MONOCHROMATOIC IMAGE DEHAZING USING ENHANCED FEATURE EXTRACTION TECHNIQUES IN DEEP LEARNING

A PROJECT REPORT

Submitted by

NISARG DOSHI [Reg No: RA1911042010106]
SAGAR BHAVSAR [Reg No: RA1911042010111]

Under the Guidance of

Dr. D. RAJESWARI
(Associate Professor, Department of Data Science and Business Systems)

In partial fulfillment of the Requirements for the Degree
Of

BACHELOR OF TECHNOLOGY
COMPUTER SCIENCE AND BUSINESS SYSTEMS

DEPARTMENT OF DATA SCIENCE AND BUSINESS SYSTEMS

FACULTY OF ENGINEERING AND TECHNOLOGY

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

NOVEMBER 2022

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Under Section3 of UGC Act,1956)

BONAFIDE CERTIFICATE

Certified that this project report titled **"MONOCHROMATOIC IMAGE DEHAZING USING ENHANCED FEATURE EXTRACTION TECHNIQUES IN DEEP LEARNING"** is the bonafide work of **"NISARG DOSHI [Reg No: RA1911042010106] and SAGAR BHAVSAR [Reg No: RA1911042010111]** who carried out the project work under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or award was conferred on an earlier occasion for this or any other candidate.

Dr. D. Rajeswari	Dr. M.Lakshmi
GUIDE	**HEAD OF THE DEPARTMENT**
Associate Professor	Dept. of DSBS
Dept. of DSBS	

Signature of Internal Examiner

Signature of External Examiner

ABSTRACT

Images photographed in foggy weather usually have poor visibility. To mitigate this problem researchers have come up with various image dehazing techniques. Now, more than ever, high-quality images that can be used to glean maximum information from autonomous systems are in high demand. This study uses different convolutional neural network (CNN) architectures to draw out essential details from the picture and localize the information recovered to reduce the haze from the picture. Three pre-processing techniques such as Airlight estimation, Contextual regularization and Boundary constraint is used in this work. Various combinations of experiments are done with three pre-processing techniques and four CNN architectures. Experimental results shows that VGG16 achieves 28.35 PSNR and DenseNet achieves 0.825 SSIM.

ACKNOWLEDGEMENT

We express our humble gratitude to **Dr C. Muthamizhchelvan**, Vice-Chancellor, SRM Institute of Science and Technology, for the facilities extended for the project work and his continued support.

We extend our sincere thanks to Dean-CET, SRM Institute of Science and Technology, **Dr T.V.Gopal**, for his invaluable support.

We wish to thank **Dr Revathi Venkataraman**, Professor & Chairperson, School of Computing, SRM Institute of Science and Technology, for her support throughout the project work.

We are incredibly grateful to our Head of the Department**, Dr M. Lakshmi** Professor, Department of Data Science and Business Systems, SRM Institute of Science and Technology, for her suggestions and encouragement at all the stages of the project work.

We want to convey our thanks to our program coordinators **Dr.E.Sasikala,** Professor, Department of Data Science and Business Systems, SRM Institute of Science and Technology, for her inputs during the project reviews and support.

We register our immeasurable thanks to our Faculty Advisor**, Dr. D. Rajeswari,** Associate professor, Department of data science and business systems, SRM Institute of Science and Technology, for leading and helping us to complete our course.

Our inexpressible respect and thanks to my guide, **Dr. D. Rajeswari,** Associate professor, Department of data science and business systems and co-guide, **Dr. R. Srinivasan,** Assistant Professor, Department of computing technologies, SRM Institute of Science and Technology, for providing me with an opportunity to pursue my project under their mentorship. They provided me with the freedom and support to explore the research topics of my interest. Their passion for solving problems and making a difference in the world has always been inspiring.

We sincerely thank the Data Science and Business Systems staff and students, SRM Institute of Science and Technology, for their help during our project. Finally, we would like to thank parents, family members, and friends for their unconditional love, constant support, and encouragement.

Nisarg Doshi

Sagar Bhavsar

TABLE OF CONTENTS

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

AI	Artificial Intelligence
ML	Machine Learning
CNN	Convolutional Neural Network
IDE	Integrated Development Environment
GPU	Graphics Processing Unit
RTX	Ray Tracing Texel eXtreme
AOD	All-In-One
GAN	Generative Adversarial Networks
RGB	Red Green Blue
PSNR	Peak Signal-to-Noise Ratio
RMSE	Root mean Squared Error
SSIM	Structural Similarity Index Measure
DCP	Dark Channel Prior
CPS	Cyber-Physical System
GUI	Graphical User Interface

LIST OF SYMBOLS

^	Conjunction
/	Division
*	Multiplication
-	Subtraction
+	Addition

CHAPTER 1

INTRODUCTION

1.1 GENERAL

With the rise of digital cameras both in the consumer market and in various sensing systems, the haze-removal of outdoor images is gaining increasing attention. Image dehazing has taken by storm numerous significant scientific fields of applications such as astronomy, medical sciences, remote sensing, surveillance, web mapping, land-use planning, agronomy, archaeology, and environmental studies. Visual data is the most crucial data comprehended and analyzed by the human brain. About one-third of the cortical area in the human brain is dedicated to analysis of visual data. As a result, image clarity is of uttermost importance for numerous imaging tasks.

Small particles suspending in the atmosphere (e.g., droplets and dusts) often scatter the light. As a consequence, the clarity of an image would be seriously degraded, which may decrease the performance of many multimedia processing systems, e.g., content-based image retrieval. Image enhancement methods developed using image processing technique to recover image details can only alleviate this problem slightly. Further, the rapid development of technologies such as artificial intelligence, machine learning and computer vision has led to renewed research into improved image dehazing techniques.

1.2 MOTIVATION

An image is a visual representation of something, which can also be viewed as a bundle of graphical data that can be extracted. Data is extracted from images in all kinds of sectors. Military and law enforcement personnel can use image data to potentially prevent military attacks, they can also be used to decipher patterns. It can be feed to AI, ML machines to train them to be more efficient. For example the ML algorithm of an autonomously driven car is fed all kinds of sign boards for the car to adjust accordingly. Similarly there are critical use cases where it is necessary for the image to be clear.

The potential for computer vision is vast, with that being said, there is still a lot of work to be done in this segment especially on the most critical part of making image unambiguous by dehazing it.

1.3 PURPOSE

A photograph taken in foggy conditions results in an image with low visibility. As depicted in Figure 1, the haze causes distant objects to lose contrast and blend in with their surroundings. The light reflected by these items is dimmed and diluted by the environment, and it interacts with light dispersed by various particles in the air before reaching the camera. Consequently, as these objects recede further from the camera, their colors fade and become more reminiscent of fog, with the degree of resemblance increasing with distance.

Dehazing is difficult because the intensity of the haze is reliant on the depth that is generally not known. Having only one fuzzy image to work with also makes the task more challenging because of the lack of input variety. Due to their reliance on either a large number of hazy photos as input or extra previous knowledge [1], most current dehazing technologies provide a reliance that is frequently unworkable in a wide range of real-world scenarios.

1.4 OBJECTIVE

When utilized to single-image dehazing with more rigorous priors or assumptions, the atmospheric scattering model, which describes how an image originates when haze is present, has demonstrated significant progress. According to this theory, a fuzzy image is one where S is the discerned blurry image and U is the ambient light, y is the transmission, representing the fraction of the luminance which is not dispersed and reaches the camera, and A is the global atmospheric light. Previous haze removal methods, based on this image degradation model, typically relied on either more depth information or multiple observations of the same scene.

$$S(x) = U(x)y(x) + Z(1 - y(x))$$

Samples of representative works are [2], [3], [4] and [5]. [3] observe how the airlight is partially polarized due to the atmospheric particles scattering some of it. From this result, a fast method for haze reduction using two pictures taken through a polarizer at different angles is developed. [6] suggests a scattering model based on physics [2], [5]. This model can recover the scene structure from two or more weather photos. [6] propose a method for dehazing images using scene depth information derived from georeferenced digital terrain or city models.

