A PROJECT REPORT ON

A ROBUST MOBILE APPLICATION TO CAPTURE IMAGES IN LOW ILLUMINATION AND TRANSFORM THEM TO CORRESPONDING HIGH QUALITY IMAGES

SUBMITTED TO THE SAVITRIBAI PHULE PUNE UNIVERSITY, PUNE
IN THE PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE AWARD OF THE DEGREE

OF

BACHELOR OF ENGINEERING (COMPUTER ENGINEERING)

SUBMITTED BY

DIPAK BANGE	**Exam No :B150334209**
ABHISHEK GAIKWAD	**Exam No :B150334247**
TEJAS GAJARE	**Exam No :B150334248**
ADITYA KHADSE	**Exam No :B150334293**

DEPARTMENT OF COMPUTER ENGINEERING

PCET'S PIMPRI CHINCHWAD COLLEGE OF ENGINEERING

Sector No. 26, Pradhikaran, Nigdi, Pimpri-Chinchwad, PUNE 411044

SAVITRIBAI PHULE PUNE UNIVERSITY
2018 -2019

CERTIFICATE

This is to certify that the project report entitled

A ROBUST MOBILE APPLICATION TO CAPTURE IMAGES IN LOW ILLUMINATION AND TRANSFORM THEM TO CORRESPONDING HIGH QUALITY IMAGES

Submitted by

DIPAK BANGE	**Exam No :B150334209**
ABHISHEK GAIKWAD	**Exam No :B150334247**
TEJAS GAJARE	**Exam No :B150334248**
ADITYA KHADSE	**Exam No :B150334293**

is a bonafide student of this institute and the work has been carried out by her under the supervision of **Prof. Dr. Swati Shinde** and it is approved for the partial fulfillment of the requirement of Savitribai Phule Pune University, for the award of the degree of **Bachelor of Engineering** (Computer Engineering).

(Prof. Dr. Swati Shinde)
Guide
Department of Computer Engineering

(Prof. Dr. K. Rajeswari)
Head
Department of Computer Engineering

(Prof. Dr. A. M. Fulambarkar)
Principal
Pimpri Chinchwad College of Engineering Pune – 44

Place : Pune
Date :

ACKNOWLEDGEMENT

We express our sincere thanks to our **Guide Prof. Dr. Swati Shinde** for her constant encouragement and support throughout our project, especially for the useful suggestions given during the course of project and having laid down the foundation for the success of this work.

We would also like to thank our **Project Coordinator, Prof. Archana Kadam,** for her assistance, genuine support and guidance from early stages of the project. We would like to thank **Prof. Dr. K. Rajeswari, Head of Computer Department** for her unwavering support during the entire course of this project work. We are very grateful to our **Principal, Prof. Dr. A. M. Fulambarkar** for providing us with an environment to complete our project successfully. We also thank all the staff members of our college and technicians for their help in making this project a success.

We also thank all the web committees for enriching us with their immense knowledge. Finally, we take this opportunity to extend our deep appreciation to our family and friends, for all that they meant to us during the crucial times of the completion of our project.

NAME OF STUDENTS

DIPAK BANGE	**Exam No :B150334209**
ABHISHEK GAIKWAD	**Exam No :B150334247**
TEJAS GAJARE	**Exam No :B150334248**
ADITYA KHADSE	**Exam No :B150334293**

ABSTRACT

Mobile phones have become a necessity in the modern world. One of its most prominent features is the camera. Mobile phone cameras are on the way to completely replace other forms of camera due to their sheer power. Millions of images are captured on mobile devices across the globe. These images are clear and crisp. But all these images are captured in daylight. Images taken in low illumination essentially turn out to be too dark to be comprehensible.

Research shows that current solutions to this problem work for dim to moderate light level but fail in extreme low light. There are certain problems involved with these techniques. Firstly, Image denoising relies on image priors. This limits the situations that image denoising will work on. Other deep learning techniques work on synthetic data and cannot be proficient on real data. Secondly, Low light image enhancement assumes that images already contain a good representation of scene content.

This project proposes to capture low illumination images and transform them to high quality images using end to end fully convolutional neural network trained on our data set of raw images shot in low aperture and their corresponding high aperture raw images. As an outcome, we will be able to transform images to high quality and identify objects.

Keywords: Computer Vision, Machine Learning, Artificial Intelligence, Image Processing

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

1.1 Overview

In deep learning, a convolutional neural network [1] (CNN, or ConvNet) is a class of deep neural networks, most commonly applied to analyzing visual imagery. We in our project are using this technique of deep learning to tackle a problem of capturing images in extremely low light conditions and then predicting how to actual still of the image would look if light was present.

We have solved it by creating our own dataset of raw images of a scene in low exposure(input) and its corresponding high exposure raw image (ground truth) using mobile phone sensors and passed it to a fully convoluted neural network of U-net architecture. Later, we also tried to improve colors in the predicted image by using transfer learning on a pretrained model for raw images on SLR digital cameras.

1.2 Motivation

Mobile phones have become a staple for every person's needs. Further, cameras in mobile phones have started to compete against professional cameras developed only for photography. But due to limited hardware and space constraints in a mobile phone, cameras are unable to outperform the professional cameras. The sensors used in mobile phones are far inferior to the ones found in professional cameras. Of the few spaces that mobile cameras lack, is low light images. Mobile cameras have been using flash for the past few years to enable the user to capture images in low light. But the images captured do not truly see what the human eyes had intended to see. We aim to enable mobile cameras to be able to compete with the professional cameras through the use of artificial intelligence.

The techniques we aim to use are based on software and hence do not add to the cost of building the mobile phone. Our method abandons the traditional image processing pipeline to move to a more data driven pipeline that is built using examples of similar past experiences rather than the image priors, which are details of images that are known prior to the image even

being captured. Abandoning the traditional pipeline helps us to achieve a more data driven approach

1.3 Problem Definition and Objectives

There are varied instances where we cannot get a high quality image in a low illumination setting where we need to take image using different methods to capture all the details. But it is impractical to these methods for extreme low light. Current methods have certain restrictions. Taking an image with high ISO leads to amplification of noise. Scaling or using histogram stretching doesn't resolve the issue of low Signal-to-Noise Ratio (SNR). Increasing exposure leads to introduction of blur in the image, although there are some solutions for it [2]. Denoising [3] does not address color bias. Enhancement works for dim to moderate light but fails for extreme low light conditions. Burst imaging fails as alignment is not possible in low illumination. Similarly, it fails for video since lucky imaging is difficult.

It has become a requirement to develop a method that can work on single image to enhance from low illumination to a high quality image. The received image must be perceptually good image. We need a method that can generate a high exposure image using the data captured during a low exposure image. The method should be implemented within a mobile app, so that it can be easily used with a mobile phone's camera.

Therefore the problem statement is "To design and implement algorithm for capturing low illumination images and transform them to high quality images using end to end fully convolutional neural network."

1.4 Project Scope & Limitations

The project aims to move towards a research based outcome initially. We will first attempt to obtain a perceptually good image from a low illumination image captured using a mobile phone. This will be done on a computer system rather than a mobile device. With satisfactory results we will proceed to an independent mobile application that will capture the image and provide the high quality counterpart. Later we can also release an app using this technique for the public to use.

In order to reach a viable and reliable solution, we need to understand what makes an image perceptually good. The source image would be taken in extreme low light. The image, when processed through traditional pipelines, will mostly consist of black pixels and be undecipherable to human eyes. The traditional pipelines fail to work in the extreme low light, which this project aims to work upon.Traditional pipelines work in extreme low light but only in two cases: either there are multiple images of the same scene, or the images are taken with long exposure. The project aims to work without either of those that is use only one image and which has been captured with short exposure.

The project does not rely on traditional image pipelines and rather follows a pipeline based on the data obtained from the raw images captured by camera sensor [4]. To minimise loss [5] of raw data, we feed a trained neural network with one raw image with short exposure and receive a transformed image which will be decipherable to human eyes.The project will work on images which are captured in low light and not in zero light. There must be photons which can be captured by the camera sensor, else this method will fail.

1.5 Methodologies of Problem solving

We used the six step problem solving model in order to solve this problem. The six steps are as follows:

1. **Define the problem**: We want to capture clear images in extreme low light conditions. The image must be captured in a smartphone. The result must be displayed in a finite amount of time.

2. **Determine the root cause**: The root cause of bad images captured is the way images are converted from RAW to JPEG format. The process does not use all the information as provided by the RAW image.

3. **Develop alternative solutions**: The current solutions include using BM3D denoising which does not work well enough for extreme low light images. Another solution is to use a pretrained model for improving the extreme low light images, but it does not display the correct colors.

4. **Select a solution**: The best solution here would be to train a neural network, which was pretrained on low light images captured on DSLR and apply transfer learning with low light images captured on a smartphone.

5. **Implement the solution**: We have implemented the solution using Tensorflow library and an Android application to capture new unseen images that can be converted to better images.

6. **Evaluate the outcome**: The results from our model display perceptually better images than those created using previously mentioned methods.

2 LITERATURE SURVEY

There are various categories within photography, one of which is low light photography. This category involves taking images in low light, which involves scenes with lighting provided by sources such as street lights, moonlight. Various methods have been used to improve visibility of scene. Some of them process a burst of images to obtain a single image, while others process a single image end-to-end to get the result.

2.1 Basic Methods

Digital cameras have had an important feature that has been taken for granted today. They have the ability to change their ISO on the fly. ISO here refers to the camera sensor's sensitivity to light. A high ISO means the sensor would be more sensitive to light and would increase the overall brightness of the image. This is the most basic method of improving visibility of a scene in low-light images.

Another basic method is to increase the exposure of the camera sensor. It can be controlled by two parameters: shutter speed and aperture. Shutter speed is the amount of time that the shutter remains open to capture the scene. A low shutter speed means that the shutter stays open for very small amount of time. The amount of light that enters the sensor is proportional to the time that the sensor is able to capture a light. For example, an image taken with shutter speed as 1/800 seconds would be darker than an image of the same scene taken with shutter speed as 1 second. A high exposure helps to get a brighter image as more photons can be captured by the camera sensor.

A camera works similar to the human eye. There exists an opening that allows light to enter a sensor that captures the light and convert it to an image. Aperture can be compared with the size of the pupil of the human eye. It controls the amount of light that can enter and be captured on the camera's sensor. A high aperture allows more light to enter and improve visibility in low-light images. Low-light images captured this using traditional pipeline is not perceptually good as it has a high amount of amount of noise. Several denoising techniques are used to further improve the visibility within the image. One of the most used techniques is 3D transform-domain filtering (BM3D).

2.2 BM3D

In 2007, K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian [6] proposed a novel image denoising strategy based on an enhanced sparse representation in transform domain. The idea was to use 2D image fragments found using matching of reference blocks and group them into 3D data arrays. They called these 3D data arrays groups. Further, they used collaborative filtering for the 3D groups. The filtering was divided into 3 successive steps: 3D transformation of a group,

shrinkage of the transform spectrum and an inverse 3D transformation.

The estimated 2D fragments were then returned to their locations. Due to multiple matching 2D fragments, they would be required to aggregate to take advantage of the redundancy. BM3D has outperformed other recent denoising techniques when used on real low light images.BM3D fails to outperform other data driven techniques as they still rely on processed image data rather than raw sensor data. The method also relies on one image of the scene. Another method tackled low-light imaging using burst photography.

2.3 Burst Photography for high dynamic range and low-light imaging on mobile cameras

In 2016, Hasinoff et al [7] proposed a computational photography pipeline that can capture, align and merge a burst of frames to reduce noise and increase dynamic range. The pipeline takes in the burst of raw frames. It then aligns the burst to merge. Then it applies colour and tone mapping to produce a single full-resolution output. The cameras in mobile phones allow auto exposure adjustment as the user moves the camera around. To get improved results in high dynamic range, they developed their own algorithm for adjusting the auto exposure within the mobile camera. The major problem with burst [8] images is that if the scene contains a moving object, it is at a different position in every frame. It becomes difficult to find the best position for the object. Hence, burst images introduce blurred objects, or ghosting if the objects move a lot faster than the shutter speed. To overcome this, they used temporal mean with alignment and merged it robustly to achieve almost zero blur compared to the temporal mean.

2.4 Learning to See in the Dark At CVPR 2018

C Chen et al [9] proposed a pipeline for processing low-light images based on end-to-end training of a convolutional neural network on short exposure images and corresponding long exposure images. The neural network would be trained on raw sensor data rather than output images. This improves performances over other methods since this method does not rely on image priors, which are assumptions made about the image and its data prior to obtaining the image.

This method works by first training a fully convolutional neural network with short exposure images, which are captured in extreme low-light images. The architecture used in the neural network is U-net. [10] Further they used 2 cameras to form the dataset required for training the neural network. The See-in-the-Dark dataset has images from Sony α7S II and Fujifilm X-T2. There are 5094 short exposure images with their reference long exposure images.

3 SOFTWARE REQUIREMENTS SPECIFICATION

3.1 Assumptions and Dependencies

- Assumptions

  1. At least some amount of light is present in Image

     In order for the input image to be transformed, it is necessary for the image to have the photon count to be greater than zero. Our project works with low SNR [11], which is low but not zero.

  2. Images are present in DNG format

  3. The camera is able to capture images in RAW format

     Cameras capture a scene in a RAW image file before converting it to a PNG or JPEG format which are easier to view, store and distribute. The converted files are also space efficient. RAW images are rather large since they capture every detail they can, while PNG or JPEG attempt to maintain a suitable ratio of quality over space. Our method is data driven and hence requires as much as data as possible. RAW images serve this purpose the best.

- Dependencies

  1. The camera is able to capture images in RAW format

     Cameras capture a scene in a RAW image file before converting it to a PNG or JPEG format which are easier to view, store and distribute. The converted files are also space efficient. RAW images are rather large since they capture every detail they can, while PNG or JPEG attempt to maintain a suitable ratio of quality over space. Our method is data driven and hence requires as much as data as possible. RAW images serve this purpose the best.

  2. The processing computer must have atleast 8GB RAM and a NVIDIA Graphics Card with cuda compatibility > 3.0

     As the server will be running on this computer, it should have enough power to receive and process the image using the trained neural network and send the predicted image again to the phone.

  3. Wifi connection or Mobile Hotspot

     The mobile phone and the processing computer should be connected to a same wifi connection so that images can be shared with a increased speed as they are in the same network

  4. TensorFlow

We have used deep neural networks [12] and hence use of TensorFlow library to train the model and for getting further results.

5. Rawpy

   In order to work on RAW images in Python, we use rawpy module.

6. python-flask

   To create a python web server, we have used this python module.

3.2 Functional Requirements

3.2.1 RAW Image Capture Ability:

RAW image format is an important aspect of this project. It is a requirement that the camera must be able to capture and store RAW images.

