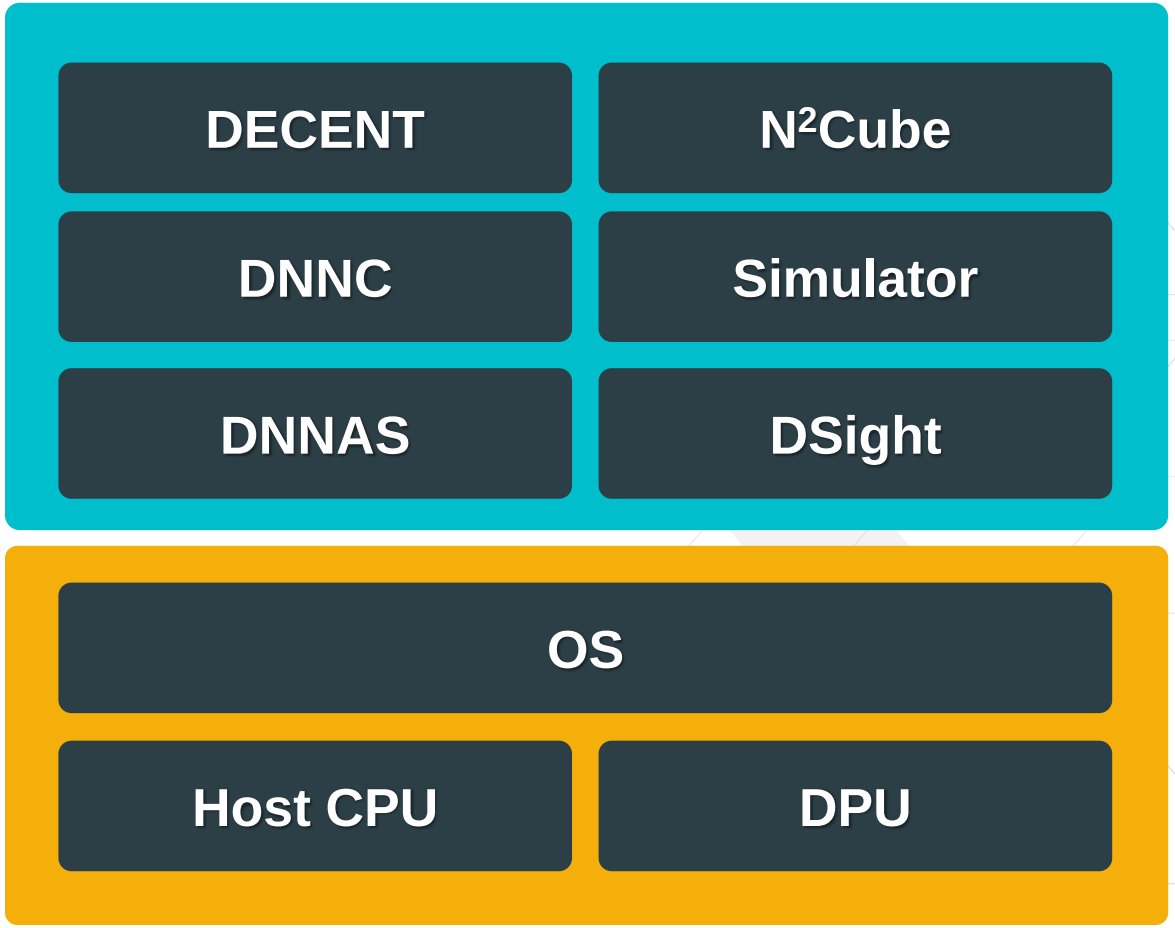
Machine Learning for Embedded Workshop

Detroit – March 12

XILINX

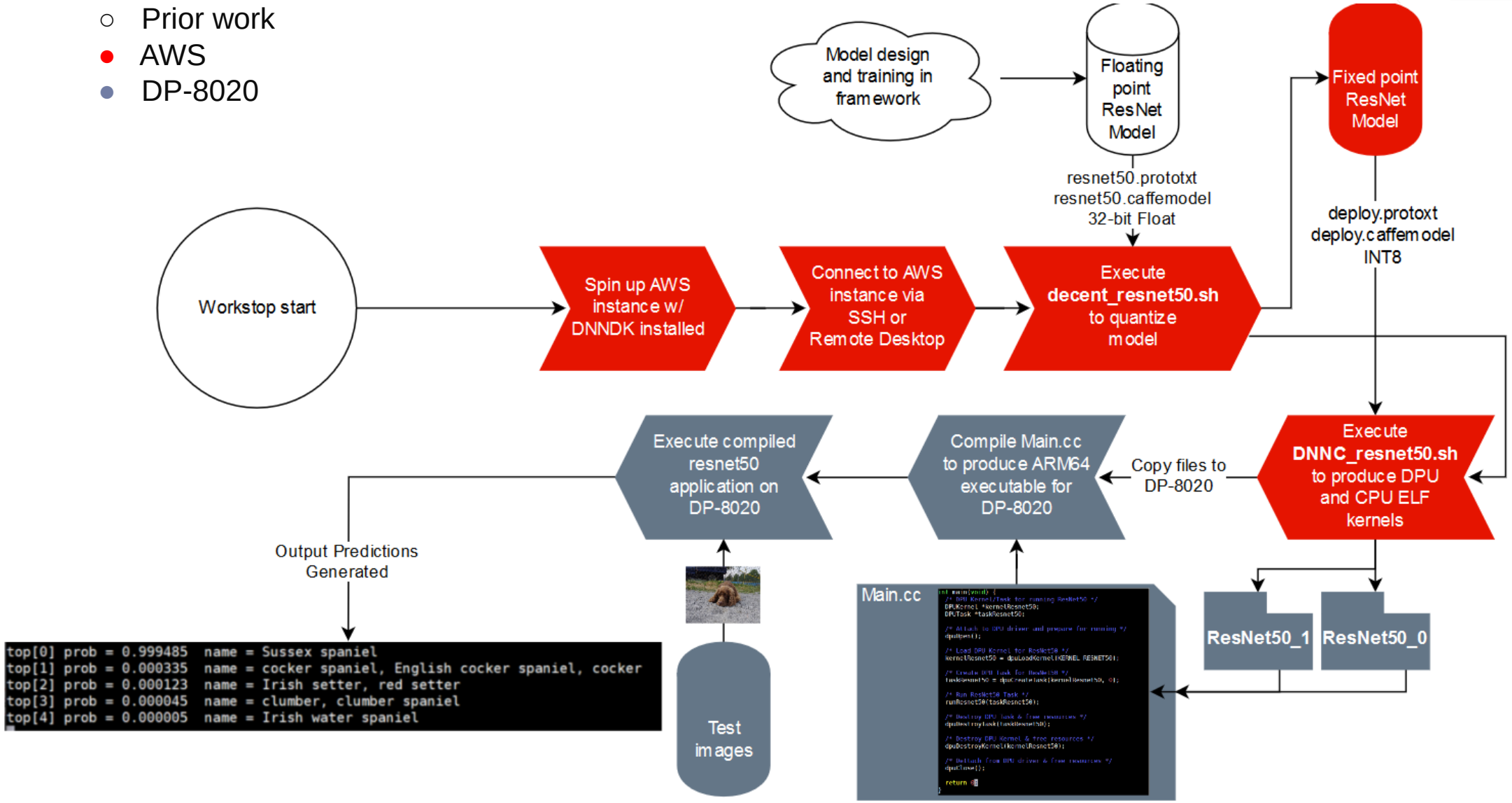
DNNDK – Deep Neural Network Development Kit

- DECENT (DEep ComprEssioN Tool)
- DNNC (Deep Neural Network Compiler)
- N^2Cube (Cube of Neural Network) Runtime
- Dsight Profiler



