차세대 반도체 디지털 혁신공유대학 사업단

POLARIS

2024년 Deep Learning Hardware 설계 경진대회

2024.03.04 (Monday)

Outlines

- About AIX2024
- Quick Start
- TODO
- Evaluation and Award

Organizing committee

- Representative organizing committee

Xuan-Truong Nguyen, Ph.D.
서울대학교

김기환, 박사 학생
서울대학교

고영훈, 박사 학생
서울대학교

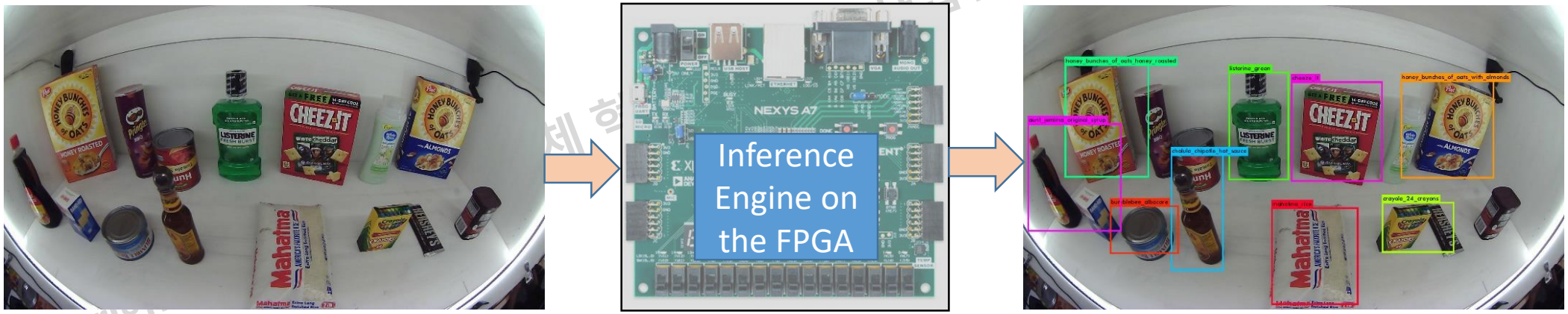
조영목, 석박사 통합 학생
서울대학교

정지윤 선생님
서울대학교

- TAs: 김기환, 고영훈, 조영목, 서울대학교
- Technical staffs: 장유진 선생님(dbwls3172@snu.ac.kr(접수 및 일정 관련)), 정지윤 선생님, 김보령
- Advisors: 이혁재교수, 이태호교수, 선우경교수, 이정원교수, 김남준교수, 이재학교수 서울대학교

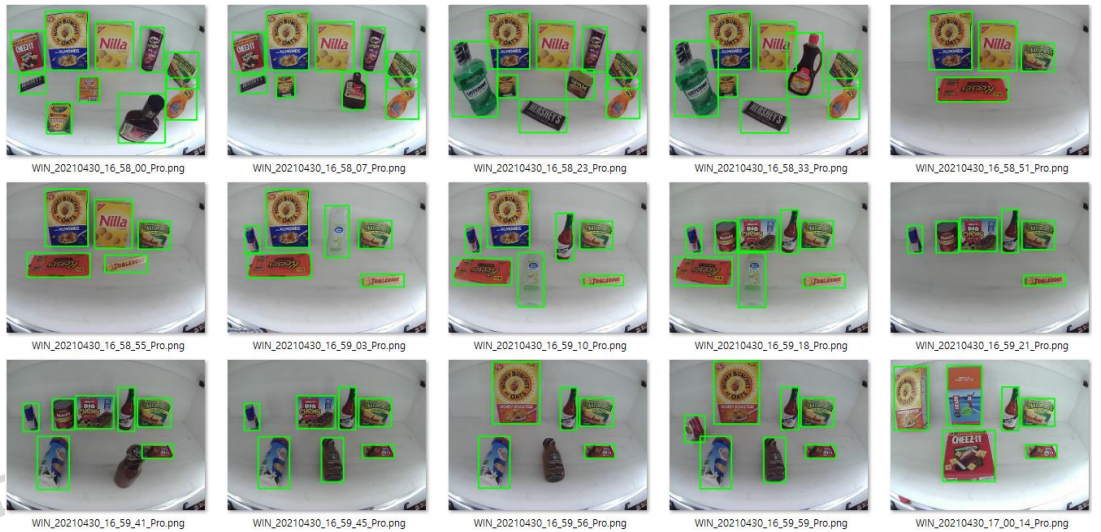
설계의 목표

무인판매대에서 상품 인식을 위하는 딥러닝 추론 하드웨어를 설계한다.

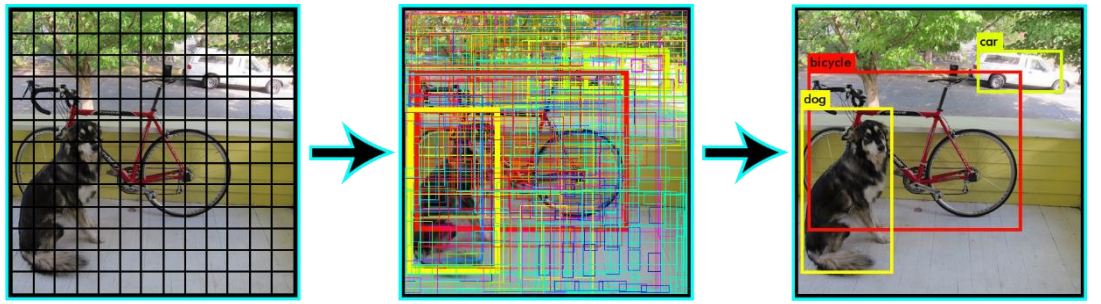


Object Detection (Application-Based)

- Training dataset
  - Collected ~100,000 images of items
  - Labeled the data
- We trained a deep neural network (DNN) for object detection based on Tiny-YOLO
  - Input: an RGB image
  - The network includes many layer types:
    - Convolutional layers w/ filters 3x3, 1x1
    - Max pooling
    - Concatenation



datasets

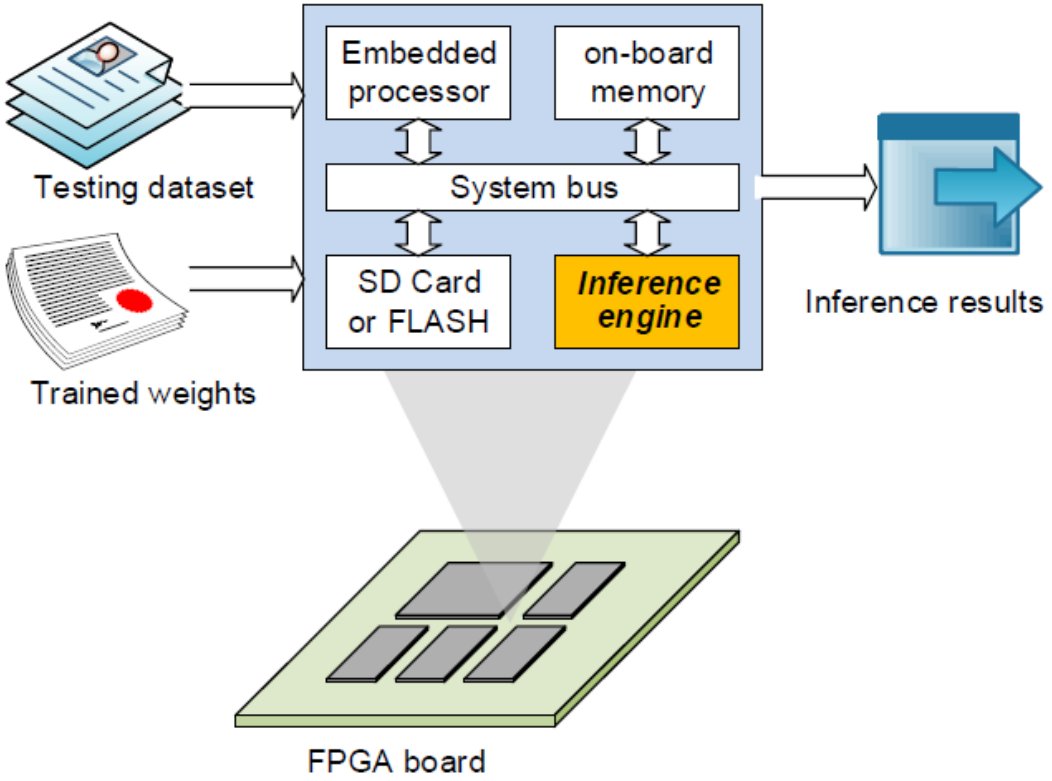


YOLO

https://pjreddie.com/darknet/yolov2/

제공되는 것과 준비할 것 (System-based)

- 추론엔진용 딥러닝 파라미터
  - Pretrained model
- 시험 데이터셋 (상품 이미지)
- 추론엔진(Inference Engine)의 reference code
  - S/W: Evaluation and Host PC (C++)
  - H/W: Components (Verilog HDL)
- Nexys A7 FPGA Board (Xilinx Artix-7 FPGA XC7A100T-1CSG324C)



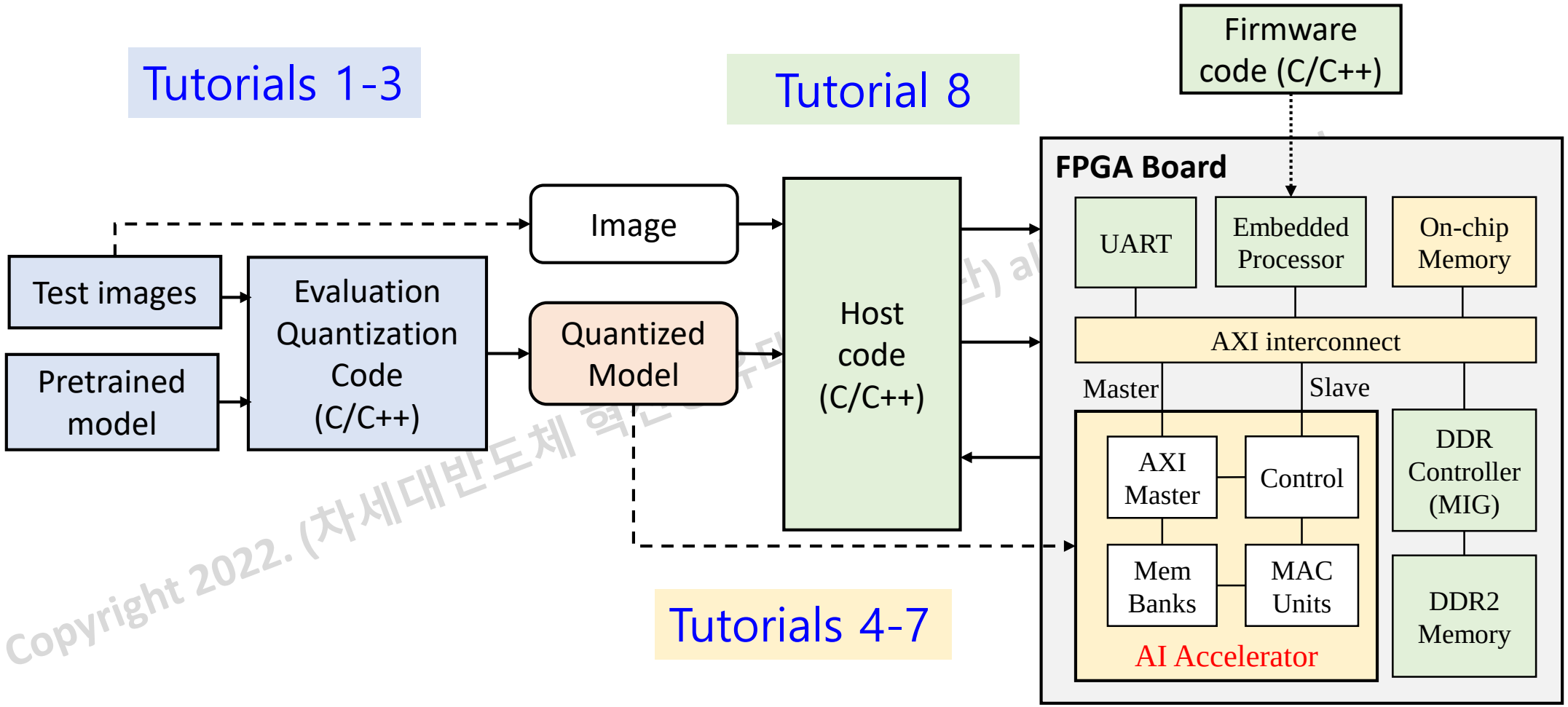
Tutorials[1]

- Tutorials are given to cover several fundamental issues in the AIX Design
  - Introduce fundamental components and their usages

#	Title	Content
1	Orientation	Introduction to AIX, code structure
2	Network	Reference S/W, network architecture, evaluation metrics
3	Quantization	Model quantization, data preparation
4	MAC	Hardware description language, computing units
5	Memory	On-chip buffer, block RAM, IP generator, MIG
6	Bus	AXI interconnect, DMA
7	Integration	System integration and verification (one layer)
8	CPU-FPGA	PC-FPGA communication

[1] **Tutorials do NOT aim to replace courses at school. Students are highly recommended to take relevant courses.**

Top Structure and Tutorials



경진대회 개최 일정

- ~~**공지 및 접수** : 24.2.8 (목)~24.2.29 (목)~~ => DONE
  - 차세대 반도체 혁신융합 7개 대학(강원대, 대구대, 서울대, 숭실대, 조선이공대, 중앙대, 포항공대)의 학부생 2~3인 구성의 팀으로 딥러닝과 하드웨어에 관심있는 누구나 참여 가능
  - 졸업생 및 본 경진대회 이전 수상자는 제외
- **오리엔테이션** : 24.3.4 (월) 17시 => TODAY
- **본선** : 24.3.4 (월) ~ 24.5.31 (금, 최종 설계 제출 마감)
  - 중간 평가 (Code, Recording Video & PPT): 2024.4.5 (금)
    - 중간평가 우수팀에게 스타벅스 쿠폰 증정
  - 최종 설계 제출 마감 및 참가자 발표 (Code, Recording Video & PPT): 2024.5.31 (금)
- **최종 심사** : 24.6.3(월)~24.6.6(목)
- **최우수팀 선정 및 시상식** : 24.6.7(금) (온라인(Zoom)으로 진행)

Registrations and Participants

- **112 responses from seven schools classified in three categories (Details in here)**
  - Accepted: **41팀 - 105명**

(서울대, 강원대, 중앙대, 숭실대)

#	이름(Name)	경진대회 참가팀 명(Team Name)
1	장지수	420
	최진태	420
2	김산	301동 골목대장
	황현태	301동 골목대장
	이재범	301동 골목대장
4	천동현	57-51534
	전민서	57-51534
	장건희	57-51534
5	강태훈	강서영
	오재영	강서영
	손서형	강서영
…	…	…
41	박주은	zi존낸드
	최서영	zi존낸드

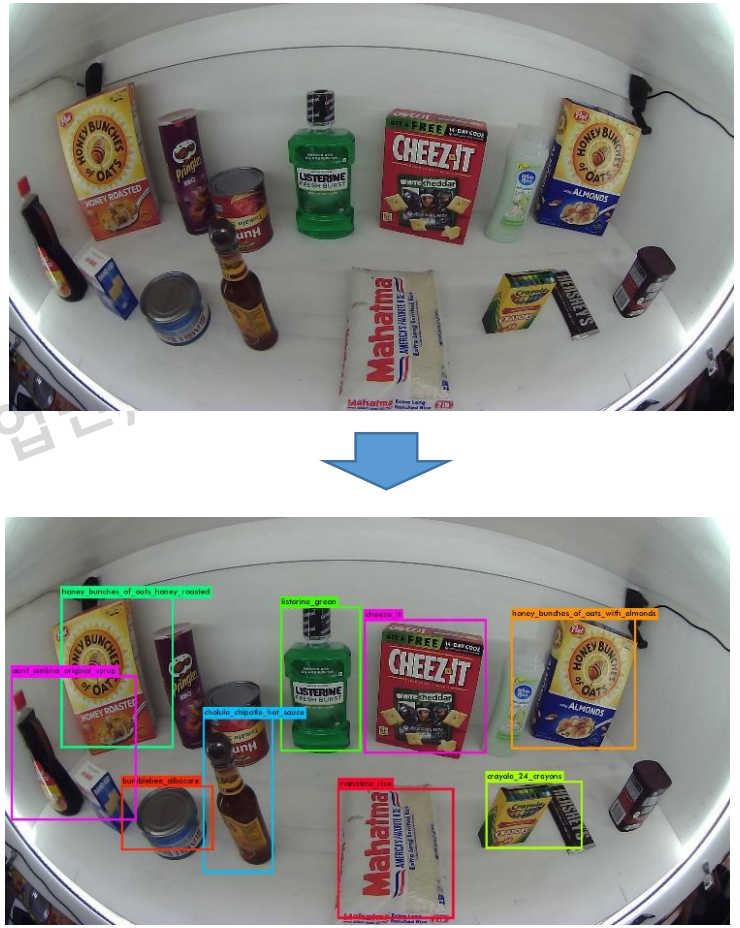
  - **Pending one-member teams**
    - They must form a team or join other two-member teams by 3/4 (Monday)
  - **Rejected: 7개대학 외 신청 3명(접수 제외)**

Outlines

- About AIX2024
- Quick Start
  - Background and AIX2024 model
  - Code structure: Compile and Run
  - Quantization
  - Hardware
- TODO
- Evaluation and Award

Image and Object Detection

- What is an image?
  - Example: An RGB image 1920×1080×3



- What is object detection?
  - Detect an object specified by
    - Object class (indicated by colors)
    - Location of an object in an image
      - Indicated by a bounding box

Test images (skeleton/bin/dataset)

