Creating Shareable Instagram-like Image Filters

Agarwal, Varun
varagarw@indiana.edu

Xiang, Zhongkun
xiangz@indiana.edu

Sogathur Govindan, Kamalanathan
ksogathu@indiana.edu

1 Introduction

It is common for people to click pictures of themselves, or let others click their pictures that could potentially be shared online with friends on various social media platforms. Naturally, it follows that one would like to look as good as possible in these pictures. There are multiple editing options at a user's discretion both within and outside the platform where a person might want to share a picture. A popular feature that photo sharing applications like Instagram offer are image filters which allow a user to apply a certain effect on a picture before sharing. The idea is to allow the user to choose from a set of filters and use on their image that they might judge to be a suitable modification before sharing with their friends. Although simple, the limitation of this feature is the finite number of pre-defined filters that a user may or may not like. The plausibility of the user to make custom changes to an image as they see fit is removed by this limitation. Outside of the platform itself, there are multiple options for a person to modify their image by making tedious changes in the image, e.g. photoshop. Learning combinations of such edits as a mathematical model, in order to be able to re-use and share it on image sharing applications such as Instagram is investigated in this work. Neural Networks are implemented to learn the entire modification and the model is saved along with the learned weights as a single filter. In this report, we will only focus on image transformations that are independent of each pixel's adjacent pixels.

2 How Do Image Filters Work?

To understand what a filter really does to an image, we can look at the way the pixels' values of an image change when a filter is applied. The plots in figure 1 and figure 2 aim to shed some light on the change. Figure 1 shows the transformation in the HSV values of the image when the filter 'Hefe' is applied on it. HSV is a different space of cognition used to represent an image similar to RGB values. The transformation for RGB values when the same image is represented in RGB space is shown in figure 2.

