




Removing rain from images by means of an autoencoder architecture

Remoción de lluvia en imágenes por medio de una arquitectura de autoencoder

Alberto Ceballos Arroyo , Sergio Robles Serrano 
German Sanchez Torres 
Universidad del Magdalena, Colombia

Open Access

Recibido:

9 de septiembre de 2019

Aceptado:

19 de noviembre de 2019

Publicado:

31 marzo de 2020

Correspondencia:

gsanchez@unimagdalena.edu.co

DOI:

<https://doi.org/10.17081/invinno.8.1.3608>



© Copyright: Investigación e Innovación en Ingenierías

Abstract

Objective: To use computational techniques for removing rain from images. This is motivated by the fact that for many computer vision systems, correctly capturing the scene is a key need, and if such systems receive images degraded by rain as input, their performance may be compromised. **Methodology:** We built a dataset comprised by 11000 synthetic rain images. We resized and normalized all the images, then we employed 9000 of them for training the autoencoder architecture. The autoencoder outputs a de-rained version of the image which is then lighting corrected in order to produce the final, de-rained image. **Results:** We determined the best autoencoder architecture was a 6-layer autoencoder. We evaluated it on the remaining 2000 images, resulting in a Mean Squared Error of 0.61 and Structural Similarity Index of 0.8493, which means a fair amount of information from the rain-degraded images was recovered. **Conclusions:** The results we obtained were superior to proposals based in the spatial / frequency domain reported in the literature. However, we determined that it is possible to improve on the current results if we consider the frequency domain as part of the architecture. Thus, options for future work include combining machine learning-based approaches with frequency domain-based image processing.

Keywords: Image processing, noise removal, rain removal, autoencoder.

Resumen

Objetivo: Usar técnicas computacionales para eliminar lluvia en imágenes. La motivación viene dada por el hecho de que, para muchos sistemas de visión por computadora, es clave capturar correctamente la escena, y si estos sistemas reciben imágenes degradadas por lluvia como entrada, su funcionamiento puede verse comprometido. **Metodología:** Se creó un conjunto de datos compuesto por 11000 imágenes sintéticas de lluvia. Estas fueron redimensionadas y normalizadas, para luego utilizar 9000 de ellas como conjunto de entrenamiento en la arquitectura *autoencoder*. El *autoencoder* genera una versión sin lluvia de la imagen, la cual es pasada a una etapa de corrección de iluminación para producir la imagen final sin lluvia. **Resultados:** Se encontró que el mejor desempeño lo cumple el autoencoder de 6 capas. Se evaluó con las 2000 imágenes restantes, lo que resultó en un error cuadrático medio de 0,61 y un índice de similitud estructural de 0,8493. Esto significa que el modelo fue capaz de recuperar una gran cantidad de información original de las imágenes degradadas por la lluvia. **Conclusiones:** Los resultados obtenidos son superiores a aquellos de la literatura que se basan en el dominio espacial / frecuencial. Se determinó, sin embargo, que es posible obtener mejores resultados si se considera el dominio de la frecuencia como parte de la arquitectura, debido a las propiedades de esta. Por lo tanto, se propone a futuro combinar enfoques basados en el aprendizaje de máquina con el procesamiento de imágenes basado en el dominio de la frecuencia

Palabras claves: Procesamiento de imágenes, eliminación de ruido, eliminación de lluvia, *autoencoder*.

Como citar (IEEE): A. Ceballos-Arroyo, S. Robles-Serrano., y G. Sanchez-Torres, "Removing rain from images by means of an autoencoder architecture", Investigación e Innovación en Ingenierías, vol. 8, n°. 1, 2020. DOI: <https://doi.org/10.17081/invinno.8.1.3608>

Introduction

Computer vision systems are prevalent in both industry and research, due to many technological applications having a strong visual component [1,2]. Several kinds of climate conditions exist, both dynamic and static [3, 4]. Among dynamic climate conditions, we can find rain. Rain has a negative impact on the performance of computer vision systems exposed to it [3,5], as a result of rain droplets having particular features and occluding the scene behind them. These elements and their streaks vary in size depending on wind and image acquisition conditions, and almost always cover several pixels [5]. Because of this, researchers treat single-image rain removal as a denoising problem [6,7].

Denoising techniques are employed to obtain an approximation of a ground-truth image from a degraded one [7]. In the literature, authors present several approaches for restoring images degraded by rain. These can be classified into two main categories: methods based on spatial and frequency analysis [4], and methods based on learning [8]. Spatial/frequency-based approaches are based on removing rain from images by means of filters or decompositions, while learning-based methods remove rain by modeling the relationship between ground-truth images and degraded images through machine learning or other kinds of learning algorithms.

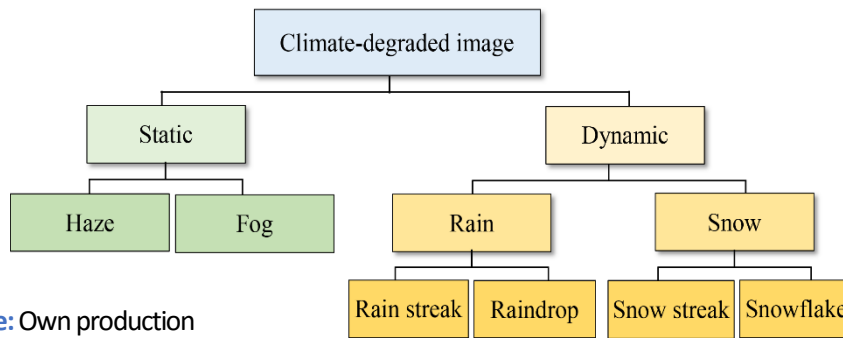
Deep learning architectures trained on synthetic images have been employed in recent work for removing noise (including rain) from images [8]. In such work, network architectures are often based on the *ResNet* residual learning framework [9]. Another approach is the use of autoencoders, which learn to reconstruct data from compressed representations and have been used for image denoising [10]. However, autoencoders have not yet been employed for removing rain from single images. For this reason, we propose implementing an autoencoder architecture for rain removal.

