# รายงานสรุปผลการศึกษา

วิชา 2110292 เอกัตศึกษาทางวิศวกรรมคอมพิวเตอร์ 2

(Individual Study in Computer Engineering II)

เรื่อง

CMV Virus Infection Detection


จัดทำโดย

นายพีรณัฐ ธีระวัฒนชัย

รหัสประจำตัวนิสิต 6231343821


เสนอ

อาจารย์ เอกพล ช่วงสุวนิช

รายงานฉบับนี้เป็นส่วนหนึ่งของการศึกษารายวิชาเอกัตศึกษาทางวิศวกรรมคอมพิวเตอร์ 2

จุฬาลงกรณ์มหาวิทยาลัย

ภาคเรียนที่ 2 ปีการศึกษา 2563

# คำนำ

รายงานฉบับนี้เป็นส่วนหนึ่งของการศึกษารายวิชาเอกัตศึกษาทางวิศวกรรมคอมพิวเตอร์ 2 (Individual Study in Computer Engineering II) จัดทำขึ้นเพื่อสรุปสิ่งที่ได้เรียนรู้จากวิชานี้ โดยได้ร่วมทำโปรเจคเรื่อง Machine Learning เรื่อง Computer Vision เกี่ยวกับการทำนาย CMV Virus จากรูปภาพ WSI ด้วยเครื่องมือ YOLOv5 โดยในรายงานเล่มนี้ประกอบด้วยเนื้อหาดังนี้

1. ความเป็นมาและวัตถุประสงค์
2. ขั้นตอนการทำงาน สิ่งที่ทำ
3. ผลลัพธ์ของงาน
4. อุปสรรคที่เกิดขึ้น วิธีแก้ปัญหา และสิ่งที่ได้รับจาก Individual Study นี้

ทั้งนี้ข้าพเจ้าหวังเป็นอย่างยิ่งว่ารายงานเล่มนี้จะเป็นประโยชน์ต่อผู้ที่ได้มาศึกษาเป็นอย่างดี และหากมีข้อผิดพลาดประการใดขออภัยมา ณ โอกาสนี้ด้วย

นายพีรณัฐ ธีระวัฒนชัย

ผู้จัดทำ

# สารบัญ

## 1. ความเป็นมาและวัตถุประสงค์

- ความเป็นมา

ในปัจจุบันการตรวจหาเชื้อไวรัสบางชนิดก็ยังคงใช้เวลานานด้วยวิธีการส่องกล้องจุลทรรศน์ไปที่เซลล์และใช้งบประมาณพอสมควร แต่ในบางครั้งเราก็สามารถใช้เทคโนโลยีมาประยุกต์ใช้เพื่อทำนายว่าเซลล์เหล่านี้มีไวรัสอยู่หรือไม่ เพื่อนำไปใช้วิเคราะห์ทางการแพทย์ต่อไป

โดยในงานนี้ผมได้รับมอบหมายให้นำข้อมูลภาพเซลล์ WSI ขนาดใหญ่ มาใช้กระบวนการ Machine Learning ในเรื่อง Computer Vision เพื่อทำนาย CMV Virus Infection จากภาพถ่าย WSI อื่นที่ไม่ได้นำมา Train และ เป้าหมายหลัก คือ ต้องการให้ Model ที่ Train ออกมามีความแม่นยำ (ค่า mAP) มากที่สุด โดยในระหว่างการทำได้พบว่าภาพที่มีการ Normalized ก่อน train จะได้ความแม่นยำในการทำนายมากกว่าภาพที่ไม่มีการ Normalized เมื่อใช้ภาพที่ test ในปริมาณมาก (ในรายงานนี้จะได้ความแม่นยำเท่ากัน เนื่องจากใช้รูปภาพทดสอบในปริมาณน้อย)

- วัตถุประสงค์

1. เพื่อทำนาย CMV Virus Infection จากภาพถ่าย WSI

2. เพื่อศึกษาการทำ Machine Learning เรื่อง Computer Vision และการใช้ YOLOv5 ใน Google Colaboratory