3.2.2 Mobile Phone with Internet Connection:

The RAW image captured will be sent to a server from within the app. The server will be remote and hence the mobile phone must have internet connection that can send the image from the mobile phone to the server. Internet connection would also be required to obtain the output from the server and store in the mobile phone.

3.2.3 Server with 16GB of RAM:

The neural network that will take the RAW image as input and provide a high-quality output requires at least 64GB of RAM to not lose any of the details.

3.2.4 Pretrained model available on the server:

The server must have a pretrained model locally available. The pretrained model can then be loaded onto a neural network in order to work on the RAW input image.

3.2.5 RAW image in DNG format:

The RAW image obtained from the mobile phone must be in DNG format. DNG is Digital Negative and is a popular open file format created by Adobe. [13] We are using it since it is open and hence would be used by most smartphone manufacturers.

3.3 External Interface Requirements

3.3.1 User Interfaces:

1. The UI must allow the user see real time view of the RAW image being captured the camera.

2. The UI must provide facility to capture image (such as a button to click).
3. The UI must provide with settings to the user, where he can tune various details of the camera such as exposure, ISO, etc.
4. The UI must allow viewing of previously captured images.
5. The UI must acknowledge and show progress of upload when the image is being sent to a server.
6. The UI must show the final transformed image as soon as it is downloaded from the server after processing.
7. The UI must inform the user in cases of being unable to connect to the server.
8. The UI must inform the user that there is an Internet connection required to process every image.

3.3.2 Hardware Interfaces:

1. The camera used in the smartphone must be able to capture RAW images.
2. The camera used in the smartphone must be able to detect low number of photons.
3. The camera used to collect images for dataset must allow setting exposure manually.
4. The camera used to collect images for dataset must allow locking ISO to certain value.
5. The camera used to collect images for dataset must be able to capture RAW images.
6. The systems used in training of neural network must allow use of GPUs.

3.3.3 Software Interfaces:

1. The app must interact with Camera2 API provided in Android.
2. The app must store RAW images in DNG file format on the smartphone.
3. The app must be able to connect to remote server and send the RAW file over internet.
4. The app must be able to retrieve the correct output image from the server.

3.4 Nonfunctional Requirements

3.4.1 Performance Requirements:

Describes the specific performance requirements for the application

- Accuracy: The system should give 80 % accuracy ideally. The image should be perceptually clear with all objects well illuminated

- Efficiency: The system should use the computing power of the processor efficiently and the response time should be minimum.

- Robustness: The system should be stable such that any error in input should not crash the entire system and restart again from the point of failure.

3.4.2 Safety Requirements:

If there is any case in which server goes down or isn't functioning then recovery methods should be applied and server should be restarted. If image file is corrupted user should be notified on the app.

3.4.3 Security Requirements:

If connection is interrupted or is poor the user should be notified and the whole image should be sent again. Images as inputs and their corresponding outputs to be also stored at server side for further analysis. Security should be provided while transferring the image from server side to receiver side.

3.4.4 Software Quality Attributes:

- Availability: The system should be available at any time as it is real time monitoring.

- Correctness: the system should give correct prediction of the corresponding dark image.

- Maintainability: the system and android app should give correct results of the image shot under low light. Live changes and progress of operations to also be shown on terminal on server side.

- Usability: the system should consider and process all kinds of raw images shot on mobile phone.

3.5 System Requirements

3.5.1 Database Requirements:

UNIX file system – All the images (inputs and their corresponding outputs) to be stored at server side before sending.

3.5.2 Software Requirements:

- TensorFlow

- Keras

- RawPy

3.5.3 Hardware Requirements:

- GPU Clustering
- Google Cloud
- Phone Camera (click RAW images) RawPy

3.6 Analysis Models: SDLC Model to be applied

3.6.1 Agile Model:

Agile SDLC model is a combination of iterative and incremental process models which focus on process adaptability and customer satisfaction by rapid delivery of working software product. Agile Methods break the product into small incremental assembles. These builds are provided in iterations. Each iteration typically lasts from one to three weeks. Every iteration involves cross functional teams working simultaneously on various different areas like -

- Planning
- Requirements Analysis
- Design
- Coding
- Unit Testing and
- Acceptance Testing

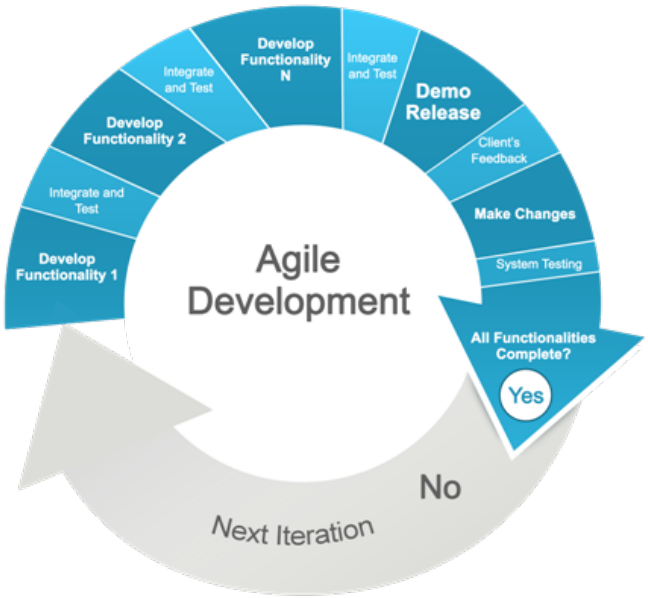


Figure 1: Agile Model

Following are the Agile Manifesto principles:

• Individuals and interactions - In Agile development, self-organization and motivation are important, as are interactions like co-location and pair programming.

• Working software - Demo working software is considered the best means of communication with the customers to understand their requirements, instead of just depending on documentation.

Advantages of Agile model:

• People and interactions are highlighted rather than process and tools.

• Face-to-face conversation is the best form of communication.

• Continuous attention to technical excellence and good design.

• Even late changes in requirements are welcomed.

• Regular adaptation to changing circumstances.

4 SYSTEM DESIGN

4.1 System Architecture

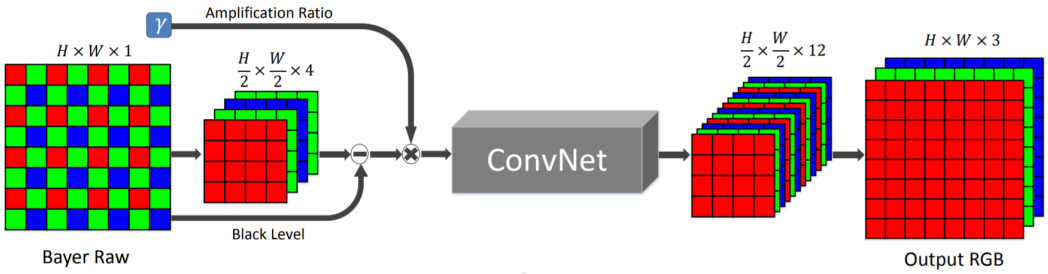


Figure 2: System Architecture

4.2 Mathematical Model

$M = \{s,\ e,\ X,\ Y,\ DD,\ NDD,\ Mem\ shared,\ Fme\}$

where,

s = Start state

Clicking a photo in low illumination

e = End state

Transformed photo with better picture quality(high illumination)

X = input

raw image from phone in .dng format which will be uploaded to the server for processing

Y = output

transformed photo received from server in .png (or .jpg) format having better picture quality in terms of illumination

DD = Deterministic data:

Result of images which are transformed to high illumination

NDD = Non-deterministic data:

Predicting the objects which are completely dark.

Mem shared:

Dataset containing short exposure images with its corresponding long exposure

Fme: Functions used.

Backpropagation

Error Function

Activation Function:

Rectifier

Adam Optimiser

Max Pooling

4.3 Data Flow Diagram

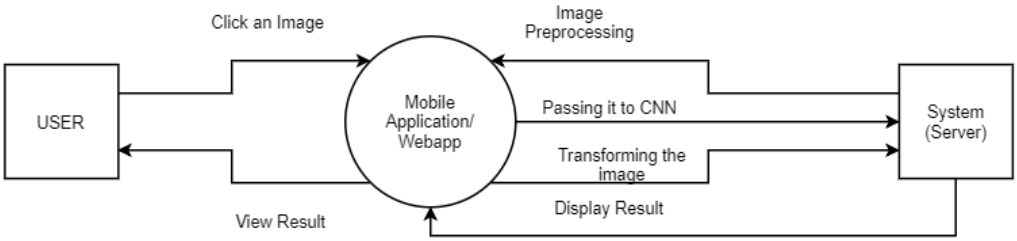


Figure 3: Application Data Flow

4.4 UML Diagrams

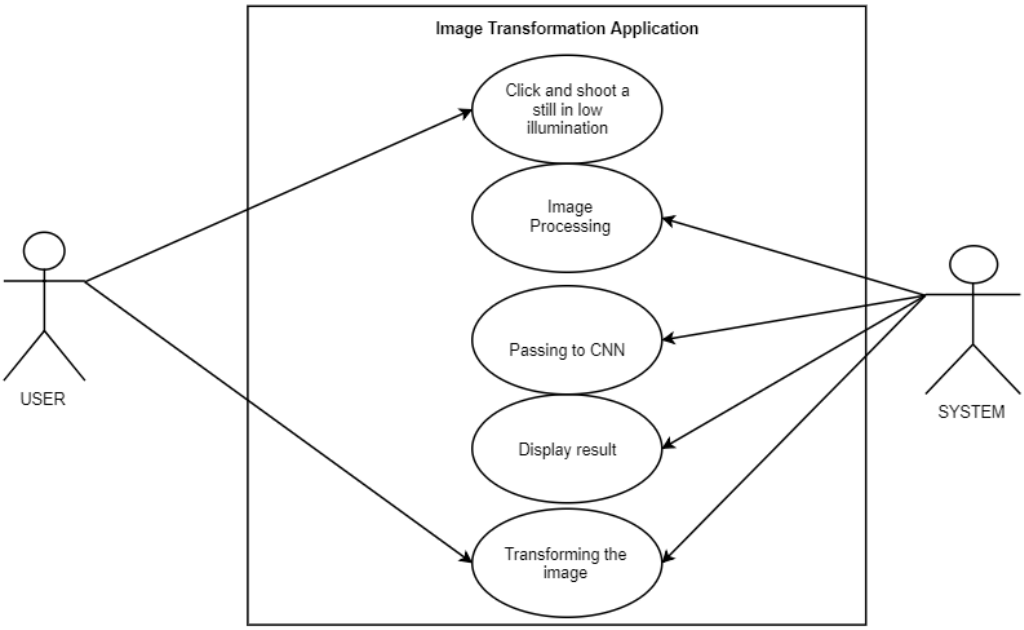


Figure 4: Application Use Case Diagram

5 PROJECT PLAN

5.1 Project Estimate

5.1.1 Reconciled Estimates

5.1.1.1 Cost Estimates

Table 1: Estimated Costs Table

Sr No	Item Description	Estimated Cost
1	Camera2 API enabled smartphone	₹15,000
2	Remote Server capable of processing image	₹60,000

5.1.1.2 Time Estimate

Phase 1

- Collection of Dataset
- Renaming of Dataset as needed

Phase 2

- Development of Neural network
- Training of previous model on collected dataset

Phase 3

- Development of Mobile Application with uploading feature
- Development of server that can accept RAW images and process them
- Testing and Validation

5.2 Project Resources

Mentors - Dr. J S Umale and Dr. Swati Shinde

Team Leader - Mr. Abhishek Gaikwad

Team Members -

1. Mr. Dipak Bange
2. Mr. Tejas Gajare
3. Mr. Aditya Khadse

For hardware, software and tools used, refer Design section

5.3 Risk Management

5.3.1 Risk Identification

For risks identification, review of scope document, requirements specifications and schedule is done. Answers to questionnaire revealed some risks. Each risk is categorized as per the categories mentioned in Pressman book. Please refer table 3.2 for all the risks. We have referred the following risk identification questionnaire:

1. Have top software and customer managers formally committed to support the project?
2. Are end-users enthusiastically committed to the project and the system/product to be built?
3. Are requirements fully understood by the software engineering team and its customers?
4. Have customers been involved fully in the definition of requirements?
5. Do end-users have realistic expectations?
6. Does the software engineering team have the right mix of skills?
7. Are project requirements stable?
8. Is the number of people on the project team adequate to do the job?
9. Do all customer/user constituencies agree on the importance of the project and on the requirements for the system/product to be built?

5.3.2 Risk Analysis

The risks for the Project can be analyzed within the constraints of time and quality

Table 2: Risk Analysis Table

			Impact		
ID	**Risk Description**	**Probability**	**Schedule**	**Quality**	**Overall**
1	Smartphone unable to capture RAW image	Low	High	High	High
2	High Latency to Remote Server	Medium	Medium	Low	Medium
3	No response from Server	Low	High	High	High

Table 3: Risk Probability Definition

Probability	**Description**
High	Probability of occurrence is $> 75\%$
Medium	Probability of occurrence is 26 - 75%
Low	Probability of occurrence is $< 25\%$

Table 4: Impact Description Table

Impact	Value	Description
High	> 10%	Schedule impact or Unacceptable quality
Medium	5 - 10%	Schedule impact or Some parts of the project have low quality
Low	< 5%	Schedule impact or Barely noticeable degradation in quality Low Impact on schedule or Quality can be incorporated

5.3.3 Overview of Risk Mitigation, Monitoring and Management

Risks are potential problems that might affect the successful completion of a software project. Risks involve uncertainty and potential losses. Risk analysis and management are intended to help a software team understand and manage uncertainty during the development process. The important thing is to remember that things can go wrong and to make plans to minimize their impact when they do. The work product is called a Risk Mitigation, Monitoring, and Management Plan (RMMM).