Regardless of the approach, some metric must be employed to measure how well algorithms restore degraded images [11]. When there exist pairs of degraded and ground-truth images, this process is straightforward: it is only necessary to measure error between both. Some quantitative metrics are the Mean Square Error (MSE), Peak Signal-to-Noise Ratio (PSNR), and Structural Similarity Index (SSIM) [11]. Qualitative metrics such as perceived visibility can also be employed, if there exists no set of ground-truth images, or if there is a need for complementing quantitative metrics [8,12].

We provide an overview of how rain degrades images in Section II. In Section III, we analyze several rain removal approaches, the autoencoder architecture, and the most frequent quality metrics currently employed. In Section IV we propose an algorithm for generating simulated rain images from a dataset of 2000 ground-truth images. Next, in Section V, we propose an autoencoder architecture for removing rain from single images. Finally,

in Sections VI and VII we show our results as well as our conclusions and insight for future research.

Figure 1. Taxonomy of visibility-degrading climate conditions in images



Source: Own production

Context

There exist several kinds of climate conditions that have an impact on image quality. The two main categories are static and dynamic conditions. Static conditions include haze and fog. Dynamic conditions are comprised of rain and snow [3,12]. Only rain is considered in the scope of this work. Rain can take different forms in images: either as a full element (rain droplet), or as a streak resulting from the camera not having enough exposure time. Figure 1 depicts this taxonomy. Due to reflection, pixels degraded by rain have higher intensity levels than their neighbors [4]. Furthermore, the shape of such elements can vary depending on their size, as well as on wind speed and direction [6].

Rain is semitransparent and does not fully occlude the objects; instead, it blurs them. In consequence, if background intensity levels are low, rain intensity levels become higher [4]. Otherwise, the latter will be lower. Furthermore, streaks left by rain when exposure time is high, share the same general direction in the whole image. Figure 2 shows an example of an image degraded by rain. The radius of rain droplets, when approximated to spheres, ranges between 0.1mm and 3.5mm, although most of them have a radius of less than 1mm [5].

The rain removal problem

We treat the issue of restoring images degraded by rain as a noise removal problem. Under many models, authors consider noise to be additive, as shown in (1).

$$G(x, y) = I_O(x, y) + I_N(x, y) \quad (1)$$

Figure 2. An example of an image degraded by rain



Source: [6]

Where G is the acquired image, I_O is the ground-truth image and I_N is the noisy layer. However, we cannot apply predetermined noise models in the case of rain-degraded images, resulting in the need for implementing more elaborate techniques. Regarding the camera, parameters such as exposure time, depth of field, and lighting conditions are relevant [6].

Previous work

In this section we describe the taxonomy of single-image rain removal techniques. This results in two main categories: spatial/frequency filtering-based methods, and learning-based methods [4,8]. Other categories, such as methods based on temporal connectivity, are outside the scope of single-image rain removal. For each category, we provide a brief description as well as details of some relevant work. We delve deeper into the strengths and limitations of each referenced paper in Table 1.

Spatial/frequency-based-filtering

One option for treating rain in single images is to employ spatial/frequency-domain filters. Spatial-based filters involve using morphological techniques or convolution filters for removing rain while retaining the highest amount of detail from the scene. Authors in the literature employ these techniques when the attributes of rain are known, and they base their filters on the size and shape of these elements.

Xu *et al.* [6] use a guided filter for removing rain. The guided filter is an edge-preserving spatial filter for removing noise in images [13]. Its inputs are the image to be processed and a guidance image which can either be the same image or a version of it where border information is highlighted. In their work, the authors [6] employ a gray-scale version of the original image as the guidance image. Their implementation removes rain to an acceptable degree; however, it does not preserve details from elements such as characters, symbols, and signs. For this reason, it is not appropriate for certain applications that might require preserving such details, although it makes segmentation tasks easier.

Frequency-based filters are instead employed to remove rain, based on the analysis of highly frequent patterns [1]. For instance, some authors remove

high frequency elements with low pass filters in the frequency domain. Although frequency-based approaches are effective by themselves, researchers may employ them together with spatial filters for better results. This can be done in several ways. One option is to decompose the image into its low frequency (LF) and high frequency (HF) components for reducing noise with a spatial filter. Another option is to employ frequency-based filters for removing rain components and spatial-based filters for smoothing the resulting image.

Zheng *et al.*, [14] improve upon the algorithm proposed in [6] by employing a multi-guided filter. They first separate the LF and HF components of the image with a guided filter. Then, they enhance the edges in the LF component. Afterward, they apply another guided filter on the HF component, which contains most of the rain, using the enhanced LF component as the guidance image. The next step consists in combining the processed LF and HF components to obtain the recovered image. The authors then generate a slightly deblurred version of the recovered image by creating a new image from the pixel-wise minimums between the ground-truth image and the recovered one. Finally, they use the deblurred image as a basis for one last run of the guided filter on the recovered image. They remove slightly more rain than [6] while retaining detail from characters, symbols and signs (see Figure 3). Processing time is however increased by cause of the guided filter executions. Furthermore, the authors do not employ any quantitative quality metric in their study.

In general, spatial and frequency domain-based methods are efficient for removing rain from single images [6, 14, 15, 16, 17]. Whereas most proposals apply filters globally, some take into account local features resulting in greatly enhanced images [18]. Running times depend on the complexity of the de-raining pipeline, with some of the more elaborate proposals being slower to run [14]. These methods do not require a set of training images, making them easier to implement when little data is available [6, 15] However, in some cases this means the default configurations of these algorithms may need to be adjusted for cases where rain is heavier [18].

Furthermore, in some proposals the resulting images are slightly blurred and small details which might be necessary for some applications are not preserved [6, 15, 17]. In some instances, however, further segmentation tasks are eased due to edge enhancement [6, 14, 19]. Another issue, which is common in this category, is the lack of quantitative methods for measuring de-raining quality, which makes it difficult to assess effectiveness [6, 14, 15, 16, 18].

Learning-based methods

Some authors propose learning-based techniques for removing rain in single images. These methods are comprised of architectures to learn to detect and remove rain droplets and streaks. Two main categories of learning-

based methods exist in rain removal literature: dictionary learning and machine learning methods.

Figure 3. Image restored by means of the multi-guided filter approach. To the left, the original image. To the right, the restored version



Source: [14]

In dictionary learning methods, the authors map visual features into sparse structures composed of elements called atoms. Atoms are linear combinations of the most representative elements from an image [7]. Once a collection of atoms (that is, a dictionary) is built, the authors can then reconstruct the image, removing certain visual elements from it. In the case of images degraded by rain, some authors partition the images into LF and HF components by means of spatial and frequency filters, such as the guided filter and the bilateral filter.

