Individual take-home midterm exam report

Traffy Fondue Image Classification

2110446 Data Science and Data Engineering

จัดทำโดย อคิราภ์ สิทธิการิยะวัตร 6330563421

Introduction

จากโจทย์ปัญหา Traffy Fondue เป็น platform รายงานปัญหาซึ่งมีการแนบรูปถ่ายที่เกี่ยวกับรายงานนั้นๆ โดยที่จากรูปถ่ายนั้นเองได้มีการ classify แบ่งปัญหาย่อยต่างๆทั้งหมดออกเป็น 10 class แล้ว ได้แก่ canal, electric, flooding, light, road, sanitary, sewer, sidewalk, stray และ traffic แต่ทว่าในการ classify รูปออกเป็น class ดังกล่าวยัง classify ทั้งถูกและผิด class สลับกันไปมา จึงต้องมีการทำ model เพื่อทำการ train ให้ classify รูปต่างๆเหล่านั้นไปสู่หมวดหมู่ปัญหาที่ถูกต้องได้ โดยที่จากโจทย์ดังกล่าวเป็นปัญหาประเภท single label image classification (1 รูปเป็นได้เพียง 1 class) ซึ่งตั้งเป้าหมายในการทำ model ให้สามารถทำการ predict class ต่างๆให้ได้เกิน0.80 macro F1 score สำหรับ private test data

Data Preparation

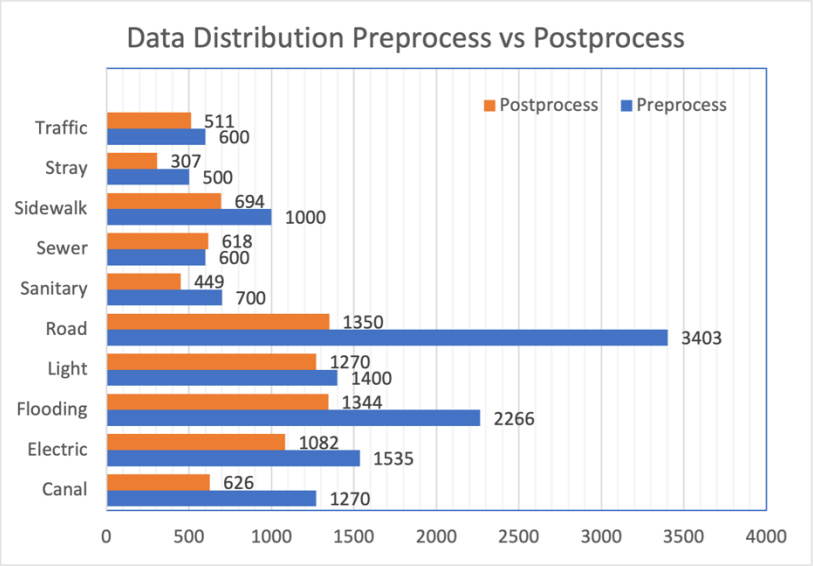
จาก raw data ทั้งหมดที่มี แบ่งออกเป็น 13274 รูปสำหรับการ train data (มี label กำกับ) และ 4648 รูปสำหรับ test data (สำหรับการทำ prediction เพื่อวัดประสิทธิภาพ macro F1 score ของ model ที่ทำ) ซึ่งในเบื้องต้น train data ที่ประกอบด้วย 10 class ย่อยที่ถูกแบ่งมาให้เรียบร้อย ประกอบด้วย canal 1270 รูป, electric 1535 รูป, flooding 2266 รูป, light 1400 รูป, road 3403 รูป, sanitary 700 รูป, sewer 600 รูป, sidewalk 1000 รูป, stray 500 รูป และ traffic 600 รูป

ในขั้นตอนแรก หลังจากการไล่ดูรูปในแต่ละ class ของ training data พบว่า ยังมีรูปจำนวนมากที่ซ้ำหรือถูก classify ไปผิด class รวมถึงยังผู้จัดทำเองยังมีความสับสนในการตีความรูป เช่น รูปทางเดินริมคลองจะนับเป็น canal หรือ sidewalk หรือรูปที่มีทั้งปัญหาของ sanitary และ sewer ในรูปเดียว ผู้จัดทำจึงต้องทำการนิยามกฎในการคัดเลือกรูปใหม่ เพื่อเป็นมาตรฐานที่จะใช้กับการแบ่ง class รูปใหม่ในขั้นตอนถัดๆไป


