

แบบรายงานทางเทคนิค (Technical Report)

การเขียนรายงานทางเทคนิค แบ่งออกเป็น 3 ส่วน พร้อมคำอธิบายแต่ละหัวข้อ ดังนี้

ส่วนที่ 1 นิยามและอธิบายโครงงาน

ชื่อโครงงาน ระบบการวิเคราะห์วิดีโอบนหลักยุทธ์ของทางบันสนสถาปัตยกรรม FPGA.....

สาขาวิศวกรรมไฟฟ้า.....

วิเคราะห์และกำหนดประเด็นปัญหา (C)

1. อธิบายลักษณะและที่มาของปัญหาทางวิศวกรรมโดยสังเขป

ปัจจุบันงานด้านระบบการวิเคราะห์วิดีโอบนหลักยุทธ์ของทางบันสนสถาปัตยกรรม FPGA กำลังมีบทบาทอย่างมากกับการสร้างสรรค์ผลงาน ซึ่งไม่จำกัดแต่เฉพาะกับคนในวงแวดล้อมเดียว แต่ยังสามารถมาประยุกต์ใช้กับงานด้านต่าง ๆ เช่น ใช้ในการตรวจจับการจราจร และการสัญจربนทึกถนนในเมืองอัจฉริยะ, เป็นระบบแจ้งเตือนสุขภาพและความปลอดภัยในโรงพยาบาล, ทำการ self-checkout และวิเคราะห์ด้านธุรกิจค้าปลีก และอื่น ๆ อีกหลากหลาย เนื่องจาก FPGA เป็น IC (Integrated Circuit) ที่สามารถ Program ได้ ซึ่ง FPGA ดีกว่า CPU ในเรื่องค่า Latency ต่ำ กินพลังงานน้อย การเชื่อมต่อกับอินพุตที่มีแบบวิดีโอที่สูงมาก(High Bandwidth) และ ความเร็วในการประมวลผลเร็วมาก ซึ่งสามารถวิเคราะห์วิดีโอบนหลักยุทธ์ของทางได้ โดยมีการนำตัว VVAS (Vitis Video Analytics SDK) ที่ออกแบบด้วยบริษัท Xilinx โดยที่ผู้จัดทำได้เห็นประโยชน์ในการนำระบบ VVAS นี้มาช่วยแก้ไขปัญหา ในการใช้งานทางด้าน AI ที่เรียกว่า Vitis AI ให้มีความสามารถในการจัดการและการทำงานของระบบ AI บนโมดูล(module) พิเศษ หรือเรียกว่า DPU (Deep Processing Unit), VCU (Video Codec Unit) และ Multi-scaler ที่เป็น HLS (High-Level Synthesis) เพื่อที่จะช่วยเพิ่มประสิทธิภาพในการทำงานของระบบ ให้สามารถวิเคราะห์วิดีโอบนหลักยุทธ์ของทางได้ บนสถาปัตยกรรม FPGA (FPGA SoC) นั้นเอง

2. กลุ่มผู้ใช้งาน

เพื่อตอบโจทย์และแก้ปัญหาให้กับอุตสาหกรรมที่สำคัญต่างๆของประเทศไทยได้แก่ อุตสาหกรรมด้านสุขภาพ อุตสาหกรรมด้านความปลอดภัย อุตสาหกรรมด้านการผลิตและโรงงาน รวมทั้งโซลูชันทางด้าน Smart City เป็นต้น

3. กระบวนการได้มาซึ่งรายละเอียดและความต้องการทางวิศวกรรม

ระบบการวิเคราะห์วิดีโอบนหลักยุทธ์ของทางบันสนสถาปัตยกรรม FPGA กำลังมีบทบาทอย่างมากกับการสร้างสรรค์ผลงาน ซึ่งไม่จำกัดแต่เฉพาะกับคนในวงแวดล้อมเดียว แต่ยังสามารถมาประยุกต์ใช้กับงานด้านต่าง ๆ เช่น ใช้ในการตรวจจับการจราจร และการสัญจربนทึกถนนในเมืองอัจฉริยะ, เป็นระบบแจ้งเตือนสุขภาพและความปลอดภัยในโรงพยาบาล, ทำการ self-checkout และวิเคราะห์ด้านธุรกิจค้าปลีก และอื่น ๆ อีกหลากหลาย เนื่องจาก FPGA เป็น IC (Integrated Circuit) ที่สามารถ Program ได้ ซึ่ง FPGA ดีกว่า CPU ในเรื่องค่า Latency ต่ำ กินพลังงานน้อย การเชื่อมต่อ

กับอินพุตที่มีแบบวิดีโอแบบหลายช่องทางได้ โดยมีการนำตัว VVAS (Vitis Video Analytics SDK) ที่ออกแบบด้วยบริษัท Xilinx โดยที่ผู้จัดทำได้เห็นประโยชน์ในการนำระบบ VVAS นี้มาช่วยแก้ไขปัญหาในการใช้งานทางด้าน AI ที่เรียกว่า Vitis AI ให้มีความสามารถในการจัดการและการทำงานของระบบ AI บนโมดูล(module) พิเศษ

ลักษณะและความมุ่งหมายของการออกแบบ (C)

1. วัตถุประสงค์

- 1.1 เพื่อพัฒนาระบบวิเคราะห์วิดีโอแบบหลายช่องทางสำหรับการ detection และ classification ในอุตสาหกรรมทางด้านความปลอดภัย
- 1.2 ทำการ Retrain โมเดล AI ชนิด YOLO (You Only Look Once) เพื่อเอาไปใช้ในอุตสาหกรรมทางด้านความปลอดภัย
- 1.3 เพื่อสามารถวิเคราะห์วิดีโอด้วยหลายช่องทางพร้อมกัน (อย่างน้อย 4 ช่องทาง)
- 1.4 เพื่อเป็นประโยชน์ในด้านการใช้งาน AI ในรูปแบบ Edge Computing ได้อย่างมีประสิทธิภาพ
- 1.5 สามารถถ่ายทอดได้ง่ายขึ้น และต่อยอดในการใช้งาน AI ที่เป็น Deep Technology ได้
- 1.6 เพื่อให้การต่อยอดในรูปแบบการใช้งานของ AI
- 1.7 เพื่อให้ผู้ที่สนใจที่ต้องการศึกษาเรียนรู้การใช้งานทางด้านระบบการวิเคราะห์วิดีโอแบบหลายช่องทางบนสภาพปัจจุบัน FPGAs
- 1.8 นำมาประยุกต์ใช้เพื่อความสะดวกรวดเร็วของผู้ใช้และผู้ที่จะศึกษาเรียนรู้ด้านการวิเคราะห์วิดีโอแบบหลายช่องทาง

