Colorizing Monochromatic Images Using Deep Learning

A Mini Project Report Submitted
In partial fulfillment of the requirement for the award of the degree of

Bachelor of Technology
In
Computer Science and Engineering -Artificial Intelligence and Machine Learning

By

A. Akshitha	**-**	**21N31A6601**
B. Nirmala Bai	**-**	**21N31A6622**
CH. Harshini	**-**	**21N31A6639**

Under the Guidance of

Ms. P. Swapna
Assistant Professor
Computational Intelligence Department
MRCET

DEPARTMENT OF COMPUTATIONAL INTELLIGENCE
MALLA REDDY COLLEGE OF ENGINEERING AND TECHNOLOGY
(Affiliated to JNTU, Hyderabad)
ACCREDITED by AICTE-NBA
Maisammaguda, Dhulapally post, Secunderabad-500014.

2024-2025

DECLARATION

I hereby declare that the project entitled **“Colorizing Monochromatic Images Using Deep Learning”** submitted to **Malla Reddy College of Engineering and Technology**, affiliated to Jawaharlal Nehru Technological University Hyderabad (JNTUH) for the award of the degree of **Bachelor of Technology** in **Computer Science and Engineering- Artificial Intelligence and Machine Learning** is a result of original research work done by me.

It is further declared that the project report or any part thereof has not been previously submitted to any University or Institute for the award of degree or diploma.

A. Akshitha(21N31A6601)

B. Nirmala Bai (21N31A6622)

CH. Harshini(21N31A6639)

MALLA REDDY COLLEGE
OF ENGINEERING & TECHNOLOGY
(AUTONOMOUS INSTITUTION - UGC, GOVT. OF INDIA)
Affiliated to JNTUH; Approved by AICTE, NBA-Tier 1 & NAAC with A-GRADE | ISO 9001:2015

CERTIFICATE

This is to certify that this is the bonafide record of the project titled **"Colorizing Monochromatic Images Using Deep Learning"** submitted by **A. Akshitha (21N31A6601), B. Nirmala Bai(21N31A6622), CH. Harshini(21N31A6639),** of B.Tech in the partial fulfillment of the requirements for the degree of **Bachelor of Technology** in **Computer Science and Engineering- Artificial Intelligence and Machine Learning**, Dept. of CI during the year 2024-2025. The results embodied in this project report have not been submitted to any other university or institute for the award of any degree or diploma.

Ms. P. Swapna	**Dr. D. Sujatha**
Assistant Professor	Professor
INTERNAL GUIDE	**HEAD OF THE DEPARTMENT**

EXTERNAL EXAMINER

Acknowledgement

We feel honored and privileged to place our warm salutation to our college Malla Reddy College of Engineering and technology (UGC-Autonomous), our Director **Dr. VSK Reddy** who gave us the opportunity to have experience in engineering and profound technical knowledge.

We are indebted to our Principal **Dr. S. Srinivasa Rao** for providing us with facilities to do our project and his constant encouragement and moral support which motivated us to move forward with the project.

We would like to express our gratitude to our Head of the Department **Dr. D. Sujatha** for encouraging us in every aspect of our system development and helping us realize our full potential.

We would like to thank our mini project coordinator **Dr. P. Hari Krishna, Associate Professor** as well as our internal guide **Ms. P. Swapna, Assistant Professor** for their structured guidance and never-ending encouragement. We are extremely grateful for valuable suggestions and unflinching co-operation throughout mini project work.

We would also like to thank all supporting staff of department of CI and all other departments who have been helpful directly or indirectly in making our mini project a success.

We would like to thank our parents and friends who have helped us with their valuable suggestions and support has been very helpful in various phases of the completion of the mini project.

By

A. Akshitha(21N31A6601)

B. Nirmala Bai (21N31A6622)

CH. Harshini(21N31A6639)

ABSTRACT

Colorization of grayscale images has become a more researched area in the recent years. A pragmatic approach to the task is to implement sophisticated image colorization techniques. We Implement colorization using different CNN models and provide their quantitative and qualitative comparison for colorization. The aim was to colorize photos and videos which were monochromatic. We have used Convolutional networks and associated models such as a pretrained Caffe (Convolutional Architecture for Fast Feature Embedding) model developed by University of California. We have used LAB (Lightness Channel A and B) forms of the images to colorize both static images and dynamic videos. Post conversion, different metrics were used to compare quality distortions. The project could be helpful for quality optimization analysis for various footages shot on black and white cameras. DeOldify is a recent automatic colorization method based on Convolutional Neural Networks which yields impressive results. The method was initially created by Jason Antic with the support of the Californian start-up Fast.ai and thus does not come from the academic research world. It uses a new type of GAN training method called NoGAN. DeOldify is an open source deep learning model, used to add high-quality colorization to grayscale images and videos with amazing results. Simply put, the goal of this deep learning model is to colorize, restore, and give new life to old images and videos. We have used DeOldify to colorize monochromatic videos.

TABLE OF CONTENTS

S.No. Topic Page No.

CHAPTER 1: INTRODUCTION 1-5

1.1 Purpose 2

1.2 Background of project 3

1.3 Scope of project 4

1.4 Project features 4

CHAPTER 2: SYSTEM REQUIREMENTS 6-9

2.1 Hardware Requirements 6

2.2 Software Requirements 6

2.3 Existing System 6

2.4 Proposed System 7

2.5 Functional Requirements 7

2.6 Non - Functional Requirements 8

CHAPTER 3: TECHNOLOGIES AND LANGUAGES USED 10-11

CHAPTER 4: SYSTEM DESIGN 12-17

4.1 Architecture 13

4.2 UML Diagrams 14

CHAPTER 5: IMPLEMENTATION 18-23

5.1 Code for Image Colorization 18

5.2 Code for Video Colorization 20

5.3 Output Screens 21

CHAPTER 6: CONCLUSION AND FUTURE SCOPE 24-24

6.1 Conclusion 24

6.2 Future Scope 24

REFERENCES 25

CHAPTER-1
INTRODUCTION

Black and white photography has been a beloved medium of visual storytelling and documentation for over a century. However, the absence of color in black and white images can sometimes limit our ability to understand and appreciate the past with realism and accuracy. Image colorization, the process of adding color to black and white images, has emerged as an exciting and innovative technique that can enhance our visual experience of historical images.

The use of neural networks for image colorization has been a recent development in this field. Neural networks, specifically convolutional neural networks (CNNs), have shown great promise for image colorization tasks. By training a neural network on a large dataset of colored and grayscale images, the network can learn to identify patterns and features in the grayscale values and predict the corresponding colors. This technique has proven to be successful in accurately adding color to black and white images while preserving their original structure.

Image colorization using neural networks has various applications in fields such as art, entertainment, and historical documentation. For instance, it can be used to bring historical figures and events to life, allowing us to better understand and appreciate the past. It can also be used in scientific and medical imaging, where color can be used to highlight specific features or patterns in the data.

However, image colorization using neural networks is not without its challenges. One of the main challenges is the quality of the dataset used to train the network. A biased or in complete dataset can lead to inaccurate or unrealistic colorizations. Additionally, neural networks can introduce artifacts into the image, which can distort the original image and compromise its accuracy.

