel ARM CPU<br>(4x) 2MB L2 + 4MB L3 |
| **Memory** | 4 GB 64-bit LPDDR4<br>25.6 GB/s | Up to 8 GB<br>128b LPDDR4<br>58 GB/s | 8 GB 128-bit LPDDR4x<br>51.2 GB/s | Up to 32GB 256-bit<br>LPDDR4x<br>137 GB/s |
| **Storage** | 16 GB eMMC | Up to 32 GB eMMC | 16 GB eMMC | 32 GB eMMC |
| **Encode** | 4K @ 30 (H.265) | 4K @ 60 (H.265) | 2x 4K @ 30 (H.265) | 4x 4K @ 60 (H.265) |
| **Decode** | 4K @ 60 (H.265) | 2x 4K @ 60 (H.265) | 2x 4K @ 60 (H.265) | 6x 4K @ 60 (H.265) |
| **Camera** | 12 (3x4 or 4x2) MIPI CSI-2 D-PHY 1.1 lanes (18 Gbps) | 12 lanes MIPI CSI-2<br>D-PHY 1.2 (30 Gbps)<br>C-PHY (41 Gbps) | 12 lanes (3x4 or 6x2)<br>MIPI CSI-2<br>D-PHY 1.2 (30 Gbps) | 16 lanes MIPI CSI-2 \| 8 lanes SLVS-EC<br>D-PHY (40 Gbps)<br>C-PHY (59 Gbps) |
| **Mechanical** | 69.6mm x 45mm<br>260 pin edge connector | 87mm x 50mm<br>400 pin connector | 69.6mm x 45mm<br>260 pin edge connector | 100mm x 87mm<br>699 pin connector |
| **Software** | JetPack SDK – Unified software release across all Jetson products | | | |

# JETSON SOFTWARE




Jetson software: developer.nvidia.com/jetson

# AGENDA

## Jetson product and marketing

## Why we need cross compile

## How to setup the cross-compile environment

1. Config directly via the environment variable.
2. Config in Makefile.
3. Build your simulation.

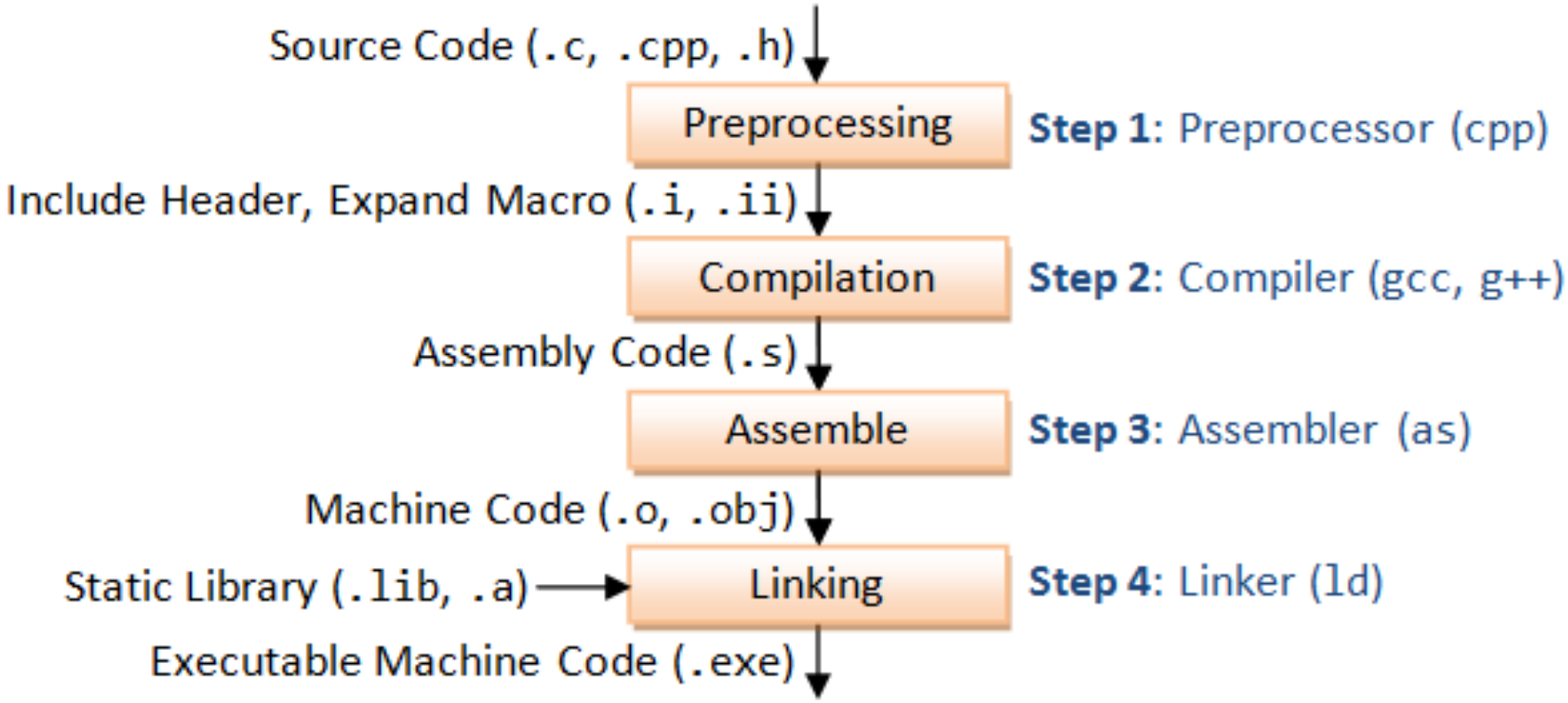
## Compile on X86 host and execute on Jetson

1. Cross compile Linux kernel.
2. Cross compile CUDA project.
3. Cross compile Multi Media API based application.
4. Cross compile Deepstream.
5. Cross compile Caffe
6. Cross compile OpenCV.

## Where can I get the prebuild version and how to build on jetson?

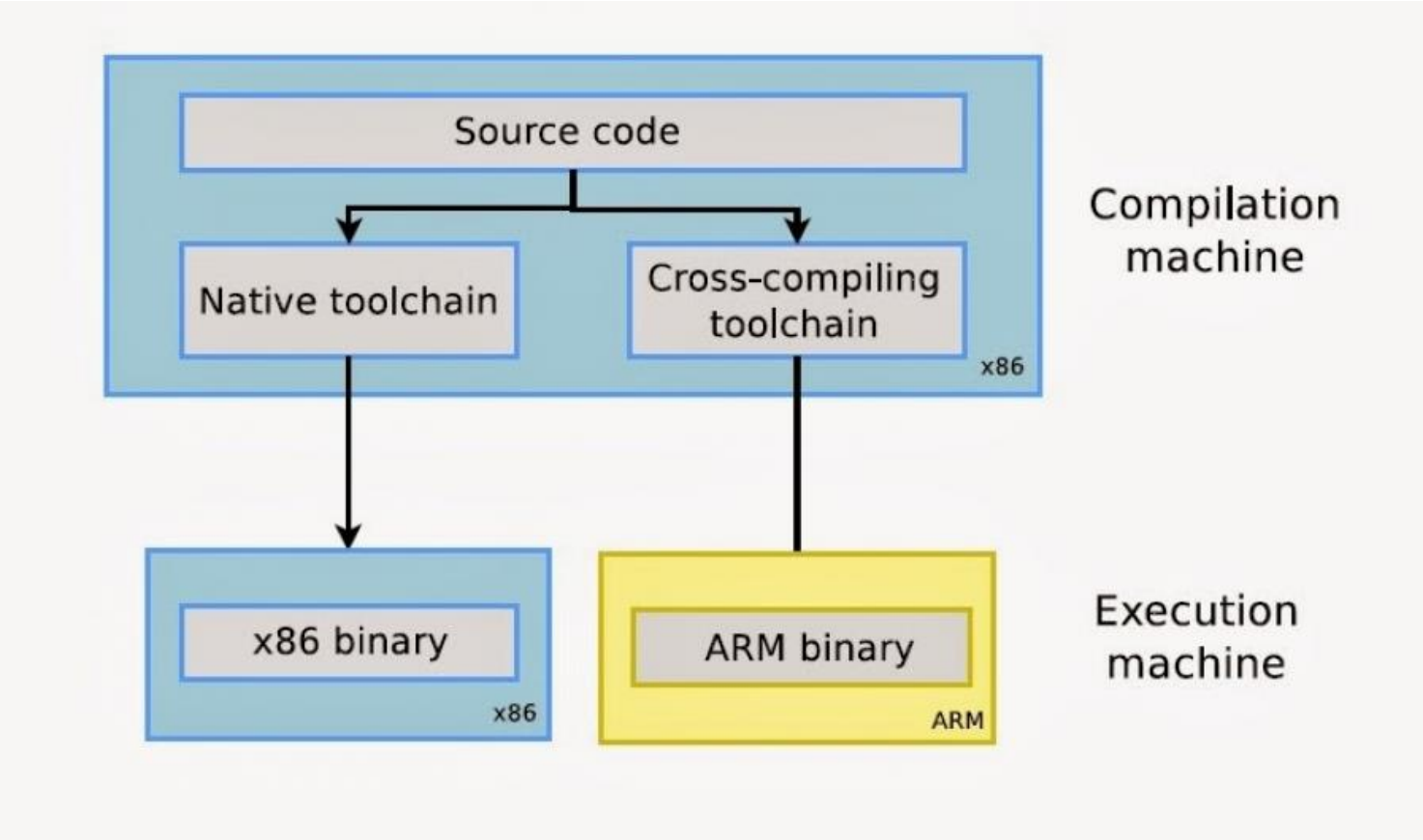
# ARCHITECTURES & INSTRUCTION SETS

## How compile works



# ARCHITECTURES & INSTRUCTION SETS

## Native compile and cross copile



# ARCHITECTURES & INSTRUCTION SETS

## What can we benefit from cross compile

- Speed
  Target platform usually designed to have low cost and low power consumption. So the speed of the CPU will be not good at compiling.
- Capability
  Target platform have less storage and less memory to sustain running a big project.
- Availability
  Not all the system for target platform has the resources to support compile natively.
- Flexibility
  A complete build environment need lots of the software to support.

# AGENDA

## Jetson product and marketing

## Why we need cross compile

## How to setup the cross-compile environment

1. Config directly via the environment variable.
2. Config in Makefile.
3. Build your simulation.

## Compile on X86 host and execute on Jetson

1. Cross compile Linux kernel.
2. Cross compile CUDA project.
3. Cross compile Multi Media API based application.
4. Cross compile Deepstream.
5. Cross compile Caffe
6. Cross compile OpenCV.

## Where can I get the prebuild version and how to build on jetson?

# SETUP THE CROSS COMPILE

## Dowload the toolchain

NVIDIA® specifies the Linaro® gcc 7.3.1 2018.05 aarch64 toolchain for:

- Cross-compiling applications to run on Jetson Linux Driver Package (L4T rel-32.
- Cross-compiling code in the L4T rel-32 source release.

This topic describes how to obtain this toolchain.

Toolchain Information

The toolchain contains the following components:

- GCC version: 7.3.1
- Binutils version: 2.28.2.20170706
- Glibc version: 2.25

Downloading the Toolchain

Download the pre-built toolchain binaries from:

http://releases.linaro.org/components/toolchain/binaries/7.3-2018.05/aarch64-linux-gnu/gcc-linaro-7.3.1-2018.05-x86_64_aarch64-linux-gnu.tar.xz

Extracting the Toolchain

Execute the following commands to extract the toolchain:

```
$ mkdir $HOME/l4t-gcc
$ cd $HOME/l4t-gcc
$ tar xf gcc-linaro-7.3.1-2018.05-x86_64_aarch64-linux-gnu.tar.xz
```

# SETUP THE CROSS COMPILE

## Options 1:

```
export CROSS_ROOT=/home/alan/rootfs-nano

export CROSS_COMPILE = /home/alan/toolchain/gcc-4.8.5-aarch64/bin

export CC = ${CROSS_COMPILE}/aarch64-unknown-linux-gnu-gcc

export CXX = ${CROSS_COMPILE}/aarch64-unknown-linux-gnu-g++

export LD = ${CROSS_COMPILE}/aarch64-unknown-linux-gnu-ld

export AR = ${CROSS_COMPILE}/aarch64-unknown-linux-gnu-ar

export AS = ${CROSS_COMPILE}/aarch64-unknown-linux-gnu-as

export RANLIB = ${CROSS_COMPILE}/aarch64-unknown-linux-gnu-ranlib

export NVCC = ${CROSS_ROOT}/usr/local/cuda/bin/nvcc

export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$SYSROOT/usr/lib/aarch64-linux-gnu:$SYSROOT/lib/aarch64-linux-gnu:$SYSROOT/lib

export LC_ALL=C

export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/home/alan/toolchain/gcc-4.8.5-aarch64/aarch64-unknown-linux-gnu/lib64:/home/alan/toolchain/gcc-4.8.5-aarch64/aarch64-unknown-linux-gnu/sysroot/lib
```

# Options 2:

```
SYSROOT=/mnt
#export CROSS_COMPILE = /home/alan/toolchain/gcc-4.8.5-aarch64/bin
CROSS_ROOT=/home/green/Downloads/toolchain/aarch64
CROSS_COMPILE=${CROSS_ROOT}/bin
CC=${CROSS_COMPILE}/aarch64-unknown-linux-gnu-gcc
CXX=${CROSS_COMPILE}/aarch64-unknown-linux-gnu-g++
LD=${CROSS_COMPILE}/aarch64-unknown-linux-gnu-ld
AR=${CROSS_COMPILE}/aarch64-unknown-linux-gnu-ar
AS=${CROSS_COMPILE}/aarch64-unknown-linux-gnu-as
RANLIB=${CROSS_COMPILE}/aarch64-unknown-linux-gnu-ranlib
NVCC=${SYSROOT}/usr/local/cuda/bin/nvcc


CFLAGS:= -Wall -Werror -std=c++11 -shared -fPIC -Wno-error=deprecated-declarations
CFLAGS+= -I$(SYSROOT)/usr/include/aarch64-linux-gnu -I$(SYSROOT)/usr/include -I${SYSROOT}/usr/local/cuda-10.0/targets/aarch64-linux/include -
I../../includes

LIBS:= -L/home/green/Downloads/toolchain/aarch64/aarch64-unknown-linux-gnu/sysroot/usr/lib /home/green/Downloads/toolchain/aarch64/aarch64-
unknown-linux-gnu/sysroot/usr/lib/libc_nonshared.a -lnvinfer -lnvparsers -L$(SYSROOT)/usr/local/cuda/lib64 -L${SYSROOT}/usr/lib/aarch64-linux-
gnu -lcudart -lcublas /home/green/Downloads/toolchain/aarch64/aarch64-unknown-linux-gnu/sysroot/usr/lib/libc_nonshared.a
LFLAGS:= -Wl,--start-group $(LIBS) -Wl,--end-group

SRCFILES:= nvdsparsebbox_ssd.cpp nvdsiplugin_ssd.cpp
TARGET_LIB:= libnvdsinfer_custom_impl_ssd.so

all: $(TARGET_LIB)

$(TARGET_LIB) : $(SRCFILES)
    $(CC) -o $@ $^ $(CFLAGS) $(LFLAGS)

clean:
    rm -rf $(TARGET_LIB)
~
```

# Setup the NFS on Jetson

```
$ sudo apt update

$ sudo apt install nfs-kernel-server

$ sudo vim /etc/exports

        / *(rw,sync,no_subtree_check, no_root_squash)

$ sudo exportfs –a

$ sudo systemctl restart nfs-kernel-server.service

$ sudo exportfs

/           (world)
```

