Personalized Image Editor

Orr Barkat, orrbarkat@mail.tau.ac.il, 302930813
Knaan Koosh, knaankoosh@mail.tau.ac.il, 305079246
Ben Ohayon, benohayon@mail.tau.ac.il, 304848518

https://github.com/bennzo/RetouchNet

Abstract

It is well guaranteed that every impressive shot published in the past decade has been post-processed. Post-processing is a very big part of what makes a photographer today. Furthermore, it consumes a great amount of his time. Post processing-Using a process to transform photos from what was captured in a camera either to be closer to what our eyes saw, or to alter the image artistically. **This project purpose is to automate this process while keeping the personal touch.**

Problem Description

In general, post processing is a combination of local touches (gradient filters, specific object manipulation etc.) and global settings of the photo. We will concentrate on global settings such as exposure, contrast, color, tone, gamma and more. These parameters effect the whole photo and not only local areas.

Photographers usually take pictures in a very high resolution (typical professional camera with 24 mega pixel sensor can produce 6000x4000 RAW image) to capture as many details as the camera is capable of. Automating post processing with deep neural networks requires a signifcant amount of CNN layers, which makes this procedure infeasible in high resolution. This is the main fallback of many methods used today in this area of research.

We limit the problem to supervised learning and handle only paired images (RAW and edited).

Prior Work

Pix2Pix (Phillip Isola, 2016) - This paper's main purpose is to create a general Image-to-Image Translation with Conditional Adversarial Networks (CGAN). The Generator used in the implementation is Unet and the Discriminator is PatchGAN. One of the issues about this model is its inability to handle high resolution inputs. The PatchGAN architecture is capable and designed to handle high resolution inputs but the generator is the computational bottleneck.

HDRnet (Gharbi, 2017) - The paper purpose is to enhances photos taken by mobile devices in FHD resolution (1920x1080) in an efficient and accurate way. Enhancement is done in an HDR style fashion- sharpening edges, enhance contrast, adjust brightness, etc. For this, the authors introduce a new neural network architecture inspired by bilateral grid processing (Chen J. P., 2007) and local affine color transforms. Their model can retouch an image but uses only L2 loss to achieve the desired results.

Exposure (Yuanming Hu, 2017) - This paper authors intended to solve the same problem as ours but in a whole different way - They used reinforcement learning and modeled the action space with a few simple filters such as- exposure, contrast, tone, etc. This method enabled them to derivate a sequence of filters that successfully transforms the RAW photo accurately. This method is highly complicated and limits the global changes to the preselected filters.

Datasets

MIT – Adobe FiveK Dataset

(Vladimir Bychkovsky, 2011)

MIT has collected 5,000 photographs taken with SLR cameras by a set of different photographers. They are all in RAW format; that is, all the information recorded by the camera sensor is preserved. They made sure that these photos cover a broad range of scenes, subjects, and lighting conditions. Five photography students were hired to adjust tones of the photos. Each of them retouched all the 5,000 photos using a software dedicated to photo adjustment (Adobe Lightroom) on which they were extensively trained.

We chose this dataset because it fits our purposes perfectly – we can show the difference in each model's results based on the sub-dataset (for a specific artist) it was trained on.

Methodology

General Idea

Our main idea is to quantify and generalize the artistic style of a photographer's retouches during post-processing. In order to achieve our goals we need a model that can handle high resolution pictures and capture the small changes in the differences between different photographers. That’s why we decided to combine HDRnet model with PatchGAN discriminator.

Losses

L2 loss - $\mathcal{L}_{L2} = \frac{1}{|\mathcal{D}|}\sum_i \left|\left|\boldsymbol{I_i} - \boldsymbol{O_i}\right|\right|^2$.

D is dataset size, I is the input image (aka "ground truth") and O is the output image (generated from the generator).

GAN loss - $\mathcal{L}_{GAN} = \mathbb{E}_{x\sim P_{data}(x)}[logD(x)] + \mathbb{E}_{z\sim P_z(z)}[\log(1 - D(G(z))]$.
D(x) is the PatchGAN result and G(z) is the generator result.