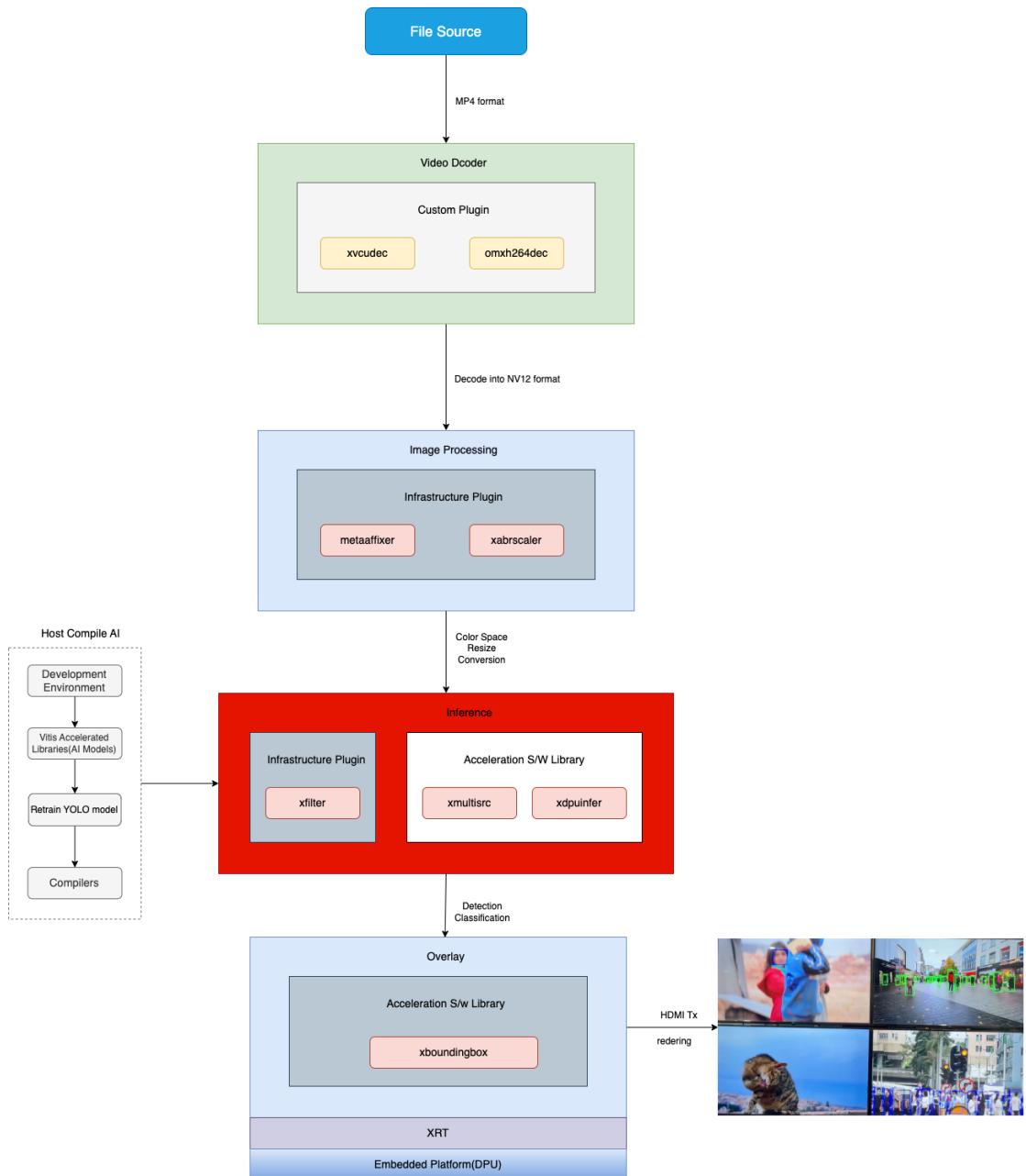
2. ลักษณะเฉพาะที่ได้คิดค้นขึ้นหรือขอบเขตในการออกแบบ

- 2.1 จะทำการประมวลผล AI แบบหลายช่องทางบนบอร์ด FPGA ที่เป็นบอร์ด ZCU104
- 2.2 ทำการปรับแต่ง AI model กับ Application (AI Detection Application) ในชีวิตจริง
- 2.3 ปรับ AI ให้สามารถใช้งานกับบอร์ด ZCU104 ที่เป็นสถาปัตยกรรม FPGA ได้

2.4 ทำการใช้งาน AI model แบบ Single stage ML ได้

การออกแบบทางวิศวกรรม (D)

จากรูปที่ 1 เป็นระบบการวิเคราะห์วิดีโอแบบหลายช่องทางบนสถาปัตยกรรม FPGA ซึ่งแต่ละบล็อกเป็นปลั๊กอินต่างๆ ที่ใช้ ที่เริ่มต้นด้วยบล็อกแรกซึ่งเป็น File Source ที่เป็นวิดีโอมี Format เป็น MP4 บล็อกต่อมาเป็น Video Decoder ที่เป็น Custom Plug-in ประกอบไปด้วยตัว Plug-in สองตัวคือ xvcudec และ omxh264dec ที่ทำงานที่แปลง format จาก MP4 ไปเป็น NV12 (เป็น format วิดีโอสำหรับทำการประมวลผล AI) ส่วนบล็อกต่อมาเป็นบล็อก Image Processing ซึ่งเป็น Infrastructure Plug-in สองตัวเช่นปลั๊กอิน mettaaffixer และ abrscaler ที่ทำงานที่ปรับขนาดของภาพให้เหมาะสมกับโมเดล AI ที่จะเอาไปใช้ ส่วนบล็อกต่อไปเป็น Inference ที่เป็นส่วนที่ทำการประมวลผล AI ซึ่งจะประกอบไปด้วย Infrastructure Plug-in ซึ่งว่า xfilter และ Acceleration S/W Library ซึ่งว่า xmultipsrc และ xdpuinfer ที่ทำงานที่ Detection และ Classification (ตามชนิดของแต่ละโมเดล AI ที่จะเอาไปใช้) และบล็อกสุดท้ายคือ Overlay ที่เป็น Acceleration S/W Library ซึ่ง xboudingbox ที่ทำงานที่ตีกรอบภาพหลังจากการ Inference เสร็จจากบล็อก Inference ข้างบนโดยกระบวนการทำงานทุกอย่างผ่าน DPU (Deep Processing Unit) ซึ่งเป็นบอร์ด FPGA



ภาพที่ 1 ภาพรวมระบบการทำงาน

ระบบนี้รับข้อมูลเข้า (Input Data/File source) จากกล้อง USB หรือจากกล้อง CSI MIPI (USB/CSI) จากไฟล์วิดีโอ (Video file) หรือจากการสตรีมผ่านโปรโตคอล RTSP (RTSP protocol) ด้วยทำการ decode โดยใช้มาตรฐาน H264

* H264 คือมาตรฐานการบีบอัดข้อมูลของสัญญาณภาพและเสียง ที่ให้ความคมชัดเท่าเดิมแต่ขนาดไฟล์จะเล็กลง

2. อธิบายถึงลักษณะทางโครงสร้าง

ด้านฮาร์ดแวร์ (Hardware)

- ทำการใส่ SD Card ไปในช่องใส่ตัว SD Card ของบอร์ด

2. ทำการเปิด Switch ของบอร์ดให้เป็นโหมด boot (Boot Model) 1(On), 2(OFF), 3(OFF), 4(OFF)
3. เชื่อมตัวบอร์ดกับตัว Monitor ผ่านพอร์ต HDMI

ลำดับ	ชื่ออุปกรณ์ในระบบ	หน้าที่
1	FPGA	เป็นบอร์ด FPGA โมเดล ZCU104 ทำนาที ประมวลผล AI และทำการ Stream ออกไป ตามจอ (Monitor)
2	Micro USB Wire	สำหรับเชื่อมคอมพิวเตอร์กับพอร์ต UART เพื่อทำการ Scan หา หรือทำการ set IP Address ของบอร์ด
3	SD Card	เก็บอิมเมจไฟล์ (Image File) ที่เอาไป Build โครงการบนบอร์ด FPGA (ZCU 104) และความมีความจุมากกว่า 8GB
4	HDMI 2.0	เพื่อการ Stream ออกที่มีขนาดความ ละเอียด 3840x2160 (4k) 60fps
5	HDMI 2.0 Wire	เชื่อกันระหว่างตัวพอร์ต HDMI ของ Monitor กับตัวบอร์ด



ด้านซอฟต์แวร์

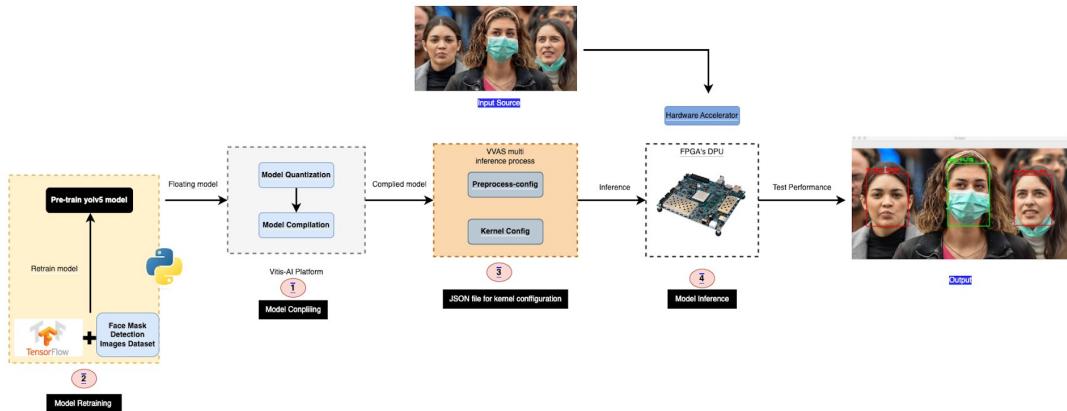
1. ทำการ Burn อิมเมจไฟล์ (Image File) เข้าไปในตัว SD Card
 2. ทำการ Compile โมเดล AI จำนวน 16 ตัวโดยทำการ Test Performance เพื่อ
เอาไปปรับใช้บนโครงงาน
 3. ทำการ Retrain โมเดล AI ชนิด YOLO (You Only Look Once) เพื่อเอาไปใช้ใน
อุตสาหกรรมความปลอดภัย
 4. ทำการการเปิดบอร์ดและทำการ boot เพื่อเริ่มการใช้งาน
 5. ทำการ Copy ตัวโมเดล AI ที่ทำการ Compile เสร็จเข้าไปที่บอร์ด
 6. ทำการ config ตัว kernel ของ DPU และ Bounding Box ที่เป็นไฟล์ JSON
- ผลที่ออกมายังเป็นการ Stream การวิเคราะห์วิดีโอ 4 ช่องพร้อมกัน