# Setup the NFS on Host

$ sudo apt update

$ sudo apt install nfs-common

$ sudo mkdir -p /mnt/rootfs

$ sudo /etc/fstab

192.168.55.1:/ /mnt/rootfs nfs defaults 0 0

$ sudo mount -t nfs 192.168.55.1:/ /mnt/rootfs

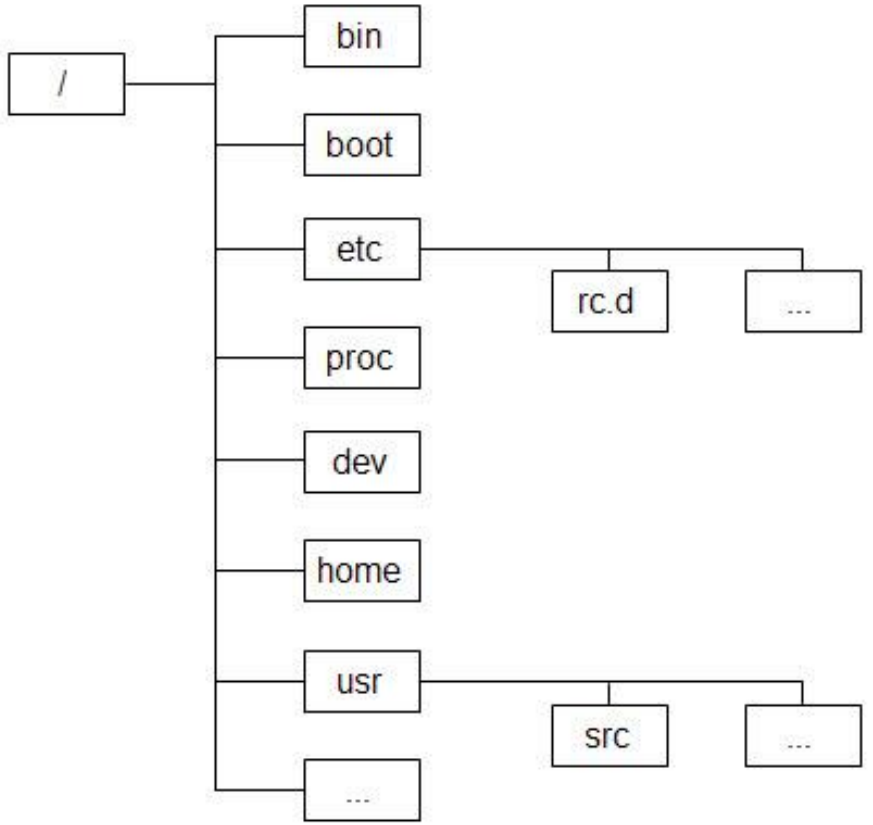
$ ls /mnt/rootfs/

```
green@C ~ $ ls /mnt/rootfs/
README.txt  bin  boot  dev  etc  home  lib  lost+found  media  mnt  opt  proc  root  run  sbin  snap  srv  sys  tmp  usr  var
```

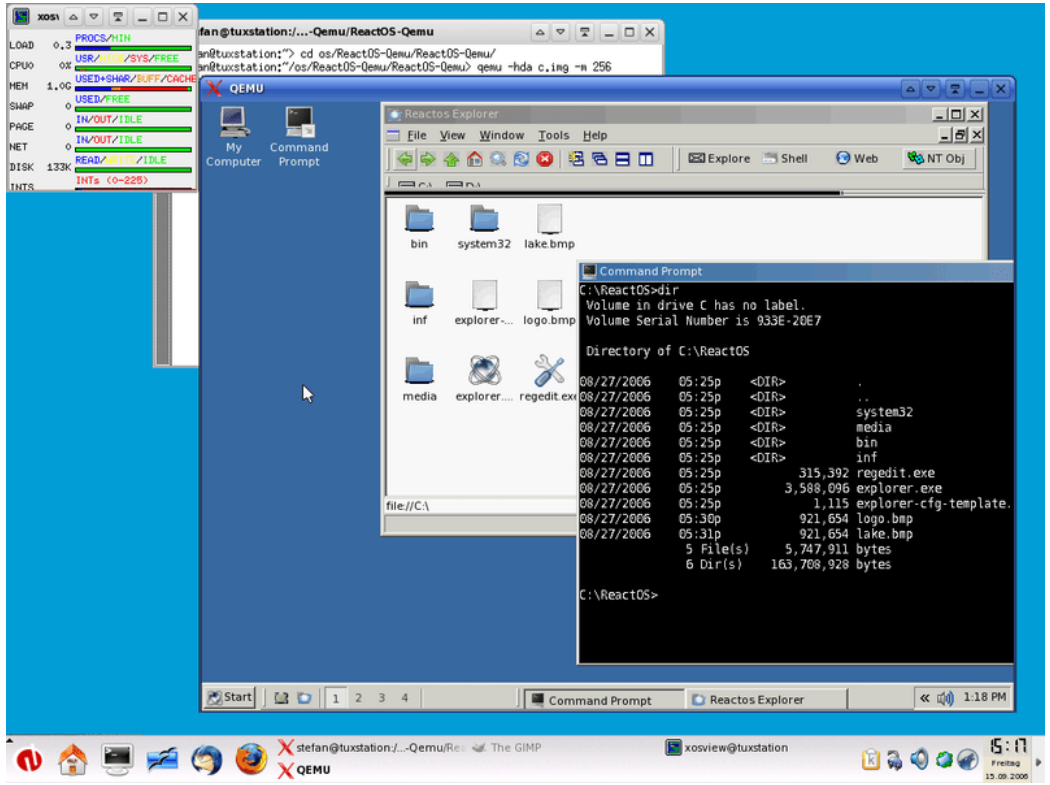
# Chroot & Qeum

## Options 3:

- 系统修复
- 系统启动时切换根目录，引导系统启动
- 特权分离



# Chroot & Qeum




```
[test@donizetti ~]$ qemu-arm ./ls --color /
bin   etc   lib64       mnt   root  srv             system-upgrade-root  var
boot  home  lost+found  opt   run   sys             tmp
dev   lib   media       proc  sbin  system-upgrade  usr
[test@donizetti ~]$ uname -a
Linux donizetti 4.6.7-300.fc24.x86_64 #1 SMP Wed Aug 17 18:48:43 UTC 2016 x86_64
 x86_64 x86_64 GNU/Linux
[test@donizetti ~]$ file ./ls
./ls: ELF 32-bit LSB executable, ARM, EABI5 version 1 (SYSV), dynamically linked
, interpreter /lib/ld-linux-armhf.so.3, for GNU/Linux 3.0.0, stripped
[test@donizetti ~]$
```

https://www.qemu.org/

# AGENDA

## Jetson product and marketing

## Why we need cross compile

## How to setup the cross-compile environment

1. Config directly via the environment variable.
2. Config in Makefile.
3. Build your simulation.

## Compile on X86 host and execute on Jetson

1. Cross compile Linux kernel.
2. Cross compile CUDA project.
3. Cross compile Multi Media API based application.
4. Cross compile Deepstream.
5. Cross compile Caffe
6. Cross compile OpenCV.

## Where can I get the prebuild version and how to build on jetson?

# Cross compile the Kernel

```
green@C ~/nvidia/nvidia_sdk $ ./source_sync.sh -t
Terminating... wrong switch: -t
Use: source_sync.sh [options]
Available general options are,
     -h     :     help
     -e     : exit on sync error
     -d [DIR] : root of source is DIR
     -t [TAG] : Git tag that will be used to sync all the sources

By default, all sources are downloaded.
Only specified sources are downloaded, if one or more of the following options are mentioned.

     -k [TAG]: Download kernel/kernel-4.9 source and optionally sync to TAG
           and download kernel/nvgpu source and sync to the same TAG
           and download kernel/nvidia source and sync to the same TAG
           and download hardware/nvidia/soc/t18x source and sync to the same TAG
           and download hardware/nvidia/platform/tegra/common source and sync to the same TAG
           and download hardware/nvidia/platform/t18x/common source and sync to the same TAG
           and download hardware/nvidia/platform/t18x/quill source and sync to the same TAG
           and download hardware/nvidia/soc/t210 source and sync to the same TAG
           and download hardware/nvidia/platform/t210/common source and sync to the same TAG
           and download hardware/nvidia/platform/t210/jetson source and sync to the same TAG
           and download hardware/nvidia/platform/t210/porg source and sync to the same TAG
           and download hardware/nvidia/soc/t19x source and sync to the same TAG
           and download hardware/nvidia/platform/t19x/common source and sync to the same TAG
           and download hardware/nvidia/platform/t19x/galen/bct source and sync to the same TAG
           and download hardware/nvidia/platform/t19x/galen/kernel-dts source and sync to the same TAG
           and download hardware/nvidia/soc/tegra source and sync to the same TAG
     -u [TAG]: Download u-boot source and optionally sync to TAG
```

https://docs.nvidia.com/jetson/l4t-multimedia/cross_platform_support.html

# Cross compile the Kernel

$ green@C ~/nvidia/nvidia_sdk/sources/kernel/kernel-4.9 $ git tag -l

```
green@C ~/nvidia/nvidia_sdk/sources/kernel/kernel-4.9 $ git tag -l
tegra-l4t-r31.0.1
tegra-l4t-r31.0.2
tegra-l4t-r31.1
tegra-l4t-r32.1
tegra-l4t-r32.1.update-01
tegra-l4t-r32.1.update-1
tegra-l4t-r32.2.0
tegra-l4t-r32.2.1
tegra-l4t-r32.2.2
tegra-l4t-r32.2.3-1
tegra-l4t-r32.3.1
```

https://docs.nvidia.com/jetson/l4t/index.html

# Cross compile the Kernel

$ green@C ~/nvidia/nvidia_sdk/sources/kernel/kernel-4.9 $  ./source_sync.sh -t tegra-l4t-r32.3.1

```
green@C ~/nvidia/nvidia_sdk $ ./source_sync.sh -t tegra-l4t-r32.3.1
Directory for kernel/kernel-4.9, /home/green/nvidia/nvidia_sdk/sources/kernel/kernel-4.9, already exists!
Syncing up with tag tegra-l4t-r32.3.1...
Switched to a new branch 'mybranch_2020-04-16-1587032356'
/home/green/nvidia/nvidia_sdk/sources/kernel/kernel-4.9 source sync'ed to tag tegra-l4t-r32.3.1 successfully!


Directory for kernel/nvgpu, /home/green/nvidia/nvidia_sdk/sources/kernel/nvgpu, already exists!
Syncing up with tag tegra-l4t-r32.3.1...
Switched to a new branch 'mybranch_2020-04-16-1587032357'
/home/green/nvidia/nvidia_sdk/sources/kernel/nvgpu source sync'ed to tag tegra-l4t-r32.3.1 successfully!


Directory for kernel/nvidia, /home/green/nvidia/nvidia_sdk/sources/kernel/nvidia, already exists!
Syncing up with tag tegra-l4t-r32.3.1...
Switched to a new branch 'mybranch_2020-04-16-1587032358'
/home/green/nvidia/nvidia_sdk/sources/kernel/nvidia source sync'ed to tag tegra-l4t-r32.3.1 successfully!


Directory for hardware/nvidia/soc/t18x, /home/green/nvidia/nvidia_sdk/sources/hardware/nvidia/soc/t18x, already exists!
Syncing up with tag tegra-l4t-r32.3.1...
Switched to a new branch 'mybranch_2020-04-16-1587032359'
/home/green/nvidia/nvidia_sdk/sources/hardware/nvidia/soc/t18x source sync'ed to tag tegra-l4t-r32.3.1 successfully!
```

https://docs.nvidia.com/jetson/l4t/index.html

# Cross compile the Kernel

```
green@C ~/nvidia/nvidia_sdk/sources $ ls
hardware  kernel  u-boot
green@C ~/nvidia/nvidia_sdk/sources $ cd kernel/kernel-4.9/
green@C ~/nvidia/nvidia_sdk/sources/kernel/kernel-4.9 $
```

```
export ARCH=arm64
export CROSS_COMPILE=${HOME}/Downloads/toolchain/gcc-linaro-7.3.1-2018.05-x86_64_aarch64-linux-gnu/bin/aarch64-linux-gnu-

export LOCALVERSION=-tegra
```

```
nvidia@nvidia-desktop:/usr/local/cuda/samples$ uname -a
Linux nvidia-desktop 4.9.140-tegra #1 SMP PREEMPT Mon Dec 9 22:52:02 PST 2019 aarch64 aarch64 aarch64 GNU/Linux
```

```
$ wget http://releases.linaro.org/components/toolchain/binaries/7.3-2018.05/aarch64-linux-gnu/gcc-linaro-7.3.1-2018.05-x86_64_aarch64-linux-gnu.tar.xz
```

# Get the config file

Jetson:

```
nvidia@nvidia-desktop:/usr/local/cuda/samples$ sudo cp /proc/config.gz /
[sudo] password for nvidia:
nvidia@nvidia-desktop:/usr/local/cuda/samples$
```

X86 Host:

```
green@C ~/nvidia/nvidia_sdk/sources/kernel/kernel-4.9 $ sudo gzip -d /mnt/rootfs/config.gz
green@C ~/nvidia/nvidia_sdk/sources/kernel/kernel-4.9 $ cp /mnt/rootfs/config .
'/mnt/rootfs/config' -> './config'
removed './config'
green@C ~/nvidia/nvidia_sdk/sources/kernel/kernel-4.9 $ cp config .config
'config' -> '.config'
green@C ~/nvidia/nvidia_sdk/sources/kernel/kernel-4.9 $ make menuconfig
scripts/kconfig/mconf  Kconfig
```

Make –j8

https://docs.nvidia.com/jetson/l4t/index.html

# KERNEL CROSS COMPILE

```
green@C ~/nvidia/nvidia_sdk/sources/kernel/kernel-4.9 $ make Image
  CHK     include/config/kernel.release
  CHK     include/generated/uapi/linux/version.h
  CHK     include/generated/utsrelease.h
  CHK     include/generated/bounds.h
  CHK     include/generated/timeconst.h
  CHK     include/generated/asm-offsets.h
  CALL    scripts/checksyscalls.sh
  CHK     scripts/mod/devicetable-offsets.h
  CHK     include/generated/compile.h
  CHK     kernel/config_data.h
green@C ~/nvidia/nvidia_sdk/sources/kernel/kernel-4.9 $ ls arch/arm64/boot/
.Image.cmd     .Image.gz.cmd  .gitignore     .zImage.cmd    Image          Image.gz       Makefile       dts/           install.sh     zImage
green@C ~/nvidia/nvidia_sdk/sources/kernel/kernel-4.9 $ ls arch/arm64/boot/Image
arch/arm64/boot/Image
green@C ~/nvidia/nvidia_sdk/sources/kernel/kernel-4.9 $ make dtbs
  CHK     scripts/mod/devicetable-offsets.h
  DTC     arch/arm64/boot/dts/_ddot_/_ddot_/_ddot_/_ddot_/_ddot_/_ddot_/hardware/nvidia/platform/t210/jetson/kernel-dts/tegra210-jetson-cv-base-p2597-2180-a00.dtb
  DTC     arch/arm64/boot/dts/_ddot_/_ddot_/_ddot_/_ddot_/_ddot_/_ddot_/hardware/nvidia/platform/t210/jetson/kernel-dts/tegra210-jetson-cv-p2597-2180-a00.dtb
  DTC     arch/arm64/boot/dts/_ddot_/_ddot_/_ddot_/_ddot_/_ddot_/_ddot_/hardware/nvidia/platform/t210/jetson/kernel-dts/tegra210-jetson-cv-p2597-2180-imx274-hdmi.dtb
  DTC     arch/arm64/boot/dts/_ddot_/_ddot_/_ddot_/_ddot_/_ddot_/_ddot_/hardware/nvidia/platform/t210/jetson/kernel-dts/tegra210-jetson-cv-p2597-2180-a00-auo-1080p-edp.dtb
  DTC     arch/arm64/boot/dts/_ddot_/_ddot_/_ddot_/_ddot_/_ddot_/_ddot_/hardware/nvidia/platform/t210/jetson/kernel-dts/tegra210-jetson-tx1-p2597-2180-a01-android-devkit.dtb
  DTC     arch/arm64/boot/dts/_ddot_/_ddot_/_ddot_/_ddot_/_ddot_/_ddot_/hardware/nvidia/platform/t210/jetson/kernel-dts/tegra210-jetson-tx1-p2597-2180-a01-devkit.dtb
  DTC     arch/arm64/boot/dts/_ddot_/_ddot_/_ddot_/_ddot_/_ddot_/_ddot_/hardware/nvidia/platform/t210/jetson/kernel-dts/tegra210-jetson-tx1-imx274-dp-hdmi.dtb
  DTC     arch/arm64/boot/dts/_ddot_/_ddot_/_ddot_/_ddot_/_ddot_/_ddot_/hardware/nvidia/platform/t210/jetson/kernel-dts/tegra210-jetson-e-base-p2595-0000-a00.dtb
  DTC     arch/arm64/boot/dts/_ddot_/_ddot_/_ddot_/_ddot_/_ddot_/_ddot_/hardware/nvidia/platform/t210/jetson/kernel-dts/tegra210-jetson-e-p2595-0000-a00-00.dtb
  DTC     arch/arm64/boot/dts/_ddot_/_ddot_/_ddot_/_ddot_/_ddot_/_ddot_/hardware/nvidia/platform/t210/jetson/kernel-dts/tegra210-jetson-tx1-p2597-2180-a02-devkit-24x7.dtb
  DTC     arch/arm64/boot/dts/_ddot_/_ddot_/_ddot_/_ddot_/_ddot_/_ddot_/hardware/nvidia/platform/t210/jetson/kernel-dts/tegra210-jetson-tx1-p2597-2180-a01-devkit_as_TM660M.dtb
  DTC     arch/arm64/boot/dts/_ddot_/_ddot_/_ddot_/_ddot_/_ddot_/_ddot_/hardware/nvidia/platform/t210/porg/kernel-dts/tegra210-p3448-0000-p3449-0000-a00.dtb
  DTC     arch/arm64/boot/dts/_ddot_/_ddot_/_ddot_/_ddot_/_ddot_/_ddot_/hardware/nvidia/platform/t210/porg/kernel-dts/tegra210-p3448-0000-p3449-0000-a01.dtb
  DTC     arch/arm64/boot/dts/_ddot_/_ddot_/_ddot_/_ddot_/_ddot_/_ddot_/hardware/nvidia/platform/t210/porg/kernel-dts/tegra210-p3448-0000-p3449-0000-a02.dtb
  DTC     arch/arm64/boot/dts/_ddot_/_ddot_/_ddot_/_ddot_/_ddot_/_ddot_/hardware/nvidia/platform/t210/porg/kernel-dts/tegra210-p3448-0002-p3449-0000-a02.dtb
```

