DATS_6203_10: Final Project

Photo Coloring
by: Group 4

Introduction

- Photo Coloring

gray pictures

color pictures

- Applications: restoring old images, 2D animation..
- Main method: Generative Adversarial Network
- Additional method: Conditional Adversarial Networks

Dataset Description

- Flickr1024 - A large-scale stereo image dataset consisting 1024 high-quality image pairs and covering diverse scenarios like animals, building, lands, plants.
- https://yingqianwang.github.io/Flickr1024/
- size: 2.64GB

Literature Review

Image colorization: https://arxiv.org/abs/1803.05400

Deep Convolutional GAN: https://arxiv.org/abs/1511.06434

Image to Image translation:
https://openaccess.thecvf.com/content_cvpr_2017/papers/Isola_Image-To-Image_Translation_With_CVPR_2017_paper.pdf

GANs - Method based on color space

- GAN: G: Generative; A: Adversarial(discriminator); N: network.
- Photo transformation: LAB color space

❖ L: Lightness - gray photo

❖ A: Green-Red tradeoff

❖ B: Blue-Yellow tradeoff

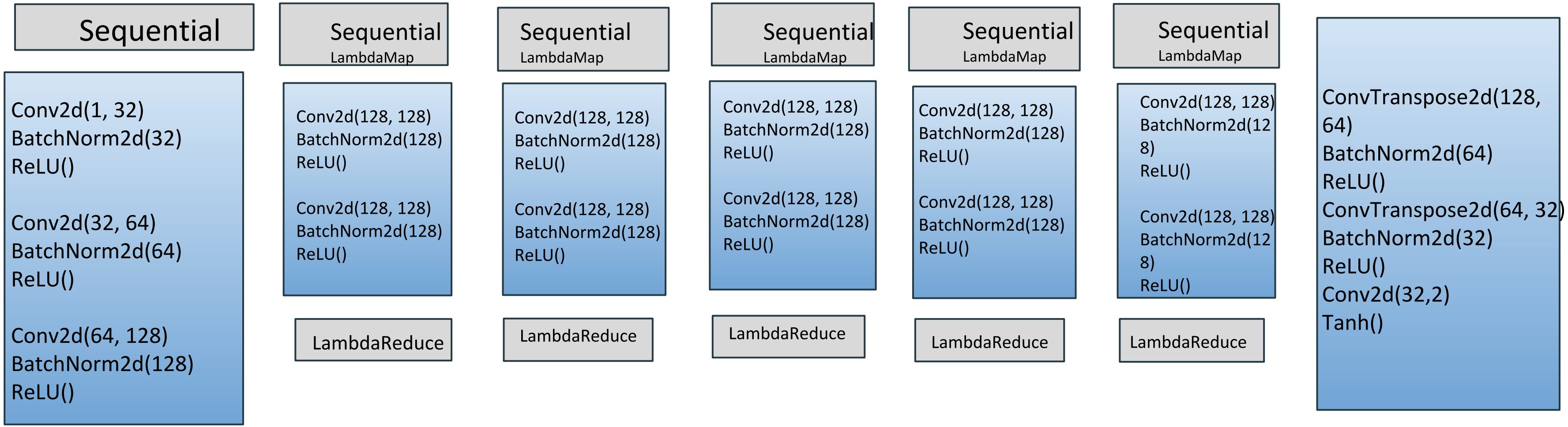
❖ Input: L channel sequence, Output: A, B channel sequence

Data Preprocessing

Package - Open CV

- Use cv2.imread() to read to photo
- Resize to (256, 256) - Due to computation power limation and training time
- Tranform color space for BGR to LAB

