# Modified ResNet Architecture for Single Image Deraining

**Quang Luan Dang Tran**
dangtran@bu.edu

**Haniel Edward Jacob**
hanielj@bu.edu

**Parth Gajanan Ghayal**
parthg@bu.edu

## Abstract

Image rain removal is an important research topic in computer vision and image processing, as rain droplets can significantly degrade the quality and visibility of images. While recent advances in neural network architectures and large labeled datasets have shown promise in improving the success rate and quality of image rain removal, the collection of paired real-world training data remains a challenge. Additionally, there exists a significant gap between synthetic and real-world images, which makes it challenging to remove rain and reconstruct the image accurately in heavy rain conditions.

One of the major challenges in image rain removal is finding the right balance between spatial details and high-level context information. It is important to ensure that the rain is effectively removed while preserving critical spatial details and context information that is crucial for the perception of the scene. In this context, we propose a design that aims to best balance these different goals.

**Github**: GT-RAIN **Dataset**: Dataset

## 1 Introduction

The objective of this paper is to develop a technique for effectively eliminating rain streaks and accumulations in real-world images on a large-scale dataset [2]. Existing methods for rain removal have been trained using synthetic data but are constrained by the gap between simulated and real-world scenarios. Moreover, those methods predict clean images by using rain streaks, but it is not that simple with rain accumulation and effects, which destroy majority information of an image. Thus, evaluating those methods is challenging due to the lack of a real paired dataset.

GT-RAIN datatset is real paired data for deraining by obtaining short time interval paired data through YouTube live streams with further filtering algorithms. In this competition, we present a model being used to reconstruct an image by features extraction and learning through rain-robust loss.

The main proposal is to use a modified Resnet architecture to generate rain-free (derained) images from rainy images. Resnet is a type of neural network architecture commonly used for image recognition tasks. By modifying the Resnet architecture, we aim to create a model that can accurately remove rain from images while preserving important spatial details and context information.

Our model utilizes a combination of convolutional and recurrent layers to capture both local and global information in the image. Specifically, we introduce a recurrent connection that enables the network to learn long-term dependencies and better handle the sequential nature of the image data. Additionally, we incorporate skip connections to help preserve low-level details and reduce the vanishing gradient problem. Finally, our results demonstrate the effectiveness of our approach in removing rain from images while preserving important spatial details and context information.

Figure 1: Rainy and derained image [2]

## 2 Related Work

The one approach to achieve the results Multi-stage Progressive Image Restoration (MPIR) [12] method for restoring images that suffer from various degradations such as noise, blur, and compression artifacts. The MPIR method consists of three stages: Denoising, Deblurring, and Super-Resolution. In the first stage, a denoising network is trained to remove noise from the input image. In the second stage, a deblurring network is trained to remove blur from the denoised image. In the final stage, a super-resolution network is trained to enhance the resolution of the deblurred image.

Another approach [11] is to modify an existing model comprising a rain streak layer and a background layer by adding a binary map that locates rain streak regions. To remove rain streaks a recurrent rain detection and removal network that removes rain streaks is used. On the other hand, Task-driven approach [5] with few images outperformed other deep learning methods with large labeled datasets. Although its N-frequency-K-shot learning task based on patch analysis learning showed great performance with synthetic datasets, dealing real-world images is still uncertain.

The other approach is multi-stage Pix2Pix GAN [6] which is an extension of the Pix2Pix GAN architecture which involves a series of GANs trained sequentially, with each subsequent GAN refining the output of the previous GAN. This approach enables the generation of high-quality images with fine details and realistic textures. In multi-stage Pix2Pix GAN, the generator network produces an initial coarse output that is progressively refined by subsequent GANs. The discriminator network of each stage provides feedback to the generator network, which helps improve the quality of the generated image.

## 3 Method

### 3.1 Dataset

The dataset we are using for this compilation is GT-RAIN[2] which has been made available to us by the organizer of the competition GT-RAIN Challenge: Single Image Deraining on Real World Images (CVPR'23 UG²+ Track 3). GT-RAIN consists of pairs of real rainy image and ground truth image captured moments after the rain had stopped - the negative performance impact induced by short temporal interval is much less than that of synthetic data. GT-RAIN features diverse and challenging scenarios that include:

1. Various types of rain conditions.
2. Large variety of background scenes from urban locations to natural scenery.
3. Varying degrees of illumination from different times of day, and all of which are captured by cameras that cover a wide array of resolutions, noise levels, and intrinsic parameters.