Despite these challenges, image colorization using neural networks is a rapidly evolving field with exciting potential. In this article, we will explore the techniques, challenges, and applications of image colorization using neural networks. We will discuss the different types of neural networks used for image colorization and the challenges associated with the process. We will also examine the various applications of this technology in fields such as art, entertainment, and historical documentation.

Image colorization using neural networks is a process that aims to add color to grayscale images. It involves the use of deep learning techniques to analyze the grayscale image and predict the color values for each pixel in the image. Neural networks are a type of machine learning algorithm that is modeled on the structure and function of the human brain, and they are particularly well-suited for tasks such as image processing.

The basic process of image colorization using neural networks involves training the network on a large dataset of colored and grayscale images. During training, the network learns the relationship between grayscale values and their corresponding color values. This allows the network to make accurate predictions of the color values for new grayscale images.

There are several types of neural networks that can be used for image colorization, including convolutional neural networks (CNNs) and generative adversarial networks (GANs). CNNs are particularly well-suited for image processing tasks, and they are often used in image colorization applications. GANs are a more recent type of neural network that can generate new images that are similar to existing images. They have been used in image colorization applications to generate realistic colorizations of grayscale images.

Image colorization using neural networks has numerous applications in various fields, including photography, film, art, and medical imaging. In the field of photography, image colorization can be used to enhance the visual appeal of historical photos or to add color to black and white images. In film, it can be used to add color to classic movies, making them more engaging for modern audiences.

1.1 Purpose:

i. **Restoring Old Photos:** Many historical or archival images are available only in black and white. Colorization can bring these photos to life, making them more relatable and engaging for modern audiences, as well as aiding in historical interpretation.

ii. **Improving Computer Vision Models:** Colorized images can enhance the training of computer vision models, particularly for tasks where color is important for recognition, such as in object detection or segmentation. Colorization can be used as a pre-processing step for improving model performance on grayscale datasets.

iii. **Artistic and Creative Uses:** Artists, photographers, and graphic designers can use automated colorization to experiment with different color palettes and styles. It allows rapid generation of visually appealing versions of monochromatic images.

iv. **Enhancing Visual Data for Scientific Applications:** In scientific fields such as medical imaging or satellite imagery, grayscale images are often used (e.g., X-rays, MRI scans, or radar images). Colorizing these images can help highlight specific features or patterns for better interpretation and analysis.

v. **Training and Learning Deep Learning Models:** Colorization is a challenging problem that combines multiple aspects of machine learning, such as feature extraction, pattern recognition, and image generation. It serves as an excellent project for training and learning complex deep learning models, including convolutional neural networks (CNNs) and generative adversarial networks (GANs).

1.2 Background of the Project:

Images are the primary source for understanding the world and obtaining information, while grayscale images cannot provide sufficient information. Compared with grayscale images, colorful images contain richer information and convey more intuitive feelings. However, due to the limitations of previous technology or special scenes, images generated in many situations are grayscale. If these grayscale images can have rich and reasonable color information, it can help humans better identify target objects and understand the scene. Potential use cases for this research include restoration of old black&white photo, automatic colorization of anime sketches ,grayscale images in medical fields etc. Colorization needs to estimate the missing color channels from a grayscale value, and reasonable colorization solutions are not unique. Colorization is still a challenging task due to its uncertainty and diversity.

1.3 Scope of project:

The scope of a deep learning project for colorizing monochromatic images involves various stages, starting with the collection and preparation of a large dataset of paired grayscale and colored images for model training. The project encompasses the design and experimentation with different deep learning architectures such as Convolutional Neural Networks (CNNs), Generative Adversarial Networks (GANs), and U-Nets to automate the colorization process. It includes optimizing the model using appropriate loss functions, such as perceptual loss and adversarial loss, and fine-tuning hyperparameters for high-quality, realistic output. The scope also covers evaluating the model's performance using both objective metrics (PSNR, SSIM) and subjective human judgment, as colorization is inherently ambiguous. The project may extend to domain-specific applications, like historical photo restoration, medical imaging, or satellite data analysis, and could incorporate user interactivity, allowing manual adjustments for more control over colorization. Real-time colorization for video or mobile applications is also within the scope, as is deployment in user-friendly interfaces. Additionally, ethical considerations, particularly regarding historical accuracy, and further research into improving model efficiency, such as using self-supervised learning, are vital parts of the scope. The ultimate goal is to create a robust, scalable solution for automatic colorization with practical applications in various industries.

1.4 Project Features:

i. **Automatic Colorization:** The system automatically predicts and applies colors to a grayscale image based on learned patterns from large datasets, requiring no manual intervention.

ii. **Contextual Understanding:** Models can infer color information based on the context of the image, such as objects, scenes, or lighting conditions, enabling realistic color application (e.g., grass being green, sky being blue).

iii. **Perceptual and Adversarial Loss:** Uses advanced loss functions like perceptual loss (which compares high-level image features) and adversarial loss (GAN-based) to produce colors that are visually appealing and realistic, not just mathematically accurate.

iv. **Transfer Learning and Pre-trained Models:** Leverages pre-trained models on large datasets (such as ImageNet) to transfer knowledge for faster training and improved performance on smaller or domain-specific datasets.

v. **User Interactivity:** Offers features for user input, allowing manual control over specific areas or colors in the image, ensuring that users can fine-tune the result according to their preferences historical accuracy.

CHAPTER-2

SYSTEM REQUIREMENTS

2.1 Hardware Requirements:

- Processor: Intel i7
- RAM: 8 GB
- Hard disk: 256 GB
- Speed: 2.7 GHZ

2.2 Software requirements:

- Operating System: Windows 7 or above
- PYCHARM
- Graphical User Interface
- Backend: Python

2.3 Existing System:

i. Existing systems for colorizing black and white images range from traditional manual methods and rule-based approaches to deep learning-based techniques.

ii. Manual colorization, though accurate, is time-consuming and requires expertise.

iii. Rule-based and example-based methods often produce unnatural results due to their reliance on predefined rules or closely matching reference images.

Drawbacks of existing system:

i. The layers does not determine color and brightness.
ii. The main drawback in existing system is it can’t be used for complex things due to restricted data.
iii. For historical photos, getting accurate colors can be difficult without historical references.

2.4 Proposed System:

i. Here we will be using OpenCV, CNN, pre-trained Caffe model.
ii. The main drawback in existing system is it can’t be used for complex things due to restricted data.
iii. So to overcome this we will be using caffe model and convert the grayscale image to LAB color space then to RGB.
iv. This process is done by converting the input into the desired output.
v. We use Graphical User Interface by which the user can interact with the system.
vi. And in this we are implementing DeOldify model for colorizing of videos.

