A PROJECT REPORT ON

Astronomical Image colorization and super-resolution using GANs

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

BACHELOR OF ENGINEERING (Computer Engineering)

BY

Shreyas Kalvankar	Exam No: B150134261
Hrushikesh Pandit	Exam No: B150134296
Pranav Parwate	Exam No: B150134299
Atharva Patil	Exam No: B150134303

DEPARTMENT OF COMPUTER ENGINEERING

K. K. Wagh Institute Of Engineering Education & Research

Hirabai Haridas Vidyanagari, Amrutdham, Panchavati, Nashik-422003
SAVITRIBAI PHULE PUNE UNIVERSITY

2020-2021

SAVITRIBAI PHULE PUNE UNIVERSITY

2020-2021

K. K. Wagh Institute Of Engineering Education & Research

CERTIFICATE

This is to certify that the Project Entitled

Astronomical Image colorization and super-resolution using GANs

Submitted by

Shreyas Kalvankar	Exam No: B150134261
Hrushikesh Pandit	Exam No: B150134296
Pranav Parwate	Exam No: B150134299
Atharva Patil	Exam No: B150134303

is a bonafide work carried out by Students under the supervision of Prof. Dr. S.M. Kamalapur and it is submitted towards the partial fulfilment of the requirement of Bachelor of Engineering (Computer Engineering) Project during academic year 2020-21.

Prof. Dr. S. M. Kamalapur
Internal Guide
Dept. of Computer Engg.

Prof. S. S. Sane
H.O.D
Dept. of Computer Engg.

Dr. K. N. Nandurkar
Principal

Place: Nashik

Date:

Acknowledgments

It gives us great pleasure in presenting the preliminary project report on ***Astronomical Image colorization and super-resolution using Generative Adversarial Networks***. We would like to take this opportunity to thank our internal guide Prof. Dr. S. M. Kamalapur for giving us all the help and guidance we needed. We are really grateful to her for her kind support. We would like to thank Prof. N. M. Shahane for providing us with his valuable guidance throughout.

We are also grateful to Prof. Dr. S.S. Sane, Head of Computer Engineering Department, K.K.Wagh Institute Of Engineering Education And Research, Nashik, for his indispensable support. We thank Prof. Dr. K.N. Nandurkar, Principal, K.K. Wagh Institute of Engineering Education and Research, Nashik, for his unwavering support. We are grateful to Gao Xian Peh and Kai Marshland whose report on the topic inspired us with the idea of using Generative Adversarial Networks in this particular domain.

Shreyas Kalvankar
Hrushikesh Pandit
Pranav Parwate
Atharva Patil
(B.E. Computer Engg.)

Abstract

Automated colorization of gray scale images has been subjected to much research within the computer vision and machine learning communities. Beyond simply being fascinating from an aesthetic and artificial intelligence perspective, such capability has broad practical applications. It is an area of research that possesses great potentials in applications: from black and white photo reconstruction, image augmentation, video restoration to image enhancement for improved interpretability.

Image downscaling is an innately lossy process. The principal objective of super resolution imaging is to reconstruct a low resolution image into a high resolution one based on a set of low-resolution images to rectify the limitations that existed while the procurement of the original low-resolution images. This is to insure better visualization and recognition for either scientific or non-scientific purposes. Even if an upscaling algorithm is particularly good, there will always be some amount of high frequency data lost from a downscale-upscale function performed on the image. Ultimately, even the best upscaling algorithms are unable to effectively reconstruct data that does not exist. Traditional methods for image upsampling rely on low-information, smooth interpolation between known pixels. Such methods can be treated as a convolution with a kernel encoding no information about the original image. A solution to the problem is by using Generative Adversarial Networks (GANs) to hallucinate high frequency data in a super scaled image that does not exist in the smaller image. Even though they increase the resolution of an image, they fail to produce the clarity desired in the super-resolution task. By using the above mentioned method, not a perfect reconstruction can be obtained albeit instead a rather plausible guess can be made at what the lost data might be, constrained to reality by a loss function penalizing deviations from the ground truth image.

A huge number of raw images are present unprocessed and unnoticed in the Hubble Legacy Archives. These raw images are typically black and white, low-resolution and unfit to be shared with the world. It takes huge amounts of hours to process them. This processing is necessary because it's difficult for astronomers to distinguish objects from the raw images. Random and synthetic noise from the sensors in the

telescope, changing optical characteristics in the system and noise from other bodies in the universe all make the processing further necessary. Furthermore, for the process of highlighting small features that ordinarily wouldn't be able to be picked out against noise of the image, we need colorization. The processing of the images is so time consuming that the images are rarely seen by human eyes. The problem is only likely to get worse. Not only is new data being continuously produced by Hubble Telescope, but new telescopes are soon to come online. A simplification of image processing by using artificial image colorization and super-resolution can be done in an automated fashion to make it easier for astronomers to visually identify and analyze objects in Hubble dataset.

INDEX

List of Figures

List of Tables

CHAPTER 1

INTRODUCTION

1.1 PROJECT IDEA

The idea of the project is to create a efficient mathematical model for image colorization and super resolution using Generative Adversarial Networks (GANs).

Having two networks compete will stimulate greater performance by the virtue of minimization of loss functions that traditional Convolutional Neural Network cant do

1.2 MOTIVATION OF THE PROJECT

This section summarizes different aspects of evolving technologies and drawbacks that served as a motivation.

- Image colorization seems to be evolving as computers get better and better at predicting missing variables. Different methods and techniques have been applied for colorizing images that have shown promising results
- Introduction of convolutional neural networks has made this task even more precise and accurate. Convoluting grayscale images to RGB provides unprecedented results with reference to visual inspection
- With the introduction of Generative Adversarial Networks, this particular task can be developed and modified to increase the efficiency and precision for colorizing images instead
- Furthermore, image upscaling is another problem that has been under research in the computer vision community. It has been studied and many approached have been developed to accurately predict the missing pixel values while upscaling an image
- Application of GANs to this discipline has successfully improved the performance and computers are getting better and better at predicting accurate missing pixel values and upscaling images many folds the original size
- All this computation power can be used for astronomical research by processing large data archives

- A large number of images lie dormant in most of the space survey data archives which never go through any kind of processing and are low resolution and black & white. These images could be processed automatically by an algorithm that will colorize and super-resolve the images which can make it easier for astronomers to visually inspect the images

1.3 PROBLEM DEFINITION AND SCOPE

1.3.1 Problem Statement

The problem can be subdivided based on the model approach. We aim to create an efficient model to colorize grayscale images that will transform the input tensor into an RGB colored output tensor. In the next part, we also aim to super-scale the tensor obtained. Thus, the next part can be summarized as making a model that will obtain a colorized image and upscale it n times the original size.

1.3.1.1 Goals and objectives

It is observed from the statement that the problem consists of multiple sub-problems. After careful reviewing, the problem can be divided into two sub-problems:

- Create an efficient model to colorize grayscale images
- Take a colorized image and upscale it n times the original size

According to the decomposed problem, we formulate the following goals and objectives.

- Auto-Colorization:
  - The first model will be given input a grayscale, low resolution image of dimensions $(64 \times 64 \times 1)$
  - The model will perform a series of mathematical operations that will increase the channel width of the image from 1 (single channel grayscale image) to 3 (RGB)

  - The output of the model will be a colorized version of the input image with dimensions $(64 \times 64 \times 3)$

- Upscaling/super-resolution:
  - The input to the model will be a colorized image of shape $(64 \times 64 \times 3)$
  - The model will increase the dimensions of the image from (64×64) to $((64 \cdot n) \times (64 \cdot n))$ by performing a series of upscaling operations and predicting information that may be lost while downscaling
  - The output of the model will be an upscaled RGB image with dimensions $((64 \cdot n) \times (64 \cdot n) \times 3)$

- The models may be combined to form a single model that will take a low resolution, grayscale image as its input and produce a high resolution, colorized image as its output

1.3.2 Statement of scope

According to the mentioned goals and objectives, the scope of the proposed solution is summarized as follows:

- The model will consist of neural networks implemented using deep learning frameworks that will accept images of input format *JPEG*
- The input will be grayscale images of size 64×64
- Input bounds:
  - Lower bound: $64 \times 64 \times 1$
  - Upper bound: no limit
- The output will be produced in two phases:
  - A colorized output of model 1 with shape $64 \times 64 \times 3$
  - A upscaled output of model 2 from the colorized output of model 1 with shape $(64 \cdot n) \times (64 \cdot n) \times 3$

- The model will:
  - take input black & white images
  - produce colorized images of the same size
  - produce upscaled images of size n times the input size (currently 64)
- The model will **not**:
  - take a colorized image as an input
  - take an image of size less than (64×64) in size
  - produce accurate upscaling or coloring albeit merely make a guess at what the lost values might be

1.3.3 Major Constraints

The problem statement is well defined to build a workable solution. However, the model proposed has several constraints. The major constraints are discussed in the following section:

- The astronomical image data required for training purposes is mostly raw. There exists no structured dataset that is already cleaned. The unavailability of a dataset is a major constraint for the project
- Scraped data from the archives is noisy and requires heavy processing and cleaning in order to be usable by the model
- The images available for download are of low resolution, which sets an upper bound on the maximal upscale factor
- The image data is large and needs high computation power to process
- The data needs to be cleaned manually as there exist no methods to automatically do this particular task
- The model involves neural networks which heavily rely on computation power for its training. The hardware required for training is not readily available because of absence of a workstation supporting heavy computations

- The training part requires large amount of memory
- Absence of an NVIDIA workstation GPU will slow down the training further

1.4 METHODOLOGIES OF PROBLEM SOLVING AND EFFICIENCY ISSUES

The proposed methodology can be segregated into several sub-categories. The details of each sub-category are discussed in the following section:

- Data gathering and processing
  - Data Scraping
    - Owing to unavailability of a dataset, raw data can be acquired by the means of web scraping
    - Images from the snapshots of entire night sky can be obtained in such a way from the Hubble Legacy Archive
  - Data Cleaning
    - The scraped data consists of snapshots of the entire night sky with 1 degree deviation of the telescope
    - This results in large amount of noisy, overexposed, irregular data images
    - This data needs to be cleaned manually before it can be used for any kind of study
- Image colorization
  - The problem of image colorization has been solved using multiple methodologies
  - Dahl (2016) used convolutional neural networks with residual encoders using the VGG16 architecture
  - Though the system performs extremely well in realistically colorizing various images, it consisted of $L2$ loss which was a function of the Euclidean distance between the pixel's blurred color channel value in the

target and predicted image

$$L2loss = \sum_{i=1}^{n} (y_{true} - y_{predicted})^2 \tag{1.1}$$

  - This is a regression based approach and the pixel-wise L2 loss will impose an averaging effect over all possible candidates and will result in dimmer and patchy colorization
  - Generative Adversarial Networks introduced by Goodfellow et al. (2014) use a minimax loss which is different than the $L2$ loss as it will choose a color to fill an area rather than averaging. This is similar to a classification based approach

- Image Upscaling
  - One of the most popular approach to image upscaling was sparse-coding. This approach assumes that images can be sparsely represented by a dictionary of atoms in some transformed domain (Jianchao Yang et al.; 2008). The dictionary is learned during the training process.
  - The main drawback for this was that the optimization algorithm was computationally expensive
  - Dong et. al explored super-resolution using convolutional neural network and calling it SRCNN (Dong et al.; 2014). They explained how CNN had many similarities to the sparse-coding-based super-resolution.
  - Kim et. al improved upon SRCNN's results using their very own model inspired from the VGG-net architecture(Kim et al.; 2016).
  - After the introduction of GANs, Ledig et. al applied them to super-resolution (SRGAN) using a network inspired by the ResNets (Ledig et al.; 2017; He et al.; 2015).
  - SR-GAN works well with for single image super-resolution as it also uses an intelligent content loss function that uses pre-trained VGG-net layers. However, Ledig et. al noted that further information could be

used if this network were to be adapted to a video, such as temporal information.

- A generative network, G, is meant to learn the underlying distribution of a data set, Y. For e.g. we can train a GAN over face images to generate images similar to those faces. With just a generative network however, we must visually assess the quality of network outputs and judge how we can adapt the network to produce more convincing results.

- With a discriminative network D, we can incorporate this tweaking directly into training. The discriminative network takes in both fabricated inputs generated by G and the real inputs from Y. It's sole purpose is to classify if the input has come from G or Y.

- The key idea is back propagation of the gradients from the results of D's classification to G so that G gets better at producing images and in turn fooling D.

- For the project, we split the data into two categories: X that serves as the data for the Y, which are its corresponding labels.

- G_1 takes in a low resolution $x \in X$ which is black & white and produces $\hat{y}$, a colorized version of x. The descriminator D, in turn takes in a colorized image and outputs the probability that the image comes from Y, instead of as outputs from G, $G(x)$. As such, if the discriminator is fooled by out generator, it should output a probability greater than 0.5 for the set of inputs coming from $G(x)$ and a probability less than 0.5 for images coming from Y.

- The same is the process for generator G_2 with the only difference being that the X is the set of colorized images but having low resolution and Y is the set of high resolution images that serve as the labels for underlying mapping of X. G_2 takes in the low resolution image $x \in X$ and produces $\hat{y}$ and the discriminator outputs a probability determining whether the image is super-resolved by G_2 or the ground truth images from Y.

1.5 SCENARIO IN WHICH MULTI-CORE, EMBEDDED AND DISTRIBUTED COMPUTING USED

A deep learning algorithm is a software construct that has certain steps that may be hardware intensive. Generative Adversarial Networks require huge amount of computing prowess to complete multiple passes of forward and backward propagation in order to train themselves. A network may consist of millions and billions of parameters which are associated with hundreds of thousands of graph nodes. To actually be able to train a network with more than a billion parameters, we need appropriately large amount of memory. Furthermore, the operations of forward and backward propagation are mathematical operations that adjust the parameters based on the gradient of the cost function to minimize the cost. This calculation, although heavy, is independent of each node and can be performed in a parallel framework. NVIDIA CuDA enabled GPUs have a CuDNN (CuDA Deep Neural Network) library that hooks the training algorithm onto the GPU memory for processing, deploying thousands and hundreds of thousand parallel threads to perform independent calculations of optimizing gradients. Such an infrastructure is expensive and requires a dedicated set up for running deep learning algorithms. For normal use cases, one can run into the problems of memory overflows while allocating tensors in the process of creation of graphs. In such cases, it is costly to buy more GPUs. One can make use of cloud services provided by Google Colab, AWS, Azure and more. These services can host runtimes that will allow users to run their deep learning algorithms over their hardware which will ensure fast and efficient training.

1.6 EXPECTED OUTCOME

The expected outcomes of the proposed methodology are as follows:

- An efficient mathematical model to be created which will describe mappings required to colorize and super-resolve low resolution grayscale images
- A brief albeit descriptive study of different approaches towards image colorization and super-resolution

- Study presenting the benefits of certain GAN architectures and their edge over other kinds of neural networks in image colorization and super-resolution

1.7 APPLICATIONS

Currently, given the number of the raw and unprocessed images in Hubble Legacy Archives, much of the images are not workable for scientific evaluation. The main application of building a GAN and automating the upscaling and colorization of these images is to help in visual classification for astronomers. Through a high resolution and colourized image, astronomical objects which would've been imperceptible to the human eye could be now visible for visual inspection. While upscaling is expected to address the poor quality of the original images, colourization will help distinguish astronomical objects and activites from the noise generated by various factors.

CHAPTER 2

LITERATURE SURVEY

2.1 IMAGE COLORIZATION

2.1.1 Hint Based Colorization

Levin et al. (n.d.) proposed using colorization hints from the user in a quadratic cost function which imposed that neighboring pixels in space-time with similar intensities should have similar colours. This was a simple but effective method but only had hints which were provided in form of imprecise colored scribbles on the grayscale input image. But with no additional information about the image, the method was able to efficiently generate high quality colorizations. Huang et al. (2005) addressed the color bleeding issue faced in this approach and solved it using adaptive edge detection. Yatziv and Sapiro (2006) used luminescence based weighting for hints to boost efficiency. Qu et al. (2006) extended the original cost function to apply color continuity over similar textures along with intensities.

Welsh et al. (2002) had proposed another approach that reduced the burden on the user by only requiring a full color example of an image with similar composition. It matched the texture and luminescence between the example and the target grayscale image and received realistic results as long as the example image was sufficiently similar.

Regardless of the scribble based or example based approach, the algorithms still needed sufficient human assistance in form of hand drawn or colorized images.

2.1.2 Deep Colorization

Owing to recent advances, the Convolutional Neural Networks are a de facto standard for solving image classification problems and their popularity continues to rise with continual improvements. CNNs are peculiar in their ability to learn and differentiate colors, patterns and shapes within an image and their ability to associate them with different classes.

Cheng et al. (2016) proposed a per pixel training for neural networks using DAISY (Tola et al.; 2008), and semantic (Long et al.; 2015) features to predict the chrominance value for each pixel, that used bilateral filtering to smooth out accidental image artifacts. With a large enough dataset, this method proved to be superior

to the example based techniques even with a simple Euclidean loss function against the ground truth values.

Finally, Dahl (2016) successfully implemented a system to automatically colorize black & white images using several ImageNet-trained layers from VGG-16 (Simonyan and Zisserman; 2015) and integrating them with auto-encoders that contained residual connections. These residual connections merged the outputs produced by the encoding VGG16 layers and the decoding portion of the network in the later stages. He et al. (2015) showed that deeper neural networks can be trained by reformulating layers to learn residual function with reference to layer inputs. Using this *Residual Connections*, He et al. (2015) created the *ResNets* that went as deep as 152 layers and won the 2015 ImageNet Challenge.

2.1.3 Generative Adversarial Networks

Goodfellow et al. (2014) introduced the adversarial framework that provides an approach to training a neural network which uses the generative distribution of $p_g(x)$ over the input data x.

Since it's inception in 2015, many extended works of GAN have been proposed over years including DCGAN (Radford et al.; 2016), Conditional-GAN (Mirza and Osindero; 2014), iGAN (Zhu et al.; 2018), Pix2Pix (Isola et al.; 2018).

Radford et al. (2016) applied the adversarial framework for training convolutional neural networks as generative models for images, demonstrating the viability of *deep convolutional generative adversarial networks*.

DCGAN is the standard architecture to generate images from random noise. Instead of generating images from random noise, Conditional-GAN (Mirza and Osindero; 2014) uses a condition to generate output image. For e.g. a grayscale image is the condition for colorization of image. Pix2Pix (Isola et al.; 2018) is a Conditional-GAN with images as the conditions. The network can learn a mapping from input image to output image and also learn a separate loss function to train this mapping. Pix2Pix is considered to be the state of the art architecture for image-image translation problems like colorization.