Table 5: RMMM Table

Risk ID	Risk	Response	Strategy
1	Smartphone unable to capture RAW image	Mitigate	Restart App
2	High Latency to Remote Server	Monitor	Check if smartphone is connected to the local network
3	No response from Server	Mitigate	Check if server is running without errors and Restart server

5.4 Project Schedule

5.4.1 Project Task Set

Table 6: Project Task Set Table

Task ID	Task Description
T1	Discussion on Domain selection
T2	Discussion on various topics in the selected domain and thrust area identification
T3	Finalising the Topic
T4	Searching current solutions that can complete the main objective of the project
T5	Discussion about how our solution would improve upon previously searched current solutions
T6	Selection of technologies for implementing the project
T7	Discussion on SRS
T8	Discussion about Operating Environments
T9	Discussion about functional and non-functional requirements
T10	SRS Review
T11	Discussion on UML Diagrams
T12	Discussion on Preparation of Mathematical Model
T13	Review of Report 1 First Draft.
T14	Report 1 final review
T15	Collection of Dataset of Images captured on Smartphone
T16	Renaming of Dataset of Images
T17	Develo ment of Neural Network
T18	Training of Neural Network on Dataset
T19	Setting up server to test images sent from browser
T20	Discussion about design of Mobile Application
T21	Development of Image Capture Module
T22	Development of Image Chooser Module
T23	Development of Image Upload Module
T24	Unit Testing of Modules
T25	Integration of All Modules
T26	Integration Testing of Application
T27	Review of Report 2 draft
T28	Report 2 final review

5.5 Task Network

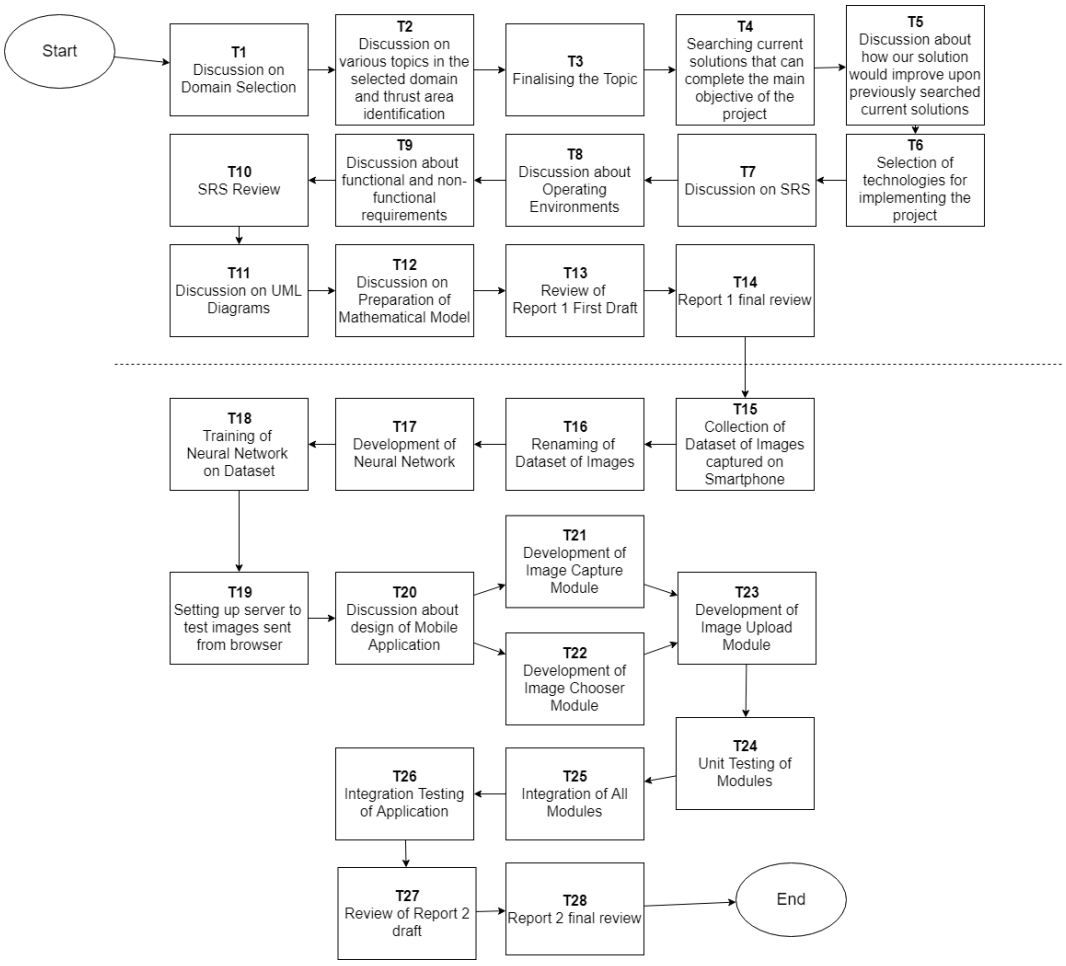


Figure 5: Task Network for our project

5.6 Timeline Chart

Table 7: Timeline Of the Project

Task ID	Timeline(Dates)
T1, T2, T3	17/06/2018 TO 23/06/2018
T4, T5	24/06/2018 TO 30/6/2018
T6	01/07/2018 TO 8/07/2018
T7, T8	9/07/2018 TO 15/07/2018
T9	17/07/2018 TO 22/07/2018
T10, T11	23/07/2018 TO 31/07/2018
T12	01/08/2018 TO 4/08/2018
T13	21/08/2018 TO 15/09/2018
T14	17/09/2018 TO 26/09/2018
T15	01/01/2019 TO 15/01/2019
T16, T17	16/01/2019 TO 30/01/2019
T18, T19, T20	01/02/2019 TO 15/02/2019
T21, T22, T23	17/02/2019 TO 27/02/2019
T24, T25, T26	15/03/2019 TO 25/03/2019
T27, T28	26/03/2019 TO 7/04/2019

5.7 Team Organization

5.7.1 Team Structure

Table 8: Team Organization Table

Sr No	Name	Roles
1	Dr J S Umale	Project Guide
2	Dr Swati Shinde	Project Guide
3	Mr Abhishek Gaikwad	Python ML Developer
4	Mr Dipak Bange	Python ML Developer
5	Mr Tejas Gajare	Android Application Developer
6	Mr Aditya Khadse	Android Application Developer

5.7.2 Management Reporting and Communication

Table 9: Management and Communication During Project Execution

Team Meeting	Discussion Points
Meeting 1	• Discussion on domain knowledge • Discussion on topic • Determination of scope of project • Setting of Objectives • Discussion on research requirement
Meeting 2	• Discussion about time, cost and budget • Setting of deadlines • Discussion about risks
Meeting 3	• Discussion on software/hardware used • Discussion on technology
Meeting 4	• Discussion on System Design • Discussion on SRS
Meeting 5	• Discussion on implemented modules • Discussion on user interface designs
Meeting 6	• Discussion on testing strategies

6 PROJECT IMPLEMENTATION

6.1 Overview of Project Modules

6.1.1 Smartphone Application

The smartphone application has a main interface, where the user can see what image will be processed and has three sub modules:

6.1.1.1 Choose file

This submodule deals with selecting an image from the smartphone. It allows the user to browse the device's memory and choose an image that was previously captured. The image must be in DNG format, as the system requires the image to be in RAW format to work, and smartphone cameras capture image in RAW in DNG format.

6.1.1.2 Capture Image

This submodule starts the camera so as to let the user preview a scene, capture it and allow it to be sent for processing. The camera app captures the image in RAW format only, hence the user does not need to worry about selecting an image, as the submodule will automatically select the last captured image.

6.1.1.3 Upload Image

This submodule uploads the image that has been loaded in the main interface to the URL shown in the main interface. The default URL can be changed by the user if required. The image is uploaded through a multipart POST request to the URL. The server is responsible for processing the image and responding with the processed output.

6.1.2 Flask Server

We have created a server written with Flask framework, which is responsible for the communication between the smartphone application and module responsible for processing the image. The server has a URL that allows POST requests from the smartphone application. We also have added the functionality that a file can be uploaded from web server by choosing the file. The server is also responsible for storing the file and then sending the file to the processing module.

6.1.3 Image Processing Unit

This module uses Tensorflow and Rawpy to process the image.

6.1.3.1 Rawpy submodule:

Rawpy first decreases black level from the image and then divides it into multiple channels

depending on the filter used. For example, the Bayer filter divides the image into 4 channels, Red, Green, Green, Blue. The channels are then feed into the neural network using Tensorflow.

6.1.3.2 Tensorflow submodule:

We have created a neural network in Tensorflow which takes in 4 channels of colors as provided by the rawpy submodule. The neural network then provides 3 channel output as Red, Green, Blue. We then use scipy to create the image from the RGB channels.

6.2 Tools and Technologies used

6.2.1 Database

- Firebase
- Local storage on server(computer)

6.2.2 Software Technologies

- Tensorflow
- Android Studio
- okhttp
- Flask

6.2.3 Hardware Tools

- Nvidia Titan X GPU
- Smartphone Camera (capable of capturing RAW images)

6.3 Algorithm Details

6.3.1 Convolutional Neural Network

CNNs, like neural networks, are made up of neurons with learnable weights and biases. Each neuron receives several inputs, takes a weighted sum over them, pass it through an activation function and responds with an output Parameter sharing is sharing of weights by all neurons in a particular feature map Local connectivity is the concept of each neural connected only to a subset of the input image (unlike a neural network where all the neurons are fully connected)

Convolutional neural networks the following components:

6.3.1.1 Convolutional Layer

The convolution layer comprises of a set of independent filters Each filter is independently convolved with the image (filters are initialized randomly and become our parameters which will be learned by the network subsequently)

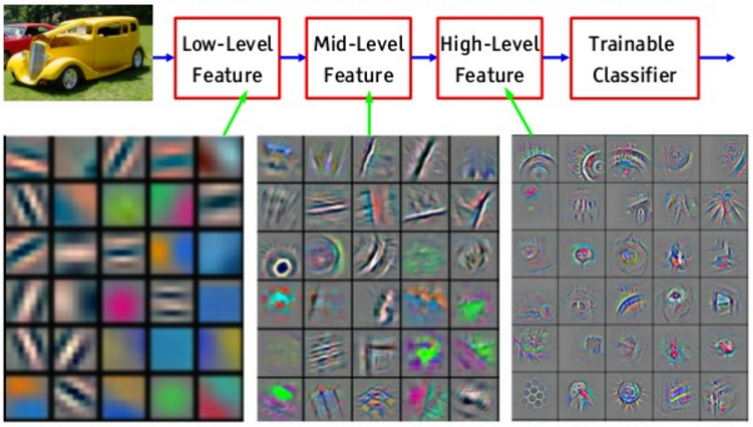


Figure 6: Feature visualization Of Convolutional Layer

6.3.1.2 Pooling Layer

Its function is to progressively reduce the spatial size of the representation to reduce the amount of parameters and computation in the network

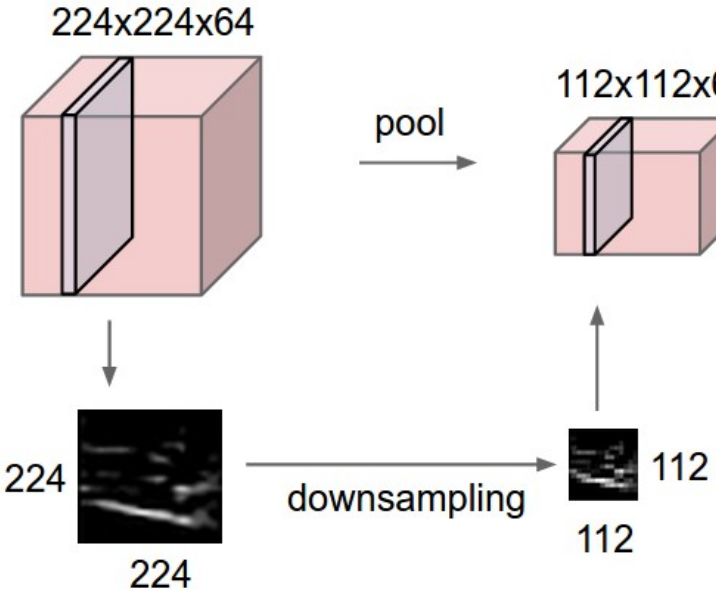


Figure 7: Pooling Of Images

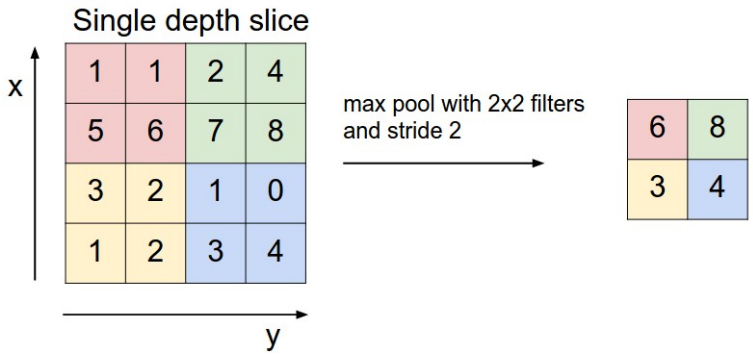


Figure 8: Max-Pooling Of Images

6.3.1.3 RELU

RELU is just a non linearity which is applied similar to neural networks.

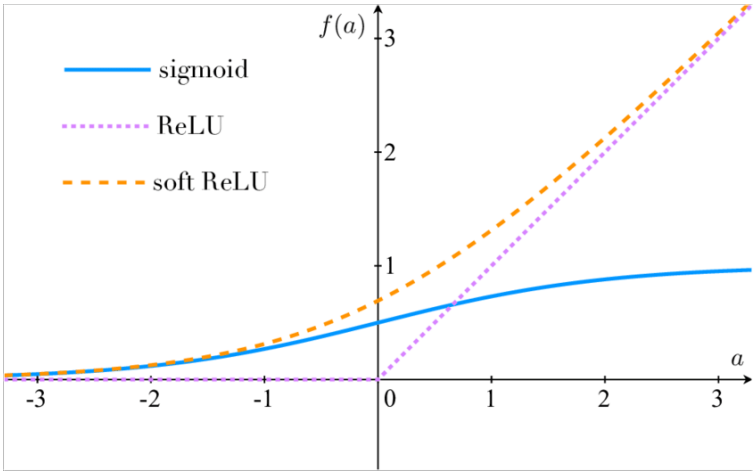


Figure 9: Comparison Of Activation Functions

7 SOFTWARE TESTING

7.1 Type of Testing

• Functional Testing:

This type of testing ignores the internal parts and focuses only on the output to check if it is as per the requirement or not. It is a Black-box type testing geared to the functional requirements of an application.

• Graphical User Interface (GUI) Testing

The objective of this GUI testing is to validate the GUI as per the business requirement. The expected GUI of the application is mentioned in the Detailed Design Document and GUI mockup screens. The GUI testing includes the size of the buttons and input field present on the screen, alignment of all text, tables and content in the tables.It also validates the menu of the application, after selecting different menu and menu items, it validates that the page does not fluctuate and the alignment remains same after hovering the mouse on the menu or sub-menu.

• Integration Testing

Testing of all integrated modules to verify the combined functionality after integration is termed as Integration Testing. Modules are typically code modules, individual applications, client and server applications on a network, etc. This type of testing is especially relevant to client/server and distributed systems.

• Incremental Integration Testing

Incremental Integration Testing is a Bottom-up approach for testing i.e continuous testing of an application when a new functionality is added. Application functionality and modules should be independent enough to test separately. This is done by programmers or by testers.

• Non-Functional Testing

It is a type of testing for which every organization having a separate team which usually called as Non-Functional Test (NFT) team or Performance team.Non-functional testing involves testing of non-functional requirements such as Load Testing, Stress Testing, Security, Volume, Recovery Testing etc. The objective of NFT testing is to ensure whether the response time of software or application is quick enough as per the business requirement.It should not take much time to load any page or system and should sustain during peak load.