3. เพื่อศึกษาการเขียนโปรแกรมภาษา Python ใน Google Colaboratory

## 2. ขั้นตอนการทำงาน สิ่งที่ทำ

- ขั้นตอนการทำงาน

1. ศึกษาการทำ Machine Learning เรื่อง Computer Vision การใช้ YOLOv3 และ การใช้ YOLOv5 จาก https://www.youtube.com/watch?v=2TH1b-aP3wo&list=PLDsYDkzmWTEi0fmC3UYCM5VZTxDVXWyIN&index=7&ab_channel=MLRS และ https://michaelohanu.medium.com/yolov5-tutorial-75207a19a3aa

2. วางแผน และ แนวทางในการทำงาน

3. เริ่มทำงานใน Google Colaboratory และรายงานความคืบหน้ากับอาจารย์ทุกสัปดาห์จนเสร็จ

4. ตรวจสอบความถูกต้องของงาน

5. ทำรูปเล่มสรุปการทำงาน

- สิ่งที่ทำ

1. Google Colaboratory : ภายในเป็นการเขียนโปรแกรมด้วย Python เพื่อตอบสนองวัตถุประสงค์ที่จะทำนาย CMV Virus Infection จากภาพถ่าย WSI

2. รูปเล่มสรุปการทำงาน : ภายในเป็นการสรุปผลงานที่ทำ อุปสรรคที่เกิดขึ้นระหว่างการทำงาน แนวทางแก้ไข และ สิ่งที่ได้จากการศึกษาครั้งนี้

## 3. ผลลัพธ์ของงาน

3.1 Preprocessing Data

- ใช้ SQL นำตำแหน่งในภาพที่เป็น CMV Virus ออกมาจาก File CU_DB.sqlite
- เก็บข้อมูลในรูปแบบชื่อรูปกับ Bounding Box เพื่อนำไปใช้สร้าง Annotation

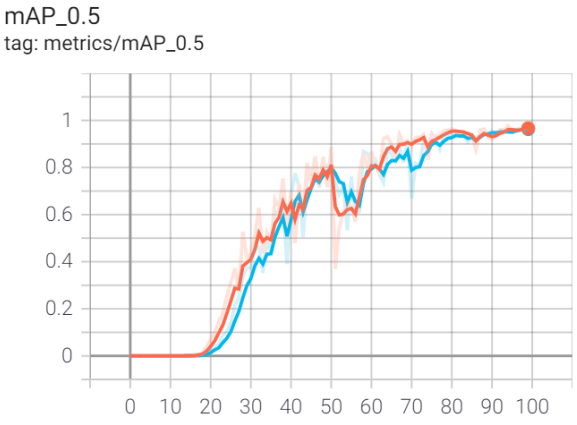
3.2 Create Annotation

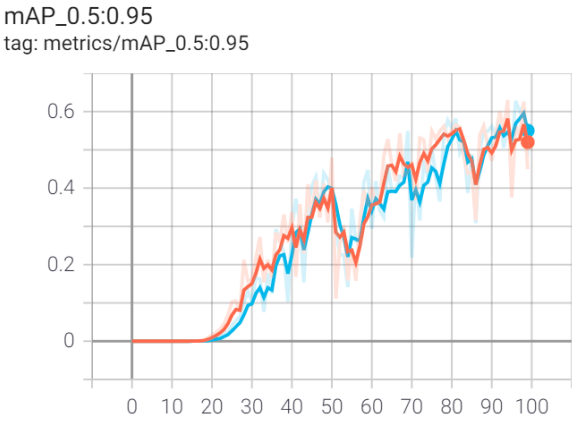
- สร้าง Annotation เป็น File .xml โดยมีการเพิ่ม noise ใน Bounding Box แบบสุ่ม และ Slide รูปในตำแหน่งที่ได้ไปเก็บไว้ใน images/train และสร้าง
- Normalize train image ด้วยการใช้ Eiganvalue and Eigenvector เพราะภาพที่ต้องใช้ test ส่วนใหญ่มีสีที่แตกต่างออกไปจากภาพที่ train