2.2 IMAGE UPSCALING

2.2.1 Frequency-domain-based SR image approach

TSAI (1984) proposed the frequency domain SR method, where SR computation was considered for the noise free low resolution images. They transformed the low resolution images into Discrete Fourier transform (DFT) and further combined it as per the relationship between the aliased DFT coefficient of the observed low resolution image and that of unknown high resolution image. Then the output is transformed back into the spatial domain where a higher resolution is now achieved.

While Frequency-domain-based SR extrapolates high frequeny information from the low resolution images and is thus useful, however they fall short in real world applications.

2.2.2 The interpolation based SR image approach

The interpolation-based SR approach constructs a high resolution image by casting all the low resolution images to the reference image and then combining all the information available from every image available. The method consists of the following three stages (i) the registration stage for aligning the low-resolution input images, (ii) the interpolation stage for producing a higher-resolution image, and (iii) the deblurring stage which enhances the reconstructed high-resolution image produced in the step ii).

However, as each low resolution image adds a few new details before finally deblurring them, this method cannot be used if only a single reference image is available.

2.2.3 Regularization-based SR image approach

Most known Bayesian-based SR approaches are maximum likelihood (ML) estimation approach and maximum a posterior (MAP) estimation approach.

While Tom and Katsaggelos (1996) proposed the first ML estimation based SR approach with the aim to find the ML estimation of high resolution image, some proposed a MAP estimation approach. MAP SR tries to takes into consideration the

prior image model to reflect the expectation of the unknown high resolution image.

2.2.4 Super Resolution - Generative Adversarial Networks (SR-GAN)

The Genrative Adversarial Network (Goodfellow et al.; 2014), has two neural networks, the Generator and the Discriminator. These networks compete with each other in a zero-sum game. Ledig et al. (2017) introduced SRGAN in 2017, which used a SRResNet to upscale images with an upscaling factor of 4x. SRGAN is currently the state of the art on public benchmark datasets.

CHAPTER 3

SOFTWARE REQUIREMENT SPECIFICATION

3.1 INTRODUCTION

3.1.1 Purpose and Scope of Document

The purpose of this project document is to create a comprehensive documentation about the methodology used in implementing the upscaling and colorization of the HSA images. The document explores the goals, objectives, resources, project plan, SRS and finally the summary and conclusion. It is hoped that this document would be beneficial in answering every query that any individual may have about this research project.

3.1.2 User profiles

Two actors have been defined in the Use case Diagram:
User: User preprocesses the image with the help of the GAN and facilitates the whole process.
GAN: It takes the input images, colorizes the image and feeds it to another GAN. The second GAN then upscales the image and a final output image is generated.

3.1.3 Use-cases

Sr No.	Use Case	Description	Actors	Assumptions
1	Input image	Image is loaded	User	Image of size 64x64 and format jpg/jpeg is uploaded
2	Colourising	GAN colourises the image	GAN	The image is colourised from its black and white format
3	Upscaling	GAN upscales the image	GAN	The image is upscaled to resolution 768x768 and is not pixelated.

Table 3.2: Use Cases

3.1.4 Use Case View

Use Case Diagram. Example is given below

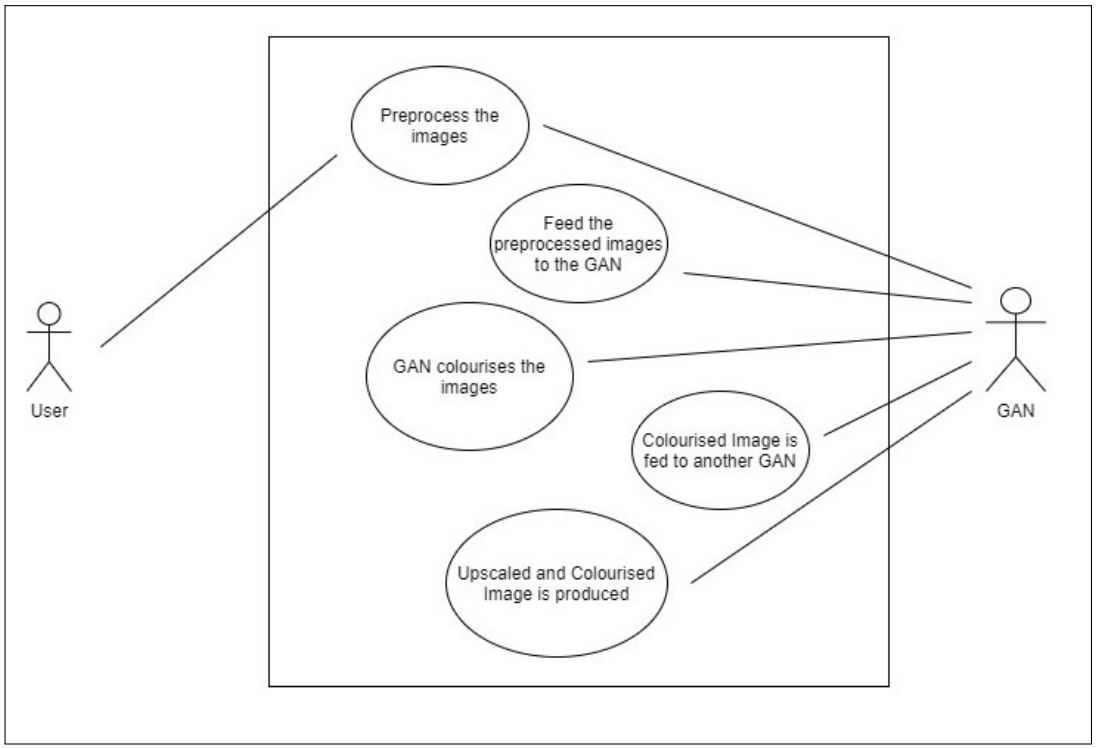


Figure 3.1: Use case diagram

3.2 SYSTEM REQUIREMENTS

3.2.1 Hardware Resources Required

The project is based Machine Learning, and the use of Tensorflow-GPU brought forward the need for a very high end hardware. Google Colab (or Colaboratory) is a free Jupyter notebook environment offered by Google which runs notebooks from Python kernels and uses Google Drive for storage.

Sr. No.	Parameter	Minimum Requirement	Justification
1	GPU type	NVIDIA CuDA enabled GPU	Training the model
2	GPU memory	>6 GB	Batch training

Table 3.3: Hardware Requirements

3.2.2 Software Resources Required

Platform :

1. Operating System: Windows/Linus
2. IDE: Jupyter Notebook
3. Programming Language: python3, javascript
4. Frameworks: Node.js, Tensorflow, plotting libraries, openCV

3.2.3 Functional Requirements:

- Data Scraping and Processing: The data pipeline should output a usable image which is RGB and has a minimum dimension of 256×256
- Colorization: The system should output a colorized version of the given grayscale input image
- Super-Resolution: The system should output a 4 times upscaled image of the input

3.2.4 Non Functional Requirements:

- Accessibility: The dataset generated would be accessible for everyone to access in open source.
- Availability: The code would be available on github
- Performance: The performance of all the GANs is recorded.

CHAPTER 4

PROJECT PLAN

4.1 PROJECT ESTIMATES

Use Waterfall model and associated streams derived from assignments 1,2, 3, 4 and 5(Annex A and B) for estimation.

4.1.1 Reconciled Estimates

4.1.1.1 Cost Estimate

The model followed is the Constructive Cost Model (COCOMO) for estimating the efforts required in the completion of the porject. Like all estimation models, the COCOMO model requires sizing information. This information can be specified in the form of:

- Object Point
- Function Point(FP)
- Lines of Source Code(KLOC)

For our project, sizing information in the form of Lines of Source Code is used. The total lines of code,
KLOC = 1000
Equations: The initial effort(Ei) in man-months is calculated using equations:

$$E = ax(KLOC)^b$$

where, a = 3.0, b = 1.12, for a semi-detached project E = Efforts in person-hours
E = 4.5 PM

$$D = ax(E)^b$$

Where, a = 2.5,
b = 0.35, for a semi-detached project
D = Duration of Project in months

D = 12 Months

4.1.1.2 Time Estimates

$$C = D * Cp * hrs$$

Where, C = Cost of project

D = Duration in Hours

Cp = Cost incurred per person-hour

hrs = hours

Total of 4.5 person-months are required to complete the project successfully.

Duration of Project D = 12 Months

The approximate duration of the project is 12 months

4.1.2 Project Resources

Project resources [People, Hardware, Software, Tools and other resources] based on Memory Sharing, IPC, and Concurrency derived using appendices to be referred.

4.2 RISK MANAGEMENT

This section discusses Project risks and the approach to managing them.

4.2.1 Risk Identification

1. Dataset needs to be processed in order to get clean data
2. Vanishing Gradients
3. Mode Collapse
4. Failure to Converge

4.2.2 Risk Analysis

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

ID	Risk Description	Probability	Impact		
			Schedule	Quality	Overall
1	Garbage images in dataset	Medium	Low	High	High
2	Vanishing Gradients	Low	Low	High	High

Table 4.2: Risk Table

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

Table 4.3: Risk Probability definitions (Pressman; 1992)

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

Table 4.5: Risk Impact definitions (Pressman; 1992)

4.2.3 Overview of Risk Mitigation, Monitoring, Management

Following are the details for each risk

Risk ID	1
Risk Description	There are Garbage images in dataset which can lead to improper training of discriminator. It can lead to undesired outputs
Category	Pre processing.
Source	Software requirement Specification document.
Probability	Low
Impact	High
Response	Mitigate
Strategy	To manually remove garbage images from the dataset
Risk Status	Occurred

Risk ID	2
Risk Description	If discriminator is too good , then generator training can fail due to Vanishing gradients
Category	Requirements
Source	Software Design Specification documentation review.
Probability	Low
Impact	High
Response	Mitigate
Strategy	Wasserstein loos and Modifed minimax loss are desigend to prevent vanishing gradients.
Risk Status	Identified

4.3 PROJECT SCHEDULE

4.3.1 Project task set

Major Tasks in the Project stages are:

- Task 1: Collecting Dataset
- Task 2: Cleaning Dataset
- Task 3: Creating GAN model of image colorization and super-resolution

- Task 4: Training the GAN model
- Task 5: Fine tuning the GAN model

4.4 TEAM ORGANIZATION

- Team of 4 members
- 1 Project guide
- 1 Project coordinator

4.4.1 Team structure

Team of 4 members

4.4.2 Management reporting and communication

- Weekly meeting with guide about the work
- Weekly meeting among team for updates and work distribution
- Daily updates on work progress

CHAPTER 5

SYSTEM DESIGN

5.1 INTRODUCTION

The project is largely inspired by Christian Ledig's SRGAN paper (Ledig et al.; 2017) and Dong et al. (2014)'s implementation of SRGANs using Tensorflow. Dahl (2016)'s introduction of residual encoding using VGG architecture and adaptation of GANs as conditional GANs by Mirza and Osindero (2014) proved to be quite effective for implementing colorization of images. We provide detailed architectural design for each respective GANs and other networks that it will be compared with.

5.2 ARCHITECTURAL DESIGN

Generative Adversarial Networks (GANs) have two competing neural network models. The generator takes in the input and generates fake images. The discriminator gets the image from both the generator and the label along with the grayscale image and it determines which pair contains the real colored image. During training, the generator and the discriminator are playing a continuous game. At each iteration, generator produces a more realistic photo, while the discriminator gets better at distinguishing the fake photos. Trained in a minimax fashion, the goal is to train a generator that produces data that is indistinguishable from the real data.

5.2.1 Image Colorization

Both the generator and discriminator are conditioned on the input x with conditional GAN. Let the generator be parameterized by θ_g and the discriminator be parameterized by θ_d. The minimax objective function can be defined as:

$$\min_{\theta_g}\max_{\theta_d}\left[\mathbb{E}_{x,y\sim p_{data}}\log D_{\theta_d}(x,y)+E_{x\sim p_{data}}\log(1-D_{\theta_d}(x,G_{\theta_g}(x))\right]$$

Where, G_{θ_g} is the output of the generator and D_{θ_d} is the output of the discriminator. We're currently not introducing any noise in our generator to keep things simple for the time being. Also, we consider $L1$ difference between input x and output y in generator. On each iteration, the discriminator would maximize θ_d according

to the above expression and generator would minimize θ_g in the following way:

$$\min_{\theta_g} \left[-\log(D_{\theta_d}(x, G_{\theta_g}(x))) + \lambda \| G_{\theta_g}(x) - y \|_1 \right]$$

With GAN, if the discriminator considers the pair of images generated by the generator to be a fake photo (not well colored), the loss will be back-propagated through discriminator and through generator. Therefore, generator can learn how to color the image correctly. At the final iteration, the parameters θ_g will be used in our generator to color grayscale images.

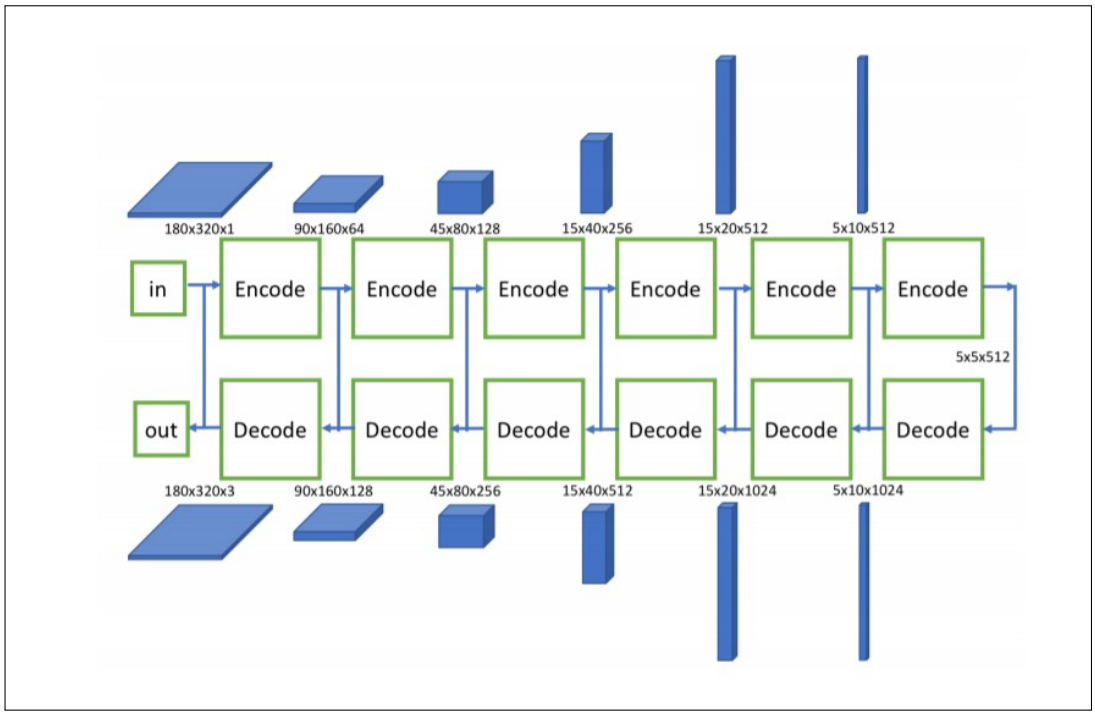


Figure 5.1: Encoder-Decoder ConvNets in Generator

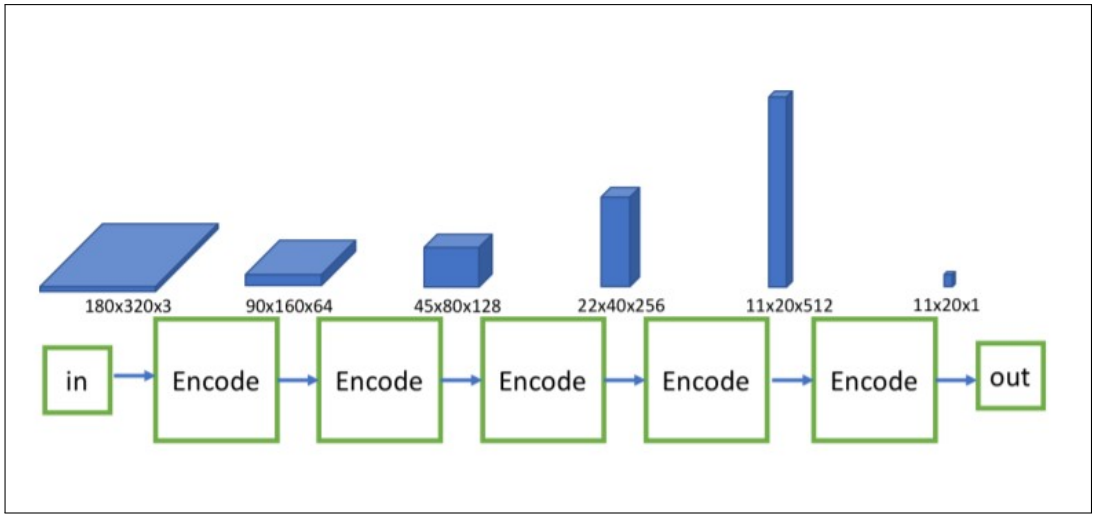


Figure 5.2: Discriminator

5.2.2 Image Super-resolution

We use the SRResNet as the generator in the SRGAN model as used by Ledig et al. (2017). It contains both the residual blocks and the skip connections, as seen in Figure 5.3. Within each residual block, there are two convolution layers followed by a Batch Normalization layer and a parametric ReLU layer. Finally, the image is then upscaled 4 times using two sub-pixel convolution layers (Shi et al.; 2016).