Workshop Flow

- Prior work
- AWS
- DP-8020



WiFi and AWS access

- **Wireless connection**
  - SSID: Renaissance_Conference Password: wings
- **Lab instructions on Github:**

https://github.com/jimheaton/Ultra96_ML_Embedded_Workshop

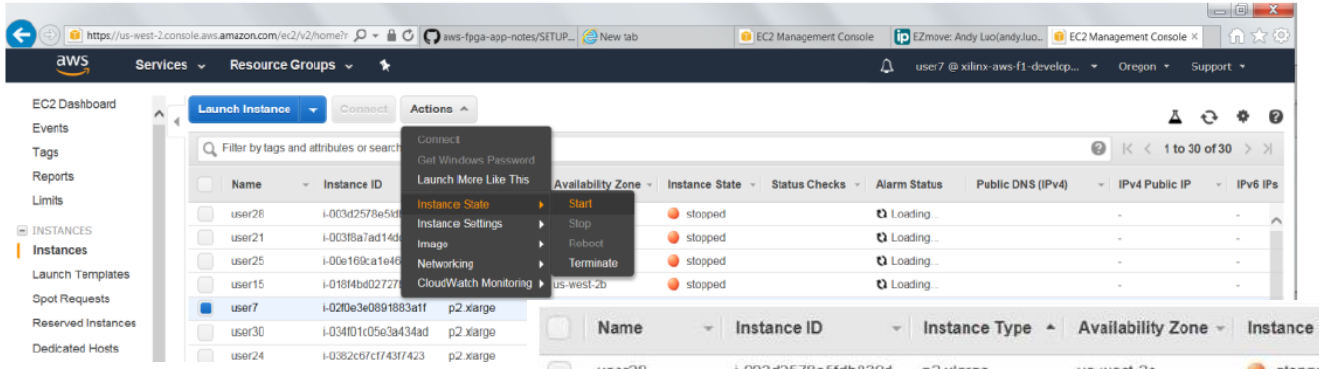
- **User your web browser to go to AWS login screen**
  - Account ID: xilinx-aws-f1-developer-labs
  - IAM user name: userxx (for example: user501)
  - Password: xlnx_ultra96
- Username for RDP/SSH is “ubuntu” password is “xlnx_ultra96”
- If you’re display doesn’t come up within 1-2 minutes enter the following in the terminal:

```
xrandr --output  DP-1 --mode 1024x768 --rate 60
```

Connect to AWS

1. https://console.aws.amazon.com/ec2/v2/home?region=ap-northeast-1#Instances:tag:Name=< IAM user name>;sort=tag:Name

2.



Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS (IPv4)	IPv4 Public IP
user28	i-003d2578e5fdb839d	p2.xlarge	us-west-2c	stopped		Loading...		-
user21	i-003f8a7ad14dc6c8b	p2.xlarge	us-west-2c	stopped		Loading...		-
user25	i-00e169ca1e46ab1c9	p2.xlarge	us-west-2c	stopped		Loading...		-
user15	i-018f4bd02727b0829	p2.xlarge	us-west-2b	stopped		Loading...		-
user7	i-02f0e3e0891883a1f	p2.xlarge	us-west-2b	running	2/2 checks...	Loading...	ec2-34-211-165-84.us-...	34.211.165.84
user30	i-034f01c05e3a434ad	p2.xlarge	us-west-2b	stopped		Loading...		-
user24	i-0382c67cf743f7423	p2.xlarge	us-west-2b	stopped		Loading...		-

XILINX

Connect to AWS (Cont.)

