

Colorization Black & White image using Generative Adversarial Network

Final Project Coding in AI (CPE 393)

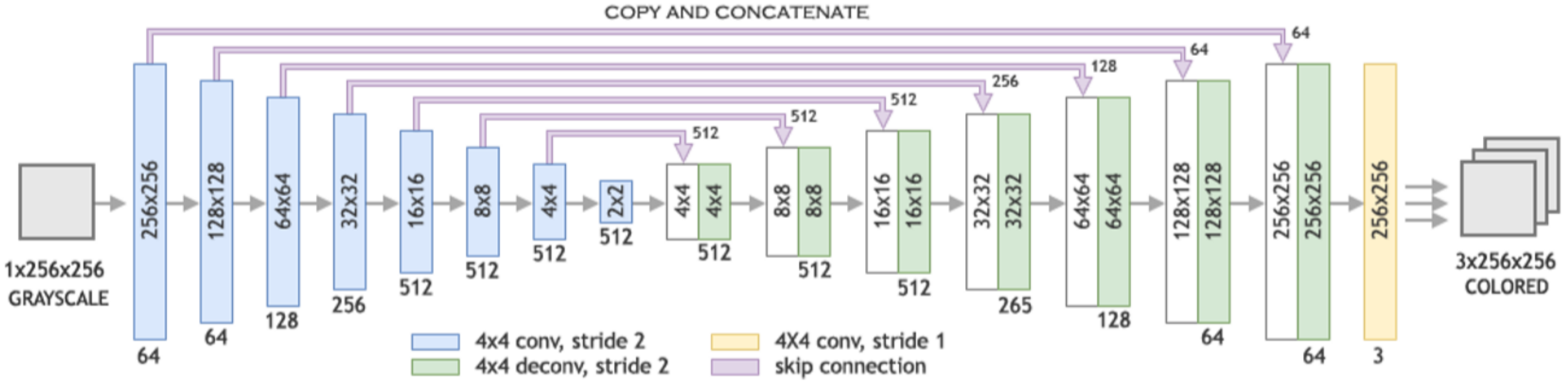
เปลี่ยนภาพขาวดำให้เป็นภาพสี

วิธีที่ใช้ในการทำ Image Colorization

- Convolutional Neural Network
- Generative Adversarial Network

ข้อแตกต่างระหว่าง CNN(UNET) กับ GAN

CNN (Unet)



Contractive Path

Expansive Path

CNN (Unet)

- โครงสร้างของ Unet
  - มี N Encoding Layers และ N Decoding Layers
  - เป็นโครงสร้างแบบสมมาตร
  - Contracting path : ในแต่ละ layer ใช้ Convolution Stride 2 เพื่อ Downsampling และตามด้วย Batch Normalization กับ Activation Function เป็น LeakyReLU
  - Expansive path : ในแต่ละ layer ใช้ Transposed Convolution Stride 2 เพื่อ Upsampling และตามด้วย Batch Normalization กับ Activation Function เป็น ReLU และ ยัง concatenate กับ layer ที่อยู่ตรงข้าม (ฝั่ง contracting path)
  - Layer สุดท้าย : เป็น 1x1 convolution ที่ทำให้ output ออกมาเป็น 3 channel