2.5 Functional Requirements:

i. **Colorization:** The system should be able to accurately colorize monochrome images, producing realistic and visually appealing results.
ii. Image Input: The system should accept various image formats (e.g., JPEG, PNG, BMP) as input.
iii. **Output Format:** The system should be able to output colorized images in different formats (e.g., JPEG, PNG, BMP).
iv. **Batch Processing:** The system should allow for batch processing of multiple images simultaneously.
v. **Customizable Parameters:** The system should provide options to adjust parameters (e.g., color palette, saturation, brightness) to fine-tune the colorization process.
vi. **Real-time Colorization:** The system should be capable of processing images in real-time for applications like live video streams or interactive demonstrations.

vii. **Pre-trained Models:** The system should offer pre-trained models for various image domains (e.g., portraits, landscapes, architecture) to improve accuracy and efficiency.

viii. **User-defined Models:** The system should allow users to train their own models using custom datasets for specific applications.

ix. **Integration with Other Tools:** The system should be compatible with other image processing or machine learning tools (e.g., OpenCV)

2.6 Non-Functional Requirements:

i. **Performance:** The system must process images and deliver results within a specified time frame (e.g., 1-2 minutes for a standard-resolution image).It should handle multiple concurrent users and simultaneous image uploads without significant performance degradation.

ii. **Scalability:** The system should scale to handle a large number of users and high volumes of image uploads, both in terms of computing resources and storage. It must support both single-image and batch processing without a noticeable drop in performance.

iii. **Usability:** The user interface must be intuitive and easy to navigate, even for non-technical users. The system should require minimal user input, with clear guidance and tooltips where needed. Feedback during the image processing (e.g., progress bars) should be provided to enhance the user experience.

iv. **Reliability:** The system must be highly reliable, with a minimal rate of failure in processing images. It should recover gracefully from unexpected errors or interruptions (e.g., server crashes), with mechanisms in place to retry failed operations.

v. **Availability:** The system should be available 99.9% of the time, with scheduled downtime minimized and communicated in advance to users. Cloud or distributed systems should ensure high availability across different regions.

vi. **Security:** User-uploaded images must be securely stored and processed to prevent unauthorized access or data breaches. The system must comply with data privacy standards (e.g., GDPR, if applicable), ensuring that user data (images) is not stored longer than necessary. Secure communication protocols (e.g., HTTPS) must be used for all data exchanges.

vii. **Portability:** The system should be compatible across multiple platforms (e.g., desktop, mobile) and browsers (e.g., Chrome, Firefox, Safari). It should be deployable across different environments (e.g., cloud services, local servers) with minimal configuration.

viii. **Responsiveness:** The system should respond quickly to user interactions, providing feedback within 1 second for UI actions (e.g., file uploads, button clicks).It must provide timely feedback to users, especially during long-running processes like image processing.

CHAPTER-3

TECHNOLOGIES USED

Technologies and Languages used to Develop

NumPy: NumPy is a popular Python library for scientific computing that provides support for creating multidimensional arrays, mathematical functions to operate on these arrays, and tools for working with them. It is widely used in fields such as data science, machine learning, engineering, and scientific research.

NumPy provides an efficient and convenient way to perform mathematical operations on large arrays of data. It is built on top of the low-level C programming language, which allows it to take advantage of the speed and efficiency of the underlying hardware. This makes it much faster than using traditional Python data structures like lists for large-scale numerical computations.

Convolutional Neural Networks (CNNs): A Convolutional Neural Network (CNN) for colorizing monochromatic (grayscale) images typically works by taking a grayscale image as input and predicting the color information for each pixel. The CNN learns to map grayscale intensities to color channels (usually RGB) through layers of convolution, non-linear activation, and up sampling. The network is trained on large datasets of paired grayscale and color images, using loss functions that measure how well the predicted colors match the true colors.

Convolutional Architecture for Fast Feature Embedding (Caffe): A Caffe model for colorizing monochromatic images uses a pre-trained deep CNN to predict color information for grayscale images. The model is trained on large datasets of color images and applies layers of convolutions and up sampling to generate RGB channels from grayscale inputs. Caffe provides an efficient framework to implement, train, and test such models.

DeOldify: DeOldify is a deep learning-based model for colorizing monochromatic (black-and-white) images and videos. It uses a combination of a GAN (Generative Adversarial Network) and a pre-trained CNN to produce realistic and vibrant colors. The model is trained on large datasets and applies attention mechanisms for better focus on details, achieving high-quality colorization results.

Tkinter: Tkinter is a Python GUI library that can be used to create simple applications for colorizing monochromatic images by integrating it with image processing libraries like OpenCV or PIL (Pillow). Tkinter handles the user interface (loading images, displaying results), while the actual colorization logic is performed by other tools or models like CNNs.

CHAPTER-4

SYSTEM DESIGN

ALGORITHM AND ARCHITECTURE

The basic workflow of the project is schematically shown in the figure below. The basic working of the functions in the module are detailed below in sub sections. The algorithm and structure of the code is presented here, but the actual code can be found in appendix.

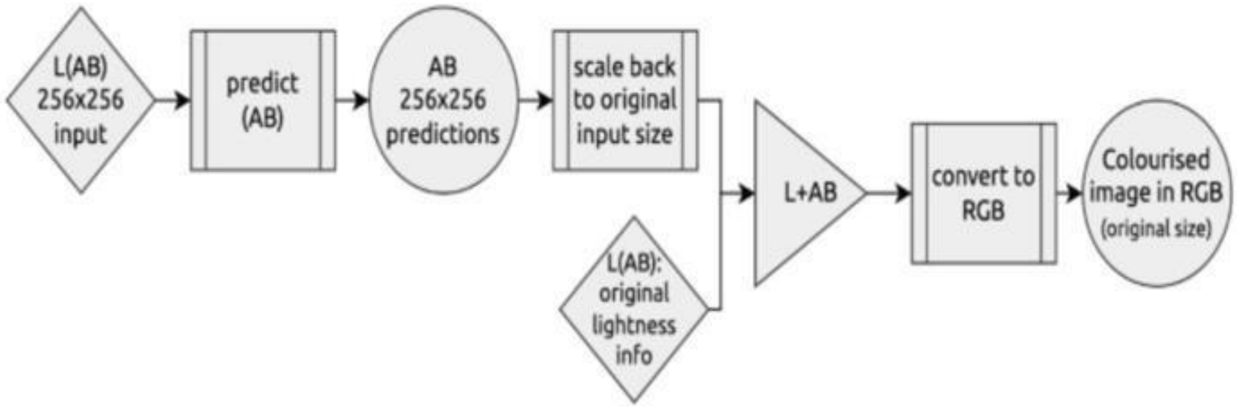

L(AB) 256x256 input
predict (AB)
AB 256x256 predictions
scale back to original input size
L(AB): original lightness info
L+AB
convert to RGB
Colourised image in RGB (original size)