green@C ~/nvidia/nvidia_sdk/sources/kernel/kernel-4.9 $  cp arch/arm64/boot/Image /mnt/rootfs/boot/

# Cross compile Multi Media API

```
export TARGET_ROOTFS=/mnt/rootfs

sudo ln -sf /mnt/rootfs/usr/lib/aarch64-linux-gnu /usr/lib/aarch64-linux-gnu

sudo ln -sf /usr/lib/aarch64-linux-gnu/libpthread.so  /lib/libpthread.so.0

sudo ln -sf /mnt/rootfs/lib/aarch64-linux-gnu /lib/aarch64-linux-gnu

sudo ln -sf /mnt/rootfs/lib/aarch64-linux-gnu/libc.so.6 /lib/libc.so.6

sudo ln -sf /mnt/rootfs/usr/lib/aarch64-linux-gnu/libc_nonshared.a  /usr/lib/libc_nonshared.a

sudo ln -sf /lib/aarch64-linux-gnu/ld-2.27.so /lib/ld-linux-aarch64.so.1
```

https://developer.nvidia.com/nvidia-jetson-linux-multimediaapireference

# Cross compile Multi Media API

```
green@C ~/Downloads/nvidia/sdkm_downloads/tegra_multimedia_api $ git diff
diff --git a/samples/Rules.mk b/samples/Rules.mk
index 7977ab1..6dd362d 100644
--- a/samples/Rules.mk
+++ b/samples/Rules.mk
@@ -76,7 +76,7 @@ NM            = $(AT) $(CROSS_COMPILE)nm
 STRIP         = $(AT) $(CROSS_COMPILE)strip
 OBJCOPY       = $(AT) $(CROSS_COMPILE)objcopy
 OBJDUMP       = $(AT) $(CROSS_COMPILE)objdump
-NVCC          = $(AT) $(CUDA_PATH)/bin/nvcc -ccbin $(filter-out $(AT), $(CPP))
+NVCC          = $(AT) $(TARGET_ROOTFS)$(CUDA_PATH)/bin/nvcc -ccbin $(filter-out $(AT), $(CPP))

 # Specify the logical root directory for headers and libraries.
 ifneq ($(TARGET_ROOTFS),)
@@ -96,7 +96,8 @@ CPPFLAGS += -std=c++11 \
        -I"$(ALGO_TRT_DIR)" \
        -I"$(TARGET_ROOTFS)/$(CUDA_PATH)/include" \
        -I"$(TARGET_ROOTFS)/usr/include/$(TEGRA_ARMABI)" \
-       -I"$(TARGET_ROOTFS)/usr/include/libdrm"
+       -I"$(TARGET_ROOTFS)/usr/include/libdrm" \
+       -I"$(TARGET_ROOTFS)/usr/include/opencv4"

 # All common dependent libraries
 LDFLAGS += \
```

https://developer.nvidia.com/nvidia-jetson-linux-multimediaapireference

# Cross compile Multi Media API

```
CUDA_PATH
TOP_DIR
CLASS_DIR
ALGO_CUDA_DIR
ALGO_TRT_DIR
CROSS_COMPILE
CROSS_COMPILE
AS
LD
CC
CPP
AR
NM
STRIP
OBJCOPY
OBJDUMP
NVCC
```

```
ALGO_TRT_DIR    := $(TOP_DIR)/samples/common/algorithm/trt

ifeq ($(shell uname -m), aarch64)
CROSS_COMPILE :=
else
CROSS_COMPILE ?= aarch64-unknown-linux-gnu-
endif
AS            = $(AT) $(CROSS_COMPILE)as
LD            = $(AT) $(CROSS_COMPILE)ld
CC            = $(AT) $(CROSS_COMPILE)gcc
CPP           = $(AT) $(CROSS_COMPILE)g++
AR            = $(AT) $(CROSS_COMPILE)ar
NM            = $(AT) $(CROSS_COMPILE)nm
STRIP         = $(AT) $(CROSS_COMPILE)strip
OBJCOPY       = $(AT) $(CROSS_COMPILE)objcopy
OBJDUMP       = $(AT) $(CROSS_COMPILE)objdump
NVCC          = $(AT) $(TARGET_ROOTFS)$(CUDA_PATH)/bin/nvcc -ccbin $(filter-out $(AT), $(CPP))

# Specify the logical root directory for headers and libraries.
ifneq ($(TARGET_ROOTFS),)
CPPFLAGS += --sysroot=$(TARGET_ROOTFS)
LDFLAGS += \
    -Wl,-rpath-link=$(TARGET_ROOTFS)/lib/$(TEGRA_ARMABI) \
    -Wl,-rpath-link=$(TARGET_ROOTFS)/usr/lib/$(TEGRA_ARMABI) \
    -Wl,-rpath-link=$(TARGET_ROOTFS)/usr/lib/$(TEGRA_ARMABI)/tegra \
    -Wl,-rpath-link=$(TARGET_ROOTFS)/$(CUDA_PATH)/lib64
endif

# All common header files
CPPFLAGS += -std=c++11 \
    -I"$(TOP_DIR)/include" \
    -I"$(TOP_DIR)/include/libjpeg-8b" \
    -I"$(ALGO_CUDA_DIR)" \
    -I"$(ALGO_TRT_DIR)" \
    -I"$(TARGET_ROOTFS)/$(CUDA_PATH)/include" \
    -I"$(TARGET_ROOTFS)/usr/include/$(TEGRA_ARMABI)" \
    -I"$(TARGET_ROOTFS)/usr/include/libdrm" \
    -I"$(TARGET_ROOTFS)/usr/include/opencv4"

# All common dependent libraries
LDFLAGS += \
    -lpthread -lv4l2 -lEGL -lGLESv2 -lX11 \
    -lnvbuf_utils -lnvjpeg -lnvosd -ldrm \
    -lcuda -lcudart \
    -lnvinfer -lnvparsers \
    -L"$(TARGET_ROOTFS)/$(CUDA_PATH)/lib64" \
    -L"$(TARGET_ROOTFS)/usr/lib/$(TEGRA_ARMABI)" \
    -L"$(TARGET_ROOTFS)/usr/lib/$(TEGRA_ARMABI)/tegra"
```

https://developer.nvidia.com/nvidia-jetson-linux-multimediaapireference

# Cross compile Multi Media API

## To run

$ ./video_dec_trt 2 ../../data/Video/sample_outdoor_car_1080p_10fps.h264 ../../data/Video/sample_outdoor_car_1080p_10fps.h264 H264 --trt-deployfile ../../data/Model/resnet10/resnet10.prototxt --trt-modelfile ../../data/Model/resnet10/resnet10.caffemodel --trt-mode 0

```
nvidia@nvidia-desktop:~/tegra_multimedia_api/samples/backend$ ./backend 1 ../../data/Video/sample_outdoor_car_1080p_10fps.h264 H264 \ --trt-de
leNet_one_class/GoogleNet_modified_oneClass_halfHD.caffemodel \ --trt-mode 0 --trt-proc-interval 1 -fps 10 ^C
nvidia@nvidia-desktop:~/tegra_multimedia_api/samples/backend$ ./backend 1 ../../data/Video/sample_outdoor_car_1080p_10fps.h264 H264 \
>     --trt-deployfile ../../data/Model/GoogleNet_one_class/GoogleNet_modified_oneClass_halfHD.prototxt \
>     --trt-modelfile ../../data/Model/GoogleNet_one_class/GoogleNet_modified_oneClass_halfHD.caffemodel \
>     --trt-mode 0 --trt-proc-interval 1 -fps 10
Net has batch_size, channel, net_height, net_width:1 3 540 960
mode has been set to 0(using fp16)
outputs coverage
outputs bboxes
Applying generic optimizations to the graph for inference.
Original: 142 layers
After dead-layer removal: 142 layers
After scale fusion: 142 layers
Fusing conv1/7x7_s2 with conv1/relu_7x7
Fusing conv2/3x3_reduce with conv2/relu_3x3_reduce
Fusing conv2/3x3 with conv2/relu_3x3
Fusing inception_3a/1x1 with inception_3a/relu_1x1
Fusing inception_3a/3x3_reduce with inception_3a/relu_3x3_reduce
Fusing inception_3a/3x3 with inception_3a/relu_3x3
Fusing inception_3a/5x5_reduce with inception_3a/relu_5x5_reduce
Fusing inception_3a/5x5 with inception_3a/relu_5x5
Fusing inception_3a/pool_proj with inception_3a/relu_pool_proj
Fusing inception_3b/1x1 with inception_3b/relu_1x1
```

https://developer.nvidia.com/nvidia-jetson-linux-multimediaapireference

# Cross compile Multi Media API

## Example

$ ./backend 1 ../../data/Video/sample_outdoor_car_1080p_10fps.h264 H264 --trt-deployfile ../../data/Model/GoogleNet_one_class/GoogleNet_modified_oneClass_halfHD.prototxt --trt-modelfile ../../data/Model/GoogleNet_one_class/GoogleNet_modified_oneClass_halfHD.caffemodel --trt-mode 0 --trt-proc-interval 1 -fps 10

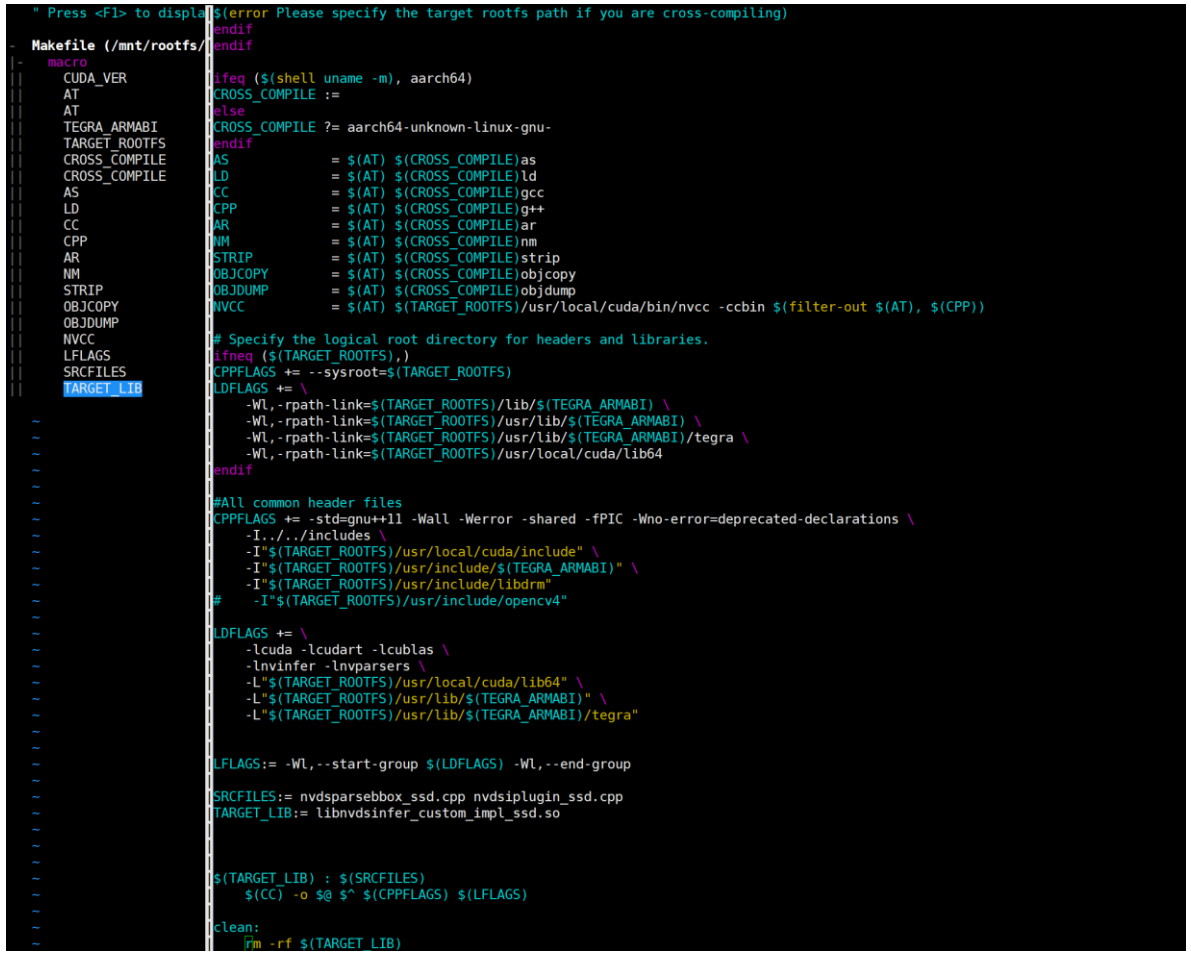
```
nvidia@nvidia-desktop:~/tegra_multimedia_api/samples/04_video_dec_trt$ ./video_dec_trt 2 ../../data/Video/sample_outdoor_car_1080p_10fps.h264      ../../data/Video/s
     --trt-modelfile ../../data/Model/resnet10/resnet10.caffemodel      --trt-mode 0
set deployfile: ../../data/Model/resnet10/resnet10.prototxt
set modefile: ../../data/Model/resnet10/resnet10.caffemodel
Net has batch_size, channel, net_height, net_width:2 3 368 640
Using cached TRT model
Deserialize required 2206394 microseconds.
outputDim c 4 w 40 h 23
outputDimsBBOX.c() 16 w 40 h 23
Opening in BLOCKING MODE
NvMMLiteOpen : Block : BlockType = 261
Opening in BLOCKING MODE
NvMMLiteOpen : Block : BlockType = 261
NVMEDIA: Reading vendor.tegra.display-size : status: 6
NVMEDIA: Reading vendor.tegra.display-size : status: 6
NvMMLiteBlockCreate : Block : BlockType = 261
Starting decoder capture loop thread
NvMMLiteBlockCreate : Block : BlockType = 261
Starting decoder capture loop thread
Video Resolution: 1920x1080
Video Resolution: 1920x1080
Resolution change successful
Input file read complete
Resolution change successful
```

https://developer.nvidia.com/nvidia-jetson-linux-multimediaapireference

# Cross compile Deepstream

export TARGET_ROOTFS=/mnt/rootfs

export CUDA_VER=10.0



# Pre-requisites for SSD

- Copy the model's label file "ssd_coco_labels.txt" from the data/ssd directory

  in TensorRT samples to this directory.

