Documentation

Of Neural Network Post Processing

V1.0

By YDMA

TABLE OF CONTENT

1. ALGORITHM

1.1 ARTIFICIAL NEURAL NETWORK

The neural network is a type of machine learning algorithms. It is based on a collection of connected units or nodes called artificial neurons, which loosely model the neurons in a biological brain. Each connection can transmit a signal from one artificial neuron to another. An artificial neuron that receives a signal can process it and then signal additional artificial neurons connected to it.[1]

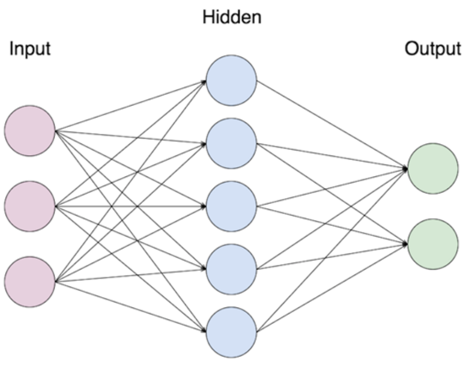


Figure 1-1 Example of Artifitial Neural Network with 1 layer

1.2 FAST NEURAL STYLE TRANSFER

Fast Neural Style Transfer (Johnson , Alahi, & Li , 2016) propose to use neural network to solve Neural Style Transfer. Compared with the original algorithm (Gatys , Ecker, & Bethge , 2016) , this paper uses the same loss function, but trains a feed-forward network to do the image translation, instead of treating it as an optimization problem, leading to way more faster processing speed.

[1] https://en.wikipedia.org/wiki/Artificial_neural_network

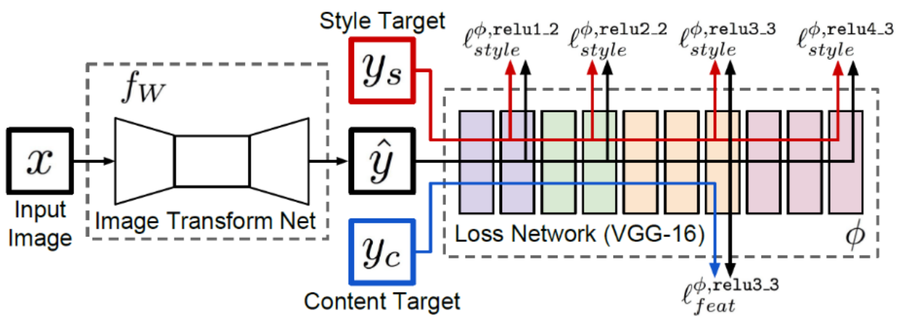

Style Target
$\ell_{style}^{\phi,relu1\_2}$
$\ell_{style}^{\phi,relu2\_2}$
$\ell_{style}^{\phi,relu3\_3}$
$\ell_{style}^{\phi,relu4\_3}$
y_s
f_W
x
$\hat{y}$
Input Image
Image Transform Net
y_c
Loss Network (VGG-16)
ϕ
$\ell_{feat}^{\phi,relu3\_3}$
Content Target

Figure 1-2 Fast Neural Style Transfer's Architecture

NNPP implements this image transform net for real-time post processing. However, the original transform net proposed by Johnson et al is more complex. NNPP has simplified the net to make it run faster on realtime.

Layer	Activation Size
Input	3 x W x H
32 x 9 x 9 conv, stride 1	32 x W x H
64 x 3 x 3 conv, stride 2	64 x W/2 x H/2
128 x 3 x 3 conv, stride 2	128 x W/4 x H/4
Residual Block, 128 filter	128 x W/4 x H/4
Residual Block, 128 filter	128 x W/4 x H/4
Residual Block, 128 filter	128 x W/4 x H/4
Residual Block, 128 filter	128 x W/4 x H/4
Residual Block, 128 filter	128 x W/4 x H/4
64 x 3 x 3 conv, stride 1/2	64 x W/2 x H/2
32 x 3 x 3 conv, stride 1/2	32 x W x H
3 x 9 x 9 conv, stride 1	3 x W x H

