Deep Photo Enhancer

Yerania Hernandez
Texas AM University
College Station, Texas
hernandez.yerania@tamu.edu

Nitin Gummidela
Texas AM University
College Station, Texas
v3ntinty@tamu.edu

ABSTRACT

Our paper proposes using an improved one-way GAN in order to provide an enhanced image. The network learns characteristics from a set of photographs that have the desired features the user prefers and applies these characteristics to the input image, transforming it into an enhanced image. The network is based on a Wasserstein GAN with additional modifications due to the infamous issue of convergence in one-way GANs. The first modification consisted of adding global features to the U-Net in order to capture the overall aesthetics of the image and locally adjusting the image. The second aspect was adding an adaptive weight scheme to reduce the sensitivity of the network on the weighting parameter of the gradient penalty and allowing for a more stable convergence. The last aspect added was individual batch normalization with the idea of improving the loss in the generator by maintaining a consistent set of inputs. With these modifications on the one-way GAN, we compare our results to the approach proposed by Chen using the PSNR scores and an evaluation on the visual quality of the images [1].

ACM Reference Format:
Yerania Hernandez and Nitin Gummidela. 2018. Deep Photo Enhancer. In *Proceedings of ACM Conference (Conference'18).* ACM, New York, NY, USA, 4 pages. https://doi.org/10.475/123_4

1 INTRODUCTION

High-quality photographs are not easily accessible despite the technology that has improved camera sensors and popularized smartphone cameras. In order to produce a high-quality image, camera sensors have to reconstruct the image based on noisy, decolorized, and low-resolution samples. Camera sensors also produce incoming light as a linear map compared to the human-perception that produces incoming light as a non-linear map. As a result, photographs do not usually appeal to the user at first glance considering they are not fulfill visual expectations. To address this problem, there are different software programs, such as the famous Adobe Photoshop, in order to enhance the color and sharpness of an image. Most of this software provide additional settings to enhance the image to the user's preference. As noted though, this is objective to the user and at times, requires a specific skillset to enhance while not distorting the image in order to successfully retouch the image properly.

Our paper proposes to provide image enhancement using deep learning techniques by following Yu-Sheng Chen's paper *"Deep Photo Enhancer"* [1]. Chen proposes using a modified one-way GAN to learn certain characteristics from a set of "good photographs" and then apply these characteristics and features to a new set of images that need to be enhanced [1]. Considering one-way GAN's are infamous for their instability to converge, he further applies this modified one-way GAN to a two-way GAN in order to achieve more consistent results. Due to our time limitation, we focused on implementing the modified one-way GAN with the additional feature of individual batch normalization that Chen only implemented for the two-way GAN. There are two original modifications that were implemented for the one-way GAN based on Chen's paper and we added an additional modification to the one-way GAN since it was used to further improve the two-way GAN. In the generator, global features were extraced from a encoding step and were concatenated in the bottleneck layer to capture additional aspects of the image, such as the overall environment setting, the global lighting, and the variation of subject and object types. These features will help the network perform proper local enhancements while maintain the overall visual appeal of the image. The second feature adapted was an "adaptive weight scheme for Wasserstein GAN (WGAN)" [1]. The one-way GAN proposed was based on the WGAN, which is highly sensitive to the weight penalty and does not easily converge as a result. Therefore, Chan proposes modifying the WGAN by providing adaptive weights to the network, which properly adjusts the lambda weight in order to ensure the Lipschitz constraint. Lastly, we implement individual batch normalization for each layer in the generator except the layers which extract the global features.

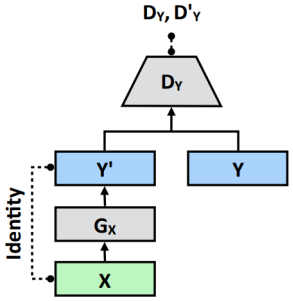

Figure 1: Overview of One-Way GAN structure

2 ALGORITHM

Our objective was to obtain a model ϕ such that takes in an input x and outputs $\phi(x)$, which is an enhanced version of x. This problem was approached as an image to image translation problem.