• Performance Testing

This term is often used interchangeably with ‘stress’ and ‘load’ testing. Performance Testing is done to check whether the system meets the performance requirements. Different performance and load tools are used to do this testing.

- System Testing

Under System Testing technique, the entire system is tested as per the requirements. It is a Black-box type testing that is based on overall requirement specifications and covers all the combined parts of a system.

- Unit Testing

Testing of an individual software component or module is termed as Unit Testing. It is typically done by the programmer and not by testers, as it requires a detailed knowledge of the internal program design and code. It may also require developing test driver modules or test harnesses.

- Usability Testing

Under Usability Testing, User-friendliness check is done. Application flow is tested to know if a new user can understand the application easily or not, Proper help documented if a user gets stuck at any point. Basically, system navigation is checked in this testing.

7.2 Test Cases and Test Results

Table 10: Test Cases

Sr No	Test Cases	Expected Output	Output From Program
1.	CHOOSE Image (dng)	"Image Loads " Image will get displayed	"Image Loads " Image is displayed
2.	CHOOSE Image(not dng)	"Please select dng image" message will get displayed	"Please select dng image" message is displayed.
3.	Capture Image(for mobile with CAMERA2 api)	"Image saved to location " image will get displayed	"Image saved to location " image is displayed
4.	Capture Image(for mobile without CAMERA2 api)	"Your mobile doesn't support camera2Api" message will get displayed	"Your mobile doesn't support camera2Api" message is displayed
5.	Enhance	"Uploading to server " message will get displayed	"Uploading to server" message is displayed
6.	Enhance after uploading image	"Waiting for Server" message will get displayed	"Waiting for server" message is displayed
7.	Enhance after server sends enhanced image	"Enhanced image displayed " Enhanced Image will get displayed	"Enhanced image displayed" Enhance Image is displayed
8.	Hold Touch image	Lower exposure image will get displayed	get redirect to the questionnaire part

8 RESULTS

8.1 Outcomes

The following are major outcomes of our project:

- We will be able to enhance and predict how the still (image) will look like in actual light which is shot in extremely low light.

- We have trained our neural network model on the dataset of raw short exposure and its corresponding long exposure images.

- We are also be able to compare results of pretrained model on digital SLR cameras, our model trained on phone images and model trained after applying transfer learning on the pretrained model of SLR cameras by images captured on mobile phones.

- We have gained advance knowledge of all the metadata of the images like mobile sensors, black level, exposure, ISO, shutter speed, white balance, etc.

- An android application based on web-view that can work on all mobile phones who can capture raw images.

8.2 Screenshots

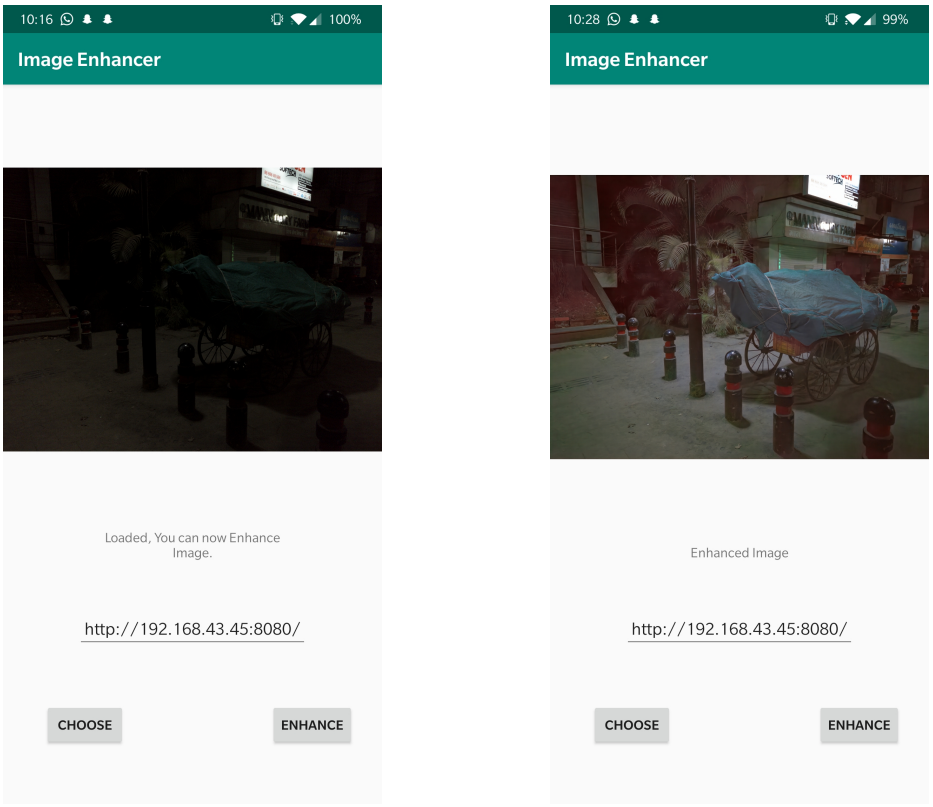


Figure 10: Screenshot of Application

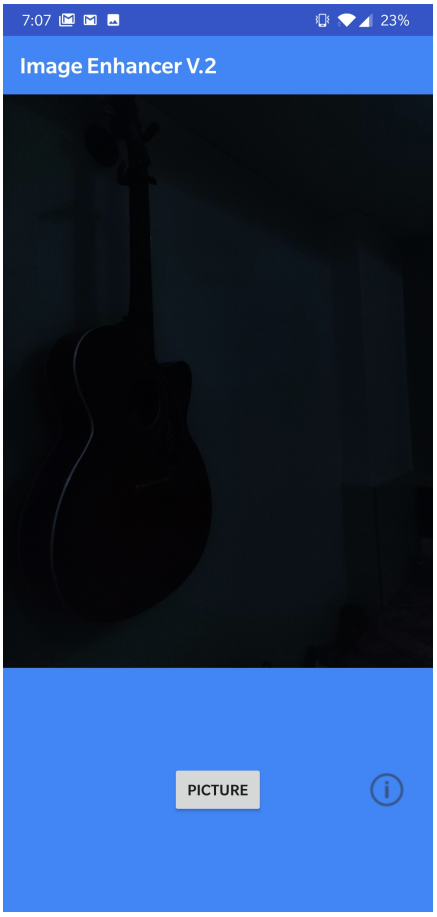


Figure 11: Screenshot of Camera Capture Application

8.3 Comparative Study of Results

Figure 12: Input image shot on mobile phone (.dng) vs. Output of pretrained model on digital SLR cameras

Figure 13: Input image shot on mobile phone (.dng) vs. Output from model trained on mobile phone cameras

Figure 14: Input image shot on mobile phone (.dng) vs. Output from model trained on SLR cameras and then on mobile phone cameras using transfer learning

Figure 15: Actual Output (ground truth) vs. Our final output

9 CONCLUSIONS AND FUTURE WORK

9.1 Conclusion

Imaging in the dark, at sub-lux conditions presents a challenge that traditional pipeline fails to overcome. In this system, we present a methodology through which a visually enhanced image is retrieved from an image which is captured in extreme low light conditions. Computational processing of low-light images proves to be inefficient and far less effective for the extreme low light conditions under consideration. A data driven approach is implemented to overcome this scenario. Specifically, we train deep neural networks to learn the image processing pipeline for low light raw data, including colour transformations, demosaicing, noise reduction, and image enhancement.

Current system that is presented in the "Learning to see in the dark" paper is the basis of our product. The deep neural network implemented in the aforementioned paper was trained using raw images captured through DSLR. This network is able to enhance images that are captured with the similar setup. To design a mobile application that shall enhance images captured from its comparatively smaller sensor, we enact the method of Transfer Learning. Through this methodology we train the deep neural network (which has already learnt to enhance images from DSLR) with our dataset of 2500 raw images captured through a mobile device. This provides us with a model suitable to satisfy our use case of enhancement of mobile camera images.

We present a mobile application through which users can capture images in extreme low light conditions and enhance them into visually per perceivable images. This application is based on client-server architecture. The mobile application will upload the raw image onto the server for processing and the corresponding enhanced image will be retrieved.

9.2 Future Work

Current limitation of the presented pipeline is that the amplification ratio must be chosen externally. It would be useful to infer a good amplification ratio from the input, akin to Auto ISO. Furthermore, we currently assume that a dedicated network is trained for a given camera sensor. Our preliminary experiments with cross-sensor generalization are encouraging, and future work could further study the generalization abilities of low-light imaging networks.

Another opportunity for future work is runtime optimization. The presented pipeline takes 0.38 and 0.66 seconds to process full-resolution Sony and Fuji images, respectively; this is not fast enough for real-time processing at full resolution, although a low-resolution preview can be produced in real time. We expect future work to yield further improvements in image quality, for example by systematically optimizing the network architecture and training procedure. We hope that the SID dataset and our experimental findings can stimulate and support such systematic investigation.

9.3 Applications

In order to capture sceneries in low light, modern cameras do not provide a reliable feature, as the images captured through traditional pipelines are unable to capture the minute details that can be captured by our method. Other solutions such as using flash light and denoising methods do not provide a perceptually good image as they work using image priors. Our method forms a model where a short exposure image is compared with a long exposure image and the model is formed by its understanding as how a short exposure image must be worked upon in order to achieve a long exposure image.

Hence, our method provides the advantage of creating long exposure images with the benefits of short exposure images. This can be used in places where low light imagery would be used:

1. Autonomous vehicles: With the rise of artificial intelligence, there are daily improvements in systems that drives vehicles without any human support. One of the most important aspects in driving is vision. Hence, with our method we can provide the system with additional information to improve the accuracy of the driving system.

2. Security: There is a great requirement of reliable low light imaging in security sector. Most surveillance cameras fail in low light conditions. Our method can improve visibility for the scene and could be extended to videos since the exposure of the images we have used is 1/30s which is basically same as using a video of 30 frames per second.

3. Low light photography: To capture low light scenes, current techniques involve adding light (using flashlight), but they end up ruining the artistic value of the photo. Our method would improve image quality and provide a closer representation to what the human eye sees.

4. Border Defence: At borders of a country, it becomes difficult to track movement in scenes outside the border without shedding light on it. Our method can be used to have a look at the entirety of the surrounding region. This will be able to do true around the clock surveillance.

5. Our system can be used in Military Applications for surveillance purposes where the conditions are sub-lux.

6. In the scope of security applications such as CCTV footage, higher level hardware needs to be implemented to retrieve more details. Using our pipeline, the less detailed images can be enhanced and more details can be captured without the need of additional hardware

7. Our pipeline can be implemented as a standalone application with tf.lite and be deployed on mobile phones with support camera2 api.

REFERENCES

[1] J. Long, E. Shelhamer, and T. Darrell. *Fully convolutional networks for semantic segmentation. In CVPR, 2015.*

[2] Z. Hu, S. Cho, J. Wang, and M.-H. Yang. *Deblurring lowlight images with light streaks.* In CVPR, 2014.

[3] V. Jain and H. S. Seung. *Natural image denoising with convolutional networks.* In NIPS, 2008. 2

[4] H. Jiang, Q. Tian, J. E. Farrell, and B. A. Wandell. *Learning the image processing pipeline. IEEE Transactions on Image Processing, 26(10), 2017.*

[5] H. Zhao, O. Gallo, I. Frosio, and J. Kautz. *Loss functions for image restoration with neural networks. IEEE Transactions on Computational Imaging, 3(1), 2017.*

[6] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian *Image denoising by sparse 3-D transform-domain collaborative filtering* IEEE Transactions on Image Processing, 16(8), 2007

[7] S. W. Hasinoff, D. Sharlet, R. Geiss, A. Adams, J. T. Barron, F. Kainz, J. Chen, and M. Levoy. *Burst photography for high dynamic range and low-light imaging on mobile cameras.* ACM Transactions on Graphics, 35(6), 2016.

[8] Addison-Wesley, Reading, Massachusetts, 1993. Z. Liu, L. Yuan, X. Tang, M. Uyttendaele, and J. Sun. *Fast burst images denoising. ACM Transactions on Graphics, 33(6), 2014.*

[9] Chen Chen, Qifeng Chen, Jia Xu, Vladlen Koltun *Learning to See in the Dark* In CVPR, 2018

[10] Olaf Ronneberger, Philipp Fischer, Thomas Bro *UNet - Convolutional Networks for Biomedical Image Segmentation* May 2015

[11] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. *Image quality assessment: From error visibility to structural similarity* IEEE Transactions on Image Processing, 13(4), 2004

[12] JurgenSchmidhuber *Deep learning in neural networks: An overview*

[13] Adobe Systems Incorporated *Digital Negative specifications*

APPENDIX A

NP-Complete Problem:

NP-complete problem, any of a class of computational problems for which no efficient solution algorithm has been found. Many significant computer-science problems belong to this class e.g., the traveling salesman problem, satisfiability problems, and graph-covering problems.

NP Hard Problem:

A problem is NP-hard if an algorithm for solving it can be translated into one for solving any NP-problem (nondeterministic polynomial time) problem. NP-hard therefore means "at least as hard as any NP-problem," although it might, in fact, be harder.

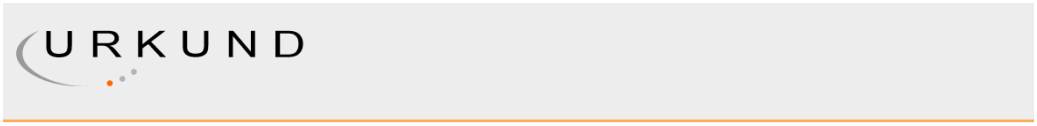
P Problem:

A problem is assigned to the P (polynomial time) class if there exists at least one algorithm to solve that problem, such that the number of steps of the algorithm is bounded by a polynomial in , where is the length of the input.

The problem being solved in the system can be considered as NP-Hard. The solution to the problem can be achieved upto some level of correctness. Correctness of the output cannot be defined as it is relative to the user requirements.

APPENDIX B

Plagiarism Report:



Urkund Analysis Result

Analysed Document:	GA8 - BE Project Report.pdf (D51096750)
Submitted:	4/26/2019 5:34:00 AM
Submitted By:	rahulpitale3@gmail.com
Significance:	15 %

Sources included in the report:

main.pdf (D50985616)
http://openaccess.thecvf.com/content_cvpr_2018/papers/Chen_Learning_to_See_CVPR_2018_paper.pdf
http://www.rebas.se/3339/

Instances where selected sources appear:

18

APPENDIX C

Copyright Details:

Copyright Office
Government of India

Acknowledgement Slip
(Date:29/04/2019)

Diary Number: **6563/2019-CO/SW**	Form Received: **Online**
Copyright Reg. of: **Computer Software**	Titled: **A robust mobile application to capture images in low illumination and transform them to corresponding high quality images**

Communication Address		
Name	**Address**	**Phone Number**
Abhishek Gaikwad	Plot No. 3-B, Mangesh Hou. Soc., Opp. RMZ Westend, Aundh, Pune-411007	9405847376

Financial Details

Payment ID	Amount	Bank Name	Payment Mode	Payment Date
197438	500	KOTAKPG	DC	29/04/2019

* For future communication please mention this DIARY No.