L2 loss is the default loss of HDRnet which is the base of our model. The use of GAN loss is one of the major changes we made in HDRnet model- Our intention is to achieve more refined results using this loss, knowing L2 loss provides a mean to push the results closer to the ground truth, even very little losses can be considered as accumulated tonal differences over the whole photo that could be very difficult to learn using L2 alone. Using a discriminator, we hope to overcome this problem.
Final Loss- $G^* = \arg\min_G \max_D \mathcal{L}_{L2} + \lambda\mathcal{L}_{GAN}$ (similar to Pix2Pix loss).

Network Architectures

The model structure is basically a GAN: it is combined of 2 models- a generator and a discriminator. The generator role is to create the correct image based on its input and the discriminator need to decide if this is an actual image from the "ground truth" pool or a "fake" created by the generator net.

Generator – HDRnet

Low-res coefficient prediction
The model receives a full resolution input (1920x1080) and down-samples it with a bilinear interpolation.
The low-res image then passed in 4 convolutional layers and splits into two process lines:

- The first extracts the **local features** using 2 convolutional layers. The spatial resolution is maintained the same (number of features is constant).
- The second path extracts **global features** using 2 convolutional layers and 3 fully connected layers. This path produces a fixed 64-dimensional vector that summarizes global information and acts as a prior to regularize the local decisions made in the first path (local).

Ultimately, both paths are combined using a linear prediction (Eq .2)[1] and results in a 16x16x64 array of features. Using another linear prediction (Eq .3)[1] on the fusion array, yields a 16x16x96 array of coefficients that can be viewed as a 16x16x8 bilateral grid where each grid cell contains 12 numbers, one for each coefficient of a 3x4 transformation matrix.

	S^1	S^2	S^3	S^4	L^1	L^2	G^1	G^2	G^3	G^4	G^5	F	A
type	c	c	c	c	c	c	c	c	fc	fc	fc	f	l
size	128	64	32	16	16	16	8	4	–	–	–	16	16
channels	8	16	32	64	64	64	64	64	256	128	64	64	96

Table 1. Generator architecture layer summary. Source: (Gharbi, 2017)

Full-res processing

[1] Equations in parenthesis refer to the equations in the HDRnet paper (Gharbi, 2017).

To process the full resolution image, we use a layer based on the bilateral grid (Chen J. P., 2007) and an upsampling operation (Chen J. A., 2016) on the grid we call slicing.The layer takes the bilateral grid from the low-res pipe and a guidance map and performs a data-dependent up-sampling on the bilateral grid using the guidance map. This operation creates 12 sliced coefficients full resolution matrices which are finally combined to assemble the result.
(Guidance map - The map is defined as a simple transformation (Eq. 6)[1] on the full resolution image. Furthermore, the parameters of this transformation are being learned during training process)