ลำดับ	ชื่ออุปกรณ์ในระบบ	หน้าที่
1	Vitis Unified Software 	Vitis AI เป็น Programming API ที่ สามารถทำให้ AI application รันบน Edge และ cloud ได้ที่มี tool (เป็น Environment สำหรับ Quantize, Compile หรือ Build ตัว AI บนแพลต ฟอร์มของ Xilinx)
2	PetaLinux tool 	เป็นตัว Linux ที่ออกแบบมาเพื่อใช้กับ อุปกรณ์ของ Xilinx โดยเฉพาะ (เป็น OS ที่ ใช้บนบอร์ด)
3	CoolTerm	เป็น Application สำหรับทำการ Scan หา IP Address ของบอร์ด

		
4	Git  	เก็บตัวอย่าง และ Source Code ที่ใช้กับ Framework ของ WVAS เช่น Pre-optimized model ของโมเดล Zoo เป็นต้น
5	Ubuntu 18.04/20.04 	สำหรับเอาไปทำการ Compile และ Retrain ตัว AI
6	Balena Etcher flashing tool 	สำหรับทำการ Burn วิมเมจไฟล์ (Image File) เข้าไปในตัว SD Card

3. ให้ระบุในลักษณะที่จะทำให้มีความชำนาญในระดับสามัญในสาขาวิชาการที่เกี่ยวข้องเข้าใจ และสามารถปฏิบัติตามได้

3.1 การพัฒนาโครงการ



ภาพที่ 2 แผนการทำการทดลองของโครงการ

Multichannel ML เป็น ML ที่ทำการประมวลผล AI ไม่เดลมากกว่า 1 ตัวในเวลาเดียวกัน โดยแบ่งเป็น 2 วิธีคือ

1. Single stage ML : เป็น ML ที่ใช้ AI ไม่เดลเพื่อทำการประมวลผลตัวเดียวบน Input images
2. Multi-stage ML: เป็น ML ที่ใช้ AI ไม่เดลเพื่อทำการประมวลผลมากกว่าหนึ่งตัวบน Input images

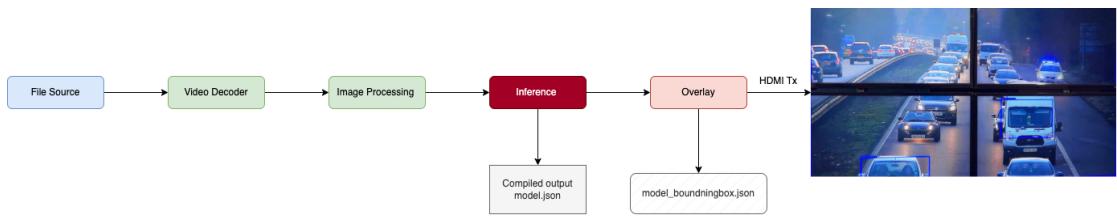
ในส่วนนี้เราจะพัฒนาระบบการวิเคราะห์วิดีโอ 4 ช่องทางโดยใช้ตัวบอร์ด FPGA ไม่เดล ZCU104 ที่ได้รับการออกแบบระบบบันทึกสามารถวิเคราะห์ช่องวิดีโอด้วยตัวบอร์ด ZCU104 ที่มีความเร็ว 30fps ซึ่งเริ่มต้นด้วยการสร้างการวิเคราะห์หนึ่งช่องโดยใช้ไปป์ไลน์ (Single ML pipeline) ของ Single Stage ML โดยใช้แพลตฟอร์มของ VVAS (Vitis Video Analytics SDK) จากนั้นเราจะขยายขนาดเพิ่มเพื่อสร้างไปป์ไลน์ที่สามารถวิเคราะห์วิดีโอ 4 ช่องทาง (Four channels ML pipeline) ที่ประมวลผล 4 ช่องทางพร้อมๆกัน

เราจะพัฒนาระบบภายใต้ 3 ส่วนได้แก่

- ทำการปรับแต่งโมเดล (AI Optimization) ที่ร่วมรักบกับแพลตฟอร์ม VVAS
- ทำการ Retrain โมเดล AI ชนิด YOLO (You Only Look Once) เพื่อเอาไปใช้งานในด้านอุตสาหกรรม
- ทำการเขียนไฟล์ JSON เพื่อควบคุมการ Config ตัว kernel ของ DPU เช่นควบคุมการประมวลผล AI และการตีกรอบขอบเขตบนวัตถุที่ตรวจพบ

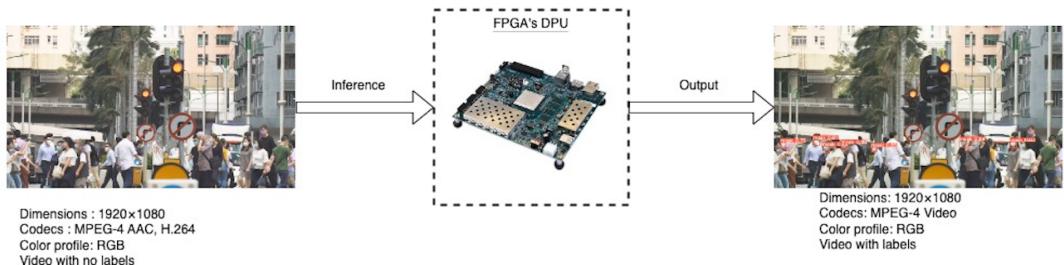
3.2 Single Stage Machine Learning

การวิเคราะห์วิดีโอแบบ Single stage ML เป็นดำเนินงานของการวิเคราะห์หนึ่งอินพุตภาพ (Single input image) โดยใช้การวิเคราะห์อย่างเดียว (Only one Machine Learning)



ดังนั้นเราต้องพูดที่ได้ก็จะเป็นวิดีโอด้วยเคราะห์เรียบร้อยโดยสามารถตีกรอบวัตถุที่เราจะทำการ Inference ยกตัวอย่างเช่น การตรวจจับรถ ตรวจจับใบหน้าคน หรือตรวจจับคนเดินทางถนน เป็นต้น

ตัวอย่างภาพที่ใช้ก่อนและหลังการทำ inference บน FPGA

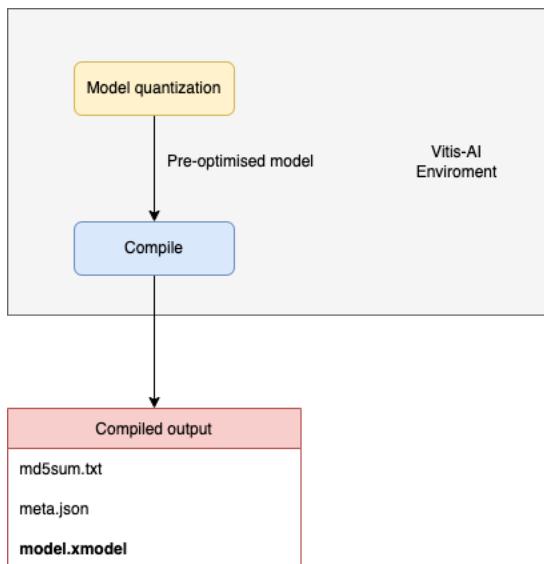


- ก่อนการทำ inference ข้อมูลเข้า (Input Data/File source) เป็นภาพที่ยังไม่มีการประมวลผล AI (ยังไม่มีการตีกรอบ)
- หลังการทำ inference เอ้าต์พุตจะเปลี่ยนเป็นภาพที่มีการประมวลผลเรียบร้อย (ภาพมีการตีกรอบ)

