

IMAGE RESTORATION USING INCEPTION

Submitted in partial fulfillment of the requirements for
the award of
Bachelor of Engineering degree in Computer Science and Engineering

by

VYTLAHEMANTH(Reg.No.37110854)

YADALA SAI VARUN(Reg.No.37110855)



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING**

**SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

**Accredited with Grade "A" by NAAC
JEPPIAAR NAGAR, RAJIV GANDHI
SALAI, CHENNAI-600 119**

MARCH-2021



SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited with "A" grade by NAAC
Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai-600119
www.sathyabama.ac.in



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this project report is the bonafide work of **VYTILA HEMANTH (Reg.No.37110854)** and **YADALA SAI VARUN (Reg.No.37110855)** who carried out the project entitled "**IMAGE RESTORATION USING INCEPTION**" under my supervision from **November 2020** to **March 2021**.

Internal Guide
Dr. A. Jesudoss, M.E., Ph.D.

Head of the Department

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

VYTLA HEMANTH hereby declare that the Project Report entitled “**IMAGE RESTORATION USING INCEPTION**” is done by me under the guidance of **Dr. A. Jesudoss, M.E., Ph.D.**, Department of Computer Science and Engineering at Sathyabama Institute of Science and Technology is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering.

DATE:

PLACE:CHENNAI

SIGNATUREOFTHECANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management of SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala, M.E., Ph.D., Dean**, School of Computing, **Dr. S. Vigneswari, M.E., Ph.D., and Dr. L. Lakshmanan, M.E., Ph.D., Heads of the Department** of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr. A. Jesudoss, M.E., Ph.D., Assistant Professor**, for his valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

Image Restoration is the process of recovering the original image by removing noise and blur from image. Image blur is difficult to avoid in many situations like photography, to remove motion blur caused by camera shake, radar imaging to remove the effect of image system response etc. Image noise is unwanted signal which comes in image from sensor such as thermal or electrical signal and environment condition such as rain, snow etc. The image degradation could come from coding artifacts, resolution limitation, transmission noise, object motion, camera shake, or a combination of them. Image decomposition is used for decomposing the distorted image into a texture layer (High Frequency HR Component) and a structure layer (Low Frequency LF Component) with the goal to separate HF and LF artifacts. The existing system is a flexible deep CNN framework which exploits the frequency characteristics of different types of artifacts. Hence, the same approach can be employed for a variety of image restoration tasks by adjusting the architecture. For reducing the artifacts with similar frequency characteristics, a quality enhancement network which adopts residual and recursive learning is proposed. Residual learning is utilized to speed up the training process and boost the performance; recursive learning is adopted to significantly reduce the number of training parameters, as well as boost the performance.

TABLE OF CONTENTS

	ABSTRACT	v
	LIST OF FIGURES	viii
CHAPTER NO	TITLE	PAGE NO
1	INTRODUCTION	1
	1.1 OVERVIEW	1
	1.2SCOPE OF THE PROJECT	1
	1.3 DOMAUN OVERVIEW	1
	1.3.1 NUMPY	2
	1.3.2 MATHPOTLIB	3
	1.3.3 DEEP LEARNING	4
2	LITERATURE SURVEY	7
	2.1 LITERATURE SURVEY 1	7
	2.2 LITERATURE SURVEY 2	8
	2.3 LITERATURE SURVEY 3	10
	2.4 LITERATURE SURVEY 4	11
3	SOFTWARE DEVELOPMENT	14
	METHODOLOGY	
	3.1 SYSTEM ANALYSIS	14
	3.2 WATERFALL MODEL	15
	3.2.1 REQUIRMENT GATHERING	15
	3.2.2 SYSTEM DESIGN	15
	3.2.3 IMPLEMENTATION	15
	3.2.4 TESTING	15
	3.2.5 DELIVERY	16
	3.2.6 MAINTENANCE	16
	3.3 RAD MODEL	16
	3.3.1 PHASE 1	16
	3.3.2 PHASE 2	17

	3.3.3 PHASE 3	17
	3.3.4 PHASE 4	18
	3.3.5 ADVANTAGES OF RAD	18
	3.4 SOFTWARE TESTING	18
	3.5 TEST DRIVEN DEVELOPMENT	20
	3.5.1 UNIT TESTING	21
	3.5.2 INTEGRATION TESTING	21
	3.5.3 SYSTEM TESTING	22
	3.5.4 SANITY TESTING	23
	3.5.5 REGRESSION TESTING	23
	3.5.6 PERFORMANCE TESTING	24
4	SYSTEM DESIGN AND METHODOLOGY	26
	4.1 EXISTING SYSTEM	26
	4.1.1 DRAWBACKS OF EXISTING SYSTEM	27
	4.2 PROPOSED SYDTEM	27
	4.2.1 ADVANTAGES OF PROPOSED SYSTEM	27
	4.3 REQUIRMENTS SPECIFICATIONS	28
	4.3.1 HARDWARE REQUIRMENTS	28
	4.3.2 SOFTWARE REQUIRMENTS	28
	4.4 SYSTEM DESIGN	28
	4.4.1 ARCHITECTURE DIAGRAM	29
	4.5 SYSTEM IMPLEMENTATION	29
	4.5.1 LIST OF MODULES	29
	4.5.2 DATA PROCESSING	30
	4.5.3 LOAD WEIGHTS	30
	4.5.4 PREDICTION	31
5	RESULTS AND DISCUSSION	33
6	CONCLUSION AND FUTURE WORK	35
	REFERENCES	36

APPENDIX	38
A) SOURCE CODE	38

LIST OF FIGURES

FIGURE NO	FIGURE NAME	PAGE NO
4.1	BLOCK DIAGRAM	29
5.1	TRAIN DATA	33
5.2	TEST DATA (1)	33
5.3	TEST DATA (2)	34

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

In Today's world the images get blur due to various reasons such as camera movement, subject movement, missed focus, insufficient depth of field, and lens softness. These images are not clearly visible. When an image is blurred we make the color transition from one side of an edge in the image to another smooth rather than sudden. The effect is to average out rapid changes in pixel intensity. The blur, or smoothing, of an image removes outlier pixels that may be noise in the image.

The purpose of Image Restoration is to compensate for or undo defects which degrade an image. Degradation comes in many forms such as motion blur, noise. In cases like motion blur it is possible to come up with a good estimate of actual blur and undo the blur to restore the original image. In cases where the image is corrupted by noise, the best way is to compensate for the degradation cause by it. This proposed system can also used for ancient Pictures or painting. It remove all unwanted blurs and noises in pictures or painting and also it can convert black and white images in to colour photos

1.2 SCOPE OF THE PROJECT

The main contributions of this project therefore are:

- Data Analysis
- Dataset Preprocessing
- Training the Model
- Testing of Dataset

1.3 DOMAIN OVERVIEW

Deep learning has evolved hand-in-hand with the digital era, which has brought about an explosion of data in all forms and from every region of the world. This data, known simply as big data, is drawn from sources like social media, internet search engines, e-commerce platforms, and online cinemas,

among others. This enormous amount of data is readily accessible and can be shared through fin tech applications like cloud computing.

Deep learning is a machine learning technique that teaches computers to do what comes naturally to humans. Deep learning is a key technology behind driverless cars, enabling them to recognize a stop sign, or to distinguish a pedestrian from a lamp post. It is the key to voice control in consumer devices like phones, tablets, TVs, and hands-free speakers. Deep learning is getting lots of attention lately and for good reason. It's achieving results that were not possible before.

In deep learning, a computer model learns to perform classification tasks directly from images, text, or sound. Deep learning models can achieve state-of-the-art accuracy, sometimes exceeding human-level performance. Models are trained by using a large set of labeled data and neural network architectures that contain many layers.