Table 1-1 FNST's Net Architecture

Layer	Activation Size
Input	3 x W x H
8 x 9 x 9 conv, stride 1	8 x W x H
16 x 3 x 3 conv, stride 2	16 x W/2 x H/2
32 x 3 x 3 conv, stride 2	32 x W/4 x H/4
Residual Block, 32 filter	32 x W/4 x H/4
Residual Block, 32 filter	32 x W/4 x H/4
16 x 3 x 3 conv, stride 1/2	16 x W/2 x H/2
8 x 3 x 3 conv, stride 1/2	8 x W x H
3 x 9 x 9 conv, stride 1	3 x W x H

Table 1-2 NNPP's Net Architecture

This package contains 7 pre-trained models with different style. All the models are trained offline with Keras, if you are interested in training your own model, please contact author.

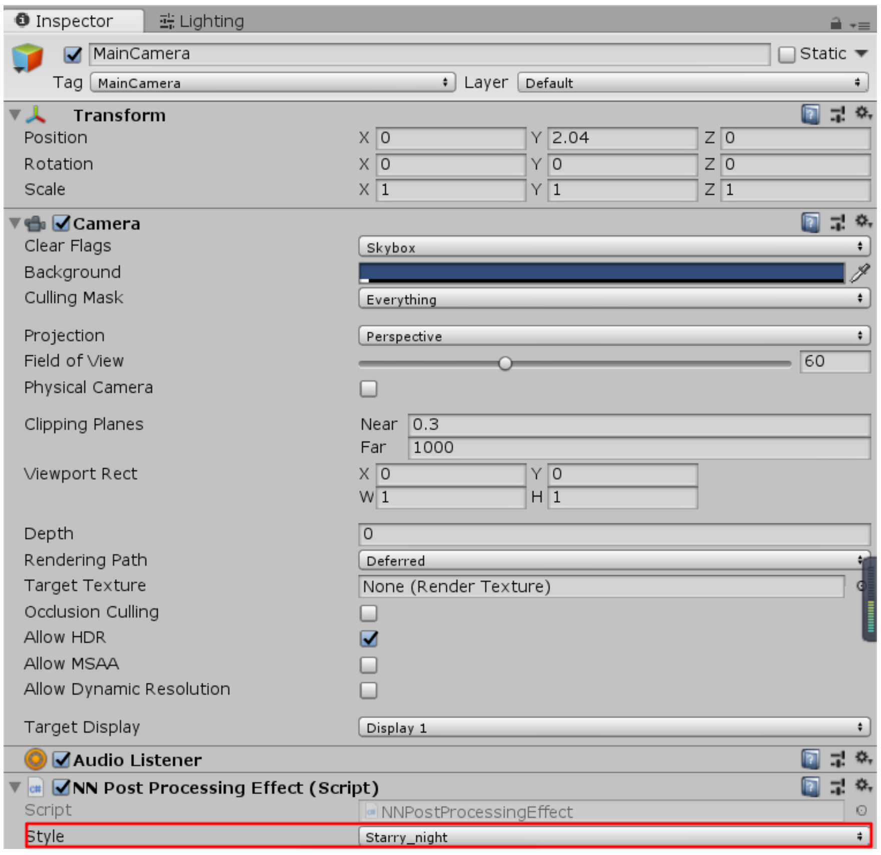
1.3 IMPLEMENT WITH COMPUTE

Most deep learning frameworks such as Pytorch and Tensorflow accelerate the forward-propagation of neural network with the help of CUDA computing platform. However, in NNPP, we implement the forward-propogation process with Compute Shader. That is the reason why this package require DX11+/Vulkan/Metal support. We found that our implemetation is equal or a little faster than pytorch or tensorflow's processing speed with the same model and resolution.