(ซ้าย) ปัญหาทางเดินริมคลอง (ขวา) ปัญหา sanitary และ sewer ในรูปเดียว

หลังจากการนิยามและตั้งกฎในการคัดรูปเสร็จเรียบร้อยแล้ว โดยที่ผู้จัดทำได้กำหนดในเบื้องต้นว่าสำหรับทุกรูปที่เป็นได้มากกว่า 2 class ขึ้นไป โดยที่เราไม่รู้ความต้องการของผู้รายงานปัญหาจริงๆว่าต้องการรายงานเรื่องใด (เช่นมีการวงกลมสีแดงบนบริเวณที่ผู้รายงานต้องการเน้นย้ำ) หรือรูปที่ไม่มีปัญหาใดๆเกี่ยวข้อง จะทำการคัดทิ้งทันที ส่วนในบาง class จะมีนิยามเพิ่มเติม ได้แก่ canal จะรวมถึงทางเดินริมคลอง, sanitary จะรวมกับปัญหาขยะในคลองหรือท่อระบายน้ำ รวมถึงการเผาขยะ และ sewer จะรวมถึงท่อสายโทรศัพท์บนทางเท้าด้วย ซึ่งทั้งหมดนี้ หลังจากนั้นจึงเริ่มทำการคัดภาพ โดยเริ่มจากการคัดรูปซ้ำแบบ exact duplicate ออกในเบื้องต้น โดยอาศัยการทำ phash กับรูปเข้ามาช่วย หลังจากนั้นจึงเริ่มคัดมือทั้งหมดในการ reassign ไปยัง class ที่ถูกต้อง และการคัดรูป near duplicate ออก ซึ่งในการคัดภาพทิ้งจะไม่ทำการลบออก แต่จะย้ายไปอยู่ใน folder อีกอันหนึ่งแทน เพื่อเอาไว้ป้องกันและตรวจทานในตอนท้ายอีกทีหนึ่ง

ภายหลังจากการคัดรูปด้วยมือ และตรวจทานทั้งหมดแล้ว พบว่า ได้ทำการคัดรูปเพื่อใช้สำหรับการเป็น train data เหลือทั้งหมด 8251 รูป แบ่งออกเป็น canal 626 รูป, electric 1082 รูป, flooding 1344 รูป, light 1270 รูป, road 1350 รูป, sanitary 449 รูป, sewer 618 รูป, sidewalk 694 รูป, stray 307 รูป และ traffic 511 รูป ซึ่งมี data distribution ตามแผนภาพด้านล่างดังต่อไปนี้



	Class	Train	Test	Validation	Sum
0	canal	501	63	62	626
1	electric	865	108	109	1082
2	flooding	1075	134	135	1344
3	light	1016	127	127	1270
4	road	1080	135	135	1350
5	sanitary	359	45	45	449
6	sewer	494	62	62	618
7	sidewalk	555	69	70	694
8	stray	246	31	30	307
9	traffic	409	51	51	511

(ซ้าย) Data distribution Preprocess (ก่อน clean) vs Postprocess (หลัง clean)

(ขวา) Class distribution ของ train, test และ validate

หลังจากการทำการ clean train data เสร็จเรียบร้อยแล้ว จึงทำการเตรียม data สำหรับการ train model โดยที่ผู้จัดทำได้แบ่ง train data ออกเป็นส่วนของ train 80%, test 10% และ validate 10% โดยในการแบ่งกำหนด seed ใน train_test_split เป็น 42 และมีการทำ stratify เพื่อให้สัดส่วน data distribution ของแต่ละ class ใน train, test, validate มีอัตราส่วนตามที่แบ่ง train, test, validate เป็น 80 : 10 : 10