Fig:1 Workflow

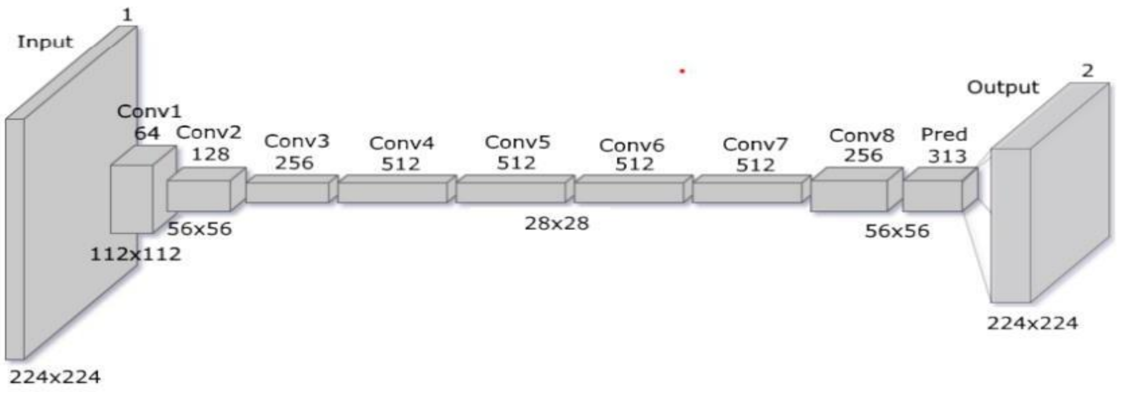


Fig:2 Plain CNN network architecture

4.1 System Architecture

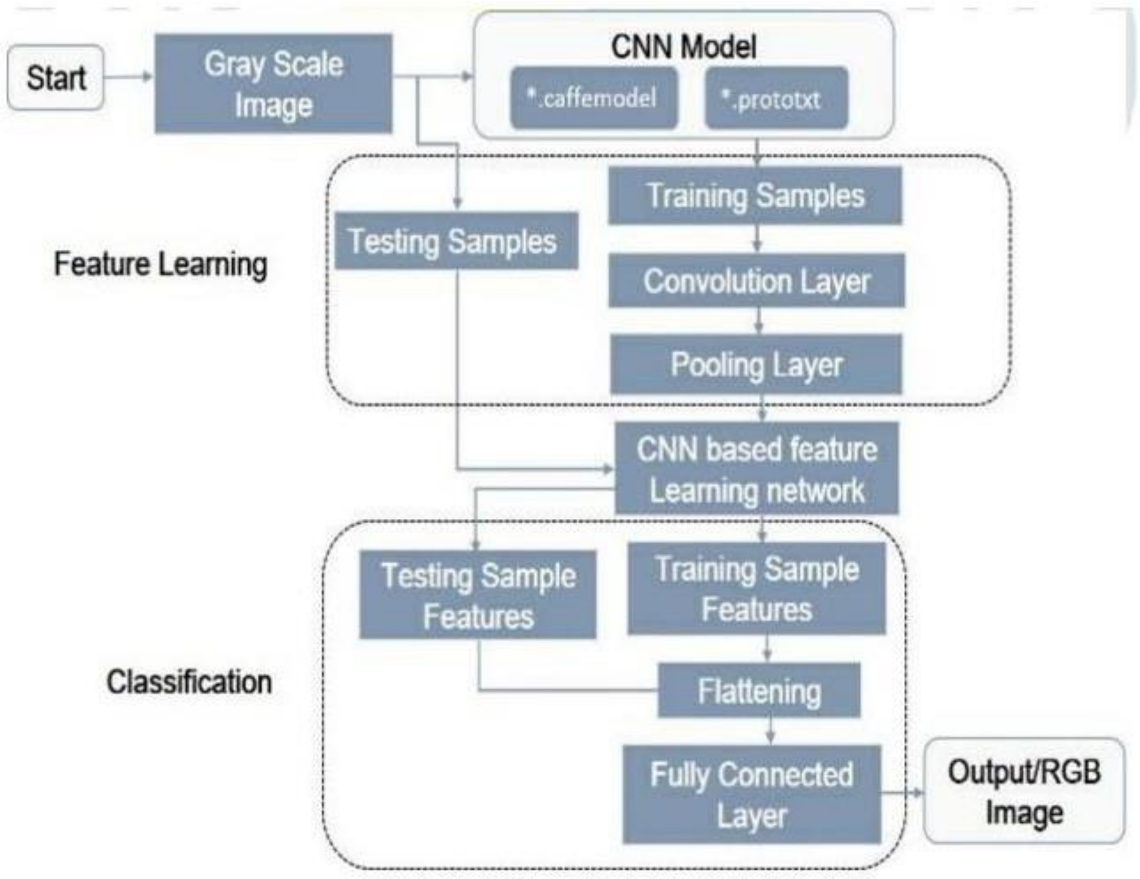


Fig:4.1 Architecture

4.2 UML Diagrams

4.2.1 Use case diagram

A use case diagram is used to represent the dynamic behavior of a system. It encapsulates the system's functionality by incorporating use cases, actors, and their relationships. It models the tasks, services, and functions required by a system/subsystem of an application. It depicts the high-level functionality of a system and also tells how the user handles a system.

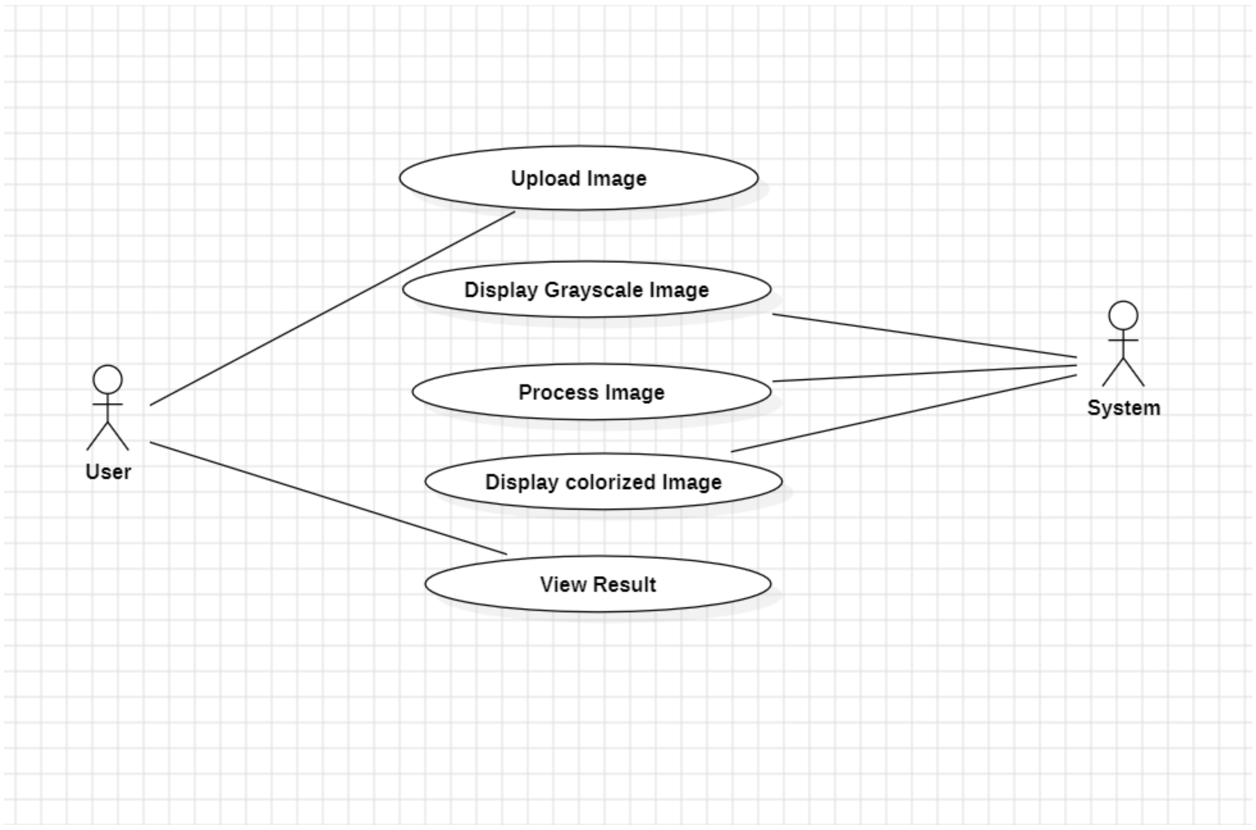


Fig:4.2 Use Case diagram

4.2.2 Sequence diagram

A sequence diagram is a type of interaction diagram because it describes how and in what order a group of objects works together. These diagrams are used by software developers and business professionals to understand requirements for a new system or to document an existing process. Sequence diagrams are sometimes known as event diagrams or event scenarios.