Most rain streaks in an image are present in its HF component, therefore, the authors can train the dictionaries on these components while ignoring the low-frequency portion. As a result, they can obtain a dictionary comprised mostly by rain atoms. Due to the properties of rain, these atoms are similar among themselves. Hence, a denoised version of the HF component can be obtained by grouping together rain atoms and moving them into another dictionary [4].

Kang, Lin, and Fu [7] propose a method for removing rain elements from the image based on dictionary learning. They first decompose the image into HF and LF components by using a bilateral filter. Next, they employ morphological components analysis (MCA) for creating a sparse dictionary of both rain and non-rain components in the HF component, represented as 15x15 patches. Afterward, they divide the dictionary into two by performing HOG (Histogram of Gradients) based atom clustering. The next step consists in carrying out sparse coding based on the two dictionaries for obtaining a rainless version of the HF component. Finally, they combine the restored HF component with the LF one for obtaining the denoised image. Their method efficiently removes rain from single images. Furthermore, it is self-contained, meaning it does not require a training dataset. However, processing times are elevated.

In general, dictionary learning-based methods are highly effective at removing rain from single images while preserving detail since they are mostly based on the processing of local features [4, 12, 20, 21, 22]. This sometimes results in some degree of detail loss when dealing with images

with elements similar to rain streaks, since some non-rain features might be treated as rain [22]. Furthermore, processing times are often elevated and the de-raining pipeline often includes spatial and frequency domain-based algorithms which add to the computational costs [4, 12]. However, unlike such methods, most dictionary learning-based proposals are evaluated with quantitative metrics such as SSIM and PSNR [4, 20, 21, 22]. In addition, they do not require sizable datasets, unlike the machine learning approach, although this results in a lower capability for generalizing to cases such as images taken under heavy rain conditions [4, 20, 22].

The machine learning approach involves training machine learning methods with datasets of noisy and clean image pairs. Ideally, these algorithms should learn the relationship between a clear, noise-free image and a counterpart degraded by rain. Although reasonable results can be obtained with traditional machine learning methods, treatment of high-resolution images is difficult to carry out without more modern deep learning techniques. Deep architectures can codify visual features in their layers and help remove complex noise conditions such as rain. However, they require sizable training datasets for learning the required relationships [8, 23].

Fu *et al.*, [8] propose using a deep detail network for removing rain from single images. In their work, they employ a deep detail network based on a deep residual network (ResNet). In such architectures, the dimensionality of the input is reduced for obtaining better results. They improve the rain removal process by training the network on the HF component of the rainy images, removing background interference. Their loss metric is the residual between the ground-truth images and their noisy counterparts. They train their algorithm with a dataset of 14000 original image-synthetic image pairs. They measure the results on the dataset using SSIM, with an average precision of 0.86. To measure quality on actual rainy images, they instead use a subjective visual quality metric. The drawbacks of their architecture include elevated training times and the need for a sizable dataset.

Mondal *et al.*, [23] proposed a deep de-raining neural network based on trainable mathematical morphological operators. Such operators are often useful for image de-raining only if their shape and size is adjusted to the conditions of individual images. Thus, by learning the shapes of dilation and erosion operators based on a dataset comprised of 1000 clean-rainy image pairs, they were able to effectively address the de-raining problem with a relatively small (and thus fast) neural network. They measured their results using the SSIM and PSNR metrics, reaching 0.92 SSIM and 28.03 PSNR. They also evaluated their proposal qualitatively, based on real rain images.

Deep learning architectures

The most common deep learning approach is to employ Artificial Neural Networks (ANNs) [24]. ANNs are biologically inspired machine learning

algorithms comprised of several layers, each of which consists of several nodes or artificial neurons. Nodes in one layer are connected to those of the next by weights, thus allowing the network to model arbitrary functions. Except for the input nodes (that is, the raw input data), ANN nodes contain activation functions for processing the data passed by previous layers. Layers in an ANN can be classified into input layers, which represent input data and dimensionality, hidden layers, which consist of activation functions that manipulate data, and the output layer, which is comprised of one or more nodes representing the target function.

The way supervised ANNs learn patterns is by processing labeled data. In each training iteration, the weights are adjusted by means of forward-propagation and back-propagation. In forward-propagation, input data is passed through every layer l in the network. These layers are comprised of several nodes. Each node consists of a linearity (see Eq. 2) and an activation function g (see Eq. 3) through which the data (input X or the output of another layer) is processed prior to being passed to the next layer. This process usually implies changes in dimensionality as the input data is passed deeper into the network.

$$Z^{[l]} = W^{[l]}A^{[l-1]} + b^{[l]} \quad (2)$$

$$A^{[l]} = g(Z^{[l]}) \quad (3)$$

Where $W^{[l]}$ is the array containing the weights and $b^{[l]}$ is the bias, both corresponding to layer l , $A^{[l-1]}$ is the activation output by layer $l-1$, and g is a non-linear activation function.

Once data reaches the last layer of a supervised ANN, the next step is to calculate prediction loss. The evaluation of how similar the results of the network are to the reference, labeled data is done by means of a loss function. For the purposes of denoising images, pixelwise comparison functions such as the Mean Squared Error are often employed (4). The total loss J (also deemed cost) is thus the average of the loss function L evaluated on the full dataset (5).

$$L^{(i)} = \frac{1}{width*height} \sum_{k=1}^{width} \sum_{j=1}^{height} (y^{(i)[k,j]} - \hat{y}^{(i)[k,j]})^2 \quad (4)$$

$$J = \frac{1}{M} \sum_{i=1}^M L(y^{(i)} - \hat{y}^{(i)}) \quad (5)$$

For a training set with M training instances, $\hat{y}^{(i)}$ is the predicted value, and $y^{(i)}$ is the ground truth value, for training example i . In the case of image denoising, $\hat{y}^{(i)}$ is a denoised version of image i and loss L is the average of the loss function for the pixels in the predicted denoised image and the ground truth image with no noise.