3.2.1 การ Compile โมเดล AI จากโมเดล Zoo ของ Xilinx

การ Compile โมเดล AI โดยใช้ Vitis-AI Compiler เป็นการแปลง (Convert) หรือปรับแต่ง (Optimize) โมเดลของ AI ให้เป็น format หนึ่งที่สามารถวิ่งได้ (Runnable) บนตัว DPU (Deep Learning Processing) ของบอร์ด FPGA

Compiler จะใช้ pre-trained หรือ pre-optimized โมเดลเป็นตัวอินพุตโมเดล โดยเริ่มต้นด้วยการแปลงอินพุตโมเดลให้เป็น format ของ XIR (Xilinx Intermediate Representation) ถูกออกแบบด้วยบริษัท Xilinx ซึ่งออกแบบมาเพื่อการใช้งานที่มีประสิทธิภาพที่ซับซ้อนสำหรับการปรับใช้กับตัว DPU บนแพลตฟอร์ม FPGA โดยการ



ขั้นตอนการ Compile นี้เราจะใช้เฟรมเวิร์คของ Caffe, TensorFlow และ Darknet ซึ่งเอาร์พุตของโมเดลที่ออกแบบอยู่ในรูปแบบของไฟล์ .xmodel ที่เป็นโมเดลที่สามารถใช้งานบนบอร์ด ZCU104 ที่เป็นบอร์ด FPGA ได้

ต่อไปนี้เป็นขั้นตอนเมื่อทำการ Compile โมเดล:

- ทำการ setup environment เช่น เตรียมพร้อม pre-train โมเดล สร้าง directory เพื่อเก็บโมเดลที่ Compile เสร็จ (Compiled output) เป็นต้น
- สร้าง script เพื่อให้ในการ compile สำหรับทุกเฟรมเวิร์คที่จะใช้

```
[aic@aic-001:~/AIC-internproject/vvas_project/source/Vitis-AI/models/AI-Model-Zoo$ cd ../../
[aic@aic-001:~/AIC-internproject/vvas_project/source/Vitis-AI$ cd vvas_model/
[aic@aic-001:~/AIC-internproject/vvas_project/source/Vitis-AI/vvas_model$ ls
arch.json  compile_cf_model.sh  compiled_output
[aic@aic-001:~/AIC-internproject/vvas_project/source/Vitis-AI/vvas_model$ cat arch.json
{"fingerprint": "0x1000020F6014406"}
[aic@aic-001:~/AIC-internproject/vvas_project/source/Vitis-AI/vvas_model$ cat compile_cf_model.sh
model_name=$1
modelzoo_name=$2
vai_c_caffe \
--prototxt ../models/AI-Model-Zoo/caffe_model/${modelzoo_name}/quantized/deploy.prototxt \
--caffemodel ../models/AI-Model-Zoo/caffe_model/${modelzoo_name}/quantized/deploy.caffemodel \
--arch ./arch.json \
--output_dir ./compiled_output/${modelzoo_name} \
--net_name ${model_name} \
--options "{\"mode\": \"normal\"}"
aic@aic-001:~/AIC-internproject/vvas_project/source/Vitis-AI/vvas_model$ ]]
```

- ทำการ Compile โมเดล

```
(vitis-ai-caffe) Vitis-AI workspace > cd vvas_model/
(vitis-ai-caffe) Vitis-AI workspace/vvas_model > ./compile_cf_model.sh cf_resnet18_imagenet_224_224_3_650_2.0
*****
* VITIS_AI Compilation - Xilinx Inc.
*****
*****Namespace batch size=1, inputs shape=None, layout="NCHW", model_files=['../models/AI-Model-Zoo/caffe_model/quantized/deploy.caffemodel'], model_type='caffe', named_inputs_shape=None, out_filename='/tmp/cf_resnet18_imagenet_224_224_3_650_2.0.org.xmodel', proto='../models/AI-Model-Zoo/caffe_model/quantized/deploy.prototxt'
(ERROR) No file found at /tmp/cf_resnet18_imagenet_224_224_3_650_2.0.org.xmodel
(vitis-ai-caffe) Vitis-AI workspace/vvas_model > ./compile_cf_model.sh resnet18_imagenet_224_224_3_650_2.0
*****
* VITIS_AI Compilation - Xilinx Inc.
*****
*****Namespace batch size=1, inputs shape=None, layout="NCHW", model_files=['../models/AI-Model-Zoo/caffe_model/fc_resnet18_imagenet_224_224_3_650_2.0/quantized/deploy.caffemodel'], model_type='caffe', named_inputs_shape=None, out_filename='/tmp/cf_resnet18_imagenet_224_224_3_650_2.0.org.xmodel', proto='../models/AI-Model-Zoo/caffe_model/fc_resnet18_imagenet_224_224_3_650_2.0/quantized/deploy.prototxt'
(INFO) Namespace batch size=1, inputs shape=None, layout="NCHW", model_files=['../models/AI-Model-Zoo/caffe_model/fc_resnet18_imagenet_224_224_3_650_2.0/quantized/deploy.caffemodel'], model_type='caffe', named_inputs_shape=None, out_filename='/tmp/cf_resnet18_imagenet_224_224_3_650_2.0.org.xmodel', proto='../models/AI-Model-Zoo/caffe_model/fc_resnet18_imagenet_224_224_3_650_2.0/quantized/deploy.prototxt'
(INFO) caffe model: /workspace/models/AI-Model-Zoo/caffe_model/fc_resnet18_imagenet_224_224_3_650_2.0/quantized/deploy.caffemodel
(INFO) caffe model: /workspace/models/AI-Model-Zoo/caffe_model/fc_resnet18_imagenet_224_224_3_650_2.0/quantized/deploy.prototxt
(INFO) parse raw model :100% | 82/82 [00:02<00:00, 33.79it/s]
(INFO) infer shape (NHW) :100% | 82/82 [00:00<00:00, 5264.31it/s]
(INFO) infer shape (NHWC) :100% | 82/82 [00:00<00:00, 8811.77it/s]
(INFO) infer shape (NCHW) :100% | 84/84 [00:00<00:00, 9426.16it/s]
(INFO) infer shape (NwC) :100% | 84/84 [00:00<00:00, 9426.16it/s]
(INFO) generate xmodel :100% | 84/84 [00:00<00:00, 805.34it/s]
(INFO) dump xmodel: /tmp/resnet18.org.xmodel
[UNIL00][INFO] Dumped model
[UNIL00][INFO] Debug mode: function
[UNIL00][INFO] Target architecture: DPUCZDX8G_ISA0_B3136_MAX_BG2
[UNIL00][INFO] Generating deploy, with op num: 172
[UNIL00][INFO] Begin to compile...
[UNIL00][INFO] Total device subgraph number 3, DPU subgraph number 1
[UNIL00][INFO] Compile done.
[UNIL00][INFO] The meta json is saved to "/workspace/vvas_model//compiled_output(cf_resnet18_imagenet_224_224_3_650_2.0/meta.json"
[UNIL00][INFO] The compiled xmodel is saved to "/workspace/vvas_model//compiled_output(cf_resnet18_imagenet_224_224_3_650_2.0/resnet18.xmodel"
[UNIL00][INFO] The compiled xmodel's md5sum is 38e0d5cc4f18cd89be318c9c9b01d8ae, and has been saved to "/workspace/vvas_model//compiled_output(cf_resnet18_imagenet_224_224_3_650_2.0/md5sum.txt"
```

สุดท้ายแล้วเอาร์พุตที่ออกแบบรอบไปด้วย 4 ไฟล์ เช่น ไฟล์ md5sum.txt meta.json model.prototxt และ model.xmodel

```
[aic@aic-001:~/AIC-internproject/vvas_project/source/Vitis-AI/vvas_model/compiled_output$ tree yolov2_voc
yolov2_voc
├── md5sum.txt
├── meta.json
└── yolov2_voc.prototxt
    └── yolov2_voc_pruned_0_77.xmodel

0 directories, 4 files
```

