CSCI 5210 Advanced Topics in Computer Graphics and Visualization

**Report of Assignment -- Colorization**

Gao Haoran 1155047156

1. Development Environment
   1. Operating System: Windows 10 (my laptop)
      1. Deep learning framework: TensorFlow 1.4
      2. Python: Python 3.5.4 :: Anaconda, Inc.
      3. GPU: NVIDIA GeForce GTX 1050
      4. CUDA Toolkit 8.0
   2. Operating System: Linux (GPU VM of CSE department)
      1. Deep learning framework: TensorFlow 1.3
      2. Python: Python 2.7.5
      3. GPU: GeForce GTX TITAN X
2. Overall Structure
   1. Image Preprocessing
      1. Decoding
      2. Resizing
      3. Converting color space: RGB to LAB
      4. Splitting LAB to get inputs and labels
      5. Normalizing labels
      6. Labelling gray images
   2. Build Dataset
      1. Mapping, filtering, shuffling, batching
      2. Iterator
   3. Estimator
      1. Deep learning model
      2. Training
      3. Prediction
      4. Evaluation
      5. Visualization (TensorBoard)
3. Hyperparameters and Computation Times
4. Experimental Details
5. Submit Files
6. Reference

**1. Development Environment**

**2. Overall Structure**

**2.a. Image Preprocessing**

Preprocess the image data for better performance of the Colorization Model.

2.a.i. Decoding

Decod the image string from the image file.

2.a.ii. Resizing

Resize the image with size [224,224] and [112,112] for different usage.

2.a.iii. Converting color space: RGB to LAB

Convert the color space from RGB to LAB.

2.a.iv. Splitting LAB to get inputs and labels

Split the LAB image array, L\* channel as input, a\*b\* channels as label.

2.a.v. Normalizing labels

Normalize the a\*b\* channels by using sigmoid function to evenly distribute the value of lab\_ab in range (0, 1).

2.a.vi. Labelling gray images

Label gray images, in which maximum value < 1 and minimum value > -1, as not valid image. The "valid" label will be used later in filtering of dataset.

**2.b. Build Dataset**

In TensorFlow, a Dataset can be used to represent an input pipeline as a collection of elements (nested structures of tensors) and a "logical plan" of transformations that act on those elements.

2.b.i. Mapping, filtering, shuffling, batching

Map the image path list to real image data stream by mapping function (image preprocessing);

Filter out invalid images (gray images);

Shuffle the input image data randomly;

Feed data to Estimator batch by batch.

2.b.ii. Iterator

Create a One Shot Iterator for iteration of Estimator.

**2.c. Estimator**

The Estimator object wraps a model which is specified by a model\_fn, which, given inputs and a number of other parameters, returns the ops necessary to perform training, evaluation, or predictions.

2.c.i Deep learning model (model\_fn)

Build the leaning model according to the network structure on the paper. Simplify the global layers because of the memory limit of GPU.

Low-Level features network (Same as the paper)

Mid-Level features network (Same as the paper)

Global features network

Only one convolutional layer, received the output of Low-Level features network and

output the tensor with shape [batch\_size, 28, 28, 32]. Then transfer it to fully

connected layers same as the paper. And through the fusion layer, combining global

and local features.

Colorization network (Same as the paper)

2.c.ii Training

Loss function: MSE

Optimerizer: Adam

2.c.iii Prediction

Predict the colorful image with input gray image by trained model. Then output the result images and truth images into corresponding folder.

2.c.iv Evaluation

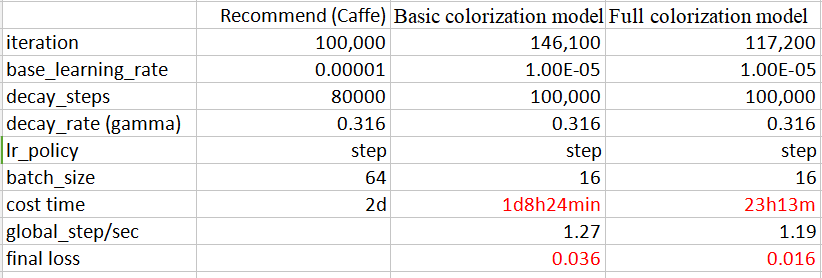
Evaluate the trained model with test dataset. Output the average loss.