On the other hand, back-propagation aims to minimize prediction loss based on the optimization of the weight and bias values. In the literature, one of the earliest optimization methods to be employed was batch gradient descent, where the optimizable parameters are updated once for every training iteration (epoch) based on the full dataset, as in (6). Once the training process finishes, the ANNs can be used to carry out predictions on unknown data. In the case of batch gradient descent, the weights and bias terms are updated as in (6) and (7).

$$W^{next} = W - \delta \Delta W \quad (6)$$

$$b^{next} = b - \delta \Delta b \quad (7)$$

where W^{next} is the updated weight array, W is the current set of weights, b^{next} is the updated bias term, b is the current bias term, δ is the learning rate (a hyperparameter which determines how fast the parameters should be optimized), and ΔW and Δb are the gradients for W and b respectively. The increase in the availability of computational resources, as well as advanced research in backpropagation methods, have allowed researchers to employ deeper machine learning architectures [25]. Deep artificial neural networks consist of many hidden layers, thus allowing for the modeling of more complex computational problems. In the case of image denoising, Convolutional Neural Networks (CNNs) are often employed. This is done because their multi-layered structure learns visual features in the form of convolutional filters.

Layers in CNNs are structured differently from standard ANNs. Hidden layers in CNNs are comprised of trainable convolutional filters, akin to traditional blurring and edge detection kernels, and non-trainable pooling filters which can be used for reducing the dimensionality of data as it is passed into the deeper layers of the network. Another difference with respect to ANNs is that convolutional filters are not fully connected, that is, connections between two layers are handled in a localized manner, greatly reducing the number of trainable parameters. In this way, the input data X is transformed and stacked into differently shaped volumes of data when passed through the convolutional layers, as described in Eq. (8):

$$A^{[l]} = g(W^{[l]} * X + b^{[l]}) \quad (8)$$

where $*$ is the convolution operator, g is an activation function, W is the weight array, b is the bias term, and A is the output of layer l . Of great importance for the convolution operation is its effect on the resulting shape of data, which is determined by the stride s and padding p parameters, as well as kernel size k . The relationship between an input volume of height and width i , and the height and width o of the output volume, is defined in Eq. (9).

$$o = \left\lfloor \frac{i+2p-k}{s} \right\rfloor + 1 \quad (9)$$

The max-pooling operation is based on stride s and padding p . In this operation, a sliding window is passed over each channel of the input volume such that the pixels in each channel of the output volume take the maximum local value in each window defined by s , p , and k , as in Eq. (10).

$$o = \left\lfloor \frac{i-k}{s} \right\rfloor + 1 \quad (10)$$

Another relevant operation for our work is the transposed convolution operation, which maps lower dimensional into bigger higher dimensional content. This is done by reversing the order of the forward and backward passes in the standard convolution operation, thus enabling us to generate denoised images with the same resolution than that of the input images. When carrying out image denoising, deep CNNs with an autoencoder architecture are often employed [10]. Autoencoders learn to compress data and to reconstruct a modified version of it depending on the objective function. In the case of image denoising, these networks learn to produce a denoised version of every pixel in the noisy image.

Autoencoder architectures

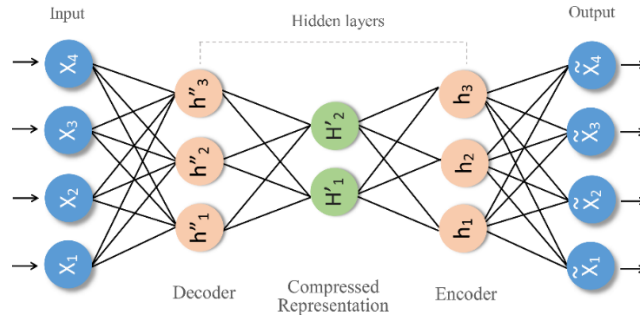
An autoencoder is a computational model comprised of a codifier and a decodifier. The main task of such a model is to learn to codify and decodify a given input. This is particularly useful when trying to represent data with a smaller set of features and when trying to reconstruct it while removing undesired features [24]. There exist several kinds of autoencoders and they can be employed to solve various problems. For instance, it is possible to train an autoencoder to return modified versions of the input data. One such example is the denoising autoencoder, which is capable of learning to remove noise from data. Autoencoders codify data (that is, they reduce it to its basic components) and then reconstruct it through the decodifier [10].

Likewise, there exist autoencoders capable of generating information. For example, an autoencoder can be trained to restore information lost in degraded sections of an image, such as human faces. This depends, mostly, on the output associated to each input, which allows the model to be adjusted in order to obtain the expected results. Figure 4 depicts a typical autoencoder architecture, whereas Figure 5 depicts our proposed methodology.

Metrics for image quality

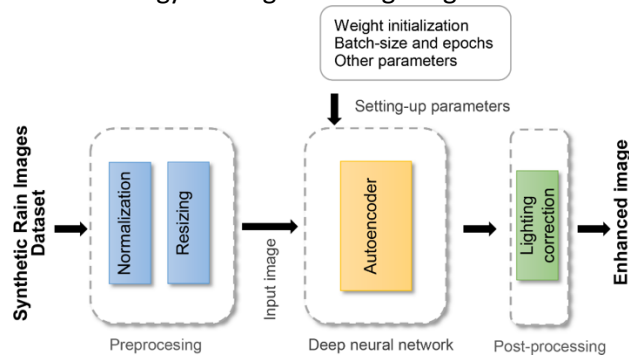
There exist several metrics in the literature to measure the accuracy of methods for removing rain in single images. These include: MSE [20], PSNR [3], SSIM [3], Visual Image Fidelity (VIF) [17], and Feature Similarity (FSIM) [17].

Figure 4. A typical autoencoder architecture



Source: Own production

Figure 5. Proposed methodology for image deraining using autoencoders



Source: Own production

Image quality metrics measure image similarity in various ways, however, all of them require a degraded image and a ground-truth image for measuring restoration quality.

These metrics are not only intended for verifying image restoration quality. For example, Sun, Fan, and Wang [12] use the SSIM metric for classifying dictionary atoms as rain and non-rain. On the other side, machine learning methods [8] may include one or more of these metrics as part of their loss functions, which guide the training process. In our work, we intend to employ three metrics to measure denoising quality: MSE, PSNR and SSIM.

