YaleNUSCollege

Low-Light Image Enhancement through Deep Learning

Silvia LARA

Capstone Final Report for BSc (Honours) in Mathematical, Computational and Statistical Sciences

AY 2017 / 2018

Yale-NUS College Capstone Project

DECLARATION & CONSENT

1. I declare that the product of this Project, the Thesis, is the end result of my own work and that due acknowledgement has been given in the bibliography and references to ALL sources be they printed, electronic, or personal, in accordance with the academic regulations of Yale-NUS College.

2. I acknowledge that the Thesis is subject to the policies relating to Yale-NUS College Intellectual Property (Yale-NUS HR 039).

ACCESS LEVEL

3. I agree, in consultation with my supervisor(s), that the Thesis be given the access level specified below: [check one only]

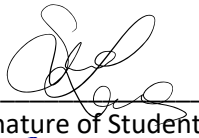
   ☒ Unrestricted access
   Make the Thesis immediately available for worldwide access.

   o Access restricted to Yale-NUS College for a limited period
   Make the Thesis immediately available for Yale-NUS College access only from _____ (mm/yyyy) to _____ (mm/yyyy), up to a maximum of 2 years for the following reason(s): (please specify; attach a separate sheet if necessary):
   _____.

   After this period, the Thesis will be made available for worldwide access.

   o Other restrictions: (please specify if any part of your thesis should be restricted)
   _____
   _____

_____
Name & Residential College of Student



_____ _____
Signature of Student Date

_____ 23/4/2018
Name & Signature of Supervisor Date

Abstract

Vision systems rely on the presence of sufficient light and low levels of noise. Images of night scenes, yet, do not follow these assumptions. In the literature, there are two main algorithms to enhance and denoise dark images, one based on fully connected neural networks and one based on illumination maps. In this project, we create a more effective denoising and enhancing algorithm using Deep Learning. Our model consists of a convolutional autoencoder network with skip connections, following the U-Net [1] architecture. We test the model with different levels of noise and darkness using the MSE and SSIM metrics on the results. In order to analyze the model in a more accurate way, we develop a metric using the results of an object detection algorithm - provided by the Google Vision API. We show that our model can generalize from synthetic noisy images to real-world images, and it can span different levels of noise and darkness. Its effectiveness is limited in the enhancing task for non-synthetic images, but the model excels at the noise removal task in both real and synthetic cases. Results show a significant visual improvement in the denoising task from existing enhancing techniques, such as LIME [2] and LLNet [3] in the regime of highly noisy images.

Statement of Contributions

I have applied an existing neural network architecture, U-Net [1], to the problem of low-light image enhancement. To the best of my knowledge, this approach has never been used before to solve this problem and the tests show that it is more effective than the references in the denoising task. The model was developed by modifying the approach proposed by [3], including convolutional layers and skip connections - as suggested by Prof. Robby TAN.

The model's code was written by me but the U-Net modules were provided by a third party, as acknowledged in the text and code. I wrote testing scripts to produce the image outputs. In order to tackle high definition images, I modified the code kindly provided by Dr Lee Cheelwoo. The scripts which calculate all the metrics used were also written by me.

The Object Detection Metric is my own invention and definition and so was the dataset of real images, which I took and compiled myself.

Acknowledgements

I am grateful to my fellow advisees, Aaron Pang and Pratyush More, for helping me debug my code and the server disruptions, for always having the right suggestion on which software to use and how to use it and for the constant motivation and support they provided throughout this project. Without them by my side this project would have been much more difficult and painful.

I am very thankful to my advisor for his feedback to my progress reports, for helping me define such an interesting problem to solve, and for guiding me through the concepts I did not know about in the field of Computer Vision. He suggested which state-of-the-art models to consider as a reference and he suggested the use of skip connections to reduce the blurriness of the output.

I would like to thank my classmates and the professors who attended my presentations, asking always the right questions and giving much appreciated feedback.

I am grateful to my original examiner, who will not be able to read my final capstone product. He had faith in this project and believed that I was in the right track. I hope he would have been proud.

My most sincere thanks go to my new examiner, who volunteered to assist me and provided very good feedback on my work, demonstrating his maximum availability.

Thank you Subra and housemates, for making me feel at home in Singapore.

Mamma, Babbo, Marco, Marisa, Nonna, Laura and Zia Rosanna: thank you for the constant, selfless and incredible support. Thank you for loving me enough to let me go to the other side of the world, but for always being here when I need you most. Grazie davvero.

Sasha, this would not have been possible without you. For the encouragement, support, patience and hours of listening. I am thankful because you inspire me to always be my better self.

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

Vision systems are becoming increasingly important in modern society. Surveillance, video analytics, and self-driving cars are just a few of the necessary building blocks of the future. Current algorithms assume enough brightness in the input image or video. In reality, vision systems have to work in the night time as well as in precarious visibility conditions. Analyzing visual data procured in the night time is especially challenging due to low light or to the presence of artificial glow (active light).

These problems are currently solved with image enhancing techniques, see Refs. [2], [3]. These algorithms do not use the structural information of the image to obtain the enhanced result. They treat each pixel's contribution equally, regardless whether it's part of the background or foreground or whether it's an artifact or not. Low-light images tend to be noisy: pixels' colors do not transition smoothly, and sometimes are just recorded as gray artifacts. These artifacts are not removable by the mentioned enhancing techniques. As of now, the task of image denoising and

image enhancing have been treated separately.

1.1.1 Project Outline

This project will focus on a solution that both denoises and recovers the daylight condition of the images, with a focus on the denoising task. We use a deep convolutional autoencoder with skip connections (UNET [1]) trained on a synthetically-dark images. This dataset is obtained by reducing the intensity and adding gaussian noise to images from the Cityscapes dataset [4]. The results of the denoising algorithm on this dataset exceeds the state of the art models' results, especially in the denoising task. When tested on real images, the model is able to reduce the noise level but the darkness level does not subside significantly.

1.2 Algorithmic background

Recently, computer vision applications have largely benefited from Deep Learning based approaches. Tasks such as scene understanding [5] and object recognition [6] have obtained their best results using Deep Learning architectures. These methods allow to gain information about higher order relationships that are not usually evident through other machine learning approaches. The only existing approach that uses Deep learning for enhancing and denoising low light images is described in Ref. [3], making this task still an unexplored territory. In Ref. [3], the authors use a Deep Learning architecture called 'Stacked Denoising Autoencoders'. In

order to understand what worked and what needs improvement in their approach, let us understand more about the concepts they rely on.

1.2.1 Autoencoders

Autoencoders are a commonly used deep learning algorithm characterized by their hourglass shape. They map their input into its lower dimensional representation, up to the bottleneck of the hourglass. They then decode this learnt under-complete representation of the input, bringing it to its original input dimension. The network is trained to minimize the difference between input and output. Initially they were used solely for unsupervised learning, but recent applications [7] show their applicability as a pre-training tool for other tasks. Being able to learn the features of an image, autoencoders have also been used as a denoising tool [7]. In our case, we will use their abilities as denoisers and feature-learners.

More mathematically, the idea of autoencoders relies on the presence of two mapping functions, an encoding one and a decoding one. The encoding one, σ, takes an input vector x into a hidden representation h, using a non linear mapping as follows:

$$f(x) = \sigma(Wx + b) \tag{1.1}$$

where W is a weight matrix of dimensions $d \times d'$ and b is a bias vector of size d'. The decoding mapping function takes the encoded result y and maps it back to a vector o in the input space. This mapping follows another non-linear relationship as the one described in Eq. 1.1, with a

new set of W' and b'. The output vector o is not to be interpreted as a reconstruction of the input, but as a abstracted version of the probabilistic distribution that maximises $p(X|O = o)$.

This leads to a loss function that minimizes the reconstruction error:

$$L(x, o) \propto -\log p(x|o) \tag{1.2}$$

1.2.2 Deep autoencoders

The properties of autoencoders can be further enhanced by stacking more than one autoencoder together. These additional layers are then trained locally to denoise corrupt versions of their output [7]. These structures follow the recent successes of deep architectures, which have demonstrated the ability to learn a greater number of parameters (as explained in Refs. [8]–[11]). These approaches showed that it is better to initialize the weights of a neural network by first using an unsupervised criterion. Through this pre-training procedure the network learns the high-level representations. Using these pre-trained set of weights when performing the supervised back-propagation through gradient descent leads to more generalizable and performant solutions.

1.2.3 Optimizing the gradient descent

Another important factor that plays a role in the performance of a deep model is the type of optimization techniques. Due to the nonlinearity of the loss function we are optimizing, different approaches can obtain significantly different results. The understanding of which technique works

better for our task will guide us when analyzing the current models in search for possible improvements in their architecture.

In general, deep architectures update their weights through a back-propagation procedure. This procedure changes each weight in the network in the opposite direction of the gradient of the objective function $J(\theta)$ that we want to minimize. Following this direction, we move down the potential landscape based on a step $\eta \nabla_\theta J(\theta)$, called "learning rate".