However, the data, which normally is unstructured, is so vast that it could take decades for humans to comprehend it and extract relevant information. Companies realize the incredible potential that can result from unraveling this wealth of information and are increasingly adapting to AI systems for automated support.

1.3.1 NUMPY

Numpy is the core library for scientific computing in Python. It provides a high-performance multidimensional array object, and tools for working with these arrays.

A numpy array is a grid of values, all of the same type, and is indexed by a tuple of non negative integers. The number of dimensions is the rank of the array; the shape of an array is a tuple of integers giving the size of the array along each dimension.

Numpy is one of the important packages mainly because this library provides you with an array data structure that holds some benefits over python

lists, such as being more compact, faster access in reading and writing items, being more convenient and more efficient.

Numpy is a Python library that is the core library for scientific computing in Python. It contains a collection of tools and techniques that can be used to solve on a computer mathematical models of problems in Science and Engineering. One of these tools is a high-performance multidimensional array object that is a powerful data structure for efficient computation of arrays and matrices. To work with these arrays, there's a vast amount of high-level mathematical functions operate on these matrices and arrays.

An array is basically nothing but pointers. It's a combination of a memory address, a data type, a shape, and strides:

- The data pointer indicates the memory address of the first byte in the array,
- The data type or dtype pointer describes the kind of elements that are contained within the array,
- The shape indicates the shape of the array.
- The strides are the number of bytes that should be skipped in memory to go to the next element. If your strides are (10,1), you need to proceed one byte to get to the next column and 10 bytes to locate the next row.

Numpy has several advantages over using core Python mathematical functions, a few of which are outlined here:

- Numpy is extremely fast when compared to core Python thanks to its heavy use of C extensions.
- Many advanced Python libraries such as Scikit-Learn, Scipy and Keras, make extensive use of the Numpy library. Therefore, if you plan to pursue a career in data science or machine learning, Numpy is a very good tool to master.
- Numpy comes with a variety of built-in functionalities, which in core Python would take a fair bit of custom code.

1.3.2 MATHPLOTLIB

Plotting of data can be extensively made possible in an interactive way by Matplotlib, which is a plotting library that can be demonstrated in Python scripts. Plotting of graphs is a part of data visualization, and this property can be achieved by making use of Matplotlib.

One of the free and open-source Python library which is basically used for technical and scientific computing is Python SciPy. Matplotlib is widely used in SciPy as most scientific calculations require plotting of graphs and diagrams.

Matplotlib can be used to create publication quality figures in a variety of hardcopy formats and interactive environments across platforms.

Another characteristic of Matplotlib is its steep learning curve, which means that users usually make rapid progress after having started.

1.3.3 DEEP LEARNING

Deep learning is a computer software that mimics the network of neurons in a brain. It is a subset of machine learning and is called deep learning because it makes use of deep neural networks. Deep learning algorithms are constructed with connected layers. It's open-source tools that make programming easier, while computing more affordable. Open-source frameworks and libraries today make it possible for companies to benefit from image recognition technology exponentially. For instance, such massive open databases as Pascal VOC and ImageNet give access to millions of tagged images.

- The first layer is called the Input Layer
- The last layer is called the Output Layer
- All layers in between are called Hidden Layers. The word deep means the network join neurons in more than two layers.

A deep neural network provides state-of-the-art accuracy in many tasks, from object detection to speech recognition. They can learn automatically, without predefined knowledge explicitly coded by the

programmers.

Image Recognition Image recognition or a computer vision is a technical discipline that deals with searching the ways to automate all the job that a human visual system can do.

Each layer represents a deeper level of knowledge,i.e., the hierarchy of knowledge. A neural network with four layers will learn more complex feature than with that with two layers.

The learning occurs in two phases.

- The first phase consists of applying a nonlinear transformation of the input and create a statistical model as output.
- The second phase aims at improving the model with a mathematical method known as derivative.

The neural network repeats these two phases hundreds to thousands of time until it has reached a tolerable level of accuracy. The repeat of this two-phase is called an iteration

Classification of Neural Networks

- **Shallow neural network:** The Shallow neural network has only one hidden layer between the input and output.
- **Deep neural network:** Deep neural networks have more than one layer.

For instance, Google LeNet model for image recognition counts 22 layers.

The computational models in Deep Learning are loosely inspired by the human brain. The multiple layers of training are called Artificial Neural Networks (ANN). **Image Processing Techniques** Generally image processing consists of several stages: image import, analysis, manipulation and image output. There are two methods of image processing: digital and analogue. In particular, digital image processing and its techniques is what this article is about. Computer algorithms play a crucial role in digital image processing.

Developers use multiple algorithms to solve different tasks, including digital image detection, analysis, reconstruction, restoration, image data

compression, image enhancement, image estimation and images pectral estimation. Major techniques of digital image processing are as follows: Image Editing, which basically means altering digital images by means of graphic software tools. Image Restoration, which refers to the estimation of a clean original image out of the corrupt image taken in order to get back the information lost. Independent Component Analysis, which separates a multivariate signal computationally into additive subcomponents.

Anisotropic Diffusion, which is often known as Perona-Malik Diffusion, makes it possible to reduce image noise without having to remove important parts of the image. Linear Filtering. It's another digital image processing technique, which refers to processing time-varying input signals and producing output signals that are subject to the constraint of linearity. Neural Networks, which are computational models widely used in machine learning for solving various tasks.

Pixelation, which often refers to turning printed images into digitized ones (such as GIF). Principal Components Analysis, a digital image processing technique that can be used for feature extraction. Partial Differential Equations, which also is dealing with effectively de-noising images. Hidden Markov Models, a technique used for image analysis in two dimensions. Wavelets, which stands for a mathematical function that's used in image compression.

CHAPTER 2

LITERATURE SURVEY

2.1 LITERATURE SURVEY 1

Title	Reconstruction of Single Image from Multiple Blurry Measured Images
Authors	Tsung-Ching Lin, Liming Hou, Hongqing Liu, Yong Li ³ , Trieu-Kien Truong
Published Year	2018
Efficiency	• Significantly outperforms many state-of-the-art methods. • Enable the categorisation of natural and man-made images. • Integrates the statistics of derivative filters into a maximum likelihood method for blind motion deblurring.
Drawbacks	• Rely on generic image priors such as the sparsity of salient features including image gradients and edges. • Difficult to recover by generic image priors • Limited to blurs caused only by a one-dimensional box kernel.
Description	In the proposed approach, the group sparse regularization on both the blur kernel and image is provided, where the sparse solution is promoted by ℓ_1-norm. In addition, the reweighted data fidelity is developed to further improve the recovery performance, where the weight is determined by the estimation error. Moreover, to reduce the undesirable noise effects in group sparse representation, distance measures are studied in the block matching process to find similar patches. In such a joint deblurring approach, a more sophisticated two-step interactive process is needed in which each step is solved by means of the well-known split Bregman iteration (SBI) algorithm, which is generally used to efficiently solve the proposed joint deblurring problem. Finally, numerical studies,

	including synthetic and real images, demonstrate that the performance of this joint estimation algorithm is superior to previous state-of-the-art algorithms in terms of both objective and subjective evaluation standards. The recovery results of real captured images using unmanned aerial vehicles (UAV) are also provided to further validate the effectiveness of the proposed method. Images containing small text are utilized. The images recovered by Algorithms are still contaminated by undesired noise, whereas the image obtained from Algorithm is still unclear because of the ringing effect. This means that the blurs are not completely removed. This is because our approach achieves a good balance of deblurring and denoising, producing a high-quality reconstructed image. Finally, images consisting of relatively large text are used. The image obtained from Algorithm seems to be unclear and is predominantly noisy, whereas the images recovered by Algorithms still contain the blur effect, especially in the region characterized by the text.
--	--

