Style Transfer with Neraul netwrok in Game

Peng Huailiang

2019-05-07

Abstract

With the fast development of AI, the model implemented with tensorflow is becoming more and more popular. Nowdays, we achieve a model trained in python tensorflow environment, and use same algorithm implemented with compute shader in unity environment. The model is used to realtime transfer image style.

1 Introduce

We build the model with GAN, while the generator is composed of Auto-Encoder. We build double models for different devices. Due to limited performance, behaviour is also different at smart phone and PC. We distribute a release version at github with here. You can preview effect as the picture:

Figure 1: different style picture achieved with tensorflow and compute shader

As the picture showed, the left is origin input picture, middle is unity transfer picture, and right is tensorflow inference picture.

2 Model

The model that we used is based on GAN as metioned by Sanakoyeu *at al* [1]. at their paper.

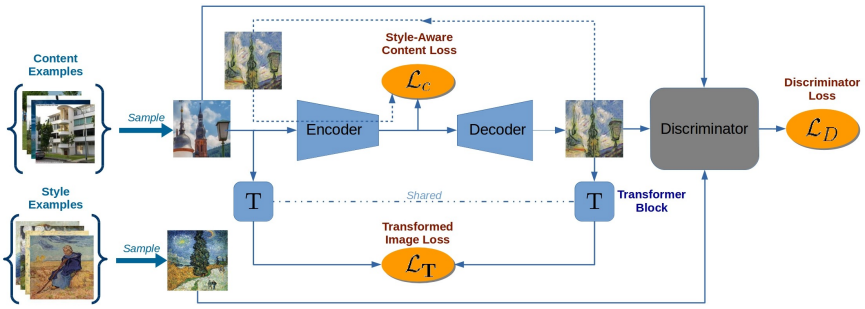

Figure 2: GAN mdoel metioned by Sanakoyeu at al

The model is composed of encoder, decoder and discriminator, while discriminator is just appeared in train duration.

The encoder is composed 6 layers, with depth is more lager and width and height is more smaller. We sample the every layer with CNN, then batch normal the cnn's output. We define all sample kernel as 3X3, and stride is setted as 2.

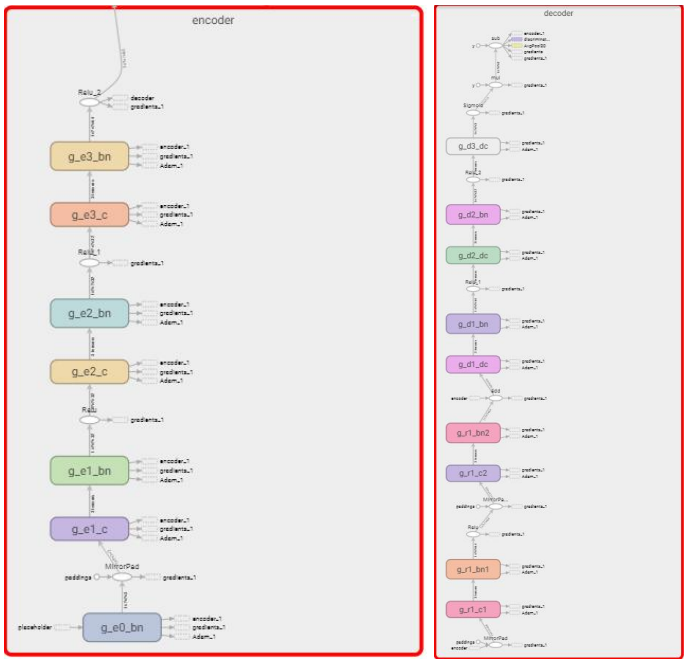

Figure 3: encoder and decoder for generator generated by tensorboard

The decoder is reversed with encoder. with the depth is more and more smaller, while width and height is become more and more larger. the depth is 3 at last, which stands for channels for picture's RGB.

Between of encoder and decoder, we use residual block for image recognition, we use that blocks to solved degradation problem for deep netwrok. The paper was writen by Kaiming *at al* [2].

In the discriminator, we define image loss as mean sequre between layer's avg-pooling, and feature loss as abs between result that origin styled panting and decoder's output.

3 Train

The trained dataset is microsoft coco dataset, while styled dataset is provided with *uni-heidelberg.de*, which can be download at here.

We trained at tensorflow environment with *Geforce GTX 1060 3GB*. It takes about 20 hours for 300000 iteration. We show the changes of kinds of netwrok argument in the picture.