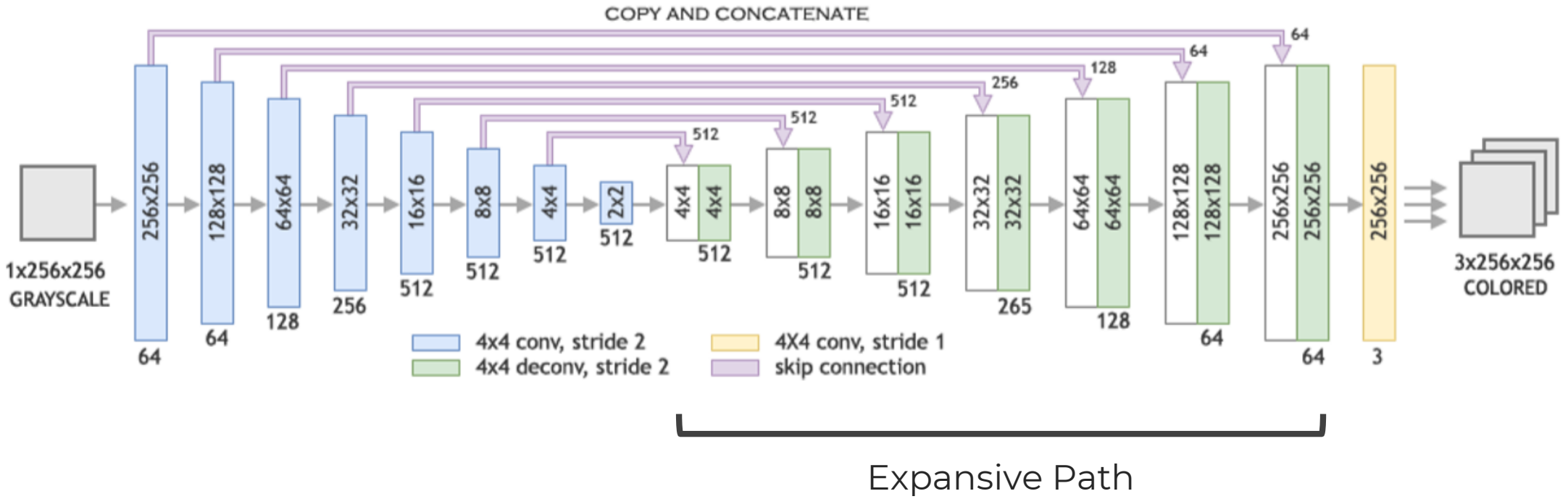
CNN (Unet)

- มีไอเดียมาจาก Encoder-Decoder Network

- เป็น **Fully convolution network model** โดยส่วนของ Fully connected layers **จะถูกแทนที่ด้วย convolutional layers ซึ่งประกอบไปด้วย upsampling** แทนที่จะเป็นการทำ pooling

- **ข้อดี** โมเดลแบบ end-to-end จะใช้ memory ไม่มาก
- **ข้อเสีย** การทำ downsampling จะทำให้เกิดการบีบอัดของ feature ในชั้นตรงกลาง ความละเอียดของตรงส่วนนี้จะถูกจำกัดด้วย GPU memory

CNN (Unet)