Mean Square Error

In statistics, the mean square error (MSE) of a model measures the average of the square of the errors, that is, the difference between the model and

the ground truth. The mean of the pixelwise difference between values of two images can be computed through MSE, as shown in Equation 11.

$$MSE = \frac{1}{MN} \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} || I(i, j) - K(i, j) ||^2 \quad (11)$$

Where M, N are the dimensions of the ground-truth image I and the restored image K.

Peak Signal-to-Noise Ratio

The Peak Signal-to-Noise Ratio (PSNR) defines the relation between the maximum possible energy for a signal and the noise that degrades its representation. PSNR is usually expressed logarithmically, using the decibel as unit. It can be computed based on MSE as shown in Equation 12.

$$PSNR = 10 * \log_{10} \left(\frac{MAX_I^2}{MSE} \right) \quad (12)$$

Where MAX_I is the maximum value the signal can take, which is usually 255 for 8-bit images.

Structural Similarity Index

Also known as SSIM, this metric is employed to measure similarity between two images. Unlike MSE and PSNR, it considers structural neighborhood information. It is calculated based on Equation 13.

$$SSIM(x, y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \quad (13)$$

Where x, y are N*N windows extracted from the ground-truth and the restored image, μ_x is the mean of x, μ_y is the mean of y, μ_x^2 is the variance of x, μ_y^2 is the variance of y, σ_{xy} is the covariance of x, y, and c_1, c_2 are stabilization parameters.

Authors of proposals for solving other image enhancement problems, employ border detection precision and histogram analysis [26] when no pairs of noisy and ground-truth images exist, albeit no such technique was employed in the reviewed literature.

Methodology

Synthetic dataset generation

We built a synthetic dataset of rain-degraded images based on the following assumptions:

- All rain streaks in an image have the same general direction.
- The angle of rain streaks relative to the Y axis ranges from -45° to 45°.

- The position of each rain streak in a single image is random.

The steps of our synthetic rain image generation algorithm are as follows:

- Generate an empty mask M with the same size as the original image I .
- Apply uniform noise on a portion p of the pixels in M .
- Apply a vertical blur filter with size L on M .
- Apply a rotation transformation on M , with random angle θ taking values from -45° to 45° .
- Combine I and M based on the screen-blend mode, as in equation 14 [27].

$$G(x, y) = 255 - (255 - I(x, y)) * (255 - M(x, y)) \quad (14)$$

Our ground-truth image database is comprised by 2200 images compiled from various datasets. These datasets are oriented toward tasks like noise removal and scene recognition, and thus provide us with a wide selection of natural scenes [28, 29,30, 31]. Figure 6 depicts an example of an image from the dataset and its synthetic counterpart.

Based on our algorithm for synthetic rain image generation, we generated 11000 synthetic images using different values of p .

Figure 6. To the left, the original image. To the right, a version of the same image degraded with synthetic rain



Source: Own production

The methodology followed in this paper is comprised of several steps. First, preprocessing (resizing and normalization) is carried out on the dataset produced in section IV. Then, a convolutional denoising autoencoder is trained and used to denoise rain images. Finally, postprocessing in the form of lighting correction is carried out on the resulting images. This process is summarized in Figure 5, while the detailed neural network architecture is described in Figure 7.

Preprocessing

Images in the dataset were normalized by subtracting the mean of the dataset and dividing each image by the standard deviation of the dataset, as

shown in Eq. (15). In this way, we intend to improve performance by only passing centered data through the network, as per the literature [24].

$$X_{normed} = \frac{\bar{X}}{\sigma_X^2} \tag{15}$$

The other preprocessing steps was to resize every image to 256x256x3 pixels and to split the dataset as shown in Table 1.

Table 1. Training, development, and test dataset splits

| Set | Percentage (%) | Images |
|-------------|----------------|--------|
| Training | 81.8% | 9000 |
| Development | 9.09% | 1000 |
| Test | 9.09% | 1000 |

Source: Own production

Autoencoder architecture

A widely employed autoencoder architecture is the convolutional autoencoder. The reason for using convolutional layers in an autoencoder is their effectiveness when built into neural networks for image-related tasks. The weights of such layers are modelled differently from traditional ones, as they are based on convolutional image filters. That is, convolutional layer weights are image filters, and they can be described as follows:

$$h^i = s(x * W^i + b^i)$$

Where * is a convolution, s is an activation function, W is the weight, b is the bias, and h is the output of the layer.

Weight initialization

The weights in the architecture were initialized using Glorot uniform initialization [32], where the initial values of weights in a layer are defined based on uniform sampling within [-r, r], as defined in (17).

$$r = \sqrt{\frac{6}{n_w^{[l-1]} + n_w^{[l]}}} \tag{17}$$

where $n_w^{[l-1]}$ is the number of weights for the previous layer l-1 and $n_w^{[l]}$ is the number of weights for the current layer l.

Batch size and epochs

A widely employed variant of gradient descent is mini-batch gradient descent. In this version of gradient descent, backpropagation is carried out independently on several subsets of the training set, also known as batches, each of them of size $m < M$. Weights are updated once for each batch for

every training epoch, which results in lower memory requirements. In addition, since weights are updated on each subset, there is a reduced chance of the backpropagation algorithm getting stuck in local minima [33]. The dataset should also be shuffled at the start of every epoch in order to further increase the effectiveness of the optimization algorithm. [34]. We set the batch-size hyperparameter $m = 32$ after testing several values, while the number of epochs was set to 20.

ADAM optimizer

Adaptive Moment Estimation (ADAM) is a gradient descent algorithm which computes adaptive learning rates [35]. This results in a smaller probability of the algorithm getting stuck on local minima. This is done by storing an exponentially decaying running average of previous gradients $V_{\Delta W}$ (estimation of the first momentum) and the square of previous gradients $S_{\Delta W}$ (estimation of the second momentum), as shown in Eqs. (18) and (19).

$$V_{\Delta W} = \beta_1 V_{\Delta W} + (1 - \beta_1) \Delta W \quad (18)$$

$$S_{\Delta W} = \beta_2 S_{\Delta W} + (1 - \beta_2) \Delta W^2 \quad (19)$$

β_1 and β_2 are hyperparameters which define the decay rate of the running averages. For our network, these are respectively set as 0.9 and 0.999 as per the literature. Both running averages are biased toward 0 during the first few epochs, so a bias-corrected version of both is calculated based on Eqs. (20) and (21).

$$V_{\Delta W}^{corrected} = \frac{V_{\Delta W}}{(1 - \beta_1^t)} \quad (20)$$

$$S_{\Delta W}^{corrected} = \frac{S_{\Delta W}}{(1 - \beta_2^t)} \quad (21)$$

With t defined as equal to the current epoch. Finally, the weight array W is updated as per Eq. (22).

$$W^{next} = W - \delta \frac{V_{\Delta W}^{corrected}}{\sqrt{S_{\Delta W}^{corrected} + \epsilon}} \quad (22)$$

ϵ is a constant set to 10^{-8} to prevent division by zero and δ is the learning rate, which was set to 0.0001.