Figure 1: Transformations in HSV Space

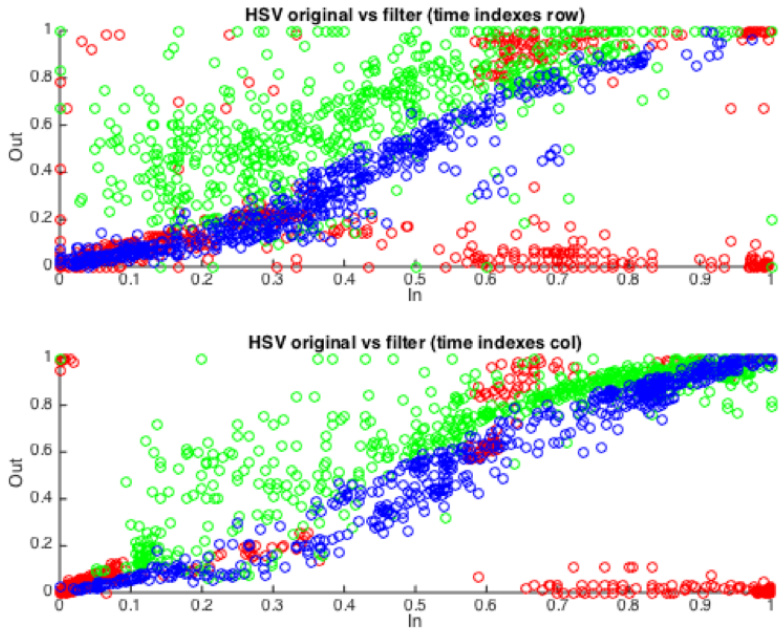


Figure 2: Transformations in the RGB Space

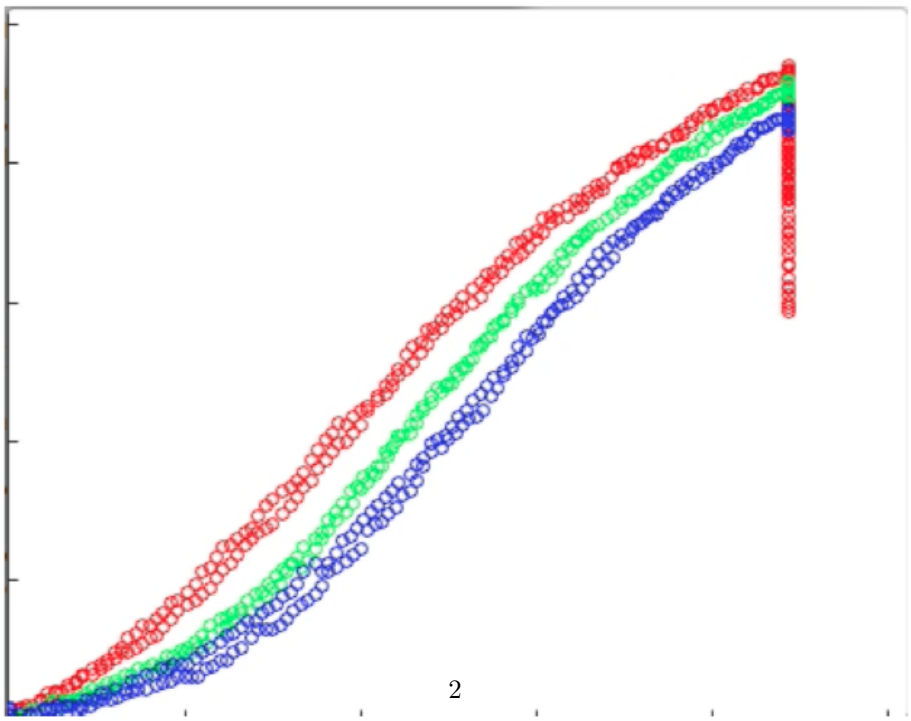


Figure 2 [Hoard, 2014] shows how image filters work at pixel level. The "Hefe" filter changes each pixel's R,G and B values based on its position in the image. As the non-linear transformation for a given position in the case of RGB space is more consistent than the one in HSV space, we decided to choose the RGB space for making our transformations to train and learn the edits as image filters.

3 Previous Work

As part of the literature survey for the project, many papers in the field were reviewed. One of the first papers was a research paper by Corey Hoard on Reverse Engineering Instagram's Hefe Filter [1]. This work provided the basic background required to construct image filters from scratch, and that learning such a transformation using neural networks was a viable approach. Luan, et al, in their paper Deep Photo Style Transfer [2] provided methods to constrain the photostyle transfer to confine the output to be in locally affine colorspace. This approach was used to ascertain the possibility of transforming the RGB colorspace to a new RGB color space defined also by the position of the pixels in the image. Geoffery, et al, in their paper Learning to Represent Spatial Transformations with Factored Higher-Order Boltzmann Machines [3] showed that spatial transformations on images can lead to changes in image in the target set which was performed on the content of the image, rather than on the intrinsic values of the image - like the color, brightness, etc of each pixel in the image. This was also input for the proceedings of the method in this paper. A major work in this area that is most relevant to our presentation in the paper was done by Colton, et al, in their paper Evolving Approximate Image Filters [4]. The feasibility of our current approach was investigated in their work wherein tree based representations for filters were described. The improvements and suggestions mentioned in their work overlaps with the methods we employed in our investigation.

4 Test Set Generation

We train two neural networks on two distinct datasets. The first data set consists of actual images, pixels of which would be the input to the first neural network we train. For every image in the dataset, there is another image which is the same but with the 'Earlybird' filter applied on it [5] , pixels of which would act as the target value of the neural network. The second data set is an artificial image data set that we generated to train the second neural network. We created 1280 images of dimension 240 x 240 with random values assigned to each of the pixels. Similarly, for every image in this dataset, there is another image which is the same but with the Earlybird filter applied on them. The individual pixels of these images act as target values of the model. The input to both neural networks is a 5 dimensional pixel whose constituents are the R,G,B

pixel values and the relative position of the pixel in the image. For a given image that is uploaded by a user on the platform, a synthetic set of test images with the same size as the one uploaded is generated. To do this, integer values are randomly chosen for each pixel in each channel from [0, 255]. Once such a test set is generated, these images can be treated as real meaningful images to train the network on. The foremost reason why such an image set is valid is that it is representative of all the images that can possibly exist for a given size. For example, if we were to create an image of size 240 x 240, it would be one out of $((256)^{240})^3$ possible images. This covers every possible image of the size 240 x 240. Thus, it is valid to assume that randomly choosing pixel values independent of other pixels would create a representative dataset, generalizing the model to images that haven't been seen before. It is noteworthy that the validity of the synthetic data set would hold only when transformations of the pixels in the image are independent of the adjacent pixels. The dataset is split into training and test datasets, where 80 percent of data will be used as training set and 20 percent will be the test set.

