DeSnowNet

Srinath Naik Ajmeera
University of California Los Angeles
Los Angeles, California
srinath@cs.ucla.edu

Abstract

This work aims to implement a paper on de-snowing and suggest potential improvements and future directions. The proposed method uses modular neural network and handles various scales of snow particles by using large receptive fields based on dilated convolutions. It also presents results obtained by training on a reduced dataset and light-weight network.

1. Introduction

This work is an effort to implement the paper [1] and study possible improvements and future directions. Snowy images constitute a major problem for vision based applications like self-driving and is challenging due to complex nature and scales of snow particles. This method investigates an approach to recover snow-free image from a snowy image. Section 2 gives an overview of proposed method and network the Section 3 mentions about the dataset used, Section 4 describes experiments and training details followed by Section 5 showing some results and finally, Section 6 discusses conclusion and possible future directions.

2. Proposed Method

A snowy image x is modeled as a combination of snow free image y and an independent snow mask z representing the translucency of the snow as described by the following equation.

$$x = a \odot z + y \odot (1 - z) \tag{1}$$

where $\odot$ represents element-wise multiplication and a represents the color aberration at each coordinate. In order to estimate the snow-free image y, it is important to accurately estimate the snow mask z and color aberration a. We use a neural network with multi scale receptive fields. The network consists of two modules, namely Translucency Recovery(TR) module and Residual Generation(RG) module. The TR module recovers areas occluded by translucent snow where as RG module generates the residual component r where portions of image is completely obscured by opaque snow particles. The following equation describes relation between final snow-free image with the output TR(y') and RG(r) modules respectively.

$$\hat{y} = y' + r \tag{2}$$

2.1. Network Structure

The Translucency Recovery module consists of Descriptor and Recovery sub-modules respectively. Descriptor is built using Inception-V4 network followed by Dilation Pyramid to account for multiple scale of snow particles. The features from the Dilation Pyramid are fed to the Recovery sub-module which has a Snow Extractor(SE) and Aberration Extractor(AE) to estimate $\hat{z}$ and a respectively. SE and AE share similar structure and use Pyramid Maxout on convolutions of various kernel sizes in a pyramid fashion. $a, \hat{z}$ along with input x are used to estimate y' from the TR module. Concatenating $y', x, \hat{z}$ forms f_c which is used as an input to RG module.

The Residual Generation module also consists of Descriptor and Recovery sub-modules, The Descriptor shares same network structure as the one in the TR module where as the Recovery sub-module follows similar to SE/AE modules but uses Pyramid Sum operation instead of maxout to generate r.

2.2. Loss Function

The exact MSE loss function as described in the paper [1] is used.

3. Dataset

A derived dataset from Snow100K is used in these experiments for training. Using the full dataset containing 50,000 images is consuming a lot of time to train each epoch(around 3 hours), therefore a reduced dataset containing 20,000 images of snow, corresponding masks and ground truth is derived by filtering first 20,000 images out of the full dataset in lexical order.

4. Experimental Details

The initial implementation of the network is in accordance with the description in the paper [1] . However, due to computational limitations on GPU RAM size in google-colab, It was only able to make a forward pass for a batch size of 2 or less. This network has around 48 million parameters, and uses a lower weight Inception-V4 network in the Descriptor modules by cutting most of the inner layer channels to half of the original network, Inspired from that idea, I have further reduced the channels at all modules to half in order to speed up the training process. This low weight network has around 12 million parameters and I could use a batch size of up to 16 for training.

Due to limitation of time and computational resources, I was only able to train this network for around 30 epochs using batch size as 16 and the same learning parameters as of the original paper. Clearly, the training error got down with each epoch and I believe that spending enough time and using better computational resource can achieve much closer results. Training is performed in multiple phases(2 in this case), as google-colab disconnects runtime after several hours. Training loss for every 100 batches is also plotted as shown in Fig 1

5. Results

Inference performed on the test set and realistic images are presented below. Random images are selected from the test data and 64 by 64 patches are fed to the model for inference. The output is stitched back to create the full snow-free image. Please note that the lower part of snow-free image is dark as it is not considered because padding is not accounted to break input image into exact patches. Given that the network has parameters $\frac{1}{4}$ of the original network and trained for just around 30 epochs, I feel that the presented results are satisfactory and much better/closer results can be obtained with proper computational resources.

5.1. Inference on Test Data

Fig 2, Fig 3 and Fig 4 show results obtained from provided test set on Small, Medium and Large snow particle combinations respectively.

5.2. Inference on Real Snow Data

Fig 5 show results obtained from realistic snowy images provided in the dataset.