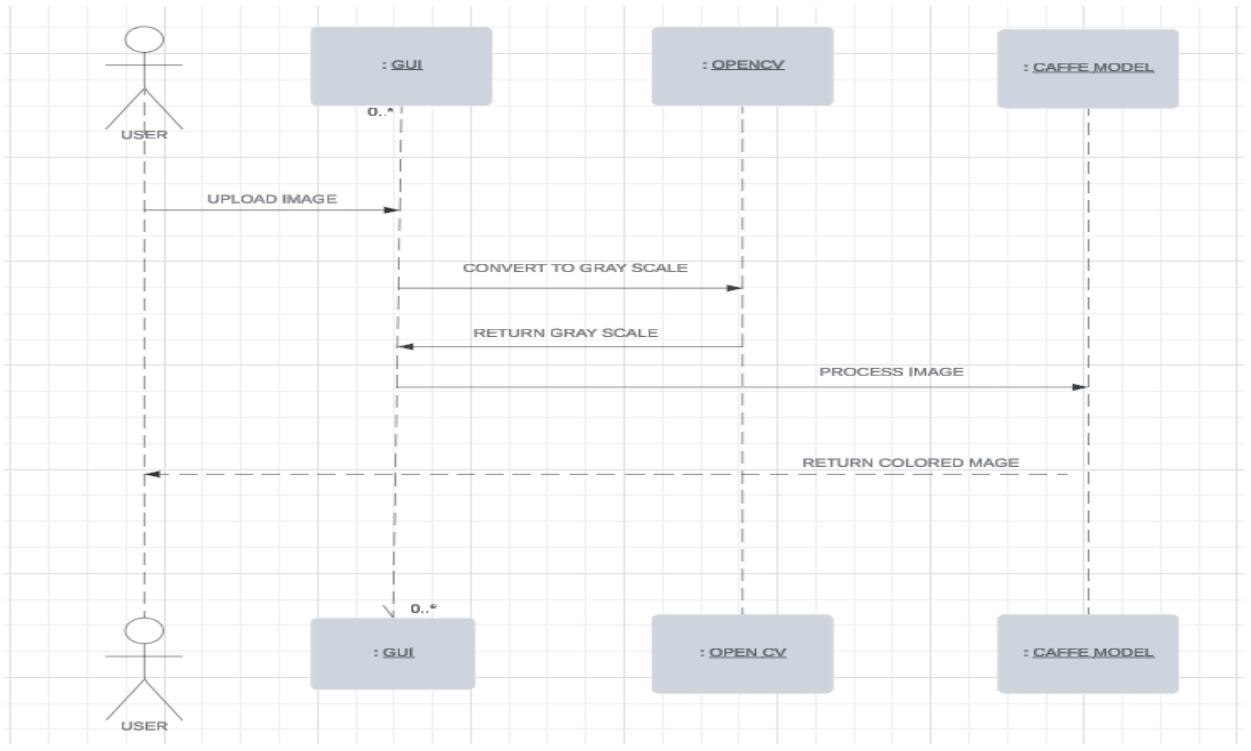


Fig:4.3 Sequence Diagram

4.2.3 Class Diagram

A class diagram depicts a static view of an application. It represents the types of objects residing in the system and the relationships between them. A class consists of its objects, and also it may inherit from other classes. A class diagram is used to visualize, describe, document various different aspects of the system, and also construct executable software code.

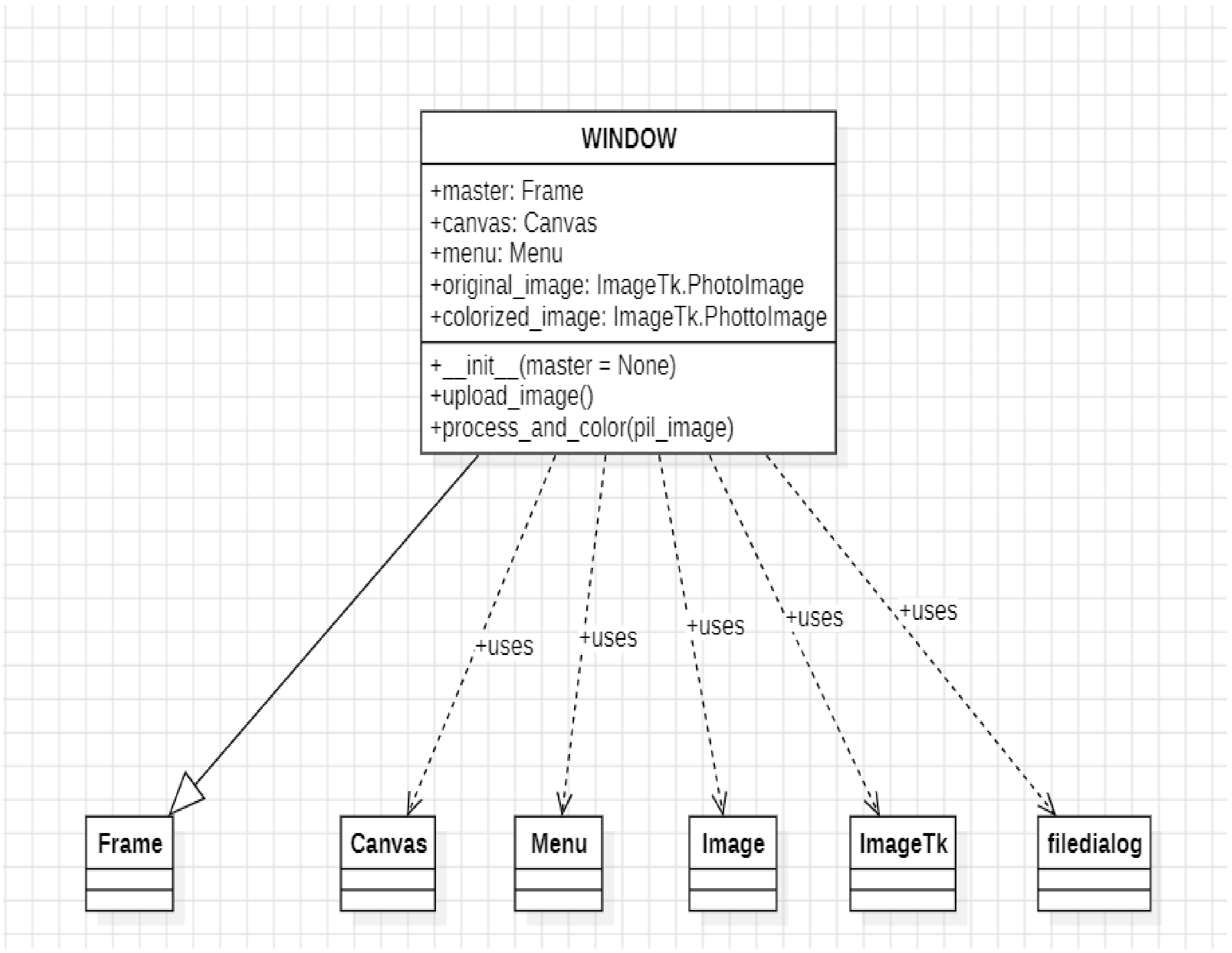


Fig:4.4 Class diagram

4.2.4 Activity diagram

An activity diagram is a type of Unified Modeling Language (UML) flowchart that shows the flow from one activity to another in a system or process. It's used to describe the different dynamic aspects of a system and is referred to as a 'behavior diagram' because it describes what should happen in the modeled system.

