MASTER'S THESIS | LUND UNIVERSITY 2018

Evaluation of Deep Learning Approaches in High Dynamic Range Imaging

Simon Johansson

Department of Computer Science
Faculty of Engineering LTH

ISSN 1650-2884
LU-CS-EX 2018-20

Evaluation of Deep Learning Approaches in High Dynamic Range Imaging

Simon Johansson
`tna11sjo@student.lu.se`

July 19, 2018

Master's thesis work carried out at Sony Mobile Communications AB.

Supervisors: Sebastian Raase, `Sebastian.Raase@Sony.com`
Jörn Janneck, `Jorn.Janneck@cs.lth.se`

Examiner: Michael Doggett, `Michael.Doggett@cs.lth.se`

Abstract

This thesis analyzes the viability of neural networks for full-scale learned image transformations, specifically with high dynamic range imaging. One problem with enhancing photographs in mobile cameras today is that quantitative improvements to the sensor, such as higher resolution, provide little enhancement. Computational photography is one venture that provides better subjective results, but such algorithms can be costly and in some cases proprietary. Estimating these transformations with neural networks is an ongoing research problem.

Using variations of residual convolutional neural networks, we have created high dynamic range images with similarity to the results of the applied algorithm. Best performance was achieved using a model that takes multiple burst images of the same scene as input. A number of artifacts were discovered in the synthesized images, most of which could be reduced given further research and improvements.

Keywords: HDR, Machine Learning, Convolutional Neural Network, Computational Photography, Burst Photography

Acknowledgements

A special thank you goes to Sebastian Raase, without whose feedback this thesis would be in a far worse shape.

I would like to thank both of my supervisors for their help, their insights wherever needed, and for all the interesting conversations.

Thanks also go out to the team at Sony, who provided an inspiring and friendly environment.

Contents

1 Introduction **7**
1.1 Problem Description 7
1.1.1 Research Question 8
1.2 Related Work 8
1.3 Scope 9

2 Background **11**
2.1 High Dynamic Range Imaging 11
2.2 Machine Learning 12
2.2.1 Training 12
2.3 Convolutional Neural Networks 13
2.3.1 Convolution 14
2.3.2 Activation Function 15
2.3.3 Backpropagation 16
2.3.4 Batching 16
2.3.5 Normalization 16
2.3.6 Residual Block 17

3 Method **19**
3.1 Dataset 19
3.1.1 Pre-processing 19
3.2 Tools 20
3.3 Network Structure 20
3.3.1 Baseline Network 20
3.3.2 Multi-Image Network 20
3.3.3 Dataflow Graph 21
3.4 Training 22

4 Evaluation **23**
4.1 Quantitative Measurements 23

4.1.1 Peak Signal-to-Noise Ratio 23
4.1.2 Structural Similarity 23
4.2 Evaluation Limitations 24

5 Results **27**
5.1 Training 27
5.2 Baseline Network 28
5.3 Multi-image Network 29
5.4 Comparison 30
5.5 Performance 32
5.6 Limitations 32

6 Discussion **35**
6.1 Performance 35
6.2 Multi-image Network 36
6.3 Limitations 36

7 Conclusions **39**
7.1 Future Work 40

Bibliography **41**

Appendix A Additional images **45**

Chapter 1

Introduction

Cameras and their sensors are more advanced today than ever. With the case of diminishing returns in quality improvement of pictures taken with higher pixel count, many experts have moved towards computational photography to produce better images. Whether the desired result is improved exposure, higher resolution or decreased noise, it can often be achieved through post-processing with few caveats. This is made especially useful when hardware is lacking, either due to cost or space.

One of the biggest challenges to taking high-quality pictures in phone cameras is dealing with low-light situations. Cell phone cameras have small apertures, which limits the number of photons they can gather, leading to noisy images in low light. They also have small sensors and therefore small pixels, which limits the number of electrons each pixel can store, leading to limited dynamic range. This is where High Dynamic Range imaging, a technique using multiple similar images of the same scene to reproduce a greater dynamic range of luminosity, excels. Dynamic range in imaging defines the ratio between the largest and the smallest quantity of luminance level.

1.1 Problem Description

The purpose of this thesis is to evaluate deep learning approaches for High Dynamic Range imaging (HDR). It is of interest to see if the effects of HDR can be applied reliably and robustly in a deep neural net. From a practical standpoint a question to answer is if the representation learned by the neural network could be applied on most images for an increase in dynamic range without significantly sacrificing image quality.

Also, another area that is explored is if taking multiple images of the same scene taken in a burst as input to the network can provide improvements to the final image.

1.1.1 Research Question

The main research question is whether deep convolutional neural networks can be used to reliably apply HDR features to an image and further explore if multiple image inputs of the same scene can provide a quality improvement to the final image.

1.2 Related Work

Hasinoff et al. [10] created the original HDR+ algorithm used for denoising and tone mapping currently used in Google's Pixel line of phones. Their main aim is to create a robust system for improving an image. One way they accomplish this is by having a merging method that is robust to misalignment. This is also the source for the dataset used in our implementation.

Gharbi et al. [6] taught a neural network to approximate the enhancement made by a reference imaging pipeline in real-time on a low performance device. They applied the neural network on a low resolution version of the image to find global and local features combined with a full scale mapping. The network is trained with a loss function based on the final produced image. Because of the nature of the network it can be used to model complex transformations where no reference implementation is available, such as human retouches. While it uses HDR+ as an example function it does not use multiple images as input.

Khademi Kalantari and Ramamoorthi [14] also propose a learning based approach to address the issue of artifacting in an HDR based algorithm. To accomplish this they take a set of three low dynamic range images of three different exposures to learn an appropriate tone mapping which is then applied to the medium exposure image.

An and Lee [7] propose a single-shot HDR imaging algorithm using a convolutional neural network (CNN). They aim to recreate under- and overexposed pixels and improve image quality with more details and less artifacts than conventional algorithms. Their approach builds on Nayar and Mitsunaga's [13] spatially varying pixel exposure (SVE) which spatially varies pixel exposures to simultaneously sample scene radiance along with dynamic range dimensions.

Eilertsen et al. [5] propose a learning based method for automatic recovery of highlight information to reconstruct a visually convincing scene from a single exposed standard image in the vein of HDR. They base their approach on a CNN design in the form of a hybrid dynamic range autoencoder.

Ignatov et al. [12] suggest a solution for super-resolution, heightening the resolution of a given low resolution image using a Residual CNN, with a unique loss that takes color, texture and content loss into account. This is done in part by the use of an adversarial CNN-discriminator and a pre-trained deep CNN. They use images from three different older cameras of the same scene as input to their network and a DSLR camera image as their ground truth. Their results show a quality improvement on par with that of the DSLR-taken images.

Chen et al. [3] successfully create a high quality rendition of an extreme low-light image using a deep convolutional neural network. It uses an external amplification ratio to decide how much an image should be lightened and can visualize what seems like a black image to the human eye with low noise.

1.3 Scope

The thesis focuses on Google's HDR+ technology as a baseline, not only because of their recently released large dataset [9], but also because of the generally positive evaluation it has received both objectively and subjectively. As the main interest lies in robustness of the transformation, this thesis does not fully explore possibilities of extreme conditions when it comes to complexity or network topology. It is assumed that quality improvements could be made with a larger network at the cost of time and performance, and equally that a structure could be made that performs better and can be evaluated faster at the cost of quality.

Chapter 2
Background

The main subjects this thesis explores are the fields of computational photography and deep learning. High Dynamic Range imaging is a methodology of combining multiple images into a single one with increased dynamic range. It is a time consuming process, often split into multiple steps such as alignment, merging and tone mapping. As these are functions applied to a set of images it is likely that deep learning is a good fit for approximation.

2.1 High Dynamic Range Imaging

Leaps and bounds have been made in the improvement of sensors and other factors that increase the quality of images, but real world capturing still proves a problematic task. One of the main issues arises in the discretization of the color space and luminance range, which often leaves photographs looking nothing like human perception of the real world.

Dynamic range in imaging defines the ratio between the largest and the smallest quantity of luminance level.

One way to achieve a higher dynamic range (HDR) is to combine multiple low dynamic range (LDR) images into a single HDR image using computational methods. Excluding special hardware, this is usually done sequentially by capturing multiple images with very little time variance and at different exposures, which are then merged in software. The merge is performed by a weighted average of the pixel values across the varying exposures, usually after aligning the images. A naive HDR algorithm generally suffers from a number of artifacts, such as ghosting due to misalignment of different exposures, excessive denoising in low light conditions leading to loss of fine detail and excessive range compression giving an unconvincing painting rendition.

Hasinoff et al. [10] developed an algorithm based on HDR named HDR+ that combines multiple, underexposed frames as a means of noise removal and further applies tone mapping to maintain local contrast while brightening shadows. What sets it apart from the standard HDR result is that overexposed and underexposed pixels cannot be reconstructed into the dynamic range, due to using the same exposure for every image. Because of this it is extremely important that the correct exposure is chosen. Using HDR+ today is made easier by the current hardware implementation of burst photography on modern phones, which can take a collection of similarly exposed images in a very short time span.

2.2 Machine Learning

Machine learning is a methodology which aims to get computers to act without being explicitly programmed. It has been applied in numerous fields, including natural language processing, object recognition, search engines, medicine and many more. Machine learning is closely related to computational statistics and is often used as a means of searching for patterns in data [2]. It has proven exceptional at finding patterns in data automatically. One principle within machine learning is *supervised learning*, which takes a data set containing training examples bundled together with their expected outcome. For example, a set of pictures of handwritten digits along with their correct labels can be used to train a network to classify numbers.

Typically, machine learning consists of two phases:

1. Training
    - A training dataset is used to estimate model parameters.
2. Prediction
    - Once the parameters have been estimated, the model can be used to classify future data.

In general the training is going to be the most time consuming part, but there are some models in which the prediction time increases with the size of the training data.

