Extracting Image Filter Presets Project

Michael Chang
Department of Computer Science
Stanford University
mchang6@stanford.edu

Leon Bi
Department of Computer Science
Stanford University
LeonBi@stanford.edu

Alex Lee
Department of Computer Science
Stanford University
alee8599@stanford.edu

Abstract

Emulating popular filter adjustments from other pictures is a difficult but highly sought after task in the photography world. We propose a model that extracts a "filter" comprised of different edits made to a picture so that edits from the reference picture can be replicated onto any other unedited pictures. Our model learns the different adjustment parameters such as contrast, brightness, and saturation from the reference picture so that these parameters can be applied onto a different image. This model manages to successfully produce results that are quite close to the state of the art Filter Style Transfer model from Lim et al. [1]

1 Introduction

Editing pictures can be quite difficult and it can take years to fully master the art of manipulating a picture. To combat this, companies such as VSCO and Instagram have created filters and presets, which are amazing for beginners wanting to emulate their favorite creators' look or a particular film's characteristics. Unfortunately, creating these filters or attempting to emulate a particular look through Photoshop or Lightroom adjustments is quite difficult.

Recently, style transfer, particularly with artistic renditions of normal everyday pictures, has been one of the more explored topics in deep learning. Many works have focused on transferring the style of a particular artist such as Picasso or Van Gogh or have focused on transferring the color and technique of a particular painting. However, there are fewer projects focused on maintaining the photorealistic qualities of an input images and focusing primarily on adjusting the color or texture of an image. Even within these projects focused on preserving the input image and solely adjusting the image's colors, many depend on a large dataset of images with similar filters applied to extract the proper filter adjustments. And because we are striving to provide a solution where one can extract the filter adjustments from a single image, these previous methods are a bit problematic.

To address this problem, we have created a simpler model, inspired by style transfer, that takes in a reference image with adjustments and learns the various adjustments that were made to the reference image. Then, we are able to take these parameters and apply the adjustments to other unedited images to replicate the adjustments from the reference image.

CS230: Deep Learning, Winter 2018, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

2 Related Work

Although there are many projects focused on transferring artistic styles [10], not many of them attempt to maintain the photorealistic elements of the original image and solely manipulate color or contrast. The model from Gatys et al. is able to transfer some of the color elements between similar images at different times in the day but often introduces distortion and unwanted textural elements from the reference image, which makes it unreliable for photorealistic style transfer [10]. Work from Luan et al [9] builds upon the photorealistic style transfer by introducing a photorealism regularization term that penalizes image distortion. This helps prevent the distortions seen in Gatys et al. [10] and is able to accurately change the input picture's lighting conditions from day to night time while accurately maintaining the color of the image. However, this approach is limited to transferring styles across similar reference and input images, which constrains the types of pictures that users transfer styles across.

Other approaches to photorealistic image style transfer include StyleBank and AutoStyle [7, 8]. StyleBank is an approach that focuses on explicit representation for image style transfer and is comprised of two components: an auto-encoder that decomposes the input image into multiple feature response maps and a collection of styles called the "StyleBank" that allows for training a multitude of different styles. By providing explicit representation of styles, the network is able to completely decouple the styles from the content of the image and allows for region-based style transfer as well as training for multiple styles without changing the auto-encoder [7]. On the other hand, AutoStyle is a different approach that takes in a keyword and input image and analyzes a number of images based off of that keyword. Then, based off of these images, the model takes a multi-step process that begins with replicating the vignetting, then the color from the collection of images referenced from the keyword, then the contrast of the keyword-based images [8]. Although transferring styles from a collection of images based on a keyword allows for more flexibility as well as robustness against outlier images, we are trying to find a solution that allows for style transfer based on one particular image rather than a collection of images.

Lastly, we will be referencing Yim et al. [1] as the current state-of-the-art as they have managed to successfully image style transfer from a single image onto another input image without any distortions. This last approach is the approach we have been most inspired by as it is the closest to the goals that we are trying to achieve. The model from Yim et al. utilizes defilterization to find the original image and then estimates the filter's parameters to figure out the edits necessary to get to the edited reference image. Our goal for this paper is to create a model that does not require this extra defilterization step and instead directly interprets the steps required to extract the filter.

