Image Deraining

Using Deconvolution Neural Networks

BITS PILANI, K.K. BIRLA GOA CAMPUS

Ashutosh Upreti : 2014B4A70784G

Himanshu Singhvi : 2014B2A70716G

Abhijeet Khasnis : 2014B3A30529G

***Abstract*—The effects of rain can degrade the visual quality of images and severely affect the performance of outdoor vision systems. Under rainy conditions, rain streaks create not only a blurring effect in images, but also haziness due to light scattering. Effective methods for removing rain streaks are required for a wide range of practical applications, such as image enhancement and object tracking. Here we build a Deconvolution Neural Network to learn the mapping relationship between rainy and derained image from the given data set.**

I. Introduction

It has been widely acknowledged that unpredictable impairments such as illumination, noise and severe weather conditions (i.e. rain, snow and fog) adversely affect the performance of many computer vision algorithms such as detection, classification and tracking. This is primarily due to the fact that these algorithms are trained using images that are captured under well-controlled conditions. For instance, the presence of heavy rain greatly impairs visual quality of the image, thus rendering face detection and verification algorithms ineffective for such degradations. A possible method to address this issue is to include images captured under unconstrained conditions in the training process of these algorithms. However, it may not be practical to collect such images for all classes in the training set, especially in a large scale setting. In addition, in this age of ubiquitous smartphone usage, images captured by smartphone cameras under difficult weather conditions undergo degradations that drastically affect the visual quality of images making the images useless for sharing and usage. In order to improve the overall quality of such degraded images for better visual appeal and to ensure enhanced performance of vision algorithms, it becomes essential to automatically remove undesirable artifacts arising due to difficult weather conditions.

Convolutional Neural Networks are very similar to ordinary Neural Networks, they are made up of neurons that have learnable weights and biases. Each neuron receives some inputs, performs a dot product and optionally follows it with a non-linearity. The whole network still expresses a single differentiable score function: from the raw image pixels on one end to class scores at the other. And they still have a loss function on the last (fully-connected) layer. ConvNet architectures make the explicit assumption that the inputs are images, which allows us to encode certain properties into the architecture. These then make the forward function more efficient to implement and vastly reduce the amount of parameters in the network.

The need for transposed convolutions(deconvolution) generally arises from the desire to use a transformation going in the opposite direction of a normal convolution, to project feature maps to a higher-dimensional space, i.e., map from a 4-dimensional space to a 16-dimensional space, while keeping the connectivity pattern of the convolution. Transposed convolutions – also called fractionally strided convolutions – work by swapping the forward and backward passes of a convolution. One way to put it is to note that the kernel defines a convolution, but whether it's a direct convolution or a transposed convolution is determined by how the forward and backward passes are computed. The transposed convolution operation can be thought of as the gradient of some convolution with respect to its input, which is usually how transposed convolutions are implemented in practice.

II. Related Work

Due to the redundant temporal information that exists in video, rain streaks can be more easily identified and removed in this domain [1]–[4]. For example, in [1] the authors first propose a rain streak detection algorithm based on a correlation model. After detecting the location of rain streaks, the method uses the average pixel value taken from the neighboring frames to remove streaks. In [2], the authors analyze the properties of rain and establish a model of visual effect of rain in frequency space. In [3], the histogram of streak orientation is used to detect rain and a Gaussian mixture model is used to extract the rain layer. In [4], based on the minimization of registration error between frames, phase congruency is used to detect and remove the rain streaks. Many of these methods work well, but are significantly aided by the temporal content of video. In this paper we try to remove rain from a single image using de-convolution neural network.

III. Architecture

This section discusses the architecture of our deconvolution network.

Below figure illustrates the detailed configuration of the entire deep network. Our trained network is composed of two parts— convolution and deconvolution networks. The convolution network corresponds to feature extractor that transforms the input image to multidimensional feature representation, whereas the deconvolution network is a shape generator that produces new image from the feature extracted from the convolution network. The final output of the network is a de-rained version of the input rainy image.

A. *Deconvolution Network for Generation: There are two major operations in any deconvolution network. These are:*

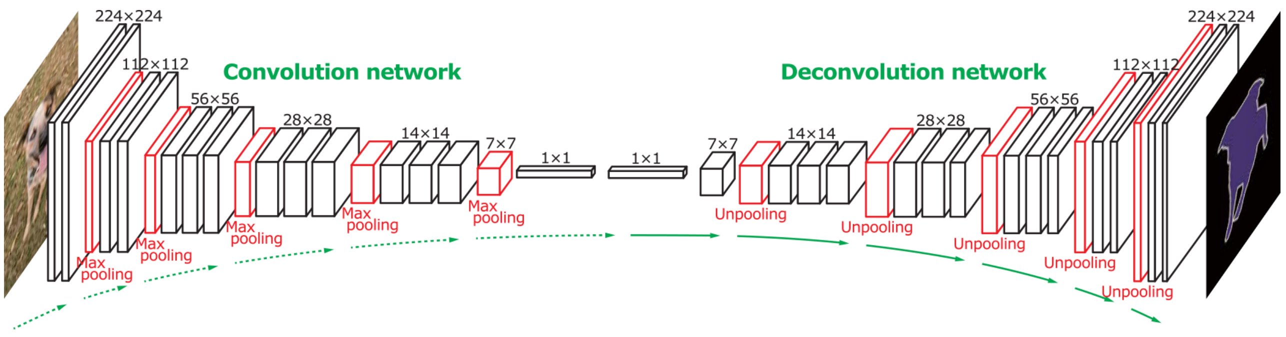
- **Unpooling**
  Pooling in convolution network is designed to filter noisy activations in a lower layer by abstracting activations in a receptive field with a single representative value. Although it helps classification by retaining only robust activations in upper layers, spatial information within a receptive field is lost during pooling, which may be critical for precise localization that is required for semantic segmentation. To resolve such issue, we employ unpooling layers in deconvolution network, which perform the reverse operation of pooling and reconstruct the original size of activations. To implement the unpooling operation, we follow the similar approach proposed in. It records the locations of maximum activations selected during pooling operation in switch variables, which are employed to place each activation back to its original pooled location.