2. HOW TO USE

2.1 SAMPLE SCENE

Open the scene at "**Assets/NNPostProcessing/DemoScene/Scene**", find the NNPostProcessingEffect script under MainCamera, select the style in dropdown menu.



MainCamera ☐ Static
Tag MainCamera Layer Default

Transform

Position	X 0	Y 2.04	Z 0
Rotation	X 0	Y 0	Z 0
Scale	X 1	Y 1	Z 1

Camera

Clear Flags	Skybox
Background	
Culling Mask	Everything
Projection	Perspective
Field of View	60
Physical Camera	☐
Clipping Planes	Near 0.3
	Far 1000
Viewport Rect	X 0 Y 0
	W 1 H 1
Depth	0
Rendering Path	Deferred
Target Texture	None (Render Texture)
Occlusion Culling	☐
Allow HDR	✓
Allow MSAA	☐
Allow Dynamic Resolution	☐
Target Display	Display 1

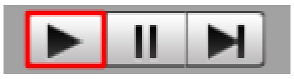
Audio Listener

NN Post Processing Effect (Script)

Script	NNPostProcessingEffect
Style	Starry_night

Figure 2-1 Select Style Model

Click play button and see it works.



2.2 For Your Project

Add NNPostProcessingEffect script under your MainCamera, select the style, and run!

2.3 For Assetbundle

All model is in "**Assets/NNPostProcessing/Resources/Model**" folder.

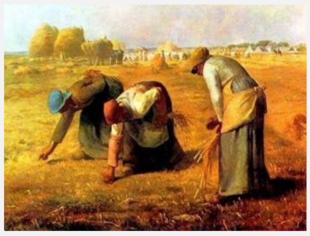
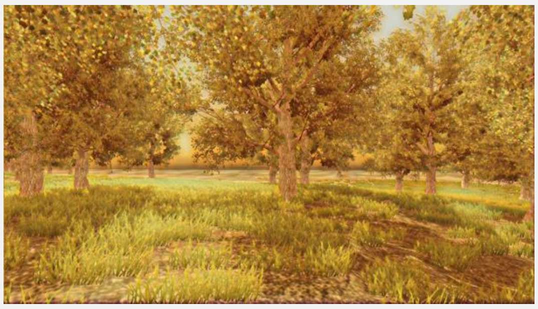
The demo scene use Resource folder to load model, if you would like to put the model in Assetbundle, please modify following code to make it works with Assetbundle.

The code to load model is at line 24 at NNModel.cs

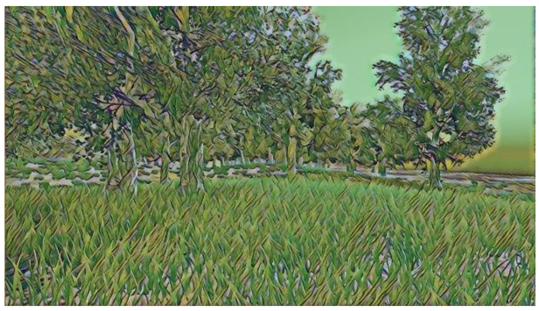
```
public void Load(string name)
{
    TextAsset text = Resources.Load<TextAsset>("Model/" + name);
    NNModelSerialize modelSerialize = JsonUtility.FromJson<NNModelSerialize>(text.text);
    ......
```

2.4 Model Library

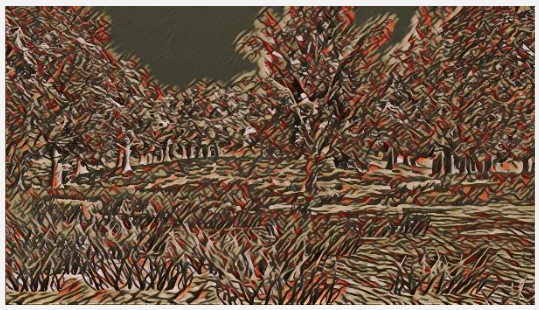
This package contains 7 pretrained models.