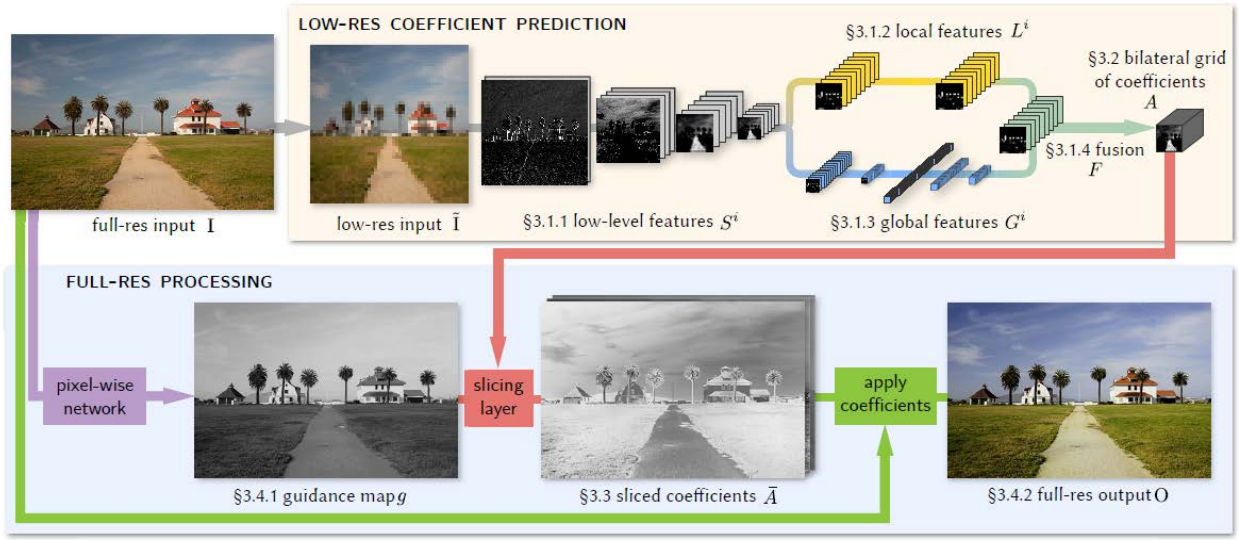

Figure 1. Complete HDRnet architecture. Source (Gharbi, 2017)

Discriminator – PatchGAN

To use GAN advantages, we needed to add a discriminator part to our model. The model needs to support high resolution inputs because of the nature of the problem. PatchGAN was a good solution because of its ability to be applied on arbitrary large pictures. It also models high-frequency structure to aid HDRnet get the desired results.
The difference between a PatchGAN and regular GAN discriminator is that rather the regular GAN maps from a full resolution image to a single scalar output, which signifies "real" or "fake", whereas the PatchGAN maps from full resolution to an NxN array of outputs X, where each X_{ij} signifies whether the patch (i,j) in the image is real or fake.

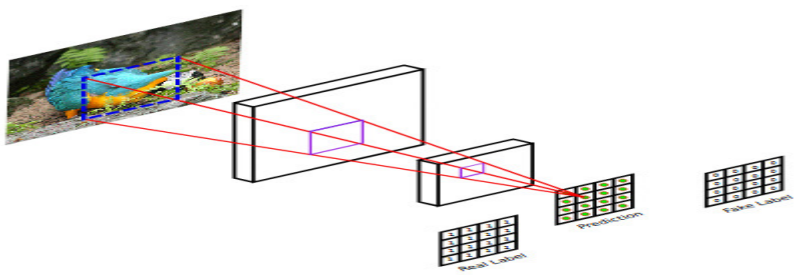

Figure 2. PatchGAN illustration

[1] Equations in parenthesis refer to the equations in the HDRnet paper (Gharbi, 2017).

Training Method

We train our model on the FiveK Dataset mentioned earlier. Each set contains 5000 full resolution input and output pairs and we optimize the generator's parameters by minimizing the L2 Loss and the discriminator's by minimizing a classic GAN Loss.

Experiments

Since originally HDRNet did not include results for the FiveK Dataset, our two main purposes were, first to check whether the generative network was expressive enough to model a photographer's editing style as an operator, and second to check if using an adversarial architecture (adding a discriminator) would improve the results qualitatively.
To do so we train our model with and without a discriminator on three different experts resulting overall in six experiments, although the training times were a bit longer than expected we decided to test more than one expert since some experts looked a bit inconsistent with their retouching and we were worried it might hinder our results.

Data

All experiments were tested on sets from the FiveK Dataset, specifically on the sets "Expert A", "Expert B" and "Expert C", the sets were downsampled to 1080x1920 to efficiently train on batches and were split into training and validation sets with a ratio of 80%-20%.

Training Parameters

We use an Adam optimizer for both generator and discriminator with an initial learning rate of 10^{-4} and regularize with weight decay of 10^{-8}, the rest of the optimizer parameters are set by default initialization. The Training sessions ran for about 30-40 epochs with batch sizes of 4-8 due to memory limitations which took an average of around 8 hours for each session.

Evaluation Metrics

We evaluate our models both quantitatively and qualitatively by calculating the pixel loss over the entire validation set and by inspecting individual failed and successfully transformed photos. Although our losses and input/output distance can't tell much about the quality of the results in terms of style transfer they can indicate whether the model successfully adapts a new operation, lastly we present a few of the transformed images.

Results

We present the results carried out to evaluate our approach to transfer style. The first experiment, as mentioned above, aims to examine how expressively powerful our model is in

terms of our goal which is photography editing style transfer. The following images show the achieved performance of our model with and without a discriminator.