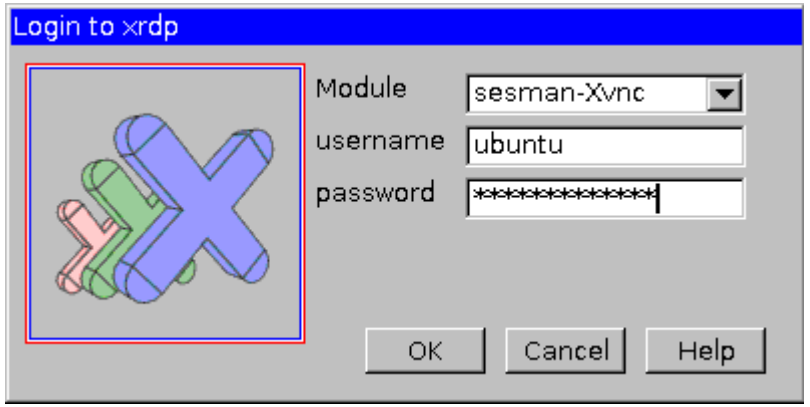
3 In the SSH client e.g. PuTTy, use the **IPv4 Public IP** of your instance

```
ssh ubuntu@<IPv4 Public IP>
```

OR

In the Remote Desktop Connection client, enter the **IPv4 Public IP** of your instance.

Then log in with username: **ubuntu** and password: **sand_xdfd**



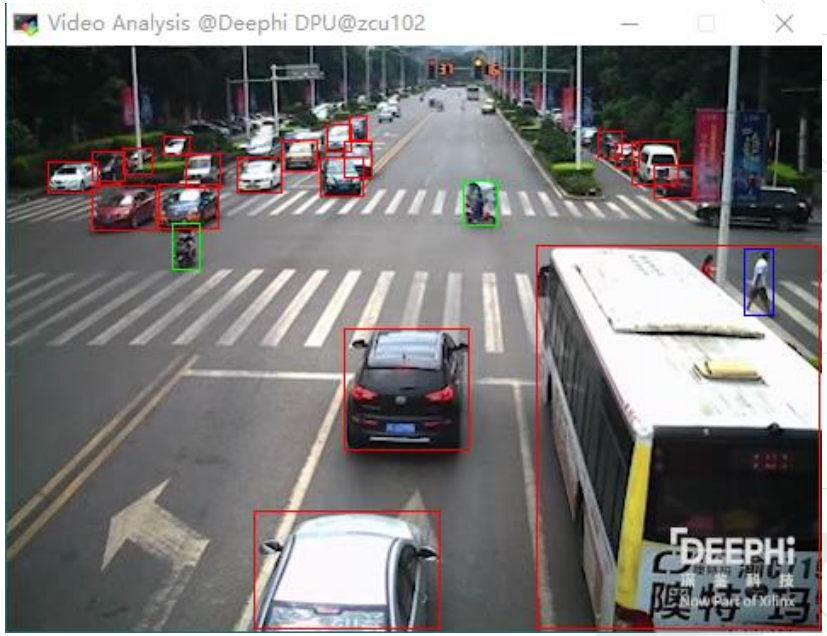
Hybrid Compilation and Execution

- **Copy dpu_resnet50_0.elf to ~/dnndk/dp-8020/samples/resnet5/model**
- **Login as a user on the DP-8020 (user: root pw: root)**
- **Run make in ~/dnndk/dp-8020/samples/resnet5 to generate resnet50**
- **Copy resnet50 to /root/samples/resnet50 of DP-8020**
  - Copy resnet50 from AWS to host PC
  - Copy resnet50 from host PC to DP-8020
- **Run**
  - **./resnet50**