ในขั้นตอนท้ายสุดของการทำ Data Preparation ผู้จัดทำได้การ resize รูปทุกรูปของทั้ง train, test, validate ให้เป็นขนาด 224x224 pixel หลังจากนั้นจึงทำการแปลงเป็น Tensor แล้ว Normalize tensor ด้วย mean และ sd เพื่อแปลงข้อมูลเป็น Z-score โดยที่ในเฉพาะส่วนของ train ผู้จัดทำได้มีการทำ Data Augmentation ด้วยการใส่ RandomHorizontalFlip และ RandomRotate (30 องศา) ก่อนแปลงเป็น Tensor เพื่อเป็นการลดโอกาสในการเกิดการ overfit ของข้อมูล และช่วยให้ sample มีหลากหลายมากขึ้น หลังจากนั้นในขั้นตอนท้ายสุดจึง Load ข้อมูลเข้า DataLoader ของ PyTorch ซึ่งได้ทำการ map รูปแต่ละรูปกับ label (class) ของรูปนั้นๆไว้เรียบร้อยแล้ว เพื่อเตรียมนำเข้าไป train ใน model ในขั้นถัดไป

Model

สำหรับในขั้นตอนของการ Train model ผู้จัดทำได้เลือกใช้ PyTorch เป็น Library ช่วยสร้าง Network โดยที่ใช้ Model EfficientNet-B3 แบบที่มีการ pretrain มาก่อนแล้ว โดยทำการแปลงให้ใน layer สุดท้ายของ network ให้มีการเรียก Linear และ Softmax 10 เพื่อให้ทำการ predict output class ออกมาเป็น 0 หรือ 1 ในแต่ละ class เท่านั้น ให้ตรงกับที่โจทย์ต้องการ โดยสาเหตุที่เลือกใช้EfficientNet-B3 คือ EfficientNet เองมีการทำ Compound Scaling คือการเพิ่มทั้ง width, depth, resolution ด้วย constant ratio ตัวเดียวกัน ทำให้ network สามารถ extract feature ได้แม่นยำขึ้น มีประสิทธิภาพที่ดีขึ้นเมื่อเทียบกับ model ตัวเก่าๆ อย่างเช่น ResNet-34 ซึ่งยังมีขนาดจำนวน parameter ที่มากกว่า EfficientNet-B3 (และในที่นี้ ผู้จัดทำใช้เพียงแค่ EfficientNet-B3 เนื่องจาก tradeoff ระหว่างประสิทธิภาพกับทรัยากรในการ run ยังคุ้มค่า)

ในการ set up ก่อนการ train ผู้จัดทำได้ใช้ cross-entropy loss function และได้ทำการใส่ class weight ลงไปเพิ่มด้วย เพื่อทำการ balance ค่าน้ำหนักของแต่ละ class ให้เป็นการลด bias ใน model เนื่องจากใน train data ที่มีจำนวนข้อมูลแต่ละ class ที่ไม่เท่ากัน class ที่มีสัดส่วนที่น้อยกว่าจึงควรมีค่าน้ำหนักในการทำนายแต่ละครั้งมากกว่า class ที่มีสัดส่วนที่เยอะกว่า โดยมีการใช้ optimizer แบบ SGD ด้วย learning rate = 0.02 และ momentum = 0.9 และใช้ scheduler เป็นแบบ StepLR ด้วย step = 7 และ gamma = 0.5 ทำการ train ทั้งหมด 20 epoch

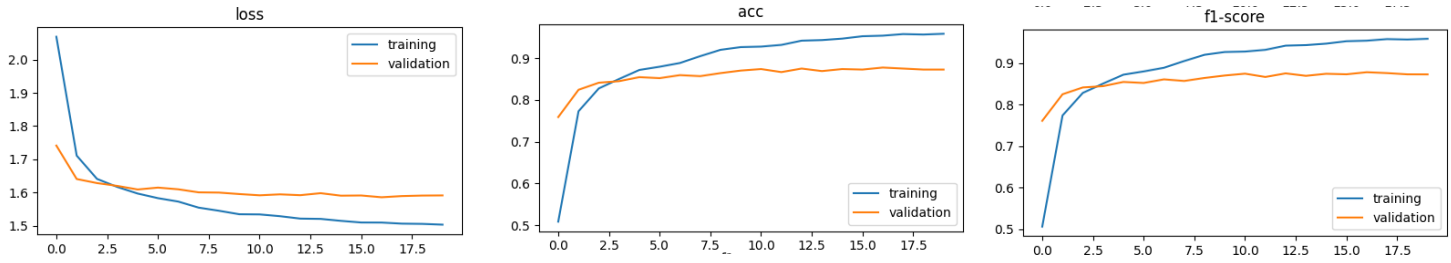
```
criterion = nn.CrossEntropyLoss(weight=class_weights,reduction='mean').to(device)
optimizer = optim.SGD(net.parameters(), lr=0.02, momentum=0.9)
scheduler = lr_scheduler.StepLR(optimizer, step_size=7, gamma=0.5)
```