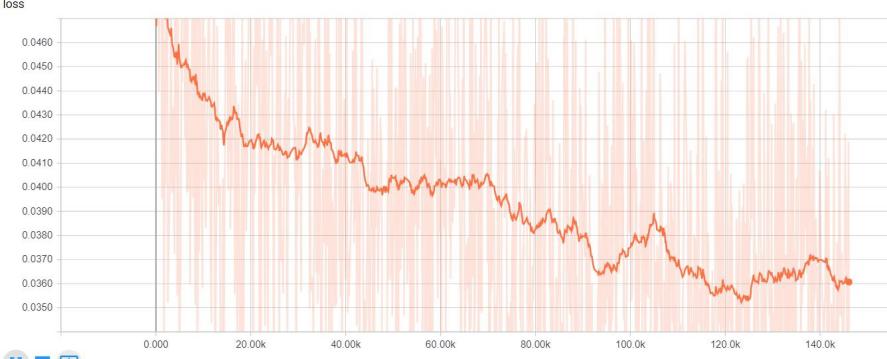
2.c.v Visualization (TensorBoard)

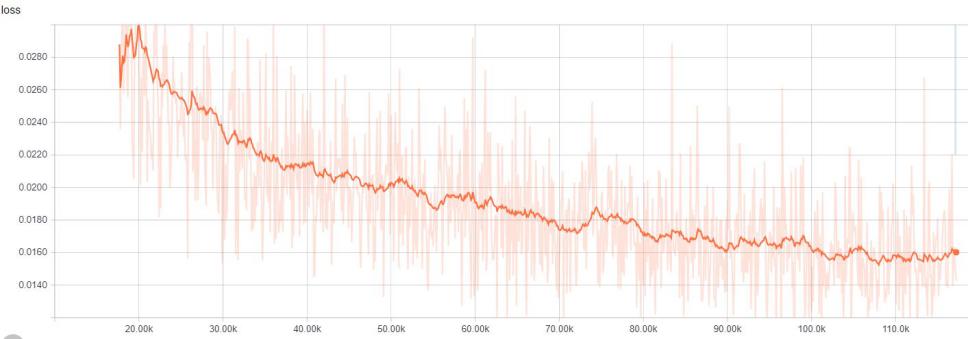
Visualize the gloabl\_step/sec and loss using graphs. Visualize the sample images (input gray images, truth images and predict images) during the training.

**3. Hyperparameters and Computation Times**

Comparsion between Basic colorization model and Full colorization model (with global network)



Basic colorization model

Full colorization model

Training Suggestions (Possible improvements):

Using full colorization model (color\_global.py)

Learning rate: 1e-5, Batch size: 64

Global network, one convolutional layer, [batch\_size, 28, 28, 32], change 32 to larger number

Train with both trian and test dataset

**4. Experimental Details**

**Background**

I chose Tensorflow as my framework because I get used to Windows system and only it has the Windows version. Moreover, I have tried to use Linux VM and found that it cannot work with GPU. It really took my much time to learn the Tensorflow, as I know nothing about this framework before. Plus, I had only a little knowledge of deep learning. But after a harsh learning procedure, I finally built a basic model of colorization.

**Two mistakes**

However, I made lots of mistakes during the implementation of the Colorization Model, which waste me much GPU resource. I did not fix all the bugs inside my program until one day before the deadline, so I did not have enough time to train the model, which is the reason why I need to submit the assignment one day after deadline.

One mistake is that the a\*b\* channels of LAB images are not normalized to range (0, 1). I wrote the normalization function for it but I did not check the correctness of this function. The other stupid mistake is that I used Sigmoid transfer layer for all the network layer and there should be only one Sigmoid transfer for last output network layer and other layers should all use ReLU transfer. Thanks to the tutor, who help me find out these two stupid mistakes, otherwise my assignment may fail.

**Two improvements**

During the debugging period, I also made some improvements. First, I implemented the global features network and the fusion layer to combine global and local features. Second, I removed the gray images in the stage of image preprocessing. These two improvements greatly facilitated my model. After implementing them, my model had a much better performance and its loss decreased faster, lower loss value within less training time.

**Training environment**

I mainly trained the model on my own laptop. Because it is too late when I found that GPU of the CSE department is available for students to apply and use. After I applied GPU and got the approval, there are only two days left. Even more unfortunately, the GPU resource of department is very limited due to the fierce competition among students. I only managed to use it for a very short period. And the configuration and environment are very different. so I modified and created a version for Tensorflow 1.3 with python 2.7.

**5. Submit Files**

color.py (basic colorization model, with clear comments)

color\_global.py (full colorization model, with clear comments, with usage)

upsample.py (TensorFlow upsampling)

pix2pix.py (TensorFlow rgb to lab)

color\_tf\_1\_3.py (TensorFlow 1.3 version)

color\_global\_tf\_1\_3.py (TensorFlow 1.3 version)

upsample\_tf\_1\_3.py (TensorFlow 1.3 version)

**6. Reference**

**Let there be Color!: Joint End-to-end Learning of Global and Local Image Priors for Automatic Image Colorization with Simultaneous Classification**

http://hi.cs.waseda.ac.jp/~iizuka/projects/colorization/data/colorization\_sig2016.pdf

**TensorFlow Upsampling (upsample.py)**

http://warmspringwinds.github.io/tensorflow/tf-slim/2016/11/22/upsampling-and-image-segmentation-with-tensorflow-and-tf-slim/

**TensorFlow RGB to LAB converting (pix2pix.py)**

https://github.com/affinelayer/pix2pix-tensorflow/blob/master/pix2pix.py