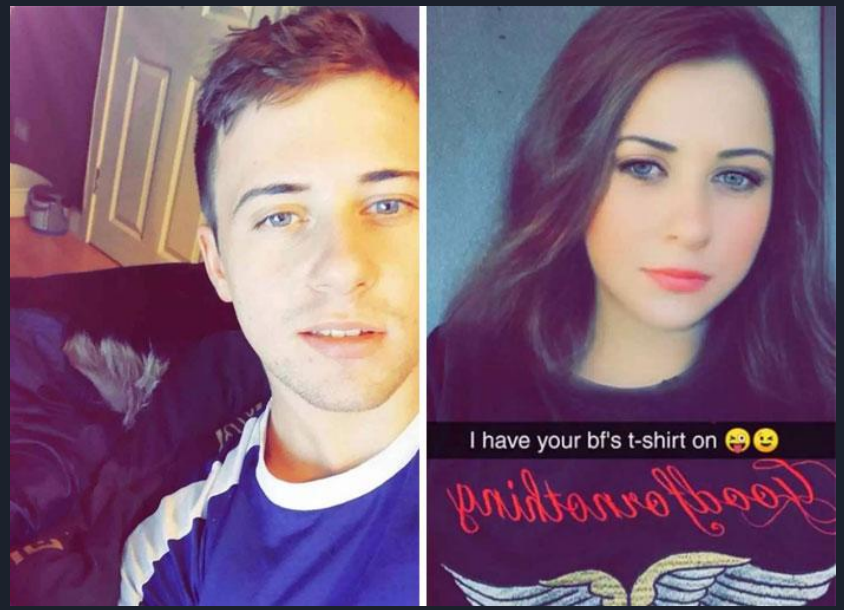
TaikaGAN: Two-Domain Image-to-image Translation for Anime-styled Illustrations

Bautista, Domingo, Tan

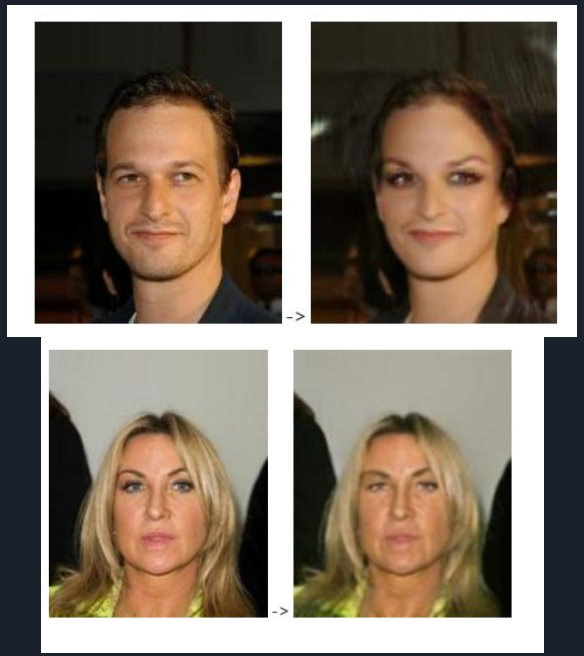
Motivation

Messaging app Snapchat has a filter that enables users to see a gender-swapped version of themselves.



Motivation

Gender-swapping has been implemented via CycleGAN on the CelebA dataset



Source: https://yanjia.li/gender-swap-and-cyclegan-in-tensorflow-2-0

Challenge

What if it can be done to anime characters?

Source: プリズム☆ま〜じカルPRISM Generations！

Ideally, this is what happens.

However, obtaining paired data is impractical.

Dataset

Raw images are obtained from visual novel site Getchu[1]

Images are then manually sorted according to gender, with some quality control enforced

Sorted images are cropped via a face detector for anime/manga[2], and then scaled to **256x256**

Badly cropped images are removed

Image count:

Male ~1,200

Female ~4,200

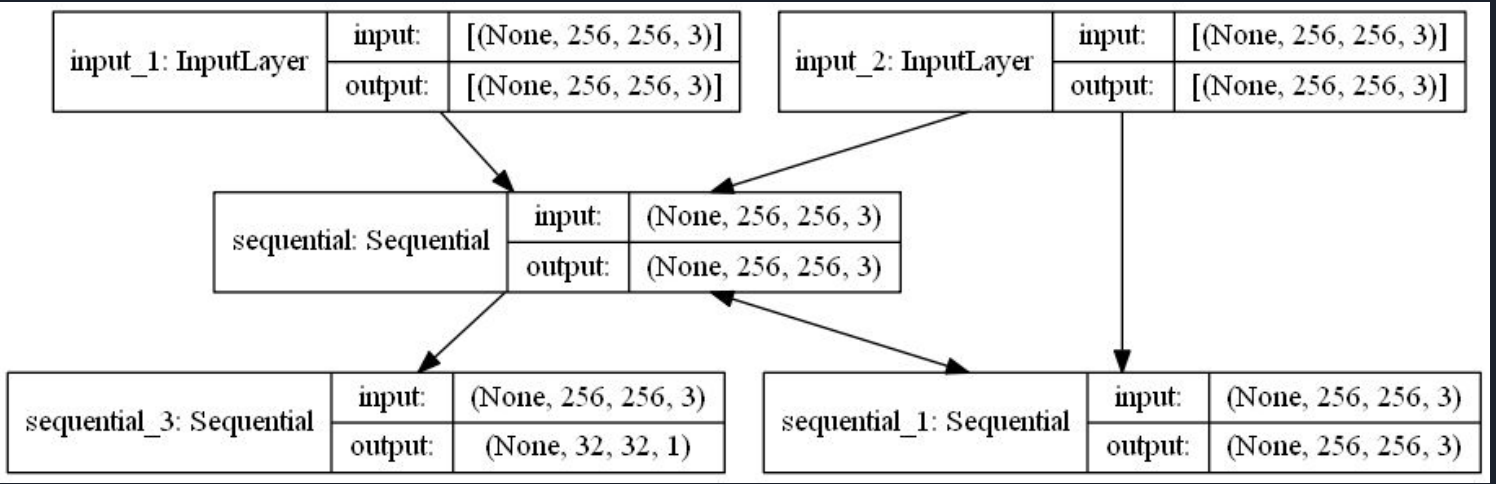
[1]http://www.getchu.com
[2]https://github.com/nagadomi/lbpcascade_animeface