- Steps to generate the UFF model from ssd_inception_v2_coco TensorFlow frozen

  graph. These steps have been referred from TensorRT sampleUffSSD README:

  1. Make sure TensorRT's uff-converter-tf package is installed.

  2. Install tensorflow-gpu package for python:

     For dGPU:

       $ pip install tensorflow-gpu

     For Jetson, refer to https://elinux.org/Jetson_Zoo#TensorFlow

  3. Download and untar the ssd_inception_v2_coco TensorFlow trained model from http://download.tensorflow.org/models/object_detection/ssd_inception_v2_coco_2017_11_17.tar.gz

  4. Navigate to the extracted directory and convert the frozen graph to uff:

     $ cd ssd_inception_v2_coco_2017_11_17

     $ python /usr/lib/python2.7/dist-packages/uff/bin/convert_to_uff.py \

       frozen_inference_graph.pb -O NMS \

       -p /usr/src/tensorrt/samples/sampleUffSSD/config.py \

       -o sample_ssd_relu6.uff

  5. Copy sample_ssd_relu6.uff to this directory.

# Cross compile Deepstream

On the Host:

#cd /mnt/rootfs/home/nvidia/deepstream/deepstream-4.0/sources/objectDetector_SSD/nvdsinfer_custom_impl_ssd/ssd_inception_v2_coco_2017_11_17

# python /usr/lib/python3.6/dist-packages/uff/bin/convert_to_uff.py /mnt/rootfs/home/nvidia/deepstream/deepstream-4.0/sources/objectDetector_SSD/nvdsinfer_custom_impl_ssd/ssd_inception_v2_coco_2017_11_17/frozen_inference_graph.pb -O NMS -p /usr/src/tensorrt/samples/sampleUffSSD/config.py -o /mnt/rootfs/home/nvidia/deepstream/deepstream-4.0/sources/objectDetector_SSD/sample_ssd_relu6.uff

# cd /mnt/rootfs/home/nvidia/deepstream/deepstream-4.0/sources/objectDetector_SSD/

# cp /usr/src/tensorrt/data/ssd/ssd_coco_labels.txt .

On the Jetson:

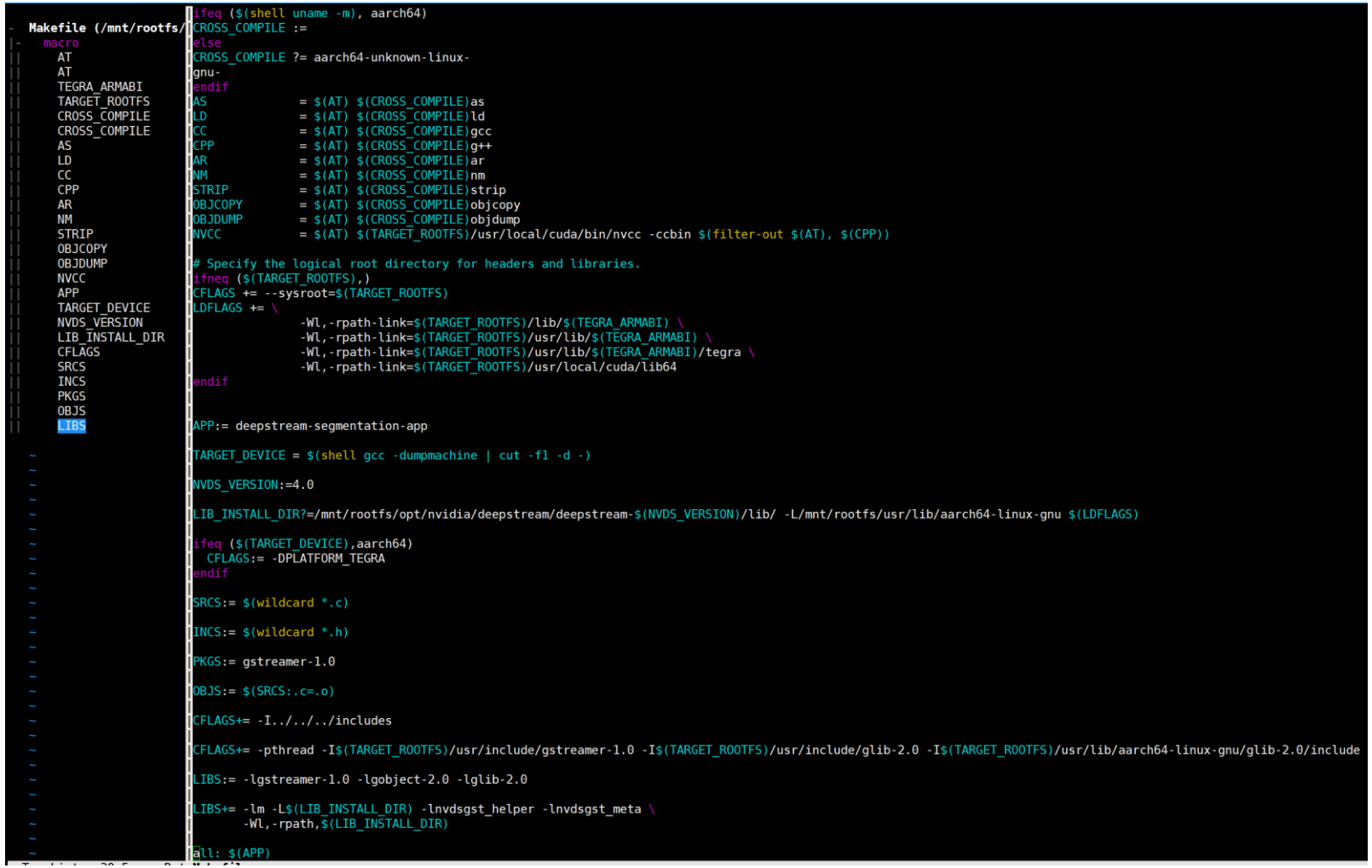
# export DISPLAY=:0

# deepstream-app -c deepstream_app_config_ssd.txt

# Cross compile DeepStream

# Cross compile DeepStream

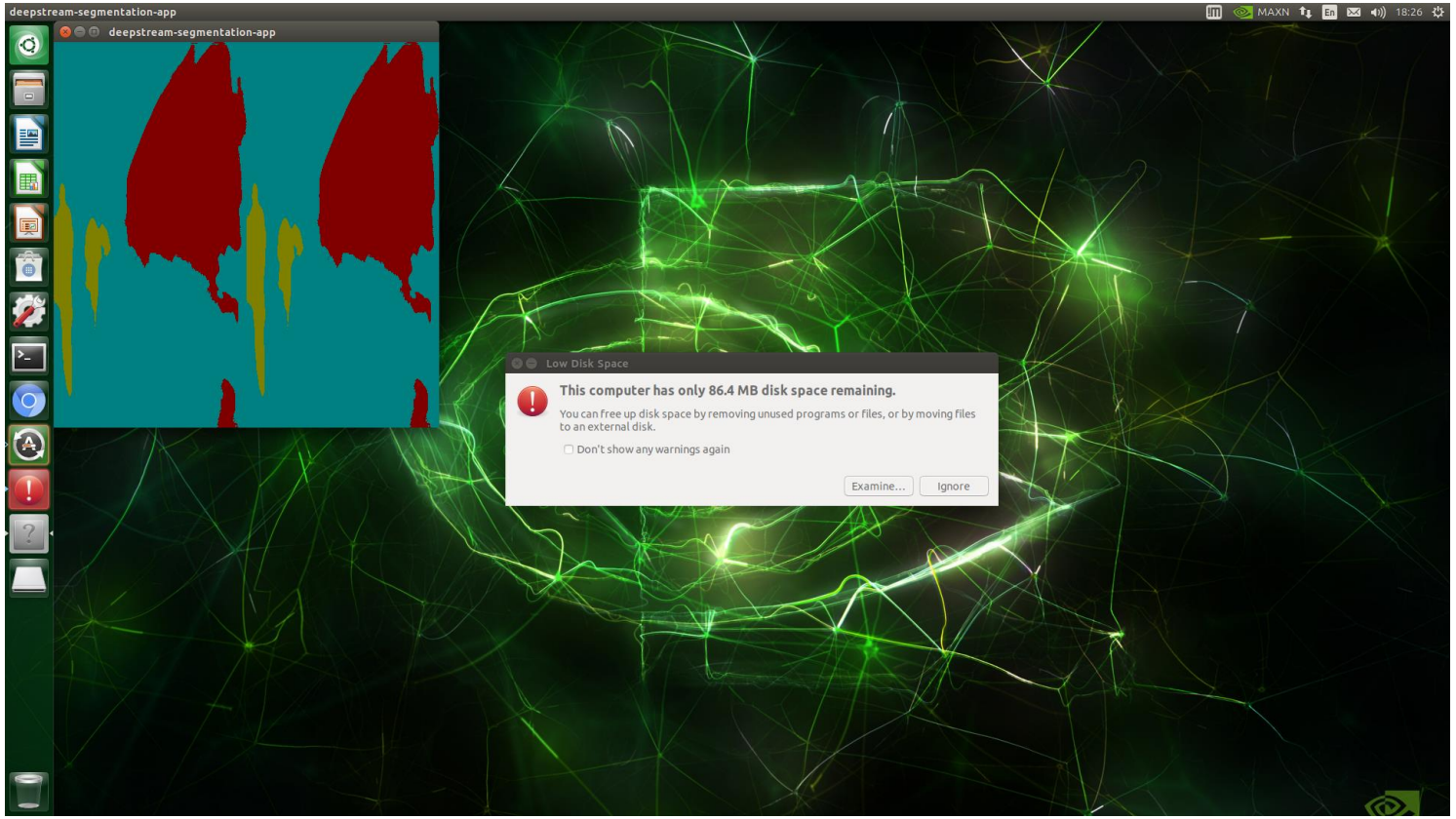
```
# cd
/mnt/rootfs/home/nvidia/deepstream/deepstream-4.0/sources/apps/sample_apps/deepstream-segmentation-test# make
```



# Cross compile DeepStream

On Jetson

```
# ./deepstream-segmentation-app
dstest_segmentation_config_semantic.txt
sample_720p.mjpeg sample_720p.mjpeg
```



# SETUP THE CROSS COMPILE

## Option 1:

```
export TARGET_ROOTFS=/mnt/rootfs
export CROSS_COMPILE=${HOME}/Downloads/toolchain/gcc-linaro-7.3.1-2018.05-x86_64_aarch64-linux-gnu/bin/aarch64-linux-gnu-
export CC = ${CROSS_COMPILE}/gcc
export CXX = ${CROSS_COMPILE}/g++
export LD = ${CROSS_COMPILE}/ld
export AR = ${CROSS_COMPILE}/ar
export AS = ${CROSS_COMPILE}/as
export RANLIB = ${CROSS_COMPILE}/ranlib
export NVCC = ${TARGET_ROOTFS}/usr/local/cuda/bin/nvcc

export LC_ALL=C
```

# SETUP THE CROSS COMPILE

## Option 2:

```
for i in /dev /dev/pts /proc /sys /run; do sudo mount -B $i /mnt$i; done
```

```
sudo mount -t proc /proc proc
sudo mount --rbind /sys  sys
sudo mount --rbind /dev dev
sudo mount --rbind /run run
```

```
sudo mount -o bind /dev  dev
sudo mount -o bind /sys sys
sudo mount -t proc /proc proc
sudo mount -o bind /run run
```

```
Sudo rm /etc/resolv.conf   echo 'nameserver 8.8.4.4' | sudo tee -a /etc/resolv.conf
```

# SETUP THE QEMU

```
for i in /dev /dev/pts /proc /sys /run; do sudo mount -B $i /mnt$i; done
```

```
sudo mount -o bind /dev  dev
sudo mount -o bind /sys sys
sudo mount -t proc /proc proc
sudo mount -o bind /run run
sudo chroot .
```

https://github.com/zhj-buffer/Cross-Compile-Jetson

# CUDA Cross Compile & Demo

$ cd /mnt/rootfs/usr/local/cuda/bin/

$ ./cuda-install-samples-10.0.sh ~/

```
Copying samples to /home/green/NVIDIA_CUDA-10.0_Samples now...
Finished copying samples.
green@C /mnt/rootfs/usr/local/cuda/bin $
```

$ cd ~/NVIDIA_CUDA-10.0_Samples

$ make -j8

$  cp bin/aarch64/linux/release/ /mnt/rootfs/home/nvidia/ -rf

# Verify on Jetson

```
nvidia@nvidia-desktop:/usr/local/cuda/samples/1_Utilities/deviceQuery$ ~/release/deviceQuery
/home/nvidia/release/deviceQuery Starting...

 CUDA Device Query (Runtime API) version (CUDART static linking)

Detected 1 CUDA Capable device(s)

Device 0: "Xavier"
  CUDA Driver Version / Runtime Version          10.0 / 10.0
  CUDA Capability Major/Minor version number:    7.2
  Total amount of global memory:                 15815 MBytes (16583041024 bytes)
  ( 8) Multiprocessors, ( 64) CUDA Cores/MP:     512 CUDA Cores
  GPU Max Clock rate:                            1377 MHz (1.38 GHz)
  Memory Clock rate:                             1377 Mhz
  Memory Bus Width:                              256-bit
  L2 Cache Size:                                 524288 bytes
  Maximum Texture Dimension Size (x,y,z)         1D=(131072), 2D=(131072, 65536), 3D=(16384, 16384, 16384)
  Maximum Layered 1D Texture Size, (num) layers  1D=(32768), 2048 layers
  Maximum Layered 2D Texture Size, (num) layers  2D=(32768, 32768), 2048 layers
  Total amount of constant memory:               65536 bytes
  Total amount of shared memory per block:       49152 bytes
  Total number of registers available per block: 65536
  Warp size:                                     32
  Maximum number of threads per multiprocessor:  2048
  Maximum number of threads per block:           1024
  Max dimension size of a thread block (x,y,z): (1024, 1024, 64)
  Max dimension size of a grid size    (x,y,z): (2147483647, 65535, 65535)
  Maximum memory pitch:                          2147483647 bytes
  Texture alignment:                             512 bytes
  Concurrent copy and kernel execution:          Yes with 1 copy engine(s)
  Run time limit on kernels:                     No
  Integrated GPU sharing Host Memory:            Yes
  Support host page-locked memory mapping:       Yes
  Alignment requirement for Surfaces:            Yes
  Device has ECC support:                        Disabled
  Device supports Unified Addressing (UVA):      Yes
  Device supports Compute Preemption:            Yes
  Supports Cooperative Kernel Launch:            Yes
  Supports MultiDevice Co-op Kernel Launch:      Yes
  Device PCI Domain ID / Bus ID / location ID:   0 / 0 / 0
  Compute Mode:
     < Default (multiple host threads can use ::cudaSetDevice() with device simultaneously) >

deviceQuery, CUDA Driver = CUDART, CUDA Driver Version = 10.0, CUDA Runtime Version = 10.0, NumDevs = 1
Result = PASS
nvidia@nvidia-desktop:/usr/local/cuda/samples/1_Utilities/deviceQuery$
```