2.2 LITERATURE SURVEY 2

Title	Image Deblurring with a Class-Specific Prior
Authors	Saeed Anwar, Cong Phuoc Huynh, Fatih Porikli
Published Year	2018
Efficiency	• Minimize the Cost • Resulting image is of high quality in terms of clarity and detail preservations • Reweighted data reliability is created to further enhance the recovery execution
Drawbacks	• ill-posed problem becomes a well-posed one • Kernel estimation can be blurry, especially when the noise is high

	• Does not perform deconvolution from a pair of blurred/noisy images
Description	This paper poses the following specific questions: (i) Does any image class information offer an advantage over existing generic priors for image quality restoration? (ii) If a class-specific prior exists, how should it be encoded into a deblurring framework to recover attenuated image frequencies? Throughout this work, we devise a class-specific prior based on the band-pass filter responses and incorporate it into a deblurring strategy. More specifically, we show that the subspace of band-pass filtered images and their intensity distributions serve as useful priors for recovering image frequencies that are difficult to recover by generic image priors. We demonstrate that our image deblurring framework, when equipped with the above priors, significantly outperforms many state-of-the-art methods using generic image priors or class-specific exemplars. This paper formulates the deblurring problem in the Fourier domain by first obtaining a bank of Butterworth band pass filters, each of which has a constant magnitude in a certain (2D) frequency band and zero elsewhere. The visual representation of band pass filters in the frequency domain are concentric circular bands (centred at the origin) with unit values. To filter an image with a Butterworth band pass filter, we first clip its frequency components in the Fourier domain to the range defined by the band pass filter (corresponding to the filter's non-zero frequencies). Subsequently, the remaining frequency components are transformed to the spatial domain via an inverse Fourier transform. The main underlying hypothesis of our novel class specific prior with the hypothesis is that the frequency components in each band span a sparse linear subspace in the Fourier domain. The algorithm commences with the initialisation of the latent image and the kernel to the given blurred image and the Dirac delta function, respectively. Subsequently, it proceeds in an iterative manner. In each iteration, we minimise the objective function with respect to w, x and k

	in alternating steps. The update steps for x and k are undertaken by fast forward and inverse Fourier transforms. After every iteration, k is centred and normalised so that the sum of its elements is unity. Meanwhile, to solve for w , we minimise the cost function using the L1 least-squares solve.
--	--

2.3 LITERATURE SURVEY 3

Title	Blind Image Deblurring Using Row-Column Sparse Representations
Authors	Mohammad Tofighi, Yuelong Li, Vishal Monga
Published Year	2017
Efficiency	• Removes camera shakes from multiple photographs • Efficient approach for solving blur estimation and image recovery utilizing multiple images • Further enhance the recovery performance
Drawbacks	• Each new prior requires the development of a custom optimization method • Non-local filters have not been used for image denoising • Does not solve the intrinsic image gradient and blur kernel
Description	In this paper, we develop a new method called Blind Image Deblurring using Row-Column Sparsity (BD-RCS) to address this issue. Specifically, we model the outer product of kernel and image coefficients in certain transformation domains as a rank-one matrix, and recover it by solving a rank minimization problem. Our central contribution then includes solving two new optimization problems involving row and column sparsity to automatically determine blur kernel and image support sequentially. The kernel and image can then be recovered through a singular value decomposition (SVD). Experimental results on linear motion deblurring demonstrate that BD-RCS can yield better results than state of the art, particularly when the

	blur is caused by large motion. This is confirmed both visually and through quantitative measures. In this paper, develop a novel image deblurring method called Blind Image Deblurring using Row-Column Sparse Representations (BD-RCS). This paper work formulates a rank-one matrix recovery problem but we set up two new optimization problems involving row and column sparsity to automatically determine blur kernel and image support respectively. Note that in the analytical development, In this paper don't pose any assumptions on the types and shapes of the kernel and thus BD-RCS is versatile across several practical blur models. In this work, we represent the image in the Haar wavelet domain but this transform is a flexible parameter in our work and its exact choice could be driven by the experimental scenario for completeness, we first briefly review the technique of formulating blind deconvolution as a rank-one matrix recovery problem. We also discuss its fundamental limitations towards practical applications To recover the image support, we simply note that k and x play symmetric roles in our model. Just as initializing the kernel support with a bounding box, we aim to initialize the image support with a "container". Indeed, in Haar wavelet the locations of dominant image coefficients largely stays stable even under large motion blurring, and through mildly thresholding the blurred image coefficients, we can achieve moderate dimensionality reduction while maintaining most of the support information.
--	--

2.4 LITERATURE SURVEY 4

Title	Blind Image Blur Estimation via Deep Learning
Authors	Ruomei Yan, Ling Shao
Published Year	2016

Efficiency	• Efficiently solve the optimization problem • Stochastic solver does not require the explicit computation of the gradient of the objective function • New priors can be implemented and tested very quickly
Drawbacks	• Are non-convex and often difficult to solve • Cannot recover all color channels of the intrinsic image simultaneously using the cross-channel information • Cannot allow for quick exploration of new ideas in this space
Description	This paper aim at identifying the blur type for each input image patch, and then estimating the kernel parameter in this paper. A learning-based method using a pre-trained Deep Neural Network (DNN) and a General Regression Neural Network (GRNN) is proposed to first classify the blur type and then estimate its parameters, taking advantages of both the classification ability of DNN and the regression ability of GRNN. To the best of our knowledge, this is the first time that pre-trained DNN and GRNN have been applied to the problem of blur analysis. Firstly, our method identifies the blur type from a mixed input of image patches corrupted by various blurs with different parameters. To this aim, a supervised DNN is trained to project the input samples into a discriminative feature space, in which the blur type can be easily classified. Then, for each blur type, the proposed GRNN estimates the blur parameters with very high accuracy. In blind image deconvolution, it is not easy to recover the PSF from a single blurred image due to the loss of information during blurring. Our goal is to classify the blurred patches into different blur types according to the blur related features and then we estimate blur parameters for each classified patch. Local autocorrelation congruency, is used as a feature for the Bayes classifier to discriminate motion blur from defocus blur because the descriptor is strongly related to the shape and value of the PSF. Later,

	alternative handcrafted features for blur classification, which gives better results without any training. Though both methods generate good results on identifying motion blur and defocus blur, the features they used are limited to a single or several blur kernels. In this paper, we propose a general feature extractor for common blur kernels with various parameters, which is closer to realistic application scenarios. When designing the DBN unsupervised training step, it is natural to use observed blurred patches as training and testing samples. However, their characteristics are not as obvious as their frequency coefficients. Hence, the logarithmic power spectra are adopted as input features for the DBN, since the PSF in the frequency domain manifests different characteristics for different blur kernels. Scaling continuous-valued input to (0, 1) worked well for pixel gray levels, but it is not necessarily appropriate for other kinds of input data.
--	--

CHAPTER 3

SOFTWARE DEVELOPMENT METHODOLOGY

3.1 SYSTEM ANALYSIS

There are various software development methodologies available in the industry. They are.

1. Agile Software Development
2. Crystal Methods
3. Dynamic Systems Development Model (DSDM)
4. Extreme Programming (XP)
5. Feature Driven Development (FDD)
6. Joint Application Development (JAD)
7. Lean Development (LD)
8. Rapid Application Development (RAD)
9. Rational Unified Process (RUP)
10. Scrum
11. Spiral
12. Systems Development Life Cycle (SDLC)
13. Waterfall Model

Performance Testing

- I. Load Testing
- II. Stress Testing
- III. Volume Testing
- IV. Security Testing
- V. Compatibility Testing
- VI. Install Testing
- VII. Recovery Testing
- VIII. Reliability Testing
- IX. Usability Testing
- X. Compliance Testing
- XI. Localization Testing

Among all these methodologies, we have decided to compare the two most popular and widely used methodologies:

- Waterfall Model
- RAD Model

3.2 WATERFALL MODEL

It is called as traditional approach. Waterfall is a linear method (sequential model) for any software application development. Here application development is segregated into a sequence of pre-defined phases.