2.0.1 Generator: The architecture of the generator is as shown in Figure 2 The generator follows an autoencoder architecture with 6 encoding and 6 decoding layers and skip connections between

Conference'18, December 2018, Washington, DC, USA

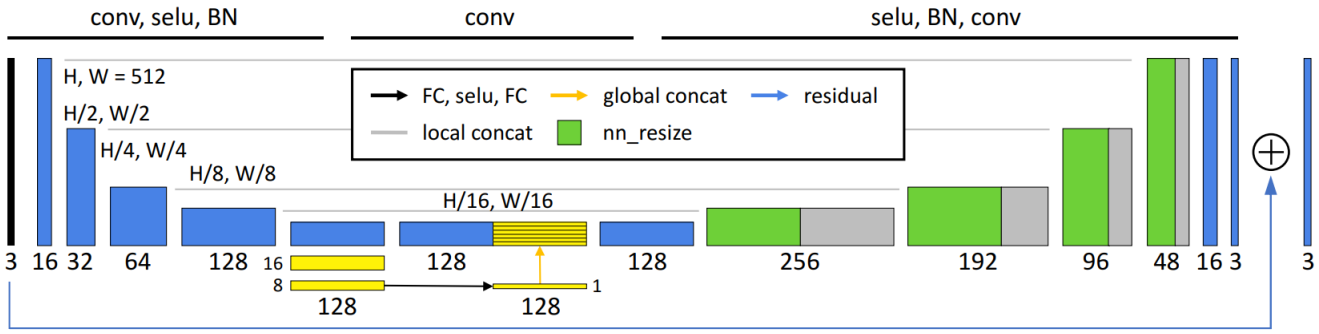

Figure 2: Generator architecture implemented

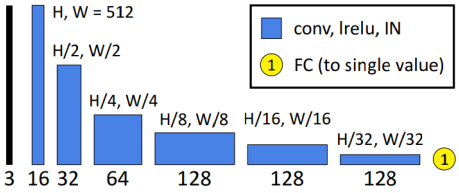

Figure 3: Discriminator architecture implemented

the corresponding layers in the encoder and the decoder sides. Additional convolutional layers were at the fourth layer to extract global features and these global features were concatenated at the bottle neck layer. Each encoder unit consists of a convolution layer with 3x3 filter, a stride of 2, and a padding of 2. Each decoder unit consists of a deconvolution layer with a 4x4 filter, a stride of 2, and a padding of 2. All the convolution and deconvolution layers are followed by scaled exponential activation unit (selu) and an individual Batch Normalization layer except for the connections with the global features. In addition, a residual connection is added between the first and the last layer.

2.0.2 Discriminator: The discriminator is shown in Figure 3. The discriminator is a classifier network that takes an input image and predicts the probability of the image being real or fake. Each block in the discriminator is a convolutional layer with a 5x5 filter, stride 2 and a padding of 2 followed by a leaky RelU activation and then by an Instance Normalization layer. The output of the last convolution layer maps to a single output with a fully connected layer followed by a leaky RelU activation.

2.0.3 Loss Functions: The objective function of the one-way GAN consists of several types of losses. The first ones are the adversarial losses for the generator and the discriminator which are given by the following equations.

$$A_G = E_y'[D_Y(y')]$$
$$A_D = E_y[D_Y(y)] - E_y'[D_Y(y')]$$

The next loss is the identity mapping loss given as follows:

$$I = E_{x,y'}[MSE(x, y')]$$

Along with these losses, a gradient penalty was added for the discriminator during our training, which is as follows:

$$P = E_{\hat{y}}[max(0, ||\Delta_{\hat{y}} D_Y(\hat{y})||_2) - 1]$$

Therefore, the overall optimization used for the discriminator is described by the following equation,

$$argmax_D[A_D - \lambda P]$$

where λ is equal to 10, and the generator is obtained using the following equation,

$$argmin_G[-A_G + \alpha P]$$

where α is the weighting factor between the adversarial loss and the identity mapping loss and was set to 1000.

3 IMPLEMENTATION

Our one-way GAN network was created using PyTorch and it was trained on an HPRC instance from Texas A&M using an ADA cluster which contains a dual NVIDIA K20 GPU. After developing the architecture described above, we developed a variety of data loaders to split our datasets into their proper training and test sets. The dataset used was MIT Adobe-5K dataset, which contained 5,000 photographs, and each photograph was retouched and enhanced by five different photographers. Photographer C ranked the highest score in the research paper based on user reviews and so result we used Photographer C's enhanced images as the ground truth for our network. It is important to note that our dataset for the input and the enhanced images each had to be partitioned into two different training sets, consisting of 2,250 images, and the final test set, consisting of 500 images. The input set would be provided to the networks in order to generate images, while the enhanced images would be used as ground truth in order to validate the images generated. The first training set was used to train the generator to learn the desired characteristics of the enhanced version, and the second training set would serve as the images that these characteristics needed to be applied to in order to enhance the image. A challenging aspect about our dataset when we first began training our generator was maintaining the quality of the image when we pass it into the network. Due to the fact that we could only use specific image formats when loading our images into the library *torchvision*, we had to convert the original DNG images into another format for usage. We explored using PNG and JPEG images, but these both lost resolution and certain aesthetics of the image, such as clarity and color. After a few trials, we used Adobe Lightroom and converted all the original DNG images into TIF images. Then,