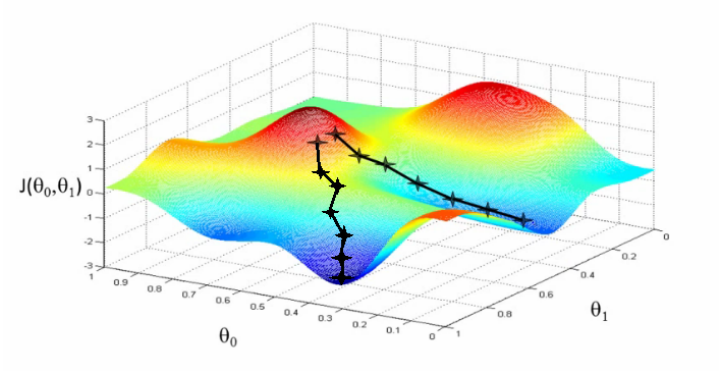


FIGURE 1.1: Visualization of Stochastic Gradient Descent as shown in Ref. [12]

Figure 1.1, shows the gradient descent path chosen for a loss function on a 2D projection. The two paths show the variability for different learning rates and optimizations.

Whenever a specific method of optimization of the gradient decent was specified, we used optimizers to adapt the learning rate.

$$\theta = \theta - \eta \nabla_\theta J(\theta) \tag{1.3}$$

We tested our algorithms using both "adadelta", presented in Ref. [13], and "adam", introduced by Ref. [14]. These techniques optimize both the

learning rate and the momentum, an additional factor γ that dampens oscillations in the descent as follows:

$$v_t = \gamma v_{t-1} + \eta \nabla_\theta J(\theta), \tag{1.4}$$

$$\theta = \theta - v_t \tag{1.5}$$

Chapter 2

Existing approaches

Low-light image enhancement is a problem that has been tackled through many different approaches, such as histogram equalization (HE) ([15]–[17]), contrast-limiting adaptive HE techniques [18], gamma adjustment, and 3D Block matching [19]. In 2017, two new different approaches obtained better results than the aforementioned literature: LIME [2] and LLNet [3].

2.1 LIME

LIME [2] is a non-deep-learning based method to solve the problem of enhancing low light images.

2.1.1 Architecture

This algorithm is a Retinex-based method: it uses an estimation of the illumination map to enhance the image. A low light image can be defined as follows:

$$\mathbf{L} = \mathbf{R} \circ \mathbf{T} \tag{2.1}$$

where **L** is the captured image, **R** is the recovered image, **T** is the illumination map and the symbol $\circ$ represents element-wise multiplication. The illumination map records the maximum intensity of each pixel in each RGB channel. The initial illumination for each pixel x is estimated as follows:

$$\hat{\mathbf{T}}(x) = \max_{c \in \{R,G,B\}} \mathbf{L}^c(x) \tag{2.2}$$

Such a definition for **T** guarantees that the recovered image will not be saturated, since:

$$\mathbf{R}(x) = \frac{\mathbf{L}(x)}{\max_c \mathbf{L}^c(x) + \epsilon} \tag{2.3}$$

where ϵ is a small constant added so that the denominator is never zero. In order to obtain a recovered image that is smooth and which preserves the structure of the original image, one needs to solve an optimization problem:

$$\min_{\mathbf{T}} \|\hat{\mathbf{T}} - \mathbf{T}\|_F^2 + \alpha \|\mathbf{W} \circ \nabla \mathbf{T}\|_1 \tag{2.4}$$

where α is a balancing coefficient, and the two norms are the Frobenious and ℓ_1 norm respectively. The elements of the last norm, **W** and ∇**T** are a weight matrix and the derivative of the illumination map. More than one weighting strategy is proposed by [3], implying the need to optimize for both the weights and the additional parameters. Once we have obtained and refined the illumination map **T**, it is possible to generate our enhanced image **R**.

2.1.2 Drawbacks

- Estimating the illumination map for each image does not allow the model to consider general trends present in many images. In other words, the algorithm does not *learn*, but it just applies a standard recipe.

- LIME does not tackle substantial levels of noise, which are often present in low-quality low-light images. Its denoising capabilities are probably going to be limited, as confirmed by testing it on the MNIST dataset.

2.2 LLNet

LLNet is a model that solves the low-light image enhancement problem using a deep autoencoder, as mentioned in the introduction.

2.2.1 Architecture

The architecture presented by [3] consists of a deep, fully connected autoencoder with the following structure: 2000 hidden units in the first encoding layer, 1600 in the second, 1200 in the third bottle-neck layer, see Fig. 2.1. The first decoding layer has 1600 units, the second 2000 and the output layer has 289 neurons (17x17). The activation function used in both decoding and encoding layer is a sigmoid:

$$\sigma(s) = \frac{1}{1 + e^{-s}} \tag{2.5}$$

Each DA is trained by error back-propagation to minimize the sparsity-regularized reconstruction loss. This is a regularized type of the typical reconstruction loss that is used for denoising autoencoders: it represents the difference between the input and the output image. The lower the value of the loss function, the closer the images are. The network is trained in two phases: a first pre-training phase and a second fine tuning phase, according to the standard stacked autoencoder procedure suggested first in [7]. The pre-training stage is performed for 30 epochs, with rates $\alpha = 0.1$ for the first two decoding layers, and learning rate $\alpha = 0.01$ for the third decoding layer. The fine-tuning phase uses the learning rate $\alpha = 0.1$ for the first 200 fine-tuning epochs, and $\alpha = 0.01$ afterwards. The training stops once the validation error is less than 0.5.

2.2.2 Drawbacks

- Fully-Connected frameworks are generally weaker than their convolutional counterparts. In this case especially, the small number of neurons sets a limit to the number of patterns that can be found. Moreover, the lack of convolution limits the detection of pixel-to-pixel relationships.

- The network in the paper has been trained with artificial noise, both Gaussian and Poisson. Thus, the results are probably not generalizable to real dark images. We will test this claim by using this algorithm on datasets such as Cityscapes [4], and our own test dataset (see Sec. 5.1).

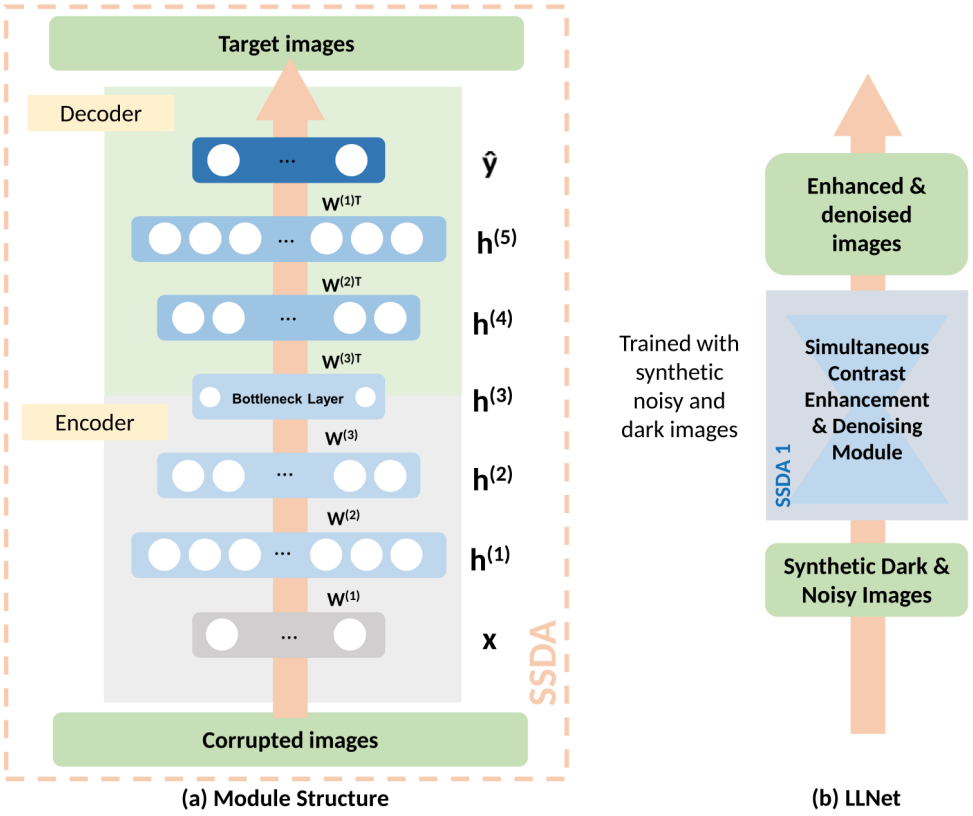


FIGURE 2.1: This is the structure of the LLNet [3]

- If trained using an optimizer (see Sec. 1.2.3), instead of the process of fine-tuning described above, and tested on darkened and noisy MNIST, it performs worse (in terms of the accuracy of the reconstructed images) than a simple convolutional network. More details are listed in Sec. 3.

Fully connected networks (FCN) flatten their input into a one dimensional array. Thus, the input image loses information about neighboring pixels - or spatial information in general. Our first step to improve this architecture was to substitute the fully connected layers with convolutional ones (details in Sec. 3.2). The next step will be to find a deeper convolutional architecture that will improve the denoising capabilities.

Chapter 3

Experiments on simple images

In this chapter we will improve on the LLNet architecture by testing its results on a dataset of hand-written digits. These relatively simple images will give help us benchmark the denoising capability of LLNet and our modifications to the original architecture. By the end of the chapter, we will have a new model which performs better than LLNet in the task of denoising and enhancing synthetic low-light handwritten digit.