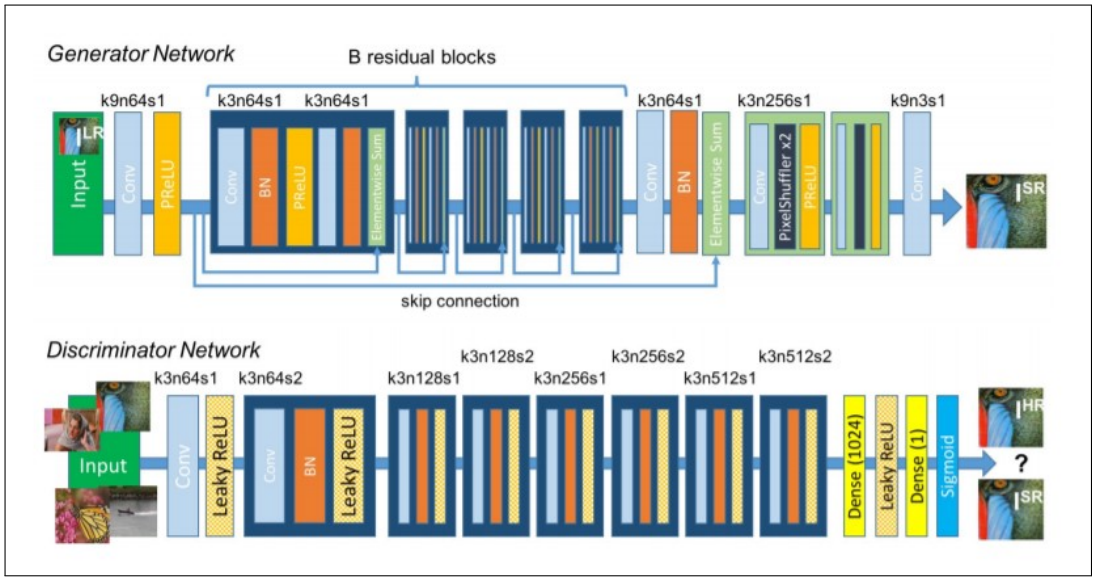


Figure 5.3: SRGAN model: SRResNet generator and discriminator

The goal of the generator is to produce high resolution images that will fool the discriminator of the GAN into thinking that it is receiving real instead of fake images.

On the other hand the discriminator's goal is to classify the images it has received as either real images or generated images from the generator. The GANs objective function is a minimax game as mentioned in the previous section. We define the minimax function for this task with trivial changes in notation and express it as:

$$\min_{\theta_g}\max_{\theta_d}\mathbb{E}_{I^{HR}\sim p_{train}(I^{HR})}\left[\log D_{\theta_d}(I^{HR})\right]+\mathbb{E}_{I^{LR}\sim p_g(I^{LR})}\left[\log(1-D_{\theta_d}(G_{\theta_g}(I^{LR})))\right]$$

where, $I^{HR}p_{train}(I^{HR})$ are the high resolution images. $I^{LR}p_g(I^{LR})$ are the input low resolution images, G_{θ_g} is the output of the generator and D_{θ_d} is the output of the discriminator. We use the perpetual loss function for VGG based content losses introduced by Ledig et al. (2017) which is a weighted sum of a content loss l^{SR}_X and an adversarial loss component $(10^{-3}l^{SR}_{Gen})$.

For the content loss, we aim to use the VGG loss introduced by Ledig et al. (2017) which is the euclidean distance between the feature representations of a reconstructed image $G_{\theta_g}(I^{LR})$ and the reference image I^{HR}:

$$l^{SR}_{VGG_{i,j}}=\frac{1}{W_{i,j}H_{i,j}}\sum_{x=1}^{W_{i,j}}\sum_{y=1}^{H_{i,j}}\left(\phi_{i,j}(I^{HR})_{x,y}-\phi_{i,j}(G_{\theta_g}(I^{LR}))_{x,y}\right)^2$$

where $W_{i,j}$ and $H_{i,j}$ represent the dimensions of the respective feature maps within VGG19 network. The adversarial generative loss l^{SR}_{Gen} is defined on the probabilities of the discriminator $D_{\theta_d}(G_{\theta_g}(I^{LR}))$ over all the training samples as:

$$l^{SR}_{Gen}=\sum_{n=1}^{N}-\log D_{\theta_d}(G_{\theta_g}(I^{LR}))$$

$D_{\theta_d}(G_{\theta_g}(I^{LR}))$ is the probability that the reconstructed image $G_{\theta_g}(I^{LR}))$ is a natural HR image. For beter gradient behavior, we minimize $-\log D_{\theta_d}(G_{\theta_g}(I^{LR}))$ instead of $\log\left[1-D_{\theta_d}(G_{\theta_g}(I^{LR}))\right]$.

5.3 MATHEMATICAL MODEL

5.3.1 Artificial Neural Networks

Figure 5.4 shows a schematic of the simplest multi-layer perceptron network, i.e. a feed-forward neural network. The network is composed of an input layer, hidden layers and output layer. Define a_i^ℓ as the i^{th} neuron of the ℓ^{th} layer and $a_j^{\ell+1}$ as the j^{th} neuron of the $(\ell+1)^{th}$ layer and w_{ij}^ℓ, b_j^ℓ is the weight and bias connecting the two neurons, then the output of the ℓ^{th} layer is given by:

$$a_j^{\ell+1} = g(\sum_{i \in N^\ell} (w_{ij}^\ell a_i^\ell + b_j^\ell)) \tag{5.1}$$

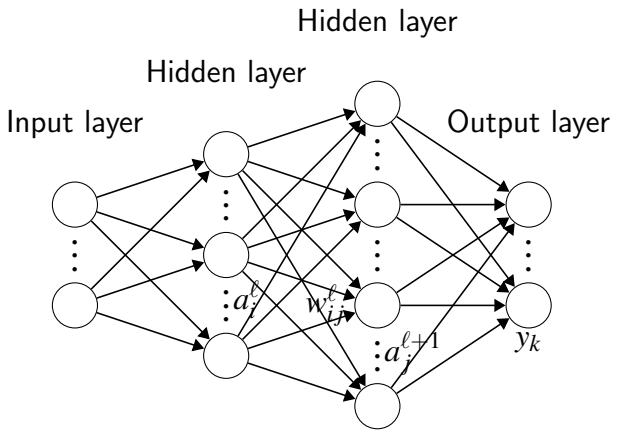


Figure 5.4: A feed forward neural network

The loss is calculated using a loss function such as cross entropy function especially for binary classification.

$$loss(y', y) = -y \log y' - (1-y) \log (1-y'). \tag{5.2}$$

where $y' \in \{0,1\}, y \in (0,1)$. This objective is to minimize this cross entropy over a batch of all training data. This is done using gradient descent where the parameters viz. weights and biases are updated to reduce the overall cross entropy loss.

5.3.2 Convolutional Neural Networks

A convolution is a linear operation that can be viewed as a multiplication or dot product of matrices. The input is a tensor of shape *height* x *width* x *channels* and

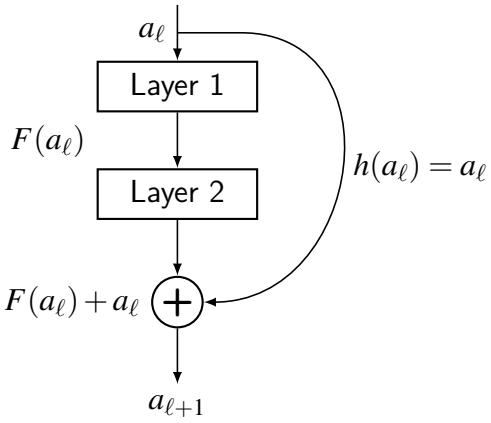


Figure 5.5: Residual Block

the convolution operation abstracts the image to a feature map (also called a kernel) of shape *kernelsize* x *kernelsize* x *kernelchannels*. The layers can be computed by:

$$a_j^{\ell+1} = g(\sum_{i \in F_j} (a_i^\ell \times k_{ij}^{\ell+1} + b_j^{\ell+1})) \tag{5.3}$$

where ℓ is the layer, g is the activation function ReLU, F_j is the receptive field, k is the convolutional kernel and b is the bias.

5.3.3 Residual Networks

Figure 5.5 shows a building block of a residual network. The residual blocks can be expressed mathematically as follows. Let $h(a)$ be an underlying mapping that is to be fit by a set of layers, where a is the input to these layers. If we hypothesize that multiple non-linear transformations by these layers can approximate the layer functions, then one can also hypothesize that a similar approximation can be made for the residual functions, i.e. $h(a) - a$, which have the same input and output dimensions. So instead of letting the underlying mapping be $h(a)$ we approximate a residual function $F(a) = h(a) - a$. Thus, the actual function becomes $h(a) = F(a) + a$. We can achieve this approximation in a feed forward neural network using a series of skip connections that perform identity mapping and jump over a few layers. Adding the outputs of these two connections gives the final output layer. Thus, the residual unit can be defined as,

$$a_{\ell+1} = h(a_\ell) + F(a_\ell, W_\ell) \tag{5.4}$$

where $a_{\ell+1}$ and a_ℓ represent the input and output for the ℓ^{th} layer and F is the residual function. The h denotes the operation in the identity mapping. The output of the layer is generally processed using an activation function such as ReLU. Let f be the ReLU activation and replacing F with the definition of feed forward activation, the residual block thus can be defined as,

$$a_{\ell+1} = f(h(a_\ell) + W_2 f(W_1 a_\ell)) \tag{5.5}$$

For the sake of simplicity, we consider no operation being performed in the identity mapping. Thus, equation 5.4 and equation 5.5 become,

$$a_{\ell+1} = f(a_\ell + F(a_\ell, W_\ell)) \tag{5.6}$$

5.3.4 Generative Adversarial Networks

A generative network, G, is supposed to learn the underlying distribution of a latent space, Y. Instead of visually assessing the quality of network outputs and judge how we can adapt the network to produce convincing results, we incorporate automatic tweaking during training by introducing a discriminative network D. The network D takes in both the fabricated outputs generated by G and real inputs from the underlying distribution Y. The network produces a probability of the image belonging to the real or fabricated space.

Let $x \in X$ be a low resolution/grayscale image and $y \in Y$ be it's underlying distribution from the latent space Y. Generator G takes in input x and produces an output $\hat{y}$. We define the mapping $x \to \hat{y}$ in the following manner:

$$G(x) = \hat{y}$$

The discriminative network D is fed the fabricated mapping $x \to \hat{y}$ and the underlying distribution of x i.e. $y \in Y$. The network D then produces a result that is a probability distribution of the input space indicating the class of the image that it

thinks the input belongs to. We define this as:

$$D\big(G(x), y\big) = p$$

where $p \in (0,1)$ is the probability that the image is fabricated or real. With conditional GAN, both generator and discriminator are conditioning on the input x. Let the generator be parameterized by θ_g and the discriminator be parameterized by θ_d. The minimax objective function can be defined as:

$$\min_{\theta_g} \max_{\theta_d} \Big[\mathbb{E}_{x,y\sim p_{data}} \log D_{\theta_d}(x,y) + E_{x\sim p_{data}} \log(1 - D_{\theta_d}(x, G_{\theta_g}(x)) \Big]$$

Where, G_{θ_g} is the output of the generator and D_{θ_d} is the output of the discriminator. We're currently not introducing any noise in our generator to keep things simple for the time being. Also, we consider $L1$ difference between input x and output y in generator. On each iteration, the discriminator would maximize θ_d according to the above expression and generator would minimize θ_g in the following way:

$$\min_{\theta_g} \Big[-\log(D_{\theta_d}(x, G_{\theta_g}(x))) + \lambda \| G_{\theta_g}(x) - y \|_1 \Big]$$

5.4 UML DIAGRAMS

5.4.1 Activity Diagram:

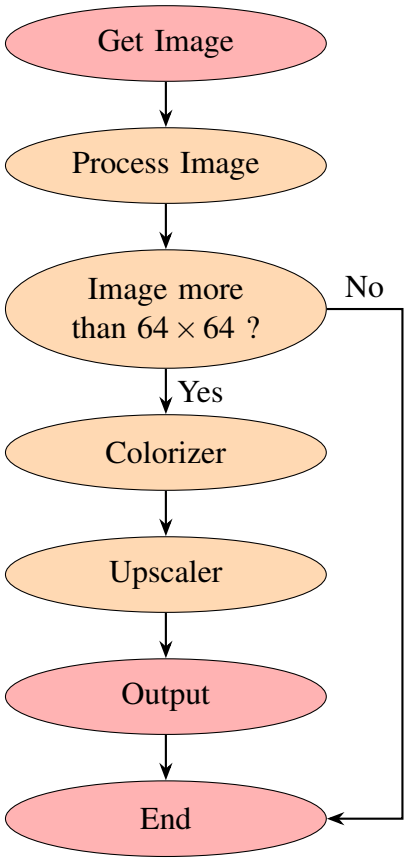


Figure 5.6: Activity Diagram

5.4.2 State Transition Diagram:

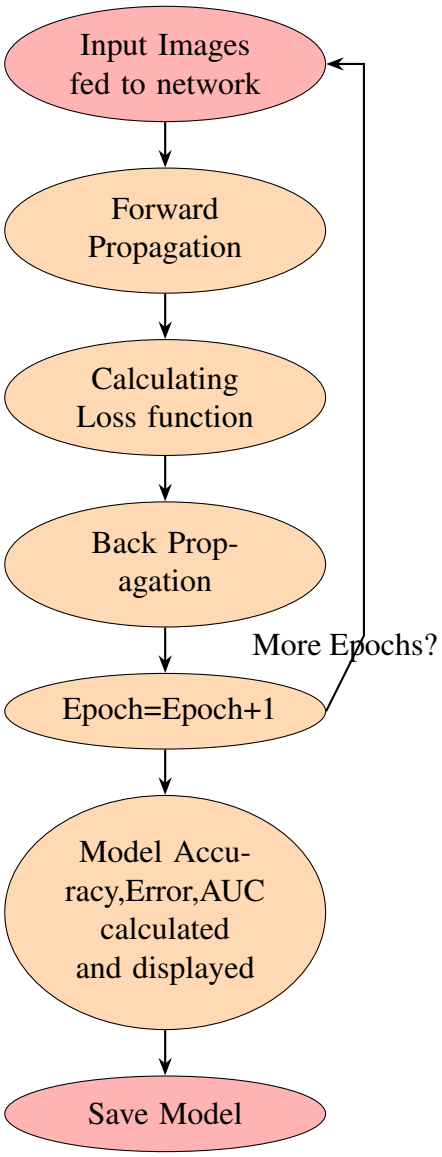


Figure 5.7: State Transition Diagram

CHAPTER 6

PROJECT IMPLEMENTATION

6.1 OVERVIEW OF PROJECT MODULES

To gather data, the primary module implemented was a data scraper written in node.js which uses puppeteer(headless chrome) to scrape the entire HLA(Hubble Legacy Archive) with the described RA and Dec co-ordinates of the night sky. The module ensured that, out of the entire data available in the Legacy Archive, a certain format of images would be downloaded and stored appropriately. After gathering the data, it was apparent that there were many other processed images present on the Hubble main and other citizen science projects conducted with Hubble Space Telescope. To scrape these, a python script using selenium was written so as to efficiently scrape processed images which had a minimum dimensions of $256 \times 256 \times 3$. With the data gathered, the main components, i.e. the models were built.

The project entails a standard set of GAN modules, broadly, a data pre-processing pipeline, a training pipeline, intermediate state logging and a testing pipeline.

- **Data Pre-processing**
  - The data pre-processing pipeline is an important part of the data pipeline that feeds the data to the training pipeline
  - The images, being too large to be loaded into the memory simultaneously, had to be loaded through a data generator
  - We implement an instance of `Keras Image Data Generator` to convert the raw data into training data
  - We create two *data generators*, one for generating gray scale images and the other for generating subsequent RGB image
  - We create another data pre-processor that converts the given RGB image into L*a*b color space
  - The images are then resized to a dimension of 64×64
  - We get a vector of image batches which will be the input to the training pipeline
- **Training Pipeline**

  - The training pipeline consists of multiple phases
  - Initially, we build the basic neural networks for generator and discriminator networks, i.e. a tensor graph which will function as a computation graph for forward and backward propagation
  - The next step is selection of an optimizing algorithm
  - After selection of optimizer, a training pipeline is implemented which includes running a session of forward and backward propagation throughout the computation graph and optimizing the trainable variables of the networks

- **Intermediate state logging**
  - Deep Learning models use up a lot of computational resources and are time consuming to train
  - Thus, it is essential to set up methods that will extract intermediate states of the model while it is still in the training loop
  - These methods are often termed as 'callbacks' and we implement different callbacks to ensure availability of training history, weights and intermediate results during documentation
  - A csv logger logs the metrics at the end of each epoch into a csv file for model evaluation
  - The model weights are saved if the loss metric improves than the earlier epoch. We implement a mechanism for early stopping where the training loop exits if the model appears to have converged

- **Testing pipeline**
  - Model testing is performed on a testing dataset that is different than the original dataset
  - Testing pipeline is implemented by using the saved weights of the trained model and the images are processed through the entire dataset
  - We evaluate the images qualitatively as well as quantitatively using visual inspection and distance metrics

6.2 TOOLS AND TECHNOLOGY USED

We use the following tools:

- Data gathering: Node.js, puppeteer, selenium
- Data Processing: openCV, Pillow
- Pre-processing & input pipeline: numpy, tensorflow-keras
- Model building & training: tensorflow 2.x
- Model evaluation: tensorflow, numpy
- Visualization: matplotlib, Pillow

6.3 ALGORITHM DETAILS

6.3.1 Generative Networks

A generative model is a class of statistical models that are a contrast to the classifier models dubbed as discriminative models. Informally speaking, a generative model creates new data instances based on the distribution of the data itself. Generative models tackle a more difficult task, i.e. to model data. A generative model might capture correlations among underlying distribution and draw conclusions. Goodfellow et al. (2014) proposed the most famous type of generative model, i.e. Generative Adversarial Network, which is composed of two smaller networks called a discriminator and a generator. The generator's task is to create fake images that convince the discriminator and the discriminator usually classifies between real and fake data.

6.3.2 Adversarial Networks

With two networks that are neural networks, the adversarial modeling framework is the most straightforward to apply. A distribution p_g over data x is called the generator's distribution. To learn this distribution, we define some prior noise on the input variables $p_z(z)$ and represent a mapping to a space with $G(z;\theta_g)$, where G is a differential function of neural network parameterized by θ_g. Another neural network

$D(x;\theta_d)$ is defined such that it provides a scalar output, representing the probability that data x comes from the input distribution rather than p_g. The goal is to train D to maximize the probability of assigning the correct label to both training examples and samples from G. The network G is trained to minimize $log(1-D(G(z))$.