INSTRUCTIONS	
For the purpose of processing the application, following documents are mandatory to send by post along with this acknowledgement slip(Applicant's Copy).	
1.	2 Copies of work
2.	DD/IPO of Rs.(as applicable) per work favouring Registrar Copyright Office payable at New Delhi (Not applicable for online payment)
3.	Authorization from author/publisher
4.	If the work is being used on goods or capable of being used on the goods
5.	If the application is being field through attorney, a specific power of attorney in original duly signed by the applicant and accepted by the attorney
6.	Search Certificate from Trade Mark Office(TM-60) (*Only in case of Artistic work*).
7.	Applicant must take a print out of the application, sign it and send along with the other documents.
Kindly send the above documents within 30 Days from the date of online submission on the following address given by herewith: .	

Office of the Registrar of Copyrights
Copyright Office, Department of Industrial Policy & Promotion
Ministry of Commerce and Industry
Boudhik Sampada Bhawan,
Plot No. 32, Sector 14, Dwarka,
New Delhi-110075
Email Address: copyright@nic.in
Telephone No.: 011-28032496, 08929474194

OFFICE COPY
THANK YOU

Base Paper: Learning To See In The Dark - CVPR, 2018

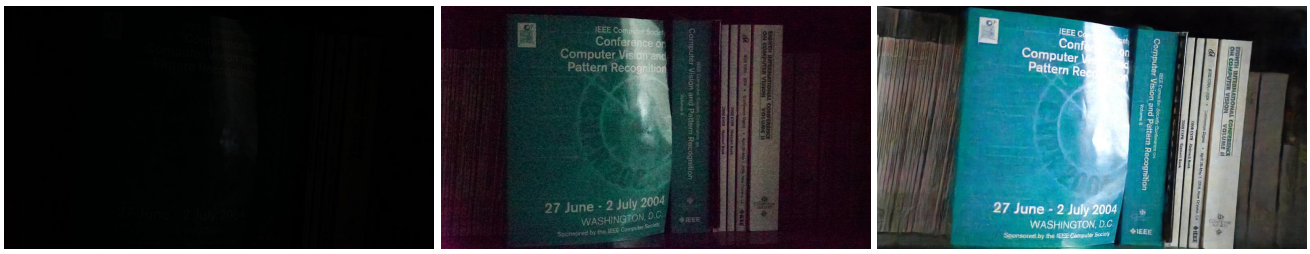
Learning to See in the Dark

Chen Chen
UIUC

Qifeng Chen
Intel Labs

Jia Xu
Intel Labs

Vladlen Koltun
Intel Labs



(a) Camera output with ISO 8,000 (b) Camera output with ISO 409,600 (c) Our result from the raw data of (a)

Figure 1. Extreme low-light imaging with a convolutional network. Dark indoor environment. The illuminance at the camera is < 0.1 lux. The Sony α7S II sensor is exposed for 1/30 second. (a) Image produced by the camera with ISO 8,000. (b) Image produced by the camera with ISO 409,600. The image suffers from noise and color bias. (c) Image produced by our convolutional network applied to the raw sensor data from (a).

Abstract

Imaging in low light is challenging due to low photon count and low SNR. Short-exposure images suffer from noise, while long exposure can induce blur and is often impractical. A variety of denoising, deblurring, and enhancement techniques have been proposed, but their effectiveness is limited in extreme conditions, such as video-rate imaging at night. To support the development of learning-based pipelines for low-light image processing, we introduce a dataset of raw short-exposure low-light images, with corresponding long-exposure reference images. Using the presented dataset, we develop a pipeline for processing low-light images, based on end-to-end training of a fully-convolutional network. The network operates directly on raw sensor data and replaces much of the traditional image processing pipeline, which tends to perform poorly on such data. We report promising results on the new dataset, analyze factors that affect performance, and highlight opportunities for future work.

1. Introduction

Noise is present in any imaging system, but it makes imaging particularly challenging in low light. High ISO can be used to increase brightness, but it also amplifies noise. Postprocessing, such as scaling or histogram stretching, can be applied, but this does not resolve the low signal-to-noise ratio (SNR) due to low photon counts. There are physical means to increase SNR in low light, including opening the aperture, extending exposure time, and using flash. But each of these has its own characteristic drawbacks. For example, increasing exposure time can introduce blur due to camera shake or object motion.

The challenge of fast imaging in low light is well-known in the computational photography community, but remains open. Researchers have proposed techniques for denoising, deblurring, and enhancement of low-light images [34, 16, 42]. These techniques generally assume that images are captured in somewhat dim environments with moderate levels of noise. In contrast, we are interested in extreme low-light imaging with severely limited illumination (e.g., moonlight) and short exposure (ideally at video rate). In this regime, the traditional camera processing pipeline breaks down and the image has to be reconstructed from the raw sensor data.

Figure 1 illustrates our setting. The environment is extremely dark: less than 0.1 lux of illumination at the camera. The exposure time is set to 1/30 second. The aperture is f/5.6. At ISO 8,000, which is generally considered high, the camera produces an image that is essentially black, despite the high light sensitivity of the full-frame Sony sensor. At ISO 409,600, which is far beyond the reach of most cameras, the content of the scene is discernible, but the image is dim, noisy, and the colors are distorted. As we will show, even state-of-the-art denoising techniques [32] fail to remove such noise and do not address the color bias. An alternative approach is to use a burst of images [24, 14], but

burst alignment algorithms may fail in extreme low-light conditions and burst pipelines are not designed for video capture (e.g., due to the use of 'lucky imaging' within the burst).

We propose a new image processing pipeline that addresses the challenges of extreme low-light photography via a data-driven approach. Specifically, we train deep neural networks to learn the image processing pipeline for low-light raw data, including color transformations, demosaicing, noise reduction, and image enhancement. The pipeline is trained end-to-end to avoid the noise amplification and error accumulation that characterize traditional camera processing pipelines in this regime.

Most existing methods for processing low-light images were evaluated on synthetic data or on real low-light images without ground truth. To the best of our knowledge, there is no public dataset for training and testing techniques for processing fast low-light images with diverse real-world data and ground truth. Therefore, we have collected a new dataset of raw images captured with fast exposure in low-light conditions. Each low-light image has a corresponding long-exposure high-quality reference image. We demonstrate promising results on the new dataset: low-light images are amplified by up to 300 times with successful noise reduction and correct color transformation. We systematically analyze key elements of the pipeline and discuss directions for future research.

2. Related Work

Computational processing of low-light images has been extensively studied in the literature. We provide a short review of existing methods.

Image denoising. Image denoising is a well-developed topic in low-level vision. Many approaches have been proposed, using techniques such as total variation [36], wavelet-domain processing [33], sparse coding [9, 28], nuclear norm minimization [12], and 3D transform-domain filtering (BM3D) [7]. These methods are often based on specific image priors such as smoothness, sparsity, low rank, or self-similarity. Researchers have also explored the application of deep networks to denoising, including stacked sparse denoising auto-encoders (SSDA) [39, 1], trainable nonlinear reaction diffusion (TNRD) [6], multi-layer perceptrons [3], deep autoencoders [26], and convolutional networks [17, 41]. When trained on certain noise levels, these data-driven methods can compete with state-of-the-art classic techniques such as BM3D and sparse coding. Unfortunately, most existing methods have been evaluated on synthetic data, such as images with added Gaussian or salt&pepper noise. A careful recent evaluation with real data found that BM3D outperforms more recent techniques on real images [32]. Joint denoising and demosaicing has also been studied, including recent work that uses deep networks [15, 10], but these methods have been evaluated on synthetic Bayer patterns and synthetic noise, rather than real images collected in extreme low-light conditions.

In addition to single-image denoising, multiple-image denoising has also been considered and can achieve better results since more information is collected from the scene [31, 23, 19, 24, 14, 29]. In particular, Liu et al. [24] and Hasinoff et al. [14] propose to denoise a burst of images from the same scene. While often effective, these pipelines can be elaborate, involving reference image selection ('lucky imaging') and dense correspondence estimation across images. We focus on a complementary line of investigation and study how far single-image processing can be pushed.

Low-light image enhancement. A variety of techniques have been applied to enhance the contrast of low-light images. One classic choice is histogram equalization, which balances the histogram of the entire image. Another widely used technique is gamma correction, which increases the brightness of dark regions while compressing bright pixels. More advanced methods perform more global analysis and processing, using for example the inverse dark channel prior [8, 29], the wavelet transform [27], the Retinex model [30], and illumination map estimation [13]. However, these methods generally assume that the images already contain a good representation of the scene content. They do not explicitly model image noise and typically apply off-the-shelf denoising as a postprocess. In contrast, we consider extreme low-light imaging, with severe noise and color distortion that is beyond the operating conditions of existing enhancement pipelines.

Noisy image datasets. Although there are many studies of image denoising, most existing methods are evaluated on synthetic data, such as clean images with added Gaussian or salt&pepper noise. The RENOIR dataset [2] was proposed to benchmark denoising with real noisy images. However, as reported in the literature [32], image pairs in the RENOIR dataset exhibit spatial misalignment. Bursts of images have been used to reduce noise in low-light conditions [24], but the associated datasets do not contain reliable ground-truth data. The Google HDR+ dataset [14] does not target extreme low-light imaging: most images in the dataset were captured during the day. The recent Darmstadt Noise Dataset (DND) [32] aims to address the need for real data in the denoising community, but the images were captured during the day and are not suitable for evaluation of low-light image processing. To the best of our knowledge, there is no public dataset with raw low-light images and corresponding ground truth. We therefore collect such a dataset to support systematic reproducible research in this area.

Sony α7S II	Filter array	Exposure time (s)	# images
x300	Bayer	1/10, 1/30	1190
x250	Bayer	1/25	699
x100	Bayer	1/10	808
Fujifilm X-T2	Filter array	Exposure time (s)	# images
x300	X-Trans	1/30	630
x250	X-Trans	1/25	650
x100	X-Trans	1/10	1117

Table 1. The See-in-the-Dark (SID) dataset contains 5094 raw short-exposure images, each with a reference long-exposure image. The images were collected by two cameras (top and bottom). From left to right: ratio of exposure times between input and reference images, filter array, exposure time of input image, and number of images in each condition.

3. See-in-the-Dark Dataset

We collected a new dataset for training and benchmarking single-image processing of raw low-light images. The See-in-the-Dark (SID) dataset contains 5094 raw short-exposure images, each with a corresponding long-exposure reference image. Note that multiple short-exposure images can correspond to the same long-exposure reference image. For example, we collected sequences of short-exposure images to evaluate burst denoising methods. Each image in the sequence is counted as a distinct low-light image, since each such image contains real imaging artifacts and is useful for training and testing. The number of distinct long-exposure reference images in SID is 424.

The dataset contains both indoor and outdoor images. The outdoor images were generally captured at night, under moonlight or street lighting. The illuminance at the camera in the outdoor scenes is generally between 0.2 lux and 5 lux. The indoor images are even darker. They were captured in closed rooms with regular lights turned off and with faint indirect illumination set up for this purpose. The illuminance at the camera in the indoor scenes is generally between 0.03 lux and 0.3 lux.

The exposure for the input images was set between 1/30 and 1/10 seconds. The corresponding reference (ground truth) images were captured with 100 to 300 times longer exposure: i.e., 10 to 30 seconds. Since exposure times for the reference images are necessarily long, all the scenes in the dataset are static. The dataset is summarized in Table 1. A small sample of reference images is shown in Figure 2. Approximately 20% of the images in each condition are randomly selected to form the test set, and another 10% are selected for the validation set.

Images were captured using two cameras: Sony α7S II and Fujifilm X-T2. These cameras have different sensors: the Sony camera has a full-frame Bayer sensor and the Fuji camera has an APS-C X-Trans sensor. This supports evaluation of low-light image processing pipelines on images produced by different filter arrays. The resolution is 4240×2832 for Sony and 6000×4000 for the Fuji images. The Sony set was collected using two different lenses.

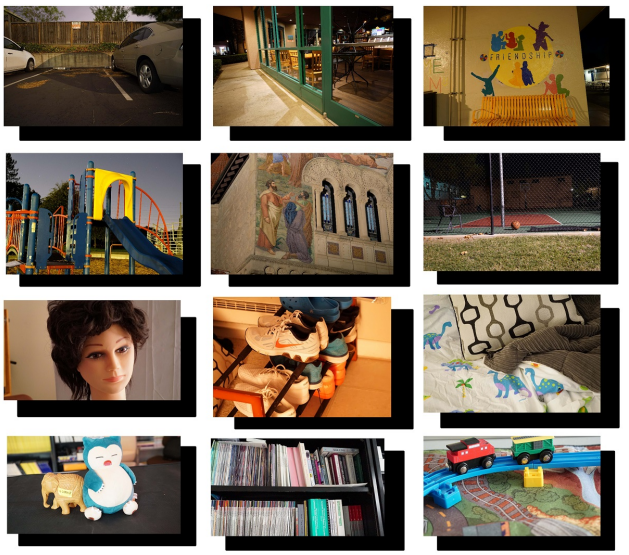


Figure 2. Example images in the SID dataset. Outdoor images in the top two rows, indoor images in the bottom rows. Long-exposure reference (ground truth) images are shown in front. Short-exposure input images (essentially black) are shown in the back. The illuminance at the camera is generally between 0.2 and 5 lux outdoors and between 0.03 and 0.3 lux indoors.

The cameras were mounted on sturdy tripods. We used mirrorless cameras to avoid vibration due to mirror flapping. In each scene, camera settings such as aperture, ISO, focus, and focal length were adjusted to maximize the quality of the reference (long-exposure) images. After a long-exposure reference image was taken, a remote smartphone app was used to decrease the exposure time by a factor of 100 to 300 for a sequence of short-exposure images. The camera was not touched between the long-exposure and the short-exposure images. We collected sequences of short-exposure images to support comparison with an idealized burst-imaging pipeline that benefits from perfect alignment.

The long-exposure reference images may still contain some noise, but the perceptual quality is sufficiently high for these images to serve as ground truth. We target applications that aim to produce perceptually good images in low-light conditions, rather than exhaustively removing all noise or maximizing image contrast.