- **Deconvolution**
  The output of an unpooling layer is an enlarged, yet sparse activation map. The deconvolution layers densify the sparse activations obtained by unpooling through convolution-like operations with multiple learned filters. However, contrary to convolutional layers, which connect multiple input activations within a filter window to a single activation, deconvolutional layers associate a single input activation with multiple outputs, as illustrated in figure given below. The output of the deconvolutional layer is an enlarged and dense activation map. We crop the boundary of the enlarged activation map to keep the size of the output map identical to the one from the preceding unpooling layer. The learned filters in deconvolutional layers correspond to bases to reconstruct shape of an input object. Therefore, similar to the convolution network, a hierarchical structure of deconvolutional layers are used to capture different level of shape details. The filters in lower layers tend to capture overall shape of an object while the class-specific fine details are encoded in the filters in higher layers. In this way, the network directly takes class-specific shape information into account for semantic segmentation, which is often ignored in other approaches based only on convolutional layers.

B. *Proposed Architecture*

Below figure shows the nature of proposed architecture. The architecture that we used for training the model is as follows:

- Each training image is 256 x 256 pixels image (resized).
- We take the input as 256 x 256 x 3 where 3 denotes the channels of RGB image.
- This input is fed to a convolutional layer of 2 x 2 filter followed by ReLU layer.
- The output of the previous layer is fed into the Batch Normalization layer.
- The output of the previous layer is again fed into a convolutional layer of 2 x 2 filter followed by ReLU layer.
- The next layer is max pooling layer with filter size of 2 x 2.
- The output of the previous layer is again fed into a convolutional layer of 3 x 3 filter followed by ReLU layer.
- The next layer is max pooling layer with filter size of 2 x 2.
- The output of the previous layer is again fed into a convolutional layer of 2 x 2 filter followed by ReLU layer.
- The output of the above network is fed into a dropout layer with keep probability of each neuron of 0.5.
- The next layer is max pooling layer with filter size of 2 x 2.
- The output of the previous layer is fed into the Batch Normalization layer.
- The next layer is an upsampling of 2 x 2 layer.
- The output of the layer is fed into a deconvolution layer.
- The next layer is an upsampling of 2 x 2 layer.
- The output of the layer is fed into a deconvolution layer.

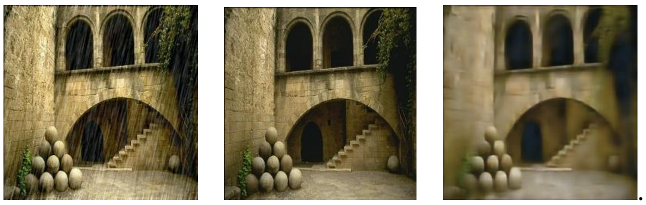


- The next layer is an upsampling of 2 x 2 layer.
- The output of the layer is fed into a deconvolution layer.
- The output of the layer is fed into a final deconvolution layer.

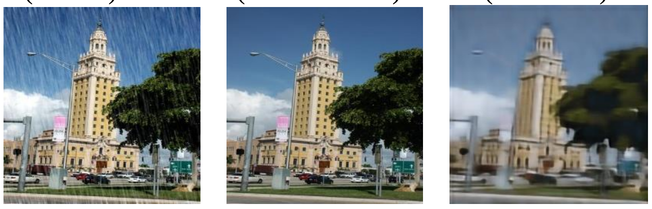
IV. Experiment

The dataset comprises of 665 rain and de-rained images having variable dimensions which were resized to 256 x 256. The code is run on Keras 1.1 on Nvidia GPU (GTX 860M). We have used mean square error (MSE) as our loss function and Adam Optimizer to minimize the loss. For calculating the accuracy on the validation set we used peek signal to noise ratio (psnr).

We are getting accuracy around 78% and psnr of around 19.6. Below are some of the results produced through our network.

.

(Rained) (Ground Truth) (DeRained)



References

[1] K. Garg and S. K. Nayar, "Detection and removal of rain from videos," in International Conference on Computer Vision and Pattern Recognition (CVPR), 2004.

[2] P. C. Barnum, S. Narasimhan, and T. Kanade, "Analysis of rain and snow in frequency space," International Journal on Computer Vision, vol. 86, no. 2-3, pp. 256–274, 2010.

[3] J. Bossu, N. Hautiere, and J.P. Tarel, "Rain or snow detection in image sequences through use of a histogram of orientation of streaks," International Journal on Computer Vision, vol. 93, no. 3, pp. 348–367, 2011..

[4] V. Santhaseelan and V. K. Asari, "Utilizing local phase information to remove rain from video," International Journal on Computer Vision, vol. 112, no. 1, pp. 71–89, 2015..

[5] He Zhang and Vishwanath Sindagi and Vishal M. Patel "Image De-raining Using a Conditional Generative Adversarial Network"

[6] Hyeonwoo Noh , Seunghoon Hong , Bohyung Han, Learning Deconvolution Network for Semantic Segmentation, Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV), p.1520-1528, December 07-13, 2015

[7] M. D. Zeiler and R. Fergus. Visualizing and understandingconvolutional networks. In ECCV, 2014.