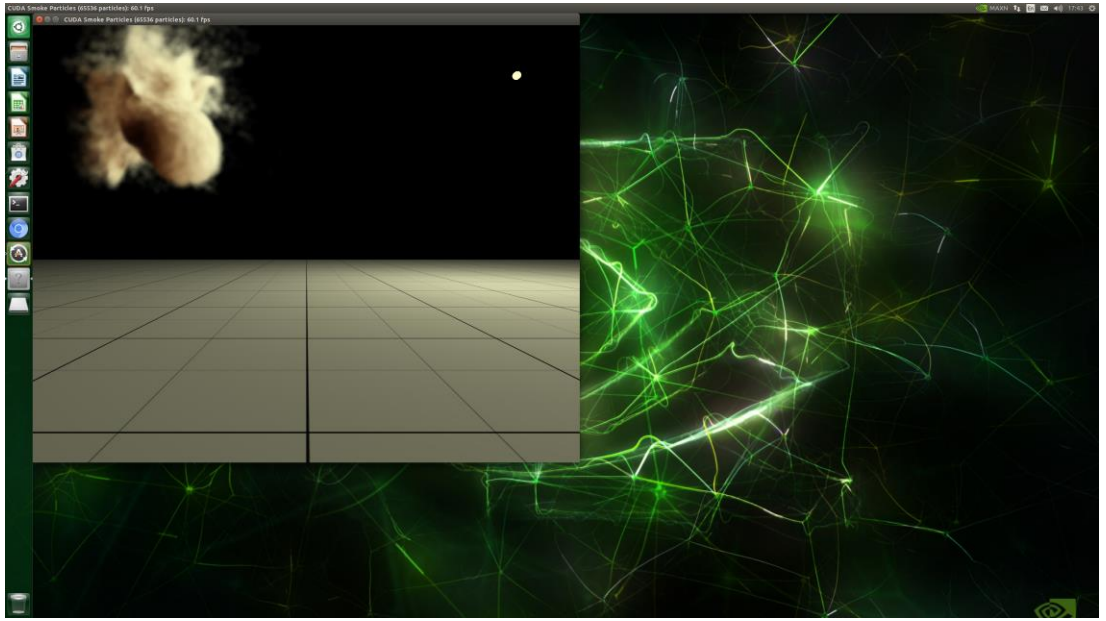
# DeviceQuery

$ cd /usr/local/cuda/samples/1_Utilities/deviceQuery
$ sudo make
$ ./deviceQuery

CUDA Capability 7.2

```
nvidia@nvidia-desktop:/usr/local/cuda/samples/1_Utilities/deviceQuery$ ./deviceQuery
./deviceQuery Starting...

 CUDA Device Query (Runtime API) version (CUDART static linking)

Detected 1 CUDA Capable device(s)

Device 0: "Xavier"
  CUDA Driver Version / Runtime Version          10.0 / 10.0
  CUDA Capability Major/Minor version number:    7.2
  Total amount of global memory:                 15815 MBytes (16583041024 bytes)
  ( 8) Multiprocessors, ( 64) CUDA Cores/MP:     512 CUDA Cores
  GPU Max Clock rate:                            1377 MHz (1.38 GHz)
  Memory Clock rate:                             1377 Mhz
  Memory Bus Width:                              256-bit
  L2 Cache Size:                                 524288 bytes
  Maximum Texture Dimension Size (x,y,z)         1D=(131072), 2D=(131072, 65536), 3D=(16384, 16
384, 16384)
  Maximum Layered 1D Texture Size, (num) layers  1D=(32768), 2048 layers
  Maximum Layered 2D Texture Size, (num) layers  2D=(32768, 32768), 2048 layers
  Total amount of constant memory:               65536 bytes
  Total amount of shared memory per block:       49152 bytes
  Total number of registers available per block: 65536
  Warp size:                                     32
  Maximum number of threads per multiprocessor:  2048
  Maximum number of threads per block:           1024
  Max dimension size of a thread block (x,y,z): (1024, 1024, 64)
  Max dimension size of a grid size    (x,y,z): (2147483647, 65535, 65535)
  Maximum memory pitch:                          2147483647 bytes
  Texture alignment:                             512 bytes
  Concurrent copy and kernel execution:          Yes with 1 copy engine(s)
  Run time limit on kernels:                     No
  Integrated GPU sharing Host Memory:            Yes
  Support host page-locked memory mapping:       Yes
  Alignment requirement for Surfaces:            Yes
  Device has ECC support:                        Disabled
  Device supports Unified Addressing (UVA):      Yes
  Device supports Compute Preemption:            Yes
  Supports Cooperative Kernel Launch:            Yes
  Supports MultiDevice Co-op Kernel Launch:      Yes
  Device PCI Domain ID / Bus ID / location ID:   0 / 0 / 0
  Compute Mode:
     < Default (multiple host threads can use ::cudaSetDevice() with device simultaneously) >

deviceQuery, CUDA Driver = CUDART, CUDA Driver Version = 10.0, CUDA Runtime Version = 10.0, Num
Devs = 1
Result = PASS
nvidia@nvidia-desktop:/usr/local/cuda/samples/1_Utilities/deviceQuery$
```

# CUDA Cross Compile & Demo



```
nvidia@nvidia-desktop:/usr/local/cuda/samples$ ~/release/smokeParticles
CUDA Smoke Particles Starting...

NOTE: The CUDA Samples are not meant for performance measurements. Results may vary when GPU Boost is enabled.

Loaded './5_Simulations/smokeParticles/data/floortile.ppm', 256 x 256 pixels
GPU Device 0: "Xavier" with compute capability 7.2

^C
nvidia@nvidia-desktop:/usr/local/cuda/samples$
```

$ nvidia@nvidia-desktop:/usr/local/cuda/samples$ ~/release/smokeParticles

# Compile Caffe & Test

## Pre-request for Caffe

wget https://github.com/Kitware/CMake/releases/download/v3.17.1/cmake-3.17.1.tar.gz

apt-get install libssl-dev

./configure

Make –j16

Make install

/usr/local/bin/cmake

https://github.com/zhj-buffer/Cross-Compile-Jetson

http://caffe.berkeleyvision.org/installation.html

# Compile Caffe & Test

## Pre-request for Caffe

git clone https://github.com/BVLC/caffe.git

https://cmake.org/download/

sudo apt install libatlas-base-dev

sudo apt install libatlas-dev

sudo apt install libopenblas-dev

sudo apt install libsnappy-dev

sudo apt install libleveldb-dev

sudo apt install liblmdb-dev

apt install libboost-all-dev

apt install libgflags-dev

apt install libgoogle-glog-dev

sudo rm /etc/resolv.conf  echo 'nameserver 8.8.4.4' | sudo tee -a /etc/resolv.conf

https://github.com/zhj-buffer/Cross-Compile-Jetson

http://caffe.berkeleyvision.org/installation.html

# Compile Caffe & Test

**Host:**

cd caffe

vim ../cmake/Cuda.cmake

set(Caffe_known_gpu_archs "72")

change CV_LOAD_IMAGE_COLOR to cv::IMREAD_COLOR
change CV_LOAD_IMAGE_GRAYSCALE to cv::IMREAD_GRAYSCALE

mkdir build

/usr/local/bin/cmake ../

make all

make test

make pycaffe

cd /media/green/M0/rootfs-xavier/home/nvidia/caffe/build

sudo cp lib /mnt/rootfs/home/nvidia/caffe/build/ -rf

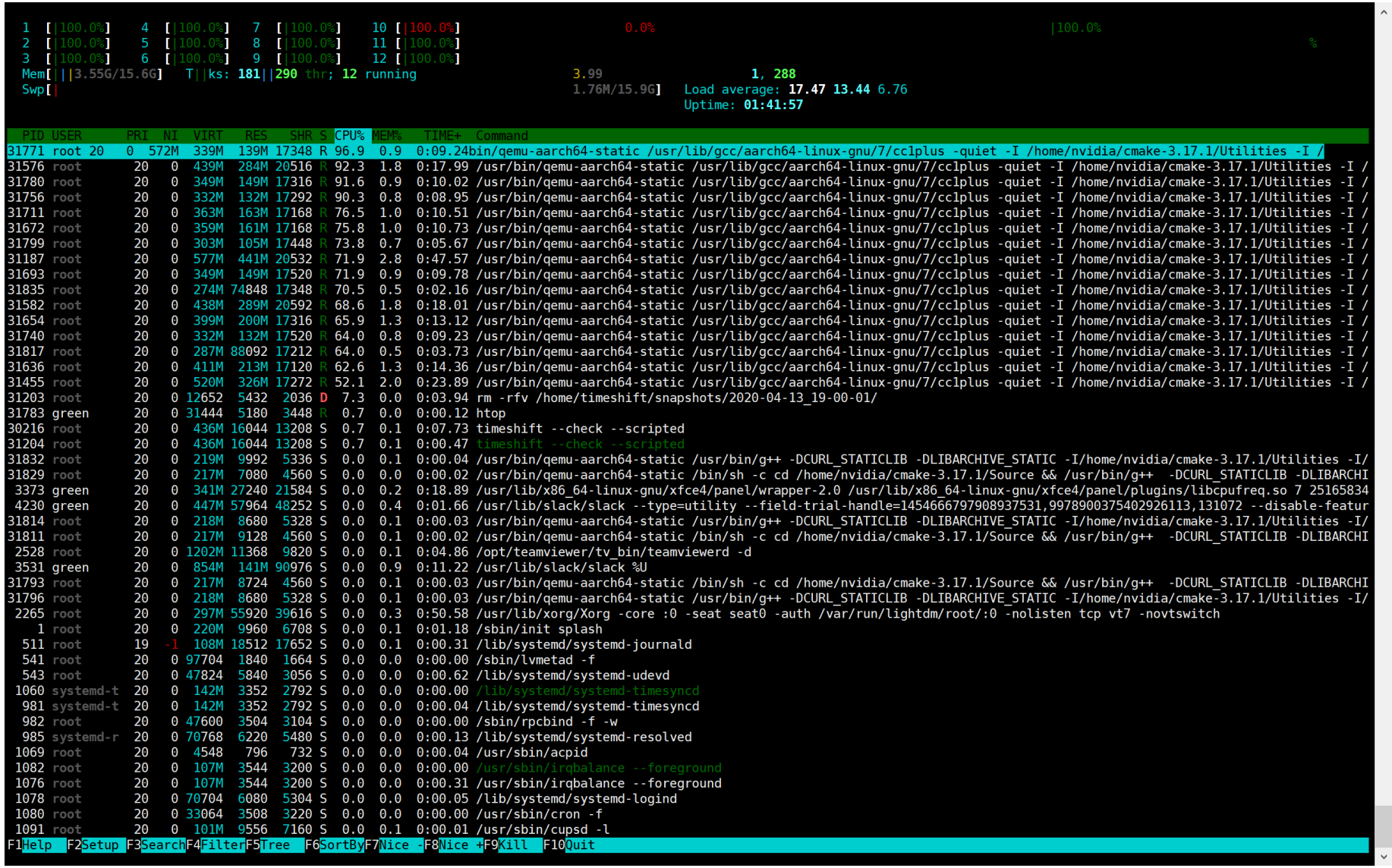
sudo cp ../python/ /mnt/rootfs/home/nvidia/caffe/ -rf

**Jetson:**

export PYTHONPATH=/home/nvidia/caffe/python:$PYTHONPATH

Python

Import caffe


1 [|100.0%] 4 [|100.0%] 7 [|100.0%] 10 [|100.0%] 0.0% |100.0%
2 [|100.0%] 5 [|100.0%] 8 [|100.0%] 11 [|100.0%]
3 [|100.0%] 6 [|100.0%] 9 [|100.0%] 12 [|100.0%]
Mem[|||3.55G/15.6G] T||ks: 181||290 thr; 12 running 3.99 1, 288
Swp[| 1.76M/15.9G] Load average: 17.47 13.44 6.76
Uptime: 01:41:57
PID USER PRI NI VIRT RES SHR S CPU% MEM% TIME+ Command
31771 root 20 0 572M 339M 139M 17348 R 96.9 0.9 0:09.24 bin/qemu-aarch64-static /usr/lib/gcc/aarch64-linux-gnu/7/cc1plus -quiet -I /home/nvidia/cmake-3.17.1/Utilities -I /
31576 root 20 0 439M 284M 20516 R 92.3 1.8 0:17.99 /usr/bin/qemu-aarch64-static /usr/lib/gcc/aarch64-linux-gnu/7/cc1plus -quiet -I /home/nvidia/cmake-3.17.1/Utilities -I /
31780 root 20 0 349M 149M 17316 R 91.6 0.9 0:10.02 /usr/bin/qemu-aarch64-static /usr/lib/gcc/aarch64-linux-gnu/7/cc1plus -quiet -I /home/nvidia/cmake-3.17.1/Utilities -I /
31756 root 20 0 332M 132M 17292 R 90.3 0.8 0:08.95 /usr/bin/qemu-aarch64-static /usr/lib/gcc/aarch64-linux-gnu/7/cc1plus -quiet -I /home/nvidia/cmake-3.17.1/Utilities -I /
31711 root 20 0 363M 163M 17168 R 76.5 1.0 0:10.51 /usr/bin/qemu-aarch64-static /usr/lib/gcc/aarch64-linux-gnu/7/cc1plus -quiet -I /home/nvidia/cmake-3.17.1/Utilities -I /
31672 root 20 0 359M 161M 17168 R 75.8 1.0 0:10.73 /usr/bin/qemu-aarch64-static /usr/lib/gcc/aarch64-linux-gnu/7/cc1plus -quiet -I /home/nvidia/cmake-3.17.1/Utilities -I /
31799 root 20 0 303M 105M 17448 R 73.8 0.7 0:05.67 /usr/bin/qemu-aarch64-static /usr/lib/gcc/aarch64-linux-gnu/7/cc1plus -quiet -I /home/nvidia/cmake-3.17.1/Utilities -I /
31187 root 20 0 577M 441M 20532 R 71.9 2.8 0:47.57 /usr/bin/qemu-aarch64-static /usr/lib/gcc/aarch64-linux-gnu/7/cc1plus -quiet -I /home/nvidia/cmake-3.17.1/Utilities -I /
31693 root 20 0 349M 149M 17520 R 71.9 0.9 0:09.78 /usr/bin/qemu-aarch64-static /usr/lib/gcc/aarch64-linux-gnu/7/cc1plus -quiet -I /home/nvidia/cmake-3.17.1/Utilities -I /
31835 root 20 0 274M 74848 17348 R 70.5 0.5 0:02.16 /usr/bin/qemu-aarch64-static /usr/lib/gcc/aarch64-linux-gnu/7/cc1plus -quiet -I /home/nvidia/cmake-3.17.1/Utilities -I /
31582 root 20 0 438M 289M 20592 R 68.6 1.8 0:18.01 /usr/bin/qemu-aarch64-static /usr/lib/gcc/aarch64-linux-gnu/7/cc1plus -quiet -I /home/nvidia/cmake-3.17.1/Utilities -I /
31654 root 20 0 399M 200M 17316 R 65.9 1.3 0:13.12 /usr/bin/qemu-aarch64-static /usr/lib/gcc/aarch64-linux-gnu/7/cc1plus -quiet -I /home/nvidia/cmake-3.17.1/Utilities -I /
31740 root 20 0 332M 132M 17520 R 64.0 0.8 0:09.23 /usr/bin/qemu-aarch64-static /usr/lib/gcc/aarch64-linux-gnu/7/cc1plus -quiet -I /home/nvidia/cmake-3.17.1/Utilities -I /
31817 root 20 0 287M 88092 17212 R 64.0 0.5 0:03.73 /usr/bin/qemu-aarch64-static /usr/lib/gcc/aarch64-linux-gnu/7/cc1plus -quiet -I /home/nvidia/cmake-3.17.1/Utilities -I /
31636 root 20 0 411M 213M 17120 R 62.6 1.3 0:14.36 /usr/bin/qemu-aarch64-static /usr/lib/gcc/aarch64-linux-gnu/7/cc1plus -quiet -I /home/nvidia/cmake-3.17.1/Utilities -I /
31455 root 20 0 520M 326M 17272 R 52.1 2.0 0:23.89 /usr/bin/qemu-aarch64-static /usr/lib/gcc/aarch64-linux-gnu/7/cc1plus -quiet -I /home/nvidia/cmake-3.17.1/Utilities -I /
31203 root 20 0 12652 5432 2036 D 7.3 0.0 0:03.94 rm -rfv /home/timeshift/snapshots/2020-04-13_19-00-01/
31783 green 20 0 31444 5180 3448 R 0.7 0.0 0:00.12 htop
30216 root 20 0 436M 16044 13208 S 0.7 0.1 0:07.73 timeshift --check --scripted
31204 root 20 0 436M 16044 13208 S 0.7 0.1 0:00.47 timeshift --check --scripted
31832 root 20 0 219M 9992 5336 S 0.0 0.1 0:00.04 /usr/bin/qemu-aarch64-static /usr/bin/g++ -DCURL_STATICLIB -DLIBARCHIVE_STATIC -I/home/nvidia/cmake-3.17.1/Utilities -I/
31829 root 20 0 217M 7080 4560 S 0.0 0.0 0:00.02 /usr/bin/qemu-aarch64-static /bin/sh -c cd /home/nvidia/cmake-3.17.1/Source && /usr/bin/g++ -DCURL_STATICLIB -DLIBARCHI
3373 green 20 0 341M 27240 21584 S 0.0 0.2 0:18.89 /usr/lib/x86_64-linux-gnu/xfce4/panel/wrapper-2.0 /usr/lib/x86_64-linux-gnu/xfce4/panel/plugins/libcpufreq.so 7 25165834
4230 green 20 0 447M 57964 48252 S 0.0 0.4 0:01.66 /usr/lib/slack/slack --type=utility --field-trial-handle=1454666797908937531,9978900375402926113,131072 --disable-featur
31814 root 20 0 218M 8680 5328 S 0.0 0.1 0:00.03 /usr/bin/qemu-aarch64-static /usr/bin/g++ -DCURL_STATICLIB -DLIBARCHIVE_STATIC -I/home/nvidia/cmake-3.17.1/Utilities -I/
31811 root 20 0 217M 9128 4560 S 0.0 0.1 0:00.02 /usr/bin/qemu-aarch64-static /bin/sh -c cd /home/nvidia/cmake-3.17.1/Source && /usr/bin/g++ -DCURL_STATICLIB -DLIBARCHI
2528 root 20 0 1202M 11368 9820 S 0.0 0.1 0:04.86 /opt/teamviewer/tv_bin/teamviewerd -d
3531 green 20 0 854M 141M 90976 S 0.0 0.9 0:11.22 /usr/lib/slack/slack %U
31793 root 20 0 217M 8724 4560 S 0.0 0.1 0:00.03 /usr/bin/qemu-aarch64-static /bin/sh -c cd /home/nvidia/cmake-3.17.1/Source && /usr/bin/g++ -DCURL_STATICLIB -DLIBARCHI
31796 root 20 0 218M 8680 5328 S 0.0 0.1 0:00.03 /usr/bin/qemu-aarch64-static /usr/bin/g++ -DCURL_STATICLIB -DLIBARCHIVE_STATIC -I/home/nvidia/cmake-3.17.1/Utilities -I/
2265 root 20 0 297M 55920 39616 S 0.0 0.3 0:50.58 /usr/lib/xorg/Xorg -core :0 -seat seat0 -auth /var/run/lightdm/root/:0 -nolisten tcp vt7 -novtswitch
1 root 20 0 220M 9960 6708 S 0.0 0.1 0:01.18 /sbin/init splash
511 root 19 -1 108M 18512 17652 S 0.0 0.1 0:00.31 /lib/systemd/systemd-journald
541 root 20 0 97704 1840 1664 S 0.0 0.0 0:00.00 /sbin/lvmetad -f
543 root 20 0 47824 5840 3056 S 0.0 0.0 0:00.62 /lib/systemd/systemd-udevd
1060 systemd-t 20 0 142M 3352 2792 S 0.0 0.0 0:00.00 /lib/systemd/systemd-timesyncd
981 systemd-t 20 0 142M 3352 2792 S 0.0 0.0 0:00.04 /lib/systemd/systemd-timesyncd
982 root 20 0 47600 3504 3104 S 0.0 0.0 0:00.00 /sbin/rpcbind -f -w
985 systemd-r 20 0 70768 6220 5480 S 0.0 0.0 0:00.13 /lib/systemd/systemd-resolved
1069 root 20 0 4548 796 732 S 0.0 0.0 0:00.04 /usr/sbin/acpid
1082 root 20 0 107M 3544 3200 S 0.0 0.0 0:00.00 /usr/sbin/irqbalance --foreground
1076 root 20 0 107M 3544 3200 S 0.0 0.0 0:00.31 /usr/sbin/irqbalance --foreground
1078 root 20 0 70704 6080 5304 S 0.0 0.0 0:00.05 /lib/systemd/systemd-logind
1080 root 20 0 33064 3508 3220 S 0.0 0.0 0:00.00 /usr/sbin/cron -f
1091 root 20 0 101M 9556 7160 S 0.0 0.1 0:00.01 /usr/sbin/cupsd -l
F1Help F2Setup F3SearchF4FilterF5Tree F6SortByF7Nice -F8Nice +F9Kill F10Quit