3.3 Train Data by YOLOv5

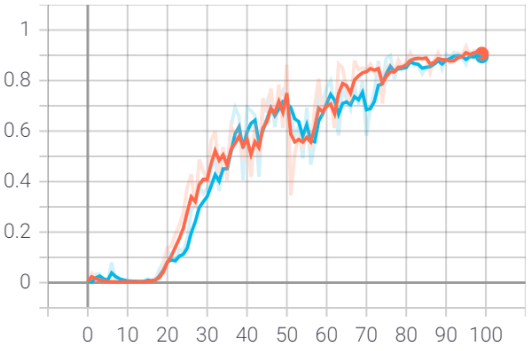
- File .txt จาก .xml ด้วยการ convert Annotation
- Train จาก image กับ normalized image ด้วย 100 epochs ได้ผลดังนี้

หมายเหตุ : สีส้มคือการ train ด้วยรูปปกติ ส่วนสีฟ้าคือการ train ด้วย normalized image

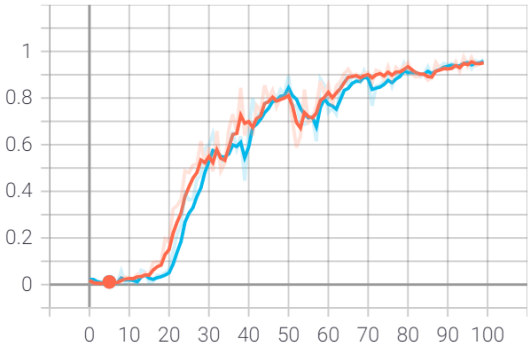


precision
tag: metrics/precision

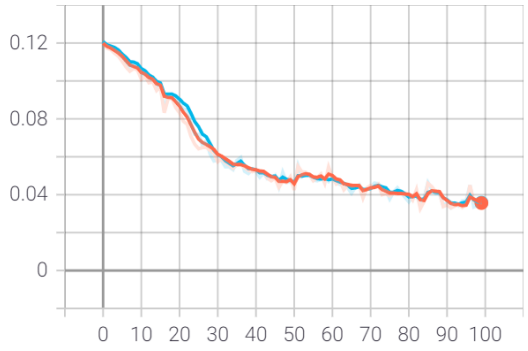



recall
tag: metrics/recall

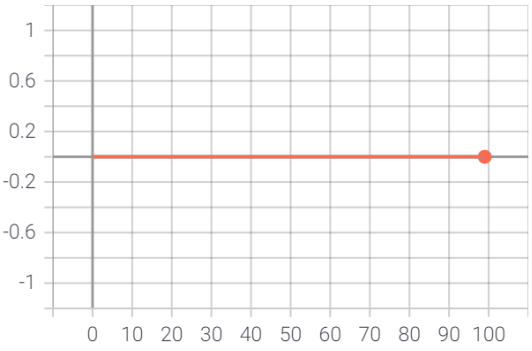



train

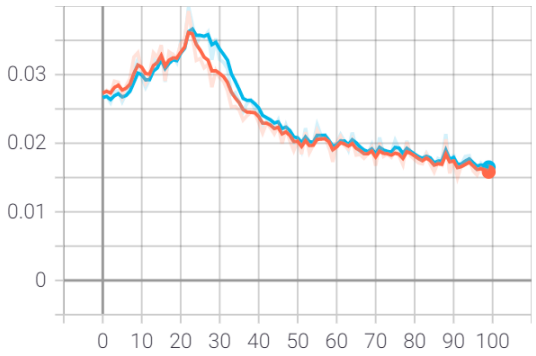
box_loss
tag: train/box_loss




cls_loss
tag: train/cls_loss




obj_loss
tag: train/obj_loss



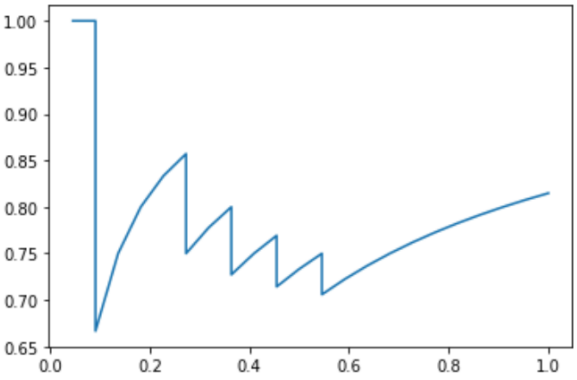

3.4 Cut image

- ตัดรูปภาพที่ต้องใช้ test ออกเป็นภาพย่อยๆ ขนาด 512x512 pixels โดยขยับ scale ทีละ 256 pixels เก็บใน folder ย่อยๆ ตามตำแหน่งในแกน y

3.5 Evaluation

- Test image ด้วย Train data ที่มาจาก Normalized image ได้ค่า mAP = 0.8393 (83.93%)

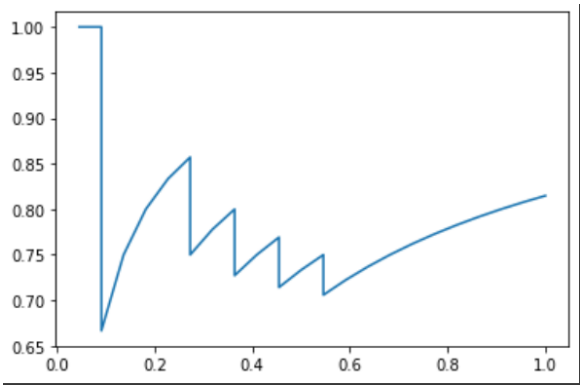
| | xmin | ymin | xmax | ymax | conf | TP | Precision | Recall | IP |
|---|---|---|---|---|---|---|---|---|---|
| 11 | 57554.0 | 13242.0 | 57647.0 | 13343.0 | 0.197266 | True | 1.000000 | 0.045455 | 1.000000 |
| 1 | 65270.0 | 13692.0 | 65385.0 | 13784.0 | 0.179688 | True | 1.000000 | 0.090909 | 1.000000 |
| 3 | 65270.0 | 13692.0 | 65385.0 | 13784.0 | 0.179688 | False | 0.666667 | 0.090909 | 1.000000 |
| 4 | 65281.0 | 15539.0 | 65344.0 | 15624.0 | 0.166016 | True | 0.750000 | 0.136364 | 0.750000 |
| 5 | 116272.0 | 29195.0 | 116349.0 | 29278.0 | 0.162109 | True | 0.800000 | 0.181818 | 0.800000 |
