

## รายงานโครงงาน วิชา Deep learning

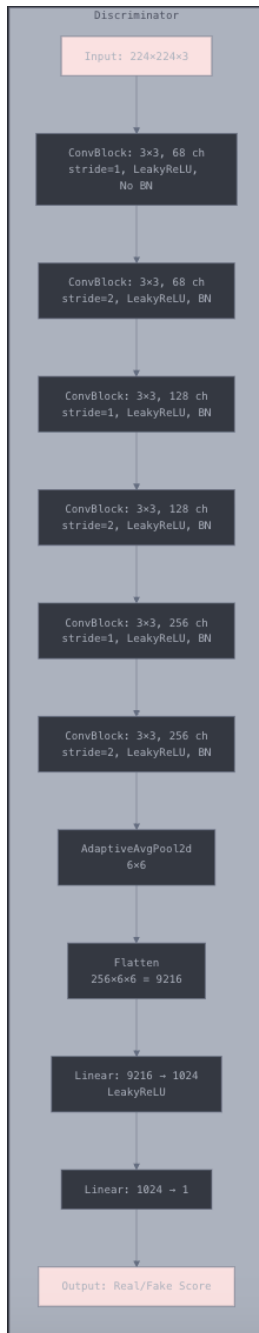
หัวข้อ :Ai ทำให้ภาพ ชัดขึ้น

ที่มาและความสำคัญ:

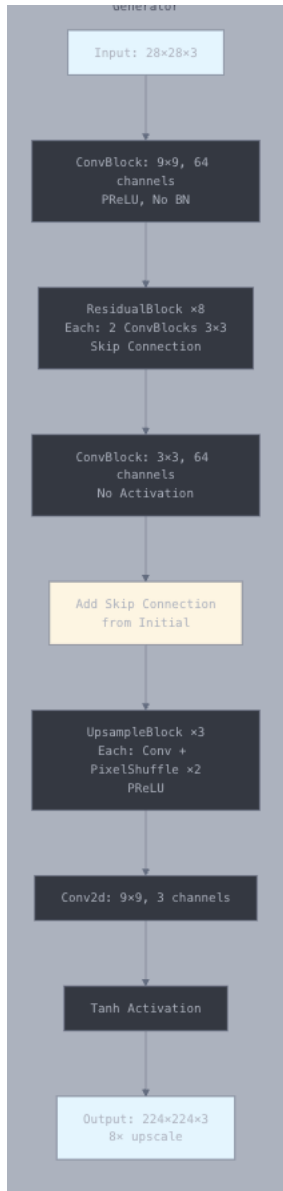
เนื่องจากโทรศัพท์ของผม มีคุณภาพของกล้องที่ไม่ได้มีคุณภาพมากนัก เลยทำให้เวลาที่เรายถ่ายรูปหรือพยายาม zoom รูป ให้ใหญ่ๆ ก็จะได้รูปที่ ไม่ค่อยชัดหรือ มีคุณภาพต่ำ ซึ่งทำให้ผมได้เกิด ความคิดที่ว่า เราจะ สามารถมาใช้ model deep learning ในการทำให้ภาพชัดได้หรือไม่ โจนการ ที่เรา ที่เรามี input เป็นรูปที่ไม่ชัดจากนั้น ให้ model ได้ไป predict ออกมาว่าเป็นอย่างไร โดยเทียบกับ รูปที่ชัด