การ setup loss function, optimizer และ scheduler

ในแต่ละ epoch จะมีการ save model ไว้ทุกๆครั้งที่ loss ของ validate set (validation loss) มีค่าต่ำกว่าทุก epoch ที่เคยผ่านมา ทั้งนี้เพื่อเป็นการลดการเกิด overfit ของ model หลังจากที่ run เสร็จสิ้นแล้วจึง load model path ที่ save ไว้เพื่อใช้เป็น model สำหรับการทำ prediction ในส่วนของ test ที่แบ่งมาจาก training data และตัว test data จริงๆที่ต้องทำ prediction เพื่อส่งผลไปตรวจคำตอบกับ Kaggle

Results

จากการ train model ด้วย train set และมี validate set ในการช่วยตรวจ ได้ทำการ plot graph ของ loss, accuracy และ F1 score ได้ผลลัพธ์ออกมาดังนี้



(ซ้าย) loss graph (กลาง) accuracy graph (ขวา) F1-score graph

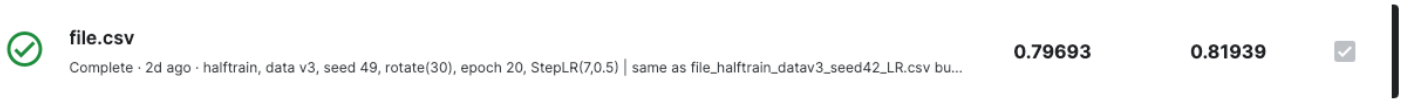
เมื่อนำ model ไป predict กับ test set ที่แบ่งออกมาจาก training data ได้ผลลัพธ์ดังที่แสดงใน Confusion Matrix ด้านล่างดังนี้ ซึ่งได้คะแนน Macro F1-Score = 0.8632