| 0 | 151583.0 | 17202.0 | 151634.0 | 17284.0 | 0.160156 | True | 0.833333 | 0.227273 | 0.833333 |
| 2 | 146173.0 | 4997.0 | 146262.0 | 5075.0 | 0.152344 | True | 0.857143 | 0.272727 | 0.857143 |
| 3 | 146173.0 | 4997.0 | 146262.0 | 5075.0 | 0.152344 | False | 0.750000 | 0.272727 | 0.857143 |
| 6 | 116538.0 | 30653.0 | 116603.0 | 30726.0 | 0.142578 | True | 0.777778 | 0.318182 | 0.777778 |
| 8 | 108147.0 | 16964.0 | 108216.0 | 17035.0 | 0.138672 | True | 0.800000 | 0.363636 | 0.800000 |
| 2 | 65291.0 | 13783.0 | 65358.0 | 13853.0 | 0.136719 | False | 0.727273 | 0.363636 | 0.800000 |
| 0 | 65291.0 | 13783.0 | 65358.0 | 13853.0 | 0.136719 | True | 0.750000 | 0.409091 | 0.750000 |
| 4 | 146676.0 | 5243.0 | 146741.0 | 5313.0 | 0.136719 | True | 0.769231 | 0.454545 | 0.769231 |
| 6 | 146676.0 | 5243.0 | 146741.0 | 5313.0 | 0.136719 | False | 0.714286 | 0.454545 | 0.769231 |
| 9 | 98946.0 | 5958.0 | 99041.0 | 6027.0 | 0.134766 | True | 0.733333 | 0.500000 | 0.733333 |
| 5 | 146737.0 | 5271.0 | 146800.0 | 5334.0 | 0.123047 | True | 0.750000 | 0.545455 | 0.750000 |
| 7 | 146737.0 | 5271.0 | 146800.0 | 5334.0 | 0.123047 | False | 0.705882 | 0.545455 | 0.750000 |
| 7 | 108698.0 | 17218.0 | 108758.0 | 17281.0 | 0.123047 | True | 0.722222 | 0.590909 | 0.722222 |
| 1 | 146059.0 | 4810.0 | 146130.0 | 4870.0 | 0.117188 | True | 0.736842 | 0.636364 | 0.736842 |
| 9 | 104504.0 | 22767.0 | 104579.0 | 22827.0 | 0.117188 | True | 0.750000 | 0.681818 | 0.750000 |
| 8 | 142611.0 | 11030.0 | 142677.0 | 11089.0 | 0.115234 | True | 0.761905 | 0.727273 | 0.761905 |