# Compile Caffe & test

```
--
-- ****************** Caffe Configuration Summary ******************
-- General:
--   Version           :   1.0.0
--   Git               :   1.0-136-g9b891540
--   System            :   Linux
--   C++ compiler      :   /usr/bin/c++
--   Release CXX flags :   -O3 -DNDEBUG -fPIC -Wall -Wno-sign-compare -Wno-uninitialized
--   Debug CXX flags   :   -g -fPIC -Wall -Wno-sign-compare -Wno-uninitialized
--   Build type        :   Release
--
--   BUILD_SHARED_LIBS :   ON
--   BUILD_python      :   ON
--   BUILD_matlab      :   OFF
--   BUILD_docs        :   ON
--   CPU_ONLY          :   OFF
--   USE_OPENCV        :   ON
--   USE_LEVELDB       :   ON
--   USE_LMDB          :   ON
--   USE_NCCL          :   OFF
--   ALLOW_LMDB_NOLOCK :   OFF
--   USE_HDF5          :   ON
--
-- Dependencies:
--   BLAS              :   Yes (Atlas)
--   Boost             :   Yes (ver. 1.65)
--   glog              :   Yes
--   gflags            :   Yes
--   protobuf          :   Yes (ver. 3.0.0)
--   lmdb              :   Yes (ver. 0.9.21)
--   LevelDB           :   Yes (ver. 1.20)
--   Snappy            :   Yes (ver. ..)
--   OpenCV            :   Yes (ver. 4.1.1)
--   CUDA              :   Yes (ver. 10.0)
--
-- NVIDIA CUDA:
--   Target GPU(s)     :   Auto
--   GPU arch(s)       :   sm_20 sm_21 sm_30 sm_35 sm_50 sm_60 sm_61
--   cuDNN             :   Yes (ver. 7.6.3)
--
-- Python:
--   Interpreter       :   /usr/bin/python2.7 (ver. 2.7.17)
--   Libraries         :   /usr/lib/aarch64-linux-gnu/libpython2.7.so (ver 2.7.17)
--   NumPy             :   /usr/lib/python2.7/dist-packages/numpy/core/include (ver 1.13.3)
--
-- Documentaion:
--   Doxygen           :   No
--   config_file       :
--
-- Install:
--   Install path      :   /home/nvidia/caffe/caffe/build/install
--
-- Configuring done
-- Generating done
-- Build files have been written to: /home/nvidia/caffe/caffe/build
root@C:/home/nvidia/caffe/caffe/build# make -j16
[  0%] Running C++/Python protocol buffer compiler on /home/nvidia/caffe/caffe/src/caffe/proto/caffe.proto
Scanning dependencies of target caffeproto
[  1%] Building CXX object src/caffe/CMakeFiles/caffeproto.dir/__/__/include/caffe/proto/caffe.pb.cc.o
```

# Compile Caffe & Test

```
--   Version           :   1.0.0
--   Git               :   1.0-136-g9b891540-dirty
--   System            :   Linux
--   C++ compiler      :   /usr/bin/c++
--   Release CXX flags :   -O3 -DNDEBUG -fPIC -Wall -Wno-sign-compare -Wno-uninitialized
--   Debug CXX flags   :   -g -fPIC -Wall -Wno-sign-compare -Wno-uninitialized
--   Build type        :   Release
--
--   BUILD_SHARED_LIBS :   ON
--   BUILD_python      :   ON
--   BUILD_matlab      :   OFF
--   BUILD_docs        :   ON
--   CPU_ONLY          :   OFF
--   USE_OPENCV        :   ON
--   USE_LEVELDB       :   ON
--   USE_LMDB          :   ON
--   USE_NCCL          :   OFF
--   ALLOW_LMDB_NOLOCK :   OFF
--   USE_HDF5          :   ON
--
-- Dependencies:
--   BLAS              :   Yes (Atlas)
--   Boost             :   Yes (ver. 1.65)
--   glog              :   Yes
--   gflags            :   Yes
--   protobuf          :   Yes (ver. 3.0.0)
--   lmdb              :   Yes (ver. 0.9.21)
--   LevelDB           :   Yes (ver. 1.20)
--   Snappy            :   Yes (ver. ..)
--   OpenCV            :   Yes (ver. 4.1.1)
--   CUDA              :   Yes (ver. 10.0)
--
-- NVIDIA CUDA:
--   Target GPU(s)     :   Auto
--   GPU arch(s)       :   sm_72
--   cuDNN             :   Yes (ver. 7.6.3)
--
-- Python:
--   Interpreter       :   /usr/bin/python2.7 (ver. 2.7.17)
--   Libraries         :   /usr/lib/aarch64-linux-gnu/libpython2.7.so (ver 2.7.17)
--   NumPy             :   /usr/lib/python2.7/dist-packages/numpy/core/include (ver 1.13.3)
--
-- Documentaion:
--   Doxygen           :   No
--   config_file       :
--
-- Install:
--   Install path      :   /home/nvidia/caffe/build/install
--
-- Configuring done
-- Generating done
-- Build files have been written to: /home/nvidia/caffe/build
[  1%] Built target caffeproto
Scanning dependencies of target caffe
[  1%] Linking CXX shared library ../../lib/libcaffe.so
[ 98%] Built target caffe
[100%] Linking CXX shared library ../lib/_caffe.so
Creating symlink /home/nvidia/caffe/python/caffe/_caffe.so -> /home/nvidia/caffe/build/lib/_caffe.so
[100%] Built target pycaffe
root@C:/home/nvidia/caffe/build#
```

# Compile Caffe & Test

Jetson:

cd /media/green/M0/rootfs-xavier/home/nvidia/caffe

sudo cp build/tools/ /mnt/rootfs/home/nvidia/caffe/build/ -rf

sudo cp models examples data /mnt/rootfs/home/nvidia/caffe/ -rf

sudo cp build/examples/ /mnt/rootfs/home/nvidia/caffe/build/

cd caffe

./data/mnist/get_mnist.sh

./examples/mnist/create_mnist.sh

./build/tools/caffe train -solver=examples/mnist/lenet_solver.prototxt

# Compile Caffe & Test

```
nvidia@nvidia-desktop:~/caffe$ ./examples/mnist/create_mnist.sh
Creating lmdb...
I0420 10:02:35.221259 11284 db_lmdb.cpp:35] Opened lmdb examples/mnist/mnist_train_lmdb
I0420 10:02:35.221801 11284 convert_mnist_data.cpp:88] A total of 60000 items.
I0420 10:02:35.221853 11284 convert_mnist_data.cpp:89] Rows: 28 Cols: 28
I0420 10:02:36.328914 11284 convert_mnist_data.cpp:108] Processed 60000 files.
I0420 10:02:36.956005 11289 db_lmdb.cpp:35] Opened lmdb examples/mnist/mnist_test_lmdb
I0420 10:02:36.956511 11289 convert_mnist_data.cpp:88] A total of 10000 items.
I0420 10:02:36.956573 11289 convert_mnist_data.cpp:89] Rows: 28 Cols: 28
I0420 10:02:37.181118 11289 convert_mnist_data.cpp:108] Processed 10000 files.
Done.
nvidia@nvidia-desktop:~/caffe$  ./examples/mnist/train_lenet.sh
I0420 10:03:21.101969 11297 caffe.cpp:204] Using GPUs 0
I0420 10:03:21.132551 11297 caffe.cpp:209] GPU 0: Xavier
I0420 10:03:21.694084 11297 solver.cpp:45] Initializing solver from parameters:
test_iter: 100
test_interval: 500
base_lr: 0.01
display: 100
max_iter: 10000
lr_policy: "inv"
gamma: 0.0001
power: 0.75
momentum: 0.9
weight_decay: 0.0005
snapshot: 5000
snapshot_prefix: "examples/mnist/lenet"
solver_mode: GPU
device_id: 0
net: "examples/mnist/lenet_train_test.prototxt"
train_state {
  level: 0
  stage: ""
}
I0420 10:03:21.694917 11297 solver.cpp:102] Creating training net from net file: examples/mnist/lenet_train_test.prototxt
I0420 10:03:21.695561 11297 net.cpp:296] The NetState phase (0) differed from the phase (1) specified by a rule in layer mnist
I0420 10:03:21.695636 11297 net.cpp:296] The NetState phase (0) differed from the phase (1) specified by a rule in layer accuracy
I0420 10:03:21.695672 11297 net.cpp:53] Initializing net from parameters:
name: "LeNet"
state {
  phase: TRAIN
  level: 0
  stage: ""
}
layer {
  name: "mnist"
  type: "Data"
  top: "data"
  top: "label"
  include {
    phase: TRAIN
  }
  transform_param {
    scale: 0.00390625
  }
  data_param {
    source: "examples/mnist/mnist_train_lmdb"
    batch_size: 64
    backend: LMDB
```