The proposal of [7] for the use of pre-processing techniques used to enhance important details in the hazy image can effectively make any model more efficient and effective is particularly interesting. We used the pre-processing techniques along with five traditional CNN models, namely, AlexNet, DenseNet, VGG16, ResNet is used by us to compare and contrast their effectiveness to get better results.

Figure 1.1: The image above shows hazy image(top left) and dehazed image (top right), boundary constrain output (bottom left) and contextual regularization output (bottom right).

1.5 SOFTWARE REQUIREMENT SPECIFICATIONS

- Desktop/laptop with minimum specification having i5, 8GB RAM
- Operating System: Windows 7/8/8.1/10/11, MacOs
- IDE: Google Colab Pro
- GPU: RTX 5000 (provided by Google Colab Pro)

CHAPTER 2

LITERATURE STUDY

2.1 WORKS RELATED TO SINGLE IMAGE DEHAZING

Dehazing a single image is a more difficult problem because less scene structure information is accessible. Additionally, tremendous progress has been made in recent years [8] - [13]. Research into novel image models and priors provides valuable insight that aids in these developments. Fattal [8] propose a new model of image generation that incorporates surface shading and scene transmission. To calculate the scene transmission from a hazy image, it is necessary to assume that the two functions are statistically uncorrelated locally and then partition the image into sections with constant albedo. Tan [9] suggests increasing the local contrast of a fuzzy picture to make it more legible. Particularly in areas with exceptionally dense hazes, this approach can yield rather convincing results. However, the restored image often has significant haloing and distorted colors.

2.2 SINGLE IMAGE HAZE REMOVAL USING DARK CHANNEL PRIOR

The fundamental finding of [10] is that maximal local patches in haze-free outdoor photographs contain almost no pixels with very little intensity in at least one color channel, leading them to suggest a dark channel prior for single picture dehazing. When combined with a gentle mating procedure, the preceding can generate a convincing, haze-free, high-quality result. The model of a picture as a factorial Markov random field proposed by [12] treats scene albedo and depth as two statistically independent latent layers. A common expectation maximization approach is utilized to do the image factorization. [12]'s method is capable of restoring a clear picture with crisp edges, although the end result is often too saturated.

2.3 WORK RELATED TO DEHAZENET, SINGLE IMAGE DEHAZING VIA MULTI-SCALE CONVOLUTION AND AOD- NET

To reconstruct *U* estimating *y* and *Z* from *S* has been proposed using a number of different deep learning-based methods. *A* novel BReLU unit for estimating *y* was introduced by [14]. Similarly, a multi-scale deep neural network was recently proposed to estimate *y* by Ren [15]. All of these approaches have the drawback of considering t exclusively in their CNN frameworks, and thus missing out on other crucial pieces of data. To solve this issue, Li *et al.* [16] suggested AOD-Net, which uses a linear transpose to combine *y* and *Z* into a single parameter.

2.4 WORK ON IMAGE DEHAZING USING GENERATIVE ADVERSARIAL NETWORKS

The successful work of [17]'s GAN in creating natural-looking pictures has sparked substantial academic research into their potential. [18] provide a combined discriminator based on GAN to determine if the matched dehazed image and the estimated transmission map are authentic in order to better include the mutual structural information between the estimated t and the dehazed outcome. [19] demonstrate how to do multi-scale image dehazing using a deep perceptual pyramid network, contemporary dense blocks, and residual blocks. With this technique, scene context is handled throughout the decoding process, due to an encoder-decoder design with a pyramid pooling module. [20] propose using a bi-directional consistency loss to estimate *y* and *Z* in order to reconstruct *J* with a fully CNN and some fine-tuned adjustments. [21] explicitly describe the relationship between *y*, *Z*, and *U* using a bilinear CNN, and then estimate *y* and *Z* using the recommended bilinear composition loss function.

CHAPTER 3

PROPESED METHODOLOGY

3.1 DATASET DESCRIPTION

For model training and evaluation, we utilized the “REalistic Single Image DEhazing” (RESIDE) dataset [22]. This huge benchmark comprises both artificial and real-world hazy pictures that may be used to test all known single-image dehazing approaches. An Outdoor Training Set is used for this project (OTS). A collection of blurry images captured in the real world that may be used to test and compare single-image dehazing techniques. The dataset is split into five subsets, each of which can be used for a different kind of training or evaluation, showcasing the variety of both the data sources and the image contents used. Both a "train" and "test" set of data were created. For the RESIDE OTS dataset, 80% of the images came from the train dataset, while the remaining 20% came from the test dataset. The dataset included about 13000 images, 350 of which were unique.

The dataset can be classified into two categories:

- Clear Images
- Hazy images

3.2 PRE PROCESSING TECHNIQUES:

Images are scaled, rearranged, and turned to arrays as part of the first processing of the dataset. Along with it the dataset also undergoes two other highly level pre-preocessing techniques which are specifically intended to be implemented to enhance the detail components necessary for effective image dehazing thus increasing the efficiency of the model.

3.2.1 BOUNDARY CONSTRAINT

This method is used to develop the fog and dust image model and to estimate the transmittance function roughly. The optimum transmittance function, which is computed, is then used to restore the low illumination dust and haze image using the nonlinear context regularization approach based on logarithmic transformation. To calculate the multiple of the logarithmic transformation, the image's maximum luminance value is used. Figure 3.1 shows a hazy image with its corresponding image which is pre-processed using boundary constraint from radiance cube.

Figure 3.1: The boundaries of the radiance cube. (Left) Blurry image. (Right) Pre-processed version made using the boundary constraint technique.

3.2.2 CONTEXTUAL REGULARIZATION

Using the boundary constraint method, the haze and dust model is created by roughly estimating the transmittance function. The optimized transmittance function, which is calculated using the nonlinear context regularization method based on logarithmic transformation, is then used to restore the low illumination fog and dust image. The multiple of the logarithmic transformation is calculated using the image's maximum luminance value.

In addition to the contextual regularization, a series of high-order filters is employed. In order to keep the image's contours intact, it uses a Laplacian operator and seven Kirsch operators. In Figure 3.2, we see a foggy image and its corresponding, contextual regularization-preprocessed counterpart.

Figure 3.2: Perspective on Contextual Regularization using Weighted L1-norm. (left) original blurry image, (right) pre-processed version using Weighted L1-norm based Contextual Regularization.

3.3 PROPOSED ARCHITECTURE

Figure 3.3 illustrates the method flow for the proposed method. The proposed system is designed as follows: A hazy image is taken as input. The image is not optimum to be directly given to the neural network for training, thus a set of preprocessing techniques are applied on the top of the hazy images following which the pre-processed image is feed to the model for evaluation, effectively providing us with the dehazed image.

In the pre-processing module, essential features are extracted from the image such as amount of air light in the atmosphere using air light estimation, the edges and depth of the content in the image using contextual regularization and boundary constraint from radiance cube. With the help of feature extraction, the information contained in an unprocessed data set can be converted into quantifiable features for further analysis.