3.2.1 REQUIREMENT GATHERING

In this stage, gathering of comprehensive data approximately what this challenge requires is done. To gather this data in a diffusion of ways, from interviews to questionnaires by interactive brainstorming. By the completion of this section, the task requirements have to be clean, and also to have a necessities file.

3.2.2 SYSTEM DESIGN

Using the well-known requirements, project team designs the solutions. During this phase, no development will be happening. But the project team starts specification such as programming language or hardware requirements.

3.2.3 IMPLEMENTATION

During this phase software development coding will be happening. Web application programmers take data from the previous stage and create a functional product. Web application programmers write source code in small pieces, which are integrated at the end of this phase or the beginning of the next.

3.2.4 TESTING

Once all coding is done, testing of the product can begin. Testers

methodically find and report any problems. If serious issues arise, your project may need to return to phase one for re-evaluation.

3.2.5 DELIVERY

In this phase, the solution is complete, and project team submits the deliverables to be deployed or released.

3.2.6 MAINTENANCE

The final solution has been implemented to the client and is being used. As troubles arises, the project team might also want to create patches and updates might also to deal with them. Again, huge troubles may also necessitate a return to segment one.

3.3 RAD MODEL

Agile is an iterative, team-based approach to development. This method highlights the speedy delivery of a software application in comprehensive functional components.

3.3.1 PHASE 1

This phase is equivalent to a project scoping meeting. Although the planning phase is condensed compared to other project management methodologies, this is a critical step for the ultimate success of the project. During this stage, web application programmers, clients (software users), and team members communicate to determine the goals and expectations for the project as well as current and potential issues that would need to be addressed during the build.

A basic breakdown of this stage involves:

- Researching the current problem
- Defining the requirements for the project
- Finalizing the requirements with each stakeholder's approval

It is important that everyone has the opportunity to evaluate the goals and

expectations for the project and weigh in. By getting approval from each key stakeholder and web application programmer, teams can avoid miscommunications and costly change orders down the road.

3.3.2 PHASE 2

Once the project is scoped out, it's time to jump right into development, building out the user design through various prototype iterations.

During this phase, clients work hand in hand with web application programmers to ensure their needs are being met at every step in the design process. It's almost like customizable software development where the users can test each prototype of the product, at each stage, to ensure it meets their expectations.

All the bugs and kinks are worked out in an iterative process. The web application programmer designs a prototype, the client (user) tests it, and then they come together to communicate on what worked and what didn't.

This method gives web application programmers the opportunity to tweak the model as they go until they reach a satisfactory design. Both the software web application programmers and the clients learn from the experience to make sure there is no potential for something to slip through the cracks.

3.3.3 PHASE 3

Phase 3 takes the prototypes and beta systems from the design phase and converts them into the working model. Because the majority of the problems and changes were addressed during the thorough iterative design phase, web application programmers can construct the final working model more quickly than they could by following a traditional project management approach.

The phase breaks down into several smaller steps:

- Preparation for rapid construction
- Program and application development
- Coding
- Unit, integration, and system testing

The software development team of programmers, coders, testers, and web application programmers work together during this stage to make sure everything is working smoothly and that the end result satisfies the client's expectations and objectives.

This third phase is important because the client still gets to give input throughout the process. They can suggest alterations, changes, or even new ideas that can solve problems as they arise.

3.3.4 PHASE 4

This is the implementation phase where the finished product goes to launch. It includes data conversion, testing, and changeover to the new system, as well as user training. All final changes are made while the coders and clients continue to look for bugs in the system.

3.3.5 ADVANTAGES OF RAD METHODOLOGY

- RAD lets you break the project down into smaller, more manageable tasks.
- The task-oriented structure allows project managers to optimize their team's efficiency by assigning tasks according to members' specialties and experience.
- Clients get a working product delivered in a shorter time frame.
- Regular communication and constant feedback between team members and stakeholders increases the efficiency of the design and build process.

3.4 SOFTWARE TESTING

Testing documentation is the documentation of artifacts that are created during or before the testing of a software application. Documentation reflects the importance of processes for the customer, individual and organization. Projects which contain all documents have a high level of maturity. Careful documentation can save the time, efforts and wealth of the organization.

If the testing or development team gets software that is not working correctly and developed by someone else, so to find the error, the team will first

need a document. Now, if the documents are available then the team will quickly find out the cause of the error by examining documentation. But, if the documents are not available then the tester need to do black box and white box testing again, which will waste the time and money of the organization. More than that, Lack of documentation becomes a problem for acceptance.

This is one of the most cost effective approaches to testing. If the documentation is not right: there will be major and costly problems. The documentation can be tested in a number of different ways to many different degrees of complexity. These range from running the documents through a spelling and grammar checking device, to manually reviewing the documentation to remove any ambiguity or inconsistency. Documentation testing can start at the very beginning of the software process and hence save large amounts of money, since the earlier a defect is found the less it will cost to be fixed. The most popular testing documentation files are test reports, plans, and checklists. These documents are used to outline the team's workload and keep track of the process.

Let's take a look at the key requirements for these files and see how they contribute to the process.

1. Test Strategy- An outline of the full approach to product testing. As the project moves along, developers, designers, product owners can come back to the document and see if the actual performance corresponds to the planned activities.
2. Test data- The data that testers enter into the software to verify certain features and their outputs. Examples of such data can be fake user profiles, statistics, media content, similar to files that would be uploaded by an end-user in a ready solution.
3. Test plans-A file that describes the strategy, resources, environment, limitations, and schedule of the testing process. It's the fullest testing document, essential for informed planning. Such a document is distributed between team members and shared with all stakeholders.
4. Test scenarios-In scenarios, testers break down the product's functionality and interface by modules and provide real-time status updates at all testing stages.

A module can be described by a single statement, or require hundreds of statuses, depending on its size and scope.

5. Test cases-If the test scenario describes the object of testing (what), a scenario describes a procedure (how). These files cover step-by-step guidance, detailed conditions, and current inputs of a testing task. Test cases have their own kinds that depend on the type of testing —functional, UI, physical, logical cases, etc. Test cases compare available resources and current conditions with desired outcomes and determine if the functionality can be released or not.
6. Traceability Matrix-This software testing documentation maps test cases and their requirements. All entries have their custom IDs —team members and stakeholders can track the progress of any tasks by simply entering its ID to the search.
7. External reports —these documents collect information on test results and can describe an entire project or a particular piece of functionality.
8. Test summary report —the file with final test results and findings, presented to stakeholders
9. Bug reports —such files keep track of newly encountered bugs and their fixes. We prefer to keep our bug documentation numbered, so it's easier to mention them in further documentation. Reports are concise and focus on offering tangible solutions. Sometimes, bug reports can only include issue description, if the team hasn't yet found the best approach to fixing the problem

3.5 TEST DRIVEN DEVELOPMENT

Test Driven Development, or TDD, is a code design technique where the programmer writes a test before any production code, and then writes the code that will make that test pass. The idea is that with a tiny bit of assurance from that initial test, the programmer can feel free to re-factor and re-factor some more to get the cleanest code they know how to write. The idea is simple, but like most simple things, the execution is hard.

Functional Testing types include:

- Unit Testing
- Integration Testing
- System Testing
- Sanity Testing.
- Smoke Testing.
- Interface Testing
- Regression Testing
- Beta/Acceptance Testing

3.5.1 UNIT TESTING

Unit testing is a level of software testing where individual units/ components of a software are tested. The purpose is to validate that each unit of the software performs as designed. A unit is the smallest testable part of any software. It usually has one or a few inputs and usually a single output. In procedural programming, a unit may be an individual program, function, procedure, etc. In object-oriented programming, the smallest unit is a method, which may belong to a base/ super class, abstract class or derived/ child class. Unit testing frameworks, drivers, stubs, and mock/ fake objects are used to assist in unit testing.