3.2.2 การกำหนดค่าของตัว kernel ของ DPU

3.2.2.1 การกำหนดค่า preprocess-config ในรูปแบบของ JSON

ไฟล์ json ที่ใช้ในเฟรมเวิร์ค VVAS นี้ เป็นไฟล์กำหนดค่า (Configuration file)

สำหรับเก็บข้อมูลที่ kernel ต้องการ เพื่อจะทำการ pre-processing บนอินพุตเฟรม (Input frame)

```
1  {
2      "vvas-library-repo": "/usr/lib/",
3      "inference-level": 1,
4      "attach-ppe-outbuf": false,
5      "kernel" : {
6          "library-name": "libvvas_xdpuinfer.so",
7          "config": {
8              "batch-size" : 0,
9              "model-name" : "densebox_320_320",
10             "model-class" : "FACEDTECT",
11             "model-format" : "BGRA",
12             "model-path" : "/usr/share/vitis_ai_library/models/",
13             "run-time-model" : false,
14             "need-preprocess" : false,
15             "performance-test" : false,
16             "debug-level" : 1,
17             "max-objects":3
18         }
19     }
20 }
21 }
```

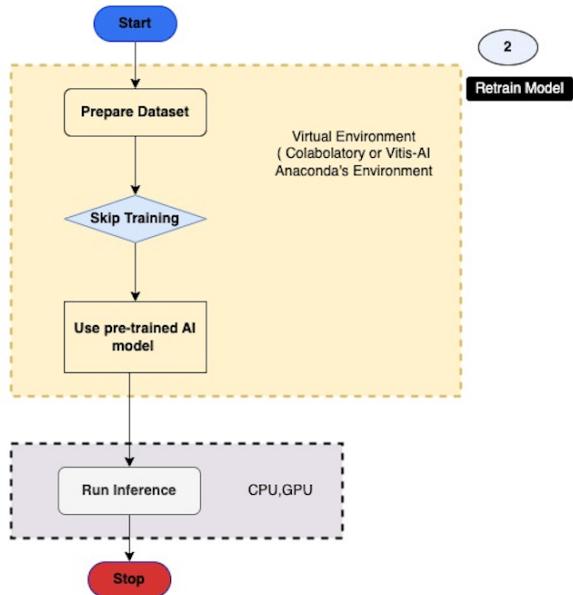
3.2.2.2 การกำหนดค่า kernel-config ในรูปแบบของ json

ไฟล์ json ที่ใช้ในเฟรมเวิร์ค VVAS นี้ เป็นไฟล์กำหนดค่า (Configuration file)

สำหรับเก็บข้อมูลที่ kernel ต้องการ เพื่อจะทำการตีกรอบ (Bounding) โมเดลที่ผ่านการ inference มาแล้ว

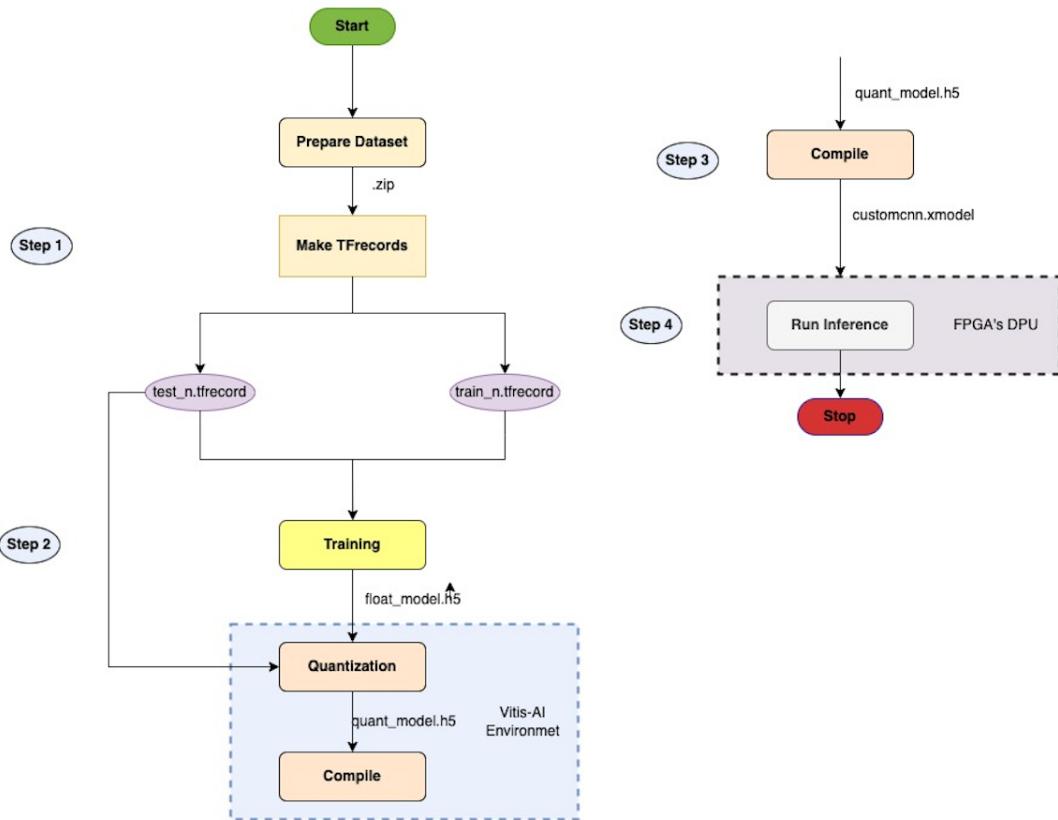
```
1  {
2      "ivav-library-repo": "/usr/lib",
3      "element-mode": "inplace",
4      "kernels" : [
5          {
6              "library-name": "libvvas_xboundingbox.so",
7              "config": {
8                  "model-name" : "densebox_320_320",
9                  "display-output" : 1,
10                 "font-size" : 0.5,
11                 "font" : 3,
12                 "thickness" : 3,
13                 "debug-level" : 0,
14                 "label-color" : { "blue" : 0, "green" : 0, "red" : 0 },
15                 "label-filter" : [ "class", "probability" ],
16                 "classes" : [
17                 ]
18             }
19         }
20     ]
21 }
22 }
```

3.2.3 การ Retrain โมเดล YOLO



การ Retrain โมเดล มีขั้นตอนเหมือนกันกับการ Train โมเดลโดยเริ่มต้นดังแต่การเตรียม dataset การ Quantization และ การ Compilation

3.2.4 การ Train โมเดล Object Classification



การรัน Inference เอไอโมเดล Object Classification บนเฟรมเวิร์กของ Tensorflow2 ประกอบไปด้วย 4 ขั้นตอนด้วยกันดังนี้

- ทำการเตรียมข้อมูล และนำข้อมูลมาสร้างเป็น TFrecords ซึ่งแบ่งออกเป็น test_n.tfrecord และ train_n.tfrecord
- นำข้อมูลมาทำการสอน (Train) ซึ่งได้เป็น Float model หลังจากนั้นนำ Float model ไปทำการ Quantization เพื่อลดขนาด weight จาก floating point ให้เป็น integer
- นำ Quantized โมเดลไปทำการ Compile เพื่อทำการปรับ Arch (Architecture) เอไอโมเดล กับ บอร์ด FPGA ให้ตรงกัน
- ทำการ Inference อี้ไอโมเดลบน FPGA เพื่อทำการทดสอบประสิทธิภาพ (FPS) ของโมเดล

การทดสอบในห้องปฏิบัติการ (I)

1. ทดสอบการการวิเคราะห์วิดีโอ 4 ช่อง

1.1 วัตถุประสงค์

- a. เพื่อทำการ Inference โมเดล AI บนบอร์ด FPGA
- b. ทำการ Stream วิดีโอด้วย 4 ช่อง
- c. ทำการ Compile โมเดล AI จำนวน 8 ตัวโดยทำการ Test Performance เพื่อเอาไปปรับใช้งานจริง
- d. ทำการ Retrain โมเดล AI ชนิด YOLO (You Only Look Once) เพื่อเอาไปใช้ในอุตสาหกรรมความปลอดภัย
- e. นำโมเดลมาทำการทดสอบประสิทธิภาพบน CPU, GPU และ FPGA