Model Name	**Style Image**	**Effect**
des_glaneuses ,		

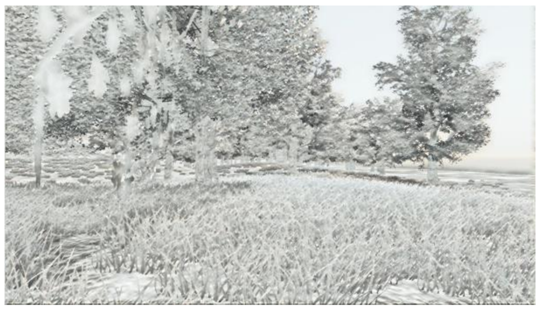
la_muse



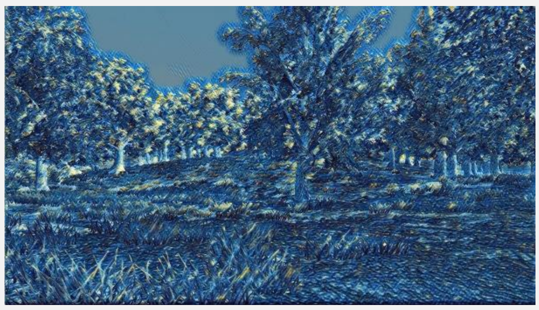
mirror



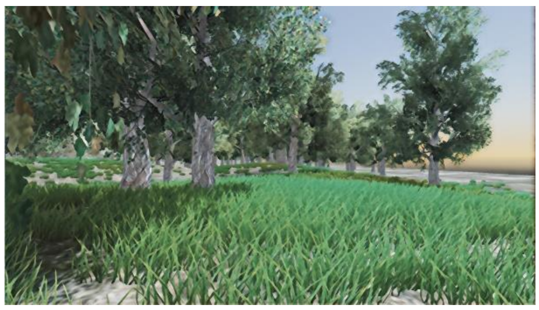
sketch



starry_night



udnie



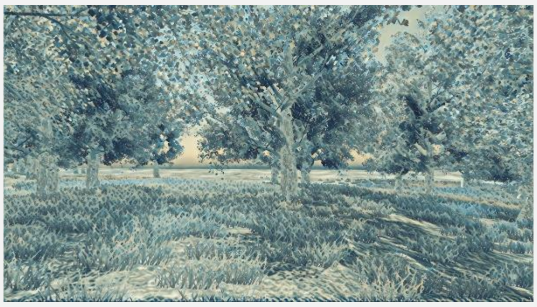
wave		

Table 2-1 All Pretrained Models

3. Profile

3.1 Runtime Speed

RESOLUTION	RTX 2070 GPU	M1000M GPU
256X256	~3ms	~40ms
720X480	~23ms	~220ms
1920X1080	~190ms	~1400ms

Table 3-1 Time to process one frame

Experiment on RTX 2070 GPU and M1000m GPU with different resolution. We can achieve 30FPS on 2070 with 720p.

3.2 Speed Comparation with Keras

Speed to process one frame. Experiment on one RTX 2070 GPU

RESOLUTION	KERAS+TENSORFLOW	NNPP
256X256	~10ms	~3ms
720X480	~39ms	~23ms
1920X1080	~250ms	~190ms

Table 3-2 Speed comparation with keras

Our implementation is 1-2x faster than keras with tensorflow backend implemation.

4. REFERENCE

Gatys Leon A., EckerAlexander S., & Bethge Matthias . (2016). Image style transfer using convolutional neural networks. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.

Johnson Justin, AlahiAlexandre , & Li Fei-Fei. (2016). Perceptual losses for real-time style transfer and super-resolution. European Conference on Computer Vision. Springer.

Contact of Author:

Email: hello_myd@126.com

Website: www.ma-yidong.com