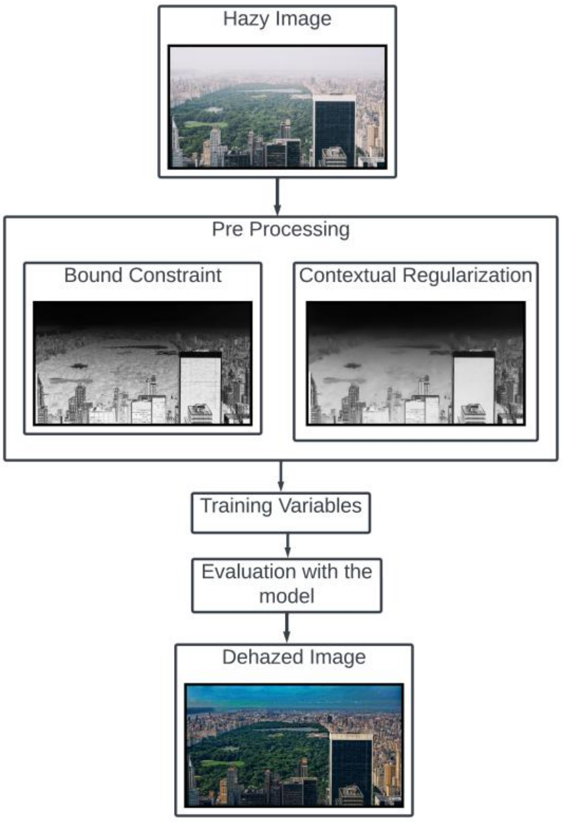


Figure 3.3: Architecture Diagram

3.4 CONVOLUTIONAL NEURAL NETWORK (CNN)

CNNs' main benefits lie in their ability to provide a dense network that efficiently makes predictions, identifies objects, etc. Multiple layers of a CNN (or ConvNet) can be trained to recognize various aspects of an image. Each image has a filter or kernel applied to it, and the results improve and become more detailed as you move through the layers. The filters can begin with basic features at the lower levels [23-24].

The filters check and identify features that uniquely represent the input object at each layer by increasing in complexity. As a result, the partially recognized image produced by one layer of convolving is fed into the next. In the final layer, an FC layer, the CNN makes the identification of the image or object it represents.

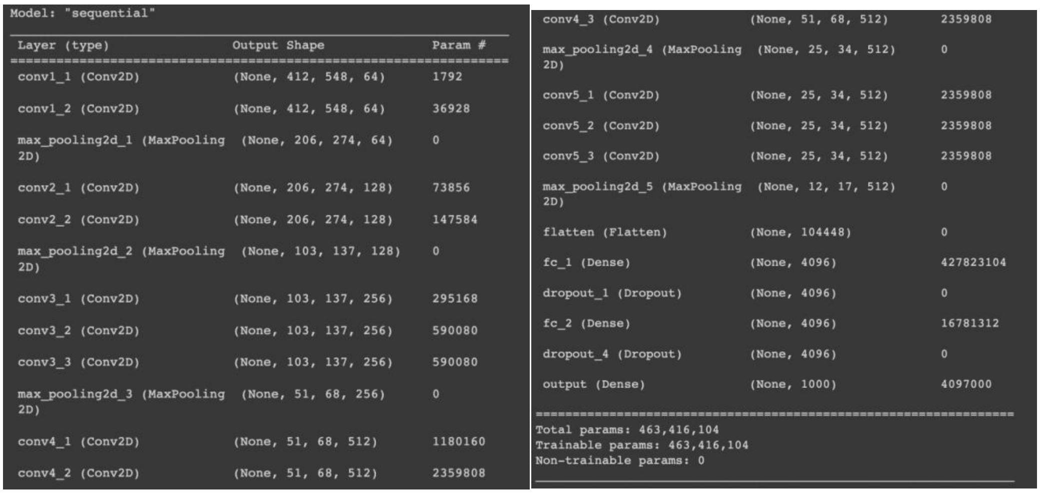
Using CNNs for deep learning is popular due to three important factors:

- CNNs eliminate the need for manual feature extraction—the features are learned directly by the CNN.
- CNNs produce highly accurate recognition results.

- CNNs can be retrained for new recognition tasks, enabling you to build on pre-existing networks.

3.4.1 VGG16

- The 16 in VGG16 indicates that there are 16 weighted layers. There are a total of 21 layers in VGG16 (13 convolutional, 5 Max Pooling, 3 Dense), but only 16 weight layers (the learnable parameters layer) [25].
- Tensor sizes of 224 and 244 with three RGB channels are accepted by VGG16 as input.
- VGG16 uses using 3x3 filter convolution layers with stride 1 and the same padding and maxpool layer of 2x2 filter with stride 2.
- The convolution and max pool layers are evenly distributed. Conv-1 has 64 available filters, Conv-2 has 128, Conv-3 has 256, and Convs 4 and 5 have 512 available filters.
- Following the convolutional layers are three Fully-Connected layers; the first two have 4096 channels each, while the third has 1000 channels and performs 1000-way ILSVRC classification. After all additional layers have been added, what is left is the soft-max layer.



Model: "sequential"

Layer (type)	Output Shape	Param #
conv1_1 (Conv2D)	(None, 412, 548, 64)	1792
conv1_2 (Conv2D)	(None, 412, 548, 64)	36928
max_pooling2d_1 (MaxPooling2D)	(None, 206, 274, 64)	0
conv2_1 (Conv2D)	(None, 206, 274, 128)	73856
conv2_2 (Conv2D)	(None, 206, 274, 128)	147584
max_pooling2d_2 (MaxPooling2D)	(None, 103, 137, 128)	0
conv3_1 (Conv2D)	(None, 103, 137, 256)	295168
conv3_2 (Conv2D)	(None, 103, 137, 256)	590080
conv3_3 (Conv2D)	(None, 103, 137, 256)	590080
max_pooling2d_3 (MaxPooling2D)	(None, 51, 68, 256)	0
conv4_1 (Conv2D)	(None, 51, 68, 512)	1180160
conv4_2 (Conv2D)	(None, 51, 68, 512)	2359808
conv4_3 (Conv2D)	(None, 51, 68, 512)	2359808
max_pooling2d_4 (MaxPooling2D)	(None, 25, 34, 512)	0
conv5_1 (Conv2D)	(None, 25, 34, 512)	2359808
conv5_2 (Conv2D)	(None, 25, 34, 512)	2359808
conv5_3 (Conv2D)	(None, 25, 34, 512)	2359808
max_pooling2d_5 (MaxPooling2D)	(None, 12, 17, 512)	0
flatten (Flatten)	(None, 104448)	0
fc_1 (Dense)	(None, 4096)	427823104
dropout_1 (Dropout)	(None, 4096)	0
fc_2 (Dense)	(None, 4096)	16781312
dropout_4 (Dropout)	(None, 4096)	0
output (Dense)	(None, 1000)	4097000

Total params: 463,416,104
Trainable params: 463,416,104
Non-trainable params: 0

Figure 3.4: Outline of VGG16Model

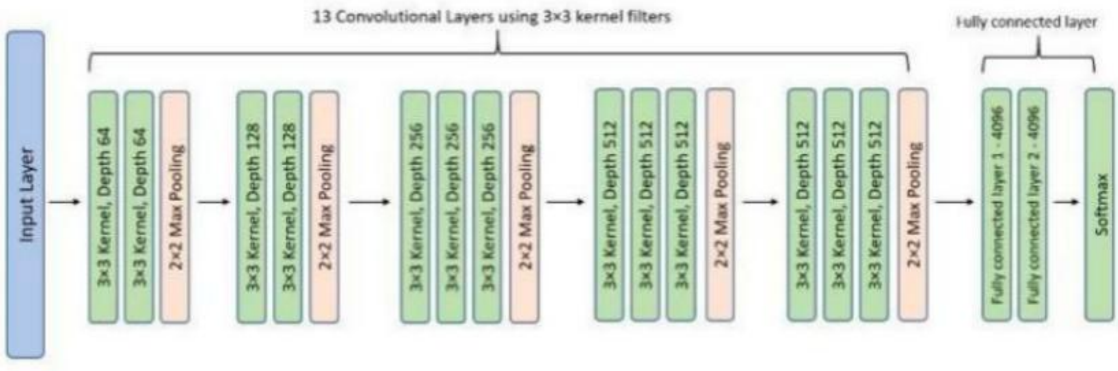


Figure 3.5: Architecture of VGG16

3.4.2 ALEXNET

- AlexNet is a deep neural network that uses deep learning. The first convolutional network to take advantage of GPU acceleration.
- Maximum pooling is accomplished by means of the pooling layers.
- Since all of the layers are connected, the size of the input is set.
- The input size is typically given as 224 by 224 by 3, but in practice it is 227 by 227 by 3. This is likely due to padding during the input process.
- Sixty million parameters make up AlexNet's total size.