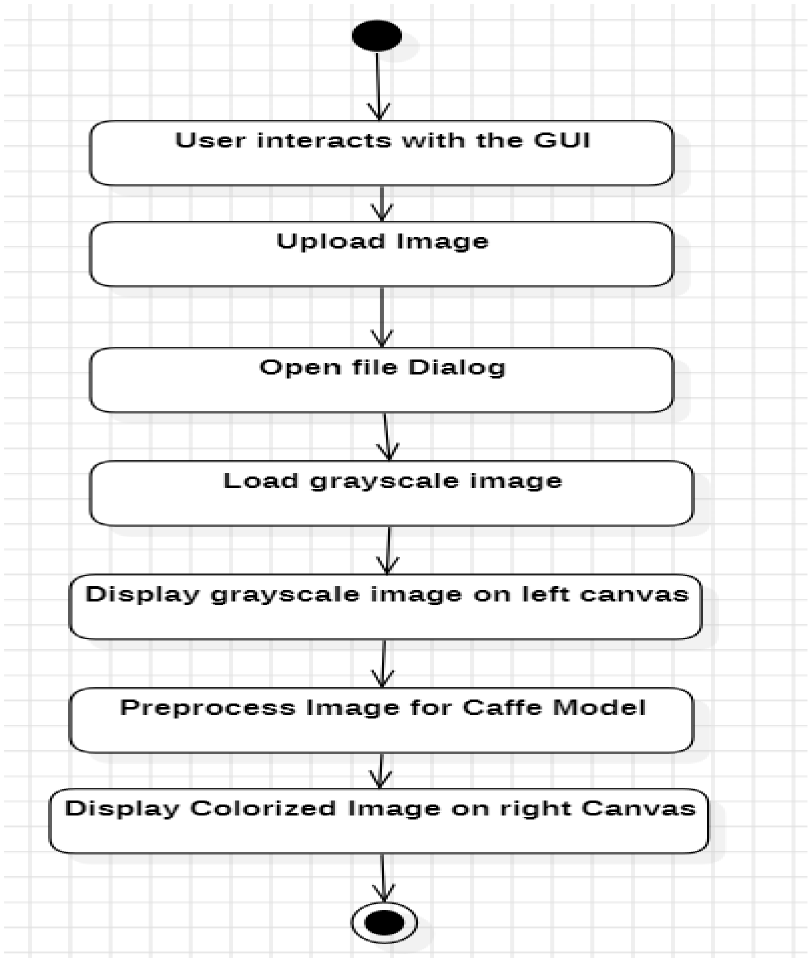

User interacts with the GUI
Upload Image
Open file Dialog
Load grayscale image
Display grayscale image on left canvas
Preprocess Image for Caffe Model
Display Colorized Image on right Canvas

Fig:4.5 Activity Diagram

CHAPTER-5

IMPLEMENTATION

5.1 Code for Image Colorization

```
import tkinter as tk
from tkinter import *
from tkinter import filedialog
from PIL import Image, ImageTk
import numpy as np
import cv2 as cv

# Load the neural network
caffe_net=cv.dnn.readNetFromCaffe("colorization_deploy_v2.prototxt","colorization_r
elease_v2.caffemodel")
pts_in_hull = np.load('pts_in_hull.npy')
pts_in_hull = pts_in_hull.transpose().reshape(2, 313, 1, 1)
caffe_net.getLayer(caffe_net.getLayerId('class8_ab')).blobs =
[pts_in_hull.astype(np.float32)]
caffe_net.getLayer(caffe_net.getLayerId('conv8_313_rh')).blobs = [np.full([1, 313],
2.606, np.float32)]

class Window(Frame):
    def _init_(self, master=None):
        super()._init_(master)
        self.master = master
        self.master.title("Image Colorization")
        self.pack(fill=BOTH, expand=1)

        # Create a canvas to display images side by side
        self.canvas = Canvas(self, width=960, height=360)  # 480x360 for each image
        self.canvas.pack()

        # Add a menu to upload the image
        self.menu = Menu(self.master)
        self.master.config(menu=self.menu)
        file_menu = Menu(self.menu)
        file_menu.add_command(label="Open", command=self.open_image)
        self.menu.add_cascade(label="File", menu=file_menu)

        self.original_image = None  # Placeholder for original grayscale image
        self.colorized_image = None  # Placeholder for colorized image
    def open_image(self):
        file_path = filedialog.askopenfilename()
        if file_path:
            # Open the image in grayscale
            pil_image = Image.open(file_path).convert('L').resize((480, 360))
            self.original_image = ImageTk.PhotoImage(pil_image)

            # Display the original grayscale image on the left side of the canvas
            self.canvas.create_image(0, 0, image=self.original_image, anchor=NW)

            # Process the image to get the colorized version
            color_image = self.process_and_color(pil_image)

            # Convert the colorized image to an ImageTk object for Tkinter display
            self.colorized_image = ImageTk.PhotoImage(color_image)

            # Display the colorized image on the right side of the canvas
            self.canvas.create_image(480, 0, image=self.colorized_image, anchor=NW)

    def process_and_color(self, pil_image):
        # Convert PIL Image (grayscale) to OpenCV format and replicate channels to
make it 3-channel (BGR)
        open_cv_image = np.array(pil_image.convert('RGB'))[:, :, ::-1]  # Convert RGB to
BGR for OpenCV
  # Preprocess the image for the Caffe model
        scaled = open_cv_image.astype('float32') / 255.0  # Scale to [0,1]
        lab_image = cv.cvtColor(scaled, cv.COLOR_BGR2Lab)  # Convert to Lab color
space
        l_channel = lab_image[:, :, 0]  # Extract the L channel

        # Resize the lightness channel to network input size
        resized_l_channel = cv.resize(l_channel, (224, 224))
        resized_l_channel -= 50  # Subtract the mean value

        # Prepare the model input
        net_input = cv.dnn.blobFromImage(resized_l_channel)

        # Perform inference
        caffe_net.setInput(net_input)
        ab_channels = caffe_net.forward()[0, :, :, :].transpose((1, 2, 0))  # Predict A and B
channels

        # Resize the ab_channels to original image size
        ab_channels = cv.resize(ab_channels, (open_cv_image.shape[1],
open_cv_image.shape[0]))
```

```
        # Merge the L channel with ab_channels
        colorized_image = np.concatenate((l_channel[:, :, np.newaxis], ab_channels),
axis=2)
        colorized_image = cv.cvtColor(colorized_image, cv.COLOR_Lab2BGR)
        colorized_image = np.clip(colorized_image * 255, 0, 255).astype('uint8')

        # Convert back to PIL format to display in Tkinter
        return Image.fromarray(cv.cvtColor(colorized_image, cv.COLOR_BGR2RGB))

root = tk.Tk()
root.geometry("960x360")  # Set window size large enough for both images
app = Window(root)
root.mainloop()
```

