Image Deraining

Project members:

- Revanth Reddy Bandaru
- Krishna Rohith Donepudi

Contents:

Table of Contents

Abstarct

Image de-raining aims at restoring clean background image from a rainy image. Several traditional optimization-based methods have been suggested for modeling and separating rain streaks from background clean image. However, due to the complex composition [1] of rain and background layers, image de-raining remains a challenging ill-posed problem. To handle this issue, we got a better and simpler baseline de-raining network progressive recurrent network (PReNet). As for loss functions, single MSE or negative SSIM losses are sufficient to train PRN and PReNet. Experiments show that PReNet perform favorably on both synthetic and real rainy images. Considering its simplicity, efficiency and effectiveness, our models are expected to serve as a suitable baseline in future deraining research.

Theory:

Rain has an undesirable negative effect on the clarity of the collected images. In situation where images are captured in rain, it can lead to a loss of information and disability in reflecting real images of the situation. Consequently, rain has become an obstacle in outdoor scientific research such as image classification, object detection, and video surveillance. Single image deraining aims at restoring clean background image from a rainy image and has drawn considerable recent research attention. Moreover, several models are designed to improve computational efficiency by utilizing lightweight networks in a cascaded scheme or a Laplacian pyramid framework[7], but at the cost of obvious degradation in deraining performance. To sum up, albeit the progress of deraining performance, the structures of deep networks become more and more complicated and diverse. As a result, it is difficult to analyze the contribution of various modules and their combinations, and to develop new models by introducing modules to existing deeper and complex deraining networks. In this paper, we aim to present a new baseline network for single image deraining to demonstrate that:

- by combining only a few modules, a better and simpler baseline network can be constructed and achieve noteworthy performance gains over state-of-the-art deeper and complex deraining networks,
- The improvement of deraining networks may ease the difficulty of training CNNs[5] to directly recover clean image from rainy image

Preprocessing:

The RainTrainH is training dataset consists of both input and target images, the input image file name starts with #rain and the Target images consists of #derain. Rain100H is testing dataset which has 100 images of rainy (Input) and 100 images of norain or clear images(Target). Here there is no index available for both Test and Train data, so creating a csv file assigning index to file names. Splitting the train dataset into train and validation data.

Creating a RainDataset class:

Initialize training_df where training_df is the csv file and training_dir is local path of dataset; Transforms are the transformations which are applied to the dataset. create a function named get item where we will get the Observations and Target paths. Here the observation images start with name "rain" and Target Images start with norain. the csv files consists of norain in the column name, so for Observations we will consider from 2nd index of the (i.e rain) so here we use [2:] to split the name. The observations and Targets are converted in pixels by converting image into r g b channels. ALL image are no of same size so we will apply transform for the inages whose width is equal to 321. Converting the image arrays to Tensor by applying transforms. ToTensor() to Observations and Targets.

Calculating mean and standard deviation:

creating a small batch to calculate mean and standard deviation. so we are passing training_csv and train_dir and batch size of 8 to RainDataset which will return observations and target. Using data loader, we will load train_datset with batch size of 8. it will create the batches.

```
batch_size = 8
train_dataset = RainDataset(inidices_csv=training_csv, datset_dir=train_dir,
                                    transform=transforms.Compose([
                                                                  #transforms.PILToTensor(),
                                                                  #transforms.ToTensor()
                                                                  ]))
loader = DataLoader(
  train_dataset,
  batch_size = batch_size)

def mean_and_std(Data):
    count = 0
    mean = torch.empty(3)
    Square = torch.empty(3)
    for images, _ in Data:
        b, c, h, w = images.shape
        no_of_pixels = b * h * w
        sum_ = torch.sum(images, dim=[0, 2, 3])
        sum_of_square = torch.sum(images ** 2,dim=[0, 2, 3])
        mean = (count * mean + sum_) / (count + no_of_pixels)
        Square = (count * Square + sum_of_square) / (count + no_of_pixels)
        count += no_of_pixels
    std = torch.sqrt(Square - mean ** 2)
    return mean,std

mean, std = mean_and_std(loader)
print("mean and std: \n", mean, std)
```

Calculating Mean and Standard devation.