Not required for today's workshop

Expected Outcome

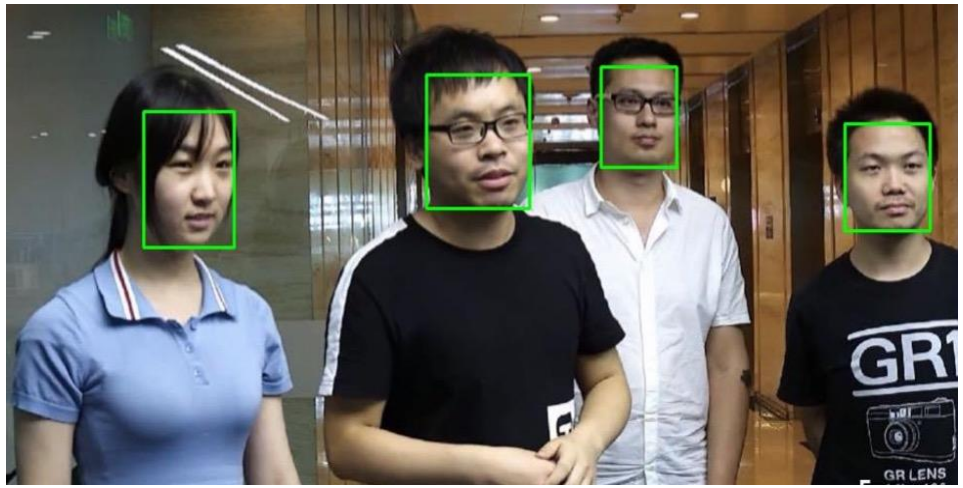
- **Resnet50**
  - Top1/5 accuracy: 0.738/0.908
  - FPS: 35
- **SSD**
  - FPS: 28



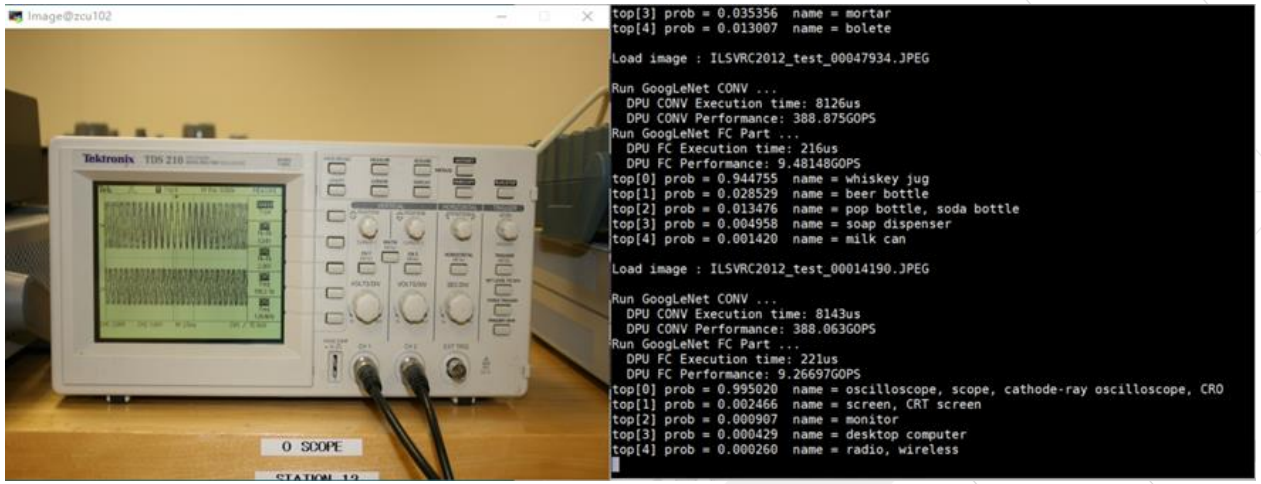
XILINX.

Expected Outcome (Cont.)

- **Densebox**
  - FPS: 30



- **Inception V1(Googlenet)**
  - Top1/5 accurary: 0.697/0.896
  - FPS: 74



Workshop Steps

- **Connecting to the AWS P2 instance**
  - You will start an EC2 P2 instance and connect to it using SSH or remote desktop
- **Experience DNNDK with Resnet50**
  - Quantize, compile and deploy a pre-trained Resnet50 model on the DP-8020
- **Optional: Go further with SSD**
  - Build a real-time multi-class object detection demo using SSD network
- **Optional: Try face detection with Densebox**
  - Build a real-time face detection demo using USB camera as input
- **Optional: Practice DNNDK with Inception V1(Googlenet)**
  - Finish the main.cc and Makefile then build and run it
- **Wrap-up and next steps**
  - Explore DNNDK after the workshop

> Follow the lab instructions step-by-step

> Lab assistants will be roaming to assist you!