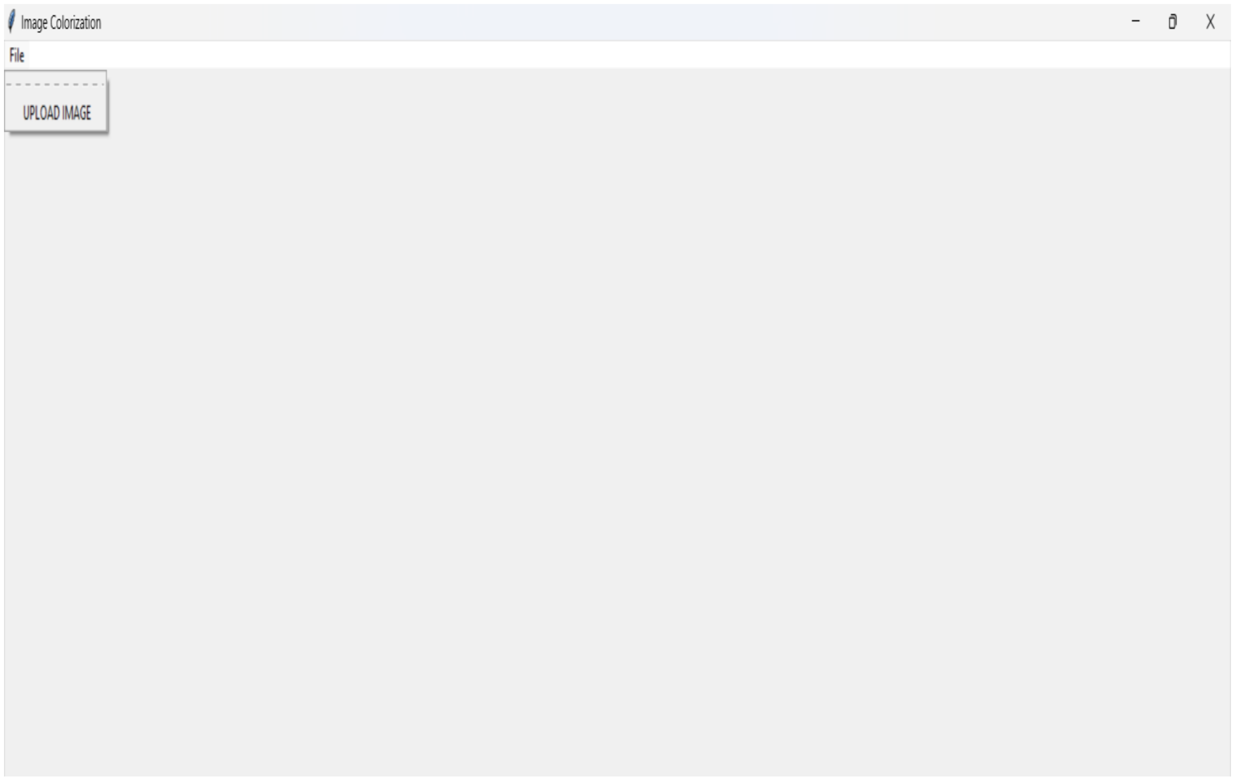
5.2 CODE FOR VIDEO COLORIZATION:

Git Clone and installing Deoldify in our device.

```
!git clone https://github.com/jantic/DeOldify.git DeOldify
#Importing the required paths as it should be called first
from deoldify import device
from deoldify.device_id import DeviceId
#choices: CPU, GPU0...GPU7
device.set(device=DeviceId.GPU0)
import torch
if not torch.cuda.is_available():
print('GPU not available.')
from os import path
#Now install the required packages.
!pip install -r requirements-colab.txt
#Again importing the required modules.
import fastai
from deoldify.visualize import *
from pathlib import Path
torch.backends.cudnn.benchmark=True
import warnings
warnings.filterwarnings("ignore", category=UserWarning,
message=".?Your .? set is empty.*?")
!mkdir 'models'
!wget https://data.deepai.org/deoldify/ColorizeVideo_gen.pth -O
./models/ColorizeVideo_gen.pth
colorizer = get_video_colorizer()27
#Colorizing the video and getting the output.
source_url = 'https://www.youtube.com/watch?v=qoMI8OdajYY' #@param
{type:"string"}
```

```
render_factor = 21 #@param {type: "slider", min: 5, max: 40}
watermarked = True #@param {type:"boolean"}
if source_url is not None and source_url !='':
        video_path = colorizer.colorize_from_url(source_url, 'video.mp4', render_factor,
watermarked=watermarked)
        show_video_in_notebook(video_path)
else:
        print('Provide a video url and try again.')
        for i in range(10,40,2):
                colorizer.vis.plot_transformed_image('video/bwframes/video/00001.jpg',
render_factor=i, display_render_factor=True, figsize=(8,8))
```