testing loss: 1.592

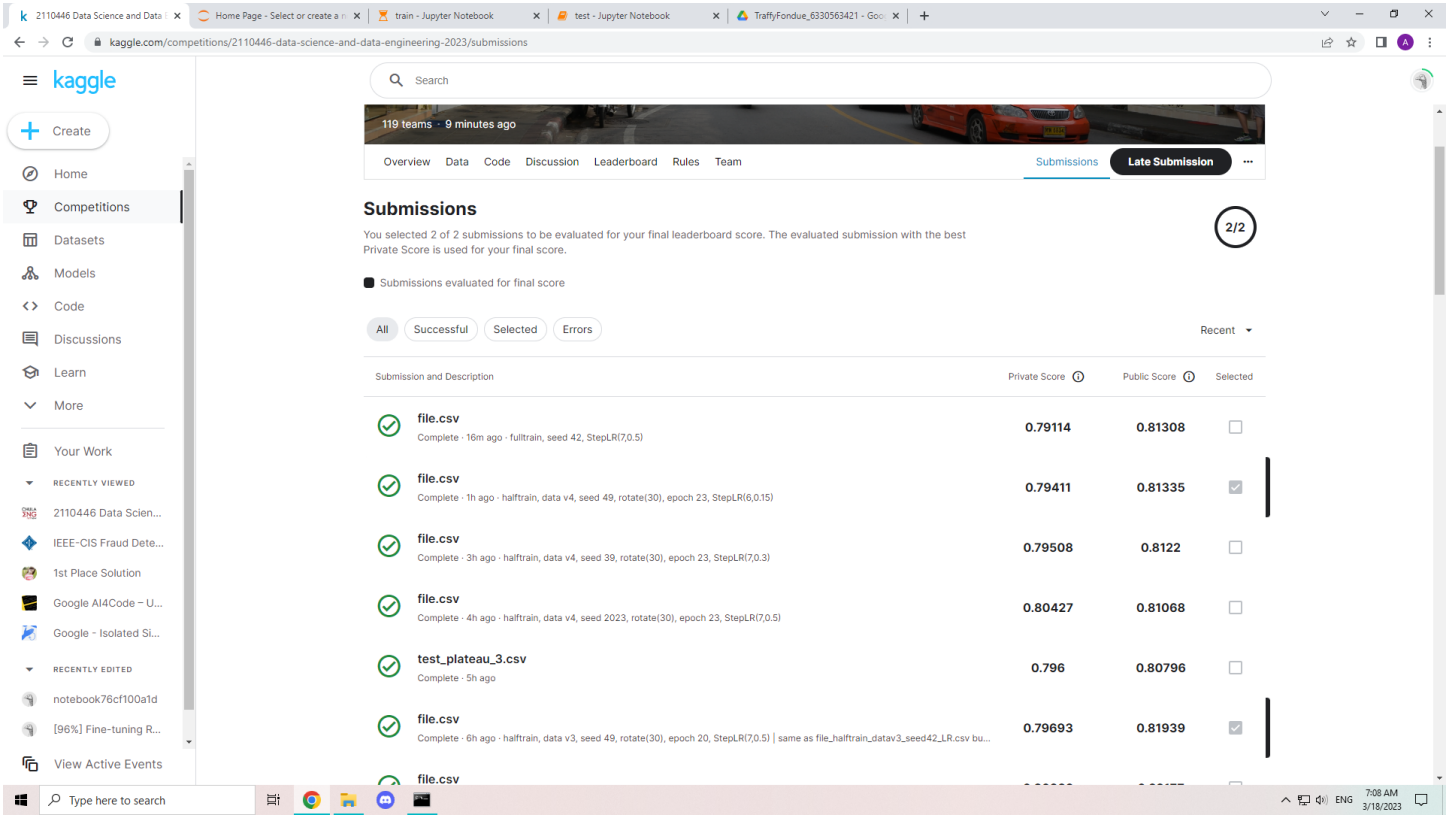
	precision	recall	f1-score	support
0	0.9000	0.8571	0.8780	63
1	0.9009	0.9259	0.9132	108
2	0.9058	0.9328	0.9191	134
3	0.9291	0.9291	0.9291	127
4	0.8760	0.8370	0.8561	135
5	0.8000	0.8889	0.8421	45
6	0.7679	0.6935	0.7288	62
7	0.8308	0.7826	0.8060	69
8	0.8750	0.9032	0.8889	31
9	0.8246	0.9216	0.8704	51
accuracy			0.8752	825
macro avg	0.8610	0.8672	0.8632	825
weighted avg	0.8748	0.8752	0.8743	825

confusion matrix ของ prediction บน test set ที่แบ่งจาก training data

เมื่อนำ model ไปทำการ predict กับ test data จริงๆ เพื่อส่งผลไปตรวจกับ Kaggle ในส่วนของ public leaderboard (60% ของ test data) ได้ Macro F1-Score = 0.81939 แต่ในขณะเดียวกันกลับได้คะแนน Macro F1-Score บน private leaderboard (40% ของ test data) = 0.79693 (รูปเต็มหน้าถัดไป)



ผลลัพธ์จาก Kaggle ตัวเลขฝั่งซ้ายคือ private leaderboard ส่วนฝั่งขวาคือ public leaderboard



ผลลัพธ์จาก Kaggle ฉบับ screenshot เต็มทั้งหน้า

Discussion

จากผล Macro F1-Score ของ private leaderboard ที่ลดลงไปจาก public leaderboard ถึง 0.02264 แสดงให้เห็นว่า ตัว model ที่ผู้จัดทำนั้นทำออกมา ยังมีความ overfit กับข้อมูล training มากเกินไป ซึ่งส่งผลให้พอเวลาเจอ unseen data ประสิทธิภาพการทำนายของ model จึงลดลงอย่างมีนัยยะสำคัญ