5 Proposed Method

Each pixel in the training input images can be turned into a vector of the form [R,G,B,X,Y]. The neural network will learn to map each vector [R,G,B,X,Y] to a new vector [R_new, G_new, B_new], where its X,Y (e.g. its position in the image) remains the same. Thus, we can think of image filters, as learned by the neural network, as a nonlinear function that takes the original pixel (vector) [R,G,B,X,Y] as input, and the output will be the filtered pixel in the form of a vector[R_new, G_new, B_new] at the same position. Therefore, the task is that of a function approximation, that an image filter will be a complex nonlinear function. The two architectures of neural networks that were considered for implementing the idea were a deep neural network with fully connected layers, and a convolutional neural network. Even though convolutional neural networks are known to work better with images, we will not be using a convolution neural network in the context of this report, because our focus is on transformations that are independent of adjacent pixels. So, using a convolutional network would only add to the complexity of the model. Based on this idea, we will be using a deep neural network with 2 hidden fully-connected layers to learn the non-linear transformation. The first layer has 12 neurons with rectified linear units as activation function, the second layer has 8 neurons with rectified linear units as activation function as well. the output layer has 3 neurons with softmax as activation function.

Details of the Neural Network and Architecture:

The first layer or the input layer has five neurons: three corresponding to the R, G, B values of each pixel and 2 corresponding to the x, y values, which are the relative positions of the pixel. The next two layers are the training layers with 12 and 8 neurons respectively. An ideal architecture would generalize the result