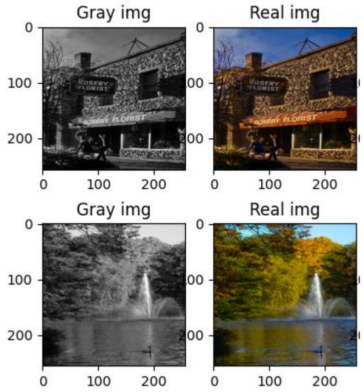
Model


Sequential
Conv2d(1, 32)
BatchNorm2d(32)
ReLU()
Conv2d(32, 64)
BatchNorm2d(64)
ReLU()
Conv2d(64, 128)
BatchNorm2d(128)
ReLU()
Sequential
LambdaMap
Conv2d(128, 128)
BatchNorm2d(128)
ReLU()
Conv2d(128, 128)
BatchNorm2d(128)
ReLU()
LambdaReduce
Sequential
LambdaMap
Conv2d(128, 128)
BatchNorm2d(128)
ReLU()
Conv2d(128, 128)
BatchNorm2d(128)
ReLU()
LambdaReduce
Sequential
LambdaMap
Conv2d(128, 128)
BatchNorm2d(128)
ReLU()
Conv2d(128, 128)
BatchNorm2d(128)
ReLU()
LambdaReduce
Sequential
LambdaMap
Conv2d(128, 128)
BatchNorm2d(128)
ReLU()
Conv2d(128, 128)
BatchNorm2d(128)
ReLU()
LambdaReduce
Sequential
LambdaMap
Conv2d(128, 128)
BatchNorm2d(12
8)
ReLU()
Conv2d(128, 128)
BatchNorm2d(12
8)
ReLU()
LambdaReduce
ConvTranspose2d(128,
64)
BatchNorm2d(64)
ReLU()
ConvTranspose2d(64, 32)
BatchNorm2d(32)
ReLU()
Conv2d(32,2)
Tanh()

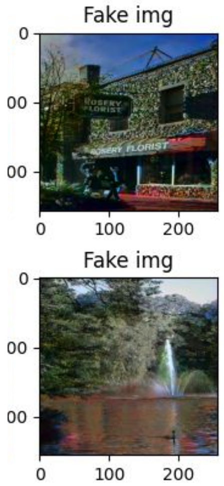
Training process

- Loss function - Cross entropy
    - Generator loss: cross entropy loss of generated images + MSE of ab channels
    - Discirmation loss: (loss of generated images + loss of real images) / 2
- Flip label generated images - 1 and real images - 0 (works)
- Soft and Noisy label: Using a random number between 0 and 0.1 to represent 0 labels (real images) and a random number between 0.9 and 1.0 to represents 1 labels (generated images) when training the discriminator. (works)

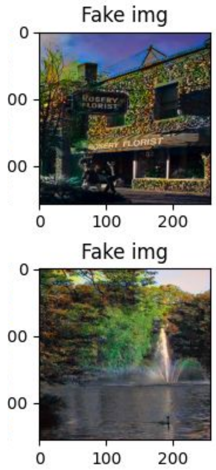
Results



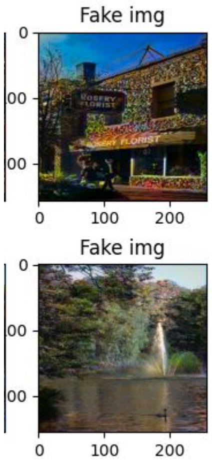
training of 100 epochs:



training of 200 epochs:



training of 300 epochs:



Conditional GANs

- predict pixels from pixels
- learn a mapping from observed image x and random noise vector z
- The objective of a conditional GAN:

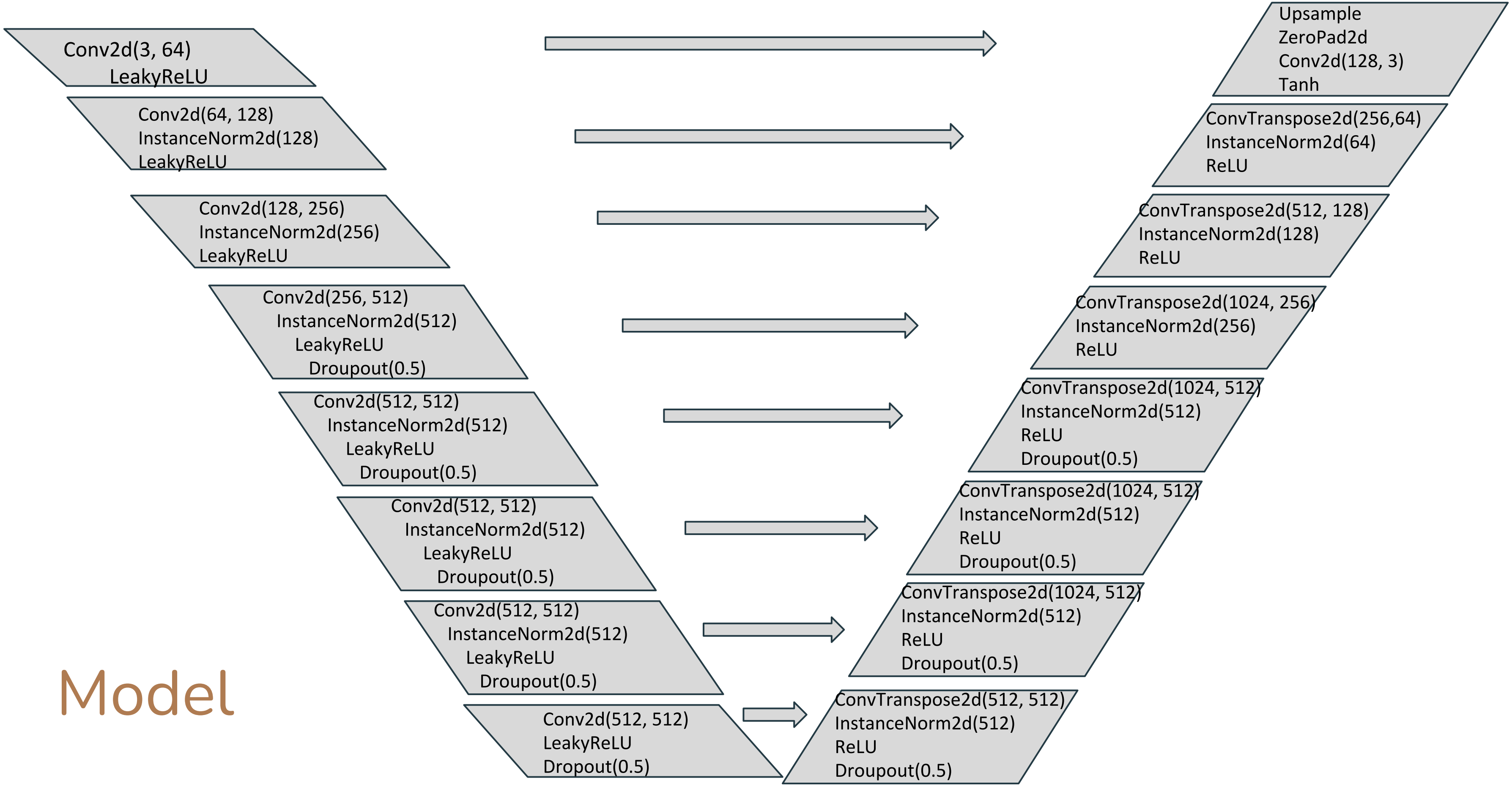
$$\mathcal{L}_{cGAN}(G, D) = \mathbb{E}_{x,y}[\log D(x, y)] + \mathbb{E}_{x,z}[\log(1 - D(x, G(x, z))],$$

- Markovian discriminator(PatchGAN): only penalizes structure at the scale of patches