- **Test images:** 232 images in two categories
  - Long: distances among objects are far, Close: items are close
- **Ground truth** (*.txt files) Each line represents a labeled object
  - [class_index x_pos y_pos width height]
- **yolohw.names**: names of all **60 classes** of products

long

close

```
12 0.19973958333333333 0.39166666666666666 0.0671875 0.2777777777777778
13 0.26484375 0.39351851851851855 0.1265625 0.2796296296296296
11 0.3419270833333333 0.38472222222222224 0.06302083333333333 0.287962962962963
6 0.4427083333333333 0.33055555555555555 0.178125 0.4425925925925926
0 0.5614583333333333 0.3842592592592593 0.090625 0.3388888888888889
9 0.6377604166666666 0.3773148148148148 0.09010416666666667 0.325
7 0.7296875 0.3699074074074074 0.14895833333333333 0.34351851851851856
1 0.19817708333333334 0.649537037037037 0.08072916666666667 0.18055555555555555
2 0.2671875 0.7041666666666667 0.08645833333333333 0.1824074074074074
3 0.30859375 0.6421296296296296 0.0859375 0.3212962962962963
4 0.3890625 0.7495370370370371 0.090625 0.1787037037037037
14 0.45598958333333334 0.7847222222222222 0.0734375 0.23425925925925928
10 0.621875 0.7532407407407408 0.25625 0.23425925925925928
5 0.765625 0.6810185185185186 0.11041666666666666 0.21574074074074076
8 0.825 0.6587962962962963 0.09270833333333334 0.23240740740740742

```

CAPP_testset_close_10000.txt

```
aunt_jemima_original_syrup
band_aid_clear_strips
bumblebee_albacore
cholula_chipotle_hot_sauce
crayola_24_crayons
hersheys_cocoa
honey_bunches_of_oats_honey_roasted
honey_bunches_of_oats_with_almonds
hunts_sauce
listerine_green
mahatma_rice
white_rain_body_wash
pringles_bbq
cheeze_it
hersheys_bar
redbull
```

yolohw.names

Deep Neural Network: AIX 2024

- AIX2024: A 22-layer DNN model for Object Detection
- Input: an RGB image (256×256×3)
- 22 Layers
  - Convolution: 11 layers
  - Max-pooling: 6 layers
  - Route: 2 layers
  - Upsample: one layer
  - YOLO: two layers

Input image

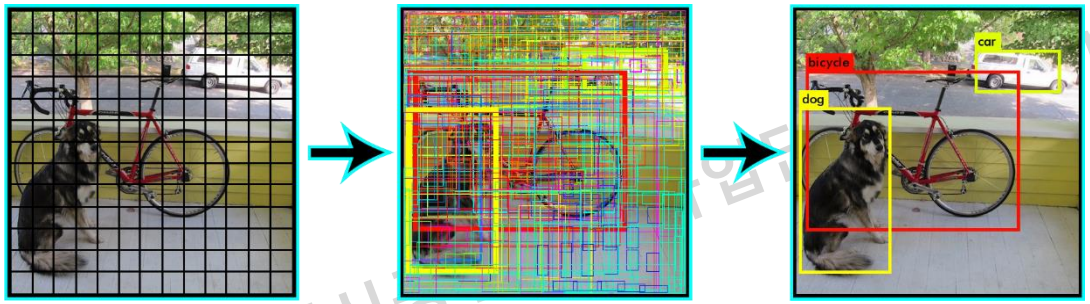
Output Tensors

```
layer     filters    size              input                output
    0 conv     16  3 x 3 / 1   256 x 256 x   3   ->   256 x 256 x  16 0.057 BF
    1 max          2 x 2 / 2   256 x 256 x  16   ->   128 x 128 x  16
    2 conv     32  3 x 3 / 1   128 x 128 x  16   ->   128 x 128 x  32 0.151 BF
    3 max          2 x 2 / 2   128 x 128 x  32   ->    64 x  64 x  32
    4 conv     64  3 x 3 / 1    64 x  64 x  32   ->    64 x  64 x  64 0.151 BF
    5 max          2 x 2 / 2    64 x  64 x  64   ->    32 x  32 x  64
    6 conv    128  3 x 3 / 1    32 x  32 x  64   ->    32 x  32 x 128 0.151 BF
    7 max          2 x 2 / 2    32 x  32 x 128   ->    16 x  16 x 128
    8 conv    256  3 x 3 / 1    16 x  16 x 128   ->    16 x  16 x 256 0.151 BF
    9 max          2 x 2 / 2    16 x  16 x 256   ->     8 x   8 x 256
   10 conv    512  3 x 3 / 1     8 x   8 x 256   ->     8 x   8 x 512 0.151 BF
   11 max          2 x 2 / 1     8 x   8 x 512   ->     8 x   8 x 512
   12 conv    256  1 x 1 / 1     8 x   8 x 512   ->     8 x   8 x 256 0.017 BF
   13 conv    512  3 x 3 / 1     8 x   8 x 256   ->     8 x   8 x 512 0.151 BF
   14 conv    195  1 x 1 / 1     8 x   8 x 512   ->     8 x   8 x 195 0.013 BF
   15 yolo
   16 route  12
   17 conv    128  1 x 1 / 1     8 x   8 x 256   ->     8 x   8 x 128 0.004 BF
   18 upsample            2x    8 x   8 x 128   ->    16 x  16 x 128
   19 route  18 8
   20 conv    195  1 x 1 / 1    16 x  16 x 384   ->    16 x  16 x 195 0.038 BF
   21 yolo
Total BFLOPS 1.035
```

- The output tensors: **8×8×195 (Layer 14)** and **16×16×195 (Layer 20)**

Deep Neural Network: Object Detection

- YOLO Network (https://pjreddie.com/darknet/yolov2/)
  - Divide an object into a grid
  - Detect if a small grid has an object



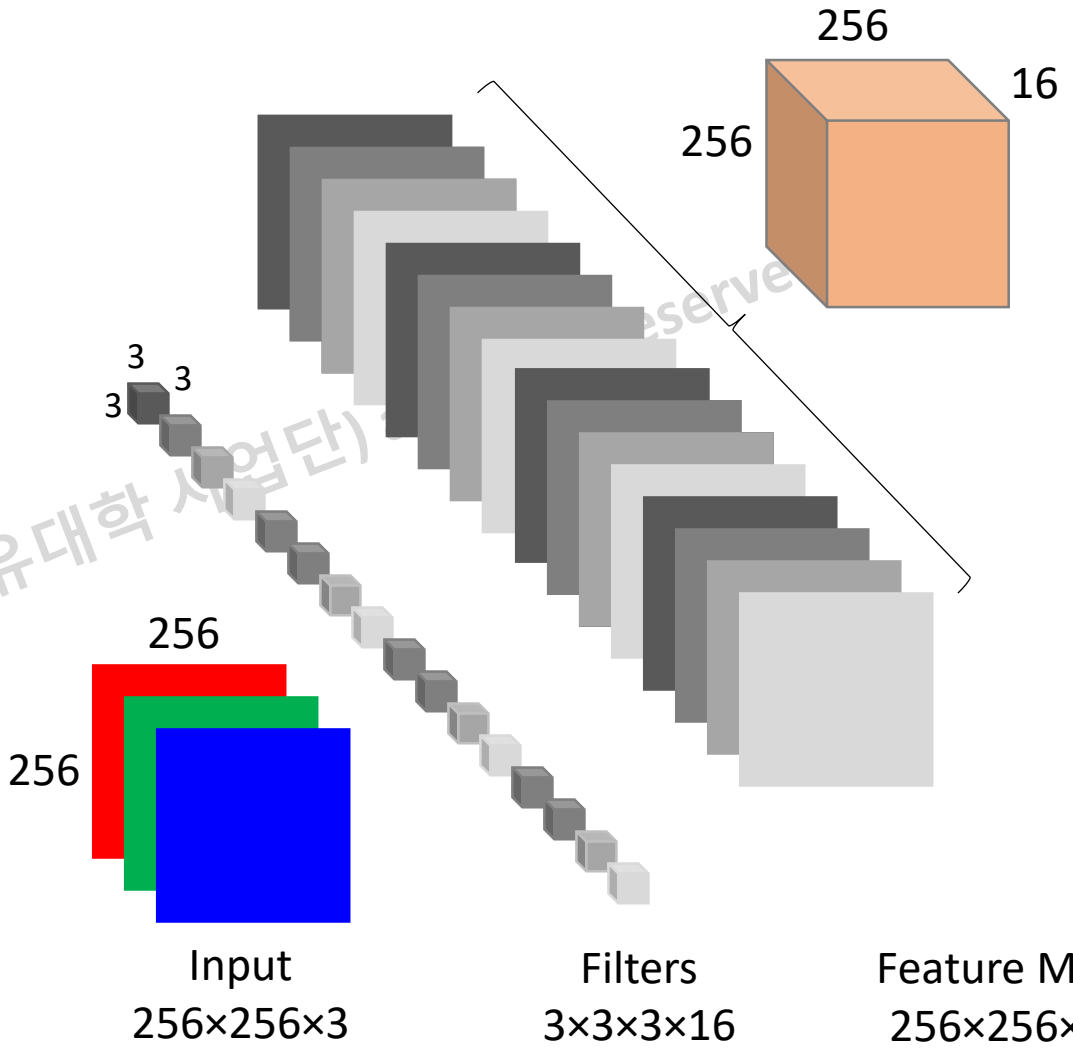
- AIX2024: A full-HD image (1920×1080×3) is resized to an RGB image (256×256×3)
  - Output **8×8×195** and **16×16×195 => Grid size: 8x8 and 16x16**
  - How about 195?
    - 195 = [4 (location) + 60 (class prob.) + 1 (obj. prob.)] * 3 (directions)

YOLOv2, https://arxiv.org/pdf/1612.08242.pdf

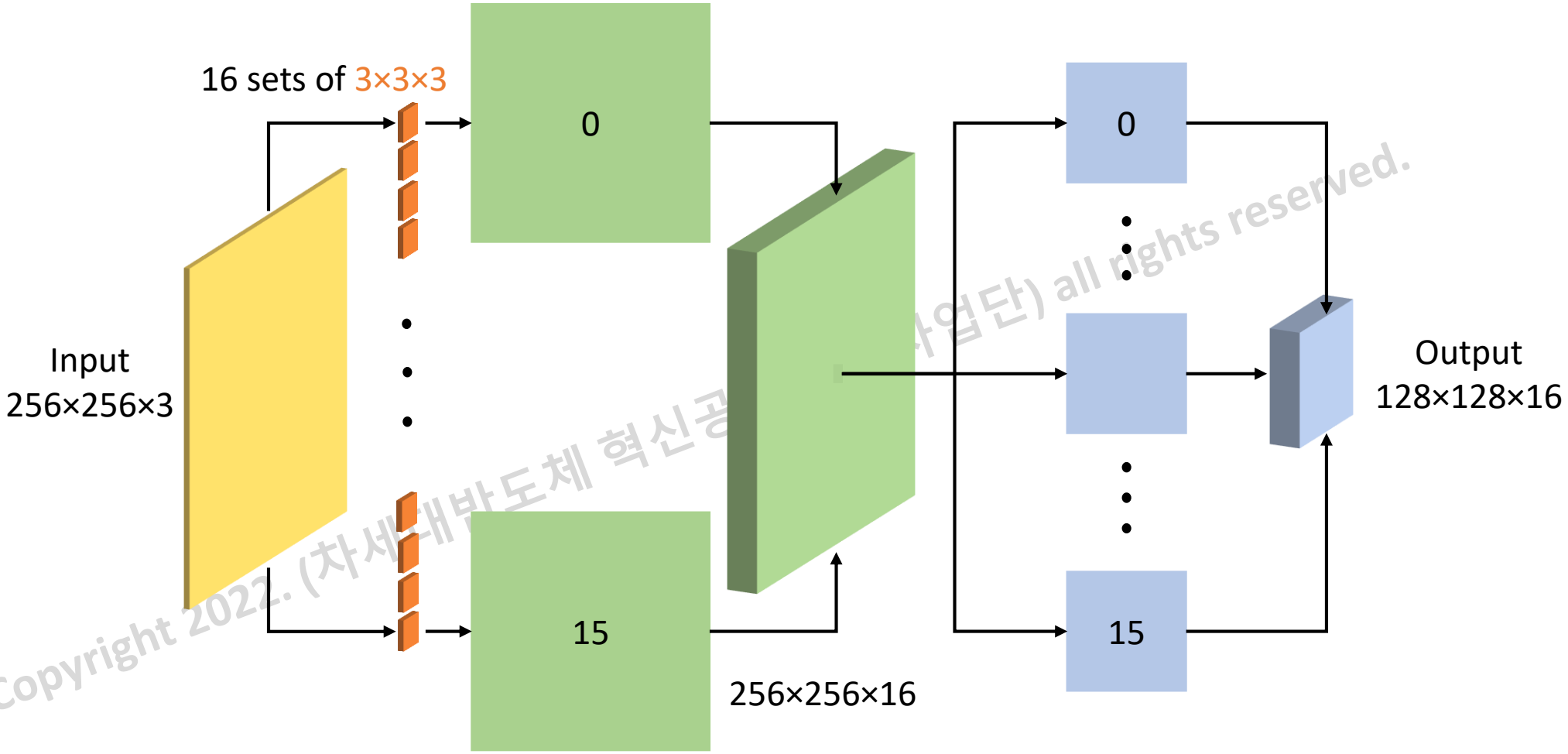
Wait. What is Convolution?

- What is a convolution layer?
  - Convolute an input image by filters to output feature maps
- Example: Layer 1

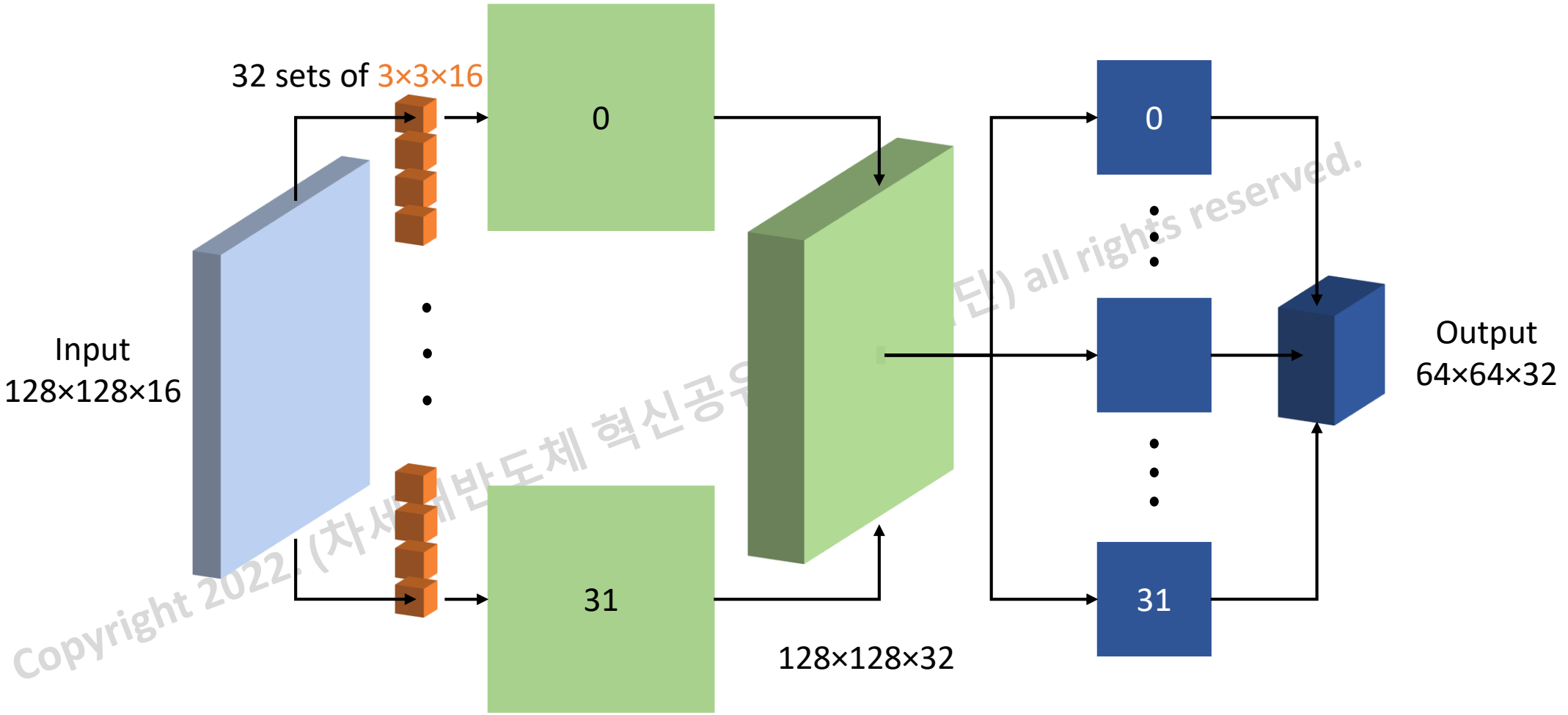
```
1: for och = 1→16
2:    for row = 1→256
3:       for col = 1→256
4:          ofm(row,col,och) =
                convolve(ifm(row,col,:),
                filter(:, :, :, och))
5:       end for
6:    end for
7: end for
```