กราฟระหว่างค่า Precision กับ Recall

- Test image ด้วย Train data ที่มาจาก Image ปกติ ได้ค่า mAP = 0.8393 (83.93%)

| | xmin | ymin | xmax | ymax | conf | TP | Precision | Recall | IP |
|---|---|---|---|---|---|---|---|---|---|
| 11 | 57554.0 | 13242.0 | 57647.0 | 13343.0 | 0.197266 | True | 1.000000 | 0.045455 | 1.000000 |
| 1 | 65270.0 | 13692.0 | 65385.0 | 13784.0 | 0.179688 | True | 1.000000 | 0.090909 | 1.000000 |
| 3 | 65270.0 | 13692.0 | 65385.0 | 13784.0 | 0.179688 | False | 0.666667 | 0.090909 | 1.000000 |
| 4 | 65281.0 | 15539.0 | 65344.0 | 15624.0 | 0.166016 | True | 0.750000 | 0.136364 | 0.750000 |
| 5 | 116272.0 | 29195.0 | 116349.0 | 29278.0 | 0.162109 | True | 0.800000 | 0.181818 | 0.800000 |
| 0 | 151583.0 | 17202.0 | 151634.0 | 17284.0 | 0.160156 | True | 0.833333 | 0.227273 | 0.833333 |
| 2 | 146173.0 | 4997.0 | 146262.0 | 5075.0 | 0.152344 | True | 0.857143 | 0.272727 | 0.857143 |
| 3 | 146173.0 | 4997.0 | 146262.0 | 5075.0 | 0.152344 | False | 0.750000 | 0.272727 | 0.857143 |
| 6 | 116538.0 | 30653.0 | 116603.0 | 30726.0 | 0.142578 | True | 0.777778 | 0.318182 | 0.777778 |
| 8 | 108147.0 | 16964.0 | 108216.0 | 17035.0 | 0.138672 | True | 0.800000 | 0.363636 | 0.800000 |
| 2 | 65291.0 | 13783.0 | 65358.0 | 13853.0 | 0.136719 | False | 0.727273 | 0.363636 | 0.800000 |
| 0 | 65291.0 | 13783.0 | 65358.0 | 13853.0 | 0.136719 | True | 0.750000 | 0.409091 | 0.750000 |
| 4 | 146676.0 | 5243.0 | 146741.0 | 5313.0 | 0.136719 | True | 0.769231 | 0.454545 | 0.769231 |
| 6 | 146676.0 | 5243.0 | 146741.0 | 5313.0 | 0.136719 | False | 0.714286 | 0.454545 | 0.769231 |
| 9 | 98946.0 | 5958.0 | 99041.0 | 6027.0 | 0.134766 | True | 0.733333 | 0.500000 | 0.733333 |
| 5 | 146737.0 | 5271.0 | 146800.0 | 5334.0 | 0.123047 | True | 0.750000 | 0.545455 | 0.750000 |
| 7 | 146737.0 | 5271.0 | 146800.0 | 5334.0 | 0.123047 | False | 0.705882 | 0.545455 | 0.750000 |
| 7 | 108698.0 | 17218.0 | 108758.0 | 17281.0 | 0.123047 | True | 0.722222 | 0.590909 | 0.722222 |
| 1 | 146059.0 | 4810.0 | 146130.0 | 4870.0 | 0.117188 | True | 0.736842 | 0.636364 | 0.736842 |