One sub-field within machine learning is called deep learning, which is a deep structure with many hidden layers in contrast to a regular artificial neural network.

2.2.1 Training

The training of machine learning models can be seen as an incremental attempt at improvement of accuracy towards a goal given by the training data. A given *iteration* consists of submitting the input, applying a function with a set of weights, evaluating the output given the correct answer and then updating the weights of the function accordingly. A complete iteration over the entire set of training data is generally referred to as an *epoch*, and is generally the accepted way of expressing training duration. A model usually requires multiple epochs in order to be properly trained, but if the volume of the training data is too small, too much training can lead to *overfitting* to the data. Overfitting is a problem that arises because of unevenly distributed or lacking data and causes the model to make assumptions about future input which might not be ideal. For example, if the training set for classifying digits consists of five times as many ones as sevens, the model might assume that ones are more likely to be the answer regardless of the input. This problem can be avoided by gathering more training data and can be alleviated by regularization techniques such as the addition of a regularization term, dropout or stopping the training early.

Loss Function

The *loss function* in machine learning describes how you evaluate the differences between the output of the model with regards to the correct answer. It maps the differences to a real number, which is then minimized in order to improve the model's predictive capabilities. The choice of loss function is very important, as it guides the entire learning process, and a well formulated loss function can significantly speed up the training. An example of a loss function, which is used in our implementation, is *mean square error* (MSE). It compares the squared difference of every pixel in an image and computes the mean. It is defined as

$$MSE = \frac{1}{mn} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} [I(i,j) - K(i,j)]^2 \tag{2.1}$$

where I, K are images, m, n are the image spatial dimensions.

2.3 Convolutional Neural Networks

Neural networks get their name from the biological representation of a brain, which consists of neurons that transmit information. Generally we visualize neural networks as graphs, with nodes as our neurons and edges connecting them making up the communication between layers [8].

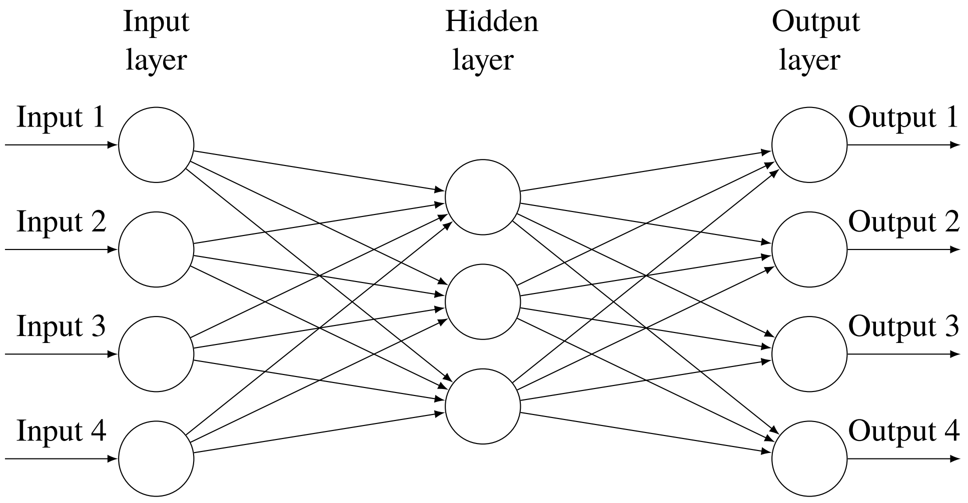


Figure 2.1: A typical representation of a neural network with one input layer, one hidden layer and one output layer. The input and output layers both have four nodes, the hidden layer has three nodes and all of the layers are fully connected.

A convolutional neural network is a network architecture that gets its name from the multiple convolutional layers that it employs. Its importance lies in the complexity reduction it provides compared to a fully connected network, with each node in one layer only being connected to a fixed number of nodes on the next layer. This is called *sparse connectivity* and makes use of a smaller viewing window remotely inspired by human vision, which is known to focus in patches [4].

2.3.1 Convolution

Convolution is a linear operation which aims to average several measurements of data (f), with the ability to specify weights (g) for a result of a weighted average.

$$(f * g)(t) = \int_{\tau=-\infty}^{\infty} f(\tau)g(t-\tau)\,d\tau \tag{2.2}$$

In convolutional neural networks, the data f is usually referred to as the *input* and the second argument g is called the *kernel*. Equation 2.2 is the definition in the continuous domain, where in the case of computers and digitalized media the input is generally discrete, resulting in equation 2.3:

$$(f * g)(t) = \sum_{\tau=-\infty}^{\infty} f(\tau)g(t-\tau) \tag{2.3}$$

In the case of neural networks, the input is usually a multidimensional array of data, for example an image. As such, the kernel also consists of a multidimensional set of adaptable parameters that can be trained. Because of these qualities, convolution is performed over more than one axis at a time. With an example of a two-dimensional image I of size (m, n) and a two-dimensional kernel K we have:

$$(I * K)(i, j) = \sum_{m}\sum_{n} I(m, n)K(i-m, j-n) \tag{2.4}$$

Figure 2.2 is a visualization example of the two-dimensional convolution in equation 2.4.

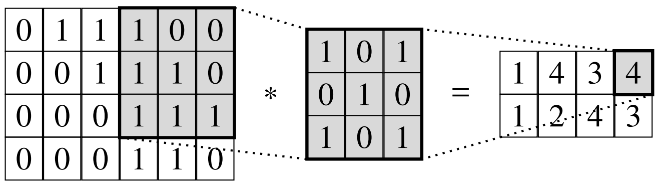


Figure 2.2: Convolution is performed by applying a sliding filter (usually referred to as the *kernel*) over the input data.

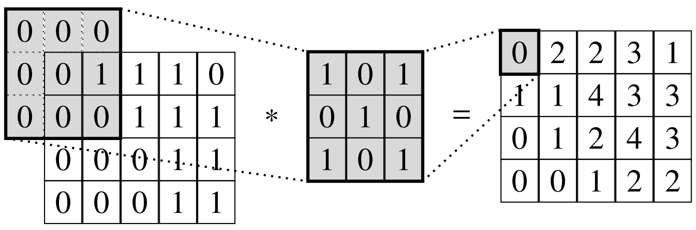


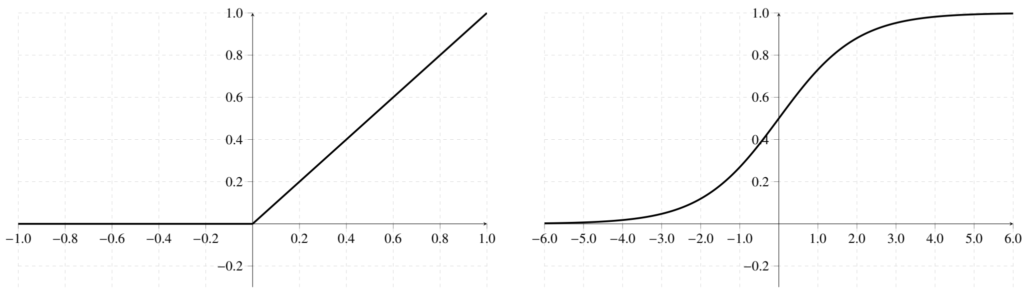
Figure 2.3: 2-D convolution with padding, causing the output to have the same size as the input.

2.3.2 Activation Function

The output of a given node is defined by its activation function, which introduces non-linear properties to the network. The main reasoning behind its usage is that without it, the output signal would be a simple linear function which has less complexity and less power to learn function mappings from data. There are a multitude of non-linear functions which are used in practice, all of which are slightly different in how they perform. One of these is the *Rectified linear unit* (ReLU), which is defined in equation 2.5.

$$R(x) = \begin{cases} 0 & x < 0 \\ x & x \geq 0 \end{cases} \tag{2.5}$$

As is evident, the ReLU activation function is almost linear, consisting of two linear components. The ReLU activation function can be seen in figure 2.4a.



(a) The Rectified Linear Unit (ReLU) activation function, defined in equation 2.5.

(b) The sigmoid activation function, defined in equation 2.6.

Figure 2.4: The two activation functions used in our implementation.

Another activation function is the *Logistic function*, usually referred to as the sigmoid function, that is defined by

$$S(x) = \frac{1}{1 + e^{-x}} \tag{2.6}$$

What sets the sigmoid apart from the ReLU is the fact that it is continuously differentiable, which helps enabling gradient-based optimization methods. Its derivative is also defined for zero where ReLU is not. The sigmoid activation function can be seen in figure 2.4b.

2.3.3 Backpropagation

There are several ways to measure the progression of the training. One is to count *iterations*, which is a complete forward pass through the network and the update of weights known as backpropagation. Another important measure is *epochs*, which is one iteration for every set of input data.

Historically, a lot of neural network implementations have used stochastic gradient descent to perform backpropagation. As the name implies it uses the gradient of the loss function to decide how the weights should be updated. There have been a lot of improvements made to this algorithm, one of the most used today is called Adam [15], which stands for *Adaptive Momentum estimation*. The main idea of the use of this implementation is that it takes momentum of the loss into account, running averages of both the gradients and the second moments of the gradients.

2.3.4 Batching

The process of training in *batches* (or the more accurate term *mini-batches*) is when you bundle a number of input together for all forward passes. This serves to both reduce training time, as more samples can be handled simultaneously which leads to more optimization, and gives a more robust gradient since it is based multiple inputs. Generally, a larger batch size will improve the results at the cost of increased memory usage during training.

2.3.5 Normalization

In neural networks, two kinds of normalization need to be considered, input normalization, which is used when inputs of different features tend to be on different scale, and batch normalization, which is used inside the network. In our case, input normalization is not needed as every input pixel is already normalized to the range $0 - 255$, but batch normalization is used.