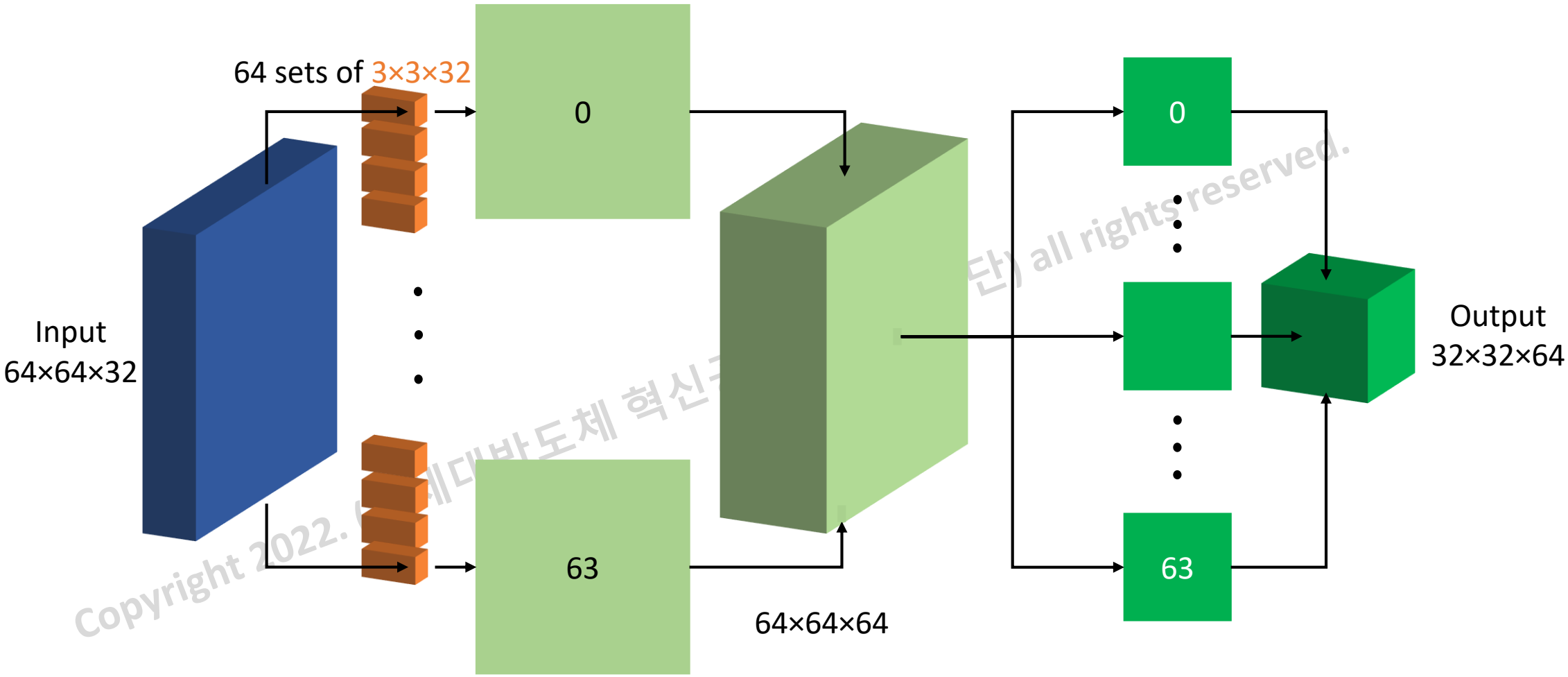
Convolution + max-pooling: Layers 0, 1



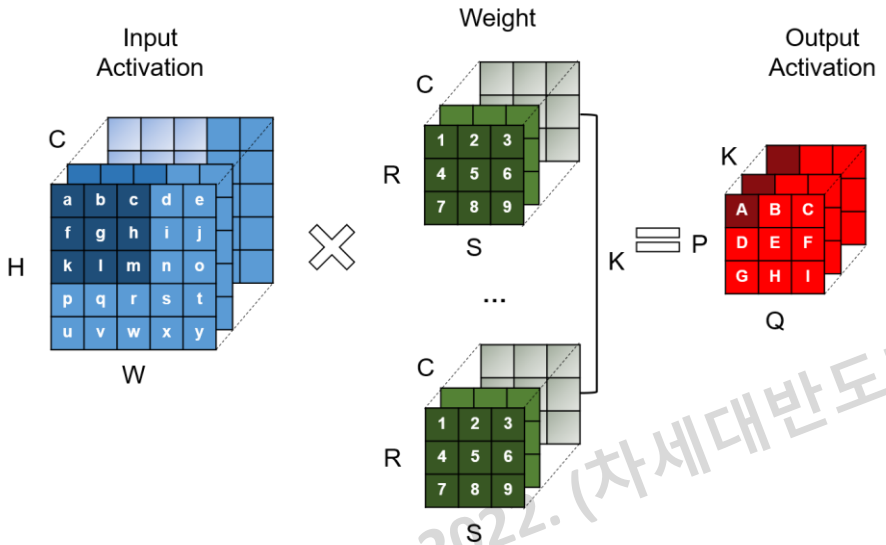
Convolution + max-pooling: Layers 2, 3



Convolution + max-pooling: Layers 4, 5



3D Convolution

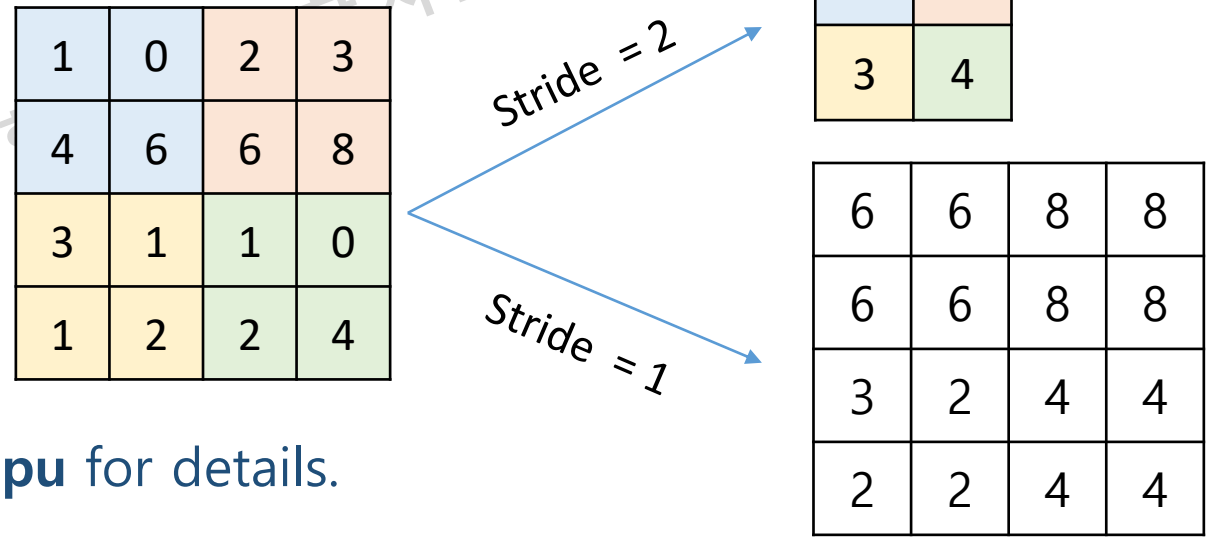



```
1:  for (k=0; k<K; k++) {
2:      for (p=0; p<P; p++) {
3:        for (q=0; q<Q; q++) {
4:          ofm[k][p][q]= 0;
5:          for (r=0; r<R; r++) {
6:            for (s=0; s<S; s++) {
7:              for (c=0; c<C; c++) {
8:                h = p * stride - pad + r;
9:                w = q * stride - pad + s;
10:                ofm[k][p][q] += ifm[c][h][w]*filter[k][c][r][s];
11:             }
12:           }
13:         }
14:         ofm[k][p][q]= Activation(ofm[k][p][q] + bias[k]);
15:       }
16:     }
17:   }
```

Activation:
+ Linear: f(x) = x
+ ReLU: f(x) = max(x,0)

Max Pooling (Stride = 2, 1)

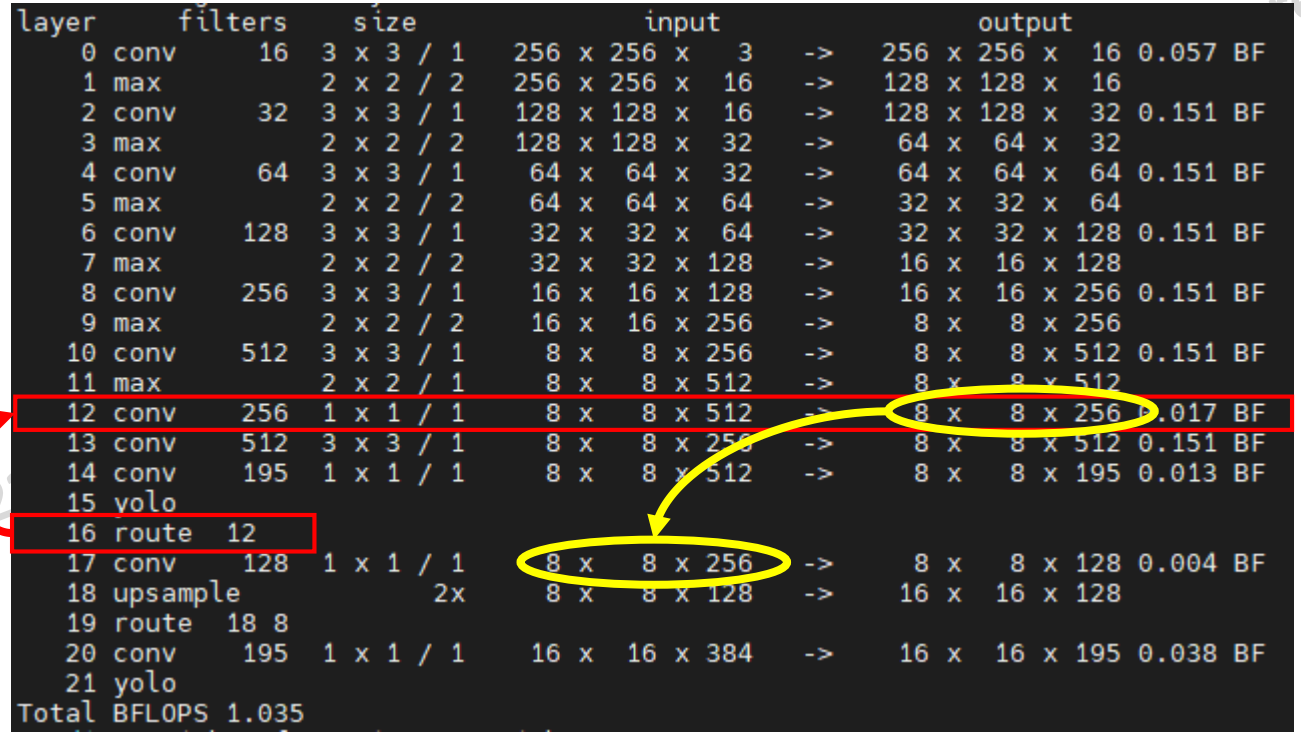
- Kernel size 2x2: Find the maximum value of all elements in a kernel
  - Reduce the spatial dimensions of tensors
  - Stride = 2: Layers 1, 3, 5, 7, 9.
    - Reduce width and height by 2x
  - Stride = 1: Layer 11



- Check **forward_maxpool_layer_cpu** for details.

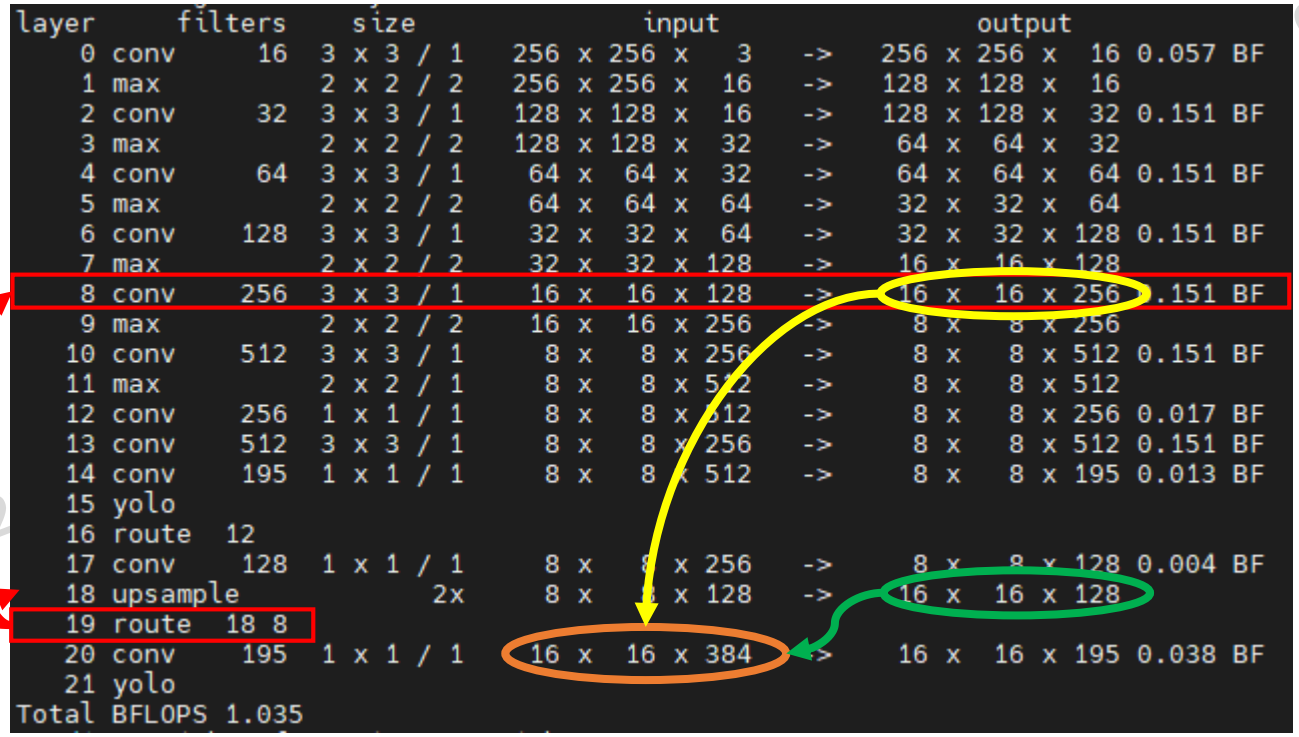
Route Layer (Layer 16)

- [route]: route 12 (Check **forward_route_layer_cpu** for details (yolov2_forward_network.c))
  - A route or routing layer gets an output from one layer
  - Layer 16 gets the output (8×8×256) from Layer 12 which will become an input for Layer 17



Route Layer (Layer 19)

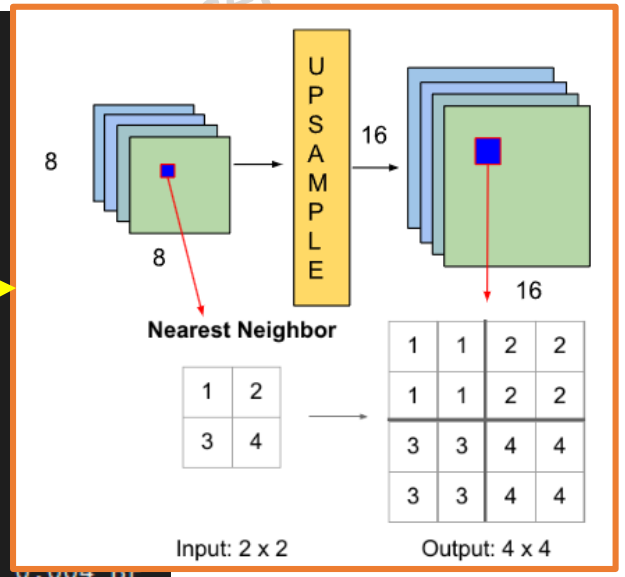
- [route]: route 18 8 (Check **forward_route_layer_cpu** for details)
  - Layer 19 **concatenates** the outputs from Layer 18 (16×16×256) and Layer 8 (16×16×256) to form an input tensor (16×16×(128+256)) for Layer 20



Up-Sample Layer

- [upsample]: (Check **forward_upsample_layer_cpu/upsample_cpu** for details)
  - An up-sampling layer enlarges a convolutional output to generate an upsampled image.
  - For example, for given feature maps 8x8x128, it generates the feature maps 16x16x128.

```
layer     filters    size              input                output
    0 conv     16  3 x 3 / 1   256 x 256 x   3   ->   256 x 256 x  16
    1 max          2 x 2 / 2   256 x 256 x  16   ->   128 x 128 x  16
    2 conv     32  3 x 3 / 1   128 x 128 x  16   ->   128 x 128 x  32
    3 max          2 x 2 / 2   128 x 128 x  32   ->    64 x  64 x  32
    4 conv     64  3 x 3 / 1    64 x  64 x  32   ->    64 x  64 x  64
    5 max          2 x 2 / 2    64 x  64 x  64   ->    32 x  32 x  64
    6 conv    128  3 x 3 / 1    32 x  32 x  64   ->    32 x  32 x 128
    7 max          2 x 2 / 2    32 x  32 x 128   ->    16 x  16 x 128
    8 conv    256  3 x 3 / 1    16 x  16 x 128   ->    16 x  16 x 256
    9 max          2 x 2 / 2    16 x  16 x 256   ->     8 x   8 x 256
   10 conv    512  3 x 3 / 1     8 x   8 x 256   ->     8 x   8 x 512
   11 max          2 x 2 / 1     8 x   8 x 512   ->     8 x   8 x 512
   12 conv    256  1 x 1 / 1     8 x   8 x 512   ->     8 x   8 x 256
   13 conv    512  3 x 3 / 1     8 x   8 x 256   ->     8 x   8 x 512
   14 conv    195  1 x 1 / 1     8 x   8 x 512   ->     8 x   8 x 195
   15 yolo
   16 route  12
   17 conv    128  1 x 1 / 1     8 x   8 x 256   ->     8 x   8 x 128
   18 upsample            2x    8 x   8 x 128   ->    16 x  16 x 128
   19 route  18 8
   20 conv    195  1 x 1 / 1    16 x  16 x 384   ->    16 x  16 x 195 0.038 BF
   21 yolo
Total BFLOPS 1.035
```



Outlines

- About AIX2024
- Quick Start
  - Background
  - Code structure: Compile and Run
  - Quantization
  - Hardware
- TODO
- Evaluation and Award

Software Development Kit Environment

- Software for model quantization
  - C/C++: OS: Windows (Visual Studio (2019[1])), Linux (gcc (7.5.0[2]) on Ubuntu 18.04)), Mac (gcc)
  - Python (3.7.9[3] or 2.7.17[4])
    - Automatically generate the directories for test images.
    - Optional: Analyze data for quantization.
  - *(1),(2),(3),(4)* ***the code is likely to work fine with other versions. We will give additional support if needed.***
- Hardware implementation
  - Xilinx Vivado 2021.1:
    - RTL simulation and verification, IP Packaging, Synthesis, and FPGA implementation.
    - Only support Windows and Linux versions. *For a Mac OS, need to install a virtual machine*

  ⇒ *You should utilize a desktop PC in the laboratories at your school*
- System Integration
  - Host code: (1) C/C++, or (2) Python
  - Xilinx Vitis 2021.1 (C/C++)
    - Make the firmware code