Algorithm 1: Minibatch stochastic gradient descent training of generative adversarial networks. The number of steps to apply to the discriminator, k, is a hyperparameter.

for *number of training examples* **do**
  for *k steps* **do**
    - Sample minibatch of m noise samples $\{z^{(1)} \dots z^{(m)}\}$ from noise prior to $p_g(z)$.
    - Sample minibatch of m examples $\{x^{(1)} \dots x^{(m)}\}$ from data generating distribution $p_{data}(x)$.
    - Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[logD\left(x^{(i)}\right) + log\left(1 - D\left(G\left(z^{(i)}\right)\right)\right)\right]$$

  end
  - Sample minibatch of m noise samples $\{z^{(1)} \dots z^{(m)}\}$ from noise prior to $p_g(z)$.
  - Update generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} log\left(1 - D\left(G\left(z^{(i)}\right)\right)\right)$$

end

6.3.3 Conditional GANs

As mentioned earlier, GANs are generative models that learn a mapping from a random noise vector z to output p_g. The mapping from vector z to an output image y can be represented as $G : z \rightarrow y$ (Goodfellow et al.; 2014). Isola et al. (2018) presented a new approach for GANs called Conditional GANs. Instead of learning just from a random noise vector, conditional GANs learn a mapping from input x and the vector z to y such that $G : \{x,z\} \rightarrow y$. By doing this, we train the discriminator to do as well as possible to detect the generator's fake outputs whereas the generator is trained to produce results that are almost indistinguishable from the real data. The

objective function of conditional GAN can be expressed as:

$$\min_G \max_D \Big[\mathbb{E}_{x,y}\big[\log D(x,y)\big] + E_{x,z}\big[\log(1-D(x,G(x,z))\big]\Big]$$

6.4 METHODOLOGY

6.4.1 Image Color space

An RGB image is essentially a rank 3 tensor of height, width and color where the last axis contains the color data of our image. The data is represented in RGB color space which has 3 numbers for every pixel indicating the amount of *Red, Green, and Blue* values the pixel has. Figure 6.1 shows that in the left part of the "main image" has blue color so the blue channel of the image has higher values and turns dark.

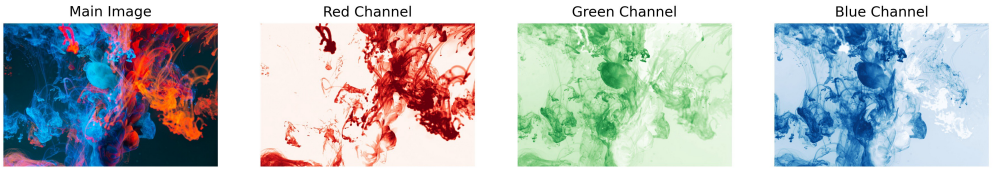


Figure 6.1: Reg, Green and Blue channels of an image

In **L*a*b** color space, we have three numbers for each pixel but these numbers have different meanings. The first channel, L, encodes the **Lightness** of each pixel and when we visualize this channel it appears as a black and white image. The ***a and *b** channels encode how much **green-red** and **yellow-blue** each pixel is, respectively. In the following image you can see each channel of L*a*b color space separately.

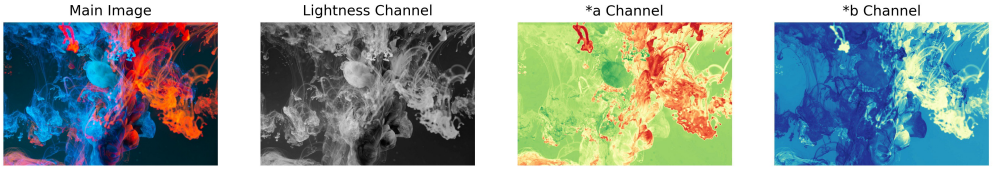


Figure 6.2: Lightness, *a and *b channels of the L*a*b colorspace

In most of the papers listed in the Literature Survey, the L*a*b color space is widely used instead of RGB to train the models. Intuitively, to the train a model for colorization, we should give it a grayscale image and we hope that it colors it. In the L*a*b colorspace, the model is fed the L channel, which is essentially a grayscale image, and we perform computations to predict the other two channels (*a and *b). After the predictions, we concatenate the channels and get a colorful image. In case of RGB, there is a need to explicitly convert the three channels down to 1 to make it a grayscale image and then feed it to the network hoping that it predicts three numbers per pixel. This is an unstable task due to sheer increase in volume of combinations from two numbers to three.

We train models using both color spaces and empirically show the significance of the L*a*b colorspace.

6.4.2 Transferred learning and model tweaking

Isola et al. (2018) proposed a general solution to many image-to-image translation tasks. We propose a similar methodology as proposed in the paper with a few minor tweaks, so as to significantly reduce the amount of training data needed and minimize the training time to achieve similar results. Initially, we use a U-net for the generator. The *encoder* of our U-net is a pre-trained ResNet-18 network with half of it's layers clipped off so as to get feature abstractions of the middle layers. Ledig et al. (2017) showed that training the generator separately in a supervised, deterministic manner helps it generalize the mapping from the input space to outputs. The idea solves the problem of *"The blind leading the blind"* that is persistent in most image-to-image translation tasks to date.

We decide to use the Common Objects in Context (COCO) dataset to pre-train our generator independently using the imagenet weights. This training will be supervised and the loss function used is *mean absolute error* or the *L1 Norm* from the target image. Even though trained deterministically,the problem of rectifying incorrect predictions still persists due to constraints over the convergence of loss metric. To combat this, we use this trained generator and train it once again in an adversarial fashion to generalize it further. We hypothesize that re-training with an adversarial

will further rectify the subtle color differences that *mae* couldn't solve.

We use a pre-trained ResNet-50 with imagenet weights as our discriminator with last few layers clipped off. The discriminative network used is something called a "Patch Discriminator" proposed by Isola et al. (2018). In a *vanilla* discriminator proposed by Goodfellow et al. (2014), the network produces a scalar output, representing the probability that the data x belongs to the input distribution and not the generator distribution p_g. Isola et al. (2018) proposed a modification to the discriminative network so as to produce, instead of a single scalar, a vector of probabilities representing different localities of the input distribution (image) x. For instance, a discriminator producing a 30×30 vector represents the probabilities every receptive field that is covered by the output vector. Thus, we can localize the corrections in the image that the generator should make.

Finally, we use the trained generator and trained discriminator and fine-tune it to fit it on our data which is rather small in size compared to the previous datasets. We train these networks in an adversarial fashion using the conditional GAN objective function (Isola et al.; 2018) and couple it with some noise introduced as the L1 norm of generated image tensor to the target image tensor. The reason behind this is to train the generator using an adversarial component while trying to minimize the Manhattan distance between the generator output and target vector space.

CHAPTER 7

SOFTWARE TESTING

7.1 TYPES OF TESTING

We test the modules of our entire setup to ensure that the modules are performing as they're intended to. We perform a kind of unit testing to evaluate the output of every function in the code.

7.2 TEST CASES AND TEST RESULTS

7.2.1 Data Gathering and Pre-processing

- Test Case 1: Image Scraping test
  - Test Case Description:
    The module function should scan through the legacy archive only in the described co-ordinates and scrape images that are RGB and of dimensions 256×256
  - Status: **Pass**
- Image Colorization:
  - RGB Color Space:
    - Test Case 2: Pre-processing System: Grayscale Images (Input distribution)
      - Test Case Description:
        The pre-processing function takes in a stream of raw data and converts the images into a numpy array with dimensions $n \times 64 \times 64 \times 1$ where n is the batch size. The output image vector should contain black & white images
      - Status: **Pass**
    - Test Case 3: Pre-processing System: Color Images (Target distribution)
      - Test Case Description:
        The target distribution should contain color counter-parts of the

respective input data. The function should output a numpy array with dimensions $n \times 64 \times 64 \times 3$ where n is the batch size.

· Status: **Pass**

– L*a*b Color Space:

* Test Case 2: Pre-processing System: RGB to L*a*b conversion

· Test Case Description:

The pre-processing function takes in a stream of raw data and converts the images into a numpy array with dimensions $n \times 64 \times 64 \times 3$ where n is the batch size. The output is an image tensor in L*a*b color space. The L channel is provided for the grayscale input

· Status: **Pass**

* Test Case 3: Pre-processing System: L*a*b to RGB conversion

· Test Case Description:

The function takes in its input as an image tensor in L*a*b colorspace. The function should output a numpy array with dimensions $n \times 64 \times 64 \times 3$ where n is the batch size and the image output is converted back to the RGB space

· Status: **Pass**

• Image Super-resolution:

– Test Case 4: Pre-processing System: Low Resolution Images (Input distribution)

* Test Case Description:

The pre-processing function takes in the raw data and produces low resolution $64 \times 64 \times 3$ images. The output should be a numpy vector of n dimensions representing the mini-batch size

* Status: **Pass**

– Test Case 5: Pre-processing System: High Resolution Images (Target distribution)

* Test Case Description:

  The target distribution contains the labelled data that is a high resolution image of dimensions $256 \times 256 \times 3$. The function produces a numpy vector of size n where n is the size of mini-batch

* Status: **Pass**

7.2.2 Model Implementation

- Image Colorization Model
  - Test Case 1: Generator network
    * Test Case Description:

      The generator network is a tensor graph implemented in tensorflow. The computation graph should have an input layer of dimensions $n \times 64 \times 64 \times 1$ which is propagated through till the final layer. The final layer output should be a tensor of dimensions $n \times 64 \times 64 \times 3$ where n is the mini-batch size

    * Status: **Pass**
  - Test Case 2: Discriminator network
    * Test Case Description:

      The discriminator network is a tensor graph implemented in tensorflow. The computation graph should have an input layer of dimensions $n \times 64 \times 64 \times 3$ which is propagated through till the final layer. The final layer output should be a tensor of dimensions $n \times 30 \times 30 \times 1$ where n is the mini-batch size

    * Status: **Pass**
- Image Super-resolution Model
  - Test Case 1: Generator network
    * Test Case Description:

      The generator network is a tensor graph. The computation graph should have an input layer of dimensions $n \times 64 \times 64 \times 3$. The final

layer output should be a tensor of dimensions $n \times 256 \times 256 \times 3$ where n is the mini-batch size

* Status: **Pass**

– Test Case 2: Discriminator network

* Test Case Description:
The discriminator network is a tensor graph. The computation graph should have an input layer of dimensions $n \times 256 \times 256 \times 3$. The final layer output should be a tensor of dimensions $n \times 1$ where n is the mini-batch size

* Status: **Pass**

7.2.3 Training Loop

- Test Case 1: Forward propagation

  – Test Case Description:
  The forward propagation function should run the computation graph over all the training samples and calculate losses of the generator and discriminator. The loss should be calculated according to the GAN objective function and final output should be the cost of the entire training epoch.

  – Status: **Pass**

- Test Case 2: Backward propagation

  – Test Case Description:
  The backward propagation should calculate gradients of the objective function based on the cost calculated during the forward propagation. The gradients should be used to update the trainable variables of the network using gradient descent with momentum and weighted average sum, i.e. Adam.

  – Status: **Pass**

- Test Case 3: Callbacks and intermediate results logging

- Test Case Description:

  While training the model, the callback functionality should keep track of the intermediate training results. It should perform the following:

  • A csv file containing history of the entire training session

  • Save the model weights on a best performance basis by keeping track of the earlier losses

  • Keep track of the convergence and stop the model training if the model converges early

- Status: **Pass**

7.2.4 Evaluation system

- Model performance evaluation
  - Test Case 1: Model history plotting
    - Test Case Description:

      The function should plot the model losses throughout the training loop and present a graph of the said plot
    - Status: **Pass**
  - Test Case 2: Output plotting
    - Test Case Description:

      The function should take in the predictions by the model and save the images with their respective ground truth images for evaluation
    - Status: **Pass**
  - Test Case 3: Model evaluation (L1 norm)
    - Test Case Description:

      The function should take in two numpy vectors consisting of predicted values and ground truths respectively. The function output should be a single number consisting of the L1 norm (Manhattan distance) between the target and predictions
    - Status: **Pass**

CHAPTER 8

RESULTS

8.1 DATASET AND EXPERIMENTAL SETUP

Initially, we started by scraping the data off Hubble Legacy Archive. Using puppeteer(headless chrome), we scraped off hundreds of thousands of colorized images it has available. The Hubble Legacy archive is slow and produces grainy images with lots of noise and unprocessed images. A filter for M101 (Messier 101) galaxy rendered more than 80 thousand images with a 1 degree difference between consecutive right ascension. The data is large and has no particularly efficient way to clean without human investment. Cleaning tens of thousands of images by handpicking noiseless and well colored images is time consuming. For training the SRGAN, we need high resolution, well colored images.

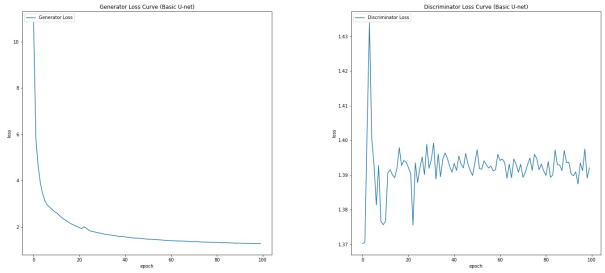
Consequently, we scraped the Hubble Heritage project instead. The Hubble Heritage project releases the highest-quality astronomical images. They are all stitched together, colorized and processed to eliminate noise. Hubble Heritage then selects the best, most striking of these for public release. However, there are only ~ 150 of these images that are actually useful. We scraped images from the main Hubble website as well so as to increase the amount of data we had. This provided an extra approximately ~ 1000 images. Each image is a JPG image with dimensions of 256×256 pixels and contains 3 channels of RGB.

We use a mini-batch gradient descent with a batch size of 10, 16 and 32 for different iterations. We use Adam optimizer with $\beta_1 = 0.5$ and $\beta_2 = 0.999$. The generator and discriminator has a learning rate of $2e-4$ which remains constant throughout the training. We train the model with different epochs ranging between 20,50 and 100, saving the best model weights determined by L1 norm between the output of the generator and the target image. We use early stopping with a patience value of 10. The image size is 64×64. Our implementation is using Python, numpy, Tensorflow and tf-keras. It takes about 24 hours to complete training over the COCO dataset and about 12 to 13 hours to fit the model on given astronomical data on a NVIDIA Tesla K80 GPU.

8.2 RESULTS AND DISCUSSIONS

In the following section, we briefly compare and summarize the results of the implemented architectures and evaluate their performance. The evaluation is done qualitatively as well as quantitatively. We compare the performance of each model using L1 and L2 norm distance between the predictions and targets. Though unreliable, this method provides us with a somewhat decent ground to perform a comparative study and evaluate the reliability of such metrics on GAN evaluation compared to qualitative, visual evaluation.

To evaluate the model performance by virtue of convergence of the objective function, we plot the generator loss throughout the training on two different networks.

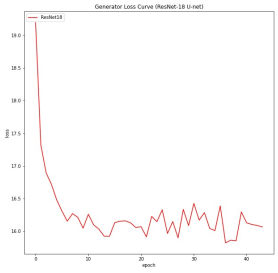


(a) U-net Generator loss (b) Discriminator loss

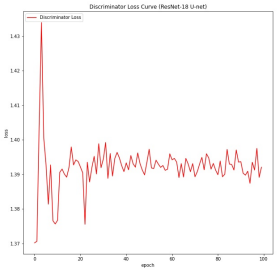
Figure 8.1: Curve plots for Basic U-net & Custom discriminator

Figure 8.1a and 8.1b shows that the generator converges quite nicely. The discriminator on the other hand oscillates because of the convergence that the generator shows. GAN losses are pretty non-intuitive but we can draw some observations from the loss curves. It seems that the pattern of oscillation and convergence repeats when networks are trained in an adversarial fashion.

Figure 8.2a and 8.2b show the same trend repeating even when trained using pre-trained weights with state of the art networks that show excellent results. It is observed that the ResNet U-net with a pre-trained ResNet-18 for its backbone converges decently in the beginning but later shows spikes when nearing the end of training loop. The ResNet50 used as a discriminator shows identical behavior as the custom discriminator above refer fig.8.1b.



(a) ResNet18 U-net Generator



(b) ResNet50 discriminator loss

Figure 8.2: Curve plots for ResNet18 U-net & ResNet50 discriminator

We draw some conclusions based on our understanding of the plots. It seems that the loss convergence doesn't signify whether the model is predicting expected results. The loss function just converges to the minimum value that the cost function descends to, permitted by the learning rate. It would normally signify that the GAN has found some optimum point in the vector space that is at the lowest potential and cant decrease any further. It essentially means that the GAN has leaned enough. Due to the large number of dimensions, owing to the high amount of trainable variables, such combinations, where the function converges, can be huge in volume. Thus, these numbers don't provide any better understanding of the bias or variance the model is facing. Also, we discover that if the loss hasn't converged well, it doesn't necessarily mean that the model hasn't learned anything. On visual inspection, the generated results show similar results to the ground truth, even with high generator losses. This might be due to presence of a content loss parameter in the loss function where we try to minimize the function by minimizing the L1 norm between the generator predictions and target.

The discriminator shows an increase in the objective loss function in the initial epochs and then settles down in the later phase around an oscillation point. This shows that the discriminator is converging to some permanent number or rather, oscillating around it. We assume this point to be a point of stability between the two networks as the networks are in a constant adversarial battle, meaning if one performs better, the other is bound to perform worse.

8.2.1 Image Colorization

We present a comparative study of the following models: Basic U-net generator with a custom, residual VGG16 discriminator, hereafter referred to as Basic U-net. A modified U-net with pre-trained ResNet18 as its backbone. We evaluate this model by training it in RGB color space to predict 3 numbers for every pixel and in L*a*b color space where the model predicts the a and b channel alone. We then further train this model to fine-tune it to the task of colorizing astronomical images.