Calculating the mean and standard deviation by iterating the loader in for loop. find the shape of each image and assigning the variables and multiplying the variables gives the no of pixels. finding the sum of the each pixel for all the images and also squared sum of pixels. Mean is calculated by dividing sum by number of pixels and count of images. Then Squared Sum is sum of sum of squares divided by count and no of pixels. standard deviation is square root of (sum of square - square(mean)).

Architecture:

To construct a easier and useful network architecture, we choose loss function, input and output are baseline for the network. Residual network (ResNet)[6] with five residual blocks (ResBlocks) are starting steps in network architecture

ResNet: A residual neural network (ResNet) is an artificial neural network (ANN). Residual neural networks utilize skip connections, or shortcuts to jump over some layers. Typical ResNet[6] models are implemented with double or triple layer skips that contain nonlinearities (ReLU) and batch normalization in between. An additional weight matrix may be used to learn the skip weights; these models are known as HighwayNets. Models with several parallel skips are referred to as DenseNets. In the context of residual neural networks, a non-residual network may be described as a plain network.

ResBlock: The ResBlock is constructed out of normal network layers connected with rectified linear units (ReLUs) and a pass-through below those feeds through the information from previous layers unchanged.

Then, progressive ResNet (PRN) is introduced by recursively unfolding the ResNet into multiple stages without the increase of model parameters