3.1 MNIST Dataset

In order to model dark and noisy images, we produced a synthetic dataset based on adding noise and darkening to MNIST. Such simple images allow us to understand the fundamental working principles of each algorithm, separating the enhancing task from the denoising task. The image modification process worked as follows:

$$Dark(Noise) = ((N(0,1) * n)) + I * d\,, \tag{3.1}$$

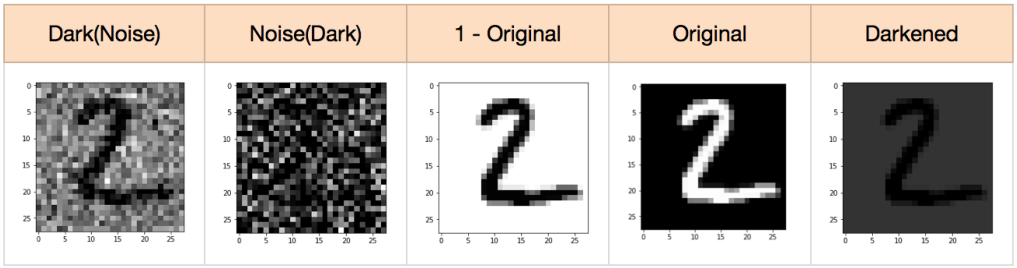


FIGURE 3.1: Table comparing the adding noise procedures on the MNIST dataset. The input to our model was the second image to the left.

where n is the noise factor, d is the darkening factor and I is the original image. The result of reversing the order of darkening and noise applications is shown in Fig. 3.1. The general behavior on such experiments should model some of the most important challenges that solving the dark image enhancement problem will present. This test will be our initial benchmark to test the denoising capabilities of each algorithm, together with their ability to produce a sharp output.

3.2 LLNet: From fully-connected to convolutional

3.2.1 Fully connected

In order to reproduce the results presented in LLNet [2], we adapted the number of layers to the size of our input, so that a the ratio between subsequent layeres is the same as the authors' recommendation. To obtain a baseline of the performance of this architecture, we omitted the pretraining stage and calculated the weights through SGD backpropagation, using the *adadelta optimizer*.

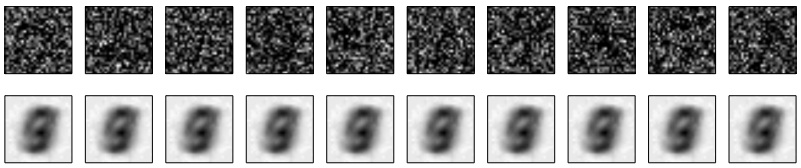


FIGURE 3.2: LLNet paper architecture - baseline result

LLNet Implementation

We implemented the LLNet architecture as described in [3], adapting the geometry to the input of the MNIST dataset. We decided to keep the ratio between the number of hidden units and the number of pixels in the input image equal to the one used in the paper. For example, since our input has size 28x28, our architecture has 117x28 hidden units on its first layer, while the LLNet implementation has a first hidden layer of 117x17 units (their input size is an image of 17x17 pixels).

Experiment 1 We chose to use the mean squared error as a loss function. In order to obtain a baseline on the general performance of the module, we adopted an optimizer (Rmsprop - optimizer proposed by Hinton in [11]) instead of the pretraing - finetuning process described in the paper. We ran the training for 100 epochs; the loss was monotonically decreasing at a very slow rate. Thus, we believe a higher number of epochs with the current parameters would not benefit the results.

The results are given in Fig. 3.2. All recovered images look very similar: a blurry white background and a blurred unrecognizable number. The output seems independent of the input, suggesting that the network

is not differentiating between the various inputs. This result seems different from what was claimed in [3]. This difference is most likely due to the lack of pretraining. Thus, in order to benchmark the performance of LLNet we will reproduce [3] in its entirety.

3.2.2 Convolutional LLNet

Fully-connected neural networks flatten the input image into a one-dimensional array. The input array does not preserve the structure of the original image, and all information about each pixel's neighborhood is lost. In order to preserve this structural information, we constructed a convolutional autoencoder, following Chollet's explanations [20]. The architecture is structured as follows: the encoding network is composed of three layers (Convolutional 2D, Maxpool, Convolutional 2D); the decoding network is composed of 5 layers ([Convolutional 2D, Upsampling] x 2, Convolutional 2D). All activation functions are reLu except the one on the last layer, which is a sigmoid. We trained our model with the following parameters:

- 100 epochs from experiments from 1 to 3.
- 128 batch size.
- Adadelta optimizer.
- 0.2, 0.2 : noise and darkening factor applied to N(0,1)

Training time:

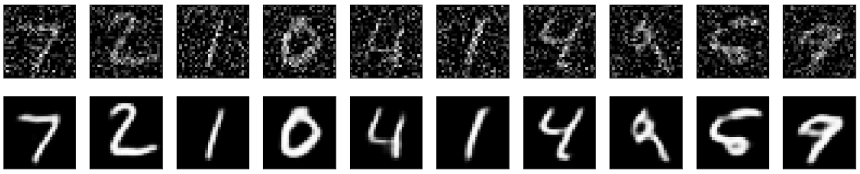


FIGURE 3.3: Output from Experiment 1

- CPU: about 3.5 minutes per epoch, for a total of 6 hours for 100 epochs.
- About 10 minutes for 100 epochs on GPU.

Experiment 1 Our MNIST input had a dark background and white numbers. After adding some Gaussian noise, we run a convolutional decoder and encoder. We then tested the trained network on unseen noisy MNIST images. The decoded images show recognizable digits 3.3. The images are not blurry and the noise has been removed.

Experiment 2 In this case we both added the noise and then darkened the image. There is some sort of resemblance between the decoded digits and actual digits. It might also be true that more epochs might have benefitted this experiment. since the loss in the last epoch was still 0.19,

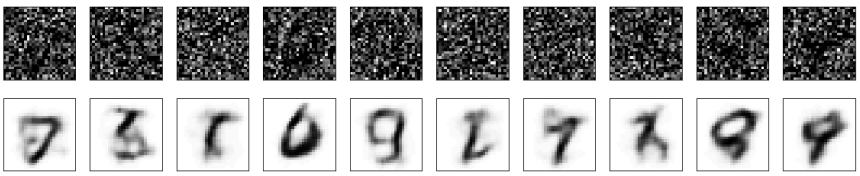


FIGURE 3.4: Output from Experiment 2, using 200 epochs.

which is quite high compared to the loss in the other experiments (0.09 in the case of the first experiment). The loss was going down very slowly, but it was monotonically decreasing. This gives hope that we still have not reached the full potential in learning, and that training for a few more epochs would increase the results. The next experiment will be a re-run of this experiments with a higher number of epochs.

Experiment 3 We increased number of epochs to 400. The loss was still going down monotonically, but with a slower rate.

3.2.3 Deep Convolutional LLNet

In order to estimate how adding layers would affect the perfomance of the model, we modified the convolutional network adding a few layers. The structure of the new network is as follows: the encoding netowork is

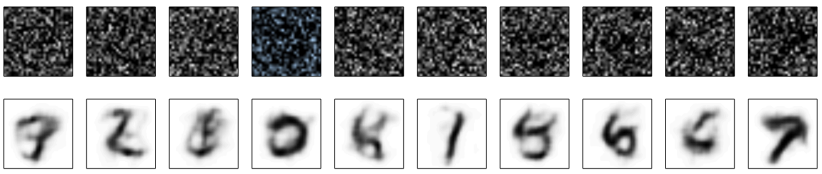


FIGURE 3.5: Output from Experiment 3, using 400 epochs.

composed of 8 layers ([Convolutional 2D, Maxpool]x4), with a dropout of 0.08; the decoding network is composed of 8 layers ([Convolutional 2D, Maxpooling] x 2, [Convolutional 2D, Upsampling] x 2). All activation functions are reLu, except for the one on the last layer, which is a sigmoid. We trained our model with the following parameters:

- 200 epochs.
- 128 batch size.
- 0.2, 0.2 : noise and darkening factor applied to N(0,1)

We tested both the network using *adadelta* and *adam* optimizers.

Experiment 1 Using adam optimizer led to the results presented in Fig. 3.6. The numbers are more recognizable than in the shallower architectures. Some of the number that are closes to the truth (shown in Fig. 3.3),

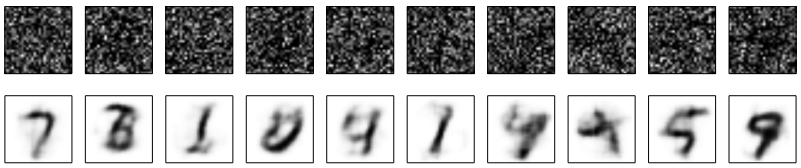


FIGURE 3.6: Output from using adam optimizer on the deep convolutional network

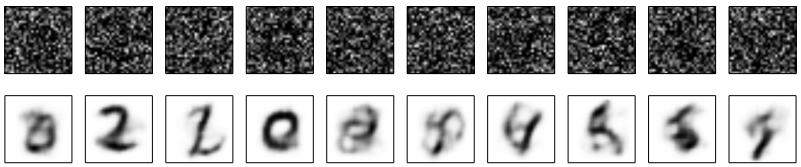


FIGURE 3.7: Output from using adadelta optimizer on the deep convolutional network

some others, such as the second number, resemble more another digit rather than their original one.