4. Method

4.1. Pipeline

After getting the raw data from an imaging sensor, the traditional image processing pipeline applies a sequence of

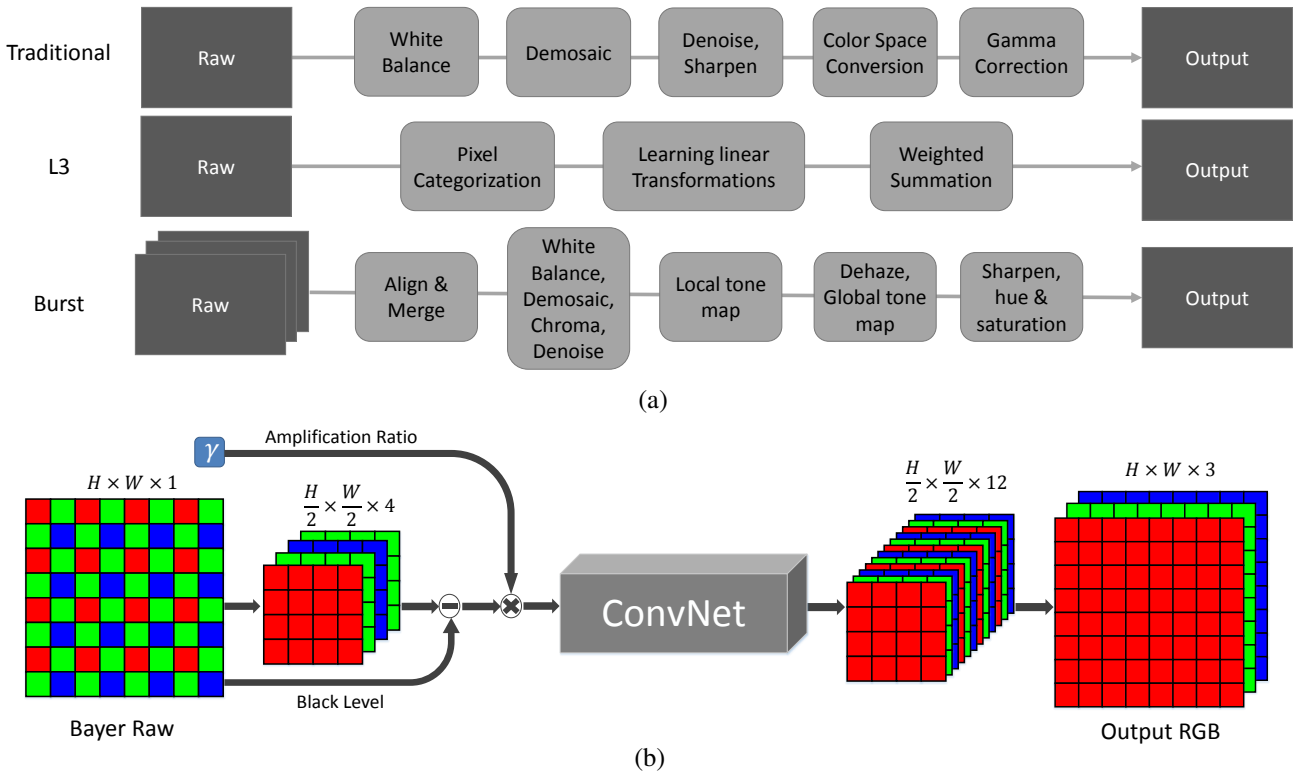


Figure 3. The structure of different image processing pipelines. (a) From top to bottom: a traditional image processing pipeline, the L3 pipeline [18], and a burst imaging pipeline [14]. (b) Our pipeline.

modules such as white balance, demosaicing, denoising, sharpening, color space conversion, gamma correction, and others. These modules are often tuned for specific cameras. Jiang et al. [18] proposed to use a large collection of local, linear, and learned (L3) filters to approximate the complex nonlinear pipelines found in modern consumer imaging systems. Yet neither the traditional pipeline nor the L3 pipeline successfully deal with fast low-light imaging, as they are not able to handle the extremely low SNR. Hasinoff et al. [14] described a burst imaging pipeline for smartphone cameras. This method can produce good results by aligning and blending multiple images, but introduces a certain level of complexity, for example due to the need for dense correspondence estimation, and may not easily extend to video capture, for example due to the use of lucky imaging.

We propose to use end-to-end learning for direct single-image processing of fast low-light images. Specifically, we train a fully-convolutional network (FCN) [22, 25] to perform the entire image processing pipeline. Recent work has shown that pure FCNs can effectively represent many image processing algorithms [40, 5]. We are inspired by this work and investigate the application of this approach to extreme low-light imaging. Rather than operating on normal sRGB images produced by traditional camera processing pipelines, we operate on raw sensor data.

Figure 3(b) illustrates the structure of the presented pipeline. For Bayer arrays, we pack the input into four channels and correspondingly reduce the spatial resolution by a factor of two in each dimension. For X-Trans arrays (not shown in the figure), the raw data is arranged in 6×6 blocks; we pack it into 9 channels instead of 36 channels by exchanging adjacent elements. We subtract the black level and scale the data by the desired amplification ratio (e.g., x100 or x300). The packed and amplified data is fed into a fully-convolutional network. The output is a 12-channel image with half the spatial resolution. This half-sized output is processed by a sub-pixel layer to recover the original resolution [37].

After preliminary exploration, we have focused on two general structures for the fully-convolutional network that forms the core of our pipeline: a multi-scale context aggregation network (CAN) recently used for fast image processing [5] and a U-net [35]. Other work has explored residual connections [20, 34, 41], but we did not find these beneficial in our setting, possibly because our input and output are represented in different color spaces. Another consideration that affected our choice of architectures is memory consumption: we have chosen architectures that can process a full-resolution image (e.g., at 4240×2832 or 6000×4000 resolution) in GPU memory. We have therefore avoided

(a) x28 (b) x87 (c) x189 (d) x366

Figure 4. The effect of the amplification factor on a patch from an indoor image in the SID dataset (Sony x100 subset). The amplification factor is provided as an external input to our pipeline, akin to the ISO setting in cameras. Higher amplification factors yield brighter images. This figure shows the output of our pipeline with different amplification factors.

fully-connected layers that require processing small image patches and reassembling them [26]. Our default architecture is the U-net [35].

The amplification ratio determines the brightness of the output. In our pipeline, the amplification ratio is set externally and is provided as input to the pipeline, akin to the ISO setting in cameras. Figure 4 shows the effect of different amplification ratios. The user can adjust the brightness of the output image by setting different amplification factors. At test time, the pipeline performs blind noise suppression and color transformation. The network outputs the processed image directly in sRGB space.

4.2. Training

We train the networks from scratch using the L_1 loss and the Adam optimizer [21]. During training, the input to the network is the raw data of the short-exposed image and the ground truth is the corresponding long-exposure image in sRGB space (processed by `libraw`, a raw image processing library). We train one network for each camera. The amplification ratio is set to be the exposure difference between the input and reference images (e.g., x100, x250, or x300) for both training and testing. In each iteration, we randomly crop a 512×512 patch for training and apply random flipping and rotation for data augmentation. The learning rate is initially set to 10^{-4} and is reduced to 10^{-5} after 2000 epochs. Training proceeds for 4000 epochs.

5. Experiments

5.1. Qualitative results and perceptual experiments

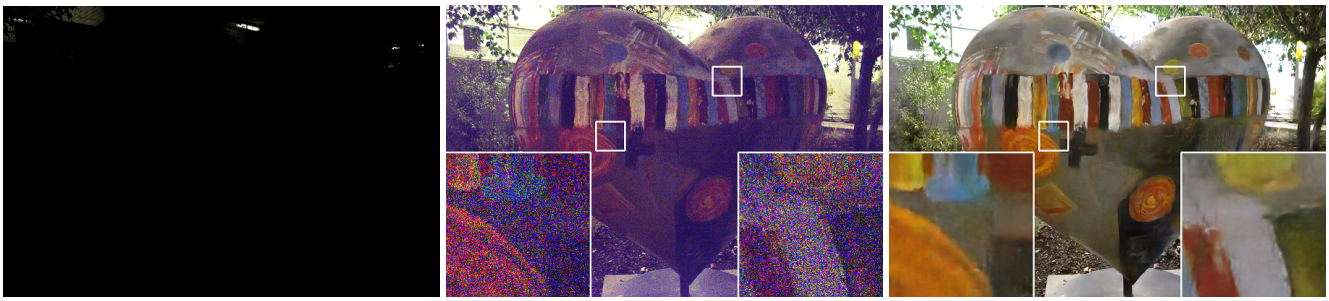
Comparison to traditional pipeline. Our initial baseline is the traditional camera processing pipeline, with amplification prior to quantization. (We use the same amplification ratio as the one given to our pipeline.) Qualitative comparisons to this baseline are shown in Figures 5, 6, and 7. Images produced by the traditional pipeline in extreme low-light conditions suffer from severe noise and color distortion.

Comparison to denoising and burst processing. The natural next step is to apply an existing denoising algorithm post-hoc to the output of the traditional pipeline. A careful recent evaluation on real data has shown that BM3D [7] outperforms more recent denoising models on real images [32]. We thus use BM3D as the reference denoising algorithm. Figure 7 illustrates the results. Note that BM3D is a non-blind denoising method and requires the noise level to be specified extrinsically as a parameter. A small noise level setting may leave perceptually significant noise in the image, while a large level may over-smooth. As shown in Figure 7, the two effects can coexist in the same image, since uniform additive noise is not an appropriate model for real low-light images. In contrast, our pipeline performs blind noise suppression that can locally adapt to the data. Furthermore, post-hoc denoising does not address other artifacts present in the output of the traditional pipeline, such as color distortion.

We also compare to burst denoising [24, 14]. Since image sequences in our dataset are already aligned, the burst-imaging pipeline we compare to is idealized: it benefits from perfect alignment, which is not present in practice. Since alignment is already taken care of, we perform burst denoising by taking the per-pixel median for a sequence of 8 images.

Comparison in terms of PSNR/SSIM using the reference long-exposure images would not be fair to BM3D and burst processing, since these baselines have to use input images that undergo different processing. For fair comparison, we reduce color bias by using the white balance coefficients of the reference image. In addition, we scale the images given to the baselines channel-by-channel to the same mean values as the reference image. These adjustments bring the images produced by the baselines closer in appearance to the reference image in terms of color and brightness. Note that this amounts to using privileged information to help the baselines.

To evaluate the relative quality of images produced by our pipeline, BM3D denoising, and burst denoising, we conduct a perceptual experiment based on blind randomized



(a) JPEG image produced by camera (b) Raw data via traditional pipeline (c) Our result

Figure 5. (a) An image captured at night by the Fujifilm X-T2 camera with ISO 800, aperture f/7.1, and exposure of 1/30 second. The illuminance at the camera is approximately 1 lux. (b) Processing the raw data by a traditional pipeline does not effectively handle the noise and color bias in the data. (c) Our result obtained from the same raw data.

A/B tests deployed on the Amazon Mechanical Turk platform [4]. Each comparison presents corresponding images produced by two different pipelines to an MTurk worker, who has to determine which image has higher quality. Image pairs are presented in random order, with random left-right order, and no indication of the provenance of different images. A total of 1180 comparisons were performed by 10 MTurk workers. Table 2 shows the rates at which workers chose an image produced by the presented pipeline over a corresponding image produced by one of the baselines. We performed the experiment with images from two subsets of the test set: Sony x300 (challenging) and Sony x100 (easier). Our pipeline significantly outperforms the baselines on the challenging x300 set and is on par on the easier x100 set. Recall that the experiment is skewed in favor of the baselines due to the oracle preprocessing of the data provided to the baselines. Note also that burst denoising uses information from 8 images with perfect alignment.

	Sony x300 set	Sony x100 set
Ours > BM3D	92.4%	59.3%
Ours > Burst	85.2%	47.3%

Table 2. Perceptual experiments were used to compare the presented pipeline with BM3D and burst denoising. The experiment is skewed in favor of the baselines, as described in the text. The presented single-image pipeline still significantly outperforms the baselines on the challenging x300 set and is on par on the easier x100 set.

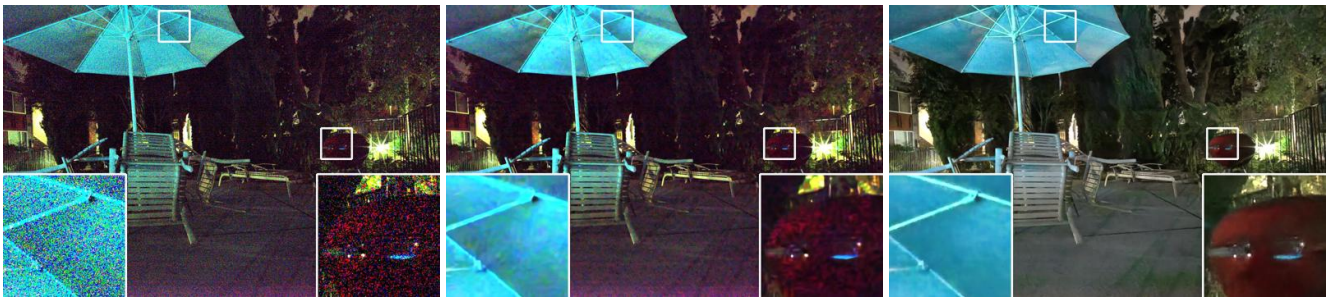
Qualitative results on smartphone images. We expect that best results will be obtained when a dedicated network is trained for a specific camera sensor. However, our preliminary experiments with cross-sensor generalization indicate that this may not always be necessary. We have applied a model trained on the Sony subset of SID to images captured by an iPhone 6s smartphone, which also has a Bayer filter array and 14-bit raw data. We used an app to manually set ISO and other parameters, and exported raw data for processing. A representative result is shown in Figure 6. The low-light data processed by the traditional pipeline suffers from severe noise and color shift. The result of our network, trained on images from a different camera, has good contrast, low noise, and well-adjusted color.

(a) Traditional pipeline (b) Our result

Figure 6. Application of a network trained on SID to a low-light raw image taken with an iPhone 6s smartphone. (a) A raw image captured at night with an iPhone 6s with ISO 400, aperture f/2.2, and exposure time 0.05s. This image was processed by the traditional image processing pipeline and scaled to match the brightness of the reference image. (b) The output of our network, with amplification ratio x100.

5.2. Controlled experiments

Table 3 (first row) reports the accuracy of the presented pipeline in terms of Peak Signal-to-Noise Ratio (PSNR) and Structural SIMilarity (SSIM) [38]. We now describe a sequence of controlled experiments that evaluate the effect of different elements in the pipeline.



(a) Traditional pipeline (b) ... followed by BM3D denoising (c) Our result

Figure 7. An image from the Sony x300 set. (a) Low-light input processed by the traditional image processing pipeline and linear scaling. (b) Same, followed by BM3D denoising. (c) Our result.