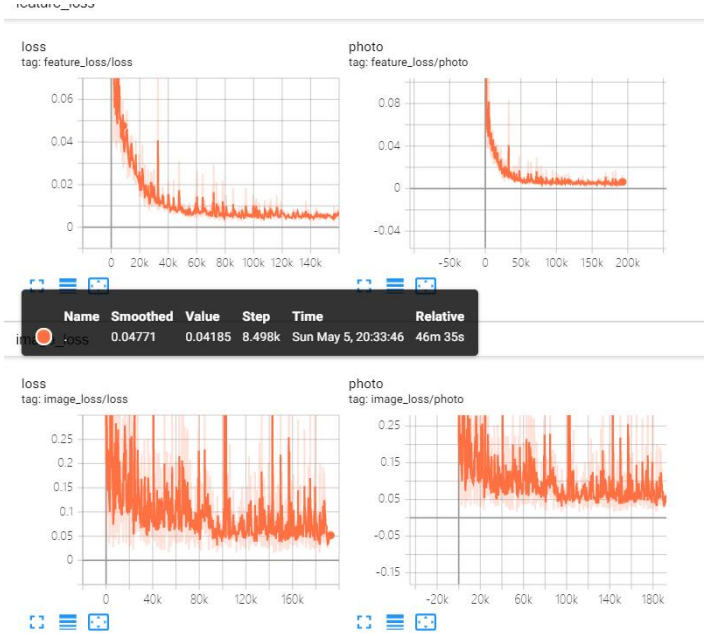

Figure 4: loss changes during training

4 Runtime

In the unity environment, we achieve a method with same algorithm, which has same effect on tensorflow. To speed up performance, we use compute shader to calculate while is running on the GPU. By the way, we did't achieves discriminator, because discriminator is not used during generating style picture.

Due to limited grammar of compute shader, we need to define groups while are composed of three dimensions threads. The threads have to be multiple of 64 or 32, and their counts have to less than 1024 in CS5, while less than 512 in CS4. so we design our model as satisfied the rule as possible to make more performance.

We design the size as 3X3 for the whole of the kernel. therefore, we design the data structure Matix3x3.cs for convolution. The buffer stored with matrix3x3 is delivered to GPU from CPU once-only.

5 Export Data

Tensorflow has its self data format saved as protobuf. Google also provids a series of API to iterate all variables defined in the checkpoint file. The tensorflow's checkpoint file stored argument is greater than 1G. The file is too large if we export that data to unity environment. When we iterate all tensor in the checkoint file, we found many tensors are unuseful. We write a tool that can export data and filter the datas that only are used in training duration, say, Adam's data at every layer.

The exported data is serialized as binary format defined by ourself. We trim some additional data such as tags data recorded in protobuf. The binary file will be parsed in csharp by the protocol defined in python. You can export network argument datas in checkpoint file, and also export session data for profile.

6 Profile

We define kinds of functions called printf and printh. We achieve those functions both in csharp and python environment. These functions are used to output different dimensions data to console. printf function is to watch second and third dimensions by fix first dimension. printh function is to watch first and second dimension by fix third dimension.

In order to view every layer clearly, we write a tool that can apply session's buffer data to a texture. We attach active function that is sigmoid to normalize every pixel to range 0-1. And draw the texture that use handled data.

As Shown in the following figure, you can drag slider easyly to view different layer with different depth.

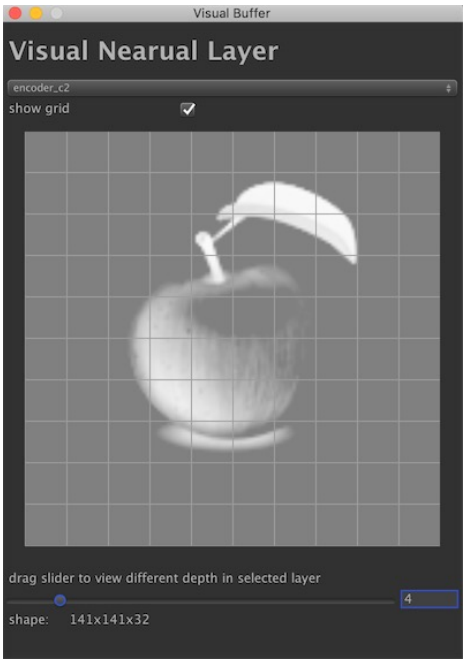

Figure 5: visual kinds of netwrok layer in unity environment

7 Performance

1. argument capacity

   The tensorflow's checkpoint file stored argument is greater than 1G, while we export binary file is only 670K. We traim a large number of unuseful data that not use during runtime.

2. frame per second

   As we test on PC with GPU Geforcce GTX 1060 3GB, complex model just run 3FPS, while the simpled model can run 29FPS. Compare to complex model, we trim layers that is deeped and unimport which is need complex calculation.

References

[1] Sabine Lang Artsiom Sanakoyeu, Dmytro Kotovenko. A style-aware content loss for real-time hd style transfer. 20(S2), 2018.

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. 20(S2), 2015.