Layer	# filters / neurons	Filter size	Stride	Padding	Size of feature map	Activation function
Input	-	-	-	-	227 x 227 x 3	-
Conv 1	96	11 x 11	4	-	55 x 55 x 96	ReLU
Max Pool 1	-	3 x 3	2	-	27 x 27 x 96	-
Conv 2	256	5 x 5	1	2	27 x 27 x 256	ReLU
Max Pool 2	-	3 x 3	2	-	13 x 13 x 256	-
Conv 3	384	3 x 3	1	1	13 x 13 x 384	ReLU
Conv 4	384	3 x 3	1	1	13 x 13 x 384	ReLU
Conv 5	256	3 x 3	1	1	13 x 13 x 256	ReLU
Max Pool 3	-	3 x 3	2	-	6 x 6 x 256	-
Dropout 1	rate = 0.5	-	-	-	6 x 6 x 256	-

Figure 3.6: Outline of AlexNet

3.4.3 DENSENET

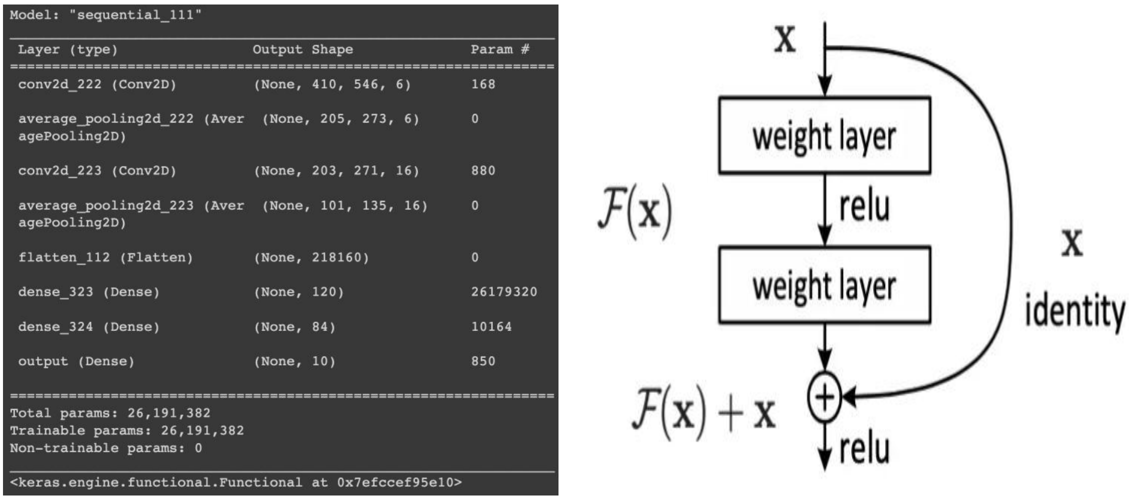
A DenseNet is a CNN with dense connections between layers, as opposed to Dense Blocks, which link all layers directly. The feature maps of each layer are passed on as inputs to the layers below it, while the feature maps of the layers above it are ignored. DenseNet-121 has the following layers:

1. One 7x7 Convolution
2. Five 8 3x3 Convolution
3. Sixtyone - 1x1 Convolution
4. Four- AvgPool
5. One - Fully Connected Layer

3.4.3 RESNET

The ResNet architecture relies heavily on residual blocks. Convolutional layers, batch normalization layers, and nonlinear activation layers like ReLu are stacked in older architectures like VGG16. This technique can function with a relatively low number of convolutional layers—around 19 layers is the upper limit for VGG models. However, follow-up studies revealed that increasing the number of layers can significantly enhance CNN performance.

The ResNet architecture introduces the straightforward idea of connecting the final output of a chain of convolution blocks with an additional intermediate input. See an example of this down below.



```
Model: "sequential_111"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_222 (Conv2D)         (None, 410, 546, 6)       168

 average_pooling2d_222 (Aver  (None, 205, 273, 6)      0
 agePooling2D)

 conv2d_223 (Conv2D)         (None, 203, 271, 16)      880

 average_pooling2d_223 (Aver  (None, 101, 135, 16)     0
 agePooling2D)

 flatten_112 (Flatten)       (None, 218160)            0

 dense_323 (Dense)           (None, 120)               26179320

 dense_324 (Dense)           (None, 84)                10164

 output (Dense)              (None, 10)                850

=================================================================
Total params: 26,191,382
Trainable params: 26,191,382
Non-trainable params: 0
_________________________________________________________________
<keras.engine.functional.Functional at 0x7efccef95e10>
```


Figure 3.7: (left) Outline of ResNet (right) Residual Learning architecture

CHAPTER 4

MODEL ESTIMATION

4.1 PEAK SIGNAL-TO-NOISE RATIO (PSNR)

The PSNR is the ratio of the strongest possible signal to the strongest possible noise that impairs the representation of a picture. The PSNR of an image can be calculated by comparing it to a theoretically perfect, noise-free version of the same image. It is defined as follows:

$$PSNR = 10log_{10}(\frac{(\mathrm{M}-1)^2}{MSE}) = 20log_{10}(\frac{\mathrm{M}-1}{RMSE}) \qquad (2)$$

Figure 4.1 : Formula for PSNR

Where, M represents the highest possible intensity levels in an image. MSE is the mean squared error and RMSE is the abbreviation of Root Mean Squared Error.

4.2 STRUCTURAL SIMILARITY INDEX MEASURE (SSIM)

Digital images and videos of all kinds can have their perceived quality predicted using a technique called the SSIM, which is also used to measure structural similarity between two objects. Similarity between two images can be evaluated with the help of SSIM. To measure or predict image quality using the SSIM index, an original uncompressed or distortion-free image must be used as a baseline.

$$\mathrm{SSIM}(x, y) = \frac{(2UxUy + K1)(2Sxy + K2)}{(U^2x + U^2y + K1)(S^2x + S^2y + K2)} \quad (3)$$

Figure 4.2 : Formula for SSIM

Where:

Ux = the pixel sample mean of x;

Uy = the pixel sample mean of y;

S^2x = variance of x;

S^2y = variance of y;

K1, K2 = variables used to make the division stable.

CHAPTER 5

EXPERIMENTAL RESULTS

The input image from this study will be hazy images from RESIDE OTS (Outdoor training set). The model will dehaze the image by recognizing noise or haze from the essential details. The dehazed image obtained will be much more usable and any analysis conducted over it will be much more accurate. The study employs a total of One Thousand Three hundred images and around 350 unique images. The size of the photographs used were 550 Pixel horizontally and 213 Pixel vertically. Deep learning algorithms are carried out in Google Colab Pro.

Figure 8 illustrates the full progression of a picture through the model's iterative processes. The figure depicts the hazzy image being sent to the boundary constraint pre-processing module, where the output, depicted in column 2, is obtained; the image is then sent to the following pre-processing module, where contextual regularization occurs, the Kirsch Filter is applied, and the morphological transformation of the image is performed. In the third column, we see the final product after various pre-processing techniques. The next step is to feed the image into the neural network. Images produced by various CNN architectures are displayed in columns 4–7.The last column shows the how the image looks in actuality. The output from the modules should be substantially better or at least more in line with the real world situation.

In Table I, we can see the SSIM and PSNR for a number of different CNN model architectures. It is possible to gauge our model's efficacy with measurements like the SSIM and PSNR. The more accurate the dehazed image is, the higher the value of both metrics should be.