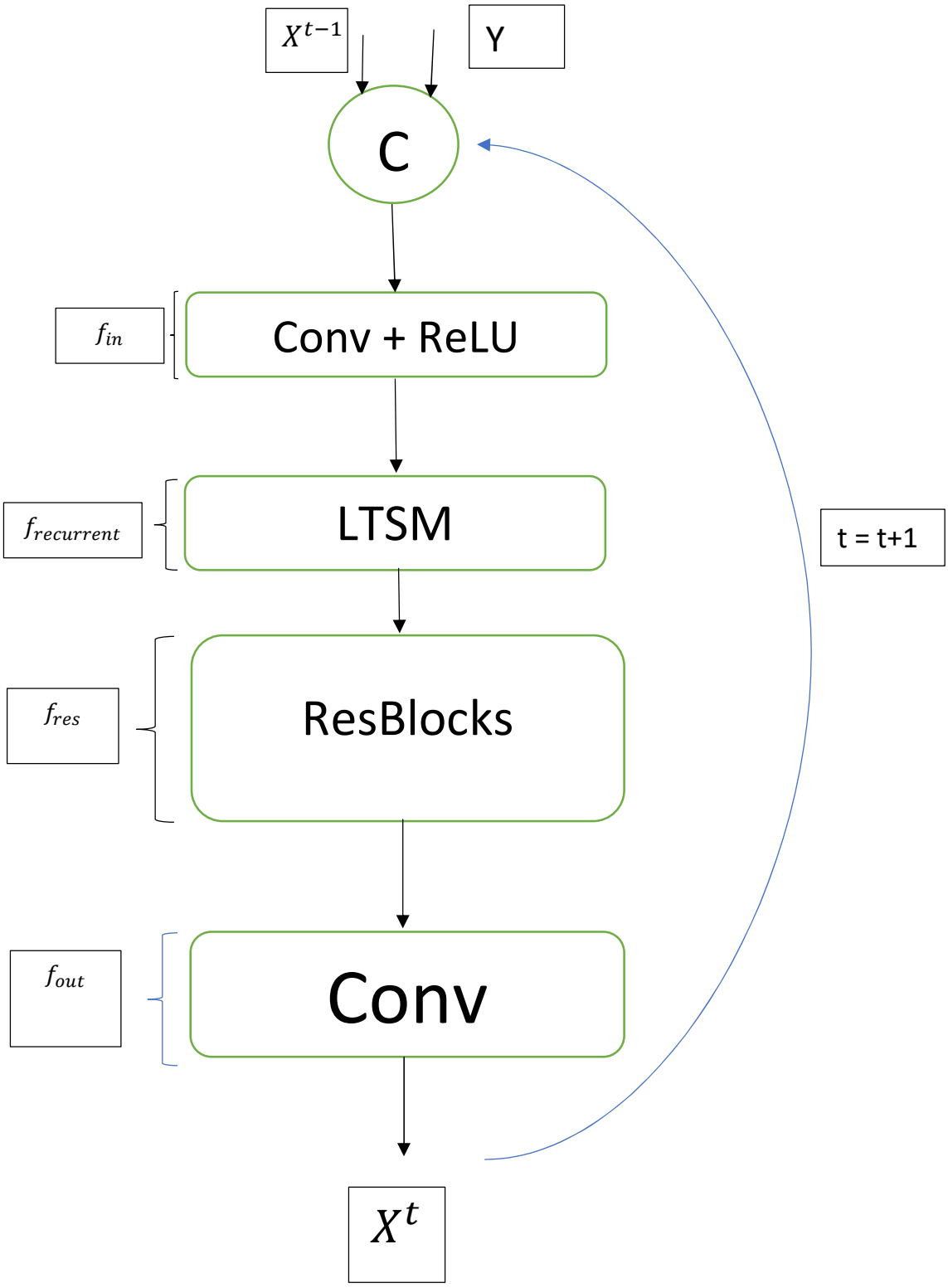


Figure 1 : The architectures of progressive networks, where f_{in} is a convolution layer with ReLU, f_{res} is ResBlocks, f_{out} is a convolution layer, frecurrent is a convolutional LSTM and c is a concat layer. f_{res} can be implemented as conventional ResBlocks or recursive ResBlocks

The impact of the loss function, input, and output of a network was additionally explored with the help of utilizing PReNet and PRN. For each ResNet, we consider network input as an actual rainy image and stepwise results. From observing the actual picture introduction its benefits from the deraining performance. For the rainy image network output, it predicts the rain streak layer from the adaptation of the residual learning formula. For neat background prediction, we use the possibility of instant understanding of PReNet and PRN models. For good deraining performance one MSE and SSIM negative loss from hyperparameters tuning[4] instead of hybrid losses with already trained PRN and PReNet

Progressive Recurrent Network

we adopt a basic ResNet with three parts: (i) a convolution layer fin receives network inputs, (ii) several residual blocks (ResBlocks) fres extract deep representation, and (iii) a convolution layer fout outputs deraining results.

We implemented a recurrent layer into PRN, by which feature dependencies across stages can be propagated to facilitate rain streak removal, resulting in our progressive recurrent network (PReNet). The only difference between PReNet and PRN is the inclusion of recurrent state s^t

$$X^{t-0.5} = f_{in}(X^{t-1}, y),$$

$$s^t = f_{recurrent}(s^{t-1}, x^{t-0.5}),$$

$$X^t = f_{out}(f_{res}(s^t)),$$

where the recurrent layer $f_{recurrent}$ takes both $X^{t-0.5}$and the recurrent state s^{t-1}as input at stage t − 1. $f_{recurrent}$ can be implemented using either convolutional Long Short-Term Memory (LSTM) or convolutional Gated Recurrent Unit(GRU)[2]. In PReNet, we choose LSTM due to its empirical superiority in image deraining.

By unfolding PReNet with T recursive stages, the deep representation that facilitates rain streak removal are propagated by recurrent states. The deraining results at intermediate stages in Fig. 1 show that the heavy rain streak accumulation can be gradually removed stage-by-stage.

Loss Function:

SSIM stands for Structural Similarity Index and is a perceptual metric to measure similarity of two images. Commonly used loss functions such as L2 (Euclidean Distance) correlate poorly with image quality because they assume pixel-wise independance. For instance, blurred images cause large perceptual but small L2 loss. Using PReNet (T = 6) as an example, we discuss the effect of

loss functions on deraining performance, including MSE loss, negative SSIM loss, and recursive supervision loss.

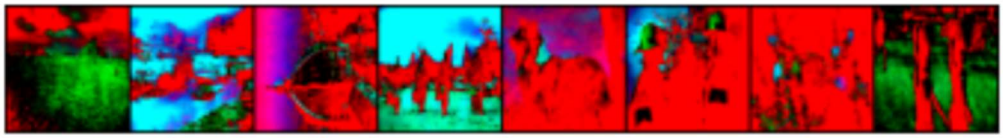
SSIM considers luminance, contrast and structure and is computed as follows:

$$\frac{(2\mu_x\mu_y+c_1)(2\sigma_{xy}+c_2)}{(\mu_x^2+\mu_y^2+c_1)(\sigma_x^2+\sigma_y^2+c_2)}$$