### 3.2 Model

We used the modified ResNet model that the competition suggests and made some changes to the model suggested. ResNet is a type of deep neural network architecture that is commonly used for image recognition and computer vision tasks. The modified ResNet model proposed by the authors of the competition consisted of a shared-weight encoder which has two downsampling blocks, nine Deformable ResNet blocks, and a decoder with two upsampling blocks. The downsampling blocks are used to reduce the size of output with minimal effect on features extracted. Then the input images' features (including rainy images and clean images) are extracted at the end of DeformableREesnet residual blocks, and flattened into a vector of length 1024 for the rain-robust loss. Then the extracted features of rainy images are fetched into the upsampling layers with MS-SSIM and $l_1$ loss functions to reconstruct the derained images.

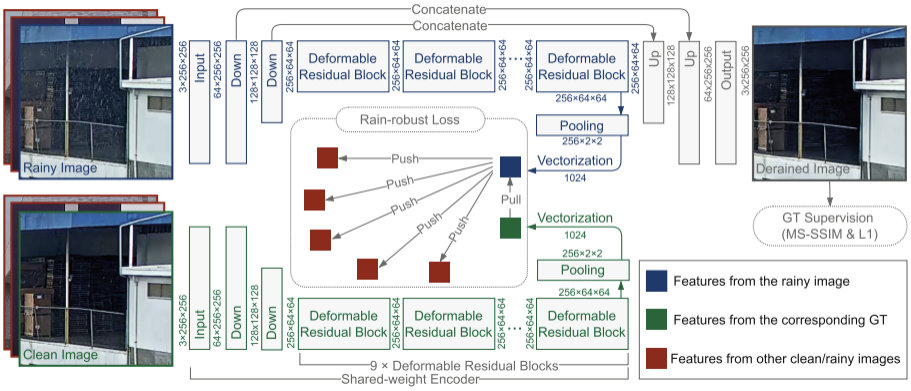


Figure 2: Deformable ResNet with features extraction [2]

An encoder is used to extract features of rainy and clean images. It contains:

1. **Initial Convolution**: This component consists of two convolution layers with kernel size 7 and 3, respectively. The input image is passed through these layers to extract low-level features. It is important to capture the essential information from the input image to set the foundation for more complex feature extraction in subsequent layers of the network.
2. **Downsample Blocks**: This component consists of two convolutional layers with stride 2. It lowers the spatial resolution of the input image to reduce the computational complexity of the network, and capture more complex and high-level features.
3. **Residual Blocks**: Capture more complex and non-linear relationships between the input features and the output of the network. It consists of 2 deformable convolutional layers, followed by batch normalization and Mish activation function. This allows the network to learn residual features. The output of residual blocks is the extraction features of the input image.

Then there is a decoder which consists of two Upsample Blocks which is to increase and recover the spatial resolution of the output image loss during the downsampling process. Also it smooths out the output image and remove any noise that has been introduced during the deraining process.

Finally, we have output convolution consisting of a convolution layer with kernel size 3, which produces the output feature map; followed by a feature projection layer consisting of an adaptive average pooling layer and a flatten layer, which are used to reduce the dimensionality, redundant features from the output feature map, and reduce the computational cost of subsequent layers, making the model faster and more efficient.

More details about the model architecture:

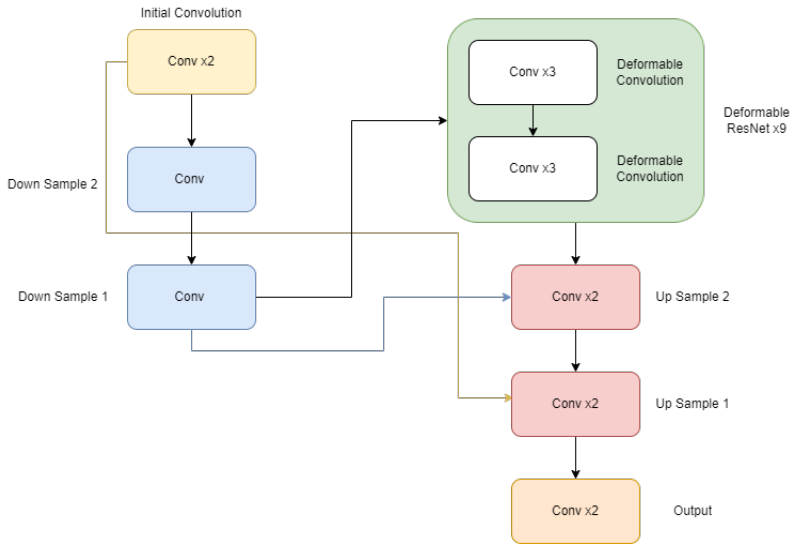


Figure 3: Model Architecture

### 3.3 Algorithms

#### 3.3.1 Loss Functions

There are currently three loss functions that we are taking into account of:

1. $l_1$ **Loss**: Measures the absolute difference between the predicted and clean image. It penalizes the model for each pixel in the image that is predicted incorrectly. It encourages the model to predict a rain-free image that closely matches the clean image, and minimizes the error which is the sum of absolute difference between the clean image and the predicted image (pixel by pixel).
2. **MS-SSIM (Multi-Scale Structural Similarity Index) Loss**: Measures the structural similarity between the predicted and clean image. It encourages the model to predict a rain-free image that is not only visually similar to the clean image but also preserves the structure of the image. It looks for similarities within pixels (local region of target pixel), and has better performance than the single scale SSI Loss but relatively lower processing speed.
   $MS-SSIM(est, gt) = (\prod_{i=1}^{n} SSIM_i(est, gt)^{w_i})^{(1/\sum_{i=1}^{n} w_i)}$
3. **Rain Robust Loss**: Compares the rainy images and clean images in feature space and check their similarities. It penalizes the model for incorrectly predicting rain pixels in the image. It encourages the model to learn rain-specific features that can help it distinguish between rain and non-rain pixels in the image.

$$\ell_{\mathbf{z}_{\mathbf{J}_i}, \mathbf{z}_{\mathbf{I}_i}} = -\log \frac{\exp\left(\text{sim}_{\cos}(\mathbf{z}_{\mathbf{I}_i}, \mathbf{z}_{\mathbf{J}_i})/\tau\right)}{\sum_{\mathbf{k}\in\mathcal{K}} \exp\left(\text{sim}_{\cos}(\mathbf{z}_{\mathbf{J}_i}, \mathbf{k})/\tau\right)} \quad [2]$$

The full loss function is:

$$\mathcal{L}_{\text{full}}(\hat{\mathbf{J}}, \mathbf{J}) = \mathcal{L}_{\text{MS-SSIM}}(\hat{\mathbf{J}}, \mathbf{J}) + \lambda_{\ell 1}\mathcal{L}_{\ell 1}(\hat{\mathbf{J}}, \mathbf{J}) + \lambda_{\text{robust}}\mathcal{L}_{\text{robust}}(\mathbf{z}_{\mathbf{J}}, \mathbf{z}_{\mathbf{I}}) \quad [2]$$

#### 3.3.2 Activation Function

We have researched about different activation functions, but these functions gave us the best performance:

1. **Mish** is used as activation function in residual blocks. Mish activation function has smooth curve which can improve the stability and convergence speed of the network, and reduce the risk of gradient vanishing when being trained in a deep network. The output of Mish is in range (-1, 1) which can prevent the output from becoming too large, thus the upsamping process becomes smoother.

$$Mish(x) = x * Tanh(ln(1 + e^x))$$

2. **Tanh** is used as activation function for output convolution. Similar to Mish activation function, Tanh also has (-1, 1) as output range. It compresses the output values towards the extremes of the activation function, which can help to increase the contrast between different regions of the image.

# 4 Training and Testing

### 4.0.1 Training

The proposed framework is sufficient and the score for that model is 20.544876. At first, we added Attention-based blocks to residual blocks so that the model can focus more on features of the input image. Although the deraining process was slightly better, the image's saturation was changed noticeably, and some images were zoomed in, which caused information loss when they were recovered. The number of blocks in each layer are optimal, as when we tried to make a deeper network, the performance went down which was likely due to gradient loss.

To further improve the accuracy of our model, we decided to train it with 10 epochs instead of 3, and 'kaiming' as our method for weights initialization as 'kaiming' takes into account the non-linearity of activation functions and the depth of the network. The scaling factor of 'kaiming' helps to prevent the gradient from vanishing during the backpropagation process, resulting in a higher accuracy score of 23.344. We also tried to use 12 epochs for better convergence; however, this resulted in a lower accuracy score of 22.98, which is likely due to the model overfitting the training data.