Dataset

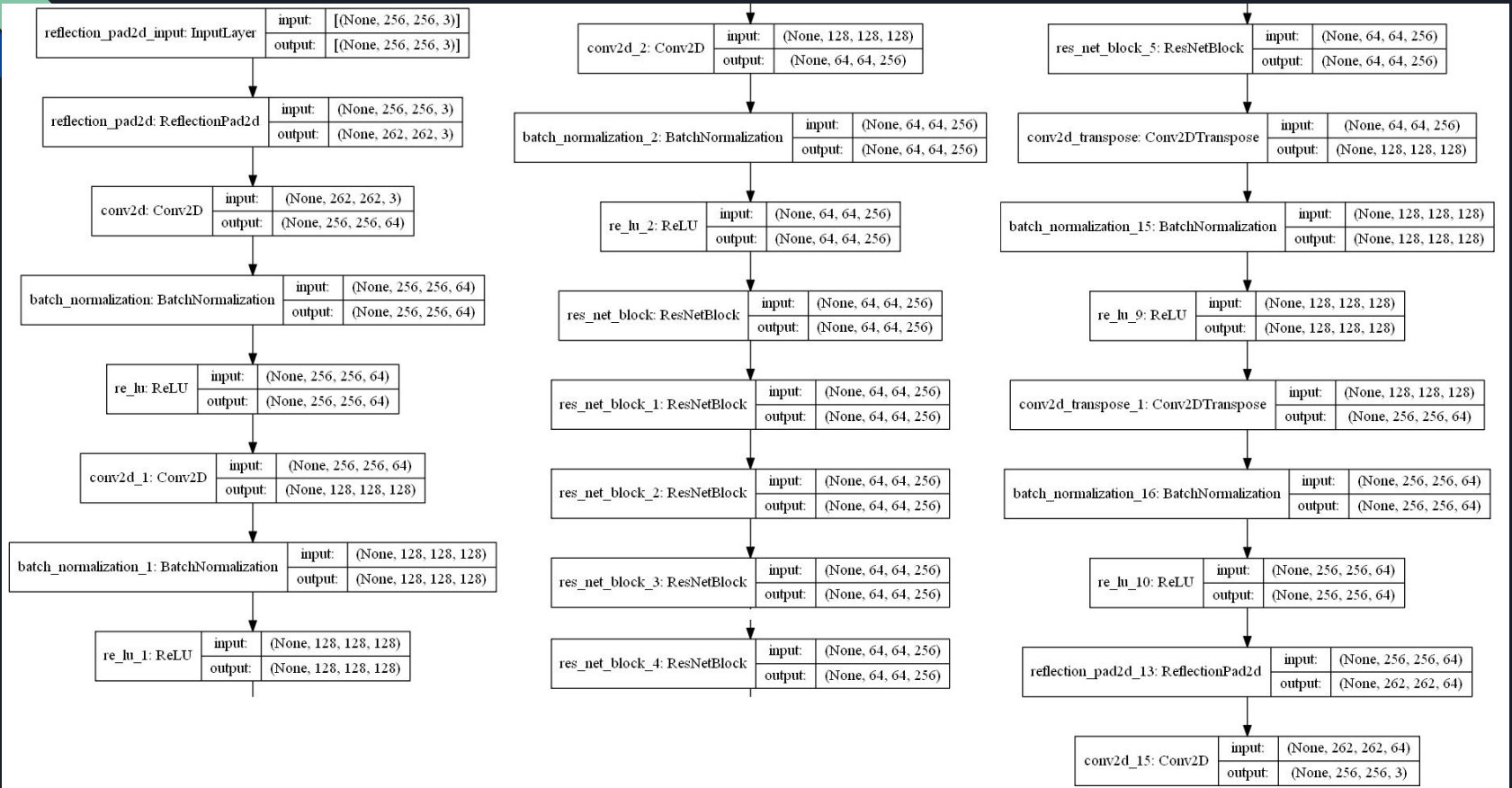
Sample of female faces

Sample of male faces

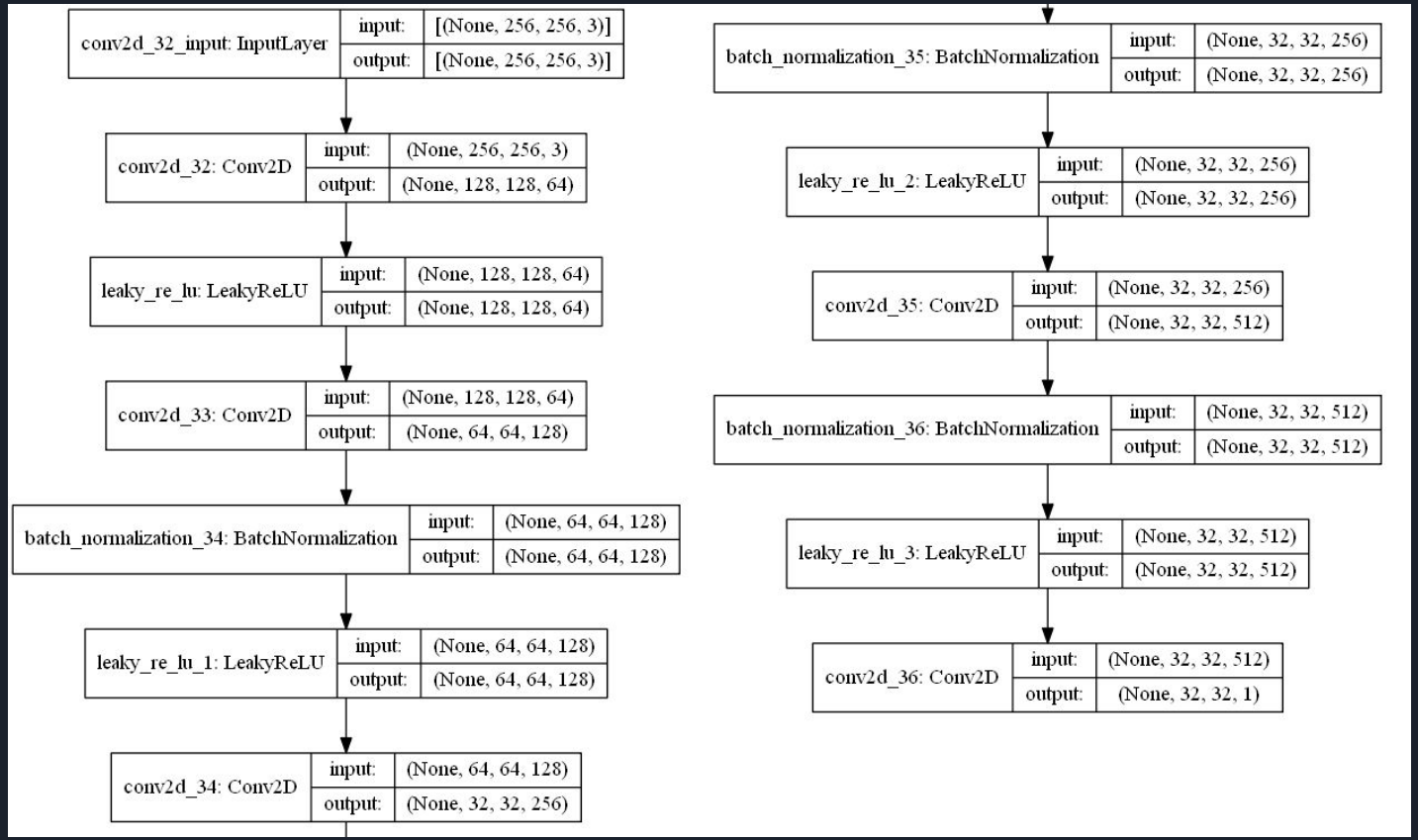
Top-level Architecture



Generator


reflection_pad2d_input: InputLayer | input: [(None, 256, 256, 3)] | output: [(None, 256, 256, 3)]
reflection_pad2d: ReflectionPad2d | input: (None, 256, 256, 3) | output: (None, 262, 262, 3)
conv2d: Conv2D | input: (None, 262, 262, 3) | output: (None, 256, 256, 64)
batch_normalization: BatchNormalization | input: (None, 256, 256, 64) | output: (None, 256, 256, 64)
re_lu: ReLU | input: (None, 256, 256, 64) | output: (None, 256, 256, 64)
conv2d_1: Conv2D | input: (None, 256, 256, 64) | output: (None, 128, 128, 128)
batch_normalization_1: BatchNormalization | input: (None, 128, 128, 128) | output: (None, 128, 128, 128)
re_lu_1: ReLU | input: (None, 128, 128, 128) | output: (None, 128, 128, 128)
conv2d_2: Conv2D | input: (None, 128, 128, 128) | output: (None, 64, 64, 256)
batch_normalization_2: BatchNormalization | input: (None, 64, 64, 256) | output: (None, 64, 64, 256)
re_lu_2: ReLU | input: (None, 64, 64, 256) | output: (None, 64, 64, 256)
res_net_block: ResNetBlock | input: (None, 64, 64, 256) | output: (None, 64, 64, 256)
res_net_block_1: ResNetBlock | input: (None, 64, 64, 256) | output: (None, 64, 64, 256)