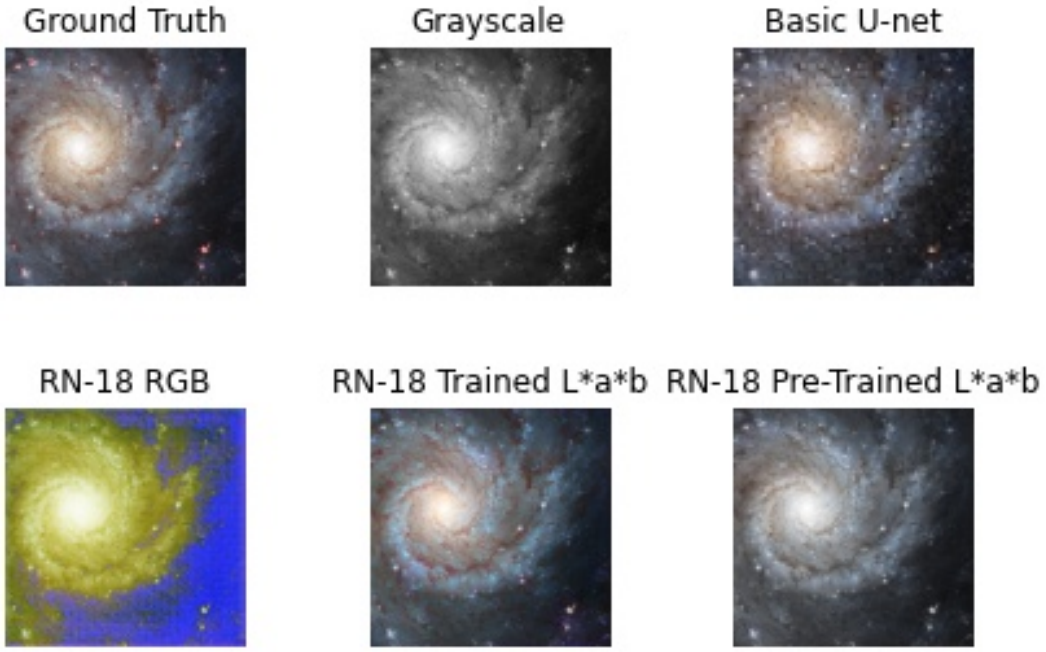


Figure 8.3: Results of the colorization study. The first two images in the top row are the ground truth and input image respectively. The next images belong to the output of following, serially: Basic U-net, ResNet18 in RGB colorspace, ResNet18 fine-tuned in L*a*b colorspace, ResNet18 pre-trained U-net in L*a*b colorspace

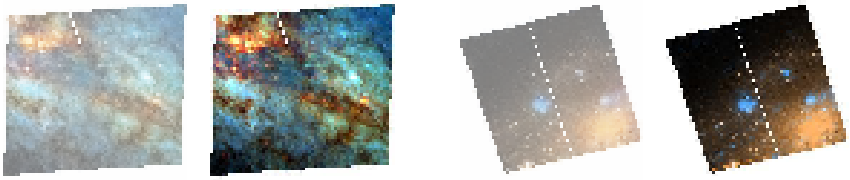
From figure 8.3 we observe that on this particular example, the Basic U-net performs better at predicting results that closely map to the ground truth but the fine-tuned ResNet-18 shows promising results over a larger set of inputs. It can also be observed that the Basic U-net architecture does a decent job of faking the sharpness in the original image and that makes the image appear more realistic as compared to rather blurry outputs from the other networks.

We also observe that the model trained in the RGB color space performs poorly at predicting values of the three color channels and that results in dominance of one channel (blue in this case) over others. This causes the output, even though

reconstructed quite accurately, to have a varied color pattern with high emphasis on one color channel. This might be because of presence of deep layers which cause gradients of certain colors to diminish over time, causing the model to be strongly biased towards the blue color in this case. An increase in the volume of training data and some random pixel shuffling in forward propagation might solve this problem.

The pre-trained ResNet18 U-net performs decently with the weights gathered by training it over the COCO dataset. The model still lacks the specific coloring intuition in astronomical images and plainly colors specific parts of the images in light colors, leaving majority of the image unaltered. This causes the images to have a slight gray tinge.

Figure 8.6 shows how the other models perform in different color spaces. We can observe that the Basic U-net model performs good at predicting the outputs but doesn't predict the brightness level of pixels quite accurately. The model seems to be overfitting on the dataset and suffers a high variance on output because there are a lot of testing samples which demonstrate the poor performance of the Basic U-net model as can be seen in figure 8.4.



(a) Sample 2 (b) Sample 2

Figure 8.4: Performance of Basic U-net over samples different from the dataset. The left part is the prediction, right part is the ground truth

(a) Ground Truth

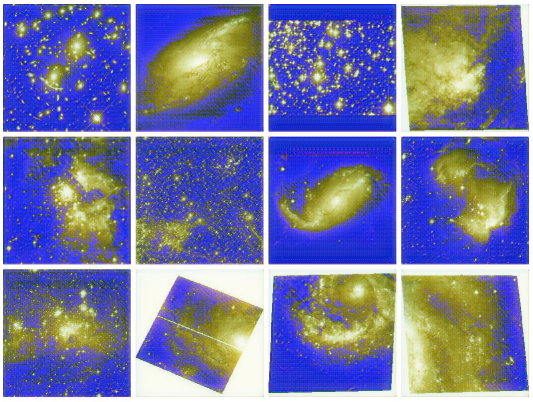
(b) ResNet18 Full trained L*a*b

Figure 8.5: Results of fine-tuned ResNet18 in L*a*b colorspace compared with ground truth images

Figure 8.5a shows the ground truth images and figure 8.5b shows the predictions of the ResNet18 full trained model in the L*a*b color space. This model seems to perform best, visually.

(a) Basic U-net

(b) ResNet18 pre-trained L*a*b



(c) ResNet18 pre-trained RGB

Figure 8.6: Results of colorization models (from top to bottom): Basic U-net, ResNet18 U-net pre-trained in L*a*b color space, ResNet18 U-net pre-trained in RGB color space

To quantitatively estimate the model performance, we measure the L1 and L2 distance between the predictions and the ground truth images in both RGB as well as L*a*b colorspace.

Model	Color Space	L1 Distance	L2 Distance
ResNet-18 (Pre-trained)	L*a*b	64.5409	2.77
ResNet-18 (Fine-tuned)	L*a*b	65.1119	2.62
ResNet-18 (Pre-trained)	RGB	125.5554	9.04
U-net	RGB	77.3273	3.986

Table 8.1: Colorization: Average per-pixel L1 & L2 distance between generated images and ground truth

Table 8.3 shows that the RGB color space performs poorly on the task of colorization. In terms of the L1 distance, the best performance is achieved on the ResNet18 U-net with pre-trained weights. This goes to prove the unreliability of distance metrics in model performance evaluation. The model, although quantitatively, performs better than the fine-tuned model but in reality, fails to produce images that might be visually appealing. The L2 distance metric shows how the fine-tuned model might, in reality, be fitting slightly better over the data to predict correct color combination as its output. We still consider the results from the fine tuned model to be better than the other performing models and further investigate the per-channel predictions by the model.

Distance	Red	Green	Blue
L1 Norm	64.6078	38.5994	92.1283
L2 Norm	3.4531	1.0253	3.8046

Table 8.2: Channel wise averages of ResNet-18 finetuned network

Table 8.2 shows the RGB channel averages of the outputs produced by the fine tuned model. It can be observed that the feature embeddings produced by the model in *a, *b color spaces maps to the green channel with the least error. This might be indicative that the model is performing poorly on images that have a high content of red-blue colors.

To further evaluate the actual outputs produced in the L*a*b color space, we compare the *a, *b channels of the ResNet18 architectures.

Model	L1 Distance		L2 Distance	
	a	b	a	b
ResNet-18 (Pre-trained)	0.00588	0.00501	0.01565	0.03208
ResNet-18 (Fine-tuned)	0.00257	0.00727	0.01721	0.03316

Table 8.3: L1 & L2 channel wise distances in L*a*b colorspace

In table 8.3, L1 distance, we can see the performance of fine-tuned model to be better in channel *a but poor in *b compared to the pre-trained model. The L2 distance metric rules out the possibility of the fine-tuned model performing better than the pre-trained model. In reality, the fine-tuned model is orders of magnitude better at predicting output abstractions with the L channel as its input, thus contradicting the quantitative results.

Model	L1 Distance			L2 Distance		
	R	G	B	R	G	B
ResNet-18 (Pre-trained)	69.2401	36.5852	87.7973	3.2890	1.0439	3.9785
ResNet-18 (Fine-tuned)	64.6078	38.5994	92.1283	3.4531	1.0253	3.8046

Table 8.4: L1 & L2 channel wise distances in RGB colorspace

We can observe the comparative results of ResNet18 pre-trained model and the fine-tuned model. It can be concluded from the observations that the fine-tuned model performs better at predicting the Red color channel but suffers in Green and Blue.

8.2.2 Image Super Resolution

We implement the basic SR-GAN proposed by Ledig and train it to improve super-resolution task. We compare the trained model with pre-trained SRGAN model, EDSR-GAN proposed by Lim et al. (2017) and WDSR-GAN proposed by Yu et al. (2018).

Figure 8.7 shows how the networks perform on the task of predicting pixels while upscaling of the image. All the networks perform really well and its difficult to distinguish the outputs visually. It may be observed that the fine-tuned network produces images that look slightly better than the other counter-parts. The best per-

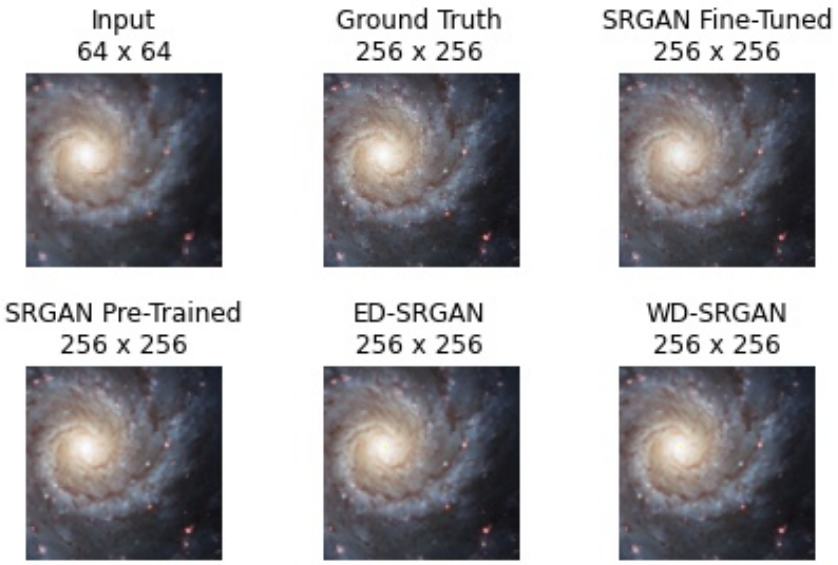


Figure 8.7: Results of the super-resolving models. The first two images in the top row are the input image and ground truth respectively. The next images belong to the output of following, serially: Fine-tuned SRGAN, pre-trained SRGAN, EDSR, WDSR

forming network seems to be the WDSR-GAN with a very little error in predicting the output.

It is evident that the model produces acceptable results on visual inspection. The main reason behind this might, again, be the random pixel shuffling between every upscaling pass.

As opposed to colorization, super-resolution needs a quantitative estimation to determine which model performs best among the give models.

Model	**L1 Distance**	**L2 Distance**
Ledig SRGAN (Fine-tuned)	87.1090	3.754
Ledig SRGAN (Pre-trained)	114.8043	5.953
ED-SRGAN	80.9414	3.684
WD-SRGAN	79.7262	3.627

Table 8.5: Super-Resolution: Per-pixel Average L1 & L2 distance between generated images and ground truth

(a) Input Distribution

(b) SRGAN fine-tuned output

Figure 8.8: Results of SRGAN. 8.8a shows the input data and 8.8b shows the corresponding output

Table 8.5 shows the L1 and L2 distances of predicted results by each model with the ground truth image. It is observed that Ledig's SRGAN, after a bit of fine-tuning performs really well in comparison to the the pre-trained version. To further improve this, Lim et al. (2017) proposed an optimized version of SRGAN by removing the unnecessary modules in the conventional resnets and showed that Enhanced Deep Residual Networks performed better at upscaling task. When trained in an adversarial manner, the ED-SRGAN performs better than the traditional SRGAN. Yu et al. (2018) further improved the idea by increasing the widening factor ($\times 2$ and $\times 4$) to ($\times 6$ and $\times 9$). This Wide Activation Deep Super Resolution network further improved the performance for single image super-resolution. When we implement this in an adversarial manner, we achieve excellent results. It is evident from the results that the best performing network is the WDSR.

CHAPTER 9

SUMMARY AND CONCLUSION

9.1 SUMMARY AND CONCLUSION

The project mainly tackles the problem of colorizing astronomical images and super-resolving them for astronomical inspection. We explore various methodologies proposed till date that efficient colorize and super-scale images with results that are significantly closer to the ground truth distribution. We scrape the data from Hubble Legacy Archive, Hubble Main website and Hubble Heritage project and created a filtered and clean dataset of 4700 images. We split the data and use 500 images, roughly 10% of the data for testing purposes. To compensate for the lack of data, we implement several pre-trained architectures and fine-tune their abstractions over our dataset to find the most effective solution.

For the colorizing model, we explore usage of U-net architectures starting from a basic U-net model and experiment with different color spaces and empirically confirm the superiority of L*a*b color space in image colorization problem. We use the state of the art ResNet-18 to provide as a backbone of the encoder and build a U-net around it. The pre-trained network over COCO dataset in RGB colorspace produces significantly weaker results as compared to the subsequent network in L*a*b colorspace. The best performing model turns out to be the ResNet18 U-net which is fine-tuned over our particular dataset to produce appealing and similar results to the ground truth.

The Super-resolution model is based largely on the SRGAN proposed by Ledig et al. (2017). We use the generator weights and sample results from the training set to inspect the results. It is found that the model performs really well on the pre-trained weights and we decide to fine-tune it to our application. After fine-tuning the model, we train other state of the art single image super-resolution models such as EDSR (Lim et al.; 2017) and WDSR (Yu et al.; 2018). These provide further insights into the problem and simultaneously, improve the results.

While studying and improving the performance of these models, we explore performance metrics of GANs and evaluation methodologies implemented to test out conditional GANs. It is evident that the loss curves of generator and discriminator do not provide us with any intuition about the model performance. We also discover that standard distance metrics cannot be used to evaluate GANs and quantitative

methods that exist to evaluate GANs are unreliable. We prove so by contradiction of qualitative samples and quantitative measurements of the best performing architecture for colorization models. However, we observe that quantitative estimation is quite reliable for the problem of single image super-resolution and can be helped to determine which model is better suited for the task.

9.2 FUTURE SCOPE

Though we obtain moderately good results, a vast amount of algorithms still remain unscratched. A more powerful model such as SE-ResNext, EfficientNet and more state-of-the-art models can be implemented and trained over millions of images from the Imagenet. With even more hardware resources and availability of data, we can explore computationally heavy models for a better approximation. An image stitching algorithm can be applied on the produced images to generate large scale astronomical images for scientific study. Colorization can be improved by the virtue of exploring different loss functions using weighted losses to reduce loss problem for low saturation regions. We can introduce a gradient penalty for the SRGAN architecture and include the WGAN (Arjovsky et al.; 2017) which will stabilize the discriminator to stay within the space of 1-Lipschitz function. Progressively growing GANs (Karras et al.; 2018) can be applied so that the dimensions can be further improved with more stability and greater sharpness.

9.3 APPLICATIONS

The images are upscaled and colourized using a completely automated algorithm which uses Deep Learning. Generative Adversarial Networks are successfully used for the implementation. The algorithm can directly be applied for creating images that can be studied by astronomers. It is anticipated that this will aid the astronomers vastly in their efforts.

REFERENCES

Arjovsky, M., Chintala, S. and Bottou, L. (2017). Wasserstein gan.

Cheng, Z., Yang, Q. and Sheng, B. (2016). Deep colorization.

Dahl, R. (2016). Automatic colorization.

Dong, C., Loy, C. C., He, K. and Tang, X. (2014). Learning a deep convolutional network for image super-resolution, *in* D. Fleet, T. Pajdla, B. Schiele and T. Tuytelaars (eds), *Computer Vision – ECCV 2014*, Springer International Publishing, Cham, pp. 184–199.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. and Bengio, Y. (2014). Generative adversarial networks.

He, K., Zhang, X., Ren, S. and Sun, J. (2015). Deep residual learning for image recognition.

Huang, Y.-C., Tung, Y.-S., Chen, J.-C., Wang, S.-W. and Wu, J.-L. (2005). An adaptive edge detection based colorization algorithm and its applications, pp. 351–354.

Isola, P., Zhu, J.-Y., Zhou, T. and Efros, A. A. (2018). Image-to-image translation with conditional adversarial networks.

Jianchao Yang, Wright, J., Huang, T. and Yi Ma (2008). Image super-resolution as sparse representation of raw image patches, *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–8.

Karras, T., Aila, T., Laine, S. and Lehtinen, J. (2018). Progressive growing of gans for improved quality, stability, and variation.

Kim, J., Lee, J. K. and Lee, K. M. (2016). Accurate image super-resolution using very deep convolutional networks, *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1646–1654.

Ledig, C., Theis, L., Huszar, F., Caballero, J., Cunningham, A., Acosta, A., Aitken, A., Tejani, A., Totz, J., Wang, Z. and Shi, W. (2017). Photo-realistic single image super-resolution using a generative adversarial network.

Levin, A., Lischinski, D. and Weiss, Y. (n.d.). Colorization using optimization, *ACM SIGGRAPH 2004 Papers*, ACM Journals.

Lim, B., Son, S., Kim, H., Nah, S. and Lee, K. M. (2017). Enhanced deep residual networks for single image super-resolution.

Long, J., Shelhamer, E. and Darrell, T. (2015). Fully convolutional networks for semantic segmentation, *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3431–3440.

Mirza, M. and Osindero, S. (2014). Conditional generative adversarial nets.

Pressman, R. S. (1992). *Software Engineering (3rd Ed.): A Practitioner's Approach*, McGraw-Hill, Inc., New York, NY, USA.

Qu, Y., Wong, T.-T. and Heng, P.-A. (2006). Manga colorization, *ACM Transactions on Graphics (TOG)* **25**(3): 1214–1220.

Radford, A., Metz, L. and Chintala, S. (2016). Unsupervised representation learning with deep convolutional generative adversarial networks.

Shi, W., Caballero, J., Huszár, F., Totz, J., Aitken, A. P., Bishop, R., Rueckert, D. and Wang, Z. (2016). Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network, *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1874–1883.

Simonyan, K. and Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition.

Tola, E., Lepetit, V. and Fua, P. (2008). A fast local descriptor for dense matching, *Proc. CVPR* .

Tom and Katsaggelos (1996). Reconstruction of a high-resolution image by simultaneous registration, restoration, and interpolation of low-resolution images, *in* Anon (ed.), *IEEE International Conference on Image Processing*, Vol. 2, IEEE, pp. 539–542. Proceedings of the 1995 IEEE International Conference on Image Processing. Part 3 (of 3) ; Conference date: 23-10-1995 Through 26-10-1995.

TSAI, R. (1984). Multiframe image restoration and registration, *Advance Computer Visual and Image Processing* **1**: 317–339.
URL: *https://ci.nii.ac.jp/naid/10026807118/en/*

Welsh, T., Ashikhmin, M. and Mueller, K. (2002). Transferring color to greyscale images, *ACM Trans. Graph.* **21**: 277–280.

Yatziv, L. and Sapiro, G. (2006). Fast image and video colorization using chrominance blending, *IEEE Transactions on Image Processing* **15**(5): 1120–1129.