Batch normalization makes sure to keep weights inside the network balanced and not take on extreme high or low values. It increases the speed of the training process as well as making each layer more robust and independent from the other layers. Each batch is scaled by the mean and variance computed for that batch, which also helps with varied data. Batch normalization is defined by

$$y_{tijk} = \frac{x_{tijk} - \mu_i}{\sqrt{\sigma_i^2 + \epsilon}} \tag{2.7}$$

where

$$\mu_i = \frac{1}{HWT}\sum_{t=1}^{T}\sum_{l=1}^{W}\sum_{m=1}^{H} x_{tilm}, \quad \sigma_i^2 = \frac{1}{HWT}\sum_{t=1}^{T}\sum_{l=1}^{W}\sum_{m=1}^{H}(x_{tilm} - \mu_i)^2 \tag{2.8}$$

Batch normalization normalizes all inputs across the batch (T) and the height (H) and width (W), with regard to the input x, the mean (μ) and variance (σ). A small epsilon (ϵ) is used to avoid division by zero.

A variation on batch normalization that is used in our implementation is *instance normalization* [16], which is very similar to batch normalization, except that it doesn't normalize across the batch. Instance normalization is defined by

$$y_{tijk} = \frac{x_{tijk} - \mu_{ti}}{\sqrt{\sigma_{ti}^2 + \epsilon}} \tag{2.9}$$

where

$$\mu_{ti} = \frac{1}{HW}\sum_{l=1}^{W}\sum_{m=1}^{H} x_{tilm}, \quad \sigma_{ti}^2 = \frac{1}{HW}\sum_{l=1}^{W}\sum_{m=1}^{H}(x_{tilm} - \mu_{ti})^2 \tag{2.10}$$

This avoids adding noise to our input which is introduced from the normalization over the entire batch, which can have both a positive and a negative effect.

2.3.6 Residual Block

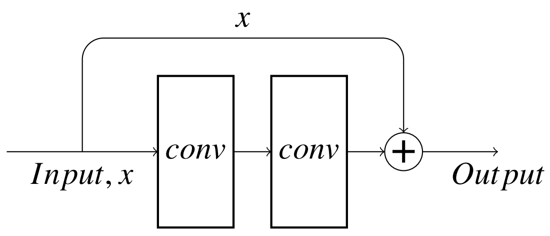


Figure 2.5: An example of a residual block, consisting of two convolutional layers and a direct connection between input and output.

With deeper neural networks becoming more and more difficult to train, different attempts have been made to alleviate the strain on resources and helping convergence. One of these methods is the use of residual blocks [11], having a skip connection over multiple layers. While residual neural networks have been shown to provide good results, they are not well understood yet. They deal with the problem of having a *vanishing gradient*, which causes the weights in the network to stop updating. This happens because of the nature of backpropagation, where the weights are updated according to the derivative of the loss function with respect to the current weight. In some cases, this gradient will become so small that it essentially stops the weights from changing their values. The way that residual networks alleviates this problem is through its construction of many short networks together, which doesn't quite solve the vanishing gradient problem since it doesn't preserve the gradient flow through the depth of the network.

Because the vanishing gradient problem is less of an issue, networks can be built to be deeper and more complex, which helps to get an accurate representation of the data.

Chapter 3
Method

The bulk of our implementation focuses on deep learning and the convolutional neural network, with a portion of pre-processing of the input data.

3.1 Dataset

The dataset used for this thesis was released by Google [9] and contains 3640 image burst captures, made up of 28461 images in total. Each burst consists of the raw burst input and metadata. The results provide both an intermediate aligned and merged image in raw format, as well as the final result image in the form of a JPG.

The images were captured using a variety of Android mobile cameras (Nexus 5/6/5X/6P, Pixel, Pixel XL) and have a varying number of burst sizes (2 to 10). Each image is generally 12-13 Mpixels and was taken with the same exposure time and gain per burst. The result images were generated using the HDR+ system in the Google Camera app [10].

This dataset is a perfect fit for this thesis, as it is quite large and fulfills all the requirements necessary to train our network. It is worth explicitly expressing that this is a dataset for HDR+ and not regular HDR and therefore does not fulfill the same purpose in terms of exposure variance.

3.1.1 Pre-processing

Before being fed into the network, the training data needs to be pre-processed. Training on full-size RAW-format images would be unfeasible and slow. The processing is done in two steps:

- Post-processing of RAW image
- Splitting images into patches of size 200x200 pixels

This way, each image created 50 patches, of 200 by 200 pixels. This increases the total input data to 182 000 different images. These patches were chosen based on their position in the photographs and were not especially curated to depict something, leaving some images to be visually uninteresting, but important for the network to understand the structure of regular scenes.

3.2 Tools

The training of the model was done on a 6-core hyperthreaded machine running Ubuntu 16.04 with an Nvidia GTX 1080 Ti graphics card and 32 GB RAM. The drivers required for deployment of training on a graphics card was CUDA 9.0 and CUDNN 7.1.

The majority of the implementation is done in Python 3.6.3, with the implementation of the neural network done in TensorFlow [1]. For tracking progress during training and generating graphs TensorBoard was used.

3.3 Network Structure

The main network used as the baseline for comparison is a *12-layer residual convolutional neural network*. The proposed solution is an end-to-end model that performs full-scale quality improvement of an image based on high dynamic range imaging methodology.

3.3.1 Baseline Network

The baseline network is inspired by the generator network put forth by Ignatov et al. [12]. The reason this is used as a baseline is that it provides a good result (even with a simpler loss function, which was examined by Ignatov et al.) for a similar problem. It also falls in the category of medium-sized networks, not too small to properly model the images but also not so massive as to become a complexity issue on its own.

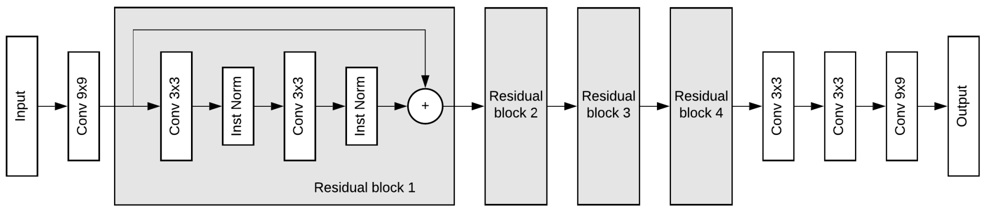


Figure 3.1: The baseline network structure, consisting of 12-layers split into 4 identical residual blocks.

3.3.2 Multi-Image Network

The Multi-Image Network (MIN) is created under the hypothesis that more input to the network will yield a positive result on the outcome. In order to have as fair of a comparison as possible, the MIN is a modification of the baseline network, containing 3 residual blocks and an early component that separately processes the input images. Each input is effectively going through 12 layers, which is similar to the baseline.

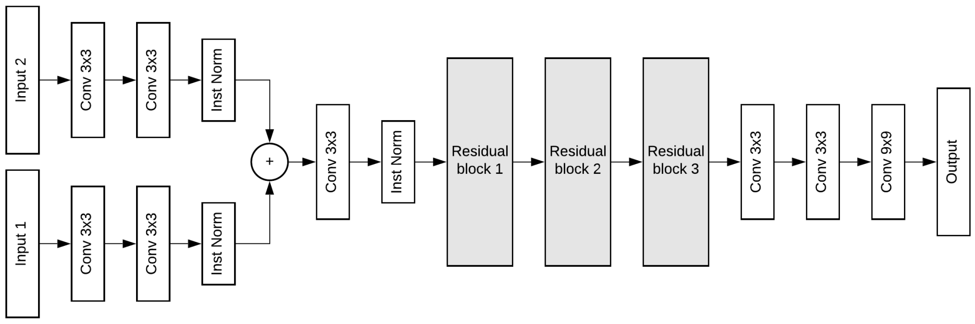


Figure 3.2: The network structure of the multi-image pipeline (in this case with 2 input images), consisting of separate processing of the input that is later combined and further fed through the network.

Further expansion into more images as input would require changes to the first 3 layers of the network. The latter part can't be made smaller without limiting the processing of the images too much.

3.3.3 Dataflow Graph

TensorFlow is made on the principle of building a dataflow graph to represent computations in terms of dependencies between individual operations. This gives the user the ability to first define the graph, and thereby the network, before execution and feed data into it. The nodes in the graph represent computations, whereas the edges represent the flow of data in terms of input and output from the nodes.

The use of dataflow graphs gives many boons for program execution in terms of optimization and usability. As the entirety of the graph is defined before execution, the system easily identifies operations that can be executed in parallel. In the same vein, it makes it possible to distribute computation to multiple units such as GPUs (Graphical Processing Unit), CPUs (Central Processing Unit) and specialized hardware, even distribution across different machines is possible. The compiler can make a number of optimizations in order to generate faster code using the information in the dataflow graph, such as merging adjacent nodes. Further, the nature of the graph is language-independent, meaning a saved model can easily be used in another programming language which improves both collaborability and allows for development in one language and deployment in a more suitable one.

3.4 Training

Training was done using the before mentioned hardware (3.2) in a number of different tests. Generally each training session took between 3-7 days, which attests to the size of the training set as well as the computational complexity of the network. The two experiments that were examined finally were

1. Baseline network trained for 17 epochs
2. Multi-Image network trained for 17 epochs

For all tests the Adam algorithm (see section 2.3.3) was used for backpropagation. The learning rate was set to 10^{-4} throughout the training processes and the batch size was set to 20, which was the biggest possible due to memory limitations in the case of the Multi-Image network. The training is performed on patches of size 200 by 200 pixels.

Chapter 4
Evaluation

4.1 Quantitative Measurements

4.1.1 Peak Signal-to-Noise Ratio

Peak Signal-to-Noise Ratio (PSNR) is a method for objectively comparing two images. In our case, it is quite useful to compare the output of the network to the ground truth images made available in the dataset, which is the result we are emulating. PSNR is also closely related to the mean square error, which is our loss function (equation 2.1). PSNR is defined as

$$\mathrm{PSNR}\,(I,K) = 10\ \log_{10}\left(\frac{\mathrm{MAX}^2}{\mathrm{MSE}\,(I,K)}\right) \tag{4.1}$$

where MAX is the maximum pixel value of the image, 255 for an 8 bit image. MSE is the mean square error defined in equation 2.1. As PSNR is defined on a logarithmically scaled value with the MSE in the denominator, the more alike the images are the higher the PSNR. Two identical images would have MSE $= 0$, which means the PSNR goes to positive infinity and is handled as a special case.