A unit can be almost anything you want it to be --a line of code, a method, or a class. Generally though, smaller is better. Smaller tests give you a much more granular view of how your code is performing. There is also the practical aspect that when you test very small units, your tests can be run fast; like a thousand tests in a second fast.

3.5.2 INTERGRATION TESTING

Integration Testing is a level of software testing where individual units are combined and tested as a group. The purpose of this level of testing is to expose faults in the interaction between integrated units. Test drivers and test stubs are used to assist in Integration Testing.

Integration testing: Testing performed to expose defects in the interfaces and in the interactions between integrated components or systems. See also component integration testing, system integration testing.

Component integration testing: Testing performed to expose defects in the interfaces and interaction between integrated components. System integration testing: Testing the integration of systems and packages; testing interfaces to external organizations (e.g. Electronic Data Interchange, Internet).

Integration tests determine if independently developed units of software work correctly when they are connected to each other. Integration testing tests integration or interfaces between components, interactions to different parts of the system such as an operating system, file system and hardware or interfaces between systems. Integration testing is a key aspect of software testing.

3.5.3 SYSTEM TESTING

System testing is a level of software testing where a complete and integrated software is tested. The purpose of this test is to evaluate the system's compliance with the specified requirements. System Testing means testing the system as a whole. All the modules/components are integrated in order to verify if the system works as expected or not. System testing is a method of monitoring and assessing the behavior of the complete and fully-integrated software product or system, on the basis of pre-decided specifications and functional requirements. It's comes under black box testing i.e. only external working features of the software are evaluated during this testing. It does not requires any internal knowledge of the coding, programming, design, etc., and is completely based on users-perspective.

A black box testing type, system testing is the first testing technique that carries out the task of testing a software product as a whole. This system testing

tests the integrated system and validates whether it meets the specified requirements of the client.

System testing is a process of testing the entire system that is fully functional, in order to ensure the system is bound to all the requirements provided by the client in the form of the functional specification or system specification documentation. In most cases, it is done next to the Integration testing, as this testing should be covering the end-to-end system's actual routine. This type of testing requires a dedicated Test Plan and other test documentation derived from the system specification document that should cover both software and hardware requirements. By this test, we uncover the errors. It ensures that all the system works as expected. We check System performance and functionality to get a quality product. System testing is nothing but testing the system as a whole. This testing checks complete end-to-end scenario as per the customer's point of view. Functional and Non-Functional tests also done by System testing.

3.5.4 SANITY TESTING

The main purpose of this testing is to determine that the changes or the proposed functionality are working as expected. If the sanity test fails, the build is rejected by the testing team to save time and money. It is performed only after the build has cleared the smoke test and been accepted by the Quality Assurance team for further testing. The focus of the team during this testing process is to validate the functionality of the application and not detailed testing. Let's assume that in an e-commerce project, there are mainly five modules-like login page, home page, user profile page, new user creation, etc. There is a defect in the login page when the password field accepts less than four alpha numeric characters and the requirement mentions that this password field should not be below eight characters. So, the bug is reported by the testing team to the development team to resolve it. When the development team fixes the defect and passes it to the testing team. Then the testing team checks if the changes done are working fine. It also checks if it does

not have an impact on other related functionalities. Now there is a functionality to update the password in the user profile page. As part of the sanity check, we would need to validate the login page as well as the profile page to ensure that the checks are working fine at both the places. Sanity tests are generally performed on build where the production deployment is required immediately -like a critical bug fix.

3.5.5 REGRESSION TESTING

Regression Testing is a type of testing that is done to verify that a code change in the software does not impact the existing functionality of the product. This is to make sure the product works fine with new functionality, bug fixes or any change in the existing feature. Previously executed test cases are re-executed in order to verify the impact of change.

Regression Testing is a Software Testing type in which test cases are re-executed in order to check whether the previous functionality of the application is working fine and the new changes have not introduced any new bugs. This test can be performed on a new build when there is a significant change in the original functionality that too even in a single bug fix.

For regression testing to be effective, it needs to be seen as one part of a comprehensive testing methodology that is cost-effective and efficient while still incorporating enough variety—such as well-designed frontend UI automated tests alongside targeted unit testing, based on smart risk prioritization—to prevent any aspects of your software applications from going unchecked. These days, many Agile work environments employing workflow practices such as XP (Extreme Programming), RUP (Rational Unified Process), or Scrum appreciate regression testing as an essential aspect of a dynamic, iterative development and deployment schedule.

3.6 PERFORMANCE TESTING

Performance testing gathers all the tests that verify an application's speed, robustness, reliability, and correct sizing. It examines several indicators such as a browser, page and network response times, server query processing time, number

of acceptable concurrent users architected, CPU memory consumption, and number/type of errors which may be encountered when using an application. Performance testing is the testing that is performed to ascertain how the components of a system are performing under a certain given situation. Resource usage, scalability, and reliability of the product are also validated under this testing.

This testing is the subset of performance engineering, which is focused on addressing performance issues in the design and architecture of a software product. Software Performance testing is type of testing perform to determine the performance of system to major the measure, validate or verify quality attributes of the system like responsiveness, Speed, Scalability, Stability under variety of load conditions. The system is tested under a mixture of load conditions and check the time required responding by the system under varying workloads.

Software performance testing involves the testing of application under test to ensure that application is working as expected under variety of load conditions. The goal of performance testing is not only find the bugs in the system but also eliminate the performance bottlenecks from the system.

Load Testing is type of performance testing to check system with constantly increasing the load on the system until the time load is reaches to its threshold value. Here Increasing load means increasing number of concurrent users, transactions & check the behavior of application under test. It is normally carried out underneath controlled environment in order to distinguish between two different systems. It is also called as “Endurance testing” and “Volume testing”.

The main purpose of load testing is to monitor the response time and staying power of application when system is performing well under heavy load. Load testing comes under the Non Functional Testing & it is designed to test the non-functional requirements of a software application. Load testing is perform to make sure that what amount of load can be withstand the application under test. The successfully executed load testing is only if the specified test cases are executed without any error in allocated time.

Simple examples of load testing:

- Testing printer by sending large job.

- Editing a very large document for testing of word processor
- Continuously reading and writing data into hard disk.
- Running multiple applications simultaneously on server.
- Testing of mail server by accessing thousands of mailboxes
- In case of zero-volume testing & system fed with zero load.

CHAPTER 4

SYSTEM DESIGN & METHODOLOGY

4.1 EXISTING SYSTEM

The existing system presents a blind image deblurring method based on a computationally efficient and effective image regularizer. The proposed regularizer is motivated by the fact that the success of recent priors mainly stems from their properties, which implicitly generate an unnatural latent image suppressing insignificant structures and preserving only salient edges. These salient edges guide the models to estimate an accurate kernel. The existing system regularizer termed smoothing-enhancing regularizer, not only assures that only salient structures in the image are preserved but also enhances these salient structures to help the model estimate the more accurate kernel. To efficiently solve the existing system model, we develop an efficient numerical approach based on the half-quadratic splitting algorithm and the lagged-fixed-point iteration scheme. The optimization scheme only requires a few additional shrinkage operations compared with the original half-quadratic splitting algorithm, making our method much faster than recent leading methods. The qualitative and quantitative experimental results show that our algorithm achieves the state-of-the-art results and can be extended to other challenging deblurring tasks, such as those involving text, face, and low-illuminated images.

The subtle image structures are harmful to kernel estimation, thus the unnatural latent images are expected to contain only high-contrast structures. Note that b_x and b_y are actually the targets for u_x and u_y , hence we can easily remove the subtle image structures via hard thresholding.