Experiment 2 Using adadelta optimizer led to the results presented in Fig. 3.7. In this case different numbers are better recognizable, specifically the second and forth numbers are closer to the truth (shown in Fig. 3.3). The numbers that were well recognizable in the previous experiment, such as the first number, the third, the sixth and the last, are here more smudged and less clear.

It is difficult to compare the two outputs since the images were generated with different additive gaussian noise. For the sake of a fair comparison, we should test them on the same image and then compare the output. Another approach would be to feed the output into the same MNIST classifier, and determine how many of the digits are recognized truthfully

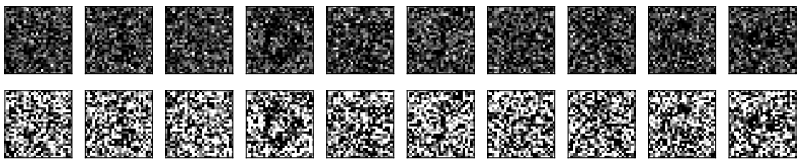


FIGURE 3.8: This first row shows the noisy MNIST images input to the LIME algorithm, the second row shows the output.

by the classifier. This would provide a numerical measure of comparison between the various experiments using MNIST. However, considering this experiment in context with the greater scope of this project, we decided to tackle this problem by making the test more complete. To achieve this, from Ch. 4, we will be testing our models on a more complex image dataset, Cityscapes.

Both results are closer to the original MNIST than the simple convolutional network architecture tested before. This trend suggests that deepening the architecture allows to learn some features before ignored.

3.3 LIME

We tested the LIME algorithm using the code provided by the authors at: https://sites.google.com/view/xjguo/lime.

3.3.1 MNIST Dataset

We tested the LIME algorithm using the same darkened MNIST dataset used to test the previous algorithms. The results are shown in Fig. 3.8. The output images are a black and white version of the greyscale input.

While the intensity of the pixels changes, the noise is not reduced. The original non-noisy MNIST figure is as perceivable in the output as it is in the input. Therefore, LIME's denoising capabilities are very limited, at least at this level of input noise.

3.4 Experiment Results

Dark images are an interesting challenge because they combine two issues: the presence of noise and the darkness aspect. An optimal algorithm addresses both problems. The experiments presented in this chapter analyzed the performance of the models on simple images. The results showed that in the case of the MNIST dataset, LLNet does reduce the amount of noise and darkness, even though it creates blurry outputs, while LIME reduces darkness effectively but it does not reduce noise.

Since LIME does not show denoising capabilities, we will focus on the optimization of our deep convolutional model's architecture based on LLNet. Having determined the capabilities of the models on toy images, we will now focus on their enhancing abilities on more complex image datasets, such as Cityscapes.

Chapter 4

Experiments on More Complex Images

Experimenting on a hand-written digits can only help us this far. In order to quantify the improvements of further modifications we need to test our models with more complex and realistic images. We will do this by synthetically darken the Cityscapes Dataset. Testing on such dataset will highlight the problem of blurring, which we will resolve by modifying our model to include skip connections. This will be our final model, which we will test qualitatively and quantitatively in different levels of noise and darkness.

4.1 Cityscapes Dataset

4.1.1 The Dataset

Cityscapes [4] is a dataset which focuses on semantic understanding of urban street scenes. The images depict a variety of urban scenes and the dataset organizes them by city. We use about 1500 images out of the 25000

FIGURE 4.1: Cityscape original image and its corresponding noisy image.

included in the dataset, using 25% of them for testing. While Cityscapes is mainly intended for assessing the performance of image segmentation algorithms, it constitutes a strong support for algorithms that need large volumes of weakly or non-annotated data.

These images are subsequently darkened by decreasing the pixel intensity by a fixed amount, the darkness parameter, and gaussian noise is added, as per the aforementioned eq. 3.1. The results are shown in Fig. 4.1

4.1.2 Adjusting to higher definition images

The portion of the Cityscape dataset we used contains images with sizes 1024 x 2048 pixels. The model we were using, however, would occupy an impractical amount of memory to process such high definition images. To do so, we would have to increase the amount of hidden units to match the image definition. Sufficiently increasing the number of the

hidden units would increase the model size significantly, exceeding our GPU memory limits.

Resizing the image

The simplest approach to address the issue is to simply resize the higher definition images into a smaller size, reducing their resolution. However, this approach would not allow us to investigate denoising capabilities in the details of the image. Moreover, noise and low resolution are both obstacles in image enhancement, and it could be difficult to differentiate visually the one from the other when observing a processed image. We will be using this method to test some of the models limitations and as a proof of concept for the model itself.

Stitching the patches

An alternative to reducing the resolution is dividing up the image into smaller patches which the algorithm can process. The stitching of the image would then happen on the processed patches, constructing the original-sized image.

Patches in testing A possible approach consists of using the patching method only in the testing phase. The model would be trained with resized images, and the testing would then happen on the patches, which will subsequently be stitched together. Even though this approach would be easy to implement, given a patching and stitching function, there are some significant drawbacks. The model was trained using images which

have common characteristics, such as the structure of an urban space as seen from a car. When the patching is performed, the smaller produced images would be only segments, or details, of these more complex images. These segments would not hold the same structural information as the whole cityscape image, and the model might be significantly affected in its prediction abilities. For example, the model might have developed the ability to map the image of a noisy car into the image of a clean car. However, it would not be as good in denoising elements that it has been exposed to more rarely, such as the image of a book. Moreover, using the patching at this stage could cause edge effects at the boundaries between patches, which could create artifacts in the final image. Since this is the simplest method, we will adopt this method first when in need to process a higher definition image.

Patches in training A more thorough approach would consist of using the image patches in the training process, as well as in the testing phase. This method has the advantage of testing the same type of images as the ones present in the training. However, the variety of images that would be included in the training process is much higher than the original variety of Cityscape. These images do not have as much in common as the whole images do. From a more practical standpoint, this approach would determine a significant increase in training time (proportional to the number of patches we would split the image into), and it is more time-consuming to implement. A possible additional extension to this

FIGURE 4.2: Input and output of the deep convolutional autoencoder model on a Cityscapes image.

project would be to apply this method on the high definition images encountered. However, for the rest of this thesis the patching technique will be used only in the testing phase.

4.2 Fighting blurriness: U-Net

4.2.1 Blurry Output

Now that we have defined a dataset to test our latest model, let us proceed with the analysis. We modified the existing deep convolutional autoencoder to take an input of 256 x 256 pixels, from the existing 28 x 28 required by MNIST. We adapted the number of hidden layers to maintain the same ratio in number of units as the previously tested model. The results look as presented in Fig. 4.2. Despite the low resolution of the image, we can see that the output itself is blurry. This is a known effect of fully convolutional autoencoders, since some spatial information is lost in the compression and decompression process [21], causing blurry output.

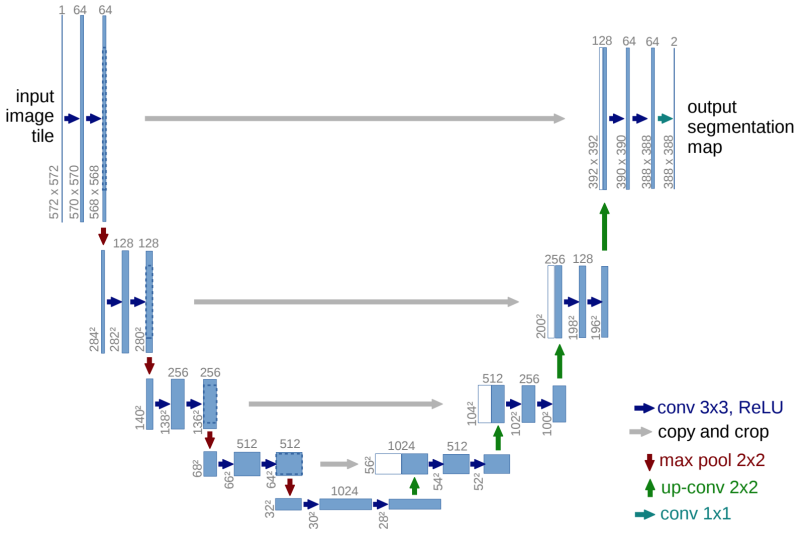


FIGURE 4.3: U-Net architecture. White boxes represent copied feature maps and the arrows represent the different operations. [1]

Skip Connections

In order to recover some of this spatial information we can merge features skipped from the earlier layers in the contracting path, connecting them to layers on the expanding path. Skip connections have been used in similar tasks, such as image segmentation, to improve robustness against appearance changes [22]. In our case, our hypothesis is that the robustness would be translated into the reduced blurriness.

4.2.2 U-Net

The U-Net architecture is a fully convolutional autoencoder which presents a very characteristic shape of a letter 'u', Fig. 4.3, due to its symmetry. The main differences between the deep convolutional autoencoder we have

and the U-Net architecture is that pooling operators are replaced by up-sampling operators, to increase the resolution of the output [1]. The skip connections are then set up between symmetrically opposite layers, as shown in Fig. 4.3.