Negative SSIM vs.. MSE: We train PReNet models by minimizing MSE loss (PReNet-MSE) and negative SSIM loss (PReNet-SSIM), and Table 1 lists their PSNR and SSIM values on Rain100H. Unsurprisingly, PReNet SSIM outperforms PReNet-MSE in terms of SSIM. We also note that PReNet-SSIM even achieves higher PSNR, partially attributing to that PReNet-MSE may be inclined to get trapped into poor solution. And PReNet trained on 1,254 samples achieves SSIM 0.605. Moreover, even PReNetr can perform better than all the competing methods.

Results:

Our progressive network is evaluated on synthetic datasets, i.e., Rain100H [10]. Our PReNet achieves significant PSNR and SSIM gains over all the competing methods. And PReNet trained on 1,254 samples achieves **SSIM 0.605**. Moreover, even PReNetr can perform better than all the competing methods.



Output test images

The output will be more clear and precise if we train with more no epochs and updating hyperparameters. However according to reference paper 50 epochs are required to train the model. There are no GUPs available so we trained with 5 epochs and Batch size of 16.

Conclusion: In this project, a better and simpler baseline network is presented for single image deraining. Instead of deeper and complex networks, we find that the simple combination of ResNet and multi-stage recursion. Moreover, the deraining performance can be further boosted by the inclusion of recurrent layer, and stage-wise result is also taken as input to each ResNet, resulting in our PReNet model. Furthermore, the network parameters can be reduced by incorporating inter- and intra-stage recursive computation (PRNr and PReNetr). And our progressive deraining networks can be readily trained with single negative SSIM or MSE loss. Extensive experiments validate the superiority of our PReNet and PReNetr on synthetic and real rainy images in comparison to state-of-the-art deraining methods. Taking their simplicity, effectiveness and efficiency into account, it is also appealing to exploit our models as baselines when developing new deraining networks.

Future Scope: PReNet performed better with RainH dataset with 1000 images, Future Increase in number of images and update the hyper-parameters and no of iterations for training the Images will be able to get more accurate deraining images.

References:

[1] Y.-L. Chen and C.-T. Hsu. A generalized low-rank appearance model for spatio-temporally correlated rain streaks. In Proceedings of the IEEE International Conference on Computer Vision, pages 1968–1975, 2013. 1, 3

[2] K. Cho, B. Van Merrienboer, C. Gulcehre, D. Bahdanau, ¨ F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078, 2014. 4

[3] C. Dong, C. C. Loy, K. He, and X. Tang. Image super-resolution using deep convolutional networks. IEEE Transactions on Pattern Analysis and Machine Intelligence, 38(2):295–307, 2016. 1

[4] Z. Fan, H. Wu, X. Fu, Y. Hunag, and X. Ding. Residualguide feature fusion network for single image deraining. In ACM Multimedia, 2018. 2, 3, 5, 6, 7

[5] X. Fu, J. Huang, X. Ding, Y. Liao, and J. Paisley. Clearing the skies: A deep network architecture for single-image rain removal. IEEE Transactions on Image Processing, 26(6):2944–2956, 2017. 1, 2, 3, 4

[6] X. Fu, J. Huang, D. Zeng, Y. Huang, X. Ding, and J. Paisley. Removing rain from single images via a deep detail network. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 1715–1723, 2017. 2, 3, 4, 6, 7, 8

[7] X. Fu, B. Liang, Y. Huang, X. Ding, and J. Paisley. Lightweight pyramid networks for image deraining. arXiv preprint arXiv:1805.06173, 2018. 1, 2, 3, 5

[8] K. Garg and S. K. Nayar. Detection and removal of rain from videos. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2004. 3

[9] S. Gu, D. Meng, W. Zuo, and L. Zhang. Joint convolutional analysis and synthesis sparse representation for single image layer separation. In Proceedings of the IEEE International Conference on Computer Vision, pages 1717–1725, 2017. 1

[10] W. Yang, R. T. Tan, J. Feng, J. Liu, Z. Guo, and S. Yan. Deep joint rain detection and removal from a single image. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 1357–1366, 2017. 2, 3, 5, 6, 7, 8