Yu, J., Fan, Y., Yang, J., Xu, N., Wang, Z., Wang, X. and Huang, T. (2018). Wide activation for efficient and accurate image super-resolution.

Zhu, J.-Y., Krähenbühl, P., Shechtman, E. and Efros, A. A. (2018). Generative visual manipulation on the natural image manifold.

ANNEXURE A

PLAGIARISM REPORT

SmallSEOTools

PLAGIARISM SCAN REPORT

Words	483	Date	April 22,2021
Characters	3326	Excluded URL	

0%	100%	0	26
Plagiarism	Unique	Plagiarized Sentences	Unique Sentences

Content Checked For Plagiarism

Automated colorization of greyscale images has been subjected to much research within the computer vision and machine learning communities. Beyond simply being fascinating from an aesthetic and artificial intelligence perspective, such capability has broad practical applications. It is an area of research that possesses great potentials in applications: from black and white photo reconstruction, image augmentation, video restoration to image enhancement for improved interpretability. Image downscaling is an innately lossy process. The principal objective of super resolution imaging is to reconstruct a low resolution image into a high resolution one based on a set of low-resolution images to rectify the limitations that existed while the procurement of the original low-resolution images. This is to insure better visualization and recognition for either scientific or non-scientific purposes. Even if an upscaling algorithm is particularly good, there will always be some amount of high frequency data lost from a downscale-upscale function performed on the image. Ultimately, even the best upscaling algorithms are unable to effectively reconstruct data that does not exist. Traditional methods for image upsampling rely on low-information, smooth interpolation between known pixels. Such methods can be treated as a convolution with a kernel encoding no information about the original image. A solution to the problem is by using Generative Adversarial Networks (GANs) to hallucinate high frequency data in a super-scaled image that does not exist in the smaller image. Even though they increase the resolution of an image, they fail to produce the clarity desired in the super-resolution task. By using the above mentioned method, not a perfect reconstruction can be obtained albeit instead a rather plausible guess can be made at what the lost data might be, constrained to reality by a loss function penalizing deviations from the ground truth image. A huge number of raw images are present unprocessed and unnoticed in the Hubble Legacy Archives. These raw images are typically black and white, low-resolution and unfit to be shared with the world. It takes huge amounts of hours to process them. This processing is necessary because it's difficult for astronomers to distinguish objects from the raw images. Random and synthetic noise from the sensors in the telescope, changing optical characteristics in the system and noise from other bodies in the universe all make the processing further necessary. Furthermore, for the process of highlighting small features that ordinarily wouldn't be able to be picked out against noise of the image, we need colorization. The processing of the images is so time consuming that the images are rarely seen by human eyes. The problem is only likely to get worse. Not only is new data being continuously produced by Hubble Telescope, but new telescopes are soon to come online. A simplification of image processing by using artificial image colorization and superresolution can be done in an automated fashion to make it easier for astronomers to visually identify and analyze objects in Hubble dataset.

Sources	Similarity

Figure A.1: Plagiarism report for Abstract

SmallSEOTools

PLAGIARISM SCAN REPORT

Words	926	Date	April 22,2021
Characters	6689	Excluded URL	

0%	100%	0	58
Plagiarism	Unique	Plagiarized Sentences	Unique Sentences

Content Checked For Plagiarism

1.3 LITERATURE SURVEY
1.3.1 Image Colorization
1.3.1.1 Hint Based Colorization
[2] proposed using colorization hints from the user in a quadratic cost function which imposed that neighboring pixels in space-time with similar intensities should have similar colours. This was a simple but effective method but only had hints which were provided in form of imprecise colored scribbles on the grayscale input image. But with no additional information about the image, the method was able to efficiently generate high quality colorizations. [3] addressed the color bleeding issue faced in this approach and solved it using adaptive edge detection. [4] used luminescence based weighting for hints to boost efficiency. [5] extended the original cost function to apply color continuity over similar textures along with intensities. [6] had proposed another approach that reduced the burden on the user by only requiring a full color example of an image with similar composition. It matched the texture and luminescence between the example and the target grayscale image and received realistic results as long as the example image was sufficiently similar. Regardless of the scribble based or example based approach, the algorithms still needed sufficient human assistance in form of hand drawn or colorized images.
1.3.1.2 Deep Colorization
Owing to recent advances, the Convolutional Neural Networks are a de facto standard for solving image classification problems and their popularity continues to rise with continual improvements. CNNs are peculiar in their ability to learn and differentiate colors, patterns and shapes within an image and their ability to associate them with different classes. [7] proposed a per pixel training for neural networks using DAISY [8], and semantic [9] features to predict the chrominance value for each pixel, that used bilateral filtering to smooth out accidental image artifacts. With a large enough dataset, this method proved to be superior to the example based techniques even with a simple Euclidean loss function against the ground truth values.
Finally, [10] successfully implemented a system to automatically colorize black & white images using several ImageNet-trained layers from VGG-16 [11] and integrating them with auto-encoders that contained residual connections. These residual connections merged the outputs produced by the encoding VGG16 layers and the decoding portion of the network in the later stages. [12] showed that deeper neural networks can be trained by reformulating layers to learn residual function with reference to layer inputs. Using this Residual Connections, [12] created the ResNets that went as deep as 152 layers and won the 2015 ImageNet Challenge.
1.3.1.3 Generative Adversarial Networks
[13] introduced the adversarial framework that provides an approach to training a neural network which uses the generative distribution of pg(x) over the input data

Figure A.2: Plagiarism report for Literature Review

x.
Since it's inception in 2015, many extended works of GAN have been proposed over years including DCGAN [14], Conditional-GAN [15], iGAN [16], Pix2Pix [17].
[14] applied the adversarial framework for training convolutional neural networks as generative models for images, demonstrating the viability of deep convolutional generative adversarial networks.
DCGAN is the standard architecture to generate images from random noise. Instead of generating images from random noise, Conditional-GAN [15] uses a condition to generate output image. For e.g. a grayscale image is the condition for colorization of image. Pix2Pix [17] is a Conditional-GAN with images as the conditions.
The network can learn a mapping from input image to output image and also learn a separate loss function to train this mapping. Pix2Pix is considered to be the state of the art architecture for image-image translation problems like colorization.
1.3.2 Image Upscaling
1.3.2.1 Frequency-domain-based SR image approach
[18] proposed the frequency domain SR method, where SR computation was considered for the noise free low resolution images. They transformed the low resolution images into Discrete Fourier transform (DFT) and further combined it as per the relationship between the aliased DFT coefficient of the observed low resolution image and that of unknown high resolution image. Then the output is transformed back into the spatial domain where a higher resolution is now achieved.
While Frequency-domain-based SR extrapolates high frequeny information from the low resolution images and is thus useful, however they fall short in real world applications.
1.3.2.2 The interpolation based SR image approach
The interpolation-based SR approach constructs a high resolution image by casting all the low resolution images to the reference image and then combining all the information available from every image available. The method consists of the following three stages (i) the registration stage for aligning the low-resolution input images, (ii) the interpolation stage for producing a higher-resolution image, and (iii) the deblurring stage which enhances the reconstructed high-resolution image produced in the step ii).
However, as each low resolution image adds a few new details before finally deblurring them, this method cannot be used if only a single reference image is available.
1.3.2.3 Regularization-based SR image approach
Most known Bayesian-based SR approaches are maximum likelihood (ML) estimation approach and maximum a posterior (MAP) estimation approach.
While [19] proposed the first ML estimation based SR approach with the aim to find the ML estimation of high resolution image, some proposed a MAP estimation approach. MAP SR tries to takes into consideration the prior image model to reflect the expectation of the unknown high resolution image.
1.3.2.4 Super Resolution - Generative Adversarial Networks (SR-GAN)
The Genrative Adversarial Network [13], has two neural networks, the Generator and the Discriminator. These networks compete with each other in a zero-sum game. [20] introduced SRGAN in 2017, which used a SRResNet to upscale images with an upscaling factor of 4x. SRGAN is currently the state of the art on public benchmark datasets

Sources	Similarity

Figure A.2: Plagiarism report for Literature Review (contd.)

SmallSEOTools

PLAGIARISM SCAN REPORT

Words	360	Date	April 22,2021
Characters	2320	Excluded URL	

0%	100%	0	21
Plagiarism	Unique	Plagiarized Sentences	Unique Sentences

Content Checked For Plagiarism

2.1 PROBLEM STATEMENT
The problem can be divided into two sub-problems:
• Create an efficient model to colorize grayscale images
• Take a colorized image and upscale it n times the original size
2.1.1 Goals and objectives
Goal and Objectives:
• Auto-Colorization:
– The first model will be given input a grayscale, low resolution image of dimensions (64641)
– The model will perform a series of mathematical operations that will increase the channel width of the image from 1 (single channel grayscale image) to 3 (RGB)
– The output of the model will be a colorized version of the input image with dimensions (64643)
• Upscaling/super-resolution:
– The input to the model will be a colorized image of shape (64643)
– The model will increase the dimensions of the image from (6464) to ((64 n)(64 n)) by performing a series of upscaling operations and predicting information that may be lost while downscaling
– The output of the model will be an upscaled RGB image with dimensions ((64 n)(64 n)3)
• The models may be combined to form a single model that will take a low resolution, grayscale image as its input and produce a high resolution, colorized image as its output
2.1.2 Statement of scope
• The model will consist of neural networks implemented using deep learning frameworks that will accept images of input format JPEG
• The input will be grayscale images of size 6464
• Input bounds:
– Lower bound: 64641
– Upper bound: no limit
• The output will be produced in two phases:
– A colorized output of model 1 with shape 64643
– A upscaled output of model 2 from the colorized output of model 1 with shape (64 n)(64 n)3
• The model will:
– take input black & white images
– produce colorized images of the same size
– produce upscaled images of size n times the input size (currently 64)

Figure A.3: Plagiarism report for Problem Definition and Scope (Part 1)

• The model will not:
– take a colorized image as an input
– take an image of size less than (6464) in size
– produce accurate upscaling or coloring albeit merely make a guess at what the lost values might be

Sources	Similarity

Figure A.3: Plagiarism report for Problem Definition and Scope (Part 1) (contd.)

SmallSEOTools

PLAGIARISM SCAN REPORT

Words	915	Date	April 22,2021
Characters	5961	Excluded URL	

2%	98%	1	55
Plagiarism	Unique	Plagiarized Sentences	Unique Sentences

Content Checked For Plagiarism

2.2 MAJOR CONSTRAINTS
• The astronomical image data required for training purposes is mostly raw. There exists no structured dataset that is already cleaned. The unavailability of a dataset is a major constraint for the project
• Scraped data from the archives is noisy and requires heavy processing and cleaning in order to be usable by the model
• The images available for download are of low resolution, which sets an upper bound on the maximal upscale factor
• The image data is large and needs high computation power to process
• The data needs to be cleaned manually as there exist no methods to automatically do this particular task
• The model involves neural networks which heavily rely on computation power for its training. The hardware required for training is not readily available because of absence of a workstation supporting heavy computations
• The training part requires large amount of memory
• Absence of an NVIDIA workstation GPU will slow down the training further
2.3 METHODOLOGIES OF PROBLEM SOLVING AND EFFICIENCY ISSUES
• Data gathering and processing
– Data Scraping
* Owing to unavailability of a dataset, raw data can be acquired by the means of web scraping
* Images from the snapshots of entire night sky can be obtained in such a way from the Hubble Legacy Archive
– Data Cleaning
* The scraped data consists of snapshots of the entire night sky with 1 degree deviation of the telescope
* This results in large amount of noisy, overexposed, irregular data images
* This data needs to be cleaned manually before it can be used for any kind of study
• Image colorization
– The problem of image colorization has been solved using multiple methodologies
– [10] used convolutional neural networks with residual encoders using the VGG16 architecture
– Though the system performs extremely well in realistically colorizing various images, it consisted of L2 loss which was a function of the Euclidean distance between the pixel's blurred color channel value in the target and predicted image
L2loss =

Figure A.4: Plagiarism report for Problem Definition and Scope (Part 2)

nå
i=1
(ytrue□ypredicted)2 (2.1)
– This is a regression based approach and the pixel-wise L2 loss will impose
an averaging effect over all possible candidates and will result in
dimmer and patchy colorization
– Generative Adversarial Networks introduced by [13] use a minimax loss
which is different than the L2 loss as it will choose a color to fill an area
rather than averaging. This is similar to a classification based approach
• Image Upscaling
– One of the most popular approach to image upscaling was sparse-coding.
This approach assumes that images can be sparsely represented by a dictionary
of atoms in some transformed domain [21]. The dictionary is
learned during the training process.
– The main drawback for this was that the optimization algorithm was
computationally expensive
– Dong et. al explored super-resolution using convolutional neural network
and calling it SRCNN [22]. They explained how CNN had many similarities to the sparse-coding-based super-resolution.
– Kim et. al improved upon SRCNN's results using their very own model
inspired from the VGG-net architecture[23].
– After the introduction of GANs, Ledig et. al applied them to superresolution
(SRGAN) using a network inspired by the ResNets [20][12].
– SR-GAN works well with for single image super-resolution as it also
uses an intelligent content loss function that uses pre-trained VGG-net
layers. However, Ledig et. al noted that further information could be
used if this network were to be adapted to a video, such as temporal
information.
• A generative network, G, is meant to learn the underlying distribution of a data
set, Y. For e.g. we can train a GAN over face images to generate images similar
to those faces. With just a generative network however, we must visually
assess the quality of network outputs and judge how we can adapt the network
to produce more convincing results.
• With a discriminative network D, we can incorporate this tweaking directly
into training. The discriminative network takes in both fabricated inputs generated
by G and the real inputs from Y. It's sole purpose is to classify if the
input has come from G or Y.
• The key idea is back propagation of the gradients from the results of D's classification
to G so that G gets better at producing images and in turn fooling
D.
• For the project, we split the data into two categories: X that serves as the data
for the Y, which are its corresponding labels.
• G1 takes in a low resolution x 2 X which is black & white and produces ˆ y,
a colorized version of x. The descriminator D, in turn takes in a colorized
image and outputs the probability that the image comes from Y, instead of as
outputs from G, G(x). As such, if the discriminator is fooled by out generator, it should output a probability greater than 0.5 for the set of inputs coming from
G(x) and a probability less than 0.5 for images coming from Y.
• The same is the process for generator G2 with the only difference being that
the X is the set of colorized images but having low resolution and Y is the
set of high resolution images that serve as the labels for underlying mapping
of X. G2 takes in the low resolution image x 2 X and produces ˆ y and the
discriminator outputs a probability determining whether the image is superresolved
by G2 or the ground truth images from Y.

Sources	Similarity
Super-Resolution on Image and Video must then visually assess the quality of network outputs and judge how we can adapt the network to produce more con-vincing results. With a discriminative network, D, we can incorporate this tweaking directly into training. The discriminative net-work takes in both fabricated inputs generated by G and inputs coming from the "real" dataset, Y. Its purpose is to http://cs231n.stanford.edu/reports/2017/pdfs/312.pdf	3%

Figure A.4: Plagiarism report for Problem Definition and Scope (Part 2) (contd.)

SmallSEOTools

PLAGIARISM SCAN REPORT

Words	424	Date	April 22,2021
Characters	3023	Excluded URL	

0%	100%	0	29
Plagiarism	Unique	Plagiarized Sentences	Unique Sentences

Content Checked For Plagiarism

2.4 SCENARIO IN WHICHMULTI-CORE, EMBEDDED AND DISTRIBUTED COMPUTING USED
A deep learning algorithm is a software construct that has certain steps that may be hardware intensive. Generative Adversarial Networks require huge amount of computing prowess to complete multiple passes of forward and backward propagation in order to train themselves. A network may consist of millions and billions of parameters which are associated with hundreds of thousands of graph nodes. To actually be able to train a network with more than a billion parameters, we need appropriately large amount of memory. Furthermore, the operations of forward and backward propagation are mathematical operations that adjust the parameters based on the gradient of the cost function to minimize the cost. This calculation, although heavy, is independent of each node and can be performed in a parallel framework. NVIDIA CuDA enabled GPUs have a CuDNN (CuDA Deep Neural Network) library that hooks the training algorithm onto the GPU memory for processing, deploying thousands and hundreds of thousand parallel threads to perform independent calculations of optimizing gradients. Such an infrastructure is expensive and requires a dedicated set up for running deep learning algorithms. For normal use cases, one can run into the problems of memory overflows while allocating tensors in the process of creation of graphs. In such cases, it is costly to buy more GPUs. One can make use of cloud services provided by Google Colab, AWS, Azure and more. These services can host runtimes that will allow users to run their deep learning algorithms over their hardware which will ensure fast and efficient training.
2.5 OUTCOME
• An efficient mathematical model to be created which will describe mappings required to colorize and super-resolve low resolution grayscale images
• A brief albeit descriptive study of different approaches towards image colorization and super-resolution
• Study presenting the benefits of certain GAN architectures and their edge over other kinds of neural networks in image colorization and super-resolution
2.6 APPLICATIONS
• Currently, given the number of the raw and unprocessed images in Hubble Legacy Archives, much of the images are not workable for scientific evaluation. The main application of building a GAN and automating the upscaling and colorization of these images is to help in visual classification for astronomers. Through a high resolution and colourized image, astronomical objects which would’ve been imperceptible to the human eye could be now visible for visual inspection. While upscaling is expected to address the poor quality of the original images, colourization will help distinguish astronomical objects and activites from the noise generated by various factors.

Figure A.5: Plagiarism report for Problem Definition and Scope (Part 3)

SmallSEOTools

PLAGIARISM SCAN REPORT

Words	130	Date	April 22,2021
Characters	878	Excluded URL	

0%	100%	0	8
Plagiarism	Unique	Plagiarized Sentences	Unique Sentences

Content Checked For Plagiarism

4.1 INTRODUCTION
4.1.1 Purpose and Scope of Document
The purpose of this project document is to create a comprehensive documentation about the methodology used in implementing the upscaling and colorization of the HSA images. The document explores the goals, objectives, resources, project plan, SRS and finally the summary and conclusion. It is hoped that this document would be beneficial in answering every query that any individual may have about this research project.
4.1.2 User profiles
Two actors have been defined in the Use case Diagram:
User: User preprocesses the image with the help of the GAN and facilitates the whole process.
GAN: It takes the input images, colorizes the image and feeds it to another GAN. The second GAN then upscales the image and a final output image is generated.

