

CS838 Project Progress Report

Sek Cheong

May 1, 2017

1 Background

We proposed a project to use deep CNN to colorize black and white images automatically. We would explore the technique transfer learning to transfer the learned weights from VGG16 to build a new model for colorizing image. We would also compare the result using transfer learning verses training the model from scratch. The report describes the progress we have made so far in our proposed project. We also address the issues we have encountered and the remaining work towards the completion of the project.

2 Progress

We started out by finding the suitable data set for our colorizing model. We then did some research on digital image processing with specific area in color theory and image compression. We defined the model based on the image compress technique we learned to best suited for the task for colorizing.

2.1 Data Set

There are a number of data sets for computer vision available publicly. Among the available image data sets the ImageNet data set is the most famous one. Many machine learning models involving vision are trained on ImageNet images. There are approximately one million image files from 1000 categories in the ImageNet data set and about 120G in size. We downloaded (took really long time!) the entire image set and looked some images. It turned that there are many images either in black and white or were not

instead of simply resizing the images into 224×224 pixels, we scale the shorter side of the image to 224 pixels and crop along the with 224 pixels. We think training on images with the correct aspect ration might give a better result for coloring. The scaling and cropping was done by the python script [1].

To construct a training example from a color image we could convert the image into gray scale image and use the gray scale image as input \mathbf{X} , and the RGB values as the target \mathbf{Y} . Note, the input is a 1×224 vector and the target is a 3×224 vector. As explained by Ryan Dahl's web blog post [3] that we could convert the image from RGB color space into YUV color space, where Y is the intensity of the image, and UV are two chrominance components of the image. This reduce the target vector \mathbf{Y} to $2 \times 224 \times 224$. The following figure shows the original image, the Y component, the U component, and the V component.



Figure 2: The original picture and its YUV components

2.2 Model

We use the VGG16 model as our starting model described as shown in ?? Figure 3. The VGG16 has an output layer of 1000 units with Softmax activation. The output layer is designed in such a way that it can be used to classify image from a thousand categories. Our initial design of transfer learning was that after loading the weights for the model we remove the final layer and replace with a $2 \times 224 \times 224$ regression unit. For activation function we use the hyperbolic tangent function. This proved to be a disaster as the number of parameters in the model skyrocketed to 545,402,688! from 138,357,544.

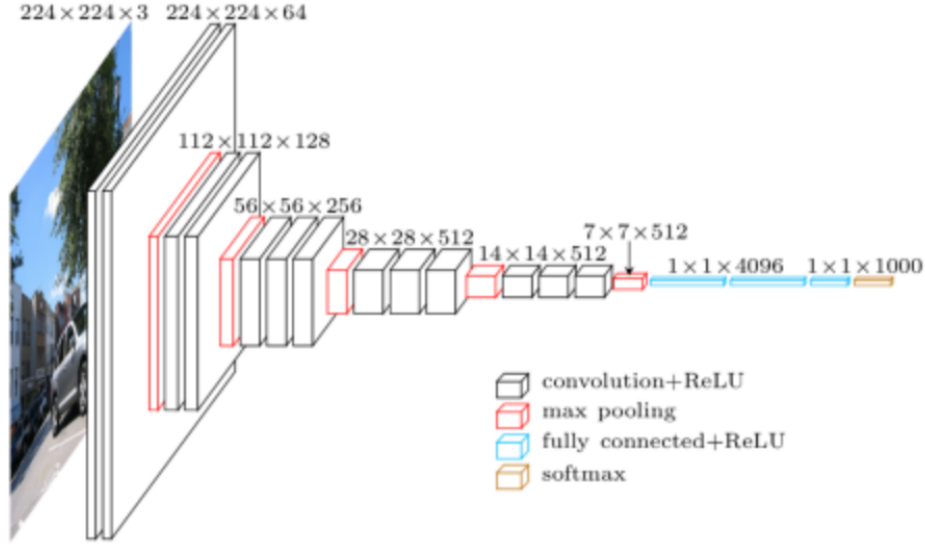


Figure 3: The VGG16 model visualization

It turns out that YUV color space is closely related YCbCr color space which was used for digital encoding of color information suited for video and still-image compression and transmission such as MPEG and JPEG [?]. For our use we could simply true UV as same as CbCr. The main difference between these two color space was scaling. Since we scale pixel values between -1 and 1 , it wouldn't make any difference for us. The nice property of YCbCr color space was that human eyes are far more sensitive to change in intensity than color. This allowed us to throw away most of the CbCr information while still maintain the image quality. The process of throwing away color information is called color space compression.