1.2 ทำการ Compile model

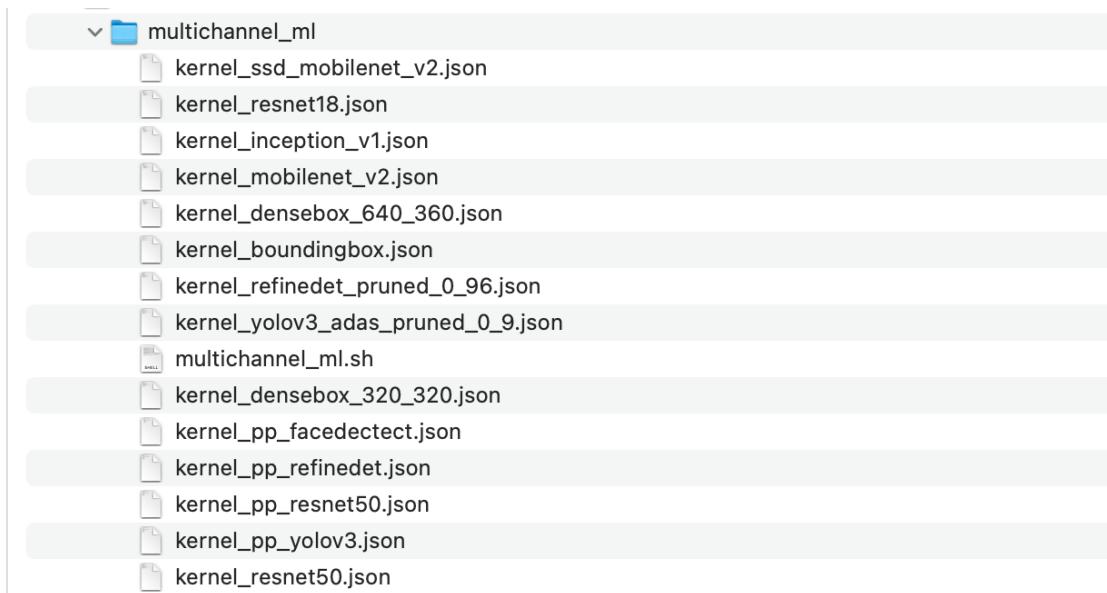
เนื่องจาก Model Zoo เป็น pre-train model ที่ยังไม่สามารถใช้งานบนบอร์ด ZCU104 ได้ ดังนั้นเราต้องทำการปรับโมเดล (Compile model) ที่ร่วงรับการกับแพลตฟอร์ม VVAS เพื่อให้สถาปัตยกรรม (Architecture) ของบอร์ด และโมเดลตรงกัน และเอาไปทดสอบประสิทธิภาพของแต่ละโมเดล

ตารางโมเดลที่ต้องปรับแต่งทดสอบประสิทธิภาพของแต่ละโมเดล

ลำดับที่	ชื่อโมเดล	ทดสอบภาพ	ทดสอบประสิทธิภาพ
1	resnet18	-	-
2	mobilenet_v2	-	-
3	inception_v1	-	-
4	ssd_adas_pruned_0_95	-	-
5	ssd_mobilenet_v2	-	-
6	ssd_pedestrian_pruned_0_97	-	-
7	yolov3_voc_tf	-	-
8	densebox_640_360	-	-

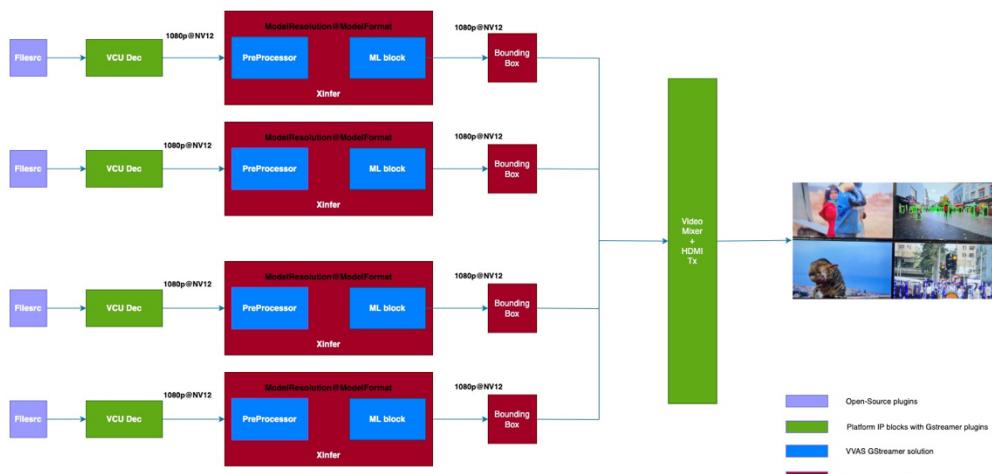
1.3 การทำการ Customize ไฟล์ JSON

ในการวิเคราะห์วิดีโอไฟล์ JSON เป็นหนึ่งในสวนประกอบของการสตรีมมิ่ง (Streaming) วิดีโอบนสมบูรณ์มากโดยเราจะทำการ Custom ไฟล์ JSON เพื่อทำการ Config ตัว kernel เพื่อไปเรียก ตัว Plug-in มาใช้สำหรับการ Inference โมเดล ตีกรอบภาพอุปกรณ์



1.4 การวิเคราะห์วิดีโอแบบหลายช่องทาง

เนื่องจากเราเคยอธิบายตั้งแต่ต้นมาส่วนประกอบหลักในการวิเคราะห์วิดีโอมี 2 อย่างคือตัวโมเดล และไฟล์ JSON นั้นเองโดยทำการวิเคราะห์วิดีโอแบบ Single stage ML และประมวลผล 4 ช่องพร้อมกัน ซึ่งจะได้อาต์พุตออกมาก็จะเป็นการ Stream 4 ช่องพร้อมกัน



Note: แต่ละข้อมูลเข้า (file source) จะเป็นวิดีโอที่ชนิดต่างๆ กันที่ยังไม่มีการประมวลผล AI (ยังไม่มีการตีกรอบ) ได้ แล้วเมื่อหลังจากการประมวลผล AI ผ่านบอร์ด FPGA เรียบร้อยอาต์พุตก็จะเป็นวิดีโอที่มีการตีกรอบ (Inference) และสตรีม (stream) ออกมาร่วมกันผ่านจอมอนิเตอร์

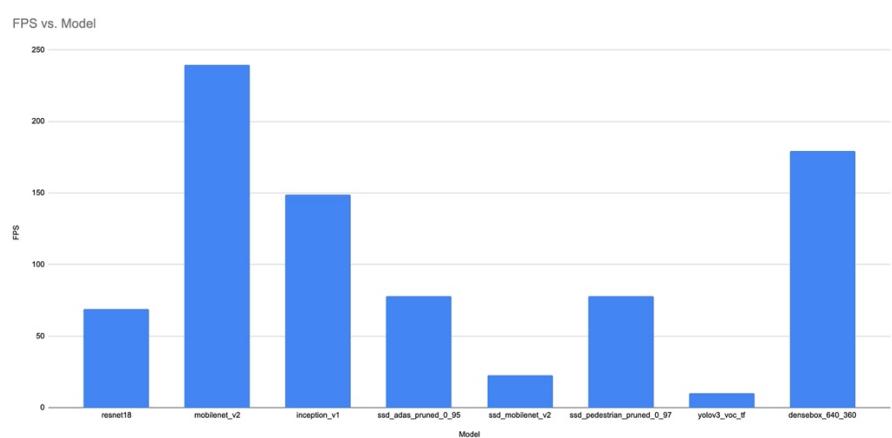
1.5 ทำการทดสอบประสิทธิภาพของโมเดล

ในบทความนี้ เรานำเสนอการวิเคราะห์เปรียบเทียบประสิทธิภาพของเฟรมต่อวินาที (FPS) บนสถาปัตยกรรมฮาร์ดแวร์ต่างๆ รวมถึง CPU, GPU และ FPGA โดยใช้โมเดล AI และการศึกษานี้มีวัตถุประสงค์เพื่อตรวจสอบสถาปัตยกรรมฮาร์ดแวร์เหล่านี้ต่อประสิทธิภาพของ FPS และระบุฮาร์ดแวร์ที่เหมาะสมที่สุดสำหรับแอปพลิเคชันการประมวลผลวิดีโอแบบเรียลไทม์ โดยนำโมเดลที่ทำการ train โดย YOLO มาทดสอบประสิทธิภาพบน CPU กับ GPU และโมเดล Classification ที่ใช้งาน Tensorflow2 มาทดสอบประสิทธิภาพบนตัว FPG