วิธีการ: ใช้ model SRGAN ในการทำนายภาพออกมาโดย ให้ ออกแบบแต่ละ layer ดังนี้

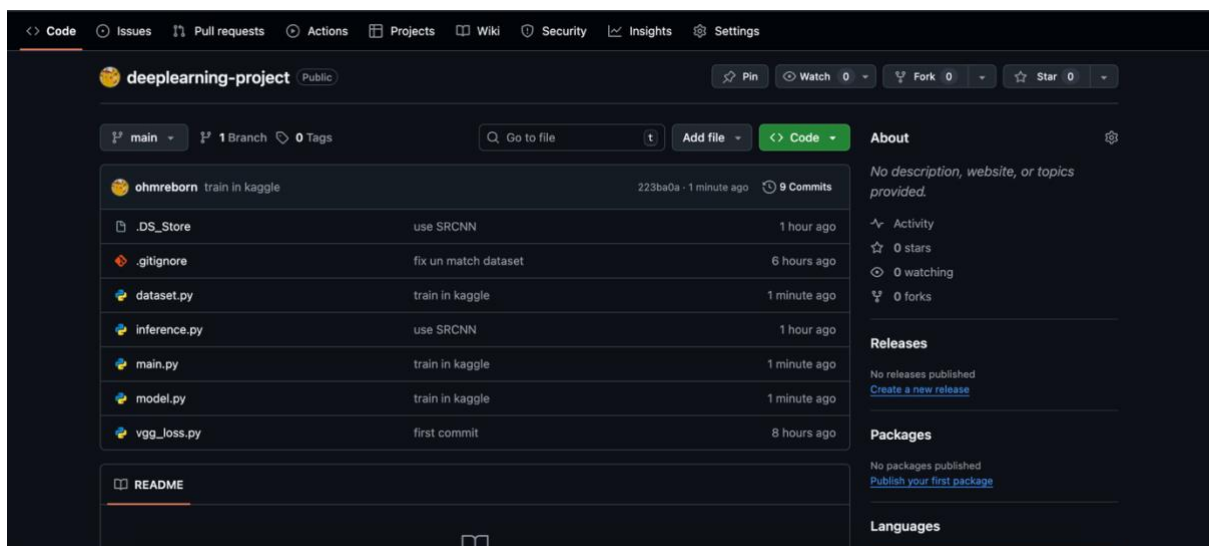
Discriminator:



## Generator:



โดย แต่ละส่วนของ code มีดังนี้



1. main.py ซึ่งเป็น โค้ดที่เอาไว้ใช้ train model ซึ่งเทรนใน Kaggle ใช้ GPU Nvidia T4 / ตัว ในการเทรน ตัวแรกเทรน Generator อีกตัวเทรนให้ discriminator
2. dataset.py เป็น code ที่ preprocess dataset ด้วยการ อ่าน dataset จาก DIV2K จากนั้นก็เอาไปให้ไฟล์ main.py เทรนต่อไป
3. model.py เป็น ส่วนที่สร้างตัว model ขึ้นมาซึ่งได้อธิบายไปก่อนหน้านี้แล้ว
4. vgg\_loss.py เป็นส่วนที่ เอา model vgg มาช่วย คำนวณค่า loss
5. inference.py เป็นส่วนที่ลอง เอา model มาใช้จริง

อธิบายวิธีการในการ train

Train discriminator:

ให้ Generator ทำให้ ภาพที่รับมา ชัดขึ้น (โดยไม่คำนวณ back propagation) จากนั้น ก็ให้ discriminator ทาย จากนั้นทำซ้ำกับ label รูปที่ ชัดจากนั้น เอา ค่าที่ได้ทั้งคู่ไปคำนวณค่า loss หาค่าเฉลี่ยออกมา จากนั้น หาค่า ด้วย gradient accumulation step เนื่องจาก ข้อมูลภาพที่ได้มีหลายขนาดเลยทำให้ไม่สามารถเอามา stack กันได้ จากนั้น ก็ปรับค่า parameters ของ discriminator เมื่อ ทำ back propagation ครบ 16 ครั้ง (ตาม gradient accumulation step)

Train generator:

ให้ generator ทำให้ภาพที่รับมาชัดขึ้น จากนั้น เอาไปให้ discriminator ทายค่าแล้วคำนวณ loss อันแรก จากนั้น ก็เอาไปเทียบ กับ ภาพชัดจริงๆ ที่ละ pixel ด้วย mean square error เป็น loss ตัวที่ 2 จากนั้น เอา model vgg มาช่วย คำนวณ l1 loss เป็น loss ตัวที่ 3  
เอา ทั้ง 3 มารวมกัน ตามสมการ

```
1
2     g_loss = pixel_loss + 1.0 * perc_loss + 1e-3 * adv_loss.to(0)
3     g_loss = g_loss / gradient_accumulation_step # 0
4     g_loss.backward() # 1
5
6     if (i + 1) % gradient_accumulation_step == 0:
7         opt_G.step()
8         opt_G.zero_grad()
```

แล้วก็ปรับค่า weight เหมือนกับ Discriminator

อ้างอิง:

Model: <https://arxiv.org/abs/1609.04802>

Dataset: <https://data.vision.ee.ethz.ch/cvl/DIV2K/>