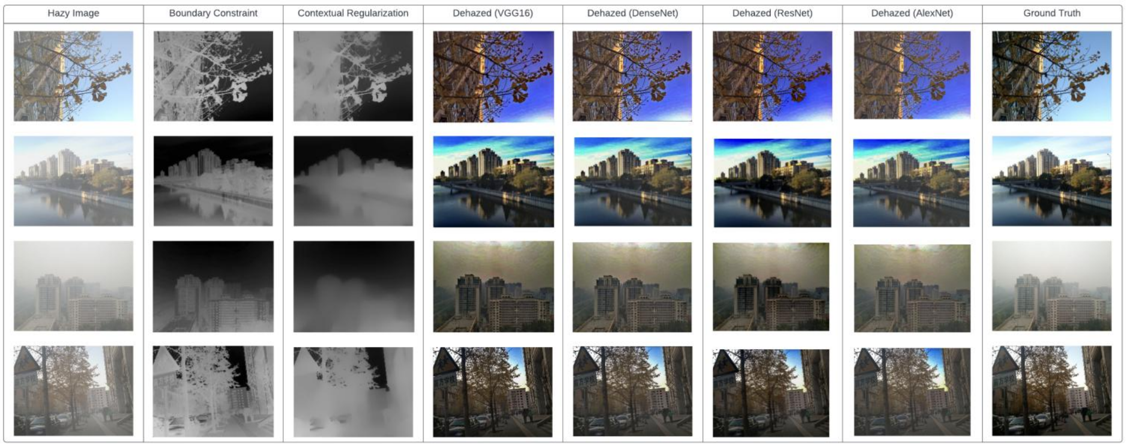


Figure 5.1 : Complete life cycle of the image in the model

The database used to calculate the metrics values in Table 5.2 is the same one used to calculate the values in Table 5.1. (SOTS). A comparison of the performance of various existing methods is performed. The methods include: DCP [26], DehazeNet [14], MSCNN [15], AOD-Net [16].

TABLE 5.1: PERFORMANCE COMPARISON ON THE OUTFOOR SOTS ON VARIOUS CNN ARCHITECTURES USED IN THIS STUDY

Deep Learning Method Used	***Average SSIM Value***	***Average PSNR Value***
VGG16	0.813	28.35
Alexnet	0.791	28.20
DenseNet	0.825	28.03
ResNet .	0.785	27.90

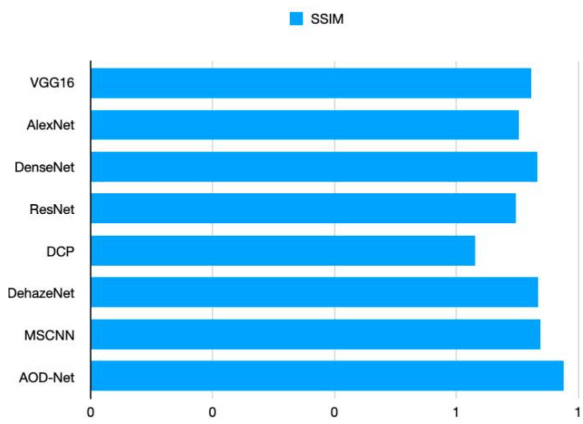
Table 5.2: Performance Comparison of Outdoor SOTS Using Preexisting Methods

	DCP	***DehazeNet***	***MSCNN***	***AOD-Net***
PSNR (dB)	18.54	26.84	21.73	24.08
SSIM	0.710	0.826	0.831	0.873

CHAPTER 6

CONCLUSION AND FUTURE WORK

In this study, multiple pre-processing strategies were employed to optimize the performance of the various CNN architectures tested. The results of the experiments demonstrated their potential for use in obtaining better results and the ability to feed more information to the neural network, thereby improving the model's accuracy. The average SSIM values show only moderate improvement when compared to the existing benchmark methods indicated in Table II, while the mean PSNR values of the various methods ranged from 27.90 in ResNet to 28.36 in VGG16, demonstrating significant strength. In future the technique also can be used with different set of images, in different environments like indoor images. Improved outcomes also can be achieved in the future by combining alternative pre-processing methods with other pre-existing models. The technique also can be used with different set of images, in different environments like indoor images.



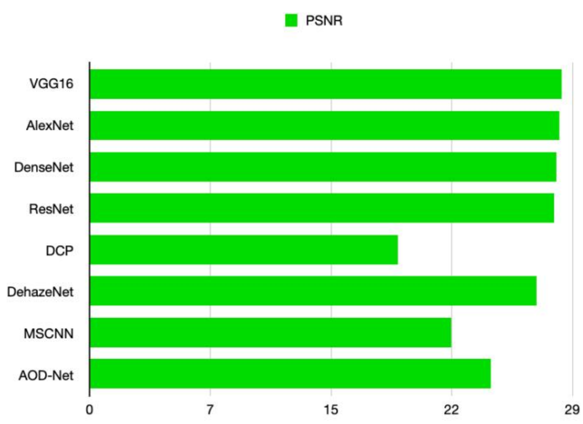


Figure 5.2 : (Top) Comparison of various SSIM value (Bottom) Comparison of various PSNR values.

CHAPTER 7

REFERENCES

[1] Y. Li, S. You, M. S. Brown, and R. T. Tan, “Haze visibility enhancement: A survey and quantitative benchmarking,” Comput. Vis. Image Understanding, vol. 165, pp. 1–16, Dec. 2017.

[2] 2013 IEEE International Conference on Computer Vision, 2013, pp. 617-624, doi: 10.1109/ICCV.2013.82.

[3] S. G. Narasimhan and S. K. Nayar. Vision and the atmosphere. IJCV, 48(3):233–254, 2002.

[4] Y. Y. Schechner, S. G. Narasimhan, and S. K. Nayar. Instant dehazing of images using polarization. In CVPR’01, volume 1, pages 325–332, 2001.

[5] S. Shwartz, E. Namer, and Y. Y. Schechner. Blind haze separation. In CVPR’06, volume 2, pages 1984–1991, 2006

[6] S. G. Narasimhan and S. K. Nayar. Contrast restoration of weather degraded images. IEEE TPAMI, 25(6):713–724, 2003.

[7] J. Kopf, B. Neubert, B. Chen, M. Cohen, D. Cohen-Or, O. Deussen, M. Uyttendaele, and D. Lischinski. Deep photo: model-based photograph enhancement and viewing. In ACM SIGGRAPH Asia 2008, pages 116:1–116:10, 2008.

[8] R. Fattal. Single image dehazing. In ACM SIGGRAPH 2008, pages 72:1–72:9, 2008.

[9] R. T. Tan. Visibility in bad weather from a single image. In CVPR’08, pages 1–8, 2008.

[10] K. He, J. Sun, and X. Tang. Single image haze removal using dark channel prior. In CVPR’09, pages 1956–1963, 2009.

[11] J. P. Tarel and N. Hautiere. Fast visibility restoration from a single color or gray level image. In ICCV’09, pages 2201– 2208, Oct. 2009.

[12] L. Kratz and K. Nishino. Factorizing scene albedo and depth from a single foggy image. In ICCV’09, pages 1701–1708, Oct. 2009.

[13] B. G. Kristofor, T. V. Dung, and Q. N. Truong. An investigation of dehazing effects on image and video coding. IEEE TIP, 21(2):662–673, 2012.

[14] B. Cai et al., “Dehazenet: An end-to-end system for single image haze removal,” IEEE Trans. on Image Proc., vol. 25, no. 11, pp. 5187–5198, 2016.

[15] W. Ren et al., “Single image dehazing via multi-scale convolutional neural networks,” in Proc. IEEE European Conf. on Comp. Vision. Springer, 2016, pp. 154–169.

[16] B. Li et al., “Aod-net: All-in-one dehazing network,” in Proc. IEEE Conf. on Comp. Vis., 2017

[17] I. Goodfellow et al., “Generative adversarial nets,” in Proc. Advances in Neural Information Proc. Systems, 2014, pp. 2672–2680.

[18] H. Zhang, V. Sindagi, and V. M. Patel, “Multi-scale single image dehazing using perceptual pyramid deep network,”in Proc. IEEE Conf. Workshop on Comp. Vis. Patt. Recog., 2018, pp. 902–911.

[19] H. Zhang and V. M. Patel, “Densely connected pyramid dehazing network,” in Proc. IEEE Conf. on Comp. Vis. Patt. Recog., 2018, pp. 3194–3203.