Sources	Similarity

Figure A.6: Plagiarism report for Software Requirement Specifications

SmallSEOTools

PLAGIARISM SCAN REPORT

Words	663	Date	April 22,2021
Characters	4412	Excluded URL	

8%	92%	3	34
Plagiarism	Unique	Plagiarized Sentences	Unique Sentences

Content Checked For Plagiarism

5.1 INTRODUCTION
The project is largely inspired by Christian Ledig's SRGAN paper [20] and Dong et. al [22] implementation of SRGANs using Tensorflow. Dahl et. al [10] introduction of residual encoding using VGG architecture and adaptation of GANs as conditional GANs by Mirza et al. [15] proved to be quite effective for implementing colorization of images. We provide detailed architectural design for each respective GANs and other networks that it will be compared with.
5.2 ARCHITECTURAL DESIGN
Generative Adversarial Networks (GANs) have two competing neural network models. The generator takes in the input and generates fake images. The discriminator gets the image from both the generator and the label along with the grayscale image and it determines which pair contains the real colored image. During training, the generator and the discriminator are playing a continuous game. At each iteration, generator produces a more realistic photo, while the discriminator gets better at distinguishing the fake photos. Trained in a minimax fashion, the goal is to train a generator that produces data that is indistinguishable from the real data.
5.2.1 Image Colorization
Both the generator and discriminator are conditioned on the input x with conditional GAN. Let the generator be parameterized by qg and the discriminator be parameterized by qd. The minimax objective function can be defined as:
Where, Gqg is the output of the generator and Dqd is the output of the discriminator.
We're currently not introducing any noise in our generator to keep things simple for the time being. Also, we consider L1 difference between input x and output y in generator. On each iteration, the discriminator would maximize qd according to the above expression and generator would minimize qg in the following way:
With GAN, if the discriminator considers the pair of images generated by the generator to be a fake photo (not well colored), the loss will be back-propagated through discriminator and through generator. Therefore, generator can learn how to color the image correctly. At the final iteration, the parameters qg will be used in our generator to color grayscale images.
5.2.2 Image Super-resolution
We use the SRResNet as the generator in the SRGAN model as used by Ledig et. al [20]. It contains both the residual blocks and the skip connections, as seen in Figure 5.3. Within each residual block, there are two convolution layers followed by a Batch Normalization layer and a parametric ReLU layer. Finally, the image is then upscaled 4 times using two sub-pixel convolution layers [24].
The goal of the generator is to produce high resolution images that will fool the discriminator of the GAN into thinking that it is receiving real instead of fake images. On the other hand the discriminator's goal is to classify the images it has received as either real images or generated images from the generator. The GANs objective function

Figure A.7: Plagiarism report for Detailed Design Document

nå
i=1
(ytrue□ypredicted)2 (2.1)
– This is a regression based approach and the pixel-wise L2 loss will impose
an averaging effect over all possible candidates and will result in
dimmer and patchy colorization
– Generative Adversarial Networks introduced by [13] use a minimax loss
which is different than the L2 loss as it will choose a color to fill an area
rather than averaging. This is similar to a classification based approach
• Image Upscaling
– One of the most popular approach to image upscaling was sparse-coding.
This approach assumes that images can be sparsely represented by a dictionary
of atoms in some transformed domain [21]. The dictionary is
learned during the training process.
– The main drawback for this was that the optimization algorithm was
computationally expensive
– Dong et. al explored super-resolution using convolutional neural network
and calling it SRCNN [22]. They explained how CNN had many similarities to the sparse-coding-based super-resolution.
– Kim et. al improved upon SRCNN's results using their very own model
inspired from the VGG-net architecture[23].
– After the introduction of GANs, Ledig et. al applied them to superresolution
(SRGAN) using a network inspired by the ResNets [20][12].
– SR-GAN works well with for single image super-resolution as it also
uses an intelligent content loss function that uses pre-trained VGG-net
layers. However, Ledig et. al noted that further information could be
used if this network were to be adapted to a video, such as temporal
information.
• A generative network, G, is meant to learn the underlying distribution of a data
set, Y. For e.g. we can train a GAN over face images to generate images similar
to those faces. With just a generative network however, we must visually
assess the quality of network outputs and judge how we can adapt the network
to produce more convincing results.
• With a discriminative network D, we can incorporate this tweaking directly
into training. The discriminative network takes in both fabricated inputs generated
by G and the real inputs from Y. It's sole purpose is to classify if the
input has come from G or Y.
• The key idea is back propagation of the gradients from the results of D's classification
to G so that G gets better at producing images and in turn fooling
D.
• For the project, we split the data into two categories: X that serves as the data
for the Y, which are its corresponding labels.
• G1 takes in a low resolution x 2 X which is black & white and produces ˆ y,
a colorized version of x. The descriminator D, in turn takes in a colorized
image and outputs the probability that the image comes from Y, instead of as
outputs from G, G(x). As such, if the discriminator is fooled by out generator, it should output a probability greater than 0.5 for the set of inputs coming from
G(x) and a probability less than 0.5 for images coming from Y.
• The same is the process for generator G2 with the only difference being that
the X is the set of colorized images but having low resolution and Y is the
set of high resolution images that serve as the labels for underlying mapping
of X. G2 takes in the low resolution image x 2 X and produces ˆ y and the
discriminator outputs a probability determining whether the image is superresolved
by G2 or the ground truth images from Y.

Sources	Similarity
Super-Resolution on Image and Video must then visually assess the quality of network outputs and judge how we can adapt the network to produce more con-vincing results. With a discriminative network, D, we can incorporate this tweaking directly into training. The discriminative net-work takes in both fabricated inputs generated by G and inputs coming from the "real" dataset, Y. Its purpose is to http://cs231n.stanford.edu/reports/2017/pdfs/312.pdf	3%

Figure A.7: Plagiarism report for Detailed Design Document (contd.)

SmallSEOTools

PLAGIARISM SCAN REPORT

Words	780	Date	April 22,2021
Characters	4938	Excluded URL	

0%	100%	0	46
Plagiarism	Unique	Plagiarized Sentences	Unique Sentences

Content Checked For Plagiarism

A.1 ARTIFICIAL NEURAL NETWORKS
Figure A.1 shows a schematic of the simplest multi-layer perceptron network, i.e. a feed-forward neural network. The network consists of an input layer, hidden layers and output layer. Define a`
i as the ith neuron of the `th layer and a`+1
j as the
jth neuron of the (`+1)th layer and w`
i j;b`
j is the weight and bias connecting the two
neurons, then the output of the `th layer is given by:
The loss is calculated using a loss function such as cross entropy function especially for binary classification.
where y0 2 f0;1g;y 2 (0;1). This objective is to minimize this cross entropy over a batch of all training data. This is done using gradient descent where the parameters viz. weights and biases are updated to reduce the overall cross entropy loss.
A.2 CONVOLUTIONAL NEURAL NETWORKS
A convolution is a linear operation that can be viewed as a multiplication or dot product of matrices. The input is a tensor of shape height x width x channels and the convolution operation abstracts the image to a feature map (also called a kernel) of shape kernelsize x kernelsize x kernelchannels. The layers can be computed by:
where ` is the layer, g is the activation function ReLU, Fj is the receptive field, k is the convolutional kernel and b is the bias.
A.3 RESIDUAL NETWORKS
Figure A.2 shows a building block of a residual network. The residual blocks can be expressed mathematically as follows. Let h(a) be an underlying mapping that is to be fit by a set of layers, where a is the input to these layers. If we hypothesize that multiple non-linear transformations by these layers can approximate the layer functions, then one can also hypothesize that a similar approximation can be made for the residual functions, i.e. h(a)□a, which have the same input and output dimensions. So instead of letting the underlying mapping be h(a) we approximate a residual function F(a) = h(a)□a. Thus, the actual function becomes h(a) = F(a)+a. We can achieve this approximation in a feed forward neural network using a series of skip connections that perform identity mapping and jump over a few layers. Adding the outputs of these two connections gives the final output layer. Thus, the residual unit can be defined as,
where a`+1 and a` represent the input and output for the `th layer and F is the residual function. The h denotes the operation in the identity mapping. The output of the layer is generally processed using an activation function such as ReLU. Let f be the ReLU activation and replacing F with the definition of feed forward activation, the

Figure A.8: Plagiarism report for Mathematical Model

residual block thus can be defined as,
For the sake of simplicity, we consider no operation being performed in the identity mapping. Thus, equation A.4 and equation A.5 become
A.4 GENERATIVE ADVERSARIAL NETWORKS
A generative network, G, is supposed to learn the underlying distribution of a latent space, Y. Instead of visually assessing the quality of network outputs and judge how we can adapt the network to produce convincing results, we incorporate automatic tweaking during training by introducing a discriminative network D. The network D takes in both the fabricated outputs generated by G and real inputs from the underlying distribution Y. The network produces a probability of the image belonging to the real or fabricated space.
Let x 2 X be a low resolution/grayscale image and y 2 Y be it's underlying distribution from the latent space Y. Generator G takes in input x and produces an output ˆ y. We define the mapping x! ˆ y in the following manner:
The discriminative network D is fed the fabricated mapping x ! ˆ y and the underlying distribution of x i.e. y 2 Y. The network D then produces a result that is a probability distribution of the input space indicating the class of the image that it thinks the input belongs to. We define this as:
where p 2 (0;1) is the probability that the image is fabricated or real. Both generator and discriminator are contitioned on the input x in conditional GAN. Let
the generator be parameterized by qg and the discriminator be parameterized by qd.
The minimax objective function can be defined as:
Where, Gqg is the output of the generator and Dqd is the output of the discriminator.
We're currently not introducing any noise in our generator to keep things simple for the time being. Also, we consider L1 difference between input x and output y in generator. On each iteration, the discriminator would maximize qd according to the above expression and generator would minimize qg in the following way:

Sources	Similarity

Figure A.8: Plagiarism report for Mathematical Model (contd.)

SmallSEOTools

PLAGIARISM SCAN REPORT

Words	542	Date	June 21,2021
Characters	3682	Excluded URL	

0%	100%	0	37
Plagiarism	Unique	Plagiarized Sentences	Unique Sentences

Content Checked For Plagiarism

To gather data, the primary module implemented was a data scraper written in node.js which uses puppeteer(headless chrome) to scrape the entire HLA(Hubble Legacy Archive) with the described RA and Dec co-ordinates of the night sky. The module ensured that, out of the entire data available in the Legacy Archive, a certain format of images would be downloaded and stored appropriately. After gathering the data, it was apparent that there were many other processed images present on the Hubble main and other citizen science projects conducted with Hubble Space Telescope. To scrape these, a python script using selenium was written so as to efficiently scrape processed images which had a minimum dimensions of 2562563. With the data gathered, the main components, i.e. the models were built.
The project entails a standard set of GAN modules, broadly, a data pre-processing pipeline, a training pipeline, intermediate state logging and a testing pipeline.
• Data Pre-processing
– The data pre-processing pipeline is an important part of the data pipeline that feeds the data to the training pipeline
– The images, being too large to be loaded into the memory simultaneously, had to be loaded through a data generator
– We implement an instance of Keras Image Data Generator to convert the raw data into training data
– We create two data generators, one for generating gray scale images and the other for generating subsequent RGB image
– We create another data pre-processor that converts the given RGB image into L*a*b color space
– The images are then resized to a dimension of 6464
– We get a vector of image batches which will be the input to the training pipeline
• Training Pipeline
– The training pipeline consists of multiple phases
– Initially, we build the basic neural networks for generator and discriminator networks, i.e. a tensor graph which will function as a computation graph for forward and backward propagation
– The next step is selection of an optimizing algorithm
– After selection of optimizer, a training pipeline is implemented which includes running a session of forward and backward propagation throughout the computation graph and optimizing the trainable variables of the networks
• Intermediate state logging
– Deep Learning models use up a lot of computational resources and are time consuming to train

Figure A.9: Plagiarism report Project Implementation

– Thus, it is essential to set up methods that will extract intermediate states of the model while it is still in the training loop
– These methods are often termed as 'callbacks' and we implement different callbacks to ensure availability of training history, weights and intermediate results during documentation
– A csv logger logs the metrics at the end of each epoch into a csv file for model evaluation
– The model weights are saved if the loss metric improves than the earlier epoch. We implement a mechanism for early stopping where the training loop exits if the model appears to have converged
• Testing pipeline
– Model testing is performed on a testing dataset that is different than the original dataset
– Testing pipeline is implemented by using the saved weights of the trained model and the images are processed through the entire dataset
– We evaluate the images qualitatively as well as quantitatively using visual inspection and distance metrics

Sources	Similarity

Figure A.9: Plagiarism report for Project Implementation (contd.)

SmallSEOTools

PLAGIARISM SCAN REPORT

Words	376	Date	June 21,2021
Characters	2494	Excluded URL	

0%	100%	0	24
Plagiarism	Unique	Plagiarized Sentences	Unique Sentences

Content Checked For Plagiarism

6.3 ALGORITHM DETAILS
6.3.1 Generative Networks
A generative model is a class of statistical models that are a contrast to the classifier models dubbed as discriminative models. Informally speaking, a generative model creates new data instances based on the distribution of the data itself. Generative models tackle a more difficult task, i.e. to model data. A generative model might capture correlations among underlying distribution and draw conclusions. Goodfellow et al. (2014) proposed the most famous type of generative model, i.e. Generative Adversarial Network, which is composed of two smaller networks called a discriminator and a generator. The generator's task is to create fake images that convince the discriminator and the discriminator usually classifies between real and fake data.
6.3.2 Adversarial Networks
With two networks that are neural networks, the adversarial modeling framework is the most straightforward to apply. A distribution pg over data x is called the generator's distribution. To learn this distribution, we define some prior noise on the input variables pz(z) and represent a mapping to a space with G(z;qg), where G is a differential function of neural network parameterized by qg. Another neural network D(x;qd) is defined such that it provides a scalar output, representing the probability
that data x comes from the input distribution rather than pg. The goal is to train D to maximize the probability of assignment of the correct label to both training examples and samples from G. The network G is trained to minimize log(1□D(G(z)).
6.3.3 Conditional GANs
As mentioned earlier, GANs are generative models that learn a mapping from a random noise vector z to output pg. The mapping from vector z to an output image y can be represented as G : z ! y (Goodfellow et al.; 2014). Isola et al. (2018) presented a new approach for GANs called Conditional GANs. Instead of learning just from a random noise vector, conditional GANs learn a mapping from input x and the vector z to y such that G : fx; zg!y. By doing this, we train the discriminator to do as well as possible to detect the generator's fake outputs whereas the generator is trained to produce results that are almost indistinguishable from the real data. The objective function of conditional GAN can be expressed as:

Sources	Similarity

Figure A.10: Plagiarism report for Algorithm Details

SmallSEOTools

PLAGIARISM SCAN REPORT

Words	783	Date	June 21,2021
Characters	5094	Excluded URL	

0%	100%	0	45
Plagiarism	Unique	Plagiarized Sentences	Unique Sentences

Content Checked For Plagiarism

6.4 METHODOLOGY
6.4.1 Image Color space
An RGB image is essentially a rank 3 tensor of height, width and color where the last axis contains the color data of our image. The data is represented in RGB color space which has 3 numbers for every pixel indicating the amount of Red, Green, and Blue values the pixel has. Figure 6.1 shows that in the left part of the “main image” has blue color so the blue channel of the image has higher values and turns dark.
In L*a*b color space, we have three numbers for each pixel but these numbers have different meanings. The first channel, L, encodes the Lightness of each pixel and when we visualize this channel it appears as a black and white image. The *a and *b channels encode how much green-red and yellow-blue each pixel is, respectively. In the following image you can see each channel of L*a*b color space separately.
In most of the papers listed in the Literature Survey, the L*a*b color space is widely used instead of RGB to train the models. Intuitively, to the train a model for colorization, we should give it a grayscale image and we hope that it colors it. In the L*a*b colorspace, the model is fed the L channel, which is essentially a grayscale image, and we perform computations to predict the other two channels (*a and *b). After the predictions, we concatenate the channels and get a colorful image. In case of RGB, there is a need to explicitly convert the three channels down to 1 to make it a grayscale image and then feed it to the network hoping that it predicts three numbers per pixel. This is an unstable task due to sheer increase in volume of combinations from two numbers to three.
We train models using both color spaces and empirically show the significance of the L*a*b colorspace.
6.4.2 Transferred learning and model tweaking
Isola et al. (2018) proposed a general solution to many image-to-image translation tasks. We propose a similar methodology as proposed in the paper with a few minor tweaks, so as to significantly reduce the amount of training data needed and minimize the training time to achieve similar results. Initially, we use a U-net for the generator. The encoder of our U-net is a pre-trained ResNet-18 network with half of it’s layers clipped off so as to get feature abstractions of the middle layers. Ledig et al. (2017) showed that training the generator separately in a supervised, deterministic manner helps it generalize the mapping from the input space to outputs. The idea solves the problem of ”The blind leading the blind” that is persistent in most image-to-image translation tasks to date.
We decide to use the Common Objects in Context (COCO) dataset to pre-train our generator independently using the imagenet weights. This training will be supervised and the loss function used is mean absolute error or the L1 Norm from the

Figure A.11: Plagiarism report Methodology

target image. Even though trained deterministically,the problem of rectifying incorrect predictions still persists due to constraints over the convergence of loss metric.
To combat this, we use this trained generator and train it once again in an adversarial fashion to generalize it further. We hypothesize that re-training with an adversarial will further rectify the subtle color differences that mae couldn't solve.
We use a pre-trained ResNet-50 with imagenet weights as our discriminator with last few layers clipped off. The discriminative network used is something called a "Patch Discriminator" proposed by Isola et al. (2018). In a vanilla discriminator proposed by Goodfellow et al. (2014), the network produces a scalar output, representing the probability that the data x belongs to the input distribution and not the generator distribution pg. Isola et al. (2018) proposed a modification to the discriminative network so as to produce, instead of a single scalar, a vector of probabilities representing different localities of the input distribution (image) x. For instance, a discriminator producing a 3030 vector represents the probabilities every receptive field that is covered by the output vector. Thus, we can localize the corrections in the image that the generator should make.
Finally, we use the trained generator and trained discriminator and fine-tune it to fit it on our data which is rather small in size compared to the previous datasets.
We train these networks in an adversarial fashion using the conditional GAN objective function (Isola et al.; 2018) and couple it with some noise introduced as the L1 norm of generated image tensor to the target image tensor. The reason behind this is to train the generator using an adversarial component while trying to minimize the Manhattan distance between the generator output and target vector space.

Sources	Similarity

Figure A.11: Plagiarism report for Methodology (contd.)