Software Code Structure: Windows/Unix

skeleton

- **3rdparty : Third party library (Don't touch)**
  - lib/ Library
  - include/ Header files
  - dll/ Dynamic Link Library
  - CLBlast/ A modern, lightweight, performant and tunable OpenCL BLAS library implements BLAS routines: *basic linear algebra subprograms* (BLAS) operating on vectors and matrices.
- **src : Source code**
  - main.c //Main file
  - yolov2_forward_network.c // Do inference for an **FP32** model
  - yolov2_forward_network_quantized.c // Do inference for an **int8 quantized** model
  - additionally.c // All functions and utilities
  - …
- **bin : Executable files and bash scripts**
  - dataset/ test images and labels, make_list_cur.py, show_images.py, size_search.py, target.txt
  - weights/ output files when saving the model's parameters layer by layer
  - *.weights, *.cfg Files to save the parameters, and the structure of the model AIX2023
  - … Scripts for Unix/Linux (*.sh), scripts for Windows (*.cmd), executable files
- **obj : Object files when compiled on Unix/Linux (Don't care)**
- **Makefile : Makefile**
- ***.sln, *.cv* : Visual studio project**

Preparing dataset: Windows OS

- Generate the image file lists (target.txt)
- IMPORTANT NOTE**:
  - If your code folder is C:/skeleton, you don't need to do this step
- Otherwise
  - Open Command Prompt and run

    cd your_directory₩skeleton₩bin₩dataset

    C:₩Python27₩python.exe make_list_cur.py

README.md | Makefile | aix2024.cfg | target.txt

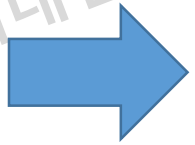
```
1  C:/skeleton/bin/dataset/CAPP_testset_close_10038.jpg
2  C:/skeleton/bin/dataset/CAPP_testset_long_10033.jpg
3  C:/skeleton/bin/dataset/CAPP_testset_long_10030.jpg
4  C:/skeleton/bin/dataset/CAPP_testset_close_10022.jpg
5  C:/skeleton/bin/dataset/CAPP_testset_long_10056.jpg
6  C:/skeleton/bin/dataset/CAPP_testset_close_10049.jpg
7  C:/skeleton/bin/dataset/CAPP_testset_long_10052.jpg
8  C:/skeleton/bin/dataset/CAPP_testset_long_10102.jpg
9  C:/skeleton/bin/dataset/CAPP_testset_long_10020.jpg
10 C:/skeleton/bin/dataset/CAPP_testset_close_10020.jpg
11 C:/skeleton/bin/dataset/CAPP_testset_long_10074.jpg
12 C:/skeleton/bin/dataset/CAPP_testset_close_10088.jpg
13 C:/skeleton/bin/dataset/CAPP_testset_close_10036.jpg
14 C:/skeleton/bin/dataset/CAPP_testset_long_10103.jpg
15 C:/skeleton/bin/dataset/CAPP_testset_long_10011.jpg
16 C:/skeleton/bin/dataset/CAPP_testset_close_10047.jpg
17 C:/skeleton/bin/dataset/CAPP_testset_long_10066.jpg
18 C:/skeleton/bin/dataset/CAPP_testset_close_10004.jpg
19 C:/skeleton/bin/dataset/CAPP_testset_long_10058.jpg
20 C:/skeleton/bin/dataset/CAPP_testset_close_10065.jpg
21 C:/skeleton/bin/dataset/CAPP_testset_close_10001.jpg
22 C:/skeleton/bin/dataset/CAPP_testset_close_10075.jpg
```

Preparing dataset: Mac/Linux OS

- Generate the image file lists
- Unix:
  - Go to the project skeleton

    cd bin/dataset

    python make_list_cur.py



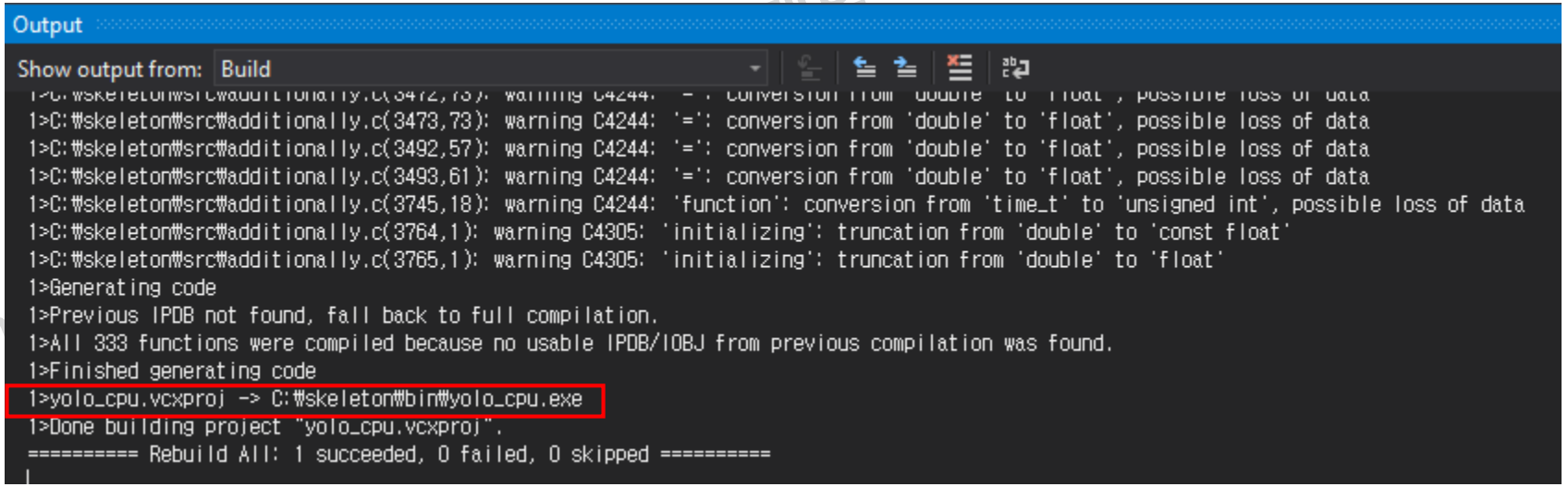
```
(base) truongnx@marlin:~/2024/AIX2024$ cd skeleton/bin/dataset/
(base) truongnx@marlin:~/2024/AIX2024/skeleton/bin/dataset$ python make_list_cur.py
CAPP_testset_long_10041
CAPP_testset_close_10086
CAPP_testset_close_10035
CAPP_testset_long_10085
CAPP_testset_close_10003
CAPP_testset_long_10023
CAPP_testset_long_10088
CAPP_testset_close_10120
CAPP_testset_long_10012
CAPP_testset_long_10066
```

README.md | Makefile | aix2024.cfg | target.txt | target.txt | target.txt

```
1  /home/truongnx/2024/AIX2024/skeleton/bin/dataset/CAPP_testset_long_10041.jpg
2  /home/truongnx/2024/AIX2024/skeleton/bin/dataset/CAPP_testset_close_10086.jpg
3  /home/truongnx/2024/AIX2024/skeleton/bin/dataset/CAPP_testset_close_10035.jpg
4  /home/truongnx/2024/AIX2024/skeleton/bin/dataset/CAPP_testset_long_10085.jpg
5  /home/truongnx/2024/AIX2024/skeleton/bin/dataset/CAPP_testset_close_10003.jpg
6  /home/truongnx/2024/AIX2024/skeleton/bin/dataset/CAPP_testset_long_10023.jpg
7  /home/truongnx/2024/AIX2024/skeleton/bin/dataset/CAPP_testset_long_10088.jpg
8  /home/truongnx/2024/AIX2024/skeleton/bin/dataset/CAPP_testset_close_10120.jpg
9  /home/truongnx/2024/AIX2024/skeleton/bin/dataset/CAPP_testset_long_10012.jpg
10 /home/truongnx/2024/AIX2024/skeleton/bin/dataset/CAPP_testset_long_10066.jpg
11 /home/truongnx/2024/AIX2024/skeleton/bin/dataset/CAPP_testset_long_10020.jpg
12 /home/truongnx/2024/AIX2024/skeleton/bin/dataset/CAPP_testset_long_10040.jpg
13 /home/truongnx/2024/AIX2024/skeleton/bin/dataset/CAPP_testset_long_10030.jpg
14 /home/truongnx/2024/AIX2024/skeleton/bin/dataset/CAPP_testset_long_10037.jpg
15 /home/truongnx/2024/AIX2024/skeleton/bin/dataset/CAPP_testset_close_10089.jpg
16 /home/truongnx/2024/AIX2024/skeleton/bin/dataset/CAPP_testset_close_10110.jpg
```

Compile and Run: Windows (1/5)

- NOTE**: Assume your code is located at C:₩skeleton
- Open the Visual Studio project and compile code the code
  - Click Build → Clean solution.
  - Now, click Build → Build Solution, and you can see the output window as follows.
  - The executable file **yolo_cpu.exe** is generated and stored in **C:₩skeleton₩bin.**



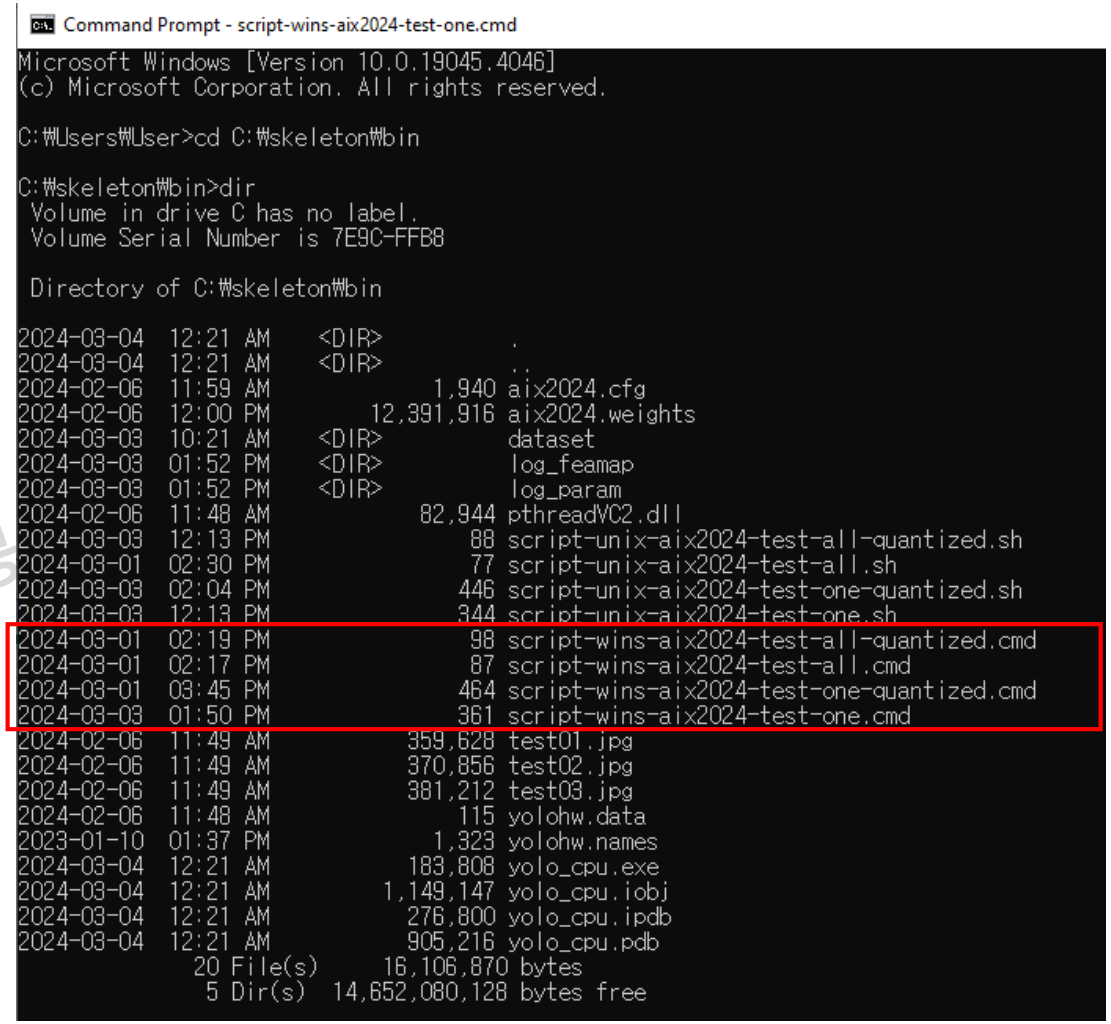
Compile and Run: Windows (2/5)

- Open your **Command Prompt and run** the scripts

  cd C:₩skeleton₩bin₩

  dir

- Scripts on Windows
  - 'script-wins-aix2024-*.cmd'



Compile and Run: Windows (3/5)

- Run a script to **test the AIX2024 model on one image**

script-wins-aix2024-test-one.cmd

yolo_cpu.exe detector test yolohw.names aix2024.cfg aix2024.weights -thresh 0.24 test01.jpg -out_filename test01-det

```
C:₩skeleton₩bin>script-wins-aix2024-test-one.cmd

C:₩skeleton₩bin>yolo_cpu.exe detector test yolohw.names aix2024.cfg aix2024.weights -thresh 0.24 test01.jpg -out_filename test01-det
layer     filters    size              input                output
    0 conv     16  3 x 3 / 1   256 x 256 x   3   ->   256 x 256 x  16 0.057 BF
    1 max          2 x 2 / 2   256 x 256 x  16   ->   128 x 128 x  16
    2 conv     32  3 x 3 / 1   128 x 128 x  16   ->   128 x 128 x  32 0.151 BF
    3 max          2 x 2 / 2   128 x 128 x  32   ->    64 x  64 x  32
    4 conv     64  3 x 3 / 1    64 x  64 x  32   ->    64 x  64 x  64 0.151 BF
    5 max          2 x 2 / 2    64 x  64 x  64   ->    32 x  32 x  64
    6 conv    128  3 x 3 / 1    32 x  32 x  64   ->    32 x  32 x 128 0.151 BF
    7 max          2 x 2 / 2    32 x  32 x 128   ->    16 x  16 x 128
    8 conv    256  3 x 3 / 1    16 x  16 x 128   ->    16 x  16 x 256 0.151 BF
    9 max          2 x 2 / 2    16 x  16 x 256   ->     8 x   8 x 256
   10 conv    512  3 x 3 / 1     8 x   8 x 256   ->     8 x   8 x 512 0.151 BF
   11 max          2 x 2 / 1     8 x   8 x 512   ->     8 x   8 x 512
   12 conv    256  1 x 1 / 1     8 x   8 x 512   ->     8 x   8 x 256 0.017 BF
   13 conv    512  3 x 3 / 1     8 x   8 x 256   ->     8 x   8 x 512 0.151 BF
   14 conv    195  1 x 1 / 1     8 x   8 x 512   ->     8 x   8 x 195 0.013 BF
   15 yolo
   16 route  12
   17 conv    128  1 x 1 / 1     8 x   8 x 256   ->     8 x   8 x 128 0.004 BF
   18 upsample            2x     8 x   8 x 128   ->    16 x  16 x 128
   19 route  18 8
   20 conv    195  1 x 1 / 1    16 x  16 x 384   ->    16 x  16 x 195 0.038 BF
   21 yolo
Total BFLOPS 1.035
Loading weights from aix2024.weights...
Done!

test01.jpg: Predicted in 0.025000 seconds.
mom_to_mom_sweet_potato_corn_apple: 67% (left_x:  240   top_y:  562   width:  155   height:  278)
pringles_bbq: 97%       (left_x:  397   top_y:  281   width:  146   height:  339)
redbull: 63%    (left_x:  552   top_y:  611   width:  141   height:  298)
dr_pepper: 93%  (left_x:  652   top_y:  406   width:  121   height:  219)
mahatma_rice: 71%       (left_x:  859   top_y:  645   width:  412   height:  383)
cheeze_it: 86%  (left_x:  969   top_y:  276   width:  289   height:  320)
cinnamon_toast_crunch: 92%      (left_x: 1178   top_y:  154   width:  461   height:  470)
crayola_24_crayons: 87% (left_x: 1225   top_y:  684   width:  171   height:  185)
white_rain_body_wash: 91%       (left_x: 1243   top_y:  263   width:  134   height:  356)
cinnamon_toast_crunch: 68%      (left_x: 1386   top_y:  296   width:  246   height:  272)
snickers: 69%   (left_x: 1395   top_y:  758   width:  134   height:  132)
hersheys_bar: 69%
crayola_24_crayons: 27% (left_x: 1548   top_y:  672   width:  153   height:  157)
arm_hammer_baking_soda: 51%     (left_x: 1556   top_y:  661   width:  157   height:  143)
Not compiled with OpenCV, saving to test01-det.png instead

C:₩skeleton₩bin>pause
Press any key to continue . . .

C:₩skeleton₩bin>
```

Compile and Run: Windows (4/5)

- Run a script to **test the AIX2024 model on one image**

script-wins-aix2024-test-one.cmd

yolo_cpu.exe detector test yolohw.names aix2024.cfg aix2024.weights -thresh 0.24 test01.jpg -out_filename test01-det

C:\skeleton\bin\test01.png

C:\skeleton\bin\test01-det.png

Compile and Run: Windows (5/5)