We experimented around different compression rate and it appeared that we can throw away 80% of the color information and the image still maintains very good color quality.



Figure 4: Image on the left with 90% color space compression

As show in the figure above, the image on the left is the original image, the image on the right has 90% of the color information removed.

Table 1: The colorize model (with color space compression) based on VGG16 architecture

Layer (type)	Output Shape	Parameters
input 1 (Input Layer)	(None, 224, 224, 3)	0
block 1 conv 1 (Conv 2D)	(None, 224, 224, 64)	1792
block 1 conv 2 (Conv 2D)	(None, 224, 224, 64)	36928
block 1 pool (Max Pooling 2D)	(None, 112, 112, 64)	0
block 2 conv 1 (Conv 2D)	(None, 112, 112, 128)	73856
block 2 conv 2 (Conv 2D)	(None, 112, 112, 128)	147584
block 2 pool (Max Pooling 2D)	(None, 56, 56, 128)	0
block 3 conv 1 (Conv 2D)	(None, 56, 56, 256)	295168
block 3 conv 2 (Conv 2D)	(None, 56, 56, 256)	590080
block 3 conv 3 (Conv 2D)	(None, 56, 56, 256)	590080
block 3 pool (Max Pooling 2D)	(None, 28, 28, 256)	0
block 4 conv 1 (Conv 2D)	(None, 28, 28, 512)	1180160
block 4 conv 2 (Conv 2D)	(None, 28, 28, 512)	2359808
block 4 conv 3 (Conv 2D)	(None, 28, 28, 512)	2359808
block 4 pool (Max Pooling 2D)	(None, 14, 14, 512)	0
block 5 conv 1 (Conv 2D)	(None, 14, 14, 512)	2359808
block 5 conv 2 (Conv 2D)	(None, 14, 14, 512)	2359808
block 5 conv 3 (Conv 2D)	(None, 14, 14, 512)	2359808
block 5 pool (Max Pooling 2D)	(None, 7, 7, 512)	0
flatten (Flatten)	(None, 25088)	0
fc 1 (Dense)	(None, 4096)	102764544
fc 2 (Dense)	(None, 4096)	16781312
colorize (Dense)	(None, 2178)	8923266

We discarded 85% of the color information, that is we resize the chrominance components UV to 15% of its original size using bi-cubic interpolation. This process reduced the UV size down to 33×33 from 224×224 . We adjusted the output layer to the size of reduced color dimension and the parameters went from 545, 402, 688 to 143, 183, 810.

2.3 Training

We trained our model on 1500 images with 50 epochs, the training set is hardly large enough to train a complex task such as colorization, but due to the limited computing resources (more on that in the Challenges section) and time constraint, we had to settle with 1500 samples. The training process took about 25 hours to complete.