Key Takeaways

1. DNNDK deploy pre-trained DNN models to Xilinx SoC easily & quickly without writing any RTL

2. DNNDK compilation are all done in AWS without installing any software

3. DNNDK supports many popular DNN models without modifying any FPGA design

Back Up

Example: SSD

- **Generate fixed-point model:**
  - Run the script in ~/dnndk/dp-8020/ssd/

output file: decent_output/deploy.caffemodel deploy.prototxt

log

```
I0910 07:03:05.698068 17141 net.cpp:222] data does not need backward computation.
I0910 07:03:05.698076 17141 net.cpp:264] This network produces output label
I0910 07:03:05.698084 17141 net.cpp:264] This network produces output mbox_conf_flatten
I0910 07:03:05.698093 17141 net.cpp:264] This network produces output mbox_loc
I0910 07:03:05.698101 17141 net.cpp:264] This network produces output mbox_priorbox
I0910 07:03:05.698196 17141 net.cpp:284] Network initialization done.
I0910 07:03:05.943905 17141 decent.cpp:307] Start Deploy
I0910 07:03:05.986526 17141 decent.cpp:315] Deploy Done!
----------------------------------------------
Output Deploy Weights: "/home/ubuntu/Compiler/ssd/decent_output/deploy.caffemodel"
Output Deploy Model:   "/home/ubuntu/Compiler/ssd/decent_output/deploy.prototxt"
```

Script

```
#/usr/bin/env bash

#working directory
work_dir=$(pwd)
#path of float model
model_dir=${work_dir}
#output directory
output_dir=${work_dir}/decent_output

./decent     fix                                    \
            -model ${model_dir}/float.prototxt      \
            -weights ${model_dir}/float.caffemodel  \
            -output_dir ${output_dir}               \
            -method 1
```

SSD -- Compilation

Before compilation, it is need to remove the Reshape layer and the subsequent layers in deploy.prototxt. (deploy.prototxt in decent_output_fix is the final file input which can be referenced)

Run the script in ~/dnndk/dp-8020/ssd/dnnc_ssd.sh

Output file: dnnc_output/dpu_ssd.elf

log

```
ubuntu@ip-172-31-18-247:~/Compiler/ssd$ ./dnnc_ssd.sh

DNNC Kernel Information

1. Overview
kernel numbers   : 1
kernel topology : kernel_graph.gv

2. Kernel Description in Detail
kernel id        : 0
kernel name      : ssd
type             : DPUKernel(Supported)
nodes            : NA
input node(s)    : conv1_1
output node(s)   : mbox_loc_94 mbox_conf_95
```

Script

```
#!/bin/bash
net=ssd
model_dir=decent_output
output_dir=dnnc_output

./dnnc --prototxt=${model_dir}/deploy.prototxt      \
       --caffemodel=${model_dir}/deploy.caffemodel \
       --output_dir=${output_dir}                   \
       --net_name=${net}                            \
       --dpu=4096FA                                 \
       --cpu_arch=arm64                             \
       --mode=debug
```

Hybrid Compilation & Execution

- **Copy dpu_resnet50_0.elf to ~/dnndk/dp-8020/samples/video_analysis/model**
- **Run make in ~/dnndk/dp-8020/samples/video_analysis to generate video_analysis**
- **Login DP-8020**
- **Copy video_analysis to /root/samples/video_analysis of DP-8020**
  - Copy video_analysis from AWS to host PC
  - Copy video_analysis from host PC to DP-8020
- **Run**
  - **./video_analysis video/structure.mp4**