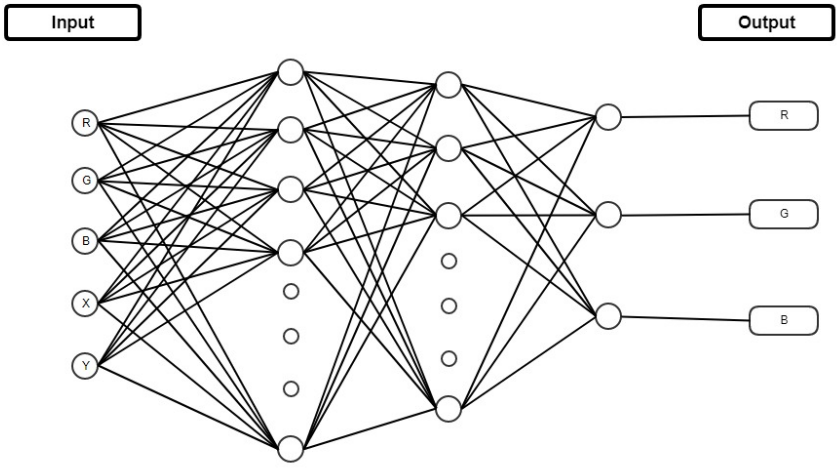


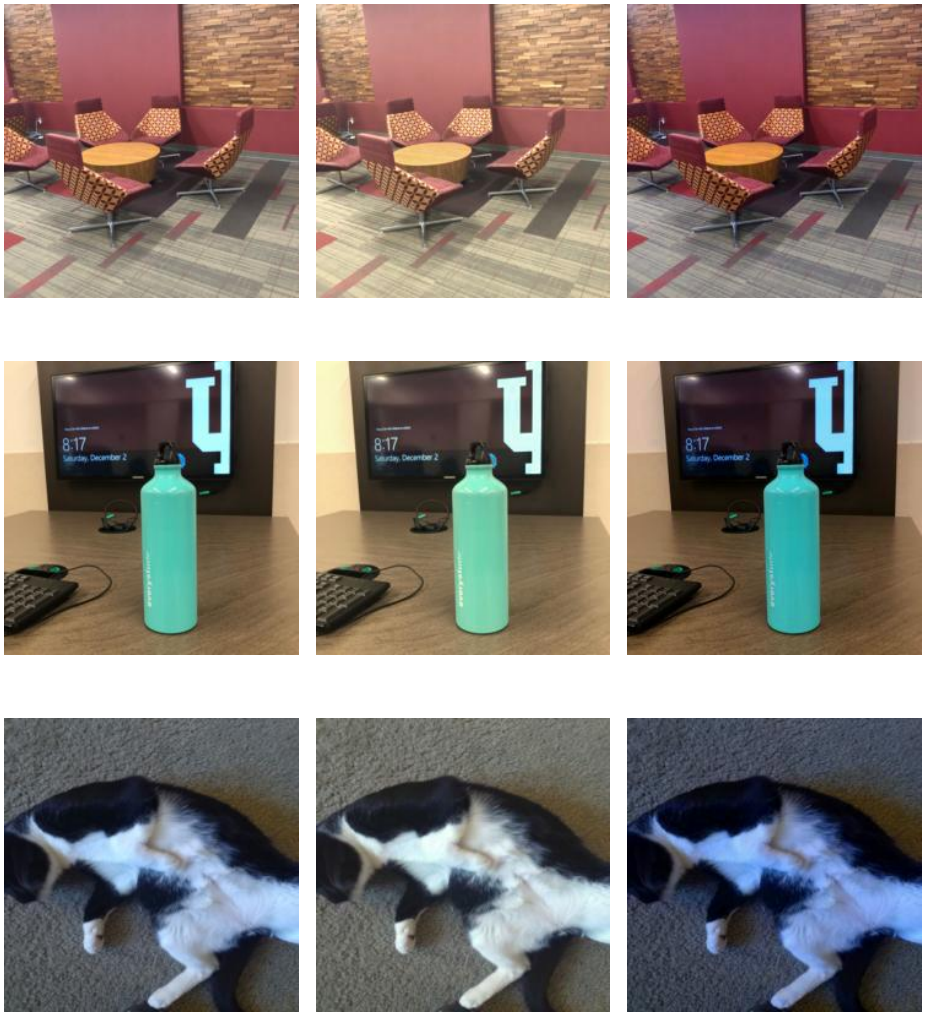
Figure 3: Neural Network Architecture

of any number of edits, irrespective of the order, into a single filter in terms of the model that would be exported as a filter. The choice of 12 and 8 neurons in the two hidden layers was arrived at after arbitrarily choosing the number of neurons for each layer in the beginning and altering the number with each test run for 10 epochs. 12 and 8 neurons in each layers appeared to generalize well for the individual filters applied as a set of transformations to each image. The results shown are for the Instagram filter 'Earlybird' (as implemented in [7], which the model has learned).

6 Results

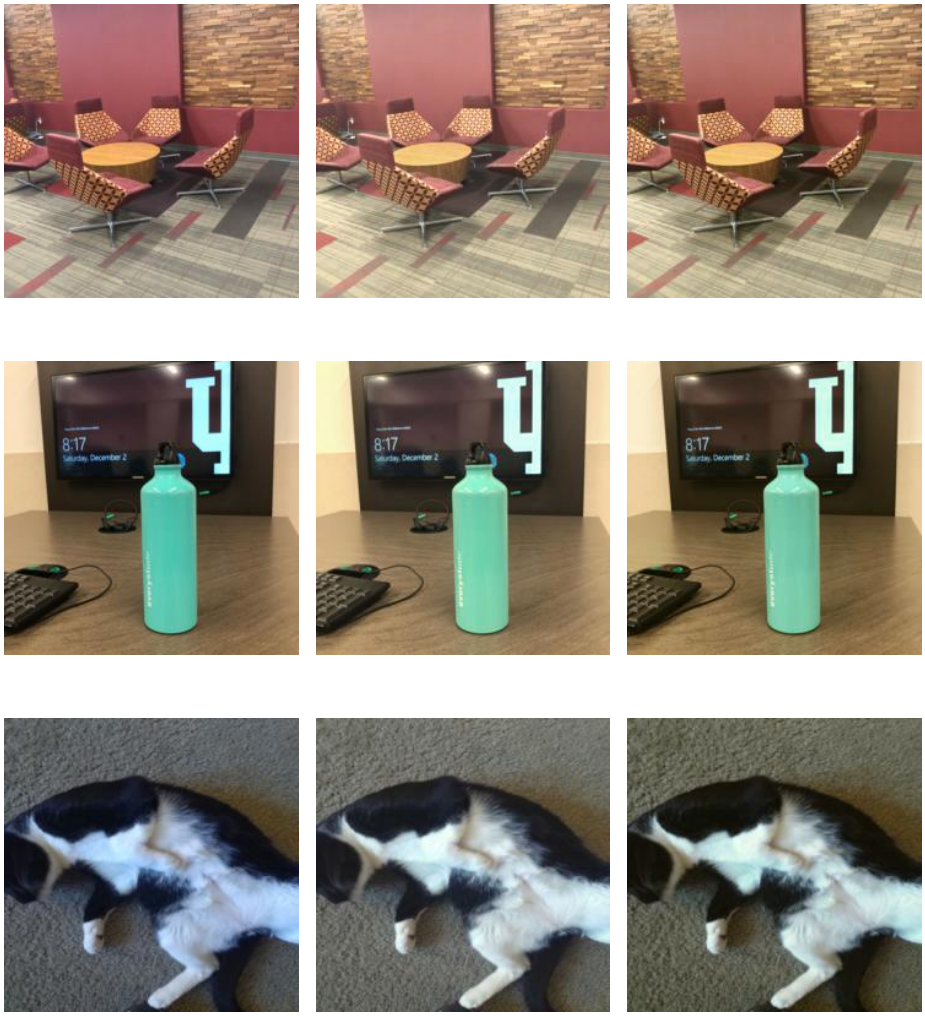
We trained our neural networks with mean square error as the loss function and the 'Adam' optimizer on Big Red 2 supercomputer. The neural network with real images as input took approximately 30 minutes to complete 30 epochs with loss of $2.7828 * 10^{-4}$, which is relatively low. We saved the model and applied the model on 5 images we took that are not part of the training set. The results for 3 of these pictures are shown in Figure 3. In figure 3.1, images in the first column are the original images, images in the second column are predicted results, and images in the third column are the ground truth images with the filter applied on them. The first set of 3 x 3 images are for the neural network trained on the set of real images (Data Set 1). The second set of 3 x3 images, as shown in Figure 3.2, are for the neural network trained on the set of synthetic images that were generated randomly (Data Set 2).