Once the kernel k for the finest level is obtained, numerous non-blind deconvolution algorithms can be used to recover the final clear image from f . As we focus on kernel estimation, we use the non-blind deconvolution algorithm to recover the original sharp image with the estimated kernel. Explicit edge selection methods usually employ ad-hoc operators such as the shock filter to

enhance step-like edges. The shock filter renders all selected strong edges step-like, which does not always hold for real images. In contrast, the proposed regularizer adaptively enhances strong edges based on the gradient magnitude, resulting in more reliable results. Moreover, the proposed model inherently incorporates strong edge selection and enhancement into one objective, hence it does not encounter the edge location problem, which is inherent in the shock filter.

4.1.1 DRAWBACKS OF EXISTING SYSTEM

- Scalability is an issue for those designs as the amount of data grow.
- Complexity of its Real Time Implementation
- Prohibitively Expensive in terms of Time and Memory Usage.
- Not Adaptable to Large Samples.

4.2 PROPOSED SYSTEM

In the proposed system interpolation is generally used to upsample low-resolution images to the target resolution, and then nonlinear networks are used to calculate super resolution results. Because network reasoning is performed on high-resolution images, such methods have a large computational overhead. Another type of algorithm calculates the nonlinear mapping of the image on a low-resolution scale and then uses deconvolution, pixel shuffle or other techniques to upsample the results to a high resolution level.

The proposed system is based on pure Inception variant without any residual connections. It can be trained without partitioning the replicas, with memory optimization to back propagation. To prevent the middle part of the network from “dying out”, the authors introduced two auxiliary classifiers.

The proposed system minimizes the modifications to the nonlinear networks and only add a few structures to their input and output ports. The nonlinear networks used in image restoration generally contain skip connections with different densities, possibly containing batch normalization, gate units and other elements.

4.2.1 ADVANTAGES OF PROPOSED SYSTEM

- Outstanding Learning Capabilities.
- Effectively improve prediction accuracy.
- Reduces the consumption of hardware resources.
- Computational Complexity is significantly reduced.
- Boost the Performance.

4.3 REQUIREMENT SPECIFICATION

This proposed system runs effectively on a computing system that has the minimum requirements

4.3.1 HARDWARE REQUIREMENT

The basic hardware required to run the program are:

- i. Hard disk of 32 GB or more.
- ii. System memory (RAM) of 512 MB.
- iii. 64 bit i3 processor-based computer or higher with 1.4 GHz clock speed.
- iv. (800 × 600) Capable video adapter and monitor for display.

4.3.2 SOFTWARE REQUIREMENT

The basic software requirements to run the program are:

- i. Windows 7, Windows 10.
- ii. Python, Numpy, Sci-learn, Eclipse IDE /Jupyter Notebook.
- iii. Web Technologies, Bootstrap.
- iv. Browser e.g. Mozilla Firefox, Safari, chrome, etc.

4.4 SYSTEM DESIGN

A UML diagram is a diagram based on the UML (Unified Modelling Language) with the purpose of visually representing a system along with its main actors, roles, actions, artifacts or classes, in order to better understand, alter, maintain, or document information about the system.

Mainly, UML has been used as a general-purpose modelling language in the field of software engineering. However, it has now found its way into

the documentation of several business processes or workflows. For example, activity diagrams, a type of UML diagram, can be used as a replacement for flowcharts. They provide both a more standardized way of modelling workflows as well as a wider range of features to improve readability and efficiency.

4.4.1 ARCHITECTURE DIAGRAM

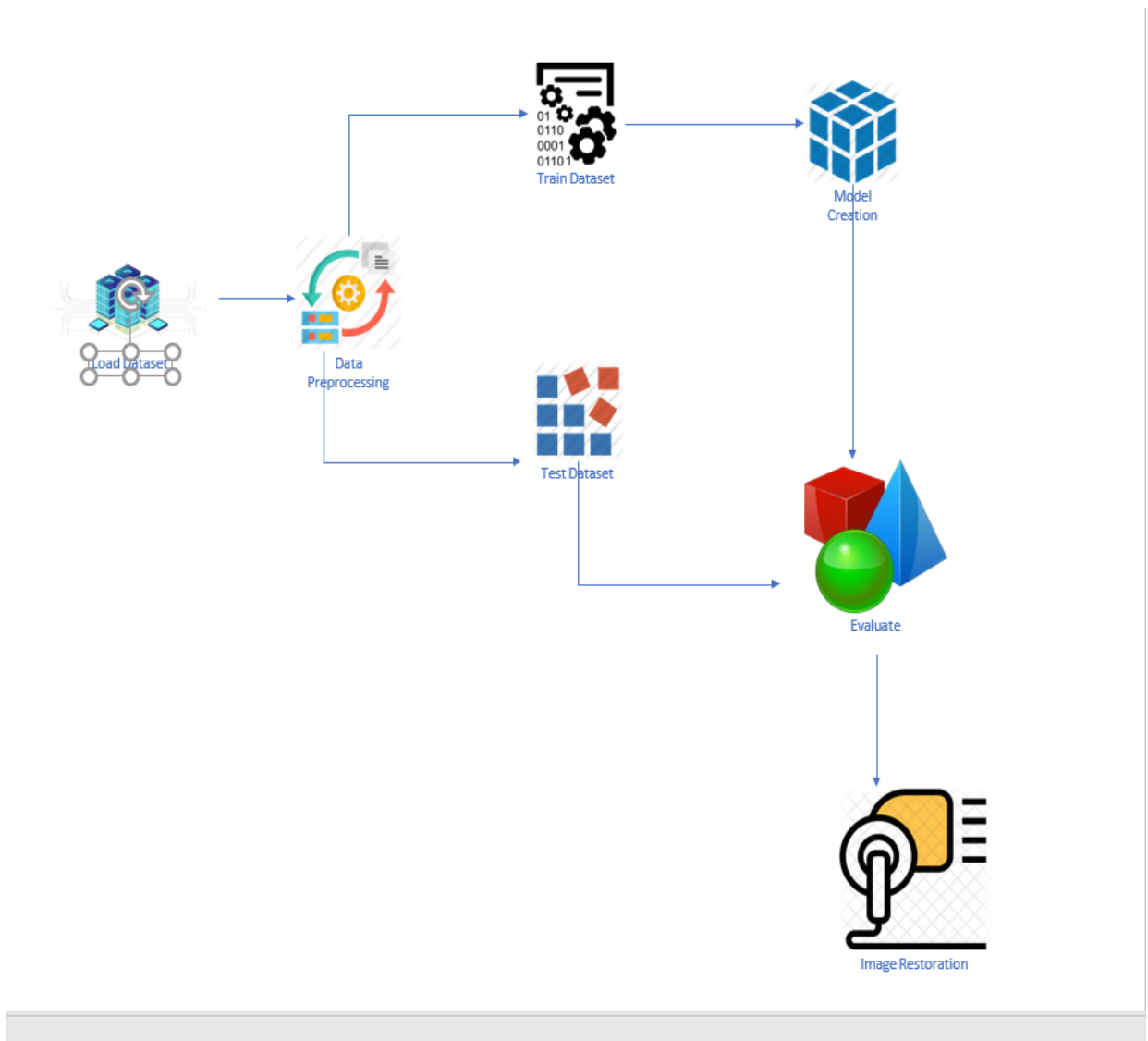


Fig 3.4.1: BLOCK DIAGRAM

4.5 SYSTEM IMPLEMENTATION

4.5.1 LIST OF MODULES

- Module 1: Data Processing
- Module 2: Load Weights
- Module 3: Prediction

4.5.2 DATA PROCESSING

Tqdm package is one of the more comprehensive packages for progress bars with python and is handy for those instances you want to build scripts that keep the users informed on the status of your application. Tqdm works on any platform (Linux, Windows, Mac, FreeBSD, NetBSD, Solaris/SunOS) in any console or in a GUI, and is also friendly with IPython/Jupyter notebooks.