ผลการทดสอบการใช้งานจริง (O)

ทดสอบการการวิเคราะห์วิดีโอ 4 ช่อง

1.1 ทำการ Compile Model



หลังจากนำ Pre-train โมเดลมาทำการทำการปรับโมเดล (Compile model) ที่ร้องรับการกับแพลตฟอร์ม VVNAS ซึ่งเอาต์พุตของโมเดลที่ออกแบบจะอยู่ในรูปแบบของไฟล์ .xmodel หลังจากนั้นทำการนำโมเดลมาทำการทดสอบภาพและประสิทธิภาพบนบอร์ด FPGA (ZCU104) ซึ่งเอาต์พุตที่อย่างได้จะอยู่ในรูปแบบของ FPS (Frame Per Second)

ตารางโน้ตโมเดลที่หลังจากปรับแต่งทดสอบประสิทธิภาพของแต่ละโมเดล

ลำดับที่	ชื่อโมเดล	ทดสอบภาพ	ทดสอบประสิทธิภาพ
1	resnet18	ผ่าน	FPS= 68.8985
2	mobilenet_v2	ผ่าน	FPS= 239.659
3	inception_v1	ผ่าน	FPS= 148.714
4	ssd_adas_pruned_0_95	ผ่าน	FPS= 77.8735
5	ssd_mobilenet_v2	ผ่าน	FPS= 22.9255
6	ssd_pedestrian_pruned_0_97	ผ่าน	FPS= 78.0783
7	yolov3_voc_tf	ผ่าน	FPS= 10.0492
8	densebox_640_360	ผ่าน	FPS= 179.286

1.2 การวิเคราะห์วิดีโอแบบหลายช่องทางบนสถาปัตยกรรม FPGA

ส่วนประกอบของการวิเคราะห์วิดีโอแบบหลายช่องทางแบ่งเป็นตัวโมเดล และไฟล์ JSON โดยที่ตัวโมเดลทำการคำนวณ (Prediction) หรือทำการตัดสินใจ (decisions) และไฟล์ JSON ทำการ Config ตัว kernel เพื่อไปเรียก ตัว Plug-in มาใช้สำหรับการ Inference โมเดล และมีการตีกรอบภาพอุปกรณ์



หลังจากทำการวิเคราะห์เอ็ตพูตที่ออกแบบเป็นวิดีโอแบบเรียลไทม์ 4 ช่องที่มี fps แต่ละช่องเท่ากับ 30 fps ซึ่งอยู่ในรูปแบบการสตรีม (streaming) ช่องพร้อมกัน ยกตัวอย่างโมเดลในภาพนี้มีอย่างเช่นเช่น

- 1) Face Detection (ด้านซ้ายข้างบน)
- 2) Object Detection (ด้านขวาข้างบน)
- 3) Object Classification (ด้านซ้ายข้างล่าง)
- 4) Pedestrian Detection (ด้านขวาข้างล่าง)

1.3 ทำการทดสอบประสิทธิภาพของโมเดล

1.3.1 การสอนโมเดล (Model Training)

ที่นี่เราทำการสอนโมเดล (Training Model) สองโมเดลคือ Yolov5 ใช้เฟรมเวิร์กของ PyTorch กับ Classification โมเดลใช้เฟรมเวิร์ก Tensorflow2 โดยที่:

- i. มีการเตรียมข้อมูล (prepare dataset) ตีกรอบข้อมูล (data labeling) เพื่อนำไป train บนโมเดล Yolov5 (detection) และนำโมเดลมาทดสอบประสิทธิภาพบน CPU กับ GPU
- ii. มีการเตรียมข้อมูล (prepare dataset) เพื่อนำไป train บนโมเดล Classification และนำโมเดลมาปรับแต่ง (Compile) เพื่อทดสอบประสิทธิภาพบนบอร์ด FPGA

Model	Epoch	Accuracy	Loss	Val_Accuracy	Val_Loss
YOLOV5	100	89.8%	0.93%	-	-
Classification	100	94.01%	23.96%	93.9%	24.51%

1.3.2 ทดสอบประสิทธิภาพของโมเดล

ในกระบวนการทดสอบนี้เราจะใช้ 3 สถาปัตยกรรมฮาร์ดแวร์มาราทำ การทดสอบโดยจัด Intel(R) Xeon(R) (Colab), NVIDIA V100 (Colab) และ FPGA เป็น CPU, GPU และ FPGA ตามลำดับ ส่วนผลลัพธ์ที่เราสนใจคือเฟรมต่อวินาที (FPS)

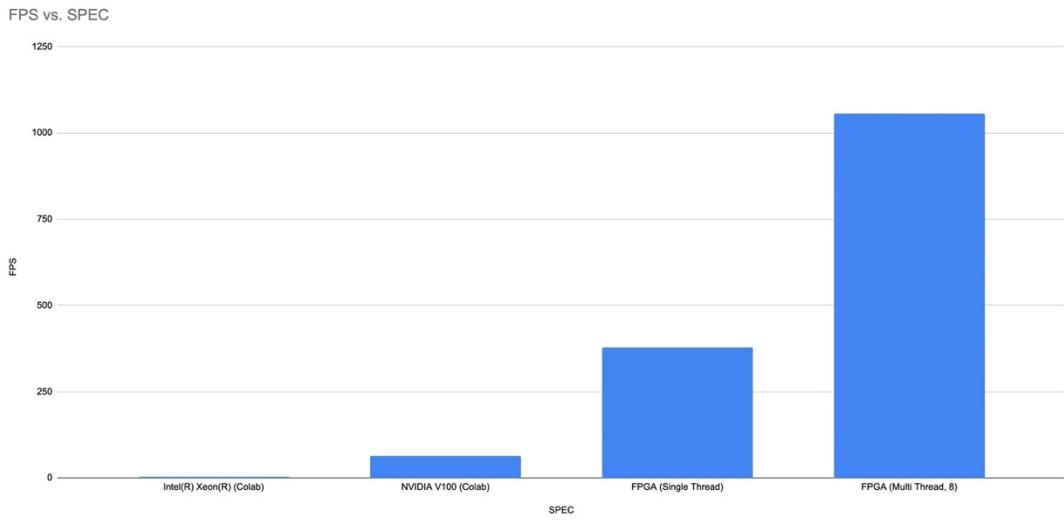
- ผลลัพธ์จากการนำโมเดล Yolov5 (detection) มาทดสอบประสิทธิภาพบน CPU กับ GPU

ลำดับที่	สถาปัตยกรรมฮาร์ดแวร์	FPS(fps)
1	Intel(R) Xeon(R) (Colab)	3.6193
2	NVIDIA V100 (Colab)	64.1026

- ผลลัพธ์จากการนำโมเดล Classification มาทดสอบประสิทธิภาพบน FPGA

ลำดับที่	สถาปัตยกรรมฮาร์ดแวร์	FPS(fps)
1	FPGA (Single Thread)	377.79
2	FPGA (Multi Thread, 8)	1055.84

กราฟสรุปผล:



Note: FPS หรือชื่อเต็มว่า Frame per second คือ ภาพต่อวินาที ซึ่งเป็นหน่วยวัดการจัดเก็บภาพสำหรับกล้องถ่ายภาพดิจิทัล จำนวนการบันทึกภาพของภาพเคลื่อนไหวในกล้องวิดีโอ หรือจำนวนภาพที่แสดงได้ของหน่วยแสดงผลต่าง ๆ ซึ่งหากมีจำนวนค่า fps ที่สูงขึ้นจะทำให้ภาพเคลื่อนไหวที่ได้มีความต่อเนื่อง และสวยงาม ไม่ทำให้ภาพนั้นเกิดการกระตุก เรียกว่าเป็นการเล่นภาพได้อย่างสมูท

ถ้าสมมุติว่าเราอยากประมวลผลภาพเคลื่อนไหวในกล้องวิดีโอที่ใน 1 วินาทีมีจำนวน 200 ภาพ ดังนั้นจากการทดสอบผลทดสอบประสิทธิภาพเราระบุได้ว่า เมื่อใช้สถาปัตยกรรม