Trainable parameters

The full network consists of 2.962.447 parameters (of which 2.959.881 are trainable) divided into an encoder and a decoder component, which are described in the next subsection.

Layers

The proposed autoencoder is comprised of several convolutional layers and can be divided into two parts: encoder and decoder. The input and output

of the network are 256x256 images which contain three channels: red, green, and blue, which are taken in by the input layer.

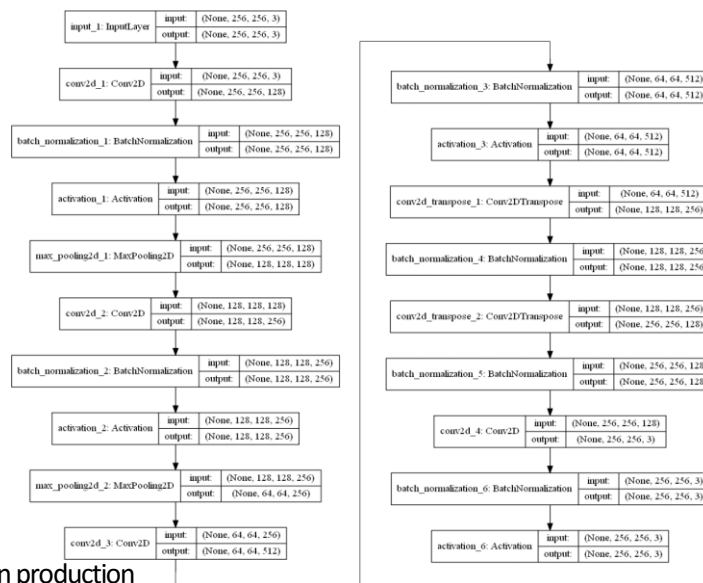
The encoder has an input layer made of 256x256x3 neurons. Then, there are two pairs of convolutional, batch normalization and max-pooling layers, and a last convolutional layer and batch normalization. These layers encode the data into a compressed 64x64x512 representation.

The decoder is comprised of two pairs of convolutional transpose and batch normalization layers, whose outputs are 256x256x128 representations of the reconstructed images. These are then converted into the final, restored image by a last pair of convolutional and batch normalization layers. Figure 7 depicts the proposed rain removal autoencoder architecture.

Implementation

The autoencoder model was implemented using the Keras deep learning library. We employed Google’s Cloud Computing Service for training with 100 training epochs, using the ADAM optimizer, measuring loss with MSE, and employing the full dataset. Prior to being fed to the network, all 11000 images were resized to 256x256x3 pixels to reduce computational costs. The rain images dataset was partitioned as in Table 1.

Figure 7. Proposed rain removal autoencoder architecture



Source: Own production

Results

We assessed the proposed architecture on the test dataset comprised by 1000 pairs of ground-truth and artificially degraded images.

We evaluated five algorithms: firstly, the traditional mean filter with a 5x5 kernel; secondly, the multi-guided filter proposed by [14]; thirdly, the dictionary learning approach by [7]; fourthly, a basic autoencoder architecture; and finally, our improved autoencoder architecture. Figure 8 depicts the results obtained with the latter. Figure 9 illustrates the increase in accuracy throughout the training process for several configurations of the autoencoder architecture, which are in turn described in Table 2.

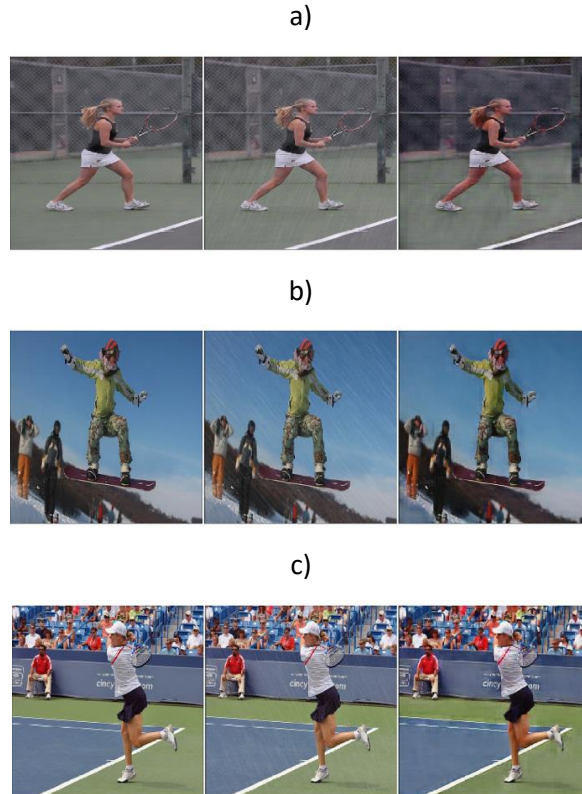
Table 3 depicts the performance of the baseline methods, a basic autoencoder architecture, and the proposed, improved autoencoder architecture. Our autoencoder-based approach improves significantly over traditional approaches based on filtering, which struggle with heavy rain conditions. Figure 10 shows the result of using our architecture on real rain-degraded images.