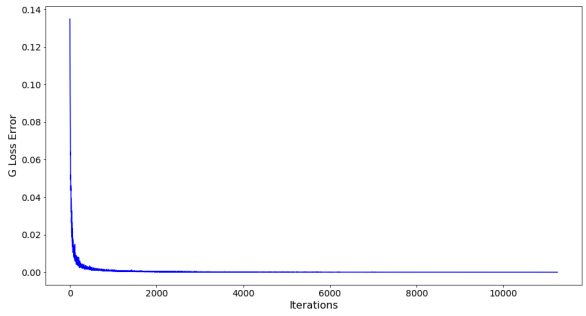

Figure 4: Generator Pre-Training Loss compared to the number of iterations

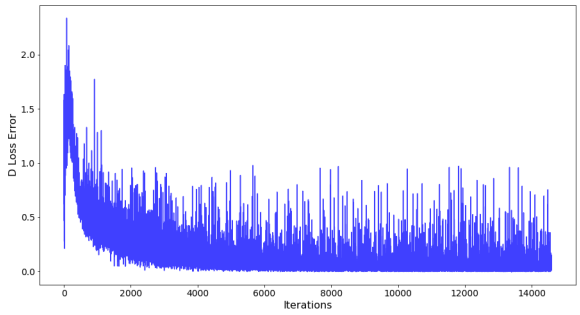

Figure 5: Discriminator training loss compared to the number of iterations

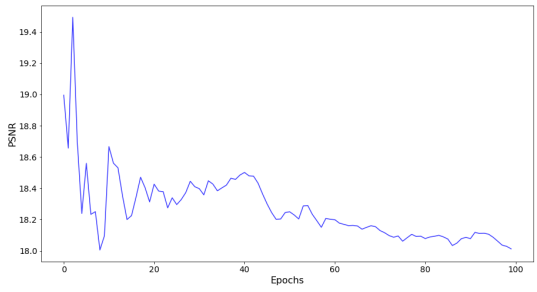

Figure 6: PSNR Score compared to the number of epochs

we randomly selected our images for the partitioning and split them into their proper data loaders.

Training the network involved two steps. Pretraining of the generator and then the GAN training. Our pre-training loop consisted of our generator network using an MSE loss and Adam as the optimizer in order to train the generator beforehand on the characteristics that will be used to apply on the input training set. However, the learning rate, batch size, and number of epochs were parameters we had to consistently test in order to allow our generator sufficient time to converge and produce similar images to the input. The learning rate was reduced from 0.001 to 0.00001 through a series of trial and error runs considering the pre-training loss would not converge or produce distorted and noisy images. Our final value, 0.00001, helped reduce our training loss from approximately 0.135 to 0.000014, as shown in Figure 4. In addition, we also changed the batch size from ten to five considering the training loop would run out of memory on the HPRC server instance. The last aspect we had to test in our pre-training loop was the number of epochs. After several runs, we achieved a low training loss after setting the epoch number to 50. Although we eventually reached these hyperparameter values, the main challenge was the amount of time this network takes to pre-train. It would consistently take over fifteen hours to obtain the results we have shown in the figure, but we had to run it a number of times to check if the loss would converge, analyzing how many epochs could we reduce it to in order to optimize the amount of time it would take, and evaluation if the images obtained were similar to the input.

Once the generator was pre-trained, we focused on analyzing and tuning our training loop. Within this training loop we used the second training set as the inputs and the first training set as the ground truths in order to produce our enhanced images. Considering we are training the discriminator and generator, we wanted to make sure it had enough epochs to converge as best as possible despite one of the main weaknesses of a GAN: convergence. GAN models are famous for oscillating and being unstable to the point where they never converge. However, considering we provide additional features, such as the adaptive weighting scheme, global features to the U-Net, and individual batch normalization, we can produce appealing images and reduce this high sensitivity and instability. As a result, we doubled the number of epochs for the training loop, running it for 100 epoch and maintained the same learning rate as the one configured for the pre-training loop. In addition to the parameters analyzed before, the training loop consisted of the discriminator and generator ratio (D/G ratio). We first investigated this ratio using a 1:1 ratio in order to analyze the results of the images. The images produced were dark and uncolorful, similar to those serving as the input. As a result, we began increasing the D/G ratio until we finalized our ratio to 50:1, the same ratio used in the research paper, considering these produced appealing images. When we finally decided on these hyperparameters, we trained the network for over 30 hours, not including the pre-training time. Once again, the time it takes for the network to train was a drawback in trying to produce adequate images.