4.1.2 Structural Similarity

Structural SIMilarity (SSIM) is a well-used quantitative measurement developed to be better correlated with the perception of the human visual system [17]. Much like PSNR, it is used to compare two images and this means we again evaluate our output based on the ground truth image. Where it differs though is that PSNR estimates an absolute error between the images whereas SSIM tries to compare perceived changes in structural information in the image. It combines a loss of structure (s), contrast (c) and luminance (l) and is generally defined as

$$\mathrm{SSIM}\,(I, K) = [l(I, K) \cdot c(I, K) \cdot s(I, K)] \tag{4.2}$$

where l, c, s are defined as

$$l(I, K) = \frac{2\mu_I\mu_K + C_1}{\mu_I^2 + \mu_K^2 + C_1} \tag{4.3}$$

$$c(I, K) = \frac{2\sigma_I\sigma_K + C_2}{\sigma_I^2 + \sigma_K^2 + C_2} \tag{4.4}$$

$$s(I, K) = \frac{\sigma_{IK} + C_3}{\sigma_I\sigma_K + \sigma_K^2 + C_3} \tag{4.5}$$

$C_1 = (0.01 \cdot \mathrm{MAX})^2$ and MAX is the maximum pixel value of the image. (μ_i, σ_i^2) are the average and variance of the image i.

$C_2 = (0.03 \cdot \mathrm{MAX})^2$.

$C_3 = \frac{C_2}{2} = \frac{(0.03 \cdot \mathrm{MAX})^2}{2}$ and σ_{IK} is the covariance between I and K.

For the purposes of comparing images of multiple channels (RGB in our case), each channel is compared in the image separately and the results are then averaged.

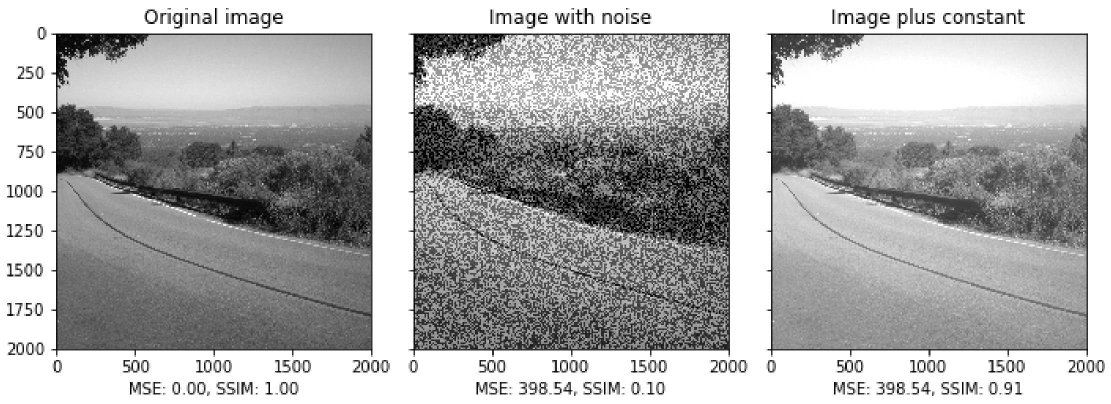


Figure 4.1: Comparison between the mean square error and the SSIM as a measurement of the quality of an image. The two rightmost images have the same MSE and therefore also the same PSNR.

4.2 Evaluation Limitations

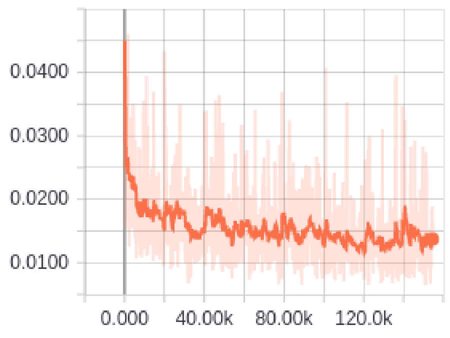
Because of the nature of the problem and the proposed solution, quantitative measurements might not be the best way to evaluate the end result of the network. Firstly, they are only comparisons between the produced image and the ground truth, which is the HDR+ generated images. This means we have to assume that the ground truth is the optimal outcome, which is completely fine for the training of the network but might be wrong and would require another kind of evaluation. Secondly, as we're entirely looking for an improvement in quality and positive subjective perception of the output, quantitative measurements tend to fall short. There is no such thing as a perfect numerical evaluation of the quality of an image, which tends to push research towards utilizing subjective blind tests for preference.

Chapter 5
Results

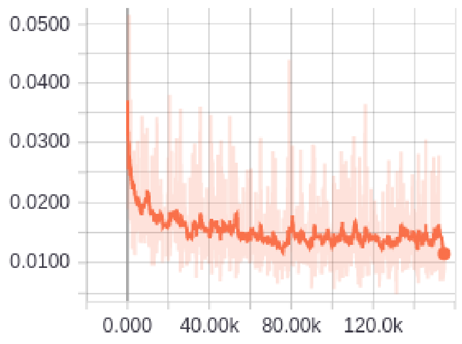
A larger number of tests were done for each type of network and the results given are the ones that gave the best quantitative scores. The evaluation is done on 50 images that were excluded from the training to give as much of an un-biased result as possible.

5.1 Training

The training process can be seen in figure 5.1. It has plateaued quite early during the training process, roughly at the tenth epoch.



(a) The loss progression of the baseline network.



(b) The loss progression of the multi-image network.

Figure 5.1: These figures show how the loss (y-axis) lowered with the number of iterations (x-axis). The graphs are displayed with a smoothing factor of 0.9.

5.2 Baseline Network

The baseline network was trained for 17 epochs, which is 154 700 iterations and it took roughly 3 days on the previously specified hardware. Example images can be seen in figure 5.2.

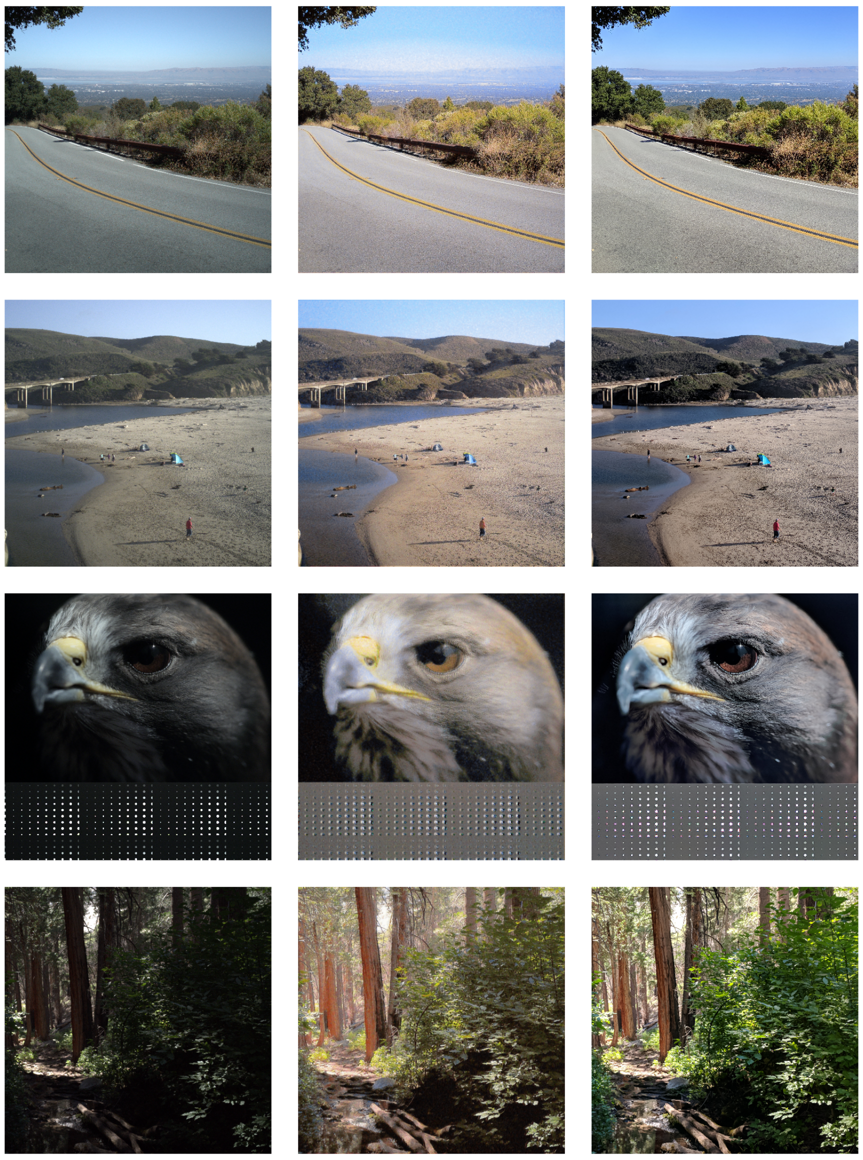


Figure 5.2: From left to right columns: Input images, images generated by the baseline network and finally the ground truth images.

5.3 Multi-image Network

The multi-image network was for comparison also trained for 17 epochs, and it took roughly 4 days. Example images can be seen in figure 5.3. More images generated by the multi-image network can be seen in Appendix A.