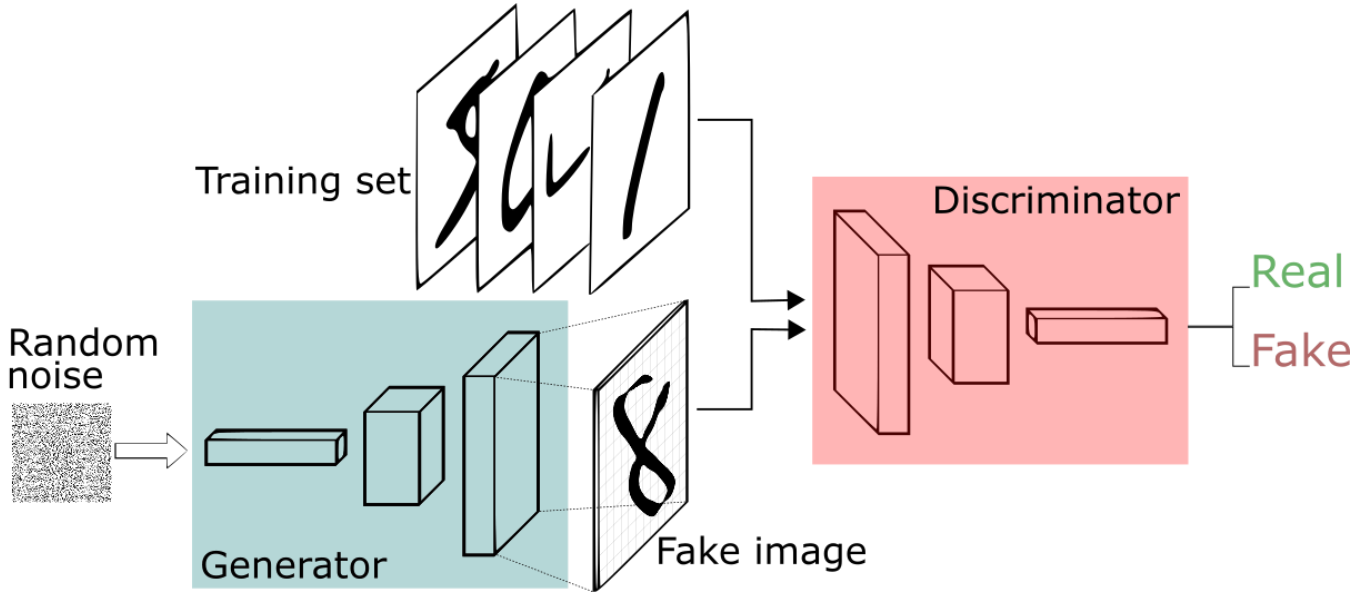


- เพื่อแก้ไขปัญหาของส่วน Bottleneck, features จาก contracting path จะถูก concatenate กับ upsampled output ในช่วงของ expansive path ของ network

GAN

- ประกอบไปตัวโครงข่ายเล็ก ๆ สองตัวประกอบกันคือ
  - **Generator Model** ทำหน้าที่ในการสร้างผลลัพธ์ของรูปภาพขาวดำที่ทำการวิเคราะห์ โดยต้องสร้างสีออกมาให้แยกไม่ออกจากข้อมูลต้นฉบับ
  - **Discriminator Model** ทำหน้าที่ในการแยกผลลัพธ์ที่ได้ออกมาจากตัว Generator model เทียบกับ ตัวอย่างที่ได้มาจากต้นฉบับ ว่ามีความเหมือนกับต้นฉบับหรือไม่
- โดยจะ train จนกว่า Generator จะสร้างรูปที่ทำให้ Discriminator แยกไม่ออก

GAN



- Generator Model รับ input เป็น Noise Z
- Discriminator Model รับ input เป็นรูปจาก Generator Model กับ Original Image

GAN

$$\min_{\theta_G} J^{(G)}(\theta_D, \theta_G) = \min_{\theta_G} \mathbb{E}_z \left[\log(1 - D(G(z)))\right],$$

- Cost Function ของ Generator Model

$$\max_{\theta_D} J^{(D)}(\theta_D, \theta_G) = \max_{\theta_D} \left(\mathbb{E}_x \left[\log(D(x))\right] + \mathbb{E}_z \left[\log(1 - D(G(z)))\right]\right)$$

- Cost Function ของ Discriminator Model

GAN

- แต่เนื่องจาก ***รูปภาพขาวดำ ไม่ใช่ noise*** แบบก่อนหน้านี้ ***จึงถือว่าไม่มี noise*** ฉะนั้นจึงถูกประยุกต์เป็น Conditional GAN และ Cost function จึงเปลี่ยนไป เพราะมีการนำ Original image มาเป็น input ของ Discriminator ด้วย

$$\min_{\theta_G} J^{(G)}(\theta_D, \theta_G) = \min_{\theta_G} -\mathbb{E}_z \left[\log(D(G(\mathbf{0}_z|x)))\right] + \lambda\|G(\mathbf{0}_z|x) - y\|_1$$

$$\max_{\theta_D} J^{(D)}(\theta_D, \theta_G) = \max_{\theta_D} \left(\mathbb{E}_y \left[\log(D(y|x))\right] + \mathbb{E}_z \left[\log(1 - D(G(\mathbf{0}_z|x)|x))\right]\right)$$