Figure 3.1: Filter Learned on Neural Network Trained on Actual Images: Original, Ground Truth Filter, Model Learned



After applying the model on those 5 images, we use structure similarity to measure the similarities between the ground truth images and the synthetically 'manufactured' images, here are the results:

Figure 3.2: Filter Learned on Neural Network Trained on Actual Images: Original, Ground Truth Filter, Model Learned



From the results, it appears the model (filter) learned from artificial random images performs better than model learned from actual images because random images cover more R,G,B values and the neural network learns better model from them. To further validate and compare the results of the two neural networks, we use structure similarity [6] to measure the similarities between the ground truth filtered images and the results obtained from the two neural

Table 1: Structural Similarity

(a) Neural Network Trained on Real Images

Image	SSIM
Image-1	0.9113
Image-2	0.8630
Image-3	0.8342
Image-4	0.8585
Image-5	0.8252

(b) Neural Network Trained on Synthetic Images

Image	SSIM
Image-1	0.9764
Image-2	0.9672
Image-3	0.9531
Image-4	0.9691
Image-5	0.9495

networks that were trained. Table 1 (a) shows the structural similarity between ground truth filtered images and the output of the neural network trained on Data Set 1. Table 1 (b) shows the structural similarities between ground truth filtered images and the output of the neural network trained on Data Set 2.

As we can see from the results, the model trained on Data Set 2 (synthetic images) performs better than the model trained on Data Set 1 (actual images). The reasoning is most likely that the synthetic data set covers a wider range of R,G,B values and the neural network learns better model from them.

7 Future Work

The neural networks trained in our study serve as a concept to a wider and more robust application. Future work involves training networks on more complicated image transformations. A limitation of our approach is that the synthetic images that were used to train a better model will not work if the pixel transformations depend on adjacent pixel. On the brighter side, our preliminary investigations reveal that pixel transformations/filters in Instagram are independent of adjacent pixels. This will further be validated and tested in the future. We also want to build frameworks for more complex transformations, such as the ones available in applications like Prisma. Learning such transformations would require us to train more complicated neural networks, such as convolutional neural networks. The final goal is to build an application which incorporates image transformation features such as increasing brightness, adding vignettes, and many more complex transformations where users can construct their own filters, which can be exported to image-sharing applications. To share the filters, which are learned in the form of a neural network, image-sharing applications would need to incorporate modules to run the trained neural networks, and build a third-party integration module to connect to the proposed application.

8 References

[1] Hoard, Corey, Reverse Engineering Instagram's Hefe Filter

[2] Luan F., Adobe S., Schechtman Adobe E., et al, Deep Style Photo Transfer

[3] Memisevic R., Hinton G., Learning to Represent Spatial Transformations with Factored Higher-Order Boltzmann Machines

[4] Colton S., Torres P., Evolving Approximate Image Filters

[5] Bakhshi S, Shamma D, et al, Why We Filter Our Photos and How It Impacts Engagement

[6] Wang Z., Bovik A. C., Sheikh H. R., Simoncelli E. P., et al, Image Quality Assessment: From Error Visibility to Structural Similarity

[7] Watermark, 'https://www.watermark.ws'