[20] R. Mondal, S. Santra, and B. Chanda, “Image dehazing by joint estimation of transmittance and airlight using bidirectional consistency loss minimized fcn,” in Proc. IEEE Conf. Workshop on Comp. Vis. Patt. Recog., 2018, pp. 920– 928.

[21] H. Yang et al., “Image dehazing using bilinear composition loss function,” in arXiv preprint arXiv:1710.00279, 2017.

[22] B. Li et al., “Benchmarking single-image dehazing and beyond,” IEEE Trans. Image Process., vol. 28, no. 1, pp. 492–505, Jan. 2019.

[23] M. Yadav, R. Goel and D. Rajeswari, "A Deep Learning Based Diabetic Retinopathy Detection from Retinal Images," 2021 International Conference on Intelligent Technologies (CONIT), 2021, pp. 1-5, doi: 10.1109/CONIT51480.2021.9498502.

[24] O. T. Khan and D. Rajeswari, "Brain Tumor detection Using Machine Learning and Deep Learning Approaches," *2022 International Conference on Advances in Computing, Communication and Applied Informatics (ACCAI)*, 2022, pp. 1-7, doi: 10.1109/ACCAI53970.2022.9752502.

[25] R. Mathur, T. Chintala and D. Rajeswari, "Identification of Illicit Activities & Scream Detection using Computer Vision & Deep Learning," *2022 6th International*

Conference on Intelligent Computing and Control Systems (ICICCS), 2022, pp. 1243-1250, doi: 10.1109/ICICCS53718.2022.9787991.

[26] K. He, J. Sun, and X. Tang, “Single image haze removal using dark channel prior,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 33, no. 12, pp. 2341– 2353, Dec. 2011.

APPENDIX

1. IMPORT NECESSARY LIBRARIES

```
!pip install -q tensorflow_model_optimization
import os
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import glob
import random
from PIL import Image
import time
import datetime
import copy

import cv2
import tensorflow as tf
from tensorflow.keras.layers import *
from tensorflow.keras.models import Model
from tensorflow.keras.losses import mean_squared_error
from tensorflow.keras.optimizers import Adam
from keras.models import Sequential

from tensorflow_model_optimization.sparsity import keras as sparsity
import tensorflow_model_optimization as tfmot
```

2. LOADING OF DATA

 # function to load the image in the form of tensors.

```
def load_image(img_path):
img = tf.io.read_file(img_path)
img = tf.io.decode_jpeg(img, channels = 3)
img = tf.image.resize(img, size = (412, 548), antialias = True)
img = img / 255.0
```

```
return img
```

 # function to get the path of individual image.

```
def data_path(orig_img_path, hazy_img_path):
train_img = []
val_img = []
orig_img = glob.glob(orig_img_path + '/*.jpg')
n = len(orig_img)
random.shuffle(orig_img)
train_keys = orig_img[:int(0.9*n)] #90% data for train, 10% for test
val_keys = orig_img[int(0.9*n):]
split_dict = {}
for key in train_keys:
split_dict[key] = 'train'
for key in val_keys:
split_dict[key] = 'val'
hazy_img = glob.glob(hazy_img_path + '/*.jpg')
for img in hazy_img:
img_name = img.split('/')[-1]
orig_path = orig_img_path + '/' + img_name.split('_')[0] + '.jpg'
if (split_dict[orig_path] == 'train'):
train_img.append([img, orig_path])
else:
val_img.append([img, orig_path])
return train_img, val_img
```

 # function to load tensor image data in batches.

```
def dataloader(train_data, val_data, batch_size):
train_data_orig = tf.data.Dataset.from_tensor_slices([img[1] for img in train_data]).map(lambda x: load_image(x))
train_data_haze = tf.data.Dataset.from_tensor_slices([img[0] for img in train_data]).map(lambda x: load_image(x))
train = tf.data.Dataset.zip((train_data_haze, train_data_orig)).shuffle(buffer_size=100).batch(batch_size)
```

```
val_data_orig = tf.data.Dataset.from_tensor_slices([img[1] for img in val_data]).map(lambda x: load_image(x))
val_data_haze = tf.data.Dataset.from_tensor_slices([img[0] for img in val_data]).map(lambda x: load_image(x))
val = tf.data.Dataset.zip((val_data_haze, val_data_orig)).shuffle(buffer_size=100).batch(batch_size)
return train, val
```