3 Dataset

For our project, we use the MS COCO dataset which will be augmented the same way by utilizing the "Layeris" package in order to randomly adjust each images' brightness, contrast, and saturation to mimic filtered images [2, 3]. This pre-processing is done differently than the way the Filter Style Transfer Between Photos paper performs their preprocessing. Our goal is to see if our more randomized creation of filters can still be extracted and replicated by their model. Therefore we will be using the MS COCO dataset specifically for this task.

MSCOCO is a large-scale object detection, segmentation and captioning dataset with over 330K images. The images have a resolution of 640x480. The dataset is separated into 1/2 training, 1/4 val, and 1/4 test. With 80 object categories and over 1.5 million object instances. **Figure 1** contains some examples of the corrections applied.

4 Methods

The baseline for our project is how well the model in the Filter Style Transfer Between Photos paper performs on our own augmented dataset. We will then compare its results with the results of our simpler model that does not use object detection. We trained their model using our augmented dataset and ran some test images through their model. See results in **figure 3**.
As you can see the model described in [3] accurately transfers the filter from the reference photo to the test photo. However, there are a couple of flaws with this approach. The first is that the filter

Figure 1: An example of an photos before (top) and after (bottom) preprocessing

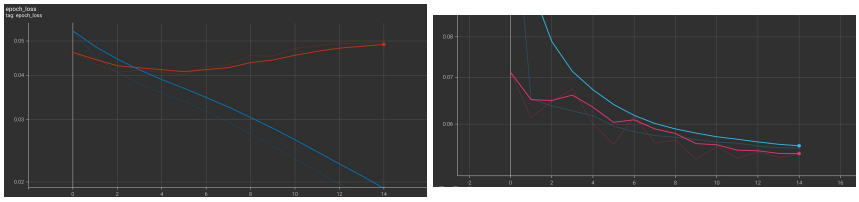

Figure 2: An example of the training loss (blue) and validation loss (red) of model 3 before regularization(left) and after l2 regularization (right)

decomposition is very complex and takes a long time to run due to the need to apply object detection to do color / hue correction. Our goal is to train a simpler model that parameterizes the photo filters in an easy to understand and apply format.

Our first attempt at our own model consisted of 3 convolutional blocks (consisting of a convolutional layer and a max-pooling layer) followed by two fully connected layers, the final output by a 4-vector. Our loss function was mean squared error between estimated features (y_prediction) and true features (y_true) over all train examples.

MSE loss = $\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$

We used this loss function instead of cosine similarity as described in our milestone since we are concerned with both correct angle and magnitude of our feature vector.

While this model performed well on the train set we generated, it did not perform well for the val set or for handpicked extreme filter values (filter feature vectors with a large norm). This we believed was due to overfitting and there not being enough extreme values represented in the train set. Therefore we added l2 regularization to out model to help it generalize. This model performed slightly worse on the train set but showed significant improvements on the val set and handpicked filter values. (**figure 2**)

We also experimented with batch normalization and dropout, but found that these had similar results to l2 normalization though performed a bit worse on both train and val sets.

Another method we tried was transfer learning by adapting a pretrained classification CNN as a feature extraction model (VGG19)[11] by removing the last fully connected layer and adding a new fully connected layer with 4 neurons. This allows the model to generate our 4-vector feature vectors for training. We used the prelearned weights to initialize the model then retrain on our generated images and feature vectors. This model performed poorly on the train / val set and is likely due to our objective being too dissimilar from the classification objective of VGG19 for transfer to be applicable.

Figure 3: An example of reference photo with filter (top) and transferred filter (bottom) using the method described in [3]

Figure 4: An example of the best performing models attempt at filter estimation, reapplied to the same photo. Original photo (left) edited photo for comparison (center) attempt at filter estimation (right)

Notable Experiment Details	**Train MSE**	**Test MSE**
Filter style transfer baseline	0.021	0.023
3 CNN blocks 2 FC w/ L2 Regularization	0.043	0.042
3 CNN blocks 2 FC	0.019	0.049
3 CNN blocks 2 FC w/ dropout	0.031	0.047
3 CNN blocks 2 FC w/ batch normalization	0.046	0.045
VGG19 w/ one FC layer	0.086	0.092