The train-test split procedure is appropriate when you have a very large dataset, a costly model to train, or require a good estimate of model performance quickly. The procedure involves taking a dataset and dividing it into two subsets. The first subset is used to fit the model and is referred to as the training dataset. The second subset is not used to train the model; Instead, the input element of the dataset is provided to the model, then predictions are made and compared to the expected values. This second dataset is referred to as the test dataset.

Train Dataset: Used to fit the machine learning model.

Test Dataset: Used to evaluate the fit machine learning model.

The objective is to estimate the performance of the machine learning model on new data: data not used to train the model. By default, the program ignores the original order of data. It randomly picks data to form the training and test set, which is usually a desirable feature in real-world applications to avoid possible artifacts existing in the data preparation process. To disable this feature, simply set the shuffle parameter as False (default =True).

The skimage.io image package is used to read the image from the file. Rescale operation resizes an image by a given scaling factor. The scaling factor can either be a single floating point value, or multiple values -one along each axis. Resize serves the same purpose, but allows to specify an output image

shape instead of a scaling factor.

4.5.3 LOAD WEIGHTS

Transfer learning is a very powerful deep learning technique which has more applications in different domains. ResNet and Inception have been central to the largest advances in image recognition performance in recent years, with very good performance at a relatively low computational cost. Inception-ResNet combines the Inception architecture, with residual connections. In Residual networks, the layers of a neural network are not restricted to sequential order, but form a graph instead.

A residual block consists of two or three sequential convolutional layers and a separate parallel identity (repeater) shortcut connection, which connects the input of the first layer and the output of the last one. Each block has two parallel paths. The left path is similar to the other networks, and consists of sequential convolutional layers + batch normalization. The right path contains the identity shortcut connection (also known as skip connection). The two paths are merged via an element-wise sum. That is, the left and right tensors have the same shape and an element of the first tensor is added to the element of the same position of the second tensor. The output is a single tensor with the same shape as the input. In effect, we propagate forward the features learned by the block, but also the original unmodified signal. The network can decide to skip some of the convolutional layers thanks to the skip connections, in effect reducing its own depth. The residual blocks use padding in such a way that the input and the output of the block have the same dimensions.

An inception block starts with a common input, and then splits it into different parallel paths (or towers). Each path contains either convolutional layers with a different-sized filter, or a pooling layer. In this way, we apply different receptive fields on the same input data. At the end of the inception block, the outputs of the different paths are concatenated.

4.5.4 PREDICTION

Image Data Generator class allows allow rotation of upto 90 degrees,

horizontal flip, horizontal and vertical shift of the data. We need to apply the training standardization over the test set. Image Data Generator will generate a stream of augmented images during training.

We will define Exponential Linear Unit (ELU) activation functions. A single fully-connected layer after the last max pooling. The padding='same' parameter. This simply means that the output volume slices will have the same dimensions as the input ones.

Batch normalization provides a way to apply data processing, similar to the standard score, for the hidden layers of the network. It normalizes the outputs of the hidden layer for each mini-batch (hence the name) in a way, which maintains its mean activation value close to 0, and its standard deviation close to 1. We can use it with both convolutional and fully connected layers. Networks with batch normalization train faster and can use higher learning rates.

CHAPTER 5

RESULTS AND DISCUSSION

In the proposed model the images are trained in two ways as training data and test data. The training data is done for 80% and test data for 20%.

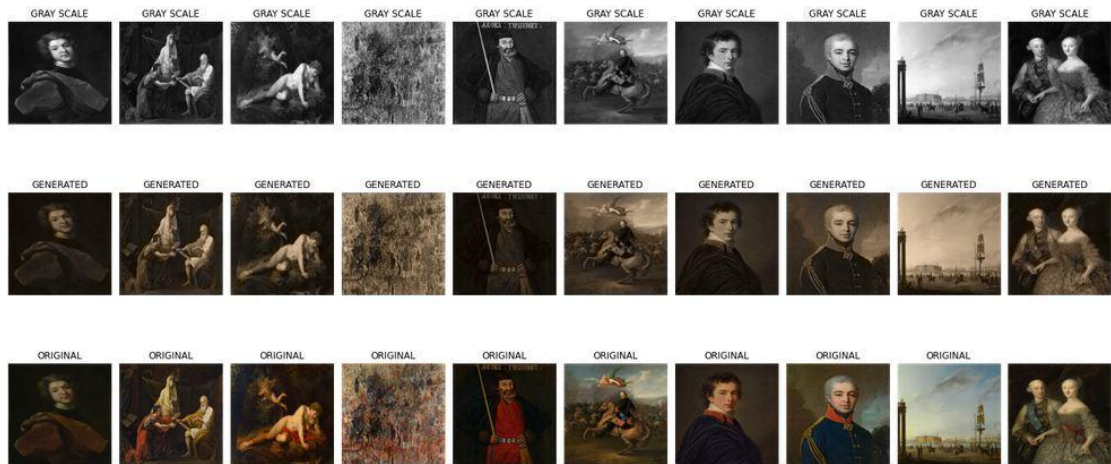


Fig 5.1 TRAIN DATA

The Fig. 5.1 represents the output of the selected images which are obtained by training from a large dataset of paintings and drawings.

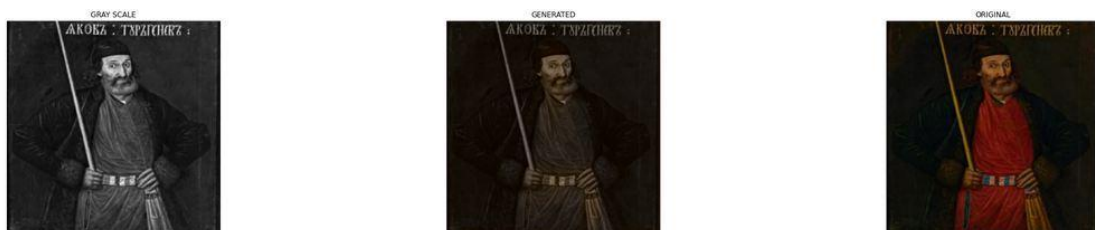


Fig 5.2 TEST DATA (1)

The Fig. 5.2 represents the output of the image obtained by the test dataset



Fig 5.3 TEST DATA (2)

The Fig. 5.3 is the output generated based on the train data and test data.

CHAPTER 6

CONCLUSION AND FUTURE WORK

6.1 CONCLUSION

The results obtained from the beginning our model is looking realistic as compared to the existing works. Thus, the model constructed using convolutional neural networks is performing satisfactorily as well as it increases the efficiency of sharpening the blur image and reduced the training time of the model as compared to the developed models like lstm .The proposed model takes sequence of blurry images in different resolutions and generates latent sharp image at each layer and provides the sharp image at full resolution by converting back into the original format. Thus, it reduces the time complexity and gives stable results. Compared with the existing two-frame architectures, the multi frame architecture of can avoid repeated computations caused by multiple inferences when aligning multiple images. We applied logic to image denoising, image superresolution and superresolution tasks.

6.2 FUTUREWORK

In future, developing image deblurring with Bi-Directional LSTM Approach. In bi-directional lstm model the data can transfer from past-lstms to future lstms and future lstms to past-ltms, So that it can develop a images blur image to high resolution image and also convert image in to sentence .In this approach texts in the images also can occur in good resolution without any blur. same way this model can used for old videos also it can deblur the texts in videos and it can help the people to see the old videos in high resolution without blur.