U-Net Implementation

Our implementation of the U-Net architecture followed the adjustments to U-Net found here in the following source code: https://github.com/pietz/unet-keras. These modifications include the use of padded convolutions, and a specified Dropout rate of 0.5. Unless otherwise specified, our experiments follow these general fixed parameters:

- Batch size: 16
- Number of epochs: 70 for 128 x 128 images, 100 for 256 x 256
- Optimizer: adam
- Loss: binary cross entropy
- Image size: 128 x 128 or 256 x 256

Binary cross entropy is a measure of the error of the network. The function measures the result of a classification model given that its output is a probability value between 0 and 1. The entropy increases the further away the predicted probability is from the label. We chose such loss function after comparing its results with other commonly used functions.

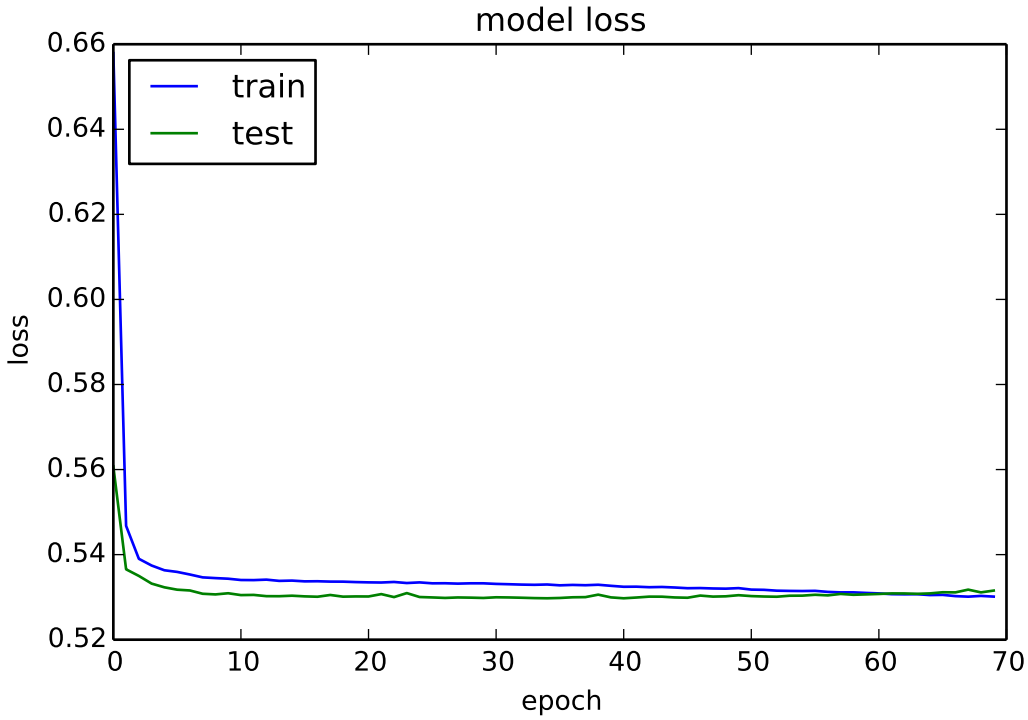


FIGURE 4.4: Loss plot for U-Net Experiment for a 128 x 128 image.

The number of epochs was determined by comparing the loss plot for the training and validation sets. The validation set loss plot generally reached a minimum around epoch number 65, to then increase again - as shown in Fig. 4.4 . This is the point in which the model stops being generalizable and starts overfitting to the training set. We then decided to stop the training at that point. When we tested 256 x 256 images, the minimum point of the validation loss was reached around epoch number 100.

4.2.3 Results

Our hypothesis predicts that using the U-Net architecture will reduce the blurriness of the output. The following plot, Fig. 4.5, compares the outputs using the U-Net approach and the deep convolutional autoencoder model.

The model without skip connections has been trained for 200 epochs, twice as long as the other one, with images of the same noise level. The output from the U-Net model is much more sharp, much cleaner and precise than the other output. In terms of running time, training the U-Net model took about 3 hours, trained on 1 GPU on a GeForce GTX 1080 machine with 8 GB RAM. The purely convolutional model takes about 30 minutes to run in the same conditions. It is clear that the U-Net architecture is better at solving our problem than the model without skip connection, and our hypothesis was verified.

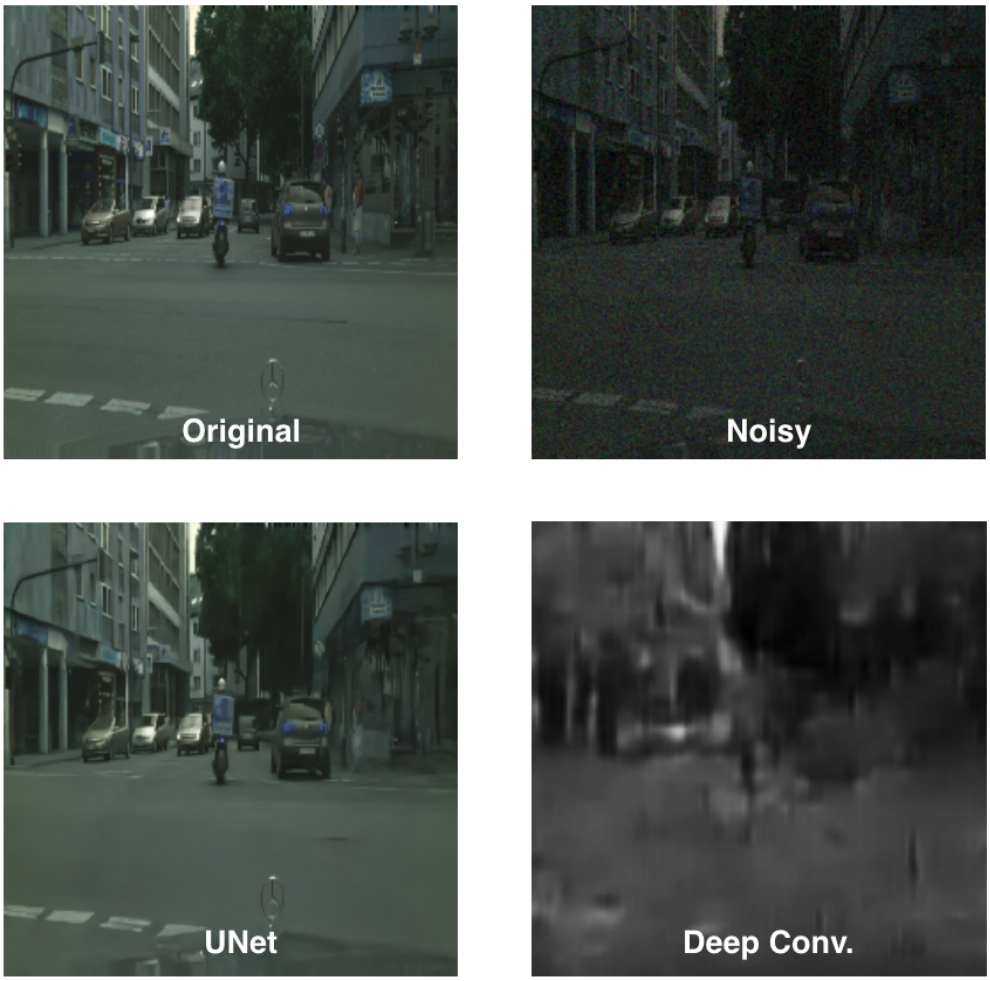


FIGURE 4.5: A comparison between the U-Net and Deep Convolutional approach (on the second row), given as input a 256 x 256 pixel Cityscape image (on the first line)

4.3 Different parameters

We now have a model that seems to perform the task required. In order to look for additional improvements, we need to find the limitations of said model. In our case, we are interested in denoising low light images, in their darkness and noise reduction aspect. These two aspects are not always achieved by the same model, as shown by the fact that LIME has shown to be an effective enhancer but a bad denoiser. Thus, we will test the limitations of our model in its susceptibility to noise and darkness. We can simply do this by changing the noise and darkness parameters in the noise equation: eq. 3.1.

4.3.1 Changing the noise level

In order to test the behavior of the model with different levels of noise, we created three images with a fixed darkness constant but with different noise constant: low, medium and high noise. We then used these images to test models that had been trained to denoise lower or the same amount of noise. Figure 4.6 shows the results of this test.