6. Conclusion and Future Directions

To conclude, this light weight network(when compared with original one) when trained on reduced dataset has given satisfactory results even for small number of training epochs. In current experiments, dropout is not considered. Further experiments using dropout might be helpful to improve accuracy. The accuracy on realistic images is the goal of this project and as for the idea implemented in the paper, how well it performs in realistic case depends on accurate modelling of synthetic data. Efforts can be put in directions of accurately modeling the synthetic data, foe example accounting for possible shapes of snow(as they are group of weighted particles moving in effect of wind and gravity, may be we can form some prior on the snow masks).

References

[1] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848, 2017. 1, 2

Figure 1. Train loss every 100 batches

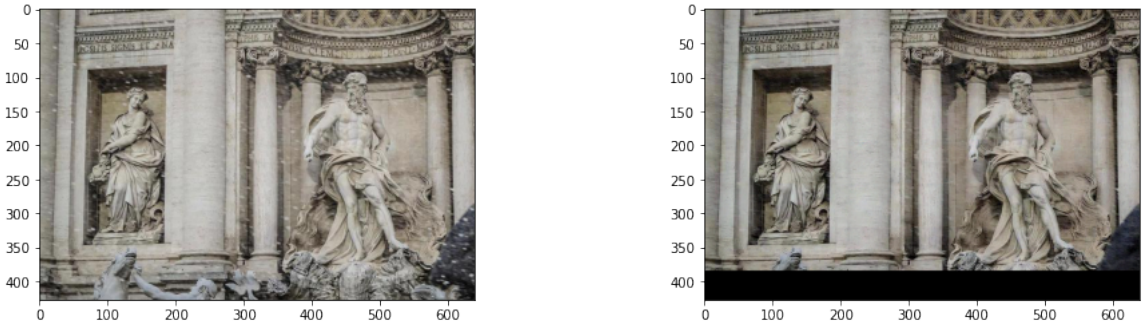

Figure 2. Results on data from Test set, Snow100K-S

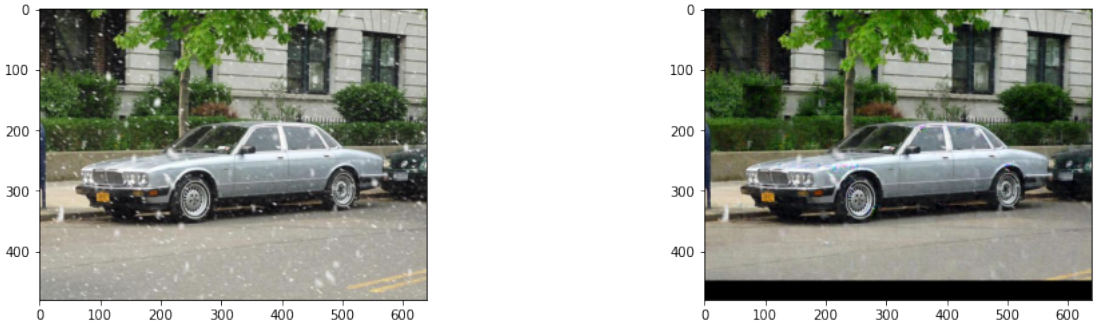

Figure 3. Results on data from Test set, Snow100K-M

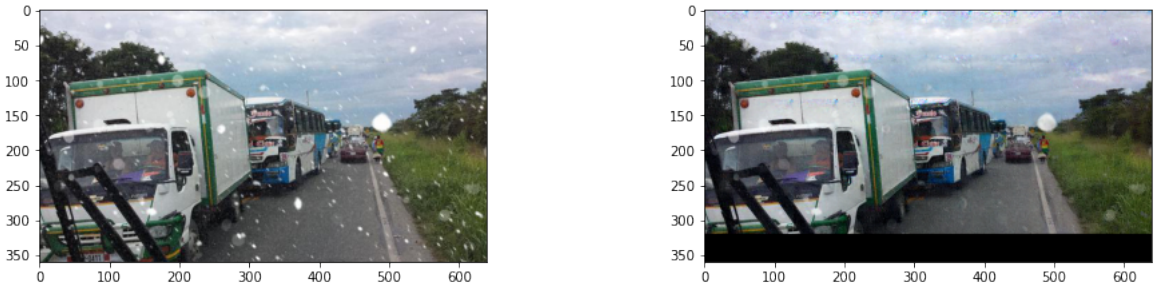

Figure 4. Results on data from Test set, Snow100K-L

Figure 5. Results on data from realistic images