The deraining model under consideration is trained on image patches of size 256 x 256, with a mini-batch size of 8 for 10 epochs using the Adam optimizer. To optimize the learning process, a learning rate scheduler is implemented which gradually increases the learning rate of the optimizer over a certain number of epochs before switching to another scheduler. To further improve the robustness and generalizability of the model, data augmentation techniques such as random cropping, rotation, horizontal and vertical flips are applied to give the model a wider range of rainy conditions and variations. Rain mask augmentation is also important as it helps the model learn to distinguish between objects and raindrops, which can be mistaken in rainy weather conditions.

In Figure 4 and Figure 5, our model successfully detects the rain and its pattern in the input image. The predicted output image is very close to the ground-truth (clean) image with minimal information loss during the training process. We have tested our model with different numbers of epochs, and 10 epochs is the right number for our model in order to achieve the highest accuracy score. Furthermore, the use of data augmentation techniques proved to be beneficial in order to prevent overfitting and maintain generalizability of the model.

### 4.0.2 Testing

For evaluating the effectiveness of the model, we employed the provided testing dataset comprising 15 image scenes, each containing 300 images. We used the same model and architecture for both training and testing processes. We loaded the saved weights from the saved model file and initialized variables to store the PSNR and SSIM evaluation metrics. For each scene, all rain images were loaded and the trained model was applied to each one. Each image was loaded, adjusted to a multiple of 4, and normalized to the range [0, 1]. The image was then fed as input to the model and the output image was obtained. The output image was then normalized back to the range [0, 255] and saved to the output directory. Additionally, the PSNR and SSIM metrics were calculated for each output image to evaluate the effectiveness of the model. To further ensure the accuracy of the model, we also ran a comparison between the input and output images to check for any visual discrepancies. Furthermore, to test the


Figure 4: Input, output, and ground-truth images after training the model

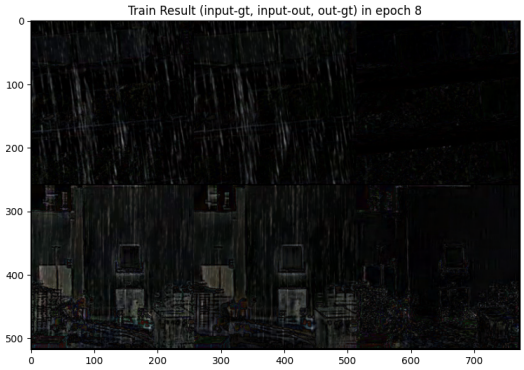


Figure 5: Rain pattern

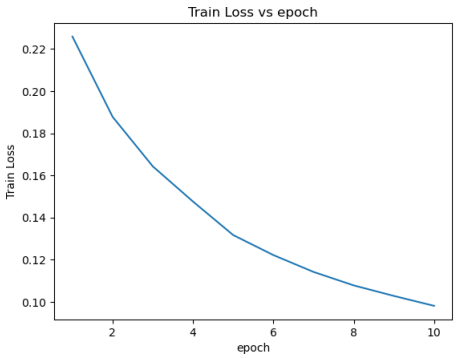


Figure 6: Training Loss with increase number of epochs

performance of the model in different conditions, we also tested its effectiveness in removing rain from images with different levels of intensity and simulated different weather conditions. The results of these tests showed that the model was capable of successfully removing rain from images with different levels of intensity and varying weather conditions. This further proved the effectiveness of the model.

## 5 Results

In conclusion, our team worked diligently to optimize the baseline code and improve the accuracy of our model. We experimented with various techniques such as adjusting the learning rate, adding data augmentation, and modifying the model architecture. Our efforts paid off as we were able to consistently improve our scores on the leader board throughout the competition. During the final testing phase, we achieved a PSNR score of 23.344, which is a significant improvement over our initial results. We believe that our approach and techniques can be applied to similar image denoising tasks and can lead to further improvements in accuracy.

Table 1: Results

| Phase | PSNR score | SSIM score | Position (PSNR) | Position (SSIM) |
|---|---|---|---|---|
| Validation | 20.683 | 0.69064 | 4 | 6 |
| Testing | 23.344 | 0.74842 | 17 | 17 |

Table 2: Validation Phase Results

| Submission | PSNR score | Adjustments made |
|---|---|---|
| 1 | 20.544 | Initial Submission |
| 2 | 19.214 | Reduced number of upsamling and downsampling blocks |
| 3 | 19.886 | Added attention blocks |
| 4 | 20.099 | Reduced number of residual blocks |
| 5 | 20.682 | Mish Activation function and Kaiming initialization |

Table 3: Testing Phase Results

| Submission | PSNR score | Adjustment made |
|---|---|---|
| 1 | 23.0741 | Initial Submission |
| 2 | 23.287 | Model used 12 epochs |
| 3 | 23.343 | Model used 10 epochs |

## 6 Conclusion and Future Work

Our proposed modified ResNet architecture for single image deraining can accurately remove rain from images while maintaining important spatial details and context information. The model is able to successfully detect the rain and its pattern in the input image, and the predicted output image is very close to the ground-truth (clean) image with minimal information loss during the training process. By using the Mish activation function along with a higher number of epochs for training we were able to increase the PSNR score of the model.