- Run a script to **test the AIX2024 model on all test images**

script-wins-aix2024-test-all.cmd

yolo_cpu.exe detector map yolohw.names aix2024.cfg aix2024.weights -thresh 0.24

```
C:¥skeleton¥bin>script-wins-aix2024-test-all.cmd

C:¥skeleton¥bin>yolo_cpu.exe detector map yolohw.names aix2024.cfg aix2024.weights -thresh 0.24
valid: Using default 'C:/skeleton/bin/dataset/target.txt'
names: Using default 'C:/skeleton/bin/yolohw.names'
layer     filters    size              input                output
    0 conv     16  3 x 3 / 1   256 x 256 x   3   ->   256 x 256 x  16 0.057 BF
    1 max          2 x 2 / 2   256 x 256 x  16   ->   128 x 128 x  16
    2 conv     32  3 x 3 / 1   128 x 128 x  16   ->   128 x 128 x  32 0.151 BF
    3 max          2 x 2 / 2   128 x 128 x  32   ->    64 x  64 x  32
    4 conv     64  3 x 3 / 1    64 x  64 x  32   ->    64 x  64 x  64 0.151 BF
    5 max          2 x 2 / 2    64 x  64 x  64   ->    32 x  32 x  64
    6 conv    128  3 x 3 / 1    32 x  32 x  64   ->    32 x  32 x 128 0.151 BF
    7 max          2 x 2 / 2    32 x  32 x 128   ->    16 x  16 x 128
    8 conv    256  3 x 3 / 1    16 x  16 x 128   ->    16 x  16 x 256 0.151 BF
    9 max          2 x 2 / 2    16 x  16 x 256   ->     8 x   8 x 256
   10 conv    512  3 x 3 / 1     8 x   8 x 256   ->     8 x   8 x 512 0.151 BF
   11 max          2 x 2 / 1     8 x   8 x 512   ->     8 x   8 x 512
   12 conv    256  1 x 1 / 1     8 x   8 x 512   ->     8 x   8 x 256 0.017 BF
   13 conv    512  3 x 3 / 1     8 x   8 x 256   ->     8 x   8 x 512 0.151 BF
   14 conv    195  1 x 1 / 1     8 x   8 x 512   ->     8 x   8 x 195 0.013 BF
   15 yolo
   16 route  12
   17 conv    128  1 x 1 / 1     8 x   8 x 256   ->     8 x   8 x 128 0.004 BF
   18 upsample            2x     8 x   8 x 128   ->    16 x  16 x 128
   19 route  18 8
   20 conv    195  1 x 1 / 1    16 x  16 x 384   ->    16 x  16 x 195 0.038 BF
   21 yolo
Total BFLOPS 1.035
Loading weights from aix2024.weights...
Done!
```

```
class_id = 43, name = campbells_chicken_noodle_soup,      ap = 99.47 %
class_id = 44, name = frappuccino_coffee,         ap = 91.53 %
class_id = 45, name = chewy_dips_chocolate_chip,          ap = 64.94 %
class_id = 46, name = chewy_dips_peanut_butter,           ap = 89.97 %
class_id = 47, name = nature_vally_fruit_and_nut,         ap = 92.30 %
class_id = 48, name = cheerios,           ap = 96.27 %
class_id = 49, name = lindt_excellence_cocoa_dark_chocolate,      ap = 81.82 %
class_id = 50, name = hersheys_symphony,          ap = 100.00 %
class_id = 51, name = campbells_chunky_classic_chicken_noodle,    ap = 94.65 %
class_id = 52, name = martinellis_apple_juice,    ap = 79.72 %
class_id = 53, name = dove_pink,          ap = 74.48 %
class_id = 54, name = dove_white,         ap = 88.93 %
class_id = 55, name = david_sunflower_seeds,      ap = 95.94 %
class_id = 56, name = monster_energy,     ap = 44.72 %
class_id = 57, name = act_ii_butter_lovers_popcorn,       ap = 86.10 %
class_id = 58, name = coca_cola_glass_bottle,     ap = 81.61 %
class_id = 59, name = twix,       ap = 85.90 %
 for thresh = 0.24, precision = 0.74, recall = 0.64, F1-score = 0.69
 for thresh = 0.24, TP = 1895, FP = 671, FN = 1064, average IoU = 55.01 %

 mean average precision (mAP) = 0.817559, or 81.76 %
Total Detection Time: 7.000000 Seconds
```

mAP = 81.76%

Compile and Run: Mac/Linux (1/3)

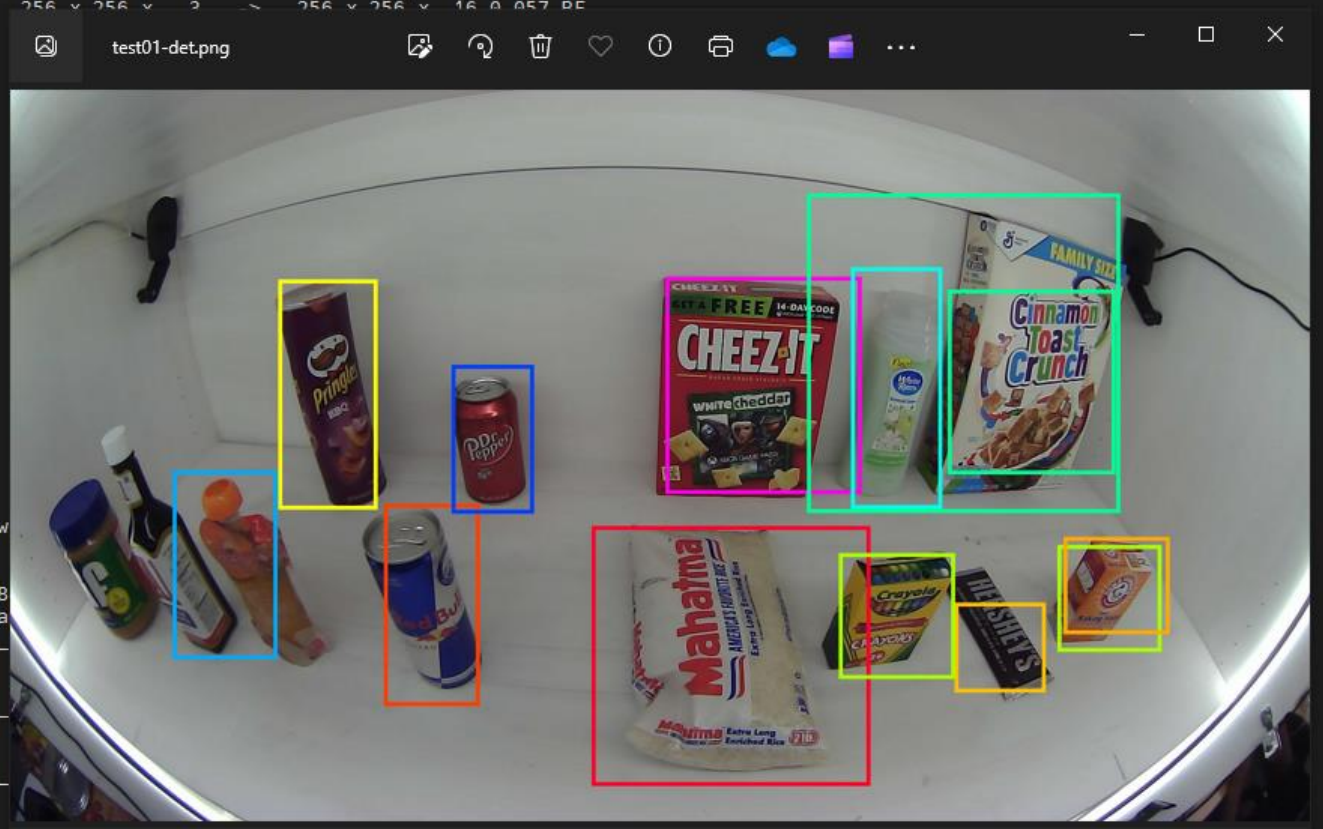
- Extract the file and type the following commands

```
cd skeleton/
make                              → Compile the code, the executable file is written to bin/
cd bin/dataset/
python make_list_cur.py           → Generate the directories for the test images
cd ..                             → Back to bin/
sh script-unix-aix2024-test-one.sh    → Test the model with one image.
sh script-unix-aix2024-test-all.sh    → Test the model with all test images.
```

Compile and Run: Mac/Linux (2/3)

```
(base) truongnx@marlin:~/2024/AIX2024/skeleton/bin$ sh script-unix-aix2024-test-one.sh
layer     filters    size              input                output
    0 conv     16  3 x 3 / 1   256 x 256 x   3   ->   256 x 256 x  16 0.057 BF
    1 max           2 x 2 / 2
    2 conv     32  3 x 3 / 1
    3 max           2 x 2 / 2
    4 conv     64  3 x 3 / 1
    5 max           2 x 2 / 2
    6 conv    128  3 x 3 / 1
    7 max           2 x 2 / 2
    8 conv    256  3 x 3 / 1
    9 max           2 x 2 / 2
   10 conv    512  3 x 3 / 1
   11 max           2 x 2 / 1
   12 conv    256  1 x 1 / 1
   13 conv    512  3 x 3 / 1
   14 conv    195  1 x 1 / 1
   15 yolo
   16 route  12
   17 conv    128  1 x 1 / 1
   18 upsample            2x
   19 route  18 8
   20 conv    195  1 x 1 / 1
   21 yolo
Total BFLOPS 1.035
Loading weights from aix2024.w
Done!

test01.jpg: Predicted in 0.118
mom_to_mom_sweet_potato_corn_a
pringles_bbq: 97%      (left_
redbull: 63%    (left_x:  552
dr_pepper: 93%  (left_x:  652
mahatma_rice: 71%      (left_
cheeze_it: 86%  (left_x:  969
cinnamon_toast_crunch: 92%
crayola_24_crayons: 87% (left_
white_rain_body_wash: 91%
cinnamon_toast_crunch: 68%
snickers: 69%   (left_x: 1395
hersheys_bar: 69%
crayola_24_crayons: 27% (left_x: 1548   top_y:  672   width:  155   height:  157)
arm_hammer_baking_soda: 51%    (left_x: 1556   top_y:  661   width:  157   height:  143)
Not compiled with OpenCV, saving to test01-det.png instead
(base) truongnx@marlin:~/2024/AIX2024/skeleton/bin$
```



Compile and Run: Mac/Linux (3/3)

- sh script-unix-aix2024-test-all.sh

```
(base) truongnx@marlin:~/2024/AIX2024/skeleton/bin$ sh script-unix-aix2024-test-all.sh
valid: Using default 'dataset/target.txt'
names: Using default 'yolohw.names'
layer     filters    size              input                output
    0 conv     16  3 x 3 / 1   256 x 256 x   3   ->   256 x 256 x  16 0.057 BF
    1 max          2 x 2 / 2   256 x 256 x  16   ->   128 x 128 x  16
    2 conv     32  3 x 3 / 1   128 x 128 x  16   ->   128 x 128 x  32 0.151 BF
    3 max          2 x 2 / 2   128 x 128 x  32   ->    64 x  64 x  32
    4 conv     64  3 x 3 / 1    64 x  64 x  32   ->    64 x  64 x  64 0.151 BF
    5 max          2 x 2 / 2    64 x  64 x  64   ->    32 x  32 x  64
    6 conv    128  3 x 3 / 1    32 x  32 x  64   ->    32 x  32 x 128 0.151 BF
    7 max          2 x 2 / 2    32 x  32 x 128   ->    16 x  16 x 128
    8 conv    256  3 x 3 / 1    16 x  16 x 128   ->    16 x  16 x 256 0.151 BF
    9 max          2 x 2 / 2    16 x  16 x 256   ->     8 x   8 x 256
   10 conv    512  3 x 3 / 1     8 x   8 x 256   ->     8 x   8 x 512
   11 max          2 x 2 / 1     8 x   8 x 512   ->     8 x   8 x 512
   12 conv    256  1 x 1 / 1     8 x   8 x 512   ->     8 x   8 x 256
   13 conv    512  3 x 3 / 1     8 x   8 x 256   ->     8 x   8 x 512
   14 conv    195  1 x 1 / 1     8 x   8 x 512   ->     8 x   8 x 195
   15 yolo
   16 route  12
   17 conv    128  1 x 1 / 1     8 x   8 x 256   ->     8 x   8 x 128
   18 upsample            2x    8 x   8 x 128   ->    16 x  16 x 128
   19 route  18 8
   20 conv    195  1 x 1 / 1    16 x  16 x 384   ->    16 x  16 x 195
   21 yolo
Total BFLOPS 1.035
Loading weights from aix2024.weights...
Done!
class_id = 49, name = lindt_excellence_cocoa_dark_chocolate,     ap = 81.82 %
class_id = 50, name = hersheys_symphony,         ap = 100.00 %
class_id = 51, name = campbells_chunky_classic_chicken_noodle,   ap = 94.65 %
class_id = 52, name = martinellis_apple_juice,   ap = 79.72 %
class_id = 53, name = dove_pink,         ap = 74.48 %
class_id = 54, name = dove_white,        ap = 88.93 %
class_id = 55, name = david_sunflower_seeds,     ap = 95.94 %
class_id = 56, name = monster_energy,    ap = 44.72 %
class_id = 57, name = act_ii_butter_lovers_popcorn,      ap = 86.10 %
class_id = 58, name = coca_cola_glass_bottle,    ap = 81.61 %
class_id = 59, name = twix,      ap = 85.90 %
 for thresh = 0.24, precision = 0.80, recall = 0.70, F1-score = 0.74
 for thresh = 0.24, TP = 2058, FP = 508, FN = 901, average IoU = 59.16 %

 mean average precision (mAP) = 0.817559, or 81.76 %
Total Detection Time: 40.000000 Seconds
```

mAP = 81.76%

Outlines

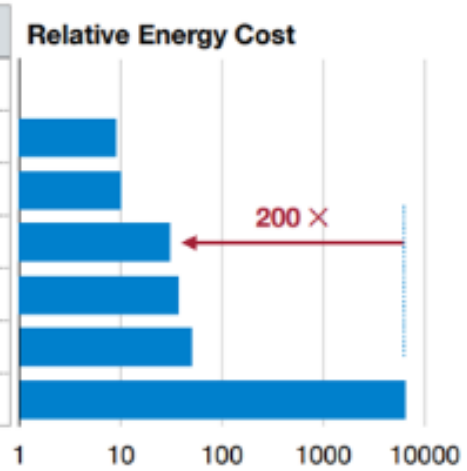
- About AIX2024
- Quick Start
  - Background
  - Code structure: Compile and Run
  - Quantization
  - Hardware
- TODO
- Evaluation and Award

Motivation for Quantization

- Two motivations are
  - Reduce storage and data transfers → Energy reduction
  - Reduce circuit complexity with low-bit-width data computing

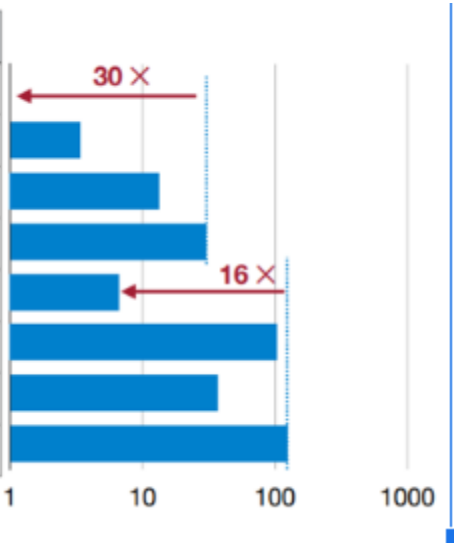
Operation	Energy [pJ]
32 bit int ADD	0.1
32 bit float ADD	0.9
32 bit Register File	1
32 bit int MULT	3.1
32 bit float MULT	3.7
32 bit SRAM Cache	5
32 bit DRAM Memory	640

Rough Energy Cost For Various Operations in 45nm 0.9V



Operation	Energy [pJ]
8 bit int ADD	0.03
32 bit int ADD	0.1
16 bit float ADD	0.4
32 bit float ADD	0.9
8 bit int MULT	0.2
32 bit int MULT	3.1
16 bit float MULT	1.1
32 bit float MULT	3.7

Rough Energy Cost For Various Operations in 45nm 0.9V



AIX2024: Quantization (Software)

- void **forward_convolutional_layer_q**(network net, layer l, network_state state)
  - **Do convolution with INT8 instead fp32**

Layer #0

Compute input_uint8
max(input*input_quant_multiplier, 255)

Convert input_uint8 and weights_int8 into matrices (im2col_cpu_int8)

Do matrix multiplication (**gemm_nn_int8_int32**)

Do bias addition (+ biases_quant)

Do activation (RELU: f(x) = max(0,x))

Do descaling or de-quantization (divide by (input_quant_multiplier* weights_quant_multiplier))

Layer #2

Compute input_uint8
max(input*input_quant_multiplier, 255)

Convert input_uint8 and weights_int8 into matrices (im2col_cpu_int8)

Do matrix multiplication (**gemm_nn_int8_int32**)

Do bias addition (+ biases_quant)

Do activation (RELU: f(x) = max(0,x))

Do descaling or de-quantization (divide by (input_quant_multiplier* weights_quant_multiplier))

Layer #4

Compute input_uint8
max(input*input_quant_multiplier, 255)

Convert input_uint8 and weights_int8 into matrices (im2col_cpu_int8)

Do matrix multiplication (**gemm_nn_int8_int32**)

Do bias addition (+ biases_quant)

Do activation (RELU: f(x) = max(0,x))

Do descaling or de-quantization (divide by (input_quant_multiplier* weights_quant_multiplier))

...