Regularization Term

Conditional GAN Architecture

- Generator Model : ใช้โครงสร้างเดียวกับ Unet
- Discriminator Model : ให้โครงสร้างคล้ายกับส่วน encoding ของ Unet
- ซึ่ง Convolution layer ทุกชั้น จะตามด้วย Batch Normalization และ Activation function เป็น LeakyReLU
- Layer สุดท้าย เป็น 1 dimension output และใช้ activation เป็น sigmoid ซึ่งจะให้ความน่าจะเป็นว่า Fake หรือ Real

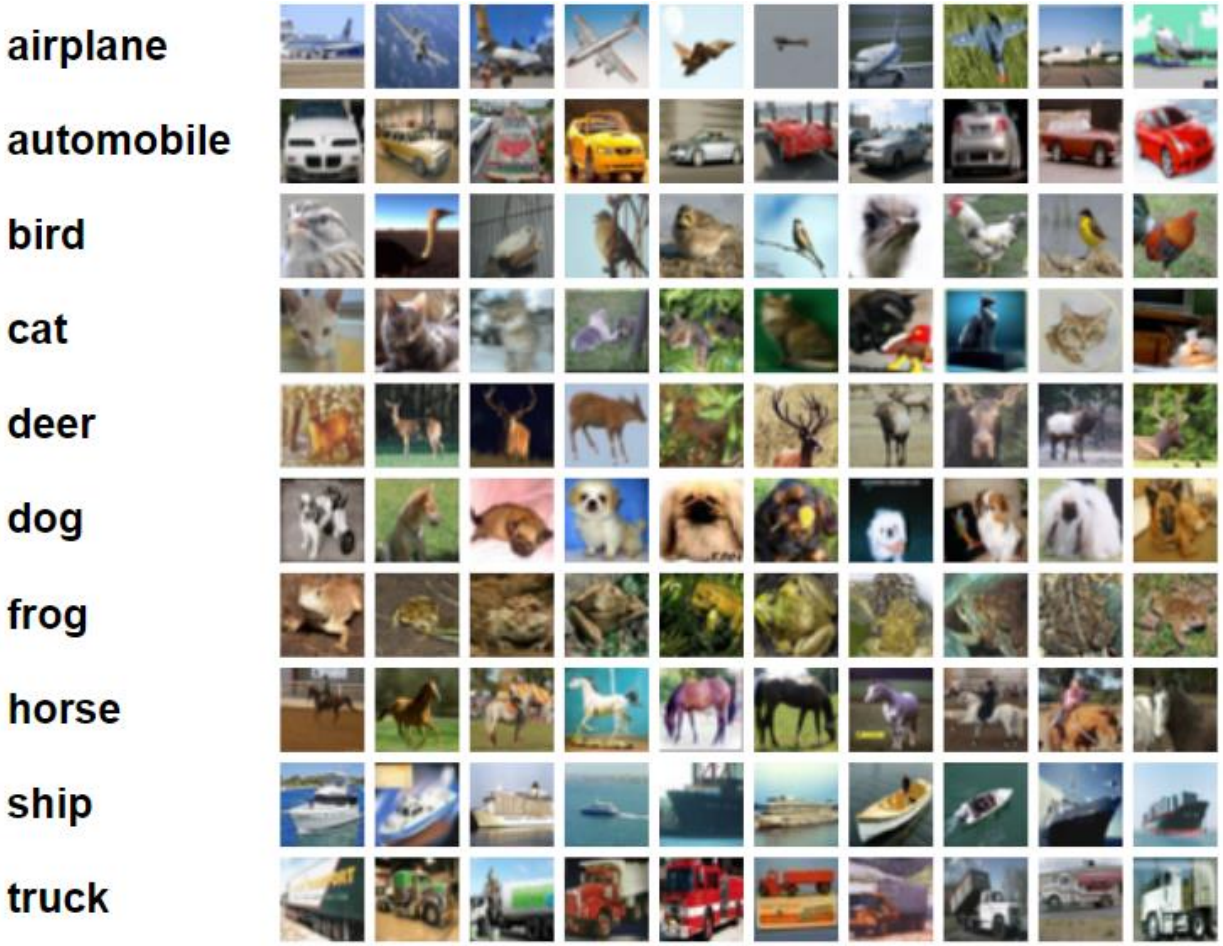