Condition	Sony	Fuji
1. **Our default pipeline**	28.88/0.787	26.61/0.680
2. U-net → CAN	27.40/0.792	25.71/0.710
3. Raw → sRGB	17.40/0.554	25.11/0.648
4. L_1 → SSIM loss	28.64/0.817	26.20/0.685
5. L_1 → L_2 loss	28.47/0.784	26.51/0.680
6. Packed → Masked	26.95/0.744	–
7. X-Trans $3 \times 3 \rightarrow 6 \times 6$	–	23.05/0.567
8. Stretched references	18.23/0.674	16.85/0.535

Table 3. Controlled experiments. This table reports mean PSNR/SSIM in each condition.

Network structure. We begin by comparing different network architectures. Table 3 (row 2) reports the result of replacing the U-net [35] (our default architecture) by the CAN [5]. The U-net has higher PSNR on both sets. Although images produced by the CAN have higher SSIM, they sometimes suffer from loss of color. A patch from the Fuji x300 set is shown in Figure 8. Here colors are not recovered correctly by the CAN.

(a) CAN (b) U-net

Figure 8. Comparison of network architectures on an image patch from the Fuji x300 test set. (a) Using the CAN structure, the color is not recovered correctly. (b) Using the U-net. Zoom in for detail.

Input color space. Most existing denoising methods operate on sRGB images that have already been processed by a traditional image processing pipeline. We have found that operating directly on raw sensor data is much more effective in extreme low-light conditions. Table 3 (row 3) shows the results of the presented pipeline when it's applied to sRGB images produced by the traditional pipeline.

Loss functions. We use the L_1 loss by default, but have evaluated many alternative loss functions. As shown in Table 3 (rows 4 and 5), replacing the L_1 loss by L_2 or SSIM [43] produces comparable results. We have not observed systematic perceptual benefits for any one of these loss functions. Adding a total variation loss does not improve accuracy. Adding a GAN loss [11] significantly reduces accuracy.

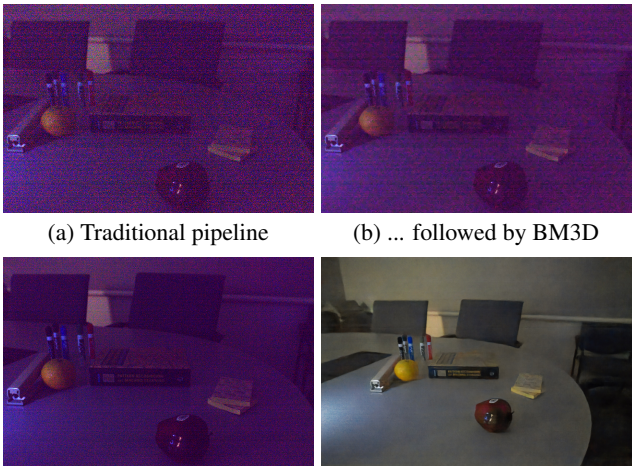
Data arrangement. The raw sensor data has all colors in a single channel. Common choices for arranging raw data for a convolutional network are packing the color values into different channels with correspondingly lower spatial resolution, or duplicating and masking different colors [10]. We use packing by default. As shown in Table 3 (row 6), masking the Bayer data (Sony subset) yields lower PSNR/SSIM than packing; a typical perceptual artifact of the masking approach is loss of some hues in the output.

The X-Trans data is very different in structure from the Bayer data and is arranged in 6×6 blocks. One option is to pack it into 36 channels. Instead, we exchange some values between neighboring elements to create a 3×3 pattern, which is packed into 9 channels. As shown in Table 3 (row 7), 6×6 packing yields lower PSNR/SSIM; a typical perceptual artifact is loss of color and detail.

Postprocessing. In initial experiments, we included histogram stretching in the processing pipeline for the reference images. Thus the network had to learn histogram stretching in addition to the rest of the processing pipeline. Despite trying many network architectures and loss functions, we were not successful in training networks to per-

(a) (b) (c) (d)

Figure 9. Effect of histogram stretching. (a) A reference image in the Sony x100 set, produced with histogram stretching. (b) Output if trained on histogram-stretched images. The result suffers from artifacts on the wall. (c) Output if trained on images without histogram stretching. The result is darker but cleaner. (d) The image (c) after histogram stretching applied in postprocessing.



(a) Traditional pipeline (b) ... followed by BM3D (c) Burst denoising (d) Our result

Figure 10. Limited signal recovery in extreme low-light conditions (indoor, dark room, 0.2 lux). (a) An input image in the Sony x300 set, processed by the traditional pipeline and amplified to match the reference. (b) BM3D denoising applied to (a). (c) Burst denoising with 8 images: the result is still bad due to the severe artifacts in all images in the burst. (d) The result of our network; loss of detail is apparent upon close examination.

form this task. As shown in Table 3 (row 8), the accuracy of the network drops significantly when histogram stretching is applied to the reference images (and thus the network has to learn histogram stretching). Our experiments suggest that our pipeline does not easily learn to model and manipulate global histogram statistics across the entire image, and is prone to overfitting the training data when faced with this task. We thus exclude histogram stretching from the pipeline and optionally apply it as postprocessing. Figure 9 shows a typical result in which attempting to learn histogram stretching yields visible artifacts at test time. The result of training on unstretched reference images is darker but cleaner.

6. Discussion

Fast low-light imaging is a formidable challenge due to low photon counts and low SNR. Imaging in the dark, at video rates, in sub-lux conditions, is considered impractical with traditional signal processing techniques. In this paper, we presented the See-in-the-Dark (SID) dataset, created to support the development of data-driven approaches that may enable such extreme imaging. Using SID, we have developed a simple pipeline that improves upon traditional processing of low-light images. The presented pipeline is based on end-to-end training of a fully-convolutional network. Experiments demonstrate promising results, with successful noise suppression and correct color transformation on SID data.

The presented work opens many opportunities for future research. Our work did not address HDR tone mapping. (Note the saturated regions in Figure 1(c).) The SID dataset is limited in that it does not contain humans and dynamic objects. The results of the presented pipeline are imperfect and can be improved in future work; the x300 subset is particularly challenging. Some artifacts in the output of the presented approach are demonstrated in Figure 10(d).

Another limitation of the presented pipeline is that the amplification ratio must be chosen externally. It would be useful to infer a good amplification ratio from the input, akin to Auto ISO. Furthermore, we currently assume that a dedicated network is trained for a given camera sensor. Our preliminary experiments with cross-sensor generalization are encouraging, and future work could further study the generalization abilities of low-light imaging networks.

Another opportunity for future work is runtime optimization. The presented pipeline takes 0.38 and 0.66 seconds to process full-resolution Sony and Fuji images, respectively; this is not fast enough for real-time processing at full resolution, although a low-resolution preview can be produced in real time.

We expect future work to yield further improvements in image quality, for example by systematically optimizing the network architecture and training procedure. We hope that the SID dataset and our experimental findings can stimulate and support such systematic investigation.

References

[1] F. Agostinelli, M. R. Anderson, and H. Lee. Adaptive multi-column deep neural networks with application to robust image denoising. In *NIPS*, 2013. 2

[2] J. Anaya and A. Barbu. RENOIR – A dataset for real low-light image noise reduction. *arXiv:1409.8230*, 2014. 2

[3] H. C. Burger, C. J. Schuler, and S. Harmeling. Image denoising: Can plain neural networks compete with BM3D? In *CVPR*, 2012. 2

[4] Q. Chen and V. Koltun. Photographic image synthesis with cascaded refinement networks. In *ICCV*, 2017. 6

[5] Q. Chen, J. Xu, and V. Koltun. Fast image processing with fully-convolutional networks. In *ICCV*, 2017. 4, 7

[6] Y. Chen and T. Pock. Trainable nonlinear reaction diffusion: A flexible framework for fast and effective image restoration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6), 2017. 2

[7] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian. Image denoising by sparse 3-D transform-domain collaborative filtering. *IEEE Transactions on Image Processing*, 16(8), 2007. 2, 5

[8] X. Dong, G. Wang, Y. Pang, W. Li, J. Wen, W. Meng, and Y. Lu. Fast efficient algorithm for enhancement of low lighting video. In *IEEE International Conference on Multimedia and Expo*, 2011. 2

[9] M. Elad and M. Aharon. Image denoising via sparse and redundant representations over learned dictionaries. *IEEE Transactions on Image Processing*, 15(12), 2006. 2

[10] M. Gharbi, G. Chaurasia, S. Paris, and F. Durand. Deep joint demosaicking and denoising. *ACM Transactions on Graphics*, 35(6), 2016. 2, 7

[11] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *NIPS*, 2014. 7

[12] S. Gu, L. Zhang, W. Zuo, and X. Feng. Weighted nuclear norm minimization with application to image denoising. In *CVPR*, 2014. 2

[13] X. Guo, Y. Li, and H. Ling. LIME: Low-light image enhancement via illumination map estimation. *IEEE Transactions on Image Processing*, 26(2), 2017. 2

[14] S. W. Hasinoff, D. Sharlet, R. Geiss, A. Adams, J. T. Barron, F. Kainz, J. Chen, and M. Levoy. Burst photography for high dynamic range and low-light imaging on mobile cameras. *ACM Transactions on Graphics*, 35(6), 2016. 1, 2, 4, 5

[15] K. Hirakawa and T. W. Parks. Joint demosaicing and denoising. *IEEE Transactions on Image Processing*, 15(8), 2006. 2

[16] Z. Hu, S. Cho, J. Wang, and M.-H. Yang. Deblurring low-light images with light streaks. In *CVPR*, 2014. 1

[17] V. Jain and H. S. Seung. Natural image denoising with convolutional networks. In *NIPS*, 2008. 2

[18] H. Jiang, Q. Tian, J. E. Farrell, and B. A. Wandell. Learning the image processing pipeline. *IEEE Transactions on Image Processing*, 26(10), 2017. 4

[19] N. Joshi and M. F. Cohen. Seeing Mt. Rainier: Lucky imaging for multi-image denoising, sharpening, and haze removal. In *ICCP*, 2010. 2

[20] J. Kim, J. K. Lee, and K. M. Lee. Accurate image super-resolution using very deep convolutional networks. In *CVPR*, 2016. 4

[21] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 5

[22] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4), 1989. 4

[23] C. Liu and W. T. Freeman. A high-quality video denoising algorithm based on reliable motion estimation. In *ECCV*, 2010. 2

[24] Z. Liu, L. Yuan, X. Tang, M. Uyttendaele, and J. Sun. Fast burst images denoising. *ACM Transactions on Graphics*, 33(6), 2014. 1, 2, 5

[25] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015. 4

[26] K. G. Lore, A. Akintayo, and S. Sarkar. LLNet: A deep autoencoder approach to natural low-light image enhancement. *Pattern Recognition*, 61, 2017. 2, 5

[27] A. Łoza, D. R. Bull, P. R. Hill, and A. M. Achim. Automatic contrast enhancement of low-light images based on local statistics of wavelet coefficients. *Digital Signal Processing*, 23(6), 2013. 2

[28] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman. Non-local sparse models for image restoration. In *ICCV*, 2009. 2

[29] H. Malm, M. Oskarsson, E. Warrant, P. Clarberg, J. Hasselgren, and C. Lejdfors. Adaptive enhancement and noise reduction in very low light-level video. In *ICCV*, 2007. 2

[30] S. Park, S. Yu, B. Moon, S. Ko, and J. Paik. Low-light image enhancement using variational optimization-based Retinex model. *IEEE Transactions on Consumer Electronics*, 63(2), 2017. 2

[31] G. Petschnigg, R. Szeliski, M. Agrawala, M. Cohen, H. Hoppe, and K. Toyama. Digital photography with flash and no-flash image pairs. *ACM Transactions on Graphics*, 23(3), 2004. 2

[32] T. Plötz and S. Roth. Benchmarking denoising algorithms with real photographs. In *CVPR*, 2017. 1, 2, 5

[33] J. Portilla, V. Strela, M. J. Wainwright, and E. P. Simoncelli. Image denoising using scale mixtures of Gaussians in the wavelet domain. *IEEE Transactions on Image Processing*, 12(11), 2003. 2

[34] T. Remez, O. Litany, R. Giryes, and A. M. Bronstein. Deep convolutional denoising of low-light images. *arXiv:1701.01687*, 2017. 1, 4

[35] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, 2015. 4, 5, 7

[36] L. I. Rudin, S. Osher, and E. Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D: Nonlinear Phenomena*, 60(1-4), 1992. 2

[37] W. Shi, J. Caballero, F. Huszár, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *CVPR*, 2016. 4

[38] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4), 2004. 6

[39] J. Xie, L. Xu, and E. Chen. Image denoising and inpainting with deep neural networks. In *NIPS*, 2012. 2

[40] L. Xu, J. Ren, Q. Yan, R. Liao, and J. Jia. Deep edge-aware filters. In *ICML*, 2015. 4

[41] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang. Beyond a Gaussian denoiser: Residual learning of deep CNN for image denoising. *IEEE Transactions on Image Processing*, 26(7), 2017. 2, 4

[42] X. Zhang, P. Shen, L. Luo, L. Zhang, and J. Song. Enhancement and noise reduction of very low light level images. In *ICPR*, 2012. 1

[43] H. Zhao, O. Gallo, I. Frosio, and J. Kautz. Loss functions for image restoration with neural networks. *IEEE Transactions on Computational Imaging*, 3(1), 2017. 7

BIJIT - BVICAM's International Journal of Information Technology
Bharati Vidyapeeth's Institute of Computer Applications and Management (BVICAM), New Delhi (INDIA)

A Torch Without Light: Low-Light Imaging for Mobile phones

Dipak Bange[1], Abhishek Gaikwad[2], Tejas Gajare[3], Aditya Khadse[4], Dr. Swati Shinde[5]

Figure 1: Image Captured in Extreme Low Light vs. Enhanced Image produced from our Convolutional Neural Network

Abstract* - *Photography used to be a hobby that required equipment such as a professional camera. Today, photography has evolved to be a daily activity conducted on an unprecedented scale due to the adoption of camera into smartphones. Low light photography has always been a weakness for these cameras. Shutter speed, light sensitivity can be controlled manually, but since smartphones have a fixed lens, aperture cannot be controlled. A high shutter speed provides better images in extreme low light, but requires the scene to be still in order to provide a high-quality image. On the other hand, various denoising algorithms have been used in the past to tackle and compensate noise due to high light sensitivity but cannot remove noise completely. In this article, we take a look at a possible solution for capturing extreme low light images using a smartphone. We compare previous results and assess possible future work.