AIX2024: Quantization (Software)

- void **forward_convolutional_layer_q**(network net, layer l, network_state state)
  - **Do convolution with INT8 instead fp32**

INT8/UINT8

Compute input_uint8 max(input*input_quant_multiplier, 255)

Layer #0

Convert input_uint8 and weights_int8 into matrices (im2col_cpu_int8)

Do matrix multiplication (**gemm_nn_int8_int32**)

Do bias addition (+ biases_quant)

Do activation (RELU: f(x) = max(0,x))

Do descaling or de-quantization (divide by (input_quant_multiplier* weights_quant_multiplier))

Compute input_uint8 (Layer#2) max(input*input_quant_multiplier, 255)

INT8/UINT8

Layer #2

Convert input_uint8 and weights_int8 into matrices (im2col_cpu_int8)

Do matrix multiplication (**gemm_nn_int8_int32**)

Do bias addition (+ biases_quant)

Do activation (RELU: f(x) = max(0,x))

Do descaling or de-quantization (divide by (input_quant_multiplier* weights_quant_multiplier))

Compute input_uint8 (Layer#4) max(input*input_quant_multiplier, 255)

INT8/UINT8

Layer #4

Convert input_uint8 and weights_int8 into matrices (im2col_cpu_int8)

Do matrix multiplication (**gemm_nn_int8_int32**)

Do bias addition (+ biases_quant)

Do activation (RELU: f(x) = max(0,x))

Do descaling or de-quantization (divide by (input_quant_multiplier* weights_quant_multiplier))

Compute input_uint8 (Next Layer) max(input*input_quant_multiplier, 255)

INT8/UINT8

…

Quantization Parameters

- void **do_quantization**(network net)
  - Weight quantization

    weight_quant_multiplier: 11 layers
  - Activation quantization

    Input_quant_multiplier: 11 layers

```
302     //{{{
303     float weight_quant_multiplier[TOTAL_CALIB_LAYER] = {
304       16,     //conv 0
305       64,     //conv 2
306       64,     //conv 4
307       64,     //conv 6
308       64,     //conv 8
309       64,     //conv 10
310       64,     //conv 12
311       64,     //conv 13
312       64,     //conv 14
313       64,     //conv 17
314       64};    //conv 20
315
316     float input_quant_multiplier[TOTAL_CALIB_LAYER] = {
317      128,     //conv 0
318       16,     //conv 2
319       16,     //conv 4
320       16,     //conv 6
321       16,     //conv 8
322       16,     //conv 10
323       16,     //conv 12
324       16,     //conv 13
325       16,     //conv 14
326       16,     //conv 17
327       16};    //conv 20
328
```

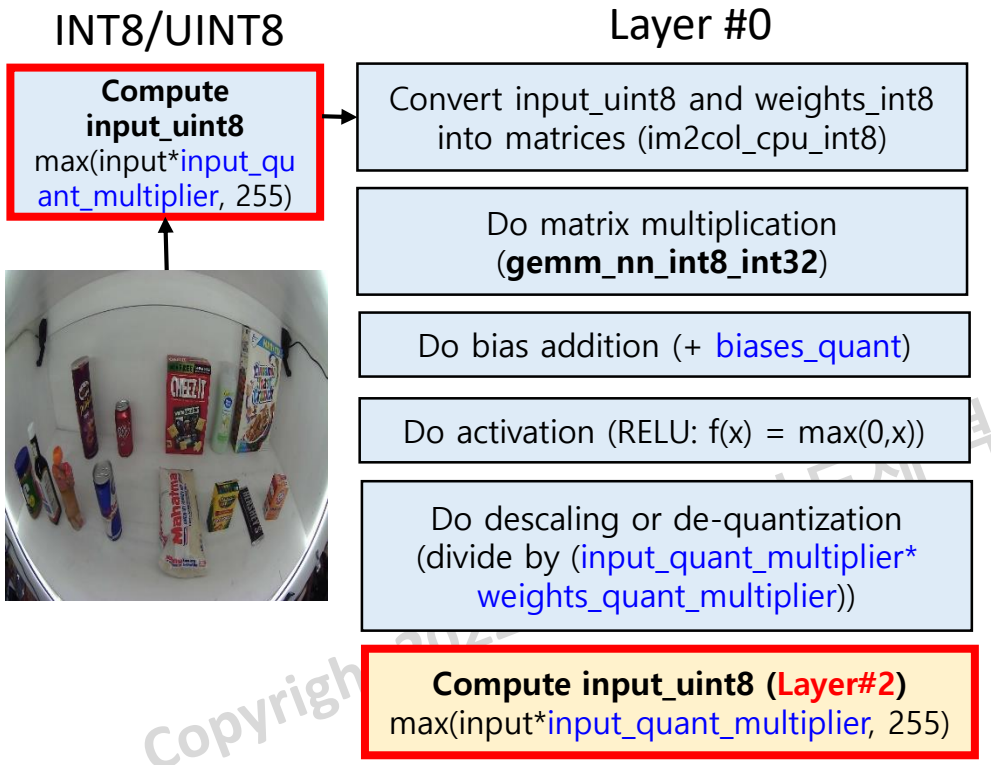
Weight Quantization

- Weight quantization (void **do_quantization**(network net))
  - weights_int8 ← max_abs(weights * weights_quant_multiplier, MAX_VAL_8)

```
339     if (l->type == CONVOLUTIONAL) { // Quantize conv layer only
340         size_t const filter_size = l->size*l->size*l->c;
341
342         int i, fil;
343
344         // ...
347             l->input_quant_multiplier   = (counter < TOTAL_CALIB_LAYER) ? input_quant_multiplier [counter] : 16;
348
349             // Weight
350             l->weights_quant_multiplier = (counter < TOTAL_CALIB_LAYER) ? weight_quant_multiplier[counter] : 16;
351
352             ++counter;
353         //}}}
354         // Weight Quantization
355         for (fil = 0; fil < l->n; ++fil) {          //
356             for (i = 0; i < filter_size; ++i) {
357                 float w = l->weights[fil*filter_size + i] * l->weights_quant_multiplier; // Scale
358                 l->weights_int8[fil*filter_size + i] = max_abs(w, MAX_VAL_8); // Clip
359             }
360         }
361
362         // Bias Quantization
363         float biases_multiplier = (l->weights_quant_multiplier * l->input_quant_multiplier);
364         for (fil = 0; fil < l->n; ++fil) {
365             float b = l->biases[fil] * biases_multiplier; // Scale
366             l->biases_quant[fil] = max_abs(b, MAX_VAL_16); // Clip
367         }
368
369         //printf(" CONV%d multipliers: input %g, weights %g, bias %g \n", j, l->input_quant_multiplier, l->weights_quant_multiplier, biases_multiplier);
370         printf(" CONV%d: \t%g \t%g \t%g \n", j, l->input_quant_multiplier, l->weights_quant_multiplier, biases_multiplier);
371     }
```

Activation Quantization: Input (Software)

- Activation quantization (void forward_convolutional_layer_q(network net, layer l, network_state s))
  - input_uint8 ← max_abs(input * input_quant_multiplier, MAX_VAL_8)




```
116  void forward_convolutional_layer_q(network net, layer l, network_state state)
117  {
118
119      int out_h = (l.h + 2 * l.pad - l.size) / l.stride + 1;     // output_height=i
120      int out_w = (l.w + 2 * l.pad - l.size) / l.stride + 1;     // output_width=in
121      int i, j;
122      int const out_size = out_h*out_w;
123
124      typedef int32_t conv_t;    // l.output
125      conv_t *output_q = calloc(l.outputs, sizeof(conv_t));
126
127      state.input_uint8 = (int8_t*)calloc(l.inputs, sizeof(uint8_t)); //state.inpu
128      int z;
129      for (z = 0; z < l.inputs; ++z) {
130          int16_t src = state.input[z] * l.input_quant_multiplier;
131          state.input_uint8[z] = max_abs(src, MAX_VAL_UINT_8);    //state.input_in
132      }
133
```

Activation Quantization: De-Scaling

- Activation quantization (void forward_convolutional_layer_q(network net, layer l, network_state s))
  - input_uint8 ← max_abs(input * input_quant_multiplier, MAX_VAL_8)

INT8/UINT8

Compute input_uint8
max(input*input_quant_multiplier, 255)

Layer #0

Convert input_uint8 and weights_int8 into matrices (im2col_cpu_int8)

Do matrix multiplication (**gemm_nn_int8_int32**)

Do bias addition (+ biases_quant)

Do activation (RELU: f(x) = max(0,x))

Do descaling or de-quantization (divide by (input_quant_multiplier* weights_quant_multiplier))

Compute input_uint8 (Layer#2)
max(input*input_quant_multiplier, 255)

INT8/UINT8

```
192     // Activation
193     if (l.activation == RELU) { ... }
198
199     // De-scaling or De-quantization
200     float ALPHA1 = 1 / (l.input_quant_multiplier * l.weights_quant_multiplier);
201     for (i = 0; i < l.outputs; ++i) { ... }
204
206     // Write data for the HW verification
207     //{{{
208     if (run_single_image_test) {
209         // Output Feature Map (OFM)
210         int z;
211         int next_input_quant_multiplier = 1;
212         for (z = state.index + 1; z < net.n; ++z) {
213             if (net.layers[z].type == CONVOLUTIONAL) {
214                 next_input_quant_multiplier = net.layers[z].input_quant_multiplier;
215                 break;
216             }
217         }
218         char file_output_femap[100];
219         snprintf(file_output_femap, sizeof(file_output_femap), "C:/skeleton/bin/log_feamap/CONV%02d_output.hex", state.index);
220         FILE* fp = fopen(file_output_femap, "w");
221
222         // Data Format: [Channel, Width, Height]
223         for (int idx = 0; idx < out_size; idx++) {  // OFM: Pixel index in ONE feature map
224             for (int chn = 0; chn < l.n; chn++) {   // OFM: Channel/index of an feature map
225                 int i = chn * out_size + idx;       // OFM: Pixel index
226                 uint8_t pixel = max_abs(l.output[i] * next_input_quant_multiplier, MAX_VAL_UINT_8);
227                 fprintf(fp, "%02x\n", pixel);
228             }
229         }
230         if (fp) fclose(fp);
231     }
232     //}}}
233     free(output_q);
234 }
```

Find input_quant_multiplier of the next layer

Test quantization: Windows (1/3)

- Run a script to **test the AIX2024 model on one image**

script-wins-aix2024-test-one-quantized.cmd

yolo_cpu.exe detector test yolohw.names aix2024.cfg aix2024.weights -thresh 0.24 test01.jpg -out_filename test01-det-quantized -quantized -save_params

```
 Quantization!

Multipler    Input    Weight    Bias
 CONV0:        128      16      2048
 CONV2:         16      64      1024
 CONV4:         16      64      1024
 CONV6:         16      64      1024
 CONV8:         16      64      1024
 CONV10:        16      64      1024
 CONV12:        16      64      1024
 CONV13:        16      64      1024
 CONV14:        16      64      1024
 CONV17:        16      64      1024
 CONV20:        16      64      1024

 Saving quantized model...

 Saving quantized weights, bias, and scale for CONV00
 Saving quantized weights, bias, and scale for CONV02
 Saving quantized weights, bias, and scale for CONV04
 Saving quantized weights, bias, and scale for CONV06
 Saving quantized weights, bias, and scale for CONV08
 Saving quantized weights, bias, and scale for CONV10
 Saving quantized weights, bias, and scale for CONV12
 Saving quantized weights, bias, and scale for CONV13
 Saving quantized weights, bias, and scale for CONV14
 Saving quantized weights, bias, and scale for CONV17
 Saving quantized weights, bias, and scale for CONV20
test01.jpg: Predicted in 0.033000 seconds.
mom_to_mom_sweet_potato_corn_apple: 50% (left_x:  202   top_y:  544   width:  193   height:  311)
pringles_bbq: 78%       (left_x:  374   top_y:  270   width:  177   height:  340)
dr_pepper: 89%  (left_x:  643   top_y:  438   width:  135   height:  136)
cheeze_it: 49%  (left_x:  944   top_y:  281   width:  215   height:  304)
white_rain_body_wash: 35%       (left_x: 1207   top_y:  275   width:  191   height:  326)
Not compiled with OpenCV, saving to test01-det-quantized.png instead

C:\skeleton\bin>pause
Press any key to continue . . .
```

Test quantization: Windows (2/3)

- Run a script to **test the AIX2024 model on one image**

script-wins-aix2024-test-one-quantized.cmd

yolo_cpu.exe detector test yolohw.names aix2024.cfg aix2024.weights -thresh 0.24 test01.jpg -out_filename test01-det-quantized -quantized -save_params

C:\skeleton\bin\test01-det.png

C:\skeleton\bin\test01-det-quantized.png

Test quantization: Windows (3/3)

- Run a script to **test the AIX2024 model on all test images**

script-wins-aix2024-test-all-quantized.cmd

yolo_cpu.exe detector map yolohw.names aix2024.cfg aix2024.weights -thresh 0.24 -quantized

Full-Precision

```
class_id = 43, name = campbells_chicken_noodle_soup,      ap = 99.47 %
class_id = 44, name = frappuccino_coffee,         ap = 91.53 %
class_id = 45, name = chewy_dips_chocolate_chip,          ap = 64.94
class_id = 46, name = chewy_dips_peanut_butter,           ap = 89.97
class_id = 47, name = nature_vally_fruit_and_nut,         ap = 92.30 %
class_id = 48, name = cheerios,           ap = 96.27 %
class_id = 49, name = lindt_excellence_cocoa_dark_chocolate,      ap = 81.82 %
class_id = 50, name = hersheys_symphony,          ap = 100.00 %
class_id = 51, name = campbells_chunky_classic_chicken_noodle,    ap = 94.65 %
class_id = 52, name = martinellis_apple_juice,    ap = 79.72 %
class_id = 53, name = dove_pink,          ap = 74.48 %
class_id = 54, name = dove_white,         ap = 88.93 %
class_id = 55, name = david_sunflower_seeds,      ap = 95.94 %
class_id = 56, name = monster_energy,     ap = 44.72 %
class_id = 57, name = act_ii_butter_lovers_popcorn,       ap = 86.10 %
class_id = 58, name = coca_cola_glass_bottle,     ap = 81.61 %
class_id = 59, name = twix,       ap = 85.90 %
 for thresh = 0.24, precision = 0.74, recall = 0.64, F1-score = 0.69
 for thresh = 0.24, TP = 1895, FP = 671, FN = 1064, average IoU = 55.01 %

 mean average precision (mAP) = 0.817559, or 81.76 %
Total Detection Time: 7.000000 Seconds
```

mAP = 81.76%

After Quantization (INT8)

```
class_id = 43, name = campbells_chicken_noodle_soup,      ap = 45.40 %
class_id = 44, name = frappuccino_coffee,         ap = 80.00 %
class_id = 45, name = chewy_dips_chocolate_chip,          ap = 35.23 %
class_id = 46, name = chewy_dips_peanut_butter,           ap = 72.98 %
class_id = 47, name = nature_vally_fruit_and_nut,         ap = 24.98 %
class_id = 48, name = cheerios,           ap = 87.83 %
class_id = 49, name = lindt_excellence_cocoa_dark_chocolate,      ap = 59.94 %
class_id = 50, name = hersheys_symphony,          ap = 98.99 %
class_id = 51, name = campbells_chunky_classic_chicken_noodle,    ap = 66.98 %
class_id = 52, name = martinellis_apple_juice,    ap = 32.10 %
class_id = 53, name = dove_pink,          ap = 22.91 %
class_id = 54, name = dove_white,         ap = 58.06 %
class_id = 55, name = david_sunflower_seeds,      ap = 71.00 %
class_id = 56, name = monster_energy,     ap = 10.08 %
class_id = 57, name = act_ii_butter_lovers_popcorn,       ap = 47.84 %
class_id = 58, name = coca_cola_glass_bottle,     ap = 65.94 %
class_id = 59, name = twix,       ap = 37.09 %
 for thresh = 0.24, precision = 0.64, recall = 0.23, F1-score = 0.34
 for thresh = 0.24, TP = 672, FP = 380, FN = 2287, average IoU = 44.83 %

 mean average precision (mAP) = 0.550382, or 55.04 %
Total Detection Time: 10.000000 Seconds
```

mAP = 55.04%

Test Quantization: Mac/Linux OS

- sh script-unix-aix2024-test-all-quantized.sh
- sh script-unix-aix2024-test-one-quantized.sh

```
(base) truongnx@marlin:~/2024/AIX2024/skeleton/bin$ sh script-unix-aix2024-test-all-quantized.sh
valid: Using default 'dataset/target.txt'
names: Using default 'yolohw.names'
layer     filters    size              input                output
    0 conv     16  3 x 3 / 1   256 x 256 x   3   ->   256 x 256 x  16 0.057 BF
    1 max          2 x 2 / 2   256 x 256 x  16   ->   128 x 128 x  16
    2 conv     32  3 x 3 / 1   128 x 128 x  16   ->   128 x 128 x  32 0.151 BF
    3 max          2 x 2 / 2   128 x 128 x  32   ->    64 x  64 x  32
    4 conv     64  3 x 3 / 1    64 x  64 x  32   ->    64 x  64 x  64 0.151 BF
    5 max          2 x 2 / 2    64 x  64 x  64   ->    32 x  32 x  64
    6 conv    128  3 x 3 / 1    32 x  32 x  64   ->    32 x  32 x 128 0.151 BF
    7 max          2 x 2 / 2    32 x  32 x 128   ->    16 x  16 x 128
    8 conv    256  3 x 3 / 1    16 x  16 x 128   ->    16 x  16 x 256 0.151 BF
    9 max          2 x 2 / 2    16 x  16 x 256   ->     8 x   8 x 256
   10 conv    512  3 x 3 / 1     8 x   8 x 256   ->     8 x   8 x 512 0.151 BF
   11 max          2 x 2 / 1     8 x   8 x 512   ->     8 x   8 x 512
   12 conv    256  1 x 1 / 1     8 x   8 x 512   ->     8 x   8 x 256 0.017 BF
   13 conv    512  3 x 3 / 1     8 x   8 x 256   ->     8 x   8 x 512 0.151 BF
   14 conv    195  1 x 1 / 1     8 x   8 x 512   ->     8 x   8 x 195 0.013 BF
   15 yolo
   16 route  12
   17 conv    128  1 x 1 / 1     8 x   8 x 256   ->     8 x   8 x 128 0.004 BF
   18 upsample           2x     8 x   8 x 128   ->    16 x  16 x 128
   19 route  18 8
   20 conv    195  1 x 1 / 1    16 x  16 x 384   ->    16 x  16 x 195 0.038 BF
   21 yolo
Total BFLOPS 1.035
Loading weights from aix2024.weights...
Done!

Multipler    Input    Weight    Bias
 CONV0:        128      16      2048
 CONV2:         16      64      1024
 CONV4:         16      64      1024
 CONV6:         16      64      1024
 CONV8:         16      64      1024
 CONV10:        16      64      1024
 CONV12:        16      64      1024
 CONV13:        16      64      1024
 CONV14:        16      64      1024
 CONV17:        16      64      1024
 CONV20:        16      64      1024
```

```
class_id = 45, name = chewy_dips_chocolate_chip,          ap = 35.23 %
class_id = 46, name = chewy_dips_peanut_butter,           ap = 72.98 %
class_id = 47, name = nature_vally_fruit_and_nut,         ap = 24.98 %
class_id = 48, name = cheerios,           ap = 87.83 %
class_id = 49, name = lindt_excellence_cocoa_dark_chocolate,      ap = 59.94 %
class_id = 50, name = hersheys_symphony,          ap = 98.99 %
class_id = 51, name = campbells_chunky_classic_chicken_noodle,    ap = 66.98 %
class_id = 52, name = martinellis_apple_juice,    ap = 32.10 %
class_id = 53, name = dove_pink,          ap = 22.91 %
class_id = 54, name = dove_white,         ap = 58.06 %
class_id = 55, name = david_sunflower_seeds,      ap = 71.00 %
class_id = 56, name = monster_energy,     ap = 10.08 %
class_id = 57, name = act_ii_butter_lovers_popcorn,       ap = 47.84 %
class_id = 58, name = coca_cola_glass_bottle,     ap = 65.94 %
class_id = 59, name = twix,       ap = 37.09 %
 for thresh = 0.24, precision = 0.72, recall = 0.26, F1-score = 0.38
 for thresh = 0.24, TP = 755, FP = 297, FN = 2204, average IoU = 49.61 %

 mean average precision (mAP) = 0.550382, or 55.04 %
Total Detection Time: 37.000000 Seconds
(base) truongnx@marlin:~/2024/AIX2024/skeleton/bin$
```