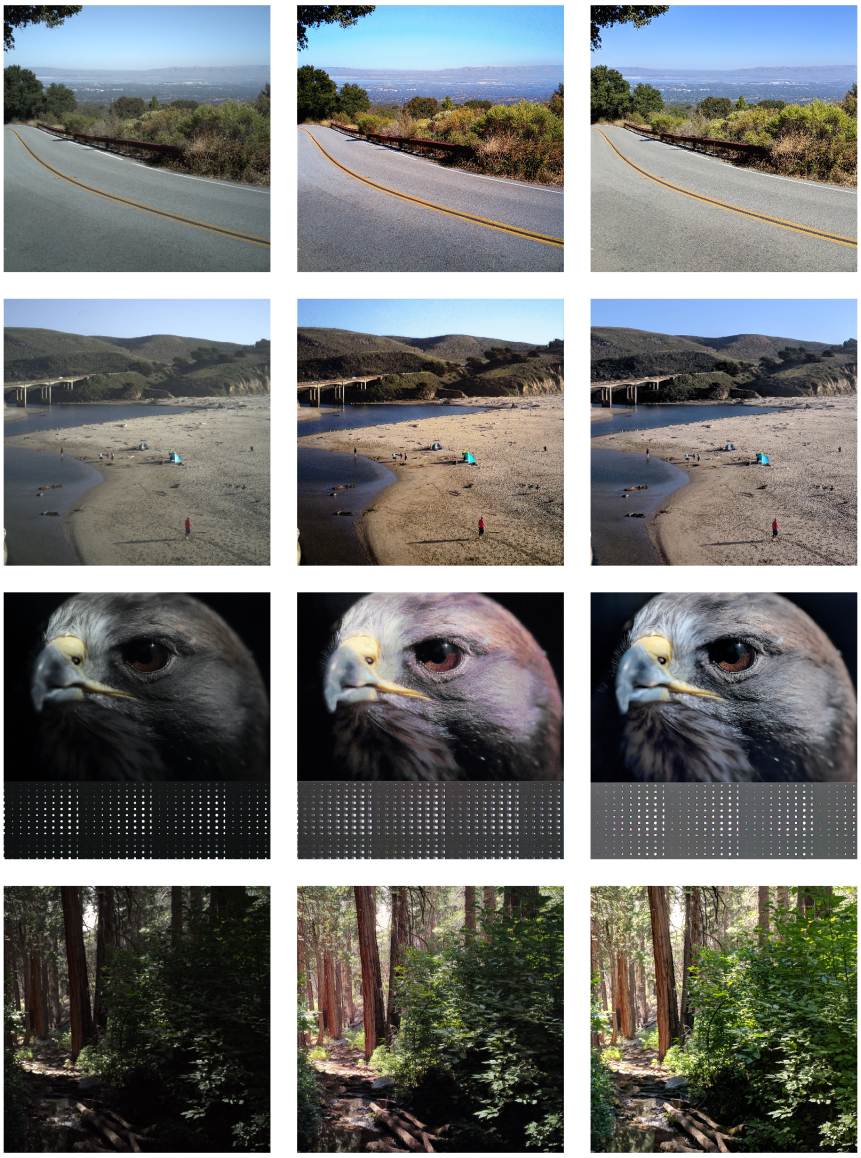


Figure 5.3: From left to right columns: Input images, images generated by the multi-image network and finally the ground truth images.

5.4 Comparison

The networks were numerically compared using PSNR and SSIM (explained in chapter 4) and the results can be found in table 5.1. Generally, a high PSNR and SSIM is desirable, with more subjective measurements generally being more important but harder to quantify.

Comparison images for the two networks and the improvements that were made from the input images can be seen in figure 5.4.

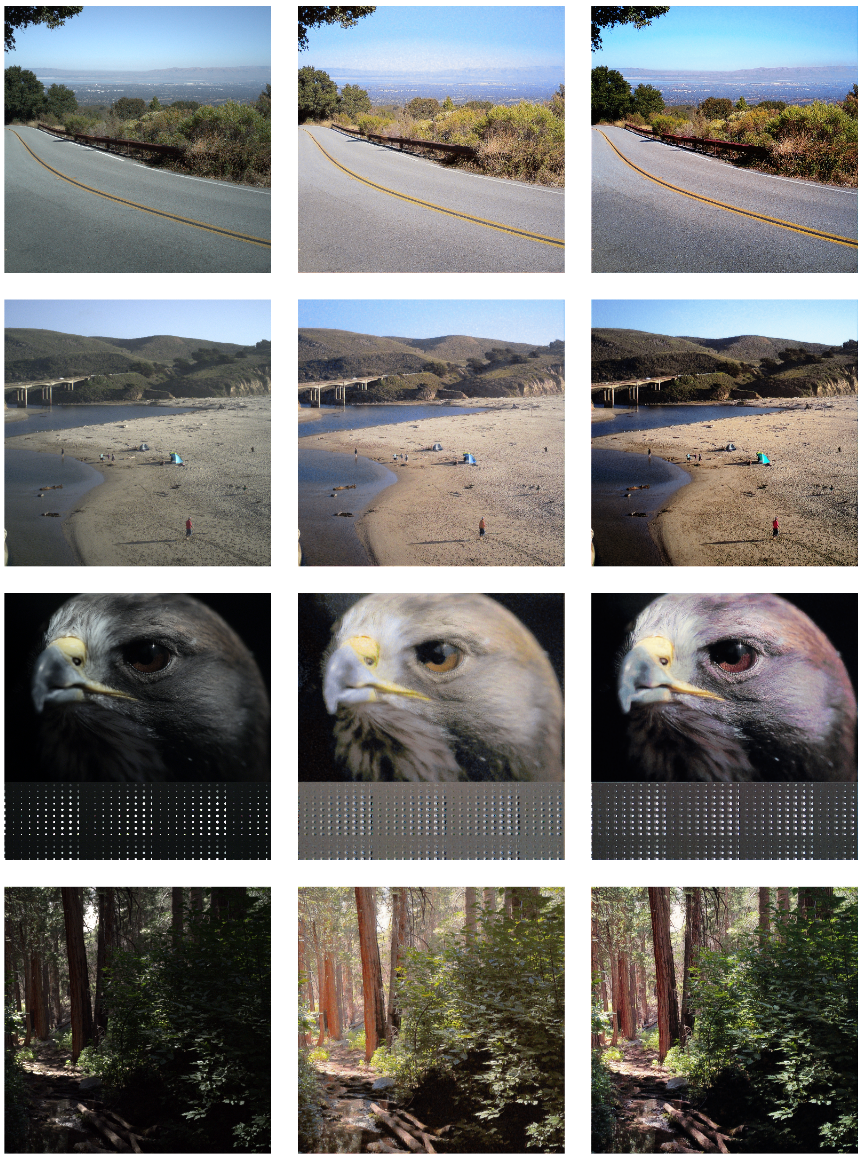


Figure 5.4: From left to right columns: Input images, images generated by the baseline network, images generated by the multi-image network.

		PSNR		SSIM	
Test	**Epochs**	μ	σ	μ	σ
Input Image	n/a	16.57	3.09	0.41	0.21
Baseline Network	17	18.96	2.80	0.55	0.13
Multi-Image Network	17	20.13	2.53	0.62	0.15

Table 5.1: The quantitative results for the input as well as the test runs compared to the ground truth HDR+ image. The PSNR and SSIM are the mean (μ) and standard deviation (σ) of 50 images.

5.5 Performance

When it comes to the performance of the networks during inference, it is clear that it is costly to perform full-size image transformations using neural networks. The maximum resolution image that can be evaluated is 2000 by 2000 pixels, and larger images cannot fit in memory (32 GB) together with the model, as it scales with the input size. The inference of the multi-image network on that resolution takes roughly 2.4 seconds on our CPU, which is within the same order of magnitude in time as performing regular HDR. For comparison, the HDR+ pipeline has a requirement of roughly 300 MB of memory, but gets significant speed-up from using all the memory of the device.

5.6 Limitations

Included in the images are a number of artifacts, introduced by the network as it tries to perform full-scale image improvements.

1. Inaccurate colorization.
2. Introduction of noise.
3. Ghosting.

All of the artifacts are present in both types of network, with no real discernible difference. Examples seen in figure 5.5.

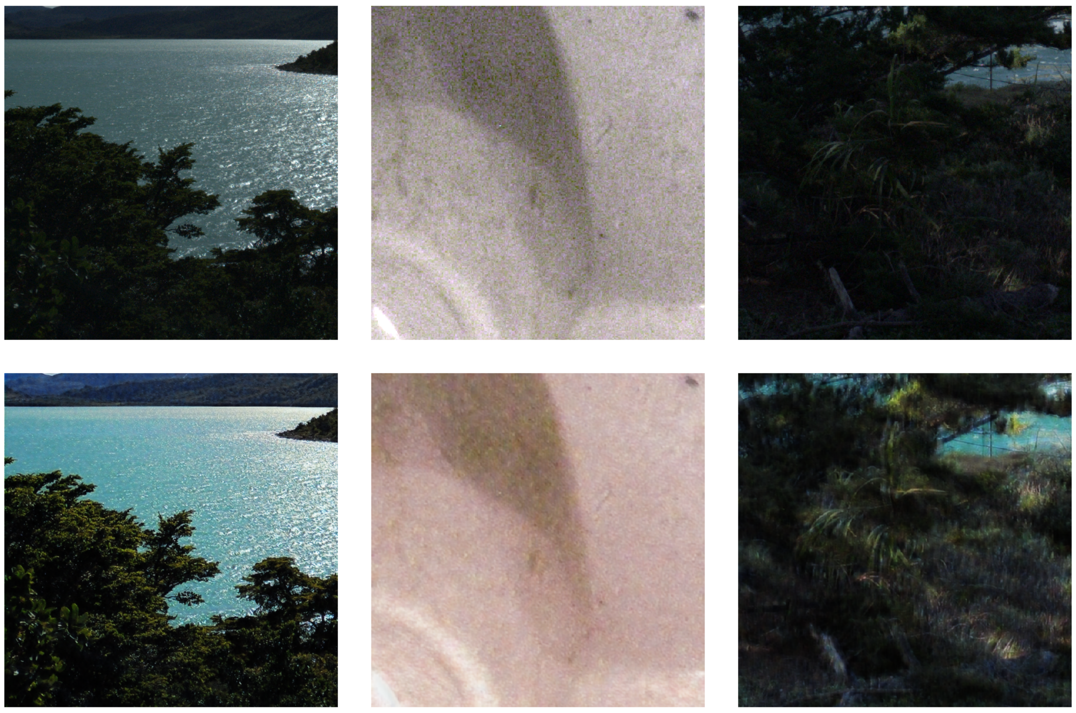


Figure 5.5: Examples of artifacts introduced by the network. Left to right: Inaccurate colorization, noise introduction and blurry, ghosting edges. Top to bottom: Original image, image generated by multi-image network.