Expert A:

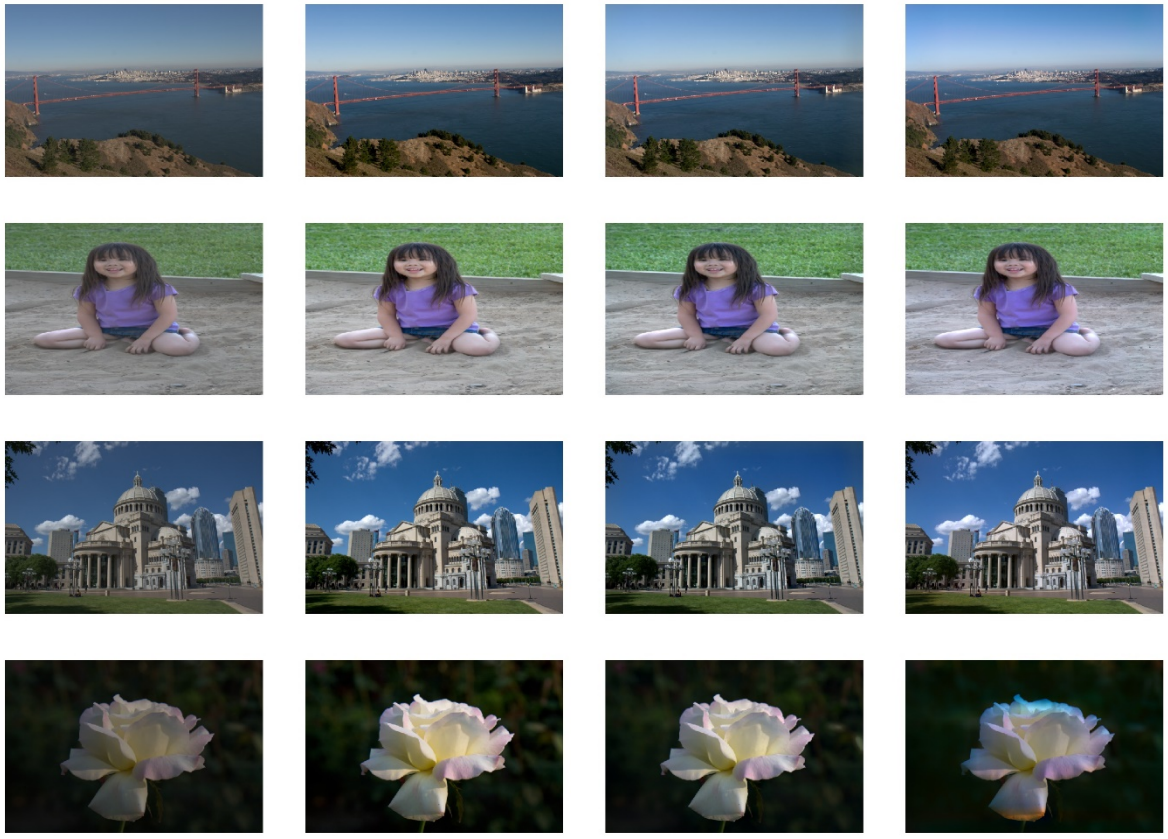

Figure 3. Photo transformation result showcase for Expert A
Order: Original, Edited, Generated, Generated (With Dis.)

Expert B:

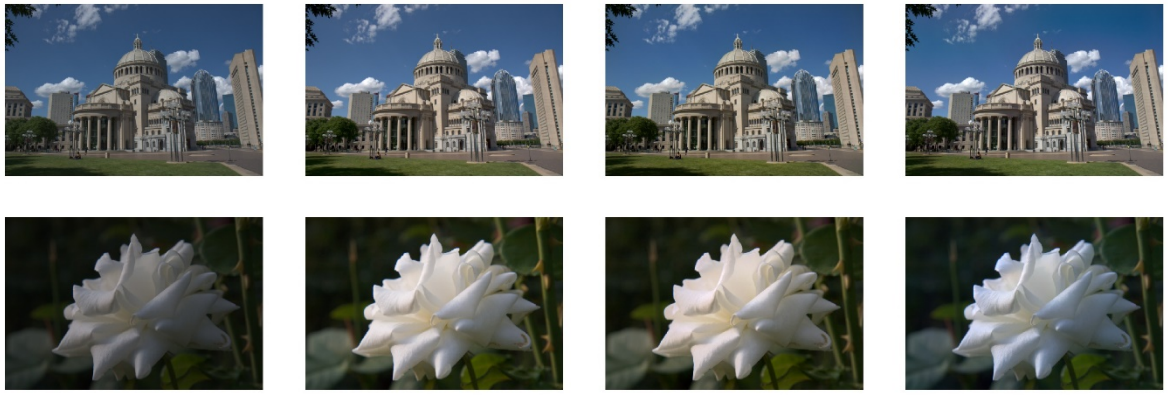

Figure 4. Photo transformation result showcase for Expert B
Order: Original, Edited, Generated, Generated (With Dis.)