Table 2. Evaluated Autoencoder Configurations

| Autoencoder model | Description |
|-------------------|--|
| Model 1 | A single, 128 filters layer to join the encoder portion of the network with the decoder. |
| Model 2 | A single, 256 filters layer to join the encoder portion of the network with the decoder. |
| Model 3 | A single, 256 filters layer to join the encoder portion of the network with the decoder. Additional (up to 256) filters in the two layers next to the central layer. |
| Final Model | The model described in the previous section. Similar to model 3, but the central layer has 512 filters instead of 256 (see Figure 7). |

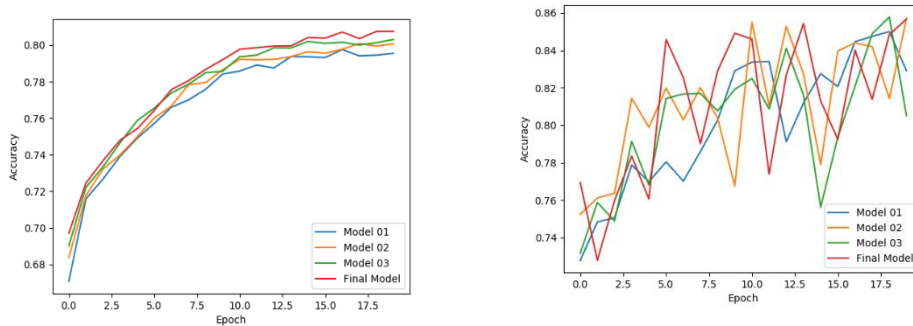
Source: Own production

Figure 8. Results obtained with the rain removal autoencoder architecture on seven images from the test portion of our dataset. The leftmost column (a) depicts the original images, the central column (b) depicts the degraded images, the rightmost column (c) depicts our results



Source: Own production

Figure 9. Training and validation accuracy for the 20 training epochs, for several autoencoder architectures with varying configurations



Source: Own production

Figure 10. Results obtained with our architecture on several real rain images. To the left, the original image; to the right, the de-rained image. Some of the images are somewhat darkened but this is proposed to be fixed by means of gamma correction



Source: Own production

Table 3. Evaluation of Method Performance

| Metric (test set) | Mean Filter | Multi-Guided Filter [14] | Dict. Learn [7] | Basic autoencoder | Proposed architect |
|-------------------|-------------|--------------------------|-----------------|-------------------|--------------------|
| MSE | 0.0107 | 0.0094 | 0.0086 | 0.0071 | 0.0061 |
| NRMSE | 0.2319 | 0.1884 | 0.2083 | 0.2175 | 0.1751 |
| PSNR | 20.0595 | 20.6408 | 21.1119 | 22.0431 | 22.5613 |
| SSIM | 0.5579 | 0.5861 | 0.7199 | 0.7312 | 0.8493 |

Source: Own production

Conclusions

After reviewing the literature, we observe there exist several approaches for removing rain from single images. Although deep learning methods have the best performance, they require sizable datasets and long training times. In contrast, dictionary learning methods can remove rain from a single image

without a training set, at the cost of additional processing time. Spatial/frequency filters result in the lowest processing costs and require no training sets, however, most such filters tend to fail when rain conditions are heavy.

In our work, we implemented a deep autoencoder architecture for removing rain from single images. Our architecture removes most rain from images while preserving detail, unlike most filter-based methods. However, colors are degraded by the network, possibly as a result of it trying to adjust for the increased brightness in images degraded by rain. This results in relatively low MSE, PSNR and SSIM scores when compared to other deep learning approaches in the literature. Regardless, our results show there is potential for employing autoencoders in rain removal tasks.

Another key point is the need to establish well-defined metrics to measure the performance of the proposed algorithms. When authors employ no metrics, there is no way to compare the results aside from subjective criteria. Although it may be hard to obtain pairs of ground-truth and degraded images our work shows it is possible to generate synthetic rain images with relative ease. We also show that results obtained by employing such images for training can be extrapolated to the removal of rain from real images.

Wrapping up, the field of single-image rain removal has seen considerable advances since the start of the 2010 decade. Future work should center on implementing more precise autoencoder architectures and carrying out in-depth analyses of the features they encode. Particularly, we are interested in combining frequency filters with autoencoder architectures in order to minimize detail loss and to improve our current results.

Bibliographic references

1. K. Park, S. Yu, and J. Jeong, "A contrast restoration method for effective single image rain removal algorithm," *2018 Int. Work. Adv. Image Technol. IWAIT 2018*, pp. 1–4, 2018. <https://doi.org/10.1109/IWAIT.2018.8369644>.
2. P. X. Minmin Shen, "A fast algorithm for rain detection and removal from videos," *Current*, pp. 1–6, 2011. <https://doi.org/10.1109/ICME.2011.6011963>.
3. Y. Sun, X. Duan, H. Zhang, and Y. Zhijing, "A Removal Algorithm of Rain and Snow from Images Based on Fuzzy Connectedness," in *International Conference on Computer Application and System Modeling*, , vol. 5, pp. 478–48 2010.

4. Y. Wang, S. Liu, C. Chen, and B. Zeng, "A Hierarchical Approach for Rain or Snow Removing in a Single Color Image," *IEEE Trans. Image Process.*, vol. 26, no. 8, pp. 3936–3950, 2017. <https://doi.org/10.1109/TIP.2017.2708502>.
5. H. Dong and X. Zhao, "Detection and removal of rain and snow from videos based on frame difference method," *Proc. 2015 27th Chinese Control Decis. Conf.*, pp. 5139–5143, 2015. <https://doi.org/10.1109/CCDC.2015.7162843>.
6. J. Xu, W. Zhao, P. Liu, and X. Tang, "Removing rain and snow in a single image using guided filter," in *2012 IEEE International Conference on Computer Science and Automation Engineering*, 2012, vol. 2, no. 2, pp. 304–307. <https://doi.org/10.1109/CSAE.2012.6272780>.
7. L. W. Kang, C. W. Lin, and Y. H. Fu, "Automatic single-image-based rain streaks removal via image decomposition," *IEEE Trans. Image Process.*, vol. 21, no. 4, pp. 1742–1755, 2012. <https://doi.org/10.1109/TIP.2011.2179057>.
8. 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 - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, 2017*, vol. 2017-Janua, pp. 1715–1723. <https://doi.org/10.1109/CVPR.2017.186>.
9. K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778. <https://doi.org/10.1109/CVPR.2016.90>.
10. L. Gondara, "Medical Image Denoising Using Convolutional Denoising Autoencoders," in *IEEE International Conference on Data Mining Workshops, ICDMW*, 2016. <https://doi.org/10.1109/ICDMW.2016.0041>.
11. H. Zhang, V. Sindagi, and V. M. Patel, "Image De-raining Using a Conditional Generative Adversarial Network," *IEEE Trans. Circuits Syst. Video Technol.*, vol. PP, no. c, pp. 1–1, 2019. <https://doi.org/10.1109/TCSVT.2019.2920407>.
12. S. H. Sun, S. P. Fan, and Y. C. F. Wang, "Exploiting image structural similarity for single image rain removal," in *IEEE International Conference on Image Processing*, pp. 4482–4486, 2014. <https://doi.org/10.1109/ICIP.2014.7025909>.
13. K. He, J. Sun, and X. Tang, "Guided Image Filtering," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, 2012. <https://doi.org/10.1109/TPAMI.2012.213>.