The last aspect of this network was the test set, which would take approximately an hour to produce all the input and output images from the generator. In addition, we decided to calculate the PSNR score in order to quantitatively compare our results to those the paper presented. We have also setup our network for the two-way GAN with the two generators and all the hyperparameters that we have finalized from this modified one-way GAN. Due to the amount of time it takes to train a one-way GAN, we presume it will take at least twice as long to train the two-way GAN and due to the limited time, we were not able to train the two-way GAN, which is the second aspect of the research paper. However, we have used all the features proposed for the two-way GAN implementation from the

Figure 7: Image comparisons of our approach, Chen's approach, and the enhanced image of Expert C

Table 1: PSNR score comparison with original paper and our approach

Model Type	PSNR Score
1-way GAN (Chen)	22.17
2-way GAN w/ iBN (Chen)	22.37
1-way GAN w/iBN (Ours)	18.01

paper on our one-way GAN implementation in order to improve our results, ease the setup of the two-way GAN, and reduce the amount of time needed for fine-tuning.

4 RESULTS AND DISCUSSION

After training our network, we demonstrate promising results quantitatively and visually. The previous work has used PSNR score in order to analyze the quality of the images. In *"Deep Photo Enhancer"* [1], the one-way GAN without any individual batch normalization performs a score of 22.17. When finally implementing the two-way GAN with individual batch normalization, the score increases to 22.37. When comparing the results from our one-way GAN with individual batch normalization to this previous work, we achieve a final PSNR score of 18.01, as shown in Figure 6. Table 1 summarizes the quantitative results from the previous work compared to our approach, where iBN is short for individual batch normalization. Although our PSNR is close to the result achieved in the previous work with a difference of 4.16, further improvement could be accomplished by decreasing the discriminator loss. Figure 5 summarizes the discriminator loss during training for our one-way GAN architecture. Similar to the previous work, we are able to provide a drop in the loss and maintain that general decrease in the curve. However, unlike the previous work, our loss peak begins at a much higher rate, over 2.0, while the previous work stays below the 0.9 mark right before the drop. This difference provides insight into the fact that we could further tune our network to reduce the discriminator loss error, which will in return increase the PSNR score. In addition, the graph demonstrates a large variation as the number of iterations increases and therefore does not allow a smooth convergence as we had expected.

There are a few factors that could affect this loss error and cause our results to be slightly different than those achieved by the author. One of them being our dataset split considering we do not know what exact images he used for the train and test set and as a result, it varies the colorization and the characteristics that the generator needs to learn before it trains the network. When analyzing the images visually, such as in Figure 7, you can note that the enhanced images focus on a strong blue colorization, compared to our images that demonstrate a light blue with a purple hue. This could be due to the images we used to pre-train our network that had stronger resemblances of purple hues and caused the majority of images with blue to converge more to a purplish coloring. Another aspect that could be affecting the loss error is an unbalance between the generator and discriminator, which could be further analyzed with the D/G ratio. Although we started with a 1:1 ratio and increased it to a 50:1 ratio considering this was the ratio the paper used, we did not further experiment with other ratios to see what would work best with our one-way GAN that contained individual batch normalization compared to the simple one-way GAN Chen implements. With additional time we could perhaps further tune our hyperparameters to achieve a higher PSNR score and higher quality images. In addition, we provide additional images in Figure 7 comparing our results to Chen's approach and the ground truth, which in this case was the enhanced image from Photographer C in the dataset. By learning from this photographer, we are able to produce reasonable enhanced images as shown. As discussed, there are a number of hyperparameters that could be further adjusted in order to achieve similar results as the author and then applied to a two-way GAN in order to stabilize the convergence that is difficult to achieve in a one-way GAN.

REFERENCES

[1] Y. Chen, Y. Wang, M. Kao, and C. Yung-Yu. 2018. Deep Photo Enhancer: Unpaired Learning for Image Enhancement from Photographs with GANs. *Ministry of Science and Technology (MOST) and MediaTek Inc* (June 2018). https://www.cmlab.csie.ntu.edu.tw/project/Deep-Photo-Enhancer/CVPR-2018-DPE.pdf8