Expert C:

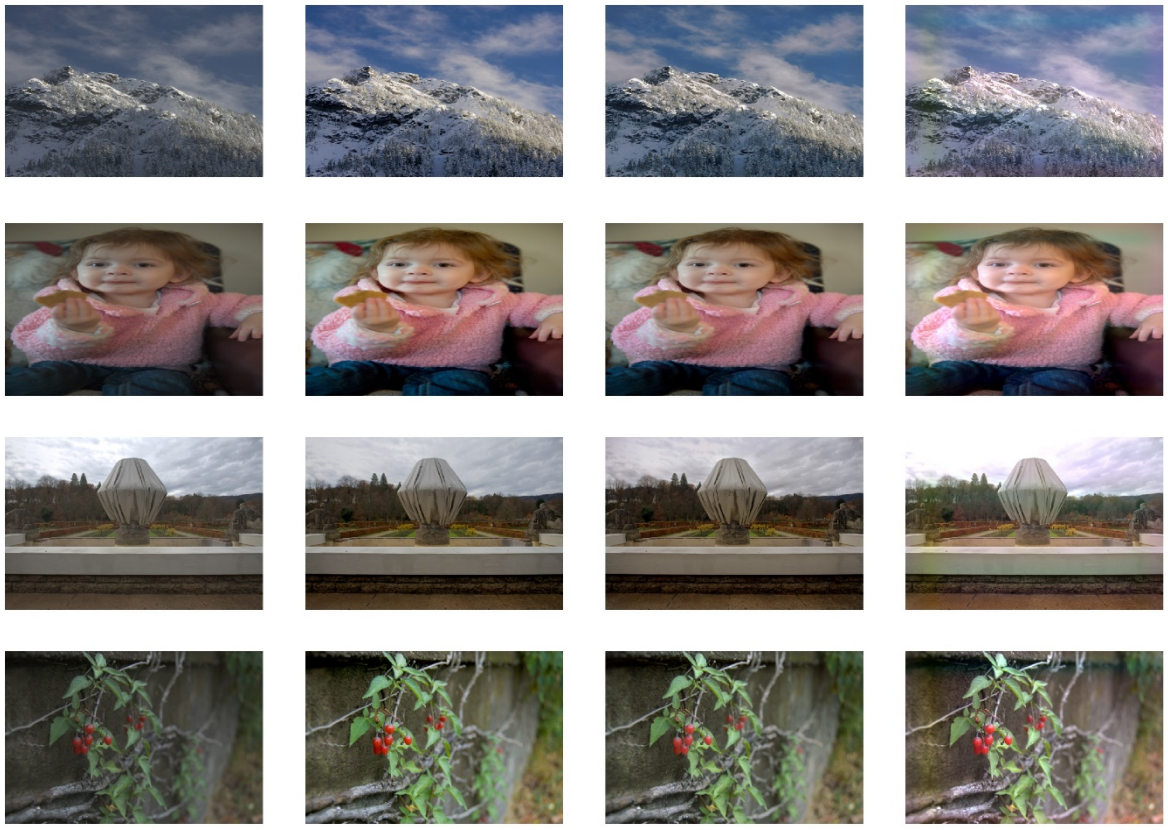

Figure 5. Photo transformation result showcase for Expert C
Order: Original, Edited, Generated, Generated (With Dis.)

As the results show, adding a discriminator didn't quite improve the performance in a way we were hoping for, that is to say we can see that certain white shades throughout all the experts get a blueish tone, moreover the overall color integrity decreases. Still, the results for the model without a discriminator were pretty satisfying for all the experts although in some examples colors were not quite as vivid as the photographer intended.