EXPERIMENT

ทำการทดลองกับ Model 4 รูปแบบ ดังนี้

- CNN(U-net)
- CNN(U-net) ใช้ร่วมกับ LeakyRelu และ BatchNormaliztion
- GAN
- GAN ใช้ร่วมกับ LeakyRelu และ BatchNormalization

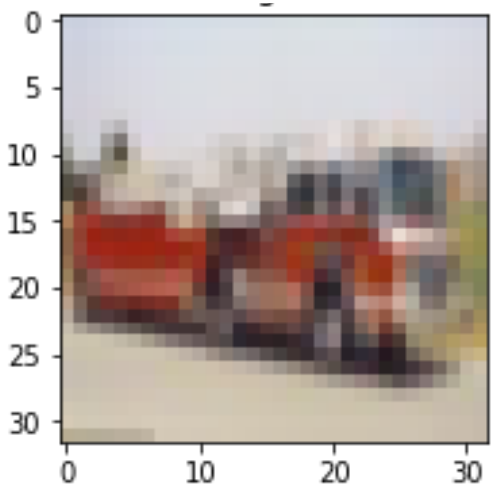
Dataset CIFAR-10

60,000 32x32 color images in 10 classes (Training 50,000, Test 10,000) – 15 EPOCH ONLY!



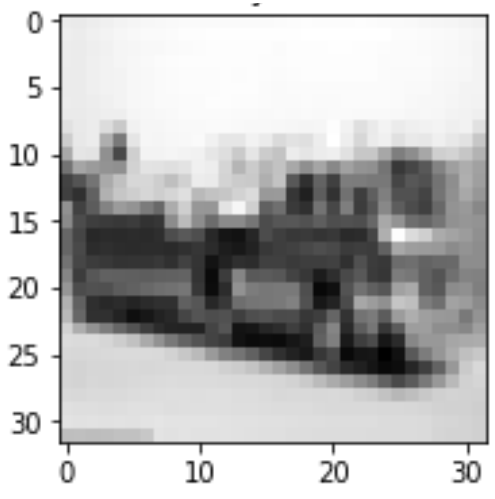
Result : Unet Vs Unet (Leak ReLU + Batch Normalize)