res_net_block_2: ResNetBlock | input: (None, 64, 64, 256) | output: (None, 64, 64, 256)
res_net_block_3: ResNetBlock | input: (None, 64, 64, 256) | output: (None, 64, 64, 256)
res_net_block_4: ResNetBlock | input: (None, 64, 64, 256) | output: (None, 64, 64, 256)
res_net_block_5: ResNetBlock | input: (None, 64, 64, 256) | output: (None, 64, 64, 256)
conv2d_transpose: Conv2DTranspose | input: (None, 64, 64, 256) | output: (None, 128, 128, 128)
batch_normalization_15: BatchNormalization | input: (None, 128, 128, 128) | output: (None, 128, 128, 128)
re_lu_9: ReLU | input: (None, 128, 128, 128) | output: (None, 128, 128, 128)
conv2d_transpose_1: Conv2DTranspose | input: (None, 128, 128, 128) | output: (None, 256, 256, 64)
batch_normalization_16: BatchNormalization | input: (None, 256, 256, 64) | output: (None, 256, 256, 64)
re_lu_10: ReLU | input: (None, 256, 256, 64) | output: (None, 256, 256, 64)
reflection_pad2d_13: ReflectionPad2d | input: (None, 256, 256, 64) | output: (None, 262, 262, 64)
conv2d_15: Conv2D | input: (None, 262, 262, 64) | output: (None, 256, 256, 3)

Discriminator



Results

Problem: model discriminator ended up memorizing the dataset and the generator did not converge on an acceptable result, due to the size of the dataset.