In addition we compared the training progress between the three examined Experts and as expected we can see that Expert B, the expert we consider the most consistent in his editing style, was the easiest one to learn. That led us to believe that the wide variety of tone, brightness and contrast changes in some of the sets (Expert A for example) might be too complicated to learn.

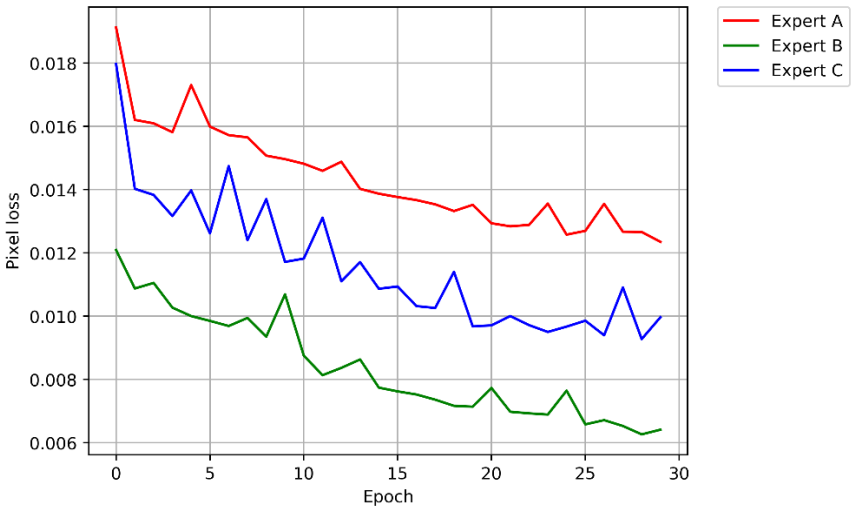

Figure 6. L2 Loss vs Epoch for three of the experts over the validation set

We evaluate quantitatively our second experiment where we checked whether adding a discriminator would improve performance:

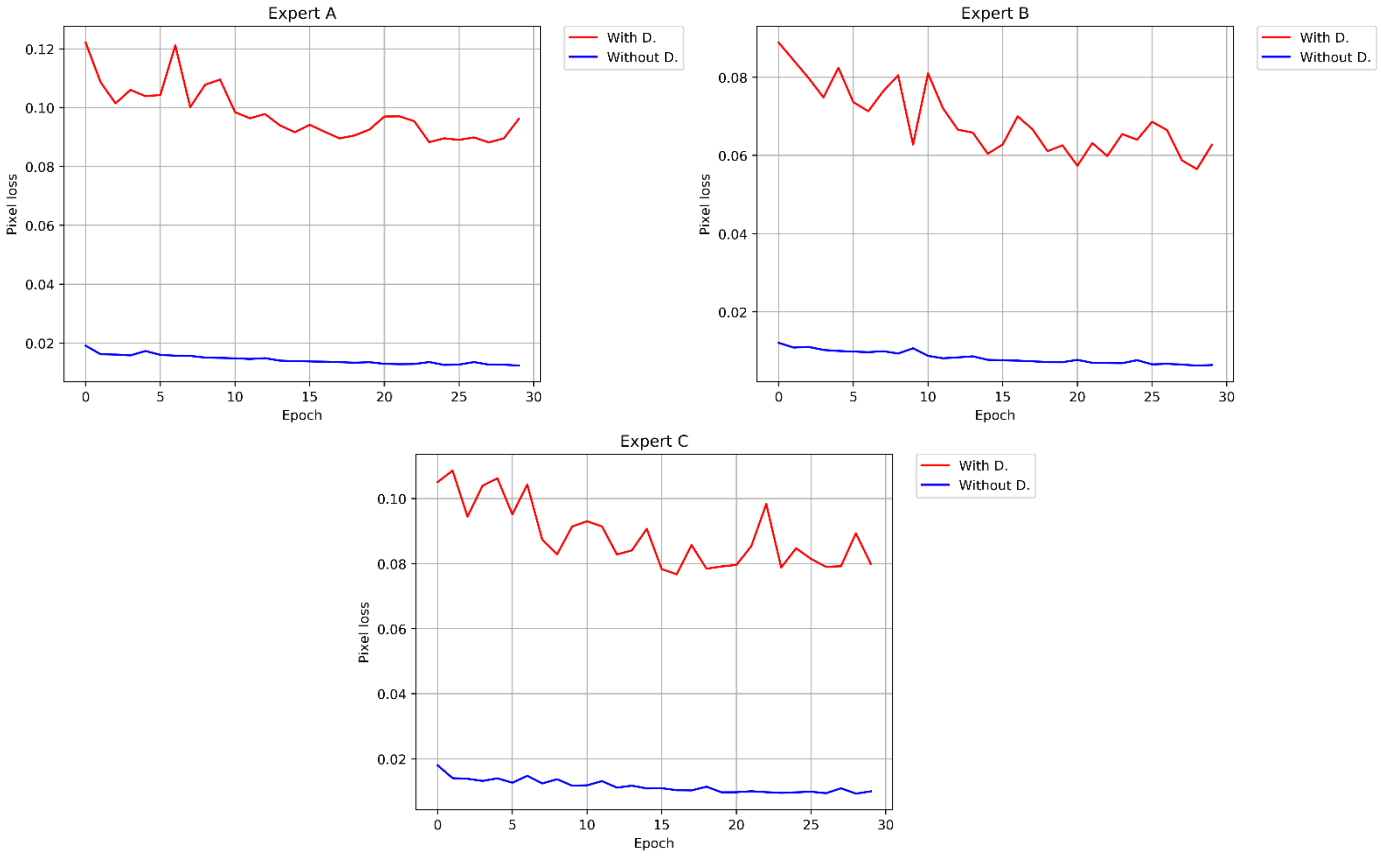

Figure 7. Training progress over the validation set for the model with and without a discriminator.

As the graphs show, training a model with a discriminator proved much harder and yielded inferior results in terms of pixel loss on the validation set. Yet we can argue whether pixel loss is relevant in our case since the qualitative comparison were reasonably equal.

Conclusions and Discussion

We have built a model based on the HDRnet paper, hoping to achieve different functionality. The model extracts local and global features from an edited photo and attempts to infer an operator which mimics the photographers editing style when applied to an unedited photo.

We hoped that treating the HDRnet as a generator and building an adversarial architecture around it would yield superior results in our case since early discussion concluded that minimizing L2 Loss alone over the dataset would not be enough to transfer style. Unfortunately adding a discriminator hindered the process as It was harder to train and didn't really improve the quality of the results.

A Possible reason for this decrease in results could be a lack of random process in the operator inference, in any case for further work we recommend checking a well written paper (Yuanming Hu, 2017), where the authors attempt to build a series of predefined simple operations on images to transfer editing style using reinforcement learning.

Code

Parts of the generator implementation (specifically the slice_and_assemble function) were taken from the recent paper (Li, 2018).

Code structure

```
├── README.md – Description and instructions
├── data – Data processing modules folder
├── models – Network modules folder
├── main.py – Module in charge of parsing run arguments
├── train.py - Training script
├── train_noD.py – Training script for a network without discriminator
└── utils.py – Various utility function used throughout the framework
```

Models

```
├── patch_gan.py – Implementation of a PatchGAN discriminator
└── rt_generator.py – Implmentation of a Generator as described in the HDRNet paper
```

References

Chen, J. A. (2016). Bilateral Guided Upsampling. *ACM Trans. Graph.*

Chen, J. P. (2007). Real-time Edge-aware Image Processing with the Bilateral Grid. *ACM Trans. Graph.*

Gharbi, M. C. (2017). Deep bilateral learning for real-time image enhancement. *ACM Transactions on Graphics (TOG)*, 118.

Li, T.-M. a.-K. (2018). Differentiable Programming for Image Processing and Deep Learning in Halide. *ACM Trans. Graph.*, 139:1-139:13.

Phillip Isola, J.-Y. Z. (2016). Image-to-Image Translation with Conditional Adversarial Networks. *CoRR*.

Vladimir Bychkovsky, S. P. (2011). Learning Photographic Global Tonal Adjustment with a Database of Input / Output Image Pairs. *The Twenty-Fourth IEEE Conference on Computer Vision and Pattern Recognition.*

Yuanming Hu, H. H. (2017). Exposure: A White-Box Photo Post-Processing Framework. *CoRR*.