กราฟระหว่างค่า Precision กับ Recall

## 4. อุปสรรคที่เกิดขึ้น วิธีแก้ปัญหา และสิ่งที่ได้รับจาก Individual Study นี้

- อุปสรรคที่เกิดขึ้น และ วิธีแก้ปัญหา

1. **อุปสรรค** : ไม่ทราบโครงสร้างของโค้ดบางส่วนที่อยู่นอกเหนือจากการไปศึกษามา และจำเป็นต้องใช้ในงานนี้

**วิธีแก้** : ไปค้นหาเพิ่มเติมในอินเตอร์เน็ต หรือ ถามเพื่อนที่ทำงานด้วยกัน

2. **อุปสรรค :** ภาพ WSI ที่ตัดแล้วมีจำนวนมากเกินไปจน Google Drive ไม่สามารถอ่านได้ทั้งหมด

**วิธีแก้** : แบ่งรูปภาพออกเป็น Folder ย่อยตามชื่อภาพหรือลำดับแถวที่ตัด

3. **อุปสรรค :** ภาพ WSI ที่ Train มีสีที่ต่างจากภาพ WSI ที่ Test พอสมควร

**วิธีแก้** : ปรึกษาอาจารย์ และ ใช้การ Normalize ภาพด้วย SVD เพื่อปรับสีภาพ Train และ Test

4. **อุปสรรค :** เวลากด Test Data บางครั้งขึ้นว่า Google Drive Overload

**วิธีแก้** : ลองกด Run ซ้ำ 2-3 ครั้ง ปรากฏว่าสามารถทำงานได้ตามปกติ

5. **อุปสรรค :** เวลาตัดภาพที่มีใน Google Drive แล้วขึ้นว่า Google Drive Quota Limit

**วิธีแก้** : ค่อยตัดรูปต่อในวันถัดไป และ ไม่เก็บรูปที่ตัดใน Folder เดียวกันมากเกินไป

- สิ่งที่ได้รับจาก Individual Study นี้

1. ได้สรุปผลว่าด้วยการ Train Data จากข้อมูลที่มีแล้วนำมาทดสอบได้ผลว่าสามารถทำนาย CMV Virus Infection ได้ความแม่นยำ 83.93% ถ้ายังไม่ Normalize และได้ 83.93% เมื่อทำการ Normalize ภาพก่อนนำไป Train และ Test

2. ได้เรียนรู้เกี่ยวกับการทำ Machine Learning ในเรื่อง Computer Vision

3. ได้เรียนรู้เกี่ยวกับการใช้ YOLOv5 มาช่วยในการทำ Object Detection

4. ได้ทบทวนการเขียนโปรแกรมด้วย Python และ การใช้ Google Colaboratory

---

## Reference


- https://www.youtube.com/watch?v=2TH1b-aP3wo&list=PLDsYDkzmWTEi0fmC3UYCM5VZTxDVXWyIN&index=7&ab_channel=MLRS
- https://michaelohanu.medium.com/yolov5-tutorial-75207a19a3aa
- https://blog.roboflow.com/how-to-train-yolov5-on-a-custom-dataset/
- https://blog.paperspace.com/how-to-implement-a-yolo-object-detector-in-pytorch/


---