5.3 Output



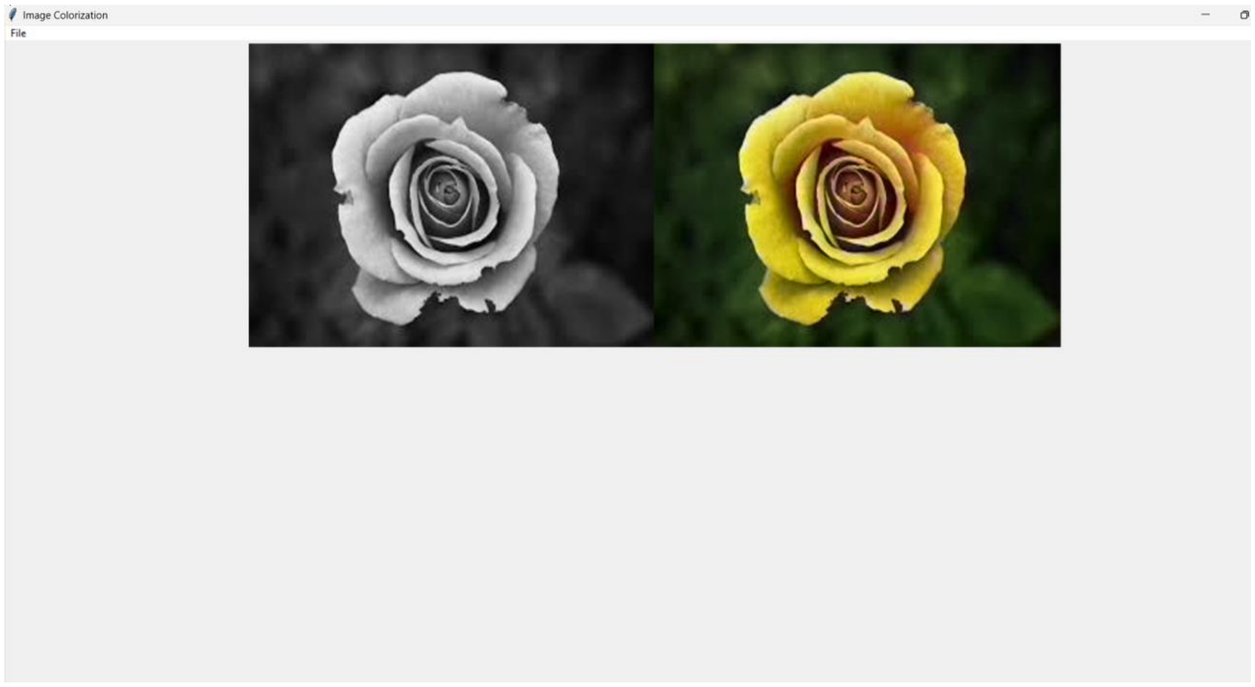


Image Colorization
File

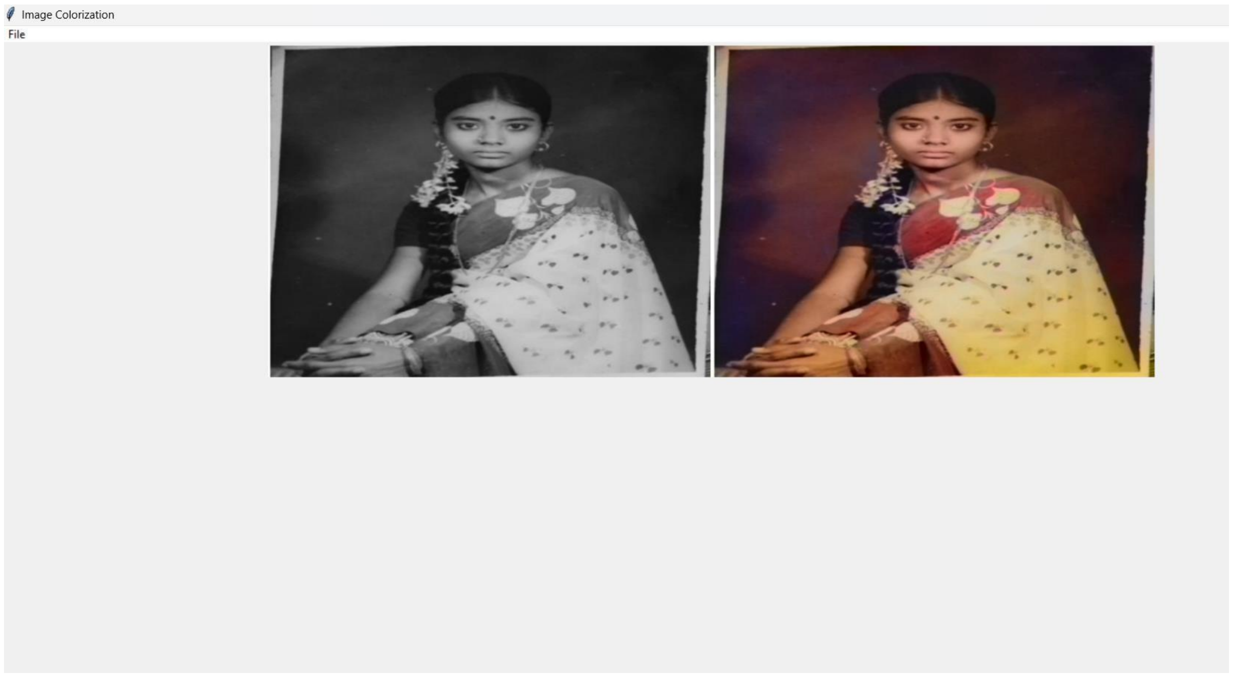


Image Colorization
File

1.00
0:38 / 2:07

CHAPTER-6
CONCLUSION & FUTURE SCOPE

6.1 Conclusion

In conclusion, image colorization using autoencoder is a promising approach for generating colorized images from grayscale images. The project involved the implementation of an autoencoder-based model for image colorization and testing its performance using various metrics. The results showed that the system was able to generate colorized images with high accuracy and quality, as measured by metrics such as MSE, SSIM, PSNR, and MOS. The system's performance was also efficient, able to generate colorized images quickly and handle a large number of input images.

The project's success indicates that autoencoder-based image colorization has great potential in various fields, such as image processing, computer vision, and multimedia applications. The project's methodology and results can be used as a foundation for further research and development in this area.

In conclusion, the implementation of the autoencoder-based image colorization model demonstrated that it is a viable and effective solution for generating colorized images from grayscale images. With further research and development, this approach can have significant applications and impact in various fields**.**

6.2 Future Scope

There are several opportunities for future work and improvements on the image colorization using autoencoder project.

One potential area of improvement is the incorporation of more complex neural network architectures, such as GANs or CNNs, to further improve the quality of the generated colorized images. Additionally, further research could explore the use of other loss functions or optimization algorithms to improve the overall performance of the system.

Another potential area of research is the development of more specialized and for medical imaging or satellite imagery. This would involve adapting the autoencoder architecture to better handle the unique characteristics and challenges of such images.

Moreover, the proposed model can be extended to handle videos or image sequences, which could be useful in applications such as video colorization or motion tracking.

REFERENCES

[1] V.Bochko, P. Valisuc, T. Alho, S. Sutlnen, J. Parkkinen, and J. Alander, “Medical image colorization using learning”, pp. 70–74, Jan. 2010.

[2] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition”, Proceedings of the IEEE, vol. 86, no. 11, pp. 2278–2324,Nov. 1998.

[3] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deepconvolutional neural networks”, in Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1, ser. NIPS’12, Lake Tahoe,Nevada: Curran Associates Inc., 2012, pp. 1097–1105.

[4] K. Chellapilla, S. Puri, and P. Simard, “High Performance Convolutional Neural Networks for Document Processing”, in Tenth International Workshop on Frontiers in Handwriting Recognition, G. Lorette, Ed., Universit´e de Rennes 1, La Baule(France): Suvisoft, Oct. 2006.

[5] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin, “Convolutional sequence to sequence learning”, CoRR, vol. abs/1705.03122, 2017.

[6] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition”,CoRR, vol. abs/1512.03385, 2015.

[7] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1”, in Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1, D. E. Rumelhart, J. L. McClelland, and C. PDP Research Group, Eds., Cambridge, MA, USA: MIT Press, 1986, ch. Learning Internal Representations by Error Propagation, pp. 318–362, isbn: 0-262-68053-X.

[8] S. Iizuka, E. Simo-Serra, and H. Ishikawa, “Let there be color!: Joint end-to-end learning of global and local image priors for automatic image colorization with simultaneous classification”, ACM Trans. Graph., vol. 35, no. 4, 110:1–110:11, Jul.2016, issn: 0730-0301.

[9] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Van-houcke, and A. Rabinovich, “Going deeper with convolutions”, CoRR, vol. abs/1409.4842,2014.

[10] F. Yu and V. Koltun, “Multi-scale context aggregation by dilated convolutions”,CoRR, vol. abs/1511.07122, 2015.