1. Intel(R) Xeon(R) (Colab) ที่เป็น CPU ที่มีความสามารถในการประมวลผลได้ 3.6193 fps ซึ่งนั้นหมายความว่าสถาปัตยกรรมนี้สามารถประมวลผลภาพเคลื่อนไหวได้แค่ 3.6193 ภาพ จาก 200 ภาพที่เป็น Input data ซึ่งทำให้ quality ของวิดีโอที่เป็น Output ไม่สมูท เหมือนเดิม
2. NVIDIA V100 (Colab) ที่เป็น GPU ที่มีความสามารถในการประมวลผลได้ 64.1026 fps ซึ่งนั้นหมายความว่าสถาปัตยกรรมนี้สามารถประมวลผลภาพเคลื่อนไหวได้แค่ 64.1026 ภาพ จาก 200 ภาพที่เป็น Input data ซึ่งทำให้ quality ของวิดีโอที่เป็น Output ไม่สมูท เหมือนเดิม (Input data)
3. FPGA (Single Thread) ที่ มี ความสามารถในการประมวลผลได้ 377.79 fps ซึ่งนั้นหมายความว่าสถาปัตยกรรมนี้สามารถประมวลผลภาพเคลื่อนไหวได้ทุกภาพเคลื่อนไหวที่เป็น Input data ซึ่งทำให้ quality ของวิดีโอที่เป็น Output ทำให้ภาพเคลื่อนไหวที่ได้มีความต่อเนื่อง และมีความสมูทเหมือนเดิม (Input data)
4. FPGA (Multi Thread, 8) ที่ มี ความสามารถในการประมวลผลได้ 1055.84 fps ซึ่งนั้นหมายความว่าสถาปัตยกรรมนี้สามารถประมวลผลภาพเคลื่อนไหวได้ทุกภาพเคลื่อนไหวที่เป็น Input data ซึ่งทำให้ quality ของวิดีโอที่เป็น Output ทำให้ภาพเคลื่อนไหวที่ได้มีความต่อเนื่อง และมีความสมูทเหมือนเดิม (Input data)

ส่วนที่ 2 บทสรุปการออกแบบ (ไม่เกิน 1 ย่อหน้า)

บทสรุปการออกแบบ

1. สรุปการออกแบบ

โครงการนี้ถูกออกแบบโดยคำนึงถึงเรื่องการวิเคราะห์วิดีโอแบบหลายช่องทางบนสถาปัตยกรรม FPGA และทำการทดสอบประสิทธิภาพบน CPU, GPU และ FPGA โดยเริ่มต้นตั้งแต่การศึกษาเกี่ยวกับข้อมูล สร้างโมเดล สอนโมเดล (Training model) การทำ Quantizing การทำ Compiling และสุดท้ายคือการปรับใช้โมเดลบนอุปกรณ์ Edge Device (FPGA) นั้นเอง

สรุปการออกแบบนี้มี 3 อย่างคือ

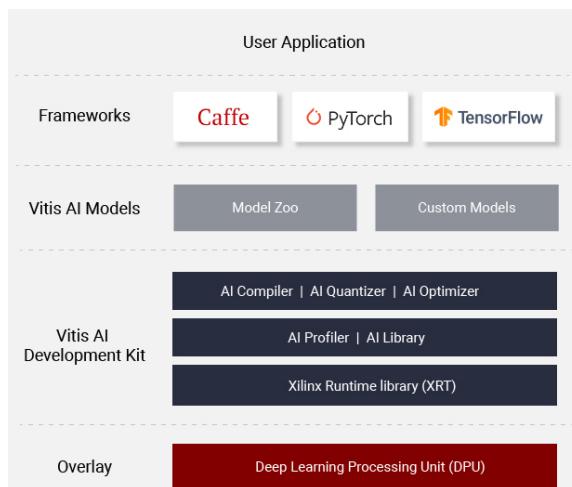
1. การทำการ Compile Model จากโมเดล Zoo ของ Xilinx
2. การวิเคราะห์วิดีโอแบบหลายช่องทางบนสถาปัตยกรรม FPGA
3. ทำการทดสอบประสิทธิภาพของโมเดลบนสถาปัตยกรรมอาร์ดแวร์ (CPU, GPU และ FPGA)

2. การอภิปรายและข้อเสนอแนะ

โครงการที่ถูกออกแบบนี้สามารถสรุปได้ว่าเป็นระบบที่ให้ความสำคัญกับการ Inference เอไอ โมเดลมากกว่าการสอน (Train) โมเดล แต่อย่างไรก็ตามการ inference เปรียบเสมือนการศึกษาในระดับรองต่อจากการสอน (Train) โมเดล เพราะว่าเราไม่สามารถเลือกทำ Inference โดยตรงโดยไม่มีพื้นฐานและความเข้าใจเกี่ยวกับการสอน AI เนื่องจากก่อนที่เราสามารถทำการ Inference นี้ได้เราจะต้องเข้าใจขั้นตอนทั้งหมดของการสอนโมเดลเหมือนกัน เช่น Dataset, สถาปัตยกรรมของโมเดล (Model Architecture), weights, loss, learning rate เป็นต้น

ระบบที่ถูกออกแบบนี้สามารถประมวลผลสตรีมวิดีโอด้วย latency ต่ำ โดยทำการตรวจจับและติดตามวัตถุ เช่น ตรวจจับคนเดินถนน ตรวจจับใบหน้าเป็นต้น โดยมีความเร็วในการประมวลผล 30 เฟรมต่อวินาทีรวมทั้งทำให้ค่า Latency น้อยมาก และนอกจากนี้เรายังสามารถขยายให้มีคุณสมบัติเพิ่มเติม เช่น ปรับใช้กับรถพลังงานไฟฟ้า (EV Car) หรือทางด้านอุตสาหกรรมที่ต้องการความเร็วในการตรวจจับที่ไวมากๆ และยังคงความแม่นยำของการวิเคราะห์เหมือนเดิมด้วยเช่นกัน

3. Low level block (การทำงานภายใน) ของโครงงาน



Vitis AI ประกอบด้วยส่วนหลักดังต่อไปนี้:

- AI Model Zoo: เป็น pre-train model ที่พร้อมสำหรับการใช้งานบนอุปกรณ์ Xilinx
- AI Quantizer: เป็น quantizer ที่ช่วยให้สามารถลดปริมาณแบบจำลอง สถาปัตย์ และ ปรับลงให้เป็น integer (ลดขนาดจาก float ให้เป็น integer)
- AI Compiler: ทำการปรับสถาปัตยกรรมของโมเดลให้ตรงกับสถาปัตยกรรมของอุปกรณ์ (FPGA) ที่นำมาใช้
- AI Profiler: วิเคราะห์ประสิทธิภาพและการใช้จุ่นเพื่อทำการ Inference ของ AI ใน เชิงลึก
- DPU: เป็น Core ของ FPGA ที่ทำงานที่ประมวลผล AI โดยเฉพาะ

ส่วนที่ 3 บรรณานุกรม อ้างอิงตาม APA

- [1] Al-Ali, F. and etc. (2020). Novel Casestudy and Benchmarking of AlexNet for Edge AI: From CPU and GPU to FPGA. **IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)**.
- [2] Ali, M. and etc. (2018). Edge Enhanced Deep Learning System for Large-scale Video Stream
- [3] Arif, A. and etc. (2020). Performance and energy-efficient implementation of a smart city application on FPGAs. **Journal of Real-Time Image Processing**. Vol. 17, pp. 729-743.
- [4] Alias, M.S., Karuppiah, E.K., Kit, C.P. and Tahir, S.M. (2012). Real-time Multiple Video Streams Processing on PC-based FPGA Platform.
- [5] Kim, S., Lee, B., Jeong, J. and Lee, M. (2012). Multi-Object Coprocessor for Multi Channel Embedded DVR System. **IEEE Transactions on Consumer Electronics**. Vol. 58, No. 4, pp. 1366-1374
- [6] Wang, J. and Gu, S. (2021). FPGA Implementation of Object Detection Acceleration Based on Vitis-AI. **International Conference on Information Science and Technology (ICIST)**

()

อาจารย์ที่ปรึกษาโครงงาน

*หมายเหตุ รายงานไม่ควรเกิน 15 หน้า