Results As expected, the quality of the image decreases with the input noise. When the model is tested on the noise level it's been trained on, the results are best (training on 0.5 and testing on 0.5, training on 1.0 and testing on 1.0). However, the between the two, the one with the lower noise presents better quality output. When the level of noise is higher than what the model has been trained on, the performance is poorer. Interestingly, in the case in which the testing noise is lower than the training noise

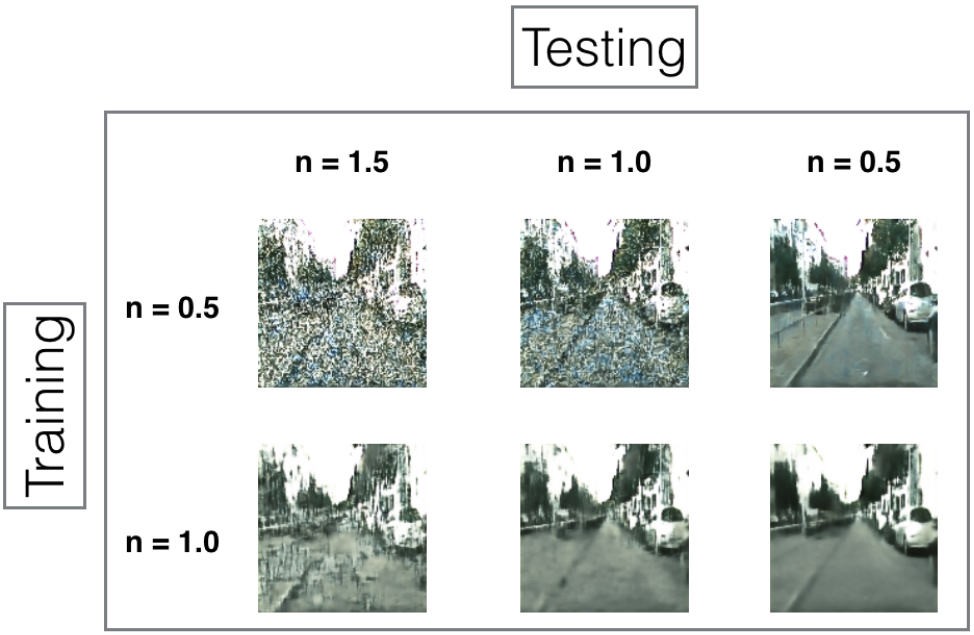


FIGURE 4.6: This table shows the response to changes in noise level in the testing given a certain noise level during the training.

(training on 1.0, testing on 0.5), we notice that the image is less detailed than the image decoded by the alternative configuration. The details of the trees are treated as noise, and therefore removed. When the model is tested on a level of noise it has not been exposed to, its performance is significantly worse - as expected.

4.3.2 Changing the darkness level

Similarly to the noise case, we created three different darkness level images (very dark, dark, low darkness). We then tested them on three different models which had been trained with three different levels of darkness (dark, low darkness, no darkness). The results of this test are shown on

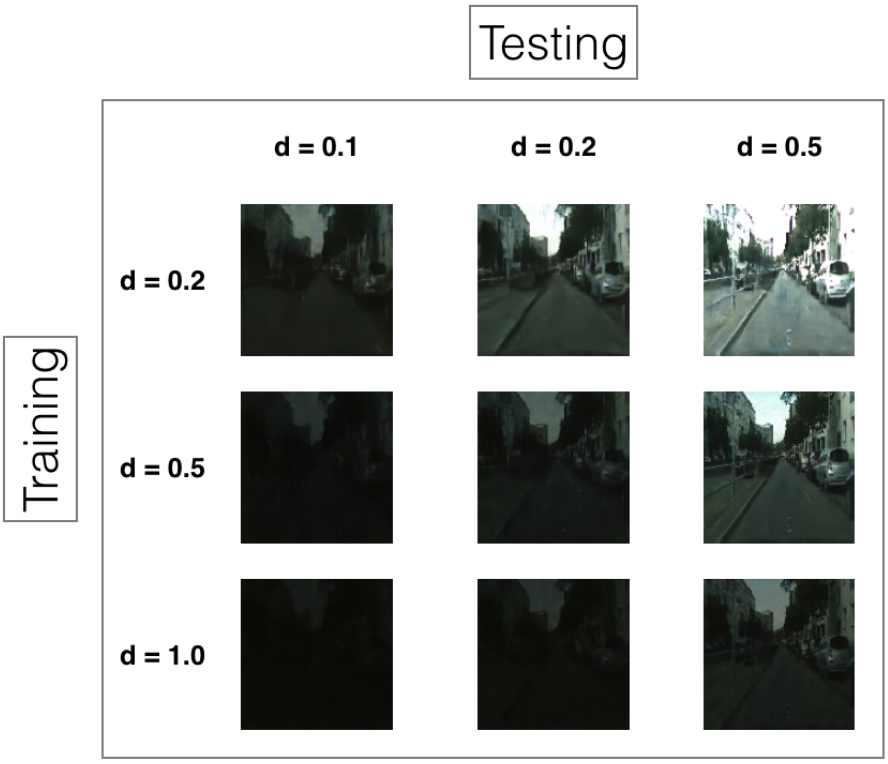


FIGURE 4.7: This table shows the response to changes in darkness level in the testing given a certain darkness level during the training.

Fig. 4.7

Results The table of results shows a symmetry along the diagonal, which shows that predictably the models that were trained to remove higher levels of darkness enhanced the input image better than those trained with lower levels. The model that was trained without the brightening ability is there as a baseline to show the darkness difference in the other images. In this case, it is difficult to say what constitutes a sufficient level of enhancement. Visual qualitative analysis can take us only this far. In

order to assess how darkness impedes vision, and how good our model is to remove this obstacle, we need to use more quantitative metrics.

4.4 Quantitative analysis

The difference between models' capabilities can be visualized and qualitatively analyzed. However, in order to find the more subtle differences between the model performance in the last two tables, we need to define more quantitative comparing strategies.

4.4.1 Mean Squared Error

A standard method used to measure image difference is the Mean Squared Error (MSE). This metric computes the difference between each pixels of two images, and it averages this quantity over the number of pixels. Mathematically, it is defined as follows:

$$MSE = \frac{1}{mn} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} [I(i,j) - K(i,j)]^2 \tag{4.1}$$

The MSE metric is a measure of the distance between two images purely from the pixel intensity standpoint.

4.4.2 Structural Similarity Index

The structural similarity index (SSIM) [23], instead, is a method to define the perceived quality of digital images, by measuring the similarity of two images. While other metrics rely on absolute errors, SSIM deals

with the perceived structural changes [24]. This allows an increased importance of the structure of the objects in the visual scene. Moreover, this metric considers phenomenons such as luminance masking and contrast masking, which weight the significance of local variation according to their context. MSE, on the other hand, only considers single pixels, ignoring the neighborhood context.

4.4.3 Google API metric

Cloud Vision API

The Cloud Vision API is an service provided by Google Cloud Platform to quickly classify images image into thousands of categories through object detection algorithms. The API can be accessed through the following URL: `https://cloud.google.com/vision/`. Users can simply upload an image and they will be shown matching categories associated to that image which have an association certainty greater than 50%, as shown in Fig. 4.8, and their relative certainty.

Object Detection Metric

The scope of this project is to create an algorithm which will make it easier for vision systems to perform detection. Thus, we thought it would be fitting to test our output using an image detection algorithm such as this.

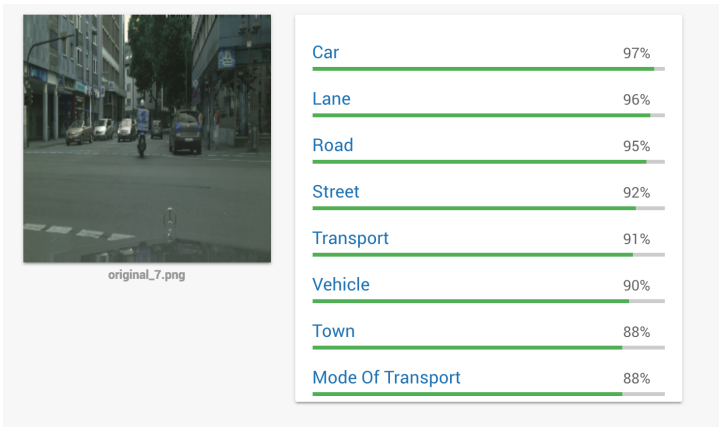


FIGURE 4.8: Result from the API when analyzing one of the clean test images.

Reasoning behind the metric A quality metric for our algorithm should take into account the accuracy of the image reconstruction, but not exclusively. A model which is heavily resistant to noise might still let some noise slip through when tested beyond its limits, but the metric should consider that the test is more challenging. In order to convey both concepts we considered the difference between which tags appear in the tests of the original image vs. the noisy image vs. the decoded image.

The Definition Let us define the tags present in the result for the noisy image as A, the tags of the original image as B and the tags of the decoded image as C.

$$S_D = \frac{B \cap C}{\frac{|B|+|C|}{2}}, \tag{4.2}$$

$$S_N = \frac{C \cap A}{\frac{|C|+|A|}{2}}, \tag{4.3}$$

$$g = \frac{S_D - S_N}{S_D + S_N}. \tag{4.4}$$

The first two equations, eq. 4.2 and eq. 4.3, define the concept of similarity between two images in terms of how many tags they have in common. The intersection between the tags is divided by the average number of tags provided by the API. Since the Google API provides only the tags which have more than 50% certainty, we can use the number of tags provided as a way to include only the most likely detected objects. Two similarities are calculated: S_D compares the decoded and the original image, S_N compares the noisy and the original image. These two quantities alone do not take into account the relation between decoding power and noise level. For this reason, we define another metric to represent the improvement due to our model's contribution. This metric is defined as g, and it consists of the difference between the similarities with a normalization constant (the sum of the similarities). This quantity is zero when the the original image is as similar to the noisy one as it is to the decoded one. The maximum value possible for g is one, if the noisy and the original image are completely dissimilar.