SmallSEOTools

PLAGIARISM SCAN REPORT

Words	965	Date	June 21,2021
Characters	6732	Excluded URL	

0%	100%	0	51
Plagiarism	Unique	Plagiarized Sentences	Unique Sentences

Content Checked For Plagiarism

7.1 TYPES OF TESTING
We test the modules of our entire setup to ensure that the modules are performing as they're intended to. We perform a kind of unit testing to evaluate the output of every function in the code.
7.2 TEST CASES AND TEST RESULTS
7.2.1 Data Gathering and Pre-processing
• Test Case 1: Image Scraping test
– Test Case Description:
The module function should scan through the legacy archive only in the described co-ordinates and scrape images that are RGB and of dimensions 256256
– Status: Pass
• Image Colorization:
– RGB Color Space:
* Test Case 2: Pre-processing System: Grayscale Images (Input distribution)
· Test Case Description:
The pre-processing function takes in a stream of raw data and converts the images into a numpy array with dimensions n 64641 where n is the batch size. The output image vector should contain black & white images
· Status: Pass
* Test Case 3: Pre-processing System: Color Images (Target distribution)
· Test Case Description:
The target distribution should contain color counter-parts of the respective input data. The function should output a numpy array
with dimensions n64643 where n is the batch size.
· Status: Pass
– L*a*b Color Space:
* Test Case 2: Pre-processing System: RGB to L*a*b conversion
· Test Case Description:
The pre-processing function takes in a stream of raw data and converts the images into a numpy array with dimensions n 64643 where n is the batch size. The output is an image tensor in L*a*b color space. The L channel is provided for the grayscale input
· Status: Pass
* Test Case 3: Pre-processing System: L*a*b to RGB conversion
· Test Case Description:
The function takes in its input as an image tensor in L*a*b colorspace.

Figure A.12: Plagiarism report Software Testing

The function should output a numpy array with dimensions
n64643 where n is the batch size and the image
output is converted back to the RGB space
· Status: Pass
• Image Super-resolution:
– Test Case 4: Pre-processing System: Low Resolution Images (Input distribution)
* Test Case Description:
The pre-processing function takes in the raw data and produces low
resolution 64643 images. The output should be a numpy vector
of n dimensions representing the mini-batch size
* Status: Pass
– Test Case 5: Pre-processing System: High Resolution Images (Target
distribution)
* Test Case Description:
The target distribution contains the labelled data that is a high resolution
image of dimensions 2562563. The function produces a
numpy vector of size n where n is the size of mini-batch
* Status: Pass
7.2.2 Model Implementation
• Image Colorization Model
– Test Case 1: Generator network
* Test Case Description:
The generator network is a tensor graph implemented in tensorflow.
The computation graph should have an input layer of dimensions
n64641 which is propagated through till the final layer. The
final layer output should be a tensor of dimensions n64643
where n is the mini-batch size
* Status: Pass
– Test Case 2: Discriminator network
* Test Case Description:
The discriminator network is a tensor graph implemented in tensorflow.
The computation graph should have an input layer of dimensions
n64643 which is propagated through till the final
layer. The final layer output should be a tensor of dimensions
n30301 where n is the mini-batch size
* Status: Pass
• Image Super-resolution Model
– Test Case 1: Generator network
* Test Case Description:
The generator network is a tensor graph. The computation graph
should have an input layer of dimensions n64643. The final
layer output should be a tensor of dimensions n2562563
where n is the mini-batch size
* Status: Pass
– Test Case 2: Discriminator network
* Test Case Description:
The discriminator network is a tensor graph. The computation graph
should have an input layer of dimensions n2562563. The
final layer output should be a tensor of dimensions n1 where n is
the mini-batch size
* Status: Pass
7.2.3 Training Loop
• Test Case 1: Forward propagation
– Test Case Description:
The forward propagation function should run the computation graph over
all the training samples and calculate losses of the generator and discriminator.
The loss should be calculated according to the GAN objective
function and final output should be the cost of the entire training epoch.
– Status: Pass
• Test Case 2: Backward propagation
– Test Case Description:
The backward propagation should calculate gradients of the objective
function based on the cost calculated during the forward propagation.
The gradients should be used to update the trainable variables of the network
using gradient descent with momentum and weighted average sum,

Figure A.12: Plagiarism report for Software Testing (contd.)

i.e. Adam.
– Status: Pass
• Test Case 3: Callbacks and intermediate results logging
– Test Case Description:
While training the model, the callback functionality should keep track of the intermediate training results. It should perform the following:
A csv file containing history of the entire training session
Save the model weights on a best performance basis by keeping track of the earlier losses
Keep track of the convergence and stop the model training if the model converges early
– Status: Pass
7.2.4 Evaluation system
• Model performance evaluation
– Test Case 1: Model history plotting
* Test Case Description:
The function should plot the model losses throughout the training loop and present a graph of the said plot
* Status: Pass
– Test Case 2: Output plotting
* Test Case Description:
The function should take in the predictions by the model and save the images with their respective ground truth images for evaluation
* Status: Pass
– Test Case 3: Model evaluation (L1 norm)
* Test Case Description:
The function should take in two numpy vectors consisting of predicted values and ground truths respectively. The function output should be a single number consisting of the L1 norm (Manhattan distance) between the target and predictions
* Status: Pass

Sources	Similarity

Figure A.12: Plagiarism report for Software Testing (contd.)

SmallSEOTools

PLAGIARISM SCAN REPORT

Words	343	Date	June 21,2021
Characters	2247	Excluded URL	

0%	100%	0	16
Plagiarism	Unique	Plagiarized Sentences	Unique Sentences

Content Checked For Plagiarism

8.1 DATASET AND EXPERIMENTAL SETUP
Initially, we started by scraping the data off Hubble Legacy Archive. Using puppeteer(headless chrome), we scraped off hundreds of thousands of colorized images it has available. The Hubble Legacy archive is slow and produces grainy images with lots of noise and unprocessed images. A filter for M101 (Messier 101) galaxy rendered more than 80 thousand images with a 1 degree difference between consecutive right ascension. The data is large and has no particularly efficient way to clean without human investment. Cleaning tens of thousands of images by handpicking noiseless and well colored images is time consuming. For training the SRGAN, we need high resolution, well colored images.
Consequently, we scraped the Hubble Heritage project instead. The Hubble Heritage project releases the highest-quality astronomical images. They are all stitched together, colorized and processed to eliminate noise. Hubble Heritage then selects the best, most striking of these for public release. However, there are only 150 of these images that are actually useful. We scraped images from the main Hubble website as well so as to increase the amount of data we had. This provided an extra approximately 1000 images. Each image is a JPG image with dimensions of 256256 pixels and contains 3 channels of RGB.
We use a mini-batch gradient descent with a batch size of 10, 16 and 32 for different iterations. We use Adam optimizer with b1 = 0:5 and b2 = 0:999. The generator and discriminator has a learning rate of 2e□4 which remains constant throughout the training. We train the model with different epochs ranging between 20,50 and 100, saving the best model weights determined by L1 norm between the output of the generator and the target image. We use early stopping with a patience value of 10. The image size is 6464. Our implementation is using Python, numpy, Tensorflow and tf-keras. It takes about 24 hours to complete training over the COCO dataset and about 12 to 13 hours to fit the model on given astronomical data on a NVIDIA Tesla K80 GPU.

Sources	Similarity

Figure A.13: Plagiarism report for Experimental Setup

SmallSEOTools

PLAGIARISM SCAN REPORT

Words	944	Date	June 21,2021
Characters	6254	Excluded URL	

0%	100%	0	59
Plagiarism	Unique	Plagiarized Sentences	Unique Sentences

Content Checked For Plagiarism

8.2 RESULTS AND DISCUSSIONS
In the following section, we briefly compare and summarize the results of the implemented architectures and evaluate their performance. The evaluation is done qualitatively as well as quantitatively. We compare the performance of each model using L1 and L2 norm distance between the predictions and targets. Though unreliable, this method provides us with a somewhat decent ground to perform a comparative study and evaluate the reliability of such metrics on GAN evaluation compared to qualitative, visual evaluation.
To evaluate the model performance by virtue of convergence of the objective function, we plot the generator loss throughout the training on two different networks. Figure 8.1a and 8.1b shows that the generator converges quite nicely. The discriminator on the other hand oscillates because of the convergence that the generator shows. GAN losses are pretty non-intuitive but we can draw some observations from the loss curves. It seems that the pattern of oscillation and convergence repeats when networks are trained in an adversarial fashion.
Figure 8.2a and 8.2b show the same trend repeating even when trained using pre-trained weights with state of the art networks that show excellent results. It is observed that the ResNet U-net with a pre-trained ResNet-18 for its backbone converges decently in the beginning but later shows spikes when nearing the end of training loop. The ResNet50 used as a discriminator shows identical behavior as the custom discriminator above refer fig.8.1b.
We draw some conclusions based on our understanding of the plots. It seems that the loss convergence doesn't signify whether the model is predicting expected results. The loss function just converges to the minimum value that the cost function descends to, permitted by the learning rate. It would normally signify that the GAN has found some optimum point in the vector space that is at the lowest potential and cant decrease any further. It essentially means that the GAN has leaned enough. Due to the large number of dimensions, owing to the high amount of trainable variables, such combinations, where the function converges, can be huge in volume. Thus, these numbers don't provide any better understanding of the bias or variance the model is facing. Also, we discover that if the loss hasn't converged well, it doesn't necessarily mean that the model hasn't learned anything. On visual inspection, the generated results show similar results to the ground truth, even with high generator losses. This might be due to presence of a content loss parameter in the loss function where we try to minimize the function by minimizing the L1 norm between the generator predictions and target.
The discriminator shows an increase in the objective loss function in the initial epochs and then settles down in the later phase around an oscillation point. This shows that the discriminator is converging to some permanent number or rather,

Figure A.14: Plagiarism report Results

oscillating around it. We assume this point to be a point of stability between the two networks as the networks are in a constant adversarial battle, meaning if one performs better, the other is bound to perform worse.

8.2.1 Image Colorization

We present a comparative study of the following models: Basic U-net generator with a custom, residual VGG16 discriminator, hereafter referred to as Basic U-net. A modified U-net with pre-trained ResNet18 as its backbone. We evaluate this model by training it in RGB color space to predict 3 numbers for every pixel and in L*a*b color space where the model predicts the a and b channel alone. We then further train this model to fine-tune it to the task of colorizing astronomical images. From figure 8.3 we observe that on this particular example, the Basic U-net performs better at predicting results that closely map to the ground truth but the finetuned ResNet-18 shows promising results over a larger set of inputs. It can also be observed that the Basic U-net architecture does a decent job of faking the sharpness in the original image and that makes the image appear more realistic as compared to rather blurry outputs from the other networks.

We also observe that the model trained in the RGB color space performs poorly at predicting values of the three color channels and that results in dominance of one channel (blue in this case) over others. This causes the output, even thoughreconstructed quite accurately, to have a varied color pattern with high emphasis on one color channel. This might be because of presence of deep layers which cause gradients of certain colors to diminish over time, causing the model to be strongly biased towards the blue color in this case. An increase in the volume of training data and some random pixel shuffling in forward propagation might solve this problem.

The pre-trained ResNet18 U-net performs decently with the weights gathered by training it over the COCO dataset. The model still lacks the specific coloring intuition in astronomical images and plainly colors specific parts of the images in light colors, leaving majority of the image unaltered. This causes the images to have a slight gray tinge.

Figure 8.6 shows how the other models perform in different color spaces. We can observe that the Basic U-net model performs good at predicting the outputs but doesn't predict the brightness level of pixels quite accurately. The model seems to be overfitting on the dataset and suffers a high variance on output because there are a lot of testing samples which demonstrate the poor performance of the Basic U-net model as can be seen in figure 8.4.Figure 8.5a shows the ground truth images and figure 8.5b shows the predictions of the ResNet18 full trained model in the L*a*b color space. This model seems to perform best, visually.

Sources	Similarity

Figure A.14: Plagiarism report for Results (contd.)

SmallSEOTools

PLAGIARISM SCAN REPORT

Words	613	Date	June 21,2021
Characters	4120	Excluded URL	

0%	100%	0	38
Plagiarism	Unique	Plagiarized Sentences	Unique Sentences

Content Checked For Plagiarism

To quantitatively estimate the model performance, we measure the L1 and L2 distance between the predictions and the ground truth images in both RGB as well as L*a*b colorspace.Table 8.3 shows that the RGB color space performs poorly on the task of colorization. In terms of the L1 distance, the best performance is achieved on the ResNet18 U-net with pre-trained weights. This goes to prove the unreliability of distance metrics in model performance evaluation. The model, although quantitatively, performs better than the fine-tuned model but in reality, fails to produce images that might be visually appealing. The L2 distance metric shows how the fine-tuned model might, in reality, be fitting slightly better over the data to predict correct color combination as its output. We still consider the results from the fine tuned model to be better than the other performing models and further investigate the per-channel predictions by the model.
Table 8.2 shows the RGB channel averages of the outputs produced by the fine tuned model. It can be observed that the feature embeddings produced by the model in *a, *b color spaces maps to the green channel with the least error. This might be indicative that the model is performing poorly on images that have a high content of red-blue colors.
To further evaluate the actual outputs produced in the L*a*b color space, we compare the *a, *b channels of the ResNet18 architectures.
In table 8.3, L1 distance, we can see the performance of fine-tuned model to be better in channel *a but poor in *b compared to the pre-trained model. The L2 distance metric rules out the possibility of the fine-tuned model performing better than the pre-trained model. In reality, the fine-tuned model is orders of magnitude better at predicting output abstractions with the L channel as its input, thus contradicting the quantitative results.
8.2.2 Image Super Resolution
We implement the basic SR-GAN proposed by Ledig and train it to improve super-resolution task. We compare the trained model with pre-trained SRGAN model, EDSR-GAN proposed by Lim et al. (2017) and WDSR-GAN proposed by Yu et al. (2018).Figure 8.7 shows how the networks perform on the task of predicting pixels while upscaling of the image. All the networks perform really well and its difficult to distinguish the outputs visually. It may be observed that the fine-tuned network produces images that look slightly better than the other counter-parts. The best performing network seems to be the WDSR-GAN with a very little error in predicting the output.
It is evident that the model produces acceptable results on visual inspection. The main reason behind this might, again, be the random pixel shuffling between every upscaling pass.
As opposed to colorization, super-resolution needs a quantitative estimation

Figure A.14: Plagiarism report for Results (contd.)

to determine which model performs best among the give models.
Table 8.4 shows the L1 and L2 distances of predicted results by each model with the ground truth image. It is observed that Ledig's SRGAN, after a bit of finetuning performs really well in comparison to the the pre-trained version. To further improve this, Lim et al. (2017) proposed an optimized version of SRGAN by removing the unnecessary modules in the conventional resnets and showed that Enhanced Deep Residual Networks performed better at upscaling task. When trained in an adversarial manner, the ED-SRGAN performs better than the traditional SRGAN. Yu et al. (2018) further improved the idea by increasing the widening factor (2 and 4) to (6 and 9). This Wide Activation Deep Super Resolution network further improved the performance for single image super-resolution. When we implement this in an adversarial manner, we achieve excellent results. It is evident from the results that the best performing network is the WDSR.

Sources	Similarity

Figure A.14: Plagiarism report for Results (contd.)

SmallSEOTools

PLAGIARISM SCAN REPORT

Words	625	Date	June 21,2021
Characters	4402	Excluded URL	

0%	100%	0	38
Plagiarism	Unique	Plagiarized Sentences	Unique Sentences

Content Checked For Plagiarism

9.1 SUMMARY AND CONCLUSION
The project mainly tackles the problem of colorizing astronomical images and super-resolving them for astronomical inspection. We explore various methodologies proposed till date that efficient colorize and super-scale images with results that are significantly closer to the ground truth distribution. We scrape the data from Hubble Legacy Archive, Hubble Main website and Hubble Heritage project and created a filtered and clean dataset of 4700 images. We split the data and use 500 images, roughly 10% of the data for testing purposes. To compensate for the lack of data, we implement several pre-trained architectures and fine-tune their abstractions over our dataset to find the most effective solution.
For the colorizing model, we explore usage of U-net architectures starting from a basic U-net model and experiment with different color spaces and empirically confirm the superiority of L*a*b color space in image colorization problem. We use the state of the art ResNet-18 to provide as a backbone of the encoder and build a U-net around it. The pre-trained network over COCO dataset in RGB colorspace produces significantly weaker results as compared to the subsequent network in L*a*b colorspace. The best performing model turns out to be the ResNet18 U-net which is fine-tuned over our particular dataset to produce appealing and similar results to the ground truth.
The Super-resolution model is based largely on the SRGAN proposed by ?. We use the generator weights and sample results from the training set to inspect the results. It is found that the model performs really well on the pre-trained weights and we decide to fine-tune it to our application. After fine-tuning the model, we train other state of the art single image super-resolution models such as EDSR (?) and WDSR (?). These provide further insights into the problem and simultaneously, improve the results.
While studying and improving the performance of these models, we explore performance metrics of GANs and evaluation methodologies implemented to test out conditional GANs. It is evident that the loss curves of generator and discriminator do not provide us with any intuition about the model performance. We also discover that standard distance metrics cannot be used to evaluate GANs and quantitative methods that exist to evaluate GANs are unreliable. We prove so by contradiction of qualitative samples and quantitative measurements of the best performing architecture for colorization models. However, we observe that quantitative estimation is quite reliable for the problem of single image super-resolution and can be helped to determine which model is better suited for the task.
9.2 FUTURE SCOPE
Though we obtain moderately good results, a vast amount of algorithms still remain unscratched. A more powerful model such as SN-ResNext, EfficientNet and

Figure A.15: Plagiarism report Summary and Conclusion

more state-of-the-art models can be implemented and trained over millions of images from the Imagenet. With even more hardware resources and availability of data, we can explore computationally heavy models for a better approximation. An image stitching algorithm can be applied on the produced images to generate large scale astronomical images for scientific study. Colorization can be improved by the virtue of exploring different loss functions using weighted losses to reduce loss problem for low saturation regions. We can introduce a gradient penalty for the SRGAN architecture and include the WGAN (?) which will stabilize the discriminator to stay within the space of 1-Lipschitz function. Progressively growing GANs (?) can be applied so that the dimensions can be further improved with more stability and greater sharpness.

9.3 APPLICATIONS

The images are upscaled and colourized using a completely automated algorithm which uses Deep Learning. Generative Adversarial Networks are successfully used for the implementation. The algorithm can directly be applied for creating images that can be studied by astronomers. It is anticipated that this will aid the astronomers vastly in their efforts.

Sources	Similarity

Figure A.15: Plagiarism report for Summary and Conclusion (contd.)