นอกจากนี้แล้ว อีกสิ่งหนึ่งที่สามารถเห็นได้อย่างชัดเจน คือจาก validation loss ใน loss graph ในขั้นตอนการ train ซึ่งเริ่มแกว่งขึ้นลงไปหรือลดลงช้ามากตั้งแต่ประมาณ epoch ที่ 6 จึงควรใช้ scheduler เพื่อปรับ learning rate ให้น้อยลงมากกว่าเดิม เพื่อให้เข้าสู่ค่า local maxima ได้ใกล้เคียงมากขึ้น ซึ่งทั้งนี้เองผู้จัดทำได้ลองปรับจูนค่าทั้ง initial learning rate, gamma รวมถึงลองเปลี่ยน scheduler เป็นทั้ง MultiStepLR หรือ ReduceOnPlateau ก็ดีแล้ว แต่ยังไม่สามารถจูนหาค่าที่เหมาะสมมากกว่าเดิมได้ ซึ่งถ้าหากสามารถปรับลดค่า learning rate ใน epoch ที่เหมาะสมมากกว่านี้ได้ ประสิทธิภาพในการทำนายก็ควรจะดีขึ้นเช่นกัน

สำหรับอีกส่วนหนึ่งที่สำคัญคือ ในขั้นตอนของการ clean data ผู้จัดทำเองอาจตีความรูปหลายรูปผิดไป ทำให้ model จึงเรียนรู้ผิดพลาดไปจาก label ผิดไปด้วย นอกเหนือจากนั้นเอง ด้วยปริมาณรูปที่มาก การคัดมือทุกรูปจึงมีโอกาสเกิดข้อผิดพลาดได้มาก ถึงแม้จะมีการตรวจทานซ้ำแล้วทีหนึ่งก็ตาม ทั้งการ reassign label ผิดไปเอง หรือการคัดรูปทิ้งเองก็ดี ซึ่งเป็นการสูญเสีย information ไป รวมถึงรูปที่เป็นประเภทของ near duplicate หรือมีความใกล้เคียงกันมากๆ ซึ่งทำให้ข้อมูลมี bias ในส่วนนี้ด้วย ทั้งนี้ในเบื้องต้น ผู้จัดทำได้ใช้การทำ image hashing เข้ามาช่วยจัดการส่วนนี้แล้วบ้าง แต่ทว่าด้วยปริมาณรูปที่มากทำให้การทำ image hashing ใช้เวลาที่ค่อนข้างจะนานจนเกินไป ผู้จัดทำจึงตัดสินใจไม่ใช้ image hashing ในการหา near duplicate คงเหลือแต่การหา exact duplicate เพียงเท่านั้น ทั้งนี้ผู้จัดทำมองว่าในส่วนนี้มีความสำคัญมากที่สุด ถ้าหากสามารถพัฒนาลด error จากส่วนนี้ได้ ผลลัพธ์จะดีขึ้นอย่างมีนัยยะสำคัญแน่นอน

Conclusion

จากการทำ Image Classification model สำหรับการ classify ปัญหาของ Traffy Fondue ผู้จัดทำสามารถทำ model ที่ได้ Macro F1-Score ของ public และ private leaderboard ไปเป็น 0.81939 และ 0.79693 ตามลำดับ ซึ่งได้ต่ำกว่าเป้าหมายที่คาดหวังไว้ที่ 0.80 private leaderboard Macro F1-Score เพียงเล็กน้อยเท่านั้น ทั้งนี้เป็นเพราะส่วนของการที่ model มีความ overfit กับ train data มากเกินไปจากการ setup train parameter ต่างๆ ที่ดีไม่พอ รวมถึงจากขั้นตอนของการ clean data ที่ยังมี human error, bias จาก duplicate data และการตีความ class ของรูปที่อาจไม่เหมือนกับที่ผู้ร้องเรียนต้องการ ซึ่งถ้าหากผู้จัดทำสามารถแก้ไขความผิดพลาดเหล่านี้ได้ model ของผู้จัดทำจะสามารถบรรลุเป้าหมายที่คาดหวังไว้ได้อย่างแน่นอน