Image De-Raining Using Deconvolutional Neural Network

Tejas Bhatia
Student, Department of Computer Science
BITS PILANI K K Birla Goa Campus
Zuarinagar, Goa, India
f20150111@goa.bits-pilani.ac.in

Shivin Thukral
Student, Department of Computer Science
BITS PILANI K K Birla Goa Campus
Zuarinagar, Goa, India
f20150350@goa.bits-pilani.ac.in

Saurabh Majumdar
Student, Department of Computer Science
BITS PILANI K K Birla Goa Campus
Zuarinagar, Goa, India
f20150078@goa.bits-pilani.ac.in

***Abstract*—Noise produced in images due to rain are a very common problem. Not only they make the background obscure, but also decrease the visual clarity of the image. This poses a problem when such rainy images have to be used for image classification or recognition tasks. In this paper, we present a simple neural network that generates a de-rained image from an input rainy image. The neural network is a simple deconvolutional neural network, which uses mean squared loss to train over a set of images, learning to remove the noise rain from the input image.**

Keywords—single image deraining, mean squared loss, deconvolutional neural net

I. Introduction

Neural networks have gained a lot of popularity recently in solving a number of image processing problems. Convolutional neural networks (CNNs) have been primarily implemented for solving deep learning tasks involving images such as image classification, generation and recognition. Very common instances include identifying faces, objects, traffic signs, along with self-driving cars. A CNN comprises of a number of convolutional and sampling layers followed by an activation and/or dense (fully connected) layers. It is able to recognize scenes and features from an image, and suggest relevant captions for the image.

II. Proposed Method

In this section, we present a detail of the neural network architecture along with a few insight into few of the terminology.

A. Network Architecture

Our neural network employs a deconvolutional neural network of a symmetric structure. At first, there is a set of three convolutional layers, each of which is activated by a ReLu activation function, and weights initialized with the Xavier initialization. The number of filters doubles in consecutive convolutional layers.

Following the convolutional layers is the second set of three deconvolutional (transpose convolutional) layers. Each of these is also activated by a ReLu activation function, and weights similarly initialized with the Xavier initialization method. The number of filters halves in consecutive deconvolutional layers, except in the last layer which has only 3 filters.

The final architecture is as follows:

CXR(K)-CXR(2K)-CXR(4K)-DXR(2K)-DXR(K)-DXR(3)

where CXR(N) is a convolutional layer with Xavier initialization and ReLu activation of N filters, and DXR(N) is a deconvolutional layer with Xavier initialization and ReLu activation.

Besides this, mean squared loss function was used as the loss function of the network. The optimizer used to minimize this loss function was Adam Optimizer, implemented along with a exponentially decaying learning rate. Mini-batch descent was used, and weights were updated for every mini-batch.

B. Terminology

1. Adam Optimizer : Adam is an optimization algorithm that can used instead of the classical stochastic gradient descent procedure to update network weights iterative based in training data. It was presented by Diederik Kingma from OpenAI and Jimm Ba from the University of Toronto in their 2015 ICLR paper (poster) titled "Adam: A Method for Stochastic Optimization".
   Adam Op is the combination of two other extensions of stochastic gradient descent, namely:
   - Adaptive Gradient Algorithm (AdaGrad)
   - Root Mean Square Propagation (RMSProp)

2. Xavier Initialization : In this method, we need to pick the weights from a Gaussian distribution with zero mean and a variance of 1/N, where N specifies the number of input neurons. In the original paper, the authors take the average of the number input neurons and the output neurons.

3. Learning Rate Decay : In order to ensure that the learning converges when the loss function is near its minima, the learning rate is decreased over time/steps so that a lesser learning rate allows a finer descent over to the minimum of the loss function.

III. Experiment And Results

In this section, we present details of the how the training was performed, as well as the values of hyperparameters that were tuned by the end of the training. Presented along with this are a few illustrations of the results on a test sample.

A. Training and Test Data

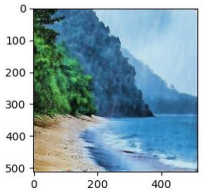
The training was performed on 700 images, in which rain had been added synthetically to obtain the corresponding rainy image from the ground truth image. The test set comprises of 100 images. Results of the network on a test sample of 21 images is mentioned in Table I. A real rainy image was also tested, the result of which can be seen visually in Fig I.