1



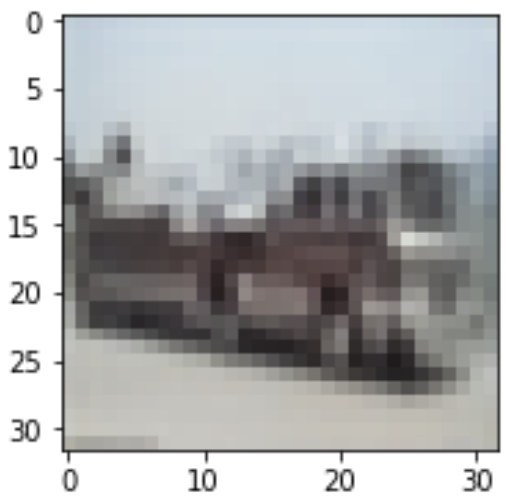
Original

2



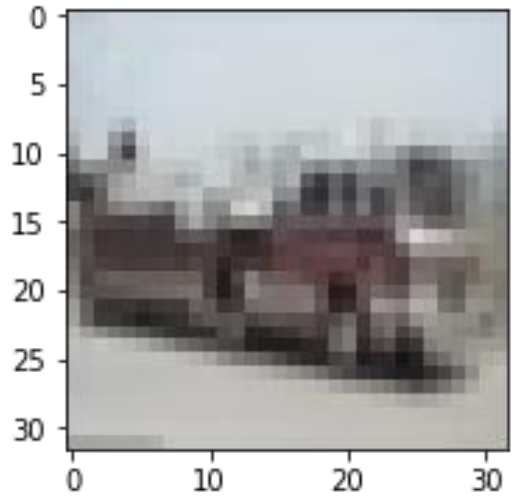
Grayscale

3



Unet

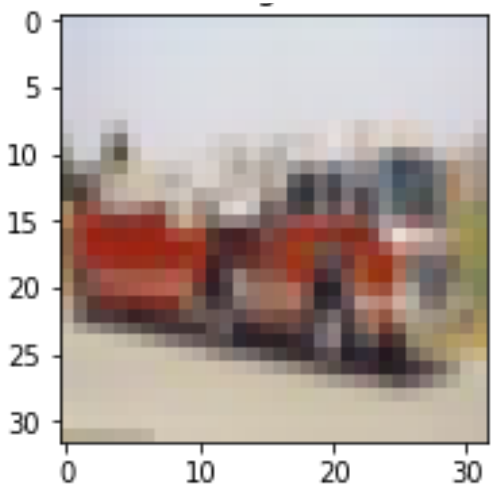
4



Unet (Leaky ReLU + Batch Normalize)

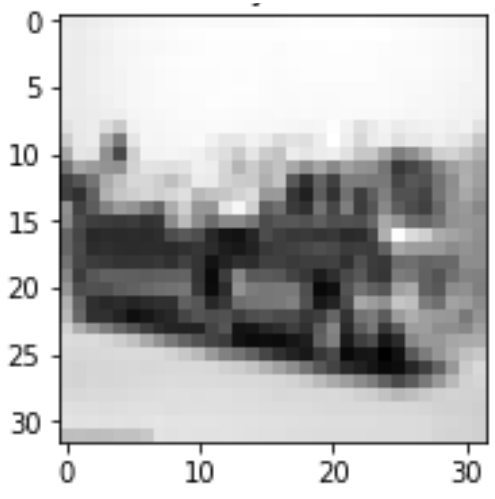
Result : GAN Vs GAN (Leak ReLU + Batch Normalize)