mAP = 55.04%

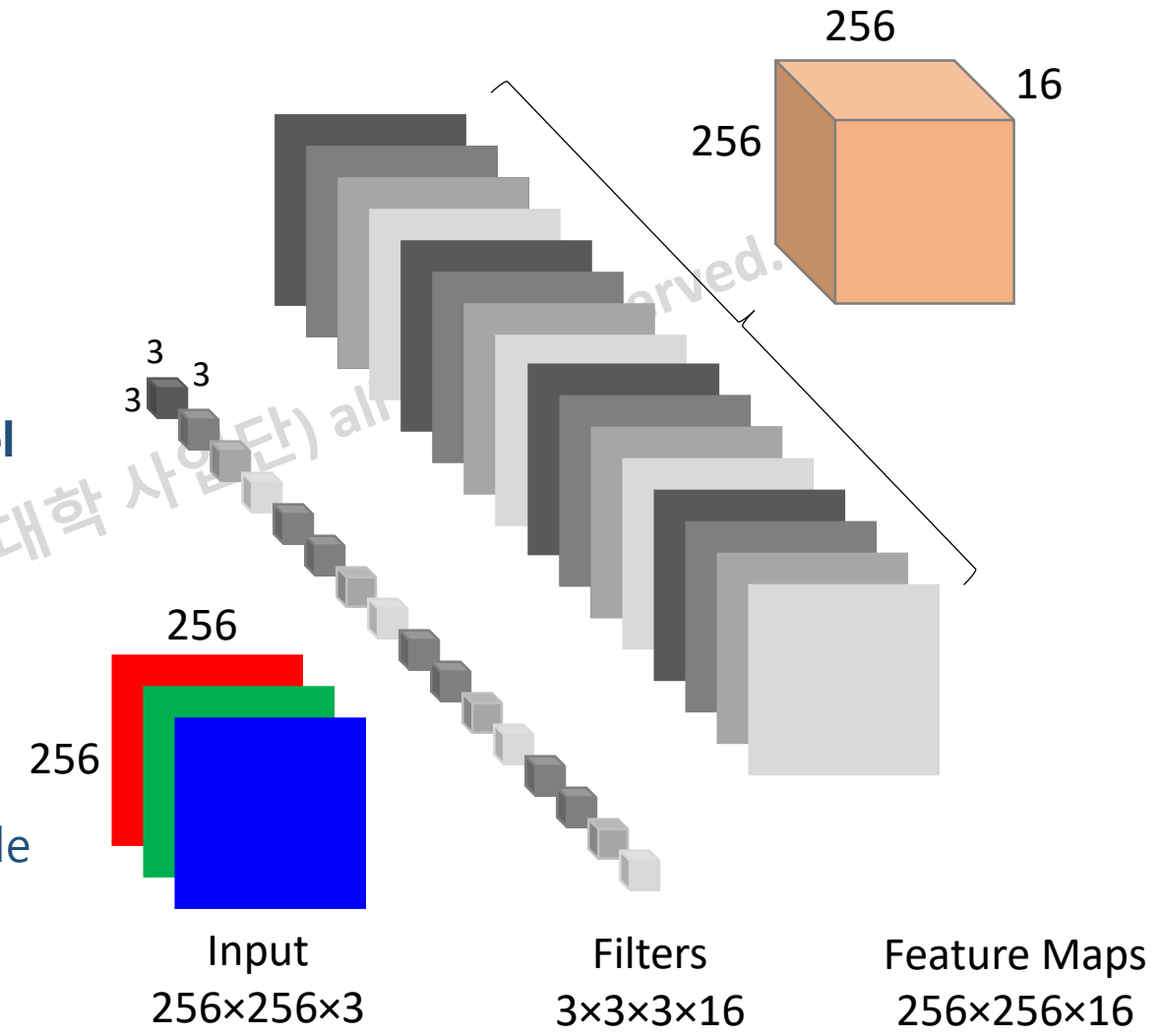
Outlines

- About AIX2024
- Quick Start
  - Background
  - Code structure: Compile and Run
  - Quantization
  - Hardware Implementation
- TODO
- Evaluation and Award

DNN Hardware Accelerator

- How to accelerate a CNN layer?
  - To generate **one output pixel**, **compute multiple multiplications and additions in parallel**
  - **Compute multiple output pixels in parallel**
- For example: Layer 1
  - Compute 1x1x16 output pixels
  - Use 9(=3x3) multipliers for one output

  ⇒ Compute up to 144 multiplications per cycle



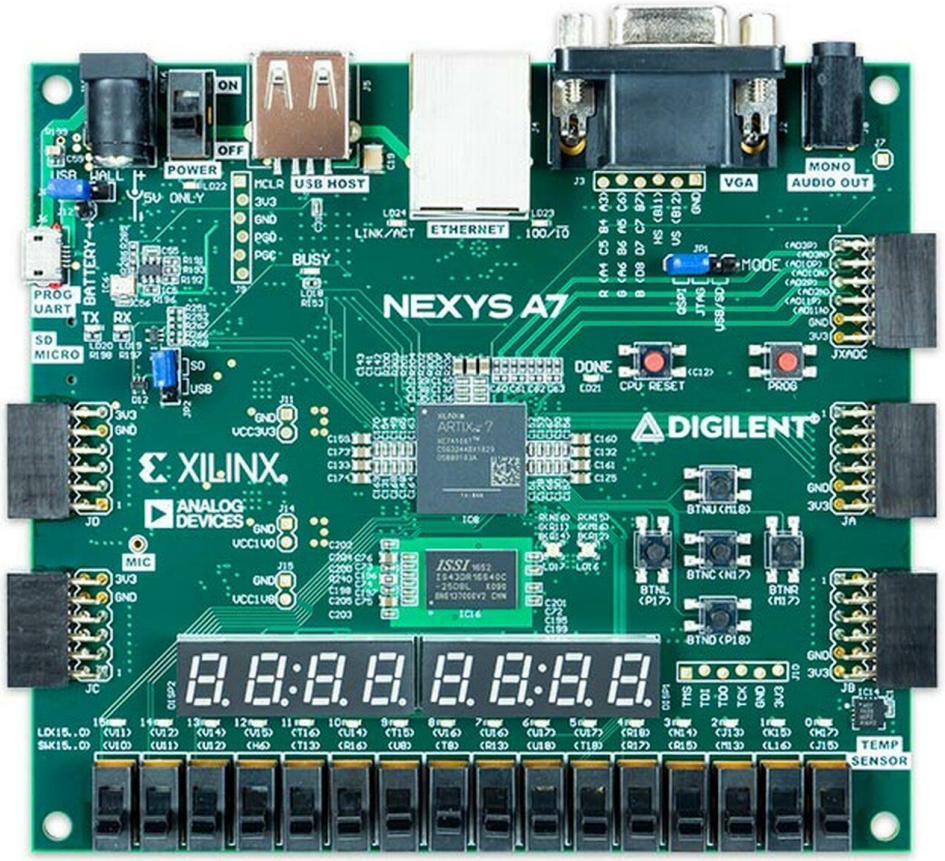
Latency: The number of operations

- Assumptions: 144 MACs => one MAC = mult + add => 288 operations
- Frequency: 100 MHz => Maximum operations: 288 * (100*10^6) = 28.8 GOPs/second
- Maximum frame rate: **27.82 (=28.8/1.035) frame per second**

```
layer     filters    size              input                output
    0 conv     16  3 x 3 / 1   256 x 256 x   3   ->   256 x 256 x  16 0.057 BF
    1 max          2 x 2 / 2   256 x 256 x  16   ->   128 x 128 x  16
    2 conv     32  3 x 3 / 1   128 x 128 x  16   ->   128 x 128 x  32 0.151 BF
    3 max          2 x 2 / 2   128 x 128 x  32   ->    64 x  64 x  32
    4 conv     64  3 x 3 / 1    64 x  64 x  32   ->    64 x  64 x  64 0.151 BF
    5 max          2 x 2 / 2    64 x  64 x  64   ->    32 x  32 x  64
    6 conv    128  3 x 3 / 1    32 x  32 x  64   ->    32 x  32 x 128 0.151 BF
    7 max          2 x 2 / 2    32 x  32 x 128   ->    16 x  16 x 128
    8 conv    256  3 x 3 / 1    16 x  16 x 128   ->    16 x  16 x 256 0.151 BF
    9 max          2 x 2 / 2    16 x  16 x 256   ->     8 x   8 x 256
   10 conv    512  3 x 3 / 1     8 x   8 x 256   ->     8 x   8 x 512 0.151 BF
   11 max          2 x 2 / 1     8 x   8 x 512   ->     8 x   8 x 512
   12 conv    256  1 x 1 / 1     8 x   8 x 512   ->     8 x   8 x 256 0.017 BF
   13 conv    512  3 x 3 / 1     8 x   8 x 256   ->     8 x   8 x 512 0.151 BF
   14 conv    195  1 x 1 / 1     8 x   8 x 512   ->     8 x   8 x 195 0.013 BF
   15 yolo
   16 route  12
   17 conv    128  1 x 1 / 1     8 x   8 x 256   ->     8 x   8 x 128 0.004 BF
   18 upsample            2x     8 x   8 x 128   ->    16 x  16 x 128
   19 route  18 8
   20 conv    195  1 x 1 / 1    16 x  16 x 384   ->    16 x  16 x 195 0.038 BF
   21 yolo
Total BFLOPS 1.035
```

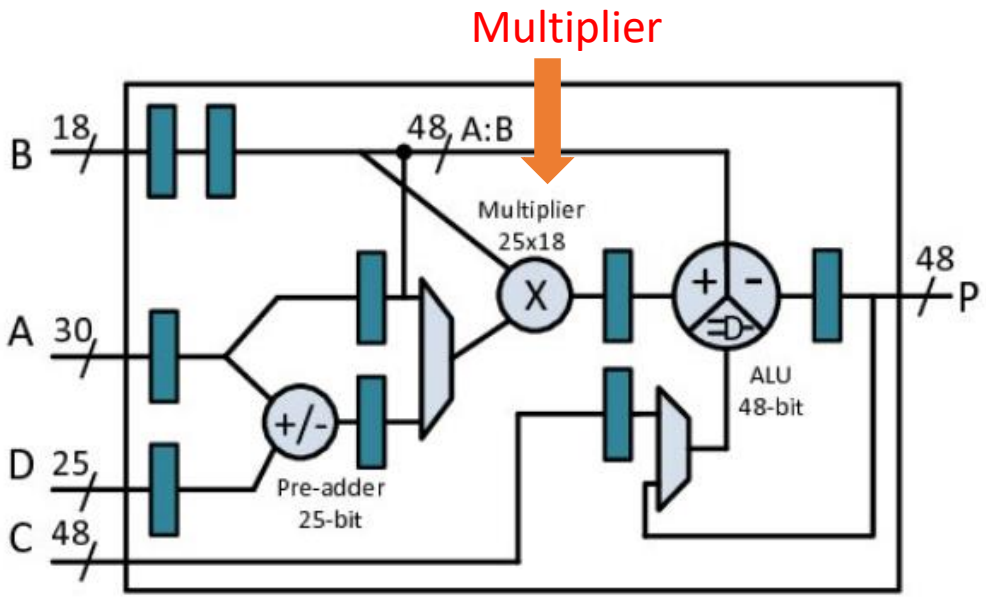
Nexys A7 FPGA board

- Xilinx Artix-7 FPGA XC7A100T-1CSG324C
- 15,850 logic slices
  - Each with four 6-input LUTs and 8 FFs
- 4,860 Kbits of fast block RAM
- 240 DSP slices (constrained to To, Ti)
- Internal clock speeds exceeding 450 MHz
- 128 MB DDR2 Memory
- USB-JTAG port for FPGA programming and communication



Wait. What is DSP?

- The Xilinx DSP48E block is a building block for DSP applications that use supported devices.
- We can implement a multiplier using LUT or DSP




```sv
`timescale 1ns / 1ps

module mul_dsp(
    clk,
    w_i,
    x_i,
    y_o
);
parameter DATA_WIDTH = 16;
input clk;
input signed  [    DATA_WIDTH -1 :0] w_i;
input signed  [    DATA_WIDTH -1 :0] x_i;
output signed  [2 * DATA_WIDTH -1 :0] y_o;

// Combinational clk
assign y_o = x_i * w_i;

endmodule
```

(a) Multiplier using DSP.

```sv
`timescale 1ns / 1ps
(*use_dsp48 ="no"*)
module mul_lut(
    clk,
    w_i,
    x_i,
    y_o
);
parameter DATA_WIDTH = 16;
input clk;
input signed  [    DATA_WIDTH -1 :0] w_i;
input signed  [    DATA_WIDTH -1 :0] x_i;
output signed  [2 * DATA_WIDTH -1 :0] y_o;

// Combinational clk
assign y_o = x_i * w_i;

endmodule
```

(b) Multiplier using LUT.

[1]. DSP48E1 https://docs.xilinx.com/r/2021.1-English/ug1483-model-composer-sys-gen-user-guide/DSP48E1
[2]. https://www.researchgate.net/publication/282398023_Mapping_for_maximum_performance_on_FPGA_DSP_blocks

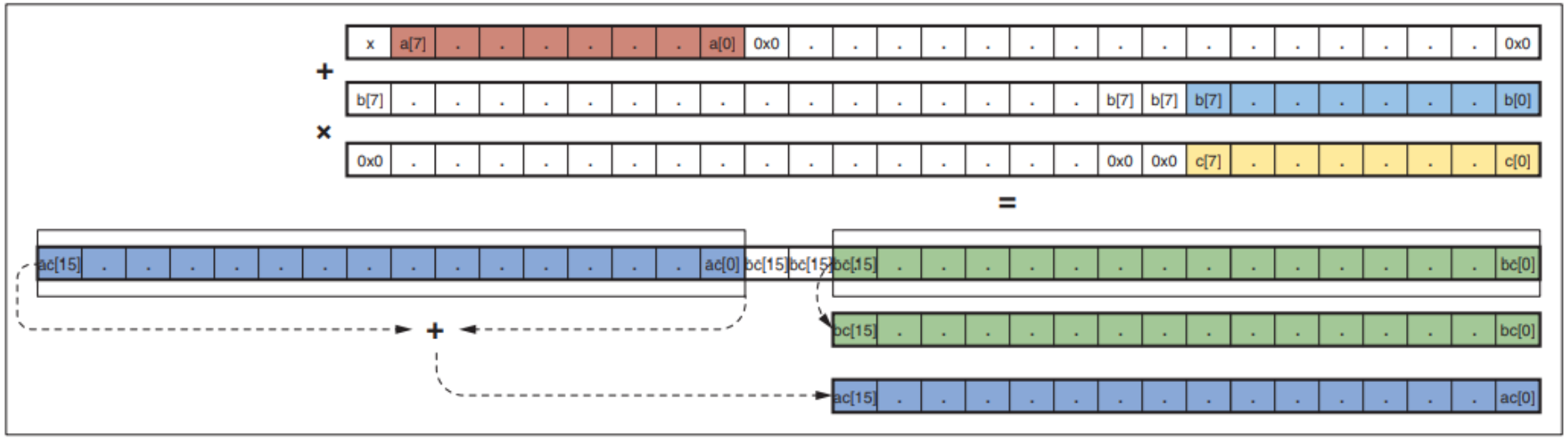
Wait. What is DSP?

- Table shows the resources for different DATA_WIDTHs.
  - 240 multipliers cost 14,640 LUTs (or 23.09% of the total LUTs in a Nexys A7-100T board)
  - **Utilizing 240 DSPs for 240 multipliers** saves 23.09% of the LUTs for other modules.

DATA_WIDTH	mul_dsp		mul_lut	
	LUT	DSP	LUT	DSP
4	23	0	23	0
6	36	0	36	0
8	61	0	61	0
9	88	0	88	0
10	102	0	102	0
11	0	1	132	0
12	0	1	148	0
16	0	1	265	0
24	0	2	595	0
32	47	4	1027	0

Multipliers and DSPs

- Can we do better? How about one DSP for two multipliers?
  - (a+b)*c = a*c + b*c => If a, b, and c are INT8, can compute (a+b)*c in one DSP

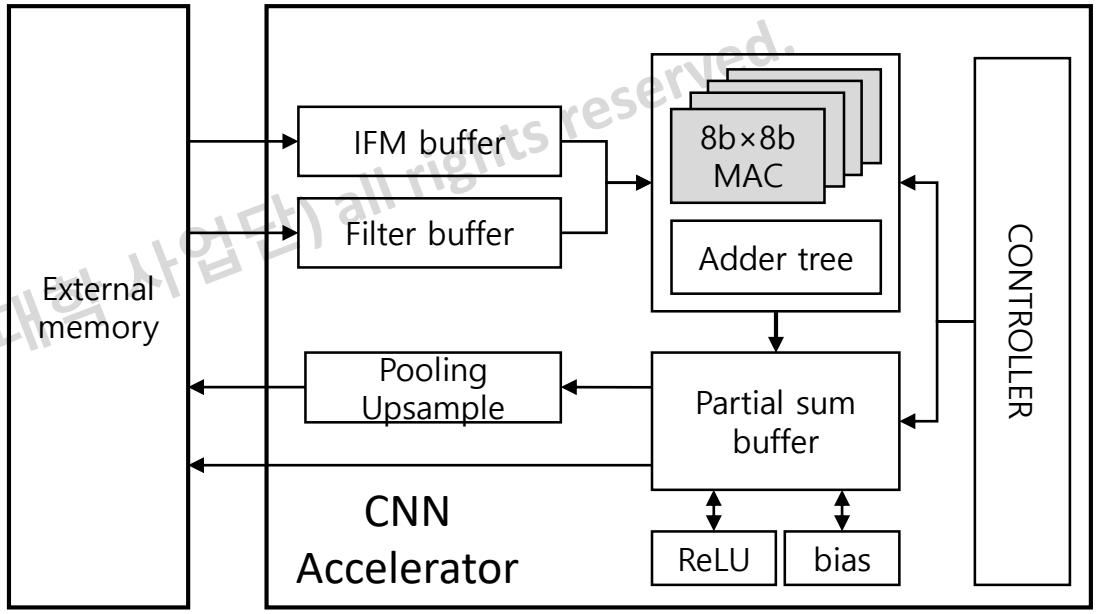


- How far can we go? How about designing **576 multipliers [1]?**
  - 480 multipliers from 240 DSPs and 96 ones from LUTs
  - Peak throughput: 112 GOPS@100MHz => **Maximum speed: ~108 frame per second**