Data Preprocessing

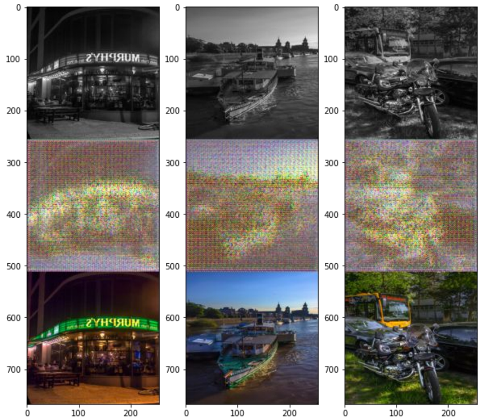
- Read the image from google drive
- Use cv2.cvtColor to convert BGR to RGB.
- To get the color image, so we turn the data into image format. We used Image.fromarray() to get the image-format RGB dataset.
- Use cv2.cvtColor again to finish the process from BGR-RGB-Gary (make sure it has three channels).
- Transformed both of them into tensors.

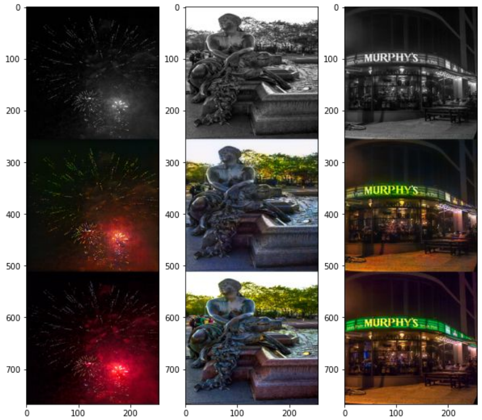
Model



Conv2d(3, 64)
LeakyReLU
Conv2d(64, 128)
InstanceNorm2d(128)
LeakyReLU
Conv2d(128, 256)
InstanceNorm2d(256)
LeakyReLU
Conv2d(256, 512)
InstanceNorm2d(512)
LeakyReLU
Droupout(0.5)
Conv2d(512, 512)
InstanceNorm2d(512)
LeakyReLU
Droupout(0.5)
Conv2d(512, 512)
InstanceNorm2d(512)
LeakyReLU
Droupout(0.5)
Conv2d(512, 512)
InstanceNorm2d(512)
LeakyReLU
Droupout(0.5)
Conv2d(512, 512)
LeakyReLU
Dropout(0.5)
Upsample
ZeroPad2d
Conv2d(128, 3)
Tanh
ConvTranspose2d(256,64)
InstanceNorm2d(64)
ReLU
ConvTranspose2d(512, 128)
InstanceNorm2d(128)
ReLU
ConvTranspose2d(1024, 256)
InstanceNorm2d(256)
ReLU
ConvTranspose2d(1024, 512)
InstanceNorm2d(512)
ReLU
Droupout(0.5)
ConvTranspose2d(1024, 512)
InstanceNorm2d(512)
ReLU
Droupout(0.5)
ConvTranspose2d(1024, 512)
InstanceNorm2d(512)
ReLU
Droupout(0.5)
ConvTranspose2d(512, 512)
InstanceNorm2d(512)
ReLU
Droupout(0.5)

Results





Performance Comparison

Performance Comparison

L*a*b* GAN

Pix2Pix GAN

Conclusion

- Automatically colorizing grayscale images using GAN to an acceptable visual degree
- The model was able to consistently produce better looking (qualitatively) images than real images
- Mis-colorization was a frequent occurrence with images containing high levels of textured details--didn't learn enough from human photos

Limation

- The dataset does not include enough person images. The result is not good on person.
- The training process is very slow.
- And the game between Generator and Discriminator is not balance. Usually, D perform better.

Improvement

- Adding person images into training
- Speeding the training process
- Changing generator structures to be banlanced with discriminator
- seeking a better quantitative metric to measure performance -- all evaluations were qualitative
- Application in coloring videos: With further training and improvement, we can turn the black and white movies into color movies.