Chapter 6

Discussion

The images produced by the networks provide a believable HDR-like effect both for the baseline network and the multi-image network. After a lot of different tests trying to improve the performance of the network, we finally ended up with the results given in chapter 5. It shows that an improvement was made with the multi-image network. The number of epochs trained is quite low and given more time/resources, better results are likely achievable. This also ties into the computational performance of the network, which is quite costly, especially in terms of memory bandwidth, both during training and inference.

There is a general problem of robustness in the generated images. The photographs all acquire the HDR tone mapping, but a number of images contain severe artifacts that visually ruin the image for the viewer.

6.1 Performance

There are a few different ways in which the performance issues can be handled. First and foremost, using the network on a lower resolution would be possible. One application that is quite common is to down-sample an image, apply a transformation and then subsequently up-sampling it again. This does however come with issues on its own, as the up-sampling process is essentially an approximation. One could also reduce the representation inside the network, so that instead of going for a full-size representation in each layer you have smaller dimensions. This is called an *autoencoder network* and has proven to perform well on similar tasks. There is also the possibility of splitting the image into overlapping patches that are run through the network separately and then merged, this would however introduce another image processing step and would not let the network handle global image effects.

6.2 Multi-image Network

The results from the multi-image network are quite promising, with higher PSNR/SSIM values compared to the similarly trained baseline network. The hypothesis that more input information would garner a better result were correct in some regards. There is, however, a disconnect in results achieved and the state of the art, in which the networks have been optimized and tweaked to their limits.

One big issue that comes with multiple images is that there needs to be an understanding between actual differences in the images compared to noise. There is also the consideration of increased computation time and memory usage and the need to take a burst of images instead of simply taking one picture. Also, because of the time difference between the burst images, a fast moving object would in all likelihood give a visually undesirable result.

These findings would have to be verified further after tweaking, as we don't know if there is an upper limit to how well this network structure can perform.

6.3 Limitations

For the limitations given in the results 5.6, there are a number of problems that were introduced by the network that were not present in the original images.

Given the nature of the dataset, which includes a number of noisy images that HDR+ accurately de-noises, the network learns a representation that introduces noise as it tries to accomplish the same result. This can be seen in figure 6.1. This is especially obvious in uniform surfaces of a single color or gradient, such as the sky. This is made worse by the fact that the dataset uses a number of different phone cameras, all likely giving different noise. In order to rectify this, either the dataset can be made less diverse, thereby splitting the different issues of the original image into smaller sub-problems, or a higher batch size can be used in order to teach the network to differentiate between noisy images and clear ones.

The main effect of the dataset that is applied to the images is that of the tone mapping, i.e. the colorization of different parts of the image. While often accurate and heightening the dynamic range of the image, it can sometimes be unreliable in terms of portrayal of reality. Examples of this effect are turquoise seas, as can be seen in figure 6.2. Whether this is an artifact from the dataset or a faulty connection within the network is hard to say, as this might be a desired effect. In order to correct it, more training on more data would bring the result closer to the ground truth. The introduction of a more complex loss would also allow for more accurate targeting of the colors, for example using the color histogram of the images. The current loss compares the color channels individually, which should be sufficient for most color transformations.

Occasionally the network introduces a ghosting effect that duplicates edges in images as can be seen in figure 6.3. Google use a reference frame that is the basis of their alignment, which means that this is the most accurate image in terms of edges. However, as they merge multiple different images with a slightly differing composition, this seems to cause confusion in the network. Theoretically this means that the multi-image network, that also takes multiple input images, would be better at suppressing this type of edge inconsistency

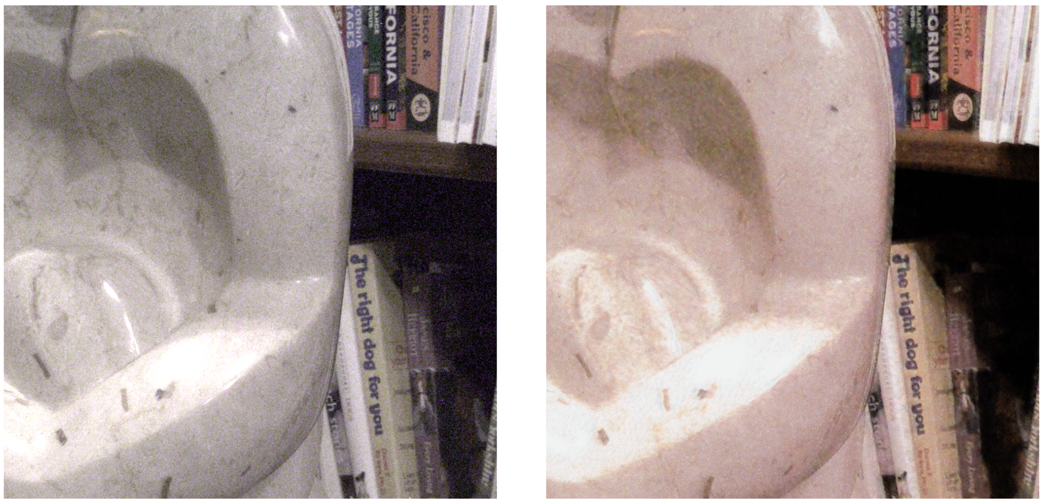


Figure 6.1: The result of the network trying to do noise cancelling where there isn't that much noise to begin with. Left image is the input image, right image is generated by the multi-image network.

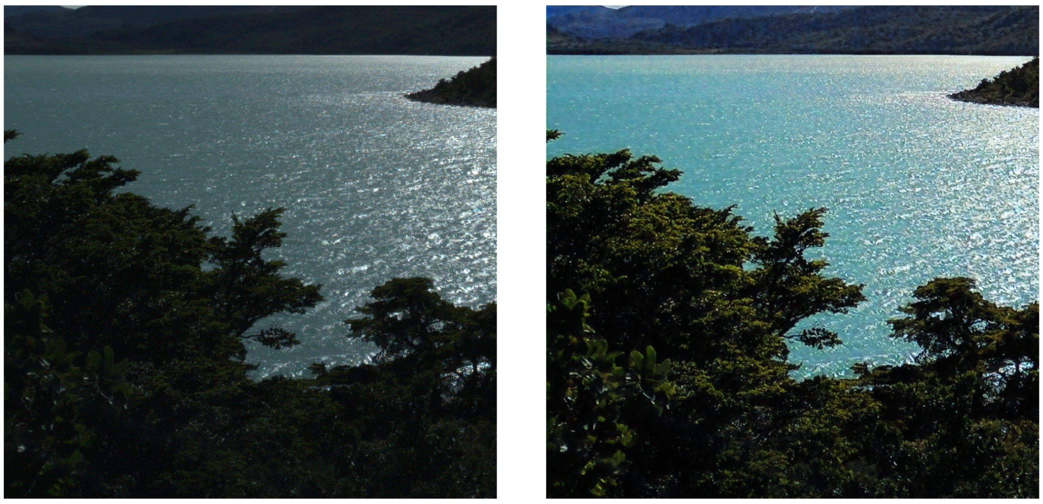


Figure 6.2: The image shows inaccurate and less lifelike colors. Left image is the input image, right image is generated by the multi-image network.

but the differences are hard to confirm. A more edge aware loss would penalize this type of generation, which can be achieved in a number of ways. For example, using SSIM would take the structure of the image into account or possibly an edge detection together with MSE. Permutations of the training images such as rotation and mirroring could also achieve a better result.

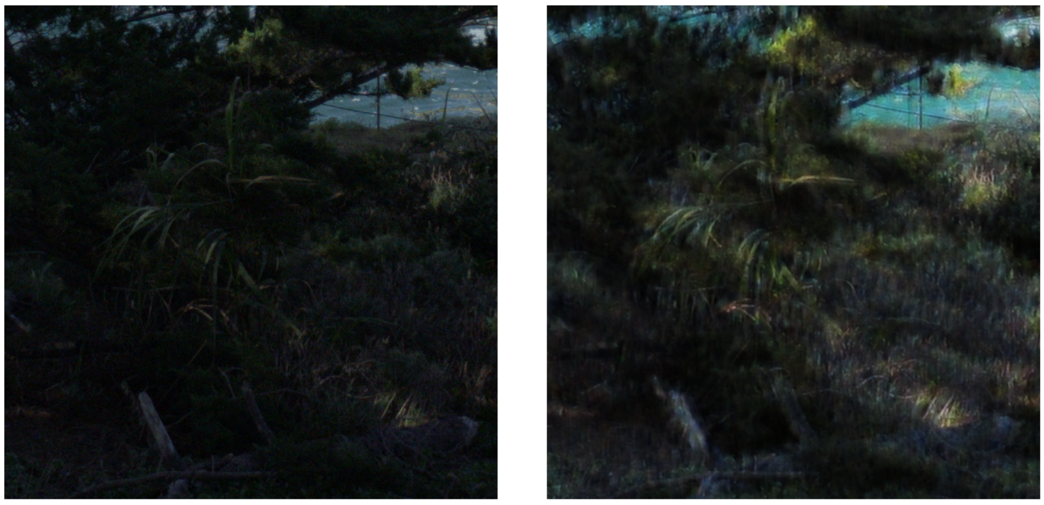


Figure 6.3: Blurry edges introduced by the network. Left image is the input image, right image is generated by the multi-image network.