3. PRE PROCESSING TECHNIQUES (AIRLIGHT ESTIMATION, BOUNDARY CONSTRAINT, CONTEXTUAL REGUALRIZATION)

```
def Airlight(HazeImg, AirlightMethod, windowSize):
if(AirlightMethod.lower() == 'fast'):
A = []
if(len(HazeImg.shape) == 3):
for ch in range(len(HazeImg.shape)):
kernel = np.ones((windowSize, windowSize), np.uint8)
minImg = cv2.erode(HazeImg[:, :, ch], kernel)
A.append(int(minImg.max()))
else:
kernel = np.ones((windowSize, windowSize), np.uint8)
minImg = cv2.erode(HazeImg, kernel)
A.append(int(minImg.max()))
return(A)

def BoundCon(HazeImg, A, C0, C1, windowSze):
if(len(HazeImg.shape) == 3):

t_b = np.maximum((A[0] - HazeImg[:, :, 0].astype(np.float)) / (A[0] - C0),
(HazeImg[:, :, 0].astype(np.float) - A[0]) / (C1 - A[0]))
t_g = np.maximum((A[1] - HazeImg[:, :, 1].astype(np.float)) / (A[1] - C0),
(HazeImg[:, :, 1].astype(np.float) - A[1]) / (C1 - A[1]))
t_r = np.maximum((A[2] - HazeImg[:, :, 2].astype(np.float)) / (A[2] - C0),
(HazeImg[:, :, 2].astype(np.float) - A[2]) / (C1 - A[2]))
```

```
MaxVal = np.maximum(t_b, t_g, t_r)
transmission = np.minimum(MaxVal, 1)
else:
transmission = np.maximum((A[0] - HazeImg.astype(np.float)) / (A[0] - C0),
(HazeImg.astype(np.float) - A[0]) / (C1 - A[0]))
transmission = np.minimum(transmission, 1)

kernel = np.ones((windowSze, windowSze), np.float)
transmission = cv2.morphologyEx(transmission, cv2.MORPH_OPEN, kernel=kernel)
return(transmission)

def CalTransmission(HazeImg, Transmission, regularize_lambda, sigma):
rows, cols = Transmission.shape

KirschFilters = LoadFilterBank()

# Normalize the filters
for idx, currentFilter in enumerate(KirschFilters):
KirschFilters[idx] = KirschFilters[idx] / np.linalg.norm(currentFilter)

# Calculate Weighting function --> [rows, cols. numFilters] --> One Weighting function
for every filter
WFun = []
for idx, currentFilter in enumerate(KirschFilters):
WFun.append(CalculateWeightingFunction(HazeImg, currentFilter, sigma))

# Precompute the constants that are later needed in the optimization step
tF = np.fft.fft2(Transmission)
DS = 0

for i in range(len(KirschFilters)):
D = psf2otf(KirschFilters[i], (rows, cols))
DS = DS + (abs(D) ** 2)
```

```
# Cyclic loop for refining t and u --> Section III in the paper
beta = 1 # Start Beta value --> selected from the paper
beta_max = 2**8 # Selected from the paper --> Section III --> "Scene Transmission Estimation"
beta_rate = 2*np.sqrt(2) # Selected from the paper

while(beta < beta_max):
gamma = regularize_lambda / beta

# Fixing t first and solving for u
DU = 0
for i in range(len(KirschFilters)):
dt = circularConvFilt(Transmission, KirschFilters[i])
u = np.maximum((abs(dt) - (WFun[i] / (len(KirschFilters)*beta))), 0) * np.sign(dt)
DU = DU + np.fft.fft2(circularConvFilt(u, cv2.flip(KirschFilters[i], -1)))

# Fixing u and solving t --> Equation 26 in the paper
# Note: In equation 26, the Numerator is the "DU" calculated in the above part of the code
# In the equation 26, the Denominator is the DS which was computed as a constant in the above code

Transmission = np.abs(np.fft.ifft2((gamma * tF + DU) / (gamma + DS)))
beta = beta * beta_rate
return(Transmission)

def LoadFilterBank():
KirschFilters = []
KirschFilters.append(np.array([[-3, -3, -3], [-3, 0, 5], [-3, 5, 5]]))
KirschFilters.append(np.array([[-3, -3, -3], [-3, 0, -3], [5, 5, 5]]))
KirschFilters.append(np.array([[-3, -3, -3], [5, 0, -3], [5, 5, -3]]))
KirschFilters.append(np.array([[5, -3, -3], [5, 0, -3], [5, -3, -3]]))
KirschFilters.append(np.array([[5, 5, -3], [5, 0, -3], [-3, -3, -3]]))
KirschFilters.append(np.array([[5, 5, 5], [-3, 0, -3], [-3, -3, -3]]))
KirschFilters.append(np.array([[-3, 5, 5], [-3, 0, 5], [-3, -3, -3]]))
```

```
KirschFilters.append(np.array([[-3, -3, 5], [-3, 0, 5], [-3, -3, 5]]))
KirschFilters.append(np.array([[-1, -1, -1], [-1, 8, -1], [-1, -1, -1]]))
return(KirschFilters)

def CalculateWeightingFunction(HazeImg, Filter, sigma):

# Computing the weight function... Eq (17) in the paper

HazeImageDouble = HazeImg.astype(float) / 255.0
if(len(HazeImg.shape) == 3):
Red = HazeImageDouble[:, :, 2]
d_r = circularConvFilt(Red, Filter)

Green = HazeImageDouble[:, :, 1]
d_g = circularConvFilt(Green, Filter)

Blue = HazeImageDouble[:, :, 0]
d_b = circularConvFilt(Blue, Filter)

WFun = np.exp(-((d_r**2) + (d_g**2) + (d_b**2)) / (2 * sigma * sigma))
else:
d = circularConvFilt(HazeImageDouble, Filter)
WFun = np.exp(-((d ** 2) + (d ** 2) + (d ** 2)) / (2 * sigma * sigma))
return(WFun)

def circularConvFilt(Img, Filter):
FilterHeight, FilterWidth = Filter.shape
assert (FilterHeight == FilterWidth), 'Filter must be square in shape --> Height must be same as width'
assert (FilterHeight % 2 == 1), 'Filter dimension must be a odd number.'

filterHalsSize = int((FilterHeight - 1)/2)
rows, cols = Img.shape
```

```
PaddedImg = cv2.copyMakeBorder(Img, filterHalsSize, filterHalsSize, filterHalsSize, filterHalsSize, borderType=cv2.BORDER_WRAP)
FilteredImg = cv2.filter2D(PaddedImg, -1, Filter)
Result = FilteredImg[filterHalsSize:rows+filterHalsSize, filterHalsSize:cols+filterHalsSize]

return(Result)

##################
def psf2otf(psf, shape):
"""
Convert point-spread function to optical transfer function.
Compute the Fast Fourier Transform (FFT) of the point-spread
function (PSF) array and creates the optical transfer function (OTF)
array that is not influenced by the PSF off-centering.
By default, the OTF array is the same size as the PSF array.
To ensure that the OTF is not altered due to PSF off-centering, PSF2OTF
post-pads the PSF array (down or to the right) with zeros to match
dimensions specified in OUTSIZE, then circularly shifts the values of
the PSF array up (or to the left) until the central pixel reaches (1,1)
position.
Parameters
----------
psf : `numpy.ndarray`
PSF array
shape : int
Output shape of the OTF array
Returns
-------
otf : `numpy.ndarray`
OTF array
Notes
-----
Adapted from MATLAB psf2otf function
"""
```

```
if np.all(psf == 0):
return np.zeros_like(psf)

inshape = psf.shape
# Pad the PSF to outsize
psf = zero_pad(psf, shape, position='corner')

# Circularly shift OTF so that the 'center' of the PSF is
# [0,0] element of the array
for axis, axis_size in enumerate(inshape):
psf = np.roll(psf, -int(axis_size / 2), axis=axis)

# Compute the OTF
otf = np.fft.fft2(psf)

# Estimate the rough number of operations involved in the FFT
# and discard the PSF imaginary part if within roundoff error
# roundoff error = machine epsilon = sys.float_info.epsilon
# or np.finfo().eps
n_ops = np.sum(psf.size * np.log2(psf.shape))
otf = np.real_if_close(otf, tol=n_ops)

return otf

def zero_pad(image, shape, position='corner'):
"""
Extends image to a certain size with zeros
Parameters
----------
image: real 2d `numpy.ndarray`
Input image
shape: tuple of int
Desired output shape of the image
position : str, optional
```

```
The position of the input image in the output one:
* 'corner'
top-left corner (default)
* 'center'
centered
Returns
-------
padded_img: real `numpy.ndarray`
The zero-padded image
"""
shape = np.asarray(shape, dtype=int)
imshape = np.asarray(image.shape, dtype=int)

if np.alltrue(imshape == shape):
return image

if np.any(shape <= 0):
raise ValueError("ZERO_PAD: null or negative shape given")

dshape = shape - imshape
if np.any(dshape < 0):
raise ValueError("ZERO_PAD: target size smaller than source one")

pad_img = np.zeros(shape, dtype=image.dtype)

idx, idy = np.indices(imshape)

if position == 'center':
if np.any(dshape % 2 != 0):
raise ValueError("ZERO_PAD: source and target shapes "
"have different parity.")
offx, offy = dshape // 2
else:
offx, offy = (0, 0)
```

```
pad_img[idx + offx, idy + offy] = image

return pad_img

def removeHaze(HazeImg, Transmission, A, delta):
'''
:param HazeImg: Hazy input image
:param Transmission: estimated transmission
:param A: estimated airlight
:param delta: fineTuning parameter for dehazing --> default = 0.85
:return: result --> Dehazed image
'''

# This function will implement equation(3) in the paper
# " https://www.cv-foundation.org/openaccess/content_iccv_2013/papers/Meng_Efficient_Image_Dehazing_2013_ICCV_paper.pdf "

epsilon = 0.0001
Transmission = pow(np.maximum(abs(Transmission), epsilon), delta)
HazeCorrectedImage = copy.deepcopy(HazeImg)
if len(HazeImg.shape) == 3:
for ch in range(len(HazeImg.shape)):
temp = ((HazeImg[:, :, ch].astype(float) - A[ch]) / Transmission) + A[ch]
temp = np.maximum(np.minimum(temp, 255), 0)
HazeCorrectedImage[:, :, ch] = temp
else:
temp = ((HazeImg.astype(float) - A[0]) / Transmission) + A[0]
temp = np.maximum(np.minimum(temp, 255), 0)
HazeCorrectedImage = temp
return(HazeCorrectedImage)
```