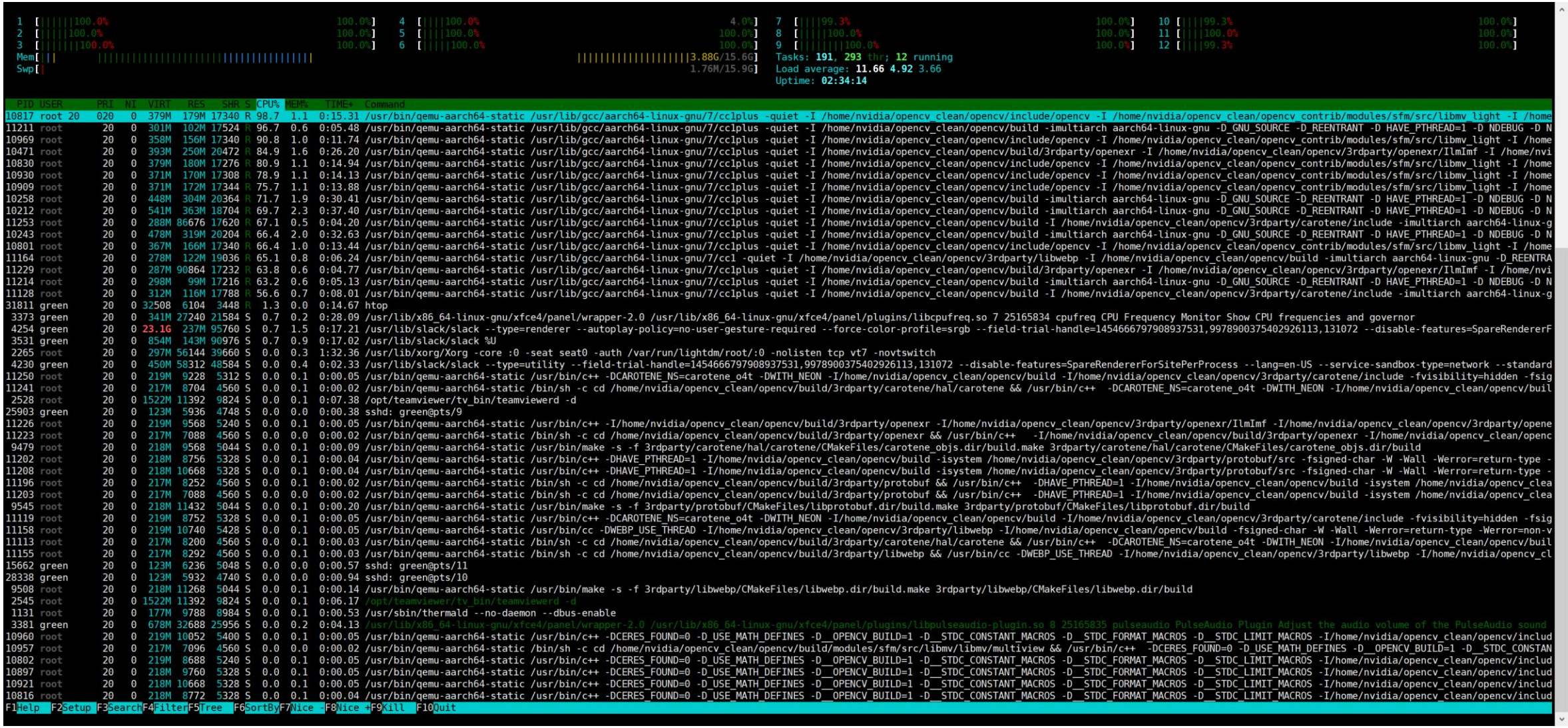
# High load

```
 1  [||||              |99.3%      100.0%]    4  [|||                |100.0%     58.9%]   7  [||                 ||||  7||||100.0%]    10 [|||                |100.0%      53.0%]
 2  [|||               |100.0%      55.0%]    5  [|||                |100.0%     58.0%]   8  [||||               |100.0%      56.0%          ||                 |100.0%      53.6%]
 3  [|||               |100.0%      56.0%]    6  [||                 |100.0%     53.0%]   9  [||               |100.0%      53.0%]    12 [|100.0%           ||100.0%      54.0%]
 Mem[|| 1.6|||||6G] ||||||s: 141, 242 thr; 2 running          |||||||  2.39                        216, 315      12 running
 Swp[|                                                                  1.76M/15.9G]   Load average: 5.73 2.23 3.82
                                                                                       Uptime: 02:06:22

  PID USER      PRI  NI  VIRT   RES   SHR S CPU% MEM%   TIME+  Command
 1486 root 20   0  593M  263M 58452 12104 R 96.7  0.4  0:01.87 /usr/bin/qemu-aarch64-static /usr/local/cuda/bin/../nvvm/bin/cicc --c++14 --gnu_version=70400 --allow_managed --unsigned
 1443 root       20   0  278M 75432 12104 R 94.7  0.5  0:02.85 /usr/bin/qemu-aarch64-static /usr/local/cuda/bin/../nvvm/bin/cicc --c++14 --gnu_version=70400 --allow_managed --unsigned
 1431 root       20   0  279M 76432 12156 R 88.2  0.5  0:02.81 /usr/bin/qemu-aarch64-static /usr/local/cuda/bin/../nvvm/bin/cicc --c++14 --gnu_version=70400 --allow_managed --unsigned
 1474 root       20   0  269M 64016 12120 R 85.5  0.4  0:02.40 /usr/bin/qemu-aarch64-static /usr/local/cuda/bin/../nvvm/bin/cicc --c++14 --gnu_version=70400 --allow_managed --unsigned
 1404 root       20   0  306M  103M 12104 R 83.6  0.6  0:03.58 /usr/bin/qemu-aarch64-static /usr/local/cuda/bin/../nvvm/bin/cicc --c++14 --gnu_version=70400 --allow_managed --unsigned
 1405 root       20   0  296M 93716 12104 R 82.2  0.6  0:03.33 /usr/bin/qemu-aarch64-static /usr/local/cuda/bin/../nvvm/bin/cicc --c++14 --gnu_version=70400 --allow_managed --unsigned
 1419 root       20   0  292M 90556 11968 R 75.0  0.6  0:03.19 /usr/bin/qemu-aarch64-static /usr/local/cuda/bin/../nvvm/bin/cicc --c++14 --gnu_version=70400 --allow_managed --unsigned
 1413 root       20   0  287M 85296 12236 R 71.7  0.5  0:03.12 /usr/bin/qemu-aarch64-static /usr/local/cuda/bin/../nvvm/bin/cicc --c++14 --gnu_version=70400 --allow_managed --unsigned
 1464 root       20   0  271M 67112 12208 R 70.4  0.4  0:02.36 /usr/bin/qemu-aarch64-static /usr/local/cuda/bin/../nvvm/bin/cicc --c++14 --gnu_version=70400 --allow_managed --unsigned
 1454 root       20   0  265M 59356 11984 R 69.7  0.4  0:02.07 /usr/bin/qemu-aarch64-static /usr/local/cuda/bin/../nvvm/bin/cicc --c++14 --gnu_version=70400 --allow_managed --unsigned
 1467 root       20   0  264M 60256 12160 R 69.7  0.4  0:02.03 /usr/bin/qemu-aarch64-static /usr/local/cuda/bin/../nvvm/bin/cicc --c++14 --gnu_version=70400 --allow_managed --unsigned
 1425 root       20   0  272M 67988 12160 R 66.4  0.4  0:02.32 /usr/bin/qemu-aarch64-static /usr/local/cuda/bin/../nvvm/bin/cicc --c++14 --gnu_version=70400 --allow_managed --unsigned
 1480 root       20   0  263M 58620 12180 R 63.8  0.4  0:01.80 /usr/bin/qemu-aarch64-static /usr/local/cuda/bin/../nvvm/bin/cicc --c++14 --gnu_version=70400 --allow_managed --unsigned
 1437 root       20   0  277M 73948 12176 R 61.2  0.5  0:02.62 /usr/bin/qemu-aarch64-static /usr/local/cuda/bin/../nvvm/bin/cicc --c++14 --gnu_version=70400 --allow_managed --unsigned
 1453 root       20   0  264M 59912 12200 R 60.5  0.4  0:02.05 /usr/bin/qemu-aarch64-static /usr/local/cuda/bin/../nvvm/bin/cicc --c++14 --gnu_version=70400 --allow_managed --unsigned
31811 green      20   0 31696  5480  3448 R  0.7  0.0  0:00.38 htop
 3373 green      20   0  341M 27240 21584 S  0.7  0.2  0:23.05 /usr/lib/x86_64-linux-gnu/xfce4/panel/wrapper-2.0 /usr/lib/x86_64-linux-gnu/xfce4/panel/plugins/libcpufreq.so 7 25165834
 2265 root       20   0  297M 56144 39660 S  0.7  0.3  1:11.05 /usr/lib/xorg/Xorg -core :0 -seat seat0 -auth /var/run/lightdm/root/:0 -nolisten tcp vt7 -novtswitch
 1471 root       20   0  217M  8612  4560 S  0.0  0.1  0:00.02 /usr/bin/qemu-aarch64-static /bin/sh -c cicc --c++14 --gnu_version=70400 --allow_managed --unsigned_chars  --diag_suppre
 1477 root       20   0  217M  7104  4560 S  0.0  0.0  0:00.02 /usr/bin/qemu-aarch64-static /bin/sh -c cicc --c++14 --gnu_version=70400 --allow_managed --unsigned_chars  --diag_suppre
 1483 root       20   0  217M  8888  4560 S  0.0  0.1  0:00.02 /usr/bin/qemu-aarch64-static /bin/sh -c cicc --c++14 --gnu_version=70400 --allow_managed --unsigned_chars  --diag_suppre
 1444 root       20   0  217M  7100  4560 S  0.0  0.0  0:00.03 /usr/bin/qemu-aarch64-static /bin/sh -c cicc --c++14 --gnu_version=70400 --allow_managed --unsigned_chars  --diag_suppre
 1398 root       20   0  217M  7100  4560 S  0.0  0.0  0:00.02 /usr/bin/qemu-aarch64-static /bin/sh -c cicc --c++14 --gnu_version=70400 --allow_managed --unsigned_chars  --diag_suppre
 1400 root       20   0  217M  7096  4560 S  0.0  0.0  0:00.02 /usr/bin/qemu-aarch64-static /bin/sh -c cicc --c++14 --gnu_version=70400 --allow_managed --unsigned_chars  --diag_suppre
 1410 root       20   0  217M  9124  4560 S  0.0  0.1  0:00.02 /usr/bin/qemu-aarch64-static /bin/sh -c cicc --c++14 --gnu_version=70400 --allow_managed --unsigned_chars  --diag_suppre
 1415 root       20   0  217M  8168  4560 S  0.0  0.0  0:00.02 /usr/bin/qemu-aarch64-static /bin/sh -c cicc --c++14 --gnu_version=70400 --allow_managed --unsigned_chars  --diag_suppre
 1422 root       20   0  217M  7108  4560 S  0.0  0.0  0:00.02 /usr/bin/qemu-aarch64-static /bin/sh -c cicc --c++14 --gnu_version=70400 --allow_managed --unsigned_chars  --diag_suppre
 1428 root       20   0  217M  7096  4560 S  0.0  0.0  0:00.02 /usr/bin/qemu-aarch64-static /bin/sh -c cicc --c++14 --gnu_version=70400 --allow_managed --unsigned_chars  --diag_suppre
 1434 root       20   0  217M  8340  4560 S  0.0  0.1  0:00.02 /usr/bin/qemu-aarch64-static /bin/sh -c cicc --c++14 --gnu_version=70400 --allow_managed --unsigned_chars  --diag_suppre
 1440 root       20   0  217M  8952  4560 S  0.0  0.1  0:00.02 /usr/bin/qemu-aarch64-static /bin/sh -c cicc --c++14 --gnu_version=70400 --allow_managed --unsigned_chars  --diag_suppre
 1447 root       20   0  217M  7096  4560 S  0.0  0.0  0:00.02 /usr/bin/qemu-aarch64-static /bin/sh -c cicc --c++14 --gnu_version=70400 --allow_managed --unsigned_chars  --diag_suppre
 1452 root       20   0  217M  7096  4560 S  0.0  0.0  0:00.02 /usr/bin/qemu-aarch64-static /bin/sh -c cicc --c++14 --gnu_version=70400 --allow_managed --unsigned_chars  --diag_suppre
 1460 root       20   0  217M  7096  4560 S  0.0  0.0  0:00.02 /usr/bin/qemu-aarch64-static /bin/sh -c cicc --c++14 --gnu_version=70400 --allow_managed --unsigned_chars  --diag_suppre
  984 root       20   0  218M 10492  4928 S  0.0  0.1  0:00.06 /usr/bin/qemu-aarch64-static /usr/local/cuda/bin/nvcc /home/nvidia/caffe/src/caffe/layers/cudnn_relu_layer.cu -c -o /hom
 1138 root       20   0  218M 10176  4928 S  0.0  0.1  0:00.06 /usr/bin/qemu-aarch64-static /usr/local/cuda/bin/nvcc /home/nvidia/caffe/src/caffe/layers/cudnn_lcn_layer.cu -c -o /home
 3381 green      20   0  678M 32688 25956 S  0.0  0.2  0:03.39 /usr/lib/x86_64-linux-gnu/xfce4/panel/wrapper-2.0 /usr/lib/x86_64-linux-gnu/xfce4/panel/plugins/libpulseaudio-plugin.so
 1306 root       20   0  218M  9572  4928 S  0.0  0.1  0:00.06 /usr/bin/qemu-aarch64-static /usr/local/cuda/bin/nvcc /home/nvidia/caffe/src/caffe/layers/batch_norm_layer.cu -c -o /hom
 1327 root       20   0  218M  9360  4904 S  0.0  0.1  0:00.05 /usr/bin/qemu-aarch64-static /usr/local/cuda/bin/nvcc /home/nvidia/caffe/src/caffe/layers/cudnn_lrn_layer.cu -c -o /home
 1238 root       20   0  218M 10968  4880 S  0.0  0.1  0:00.06 /usr/bin/qemu-aarch64-static /usr/local/cuda/bin/nvcc /home/nvidia/caffe/src/caffe/layers/cudnn_pooling_layer.cu -c -o /
 1252 root       20   0  218M 10484  4984 S  0.0  0.1  0:00.05 /usr/bin/qemu-aarch64-static /usr/local/cuda/bin/nvcc /home/nvidia/caffe/src/caffe/layers/bnll_layer.cu -c -o /home/nvid
 1161 root       20   0  218M  9332  4932 S  0.0  0.1  0:00.06 /usr/bin/qemu-aarch64-static /usr/local/cuda/bin/nvcc /home/nvidia/caffe/src/caffe/layers/crop_layer.cu -c -o /home/nvid
 1178 root       20   0  218M 10536  4928 S  0.0  0.1  0:00.05 /usr/bin/qemu-aarch64-static /usr/local/cuda/bin/nvcc /home/nvidia/caffe/src/caffe/layers/conv_layer.cu -c -o /home/nvid
 1213 root       20   0  218M  9540  4880 S  0.0  0.1  0:00.05 /usr/bin/qemu-aarch64-static /usr/local/cuda/bin/nvcc /home/nvidia/caffe/src/caffe/layers/base_data_layer.cu -c -o /home
32413 root       20   0  231M 20500 10748 S  0.0  0.1  0:00.15 /usr/bin/qemu-aarch64-static /usr/local/bin/cmake -D verbose:BOOL= -D build_configuration:STRING=Release -D generated_fi
28338 green      20   0  123M  5932  4740 S  0.0  0.0  0:00.05 sshd: green@pts/10
F1Help  F2Setup F3SearchF4FilterF5Tree  F6SortByF7Nice -F8Nice +F9Kill  F10Quit
```

# Compile Opencv & Test

```
--     PFM:                         YES
--
--   Video I/O:
--     DC1394:                      NO
--     FFMPEG:                      NO
--       avcodec:                   NO
--       avformat:                  NO
--       avutil:                    NO
--       swscale:                   NO
--       avresample:                NO
--     GStreamer:                   YES (1.14.5)
--     v4l/v4l2:                    YES (linux/videodev2.h)
--
--   Parallel framework:            pthreads
--
--   Trace:                         YES (with Intel ITT)
--
--   Other third-party libraries:
--     Lapack:                      NO
--     Eigen:                       YES (ver 3.3.4)
--     Custom HAL:                  YES (carotene (ver 0.0.1))
--     Protobuf:                    build (3.5.1)
--
--   NVIDIA CUDA:                   YES (ver 10.0, CUFFT CUBLAS)
--     NVIDIA GPU arch:             53 62 72
--     NVIDIA PTX archs:
--
--   cuDNN:                         YES (ver 7.6.3)
--
--   OpenCL:                        YES (no extra features)
--     Include path:                /home/nvidia/opencv_clean/opencv/3rdparty/include/opencl/1.2
--     Link libraries:              Dynamic load
--
--   Python 2:
--     Interpreter:                 /usr/bin/python2.7 (ver 2.7.17)
--     Libraries:                   /usr/lib/aarch64-linux-gnu/libpython2.7.so (ver 2.7.17)
--     numpy:                       /usr/lib/python2.7/dist-packages/numpy/core/include (ver 1.13.3)
--     install path:                lib/python2.7/dist-packages/cv2/python-2.7
--
--   Python 3:
--     Interpreter:                 /usr/bin/python3 (ver 3.6.9)
--     Libraries:                   /usr/lib/aarch64-linux-gnu/libpython3.6m.so (ver 3.6.9)
--     numpy:                       /usr/local/lib/python3.6/dist-packages/numpy/core/include (ver 1.18.2)
--     install path:                lib/python3.6/dist-packages/cv2/python-3.6
--
--   Python (for build):            /usr/bin/python2.7
--
--   Java:
--     ant:                         NO
--     JNI:                         NO
--     Java wrappers:               NO
--     Java tests:                  NO
--
--   Install to:                    /usr/local
-- -----------------------------------------------------------------
--
-- Configuring done
-- Generating done
-- Build files have been written to: /home/nvidia/opencv_clean/opencv/build
root@C:/home/nvidia/opencv_clean/opencv/build#
```