Chapter 7
Conclusions

While there is a big improvement from the input image to the synthesized image, the quantitative scores are somewhat low compared to the state of the art in similar applications. This can have several different reasons, but the most likely one is that either the dataset doesn't represent the possible image space properly or the application of HDR+ is too complex to model with a network of our topology and size. There exists a problem with robustness, related to the number of artifacts that can be created in the images, which makes this method of applying the desired transformation less desirable. Whereas most images are quite good and generate a positive result, as can be seen in Appendix A, the images that do contain issues are problematic if it were to be used in a commercial application. One could imagine a situation in which the end user could choose from a number of different images to make this less of an issue, but it is not an attractive solution.

As for performance, the requirements on CPU/GPU and memory makes this type of application unfeasible for mobile deployment at high resolutions. There is the possibility of using a server-based structure to run the network, but that would require uploading the image before the expensive computation and then return the image to the mobile device, which would be quite expensive in terms of bandwidth and data usage on the consumer's side.

When comparing to similar work, Gharbi et al. [6] achieved a PSNR of 28.6 using their Deep Bilateral Learning algorithm which takes knowledge of image transformations into account when creating their full pipeline. As their network is only applied on low resolution to generate a bilateral grid containing coefficients for applying the changes in the image on full resolution, their method is significantly faster.

Another quite similar network by Ignatov et al. [12] also achieves good results with the slightly lower PSNR of 21.81, but with an SSIM of 0.947 when improving an image from a Sony Xperia Z cell phone camera and comparing to an image taken by a Canon 70D DSLR camera. However, their problem is fundamentally different as they simply try to improve the quality of an image. Nonetheless their results are very impressive and create images that rival those of HDR+.

7.1 Future Work

The possible improvements that could be made to the network are:

1. Application of network on RAW photos.
2. Increased batch size training.
3. More complex loss function.
4. Changing network topology.

These areas would need to be explored as well as tweaking of hyperparameters, something that is simply out of scope for this thesis. The more parameters that are available for tuning, the more possible combinations exist. While some settings are binary and always have a positive impact, like using RAW images as input, some are complex and continuous, like the choice of learning rate.

With a disregard of performance, there are several areas left unexplored for this type of network. Simply exchanging the dataset and training on another problem would be interesting, having more of a specified application than HDR could certainly provide good results. The most interesting application in our mind would be low-light scenes, attempting to restore information that simply isn't visible to the human eye. In many ways this correlates with HDR, but is a more niche case. More problems to explore would be restoration of noisy or blurry images.

With regards to performance, further exploration research would be best spent on ways to apply the network on a lower resolution as a way to reach a performance that would be competitive on mobile platforms compared to the original algorithm. Therefore, a method for smart upsampling with regards to the input would probably be a good approach to explore.

Bibliography

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Gregory S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian J. Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Józefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Gordon Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul A. Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda B. Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. *CoRR*, abs/1603.04467, 2016.

[2] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[3] C. Chen, Q. Chen, J. Xu, and V. Koltun. Learning to See in the Dark. *ArXiv e-prints*, May 2018.

[4] Michael Eickenberg, Alexandre Gramfort, Gaël Varoquaux, and Bertrand Thirion. Convolutional network layers map the function of the human visual system. *NeuroImage*, 152, 2017.

[5] Gabriel Eilertsen, Joel Kronander, Gyorgy Denes, Rafał Mantiuk, and Jonas Unger. HDR image reconstruction from a single exposure using deep CNNs. *ACM Transactions on Graphics (TOG)*, 36(6), 2017.

[6] Michaël Gharbi, Jiawen Chen, Jonathan T Barron, Samuel W Hasinoff, and Frédo Durand. Deep bilateral learning for real-time image enhancement. *ACM Transactions on Graphics (TOG)*, 36(4):118, 2017.

[7] Vien Gia An and Chul Lee. Single-shot high dynamic range imaging via deep convolutional neural network. *Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, 2017.

[8] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[9] Samuel W. Hasinoff. Introducing the HDR+ Burst Photography Dataset, 2017.

[10] Samuel W. Hasinoff, Dillon Sharlet, Ryan Geiss, Andrew Adams, Jonathan T. Barron, Florian Kainz, Jiawen Chen, and Marc Levoy. Burst photography for high dynamic range and low-light imaging on mobile cameras. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia)*, 35(6), 2016.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. *CoRR*, abs/1512.03385, 2015.

[12] Andrey Ignatov, Nikolay Kobyshev, Radu Timofte, Kenneth Vanhoey, and Luc Van Gool. DSLR-Quality Photos on Mobile Devices with Deep Convolutional Networks. *IEEE International Conference on Computer Vision (ICCV)*, 2017.

[13] Shree K. Nayar and Tomoo Mitsunaga. High dynamic range imaging: Spatially varying pixel exposures. *IEEE Conference on Computer Vision and Pattern Recognition*, 2000.

[14] Nima Khademi Kalantari and Ravi Ramamoorthi. Deep High Dynamic Range Imaging of Dynamic Scenes. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2017)*, 36(4), 2017.

[15] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *CoRR*, abs/1412.6980, 2014.

[16] Dmitry Ulyanov, Andrea Vedaldi, and Victor S. Lempitsky. Instance Normalization: The Missing Ingredient for Fast Stylization. *CoRR*, abs/1607.08022, 2016.

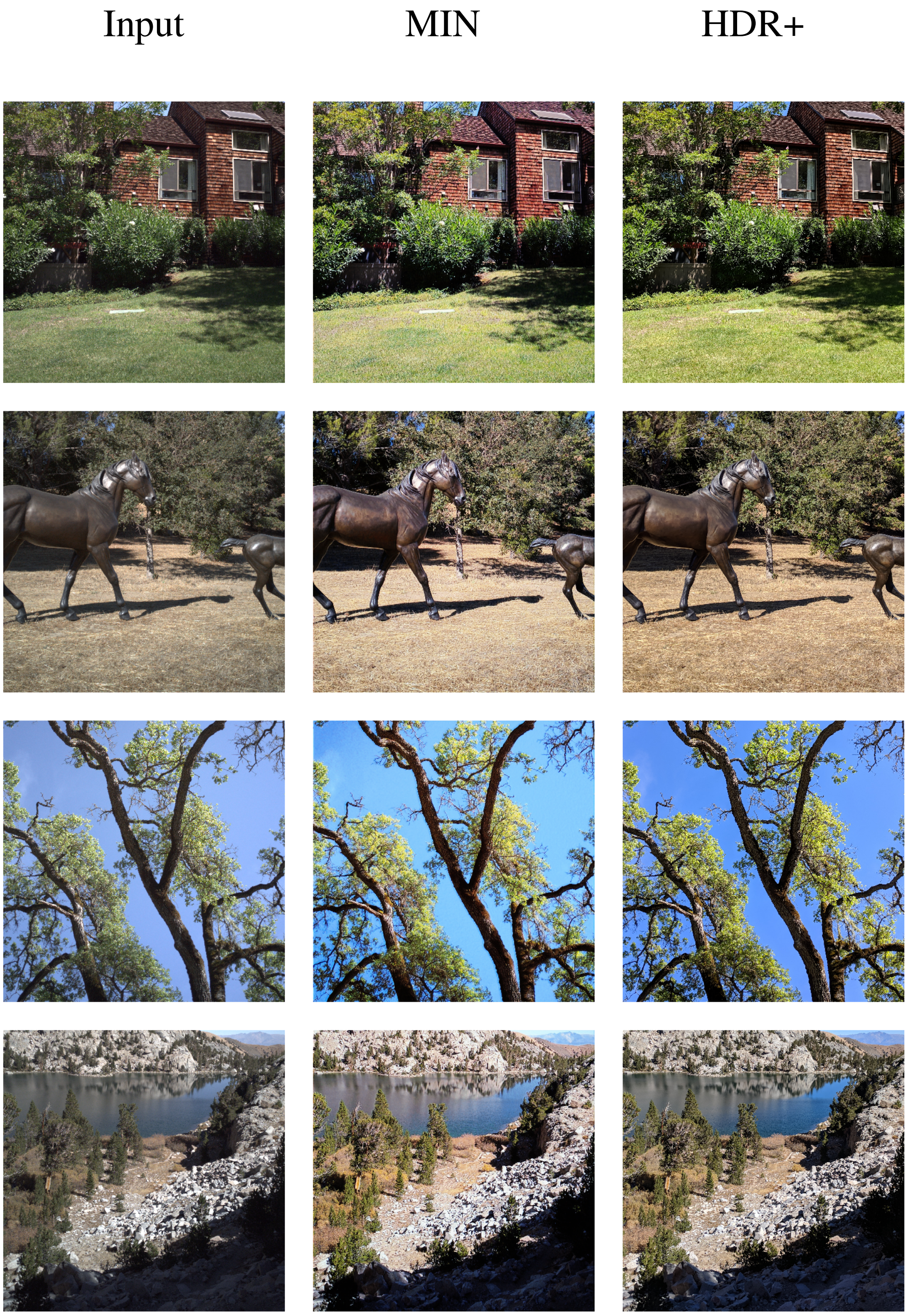
[17] Zhou Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, April 2004.