Even though this metric encompasses more aspects than the other

two, there still are some issues. In our similarity metrics we only include tags which are exactly the same. Thus, categories like 'Transport' and 'Mode of transport' would be treated as different, even though the meaning is the same. Moreover, the similarity metric treats equally categories with different levels of relevance. These issues could be addressed by modifying our definition to be even more complex, but we decided against modifying it.

4.4.4 Applying the metrics

The numbers

We tested our noise and darkness comparison tables using the metrics just proposed. We calculated the similarity according to the metrics using the scikit-learn image python package, which includes an implementation if the SSIM. The comparison tables now look as follows:

TABLE 4.1: Darkness comparison using MSE and SSIM

Test Noise	**Noisy**		**Decoded n = 0.5**		**Decoded n = 1.0**	
	MSE	SSIM	MSE	SSIM	MSE	SSIM
n = 1.5	48	0.05	113.36	0.03	104.1	0.08
n = 1.0	45.1	0.05	102	0.031	96.3	0.17
n = 0.5	43	0.07	94.7	0.17	92.9	0.21

This table represents the quantitative measure of the results showed in 4.7 using MSE error and SSIM index. The rows represent the darkening factor used to create the test images, following eq. 3.1. A lower number represents a lower intensity, therefore a smaller d indicates a darker image.

The results for the object detection metric, g, are as follows:

TABLE 4.2: Noise comparison using MSE and SSIM

Test Darkness	**Noisy**		**Decoded d = 0.2**		**Decoded d = 0.5**		**Decoded d = 1.0**	
	MSE	SSIM	MSE	SSIM	MSE	SSIM	MSE	SSIM
d = 0.2	60.3	0.01	41.1	0.24	55.16	0.07	59.7	0.019
d = 0.5	55.3	0.02	43.18	0.31	45.4	0.17	54.8	0.064
d = 1.0	43.15	0.076	94.6	0.165	44.1	0.31	41.9	0.21

This table represents the quantitative measure of the results showed in 4.6 using the MSE error and SSIM index. The rows represent the amount of noise used to create the test images, following eq. 3.1.

TABLE 4.3: Noise comparison using the Object Detection similarities

Test noise	**Noisy**	**Decoded n = 0.5**	**Decoded n = 1.0**
n = 1.5	0	0.05	0.15
n = 1.0	0.05	0.12	0.43
n = 0.5	0.10	0.61	0.68

This table represents the quantitative measure of the results showed in 4.6 using the Object detection similarities. The rows represent the amount of noise used to create the test images, following eq. 3.1.

Understanding the numbers

At first glance, these results might be surprising. First, some patterns do not follow our qualitative interpretation of the result. Second, the metrics do not seem to agree with each other.

MSE The MSE measure does not seem to fit our qualitative interpretation of the results. In the noise comparison table, Table 4.2, the error decreases with noise applied, as expected. The error also decreases for

TABLE 4.4: Darkness comparison using the Object Detection similarities

Test darkness	**Noisy**	**Decoded d = 0.2**	**Decoded d = 0.5**	**Decoded d = 1.0**
d = 0.2	0	0.18	0.09	0.04
d = 0.5	0	0.61	0.13	0.09
d = 1.0	0.11	0.82	0.58	0.45

This table represents the quantitative measure of the results showed in 4.7 using the Object detection similarities. The rows represent the darkening factor used to create the test images, following eq. 3.1. A lower number represents a lower intensity, therefore a smaller d indicates a darker image.

the models which were trained with higher noise, following our qualitative observations. However, in the darkness comparison table, Table 4.1, the error seems to increase for decreasing darkness for one model and decrease slightly for the other two models. For some models, there seems to be a big difference between the images trained with $d = 1$ (bright images) and $d = 0.5$ (mid-dark images), while for some others that difference is minimal. Thus, MSE does not seem to be a reliable metric for the darkness comparison.

MSE is a pixel to pixel comparison, which means the metric is not robust to what is perceived as an invariance. For example, two identical images where the pixels are translated by one position would have a very big MSE error, even though they would look exactly the same to a human.

SSIM SSIM should address this problem, and be more robust to invariance. The structural similarity between decoded and original images, though, also seem to not follow our qualitative analysis as expected. In

the noise comparison table, Table 4.2, the value of the index does follow our qualitative observation. On the other hand, in the darkness comparison table, the error does not show specific trends at all.

As in the MSE case, it seems that the darkness comparison task does not fall within the scope of this metric.

Object Detection Similarities The Object Detection Similarities is the only metric that seems to agree with our observations. This is somehow expected, since its framework has been trained on the way that the human brain classifies and recognizes changes in images, by comparing the objects it detects in those images. In sec. 4.4.4, we decided to only include the similarities and not the improvement metric since the zero values in the noise similarity, S_N would give skewed results. In both tables, the similarity decreases with increasing levels of difficulty - as expected.

Chapter 5

Model Analysis

5.1 Real Images

Synthetic images are a good tool to control the parameters which we are interested in testing in order to optimize the model's training. However, even though they address both the problem of darkness and the problem of noise, they do not faithfully represent reality.

The ultimate test for our model will be to show its generalization ability to real images. This would not only tell us something about our model, but it will also delineate the ways in which synthetic images differ from real world images.

5.1.1 Our Dataset

In order to tackle different levels of noise, we created a small testing datasets consisting of pictures of objects presents in a room under a source of increasingly bright light. The light intensity was indicated with an index from 1 to 20. Potentially, this dataset can act as a way to measure the

breaking point of the model, or at least as an indication of how the model behaves increasingly low-light real conditions.

Model performance

Figure 5.1 shows the performance of the model on the higher definition images included in the real-image dataset. The testing was performed using the patching technique described earlier. One can notice artifacts on the reconstructed image, probably due to the stitching of the tiles.

Table 5.1 shows the results of applying our model to some more real images. We present the original and enhanced images next to each other. Most images, apart from the middle and dark image of the magazine, show significant improvement. The brightness is significantly higher. Some images appear to be too dark for our model to tackle, maybe because we did not train it with such a low intensity. Some details of the books are treated as noise, and thus smoothened out, highlighting the denoising capabilities of the model. Noise due to darkness is almost completely smoothened out, as can be noticed from the first scene, dark image.

5.2 Conclusion

This project was aimed at developing an algorithm which would be able to denoise and reduce the darkness of low light images. We started by exploring some of the current approaches that are used to solve the problem of low light image enhancement. After analyzing the output of those

FIGURE 5.1: Higher definition images processed using patching applied in the testing phase

methods, we decided that in order to keep the denoising ability within our scope, we should follow the structure of the LLNet approach. This approach, however, had significant limitations, such as its inability to denoise images in cases of noisy input. We modified its architecture keeping the autoencoder structure. Our proposed method includes convolutional layers and skip connections. Our final model, and every modification along the way, outperforms the LLNet model when tested on synthetic images.

We tested the model in its robustness to noise and darkness levels. We initially performed qualitative analysis. We then considered some metrics of quality used in similar cases, but we noticed that they proved our qualitative observations only on the denoising front. We then decided to test our work following more closely the scope of this project, which includes the possibility of using this algorithm in combination with other vision systems, such as object detecting algorithms. We did this by developing our own metric using the Google Vision API, following the results of an object detection algorithm.

Finally, we tested the model on real life images. Even though the performance was not as accurate as in the synthetic images, our images were significantly denoised. In case of very dark input our model was not able to brighten it significantly - suggesting that more work needs to be done in this direction. The real images we tested on were not very noisy. Thus, they could not highlight the denoising capabilities of our model. The strengths of our model reside on its strong denoising ability which also

has a brightening counterpart. In this regime, it outperforms the state-of-the-art architectures. For this reason, we propose the use of the model in case of highly noisy dark input, such as low-definition footage from CCTV cameras.

5.2.1 Future direction

Bright/Dark Image Dataset

In order to achieve the best accuracy when testing on real images, it is necessary to also train on real images. As of now, no dataset exists with such capabilities. Thus, the result depends strongly on how the synthetic images resemble real ones. If we could take this task out of the equation, we would be able to focus on optimizing the right algorithm.

Creating such a dataset would not be easy. The images in the dataset should be couples of bright and dark images which represent the exact same scene. The task of compiling such images could be made easier by limiting the images to ones taken indoors, where lighting can be easily manipulated. However, training on such a dataset would not be of help to vision systems like self-driving cars which are interested in outdoor low-light situations.

GANs

A way to avoid having to train on a real dataset could be using Generative Adversarial Networks (GANs) [25]. GANs constituted a breakthrough in the field of deep learning. GANs use two interacting neural

networks. In our case, one network has the function of generator of images, and one has the function of discriminating images that do not look real. The architecture is optimized until the generator is able to produce images that look so real that the discriminator cannot tell that they are not. The natural continuation of this project would be to use our U-Net autoencoder as a generator, and the development of a discriminator. This approach has been used on similar problems before, when dealing with other types of noise. A recent paper [26], for example, uses an architecture similar to the one just proposed to remove raindrops from images, obtaining very good results.

TABLE 5.1: Model applied on real images from our dataset. Horizontally, we have different levels of darkness; vertically, different scenes.

Bibliography

[1] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation", 2015. arXiv: 1505.04597. [Online]. Available: http://arxiv.org/abs/1505.04597 (cit. on pp. iii, iv, 2, 28, 29).