Index Terms – Introduction, Existing Methods, Dataset, Using Transfer Learning, Training, Results, Conclusion, Future Work, References

NOMENCLATURE

DNG: Digital Negative Specification
ARW: Sony Alpha Raw
BM3D: Block-Matching and 3D Filtering
DSLR: Digital Single-Lens Reflex
ISO: International Organization of Standardization
RGGB: Red Green Green Blue

1.0 INTRODUCTION

Mobile phones have become a staple for every person's needs. Further, cameras in mobile phones have started to compete against professional cameras developed only for photography. But due to limited hardware and space constraints in a mobile phone, cameras are unable to outperform the professional cameras. The sensors used in mobile phones are far inferior to the ones found in professional cameras. Of the few spaces that mobile cameras lack, is low light images. Mobile cameras have been using flash for the past few years to enable the user to capture images in low light. But the images captured do not truly see what the human eyes had intended to see.

We aim to enable mobile cameras to be able to compete with the professional cameras through the use of artificial intelligence. The techniques we aim to use are based on software and hence do not add to the cost of building the mobile phone. Our method abandons the traditional image processing pipeline to move to a more data driven pipeline that is built using examples of similar past experiences rather than the image priors, which are details of images that are known prior to the image even being captured. Abandoning the traditional pipeline helps us to achieve a more data driven approach.

2.0 EXISTING METHODS

2.1 Manual Control

Digital cameras have had an important feature that has been taken for granted today. They have the ability to change their ISO on the fly. ISO here refers to the camera sensor's sensitivity to light. A high ISO means the sensor would be more sensitive to light and would increase the overall brightness of the image. This is the most basic method of improving visibility of a scene in low-light images.

Another manual control is to increase the exposure of the camera sensor. It can be controlled by two parameters: shutter speed and aperture. Shutter speed is the amount of time that the shutter remains open to capture the scene. A low shutter speed means that the shutter stays open for very small amount of time. The amount of light that enters the sensor is proportional to the time that the sensor is able to capture a light. For example, an image taken with shutter speed as 1/800 seconds would be darker than an image of the same scene taken with shutter speed as 1 second. A high exposure helps to get a brighter image as more photons can be captured by the camera sensor.

A camera works similar to the human eye. There exists an opening that allows light to enter a sensor that captures the light and convert it to an image. Aperture can be compared with the size of the pupil of the human eye. It controls the amount of light that can enter and be captured on the camera's sensor. A high aperture allows more light to enter and improve visibility in low-light images. Low-light images captured this using traditional pipeline is not perceptually good as it has a high amount of amount of noise. Several denoising techniques are used to further improve the visibility within the image. One of the most used techniques is 3D transform-domain filtering (BM3D).

2.2 BM3D

In 2007, K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian proposed a novel image denoising strategy based on an enhanced sparse representation in transform domain [1]. The idea was to use 2D image fragments found using matching of reference blocks and group them into 3D data arrays. They called these 3D data arrays groups. Further, they used collaborative filtering for the 3D groups. The filtering was divided into 3 successive steps: 3D transformation of a group, shrinkage of the transform spectrum and an inverse 3D transformation. The estimated 2D fragments were then returned to their locations. Due to multiple matching 2D fragments, they would be required to aggregate to take advantage of the redundancy.

BM3D has outperformed other recent denoising techniques when used on real low light images [2]. BM3D fails to outperform other data driven techniques as they still rely on processed image data rather than raw sensor data. The method also relies on one image of the scene. Another method tackled low-light imaging using burst photography.

2.3 Burst photography for High Dynamic Range and low-light imaging on mobile cameras

In 2016, Hasinoff et al [3] proposed a computational photography pipeline that can capture, align and merge a burst of frames to reduce noise and increase dynamic range. The pipeline takes in the burst of raw frames. It then aligns the burst to merge. Then it applies colour and tone mapping to produce a single full-resolution output.

The cameras in mobile phones allow auto exposure adjustment as the user moves the camera around. To get improved results in high dynamic range, they developed their own algorithm for adjusting the auto exposure within the mobile camera.

The major problem with burst images is that if the scene contains a moving object, it is at a different position in every frame. It becomes difficult to find the best position for the object. Hence, burst images introduce blurred objects, or ghosting if the objects move a lot faster than the shutter speed. To overcome this, they used temporal mean with alignment and merged it robustly to achieve almost zero blur compared to the temporal mean.

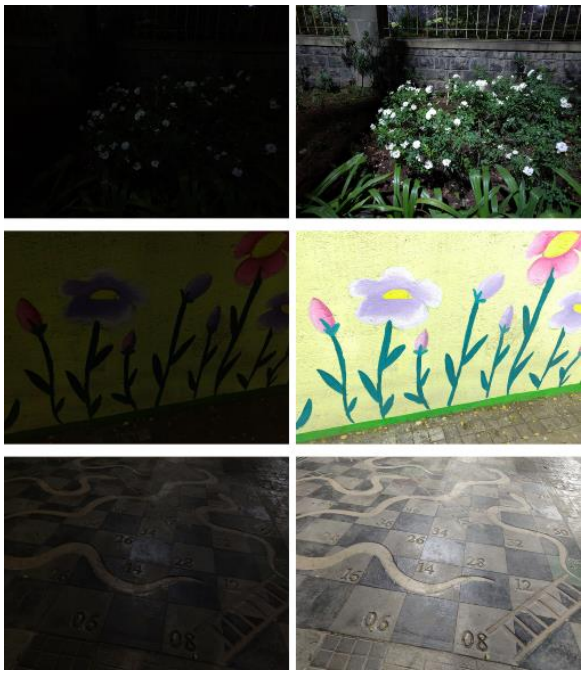
2.4 Learning to See in the Dark

At CVPR 2018, C Chen et al [4] proposed a pipeline for processing low-light images based on end-to-end training of a convolutional neural network on short exposure images and corresponding long exposure images. The neural network would be trained on raw sensor data rather than output images. This improves performances over other methods since this method does not rely on image priors, which are assumptions made about the image and its data prior to obtaining the image.

This method works by first training a fully convolutional neural network with short exposure images, which are captured in extreme low-light images. The architecture used in the neural network is U-net [5]. Further they used 2 cameras to form the dataset required for training the neural network. The See-in-the-Dark dataset has images from Sony α7S II and Fujifilm X-T2. There are 5094 short exposure images with their reference long exposure images.

3.0 DATASET

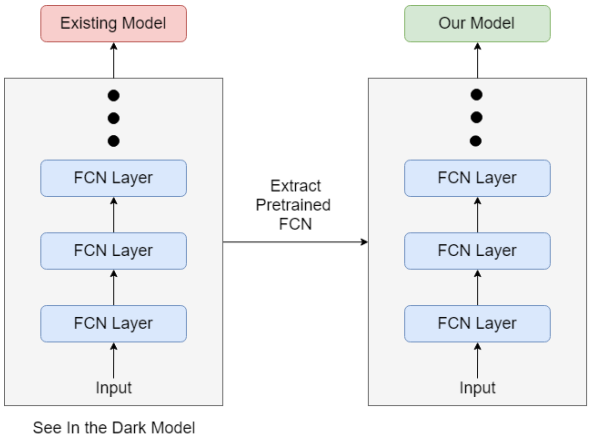
We collected a dataset of 592 images made up of 165 unique scenes. The dataset was collected using the camera of OnePlus 6 (16 MP Sony Exmor IMX519 (f/1.7, OIS + EIS) + 20 MP Sony Exmor IMX376K (f/1.7)). The images were stored in RAW format supported by smartphones, which is DNG format. The camera has a Bayer filter and has a black level of 63. The Bayer filter has pattern of RGGB.



Every unique scene has at least one short exposure image and a corresponding long exposure image. We captured the long exposure image by trying to capture the scene by changing the values until we reached an image that looked perceptually good.

The objective of collecting the dataset was to enable training of a model for the specific task of processing images captured on a smartphone. This dataset can be considered as the target task.

4.0 USING TRANSFER LEARNING



See In the Dark Model

Our current solution is based on the pretrained model trained on raw images on Sony α7S II camera (in .arw format) with black level of 512 and Bayer filter has pattern of RGGB. Using Transfer Learning [6] over this pretrained model we trained the network for the dataset of raw images (in .dng format) captured on OnePlus 6 to increase the accuracy and colors of the image for low light condition on mobile phones.

5.0 TRAINING

We train the fully convolutional neural network [7] using the L1 loss and the Adam optimizer. During training, the input to the network is the raw data of the short-exposed image and the ground truth is the corresponding long-exposure image in sRGB space (processed by libraw, a raw image processing library). We train one network for OnePlus 6. The amplification ratio is set to be the exposure difference between the input and reference images (e.g., x100, x250, or x300) for both training and testing. In each iteration, we randomly crop a 512×512 patch for training and apply random flipping and rotation for data augmentation. The learning rate is initially set to 10−4 and is reduced to 10−5 after 2000 epochs. Training proceeds for 5000 epochs.

6.0 METHOD

The working of the experiment can be broadly classified in three phases:

6.1 Pre-processing

The mobile application captures raw (DNG format) image. Rawpy is used to open the image. The image is in Bayer filter (R, G, G, B arranged in a 2x2 matrix for complete image) format and hence is converted into 4-channels (RGBG) of numpy array. Black level (63 for One Plus 6) is subtracted from the channels and divided by 216-1 for normalizing the values. The ground truth image converted to its 16-bit counterpart using Rawpy and pixel values are normalized.

6.2 Processing

After pre-processing, the 4 channels are passed to the fully convolutional neural network. Fully convolutional networks (FCNs) are a rich class of models that address many pixel wise tasks. FCNs for semantic segmentation dramatically improve accuracy by transferring pre trained classifier weights, fusing different layer representations, and learning end-to-end on whole images. End-to-end, pixel-to-pixel operation simultaneously simplifies and speeds up learning and inference [7]. U-net architecture is used for the network as it is more effective [4]. A random block of 512 x 512 is selected from image and fed to the network. The network then uses max pooling, ReLU and upsampling functions throughout the network [5]. Amplification

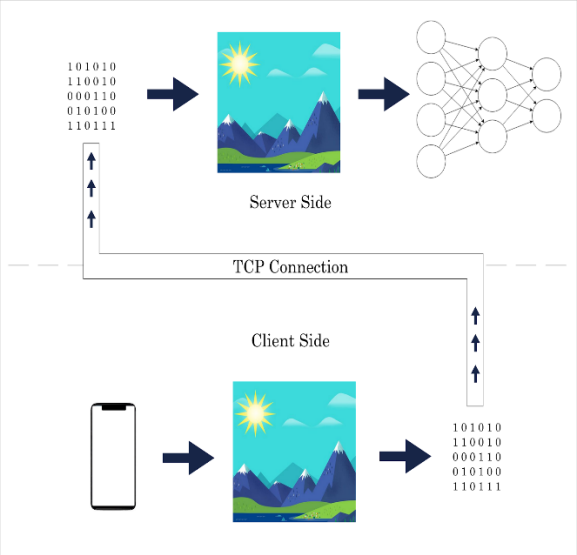
ratio is fed to the network in order to define how much brighter the image must be. The ratio we chose is depends on ground truth and input image exposure. The network outputs the processed image directly in sRGB space.

6.3 Post-Processing

Image pixel values converted in range 0-255.The image (sRGB space) obtained after processing is the enhanced version of the input image converted to .png extension. It is sent back again from the computer to the mobile phone where it can be viewed in the application. In our experiment have not used any post processing image enhancing algorithm as we wanted to deal with raw images as they are.

6.4 File Transfer To server for Processing

To provide user a seamless experience of capturing a visually perceivable image, our system shifts the system load onto server side rather than on client's mobile phone. Computationally intensive tasks are executed onto the server. The RAW image captured by the user is transferred to the server where the trained model exists. This file transfer is performed using two methods, 1) Peer-to-Peer File, Transfer 2) Http Multipart Upload



7.0 RESULTS

Figure 2: Input image shot on mobile phone (.dng) vs. Output of pretrained model on digital SLR cameras

Figure 3: Input image shot on mobile phone (.dng) vs. Output from model trained on mobile phone cameras

Figure 4: Input image shot on mobile phone (.dng) vs. Output from model trained on DSLR cameras and then on mobile phone cameras using transfer learning

8.0 CONCLUSION

Imaging in the dark, at sub-lux conditions presents a challenge that traditional pipeline fails to overcome. In this system, we present a methodology through which a visually enhanced image is retrieved from an image which is captured in extreme low light conditions. Computational processing of low-light images proves to be inefficient and far less effective for the extreme low light conditions under consideration. A data driven approach is implemented to overcome this scenario. Specifically, we train deep neural networks to learn the image processing pipeline for low light raw data, including colour transformations, demosaicing, noise reduction, and image enhancement.

Current system that is presented in the "Learning to see in the dark" paper [4] is the basis of our product. The deep neural network implemented in the aforementioned paper was trained using raw images captured through DSLR. This network is able to enhance images that are captured with the similar setup. To design a mobile application that shall enhance images captured from its comparatively smaller sensor, we enact the method of Transfer Learning. Through this methodology we train the existing trained model with our dataset captured through a mobile device. This provides us with a model suitable to satisfy our use case of enhancement of mobile camera images. We present a mobile application through which users can capture images in extreme low light conditions and enhance them into visually per perceivable images. This application is based on client-server architecture. The mobile application will upload the raw image onto the server for processing and the corresponding enhanced image will be retrieved.

9.0 FUTURE WORK

Current limitation of the presented pipeline is that the amplification ratio must be chosen externally. It would be useful to infer a good amplification ratio from the input, akin to Auto ISO. Furthermore, we currently assume that a dedicated network is trained for a given camera sensor. Our preliminary experiments with cross-sensor generalization are encouraging, and future work could further study the generalization abilities of low-light imaging networks. Another opportunity for future work is runtime optimization. The presented pipeline takes 0.66 seconds to process full-resolution One plus 6 images; this is not fast enough for real-time processing at full resolution, although a low-resolution preview can be produced in real time. We expect future work to yield further improvements in image quality, for example by systematically optimizing the network architecture and training procedure.

10.0 REFERENCES

Journal References

[1]. K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, *"Image denoising by sparse 3-D transform-domain collaborative filtering IEEE Transactions on Image Processing"*, 16(8), 2007

[2]. V. Jain and H. S. Seung, *"Natural image denoising with convolutional networks"*, NIPS, Feb 2008.

[3]. S. W. Hasino, D. Sharlet, R. Geiss, A. Adams, J. T. Barron, F. Kainz, J. Chen, and M. Levoy, *"Burst photography for high dynamic range and low-light imaging on mobile cameras"*, ACM Transactions on Graphics, 35(6), 2016.

[4]. Chen, Qifeng Chen, Jia Xu, Vladlen Koltun, *"Learning to See in the Dark"*, CVPR, 2018.

[5]. Olaf Ronneberger, Philipp Fischer, Thomas Bro, *"U-Net - Convolutional Networks for Biomedical Image Segmentation"*, May 2015.

[6]. Lisa A. Burke, Holly M. Hutchins, *"Training Transfer: An Integrative Literature Review"*, Sept 2007.

[7]. J. Long, E. Shelhamer, and T. Darrell, *"Fully convolutional networks for semantic segmentation"*, CVPR, 2015.