14. X. Zheng, Y. Liao, W. Guo, X. Fu, and X. Ding, "Single-image-based rain and snow removal using multi-guided filter," in *ICONIP: International Conference on Neural Information Processing*, vol. 8228 LNCS, no. PART 3, pp. 258–265, 2013. https://doi.org/10.1007/978-3-642-42051-1_33.
15. J. Xu, W. Zhao, P. Liu, and X. Tang, "An Improved Guidance Image Based Method to Remove Rain and Snow in a Single Image," *Comput. Inf. Sci.*, vol. 5, no. 3, pp. 49–55, 2012. <https://doi.org/10.5539/cis.v5n3p49>.
16. S. C. Pei, Y. T. Tsai, and C. Y. Lee, "Removing rain and snow in a single image using saturation and visibility features," in *2014 IEEE International Conference on Multimedia and Expo Workshops, ICMEW 2014*, pp. 2–7, 2014. <https://doi.org/10.1109/ICMEW.2014.6890551>.
17. Z. Zeng, Y. Li, and I. King, "Content-Adaptive Rain and Snow Removal Algorithms for Single Image," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, pp. 439–448, 2014. https://doi.org/10.1007/978-3-319-12436-0_49.
18. C. Liu, Y. Pang, J. Wang, A. Yang, and J. Pan, "Frequency Domain Directional Filtering Based Rain Streaks Removal from a Single Color Image," in *International Conference on Intelligent Computing*, pp. 415–424, 2014. https://doi.org/10.1007/978-3-319-09333-8_45.
19. Y. Li, R. T. Tan, X. Guo, J. Lu, S. Member, and M. S. Brown, "Rain Streak Removal Using Layer Priors," vol. 26, no. 8, pp. 3874–3885, 2017. <https://doi.org/10.1109/CVPR.2016.299>.
20. L. J. Deng, T. Z. Huang, X. Le Zhao, and T. X. Jiang, "A directional global sparse model for single image rain removal," *Appl. Math. Model.*, vol. 59, pp. 662–679, 2018. <https://doi.org/10.1016/j.apm.2018.03.001>.
21. S. Du, Y. Liu, M. Ye, Z. Xu, J. Li, and J. Liu, "Single image deraining via decorrelating the rain streaks and background scene in gradient domain," *Pattern Recognit.*, vol. 79, pp. 303–317, 2018. <https://doi.org/10.1016/j.patcog.2018.02.016>.
22. Y. Luo, Y. Xu, and H. Ji, "Removing rain from a single image via discriminative sparse coding," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015. <https://doi.org/10.1109/ICCV.2015.388>.
23. R. Mondal, P. Purkait, S. Santra, and B. Chanda, "Morphological Networks for Image De-raining," in *International Conference on*

- Discrete Geometry for Computer Imagery*, pp. 262–275, 2019. https://doi.org/10.1007/978-3-030-14085-4_21.
24. I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. The MIT Press, 2016. <https://doi.org/10.1007/s10710-017-9314-z>.
 25. Q. Zhang, L. T. Yang, Z. Chen, and P. Li, “A survey on deep learning for big data,” *Inf. Fusion*, vol. 42, no. October 2017, pp. 146–157, 2018. <https://doi.org/10.1016/j.inffus.2017.10.006>.
 26. K. Iqbal, M. Odetayo, and A. James, “Enhancing the low quality images using Unsupervised Colour Correction Method,” 2010, pp. 1703–1709. <https://doi.org/10.1109/ICSMC.2010.5642311>.
 27. Adobe, “PDF Blend Modes : Addendum,” 2006.
 28. A. E. Dirik and N. Nemon, “Image tamper detection based on demosaicing artifacts,” *Comput. Eng.*, pp. 1497–1500, 2009. <https://doi.org/10.1109/ICIP.2009.5414611>.
 29. P. Arbeláez, M. Maire, C. Fowlkes, and J. Malik, “Contour detection and hierarchical image segmentation,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 5, pp. 898–916, 2011. <https://doi.org/10.1109/TPAMI.2010.161>.
 30. G. Schaefer and M. Stich, “UCID: an uncompressed color image database,” in *Proceedings of SPIE 5307*, 2003. <https://doi.org/10.1117/12.525375>.
 31. X. Fu, J. Huang, X. Ding, Y. Liao, and J. Paisley, “Clearing the skies: A deep network architecture for single-image rain removal,” *IEEE Trans. Image Process.*, vol. 26, no. 6, pp. 2944–2956, 2017. <https://doi.org/10.1109/TIP.2017.2691802>.
 32. X. Glorot and Y. Bengio, “Understanding the difficulty of training deep feedforward neural networks,” *Proc. Thirteen. Int. Conf. Artif. Intell. Stat.*, vol. 9, pp. 249–256, 2010. <https://doi.org/10.1.1.207.2059>
 33. S. Khirirat, H. R. Feyzmahdavian, and M. Johansson, “Mini-batch gradient descent: Faster convergence under data sparsity,” *2017 IEEE 56th Annu. Conf. Decis. Control. CDC 2017*, vol. 2018-January, no. Cdc, pp. 2880–2887, 2018. <https://doi.org/10.1109/CDC.2017.8264077>.
 34. K. Yuan, B. Ying, J. Liu, and A. H. Sayed, “Variance-Reduced Stochastic Learning by Networked Agents under Random Reshuffling,” *IEEE Trans. Signal Process.*, vol. 67, no. 2, pp. 351–366, 2019. <https://doi.org/10.1109/TSP.2018.2872003>.

35. D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," in *International Conference on Learning Representations*, 2015.