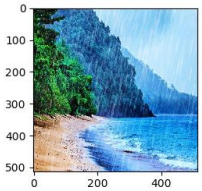
B. Hyperparameters

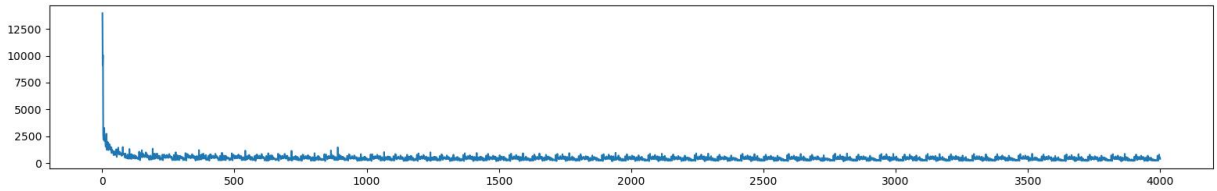
- Learning Rate : The Adam optimization was started with an initial learning rate of 0.001, implemented with a exponential decay of 0.1 after every 500 steps.
- Mini-Batch Size : A mini-batch size of 8 images was used.
- Training steps : A total number of 8,000 iterations (each iteration of 8 images each) was run over the entire 700 images training set.
- Value of K : In the convolutional and deconvolutional layers, value K = 16 was used.
- Stride and padding : A stride of 1 and no padding was used for all the layers
- Filter size : The filter size used for the six layers was 3x3, 5x5, 4x4, 4x4, 5x5, 3x3 respectively.

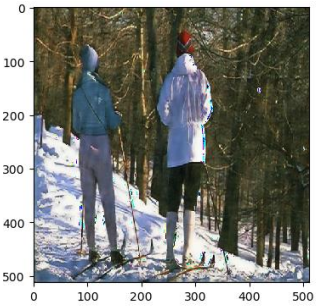
Sample test image	PSNR (in dB)	Accuracy (MSE)
1	20.6213	563.56
2	19.5002	729.56
3	25.2283	195.09
4	17.6647	1113.27
5	20.4798	582.22
6	18.2225	979.09
7	21.7190	437.70
8	22.6094	356.56
9	24.6081	254.84
10	22.1286	398.33
11	18.2233	978.92
12	17.3309	1209.75
13	23.5586	286.56
14	19.1395	792.73
15	27.0886	127.12
16	26.1534	157.66
17	21.2687	485.52
18	18.4883	190.97
19	21.4897	461.42
20	21.4241	468.45
21	18.8646	844.52
Average	21.2021	587.80

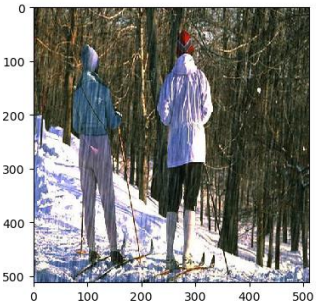
Table I. Test sample results

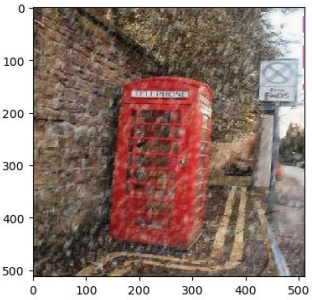


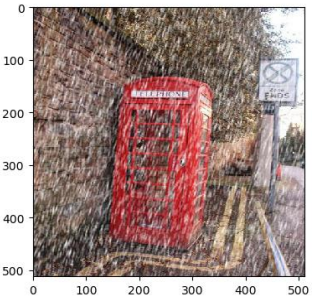


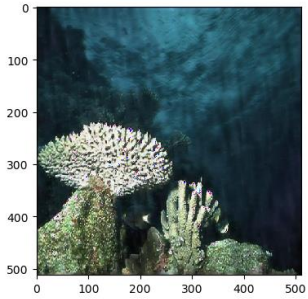


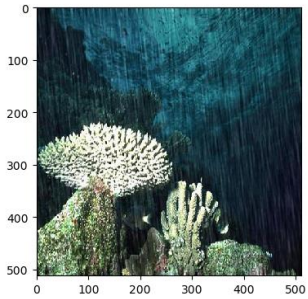












Conclusion

In this paper, we proposed a neural network architecture for the purpose of deraining images. A simple deconvolutional neural network was used. While there are a number of techniques in deep learning out there such as General Adversarial Networks (GANs) and Recurrent Neural Networks (RNNs), our architecture also achieved noteworthy results, producing an average PSNR of >20 dB.

Acknowledgment

This work was supported by Professor Tirtharaj Dash, of Department of Computer Sciences, BITS Pilani K K Birla Campus Institute.

References

[1] Li Xu, Jimmy SJ Ren, Ce Liu, and Jiaya Jia, "Deep Convolutional Neural Network for Image Deconvolution"

[2] Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han, "Learning Deconvolution Network for Semantic Segmentation".