4. CNN MODEL (VGG16, ALEXNET, DENSENET, RESNET)

```
def vgg16():
model = Sequential()
model.add(Conv2D(filters=64,
kernel_size=(3, 3),
padding='same',
activation='relu',
input_shape=(412,548,3),
name='conv1_1'))
model.add(Conv2D(filters=64,
kernel_size=(3, 3),
padding='same',
activation='relu',
name='conv1_2'))
model.add(MaxPooling2D(pool_size=(2,2),
strides=(2,2),
name='max_pooling2d_1'))

model.add(Conv2D(filters=128,
kernel_size=(3, 3),
padding='same',
activation='relu',
name='conv2_1'))
model.add(Conv2D(filters=128,
kernel_size=(3, 3),
padding='same',
activation='relu',
name='conv2_2'))
model.add(MaxPooling2D(pool_size=(2,2),
strides=(2,2),
name='max_pooling2d_2'))

model.add(Conv2D(filters=256,
kernel_size=(3, 3),
padding='same',
```

```
activation='relu',
input_shape=(412,548,3),
name='conv3_1'))
model.add(Conv2D(filters=256,
kernel_size=(3, 3),
padding='same',
activation='relu',
name='conv3_2'))
model.add(Conv2D(filters=256,
kernel_size=(3, 3),
padding='same',
activation='relu',
name='conv3_3'))
model.add(MaxPooling2D(pool_size=(2,2),
strides=(2,2),
name='max_pooling2d_3'))

model.add(Conv2D(filters=512,
kernel_size=(3, 3),
padding='same',
activation='relu',
input_shape=(412,548,3),
name='conv4_1'))
model.add(Conv2D(filters=512,
kernel_size=(3, 3),
padding='same',
activation='relu',
name='conv4_2'))
model.add(Conv2D(filters=512,
kernel_size=(3, 3),
padding='same',
activation='relu',
name='conv4_3'))
model.add(MaxPooling2D(pool_size=(2,2),
```

```
strides=(2,2),
name='max_pooling2d_4'))

model.add(Conv2D(filters=512,
kernel_size=(3, 3),
padding='same',
activation='relu',
input_shape=(412,548,3),
name='conv5_1'))
model.add(Conv2D(filters=512,
kernel_size=(3, 3),
padding='same',
activation='relu',
name='conv5_2'))
model.add(Conv2D(filters=512,
kernel_size=(3, 3),
padding='same',
activation='relu',
name='conv5_3'))

model.add(MaxPooling2D(pool_size=(2,2),
strides=(2,2),
name='max_pooling2d_5'))

model.add(Flatten(name='flatten'))
model.add(Dense(4096, activation='relu', name='fc_1'))
model.add(Dropout(0.5, name='dropout_1'))
model.add(Dense(4096, activation='relu', name='fc_2'))
model.add(Dropout(0.5, name='dropout_4'))
model.add(Dense(1000, activation='softmax', name='output'))
# model.add(Reshape((412,548,3)))
model.summary()
# return Model(inputs=model.input, outputs=model.get_layer('output').output)
return Model(inputs=model.input, outputs=model.input)
```

PAPER PUBLICATION STATUS

We have Submitted our research paper to multiple conference waiting for approval. Proof of submission is hereby attached.

1. ICECONF - 2023

The status for ICECONF is that the paper is recommended for publication, but minor revisions are recommended by the reviewer(s).

On Tue, Nov 15, 2022 at 12:23 PM ICECONF St. Joseph's Institute of Technology <iceconf@stjosephstechnology.ac.in> wrote:

Dear Author,

Manuscript ID: ICECONF-23-T3-309 entitled " Monochromatic Image Dehazing Using Enhanced Feature Extraction Techniques in Deep Learning " which you submitted to International Conference on Artificial Intelligence and Knowledge Discovery in Concurrent Engineering (ICECONF -2023) has been reviewed. The comments of the reviewer(s) are included below of this letter.

The reviewer(s) have recommended publication, but also suggest some minor revisions to your manuscript. Therefore, I invite you to respond to the reviewer(s)' comments and revise your manuscript.

Reviewer(s)' Comments to Author:

1. Kindly reduce the plagiarism and require <=18% (Including reference section).
2. Required Performance analysis graph that must be justified with performance results. Result oriented paper must have supporting table & figures.
3. In the Results Section, the performance results should be justified with a performance graph. (requires 2 or 3 graphs and tables).
4. Every Reference must be cited inside the paper. Citations should be sequential ascending order in number format.
5. Strictly follow the IEEE Structure, i.e., Introduction, Related Work, Proposed Methodology, Results & Discussion and Conclusion, finally reference section.
6. Maintain a minimum 15 recent year papers.
7. Every figure must have one figure caption, number & respective callouts.
8. Every table must have one table caption, number & respective callouts.
9. Table must be drawn as an editable table format.
10. Rewrite equation using equation editor tool or Mathtype.

Kindly rectify the above corrections and submit the revised paper to iceconf@stjosephstechnology.ac.in **on or before 17.11.2022. While sending your manuscript, mention your paper id in the subject line and rename your file name as paper id without fail.**

Best regards,
Organizing Committee,
ICECONF-2023,
St. Joseph's Institute of Technology, OMR, Chennai-119.

2. SMARTGENCON2022 AND ICICI 2023

The paper is submitted to the conference and waiting for approval.

Hello,

The following submission has been created.

Track Name: SMARTGENCON2022

Paper ID: 464

Paper Title: Monochromatic Image Dehazing Using Enhanced Feature Extraction Techniques in Deep Learning

Abstract:
Images photographed in foggy weather usually have poor visibility. To mitigate this problem researchers have come up with various image dehazing techniques. Now, more than ever, high-quality images that can be used to glean maximum information from autonomous systems are in high demand. This study uses different convolutional neural network (CNN) architectures to draw out essential details from the picture and localize the information recovered to reduce the haze from the picture. Three pre-processing techniques such as Air light estimation, Contextual regularization and Boundary constraint is used in this work. Various combinations of experiments are done with three pre-processing techniques and four CNN architectures. Experimental results shows that VGG16 achieves 28.35 PSNR and DenseNet achieves 0.825 SSIM.

Created on: Tue, 15 Nov 2022 05:35:50 GMT

Last Modified: Tue, 15 Nov 2022 05:35:50 GMT

Authors:
- nd1894@srmist.edu.in
- sb7908@srmist.edu.in
- drajiit@gmail.com (Primary)
- srinivar@srmist.edu.in

Secondary Subject Areas: Not Entered
Submission Files: Image_dehazing_IEEE.pdf (1 Mb, Tue, 15 Nov 2022 05:35:38 GMT)

Dear authors,

We received your submission to ICICI 2023 (International Conference on Intelligent Data Communication Technologies and Internet of Things):

Authors : Nisarg Doshi, Sagar Bhavsar, Rajeswari Devarajan and Srinivasan R

PLAGIARISM REPORT

image dehazing

ORIGINALITY REPORT

9%	6%	8%	2%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	www.cv-foundation.org Internet Source	1%
2	Qinghua Mao, Yufei Wang, Xuhui Zhang, Xiaoyong Zhao, Guangming Zhang, Kundayi Mushayi. "Clarity method of fog and dust image in fully mechanized mining face", Machine Vision and Applications, 2022 Publication	1%
3	openaccess.thecvf.com Internet Source	1%
4	ijisrt.com Internet Source	1%
5	Submitted to Edith Cowan University Student Paper	1%
6	Submitted to Liverpool John Moores University Student Paper	1%
7	arxiv.org Internet Source	1%

8	docplayer.net Internet Source	<1 %
9	Chia-Hung Yeh, Chih-Hsiang Huang, Li-Wei Kang, Min-Hui Lin. "Single Image Dehazing via Deep Learning-based Image Restoration", 2018 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC), 2018 Publication	<1 %
10	Wei Liu, Xianxu Hou, Jiang Duan, Guoping Qiu. "End-to-End Single Image Fog Removal Using Enhanced Cycle Consistent Adversarial Networks", IEEE Transactions on Image Processing, 2020 Publication	<1 %
11	opus.lib.uts.edu.au Internet Source	<1 %
12	7321f47c-1fec-4851-906d-76694fca355c.filesusr.com Internet Source	<1 %
13	Ziqi Zhang, Zeyu Li, Kun Wei, Siduo Pan, Cheng Deng. "A Survey on Multimodal-Guided Visual Content Synthesis", Neurocomputing, 2022 Publication	<1 %
14	Chia-Hung Yeh, Chih-Hsiang Huang, Li-Wei Kang. "Multi-Scale Deep Residual Learning-	<1 %

Based Single Image Haze Removal via Image Decomposition", IEEE Transactions on Image Processing, 2020
Publication

15	Gaofeng Meng, Ying Wang, Jiangyong Duan, Shiming Xiang, Chunhong Pan. "Efficient Image Dehazing with Boundary Constraint and Contextual Regularization", 2013 IEEE International Conference on Computer Vision, 2013 Publication	<1 %

Exclude quotes	On	Exclude matches	< 3 words
Exclude bibliography	On		