Figure 5: A table showing the models attempted and their related MSE (mean square error) for both train and val sets

5 Discussion

In total we worked with 6 distinct models whose performances are listed in **figure 5**. No model beat the baseline in performance, but model 3, the model containing 3 convolutional blocks and 2 fully connected layers with l2 normalization, performed comparably to the baseline qualitatively while having much simpler architecture. The metric we are using is MSE (mean squared error), which has the same equation as the MSE used in our loss function. Quantitatively this model achieved a training MSE of .043 and validation loss of .042 while also achieving qualitatively good looking results in filter transfer as shown in **figure 4**.
We did hyperparameter tuning on this model to try to achieve the best performance. We attempted a variety of learning rates, batch-size, and regularization power, and arrived at the best performance with a learning rate of .001, batch sizer of 32, and regularization of 0.1.
The algorithm fails to replicate very extreme filter values (feature vectors with extreme norms) (completely blown-out or dark photos). We attribute this to extreme values being underrepresented in our generated dataset as we generated the filter corrections from a normal distribution so extreme values are unlikely to appear. This is not an issue with the model as these extreme values are outside of our intended use-case and we can consider such input degenerate.

6 Conclusion/Future Work

We ultimately found that an architecture with 3 CNN blocks, 2 fully connected layers including L2 regularization performed the best. We decided to include L2 regularization once we found that the validation loss was increasing and that our model was over fitting. While we did not quantitatively outperform the state of the art baseline architecture we discussed, we did achieve a similar qualitative result with a lot less computational power and architectural complexity. In the future we hope to see if we can improve our model via more data or more computational resources. We also hope to explore forms of style transfer to see if rather than directly extracting specific image parameters like color and saturation, we can leverage a different technique to achieve better results.

7 Contributions

Alex focused on researching related works and the state of the art in the field. Leon and Michael generated the dataset and setup the infrastructure. All three of us worked together to find optimal architectures and hyperparameters for our model.

References

[1] Yim J., Yoo J., Do W., Kim B., Choe J. (2020) Filter Style Transfer Between Photos. In: Vedaldi A., Bischof H., Brox T., Frahm JM. (eds) Computer Vision – ECCV 2020. ECCV 2020. Lecture Notes in Computer Science, vol 12351. Springer, Cham. https://doi.org/10.1007/978-3-030-58539-6_7

[2] Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Doll´ar, P., Zitnick, C.L.: Microsoft coco: Common objects in context. In: European conference on computer vision. pp. 740–755. Springer (2014)

[3] Do Su-seok in my heart: layer.is (2019), https://github.com/subwaymatch/layer-is-python

[4] E. Reinhard, M. Adhikhmin, B. Gooch and P. Shirley, "Color transfer between images," in IEEE Computer Graphics and Applications, vol. 21, no. 5, pp. 34-41, July-Aug. 2001, doi: 10.1109/38.946629.

[5] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for largescale image recognition, 2015.

[6] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., … Chintala, S. (2019). PyTorch: An Imperative Style, High-Performance Deep Learning Library. In Advances in Neural Information Processing Systems 32 (pp. 8024–8035). Curran Associates, Inc. Retrieved from http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[7] Chen, D., Yuan, L., Liao, J., Yu, N., Hua, G. (2017). StyleBank: An Explicit Representation for Neural Image Style Transfer.

[8] Liu, Y., Cohen, M., Uyttendaele, M., Rusinkiewicz, S. (2014). AutoStyle: Automatic Style Transfer from Image Collections to Users' Images. Computer Graphics Forum, 33. https://doi.org/10.1111/cgf.12409

[9] Luan, F., Paris, S., Shechtman, E., Bala, K. (2017). Deep Photo Style Transfer.

[10] L. A. Gatys, A. S. Ecker and M. Bethge, "Image Style Transfer Using Convolutional Neural Networks," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 2414-2423, doi: 10.1109/CVPR.2016.265.

[11] Simonyan K., Zisserman A.: "Very Deep Convolutional Networks for Large-Scale Image Recognition", 2014; [http://arxiv.org/abs/1409.1556 arXiv:1409.1556].