1



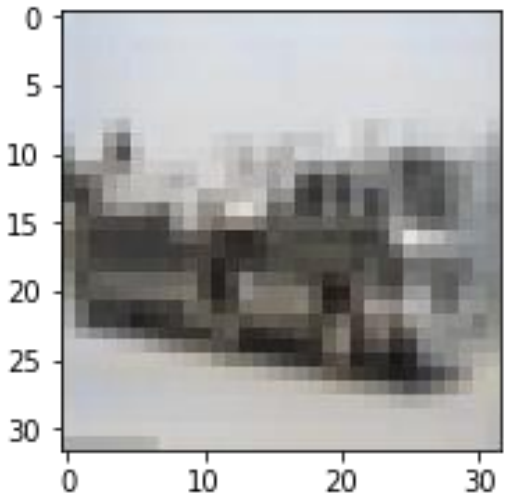
Original

2



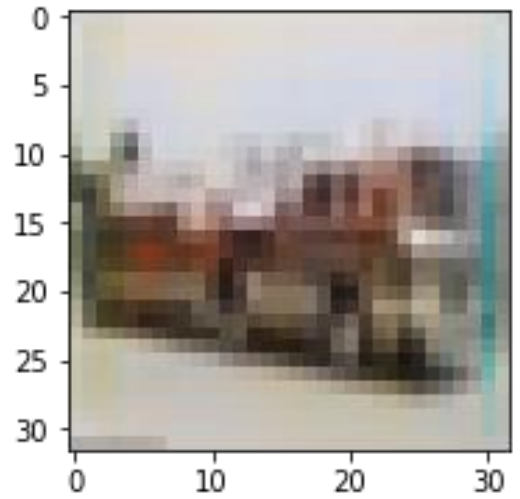
Grayscale

3



GAN

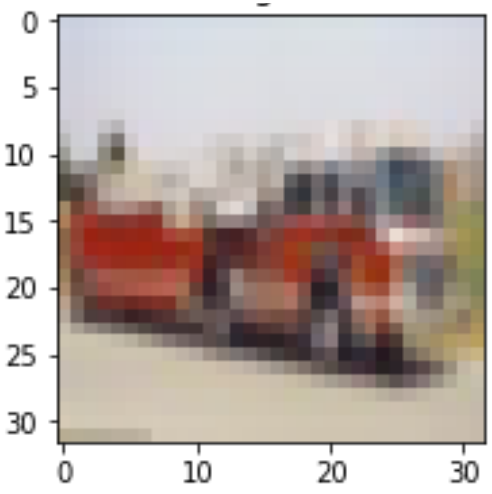
4



GAN (Leaky ReLU + Batch Normalize)

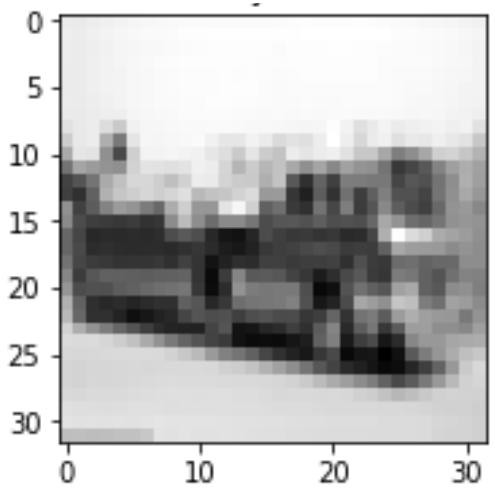
Result : Best Unet Vs Best GAN

1



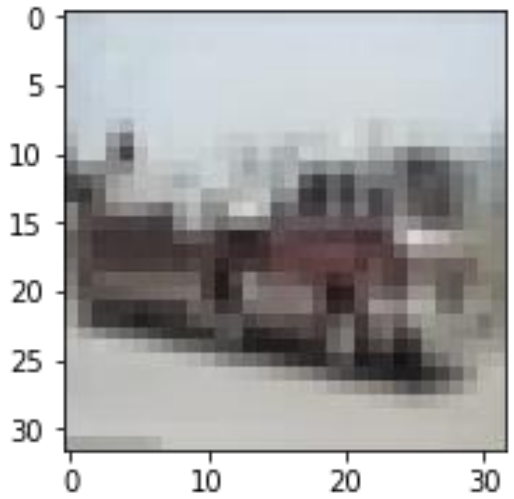
Original

2



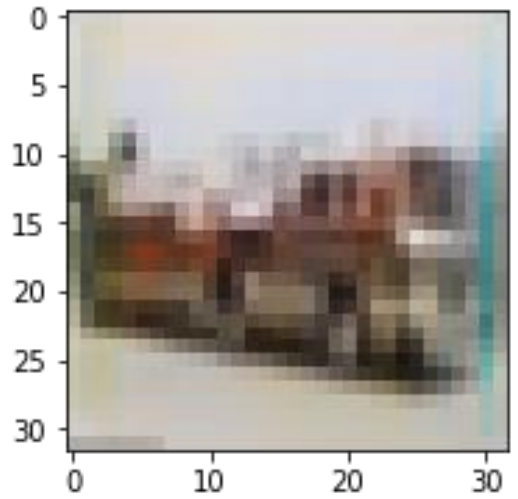
Grayscale

3



Unet (Leaky ReLU + Batch Normalize)

4



GAN (Leaky ReLU + Batch Normalize)