[1] Minsik Kim et al. "Low-latency FPGA Accelerator for YOLOv3-tiny with Flexible Layerwise Mapping and Dataflow," IEEE Transactions on Circuits and Systems I: Regular Papers Volume: 71, Issue: 3, March 2024)

Inference Engine

- Processing Element (PE): MAC, Adder Tree
- Buffers
  - Input feature map (IFM)
  - Filters
  - Intermediate results or partial sum
- Pooling or Upsampling
- Controller



Inference Engine

- How to accelerate a CNN layer?
  - To generate one output pixel, compute multiplications and additions in parallel (Ti)
  - Compute multiple output pixels in parallel (To)

Convert input_uint8 and weights_int8 into matrices (im2col_cpu_int8)

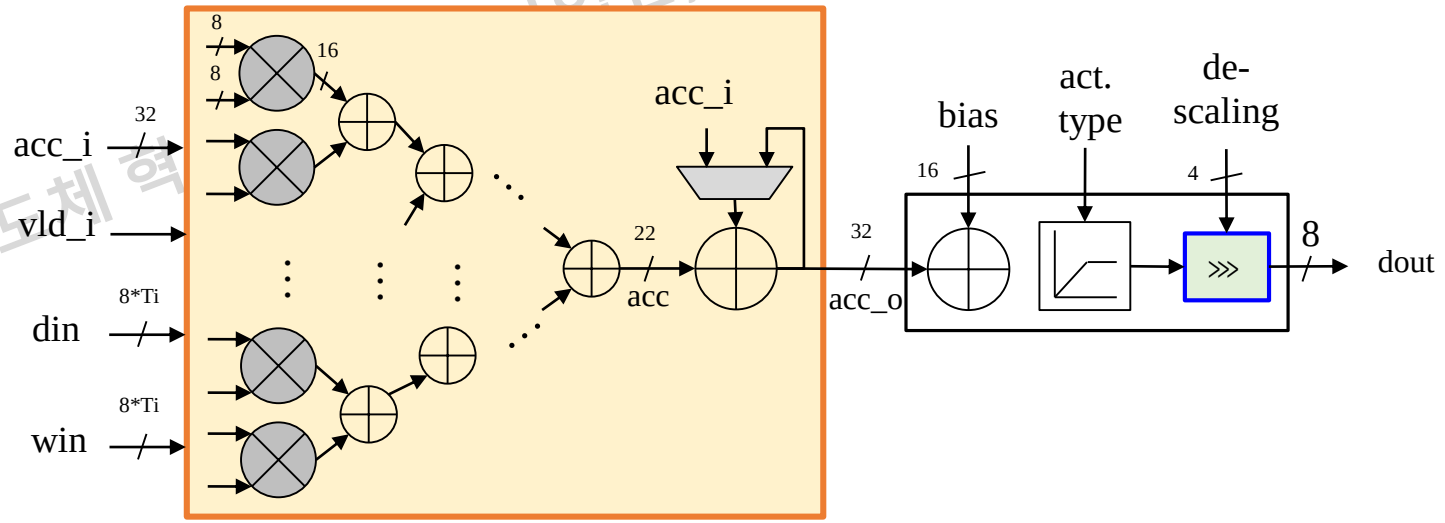
Do matrix multiplication (**gemm_nn_int8_int32**)

Do bias addition (+ biases_quant)

Do activation (RELU: f(x) = max(0,x))

Do descaling or de-quantization (divide by (input_quant_multiplier* weights_quant_multiplier))

Compute input_uint8
max(input*input_quant_multiplier, 255)



Verification: Check Model Parameters

```
387             char weightfile [100];
388             char biasfile   [100];
389             char scalefile  [100];
390
391             sprintf(weightfile  , "C:/skeleton/bin//log_param/CONV%02d_param_weight.hex", j);
392             sprintf(biasfile    , "C:/skeleton/bin//log_param/CONV%02d_param_biases.hex", j);
393             sprintf(scalefile   , "C:/skeleton/bin//log_param/CONV%02d_param_scales.hex", j);
394             FILE* fp_w = fopen(weightfile, "w");
395             FILE* fp_b = fopen(biasfile, "w");
396             FILE* fp_s = fopen(scalefile, "w");
397
398             int f;
399             for (f = 0; f < l->n; f++) {    // Out_channel
400                 // Weights
401                 //{{{
402                 for (int i = 0; i < filter_size; ++i) {     // Filter size
403                     int w_index = f * filter_size + i;
404                     fprintf(fp_w, "%08x\n", l->weights_int8[w_index]);
405                 }
406                 //}}}
407
408                 // Biases
409                 //{{{
410                 fprintf(fp_b, "%08x\n", l->biases_quant[f]);
411                 //}}}
412
413                 // Dequantization or Scaling
414                 //{{{
415                 // Find the input quantization factor for the next CONV layer
416                 int next_input_quant_multiplier = 1;
417                 for (int z = l->index+ 1; z < net.n; ++z) {
418                     if (net.layers[z].type == CONVOLUTIONAL) {
419                         next_input_quant_multiplier = net.layers[z].input_quant_multiplier;
420                         break;
421                     }
422                 }
423
424                 int scale = (l->input_quant_multiplier * l->weights_quant_multiplier) / next_input_quant_multiplier;
425                 fprintf(fp_s, "%08x\n", scale);
426                 //}}}
427             }
```

CONV00_param_weight.hex

```
413  0000000d
414  00000000
415  00000002
416  00000003
417  00000012
418  00000013
419  00000002
420  00000019
421  fffffffd
422  00000006
423  00000008
424  0000000a
425  00000003
426  0000001b
427  fffffff1
428  fffffff8
429  0000000d
430  fffffff5
431  fffffe3
432  ffffffed
```

CONV00_param_biases.hex

```
1   000002c2
2   ffffffc0
3   00000791
4   fffffd03
5   00000ab8
6   00000c44
7   00000053
8   000004c5
9   0000097a
10  fffe7f0
11  00000763
12  000006bb
13  000003e7
14  0000044b
15  000001bb
16  fffff6e6
```

CONV00_param_scales.hex

```
1   00000080
2   00000080
3   00000080
4   00000080
5   00000080
6   00000080
7   00000080
8   00000080
9   00000080
10  00000080
11  00000080
12  00000080
13  00000080
14  00000080
15  00000080
16  00000080
```

Verification: Check Input Feature Maps

```
148     if (run_single_image_test) {
149         // Input Feature Map (IFM)
150         char file_input_femap[100];
151         snprintf(file_input_femap, sizeof(file_input_femap), "C:/skeleton/bin/log_feamap/CONV%02d_input.hex", state.index);
152         FILE* fp = fopen(file_input_femap, "w");
153
154         // Data Format: [Channel, Width, Height]
155         for (int idx = 0; idx < l.h * l.w; idx++) {  // IFM: Pixel index in ONE feature map
156             for (int chn = 0; chn < l.c; chn++) {    // IFM: Channel/index of an feature map
157                 int i = chn * l.h * l.w + idx;       // IFM: Pixel index
158                 uint8_t pixel = state.input_uint8[i];
159                 fprintf(fp, "%02x\n", pixel);
160             }
161         }
162         if (fp) fclose(fp);
163     }
164     //
165
166     //
167     int m = l.n;
168     int k = l.size*l.size*l.c;
169     int n = out_h*out_w;
170     int8_t *a = l.weights_int8;
171     uint8_t* b = (uint8_t*)state.workspace; //int8_t *b = (int8_t *)state.workspace;
172     conv_t *c = output_q;    // int32_t
173
174     // Use GEMM (as part of BLAS)
175     im2col_cpu_int8(state.input_uint8, l.c, l.h, l.w, l.size, l.stride, l.pad, b);
176
177     int t;    // multi-thread gemm
178     #pragma omp parallel for
179     for (t = 0; t < m; ++t) {
180         gemm_nn_int8_int32(1, n, k, 1, a + t*k, k, b, n, c + t*n, n);
181     }
182     free(state.input_uint8);    //free(state.input_int8);
```

int const run_single_image_test = 1;
(yolov2_forward_network_quantized.c)

Layer 0
256×256×3

CONV00_input.hex

196589	04
196590	05
196591	04
196592	04
196593	05
196594	05
196595	04
196596	07
196597	04
196598	05
196599	07
196600	08
196601	07
196602	0d
196603	05
196604	05
196605	07
196606	06
196607	05
196608	08

Layer 2
128×128×16

CONV02_input.hex

262125	07
262126	08
262127	08
262128	00
262129	0e
262130	04
262131	0b
262132	00
262133	13
262134	16
262135	00
262136	05
262137	1a
262138	00
262139	12
262140	1a
262141	08
262142	06
262143	05
262144	00

Layer 4
64×64×32

CONV04_input.hex

131053	0c
131054	07
131055	18
131056	06
131057	19
131058	03
131059	05
131060	01
131061	01
131062	00
131063	02
131064	0e
131065	00
131066	08
131067	1a
131068	00
131069	04
131070	17
131071	1a
131072	04

Layer 20:
16×16×384

CONV20_input.hex

98285	00
98286	0e
98287	00
98288	00
98289	12
98290	07
98291	00
98292	00
98293	16
98294	00
98295	0e
98296	12
98297	00
98298	23
98299	00
98300	00
98301	00
98302	49
98303	00
98304	20

Verification: Check Output Feature Maps

```
192     // Activation
193     if (l.activation == RELU) {
194         for (i = 0; i < l.n * out_size; ++i) {
195             output_q[i] = (output_q[i] > 0) ? output_q[i] : 0;
196         }
197     }
198
199     // De-scaling or De-quantization
200     float ALPHA1 = 1 / (l.input_quant_multiplier * l.weights_quant_multiplier);
201     for (i = 0; i < l.outputs; ++i) { ... }
204
205
206     // Write data for the HW verification
207     //{{{
208     if (run_single_image_test) {
209         // Output Feature Map (OFM)
210         int z;
211         int next_input_quant_multiplier = 1;
212         for (z = state.index + 1; z < net.n; ++z) {
213             if (net.layers[z].type == CONVOLUTIONAL) {
214                 next_input_quant_multiplier = net.layers[z].input_quant_multiplier;
215                 break;
216             }
217         }
218         char file_output_femap[100];
219         snprintf(file_output_femap, sizeof(file_output_femap), "C:/skeleton/bin/log_feamap/CONV%0
220         FILE* fp = fopen(file_output_femap, "w");
221
222         // Data Format: [Channel, Width, Height]
223         for (int idx = 0; idx < out_size; idx++) {  // OFM: Pixel index in ONE feature map
224             for (int chn = 0; chn < l.n; chn++) {   // OFM: Channel/index of an feature map
225                 int i = chn * out_size + idx;       // OFM: Pixel index
226                 uint8_t pixel = max_abs(l.output[i] * next_input_quant_multiplier, MAX_VAL_UINT_8
227                 fprintf(fp, "%02x\n", pixel);
228             }
229         }
230         if (fp) fclose(fp);
231     }
232     //}}}
233     free(output_q);
234 }
```

Layer 0
256×256×16

CONV00_output.hex

1048557	05
1048558	06
1048559	05
1048560	00
1048561	0a
1048562	02
1048563	0b
1048564	00
1048565	11
1048566	14
1048567	00
1048568	02
1048569	16
1048570	00
1048571	11
1048572	15
1048573	05
1048574	06
1048575	02
1048576	00

Layer 2
128×128×32

CONV02_output.hex

524269	0b
524270	07
524271	08
524272	00
524273	19
524274	03
524275	00
524276	00
524277	00
524278	00
524279	00
524280	0e
524281	00
524282	00
524283	1a
524284	00
524285	00
524286	17
524287	18
524288	00

Layer 4
64×64×64

CONV06_output.hex

131053	00
131054	02
131055	0f
131056	08
131057	00
131058	00
131059	00
131060	00
131061	00
131062	00
131063	04
131064	00
131065	06
131066	00
131067	00
131068	10
131069	00
131070	06
131071	00
131072	05

Layer 20:
16×16×195

CONV20_output.hex

49901	fc
49902	f9
49903	f9
49904	f6
49905	f9
49906	f9
49907	f7
49908	f7
49909	f6
49910	f9
49911	f8
49912	f8
49913	f6
49914	f7
49915	f5
49916	f8
49917	f8
49918	f5
49919	fa
49920	f6

Outlines

- About AIX2024
- Quick Start
  - Background
  - Code structure: Compile and Run
  - Quantization
  - Hardware
- TODO
- Evaluation and Award

Tasks and Requirements

- 중간 평가 (4.5): **Only for progress checking, No elimination**
  - Software: Do quantization (Find a set of input and weight multipliers).
  - Implement an accelerator engine
  - Do RTL simulation for **the three first CONV layers**.

  ⇒ Submit your code, captured results, and report(1).
- 최종 평가 (5.31)
  - Complete the design for an accelerator engine
    - Compare the output from the hardware simulation and the software.
  - Do FPGA implementation
    - Pack an IP and integrate it to the system
    - Synthesize the top system
  - Verify the design on the board (Output, Speed)

  ⇒ Submit your code, captured results, and report(1).

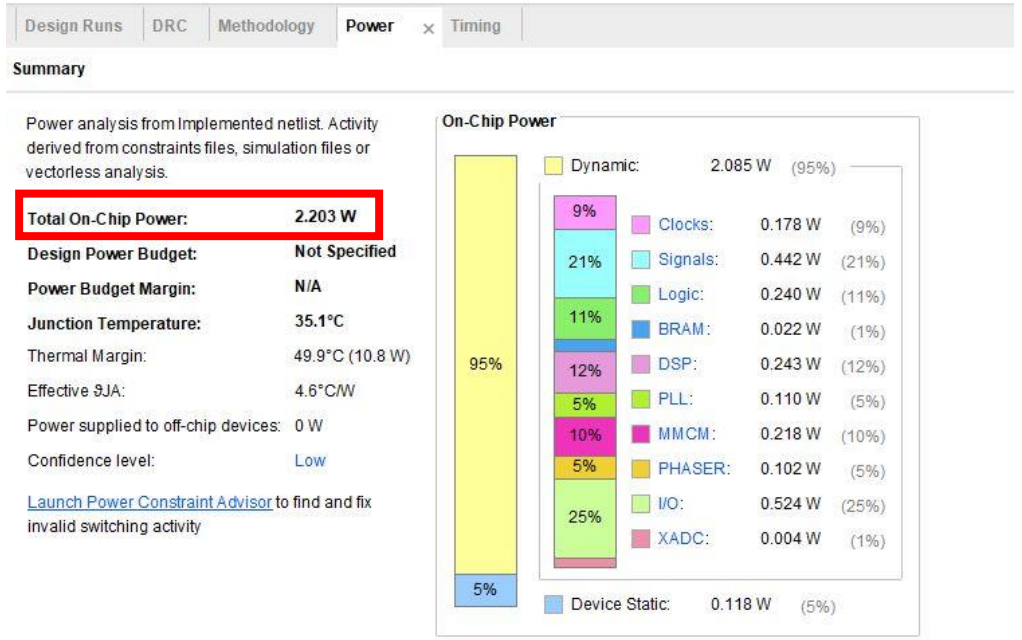
(1)-(2) The template and the submission guidelines will be provided later.

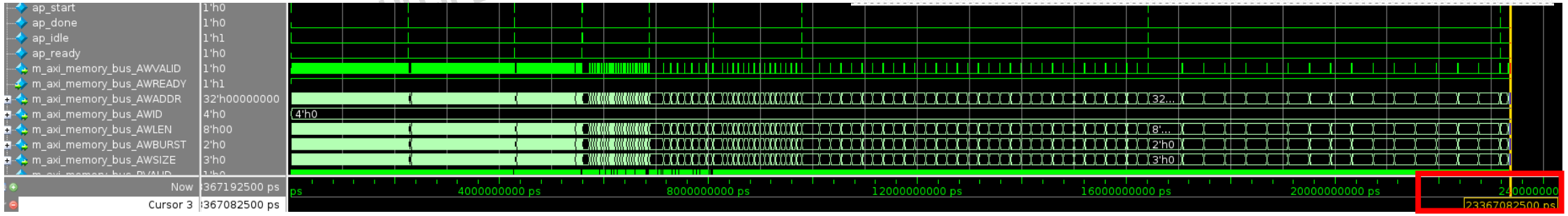
Evaluation: Midterm (Progress checking)

- Software (25p)
  - Check how to understand a Deep network operation, and prepare the data for H/W implementation.
  - ***Mean Average Precision*** (mAP(%)): Evaluate the accuracy of the model and activation quantization.
  - ***Test vector: Generate input data*** (e.g., an input image, filters) and the ***output data*** (e.g., feature maps for Layers 1-3)
- Hardware Design (50p)
  - Design DSP/MAC, buffers, and a controller (e.g., FSM) for multi-layer operations
  - Implement normalization (e.g., scaling/descaling, adding a bias), activation, quantization, and max-pooling.
  - Verify the outputs of RTL simulation for three CONV layers.
- Presentation (25p)
  - Explain the results.
  - Explain your quantization method, dataflow, implementation ideas, and algorithms.
  - Distribute tasks among members, and make a plan

결과평가: Accuracy/Speed/Power

- **Accuracy:** Mean average precision (mAP)
  - Calculate the AP at IoU threshold 0.5 for INT8
- **Inference speed:** measures the number of frames your CNN accelerator IP processes in a second
  - RTL simulation result: 23.3@200MHz, 2 cycles for DRAM latency (Use the FPGA clock).
- **Energy:**
  - Implement your design on an FPGA board
  - Measure the on-chip power of the design after FPGA implementation





Evaluation: Final

- Objective evaluation[1]
  - **10^4 / Energy * ReLU (mMAP – 0.2) * ReLU (fps – 5).**
  - Where mAP is the INT8 quantized accuracy

[1] *if the team completes the design until the FPGA verification.*

- Subjective evaluation[2]
  - Software (25p)
  - Hardware Design (50p)
  - Presentation (25p)

[2] *When the design is NOT completed.*

대회 최우수팀 수상 및 상품

- 평가 점수 기준 최우수팀 및 우수팀 선정 (3등까지)
  - **1등 :** 노트북 (one team)
  - **2등 :** 갤럭시 탭 (one team)
  - **3등 :** **1**갤럭시 버즈 (two teams)
- 중간 평가 결과 우수한 팀에게 Starbucks coffee coupons 증정

Q&A