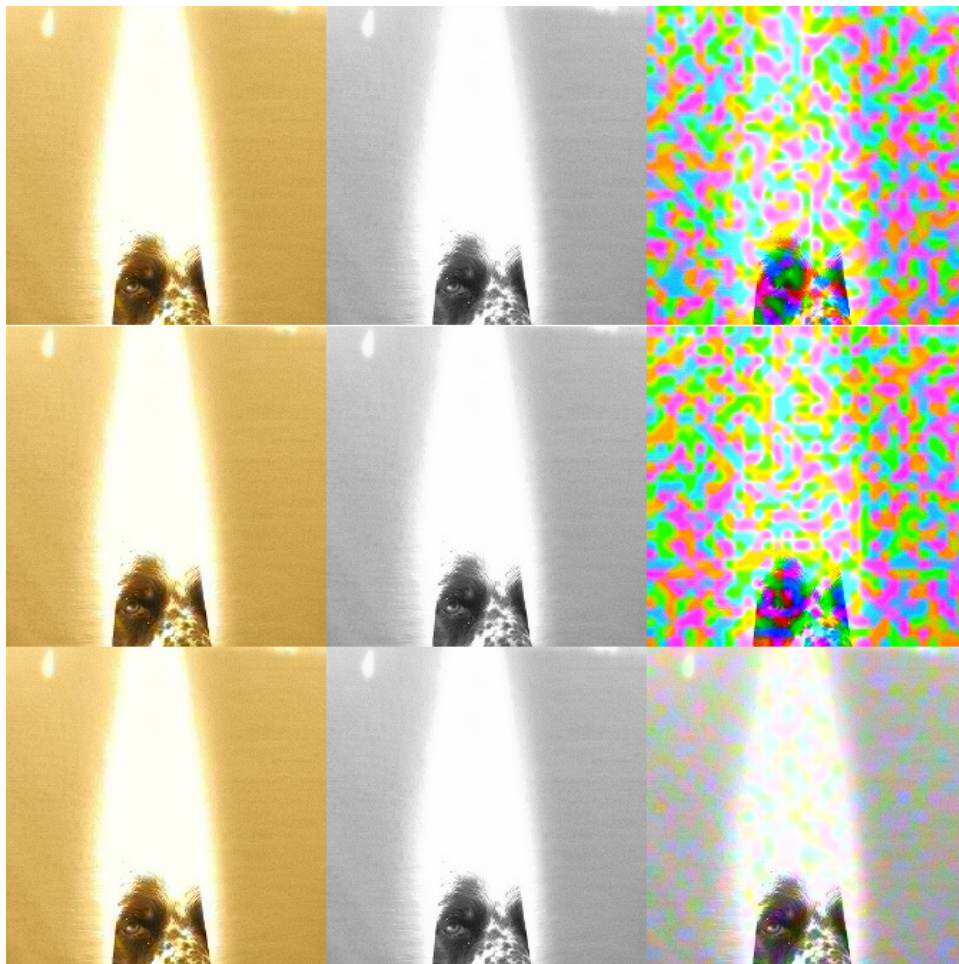


Figure 5: Predictions in various epoch, left the ground truth, middle the input image, right the predicted image

3 Challenges

GPU does work!

4 Remaining Tasks

We use VGG16 model as a starting point and forward an image into the VGG16 model, we then extract a few layers from the VGG model and down-scale the image to match the extracted layers. The extracted layers should contain useful information of the image. We will construct a model based on the extracted layers to colorize images.

Our model is basically a learned function that takes the gray level pixels as input and maps them into RGB color pixels. It is not possible to achieve the perfect mapping from gray level images to color images, as the conversion from color images to gray level images is not a one-to-one correspondent function.

We choose the Keras framework with Tensorflow backend for the experiment. We choose Keras due to its user friendliness and the ability to use GPU as the computing core. The experiment will be carried out on a computer equipped with a 3.7GHz Intel quad core processor, 16G of memory and nVidia GeForce GTX 660 graphics card with 2G memory.

There is a pre-trained VGG16 model for Keras created by Francois Chollet [2] from github, we plan to use this model and extend it to coloring images.

We split out constructed data set into the training, tune, and test set. We use a MSE of the each pixel of the colorized image and the true color image as a measure of accuracy.

References

- [1] CHEONG, S. Python script for preprocessing the image data. GitHub source code <https://github.com/sekcheong/colorize/blob/master/src/preprocess.py>.
- [2] CHOLLET, F. Vgg16 model for keras. The trained model can be downloaded from GitHub <https://github.com/fchollet/deep-learning-models>.

- [3] DAHL, R. Automatic colorization. <http://tinyclouds.org/colorize/>.
- [4] M.J. HUISKES, M. L. The mir flickr retrieval evaluation. *ACM International Conference on Multimedia Information Retrieval* (2008).