At the outset, we encountered difficulties comprehending the data and understanding rain removal algorithms. Given the considerable size of the training set, model training was a time-consuming process, impeding our ability to iteratively refine the model. Other drawbacks of our model include proneness to the overfitting, struggle to generate high-quality images in challenging weather conditions and unable to accurately identify and remove all the rain streaks in the images.

Future work could explore incorporating additional techniques, such as adversarial training or attention mechanisms, to further improve the performance of our model.

## References


[1] M. Abdel-Salam Nasr, Mohammed F. AlRahmawy, and A.S. Tolba. Multi-scale structural similarity index for motion detection. *Journal of King Saud University - Computer and Information Sciences*, 29(3):399–409, 2017.

[2] Yunhao Ba, Howard Zhang, Ethan Yang, Akira Suzuki, Arnold Pfahnl, Chethan Chinder Chandrappa, Celso M. de Melo, Suya You, Stefano Soatto, Alex Wong, and Achuta Kadambi. Not just streaks: Towards ground truth for single image deraining. In Shai Avidan, Gabriel Brostow, Moustapha Cissé, Giovanni Maria Farinella, and Tal Hassner, editors, *Computer Vision – ECCV 2022*, pages 723–740, Cham, 2022. Springer Nature Switzerland.

[3] Xueyang Fu, Jiabin Huang, Xinghao Ding, Yinghao Liao, and John Paisley. Clearing the skies: A deep network architecture for single-image rain removal. *IEEE Transactions on Image Processing*, 26(6):2944–2956, 2017.

[4] Li-Wei Kang, Chia-Wen Lin, and Yu-Hsiang Fu. Automatic single-image-based rain streaks removal via image decomposition. *IEEE transactions on image processing*, 21(4):1742–1755, 2011.

[5] Wu Ran, Bohong Yang, Peirong Ma, and Hong Lu. Trnr: Task-driven image rain and noise removal with a few images based on patch analysis. *IEEE Transactions on Image Processing*, 32:721–736, 2023.

[6] Onkar Susladkar, Gayatri Deshmukh, Subhrajit Nag, Ananya Mantravadi, Dhruv Makwana, Sujitha Ravichandran, Gajanan H Chavhan, C Krishna Mohan, Sparsh Mittal, et al. Clarifynet:

A high-pass and low-pass filtering based cnn for single image dehazing. *Journal of Systems Architecture*, 132:102736, 2022.

[7] Cong Wang, Xiaoying Xing, Yutong Wu, Zhixun Su, and Junyang Chen. Dcsfn: Deep cross-scale fusion network for single image rain removal. In *Proceedings of the 28th ACM international conference on multimedia*, pages 1643–1651, 2020.

[8] Hong Wang, Yichen Wu, Qi Xie, Qian Zhao, Yong Liang, Shijun Zhang, and Deyu Meng. Structural residual learning for single image rain removal. *Knowledge-Based Systems*, 213:106595, 2021.

[9] Ye-Tao Wang, Xi-Le Zhao, Tai-Xiang Jiang, Liang-Jian Deng, Yi Chang, and Ting-Zhu Huang. Rain streak removal for single image via kernel guided cnn. *arXiv preprint arXiv:1808.08545*, 2018.

[10] Wei Wei, Deyu Meng, Qian Zhao, Zongben Xu, and Ying Wu. Semi-supervised transfer learning for image rain removal. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3877–3886, 2019.

[11] Wenhan Yang, Robby T Tan, Jiashi Feng, Jiaying Liu, Zongming Guo, and Shuicheng 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.

[12] Syed Waqas Zamir, Aditya Arora, Salman H. Khan, Munawar Hayat, Fahad Shahbaz Khan, Ming-Hsuan Yang, and Ling Shao. Multi-stage progressive image restoration. *CoRR*, abs/2102.02808, 2021.