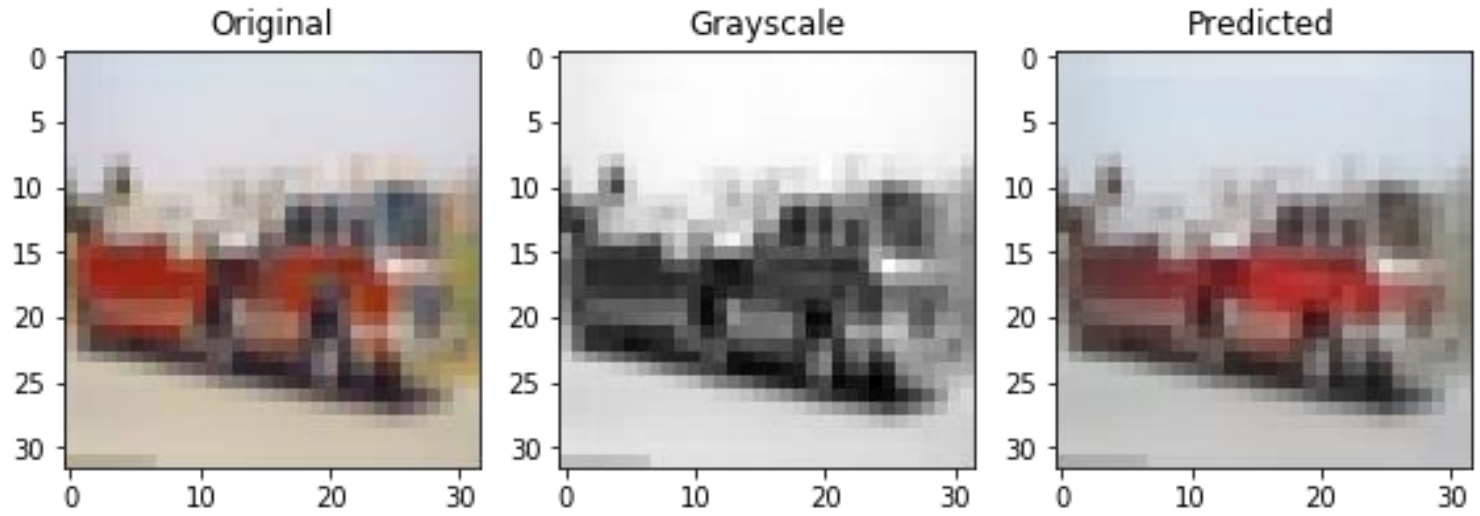
Result of performance

Dataset	Network	Epoch	Accuracy	Loss
CIFAR-10	CNN(U-net)	15	0.012071	0.556810
CIFAR-10	CNN(U-net) (LeakyRelu)	15	0.012075	0.556328
CIFAR-10	GAN(U-net)	15	0.012059	0.559959
CIFAR-10	GAN(U-net) (LeakyRelu)	15	0.012037	0.567164

สรุปข้อผิดพลาด

- ขั้นตอนการ train GAN ไม่ได้ใส่ตัววัดผลอีกหนึ่งตัวที่เรียก MAE (Mean absolute error) เป็นหาค่าเฉลี่ยของ error ทั้งหมดจากรูปภาพที่สร้างขึ้นมาเมื่อเทียบกับภาพจริง ทำให้การวัดผลไม่มีประสิทธิภาพมากเพียงพอ
- Train model ของ GAN ไม่ทัน เนื่องจาก GAN ใช้เวลาเยอะมาก ทำให้เวลาที่ใช้ในการเทรนไม่เพียงพอ ได้มากสุดที่ 15 epoch
- เราไป focus กับการใช้รูปที่มีขนาดใหญ่ก่อน คือ ใช้ dataset place 365 แต่ใช้เวลามากเกินไป สุดท้ายกลับมา focus ที่ dataset ขนาดเล็ก เพราะต้องการให้มีความคืบหน้า
- ใน paper ที่อ้างอิงใช้รูปภาพที่เป็น L*a*b แต่ในการทดลองเราใช้ RGB

THANK YOU



Unet (Leaky ReLU + Batch Normalize) 50 epoch