[2] X. Guo, Y. Li, and H. Ling, "LIME: Low-light image enhancement via illumination map estimation", *IEEE Transactions on Image Processing*, 2017, ISSN: 10577149. DOI: 10.1109/TIP.2016.2639450 (cit. on pp. iii, 1, 7, 14).

[3] K. G. Lore, A. Akintayo, and S. Sarkar, "LLNet: A deep autoencoder approach to natural low-light image enhancement", *Pattern Recognition*, 2017, ISSN: 00313203. DOI: 10.1016/j.patcog.2016.06.008. arXiv: 1511.03995 (cit. on pp. iii, iv, 1, 2, 7–9, 11, 15, 16).

[4] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The Cityscapes Dataset for Semantic Urban Scene Understanding", in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016 (cit. on pp. 2, 10, 23).

[5] C. Couprie, C. C. Farabet, L. Najman, and Y. LeCun, "Indoor Semantic Segmentation using Depth Information", in *International Conference on Learning Representations (ICLR2013)*, 2013, pp. 1–8, ISBN: 978-1-4799-5118-5. DOI: 10.1109/CVPR.2014.81. arXiv: 1301.3572. [Online]. Available: https://arxiv.org/pdf/1301.3572.pdfhttp://arxiv.org/abs/1301.3572https://nyuscholars.nyu.edu/en/publications/indoor-semantic-segmentation-using-depth-information (cit. on p. 2).

[6] A. Krizhevsky, I. Sutskever, and G Hinton, "{ImageNet} Classification with Deep Convolutional Neural Networks", in *Advances in Neural Information Processing Systems 25 (NIPS'2012)*, Curran Associates Inc., 2012, pp. 1097–1105. [Online]. Available: https://dl.acm.org/citation.cfm?id=2999257 (cit. on p. 2).

[7] P. Vincent and H. Larochelle, "Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion Pierre-Antoine Manzagol", *Journal of Machine Learning Research*, vol. 11, pp. 3371–3408, 2010, ISSN: 15324435. DOI: 10.1111/1467-8535.00290. arXiv: 0-387-31073-8 (cit. on pp. 3, 4, 10).

[8] Y. Bengio, Y. {LeCun}, and Y. Lecun, "Scaling Learning Algorithms towards AI", *Large Scale Kernel Machines*, no. 1, pp. 321–360, 2007, ISSN: 00099104. DOI: 10.1.1.72.4580. arXiv: arXiv:1011.1669v3. [Online]. Available: https://nyuscholars.nyu.edu/en/publications/scaling-learning-algorithms-towards-ai (cit. on p. 4).

[9] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy Layer-Wise Training of Deep Networks", *Advances in Neural Information Processing Systems*, vol. 19, no. 1, p. 153, 2007, ISSN: 01628828. DOI: citeulike-article-id:4640046. arXiv: 0500581 [submit]. [Online]. Available: https://papers.nips.cc/paper/3048-greedy-layer-wise-training-of-deep-networks.pdfhttp://papers.nips.cc/paper/3048-greedy-layer-wise-training-of-deep-networks.pdf (cit. on p. 4).

[10] G. E. Hinton, S. Osindero, and Y.-W. W. Teh, "A Fast Learning Algorithm for Deep Belief Nets", *Neural Computation*, vol. 18, no. 7, pp. 1527–1554, 2006, ISSN: 0899-7667. DOI: 10.1162/neco.2006.18.7.1527. arXiv: 1111.6189v1. [Online]. Available: http://www.cs.toronto.edu/{~}fritz/absps/ncfast.pdfhttp://www.ncbi.nlm.nih.gov/pubmed/16764513http://www.mitpressjournals.org/doi/10.1162/neco.2006.18.7.1527 (cit. on p. 4).

[11] G. Hinton, N. Srivastava, and K. Swersky, *Neural Networks for Machine Learning Lecture 6a Overview of mini--batch gradient descent*. [Online]. Available: https://www.cs.toronto.edu/{~}tijmen/csc321/slides/lecture{_}slides{_}lec6.pdf (visited on 10/23/2017) (cit. on pp. 4, 15).

[12] V. Vryniotis, *Tuning the learning rate in Gradient Descent | Datumbox*, 2013. [Online]. Available: http://blog.datumbox.com/tuning-the-learning-rate-in-gradient-descent/ (visited on 11/25/2017) (cit. on p. 5).

[13] M. D. Zeiler, "ADADELTA: An Adaptive Learning Rate Method", *CoRR*, vol. abs/1212.5, 2012 (cit. on p. 5).

[14] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization.", *CoRR*, vol. abs/1412.6, 2014. [Online]. Available: http://dblp.uni-trier.de/db/journals/corr/corr1412.html{\#}KingmaB14 (cit. on p. 5).

[15] H. D. Cheng and X. J. Shi, "A simple and effective histogram equalization approach to image enhancement", *Digital Signal Processing*, vol. 14, pp. 158–170, 2004. DOI: 10.1016/j.dsp.2003.07.002. [Online]. Available: www.elsevier.com/locate/dsp (cit. on p. 7).

[16] P. Trahanias and A. Venetsanopoulos, "Color image enhancement through 3-D histogram equalization", *Proceedings., 11th IAPR International Conference on Pattern Recognition. Vol. IV. Conference D: Architectures for Vision and Pattern Recognition,*, pp. 545–548, 1992. DOI: 10.1109/ICPR.1992.202045. [Online]. Available: http://ieeexplore.ieee.org/document/202045/http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=202045 (cit. on p. 7).

[17] S. M. Pizer, E. P. Amburn, J. D. Austin, R. Cromartie, A. Geselowitz, T. Greer, B. ter Haar Romeny, J. B. Zimmerman, and K. Zuiderveld, "Adaptive Histogram Equalization and its Variations.", *Computer vision, graphics, and image processing*, vol. 39, no. 3, pp. 355–368, 1987, ISSN: 0734189X. DOI: 10.1016/S0734-189X(87)80186-X. [Online].

Available: http://www.sciencedirect.com/science/article/pii/S0734189X8780186X?via{\%}3Dihub (cit. on p. 7).

[18] E. D. Pisano, S. Zong, B. M. Hemminger, M. DeLuca, R. E. Johnston, K. Muller, M. P. Braeuning, and S. M. Pizer, "Contrast Limited Adaptive Histogram Equalization image processing to improve the detection of simulated spiculations in dense mammograms", *Journal of Digital Imaging*, vol. 11, no. 4, pp. 193–200, 1998, ISSN: 0897-1889. DOI: 10.1007/BF03178082. [Online]. Available: http://link.springer.com/10.1007/BF03178082 (cit. on p. 7).

[19] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Image Denoising by Sparse 3-D Transform-Domain Collaborative Filtering", *IEEE Transactions on Image Processing*, vol. 16, no. 8, pp. 2080–2095, 2007, ISSN: 1057-7149. DOI: 10.1109/TIP.2007.901238. [Online]. Available: http://ieeexplore.ieee.org/document/4271520/ (cit. on p. 7).

[20] F. Chollet, *Building Autoencoders in Keras*, 2016. [Online]. Available: https://blog.keras.io/building-autoencoders-in-keras.html (visited on 10/10/2017) (cit. on p. 16).

[21] M. Drozdzal, E. Vorontsov, G. Chartrand, S. Kadoury, and C. Pal, "The Importance of Skip Connections in Biomedical Image Segmentation", ISSN: 16113349. DOI: 10.1007/978-3-319-46976-8_19. arXiv: 1608.04117. [Online]. Available: https://arxiv.org/pdf/1608.04117.pdf (cit. on p. 27).

[22] T. Yamashita, "Multiple Skip Connections and Dilated Convolutions for Semantic Segmentation", *Workshop on IEEE Intelligent Vehicle Symposium*, 2017. [Online]. Available: http://www.vision.cs.chubu.ac.jp/MPRG/C{_}group/C086{_}yamashita2017.pdf (cit. on p. 28).

[23] Z. 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*, vol. 13, no. 4, 2004. DOI: 10.1109/TIP.2003.819861. [Online]. Available: http://www.cns.nyu.edu/pub/lcv/wang03-reprint.pdf (cit. on p. 36).

[24] A. Horé and D. Ziou, "Image quality metrics: PSNR vs. SSIM", in *Proceedings - International Conference on Pattern Recognition*, IEEE, 2010, pp. 2366–2369, ISBN: 9780769541099. DOI: 10.1109/ICPR.2010.579. [Online]. Available: http://ieeexplore.ieee.org/document/5596999/ (cit. on p. 37).

[25] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative Adversarial Nets", *Advances in Neural Information Processing Systems 27*, pp. 2672–2680, 2014, ISSN: 10495258. DOI: 10.1017/CBO9781139058452. arXiv: arXiv:1406.2661v1. [Online]. Available: https://papers.nips.cc/paper/5423-generative-adversarial-nets.pdfhttp://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf (cit. on p. 48).

[26] R. Qian, R. T. Tan, W. Yang, J. Su, and J. Liu, "Attentive Generative Adversarial Network for Raindrop Removal from a Single Image", 2017. arXiv: 1711.10098. [Online]. Available: http://arxiv.org/abs/1711.10098 (cit. on p. 49).