REFERENCES

- [1] Z. Zha, X. Yuan, B. Wen, J. Zhang, J. Zhou, and C. Zhu, “Image restoration using joint patchgroup-based sparse representation,” *IEEE Transactions on Image Processing*, vol. 29, pp. 7735–7750, 2020.
- [2] J. Yang, J. Wright, T. S. Huang, and Y. Ma, “Image super-resolution via sparse representation,” *IEEE Transactions on Image Processing*, vol. 19, no. 11, pp. 2861–2873, Nov 2010.
- [3] W. Dong, L. Zhang, G. Shi, and X. Li, “Nonlocally centralized sparse representation for image restoration,” *IEEE Transactions on Image Processing*, vol. 22, no. 4, pp. 1620–1630, April 2013.
- [4] Z. Zha, X. Yuan, B. Wen, J. Zhang, J. Zhou, and C. Zhu, “Simultaneous nonlocal self- similarity prior for image denoising,” in *2019 IEEE International Conference on Image Processing (ICIP)*, Sep. 2019, pp. 1119–1123.
- [5] B. Wen, S. Ravishankar, and Y. Bresler, “Structured overcomplete sparsifying transform learning with convergence guarantees and applications,” *International Journal of Computer Vision*, vol. 114, no. 2-3, pp. 137–167, 2015.
- [6] J. Xu, L. Zhang, W. Zuo, D. Zhang, and X. Feng, “Patch group based nonlocal self-similarity prior learning for image denoising,” in *2015 IEEE International Conference on Computer Vision (ICCV)*, Dec 2015, pp. 244–252.
- [7] Z. Zha, X. Yuan, B. Wen, J. Zhou, J. Zhang, and C. Zhu, “From rank estimation to rank approximation: Rank residual constraint for image restoration,” *IEEE Transactions on Image Processing*, vol. 29, pp. 3254–3269, 2020.
- [8] X. Jia, S. Liu, X. Feng, and L. Zhang, “Focnet: A fractional optimal control network for image denoising,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 6054–6063.
- [9] Z. Zha, B. Wen, X. Yuan, J. Zhou, and C. Zhu, “Reconciliation of group sparsity and low-rank models for image restoration,” in *2020 IEEE International*

Conference on Multimedia and Expo (ICME). IEEE, 2020, pp. 1–6.

[10] S. Li, Y. Chen, R. Jiang, and X. Tian, “Image denoising via multi-scale gated fusion network,” *IEEE Access*, vol. 7, pp. 49392–49402, 2019. P. Liu, Y. Hong, and Y. Liu, “Deep differential convolutional network for single image superresolution,” *IEEE Access*, vol. 7, pp. 37555–37564, 2019.

[11] V. Itier, F. Kucharczak, O. Strauss, and W. Puech, “Interval-valued JPEG decompression for artifact suppression,” in *Proc. 8th Int. Conf. Image Process. Theory, Tools Appl. (IPTA)*, Nov. 2018, pp. 1–6.

[12] Z. Dou, K. Gao, X. Zhang, and H. Wang, “Fast blind image deblurring using smoothing- enhancing regularizer,” *IEEE Access*, vol. 7, pp. 90904–90915, 2019

APPENDIX

A) SOURCE CODE

```
def image_deblurring():  
  
    embed_input = Input(shape=(1000,))  
  
  
    encoder_input = Input(shape=(256, 256, 1,))  
  
    encoder_output = Conv2D(128, (3,3), activation='relu',  
padding='same',strides=1)(encoder_input)  
  
    encoder_output = MaxPooling2D((2, 2), padding='same')(encoder_output)  
  
    encoder_output = Conv2D(128, (4,4), activation='relu',  
padding='same')(encoder_output)  
  
    encoder_output = Conv2D(128, (3,3), activation='relu',  
padding='same',strides=1)(encoder_output)  
  
    encoder_output = MaxPooling2D((2, 2), padding='same')(encoder_output)  
  
    encoder_output = Conv2D(256, (4,4), activation='relu',  
padding='same')(encoder_output)  
  
    encoder_output = Conv2D(256, (3,3), activation='relu',  
padding='same',strides=1)(encoder_output)  
  
    encoder_output = MaxPooling2D((2, 2), padding='same')(encoder_output)  
  
    encoder_output = Conv2D(256, (4,4), activation='relu',  
padding='same')(encoder_output)  
  
    encoder_output = Conv2D(256, (3,3), activation='relu',
```

```

padding='same')(encoder_output)

encoder_output = Conv2D(256, (3,3), activation='relu',
padding='same')(encoder_output)


fusion_output = RepeatVector(32 * 32)(embed_input)
fusion_output = Reshape([32, 32, 1000])(fusion_output)
fusion_output = concatenate([encoder_output, fusion_output], axis=3)
fusion_output = Conv2D(256, (1, 1), activation='relu',
padding='same')(fusion_output)


decoder_output = Conv2D(128, (3,3), activation='relu',
padding='same')(fusion_output)

decoder_output = Conv2D(64, (3,3), activation='relu',
padding='same')(decoder_output)

decoder_output = UpSampling2D((2, 2))(decoder_output)

decoder_output = Conv2D(128, (3,3), activation='relu',
padding='same')(decoder_output)

decoder_output = UpSampling2D((2, 2))(decoder_output)

decoder_output = Conv2D(64, (4,4), activation='relu',
padding='same')(decoder_output)

decoder_output = Conv2D(64, (3,3), activation='relu',
padding='same')(decoder_output)

decoder_output = Conv2D(32, (2,2), activation='relu',
padding='same')(decoder_output)

decoder_output = Conv2D(2, (3, 3), activation='tanh',
padding='same')(decoder_output)

decoder_output = UpSampling2D((2, 2))(decoder_output)

```

```

return Model(inputs=[encoder_input, embed_input], outputs=decoder_output)

model = image_deblurring()

model.compile(optimizer='adam', loss='mean_squared_error')

model.summary()

plt.figure(figsize=(20, 10))

for i in range(10):

    # grayscale

    plt.title("ORIGINAL")

    plt.subplot(3, 10, i + 1)

    plt.imshow(rgb2gray(x_test)[i].reshape(256, 256))

    plt.gray()

    plt.axis('off')


    # recolorization

    plt.title("GRAY SCALE")

    plt.subplot(3, 10, i + 1 + 10)

    plt.imshow(decoded_imgs[i].reshape(256, 256, 3))

    plt.axis('off')


    # original

    plt.title("GENERATED")

    plt.subplot(3, 10, i + 1 + 20)

    plt.imshow(x_test[i].reshape(256, 256, 3))

    plt.axis('off')

```

```

plt.tight_layout()

plt.show()

i=4

plt.title("RESULTS")

plt.axis('off')

plt.figure(figsize=(50,10))


plt.subplot(1, 10,1)

plt.imshow(rgb2gray(x_test)[i].reshape(256, 256))

plt.gray()

plt.axis('off')


# recolorization

plt.title("GRAY SCALE")

plt.subplot(1, 10,3)

plt.imshow(decoded_imgs[i].reshape(256, 256,3))

plt.axis('off')


# original

plt.title("GENERATED")

plt.subplot(1, 10,5)

plt.imshow(x_test[i].reshape(256, 256,3))

plt.axis('off')

```

```
plt.title("ORIGINAL")
```

```
plt.tight_layout()
```

```
plt.show()
```

```
plt.title("RESULTS")
```

```
plt.axis('off')
```

```
plt.figure(figsize=(50,10))
```

```
plt.subplot(1, 10,1)
```

```
plt.imshow(X_test[0])
```

```
plt.gray()
```

```
plt.axis('off')
```

```
plt.title("INPUT")
```

```
plt.subplot(1, 10,3)
```

```
plt.imshow(decoded_imgs[0].reshape(256, 256,3))
```

```
plt.axis('off')
```

```
plt.title("OUTPUT")
```

```
plt.tight_layout()
```

```
plt.show()
```