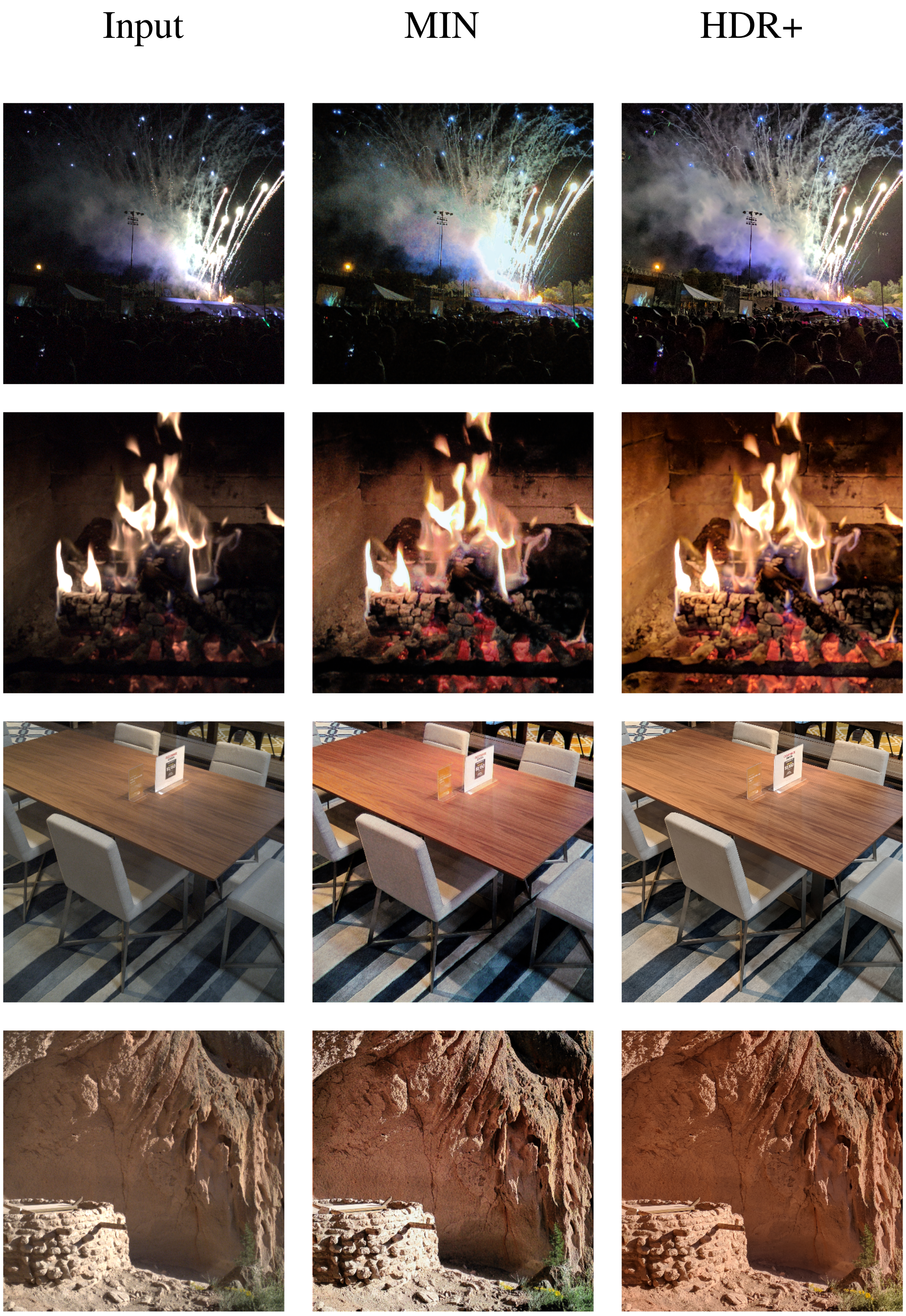
Appendices

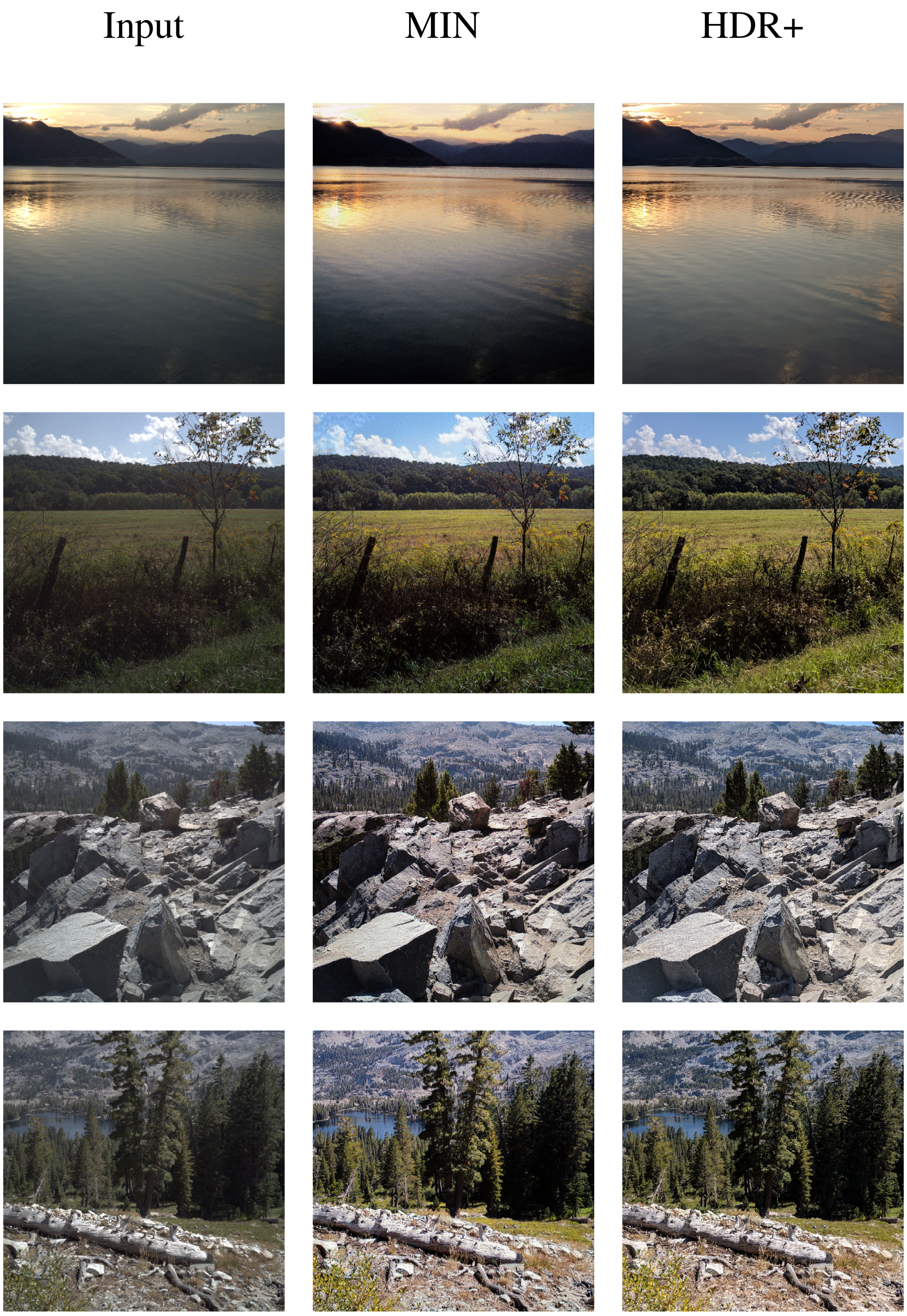
Appendix A
Additional images

This appendix contains a number of extra images to show the performance of the Multi-Image Network (MIN).

The left-most column contains the input images, the middle column is the generated images from the Multi-Image Network and the right-most column is the ground truth HDR+ images.


Input
MIN
HDR+


Input
MIN
HDR+


Input
MIN
HDR+

INSTITUTIONEN FÖR DATAVETENSKAP | LUNDS TEKNISKA HÖGSKOLA | PRESENTATIONSDAG 2018-08-30

EXAMENSARBETE Evaluation of Deep Learning Approaches for High Dynamic Range Imaging
STUDENT Simon Johansson
HANDLEDARE Jörn Janneck (LTH)
EXAMINATOR Michael Doggett (LTH)

Utvärdering av Deep Learning metoder för HDR

POPULÄRVETENSKAPLIG SAMMANFATTNING **Simon Johansson**

Efter-processering blir allt vanligare i bildhantering för att nå resultat som ögat föredrar. Detta arbete försöker uppnå goda resultat för en sån teknik, HDR, med hjälp av neurala nätverk och bedöma dess lämplighet för fullskaliga bildtransformationer.

Förbättring av kamerasensorer ger idag inte så stor skillnad på bildkvalité, speciellt inte på mobila platformar där allt måste bli mindre. Många vänder sig då till olika metoder för att förbättra bilder efter att de har tagits med olika beräkningsmetoder. En sådan metod är High Dynamic Range imaging, eller HDR, som tar en samling av bilder tagna med kort slutartid och litet tidsintervall och slår ihop dem till ett bättre fotografi med större intensitetsomfång. Det finns en rad olika algoritmer för HDR, som ständigt utvecklas och blir bättre, idag finns det till exempel metoder som även reducerar brus.

Fokuset i examensarbetet var att återskapa Google's HDR+ algoritm med hjälp av ett djupt neuralt nätverk och avgöra om detta typ av nätverk var lämpligt, både i mån av kvalité och prestanda. Neurala nätverk har förmågan att återskapa vilken funktion som helst, det är bara en fråga om olika bestämbara parametrar, så som nätverksstruktur, träningshastighet och fullständigt beskrivande data. Eftersom både bildkvalité och prestanda är av intresse blir detta en balansgång, då nätverket inte kan vara för komplext eller för simpelt för att kunna representera informationen i bilderna. Ett försök till förbättring genom att emulera själva HDR algoritmen med att använda flera bilder gjordes också, ledd av hypotesen att ju mer information om själva bilden som nätverket kan ta del av, desto bättre resultat kommer den bidra med.

Förbättring av ursprungsbilden av nätverket

Resultatet visar på att fullskalig bildtransformation är mycket resurskrävande, både i processorkraft och minneskapacitet. Nätverket återskapar en HDR-liknande bild som är trovärdig, med spektakulära färger och hög varians mellan snarlika färger. Det inför dock en rad artifakter, så som introduktion av brus och skuggningar av kanter vilket gör att robustheten av metoden försämras. Nätverket som använder sig av flera bilder genererar bättre resultat än det nätverk som bara tittar på en bild.