# Compile Opencv & Test



# Compile Opencv & Test

```
Scanning dependencies of target opencv_superres
[ 98%] Building CXX object modules/superres/CMakeFiles/opencv_superres.dir/src/super_resolution.cpp.o
[ 98%] Building CXX object modules/superres/CMakeFiles/opencv_superres.dir/src/btv_l1_cuda.cpp.o
[ 98%] Building CXX object modules/superres/CMakeFiles/opencv_superres.dir/src/frame_source.cpp.o
[ 98%] Building CXX object modules/superres/CMakeFiles/opencv_superres.dir/src/btv_l1.cpp.o
[ 98%] Building CXX object modules/superres/CMakeFiles/opencv_superres.dir/src/optical_flow.cpp.o
[ 98%] Building CXX object modules/superres/CMakeFiles/opencv_superres.dir/src/input_array_utility.cpp.o
[ 98%] Building CXX object modules/superres/CMakeFiles/opencv_superres.dir/opencl_kernels_superres.cpp.o
Scanning dependencies of target opencv_videostab
[ 98%] Building CXX object modules/videostab/CMakeFiles/opencv_videostab.dir/src/frame_source.cpp.o
[ 98%] Building CXX object modules/videostab/CMakeFiles/opencv_videostab.dir/src/fast_marching.cpp.o
[ 98%] Building CXX object modules/videostab/CMakeFiles/opencv_videostab.dir/src/inpainting.cpp.o
[ 98%] Building CXX object modules/videostab/CMakeFiles/opencv_videostab.dir/src/deblurring.cpp.o
[ 98%] Building CXX object modules/videostab/CMakeFiles/opencv_videostab.dir/src/motion_stabilizing.cpp.o
[ 98%] Building CXX object modules/videostab/CMakeFiles/opencv_videostab.dir/src/optical_flow.cpp.o
[ 98%] Building CXX object modules/videostab/CMakeFiles/opencv_videostab.dir/src/log.cpp.o
[ 98%] Building CXX object modules/videostab/CMakeFiles/opencv_videostab.dir/src/global_motion.cpp.o
[ 98%] Building CXX object modules/videostab/CMakeFiles/opencv_videostab.dir/src/stabilizer.cpp.o
[ 98%] Building CXX object modules/videostab/CMakeFiles/opencv_videostab.dir/src/outlier_rejection.cpp.o
[ 98%] Building CXX object modules/videostab/CMakeFiles/opencv_videostab.dir/src/wobble_suppression.cpp.o
[ 98%] Linking CXX shared library ../../lib/libopencv_superres.so
[ 98%] Built target opencv_superres
Scanning dependencies of target opencv_test_superres
Scanning dependencies of target opencv_perf_superres
[ 98%] Building CXX object modules/superres/CMakeFiles/opencv_test_superres.dir/test/test_main.cpp.o
[ 98%] Building CXX object modules/superres/CMakeFiles/opencv_test_superres.dir/test/test_superres.cpp.o
[ 98%] Building CXX object modules/superres/CMakeFiles/opencv_perf_superres.dir/perf/perf_main.cpp.o
[ 98%] Building CXX object modules/superres/CMakeFiles/opencv_perf_superres.dir/perf/perf_superres.cpp.o
[ 98%] Linking CXX executable ../../bin/opencv_test_superres
[ 98%] Built target opencv_test_superres
[ 99%] Linking CXX shared library ../../lib/libopencv_videostab.so
[ 99%] Built target opencv_videostab
Scanning dependencies of target opencv_test_videostab
Scanning dependencies of target opencv_python3
Scanning dependencies of target opencv_python2
[ 99%] Building CXX object modules/videostab/CMakeFiles/opencv_test_videostab.dir/test/test_stabilizer.cpp.o
[ 99%] Building CXX object modules/videostab/CMakeFiles/opencv_test_videostab.dir/test/test_main.cpp.o
[ 99%] Building CXX object modules/videostab/CMakeFiles/opencv_test_videostab.dir/test/test_motion_estimation.cpp.o
[ 99%] Building CXX object modules/python2/CMakeFiles/opencv_python2.dir/__/src2/cv2.cpp.o
[ 99%] Building CXX object modules/python3/CMakeFiles/opencv_python3.dir/__/src2/cv2.cpp.o
[ 99%] Linking CXX executable ../../bin/opencv_perf_superres
[ 99%] Built target opencv_perf_superres
[ 99%] Linking CXX executable ../../bin/opencv_test_videostab
[ 99%] Built target opencv_test_videostab
[ 99%] Linking CXX shared module ../../lib/python3/cv2.cpython-36m-aarch64-linux-gnu.so
[ 99%] Built target opencv_python3
[100%] Linking CXX shared module ../../lib/cv2.so
[100%] Built target opencv_python2
root@C:/home/nvidia/opencv_clean/opencv/build# `
>
```

# Compile Opencv & Test

/usr/local/bin/cmake -D WITH_CUDA=ON  -D BUILD_CUDA_STUBS=ON  -D OPENCV_EXTRA_MODULES_PATH=../../opencv_contrib/modules ../

Make –j16

sudo cp build/lib/cv2.so  /mnt/rootfs/usr/local/lib/python2.7/dist-packages/

export LD_LIBRARY_PATH=/home/nvidia/opencv/lib:$LD_LIBRARY_PATH

Python
Import cv2

```
nvidia@nvidia-desktop:~/opencv/lib$ python
Python 2.7.17 (default, Nov  7 2019, 10:07:09)
[GCC 7.4.0] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import cv2
>>> cv2.__version__
'4.3.0-dev'
>>>
```

# AGENDA

## Jetson product and marketing

## Why we need cross compile

## How to setup the cross-compile environment

1. Config directly via the environment variable.
2. Config in Makefile.
3. Build your simulation.

## Compile on X86 host and execute on Jetson

1. Cross compile Linux kernel.
2. Cross compile CUDA project.
3. Cross compile Multi Media API based application.
4. Cross compile Deepstream.
5. Cross compile Caffe
6. Cross compile OpenCV.

## Where can I get the prebuild version and how to build on jetson?

# JETSON DLOWNLOAD CENTER

Prebuild  Jetson Image:

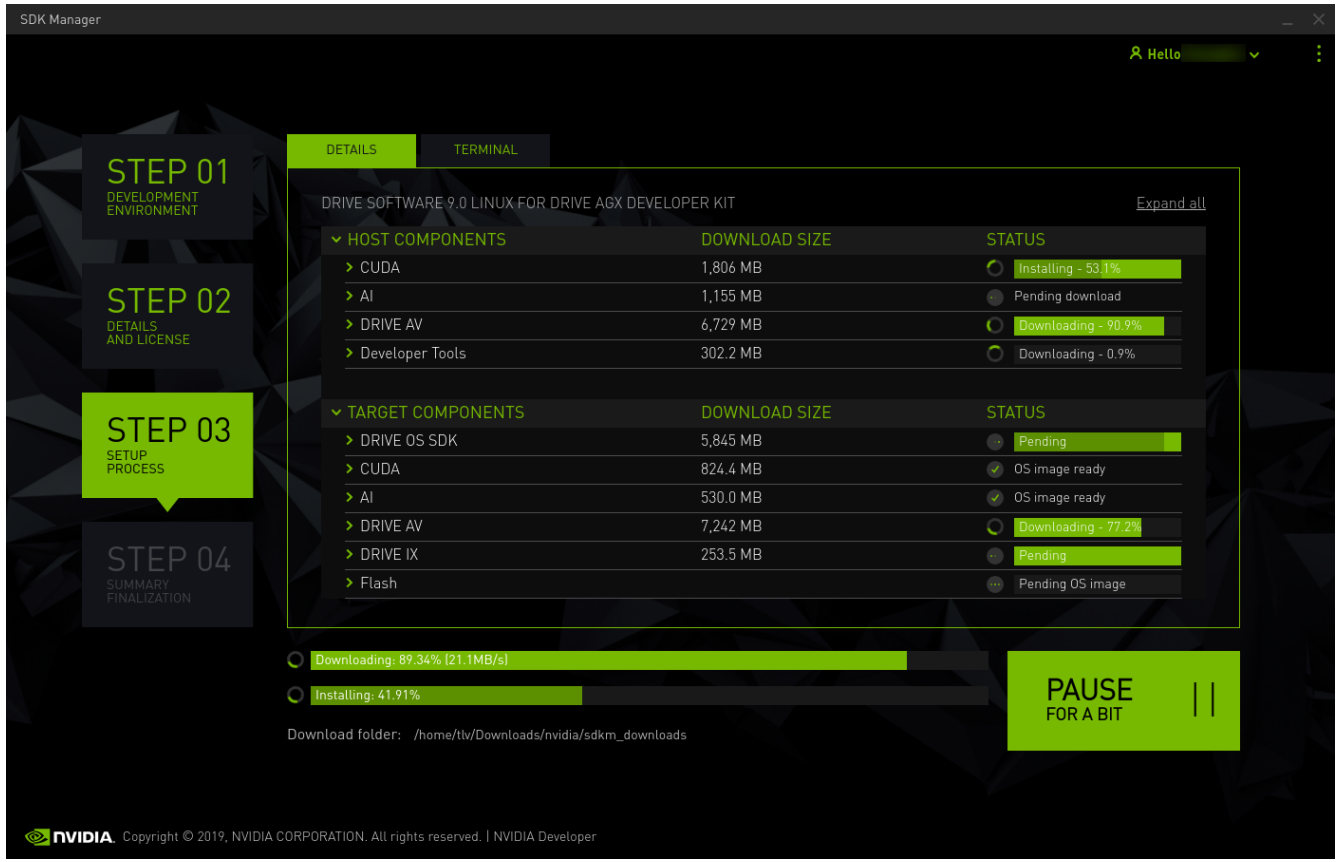
https://developer.nvidia.com/jetson-nano-sd-card-image

```
Welcome to fdisk (util-linux 2.31.1).
Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.


Command (m for help): p
Disk jetson-nano-sd-r32.1-2019-03-18.img: 12 GiB, 12884901888 bytes, 25165824 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: gpt
Disk identifier: D048AD43-24FD-4DED-B06E-7BB8ED98158C

Device                                Start      End  Sectors  Size Type
jetson-nano-sd-r32.1-2019-03-18.img1  24576 25165790 25141215   12G Linux filesystem
jetson-nano-sd-r32.1-2019-03-18.img2   2048     2303      256  128K Linux filesystem
jetson-nano-sd-r32.1-2019-03-18.img3   4096     4991      896  448K Linux filesystem
jetson-nano-sd-r32.1-2019-03-18.img4   6144     7295     1152  576K Linux filesystem
jetson-nano-sd-r32.1-2019-03-18.img5   8192     8319      128   64K Linux filesystem
jetson-nano-sd-r32.1-2019-03-18.img6  10240    10623      384  192K Linux filesystem
jetson-nano-sd-r32.1-2019-03-18.img7  12288    13439     1152  576K Linux filesystem
jetson-nano-sd-r32.1-2019-03-18.img8  14336    14463      128   64K Linux filesystem
jetson-nano-sd-r32.1-2019-03-18.img9  16384    17663     1280  640K Linux filesystem
jetson-nano-sd-r32.1-2019-03-18.img10 18432    19327      896  448K Linux filesystem
jetson-nano-sd-r32.1-2019-03-18.img11 20480    20735      256  128K Linux filesystem
jetson-nano-sd-r32.1-2019-03-18.img12 22528    22687      160   80K Linux filesystem

Partition table entries are not in disk order.

Command (m for help):
```

From here you can see the rootfs start from block 24576 with block size 512bytes. So We need to mount the image from offset 24576 * 512 = 12582912

```
root@C:~# mount -o loop,offset=12582912 jetson-nano-sd-r32.1-2019-03-18.img /mnt
root@C:~# ls /mnt/
```

# SDKMANAGER




https://developer.nvidia.com/nvidia-sdk-manager

# Backup and Store

To clone a Jetson device and flash

Clone:

- sudo ./flash.sh -r -k APP -G <clone> <target> mmcblk0p1

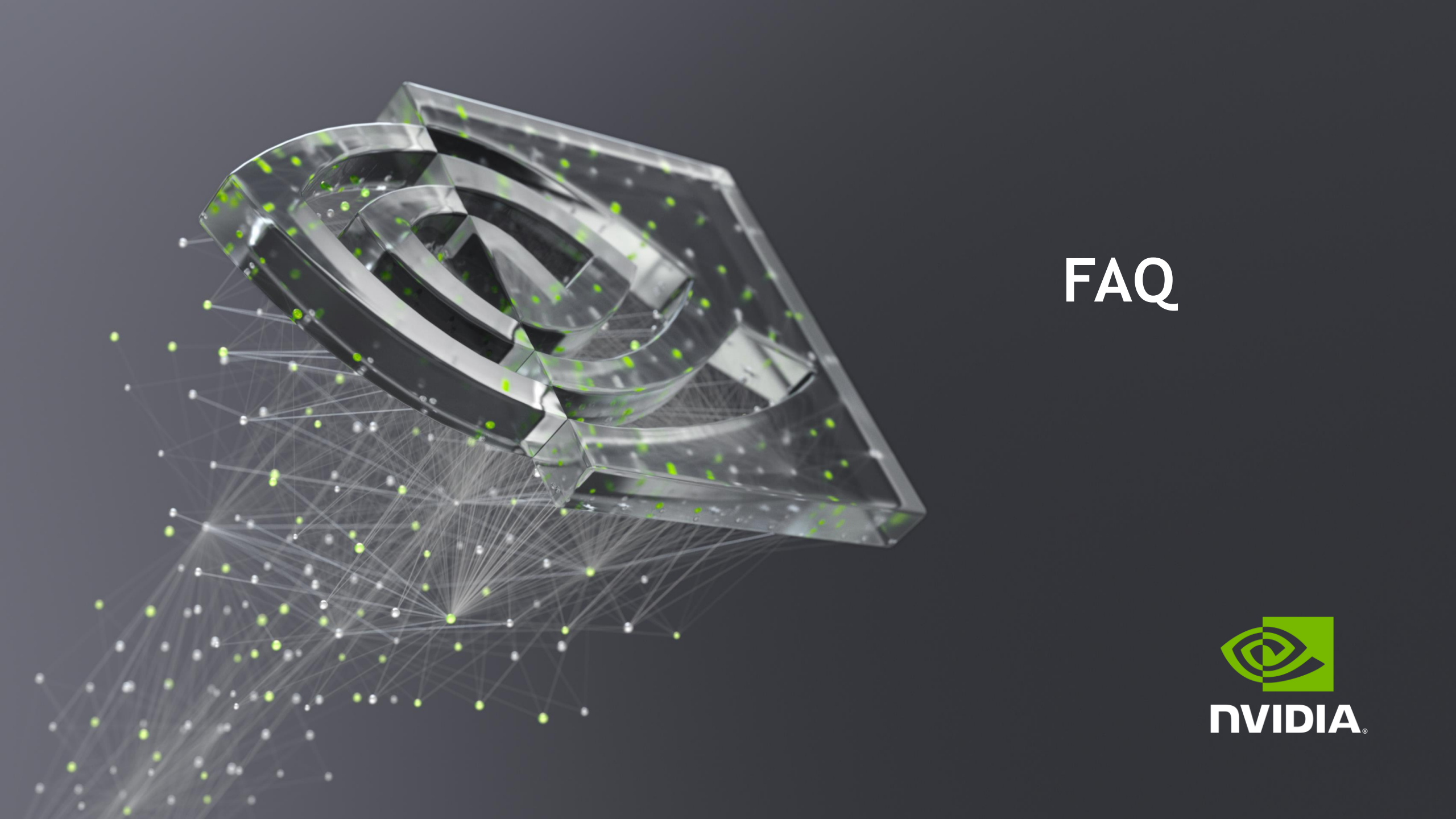
Customization:

- simg2img system.img system.img.raw
- sudo mount -t ext4 -o loop system.img.raw /mnt

Store:

- sudo img2simg system.img.raw system.img
- sudo cp system.img bootloader/system.img
- sudo ./flash.sh -r -k APP <target> mmcblk0p1
- Backup the ubuntu system and make a new Image based on it.

https://docs.nvidia.com/jetson/l4t/index.html#page/Tegra%2520Linux%2520Driver%2520Package%2520Development%2520Guide%2Fflashing.html%23



# FAQ

NVIDIA

# QUETIONS

1. How about chroot to a mounted nfs rootfs?

2. How about compiling tensorflow?